FUJITSU SEMICONDUCTOR
CONTROLLER MANUAL

F²MC-16LX
16-BIT MICROCONTROLLER
MB90580 SERIES
HARDWARE MANUAL

FUJITSU

PREFACE

Thank you for selecting FUJITSU Semiconductor Devices.

The FUJITSU MB90580 series has been developed as one general-application version of the $F^2MC^{®}$*16LX series of original 16-bit one-chip microcontrollers for ASIC (application specific IC) applications. This manual describes the functions and operations of the MB90580 series, and is intended for use by engineers actually designing products using these semiconductors. Please be sure to read it carefully.

*: F^2MC is an abbreviation for FUJITSU Flexible Microcontroller, and is a registered trademark of Fujitsu.

This document is organized as follows.

Chapter 1 OVERVIEW

This section presents an overview of MB90580 series features and functions.

Chapter 2 CPU

This section describes the functions of the F^2MC-16LX series CPU.

Chapter 3 MEMORY

This section describes the functions of the F^2MC-16LX series memory.

Chapter 4 CLOCK AND RESET

This section describes the functions of the MB90580 series clocks and resets.

Chapter 5 WATCHDOG TIMER, TIME BASE TIMER, AND WATCH TIMER FUNCTION

This section describes the functions and operation of the MB90580 series watchdog timer, timebase timer and watch timer function.

Chapter 6 LOW POWER CONTROL CIRCUIT

This section describes the MB90580 series low power control circuits (CPU intermittent operation function, oscillator stabilization wait time, PLL clock multiplier function).

Chapter 7 INTERRUPT

This section describes the functions of each MB90580 each interrupt and interrupt source.

Chapter 8 PARALLEL PORTS

This section describes the functions of the MB90580 series parallel port.

Chapter 9 DTP/EXTERNAL INTERRUPT

This section describes the function and operation of the MB90580 series DTP and external interrupts.

Chapter 10 DELAY INTERRUPT MODULE

This section describes the functions and operation of the MB90580 series delay interrupt module.

Chapter 11

Chapter 12 COMMUNICATION PRESCALER

This section describes the MB90580 series communication prescaler.

Chapter 13 UART

This section describes the function and operation of the MB90580 UART.

Chapter 14 IE BUS

This section describes the functions and operation of the MB90580 series IE Bus.

Chapter 15 8/16-BIT PPG

This section describes the functions and operation of the MB90580 series 8/16-bit PPG.

Chapter 16 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)

This section describes the functions and operation of the MB90580 series 16-bit reload timer.

Chapter 17 A/D CONVERTERr

This section describes the functions and oeration of the MB90580 series A/D converter.

Chapter 18 D/A CONVERTER

This section describes the functions and oeration of the MB90580 series D/A converter

Chapter 19 PULSE WIDTH COUNTER (PWC) TIMER

This section describes the functions and oeration of the MB90580 series pulse width counter (PWC) timer.

Chapter 20 CLOCK MONITOR FUNCTION

This section describes the functions of the MB90580 series clock monitor function.

Chapter 21 16-bit I/O Timers

This section describes the functions and operation of the MB90580 series 16-bit I/O timers which consists of 16-bit free-run timer, 2 output compare regsiters and 4 input capture registers.

Chapter 22 ROM CORRECTION module

This section describes the function and operation of the MB90580 series rom correction module.

Chapter 23 ROM MIRRORING MODULE

This section describes the function of the MB90580 series ROM mirrorling module.

Appendix A I/O MAP

The appendix A provides I/O maps, and low power mode status transition charts.

Appendix B INSTRUCTION

The appendix B describes addressing in the $F^2MC^{®}$*-16LX series, and provides instruction lists and instruction maps.

Appendix C PROGRAMMING THE FLASH MEMORY ON THE MB90F584

The appendix C provides programming method of the flash memory on the MB90F584.

CONTENTS

FIGURES

TABLES

Chapter 1: Overview

The MB90580 series 16-bit microcontrollers are designed for applications that require high-speed real-time processing. These microcontrollers feature functions that are suitable for controlling car audio and electronic appliances.

1.1 Features

• Clock

Embedded PLL Clock Multiplication Circuit
Operating clock (PLL clock) can e selected from divided-by-2 of oscillation or one to four times the oscillation (at oscillation of 4 MHz, 4 MHz to 16 MHz).
Minimum instruction execution time of 83.3ns (at oscillation of 4 MHz, three times the PLL clock, operation at Vcc of 5.0 V)

• CPU addressing space of 16 Mbytes

Internal addressing of 24-bit
External accessing can be performed by selecting 8/16-bit bus width (external bus mode)

• Instruction set optimized for controller applications

Rich data types (bit, byte, word, long word)
Rich addressing mode (23 types)
High code efficiency
Enhanced precision calculation realized by the 32-bit accumulatorInstruction set designed for high level language (C) and multi-task operations
Adoption of system stack pointer
Enhanced pointer indirect instructions
Barrel shift instructions

• Enhanced execution speed

4-byte instruction queue

• Enhanced interrupt function

8 levels, 32 factors

• Automatic data transmission function independent of CPU operation

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

• Low-power consumption (stand-by) mode

Sleep mode (mode in which CPU operating clock is stopped)
Timebase timer mode (mode in which other than oscillation and timebase timer are stopped)
Stop mode (mode in which oscillation is stopped)
CPU intermittent operation mode
Hardware stand-by mode

• I/O port

Maximum of 77 ports

• IE Bus :1 channels

small scale two-line serial bus interface for automotive and general industrial application
Maximium transfer rate is 27 Kbps

- Timers

  18-bit Timebase counter/watchdog timer: 1 channel
  Watch-dog timer : 1 channel
  15-bit Watch timer : 1 channel
  8/16-bit PPG timer: 8-bit × 2 channels or 16-bit × 1 channel
  16-bit re-load timer: 3 channels
  16-bit PWC timer (with noise filter) : 1 channel
  16-bit I/O timer (16-bit free-run timer): 1 channel

- Input capture (ICU) : 4 channels

  Generates an interrupt request by latching a 16-bit free-run timer counter value upon detection of an edge input to the pin.

- Output compare (OCU) : 2 channels

  Generates an interrupt request and reverse the output level upon detection of a match between the 16-bit free-run timer counter value and the compare setting value.

- UART : 5 channels

  With full-duplex double buffer (8-bit length)
  Clock asynchronized or clock synchronized transmission (with start and stop bits) can be selectively used.

- TP/external interrupt circuit : 8 channels

  A module for starting extended intelligent I/O service (EI^2OS) and generating an external interrupt triggered by an external input.

- Delayed interrupt generation module

  Generates an interrupt request for switching tasks.

- Clock monitor function

  Output the clock to I/O port (Dividing the machine clock by 2 to 2^8.

- ROM correction module

  Replace the internal ROM code by small external circuit.

- ROM mirroring module

  Used to increase the coding efficiency.

- 10-bit A/D converter : 8 channels

  10-bit resolution can be selectively used.
  Starting by an external trigger input.

- 8-bit D/A converter : 2 independent channels

  8-bit resolution.
  R-2R typet.

- Package

  LQFP-100, QFP-100

- Process

  CMOS technology

1.2 Product Lineup

Internal Configuration

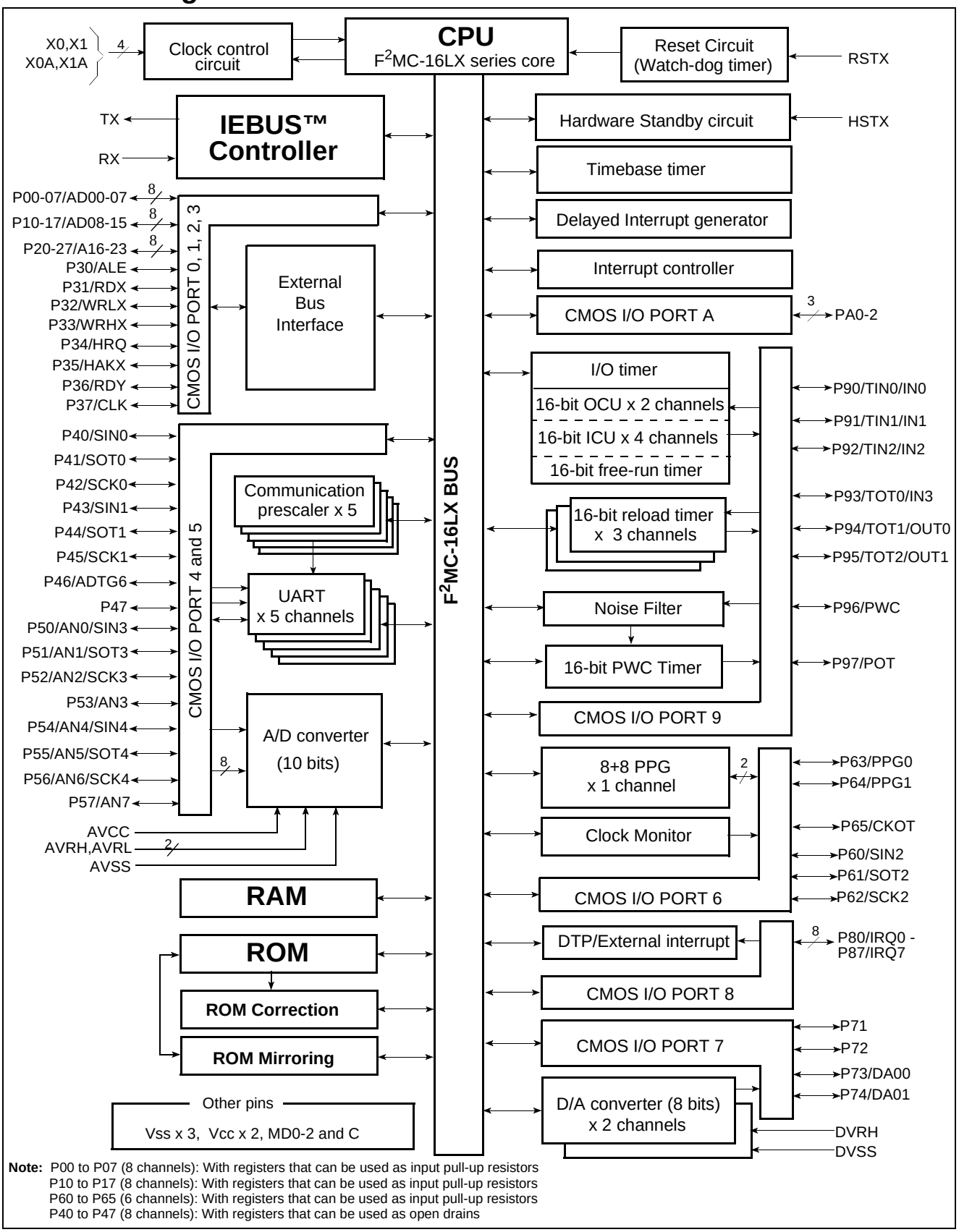
Table 1.2a lists the product lineup of the MB90580 series. All products are functionally identical except for ROM and RAM sizes.

Table 1.2a MB90580 series product lineup

	MB90V580	MB90583	MB90F583
ROM size	_____	Mask ROM 128 Kbytes	Flash ROM 128 Kbytes
RAM size	6kByte	6kByte	6kByte
Others			

Note: MB90V580 is the evaluation device of MB90580 series, that has no internal ROM incorporated. However it has 6Kbytes of internal RAM and the internal resources. The package of MB90V580 is PGA-256C-A02.

1.3 Block Diagram



Note: P00 to P07 (8 channels): With registers that can be used as input pull-up resistors
P10 to P17 (8 channels): With registers that can be used as input pull-up resistors
P60 to P65 (6 channels): With registers that can be used as input pull-up resistors
P40 to P47 (8 channels): With registers that can be used as open drains

Figure 1.3a Block Diagram of MB90580 Series

1.4 Pin Assignment

1.4.1 LQFP-100 Pin Assignment

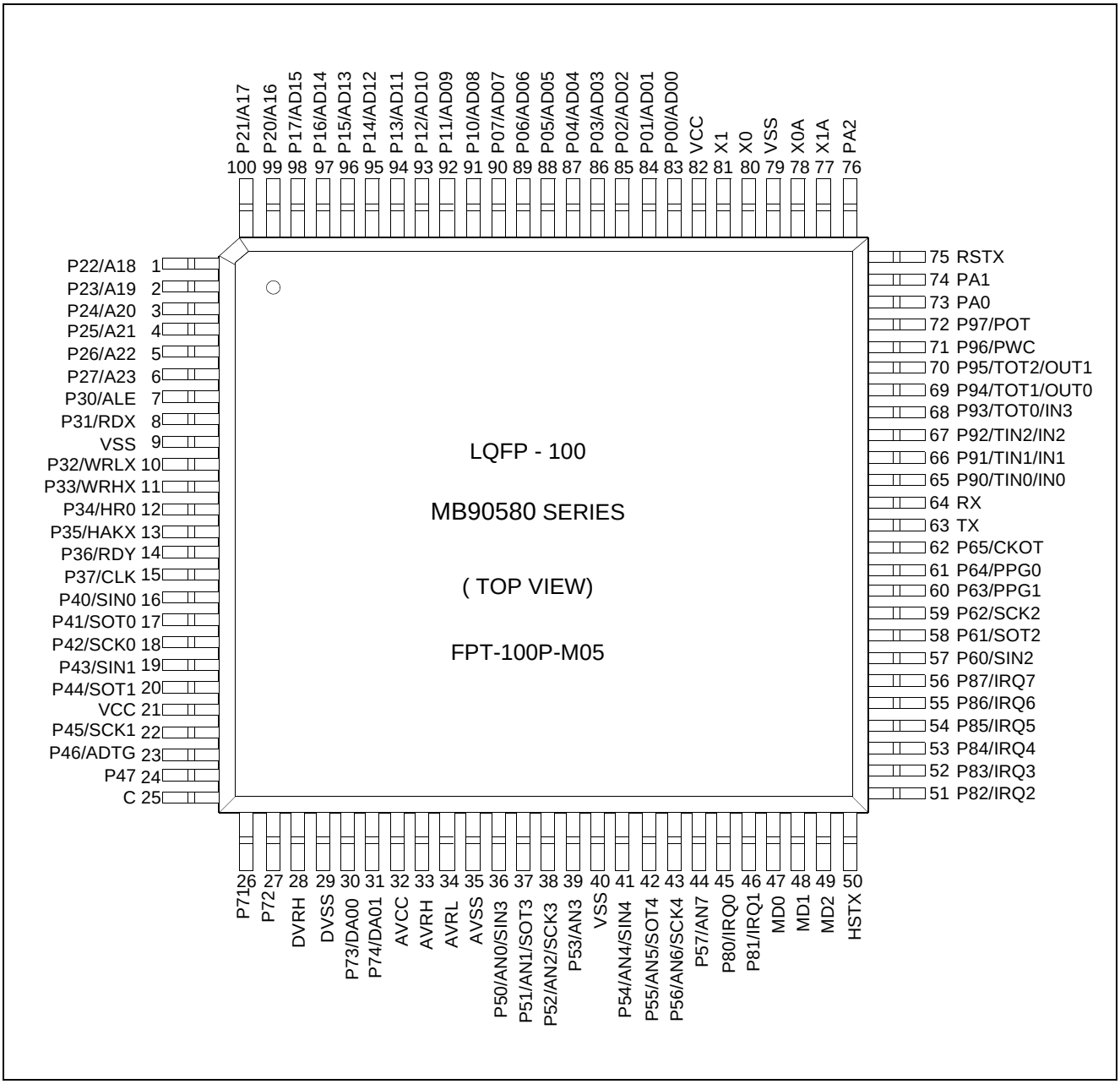


Figure 1.4a Pin Assignment of MB90580 (LQFP-100)

1.4.1 QFP-100 Pin Assignment

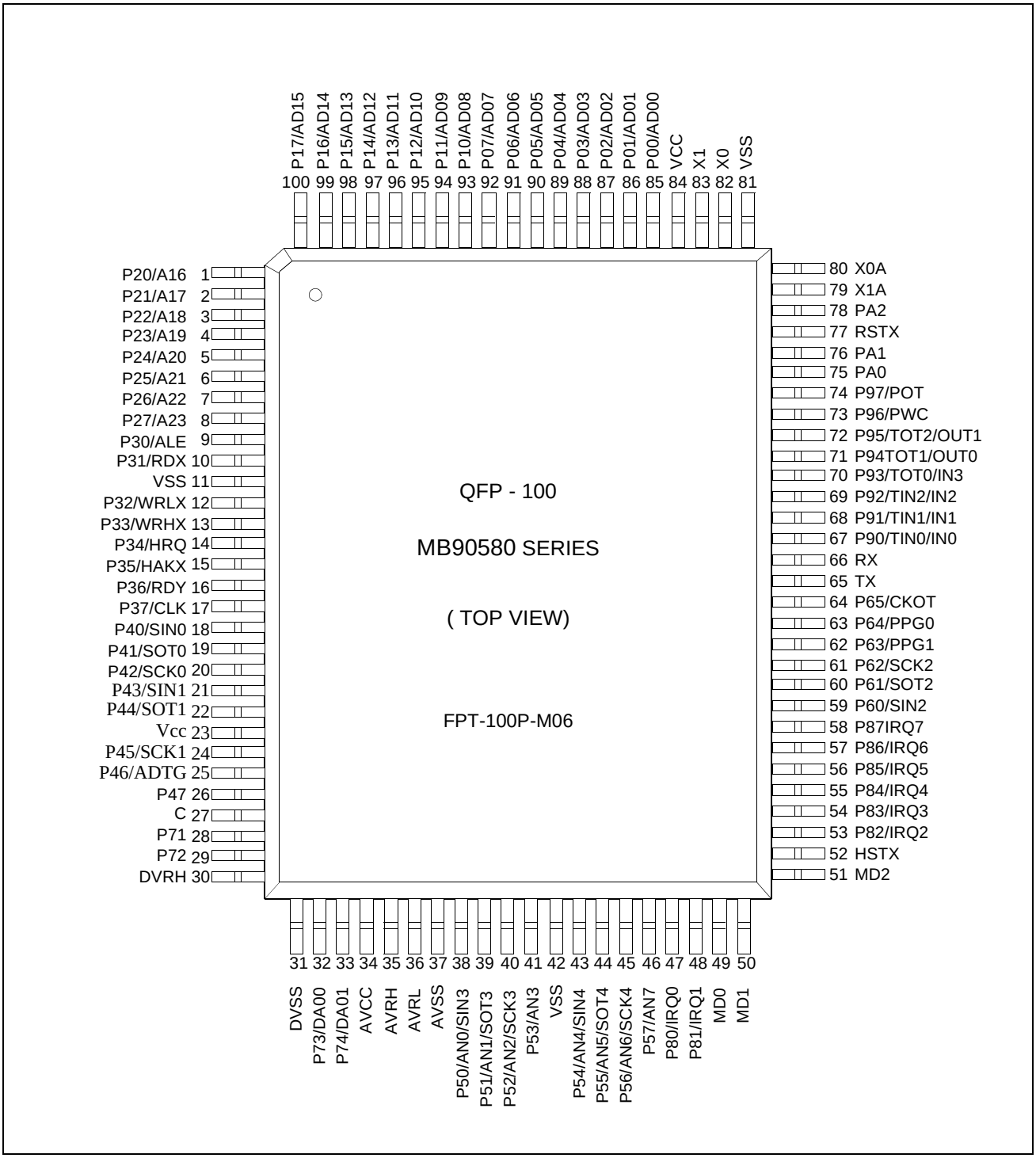


Figure 1.4b Pin Assignment of MB90580 (QFP-100)

1.5 Pin Functions

Table 1.5a to Table 1.5d lists the functions. Table 1.5e to Table 1.5g list the I/O circuit formats.

Table 1.5a Pin functions (1/4) (STBC: With standby control)

QFP	LQFP	Pin name	I/O Circuit	Function
82	80	X0	A	Oscillator pin
83	81	X1	A	Oscillator pin
52	50	HSTX	C	Hardware standby input pin
77	75	RSTX	B	Reset input pin
85 to 92	83 to 90	P00 to P07	D (CMOS/H)	General-purpose I/O ports A pull-up resistor can be assigned (RD07-00='1') by using the pull-up resistor setting register (RDR0). (D07-00='1': Invalid when set as output)
		AD00 to AD07		Low-order data I/O or low-order address output (AD00 to 07) in external bus mode
93 to 100	91 to 98	P10 to P17	D (CMOS/H)	General-purpose I/O ports A pull-up resistor can be assigned (RD17-10='1') by using the pull-up resistor setting register (RDR1). (D17-10='1': Invalid when set as output)
		AD08 to AD15		High-order data I/O or medium-order address output (AD08 to 15) in external 16-bit bus mode
1 to 8	99 to 6	P20 to P27	F (CMOS/H)	General-purpose I/O ports Pins A16 to A19 when the corresponding bit of the HACR register is '0' in external bus mode
		A16 to A23		High-order address output (A16 to A19) when the corresponding bit of the HACR register is '1' in external bus mode
9	7	P30	F (CMOS/H)	General-purpose I/O port ALE pin in external bus mode
		ALE		Address fetch enable signal pin
10	8	P31	F (CMOS/H)	General-purpose I/O port RDX pin in external bus mode
		RDX		Read strobe output (RDX) pin
12	10	P32	F (CMOS/H)	General-purpose I/O port WRX pin when the WRE bit is '1' in external bus mode
		WRLX		Low-order data write strobe output (WRLX) pin
13	11	P33	F (CMOS/H)	General-purpose I/O port WRHX pin when the WRE bit of the EPCR register is '1' in external 16-bit bus mode
		WRHX		High-order data write strobe output (WRHX) pin
14	12	P34	F (CMOS/H)	General-purpose I/O port HRQ pin when the HDE bit of the EPCR register is '1' in external bus mode
		HRQ		Hold request input (HRQ) pin
15	13	P35	F (CMOS/H)	General-purpose I/O port HAKX pin when the HDE bit of the EPCR register is '1' in external bus mode
		HAKX		Hold acknowledgment output (HAKX) pin
16	14	P36	F (CMOS/H)	General-purpose I/O port RDY pin when the RYE bit of the EPCR register is '1' in external bus mode
		RDY		External ready input (RDY) pin
17	15	P37	F (CMOS/H)	General-purpose I/O port CLK pin when the CKE bit of the EPCR register is '1' in external bus mode
		CLK		Machine cycle clock output (CLK) pin
18	16	P40	E (CMOS/H)	General-purpose I/O port Serial input (SIN0) during UART0 operation Open drain output port when OD40 of the open drain control setting register (ODR4) is set to '1' (D40='0': Invalid when set as input)
		SINO		UART0 serial data input (SIN0) pin

Table 1.5b Pin functions (2/4)

QFP	LQFP	Pin name	I/O Circuit	Function
19	17	P41	E (CMOS/H)	General-purpose I/O port SOT0 pin when the SOE bit of the UMC register is '1' Open drain output port when OD41 of the open drain control setting register (ODR4) is set to '1' (D41='0': Invalid when set as input)
		SOT0		UART0 serial data output (SOT0) pin
20	18	P42	E (CMOS/H)	General-purpose I/O port SOT0 pin when the SOE bit of the UMC register is '1' Open drain output port when OD41 of the open drain control setting register (ODR4) is set to '1' (D41='0': Invalid when set as input)
		SCK0		UART0 serial clock I/O (SCK0) pin
21	19	P43	E (CMOS/H)	General-purpose I/O port Serial input (SIN1) during extended I/O serial operation Open drain output port when OD43 of the open drain control setting register (ODR4) is set to '1' (D43='0': Invalid when set as input)
		SIN1		UART1 serial data input (SIN1) pin
22	20	P44	E (CMOS/H)	General-purpose I/O port SOT1 pin when the SOE bit of the UMC register is '1' Open drain output port when OD44 of the open drain control setting register (ODR4) is set to '1' (D44='0': Invalid when set as input)
		SOT1		UART1 serial data output (SOT1) pin
24	22	P45	E (CMOS/H)	General-purpose I/O port Clock input (SCK1) during extended I/O serial operation in external shift clock mode SCK1 pin when the SOE bit of the UMC register is '1' Open drain output port when OD45 of the open drain control setting register (ODR4) is set to '1' (D45='0': Invalid when set as input)
		SCK1		UART1 serial clock I/O (SCK1) pin
25	23	P46	E (CMOS/H)	General-purpose I/O port Open drain output port when OD46 of the open drain control setting register (ODR4) is set to '1' (D46='0': Invalid when set as input)
		ADTG		A/D converter external trigger input pin
26	24	P47	E (CMOS/H)	General-purpose I/O port Open drain output port when OD47 of the open drain control setting register (ODR4) is set to '1' (D47='0': Invalid when set as input)
38	36	P50	G (CMOS/H)	General-purpose I/O port
		AN0		Analog input pin (AN0) during A/D converter operation
		SIN3		UART3 serial data input (SIN3) pin
39	37	P51	G (CMOS/H)	General-purpose I/O port
		AN1		Analog input pin (AN1) during A/D converter operation
		SOT3		UART3 serial data output (SOT3) pin
40	38	P52	G (CMOS/H)	General-purpose I/O port
		AN2		Analog input pin (AN2) during A/D converter operation
		SCK3		UART3 serial data output (SOT3) pin
41	39	P53	G (CMOS/H)	General-purpose I/O port
		AN3		Analog input pins (AN3) during A/D converter operation
43	41	P54	G (CMOS/H)	General-purpose I/O port
		AN4		Analog input pin (AN4) during A/D converter operation
		SIN4		UART4 serial data input (SIN4) pin
44	42	P55	G (CMOS/H)	General-purpose I/O port
		AN5		Analog input pin (AN5) during A/D converter operation
		SOT4		UART4 serial data output (SOT4) pin

Table 1.5c Pin functions (3/4)

QFP	LQFP	Pin name	I/O Circuit	Function
45	43	P56	G (CMOS/H)	General-purpose I/O port
		AN6		Analog input pin (AN6) during A/D converter operation
		SCK4		UART4 serial data output (SOT4) pin
46	44	P57	G (CMOS/H)	General-purpose I/O port
		AN7		Analog input pins (AN7) during A/D converter operation
27	25	C		0.1uf capacitor connection pin for voltage supply stabilization.
28	26	P71	F (CMOS/H)	General-purpose I/O port SOT3 pin when the SOE bit of the UMC register is '1'
29	27	P72	F (CMOS/H)	General-purpose I/O port Clock input (SCK3) during UART1 operation in external shift clock mode SCK3 pin when the SOE bit of the UMC register is '1'
32	30	P73	H (CMOS/H)	General-purpose I/O port D/A output pin when the DAE0 bit of the D/A control register (DACR) is '1'
		DAO0		D/A output '0' pin during D/A converter operation
33	31	P74	F (CMOS/H)	General-purpose I/O port D/A output pin when the DAE1 bit of the D/A control register (DACR) is '1'
		DAO1		D/A output '1' pin during D/A converter operation
47	45	P80	F (CMOS/H)	General-purpose I/O port
		IRQ0		External interrupt request I/O 0
48	46	P81	F (CMOS/H)	General-purpose I/O port
		IRQ1		External interrupt request I/O 1
53	51	P82	F (CMOS/H)	General-purpose I/O port
		IRQ2		External interrupt request I/O 2
54	52	P83	F (CMOS/H)	General-purpose I/O port
		IRQ3		External interrupt request I/O 3
55	53	P84	F (CMOS/H)	General-purpose I/O port
		IRQ4		External interrupt request I/O 4
56	54	P85	F (CMOS/H)	General-purpose I/O port
		IRQ5		External interrupt request I/O 5
57	55	P86	F (CMOS/H)	General-purpose I/O port Always enabled (STBC)
		IRQ6		External interrupt request I/O 6
58	56	P87	F (CMOS/H)	General-purpose I/O port Always enabled (STBC)
		IRQ7		External interrupt request I/O 7
59	57	P60	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned (RD60='1') by using the pull-up resistor setting register (RDR6). (D60='1': Invalid when set as output)
		SIN2		UART2 serial data input (SIN2) pin
60	58	P61	D (CMOS/H)	General-purpose I/O port SOT1 pin when the SOE bit of the UMC register is '1' A pull-up resistor can be assigned (RD61='1') by using the pull-up resistor setting register (RDR6). (D61='1': Invalid when set as output)
		SOT2		UART2 serial data output (SOUT2) pin
61	59	P62	D (CMOS/H)	General-purpose I/O port Clock input (SCK2) during UART1 operation in external shift clock mode SCK1 pin when the SOE bit of the UMC register is '1' A pull-up resistor can be assigned (RD62='1') by using the pull-up resistor setting register (RDR6). (D62='1': Invalid when set as output)
		SCK2		UART2 serial clock I/O (SCK2) pin

Table 1.5d Pin functions (4/4)

QFP	LQFP	Pin name	I/O Circuit	Function
62	60	P63	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned (RD63='1') by using the pull-up resistor setting register (RDR6). (D63='1': Invalid when set as output)
		PPG00		PPG00 output when PPG is enabled
63	61	P64	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned (RD64='1') by using the pull-up resistor setting register (RDR6). (D64='1': Invalid when set as output)
		PPG01		PPG01 output when PPG is enabled
64	62	P65	D (CMOS/H)	General-purpose I/O port A pull-up resistor can be assigned (RD65='1') by using the pull-up resistor setting register (RDR6). (D65='1': Invalid when set as output)
		CKOT		CKOT output during CKOT operation
65	63	TX	I	IEBus output when IEBus is enabled
66	64	RX	J	IEBus input when IEBus is enabled
67 to 69	65 to 67	P90 to P92	F (CMOS/H)	General-purpose I/O port
		TIN0 to TIN2		Event input pins for reload timers 0,1 and 2. As these inputs are used continuously during reload timer input operation, outputs to these pins from other functions must be avoided unless performed intentionally.
		IN0 to IN2		Input capture channels 0 - 2 trigger inputs
70	68	P93	F (CMOS/H)	General-purpose I/O port
		TOT0		Output pins for reload timer 0. This function applies when the output for reload timers 0 is enabled.
		IN3		Input capture channel 3 trigger input
71 to 72	69 to 70	P94 to P95	F (CMOS/H)	General-purpose I/O port
		TOT1, TOT2		Output pins for reload timers 1 and 2. This function applies when the outputs for reload timers 1 and 2 are enabled.
		OUT0, OUT1		Output comparison channels 0 - 1 event outputs
73	71	P96	F (CMOS/H)	General-purpose I/O port
		PWC		PWC input
74	72	P97	F (CMOS/H)	General-purpose I/O port
75, 76	73, 74	PA0, PA1	F (CMOS/H)	General-purpose I/O port
78	76	PA2	F (CMOS/H)	General-purpose I/O port
79	77	X1A	A	Oscillator input
80	78	X0A	A	Oscillator input
34	32	AV_{CC}	—	A/D converter power supply pin
37	35	AV_{SS}	—	A/D converter power supply pin
35	33	AVRH	—	A/D converter external reference power supply pin
36	34	AVRL	—	A/D converter external reference power supply pin
30	28	DVRH	—	D/A converter external reference power supply pin
31	29	DVSS	—	D/A converter power supply pin
49 to 51	47 to 49	MD0 to MD2	C	Operation mode specification input pin Connect directly to Vcc or Vss.
23, 84	21, 82	V_{CC}	—	Power supply (5 V) input pin
11, 42, 81	9, 40, 79	VSS	—	Power supply (0 V) input pin

Table 1.5e I/O circuit format (1)

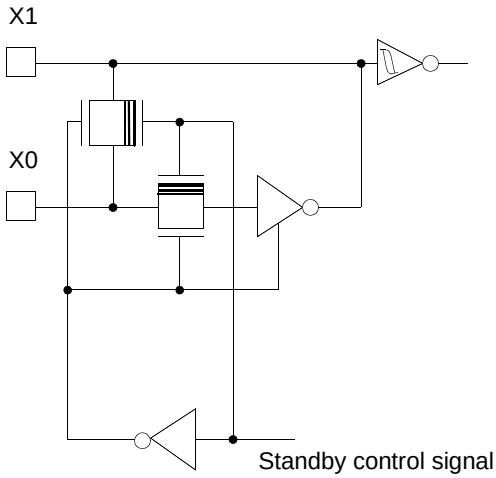
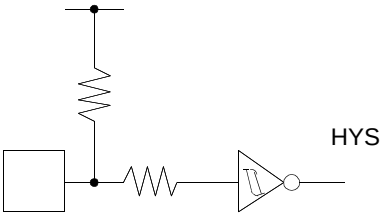
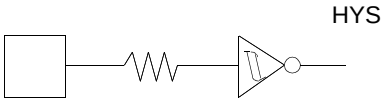
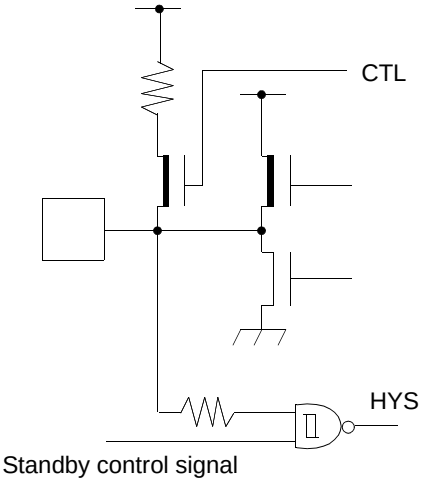
Class	Circuit	Remarks
A		• Oscillation feedback resistor: 1 MΩ approx.
B		• Hysteresis input with pull-up Resistor: 50 kΩ approx.
C		• Hysteresis input port
D		• With input pull-up resistor control • CMOS level output • Hysteresis input with standby control Resistor: 50 kΩ approx.

Table 1.5f I/O circuit format (2)

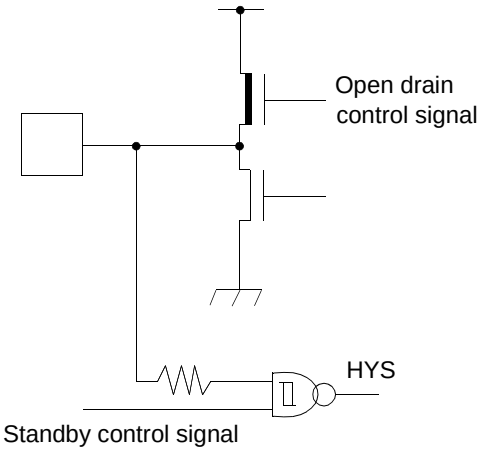
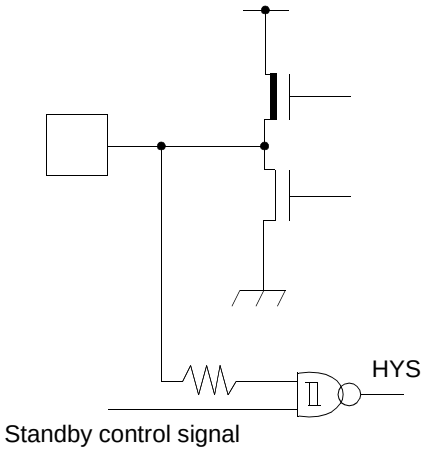
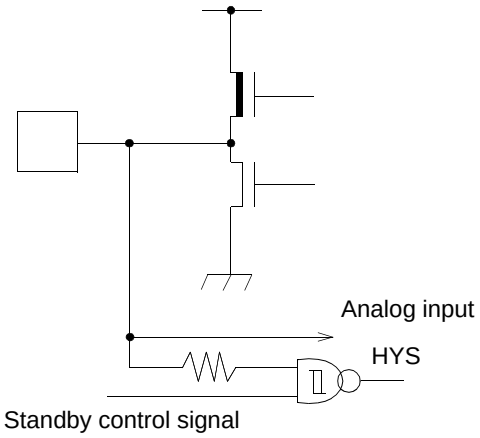
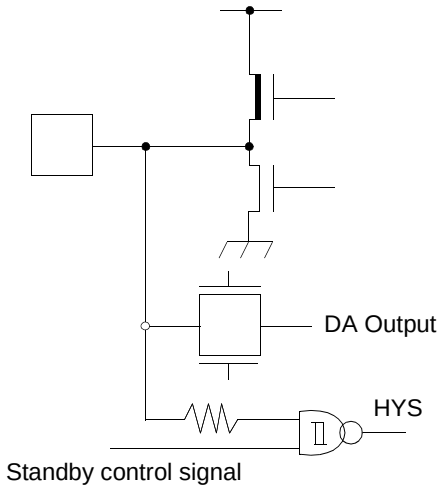
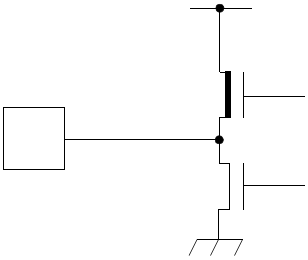
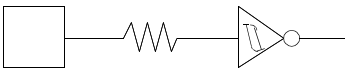
Class	Circuit	Remarks
E		• CMOS level output • With open drain control • Hysteresis input with standby contro
F		• CMOS level output • Hysteresis input with standby control
G		• CMOS level output • Hysteresis input with standby control • Analog input

Table 1.5g I/O circuit format (3)

Class	Circuit	Remarks
H		• CMOS level output • Hysteresis input with standby control • Analog output • Shared with DA output
I		• CMOS level output
J		• Hysteresis input

1.6 Handling the Device

(1) Preventing latch-up

CMOS IC chips may suffer latch-up under the following conditions:

A voltage higher than Vcc or lower than Vss is applied to an input or output pin.

A voltage higher than the rated voltage is applied between Vcc and Vss.

The AVcc power supply is applied before the Vcc voltage.

Latch-up may increase the power supply current drastically, causing thermal damage to the device.

(2) Handling unused input pins

Do not leave unused input pins open, as doing so may cause misoperation of the device. Use a pull-up or pull-down resistor.

(3) Using external clock

To use external clock, drive the X0 and X1 pins in reverse phase.

Figure 1.6a is a diagram of how to use external clock..

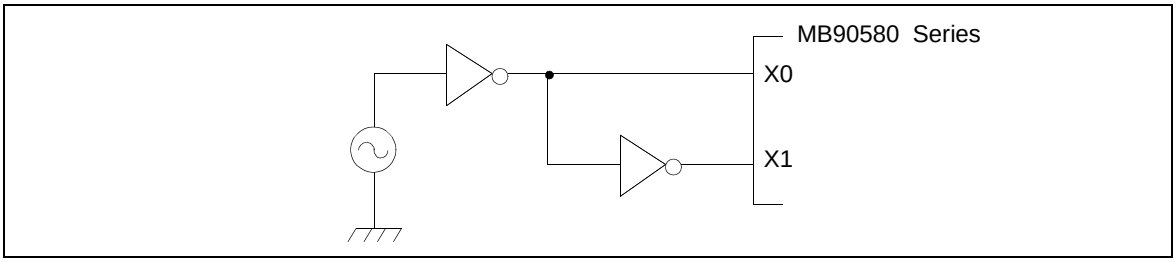


Figure 1.6a Using external clock

(4) Power supply pins (Vcc/Vss)

Ensure that all Vcc-level power supply pins are at the same potential. In addition, ensure the same for all Vss-level power supply pins. (See the figure below.) If there are more than one Vcc or Vss system, the device may operate incorrectly even within the guaranteed operating range.

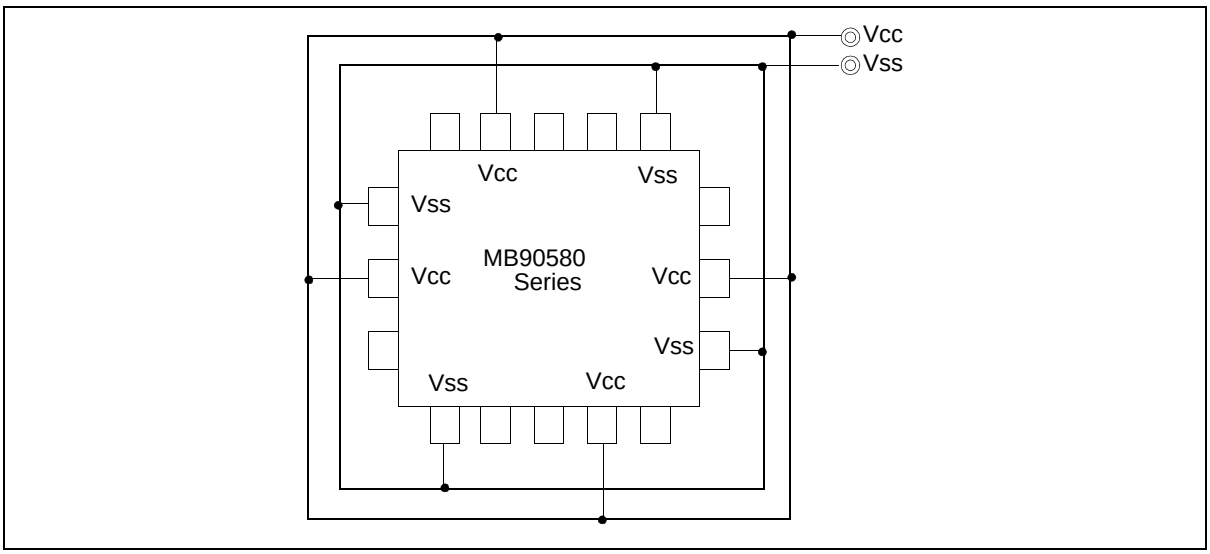


Figure 1.6b Connection of Power pins

Chapter 2: CPU

2.1 CPU

The F^{2}MC-16LX CPU core is a 16-bit CPU designed for applications that require high-speed real-time processing, such as home-use or vehicle-mounted electronic appliances. The F^{2}MC-16LX instruction set is designed for controller applications, and is capable of high-speed, highly efficient control processing.

In addition to 16-bit data, the F^{2}MC-16LX CPU core can process 32-bit data by using an internal 32-bit accumulator. (32-bit data can be processed with some instructions.) Up to 16 Mbytes of memory space (expandable) can be used, which can be accessed by either the linear pointer or bank method. The instruction system, based on the F^{2}MC-8 A-T architecture, has been reinforced by adding instructions compatible with high-level languages, expanding addressing modes, reinforcing multiplication and division instructions, and enhancing bit processing. The features of the F^{2}MC-16LX CPU are explained below.

- Minimum instruction execution time: 62.5 ns (at 4-MHz oscillation, 4 times multiplication)
- Maximum memory space: 16 Mbytes, accessed in linear or bank mode
- Instruction set optimized for controller applications
  Rich data types: Bit, byte, word, long word
  Extended addressing modes: 23 types
  High-precision operation (32-bit length) based on 32-bit accumulator
  Signed multiply and division, enhanced RETI instruction
- Powerful interrupt functions
  Eight priority levels (programmable)
- CPU-independent automatic transfer
  Up to 16 channels of extended intelligent I/O service
- Instruction set compatible with high-level language (C)/multitasking
  System stack pointer/instruction set symmetry/barrel-shift instructions
- Improved execution speed: 4-byte queue

2.1.1 Memory space

■ Outline of CPU memory space

An F^{2}MC-16LX CPU has a 16-Mbyte memory space. All data program input and output managed by the F^{2}MC-16LX CPU are located in this 16-Mbyte memory space. The CPU accesses the resources by indicating their addresses using a 24-bit address bus. (See Figure 2.1.1a.).

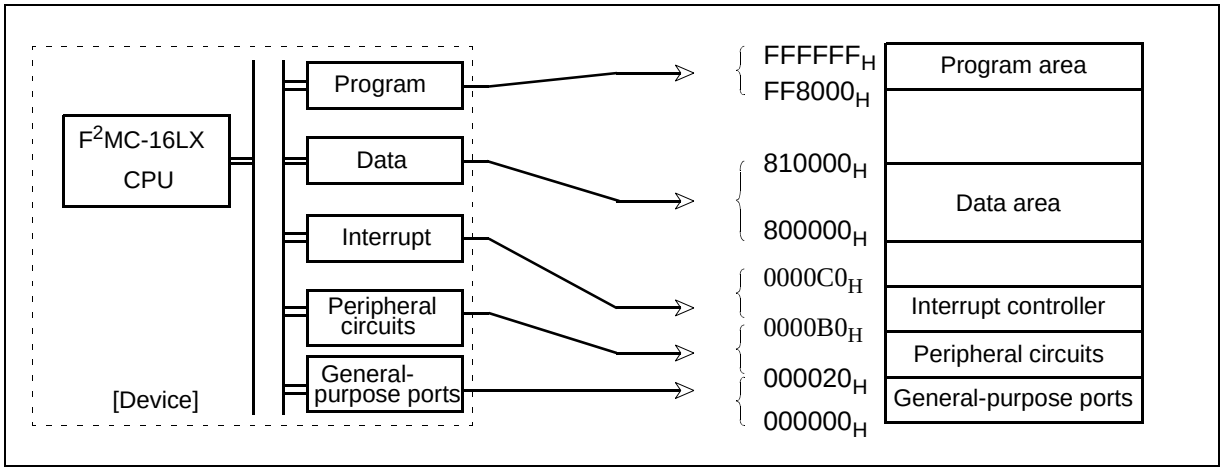


Figure 2.1.1a Sample relationship between F^{2}MC-16LX system and memory map

■ Address generation types

The F^{2}MC-16LX CPU has two address generation methods. One is the linear method in which an entire 24-bit address is specified by an instruction. The other method is the bank method in which the high-order eight bits of an address is specified by an appropriate bank register while the low-order 16 bits of the same address is specified by an instruction.

There are two types of linear method. One specifies a 24-bit address directly by using operands. The other method cites the low-order 24 bits of a 32-bit general-purpose register value as an address. (See Figure 2.1.1b.)

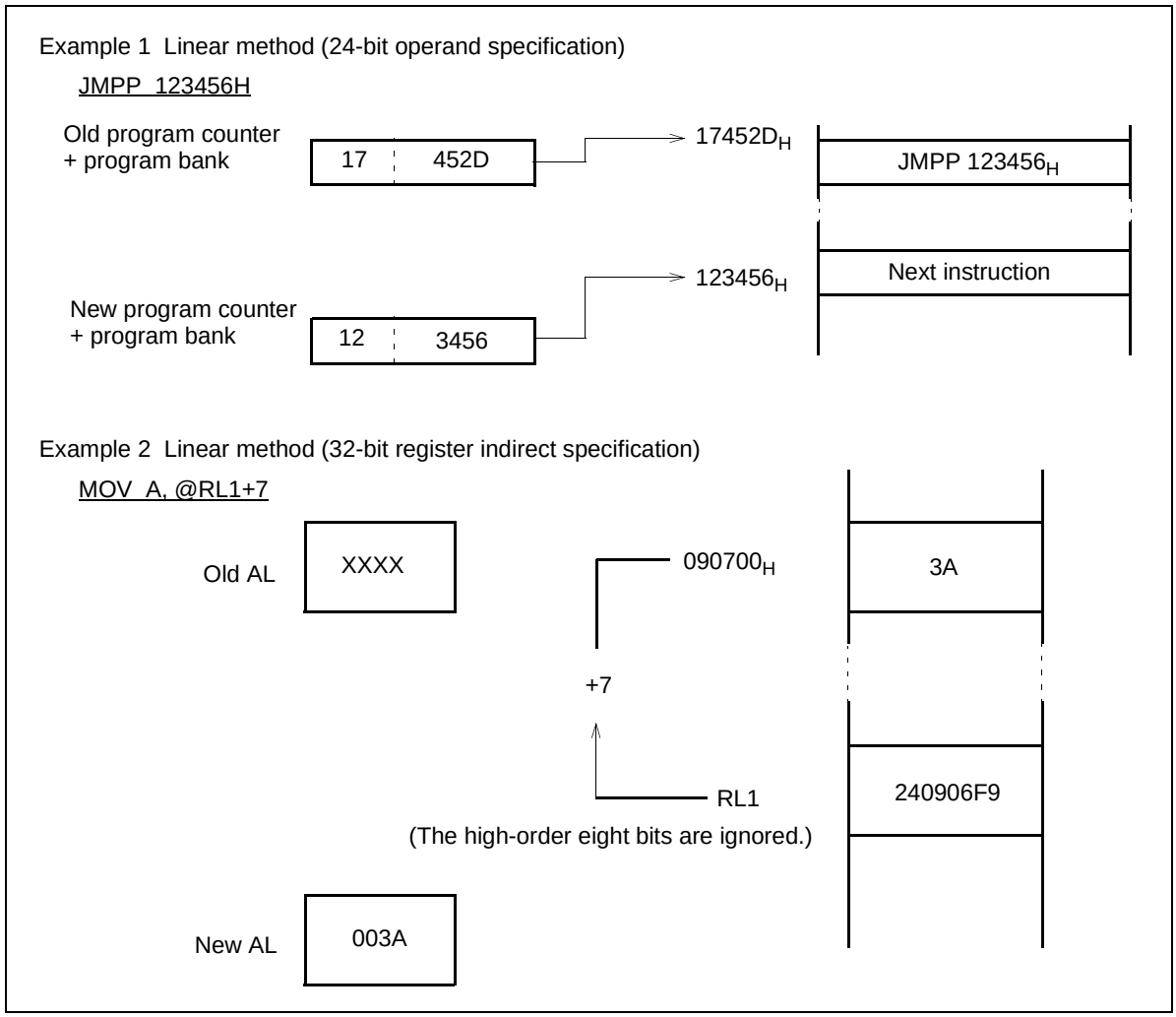


Figure 2.1.1b Sample linear addressing

■ Bank addressing types

In the bank method, the 16-Mbyte space is divided into 256 64-Kbyte banks. The following five bank registers are used to specify the banks corresponding to each space:

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

The 64-Kbyte bank specified by the PCB is called a program (PC) space. The PC space contains instruction codes, vector tables, and immediate value data, for example.

The 64-Kbyte bank specified by the DTB is called a data (DT) space. The DT space contains readable/ writable data, and control/data registers for internal and external resources.

The 64-Kbyte bank specified by the USP or SSP is called a stack (SP) space. The SP space is accessed when a stack access occurs during a push/pop instruction or interrupt register saving. The S flag in the condition code register determines the stack space to be accessed.

The 64-Kbyte bank specified by the ADB is called an additional (AD) space. The AD space, for example, contains data that cannot fit into the DT space.

Table 2.1.1a lists the default spaces used in each addressing mode, which are pre-determined to improve instruction coding efficiency. To use a non-default space for an addressing mode, specify a prefix code corresponding to a bank before the instruction. This enables access to the bank space corresponding to the specified prefix code.

After reset, the DTB, USB, SSB, and ADB are initialized to 00H. The PCB is initialized to a value specified by the reset vector. After reset, the DT, SP, and AD spaces are allocated in bank 00H (000000H to 00FFFFH), and the PC space is allocated in the bank specified by the reset vector.

Table 2.1.1a Default space

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

Figure 2.1.1c is an example of a memory space divided into register banks.

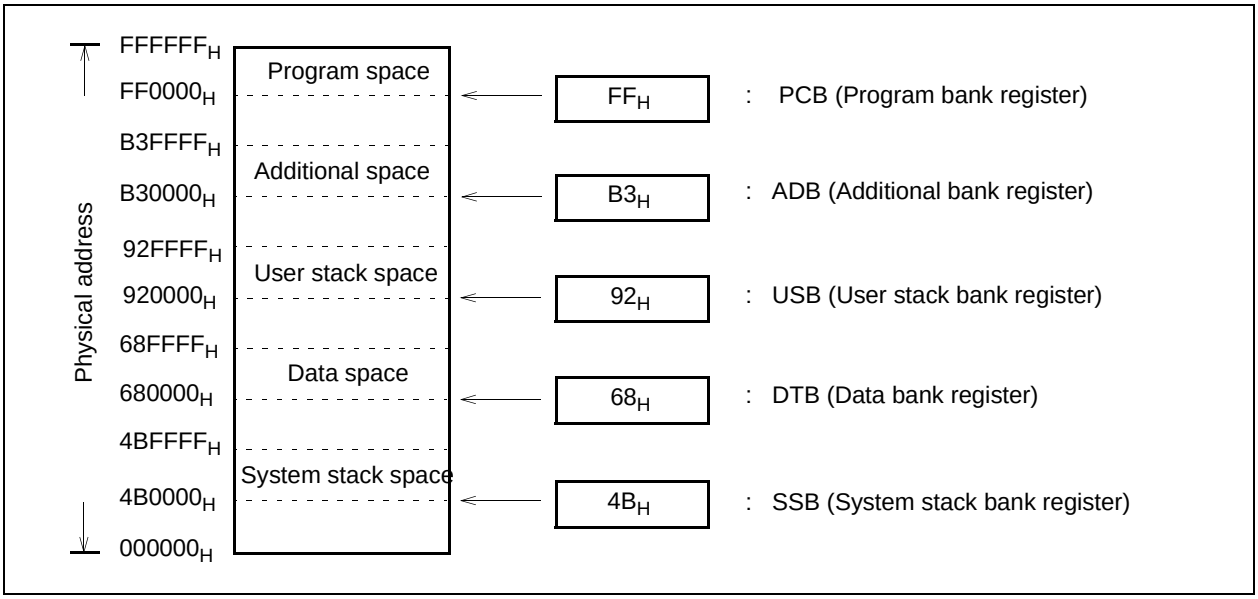


Figure 2.1.1c Physical addresses of each space

■ Multi-byte data allocation in memory space

Figure 2.1.1d is a diagram of multi-byte data configuration in memory. The low-order eight bits of a data item are stored at address n, then address n+1, address n+2, address n+3, etc.

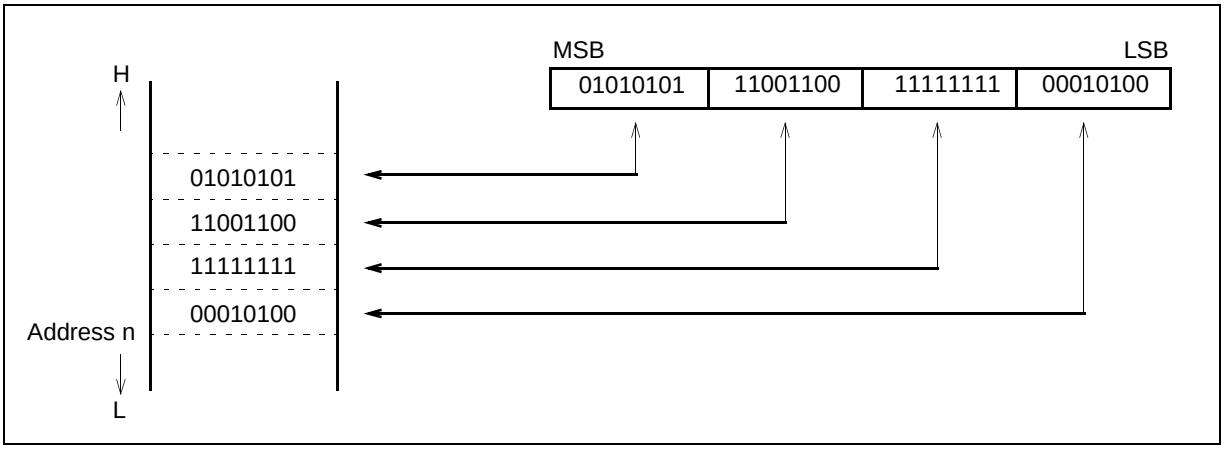


Figure 2.1.1d Sample allocation of multi-byte data in memory

Data is written to memory from the low-order addresses. Therefore, for a 32-bit data item, the low-order 16 bits are transferred before the high-order 16 bits.

If a reset signal is input immediately after the low-order bits are written, the high-order bits might not be written.

■ Accessing multi-byte data

Fundamentally, accesses are made within a bank. For an instruction accessing a multi-byte data item, address FFFFH is followed by address 0000H of the same bank. Figure 2.1.1e is an example of an instruction accessing multi-byte data.

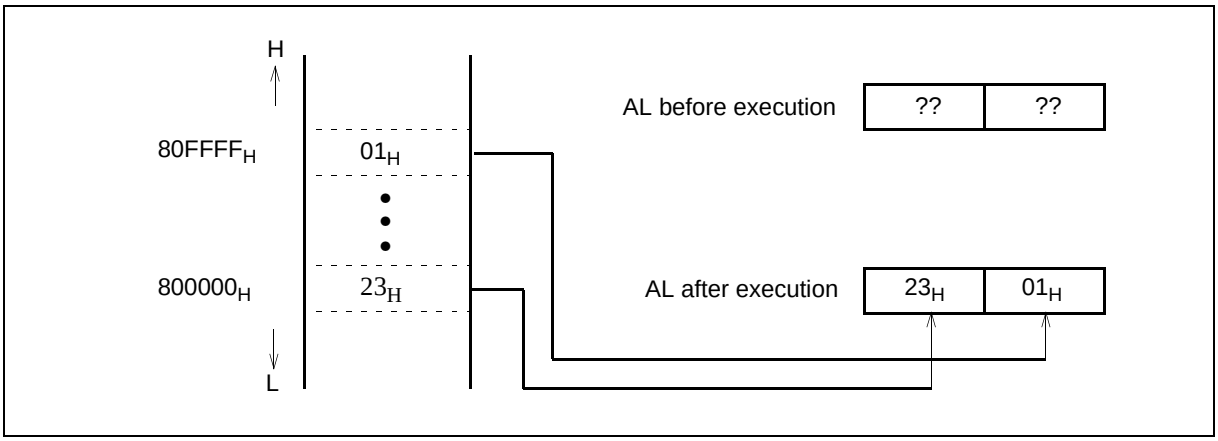


Figure 2.1.1e Execution of MOVW A, 080FFFFH

2.1.2 Registers

The F^{2}MC-16LX registers are largely classified into two types: special registers in the CPU and general-purpose registers in memory. The special registers are dedicated internal hardware of the CPU, and their applications are limited by the CPU architecture. The general-purpose registers share the CPU address space with RAM. The general-purpose registers are the same as the special registers in that they can be accessed without using an address. The applications of the general-purpose registers can be specified by the user however, as is ordinary memory space.

■ Special registers

The F^{2}MC-16LX has the following 13 special registers:

- Accumulator (A=AH:AL): Two 16-bit accumulators (Can be used as a single 32-bit accumulator.)
- User stack pointer (USP): 16-bit pointer indicating the user stack area
- System stack pointer (SSP): 16-bit pointer indicating the system stack area
- Processor status (PS): 16-bit register indicating the system status
- Program counter: 16-bit register holding the address of the program
- Program bank register: 8-bit register indicating the PC space
- Data bank register: 8-bit register indicating the DT space
- User stack bank register (USB): 8-bit register indicating the user stack space
- System stack bank register (SSB): 8-bit register indicating the system stack space
- Additional bank register (ADB): 8-bit register indicating the AD space
- Direct page register (DPR): 8-bit register indicating a direct page

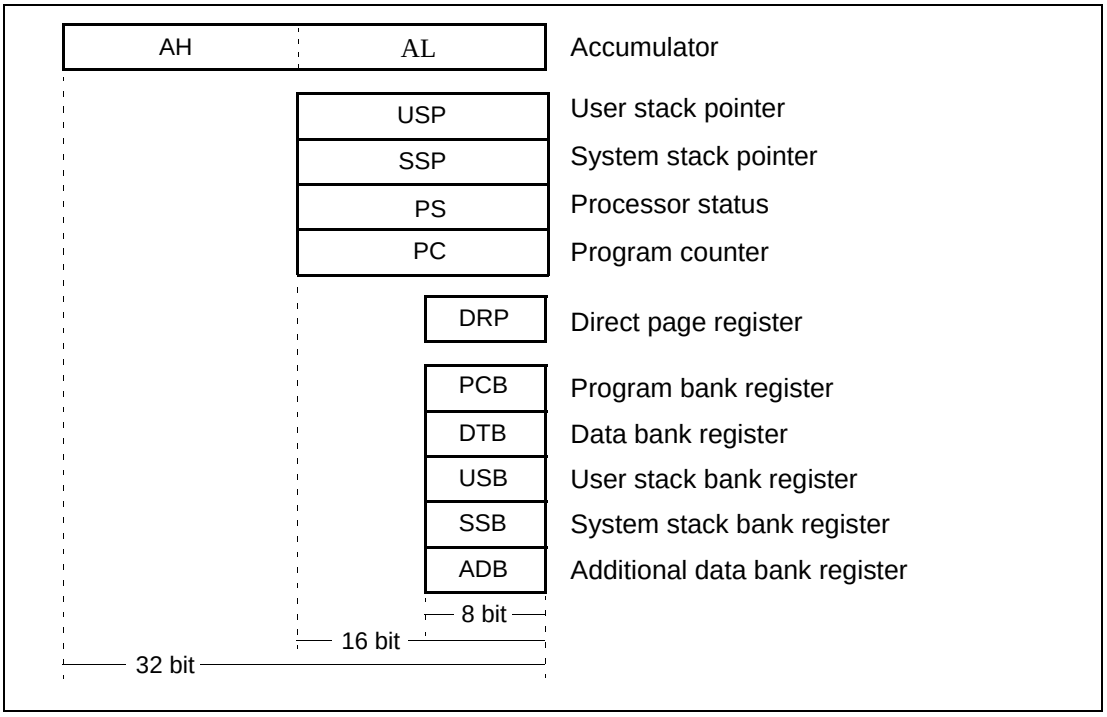


Figure 2.1.2a Special registers

■ General-purpose registers

The F^{2}MC-16LX general-purpose registers are located from addresses 000180H to 00037FH (maximum configuration) of main storage. The register bank pointer (RP) indicates which of the above addresses are currently being used as a register bank. Each bank has the following three types of registers. These registers are mutually dependent as described in Figure 2.1.2b.

- R0 to R7: 8-bit general-purpose register
- RW0 to RW7: 16-bit general-purpose register
- RL0 to RL3: 32-bit general-purpose register

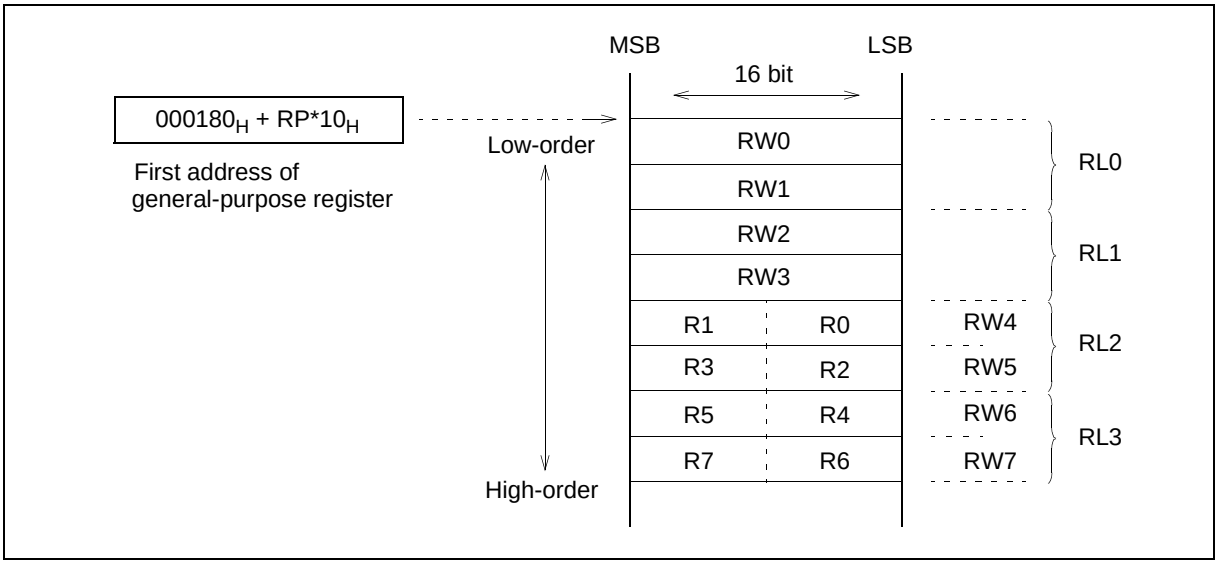


Figure 2.1.2b General-purpose registers

The relationship between the high-order and low-order bytes of a byte or word register is expressed as follows:

$$RW_{(i+4)} = R_{(i*2+1)}*256+R_{(i*2)} \quad [i=0 \text{ to } 3]$$

The relationship between the high-order and low-order bytes of Rli and RW can be expressed as follows:

$$RL_{(i)} = RW_{(i*2+1)}*65536+RW_{(i*2)} \quad [i=0 \text{ to } 3]$$

■ Program counter (PC)

The PC register is a 16-bit counter that indicates the low-order 16 bits of the memory address of an instruction code to be executed by the CPU. The high-order eight bits of the address are indicated by the PCB. The PC register is updated by a conditional branch instruction, subroutine call instruction, interrupt, or reset.

The PC register can also be used as a base pointer for operand access.

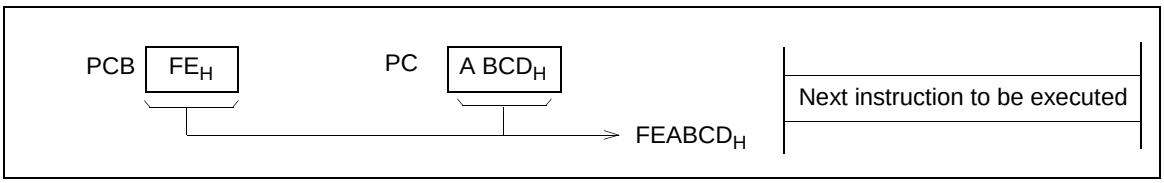


Figure 2.1.2c Program counter

■ Accumulator (A)

The A register consists of two 16-bit arithmetic operation registers (AH and AL). The A register is used as a temporary storage for operation results and transfer data. During 32-bit data processing, AH and AL are used together. Only AL is used for word processing in 16-bit data processing mode or for byte processing in 8-bit data processing mode (see Figures 2.1.9 and 2.1.10). The data stored in the A register can be operated upon with the data in memory or registers (Ri, Rwi, or Rli). In the same manner as with the F^2MC-8L, when a word or shorter data item is transferred to AL, the previous data item in AL is automatically sent to AH (data preservation function). The data preservation function and operation between AL and AH help improve processing efficiency.

When a byte or shorter data item is transferred to AL, the data is sign-extended or zero-extended and stored as a 16-bit data item in AL. The data in AL can be handled either as word or byte long.

When a byte-processing arithmetic operation instruction is executed on AL, the high-order eight bits of AL before operation are ignored. The high-order eight bits of the operation result all become zeroes.

The A register is not initialized by a reset. The A register holds an undefined value immediately after a reset.

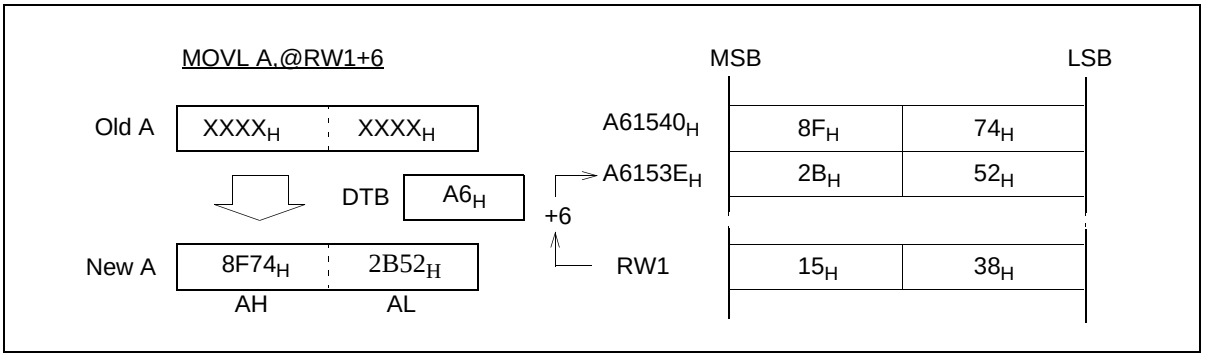


Figure 2.1.2d 32-bit data transfer

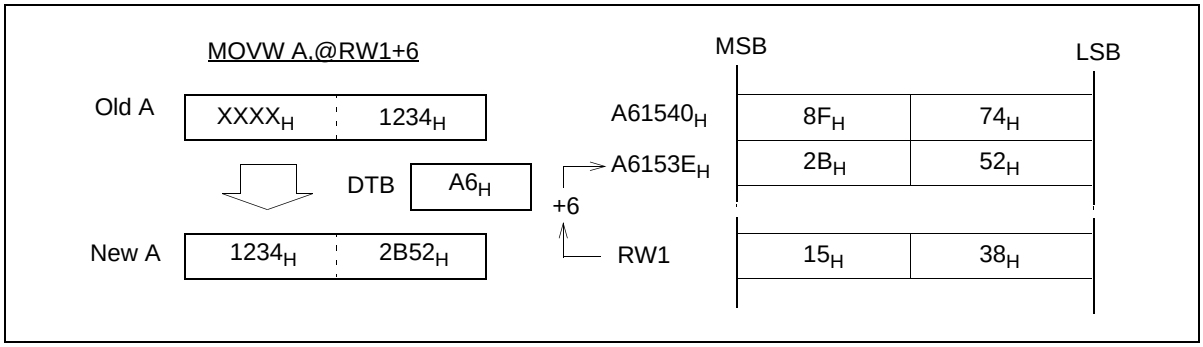


Figure 2.1.2e AL-AH transfer

■ User stack pointer (USP) and system stack pointer (SSP)

USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data in the event of a push/pop instruction or subroutine execution. The USP and SSP registers are used by stack instructions. The USP register is enabled when the S flag in the processor status register is '0,' and the SSP register is enabled when the S flag is '1' (see Figure 2.1.2f). Since the S flag is set when an interrupt is accepted, register values are always saved in the memory area indicated by SSP during interrupt processing. SSP is used for stack processing in an interrupt routine, while USP is used for stack processing outside an interrupt routine. If the stack space is not divided, use only the SSP.

During stack processing, the high-order eight bits of an address are indicated by SSB (for SSP) or USB (for USP). USP and SSP are not initialized by a reset. Instead, they hold undefined values.

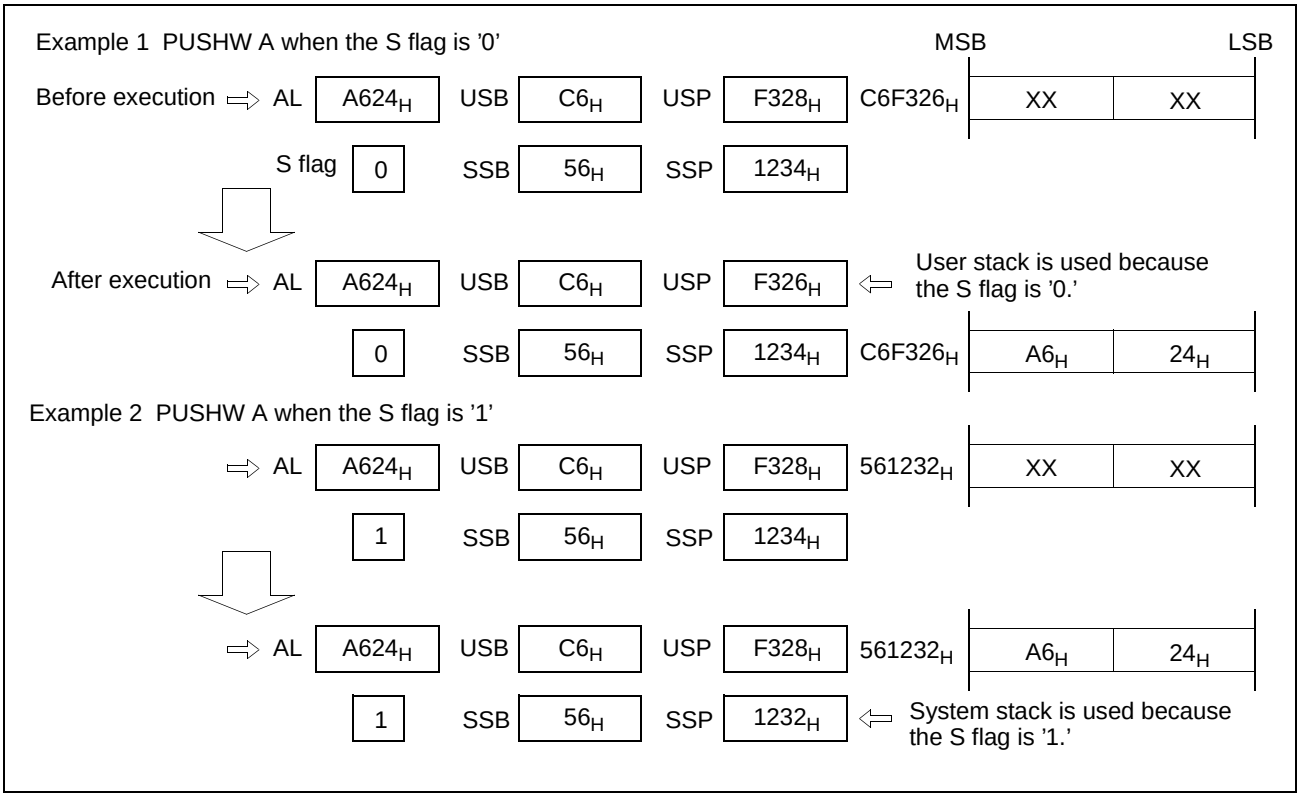


Figure 2.1.2f Stack manipulation instruction and stack pointer

Note: Specify an even-numbered address in the stack pointer whenever possible.

■ Processor status (PS)

The PS register consists of the bits controlling the CPU Operation and the bits indicating the CPU status.

As shown in Figure 2.1.2g, the high-order byte of the PS register consists of a register bank pointer (RP) and an interrupt level mask register (ILM). The RP indicates the start address of a register bank. The low-order byte of the PS register is a condition code register (CCR), containing the flags to be set or reset depending on the results of instruction execution or interruptoccurrences.

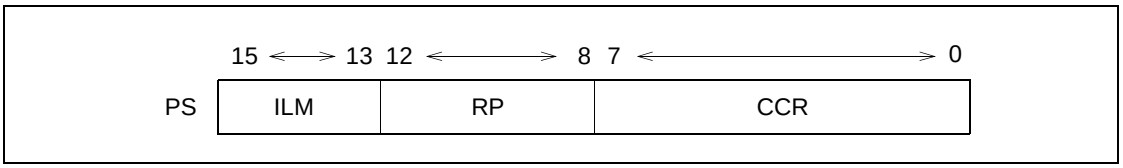


Figure 2.1.2g PS structure

(1)Condition code register (CCR)

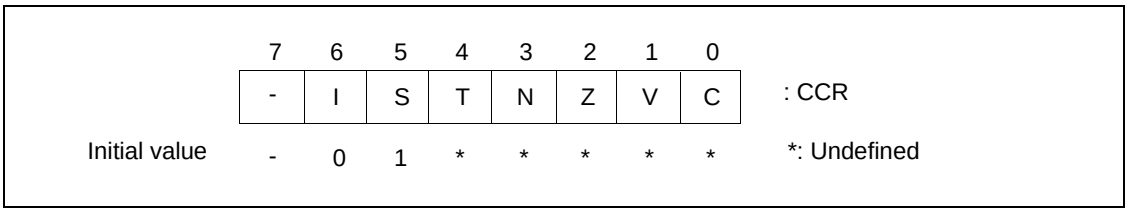


Figure 2.1.2h Condition code register configuration

I:Interrupt enable flag: Interrupts other than software interrupts are enabled when the I flag is 1 and are masked when the I flag is 0. The I flag is cleared by a reset.

S:Stack flag: When the S flag is 0, USP is enabled as the stack manipulation pointer. When the S flag is 1, SSP is enabled as the stack manipulation pointer. The S flag is set by an interrupt reception or a reset.

T:Sticky bit flag: 1 is set in the T flag when there is at least one ’1’ in the data shifted out from the carry after execution of a logical right/arithmetic right shift instruction. Otherwise, 0 is set in the T flag. In addition, ’0’ is set in the T flag when the shift amount is zero.

N:Negative flag: The N flag is set when the MSB of the operation result is ’1,’ and is otherwise cleared.

Z:Zero flag: The Z flag is set when the operation result is all zeroes, and is otherwise cleared.

V:Overflow flag: The V flag is set when an overflow of a signed value occurs as a result of operation execution and is otherwise cleared.

C:Carry flag: The C flag is set when a carry-up or carry-down from the MSB occurs as a result of operation execution and is otherwise cleared.

(2) Register bank pointer (RP)

The RP register indicates the relationship between the general-purpose registers of the F2MC-16LX and the internal RAM addresses. Specifically, the RP register indicates the first memory address of the currently used register bank in the following conversion expression: [00180H + (RP)*10H] (see Figure 2.1.2i). The RP register consists of five bits, and can take a value between 00H and 1FH. Register banks can be allocated at addresses from 000180H to 000370H in memory.

Even within that range, however, the register banks cannot be used as general-purpose registers if the banks are not in internal RAM. The RP register is initialized to all zeroes by a reset. An instruction may transfer an eight-bit immediate value to the RP register; however, only the low-order five bits of that data are used.

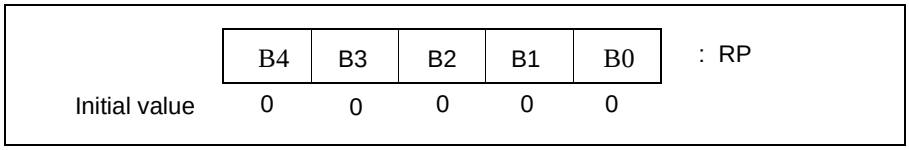


Figure 2.1.2i Register bank pointer

(3) Interrupt level mask register (ILM)

The ILM register consists of three bits, indicating the CPU interrupt masking level. An interrupt request is accepted only when the level of the interrupt is higher than that indicated by these three bits. Level 0 is the highest priority interrupt, and level 7 is the lowest priority interrupt (see Table 2.1.2a). Therefore, for an interrupt to be accepted, its level value must be smaller than the current ILM value. When an interrupt is accepted, the level value of that interrupt is set in ILM. Thus, an interrupt of the same or lower level cannot be accepted subsequently. ILM is initialized to all zeroes by a reset. An instruction may transfer an eight-bit immediate value to the ILM register, but only the low-order three bits of that data are used.

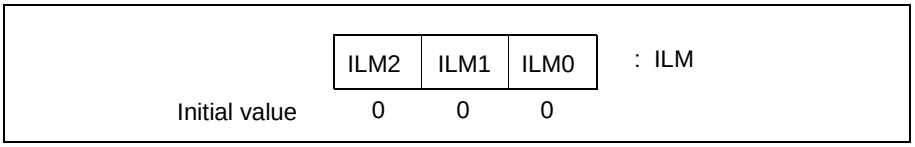


Figure 2.1.2j Interrupt level register

Table 2.1.2a Levels indicated by the interrupt level mask (ILM) register

ILM2	ILM1	ILM0	Level value	Acceptable interrupt level
0	0	0	0	Interrupt disabled
0	0	1	1	0 only
0	1	0	2	Level value smaller than 1
0	1	1	3	Level value smaller than 2
1	0	0	4	Level value smaller than 3
1	0	1	5	Level value smaller than 4
1	1	0	6	Level value smaller than 5
1	1	1	7	Level value smaller than 6

■ Register bank

A register bank consists of eight words. The register bank can be used as the following general-purpose registers for arithmetic operations: byte registers R0 to R7, word registers RW0 to RW7, and long word registers RL0 to RL3. In addition, the register bank can be used as instruction pointers.

Table 2.1.2b lists the functions of the registers. Table 2.1.2c indicates the relationship between the registers.

In the same manner as for an ordinary RAM area, the register bank values are not initialized by a reset. The status before a reset is maintained. When the power is turned on, however, the register bank will have an undefined value.

Table 2.1.2b Register functions

R0 to R7	Used as operands of instructions. Note: R0 is also used as a counter for barrel shift or normalization instructions.
RW0 to RW7	Used as pointers. Used as operands of instructions. Note: RW0 is used as a counter for string instructions.
RL0 to RL3	Used as long pointers. Used as operands of instructions.

Table 2.1.2c Relationship between registers

<table>
<tr><td rowspan="4"></td><td>RW0</td><td rowspan="2">RL0</td></tr>
<tr><td>RW1</td></tr>
<tr><td>RW2</td><td rowspan="2">RL1</td></tr>
<tr><td>RW3</td></tr>
<tr><td>R0</td><td rowspan="2">RW4</td><td rowspan="4">RL2</td></tr>
<tr><td>R1</td></tr>
<tr><td>R2</td><td rowspan="2">RW5</td></tr>
<tr><td>R3</td></tr>
<tr><td>R4</td><td rowspan="2">RW6</td><td rowspan="4">RL3</td></tr>
<tr><td>R5</td></tr>
<tr><td>R6</td><td rowspan="2">RW7</td></tr>
<tr><td>R7</td></tr>
</table>

■ Program counter bank register (PCB) <Initial value: Value in reset vector>

Data bank register(DTB) <Initial value: 00H>

User stack bank register(USB) <Initial value: 00H>

System stack bank register(SSB) <Initial value: 00H>

Additional data bank register(ADB) <Initial value: 00H>

Each bank register indicates the memory bank where the PC, DT, SP (user), SP (system), or AD space is allocated. All bank registers are one byte long. PCB is initialized to 00H by a reset. Bank registers other than PCB can be read or written to. PCB can be read but cannot be written to.

PCB is updated when the JMPP, CALLP, RETP, RETI, or RETF instruction branching to the entire 16-Mbyte space is executed or when an interrupt occurs. For operation of each register, see Chapter 2, Section 2.1.1, "Memory space."

■ Direct page register (DPR) <Initial value: 01H>

DPR specifies addr8 to addr15 of the instruction operands in direct addressing mode as shown in Figure 2.1.2k. DPR is eight bits long, and is initialized to 01H by a reset. DPR can be read or written to by an instruction.

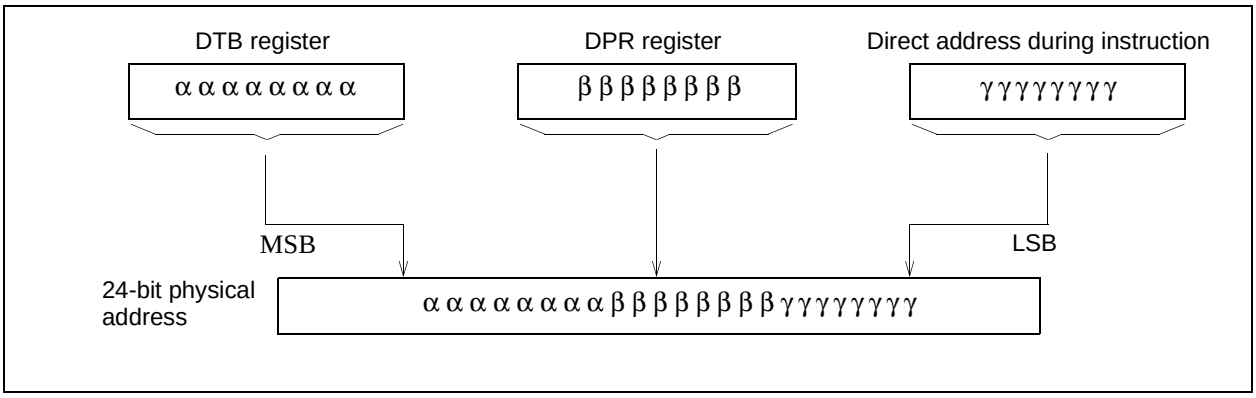


Figure 2.1.2k Generating a physical address in direct addressing mode

2.1.3 Prefix codes

Placing a prefix code before an instruction partially changes the operation of the instruction. Three types of prefix codes can be used: bank select prefix, common register bank prefix, and flag change disable prefix.

■ Bank select prefix

The memory space used for accessing data is determined for each addressing mode.

When a bank select prefix is placed before an instruction, the memory space used for accessing data by that instruction can be selected regardless of the addressing mode.

Table 2.1.3a Bank select prefix

Bank select prefix	Space selected
PCB	PC space
DTB	Data space
ADB	AD space
SPB	Either the SSP or USP space is used according to the stack flag value.

Use the following instructions with care:

(1) String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)

The bank register specified by an operand is used regardless of the prefix.

(2)Stack manipulation instructions (PUSHW, POPW)

SSB or USB is used according to the S flag regardless of the prefix.

(3)I/O access instructions

MOV A, io / MOV io, A /MOVX A, io / MOVW A, io /MOVW io, A / MOV io, #imm8
MOVW io, #imm16 / MOVB A, io:bp / MOVB io:bp, A /SETB io:bp / CLRB io:bp
BBC io:bp, rel / BBS io:bp, rel WBTC, WBTS

The IO space of the bank is used regardless of the prefix.

(4)Flag change instructions (AND CCR,#imm8, OR CCR,#imm8)

The instruction is executed normally, but the prefix affects the next instruction.

(5)POPW PS

SSB or USB is used according to the S flag regardless of the prefix. The prefix affects the next instruction.

(6)MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

(7)RETI

SSB is used regardless of the prefix.

■ Common register bank prefix (CMR)

To simplify data exchange between multiple tasks, the same register bank must be accessed relatively easily regardless of the RP value. When CMR is placed before an instruction that accesses a register bank, that instruction accesses the common bank (the register bank selected when RP=0) at addresses from 000180H to 00018FH regardless of the current RP value. Use the following instructions with care:

(1)String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)

If an interrupt request occurs during execution of a string instruction with a prefix code, the prefix code becomes invalid when the string instruction is resumed after the interrupt is processed. Thus, the string instruction is executed falsely after the interrupt is processed. Do not prefix any of the above string instructions with CMR.

(2)Flag change instructions (AND CCR,#imm8, OR CCR,#imm8, POPW PS)

The instruction is executed normally, but the prefix affects the next instruction.

(3)MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

■ Flag change disable prefix

To disable flag changes, use the flag change disable prefix code (NCC). Placing NCC before an instruction disables flag changes associated with that instruction. Use the following instructions with care:

(1) String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)

If an interrupt request occurs during execution of a string instruction with a prefix code, the prefix code becomes invalid when the string instruction is resumed after the interrupt is processed. Thus, the string instruction is executed incorrectly after the interrupt is processed. Do not prefix any of the above string instructions with NCC.

(2) Flag change instructions (AND CCR,#imm8, OR CCR,#imm8, POPW PS)

The instruction is executed normally, but the prefix affects the next instruction.

(3)Interrupt instructions (INT #vct8, INT9, INT addr16, INTP addr24, RETI)

CCR changes according to the instruction specifications regardless of the prefix.

(4)JCTX @A

CCR changes according to the instruction specifications regardless of the prefix.

(5)MOV ILM,#imm8

The instruction is executed normally, but the prefix affects the next instruction.

■ Interrupt disable instructions

Interrupt requests are not sampled for the following ten instructions:

• MOV ILM,#imm8	• PCB	• SPB	• OR CCR,#imm8	• NCC
• AND CCR,#imm8	• ADB	• CMR	•POPW PS	• DTB

If a valid interrupt request occurs during execution of any of the above instructions, the interrupt can be processed only when an instruction other than the above is executed. For details, see Figure 2.1.3a.

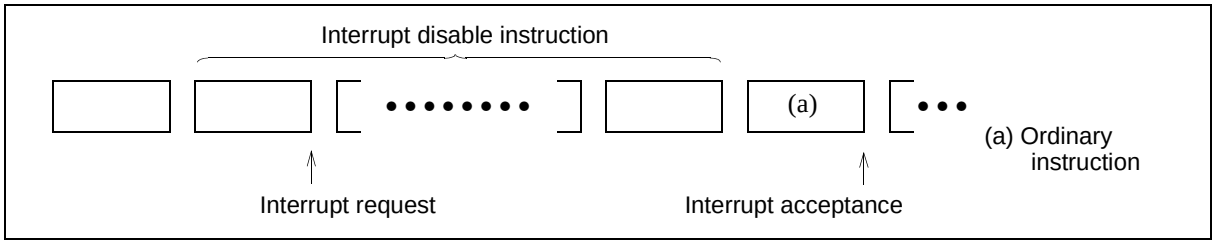


Figure 2.1.3a Interrupt disable instruction

■ Restrictions on interrupt disable instructions and prefix instructions

When a prefix code is placed before an interrupt disable instruction, the prefix code affects the first instruction after the code other than the interrupt disable instruction.

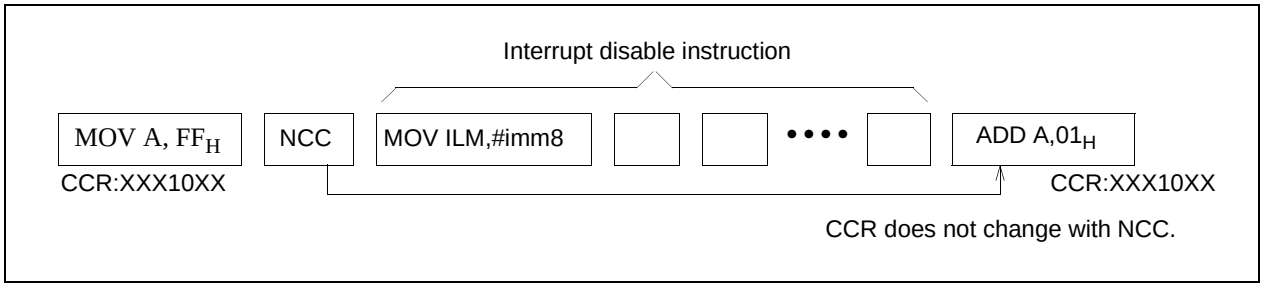


Figure 2.1.3b Interrupt disable instructions and prefix codes

■ Consecutive prefix codes

When competitive prefix codes are placed consecutively, the latter becomes valid.

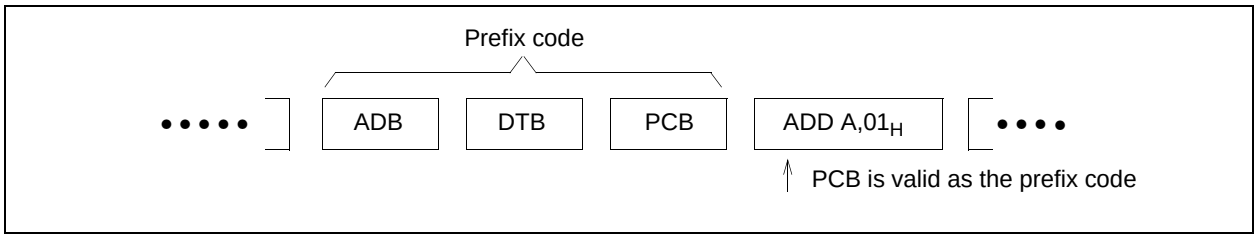


Figure 2.1.3c Consecutive prefix codes

In the figure above, competitive prefix codes are PCB, ADB, DTB, and SPB.

Chapter 3: Memory

3.1 Memory Access Modes

In the F^{2}MC-16LX, there are several modes for access methods, access areas, and test methods. In this module, the following classifications apply:

Table 3.1a Memory Access Mode

Operation Mode	Bus Mode	Access Mode (External data bus width)
RUN	Single Chip	———
	Internal ROM, external bus	8 bit
		16 bit
	External ROM, external bus	8 bit
		16 bit
EPROM Write	———	———
Test Functions	———	———

■ Operation mode

Operation mode means the mode for controlling the device operation status. The operation mode is specified by the MDx mode setting pin and the Ex bit in mode data. By selecting an operation mode, normal operation, internal test program activation, or special test function activation can be performed.

■ Bus mode

Bus mode means the mode for controlling the internal ROM operation and external access function. The bus mode is specified by the MDx mode setting pin and the Mx bit in mode data. The MDx mode setting pin specifies the bus mode for reading the reset vector and mode data, and the Mx bit in mode data specifies the bus mode for normal operation.

■ Access mode

Access mode means the mode for controlling the external data bus width. The access mode is specified by the MDx mode setting pin and the Sx bit in mode data. By selecting an access mode, an 8- or 16-bit external data bus is specified.

3.1.1 Mode pins

Table 3.1.1a describes the operations specified by combinations of the MD2 to MD0 external pins.

Table 3.1.1a Mode pins and modes

Mode pin setting MD2 MD1 MD0	Mode name	Reset vector access area	External data bus width	Remarks
0 0 0	External vector mode 0	External	8 bits	
0 0 1	External vector mode 1	External	16 bits	Reset vector, 16-bit bus width access
0 1 0	(Specification inhibited)			
0 1 1	Internal vector mode	Internal	(Mode data)	Reset sequence are based on mode data.
1 0 0	(Specification inhibited)			
1 0 1				
1 1 0				
1 1 1	EPROM write	——	——	

Note: When using internal vector mode 0, the initial value of IOBS and LMBS are '0'. If IOBS and LMBS are set afterwards, the address range 0000C0H..0000FFH and 002000H..7FFFFFH will use 16-bit data bus.

3.1.2 Mode data

Mode data is stored at $FFFFDF_H$ of main memory and used for controlling the CPU operation. This data is fetched during a reset sequence and stored in the mode register inside the device. The mode register value can be changed only by a reset sequence.

The setting of this register is valid after the reset sequence.

Always set the reserved bits to '0.'

Here is a diagram of the setting of the bits.

Mode data

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: $FFFFDF_H$	M1	M0	—	—	S0	—	—	—	

[bit 7, 6] : Bus mode setting bits

These bits are used to specify the operation mode after the reset sequence is completed. Here shows the relationship between the bits and the functions.

M1	M0	Function	Remarks
0	0	Single chip mode	
0	1	Internal ROM and external bus mode	
1	0	External ROM and external bus mode	
1	1	(Inhibited)	

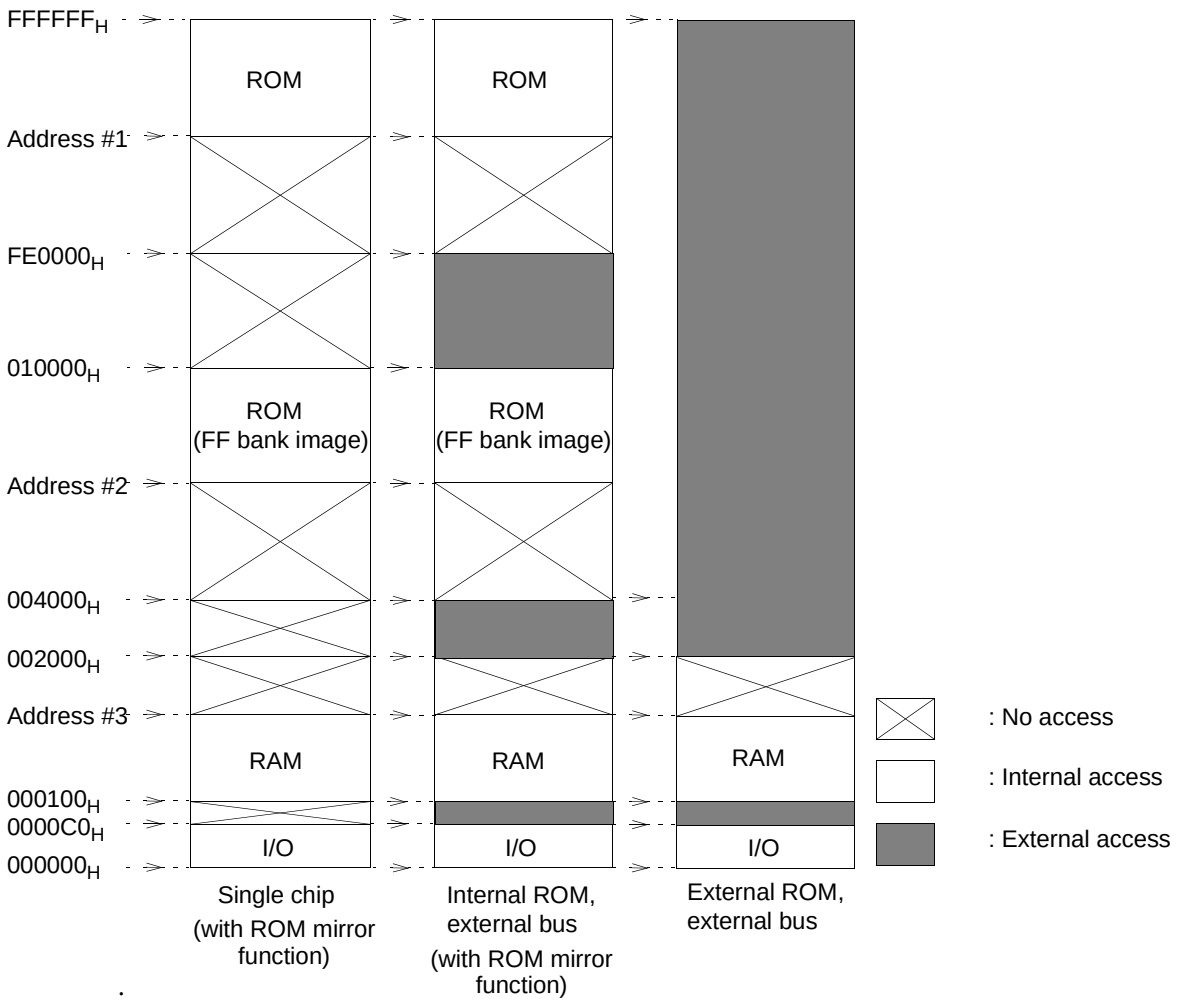
[bit 3] : Mode setting bits

These bits are used to specify the bus mode or access mode after the reset sequence is completed. The following table shows the relationship between the bits and the functions.

S0	Function	Remarks
0	External data bus, 8-bit mode	
1	External data bus, 16-bit mode	

3.1.3 Bus Mode

Figure 3.1.3a shows correspondence between the access areas and physical addresses for each bus mode.



Device	Address #1	Address #2	Address #3
MB90583	FE0000H	004000H	001900H
MB90F583	FE0000H	004000H	001900H
MB90V580	(FE0000H)	004000H	001900H

Note: If ROM mirroring function is not wanted, please refer to Chapter 22, ROM Mirroring Module.

Note: The high-order portion of bank 00 gives the image of the FF bank ROM to make the small model of the C compiler effective. Since the low-order 16 bits are the same, the table in ROM can be referenced without using the far specification in the pointer declaration.For example, an attempt to access 00C000$_H$ accesses the value at FFC000$_H$ in ROM.

Note: The ROM area in bank FF exceeds 48 Kbytes, and its entire image cannot be shown in bank 00.

Note: With the MB90F583 and MB90583, the image between FF4000H and FFFFFF$_H$ is visible in bank 00, while the image between FE0000$_H$ and FF3FFF$_H$ is visible only in bank FE or FF

Figure 3.1.3a Access areas and physical addresses in each bus mode

■ Recommended setting

Table 3.1.3a lists a sample recommended setting of mode pins and mode data.

Table 3.1.3a Sample recommended setting of mode pins and mode data

Sample setting	MD2	MD1	MD0	M1	M0	S0
Single chip	0	1	1	0	0	×
Internal ROM and external bus mode, 16-bit bus	0	1	1	0	1	1
Internal ROM and external bus mode, 8-bit bus	0	1	1	0	1	0
External ROM and external bus mode, 16-bit bus, vector 16 bus width	0	0	1	1	0	1
External ROM and external bus mode, 8-bit bus	0	0	0	1	0	0

Note: I/O signals appearing on an external pin connected to this module vary with the mode.

Table 3.1.3b lists the external pin function in each modes

Table 3.1.3b Modes and related external pin operations

<table>
<tr><th rowspan="3">Pin name</th><th colspan="4">Function</th></tr>
<tr><th rowspan="2">Single chip</th><th colspan="2">External bus extension</th><th rowspan="2">EPROM write</th></tr>
<tr><th>8 bits</th><th>16 bits</th></tr>
<tr><td>P07 to P00</td><td rowspan="10">Port</td><td colspan="2">AD07 to 00</td><td>D07 to 00</td></tr>
<tr><td>P17 to P10</td><td>A15 to 08</td><td>AD15 to 08</td><td>A15 to 08</td></tr>
<tr><td>P27 to P20</td><td colspan="2">A23 to 16</td><td>A07 to 00</td></tr>
<tr><td>P30</td><td colspan="2">ALE</td><td>A16</td></tr>
<tr><td>P31</td><td colspan="2">RDX</td><td>CEX</td></tr>
<tr><td>P32</td><td>WRX</td><td>WRLX</td><td>OEX</td></tr>
<tr><td>P33</td><td>Port</td><td>WRHX</td><td>PGMX</td></tr>
<tr><td>P34</td><td colspan="2">HRQ</td><td rowspan="4">Unused</td></tr>
<tr><td>P35</td><td colspan="2">HAKX</td></tr>
<tr><td>P36</td><td colspan="2">RDY</td></tr>
<tr><td>P37</td><td colspan="2">CLK</td></tr>
</table>

Note: The high-order bits of an address and WRX, WRLX, WRHX, HAKX, HRQ, RDY, and CLK can be used as ports depending on function selection.

3.2 External Memory Access

To access external memory and peripherals, the F^{2}MC-16LX supplies the following address, data, and control signals:

- ▷ CLK (P37) : Machine cycle clock (KBP)
- ▷ RDY (P36) : External ready input pin
- ▷ WRHX (P33) : Write signal for high-order 8 bits of data bus
- ▷ WRLX (P32) : Write signal for low-order 8 bits of data bus
- ▷ RDX (P31) : Read signal
- ▷ ALE (P30) : Address latch enable signal

The external bus pin control circuit controls the external bus pins for externally extending the CPU address/data bus.

3.2.1 Block diagram

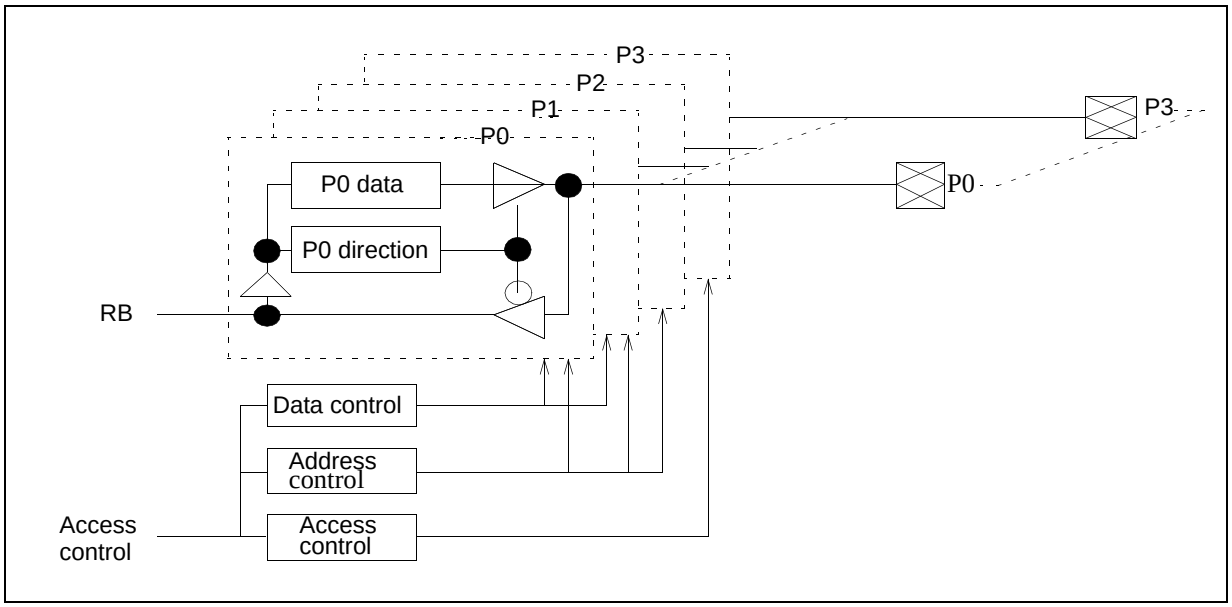


Figure 3.2.1a External bus pin control circuit

3.2.2 Registers and Register details

Automatic ready function selection register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 0000A5$_H$	IOR1	IOR0	HMR1	HMR0	—	—	LMR1	LMR0	ARSR
Read/write ⇨	(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	
Initial value ⇨	(0)	(0)	(1)	(1)	(-)	(-)	(0)	(0)	

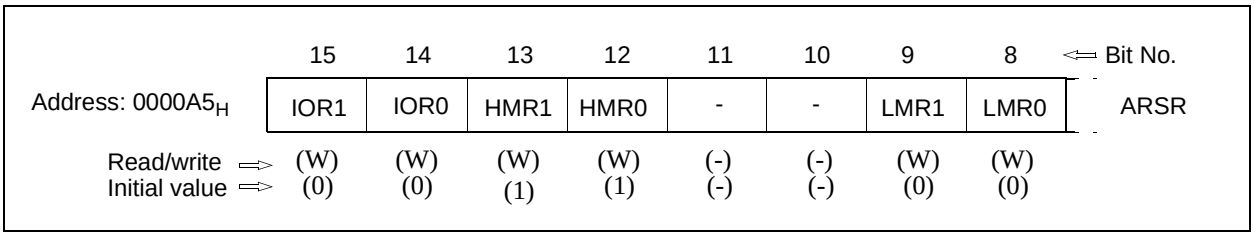
External address output control register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 0000A6$_H$	E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Bus control signal selection register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: 0000A7$_H$	CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	
Initial value ⇨	(0)	(0)	(0)	(0)	(1/0)	(0)	(0)	(-)	

3.2.2.1 Automatic ready function selection register



[bits 15 and 14]: IOR1 and IOR0

These bits specify the automatic wait function for external access to the area between 000000_H and $0000FF_H$.

IOR1	IOR0	Setting
0	0	Automatic wait disabled
0	1	One machine cycle of automatic wait for external access
1	0	Two machine cycles of automatic wait for external access
1	1	Three machine cycles of automatic wait for external access

The initial value is ’00B.’

[bits 13 and 12]: HMR1 and HMR0

These bits specify the automatic wait function for external access to the area between 800000_H and $FFFFFF_H$.

HMR1	HMR0	Setting
0	0	Automatic wait disabled
0	1	One machine cycle of automatic wait for external access
1	0	Two machine cycles of automatic wait for external access
1	1	Three machine cycles of automatic wait for external access

The initial value is ’11B.’

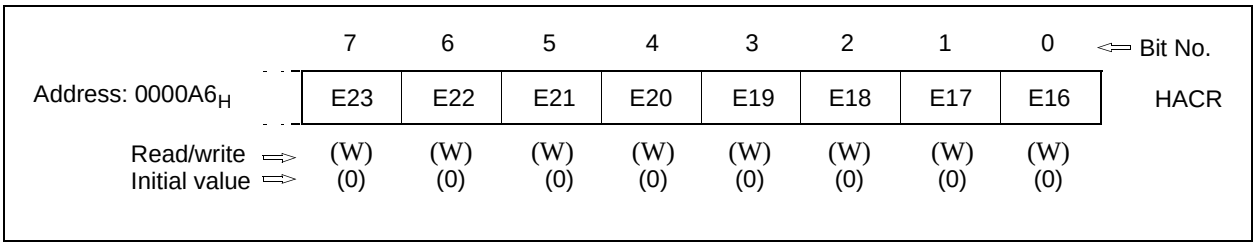
[bits 9 and 8]: LMR1 and LMR0

These bits specify the automatic wait function for external access to the area between 002000_H and $7FFFFF_H$.

LMR1	LMR0	Setting
0	0	Automatic wait disabled
0	1	One machine cycle of automatic wait for external access
1	0	Two machine cycles of automatic wait for external access
1	1	Three machine cycles of automatic wait for external access

The initial value is ’00B.’

3.2.2.2 External address output control register



This register controls the external output of addresses (A19 to A16). The bits corresponds to addresses A19 to A16, controlling the address output pins as described below.

Table 3.2.0a Selecting the high-order address bit output control

0	The corresponding pin is used as an address output (Axx).
1	The corresponding pin is used as an I/O port (Pxx).

This register cannot be accessed when the device is in single chip mode. In that case, all pins are used as I/O ports regardless of the value of this register.

All bits of this register are write-only bits. '1' is always read from these bits.

These bits are initialized to '0' upon a reset.

3.2.2.3 Bus control signal selection register

Bus control signal selection register

	15	14	13	12	11	10	9	8	⇐ Bit No.
Address: $0000A7_H$	CKE	RYE	HDE	IOBS	HMBS	WRE	LMBS	—	ECSR
Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(-)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(-)	

This register is used to set the bus control function in external bus mode.

This register cannot be accessed when the device is in single chip mode. In that case, all pins are used as I/O ports regardless of the value of this register.

All bits of this register are write-only bits. '1' is always read from these bits.

[bit 15]: CKE

This bit controls the external clock (CLK) output as described below.

0	I/O port (P37) operation (clock output disabled)
1	Clock signal (CLK) output enabled

This bit is initialized to '0' upon a reset.

[bit 14]: RYE

This bit controls the external ready (RDY) input as described below.

0	I/O port (P36) operation (external RDY input disabled) [default]
1	External ready (RDY) input enabled

This bit is initialized to '0' upon a reset.

[bit 13]: HDE

This bit specifies whether to enable I/O of hold-related pins. This bit controls the hold request input (HRQ) and hold acknowledge output (HAKX) pins as described below.

0	I/O port (P35 and P34) operation (Hold function I/O disabled) [default]
1	Hold request (HRQ) input/hold acknowledge (HAKX) output enabled

This bit is initialized to '0' upon a reset.

[bit 12]: IOBS

This bit specifies the bus size when an area between 0000C0H and 0000FFH is externally accessed in 16-bit external data bus mode. The size is controlled as described below.

0	16-bit bus size access [default]
1	8-bit bus size access

This bit is initialized to '0' upon a reset.

[bit 11]: HMBS

This bit specifies the bus size when an area between 800000_H and $FFFFFF_H$ is externally accessed in 16-bit external data bus mode. The size is controlled as described below.

0	16-bit bus size access [default in mode other than external vector mode 2]
1	8-bit bus size access [default in external vector mode 2]

This bit is initialized to ’1’ upon a reset in external vector mode 2. In any other mode, this bit is initialized to ’0’ upon a reset.

[bit 10]: WRE

This bit controls the output of the external write signal pin (WRHX and WRLX pins in 16-bit bus mode and WRX pin in 8-bit bus mode) as described below.

0	I/O port (P33 and P32) operation (write signal output disabled) [default]
1	Write strobe signal (WRHX/WRLX or WRX) output enabled

In 8-bit external data bus mode, P33 is used as an I/O port regardless of the value of this register.

This bit is initialized to ’0’ upon a reset.

[bit 9]: LMBS

This bit specifies the bus size when an area between 002000_H and $7FFFFF_H$ is externally accessed in 16-bit external data bus mode. The size is controlled as described below.

0	16-bit bus size access [default]
1	8-bit bus size access

This bit is initialized to ’1’ upon a reset.

Note: In 16-bit bus mode, set P33 and P32 in input mode (set ’0’ in bits 3 and 2 of DDR3) when enabling the WRHX and WRLX functions by the WRE bit. In 8-bit bus mode, set P32 in input mode (set ’0’ in bit 2 of DDR3) when enabling the WRX function by the WRE bit. Even if RDY or HRQ input is enabled by the RYE or HDE bit, the I/O port function of the port is valid. Therefore, ensure that ’0’ (input mode) is written to the DDR3 bit corresponding to the port.

3.2.1 Operations

MB90580 has a variety of access method and access area modes. See Section 3.1 ,“Memory Access Modes”

(1) External memory access control signals

External memory is accessed in three cycles while the ready function is not used. Figure 3.2.4 shows the concept of external access timing.

The 8-bit bus width access in external 16-bit bus mode is used to read or write to an 8-bit peripheral chip when both 8- and 16-bit peripheral chips are connected to the external bus. Sine the 8-bit bus width access is executed using the low-order 8 bits of the data bus, ensure that the 8-bit peripheral chips are connected to the low-order 8 bits of the data bus.

Use the HMBS, LMBS, and IOBS bits of EPCR to specify whether to perform 16- or 8-bit bus width access in external 16-bit bus mode.

If only an address and ALE assert signal are output and RDX, WRX, WRLX, and WRHX are not asserted, the bus operation may not be actually performed. Ensure that a peripheral chip is not accessed by only an ALE signal.

■ External 8-bit bus mode

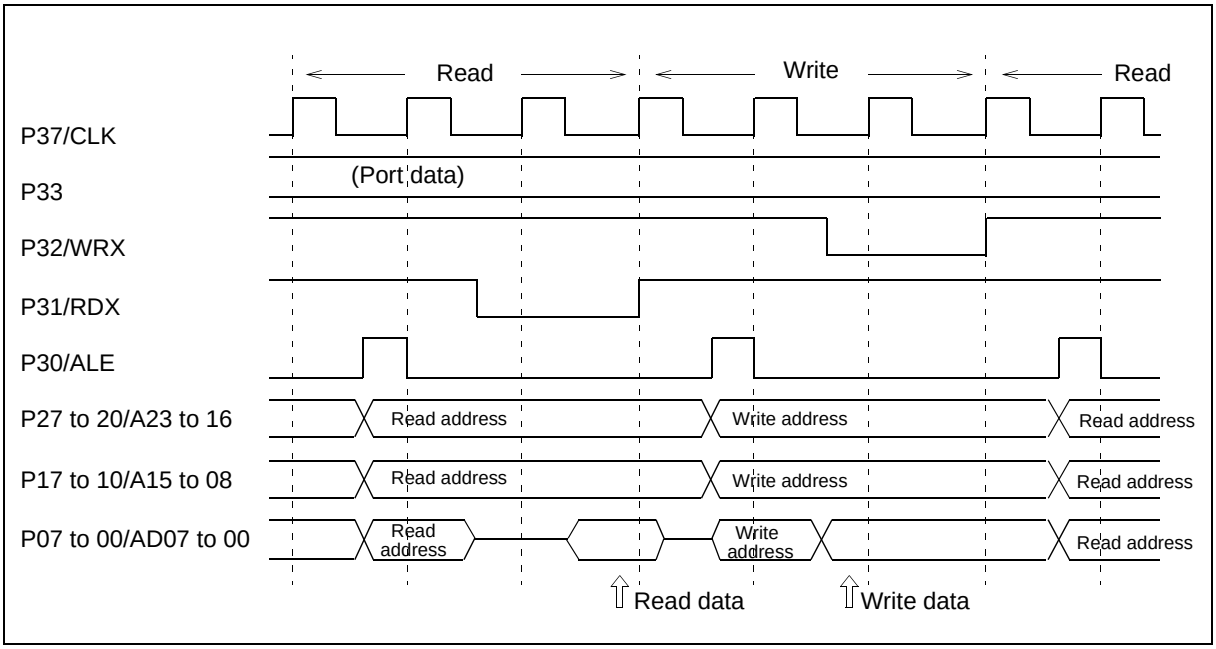
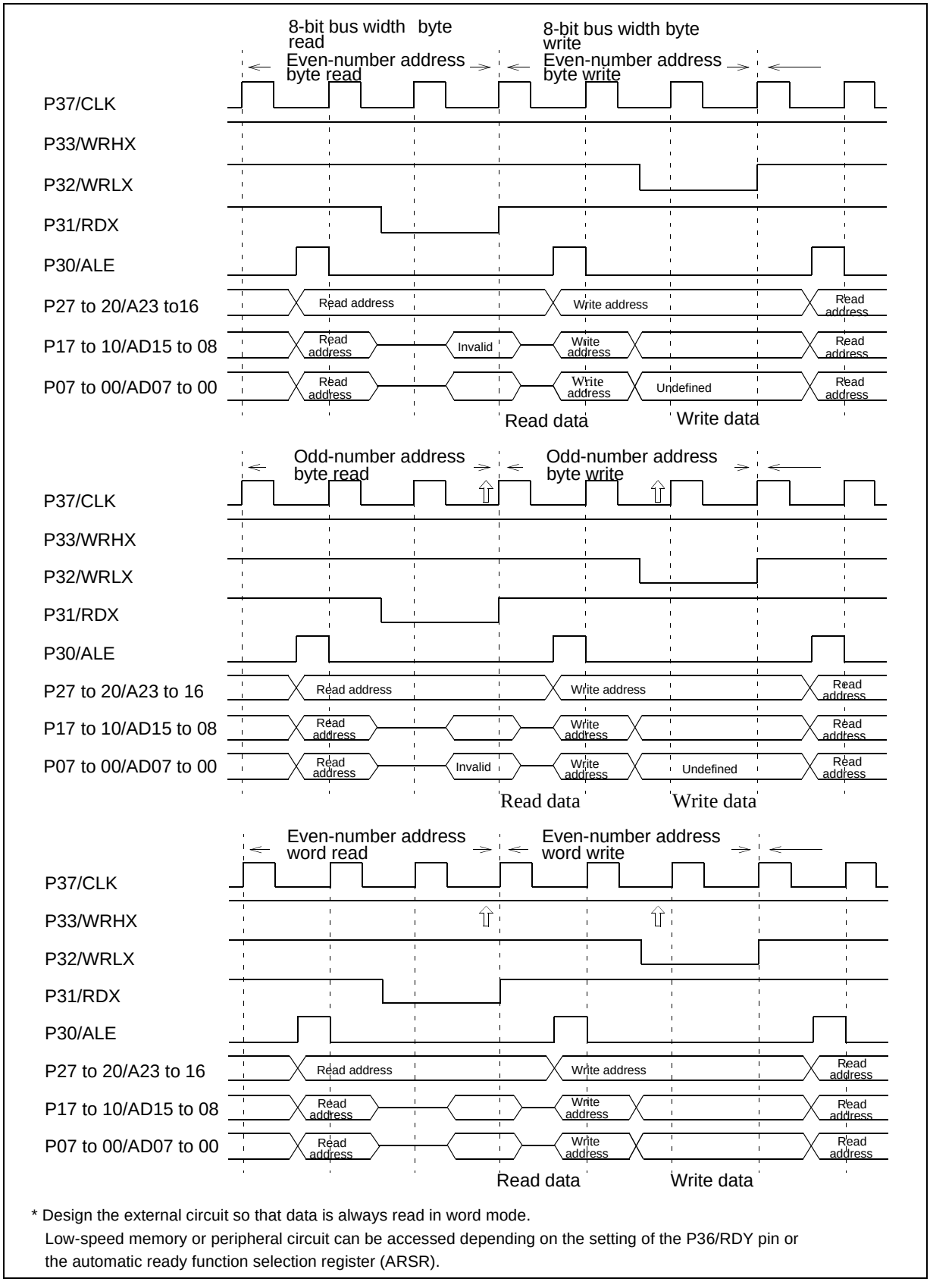


Figure 3.2.1a External memory access timing chart

■ External 16-bit bus mode



* Design the external circuit so that data is always read in word mode.
Low-speed memory or peripheral circuit can be accessed depending on the setting of the P36/RDY pin or the automatic ready function selection register (ARSR).

Figure 3.2.1b External memory access timing chart

(2) Ready function

When the RYE bit of the bus control signal selection register (EPCR) is set to '1,' a wait cycle is inserted while an L level signal appears at the R36/RDY pin in the event of an access to an external area. Thus, the access cycle can be extended.

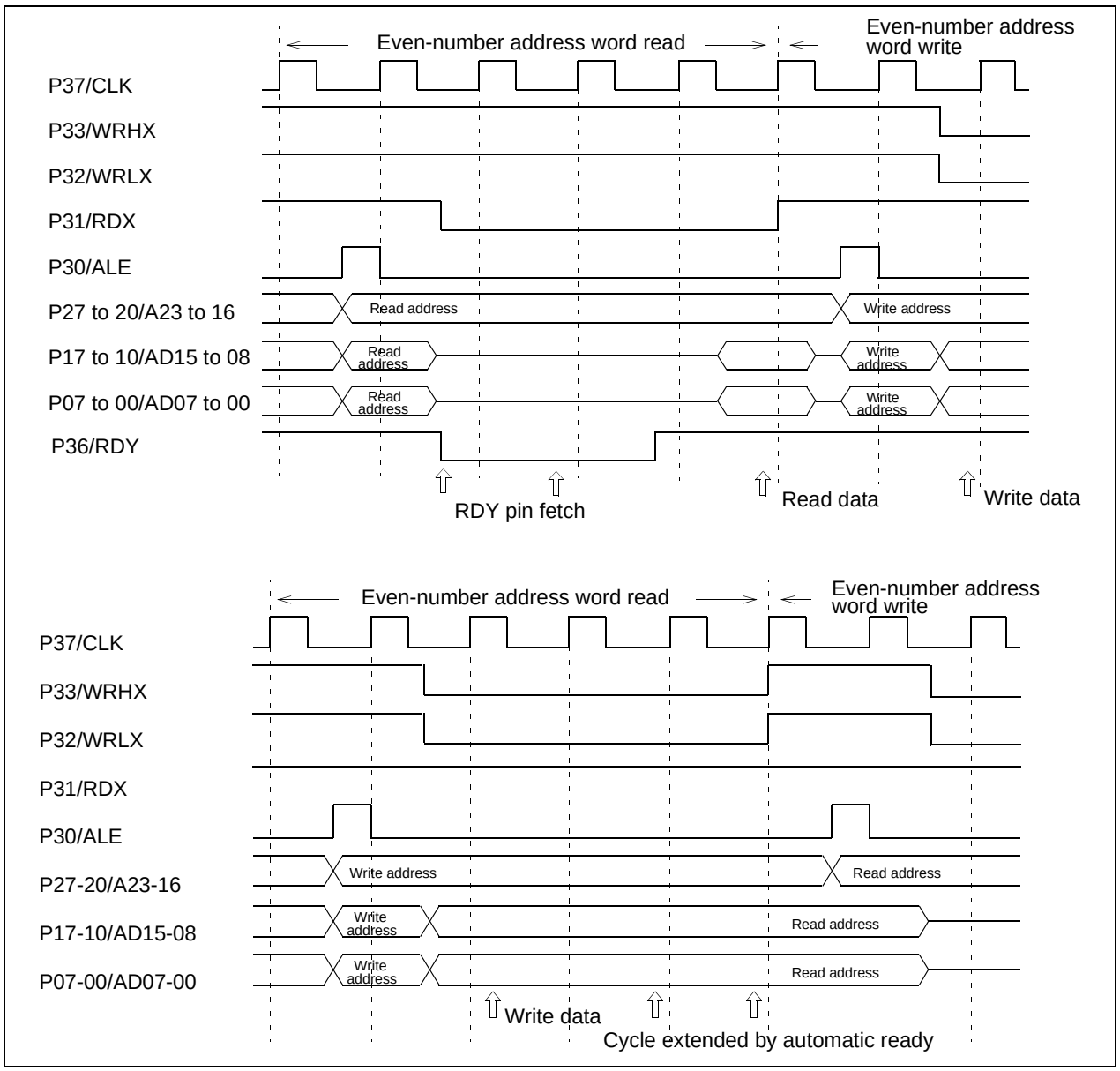


Figure 3.2.1c Ready timing chart

The F^{2}MC-16LX has two types of automatic ready functions for external memory. The automatic ready function automatically inserts one to three wait cycles without an external circuit under the following conditions. The function inserts the wait cycles when an access is made to a low-order address external area located between addresses 002000_H and $7FFFFF_H$ or to a high-order address external area located between addresses 800000_H and $FFFFFF_H$. This function extends the access cycle. The automatic ready function is activated by setting the LMR1/LMR0 bits (low-order address external area) or the HMR1/HMR0 bits (high-order address external area) of ARSR. In addition, the F^{2}MC-16LX has another built-in automatic ready function for external I/O. This automatic ready function automatically inserts one to three wait cycles without an external circuit when an access is made to an external area located between addresses $0000C0_H$ and $0000FF_H$. This function extends the access cycle. This automatic ready function is activated by setting the IOR1/IOR0 bits of ARSR.

When the RYE bit of EPCR is set to ’1,’ the wait cycle continues if an L level signal appears at the R36/RDY pin at the end of either automatic ready cycle.

(3) Hold function

When the HDE bit of EPCR is set to ’1,’ the external bus hold function by the P34/HRQ and P35/HAKX pins is enabled. When an H level signal is input to the P34/HRQ pin, the hold state starts at the end of the CPU instruction (at the end of processing for one element data item in the case of a string instruction), an L level signal is output from the P35/HAKX pin, and the following pins are set to high impedance:

- Address output P27/A23 to P20/A16
- Address/data I/O P17/D15 to P00/D00
- Bus control signal P30/ALE, P31/RDX, P32/WRLX, P33/WRHX

The above function enables the use of an external bus by a device external circuit.

When an L level signal is input to the P34/HRQ pin, an H level signal is output from the P35/HAKX pin, the external pin status is restored, and the CPU resumes operation.

In the STOP state, no hold request input is accepted.

■ Hold timing (in external bus 16-bit mode)

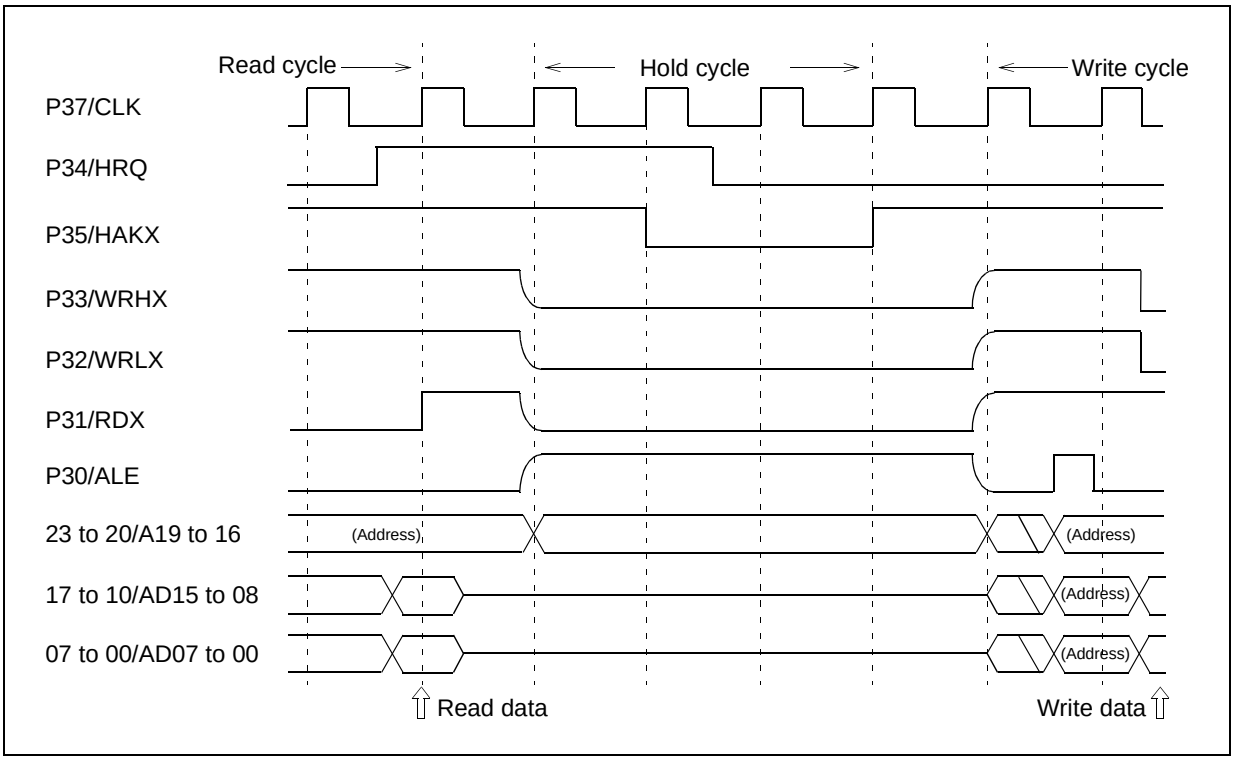


Figure 3.2.1d Hold timing

Chapter 4:
Clock and Reset

4.1 Clock Generator

The clock generator controls internal clock operation, including such functions as sleep, timer, stop, and PLL multiplication. This internal clock is called the machine clock, and one cycle of the machine clock is called a machine cycle. A clock based on the source oscillation is called the main clock, and a clock based on the internal VCO oscillation is called the PLL clock.

Note: When the operating voltage is 5 V, the OSC source oscillation can be between 3 MHz and 16 MHz. The highest operating frequency for the CPU and peripheral resource circuits is 16 MHz, however. Normal operation is not guaranteed if a multiplication factor resulting in a higher frequency than 16 MHz is specified. For example, if the source oscillation is 16 MHz, only 1 can be specified as the multiplication factor.

The lowest operating frequency of the VCO oscillation is 4 MHz, and an oscillation below 4 MHz must not be specified.

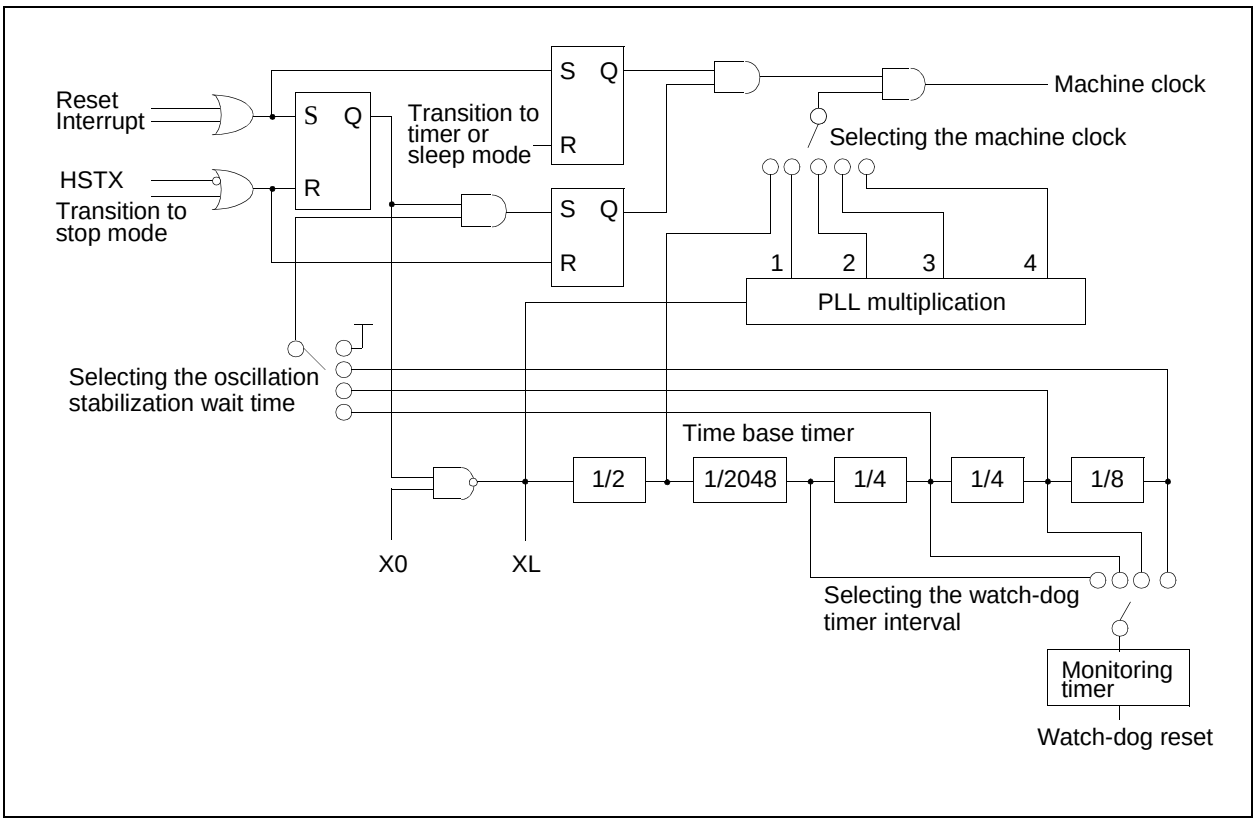


Figure 4.1a Clock generator circuit block diagram

4.2 Reset Causes

When a reset cause occurs, F^{2}MC-16LX terminates the currently executing processing and waits for the release of reset signal. A reset can be caused by the following factors:

- ❍ Power-on reset
- ❍ Hardware standby release
- ❍ Watch-dog timer overflow
- ❍ External reset request via RSTX pin
- ❍ Reset request by software

Right after stop mode release or power on reset, the MCU will wait for the stabilization time before resumption of any activities.

When reset occurs, F^{2}MC-16LX will stop all operation at once and wait for the release of reset.

The content of watchdog timer control register will change according to the reset cause. Thus, the cause of previous reset can be known.

Note: While an external bus is used, the address generated by the device is undefined when a reset cause occurs. All external bus access signals, including RDX and WRX, become inactive.

Table 4.2a Reset causes

Reset	Cause	Machine clock	Watch-dog timer	Oscillation stabilization wait
Power-on	When the power is turned on	Main clock	Stop	Yes
Hardware standby	'L' level input to HSTX pin	Main clock	Stop	Yes
Watch-dog timer	Watch-dog timer overflow	Main clock	Stop	Yes
External pin	'L' level input to RSTX pin	Previous status maintained	Previous status maintained	No
Software	'0' written to RST bit of STBYC	Previous status maintained	Previous status maintained	No

* In stop or hardware standby mode, a reset input allows for oscillation stabilization time regardless of the reset cause.
* The oscillation stabilization time for a power-on reset is fixed to 2^{18} cycles of source oscillation. For other types of reset, the oscillation stabilization wait time is determined by CS1 and CS0 of the clock selection register.

As shown in Figure 4.2a, each reset cause has a corresponding flip-flop. The contents of the flip-flop can be obtained by reading the watch-dog timer control register. If identifying the reset cause is required after the reset is released, ensure that the value read from the watch-dog timer control register is processed by software and processing branches to an appropriate program.

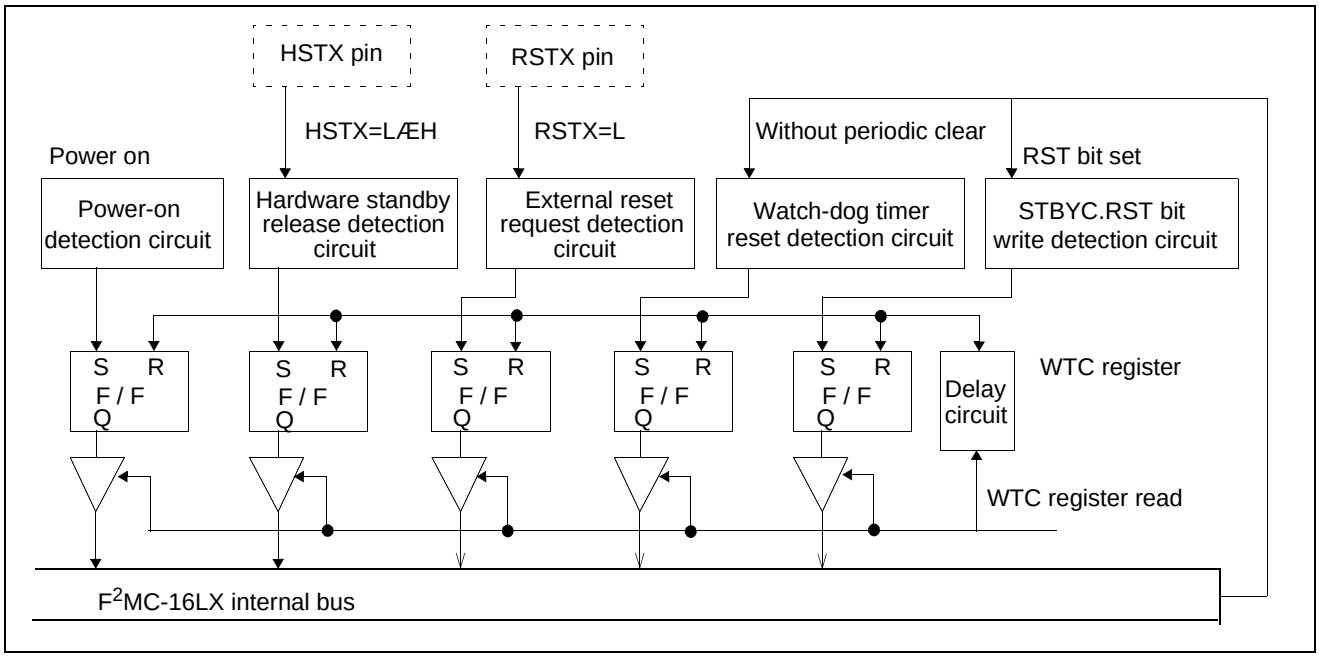


Figure 4.2a Reset cause bit block diagram

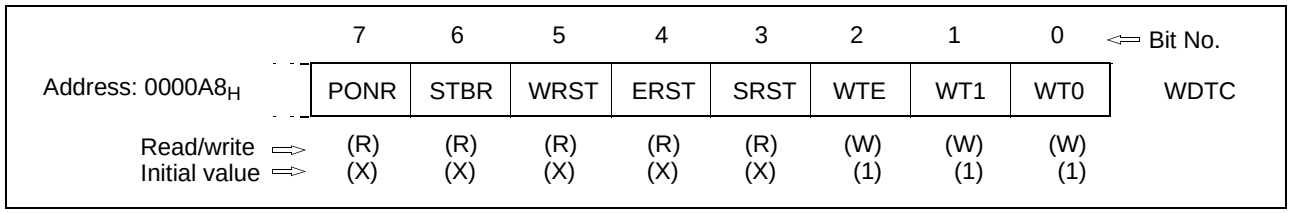


Figure 4.2b WDTC (watch-dog timer control register)

When there are multiple reset causes, the corresponding reset cause bits in the watch-dog timer control register are set. Therefore, if an external reset request and a watch-dog reset occur at the same time, both the ERST and WRST bits are set to 1.

A power-on reset is an exception; while the PONR bit is 1, the values of other bits do not indicate the correct reset causes. Therefore, design software so that the other reset cause bit values are ignored while the PONR bit is set to 1.

Table 4.2b Reset cause bits

Reset cause	PONR	STBR	WRST	ERST	SRST
Power-on	1	———	———	———	———
Hardware standby	*	1	*	*	*
Watch-dog timer	*	*	1	*	*
External pin	*	*	*	1	*
RST bit	*	*	*	*	1

(An asterisk (*) in the table means that the previous value is maintained.)

Note: A reset cause bit is cleared only by reading the watch-dog timer control register. Thus, once a reset occurs, the corresponding reset cause bit remains 1 even if another reset cause occurs.

4.3 Operation after reset release

When a reset cause is removed, the F^{2}MC-16LX immediately outputs the address in which the reset vector is stored, then fetches the reset vector and mode data. The reset vector and mode data are assigned to the four bytes between FFFFDC$_H$ and FFFFDF$_H$. After reset is released, the reset vector and mode data are transferred to the registers by the hardware as described in Figure 4.3a.

Use the mode pin to specify whether to read the reset vector and mode data from internal ROM or from external memory. When the mode pin is set to external vector mode, the F^{2}MC-16LX reads the reset vector and mode data from external memory. When using the F^{2}MC-16LX in single chip mode or internal ROM external bus mode, Fujitsu recommends specifying internal vector mode.

The bus mode after the reset vector and mode data are read is specified by the mode data.

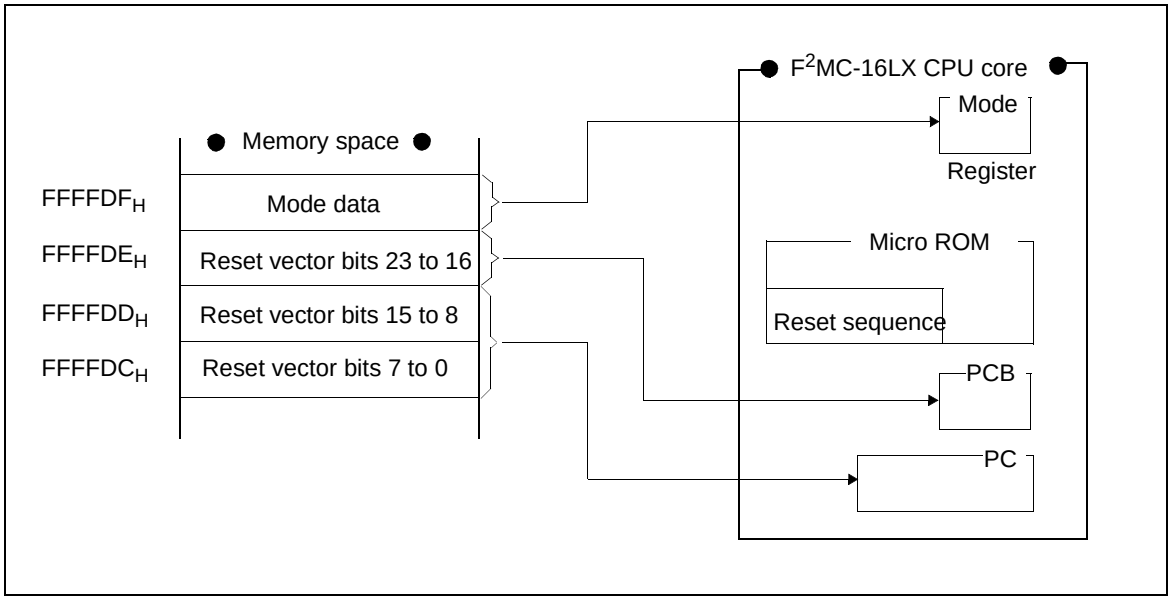


Figure 4.3a Source and destination of reset vector and mode data

Chapter 5:
Watchdog Timer, Timebase Timer, and Watch Timer Functions

5.1 Outline

Watch-Dog Timer

The watchdog timer consists of 2-bit counter that uses to carry signal from the 18-bit timebase timer or the 15-bit watch timer as a clock source, a control register, and a watchdog reset controller. The watch-dog timer function enables detection of program surge. If the watch-dog timer is not accessed within the specified time due to, for example, a program surge, the watch-dog timer resets the system.

Time Base Timer

The timebase timer consists of 18-bit counter and a circuit that control interval interrupts. The timebase timer also has a function for supplying operating clocks for the timer output for the oscillation stabilization time or the watchdog timer etc.it is counting up in synchronization to the internal count clock (divided-by-2 of oscillation) with an interval timer function for selecting an interval time from four types of 2^{12}/HCLK, 2^{14}/HCLK, 2^{16}/HCLK, and 2^{19}/HCLK. (HCLK is the main clock.) Note taht the timebase timer uses the main clock, regardless of the MCS bit and SCS bit in CKSCR.

Watch Timer

The watch timer consists of 15-bit counter and a circuit that controls interval interrupts. The watch timer functions as the clock source for the watchdog counter, as the timer for the subclock stabilization wait, and as an interval timer that generates interrupts at a given period. Note that the watch timer uses the sub clock, regardless of the MCS bit and SCS bit in CKSCR.

5.2 Block diagram

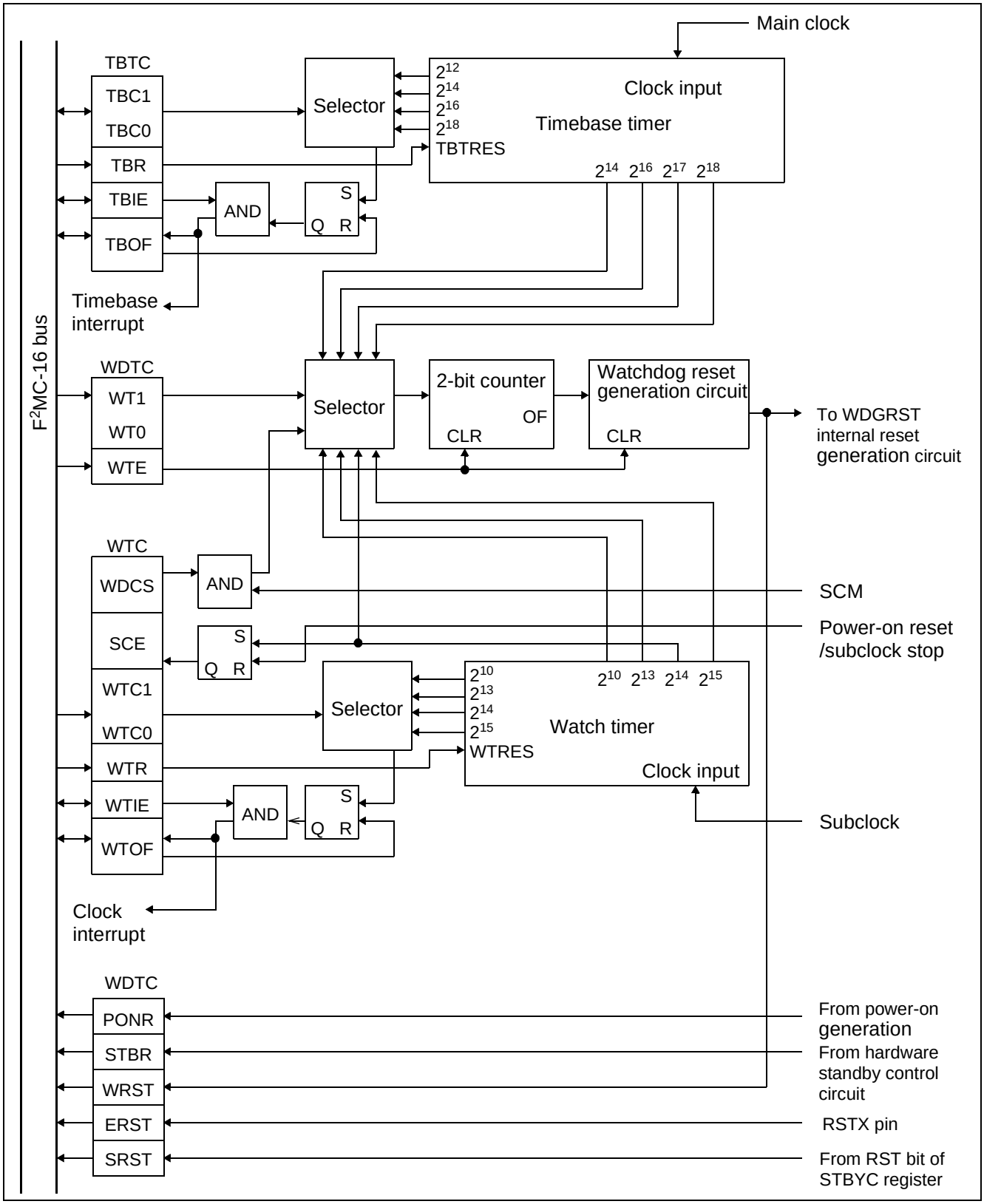


Figure 5.2a Watchdog Timer, Timebase Timer, and Watch Timer Block Diagram

5.3 Registers and register details

Watch-Dog timer control register

	7	6	5	4	3	2	1	0	
	⇦ Bit number								
Address : 0000A8$_H$	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	WDTC
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(W)	(W)	(W)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(1)	(1)	(1)	

Timer base timer control register

	15	14	13	12	11	10	9	8	
	⇦ Bit number								
Address: 0000A9$_H$	Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0	TBTC
Read/write ⇨	(-)	(-)	(-)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(-)	(-)	(0)	(0)	(1)	(0)	(0)	

Watch timer control register

	7	6	5	4	3	2	1	0	
	⇦ Bit number								
Address: 0000AA$_H$	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0	WTC
Read/write ⇨	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(X)	(0)	(0)	(0)	(0)	(0)	(0)	

5.3.1 WDTC (Watch-Dog Timer Control Register)

Watch-Dog timer control register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 0000A8$_H$	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	WDTC
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(W)	(W)	(W)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(1)	(1)	(1)	

Don’t use read-modify-write command to access this register, otherwise malfunction will occur.

[bits 7 to 3] PONR, STBR, WRST, ERST, and SRST

These flags indicate the reset causes. The flags are set upon a reset as described in Table 5.3.1a.

All bits are cleared to '0' after the WDTC register is read. The WDTC register is a read-only register. This is a read-only register. Note that during power-on only, the contents of the bits that indicate sources other than power-on are not guaranteed. Therefore, software should be designed to ignore the other bits when the PONR bit is "1".

Table 5.3.1a Reset cause registers

Reset cause	PONR	STBR	WRST	ERST	SRST
Power-on	1	—	—	—	—
Hardware standby	*	1	*	*	*
Watch-dog timer	*	*	1	*	*
External pin	*	*	*	1	*
RST bit	*	*	*	*	1

(*: The previous value is maintained.)

[bit 2] WTE

While the watch-dog timer is stopped, writing '0' to this bit activates the watch-dog timer. Subsequently, writing '0' clears the watch-dog timer counter. Writing '1' has no effect.

The watch-dog timer is stopped by power-on, hardware standby, or reset by watch-dog timer. '1' is always read from this bit.

[Bits 1, 0] WT1, WT0

These bits select the watchdog interval time. Only the data written when the watchdog timer is started up is valid. Data written to these bits at any time other than watchdog startup is ignored. Note that the clock that is input to the watchdog timer is selected according to the result of ANDing the WDCS bit of the WTC and the SCM bit of the LPMCR. In other words, if WDCS is set to "1", then the timebase timer output can be selected if the main clock and the PLL clock are selected, and the watch timer output can be selected if the subclock is selected.

The interval time settings are shown in Table 5.3.1a.

These bits are write-only bits.

Table 5.3.1b Watchdog Timer Interval Selection Bits

WDCS/ SCM	WT1	WT0	Interval Time (Source oscillation: 4 MHz) Minimum	Maximum
1	0	0	Approx. 3.58 ms	Approx. 4.61 ms
1	0	1	Approx. 14.33 ms	Approx. 18.43 ms
1	1	0	Approx. 57.23 ms	Approx. 73.73 ms
1	1	1	Approx. 458.75 ms	Approx. 589.82 ms
0	0	0	Approx. 0.109 s	Approx. 0.141 s
0	0	1	Approx. 0.875 s	Approx. 1.125 s
0	1	0	Approx. 1.75 s	Approx. 2.25 s
0	1	1	Approx. 3.5 s	Approx. 4.5 s

Note: The maximum interval value is the value when the time base counter or the clock counter are not reset during watchdog operation.

5.3.2 TBTC (Time Base Timer Control Register)

Timer base timer control register

	15	14	13	12	11	10	9	8	⇦ Bit number
Address: 0000A9$_H$	Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0	TBTC
Read/write ⇨	(-)	(-)	(-)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(-)	(-)	(0)	(0)	(1)	(0)	(0)	

Note: Don't use read-modify-write command to access this register, otherwise malfunction will occur.

[bit 15] Reserved

This is a reserved bit. When writing data to this register, ensure that '1' is written to this bit.

[bit 12] TBIE

This bit is used to enable interval interrupts based on the time base timer. Writing '1' to this bit enables interrupts, and writing '0' disables interrupts. This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 11] TBOF

This is an interrupt request flag for the time base timer. While the TBIE bit is '1,' an interrupt request is issued when '1' is written to TBOF. This bit is set to '1' for each interval specified with the TBC1 and TBC0 bits.

This bit is cleared by writing '0,' by switching to stop or hardware standby mode, or by a reset. Writing '1' has no effect.

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

[bit 10] TBR

This bit clears all bits of the time base timer counter to '0.'

Writing '0' clears the time base counter.

Writing '1' has no effect.

'1' is always read from this bit.

Note: Time base timer interrupt should be masked by either TBIE bit or ILM bit of CPU before clearing the TBOF bit.

[bits 9 and 8] TBC1 and TBC0

These bits are used to set the time base timer interval.

Table 5.3.2a Selecting the time base timer interval

TBC1	TBC0	Interval at 4 MHz source oscillation	Machine Clock Cycle
0	0	1.024 ms	2^{12} cycle
0	1	4.096 ms	2^{14} cycle
1	0	16.384 ms	2^{16} cycle
1	1	131.072 ms	2^{19} cycle

5.3.3 Watch Timer Control Register (WTC)

Watch timer control register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address: 0000AA$_H$	WDCS	SCE	WTIE	WTOF	WTR	WTC2	WTC1	WTC0	WTC
Read/write ⇨	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(X)	(0)	(0)	(0)	(0)	(0)	(0)	

[Bit 7] WDCS

This bit selects whether to use the clock signal from the watch timer or from the timebase timer for the watchdog timer input clock when the main clock and PLL clock are selected. When this bit is "0", the clock signal from the watch timer is selected; when this bit is "1", the clock signal from the timebase timer is selected. In short, if WDCS is set to "1", then the timebase timer output can be selected if the main clock and the PLL clock are selected, and the watch timer output can be selected if the subclock is selected.

This bit is initialized to "1" by a power-on reset.

Note: When WDCS is set to "1", because the timebase timer output and the watch timer output are asynchronous, there is a possibility that the watchdog timer count may advance. Therefor, when WDCS is set to "1", it is necessary to clear the watchdog timer before and after changing the clock mode.

[Bit 6] SCE

This bit indicates that the subclock oscillation stabilization waiting period has elapsed. When this bit is "0", it indicates that the oscillation stabilization period is currently in progress. The oscillation stabilization period is fixed at 2^{14}cycles (subclock). This bit is initialized to "0" by a power-on reset and by stopping.

[Bit 5] WTIE

This bit enables interval interrupts by the watch timer. When this bit is set to "1", interrupts are enabled; when set to "0", interrupts are disabled. This bit is initialized to "0" by a reset. This bit can be read and written.

[Bit 4] WTOF

This bit is the watch timer interrupt request flag. When the WTIE bit is "1", an interrupt request is generated if WTOF is set to "1". This bit is set to "1" at the intervals set by the WTC1 and WTC0 bits. This bit is cleared by writing a "0", by switching to stop mode or hardware standby mode, and by a reset. Writing "1" to this bit has no meaning.

When this bit is read by a read-modify-write instruction, a "1" is read.

[Bit 3] WTR

This bit clears all of the watch timer counter bits to "0". The clock counter is cleared by writing a "0" to this bit. Writing "1" to this bit has no meaning. Reading this bit returns a "1".

[Bits 2, 1, 0] WTC2, WTC1, WTC0

These bits set the watch timer interval. The interval settings are shown in Table 5.3.3a. These bits are initialized to "000" by a reset. These bits can be read and written.

When writing these bits, clear bit 4 (WTOF) at the same time.

Table 5.3.3a Watch Timer Interval Selection

WTC2	WTC1	WTC0	Interval time when subclock is 32 kHz
0	0	0	15.625 ms
0	0	1	31.25 ms
0	1	0	62.5 ms
0	1	1	0.125 s
1	0	0	0.250 s
1	0	1	0.500 s
1	1	0	1.000 s
1	1	1	–

5.4 Operation

5.4.1 Watch-Dog Timer

The watch-dog timer function enables detection of program surge.

If the watch-dog timer is not accessed within the specified time due to, for example, a program surge, the watch-dog timer resets the system.

(1) Activation

The watch-dog timer is activated by writing '0' to the WTE bit of the WTC register while the watch-dog timer is stopped. At the same time, the WT1 and WT0 bits are used to set the watch-dog timer reset interval. Only the interval setting specified during activation is valid.

(2) Watch-dog counter

Once the watch-dog timer is activated, the watch-dog timer counter must be periodically cleared within the program. Writing '0' to the WTE bit of the WTC register clears the watch-dog counter. The watch-dog counter consists of a two-bit counter which uses the carry signals of the time base counter as a clock source. Therefore, the watch-dog reset time may become shorter than the setting if the time base counter is cleared.

Figure 5.4.1a is a diagram of the watch-dog timer operation.

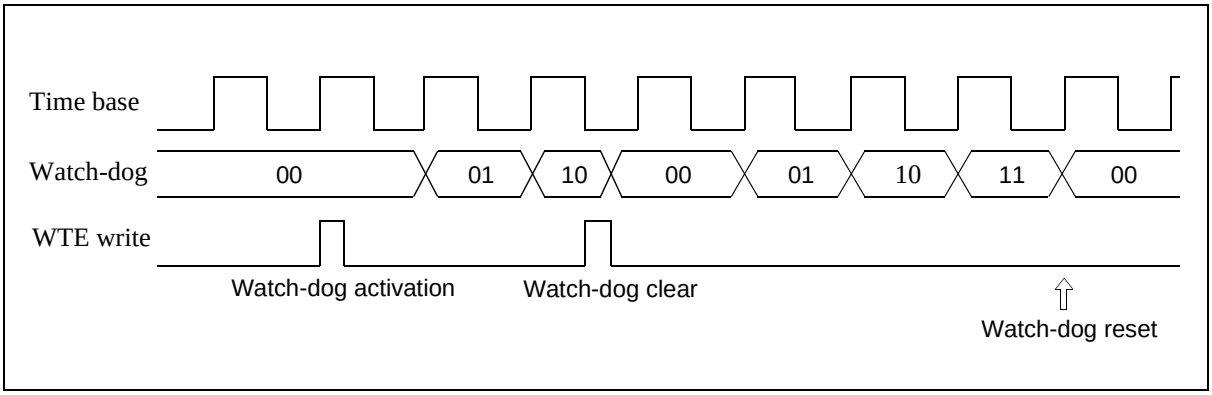


Figure 5.4.1a Watch-dog timer operation

(3) Watch-dog stop

Once activated, the watch-dog timer is initialized and stopped only by power-on, hardware standby, or reset by watch-dog. Reset by an external pin or software merely clears the watch-dog counter without stopping the watch-dog function.

(4) Others

The watch-dog counter is cleared by a reset, transition to sleep or stop mode, or hold acknowledgment signal in addition to writing the WTE bit.

5.4.2 Time Base Timer

The time base timer functions as a watch-dog timer clock source, timer for waiting for the oscillation to stabilize, and interval timer for generating interrupts at specified intervals.

(1) Time base counter

The time base counter consists of an 18-bit counter for a clock generated by dividing the source oscillation input by two. This clock is used to generate the machine clock. While the source oscillation is input, the time base counter keeps counting. The time base counter is cleared by a power-on reset, transition to stop or hardware standby mode, shifting from the main clock to the PLL clock through the setting of the MCS bit in the CKSCR register, shifting from the main clock to the subclock through the setting of the SCS bit in the CKSCR register, or writing '0' to the TBR bit of the TBTC register.

(2) Interval interrupt function

Interrupts are generated at specified intervals according to the carry signals of the time base counter. The TBOF flag is set at the intervals specified with the TBC1 and TBC0 bits of the TBTC register. The flag is written to reference to the time at which the time base timer is cleared last.

If a shift is made from the main clock mode to the PLL clock mode, the timebase timer is cleared, since it is used as the timer for the PLL clock oscillation stabilization waiting period.

In addition, if a shift is made from the main clock mode to the subclock mode, the timebase timer is cleared, since it is used as the timer for the main clock oscillation stabilization waiting period.

Upon transition to stop or hardware standby mode, the time base timer is used as a timer for waiting for the oscillation to stabilize upon recovery. Therefore, the TBOF flag is immediately cleared upon mode transition.

5.4.3 Watch Timer

The watch timer functions as the clock source for the watchdog counter, as the timer for the subclock stabilization wait, and as an interval timer that generates interrupts at a given period.

(1) Watch timer

The watch timer is a 15-bit counter that counts the source oscillation input which is used to generate the machine clock. The watch timer always continues its counting operation as long as the source oscillation is being input. The watch timer is cleared by: power-on reset, shifting to stop mode or hardware standby mode, and writing "0" to the WTR bit in the WTC register.

The watchdog counter and the interval interrupts, both of which utilize the watch timer output, are affected by the watch timer being cleared.

(2) Interval interrupt function

This function generates interrupts at a given period based on the clock counter carry signal. This function sets the WTOF flag at a regular interval, which is set by the WTC1 and WTC0 bits in the WDTC register. The timing for the setting of this flag is based on the time when the watch timer was last cleared.

If a shift is made to stop mode or hardware standby mode, the WTOF flag is cleared at the same time as the mode shift, since the watch timer is used for the oscillation stabilization waiting period during recovery.

Chapter 6:
Low Power Control Circuit

6.1 Outline

The following are the operating modes: PLL clock mode, PLL sleep mode, PLL watch mode, pseudo-watch mode, main clock mode, main sleep mode, main watch mode, main stop mode, subclock mode, sub sleep mode, sub watch mode, sub stop mode, and hardware standby mode. Aside from the PLL clock mode, all of the other operating modes are low power consumption modes.

In main clock mode and main sleep mode, only the main clock (main OSC oscillation clock) and the subclock (sub OSC oscillation clock) operate. In these modes, the main clock divided by two is used as the operation clock, the subclock (sub OSC oscillation clock) is used as the watch clock, and the PLL clock (VCO oscillation clock) is stopped.

In subclock mode and sub sleep mode, only the subclock (sub OSC oscillation clock) operates. The subclock is used as the operation clock, and the main clock and the PLL clock are stopped.

In PLL sleep mode and main sleep mode, only the CPU's operation clock is stopped, all clocks other than the CPU clock operate.

In pseudo-watch mode, only the watch timer and the timebase timer operate.

In PLL watch mode, main watch mode, and sub-watch mode, only the watch timer operates. Only the subclock is in operation in this mode; the main clock and the PLL clock are stopped. (The difference among PLL watch mode, main watch mode, and sub-watch mode is that the operating mode upon recovery from an interrupt is PLL clock mode, main clock mode, or subclock mode, respectively. There are no differences in the watch mode operations.)

The main stop mode, sub stop mode, and hardware standby mode stop oscillation, making it possible to retain data while consuming the least amount of power possible. (The difference between main stop mode and sub stop mode is that the operating mode upon recovery from an interrupt is main clock mode or subclock mode, respectively. There are no differences in the stop mode operations.)

The CPU intermittent operation function intermittently runs the clock supplied to the CPU when accessing registers, on-chip memory, on-chip resources, and the external bus. Processing is possible with lower power consumption by reducing the execution speed of the CPU while supplying a high-speed clock to the on-chip resources.

The PLL clock multiplier can be selected as either 2, 4, 6, or 8 by setting the CS1 and CS0 bits. The selected clock divided by two is used as the machine clock.

The WS1 and WS0 bits can be used to set the main clock oscillation stabilization waiting period for when stop mode and hardware standby mode are released.

6.2 Block Diagram

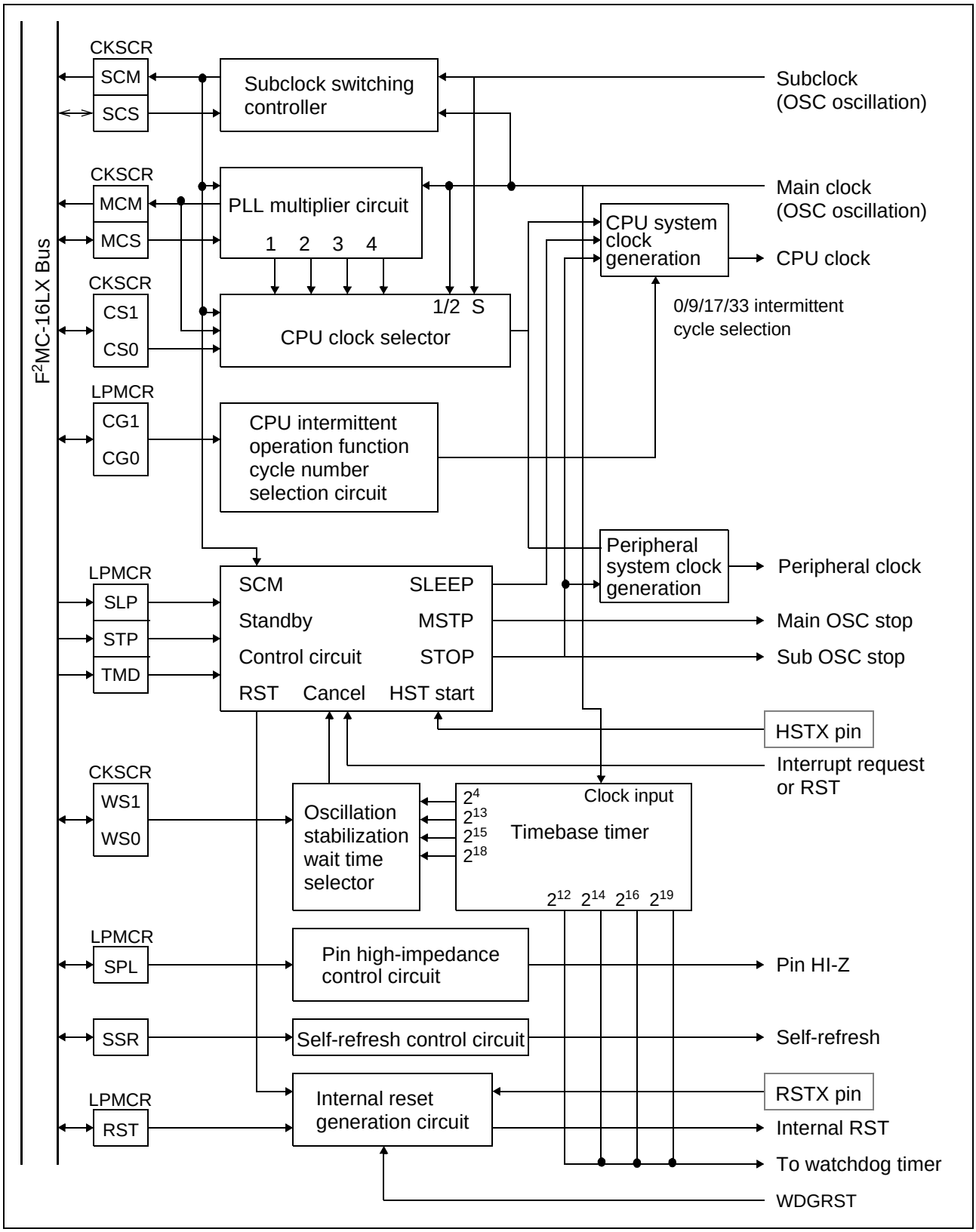


Figure 6.2a Low-power consumption control circuit and clock generator

6.3 Registers and register details

Clock selection register

	15	14	13	12	11	10	9	8	
Address: 0000A1$_H$	SCM	MCM	WS1	WS0	SCS	MCS	CS1	CS0	CKSCR
Read/write	(R)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	

Low power mode control register

	7	6	5	4	3	2	1	0	
Address: 0000A0$_H$	STP	SLP	SPL	RST	TMD	CG1	CG0	SSR	LPMCR
Read/write	(W)	(W)	(R/W)	(W)	(W)	(R/W)	(R/W)	(R/W)	
Initial value	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	

6.3.1 LPMCR (Low power mode control register)

Low power mode control register

	7	6	5	4	3	2	1	0	
Address: 0000A0$_H$	STP	SLP	SPL	RST	TMD	CG1	CG0	SSR	LPMCR
Read/write	(W)	(W)	(R/W)	(W)	(W)	(R/W)	(R/W)	(R/W)	
Initial value	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	

[Bit 7] STP

Writing a "1" to this bit changes the mode to pseudo-watch mode (CKSCR. MCS = 0 and SCS = 1) or stop mode (CKSCR. MCS = 1 or SCS = 0). Writing a "0" to this bit has no effect. This bit is cleared to "0" by a reset, wake-up from watch or stop mode. This bit is a write-only bit. When this bit is read, "0" is always returned.

[Bit 6] SLP

Writing a "1' to this bit changes the mode to sleep mode. Writing a "0" to this bit has no effect. This bit is cleared to "0" by a reset, wake-up from sleep or stop mode.

If a "1' is written to both the STP bit and the SLP bit simultaneously, the mode changes to either pseudo-watch mode or to stop mode. This bit is a write-only bit. When this bit is read, "0" is always returned.

[Bit 5] SPL

When this bit is "0", the level of external pins in watch mode or stop mode is retained. When this bit is "1", the external pins in watch mode or stop mode go to high-impedance. This bit is cleared to "0" by a reset. This bit can be read and written.

[Bit 4] RST

Writing a "0" to this bit generates an internal reset signal in three machine cycles. Writing a "1' to this bit has no effect. When this bit is read, a "1" is returned.

[Bit 3] TMD

Writing a "0" to this bit changes the mode to watch mode. Writing a "1" to this bit has no effect. This bit is set to "1" by a reset, wake-up from watch or stop mode. This bit is a write-only bit. When this bit is read, "1" is always returned.

[Bits 2, 1] CG1, CG0

These bits set the number of clock pause cycles for the CPU intermittent operation function.

These bits are initialized to "00" by a reset due to power-on, hardware standby, or a reset by the watchdog timer. These bits are not initialized by resets due to other sources. These bits can be read or written.

Table 6.3.1a CG Bit Setting

CG1	CG0	Number of CPU clock pause cycles
0	0	0 cycles (CPU clock = resource clock)
0	1	9 cycles (CPU clock: resource clock = 1: approximately 3 to 4)
1	0	17 cycles (CPU clock: resource clock = 1: approximately 5 to 6)
1	1	33 cycles (CPU clock: resource clock = 1: approximately 9 to 10)

[Bit 0] SSR

When this bit is set to "1", DRAMC self-refresh control is performed in sleep (main/PLL) mode, watch mode, and stop mode. This bit is cleared to "0" by a refresh. This bit can be read and written.

Note: SSR has no function if there is no DRAMC on chip.

6.3.2 CKSCR (Clock selection register)

Clock selection register

	15	14	13	12	11	10	9	8 ⇐ Bit No.
Address: 0000A1$_H$	SCM	MCM	WS1	WS0	SCS	MCS	CS1	CS0 CKSCR
Read/write ⇨	(R)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value ⇨	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)

[Bit 15] SCM

This bit indicates whether the main clock or the subclock is selected as the machine clock. When this bit is "0", it indicates that the subclock is selected; when this bit is "1", it indicates that the main clock is selected. If SCS = 0 and SCM = 1, it indicates that the main clock oscillation stabilization waiting period is in progress.

[Bit 14] MCM

This bit indicates whether the main clock or the PLL clock is selected as the machine clock. When this bit is "0", it indicates that the PLL clock is selected; when this bit is "1", it indicates that the main clock is selected. If MCS = 0 and MCM = 1, it indicates that the PLL clock oscillation stabilization waiting period is in progress. Note that the PLL clock oscillation stabilization waiting period is fixed at 2^{12} main clock cycles.

[Bits 13, 12] WS1, WS0

These bits set the main clock oscillation stabilization waiting period upon wake-up from stop mode or hardware standby mode is released.

These bits are initialized to "11" by a power-on reset; these bits are not initialized by a reset due to other sources. These bits can be read and written.

Table 6.3.2a WS Bit Settings

WS1	WS0	Oscillation stabilization waiting period (source oscillation at 4 MHz)
0	0	No oscillation stabilization waiting period
0	1	Approx. 1.02 ms (count of 2^{14} of the source oscillation)
1	0	Approx. 8.19 ms (count of 2^{16} of the source oscillation)
1	1	Approx. 65.54 ms (count of 2^{18} of the source oscillation)

[Bit 11] SCS

This bit selects either the main clock or the subclock as the machine clock. When a "0" is written to this bit, the subclock is selected; when a "1" is written to this bit, the main clock is selected. If a "1" is written to this bit while it is "0", the oscillation stabilization waiting period for the main clock is generated; therefore, the timebase timer is automatically cleared. In addition, the subclock (as is) is used for the operation clock when the subclock is selected. (When the source oscillation is 32 kHz, the operation clock is 32 KHz.) When SCS and MCS are both set to "0", SCS takes priority and the subclock is selected.

This bit is initialized to "1" by a reset due to power-on, hardware standby, the watchdog timer, an external source, or software.

[Bit 10] MCS

This bit selects either the main clock or the PLL clock as the machine clock. When a "0" is written to this bit, the PLL clock is selected; when a "1" is written to this bit, the main clock is selected. If a "0" is written to this bit while it is "1", the oscillation stabilization waiting period for the PLL clock is generated; therefore, the timebase timer is automatically cleared. Note that the PLL clock oscillation stabilization waiting period is fixed at 2^{12} main clock cycles. In addition, the main clock divided by two is used for the operation clock when the main clock is selected. (When the source oscillation is 4 MHz, the operation clock is 2 MHz.)

This bit is initialized to "1" by a reset due to power-on, hardware standby, or the watchdog timer.

[Bits 9, 8] CS1, CS0

These bits select the PLL clock multiplier. These bits are not initialized by a reset initiated by an external pin or the RST bit. These bits are initialized to "00" by a reset due to power-on, hardware standby, and the watchdog timer.

Writing to these bits is suppressed when the MCS bit is "0". Set the MCS bit to "1" (main clock mode) first and then overwrite the CS bits.

These bits can be read and written.

Table 6.3.2b CS Bit Settings

CS1	CS0	Machine clock (source oscillation at 4 MHz)
0	0	4 MHz (operation frequency = OSC oscillation frequency)
0	1	8 MHz (operation frequency = OSC oscillation frequency × 2)
1	0	12 MHz (operation frequency = OSC oscillation frequency × 3)
1	1	12 MHz (operation frequency = OSC (3 MHz) × 4)

6.4 Operations

The status of each chip block in each operating mode is shown in Table 6.4a

Table 6.4a Low Power Consumption Mode Operating Statuses

	Transition condition	Sub oscillation	Main oscillation	Clock	CPU	Peripherals	Pins	Exit method
Subclock	SCS=0 MCS=x	Operating	Stopped	Operating	Operating	Operating	Operating	Reset Interrupt
Sub sleep	SCS=0 MCS=x SLP=1	Operating	Stopped	Operating	Stopped	Operating	Operating	Reset Interrupt
Main sleep	SCS=1 MCS=1 SLP=1	Operating	Operating	Operating	Stopped	Operating	Operating	Reset Interrupt
PLL sleep	SCS=1 MCS=0 SLP=1	Operating	Operating	Operating	Stopped	Operating	Operating	Reset Interrupt
Pseudo-watch (SPL=0)	SCS=1 MCS=0 STP=1	Operating	Operating	Stopped	Stopped	Stopped	Main-tained	Reset Interrupt
Pseudo-watch (SPL=1)	SCS=1 MCS=0 STP=1	Operating	Operating	Stopped	Stopped	Stopped	HI-Z	Reset Interrupt
Watch (SPL=0)	SCS=x MCS=x TMD=0	Operating	Stopped	Stopped	Stopped	Stopped	Main-tained	Reset Interrupt
Watch (SPL=1)	SCS=x MCS=x TMD=0	Operating	Stopped	Stopped	Stopped	Stopped	HI-Z	Reset Interrupt
Stop (SPL=0)	MCS=1 or SCS=0 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	Main-tained	Reset Interrupt
Stop (SPL=1)	MCS=1 or SCS=0 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	Reset Interrupt
Hard-ware standby	HSTX=L	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	HSTX=H

6.4.1 Sleep mode

- Transition to sleep mode

  The standby control circuit is set to sleep mode by writing a "1" to the SLP bit, a "1' to the TMD bit, and a "0" to the STP bit in the low power consumption mode control register. In sleep mode, only the clock supplied to the CPU is stopped; in this mode, the CPU stops, but the peripheral circuits continue to operate.

  If an interrupt request is generated when the "1" is written to the SLP bit, the standby control circuit does not go into sleep mode. In this case, if the CPU is not accepting interrupts, the next instruction is executed; if the CPU is accepting interrupts, processing branches immediately to the interrupt processing routine.

  The contents of the accumulator and other dedicated registers, as well as the contents of RAM, are maintained in sleep mode.

- Releasing sleep mode

  The standby control circuit is used for wake-up from sleep mode when a reset signal is input or when an interrupt is generated. If a wake-up from sleep mode was done by a reset source, the device enters the reset state after wake-up from sleep mode is completed.

  If an interrupt request higher than level 7 is generated by a peripheral circuit, etc., while the device is in sleep mode, the standby control circuit is used for wake-up from sleep mode. Once wake-up from sleep mode is completed, the interrupt is handled in the normal manner. If the settings of the I flag, ILM bits, and the interrupt control register (ICR) are all set so that the interrupt is accepted, then the CPU executes interrupt processing. If the settings do not permit the interrupt to be accepted, then processing resumes from the instruction that follows the instruction that put the device into sleep mode.

6.4.2 Pseudo-watch mode

- Transition to pseudo-watch mode

  The standby control circuit is set to pseudo-watch mode by writing a "1" to the SCS bit and a "0" to the MCS bit in the clock selection register, and a "1" to the TMD bit and a "1" to the STP bit in the low power consumption mode control register. In pseudo-watch mode, all clocks stop, except for the source oscillation (main and sub), the watch timer, and the timebase timer. Practically all chip functions cease.

  In addition, the SPL bit in the low power consumption mode control register can be used to control whether I/O pins maintain their previous states or go to high impedance state in pseudo-watch mode.

  If an interrupt request is generated when the "1" is written to the STP bit, the standby control circuit does not shift to pseudo-watch mode.

  The contents of the accumulator and other dedicated registers, as well as the contents of RAM, are hold in pseudo-watch mode.

- Exit from pseudo-watch mode

  The standby control circuit is used for exit from pseudo-watch mode when a reset signal is input or when an interrupt is generated. If an exit from pseudo-watch mode was performed by a reset source, the device enters the reset state after exit from pseudo-watch mode.

  When recovering from pseudo-watch mode, the standby control circuit is activated first for exit from pseudo-watch mode, and then begins waiting for the PLL clock oscillation stabilization wait time to elapse. Therefore, even if the exit from of pseudo-watch mode is due to a reset source, the main clock is used for the reset sequence.

  If an interrupt request higher than level 7 is generated by a peripheral circuit, etc., while the device is in pseudo-watch mode, the standby control circuit is activated for exit from pseudo-watch mode. Once exit from pseudo-watch mode is completed, the interrupt is handled in the normal manner. If the settings of the I flag, ILM bits, and the interrupt control register (ICR) are all set so that the interrupt is accepted, then the CPU executes interrupt processing. If the settings do not permit the interrupt to be accepted, then

processing resumes from the instruction that follows the last instruction that put the device into pseudo-watch mode.

6.4.3 Watch mode

- Transition to watch mode

  The standby control circuit is set to watch mode by writing a "0" to the TMD bit in the low power consumption mode control register. In watch mode, all clocks stop, except for the sub-source oscillation and the watch timer. Practically all chip functions cease.

  In addition, the SPL bit in the low power consumption mode control register can be used to control whether I/O pins maintain their previous states or go to high impedance state in watch mode.

  If an interrupt request is generated when the "1" is written to the TMD bit, the standby control circuit does not shift to watch mode.

  The contents of the accumulator and other dedicated registers, as well as the contents of RAM, are maintained in watch mode.

- Exit from watch mode

  The standby control circuit is used for exit from watch mode when a reset signal is input or when an interrupt is generated. If watch mode was released by a reset source, the device enters the reset state after exit from watch mode.

  When recovering from sub-watch mode, the standby control circuit is activated first for exit from watch mode, and then immediately enters subclock mode. Therefore, even if the wake-up from sub-watch mode is due to a reset source, the sub-clock is used for the reset sequence.

  When recovering from main watch mode or PLL watch mode, the standby control circuit is activated first for exit from watch mode, and then begins waiting for the main clock oscillation stabilization period to elapse. Therefore, even if the exit from watch mode is due to a reset source, the sub-clock is used for the reset sequence.

  If an interrupt request higher than level 7 is generated by a peripheral circuit, etc., while the device is in watch mode, the standby control circuit is activated for exit from watch mode. Once exit from watch mode is completed, the interrupt is handled in the normal manner. If the settings of the I flag, ILM bits, and the interrupt control register (ICR) are all set so that the interrupt is accepted, then the CPU executes interrupt processing. If the settings do not permit the interrupt to be accepted, then processing resumes from the instruction that follows the last instruction that put the device into watch mode.

6.4.4 Stop mode

- Transition to stop mode

  The standby control circuit is set to stop mode by writing a "0" to the SCS bit and a "1" to the MCS bit in the clock selection register, and a "1" to the STP bit in the low power consumption mode control register. In stop mode, all oscillation sources (sub and main) stop. All chip functions cease. As a result, data can be retained with the barest minimum of power consumption.

  In addition, the SPL bit in the LPMCR can be used to control whether I/O pins maintain their previous states or go to high impedance state in stop mode.

  If an interrupt request is generated when the "1" is written to the STP bit, the standby control circuit does not go into stop mode.

  The contents of the accumulator and other dedicated registers, as well as the contents of RAM, are maintained in stop mode.

- Exiting stop mode

  The standby control circuit releases stop mode when a reset signal is input or when an interrupt is generated. If stop mode was released by a reset source, the device enters the reset state after stop mode is released.

  When recovering from sub-stop mode, the standby control circuit first begins waiting for the sub-clock oscillation stabilization waiting period to elapse, and then exits stop mode. Therefore, even if the exit from stop mode is due to a reset source, the reset sequence is executed after the sub-clock oscillation stabilization waiting period elapses.

  When recovering from main stop mode, the standby control circuit first begins waiting for the main clock oscillation stabilization waiting period to elapse, and then exits stop mode. Therefore, even if the exit from stop mode is due to a reset source, the reset sequence is executed after the main clock oscillation stabilization waiting period elapses.

  If an interrupt request higher than level 7 is generated by a peripheral circuit, etc., while the device is in stop mode, the standby control circuit exits stop mode. After exit from sub-stop mode, and after the sub-clock oscillation stabilization waiting period has elapsed, the interrupt is handled in the normal manner. If the settings of the I flag, ILM bits, and the interrupt control register (ICR) are all set so that the interrupt is accepted, then the CPU executes interrupt processing. If the settings do not permit the interrupt to be accepted, then processing resumes from the instruction that follows the last instruction that put the device into stop mode.

  After exit from main stop mode, and after the main clock oscillation stabilization waiting period (specified by the WS1 and WS0 bits in the CKSCR) has elapsed, the interrupt is handled in the normal manner. If the settings of the I flag, ILM bits, and the interrupt control register (ICR) are all set so that the interrupt is accepted, then the CPU executes interrupt processing. If the settings do not permit the interrupt to be accepted, then processing resumes from the instruction that follows the last instruction that put the device into stop mode.

6.4.5 Hardware standby mode

- Transition to hardware standby mode

  By setting the HSTX pin to low level, it is possible to set the standby control circuit to hardware standby mode, regardless of the current status. In hardware standby mode, oscillation stops and all I/O pins go to high impedance as long as the HSTX pin is low, regardless of any other statuses, including resets.

  Although the contents of internal RAM are maintained in hardware standby mode, the accumulator and other dedicated registers are all initialized.

- Waking up from hardware standby mode

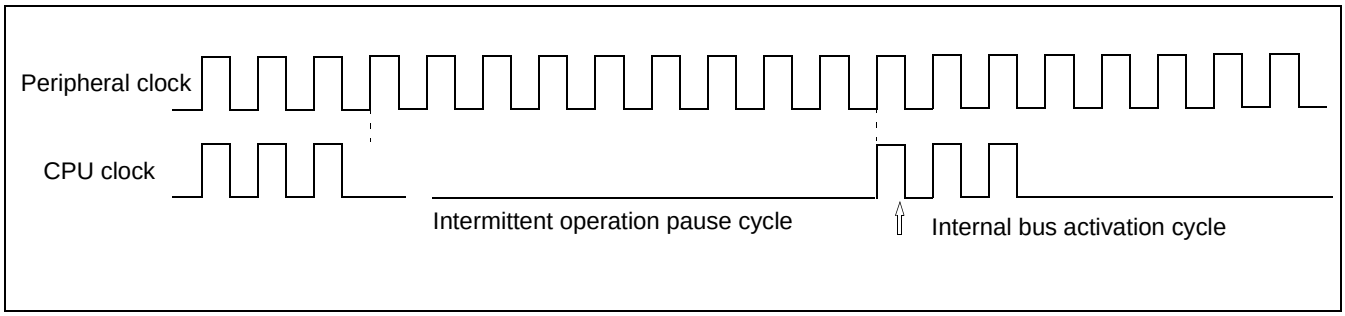
  Wake-up from hardware standby mode can only be executed through the HSTX pin. When the HSTX pin goes high, the standby control circuit is activated for wake-up from hardware standby mode and the device begins waiting for oscillation stabilization after the internal reset signal is enabled. After the main clock oscillation stabilization waiting period elapses, the standby control circuit releases the internal reset, after which the CPU begins execution, starting from the reset sequence.

6.4.6 CPU intermittent operation function

The CPU intermittent operation function regularly stops the clock supplied to the CPU for a given period of time when accessing registers, on-chip memory, on-chip resources, and the external bus, delaying the start of the internal bus cycle. Processing is possible with lower power consumption by reducing the execution speed of the CPU while supplying a high-speed clock to the on-chip resources. The CG1 and CG0 bits select the number of pause cycles in the clock supplied to the CPU.

Note that the same clock is used for external bus operations as for resources.

In addition, the instruction execution time when the CPU intermittent operation function is used can be calculated by adding a compensation factor (the number of register, on-chip memory, on-chip resource, and external bus access multiplied by the number of pause cycles) to the normal execution time.



6.4.7 Setting the main clock oscillation stabilization waiting period

The WS1 and WS0 bits can be used to set the main clock oscillation stabilization waiting period for wake-up from stop mode and hardware standby mode. The oscillation stabilization waiting period should be set in accordance with the type and characteristics of the oscillation circuit and oscillator connected to the X0 and X1 pins.

These bits are not initialized in the event of a reset, except for a power-on reset. If a power-on reset is generated, these bits are initialized to "11". Therefore, when power is first applied, the main clock oscillation stabilization waiting period is set to approximately a count of 2^{18} pulses of the source oscillation.

6.4.8 Switching the machine clock

- Main clock/PLL clock switching

  Switching between the main clock and the PLL clock is accomplished by writing to the MCS bit in the CKSCR register.

  If the MCS bit is overwritten from a "1" to a "0", the machine clock switches from the main clock to the PLL clock, once the PLL clock oscillation stabilization waiting period passes (2^{11}machine clocks).

  If the MCS bit is overwritten from a "0" to a "1", the machine clock switches from the PLL clock to the main clock, at the point when the edges of the PLL clock and the main clock match (after 1 to 8 PLL clocks).

  Because the machine clock does not switch immediately after the MCS bit is overwritten, when performing operations on resources that depend on the machine clock, always reference the MCM bit and make sure that the machine clock was switched before performing the operation on the resource.

- Main clock/sub-clock switching

  Switching between the main clock and the sub-clock is accomplished by writing to the SCS bit in the CKSCR register.

  If the SCS bit is overwritten from a "1" to a "0", the machine clock switches from the main clock to the sub-clock when the sub-clock edge is detected.

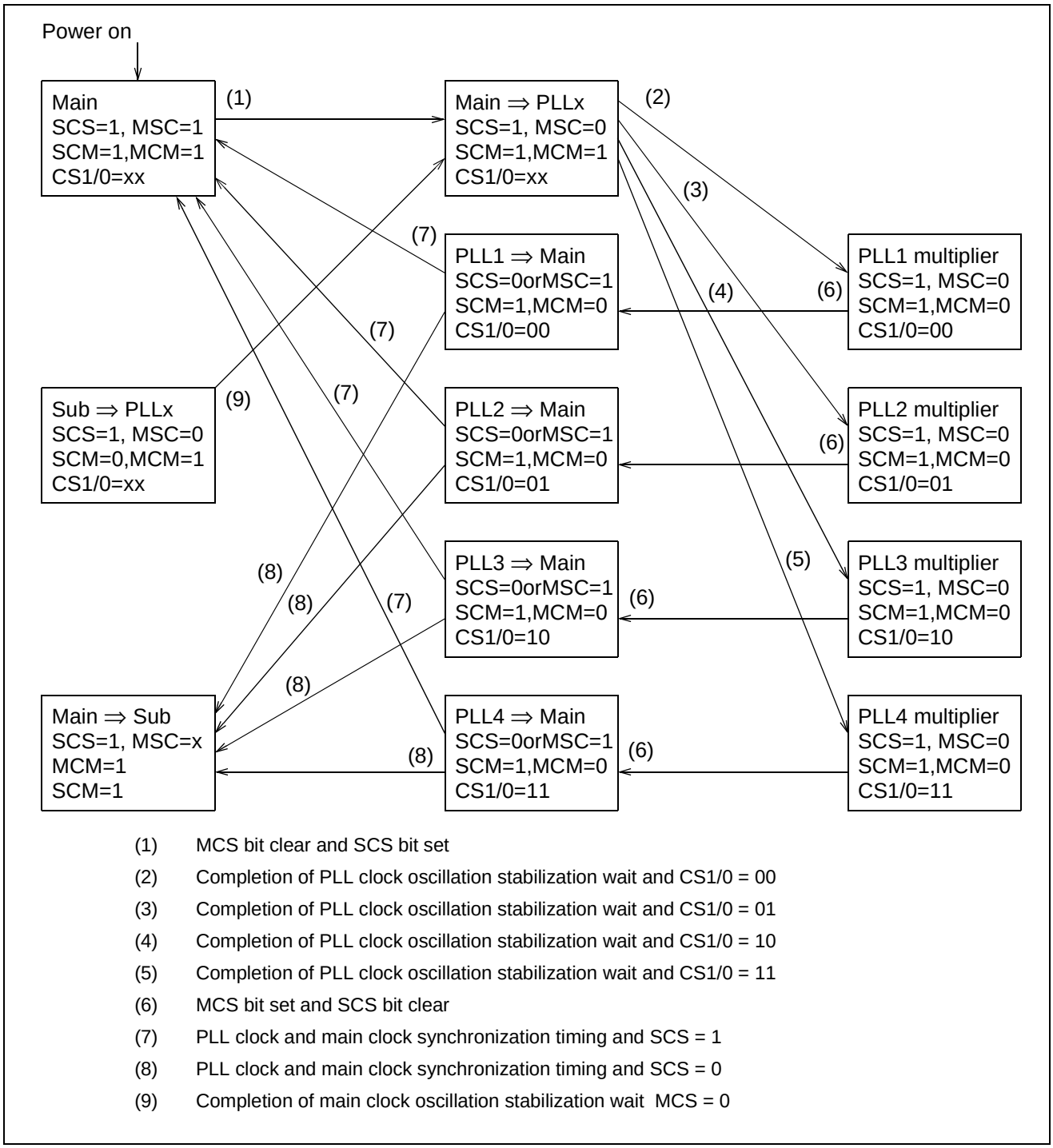
  If the SCS bit is overwritten from a "0" to a "1", the machine clock switches from the sub-clock to the main clock after the main clock oscillation stabilization waiting period elapses.

  Because the machine clock does not switch immediately after the SCS bit is overwritten, when performing operations on resources that depend on the machine clock, always reference the SCM bit and make sure that the machine clock was switched before performing the operation on the resource.

- Machine clock initialization

  The MCS bit and the SCS bit are not initialized by a reset caused by an external pin or the RST bit. After other types of resets, these bits are each initialized to "1".

  Figure 6.4.8a and Figure 6.4.8b show the clock selection state diagram.



(1) MCS bit clear and SCS bit set
(2) Completion of PLL clock oscillation stabilization wait and CS1/0 = 00
(3) Completion of PLL clock oscillation stabilization wait and CS1/0 = 01
(4) Completion of PLL clock oscillation stabilization wait and CS1/0 = 10
(5) Completion of PLL clock oscillation stabilization wait and CS1/0 = 11
(6) MCS bit set and SCS bit clear
(7) PLL clock and main clock synchronization timing and SCS = 1
(8) PLL clock and main clock synchronization timing and SCS = 0
(9) Completion of main clock oscillation stabilization wait MCS = 0

Figure 6.4.8a Clock Selection State Transition Diagram (1)

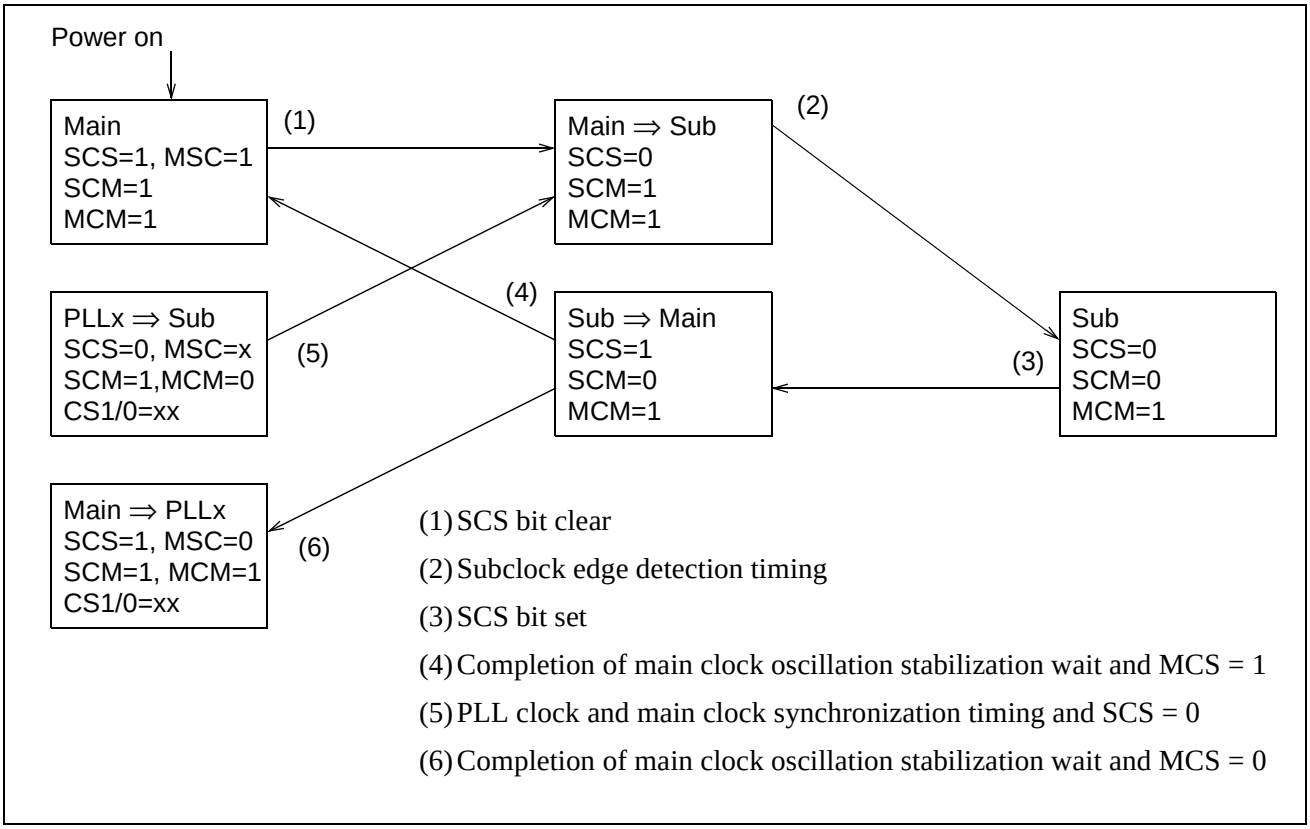


Figure 6.4.8b Clock Selection State Transition Diagram (2)

6.4.9 State transition

Figure 6.4.9a to Figure 6.4.9d show the state transitions in low power consumption mode.

In order to keep the state transition diagrams simple, they depict simultaneously occurring events as occurring in stages. In actuality, however, state transitions occur immediately. For example, when MCS is set to "1" and SLP is set to "1" simultaneously in PLL clock mode, the state transition diagrams show the mode changing once to PM transition mode and then to PM transition sleep, but in actuality, the mode changes immediately from PLL clock mode to PM transition sleep. In addition, when a reset occurs in sub sleep mode, the state transition diagrams show the mode changing once to sub mode and then to the main oscillation stabilization period, but in actuality, the mode shifts immediately from sub sleep mode to the main oscillation stabilization period.

MCS: MCS bit (clock selection register) (PLL clock mode is selected when MCS = 0)

SCS: SCS bit (clock selection register) (sub-clock mode is selected when SCS = 0)

STP: STP bit (low power consumption mode register) (sleep mode is selected when SLP = 0)

SLP: SLP bit (low power consumption mode register) (sleep mode is selected when SLP = 0)

TMD: TMD bit (low power consumption mode register) (watch mode is selected when TMD = 0)

MCM: MCM bit (clock selection register) (PLL clock is in use when MCM = 0)

SCM: SCM bit (clock selection register) (sub-clock is in use when SCM = 0)

SCD: Sub-clock oscillation stopped (sub-clock oscillation is stopped when SCD = 1)

MCD: Main clock oscillation stopped (main clock oscillation is stopped when MCD = 1)

PCD: PLL clock oscillation stopped (PLL clock oscillation is stopped when PCD = 1)

Table 6.4.9a List of Transition Conditions

State before transition	Transition conditions	State after transition
Power on	01 Main oscillation stabilization waiting period completed	Main mode
Main oscillation stabilization	05 Main oscillation stabilization waiting period completed	Main mode
Main mode	06 SCS = 0 written	MS transition mode
	07 SCS = 1•MCS = 0 written	MP transition mode
	31 TMD = 1•STP = 0•SLP = 1 written	Main sleep
	32 TMD = 0 written	Main watch transition
	33 TMD = 1•STP = 1 written	Main stop
PLL mode	21 SCS = 0 written	PS transition mode
	20 SCS = 1•MCS = 1 written	PM transition mode
	59 TMD = 1•STP = 0•SLP = 1 written	PLL sleep
	58 TMD = 0 written	PLL watch transition P
	57 TMD = 1•STP = 1 written	Pseudo-watch transition
Sub mode	10 SCS = 1•MCS = 1 written	SM transition mode
	12 SCS = 1•MCS = 0 written	SP transition mode
	11 Reset initiated	Main oscillation stabilization
	42 TMD = 1•STP = 0•SLP = 1 written	Sub-sleep
	43 TMD = 0 written	Sub-watch
	44 TMD = 1•STP = 1 written	Sub-stop
PM transition mode	13 PLL → main switching timing wait completed	Main mode
	38 TMD = 1•STP = 0•SLP = 1 written	PM transition sleep
	39 TMD = 0 written and PLL → main switching timing wait completed	Main watch transition
	40 TMD = 1 and STP = 1 written and PLL → main switching timing wait completed	Main stop

Table 6.4.9a List of Transition Conditions (Continued)

State before transition	Transition conditions	State after transition
SM transition mode	02 Main oscillation stabilization waiting period completed	Main mode
	03 Reset initiated or interrupt	Main oscillation stabilization
	04 SCS = 0 written	Sub mode
	27 TMD = 1•STP = 0•SLP = 1 written	SM transition sleep
	28 TMD = 0 and main oscillation stabilization waiting period completed	Main watch
	29 TMD = 1 and STP = 1 written and main oscillation stabilization waiting period completed	Main stop
MP transition mode	16 PLL oscillation stabilization waiting period completed	PLL mode
	14 SCS = 1•MCS = 1 written	Main mode
	15 SCS = 0 written	MS transition mode
	68 TMD = 1•STP = 0•SLP = 1 written	MP transition sleep
	70 TMD = 0 written	PLL watch transition M
	69 TMD = 1•STP=1 written	Pseudo-watch mode
SP transition mode	17 Main oscillation stabilization waiting period completed	MP transition mode
	18 MCS = 1 written	SM transition mode
	19 Reset initiated	Main oscillation stabilization
	75 TMD = 1•STP = 0•SLP = 1 written	SP transition sleep
	76 TMD = 0 written	PLL watch
	78 TMD = 1 and STP = 1 written and main oscillation stabilization waiting period completed	Pseudo-watch mode
MS transition mode	09 Main → sub-clock switching timing wait completed	Sub mode
	08 Reset initiated	Main mode
	51 TMD = 1•STP = 0•SLP = 1 written	MS transition sleep
	52 TMD = 0 written and main → sub switching wait completed	Sub watch
	53 TMD = 1 and STP = 1 written and main → sub switching wait completed	Sub mode
PS transition mode	23 PLL → main clock switching timing wait completed	MS transition mode
	22 SCS = 1 written	PM transition mode
	56 TMD = 1•STP = 0•SLP = 1 written	PS transition sleep
Main sleep	26 Interrupt or reset initiated	Main mode
SM transition sleep	24 Main oscillation stabilization waiting period completed	Main sleep
	25 Interrupt or reset initiated	SM transition mode
PM transition sleep	34 PLL → main clock switching timing wait completed	Main sleep
	35 Interrupt or reset initiated	PM transition mode
PLL sleep	63 Interrupt or reset initiated	PLL mode
MP transition sleep	66 PLL oscillation stabilization waiting period completed	PLL sleep
	67 Interrupt or reset initiated	MP transition mode
SP transition sleep	73 Main oscillation stabilization waiting period completed	MP transition sleep
	74 Interrupt or reset initiated	SP transition mode
Sub-sleep	46 Interrupt or reset initiated	Sub mode

Table 6.4.9a List of Transition Conditions (Continued)

State before transition	Transition conditions	State after transition
MS transition sleep	49 Main → sub-clock switching timing wait completed	Sub-sleep
	50 Interrupt or reset initiated	MS transition mode
PS transition sleep	54 PLL → main clock switching timing wait completed	MS transition sleep
	55 Interrupt or reset initiated	PS transition mode
Main watch	30 Interrupt or reset initiated	SM transition mode
Main watch transition	36 Main → sub-clock switching timing wait completed	Main watch
	37 Interrupt or reset initiated	Main mode
PLL watch	77 Interrupt or reset initiated	SP transition mode
PLL watch transition M	72 Main → sub-clock switching timing wait completed	PLL watch
	71 Interrupt or reset initiated	MP transition mode
PLL watch transition P	65 PLL → main clock switching timing wait completed	PLL watch transition M
	64 Interrupt or reset initiated	PLL mode
Sub watch	47 Interrupt or reset initiated	Sub mode
Main stop	41 Interrupt or reset initiated	Main oscillation stabilization
Pseudo-watch	62 Interrupt or reset initiated	MP transition mode
Pseudo-watch transition	61 PLL → main clock switching timing wait completed	Pseudo-watch mode
	60 Interrupt or reset initiated	PLL mode
Sub stop	48 Interrupt	Sub oscillation stabilization
	79 Reset initiated	Main oscillation stabilization
Sub oscillation stabilization	45 Subclock oscillation stabilization waiting period completed	Sub mode
	80 Reset initiated	Main oscillation stabilization

State Transition Diagrams

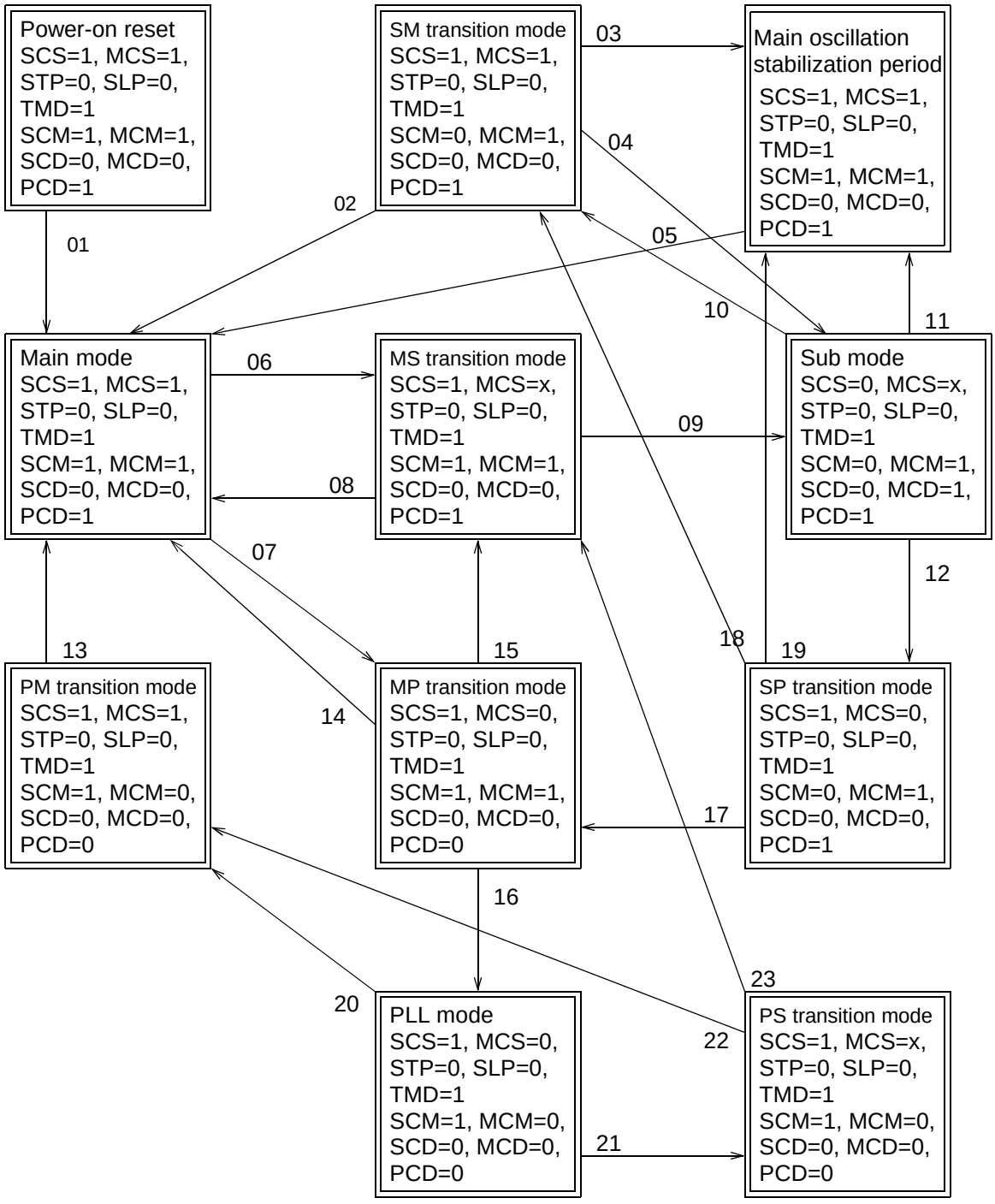


Figure 6.4.9a Low Power Consumption Mode Transition Diagram A

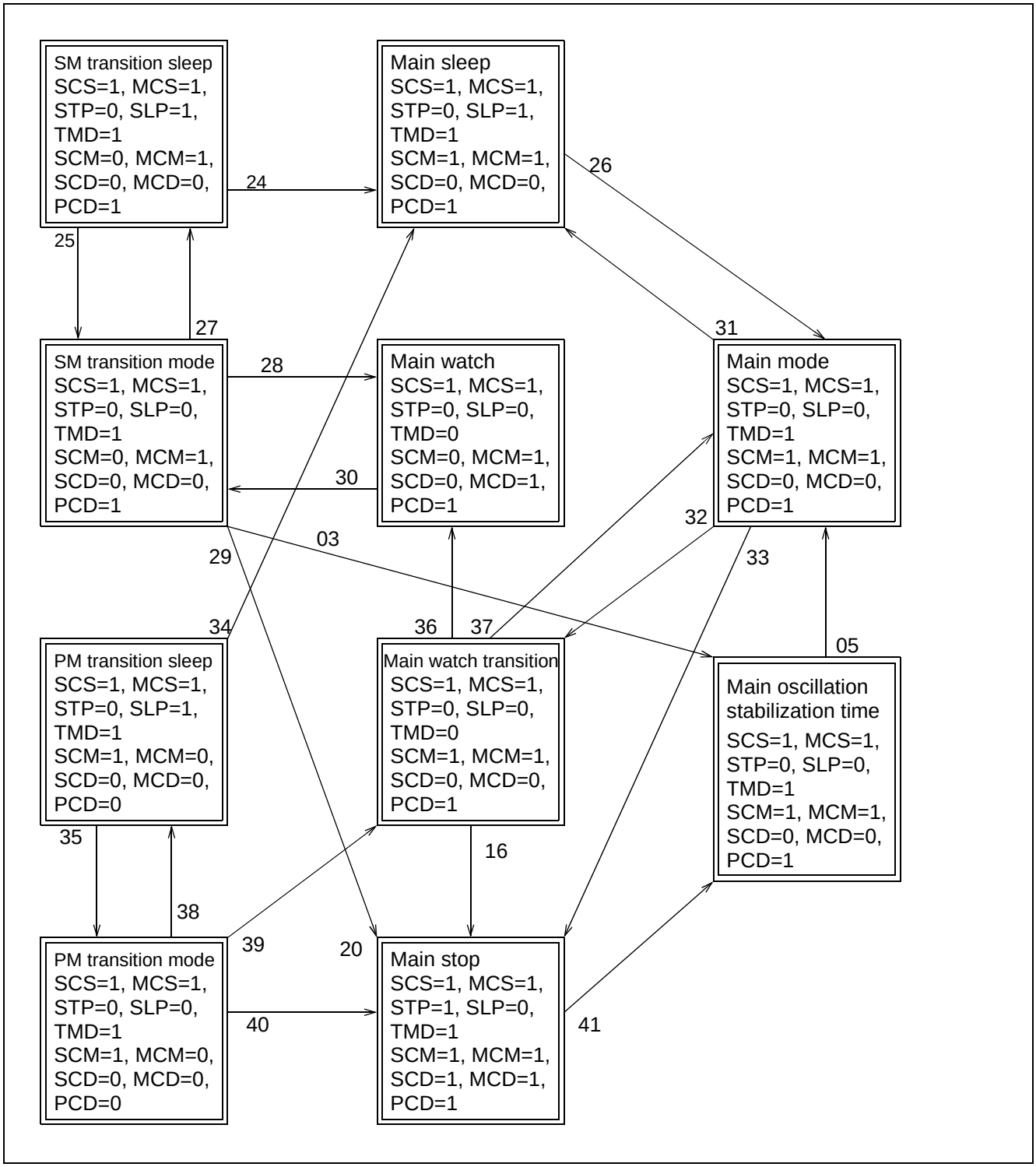


Figure 6.4.9b Low Power Consumption Mode Transition Diagram B

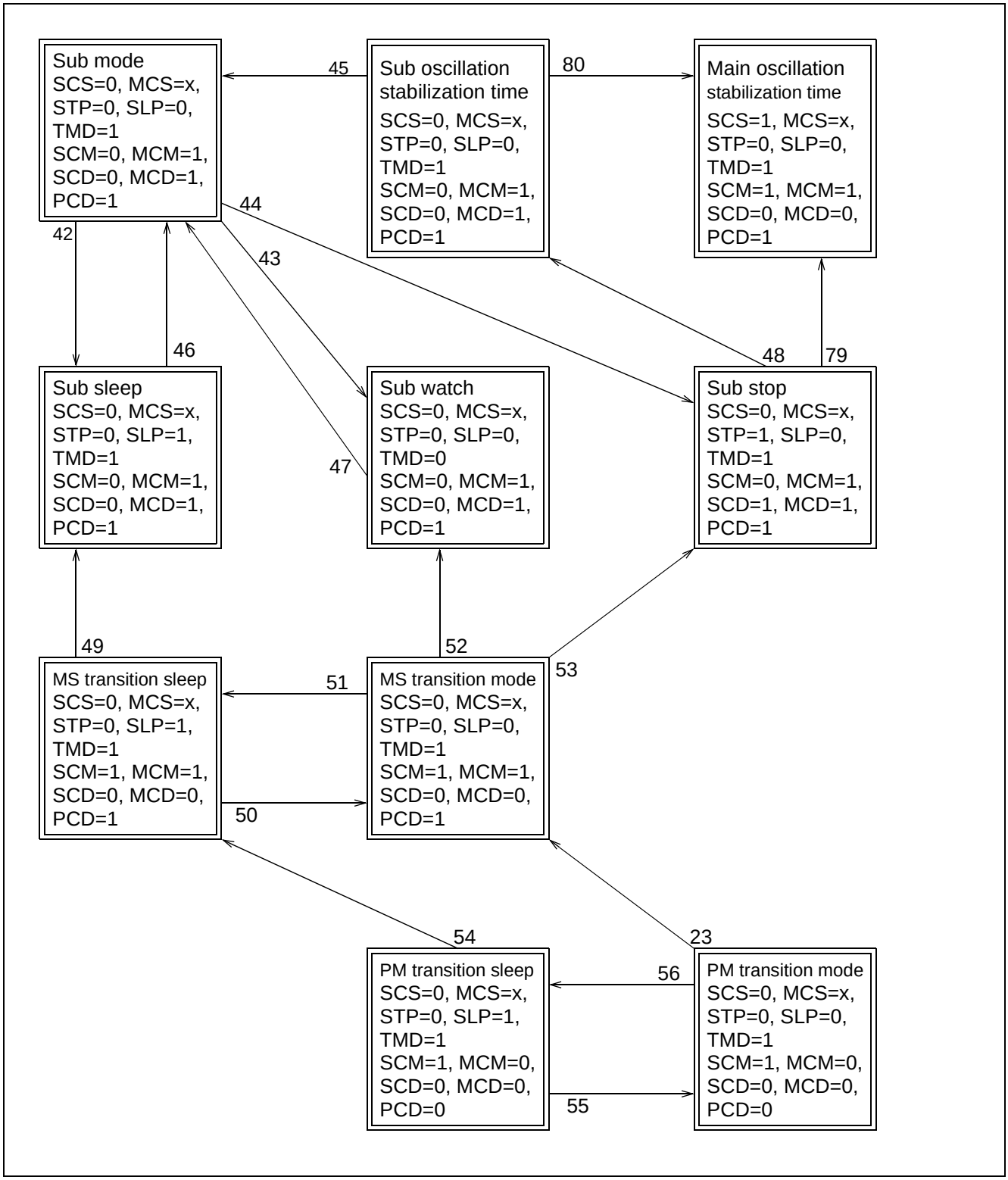


Figure 6.4.9c Low Power Consumption Mode Transition Diagram C

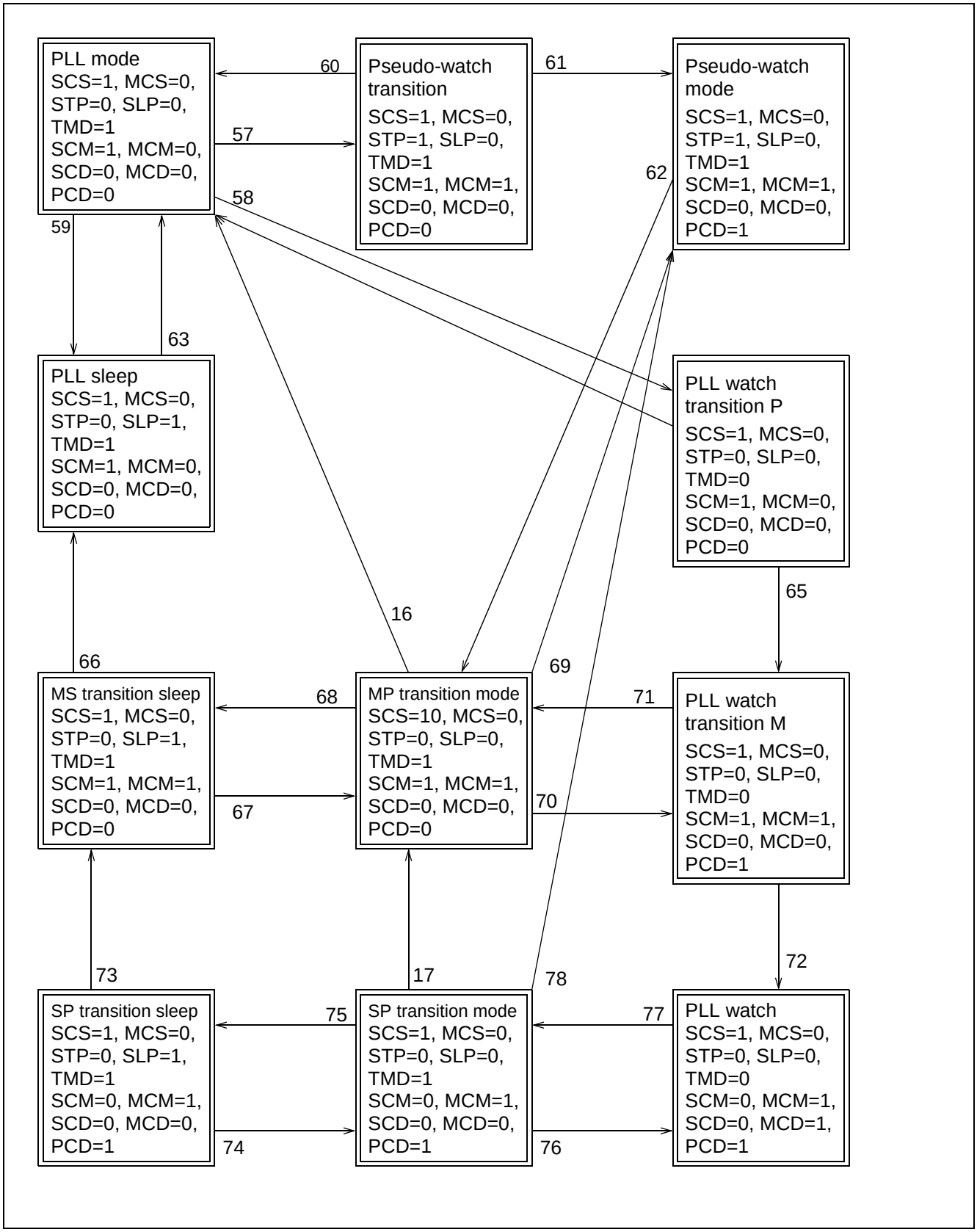


Figure 6.4.9d Low Power Consumption Mode Transition Diagram D

Chapter 7: Interrupt

7.1 Outline

The F^{2}MC-16LX has interrupt functions that terminate the currently executing processing and transfer control to another specified program when a specified event occurs. There are four types of interrupt functions:

- Hardware interrupt: ..... Interrupt processing due to an internal resource event
- Software interrupt: ..... Interrupt processing due to a software event occurrence instruction
- Extended intelligent I/O service (EI^{2}OS): ... Transfer processing due to an internal resource event
- Exception: ..... Termination due to an operation exception

7.2 Causes of Interrupt

Table 7.2a Interrupt causes, interrupt vectors, and interrupt control registers

Interrupt cause	IIOS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Reset	×	# 08	$FFFFDC_H$	——	——
INT9 instruction	×	# 09	$FFFFD8_H$	——	——
Exception	×	# 10	$FFFFD4_H$	——	——
A/D converter	❍	# 11	$FFFFD0_H$	ICR00	$0000B0_H$
Time base Timer	×	# 12	$FFFFCC_H$		
DTP0 (external 0) / UART3 reception completion	❍	# 13	$FFFFC8_H$	ICR01	$0000B1_H$
DTP1 (external 1) / UART4 reception completion	❍	# 14	$FFFFC4_H$		
DTP2 (external 2) / UART3 transmission completion	❍	# 15	$FFFFC0_H$	ICR02	$0000B2_H$
DTP3 (external 3) / UART4 transmission completion	❍	# 16	$FFFFBC_H$		
DTP4 - 7 (external 4 - 7)	❍	# 17	$FFFFB8_H$	ICR03	$0000B3_H$
Output compare (channel 1)	❍	# 18	$FFFFB4_H$		
UART2 reception completion	❍	# 19	$FFFFB0_H$	ICR04	$0000B4_H$
UART1 reception completion	❍	# 20	$FFFFAC_H$		
Input capture (channel 3)	❍	# 21	$FFFFA8_H$	ICR05	$0000B5_H$
Input capture (channel 2)	❍	# 22	$FFFFA4_H$		
Input capture (channel 1)	❍	# 23	$FFFFA0_H$	ICR06	$0000B6_H$
Input capture (channel 0)	❍	# 24	$FFFF9C_H$		
8/16-bit PPG0 counter borrow	×	# 25	$FFFF98_H$	ICR07	$0000B7_H$
16-bit reload timer 2 - 0	❍	# 26	$FFFF94_H$		
Time prescalar	×	# 27	$FFFF90_H$	ICR08	$0000B8_H$
Output Compare (channel 0)	❍	# 28	$FFFF8C_H$		
UART2 transmission completion	❍	# 29	$FFFF88_H$	ICR09	$0000B9_H$
PWC timer	❍	# 30	$FFFF84_H$		
UART1 transmission completion	❍	# 31	$FFFF80_H$	ICR10	$0000BA_H$
16-bit free run timer overflow	❍	# 32	$FFFF7C_H$		
UART0 transmission completion	❍	# 33	$FFFF78_H$	ICR11	$0000BB_H$
8/16-bit PPG1 counter borrow	×	# 34	$FFFF74_H$		
IEBus reception completion	◎	# 35	$FFFF70_H$	ICR12	$0000BC_H$
IEBus transmission completion	◎	# 37	$FFFF68_H$	ICR13	$0000BD_H$
UART0 reception completion	◎	# 39	$FFFF60_H$	ICR14	$0000BE_H$
Reserved	×	# 41	$FFFF58_H$	ICR15	$0000BF_H$
Delayed interrupt	×	# 42	$FFFF54_H$		

❍ : The interrupt request flag is cleared by the IIOS interrupt clear signal.
◎ : The interrupt request flag is cleared by the IIOS interrupt clear signal. A stop request is available.
× : The interrupt request flag is not cleared by the IIOS interrupt clear signal.

Note: For a resource with two interrupt causes for a single interrupt number, both interrupt request flags are cleared by the IIOS interrupt clear signal.

7.3 Interrupt Vector

Table 7.3a MB90580 interrupt assignment table (1/2)

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 0	FFFFFC $_H$	FFFFFD $_H$	FFFFFE $_H$	Unused	#0	None
⋮	⋮	⋮	⋮	⋮	⋮	⋮
INT 7	$FFFFE0_H$	$FFFFE1_H$	$FFFFE2_H$	Unused	#7	None
INT 8	$FFFFDC_H$	$FFFFDD_H$	$FFFFDE_H$	FFFFDF	#8	(RESET vector)
INT 9	$FFFFD8_H$	$FFFFD9_H$	$FFFFDA_H$	Unused	#9	None
INT 10	$FFFFD4_H$	$FFFFD5_H$	$FFFFD6_H$	Unused	#10	<Exception>
INT 11	$FFFFD0_H$	$FFFFD1_H$	$FFFFD2_H$	Unused	#11	A/D
INT 12	$FFFFCC_H$	$FFFFCD_H$	$FFFFCE_H$	Unused	#12	Time base Timer
INT 13	$FFFFC8_H$	$FFFFC9_H$	$FFFFCA_H$	Unused	#13	DTP0 (External interrupt #0) / UART3 reception completion
INT 14	$FFFFC4_H$	$FFFFC5_H$	$FFFFC6_H$	Unused	#14	DTP1 (External interrupt #1) / UART4 reception completion
INT 15	$FFFFC0_H$	$FFFFC1_H$	$FFFFC2_H$	Unused	#15	DTP2 (External interrupt #2) / UART3 transmission completion
INT 16	$FFFFBC_H$	$FFFFBD_H$	$FFFFBE_H$	Unused	#16	DTP3 (External interrupt #3) / UART4 transmission completion
INT 17	$FFFFB8_H$	$FFFFB9_H$	$FFFFBA_H$	Unused	#17	DTP4 - 7 (External interrupt #4 - #7)
INT 18	$FFFFB4_H$	$FFFFB5_H$	$FFFFB6_H$	Unused	#18	Output compare (channel 1)
INT 19	$FFFFB0_H$	$FFFFB1_H$	$FFFFB2_H$	Unused	#19	UART2 reception completion
INT 20	$FFFFAC_H$	$FFFFAD_H$	$FFFFAE_H$	Unused	#20	UART1 reception completion
INT 21	$FFFFA8_H$	$FFFFA9_H$	$FFFFAA_H$	Unused	#21	Input capture (channel 3)
INT 22	$FFFFA4_H$	$FFFFA5_H$	$FFFFA6_H$	Unused	#22	Input capture (channel 2)
INT 23	$FFFFA0_H$	$FFFFA1_H$	$FFFFA2_H$	Unused	#23	Input capture (channel 1)
INT 24	$FFFF9C_H$	$FFFF9D_H$	$FFFF9E_H$	Unused	#24	Input capture (channel 0)
INT 25	$FFFF98_H$	$FFFF99_H$	$FFFF9A_H$	Unused	#25	8/16-bit PPG0 counter borrow
INT 26	$FFFF94_H$	$FFFF95_H$	$FFFF96_H$	Unused	#26	16-bit reload timer 2 - 0
INT 27	$FFFF90_H$	$FFFF91_H$	$FFFF92_H$	Unused	#27	Time prescalar
INT 28	$FFFF8C_H$	$FFFF8D_H$	$FFFF8E_H$	Unused	#28	Output compare (channel 0)
INT 29	$FFFF88_H$	$FFFF89_H$	FFFF8A $_H$	Unused	#29	UART2 transmission completion
INT 30	$FFFF84_H$	$FFFF85_H$	FFFF86 $_H$	Unused	#30	PWC timer
INT 31	$FFFF80_H$	$FFFF81_H$	FFFF82 $_H$	Unused	#31	UART1 transmission completion
INT 32	$FFFF7C_H$	$FFFF7D_H$	FFFF7E $_H$	Unused	#32	16-bit free run timer overflow
INT 33	$FFFF78_H$	$FFFF79_H$	FFFF7A $_H$	Unused	#33	UART0 transmission completion
INT 34	$FFFF74_H$	$FFFF75_H$	FFFF76 $_H$	Unused	#34	8/16 bit PPG 1 counter borrow
INT 35	$FFFF70_H$	$FFFF71_H$	FFFF72 $_H$	Unused	#35	IEBus reception completion
INT 36	$FFFF6C_H$	$FFFF6D_H$	FFFF6E $_H$	Unused	#36	None
INT 37	$FFFF68_H$	$FFFF69_H$	FFFF6A $_H$	Unused	#37	IEBus transmission completion
INT 38	FFFF64 $_H$	$FFFF65_H$	FFFF66 $_H$	Unused	#38	None
INT 39	$FFFF60_H$	$FFFF61_H$	FFFF62 $_H$	Unused	#39	UART0 reception completion
INT 40	$FFFF5C_H$	$FFFF5_H$	$FFFF5E_H$	Unused	#40	None
INT 41	$FFFF58_H$	$FFFF59_H$	FFFF5A $_H$	Unused	#41	(RESERVED)
INT 42	$FFFF54_H$	$FFFF55_H$	$FFFF56_H$	Unused	#42	Delayed interrupt

7.4 Hardware Interrupt

7.4.1 Overview

In response to an interrupt request signal from an internal resource, the CPU pauses current program execution and transfers control to the interrupt processing program defined by the user. This function is called the hardware interrupt function. A hardware interrupt occurs when relevant conditions are satisfied as a result of two operations: comparison between the interrupt request level and the value in the interrupt level mask register of PS of the CPU, and hardware reference to the I flag value in PS. The CPU performs the following processing when a hardware interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- Sets ILM in the PS register. The currently requested interrupt level is automatically set.
- Fetches the corresponding interrupt vector value and branches to the processing indicated by that value.

7.4.2 Structure

Hardware interrupts are handled by the following three sections:

- Internal resources .....Interrupt enable and request bits: Used to control interrupt requests from resources.
- Interrupt controller.....ICR:Assigns interrupt levels and determines the priority levels of simultaneously requested interrupts.
- CPU .....I and ILM:Used to compare the requested and current interrupt levelsand to identify the interrupt enable status.
  Microcode:Interrupt processing step

The status of these sections are indicated by the resource control registers for internal resources, the ICR for the interrupt controller, and the CCR value for the CPU. To use a hardware interrupt, set the three sections beforehand by using software.

The interrupt vector table referenced during interrupt processing is assigned to addresses $FFFC_H$ to $FFFFFF_H$ in memory. These addresses are shared with software interrupts.

7.4.3 Operation

An internal resource that has the hardware interrupt request function has an interrupt request flag and interrupt enable flag. The interrupt request flag indicates whether an interrupt request exists, and the interrupt enable flag indicates whether the relevant internal resource requests an interrupt to the CPU. The interrupt request flag is set when an event occurs that is unique to the internal resource. When the interrupt enable flag indicates "enable," the resource issues an interrupt request to the interrupt controller.

When two or more interrupt requests are received at the same time, the interrupt controller compares the interrupt levels (IL) in ICR, selects the request at the highest level (the smallest IL value), then reports that request to the CPU. If multiple requests are at the same level, the interrupt controller selects the request with the lowest interrupt number. The relationship between the interrupt requests and ICRs is determined by the hardware.

The CPU compares the received interrupt level and the ILM in the PS register. If the interrupt level is smaller than the ILM value and the I bit of the PS register is set to 1, the CPU activates the interrupt processing microcode after the currently executing instruction is completed. The CPU references the ISE bit of the ICR of the interrupt controller at the beginning of the interrupt processing microcode, checks that the ISE bit is 0 (interrupt), and activates the interrupt processing body.

The interrupt processing body saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP, fetches three bytes of interrupt vector and loads them onto PC and PCB,

updates the ILM of PS to a level value of the received interrupt, sets the S flag, then performs branch processing. As a result, the interrupt processing program defined by the user is executed next.

Figure 7.4.3a illustrates the flow from the occurrence of a hardware interrupt until there is no interrupt request in the interrupt processing program. Figure 7.4.3b is a diagram of the hardware interrupt operation flow.

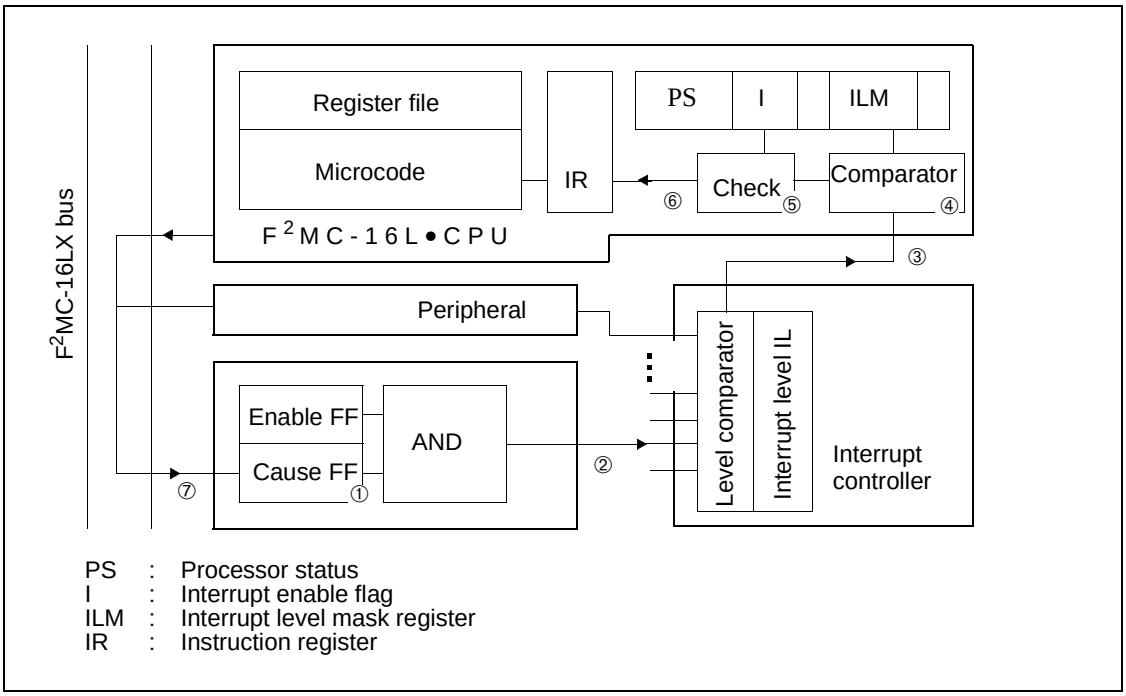


Figure 7.4.3a Occurrence and release of hardware interrupt

① An interrupt cause occurs in a peripheral.

② The interrupt enable bit in the peripheral is referenced. If interrupts are enabled, the peripheral issues an interrupt request to the interrupt controller.

③ Upon reception of the interrupt request, the interrupt controller determines the priority levels of simultaneously requested interrupts. Then, the interrupt controller transfers the interrupt level of the corresponding interrupt to the CPU.

④ The CPU compares the interrupt level requested by the interrupt controller with the ILM bit of the processor status register.

⑤ If the comparison shows that the requested level is higher than the current interrupt processing level, the I flag value of the same processor status register is checked.

⑥ If the check in step ⑤ shows that the I flag indicates interrupt enable status, the requested level is written to the ILM bit. Interrupt processing is performed as soon as the currently executing instruction is completed, then control is transferred to the interrupt processing routine.

⑦When the interrupt cause of step ① is cleared by software in the user interrupt processing routine, the interrupt request is completed.

The time required for the CPU to execute the interrupt processing in steps ⑥ and ⑦ is shown below.

Interrupt start : 24 + 6 x Table 7.4.3a machine cycles

Interrupt return : 15 + 6 x Table 7.4.3a machine cycles (RETI instruction)

Table 7.4.3a Compensation values for interrupt processing cycle count

Address indicated by the stack pointer	Cycle count compensation value
External area, 8-bit data bus	+4
External area, even-numbered address	+1
External area, odd-numbered address	+4
Internal area, even-numbered address	0
Internal area, odd-numbered address	+2

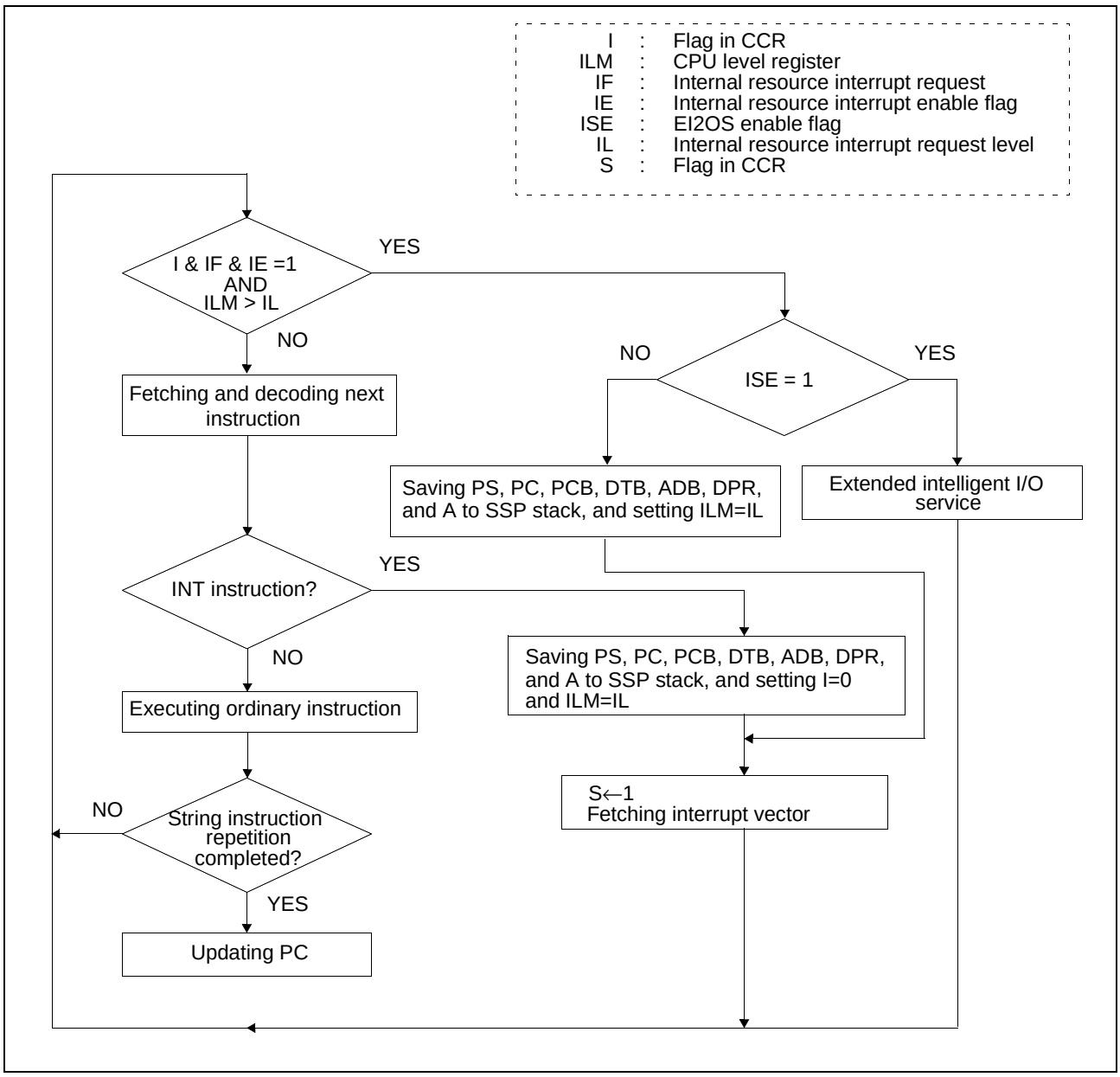


Figure 7.4.3b Hardware interrupt operation flow

7.4.4 Hardware Interrupt Ocurrence When Internal Resource Is Being Accessed

When internal I/O area is being asscessed, the CPU will not response to hardware interrupt immediately, there will be one instruction delay. Please refer to Chapter 2, section 2.1.3 for details.

7.4.5 Interrupt Inhibit Instruction

If F^{2}MC-16LX is executing interrupt inhibit instructions, the CPU will not response to hardware interrupt request immediately, there will be one instruction delay. Please refer to Chapter 2, section 2.1.3 for details.

7.4.6 Multiple Interrupts

The F^{2}MC-16LX CPU supports multiple interrupts. If an interrupt of a higher level occurs while another interrupt is being processed, control is transferred to the high-level interrupt after the currently executing instruction is completed. After processing of the high-level interrupt is completed, the original interrupt processing is resumed. An interrupt of the same or lower level may be generated while another interrupt is being processed. If this happens, the new interrupt request is suspended until the current interrupt processing is completed, unless the ILM value or I flag is changed by an instruction. The extended intelligent I/O service cannot be activated from multiple sources; while an extended intelligent I/O service is being processed, all other interrupt requests or extended intelligent I/O service requests are suspended.

7.4.7 Register Saving In Stack Upon Interrupt

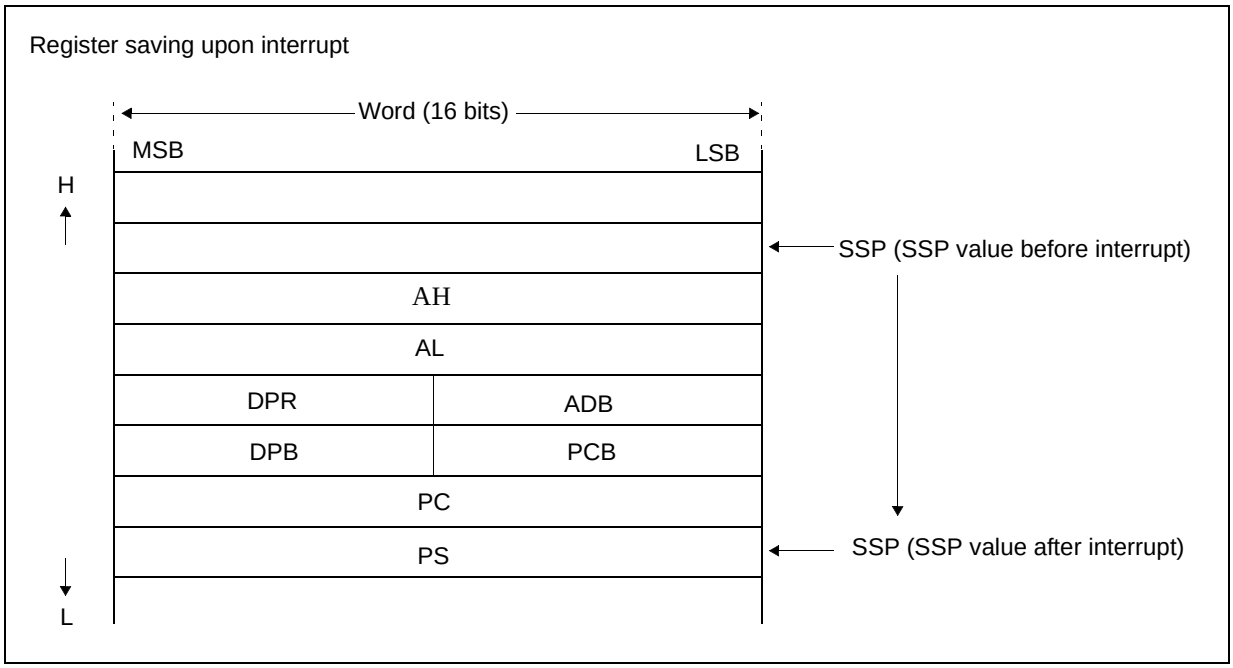


Figure 7.4.7a Registers saved in stack

7.4.8 Precaution in Using Hardware Interrupt

When there is an hardware interrupt, the interrupt request flag should be cleared before leaving the corresponding interrupt routine to avoid malfunction.

Some of the resources' interrupt request flag will be cleared automatically when certain register(s) is(are) read. In this case, please read those registers to clear the interrupt request flag before leaving the interrupt routine.

7.5 Software Interrupt

7.5.1 Overview

In response to execution of a special instruction, control is transferred from the program currently executed by the CPU to the interrupt processing program defined by the user. This is called the software interrupt function. A software interrupt occurs always when the software interrupt instruction is executed. The CPU performs the following processing when a software interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- Sets I in the PS register. Interrupts are automatically disabled.
- Fetches the corresponding interrupt vector value, then branches to the processing indicated by that value.

A software interrupt request issued by the INT instruction has no interrupt request or enable flag. A software interrupt request is always issued by executing the INT instruction.

The INT instruction does not have an interrupt level. Therefore, the INT instruction does not update ILM. The INT instruction clears the I flag to suspend subsequent interrupt requests.

7.5.2 Structure

Software interrupts are handled within the CPU:

- CPU ..... Microcode: Interrupt processing step

As shown in Table 7.3a, software interrupts share the same interrupt vector area with hardware interrupts. For example, interrupt request number INT 13 is used for external interrupt #0 of a hardware interrupt as well as for INT #13 of a software interrupt. Therefore, external interrupt #0 and INT #13 call the same interrupt processing routine.

7.5.3 Operation

When the CPU fetches and executes the software interrupt instruction, the software interrupt processing microcode is activated. The software interrupt processing microcode saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP. The microcode then fetches three bytes of interrupt vector and loads them onto PC and PCB, resets the I flag, and sets the S flag. Then, the microcode performs branch processing. As a result, the interrupt processing program defined by the user application program is executed next.

Figure 7.5.3a illustrates the flow from the occurrence of a software interrupt until there is no interrupt request in the interrupt processing program.

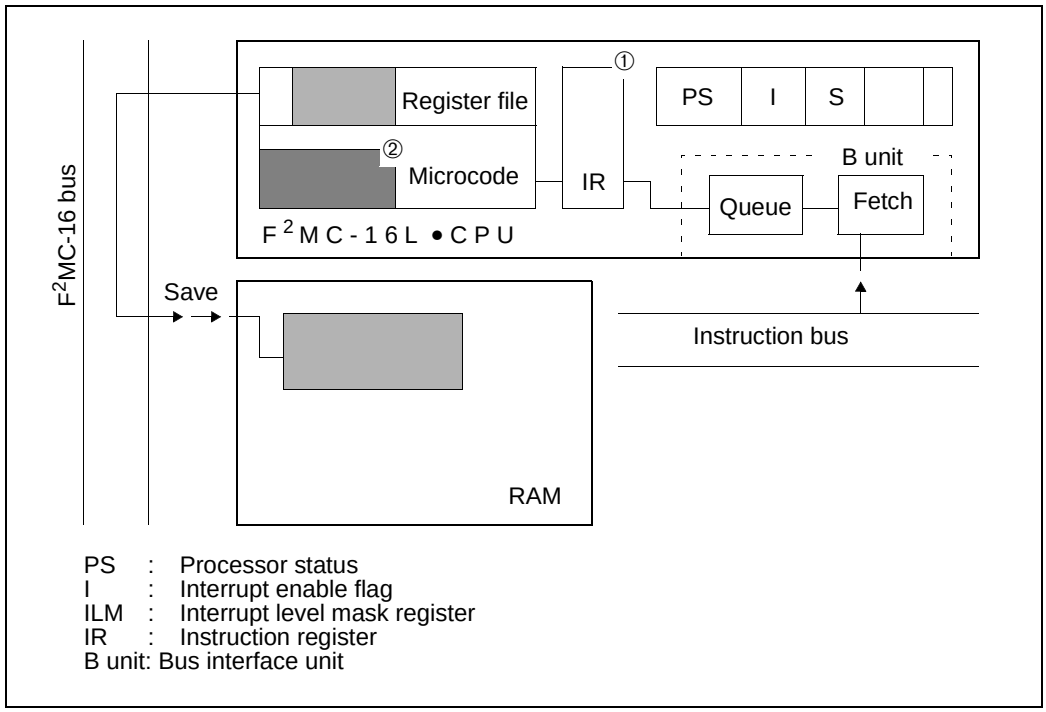


Figure 7.5.3a Occurrence and release of software interrupt

① The software interrupt instruction is executed.

② Special CPU registers in the register file are saved according to the microcode corresponding to the software interrupt instruction.

③The interrupt processing is completed with the RETI instruction in the user interrupt processing routine.

7.5.4 Others

When the program bank register (PCB) is FFH, the CALLV instruction vector area overlaps the table of the INT #vct8 instruction. When designing software, ensure that the CALLV instruction does not use the same address as that of the #vct8 instruction.

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

7.6.1 Overview

EI^{2}OS is a type of hardware interrupt operation that automatically transfers data between I/O and memory. Conventionally, data is transferred between I/O and memory by an interrupt processing program. EI^{2}OS, however, enables data to be transferred as if in DMA mode. EI^{2}OS has the following advantages over the conventional interrupt processing method:

- ▷ Writing a transfer program is unnecessary, thus the entire program size can be small.
- ▷ No internal register is used for transfer. Therefore, saving the register values is unnecessary, resulting in a higher transfer speed.
- ▷ I/O can stop transfer at any time. Therefore, unnecessary data is not transferred.
- ▷ The buffer address can be incremented, decremented, or left unupdated.
- ▷ The I/O address can be incremented, decremented, or left unupdated (when the buffer address is updated).

At the end of EI^{2}OS processing, the CPU automatically branches to the interrupt processing routine after setting the end condition. Therefore, the user can identify the end condition type.

Figure 7.6.1a outlines the EI^{2}OS.

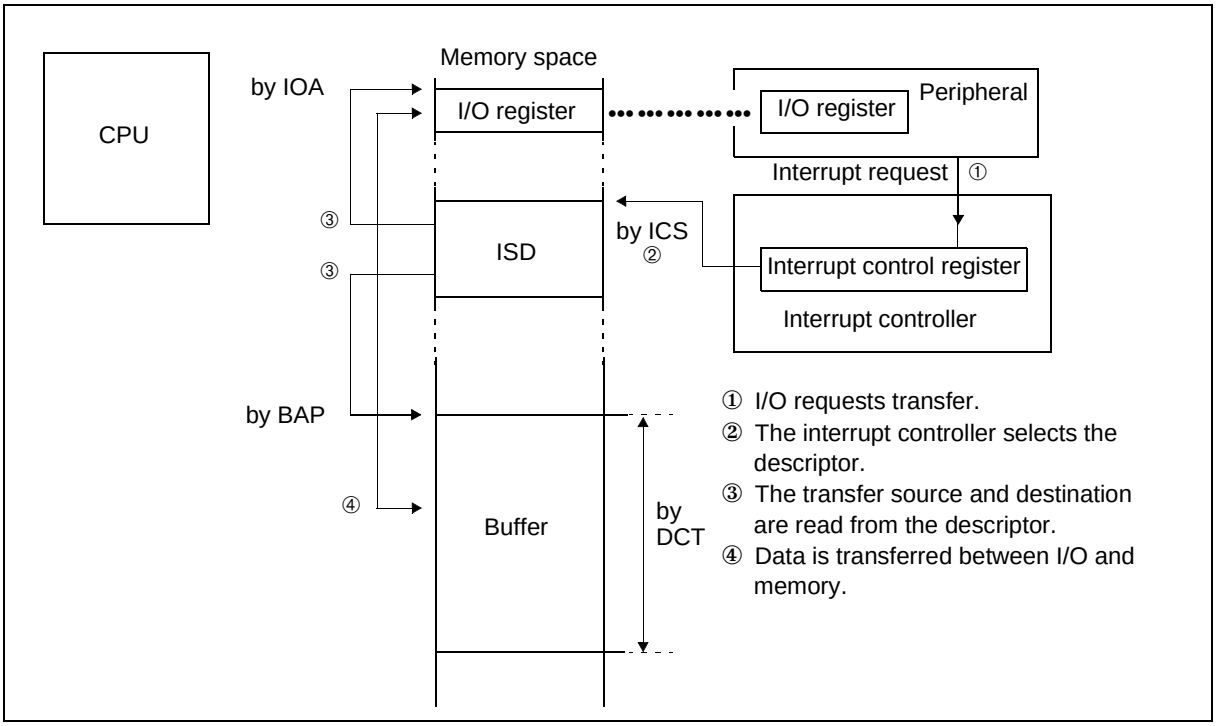


Figure 7.6.1a Outline of extended intelligent I/O service

Note: Notes:The area that can be specified by IOA is between 000000_H and $00FFFF_H$.
The area that can be specified by BAP is between 000000_H and $FFFFFF_H$.
The maximum transfer count that can be specified by DCT is 65,536.

7.6.2 Structure

EI^{2}OS is handled by the following four sections:

- Internal resources ..... Interrupt enable and request bits: Used to control interrupt requests from resources.
- Interrupt controller — ICR:Assigns interrupt levels, determines the priority levels of simultaneously requested interrupts, and selects the EI^{2}OS operation.
- CPU ..... I and ILM:Used to compare the requested and current interrupt levels and to identify the interrupt enable status.
  Microcode:E^{2}OS processing step
- RAM ..... Descriptor:Describes the EI^{2}OS transfer information.

Each register is described below.

(1) Interrupt control register (ICR)

The interrupt control register is in the interrupt controller. This register corresponds to I/Os that have the interrupt function. This register has the following three functions:

- Sets the interrupt level of the corresponding peripheral.
- Selects whether to handle the interrupt of the corresponding peripheral as an ordinary interrupt or as an extended intelligent I/O service.
- Selects the extended intelligent I/O service channel.

Do not access this register by a read-modify-write instruction, as doing so causes misoperation.

Interrupt control register (ICR)

	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	⇦ Bit number
Address : $B0_H$–BF_H	ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	IL0	when written
Read/write⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value⇨	(0)	(0)	(0)	(0)	0)	(1)	(1)	(1)	

	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	⇦ Bit number
Address : $B0_H$–BF_H	—	—	S1	S0	ISE	IL2	IL1	IL0	when read
Read/write⇨	(–)	(–)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value⇨	(X)	(X)	(0)	(0)	0)	(1)	(1)	(1)	

Note: • ICS3 to ICS0 are valid only when EI^{2}OS is activated. Set ISE to '1' to activate EI^{2}OS, and to '0' not to activate it. When EI2OS is not to be activated, any value can be written to ICS3 to ICS0.

* '1' is always read.

- ICS1 and ICS0 are valid for write only. S1 and S0 are valid for read only.

[bits 15 to 12] or [bits 7 to 4] ICS3 to ICS0

These bits are used to select the EI^{2}OS channel. These bits are write-only. The value specified in these bits determines the address of the extended intelligent I/O service descriptor in memory, which is explained later. ICS is initialized upon a reset.

Table 7.6.2a shows the correspondence between ICS, channel numbers, and descriptor addresses.

Table 7.6.2a ICS bits, channel numbers, 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

[bits 13 and 12] or [bits 5 and 4] S0 and S1

These are EI^{2}OS end status bits. These bits are read-only. When the EI^{2}OS is completed, the end condition can be identified by checking the value in these bits. These bits are set to '00' upon a reset.

Table 7.6.2b shows the relationship between the S bits and end conditions

.

Table 7.6.2b S bits and end conditions

S1	S0	End condition
0	0	Reserved
0	1	Count completion
1	0	Reserved
1	1	Resource request

[bit 11] or [bit 3] ISE

This is the EI^{2}OS enable bit. This bit can be read or written to. Upon issuance of an interrupt request, EI^{2}OS is activated if this bit is set to '1' and the interrupt sequence is activated if this bit is set to '0.' If the EI^{2}OS end condition is satisfied (the S1 and S0 bits are not '00'), the ISE bit is cleared to '0.' If the corresponding peripheral does not have the EI^{2}OS function, the software must set ISE to '0.'

This bit is initialized to '0' upon a reset.

[bits 10 to 8] or [bits 2 to 0] IL0, IL1, and IL2

These are interrupt level setting bits. Specify the interrupt level of the corresponding internal resource. These bits can be read and written to. These bits are initialized to level 7 (no interrupt) upon a reset. Table 7.6.2c describes the relationship between the interrupt level setting bits and interrupt levels.

Table 7.6.2c Interrupt level setting bits and interrupt levels

IL2	IL1	IL0	Level
0	0	0	0 (Highest interrupt level)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6 (Lowest interrupt level)
1	1	1	7 (No interrupt)

(2)Extended intelligent I/O service descriptor (ISD)

The extended intelligent I/O service descriptor exists between 000100_H and $00017F_H$ in internal RAM, and consists of the following items:

- ▷ Data transfer control data
- ▷ Status data
- ▷ Buffer address pointer

Figure 7.6.2a shows the configuration of the extended intelligent I/O service descriptor.

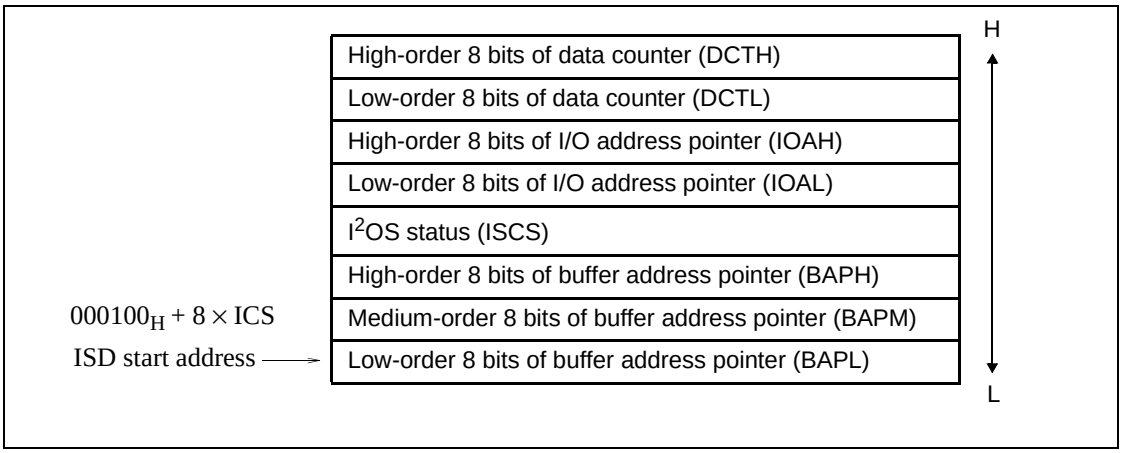
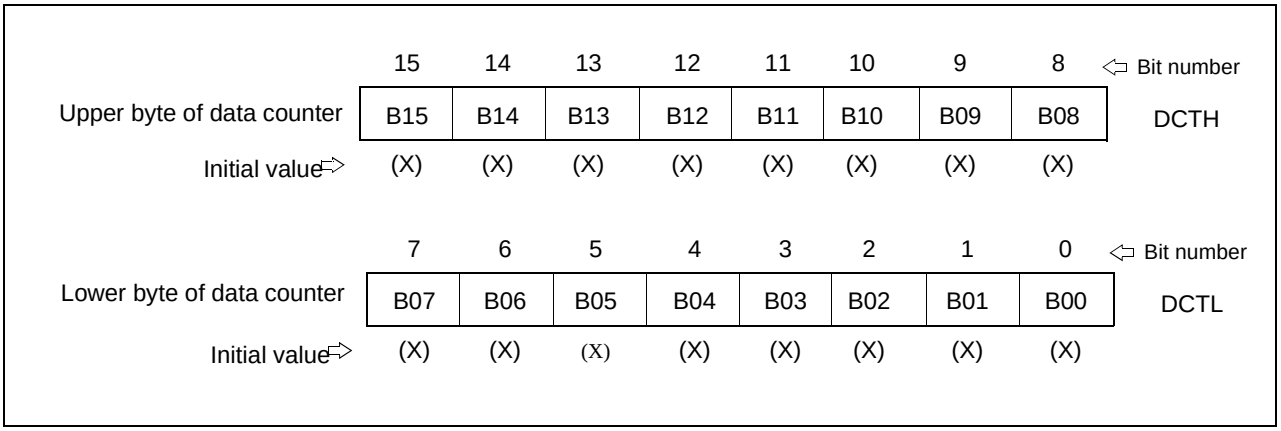


Figure 7.6.2a Extended intelligent I/O service descriptor configuration

■ Data counter (DCT)

This is a 16-bit register that works as a counter corresponding to the number of data items transferred. This counter is decremented by one before data transfer. EI^{2}OS is terminated when this counter reaches 0.



■ I/O register address pointer (IOA)

This is a 16-bit register that indicates the low-order address (A15 to A0) of the buffer and I/O register used for data transfer to and from the buffer. The high-order address (A23 to A16) are all zeroes, and any I/O between addresses 000000_H and $00FFFF_H$ can be specified.

	15	14	13	12	11	10	9	8	⇦ Bit number
Upper address pointer	A15	A14	A13	A12	A11	A10	A09	A08	IOAH
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	
	7	6	5	4	3	2	1	0	⇦ Bit number
Lower address pointer	A07	A06	A05	A04	A03	A02	A01	A00	IOAL
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

■ EI^{2}OS status register (ISCS)

This eight-bit register indicates the update direction (increment/decrement), transfer data format (byte/word), and transfer direction of the buffer address pointer and the I/O register address pointer. This register also indicates whether the buffer address pointer or I/O register address pointer is updated or fixed.

	7	6	5	4	3	2	1	0	⇦ Bit number
	—	—	—	IF	BW	BF	DIR	SE	
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)	

* Always write 0 to bits 7 to 5 of ISCS.
Each bit is described below.

[bit 4] IF : Specify whether the I/O register address pointer is updated or fixed.

0	The I/O register address pointer is updated after data transfer.
1	The I/O register address pointer is not updated after data transfer.

Note: Only increment is allowed.

[bit 3] BW: Specify the transfer data length.

0	Byte
1	Word

[bit 2] BF: Specify whether the buffer address pointer is updated or fixed.

0	The buffer address pointer is updated after data transfer.
1	The buffer address pointer is not updated after data transfer.

Note: Only the low-order 16 bits of the buffer address are updated. Only increment is allowed.

[bit 1] DIR: Specify the data transfer direction.

0	I/O → Buffer
1	Buffer → I/O

[bit 0] SE: Control the termination of the extended intelligent I/O service based on resource requests.

0	The extended intelligent I/O service is not terminated by a resource request.
1	The extended intelligent I/O service is terminated by a resource request.

■ Buffer address pointer (BAP)

This 24-bit register holds the address used for the next EI^{2}OS transfer. BAP exists for each EI^{2}OS channel. Therefore, each EI^{2}OS channel can be used for transfer with anywhere in the 16-Mbyte space.

Note: If the BF bit of ISCS is set to '0' (update enabled), only the low-order 16 bits of BAP changes and BAPH does not change.

7.6.3 Operation

.

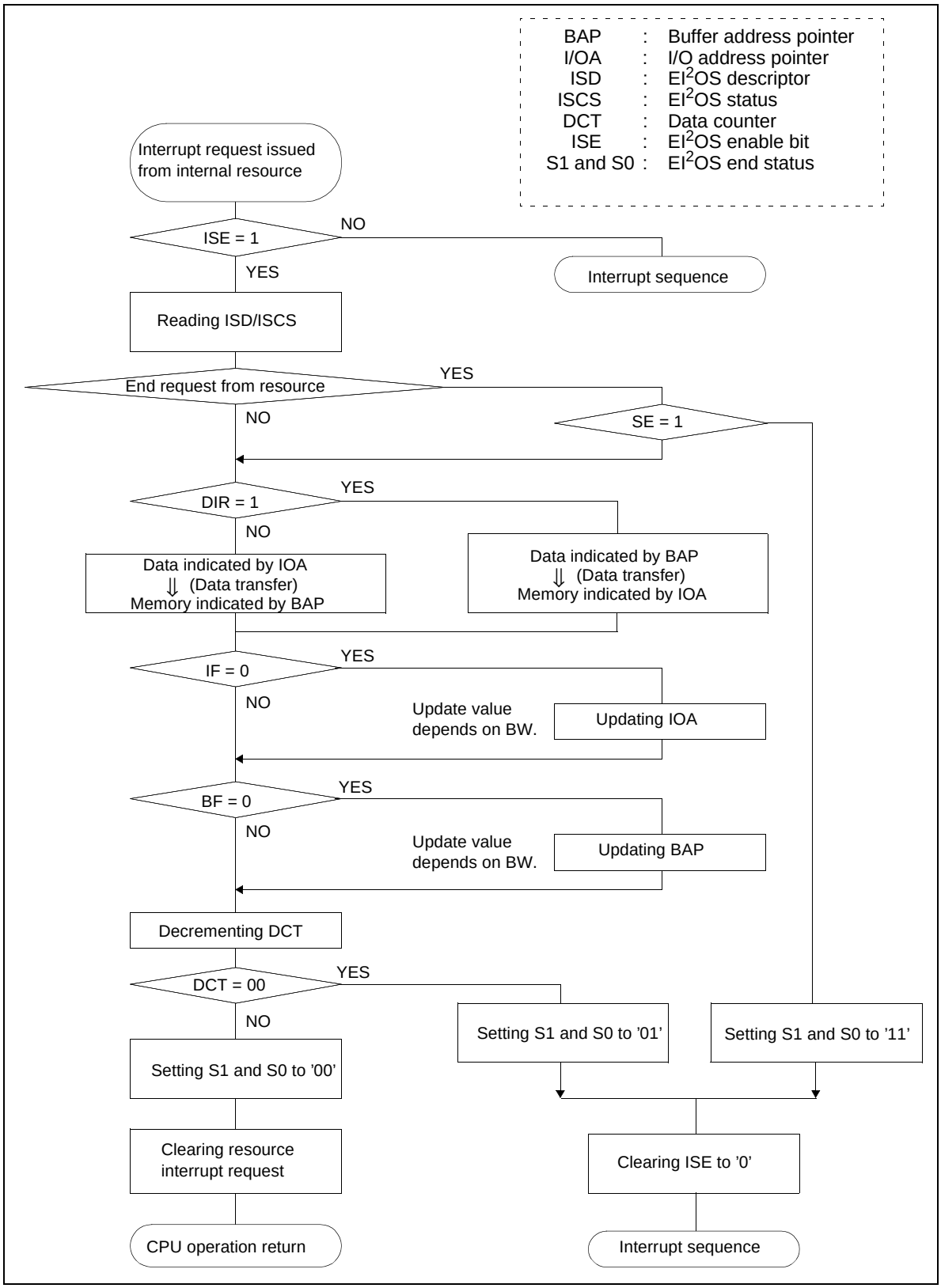


Figure 7.6.3a EI^{2}OS operation flow

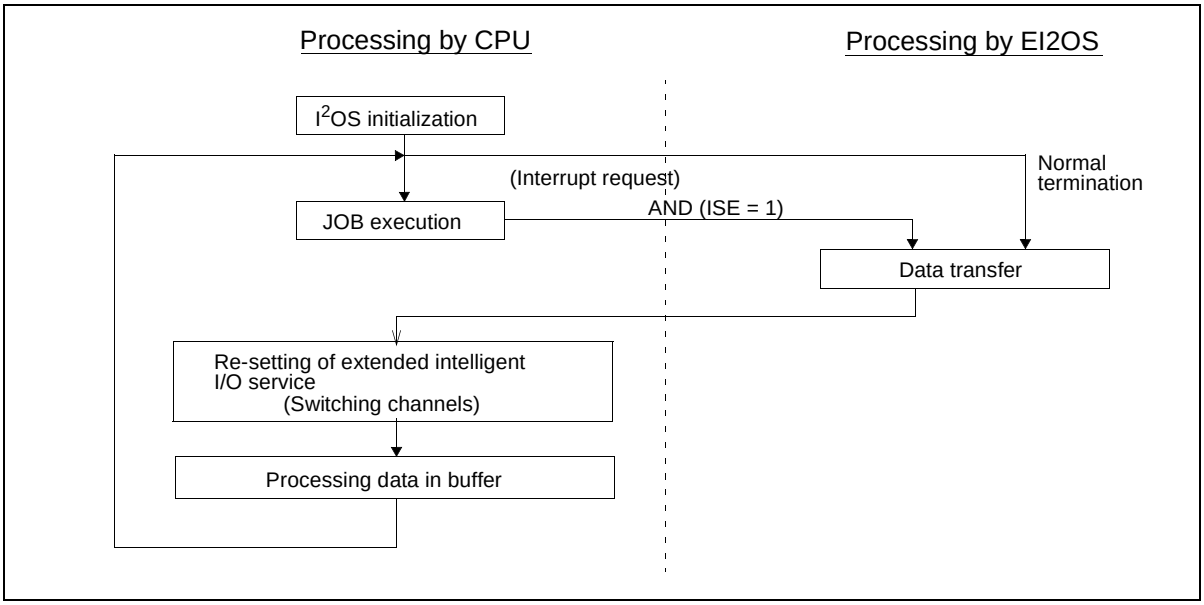


Figure 7.6.3b EI^{2}OS use flow

7.6.4 EI^{2}OS Execution Time

(1) When data transfer continues (when the stop condition is not satisfied)

EI2OS Execution Time = (value in Table 7.6.4a + value in Table 7.6.4b) machine cycle

Table 7.6.4a Execution time when the extended I2OS continues

ISCS SE bit		Set to '0'		Set to '1'	
I/O address pointer		Fixed	Updated	Fixed	Updated
Buffer address pointer	Fixed	32	34	33	35
	Updated	34	36	35	37

(2) When a stop request is issued from a resource

EI2OS Execution Time = (36 + 6 × value of Table 7.4.3a) machine cycles

(3) When the counting is completed

EI2OS Execution Time = (value of Table 7.6.4a + value of Table 7.6.4b + (21 + 6 × value of Table 7.4.3a) machine cycles

Table 7.6.4b Data transfer compensation values for extended I2OS execution time

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

B : Byte data transfer
E : Even address word transfer
8 : 8-bit external bus word transfer
O : Odd address word transfer

7.7 Exceptions

The F^{2}MC-16LX performs exception processing when the following event occurs:

- Execution of an undefined instruction

Exception processing is fundamentally the same as interrupt processing. When an exception is detected between instructions, exception processing is performed separately from ordinary processing. In general, exception processing is performed as a result of an unexpected operation. Fujitsu recommends using exception processing only for debugging or for activating emergency recovery software.

7.7.1 Exception due to execution of an undefined instruction

The F^{2}MC-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 performed. Specifically, the AL, AH, DPR, DTB, ADB, PCB, PC, and PS values are saved into the system stack, and processing branches to the routine indicated by the interrupt number 10 vector. In addition, the I flag is cleared and the S flag is set. The PC value saved in the stack is the address at which the undefined instruction is stored. Processing can be restored by the RETI instruction, but is of no use, however, because the same exception occurs again.

Chapter 8:
Parallel Ports

8.1 Outline

In MB90580 series, there are 10 parallel ports which are as follows:

- Port 0 (8 CMOS I/O pins)
- Port 1 (8 CMOS I/O pins)
- Port 2 (8 CMOS I/O pins)
- Port 3 (8 CMOS I/O pins)
- Port 4 (8 CMOS I/O pins with open-drain control)
- Port 5 (8 CMOS I/O pins)
- Port 6 (6 CMOS I/O pins)
- Port 7 (4 CMOS I/O pins)
- Port 8 (8 CMOS I/O pins)
- Port 9 (8 CMOS I/O pins)
- Port A (3 CMOS I/O pins)

Each pin of the ports can be specified as input or output using the direction register if the corresponding peripheral does not use the pin. When a pin is specified as input, the value of the pin level is read from a data register. When a pin is specified as output, the data register latch value is read from the data register. The above also applies to a read operation for a read-modify-write instruction.

When a data register is read while the corresponding port is used as a control output, control output value is read from the data register regardless of the direction register value.

When an input pin is changed into an output pin, care must be taken to use a read-modify-write instruction (such as a bit set instruction) to set output data in the data register beforehand. In this case, the data input from the pin is read instead of the data register latch value.

8.2 Block Diagram

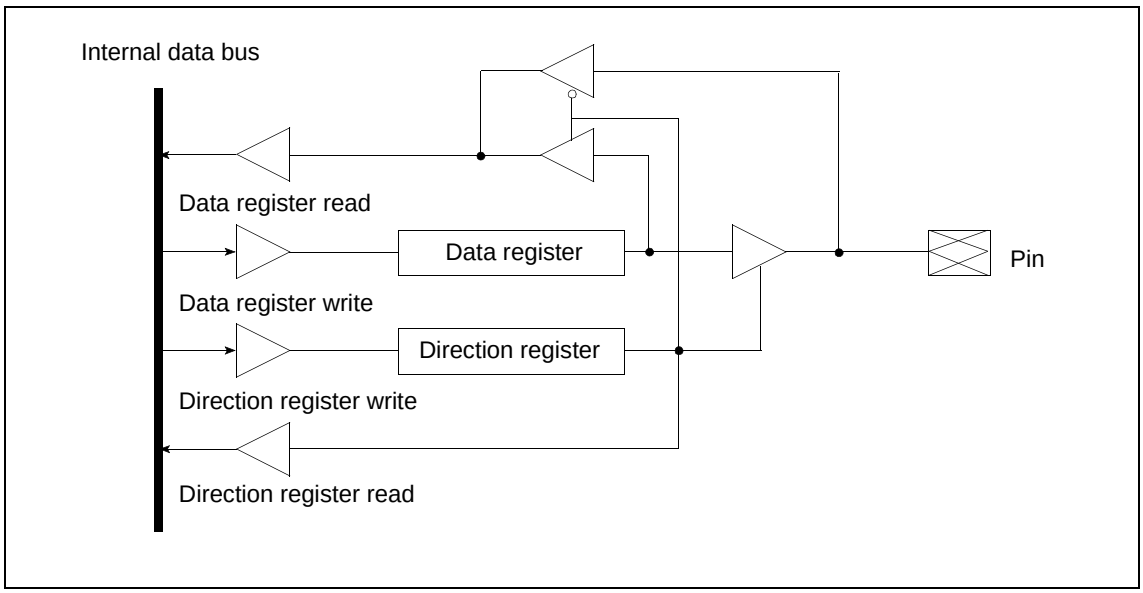


Figure 8.2a Block diagram of I/O port

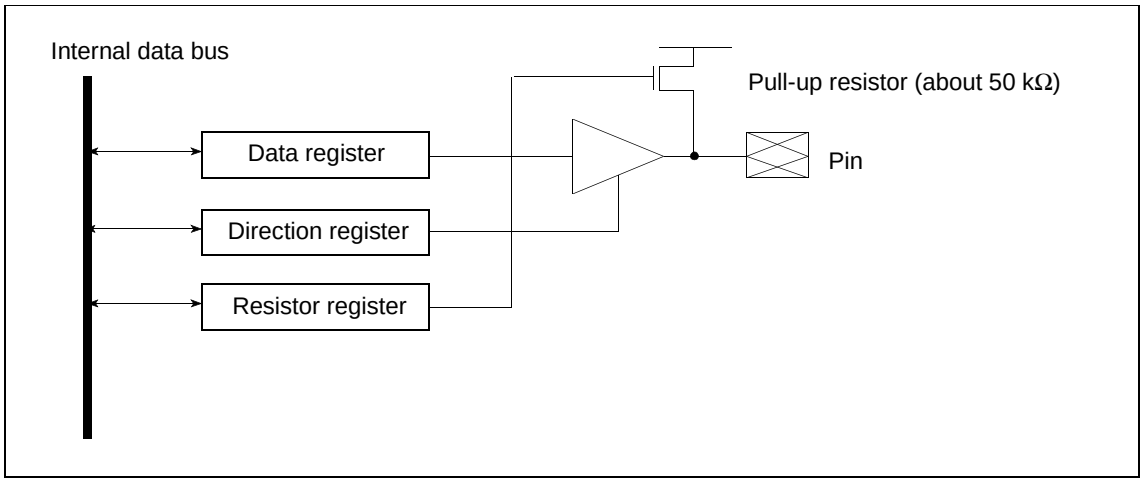


Figure 8.2b Block diagram of input resistor register

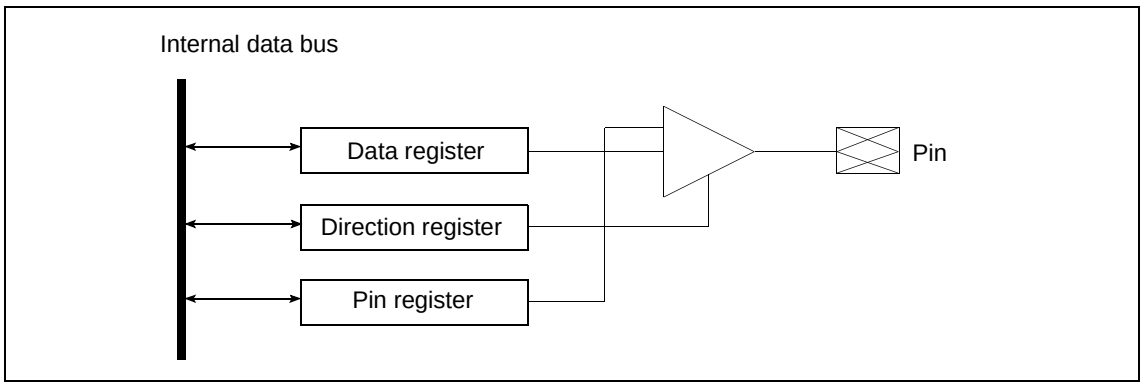


Figure 8.2c Block diagram of Output pin register

8.3 Registers and register details

	Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000000_H		P07	P06	P05	P04	P03	P02	P01	P00	Port 0 data register (PDR0)
Address : 000001_H		P17	P17	P15	P14	P13	P12	P11	P10	Port 1 data register (PDR1)
Address : 000002_H		P27	P26	P25	P24	P23	P22	P21	P20	Port 2 data register (PDR2)
Address : 000003_H		P37	P36	P35	P34	P33	P32	P31	P30	Port 3 data register (PDR3)
Address : 000004_H		P47	P46	P45	P44	P43	P42	P41	P40	Port 4 data register (PDR4)
Address : 000005_H		P57	P56	P55	P54	P53	P52	P51	P50	Port 5 data register (PDR5)
Address : 000006_H		—	—	P65	P64	P63	P62	P61	P60	Port 6 data register (PDR6)
Address : 000007_H		—	—	—	P74	P73	P72	P71	—	Port 7 data register (PDR7)
Address : 000008_H		P87	P86	P85	P84	P83	P82	P81	P80	Port 8 data register (PDR8)
Address : 000009_H		P97	P96	P95	P94	P93	P92	P91	P90	Port 9 data register (PDR9)
Address : $00000A_H$		—	—	—	—	—	PA2	PA1	PA0	Port A data register (PDRA)
	Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000010_H		D07	D06	D05	D04	D03	D02	D01	D00	Port 0 data register (DDR0)
Address : 000011_H		D17	D17	D15	D14	D13	D12	D11	D10	Port 1 data register (DDR1)
Address : 000012_H		D27	D26	D25	D24	D23	D22	D21	D20	Port 2 data register (DDR2)
Address : 000013_H		D37	D36	D35	D34	D33	D32	D31	D30	Port 3 data register (DDR3)
Address : 000014_H		D47	D46	D45	D44	D43	D42	D41	D40	Port 4 data register (DDR4)
Address : 000015_H		D57	D56	D55	D54	D53	D52	D51	D50	Port 5 data register (DDR5)
Address : 000016_H		—	—	D65	D64	D63	D62	D61	D60	Port 6 data register (DDR6)
Address : 000017_H		—	—	—	D74	D73	D72	D71	—	Port 7 data register (DDR7)
Address : 000018_H		D87	D86	D85	D84	D83	D82	D81	D80	Port 8 data register (DDR8)
Address : 000019_H		D97	D96	D95	D94	D93	D92	D91	D90	Port 9 data register (DDR9)
Address : $00001A_H$		—	—	—	—	—	DA2	DA1	DA0	Port A data register (DDRA)
	Bit	15	14	13	12	11	10	9	8	
Address : $00001B_H$		OD47	OD46	OD45	OD44	OD43	OD42	OD41	OD40	Port 4 pin register (ODR4)
	Bit	7	6	5	4	3	2	1	0	
Address : $00001C_H$		ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 5 analog input enable register (ADER)
	Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : $00008C_H$		RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	Port 0 resistor register (RDR0)
Address : $00008D_H$		RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	Port 1 resistor register (RDR1)
Address : $00008E_H$		—	—	RD65	RD64	RD63	RD62	RD61	RD60	Port 6 resistor register (RDR6)
	Bit	15	14	13	12	11	10	9	8	
Address : $0000A3_H$		—	—	—	—	LNB	LNA	LN9	LN8	Low Noise Output Select register (Upper) (LNSRH)
	Bit	7	6	5	4	3	2	1	0	
Address : $0000A2_H$		LN7	LN6	LN5	LN4	LN3	LN2	LN1	LN0	Low Noise Output Select register (Lower) (LNSRL)

Figure 8.3a Registers of Parallel Ports

8.3.1 Port data register

	7	6	5	4	3	2	1	0	Initial value	Access
PDR0 Address: 000000_H	P07	P06	P05	P04	P03	P02	P01	P00	xxxxxxxx	R/W
	15	14	13	12	11	10	9	8		
PDR1 Address: 000001_H	P17	P16	P15	P14	P13	P12	P11	P10	xxxxxxxx	R/W
	7	6	5	4	3	2	1	0		
PDR2 Address: 000002_H	P27	P26	P25	P24	P23	P22	P21	P20	xxxxxxxx	R/W
	15	14	13	12	11	10	9	8		
PDR3 Address: 000003_H	P37	P36	P35	P34	P33	P32	P31	P30	xxxxxxxx	R/W
	7	6	5	4	3	2	1	0		
PDR4 Address: 000004_H	P47	P46	P45	P44	P43	P42	P41	P40	xxxxxxxx	R/W
	15	14	13	12	11	10	9	8		
PDR5 Address: 000005_H	P57	P56	P55	P54	P53	P52	P51	P50	xxxxxxxx	R/W
	7	6	5	4	3	2	1	0		
PDR6 Address: 000006_H	—	—	P65	P64	P63	P62	P61	P60	--xxxxxx	R/W
	15	14	13	12	11	10	9	8		
PDR7 Address: 000007_H	—	—	—	P74	P73	P72	P71	—	---xxxx-	R/W
	7	6	5	4	3	2	1	0		
PDR8 Address: 000008_H	P87	P86	P85	P84	P83	P82	P81	P80	xxxxxxxx	R/W
	15	14	13	12	11	10	9	8		
PDR9 Address: 000009_H	P97	P96	P95	P94	P93	P92	P91	P90	xxxxxxxx	R/W
	7	6	5	4	3	2	1	0		
PDRA Address: $00000A_H$	—	—	—	—	—	PA2	PA1	PA0	-----xxx	R/W

Note: Note that R/W for I/O ports differ from R/W for memory in the following points:

- Input mode

  Read: The level of the corresponding pin is read.

  Write: Data is written to an output latch.

- Output mode

  Read: The data register latch value is read.

  Write: The data is output to the corresponding pin.

8.3.2 Port direction registers

	7	6	5	4	3	2	1	0	Initial value	Access
DDR0 Address: 000010_H	D07	D06	D05	D04	D03	D02	D01	D00	00000000_B	R/W

	15	14	13	12	11	10	9	8		
DDR1 Address: 000011_H	D17	D16	D15	D14	D13	D12	D11	D10	00000000_B	R/W

	7	6	5	4	3	2	1	0		
DDR2 Address: 000012_H	D27	D26	D25	D24	D23	D22	D21	D20	00000000_B	R/W

	15	14	13	12	11	10	9	8		
DDR3 Address: 000013_H	D37	D36	D35	D34	D33	D32	D31	D30	00000000_B	R/W

	7	6	5	4	3	2	1	0		
DDR4 Address: 000014_H	D47	D46	D45	D44	D43	D42	D41	D40	00000000_B	R/W

	15	14	13	12	11	10	9	8		
DDR5 Address: 000015_H	D57	D56	D55	D54	D53	D52	D51	D50	00000000_B	R/W

	7	6	5	4	3	2	1	0		
DDR6 Address: 000016_H	—	—	D65	D64	D63	D62	D61	D60	$--000000_B$	R/W

	15	14	13	12	11	10	9	8		
DDR7 Address: 000017_H	—	—	—	D74	D73	D72	D71	—	$- - - 0000-_B$	R/W

	7	6	5	4	3	2	1	0		
DDR8 Address: 000018_H	D87	D86	D85	D84	D83	D82	D81	D80	00000000_B	R/W

	15	14	13	12	11	10	9	8		
DDR9 Address: 000019_H	D97	D96	D95	D94	D93	D92	D91	D90	00000000_B	R/W

	7	6	5	4	3	2	1	0		
DDRA Address: $00001A_H$	—	—	—	—	—	DA2	DA1	DA0	$- - - - -000_B$	R/W

When a pin is used as a port, the corresponding pin is controlled as described below:

0	Input mode [initial value]
1	Output mode

8.3.3 Output pin register

Port 4 pin register

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 00001B$_H$	OD47	OD46	OD45	OD44	OD43	OD42	OD41	OD40	ODR4
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 controls the open drain in output mode.

0	Standard output port in output mode [initial value]
1	Open drain output port in output mode

Note: This register is not used in input mode. (Output Hi-z)

Note: Input or output mode is determined by the direction register (DDR).

Note: No pull-up resistor is used during hardware standby and stop (SPL=1). (High impedance)

Note: This function is inhibited when an external bus is used. When using an external bus, do not write data in this register.

8.3.4 Input resistor register

Port 0 resistor register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 00008C$_H$	RD07	RD06	RD05	RD04	RD03	RD02	RD01	RD00	RDR0
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	

Port 1 resistor register

	15	16	15	14	13	12	11	10	⇦ Bit number
Address : 00008D$_H$	RD17	RD16	RD15	RD14	RD13	RD12	RD11	RD10	RDR1
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	

Port 6 resistor register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 00008E$_H$	—	—	RD65	RD64	RD63	RD62	RD61	RD60	RDR6
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	

This register controls whether to use a pull-up resistor in input mode.

0	No pull-up resistor used in input mode. [initial value]
1	Pull-up resistor used in input mode.

Note: This register has no use in output mode (no pull-up resistor is used).

Note: Input or output mode is determined by the direction register (DDR).

Note: No pull-up resistor is used during hardware standby and stop (SPL=1). (High impedance)

Note: This function is inhibited when an external bus is used. When using an external bus, do not write data in this register.

8.3.5 Analogue Input Enable Register

Port 5 analogue enable register

	7	6	5	4	3	2	1	0	
									⇦ Bit number
Address : 00001C$_H$	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	ADER
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	

This register controls the .behaviour of port 5.

0	Port input mode
1	Analogue input mode [initial value]

Note: When an intermediate voltage level is applied to the pin during port input mode, a leakage current will be induced. In this case, configure the pin to analogue input mode instead.

8.3.6 Low Noise Output Select Register

Low Noise Output Select register (Upper)

	15	14	13	12	11	10	9	8	
									⇦ Bit number
Address : 0000A3$_H$	—	—	—	—	LNB	LNA	LN9	LN8	LNSRH
Read/write ⇨	—	—	—	—	R/W	R/W	R/W	R/W	
Initial value ⇨	—	—	—	—	0	0	0	0	

Low Noise Output Select register (Lower)

	7	6	5	4	3	2	1	0	
									⇦ Bit number
Address : 0000A2$_H$	LN7	LN6	LN5	LN4	LN3	LN2	LN1	LN0	LNSRL
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	

These two register are used to select the low noise output buffer for Port 0 to Port A and the TX output of IE bus.

[bit 15 - 12] - unused bits

[bit 11] - LNB controls TX pin of IE bus

0	Normal output buffer [initial value]
1	Low noise output buffer

[bit 10] - LNA controls Port A

0	Normal output buffer [initial value]
1	Low noise output buffer

[bit 9] - LN9 controls Port 9

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 8] - LN8 controls Port 8

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 7] - LN7 controls Port 7

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 6] - LN6 controls Port 6

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 5] - LN5 controls Port 5

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 4] - LN4 controls Port 4

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 3] - LN3 controls Port 3

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 2] - LN2 controls Port 2

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 1] - LN1 controls Port 1

0	Normal output buffer	[initial value]
1	Low noise output buffer	

[bit 0] - LN0 control Port 0

0	Normal output buffer	[initial value]
1	Low noise output buffer	

Note: These two register are not available for MB90V580.
Note: When low noise output buffer is selected, the driving power will be decreased.

Chapter 9:
DTP/External Interrupt

9.1 Outline

The DTP (Data Transfer Peripheral) is a peripheral block that interfaces external peripherals to the F^2MC-16LX CPU. The DTP receives DMA and interrupt processing requests from external peripherals and passes requests to the F^2MC-16LX CPU to activate the extended intelligent I/O service (EI^2OS) or interrupt processing. Two request levels ("H" and "L") are provideed for the extended intelligent I/O service (EI^2OS). For external interrupt requests, generating interrupts on a rising edge or falling edge as well as "H" and "L" level can be selected, giving a total of four types.

9.2 Block Diagram

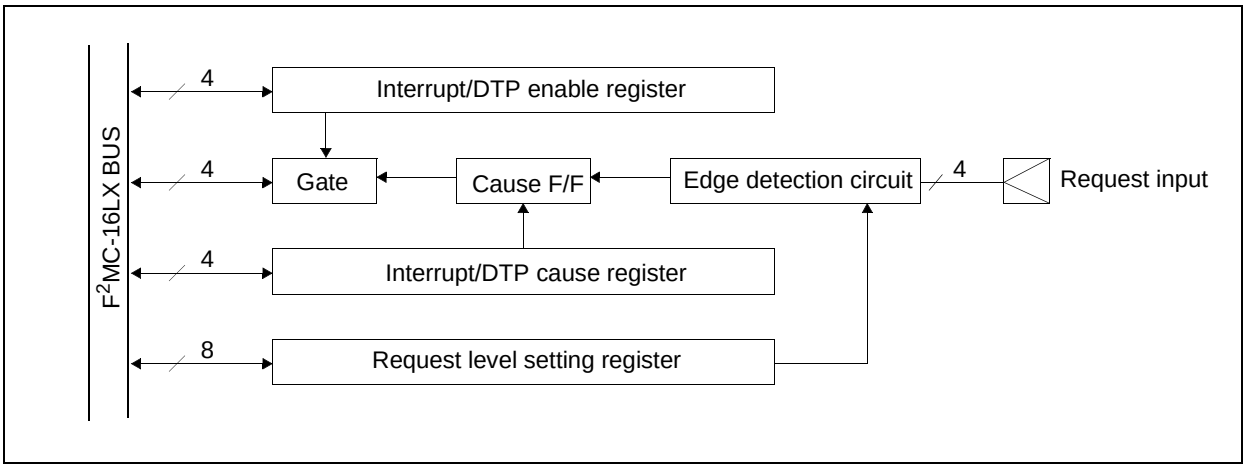
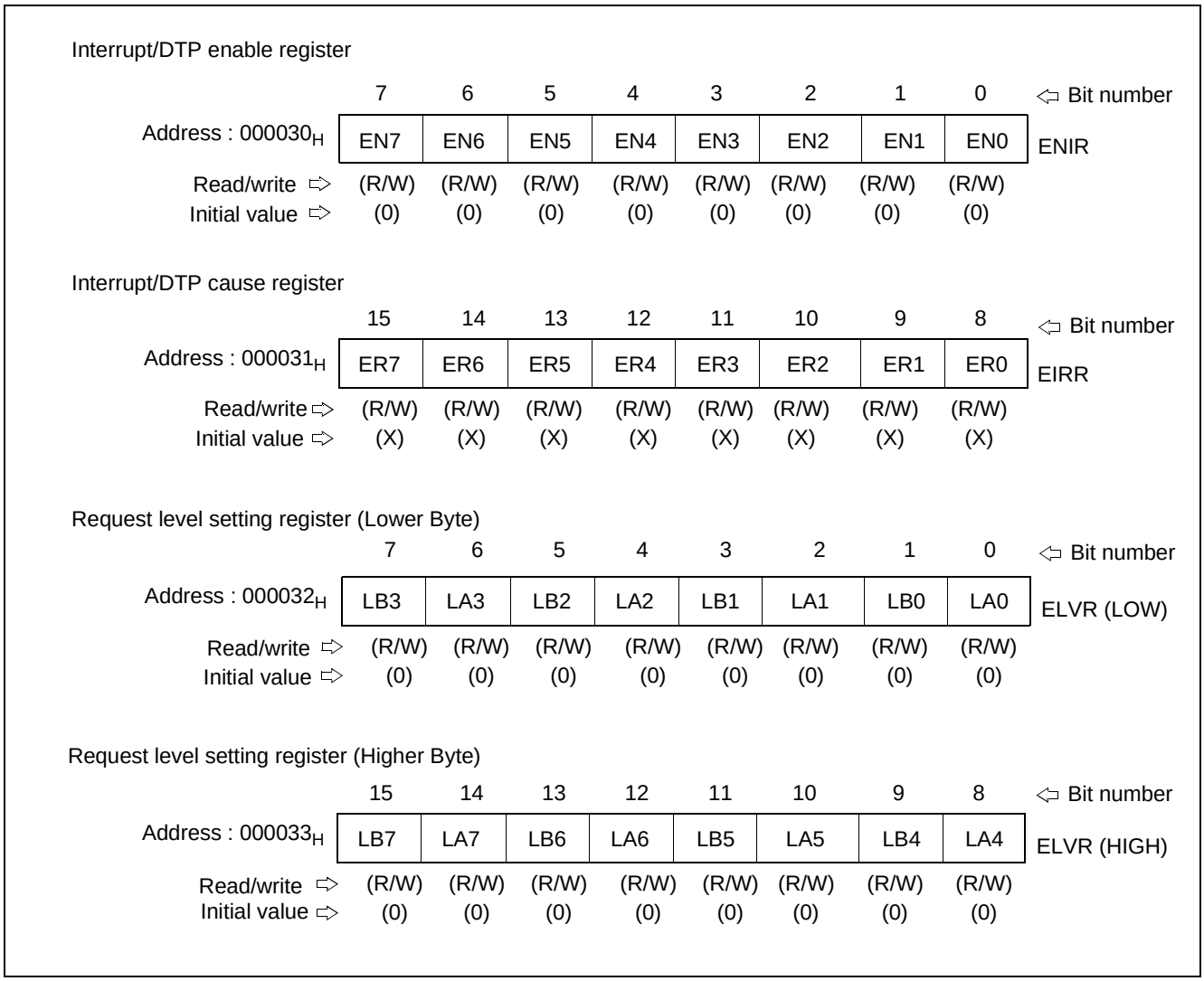
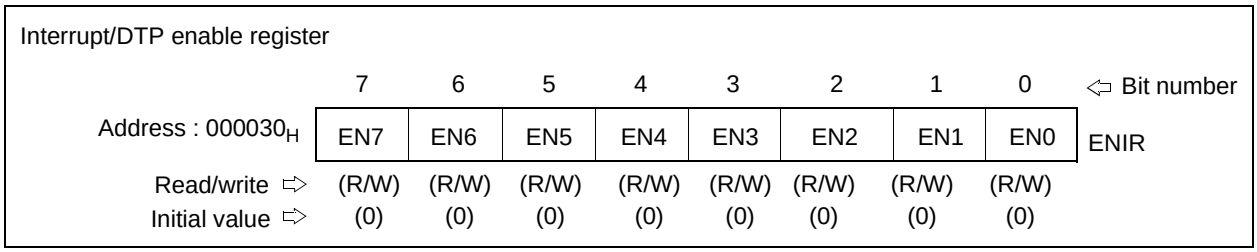


Figure 9.2a Block diagram of DTP/External Interrupt

9.3 Registers and Register Details



9.3.1 Interrupt/DTP enable register (ENIR: Enable interrupt request register)



ENIR enables the function to issue a request to the interrupt controller using a device pin as an external interrupt/DTP request input. A pin corresponding to a '1' bit of this register is used as an external interrupt/DTP request input. A pin corresponding to a '0' bit holds the external interrupt/DTP request input cause, but does not issue a request to the interrupt controller.

9.3.2 Interrupt/DTP cause register (EIRR: External interrupt request register)

Interrupt/DTP cause register

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 000031_H	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	EIRR
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)	

When read, EIRR indicates the current external interrupt/DTP requests. When written, EIRR clears the flip-flop values indicating those requests. External interrupt/DTP requests exist at the pins corresponding to the '1' bits of this register. Writing '0' to a bit of this register clears the corresponding request flip-flop value. Writing '1' performs no operation. '1' is always read from this register by a read-modify-write instruction.

9.3.3 Request level setting register (ELVR: External level register)

Request level setting register (Lower Byte)

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 000032_H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	ELVR (LOW)
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)	

Request level setting register (Higher Byte)

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 000033_H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	ELVR (HIGH)
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)	

ELVR is used to select a request detection factor. Each pin is assigned two bits as described in the table below. If a request is to be detected based on a level, the register value is maintained while the input is active even when it is cleared.

LBx	LAx	Interrupt request detection factor
0	0	L level pin input
0	1	H level pin input
1	0	Rising edge pin input
1	1	Falling edge pin input

9.4 Operations

9.4.1 External interrupts

Once an external interrupt request is set, this resource issues an interrupt request signal to the interrupt controller when a request specified by the ELVR register is input to the corresponding pin. The interrupt controller identifies the priority levels of the simultaneous interrupts, and issues an interrupt request to the F^{2}MC-16 CPU if the interrupt from this resource has the highest priority level. The F^{2}MC-16 CPU compares the ILM bit of its internal CCR register and the interrupt request. If the interrupt level of the request is higher than that indicated by the ILM bit, the F^{2}MC-16 CPU activates the hardware interrupt processing microprogram as soon as the currently executing instruction is terminated.

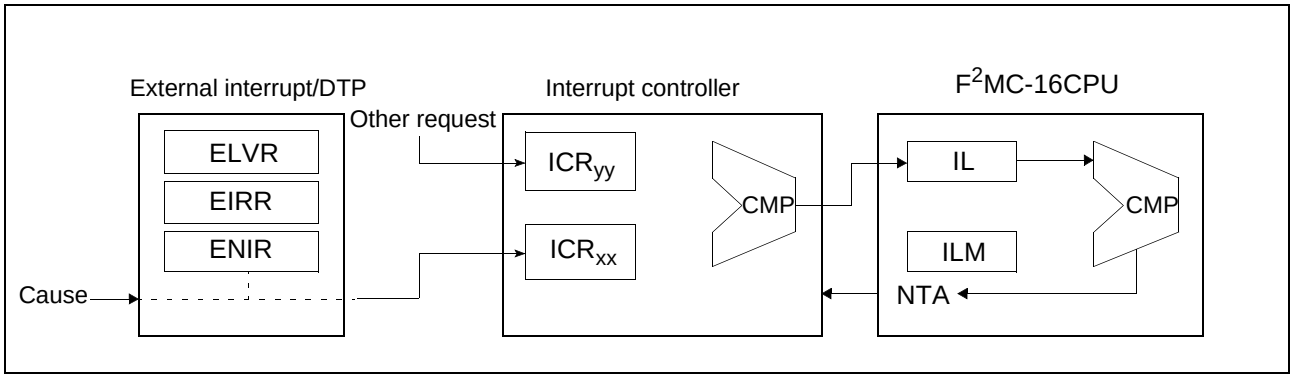


Figure 9.4.1a External interrupt

In the hardware interrupt processing microprogram, the CPU reads the ISE bit information from the interrupt controller, identifies that the request is for interrupt processing based on that information, and branches to the interrupt processing microprogram. The interrupt processing microprogram reads the interrupt vector area and issues an interrupt acknowledgment signal for the interrupt controller. Then, the microprogram transfers the jump destination address of the macro instruction generated from the vector to the program counter, and executes the user interrupt processing program.

9.4.2 DTP operation

To activate the intelligent I/O service, the user program initially sets the address of a register, assigned between 000000_H and $0000FF_H$, in the I/O address pointer of the intelligent I/O service descriptor. Then, the user program sets the start address of the memory buffer in the buffer address pointer.

The DTP operation sequence is almost the same as for external interrupts. The operation is identical until the CPU activates the hardware interrupt processing microprogram. Then, for the DTP, control is transferred to the intelligent I/O service processing microprogram, since the ISE bit read by the CPU within the hardware interrupt processing microprogram indicates the DTP. Once the intelligent I/O service is activated, a read or write signal is sent to the addresses external peripheral, and data is transferred between the peripheral and the chip. The external peripheral must cancel the interrupt request to this chip within three machine cycles after the transfer is made. When the transfer is completed, the descriptor is updated, and the interrupt controller generates a signal that clears the transfer cause. Upon receiving the signal to clear the transfer cause, this resource clears the flip-flop holding the cause and prepares for the next request from the pin. For details of the intelligent I/O service processing, refer to the MB90700 Programming Manual.

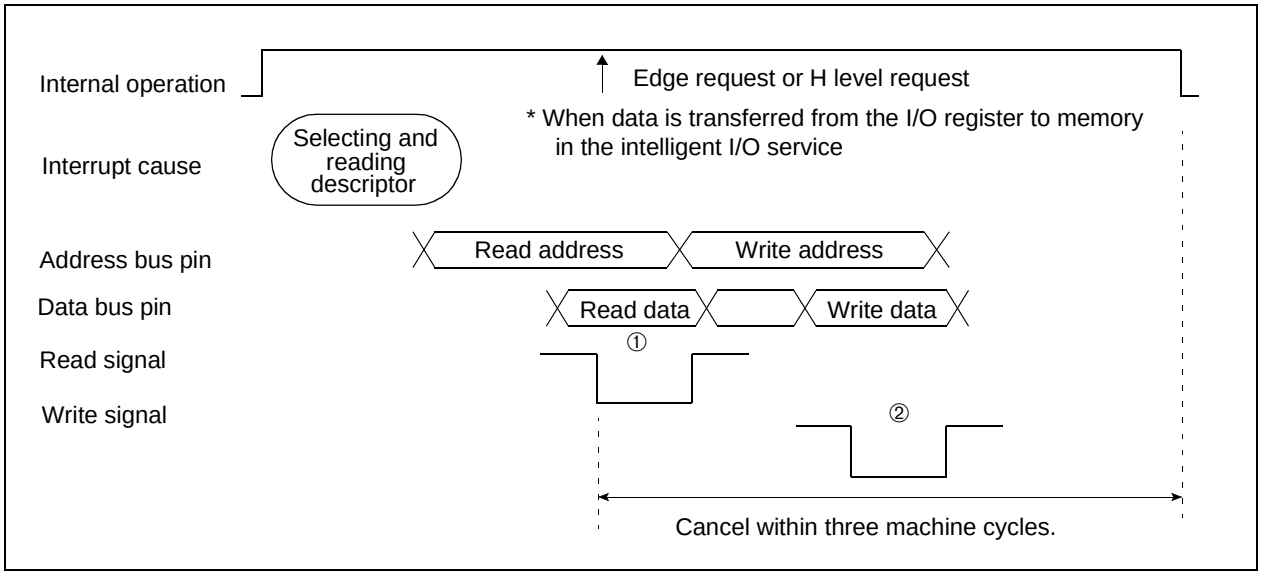


Figure 9.4.2a Timing to cancel the external interrupt at the end of DTP operation

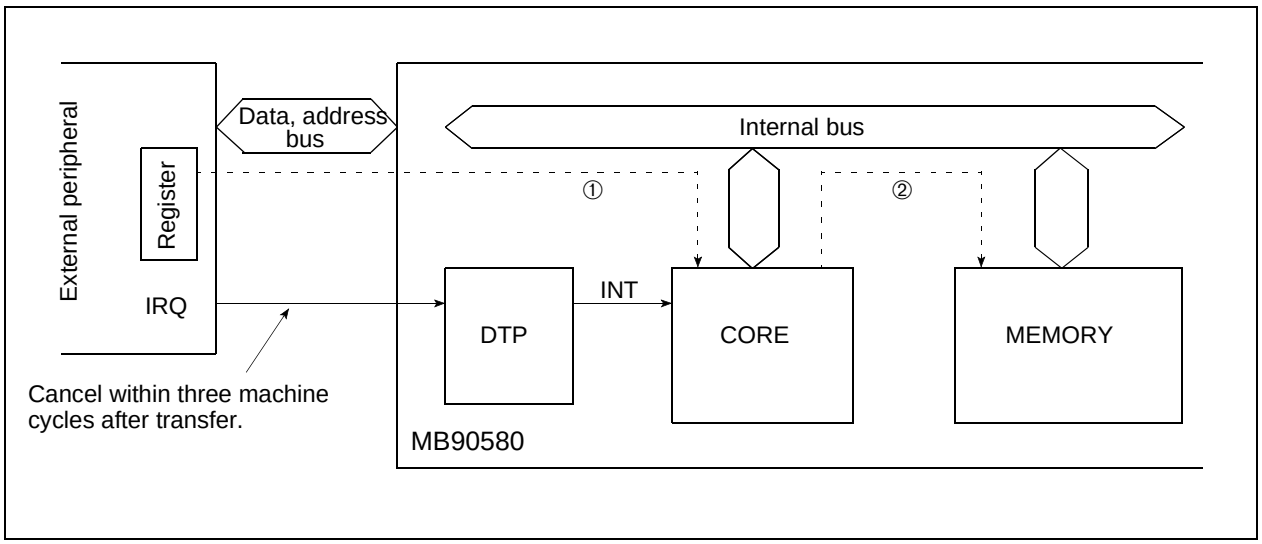


Figure 9.4.2b Sample interface to the external peripheral

9.4.3 Switching between external interrupt and DTP requests

To switch between external interrupt and DTP requests, use the ISE bit in the ICR register corresponding to this resource, which is in the interrupt controller. Each pin is individually assigned ICR. Thus, a pin is used for a DTP request if ’1’ is written to the ISE bit of the corresponding ICR, and is used for an external interrupt request if ’0’ is written to the bit.

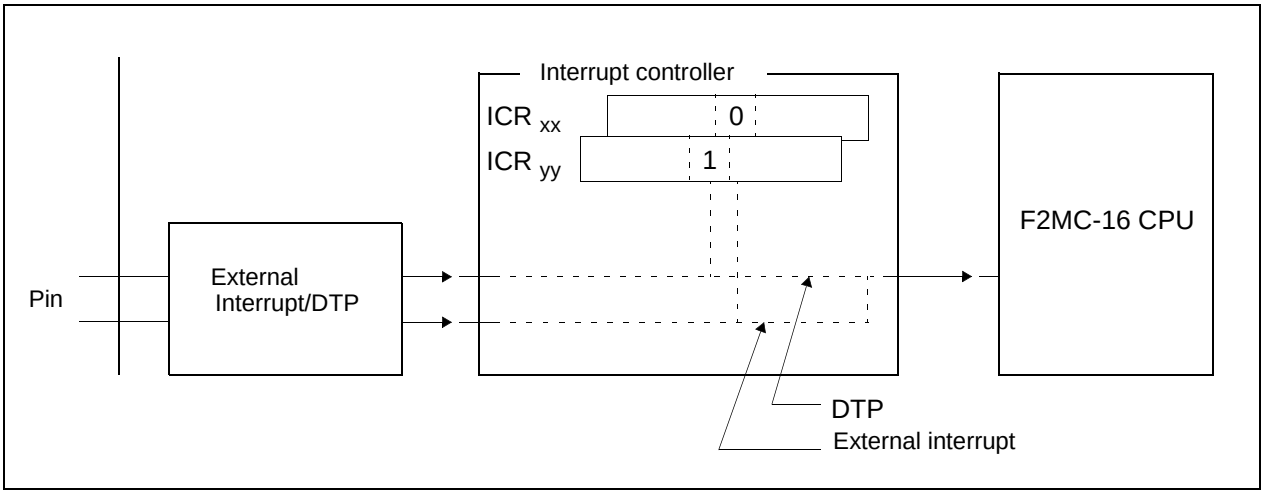


Figure 9.4.3a Switching between external interrupt and DTP requests

9.5 Notes on use

9.5.1 Conditions on the externally connected peripheral when DTP is used

DTP supports only external peripherals that automatically clear a request once a transfer is completed. The system must be designed so that a transfer request is canceled within three machine cycles (provisional) after transfer operation starts. Otherwise, this resource assumes that a transfer request is issued.

9.5.2 Recovery from standby

To use an external interrupt to recover from the standby state in clock stop mode, use an H level request as an input request. A L level request may result in misoperation. If an edge request is used, recovery from the standby state in clock stop mode cannot be performed.

9.5.3 External interrupt/DTP operation procedure

To set registers in the external interrupt/DTP, follow the steps below:

1. Disable the bits corresponding to the enable register.
2. Set the bits corresponding to the request level setting register.
3. Clear the bits corresponding to the cause register.
4. Enable the bits corresponding to the enable register.

(Steps 3. and 4. can be simultaneously performed by word specification.)

To set a register in this resource, ensure that the enable register is disabled. Before enabling the enable register, ensure that the cause register is cleared. Clearing the cause register prevents a false interrupt cause from being determined while registers are set or interrupts are enabled.

9.5.4 External interrupt request level

①To detect an edge for a edge request level, the pulse width must be at least three machine cycles.

②If the request input level is related to level setting, the request to the interrupt controller is kept active. Because of the internal hold circuit, the request is kept active even if it is input from the external device and then canceled. To cancel the request to the interrupt controller, clear the cause hold circuit.

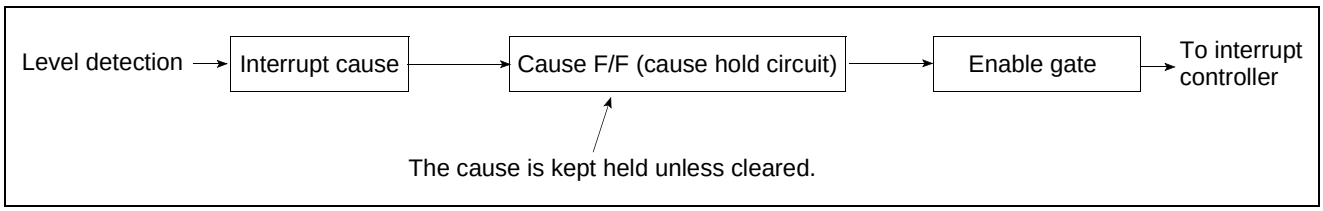


Figure 9.5.4a Clearing the cause hold circuit upon level set

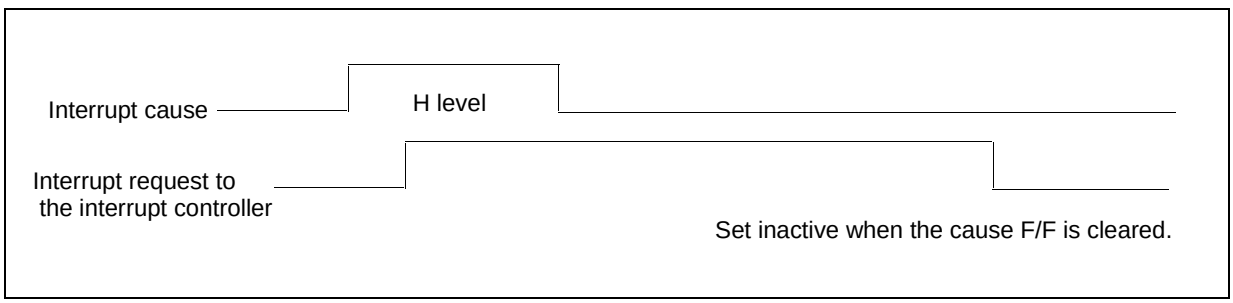


Figure 9.5.4b Interrupt cause and interrupt request to the interrupt controller while interrupts are enabled

Chapter 10:
Delayed Interrupt Generation Module

10.1 Outline

The delayed interrupt generation module generates interrupts for switching tasks for development on a real-time operating system (REALOS series). The module can be used to generate softwarewise generates hardware interrupt requests to the CPU and cancel the interrupts.

This module does not conform to the extended intelligent I/O service (EI^{2}OS).

10.2 Block Diagram

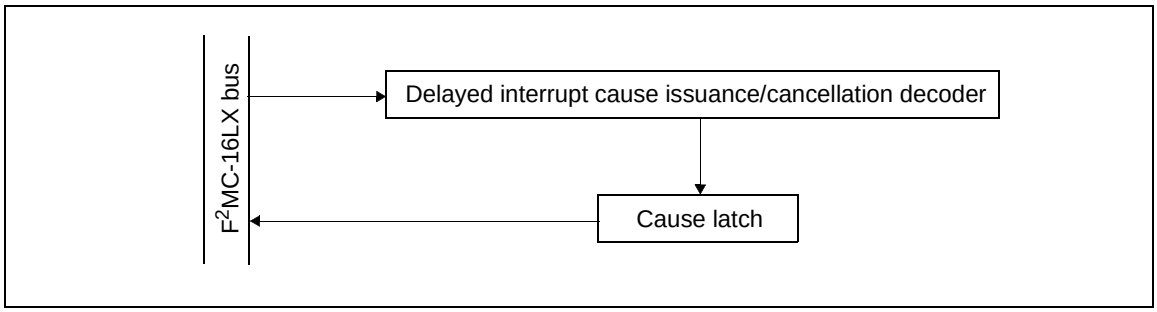


Figure 10.2a Block diagram of Delayed Interrupt Generation Module

10.3 Registers and Register Details

Delayed interrupt cause issuance/cancellation register (DIRR: Delayed interrupt request register)

Delayed interrupt cause issuance/cancellation register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 00009F$_H$	—	—	—	—	—	—	—	R0	DIRR
Read/write ⇨	(–)	(–)	(–)	(–)	(–)	(–)	(–)	(R/W)	
Initial value ⇨	(–)	(–)	(–)	(–)	(–)	(–)	(–)	(0)	

DIRR controls issuance and cancellation of delayed interrupt requests. Writing ’1’ to this register issues a delayed interrupt request, and writing ’0’ cancels the delayed interrupt request. Upon a reset, the request is canceled. Either ’0’ or ’1’ can be written to the reserved bit area. To access this register, use the set bit or clear bit instruction for future expansions.

10.4 Operations

10.4.1 Delayed interrupt occurrence

When the CPU writes ’1’ to the relevant bit of DIRR by software, the request latch in the delayed interrupt source module is set and an interrupt request is issued to the interrupt controller. If this interrupt has the highest priority or if there is no other interrupt request, the interrupt controller issues an interrupt request to the F^{2}MC-16 CPU. The F^{2}MC-16 CPU compares the ILM bit of its internal CCR register and the interrupt request, and starts the hardware interrupt processing microprogram as soon as the current instruction is completed if the interrupt level of the request is higher than that of the ILM bit. The interrupt processing routine for this interrupt is thus executed.

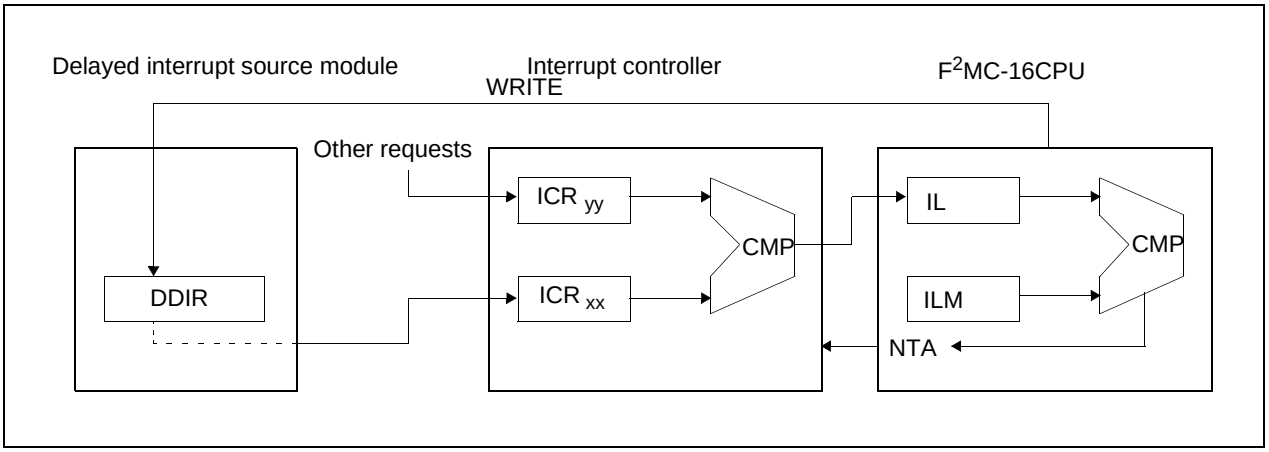


Figure 10.4.1a Delayed interrupt issuance

The interrupt cause is cleared and tasks are switched by writing ’0’ to the corresponding bit of DDIR in the interrupt processing routine.

10.5 Notes on operation

10.5.1 Delayed interrupt request lock

This lock is set by writing ’1’ to the corresponding bit of DIRR, and is cleared by writing ’0’ to the same bit. Therefore, interrupt processing is reactivated immediately after control returns from interrupt processing, unless the software is designed so that the cause of the interrupt is cleared within the interrupt processing routine.

Chapter 11:
Communication Prescaler

11.1 Outline

The operation clock for the UART is obtained by dividing the machine clock. UART is designed so that a constant baud rate can be obtained for a variety of machine clocks by the user of the communication prescaler. The Clock Division Control Register (CDCR) controls the machine clock division.

11.2 Block Diagram

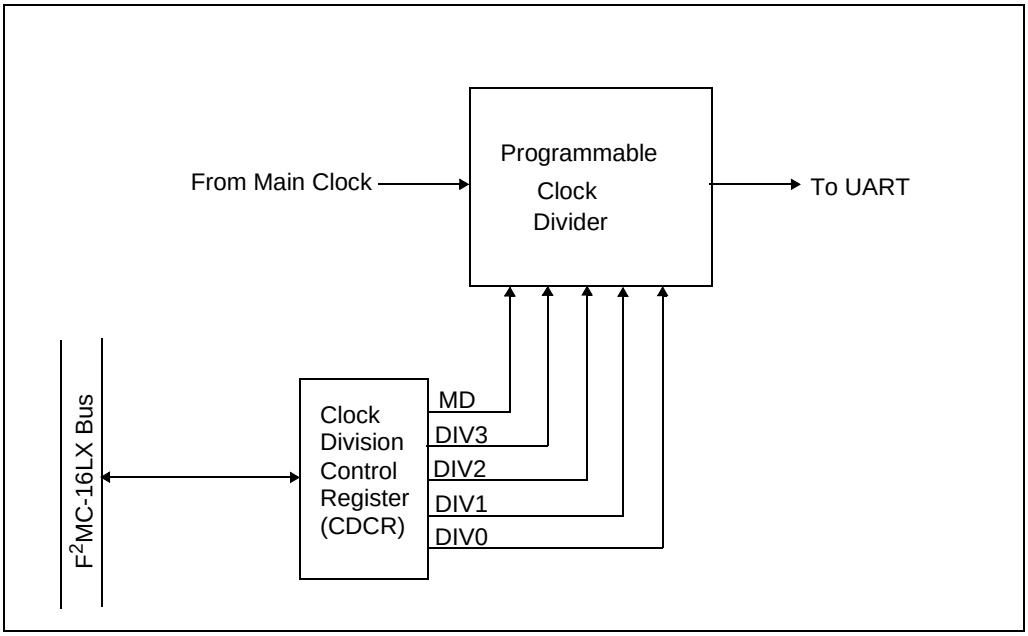


Figure 11.2a Block diagram of Communication Prescaler

11.3 Register and Register Details

11.3.1 Clock Division Control Registers

Clock Division Control Register 0, 1, 2, 3, 4

Address : $00002C_H$ $00002E_H$ 000034_H 000087_H $00008F_H$	15	14	13	12	11	10	9	8	⇦ Bit number
	MD	—	—	—	DIV3	DIV2	DIV1	DIV0	CDCR0 CDCR1 CDCR2 CDCR3 CDCR4
Read/write ⇨	(R/W)	(–)	(–)	(–)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(0)	(–)	(–)	(–)	(1)	(1)	(1)	(1)	

[bit 15] MD (Machine clock divide mode select):

This bit is used to control the operation of the communication prescaler.

0	The communication prescaler is disabled. [initial value]
1	The communication prescaler is enabled.

[bits 11, 10, 9, and 8] DIV3 to DIV0 (Divide 3 to 0):

These bits are used to determine the machine clock division ratio.

DIV3	DIV2	DIV1	DIV0	Division ratio
1	1	1	1	Reserved [initial value]
1	1	1	0	2
1	1	0	1	3
1	1	0	0	4
1	0	1	1	5
1	0	1	0	6
1	0	0	1	7
1	0	0	0	8

Note: When the division ratio is changed, allow two cycles for the clock to stabilize before starting communication.

Note: In actual application, please use the values other than ‘1111’.

11.4 Operations

Depending on the machine clock ϕ to be used, the communication prescaler register should be set as follows. For details please refer to Chapter 12, UART.

.

machine clock ϕ	div	DIV3	DIV2	DIV1	DIV0	ϕ/div
4 MHz	4	1	1	0	0	1 MHz
6 MHz	6	1	0	1	0	
8 MHz	8	1	0	0	0	
6 MHz	3	1	1	0	1	2 MHz
8 MHz	4	1	1	0	0	
10 MHz	5	1	0	1	1	
12 MHz	6	1	0	1	0	
14 MHz	7	1	0	0	1	
16 MHz	8	1	0	0	0	
8 MHz	2	1	1	1	0	4 MHz
12 MHz	3	1	1	0	1	
16 MHz	4	1	1	0	0	

When using the machine clock and the div at a different setting other than those mentioned above, ϕ/div should not exceed 4.25 MHz.

Chapter 12: UART

12.1 Outline

UART is a serial I/O port for asynchronous communications or CLK synchronous communications. UART has the following features:

- Full-duplex double buffers
- Asynchronous or CLK synchronous communications
- Multi-processor mode
- Built-in dedicated baud rate generator

  Asynchronous: 9615, 31250, 4808, 2404, 1202 bps
  CLK synchronous: 1 M, 500 K, 250 K, 125 K, 62.5 Kbps
  } (At an internal machine clock of 6, 8, 10, 12, or 16 MHz)

- Flexible baud rate setting by external clock
- Error detection (parity, framing, and overrun)
- NRZ sign transfer signals
- Intelligent I/O service

12.2 Block Diagram

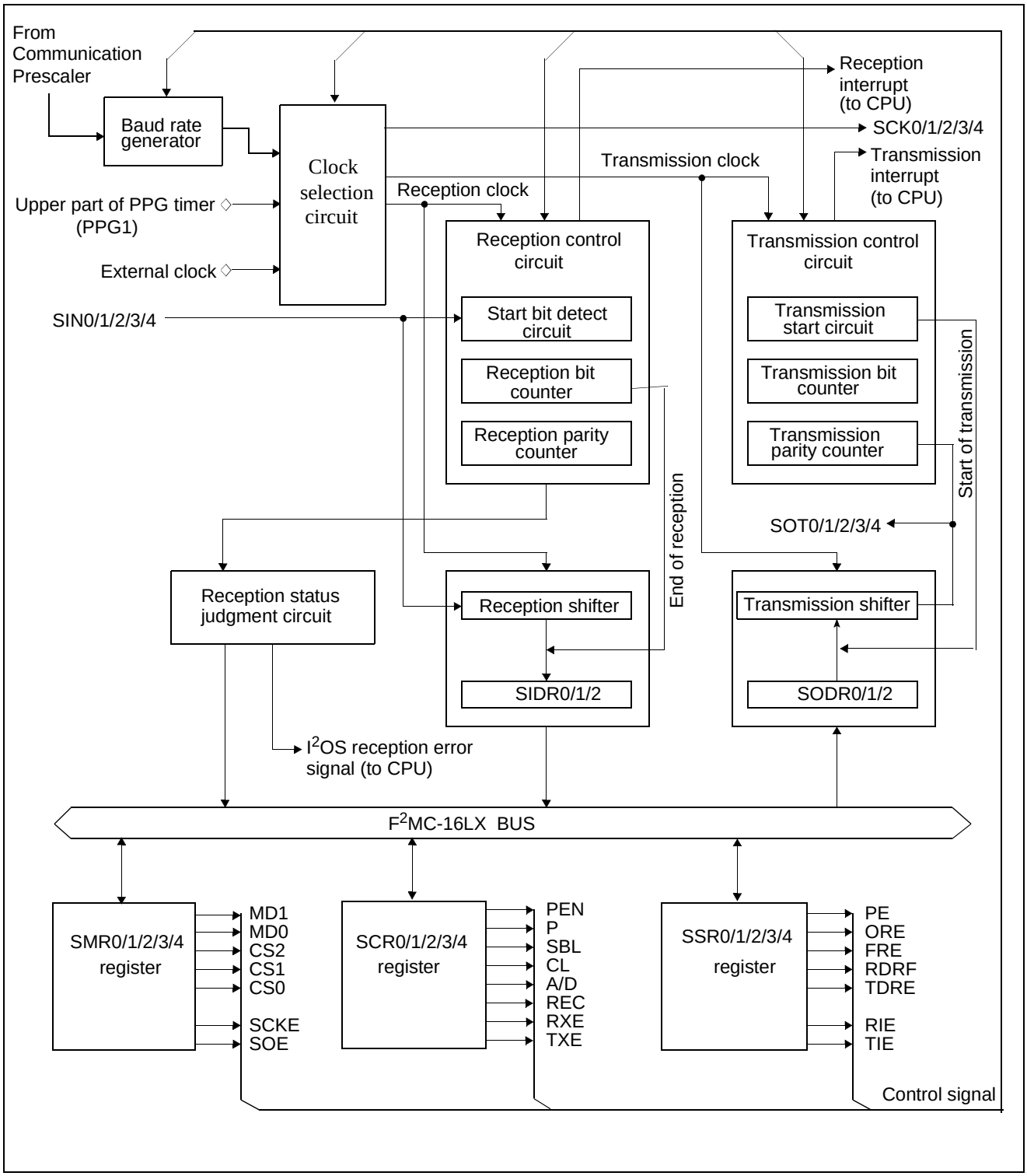


Figure 12.2a Block diagram of UART

12.3 Register and Register Details

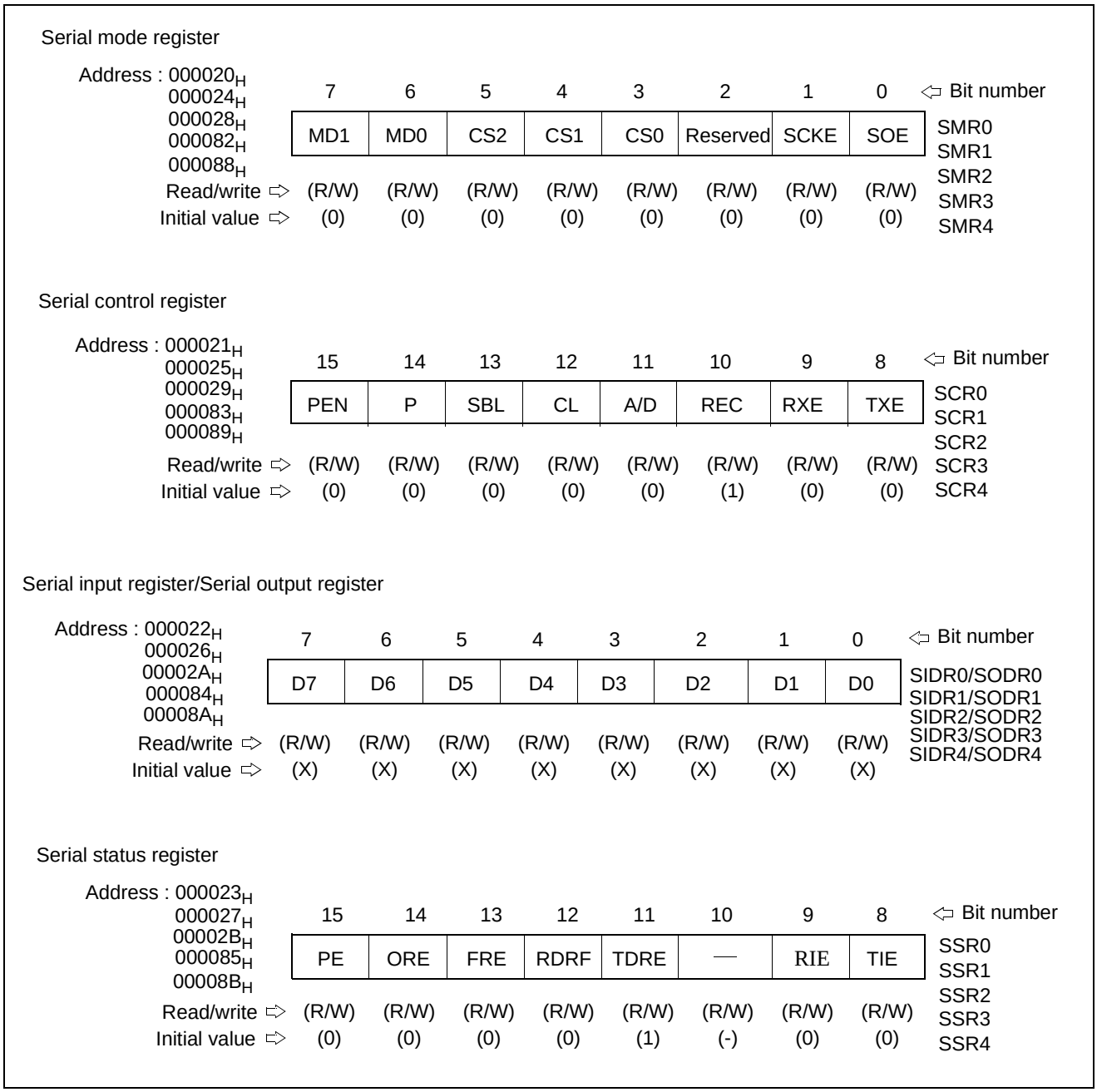


Figure 12.3a Registers of UART

12.3.1 Serial Mode Register (SMR0/1/2/3/4)

Serial mode register

Address :	7	6	5	4	3	2	1	0	⇦ Bit number
000020_H 000024_H 000028_H 000088_H 000082_H	MD1	MD0	CS2	CS1	CS0	Reserved	SCKE	SOE	SMR0 SMR1 SMR2 SMR3 SMR4
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)	

The SMR register specifies the UART operation mode. Set the operation mode while the UART is stopped. Do not write data in this register during UART operation.

[bits 7 and 6] MD1 and MD0 (Mode select):

These bits are used to select the UART operation mode.

Mode	MD1	MD0	Operation mode
0	0	0	Asynchronous normal mode
1	0	1	Asynchronous multi-processor mode
2	1	0	CLK synchronous mode
-	1	1	Setting inhibited

Note: In CLK asynchronous multi-processor mode (mode 1), two or more slave CPUs are connected to a single host CPU. This resource cannot identify the format of the received data.
Therefore, this resource only supports a master in multi-processor mode.
Since the parity check function cannot be used, write '0' to PEN of the SCR register.

[bits 5 to 3] CS2, CS1, and CS0 (Clock select):These bits are used to select the baud rate clock source.

When a dedicated baud rate generator is selected, the baud rate is determined at the same time.

CS2	CS1	CS0	Clock input
000_B to 100_B			Dedicated baud rate generator
1	0	1	reserved
1	1	0	Internal timer
1	1	1	External clock

Note: If an internal timer is selected, timer 0 is used for UART0 and UART3 in the MB90580 series.

Note: If an internal timer is selected, timer 1 is used for UART1 and UART4 in the MB90580 series.

Note: If an internal timer is selected, timer 0 is used for UART2 in the MB90580 series.

[bit 2] Reserved bit

Always write '0' to this bit.

[bit 1] SCKE (SCLK enable):

This bit is used to specify whether to use the SCK0 pin as a clock input pin or clock output pin in CLK synchronous mode (mode 2) communication.

Set '0' in this bit in CLK asynchronous mode or external clock mode.

0	The SCK0 pin is used as a clock input pin.	[initial value]
1	The SCK0 pin is used as a clock output pin.	

Note: To use the SCK0 pin as a clock input pin, an external clock source must have been selected.

[bit 0] SOE (Serial output enable):

This bit is used to specify whether the external pin is used as a serial output pin (SOT0) or I/O port pin.

0	The external pin is used as a general-purpose I/O port pin.	[initial value]
1	The external pin is used as a serial data output (SOT0) pin.	

12.3.2 Serial Control Register (SCR0/1/2/3/4)

Serial control register

Address :	15	14	13	12	11	10	9	8	⇦ Bit number
000021_H 000025_H 000029_H 000083_H 000089_H	PEN	P	SBL	CL	A/D	REC	RXE	TXE	SCR0 SCR1 SCR2 SCR3 SCR4
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)	(1)	(0)	(0)	

The SCR register controls the transfer protocol for serial communications.

[bit 15] PEN (Parity enable):

This bit is used to specify whether to perform serial data communication using a parity bit.

0	With parity	[initial value]
1	Without parity	

Note: A parity bit can be added only in normal asynchronous communication mode (mode 0). No parity bit can be added in multi-processor mode (mode 1) or CLK synchronous communication mode (mode 2).

[bit 14] P (Parity):

This bit is used to specify an even- or odd-numbered parity for data communications with parity.

0	Even-numbered parity	[initial value]
1	Odd-numbered parity	

[bit 13] SBL (Stop bit length)

This bit is used to specify the length of the stop bit, which is used as a frame end mark in asynchronous communications.

0	1 stop bit	[initial value]
1	2 stop bits	

[bit 12] CL (Character length):

This bit is used to specify the data length of each frame to be sent or received.

0	7-bit data	[initial value]
1	8-bit data	

Note: 7-bit data can be handled only in normal synchronous communication mode (mode 0). Specify 8-bit data in multi-processor mode (mode 1) or CLK synchronous communication mode (mode 2).

[bit 11> A/D (Address/data):

This bit is used to specify the data format of the frame to be sent or received in multi-processor asynchronous communication mode (mode 1).

0	Data frame [initial value]
1	Address frame

[bit 10] REC (Receiver error clear):

This bit is used to clear the SSR register error flags (PE, ORE, and FRE). Writing '1' to this bit is invalid.

'1' is always read from this bit.

[bit 9] RXE (Receiver enable):

This bit is used to control UART reception.

0	Disables reception. [initial value]
1	Enables reception.

Note: If reception is disabled while data is being received (being input to the reception shift register), reception is terminated when the reception of that frame is completed and the reception data is stored in the reception data buffer (SIDR register).

[bit 8] TXE (Transmitter enable):

This bit is used to control UART transmission.

0	Disables transmission [initial value]
1	Enables transmission.

Note: If transmission is disabled while data is being transmitted (being output from the transmission register), transmission is terminated after all data in the transmission data buffer (SODR register) has been output.

12.3.3 Serial Input Data Register (SIDR0/1/2/3/4)/ Serial Ouput Data Register (SODR0/1/2/3/4)

Serial input register/Serial output register

Address :	7	6	5	4	3	2	1	0	⇦ Bit number
000022_H 000026_H $00002A_H$ 000084_H $00008A_H$	D7	D6	D5	D4	D3	D2	D1	D0	SIDR0/SODR0 SIDR1/SODR1 SIDR2/SODR2 SIDR3/SODR3 SIDR4/SODR4
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)	

These registers are data buffer registers for transmission and reception.

When a data item is seven bits long, the high-order one bit (D7) is invalid. To write a data item in the SODR register, ensure that '1' is written to TDRE of the SSR register.

Note: Writing a data item at this address means to write it to the SODR register. Reading this address means to read the SIDR register.

12.3.4 Serial Status Register (SSR0/1/2/3/4)

Serial input register/Serial output register

Address :	7	6	5	4	3	2	1	0	⇦ Bit number
000022_H 000026_H $00002A_H$ 000084_H $00008A_H$	D7	D6	D5	D4	D3	D2	D1	D0	SIDR0/SODR0 SIDR1/SODR1 SIDR2/SODR2 SIDR3/SODR3 SIDR4/SODR4
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)	

The SSR register consists of the flags indicating the UART operation.

[bit 15] PE (Parity error)

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

To clear a set flag, write '0' to the REC bit (bit 10) of the SCR register.

When this bit is set, the data in SIDR is invalid.

0	No parity error has occurred. [initial value]
1	A parity error has occurred.

[bit 14] ORE (Overrun error):

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

To clear a set flag, write '0' to the REC bit (bit 10) of the SCR register.

When this bit is set, the data in SIDR is invalid.

0	No overrun error has occurred. [initial value]
1	An overrun error has occurred.

[bit 13] FRE (Framing error)

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

To clear a set flag, write '0' to the REC bit (bit 10) of the SCR register.

When this bit is set, the data in SIDR is invalid.

0	No framing error has occurred.	[initial value]
1	A framing error has occurred.	

[bit 12] RDRF (Receiver data register full):

This interrupt request flag indicates that the SIDR register contains received data.

This flag is set when received data is loaded into the SIDR register. This flag is automatically cleared when the SIDR register is read.

0	No received data exists.	[initial value]
1	Received data exists.	

[bit 11] TDRE (Transmitter data register empty):

This interrupt request flag indicates that transmission data can be written into the SODR register.

This flag is cleared when transmission data is written into the SODR register. Then, when the written data is loaded into the transmission shifter and transfer starts, this flag is set again, indicating the next transmission data item can be written.

0	Writing transmission data is disabled.	
1	Writing transmission data is enabled.	[initial value]

0: Writing transmission data is disabled.

1: Writing transmission data is enabled.

[bit 9] RIE (Receiver interrupt enable):

This bit is used to control reception interrupts.

0	Interrupts are disabled	[initial value]
1	Interrupts are enabled.	

Note: Reception interrupt causes include normal reception by RDRF in addition to errors due to PE, ORE, and FRE.

[bit 8] TIE (Transmitter interrupt enable):

This bit is used to control transmission interrupts.

0	Interrupts are disabled	[initial value]
1	Interrupts are enabled.	

Note: Transmission interrupt causes include transmission requests by TDRE.

12.4 Operations

12.4.1 Operation modes

Table 12.4.1a lists the operation modes of UART. The modes can be switched by setting a value in the SMR or SCR register.

Table 12.4.1a UART operation modes

Mode	Parity	Data length	Operation mode	Stop bit length
0	Yes/No	7	Asynchronous normal mode	1 bit or 2 bits
	Yes/No	8		
1	No	8 + 1	Asynchronous multi-processor mode	
2	No	8	CLK synchronous mode	No

Note: In asynchronous mode, the stop bit length can be specified for transmission only; the stop bit is always one bit long for reception. If a two-bit stop bit length is specified, the UART does not operate.

Note: When using clock synchronous mode, start bit and stop bit is not attached to the data byte.

12.4.2 UART clock selection

(1) Dedicated baud rate generator

When a dedicated baud rate generator is selected, the following baud rates are used:

Table 12.4.2a Baud rate (f indicates the machine clock.)

CS2	CS1	CS0	Asynchronous	Calculation
0	0	0	9615	$(\varnothing \div div) / (8 \times 13 \times 2)$
0	0	1	4808	$(\varnothing \div div) / (8 \times 13 \times 2^2)$
0	1	0	2404	$(\varnothing \div div) / (8 \times 13 \times 2^3)$
0	1	1	1202	$(\varnothing \div div) / (8 \times 13 \times 2^4)$
1	0	0	31250	$(\varnothing \div div) / 2^6$

CS2	CS1	CS0	CLK synchronous	Calculation
0	0	0	1 M	$(\varnothing \div div) / 2$
0	0	1	500 K	$(\varnothing \div div) / 2^2$
0	1	0	250 K	$(\varnothing \div div) / 2^3$
0	1	1	125 K	$(\varnothing \div div) / 2^4$
1	0	0	62.5 K	$(\varnothing \div div) / 2^5$

Note: ∅ : Machine clock
div: Division ration
Please refer to chaper 11, Communication Prescaler.

(2) Internal timer

When ’110’ is set in CS2 to CS0 and an internal timer is selected, the 16-bit timer (timer 0) is used in reload mode. The baud rate is calculated as described below in this case:

Asynchronous: $(\varnothing \div N)/(16 \times 2 \times (n+1))$

CLK synchronous: $(\varnothing \div N)/(2 \times (n+1))$

$\varnothing$: Machine clock

N: Timer count clock sourc

n: Timer reload value

Table 12.4.2b lists the baud rates and reload values (decimal) at a machine clock of 7,3728 MHz

.

Table 12.4.2b Baud rates and reload values

	N=2^1 (Machine clock/2)	**N=2^3 (Machine clock/8)**
38400	2	——
19200	5	——
9600	11	2
4800	23	5
2400	47	11
1200	95	23
600	191	47
300	383	95

When an internal timer (16-bit timer 0) is selected as a baud rate clock source, the 16-bit timer 0 output (TOUT0) is connected inside the controller. Therefore, externally connecting the 16-bit timer 0 external pin (TOT0) to the external clock input pin (SCK0) of the UART is unnecessary. If the timer 0 output pin is not used for other purposes, it can be used as an I/O port pin.

(3) External clock

When ’111’ is set in CS2 to CS0 and an external clock is selected, the baud rate can be expressed as described below, assuming f to be the external clock frequency:

Asynchronous: f/16

CLK synchronous: f

Note: f is up to 1 MHz.

12.4.3 Asynchronous mode

(1) Transfer data format

UART handles NRX (non return to zero) format data only. Figure 12.4.3a gives the data format.

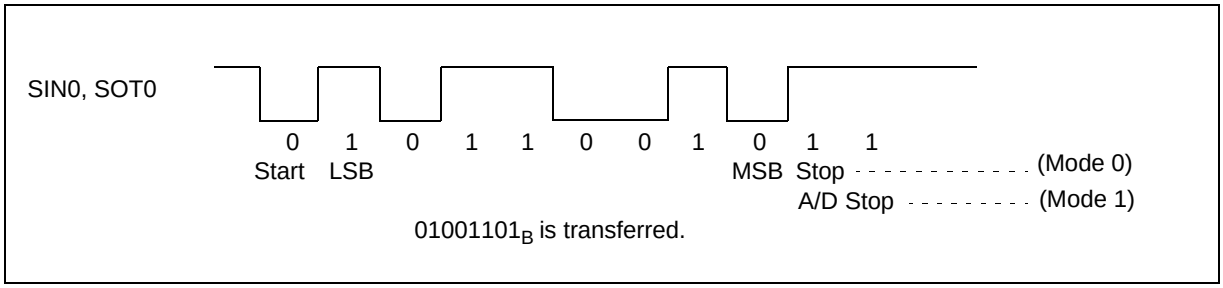


Figure 12.4.3a Transfer data format (modes 0 and 1)

As shown in Figure 12.4.3a, the transfer data always starts from the start bit ('L' level data), transfer is based on the data bit length specified by the first LSB, and transfer ends at the stop bit ('H' level data). When external clock is selected, ensure that the clock is input.

In normal mode (mode 0), the data length can be 7 or 8 bits. In multi-processor mode (mode 1), the data length must be 8 bits. In multi-processor mode, no parity bit can be added. Instead, the A/D bit is always added.

(2) Reception

Data is always received while '1' is written to the RXE bit (bit 9) of the SCR register.

When the start bit appears in the reception line, one frame of data is received according to the data format determined by the SCR register. Once a frame has been received, an error flag is set if an error has occurred, and the RDRF flag (bit 12 of the SSR register) is set. At that time, if '1' is written to the RIE bit (bit 9) of the same SSR register, a reception interrupt is issued to the CPU. Check the flags of the SSR register. Read the SIDR register if the reception has been normal. If an error has occurred, take appropriate measures.

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

(3) Transmission

Transmission data is written into the SODR register when '1' is set in the TXE bit (bit 8) of the SSR register. Then, if '1' is written to the TXE bit (bit 8) of the SCR register, transmission is performed.

When the data set in the SODR register is loaded into the transmission shift register and transmission starts, the TDRE flag is set again and the next transmission data item can be set. At that time, if '1' is written to the TIE bit (bit 8) of the same SSR register, a transmission interrupt is issued to the CPU, requesting to set the transmission data in the SODR register.

The TDRE flag is cleared when data is written to the SODR register.

12.4.4 CLK synchronous mode

(1) Transfer data format

UART handles NRX (non return to zero) format data only. Figure 12.4.4a shows the transmission/reception clock and data.

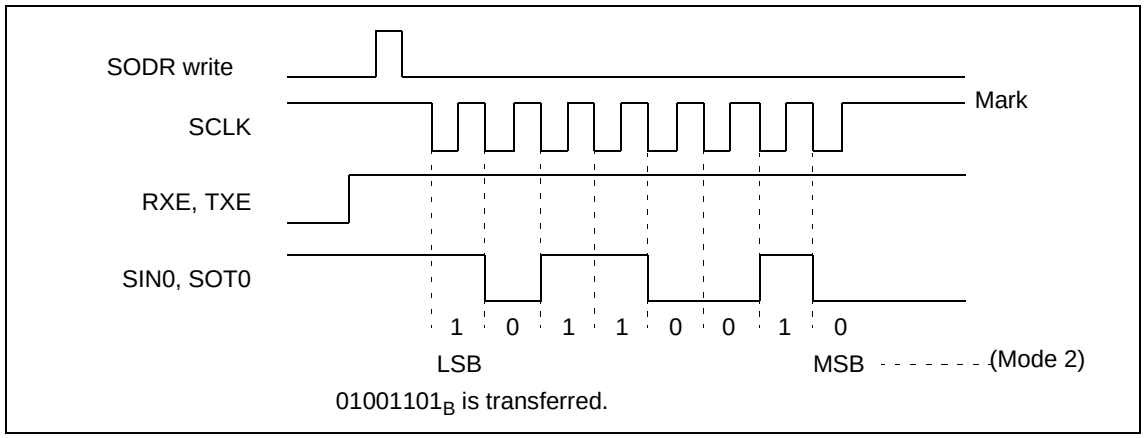


Figure 12.4.4a Transfer data format (mode 2)

When the internal clock (dedicated baud rate generator or internal timer) is selected, a data reception synchronization clock is automatically generated upon data transmission.

When an external clock is selected, it is necessary to supply precisely one byte of clock after it is confirmed that the transmission data buffer (SODR register) of the transmission UART contains data (the TDRE flag is '0'). Ensure that the signal is at the mark level before and after transmission.

Only 8-bit data can be handled, and no parity bit can be added. Since there is no start or stop bits, no errors are detected except for an overrun error.

(2) Initialization

The control register values for CLK synchronous mode are listed below.

① SMR register

MD1 and MD0 : 10

CS2, CS1, and CS0: Clock input

SCKE : Dedicated baud rate generator or internal timer: 1External clock: 0

SOE : Transmission and reception: 1 Reception only: 0

② SCR register

PEN : 0

P, SBL, A/D : Invalid

CL : 1

REC : 0 (To be initialized)

RXE and TXE : 1 written to one or both

③ SSR register

RIE : Interrupts are enabled: 1 Interrupts are disabled: 0

TIE : 0

(3) Start of communication

Communication is started by writing data in the SODR register. Virtual transmission data must be written to the SODR register even when only reception is to be performed.

(4) End of communication

The end of communication can be checked by '1' written to the RDRF flag of the SSR register. Use the ORE bit of the SSR register to check whether communication has been successful.

12.4.5 Interrupt occurrence and flag set timing

UART has five flags and two interrupt causes.

The five flags are PE, ORE, FRE, RDRF, and TDRE. PE indicates a parity error, ORE indicates an overrun error, and FRE indicates a framing error. These three flags are set when the corresponding error occurs during reception, and are cleared when '0' is written to REC of the SCR register. RDRF is set when the received data is loaded into the SIDR register, and is cleared when the SIDR register is read. The parity detection function is not available in mode 1, and the parity and framing error detection functions are not available in mode 2. TDRE is set when the SODR register becomes empty and can be written to. TDRE is cleared when the SODR register is written to.

The two interrupt causes are for reception and transmission. During reception, an interrupt is requested by PE, ORE, FRE, and RDRF. During transmission, an interrupt is requested by TDRE. The timing to set interrupt flags in each operation mode is described below.

(1) Reception in mode 0

The PE, ORE, FRE, and RDRF flags are set when reception is completed and the last stop bit is detected. Then, an interrupt request is issued to the CPU. If the PE, ORE, and FRE flags are active, the data in SIDR is invalid.

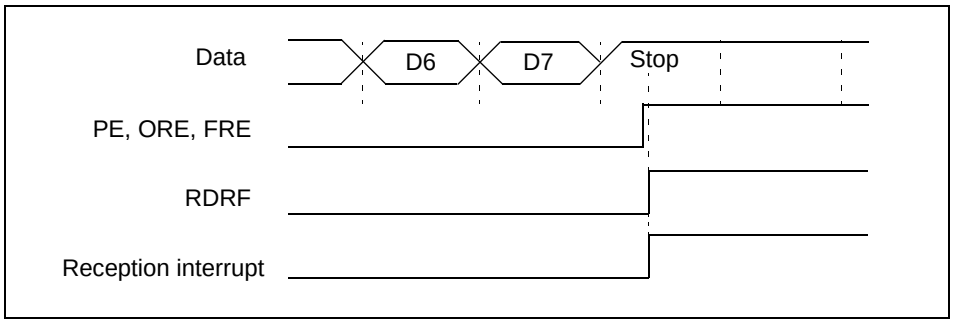


Figure 12.4.5a Timing to set PE, ORE, FRE, and RDRF (mode 0)

(2) Reception in mode 1

The ORE, FRE, and RDRF flags are set when reception is completed and the last stop bit is detected. Then, an interrupt request is issued to the CPU. Since the receivable data length is eight bits, the ninth bit indicating the address and data is invalid. If the ORE and FRE flags are active, the data in SIDR is invalid.

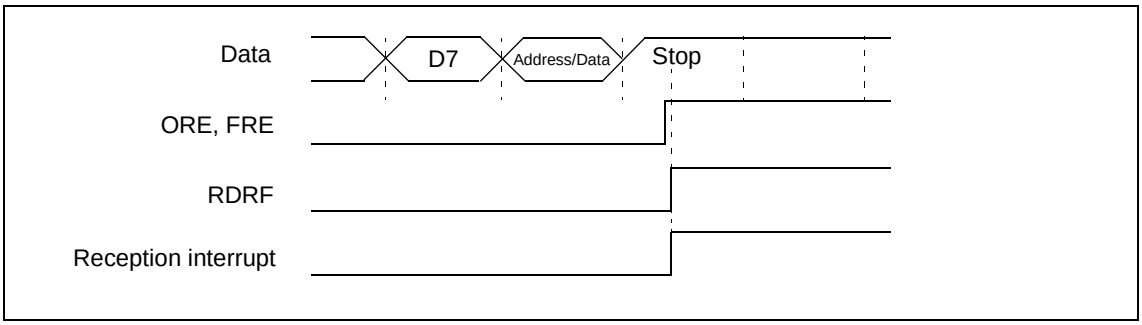


Figure 12.4.5b Timing to set ORE, FRE, and RDRF (mode 1)

(3) Reception in mode 2

The ORE and RDRF flags are set when reception is completed and the last data item (D7) is detected.

Then, an interrupt request is issued to the CPU. If the ORE flag is active, the data in SIDR is invalid.

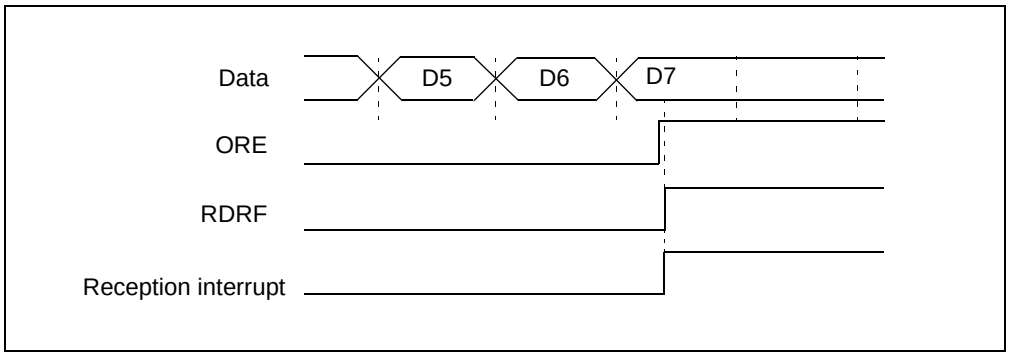


Figure 12.4.5c Timing to set ORE and RDRF (mode 2)

(4) Transmission in modes 0, 1, and 2

TDRE is cleared when a data item is written into the SODR register. When the data item is transferred to the internal shift register and the next data item can be written, TDRE is set and an interrupt request is issued to the CPU. If '0' is set in TXE (or additionally RXE in mode 2) of the SCR register during transmission, '1' is set in TDRE of the SSR register. Then, the transmission shifter stops and the transmission by UART is disabled. If '0' is set in TXE (or additionally RXE in mode 2) of the SCR register during transmission, the data set in the SODR register is transmitted before transmission stops.

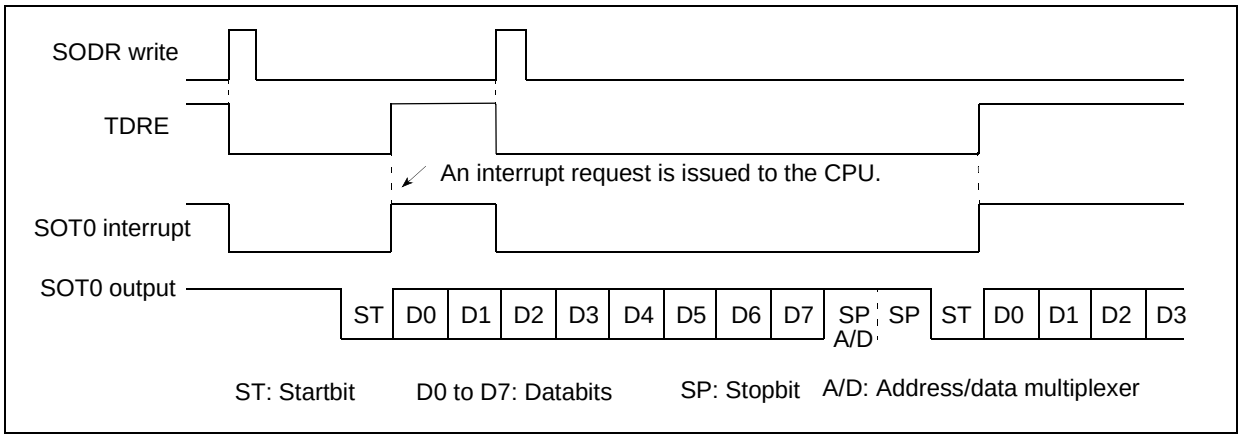


Figure 12.4.5d Timing to set TDRE (modes 0 and 1)

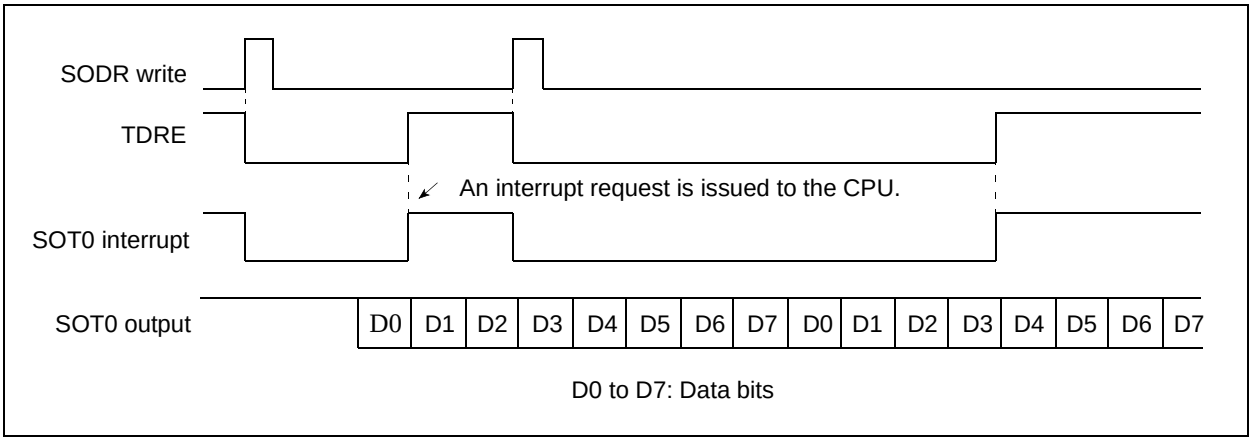


Figure 12.4.5e Timing to set TDRE (mode 2)

12.4.6 I²OS (Intelligent I/O service)

For I²OS, see the section describing I²OS.

12.4.7 Notes on use

To set a communication mode, ensure that UART is not in operation. The data sent or received during mode setting is not guaranteed.

12.4.8 Application

Mode 1 is used when two or more slave CPUs are connected to a single host CPU (see Figure 12.4.8a). This resource only supports the host side communication interface.

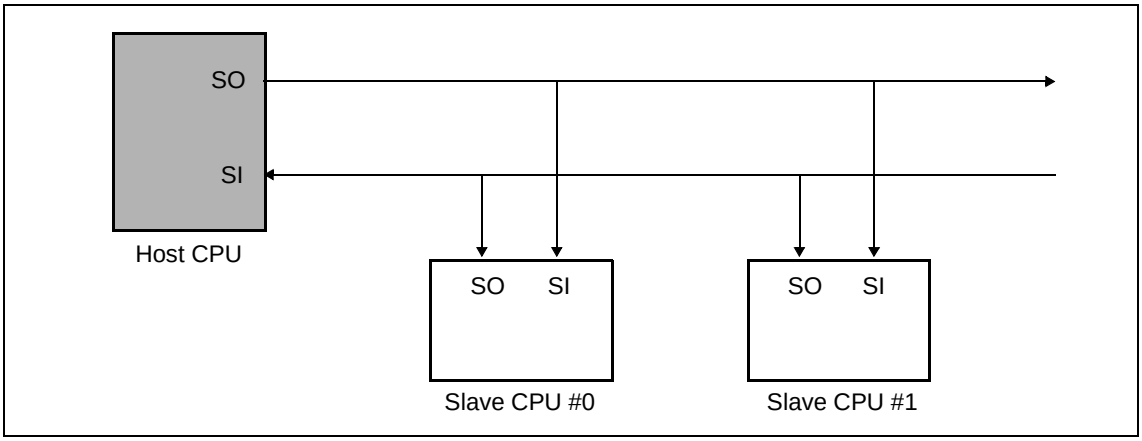


Figure 12.4.8a Sample system configuration in mode 1

Communication starts when the host CPU transfers address data. Address data means the data used when ’1’ is set in A/D of the SCR register. The address data determines the destination slave CPU and enables communication between the host and slave CPUs. Ordinary data means the data used when ’0’ is set in A/D of the SCR register. Figure 12.4.8b shows the flowchart.

In mode 1, the parity check function cannot be used. Therefore, set ’0’ in the PEN bit of the SCR register.

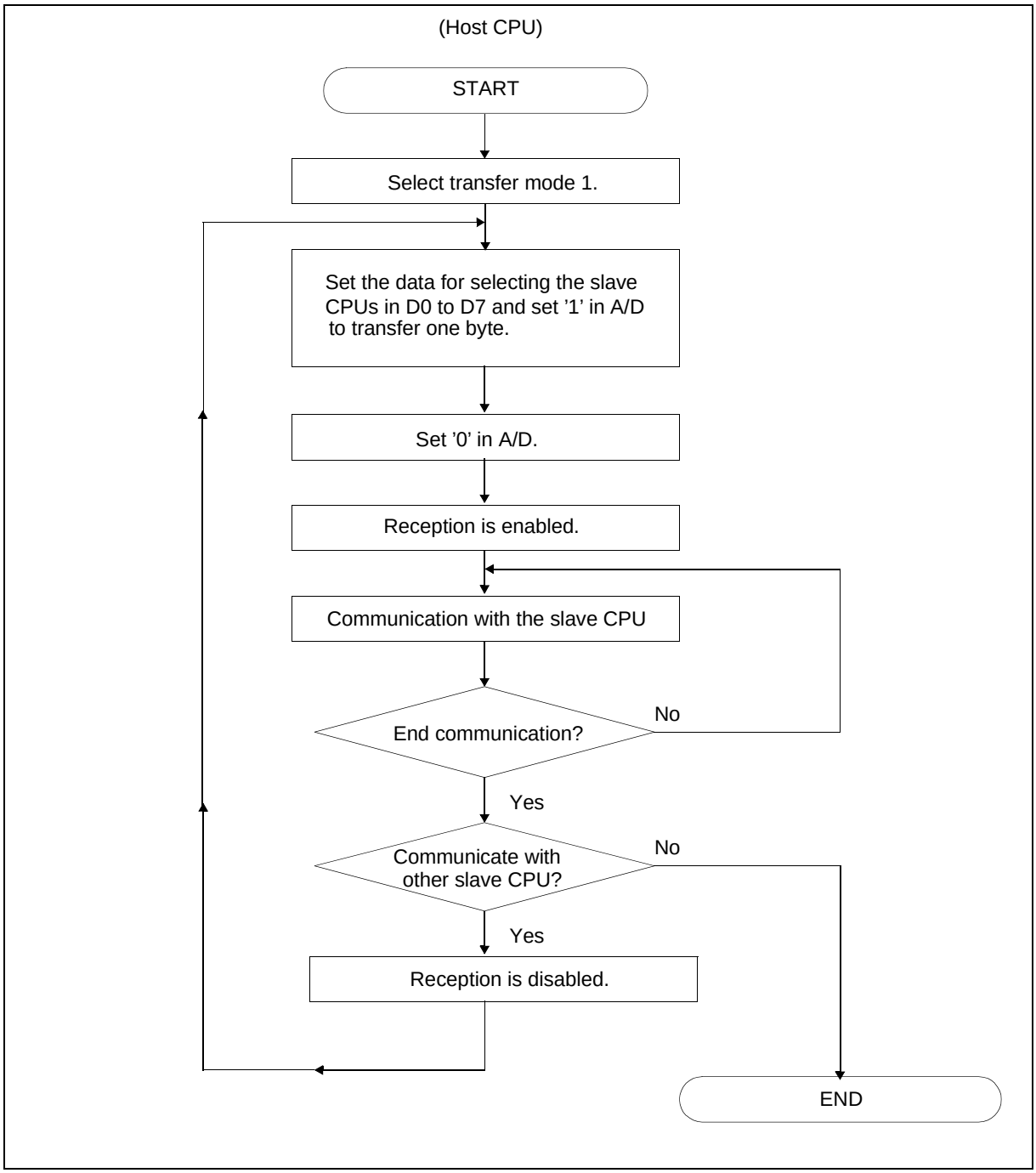


Figure 12.4.8b Flow chart of communication in mode 1

Chapter 13:
IE Bus

13.1 Outline

IEBus (Inter Equipment Bus) is a small-scale two-line serial bus interface intended to transfer data between equipment and equipment. It is designed for use in automotive and general industrial applications.

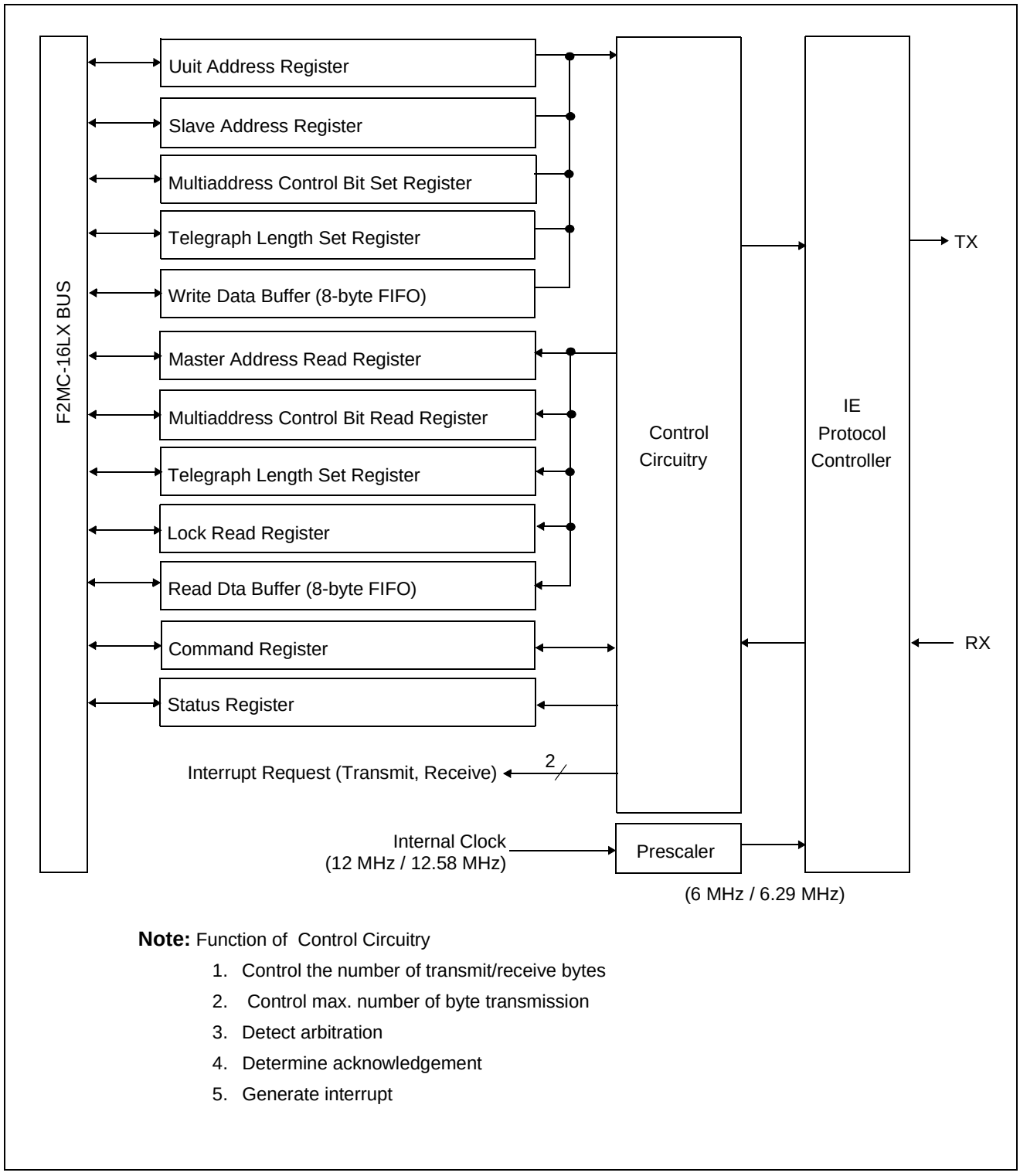
The communication protocol of the IEBus has the following features:

- Multi-master method
  All the units connected to the IEBus can transfer data to the other units.
- Multiaddress communication function (one unit can communicate to two or more units simultaneously)
  Group communication: multiaddress communication to group unit
  Broadcasting communication: multiaddress communication to all units
- Three different transfer rates can be selected:

Mode	IE Bus Internal Frequency at 6 MHz	IE Bus Internal Frequency at 6 MHz
0	Approx. 3.9 Kbps	Approx. 4.1 Kbps
2	Approx. 17 Kbps	Approx. 18 Kbps
3	Approx. 26 Kbps	Approx. 27 Kbps

- Transmit Data Buffer : 8-Byte FIFO
- Receive Data Buffer : 8-Byte FIFO
- CPU internal operation frequency: 12 MHz, 12.58 MHz

13.2 Block Diagram



Note: Function of Control Circuitry

1. Control the number of transmit/receive bytes
2. Control max. number of byte transmission
3. Detect arbitration
4. Determine acknowledgement
5. Generate interrupt

Figure 13.2a Block Diagram of IE Bus

13.3 Registers and Register Details

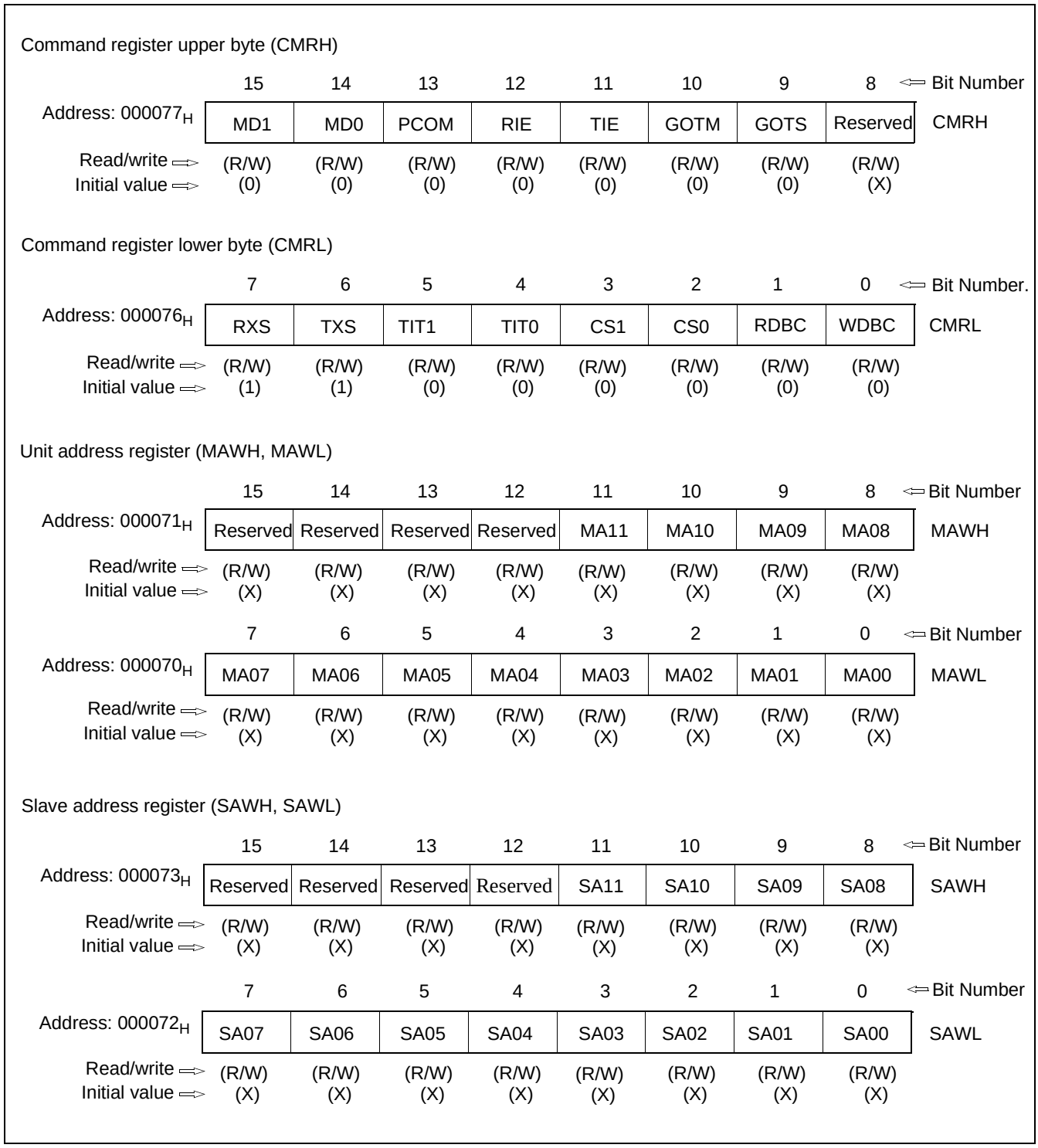


Figure 13.3a Registers of IE BUS (1/3)

Mutliaddress, control bit set register (DCWR)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000075_H	DO3	DO2	DO1	DO0	C3	C2	C1	C0	DCWR
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)	

Telegraph length set register (DEWR)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000074_H	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	DECR
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)	

Status register upper byte (STRH)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000079_H	COM	TE	PEF	ACK	RIF	TIF	TSL	EOD	STRH
Read/write ⇨	(R)	(R/W)	(R)	(R)	(R/W)	(R/W)	(R)	(R)	
Initial value ⇨	(0)	(0)	(X)	(X)	(0)	(0)	(0)	(0)	

Status register lower byte (STRL)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000078_H	WDBF	RDBF	WDBE	RDBE	ST3	ST2	ST1	ST0	STRL
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(0)	(0)	(1)	(1)	(X)	(X)	(X)	(X)	

Lock read register (LRRH, LRRL)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: $00007B_H$	Reserved	Reserved	Reserved	LOC	LD11	LD10	LD09	LD08	LRRH
Read/write ⇨	(R)	(R)	(R/W)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(1)	(1)	(1)	(0)	(X)	(X)	(X)	(X)	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: $00007A_H$	LD07	LD06	LD05	LD04	LD03	LD02	LD01	LD00	LRRL
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Figure 13.3b Registers of IE BUS (2/3)

Master address read register (MARH, MARL)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 00007D$_H$	Reserved	Reserved	Reserved	Reserved	MA11	MA10	MA09	MA08	MARH
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(1)	(1)	(1)	(1)	(X)	(X)	(X)	(X)	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 00007C$_H$	MA07	MA06	MA05	MA04	MA03	MA02	MA01	MA00	MARL
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Multiaddress, control bit read register (DCRR)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 00007F$_H$	DO3	DO2	DO1	DO0	C3	C2	C1	C0	DCRR
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(0)	(0)	(0)	(X)	(X)	(X)	(X)	(X)	

Telegraph length read register (DERR)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 00007E$_H$	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	DERR
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Read data buffer (RDB)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000081$_H$	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	RDB
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Write data buffer (WDB)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000080$_H$	WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0	WDB
Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Figure 13.3c Registers of IE BUS (3/3)

13.3.1 Command register upper byte (CMRH)

Command register upper byte (CMRH)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000077_H	MD1	MD0	PCOM	RIE	TIE	GOTM	GOTS	Reserved	CMRH
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)	(X)	

[bits 15 and 14] MD1, MD0 (Mode select):

These bits are used to select the IEBus operation mode.

Table 13.3.1a Transmission mode

MD1	MD0	Operation mode
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Setting inhibited

[bit 13] PCOM (Communication enable):

This bit is used to enable IEBus communication. When PCOM is written ‘1’, the COM flag in status register (STRH) is set and then the communication is enabled. When PCOM is written ‘0’, the communication is ended. Please set this bit to ‘1’ if the COM flag in the status register is ‘0’.

[bit 12] RIE (Receive interrupt enable):

This bit controls receive interrupt as described below.

0	Receive interrupt disabled
1	Receive interrupt enabled

The receive interrupt is occurred under the following condition:

- The eight byte Receive data buffer (RDB) is full.
- Data reception is finished normally.
- The communication has ended without receiving the number of data specified by telegraph length field in one communication frame.
- When arbitration lost, the unit cannot be selected as slave unit.

[bit 11] TIE (Transmit interrupt enable):

This bit controls transmit interrupt as described below.

0	Transmit interrupt disabled
1	Transmit interrupt enabled

The transmit interrupt is occurred under the following condition:

- In master transmit, after master address field has been transmitted, the master unit has won in arbitration.
- In slave transmit, the control bits requesting for data transmit are received from master unit.
- Writing the number of data bytes (controlled by TIT1, TIT0 bits of command register, CMRL) into write data buffer (WDB) is requested.
- Transmission of the number of data specified by telegraph length field has been completed in one communication frame.
- The communication has ended without transmitting the number of data specified by telegraph length field in one communication frame.

[bit 10] GOTM (Master transmit):

This bit indicates the start of transmission. After the communication inhibit state has been released, when GOTM is set to ‘1’, data transmission begins. This bit is written ‘1’ only and alwaays read “0”. Writing ‘0’ to this bit has no meaning.

[bit 9] GOTS (Slave transmit):

This bit indicates the start of transmission. After the communication inhibit state has been released, when GOTS is set to ‘1’, data transmission begins. This bit is written ‘1’ only and always read”0”. Writing ‘0’ to this bit has no meaning.

Table 13.3.1b Setting for GOTM and GOTS

GOTM	GOTS	Abritration	Slave Transmit	Operation
0	0	none	not allowed	slave receive
0	1	none	allowed	slave transmit
1	0	present	not allowed	after abritration lost, it can change to slave receive
1	1	present	allowed	after abritration lost, it can change to slave transmit

[bit 8] Reserved bits

Always write ‘1’ to this bit and the read value is undefined.

13.3.2 Command register lower byte (CMRL)

Command register lower byte (CMRL)

	7	6	5	4	3	2	1	0	⇐ Bit Number.
Address: 000076_H	RXS	TXS	TIT1	TIT0	CS1	CS0	RDBC	WDBC	CMRL
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(1)	(0)	(0)	(0)	(0)	(0)	(0)	

[bit 7] RXS

RX input pin polarity selected for external driver/receiver.

Table 13.3.2a Interval for the occurrence of data transmit interrupt

RXS	RX input status
0	RX pin as postive logic input. Logic ‘1’ is high level and Logic ‘0’ is Low level.
1	RX pin as negative logic input. Logic ‘1’ is low level and Logic ‘0’ is high level.

[bit 6] TXS

TX output pin polarity selected for external driver/receiver.

Table 13.3.2b Interval for the occurrence of data transmit interrupt

TXS	TX output
0	TX pin as postive logic output. Logic ‘1’ is high level and Logic ‘0’ is Low level.
1	TX pin as negative logic output. Logic ‘1’ is low level and Logic ‘0’ is high level.

Note1: For MB90580 series, during reset, TX pin will output ‘L’. If the driver/receiver used is in positive logic (Driver/receiver enable at ‘L’), TX outputs ‘L’ from reset to bit setting that will generate a communication error when there is a communication between other communication units. When it happens, it needs a outside circuit to input ‘H’ to the driver/receiver from reset to bit setting.

[bit 5, 4] TIT1, TIT0 (Data transmit interrupt control bits)

These bits control the time interval of the occurrence of interrupt for writing transmit data in write data buffer (WDB).

Table 13.3.2c Interval for the occurrence of data transmit interrupt

TIT1	TIT0	Timing for interrupt occurs
0	0	More than one byte data can be written in WDB
0	1	More than two byte data can be written in WDB
1	0	More than four byte data can be written in WDB
1	1	Eight byte data can be written in WDB

[bit 3, 2] CS1, CS0 (Cycle select):

These bits control both the CPU internal clock cycle and IEBUS controller clock cycle and CS1 and CS0 must be set to ‘0’. .

Table 13.3.2d Internal clock frequency

CS1	CS0	CPU internal clock ϕ	IEBus internal clock	Equation
0	0	12MHz (12.58 MHz)	6MHz(6.29MHz)	$\phi_{IE} = \phi/2$
0	1	Setting inhibited	----	----
1	0	Setting inhibited	----	----
1	1	Setting inhibited	----	----

Note1: The CPU and IEBus internal clock frequency are calculated by the above equation provided that the CPU operating frequency is inside the guaranteed range.

Note2: The accuracy of clock cycle calculation for mode 0 and 1 is ±1.5%, and for mode 2 is ±0.5%.

[bit 1] RDBC (Read data buffer clear):

This bit is used to clear the 8-byte read data buffer, RDB. When this bit is set to ‘1’, all the eight bytes in RDB are cleared. (RDBE = 1)

This bit is always read as ‘0’.

[bit 0] WDBC (Write data buffer clear):

This bit is used to clear the 8-byte write data buffer, WDB. When this bit is set to ‘1’, all the eight bytes in WDB are cleared and WDB becomes empty (WDBE = 1)

This bit is alwarys read as ‘0’.

For the communication with no. of byte transfer greater than the maximum transfer byte per frame, the data written in the previous frame will no longer valid if ‘1’ is written to this bit. The data written in the current frame will be transmitted first. In converse, if ‘0’ is written to this bit, the unsent data in the previous frame will be prioritized to be sent first.

If the transmission is terminated due to timing error, even thought ‘0’ is written to this bit, the transmission will be started from the next data and not the last one which has not been sent completely.

13.3.3 Unit address register (MAWH, MAWL)

Unit address register (MAWH, MAWL)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000071_H	Reserved	Reserved	Reserved	Reserved	MA11	MA10	MA09	MA08	MAWH
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)	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000070_H	MA07	MA06	MA05	MA04	MA03	MA02	MA01	MA00	MAWL
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)	

These two registers MAWH, MAWL are used to set its own unit address (12 bits). When the unit is configured as master, the unit address set in MAWH, MAWL is transmitted as master address. When it is configured as slave mode, this unit address is used to compare with the received slave address.

Bit 15 to 12 are reserved bits and always write ‘1’ to them. The read values are undefined.

Note: Make sure to set the unit address before the communication inhibit state is released.

13.3.4 Slave address register (SAWH, SAWL)

Slave address register (SAWH, SAWL)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000073_H	Reserved	Reserved	Reserved	Reserved	SA11	SA10	SA09	SA08	SAWH
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)	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000072_H	SA07	SA06	SA05	SA04	SA03	SA02	SA01	SA00	SAWL
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)	

These two registers SAWH, SAWL are used to set the slave address (12 bits) for master transmit.

Bit 15 to 12 are reserved bits and always write ‘1’ to them. The read values are undefined.

Note: Make sure to set the slave address before the communication inhibit state is released.

13.3.5 Mutliaddress, control bit set register (DCWR)

Mutliaddress, control bit set register (DCWR)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000075_H	DO3	DO2	DO1	DO0	C3	C2	C1	C0	DCWR
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)	

[bit 15, 14, 13, 12] DO3, DO2, DO1, DO0 (Multiaddress/normal communication select bits):

These bits are used to select multiaddress (more than one slave) or normal communication (one slave).

For multiaddress communication, DO3-0 is set to '0000' and then the multiaddress bit in communication frame is sent out as '0'.

For normal communication, DO3-0 is set to '1000' and then the multiaddress bit in communication frame is sent out as '1'.

Always write '0' to bit 14, 13, and 12, and when reading, these bits always return '0'.

[bit 11, 10, 9, 8] C3, C2, C1, C0 (Control bits):

These bits are used to control IEBus communication.

Table 13.3.5a Control bits setting

	C3[Note 1]	C2	C1	C0	Control Operation
0H	0	0	0	0	Slave status read
1H	0	0	0	1	Undefined
2H	0	0	1	0	Undefined
3H	0	0	1	1	Data read and lock
4H	0	1	0	0	Lock address read (Lower 8 bits)
5H	0	1	0	1	Lock address read (Upper 4 bits)
6H	0	1	1	0	Slave status read and unlock
7H	0	1	1	1	Data read
8H	1	0	0	0	Undefined
9H	1	0	0	1	Undefined
AH	1	0	1	0	Command write and lock
BH	1	0	1	1	Data write and lock
CH	1	1	0	0	Undefined
DH	1	1	0	1	Undefined
EH	1	1	1	0	Command write
FH	1	1	1	1	Data write

Note1: The transfer direction of telegraph length bits in telegraph length field and data bits in data field are controlled by the value of C3 as follows:
When C3 is '1': Transfer from master unit to slave unit
When C3 is '0': Transfer from slave unit to master unit

Note2: 3H, 6H, AH, BH are the lock and unlock function selection bits.

Note3: When sending the undefined control bits like 1H, 2H, 8H, 9H, CH, DH, no acknowledge will be returned.

13.3.6 Telegraph length set register (DEWR)

Telegraph length set register (DEWR)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000074_H	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	DECR
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 is used to set the number of data bytes to be transmitted and is valid only for data transmission.

Table 13.3.6a Number of transmit data bytes setting

DE7-0	Number of transmit data bytes
01H	1 byte
02H	2 bytes
..	..
..	..
FFH	255 bytes
00H	256 bytes

During slave transmit, make sure to set the transmit data count byte to ‘1’ when the control frame are 0H (Reading Slave Status), 4H (Reading Lock-address of the lower 8-bit), 5H (Reading Lock-address of the upper 8-bit), 6H (Reading slave -status and diabling lock).

If the no. of bytes required to transfer is greater than the maximum no of transfer byte per frame, it will result in multiframe communciation. In that case, DEWR should be set using the following formula.

DEWR = DERR - Maximum no. of transfer bytes per frame

Upon the completion of one frame, the remaining number of byte required to transfer can be obtained by deducting the maximum no. of transfer byte per frame from DERR. This value is used to set DEWR for the next frame.

13.3.7 Status register upper byte (STRH)

Status register upper byte (STRH)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000079_H	COM	TE	PEF	ACK	RIF	TIF	TSL	EOD	STRH
Read/write ⇨	(R)	(R/W)	(R)	(R)	(R/W)	(R/W)	(R)	(R)	
Initial value ⇨	(0)	(0)	(X)	(X)	(0)	(0)	(0)	(0)	

[bit 15] COM (Communication status):

This bit indicates the communication status as described below.

0	Communication is prohibited
1	Communication is enabled

When this bit is ‘0’ and the PCOM bit in command register (CMRH) is written ‘1’, this bit is set.

When communication ends, this bit will be cleared.

.

[bit 14] TE (Timing error):

This bit is set when timing error has occurred during communication. Writing ‘0’ will clear this bit.

This bit is written ‘0’ only, there is no meaning for writing ‘1’.

[bit 13] PEF (Parity error):

This bit is set when parity error has been detected.

0	No parity error
1	Parity error

In receive side, if this bit is set, the acknowledge bit will not be returned. This bit will be cleared after the communication inhibit state is released.

[bit 12] ACK (Acknowledge bit):

This bit indicates

0	The acknowledge bit is ‘0’
1	The acknowledge bit is ‘1’

During normal communication, acknowlege bit will be returned after each data received correctly. This bit will be cleared after communication inhibit state is released.

This bit has no meaning in multiaddress communication and the read value is indefined.

[bit 11] RIF (Receive interrupt flag):

This bit is set when receive interrupt is occurred.

0	No receive interrupt request
1	Have receive interrupt request

This bit is cleared by writing ‘0’ to this bit or after the extended intelligent I/O service has been served. This bit is written ‘0’ only.

[bit 10] TIF (Transmit interrupt flag):

This bit is set when transmit interrupt is occurred.

0	No transmit interrupt request
1	Have transmit interrupt request

This bit is cleared by writing ‘0’ to this bit or after the extended intelligent I/O service has been served. This bit is written ‘0’ only.

[bit 9] TSL (Transmit limit):

This bit is set when the maximum number of data bytes that can be transmitted in one communication frame has been reached. And this bit is cleared when next communication frame starts.

[bit 8] EOD (End of data):

This bit is set when the number of data bytes specified by telegraph length field has been transferred completely. It means communication ends normally. This bit is cleared when next communication frame starts.

13.3.8 Status register lower byte (STRL)

Status register lower byte (STRL)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000078_H	WDBF	RDBF	WDBE	RDBE	ST3	ST2	ST1	ST0	STRL
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(0)	(0)	(1)	(1)	(X)	(X)	(X)	(X)	

[bit 7] WDBF (Write data buffer full):

This flag indicates the status of the write data buffer (WDB).

0	Write data buffer is not full
1	Write data buffer is full

This bit is set when WDB is full and cleared when at least one byte of data can be written into WDB.

[bit 6] RDBF (Read data buffer full):

This flag indicates the status of the read data buffer (RDB).

0	Read data buffer is not full
1	Read data buffer is full

This bit is set when RDB is full and cleared when at least one byte of data can be received and stored in RDB.

[bit 5] WDBE (Write data buffer empty):

This flag indicates the status of the write data buffer (WDB).

0	Write data buffer is not empty
1	Write data buffer is empty

This bit is set when WDB is empty and cleared when data is written into WDB. Writing ‘1’ to WDBC in command register CMRL will set this bit.

[bit 4] RDBE (Read data buffer empty):

This flag indicates the status of the read data buffer (RDB).

0	Read data buffer is not empty
1	Read data buffer is empty

This bit is set when RDB is empty and cleared when data is received and stored in RDB. Writing ‘1’ to RDBC in command register CMRL will set this bit.

[bit 3-0] ST3, ST2, ST1, ST0 (Operation status bits)

These bits indicates the communication status of the unit and generates the corresponding interrupt during transmission or reception. By reading these bits, the communication status of the unit can be known.

Table 13.3.8a Status flag

ST3	ST2	ST1	ST0	Status	State
0	0	0	0	Master/slave transmit	Transmit starts
0	0	0	1		During transmission
0	0	1	0		Transmit ends normally
0	0	1	1		Ends without all data being transmitted
0	1	0	0	Master receive	Master receive starts
0	1	0	1		Master receive data full
0	1	1	0		Master receive ends normally
0	1	1	1		Ends without all data being received
1	0	0	0	Slave receive	Slave receive starts
1	0	0	1		Slave receive data buffer full
1	0	1	0		Slave receive ends normally
1	0	1	1		Ends without all data being received
1	1	0	0	Multiaddress receive	Multiaddress receive starts
1	1	0	1		Multiaddress receive data buffer full
1	1	1	0		Multiaddress receive ends normally
1	1	1	1		Ends without all data being received

For more detail description on setting these four bits, please refer to Table 13.5.2a.

13.3.9 Lock read register (LRRH, LRRL)

Lock read register (LRRH, LRRL)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: $00007B_H$	Reserved	Reserved	Reserved	LOC	LD11	LD10	LD09	LD08	LRRH
Read/write ⇨	(R)	(R)	(R/W)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(1)	(1)	(1)	(0)	(X)	(X)	(X)	(X)	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: $00007A_H$	LD07	LD06	LD05	LD04	LD03	LD02	LD01	LD00	LRRL
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

[bit 15-13] Reserved bits:

Always reading ‘1’ from these bits.

[bit 12] LOC (Lock check):

This bit indicates the status whether the unit is locked or not from other unit.

0	Does not lock
1	Locked

Writing ‘0’ to this bit will unlock the unit itself. Writing ‘1’ to this bit has no meaning.

[bit 11-0] LD11 - LD00 (Lock address):

These bits store the lock address, the address of the master that has executed locking to the unit. When the unit is not locked. there is no meaning for these bits.

Note: In IEBus communication, the lock function is used to transmit a message over two or more communication frames. If the master that has executed locking was down before executing the unlocked command, the locked unit cannot receive data anymore. So in order to prevent such condition, the unit in the system that supporting lock function need checking the lock status periodically by reading this lock read register. And the unit can unlock itself by writing ‘0’ to LOC bit..

13.3.10 Master address read register (MARH, MARL)

Master address read register (MARH, MARL)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 00007D$_H$	Reserved	Reserved	Reserved	Reserved	MA11	MA10	MA09	MA08	MARH
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(1)	(1)	(1)	(1)	(X)	(X)	(X)	(X)	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 00007C$_H$	MA07	MA06	MA05	MA04	MA03	MA02	MA01	MA00	MARL
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

[bit 15 - 12] Reserved bits:

Always reading ‘1’ from these bits.

[bit 11 - 0] MA11 - MA00 (Master address):

In slave mode, these bits store the address of the master that has won the arbitration in master address field.

If the unit itself is the master, then the unit address stored in unit address register (MAWH, MAWL) will be read out.

13.3.11 Multiaddress, control bit read register (DCRR)

Multiaddress, control bit read register (DCRR)

	15	14	13	12	11	10	9	8	⇦Bit Number
Address: 00007F$_H$	DO3	DO2	DO1	DO0	C3	C2	C1	C0	DCRR
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(0)	(0)	(0)	(X)	(X)	(X)	(X)	(X)	

[bit 15-12] DO3, DO2, DO1, DO0 (Multiaddress/normal communication bits):

In slave mode, the received multiaddress bit from the master is stored in bit DO0.

If the unit itself is the master, the multiaddress/normal communication set bits (DO3-0) in multiaddress, control bit set register (DCWR) is read out.

Normal communction: (0001B)

Multiaddress communication: (0000B)

DO3~0 always read as “0”.

[bit 11-8] C3, C2, C1, C0 (Control bits)

In slave mode, the received control bits from the master are stored in these bits.

If the unit itself is the master, the control bits (C3-0) in multiaddress, control bit set register (DCWR) is read out. These bits are set after the control field has been received and acknowledge bit was detected.

For more detail description, please refer to Table 13.3.5a.

13.3.12 Telegraph length read register (DERR)

Telegraph length read register (DERR)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: $00007E_H$	DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0	DERR
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

if the unit itself is the receiver, this register stores the number of data specified by telegraph length field.

If the unit itself is the transmitter, the telegraph length bits in telegraph length set register (DEWR) are read. This register is set after the following.

1. When used as master unit
   (a) In transmit mode, the number of transmit data bytes is written into DEWR
   (b) In receive mode, the telegraph length field is received
   (c) Communication ends
2. When used as slave unit
   (a) In transmit mode, the number of transmit data bytes is written into DEWR
   (b) In receive mode, the telegraph length field is received
   (c) Communication ends

13.3.13 Read data buffer (RDB)

Read data buffer (RDB)

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 000081_H	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	RDB
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

This register (internally is a 8-byte FIFO buffer) stores received data in data field of the communication frame. When eight byte data have been received, RDB becomes full and receive interrupt is generated. Then data in RDB should be read out before the next coming byte of data is received as shown in Table 13.3.13a . Otherwise, error will be occurred.

When error occurs in multiaddress reception, the communication ends. But when error occurs in normal reception, the acknowledge bit will not returned to the transmitter. Then the transmitter will resend data again until the maximum number of data transmitted is reached.

Even though RDB is not full, the receive interrupt will be generated when the number of data specified in the telegraph field have been received, or the maximum number of data received in one communication frame is reached. Once the receive interrupt has occurred, the data in RDB should be read out.

Writing ‘1’ to WDBC in CMRL will clear all data in the buffer and return it as empty state.

This register can only be read when noit empty

Table 13.3.13a Time Required for next data receive after receive buffer full interrupt occurred

	Maximum Time (us)	No. of Cycles
Mode 0	1580	19000
Mode 1	400	4800
Mode 2	290	3400

13.3.14 Write data buffer (WDB)

Write data buffer (WDB)

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000080_H	WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0	WDB
Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

This register (internally is a 8-byte FIFO buffer) stored data to be transmitted in data field of the communication frame. The data write interrupt timing is set by the two bits TIT1, TIT0 in command register (CMRL).

When the write interrupt occurs, next data is requested to write into WDB. When all data has been transmitted (WDB is empty), and the data cannot be writtenin RDB within the time listed in Table 13.3.14a, it will result in an error and the transmission will be terminated. Writing '1' to WDBC bit in command register CMRL will clear the buffer and return it as empty state.

This register can only be written when not full.

Table 13.3.14a Data write time after WDB empty interrupt

	Maximum Time	No. of cycles
Mode 0	1580	19000
Mode 1	400	4800
Mode 2	290	3400

13.4 IEBus Communication Protocol

13.4.1 Overview

IEBus (Inter Equipment Bus) is a small-scale two-line serial bus interface intended to transfer data between equipment and equipment.

- Communication method
  Data are transferred by means of half duplex asynchronous communication.
- Multi-master method
  All the units connected to the IEBus can transfer data to the other units.
- Multiaddress communication function (one unit can communicate to two or more units simultaneously)
  Group communication: multiaddress communication to group units
  Broadcasting communication: multiaddress communication to all units
- Three transfer rates can be selected:

Table 13.4.1a IEBus transfer rates

Mode	IEBus internal clock = 6MHz	IEBus internal clock = 6.29MHz	Maximum number of transfer bytes (bytes / frame)
0	approx. 3.9Kbps	approx. 4.1 Kbps	16
1	approx. 17Kbps	approx. 18 Kbps	32
2	approx. 26Kbps	approx. 27Kbps	128

- Access control: CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
- The priority to occupy the IEBus:
  (1) Multiaddress communication takes precedence over normal communication (communication between one unit and another).
  (2) The lower master address takes precedence over the higher address.
- Communication scale [Note]

  Number of units: 50 max.

  Cable length: 150m max. (when twisted pair cable with resistance less than 0.1 W/m is used)

  loading Capacitance: 8000pF max. (between BUS- and BUS+) when IEBus internal clock is 6 MHz
  7100pF max. (between BUS- and BUS+) when IEBus internal clock is 6.29MHz

  Terminal resistance: 120 W.

  Note: The system sacle depends on the IE Bus driver/receiver that is used.

13.4.2 Determining bus mastership (arbitration)

The equipment connected to the IEBus performs an operation to occupy the bus when it controls another equipment. This operation is called arbitration.

Arbitration is to grant the bus mastership to one of several units that have simultaneously started transmission. As only one equipment acquires the bus mastership as a result of arbitration, the following priority is used to determine which equipment acquires the bus mastership:

- Priority on type of communication
  Multiaddress communication takes precedence over normal communication.
- Priority on master address
  If the type of communication is the same, the lower master address takes precedence over the higher one.
  Example: The master address consists of 12 bits, the unit at 000H has the highest priority, and the unit at FFFH has the lowest priority.

13.4.3 Communication mode

The IEBus provides three communication modes, in which each has a different transfer rate. The transfer rate in each communication mode and the maximum number of transfer bytes in one communication frame are shown as below:

Table 13.4.3a Transfer rate and maximum number of transfer byte in each communication mode

Mode	Maximum number of transfer bytes (bytes / frame)	Effective transfer rate (bps) Note1	
		IEBus internal clock = 6MHz Note2	IEBus internal clock = 6.29 MHz Note2
0	16	approx. 3.9Kbps	approx. 4.1 Kbps
1	32	approx. 17Kbps	approx. 18 Kbps
2	128	approx. 26Kbps	approx. 27Kbps

Note1: Effective transfer rate is measured when the maximum number of data bytes have been transferred.

Note2: The relationship between IEBus internal clock and CPU clock is referred to Table 13.3.2d.

Caution1: The required communication mode should be selected for each equipment connected to the IEBus before communication is started. Also, the communication is not performed correctly unless the communication mode of the master and the slave unit are the same.

Caution2: Be sure that both the communication mode and IEBus internal clock are the same for all units connected to the IEBus. Even though the same communication mode is selected, the communication is still performed incorrectly if the IEBus internal clock frequency is different.

13.4.4 Communication address

In IEBus, each equipment is assigned to a specific 12-bit communication address. The communication address is consisted of:

Higher 4 bits:group number (identify which group the equipment belongs to)

Lower 8 bits:unit number (identify each equipment in one group)

13.4.5 Multiaddress communication

In normal communication mode, the communication is performed on a one-to-one basis, i.e. only one master and one slave. In contrast, multiaddress communication allows the master transmitting data to more than one slave. As more than one slave exist in the IEBus, none of them returns an acknowledge signal in the communication.

The multiaddress bit is used to select either multiaddress communication or normal communication. For detail description, refer to (6) transfer protocol.

Multiaddress communication has the following two modes:

- Group multiaddress communication
  Communicating with equipments having the same group number as specifying in higher 4 bits of the communication address.
- Broadcasting communication
  Communicating with all equipments, regardless of the value of the group number

The slave address specified in slave address field is used to identify either group multiaddress or broadcasting communication. For detail description, refer to section Section 13.4.6, “Transfer protocol".

13.4.6 Transfer protocol

The signal transmit format of the IEBus is shown as below

Field Name		Header		Master address field		Slave address field			Control field			Telegraph length field			Data field						
No. of bits		1	1	12	1	12	1	1	4	1	1	8	1	1	8	1	1		8	1	1
Frame format		Start bit	Multi-address bit	Master address bits	P	Slave address bits	P	A	Control bits	P	A	Telegraph length bits	P	A	Data bits	P	A	- - - -	Data bits	P	A
Transmit time	Mode 0	approx. 7330 µs													approx. 1590 x N µs						
	Mode 1	approx. 2090 µs													approx. 410 x N µs						
	Mode 2	approx. 1590 µs													approx. 300 x N µs						

Note1: P: parity bit
A: acknowledge bit where A=0: ACK and A=1: NAK

Note2: The acknowledge bit is ignored in multiaddress communication.

(1) Header

The header field is consisted of start bit and multiaddress bit.

- Start bit
  The start bit is a signal used to inform the other units that data transmission starts. The unit initiating the data transmission outputs a low-level signal (start bit) for a specific time and then outputs the multiaddress bit.
  When one unit tries to output the start bit, but it found that another unit has already output a start bit, then the unit does not output the start bit. But it waits for the end of the start bit output by the another unit and outputs the multiaddress bit in synchronization with the output end timing of the start bit.
  The units other than the one that has started transmission detect this start bit and enters the receive status.
- Multiaddress bit
  This bit indicates whether the master selects multiaddress or normal communication. When the multiaddress bit is ‘0’ for multiaddress communication and is ‘1’ for normal communication. Moreover, multiaddress communication is divided into two modes, group multiaddress and broadcasting communication. These two modes are identified by the value of the slave address.
  In multiaddress communication, since there are two or more slave units, the acknowledge bit for each field following the master field is not returned.
  If at the same time, two or more units start transmission of a communication frame, multiaddress communication takes precedence over normal communication and is the winner in arbitration.

(2) Master address field

This field is outputted by the master to identify its address for other units being communicated and is consisted of 12 bit of master address with MSB transmitting first and 1 parity bit.
If at the same time, two or more units starts transmitting the multiaddress bit of the same value, judgement of arbitration is based on the master address field value. Everytime when the unit transmits one bit of its unit address, it compares the data output with the data on the IEbus. If they are found to be different, the unit lost the arbitration and then stops transmission and enters receive status.
Since the IEBus is configured as wired AND, the unit having the least master address among the units participating in arbitration (arbitration masters) wins arbitration. After 12-bit master address is transmitted out, only one unit remains in transmit status as the master. This master then outputs the parity bit and let other units confirming that the transmit master address data contains no error. After that, the slave address field is output.

Note: Even parity is used for parity check. If the total number of ‘1’ in master address bits is odd, the parity bit will be set as ‘1’.

(3) Slave address field

This field outputs the address of the other unit with which the master is to communicate and is consisted of 12 bits of slave address with MSB transmitting first, parity bit and acknowledge bit.

After a 12-bit slave address has been transmitted, a parity bit is output to ensure that the slave address is not received by mistake. Then the master unit detects an acknowledge signal from slave unit to confirm its existence on the IEBus. After the detection of acknowledge signal, the master unit starts outputting the control field. However in multiaddress communication mode, the master starts outputting the control field without detecting the acknowledge bit.

In the slave side, when it detects that the slave address has coincided with its own unit address, and the parity bit in both master address field and slave address field are even, it outputs an acknowledge signal. However, if parity bit is odd, the slave unit judges that either master address or slave address has been received incorrectly, and then acknowledge signal is not returned. At this moment, the master unit enters the standby (monitor) status, and communication ends.

In multiaddress communication mode, the slave address is used to identify whether it is group multiaddress or broadcasting communication as follows:
When slave address is FFFH: Broadcasting communication
When slave address is not FFFH: Group multiaddress communication

Note: The group number for group multiaddress communication is identified by the higher 4 bits of the slave address.

(4) Control field

This field is used to control the type of following data field and direction of data transfer between master and slave. This field is consisted of 4 bits of control bit with MSB transmitting first, parity bit and acknowledge bit.

If even parity is checked and the slave can execute the function requested by the master, the slave returns an acknowledge signal and then proceeds to the telegraph length field. Even though the parity is even, but if the slave unit cannot execute the function requested by the master, or if the parity is odd, the slave unit does not output the acknowledge signal and returns to the standby (monitor) status.

After the master confirms the return of acknowledge signal, it proceeds to the telegraph length field. If the master cannot receive the acknowledge signal, it enters the standby status, and communication ends. In multiaddress communication mode, the master unit does not detect the acknowledge signal, but proceeds to the telegraph length field.

(5) Telegraph length field

This field is used to indicate the number of bytes of transmit data from the transmitter to receiver. This field is consisted of 8 bits of telegraph length with MSB transmitting first, parity bit and acknowledge bit. Table 13.4.6a shows the relationship between the telegraph length field and the number of transmit data bytes.

Table 13.4.6a Number of transmit data bytes setting

Telegraph length bits (HEX)	Number of transmit data bytes
01H	1 byte
02H	2 bytes
..	..
..	..
FFH	255 bytes
00H	256 bytes

Note: According to the communication mode being set, if the number of transmit data bytes set in telegraph length field is greater than the maximum number of transmit data bytes per frame, then communication with multi-frame is performed. In this

case, the second and following communication frames will transmit the remaining data bytes specified in the telegraph length field.

The function of telegraph length field differs when the master is in transmit mode (bit 3 of control bits is '1') or receive mode (bit 3 of control bits is '0') as follow:

- Master transmit mode
  The telegraph length bits and parity bit are output by the master unit. If even parity is detected by the slave, it returns the acknowledge signal. Then the master proceeds to the data field. But in multiaddress communication mode, the slave does not return any acknowledge signal.
  If odd parity is detected, the slave judges that the telegraph length bits have not been correctly received, and then go into standby (monitor) mode without returning acknowledge signal. At the same time, the master also goes into standby status, and communication ends.
- Master receive mode
  The slave outputs the telegraph length bits and parity bit. If even parity is detected by the master, it returns the acknowledge signal. But if odd parity is detected, the master judges that the telegraph length bits have not been correctly received, and goes into the standby status without returning acknowledge signal. Then the slave also goes into standby status, and communication ends.

(6) Data field

This field is used by the master to transmit/receive data to/from the slave. This field is consisted of eight data bits with transmitting MSB first, parity bit and acknowledge bit.

Multiaddress communication can only be performed when the master unit transmits data and the acknowledge signal is ignored. The operations for master transmits and receives data are described as follow:

- Master transmit mode
  When the master writes data to a slave, it transmits data bits and parity bit to the slave. Then if even parity is detected by the slave and its receive data buffer is not full, the slave returns the acknowledge signal. If odd parity is detected or the receive buffer is full, the slave rejects accepting the corresponding data, and does not return the acknowledge signal.
  If the acknowledge is not detected by the master, it retransmits the same data until the acknowledge bit is detected or the maximum number of transmit bytes is exceeded.
  If the parity is even and the acknowledge signal is returned from the slave, the master transmits the next available data if the maximum number of transmit bytes is not exceeded.
  In multiaddress communication mode, the slave unit does not return the acknowledge signal, and the master transmits data on a 1-byte-by-1-byte basis.
- Master receive mode
  When the master reads data from the slave, the master outputs a synchronous signal corresponding to all the read bits. Then the slave outputs the data and parity bit to the IEBus in response to the synchronous signal output by the master. After that, the master reads these bits and confirm the parity check.
  If odd parity is detected or the master's receive buffer is full, it rejects accepting the data and does not return the acknowledge signal. If the maximum number of transmit bytes per frame is not exceeded, the master repeatedly reads the same data.
  If even parity is detected and the master's receive buffer is not full, the master accepts the data and returns the acknowledge signal. If the maximum number of transmit bytes per frame is not exceeded, the master reads the next data.

(7) Parity bit

The parity bit is used to confirm that the transmit data contains no error. It is appended to master address bits, slave address bits, control bits, telegraph length bits and data bits.

The parity is an even parity. If the number of '1' bits in the data is odd, then the parity bit is set to '1'. If the number of '1' bits in the data is even, then the parity bit is set to '0'.

(8) Acknowledge bit

An acknowledge bit is appended to the following location to confirm whether data has been correctly received in the normal communication mode (communication between one unit and another):

- At the end of slave address field
- At the end of control field
- At the end of telegraph length field

- At the end of data field

The acknowledge bit is defined as:
‘0’: The transmit data is recognized (ACK)
‘1’: The transmit data is not recognized (NAK)

The acknowledge bit is ignored in multiaddress communication.

1. Acknowledge bit at the end of slave field

   The acknowledge bit at the end of the slave field is treated as NAK in any of the following cases, and then transmission is aborted:

   - If the parity of the master address bits or slave address bits is not correct
   - If a timing error (error in bit format) occurs
   - If the specific slave unit does not exist

2. Acknowledge bit at the end of the control field

   The acknowledge bit at the end of the control field is treated as NAK in any of the following cases, and then transmission is aborted:

   - If the parity of the control bits is not correct
   - If bit 3 of the control bits is ‘1’ (write operation) but the slave receive buffer[Note] is full
   - If the control bits indicate data read (3H, 7H) but the slave transmit buffer[Note] is empty
   - If the slave has been locked, but value of 3H, 6H, 7H, AH, BH, EH or FH in the control bits are requested by another unit other than the one has set locking
   - If the control bits indicate reading of a lock address (4H) but the slave has not been locked.
   - If a timing error occurs
   - If undefined control bit values are set

   Note: Refer to slave status (SSR) in (7)

3. Acknowledge bit at the end of the telegraph length field

   The acknowledge bit at the end of the telegraph length field is treated as NAK in any of the following cases, and transmission is aborted:

   - If the parity of the telegraph length bits is not correct
   - If a timing error occurs

4. Acknowledge bit at the end of data field

   The acknowledge bit at the end of the data field is treated as NAK in any of the following cases, and then transmission is aborted:

   - If the parity of the data bits is not correct [Note]
   - If a timing error occurs after the previous acknowledge bit has been transmitted
   - If the receive buffer has become full and thus no more data can be accepted

   Note: The same data is retransmitted if the maximum number of transmit data bytes per frame is not exceeded.

13.4.7 Transmit data

The content in data field is controlled by the control bits in control field and is shown below:

Table 13.4.7a Control bits setting

	Bit 3 Note 1	Bit 2	Bit 1	Bit 0	Function Note 2
0H	0	0	0	0	Slave status (SSR) read
1H	0	0	0	1	Undefined
2H	0	0	1	0	Undefined
3H	0	0	1	1	Data read and lock
4H	0	1	0	0	Lock address read (Lower 8 bits)
5H	0	1	0	1	Lock address read (Upper 4 bits)
6H	0	1	1	0	Slave status (SSR) read and unlock
7H	0	1	1	1	Data read
8H	1	0	0	0	Undefined
9H	1	0	0	1	Undefined
AH	1	0	1	0	Command write and lock
BH	1	0	1	1	Data write and lock
CH	1	1	0	0	Undefined
DH	1	1	0	1	Undefined
EH	1	1	1	0	Command write
FH	1	1	1	1	Data write

Note1: The direction in which telegraph length bits in the telegraph length field and data in the data field are transferred is changed depending on the value of Bit 3 as follows:
When Bit 3 is '1': Transfer from master unit to slave unit
When Bit 3 is '0': Transfer from slave unit to master unit

Note2: Control bits 3H, 6H, AH, BH are used to lock and unlock the unit.
When those undefined control bits 1H, 2H, 8H, 9H, CH, DH has been received, the acknowledge bit is not returned.

When the slave is locked, it can only execute the following command requesting by other units besides of the master executing the lock command:

Table 13.4.7b The control command that can be executed by a locked slave unit

	Bit 3	Bit 2	Bit 1	Bit 0	Function
0H	0	0	0	0	Slave status (SSR) read
4H	0	1	0	0	Lock address read (Lower 8 bits)
5H	0	1	0	1	Lock address read (Upper 4 bits)

(1) Slave status (SSR) read (control bits: 0H, 6H)

By reading the slave status, the master can understand why the slave has not returned the acknowledge bit (ACK). The slave status is determined in respect to the result of the last communication performed by the slave unit. Moreover all slaves can supply the information of slave status as configured below:

Slave Status

MSB							LSB
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Table 13.4.7c Meaning of Slave Status

Bit	Value	Meaning	
Bit 0 Note1	0	Slave transmit buffer is empty	
	1	Slave transmit buffer is not empty	
Bit 1 Note2	0	Slave receive buffer is not full	
	1	Slave receive buffer is full	
Bit 2	0	Unit is not locked	
	1	Unit is locked	
Bit 3	0	Fixed to '0'	
Bit 4 Note3	0	Slave transmission is stopped	
	1	Slave transmission is enabled	
Bit 5	0	Fixed to '0'	
Bit 6 Bit 7	00	Mode 0	Indicates the highest mode that is supported by the unit Note 4
	01	Mode 1	
	10	Mode 2	
	11	Prohibited	

Note1: The slave transmit buffer is the buffer accessed when data is read (control bits: 3H, 7H). This buffer is same as the write data buffer (WDB).

Note2: The slave receive buffer is the buffer accessed when data is written (control bits: 8H, AH, BH, EH, FH). This buffer is same as the read data buffer (RDB).

Note3: This bit can be selected by the PCOM bit in command register (CMRH).

Note4: For MB90580 series, 10 is fixed.

(2) Data/command transfer (control bits: read (3H, 7H), write (AH, BH, EH, FH))

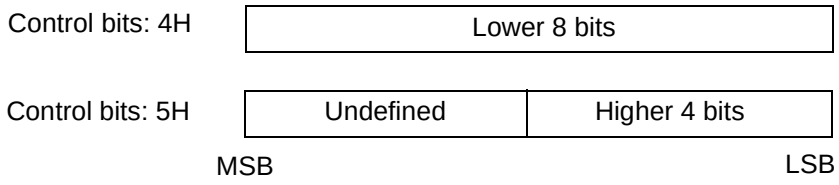
If the control bits indicate data read (3H, 7H), the data in the data buffer of the slave is read by the master. If the control bits indicate data write (BH, FH) or command write (AH, EH), the data received by the slave is processed in accordance with the operation regulation of that slave.

Note1: The data or command can be selected with the user's own decision in the corresponding system.

Note2: The slave is also locked when control bits are 3H, AH and BH.

(3) Read lock address (control bits: 4H, 5H)

When the lock address is read (control bits: 4H, 5H), the address (12-bit) of the master that has issued the lock instruction is configured in 1-byte units as shown below and is read.



(4) Locking and unlocking

The lock function is used to transmit a message over two or more communication frames. The unit that has been locked cannot receive data from any unit other than the one that has locked it.

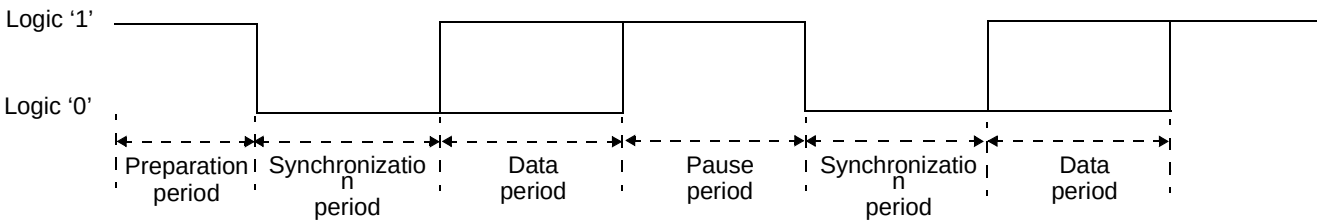
A unit is locked or unlocked as follows:

- Locking
  When the lock command has been executed (control bits: 3H, AH, BH) and the acknowledge bit ‘0’ in the telegraph length field has been transmitted or received, but the number of data bytes specified by the telegraph length bits cannot be transmitted or received successfully within the communication frame, then the slave is locked by the master, and the bit related to locking (bit 2) in the slave status is set.
- Unlocking
  When the lock command (control bits: 3H, AH, BH) or unlock command (control bits: 6H) has been executed, and the number of data bytes specified by the telegraph length bits can be transmitted or received successful within one communication frame, then the slave is unlocked by the master and the bit related to locking (bit 2) in the slave status is cleared.

Furthermore, the slave is not locked or unlocked while multiaddress communication is performed.

13.4.8 Bit format

The format of the bits constituting an IEBus communication frame is shown below:



Logic ‘1’: voltage difference between inter-bus wires (BUS+ and BUS-)is below 20 mV (low level)

Logic ‘0’: voltage difference between inter-bus wires (BUS+ and BUS-) is above 120 mV (high level)

Preparation period: First low-level period (logic ‘1’)

Synchronization period: Next high-level period (logic ‘0’)

Data period: Period indicating bit value (logic ‘1’ = low level, logic’0’ = high level)

The length of synchronization period and data period are almost the same.

The IEBus establishes synchronization for each bit. The specifications of the time of the entire bit and the time of the period assigned to the bit differ depending on the type of the transmit bit, and whether the unit is master or slave.

Moreover, the specified interval for every period (preparation, synchronization, data) in the communication are detected by both the master and slave. If data cannot be detected within that specified interval, timing error occurs in both master and slave, and then the communication ends and goes into standby mode.

13.5 Operation

13.5.1 IEBus control

(1) Master transmit

The unit is set as master transmit to transmit data to the slave by sending data/command control bits as AH, BH, EH or FH. The sequences for operating in master transmit are described as below:

1. The master address is written in unit address register (MAWH, MAWL), the slave address is written in slave address register (SAWH, SAWL), multiaddress bit and control bits are written into multiaddress, control bit set register (DCWR). After that, the command register (CMRH) is set to release communication inhibit mode.
2. When the master has won the arbitration (after the master address field is finished), the state code (0H) indicating start of transmission is set in ST3-0 of status register (STRL) and transmit interrupt occurs. At this time, the number of transmitted data byte is required to write into telegraph length set register (DEWR) and transmit data is set in write data buffer (WDB), unless the WDb is not full.
3. When one byte of data is transmitted, the number of data in WDB is deducted by one. According to the setting of TIT1, TIT0 in command register (CMRL) that control the interval for writing data in WDB, the transmit interrupt occurs. At that time, the state code equals ‘1H’ indicating data transmission and WDB is not full. Therefore, data should be written in WDB.
4. If the specified number of data or command has been transmitted correctly, the state code (2H) indicating transmission ends normally is set in ST3-0 of status register (STRL), the EOD bit in STRH is set and transmit interrupt occurs.
5. If error occurs during transmission or in multi-frame communication the number of data byte specified in telegraph bit set register (DEWR) cannot be transmitted completely, the state code (3H) indicating transmission ends without all data are transmitted is set in ST3-0, and transmit interrupt occurs. At this time, the content of communication error can be known by checking the status of TSL, PEF, TE in status register (STRH).

When timing error is occurred during the transmission, the data stored in WDB can’t be transmitted. If the TE bit is cleared and re-transnit sequence is executed, those data left in WDB, excluding the data byte that has timing error will be re-transmit. In order to perform new data transmission, the bit WDBC in CMRL must be written ‘1’ to clear the write data buffer.

(2) Slave transmit

■ Data transmit

This mode is set when the master sends control bits either 3H or 7H to slave and requests it to transmit data back to the master. The sequences for operating in slave data transmit are described as below:

1. After receiving master control code 3H or 7H, the data code (0H) indicating transmit start is set to ST3-0 of status register and the transmit interrupt is occurred. At this time, set the telegraph length set register (DEWR) with the nunmber of byte of data which are required to be transmitted. The transmit data can be written to WDB, provided that WDB is not full.
2. During the start of telegraph field transmission, the status register bits ST3-0 are set to 1H indicating data transmission in progress and transmit interrupt occurs. At that moment, transmit data can be written to the WDB, provided that WDB is not full.
3. When one byte of data is transmitted, the number of data in WDB is deducted by one. According to the setting of TIT1, TIT0 in command register (CMRL) that control the timing for writing data in WDB, the transmit interrupt occurs. Data can be written in WDB if state code is still (1H) and WDB is not full.
4. If the specified number of data or command has been transmitted correctly, the state code (2H) indicating transmission ends normally is set in ST3-0 of status register (STRL), the EOD bit in STRH is set and transmit interrupt occurs.

5. If error occurs during transmission or in multi-frame communication the number of data byte specified in DEWR cannot be transmitted completely, the state code (3H) indicating transmission terminated without all data transmitted is set in STRL:ST3-0. Transmit interrupt will occur. At this time, the content of communication error can be known by checking the status of TSL, PEF, TE in status register (STRH).

The interval between setting ST3-0 and the first data is transmitted out from WDB is shown below:

Table 13.5.1a Time required to write transmit data to WDB after transmit interrupt has occurred

Mode	Time (µS)	Number of cycles
0	approx. 158	approx. 1900
1	approx. 40	approx. 480
2	approx. 29	approx. 350

Note :

1. Number of transmit bytes and transmit data can be set during the interrupt generated after the receiving of control bits.
2. As the time between the WDB empty interrupt and the next telegraph bit transmit interrupt is very short, thus it is recommended to take the following precautions when the first transmit data is required to write into WDB.
   - Only write data to WDB after the WDB empty confirmation.
   - In case of setting the transmit data byte count, it is required to set the value within the time listed in Table 13.5.1a. The default value of the transmit data byte count will transmit 256bytes.
   - If at least 1 byte of data is not set within the time listed in below after the transmit interrupt, WDB will be detected as empty. Error will be occurred after the transmission of telegraph field and the communication will be terminated.

■ Slave status, lock address transmit

When the control bits 0H, 4H, 5H, 6H has been received from the master, the slave status, lock address are automatically transmitted to the master. In this way, there is no need to write data into WDB, but it is required to set 1H to the telegraph bit setting register.

(3) Master receive

The unit is set as master receive for getting data, slave status and lock address from the slave by first sending control bits 0H, 3H, 4H, 5H, 6H or 7H. The sequences for operating as master receive are described as below:

1. When the slave receives the control bits, it transmits the telegraph length bit. Then after the master receives these telegraph length bits and returns the acknowledge bit, the number of received data byte is written into the telegraph length read register (DERR). At this moment, no interrupt occurs.
2. After the acknowledge bit in telegraph length field is sent by the master, data reception will be started. And for each received data byte, the master stores it in the read data buffer (RDB).
3. After eight bytes of data are received, the state code (5H) is set in ST3-0 of status register (STRL), receive interrupt occurs.
4. When the last byte of data is received and stored in RDB within one communication frame, the state code (5H) is set in ST3-0 and receive interrupt occurs. This interrupt will be generated even thought RDB is not full.
5. If error is occurred during reception, or the maximium number of data byte has been received in one communication frame, the master cannot received the number of data byte specified in telegraph field and communication is terminated. The state code (7H) indicating master receive ends without all data are received is set in ST3-0, and receive interrupt occurs.

(4) Slave receive

This mode is set when the slave unit receive control bits AH, BH, EH or FH from the master. The sequences for operating as slave receive are described as below:

1. After the slave returns the acknowledge bit in telegraph length field, the number of receive data byte is written in the telegraph length read register (DERR). At this moment, no interrupt occurs.
2. Following the telegraph length field is the data field, the master starts transmitting data and each received data byte is stored in the read data buffer (RDB).
3. After eight bytes of data are received, the state code (9H) indicating slave receive buffer full is set in ST3-0 of status register (STRL), and receive interrupt occurs. If the receive interrupt occurs, the RDB can be read after the confirmation of buffer not empty.
4. When the last byte of data in one communication frame is received and stored in RDB, the state code (AH) indicating slave receive ends normally is set in ST3-0 and receive interrupt occurs. This interrupt will occur even thought the buffer is not full.
5. If error is occurred during reception or the maximium number of data byte has been re3ceived in one commmunication frame, the slave cannot receive the number pf data byte specified in telegraph field and communication is terminated.. The state code (BH) indicating slave receive ends without all data are received is set in ST3-0, and receive interrupt occurs.

(5) Multiaddress receive

1. After the slave has received the telegraph length field, the number of receive data byte is written in the telegraph length read register (DERR). At this moment, no interrupt occurs.
2. After the telegraph lenght field is received, each correctly received data byte is stored in the read data buffer (RDB).
3. After eight bytes of data are received, the state code (DH) indicating multiaddress receive buffer full is set in ST3-0 of status register (STRL), and receive interrupt occurs. If the receive interrupt occurs, the RDB can be read after the confirmation of buffer not empty.
4. When the last byte of data in one communication frame is received and stored in RDB, the state code (EH) indicating multiaddress receive ends normally is set in ST3-0 and receive interrupt occurs. This interrupt will occur even thought the buffer is not full.
5. If error is occurred during reception or the maximium number of data byte has been received in one communication frame, the slave cannot receive the number of data byte specified in telegraph field and communication is terminated. The state code (FH) indicating multiaddress receive ends without all data are received is set in ST3-0, and receive interrupt occurs.

For detail description on ST3-0, please refer to Table 13.5.2a.

13.5.2 Communication status

In the status register, there are four bits ST3-0 indicating the status code. After the status code has been set, interrupt request is generated. During the interrupt routine, the communication status can be investigated by reading the status register. But at the beginning of master, slave and multiaddress receive, no interrupt will be generated

(1) Master, slave data transmit (transmit interrupt occurs)

When the unit won the arbitration in multiaddress or master address field, it becomes master unit. Then data/command is transmitted to or data is received from the slave, and the status code ST3-0 is set and shown as below:

Table 13.5.2a Meaning of status code ST3-0 for master, slave transmit

Code Name	Code ST3-0	Content
Transmit starts	0000	Indicates start of master/slave transmission. 1) master transmit Indicates the master address field in communication frame has been transmitted, and the unit has won in arbitration as the master. 2) slave transmit Indicates that the unit has received control bits 0H, 3H, 4H, 5H, 6H, 7H from the master that requests data transmission, and slave data transmission is started.
Transmit data	0001	Indicates that data is transmitting by Master unit or Slave unit. This control code will be set after the starting of telegraph length field transmission.
Transmit ends normally	0010	Indicates that the number of data transmit specified by telegraph length field has been completed within one communication frame
Ends without all data being transmitted	0011	Indicates that the communication has ended without transmitting the number of data specified by telegraph length field in one communication frame.

(2) Master receive (receive interrupt occurs)

When the unit won the arbitration in multiaddress or master address field, it becomes master unit. Then data, status or log address are received from slave unit, and the status code ST3-0 is set and shown as below:

Table 13.5.2b Meaning of status code ST3-0 for master receive

Code Name	Code ST3-0	Content
Master receive starts	0100	Indicates that the master has received the telegraph field correctly from the slave and master reception is started but receive interrupt does not occur at this moment.
Master receive data full	0101	Indicates that the receive data buffer RDB for master reception is full (eight byte of data has been received), and the host controller is requested to read data from the RDB.
Master receive ends normally	0110	Indicates the number of data specified by the telegraph field has been received within one communication frame.
Ends without all data being received	0111	Indicates that the communication has ended without receiving the number of data specified by telegraph length field in one communication frame.

(3) Slave receive (receive interrupt occurs)

When data/command is received from the master unit, the status code ST3-0 is set and shown as below:

Table 13.5.2c Meaning of status code ST3-0 for slave receive

Code Name	Code ST3-0	Content
Slave receive starts	1000	Indicates that the slave unit has received the telegraph field correctly from master unit and slave reception is started but receive interrupt does not occur at this moment.
Slave receive data full	1001	Indicates that the receive data buffer RDB for slave reception is full (eight byte of data has been received), and the host controller is requested to read data from the RDB.
Slave receive ends normally	1010	Indicates the number of data specified by the telegraph field has been received within one communication frame.
Ends without all data being received	1011	Indicates that the communication has ended without receiving the number of data specified by telegraph length field in one communication frame.

(4) Multiaddress receive (receive interrupt occurs)

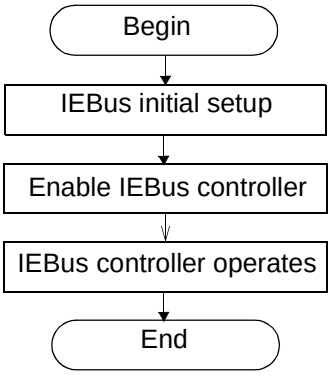
When the data/command in multiaddress communication are received from the slave unit, the status code ST3-0 is set and shown as below:

Table 13.5.2d Meaning of status code ST3-0 for multiaddress receive

Code Name	Code ST3-0	Content
Multiaddress receive starts	1100	Indicates that the slave unit has received the telegraph field correctly from master unit and multiaddress reception is started but receive interrupt does not occur at this moment.
Multiaddress receive data full	1101	Indicates that the receive data buffer RDB for slave reception is full (eight byte of data has been received), and the host controller is requested to read data from the RDB.
Multiaddress receive ends normally	1110	Indicates the number of data specified by the telegraph field has been received within one communication frame.
Ends without all data being received	1111	Indicates that the communication has ended without receiving the number of data specified by telegraph length field in one communication frame.

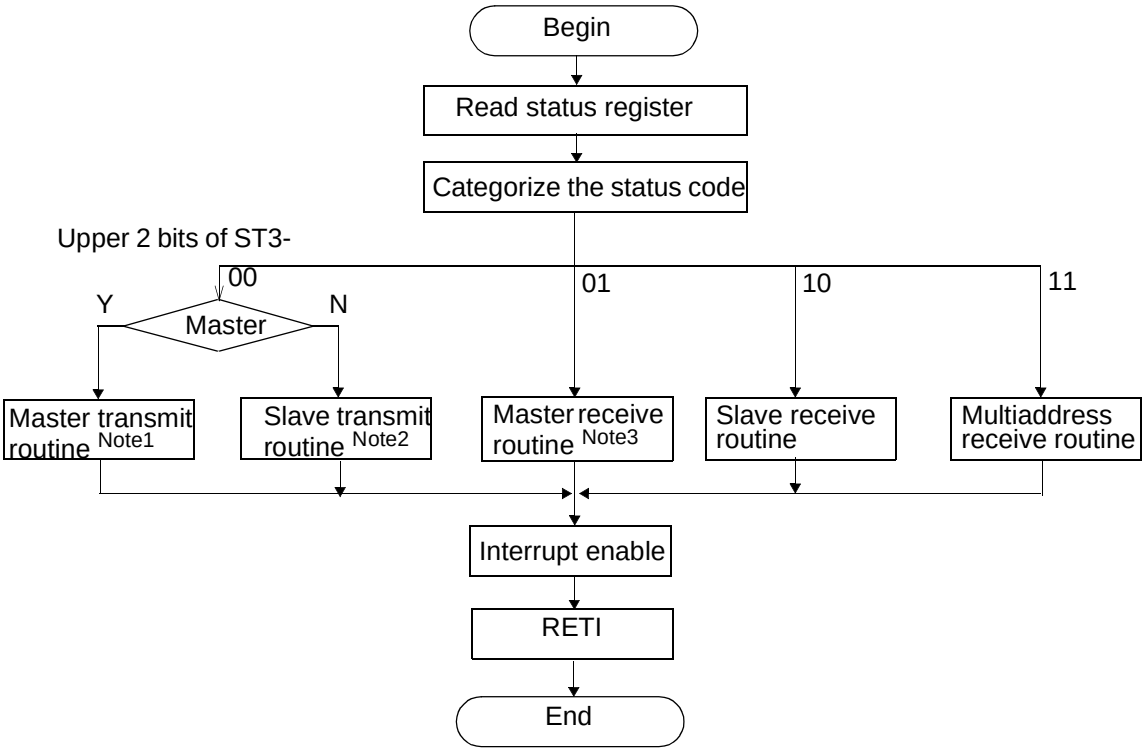
13.5.3 Program flow example for IEBus controller

(1) Main routine



(2) Interrupt routine

This routine is executed when start of transmission or end of reception. In interrupt routine, the status code (ST3-0) in status register STRL is read, then the transmit data can be written or receive data can be read.



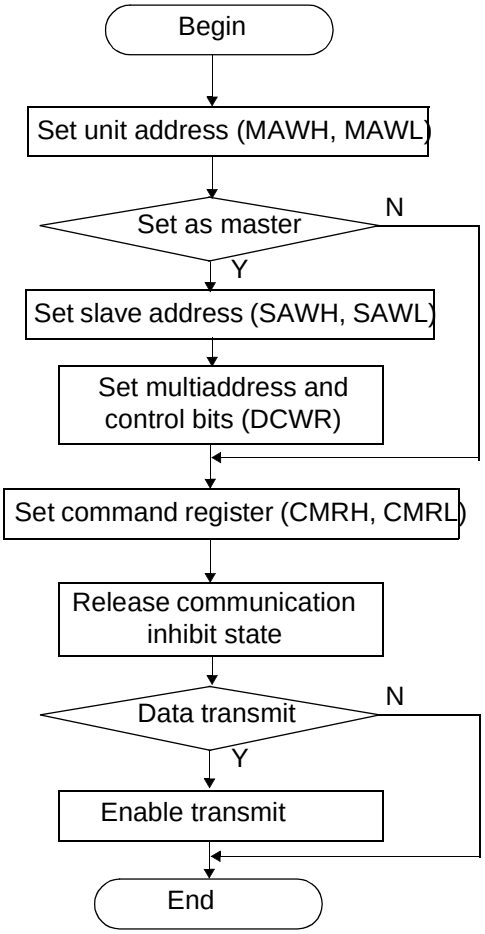
Note1: Refer to master transmit routine

Note2: Refer to slave transmit routine

Note3: Refer to master receive routine

(3) IEBus initial setup

The initial setup sequence includes setting its unit address, the command register and releasing the communication inhibit state. If the unit is not set as master, there is no need to set the slave address in slave address register. In converse, if the unit is set as master, there is no need to set the mutiaddress byte and control byte.



When the unit is set as master, the unit address becomes master address. When the unit is set as slave, the unit address is used to compared with that in slave address field.

For example selecting the communication mode in command register CMRH, CMRL

The bit PCOM in command register CMRH is set to ‘1’.

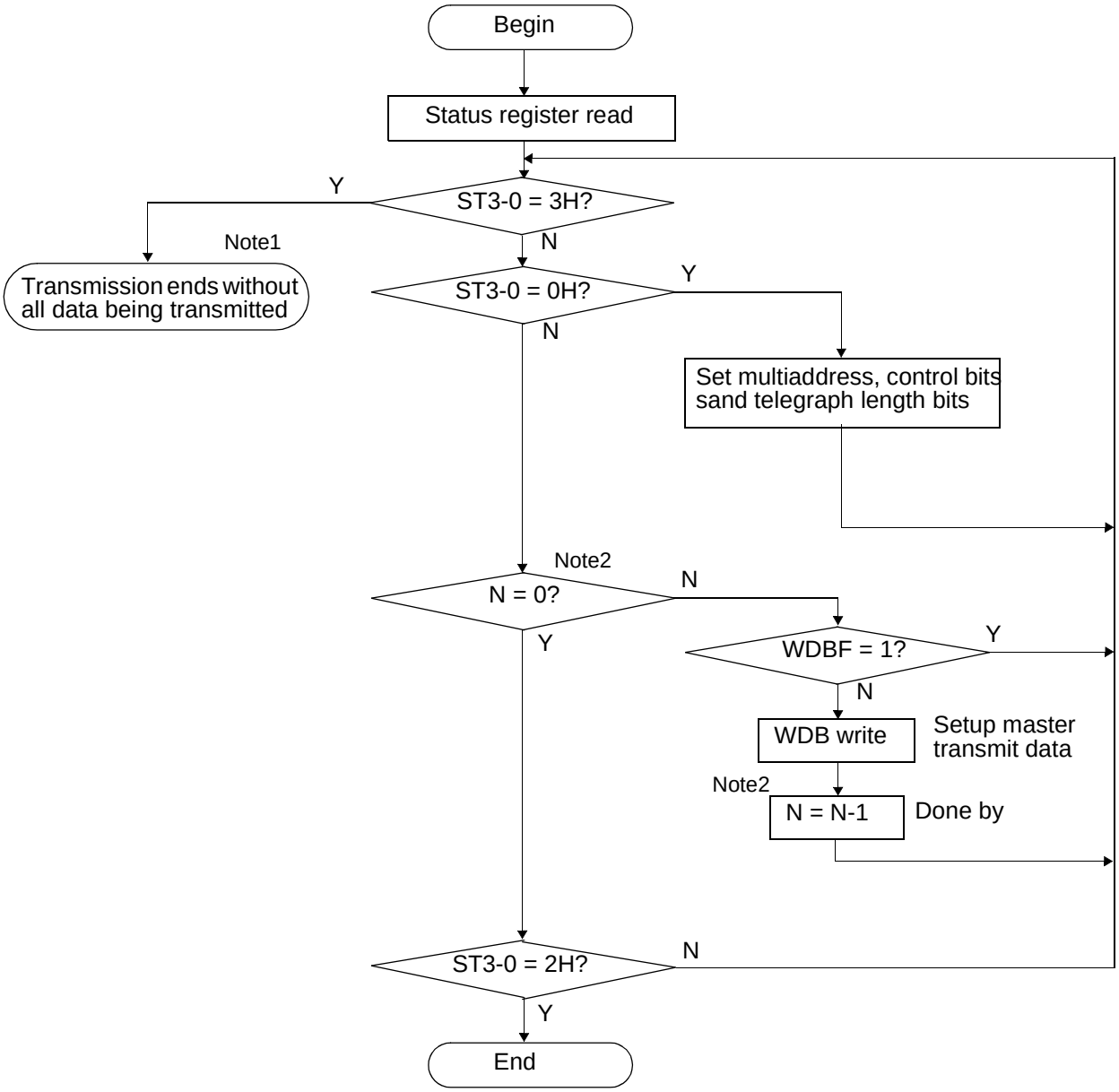
During slave receive, .GOTM and GOTS are not

Set GOTS bit to ‘1’ when GOTM bit is ‘1’/

Note: It can be executed by write operation of command register.

(4) Master transmit routine

After the communication inhibit state is released, the unit won the arbitration and acts as master. Then master transmit routine is used to transmit data to the slave. This routine is executed inside interrupt routine with ST3-0 bits (upper 2 bits are 00) in status register indicating the status as master transmit has been set.



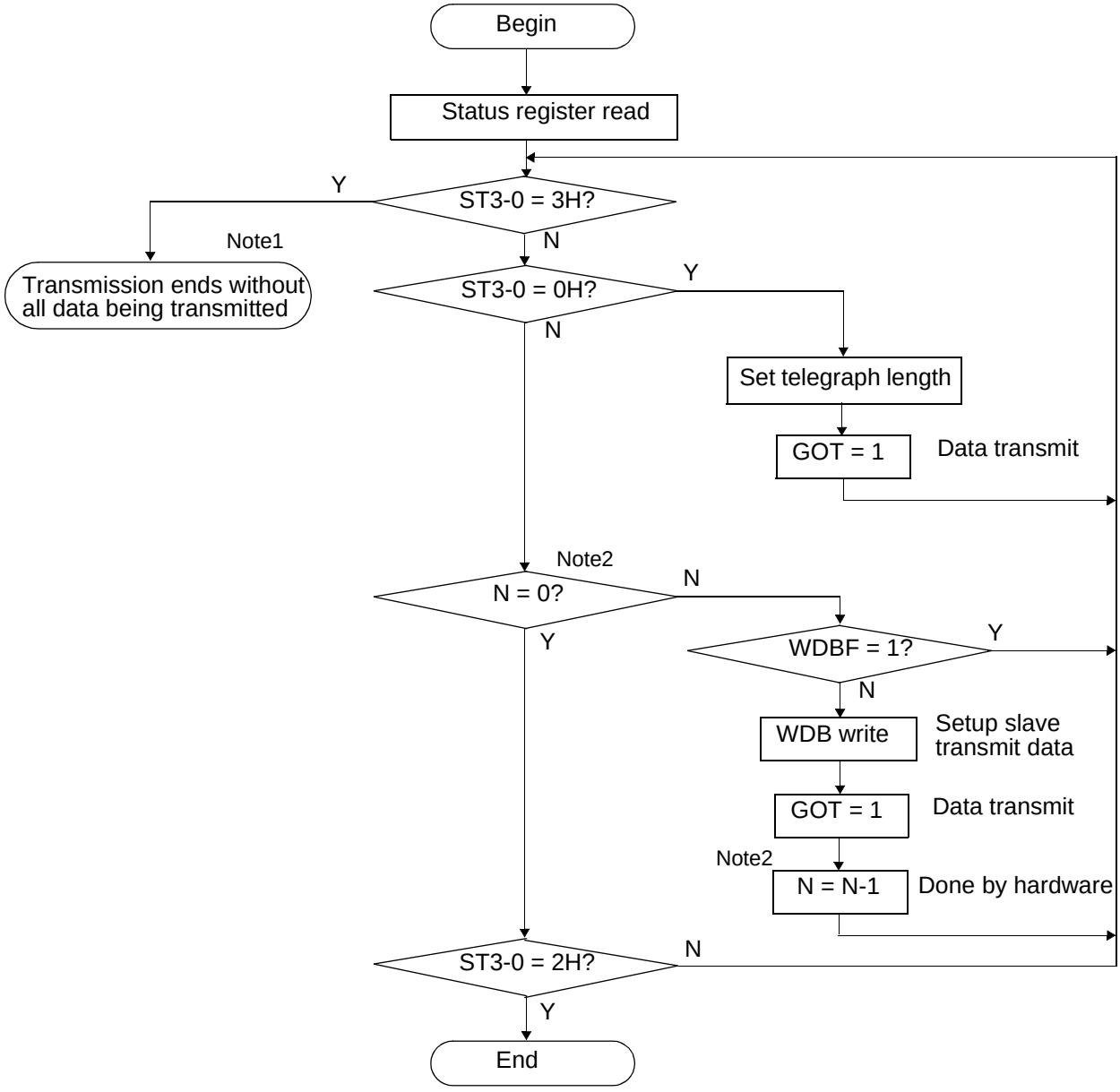
Note1: The reason for the abnormal termination of transmission can be known by reading the TE, PEF and ACK bits of status register. The remaining data that can't be transmitted will be sent out when GOTM bit in command register CMRH is written '1' again.
In order to stop the master transmit and clear the WDB, the bit WDBC in CMRL is written '1'.

Note2: Please do not wrtie WDB when WDBF=1

Note3: N is the number of data byte for master transmit

(5) Slave transmit routine

After the slave receives the control bits and is set as slave transmit, this routine is used to transmit data to the master. This routine is executed inside interrupt routine with ST3-0 bits (upper 2 bits are 00) in status register indicating the status as slave transmit has been set.



Note1: The reason for the abnormal termination of transmission can be known by reading the TE, PEF and ACK bits of status register. The remaining data that can't be transmitted will be sent out when GOTS bit in command register CMRH is written '1' again.
In order clear the WDB, the bit WDBC in CMRL is written '1'.

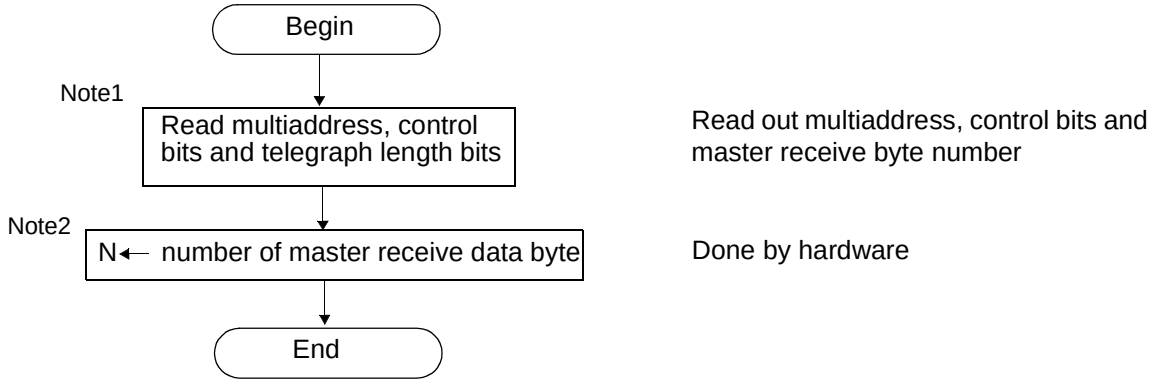
Note2: Please do not wrtie WDB when WDBF=1

Note3: N is the number of data byte for slave transmit

(6) Master receive routine

After the master transmit the control bits, this routine is used for the master to receive data, slave address or log address from the slave. This routine is consisted of four parts depending on the content of ST3-0.

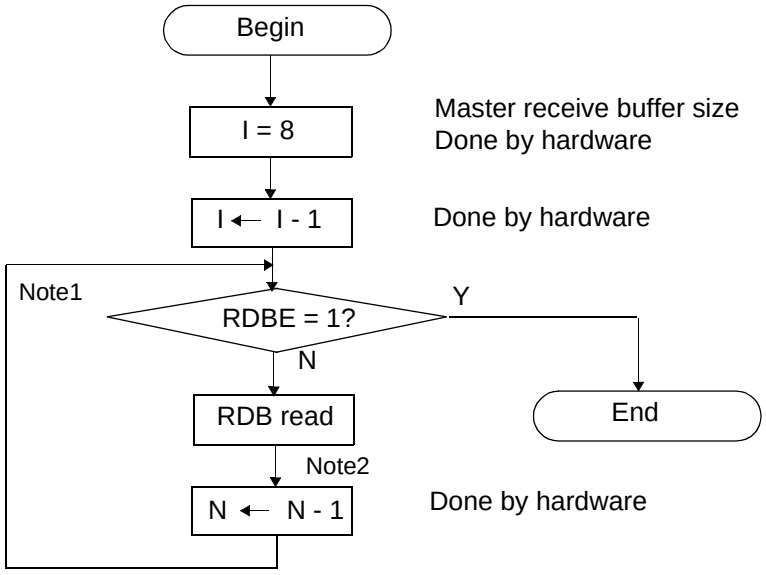
1. Start of master reception (ST3-0 is 4H)
   When the master receive the telegraph length field from the slave correctly, the status code 4H is set and the master reception starts. However, interrupt will not occur at that time..



Note1: The status register need not be read because each registers are set. However, it is required to take care the timing of setting the registers. Before updating the register contents, read the registers first.

Note2: N is the number of data byte for master receive.

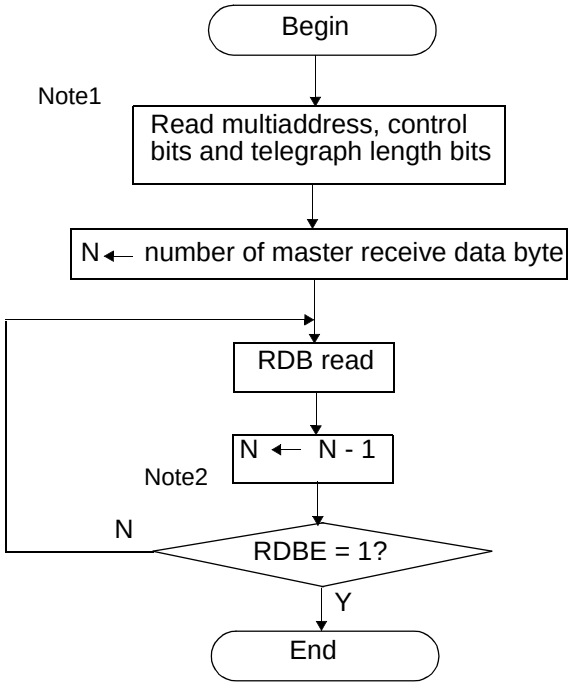
2. Master received data read request (ST3-0 is 5H)



Note1: Do not read RDB when RDBE=1.

Note2: N is the number of data byte for master receive.

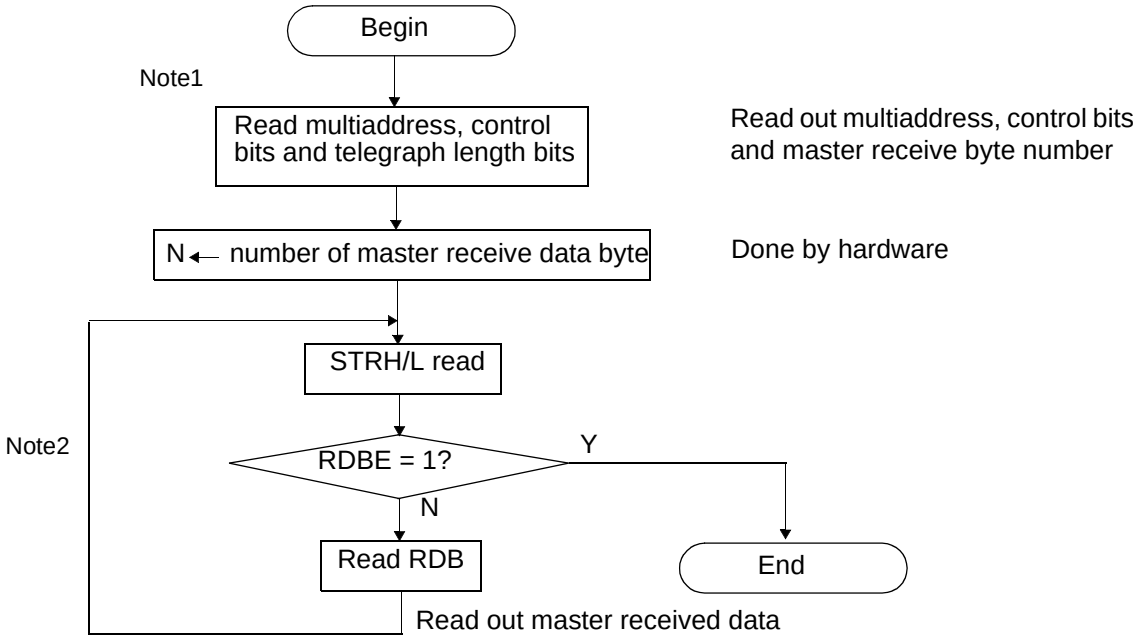
3. Master receive ends normally (ST3-0 is 6H)



Note1: It is not required to read the status as these data are stored in different registers. As the data will be updated, read the register before the next communication frame.

Note2: Do not read RDB when RDBE = 1.

4. Master reception ends without all data being received (ST3-0 is 7H)



Note1: It is not required to read the status as these data are stored in different registers. As the data will be updated, read the register before the next communication frame.

Note2: Do not read RDB when RDBE = 1.

The routine for slave receive and multiaddress receive is the same as that of master receive but the status code is different. Please refer to section 13.5.2 for the status code of slave receive and multiaddress receive.

13.5.4 Timing Diagram of Multiple Frame Transmission

1. When setting ‘1’ on WDBC (Master side of master transmission)

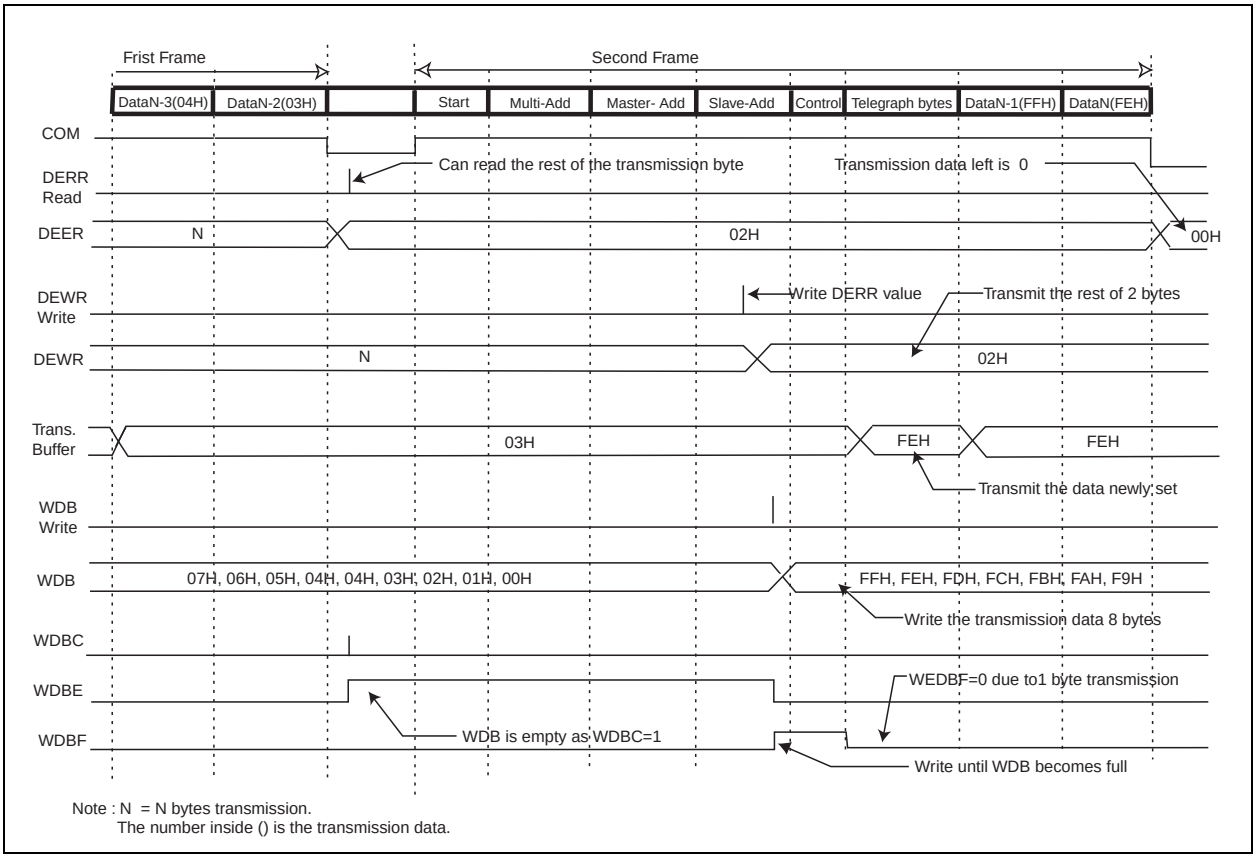
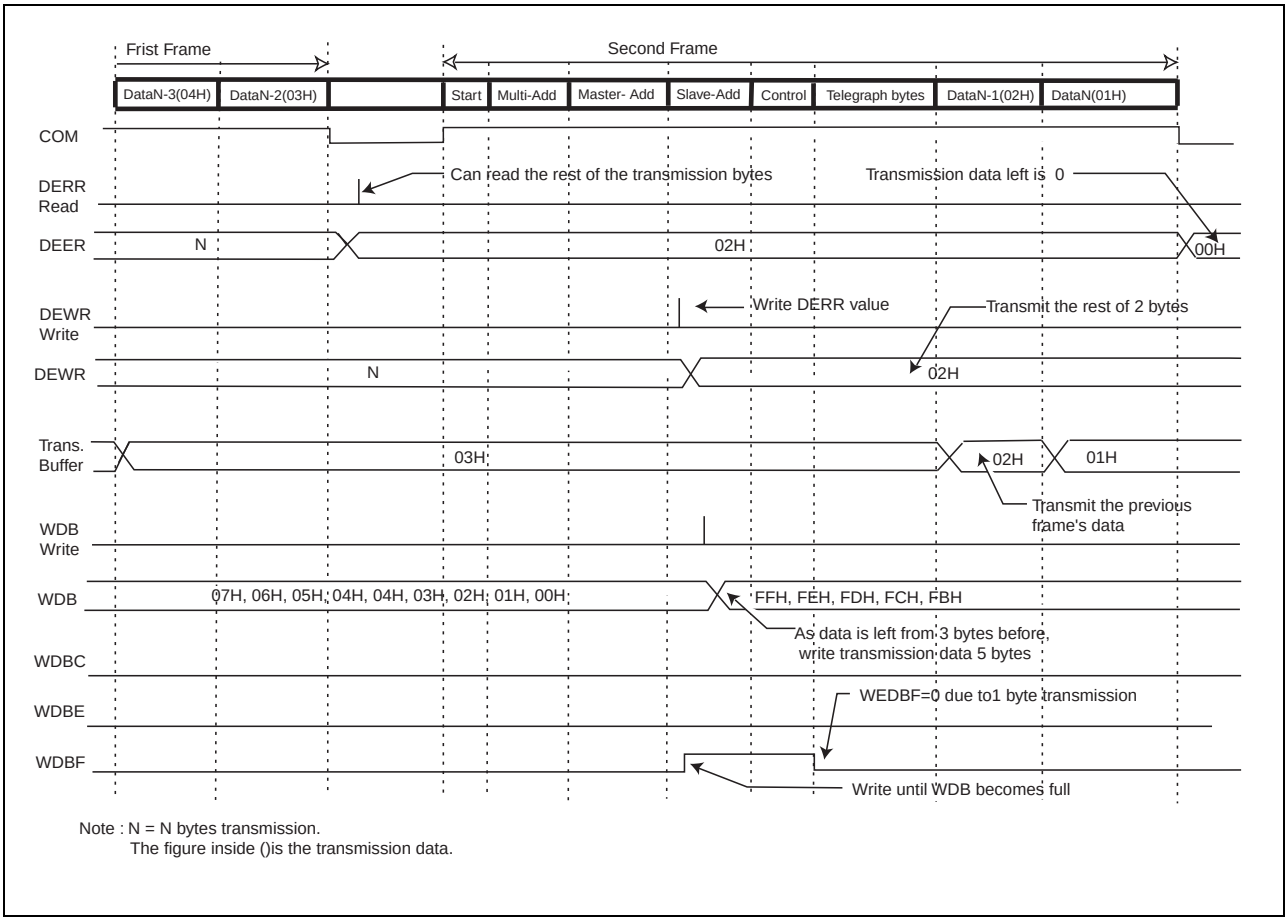


Figure 13.5.4a When setting ‘1’ on WDBC (Master side of master transmission)

2. When setting ‘0’ on WDBC (Master side of master transmission)



Note : N = N bytes transmission.
The figure inside ()is the transmission data.

Figure 13.5.4b When setting ‘0’ on WDBC (Master side of master transmission)

13.5.5 Timing diaram of transmission data when an error is generated

1. The following is an example when the master transmission, an error is generated at the second byte data on the slave side. NAK is received by the master. the following data is transmitted at the second frame.

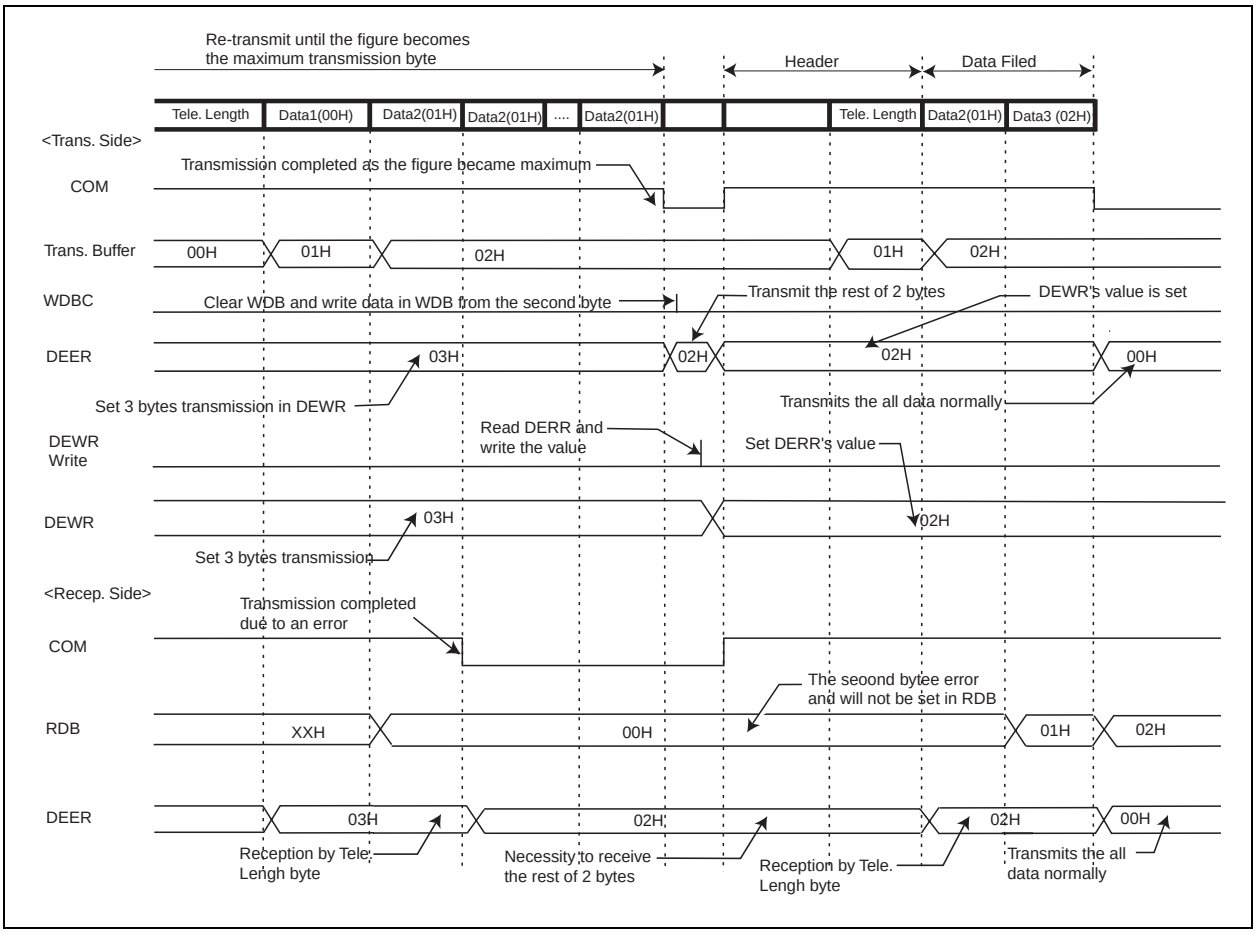


Figure 13.5.5a Error happened on the Slave side when master transmission

2. The following is an example when the master transmission, an error is generated at the second byte data on the master side. The following data is transmitted at the second frame.

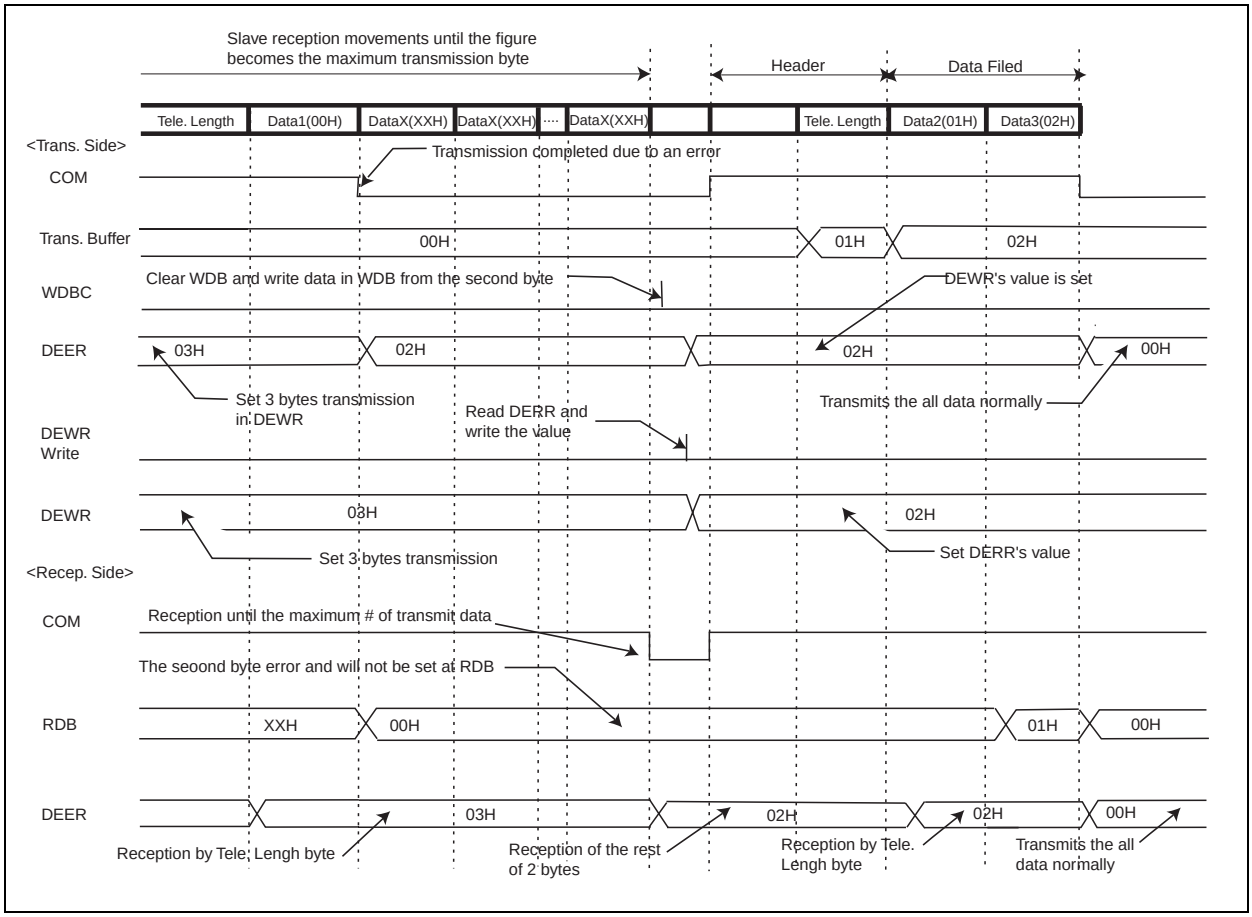


Figure 13.5.5b Error happened on the Master side when master transmission

Chapter 14:
8/16-Bit PPG

14.1 Outline

The 8/16-bit PPG timer is an 8-bit reload timer module, and outputs PPG by control pulse output according to timer operation.

The hardware includes two eight-bit down counters, four eight-bit reload registers, one 16-bit control register, two external pulse output pins, and two interrupt outputs. The following functions are implemented:

- 8-bit PPG output 2-CH independent operation mode
  This is a mode for operating independent 2-CH 8-bit PPG timer, in which PPG0 and PPG1 pins correspond to outputs from PPG0 and PPG1 respectively.
- 16-bit PPG output operation mode
  In this mode, PPG0 and PPG1 are combined to be operated as a 1-CH 8/16-bit PPG timer operating as a 16-bit timer. Because PRG0 and PRG1 outputs are reversed by an underflow from PRG1 outputting the same output pulses from PRG0 and PRG1 pins.
- 8 + 8 bit PPG output operation mode
  In this mode, PPG0 is operated as an 8-bit pre-scaler, in which an underflow output of PPG0 is used as a clock source for PPG1. A toggle output of PPG0 and PPG output of PPG1 are output from PPG0 and PPG1 respectively.
- PPG output operation
  The 8/16-bit PPG timer can output pulse waveforms with variable period and duty ratio. Also, it can be used as D/A converter in conjunction with an external circuit.

14.2 Block Diagram

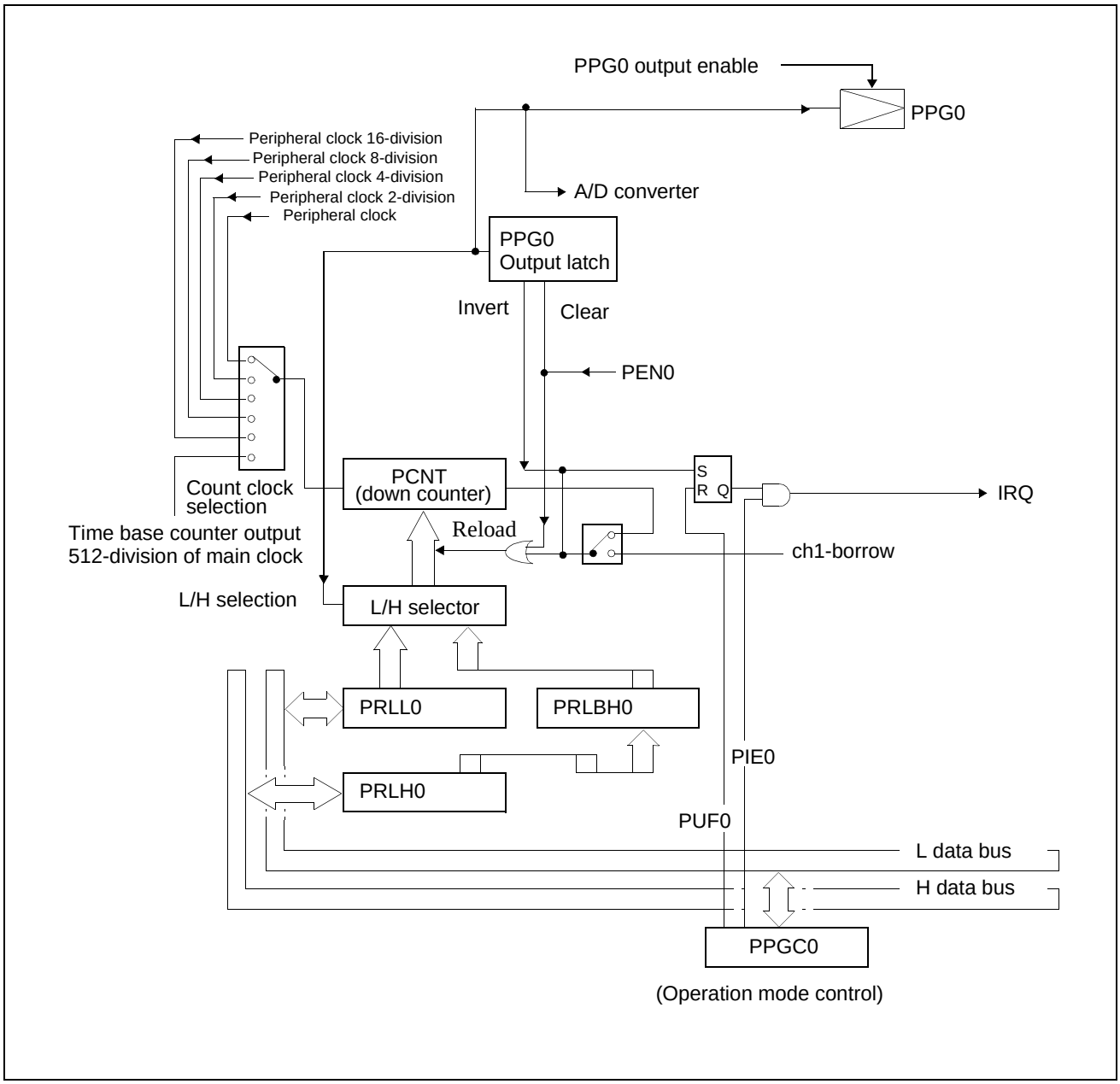


Figure 14.2a 8-bit PPG ch0 block diagram

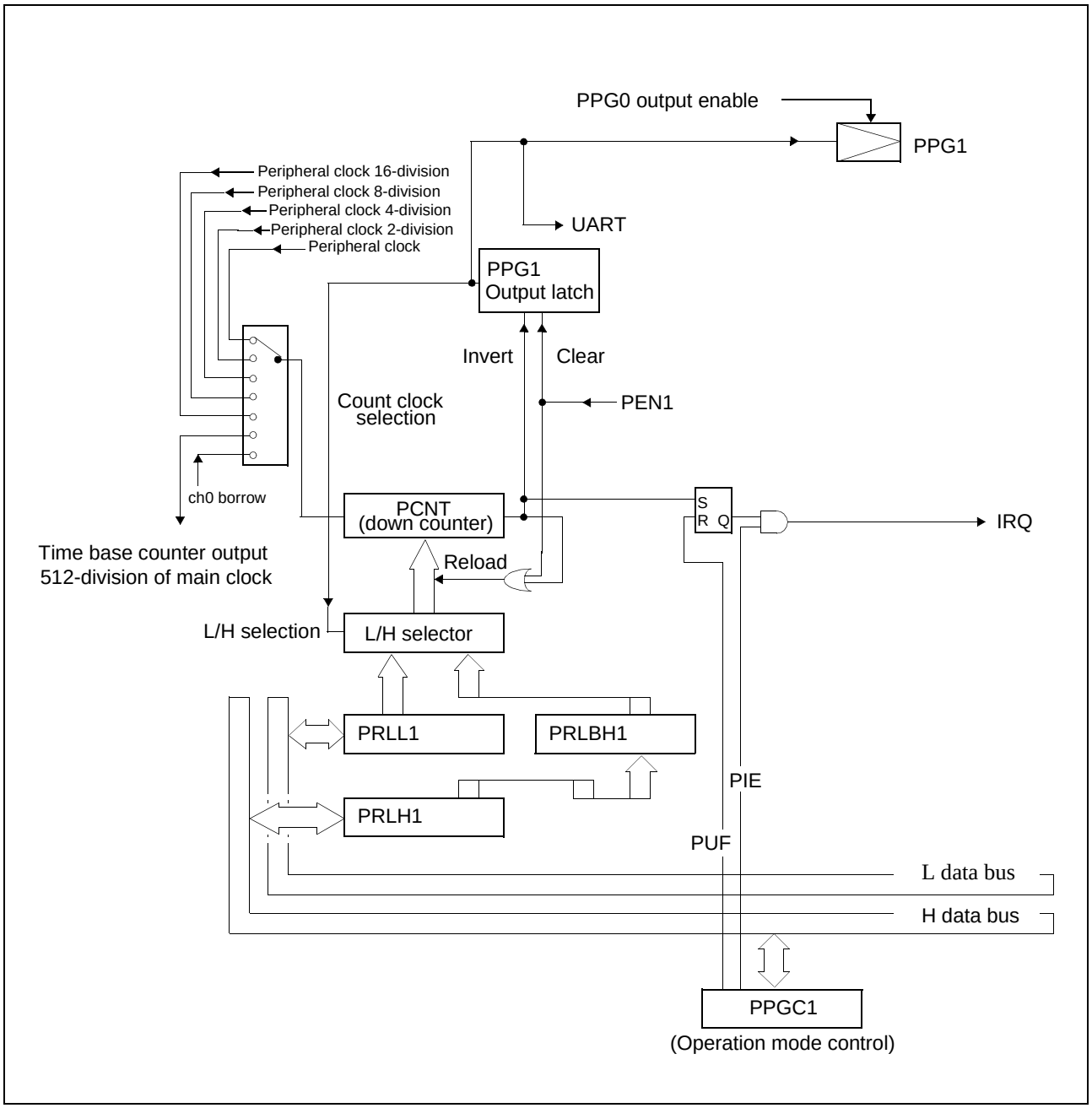


Figure 14.2b 8-bit PPG ch1 block diagram

14.3 Registers and Register Details

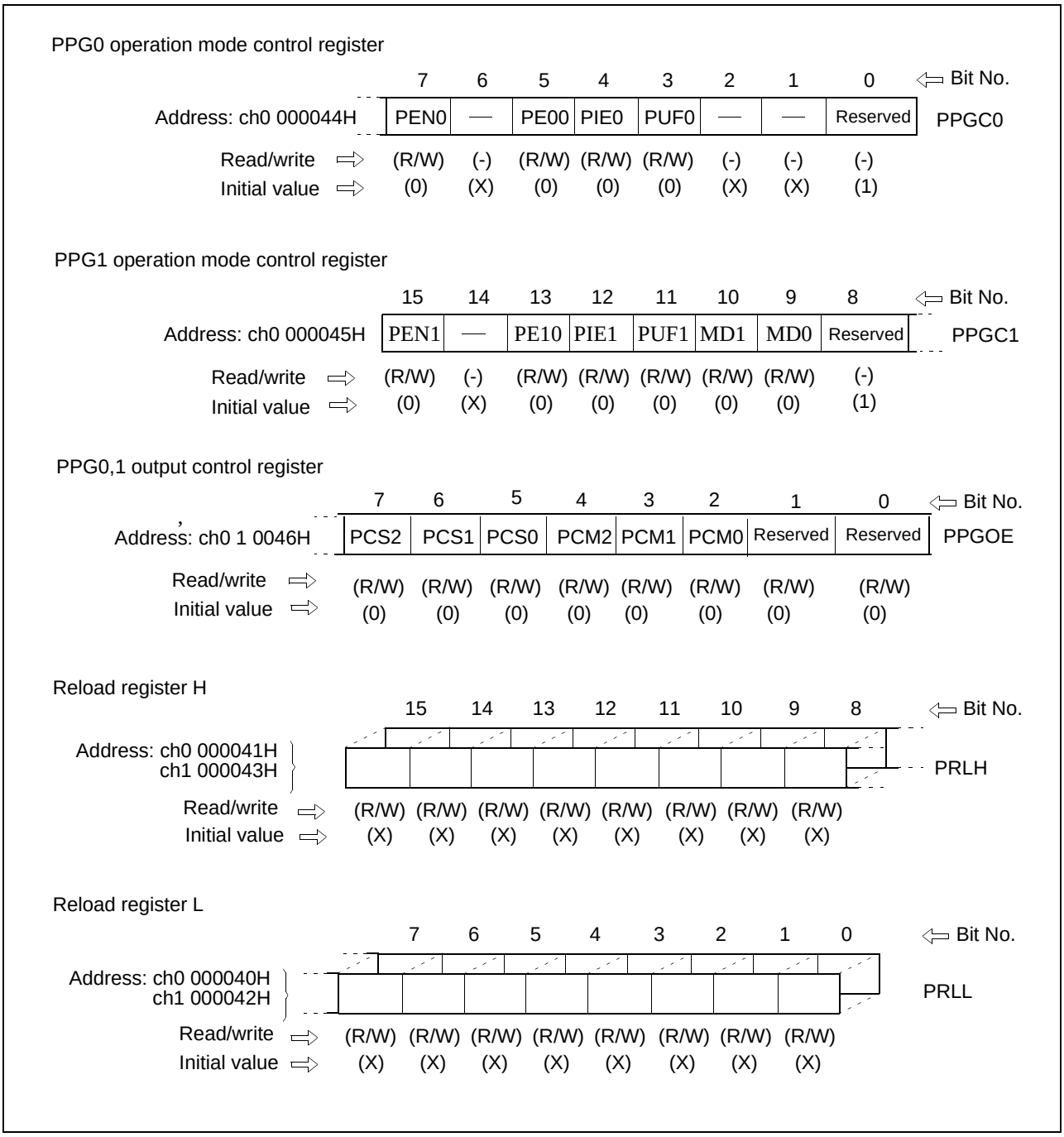


Figure 14.3a Registers of 8/16-bit PPG

14.3.1 PPG0 operation mode control register (PPGC0)

PPG0 operation mode control register

	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: ch0 000044H	PEN0	—	PE00	PIE0	PUF0	—	—	Reserved	PPGC0
Read/write ⇨	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(-)	(-)	(-)	
Initial value ⇨	(0)	(X)	(0)	(0)	(0)	(X)	(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 selects the PPG operation mode as described below.

PEN0	Operation
0	Stop ('L' level output maintained) [initial value]
1	PPG operation enabled

Setting this bit to 1 makes the PPG start counting.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 5] PE00 (PPG output enable 0): PPG0 pin output enable bit

This bit controls the PPG0 pulse output external pin as described below.

PE00	Operation
0	General-purpose port pin (pulse output disabled) [initial value]
1	PPG0 = pulse output pin (pulse output enabled)

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 4] PIE0 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

PIE0	Operation
0	Interrupt disabled [initial value]
1	Interrupt enabled

While '1' is written to this bit, an interrupt request is issued as soon as '1' is written to PUF0. 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.

Note: PIE0 is assigned the same interrupt vector number as that of 16-bit reload timer. When using EI^{2}OS in 16-bit reload timer, write '0' to PIE0.

[bit 3] PUF0 (PPG underflow flag): PPG counter underflow bit

This bit controls the PPG counter underflow as described below.

PUF0	Operation	
0	PPG counter underflow is not detected	[initial value]
1	PPG counter underflow is detected	

In 8-bit PPG 2ch mode or 8-bit prescaler + 8-bit PPG mode, ’1’ is written to this bit when an underflow occurs as a result of the ch0 counter value becoming between 00H and FFH. In 16-bit PPG 1ch mode, ’1’ is written to this bit when an underflow occurs as a result of the ch1/ch0 counter value becoming between 0000H and FFFFH. To set this bit to ’0,’ write ’0.’ Writing ’1’ to this bit is invalid. Upon a read operation during a read-modify-write instruction, ’1’ is read.

This bit is initialized to ’0’ 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.

14.3.2 PPG1 operation mode control register (PPGC1)

PPG1 operation mode control register

	15	14	13	12	11	10	9	8	⇦ Bit No.
Address: ch0 000045H	PEN1	—	PE10	PIE1	PUF1	MD1	MD0	Reserved	PPGC1
Read/write ⇨	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	
Initial value ⇨	(0)	(X)	(0)	(0)	(0)	(0)	(0)	(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 selects the PPG operation mode as described below.

PEN1	Operation
0	Stop ('L' level output maintained) [initial value]
1	PPG operation enabled

Setting this bit to 1 makes the PWM start counting.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 13] PE10 (PPG output enable 1): PPG1 pin output enable bit

PE10	Operation
0	General-purpose port pin (pulse output disabled) [initial value]
1	PPG1 = pulse output pin (pulse output enabled)

This bit controls the PPG1 pulse output external pin as described below.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 12] PIE1 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

PIE1	Operation
0	Interrupt disabled [initial value]
1	Interrupt enabled

While '1' is set in this bit, an interrupt request is issued as soon as '1' is written to PUF1. 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.

Note: PIE1 is assigned the same interrupt vector number as that of UART 0 transmission complete. When using EI^{2}OS in UART 0 transmissioin complete, write '0' to PIE1.

[bit 11] PUF1 (PPG underflow flag): PPG counter underflow bit

This bit controls the PPG counter underflow as described below.

PUF1	Operation
0	PPG counter underflow is not detected [initial value]
1	PPG counter underflow is detected

In 8-bit PPG 2ch mode or 8-bit prescaler + 8-bit PPG mode, ’1’ is written to this bit when an underflow occurs as a result of the ch1 counter value becoming between 00H and FFH. In 16-bit PPG 1ch mode, ’1’ is written to this bit when an underflow occurs as a result of the ch1/ch0 counter value becoming between 0000H and FFFFH. To set ’0’ in this bit, write ’0.’ Writing ’1’ to this bit is invalid. Upon a read operation during a read-modify-write instruction, ’1’ is read.

This bit is initialized to ’0’ 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 (setting inhibited)
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 PPGC0, always write 1 to this bit.

14.3.3 PPG0, 1 output pin control register (PPGOE)

PPG0,1 output control register

	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: ch0 1 0046H	PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	Reserved	Reserved	PPGOE
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 is an 8-bit control register that controls the pin output of this block.

[bits 7 to 5) PCS2 to 0 (PPG count select): Count clock selection bit

These bits select the channel 1 down counter operation clock 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-ms machine clock, 16 MHz)
1	0	0	Peripheral clock/16 (1-ms machine clock, 16 MHz)
1	1	1	Clock input from time base counter (128-ms, 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 PCS1 bit is invalid.

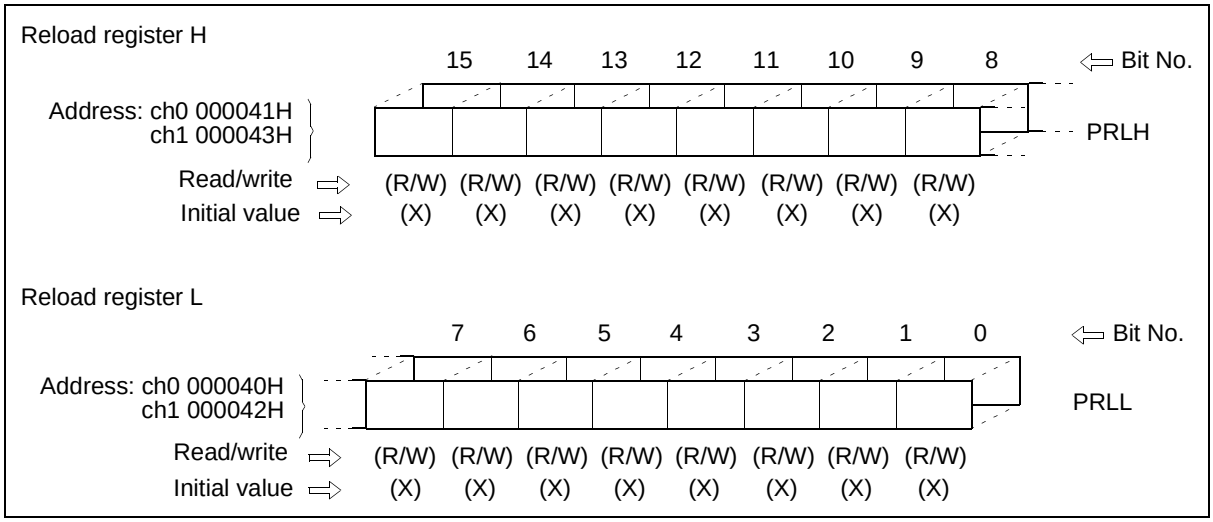
[bits 4 to 2] PCM2 to 0 (PPG count mode): Count clock selection bit

These bits select the channel 0 down counter operation clock 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-ms machine clock, 16 MHz)
1	0	0	Peripheral clock/16 (1-ms machine clock, 16 MHz)
1	1	1	Clock input from time base counter (128-ms, 4-Mhz source

This bit is initialized to ’000’ upon a reset. This bit is readable and writable.

14.3.4 Reload register (PRLL/PRLH)



These are 8-bit registers that hold the reload values for the PCNT down counter. Their roles are described below.

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, setting different values in PRLL and PRLH of ch1 may cause the PPG waveform of ch1 to vary in each cycle. Write the same value to PRLL and PRLH of ch0.

14.4 Operations

This block has 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 side (PRLL) and the other is for the H side (PRLH). The values stored in these registers are reloaded into the 8-bit down counter (PCNT), from the L side and H side in turn. Thus, the values are decremented for each count clock, and the pin output (PPG) value is inverted upon a reload caused by a counter borrow. This operation results in L-wide or H-wide pulse outputs, corresponding to the reload register value.

The operation is started and resumed by writing data in the corresponding register bit.

The table below lists the relationship between the reload operation and pulse outputs.

Table 14.4a Reload operation and pulse output

Reload operation		Pin output change		
PRLH	PCNT	PPG0x/1x [0	1]	Rise
PRLL	PCNT	PPP0x/1x [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 at any cycles. The PPG0 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 pulse 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 of the pulse wave to the L-level period). PPG continues operation until stop is specified explicitly.

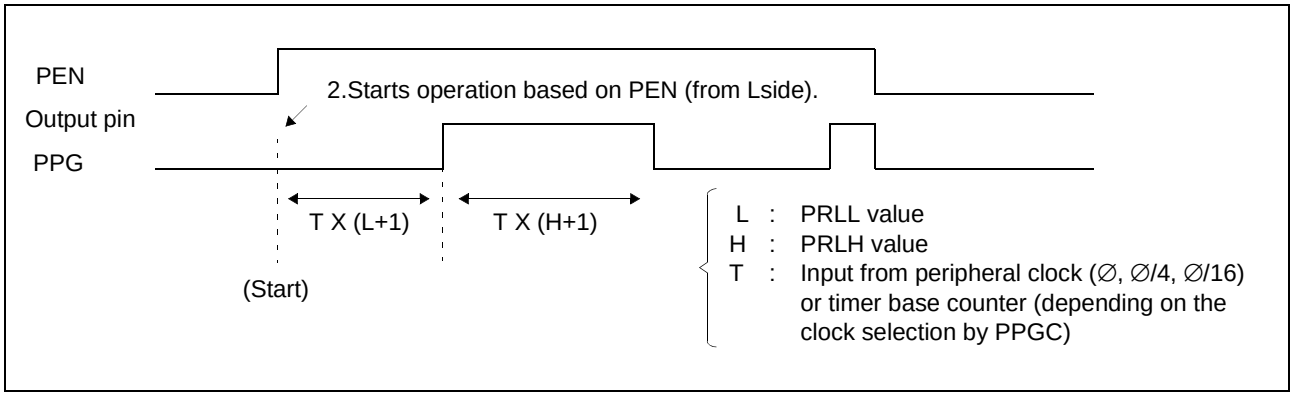


Figure 14.4a PPG output operation, output waveform

(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 00_H during 8-bit PPG operation or 0000_H during 16-bit PPG operation, the pulse width is equivalent to one count clock cycle. In addition, note that when the reload register value is FF_H during 8-PPG operation, the pulse width is equivalent to 256 count clock cycles. When the reload register value is $FFFF_H$ during 16-bit PPG operation, the pulse width is equivalent to 65536 count clock cycles. An example of pulse width calculation is given below.

P1=T X (L+1)
Ph=T X (H+1)

L : PRLL value
H : value
T : Input clock cycle
Ph : High pulse width
Pl : Low pulse width

(4) Count clock selection

The count clock used for the operation of this block is supplied from a peripheral clock or time base counter. The count clock can be selected from six types.

Select ch0 clock at bit 4 to 2 (PCM2 to 0) of the PPGOE register, and ch1 clock at bit 7 to S (PCS2 to 0) of the PPGOE register.

The clock is selected from a peripheral clock 1/16 to 1 times higher than a machine clock or an input clock from a time base counter.

In 8-bit prescaler + 8-bit PPG mode or 16-bit PPG mode, however, the value in bit 14 (PCS1) of the PPGC1 register is invalid. The register is invalid because ch1 PPG receives a count clock from ch0.

When the time base counter input is used, the first count cycle after a trigger or a stop may be shifted. The cycle may also be shifted if the time base counter is cleared during operation of this module.

In 8-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.

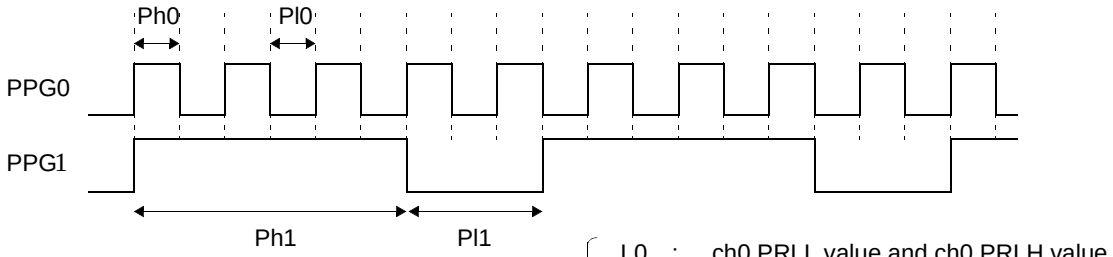
(5) Pulse pin output control

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. Use bit 5 (PE0) of the PPGC0 register for the PPG0 pin, bit 13 (PE1) of the PPGC1 register for the PPG1 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 any external pin.

In 8-bit prescaler + 8-bit PPG mode, the 8-bit prescaler toggle output waveform is output from PPG0, while the 8-bit PPG waveform is output from PPG1. The figure below is a diagram of output waveforms in this mode.



Pl0 = T x (L0+1)
Ph0 = T x (L0+1)
Pl1 = T x (L0+1) x (Ll+1)
Ph1 = T x (L0+1) x (Hl+1)

L0 : ch0 PRLL value and ch0 PRLH value
L1 : ch1 PRLL value
H1 : ch1 PRLH value
T : Input clock cycle
Ph0 : PPG0 high pulse width
Pl0 : PPG0 low pulse width
Ph1 : PPG1 high pulse width
Pl1 : PPG1 low pulse width

Note : Set the same value in ch0 PRLL and ch0 PRLH.

Figure 14.4b 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 causes for PUF0 and PUF1.

(7) Default values of hardware components

The hardware components of this block are initialized to the following values when reset:

<Registers>
- PPGC0 ⇨ $0X000001_B$
- PPGC1 ⇨ 00000001B
- PPGOE XXXXXX00B

<Pulse outputs>
PPG0 ⇨ 'L'
PPG1 ⇨ 'L'
PE0 ⇨ PPG0 output disabled
PE1 ⇨ PPG1 output disabled

<Interrupt requests>
IRQ0 ⇨ 'L'
IRQ1 ⇨ 'L'

Hardware components other than the above are not initialized.

(8) 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 PRLL and PRLH. If two byte transfer instructions are used to write a data item to these registers, a pulse of unexpected width may be output depending on the timing.

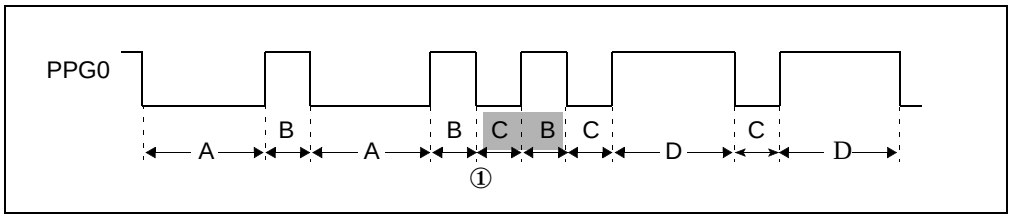


Figure 14.4c Write timing chart

Assume that PRLL is updated from A to C before point ① in the time chart above, and PRLH is updated from B to D after point ①. Since the PRL values at point ① are PRLL=C and PRLH=B, a pulse of L side count value C and H side count value B is output only once.

Similarly, to write data in PRL of ch0 and ch1 in 16-bit PPG mode, use a long word transfer instruction, or use word transfer instructions in the order of ch0 and then ch1. In this mode, the data is only temporarily written to ch0 PRL. Then, the data is actually written into ch0 PRL when the ch1 PRL is written to.

In a mode other than 16-bit PPG mode, ch0 and ch1 PRL are written independently.

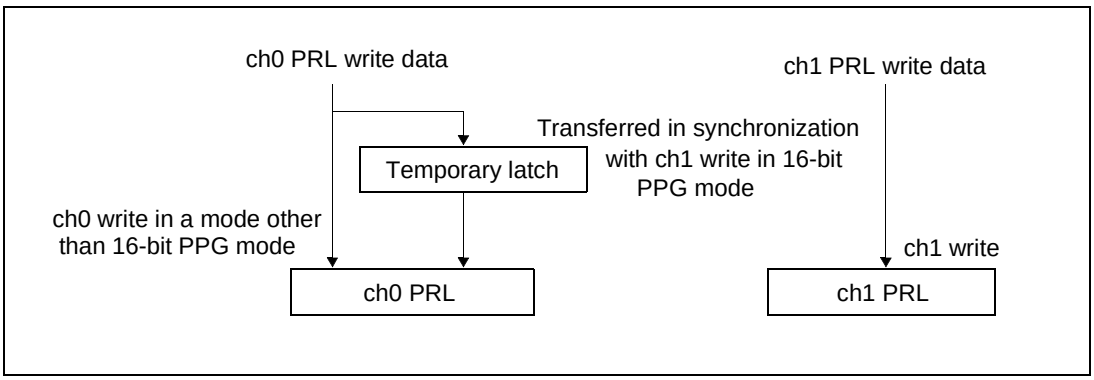


Figure 14.4d PRL write operation block diagram

Chapter 15:
16-Bit Reload Timer (with Event Count Function)

15.1 Outline

The 16-bit reload timer 1 consists of a 16-bit down-counter, a 16-bit reload register, one input pin (TIN) and one output pin (TOUT), and a control register.

It has an internal clock mode for counting down in synchronization to three types of internal clocks and an event count mode for counting down detecting a given edge of the pulse input to the external bus pin, and either of the two functions can be selectively used.

For this timer, an “underflow” is defined as the timing of transition from the counter value of “0000_H” to “$FFFF_H$”. According to this definition, an underflow occurs after [re-load register setting value + 1] counts.

In operating the counter, the re-load mode for repeating counting operation after re-loading a counter value after an underflow or the one-shot mode for stopping the counting operation after an underflow can be selectively used.

Because the timer can generate an interrupt upon an underflow, the timer conforms to the extended intelligent I/O service (EI^2OS).

The output pin (TOUT) outputs a toggle output waveform in reload mode or a square waveform during counting in one-shot mode. The input pin (TIN) functions as the event input in event count mode, or as the trigger input or gate input in internal clock mode.

15.2 Block Diagram

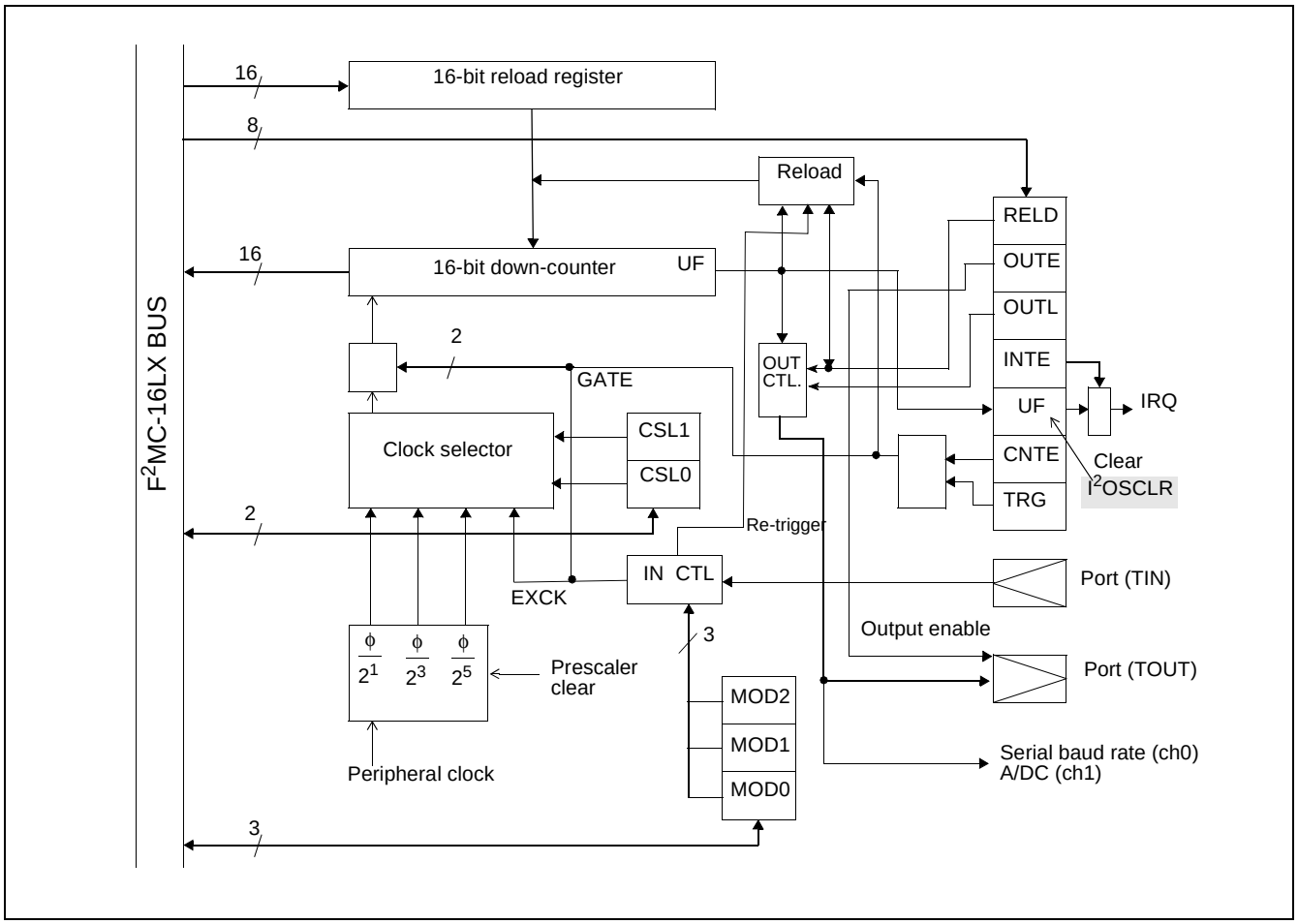


Figure 15.2a Block Diagram of 16-Bit Reload Timer

15.3 Registers and Register Details

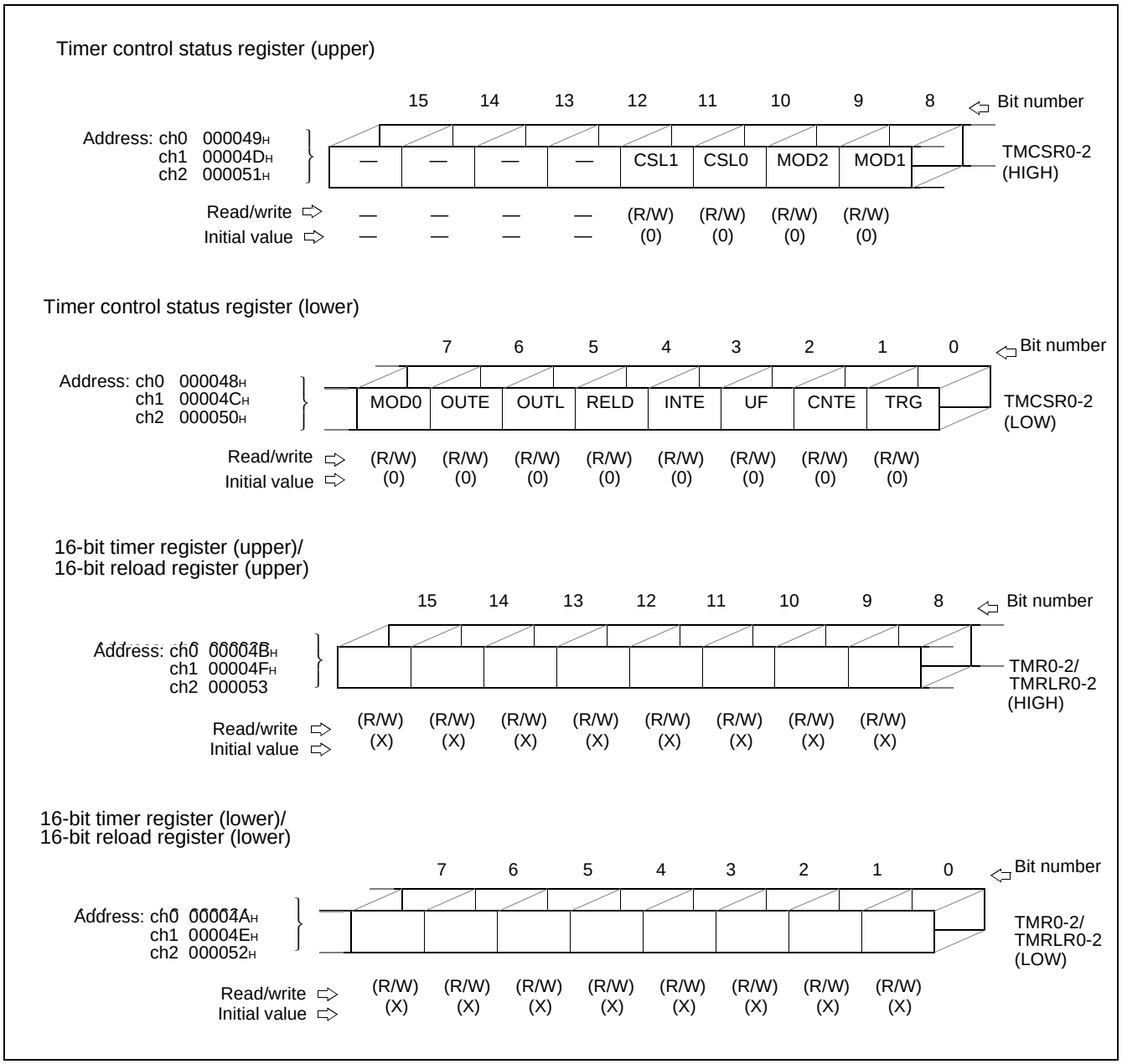


Figure 15.3a Registers of 16-Bit Reload Timer

15.3.1 Timer control status register (TMCSR)

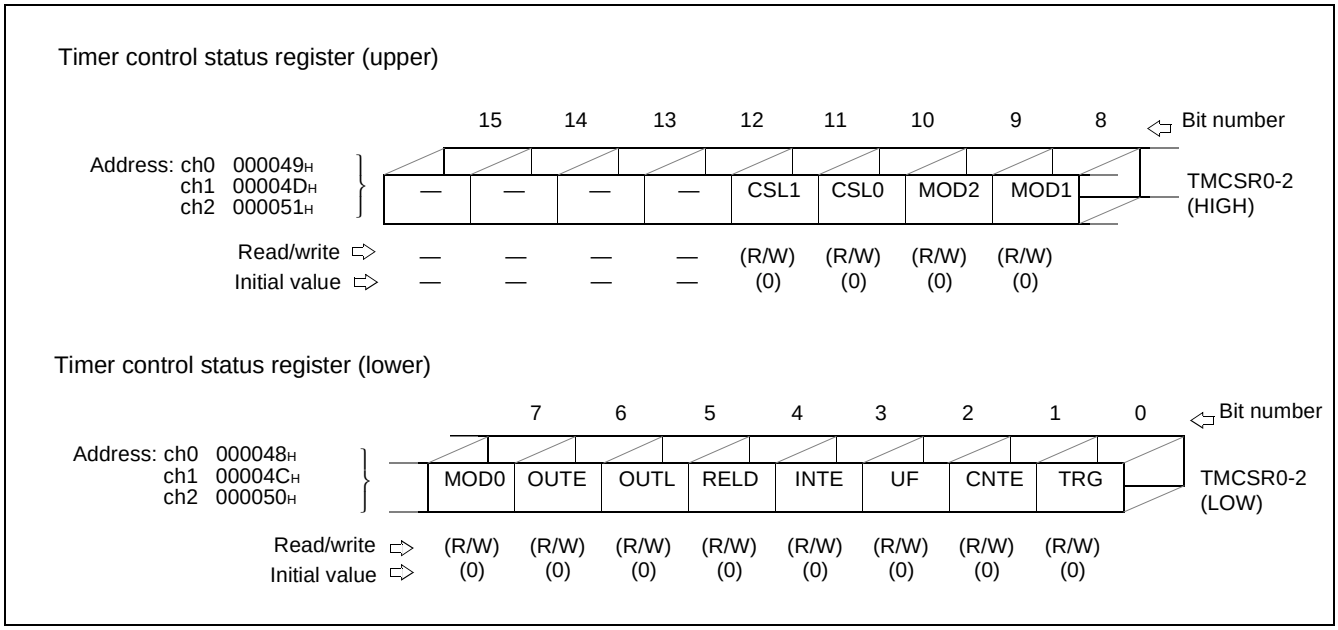


Figure 15.3.1a Timer Control Status Register

Controls the operation mode and interrupts for the 16-bit timer. Only modify bits other than UF, CNTE, and TRG when CNTE = “0”.

[Bits 11, 10] CSL1, CSL0 (Clock select 1, 0)

The count clock select bits. The following tMable lists the selected clock sources.LL

CSL1	CSL0	Clock Source (Machine cycle ϕ = 16 MHz)
0	0	$\phi/2^1$ (0.125 µs)
0	1	$\phi/2^3$ (0.5 µs)
1	0	$\phi/2^5$ (2.0 µs)
1	1	External event count mode

[Bits 9, 8, 7] MOD2, MOD1, MOD0

These bits set the operation mode and I/O pin functions.

The MOD2 bit selects the I/O functions. When MOD2 = “0”, the input pin functions as a trigger input. In this case, the reload register contents is loaded to the counter when an active edge is input to the input pin and count operation proceeds. When MOD2 = “1”, the timer operates in gate counter mode and the input pin functions as a gate input. In this mode, the counter only counts while an active level is input to the input pin.

The MOD1 and 0 bits set the pin functions for each mode. The following tables list the MOD2, 1, 0 bit settings

Internal clock mode (CSL0, 1 = “00”, “01”, or “10”)

MOD2	MOD1	MOD0	Input Pin Function	Active Edge or Level
0	0	0	Trigger disabled	—
0	0	1	Trigger input	Rising edge
0	1	0	⇑	Falling edge
0	1	1	⇑	Both edges
1	×	0	Gate input	“L” level
1	×	1	⇑	“H” level

Event counter mode (CSL0,1 = “11”)

MOD2	MOD1	MOD0	Input Pin Function	Active Edge or Level
X	0	0	—	—
	0	1	Trigger input	Rising edge
	1	0	⇑	Falling edge
	1	1	⇑	Both edges

Note: Bits marked as X in the table can be set to any value.

[Bit 6] OUTE

Output enable bit. The TOUT pin functions as a general-purpose port when this bit is “0” and as the timer output pin when this bit is “1”. In reload mode, the output waveform toggles. In one-shot mode, TOUT outputs a square waveform that indicates that counting is in progress.

Note: For reload timer 1 and 2, TOUT is multiplexed with P94/OUT0 and P95/OUT1 respectively. If output capture is enabled, it has higher priority than reload timer output.

[Bit 5] OUTL

This bit sets the output level for the TOUT pin. When OUTL is “0” or “1”, the output pin level is opposite

[Bit 4] RELD (Reload)

This bit enables reload operations. When RELD is “1”, the timer operates in reload mode. In this mode, the timer loads the reload register contents into the counter and continues counting whenever an underflow occurs (when the counter value changes from 0000_H to $FFFF_H$). When RELD is “0”, the timer operates in one-shot mode. In this mode, the count operation stops when an underflow occurs due to the counter value changing from 0000_H to $FFFF_H$..

OUTE	RELD	OUTL	Output Waveform
0	X	X	General-purpose port
1	0	0	Output an “H” level square waveform during counting.
1	0	1	Output an “L” level square waveform during counting.
1	1	0	Toggle output. “L” level at count start.
1	1	1	Toggle output. “H” level at count start.

[Bit 3] INTE (Interrupt enable)

Timer interrupt request enable bit. When INTE is “1”, an interrupt request is generated when the UF bit changes to “1”. When INTE is “0”, no interrupt request is generated, even when the UF bit changes to “1”.

[Bit 2] UF (Underflow)

Timer interrupt request flag. UF is set to “1” when an underflow occurs (when the counter value changes from 0000_H to $FFFF_H$). Cleared by writing “0” or by the intelligent I/O service. Writing “1” to this bit has no meaning. Read as “1” by read-modify-write instructions.

[Bit 1] CNTE (Count enable)

Timer count enable bit. Writing “1” to CNTE sets the timer to wait for a trigger. Writing “0” stops count operation.

[Bit 0] TRG (Trigger)

Software trigger bit. Writing “1” to TRG applies a software trigger, causing the timer to load the reload register contents to the counter and start counting. Writing “0” has no meaning. Reading always returns “0”. Applying a trigger using this register is only valid when CNTE = “1”. Writing “1” has no effect if CNTE = “0”.

15.3.2 TMR (16-bit timer register)/TMRLR (16-bit reload register)

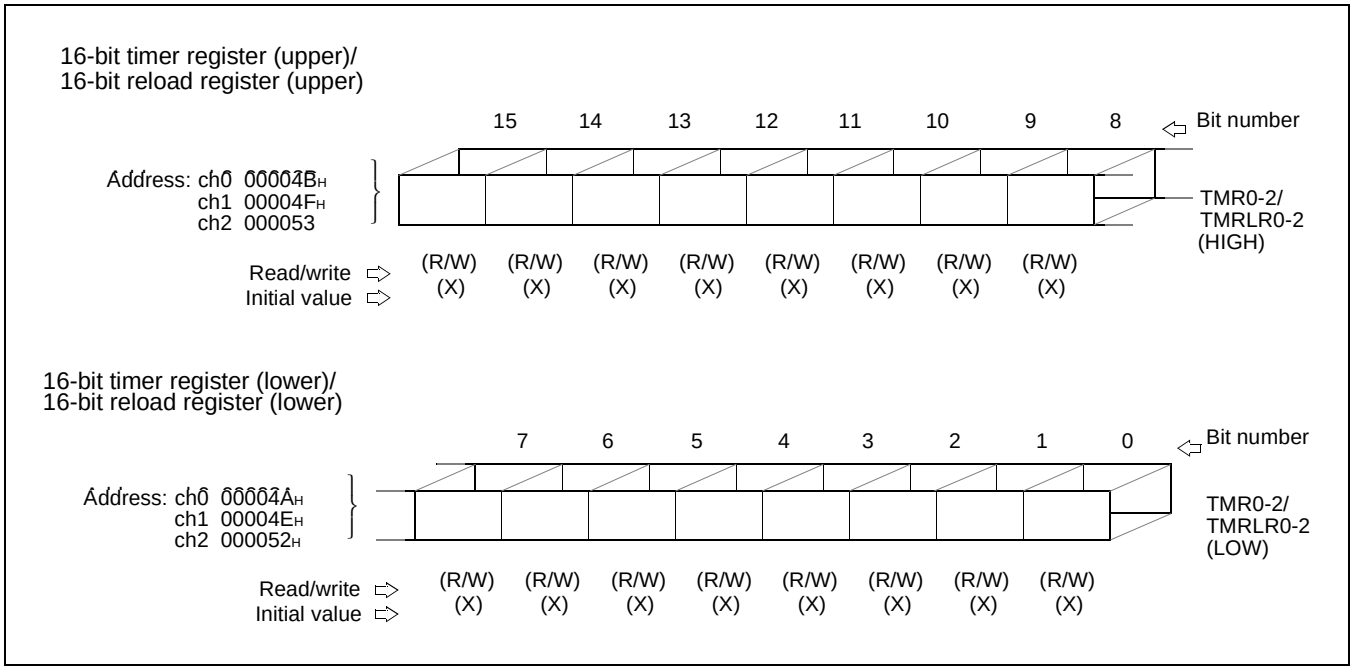


Figure 15.3.2a 16-Bit Timer Register and 16-Bit Reload Register

■ TMR contents (for reading)

Reading this register reads the count value of the 16-bit timer. The initial value is undefined. Always read this register using word move instructions.

■ TMRLR contents (for writing)

The 16-bit reload register holds the initial count value. The initial value is undefined. Always write to this register using word transfer instructions.

15.4 Operation

15.4.1 Internal clock operation

The machine clock divided by 2^1, 2^3, or 2^5 can be selected as the clock sources for operating the timer from an internal divide clock. The external input pin can be selected as either a trigger input or gate input by a register setting.

Writing “1” to both the CNTE and TRG bits in the control register enables and starts counting simultaneously. Using the TRG bit as a trigger input is always available when the timer is enabled (CNTE = “1”), regardless of the operation mode.

Figure 15.4.1a shows counter activation and counter operation. A time period T (T: machine cycle) is required from the counter start trigger being input until the reload register data is loaded into counter.

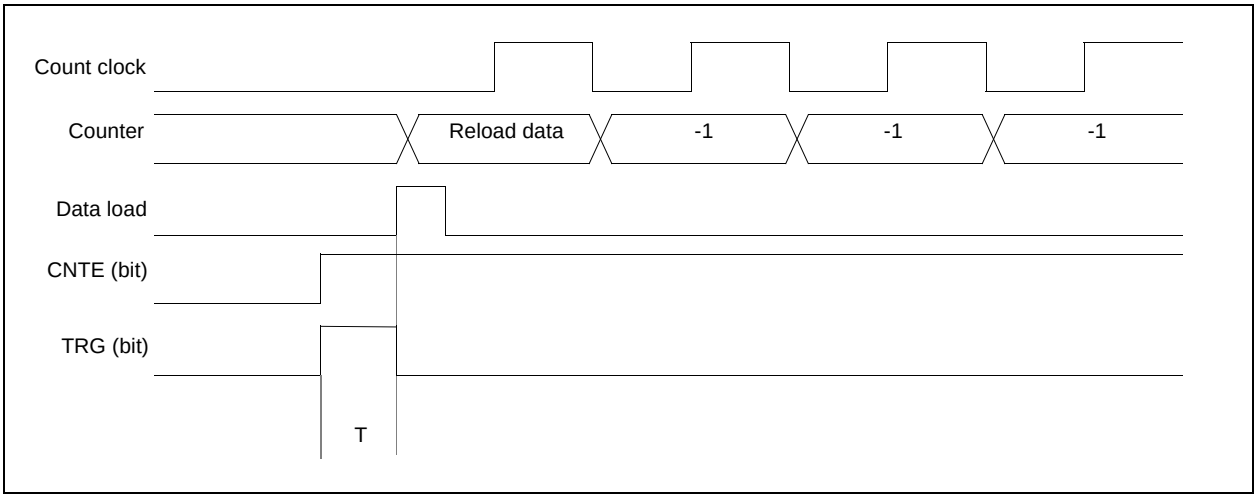


Figure 15.4.1a Counter Activation and Operation

15.4.2 Underflow operation

An underflow is defined for this timer as the time when the counter value changes from 0000_H to $FFFF_H$. Therefore, an underflow occurs after (reload register setting + 1) counts.

If the RELD bit in the control register is “1” when the underflow occurs, the contents of the reload register is loaded into the counter and counting continues. When RELD is “0”, counting stops with the counter at $FFFF_H$.

The UF bit in the control register is set when the underflow occurs. If the INTE bit is “1” at this time, an interrupt request is generated.

Figure 15.4.2a shows the operation when an underflow occurs.

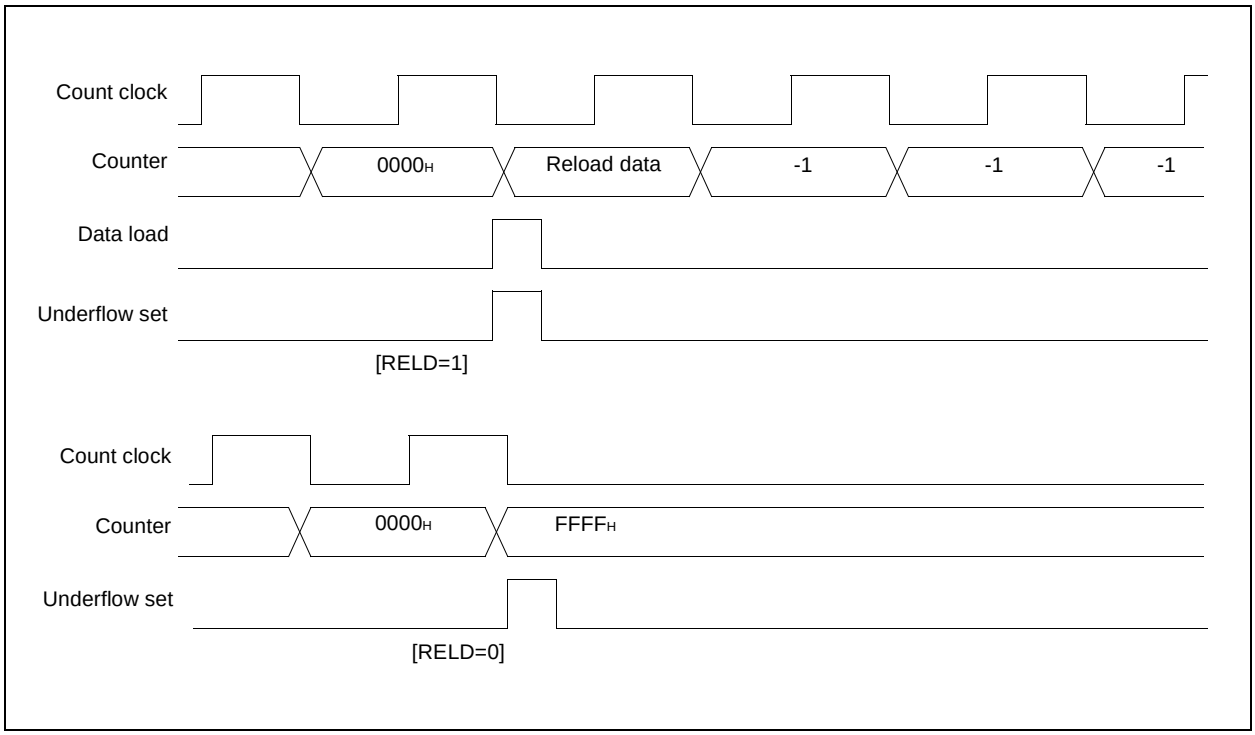


Figure 15.4.2a Underflow Operation

15.4.3 Input pin functions (for internal clock mode)

The TIN pin can be used as either a trigger input or a gate input when an internal clock is selected as the clock source. When used as a trigger input, input of an active edge causes the timer to load the reload register contents to the counter and then start count operation after clearing the internal prescaler. Input a pulse width of at least 2T (T is the machine cycle) to TIN.

Figure 15.4.3a shows the operation of trigger input.

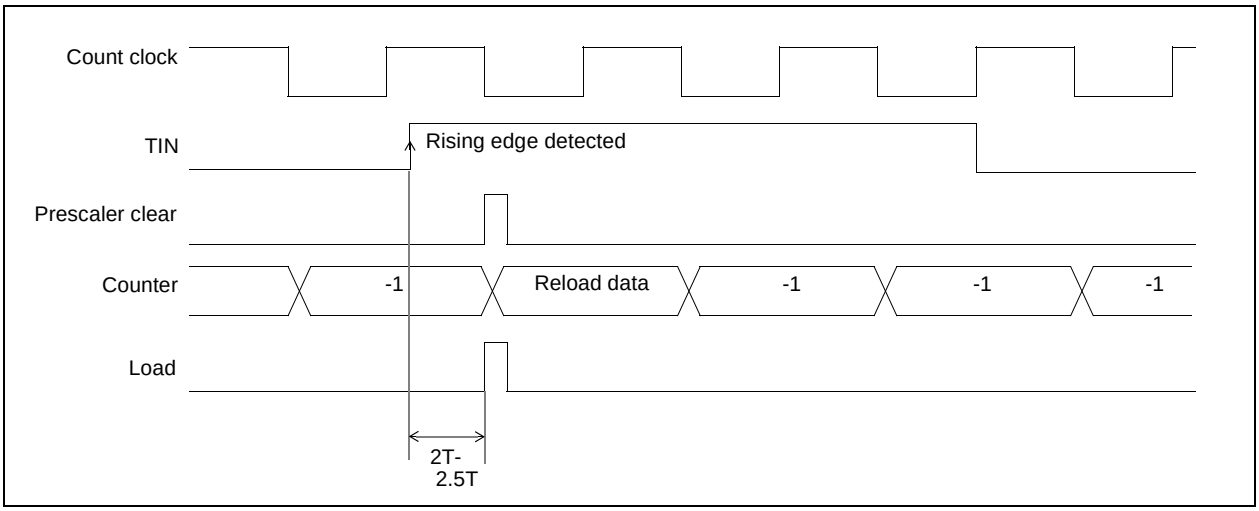


Figure 15.4.3a Trigger Input Operation

When used as a gate input, the counter only counts while the active level specified by the MOD0 bit of the control register is input to the TIN pin. In this case, the count clock continues to operate unless stopped. The software trigger can be used in gate mode, regardless of the gate level. Input a pulse width of at least 2T (T is the machine cycle) to the TIN pin. Figure 15.4.3b shows the operation of gate input.

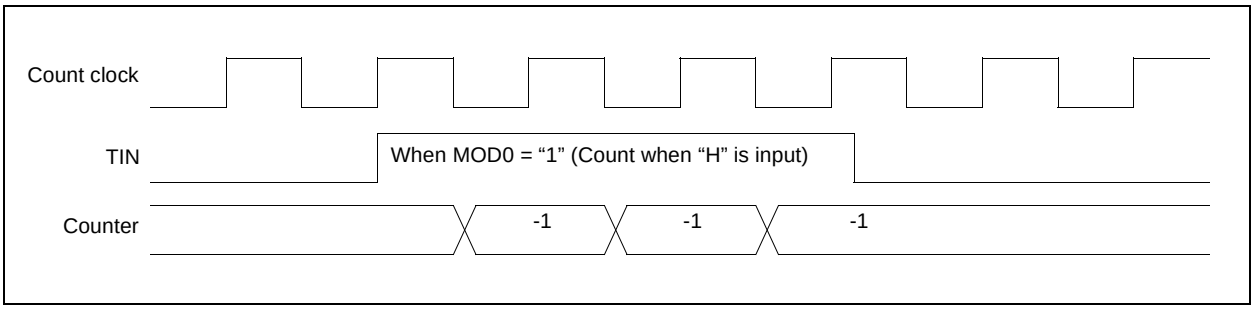


Figure 15.4.3b Gate Input Operation

15.4.4 External event counter

The TIN pin functions as an external event input pin when an external clock is selected. The counter counts on the active edge specified in the register. Input a pulse width of at least 4T (T is the machine cycle) to the TIN pin.

15.4.5 Output pin functions

In reload mode, the TOUT pin performs toggle output (inverts at each underflow). In one-shot mode, the TOUT pin functions as a pulse output that outputs a particular level while the count is in progress. The OUTL bit of the control register sets the output polarity. When OUTL = “0”, the initial value for toggle output is “0” and the one-shot pulse output is “1” while the count is in progress. The output waveforms are opposite when OUTL = “1”.

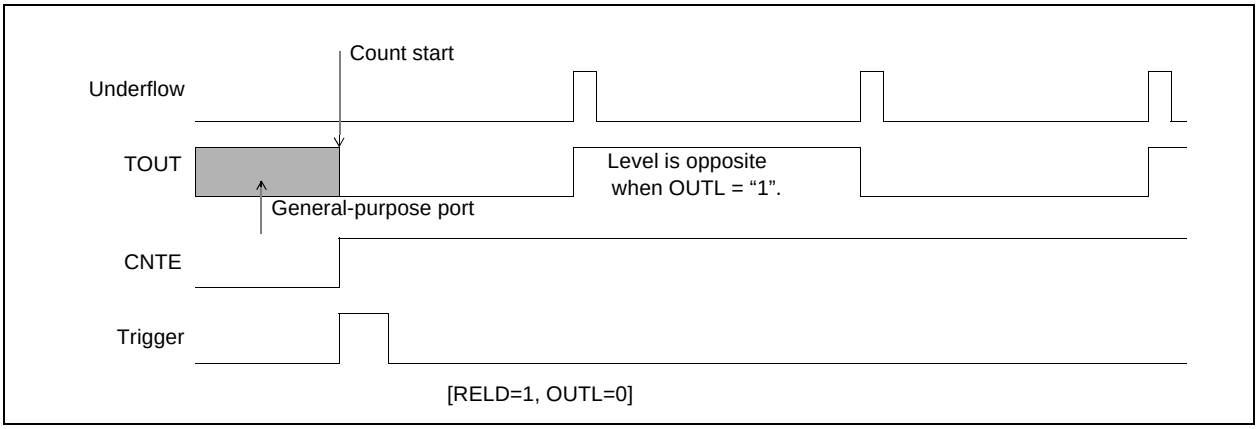


Figure 15.4.5a Output Pin Functions (1)

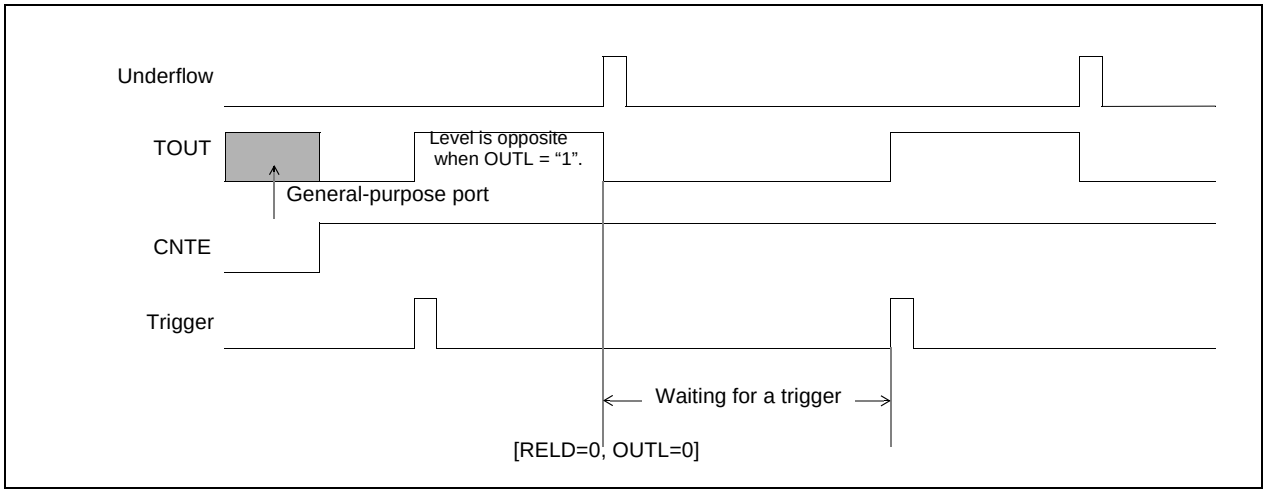


Figure 15.4.5b Output Pin Functions (2)

15.4.6 Intelligent I/O service (I^2OS) function and interrupts

The timer includes a circuit that supports I^2OS. The timer can activate I^2OS when an underflow occurs. I^2OS can be used with both timers on this product. However, as both timers (ch0 and ch1) are connected to the same interrupt control register (ICRx) in the interrupt controller, ch0 and ch1 cannot be assigned to different I^2OS services. Also, as the two timers have different interrupt vectors, they can be assigned to two different interrupt services. However, as ch0 and ch1 share an interrupt control register as described above, the same interrupt level applies to both channels.

15.4.7 Counter operation state

The counter state is determined by the CNTE bit in the control register and the internal WAIT signal. Available states are: CNTE = “0” and WAIT = “1” (STOP state), CNTE = “1” and WAIT = “1” (WAIT state for trigger), and CNTE = “1” and WAIT = “0” (RUN state). Figure 15.4.7a shows the transitions between each state.

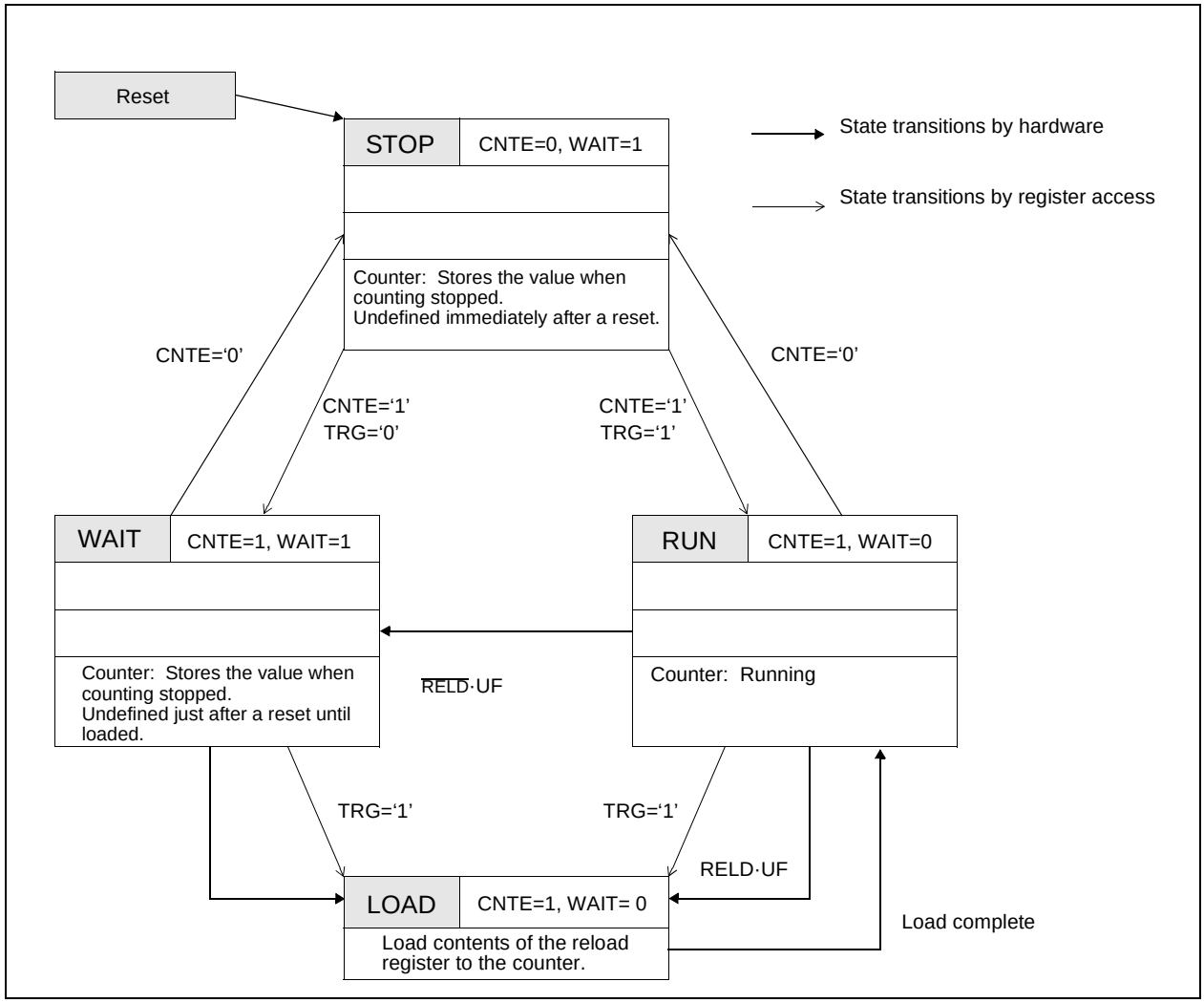


Figure 15.4.7a Counter State Transitions

Chapter 16:
A/D Converter

16.1 Outline

The A/D converter converts analog input voltages into digital values. The A/D converter has the following features:

- Conversion time: 5.2 μs min. per channel (at 16 MHz machine clock)
- RC sequential compare conversion format with sample and hold circuit
- 10-bit resolution
- Analog input selected from eight channels by programming

  Single conversion mode: One channel is selected for conversion.

  Scan conversion mode: Voltages in multiple consecutive channels are converted. Up to eight channels can be programmed.

  Continuous conversion mode: Voltages in the specified channel are converted repeatedly.

  Stop conversion mode: Voltages in a single channel are converted, then the system pauses and stands by for the next activation. (The conversion start points can be synchronized.)
- At the end of A/D conversion, a relevant interrupt request can be issued to the CPU. This interrupt can be used to activate I2OS, which transfers A/D conversion result data to memory. This feature is suitable for continuous processing.
- The activation factors can be selected from software, external trigger (falling edge), or timer (rising edge).

16.2 Block Diagram

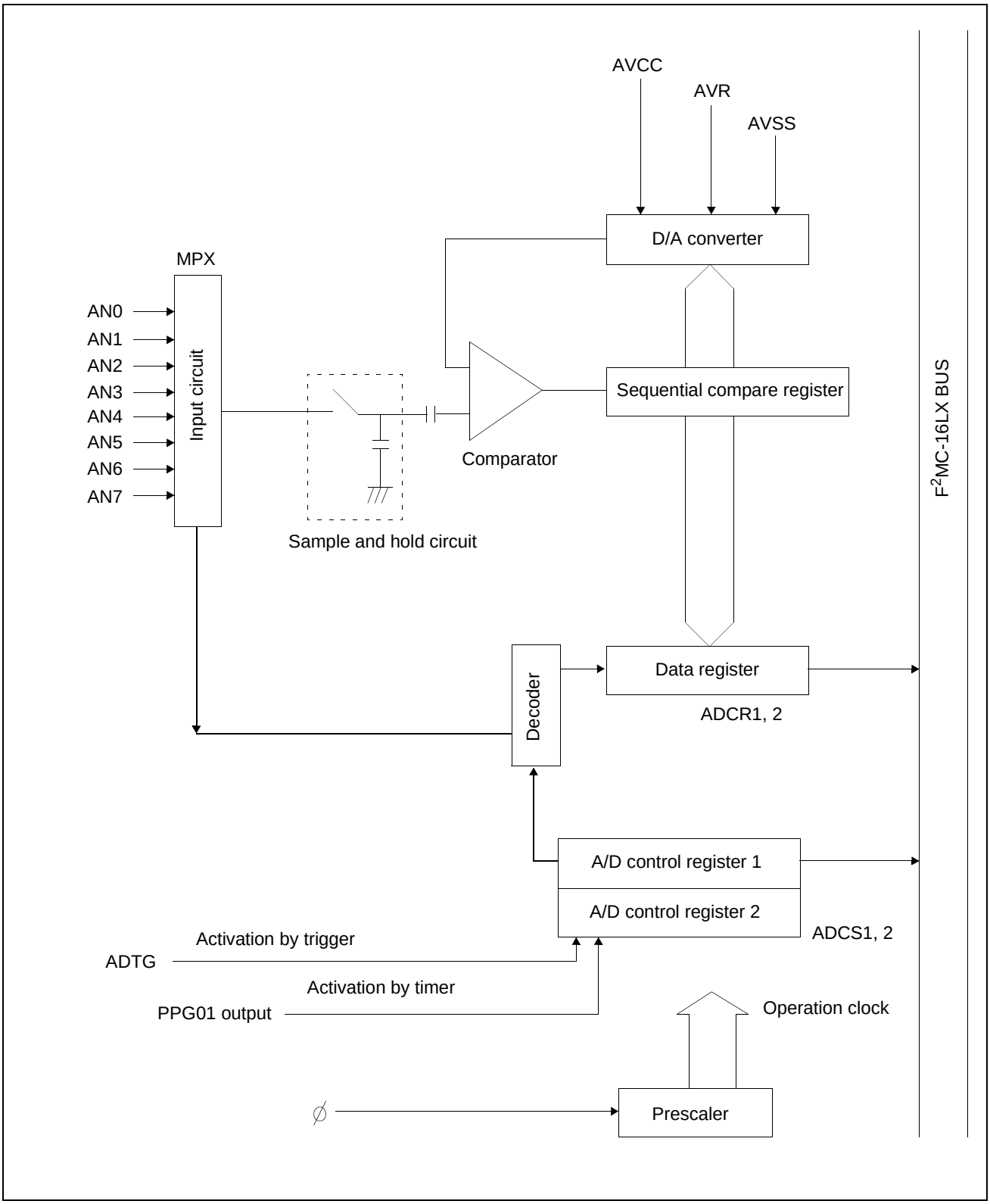


Figure 16.2a Block Diagram of A/D converter

16.3 Registers and Register Details

Control Status Registers (Upper Byte)

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 000037_H	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	DA	ADCS2
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Control Status Registers (Lower Byte)

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 000036_H	MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	ADCS1
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)	

Data Registers (Upper Byte)

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 000039_H	Reserved	ST1	ST0	CT1	CT0	——	D9	D8	ADCR2
Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(–)	(R)	(R)	
Initial value ⇨	(0)	(0)	(0)	(0)	(1)	(–)	(X)	(X)	

Data Registers (Lower Byte)

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 000038_H	D7	D6	D5	D4	D3	D2	D1	D0	ADCR1
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

Figure 16.3a Registers of A/D Converter

16.3.1 Control status registers (ADCS1 and ADCS2)

These registers are used to control the A/D converter and display the status.

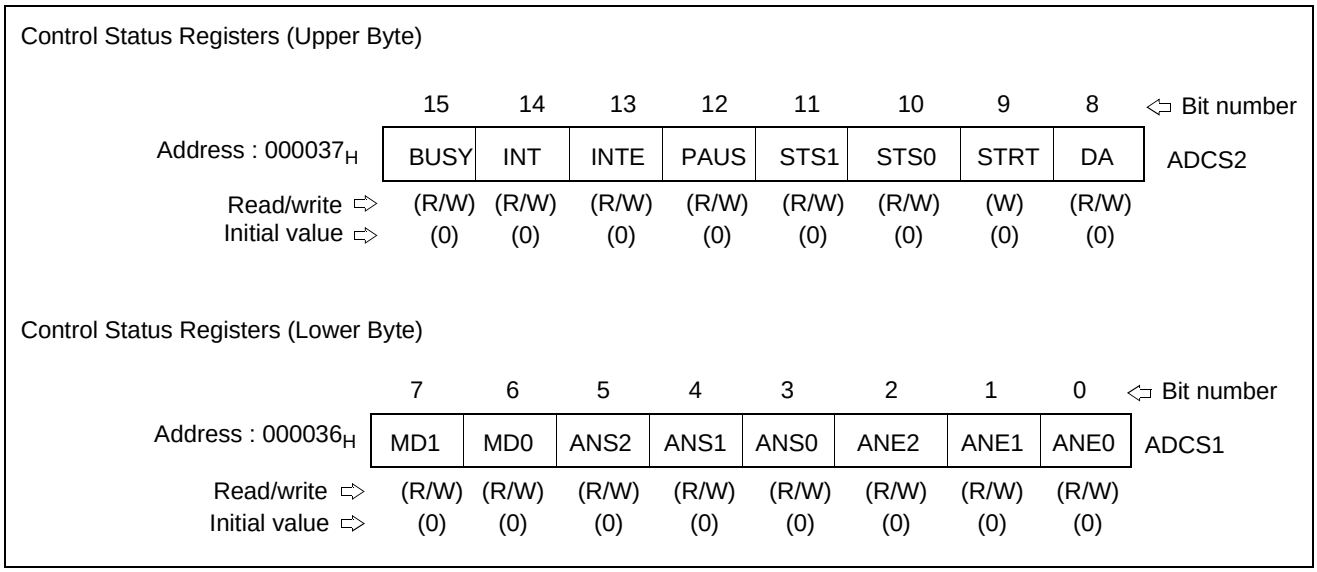


Figure 16.3.1a Control Status Registers

Note: Do not update ADCS1 during A/D conversion.

[bit 15] BUSY (busy flag and stop):

Read: This bit indicates the A/D converter operation. This bit is set when the A/D conversion is activated, and cleared when the conversion ends.
Write: Writing ’0’ to this bit during A/D conversion forces the conversion to terminate. This features is used for forced stop in continuous or stop mode.
’1’ cannot be written to the operation display bit. With a read-modify-write instruction.
’1’ is read from this bit. In single mode, this bit is cleared at the end of A/D conversion.
In continuous or stop mode, this bit is not cleared until conversion is stopped by writing ’0.’
This bit is initialized to ’0’ upon a reset.
Do not perform forced termination and activation by software simultaneously. (BUSY=0, STRT=1)

[bit 14] INT (Interrupt): A data display bit

This bit is set when conversion data is written to ADCR.
An interrupt request is issued if this bit is set while bit 5 (INTE) is ’1.’ In addition, I2OS is activated if it is enabled. Writing ’1’ has no effect.
This bit is cleared by writing ’0’ or by an I2OS interrupt clear signal.
Note: To clear this bit by writing ’0,’ ensure that A/D conversion is not in progress.
This bit initialized to ’0’ upon a reset.

[bit 13] INTE (Interrupt enable): This bit is used to enable or disable interrupts at the end of conversion..

0	Interrupts are disabled. [initial value]
1	Interrupts are enabled.

Set this bit when using I2OS. I2OS is activated when an interrupt request is issued.
Upon a reset, this bit is initialized to ’0.’

[bit 12] PAUS (A/D conversion pause):

This bit is set when the A/D conversion is paused.
Only one register is available for storing the A/D conversion result. Therefore, unless the conversion results are transferred by I2OS, the result data would be continuously updated and destroyed in continuous conversion.
To prevent the above condition, the system is designed so that a data register value must be transferred by I2OS before the next conversion data is saved. A/D conversion pauses during that period. A/D conversion is resumed at the end of transfer by I2OS.
This register is valid only when I2OS is used.
* For the conversion data protection function, see Section 2.7.4, "Operations."
Upon a reset, this bit is initialized to '0.'

[bits 11 and 10] STS1 and STS0 (Start source select):

Upon a reset, these bits are initialized to '00.'
These bits are used to select the A/D conversion activation factor.

STS1	STS0	Function
0	0	Activation by software
0	1	Activation by external pin trigger and software
1	0	Activation by timer and software
1	1	Activation by external pin trigger, timer, and software

In a mode allowing two or more activation factors, A/D conversion is activated by the factor that is input first.
The activation factor changes as soon as it is updated. Thus, take care when updating it during A/D conversion.

* The external pin trigger is detected by the falling edge. If this bit is updated to external trigger activation while the external trigger input level is 'L,' A/D may be activated at once.
* When timer is selected, PPG1 output is selected.

[bit 9] STRT (Start):

A/D conversion is activated when '1' is written to this bit.
To reactivate A/D conversion, write '1' to this bit again.
In stop mode, restart is disabled due to the operation functions.
Upon a reset, this bit is initialized to '0.'

Note: Do not perform forced termination and activation by software simultaneously. (BUSY=0, STRT=1)

[bit 8] DA

This is a test bit. Always write '0' to this bit.

[bits 7 and 6] MD1 and MD0 (A/D converter mode set):

These bits are used to set the A/D converter operation mode.

MD1	MD0	Operation mode
0	0	Single mode. Reactivation during operation is allowed.
0	1	Single mode. Reactivation during operation is not allowed.
1	0	Continuous mode. Reactivation during operation is not allowed.
1	1	Stop mode. Reactivation during operation is not allowed.

Single mode: A/D conversion is continuously performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0. The conversion stops once it has been done for all these channels.

Continuous mode: A/D conversion is repeatedly performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0.

Stop mode: A/D conversion is performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0, pausing for each channel. The A/D conversion is resumed upon an activation factor.

Upon a reset, these bits are initialized to '00.'

Note: When activated in continuous or stop mode, A/D conversion continues until it is stopped by the BUSY bit.

Note: The conversion is stopped by writing '0' to the BUSY bit.

Note: In single, continuous, or stop mode, reactivation is disabled regardless of the activation factor (timer, external trigger, or software).

[bits 5, 4, and 3] ANS2, ANS1, and ANS0 (Analog start channel set):

Use these bits to specify the start channel for A/D conversion.
When the A/D converter is activated, A/D conversion starts from the channel selected with these bits.

ANS2	ANS1	ANS0	Start channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

* Read

During A/D conversion, the current conversion channel is read from these bits. If the system is stopped in stop mode, the previous conversion channel is read.

* Upon a reset, these bits are initialized to '000.'

[bits 2, 1, and 0] ANE2, ANE1, and ANE0 (Analog end channel set):

Use these bits to set the A/D conversion end channel.

ANE2	ANE1	ANE0	End channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

* When the same channel is written to ANE2 to ANE0 and ANS2 to ANS0, conversion is performed for one channel only (single conversion).
* In continuous or stop mode, operation returns to the start channel specified in ANS2 to ANS0 after the conversion is completed for the channel specified in ANE2 to ANE0.
* If the ANS value is smaller than the ANE value, conversion starts from the ANS channel. Then, once conversion is complete up to channel 7, operation returns to channel 0 and conversion is performed up to the ANE channel.
* Upon a reset, these bits are initialized to ’000.’

Example: ANS=6, ANE=3, single mode
Conversion is performed in the following sequence: CH6, CH7, CH0, CH1, CH2, CH3

16.3.2 ADCR1 and ADCR0 (Data registers)

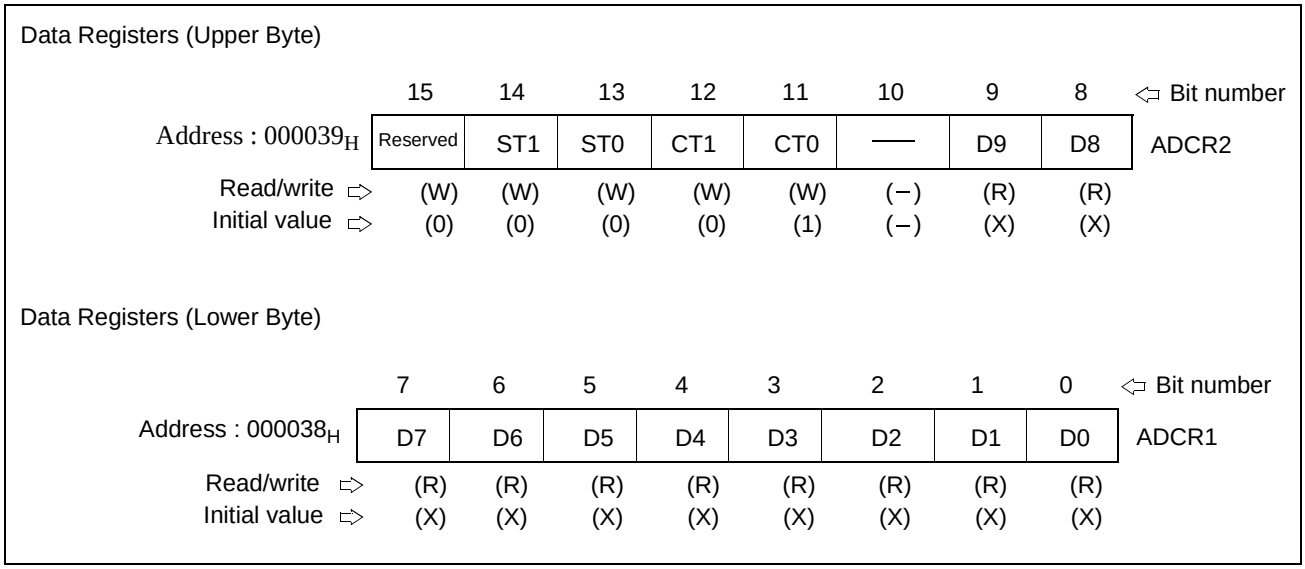


Figure 16.3.2a Data Registers

[bit 15]

This is reserved bit. This bit should be written to '1' before AD conversion. Never write '0' to this bit.

Note: Reading this bit always returns "1".

[bit 14, 13] : ST1, ST0 (Sampling Time)

These bits is used for setting the sampling time in terms of machine cycle.

ST1	ST0	Sampling time machine cycle	Sampling time
0	0	64 machine cycle	4ms at 16MHz machine clock
0	1	Reserved	
1	0	Reserved	
1	1	4096 machine cycle	256ms at 16MHz machine clock

Note: Reading these bits always return "1".

[bit 12, 11] : CT1, CT0 (Compare Time)

These bits is used for setting the comparsion time in terms of machine cycle.

CT1	CT0	Comparsion time machine cycle	Comparsion time
0	0	176 machine cycle	22ms at 8MHz machine clock
0	1	352 machine cycle	22ms at 16MHz machine clock
1	0	Reserved	
1	1	Reserved	

Note1: When the bits is set to ‘00’, the machine clock should not be higher than 8MHz.

Note2: Reading these bits always return “1”.

[bit 9 to bit 0] : D9 to D0 (ADCR1:1,0 and ADCR0)

ADCR1:1,0 and ADCR0 stores the AD conversion result. These register values are updated each time conversion is completed. Usually, the final conversion value is stored in these bits.
Upon a reset, these registers are undefined.
The conversion data protection function is available. See Section 2.7.4, "Operations."

Note: Ensure that no data is written to these registers during A/D conversion.

16.4 Operations

The A/D converter operates in the sequential compare format, and has a 8-bit resolution.

Since the A/D converter has only one register (8 bits) for storing the conversion result, the conversion data registers (ADCR0) are updated each time conversion is completed. Thus, the A/D converter must not be used alone for continuous conversion. Use the F^2MC-16 intelligent I/O service function to transfer converted data to memory while conversion is in progress.

The operation modes are explained below.

(1) Single mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits. The converter stops operation after the conversion is completed for the end channel specified with the ANE bits. If the start and end channels are the same (ANS=ANE), conversion is performed only for one channel.
Example:

ANS = 0 0 0 , ANE = 0 1 1

Start → AN0 → AN1 → AN2 → AN3 → End

ANS = 0 1 0 , ANE = 0 1 0

Start → AN2 → End

(2) Continuous mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits. After the conversion is completed for the end channel specified with the ANE bits, conversion is repeated from the analog inputs of the ANS. If the start and end channels are the same (ANS=ANE), conversion for one channel is repeated.
Example:

ANS = 0 0 0 , ANE = 0 1 1

Start → AN0 → AN1 → AN2 → AN3 → AN0 ·····→ Repeat

ANS = 0 1 0 , ANE = 0 1 0

Start → AN2 → AN2 → AN2 ·····→ Repeat

In continuous mode, conversion is repeated until ’0’ is written to the BUSY bit. (Writing ’0’ to the BUSY bit forces the operation to end.) If the operation is terminated forcibly, conversion stops before conversion is completed. (Upon a forced termination, the conversion register stores the previous data that has been converted completely.)

(3) Stop mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits, pausing each time conversion for one channel is completed. To release pausing, activate the A/D converter again.
After the conversion is completed for the end channel specified with the ANE bits, conversion is repeated from the analog inputs of the ANS. If the start and end channels are the same (ANS=ANE), conversion is performed only for one channel.
Example:

ANS = 0 0 0 , ANE = 0 1 1

Start → AN0 → End → Restart → AN1 → End → Restartë → AN2 → End → Restart →
→ AN3 → End → Restart → AN0 ·····→ Repeat

ANS = 0 1 0 , ANE = 0 1 0

Start → AN2 → End → Restart → AN2 → End → Restartë → AN2 ·····→ Repeat

Only the activation factors specifies with STS1 and STS0 are used.
In this mode, start of conversion can be synchronized.

(4) Conversion using I^2OS

Sample flow from A/D conversion activation to transfer of converted data (continuous mode)

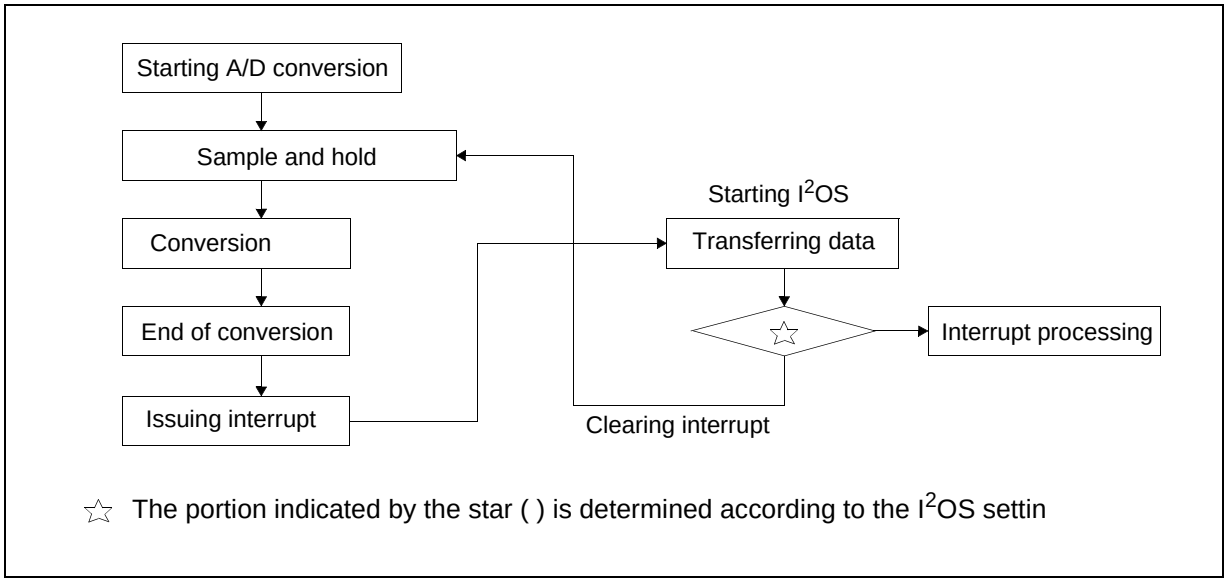


Figure 16.4a Flow chart of A/D Conversion

Usage

- Starting I^{2}OS in single mode
  - •To terminate conversion after analog inputs AN1 to AN3 are converted
  - •To transfer conversion data sequentially to addresses 200H to 206H
  - •To start conversion by software
  - •To use the highest interrupt level

I^{2}OS setting

```
        MOV   ICR3   #08H   ..... ①
        MOV   BAPL,  #00H   ..... ②
        MOV   BAPM,  #02H   ..... ③
        MOV   BAPH,  #00H   ..... ④
        MOV   ISCS,  #18H   ..... ⑤
        MOV   IOA,   #38H   ..... ⑥
        MOV   DCT,   #03H   ..... ⑦
```

A/D converter setting

```
        MOV   ADCS1  #0BH   ..... ⑧
        MOV   ADCS2  #A2H   ..... ⑨
```

Interrupt sequence

```
        RETI                ..... ⑩
```

① Specifies the highest interrupt level, I2OS activation upon an interrupt, and the descriptor address.

②③④ Specifies the transfer destination address of converted data.

⑤ Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is terminated in response to a request from a resource.

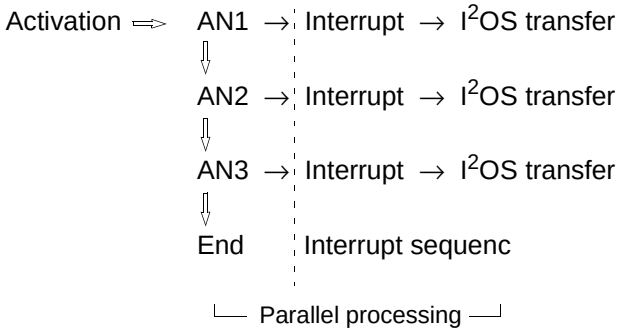
⑦ I^{2}OS transfer is performed three times. This count is the same as the conversion count.

⑧ Specifies single mode, start channel AN1, and end channel AN3.

⑨ Specifies activation by software and start of A/D conversion.

⑩ Specifies return from an interrupt.

ICR3 : Interrupt control register
BAPL : Buffer address pointer, low-order
BAPM : Buffer address pointer, medium-order
BAPH : Buffer address pointer, high-order
ISCS : I^{2}OS status register
I/OA : I/O address counter
DCT : Data counter



Usage

- Starting I^2OS in continuous mode
  - •To convert analog inputs AN3 to AN5 and obtain two conversion data items for each channel
  - •To transfer conversion data sequentially to addresses 600H to 60CH
  - •To start conversion by external edge input
  - To use the highest interrupt level

I^2OS setting

```
        MOV   ICR3    #08H   ..... ①
        MOV   BAPL,   #00H   ..... ②
        MOV   BAPM,   #06H   ..... ③
        MOV   BAPH,   #00H   ..... ④
        MOV   ISCS,   #08H   ..... ⑤
        MOV   I / OA, #38H   ..... ⑥
        MOV   DCT,    #06H   ..... ⑦
A/D converter setting
        MOV   ADCS1   #9DH   ..... ⑧
        MOV   ADCS2   #A4H   ..... ⑨
Interrupt sequence
        MOV   ADCS2   #00H   ..... ⑩
        RET
```

① Specifies the highest interrupt level, I2OS activation upon an interrupt, and the descriptor address.

②③④ Specifies the transfer destination address of converted data.

⑤ Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is terminated in response to a request from a resource.

⑥ Transfer source address

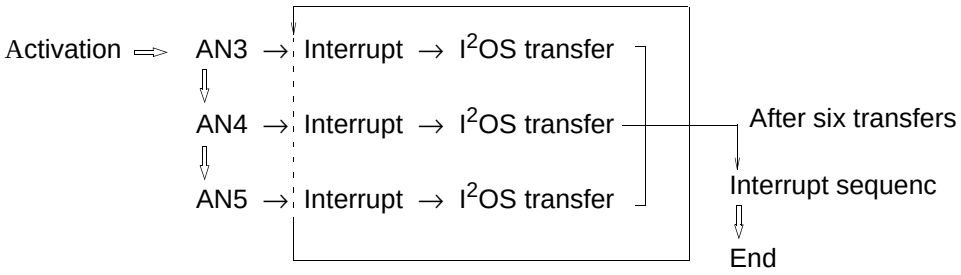
⑦ I^2OS transfer is performed six times. Data is transferred for three channels $\times 2$.

⑧ Specifies continuous mode, start channel AN3, and end channel AN5.

⑨ Specifies activation by external edge and start of A/D conversion.

⑩ Specifies return from an interrupt.

ICR3 : Interrupt control register
BAPL : Buffer address pointer, low-order
BAPM : Buffer address pointer, medium-order
BAPH : Buffer address pointer, high-order
ISCS : I^2OS status register
I/OA : I/O address counter
DCT : Data counter



Usage

- Starting I^2OS in stop mode
  - •To convert analog input AN3 12 times at fixed intervals
  - •To transfer conversion data sequentially to addresses 600H to 618H
  - •To start conversion by external edge input
  - To use the highest interrupt level

I^2OS setting

```
MOV   ICR3    #08H   ..... ①
MOV   BAPL,   #00H   ..... ②
MOV   BAPM,   #06H   ..... ③
MOV   BAPH,   #00H   ..... ④
MOV   ISCS,   #08H   ..... ⑤
MOV   I / OA, #38H   ..... ⑥
MOV   DCT,    #0CH   ..... ⑦
```

A/D converter setting

```
MOV   ADCS1   #DBH   ..... ⑧
MOV   ADCS2   #A4H   ..... ⑨
```

Interrupt sequence

```
MOV   ADCS2   #00H   ..... ⑩
RET
```

① Specifies the highest interrupt level, I2OS activation upon an interrupt, and the descriptor address.

②③④ Specifies the transfer destination address of converted data.

⑤ Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is terminated in response to a request from a resource.

⑥ Transfer source address

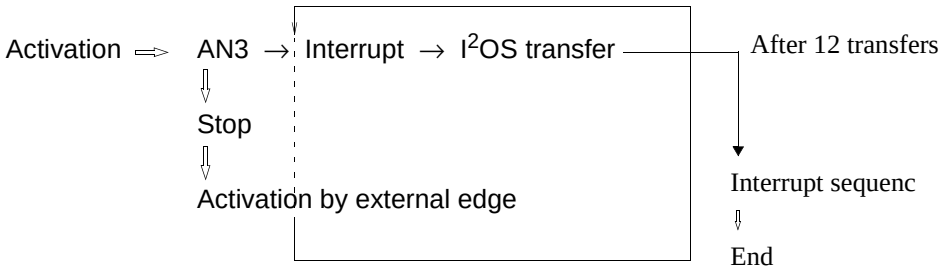
⑦ I^2OS transfer is performed 12 times.

⑧ Specifies continuous mode, start channel AN3, and end channel AN3 (one-channel conversion).

⑨ Specifies activation by external edge and start of A/D conversion.

⑩ Specifies return from an interrupt.

ICR3 : Interrupt control register
BAPL : Buffer address pointer, low-order
BAPM : Buffer address pointer, medium-order
BAPH : Buffer address pointer, high-order
ISCS : I^2OS status register
I/OA :I/O address counter
DCT : Data counter



(5) Conversion data protection

The A/D converter has a conversion data protection function that enables continuous conversion and preservation of multiple data items using I^2OS.
Since there is only one conversion data register, its value is updated each time conversion is completed. Thus, continuous data conversion results in the loss of the previous data due to storage of the new data. To prevent this situation, the A/D converter pauses after conversion if the previous data item has not been transferred to memory by I^2OS. The converted data is not saved until the previous data is transferred to memory.
The pause is released after data is transferred to memory by I^2OS.
If the previous data has been transferred to memory, the A/D converter continues operation without pausing.

Note: * This function is related to the INT and INTE bits of ADCS2.

The data protection function operates only when interrupts are enabled (INTE=1).

If interrupts are disabled (INTE=0), this function is disabled. Continuous A/D conversion results in loss of previous data, since the converted data items are saved to the register one after another.

If I^2OS is not used while interrupts are enabled (INTE=1), the INT bit is not cleared. Thus, the data protection function works and the A/D converter pauses. In this case, clearing the INT bit in the interrupt sequence releases the pause.

If the A/D converter is pausing during I^2OS operation, disabling interrupts may restart the A/D converter. In this case, the value in the conversion data register may be changed without being transferred.

Restarting the A/D converter while it is pausing destroys the standby data.

Flow of data protection function (when I^2OS is used)

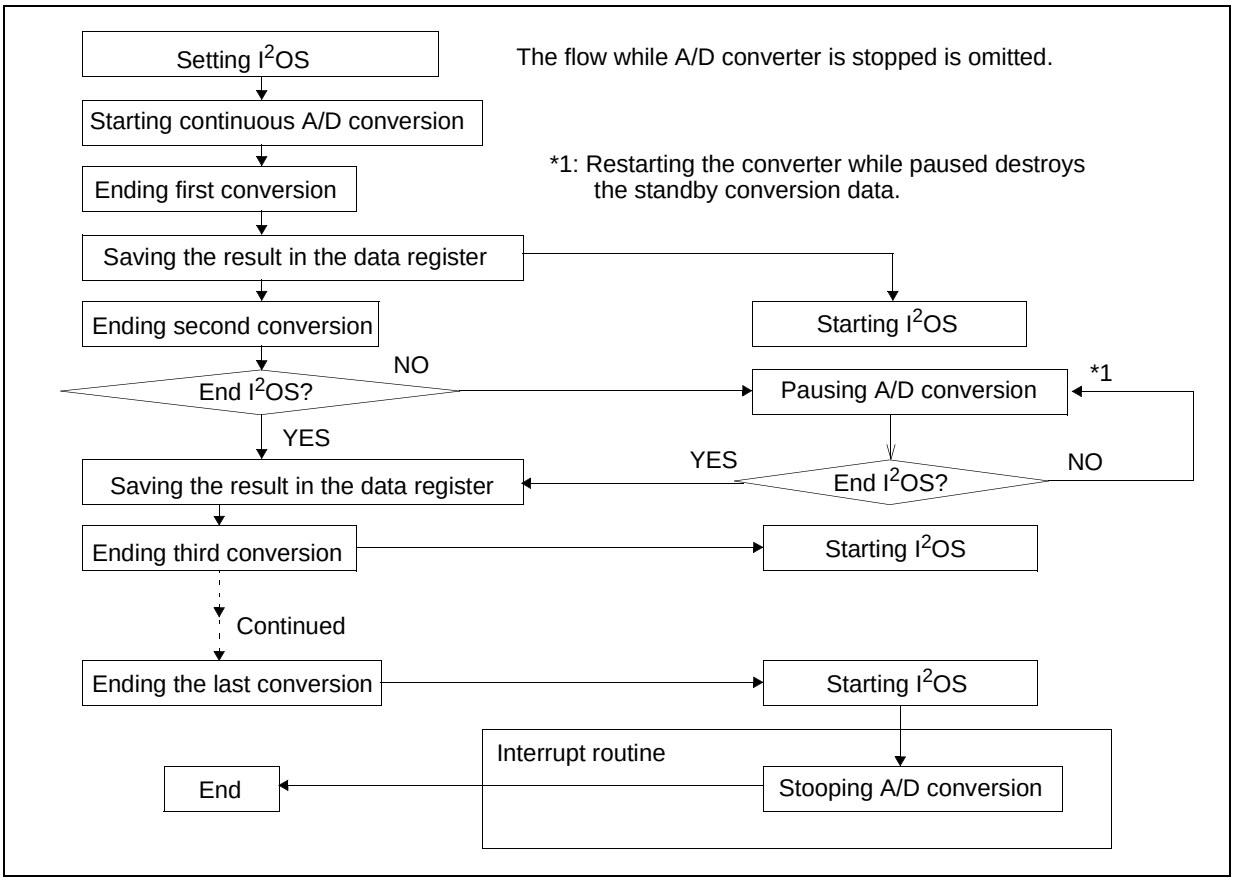


Figure 16.4b Flow Chart of Data Protection Function

16.5 Notes on use

To start the A/D converter upon an external trigger or internal timer, A/D activation factor bits STS1 and STS0 of the ADCS2 register are used. Ensure that the input values of the external trigger or internal timer are inactive. If the values are active, A/D conversion may start immediately.
When setting STS1 and STS0, ensure that '1' (input) is specified for ADTG and '0' (output) is specified for the internal timer (timer 2).

16.5.1 Other considerations

Always write '1' to the ADER bit corresponding to a pin used as analog input.

Analog input enable register

Bit	15	14	13	12	11	10	9	8	
Address: 00001C$_H$	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	ADER
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)	

Port 5 pins are controlled as described below.

0: Port input mode

1: Analog input mode

'1' is set upon a reset.

Chapter 17:
D/A Converter

17.1 Outline

This is an R-2R format D/A converter, having an eight-bit resolution. The D/A converter has two channels. Output control can be performed independently for the two channels using the D/A control register.

17.2 Block Diagram

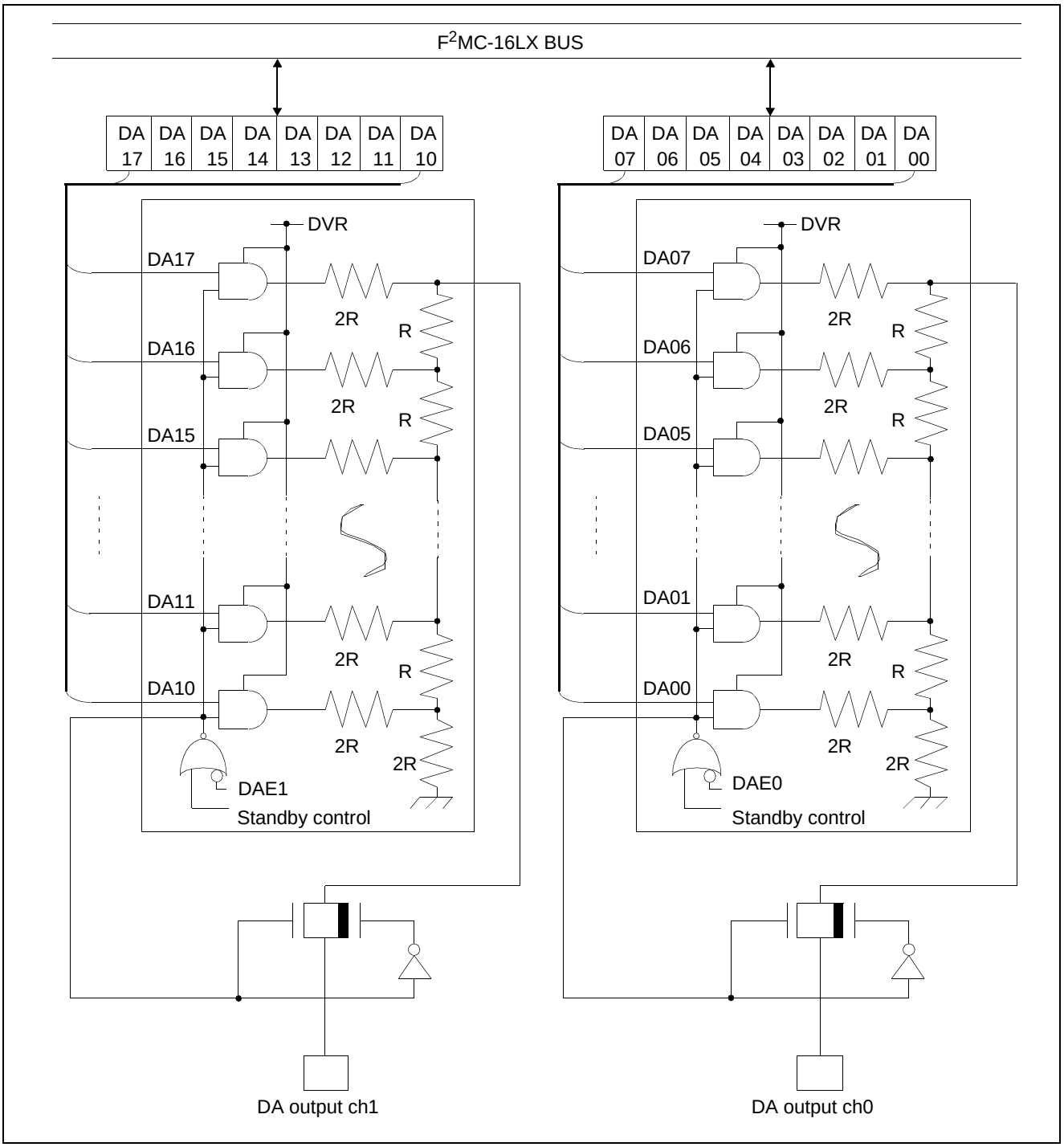


Figure 17.2a Block Diagram of D/A Cobverter

17.3 Registers and Register Details

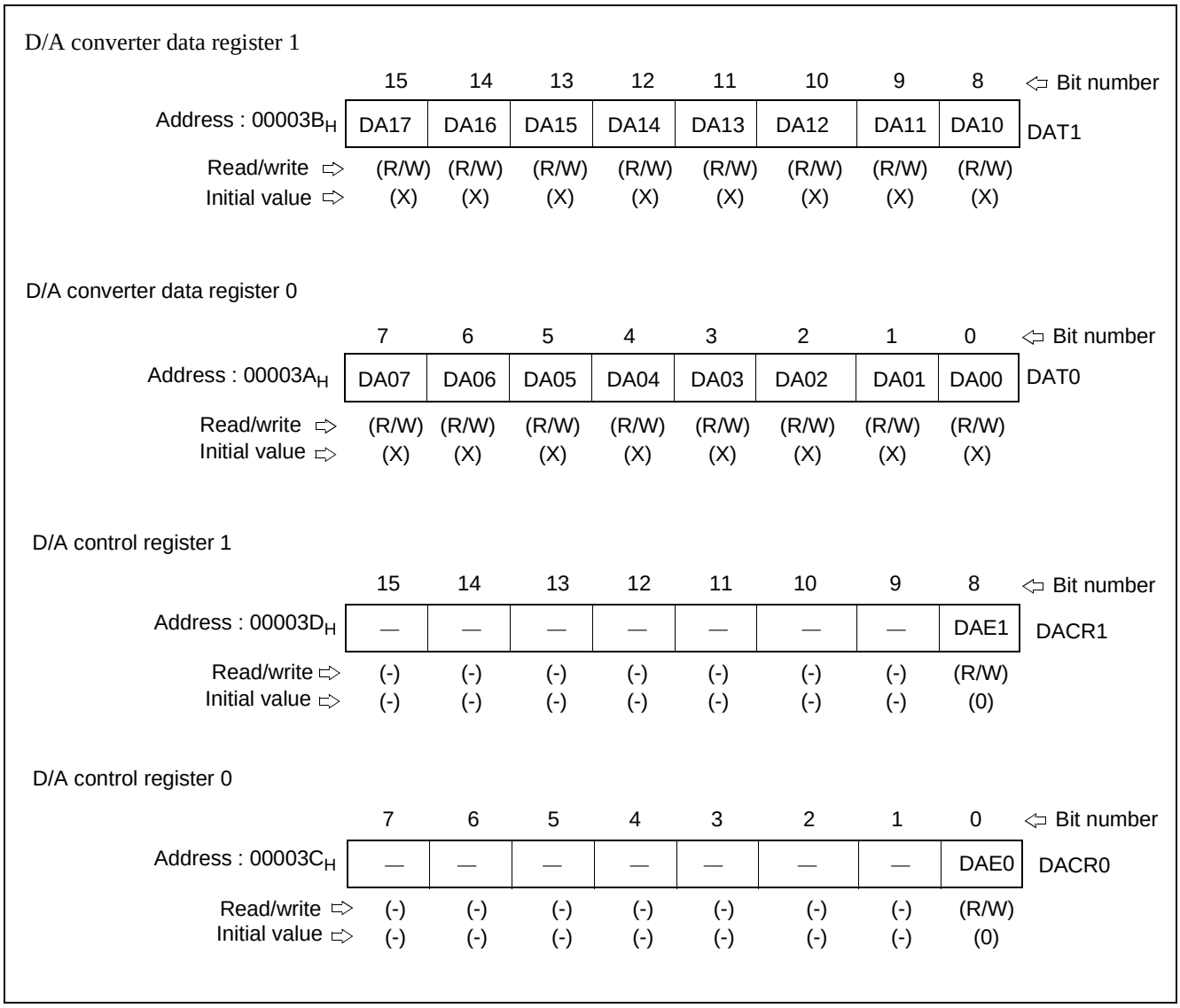


Figure 17.3a Register of D/A Converter

17.3.1 DAT0/1 (D/A data register)

D/A converter data register 1

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 00003B$_H$	DA17	DA16	DA15	DA14	DA13	DA12	DA11	DA10	DAT1
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)	

D/A converter data register 0

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 00003A$_H$	DA07	DA06	DA05	DA04	DA03	DA02	DA01	DA00	DAT0
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)	

[bits 15 to 8] DA17 to DA10

These bits are used to set the output voltage of D/A converter ch1.

These bits are not initialized upon a reset. These bits are readable and writable.

[bits 7 to 0] DA07 to DA00

These bits are used to set the output voltage of D/A converter ch0.

These bits are not initialized upon a reset. These bits are readable and writable.

17.3.2 DACR0/1 (D/A control register)

D/A control register 1

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 00003D$_H$	—	—	—	—	—	—	—	DAE1	DACR1
Read/write ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

D/A control register 0

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 00003C$_H$	—	—	—	—	—	—	—	DAE0	DACR0
Read/write ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	

[bit 0] DAE1 and DAE0

These bits are used to enable or disable the D/A converter output. DAE1 controls channel 1 output, while DAE0 controls channel 0 output.

When ’1’ is written to these bits, D/A output is enabled. When ’0’ is set, D/A output is disabled.

These bits are initialized to ’0’ upon a reset. These bits are readable and writable.

17.4 Operations

D/A output is started by writing a desired D/A output value to the D/A data register (DADR) and setting '1' to the enable bit for the corresponding D/A output channel in the D/A control register (DACR).

Disabling D/A output turns off the analog switch that is inserted serially into the output of each D/A converter channel. In addition, the D/A converter is internally cleared to '0' and the path of the DC current is shut down. This also applies in stop mode.

Table 17.4a shows the theoretical values of D/A converter output voltages

The D/A converter output voltages are between 0 V and 255/256 V × DVR. The output voltage range is changed by regulating the DVR voltage externally.

The D/A converter output does not have an internal buffer amplifier. Since an analog switch (=100 Ω) is serially inserted into the output, allow sufficient settling time when applying an external output load.

Table 17.4a Theoretical values of D/A converter output voltages

Values written to DA07 to DA00 and DA17 to DA10	Theoretical values of output voltages
00_H	0/256 × DVR (=0 V)
01_H	1/256 × DVR
02_H	2/256 × DVR
⋮	⋮
FD_H	253/256 × DVR
FE_H	254/256 × DVR
FF_H	255/256 × DVR

Chapter 18:
Pulse Width Counter (PWC) Timer

18.1 Outline

This module is a multi-function 16-bit up-counter with a reload function and a function for counting pulse widths on the input signal. The module hardware consists of a 16-bit up-counter, input pulse divider, divide ratio control register, four count input pins, one pulse output pin, and a 16-bit control register. These perform the following functions.

Timer function:

- Interrupt requests can be generated at specified time intervals.
- A pulse signal can be output synchronized with the timer period.
- The counter clock can be selected from three internal clocks.

Pulse width count function:

- Measures the time between events on an external pulse input.
- The counter clock can be selected from three internal clocks.
- Count modes H pulse width (⇑ to ⇓)/L pulse width (⇓ to ⇑)

  Rising edge period (⇑ to ⇑)/Falling edge period (⇓ to ⇓)

  Inter-edge count (⇑ or ⇓ to ⇓ or ⇑)
- Using the 8-bit input divider, the module can divide an input pulse signal by 2^{2n} (n = 1, 2, 3, 4) and measure the period.
- An interrupt request can be generated on count completion.
- Single-shot or continuous counting can be selected.

The MB90580 series contains one PWC timer channels.

18.2 Block Diagram

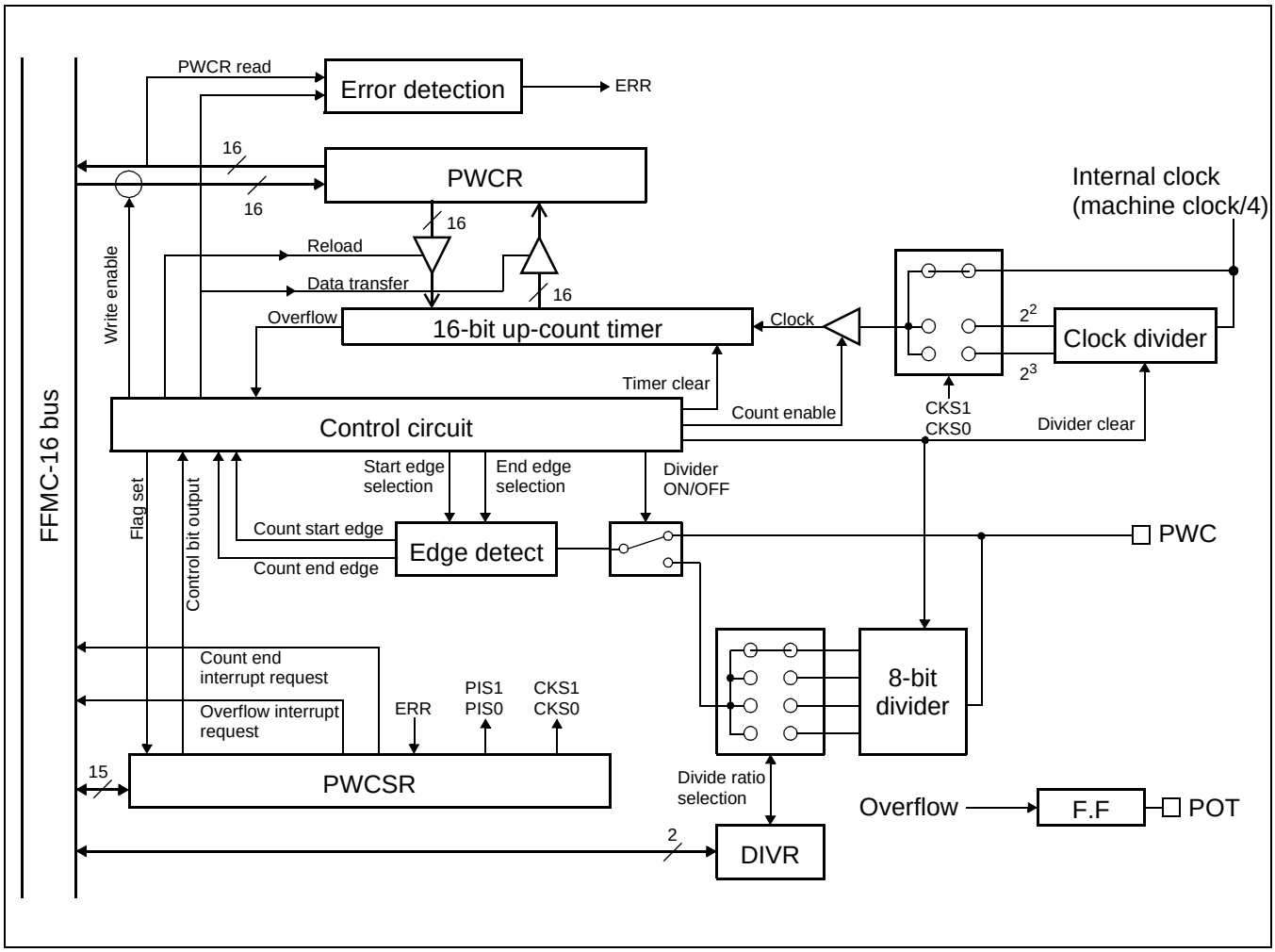


Figure 18.2a lock Diagram of Pulse Width Counter Timer

18.3 Regiaters and Register Details

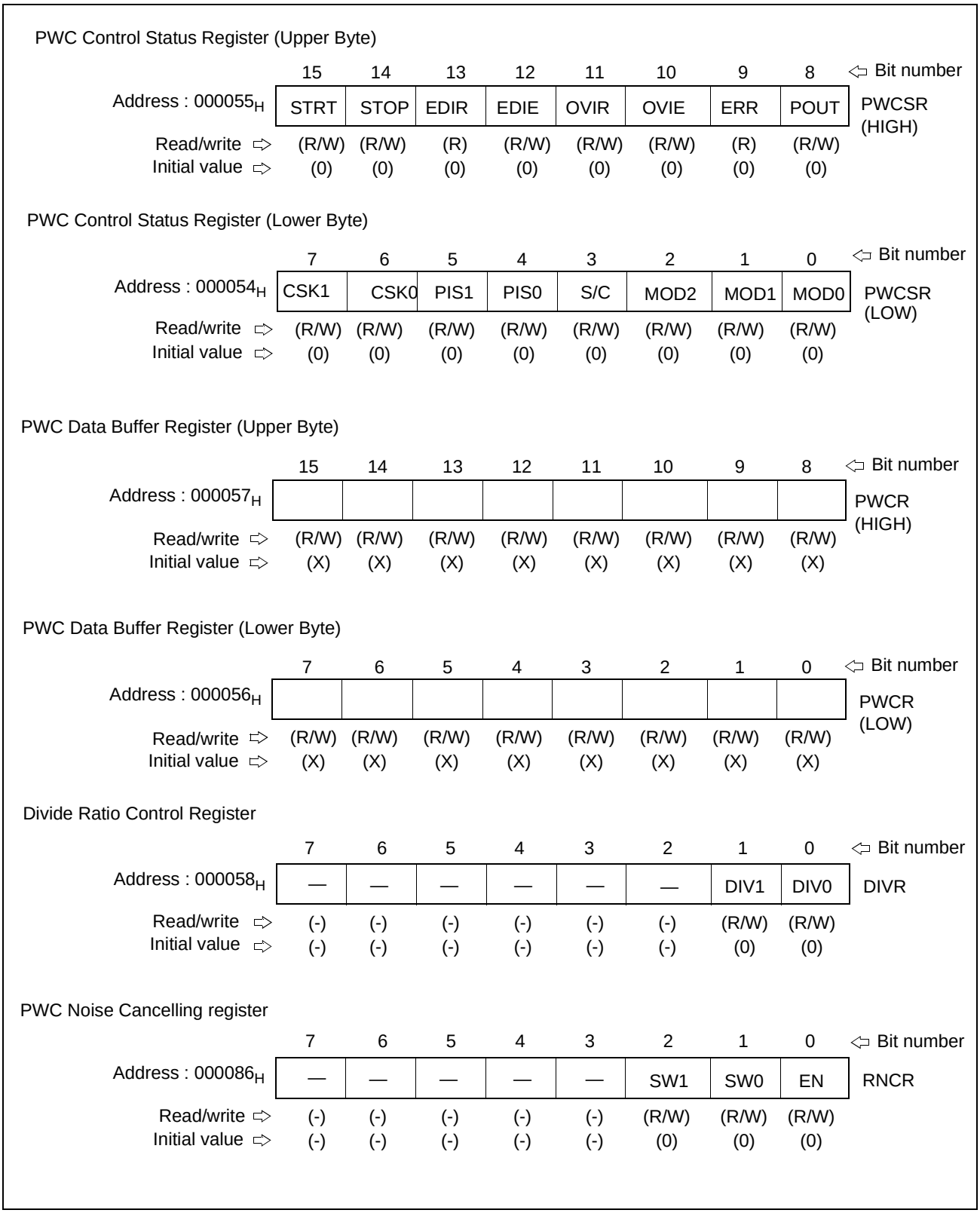


Figure 18.3a Register of Pulse Width Counter Timer

18.3.1 PWC control status register (PWCSR)

PWC Control Status Register (Upper Byte)

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 000055_H	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	POUT	PWCSR (HIGH)
Read/write ⇨	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

PWC Control Status Register (Lower Byte)

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 000054_H	CSK1	CSK0	PIS1	PIS0	S/C	MOD2	MOD1	MOD0	PWCSR (LOW)
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)	

The PWCSR is used to control the operation of the PWC timer and to read the PWC timer status.

[bit 15] STRT (Start) & [bit 14] STOP (Stop)

These bits start, restart, and stop the 16-bit up-count timer. Reading the bits returns the operating state of the timer. The bit functions are as follows.

Function of STRT and STOP bits when they are written. (Operation control)

STRT	STOP	Operation Control Function
0	0	No function. Has no effect on operation.
0	1	Starts or restarts the timer (count enable). Note: The clear bit instruction can be used.
1	0	Forcibly halts the operation of the timer (count disable). Note: The clear bit instruction can be used.
1	1	No function. Has no effect on operation.

Meaning of the STRT and STOP bits when they are read. (Operating status indication)

STRT	STOP	Operating Status Indication
0	0	Timer is halted (not started or count ended). (Initial value)
1	1	Timer is counting (count in progress).

After a reset: Initialized to 00_B.

Readable and writable. Note that the meanings of the bits differ for reading and writing.

Always read as 11_B by read-modify-write instructions regardless of the actual values.

Although bit manipulation instructions (such as the bit clear instruction) can be used to write to the STRT and STOP bits to start and stop the timer, bit manipulation instructions cannot be used to read the operating status (as these always indicate that the timer is operating).

[bit 13] EDIR (End interrupt request)

This flag indicates when counting ends in pulse width count mode. A count end interrupt request is generated if the interrupt is enabled (bit 12: EDIE = "1") when this bit is set.

Set timing	Set when pulse width counting ends (when the count result is placed in PWCR).
Clear timing	Cleared by reading PWCR (the count result).

Note: This bit has no meaning in timer mode.

After a reset: Initialized to "0".

Read-only. Writing to the bit does not change the value.

[bit 12] EDIE (End interrupt enable)

Controls the count end interrupt request in pulse width count mode as follows.

0	Disable output of count end interrupt requests (do not generate an interrupt when EDIR is set). (Initial value)
1	Enable output of count end interrupt requests (generate an interrupt when EDIR is set).

Note: Always set to "0" during timer mode.

After a reset: Initialized to "0".

Readable and writable.

[bit 11] OVIR (Overflow interrupt request)

This flag indicates when the 16-bit up-count timer overflows from $FFFF_H$ to 0000_H. Operates in all modes. A timer overflow interrupt request is generated if the interrupt is enabled (bit 10: OVIE = "1") when this bit is set.

Set timing	Set when a timer overflow occurs ($FFFF_H$ to 0000_H).
Clear timing	Cleared by writing "0" or by the extended intelligent I/O service.

After a reset: Initialized to "0".

Readable and writable. However, only writing "0" is valid. Writing "1" does not change the bit value. Read-modify-write instructions always read the bit as "1" regardless of the actual bit value.

[bit 10] OVIE (Overflow interrupt enable)

Controls the timer overflow interrupt request as follows.

0	Disable output of overflow interrupt requests (do not generate an interrupt when OVIR is set). (Initial value)
1	Enable output of overflow interrupt requests (generate an interrupt when OVIR is set).

After a reset: Initialized to "0".

Readable and writable.

[bit 9] ERR (Error)

This flag is used when continuous counting is performed in pulse width count mode. The flag indicates that the next count has completed before the previous count result has been read from PWCR. When this occurs, PWCR is overwritten with the new count result and the previous result is lost. Counting continues regardless of the value of this bit.

Set timing	Set when a count result that has not been read is overwritten by the next result.
Clear timing	Cleared by reading PWCR (the count result).

After a reset: Initialized to "0".

Read-only. Writing to the bit does not change the value.

[bit 8] POUT (Pulse output)

In timer mode, this bit is inverted each time the 16-bit up-count timer overflows from $FFFF_H$ to 0000_H.

The bit has no meaning in pulse width count mode.

Set timing	Set when the timer overflows from $FFFF_H$ to 0000_H when the value of POUT is "0", or by writing "1" when the timer is halted.
Clear timing	Cleared when the timer overflows from $FFFF_H$ to 0000_H when the value of POUT is "1", by writing "0" when the timer is halted, or by a reset.

After a reset: Initialized to "0".

Readable and writable. However, the bit can only be written to when the timer is halted (when bit 15 and bit 14: STRT and STOP are both "0"). The value of the bit does not change if written to during timer operation (when bit 15 and bit 14: STRT and STOP are both "1").

[bits 7, 6] CKS1, CKS0 (Clock select 1, 0)

These bits select the internal count clock as follows.

CSK1	CSK0	Count Clock Selection
0	0	Machine cycle divided by 4 (0.25µs for a 16MHz machine cycle) (Initial value)
0	1	Machine cycle divided by 16 (1.0µs for a 16MHz machine cycle)
1	0	Machine cycle divided by 32 (2.0µs for a 16MHz machine cycle)
1	1	**Note:** Prohibited setting

After a reset: Initialized to "00_B".

Readable and writable. However, setting "11_B" is prohibited.

Note: Changing the setting after activating the timer is prohibited. Only write to these bits before starting or after halting the timer.

[bits 5, 4] PIS1, PIS0 (Pulse input select)

These bits select the input pin on which to perform pulse width counting.

PIS1	PIS0	Count Input Pin Selection
0	0	Always set this value. (Initial value)
0	1	Setting unavailable (Do not set any of these values.)
1	0	
1	1	

After a reset: Initialized to "00_B".

Readable and writable.

Note: Changing the setting after activating the timer is prohibited. Only write to these bits before starting or after halting the timer.

Note: When developing software for the MB90580 series, always set these bits to "00_B".

[bit 3] S/C (Single/Continuous)

Select the count mode as follows.

S/C	Count Mode Selection	Timer Mode	Pulse Width Count Mode
0	Single-shot count mode (Initial value)	No reload (single-shot)	Halt after one count.
1	Continuous count mode	Perform reload (reload timer)	Continuous counting: Buffer register enabled

After a reset: Initialized to "0".

Readable and writable.

Note: Changing the setting after activating the timer is prohibited. Only write to these bits before starting or after halting the timer.

[bits 2, 1, 0] MOD2, MOD1, MOD0 (MOD2, 1, 0)

These bits select the operation mode and the pulse edges for width counting.

MOD2	MOD1	MOD0	Operation Mode/Count Edge Selection
0	0	0	Timer mode, no pulse output (Initial value)
0	0	1	Timer mode, pulse output enabled (using the POT pin): Reload mode only
0	1	0	Inter-edge pulse width count mode (⇑ or ⇓ to ⇓ or ⇑) *
0	1	1	Divided period count mode (using input divider) *
1	0	0	Rising-edge to rising-edge count mode (⇑ to ⇑). *
1	0	1	"H" pulse width count mode(⇑ to ⇓). *
1	1	0	"L" pulse width count mode(⇓ to ⇑). *
1	1	1	Falling-edge to falling-edge count mode (⇓ to ⇓). *

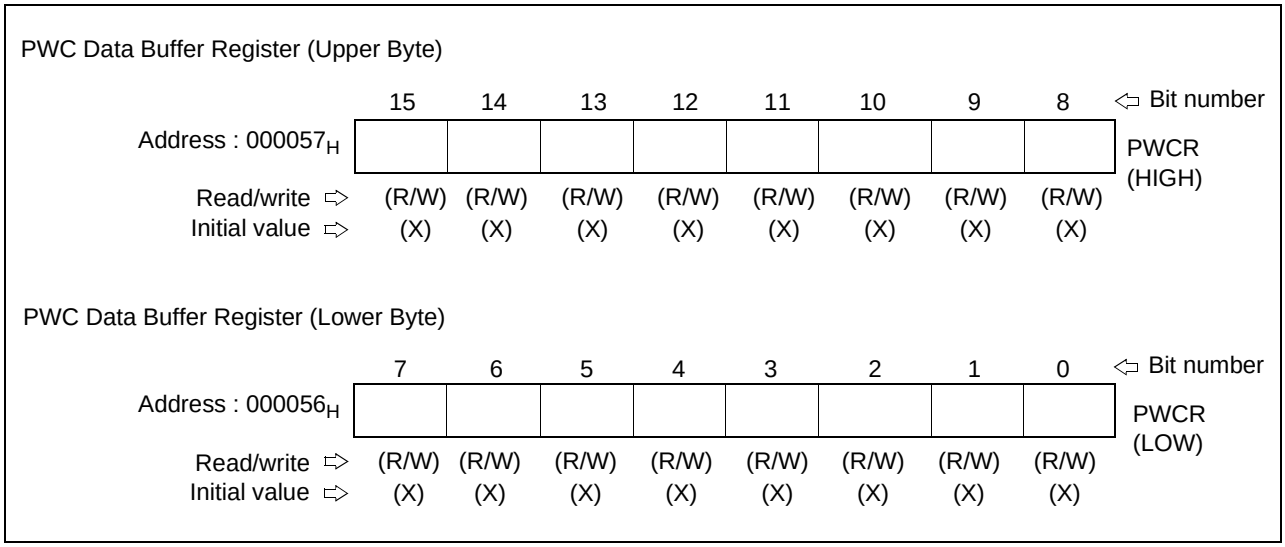
After a reset: Initialized to "000_B".

Readable and writable.

Note: Changing the setting after activating the timer is prohibited. Only write to these bits before starting or after halting the timer.

Note: When continuous count mode is set for the settings marked with an asterisk (*), the divider circuit for the internal count clock is not cleared when the count ends so as to accumulate the number of edges. In all other modes, the divider circuit for the internal count clock is cleared when the count ends.

18.3.2 PWC data buffer register (PWCR)



(1) In timer mode

In reload timer operation mode (bit 3 S/C of PWCSR = "1"), this register stores the reload value. In this case, the register is readable and writable.

In single-shot timer operation mode (bit 3 S/C of PWCSR = "0"), accessing this register directly accesses the up-count timer. Although both reading and writing are allowed in this mode, only write to the register when the timer is halted. The register can be read at any time to read the current timer value.

(2) In pulse width count mode ♦ Read-only ♦

In continuous count mode (bit 3 S/C of PWCSR = "1"), this register acts as a buffer register to store the previous count result. In this case, the register is read-only and writing does not change the register value.

In single-shot count mode (bit 3 S/C of PWCSR = "0"), accessing this register directly accesses the up-count timer. The register is read-only in this mode also and writing does not change the register value. The register can be read at any time to read the current timer value. After the count ends, the register stores the count result.

Note: Always use word transfer instructions to access this register.

After a reset: Initialized to "0000_H".

18.3.3 Divide Ratio Control Register (DIVR)

Divide Ratio Control Register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 000058_H	—	—	—	—	—	—	DIV1	DIV0	DIVR
Read/write ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	(R/W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(-)	(-)	(0)	(0)	

This register is only used in divided period count mode (bits 2, 1, 0: MOD2, 1, 0 of PWCSR = "011").

In divided period count mode, pulses input from the count pin are divided by the divide ratio set in this register and the period of the divided signal is measured. The divide ratio is selected as follows.

DIV1	DIV0	Divide Ratio Selection
0	0	2^2 = divide by 4 (Initial value)
0	1	2^4 = divide by 16
1	0	2^6 = divide by 64
1	1	2^8 = divide by 256

After a reset: Initialized to "00_B".

Readable and writable.

Note: Changing the setting after activating the timer is prohibited. Only write to these bits before starting or after halting the timer.

18.3.4 PWC noise cancelling register (RNCR)

PWC Noise Cancelling register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 000086_H	—	—	—	—	—	SW1	SW0	EN	RNCR
Read/write ⇨	(-)	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(-)	(0)	(0)	(0)	

The PWC noise removal circuit is used for removing noises form the input signal. H level and L level detection will be applied to the input signal after it was ‘cleaned’ by the noise filter.

Noise removal circuit is a digital low pass filter, the filter remove the high frequency components of the input signal. The noise-removed signal is called ‘RMCSIG’. This signal has the same polarilty with the orginial input signal, but the there may be slight phase difference. The SW bits of the noise cancelling register specifies the noise pulse width which can be removed by the filter circuit.

This noise cancelling register is a 8-bit register, when reset, all bits will be initialized to 0.

[bits 2, 1] SW1, SW0

SW1 and SW0 is the clock mode selection bit which specify the noise pulse width to be removed. The timing of the following table assumes the main clock is 16MHz.

SW1	SW0	Input Clock	Noise Pulse Width
0	0	0.5 MHz	2.0 µs
0	1	31.25 KHz	32.0 µs
1	0	15.62 KHz	64.0 µs
1	1	7.81 KHz	128.0 µs

[bits 0] EN

EN bit is used for enabling this noise cancelling function.

0	Noise cancelling function disabled (Initial value)
1	Noise canncelling function enabled

18.4 Operations

(1) Summary of Operation

This block is a multi-function timer based on a 16-bit up-count timer and incorporating a count input pin and 8-bit input divider. The block has two main functions: a timer function and a pulse width count function. Two types of count clock can be selected for either function. The following describes the basic functions and operation of each of these functions.

(a) Timer Function

This function is an up-count timer which can be selected to operate in reload or single-shot mode.

Once started, the timer counts on each count clock.

An interrupt request can be generated when an overflow from $FFFF_H$ to 0000_H occurs.

When an overflow occurs:

- Single-shot mode: ..... The count stops.
- Reload mode: ..... The timer is reloaded with the contents of the reload register and the count restarts.

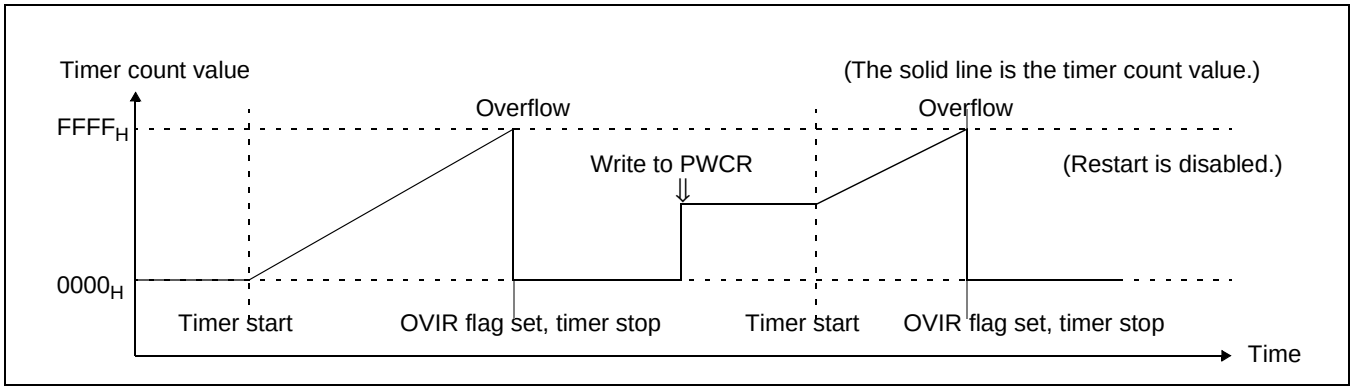


Figure 18.4a Timer Operation (Single-Shot Mode)

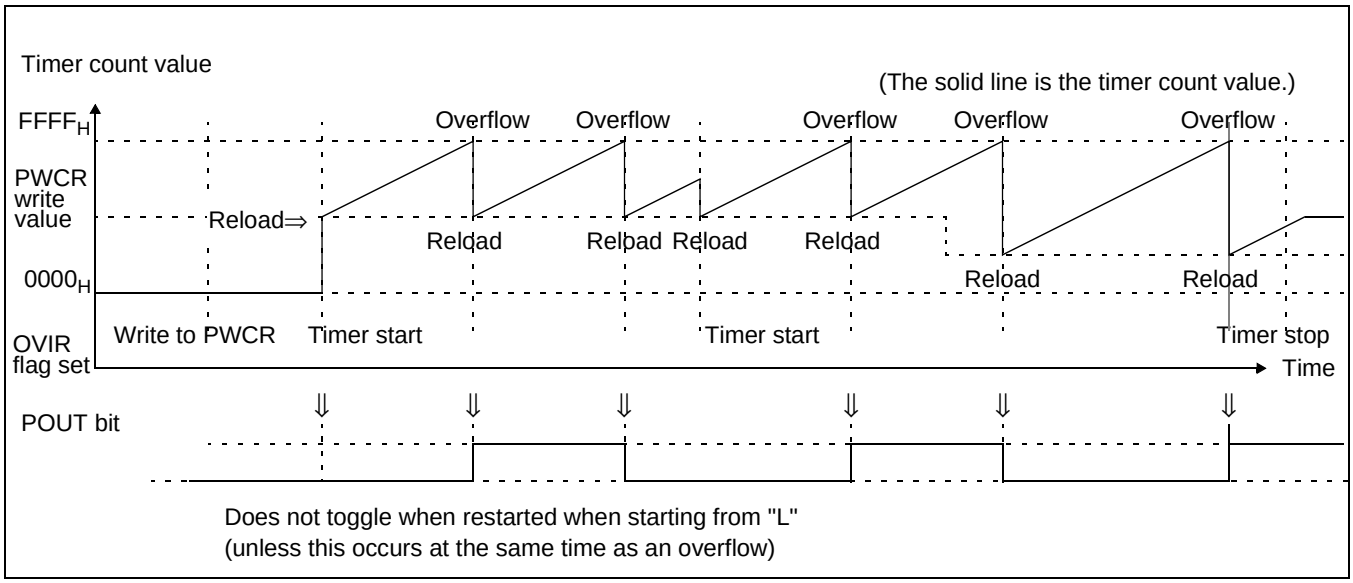


Figure 18.4b Timer Operation (Reload Mode)

(2) Pulse Width Count Function

This function counts the time period between specified events on an input pulse.

After the function is activated, the count does not start until the specified count start edge is input. The counter is cleared to "0000_H" and counting starts when the start edge is detected. The count halts when the end edge is detected. The count value at the end of this period is stored in the register as the pulse width.

An interrupt request can be generated when the count ends or when an overflow occurs. After counting completes:

- Single-shot count mode: ... Operation halts.
- Continuous count mode: ... The timer value is transferred to the buffer register and the count halts until the next start edge is input.

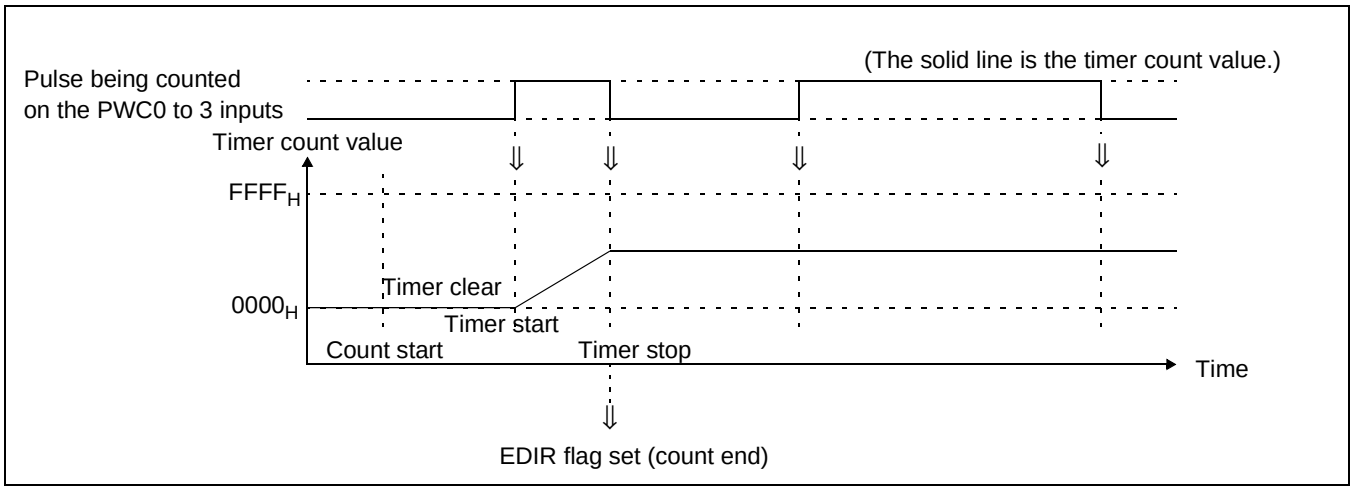


Figure 18.4c Pulse Width Count Operation (Single-Shot Count Mode, "H" Width Count Mode)

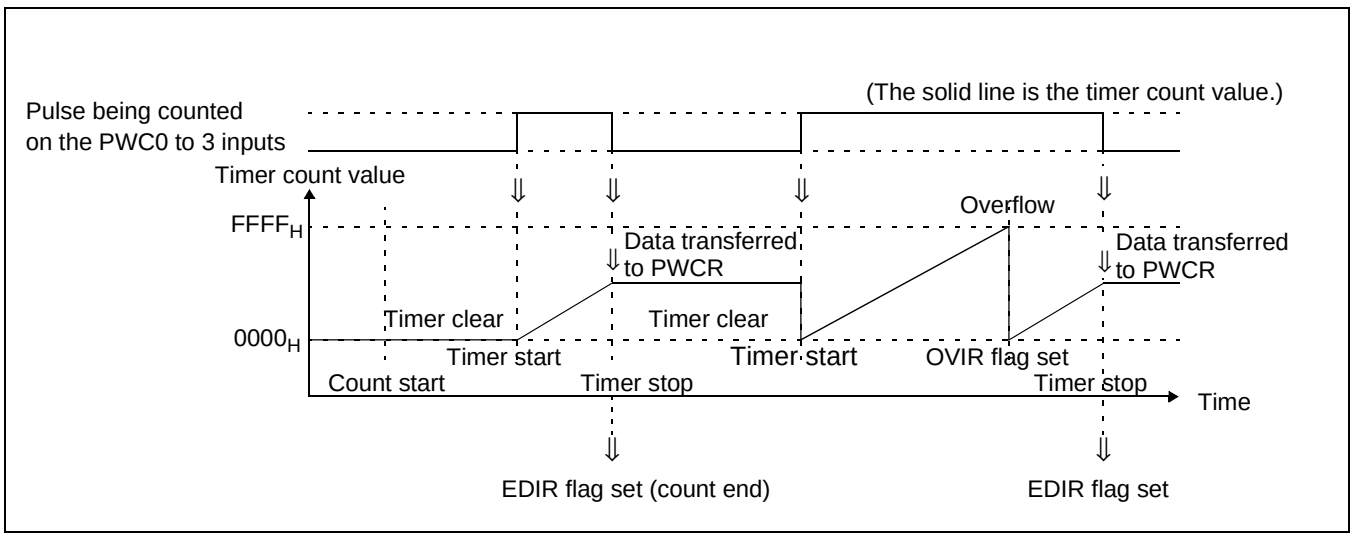


Figure 18.4d Pulse Width Count Operation (Continuous Count Mode, "H" Width Count Mode)

(3) Count Clock Selection

The timer count clock can be selected from three internal clock sources. The available clock sources are listed below.

Table 18.4a Count Clock Selection

PWCSR/bit7, 6:CKS1, 0	Selected Internal Count Clock
00_B	Machine cycle divided by 4 (0.25µs for a 16MHz machine cycle) (Initial value)
01_B	Machine cycle divided by 16 (1.0µs for a 16MHz machine cycle)
10_B	Machine cycle divided by 32 (2.0µs for a 16MHz machine cycle)

The selection is initialized to "machine cycle divided by 4" after a reset.

Note: Always select the count clock before starting the timer.

(4) Operation Mode Selection

The operation mode and count mode are selected by PWCSR settings.

- Operation mode setting PWCSR bits 2, 1, and 0: Bits MOD2, MOD1, and MOD0
  (Selects timer or pulse width count mode and specifies which edges control counting.)
- Count mode setting PWCSR bit 3: S/C bit
  (Selects single-shot or continuous counting, or reload or single-shot operation.)

The following lists the operation modes selected by the mode setting bits.

Figure 18.4e Operation Mode Selection

Operation Mode			S/C	MOD2	MOD1	MOD0
Timer	Single-shot timer		0	0	0	0
	Reload timer		1	0	0	0
	Setting prohibited		1	0	0	1
Pulse width count	⇑ or ⇓ to ⇑ or ⇓	Single-shot count: Buffer not used	0	0	1	0
	Counts between all edges	Continuous count: Buffer used	1	0	1	0
	Divided period count	Single-shot count: Buffer not used	0	0	1	1
	(divide by 1 to 256)	Continuous count: Buffer used	1	0	1	1
	⇑ to ⇑	Single-shot count: Buffer not used	0	1	0	0
	Rising-edge to rising-edge count	Continuous count: Buffer used	1	1	0	0
	⇑ to ⇓	Single-shot count: Buffer not used	0	1	0	1
	"H" pulse width count	Continuous count: Buffer used	1	1	0	1
	⇓ to ⇑	Single-shot count: Buffer not used	0	1	1	0
	"L" pulse width count	Continuous count: Buffer used	1	1	1	0
	⇓ to ⇓	Single-shot count: Buffer not used	0	1	1	1
	Falling-edge to falling-edge count	Continuous count: Buffer used	1	1	1	1

The initial value after a reset selects single-shot timer mode.

Note: Always select the operation mode before starting the timer.

(5) Starting and Stopping the Timer and Pulse Width Count

Starting, restarting, and forcibly halting each operation is performed using bits 15 and 14 (STRT and STOP) of PWCSR. Writing "0" to the STRT bit starts or restarts operation and writing "0" to the STOP bit forcibly halts operation. However, neither bit performs its operation if the values written to the two bits are contradictory. When using instructions other than bit manipulation instructions (byte or larger instructions), only write the following bit combinations.

Table 18.4b Start and Stop Bit Functions

Function	STRT	STOP
Start or restart timer or pulse width count.	0	1
Forcibly halt timer or pulse width count.	1	0

When using a bit manipulation instruction (clear bit instruction), writing of the above combinations is enforced automatically by hardware so no particular care is required.

(a) Operation After Starting

- Timer mode: ..... The count operation starts immediately.
- Pulse width count mode: ... The count does not start until the count start edge is input.
  After detecting the count start edge, the 16-bit up-count timer is cleared to 0000_H and counting starts.

(b) Restarting the Timer

Re-applying the start command (writing "0" to the STRT bit) while the timer is still operating after starting in timer mode or pulse width count mode is called restarting. The operation performed for a restart depends on the mode, as follows.

- Single-shot timer mode: .... No effect on the operation.
- Reload timer mode: ..... Performs a reload and continues operation.
  If the restart occurs at the same time as an overflow, the overflow flag (OVIR) is set and the POUT bit inverted.
- Pulse width count mode: ... Has no effect on the operation if the timer is waiting for the count start edge.
  If applied during a count, the count halts and the timer returns to the "waiting for a count start edge" state. If the restart occurs at the same time as a count end edge is detected, the count end flag (EDIR) is set and, in continuous count mode, the count result is transferred to PWCR.

(c) Stopping the Timer

In single-shot timer mode or single-shot count mode, the count halts automatically when the timer overflows or the count ends and therefore you do not need to explicitly stop the timer. However, you must forcibly stop the timer in other modes or if you wish to stop the timer before it halts automatically.

(d) Checking the Operating State

The STRT and STOP bits described previously function as indicator bits for the operating state of the timer when read. The table below lists the bit meanings.

Table 18.4c Operating State Indicator Bit Functions

STRT	STOP	Operating State
0	0	The timer is stopped (other than when waiting for a count start edge). Indicates that the timer has not been started or that counting has ended.
1	1	The timer is counting or waiting for a count start edge.

The STRT and STOP bits both have the same value when read. However, as the bits always have the value "1_B" when read by read-modify-write instructions (such as bit manipulation instructions), do not use these instructions to read the bit values.

(6) Clearing the Timer

The 16-bit up-count timer is cleared to 0000_H in the following cases.

- A reset
- When counting starts after detection of a count start edge in pulse width count mode(6)

(7) Details of Timer Mode Operation

(a) Single-Shot Operation Mode

When the timer is started in this mode, the timer counts up on each count clock. The timer automatically stops when an overflow from $FFFF_H$ to 0000_H occurs.

If PWCR is set before starting the timer, the count starts from the set value. In this case, the set value is not saved and PWCR contains the current count value.

Bit 8 (POUT) of PWCSR is inverted when an overflow occurs but the value is not output from the pin in this mode, even if pulse output mode is specified.

(b) Reload Operation Mode

When the timer is started in this mode, the reload value in PWCR is set to the timer and the timer counts up on each count clock. When an overflow from $FFFF_H$ to 0000_H occurs, the reload value in PWCR is set again to the timer (reloading), the POUT bit (bit 8) of PWCSR is inverted, and the count operation repeated. The timer does not stop until forcibly halted by writing to the STOP bit of PWCSR or until a reset occurs.

The reload value set to PWCR before starting the timer is stored during counting and is set to the timer when the timer is started or restarted and each time an overflow occurs. If the set value is changed during counting, the new reload value is used when the next overflow or restart occurs.

(c) Timer Value and Reload Value

In single-shot operation mode, accessing PWCR directly accesses the up-count timer. Writing a value to PWCR writes the value directly to the timer and reading PWCR during count operation reads the current timer value. Setting a value to PWCR before starting the timer causes the count to start from the specified value.

In reload operation mode, the up-count timer cannot be accessed and PWCR acts as the reload register (stores the reload value). The value written to PWCR is set to the timer when the timer is started or restarted and each time an overflow occurs. Reading PWCR reads the stored reload value.

The value in PWCR and the timer value are indeterminate if the timer is set to single-shot mode after forcibly halting operation in reload mode. Therefore, always set a value before using the timer.

The value in PWCR is indeterminate if the timer is set to reload mode after forcibly halting operation in single-shot mode. Therefore, always set a value before using the timer.

(d) Generation of Interrupt Requests

Interrupt requests can be generated by overflows when operating in timer mode. When an overflow occurs due to the timer counting up, the overflow flag is set and an interrupt request is generated if the overflow interrupt request is enabled.

(e) Timer Period

If the timer is started in single-shot mode after setting 0000_H to PWCR, the timer overflows after 65536 counts and the count stops. The following formula calculates the time from the timer starting to the timer stopping.

$$T_1 = (65536 - n_1) \times t$$

- T_1 … Time from start to stop (µs)
- n_1 … Timer value set in PWCR when the timer starts
- t … Count clock period (µs)

If the timer is started in reload mode after setting 0000_H to PWCR, the timer overflows after each 65536 counts. The following formulas calculate the reload period and the period of the POT pin output pulse.

$$T_R = (65536 - n_R) \times t$$
$$T_{POUT} = T_R \times 2$$

- T_R … Reload period (overflow period) (µs)
- T_{POUT} … Period of the POT pin output pulse (µs)
- n_R … Reload value stored in PWCR
- t … Count clock period (µs)

(f) Count Clock and Maximum Period

For timer mode, the maximum period is when 0000_H is set to PWCR.

The following table lists the count clock period and maximum timer period for a 16MHz machine cycle (indicated by ϕ below).

Table 18.4d Count Clock and Period

Count Clock Selection	CKS1, 0 = 00 (ϕ/4)	CKS1, 0 = 01 (ϕ/16)	CKS1, 0 = 10 (ϕ/32)
Count clock period	0.25µs	1µs	2µs
Maximum timer period	16.38ms	65.5ms	131.1ms

(g) Timer Operation Flowchart

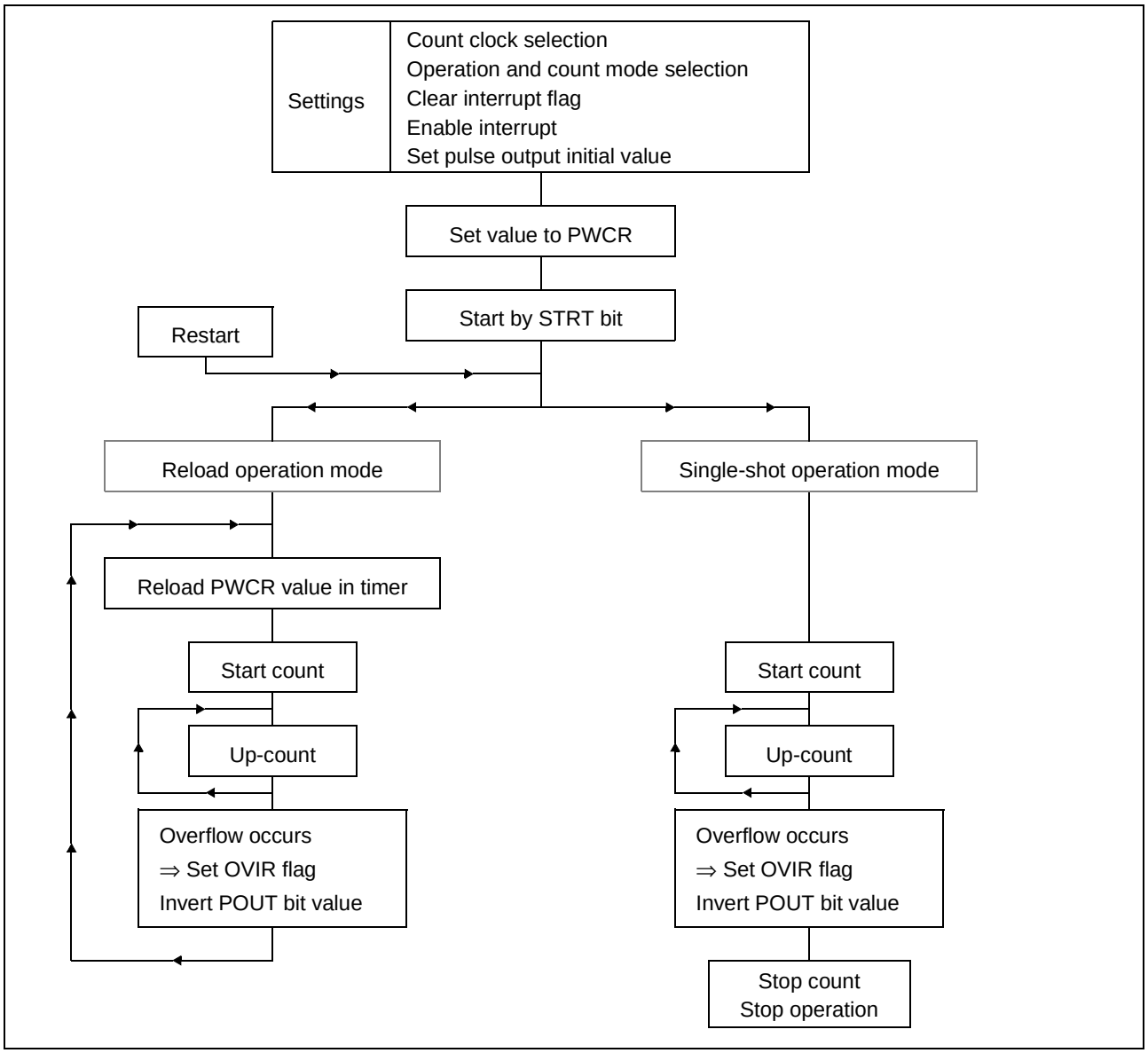


Figure 18.4f Flowchart of Timer Mode Operation

(8) Details of Pulse Width Count Mode Operation

(a) Count Input Pins and Pin Selection

The pins used to input the signal for pulse width counting are fixed as pin PWC0 for ch0, PWC1 for ch1, PWC2 for ch2, and PWC3 for ch3. Always set bits 4 and 5 of PWCSR to "00" on the MB90580.

Table 18.4e Count Input Pin Selection (n = 3 to 0)

PIS1	PIS0	Count Input Pin Selection
0	0	The PWCn pin for the channel (Initial value)
0	1	Setting unavailable (Do not set any of these values.)
1	0	
1	1	

Note: Only select or change the count input pin while the timer is halted.

(b) Single-Shot Counting and Continuous Counting

Pulse width count mode has a mode to perform a count once only and a mode to perform pulse width counting continuously. The following lists the differences between the two modes.

- Single-shot count mode: .. When the first count end edge is input, the timer stops counting, the count end flag (EDIR) in PWCSR is set, and no further count is performed.
  (However, if a restart is specified at the same time, the timer goes to the "waiting for a count start edge" state.)
- Continuous count mode: ... When a count end edge is input, the timer stops counting, the count end flag (EDIR) in PWCSR is set, and the count remains stopped until the next count start edge is input. When the next count start edge is input, the timer is cleared to 0000_H and counting restarts.
  The count result in the timer is transferred to PWCR when the count ends.

The S/C bit in PWCSR selects the mode (see (3) Operation Mode Selection).

Note: Only select or change the count mode while the timer is halted.

Note: For any of the pulse width count modes used with continuous count mode, the divider circuit for the internal count clock is not cleared when the count ends. Therefore, the result in continuous count modes is the accumulated number of edges.

(c) Count Result Data

The handling of the count result and timer value and the function of PWCR differ for single-shot count mode and continuous count mode. The differences are as follows.

- Single-shot count mode: ... Reading PWCR during timer operation reads the current timer value.
  Reading PWCR after the count has ended reads the count result.
- Continuous count mode: ... The count result in the timer is transferred to PWCR when the count ends.
  Reading PWCR reads the result of the previous count. PWCR continues to store the previous count result while counting is in progress. The timer value during counting cannot be read.

In continuous count mode, if the previous count result is not read before the next count completes, the new count result overwrites the old value. If this occurs, the error flag (ERR) in PWCSR is set. The error flag (ERR) is automatically cleared when PWCR is read.

(d) Count Mode and Count Operation

The count mode can be selected from five different modes. The mode determines which part of the input pulse to measure. To accurately measure the width of high frequency pulses, a mode is available to divide the input pulses by a specified ratio and to measure the resulting period. The following describes each mode.

Table 18.4f Count Modes

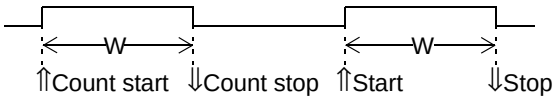
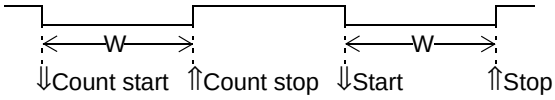
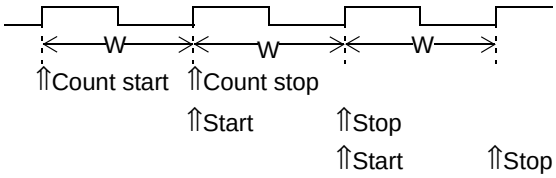
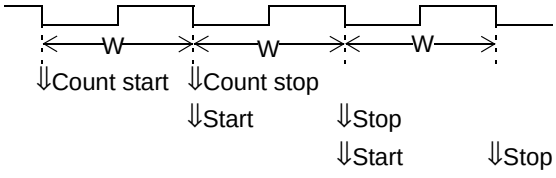
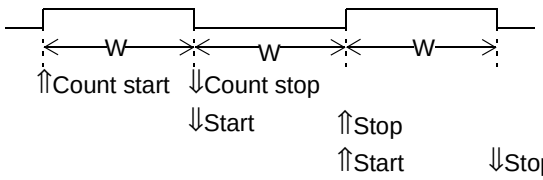
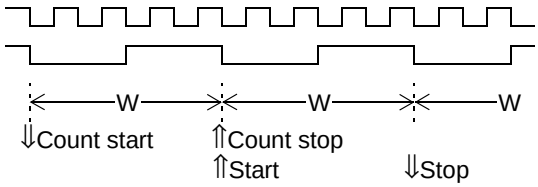
Count Mode	MOD2	MOD1	MOD0	Count Operation (w: Pulse width being measured)
H pulse width count	1	0	1	 Measures the width of the "H" period. • Count (meaOPsurement) start: Rising edge detected • Count (measurement) end: Falling edge detected
L pulse width count	1	1	0	 Measures the width of the "L" period. • Count (measurement) start: Falling edge detected • Count (measurement) end: Rising edge detected
Rising edge to rising edge period count	1	0	0	 Measures the time between rising edges. • Count (measurement) start: Rising edge detected • Count (measurement) end: Rising edge detected
Falling edge to falling edge period count	1	1	1	 Measures the time between falling edges • Count (measurement) start: Falling edge detected • Count (measurement) end: Falling edge detected

Table 18.4f Count Modes (Continued)

Count Mode	MOD2	MOD1	MOD0	Count Operation (w: Pulse width being measured)
Inter-edge pulse width count	0	1	0	 Measures the width between consecutive input edges. • Count (measurement) start: Edge detected • Count (measurement) end: Edge detected
Divided period count	0	1	1	 (Divide by 4 example shown) The input pulses are divided by the divide ratio set in the divide ratio register (DIVR) and the resulting period measured. • Count (measurement) start: Falling edge detected after operation started • Count (measurement) end: End of one period of the divided signal

In all modes, the timer does not count during the time between starting the count and a count start edge being input. After the count start edge is input, the timer is cleared to 0000_H and the timer counts up on each count clock until a count end edge is input.

The following operations are performed when a count end edge is input.

(1) The count end flag (EDIR) in PWCSR is set.

(2) The timer stops counting (unless the timer is restarted at the same time).

(3) In continuous count mode:The timer value (count result) is transferred to PWCR and the count remains stopped until the next count start edge is input.

(4) In single-shot count mode:The timer stops counting (unless the timer is restarted at the same time).

In continuous count mode, the end edge also acts as the next start edge in some modes, including inter-edge pulse width count mode and period count mode.

(e) Minimum Input Pulse Width

Pulses input to the pulse width count input pins (PWC3 to PWC0) must be longer than the minimum input pulse width shown below.

Pulse width:..... 2 machine cycles (≥ 0.125µs for a 16MHz machine clock)

However, input pulses shorter than the above specification may be recognized as valid pulses in some cases.

The PWC inputs do not have a filter function in the MB90580 series. If required, use a filter or similar circuit externally.

(f) Pulse Width/Period Calculation

Calculate the width or period of the measured pulse from the count result read from PWCR after the count ends as follows.

$$T_w = n \times t \div D_{IV}\ (\mu s)$$

- T_w … Measured pulse width or period (µs)
- n … Count result stored in PWCR
- t … Count clock period (µs)
- D_{IV} … Divide ratio set in the divide ratio register (DIVR)
  (Use the value 1 for modes other than divided period count mode.)

(g) Pulse Width/Period Count Range

The range of pulse widths/periods that can be measured depends on the count clock and the divide ratio of the input divider.

The table below lists the measurement range for a 16MHz machine cycle (indicated by ϕ below).

Table 18.4g Pulse Width Count Range

Divide Ratio	DIV1.0	CKS1, 0 = 00 (ϕ/4)	CKS1, 0 = 01 (ϕ/16)	CKS1, 0 = 10 (ϕ/32)
No division	–	0.125µs to 16.38ms [0.25µs]	0.125µs to 65.5ms [1.6µs]	0.2µs to 131ms [3.2µs]
Divide by 4	00_B	0.125µs to 4.10ms [62.5µs]	0.125µs to 16.38ms [0.4µs]	0.2µs to 32.75ms [800ns]
Divide by 16	01_B	0.125µs to 1024µs [15.6ns]	0.125µs to 4.10ms [0.1µs]	0.2µs to 8.19ms [200ns]
Divide by 64	10_B	0.125µs to 256µs [3.91ns]	0.125µs to 1024µs [25.0ns]	0.2µs to 2.048ms [50.0ns]
Divide by 256	11_B	0.125µs to 64µs [0.98ns]	0.125µs to 256µs [6.25ns]	0.2µs to 512ms [12.5ns]

Note: The figures in [ ] indicate the resolution per bit.

(h) Generation of Interrupt Requests

The following two interrupt requests can be generated in pulse width count mode.

(1) Timer overflow interrupt request

If an overflow occurs during counting, the overflow flag is set and, if the overflow interrupt request is enabled, an interrupt request is generated.

(2) Count end interrupt request

When the count end edge is detected, the count end flag (EDIR) in PWCSR is set and, if the count end interrupt request is enabled, an interrupt request is generated.

The count end flag (EDIR) is automatically cleared by reading PWCR.

(i) Flowchart of the Pulse Width Count Operation

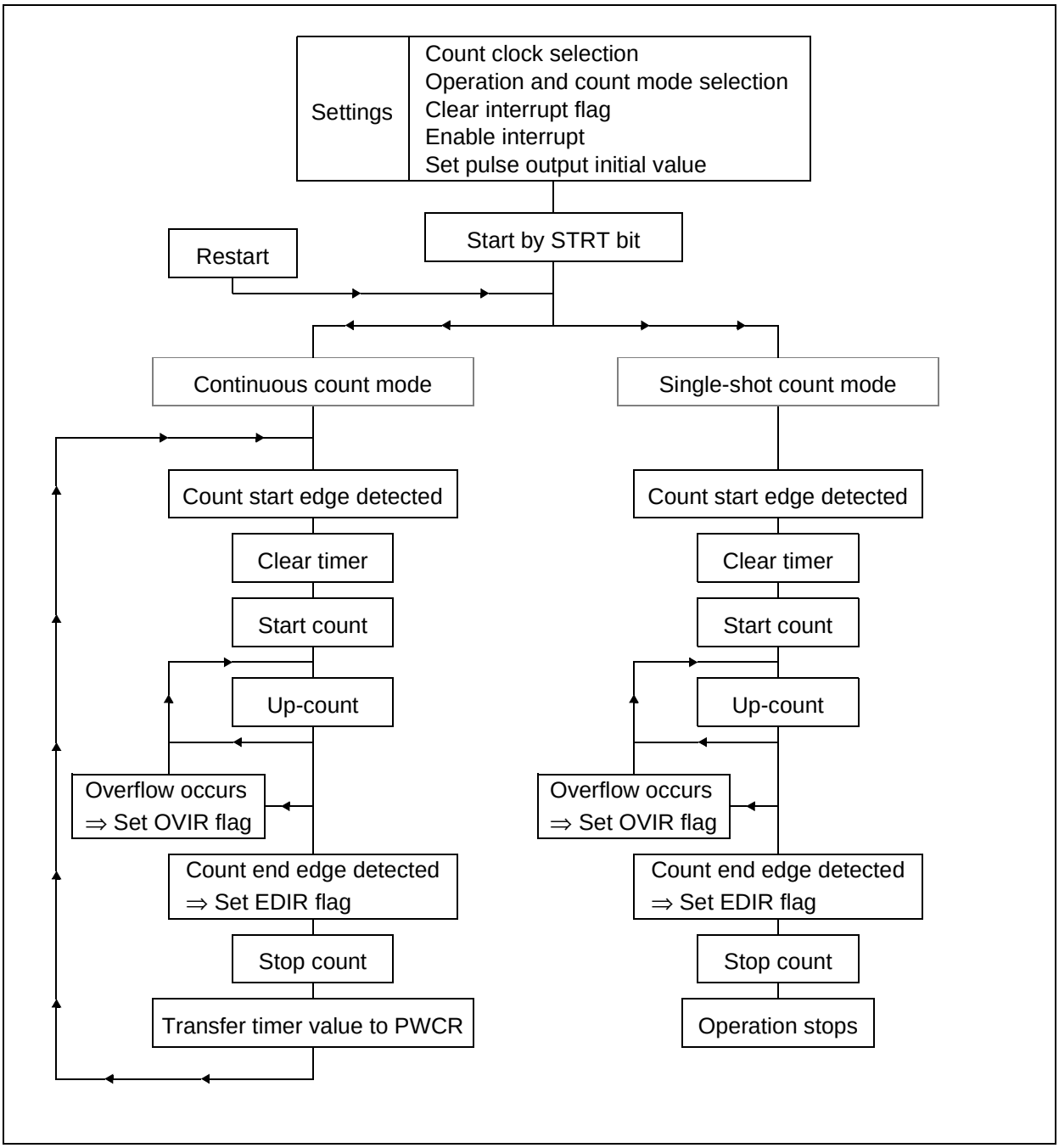


Figure 18.4g Flowchart of Operation in Pulse Width Count Mode

(9) Initial State

- The initial values of each register are:

  PWCSR ⇒ $(00000000\ 00000000)_B$

  PWCR ⇒ $(00000000\ 00000000)_B$

  DIVR ⇒ $(XXXXXX00)_B$

18.5 Precautions

(1) Changing Register Values

Changing the values of the following PWCSR bits when the timer is operating is prohibited. Only change bit values before starting the timer or after operation stops.

[bits 7, 6] CKS1, CKS0: Clock selection bits

[bits 5, 4] PIS1, PIS0: Count input pin selection bits

[bit 3] S/C: Count mode (single-shot or continuous) selection bit

[bits 2, 1, 0] MOD2, MOD1, MOD0: Operating mode and count edge selection bits

Note that the value of the pulse output level indication bit (POUT: bit 8) does not change if the bit is written to when the timer is operating.

Changing the DIVR value when the timer is operating is prohibited. Only change the DIVR value before starting the timer or after operation stops.

(2) Count End Flag in Timer Mode

The value of the count end interrupt request flag (EDIR) in PWCSR has no meaning in timer mode. Therefore, always set the enable bit for the count end interrupt request (EDIE) in PWCSR to "0".

(3) STRT and STOP bits in PWCSR

Note that the meaning of these two bits differs depending on whether they are being read or written (see the register description for details).

Also note that read-modify-write instructions always read the bits as "11_B" regardless of the actual values. Therefore, bit manipulation instructions cannot be used to read the operation state (as the result will always indicate "operating").

However, bit manipulation instructions (such as the bit clear instruction) can be used to write to the STRT or STOP bit to start or stop the timer.

(4) Clearing the Timer

In pulse width count mode, the timer is cleared by the count start edge and therefore the previous data in the timer has no meaning.

(5) Clock Selection Bits

Setting "11_B" to the clock selection bits (CKS1, CKS0: bits 7, 6) in PWCSR is prohibited.

(6) PWCR and Timer Value When Changing Mode

The value in PWCR and the timer value are indeterminate if the timer is set to single-shot mode after forcibly halting operation in reload mode. Therefore, always set a value before using the timer.

The value in PWCR is indeterminate if the timer is set to reload mode after forcibly halting operation in single-shot mode. Therefore, always set a value before using the timer.

When changing from pulse width count mode to timer mode, always set a value to PWCR before starting the timer.

(7) Minimum Input Pulse Width

The following restriction applies to pulses input to the pulse width count input pins.

- Minimum input pulse width: Machine cycle divided by 2 (≥ 0.125µs for a 16MHz machine cycle)
- Maximum input frequency: Machine cycle divided by 4 (≥ 4MHz for a 16MHz machine cycle)

The operation of the timer if pulses of shorter width or higher frequency are input is not guaranteed. If it is possible that such noise may be present on the input signal, use an external filter or similar circuit to suppress the noise.

(8) Divided Period Count Mode

Note that the input pulses are divided when divided period count mode is used in pulse width count mode and therefore the pulse width calculated from the count result is an average value.

(9) Restarting the Timer During Operation

Depending on the timing, the following may occur when the timer is restarted after starting the count operation.

(a) If the restart occurs at the same time as an overflow in reload timer mode:

The timer restarts but the overflow flag (OVIR) is set and the POUT bit inverted. (That is, the same operations are performed as for a normal overflow.)

(b) If the restart occurs at the same time as the count end edge in single-shot pulse width count mode:

The timer restarts and waits for a count start edge but the count end flag (EDIR) is also set.

(c) If the restart occurs at the same time as the count end edge in continuous pulse width count mode:

The timer restarts and waits for a count start edge but the count end flag (EDIR) is also set and the count result at that time is transferred to PWCR.

When restarting the timer while it is still operating, take note of the operation of the flags as described above and perform interrupt and other control accordingly.

(10) Pulse Width Count Mode Using Continuous Count Mode

Note that, when performing continuous counting in this mode, the divider circuit for the internal count clock is not cleared and therefore the number of edges below the count clock is added to the result.

Chapter 19:
Clock Monitor Function

19.1 Outline

Clock Monitor Function is used to output the machine clock to a port pin. This clock output is generated by dividing the machineclock by 2^1 to 2^8.

19.2 Block Diagram

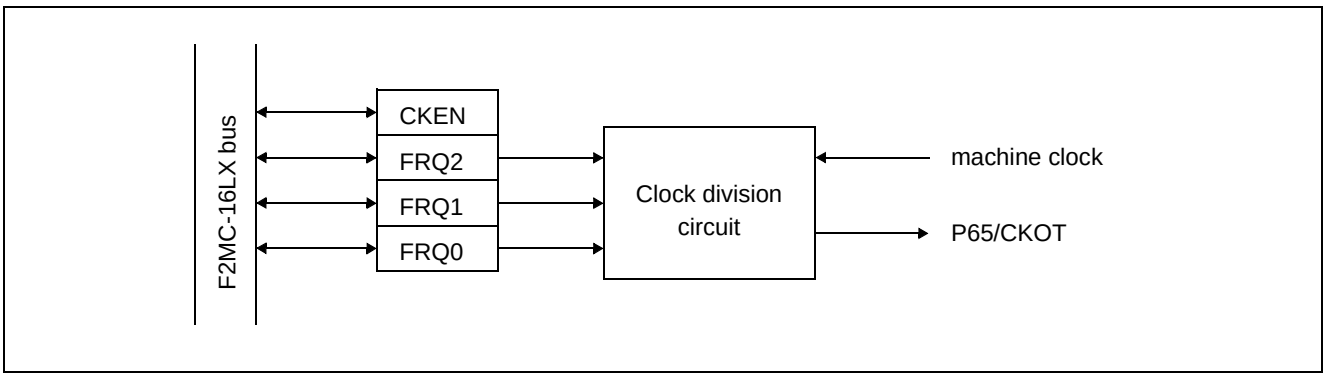


Figure 19.2a Block Diagram of Clock Monitor Function

19.3 Registers and Register Details

Clock Output Enable Register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 00003E$_H$	—	—	—	—	CKEN	FRQ2	FRQ1	FRQ0	CLKR
Read/write ⇨	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(0)	(0)	(0)	(0)	

Figure 19.3a Registers of Clock Monitor Function

19.3.1 Clock output enable register (CLKR)

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : 00003E$_H$	—	—	—	—	CKEN	FRQ2	FRQ1	FRQ0	CLKR
Read/write ⇨	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(-)	(-)	(-)	(-)	(0)	(0)	(0)	(0)	

[bit 3] : CKEN

CKOT output enable bit.

0	Ordinary port
1	CKOT output

[bits 2, 1, and 0] FRQ2, FRQ1, and FRQ0

These bits are used to select the clock output frequency.

FRQ2	FRQ1	FRQ0	Output clock	∅=16 MHz	∅=8 MHz	∅=4 MHz
0	0	0	$\varnothing/2^1$	125 ns	250 ns	500 ns
0	0	1	$\varnothing/2^2$	250 ns	500 ns	1 µs
0	1	0	$\varnothing/2^3$	500 ns	1 µs	2 µs
0	1	1	$\varnothing/2^4$	1 µs	2 µs	4 µs
1	0	0	$\varnothing/2^5$	2 µs	4 µs	8 µs
1	0	1	$\varnothing/2^6$	4 µs	8 µs	16 µs
1	1	0	$\varnothing/2^7$	8 µs	16 µs	32 µs
1	1	1	$\varnothing/2^8$	16 µs	32 µs	64 µs

Chapter 20:
16-Bit I/O Timer

20.1 Outline

The 16-bit I/O timer consists of a 16-bit free-run timer, two output compare modules, and four input capture modules. The count values of this timer are used as the base timer for output compare and input capture. Using this function, two independent waveforms can be output based on 16-bit free-run timer to enable measurement of input pulse withs and external clock cycles.

- Four types of counter clock are available.
- An interrupt can be generated upon a counter value overflow.
- The counter value can be initialized upon a match with compare register 0, depending on the mode.

❑ 16-bit free-run timer (×1)

The 16-bit free-run timer consists of a 16-bit up counter, control register, and prescaler. The 16-bit up counter is used to counting up in synchronization to the machine clock, in which an interrupt factor can be selected from the overflow interrupt and four types of timer intermediate bit interrupt to be operated as an interval timer.

- Four types of counter clock are available.
  Internal clock: ∅/4, ∅/16, ∅/32, ∅/64
- An interrupt can be generated upon a counter value overflow or a match with compare register 0. (Compare match can be used only in an appropriate mode.)
- The counter value can be initialized to '0000H' upon a reset, software clear, or match with compare register 0.

The free-run timer can be used to generating reference timing signals for the input capture (ICU) and output compare (OCU).

❑ Output compare (×2)

The output compare (OCU) consists of two 16-bit compare registers, compare output latch, and control register.

An interrupt request can be generated for each channel upon a match detection by performing time-division comparison between the OCU compare data register setting value and the counter value of the 16-bit free-run timer.

When the 16-bit free-run timer value matches the compare register value, the output level is reversed and an interrupt is issued.

- The four compare registers can be used independently.
  Output pins and interrupt flags corresponding to compare registers
- Output pins can be controlled based on pairs of the four compare registers.
  Output pins can be reversed by using the four compare registers.
- Initial values for output pins can be set.
- Interrupts can be generated upon a compare match.

❑ Input capture (×4)

The input capture (ICU) generates an interrupt request to the CPU simultaneously with a storing operation of current counter value of the 16-bit free-run timer to the ICU data register (IPCP) upon an input of a trigger edge to the external pin.

There are four sets (four channels) of the input capture external pins and ICU data registers (ICDR), enabling measurements of maximum of four events.

- The input capture has four sets of external pins (IN0 to IN3) and ICU registers (IPCP0~3), enabling measurements of maximum of four events.
- A trigger edge direction can be selected from rising/falling/both edges.
- The input capture can be set to generate an interrupt request at the storage timing of the counter value of the 16-bit free-run counter to the ICU data register (IPCP).
- The input compare conforms to the extended intelligent I/O service (EI^2OS).
- The input capture function is suited for measurements of intervals (frequencies) and pulse-widths.

A reset clears the timer counter value for the 16-bit free-run timer to all zeroes.

20.2 Block Diagram

20.2.1 Overall Block Diagram of 16-bit I/O Timer

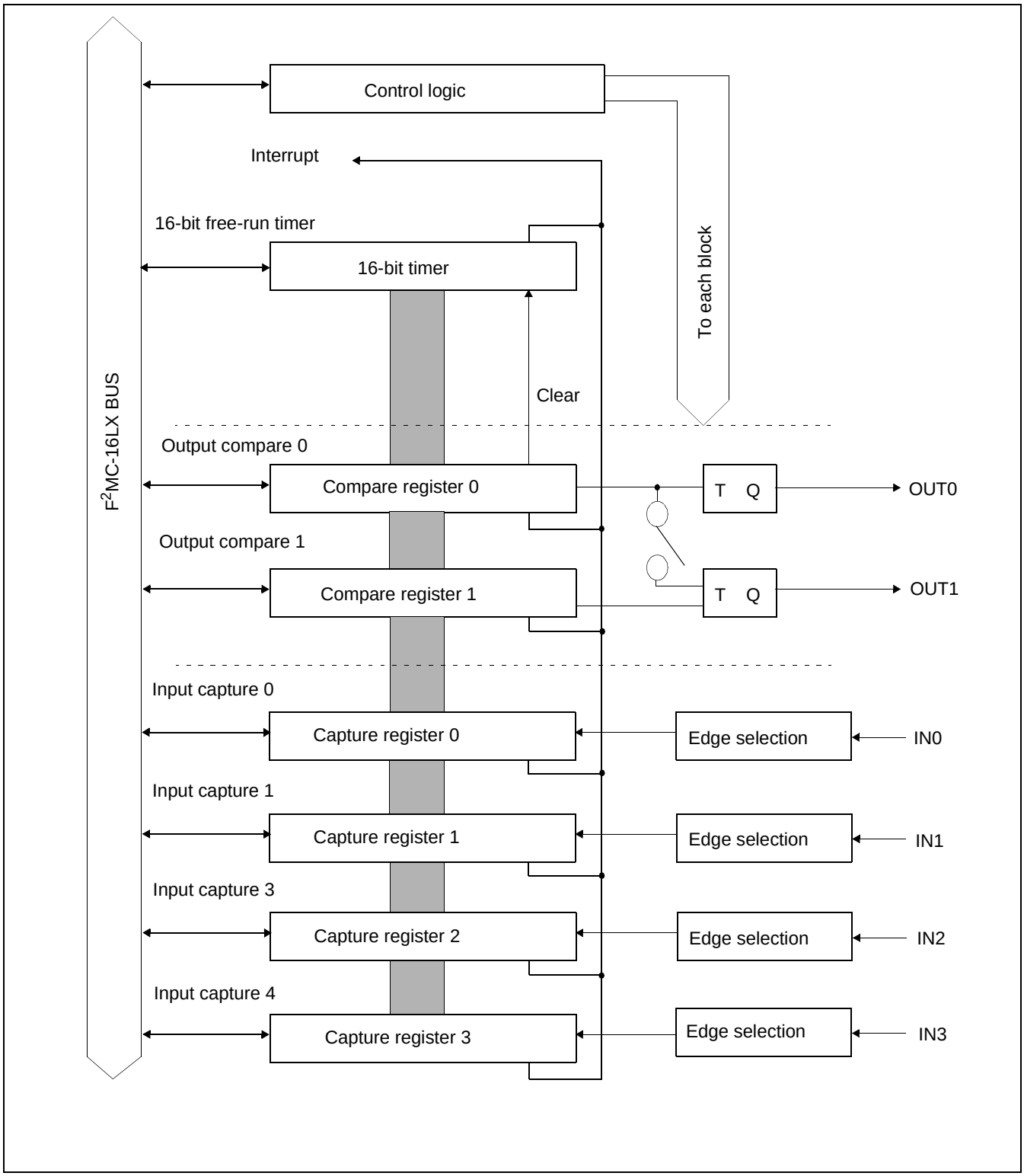


Figure 20.2.1a Overall Block diagram of 16-bit I/O Timer

20.2.2 Block Diagram of 16-bit free-run timer

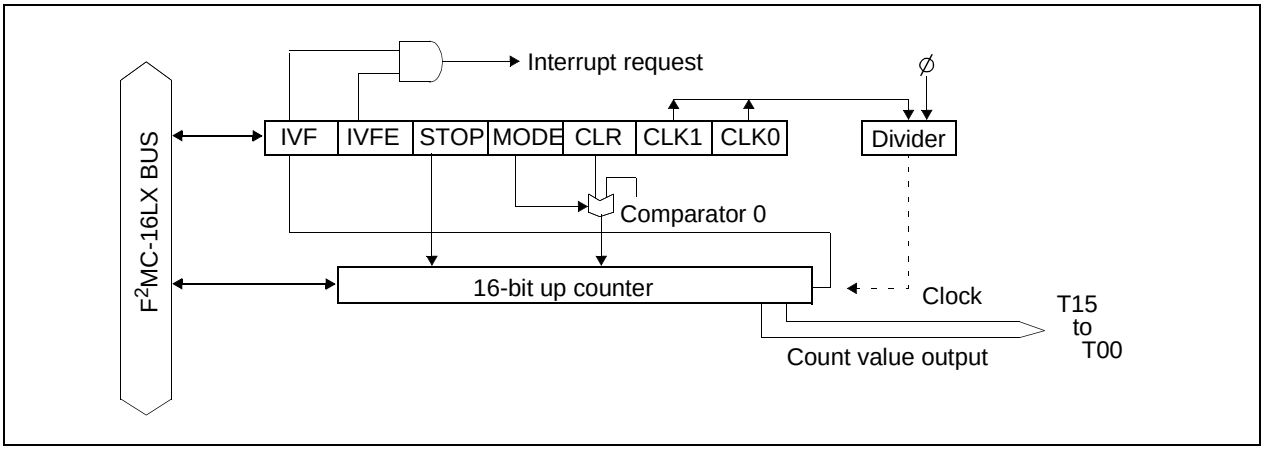


Figure 20.2.2a Block diagram of 16-bit free-run timer

20.2.3 Block Diagram of Output Comparison

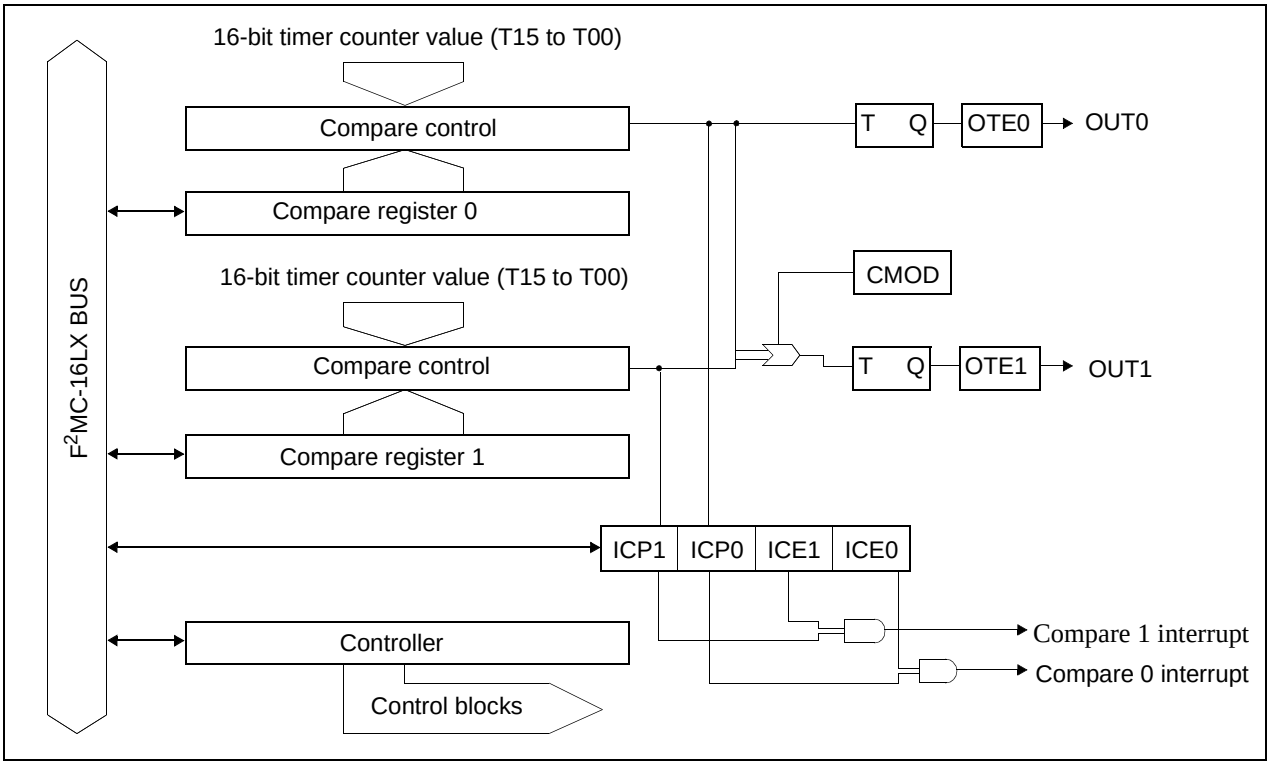


Figure 20.2.3a Block diagram of Output Comparison

20.2.4 Block Diagram of Input Capture

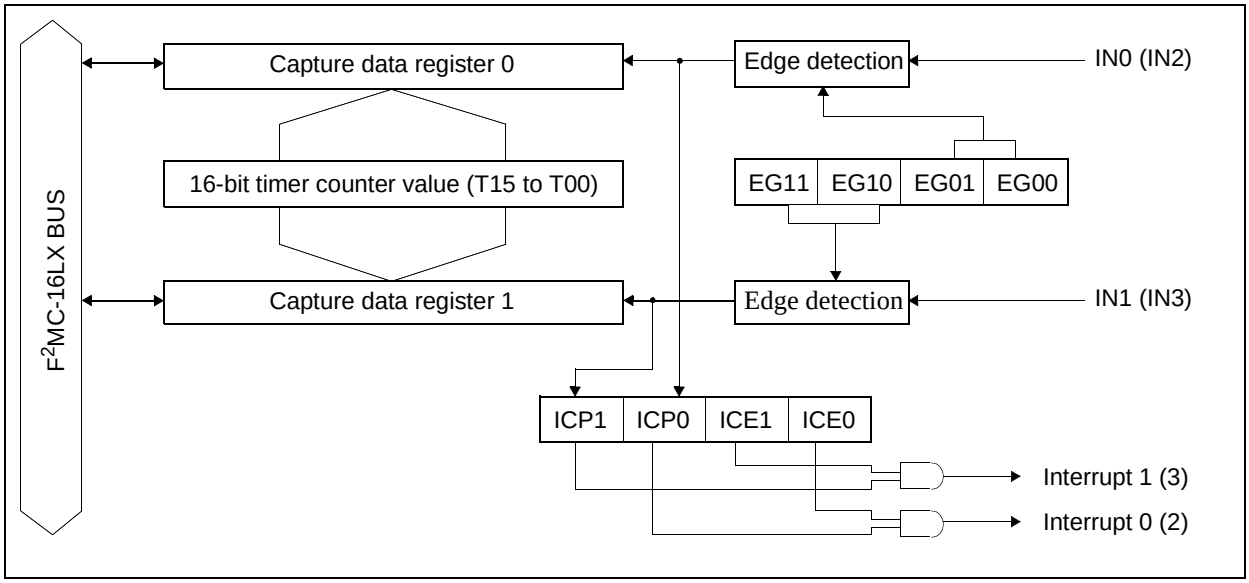


Figure 20.2.4a Block diagram of Input Capture

20.3 Registers and Register Details

20.3.1 16-bit free-run timer

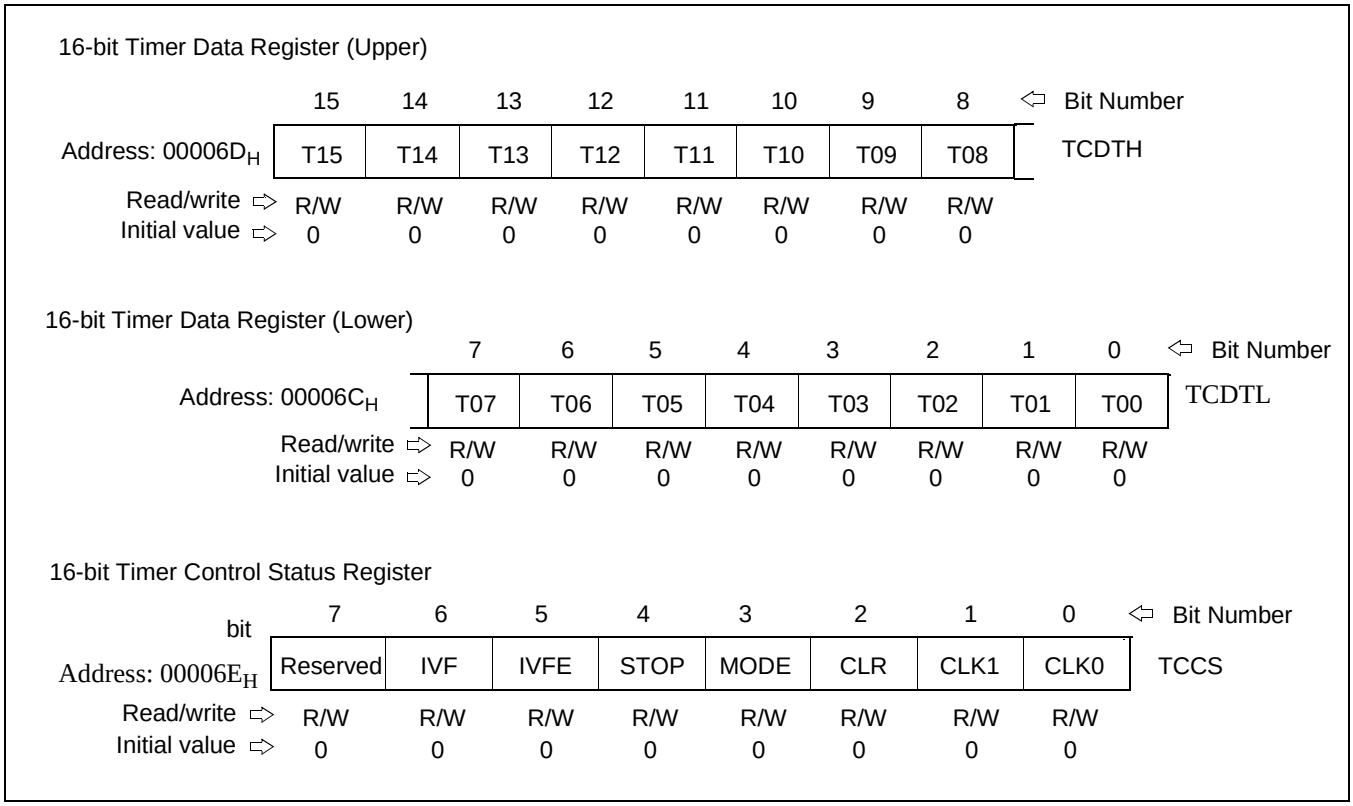
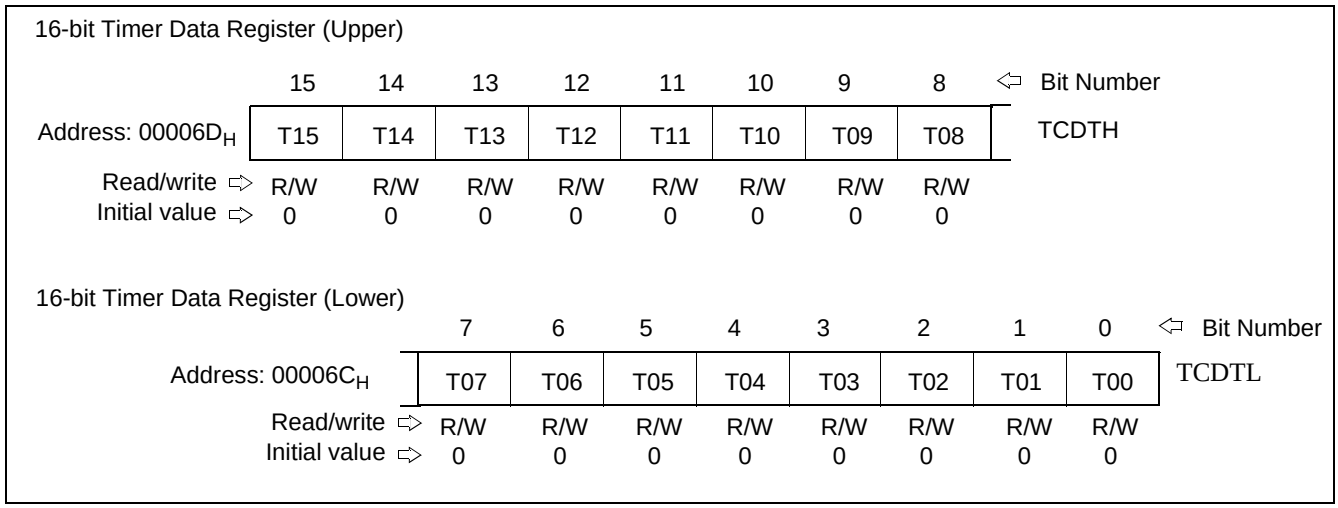


Figure 20.3.1a Registers of 16-bit free-run timer

20.3.1.1 16-bit free-run timer data register



The data register can read the count value of the 16-bit free-run timer. The counter value is cleared to ’0000’ upon a reset. The timer value can be set by writing a value to this register. However, ensure that the value is written while the operation is stopped (STOP=1). The data register must be accessed in word access mode.

The 16-bit free-run 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 (This can be performed only in an appropriate mode.)

20.3.1.2 16-bit free-run timer control status register

16-bit Timer Control Status Register

bit	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: $00006E_H$	Reserved	IVF	IVFE	STOP	MODE	CLR	CLK1	CLK0	TCCS
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	

[bit 7] Reserved bit

Always write ’0’ to this bit.

[bit 6] IVF

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

If the 16-bit free-run timer overflows, or if the counter is cleared by a match with compare register 0 in a certain mode, ’1’ is written 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 5] IVFE

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

0	Interrupt disabled (default)
1	Interrupt enabled

[bit 4] STOP

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

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

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

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

[bit 3] MODE

The MODE bit is used to set the initialization condition of the 16-bit free-run timer.

When ’0’ is set, the counter value can be initialized by a 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 a 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 counter value is initialized where the count value is changed.

[bit 2] CLR

The CLR bit initializes the operating 16-bit free-run 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 where 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 1 and 0] 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.

CLK1	CLK0	Count clock	∅=16 MHz	∅=8 MHz	∅=4 MHz	∅=1 MHz
0	0	∅/4	0.25 µs	0.5 µs	1 µs	4 µs
0	1	∅/16	1 µs	2 µs	4 µs	16 µs
1	0	∅/64	4 µs	8 µs	16 µs	64 µs
1	1	∅/256	16 µs	32 µs	64 µs	256 µs

* ∅ = Machine clock

20.3.2 Output comparison

The output compare module consists of 16-bit compare registers, compare output pins, and control register. If the value written to the compare register of this module matches the 16-bit free-run timer value, the output level of the pin can be reversed and an interrupt can be issued.

- Two compare registers exist that can be used independently. Depending on the setting, the two compare registers can be used to control pin outputs.
- The initial value for the pin output can be specified.
- An interrupt can be issued upon a match as a result of comparison.

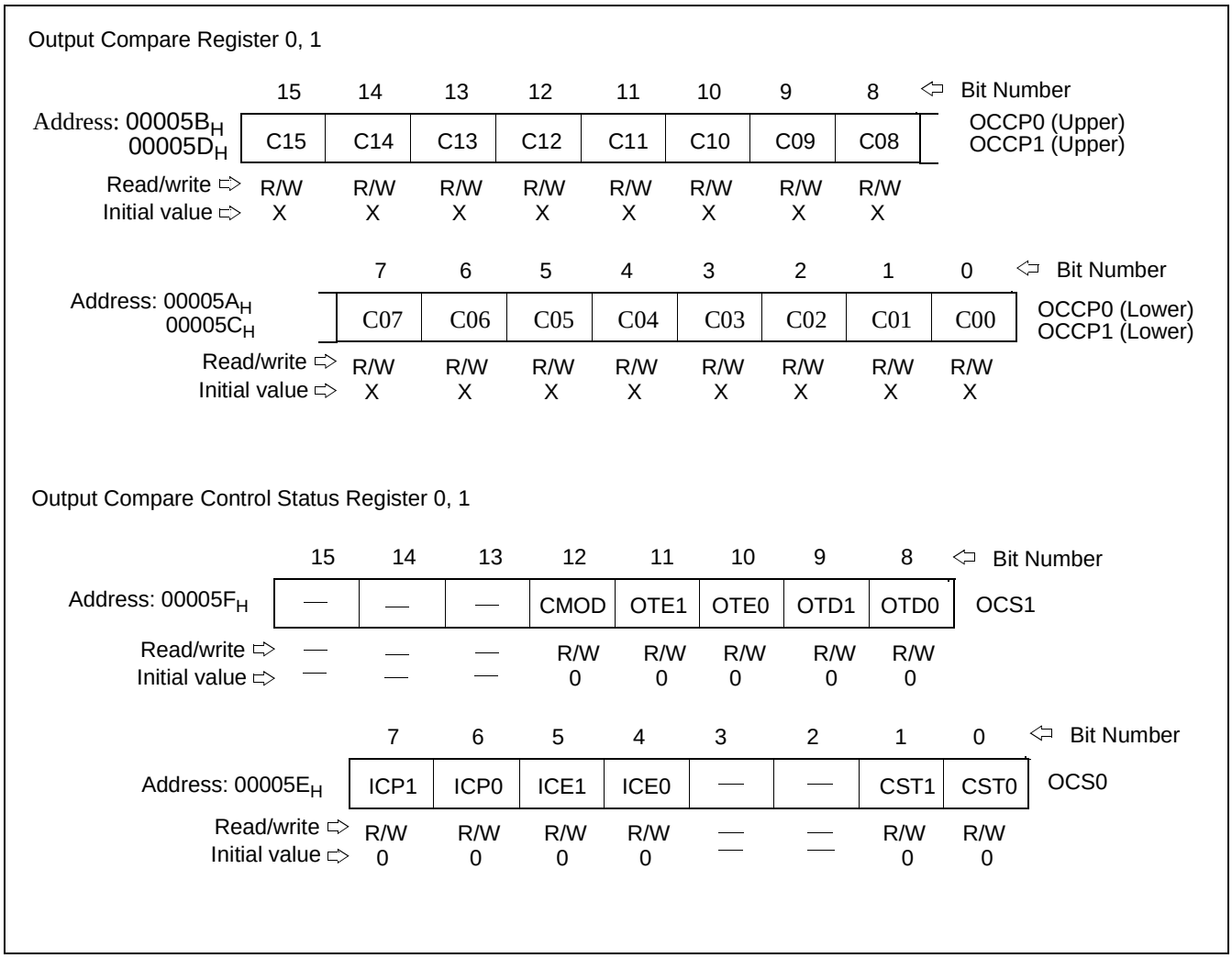


Figure 20.3.2a Registers of output comparsion

20.3.2.1 Compare register

Output Compare Register 0, 1

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: $00005B_H$ / $00005D_H$	C15	C14	C13	C12	C11	C10	C09	C08	OCCP0 (Upper) / OCCP1 (Upper)
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	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: $00005A_H$ / $00005C_H$	C07	C06	C05	C04	C03	C02	C01	C00	OCCP0 (Lower) / OCCP1 (Lower)
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 16-bit compare register is compared with the 16-bit free-run timer. Since the initial register value is undefined, set a value before enabling the register. This register must be accessed in word mode. When the value of this register matches that of the 16-bit free-run timer, a compare signal is generated and the output compare interrupt flag is set. If output is enabled, the output level corresponding to the compare register is reversed.

20.3.2.2 Control status register

Output Compare Control Status Register 0, 1

	15	14	13	12	11	10	9	8	⇦ Bit Number
Address: 00005F$_H$	—	—	—	CMOD	OTE1	OTE0	OTD1	OTD0	OCS1
Read/write ⇨	—	—	—	R/W	R/W	R/W	R/W	R/W	
Initial value ⇨	—	—	—	0	0	0	0	0	

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 00005E$_H$	ICP1	ICP0	ICE1	ICE0	—	—	CST1	CST0	OCS0
Read/write ⇨	R/W	R/W	R/W	R/W	—	—	R/W	R/W	
Initial value ⇨	0	0	0	0	—	—	0	0	

[bits 15, 14, and 13] Unused bits

[bit 12] CMOD

CMOD is used to switch the pin output level reverse mode upon a match while pin output is enabled (OTE1=1 or OTE0=1).

- When CMOD=0 (default), the output level of the pin corresponding to the compare register is reversed.

  OUT0: The level is reversed upon a match with compare register 0.
  OUT1: The level is reversed upon a match with compare register 1.
- When CMOD=1, the output level is reversed for compare register 0 in the same manner as for CMOD=0. The output level of the pin corresponding to compare register 1 (OUT1), however, is reversed upon a match with compare register 0 or 1. If compare registers 0 and 1 have the same value, the same operation as with a single compare register is performed.

  OUT0: The level is reversed upon a match with compare register 0.
  OUT1: The level is reversed upon a match with compare register 0 or 1.

[bits 11 and 10] OTE1 and OTE0

These bits are used to enable output compare pin output. The initial value for these bits is '0.'

0	General-purpose port (default)
1	Output compare pin output

* OTE1: Corresponds to output compare 1/3 OTE0: Corresponds to output compare 0/2

* OUT0/1 are multiplexed with P94/TOUT1 and P95/TOUT2 respectively. Whne both output capture and reload timer output are enabled. OUT0/OUT1 get the higher priority.

[bits 9 and 8] OTD1 and OTD0

These bits are used to change the pin output level when the output compare pin output is enabled. The initial value of the compare pin output is '0.' Ensure that the compare operation is stopped before a value is written. When read, these bits indicate the output compare pin output value.

0	Sets '0' for the compare pin output. (default)
1	Sets '1' for the compare pin output.

* OTD1: Corresponds to output compare 1/3 OTD0: Corresponds to output compare 0/2

[bits 7 and 6] ICP1 and ICP0

These bits are used as output compare interrupt flags. ’1’ is written to these bits when the compare register value matches the 16-bit free-run timer value. While the interrupt request bits (ICE1 and ICE0) are enabled, an output compare interrupt occurs when the ICP1 and ICP0 bits are set. These bits are cleared by writing ’0.’

Writing ’1’ has no effect. ’1’ is always read by a read-modify-write instruction.

0	No compare match (default)
1	Compare match

* ICP1: Corresponds to output compare 1/3 ICP0: Corresponds to output compare 0/2

[bits 5 and 4] ICE1 and ICE0

These bits are used as output compare interrupt enable flags. While the ’1’ is written to these bits, an output compare interrupt occurs when an interrupt flag (ICP1 or ICP0) is set.

0	Output compare interrupt disabled (default)
1	Output compare interrupt enabled

* ICE1: Corresponds to output compare 1/3 ICE0: Corresponds to output compare 0/2

[bits 3 and 2] Unused bits

[bits 1 and 0] CST1 and CST0

The*se bits are used to enable the comparison with 16-bit free-run timer.

0	Compare operation disabled (default)
1	Compare operation enabled

Ensure that a value is written to the compare register before the compare operation is enabled.

* CST1: Corresponds to output compare 1/3 CST0: Corresponds to output compare 0/2

Note: Since output compare is synchronized with the 16-bit free-run timer clock, stopping the 16-bit free-run timer stops compare operation.

20.3.3 Input capture

This module detects a rising or falling edge or both edges of an externally input signal and stores the 16-bit free-run timer value in a register. In addition, this module can generate an interrupt upon detection of an edge. The input capture module consists of an input capture data register and a control register. Each input capture has a corresponding external input pin.

- 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.

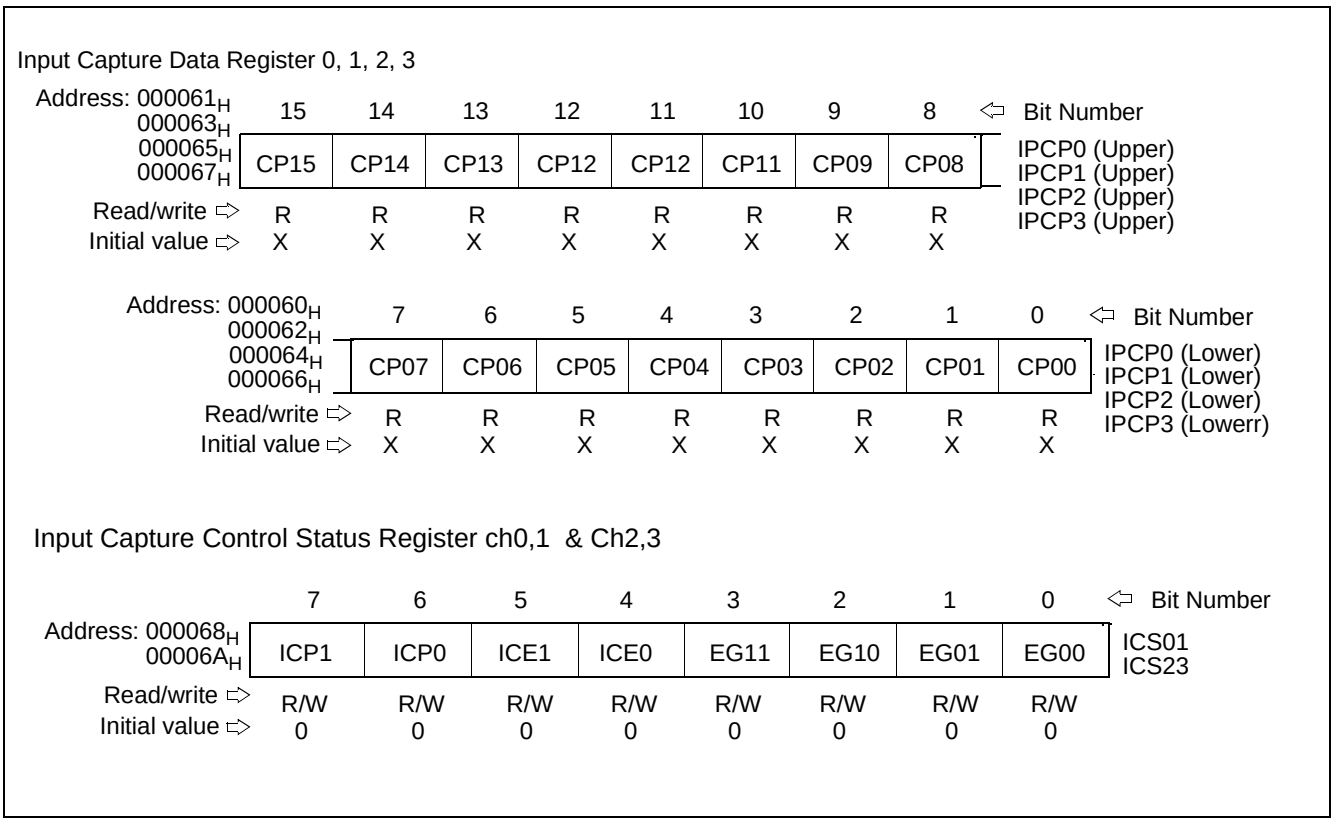


Figure 20.3.3a Register of input capture

20.3.3.1 Input capture data register

Input Capture Data Register 0, 1, 2, 3

Address: 000061_H / 000063_H / 000065_H / 000067_H	15	14	13	12	11	10	9	8	⇦ Bit Number
	CP15	CP14	CP13	CP12	CP12	CP11	CP09	CP08	IPCP0 (Upper) IPCP1 (Upper) IPCP2 (Upper) IPCP3 (Upper)
Read/write ⇨	R	R	R	R	R	R	R	R	
Initial value ⇨	X	X	X	X	X	X	X	X	

Address: 000060_H / 000062_H / 000064_H / 000066_H	7	6	5	4	3	2	1	0	⇦ Bit Number
	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	IPCP0 (Lower) IPCP1 (Lower) IPCP2 (Lower) IPCP3 (Lowerr)
Read/write ⇨	R	R	R	R	R	R	R	R	
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.)

20.3.3.2 Control status register

Input Capture Control Status Register ch0,1 & Ch2,3

	7	6	5	4	3	2	1	0	⇦ Bit Number
Address: 000068_H $00006A_H$	ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	ICS01 ICS23
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	

[bits 7 and 6] ICP1 and ICP0

These bits are used as input capture interrupt flags. '1' is written 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 '0' has no effect. '1' is always read by a read-modify-write instruction.

0	No valid edge detection (default)
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 '1' is written to these bits, 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 an external input. 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.

20.4 Operations

20.4.1 16-bit free-run timer

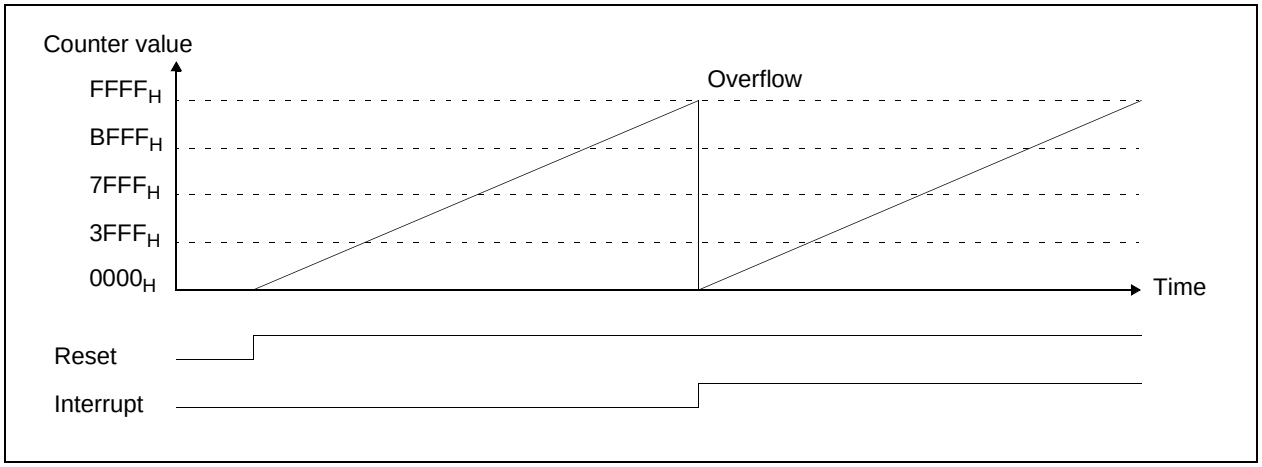
The 16-bit free-run timer starts counting from counter value ’0000’ after the reset is released. The counter value is used as the reference time for the 16-bit output compare and 16-bit input capture operations.

The counter value is cleared in the following conditions:

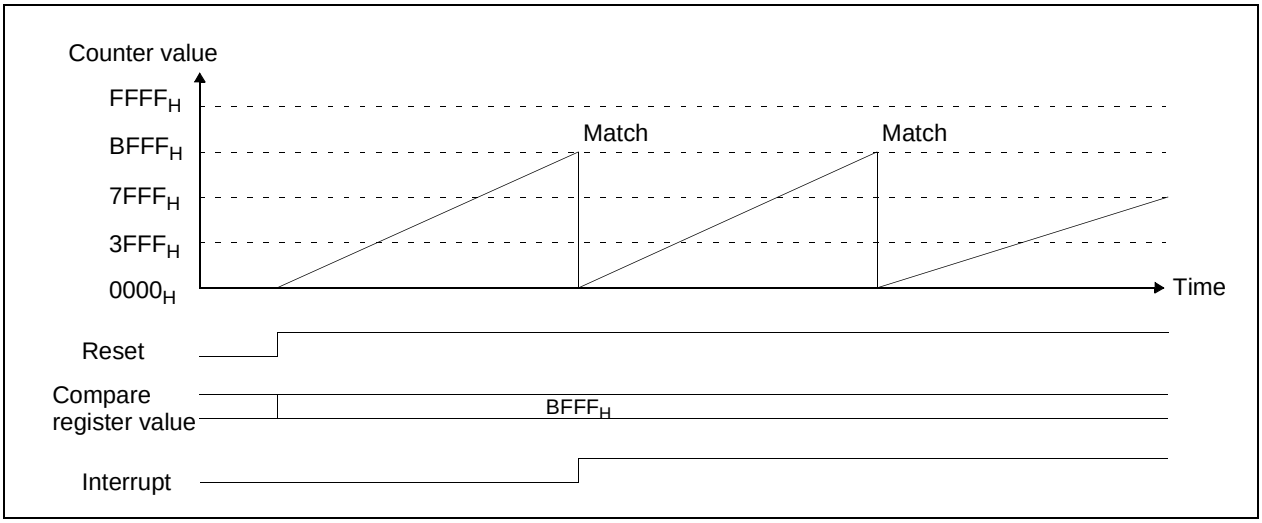
- When an overflow occurs.
- When a match with output compare register 0 occurs. (This depends on the mode.)
- When '1' is written to the CLR bit of the TCCS register during operation.
- When '0000' is written to the TCDC register during stop.
- Reset

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

■ Clearing the counter by an overflow



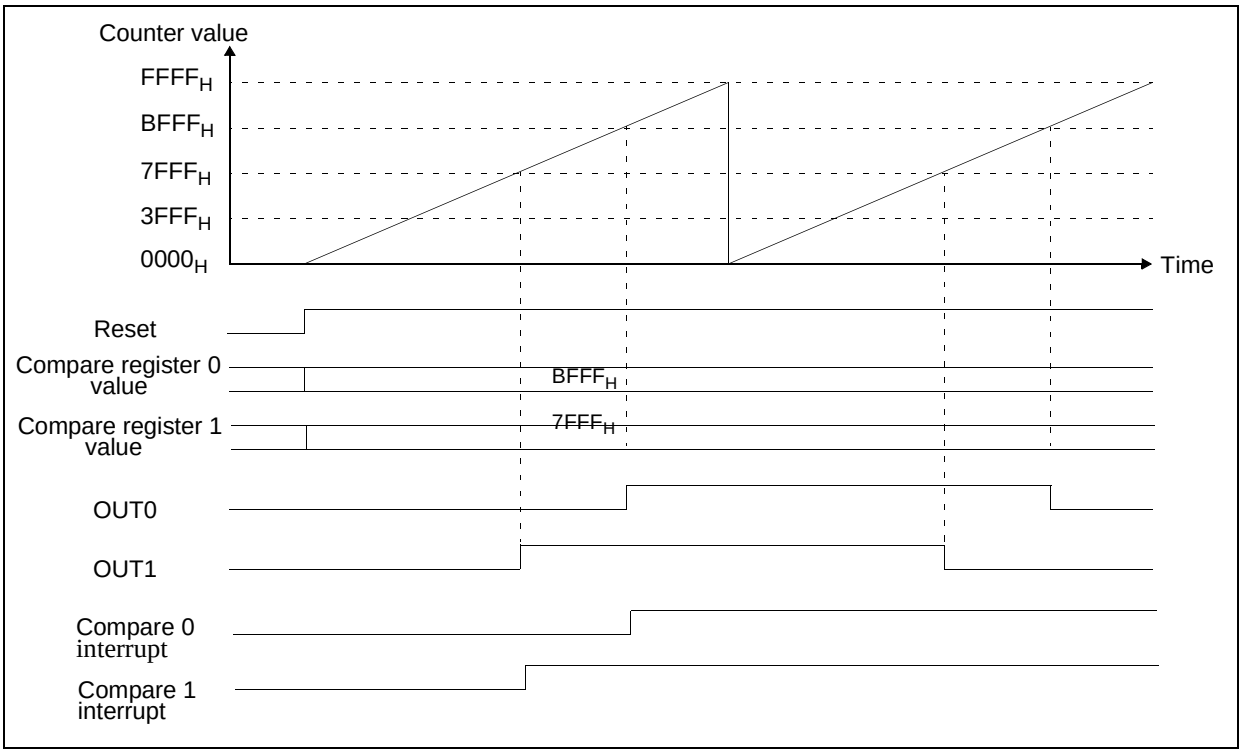
■ Clearing the counter upon a match with output compare register 0



20.4.2 16-bit output compare

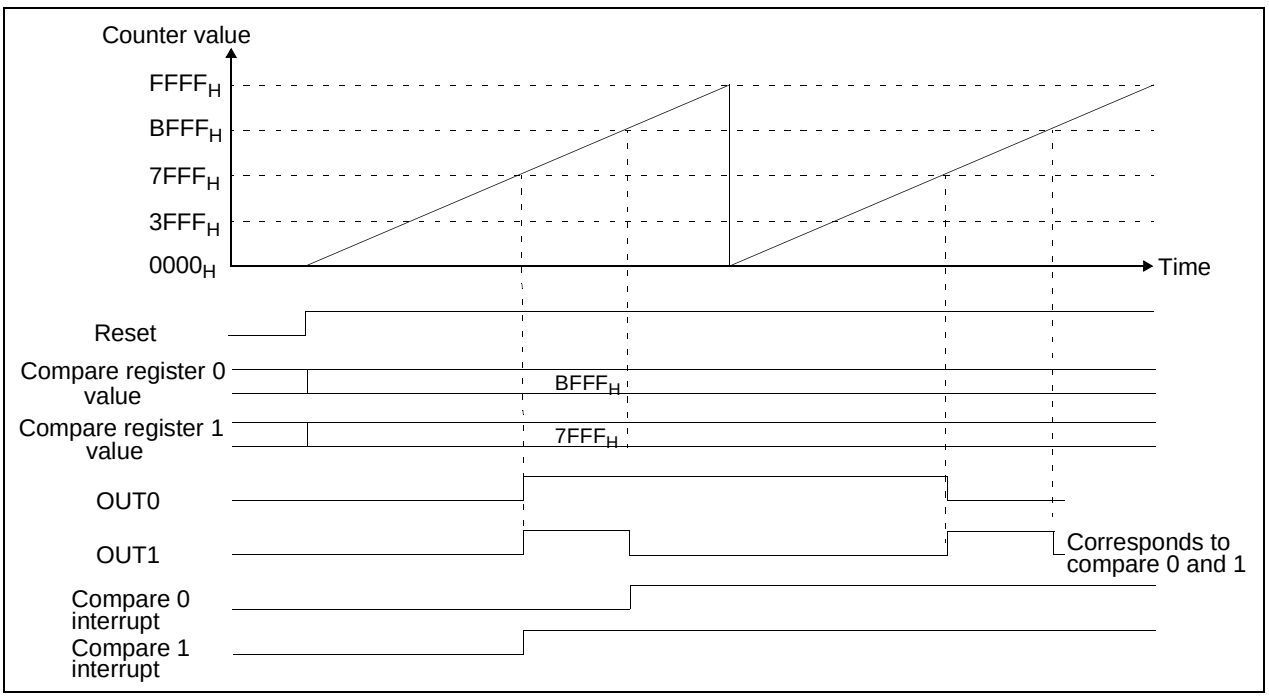
In 16-bit output compare operation, an interrupt request flag can be set and the output level can be reversed when the specified compare register value matches the 16-bit free-run timer value.

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



20.4.3 16-bit input capture

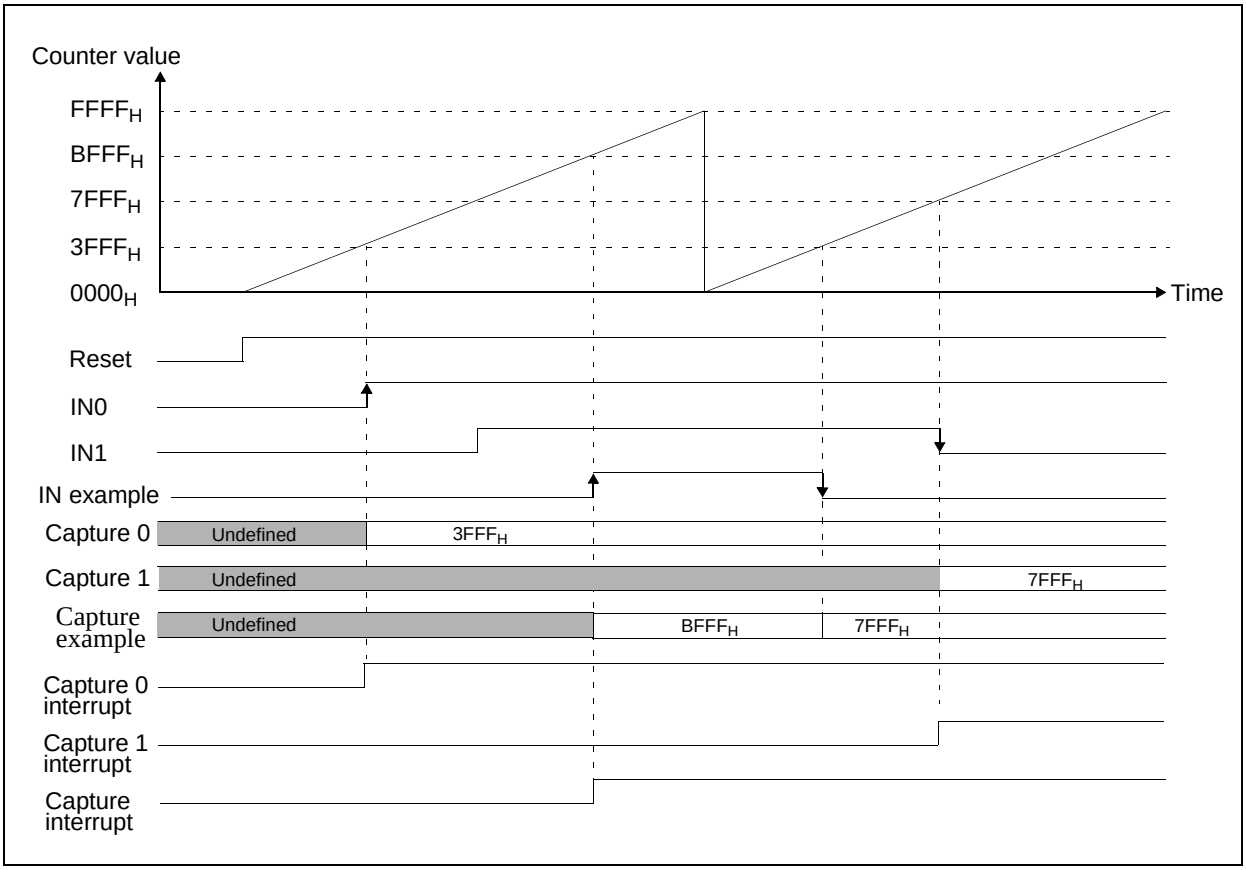
In 16-bit input capture operation, an interrupt can be generated upon detection of a specified valid edge, fetching the 16-bit free-run timer value and writing it to the capture register.

■ Sample input capture fetch timing

Capture 0: Rising edge

Capture 1: Falling edge

Capture example: Both edges

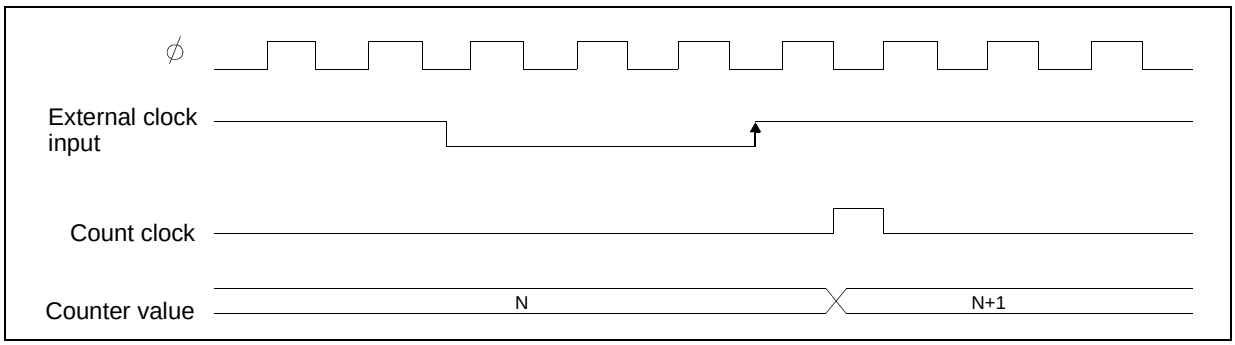


20.5 Timing

20.5.1 16-bit free-run timer count timing

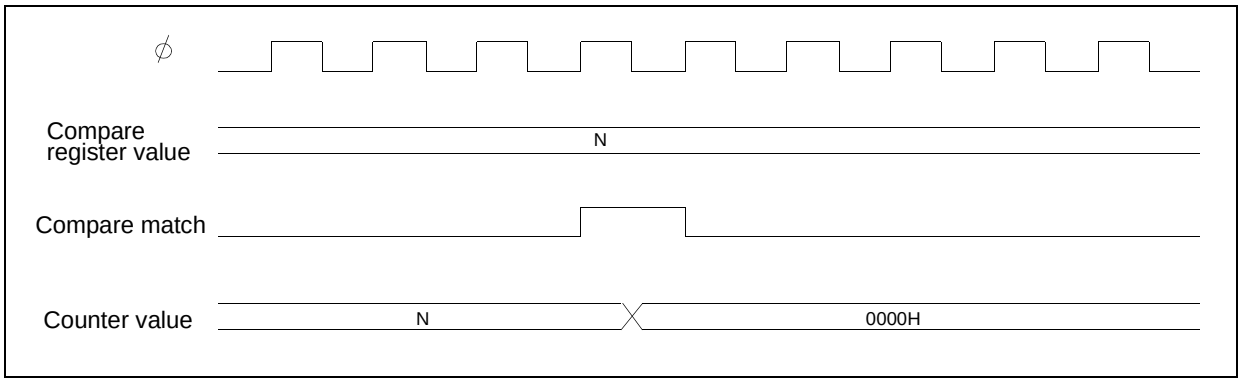
The 16-bit free-run timer is incremented based on the input clock (internal or external clock). When external clock is selected, the 16-bit free-run timer is incremented at the rising edge.

■ Free-run timer count timing



The counter can be cleared upon a reset, software clear, or a match with 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-run timer clear timing (match with compare register 0)

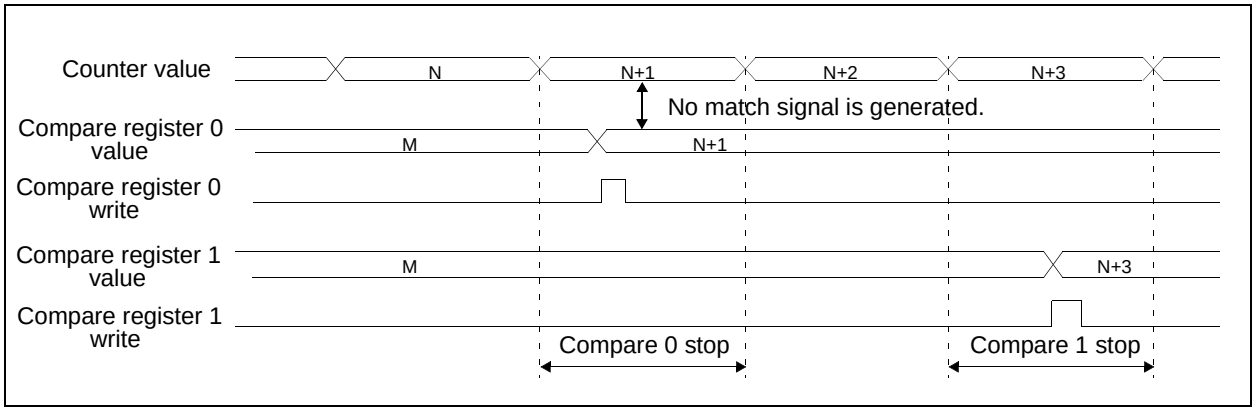


20.5.2 Output compare timing

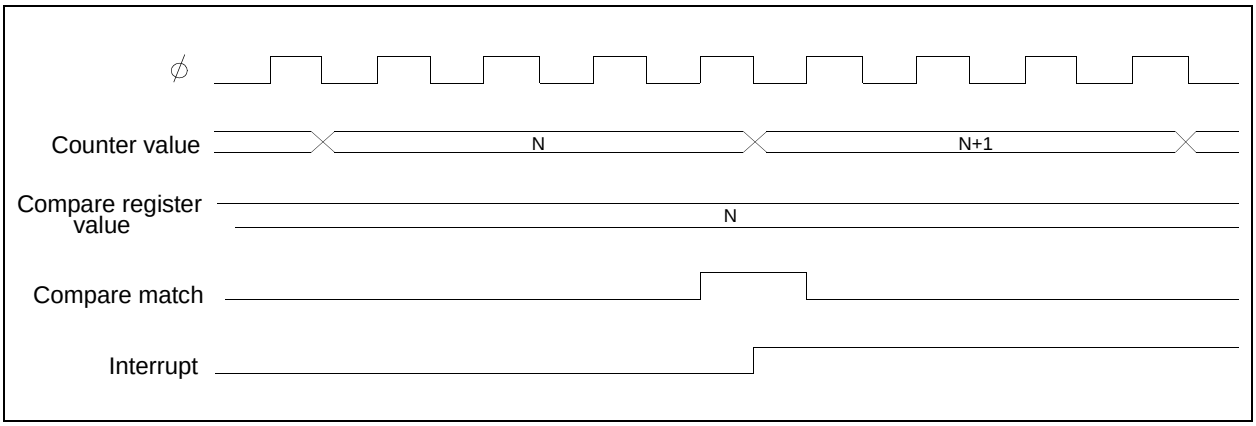
In output compare operation, a compare match signal is generated when the free-run timer value matches the specified compare register value. The output value can be reversed and an interrupt can be issued. The output reverse timing upon a compare match is synchronized with the counter count timing.

■ Compare operation upon update of compare register

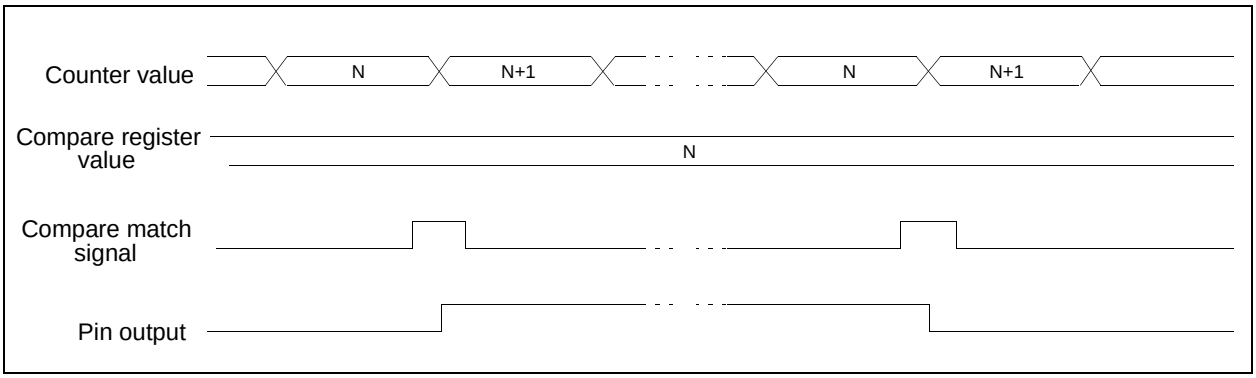
When the compare register is updated, comparison with the counter value is not performed.



■ Interrupt timing

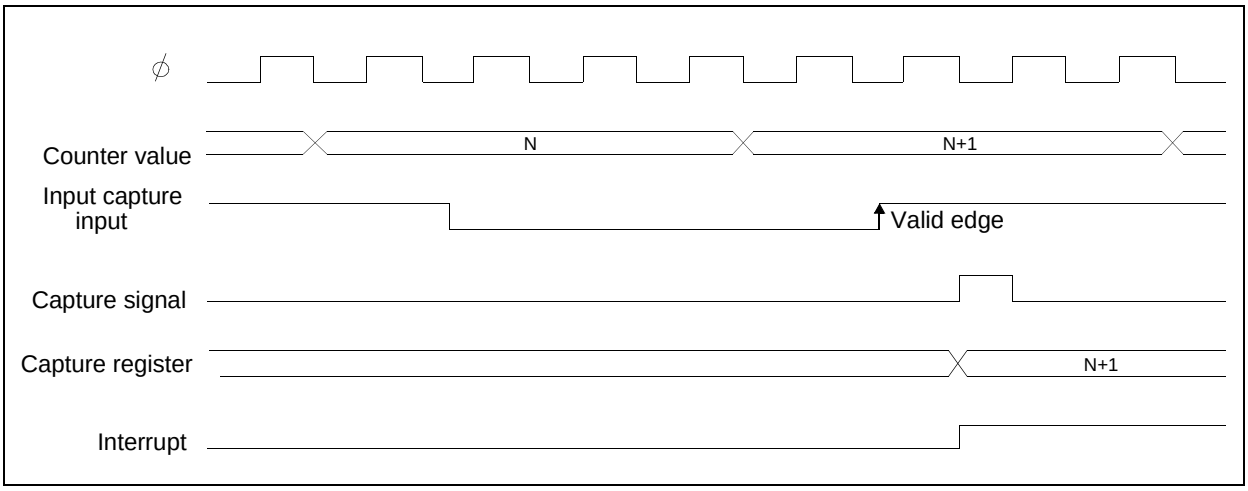


■ Output pin change timing



20.5.3 Input capture input timing

■ Capture timing for input signals



Chapter 21:
ROM Correction Module

21.1 Outline

When the setting of the address is the same 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 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.

21.2 Block Diagram

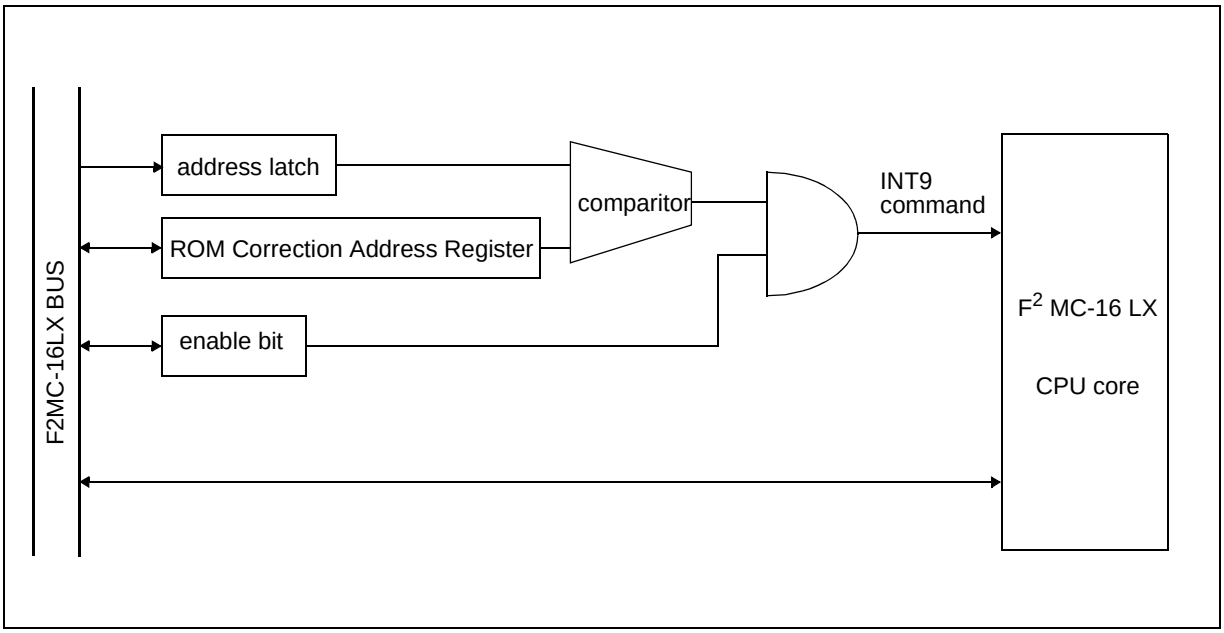


Figure 21.2a Block Diagram of ROM Correction Module

21.3 Registers and Register Details

Program Address Detect Register 0/1

	byte	byte	byte	access	initial value
PADR0 1FF2H/1FF1H/1FF0H				R/W	undefined
PADR1 1FF5H/1FF4H/1FF3H				R/W	undefined

Program Address Detect Control Status Register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : $009E_H$	—	—	—	—	AD1E	—	AD0E	—	PACSR
Read/write ⇨	(–)	(–)	(–)	(–)	(R/W)	(–)	(R/W)	(–)	
Initial value ⇨	(–)	(–)	(0)	(0)	(0)	(0)	(0)	(0)	

Figure 21.3a Registers of ROM Correction Module

21.3.1 Program Address Detect Register 0/1 (PADR0/PADR1)

These registers hold the addresses for the comparison with program counter. If there is an agreement and when the corresponding ADCSR interrupt enable bit is at ‘ 1’, this module demands the CPU to execute the INT9 instruction.

If the corresponding interrupt enalble bit is ‘ 0’, nothing will occur even there is a match.

Program Address Detect Register 0/1

	byte	byte	byte	access	initial value
PADR0 1FF2H/1FF1H/1FF0H				R/W	undefined
PADR1 1FF5H/1FF4H/1FF3H				R/W	undefined

The correspondance to the PACSR will be as follows.

ROM correction register	Compare enable bit
PADR0	AD0E
PADR1	AD1E

21.3.2 Program Address detect Control Status Register (PACSR)

Program Address Detect Control Status Register

	7	6	5	4	3	2	1	0	⇦ Bit number
Address : $009E_H$	—	—	—	—	AD1E	—	AD0E	—	PACSR
Read/write ⇨	(–)	(–)	(–)	(–)	(R/W)	(–)	(R/W)	(–)	
Initial value ⇨	(–)	(–)	(0)	(0)	(0)	(0)	(0)	(0)	

This register provides control bits and status bit for the ROM correction function.

[bit 5~4]

These are the reserved bits, be sure to write ‘ 0’.

[bit 3]: AD1E (Compare Enable 1)

This is the ADR1 enable bit.

When this bit is at ‘ 1’, this module compares the PADR1 register and the program counter. If there is an agreement, the INT9 instruction is sent to the CPU.

[bit 2]:

This is a 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]:

This is a reserved bit.

21.4 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 instructions.

Note: When the address detection register and the program counter are in agreement, the internal data bus content will be forced to be ‘ 01H’, so interrupt INT9 will be executed. Before changing the content of the address detect 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.

21.5 Application Example

(1) System Structure

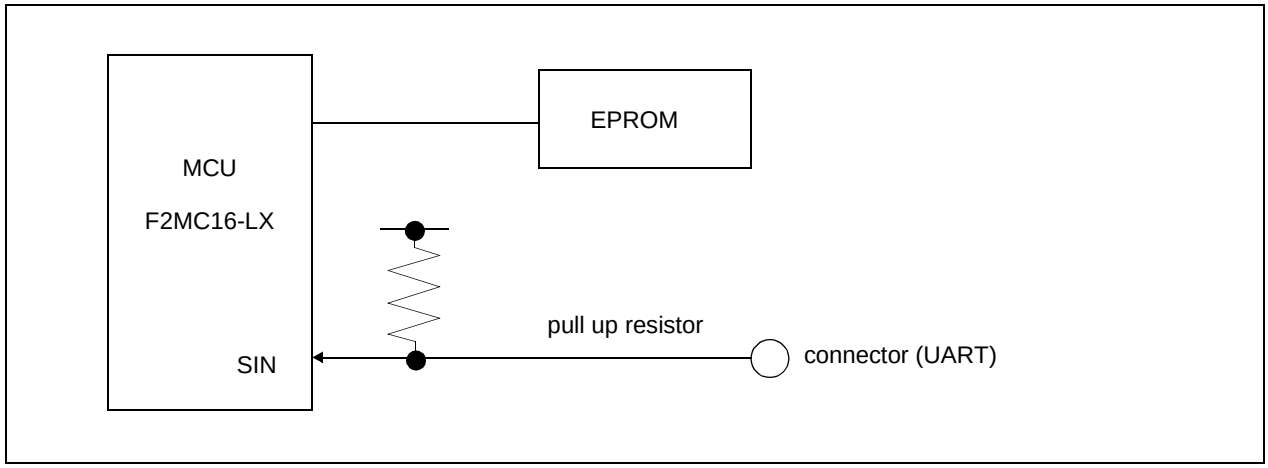


Figure 21.5a System Structure Example

(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. 0 (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 will write that information into the EEPROM.

(5) Reset Sequence

After resetting, the MCU reads the content of the EEPROM. If the byte number of the corrected program is not ‘ 0’, the body of the corrected program will be read from the EEPROM and written in the RAM. Then the MCU sets the correction address either on PADR0 or on PADR1 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 INT9 service routine looks 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 stack program couter value. The information stacked during interrupt will be discarded.

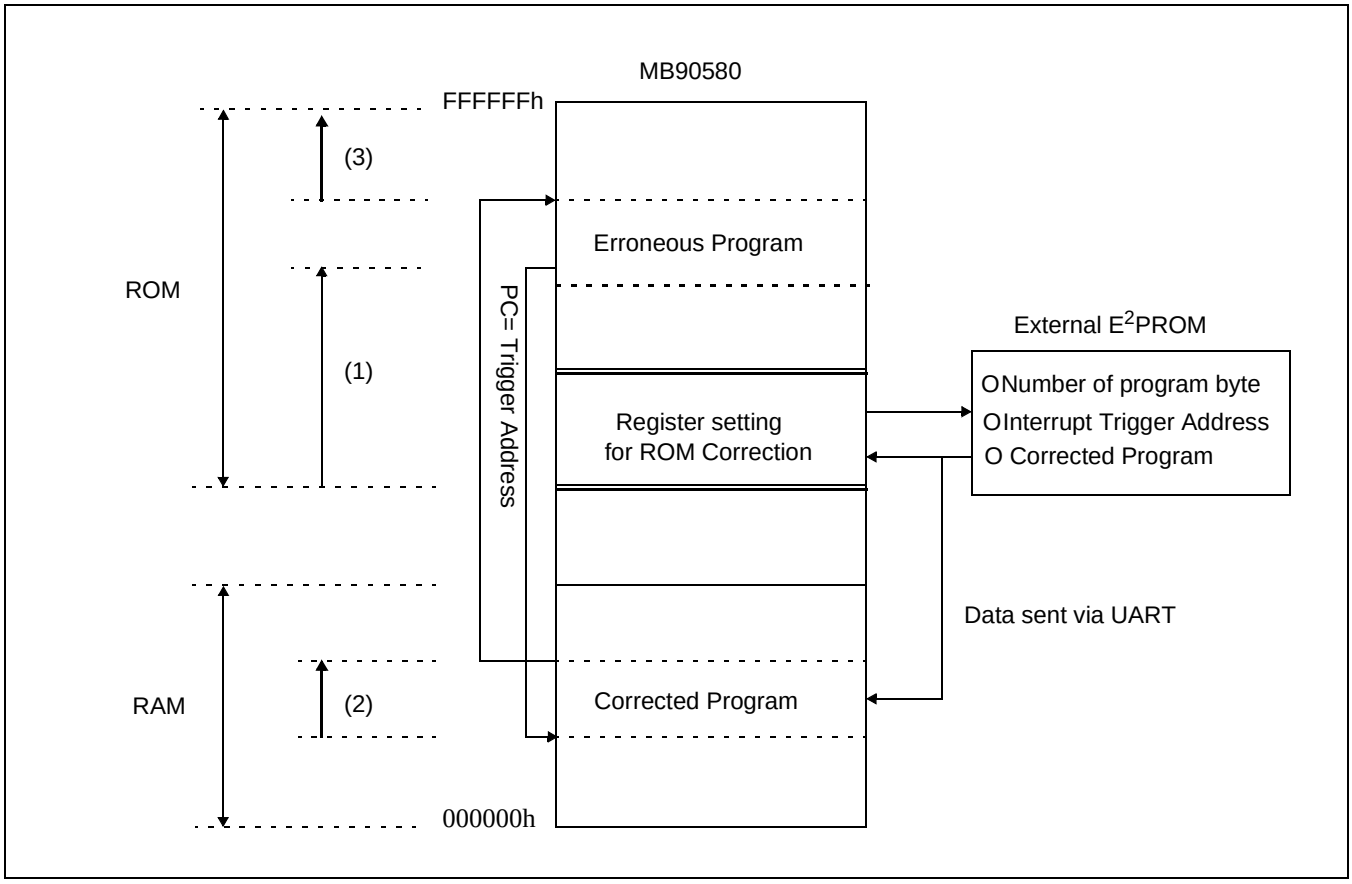


Figure 21.5b ROM Correction Processing Example

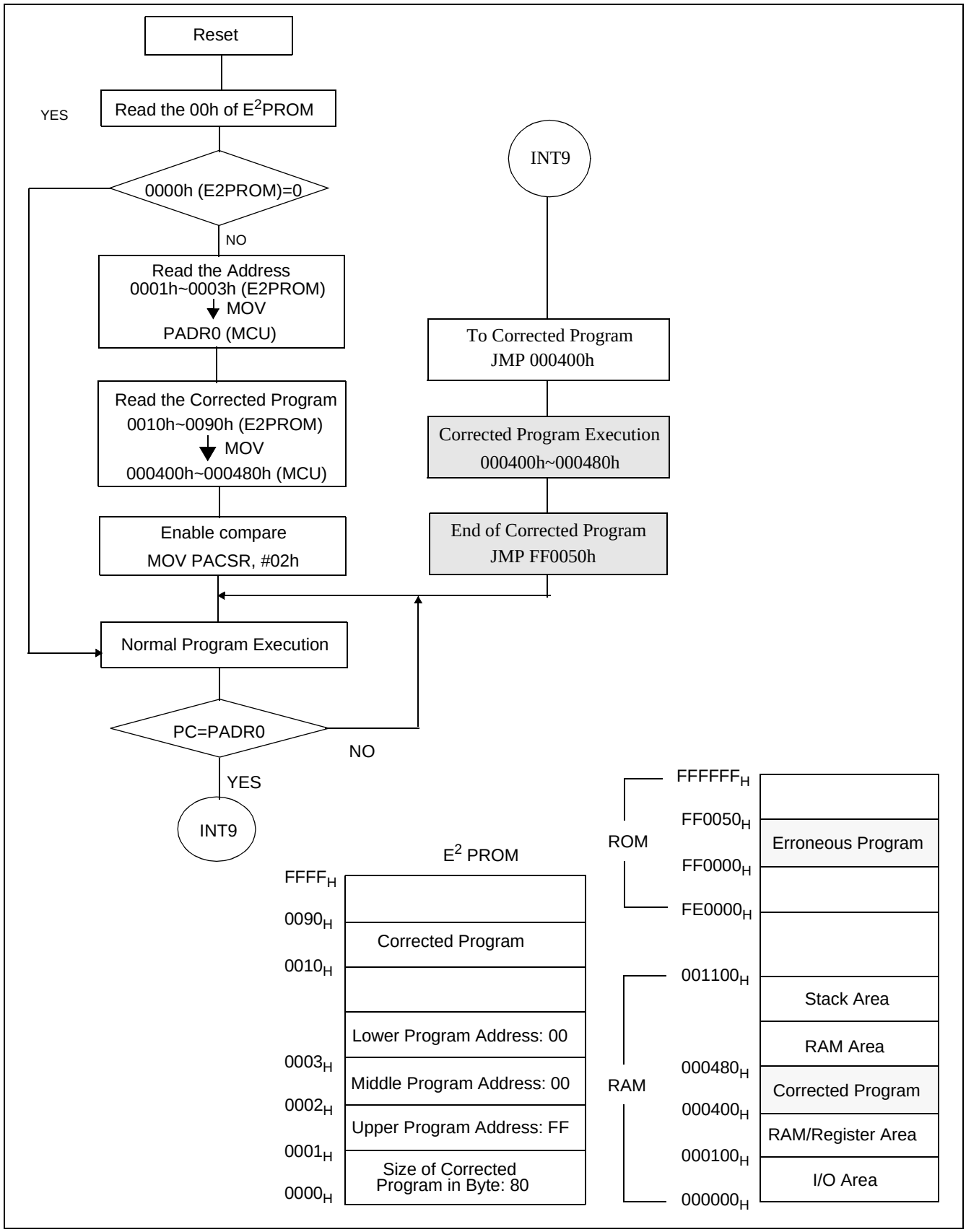


Figure 21.5c ROM Correction Processing Flow Diagram

Chapter 22:
ROM Mirroring Module

22.1 Outline

In ROM Mirroring Module the FF bank of the ROM can be seen through the 00 bank when chosen during register setting.

22.2 Block Diagram

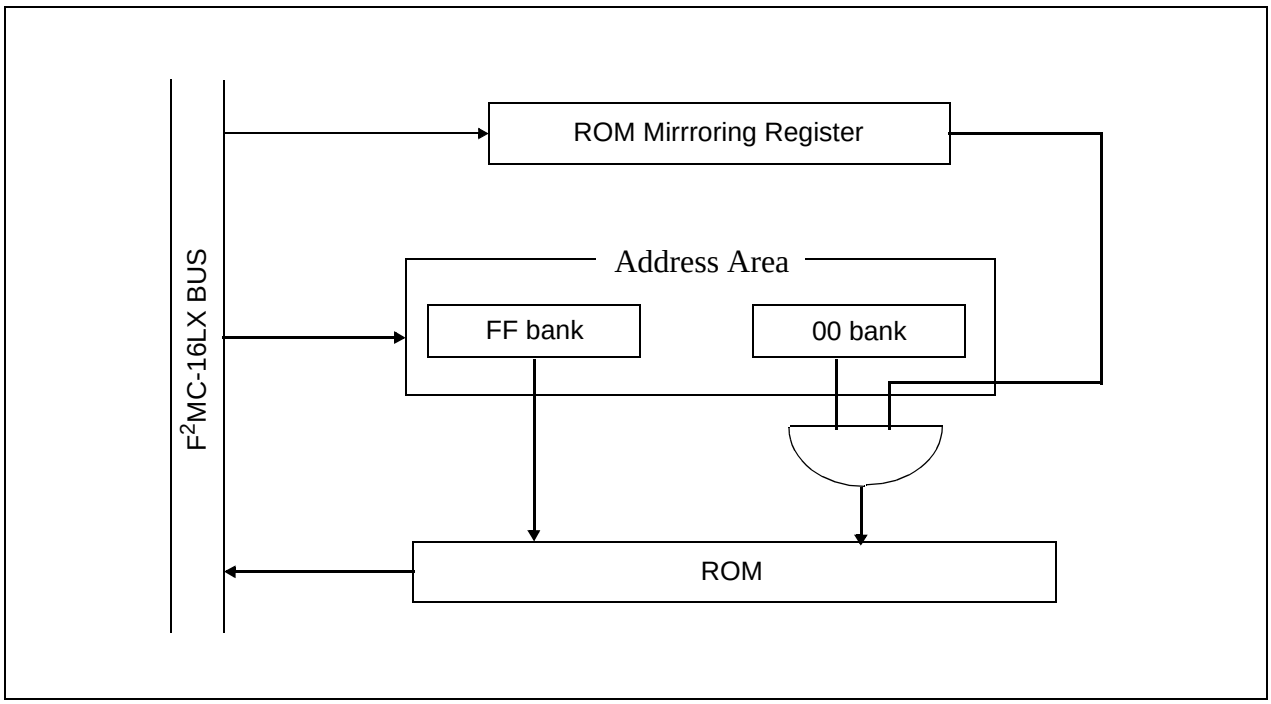


Figure 22.2a Block Diagram of ROM Mirroring Module

22.3 Registers and Register Details

ROM Mirror Function Select Register

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 0006F$_H$	—	—	—	—	—	—	—	MI	ROMM
Read/write ⇨	(–)	(–)	(–)	(–)	(–)	(–)	(–)	(W)	
Initial value ⇨	(–)	(–)	(–)	(–)	(–)	(–)	(–)	(1)	

Figure 22.3a Register of ROM Mirroring Module

22.3.1 ROM Mirror Function Select Register

ROM Mirror Function Select Register

	15	14	13	12	11	10	9	8	⇦ Bit number
Address : 0006F$_H$	—	—	—	—	—	—	—	MI	ROMM
Read/write ⇨	(–)	(–)	(–)	(–)	(–)	(–)	(–)	(W)	
Initial value ⇨	(–)	(–)	(–)	(–)	(–)	(–)	(–)	(1)	

Note: Do not access this register when the addresses 004000$_H$~00FFFF$_H$ is being accessed.

[bit 8] : MI

The ROM data in the FF bank can also be found in the 00 bank when ‘ 1’ is written to this bit. However, such as memory mapping will not be done when this bit is written to ‘ 0’. This bit is write only.

The memory during single chip mode and during internal ROM external bus mode will be as shown below.

Note: Only FF4000~FFFFFF is mirrorred to 004000~00FFFF when ROM mirroring functing is activated. Therefore, addresses FFF000~FF3FFF will not be mirrorred to 00 bank.

	MB90583	**MB90F583**	**MB90V580**
Address 1	FE0000$_H$	FE0000$_H$	FE0000$_H$
Address 2	001900$_H$	001900$_H$	001900$_H$

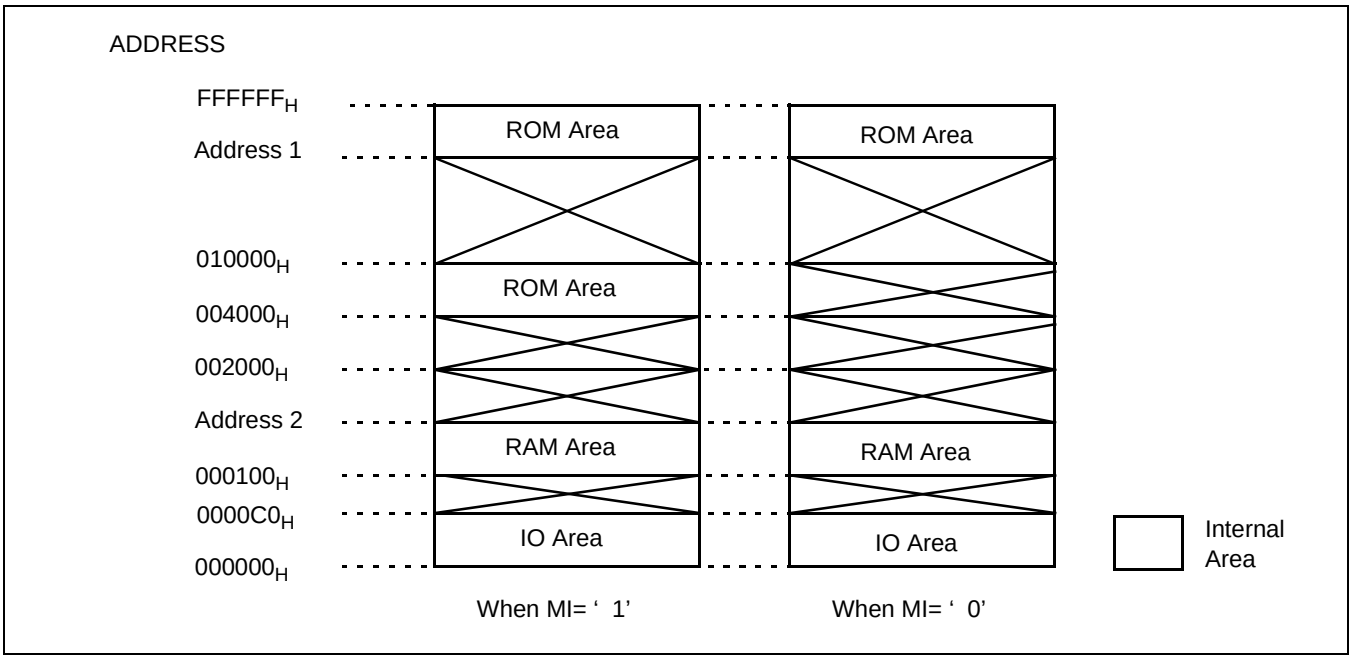


Figure 22.3b Memory in Single Chip Mode

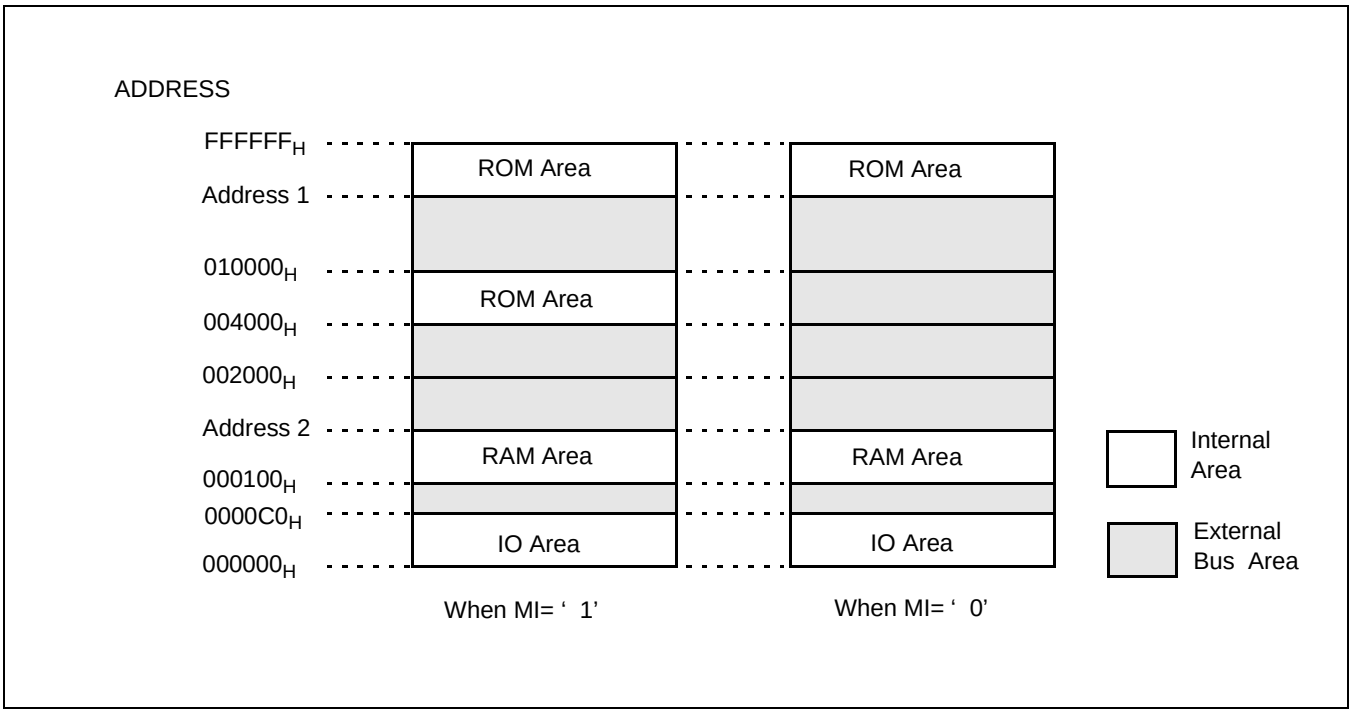


Figure 22.3c Memory in Internal ROM External Bus Mode

Appendix A: I/O Map

A.1 I/O Map

Table A.1a lists the addresses assigned to the registers of each microcontroller resource

Table A.1a I/O map

Address	Register	Abbreviation	Access	Resource	Initial value
00 H	Port 0 data register	PDR0	R/W	Port 0	XXXXXXXX
01 H	Port 1 data register	PDR1	R/W	Port 1	XXXXXXXX
02 H	Port 2 data register	PDR2	R/W	Port 2	XXXXXXXX
03 H	Port 3 data register	PDR3	R/W	Port 3	XXXXXXXX
04 H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX
05 H	Port 5 data register	PDR5	R/W	Port 5	XXXXXXXX
06 H	Port 6 data register	PDR6	R/W	Port 6	--XXXXXX
07 H	Port 7 data register	PDR7	R/W	Port 7	---XXXX-
08 H	Port 8 data register	PDR8	R/W	Port 8	XXXXXXXX
09 H	Port 9 data register	PDR9	R/W	Port 9	XXXXXXXX
0A H	Port A data register	PDRA	R/W	Port A	-----XXX
0B to 0F H	Reserved area				
10 H	Port 0 direction register	DDR0	R/W	Port 0	00000000
11 H	Port 1 direction register	DDR1	R/W	Port 1	00000000
12 H	Port 2 direction register	DDR2	R/W	Port 2	00000000
13 H	Port 3 direction register	DDR3	R/W	Port 3	00000000
14 H	Port 4 direction register	DDR4	R/W	Port 4	00000000
15 H	Port 5 direction register	DDR5	R/W	Port 5	00000000
16 H	Port 6 direction register	DDR6	R/W	Port 6	--000000
17 H	Port 7 direction register	DDR7	R/W	Port 7	---0000-
18 H	Port 8 direction register	DDR8	R/W	Port 8	00000000
19 H	Port 9 direction register	DDR9	R/W	Port 9	00000000
1A H	Port A direction register	DDRA	R/W	Port A	-----000
1B H	Port 4 pin register	ODR4	R/W	Port 4	00000000
1C H	Analog input enable register	ADER	R/W	Port 5, A/D	11111111
1D to 1F H	Reserved area				
20 H	Serial mode register 0	SMR0	R/W	UART0	00000000
21 H	Serial control register 0	SCR0	R/W		00000100
22 H	Serial input register/serial output register 0	SIDR/ SODR0	R/W		XXXXXXXX
23 H	Serial status register 0	SSR0	R/W		00001-00
24 H	Serial mode register 1	SMR1	R/W	UART1	00000000
25 H	Serial control register 1	SCR1	R/W		00000100
26 H	Serial input register/serial output register 1	SIDR/ SODR1	R/W		XXXXXXXX
27 H	Serial status register 1	SSR1	R/W		00001-00
28 H	Serial mode register 2	SMR2	R/W	UART2	00000000
29 H	Serial control register 2	SCR2	R/W		00000100
2A H	Serial input register/serial output register 2	SIDR/ SODR2	R/W		XXXXXXXX
2B H	Serial status register 2	SSR2	R/W		00001-00

Table A.1a I/O map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
2C H	Clock division control register 0	CDCR0	R/W	Communication prescaler 0	0---1111
2D H	Reserved area				
2E H	Clock division control register 1	CDCR1	R/W	Communication prescaler 1	0---1111
2F H	Reserved area				
30 H	Interrupt /DTP enable register	ENIR	R/W	DTP/external interrupt	00000000
31 H	Interrupt/DTP cause register	EIRR	R/W		XXXXXXXX
32 H	Request level setting register	ELVR	R/W		00000000
33 H					00000000
34 H	Clock division control register 2	CDCR2	R/W	Communication prescaler 2	0---1111
35 H	Reserved area				
36 H	Control status register	ADCS1	R/W	A/D converter	00000000
37 H		ADCS2			00000000
38 H	Data register	ADCR1	R		XXXXXXXX
39 H		ADCR2			00001--XX
3A H	D/A converter data register 0	DAT0	R/W	D/A converter	XXXXXXXX
3B H	D/A converter data register 1	DAT1	R/W		XXXXXXXX
3C H	D/A control register 0	DACR0	R/W		- - - - - - - 0
3D H	D/A control register 1	DACR1	R/W		- - - - - - - 0
3E H	Clock output enable register	CLKR	R/W	Clock monitor function	- - - - 0000
3F H	Reserved area				
40 H	Reload register L (ch.0)	PRLL0	R/W	8-/16-bit PPG	XXXXXXXX
41 H	Reload register H (ch.0)	PRLH0	R/W		XXXXXXXX
42 H	Reload register L (ch.1)	PRLL1	R/W		XXXXXXXX
43 H	Reload register H (ch.1)	PRLH1	R/W		XXXXXXXX
44 H	PPG0 operation mode control register	PPGC0	R/W		0X000XX1
45 H	PPG1 operation mode control register	PPGC1	R/W		0X000001
46 H	PPG0 and PPG1 output control register	PPGOE	R/W		00000000
47 H	Reserved area				
48 H	Control status register 0	TMCSR0	R/W	16-bit reload timer 0	00000000
49 H					---- 0000
4A H	16-bit timer register 0 / 16-bt reload register 0	TMR0 / TMRLR0	R/W		XXXXXXXX
4B H					XXXXXXXX
4C H	Control status register 1	TMCSR1	R/W	16-bit reload timer 1	00000000
4D H					---- 0000
4E H	16-bit timer register 1 / 16-bt reload register 1	TMR1 / TMRLR1	R/W		XXXXXXXX
4F H					XXXXXXXX
50 H	Control status register 2	TMCSR2	R/W	16-bit reload timer 2	00000000
51 H					---- 0000
52 H	16-bit timer register 2 / 16-bt reload register 2	TMR2 / TMRLR2	R/W		XXXXXXXX
53 H					XXXXXXXX
54 H	PWC control status register	PWCSR	R/W	16-bit PWC timer	00000000
55 H					00000000
56 H	PWC data buffer register	PWCR	R/W		XXXXXXXX
57 H					XXXXXXXX
58 H	Divide ratio control register	DIVR	R/W		------00
59 H	Reserved area				

Table A.1a I/O map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
5A H	Output Compare Register 0	OCCP0	R/W	Output Compare (Channel 0 To 1)	XXXXXXXX
5B H					XXXXXXXX
5C H	Output Compare Register 1	OCCP1	R/W		XXXXXXXX
5D H					XXXXXXXX
5E H	Output Compare Control Status Register 0	OCS0	R/W		0000--00
5F H	Output Compare Control Status Register 1	OCS1	R/W		---00000
60 H	Input Capture Register 0	IPCP0	R	Input Capture (Channel 0 To 3)	XXXXXXXX
61 H			R		XXXXXXXX
62 H	Input Capture Register 1	IPCP1	R		XXXXXXXX
63 H			R		XXXXXXXX
64 H	Input Capture Register 2	IPCP2	R		XXXXXXXX
65 H			R		XXXXXXXX
66 H	Input Capture Register 3	IPCP3	R		XXXXXXXX
67 H			R		XXXXXXXX
68 H	Input Capture Control Status Register Ch0,1	ICS01	R/W		00000000
69 H	Reserved area				
6A H	Input Capture Control Status Register Ch2,3	ICS23	R/W		00000000
6B H	Reserved area				
6C H	16-bit Timer Data Register (Low)	TCDTL	R/W	Free Run Timer	00000000
6D H	16-bit Timer Data Register (High)	TCDTH	R/W		00000000
6E H	16-bit Timer Control Status Register	TCCS	R/W		00000000
6F H	Rom Mirror Function Select Register	ROMM	W	ROM mirroring Module	---- ---1
70 H	Unit Address Register (Low)	MAWL	R/W	IEBus Interface	XXXXXXXX
71 H	Unit Address Register (High)	MAWH	R/W		XXXXXXXX
72 H	Slave Address Register (Low)	SAWL	R/W		XXXXXXXX
73 H	Slave Address Register (High)	SAWH	R/W		XXXXXXXX
74 H	Telegraph Length Set Register	DEWR	R/W		00000000
75 H	Multiaddress, Control Bit Set Register	DCWR	R/W		00000000
76 H	Command Register (Low)	CMRL	R/W		XX00000
77 H	Command Register (High)	CMRH	R/W		000000XX
78 H	Status Register (Low)	STRL	R		0011XXXX
79 H	Status Register (High)	STRH	R/W		00000000
7A H	Lock Read Register (Low)	LRRL	R		XXXXXXXX
7B H	Lock Read Register (High)	LRRH	R		XXX0XXXX
7C H	Master Address Read Register (Low)	MARL	R		XXXXXXXX
7D H	Master Address Read Register (High)	MARH	R		XXXXXXXX
7E H	Telegraph Length Read Register	DERR	R		XXXXXXXX
7F H	Multiaddress, Control Bit Read Register	DCRR	R		000XXXXX
80 H	Write Data Buffer	WDB	R/W		XXXXXXXX
81 H	Read Data Buffer	RDB	R		XXXXXXXX
82 H	Serial mode register 3	SMR3	R/W	UART3	00000000
83 H	Serial control register 3	SCR3	R/W		00000100
84 H	Serial input register/serial output register 3	SIDR/ SODR3	R/W		XXXXXXXX
85 H	Serial status register 3	SSR3	R/W		00001-00
86H	PWC Noise cancelling register	RNCR	R/W	PWC noise filter	---- -000
87H	Clock division control register 3	CDCR3	R/W	Communication prescaler 3	0---1111

Table A.1a I/O map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
88_H	Serial mode register 4	SMR4	R/W	UART4	00000000
89_H	Serial control register 4	SCR4	R/W		00000100
$8A_H$	Serial input register/serial output register 4	SIDR/ SODR4	R/W		XXXXXXXX
$8B_H$	Serial status register 4	SSR4	R/W		00001-00
$8C_H$	Port 0 resistor register	RDR0	R/W	Port 0	00000000
$8D_H$	Port 1 resistor register	RDR1	R/W	Port 1	00000000
$8E_H$	Port 6 resistor register	RDR6	R/W	Port 6	--000000
$8F_H$	Clock division control register 4	CDCR4	R/W	Communication prescaler 4	0---1111
90_H to $9D_H$	Reserved area				
9E H	Program address detect control status register	PACSR	R/W	ROM correction module	--000000
9F H	Delayed interrupt cause occurrence/ release register	DIRR	R/W	Delayed interrupt occurrence module	- - - - - - -0
A0 H	Low-power mode register	LPMCR	R/W	Low power	00011000
A1 H	Clock selection register	CKSCR	R/W		11111100
A2 H	Low noise output select register (Lower)	LNSRL	R/W	I/O port	00000000
A3 H	Low noise output select register (Upper)	LNSRH	R/W	I/O port	----0000
A4 H	Reserved area				
A5 H	Automatic read function selection register	ARSR	W	External bus interface	0011- -00
A6 H	External address output control register	HACR	W		0000000
A7 H	Bus control signal selection register	ECSR	W		0000000-
A8 H	Watchdog control register	WDTC	R/W	Watchdog timer	XXXXX111
A9 H	Time base timer control register	TBTC	R/W	Time base timer	1- -00100
AA_H	Watch timer control register	WTC	R/W	Watch Timer	1X000000
AB_H to AD_H	Reserved area				
AE_H	Flash control register	FMCS	R/W	Flash interface	000X0XX0
AF_H	Reserved area				
B0 H	Interrupt control register 00	ICR00	R/W	Interrupt controller	00000111
B1 H	Interrupt control register 01	ICR01	R/W		00000111
B2 H	Interrupt control register 02	ICR02	R/W		00000111
B3 H	Interrupt control register 03	ICR03	R/W		00000111
B4 H	Interrupt control register 04	ICR04	R/W		00000111
B5 H	Interrupt control register 05	ICR05	R/W		00000111
B6 H	Interrupt control register 06	ICR06	R/W		00000111
B7 H	Interrupt control register 07	ICR07	R/W		00000111
B8 H	Interrupt control register 08	ICR08	R/W		00000111
B9 H	Interrupt control register 09	ICR09	R/W		00000111
BA H	Interrupt control register 10	ICR10	R/W		00000111
BB H	Interrupt control register 11	ICR11	R/W		00000111
BC H	Interrupt control register 12	ICR12	R/W		00000111
BD H	Interrupt control register 13	ICR13	R/W		00000111
BE H	Interrupt control register 14	ICR14	R/W		00000111
BF H	Interrupt control register 15	ICR15	R/W		00000111
CO to FF H	External area	—	—	—	—
100_H to $\#_H$	RAM area	—	—	—	—
$\#_H$ to $1FEF_H$	Reserved area	—	—	—	—

Table A.1a I/O map (Continued)

Address	Register	Abbreviation	Access	Resource	Initial value
1FF0 $_H$	Program address detection register 0	PADR0	R/W	Program patch manipulation	XXXXXXXX
1FF1 $_H$	Program address detection register 1		R/W		XXXXXXXX
1FF02 $_H$	Program address detection register 2		R/W		XXXXXXXX
1FF3 $_H$	Program address detection register 3	PADR1	R/W		XXXXXXXX
1FF4 $_H$	Program address detection register 4		R/W		XXXXXXXX
1FF5 $_H$	Program address detection register 5		R/W		XXXXXXXX
$1FF6_H$ to $1FFF_H$	Reserved area	—	—	—	—

APPENDIX B: Instructions

B.1 Addressing

In the F^{2}MC-16LX, the address format is determined by either the instruction's effective address specification, or by the instruction code itself (implied addressing).

B.1.1 Effective address field

The address formats specified in the effective address field are shown in Table B.1.1a.

Table B.1.1a Effective Address Field

Code	Notation	Address format	Default bank
00 01 02 03 04 05 06 07	R0 RW0 RL0 R1 RW1 (RL0) R2 RW2 RL1 R3 RW3 (RL1) R4 RW4 RL2 R5 RW5 (RL2) R6 RW6 RL3 R7 RW7 (RL3)	Register direct Starting from the left, "ea" corresponds to the byte, word and long-word types.	None
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3	Register indirect	DTB DTB ADB SPB
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+	Register indirect with post-incrementing	DTB DTB ADB SPB
10 11 12 13	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8	Register indirect with 8-bit displacement	DTB DTB ADB SPB
14 15 16 17	@RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8	Register indirect with 8-bit displacement	DTB DTB ADB SPB
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16	Register indirect with 16-bit displacement	DTB DTB ADB SPB
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	DTB DTB PCB DTB

B.1.2 Addressing Details

(1) Immediate value (#imm)

This format specifies the operand value directly.

- #imm4
- #imm8
- #imm16
- #imm32

(2) Compressed direct address (dir)

In this format, the operand specifies the low-order 8 bits of the memory address. Bits 8 to 15 of the address are specified by the DPR. Bits 16 to 23 of the address are indicated by the DTB.

(3) Direct address (addr16)

In this format, the operand specifies the low-order 16 bits of the memory address. Bits 16 to 23 of the address are indicated by the DTB.

(4) Register direct

This format specifies a direct register as the operand.

General-purpose registers

Byte:	R0, R1, R2, R3, R4, R5, R6, R7
Word:	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
Long word:	RL0, RL1, RL2, RL3

Dedicated registers

Accumulator:	A, AL
Pointer:	SP
Bank:	PCB, DTB, USB, SSB, ADB
Page:	DPR
Control:	PS, CCR, RP, ILM

Note: Regarding the SP, either the USP or the SSP is selected and used, depending on the value of the S bit in the CCR. In addition, in a branching instruction, the PC is implicitly specified, and is not described in the instruction operand.

(5) Register indirect (@RWj j = 0 to 3)

This format accesses the memory address indicated by the contents of the general-purpose register RWj. When RW0/RW1 is used, bits 16 to 23 of the address are indicated by DTB; if RW3 is used, bits 16 to 23 of the address are indicated by SPB, and if RW2 is used, bits 16 to 23 of the address are indicated by ADB.

(6) Register indirect with post-incrementing (@RWj+ j = 0 to 3)

This format accesses the memory address indicated by the contents of the general-purpose register RWj. After the operand operation, RWj is incremented by the data length of the operand (by 1 for a byte, 2 for a word, and 4 for a long-word). When RW0/RW1 is used, bits 16 to 23 of the address are indicated by DTB; if RW3 is used, bits 16 to 23 of the address are indicated by SPB, and if RW2 is used, bits 16 to 23 of the address are indicated by ADB. Note that if the post-incremented result is the address of the register for which the increment specification was made, the value that is referenced subsequently is the incremented value. In addition, in such a case, if the instruction was a write instruction, the data written by the instruction is given priority, so the register that was to have been incremented contains the write data in the end.

(7) Register indirect with displacement (@RWi + disp8 i= 0 to 7/@RWj + disp16 j = 0 to 3)

This format accesses the memory address indicated by the sum of the contents of the general-purpose register RWj and the displacement value. The displacement value can be one of two types, either a byte or a word, and is added as a signed value. When RW0, RW1, RW4, or RW5 is used, bits 16 to 23 of the address are indicated by DTB; if RW3 or RW7 is used, bits 16 to 23 of the address are indicated by SPB, and if RW2 or RW6 is used, bits 16 to 23 of the address are indicated by ADB.

(8) Register indirect with base index (@RW0 + RW7, @RW1 + RW7)

This format accesses the memory address indicated by the sum of the contents of the general-purpose register and either RW0 or RW1. Bits 16 to 23 of the address are indicated by DTB.

(9) Program counter indirect with displacement (@PC + disp16)

This format accesses the memory address indicated by the sum of the “instruction address + 4 + disp16”. The displacement value is a word length value. Bits 16 to 23 of the address are indicated by PCB.

The operand address is generally regarded as “the next instruction address + disp16”, but note that this does not hold true for the instructions indicated below:

- DBNZ eam, rel
- DWBNZ eam, rel
- MOV eam, #imm8
- MOVW eam, #imm16
- CBNE eam, #imm8, rel
- CWBNE eam, #imm16, rel

(10) Accumulator indirect (@A)

This format has two types: one in which the contents of AL specify bits 00 to 15 of the address and DTB indicates bits 16 to 23; and one in which the low-order 24 bits of A specify bits 00 to 23 of the address.

(11) I/O direct (io)

In this format, the memory address of the operand is specified directly by the 8-bit displacement value. Regardless of the value of DTB and DPR, the I/O space from 000000_H to $0000FF_H$ is accessed. The access space specification prefix has no effect on this addressing format.

(12) Long register indirect with displacement (@RLi + disp8 i = 0 to 3)

This format accesses the memory address indicated by the low-order 24 bits of the sum of the contents of the general-purpose register RLi plus the displacement value. The displacement value is 8 bits, and is added as a signed numeral.

(13) Compressed direct bit address (dir:bp)

This format specifies the low-order 8 bits of the memory address with the operand. In addition, bits 8 to 15 of the address are indicated by DPR. Finally, bits 16 to 23 of the address are indicated by DTB. The bit position is indicated by ":bp", with larger numbers being closer to the MSB and smaller numbers being closer to the LSB.

(14) I/O direct bit address (io:bp)

This format directly specifies a bit within a physical address from 000000_H to $0000FF_H$. The bit position is indicated by ":bp", with larger numbers being closer to the MSB and smaller numbers being closer to the LSB.

(15) Direct bit address (addr16:bp)

This format directly specifies any bit within a 64-kilobyte region. Bits 16 to 23 of the address are indicated by DTB. The bit position is indicated by ":bp", with larger numbers being closer to the MSB and smaller numbers being closer to the LSB.

(16) Register list (rlst)

This format specifies the register that is the target of a stack push/pop instruction.

MSB LSB

RW7	RW6	RW5	RW4	RW3	RW2	RW1	RW0

A register is selected when the corresponding bit is "1", and is not selected when the corresponding bit is "0".

Fig. B.1.2a Register List Configuration

(17) Program counter relative branching address (rel)

With this format, the address of the destination of a branching instruction is the sum of the value of the PC and the 8-bit displacement value. If the result exceeds 16 bits, the amount of the overflow is ignored and the bank register is not incremented or decremented; therefore, the address is kept within a 64-kilobyte bank. This format is used in unconditional and conditional branching instructions. Bits 16 to 23 of the address are indicated by PCB.

(18) Direct branching address (addr16)

With this format, the address of the destination of a branching instruction is specified directly by the displacement value. The displacement value is 16 bits, and indicates the branching destination within a logical memory space. This format is used in unconditional branching instructions and subroutine call instructions. Bits 16 to 23 of the address are indicated by PCB.

(19) Physical direct branching address (addr24)

With this format, the address of the destination of a branching instruction is specified directly by the displacement value. The displacement value is 24 bits, and specifies the physical address of the branching destination. This format is used in unconditional branching instructions, subroutine call instructions, and software interrupt instructions.

(20) Accumulator indirect branching address (@A)

In this format, the 16 bits of the accumulator AL specify the branching destination address. This address indicates a branching destination within a bank space; in this case, bits 16 to 23 of the address are indicated by the PCB. In the case of JCTX, however, bits 16 to 23 of the address are indicated by DTB. This format is used in unconditional branching instructions.

(21) Vector address (#vct)

The contents of the specified vector become the branching destination address. There are two data lengths for vector numbers: 4 bits and 8 bits. This format is used in subroutine call instructions and software interrupt instructions.

(22) Indirect specification branching address (@ear)

The word data in the address indicated by “ear” is the branching destination address.

(23) Indirect specification branching address (@eam)

The word data in the address indicated by “eam” is the branching destination address.

B.2 Instruction Set

Table B.2a Explanation of Items in Table of Instructions

Item	Explanation
Mnemonic	Upper-case letters and symbols: .....Described as they appear in assembler. Lower-case letters: .....Replaced when described in assembler. Numbers after lower-case letters: .....Indicate the bit width within the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table 4.2.4 for details about meanings of letters in items.
RG	Indicates the register access count during execution of instruction. Used to calculate compensation values for CPU intermittent operation.
B	Indicates the compensation value for calculating the number of actual cycles during execution of instruction. The number of actual cycles during execution of instruction is the compensation value summed with the value in the "~" column.
Operation	Indicates operation of instruction.
LH	Indicates special operations involving bits 15 through 08 of the accumulator. Z:.....Transfers "0". X:.....Sign-extended transfer through sign extension. -:.....Transfers nothing.
AH	Indicates special operations involving the high-order 16 bits in the accumulator. *:.....Transfers from AL to AH. -:.....No transfer. Z:.....Transfers 00 to AH. X:.....Transfers 00_H or FF_H to AH using sign extension AL.
I	Indicates the status of each of the following flags: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), and C (carry). *:.....Changes due to execution of instruction. -:.....No change. S:.....Set by execution of instruction. R: .....Reset by execution of instruction.
S	
T	
N	
Z	
V	
C	
RMW	Indicates whether the instruction is a read-modify-write instruction (a single instruction that reads data from memory, etc., processes the data, and then writes the result to memory.). *:.....Instruction is a read-modify-write instruction. -:.....Instruction is not a read-modify-write instruction. Note: A read-modify-write instruction cannot be used on addresses that have different meanings depending on whether they are read or written.

- Number of execution cycles

  The number of cycles required for the execution of an instruction is obtained by summing the value shown in the table for the “number of cycles” for the instruction in question, the compensation value (which depends on certain conditions), and the “number of cycles” needed for the program fetch.

  When fetching a program in memory connected to the 16-bit bus, such as on-chip ROM, a program fetch is performed for each two-byte (word) boundary crossed by the instruction being executed; therefore, if there is any interference with data access, etc., the number of execution cycles increases.

  When fetching a program in memory connected to the 8-bit external data bus, a program fetch is performed for each byte of the instruction being executed; therefore, if there is any interference with data access, etc., the number of execution cycles increases.

  In CPU intermittent operation, each access to general-purpose registers, internal ROM, internal RAM, internal I/O functions or external bus causes the CPU clock to pause for a fixed number of cycles determined by the CG1/CG0 bits in the low power consumption mode control register. For this reason, the number of machine clock cycles required to execute an instruction under CPU intermittent operation is the normal number of cycles plus an offset number of cycles that is derived by multiplying the number of access operations by the length (in cycles) of the fixed pause.

Table B.2b Explanation of Symbols in Table of Instructions

Symbol	Explanation
A	32-bit accumulator The bit length varies according to the instruction. Byte: ..... Low-order 8 bits of AL Word: ..... 16 bits of AL Long: ..... 32 bits of AL:AH
AH AL	High-order 16 bits of A Low-order 16 bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	Program bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir addr16 addr24 ad24 0 to 15 ad24 16 to 23	Compact direct addressing Direct addressing Physical direct addressing Bits 0 to 15 of addr24 Bits 16 to 23 of addr24
io	I/O area (000000_H to $0000FF_H$)

#imm4
#imm8
#imm16
#imm32
ext(imm8)
	4-bit immediate data
	8-bit immediate data
	16-bit immediate data
	32-bit immediate data
	16-bit data signed and extended from 8-bit immediate data

disp8 disp16	8-bit displacement 16-bit displacement
bp	Bit offset value
vct4 vct8	Vector number (0 to 15) Vector number (0 to 255)
( )b	Bit address
rel ear eam	Branch specification relative to PC Effective addressing (codes 00 to 07) Effective addressing (codes 08 to 1F)
rlst	Register list

Table B.2c Effective Address Fields

Code	Notation	Address format	Number of bytes in address extension [Note]
00 01 02 03 04 05 06 07	R0 RW0 RL0 R1 RW1 (RL0) R2 RW2 RL1 R3 RW3 (RL1) R4 RW4 RL2 R5 RW5 (RL2) R6 RW6 RL3 R7 RW7 (RL3)	Register direct “ea” corresponds to byte, word, and long-word types, starting from the left	–
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3	Register indirect	0
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+	Register indirect with post-incrementing	0
10 11 12 13 14 15 16 17	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8 @RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8	Register indirect with 8-bit displacement	1
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16	Register indirect with 16-bit displacement	2
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	0 0 2 2

Note: The number of bytes for address extension is indicated by the “+” symbol in the “#” (number of bytes) column in the Table of Instructions and by the number of bytes in the detailed instruction rules.

Table B.2d Number of Execution Cycles for Each Form of Addressing

Code	Operand	(a)	
		Number of execution cycles for each form of addressing	Number of accesses for each form of addressing
00 to 07	Ri RWi RLi	Listed in Table of Instructions	Listed in Table of Instructions
08 to 0B	@RWj	2	1
0C to 0F	@RWj+	4	2
10 to 17	@RWi+disp8	2	1
18 to 1B	@RWj+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

Note: "(a)" is used in the "~" (number of cycles) column, column B (compensation value) and in the detailed instruction rules in the Table of Instructions.

Table B.2e Compensation Values for Number of Cycles Used to Calculate Number of Actual Cycles

Operand	(b) byte		(c) word		(d) long	
	Cycles	Access cycles	Cycles	Access cycles	Cycles	Access cycles
Internal register	+0	1	+0	1	+0	2
Internal RAM even address Internal RAM odd address	+0 +0	1 1	+0 +2	1 2	+0 +4	2 4
Even address on external data bus (16 bits) Odd address on external data bus (16 bits)	+1 +1	1 1	+1 +4	1 2	+2 +8	2 4
External data bus (8 bits)	+1	1	+4	2	+8	4

Note: "(b)", "(c)", and "(d)" are used in the "~" (number of cycles) column, column B (compensation value) and in the detailed instruction rules in the Table of Instructions.

When the external data bus is used, it is necessary to add in the number of weighted cycles used for ready input and automatic ready.

Table B.2f Compensation Values for Number of Cycles Used to Calculate Number of Program Fetch Cycles

Instruction	Byte boundary	Word boundary
Internal memory	–	+2
External data bus (16 bits)	–	+3
External data bus (8 bits)	+3	–

Note: When the external data bus is used, it is necessary to add in the number of weighted cycles used for ready input and automatic ready.

Because instruction execution is not slowed down by all program fetches in actuality, these compensation values should be used for “worst case” calculations.

B.2.1 F²MC-16LX Instruction Set (351 Instructions)

Table B.2.1a Transfer Instructions (Byte) (41 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV	A,dir	2	3	0	(b)	byte (A) ← (dir)	Z	*	-	-	-	*	*	-	-	-
MOV	A,addr16	3	4	0	(b)	byte (A) ← (addr16)	Z	*	-	-	-	*	*	-	-	-
MOV	A,Ri	1	2	1	0	byte (A) ← (Ri)	Z	*	-	-	-	*	*	-	-	-
MOV	A,ear	2	2	1	0	byte (A) ← (ear)	Z	*	-	-	-	*	*	-	-	-
MOV	A,eam	2+	3+(a)	0	(b)	byte (A) ← (eam)	Z	*	-	-	-	*	*	-	-	-
MOV	A,io	2	3	0	(b)	byte (A) ← (io)	Z	*	-	-	-	*	*	-	-	-
MOV	A,#imm8	2	2	0	0	byte (A) ← (imm8)	Z	*	-	-	-	*	*	-	-	-
MOV	A,@A	2	3	0	(b)	byte (A) ← ((A))	Z	-	-	-	-	*	*	-	-	-
MOV	A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	Z	*	-	-	-	*	*	-	-	-
MOVN	A,#imm4	1	1	0	0	byte (A) ← imm4	Z	*	-	-	-	R	*	-	-	-
MOVX	A,dir	2	3	0	(b)	byte (A) ← (dir)	X	*	-	-	-	*	*	-	-	-
MOVX	A,addr16	3	4	0	(b)	byte (A) ← (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX	A,Ri	2	2	1	0	byte (A) ← (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX	A,ear	2	2	1	0	byte (A) ← (ear)	X	*	-	-	-	*	*	-	-	-
MOVX	A,eam	2+	3+(a)	0	(b)	byte (A) ← (eam)	X	*	-	-	-	*	*	-	-	-
MOVX	A,io	2	3	0	(b)	byte (A) ← (io)	X	*	-	-	-	*	*	-	-	-
MOVX	A,#imm8	2	2	0	0	byte (A) ← (imm8)	X	*	-	-	-	*	*	-	-	-
MOVX	A,@A	2	3	0	(b)	byte (A) ← ((A))	X	-	-	-	-	*	*	-	-	-
MOVX	A,@RWi+disp8	2	5	1	(b)	byte (A) ← ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX	A,@RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOV	dir,A	2	3	0	(b)	byte (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV	addr16,A	3	4	0	(b)	byte (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,A	1	2	1	0	byte (Ri) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV	ear,A	2	2	1	0	byte (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV	eam,A	2+	3+(a)	0	(b)	byte (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV	io,A	2	3	0	(b)	byte (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV	@RLi+disp8,A	3	10	2	(b)	byte ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,ear	2	3	2	0	byte (Ri) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,eam	2+	4+(a)	1	(b)	byte (Ri) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOV	ear,Ri	2	4	2	0	byte (ear) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV	eam,Ri	2+	5+(a)	1	(b)	byte (eam) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,#imm8	2	2	1	0	byte (Ri) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV	io,#imm8	3	5	0	(b)	byte (io) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV	dir,#imm8	3	5	0	(b)	byte (dir) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV	ear,#imm8	3	2	1	0	byte (ear) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV	eam,#imm8	3+	4+(a)	0	(b)	byte (eam) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV	@AL,AH / MOV @A,T	2	3	0	(b)	byte ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCH	A,ear	2	4	2	0	byte (A) ←→ (ear)	Z	-	-	-	-	-	-	-	-	-
XCH	A,eam	2+	5+(a)	0	2×(b)	byte (A) ←→ (eam)	Z	-	-	-	-	-	-	-	-	-
XCH	Ri,ear	2	7	4	0	byte (Ri) ←→ (ear)	-	-	-	-	-	-	-	-	-	-
XCH	Ri,eam	2+	9+(a)	2	2×(b)	byte (Ri) ←→ (eam)	-	-	-	-	-	-	-	-	-	-

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1b Transfer Instructions (Word/Long-Word) (38 Instructions)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A,dir	2	3	0	(c)	word (A) ← (dir)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr16	3	4	0	(c)	word (A) ← (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW A,SP	1	1	0	0	word (A) ← (SP)	-	*	-	-	-	*	*	-	-	-
MOVW A,RWi	1	2	1	0	word (A) ← (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW A,ear	2	2	1	0	word (A) ← (ear)	-	*	-	-	-	*	*	-	-	-
MOVW A,eam	2+	3+(a)	0	(c)	word (A) ← (eam)	-	*	-	-	-	*	*	-	-	-
MOVW A,io	2	3	0	(c)	word (A) ← (io)	-	*	-	-	-	*	*	-	-	-
MOVW A,@A	2	3	0	(c)	word (A) ← ((A))	-	-	-	-	-	*	*	-	-	-
MOVW A,#imm16	3	2	0	0	word (A) ← imm16	-	*	-	-	-	*	*	-	-	-
MOVW A,@RWi+disp8	2	5	1	(c)	word (A) ← ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,@RLi+disp8	3	10	2	(c)	word (A) ← ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW dir,A	2	3	0	(c)	word (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW addr16,A	3	4	0	(c)	word (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW SP,A	1	1	0	0	word (SP) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,A	1	2	1	0	word (RWi) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW ear,A	2	2	1	0	word (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW eam,A	2+	3+(a)	0	(c)	word (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW io,A	2	3	0	(c)	word (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RWi+disp8,A	2	5	1	(c)	word ((RWi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RLi+disp8,A	3	10	2	(c)	word ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,ear	2	3	2	0	word (RWi) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,eam	2+	4+(a)	1	(c)	word (RWi) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVW ear,RWi	2	4	2	0	word (ear) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW eam,RWi	2+	5+(a)	1	(c)	word (eam) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,#imm16	3	2	1	0	word (RWi) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW io,#imm16	4	5	0	(c)	word (io) ← imm16	-	-	-	-	-	-	-	-	-	-
MOVW ear,#imm16	4	2	1	0	word (ear) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW eam,#imm16	4+	4+(a)	0	(c)	word (eam) ← imm16	-	-	-	-	-	-	-	-	-	-
MOVW @AL,AH / MOVW @A,T	2	3	0	(c)	word ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCHW A,ear	2	4	2	0	word (A) ←→ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW A,eam	2+	5+(a)	0	2×(c)	word (A) ←→ (eam)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,ear	2	7	4	0	word (RWi) ←→ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,eam	2+	9+(a)	2	2×(c)	word (RWi) ←→ (eam)	-	-	-	-	-	-	-	-	-	-
MOVL A,ear	2	4	2	0	long (A) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVL A,eam	2+	5+(a)	0	(d)	long (A) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVL A,#imm32	5	3	0	0	long (A) ← imm32	-	-	-	-	-	*	*	-	-	-
MOVL ear,A	2	4	2	0	long (ear1) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVL eam,A	2+	5+(a)	0	(d)	long (eam1) ← (A)	-	-	-	-	-	*	*	-	-	-

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1c Addition and Subtraction Instructions (Byte/Word/Long-Word) (42 Instructions)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD A,#imm8	2	2	0	0	byte (A) ← (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD A,dir	2	5	0	(b)	byte (A) ← (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD A,ear	2	3	1	0	byte (A) ← (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD ear,A	2	3	2	0	byte (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD eam,A	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC A	1	2	0	0	byte (A) ← (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,ear	2	3	1	0	byte (A) ← (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDDC A	1	3	0	0	byte (A) ← (AH) + (AL) + (C) (hexadecimal)	Z	-	-	-	-	*	*	*	*	-
SUB A,#imm8	2	2	0	0	byte (A) ← (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB A,dir	2	5	0	(b)	byte (A) ← (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB A,ear	2	3	1	0	byte (A) ← (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB ear,A	2	3	2	0	byte (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB eam,A	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC A	1	2	0	0	byte (A) ← (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,ear	2	3	1	0	byte (A) ← (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBDC A	1	3	0	0	byte (A) ← (AH) - (AL) - (C) (hexadecimal)	Z	-	-	-	-	*	*	*	*	-
ADDW A	1	2	0	0	word (A) ← (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW A,ear	2	3	1	0	word (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW A,#imm16	3	2	0	0	word (A) ← (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW ear,A	2	3	2	0	word (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW eam,A	2+	5+(a)	0	2×(c)	word (eam) ← (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW A,ear	2	3	1	0	word (A) ← (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW A	1	2	0	0	word (A) ← (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW A,ear	2	3	1	0	word (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW A,#imm16	3	2	0	0	word (A) ← (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW ear,A	2	3	2	0	word (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW eam,A	2+	5+(a)	0	2×(c)	word (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW A,ear	2	3	1	0	word (A) ← (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW A,eam	2+	4+(a)	0	(c)	word (A) ← (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL A,ear	2	6	2	0	long (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL A,#imm32	5	4	0	0	long (A) ← (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL A,ear	2	6	2	0	long (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL A,eam	2+	7+(a)	0	(d)	long (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL A,#imm32	5	4	0	0	long (A) ← (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1d Increment and Decrement Instructions (Byte/Word/Long-Word) (12 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) ← (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) ← (ear) - 1	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) ← (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2×(c)	word (eam) ← (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) ← (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2×(c)	word (eam) ← (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	7	4	0	long (ear) ← (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2×(d)	long (eam) ← (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) ← (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2×(d)	long (eam) ← (eam) - 1	-	-	-	-	-	*	*	*	-	*

Table B.2.1e Compare Instructions (Byte/Word/Long-Word) (11 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	3+(a)	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	3+(a)	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1f Unsigned Multiplication and Division Instructions (Word/Long-Word) (11 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU	A	1	*1	0	0	word (AH) / byte (AL) Quotient → byte (AL) Remainder → byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU	A,ear	2	*2	1	0	word (A) / byte (ear) Quotient → byte (A) Remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU	A,eam	2+	*3	0	*6	word (A) / byte (eam) Quotient → byte (A) Remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,ear	2	*4	1	0	long (A) / word (ear) Quotient → word (A) Remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,eam	2+	*5	0	*7	long (A) / word (eam) Quotient → word (A) Remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MULU	A	1	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A	1	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3 when dividing into zero, 7 when an overflow occurs, and 15 normally.

*2: 4 when dividing into zero, 8 when an overflow occurs, and 16 normally.

*3: 6 + (a) when dividing into zero, 9 + (a) when an overflow occurs, and 19 + (a) normally.

*4: 4 when dividing into zero, 7 when an overflow occurs, and 22 normally.

*5: 6 + (a) when dividing into zero, 8 + (a) when an overflow occurs, and 26 + (a) normally.

*6: (b) when dividing into zero or when an overflow occurs, and 2 × (b) normally.

*7: (c) when dividing into zero or when an overflow occurs, and 2 × (c) normally.

*8: 3 when byte (AH) is zero, and 7 when byte (AH) is not 0.

*9: 4 when byte (ear) is zero, and 8 when byte (ear) is not 0.

*10: 5 + (a) when byte (eam) is zero, and 9 + (a) when byte (eam) is not 0.

*11: 3 when word (AH) is zero, and 11 when word (AH) is not 0.

*12: 4 when word (ear) is zero, and 12 when word (ear) is not 0.

*13: 5 + (a) when word (eam) is zero, and 13 + (a) when word (eam) is not 0.

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1g Signed Multiplication and Division Instructions (Word/Long-Word) (11 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV	A	1	*1	0	0	word (AH) / byte (AL) Quotient → byte (AL) Remainder → byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV	A,ear	2	*2	1	0	word (A) / byte (ear) Quotient → byte (A) Remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV	A,eam	2+	*3	0	*6	word (A) / byte (eam) Quotient → byte (A) Remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIVW	A,ear	2	*4	1	0	long (A) / word (ear) Quotient → word (A) Remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW	A,eam	2+	*5	0	*7	long (A) / word (eam) Quotient → word (A) Remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MUL	A	2	*8	0	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL	A,ear	2	*9	1	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL	A,eam	2+	*10	0	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULW	A	2	*11	0	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW	A,ear	2	*12	1	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW	A,eam	2+	*13	0	(c)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

*1: 3 when dividing into zero, 8 or 18 when an overflow occurs, and 18 normally.

*2: 3 when dividing into zero, 10 or 21 when an overflow occurs, and 22 normally.

*3: 4 + (a) when dividing into zero, 11 + (a) or 22 + (a) when an overflow occurs, and 23 + (a) normally.

*4: When dividend is positive: 4 when dividing into zero, 10 or 29 when an overflow occurs, and 30 normally.

When dividend is negative: 4 when dividing into zero, 11 or 30 when an overflow occurs, and 31 normally.

*5: When dividend is positive: 4+ (a) when dividing into zero, 11+ (a) or 30+ (a) when an overflow occurs, and 31+ (a) normally.

When dividend is negative: 4+ (a) when dividing into zero, 12+ (a) or 31+ (a) when an overflow occurs, and 32+ (a) normally.

*6: (b) when dividing into zero or when an overflow occurs, and 2 × (b) normally.

*7: (c) when dividing into zero or when an overflow occurs, and 2 × (c) normally.

*8: 3 when byte (AH) is zero, 12 when the result is possible, and 13 when the result is negative.

*9: 3 when byte (ear) is zero, 12 when the result is possible, and 13 when the result is negative.

*10: 4 + (a) when byte (eam) is zero, 13 + (a) when the result is positive, and 14 + (a) when the result is negative.

*11: 3 when word (AH) is zero, 12 when the result is possible, and 13 when the result is negative.

*12: 3 when word (ear) is zero, 16 when the result is possible, and 19 when the result is negative.

*13: 4 + (a) when word (eam) is zero, 17 + (a) when the result is positive, and 20 + (a) when the result is negative.

Note: Two cycle counts are given for overflows occurring from DIV or DIVW instructions, because the overflow may be detected before or after execution.

The contents of AL are destroyed when an overflow occurs from a DIV or DIVW instruction.

For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1h Logical 1 Instructions (Byte/Word) (39 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND	A,#imm8	2	2	0	0	byte (A) ← (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND	A,ear	2	3	1	0	byte (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND	A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND	ear,A	2	3	2	0	byte (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
AND	eam,A	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR	A,#imm8	2	2	0	0	byte (A) ← (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR	A,ear	2	3	1	0	byte (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR	A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR	ear,A	2	3	2	0	byte (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
OR	eam,A	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR	A,#imm8	2	2	0	0	byte (A) ← (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR	A,ear	2	3	1	0	byte (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR	A,eam	2+	4+(a)	0	(b)	byte (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR	ear,A	2	3	2	0	byte (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XOR	eam,A	2+	5+(a)	0	2×(b)	byte (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT	A	1	2	0	0	byte (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOT	ear	2	3	2	0	byte (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOT	eam	2+	5+(a)	0	2×(b)	byte (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW	A	1	2	0	0	word (A) ← (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	A,#imm16	3	2	0	0	word (A) ← (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW	A,ear	2	3	1	0	word (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW	A,eam	2+	4+(a)	0	(c)	word (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW	ear,A	2	3	2	0	word (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	eam,A	2+	5+(a)	0	2×(c)	word (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW	A	1	2	0	0	word (A) ← (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW	A,#imm16	3	2	0	0	word (A) ← (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW	A,ear	2	3	1	0	word (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW	A,eam	2+	4+(a)	0	(c)	word (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW	ear,A	2	3	2	0	word (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	-
ORW	eam,A	2+	5+(a)	0	2×(c)	word (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW	A	1	2	0	0	word (A) ← (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW	A,#imm16	3	2	0	0	word (A) ← (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW	A,ear	2	3	1	0	word (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW	A,eam	2+	4+(a)	0	(c)	word (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW	ear,A	2	3	2	0	word (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW	eam,A	2+	5+(a)	0	2×(c)	word (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW	A	1	2	0	0	word (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOTW	ear	2	3	2	0	word (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	-
NOTW	eam	2+	5+(a)	0	2×(c)	word (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1i Logical 2 Instructions (Long-Word) (6 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL	A,ear	2	6	2	0	long (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL	A,eam	2+	7+(a)	0	(d)	long (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL	A,ear	2	6	2	0	long (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL	A,eam	2+	7+(a)	0	(d)	long (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL	A,ear	2	6	2	0	long (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL	A,eam	2+	7+(a)	0	(d)	long (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

Table B.2.1j Sign Inversion Instructions (Byte/Word) (6 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG	A	1	2	0	0	byte (A) ← 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG	ear	2	3	2	0	byte (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG	eam	2+	5+(a)	0	2+(b)	byte (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW	A	1	2	0	0	word (A) ← 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW	ear	2	2	2	0	word (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW	eam	2+	5+(a)	0	2+(c)	word (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*

Table B.2.1k Normalize Instruction (Long-Word) (1 Instruction)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML	A,R0	2	*1	1	0	long (A) ← Shift to the position where 1 was formerly placed byte (R0) ← Number of shifts at that time	-	-	-	-	-	-	*	-	-	-

*1: 4 when the contents of the accumulator are all zeroes, 6 + (R0) in all other cases.

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1l Shift Instructions (Byte/Word/Long-Word) (18 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC	A	2	2	0	0	byte (A) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	-
ROLC	A	2	2	0	0	byte (A) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	-
RORC	ear	2	3	2	0	byte (ear) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	-
RORC	eam	2+	5+(a)	0	2×(b)	byte (eam) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	*
ROLC	ear	2	3	2	0	byte (ear) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	-
ROLC	eam	2+	5+(a)	0	2×(b)	byte (eam) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	*
ASR	A,RO	2	*1	1	0	byte (A) ← Arithmetic right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSR	A,RO	2	*1	1	0	byte (A) ← Logical right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSL	A,RO	2	*1	1	0	byte (A) ← Logical left barrel shift (A,R0)	-	-	-	-	-	*	*	-	*	-
ASRW	A	1	2	0	0	word (A) ← Arithmetic right shift (A,1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW	A /SHRW A	1	2	0	0	word (A) ← Logical right shift (A,1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW	A /SHLW A	1	2	0	0	word (A) ← Logical left shift (A,1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW	A,R0	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSRW	A,R0	2	*1	1	0	word (A) ← Logical right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSLW	A,R0	2	*1	1	0	word (A) ← Logical left barrel shift (A,R0)	-	-	-	-	-	*	*	-	*	-
ASRL	A,R0	2	*2	1	0	long (A) ← Arithmetic right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSRL	A,R0	2	*2	1	0	long (A) ← Logical right barrel shift (A,R0)	-	-	-	-	*	*	*	-	*	-
LSLL	A,R0	2	*2	1	0	long (A) ← Logical left barrel shift (A,R0)	-	-	-	-	-	*	*	-	*	-

*1: 6 when R0 is 0, 5 + (R0) in all other cases.

*2: 6 when R0 is 0, 6 + (R0) in all other cases.

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1m Branch 1 Instructions (31 Instructions)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ / BEQ rel	2	*1	0	0	Branch when (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ / BNE rel	2	*1	0	0	Branch when (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC / BLO rel	2	*1	0	0	Branch when (C) = 1	-	-	-	-	-	-	-	-	-	-
BNC / BHS rel	2	*1	0	0	Branch when (C) = 0	-	-	-	-	-	-	-	-	-	-
BN rel	2	*1	0	0	Branch when (N) = 1	-	-	-	-	-	-	-	-	-	-
BP rel	2	*1	0	0	Branch when (N) = 0	-	-	-	-	-	-	-	-	-	-
BV rel	2	*1	0	0	Branch when (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV rel	2	*1	0	0	Branch when (V) = 0	-	-	-	-	-	-	-	-	-	-
BT rel	2	*1	0	0	Branch when (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT rel	2	*1	0	0	Branch when (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT rel	2	*1	0	0	Branch when (V) xor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE rel	2	*1	0	0	Branch when (V) xor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE rel	2	*1	0	0	Branch when ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT rel	2	*1	0	0	Branch when ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS rel	2	*1	0	0	Branch when (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI rel	2	*1	0	0	Branch when (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA rel	2	*1	0	0	Unconditional branching	-	-	-	-	-	-	-	-	-	-
JMP @A	1	2	0	0	word (PC) ← (A)	-	-	-	-	-	-	-	-	-	-
JMP addr16	3	3	0	0	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
JMP @ear	2	3	1	0	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
JMP @eam	2+	4+(a)	0	(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
JMPP @ear *1	2	5	2	0	word (PC) ← (ear), (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP @eam *1	2+	6+(a)	0	(d)	word (PC) ← (eam), (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP addr24	4	4	0	0	word (PC) ← ad24 0-15, (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL @ear *2	2	6	1	(c)	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-	-
CALL @eam *2	2+	7+(a)	0	2×(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-	-
CALL addr16 *3	3	6	0	(c)	word (PC) ← addr16	-	-	-	-	-	-	-	-	-	-
CALLV #vct4 *3	1	7	0	2×(c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP @ear *4	2	10	2	2×(c)	word (PC) ← (ear) 0-15, (PCB) ← (ear)16-23	-	-	-	-	-	-	-	-	-	-
CALLP @eam *4	2+	11+(a)	0	*2	word (PC) ← (eam) 0-15, (PCB) ← (eam)16-23	-	-	-	-	-	-	-	-	-	-
CALLP addr24 *5	4	10	0	2×(c)	word (PC) ← addr0-15, (PCB) ← addr16-23	-	-	-	-	-	-	-	-	-	-

*1: 4 when branching, 3 when not branching.

*2: 3 × (c) + (b)

Note 1: Read (word) branch address.

Note 2: W: Save (word) into stack; R: read (word) branch address.

Note 3: Save (word) into stack.

Note 4: W: Save (long-word) into W stack; R: read (long-word) R branch address.

Note 5: Save (long-word) into stack.

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1n Branch 2 Instructions (19 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE	A,#imm8,rel	3	*1	0	0	Branch when byte (A) ≠ imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	A,#imm16,rel	4	*1	0	0	Branch when word (A)≠ imm16	-	-	-	-	-	*	*	*	*	-
CBNE	ear,#imm8,rel	4	*2	1	0	Branch when byte (ear)≠ imm8	-	-	-	-	-	*	*	*	*	-
CBNE	eam,#imm8,rel	4+	*3	0	(b)	Branch when byte (eam)≠ imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	ear,#imm16,rel	5	*4	1	0	Branch when word (ear)≠ imm16	-	-	-	-	-	*	*	*	*	-
CWBNE	eam,#imm16,rel	5+	*3	0	(c)	Branch when word (eam)≠ imm16	-	-	-	-	-	*	*	*	*	-
DBNZ	ear,rel	3	*5	2	0	Branch when byte (ear)=(ear)-1, (ear)≠ 0	-	-	-	-	-	*	*	*	-	-
DBNZ	eam,rel	3+	*6	2	2×(b)	Branch when byte (eam)=(eam)-1, (eam)≠ 0	-	-	-	-	-	*	*	*	-	*
DWBNZ	ear,rel	3	*5	2	0	Branch when word (ear)=(ear)-1, (ear)≠ 0	-	-	-	-	-	*	*	*	-	-
DWBNZ	eam,rel	3+	*6	2	2×(c)	Branch when word (eam)=(eam)-1, (eam)≠ 0	-	-	-	-	-	*	*	*	-	*
INT	#vct8	2	20	0	8×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT	addr16	3	16	0	6×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP	addr24	4	17	0	6×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9		1	20	0	8×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI		1	11	0	*7	Recovery from interrupt	-	-	*	*	*	*	*	*	*	-
LINK	#imm8	2	6	0	(c)	At the entrance of function, save old frame pointers into a stack, set up new frame pointers, reserve area for local pointers.	-	-	-	-	-	-	-	-	-	-
UNLINK		1	5	0	(c)	At the exit of function, recover the old frame pointers from the stack.	-	-	-	-	-	-	-	-	-	-
RET	*1	1	4	0	(c)	Recover from the subroutine.	-	-	-	-	-	-	-	-	-	-
RETP	*2	1	6	0	(d)	Recover from the subroutine.	-	-	-	-	-	-	-	-	-	-

*1: 5 when branching, 4 when not branching

*2: 13 when branching, 12 when not branching

*3: 7 + (a) when branching, 6 + (a) when not branching

*4: 8 when branching, 7 when not branching

*5: 7 when branching, 6 when not branching

*6: 8 + (a) when branching, 7 + (a) when not branching

*7: 3 × (b) + 2 × (c) when an interrupt request is generated, 6 × (c) at recovery.

Note 1: Return from stack (word)

Note 2: Return from stack (long)

Note 3: RWj+ addressing mode should not be used with the CBNE/CWBNE instructions.

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1o Other Control Instructions (Byte/Word/Long-Word) (36 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW	A	1	4	0	(c)	word (SP) ← (SP) -2, ((SP)) ← (A)	-	-	-	-	-	-	-	-	-	-
PUSHW	AH	1	4	0	(c)	word (SP) ← (SP) -2, ((SP)) ← (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW	PS	1	4	0	(c)	word (SP) ← (SP) -2, ((SP)) ← (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW	rlst	2	*3	+&	*4	(SP) ← (SP) - 2n, ((SP)) ← (rlst)	-	-	-	-	-	-	-	-	-	-
POPW	A	1	3	0	(c)	word (A) ← ((SP)), (SP) ← (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW	AH	1	3	0	(c)	word (AH) ← ((SP)), (SP) ← (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW	PS	1	4	0	(c)	word (PS) ← ((SP)), (SP) ← (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW	rlst	2	*2	+&	*4	(rlst) ← ((SP)), (SP) ← (SP)	-	-	-	-	-	-	-	-	-	-
JCTX	@A	1	14	0	6×(c)	Context switching instruction	-	-	*	*	*	*	*	*	*	-
AND	CCR,#imm8	2	3	0	0	byte (CCR) ← (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR	CCR,#imm8	2	3	0	0	byte (CCR) ← (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV	RP,#imm8	2	2	0	0	byte (RP) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV	ILM,#imm8	2	2	0	0	byte (ILM) ← imm8	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,ear	2	3	1	0	word (RWi) ← ear	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,eam	2+	2+(a)	1	0	word (RWi) ← eam	-	-	-	-	-	-	-	-	-	-
MOVEA	A,ear	2	1	0	0	word (A) ← ear	-	*	-	-	-	-	-	-	-	-
MOVEA	A,eam	2+	1+(a)	0	0	word (A) ← eam	-	*	-	-	-	-	-	-	-	-
ADDSP	#imm8	2	3	0	0	word (SP) ← ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP	#imm16	3	3	0	0	word (SP) ← imm16	-	-	-	-	-	-	-	-	-	-
MOV	A,brgl	2	*1	0	0	byte (A) ← (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV	brg2,A	2	1	0	0	byte (brg2) ← (A)	-	-	-	-	-	*	*	-	-	-
NOP		1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB		1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB		1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB		1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB		1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC		1	1	0	0	Prefix code for flag unchange setting	-	-	-	-	-	-	-	-	-	-
CMR		1	1	0	0	Prefix for common register banks	-	-	-	-	-	-	-	-	-	-

*1: PCB, ADB, SSB, USB, and SPB: ...1 cycle
DTB, DPR: .....2 cycles

*2: 7 + 3 × (pop count) + 2 × (last register number to be popped), 7 when RLST = 0

*3: 29 + 3 × (pop count) - 3 × (last register number to be popped), 8 when RLST = 0

*4: Pop count x (c), or push count x (c)

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1p Bit Manipulation Instructions (22 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB	A,dir:bp	3	5	0	(b)	byte (A) ← (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,addr16:bp	4	5	0	(b)	byte (A) ← (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,io:bp	3	4	0	(b)	byte (A) ← (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	dir:bp,A	3	7	0	2×(b)	bit (dir:bp)b ← (A)	-	-	-	-	-	*	*	-	-	*
MOVB	addr16:bp,A	4	7	0	2×(b)	bit (addr16:bp)b ← (A)	-	-	-	-	-	*	*	-	-	*
MOVB	io:bp,A	3	6	0	2×(b)	bit (io:bp)b ← (A)	-	-	-	-	-	*	*	-	-	*
SETB	dir:bp	3	7	0	2×(b)	bit (dir:bp)b ← 1	-	-	-	-	-	-	-	-	-	*
SETB	addr16:bp	4	7	0	2×(b)	bit (addr16:bp)b ← 1	-	-	-	-	-	-	-	-	-	*
SETB	io:bp	3	7	0	2×(b)	bit (io:bp)b ← 1	-	-	-	-	-	-	-	-	-	*
CLRB	dir:bp	3	7	0	2×(b)	bit (dir:bp)b ← 0	-	-	-	-	-	-	-	-	-	*
CLRB	addr16:bp	4	7	0	2×(b)	bit (addr16:bp)b ← 0	-	-	-	-	-	-	-	-	-	*
CLRB	io:bp	3	7	0	2×(b)	bit (io:bp)b ← 0	-	-	-	-	-	-	-	-	-	*
BBC	dir:bp,rel	4	*1	0	(b)	Branch when (dir:bp)b = 0	-	-	-	-	-	-	*	-	-	-
BBC	addr16:bp,rel	5	*1	0	(b)	Branch when (addr16:bp)b = 0	-	-	-	-	-	-	*	-	-	-
BBC	io:bp,rel	4	*2	0	(b)	Branch when (io:bp)b = 0	-	-	-	-	-	-	*	-	-	-
BBS	dir:bp,rel	4	*1	0	(b)	Branch when (dir:bp)b = 1	-	-	-	-	-	-	*	-	-	-
BBS	addr16:bp,rel	5	*1	0	(b)	Branch when (addr16:bp)b = 1	-	-	-	-	-	-	*	-	-	-
BBS	io:bp,rel	4	*2	0	(b)	Branch when (io:bp)b = 1	-	-	-	-	-	-	*	-	-	-
SBBS	addr16:bp,rel	5	*3	0	2×(b)	Branch when (addr16:bp) b = 1, bit = 1	-	-	-	-	-	-	*	-	-	*
WBTS	io:bp	3	*4	0	*5	Wait until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC	io:bp	3	*4	0	*5	Wait until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1: 8 when branching, 7 when not branching

*2: 7 when branching, 6 when not branching

*3: 10 when condition is satisfied, 9 when not satisfied

*4: Undefined count

*5: Until condition is satisfied

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

Table B.2.1q Accumulator Manipulation Instructions (Byte/Word) (6 Instructions)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 ←→ (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW / XCHW A,T	1	2	0	0	word (AH) ←→ (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	byte signed extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	word signed extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	word zero extension	-	Z	-	-	-	R	*	-	-	-

Table B.2.1r String Instructions (10 Instructions)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	+&	*3	byte transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	+&	*3	byte transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	+&	*4	byte search @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	+&	*4	byte search @AH- ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6	+&	*3	byte fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	+)	*6	word transfer @AH+ ← @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	+)	*6	word transfer @AH- ← @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	+)	*7	word search @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	+)	*7	word search @AH- ← AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6	+)	*6	word fill @AH+ ← AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

*1: 5 when RW0 is 0, 4 + 7 × (RW0) for count out, and 7n + 5 when match occurs.

*2: 5 when RW0 is 0, 4 + 8 × (RW0) in any other case.

*3: (b) × (RW0) + (b) × (RW0): when accessing a source and a destination in different areas, the value of item (b) should be computed separately for each.

*4: (b) × n

*5: 2 × (RW0)

*6: (c) × (RW0) + (c) × (RW0): when accessing a source and a destination in different areas, the value of item (c) should be computed separately for each.

*7: (c) × n

*8: 2 × (RW0)

m : RW0 value (counter value)

n : Loop count

Note: For an explanation of “(a)” to “(d)” in the colunm “B”, see Table B.2d and Table B.2e.

B.3 Instruction Map

Because the F^{2}MC-16LX operation codes each consist of one or two bytes, the instruction map consists of numerous pages. The structure of the instruction map is shown below.

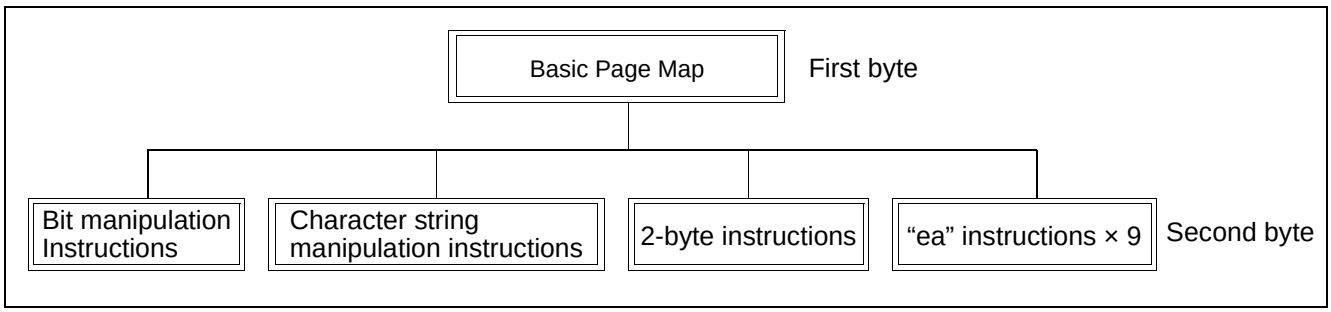
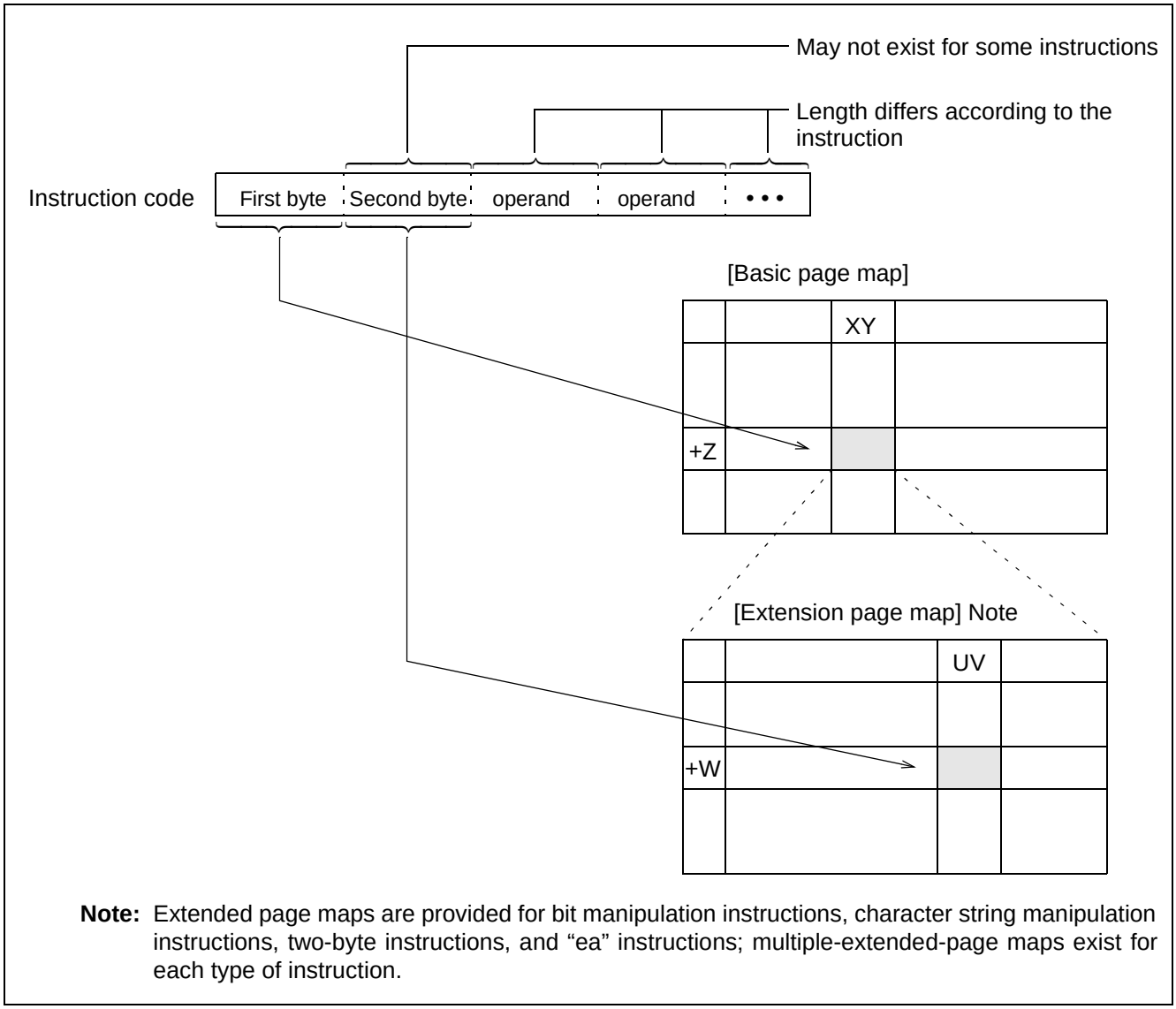


Fig. B.3a Structure of F^{2}MC-16LX Instruction Map

Instructions that consist of only one byte (such as NOP) are concluded on the basic page. Regarding instructions that require two bytes (such as MOVS), the existence of the map for the second byte is indicated when the first byte is referenced, so it is clear that it is necessary to use the following byte to reference the map for the second byte.

The correspondence between the actual instruction code and the instruction map is shown below.



Note: Extended page maps are provided for bit manipulation instructions, character string manipulation instructions, two-byte instructions, and “ea” instructions; multiple-extended-page maps exist for each type of instruction.

Fig. B.3b Correspondence between Actual Instructions and the Instruction Maps

B.3.1 Basic Page Map

Table B.3.1a Basic Page Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	NOP	CMR	ADD A, dir	ADD A, #8	MOV A, dir	MOV A, io	BRA rel	ea instructions (1)	MOV A, Ri	MOV Ri, A	MOV Ri, #8	MOVX A, Ri	MOVX A,@RWi+d8	MOVN A, #4	CALLV #4	BZ /BEQ rel
+1	INT9	NCC	SUB A, dir	SUB A, #8	MOV dir, A	MOV io, A	JMP @A	ea instructions (2)								BNZ/BNE rel
+2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A, #8	MOV A, addr16	JMP addr16	ea instructions (3)								BC /BLO rel
+3	NEG A	JCTX @A	CMP A	CMP A, #8	MOVX A, #8	MOV addr 16, A	JMPP addr24	ea instructions (4)								BNC/BHS rel
+4	PCB	EXT	AND CCR, #8	AND A, #8	MOV dir, #8	MOV io, #8	CALL addr16	ea instructions (5)								BN rel
+5	DTB	ZEXT	OR CCR, #8	OR A, #8	MOVX A, dir	MOVX A, io	CALLP addr24	ea instructions (6)								BP rel
+6	ADB	SWAP	DIVU A	XOR A, #8	MOVW A, SP	MOVW io, #16	RETP	ea instructions (7)								BV rel
+7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP, A	MOVX A, addr16	RET	ea instructions (8)								BNV rel
+8	LINK imm#8	ADDL A, #32	ADDW A	ADDW A, #16	MOVW A, dir	MOVW A, io	INT #vct8	ea instructions (9)	MOVW A, RWi	MOVW RWi, A	MOVW RWi, #16	MOVW @RWi+dB	MOVW @RWi+dB, A			BT rel
+9	UNLINK	SUBL A, #32	SUBW A	SUBW A, #16	MOVW dir, A	MOVW io, A	INT addr16	MOVEA RWi, ea								BNT rel
+A	MOV RP, #8	MOV ILM, #8	CBNE A, #8, rel	CWBNE A, #16, rel	MOVW A, #16	MOVW A, addr16	INTP addr24	MOV Ri, ea								BLT rel
+B	NEGW A	CMPL A, #32	CMPW A	CMPW A, #16	MOVL A, #32	MOVW addr16, A	RETI	MOVW RWi, ea								BGE rel
+C	LSLW A	EXTW	ANDW A	ANDW A, #16	PUSHW A	POPW A	Bit operation instructions	MOV ea, Ri								BLE rel
+D		ZEXTW	ORW A	ORW A, #16	PUSHW AH	POPW AH		MOVW ea, RWi								BGT rel
+E	ASRW A	SWAPW	XORW A	XORW A, #16	PUSHW PS	POPW PS	String operation instructions	XCH Ri, ea								BLS rel
+F	LSRW A	ADDSP #16	MULUW A	NOTW A	PUSHW rlst	POPW rlst	Two-byte instructions	XCRW RWi, ea								BHI rel

Table B.3.1b Bit Manipulation Instruction Map (First byte = 6 CH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVB A, io:bp		MOVB io:bp, A		CLRB io:bp		SETB io:bp		BBC io:bp, rel		BBS io:bp, rel		WBTS io:bp		WBTC io:bp	
+ 1																
+ 2																
+ 3																
+ 4																
+ 5																
+ 6																
+ 7																
+ 8	MOVB A, dir:bp	MOVB A, addr16:bp	MOVB dir: bp,A	MOVB addr16: bp, A	CLRB dir:bp	CLRB addr16:bp	SETB dir:bp	SETB addr16:bp	BBC dir:bp,rel	BBC ad16:bp, rel	BBS dir:bp, rel	BBS ad 16:bp, rel				SBBS addr16:bp
+ 9																
+ A																
+ B																
+C																
+D																
+ E																
+ F																

Table B.3.1c Character String Manipulation Instruction Map (First byte = 6EH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVSI PCB, PCB	MOVSD	MOVSWI	MOVSWD					SCEQI PCB	SCEQD PCB	SCWEQI PCB	SCWEQD PCB	FILSI PCB		FILSWI PCB	
+1	PCB, DTB								DTB	DTB	DTB	DTB	DTB		DTB	
+2	PCB, ADB								ADB	ADB	ADB	ADB	ADB		ADB	
+3	PCB, SPB								SPB	SPB	SPB	SPB	SPB		SPB	
+4	DTB, PCB															
+5	DTB, DTB															
+6	DTB, ADB															
+7	DTB, SPB															
+8	ADB, PCB															
+9	ADB, DTB															
+A	ADB, ADB															
+B	ADB, SPB															
+C	SPB, PCB															
+D	SPB, DTB															
+E	SPB, ADB															
+F	SPB, SPB															

Table B.3.1d Two-byte Instruction Map (First byte = 6FH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV A, DTB	MOV DTB, A	MOVX A, @RL0 +d8	MOV @RL0+d8, A	MOV A, @RL0+d8											
+1	MOV A, ADB	MOV ADB, A														
+2	MOV A, SSB	MOV SSB, A	MOVX A, @RL1+d8	MOV @RL1+d8, A	MOV A, @RL1+d8											
+3	MOV A, USB	MOV USB, A														
+4	MOV A. DPR	MOV DPR, A	MOVX A, @RL2+d8	MOV @RL2+d8, A	MOV A, @RL2+d8											
+5	MOV A. @A	MOV @AL, AH														
+6	MOV A, PCB	MOVX A, @A	MOVX A, @RL3+d8	MOV @RL3+d8, A	MOV A, @RL3+d8											
+7	ROLC A	RORC A														
+8				MOVW @RL0+d8, A	MOVW A, @RL0+d8			MUL A								
+9								MULW A								
+A				MOVW @RL1+d8, A	MOVW A, @RL1+d8			DIVU A								
+B																
+C	LSLW A, R0	LSLL A, R0	LSL A, R0	MOVW @RL2+d8, A	MOVW A, @RL2+d8											
+D	MOVW A, @A	MOVW @AL, AH	NRML A, R0													
+E	ASRW A, R0	ASRL A, R0	ASR A, R0	MOVW @RL3+d8, A	MOVW A, @RL3+d8											
+F	LSRW A, R0	LSRL A, R0	LSR A, R0													

Table B.3.1e "ea" Instructions 1 (First byte = 70H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDL A, RL0	ADDL A, @RW0+d8	SUBL A, RL0	SUBL A, @RW0+d8	CWBNE RW0, #16, rel	CWBNE @RW0+d8, #16, rel	CMPL A, RL0	CMPL A, @RW0+d8	ANDL A, RL0	ANDL A, @RW0+d8	ORL A, RL0	ORL A, @RW0+d8	XORL A, RL0	XORL A, @RW0+d8	CBNE R0, #8, rel	CBNE @RW0+d8, #8, rel
+1	ADDL A, RL0	ADDL A, @RW1+d8	SUBL A, RL0	SUBL A, @RW1+d8	CWBNE RW1, #16, rel	CWBNE @RW1+d8, #16, rel	CMPL A, RL0	CMPL A, @RW1+d8	ANDL A, RL0	ANDL A, @RW1+d8	ORL A, RL0	ORL A, @RW1+d8	XORL A, RL0	XORL A, @RW1+d8	CBNE R1, #8, rel	CBNE @RW1+d8, #8, rel
+2	ADDL A, RL1	ADDL A, @RW2+d8	SUBL A, RL1	SUBL A, @RW2+d8	CWBNE RW2, #16, rel	CWBNE @RW2+d8, #16, rel	CMPL A, RL1	CMPL A, @RW2+d8	ANDL A, RL1	ANDL A, @RW2+d8	ORL A, RL1	ORL A, @RW2+d8	XORL A, RL1	XORL A, @RW2+d8	CBNE R2, #8, rel	CBNE @RW2+d8, #8, rel
+3	ADDL A, RL1	ADDL A, @RW3+d8	SUBL A, RL1	SUBL A, @RW3+d8	CWBNE RW3, #16, rel	CWBNE @RW3+d8, #16, rel	CMPL A, RL1	CMPL A, @RW3+d8	ANDL A, RL1	ANDL A, @RW3+d8	ORL A, RL1	ORL A, @RW3+d8	XORL A, RL1	XORL A, @RW3+d8	CBNE R3, #8, rel	CBNE @RW3+d8, #8, rel
+4	ADDL A, RL2	ADDL A, @RW4+d8	SUBL A, RL2	SUBL A, @RW4+d8	CWBNE RW4, #16, rel	CWBNE @RW4+d8, #16, rel	CMPL A, RL2	CMPL A, @RW4+d8	ANDL A, RL2	ANDL A, @RW4+d8	ORL A, RL2	ORL A, @RW4+d8	XORL A, RL2	XORL A, @RW4+d8	CBNE R4, #8, rel	CBNE @RW4+d8, #8, rel
+5	ADDL A, RL2	ADDL A, @RW5+d8	SUBL A, RL2	SUBL A, @RW5+d8	CWBNE RW5, #16, rel	CWBNE @RW5+d8, #16, rel	CMPL A, RL2	CMPL A, @RW5+d8	ANDL A, RL2	ANDL A, @RW5+d8	ORL A, RL2	ORL A, @RW5+d8	XORL A, RL2	XORL A, @RW5+d8	CBNE R5, #8, rel	CBNE @RW5+d8, #8, rel
+6	ADDL A, RL3	ADDL A, @RW6+d8	SUBL A,RL3	SUBL A, @RW6+d8	CWBNE RW6, #16, rel	CWBNE @RW6+d8, #16, rel	CMPL A, RL3	CMPL A, @RW6+d8	ANDL A, RL3	ANDL A, @RW6+d8	ORL A, RL3	ORL A, @RW6+d8	XORL A, RL3	XORL A, @RW6+d8	CBNE R6, #8, rel	CBNE @RW6+d8, #8, rel
+7	ADDL A, RL3	ADDL A, @RW7+d8	SUBL A, RL3	SUBL A, @RW7+d8	CWBNE RW7, #16, rel	CWBNE @RW7+d8, #16, rel	CMPL A, RL3	CMPL A, @RW7+d8	ANDL A, RL3	ANDL A, @RW7+d8	ORL A, RL3	ORL A, @RW7+d8	XORL A, RL3	XORL A, @RW7+d8	CBNE R7, #8, rel	CBNE @RW7+d8, #8, rel
+8	ADDL A, @RW0	ADDL A, @RW0+d16	SUBL A, @RW0	SUBL A, @RW0+d16	CWBNE @ RW0, #16, rel	CWBNE @RW0+d16, #16, rel	CMPL A, @RW0	CMPL A, @RW0+d16	ANDL A, @RW0	ANDL A, @RW0+d16	ORL A, @RW0	ORL A, @RW0+d16	XORL A, @RW0	XORL A, @RW0+d16	CBNE @RW0, #8, rel	CBNE @RW0+d16, #8, rel
+9	ADDL A, @RW1	ADDL A, @RW1+d16	SUBL A, @RW1	SUBL A, @RW1+d16	CWBNE @ RW1, #16, rel	CWBNE @RW1, d16, #16, rel	CMPL A, @RW1	CMPL A, @RW1+d16	ANDL A, @RW1	ANDL A, @RW1+d16	ORL A, @RW1	ORL A, @RW1+d16	XORL A, @RW1	XORL A, @RW1+d16	CBNE @RW1, #8, rel	CBNE @RW1+d16, #8, rel
+A	ADDL A, @RW2	ADDL A, @RW2+d16	SUBL A, @RW2	SUBL A, @RW2+d16	CWBNE @ RW2, #16, rel	CWBNE @RW2+d16, #16, rel	CMPL A, @RW2	CMPL A, @RW2+d16	ANDL A, @RW2	ANDL A, @RW2+d16	ORL A, @RW2	ORL A, @RW2+d16	XORL A, @RW2	XORL A, @RW2+d16	CBNE @RW2, #8, rel	CBNEC @RW2+d16, #8, rel
+B	ADDL A, @RW3	ADDL A, @RW3+d16	SUBL A, @RW3	SUBL A, @RW3+d16	CWBNE @ RW3, #16, rel	CWBNE @RW3+d16, #16, rel	CMPL A, @RW3	CMPL A, @RW3+d16	ANDL A, @RW3	ANDL A, @RW3+d16	ORL A, @RW3	ORL A, @RW3+d16	XORL A, @RW3	XORL A, @RW3+d16	CBNE @RW3, #8, rel	CBNE @RW3+d16, #8, rel
+C	ADDL A, @RW0+	ADDL A, @RW0+RW7	SUBL A, @RW0+	SUBL A, @RW0+RW7	Prohibit	CWBNE @RW0+RW7 #16, rel	CMPL A, @RW0+	CMPL A, @RW0+RW7	ANDL A, @RW0+	ANDL A, @RW0+RW7	ORL A, @RW0+	ORL A, @RW0+RW7	XORL A, @RW0+	XORL A, @RW0+RW7	Prohibit	CBNE @RW0+RW7 #8, rel
+D	ADDL A, @RW1+	ADDL A, @RW1+RW7	SUBL A, @RW1+	SUBL A, @RW1+RW7	Prohibit	CWBNE @RW1+RW7 #16, rel	CMPL A, @RW1+	CMPL A, @RW1+RW7	ANDL A, @RW1+	ANDL A, @RW1+RW7	ORL A, @RW1+	ORL A, @RW1+RW7	XORL A, @RW1+	XORL A, @RW1+RW7	Prohibit	CBNE @RW1+RW7 #8, rel
+E	ADDL A, @RW2+	ADDL A, @PC+d16	SUBL A, @RW2+	SUBL A, @PC+d16	Prohibit	CWBNE @PC+d16, #16, rel	CMPL A, @RW2+	CMPL A, @PC+d16	ANDL A, @RW2+	ANDL A, @PC+d16	ORL A, @RW2+	ORL A, @PC+d16	XORL A, @RW2+	XORL A, @PC+d16	Prohibit	CBNE @PC+d16, #8, rel
+F	ADDL A, @RW3+	ADDL A, addr16	SUBL A, @RW3+	SUBL A, addr16	Prohibit	CWBNE addr16, #16, rel	CMPL A, @RW3+	CMPL A, addr16	ANDL A, @RW3+	ANDL A, addr16	ORL A, @RW3+	ORL A, addr16	XORL A, @RW3+	XORL A, addr16	Prohibit	CBNE addr16, #8, rel

Table B.3.1f “ea” Instructions 22 (First byte = 71H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMPP @RL0	JMPP @@RW0+d8	CALLP @RL0	CALLP @@RW0+d8	INCL RL0	INCL @RW0+d8	DECL RL0	DECL @RW0+d8	MOVL A, RL0	MOVL A, @RW0+d8	MOVL RL0, A	MOVL @RW0+d8, A	MOV R0, #8	MOV @RW0+d8, #8	MOVEA A, RW0	MOVEA A, @RW0+d8
+1	JMPP @RL0	JMPP @@RW1+d8	CALLP @RL0	CALLP @@RW1+d8	INCL RL0	INCL @RW1+d8	DECL RL0	DECL @RW1+d8	MOVL A, RL0	MOVL A, @RW1+d8	MOVL RL0, A	MOVL @RW1+d8, A	MOV R1, #8	MOV @RW1+d8, #8	MOVEA A, RW1	MOVEA A, @RW1+d8
+2	JMPP @RL1	JMPP @@RW2+d8	CALLP @RL1	CALLP @@RW2+d8	INCL RL1	INCL @RW2+d8	DECL RL1	DECL @RW2+d8	MOVL A, RL1	MOVL A, @RW2+d8	MOVL RL1, A	MOVL @RW2+d8, A	MOV R2, #8	MOV @RW2+d8, #8	MOVEA A, RW2	MOVEA A, @RW2+d8
+3	JMPP @RL1	JMPP @@RW3+d8	CALLP @RL1	CALLP @@RW3+d8	INCL RL1	INCL @RW3+d8	DECL RL1	DECL @RW3+d8	MOVL A, RL1	MOVL A, @RW3+d8	MOVL RL1, A	MOVL @RW3+d8, A	MOV R3, #8	MOV @RW3+d8, #8	MOVEA A, RW3	MOVEA A, @RW3+d8
+4	JMPP @RL2	JMPP @@RW4+d8	CALLP @RL2	CALLP @@RW4+d8	INCL RL2	INCL @RW4+d8	DECL RL2	DECL @RW4+d8	MOVL A, RL2	MOVL A, @RW4+d8	MOVL RL2, A	MOVL @RW4+d8, A	MOV R4, #8	MOV @RW4+d8, #8	MOVEA A, RW4	MOVEA A, @RW4+d8
+5	JMPP @RL2	JMPP @@RW5+d8	CALLP @RL2	CALLP @@RW5+d8	INCL RL2	INCL @RW5+d8	DECL RL2	DECL @RW5+d8	MOVL A, RL2	MOVL A, @RW5+d8	MOVL RL2, A	MOVL @RW5+d8, A	MOV R5, #8	MOV @RW5+d8, #8	MOVEA A, RW5	MOVEA A, @RW5+d8
+6	JMPP @RL3	JMPP @@RW6+d8	CALLP @RL3	CALLP @@RW6+d8	INCL RL3	INCL @RW6+d8	DECL RL3	DECL @RW6+d8	MOVL A, RL3	MOVL A, @RW6+d8	MOVL RL3, A	MOVL @RW6+d8, A	MOV R6, #8	MOV @RW6+d8, #8	MOVEA A, RW6	MOVEA A, @RW6+d8
+7	JMPP @RL3	JMPP @@RW7+d8	CALLP @RL3	CALLP @@RW7+d8	INCL RL3	INCL @RW7+d8	DECL RL3	DECL @RW7+d8	MOVL A, RL3	MOVL A, @RW7+d8	MOVL RL3, A	MOVL @RW7+d8, A	MOV R7, #8	MOV @RW7+d8, #8	MOVEA A, RW7	MOVEA A, @RW7+d8
+8	JMPP @@RW0	JMPP @@RW0+d16	CALLP @@RW0	CALLP @@RW0+d16	INCL @RW0	INCL @RW0+d16	DECL @RW0	DECL @RW0+d16	MOVL A, @RW0	MOVL A, @RW0+d16	MOVL @RW0, A	MOVL @RW0+d16, A	MOV @RW0, #8	MOV @RW0+d16, #8	MOVEA A, @RW0	MOVEA A, @RW0+d16
+9	JMPP @@RW1	JMPP @RW1+d16	CALLP @@RW1	CALLP @RW1+d16	INCL @RW1	INCL @RW1+d16	DECL @RW1	DECL @RW1+d16	MOVL A, @RW1	MOVL A, @RW1+d16	MOVL @RW1, A	MOVL @RW1+d16, A	MOV @RW1, #8	MOV @RW1+d16, #8	MOVEA A, @RW1	MOVEA A, @RW1+d16
+A	JMPP @@RW2	JMPP @RW2+d16	CALLP @@RW2	CALLP @RW2+d16	INCL @RW2	INCL @RW2+d16	DECL @RW2	DECL @RW2+d16	MOVL A, @RW2	MOVL A, @RW2+d16	MOVL @RW2, A	MOVL @RW2+d16, A	MOV @RW2, #8	MOV @RW2+d16, #8	MOVEA A, @RW2	MOVEA A, @RW2+d16
+B	JMPP @@RW3	JMPP @RW3+d16	CALLP @@RW3	CALLP @RW3+d16	INCL @RW3	INCL @RW3+d16	DECL @RW3	DECL @RW3+d16	MOVL A,@ RW3	MOVL A, @RW3+d16	MOVL @RW3, A	MOVL @RW3+d16, A	MOV @RW3, #8	MOV @RW3+d16, #8	MOVEA A, @RW3	MOVEA A, @RW3+d16
+C	JMPP @@RW0+	JMPP @RW0+RW7	CALLP @@RW0+	CALLP @RW0+RW7	INCL @RW0+	INCL @RW0+RW7	DECL @RW0+	DECL @RW0+RW7	MOVL A, @RW0+	MOVL A, @RW0+RW7	MOVL @RW0+, A	MOVL @RW0+RW7, A	MOV @RW0+, #8	MOV @RW0+RW7, #8	MOVEA A, @RW0+	MOVEA A, @RW0+RW7
+D	JMPP @@RW1+	JMPP @RW1+RW7	CALLP @@RW1+	CALLP @RW1+RW7	INCL @RW1+	INCL @RW1+RW7	DECL @RW1+	DECL @RW1+RW7	MOVL A, @RW1+	MOVL A, @RW1+RW7	MOVL @RW1+, A	MOVL @RW1+RW7, A	MOV @RW1+, #8	MOV @RW1+RW7, #8	MOVEA A, @RW1+	MOVEA A, @RW1+RW7
+E	JMPP @@RW2+	JMPP @@PC+d16	CALLP @@RW2+	CALLP @@PC+d16	INCL @RW2+	INCL @PC+d16	DECL @RW2+	DECL @PC+d16	MOVL A, @RW2+	MOVL A, @PC+d16	MOVL @RW2+, A	MOVL @PC+d16, A	MOV @RW2+, #8	MOV @PC+d16, #8	MOVEA A, @RW2+	MOVEA A, @PC+d16
+F	JMPP @@RW3+	JMPP @addr16	CALLP @@RW3+	CALLP @addr16	INCL @RW3+	INCL addr16	DECL @RW3+	DECL addr16	MOVL A, @RW3+	MOVL A, addr16	MOVL @RW3+, A	MOVL addr16, A	MOV @RW3+, #8	MOV addr16, #8	MOVEA A, @RW3+	MOVEA A, addr16

Table B.3.1g "ea" Instructions 3 (First byte = 72H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ROLC R0	ROLC @RW0+d8	RORC R0	RORC @RW0+d8	INC R0	INC @RW0+d8	DEC R0	DEC @RW0+d8	MOV A, R0	MOV A, @RW0+d8	MOV R0, A	MOV @R W0+d8, A	MOVX A, R0	MOVX A, @RW0+d8	XCH A, R0	XCH A, @RW0+d8
+1	ROLC R1	ROLC @RW1+d8	RORC R1	RORC @RW1+d8	INC R1	INC @RW1+d8	DEC R1	DEC @RW1+d8	MOV A, R1	MOV A, @RW1+d8	MOV R1, A	MOV @R W1+d8, A	MOVX A, R1	MOVX A, @RW1+d8	XCH A, R1	XCH A, @RW1+d8
+2	ROLC R2	ROLC @RW2+d8	RORC R2	RORC @RW2+d8	INC R2	INC @RW2+d8	DEC R2	DEC @RW2+d8	MOV A, R2	MOV A, @RW2+d8	MOV R2, A	MOV @R W2+d8, A	MOVX A, R2	MOVX A, @RW2+d8	XCH A, R2	XCH A, @RW2+d8
+3	ROLC R3	ROLC @RW3+d8	RORC R3	RORC @RW3+d8	INC R3	INC @RW3+d8	DEC R3	DEC @RW3+d8	MOV A, R3	MOV A, @RW3+d8	MOV R3, A	MOV @R W3+d8, A	MOVX A, R3	MOVX A, @RW3+d8	XCH A, R3	XCH A, @RW3+d8
+4	ROLC R4	ROLC @RW4+d8	RORC R4	RORC @RW4+d8	INC R4	INC @RW4+d8	DEC R4	DEC @RW4+d8	MOV A, R4	MOV A, @RW4+d8	MOV R4, A	MOV @R W4+d8, A	MOVX A, R4	MOVX A, @RW4+d8	XCH A, R4	XCH A, @RW4+d8
+5	ROLC R5	ROLC @RW5+d8	RORC R5	RORC @RW5+d8	INC R5	INC @RW5+d8	DEC R5	DEC @RW5+d8	MOV A, R5	MOV A, @RW5+d8	MOV R5, A	MOV @R W5+d8, A	MOVX A, R5	MOVX A, @RW5+d8	XCH A, R5	XCH A, @RW5+d8
+6	ROLC R6	ROLC @RW6+d8	RORC R6	RORC @RW6+d8	INC R6	INC @RW6+d8	DEC R6	DEC @RW6+d8	MOV A, R6	MOV A, @RW6+d8	MOV R6, A	MOV @R W6+d8, A	MOVX A, R6	MOVX A, @RW6+d8	XCH A, R6	XCH A, @RW6+d8
+7	ROLC R7	ROLC @RW7+d8	RORC R7	RORC @RW7+d8	INC R7	INC @RW7+d8	DEC R7	DEC @RW7+d8	MOV A, R7	MOV A, @RW7+d8	MOV R7, A	MOV @R W7+d8, A	MOVX A, R7	MOVX A, @RW7+d8	XCH A, R7	XCH A, @RW7+d8
+8	ROLC @RW0	ROLC @RW0+d16	RORC @RW0	RORC @RW0+d16	INC @RW0	INC @RW0+d16	DEC @RW0	DEC @RW0+d16	MOV A, @RW0	MOV A, @RW0+d16	MOV @RW0, A	MOV @R W0+d16, A	MOVX A, @RW0	MOVX A, @RW0+d16	XCH A, @RW0	XCH A, @RW0+d16
+9	ROLC @RW1	ROLC @RW1+d16	RORC @RW1	RORC @RW1+d16	INC @RW1	INC @RW1+d16	DEC @RW1	DEC @RW1+d16	MOV A, @RW1	MOV A, @RW1+d16	MOV @RW1, A	MOV @R W1+d16, A	MOVX A, @RW1	MOVX A, @RW1+d16	XCH A, @RW1	XCH A, @RW1+d16
+A	ROLC @RW2	ROLC @RW2+d16	RORC @RW2	RORC @RW2+d16	INC @RW2	INC @RW2+d16	DEC @RW2	DEC @RW2+d16	MOV A, @RW2	MOV A, @RW2+d16	MOV @RW2, A	MOV @R W2+d16, A	MOVX A, @RW2	MOVX A, @RW2+d16	XCH A, @RW2	XCH A, @RW2+d16
+B	ROLC @RW3	ROLC @RW3+d16	RORC @RW3	RORC @RW3+d16	INC @RW3	INC @RW3+d16	DEC @RW3	DEC @RW3+d16	MOV A, @RW3	MOV A, @RW3+d16	MOV @RW3, A	MOV @R W3+d16, A	MOVX A, @RW3	MOVX A, @RW3+d16	XCH A, @RW3	XCH A, @RW3+d16
+C	ROLC @RW0+	ROLC @RW0+RW7	RORC @RW0+	RORC @RW0+RW7	INC @RW0+	INC @RW0+RW7	DEC @RW0+	DEC @RW0+RW7	MOV A, @RW0+	MOV A, @RW0+RW7	MOV @RW0+, A	MOV @R W0+RW7, A	MOVX A, @RW0+	MOVX A, @RW0+RW7	XCH A, @RW0+	XCH A, @RW0+RW7
+D	ROLC @RW1+	ROLC @RW1+RW7	RORC @RW1+	RORC @RW1+RW7	INC @RW1+	INC @RW1+RW7	DEC @RW1+	DEC @RW1+RW7	MOV A, @RW1+	MOV A, @RW1+RW7	MOV @RW1+, A	MOV @R W1+RW7, A	MOVX A, @RW1+	MOVX A, @RW1+RW7	XCH A, @RW1+	XCH A, @RW1+RW7
+E	ROLC @RW2+	ROLC @PC+d16	RORC @RW2+	RORC @PC+d16	INC @RW2+	INC @PC+d16	DEC @RW2+	DEC @PC+d16	MOV A, @RW2+	MOV A, @PC+d16	MOV @RW2+, A	MOV @P C+d16, A	MOVX A, @RW2+	MOVX A, @PC+d16	XCH A, @RW2+	XCH A, @PC+d16
+F	ROLC @RW3+	ROLC addr16	RORC @RW3+	RORC addr16	INC @RW3+	INC addr16	DEC @RW3+	DEC addr16	MOV A, @RW3+	MOV A, addr16	MOV @RW3+, A	MOV addr16, A	MOVX A, @RW3+	MOVX A, addr16	XCH A, @RW3+	XCH A, addr16

Table B.3.1h "ea" Instructions 4 (First byte = 73H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMP @RW0	JMP @@RW0+d8	CALL @RW0	CALL @@RW0+d8	INCW RW0	INCW @RW0+d8	DECW RW0	DECW @RW0+d8	MOVW A, RW0	MOVW A, @RW0+d8	MOVW RW0, A	MOVW @RW0+d8, A	MOVW RW0, #16	MOVW @RW0+d8, #16	XCHW A, RW0	XCHW A, @RW0+d8
+1	JMP @RW1	JMP @@RW1+d8	CALL @RW1	CALL @@RW1+d8	INCW RW1	INCW @RW1+d8	DECW RW1	DECW @RW1+d8	MOVW A, RW1	MOVW A, @RW1+d8	MOVW RW1, A	MOVW @RW1+d8, A	MOVW RW1, #16	MOVW @RW1+d8, #16	XCHW A, RW1	XCHW A, @RW1+d8
+2	JMP @RW2	JMP @@RW2+d8	CALL @RW2	CALL @@RW2+d8	INCW RW2	INCW @RW2+d8	DECW RW2	DECW @RW2+d8	MOVW A, RW2	MOVW A, @RW2+d8	MOVW RW2, A	MOVW @RW2+d8, A	MOVW RW2, #16	MOVW @RW2+d8, #16	XCHW A, RW2	XCHW A, @RW2+d8
+3	JMP @RW3	JMP @@RW3+d8	CALL @RW3	CALL @@RW3+d8	INCW RW3	INCW @RW3+d8	DECW RW3	DECW @RW3+d8	MOVW A, RW3	MOVW A, @RW3+d8	MOVW RW3, A	MOVW @RW3+d8, A	MOVW RW3, #16	MOVW @RW3+d8, #16	XCHW A, RW3	XCHW A, @RW3+d8
+4	JMP @RW4	JMP @@RW4+d8	CALL @RW4	CALL @@RW4+d8	INCW RW4	INCW @RW4+d8	DECW RW4	DECW @RW4+d8	MOVW A, RW4	MOVW A, @RW4+d8	MOVW RW4, A	MOVW @RW4+d8, A	MOVW RW4, #16	MOVW @RW4+d8, #16	XCHW A, RW4	XCHW A, @RW4+d8
+5	JMP @RW5	JMP @@RW5+d8	CALL @RW5	CALL @@RW5+d8	INCW RW5	INCW @RW5+d8	DECW RW5	DECW @RW5+d8	MOVW A, RW5	MOVW A, @RW5+d8	MOVW RW5, A	MOVW @RW5+d8, A	MOVW RW5, #16	MOVW @RW5+d8, #16	XCHW A, RW5	XCHW A, @RW5+d8
+6	JMP @RW6	JMP @@RW6+d8	CALL @RW6	CALL @@RW6+d8	INCW RW6	INCW @RW6+d8	DECW RW6	DECW @RW6+d8	MOVW A, RW6	MOVW A, @RW6+d8	MOVW RW6, A	MOVW @RW6+d8, A	MOVW RW6, #16	MOVW @RW6+d8, #16	XCHW A, RW6	XCHW A, @RW6+d8
+7	JMP @RW7	JMP @@RW7+d8	CALL @RW7	CALL @@RW7+d8	INCW RW7	INCW @RW7+d8	DECW RW7	DECW @RW7+d8	MOVW A, RW7	MOVW A, @RW7+d8	MOVW RW7, A	MOVW @RW7+d8, A	MOVW RW7, #16	MOVW @RW7+d8, #16	XCHW A, RW7	XCHW A, @RW7+d8
+8	JMP @@RW0	JMP @@RW0+d16	CALL @@RW0	CALL @@RW0+d16	INCW @RW0	INCW @RW0+d16	DECW @RW0	DECW @RW0+d16	MOVW A, @RW0	MOVW A, @RW0+d16	MOVW @RW0, A	MOVW @RW0+d16, A	MOVW @RW0, #16	MOVW@RW0+d16, #16	XCHW A, @RW0	XCHW A, @RW0+d16
+9	JMP @@RW1	JMP @@RW1+d16	CALL @@RW1	CALL @@RW1+d16	INCW @RW1	INCW @RW1+d16	DECW @RW1	DECW @RW1+d16	MOVW A, @RW1	MOVW A, @RW1+d16	MOVW @RW1. A	MOVW @RW1+d16, A	MOVW @RW1, #16	MOVW@RW1+d16, #16	XCHW A, @RW1	XCHW A, @RW1+d16
+A	JMP @@RW2	JMP @@RW2+d16	CALL @@RW2	CALL @@RW2+d16	INCW @RW2	INCW @RW2+d16	DECW @RW2	DECW @RW2+d16	MOVW A, @RW2	MOVW A, @RW2+d16	MOVW @RW2, A	MOVW @RW2+d16, A	MOVW @RW2, #16	MOVW@ RW2+d16, #16	XCHW A, @RW2	XCHW A, @RW2+d16
+B	JMP @@RW3	JMP @@RW3+d16	CALL @@RW3	CALL @@RW3+d16	INCW @RW3	INCW @RW3+d16	DECW @RW3	DECW @RW3+d16	MOVW A, @RW3	MOVW A, @RW3+d16	MOVW @RW3, A	MOVW @RW3+d16, A	MOVW @RW3, #16	MOVW @RW3+d16, #16	XCHW A, @RW3	XCHW A, @RW3+d16
+C	JMP @@RW0+	JMP @@RW0+RW7	CALL @@RW0+	CALL @ @RW0+RW7	INCW @RW0+	INCW @RW0+RW7	DECW @RW0+	DECW @RW0+RW7	MOVW A, @RW0+	MOVW A, @RW0+RW7	MOVW @RW0+, A	MOVW @RW0+RW7, A	MOVW @ RW0+, #16	MOVW@RW0+RW7, #16	XCHW A, @RW0+	XCHW A, @RW0+RW7
+D	JMP @@RW1+	JMP @@RW1+RW7	CALL @@RW1+	CALL @ @RW1+RW7	INCW @RW1+	INCW @RW1+RW7	DECW @RW1+	DECW @RW1+RW7	MOVW A, @RW1+	MOVW A, @RW1+RW7	MOVW @RW1+, A	MOVW @RW1+RW7, A	MOVW @ RW1+, #16	MOVW @RW1+RW7, #16	XCHW A, @RW1+	XCHW A, @RW1+RW7
+E	JMP @@RW2+	JMP @@PC+d16	CALL @@RW2+	CALL @@PC+d16	INCW @RW2+	INCW @PC+d16	DECW @RW2+	DECW @PC+d16	MOVW A, @RW2+	MOVW A, @PC+d16	MOVW @RW2+, A	MOVW @PC+d16, A	MOVW @ RW2+, #16	MOVW @PC+d16, #16	XCHW A, @RW2+	XCHW A, @PC+d16
+F	JMP @@RW3+	JMP @addr16	CALL @@RW3+	CALL @addr16	INCW @RW3+	INCW addr16	DECW @RW3+	DECW addr16	MOVW A, @RW3+	MOVW A, addr16	MOVW @RW3+, A	MOVW addr16, A	MOVW @ RW3+, #16	MOVW addr16, #16	XCHW A, @RW3+	XCHW A, addr16

Table B.3.1i “ea” Instructions 5 (First byte = 74H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A, R0	ADD A, @RW0+d8	SUB A, R0	SUB A, @RW0+d8	ADDC A, R0	ADDC A, @RW0+d8	CMP A, R0	CMP A, @RW0+d8	AND A, R0	AND A, @RW0+d8	OR A, R0	OR A, @RW0+d8	XOR A, R0	XOR A, @RW0+d8	DBNZ R0, r	DBNZ @ RW0+d8, r
+1	ADD A, R1	ADD A, @RW1+d8	SUB A, R1	SUB A, @RW1+d8	ADDC A, R1	ADDC A, @RW1+d8	CMP A, R1	CMP A, @RW1+d8	AND A, R1	AND A, @RW1+d8	OR A, R1	OR A, @RW1+d8	XOR A, R1	XOR A, @RW1+d8	DBNZ R1, r	DBNZ @ RW1+d8, r
+2	ADD A, R2	ADD A, @RW2+d8	SUB A, R2	SUB A, @RW2+d8	ADDC A, R2	ADDC A, @RW2+d8	CMP A, R2	CMP A, @RW2+d8	AND A, R2	AND A, @RW2+d8	OR A, R2	OR A, @RW2+d8	XOR A, R2	XOR A, @RW2+d8	DBNZ R2, r	DBNZ @ RW2+d8, r
+3	ADD A, R3	ADD A, @RW3+d8	SUB A, R3	SUB A, @RW3+d8	ADDC A, R3	ADDC A, @RW3+d8	CMP A, R3	CMP A, @RW3+d8	AND A, R3	AND A, @RW3+d8	OR A, R3	OR A, @RW3+d8	XOR A, R3	XOR A, @RW3+d8	DBNZ R3, r	DBNZ @ RW3+d8, r
+4	ADD A, R4	ADD A, @RW4+d8	SUB A, R4	SUB A, @RW4+d8	ADDC A, R4	ADDC A, @RW4+d8	CMP A, R4	CMP A, @RW4+d8	AND A, R4	AND A, @RW4+d8	OR A, R4	OR A, @RW4+d8	XOR A, R4	XOR A, @RW4+d8	DBNZ R4, r	DBNZ @ RW4+d8, r
+5	ADD A, R5	ADD A, @RW5+d8	SUB A, R5	SUB A, @RW5+d8	ADDC A, R5	ADDC A, @RW5+d8	CMP A, R5	CMP A, @RW5+d8	AND A, R5	AND A, @RW5+d8	OR A, R5	OR A, @RW5+d8	XOR A, R5	XOR A, @RW5+d8	DBNZ R5, r	DBNZ @ RW5+d8, r
+6	ADD A, R6	ADD A, @RW6+d8	SUB A, R6	SUB A, @RW6+d8	ADDC A, R6	ADDC A, @RW6+d8	CMP A, R6	CMP A, @RW6+d8	AND A, R6	AND A, @RW6+d8	OR A, R6	OR A, @RW6+d8	XOR A, R6	XOR A, @RW6+d8	DBNZ R6, r	DBNZ @ RW6+d8, r
+7	ADD A, R7	ADD A, @RW7+d8	SUB A, R7	SUB A, @RW7+d8	ADDC A. R7	ADDC A, @RW7+d8	CMP A, R7	CMP A, @RW7+d8	AND A, R7	AND A, @RW7+d8	OR A, R7	OR A, @RW7+d8	XOR A, R7	XOR A, @RW7+d8	DBNZ R7, r	DBNZ @ RW7+d8, r
+8	ADD A, @RW0	ADD A, @RW0+d16	SUB A, @RW0	SUB A, @RW0+d16	ADDC A, @RW0	ADDC A, @RW0+d16	CMP A, @RW0	CMP A, @RW0+d16	AND A, @RW0	AND A, @RW0+d16	OR A, @RW0	OR A, @RW0+d16	XOR A, @RW0	XOR A, @RW0+d16	DBNZ @RW0, r	DBNZ RW0+d16, r
+9	ADD A, @RW1	ADD A, @RW1+d16	SUB A, @RW1	SUB A, @RW1+d16	ADDC A, @RW1	ADDC A, @RW1+d16	CMP A, @RW1	CMP A, @RW1+d16	AND A, @RW1	AND A, @RW1+d16	OR A, @RW1	OR A, @RW1+d16	XOR A, @RW1	XOR A, @RW1+d16	DBNZ @RW1, r	DBNZ RW1+d16, r
+A	ADD A, @RW2	ADD A, @RW2+d16	SUB A, @RW2	SUB A, @RW2+d16	ADDC A, @RW2	ADDC A, @RW2+d16	CMP A, @RW2	CMP A, @RW2+d16	AND A, @RW2	AND A, @RW2+d16	OR A, @RW2	OR A, @RW2+d16	XOR A, @RW2	XOR A, @RW2+d16	DBNZ @RW2, r	DBNZ RW2+d16, r
+B	ADD A, @RW3	ADD A, @RW3+d16	SUB A, @RW3	SUB A, @RW3+d16	ADDC A, @RW3	ADDC A, @RW3+d16	CMP A, @RW3	CMP A, @RW3+d16	AND A, @RW3	AND A, @RW3+d16	OR A, @RW3	OR A, @RW3+d16	XOR A, @RW3	XOR A, @RW3+d16	DBNZ @RW3, r	DBNZ RW3+d16, r
+C	ADD A, @RW0+	ADD A, @RW0+RW7	SUB A, @RW0+	SUB A, @RW0+RW7	ADDC A, @RW0+	ADDC A, @RW0+RW7	CMP A, @RW0+	CMP A, @RW0+RW7	AND A, @RW0+	AND A, @RW0+RW7	OR A, @RW0+	OR A, @RW0+RW7	XOR A, @RW0+	XOR A, @RW0+RW7	DBNZ @RW0+, r	DBNZ RW0+RW7, r
+D	ADD A, @RW1+	ADD A, @RW1+RW7	SUB A, @RW1+	SUB A, @RW1+RW7	ADDC A, @RW1+	ADDC A, @RW1+RW7	CMP A, @RW1+	CMP A, @RW1+RW7	AND A, @RW1+	AND A, @RW1+RW7	OR A, @RW1+	OR A, @RW1+RW7	XOR A, @RW1+	XOR A, @RW1+RW7	DBNZ @RW1+, r	DBNZ RW1+RW7, r
+E	ADD A, @RW2+	ADD A, @PC+d16	SUB A, @RW2+	SUB A, @PC+d16	ADDC A, @RW2+	ADDC A, @PC+d16	CMP A, @RW2+	CMP A, @PC+d16	AND A, @RW2+	AND A, @PC+d16	OR A, @RW2+	OR A, @ PC+d16	XOR A, @RW2+	XOR A, @PC+d16	DBNZ @RW2+, r	DBNZ @ PC+d16, r
+F	ADD A, @RW3+	ADD A, addr16	SUB A, @RW3+	SUB A, addr16	ADDC A, @RW3+	ADDC A, addr16	CMP A, @RW3+	CMP A, addr16	AND A, @RW3+	AND A, addr16	OR A, @RW3+	OR A, addr16	XOR A, @RW3+	XOR A, addr16	DBNZ @RW3+, r	DBNZ @ addr16, r

Table B.3.1j “ea” Instructions 6 (First byte = 75H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD R0, A	ADD @RW0+d8, A	SUB R0, A	SUB @RW0+d8, A	SUBC A, R0	SUBC A, @RW0+d8	NEG R0	NEG @RW0+d8	AND R0, A	AND @RW0+d8, A	OR R0, A	OR @RW0+d8, A	XOR R0, A	XOR @RW0+d8, A	NOT R0	NOT @RW0+d8
+1	ADD R1, A	ADD @RW1+d8, A	SUB R1, A	SUB @RW1+d8, A	SUBC A, R1	SUBC A, @RW1+d8	NEG R1	NEG @RW1+d8	AND R1, A	AND @RW1+d8, A	OR R1, A	OR @RW1+d8, A	XOR R1, A	XOR @RW1+d8, A	NOT R1	NOT @RW1+d8
+2	ADD R2, A	ADD @RW2+d8, A	SUB R2, A	SUB @RW2+d8, A	SUBC A, R2	SUBC A, @RW2+d8	NEG R2	NEG @RW2+d8	AND R2, A	AND @RW2+d8, A	OR R2, A	OR @RW2+d8, A	XOR R2, A	XOR @RW2+d8, A	NOT R2	NOT @RW2+d8
+3	ADD R3, A	ADD @RW3+d8, A	SUB R3, A	SUB @RW3+d8, A	SUBC A, R3	SUBC A, @RW3+d8	NEG R3	NEG @RW3+d8	AND R3, A	AND @RW3+d8, A	OR R3, A	OR @RW3+d8, A	XOR R3, A	XOR @RW3+d8, A	NOT R3	NOT @RW3+d8
+4	ADD R4, A	ADD @RW4+d8, A	SUB R4, A	SUB @RW4+d8, A	SUBC A, R4	SUBC A, @RW4+d8	NEG R4	NEG @RW4+d8	AND R4, A	AND @RW4+d8, A	OR R4, A	OR @RW4+d8, A	XOR R4, A	XOR @RW4+d8, A	NOT R4	NOT @RW4+d8
+5	ADD R5, A	ADD @RW5+d8, A	SUB R5, A	SUB @RW5+d8, A	SUBC A, R5	SUBC A, @RW5+d8	NEG R5	NEG @RW5+d8	AND R5, A	AND @RW5+d8, A	OR R5, A	OR @RW5+d8, A	XOR R5, A	XOR @RW5+d8, A	NOT R5	NOT @RW5+d8
+6	ADD R6, A	ADD @RW6+d8, A	SUB R6, A	SUB @RW6+d8, A	SUBC A, R6	SUBC A, @RW6+d8	NEG R6	NEG @RW6+d8	AND R6, A	AND @RW6+d8, A	OR R6, A	OR @RW6+d8, A	XOR R6, A	XOR @RW6+d8, A	NOT R6	NOT @RW6+d8
+7	ADD R7, A	ADD @RW7+d8, A	SUB R7, A	SUB @RW7+d8, A	SUBC A, R7	SUBC A, @RW7+d8	NEG R7	NEG @RW7+d8	AND R7, A	AND @RW7+d8, A	OR R7, A	OR @RW7+d8, A	XOR R7, A	XOR @RW7+d8, A	NOT R7	NOT @RW7+d8
+8	ADD @RW0, A	ADD @RW0+d16, A	SUB @RW0, A	SUB @RW0+d16, A	SUBC A, @RW0	SUBC A, @RW0+d16	NEG @RW0	NEG @RW0+d16	AND @RW0, A	AND @RW0+d16, A	OR @RW0, A	OR @RW0+d16, A	XOR @RW0, A	XOR @RW0+d16, A	NOT @RW0	NOT @RW0+d16
+9	ADD @RW1, A	ADD @RW1+d16, A	SUB @RW1, A	SUB @RW1+d16, A	SUBC A, @RW1	SUBC A, @RW1+d16	NEG @RW1	NEG @RW1+d16	AND @RW1, A	AND @RW1+d16, A	OR @RW1, A	OR @RW1+d16, A	XOR @RW1, A	XOR @RW1+d16, A	NOT @RW1	NOT @RW1+d16
+A	ADD @RW2, A	ADD @RW2+d16, A	SUB @RW2, A	SUB @RW2+d16, A	SUBC A, @RW2	SUBC A, @RW2+d16	NEG @RW2	NEG @RW2+d16	AND @RW2, A	AND @RW2+d16, A	OR @RW2, A	OR @RW2+d16, A	XOR @RW2, A	XOR @RW2+d16, A	NOT @RW2	NOT @RW2+d16
+B	ADD @RW3, A	ADD @RW3+d16, A	SUB @RW3, A	SUB @RW3+d16, A	SUBC A, @RW3	SUBC A, @RW3+d16	NEG @RW3	NEG @RW3+d16	AND @RW3, A	AND @RW3+d16, A	OR @RW3, A	OR @RW3+d16, A	XOR @RW3, A	XOR @RW3+d16, A	NOT @RW3	NOT @RW3+d16
+C	ADD @RW0+, A	ADD @W0+RW7, A	SUB @RW0+, A	SUB @W0+RW7, A	SUBC A, @RW0+	SUBC A, @RW0+RW7	NEG @RW0+	NEG @RW0+RW7	AND @RW0+, A	AND @W0+RW7, A	OR @RW0+, A	OR @W0+RW7, A	XOR @RW0+, A	XOR @W0+RW7, A	NOT @RW0+	NOT @RW0+RW7
+D	ADD @RW1+, A	ADD @W1+RW7, A	SUB @RW1+, A	SUB @W1+RW7, A	SUBC A, @RW1+	SUBC A, @RW1+RW7	NEG @RW1+	NEG @RW1+RW7	AND @RW1+, A	AND @W1+RW7, A	OR @RW1+, A	OR @W1+RW7, A	XOR @RW1+, A	XOR @W1+RW7, A	NOT @RW1+	NOT @RW1+RW7
+E	ADD @RW2+, A	ADD @PC+d16, A	SUB @RW2+, A	SUB @PC+d16, A	SUBC A, @RW2+	SUBC A, @PC+d16	NEG @RW2+	NEG @PC+d16	AND @RW2+, A	AND @PC+d16, A	OR @RW2+, A	OR @PC+d16, A	XOR @RW2+, A	XOR @PC+d16, A	NOT @RW2+	NOT @PC+d16
+F	ADD @RW3+, A	ADD addr16, A	SUB @RW3+, A	SUB addr16, A	SUBC A, @RW3+	SUBC A, addr16	NEG @RW3+	NEG addr16	AND @RW3+, A	AND addr16, A	OR @RW3+, A	OR addr16, A	XOR @RW3+, A	XOR addr16, A	NOT @RW3+	NOT addr16

Table B.3.1k "ea" Instructions 7 (First byte = 76H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW A, RW0	ADDW A, @RW0+d8	SUBW A, RW0	SUBW A, @RW0+d8	ADDCW A, RW0	ADDCW A, @RW0+d8	CMPW A, RW0	CMPW A, @RW0+d8	ANDW A, RW0	ANDW A, @RW0+d8	ORW A, RW0	ORW A, @RW0+d8	XORW A, RW0	XORW A, @RW0+d8	DWBNZ RW0, r	DWBNZ @ RW0+d8, r
+1	ADDW A, RW1	ADDW A, @RW1+d8	SUBW A, RW1	SUBW A, @RW1+d8	ADDCW A, RW1	ADDCW A, @RW1+d8	CMPW A, RW1	CMPW A, @RW1+d8	ANDW A, RW1	ANDW A, @RW1+d8	ORW A, RW1	ORW A, @RW1+d8	XORW A, RW1	XORW A, @RW1+d8	DWBNZ RW1, r	DWBNZ @ RW1+d8, r
+2	ADDW A, RW2	ADDW A, @RW2+d8	SUBW A, RW2	SUBW A, @RW2+d8	ADDCW A, RW2	ADDCW A, @RW2+d8	CMPW A, RW2	CMPW A, @RW2+d8	ANDW A, RW2	ANDW A, @RW2+d8	ORW A, RW2	ORW A, @RW2+d8	XORW A, RW2	XORW A, @RW2+d8	DWBNZ RW2, r	DWBNZ @ RW2+d8, r
+3	ADDW A, RW3	ADDW A, @RW3+d8	SUBW A, RW3	SUBW A, @RW3+d8	ADDCW A, RW3	ADDCW A, @RW3+d8	CMPW A, RW3	CMPW A, @RW3+d8	ANDW A, RW3	ANDW A, @RW3+d8	ORW A, RW3	ORW A, @RW3+d8	XORW A, RW3	XORW A, @RW3+d8	DWBNZ RW3, r	DWBNZ @ RW3+d8, r
+4	ADDW A, RW4	ADDW A, @RW4+d8	SUBW A, RW4	SUBW A, @RW4+d8	ADDCW A, RW4	ADDCW A, @RW4+d8	CMPW A, RW4	CMPW A, @RW4+d8	ANDW A, RW4	ANDW A, @RW4+d8	ORW A, RW4	ORW A, @RW4+d8	XORW A, RW4	XORW A, @RW4+d8	DWBNZ RW4, r	DWBNZ @ RW4+d8, r
+5	ADDW A, RW5	ADDW A, @RW5+d8	SUBW A, RW5	SUBW A, @RW5+d8	ADDCW A, RW5	ADDCW A, @RW5+d8	CMPW A, RW5	CMPW A, @RW5+d8	ANDW A, RW5	ANDW A, @RW5+d8	ORW A, RW5	ORW A, @RW5+d8	XORW A, RW5	XORW A, @RW5+d8	DWBNZ RW5, r	DWBNZ @ RW5+d8, r
+6	ADDW A, RW6	ADDW A, @RW6+d8	SUBW A, RW6	SUBW A, @RW6+d8	ADDCW A, RW6	ADDCW A, @RW6+d8	CMPW A, RW6	CMPW A, @RW6+d8	ANDW A, RW6	ANDW A, @RW6+d8	ORW A, RW6	ORW A, @RW6+d8	XORW A, RW6	XORW A, @RW6+d8	DWBNZ RW6, r	DWBNZ @ RW6+d8, r
+7	ADDW A, RW7	ADDW A, @RW7+d8	SUBW A, RW7	SUBW A, @RW7+d8	ADDCW A, RW7	ADDCW A, @RW7+d8	CMPW A, RW7	CMPW A, @RW7+d8	ANDW A, RW7	ANDW A, @RW7+d8	ORW A, RW7	ORW A, @RW7+d8	XORW A, RW7	XORW A, @RW7+d8	DWBNZ RW7, r	DWBNZ @ RW7+d8, r
+8	ADDW A, @RW0	ADDW A, @RW0+d16	SUBW A, @RW0	SUBW A, @RW0+d16	ADDCW A, @RW0	ADDCW A, @RW0+d16	CMPW A, @RW0	CMPW A, @RW0+d16	ANDW A, @RW0	ANDW A, @RW0+d16	ORW A, @RW0	ORW A, @RW0+d16	XORW A, @RW0	XORW A, @RW0+d16	DWBNZ @RW0, r	DWBNZ @R W0+d16, r
+9	ADDW A, @RW1	ADDW A, @RW1+d16	SUBW A, @RW1	SUBW A, @RW1+d16	ADDCW A, @RW1	ADDCW A, @RW1+d16	CMPW A, @RW1	CMPW A, @RW1+d16	ANDW A, @RW1	ANDW A, @RW1+d16	ORW A, @RW1	ORW A, @RW1+d16	XORW A, @RW1	XORW A, @RW1+d16	DWBNZ @RW1, r	DWBNZ @R W1+d16, r
+A	ADDW A, @RW2	ADDW A, @RW2+d16	SUBW A, @RW2	SUBW A, @RW2+d16	ADDCW A, @RW2	ADDCW A, @RW2+d16	CMPW A, @RW2	CMPW A, @RW2+d16	ANDW A, @RW2	ANDW A, @RW2+d16	ORW A, @RW2	ORW A, @RW2+d16	XORW A, @RW2	XORW A, @RW2+d16	DWBNZ @RW2, r	DWBNZ @R W2+d16, r
+B	ADDW A, @RW3	ADDW A, @RW3+d16	SUBW A, @RW3	SUBW A, @RW3+d16	ADDCW A, @RW3	ADDCW A, @RW3+d16	CMPW A, @RW3	CMPW A, @RW3+d16	ANDW A, @RW3	ANDW A, @RW3+d16	ORW A, @RW3	ORW A, @RW3+d16	XORW A, @RW3	XORW A, @RW3+d16	DWBNZ @RW3, r	DWBNZ @R W3+d16, r
+C	ADDW A, @RW0+	ADDW A, @RW0+RW7	SUBW A, @RW0+	SUBW A, @RW0+RW7	ADDCW A, @RW0+	ADDCW A, @RW0+RW7	CMPW A, @RW0+	CMPW A, @RW0+RW7	ANDW A, @RW0+	ANDW A, @RW0+RW7	ORW A, @RW0+	ORW A, @RW0+RW7	XORW A, @RW0+	XORW A, @RW0+RW7	DWBNZ @RW0+, r	DWBNZ @R W0+RW7, r
+D	ADDW A, @RW1+	ADDW A, @RW1+RW7	SUBW A, @RW1+	SUBW A, @RW1+RW7	ADDCW A, @RW1+	ADDCW A, @RW1+RW7	CMPW A, @RW1+	CMPW A, @RW1+RW7	ANDW A, @RW1+	ANDW A, @RW1+RW7	ORW A, @RW1+	ORW A, @RW1+RW7	XORW A, @RW1+	XORW A, @RW1+RW7	DWBNZ @RW1+, r	DWBNZ @R W1+RW7, r
+E	ADDW A, @RW2+	ADDW A, @PC+d16	SUBW A, @RW2+	SUBW A, @PC+d16	ADDCW A, @RW2+	ADDCW A, @PC+d16	CMPW A, @RW2+	CMPW A, @PC+d16	ANDW A, @RW2+	ANDW A, @PC+d16	ORW A, @RW2+	ORW A, @ PC+d16	XORW A, @RW2+	XORW A, @PC+d16	DWBNZ @RW2+, r	DWBNZ @ PC+d16, r
+F	ADDW A, @RW3+	ADDW A, addr16	SUBW A, @RW3+	SUBW A, addr16	ADDCW A, @RW3+	ADDCW A, addr16	CMPW A, @RW3+	CMPW A, addr16	ANDW A, @RW3+	ANDW A, addr16	ORW A, @RW3+	ORW A, addr16	XORW A, @RW3+	XORW A, addr16	DWBNZ @RW3+, r	DWBNZ addr16, r

Table B.3.1l "ea" Instructions 8 (First byte = 77H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW RW0, A	ADDW @R W0+d8, A	SUBW RW0, A	SUBW @R W0+d8, A	SUBCW A, RW0	SUBCW A, @RW0+d8	NEGW RW0	NEGW @RW0+d8	ANDW RW0, A	ANDW @R W0+d8, A	ORW RW0, A	ORW @R W0+d8, A	XORW RW0, A	XORW @R W0+d8, A	NOTW RW0	NOTW @RW0+d8
+1	ADDW RW1, A	ADDW @R W1+d8, A	SUBW RW1, A	SUBW @R W1+d8, A	SUBCW A, RW1	SUBCW A, @RW1+d8	NEGW RW1	NEGW @RW1+d8	ANDW RW1, A	ANDW @R W1+d8, A	ORW RW1, A	ORW @R W1+d8, A	XORW RW1, A	XORW @R W1+d8, A	NOTW RW1	NOTW @RW1+d8
+2	ADDW RW2, A	ADDW @R W2+d8, A	SUBW RW2, A	SUBW @R W2+d8, A	SUBCW A, RW2	SUBCW A, @RW2+d8	NEGW RW2	NEGW @RW2+d8	ANDW RW2, A	ANDW @R W2+d8, A	OWR RW2, A	ORW @R W2+d8, A	XORW RW2, A	XORW @R W2+d8, A	NOTW RW2	NOTW @RW2+d8
+3	ADDW RW3, A	ADDW @R W3+d8, A	SUBW RW3, A	SUBW @R W3+d8, A	SUBCW A, RW3	SUBCW A, @RW3+d8	NEGW RW3	NEGW @RW3+d8	ANDW RW3, A	ANDW @R W3+d8, A	ORW RW3, A	ORW @R W3+d8, A	XORW RW3, A	XORW @R W3+d8, A	NOTW RW3	NOTW @RW3+d8
+4	ADDW RW4, A	ADDW @R W4+d8, A	SUBW RW4, A	SUBW @R W4+d8, A	SUBCW A, RW4	SUBCW A, @RW4+d8	NEGW RW4	NEGW @RW4+d8	ANDW RW4, A	ANDW @R W4+d8, A	ORW RW4, A	ORW @R W4+d8, A	XORW RW4, A	XORW @R W4+d8, A	NOTW RW4	NOTW @RW4+d8
+5	ADDW RW5, A	ADDW @R W5+d8, A	SUBW RW5, A	SUBW @R W5+d8, A	SUBCW A, RW5	SUBCW A, @RW5+d8	NEGW RW5	NEGW @RW5+d8	ANDW RW5, A	ANDW @R W5+d8, A	ORW RW5, A	ORW @R W5+d8, A	XORW RW5, A	XORW @R W5+d8, A	NOTW RW5	NOTW @RW5+d8
+6	ADDW RW6, A	ADDW @R W6+d8, A	SUBW RW6, A	SUBW @R W6+d8, A	SUBCW A, RW6	SUBCW A, @RW6+d8	NEGW RW6	NEGW @RW6+d8	ANDW RW6, A	ANDW @R W6+d8, A	ORW RW6, A	ORW @R W6+d8, A	XORW RW6, A	XORW @R W6+d8, A	NOTW RW6	NOTW @RW6+d8
+7	ADDW RW7, A	ADDW @R W7+d8, A	SUBW RW7, A	SUBW @R W7+d8, A	SUBCW A,RW7	SUBCW A, @RW7+d8	NEGW RW7	NEGW @RW7+d8	ANDW RW7, A	ANDW @R W7+d8, A	ORW RW7, A	ORW @R W7+d8, A	XORW RW7, A	XORW @R W7+d8, A	NOTW RW7	NOTW @RW7+d8
+8	ADDW @RW0, A	ADDW @R W0+d16, A	SUBW @RW0, A	SUBW @R W0+d16, A	SUBCW A, @RW0	SUBCW A, @RW0+d16	NEGW @RW0	NEGW @RW0+d16	ANDW @RW0, A	ANDW @R W0+d16, A	ORW @RW0, A	ORW @R W0+d16, A	XORW @RW0, A	XORW @R W0+d16, A	NOTW @RW0	NOTW @RW0+d16
+9	ADDW @RW1, A	ADDW @R W1+d16, A	SUBW @RW1, A	SUBW @R W1+d16, A	SUBCW A, @RW1	SUBCW A, @RW1+d16	NEGW @RW1	NEGW @RW1+d16	ANDW @RW1, A	ANDW @R W1+d16, A	ORW @RW1, A	ORW @R W1+d16, A	XORW @RW1, A	XORW @R W1+d16, A	NOTW @RW1	NOTW @RW1+d16
+A	ADDW @RW2, A	ADDW @R W2+d16, A	SUBW @RW2, A	SUBW @R W2+d16, A	SUBCW A, @RW2	SUBCW A, @RW2+d16	NEGW @RW2	NEGW @RW2+d16	ANDW @RW2, A	ANDW @R W2+d16, A	ORW @RW2, A	ORW @R W2+d16, A	XORW @RW2, A	XORW @R W2+d16, A	NOTW @RW2	NOTW @RW2+d16
+B	ADDW @RW3, A	ADDW @R W3+d16, A	SUBW @RW3, A	SUBW @R W3+d16, A	SUBCW A, @RW3	SUBCW A, @RW3+d16	NEGW @RW3	NEGW @RW3+d16	ANDW @RW3, A	ANDW @R W3+d16, A	ORW @RW3, A	ORW @R W3+d16, A	XORW @RW3, A	XORW @R W3+d16, A	NOTW @RW3	NOTW @RW3+d16
+C	ADDW @RW0+ A	ADDW @R W0+RW7	SUBW @RW0+, A	SUBW @R W0+RW7, A	SUBCW A, @RW0+	SUBCW A, @RW0+RW7	NEGW @RW0+	NEGW @RW0+RW7	ANDW @RW0+, A	ANDW @R W0+RW7, A	ORW @RW0+, A	ORW @R W0+RW7, A	XORW @RW0+, A	XORW @R W0+RW7, A	NOTW @RW0+	NOTW @RW0+RW7
+D	ADDW @RW1+, A	ADDW @R W1+RW7	SUBW @RW1+, A	SUBW @R W1+RW7, A	SUBCW A, @RW1+	SUBCW A, @RW1+RW7	NEGW @RW1+	NEGW @RW1+RW7	ANDW @RW1+, A	ANDW @R W1+RW7, A	ORW @RW1+, A	ORW @R W1+RW7, A	XORW @RW1+, A	XORW @R W1+RW7, A	NOTW @RW1+	NOTW @RW1+RW7
+E	ADDW @RW2+, A	ADDW @P C+d16, A	SUBW @RW2+, A	SUBW @P C+d16, A	SUBCW A, @RW2+	SUBCW A, @PC+d16	NEGW @RW2+	NEGW @PC+d16	ANDW @RW2+, A	ANDW @P C+d16, A	ORW @RW2+, A	ORW @P C+d16, A	XORW @RW2+, A	XORW @P C+d16, A	NOTW @RW2+	NOTW @PC+d16
+F	ADDW @RW3+, A	ADDW addr16, A	SUBW @RW3+, A	SUBW addr16, A	SUBCW A, @RW3+	SUBCW A, addr16	NEGW @RW3+	NEGW addr16	ANDW @RW3+, A	ANDW addr16, A	ORW @RW3+, A	ORW addr16, A	XORW @RW3+, A	XORW addr16, A	NOTW @RW3+	NOTW addr16

Table B.3.1m “ea” Instructions 9 (First byte = 78H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MULU A, R0	MULU A, @RW0+d8	MULUW A, RW0	MULUW A, @RW0+d8	MUL A, R0	MUL A, @RW0+d8	MULW A, RW0	MULW A, @RW0+d8	DIVU A, R0	DIVU A, @RW0+d8	DIVUW A, RW0	DIVUW A, @RW0+d8	DIV A, R0	DIV A, @RW0+d8	DIVW A, RW0	DIVW A, @RW0+d8
+1	MULU A, R1	MULU A, @RW1+d8	MULUW A, RW1	MULUW A, @RW1+d8	MUL A, R1	MUL A, @RW1+d8	MULW A, RW1	MULW A, @RW1+d8	DIVU A, R1	DIVU A, @RW1+d8	DIVUW A, RW1	DIVUW A, @RW1+d8	DIV A, R1	DIV A, @RW1+d8	DIVW A, RW1	DIVW A, @RW1+d8
+2	MULU A, R2	MULU A, @RW2+d8	MULUW A, RW2	MULUW A, @RW2+d8	MUL A, R2	MUL A, @RW2+d8	MULW A, RW2	MULW A, @RW2+d8	DIVU A, R2	DIVU A, @RW2+d8	DIVUW A, RW2	DIVUW A, @RW2+d8	DIV A, R2	DIV A, @RW2+d8	DIVW A, RW2	DIVW A, @RW2+d8
+3	MULU A, R3	MULU A, @RW3+d8	MULUW A, RW3	MULUW A, @RW3+d8	MUL A, R3	MUL A, @RW3+d8	MULW A, RW3	MULW A, @RW3+d8	DIVU A, R3	DIVU A, @RW3+d8	DIVUW A, RW3	DIVUW A, @RW3+d8	DIV A, R3	DIV A, @RW3+d8	DIVW A, RW3	DIVW A, @RW3+d8
+4	MULU A, R4	MULU A, @RW4+d8	MULUW A, RW4	MULUW A, @RW4+d8	MUL A, R4	MUL A, @RW4+d8	MULW A, RW4	MULW A, @RW4+d8	DIVU A, R4	DIVU A, @RW4+d8	DIVUW A, RW4	DIVUW A, @RW4+d8	DIV A, R4	DIV A, @RW4+d8	DIVW A, RW4	DIVW A, @RW4+d8
+5	MULU A, R5	MULU A, @RW5+d8	MULUW A, RW5	MULUW A, @RW5+d8	MUL A, R5	MUL A, @RW5+d8	MULW A, RW5	MULW A, @RW5+d8	DIVU A, R5	DIVU A, @RW5+d8	DIVUW A, RW5	DIVUW A, @RW5+d8	DIV A, R5	DIV A, @RW5+d8	DIVW A, RW5	DIVW A, @RW5+d8
+6	MULU A, R6	MULU A, @RW6+d8	MULUW A, RW6	MULUW A, @RW6+d8	MUL A, R6	MUL A, @RW6+d8	MULW A, RW6	MULW A, @RW6+d8	DIVU A, R6	DIVU A, @RW6+d8	DIVUW A, RW6	DIVUW A, @RW6+d8	DIV A, R6	DIV A, @RW6+d8	DIVW A, RW6	DIVW A, @RW6+d8
+7	MULU A, R7	MULU A, @RW7+d8	MULUW A, RW7	MULUW A, @RW7+d8	MUL A, R7	MUL A, @RW7+d8	MULW A, RW7	MULW A, @RW7+d8	DIVU A, R7	DIVU A, @RW7+d8	DIVUW A, RW7	DIVUW A, @RW7+d8	DIV A, R7	DIV A, @RW7+d8	DIVW A, RW7	DIVW A, @RW7+d8
+8	MULU A, @RW0	MULU A, @RW0+d16	MULUW A, @RW0	MULUW A, @RW0+d16	MUL A, @RW0	MUL A, @RW0+d16	MULW A, @RW0	MULW A, @RW0+d16	DIVU A, @RW0	DIVU A, @RW0+d16	DIVUW A, @RW0	DIVUW A, @RW0+d16	DIV A, @RW0	DIV A, @RW0+d16	DIVW A, @RW0	DIVW A, @RW0+d16
+9	MULU A, @RW1	MULU A, @RW1+d16	MULUW A, @RW1	MULUW A, @RW1+d16	MUL A, @RW1	MUL A, @RW1+d16	MULW A, @RW1	MULW A, @RW1+d16	DIVU A, @RW1	DIVU A, @RW1+d16	DIVUW A, @RW1	DIVUW A, @RW1+d16	DIV A, @RW1	DIV A, @RW1+d16	DIVW A, @RW1	DIVW A, @RW1+d16
+A	MULU A, @RW2	MULU A, @RW2+d16	MULUW A, @RW2	MULUW A, @RW2+d16	MUL A, @RW2	MUL A, @RW2+d16	MULW A, @RW2	MULW A, @RW2+d16	DIVU A, @RW2	DIVU A, @RW2+d16	DIVUW A, @RW2	DIVUW A, @RW2+d16	DIV A, @RW2	DIV A, @RW2+d16	DIVW A, @RW2	DIVW A, @RW2+d16
+B	MULU A, @RW3	MULU A, @RW3+d16	MULUW A, @RW3	MULUW A, @RW3+d16	MUL A, @RW3	MUL A, @RW3+d16	MULW A, @RW3	MULW A, @RW3+d16	DIVU A, @RW3	DIVU A, @RW3+d16	DIVUW A, @RW3	DIVUW A, @RW3+d16	DIV A, @RW3	DIV A, @RW3+d16	DIVW A, @RW3	DIVW A, @RW3+d16
+C	MULU A, @RW0+	MULU A, @RW0+RW7	MULUW A, @RW0+	MULUW A, @RW0+RW7	MUL A, @RW0+	MUL A, @RW0+RW7	MULW A, @RW0+	MULW A, @RW0+RW7	DIVU A, @RW0+	DIVU A, @RW0+RW7	DIVUW A, @RW0+	DIVUW A, @RW0+RW7	DIV A, @RW0+	DIV A, @RW0+RW7	DIVW A, @RW0+	DIVW A, @RW0+RW7
+D	MULU A, @RW1+	MULU A, @RW1+RW7	MULUW A, @RW1+	MULUW A, @RW1+RW7	MUL A, @RW1+	MUL A, @RW1+RW7	MULW A, @RW1+	MULW A, @RW1+RW7	DIVU A, @RW1+	DIVU A, @RW1+RW7	DIVUW A, @RW1+	DIVUW A, @RW1+RW7	DIV A, @RW1+	DIV A, @RW1+RW7	DIVW A, @RW1+	DIVW A, @RW1+RW7
+E	MULU A, @RW2+	MULU A, @PC+d16	MULUW A, @RW2+	MULUW A, @PC+d16	MUL A, @RW2+	MUL A, @PC+d16	MULW A, @RW2+	MULW A, @PC+d16	DIVU A, @RW2+	DIVU A, @PC+d16	DIVUW A, @RW2+	DIVUW A, @ PC+d16	DIV A, @RW2+	DIV A, @PC+d16	DIVW A, @RW2+	DIVW A, @ PC+d16
+F	MULU A, @RW3+	MULU A, addr16	MULUW A, @RW3+	MULUW A, addr16	MUL A, @RW3+	MUL A, addr16	MULW A, @RW3+	MULW A, addr16	DIVU A, @RW3+	DIVU A, addr16	DIVUW A, @RW3+	DIVUW A, addr16	DIV A, @RW3+	DIV A, addr16	DIVW A, @RW3+	DIVW A, addr16

Table B.3.1n MOVEA RWi, ea (First byte = 79H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVEA RW0, RW0	MOVEA RW0, @RW0+d8	MOVEA RW1, RW0	MOVEA RW1, @RW0+d8	MOVEA RW2, RW0	MOVEA RW2, @RW0+d8	MOVEA RW3, RW0	MOVEA RW3, @RW0+d8	MOVEA RW4, RW0	MOVEA RW4, @RW0+d8	MOVEA RW5, RW0	MOVEA RW5, @RW0+d8	MOVEA RW6, RW0	MOVEA RW6 ,@RW0+d8	MOVEA RW7, RW0	MOVEA RW7, @RW0+d8
+1	MOVEA RW0, RW1	MOVEA RW0, @RW1+d8	MOVEA RW1, RW1	MOVEA RW1, @RW1+d8	MOVEA RW2, RW1	MOVEA RW2, @RW1+d8	MOVEA RW3, RW1	MOVEA RW3, @RW1+d8	MOVEA RW4, RW1	MOVEA RW4, @RW1+d8	MOVEA RW5, RW1	MOVEA RW5, @RW1+d8	MOVEA RW6, RW1	MOVEA RW6 ,@RW1+d8	MOVEA RW7, RW1	MOVEA RW7, @RW1+d8
+2	MOVEA RW0, RW2	MOVEA RW0, @RW2+d8	MOVEA RW1, RW2	MOVEA RW1, @RW2+d8	MOVEA RW2, RW2	MOVEA RW2, @RW2+d8	MOVEA RW3, RW2	MOVEA RW3, @RW2+d8	MOVEA RW4, RW2	MOVEA RW4, @RW2+d8	MOVEA RW5, RW2	MOVEA RW5, @RW2+d8	MOVEA RW6, RW2	MOVEA RW6 ,@RW2+d8	MOVEA RW7, RW2	MOVEA RW7, @RW2+d8
+3	MOVEA RW0, RW3	MOVEA RW0, @RW3+d8	MOVEA RW1, RW3	MOVEA RW1, @RW3+d8	MOVEA RW2, RW3	MOVEA RW2, @RW3+d8	MOVEA RW3, RW3	MOVEA RW3, @RW3+d8	MOVEA RW4, RW3	MOVEA RW4, @RW3+d8	MOVEA RW5, RW3	MOVEA RW5, @RW3+d8	MOVEA RW6, RW3	MOVEA RW6 ,@RW3+d8	MOVEA RW7, RW3	MOVEA RW7, @RW3+d8
+4	MOVEA RW0, RW4	MOVEA RW0, @RW4+d8	MOVEA RW1, RW4	MOVEA RW1, @RW4+d8	MOVEA RW2, RW4	MOVEA RW2, @RW4+d8	MOVEA RW3, RW4	MOVEA RW3, @RW4+d8	MOVEA RW4, RW4	MOVEA RW4, @RW4+d8	MOVEA RW5, RW4	MOVEA RW5, @RW4+d8	MOVEA RW6, RW4	MOVEA RW6 ,@RW4+d8	MOVEA RW7, RW4	MOVEA RW7, @RW4+d8
+5	MOVEA RW0, RW5	MOVEA RW0, @RW5+d8	MOVEA RW1, RW5	MOVEA RW1, @RW5+d8	MOVEA RW2, RW5	MOVEA RW2, @RW5+d8	MOVEA RW3, RW5	MOVEA RW3, @RW5+d8	MOVEA RW4, RW5	MOVEA RW4, @RW5+d8	MOVEA RW5, RW5	MOVEA RW5, @RW5+d8	MOVEA RW6, RW5	MOVEA RW6 ,@RW5+d8	MOVEA RW7, RW5	MOVEA RW7, @RW5+d8
+6	MOVEA RW0, RW6	MOVEA RW0, @RW6+d8	MOVEA RW1, RW6	MOVEA RW1, @RW6+d8	MOVEA RW2, RW6	MOVEA RW2, @RW6+d8	MOVEA RW3, RW6	MOVEA RW3, @RW6+d8	MOVEA RW4, RW6	MOVEA RW4, @RW6+d8	MOVEA RW5, RW6	MOVEA RW5, @RW6+d8	MOVEA RW6, RW6	MOVEA RW6 ,@RW6+d8	MOVEA RW7, RW6	MOVEA RW7, @RW6+d8
+7	MOVEA RW0, RW7	MOVEA RW0, @RW7+d8	MOVEA RW1, RW7	MOVEA RW1, @RW7+d8	MOVEA RW2, RW7	MOVEA RW2, @RW7+d8	MOVEA RW3, RW7	MOVEA RW3, @RW7+d8	MOVEA RW4, RW7	MOVEA RW4, @RW7+d8	MOVEA RW5, RW7	MOVEA RW5, @RW7+d8	MOVEA RW6, RW7	MOVEA RW6 ,@RW7+d8	MOVEA RW7, RW7	MOVEA RW7, @RW7+d8
+8	MOVEA RW0, @RW0	MOVEA RW0, @RW0+d16	MOVEA RW1, @RW0	MOVEA RW1, @RW0+d16	MOVEA RW2, @RW0	MOVEA RW2, @RW0+d16	MOVEA RW3, @RW0	MOVEA RW3, @RW0+d16	MOVEA RW4, @RW0	MOVEA RW4, @RW0+d16	MOVEA RW5, @RW0	MOVEA RW5, @RW0+d16	MOVEA RW6, @RW0	MOVEA RW6 ,@RW0+d16	MOVEA RW7, @RW0	MOVEA RW7, @RW0+d16
+9	MOVEA RW0, @RW1	MOVEA RW0, @RW1+d16	MOVEA RW1, @RW1	MOVEA RW1, @RW1+d16	MOVEA RW2, @RW1	MOVEA RW2, @RW1+d16	MOVEA RW3, @RW1	MOVEA RW3, @RW1+d16	MOVEA RW4, @RW1	MOVEA RW4, @RW1+d16	MOVEA RW5, @RW1	MOVEA RW5, @RW1+d16	MOVEA RW6, @RW1	MOVEA RW6 ,@RW1+d16	MOVEA RW7, @RW1	MOVEA RW7, @RW1+d16
+A	MOVEA RW0, @RW2	MOVEA RW0, @RW2+d16	MOVEA RW1, @RW2	MOVEA RW1, @RW2+d16	MOVEA RW2, @RW2	MOVEA RW2, @RW2+d16	MOVEA RW3, @RW2	MOVEA RW3, @RW2+d16	MOVEA RW4, @RW2	MOVEA RW4, @RW2+d16	MOVEA RW5, @RW2	MOVEA RW5, @RW2+d16	MOVEA RW6, @RW2	MOVEA RW6 ,@RW2+d16	MOVEA RW7, @RW2	MOVEA RW7, @RW2+d16
+B	MOVEA RW0, @RW3	MOVEA RW0, @RW3+d16	MOVEA RW1, @RW3	MOVEA RW1, @RW3+d16	MOVEA RW2, @RW3	MOVEA RW2, @RW3+d16	MOVEA RW3, @RW3	MOVEA RW3, @RW3+d16	MOVEA RW4, @RW3	MOVEA RW4, @RW3+d16	MOVEA RW5, @RW3	MOVEA RW5, @RW3+d16	MOVEA RW6, @RW3	MOVEA RW6 ,@RW3+d16	MOVEA RW7, @RW3	MOVEA RW7, @RW3+d16
+C	MOVEA RW0, @RW0+	MOVEA RW0, @RW0+RW7	MOVEA RW1, @RW0+	MOVEA RW1, @RW0+RW7	MOVEA RW2, @RW0+	MOVEA RW2, @RW0+RW7	MOVEA RW3, @RW0+	MOVEA RW3, @RW0+RW7	MOVEA RW4, @RW0+	MOVEA RW4, @RW0+RW7	MOVEA RW5, @RW0+	MOVEA RW5, @RW0+RW7	MOVEA RW6, @RW0+	MOVEA RW6 ,@RW0+RW7	MOVEA RW7, @RW0+	MOVEA RW7, @RW0+RW7
+D	MOVEA RW0, @RW1+	MOVEA RW0, @RW1+RW7	MOVEA RW1, @RW1+	MOVEA RW1, @RW1+RW7	MOVEA RW2, @RW1+	MOVEA RW2, @RW1+RW7	MOVEA RW3, @RW1+	MOVEA RW3, @RW1+RW7	MOVEA RW4, @RW1+	MOVEA RW4, @RW1+RW7	MOVEA RW5, @RW1+	MOVEA RW5, @RW1+RW7	MOVEA RW6, @RW1+	MOVEA RW6 ,@RW1+RW7	MOVEA RW7, @RW1+	MOVEA RW7, @RW1+RW7
+E	MOVEA RW0, @RW2+	MOVEA RW0, @PC+d16	MOVEA RW1, @RW2+	MOVEA RW1, @PC+d16	MOVEA RW2, @RW2+	MOVEA RW2, @PC+d16	MOVEA RW3, @RW2+	MOVEA RW3, @PC+d16	MOVEA RW4, @RW2+	MOVEA RW4, @PC+d16	MOVEA RW5, @RW2+	MOVEA RW5, @PC+d16	MOVEA RW6, @RW2+	MOVEA RW6 ,@PC+d16	MOVEA RW7, @RW2+	MOVEA RW7, @PC+d16
+F	MOVEA RW0, @RW3+	MOVEA RW0, addr16	MOVEA RW1, @RW3+	MOVEA RW1, addr16	MOVEA RW2, @RW3+	MOVEA RW2, addr16	MOVEA RW3, @RW3+	MOVEA RW3, addr16	MOVEA RW4, @RW3+	MOVEA RW4, addr16	MOVEA RW5, @RW3+	MOVEA RW5, addr16	MOVEA RW6, @RW3+	MOVEA RW6 ,addr16	MOVEA RW7, @RW3+	MOVEA RW7, addr16

Table B.3.1o MOV Ri, ea (First byte = 7AH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0	MOV R0, @RW0+d8	MOV R1, R0	MOV R1, @RW0+d8	MOV R2, R0	MOV R2, @RW0+d8	MOV R3, R0	MOV R3, @RW0+d8	MOV R4, R0	MOV R4, @RW0+d8	MOV R5, R0	MOV R5, @RW0+d8	MOV R6, R0	MOV R6, @RW0+d8	MOV R7, R0	MOV R7, @RW0+d8
+1	MOV R0, R1	MOV R0, @RW1+d8	MOV R1, R1	MOV R1, @RW1+d8	MOV R2, R1	MOV R2, @RW1+d8	MOV R3, R1	MOV R3, @RW1+d8	MOV R4, R1	MOV R4, @RW1+d8	MOV R5, R1	MOV R5, @RW1+d8	MOV R6, R1	MOV R6, @RW1+d8	MOV R7, R1	MOV R7, @RW1+d8
+2	MOV R0, R2	MOV R0, @RW2+d8	MOV R1, R2	MOV R1, @RW2+d8	MOV R2, R2	MOV R2, @RW2+d8	MOV R3, R2	MOV R3, @RW2+d8	MOV R4, R2	MOV R4, @RW2+d8	MOV R5, R2	MOV R5, @RW2+d8	MOV R6, R2	MOV R6, @RW2+d8	MOV R7, R2	MOV R7, @RW2+d8
+3	MOV R0, R3	MOV R0, @RW3+d8	MOV R1, R3	MOV R1, @RW3+d8	MOV R2, R3	MOV R2, @RW3+d8	MOV R3, R3	MOV R3, @RW3+d8	MOV R4, R3	MOV R4, @RW3+d8	MOV R5, R3	MOV R5, @RW3+d8	MOV R6, R3	MOV R6, @RW3+d8	MOV R7, R3	MOV R7, @RW3+d8
+4	MOV R0, R4	MOV R0, @RW4+d8	MOV R1, R4	MOV R1, @RW4+d8	MOV R2, R4	MOV R2, @RW4+d8	MOV R3, R4	MOV R3, @RW4+d8	MOV R4, R4	MOV R4, @RW4+d8	MOV R5, R4	MOV R5, @RW4+d8	MOV R6, R4	MOV R6, @RW4+d8	MOV R7, R4	MOV R7, @RW4+d8
+5	MOV R0, R5	MOV R0, @RW5+d8	MOV R1, R5	MOV R1, @RW5+d8	MOV R2, R5	MOV R2, @RW5+d8	MOV R3, R5	MOV R3, @RW5+d8	MOV R4, R5	MOV R4, @RW5+d8	MOV R5, R5	MOV R5, @RW5+d8	MOV R6, R5	MOV R6, @RW5+d8	MOV R7, R5	MOV R7, @RW5+d8
+6	MOV R0, R6	MOV R0, @RW6+d8	MOV R1, R6	MOV R1, @RW6+d8	MOV R2, R6	MOV R2, @RW6+d8	MOV R3, R6	MOV R3, @RW6+d8	MOV R4, R6	MOV R4, @RW6+d8	MOV R5, R6	MOV R5, @RW6+d8	MOV R6, R6	MOV R6, @RW6+d8	MOV R7, R6	MOV R7, @RW6+d8
+7	MOV R0, R7	MOV R0, @RW7+d8	MOV R1, R7	MOV R1, @RW7+d8	MOV R2, R7	MOV R2, @RW7+d8	MOV R3, R7	MOV R3, @RW7+d8	MOV R4, R7	MOV R4, @RW7+d8	MOV R5, R7	MOV R5, @RW7+d8	MOV R6, R7	MOV R6, @RW7+d8	MOV R7, R7	MOV R7, @RW7+d8
+8	MOV R0, @RW0	MOV R0, @RW0+d16	MOV R1, @RW0	MOV R1, @RW0+d16	MOV R2, @RW0	MOV R2, @RW0+d16	MOV R3, @RW0	MOV R3, @RW0+d16	MOV R4, @RW0	MOV R4, @RW0+d16	MOV R5, @RW0	MOV R5, @RW0+d16	MOV R6, @RW0	MOV R6, @RW0+d16	MOV R7, @RW0	MOV R7, @RW0+d16
+9	MOV R0, @RW1	MOV R0, @RW1+d16	MOV R1, @RW1	MOV R1, @RW1+d16	MOV R2, @RW1	MOV R2, @RW1+d16	MOV R3, @RW1	MOV R3, @RW1+d16	MOV R4, @RW1	MOV R4, @RW1+d16	MOV R5, @RW1	MOV R5, @RW1+d16	MOV R6, @RW1	MOV R6, @RW1+d16	MOV R7, @RW1	MOV R7, @RW1+d16
+A	MOV R0, @RW2	MOV R0, @RW2+d16	MOV R1, @RW2	MOV R1, @RW2+d16	MOV R2, @RW2	MOV R2, @RW2+d16	MOV R3, @RW2	MOV R3, @RW2+d16	MOV R4, @RW2	MOV R4, @RW2+d16	MOV R5, @RW2	MOV R5, @RW2+d16	MOV R6, @RW2	MOV R6, @RW2+d16	MOV R7, @RW2	MOV R7, @RW2+d16
+B	MOV R0, @RW3	MOV R0, @RW3+d16	MOV R1, @RW3	MOV R1, @RW3+d16	MOV R2, @RW3	MOV R2, @RW3+d16	MOV R3, @RW3	MOV R3, @RW3+d16	MOV R4, @RW3	MOV R4, @RW3+d16	MOV R5, @RW3	MOV R5, @RW3+d16	MOV R6, @RW3	MOV R6, @RW3+d16	MOV R7, @RW3	MOV R7, @RW3+d16
+C	MOV R0, @RW0+	MOV R0, @RW0+RW7	MOV R1, @RW0+	MOV R1, @RW0+RW7	MOV R2, @RW0+	MOV R2, @RW0+RW7	MOV R3, @RW0+	MOV R3, @RW0+RW7	MOV R4, @RW0+	MOV R4, @RW0+RW7	MOV R5, @RW0+	MOV R5, @RW0+RW7	MOV R6, @RW0+	MOV R6, @RW0+RW7	MOV R7, @RW0+	MOV R7, @RW0+RW7
+D	MOV R0, @RW1+	MOV R0, @RW1+RW7	MOV R1, @RW1+	MOV R1, @RW1+RW7	MOV R2, @RW1+	MOV R2, @RW1+RW7	MOV R3, @RW1+	MOV R3, @RW1+RW7	MOV R4, @RW1+	MOV R4, @RW1+RW7	MOV R5, @RW1+	MOV R5, @RW1+RW7	MOV R6, @RW1+	MOV R6, @RW1+RW7	MOV R7, @RW1+	MOV R7, @RW1+RW7
+E	MOV R0, @RW2+	MOV R0, @PC+d16	MOV R1, @RW2+	MOV R1, @PC+d16	MOV R2, @RW2+	MOV R2, @PC+d16	MOV R3, @RW2+	MOV R3, @PC+d16	MOV R4, @RW2+	MOV R4, @PC+d16	MOV R5, @RW2+	MOV R5, @PC+d16	MOV R6, @RW2+	MOV R6, @PC+d16	MOV R7, @RW2+	MOV R7, @PC+d16
+F	MOV R0, @RW3+	MOV R0, @addr16	MOV R1, @RW3+	MOV R1, addr16	MOV R2, @RW3+	MOV R2, addr16	MOV R3, @RW3+	MOV R3, addr16	MOV R4, @RW3+	MOV R4, addr16	MOV R5, @RW3+	MOV R5, addr16	MOV R6, @RW3+	MOV R6, addr16	MOV R7, @RW3+	MOV R7, addr16

Table B.3.1p MOVW RWi, ea (First byte = 7BH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVW RW0, RW0	MOVW RW0, @RW0+d8	MOVW RW1, RW0	MOVW RW1, @RW0+d8	MOVW RW2, RW0	MOVW RW2, @RW0+d8	MOVW RW3, RW0	MOVW RW3, @RW0+d8	MOVW RW4, RW0	MOVW RW4, @RW0+d8	MOVW RW5, RW0	MOVW RW5, @RW0+d8	MOVW RW6, RW0	MOVW RW6, @RW0+d8	MOVW RW7, RW0	MOVW RW7, @RW0+d8
+1	MOVW RW0, RW1	MOVW RW0, @RW1+d8	MOVW RW1, RW1	MOVW RW1, @RW1+d8	MOVW RW2, RW1	MOVW RW2, @RW1+d8	MOVW RW3, RW1	MOVW RW3, @RW1+d8	MOVW RW4, RW1	MOVW RW4, @RW1+d8	MOVW RW5, RW1	MOVW RW5, @RW1+d8	MOVW RW6, RW1	MOVW RW6, @RW1+d8	MOVW RW7, RW1	MOVW RW7, @RW1+d8
+2	MOVW RW0, RW2	MOVW RW0, @RW2+d8	MOVW RW1, RW2	MOVW RW1, @RW2+d8	MOVW RW2, RW2	MOVW RW2, @RW2+d8	MOVW RW3, RW2	MOVW RW3, @RW2+d8	MOVW RW4, RW2	MOVW RW4, @RW2+d8	MOVW RW5, RW2	MOVW RW5, @RW2+d8	MOVW RW6, RW2	MOVW RW6, @RW2+d8	MOVW RW7, RW2	MOVW RW7, @RW2+d8
+3	MOVW RW0, RW3	MOVW RW0, @RW3+d8	MOVW RW1, RW3	MOVW RW1, @RW3+d8	MOVW RW2, RW3	MOVW RW2, @RW3+d8	MOVW RW3, RW3	MOVW RW3, @RW3+d8	MOVW RW4, RW3	MOVW RW4, @RW3+d8	MOVW RW5, RW3	MOVW RW5, @RW3+d8	MOVW RW6, RW3	MOVW RW6, @RW3+d8	MOVW RW7, RW3	MOVW RW7, @RW3+d8
+4	MOVW RW0, RW4	MOVW RW0, @RW4+d8	MOVW RW1, RW4	MOVW RW1, @RW4+d8	MOVW RW2, RW4	MOVW RW2, @RW4+d8	MOVW RW3, RW4	MOVW RW3, @RW4+d8	MOVW RW4, RW4	MOVW RW4, @RW4+d8	MOVW RW5, RW4	MOVW RW5, @RW4+d8	MOVW RW6, RW4	MOVW RW6, @RW4+d8	MOVW RW7, RW4	MOVW RW7, @RW4+d8
+5	MOVW RW0, RW5	MOVW RW0, @RW5+d8	MOVW RW1, RW5	MOVW RW1, @RW5+d8	MOVW RW2, RW5	MOVW RW2, @RW5+d8	MOVW RW3, RW5	MOVW RW3, @RW5+d8	MOVW RW4, RW5	MOVW RW4, @RW5+d8	MOVW RW5, RW5	MOVW RW5, @RW5+d8	MOVW RW6, RW5	MOVW RW6, @RW5+d8	MOVW RW7, RW5	MOVW RW7, @RW5+d8
+6	MOVW RW0, RW6	MOVW RW0, @RW6+d8	MOVW RW1, RW6	MOVW RW1, @RW6+d8	MOVW RW2, RW6	MOVW RW2, @RW6+d8	MOVW RW3, RW6	MOVW RW3, @RW6+d8	MOVW RW4, RW6	MOVW RW4, @RW6+d8	MOVW RW5, RW6	MOVW RW5, @RW6+d8	MOVW RW6, RW6	MOVW RW6, @RW6+d8	MOVW RW7, RW6	MOVW RW7, @RW6+d8
+7	MOVW RW0, RW7	MOVW RW0, @RW7+d8	MOVW RW1, RW7	MOVW RW1, @RW7+d8	MOVW RW2, RW7	MOVW RW2, @RW7+d8	MOVW RW3, RW7	MOVW RW3, @RW7+d8	MOVW RW4, RW7	MOVW RW4, @RW7+d8	MOVW RW5, RW7	MOVW RW5, @RW7+d8	MOVW RW6, RW7	MOVW RW6, @RW7+d8	MOVW RW7, RW7	MOVW RW7, @RW7+d8
+8	MOVW RW0, @RW0	MOVW RW0, @RW0+d16	MOVW RW1, @RW0	MOVW RW1, @RW0+d16	MOVW RW2, @RW0	MOVW RW2, @RW0+d16	MOVW RW3, @RW0	MOVW RW3, @RW0+d16	MOVW RW4, @RW0	MOVW RW4, @RW0+d16	MOVW RW5, @RW0	MOVW RW5, @RW0+d16	MOVW RW6, @RW0	MOVW RW6, @RW0+d16	MOVW RW7, @RW0	MOVW RW7, @RW0+d16
+9	MOVW RW0, @RW1	MOVW RW0, @RW1+d16	MOVW RW1, @RW1	MOVW RW1, @RW1+d16	MOVW RW2, @RW1	MOVW RW2, @RW1+d16	MOVW RW3, @RW1	MOVW RW3, @RW1+d16	MOVW RW4, @RW1	MOVW RW4, @RW1+d16	MOVW RW5, @RW1	MOVW RW5, @RW1+d16	MOVW RW6, @RW1	MOVW RW6, @RW1+d16	MOVW RW7, @RW1	MOVW RW7, @RW1+d16
+A	MOVW RW0, @RW2	MOVW RW0, @RW2+d16	MOVW RW1, @RW2	MOVW RW1, @RW2+d16	MOVW RW2, @RW2	MOVW RW2, @RW2+d16	MOVW RW3, @RW2	MOVW RW3, @RW2+d16	MOVW RW4, @RW2	MOVW RW4, @RW2+d16	MOVW RW5, @RW2	MOVW RW5, @RW2+d16	MOVW RW6, @RW2	MOVW RW6, @RW2+d16	MOVW RW7, @RW2	MOVW RW7, @RW2+d16
+B	MOVW RW0, @RW3	MOVW RW0, @RW3+d16	MOVW RW1, @RW3	MOVW RW1, @RW3+d16	MOVW RW2, @RW3	MOVW RW2, @RW3+d16	MOVW RW3, @RW3	MOVW RW3, @RW3+d16	MOVW RW4, @RW3	MOVW RW4, @RW3+d16	MOVW RW5, @RW3	MOVW RW5, @RW3+d16	MOVW RW6, @RW3	MOVW RW6, @RW3+d16	MOVW RW7, @RW3	MOVW RW7, @RW3+d16
+C	MOVW R W0, @RW0+	MOVW RW0, @RW0+RW7	MOVW R W1, @RW0+	MOVW RW1, @RW0+RW7	MOVW R W2, @RW0+	MOVW RW2, @RW0+RW7	MOVW R W3, @RW0+	MOVW RW3, @RW0+RW7	MOVW R W4, @RW0+	MOVW RW4, @RW0+RW7	MOVW R W5, @RW0+	MOVW RW5, @RW0+RW7	MOVW R W6, @RW0+	MOVW RW6, @RW0+RW7	MOVW R W7, @RW0+	MOVW RW7, @RW0+RW7
+D	MOVW R W0, @RW1+	MOVW RW0, @RW1+RW7	MOVW R W1, @RW1+	MOVW RW1, @RW1+RW7	MOVW R W2, @RW1+	MOVW RW2, @RW1+RW7	MOVW R W3, @RW1+	MOVW RW3, @RW1+RW7	MOVW R W4, @RW1+	MOVW RW4, @RW1+RW7	MOVW R W5, @RW1+	MOVW RW5, @RW1+RW7	MOVW R W6, @RW1+	MOVW RW6, @RW1+RW7	MOVW R W7, @RW1+	MOVW RW7, @RW1+RW7
+E	MOVW R W0, @RW2+	MOVW RW0, @PC+d16	MOVW R W1, @RW2+	MOVW RW1, @PC+d16	MOVW R W2, @RW2+	MOVW RW2, @PC+d16	MOVW R W3, @RW2+	MOVW RW3, @PC+d16	MOVW R W4, @RW2+	MOVW RW4, @PC+d16	MOVW R W5, @RW2+	MOVW RW5, @PC+d16	MOVW R W6, @RW2+	MOVW RW6, @PC+d16	MOVW R W7, @RW2+	MOVW RW7, @PC+d16
+F	MOVW R W0, @RW3+	MOVW RW0, addr16	MOVW R W1, @RW3+	MOVW RW1, addr16	MOVW R W2, @RW3+	MOVW RW2, addr16	MOVW R W3, @RW3+	MOVW RW3, addr16	MOVW R W4, @RW3+	MOVW RW4, addr16	MOVW R W5, @RW3+	MOVW RW5, addr16	MOVW R W6, @RW3+	MOVW RW6, addr16	MOVW R W7, @RW3+	MOVW RW7, addr16

Table B.3.1q MOV ea, Ri (First byte = 7CH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0	MOV @R W0+d8, R0	MOV R0, R1	MOV @R W0+d8, R1	MOV R0, R2	MOV @R, W0+d8, R2	MOV R0, R3	MOV @R W0+d8, R3	MOV R0, R4	MOV @R W0+d8, R4	MOV R0, R5	MOV @R W0+d8, R5	MOV R0, R6	MOV @R W0+d8, R6	MOV R0, R7	MOV @R W0+d8, R7
+1	MOV R1, R0	MOV @R W1+d8, R0	MOV R1, R1	MOV @R W1+d8, R1	MOV R1, R2	MOV @R W1+d8, R2	MOV R1, R3	MOV @R W1+d8, R3	MOV R1, R4	MOV @R W1+d8, R4	MOV R1, R5	MOV @R W1+d8, R5	MOV R1, R6	MOV @R W1+d8, R6	MOV R1, R7	MOV @R W1+d8, R7
+2	MOV R2, R0	MOV @R W2+d8, R0	MOV R2, R1	MOV @R W2+d8, R1	MOV R2, R2	MOV @R W2+d8, R2	MOV R2, R3	MOV @R W2+d8, R3	MOV R2, R4	MOV @R, W2+d8, R4	MOV R2, R5	MOV @R W2+d8, R5	MOV R2, R6	MOV @R W2+d8, R6	MOV R2, R7	MOV @R W2+d8, R7
+3	MOV R3, R0	MOV @R W3+d8, R0	MOV R3, R1	MOV @R W3+d8, R1	MOV R3, R2	MOV @R W3+d8, R2	MOV R3, R3	MOV @R W3+d8, R3	MOV R3, R4	MOV @R, W3+d8, R4	MOV R3, R5	MOV @R W3+d8, R5	MOV R3, R6	MOV @R W3+d8, R6	MOV R3, R7	MOV @R W3+d8, R7
+4	MOV R4, R0	MOV @R W4+d8, R0	MOV R4, R1	MOV @R W4+d8, R1	MOV R4, R2	MOV @R W4+d8, R2	MOV R4, R3	MOV @R W4+d8, R3	MOV R4, R4	MOV @R W4+d8, R4	MOV R4, R5	MOV @R W4+d8, R5	MOV R4, R6	MOV @R W4+d8, R6	MOV R4, R7	MOV @R W4+d8, R7
+5	MOV R5, R0	MOV @R W5+d8, R0	MOV R5, R1	MOV @R W5+d8, R1	MOV R5, R2	MOV @R W5+d8, R2	MOV R5, R3	MOV @R W5+d8, R3	MOV R5, R4	MOV @R W5+d8, R4	MOV R5, R5	MOV @R W5+d8, R5	MOV R5, R6	MOV @R W5+d8, R6	MOV R5, R7	MOV @R W5+d8, R7
+6	MOV R6, R0	MOV @R W6+d8, R0	MOV R6, R1	MOV @R W6+d8, R1	MOV R6, R2	MOV @R W6+d8, R2	MOV R6, R3	MOV @R W6+d8, R3	MOV R6, R4	MOV @R W6+d8, R4	MOV R6, R5	MOV @R W6+d8, R5	MOV R6, R6	MOV @R W6+d8, R6	MOV R6, R7	MOV @R W6+d8, R7
+7	MOV R7, R0	MOV @R W7+d8, R0	MOV R7, R1	MOV @R W7+d8, R1	MOV R7, R2	MOV @R W7+d8, R2	MOV R7, R3	MOV @R W7+d8, R3	MOV R7, R4	MOV @R W7+d8, R4	MOV R7, R5	MOV @R W7+d8, R5	MOV R7, R6	MOV @R W7+d8, R6	MOV R7, R7	MOV @R W7+d8, R7
+8	MOV @RW0, R0	MOV @RW W0+d16, R0	MOV @RW0, R1	MOV @RW 0+d16, R1	MOV @RW0, R2	MOV @RW 0+d16, R2	MOV @RW0, R3	MOV @RW 0+d16, R3	MOV @RW0, R4	MOV @RW 0+d16, R4	MOV @RW0, R5	MOV @RW 0+d16, R5	MOV @RW0, R6	MOV @RW 0+d16, R6	MOV @RW0, R7	MOV @RW 0+d16, R7
+9	MOV @RW1, R0	MOV @RW W1+d16, R0	MOV @RW1, R1	MOV @RW 1+d16, R1	MOV @RW1, R2	MOV @RW 1+d16, R2	MOV @RW1, R3	MOV @RW 1+d16, R3	MOV @RW1, R4	MOV @RW 1+d16, R4	MOV @RW1, R5	MOV @RW 1+d16, R5	MOV @RW1, R6	MOV @RW 1+d16, R6	MOV @RW1, R7	MOV @RW 1+d16, R7
+A	MOV @RW2, R0	MOV @RW W2+d16, R0	MOV @RW2, R1	MOV @RW 2+d16, R1	MOV @RW2, R2	MOV @RW 2+d16, R2	MOV @RW2, R3	MOV @RW 2+d16, R3	MOV @RW2, R4	MOV @RW 2+d16, R4	MOV @RW2, R5	MOV @RW 2+d16, R5	MOV @RW2, R6	MOV @RW 2+d16, R6	MOV @RW2, R7	MOV @RW 2+d16, R7
+B	MOV @RW3, R0	MOV @RW W3+d16, R0	MOV @RW3, R1	MOV @RW 3+d16, R1	MOV @RW3, R2	MOV @RW 3+d16, R2	MOV @RW3, R3	MOV @RW 3+d16, R3	MOV @RW3, R4	MOV @RW 3+d16, R4	MOV @RW3, R5	MOV @RW 3+d16, R5	MOV @RW3, R6	MOV @RW 3+d16, R6	MOV @RW3, R7	MOV @RW 3+d16, R7
+C	MOV @RW0+, R0	MOV @RW W0+RW7, R0	MOV @RW0+, R1	MOV @RW 0+RW7, R1	MOV @RW0+, R2	MOV @RW 0+RW7, R2	MOV @RW0+, R3	MOV @RW 0+RW7, R3	MOV @RW0+, R4	MOV @RW 0+RW7, R4	MOV @RW0+, R5	MOV @RW 0+RW7, R5	MOV @RW0+, R6	MOV @RW 0+RW7, R6	MOV @RW0+, R7	MOV @RW 0+RW7, R7
+D	MOV @RW1+, R0	MOV @RW W1+RW7, R0	MOV @RW1+, R1	MOV @RW 1+RW7, R1	MOV @RW1+, R2	MOV @RW 1+RW7, R2	MOV @RW1+, R3	MOV @RW 1+RW7, R3	MOV @RW1+, R4	MOV @RW 1+RW7, R4	MOV @RW1+, R5	MOV @RW 1+RW7, R5	MOV @RW1+, R6	MOV @RW 1+RW7,R6	MOV @RW1+, R7	MOV @RW 1+RW7, R7
+E	MOV @RW2+, R0	MOV PC+d16, R0	MOV @RW2+, R1	MOV PC+d16, R1	MOV @RW2+, R2	MOV PC+d16, R2	MOV @RW2+, R3	MOV PC+d16, R3	MOV @RW2+, R4	MOV PC+d16, R4	MOV @RW2+, R5	MOV PC+d16, R5	MOV @RW2+, R6	MOV PC+d16, R6	MOV @RW2+, R7	MOV PC+d16, R7
+F	MOV @RW3+, R0	MOV addr16, R0	MOV @RW3+, R1	MOV addr16, R1	MOV @RW3+, R2	MOV addr16, R2	MOV @RW3+, R3	MOV addr16, R3	MOV @RW3+, R4	MOV addr16, R4	MOV @RW3+, R5	MOV addr16, R5	MOV @RW3+, R6	MOV addr16, R6	MOV @RW3+, R7	MOV addr16, R7

Table B.3.1r MOVW ea, RWi (First byte = 7DH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVW RW0, RW0	MOVW @RW0+d8, RW0	MOVW RW0, RW1	MOVW @RW0+d8, RW1	MOVW RW0, RW2	MOVW @RW0+d8, RW2	MOVW RW0, RW3	MOVW @RW0+d8, RW3	MOVW RW0, RW4	MOVW @RW0+d8, RW4	MOVW RW0, RW5	MOVW @RW0+d8, RW5	MOVW RW0, RW6	MOVW @RW0+d8, RW6	MOVW RW0, RW7	MOVW @RW0+d8, RW7
+1	MOVW RW1, RW0	MOVW @RW1+d8, RW0	MOVW RW1, RW1	MOVW @R1+d8, RW1	MOVW RW1, RW2	MOVW @R1+d8, RW2	MOVW RW1, RW3	MOVW @RW1+d8, RW3	MOVW RW1, RW4	MOVW @RW1+d8, RW4	MOVW RW1, RW5	MOVW @RW1+d8, RW5	MOVW RW1, RW6	MOVW @RW1+d8, RW6	MOVW RW1, RW7	MOVW @RW1+d8, RW7
+2	MOVW RW2, RW0	MOVW @RW2+d8, RW0	MOVW RW2, RW1	MOVW @RW2+d8, RW1	MOVW RW2, RW2	MOVW @RW2+d8, RW2	MOVW RW2, RW3	MOVW @RW2+d8, RW3	MOVW RW2, RW4	MOVW @RW2+d8, RW4	MOVW RW2, RW5	MOVW @RW2+d8, RW5	MOVW RW2, RW6	MOVW @RW2+d8, RW6	MOVW RW2, RW7	MOVW @RW2+d8, RW7
+3	MOVW RW3, RW0	MOVW @RW3+d8, RW0	MOVW RW3, RW1	MOVW @RW3+d8, RW1	MOVW RW3, RW2	MOVW @RW3+d8, RW2	MOVW RW3, RW3	MOVW @RW3+d8, RW3	MOVW RW3, RW4	MOVW @RW3+d8, RW4	MOVW RW3, RW5	MOVW @RW3+d8, RW5	MOVW RW3, RW6	MOVW @RW3+d8, RW6	MOVW RW3, RW7	MOVW @RW3+d8, RW7
+4	MOVW RW4, RW0	MOVW @RW4+d8, RW0	MOVW RW4, RW1	MOVW @RW4+d8, RW1	MOVW RW4, RW2	MOVW @RW4+d8, RW2	MOVW RW4, RW3	MOVW @RW4+d8, RW3	MOVW RW4, RW4	MOVW @RW4+d8, RW4	MOVW RW4, RW5	MOVW @RW4+d8, RW5	MOVW RW4, RW6	MOVW @RW4+d8, RW6	MOVW RW4, RW7	MOVW @RW4+d8, RW7
+5	MOVW RW5, RW0	MOVW @RW5+d8, RW0	MOVW RW5, RW1	MOVW @RW5+d8, RW1	MOVW RW5, RW2	MOVW @RW5+d8, RW2	MOVW RW5, RW3	MOVW @RW5+d8, RW3	MOVW RW5, RW4	MOVW @RW5+d8, RW4	MOVW RW5, RW5	MOVW @RW5+d8, RW5	MOVW RW5, RW6	MOVW @RW5+d8, RW6	MOVW RW5, RW7	MOVW @RW5+d8, RW7
+6	MOVW RW6, RW0	MOVW @RW6+d8, RW0	MOVW RW6, RW1	MOVW @RW6+d8, RW1	MOVW RW6, RW2	MOVW @RW6+d8, RW2	MOVW RW6, RW3	MOVW @RW6+d8, RW3	MOVW RW6, RW4	MOVW @RW6+d8, RW4	MOVW RW6, RW5	MOVW @RW6+d8, RW5	MOVW RW6, RW6	MOVW @RW6+d8, RW6	MOVW RW6, RW7	MOVW @RW6+d8, RW7
+7	MOVW RW7, RW0	MOVW @RW7+d8, RW0	MOVW RW7, RW1	MOVW @RW7+d8, RW1	MOVW RW7, RW2	MOVW @RW7+d8, RW2	MOVW RW7, RW3	MOVW @RW7+d8, RW3	MOVW RW7, RW4	MOVW @RW7+d8, RW4	MOVW RW7, RW5	MOVW @RW7+d8, RW5	MOVW RW7, RW6	MOVW @RW7+d8, RW6	MOVW RW7, RW7	MOVW @RW7+d8, RW7
+8	MOVW @RW0, RW0	MOVW@RW0+d16, RW0	MOVW @RW0, RW1	MOVW@RW0+d16, RW1	MOVW @RW0, RW2	MOVW@RW0+d16, RW2	MOVW @RW0, RW3	MOVW@RW0+d16, RW3	MOVW @RW0, RW4	MOVW@RW0+d16, RW4	MOVW @RW0, RW5	MOVW@RW0+d16, RW5	MOVW @RW0, RW6	MOVW@RW0+d16, RW6	MOVW @RW0, RW7	MOVW@RW0+d16, RW7
+9	MOVW @RW1, RW0	MOVW@RW1+d16, RW0	MOVW @RW1, RW1	MOVW@RW1+d16, RW1	MOVW @RW1, RW2	MOVW@RW1+d16, RW2	MOVW @RW1, RW3	MOVW@RW1+d16, RW3	MOVW @RW1, RW4	MOVW@RW1+d16, RW4	MOVW @RW1, RW5	MOVW@RW1+d16, RW5	MOVW @RW1, RW6	MOVW@RW1+d16, RW6	MOVW @RW1, RW7	MOVW@RW1+d16, RW7
+A	MOVW @RW2, RW0	MOVW@RW2+d16, RW0	MOVW @RW2, RW1	MOVW@RW2+d16, RW1	MOVW @RW2, RW2	MOVW@RW2+d16, RW2	MOVW @RW2, RW3	MOVW@RW2+d16, RW3	MOVW @RW2, RW4	MOVW@RW2+d16, RW4	MOVW @RW2, RW5	MOVW@RW2+d16, RW5	MOVW @RW2, RW6	MOVW@RW2+d16, RW6	MOVW @RW2, RW7	MOVW@RW2+d16, RW7
+B	MOVW @RW3, RW0	MOVW@RW3+d16, RW0	MOVW @RW3, RW1	MOVW@RW3+d16, RW1	MOVW @RW3, RW2	MOVW@RW3+d16, RW2	MOVW @RW3, RW3	MOVW@RW3+d16, RW3	MOVW @RW3, RW4	MOVW@RW3+d16, RW4	MOVW @RW3, RW5	MOVW@RW3+d16, RW5	MOVW @RW3, RW6	MOVW@RW3+d16, RW6	MOVW @RW3, RW7	MOVW@RW3+d16, RW7
+C	MOVW @RW0+, RW0	MOVW@RW0+RW7, RW0	MOVW @RW0+, RW1	MOVW@RW0+RW7, RW1	MOVW @RW0+, RW2	MOVW@RW0+RW7, RW2	MOVW @RW0+, RW3	MOVW@RW0+RW7, RW3	MOVW @RW0+, RW4	MOVW@RW0+RW7, RW4	MOVW @RW0+, RW5	MOVW@RW0+RW7, RW5	MOVW @RW0+, RW6	MOVW@RW0+RW7, RW6	MOVW @RW0+, RW7	MOVW@RW0+RW7, RW7
+D	MOVW @RW1+, RW0	MOVW@RW1+RW7, RW0	MOVW @RW1+, RW1	MOVW@RW1+RW7, RW1	MOVW @RW1+, RW2	MOVW@RW1+RW7, RW2	MOVW @RW1+, RW3	MOVW@RW1+RW7, RW3	MOVW @RW1+, RW4	MOVW@RW1+RW7, RW4	MOVW @RW1+, RW5	MOVW@RW1+RW7, RW5	MOVW @RW1+, RW6	MOVW@RW1+RW7, RW6	MOVW @RW1+, RW7	MOVW@RW1+RW7, RW7
+E	MOVW @RW2+, RW0	MOVW @PC+d16, RW0	MOVW @RW2+, RW1	MOVW @PC+d16, RW1	MOVW @RW2+, RW2	MOVW @PC+d16, RW2	MOVW @RW2+, RW3	MOVW @PC+d16, RW3	MOVW @RW2+, RW4	MOVW @PC+d16, RW4	MOVW @RW2+, RW5	MOVW @PC+d16, RW5	MOVW @RW2+, RW6	MOVW @PC+d16, RW6	MOVW @RW2+, RW7	MOVW @PC+d16, RW7
+F	MOVW @RW3+, RW0	MOVW addr16, RW0	MOVW @RW3+, RW1	MOVW addr16, RW1	MOVW @RW3+, RW2	MOVW addr16, RW2	MOVW @RW3+, RW3	MOVW addr16, RW3	MOVW @RW3+, RW4	MOVW addr16, RW4	MOVW @RW3+, RW5	MOVW addr16, RW5	MOVW @RW3+, RW6	MOVW addr16, RW6	MOVW @RW3+, RW7	MOVW addr16, RW7

Table B.3.1s CH Ri, ea (First byte = 7EH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	XCH R0, R0	XCH R0, @RW0+d8	XCH R1, R0	XCH R1, @RW0+d8	XCH R2, R0	XCH R2, @RW0+d8	XCH R3, R0	XCH R3, @RW0+d8	XCH R4, R0	XCH R4, @RW0+d8	XCH R5, R0	XCH R5, @RW0+d8	XCH R6, R0	XCH R6, @RW0+d8	XCH R7, R0	XCH R7, @RW0+d8
+1	XCH R0, R1	XCH R0, @RW1+d8	XCH R1, R1	XCH R1, @RW1+d8	XCH R2, R1	XCH R2, @RW1+d8	XCH R3, R1	XCH R3, @RW1+d8	XCH R4, R1	XCH R4, @RW1+d8	XCH R5, R1	XCH R5, @RW1+d8	XCH R6, R1	XCH R6, @RW1+d8	XCH R7, R1	NOTW R7, @RW1+d8
+2	XCH R0, R2	XCH R0, @RW2+d8	XCH R1, R2	XCH R1, @RW2+d8	XCH R2, R2	XCH R2, @RW2+d8	XCH R3, R2	XCH R3, @RW2+d8	XCH R4, R2	XCH R4, @RW2+d8	XCH R5, R2	XCH R5, @RW2+d8	XCH R6, R2	XCH R6, @RW2+d8	XCH R7, R2	XCH R7, @RW2+d8
+3	XCH R0, R3	XCH R0, @RW3+d8	XCH R1, R3	XCH R1, @RW3+d8	XCH R2, R3	XCH R2, @RW3+d8	XCH R3, R3	XCH R3, @RW3+d8	XCH R4, R3	XCH R4, @RW3+d8	XCH R5, R3	XCH R5, @RW3+d8	XCH R6, R3	XCH R6, @RW3+d8	XCH R7, R3	NOTW R7, @RW3+d8
+4	XCH R0, R4	XCH R0, @RW4+d8	XCH R1, R4	XCH R1, @RW4+d8	XCH R2, R4	XCH R2, @RW4+d8	XCH R3, R4	XCH R3, @RW4+d8	XCH R4, R4	XCH R4, @RW4+d8	XCH R5, R4	XCH R5, @RW4+d8	XCH R6, R4	XCH R6, @RW4+d8	XCH R7, R4	NOTW R7, @RW4+d8
+5	XCH R0, R5	XCH R0, @RW5+d8	XCH R1, R5	XCH R1, @RW5+d8	XCH R2, R5	XCH R2, @RW5+d8	XCH R3, R5	XCH R3, @RW5+d8	XCH R4, R5	XCH R4, @RW5+d8	XCH R5, R5	XCH R5, @RW5+d8	XCH R6, R5	XCH R6, @RW5+d8	XCH R7, R5	XCH R7, @RW5+d8
+6	XCH R0, R6	XCH R0, @RW6+d8	XCH R1, R6	XCH R1, @RW6+d8	XCH R2, R6	XCH R2, @RW6+d8	XCH R3, R6	XCH R3, @RW6+d8	XCH R4, R6	XCH R4, @RW6+d8	XCH R5, R6	XCH R5, @RW6+d8	XCH R6, R6	XCH R6, @RW6+d8	XCH R7, R6	XCH R7, @RW6+d8
+7	XCH R0, R7	XCH R0, @RW7+d8	XCH R1, R7	XCH R1, @RW7+d8	XCH R2, R7	XCH R2, @RW7+d8	XCH R3, R7	XCH R3, @RW7+d8	XCH R4, R7	XCH R4, @RW7+d8	XCH R5, R7	XCH R5, @RW7+d8	XCH R6, R7	XCH R6, @RW7+d8	XCH R7, R7	XCH R7, @RW7+d8
+8	XCH R0, @RW0	XCH R0, @RW0+d16	XCH R1, @RW0	XCH R1, @RW0+d16	XCH R2, @RW0	XCH R2, @RW0+d16	XCH R3, @RW0	XCH R3, @RW0+d16	XCH R4, @RW0	XCH R4, @RW0+d16	XCH R5, @RW0	XCH R5, @RW0+d16	XCH R6, @RW0	XCH R6, @RW0+d16	XCH R7, @RW0	XCH R7, @RW0+d16
+9	XCH R0, @RW1	XCH R0, @RW1+d16	XCH R1, @RW1	XCH R1, @RW1+d16	XCH R2, @RW1	XCH R2, @RW1+d16	XCH R3, @RW1	XCH R3, @RW1+d16	XCH R4, @RW1	XCH R4, @RW1+d16	XCH R5, @RW1	XCH R5, @RW1+d16	XCH R6, @RW1	XCH R6, @RW1+d16	XCH R7, @RW1	XCH R7, @RW1+d16
+A	XCH R0, @RW2	XCH R0, @RW2+d16	XCH R1, @RW2	XCH R1, @RW2+d16	XCH R2, @RW2	XCH R2, @RW2+d16	XCH R3, @RW2	XCH R3, @RW2+d16	XCH R4, @RW2	XCH R4, @RW2+d16	XCH R5, @RW2	XCH R5, @RW2+d16	XCH R6, @RW2	XCH R6, @RW2+d16	XCH R7, @RW2	XCH R7, @RW2+d16
+B	XCH R0, @RW3	XCH R0, @RW3+d16	XCH R1, @RW3	XCH R1, @RW3+d16	XCH R2, @RW3	XCH R2, @RW3+d16	XCH R3, @RW3	XCH R3, @RW3+d16	XCH R4, @RW3	XCH R4, @RW3+d16	XCH R5, @RW3	XCH R5, @RW3+d16	XCH R6, @RW3	XCH R6, @RW3+d16	XCH R7, @RW3	XCH R7, @RW3+d16
+C	XCH R0, @RW0+	XCH R0, @RW0+RW7	XCH R1, @RW0+	XCH R1, @RW0+RW7	XCH R2, @RW0+	XCH R2, @RW0+RW7	XCH R3, @RW0+	XCH R3, @RW0+RW7	XCH R4, @RW0+	XCH R4, @RW0+RW7	XCH R5, @RW0+	XCH R5, @RW0+RW7	XCH R6, @RW0+	XCH R6, @RW0+RW7	XCH R7, @RW0+	XCH R7, @RW0+RW7
+D	XCH R0, @RW1+	XCH R0, @RW1+RW7	XCH R1, @RW1+	XCH R1, @RW1+RW7	XCH R2, @RW1+	XCH R2, @RW1+RW7	XCH R3, @RW1+	XCH R3, @RW1+RW7	XCH R4, @RW1+	XCH R4, @RW1+RW7	XCH R5, @RW1+	XCH R5, @RW1+RW7	XCH R6, @RW1+	XCH R6, @RW1+RW7	XCH R7, @RW1+	XCH R7, @RW1+RW7
+E	XCH R0, @RW2+	XCH R0, @PC+d16	XCH R1, @RW2+	XCH R1, @PC+d16	XCH R2, @RW2+	XCH R2, @PC+d16	XCH R3, @RW2+	XCH R3, @PC+d16	XCH R4, @RW2+	XCH R4, @PC+d16	XCH R5, @RW2+	XCH R5, @PC+d16	XCH R6, @RW2+	XCH R6, @PC+d16	XCH R7, @RW2+	XCH R7, @PC+d16
+F	XCH R0, @RW3+	XCH R0, addr16	XCH R1, @RW3+	XCH R1, addr16	XCH R2, @RW3+	XCH R2, addr16	XCH R3, @RW3+	XCH R3, addr16	XCH R4, @RW3+	XCH R4, addr16	XCH R5, @RW3+	XCH R5, addr16	XCH R6, @RW3+	XCH R6, addr16	XCH R7, @RW3+	XCH R7, addr16

Table B.3.1t XCHW RWi, ea (First byte = 7FH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	XCHW RW0, RW0	XCHW RW0, @RW0	XCHW RW0, RW1	XCHW RW1, @RW0	XCHW RW2, RW0	XCHW RW2, @RW0	XCHW RW3, RW0	XCHW RW3, @RW0	XCHW RW4, RW0	XCHW RW4, @RW0	XCHW RW5, RW0	XCHW RW5, @RW0	XCHW RW6, RW0	XCHW RW6, @RW0	XCHW RW7, RW0	XCHW RW7, @RW0
+1	XCHW RW0, RW1	XCHW RW0, @RW1	XCHW RW0, RW1	XCHW RW1, @RW1	XCHW RW2, RW1	XCHW RW2, @RW1	XCHW RW3, RW1	XCHW RW3, @RW1	XCHW RW4, RW1	XCHW RW4, @RW1	XCHW RW5, RW1	XCHW RW5, @RW1	XCHW RW6, RW1	XCHW RW6, @RW1	XCHW RW7, RW1	XCHW RW7, @RW1
+2	XCHW RW0, RW2	XCHW RW0, @RW2	XCHW RW0, RW2	XCHW RW1, @RW2	XCHW RW2, RW2	XCHW RW2, @RW2	XCHW RW3, RW2	XCHW RW3, @RW2	XCHW RW4, RW2	XCHW RW4, @RW2	XCHW RW5, RW2	XCHW RW5, @RW2	XCHW RW6, RW2	XCHW RW6, @RW2	XCHW RW7, RW2	XCHW RW7, @RW2
+3	XCHW RW0, RW3	XCHW RW0, @RW3	XCHW RW1, RW3	XCHW RW1, @RW3	XCHW RW2, RW3	XCHW RW2, @RW3	XCHW RW3, RW3	XCHW RW3, @RW3	XCHW RW4, RW3	XCHW RW4, @RW3	XCHW RW5, RW3	XCHW RW5, @RW3	XCHW RW6, RW3	XCHW RW6, @RW3	XCHW RW7, RW3	XCHW RW7, @RW3
+4	XCHW RW0, RW4	XCHW RW0, @RW4	XCHW RW1, RW4	XCHW RW1, @RW4	XCHW RW2, RW4	XCHW RW2, @RW4	XCHW RW3, RW4	XCHW RW3, @RW4	XCHW RW4, RW4	XCHW RW4, @RW4	XCHW RW5, RW4	XCHW RW5, @RW4	XCHW RW6, RW4	XCHW RW6, @RW4	XCHW RW7, RW4	XCHW RW7, @RW4
+5	XCHW RW0, RW5	XCHW RW0, @RW5	XCHW RW1, RW5	XCHW RW1, @RW5	XCHW RW2, RW5	XCHW RW2, @RW5	XCHW RW3, RW5	XCHW RW3, @RW5	XCHW RW4, RW5	XCHW RW4, @RW5	XCHW RW5, RW5	XCHW RW5, @RW5	XCHW RW6, RW5	XCHW RW6, @RW5	XCHW RW7, RW5	XCHW RW7, @RW5
+6	XCHW RW0, RW6	XCHW RW0, @RW6	XCHW RW1, RW6	XCHW RW1, @RW6	XCHW RW2, RW6	XCHW RW2, @RW6	XCHW RW3, RW6	XCHW RW3, @RW6	XCHW RW4, RW6	XCHW RW4, @RW6	XCHW RW5, RW6	XCHW RW5, @RW6	XCHW RW6, RW6	XCHW RW6, @RW6	XCHW RW7, RW6	XCHW RW7, @RW6
+7	XCHW RW0, RW7	XCHW RW0, @RW7	XCHW RW1, RW7	XCHW RW1, @RW7	XCHW RW2, RW7	XCHW RW2, @RW7	XCHW RW3, RW7	XCHW RW3, @RW7	XCHW RW4, RW7	XCHW RW4, @RW7	XCHW RW5, RW7	XCHW RW5, @RW7	XCHW RW6, RW7	XCHW RW6, @RW7	XCHW RW7, RW7	XCHW RW7, @RW7
+8	XCHW RW0, @RW0	XCHW RW0, @RW0	XCHW RW1, @RW0	XCHW RW1, @RW0	XCHW RW2, @RW0	XCHW RW2, @RW0	XCHW RW3, @RW0	XCHW RW3, @RW0	XCHW RW4, @RW0	XCHW RW4, @RW0	XCHW RW5, @RW0	XCHW RW5, @RW0	XCHW RW6, @RW0	XCHW RW6, @RW0	XCHW RW7, @RW0	XCHW RW7, @RW0
+9	XCHW RW0, @RW1	XCHW RW0, @RW1	XCHW RW1, @RW1	XCHW RW1, @RW1	XCHW RW2, @RW1	XCHW RW2, @RW1	XCHW RW3, @RW1	XCHW RW3, @RW1	XCHW RW4, @RW1	XCHW RW4, @RW1	XCHW RW5, @RW1	XCHW RW5, @RW1	XCHW RW6, @RW1	XCHW RW6, @RW1	XCHW RW7, @RW1	XCHW RW7, @RW1
+A	XCHW RW0, @RW2	XCHW RW0, @RW2	XCHW RW1, @RW2	XCHW RW1, @RW2	XCHW RW2, @RW2	XCHW RW2, @RW2	XCHW RW3, @RW2	XCHW RW3, @RW2	XCHW RW4, @RW2	XCHW RW4, @RW2	XCHW RW5, @RW2	XCHW RW5, @RW2	XCHW RW6, @RW2	XCHW RW6, @RW2	XCHW RW7, @RW2	XCHW RW7, @RW2
+B	XCHW RW0, @RW3	XCHW RW0, @RW3	XCHW RW1, @RW3	XCHW RW1, @RW3	XCHW RW2, @RW3	XCHW RW2, @RW3	XCHW RW3, @RW3	XCHW RW3, @RW3	XCHW RW4, @RW3	XCHW RW4, @RW3	XCHW RW5, @RW3	XCHW RW5, @RW3	XCHW RW6, @RW3	XCHW RW6, @RW3	XCHW RW7, @RW3	XCHW RW7, @RW3
+C	XCHW R0, @RW0+	XCHW R0, @RW0+	XCHW RW0, W1, @RW0+	XCHW RW1, @RW0+	XCHW R2, @RW0+	XCHW R2, @RW0+	XCHW R3, @RW0+	XCHW R3, @RW0+	XCHW R4, @RW0+	XCHW R4, @RW0+	XCHW R5, @RW0+	XCHW R5, @RW0+	XCHW R6, @RW0+	XCHW R6, @RW0+	XCHW R7, @RW0+	XCHW R7, @RW0+
+D	XCHW R0, @RW1+	XCHW R0, @RW1+	XCHW RW0, W1, @RW1+	XCHW RW1, @RW1+	XCHW R2, @RW1+	XCHW R2, @RW1+	XCHW R3, @RW1+	XCHW R3, @RW1+	XCHW R4, @RW1+	XCHW R4, @RW1+	XCHW R5, @RW1+	XCHW R5, @RW1+	XCHW R6, @RW1+	XCHW R6, @RW1+	XCHW R7, @RW1+	XCHW R7, @RW1+
+E	XCHW R0, @RW2+	XCHW R0, @PC +d16	XCHW RW0, W1, @RW2+	XCHW RW1, @PC+d16	XCHW R2, @RW2+	XCHW R2, @PC+d16	XCHW R3, @RW2+	XCHW R3, @PC+d16	XCHW R4, @RW2+	XCHW R4, @PC+d16	XCHW R5, @RW2+	XCHW R5, @PC+d16	XCHW R6, @RW2+	XCHW R6, @PC+d16	XCHW R7, @RW2+	XCHW R7, @PC+d16
+F	XCHW R0, @RW3+	XCHW RW0, W0, @RW3+	XCHW RW0, W1, @RW3+	XCHW RW1, addr16	XCHW R2, @RW3+	XCHW R2, addr16	XCHW R3, @RW3+	XCHW R3, addr16	XCHW R4, @RW3+	XCHW R4, addr16	XCHW R5, @RW3+	XCHW R5, addr16	XCHW R6, @RW3+	XCHW R6, addr16	XCHW R7, @RW3+	XCHW RW7, addr16

Appendix C:
The Flash Memory in the MB90F583

C.1 Outline

There is a 1M-bit Flash memory (128K word x 8/64K word x 16) located at the FE~FF bank of the CPU memory map in MB90F583. With the flash memory interface circuit, it is possible for read access from and program access to the CPU. Programming or erasing the flash memory are done by the CPU operation instruction through the flash memory interface circuit. Therefore, with proper CPU control software, it is possible re-programming the flash memory of on-board MB90F583. That means changing of the data in the flash memory of on-board MB90F583 can be done.

- **Features**

  128K word x 8/64K word x 16 bit (16K+8K+8K+32K+64K sector architecture)

  Compatible with JEDEC standard command

  Automatic Algorithm (Embedded^{TM} Algorithm: same as MBM29F400TA)

  - Automatic Program Algorithm
  - Automatic Erase Algorithm

  Sector Erase Suspend/Sector Erase Resume function available

  Program/Erase cycle completion can be detected by data polling, toggle bit and CPU interrupt

  Sector erase function (any combination of sector)

  Number of programming/erasing: 10,000 times (minimum)

  Note: Embedded^{TM} Algorithm is trademarks of Advanced Mirco device, Inc.

- **Program/Erase operation**

  The flash memory of MB90F583 cannot be programmed and read in the same time. When programming or erasing the flash memory, the programming data will be copied to the RAM first; and then executing programming or erasing the flash memory in the RAM. This keeps programming and erasing the flash memory as simple as a writing operation.

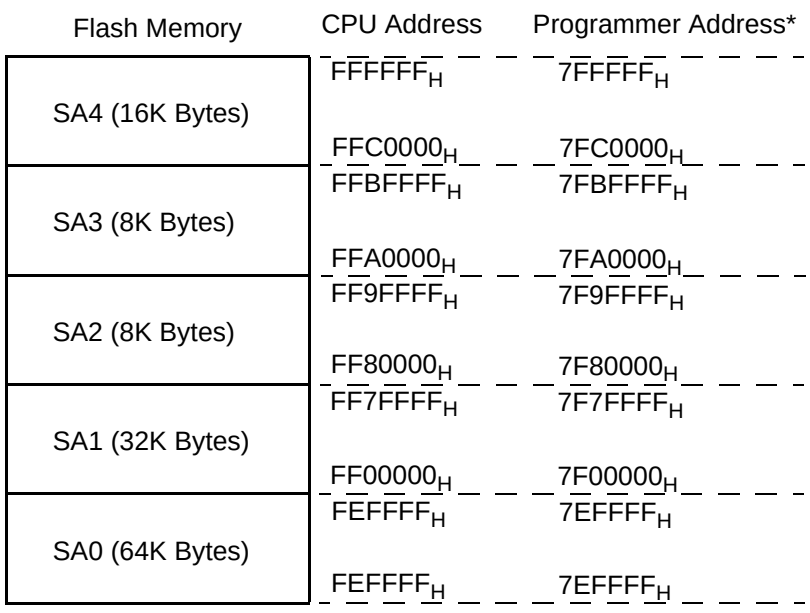
- **Register**

Flash Control Register (FMCS)

	7	6	5	4	3	2	1	0	⇐ Bit Number
Address: 0000AE$_H$	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(W)	(W)	(R/W)	(W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)	

C.2 Sector Structure of 1M Bit Flash Memory

Sector structure of 1M bit flash memory in MB90F583 is shown in Figure C.2a. The address in the Figure C.2a shows upper and lower address of each sector. When accessing from CPU, SA0 is set in the FE bank register and SA1~4 are set in the FF Bank register.



***Programmer address:**

The programmer address is equivalent to the CPU address map where data is programmed to or erased from flash memory by the parallel writer (Minato Electronic: Model 1890A). When programmed to or erased from the flash memory by a general-purpose programmer, this address is needed to specified.

Figure C.2a Sector structure of 1M bit flash memory

C.3 Flash Control Register (FMCS)

Flash control register (FMCS) is a register which is used during programming or erasing the flash memory.

Flash Control Register (FMCS)

	7	6	5	4	3	2	1	0	⇐ Bit Number
Address: 0000AE$_H$	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(W)	(W)	(R/W)	(W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)	

[bit 7] INTE (INTerrupt Enable)

This bit is used to enable an interrupt to the CPU when the operation of programming/erasing the flash memory is completed. An interrupt to the CPU will be generated when the INTE bit is “1” and the RDYINT bit is a “1”. When the INTE bit is “0”, an interrupt will not be generated.

INTE	Interrupt Enable
0	Interrupt enable when program/erase cycle is completed
1	Interrupt disable when program/erase cycle is completed

[bit 6] RDYINT (ReaDY INTerrupt)

This bit is used to show the programming/erasing operation status of the flash memory. This bit will be set to “1” when the flash memory program/erase cycle is completed. After flash memory program/erase cycle is completed and the bit is still “0”, programing/erasing the flash memory is not allowed. Only when this bit is changed to “1”, programming/erasing the flash memory is allowed. Writing “0” will clear this bit to “0” and writing “1” to this bit will be ignored. When Automatic Algorithm (refer to Section C.4, Automatic Algorithm Initiation Method) is completed, this bit will be set to “1”. “1” is always read when read modify write (RWM) is operated.

RDYINT	Ready interrupt
0	Programming/Erasing operation is on-going
1	Programming/Erasing operation is completed (interrupt request generated)

[bit 5] WE (Write Enable)

This bit is “write enable” for the flash memory. When this bit is set to “1”, the flash memory can be programed/erased right after the command sequence to FE~FF bank is issued. Furthermore, this bit is used to start the command for programming/erasing the flash memory. It is recommended to always keep this bit set to “0”, so that the flash memory will not be programmed or erased accidentally.

WE	Write Enable
0	Disable programming/erasing flash memory
1	Enable programming/erasing flash memory

[bit 4] RDY (ReaDY)

This bit is used to indicate whether the flash memory is ready for programming/erasing. When this bit is set to “0”, programming or erasing the flash memory is not allowed. However, it is possible to issue read/reset command and sector erase suspend command when this bit is “0”.

RDY	Ready
0	Programming/erasing is operating
1	Programming/erasing is completed (next data programming/erasing is enabled.

[bit 3] Reserved bit

This bit is reserved. It is recommended to always set this bit to “0” during normal operation.

[bit 0] Reserved bit

This bit is reserved. It is recommended to always set this bit to “0” during normal operation.

[bit 2, 0] LPM1, LPM0 (Low Power Mode)

When accessing flash memory, these two bits are used to control the power consumption of the flash memory. This bit cannot be set to “00”. After reset, these bit must be set to “01”, “10” or “11”. Since the flash memory access time by the CPU will be changed according to the frequency of the operating clock, it is recommended to set these bit according to the operating clock frequency of the CPU.

LMP0	LMP1	Low Power Mode
0	**0**	Initial value (Access prohibited)
0	**1**	Low power mode (Internal operation frequency < 4 MHz)
1	**0**	Low power mode (Internal operation frequency < 8 MHz)
1	**1**	Low power mode (Internal operation frequency < 16 MHz)

Note: RDYINT bit and RDY bit cannot be changed in the same time. Either one of these two bits should be changed when writing the control software.

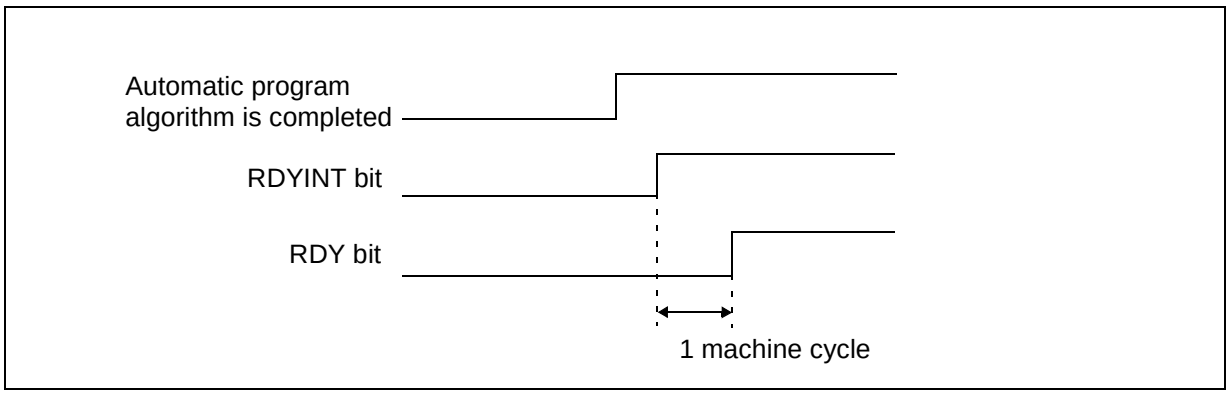


Figure C.3a Timing of RDYINT and RDY

C.4 Automatic Algorithm Initiation Method

To start the Automatic Algorithm in the flash memory, there are five types of commands, 2 types of read/reset, programming, chip erase and sector erase. For sector erase, there are the sector erase suspend and the sector erase resume command.

Table C.4a shows the commands used during programming/erasing the flash memory. Although the data shown in the command are all in byte, it is necessary to use word access to write the data. At this time, the upper byte of the data will be ignored.

Table C.4a Command Sequence Definitions

Command Sequence	Bus Write Cycle Req'd	1st Bus Write Cycle		2nd Bus Write Cycle		3rd Bus Write Cycle		4th Bus Write Cycle		5th Bus Write Cycle		6th Bus Write Cycle	
		Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Read/Reset*	1	FxXXXX	XXF0	—	—	—	—	—	—	—	—	—	—
Read/Reset*	3	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXF0	RA	RD	—	—	—	—
Programming	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXA0	PA (even)	PD (word)	—	—	—	—
Chip Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX10
Sector Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	SA (even)	XX30
Sector Erase Suspend	Sector erase is suspend by inputting the address "FxXXXX", data "xxB0$_H$"												
Sector Erase Resume	Sector erase is resumed by inputting the address "FxXXXX", data "xx30$_H$"												

Note: The address Fx in Table C.4a is either FE or FF for MB90F583. When using above commands, the accessible bank value for the device must be used to replaced Fx

The address found in the Table C.4a is corresponding to the CPU memory address. All address and data written in hexadecimal and "X" is arbitrary value.

RA: Read address

PA: Program address , only even number address can be selected

SA: Sector address, refer to Section C.2, Sector Structure of 1M Bit Flash Memory.

RD: Read data

PD: program data, only word data can be selected

* : The 2 types of read/reset command can be reset the flash memory to read mode.

C.5 Execution Status of Automatic Algorithm

In the flash memory, the programming or erasing can be done by Automatic Algorithm, so that there is a Hardware Sequence Flag in the flash memory, which indicates the operation status and the operation completion. In the Automatic Algorithm, internal flash memory operation status can be checked by the hardware sequence flag which will be discussed in this section.

- **Hardware Sequence Flag**

  Hardware Sequence Flag consists of 4 flags, DQ7 (Data polling flag), DQ6 (Toggle bit flag), DQ5 (Exceeded timing limits flag) and DQ3 (Sector erase timer flag). These flag are used to check whether the programming or erasing the flash memory is completed and whether erase code are valid.

  Hardware sequence flag is a checking point when performing read access to the address of the sector in the flash memory and after issuing the command sequence (see Table C.4a). Table C.5a shows the bit assignment of the hardware sequence flag.

Table C.5a Hardware sequence flag's bit assignment

Bit number	7	6	5	4	3	2	1	0
Hardware sequence Flag	DQ7	DQ6	DQ5	—	DQ3	—	—	—

To check whether the Automatic Program/Erase Algorithm is under processing, it can be determined by either checking the hardware sequence flag or RDY bit of the flash control register (FMCS). After programming/erasing operation is completed, the flash memory will return to read/reset status. When making a control software, it is necessary to check the Automatic Program/Erase Algorithm completion by either the hardware sequence flag or RDY bit of the flash control register (FMCS) before going to other process such as reading data. It is also possible to check the next and the following sector erase code issued is valid by the hardware sequence flag. Table C.5b shows the function of each hardware sequence flag.

Table C.5b Hardware Sequence Flag

Status		DQ7	DQ6	DQ5	DQ3
Status Change in normal operation	Programming → Programming complete (When program address is indicated)	$\overline{DQ7}$ → DATA:7	Toggle → DATA:6	0 → DATA:5	0 → DATA:3
	Chip/Sector erase → Erase is completed	0 → 1	Toggle → Stop	0 → 1	1
	Sector erase wait → Erase start	0	Toggle	0 → 1	1
	Sector erase → Sector erase suspend (Sector being erased)	0 → 1	Toggle → 1	0	1 → 0
	Sector erase suspend → Sector erase resume (Sector being erased)	1 → 0	1 → Toggle	0	0 → 1
	Sector erase suspend is in progress (Sector not being erased)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal Operation	Programming operation	$\overline{DQ7}$	Toggle	1	0
	Chip/Sector erase	0	Toggle	1	1

C.5.1 Data polling flag (DQ7)

Data polling flag is used to indicate whether the Automatic Algorithm is executing or completed by using data polling function. Table C.5.1a shows the status change of the data polling flag.

- **Programming**

  During Automatic Program Algorithm is executing, an attempt to read the flash memory will output the complement of the last written data to DQ7, rather than the value at the address specified by the current address signal.

- **Chip/Sector Erase**

  During chip erase/sector erase operation is in progress, an attempt to read the flash memory will output "0" to DQ7. Upon completion of chip erase/sector erase, an attempt to read the flash memory will output "1" to DQ7.

- **Sector Erase Suspend**

  During sector erase suspend is in progress, an attempt to read the flash memory will output "1" to DQ7, if the address is within the sector which is being erased. If the address is not within the sector being erased, the flash memory will output bit 7 (DATA:7) of the read value of the address which is pointed. By looking at the toggle bit flag (DQ6) together, it is possible to know whether the present sector is in sector erase suspend mode, or to know which sector is being erased.

Note: An attempt to read access to the address where Automatic Algorithm is starting will be ignored. After receiving the completion status of data polling flag (DQ7), an attempt to read access will be allowed. Hence, a read access from Automatic Algorithm completion should be done after the read access of the data polling completion.

Table C.5.1a Status Change of data polling flag (DQ7)

- **Status Change in normal operation**

Operation status	Programming → complete	Chip/sector erase → complete	Sector erase wait → start	Sector erase → suspend (Sector being erased)	Sector erase suspend → resume (Sector being erased)	Sector erase being suspended (Sector not being erased)
DQ7	$\overline{DQ7}$ → DATA:7	0 → 1	0	0 → 1	1 → 0	DATA:7

- **Status Change in abnormal operation**

Operation Status	Programming Operation	Chip/sector erase operation
DQ7	$\overline{DQ7}$	0

C.5.2 Toggle bit flag (DQ6)

Toggle bit flag is used to indicate whether the Automatic Algorithm is in progress or is completed by using toggle bit function. Table C.5.2a shows status change of the toggle bit flag.

- **Programming, Chip and Sector Erase**

  During Automatic Program or Erase Algorithm, successive attempts to read access to the flash memory will result in toggling DQ6 between “1” and “0”. When Automatic Program Algorithm and Automatic Chip/Sector Erase Algorithm is completed and continuous read access is attempted, the flash memory will stop the DQ6 toggling and output the value of bit 6 of the address specified by the current address signals.

- **Sector Erase Suspend**

  When an attempt to read access in sector erase suspend mode, read value will be “1” if the address is specified within the sector being sector erased. If the specified address is not within to the sector being sector erased, the flash memory will output the value of bit 6 of the address specified by the current address signals.

Note: In programming operation, if the sector to be programmed is write-protected, the DQ6 will be toggled for about 2 μS and then stop toggling without having the data change. In erasing operation, if all sectors are write-protected, the DQ6 will be toggled for about 100 μS and then go back into read/reset mode without having the data change.

Table C.5.2a Status Change of toggle bit flag (DQ6)

- **Status Change in normal operation**

Operation status	Programming → complete	Chip/sector erase → complete	Sector erase wait → start	Sector erase → suspend (Sector being erased)	Sector erase suspend → resume (Sector being erased)	Sector erase being suspended (Sector not being erased)
DQ6	Toggle → DATA:6	Toggle → Stop	Toggle	Toggle → 1	1 → Toggle	DATA:6

- **Status Change in abnormal operation**

Operation Status	Programming Operation	Chip/sector erase operation
DQ6	Toggle	Toggle

C.5.3 Exceeded timing limits flag (DQ5)

Exceeded timing limits flag is used to indicate whether Automatic Algorithm has executed beyond the time (internal pulse count) specified in the flash memory. Table C.5.3a shows status change of the exceeded timing limits flag.

- **Programming, Chip and Sector Erase**

  An attempt to read access after programming or chip/sector erase operation will output “0” to DQ5 if Automatic Algorithm has executed within the time (internal pulse count) specified in the flash memory. If it is beyond the limit, “1” will be output to DQ5. With irrespective of he Automatic Algorithm operation status, It is used to determine whether the program/erase operation has suceeded. Thus, when “1” is read, it shows that programming or erasing operation is failed if Automatic Algorithm is regarded as still being executed by data polling function or toggle bit function.

  For an example, If the user tries to write “1” to the flash memory address where “0” is written, a failure will occur. In this case, flash memory will be locked and Automatic Algorithm will not be completed. Consequently, valid data will not be outputted from the data polling flag (DQ7). In the case of toggle bit flag (DQ6), the toggle operation on bit 6 will not stopped and bit 5 output “1” to the exceeded timing limits flag (DQ5). It means that the flash memory is not defective and it has been used incorrectly. The operation will return to normal after executing a reset command.

Table C.5.3a Status Change of exceeded timing limits flag (DQ5)

- **Status Change in normal operation**

Operation status	Programming → complete	Chip/sector erase → complete	Sector erase wait → start	Sector erase → suspend (Sector being erased)	Sector erase suspend → resume (Sector being erased)	Sector erase being suspended (Sector not being erased)
DQ5	0 → DATA5	0 → 1	0	0	0	DATA:5

- **Status Change in abnormal operation**

Operation Status	Programming Operation	Chip/sector erase operation
DQ5	1	1

C.5.4 Sector erase timer flag (DQ3)

Sector erase timer flag is used to indicate whether the Automatic Algorithm is executed beyond the sector erase wait time after the sector erase command is issued. Table C.5.4a shows status change of the sector erase timer flag.

- **During sector erase operation**

  An attempt to read access after sector erase command is issued will output “0” to DQ3 if Automatic Algorithm is executed within the sector erase wait time. “1” will be output to DQ3 if the Automatic Algorithm is executed beyond the sector erase wait time. If the data polling flag or toggle bit flag indicates that the Automatic Erase Algorithm is operating and DQ3 is “1”, internally controlled erasing is started. Attempts to issue erase code and command other than sector erase suspend to the sector will be ignored until the erasing is completed. When DQ3 is “0”, issuing the additional sector erase code will be accepted. To ensure the command has been accepted, the control software should check the status of DQ3 prior to each subsequent sector erase command. If DQ3 was “1” on the status checking, the command may not be accepted.

- **During sector erase suspend**

  When read accessing during sector erase suspend, “1” will be output to DQ3 if the specified address is within the sector which is being erased. If it does not within the sector being erased, the flash memory will output the value of bit 3 (DATA:3) at the address specified by the current address signals.

Table C.5.4a Status Change of sector erase timer flag (DQ3)

- **Status Change in normal operation**

Operation status	Programming → complete	Chip/sector erase → complete	Sector erase wait → start	Sector erase → suspend (Sector being erased)	Sector erase suspend → resume (Sector being erased)	Sector erase being suspended (Sector not being erased)
DQ3	0 → DATA:3	1	0 → 1	1 → 0	0 → 1	DATA:3

- **Status Change in abnormal operation**

Operation Status	Programming Operation	Chip/sector erase operation
DQ3	0	1

C.6 Details of Flash Memory Programming/Erasing

This section describes the following: command generated for initiating Automatic Algorithm, read/reset of flash memory, programming, chip erase, sector erase suspend and sector erase resume.

Flash memory can execute Automatic Algorithm when repeating the bus write cycle in read/reset, programming, chip erase, sector erase, sector erase suspend and sector erase resume command sequence. The bus write cycle must be sent continuously. Completion of the Automatic Algorithm can be determined by checking the hardware sequence flag such as data polling function. If it is completed correctly, the flash memory will return to read/reset status.

C.6.1 Read/reset status

This section describes how to isssue read/reset command to make flash memory returning to read/reset status.

To make the flash memory returning to read/reset status, the command sequence of the read/reset command found in command sequence table (refer to Section C.4, Automatic Algorithm Initiation Method, Table C.4a) can be used and it needs to be continuously sent to the target sector in the flash memory.

There are two kinds of bus write cycles in read/reset command, 1 and 3 bus write cycles, but there are no significant difference between them.

Since read/reset is the initial state, the flash memory will always go to this state when power-on and any command is correctly implemented. Read/reset status is a waiting state for other command. Normal read access can be done in the read/reset status.

Programming access is possible from the CPU. This command is not necessary for reading data in normal read access. This command is used when the operation is not completed correctly for some reasons, or when automatic algorithm needs to be reseted.

C.6.2 Data Programming

This section will describe how to issue the programming command to program the flash memory.

To initiate Automatic Program Algorithm in the flash memory, the programming command found in command sequence table (refer to Section C.4, Automatic Algorithm Initiation Method, Table C.4a) can be used and it needs to be sent continuously to the target sector in the flash memory. Automatic Algorithm will be initiated and automatic programming will start when data programming to the target address is completed at the 4th cycle.

- **Specifying Address**

  Specified programming address must be even number. It is not possible to program correctly on odd numbered addresses, so that it is necessary to program by word data unit with the even numbered address. Programming can be done in any address and can go over sector boundary. However, only one word can be programmed with a single programming command.

- **Precautions on Data Programming**

  It is impossible to program data from "1" to "0". When programming data from "1" to "0", data polling function (DQ7) and toggle bit function (DQ6) will not be completed. In this case, either the flash memory is considered to have error and programming timing limit will exceed making the exceeded timing limits flag (DQ5) to output an error, or "1" will be assumed to have been programmed. When reading data in the read/reset status, the data remains "0". Only erase operation can change data from "0" to "1".

  While automatic programming the flash memory is under processing, all other command will be ignored. If hardware reset is initiated during programming, take a good care on it. It is because the data being programmed to the address will not be guaranteed.

- **Data Programming Procedure**

  Figure C.6.2a shows the example procedure of programming the flash memory. By checking the hardware sequence flag (refer to Section C.5, Execution Status of Automatic Algorithm), the status of Automatic Algorithm in flash Memory can be determined. Data polling flag (DQ7) is used to check whether programming is completed. The data read from DQ7 is the data found in the next programming address. It is necessary to re-check the data polling flag bit (DQ7) even if the exceeded timing limits flag (DQ5) is "1". It is because data polling flag (DQ7) and the exceed time limit flag (DQ5) will change in the same time. Furthermore, it is not need to re-check the toggle bit flag(DQ6) since it will stop at the same time when exceeded timing limits flag (DQ5) changes to "1".

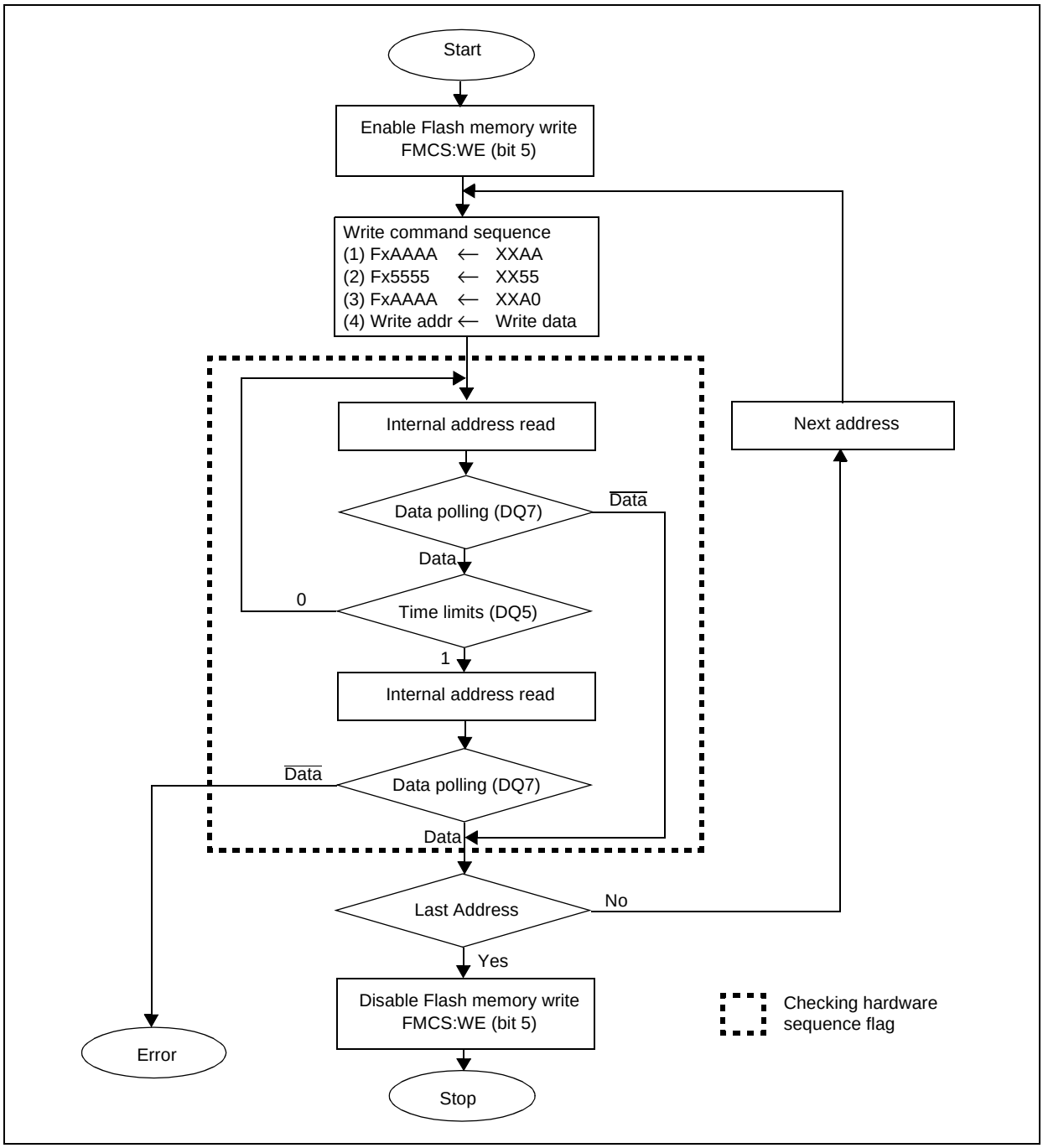


Figure C.6.2a Example procedure of programming the flash memory

C.6.3 Chip Erase

This section will describe how to issue the chip erase command to erase the whole chip.

To erase all data from the flash memory, the chip erase command found in command sequence table (refer to Section C.4, Automatic Algorithm Initiation Method, Table C.4a) can be used and needs to be send continuously to the target address in the flash memory.

Chip erase command is executed by six bus write cycles. Chip erase operation will start after 6th bus write cycle is finished. It is no necessary for the user to program the flash memory before erasing the chip. During Automatic Erase Algorithm execution, flash Memory will automatically write "0" to all bits before chip erase is operated.

C.6.4 Sector Erase

This section will describe how to issue the sector erase command to erase any sector in the flash memory.

Single sector or multiple sector can be erased in the same time.

To erase a sector in the flash Memory, the sector erase command found in command sequence table (refer to Section C.4, Automatic Algorithm Initiation Method, Table C.4a) can be used and needs to be sent continuously to the target sector in the flash memory.

- **Specifying sector**

  Sector erase command is executed by six bus write cycles. 50 μS of sector erase wait time will be started after issuing sector erase code (30H) to the accessible even numbered address of the sector in the 6th bus write cycle. When erasing several sectors in the same time, the erase code (30H) to the address of sectors to be erased needs to be issued.

- **Precautions on Specifying Multiple Sectors**

  Sector erase operation will be started when the 50 μS of sector erase wait time is completed after the last sector erase code is issued. When erasing several sectors, it is necessary to input the address and the erase code of the following sector to erase in the 50 μS of sector erase wait time. However, the sector erase operation may not be accepted even after this wait time. It is necessary to check the sector erase timer flag (DQ3) to ensure whether the sector erase code issued was valid. At this time, the address to read the sector erase timer flag (DQ3) should be specified to the sector to be erased.

- **Sector Erase Procedure**

  By checking hardware sequence flag, the status of Automatic Algorithm in flash memory can be determined (refer to 1.5 Automatic algorithm execution status). Figure C.6.4a shows the example procedure of sector erase in the Flash Memory. In this example procedure, the toggle bit flag (DQ6) is used to check erase completion Take note that the data read for DQ6 is the data in the sector that will be erased. It is not necessary to check the data polling flag (DQ7) even if the exceeded timing limits flag (DQ5) is "1". It is because data polling flag (DQ7) will change at the same time the exceed timing limits flag (DQ5) is changed. Furthermore, it is necessary to re-check the toggle bit flag (DQ6) since it will stop at the same time exceeded timing limits flag (DQ5) changes to "1".

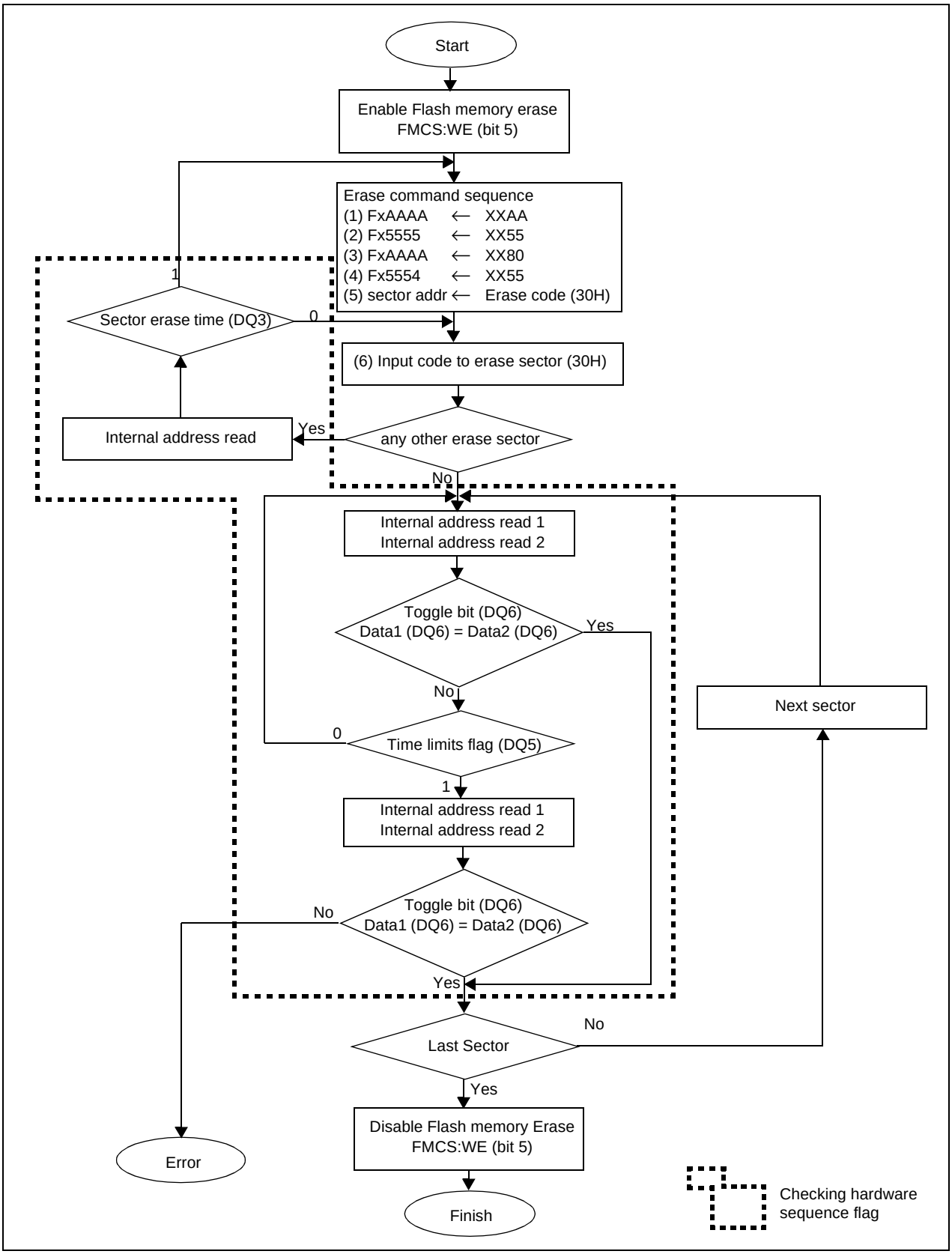


Figure C.6.4a Example flowchart of erasing flash memory

C.6.5 Suspend Sector Erase

This section will describe how to issue the sector erase suspend command to suspend sector erase operation in the flash memory. During sector erase, it is possible to read data from the sector which is not being erased.

To suspend sector erase in flash memory, the sector erase suspend command found in command sequence table (refer to Section C.4, Automatic Algorithm Initiation Method, Table C.4a) can be used and needs to be sent continuously to the target sector in the flash memory.

During sector erase suspend command is executing, it is possible to read data from the sector that is not being erased. This enables only reading from the sector but it is not possible to programming the sector. This command is valid only during sector erasing time including the erase wait time, However, this command will be ignored when chip erase is operating or programming is operated.

It will be implemented by issuing erase suspend code (B0H) to the flash memory. The address will be specified to any address within the Flash Memory.

Sector erase suspend command will be ignored during another erase suspend command. If the sector erase suspend command is issued during sector erase wait, sector erase wait will be ended suddenly and sector erase will be suspended. If sector erase suspend command is issued when sector erase is operating after sector erase wait, it will go to sector erase suspend status after maximum of 15 μS.

C.6.6 Resume Sector Erase

This section will describe how to issue the sector erase resume command to restart the suspended sector erase in the flash memory.

To restart the suspended sector erase, the sector erase resume command found in command sequence table (refer to Section C.4, Automatic Algorithm Initiation Method, Table C.4a) can be used and needs to be sent continuously to the target sector in the flash memory.

Sector erase resume command is used to restart the sector erase operation form sector erase suspend status. It will be implemented by issuing the sector erase resume code (30H) to the flash memory. The address can be specified to any address within the flash memory. Sector erase resume command will be ignored during sector erase is operating.

FUJITSU LIMITED

For further information please contact:

Japan
FUJITSU LIMITED
Corporate Global Business Support Division
Electronic Devices
KAWASAKI PLANT, 4-1-1, Kamikodanaka
Nakahara-ku, Kawasaki-shi
Kanagawa 211-8588, Japan
Tel: (044) 754-3763
Fax: (044) 754-3329

http://www.fujitsu.co.jp/

North and South America
FUJITSU MICROELECTRONICS, INC.
Semiconductor Division
3545 North First Street
San Jose, CA 95134-1804, USA
Tel: (408) 922-9000
Fax: (408) 922-9179

Customer Response Center
Mon. - Fri.: 7 am - 5 pm (PST)
Tel: (800) 866-8608
Fax: (408) 922-9179

http://www.fujitsumicro.com/

Europe
FUJITSU MIKROELEKTRONIK GmbH
Am Siebenstein 6-10
D-63303 Dreieich-Buchschlag
Germany
Tel: (06103) 690-0
Fax: (06103) 690-122

http://www.fujitsu-ede.com/

Asia Pacific
FUJITSU MICROELECTRONICS ASIA PTE LTD
#05-08, 151 Lorong Chuan
New Tech Park
Singapore 556741
Tel: (65) 281-0770
Fax: (65) 281-0220

http://www.fmap.com.sg/

F9806

Known bugs in HM MB90580

1. Chapter 20.4.5 Output Compare Unit
==========

The documentation refers to outputs OUT0/1 and OUT2/3.
There does not exist OUT2 and OUT3 (see pinning).
So compare register 0 corresponds to OUT0 only and compare register 1 to OUT1.

last updated : 05-03-98
TKa