FUJITSU SEMICONDUCTOR
CONTROLLER MANUAL

CM43-10106-1E

F²MC-16L
16-BIT MICROCONTROLLER
MB90610A SERIES
HARDWARE MANUAL

FUJITSU

Preface

Thank you for purchasing a Fujitsu semiconductor product.

The MB90610A series has been developed as a general-purpose product in the F^{2}MC-16L series. The F^{2}MC-16L series are proprietary 16-bit single-chip microcontrollers that can be used as application specific ICs (ASICs).

This manual describes the functions and operation of the MB90610A series and is aimed at engineers who are using the chip to develop products. For details on the instruction set, see the "F^{2}MC-16L Programming Manual".

*: F^{2}MC stands for Fujitsu Flexible Microcontroller.

This manual is organized as follows.

Chapter 1 General
Describes the MB90610A series features, product range, block diagram, pin assignment, and notes on device operation.

Chapter 2 Hardware
Describes the internal structure of the F^{2}MC-16L series CPU and the internal hardware specifications of the MB90610A series.

Chapter 3 Operation
Describes the clock generator, reset, interrupts, memory access modes, low power modes, and other features of the MB90610A series.

Chapter 4 Instructions
Summarizes the F^{2}MC-16L series instruction set.

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

CONTENTS

Chapter 1: General

The MB90610A series are general-purpose, high performance 16-bit microcontrollers designed for applications requiring high speed real-time processing in industry, office equipment, process control, and other fields.

The instruction set follows the F^{2}MC-8 series AT architecture with additional high level language instructions, enhanced addressing modes, improved multiplication and division instructions, and bit manipulation instructions. Furthermore, a 32-bit accumulator enables processing of long-word data.

The internal peripheral resources consist of a 3-channel serial port incorporating a UART function (and supporting I/O expansion serial mode), 8-channel 10-bit A/D converter, 2-channel PPG, 2-channel 16-bit reload timer, 8-channel chip select output, and eight external interrupts.

Also, multiplexed or non-multiplexed operation can be selected for the address/data bus.

1.1 Features

(1) Minimum instruction execution time (Standard F^{2}MC-16 features)**:**

62.5 nS at 16 MHz internal operation

Uses PLL clock multiplication

(2) Instruction set optimized for controller applications (Standard F^{2}MC-16 features)

- Wide range of data types (bit, byte, word, and long word)
- Wide range of addressing modes: 23 modes
- High code efficiency
- High accuracy operations are enhanced by use of a 32-bit accumulator.

(3) Enhanced high level language (C) and multitasking support instructions

(Standard F^{2}MC-16 features)

- Use of a system stack pointer
- Enhanced pointer indirect instructions
- Barrel shift instructions

(4) Improved execution speed (Standard F^{2}MC-16 features)**:**

Four byte instruction queue

(5) Powerful interrupt function from 24 sources in 8 levels (Standard F^{2}MC-16 features)

(6) Automatic data transfer that is independent of the CPU (Standard F^{2}MC-16 features)

(7) Multiplexed or non-multiplexed operation can be selected for the address/data bus

(8) General-purpose ports

Non-multiplexed mode: 36 ports max.

Multiplexed mode: 52 ports max.

(9) UART (SCI): 3ch

- For either asynchronous or clocked serial transfer (I/O expansion serial)

(10) A/D converter: 8ch (10-bit)

- 8-bit conversion mode also available

(11) PPG (programmable pulse generator): 2ch

(12) 16-bit reload timer: 2ch

(13) Chip select output: 8ch

(14) External interrupts: 8ch

(15) 18-bit timebase timer

- Watchdog timer function

(16) PLL clock multiplier function

(17) CPU intermittent operation function

(18) Various standby modes

(19) SQFP-100 or QFP-100 package

(20) CMOS technology

1.2 Product Range

Table 1.2.1 lists the MB90610A series product range. Features other than ROM and RAM size are the same for all products.

Table 1.2.1 MB90610A Series Product Range

	MB90611A		MB90V610A
ROM size	-		
RAM size	1KB		4KB
Other	No ROM		Evaluation device

* At the time of writing this manual, actual product range details are still to be confirmed.
  The above product range is provisional and does not guarantee the future availability of products.

1.3 Block Diagram

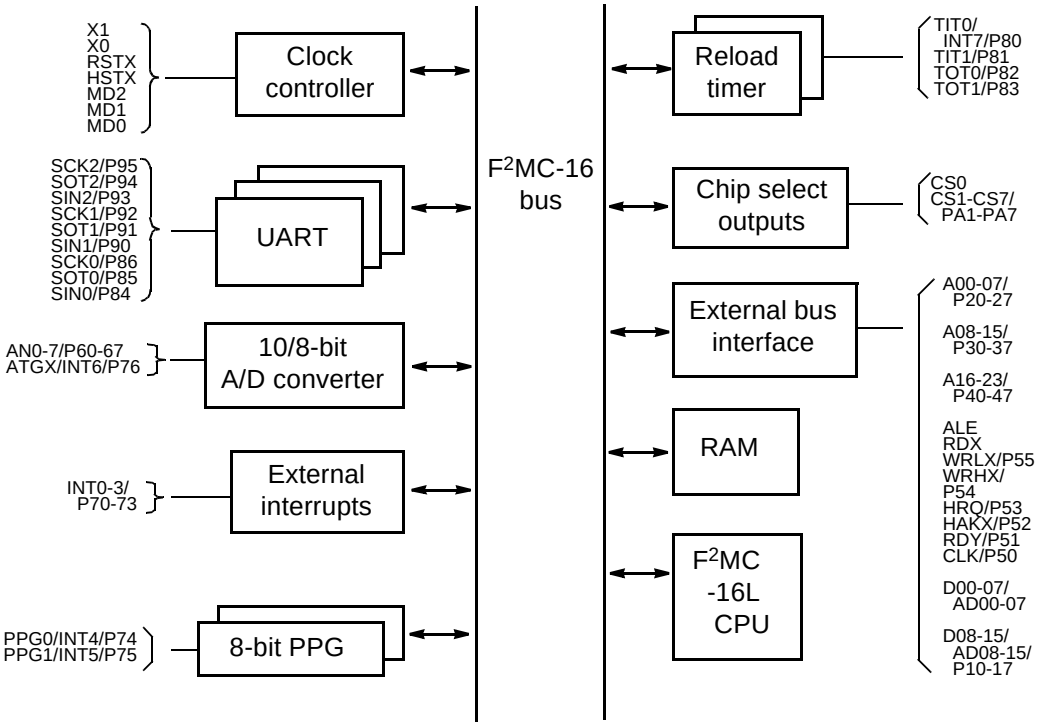


Fig. 1.3.1 Block Diagram of the MB90610A Internal Structure

1.4 Pin Assignment

1.4.1 SQFP-100 Pin Assignment

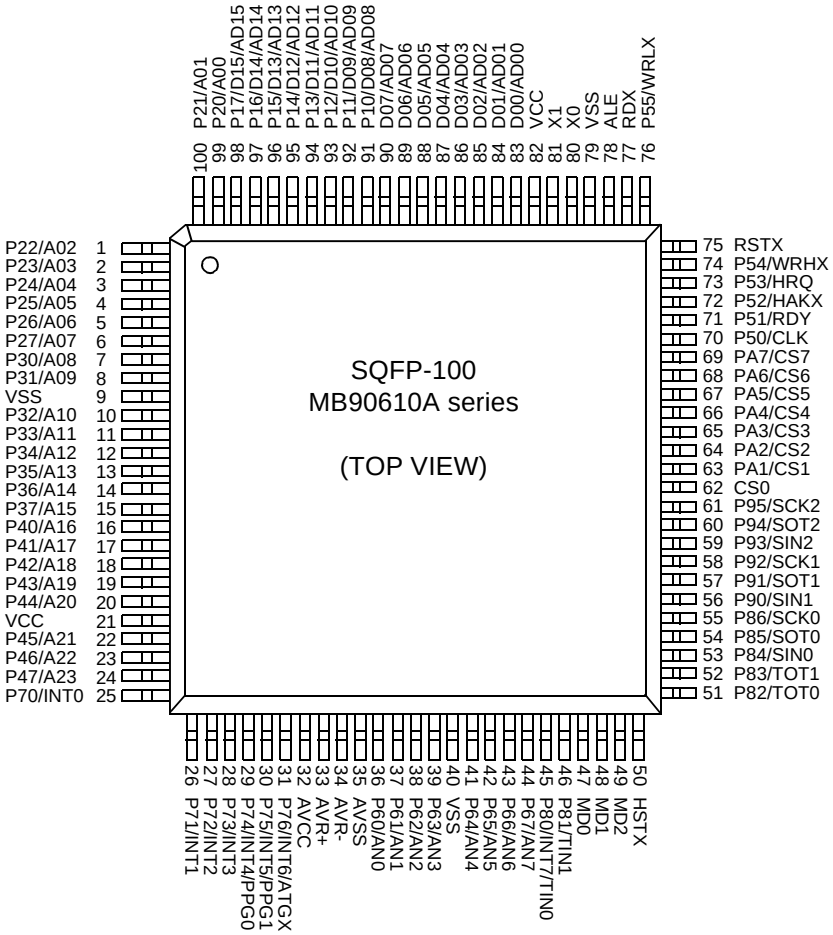


Fig. 1.4.1 SQFP-100 Pin Assignment

1.4.2 QFP-100 Pin Assignment

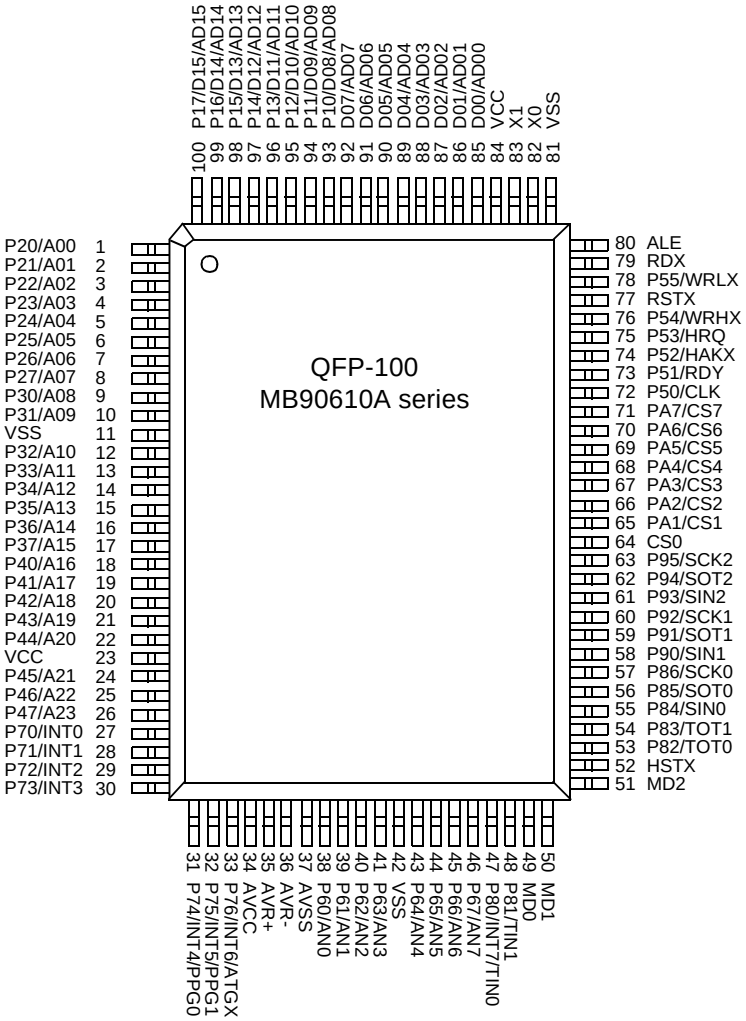


Fig. 1.4.2 QFP-100 Pin Assignment

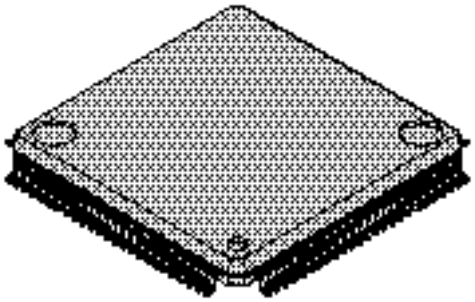
1.5 Package Dimensions

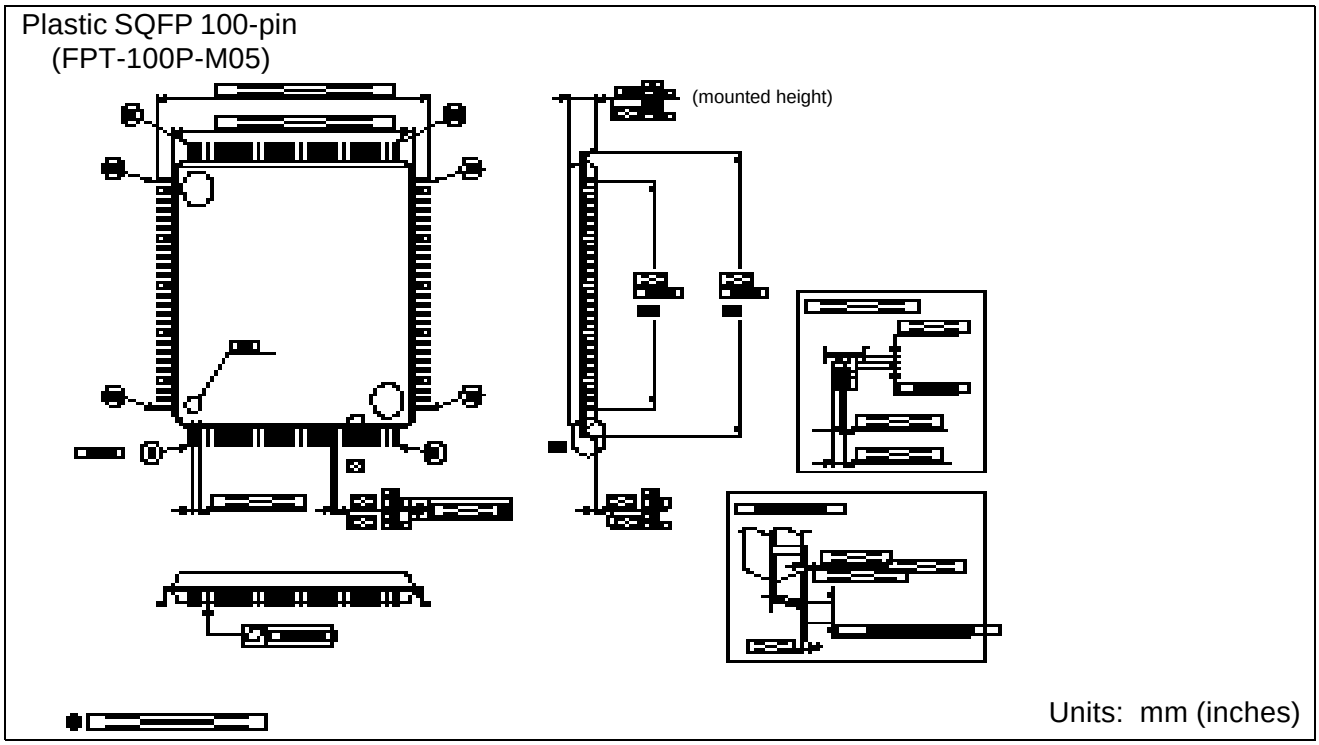
1.5.1 SQFP-100 Package Dimensions

FPT-100P-

Reference

EIAJ Code: *QFP100-P-1414-1

Plastic SQFP 100-pin		
 (FPT-100P-M05)	Lead pitch	0.50 mm
	Package width × length	14 × 14 mm
	Lead shape	Gull-wing
	Sealing	Plastic mold



* The above package dimensions are for reference only. Please confirm the actual dimensions separately.

Fig. 1.5.1 SQFP-100 Package Dimensions

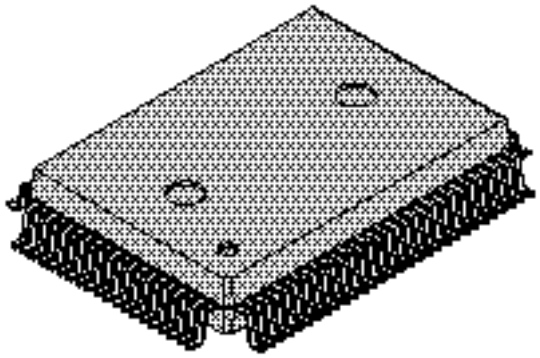
1.5.2 QFP-100 Package Dimensions

FPT-100P-

Reference

EIAJ Code: *QFP100-P-1420-4

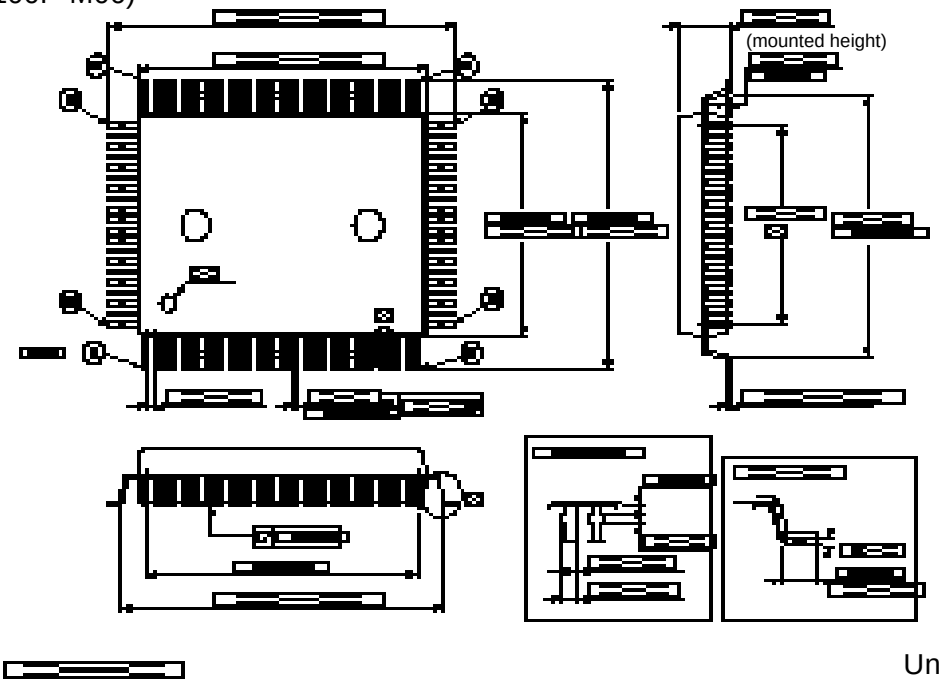
Plastic QFP 100-pin



(FPT-100P-M06)

Lead pitch	0.65 mm
Package width × length	14 × 20 mm
Lead shape	Gull-wing
Sealing	Plastic mold
Length of pin flat section	0.80 mm

Plastic QFP 100-pin
(FPT-100P-M06)



Units: mm (inches)

* The above package dimensions are for reference only. Please confirm the actual dimensions separately.

Fig. 1.5.2 QFP-100 Package Dimensions

1.6 Pin Descriptions

Table 1.6.1 MB90610A Pin Descriptions (1)

Pin No.		Pin Name	Circuit Type	Function
QFP	SQFP			
82 83	80 81	X0 X1	A (Oscillator)	Crystal oscillator pins
85 to 92	83 to 90	D00 to D07	K (TTL)	In non-multiplex mode, the I/O pins for the lower 8 bits ofthe external data bus.
		AD00 to AD07		In multiplexed mode, the I/O pins for the lower 8 bits of the external address/data bus.
93 to 100	91 to 98	P10 to P17	K (TTL)	General-purpose I/O ports. This applies in non-multiplexed mode with an 8-bit external data bus.
		D08 to D15		In non-multiplexed mode, the I/O pins for the upper 8 bits of the external data bus. This applies when using a 16-bit external data bus.
		AD08 to AD15		In multiplexed mode, the I/O pins for the upper 8 bits of the external address/data bus.
1 to 8	99 100 1 to 6	P20 to P27	B (CMOS)	General-purpose I/O ports. This applies in multiplexed mode.
		A00 to A07		In non-multiplexed mode, the output pins for the lower 8 bits of the external address bus.
9 10 12 to 17	7 8 10 to 15	P30 to P37	B (CMOS)	General-purpose I/O ports. This applies in multiplexed mode.
		A08 to A15		In non-multiplexed mode, the output pins for the upper 8 bits of the external address bus.
18 to 22 24 to 26	16 to 20 22 to 24	P40 to P47	B (CMOS)	General-purpose I/O ports. This applies when the upper address control register specifies port operation.
		A16 to A23		The output pins for A16 to 23 of the external address bus. This applies when the upper address control register specifies address operation.
27 to 30	25 to 28	P70 to P73	H (CMOS/H)	General-purpose I/O ports. This applies in all cases.
		INT0 to INT3		External interrupt request input pins. As the inputs operate continuously when external interrupts are enabled, output to the pins from other functions must be stopped unless doneintentionally.
31 32	29 30	P74 to P75	H (CMOS/H)	General-purpose I/O ports. This applies when the waveform outputs for PPG timers 0 and 1 are disabled.
		INT4 to INT5		External interrupt request input pins. As the inputs operate continuously when external interrupts are enabled, output to the pins from other functions must be stopped unless doneintentionally.
		PPG0 to PPG1		Output pins for PPG timers 0 and 1. This applies when the waveform outputs for PPG timers 0 and 1 are enabled.
33	31	P76	H (CMOS/H)	General-purpose I/O port. This applies in all cases.
		INT6		External interrupt request input pin. As the input operatescontinuously when the external interrupt is enabled, outputto the pin from other functions must be stopped unless doneintentionally.
		ATGX		Trigger input pin for the A/D converter. As the input operates continuously when the A/D converter inputs are operating, output to the pin from other functions must be stopped unless done intentionally.
34	32	AVCC	Power supply	Power supply for the analog circuits. Do not switch this power supply on or off unless a voltage greater than AVCC is applied to VCC.
35	33	AVR+	Power supply	Analog circuit reference voltage input. Do not switch the voltage to this pin on or off unless a voltage greater than AVR+ is applied to AVCC.
36	34	AVR-	Power supply	Analog circuit reference voltage input
37	35	AVSS	Power supply	Ground level for the analog circuits

Table 1.6.1 MB90610A Pin Descriptions (2)

Pin No.		Pin Name	Circuit Type	Function
QFP	SQFP			
38 to 41 43 to 46	36 to 39 41 to 44	P60 to P67	C (AD)	Open-drain output ports. This applies when port operation is specified in the analog input enable register.
		AN0 to AN7		Analog input pins for the A/D converter. This applies when A/D operation is specified in the analog input enable register.
47	45	P80	H (CMOS/H)	General-purpose I/O port. This applies in all cases.
		INT7		External interrupt request input pin. As the input operatescontinuously when the external interrupt is enabled, outputto the pin from other functions must be stopped unless doneintentionally.
		TIN0		Event input pin for reload timer 0. As the input operates continuously when the reload timer is set to input operation,output to the pin from other functions must be stopped unless done intentionally.
48	46	P81	D (CMOS/H)	General-purpose I/O port. This applies in all cases.
		TIN1		Event input pin for reload timer 1. As the input operates continuously when the reload timer is set to input operation,output to the pin from other functions must be stopped unless done intentionally.
49 to 51	47 to 49	MD0 to MD2	E (CMOS)	Input pins for specifying the operation mode. Connect directly to VCC or VSS.
52	50	HSTX	F (CMOS/H)	Hardware standby input pin
53 54	51 52	P82 to P83	D (CMOS/H)	General-purpose I/O ports. This applies when output is disabled for reload timers 0 and 1.
		TOT0 to TOT1		Output pins for reload timers 0 and 1. This applies when output is enabled for reload timers 0 and 1.
55	53	P84	D (CMOS/H)	General-purpose I/O port. This applies in all cases.
		SIN0		Serial data input pin for UART0. As the input operates continuously when UART0 is set to input operation, output to thepin from other functions must be stopped unless done intentionally.
56	54	P85	D (CMOS/H)	General-purpose I/O port. This applies when serial data output is disabled for UART0.
		SOT0		Serial data output pin for UART0. This applies when serial data output is enabled for UART0.
57	55	P86	D (CMOS/H)	General-purpose I/O port. This applies when the UART0 clockoutput is disabled.
		SCK0		Clock I/O pin for UART0. This applies when the UART0 clock output is enabled. As the input operates continuously when UART0 is set to input operation, output to the pin from otherfunctions must be stopped unless done intentionally.
58	56	P90	D (CMOS/H)	General-purpose I/O port. This applies in all cases.
		SIN1		Serial data input pin for UART1. As the input operates continuously when UART1 is set to input operation, output to thepin from other functions must be stopped unless done intentionally.
59	57	P91	D (CMOS/H)	General-purpose I/O port. This applies when serial data output is disabled for UART1.
		SOT1		Serial data output pin for UART1. This applies when serial data output is enabled for UART1.
60	58	P92	D (CMOS/H)	General-purpose I/O port. This applies when the UART1 clockoutput is disabled.
		SCK1		Clock I/O pin for UART1. This applies when the UART1 clock output is enabled. As the input operates continuously when UART1 is set to input operation, output to the pin from otherfunctions must be stopped unless done intentionally.

Table 1.6.1 MB90610A Pin Descriptions (3)

Pin No.		Pin Name	Circuit Type	Function
QFP	SQFP			
61	59	P93	D (CMOS/H)	General-purpose I/O port. This applies in all cases.
		SIN2		Serial data input pin for UART2. As the input operates continuously when UART2 is set to input operation, output to thepin from other functions must be stopped unless done intentionally.
62	60	P94	D (CMOS/H)	General-purpose I/O port. This applies when serial data output is disabled for UART2.
		SOT2		Serial data output pin for UART2. This applies when serial data output is enabled for UART2.
63	61	P95	D (CMOS/H)	General-purpose I/O port. This applies when the UART2 clockoutput is disabled.
		SCK2		Clock I/O pin for UART2. This applies when the UART2 clock output is enabled. As the input operates continuously when UART2 is set to input operation, output to the pin from otherfunctions must be stopped unless done intentionally.
64	62	CS0	J (CMOS)	Chip select pin for program ROM
65 to 71	63 to 69	PA1 to PA7	I (CMOS)	General-purpose I/O ports. This applies for pins with chip select output disabled by the chip select control register.
		CS1 to CS7		Output pins for the chip select function. This applies for pins with chip select output enabled by the chip select control register.
72	70	P50	I (CMOS)	General-purpose I/O port. This applies when CLK output is disabled.
		CLK		CLK output pin. This applies when CLK output is enabled.
73	71	P51	L (TTL)	General-purpose I/O port. This applies when the external ready function is disabled.
		RDY		Ready input pin. This applies when the external ready function is enabled.
74	72	P52	I (CMOS)	General-purpose I/O port. This applies when the hold function is disabled.
		HAKX		Hold acknowledge output pin. This applies when the hold function is enabled.
75	73	P53	L (TTL)	General-purpose I/O port. This applies when the hold function is disabled.
		HRQ		Hold request input pin. This applies when the hold functionis enabled.
76	74	P54	I (CMOS)	General-purpose I/O port. This applies in 8-bit external bus mode or when output is disabled for the WR pin.
		WRHX		Write strobe output pin for the upper 8 bits of the data bus. This applies in 16-bit external bus mode and when output is enabled for the WR pin.
77	75	RSTX	G (CMOS/H)	External reset request input pin
78	76	P55	I (CMOS)	General-purpose I/O port. This applies when output is disabled for the WR pin.
		WRLX		Write strobe output pin for the lower 8 bits of the data bus. This applies when output is enabled for the WR pin.
79	77	RDX	J (CMOS)	Read strobe output pin for the data bus
80	78	ALE	J (CMOS)	Address latch enable output pin
23 84	21 82	VCC	Power supply	Power supply for the digital circuits
11 42 81	9 40 79	VSS	Power supply	Ground level for the digital circuits

Table 1.6.3 I/O Circuit Configurations (1)

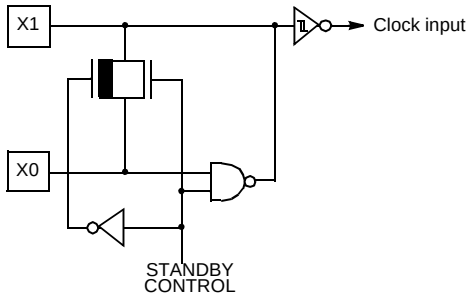
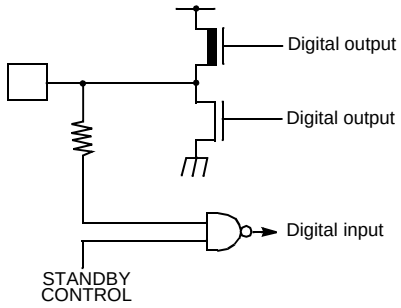
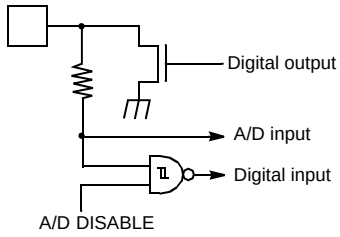
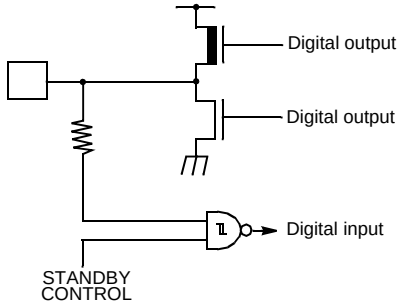
Type	Circuit Configuration	Remarks
A		• MAX. 3 MHz to 32 MHz • Oscillator feedback resistance: approximately 1 MΩ
B		• CMOS level I/O With standby control
C		• N-channel open drain output CMOS level hysteresis input With AD control
D		• CMOS level output CMOS level hysteresis input With standby control

Table 1.6.3 I/O Circuit Configurations (2)

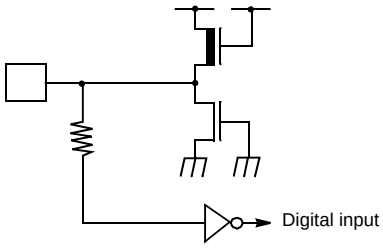
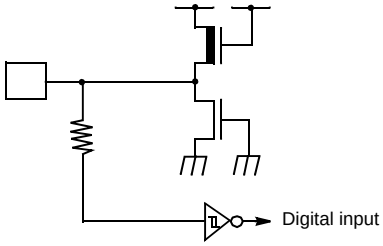
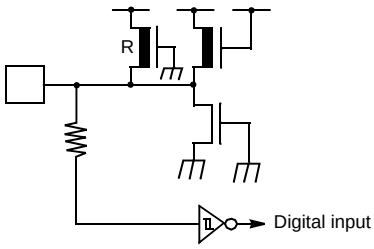
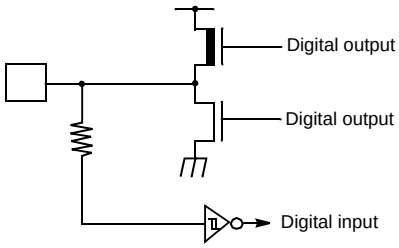
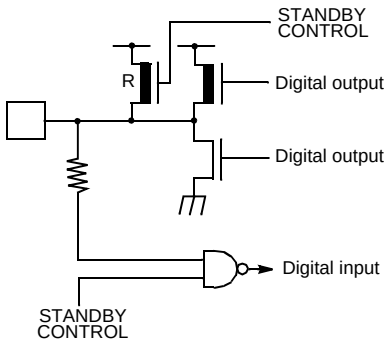
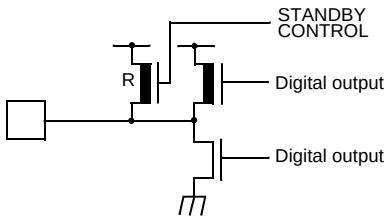
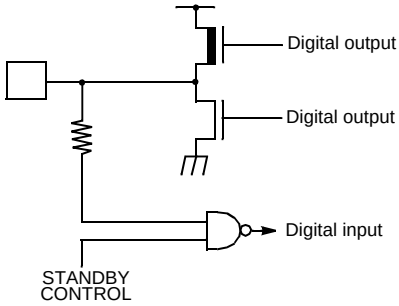
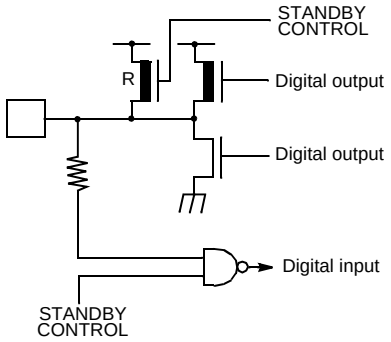
Type	Circuit Configuration	Remarks
E		• CMOS level input No standby control
F		• CMOS level hysteresis input No standby control
G		• CMOS level hysteresis input No standby control • With pull-up
H		• CMOS level output • CMOS level hysteresis input No standby control

Table 1.6.3 I/O Circuit Configurations (3)

Type	Circuit Configuration	Remarks
I		• CMOS level I/O • Pull-up resistor approximately 50 kΩ • Pin goes to high impedance during stop mode.
J		• CMOS level output • Pull-up resistor approximately 50 kΩ • Pin goes to high impedance during stop mode.
K		• CMOS level output • TTL level input With standby control
L		• CMOS level output • TTL level input • Pull-up resistor approximately 50 kΩ • Pin goes to high impedance during stop mode.

Note: For pins with pull-up resistors, the resistance is disconnected when the pin outputs the "L" level or when in the standby state.

(1) Preventing latch-up

Latch-up occurs in a CMOS IC if a voltage greater than VCC or less than VSS is applied to an input or output pin or if the voltage applied across VCC and VSS exceeds the rating. If latch-up occurs, the power supply current increases rapidly resulting in thermal damage to circuit elements. Therefore, ensure that maximum ratings are not exceeded in circuit operation.
For the same reason, also ensure that the analog supply voltage does not exceed the digital supply voltage.

(2) Connecting unused pins

Leaving unused input pins unconnected can cause misoperation. Always pull-up or pull-down unused pins.

(3) Cautions when using an external clock

Drive the X0 pin only when using an external clock. Figure 1.7.1 shows an example of how to use an external clock.

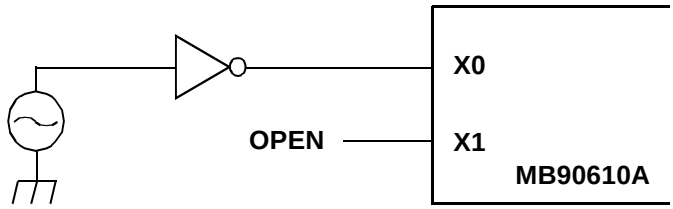


Fig. 1.7.1 Example of Using an External Clock

(4) Power supply pins

When multiple VCC and VSS pins are provided, connect all VCC and VSS pins to supply or ground externally. Although pins at the same potential are connected together in the internal device design so as to prevent misoperation such as latch-up, connecting all VCC and VSS pins appropriately minimizes unwanted radiation, prevents misoperation of strobe signals due to increases in the ground level, and keeps the overall output current rating.

Also, take care to connect VCC and VSS to a low impedance current source.
Connection of a bypass capacitor (a ceramic capacitor of approximately 0.1 µF connected close to the device) between VCC and VSS is recommended.

(5) Crystal oscillator circuit

Noise in the vicinity of the X0 and X1 pins can be a cause of device misoperation. Place X0, X1, the crystal oscillator (or ceramic oscillator), and the bypass capacitor to ground as close together as possible. Also, design the circuit board so that the wiring for the crystal oscillator circuit does not cross other wiring.

A printed circuit board design that surrounds the X0 and X1 pins with ground provides for stable operation and is strongly recommended.

(6) A/D converter power supply and the turn-on sequence for analog inputs

Do not apply current to the A/D converter power supply (AV_{CC}, AV_{R+}, AV_{R-}) or analog inputs (AN0 to

AN7) until the digital power supply (V_{CC}) is turned on.

When turning the device off, turn off the digital power supply after cutting the A/D converter power supply and analog inputs.

When turning the power on or off, ensure that AV_{R+} does not exceed AV_{CC}.

Chapter 2: Hardware

2.1 CPU

2.1.1 Memory Space

■ Outline of the CPU memory space

The program, data, and I/O managed by the F²MC-16L CPU are all located in the CPU's 16MB memory space. The CPU accesses each resource by setting the corresponding address on the 24-bit address bus (Figure 2.1.1).

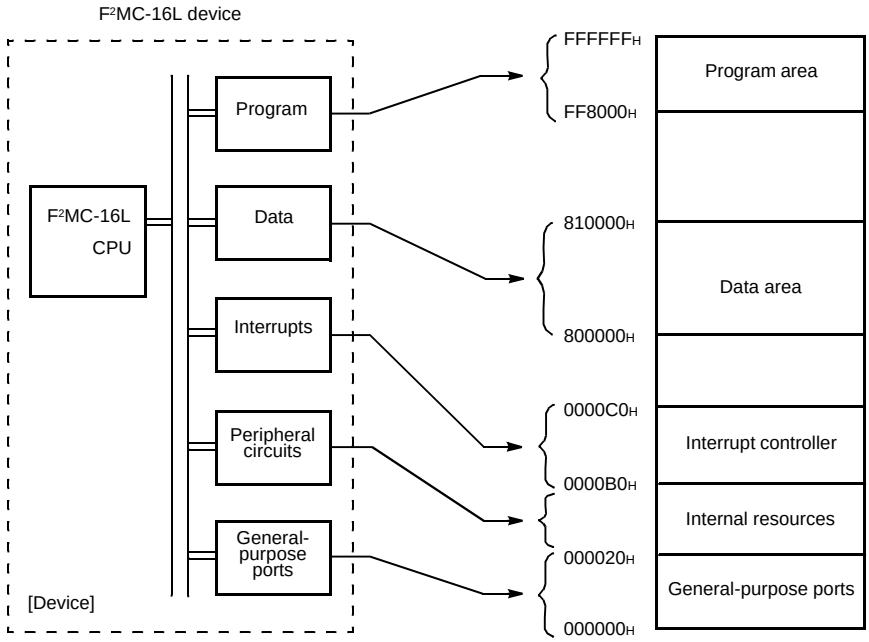


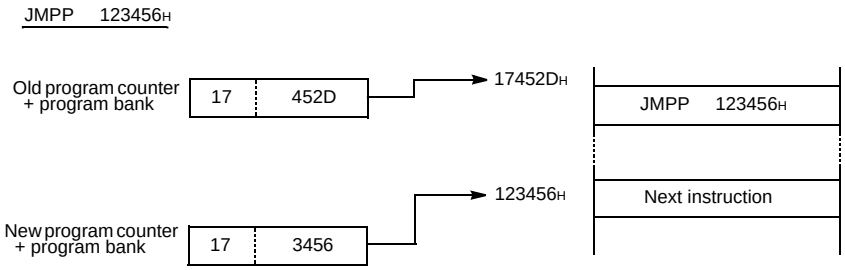
Fig. 2.1.1 Example of the Relationship Between the F²MC-16L System and Memory Map

■ Address generation modes

Address generation in the F²MC-16L can be broadly divided into two modes: linear addressing and bank addressing. In linear addressing, the instruction specifies the entire 24-bit address. In bank addressing, the upper 8 bits of the address are set in a bank register based on the application and the instruction specifies the lower 16 bits of the address.

Linear addressing can be further divided into two types. In one method, the operand specifies the 24-bit address directly. In the other method, the lower 24 bits of a 32-bit general-purpose register are used as the address. (Figure 2.1.2)

Example 1: Linear addressing with a 24-bit operand specified



Example 2: Linear addressing using 32-bit register indirect addressing

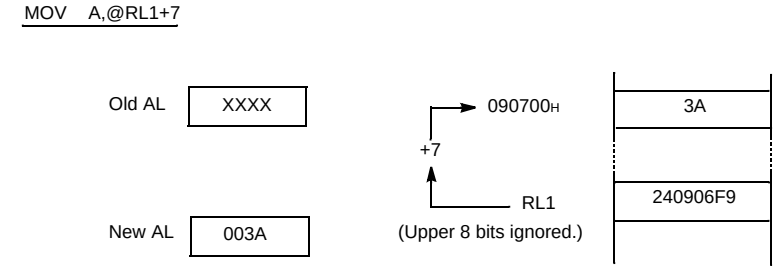


Fig. 2.1.2 Example of Linear Address Generation

■ Bank addressing modes

Bank addressing splits the 16MB address space into 256 banks of 64KB. The bank registers specify the bank address. Five types of bank register are provided. Table 2.1.1 lists the memory space accessed and the main use for each bank register.

Table 2.1.1 Memory Space Accessed by Each Bank Register

Bank Register Name	Memory Space Name	Main Application	Initial Value After Reset
Program bank register (PCB)	Program (PC) space	Stores instruction code, vector tables, and immediate data	FF$_H$
Data bank register (DTB)	Data (DT) space	Stores readable and writable data. Accesses control registers and data registers for internal and external peripherals.	00$_H$
User stack bank register (USB)	Stack (SP) space	Area used for stack access such as by PUSH and POP instructions or register saving at interrupts. SSB is used when S=1 in CCR. USB is used when S=0 in CCR.	00$_H$
System stack bank register (SSB)			00$_H$
Additional bank register (ADB)	Additional (AD) space	Stores data such as data that is too large for the data (DT) space.	00$_H$

After a reset, the DT, SP, and AD spaces are allocated to bank 00 (000000$_H$ to 00FFFF$_H$) and the PC space is allocated to bank FF (FF0000$_H$ to FFFFFF$_H$).

To improve instruction code efficiency, instructions have a default space for each addressing mode. Table 2.1.2 lists the defaults. Prefix codes can be prefixed to instructions to specify a space other than the default for the addressing mode used. The system accesses the space corresponding to the prefix code.

Table 2.1.2 Default Memory Spaces

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

Figure 2.1.3 shows an example of the division of memory space into banks and each bank register.

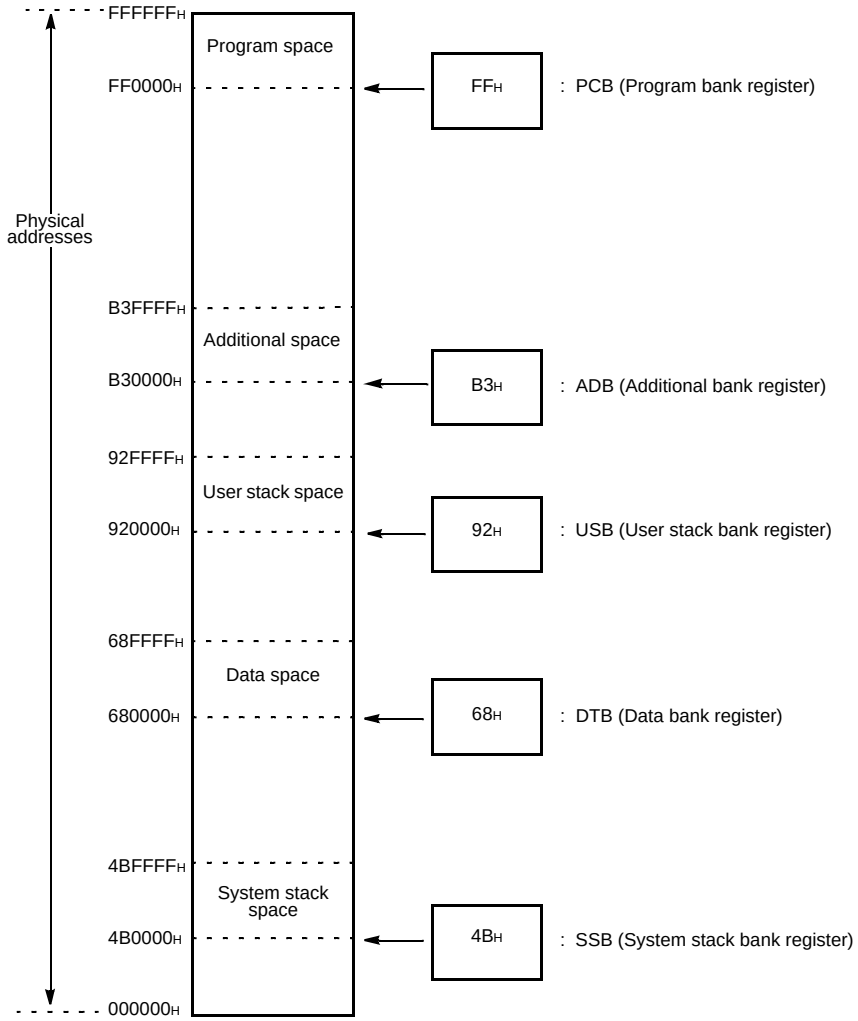


Fig. 2.1.3 Example of Physical Addresses of Each Memory Space

■ Memory space layout for multi-byte data

Figure 2.1.4 shows the data configuration of multi-byte data in memory. The lower 8 bits are placed at location n and subsequent bytes placed at locations n+1, n+2, n+3, etc.

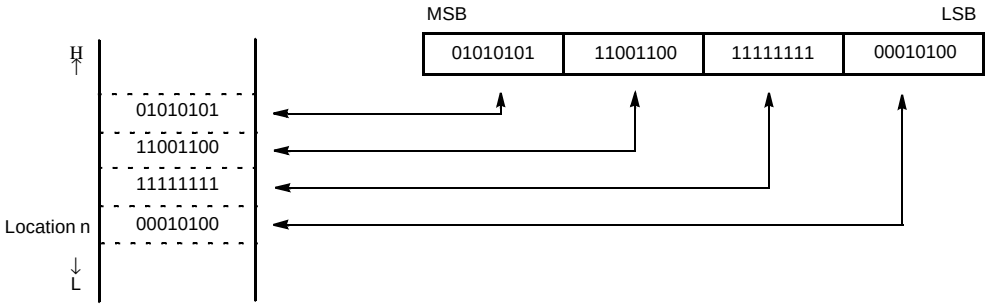


Fig. 2.1.4 Example of Memory Layout for Multi-Byte Data

Memory is written to from the lowest address. Therefore, for 32-bit data, the lower 16 bits are transferred first, followed by the upper 16 bits.

If a reset signal is input immediately after writing the lower data bits, writing the upper data bits may not occur.

■ Accessing multi-byte data

All access occurs within a bank. Therefore, for instructions that access multi-byte data, the next address after location FFFFH is location 0000H in the same bank. Figure 2.1.5 shows an execution example for an instruction that accesses multi-byte data.

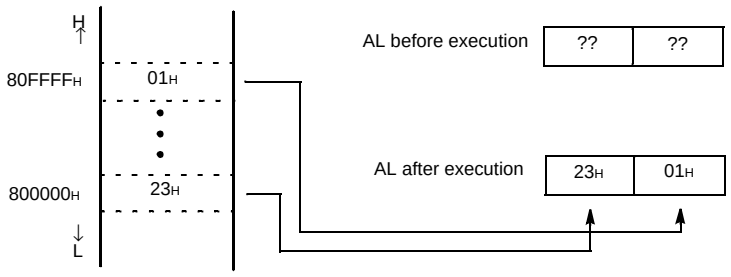


Fig. 2.1.5 Execution of MOVW A,080FFFFH

2.1.2 Registers

The F²MC-16L registers can be broadly divided into two categories: dedicated registers located in the CPU and general-purpose registers located in internal RAM. Dedicated registers exist as specific hardware in the CPU and their use is limited by the CPU architecture. In contrast, general-purpose registers are located in RAM in the CPU's address space. Like special registers, general-purpose registers can be accessed without specifying an address. However, general-purpose registers can also be used as specified by the user, in the same way as standard memory. Figure 2.1.6 shows the layout of the dedicated and general-purpose registers in the device.

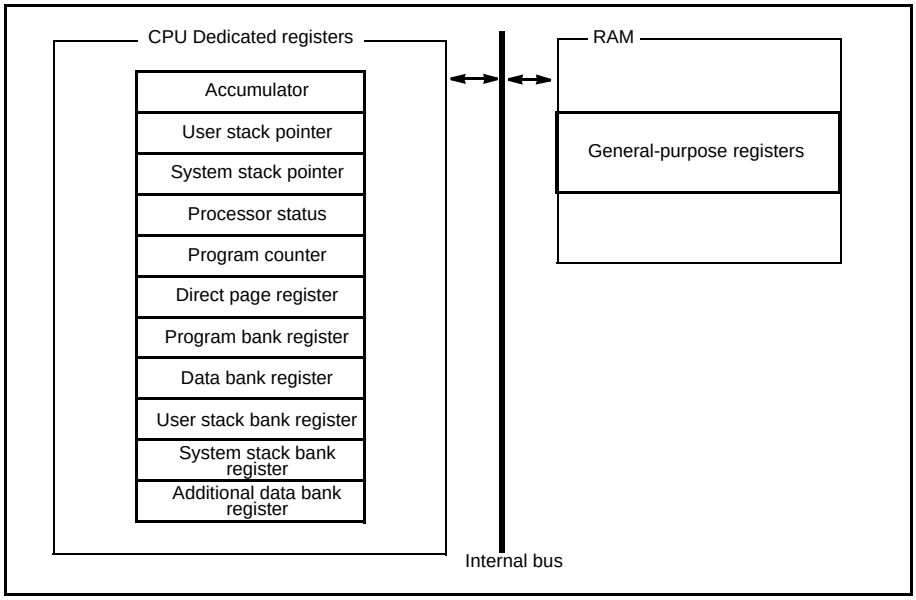


Fig. 2.1.6 Special and General-Purpose Registers

■ Dedicated registers

Table 2.1.3 lists the eleven dedicated registers in the F²MC-16L.

Table 2.1.3 Dedicated Registers

Structure	Register Name	Function
AH | AL	Accumulator	2 × 16-bit registers used to save operation results and similar. Can be combined as a single 32-bit register.
USP	User stack pointer	16-bit pointer that specifies the user stack area
SSP	System stack pointer	16-bit pointer that specifies the system stack area
PS	Processor status	16-bit register that indicates the system status
PC	Program counter	16-bit register that stores the address containing the program
DPR	Direct page register	8-bit register that specifies the direct page
PCB	Program bank register	8-bit register that specifies the program space
DTB	Data bank register	8-bit register that specifies the data space
USB	User stack bank register	8-bit register that specifies the user stack space
SSB	System stack bank register	8-bit register that specifies the system stack space
ADB	Additional data bank register	8-bit register that specifies the additional space

■ Accumulator (A)

The accumulator consists of two 16-bit operation registers: AH and AL. The accumulator is used for temporary storage of operation results and data moves. AH and AL can be combined for 32-bit data processing. For 16-bit word processing or 8-bit byte processing, the AL register only is used. (See Figures 2.1.7 and 2.1.8.)

Operations can be performed on accumulator data and memory or register (Ri, RWi, or RLi) data. Like the F²MC-8, when word-length or shorter data is transferred to the F²MC-16's AL register, the previous content of AL is automatically transferred to AH (the data retention function). The data keep function and AL-AH operations increase the processing efficiency of the device. (Figure 2.1.8)

MOVL A,@RW1+6 (This instruction performs a long-word read from the location specified by adding an 8-bit offset value to RW1 and places the data in the accumulator.)

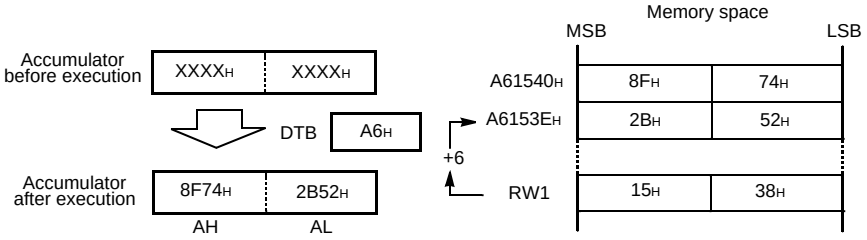


Fig. 2.1.7 32-Bit Data Move Example

MOVW A,@RW1+6 (This instruction performs a single-word read from the location specified by adding an 8-bit offset value to RW1 and places the data in the accumulator.)

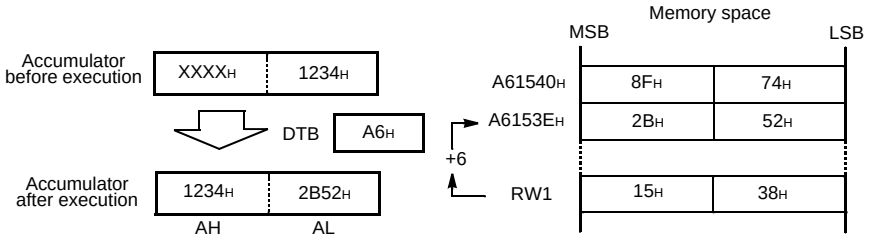


Fig. 2.1.8 AL-AH Move Example

When moving byte-length or shorter data to AL, the data placed in AL is sign extended or zero extended to a length of 16 bits. Data in AL can be treated as word-length or byte-length. When the CPU executes a byte-length arithmetic instruction on AL, the operation ignores the upper 8 bits of AL and sets the upper 8 bits of the result to zero. (See Figures 2.1.9 and 2.1.10.)

The accumulator is not initialized by a reset. The value of the accumulator after a reset is undefined.

MOV A,3000H (This instruction zero extends the data at location 3000H and places the result in AL.)

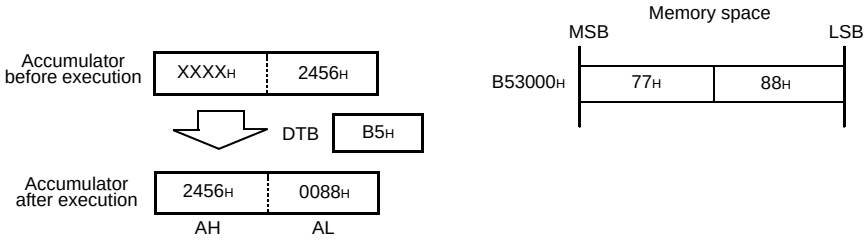


Fig. 2.1.9 Zero Extend Execution Example

MOVX A,3000H (This instruction sign extends the data at location 3000H and places the result in AL.)

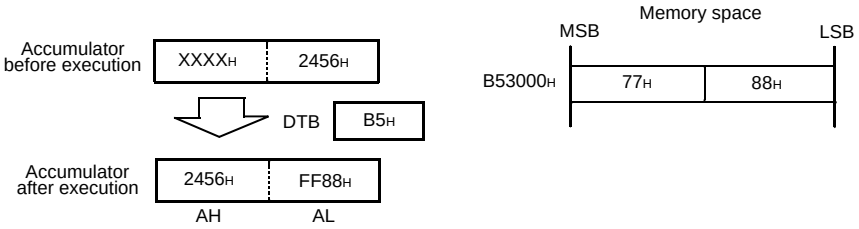


Fig. 2.1.10 Sign Extend Execution Example

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

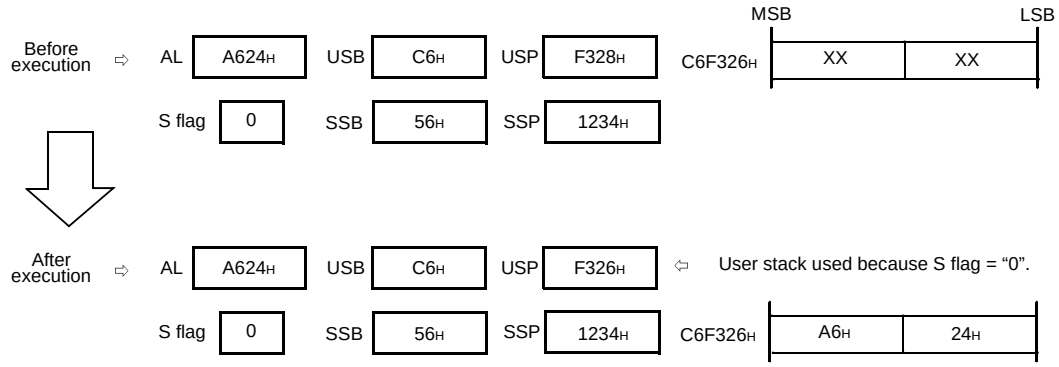
The USP and SSP are 16-bit registers that specify the memory address for saving and restoring data at PUSH/POP instruction or subroutine execution. Stack instructions operate in the same way on both USP and SSP. The instructions use USP if the S flag in the processor status (PS) register is set to "0" and SSP if the S flag is set to "1" (see Figure 2.1.11).

The S flag is set to "1" when an interrupt is received. Therefore, when an interrupt occurs, the processor always saves registers to the memory specified by SSP. Normally, stack processing in interrupt routines uses SSP and stack processing other than in interrupt routines uses USP. If separate stack spaces are not necessary, use SSP only.

Always set an even numbered address to the stack pointer. Setting an odd numbered address splits word access in two and reduces efficiency.

SSB specifies the upper 8 bits of the stack address for SSP and USB specifies the upper 8 bits for USP. USP and SSP are not initialized by a reset. The values of USP and SSP after a reset are undefined.

Example 1: Executing PUSHW A when the S flag is "0".



Example 2: Executing PUSHW A when the S flag is "1".

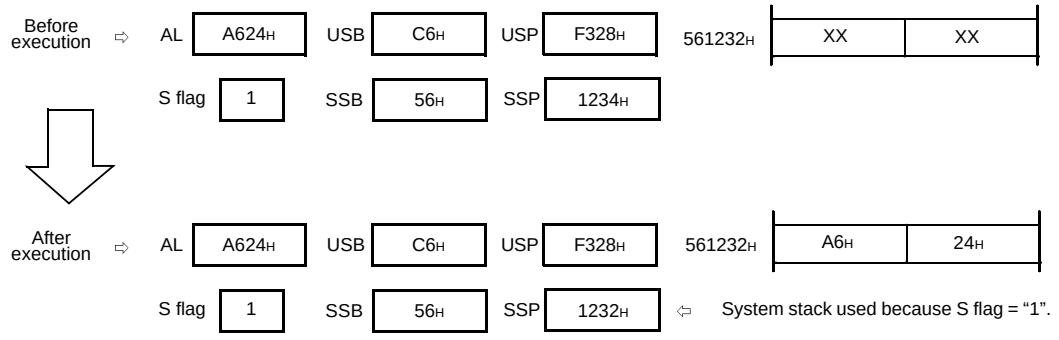


Fig. 2.1.11 Stack Manipulation Instructions and Stack Pointers

■ Processor status (PS)

The processor status register consists of control bits that control the CPU operation and status bits that indicate the CPU status. Figure 2.1.12 shows the structure of PS. The upper byte contains the register bank pointer (RP), which indicates the top address of the register bank, and the interrupt level mask register (ILM). The lower byte contains the condition code register (CCR). The CCR consists of flags which are set to "0" or "1" by instruction execution results, interrupt generation, or other events.

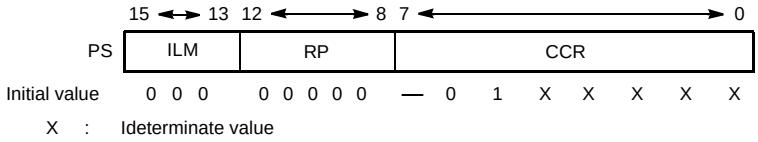


Fig. 2.1.12 PS Structure

(1) Condition code register (CCR)

Figure 2.1.13 shows the structure of the condition code register.

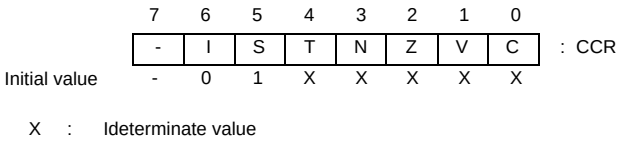


Fig. 2.1.13 Condition Code Register Structure

I: Interrupt enable flag	For all interrupts other than software interrupts, setting I to "1" enables the interrupts and setting I to "0" masks the interrupts. Cleared by a reset.
S: Stack flag	Stack operations use USP if the S flag is set to "0" and SSP if the S flag is set to "1". Set by a reset or on receiving an interrupt.
T: Sticky bit flag	Set to "1" if the data shifted out of carry during a logical or arithmetic right shift instruction contains one or more "1"s. Otherwise, set to "0". Also set to "0" if the shift amount is zero.
N: Negative flag	Set if the MSB of an operation result is "1", cleared if the MSB is "0".
Z: Zero flag	Set if an operation result is all "0"s, cleared otherwise.
V: Overflow flag	Set if an overflow occurs for a signed value as the result of the execution of an operation. Cleared if no overflow occurs.
C: Carry flag	Set if a carry-up or carry-down occurs for the MSB as the result of the execution of an operation. Cleared if no carry occurs.

(2) Register bank pointer (RP)

The RP register specifies the relationship between the F²MC-16L's general-purpose registers and the addresses in internal RAM where the registers are located. The top memory address of the register bank currently in use is specified by the conversion formula: [000180H + (RP)*10H] (Figure 2.1.14). RP consists of 5 bits and can be set in the range 00H to 1FH. This allows register banks to be located in memory between 000180H and 00037FH. However, addresses in this range can only be used as general-purpose registers if the address is in internal RAM. RP is initialized to 00H by a reset.

RP can be set by an 8-bit immediate move instruction but only the lower 5 bits are used.

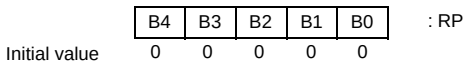


Fig. 2.1.14 Register Bank Pointer

(3) Interrupt Level Mask Register (ILM)

The ILM consists of 3 bits and specifies the interrupt mask level for the CPU. The CPU only receives interrupts with a higher priority level than the level specified in ILM. Zero is the highest interrupt level and seven is the lowest interrupt level (see Table 2.1.4). Therefore, to receive an interrupt, the level value of the interrupt request must be less than the value set in ILM. When an interrupt is received, the interrupt level value is set in ILM. This disables the subsequent reception of interrupts of the same or lower priority. ILM is initialized to all zeros by a reset. Instruction allows 8-bit immediate value transfer to ILM but only the upper 3 bits are actually used.

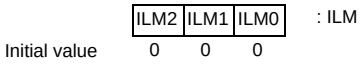


Fig. 2.1.15 Interrupt Level Register

Table 2.1.4 Priority Levels for the Interrupt Level Mask Register (ILM)

ILM2	ILM1	ILM0	Level	Allowed Interrupt Levels
0	0	0	0	All interrupts prohibited
0	0	1	1	0 only
0	1	0	2	Level 1 or less
0	1	1	3	Level 2 or less
1	0	0	4	Level 3 or less
1	0	1	5	Level 4 or less
1	1	0	6	Level 5 or less
1	1	1	7	Level 6 or less

■ Program counter (PC)

The PC is a 16-bit counter. The PC specifies the lower 16 bits of the memory address of the instruction code to be executed by the CPU. PCB specifies the upper 8 bits of the address. Operations that update the content of the PC include conditional branch instructions, subroutine call instructions, interrupts, and resets.

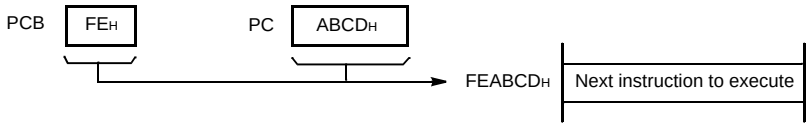


Fig. 2.1.16 Program Counter

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

As shown in Figure 2.1.17, the DPR specifies addr8 to addr15 of the operand for direct addressing instructions. DPR is an 8-bit register and is initialized to 01H by a reset. The DPR can be read from or written to by instructions.

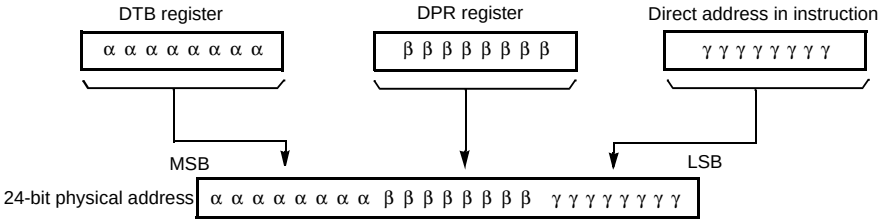


Fig. 2.1.17 Physical Address Generation Using Direct Addressing

■ Program counter bank register (PCB) <Initial value: value from 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>

The bank registers specify the memory banks for the PC space, DT space, SP space (user), SP space (system), and AD space respectively. All bank registers consist of 8 bits. A reset initializes PCB to FFH and other bank registers to 00H. The bank registers other than PCB permit reading and writing. The PCB permits reading but not writing. The PCB is written to on execution of branch instructions (JMPP, CALLP, RETP, and RETI) that operate in the total (16MB) memory space, on execution of a software interrupt instruction, and when exceptions or hardware interrupts occur. See 2.1.1 "Memory Space" for details on the operation of each register.

■ General-purpose registers

The general-purpose registers of the F²MC-16L are located in RAM at 000180H to 00037FH in the memory map. The register bank pointer (RP) specifies the section of memory that is currently used as the register bank. Each bank contains the following three register types. The registers are not independent. Figure 2.1.18 shows the relationship between the registers.

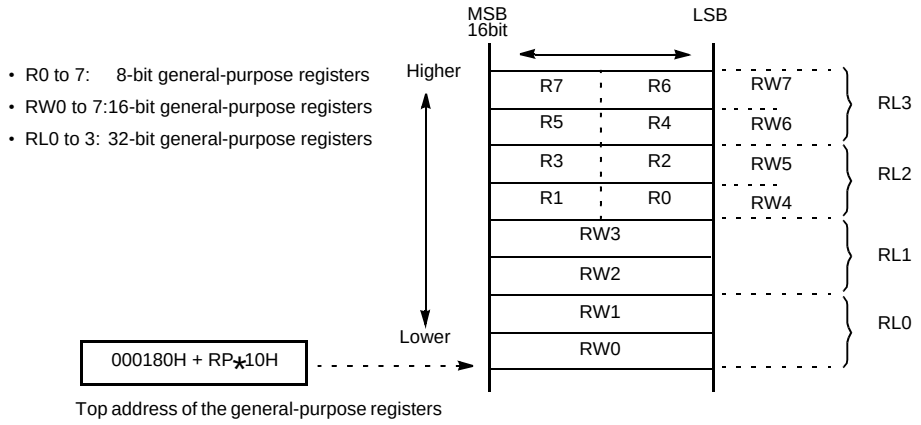


Fig. 2.1.18 General-Purpose Registers

■ Register bank

The register bank consists of eight 16-bit registers: byte registers R0 to R7, word registers RW0 to RW7, and long word registers RL0 to RL3. The registers can be used as general-purpose registers in various operations and as pointers in various instructions. RL0 to RL3 can also be used as linear pointers to directly access the entire memory space. Table 2.1.5 lists the function of each register.

As for standard RAM, the register contents are not initialized by a reset and the registers maintain the values that they had prior to the reset. However, the register contents are indeterminate at power on.

Table 2.1.5 Register Functions

R0 to R7	Used as instruction operands. **Note:** R0 is also used as the barrel shift counter and normalize instruction counter.
RW0 to RW7	Used as pointers. Used as instruction operands. **Note:** RW0 is also used as the string instruction counter.
RL0 to RL3	Used as long pointers. Used as instruction operands.

2.1.3 Prefix Codes

Placing a prefix code in front of an instruction modifies the operation of the instruction. Three types of prefix codes are available: bank select prefixes, common register bank prefixes, and flag change inhibit prefixes.

■ Bank select prefix

The memory space used for data access is determined by the addressing mode. Placing a bank select prefix in front of an instruction selects a specific memory space for data access by the instruction, irrespective of the addressing mode. Table 2.1.6 lists the memory space selected by each bank select prefix.

Table 2.1.6 Bank Select Prefix

Bank Select Prefix	Memory Space
PCB	PC space
DTB	Data space
ADB	AD space
SPB	If the S flag in CCR is "0", selects the user stack space. If the S flag is "1", selects the system stack space.

However, note that the instructions listed in Table 2.1.7 ignore the bank select prefix. Also, for the instructions listed in Table 2.1.8, the effect of the bank select prefix is passed on to the next instruction.

Table 2.1.7 Instructions that Ignore the Bank Select Prefix

Instruction Type	Instruction	Effect of the Bank Select Prefix
String instructions	MOVS MOVSW SCEQ SCWEQ FILS FILSW	Uses the bank register specified in the operand, whether or not a prefix is present.
Stack manipulation instructions	PUSHW POPW	Uses USB (if the S flag = 0) or SSB (if the S flag = 1), whether or not a prefix is present.
I/O access instructions	MOV A,io MOVX A,io MOVW A,io MOV io,A MOVW io,A MOV io,#imm8 MOVW io,#imm16 MOVB A,io:bp MOVB io:bp,A SETB io:bp CLRB io:bp BBC io:bp,rel BBS io:bp,rel WBTC io:bp WBTS io:bp	Accesses the memory space 000000H to 0000FFH, whether or not a prefix is present.
Interrupt return instruction	RETI	Uses SSB, whether or not a prefix is present.

Table 2.1.8 Instructions for Which the Effect of the Bank Select Prefix is Passed on to the Next Instruction

Instruction Type	Instruction
Flag modify instructions	AND CCR,#imm8 OR CCR,#imm8
PS restore instruction	POPW PS
ILM set instruction	MOV ILM,#imm8

■ When multiple prefix codes are specified

When multiple conflicting prefix codes are specified, the final code specified is used.

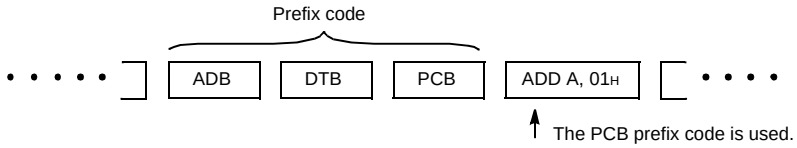


Fig. 2.1.19 Multiple Prefix Codes

■ Common register bank prefix (CMR)

To simplify data exchange between tasks, a predetermined common register bank is required that can be accessed by a comparatively simple procedure unaffected by the value of RP at the time. Placing CMR in front of an instruction that accesses a register bank changes the accessed register bank to the common bank located at 000180H to 00018FH (the bank selected when RP = 0), irrespective of the current value of RP. However, care is required with the instructions listed in Table 2.1.9.

Table 2.1.9 Instructions Requiring Care When Using the Common Register Bank Prefix

Instruction Type	Instruction	Explanation
String instructions	MOVS MOVSW SCEQ SCWEQ FILS FILSW	Do not use the CMR prefix with string instructions.
Flag modify instructions	AND CCR,#imm8 OR CCR,#imm8	The effect of the prefix is passed on to the next instruction.
PS restore instruction	POPW PS	The effect of the prefix is passed on to the next instruction.
ILM set instruction	MOV ILM,#imm8	The effect of the prefix is passed on to the next instruction.

■ Flag change inhibit prefix

Use the flag change inhibit prefix code (NCC) to inhibit unwanted changes to the flags. Placing NCC in front of an instruction prevents instruction execution from changing the flags. The prefix inhibits changes to the T, N, Z, V, and C flags.

However, care is required with the instructions listed in Table 2.1.10.

Table 2.1.10 Instructions Requiring Care When Using the Flag Change Inhibit Prefix

Instruction Type	Instructions	Explanation
String instructions	MOVS MOVSW SCEQ SCWEQ FILS FILSW	Do not use the NCC prefix with string instructions.
Flag modify instructions	AND CCR,#imm8 OR CCR,#imm8	The instruction changes CCR as usual, whether or not a prefix is present. The effect of the prefix is passed on to the next instruction.
PS restore instruction	POPW PS	The instruction changes CCR as usual, whether or not a prefix is present^\The effect of the prefix is passed on to the next instruction.
ILM set instruction	MOV ILM,#imm8	The effect of the prefix is passed on to the next instruction.
Interrupt instructions Interrupt return instructions	INT #vct8 INT9 INT addr16 INTP addr24 RETI	The instruction changes CCR as usual, whether or not a prefix is present.
Context switch instruction	JCTX @A	The instruction changes CCR as usual, whether or not a prefix is present.

■ Interrupt inhibiting instructions

Hardware interrupt requests are not detected and interrupt requests are ignored for the ten types of instruction listed in Table 2.1.11.

Table 2.1.11 Hardware Interrupt Inhibiting Instructions

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

Therefore, if a valid hardware interrupt request occurs during execution of one of these instructions, interrupt processing does not start until after the execution of the next instruction of a type other than those listed above. Figure 2.1.20 shows an example.

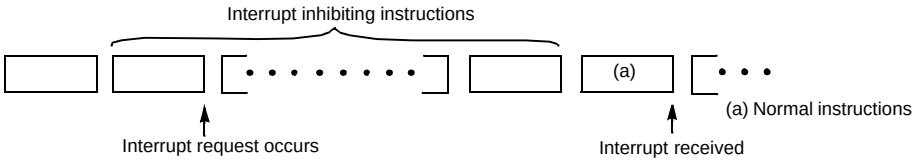


Fig. 2.1.20 Interrupt Inhibiting Instructions

■ Restrictions for interrupt inhibiting instructions and prefix instructions

If a prefix code is placed in front of an interrupt inhibiting instruction, the effect of the prefix code is passed on to the next instruction that is not an interrupt inhibiting instruction. Figure 2.1.21 shows an example.

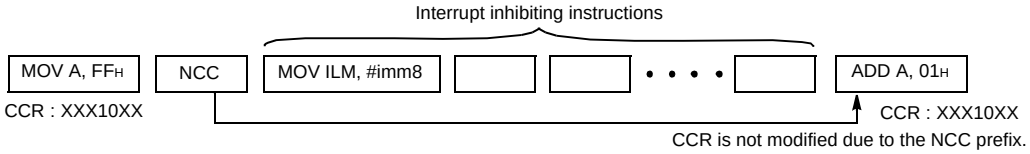


Fig. 2.1.21 Interrupt Inhibiting Instructions and Prefix Codes

2.1.4 Interrupts, Extended Intelligent I/O Service, and Exceptions

The F²MC-16L has functions that halt the currently executing processing and pass control to another defined program when an event or similar occurs. The functions can be divided into the following four types.

● Hardware interrupt	Interrupt processing triggered by an event occurring in an internal resource circuit.
● Software interrupt	Interrupt processing triggered by a software event instruction.
● Extended intelligent I/O service (EI²OS)	Data transfer processing triggered by an event occurring in an internal resource circuit.
● Exception	Interrupt processing triggered by an exception.

■ Hardware interrupts

(1) Outline

The hardware interrupt function temporarily interrupts the program currently executing in the CPU in response to an interrupt request signal from an internal resource circuit. Control is passed to a user-defined interrupt handling program.

To activate a hardware interrupt, the hardware compares the interrupt level of the interrupt request with the interrupt level mask register (ILM) in the PS register of the CPU and checks the value of the I flag in the PS register. The hardware interrupt activates if the conditions are valid. The CPU performs the following processing when a hardware interrupt occurs.

- The A, DPR, ADB, DTB, PCB, PC, and PS registers in the CPU are saved on the system stack.
- The current interrupt level is stored in ILM in the PS register.
- Execution branches to the corresponding interrupt vector.

(2) Configuration

The mechanisms relating to hardware interrupts can be divided into the following three groups.

- Internal resource circuits — Interrupt enable bit and interrupt request bit: Controls interrupt requests from internal resources.
- Interrupt controller — ICR: Assigns interrupt levels and prioritizes simultaneous interrupts.
- CPU — I, ILM: Compares the level of the interrupt request with the current level. Stores the interrupt enable status.

  Microcode: Executes the interrupt processing steps.

Each mechanism is represented by the content of its respective registers: internal resource control registers for internal resource circuits, the ICR for the interrupt controller, and the CCR for the CPU. When using hardware interrupts, the program must first set values to these three register types. See "Interrupt Control Register (ICR)" in the "Extended Intelligent I/O Service" section for details on the ICR.

The interrupt vector table referenced during interrupt processing is located in the memory area FFFC00H to FFFFFFH. The area is shared with software interrupts. Table 2.1.12 lists the allocation of interrupt numbers and interrupt vectors.

Table 2.1.12 Allocation of Interrupt Numbers and Interrupt Vectors

Software Interrupt Instruction	Vector Address L	Vector Address M	Vector Address H	Mode Register	Interrupt Number	Hardware Interrupt
INT 0	FFFFFCH	FFFFFDH	FFFFFEH	Unused	#0	None
⋮	⋮	⋮	⋮	⋮	⋮	⋮
INT 7	FFFFE0H	FFFFE1H	FFFFE2H	Unused	#7	None
INT 8	FFFFDCH	FFFFDDH	FFFFDEH	FFFFDF	#8	(Reset vector)
INT 9	FFFFD8H	FFFFD9H	FFFFDAH	Unused	#9	None
INT 10	FFFFD4H	FFFFD5H	FFFFD6H	Unused	#10	<Exception>
INT 11	FFFFD0H	FFFFD1H	FFFFD2H	Unused	#11	Hardware interrupt #0
INT 12	FFFFCCH	FFFFCDH	FFFFCEH	Unused	#12	Hardware interrupt #1
INT 13	FFFFC8H	FFFFC9H	FFFFCAH	Unused	#13	Hardware interrupt #2
INT 14	FFFFC4H	FFFFC5H	FFFFC6H	Unused	#14	Hardware interrupt #3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
INT 254	FFFC04H	FFFC05H	FFFC06H	Unused	#254	Free
INT 255	FFFC00H	FFFC01H	FFFC02H	Unused	#255	<Stack fault>

(3) Operation

Internal resources with a hardware interrupt function have an "interrupt request flag" that indicates whether an interrupt request is present and an "interrupt enable flag" that selects whether or not the resource circuit can send interrupt requests to the CPU. The interrupt request flag is set by specific events in the internal resource circuit. If the interrupt enable flag is set to "enabled", the internal resource circuit passes the interrupt request to the interrupt controller.

The interrupt controller compares the interrupt level (IL) in the ICR for each simultaneously occurring interrupt request, selects the request with the highest level (the lowest IL value), and notifies the CPU. If more than one request occurs with the same level, the request with the lowest interrupt number has priority. See Section 2.2.3 "Interrupt Vector Allocation" for details on the relationship between each interrupt request and ICR.

The CPU compares the received interrupt level with ILM in the PS register. If the interrupt level (IL) is less than ILM and the I flag in the PS register is "1", the CPU activates the interrupt processing microcode after completing execution of the current instruction. The interrupt processing microcode first checks that the ISE bit in the interrupt controller's ICR register is set to "0" (indicating an interrupt), then activates the main body of interrupt processing.

After saving the A, DPR, ADB, DTB, PCB, PC, and PS registers (12 bytes) to the memory area specified by SSB and SSP, interrupt processing reads the 3-byte interrupt vector and loads the vector to PC and PCB, changes ILM in the PS register to the level of the received interrupt request, sets the S flag to "1", then executes branch processing.
As a result, the next executed instruction is the user-defined interrupt processing program.

Figure 2.1.22 shows the flow of processing from the generation of a hardware interrupt until the interrupt request is completed by the interrupt processing program. Figure 2.1.23 shows the operation flow for a hardware interrupt.

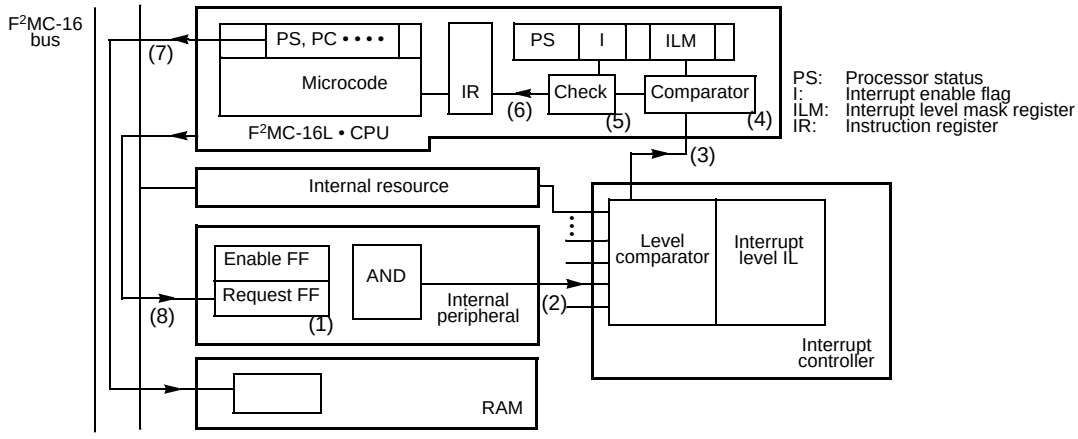


Fig. 2.1.22 Steps from Generation to Release of a Hardware Interrupt

(1) Interrupt source generated in the internal resource.
(2) If interrupts are enabled in the internal resource by the interrupt enable bit, the interrupt request is passed from the resource to the interrupt controller.
(3) The interrupt controller receives the interrupt request, evaluates the priorities of any simultaneous interrupt requests, then passes the interrupt level of the highest priority interrupt to the CPU.
(4) The CPU compares the level of the interrupt request from the interrupt controller with ILM in the processor status register.
(5) If the result of comparison is that the priority is higher than for the current interrupt processing level, the CPU then checks the I flag in the processor status register.
(6) If checking the I flag in step 5 indicates that interrupts are enabled, the CPU waits until the current instruction completes executing, then sets the request level in ILM.
(7) The CPU saves the registers, then passes control by branching to the interrupt processing routine.
(8) The interrupt request completes when the user's interrupt processing routine software clears the interrupt source generated in step 1.

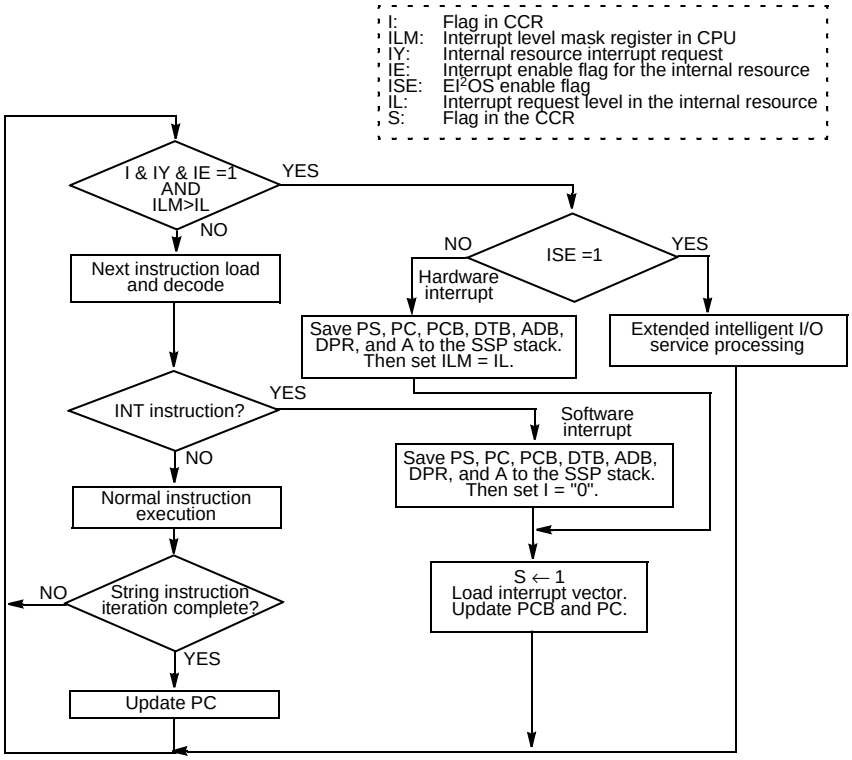


Fig. 2.1.23 Interrupt Operation Flow

(4) Example procedure for using hardware interrupts

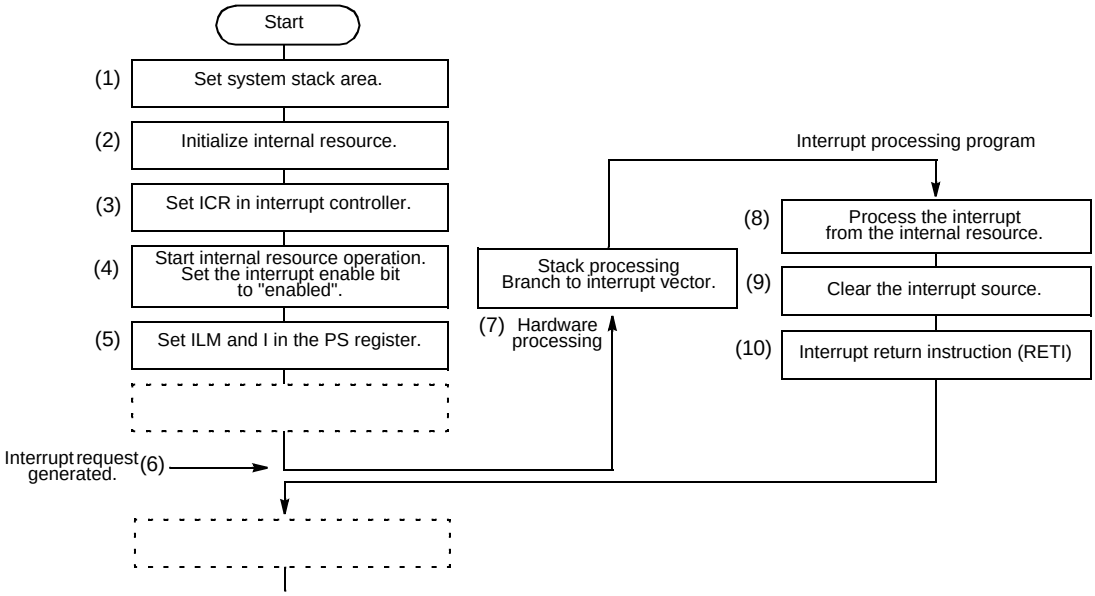


Fig. 2.1.24 Example Procedure for Using Hardware Interrupts

(1) Set the system stack area.
(2) Initialize the internal resource that can generate interrupt requests.
(3) Set ICR in the interrupt controller.
(4) Set the internal resource to the operating state and set the interrupt enable bit to "enabled".
(5) Set the ILM and I flag in the CPU to enable the reception of interrupts.
(6) Generation of an interrupt in the internal resource triggers a hardware interrupt request.
(7) The interrupt processing hardware saves the registers and branches to the interrupt processing program.
(8) The interrupt processing program performs the necessary processing for the internal resource that generated the interrupt.
(9) Release the interrupt request from the internal resource circuit.
(10) Execute the interrupt return instruction and return to the previous program.

(5) Hardware interrupt requests during a write to an internal resource area

Hardware interrupt requests cannot be received during a write to an internal resource area. This is to prevent misoperation of CPU interrupt processing due to an interrupt request occurring during an update of the interrupt control register in an internal resource. The internal resource area refers to the area allocated for the control and data registers of internal resources (not the I/O addressing area between 000000H and 0000FFH).

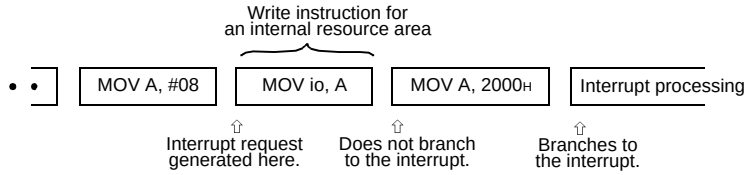


Fig. 2.1.25 Hardware Interrupt Request During a Write to an Internal Resource Area

(6) Interrupt inhibiting instructions

Some F²MC-16L instructions do not detect hardware interrupt requests. These are called interrupt inhibiting instructions. See Table 2.1.11 for details.

(7) Multiple interrupts

The F²MC-16L CPU supports multiple interrupts. If, during the processing of an interrupt, a second interrupt with a higher priority level occurs, control passes to the second interrupt on completion of the currently executing instruction. Control returns to the original interrupt processing when the higher priority interrupt completes.

If an interrupt of the same or lower priority occurs during interrupt processing, the new interrupt request is held until the current interrupt completes processing (unless the ILM or I flag has been modified by an instruction).

The extended intelligent I/O service does not support overlapping operation. Any interrupt or extended intelligent I/O service request that occurs during processing of the extended intelligent I/O service is held.

(8) Saved registers

Figure 2.1.26 shows the order in which registers are saved on the stack.

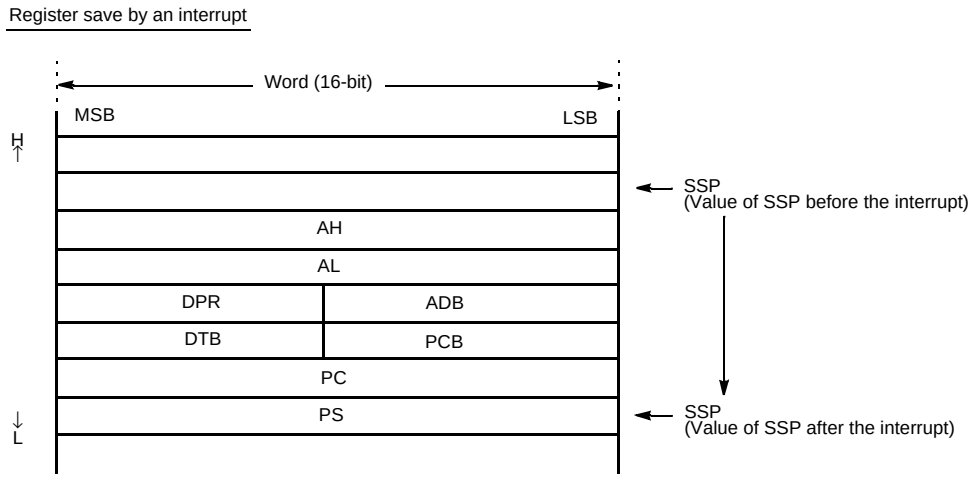


Fig. 2.1.26 Registers Saved on the Stack

(9) Cautions

For some internal resources, reading the control or data registers clears any interrupt request. Clearing an interrupt source by performing a read after generation of an interrupt request but before control has passed to the interrupt processing hardware causes misoperation.
Therefore, when using internal resources for which interrupt requests are cleared by register read operations, do not perform register read operations when interrupts are generated.

■ Software interrupts

(1) Overview

The software interrupt function transfers control from the program currently executing in the CPU to a user-defined interrupt processing program on execution of a special instruction. Software interrupts are always activated by the execution of the software interrupt instruction. The CPU performs the following processing when a software interrupt occurs.

- The A, DPR, ADB, DTB, PCB, PC, and PS registers in the CPU are saved on the system stack.
- The I flag in the PS register is set to "0" to inhibit hardware interrupts.
- Execution branches to the corresponding interrupt vector.

Interrupt requests initiated by the software interrupt instruction (INT) do not have an interrupt request flag or interrupt enable flag. Executing the INT instruction always generates an interrupt request.
The INT instruction does not have interrupt levels. Accordingly, the INT instruction does not modify ILM. Instead the instruction sets the I flag to "0" to hold any subsequent interrupt requests.

(2) Configuration

All software interrupt mechanisms are located in the CPU. The software interrupt instruction must be executed to use a software interrupt.

As shown in Table 2.1.12, software and hardware interrupt vectors share the same area. For example, interrupt request number INT 11 is used by both hardware interrupt #0 and software interrupt INT #11. Therefore, hardware interrupt #0 and INT #11 call the same interrupt processing routine.

(3) Operation

On executing the software interrupt instruction, the CPU activates the software interrupt processing microcode. After saving the A, DPR, ADB, DTB, PCB, PC, and PS registers (12 bytes) to the memory area specified by SSB and SSP, the software interrupt processing microcode reads the 3-byte interrupt vector and loads the vector to PC and PCB, sets the I flag to "0" and the S flag to "1", then executes branch processing. As a result, the next executed instruction is the user-defined interrupt processing program.

Figure 2.1.27 shows the flow of processing from the generation of a software interrupt until the interrupt request is completed by the interrupt processing program.

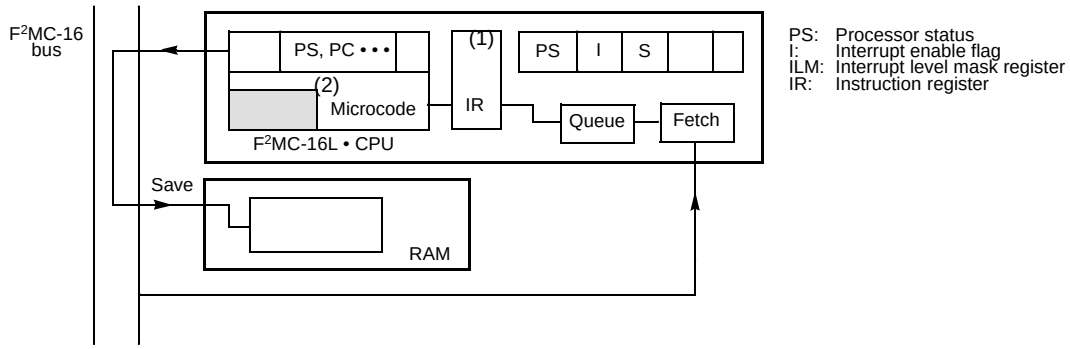


Fig. 2.1.27 Steps from Generation to Release of a Software Interrupt

(1) A software interrupt instruction is executed.0
(2) The microcode for the software interrupt instruction saves the dedicated registers.
(3) Interrupt processing completes on execution of the RETI instruction in the user interrupt processing routine.

(4) Cautions

When the program bank register (PCB) is FFH, the vector area for the CALLV instruction overlaps the table for the INT #vct8 instruction. Take note of the address overlap of the CALLV and INT #vct8 instructions when developing software.

■ Extended intelligent I/O service (EI²OS)

The EI²OS function automatically transfers data between I/O and memory. The function provides DMA-style I/O data handling and replaces I/O data handling by interrupt processing programs. The function has the following advantages over interrupt processing.

- No data transfer program is required. This reduces total program size.
- Data transfer does not use internal registers. This increases transfer speed because register saving is not required.
- The I/O can stop data transfer as required. This eliminates unnecessary data transfer.
- Incrementing or no-change of the buffer address can be selected.
- Incrementing or no-change of the I/O register address can be selected.

On completion, EI²OS sets the completion conditions then automatically branches to an interrupt processing routine. This allows the user to determine the completion conditions.

To implement EI²OS, hardware is distributed between two locations. The blocks have the following registers and descriptors.

• Interrupt control register	Located in the interrupt controller. Specifies the ISD address.
• Extended intelligent I/O service descriptor (ISD)	Located in RAM. Holds the transfer mode, I/O address, transfer count, and buffer address.

Figure 2.1.28 shows an overview of the extended intelligent I/O service.

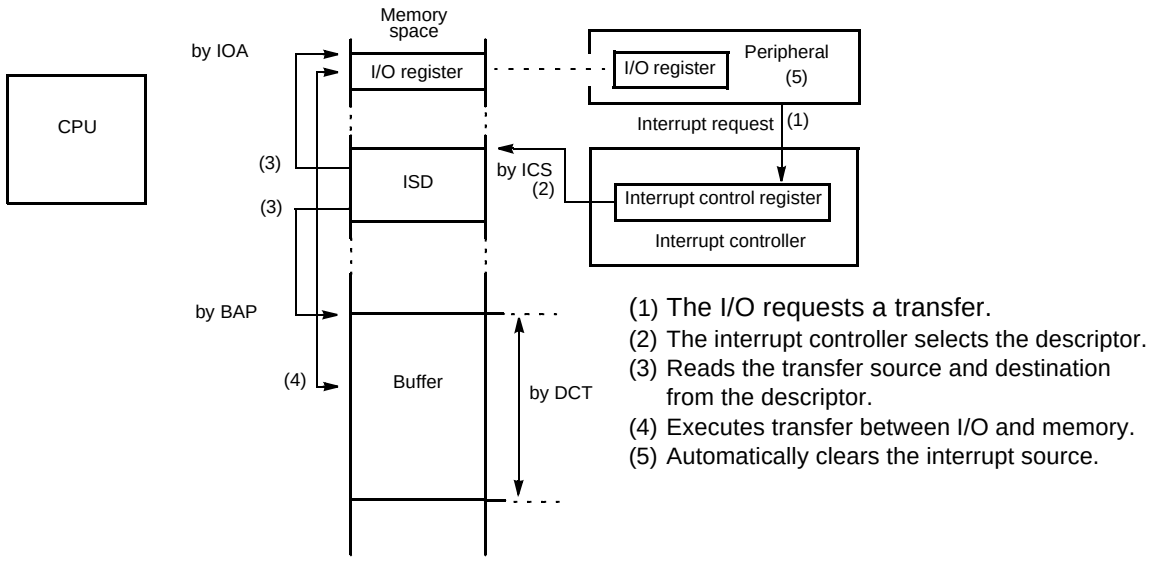


Fig. 2.1.28 Overview of the Extended Intelligent I/O Service

Note: The area that can be specified by IOA is 000000H to 00FFFFH. The area that can be specified by BAP is 000000H to FFFFFFH. The maximum transfer count that can be specified by DCT is 65536.

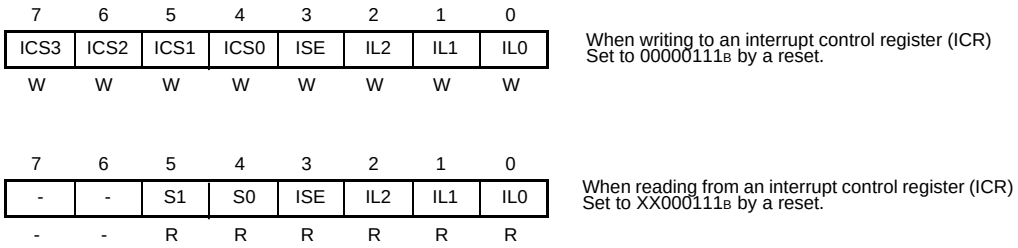
■ Interrupt control registers (ICR00 to 15)

The interrupt control registers are located in the interrupt controller and are provided for each I/O that has an interrupt function. See 2.2.3 "Interrupt Vector Allocation" for details on interrupts and ICRs. The registers perform the following three functions.

- Sets the interrupt level for the corresponding internal resource.
- Selects whether the corresponding internal resource uses a standard interrupt or the extended intelligent I/O service.
- Selects the extended intelligent I/O service channel.

Do not access these registers using read-modify-write instructions as this can cause misoperation.

Figure 2.1.29 shows the bit structure of the interrupt control registers.



Note: ICS3 to 0 are only meaningful when EI^{2}OS is active.
Set ISE to "1" when using EI^{2}OS and set ISE to "0" when not using EI^{2}OS.
Any values can be set to ICS3 to 0 if EI^{2}OS is not used.

Fig. 2.1.29 Interrupt Control Registers (ICR)

[Bits 2 to 0] IL0, IL1, IL2: Interrupt level setting bits

The interrupt level setting bits specify the interrupt level of the corresponding internal resource. The bits are readable and writable. The bits are initialized to level 7 (no interrupt) by a reset. Table 2.1.13 lists the relationship between the interrupt level setting bits and each interrupt level.

Table 2.1.13 Correspondence Between 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)

[Bit 3] ISE: Extended intelligent I/O service enable bit

The EI²OS enable bit. The processor activates EI²OS if ISE is "1" when an interrupt occurs. If the ISE bit is "0", the processor activates the interrupt sequence. The bit is readable and writable. The ISE bit changes to "0" when EI²OS completes (count completion or completion by the internal resource). If the internal resource does not have an EI²OS function, set the ISE bit to "0" by software.

The ISE bit is initialized to "0" by a reset.

[Bits 7 to 4] ICS3 to 0: Extended intelligent I/O service channel select bits

The EI²OS channel select bits specify the EI²OS channel. The bits are write-only. The value set in ICS determines the address of the extended intelligent I/O service descriptor (ISD). ICS is initialized to "0000" by a reset.

Table 2.1.14 lists the correspondence between ICS, the channel number, and the descriptor address.

Table 2.1.14 Correspondence Between ICS, Channel Number, and Descriptor Address

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

[Bits 5, 4] S1, S0: Extended intelligent I/O service status

The extended intelligent I/O service status bits can be referenced to determine the operation status and completion status of EI²OS. The bits are read-only. The bits are initialized to "00" by a reset.

Table 2.1.15 lists the relationship between the S bits and EI²OS status.

Table 2.1.15 S Bits and EI²OS Status

S1	S0	EI²OS Status
0	0	EI²OS active or inactive
0	1	Stopped due to count completion
1	0	Reserved
1	1	Stopped by request from the internal resource

❑ Extended intelligent I/O service descriptor (ISD)

The extended intelligent I/O service descriptors are located in internal RAM between 000100H to 00017FH. ISD consist of the following items.

- Various control data for data transfer
- Status data
- Buffer address pointer

Figure 2.1.30 shows the structure of the extended intelligent I/O service descriptor.

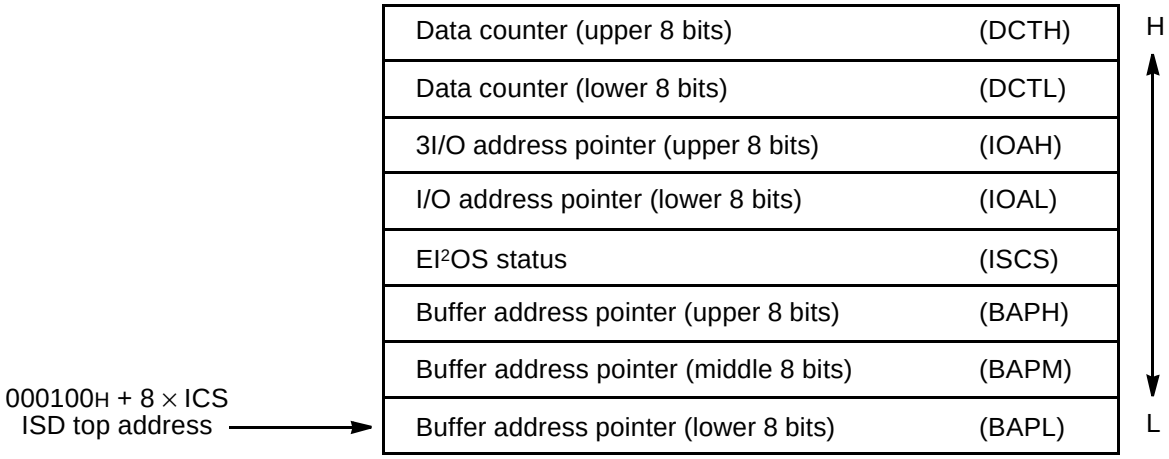


Fig. 2.1.30 Structure of the Extended Intelligent I/O Service Descriptor

❍ Data counter (DCT)

A 16-bit register used as a counter for the number of data transferred. The counter is decremented by one before each data transfer. EI²OS completes when this counter reaches zero. Figure 2.1.31 shows the structure of the data counter.

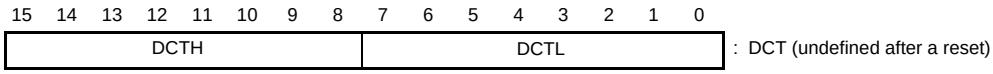


Fig. 2.1.31 Data Counter Structure

❍ I/O register address pointer (IOA)

A 16-bit register that indicates the lower 16 bits (A15 to A0) of the address of the I/O register for data transfer to or from the buffer. The upper address (A23 to A16) is all zeros. This allows I/O to be specified anywhere between locations 000000H and 00FFFFH. Figure 2.1.32 shows the structure of IOA.

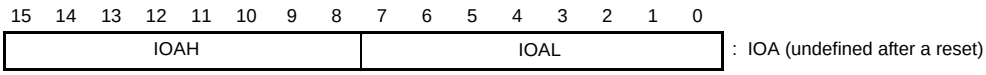


Fig. 2.1.32 Structure of the I/O Register Address Pointer

❍ EI²OS status register (ISCS)

An 8-bit register that specifies whether the buffer address pointer and I/O register address pointer are updated or fixed, the transfer data length (byte or word), and the transfer direction. Figure 2.1.33 shows the structure of ISCS.

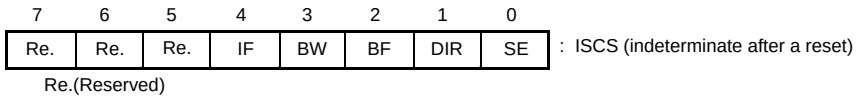


Fig. 2.1.33 ISCS Structure

The meaning of each bit is as follows.

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

0 :Update the I/O register address pointer after data transfer.
1 :Do not update the I/O register address pointer after data transfer.
Note : Only incrementing is available.

[Bit 3] BW: Specifies the data transfer length

0 :Byte
1 :Word

[Bit 2] BF: Specifies whether the buffer address pointer is updated or fixed.

0 :Update the buffer address pointer after data transfer.
1 :Do not update the buffer address pointer after data transfer.
Note: Updating changes only the lower 16 bits of the buffer address pointer.
Only incrementing is available.

[Bit 1] DIR: Specifies the data transfer direction.

0 :I/O address pointer → Buffer address pointer
1 :Buffer address pointer → I/O address pointer

[Bit 0] SE: Controls completion of the extended intelligent I/O service by request from the internal resource.

0 :Do not terminate on request from the internal resource.
1 :Terminate on request from the internal resource.

❑ Buffer address pointer (BAP)

A 24-bit register that stores the address to use for the next EI²OS transfer. As a separate BAP is provided for each EI²OS channel, each EI²OS channel can perform data transfer to any location in the 16MB memory space. If updating is specified by the BF bit in ISCS, the lower 16 bits only of BAP are updated and BAPH does not change.

Figure 2.1.34 shows the structure of BAP.

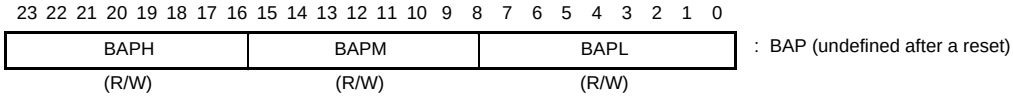


Fig. 2.1.34 Structure of the Buffer Address Pointer

Figure 2.1.35 shows the operation flow of EI²OS. Figure 2.1.36 shows the procedure for using EI²OS.

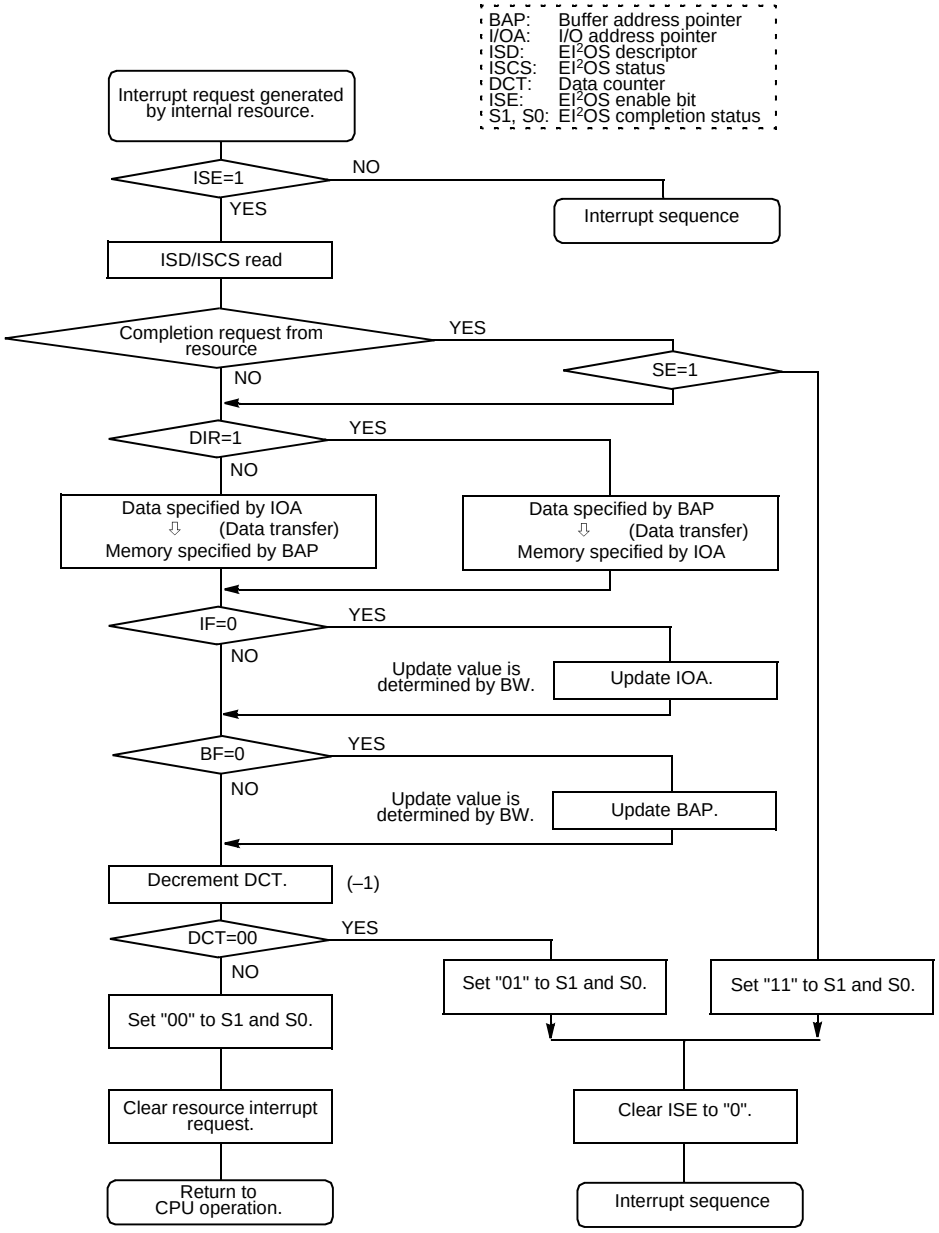


Fig. 2.1.35 EI²OS Operation Flow

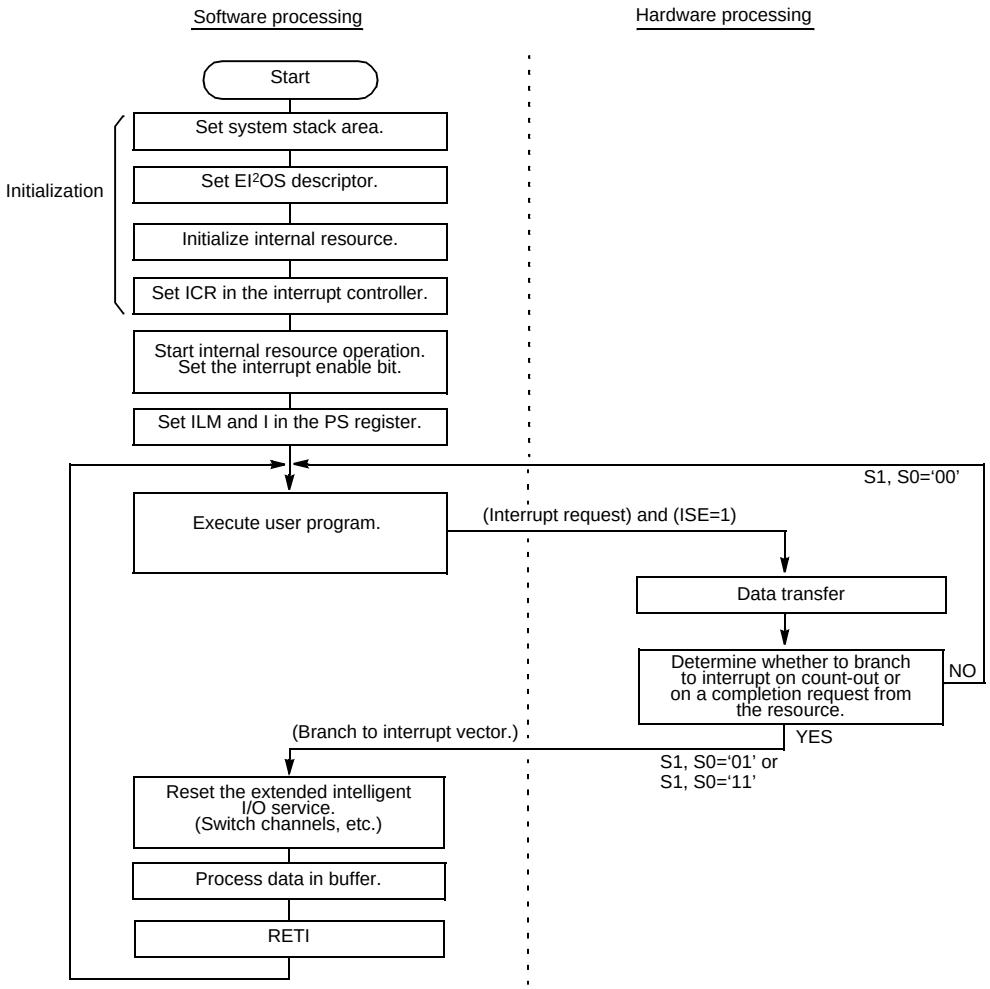


Fig. 2.1.36 Procedure for Using EI²OS

■ Exception processing

The F²MC-16L generates exceptions in the following cases and performs exception processing.

(1) Execution of an undefined instruction

In principle, exception processing is the same as interrupt processing. When an exception is detected at an instruction boundary, control passes from standard processing to exception processing. In general, as exception processing occurs as a result of an unexpected operation, it is recommended that exception processing only be used for purposes such as debugging or to activate emergency recovery software.

❑ Generation of an exception by execution of an undefined instruction

The F²MC-16L treats all codes not defined in the instruction map as undefined instructions. Executing an undefined instruction triggers the same processing as executing the software interrupt instruction "INT 10". That is, AL, AH, DPR, DTB, ADB, PCB, PC, and PS are saved on the system stack, the I flag is set to "0" and the S flag to "1", and execution branches to the program specified by the vector for interrupt 10. The value of PC saved on the stack contains the address of the undefined instruction. For instruction codes of 2 bytes or more, the PC value on the stack contains the address of the code that was recognized as undefined. Therefore, although returning by the RETI instruction is possible, this is meaningless as it only triggers another exception.

2.1.5 Standby Control Register Access

The device is set to the low power modes (stop mode or sleep mode) by writing to the low power mode control register. In this case, use one of the instructions listed in Table 2.1.16. Correct operation cannot be guaranteed if you change to a low power mode using an instruction other than the instructions listed in Table 2.1.16. However, any instruction can be used on the low power mode control register when controlling functions other than changing to a low power mode.

Always write to an even-numbered address when performing a word-length write to the low power mode control register. Changing to a low power mode by writing to an odd-numbered address may cause misoperation.

Table 2.1.16 Instructions to Use When Changing to a Low Power Mode

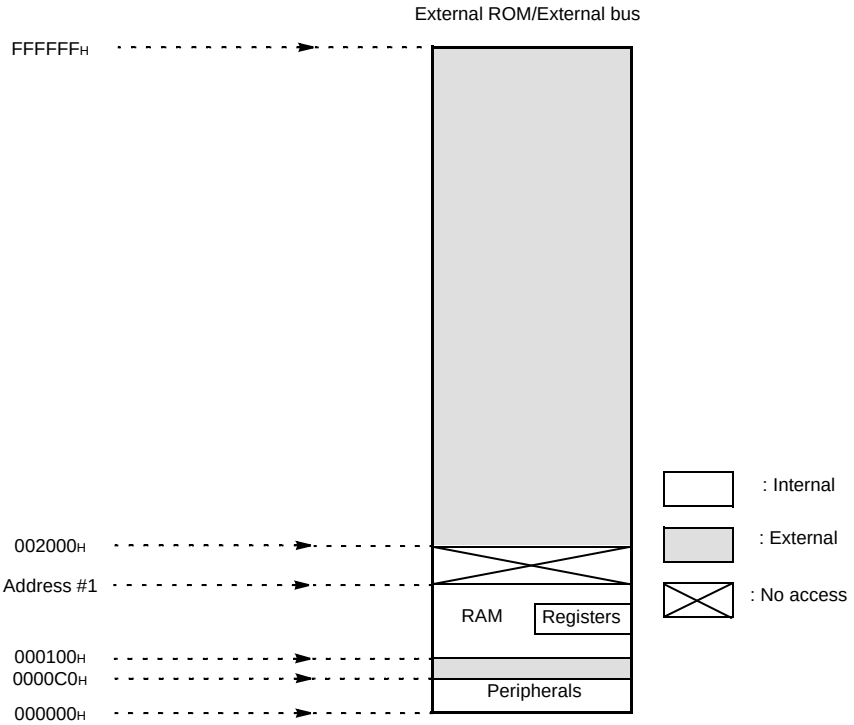
MOV io,#imm8	MOV dir,#imm8	MOV eam,#imm8	MOV eam,Ri
MOV io,A	MOV dir,A	MOV addr16,A	MOV eam,A
MOV @RLi+disp8,A	MOVP addr24,A		
MOVW io,#imm16	MOVW dir,#imm16	MOVW eam,#imm16	MOVW eam,RWi
MOVW io,A	MOVW dir,A	MOVW addr16,A	MOVW eam,A
MOVW @RLi+disp8,A	MOVPW addr24,A		
SETB io:bp	SETB dir:bp	SETB addr16:bp	

2.2 Map

This section describes the allocation of memory space, I/O space, and interrupt numbers in the MB90610A.

2.2.1 Memory Space for Each Mode

Figure 2.2.1 shows the MB90610A memory space. Only external ROM/external bus mode is available.



Product	Address #1
MB90611A	000500H
MB90V610A	001100H

Fig. 2.2.1 Memory Space Allocation for Each MB90610A Mode

Note: When output of the upper address (A23 to A16) is disabled, the MB90610A series can only access a maximum of 64KB.

2.2.2 I/O Map

The following shows the MB90610A I/O map.

Table 2.2.2 MB90610A I/O Map (1)

Address	Register	Abbreviation	Access	Resource Name	Initial Value
000000H	Free		Note 3		
000001H	Port 1 data register	PDR1	R/W*	Port 1 Note 8	XXXXXXXX
000002H	Port 2 data register	PDR2	R/W*	Port 2 Note 7	XXXXXXXX
000003H	Port 3 data register	PDR3	R/W*	Port 3 Note 7	XXXXXXXX
000004H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX
000005H	Port 5 data register	PDR5	R/W	Port 5	--XXXXXX
000006H	Port 6 data register	PDR6	R/W	Port 6	11111111
000007H	Port 7 data register	PDR7	R/W	Port 7	-XXXXXXX
000008H	Port 8 data register	PDR8	R/W	Port 8	-XXXXXXX
000009H	Port 9 data register	PDR9	R/W	Port 9	--XXXXXX
00000AH	Port A data register	PDRA	R/W	Port A	XXXXXXX-
00000BH to 000010H	Free		Note 3		
000011H	Port 1 direction register	DDR1	R/W*	Port 1 Note 8	00000000
000012H	Port 2 direction register	DDR2	R/W*	Port 2 Note 7	00000000
000013H	Port 3 direction register	DDR3	R/W*	Port 3 Note 7	00000000
000014H	Port 4 direction register	DDR4	R/W	Port 4	00000000
000015H	Port 5 direction register	DDR5	R/W	Port 5	--000000
000016H	Analog input enable register	ADER	R/W	Port 6	11111111
000017H	Port 7 direction register	DDR7	R/W	Port 7	-0000000
000018H	Port 8 direction register	DDR8	R/W	Port 8	-0000000
000019H	Port 9 direction register	DDR9	R/W	Port 9	--000000
00001AH	Port A direction register	DDRA	R/W	Port A	0000000-
00001BH to 00001FH	Free		Note 3		
000020H	Mode register 0	SMR0	R/W!	UART0 (SCI)	00000000
000021H	Control register 0	SCR0	R/W!		00000100
000022H	Input data register 0/Output data register 0	SIDR0/SODR0	R/W		XXXXXXXX
000023H	Status register 0	SSR0	R/W!		00001-00
000024H	Mode register 1	SMR1	R/W!	UART1 (SCI)	00000000
000025H	Control register 1	SCR1	R/W!		00000100
000026H	Input data register 1/Output data register 1	SIDR1/SODR1	R/W		XXXXXXXX
000027H	Status register 1	SSR1	R/W!		00001-00

Table 2.2.2 MB90610A I/O Map (2)

Address	Register	Abbreviation	Access	Resource Name	Initial Value
000028H	Interrupt/DTP enable register	ENIR	R/W	DTP/External interrupt	00000000
000029H	Interrupt/DTP source register	EIRR	R/W		00000000
00002AH	Interrupt level setting register	ELVR	R/W		00000000
00002BH					00000000
00002CH	AD control status register	ADCS	R/W!	A/D converter	00000000
00002DH					00000000
00002EH	AD data register	ADCR	R/W! Note 4		XXXXXXXX
00002FH					000000XX
000030H	PPG0 operation mode control register	PPGC0	R/W	PPG0	000000-1
000031H	PPG1 operation mode control register	PPGC1	R/W	PPG1	000000-1
000032H to 000033H	Free		Note 3		
000034H	PPG0 reload register	PRL0	R/W	PPG0	XXXXXXXX
000035H					XXXXXXXX
000036H	PPG1 reload register	PRL1	R/W	PPG1	XXXXXXXX
000037H					XXXXXXXX
000038H	Control status register	TMCSR0	R/W!	16-bit reload timer 0	00000000
000039H					----0000
00003AH	16-bit timer register/16-bit reload register	TMR0/TMRLR0	R/W		XXXXXXXX
00003BH					XXXXXXXX
00003CH	Control status register	TMCSR1	R/W!	16-bit reload timer 1	00000000
00003DH					----0000
00003EH	16-bit timer register/16-bit reload register	TMR1/TMRLR1	R/W		XXXXXXXX
00003FH					XXXXXXXX
000040H to 000043H	Free		Note 3		
000044H	Mode register 2	SMR2	R/W!	UART2 (SCI)	00000000
000045H	Control register 2	SCR2	R/W!		00000100
000046H	Input data register 2/Output data register 2	SIDR2/SODR2	R/W		XXXXXXXX
000047H	Status register 2	SSR2	R/W!		00001-00
000048H	CS control register 0	CSCR0	R/W	Chip select function	----0000
000049H	CS control register 1	CSCR1	R/W		----0000
00004AH	CS control register 2	CSCR2	R/W		----0000
00004BH	CS control register 3	CSCR3	R/W		----0000
00004CH	CS control register 4	CSCR4	R/W		----0000
00004DH	CS control register 5	CSCR5	R/W		----0000
00004EH	CS control register 6	CSCR6	R/W		----0000
00004FH	CS control register 7	CSCR7	R/W		----0000

Table 2.2.2 MB90610A I/O Map (3)

Address	Register	Abbreviation	Access	Resource Name	Initial Value
000050H	Free		Note 3		
000051H	UART0 (SCI) machine clock divide-by-n control register	CDCR0	W	UART0 (SCI)	----1111
000052H	Free		Note 3		
000053H	UART1 (SCI) machine clock divide-by-n control register	CDCR1	W	UART1 (SCI)	----1111
000054H	Free		Note 3		
000055H	UART2 (SCI) machine clock divide-by-n control register	CDCR2	W	UART2 (SCI)	----1111
000056H to 00008FH	Free		Note 3		
000090H to 00009EH	Reserved system area		Note 1		
00009FH	Delay interrupt generate/ release register	DIRR	R/W	Delay interrupt generation module	-------0
0000A0H	Low power mode control register	LPMCR	R/W!	Low power consumption	00011000
0000A1H	Clock selection register	CKSCR	R/W!	Low power consumption	11111100
0000A2H to 0000A4H	Free		Note 3		
0000A5H	Auto-ready function selection register	ARSR	W	External pins	0011--00
0000A6H	External address output control register	HACR	W	External pins	00000000
0000A7H	Bus control signal selection register	ECSR	W	External pins	-00*0000
0000A8H	Watchdog timer control register	WDTC	R/W!	Watchdog timer	XXXXX111
0000A9H	Timebase timer control register	TBTC	R/W!	Timebase timer	1--00100
0000AAH to 0000AFH	Free		Note 3		

Table 2.2.2 MB90610A I/O Map (4)

Address	Register	Abbreviation	Access	Resource Name	Initial Value
0000B0H	Interrupt control register 00	ICR00	R/W!	Interrupt controller	00000111
0000B1H	Interrupt control register 01	ICR01	R/W!		00000111
0000B2H	Interrupt control register 02	ICR02	R/W!		00000111
0000B3H	Interrupt control register 03	ICR03	R/W!		00000111
0000B4H	Interrupt control register 04	ICR04	R/W!		00000111
0000B5H	Interrupt control register 05	ICR05	R/W!		00000111
0000B6H	Interrupt control register 06	ICR06	R/W!		00000111
0000B7H	Interrupt control register 07	ICR07	R/W!		00000111
0000B8H	Interrupt control register 08	ICR08	R/W!		00000111
0000B9H	Interrupt control register 09	ICR09	R/W!		00000111
0000BAH	Interrupt control register 10	ICR10	R/W!		00000111
0000BBH	Interrupt control register 11	ICR11	R/W!		00000111
0000BCH	Interrupt control register 12	ICR12	R/W!		00000111
0000BDH	Interrupt control register 13	ICR13	R/W!		00000111
0000BEH	Interrupt control register 14	ICR14	R/W!		00000111
0000BFH	Interrupt control register 15	ICR15	R/W!		00000111
0000C0H to 0000FFH	External area Note 2				

Note 1: Access prohibited.

Note 2: This is the only external access area in the area below address 0000FFH. Access this address as an external I/O area.

Note 3: Areas marked as "free" in the I/O map are reserved areas. These areas are accessed by internal access. No access signals are output on the external bus.

Note 4: Only bit 15 can be written. The other bits are written to by the test function. Reading bits 10 to15 returns zeros.

Note 5: The R/W! symbol in the access column indicates that some bits are read-only or write-only. See the resource's register list for details.

Note 6: Using a read-modify-write instruction (such as the bit set instruction) to access one of the registers indicated by R/W!, R/W*, or W in the access column sets the specified bit to the desired value. However, this can cause misoperation if the other register bits include write-only bits. Therefore, do not use read-modify-write instructions to access these registers.

Note 7: This register is only available when the address/data bus is in multiplex mode. Access to the register is prohibited in non-multiplex mode.

Note 8: This register is only available when the external data bus is in 8-bit mode. Access to the register is prohibited in 16-bit mode.

Initial values

0: The initial value for this bit is zero.
1: The initial value for this bit is one.
*: The initial value for this bit is one or zero. (Determined by the level of the MD0 to 2 pins.)
X: The initial value for this bit is indeterminate.
-: This bit is not used. The initial value is indeterminate.

Note: The initial values listed for write-only bits are the initial values set by a reset. They are not the values returned by a read.

Also, LPMCR, CKSCR, and WDTC are sometimes initialized and sometimes not initialized, depending on the reset type. The listed initial values are for when these registers are initialized.

2.2.3 Interrupt Vector Allocation

The following shows the interrupt vector allocation in the MB90610A.

Table 2.2.3 MB90610A Interrupt Vector Allocation

Interrupt	I²OS Support	Interrupt vector			Interrupt Control Register	
		Number		Address	ICR	Address
Reset	×	#08	08H	FFFFDCH	-	-
INT 9 instruction	×	#09	09H	FFFFD8H	-	-
Exception	×	#10	0AH	FFFFD4H	-	-
External interrupt #0	O	#11	0BH	FFFFD0H	ICR00	0000B0H
External interrupt #1	O	#13	0DH	FFFFC8H	ICR01	0000B1H
External interrupt #2	O	#15	0FH	FFFFC0H	ICR02	0000B2H
External interrupt #3	O	#17	11H	FFFFB8H	ICR03	0000B3H
External interrupt #4	O	#19	13H	FFFFB0H	ICR04	0000B4H
External interrupt #5	O	#21	15H	FFFFA8H	ICR05	0000B5H
External interrupt #6	O	#23	17H	FFFFA0H	ICR06	0000B6H
UART0 transmit complete	O	#24	18H	FFFF9CH		
External interrupt #7	O	#25	19H	FFFF98H	ICR07	0000B7H
UART1 transmit complete	O	#26	1AH	FFFF94H		
PPG #0	×	#27	1BH	FFFF90H	ICR08	0000B8H
PPG #1	×	#28	1CH	FFFF8CH		
16-bit reload timer #0	O	#29	1DH	FFFF88H	ICR09	0000B9H
16-bit reload timer #1	O	#30	1EH	FFFF84H		
A/DC measurement complete	O	#31	1FH	FFFF80H	ICR10	0000BAH
UART2 transmit complete	O	#33	21H	FFFF78H	ICR11	0000BBH
Timebase timer interval interrupt	×	#34	22H	FFFF74H		
UART2 receive complete	◎	#35	23H	FFFF70H	ICR12	0000BCH
UART1 receive complete	◎	#37	25H	FFFF68H	ICR13	0000BDH
UART0 receive complete	◎	#39	27H	FFFF60H	ICR14	0000BEH
Delay interrupt generation module	×	#42	2AH	FFFF54H	ICR15	0000BFH

Note: O indicates I²OS support (no stop request), ◎ indicates I²OS support (with stop request), and × indicates no I²OS support. Do not set I²OS activation in ICRXX for resources that do not support I²OS.

2.2.4 Clock Supply Map

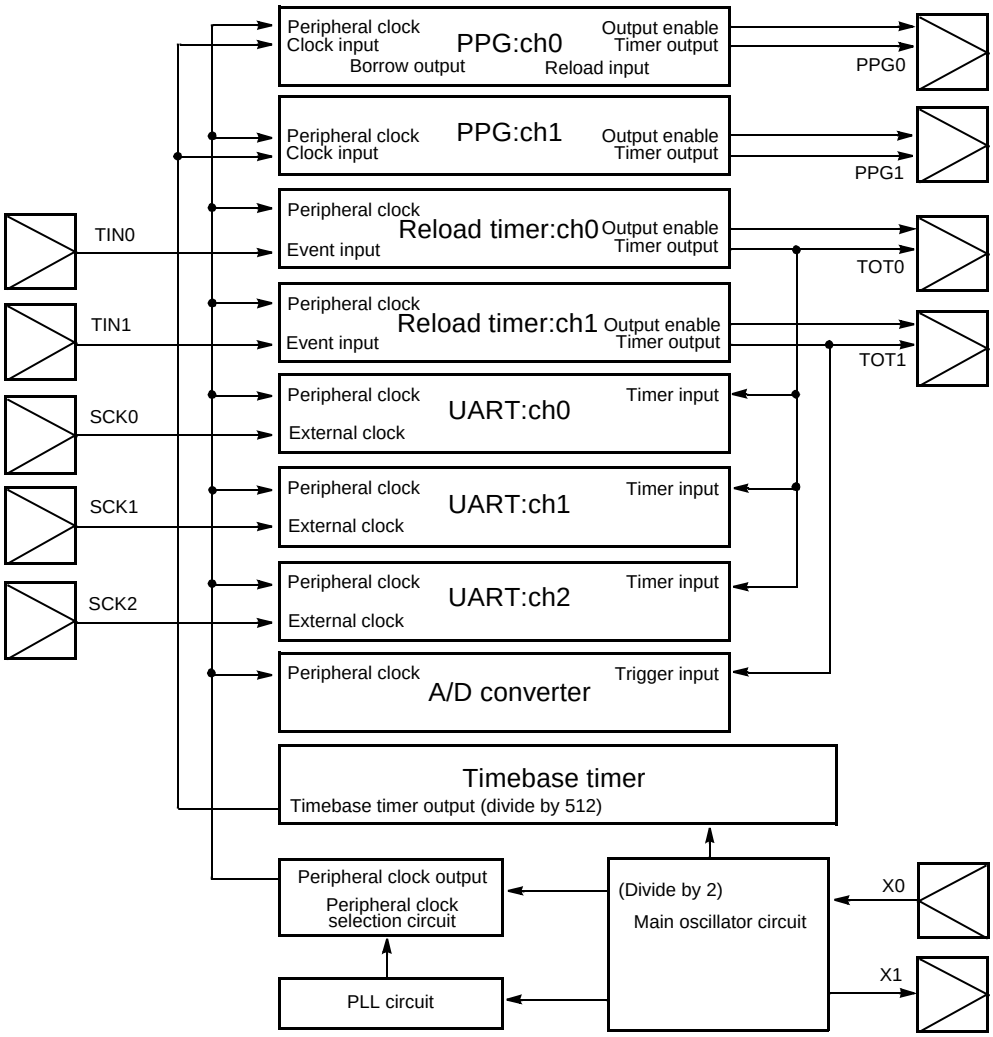


Fig. 2.2.2 Clock Supply Map

2.3 Parallel Ports

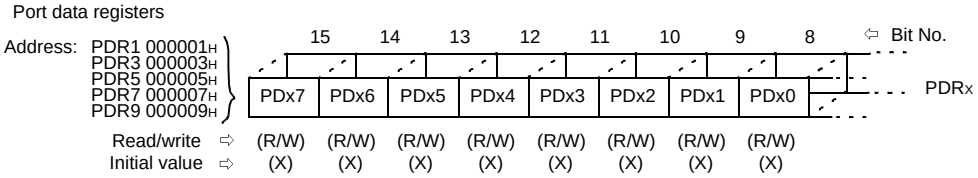
The MB90610A has 58 I/O pins, 18 output pins, and 8 open drain output pins.

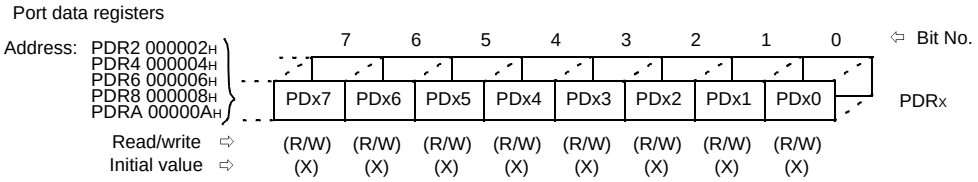
Ports 1 to 5 and ports 7 to A are I/O ports. The ports are inputs when the corresponding direction register bit is "0" and outputs when the corresponding bit is "1".

Port 1 is only available when the external data bus is in 8-bit mode. Access is prohibited in 16-bit mode. Ports 2 and 3 are only available when the address/data bus is in multiplex mode. Access is prohibited in non-multiplex mode.

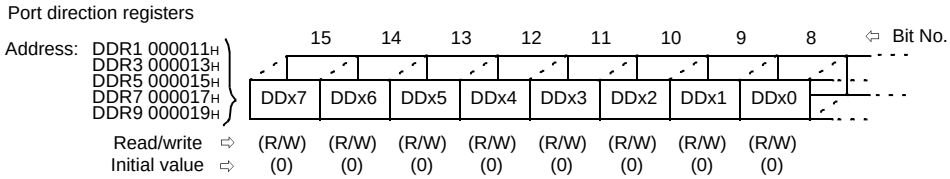
Port 6 is an open drain port. Port 6 can only be used as a port when the analog input enable register is "0".

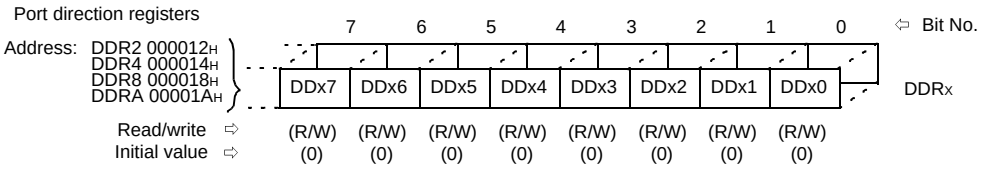
2.3.1 Registers





Note: No register bits are provided for bits 6 and 7 of port 5.
No register bit is provided for bit 7 of port 7.
No register bit is provided for bit 7 of port 8.
No register bits are provided for bits 6 and 7 of port 9.
No register bit is provided for bit 0 of port A.





Note: No register bits are provided for bits 6 and 7 of port 5.
No register bit is provided for bit 7 of port 7.
No register bit is provided for bit 7 of port 8.
No register bits are provided for bits 6 and 7 of port 9.
No register bit is provided for bit 0 of port A.
Port 6 does not have a DDR.

Analog input enable register

	15	14	13	12	11	10	9	8	⇦ Bit No.
ADER 000016H	ADE7	ADE6	PADE5	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)	

2.3.2 Block Diagram

■ I/O ports

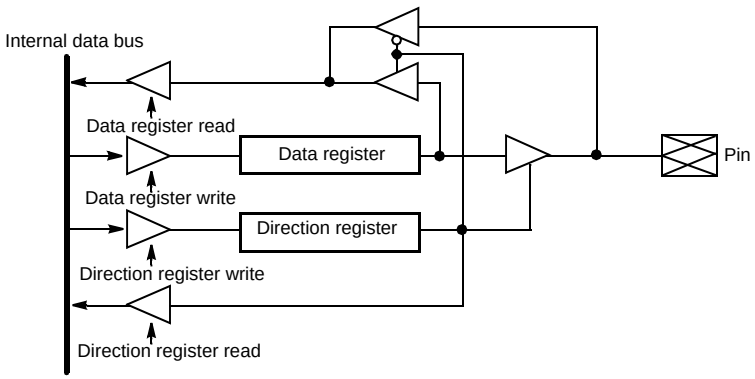


Fig. 2.3.1 I/O Port Block Diagram

■ Open drain port (Also used as analog inputs)

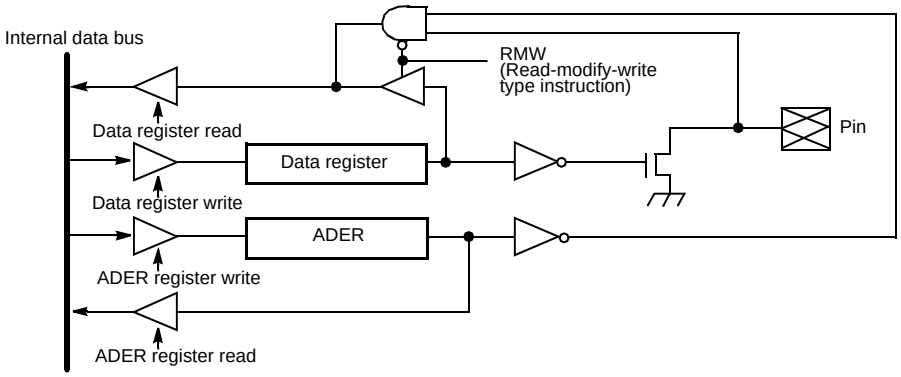
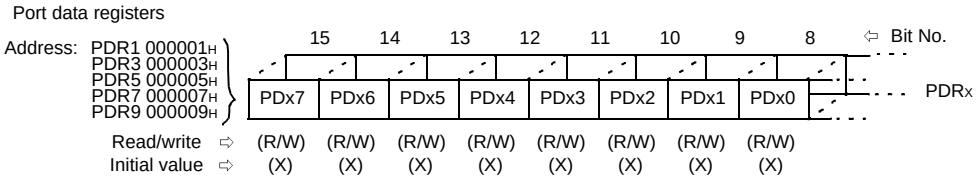


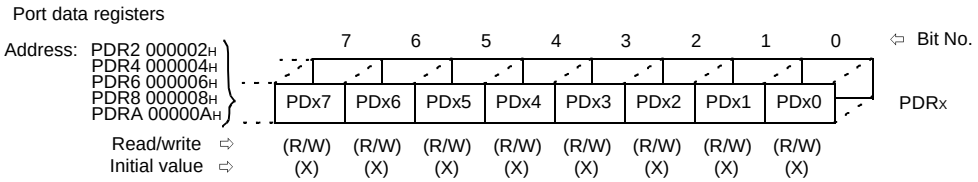
Fig. 2.3.2 Open Drain Port Block Diagram

2.3.3 Register Details

(1) PDR1, 2, 3, 4, 5, 7, 8, 9, A (Port data registers)

■ Register layout





Note: No register bits are provided for bits 6 and 7 of port 5.
No register bit is provided for bit 7 of port 7.
No register bit is provided for bit 7 of port 8.
No register bits are provided for bits 6 and 7 of port 9.
No register bit is provided for bit 0 of port A.

Port 1 is only available when the external data bus is in 8-bit mode. Access is prohibited in 16-bit mode.

Ports 2 and 3 are only available in multiplex mode. Access is prohibited in non-multiplex mode.

■ Register contents

When the pins on ports other than port 6 are not set as output pins for their respective peripherals, each pin can be individually specified as either an input or an output by setting the direction register. When specified as inputs, reading the data register reads the levels applied to the pins. When specified as outputs, reading the data register reads the data register latch value. The same applies when reading as part of a read-modify-write.

When used as control outputs, reading the data register reads the control output value, irrespective of the direction register value.

Note: Accessing these registers using a read-modify-write instruction (bit setting and similar instructions) sets the specified bit to the desired value. However, if the other register bits include bits set as inputs, the output register values for these bits are overwritten with the current input values at the pins. Therefore, when switching a pin from input to output, write the desired value to the PDR before setting the bit as an output in the DDR.

Note: Reading and writing to I/O ports differs from reading and writing to memory as follows.

- Input mode
  Reading: The read value is the level at the corresponding pin.
  Writing: The write data is stored in the output latch. The data is not output to the pin.

- Output mode
  Reading: The read value is the value stored in the PDR.
  Writing: The write data is stored in the output latch and output to the pin.

Note: Note that the R/W operation for port 6 differs from other ports.

Port 6 (P67 to P60) is a general-purpose I/O port with open drain outputs. The port shares pins with the analog inputs. Always set the corresponding ADER bit to "0" when using as a general-purpose port. When using port 6 as an input port, set the output data register value to "1" to turn off the open drain output transistor and provide a pull-up resistor to the external pin. Use one of the following two operations for reading, depending on the instruction used.

- Reading using a read-modify-write type instruction
  Reads the contents of the output data register. Bits not specified by the instruction do not change, even if the pin is forcibly driven to "0" from outside the device.

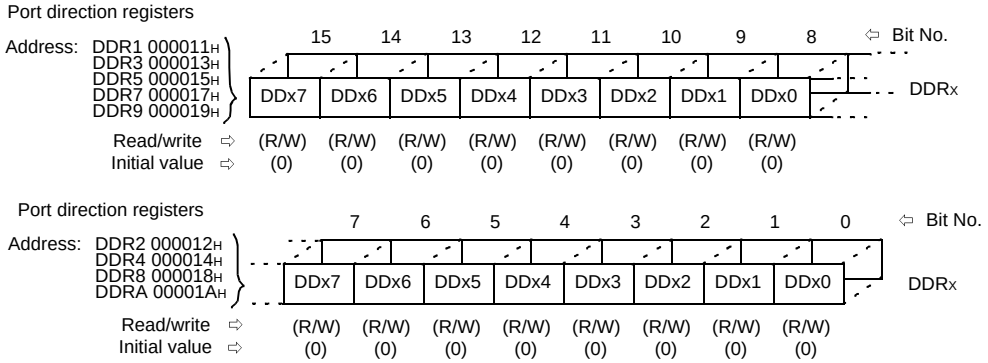
- Reading using other instructions
  Reads the pin level.

When used as an output port, writing the desired value to the corresponding output data register changes the pin value.

Reading always returns "0" for pins for which the corresponding bit in the analog input enable register is set to "1".

(2) DDR1, 2, 3, 4, 5, 7, 8, 9, A (Port direction registers)

■ Register layout



Note: No register bits are provided for bits 6 and 7 of port 5.
No register bit is provided for bit 7 of port 7.
No register bit is provided for bit 7 of port 8.
No register bit is provided for bit 0 of port A.
No register bits are provided for bits 6 and 7 of port 9.
Port 6 does not have a DDR.

Port 1 is only available when the external data bus is in 8-bit mode. Access is prohibited in 16-bit mode.

Ports 2 and 3 are only available in multiplex mode. Access is prohibited in non-multiplex mode.

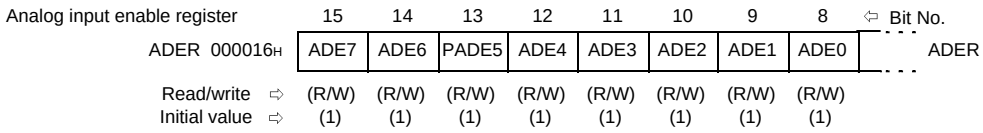
■ Register contents

When pins are used as ports, the register bits control the corresponding pins as follows.

0: Input mode
1: Output mode

Bits are set to "0" by a reset.

(3) ADER (Analog input enable register)



■ Register contents
Controls each pin of port 6 as follows.

0: Port input mode
1: Analog input mode
Bits are set to "1" by a reset.

Note: Inputting an intermediate level signal in port input mode causes a leak current to flow. Therefore, set to analog input mode when applying an analog input.

2.3.4 Port Pin Allocation

Ports 1, 4, and 5 on the MB90610A series share pins with the external bus. The bus mode and register settings select the pin functions. Table 2.3.1 lists the port pin allocation for each mode.

Table 2.3.1 Port Pin Allocation for Each Mode

Pin	Function							
	Non-Multiplex mode				Multiplex mode			
	External Address Control				External Address Control			
	Enable (Address)		Disable (Port)		Enable (Address)		Disable (Port)	
	External Bus Width		External Bus Width		External Bus Width		External Bus Width	
	8-bit	16-bit	8-bit	16-bit	8-bit	16-bit	8-bit	16-bit
D07 to D00/ AD07 to AD00	D07 to D00				AD07 to AD00			
P17 to P10/D15 to D08/AD15 to AD08PortD15 to D08	Port	D15 to D08	Port	D15 to D08	A15 to A08	AD15 to AD08	A15 to A08	AD15 to AD08
P27 to P20/A07 to A00	A07 to A00		A07 to A00		Port			
P37 to P30/A15 to A08	A15 to A08		A15 to A08					
P47 to P40/A23 to A16	A23 to A16		Port		A23 to A16		Port	
ALE	ALE				ALE			
RDX	RDX				RDX			
P55/WRLX	WRLX				WRLX			
P54/WRHX	Port	WRHX	Port	WRHX	Port	WRHX	Port	WRHX
P53/HRQ	HRQ				HRQ			
P52/HAKX	HAKX				HAKX			
P51/RDY	RDY				RDY			
P50/CLK	CLK				CLK			

Note: The upper address, WRLX, WRHX, HAKX, HRQ, RDY, and CLK can be set for use as ports by function selection.

2.4 UART 0/1/2 (SCI)

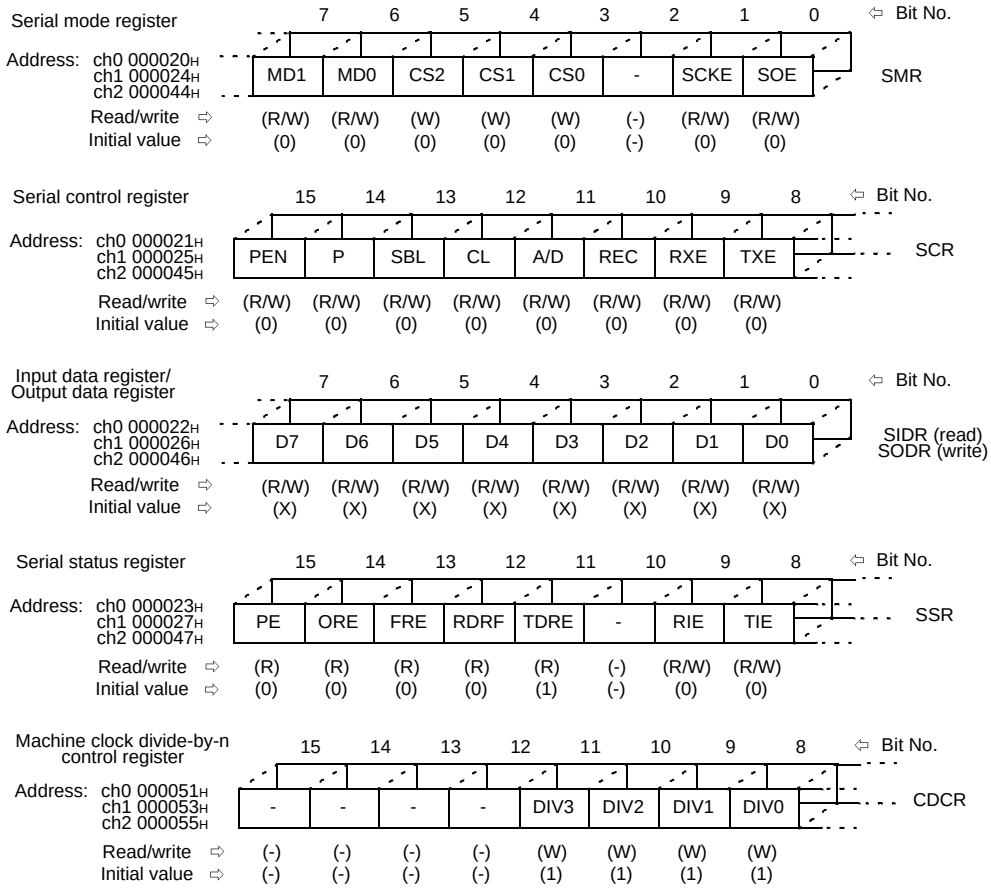
UART 0, 1, and 2 are serial I/O ports that can be used for CLK asynchronous (start-stop synchronization) or CLK synchronous (I/O expansion serial) data transfer. The ports have the following features.

- Full duplex, double buffered
- Supports CLK asynchronous (start-stop synchronization) and CLK synchronous (I/O expansion serial) data transfer
- Multiprocessor mode
- Internal dedicated baud rate generator

  CLK asynchronous: 62500/31250/19230/9615/4808/2404/1202 bps
  CLK synchronous: 2M/1M/500K/250K bps

- Supports flexible baud rate setting using an external clock
- Error detect function (parity, framing, and overrun)
- NRZ type transfer signal
- Intelligent I/O service support

2.4.1 Registers



2.4.2 Block Diagram

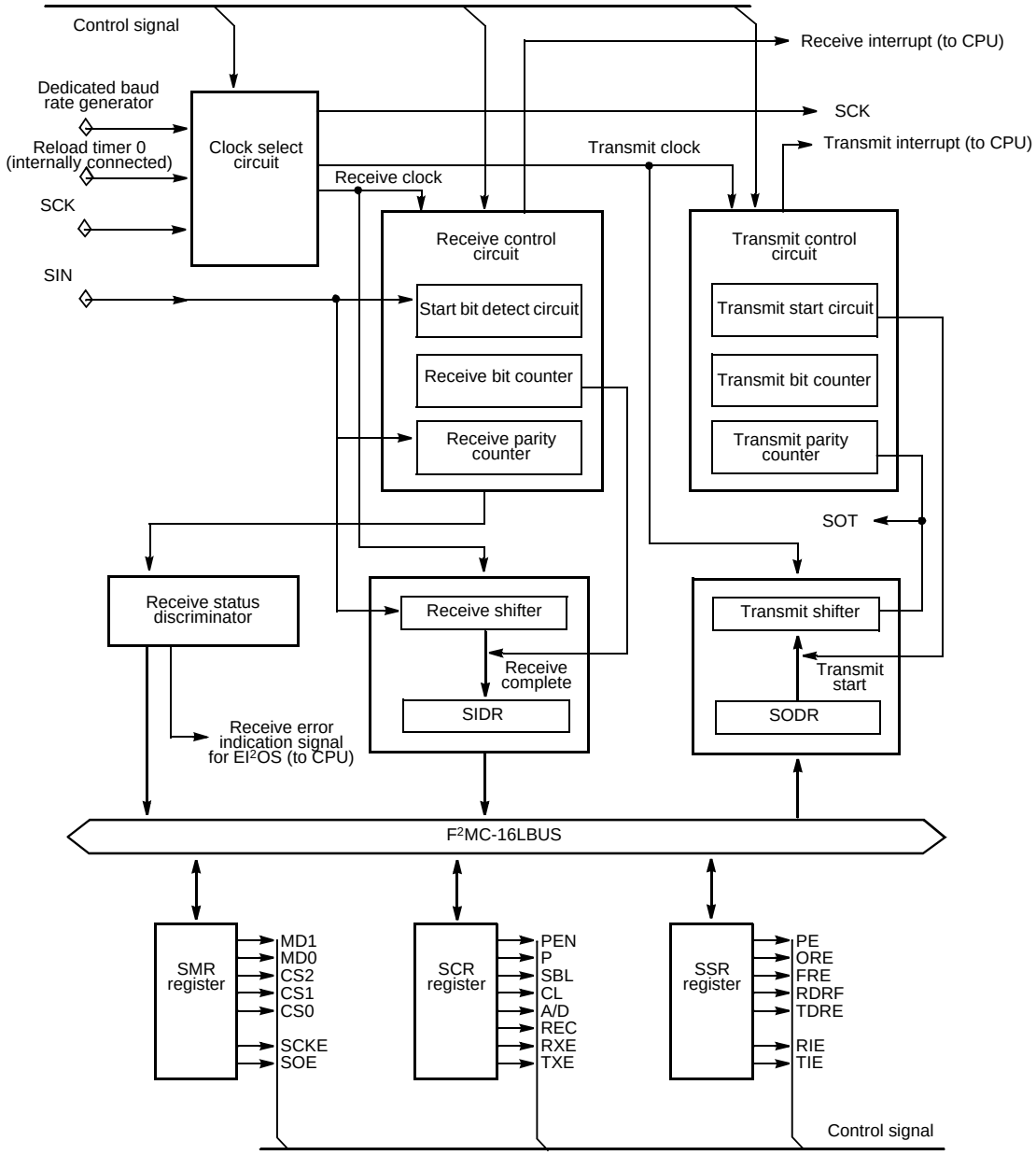
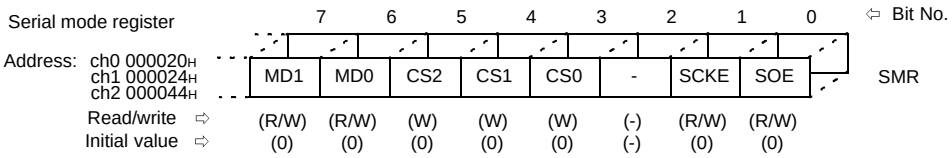


Fig. 2.4.1 Overall Block Diagram

2.4.3 Register Details

(1) SMR (Serial mode register)



Note 1: SMR specifies the operation mode of the UART. Set the operation mode while operation is halted. Do not write to this register during UART operation.

Note 2: Do not use read-modify-write instructions to access the register as this can cause misoperation.

[Bits 7, 6] MD1, MD0 (MoDe select):
Tese bits select the UART operation mode.

Table 2.4.1 Operation Mode Selection

Mode	MD1	MD0	Operation Mode
0	0	0	CLK asynchronous (start-stop synchronization) Normal mode
1	0	1	CLK asynchronous (start-stop synchronization) Multiprocessor mode
2	1	0	CLK synchronous mode
-	1	1	Prohibited setting

Note: The CLK asynchronous mode, mode 1 (multiprocessor), is used when a number of slave CPUs are connected to a single host CPU. As the UART resource cannot determine the data format of the received data, "master" multiprocessor mode only is supported.

Also, as the parity check function cannot be used, set the PEN bit of the SCR register to "0".

[Bits 5, 4, 3] CS2, CS1, CS0 (Clock Select):
Selects the baud rate clock source. When the dedicated baud rate generator is selected, this also selects the baud rate.

Table 2.4.2 Clock Input Selection

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

Note: The MB90610A series has two 16-bit timers. Selecting the internal timer selects timer 0.
CS2, CS1, and CS0 are write-only. Reading always returns "1".

[Bit 2] Free:

Accessing this bit is prohibited.

[Bit 1] SCKE (SCLK enable):

Specifies whether to use the SCK0 pin as the clock input pin or clock output pin for CLK synchronous mode (mode 2).

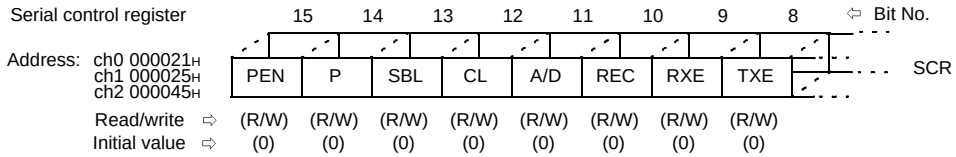
Set to zero in CLK asynchronous mode or external clock mode.
0: Operate as the clock input pin.
1: Operate as the clock output pin.

Note: When using this pin as the clock input pin, external clock source operation must be selected.

[Bit 0] SOE (Serial output enable):

The external pin (SOT) can also be used as a general-purpose I/O port. This bit specifies whether to use the pin as the serial output pin or I/O port pin.
0: Operate as a general-purpose I/O port pin.
1: Operate as a serial data output pin (SOT).

(2) SCR (Serial control register)



SCR controls the transmission protocol for serial communications.

[Bit 15] PEN (Parity Enable):

Specifies whether to add a parity bit in serial data I/O.
0: Do not use parity.
1: Use parity.

Note: Parity can only be used in the normal mode (mode 0) of CLK asynchronous (start-stop synchronization) data transfer. Parity cannot be used in multiprocessor mode (mode 1) and CLK synchronous data transfer mode (mode 2).

[Bit 14] P (Parity):

Specifies even or odd parity when parity is selected for data transfer.
0: Even parity
1: Odd parity

[Bit 13] SBL (Stop Bit Length):

Specifies the number of stop bits. The stop bits are the frame end mark for CLK asynchronous (start-stop synchronization) data transfer.
0: 1 stop bit
1: 2 stop bits

[Bit 12] CL (Character Length):

Specifies the number of data bits in a transmit or receive frame.
0: 7-bit data
1: 8-bit data

Note: Using 7 data bits is only available in the normal mode (mode 0) of CLK asynchronous (start-stop synchronization) data transfer. Use 8 data bits for multiprocessor mode (mode 1) and CLK synchronous data transfer mode (mode 2).

[Bit 11] A/D (Address/Data):

Specifies the data format for transmit and receive frames in multiprocessor mode (mode 1) of CLK asynchronous (start-stop synchronization) data transfer.
0: Data frame
1: Address frame

[Bit 10] REC (Receiver Error Clear):

Writing "0" clears the SSR register error flags (PE, ORE, and FRE).

Writing "1" has no effect. Reading always returns '1".

Note: When the UART is operating and receive interrupts enabled, only write "0" to REC when one of PE, ORE, or FRE is "1".

[Bit 9] RXE (Receiver Enable):

Controls the UART receive operation.
0: Disable receive operation.
1: Enable receive operation.

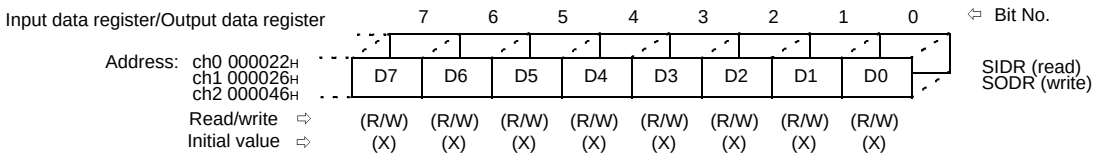
Note: If reception is disabled during a receive (when data is being input to the receive shift register), the receive operation halts after reception of the frame has completed and the data is stored in the receive data buffer register (SIDR).

[Bit 8] TXE (Transmitter Enable):

Controls the UART transmit operation.
0: Disable transmit operation.
1: Enable transmit operation.

Note: If transmission is disabled during a transmit (when data is being output from the transmit register), the transmit operation halts after the transmit data buffer register (SODR) becomes empty.

(3) SIDR (Input data register) and SODR (Output data register)



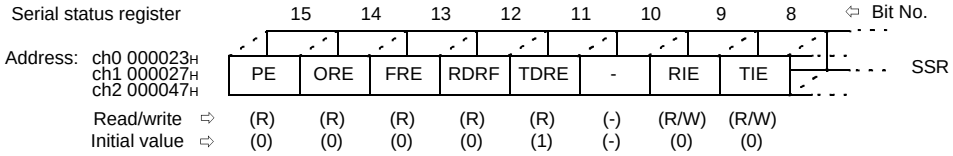
These registers are the receive and transmit data buffers.

The most significant bit (D7) is ignored if the data length is 7 bits. Only write to the SODR register when TDRE in the SSR register is "1". Only read the SIDR register when RDRF in the SSR register is "1".

Note1: Writing to this address writes to the SODR register. Reading reads from the SIDR register.

Note 2: Do not use read-modify-write instructions to access the register as this can cause misoperation.

(4) SSR (Serial status register)



SSR contains the flags that indicate the operation status of the UART.

[Bit 15] PE (Parity Error):

The flag is an interrupt request flag and is set when a receive parity error occurs.

Once set, writing "0" to the REC bit (bit 10) in the SCR register clears the flag.

When this flag is set, the data in SIDR is invalid.
0: No parity error
1: Parity error

[Bit 14] ORE (Over Run Error):

The flag is an interrupt request flag and is set when a receive overrun error occurs.
Once set, writing "0" to the REC bit (bit 10) in the SCR register clears the flag.

When this flag is set, the data in SIDR is invalid.
0: No overrun error
1: Overrun error

[Bit 13] FRE (Framing Error):

The flag is an interrupt request flag and is set when a receive framing error occurs.

Once set, writing "0" to the REC bit (bit 10) in the SCR register clears the flag.

When this flag is set, the data in SIDR is invalid.
0: No framing error
1: Framing error

[Bit 12] RDRF (Receiver Data Register Full):

The flag is an interrupt request flag and indicates that receive data is present in the SIDR register.

The bit is set when the receive data is loaded into the SIDR register and automatically cleared by reading the SIDR register.
0: No receive data
1: Receive data present

[Bit 11] TDRE (Transmitter Data Register Empty):

The flag is an interrupt request flag and indicates that transmit data can be written to the SODR register.

Writing data to SODR clears the flag. The flag is set again when the data is loaded to the transmit shifter and transmission starts. This indicates that the next transmit data can be written.
0: Writing transmit data is disabled.
1: Writing transmit data is enabled.

[Bit 9] RIE (Receiver Interrupt Enable):

Controls receive interrupts.
0: Disable interrupts.
1: Enable interrupts.

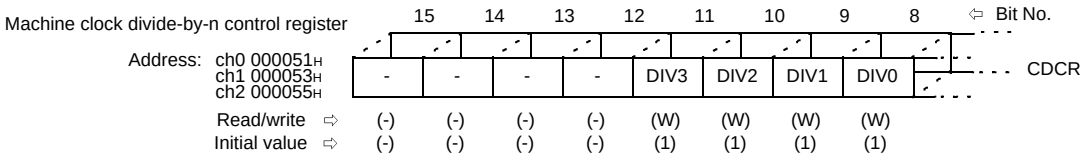
Note: Receive interrupt sources are generated by an error due to PE, ORE, or FRE as well as by RDRF (normal receive).

[Bit 8] TIE (Transmitter Interrupt Enable):

Controls transmit interrupts.
0: Disable interrupts.
1: Enable interrupts.

Note: Transmit interrupts are generated when a TDRE transmit request is present.

(5) CDCR (Machine clock divide-by-n control register)



Note 1: Do not use read-modify-write instructions to access the register as this can cause misoperation.

The operating clock for the UART is obtained by dividing the machine clock. The divider is designed to obtain specific baud rates for a range of machine clocks. The CDCR register controls the machine clock divider.

[Bit 11, 10, 9, 8] DIV3 to 0 (DIVide 3 to 0)

Sets the machine clock divide ratio.

Table 2.5.3 Machine Clock Divide Ratio

DIV3 to DIV0	Divide Ratio
1110_B	Divide by 2
1101_B	Divide by 3
1100_B	Divide by 4
1011_B	Divide by 5
1010_B	Divide by 6
1001_B	Divide by 7
1000_B	Divide by 8

Set the CDCR register bits as follows for the different machine clocks ϕ. These settings obtain the baud rates listed in Table 2.4.4.

Machine Clockϕ	div	DIV3	DIV2	DIV1	DIV0	ϕ÷div
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 different machine clock or div settings to those listed above, ensure that the product of ϕ ÷ div does not exceed 4 MHz.

2.4.4 Operation

(1) Operating modes

Table 2.4.3 lists the UART operating modes. Set the SMR and SCR registers to switch between modes.

Table 2.4.3 UART Operating Modes

Mode	Parity	Data Length	Operating Mode	Number of Stop Bits
0	On/Off	7	CLK asynchronous (start-stop synchronization) Normal mode	1 bit or 2 bits
	On/Off	8		
1	Off	8+1	CLK asynchronous (start-stop synchronization) Multi-processor mode	
2	Off	8	CLK synchronous mode	None

However, for CLK asynchronous (start-stop synchronization) mode, the number of stop bits can only be set for transmission. The number of receive stop bits is always one. Do not set modes other than those listed above. The UART does not operate if an invalid mode is set.

Note: UART CLK synchronous mode uses clock control (I/O expansion serial). No start or stop bit is added to the data in clock synchronous mode.

(2) UART clock selection

a) Dedicated baud rate generator

The baud rate when the dedicated baud rate generator is selected is as follows.

Table 2.4.4 Baud Rate

(i) ϕ ÷div = 2 MHz

CS2	CS1	CS0	Asynchronous (start-stop synchronization)	CLK Synchronous
0	0	0	9615 (1)	Prohibited setting
0	0	1	31250 (2)	
0	1	0	4808 (3)	1 M
0	1	1	2404	500 k
1	0	0	1202	250 k

(ii) ϕ ÷div = 4 MHz

CS2	CS1	CS0	Asynchronous (start-stop synchronization)	CLK Synchronous
0	0	0	19230 (1)	Prohibited setting
0	0	1	Prohibited setting	
0	1	0	9615 (3)	2 M
0	1	1	4808	1 M
1	0	0	2404	500 k

For these cases, the baud rate is calculated as follows.

(1)	(ϕ ÷div)/(16 × 13)
(2)	(ϕ ÷div)/2^6
(3)	(ϕ ÷div)/(16 × 13 × 2)

(Where, ϕ is the machine cycle.)

b) Internal timer

Selecting the internal timer by setting "110" to CS2 to 0 sets the 16-bit timer (timer 4) to operate in reload mode. The baud rate is calculated as follows.

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

N: Count clock source for timer
n: Timer reload value

Table 2.4.5 lists the relationship between baud rate and reload value (hexadecimal) for a machine cycle of 7.3728 MHz.

Table 2.4.5 Baud Rate and Reload Value

Reload Value / Baud Rate	N = 2^1 (Machine cycle divided by 2)	N = 2^3 (Machine cycle divided by 8)
38400	2	-
19200	5	-
9600	11	2
4800	23	5
2400	47	11
1200	95	23
600	191	47
300	383	95

When the internal timer (16-bit timer 0) is selected as the baud rate clock source, the TOT0 output of 16-bit timer 0 is connected inside the controller. Therefore, the TOT0 output of 16-bit timer 0 does not have to be connected externally to the UART external clock input pin (SCK). Also, if the output pin of timer 0 is not used elsewhere, the pin can be used as an I/O port pin.

c) External clock

The baud rate is calculated as follows when the external clock is selected by setting "111" to CS2 to 0. Here, f is the frequency.

CLK asynchronous (start-stop synchronization)	f/16
CLK synchronous	f

The maximum external clock frequency (f) is 1 MHz.

(3) CLK asynchronous (start-stop synchronization) mode

a) Transfer data format

The UARTs only handle Non-Return-to-Zero (NRZ) type data. Figure 2.4.3 shows the data format.

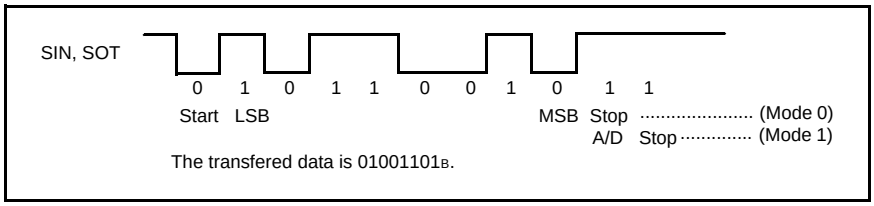


Fig. 2.4.3 Transfer Data Format (Mode 0 and 1)

As shown in Figure 2.4.3, the transfer data always starts from the start bit (L level data), the specified number of data bits are transmitted LSB-first, then transmission ends with the stop bit (H level data). Input a continuous clock signal if external clock operation is selected.

The number of data bits can be set to 7 or 8 bits in normal mode (mode 0). In multiprocessor mode (mode 1), the number of data bits must be 8. Also parity cannot be used in multiprocessor mode. Instead, the A/D bit is always added.

b) Receive operation

Reception operates continuously when the RXE bit (bit 9) of the SCR register is "1".

After a start bit is received on the receive line, one frame of data is received in the data format specified by the SCR register. After reception of the frame is complete, an error flag is set if any error occurred, then the RDRF flag (bit 12 of the SSR register) is set. At this time, a receive interrupt is sent to the CPU if the RIE bit (bit 9) of the SSR register is set to "1". Check the flags in the SSR register, then read the SIDR register if reception was normal or perform any required processing if an error occurred.

Reading the SIDR register clears the RDRF flag.

c) Transmit operation

Write the transmit data to the SODR register when the TDRE flag (bit 11) in the SSR register is "1". The UART transmits the data if the TXE bit (bit 8) of the SCR register is "1".

The TDRE flag is set again when the data set in the SODR register is loaded to the transmit shift register to start transmission. The next transmit data can now be set. If the TIE bit (bit 8) of the SSR register is set to "1", a transmit interrupt is sent to the CPU requesting input of the next transmit data to SODR.

Setting data to the SODR register clears the TDRE flag.

(4) CLK synchronous mode

a) Transfer data format

The UARTs only handle Non Return to Zero (NRZ) type data. Figure 2.4.4 shows the relationship between the transmit/receive clock and the data.

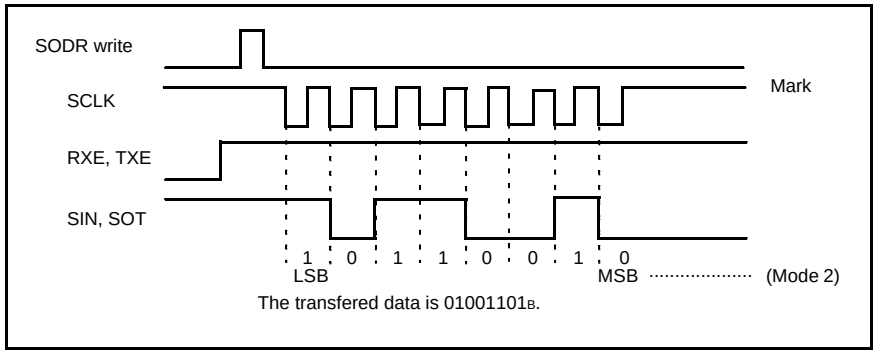


Fig. 2.4.4 Transfer Data Format (Mode 2)

When an internal clock is selected (dedicated baud rate generator or internal timer), the synchronizing clock for data reception is automatically generated when transmitting data.

When an external clock is selected, exactly one byte of clock signal must be supplied after checking that data is present in the transmit data buffer (SODR register) of the transmitter UART (the TDRE flag is "0"). Note that SCLK must be at the mark level before starting transmission and after transmission is complete.

Only 8-bit data can be used and parity is not available. As no start or stop bits are used, the only error that can be detected is an overrun error.

b) Initialization

The following lists the settings for each control register when using CLK synchronous mode.

(1)SMR register

MD1, MD0:	"10"
CS2, CS1, CS0:	Specify the clock input.
SCKE:	When the dedicated baud rate generator or internal timer is used: "1" When an external clock is used: "0"
SOE:	When transmitting: "1" When receiving only: "0"

(2)SCR register

PEN:	"0"
P, SBL, A/D:	These bits have no meaning.
CL:	"1"
REC:	"0" (for initialization)
RXE, TXE:	Set at least one of these to "1".

(3)SSR register

RIE:	Set "1" to use interrupts. Set "0" for no interrupts.
TIE:	"0"

c) Starting communications

Writing to the SODR register starts communications. Note that temporary data must be written to the SODR register, even if not performing transmission.

d) Ending communications

The RDRF flag in the SSR register changes to "1" to confirm the end of communications. Check the ORE bit in the SSR register to determine whether communication was performed correctly.

(5) Interrupt generation and flag set timings

The UARTs have five flags and two interrupts sources.

The five flags are PE, ORE, FRE, RDRF, and TDRE. PE indicates a parity error, ORE an overrun error, and FRE a framing error. These flags are set if a receive error occurs and are cleared by writing "0" to RED in the SCR register. RDRF is set when receive data is loaded into the SIDR register. Reading SIDR clears RDRF. Note that the parity detect function is not available in mode 1 and the parity and framing error detect functions are not available in mode 2. TDRE is set when the SODR register becomes empty and available for writing. Writing to the SODR register clears TDRE.

The two interrupts are the receive and transmit interrupts. For reception, the PE, ORE, FRE, and RDRF flags request an interrupt. For transmission, the TDRE flag requests an interrupt. The following describes the interrupt flag set timings for each operating mode.

a) Receive operation in mode 0

The PE, ORE, FRE, and RDRF flags are set and an interrupt request to the CPU generated when the final stop bit is detected indicating the end of reception. The data in SIDR is invalid when either the PE, ORE, or FRE bit is active.

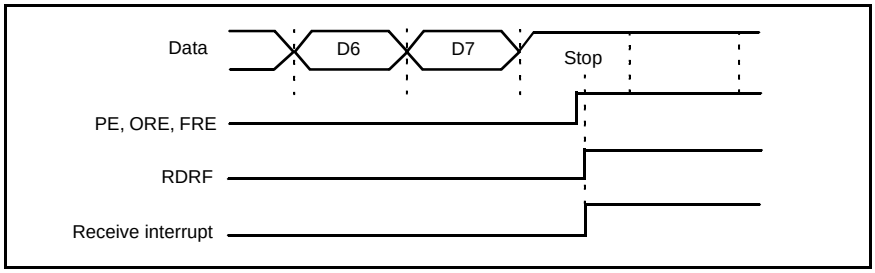


Fig. 2.4.5 ORE, FRE, and RDRF Set Timing (Mode 0)

b) Receive operation in mode 1

The ORE, FRE, and RDRF flags are set and an interrupt request to the CPU generated when the final stop bit is detected indicating the end of reception. As only 8 data bits can be received, the final ninth bit containing the address/data indicator is not valid. The data in SIDR is invalid when either the ORE or FRE bit is active.

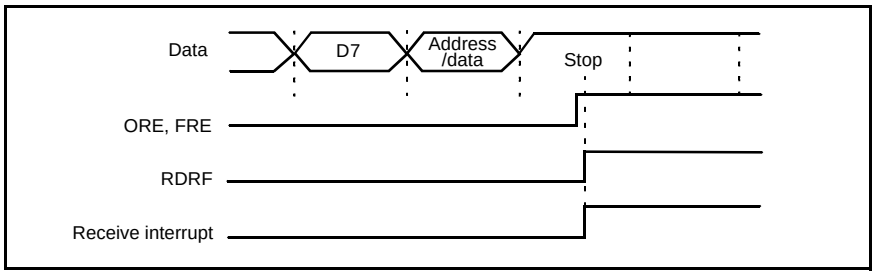


Fig. 2.4.6 ORE, FRE, and RDRF Set Timing (Mode 1)

c) Receive operation in mode 2

The ORE and RDRF flags are set and an interrupt request to the CPU generated when the final data bit (D7) is detected indicating the end of reception. The data in SIDR is invalid when the ORE bit is active.

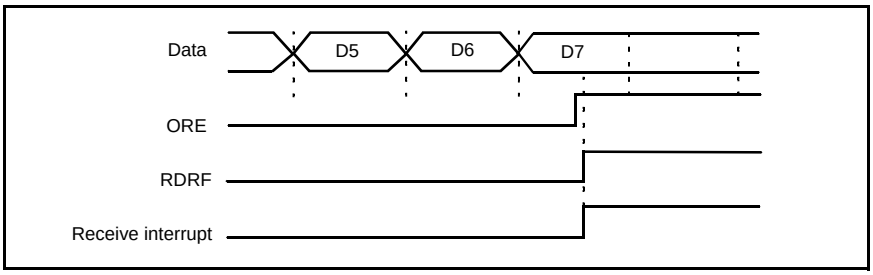


Fig. 2.4.7 ORE and RDRF Set Timing (Mode 2)

d) Transmit operation in mode 0 and mode 1

Writing data to the SODR register clears TDRE. TDRE is set and an interrupt request to the CPU generated when the data written in the SODR register is transferred to the internal shift register and the next data is able to be written to SODR. Writing "0" to TXE (also to RXE in mode 2) in the SCR register during transmission disables transmission operation after TDRE in the SSR register goes to "1" and the transmit shifter stops. If "0" is written to TXE (also to RXE in mode 2) in the SCR register while transmission is in progress, the data written to the SODR register before transmission was halted is transmitted.

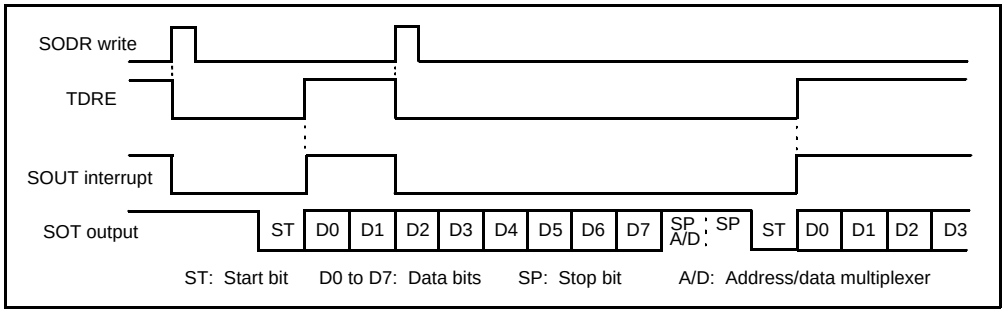


Fig. 2.4.8 TDRE Set Timing (Mode 0, 1)

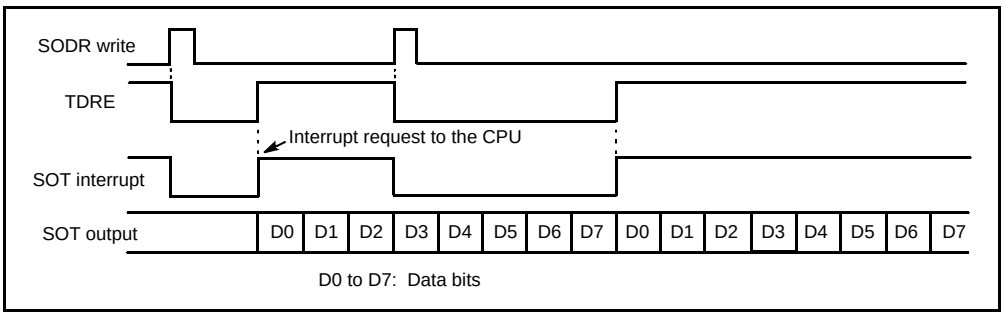


Fig. 2.4.9 TDRE Set Timing (Mode 2)

(6) EI²OS (Extended intelligent I/O service)

See Section 3.3.2 "EI²OS" for details on EI²OS.

2.4.5 Cautions

Set the communication mode when operation is halted. Data transmitted or received during mode setting cannot be guaranteed.

2.4.6 Application Example

Mode 1 is used when a number of slave CPUs are connected to a host CPU (see Figure 2.4.10). This resource only supports the host-side communication interface.

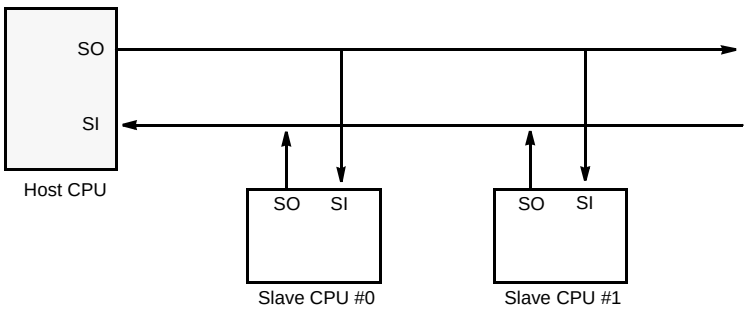


Fig. 2.4.10 Example System Configuration Using Mode 1

Communication starts with the host CPU transmitting address data. Address data is data transmitted with the A/D bit in the SCR register set to "1". The address selects the slave CPU with which to communicate and enables communication with the host CPU. Normal data is data transmitted with the A/D bit in the SCR register is set to "0". Figure 2.4.11 shows a flow chart of operation in this mode.

As the parity check function is not available in this mode, set the PEN bit in the SCR register to "0".

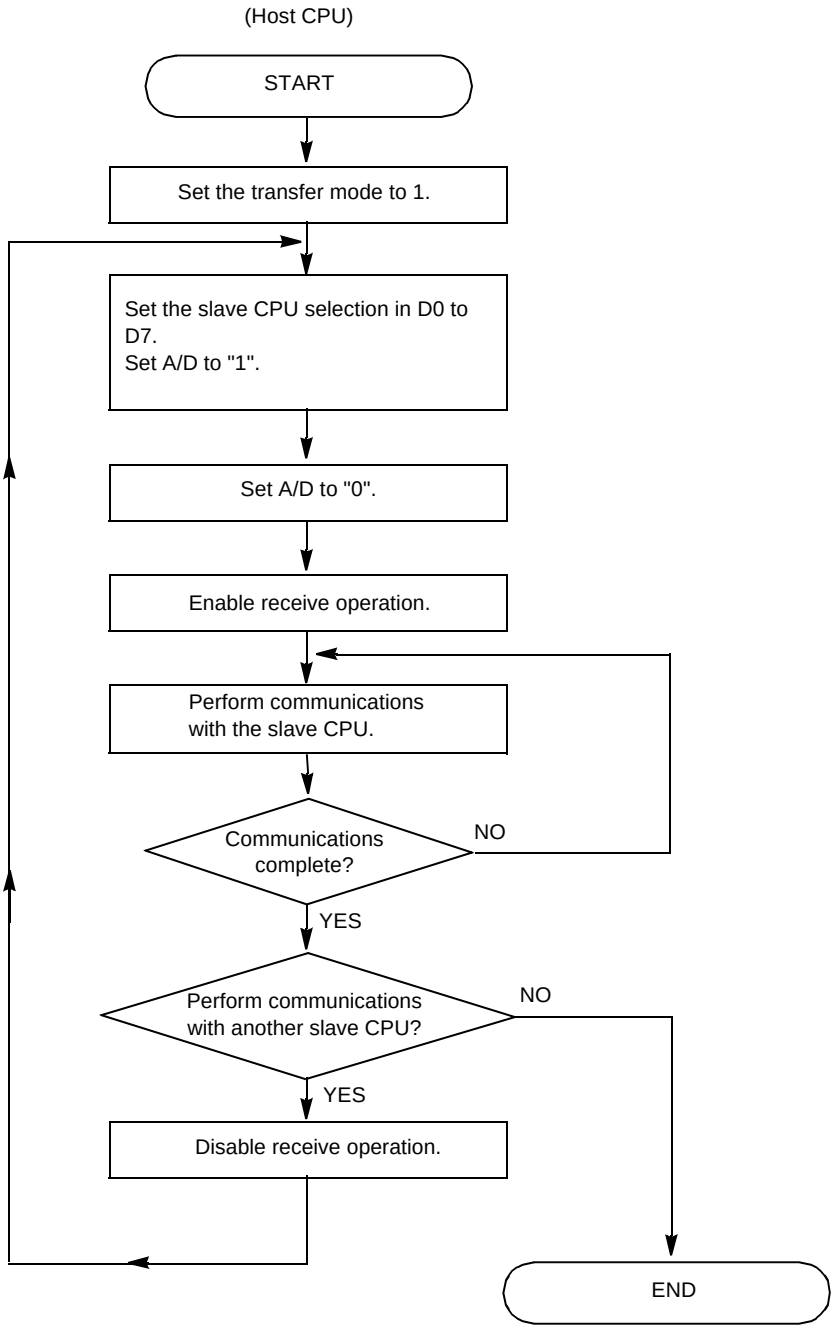


Fig. 2.4.11 Communication Flowchart for Mode 1 Operation

2.5 10-Bit 8-Input A/D Converter (With 8-Bit Resolution Mode)

The 10-bit 8-input A/D converter converts analog input voltages to digital values. The A/D converter has the following features.

- Conversion time: Minimum of 6.13 µs per channel (98 machine cycles for a 16 MHz machine clock - including sample and hold time)
- Sample and hold time: Minimum of 3.75 µs per channel (60 machine cycles for a 16 MHz machine clock)
- Uses RC-type successive approximation conversion with a sample-and-hold circuit.
- 10-bit or 8-bit resolution
- Eight program-selectable analog input channels

  Single conversion mode: Selectively convert a single channel.
  Scan conversion mode: Continuously convert multiple channels. Maximum of 8 program-selectable channels.
  Continuous conversion mode: Repeatedly convert specified channels.
  Stop conversion mode: Convert one channel then halt until the next activation. (Enables synchronization of the conversion start timing.)
- An A/D conversion completion interrupt request to the CPU can be generated on the completion of A/D conversion. This interrupt can activate I²OS to transfer the result of A/D conversion to memory and is suitable for continuous processing.
- Activation by software, external trigger (falling edge), or timer (rising edge) can be selected.

2.5.1 Registers

Control status register (upper)

	15	14	13	12	11	10	9	8	⇦ Bit No.
Address: 00002DH	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	Reserved	ADCS1
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(-)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0	(0)	

Control status register (lower)

	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: 00002CH	MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	ADCS0
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 register (upper)

	15	14	13	12	11	10	9	8	⇦ Bit No.
Address: 00002FH	S10	-	-	-	-	-	D9	D8	ADCR1
Read/write ⇨	(R/W)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(X)	(X)	

Data register (lower)

	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: 00002EH	D7	D6	D5	D4	D3	D2	D1	D0	ADCR0
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

2.5.2 Block Diagram

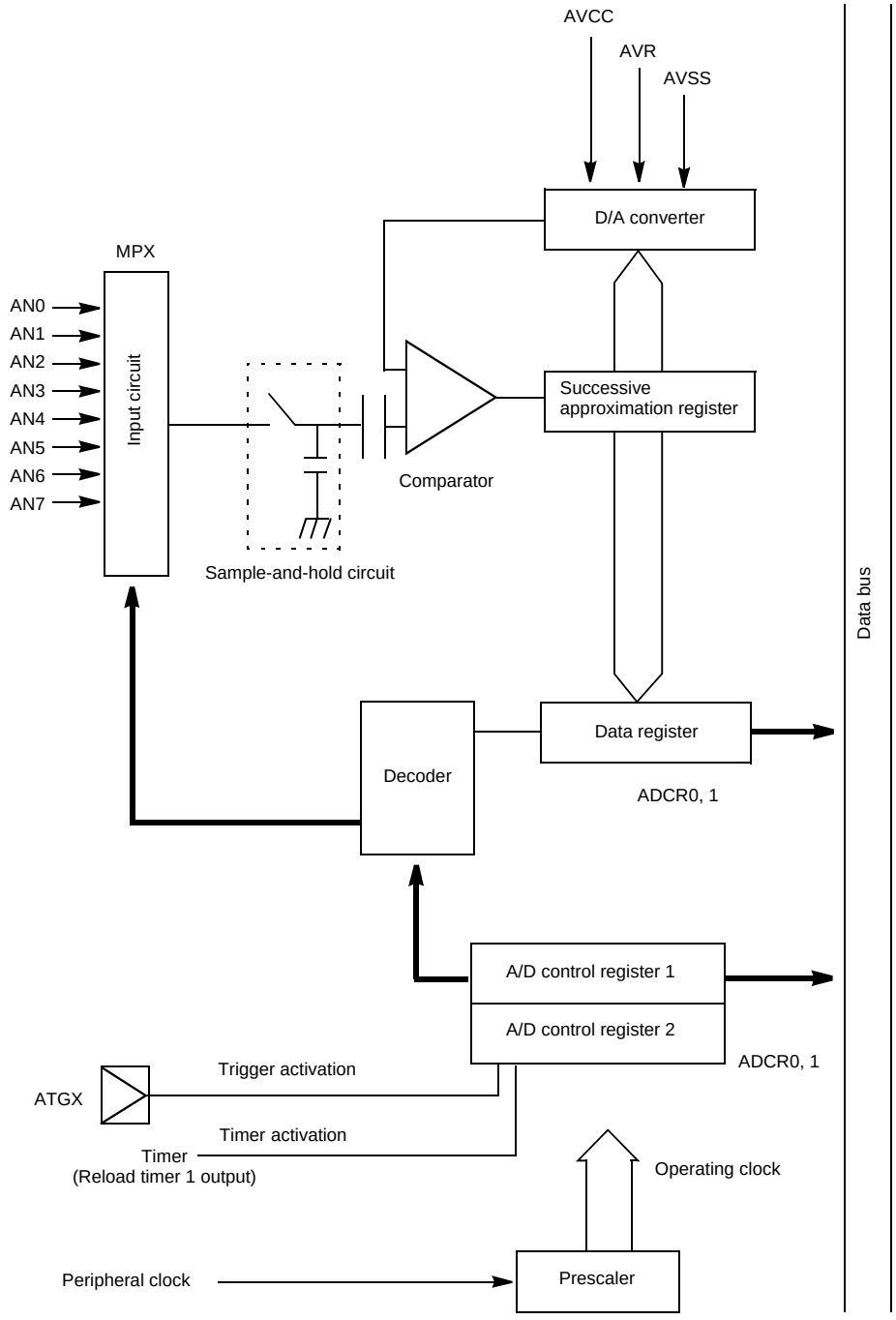
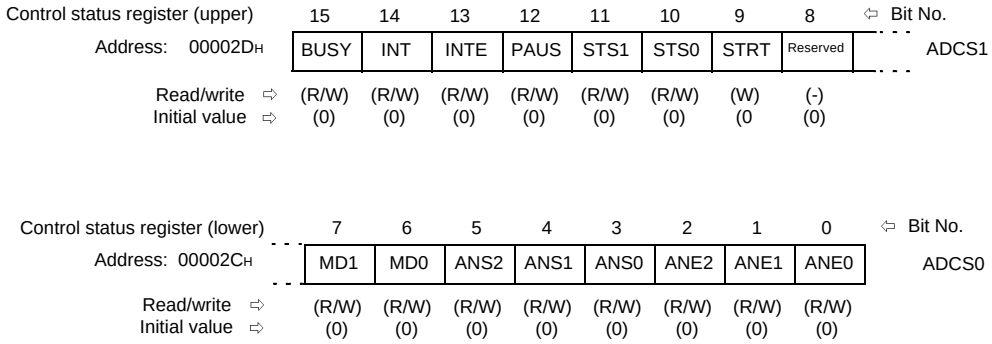


Fig. 2.5.1 Block Diagram

2.5.3 Register Details

(1) ADCS1, 0 (Control status register)

■ Register layout



■ Register contents

Note 1: This register is used for control and status indication for the A/D converter. Do not modify ADCS0 during A/D conversion.

Note 2: Do not access this register using read-modify-write instructions as this can cause misoperation.

■ Bit meanings

[Bit 15] BUSY (Busy flag and stop)

On reading: A/D converter operation indication bit. Set on activation of A/D conversion and cleared on completion.
On writing: Writing "0" to this bit during A/D conversion forcibly terminates conversion. Use to forcibly terminate in continuous and stop modes.

Writing "1" to this bit has no effect. RMW instructions read a value of "1". Cleared on the completion of A/D conversion in single conversion mode. In continuous and stop mode, the flag is not cleared until conversion is terminated by writing "0". Initialized to "0" by a reset.

Note: Do not specify forcible termination and software activation (BUSY="0" and STRT="1") at the same time.

[Bit 14] INT (Interrupt)

Data indication bit. This bit is set when conversion data is stored in ADCR. When INTE (bit 13) is "1", an interrupt request is generated when this bit is set. When I²OS activation is enabled, I²OS is activated. Writing "1" has no meaning. The bit is cleared by I²OS data transfer or by writing "0" to the bit from the CPU. The bit is initialized to "0" by a reset.

Note: Only clear this bit (by writing "0" from the CPU) while conversion is halted.

[Bit 13] INTE (Interrupt enable)

This bit enables or disables the conversion completion interrupt.
0: Disable interrupt
1: Enable interrupt

Always set this bit when using I²OS. Generation of an interrupt request activates I²OS.
Initialized to "0" by a reset.

[Bit 12] PAUS (A/D converter pause)

This bit is set when A/D conversion temporarily halts.

The A/D converter has only one register to store the conversion result. Therefore, in continuous conversion mode, the previous conversion result is lost when the next conversion result is stored in the register if the previous data has not been read by the CPU. To avoid this problem when using continuous conversion mode, use I²OS to transfer the conversion result to memory automatically after each conversion is complete. However, there may be cases when transfer of the conversion data does not occur before completion of the next conversion (for example, if multiple interrupts occur). This bit is provided to deal with such cases. The bit is set from the time conversion completes until the content of the data register is transferred by I²OS. This halts A/D conversion during this period and prevents the next conversion data from being stored in the register. The A/D converter then automatically restarts after completion of data transfer by I²OS. This bit is only meaningful when I²OS is used. The bit is initialized to "0" by a reset.

[bits 11, 10] STS1, STS0 (Start source select)

These bits select the A/D activation source. Initialized to "00" by a reset.

Table 2.5.1 STS Bit Settings

STS1	STS0	Function
0	0	Software activation
0	1	Software activation and external trigger pin activation
1	0	Software activation and timer activation
1	1	Software activation, external trigger pin activation, and timer activation

In multiple-activation modes, the first activation to occur starts A/D conversion. The activation source changes immediately on writing to the register. Therefore, when switching activation modes during A/D operation, switch the mode while the new activation source is not active.

Setting the external trigger pin as an activation source when the external trigger input level is "L" may start A/D conversion.

[Bit 9] STRT (Start)

Writing "1" to this bit starts A/D operation. Write "1" again to restart conversion. Restarting does not function in stop mode. Initialized to "0" by a reset.

Note: Do not specify forcible termination and software activation (BUSY="0" and STRT="1") at the same time.

[Bit 8] Test bit

Test bit. Always write "0" to this bit.

[Bits 7, 6] MD1, MD0 (A/D converter MoDe set)

These bits set the operation mode. Table 2.5.2 lists the available modes.

Table 2.5.2 MD Bit Settings

MD1	MD0	Operation Mode
0	0	Single conversion mode. Re-activation during operation is always available. (Initial value)
0	1	Single conversion mode. Re-activation during operation is not available.
1	0	Continuous conversion mode. Re-activation during operation is not available.
1	1	Stop mode. Re-activation during operation is not available.

Single conversion mode: Performs one A/D conversion for each channel from the start to the end channel, then halts. ANS2 to ANS0 specifies the start channel and ANE2 TO ANE0 specifies the end channel.

Continuous conversion mode: Repeatedly performs A/D conversion for the start to end channels. ANS2 to ANS0 specifies the start channel and ANE2 to ANE0 specifies the end channel.

Stop mode: Performs one A/D conversion each time for the start to end channels. Halts temporarily after each conversion and re-activates when a trigger occurs. ANS2 to ANS0 specifies the start channel and ANE2 to ANE0 specifies the end channel.

- When A/D conversion is activated in continuous mode or stop mode, conversion operates continuously until stopped by the BUSY bit.
-
- Write "0" to the BUSY bit to stop conversion.
- When re-activation is not available in single, continuous, or stop mode, this applies to activation by timer, external trigger, and software.

[Bits 5, 4, 3] ANS2, ANS1, ANS0 (ANalog Start channel set)

These bits set the start channel for A/D conversion. Activating the A/D converter starts A/D conversion from the channel specified by these bits.

Table 2.5.3 Conversion Start Channel Setting by the ANS 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

Reading these bits during A/D conversion reads the channel number currently being converted. Reading these bits while A/D conversion is halted in stop mode reads the most recently converted channel. Initialized to "000" by a reset.

[Bits 2, 1, 0] ANE2, ANE1, ANE0 (ANalog End channel set)

These bits set the end channel for A/D conversion. Activating the A/D converter performs A/D conversion up to the channel specified by these bits.

Table 2.5.4 Conversion End Channel Setting by the ANE Bits

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

Setting the same channel as ANS2 to ANS0 specifies conversion for that channel only. In continuous or stop mode, conversion is performed up to the channel specified by these bits. Conversion then starts again from the start channel specified by ANS2 to ANS0. If ANS > ANE, conversion starts with the channel specified by ANS, continues up to AN7, starts again from AN0, and ends with the channel specified by ANE. The following shows an example.

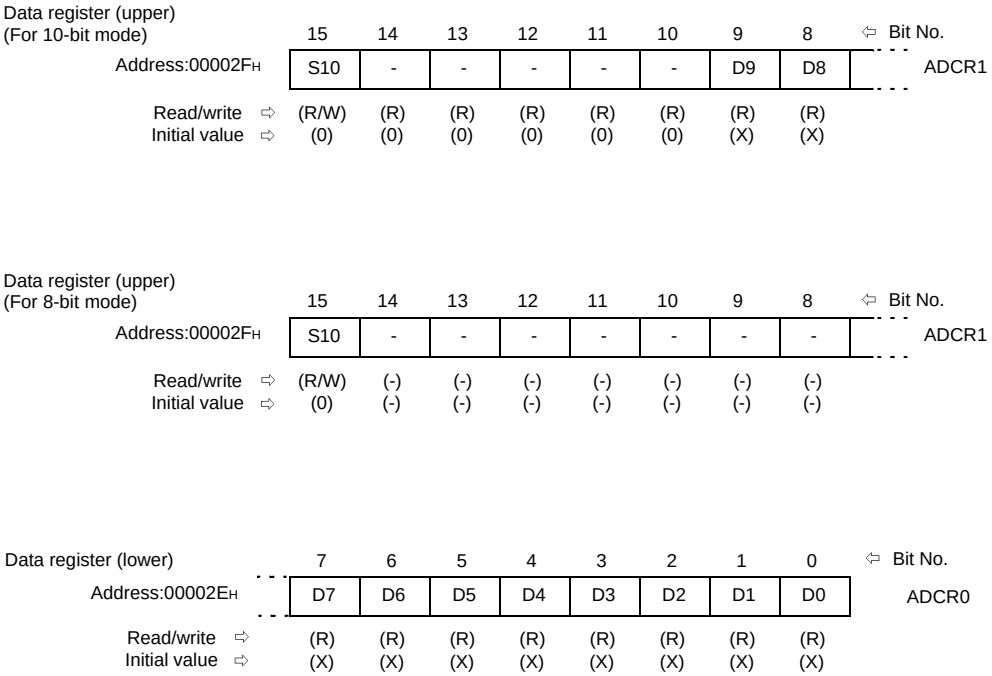
Example: Channel setting ANS = 6ch, ANE = 3ch, single conversion mode

Operation Conversion channel 6ch → 7ch → 0ch → 1ch → 2ch → 3ch

Initialized to "000" by a reset.

(2) ADCR1, ADCR0 (Data registers)

■ Register layout



■ Register contents

These registers store the digital result of A/D conversion. The resolution of the conversion result is 10 bits when bit S10 is "0" and 8 bits when bit S10 is "1". ADCR1 stores the upper 2 bits of the conversion result and ADCR0 stores the lower 8 bits. The D9 to D0 value is updated at the completion of every conversion. The registers normally hold the previous conversion value. Other than S10, the register values after a reset are indeterminate. S10 is initialized to "0" by a reset.

Reading bits 15 to 10 of ADCR1 always returns "0" in 10-bit mode. In 8-bit mode, all contents of ADCR1 are undefined and the result is stored in ADCR0.

Only write to S10 before conversion, when A/D operation is halted. Writing to S10 after conversion causes the contents of ADCR to become indeterminate.

2.5.4 Operation

The A/D converter operates using the successive approximation method with 10 or 8-bit resolution. As only one 16-bit register is provided to store conversion results, the previous conversion data is lost each time conversion completes. Therefore, I^2OS must be used to successively transfer conversion data to memory when using continuous conversion mode.

The following describes the operating modes.

■ Single conversion mode

In this mode, the A/D converter converts each analog input channel from the start channel specified by ANS to the end channel specified by ANE. Conversion completes on reaching the end channel and A/D operation halts. When the start and end channels are the same (ANS = ANE), conversion is only performed for the channel specified in ANS.

Example 1: ANS = "000", ANE = "011"
⇨ Start → AN0 → AN1 → AN2 → AN3 → End

Example 2: ANS = "010", ANE = "010"
⇨ Start → AN2 → End

● I^2OS activation example in single conversion mode

- Convert analog inputs AN1 to AN3, then end.
- Sequentially transfer the conversion data to locations 200H to 205H.
- Activate by software.
- Maximum interrupt priority level

❑ Setting I^2OS

▷Set ICR in the interrupt controller.

```
MOV ICR10, #08H .....①
```

▷Set the I^2OS descriptor.

```
MOV BAPL, #00H ..... ②
MOV BAPM, #02H.....③
MOV BAPH, #00H .....④
MOV ISCS, #18H .....⑤
MOV IOAL, #2EH.....⑥
MOV IOAH, #00H.....⑥
MOV DCTL, #03H .....⑦
MOV DCTH, #00H.....⑦
```

❑ Set the A/D converter.

```
MOV ADCS0, #0BH.....⑧
MOV ADCS1, #A2H .....⑨
```

❑ Other processing

```
:
:
:
```

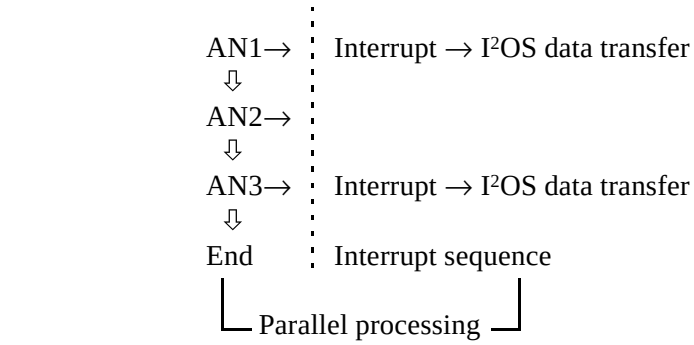
❑ I^2OS completion interrupt sequence

```
MOV ADCS1,#80H
RETI ..... ⑩
```

(1) Set maximum interrupt priority, specify I^2OS activation by the interrupt, and set the descriptor address.
(2)(3)(4) Set the destination address for the conversion data.
(5) Set word data transfer, increment the destination address after each transfer, and transfer from I/O to memory.
(6) Set the A/D converter result register.
(7) Perform I^2OS data transfer three times (same as the number of conversions).
(8) Set single conversion mode, start channel AN1, and end channel AN3.

(9) Specify software activation. Start A/D conversion.
(10) Return from interrupt.

ICR10: Interrupt control register
BAPL: Buffer address pointer (lower)
BAPM: Buffer address pointer (middle)
BAPH: Buffer address pointer (upper)
ISCS: I^2OS status register
I/OAL: I/O address register (lower)
I/OAH: I/O address register (upper)
DCTL: Data counter (lower)
DCTH: Data counter (upper)



■ Continuous conversion mode

In this mode, the A/D converter converts each analog input channel from the start channel specified by ANS to the end channel specified by ANE. On reaching the end channel, A/D conversion operation returns to the ANS analog input and the cycle starts again. When the start and end channels are the same (ANS = ANE), conversion is performed continuously for the channel specified in ANS.

Example 1: ANS = "000", ANE = "011"
⇨ Start → AN0 → AN1 → AN2 → AN3 → AN0 ··➤ Repeat

Example 2: ANS = "010", ANE = "010"
⇨ Start → AN2 → AN2 → AN2 ··➤ Repeat

In continuous mode, conversion repeats continuously until "0" is written to the BUSY bit. Writing "0" to the BUSY bit forcibly terminates operation.

Forcibly terminating operation halts the A/D converter mid-conversion and the value being converted at the time cannot be obtained. The result in ADCR is the most recent conversion value prior to halting.

■ I²OS activation example in continuous conversion mode

- Convert analog inputs AN3 to AN5 and collect two conversion data for each channel.
- Sequentially transfer the conversion data to locations 600H to 60BH.
- Activate by an external edge input.
- Maximum interrupt priority level

❑ Setting I²OS

▷Set ICR in the interrupt controller.

```
MOV ICR10,#08H .....(1)
```

▷Set the I²OS descriptor.

```
MOV BAPL,#00H.....(2)
MOV BAPM,#06H.....(3)
MOV BAPH,#00H .....(4)
MOV ISCS,#18H .....(5)
MOV IOAL,#2EH.....(6)
MOV IOAH,#00H.....(6)
MOV DCTL,#06H .....(7)
MOV DCTH,#00H.....(7)
```

❑ Set the A/D converter.

```
MOV ADCS0,#9DH .....(8)
MOV ADCS1,#A4H .....(9)
```

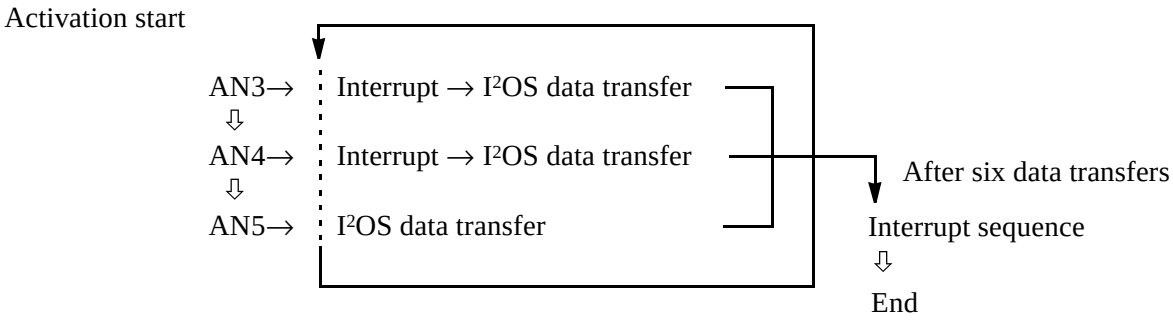
❑ Other processing

```
:
:
:
```

❑ I²OS completion interrupt sequence

```
MOV ADCS1,#80H .....(10)
RETI
```

(1) Set maximum interrupt priority, specify I²OS activation by the interrupt, and set the descriptor address.
(2)(3)(4)Set the destination address for the conversion data.
(5) Set word data transfer, increment the destination address after each transfer, and transfer from I/O to memory.
(6) Set the source address for data transfer.
(7) Perform I²OS data transfer six times (3ch × 2 data transfers).
(8) Set continuous conversion mode, start channel AN3, and end channel AN5.
(9) Specify external edge activation. Start A/D conversion.
(10) Return from interrupt.



■ Stop mode

In this mode, the A/D converter converts each analog input channel from the start channel specified by ANS to the end channel specified by ANE. However, conversion operation temporarily halts after each conversion. The next trigger received releases the temporary halt. On reaching the end channel specified by ANE, operation returns to the ANS analog input and the cycle starts again.

When the start and end channels are the same (ANS = ANE), conversion is only performed for the channel specified in ANS.

Example 1: ANS = "000", ANE = "011"
⇨ Start → AN0 → halt → activate → AN1 → halt → activate → AN2 → halt → activate → AN3 → halt → activate → AN0 ⇢ Repeat

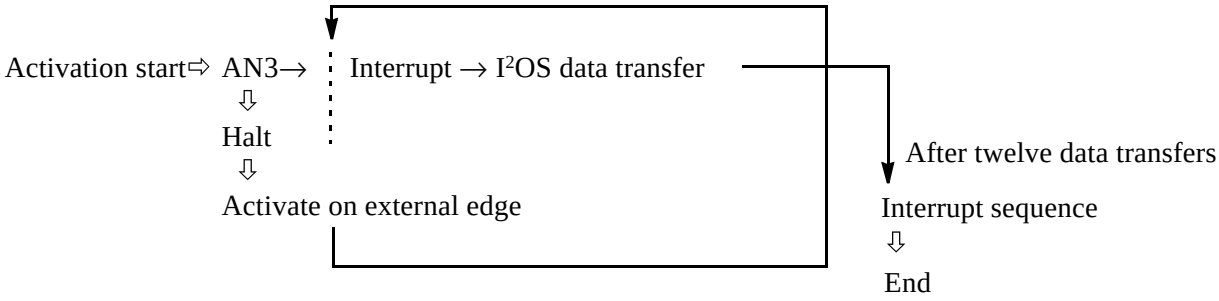
Example 2: ANS = "010", ANE = "010"
⇨ Start → AN2 → halt → activate → AN2 → halt → activate → AN2 ⇢ Repeat

The available activation sources are the sources set in STS1, 0. This mode can be used to synchronize the conversion start timing.

● I²OS activation example in stop mode
- Convert analog input AN3 twelve times at fixed intervals.
- Sequentially transfer the conversion data to locations 600H to 617H.
- Activate by an external edge input.
- Maximum interrupt priority level

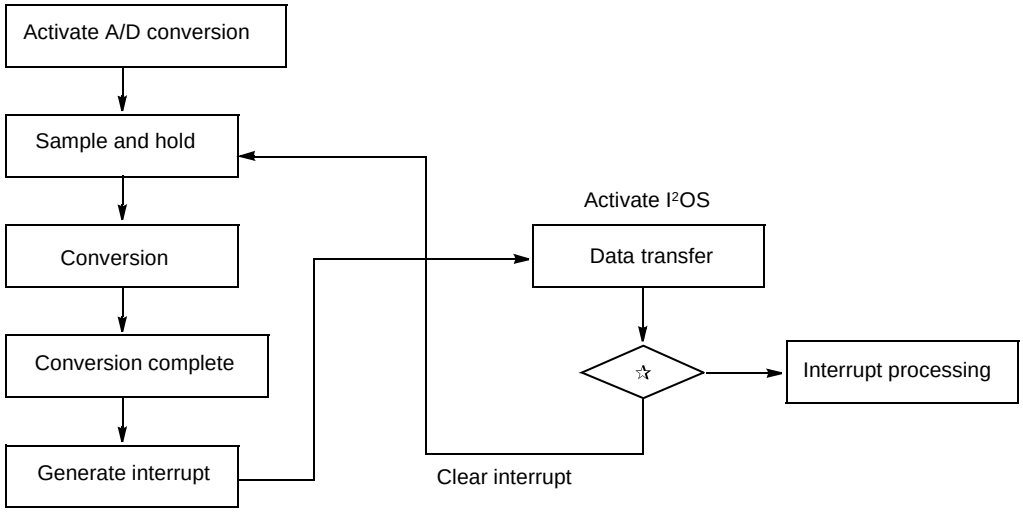
❑ Setting I²OS
▷Set ICR in the interrupt controller.
```
MOV ICR10,#08H .....①
```
▷Set the I²OS descriptor.
```
MOV BAPL,#00H.....②
MOV BAPM,#06H.....③
MOV BAPH,#00H .....④
MOV ISCS,#18H .....⑤
MOV IOAL,#2EH.....⑥
MOV IOAH,#00H.....⑥
MOV DCTL,#0CH.....⑦
MOV DCTH,#00H.....⑦
```
❑ Set the A/D converter.
```
MOV ADCS0,#DBH.....⑧
MOV ADCS1,#A4H .....⑨
```
❑ Other processing
```
:
:
:
```
❑ I²OS completion interrupt sequence
```
MOV ADCS1,#80H .....⑩
RETI
```

(1) Set maximum interrupt priority, specify I²OS activation by the interrupt, and set the descriptor address.
(2)(3)(4) Set the destination address for the conversion data.
(5) Set word data transfer, increment the destination address after each transfer, transfer from I/O to memory, and termination on a request from the resource.
(6) Set the source address for data transfer.
(7) Perform I²OS data transfer twelve time.
(8) Set stop mode, start channel AN3, and end channel AN3 (single-channel conversion).
(9) Specify external edge activation. Start A/D conversion.
(10) Return from interrupt.



■ Conversion operation using I²OS

The following shows an example using continuous mode. The example shows the steps from A/D conversion activation to conversion data transfer.



☆ Depends on the I²OS setting.

Fig. 2.5.2 I²OS Example

■ Conversion data protection function

A feature of the A/D converter is the conversion data protection function which uses I²OS to perform continuous conversion and save multiple data.

The A/D converter has only one conversion data register. Therefore, when continuous A/D conversion saves the new conversion data at conversion completion, this overwrites the previous conversion data. To protect the previous data, the A/D converter has a function to temporarily halt the A/D converter and prevent conversion data being written to the register (even if conversion has already completed) until the previous data has been transferred to memory by I²OS.

The temporary halt releases after I²OS transfers the data to memory.

If the previous data has already been transferred, the A/D converter continues without a temporary halt.

Note: This function depends on the INT and INTE bits in ADCS1.

The data protect function only operates when interrupts are enabled (INTE = "1").

The function does not operate when interrupts are disabled (INTE = "0"). In this case, performing continuous A/D conversion continuously overwrites the register data with new conversion results. Note that the INT bit is not cleared when interrupts are enabled (INTE = "1") but I²OS is not used. As a result, the data protection function causes the A/D converter to temporarily halt. In this case, use the interrupt sequence to clear the INT bit and release the halt.

The A/D converter re-starts if interrupts are disabled during I²OS operation when the A/D converter is temporarily halted. This may result in the conversion data register contents changing before data transfer occurs.

Also, the data waiting to be written to the register is lost if an activation is received while the A/D converter is temporarily halted.

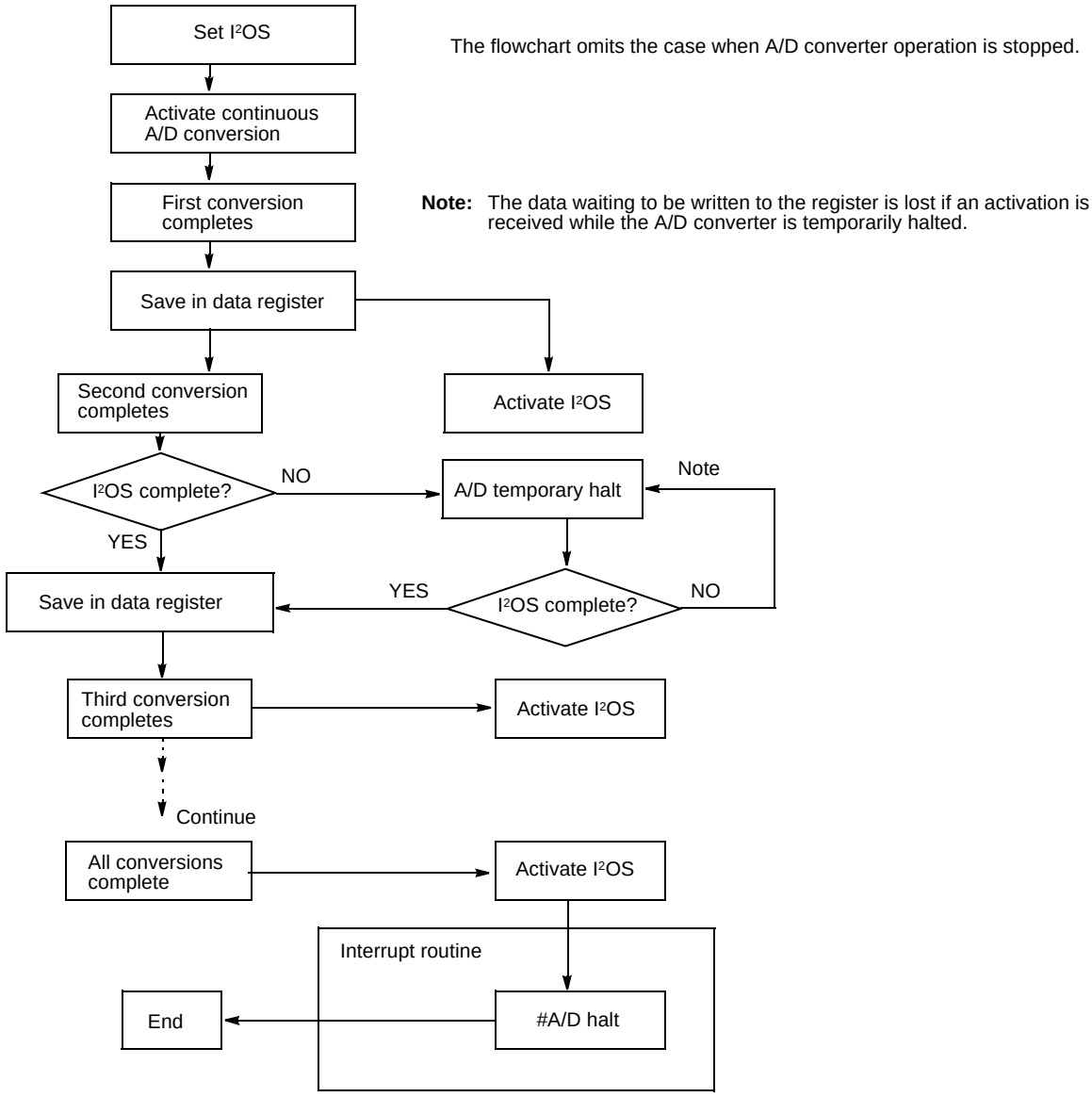


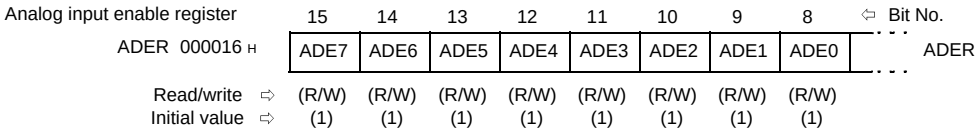
Fig. 2.5.3 Data Protection Function Flowchart Using I²OS

Note: When setting the A/D converter to be activated by an external trigger or internal timer, set the external trigger or internal timer input value to inactive when setting the start source select bits (STS1, 0) in the ADCS1 register. If the input is active, the A/D converter may start to operate.

Set ADTRG = "1" to the input state and internal timer (16-bit reload timer 1) = "0" to the output state when setting STS1, 0.

2.5.5 Other Points to Note

Always set the corresponding ADER bit to "1" for each analog input pin used.



Controls each pin of port 6 as follows.

0: Port input mode
1: Analog input mode

Initialized to "1" by a reset.

Note: Set to analog input mode when inputting an analog signal. Inputting an intermediate level signal to a pin set as an input port causes a leak current to flow.

Note: The ATGX pin is shared with other functions. Unless done intentionally, halt input (INT7) to the pin from other functions.

2.6 PPG

This block contains the 8-bit reload timer module. The block performs PPG output in which the pulse output is controlled by the operation of the timer.

The hardware consists of two 8-bit down-counters, four 8-bit reload registers, one 16-bit control register, two external pulse output pins, and two interrupt outputs. The unit has the following functions.

- 8-bit PPG output in 2-channel independent operation mode: Two independent PPG output channels are available.
- 16-bit PPG output operation mode: One 16-bit PPG output channel is available.
- 8+8-bit PPG output operation mode: Variable-period 8-bit PPG output operation is available by using the output of ch0 as the clock input to ch1.
- PPG output operation: Outputs pulse waveforms with variable period and duty ratio.

  Can be used as a D/A converter in conjunction with an external circuit.

2.6.1 Registers

PPG0 operation mode control register

Address: ch0 000030H

Bit No.	7	6	5	4	3	2	1	0
	PEN0	-	POE0	PIE0	PUF0	PCM1	PCM0	Reserved
Read/write	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)
Initial value	(0)	(-)	(0)	(0)	(0)	(0)	(0)	(1)

PPGC0

PPG1 operation mode control register

Address: ch1 000031H

Bit No.	15	14	13	12	11	10	9	8
	PEN1	PCS1	POE1	PIE1	PUF1	MD1	MD0	Reserved
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)
Initial value	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(1)

PPGC1

Reload register H

Address: ch0 000035H
ch1 000037H

Bit No.	15	14	13	12	11	10	9	8
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)

PRLH

Reload register L

Address: ch0 000034H
ch1 000036H

Bit No.	7	6	5	4	3	2	1	0
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

PRLL

2.6.2 Block Diagram

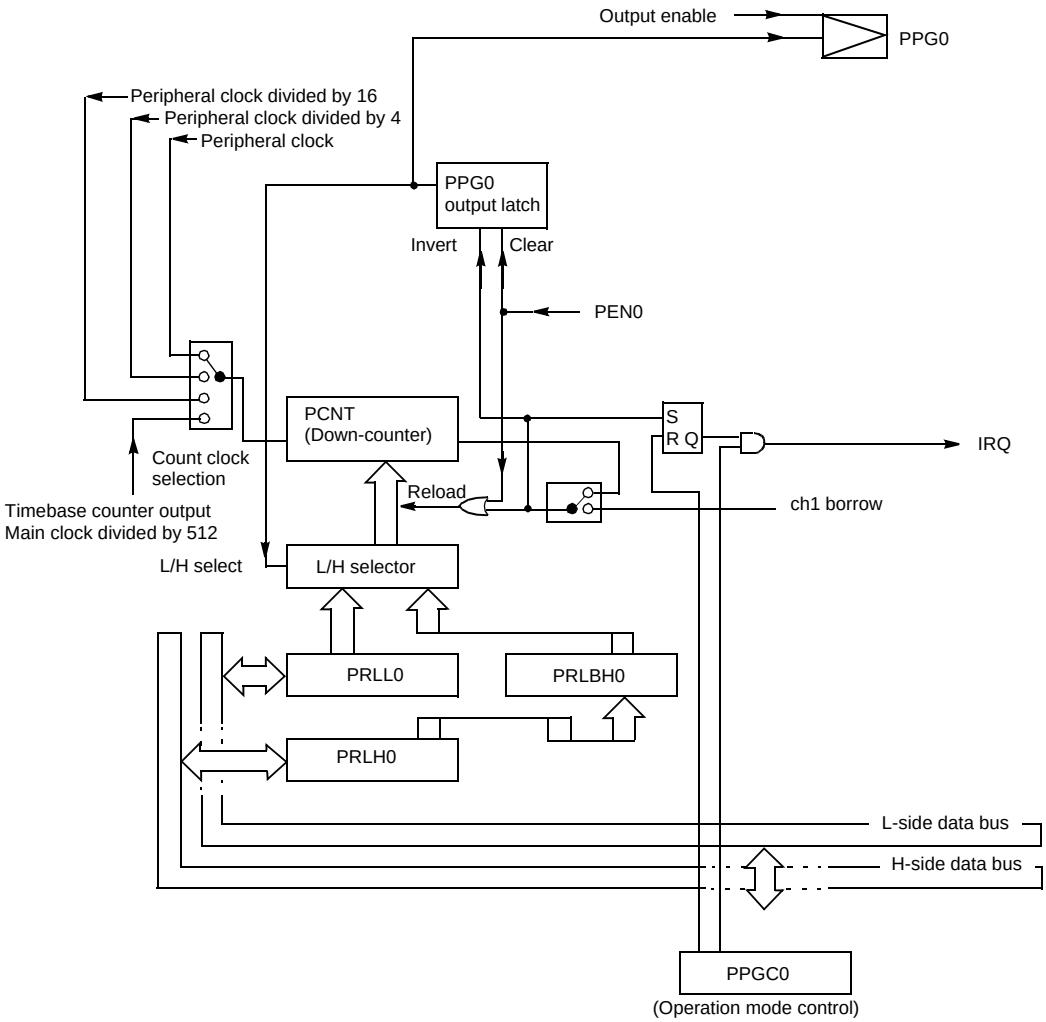


Fig. 2.6.1 Block Diagram of 8-Bit PPG ch0

2.6.2 Block Diagram

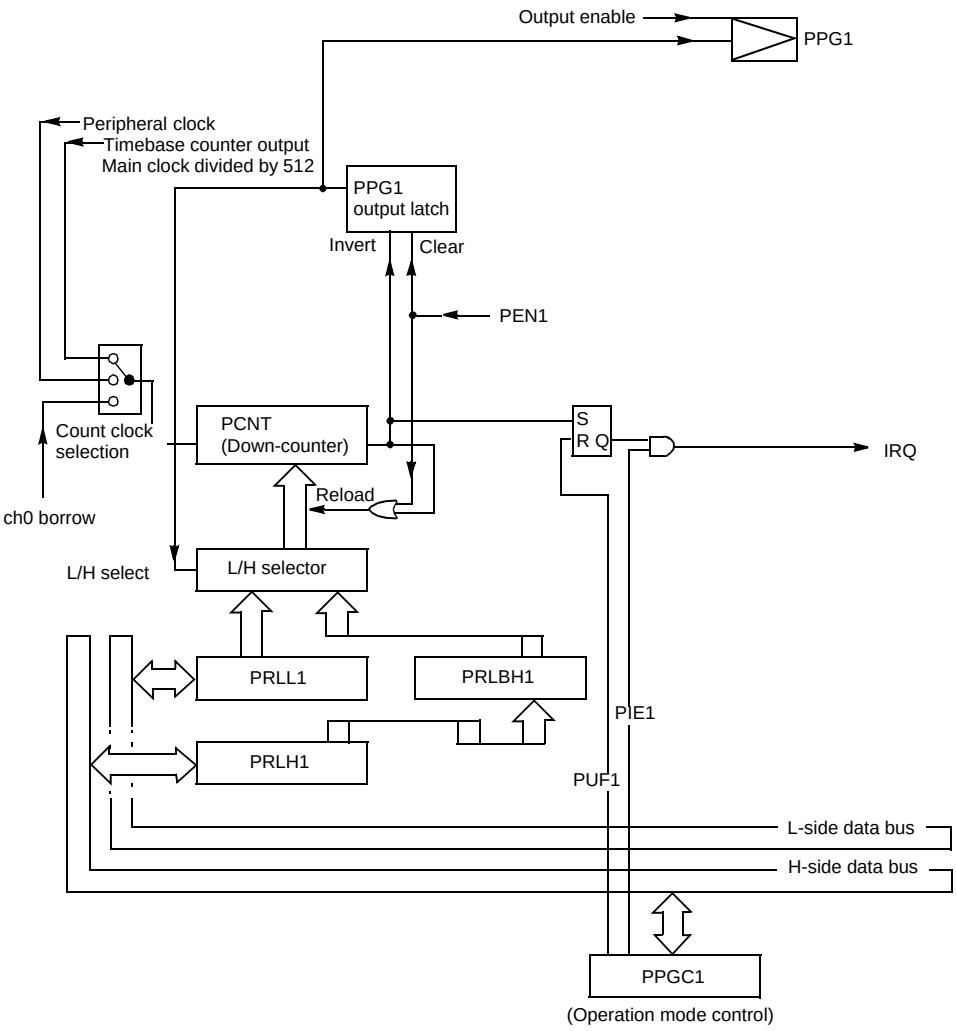


Fig. 2.6.2 Block Diagram of 8-Bit PPG ch1

2.6.3 Register Details

(1) PPGC0 (PPG0 operation mode control register)

PPG0 operation mode control register	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: ch0 000030H	PEN0	-	POE0	PIE0	PUF0	PCM1	PCM0	Reserved	PPGC0
Read/write ⇨	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	
Initial value ⇨	(0)	(-)	(0)	(0)	(0)	(0)	(0)	(1)	

A 6-bit control register that selects the operation mode of the block, controls pin output, selects the count clock, and controls the trigger.

[Bit 7] PEN0 (Ppg ENable): Operation enable bit
Starts PPG operation or selects the operation mode as follows.

PEN0	Operation State
0	Operation stopped (output held at "L" level)
1	PPG operation enabled

Writing "1" to this bit starts the PPG counting.
Initialized to "0" by a reset. Readable and writable.

[Bit 5] POE0 (Ppg Output Enable): PPG pin output enable bit
Controls the external pulse output pin (PPG0) as follows.

0	General-purpose port pin (pulse output disabled)
1	PPG0 = pulse output pin (pulse output enabled)

Initialized to "0" by a reset. Readable and writable.

[Bit 4] PIE0 (Ppg Interrupt Enable): PPG interrupt enable bit
Controls PPG interrupt enable as follows.

0	Interrupts disabled
1	Interrupts enabled

When this bit is "1", an interrupt request is generated when PUF0 becomes "1". No interrupt request is generated if this bit is "0".

Initialized to "0" by a reset. Readable and writable.

[Bit 3] PUF0 (Ppg Underflow Flag): PPG counter underflow bit
The meaning of the PPG counter underflow bit is as follows.

0	No PPG counter underflow detected.
1	PPG counter underflow detected.

In 2-channel 8-bit PPG mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" when an underflow occurs in ch0 (when the counter value changes from 00H to FFH). In single channel 16-bit PPG mode, this bit is set to "1" when an underflow occurs in ch1/ch0 (when the counter value changes from 0000H to FFFFH). Writing "0" to this bit changes the bit to "0". Writing "1" has no meaning. Reading returns "1" in read-modify-write instructions.

Initialized to "0" by a reset. Readable and writable.

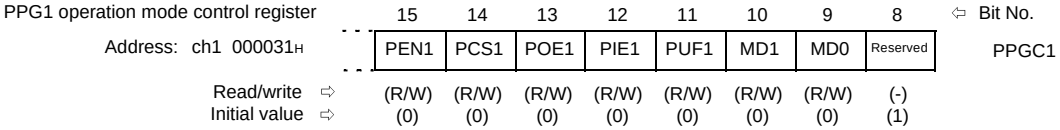
[Bits 2, 1] PCM1/0 (Ppg Count Mode): Count clock selection bits
Selects the operating clock for the down-counter as follows.

PCM1	PCM0	Operation Mode
0	0	Peripheral clock (62.5ns for a 16 MHz machine clock)
0	1	Peripheral clock/4 (250ns for a 16 MHz machine clock)
1	0	Peripheral clock/16 (1 µs for a 16 MHz machine clock)
1	1	Input clock from the timebase counter (128 µs for a 4 MHz source oscillator)

Initialized to "00" by a reset. Readable and writable.

[Bit 0] Reserved bit. Always set to "1" when setting PPGC0.

(2) PPGC1 (PPG1 operation mode control register)



A 7-bit control register that selects the operation mode of the block, controls pin output, selects the count clock, and controls the trigger.

[Bit 15] PEN1 (Ppg ENable): Operation enable bit
Starts PPG operation or selects the operation mode as follows.

PEN1	Operation State
0	Operation stopped (output held at "L" level)
1	PPG operation enabled

Writing "1" to this bit starts PPG counting.

Initialized to "0" by a reset. Readable and writable.

[Bit 14] PCS1 (Ppg Count Select): Count clock selection bit
Selects the operating clock for the down-counter as follows.

0	Peripheral clock (62.5ns for a 16 MHz machine clock)
1	Input clock from the timebase counter (128 µs for a 4 MHz source oscillator)

Initialized to "0" by a reset. Readable and writable.

Note: The PCS1 bit setting is ignored in 8-bit prescaler + 8-bit PPG mode and 16-bit PPG mode. In these modes, the count clock for ch1 of the PPG is received from ch0.

[Bit 13] POE1 (Ppg Output Enable): PPG pin output enable bit
Controls the external pulse output pin (PPG1) as follows.

0	General-purpose port pin (pulse output disabled)
1	PPG1 = pulse output pin (pulse output enabled)

Initialized to "0" by a reset. Readable and writable.

[Bit 12] PIE1 (Ppg Interrupt Enable): PPG interrupt enable bit
Controls PPG interrupt enable as follows.

0	Interrupts disabled
1	Interrupts enabled

When this bit is "1", an interrupt request is generated when PUF1 becomes "1". No interrupt request is generated if this bit is "0".

Initialized to "0" by a reset. Readable and writable.

[Bit 11] PUF1 (Ppg Underflow Flag): PPG counter underflow bit
The meaning of the PPG counter underflow bit is as follows.

0	No PPG counter underflow detected.
1	PPG counter underflow detected.

In 2-channel 8-bit PPG mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" when an underflow occurs in ch1 (when the counter value changes from 00H to FFH). In single channel 16-bit PPG mode, this bit is set to "1" when an underflow occurs in ch1/ch0 (when the counter value changes from 0000H to FFFFH). Writing "0" to this bit changes the bit to "0". Writing "1" has no meaning. Reading returns "1" in read-modify-write instructions.

Initialized to "0" by a reset. Readable and writable.

[Bits 10, 9] MD1, 0 (PPG count MoDe): Operation mode selection bits
Selects the operating mode of the PPG timer as follows.

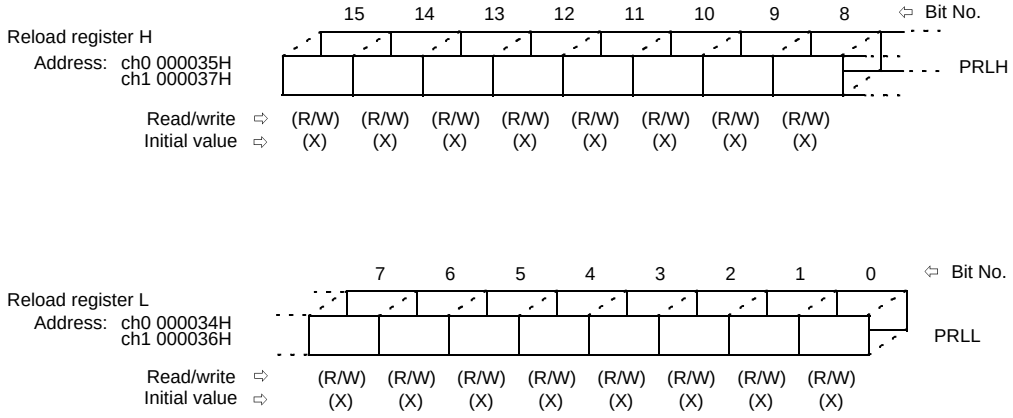
MD1	MD0	Operation Mode
0	0	2-channel independent 8-bit PPG mode
0	1	Single channel 8-bit prescaler + 8-bit PPG mode
1	0	Reserved (prohibited setting)
1	1	Single channel 16-bit PPG mode

Initialized to "00" by a reset. Readable and writable.

Note: Do not set these bits to "10".
Note: Do not set the PEN0 bit of PPGC0 and the PEN1 bit of PPGC1 to "01" when the MD bits are set to "01". Setting the PEN0 and PEN1 bits to "11" or "00" is recommended.
Note: When setting the MD bits to "11", write to PPGC0 and PPGC1 using a word write and set the PEN0 and PEN1 bits to "11" or "00" at the same time.

[Bit 8] Reserved bit. Always set to "1" when setting PPGC1.

(3) PRLL/PRLH (Reload registers)



Two 8-bit registers that store the reload values loaded to the down-counter PCNT. The registers are used as follows.

Register	Function
PRLL	Stores the L-side reload value
PRLH	Stores the H-side reload value

Both registers are readable and writable.

Note: The ch1 PPG waveform may be different for each cycle if different values are set in PRLL and PRLH for ch0 when using 8-bit prescaler + 8-bit PPG mode. Therefore, setting the same value in both ch0 registers (PRLL and PRLH) is recommended.

2.6.4 Operation

The block consists of two 8-bit PPG units. The two channels can be used independently (2-channel independent mode) or linked for 8-bit prescaler + 8-bit PPG mode or single-channel 16-bit PPG mode. This gives a total of three different operation modes.

Both 8-bit PPG units have L-side and H-side 8-bit reload registers (PRLL and PRLH). The values written to these registers are used alternately to reload the 8-bit down-counter (PCNT) for the L-side and H-side counts respectively. The PCNT value is down-counted with each count clock until a borrow triggers a reload and inverts the output pin (PPG) level. As a result of this operation, the output pin (PPG) outputs pulses with an H-width and L-width determined by the reload register values.

Operation is started or re-started by writing to a register bit.

The following table shows the relationship between the reload operation and pulse output.

Table 1 Relationship Between Reload Operation and Pulse Output

Reload Operation	Output Change at Pin
PRLH ⇨ PCNT	PPG0/1 [0 ⇨ 1] ↑ Rise
PRLL ⇨ PCNT	PPG0/1 [1 ⇨ 0] ↓ Fall

Also, when bit 4 (PIE0) of PPGC0 or bit 12 (PIE1) of PPGC1 is "1", an interrupt request is output when the corresponding borrow occurs as the counter value changes from 00H to FFH. (For 16-bit PPG mode, the borrow occurs as the counter value changes from 0000H to FFFFH.)

(1) Operation modes

The block has a total of three different operation modes: 2-channel independent mode, 8-bit prescaler + 8-bit PPG mode, and single-channel 16-bit PPG mode.

In 2-channel independent mode, the two channels operate as independent 8-bit PPGs. The PPG output of ch0 is connected to the PPG0 pin and the PPG output of ch1 is connected to the PPG1 pin.

In 8-bit prescaler + 8-bit PPG mode, ch0 operates as an 8-bit prescaler and ch1 counts on each ch0 borrow. This operation mode produces a variable-period 8-bit PPG waveform output. The prescaler output of ch0 is connected to the PPG0 pin and the PPG output of ch1 is connected to the PPG1 pin.

In single-channel 16-bit PPG mode, the upper and lower registers of ch0 and ch1 are linked to operate as a single 16-bit PPG. (That is, the PRLL0 and PRLL1 registers and the PRLH0 and PRLH1 registers are linked.) The 16-bit PPG output is connected to both the PPG0 and PPG1 pins.

(2) PPG output operation

Set "1" to bit 7 (PEN0) of the PPGC0 (PPG operation mode control) register or bit 15 (PEN1) of the PPGC1 register to activate PPG0 or PPG1 respectively and start counting. After starting operation, writing "0" to bit 7 (PEN0) of the PPGC0 register or bit 15 (PEN1) of the PPGC1 register stops count operation and holds the pulse output at the "L" level.

In 8-bit prescaler + 8-bit PPG mode, do not start ch1 operation when ch0 is stopped.

In single-channel 16-bit PPG mode, always start or stop operation by bit 7 (PEN0) of the PPGC0 register and bit 15 (PEN1) of the PPGC1 register together.

The following describes PPG output operation.

Variable frequency and duty ratio pulse waveforms are output continuously during PPG operation. (The duty ratio is the ratio of the H-level and L-level durations in the pulse waveform.) Once started, PPG pulse waveform output does not stop until operation stop is specified.

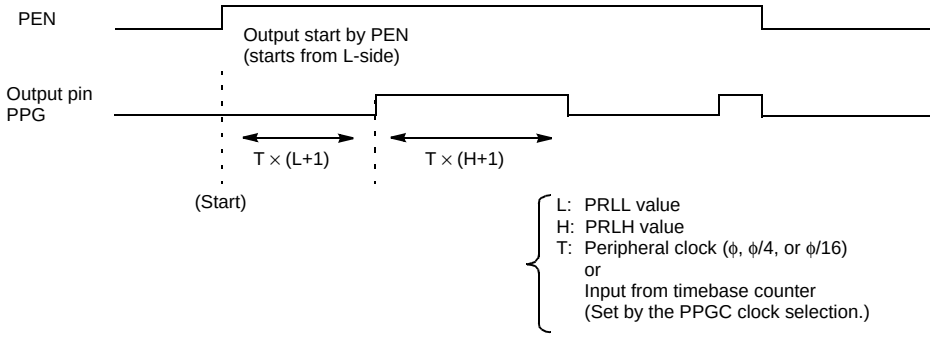


Fig. 1.0.3 Figure 2.6.3 PPG Output Operation and Output Waveforms

(3) Relationship between the reload values and pulse width

The output pulse width is the product of the value written in the reload register + 1 and the count clock period. Note that a reload register value of 00H for 8-bit PPG operation or 0000H for 16-bit PPG operation generates a pulse width of one count clock cycle. Also, a reload register value of FFH in 8-bit PPG operation generates a pulse width of 256 count clock cycles and a reload register value of FFFFH in 16-bit PPG operation generates a pulse width of 65536 count clock cycles. The pulse width is calculated as follows.

PI = T×(L+1)
Ph = T×(H+1)

L: PRLL value
H: PRLH value
T: Input clock period
Ph: H-side pulse width
PI: L-side pulse width

(4) Count clock selection

This block can select from four count clock inputs, using either the peripheral clock or timebase counter input. (ch1 of the PPG can only select from two count clocks.)

When bits 2 and 1 (PCM1, 0) of the PPGC0 register are "00", the count clock counts once for each peripheral clock count.

When bits 2 and 1 (PCM1, 0) of the PPGC0 register are "01", the count clock counts once for each four peripheral clock counts.

When bits 2 and 1 (PCM1, 0) of the PPGC0 register are "10", the count clock counts once for each 16 peripheral clock counts.

When bits 2 and 1 (PCM1, 0) of the PPGC0 register are "11", the count clock counts once for each timebase counter input.

When bit 14 (PCS1) of the PPGC1 register is "0", the count clock counts once for each peripheral clock count.

When bit 14 (PCS1) of the PPGC1 register is "1", the count clock counts once for each timebase counter input.

However, bit 14 (PCS1) of the PPGC1 register has no meaning in 8-bit prescaler + 8-bit PPG mode and 16-bit PPG mode. In these modes, ch1 of the PPG receives its count clock from ch0.

When using the timebase counter input, note that the count period may vary for the first count after activation by the trigger or the first count after a stop. The period may also vary if the timebase counter is cleared during operation of this module.

Also, when using 8-bit prescaler + 8-bit PPG mode, note that the period of the first count may vary if ch0 is already operating when ch1 is changed from stopped to activated.

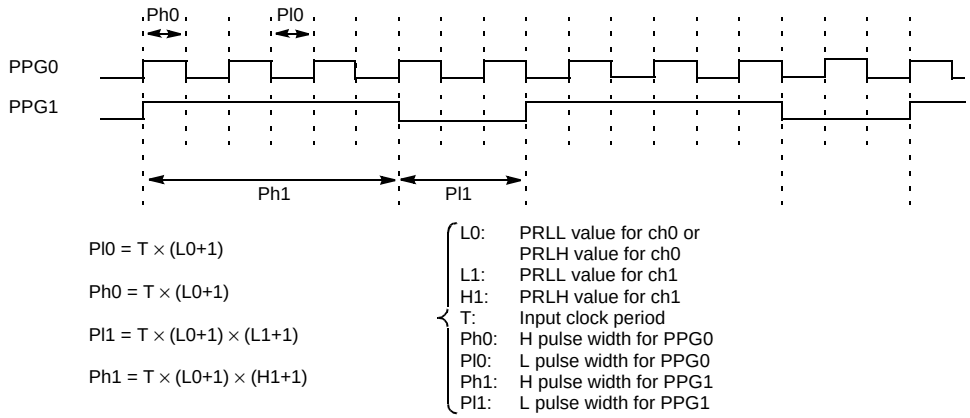
(5) Controlling pulse output from the pin

The pulse outputs generated by this module can be output from external pins PPG0 and PPG1.

Bit 5 (POE0) of PPGC0 enables external output from pin PPG0 and bit 13 (POE1) of PPGC1 enables external output from pin PPG1. When these bits are set to "0" (initial value), pulse output from the external pin is disabled and the pin functions as a general-purpose port. Setting the bits to "1" enables pulse output from the external pin.

As PPG0 and PPG1 output the same waveforms in 16-bit PPG mode, enabling either of these external output pins produces the same output.

In 8-bit prescaler + 8-bit PPG mode, PPG0 outputs the toggle waveforms from the 8-bit prescaler and PPG1 outputs the 8-bit PPG waveforms. The following shows an example of the output waveforms in this mode.



Note: Setting the same value to PRLL and PRLH for ch0 is recommended.

Fig. 1.0.4 8+8 PPG Output Operation and Output Waveforms

(6) Interrupts

The interrupts for this module become active when the reload value counts-out and a borrow occurs.

In 2-channel 8-bit PPG mode and 8-bit prescaler + 8-bit PPG mode, interrupt requests are generated for each counter borrow. In 16-bit PPG mode, PUF0 and PUF1 are set simultaneously by the 16-bit counter borrow. Therefore, enabling only one of PIE0 and PIE1 is recommended so as to produce only one source of interrupt. Clearing the sources of interrupt in both PUF0 and PUF1 is also recommended.

(7) Initial values for each hardware element

The hardware elements in this block are initialized by a reset as follows.
<Registers>

- PPGC0 ⇨ 0X000001B
- PPGC1 ⇨ 00000001B

<Pulse output>
PPG0 ⇨ "L"
PPG1 ⇨ "L"
POE0 ⇨ PPG0 output inhibit
POE1 ⇨ PPG1 output inhibit
<Interrupt request>
IRQ0 ⇨ "L"
IRQ1 ⇨ "L"

Hardware elements other than those listed above are not initialized.

Note: Write timings to the reload register

Writing to the reload registers (PRLL and PRLH) using word move instructions is recommended in modes other than 16-bit PPG mode. Writing using two byte transfer instructions may result in output of an unexpected pulse width, depending on the timing.

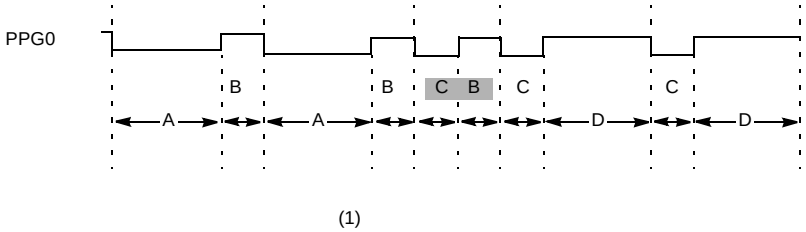


Fig. 1.0.5 Write Timing Timechart

If the value of PRLL is changed from A to C before timing (1) on the timechart and the value of PRLH is changed from B to D after timing (1), the value of PRL at timing 1 is PRLL=C and PRLH=B. This results in the output of one pulse cycle with an L pulse width of C counts and an H pulse width of B counts.

Similarly, in 16-bit PPG mode, write to the PRL registers for ch0 and ch1 using a long word move instruction, or write to ch0 first and ch1 second using word move instructions. In this mode, writing to the ch0 PRL register only performs a temporary write. The actual write to the ch0 PRL register is performed at the same timing as the write to the ch1 PRL register.

In modes other than 16-bit PPG mode, the PRL registers for ch0 and ch1 can be written to independently.

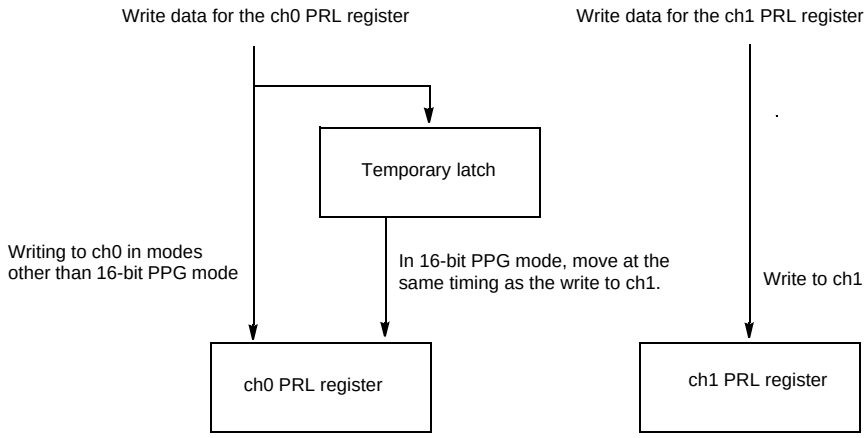


Fig. 1.0.6 Block Diagram of the Steps for Writing to PRL

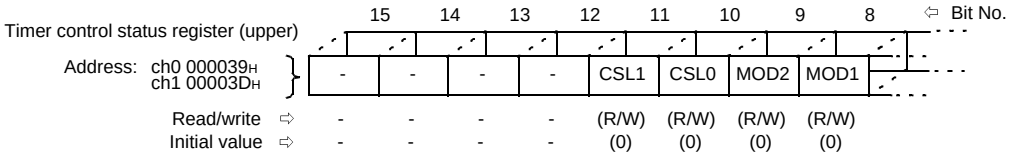
Note: The PPG0 and PPG1 pins are shared with other functions. Unless done intentionally, halt input (INT5 and INT6) to the pins from other functions.

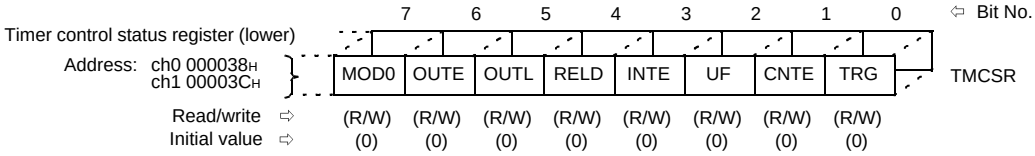
2.7 16-Bit Reload Timer (With Event Count Function)

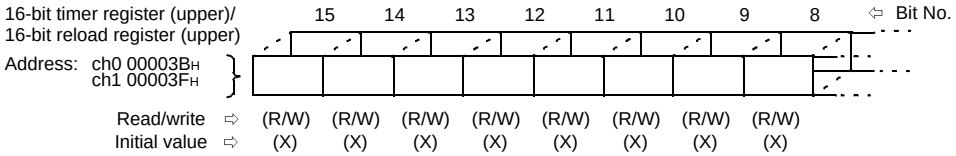
The 16-bit reload timers consists of a 16-bit down-counter, a 16-bit reload register, one input (TIN) and one output (TOT) pin, and a control register. The input clock can be selected from one external clock and three types of internal clock. The output pin (TOT) outputs a toggle waveform in reload mode and a rectangular waveform during counting in one-shot mode. The input pin (TIN) functions as the event input in event count mode and as the trigger input or gate input in internal clock mode.

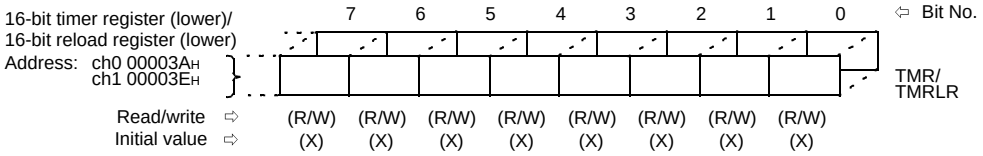
This product has two internal 16-bit reload timer channels.

2.7.1 Registers









2.7.2 Block Diagram

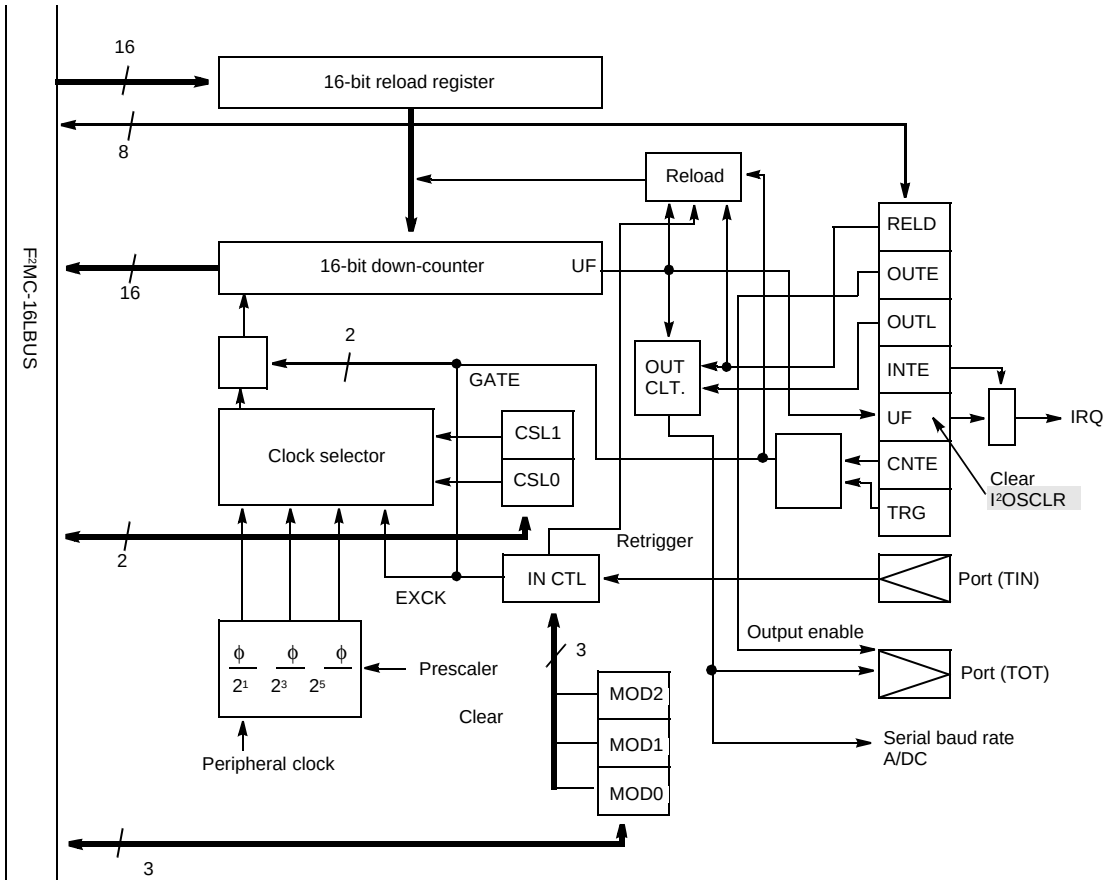
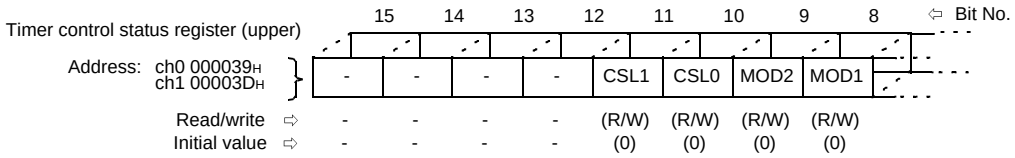


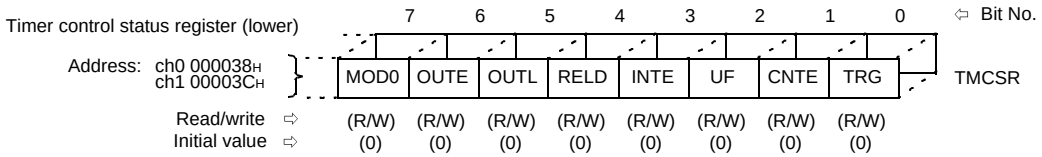
Fig. 2.7.1 Block Diagram

2.7.3 Register Details

(1) TMCSR (Timer control status register)

■ Register layout





■ Register contents

Controls the operation mode and interrupts for the 16-bit timer. Only modify bits other than UF, CNTE, and TRG when CNTE = "0".

■ Bit meanings

[Bits 11, 10] CSL1, CSL0 (Clock SeLect 1, 0)

The count clock select bits. Table 2.7.1 lists the clock source selections.

Table 2.7.1 CSL Bit Clock Source Settings

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 retrigger input. In this case, the reload register 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, 0 bits set the pin functions for each mode. Tables 2.7.2 and 2.7.3 list the MOD2, 1, 0 bit settings.

Table 2.7.2 MOD2, 1, 0 Bit Settings (1)

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

Table 2.7.3 MOD2, 1, 0 Bit Settings (2)

Event counter mode (CSL0, 1 = "11")

MOD2	MOD1	MOD0	Input Pin Function	Active Edge or Level
×	0	0	-	-
	0	1	Trigger input	Rising edge
	1	0	⇧	Falling edge
	1	1	⇧	Both edges

Note: Bits marked as × in the table can be set to any value.

[Bit 6] OUTE
Output enable bit. The TOT 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, TOT outputs a rectangular waveform that indicates when counting is in progress.

[Bit 5] OUTL
This bit sets the output level for the TOT pin. When OUTL is "0" or "1", the output pin level is opposite.

Table 2.7.4 OUTE, RELD, and OUTL Settings

OUTE	RELD	OUTL	Output Waveform
0	x	x	General-purpose port
1	0	0	Output an "H" rectangular waveform during counting.
1	0	1	Output an "L" rectangular waveform during counting.
1	1	0	Toggle output. "L" level at count start.
1	1	1	Toggle output. "H" level at count start.

[Bit 4] RELD (RELoaD)
This bit enables reload operations. When RELD = "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 0000H to FFFFH). When RELD = "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 0000H to FFFFH.

[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 when the UF bit changes to "1".

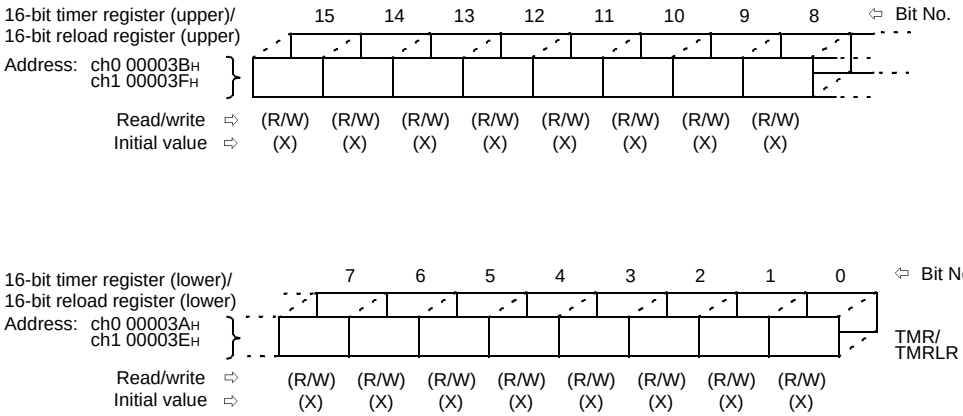
[Bit 2] UF (UnderFlow)
Timer interrupt request flag. UF is set to "1" when an underflow occurs (when the counter value changes from 0000H to FFFFH). Cleared by writing "0" or by the intelligent I/O service. Writing "1" to this bit has no meaning. Read as "1" by read-modify-write instructions.

[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".

(2) TMR (16-bit timer register)/TMRLR (16-bit reload register)

■ Register layout



■ TMR contents

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

The 16-bit reload register holds the initial count value. The initial value is indeterminate. Always write to this register using word move instructions.

2.7.4 Operation

(1) Internal clock operation

The machine clock divided by 2^1, 2^3, or 2^5 can be selected as the clock source when operating the timer from an internal 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 2.7.2 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.

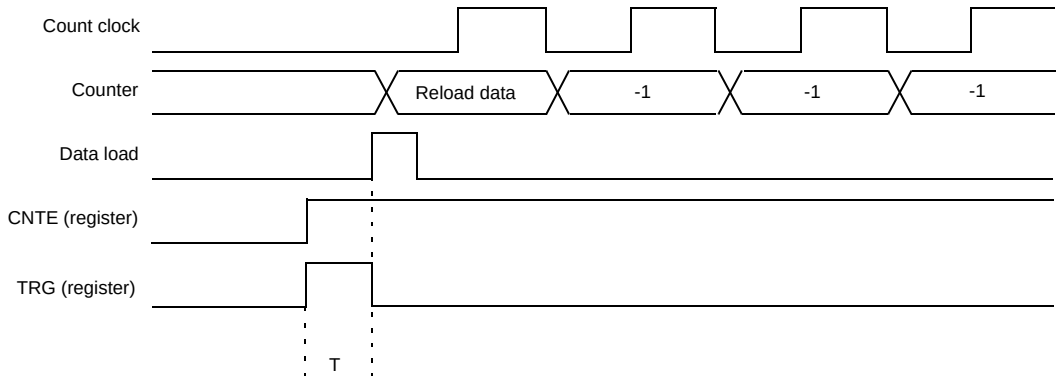


Fig. 2.7.2 Counter Activation and Operation

(2) Underflow operation

An underflow is defined for this timer as the time when the counter value changes from 0000H to FFFFH. 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 FFFFH.

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 2.7.3 shows the operation when an underflow occurs.

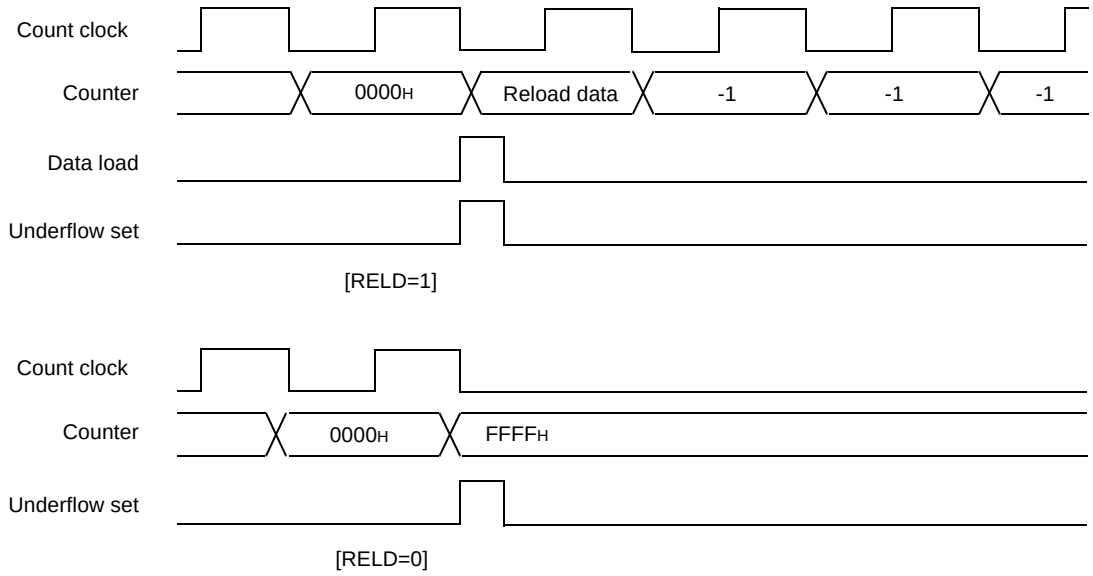


Fig. 2.7.3 Underflow Operation

(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 contents of the reload register into the counter, clear the internal prescaler, and start counting. Input a pulse width of at least 2×T (T is the machine cycle) to TIN. Figure 2.7.4 shows the operation of trigger input.

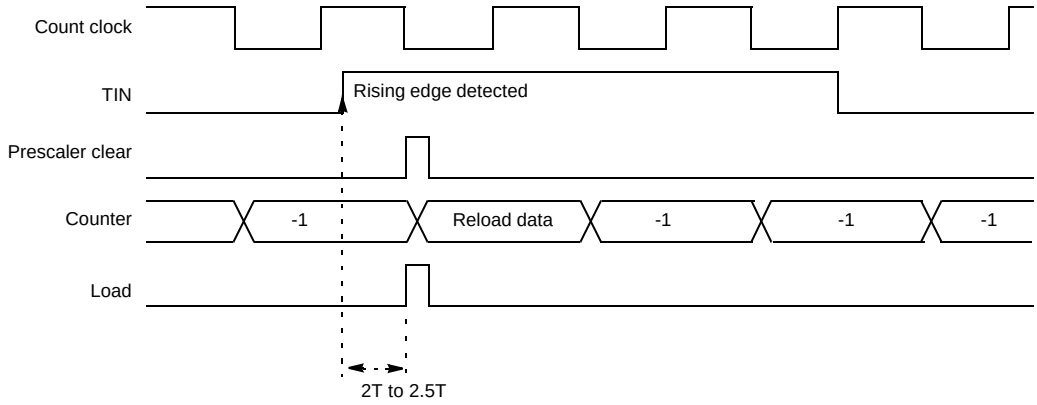


Fig. 2.7.4 Trigger Input Operation

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. The count clock operates continuously during this time. The software trigger can be used in gate mode, regardless of the gate level. Input a pulse width of at least 2×T (T is the machine cycle) to the TIN pin. Figure 2.7.5 shows the operation of gate input.

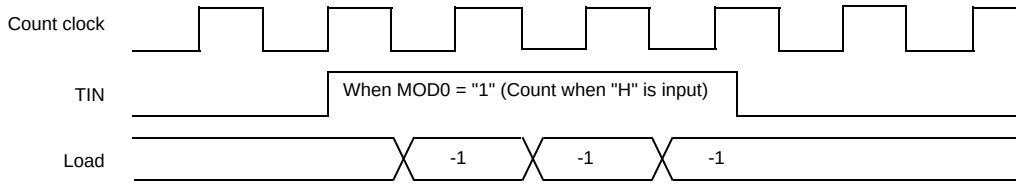


Fig. 2.7.5 Gate Input Operation

(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 4×T (T is the machine cycle) to the TIN pin.

(5) Output pin functions

In reload mode, the TOT pin performs toggle output (inverts at each underflow). In one-shot mode, the TOT 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 opposite levels are output when OUTL = "1".

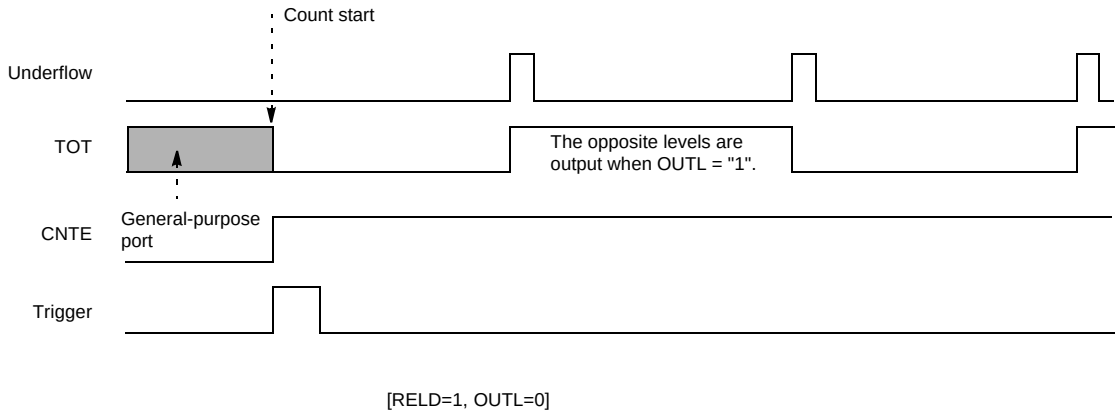


Fig. 2.7.6 Output Pin Functions (1)

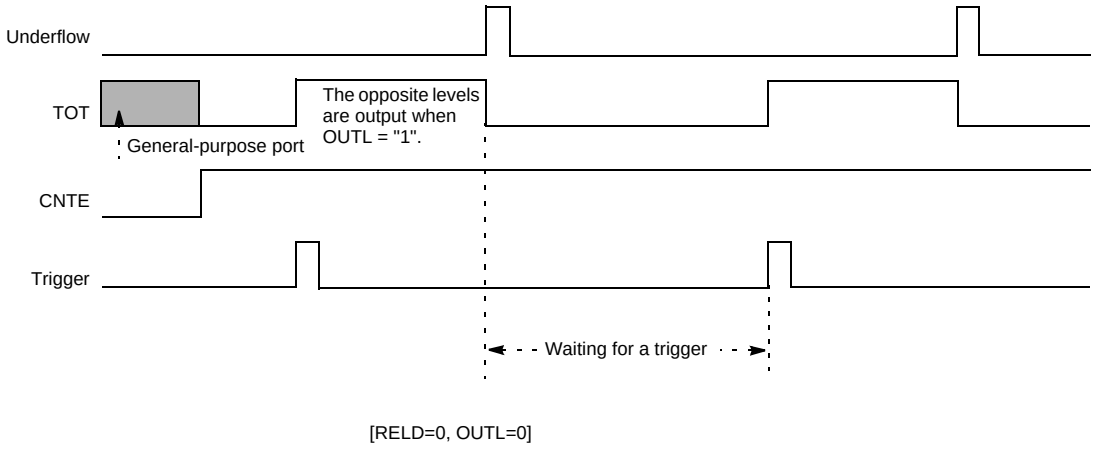


Fig. 2.7.7 Output Pin Functions (2)

(6) Intelligent I/O service (I²OS) function and interrupts

The timer includes a circuit that supports I²OS. The timer can activate I²OS when an underflow occurs. I²OS 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 I²OS services. Also, as the two timers have different interrupt vectors, they can be assigned to different interrupt services. However, as ch0 and ch1 share an interrupt control register as described above, the same interrupt level applies to both channels.

(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), and CNTE = "1" and WAIT = "0" (RUN state). Figure 2.7.8 shows the transitions between each state.

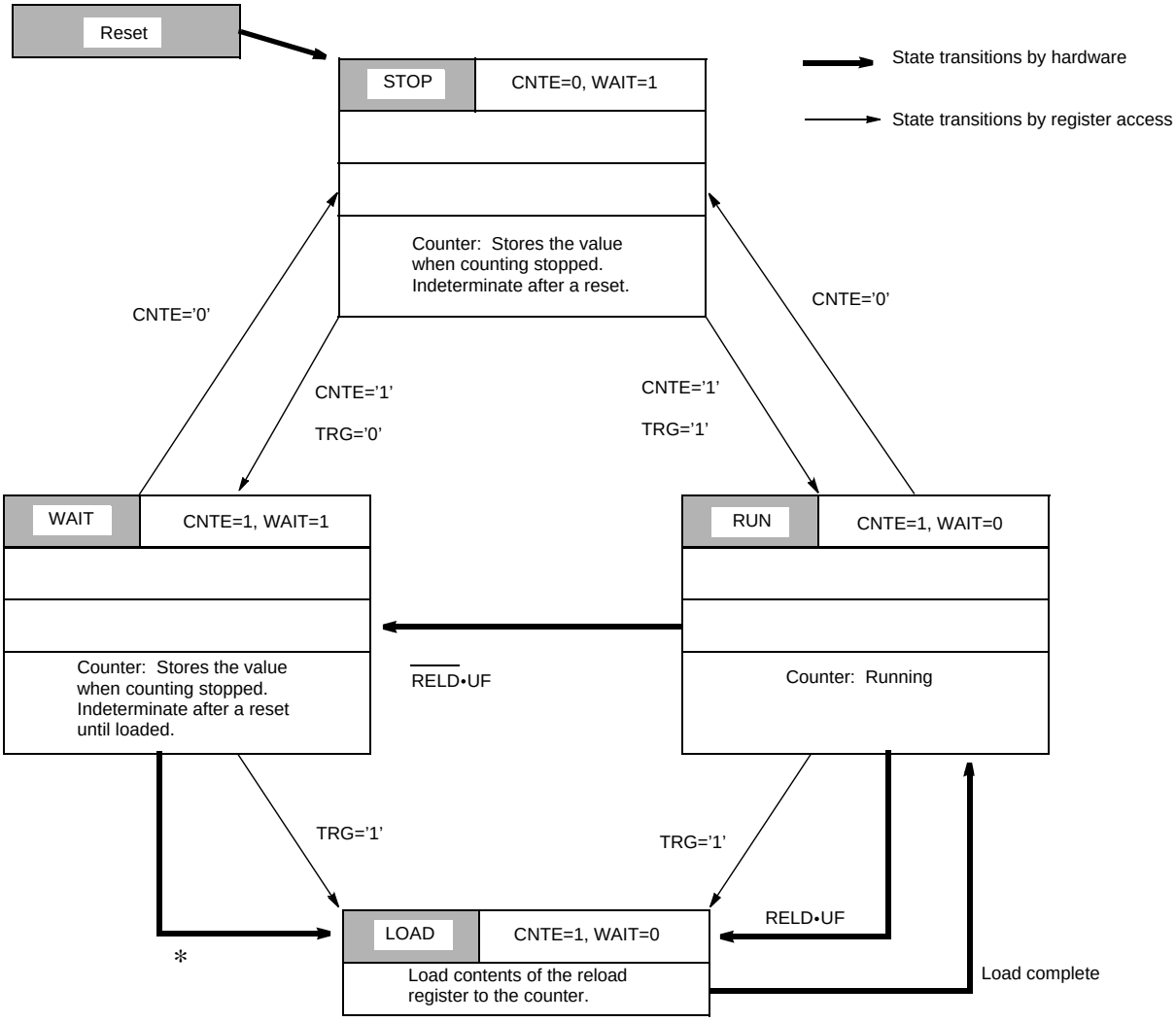


Fig. 2.7.8 Counter State Transitions

2.8 Chip Select Function

2.8.1 Outline

This module generates chip select signals to simplify connection of memory or I/O devices. The module has 8 chip select output pins. The hardware outputs the chip select signals from the pins when it detects access of an address in the areas specified in the pin registers.

2.8.2 Block Diagram

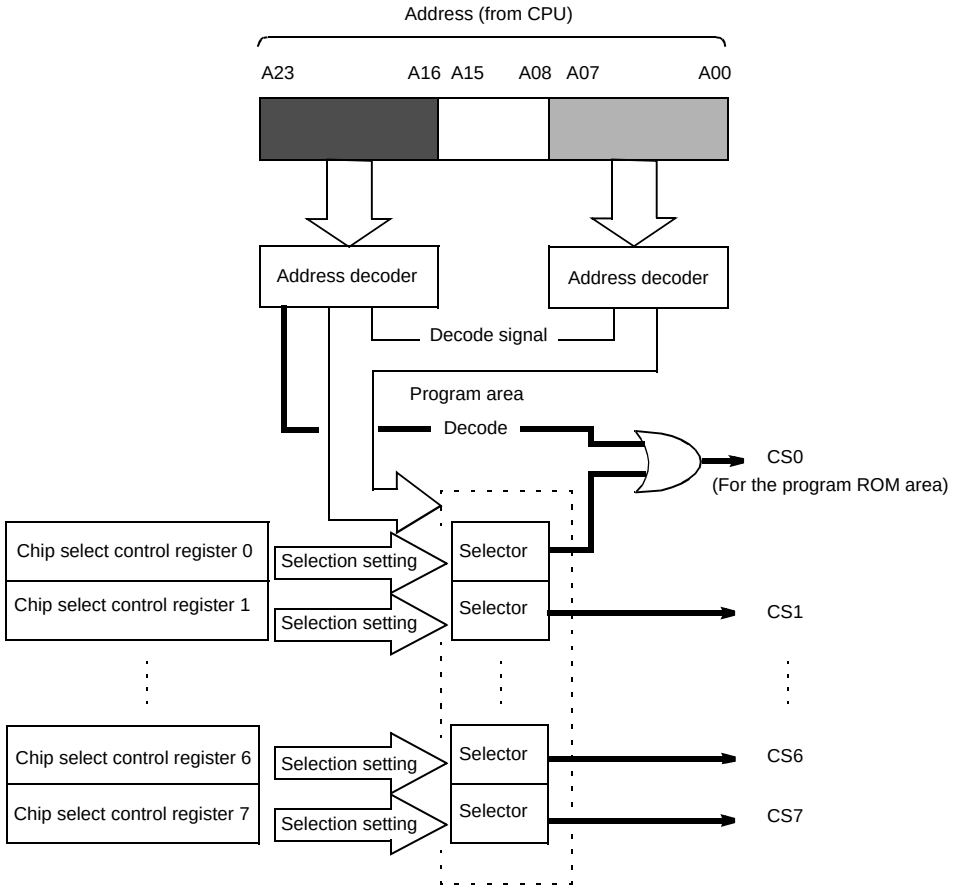


Fig. 2.8.1 Block Diagram of Chip Select

2.8.3 Registers

Chip select control register

Address: 000049H, 00004BH, 00004DH, 00004FH

15	14	13	12	11	10	9	8
-	-	-	-	ACTL	OPEL	CSA1	CSA0

(odd numbers: CSCR1, 3, 5, and 7)

Address: 000048H, 00004AH, 00004CH, 00004EH

7	6	5	4	3	2	1	0
-	-	-	-	ACTL	OPEL	CSA1	CSA0

(even numbers: CSCR0, 2, 4, and 6)

2.8.4 Register Details

○ Chip select control registers (CSCR0 to 7)

15/07	14/06	13/05	12/04	11/03	10/02	09/01	08/00	⇦ Bit No.
-	-	-	-	ACTL	OPEL	CSA1	CSA0	CSCR0 to 7
-	-	-	-	0	0	0	0	Initial value
-	-	-	-	R/W	R/W	R/W	R/W	Read/write

[Bits 15/07 to 12/04]: Unused bits
Unused bits. The read value is undefined.

[Bit 11/03]: ACTL
These bits set the active level for pins CS0 to 7. Set as follows.

"0": Output "L" from the CS0 to 7 pin when an address is decoded.

"1": Output "H" from the CS0 to 7 pin when an address is decoded.

[Bit 10/02]: OPEL (Pin output is always enabled for CS0.)
These bits enable or disable external output for pins CS1 to 7. The settings are as follows. Note that CS0 is always enabled and so setting is not necessary.

"0": Disable decoding output from the CS1 to 7 pin.

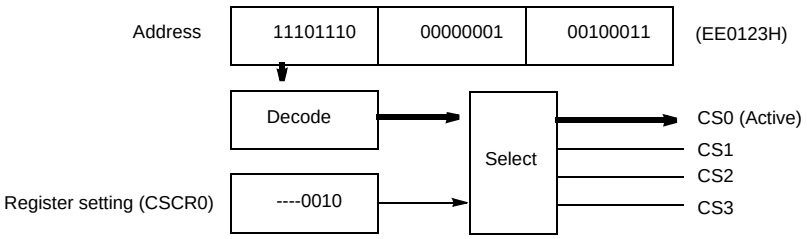
"1": Enable decoding output from the CS1 to 7 pin.

[Bits 09/01 to 08/00]: CSA1, CSA0
These bits select the address decode range (ROM, RAM, or external circuit) for each chip select pin. See Table 2.8.1 for details of the ranges and sizes.

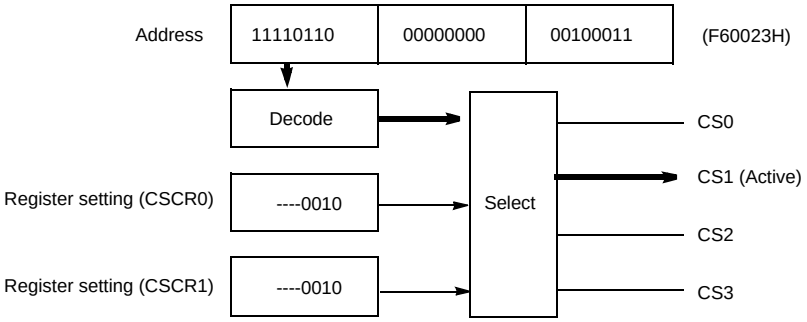
2.8.5 Operation

○ When the CPU accesses program or data, the chip select module decodes the highest and lowest order bytes of the address and activates the chip select signals based on the output setting registers.

[Example 1]



[Example 2]



- This module's chip select function sets the CS0 to 7 pin signals to active when the CPU accesses an area specified by the CSA1 and CSA0 bits in the output setting register. The table below lists the bit settings.
- Output can be masked by setting the OPEL bit in the output setting register.
- The active level of the CS0 to 7 pins can be changed using the ACTL bit in the output setting register.

Table 2.8.1 Address Decoding Areas

Pin	CSA		Decode Area	Area Size (Bytes)	Remarks
	1	0			
CS0	0	0	F00000H to FFFFFFH	1 Mbyte	Goes active when a program ROM area or program vector fetch occurs.
	0	1	F80000H to FFFFFFH	512 Kbyte	
	1	0	FE0000H to FFFFFFH	128 Kbyte	
	1	1	-	For future use	
CS1	0	0	E00000H to EFFFFFH	1 Mbyte	Use for data ROM or RAM areas, or for external circuits.
	0	1	F00000H to F7FFFFH	512 Kbyte	
	1	0	FC0000H to FDFFFFH	128 Kbyte	
	1	1	68FF80H to 68FFFFH	128 byte	
CS2	0	0	-	For future use	Use for data ROM or RAM areas, or for external circuits.
	0	1	FA0000H to FBFFFFH	128 Kbye	
	1	0	68FF80H to 68FFFFH	128 byte	
	1	1	68FF00H to 68FF7FH	128 byte	
CS3	0	0	F80000H to F9FFFFH	128 Kbyte	Use for I/O or RAM areas, or for external circuits.
	0	1	68FF00H to 68FF7FH	128 byte	
	1	0	68FE80H to 68FEFFH	128 byte	
	1	1	0000C0H to 0000FFH	64 byte	
CS4	0	0	-	For future use	Use for I/O or RAM areas, or for external circuits.
	0	1	68FE80H to 68FEFFH	128 byte	
	1	0	0000C0H to 0000FFH	64 byte	
	1	1	0000E0H to 0000FFH	32 byte	
CS5	0	0	68FF80H to 68FFFFH	128 byte	Use for I/O or RAM areas, or for external circuits.
	0	1	0000C0H to 0000FFH	64 byte	
	1	0	0000E0H to 0000FFH	32 byte	
	1	1	0000D8H to 0000DFH	8 byte	
CS6	0	0	68FF00H to 68FF7FH	128 byte	Use for I/O or RAM areas, or for external circuits.
	0	1	0000C0H to 0000FFH	64 byte	
	1	0	0000E0H to 0000FFH	32 byte	
	1	1	0000D0H to 0000D7H	8 byte	
CS7	0	0	0000C0] to 0000FFH	64 byte	Use for I/O or RAM areas, or for external circuits.
	0	1	0000E0H to 0000FFH	32 byte	
	1	0	0000C8H to 0000CFH	8 byte	
	1	1	-	For future use	

< NOTE >

- As the CS0 pin outputs the decode signal for the program vector fetch after a reset (initial state) (from the 128Kbyte program ROM region between FE0000H and FFFFFFH), always use this pin for program ROM. The active output level for this pin is "L".
- As the chip select decode signal pins are initialized as port inputs, you must set each setting register explicitly. (Except for the CS0 pin which is a dedicated pin and does not need to be set.)

Pin		CS0	CS1	CS2	CS3	CS4	CS5	CS6	CS7
CSA	1	001-	0011	-011	0011	-011	0011	0011	011-
	0	010-	0101	-101	0101	-101	0101	0101	010-

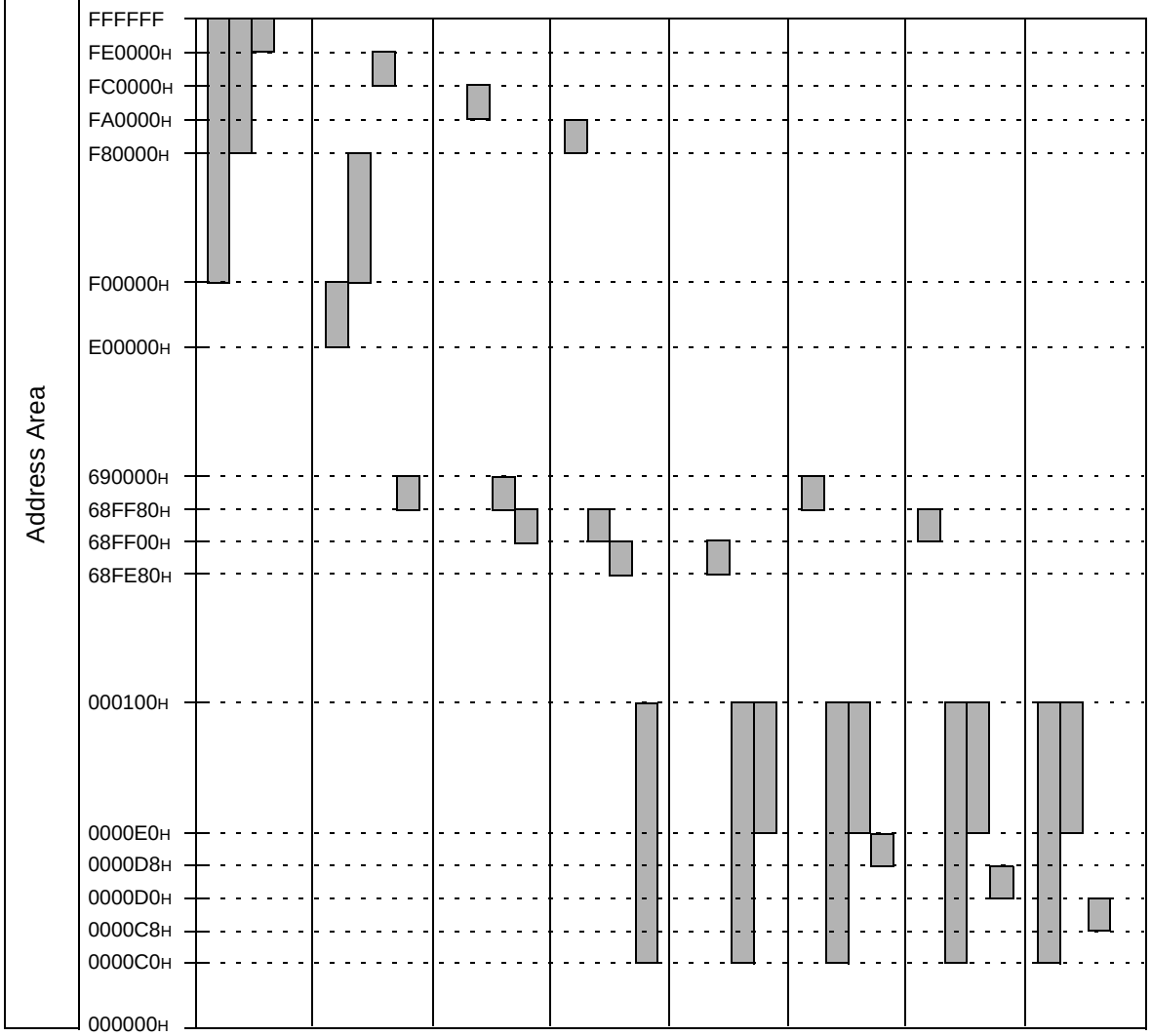


Fig. 2.8.2 Address Decoding Map
(See Table 2.8.1 for the details of the decoding areas)

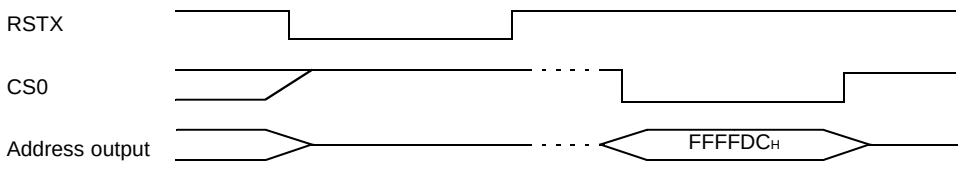


Fig. 2.8.3 CS0 Output after Release of a Reset

2.9 DTP/External Interrupts

The DTP (Data Transfer Peripheral) is a peripheral block that interfaces external peripherals to the F²MC-16L CPU. The DTP receives DMA and interrupt processing requests from external peripherals and passes the requests to the F²MC-16L CPU to activate the extended intelligent I/O service or interrupt processing. Two request levels ("H" and "L") are provided for extended intelligent I/O. For external interrupt requests, generation of interrupts on a rising or falling edge as well as on "H" and "L" levels can be selected, giving a total of four types.

2.9.1 Registers

Interrupt/DTP enable register

Address: 000028H — ENIR

Bit No.	7	6	5	4	3	2	1	0
	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
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)

Interrupt/DTP register

Address: 000029H — EIRR

Bit No.	15	14	13	12	11	10	9	8
	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0
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 (upper)

Address: 00002BH

Bit No.	15	14	13	12	11	10	9	8
	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4
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 (lower)

Address: 00002AH — ELVR

Bit No.	7	6	5	4	3	2	1	0
	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0
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)

2.9.2 Block Diagram

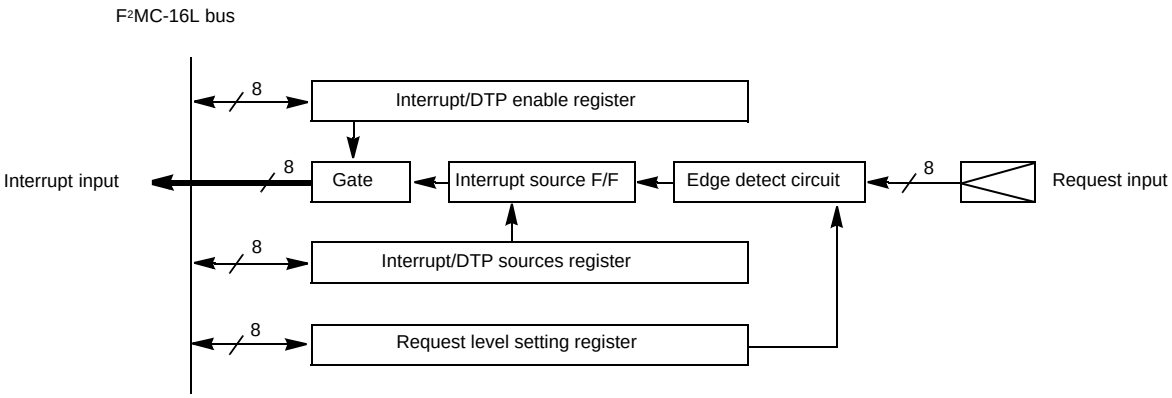
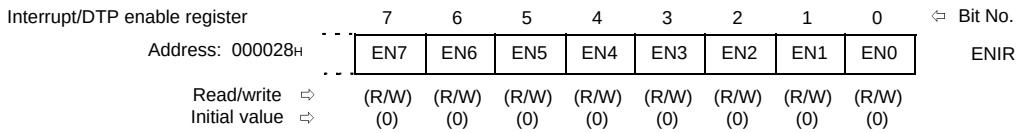


Fig. 2.9.1 Block Diagram

2.9.3 Register Details

(1) ENIR (Interrupt/DTP enable register)

■ Register layout

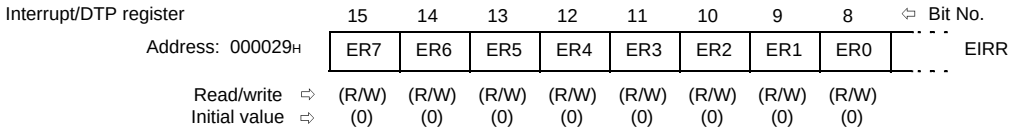


■ Register contents

The ENIR register enables or disables the function that generates interrupt requests to the interrupt controller using device pins as external interrupt or DTP request inputs. Writing "1" to a register bit sets the corresponding pin as an external interrupt or DTP request input and enables the generation of interrupt requests to the interrupt controller. Pins corresponding to bits set to "0" hold external interrupt or DTP request inputs but do not pass the request to the interrupt controller.

(2) EIRR (Interrupt/DTP register)

■ Register layout



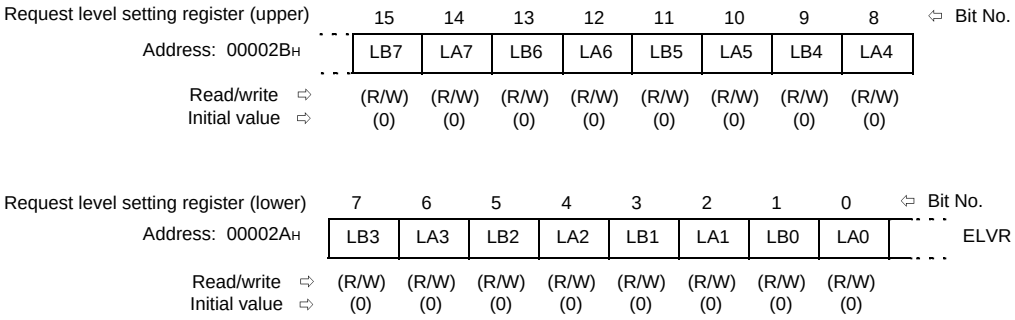
■ Register contents

Reading EIRR indicates whether any external interrupt or DTP requests are present. Writing to EIRR clears the value of the flip-flop that indicates the request. Bits that have the value "1" on reading the register indicate that an external interrupt or DTP request for the corresponding pin is present. Writing "0" to bits in this register clears the request flip-flop for the corresponding bit. Writing "1" has no effect. Always read as "1" by read-modify-write type instructions.

Note: When interrupts are enabled, only write "0" to the bit that is set to "1". Avoid unconditionally clearing other flag bits when clearing the interrupt flag bit for a received interrupt.

(3) ELVR (Request level setting register)

■ Register layout



■ Register contents

The ELVR register selects how to detect requests. The selection for each pin is set using two bits as shown below. If level-detection is selected for a pin, clearing the request when the active level is input causes the request to be set again.

Table 2.9.1 ELVR Settings

LBx	LAx	Operation
0	0	Generate a request when an input is "L" level.
0	1	Generate a request when an input is "H" level.
1	0	↑Generate a request at a rising edge.
1	1	↓Generate a request at a falling edge.

2.9.4 Operation

(1) External interrupt operation

If the request specified in the ELVR register is input to a pin after the pin has been set for external interrupts, the resource generates an interrupt request signal to the interrupt controller. The interrupt controller prioritizes any simultaneously occurring interrupts. When the interrupt request from this resource has the highest priority, the interrupt controller passes the request to the F^{2}MC-16L CPU. The F^{2}MC-16L CPU compares the interrupt request with the ILM bits of the CCR register in the CPU. If the priority of the request is higher than the priority set in the ILM bits, the hardware interrupt processing microprogram is activated after completion of the currently executing instruction.

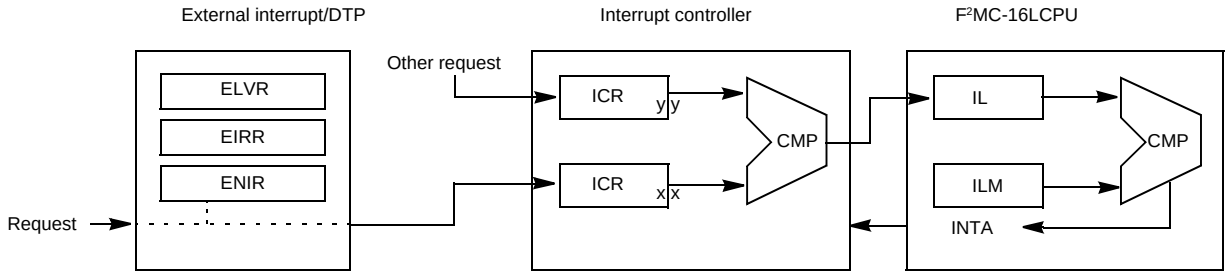


Fig. 2.9.2 External Interrupt Operation

Under control of the hardware interrupt processing microprogram, the CPU reads the ISE bit from the interrupt controller to identify that the current request is for interrupt processing, then branches to the interrupt processing microprogram. The interrupt processing microprogram reads the interrupt vector area, sends an interrupt acknowledge to the interrupt controller, and loads the macro instruction branch destination address created from the vector into the program counter to execute the user interrupt processing program.

(2) DTP operation

Set the I/O address pointer and buffer address pointer in the extended intelligent I/O service descriptor as part of user program initialization for activating the extended intelligent I/O service. Set the address of a register between 000000H and 00FFFFH as the I/O address pointer and set the top address of the memory buffer as the buffer address pointer.

The operation sequence for DTP is exactly the same as for an external interrupt up to the point where the CPU activates the hardware interrupt processing microprogram. For DTP, the ISE bit read by the CPU in the hardware interrupt processing microprogram specifies DTP operation. This causes control to pass to the microprogram for the extended intelligent I/O service. On activation, the extended intelligent I/O service sends read or write signals to the external peripheral address and performs data transfer with this chip. The external peripheral must remove the interrupt request for the chip within 3 machine cycles of performing data transfer. Descriptor updating and similar operations are performed at the completion of data transfer, then the interrupt controller generates the signal to clear the sources of transfer. On receiving the transfer source clear signal, the DTP/external interrupt resource clears the flip-flop holding the sources and waits for the next request from the pin. See "3.3.3 Extended Intelligent I/O Service" for details on extended intelligent I/O service processing.

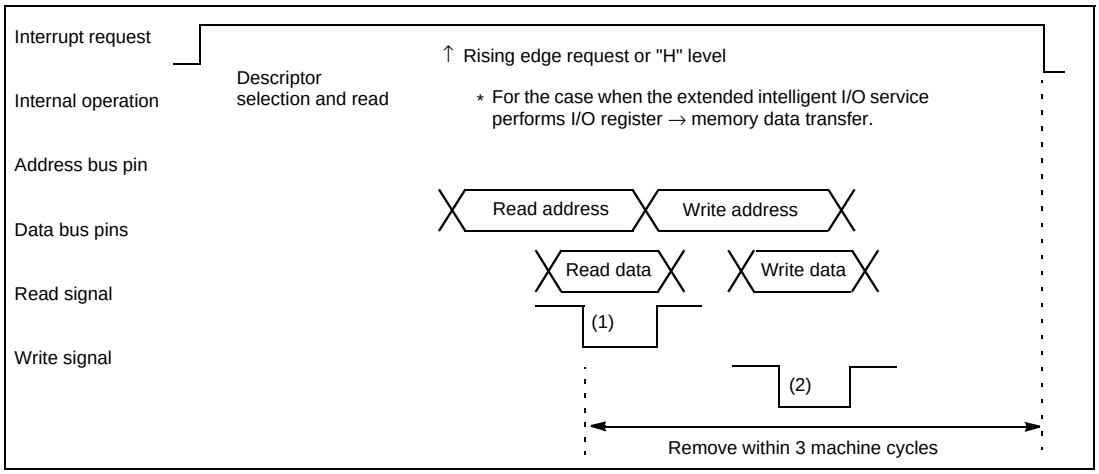


Fig. 2.9.3 External Interrupt Remove Timing at the Completion of DTP Operation

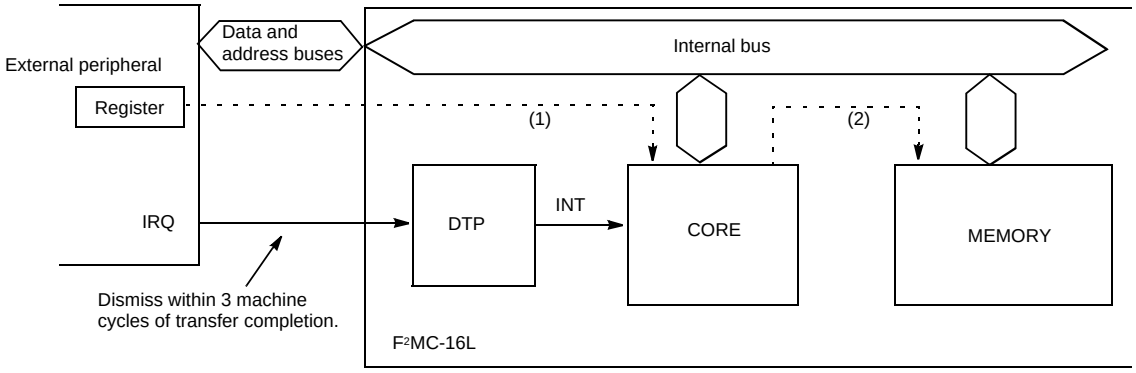


Fig. 2.9.4 Example Interface Circuit with External Peripheral

(3) Switching between external interrupt requests and DTP requests

Switching between external interrupt requests and DTP requests is performed by setting the ISE bit in the ICR register for the DTP/external interrupt resource. The ICR registers are located in the interrupt controller. Separate ICR registers are provided for each pin. Writing "1" to the ISE bit of the ICR register for a particular pin specifies DTP request operation. Writing "0" to the ISE bit specifies external interrupt request operation.

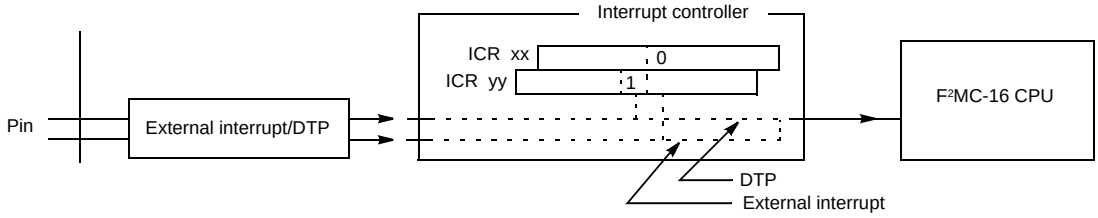


Fig. 2.9.5 Switching Between External Interrupt Requests and DTP Requests

2.9.5 Notes on Use

(1) Conditions for externally connected peripherals when using DTP

DTP only supports external peripherals that can automatically clear requests in response to the data transfer. If the transfer request is not removed within 3 machine cycles of starting data transfer, the DTP/external interrupt resource treats the request as a new transfer request.

(2) Recovery from standby

Specify an "H" level input request when using an external interrupt to recover from the standby state in clock stop mode. Using an "L" level request may cause misoperation. The system cannot recover from the standby state in clock stop mode by an edge request.

(3) Operation procedure for external interrupt/DTP

Use the following procedure to set the registers in the external interrupt/DTP resource.

1. Set the corresponding bit in the enable register to "disabled".
2. Set the corresponding bits in the request level setting register.
3. Clear the corresponding bit in the interrupt/DTP request register.
4. Set the corresponding bit in the enable register to "enabled".

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

Always set the enable register to "disabled" when setting the resource registers. Also, always clear the interrupt/DTP request register before setting the enable register to "enabled". This is to prevent generation of a source of unintended interrupt when setting registers or when enabling interrupts.

(4) External interrupt request level

① A pulse width of at least 3 machine cycles is required to detect an edge when the request level is set to edge detect.

② When the request level is set to level detect, a request to the interrupt controller remains active even if the external request input is removed. This is because external request inputs are held in an internal request hold circuit. The internal request hold circuit must be cleared to remove the request to the interrupt controller.

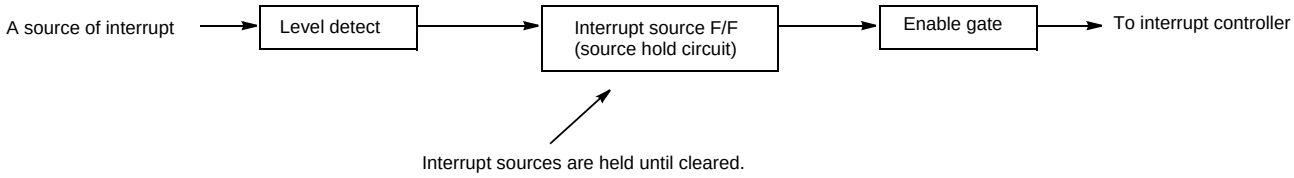


Fig. 2.9.6 Clearing the Interrupt Source Hold Circuit for Level Detect Operation

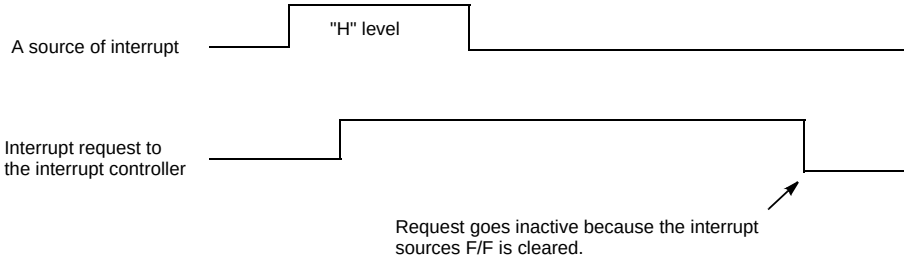


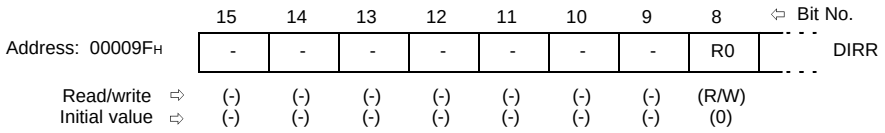
Fig. 2.9.7 Interrupt Hold F/F and Interrupt Request to the Interrupt Controller When Interrupts are Enabled

Note: The INT4 to 7 pins are shared with other functions. Unless being done intentionally, halt input or output (A/D activation, PPG output) to these pins from other functions.

2.10 Delay Interrupt Generation Module

The delay interrupt generation module is used to generate the task switching interrupt. Interrupt requests to the F²MC-16L CPU can be generated and cleared by software using this module.

2.10.1 Register



2.10.2 Block Diagram

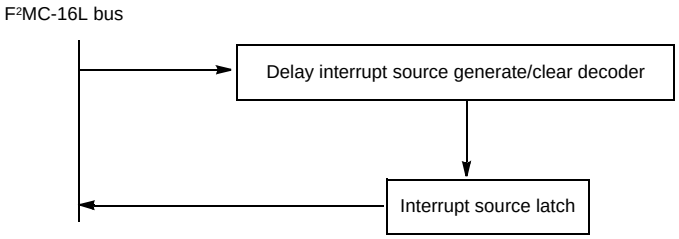
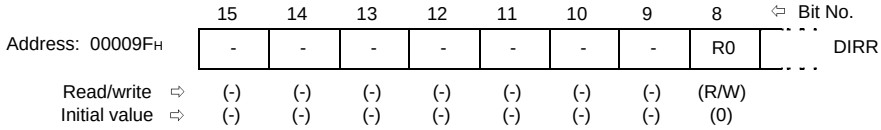


Fig. 2.10.1 Block Diagram

2.10.3 Register Details

(1) DIRR (Delay interrupt source generate/clear register)

■ Register layout



■ Register contents

The DIRR register controls generation and clearing of delay interrupt requests. Writing "1" to the register generates a delay interrupt request. Writing "0" to the register clears a delay interrupt request. The register is set to the interrupt cleared state by a reset. Either "0" or "1" can be written to the reserved bits. However, considering possible future extensions, it is recommended that the set bit and clear bit instructions are used for register access.

2.10.4 Operation

(1) Delay interrupt generation

When the CPU writes "1" to the R0 bit of the DIRR register by software, a request latch in the delay interrupt generation module is set and an interrupt request to the interrupt controller generated. If no other interrupts are present, or the other interrupts have a lower priority, the interrupt controller passes the interrupt request to the F²MC-16L CPU. The F²MC-16L CPU compares the interrupt request with the ILM bits of the CCR register in the CPU. If the priority of the request is higher than the priority set in the ILM bits, the hardware interrupt processing microprogram is activated after completion of the currently executing instruction. As a result, the interrupt processing routine for this interrupt is executed.

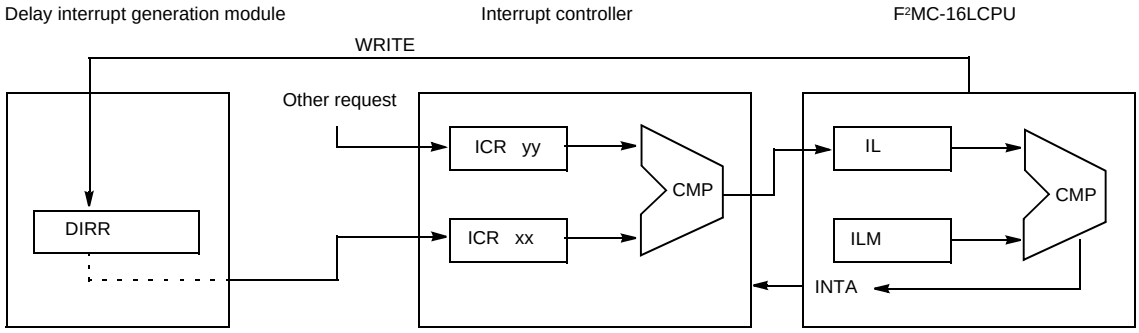


Fig. 2.10.2 Delay Interrupt Generation Operation

Write "0" to the R0 bit in the DIRR register to clear the source of interrupt and perform task switching in the interrupt processing routine.

2.10.5 Notes on Use

(1) Delay interrupt request latch

This latch is set when "1" is written to the R0 bit of the DIRR register and cleared by writing "0" to the R0 bit. Accordingly, interrupt processing is activated again on returning from the interrupt processing routine if the routine's software does not clear the source of interrupt. Take care when developing software that this situation does not occur.

2.11 Watchdog Timer and Timebase Timer Functions

The watchdog timer consists of a 2-bit watchdog counter, a control register, and a watchdog reset controller. The watchdog counter uses the carry-up signal from the 18-bit timebase timer as its clock source. In addition to the 18-bit timer, the timebase timer contains an interval interrupt control circuit. The timebase timer uses the main clock, regardless of the value of the MCS bit in the CKSCR register. Figure 2.11.1 shows the structure of the watchdog and timebase timers.

2.11.1 Registers

Watchdog timer control register

	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: 0000A8H	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	WDTC
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)	(1)	(1)	(1)	

Timebase timer control register

	15	14	13	12	11	10	9	8	⇦ Bit No.
Address: 00000A9H	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)	

2.11.2 Block Diagram

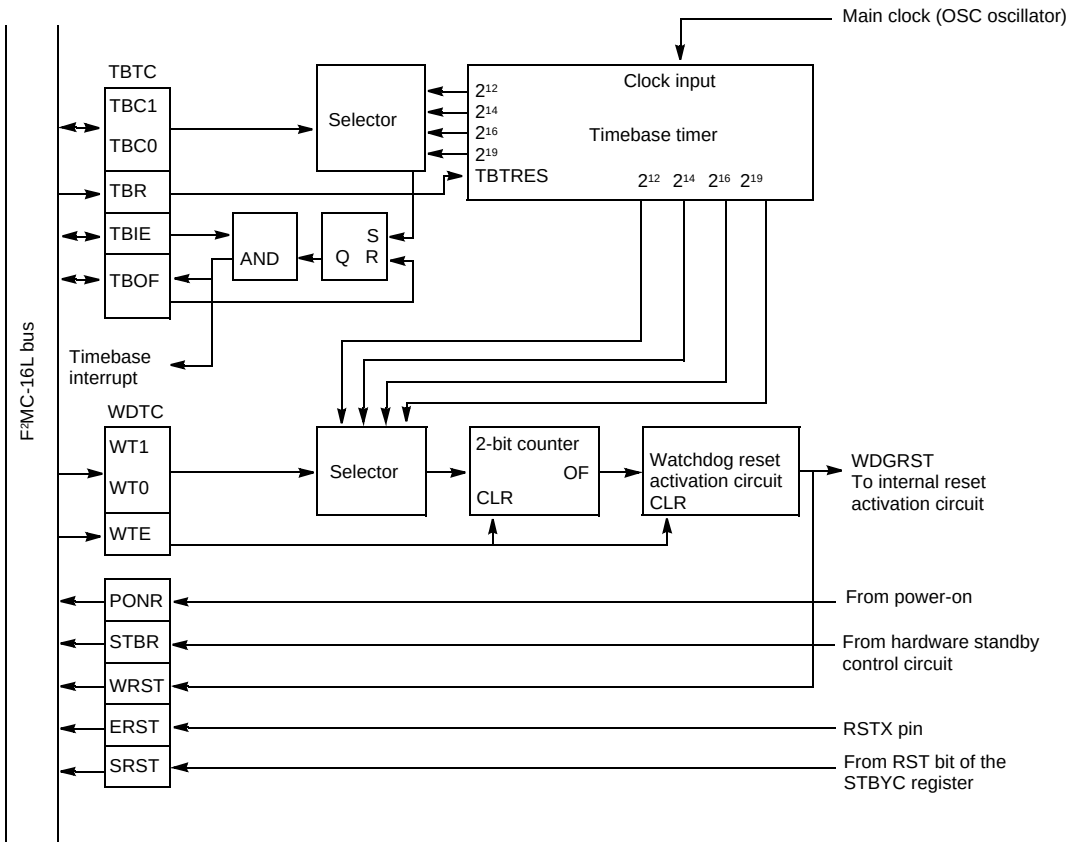


Fig. 2.11.1 Block Diagram of the Watchdog and Timebase Timers

2.11.3 Register Details

(1) WDTC (Watchdog timer control register)

■ Register layout

Watchdog timer control register	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: 0000A8H	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)	

Note 1: Do not access this register using read-modify-write instructions as this can cause misoperation.

■ Register contents

This register contains various watchdog timer control bits and reset type flag bits.

■ Bit meaning

[Bits 7 to 3] PONR, STBR, WRST, ERST, SRST
Reset type flags. When a reset occurs, these bits are set as shown in Table 2.11.1, depending on the reset type. Reading WDTC clears all these bits to "0". These bits are read-only. At power-on, the values of bits other than the power-on flag are undefined. Accordingly, when developing software, ignore the other bits if the PONR bit is "1".

Table 2.11.1 Value of the Reset Type Bits for Each Reset Type

Reset Type	PONR	STBR	WRST	ERST	SRST
Power-on	1	-	-	-	-
Hardware standby	*	1	*	*	*
Watchdog timer	*	*	1	*	*
External pin	*	*	*	1	*
RST bit	*	*	*	*	1

(The values indicated by * hold their previous values.)

[Bit 2] WTE
Writing "0" to this bit when the watchdog timer is stopped starts the watchdog timer running. Writing "0" to this bit for the second and subsequent times clears the watchdog timer counter. Writing "1" has no meaning.

The watchdog timer stops after a power-on, hardware standby, or watchdog timer reset. Reading this bit returns "1".

[Bits 1, 0] WT1, 0
Interval time selection bits for the watchdog timer. Only the data set when the watchdog timer starts is used. Values set at other times have no meaning. Table 2.11.2 lists the interval times.
These bits are write-only.

Table 2.11.2 Watchdog Timer Interval Selection Bits

WT1	WT0	Interval Time (4 MHz clock source)		Number of Main Clock Cycles
		Min.	Max.	
0	0	Approx. 3.58 ms	Approx. 4.61 ms	$2^{14}\pm2^{11}$ cycles
0	1	Approx. 14.33 ms	Approx. 18.43 ms	$2^{16}\pm2^{13}$ cycles
1	0	Approx. 57.34 ms	Approx. 73.73 ms	$2^{18}\pm2^{15}$ cycles
1	1	Approx. 458.75 ms	Approx. 589.82 ms	$2^{21}\pm2^{18}$ cycles

Note: The maximum interval value is for when the timebase counter is not reset while the watchdog timer is running.

(2) TBTC (Timebase timer control register)

■ Register layout

Timebase timer control register

	15	14	13	12	11	10	9	8	⇦ Bit No.
Address: 00000A9H	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 1: Do not access this register using read-modify-write instructions as this can cause misoperation.

■ Register contents
Controls the operation of the timebase timer and sets the interval interrupt time.

■ Bit meaning
[Bit 15] Test bit
Test bit. Always write "1" to this bit.

[Bits 14, 13] Unused

[Bit 12] TBIE
Interval interrupt enable bit for the timebase timer. Interrupts are enabled when TBIE is "1" and disabled when TBIE is "0". Cleared to "0" by a reset. The bit is readable and writable.

[Bit 11] TBOF
The timebase timer interrupt request flag. An interrupt request is generated if TBOF changes to "1" when TBIE is "1". TBOF is periodically set to "1" with the interval set by the TBC1, 0 bits. Writing "0", changing to stop or hardware standby mode, using the TBR bit to clear the timebase timer, or generating a reset clears TBOF. Writing "1" has no meaning.

Always read as "1" by read-modify-write type instructions.

Note: When clearing the TBOF bit, mask timebase timer interrupts using the TBIE bit or the ILM bits in the CPU.

[Bit 10] TBR
Clears all bits of the timebase timer counter to "0". Writing "0" clears the timebase timer counter and also clears the TBOF bit. Writing "1" has no meaning. Reading this bit returns "1".

Note: Mask timebase timer interrupts using the TBIE bit or the ILM bits in the CPU when clearing the TBOF bit.

[Bits 9, 8] TBC1, 0
Interval setting bits for the timebase timer. Table 2.11.3 lists the interval settings. Cleared to "00" by a reset. These bits are readable and writable.

Table 2.11.3 Timebase Timer Interval Selection

TBC1	TBC0	Interval Time for a 4 MHz Clock Source	Number of Main Clock Cycles
0	0	1.024 ms	2^{12} cycles
0	1	4.096 ms	2^{14} cycles
1	0	16.384 ms	2^{16} cycles
1	1	131.072 ms	2^{19} cycles

2.11.4 Operation

(1) Watchdog Timer

The watchdog timer function can be used to detect program loop. If, due to a program runaway or similar problem, "0" is not written to the WTE bit within the required time, the watchdog timer generates a watchdog reset request.

■ Activation
To start the watchdog timer, write "0" to the WTE bit in the WDTC register when the watchdog timer is stopped. At the same time, set the watchdog timer reset interval in the WT1 and WT0 bits. Only the interval data set at the time the watchdog timer is started is used.

■ Preventing a watchdog timer reset
Once the watchdog timer has started, the program must periodically clear the 2-bit watchdog counter. Specifically, the program must periodically write "0" to the WTE bit in the WDTC register. The watchdog counter consists of a 2-bit counter that uses the carry-up signal from the 18-bit timebase counter as its clock source. Therefore, the time until a watchdog timer reset occurs is lengthened if the timebase timer is cleared.

Figure 2.11.2 shows the operation of the watchdog timer.

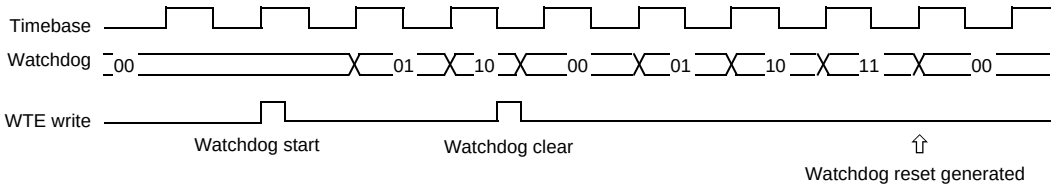


Fig. 2.11.2 Watchdog Timer Operation

■ Stopping the watchdog timer

Once started, the watchdog timer is only initialized and stopped by a power-on, hardware standby, or watchdog reset. External pin or software resets clear the watchdog counter but do not stop the watchdog function.

■ Other

In addition to writing to the WTE bit, the watchdog counter is also cleared by the generation of a reset, changing to sleep or stop mode, or the hold acknowledge signal.

(2) Timebase timer

The timebase timer functions as the watchdog counter clock source, as a delay timer to wait while the main clock and PLL clock stabilize, and as an interval timer for generating interrupts with a fixed period.

■ Timebase timer

The timebase timer consists of an 18-bit counter that counts the source oscillator input used to generate the machine clock. The timebase timer counts continuously while an oscillator input is present. A power-on reset, changing to stop or hardware standby mode, using the MCS bit in the CKSCR register to change from the main clock to the PLL clock, or writing "0" to the TBR bit in the TBTC register clears the timebase timer.

The watchdog counter and interval interrupt both use the timebase timer output and are affected by clearing the timebase timer.

■ Interval interrupt function

The interval interrupt function generates a fixed-period interrupt using the carry-up signal from the timebase counter. The TBOF flag is set periodically, with the interval determined by the TBC1, 0 bits of the TBTC register. When the TBOF flag is set depends on when the timebase timer was last cleared.

As the timebase timer acts as a delay timer to wait for the PLL oscillation to stabilize when changing from main clock mode to PLL clock mode, the timebase timer is cleared at this time.

As the timebase timer acts as a delay timer to wait for oscillation to stabilize when recovering from stop or hardware standby mode, the TBOF flag is cleared when changing to these modes.

2.12 Low Power Control Circuits (CPU Intermittent Operation Function, Oscillation Stabilization Delay Time, and Clock Multiplier Function)

The following operation modes are available: PLL clock mode, PLL sleep mode, timer mode, main clock mode, main sleep mode, stop mode, and hardware standby mode. Operation modes other than PLL clock mode are classified as low power consumption modes.

In main clock mode and main sleep mode, the device operates on the main clock only (OSC oscillator clock). The PLL clock (VCO oscillator clock) is stopped in these modes and the main clock divided by 2 is used as the operating clock.

In PLL sleep mode and main sleep mode, the CPU's operating clock only is stopped and other elements continue to operate.

In watch mode, only the timebase timer operates.

Stop mode and hardware standby mode stop the oscillator. These modes maintain existing data with minimum power consumption.

The CPU intermittent operation function provides an intermittent clock to the CPU when register, internal memory, internal resource, or external bus access is performed. This function reduces power consumption by lowering the CPU execution speed while still providing a high-speed clock to internal resources.
The PLL clock multiplier ratio can be set to 1, 2, 3, or 4 by the CS1, 0 bits.

The WS1, 0 bits set the delay time to wait for the main clock oscillation to stabilize when recovering from stop mode or hardware standby mode.

2.12.1 Registers

Low power mode control register

Bit No.	7	6	5	4	3	2	1	0
Address: 0000A0H	STP	SLP	SPL	RST	Reserved	CG1	CG0	Reserved
Read/write	(W)	(W)	(R/W)	(W)	(-)	(R/W)	(R/W)	(-)
Initial value	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)

LPMCR

Clock select register

Bit No.	15	14	13	12	11	10	9	8
Address: 0000A1H	Reserved	MCM	WS1	WS0	Reserved	MCS	CS1	CS0
Read/write	(-)	(R)	(R/W)	(R/W)	(-)	(R/W)	(R/W)	(R/W)
Initial value	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)

CKSCR

2.12.2 Block Diagram

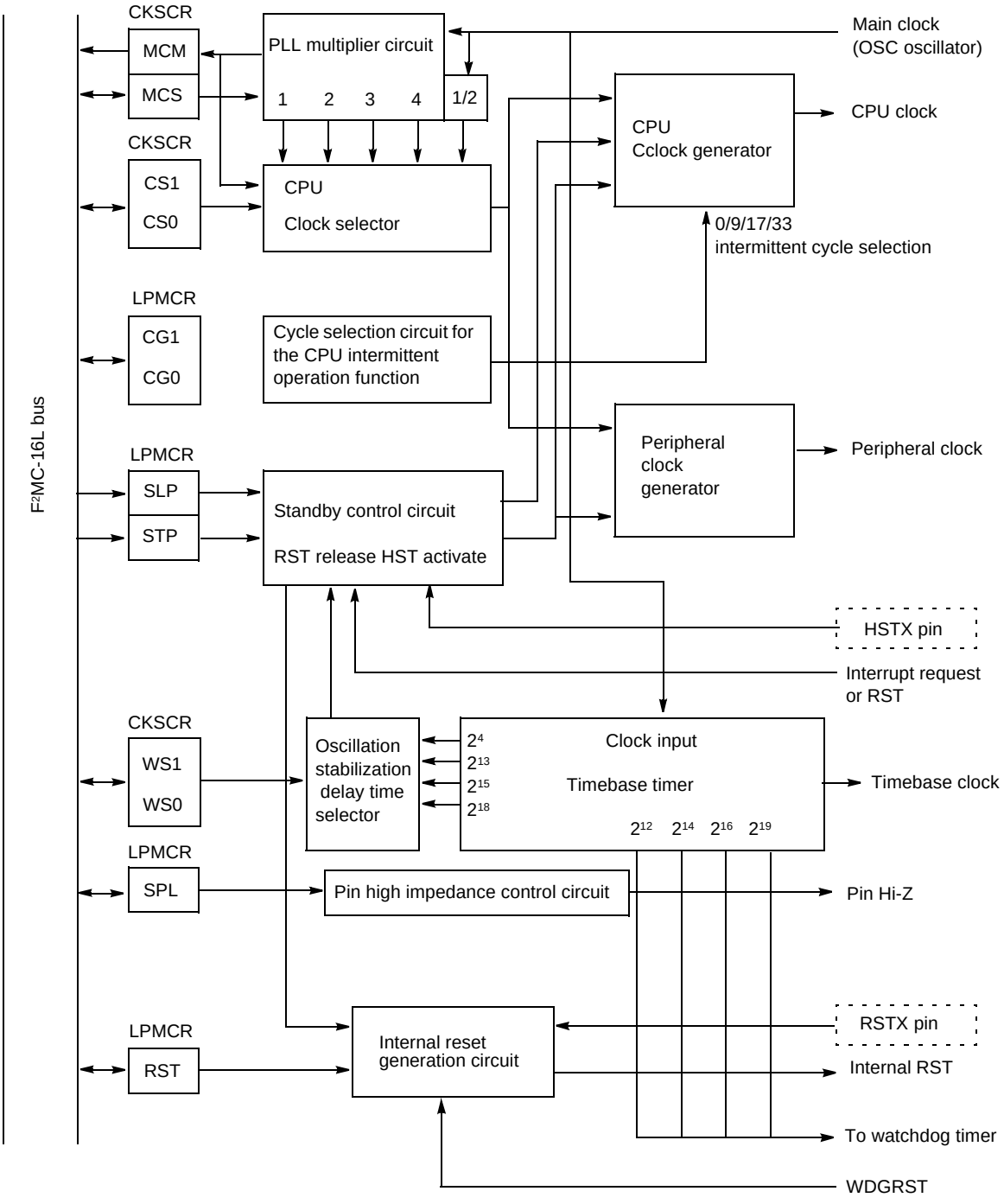


Fig. 2.12.1 Low Power Control Circuit and Clock Generators

2.12.3 Register Details

(1) LPMCR (Low power mode control register)

■ Register layout

Note 1: Do not access this register using read-modify-write instructions as this can cause misoperation.

■ Register contents

[Bit 7] STP

Writing "1" to this bit changes to timer mode (if MCS = "0" in CKSCR) or stop mode (if MCS = "1" in CKSCR). Writing "0" to this bit has no effect. STP is cleared to "0" by clearing reset or watch or stop mode. The bit is write-only. Reading always returns "0".

[Bit 6] SLP

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

The device changes to watch or stop mode if "1" is written to both the STP and SLP bits at the same time. SLP is a write-only bit. Reading always returns "0".

[Bit 5] SPL

External pins hold their levels during timer or stop mode if this bit is "0". External pins go to high impedance during watch or stop mode if this bit is "1". SPL is cleared to "0" by a reset. The bit is readable and writable.

[Bit 4] RST

Writing "0" to this bit generates an internal reset signal for 3 machine cycles. Writing "1" to this bit has no effect. Reading always returns "1".

[Bits 2, 1] CG1, CG0

Sets the number of cycles to temporarily stop the clock for the CPU intermittent operation function.
Initialized to "00" by a power-on, hardware standby, or watchdog timer reset. Not initialized by other resets. The bits are readable and writable.

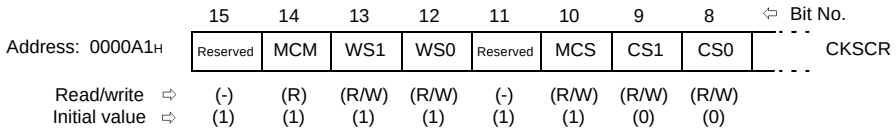
The CPU intermittent operation function halts the CPU clock for a set period when register, internal memory, internal resource, or external bus access is performed and delays the start of the internal bus cycle. This function reduces power consumption by lowering the CPU execution speed while still providing a high-speed clock to internal resources.

Table 2.12.1 CG Bit Settings

CG1	CG0	Number of Cycles to Stop CPU Clock
0	0	0 cycles (CPU clock = resource clock)
0	1	9 cycles (The ratio of the CPU clock to the resource clock is approx. 1:3 to 4)
1	0	17 cycles (The ratio of the CPU clock to the resource clock is approx. 1:5 to 6)
1	1	33 cycles (The ratio of the CPU clock to the resource clock is approx. 1:9 to 10)

(2) CKSCR (Clock selection register)

■ Register layout



■ Register content

[Bit 14] MCM

Indicates whether the main clock or PLL clock is selected as the machine clock. "0" indicates that the PLL clock is selected. "1" indicates that the main clock is selected. MCS = "0" and MCM = "1" indicates that the device is currently waiting for the PLL clock oscillation to stabilize. The delay time to wait for the PLL clock to stabilize is fixed at 2^{13} main clock cycles.

[Bits 13, 12] WS1, WS0

Sets the delay time to wait for the main clock to stabilize after wake-up from stop mode or hardware standby mode, or after a watchdog reset.

Initialized to "11" by a power-on reset. Not initialized by other resets. The bits are readable and writable.

Table 2.12.2 WS Bit Settings

WS1	WS0	Delay Time for Oscillation to Stabilize (For a 4 MHz source clock)
0	0	Do not delay for the main clock to stabilize.
0	1	Approx. 2.05ms (2^{13} source clock counts)
1	0	Approx. 8.19ms (2^{15} source clock counts)
1	1	Approx. 65.54ms (2^{18} source clock counts)

[Bit 10] MCS
Selects the main clock or PLL clock as the machine clock. Writing "0" to this bit selects the PLL clock and writing "1" selects the main clock. Writing "0" when the bit is "1" automatically clears the timebase timer and the TBOF bit in the timebase timer control register so as to generate the delay time for the PLL clock to stabilize. The delay time for the PLL clock to stabilize is fixed at 2^{13} main clock cycles (approximately 2ms for a 4 MHz source clock).

The operating clock when the main clock is selected is the main clock divided by 2. (This gives a 2 MHz operating clock for a 4 MHz source clock.)

MCS is initialized to "1" by a power-on, hardware standby, or watchdog reset.

Note: When the value of MCS is "1", use the TBIE bit or the ILM bits in the CPU to mask the timebase timer interrupt before writing "0" to MCS.

Also, you may not be able to write "0" to the MCS bit for 8 machine cycles after setting the bit to "1". Always wait for at least 8 machine cycles before writing.

[Bits 9, 8] CS1, CS0
Selects the multiplier ratio for the PLL clock. The bits are initialized to "00" by a power-on reset but are not initialized by resets triggered by the external pin or RST bit.
Writing to these bits is not allowed when the MCS bit is "0". To set the CS bits, first set the MCS bit to "1" (main clock mode).
The bits are readable and writable.

Table 2.12.3 CS Bit Settings

CS1	CS0	Machine Clock (For a 4 MHz source clock)
0	0	4 MHz (Operating frequency = OSC oscillation frequency)
0	1	8 MHz (Operating frequency = OSC oscillation frequency * 2)
1	0	12 MHz (Operating frequency = OSC oscillation frequency * 3)
1	1	16 MHz (Operating frequency = OSC oscillation frequency * 4)

2.12.4 Operation

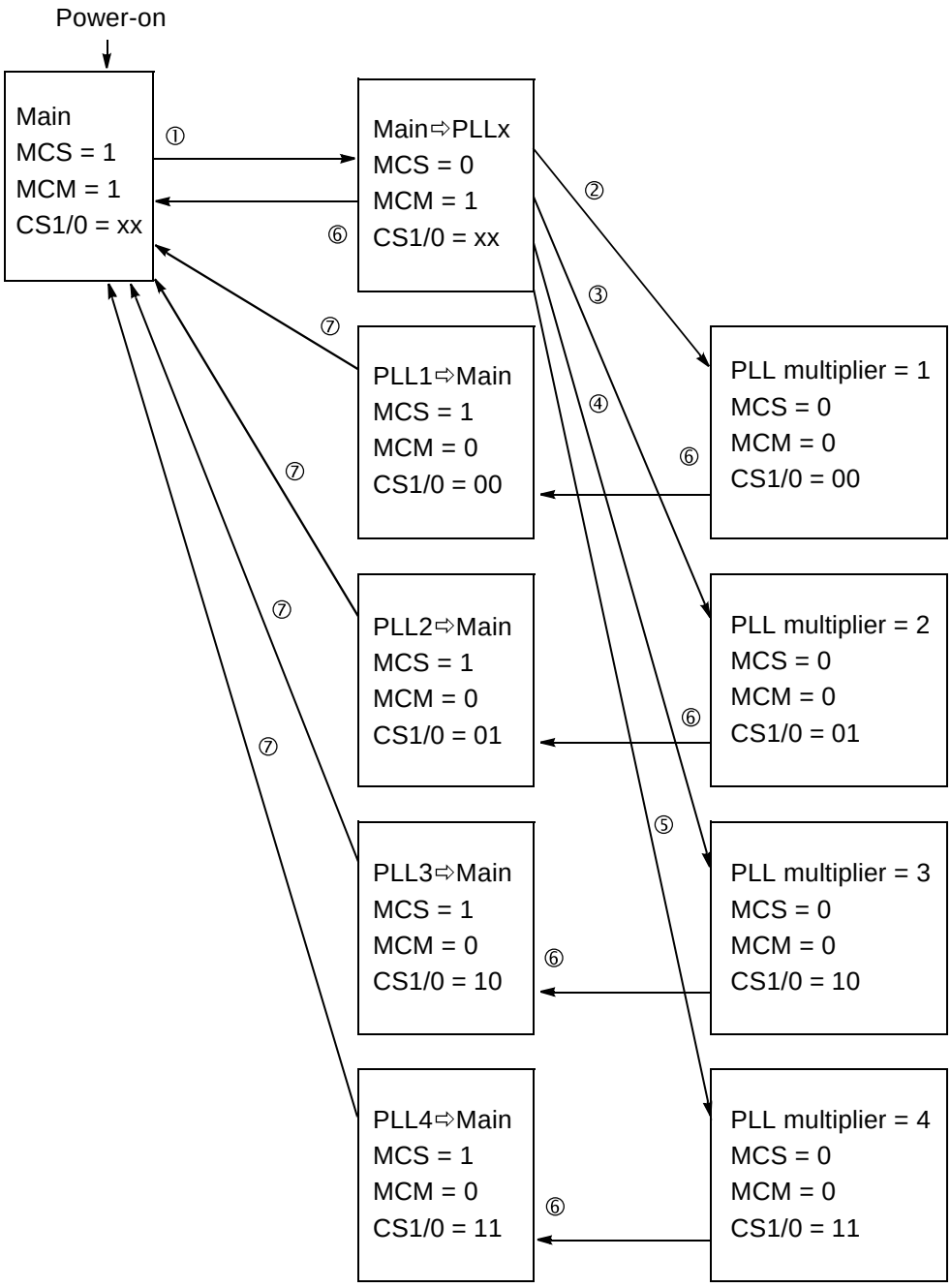
See Section 3.5 "Low Power Modes" for details on the operation of low power consumption modes.

(1) State transitions for clock switching

Figure 2.12.2 shows the state transitions for clock switching.
The oscillation stabilization delay time for the PLL clock is fixed at 2^{13} main clock cycles (approximately 2ms for a 4 MHz source clock).

Note: When operating on 5V, the OSC source clock can be in the range 3MHz to 16 MHz. However, the maximum operating frequency for the CPU and peripheral resource circuits is 16 MHz. The device will not operate correctly if the multiplier setting results in this maximum being exceeded. For example, always set the multiplier to 1 if using a 16 MHz source clock.

Also, the minimum operating frequency for the VCO is 4 MHz. Do not specify an oscillation less than this frequency.



① MCS bit cleared.
② PLL clock oscillation stabilization delay complete and CS1/0 = "00"
③ PLL clock oscillation stabilization delay complete and CS1/0 = "01"
④ PLL clock oscillation stabilization delay complete and CS1/0 = "10"
⑤ PLL clock oscillation stabilization delay complete and CS1/0 = "11"
⑥ MCS bit set (including a hardware standby or watchdog reset)
⑦ PLL clock and main clock synchronized timing

Fig. 2.12.2 State Transition Diagram for Clock Switching

2.13 External Bus Pin Control Circuit

The external bus pin control circuit controls the external bus pins that extend the CPU's address/data bus outside the device.

2.13.1 Registers

Auto-ready function select register

	15	14	13	12	11	10	9	8	⇦ Bit No.
Address: 0000A5H	IOR1	IOR0	HMR1	HMR0	-	-	LMR1	LMR0	ARSR
Read/write ⇨	(W)	(W)	(W)	(W)	(-)	(-)	(W)	(W)	
Initial value ⇨	(0)	(0)	(1)	(1)	(-)	(-)	(0)	(0)	

External address output control register

	7	6	5	4	3	2	1	0	⇦ Bit No.
Address: 0000A6H	E23	E22	E21	E20	E19	E18	E17	E16	HACR
Read/write ⇨	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

Bus control signal select register

	15	14	13	12	11	10	9	8	⇦ Bit No.
Address: 0000A7H	-	LMBS	WRE	HMBS	IOBS	HDE	RYE	CKE	ECSR
Read/write ⇨	(-)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value ⇨	(-)	(0)	(0)	(1/0)	(0)	(0)	(0)	(0)	

2.13.2 Block Diagram

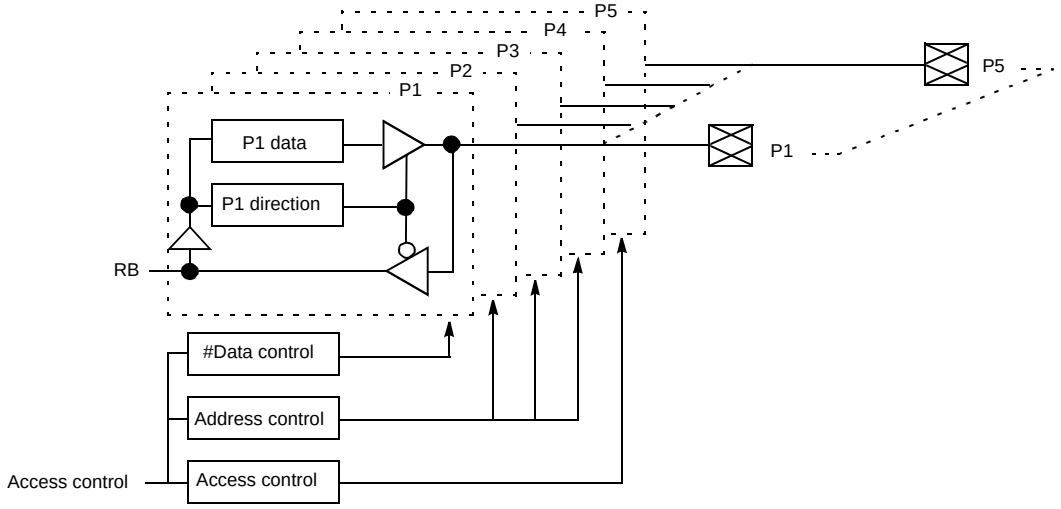


Fig. 2.13.1 External Bus Pin Control Circuit

2.13.3 Register Details

(1) Auto-ready function select register

■ Register layout

■ Register contents

[Bits 15, 14] IOR1, IOR0

These two bits specify the automatic wait function for external access of the area between 0000C0H and 0000FFH. The settings are as follows.

Table 2.13.1 Automatic Wait Function Selection for the External I/O Area

IOR1	IOR0	Setting
0	0	Automatic wait disabled
0	1	Insert an automatic wait of 1 machine cycle during external access.
1	0	Insert an automatic wait of 2 machine cycles during external access.
1	1	Insert an automatic wait of 3 machine cycles during external access.

The bits are initialized to "00B".

[Bits 13, 12] HMR1, HMR0

These two bits specify the automatic wait function for external access of the area between 800000H and FFFFFFH. The settings are as follows.

Table 2.13.2 Automatic Wait Function Selection for the Upper Memory Area

HMR1	HMR0	Setting
0	0	Automatic wait disabled
0	1	Insert an automatic wait of 1 machine cycle during external access.
1	0	Insert an automatic wait of 2 machine cycles during external access.
1	1	Insert an automatic wait of 3 machine cycles during external access.

The bits are initialized to "11B".

[Bits 9, 8] LMR1, LMR0
These two bits specify the automatic wait function for external access of the area between 002000H and 7FFFFFH. The settings are as follows.

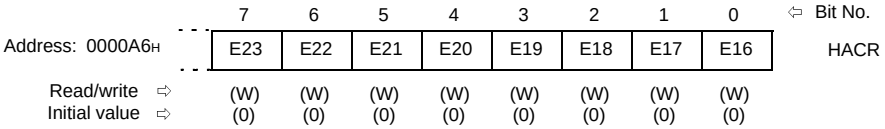
Table 2.13.3 Automatic Wait Function Selection for the Lower Memory Area

LMR1	LMR0	Setting
0	0	Automatic wait disabled
0	1	Insert an automatic wait of 1 machine cycle during external access.
1	0	Insert an automatic wait of 2 machine cycles during external access.
1	1	Insert an automatic wait of 3 machine cycles during external access.

The bits are initialized to "00B".

(2) External address output control register

■ Register layout



■ Register contents
This register controls external output of address bits A23 to A16. Each bit controls its corresponding address bit (A23 to A16) as follows.

Table 2.13.4 Output Control Selection for the Upper Address Bits

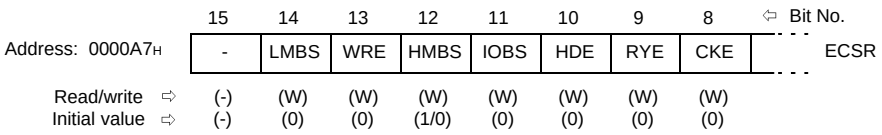
1	Set the corresponding pin as an address output (AXX).
0	Set the corresponding pin as an I/O port (PXX).

This register cannot be accessed when the device is in single-chip mode. In single-chip mode, all the pins are set as I/O ports regardless of the contents of this register.

All bits in this register are write-only. Reading always returns "1".
The bits are initialized to "0" by a reset.

(3) Bus control signal select register

■ Register layout



■ Register contents
This register sets the bus operation control functions for external bus mode.

The register cannot be accessed when the device is in single-chip mode. In single-chip mode, all the pins are set as I/O ports regardless of the contents of this register.

All bits in this register are write-only. Reading always returns "1".

[Bit 14] LMBS
Specifies the bus size for external access of the area between 002000H and 7FFFFFH in 16-bit external data bus mode. The settings are as follows.

Table 2.13.5 Bus Size Selection for the Lower Memory Area

0	16-bit bus size access [Initial value]
1	8-bit bus size access

The bit is initialized to "0" by a reset.

[Bit 13] WRE
Controls the output of the external write signal (the WRHX and WRLX pins in 16-bit bus mode and the WRX pin in 8-bit bus mode) as follows.

Table 2.13.6 Output Control Selection for External Writes

0	Operate as I/O ports (P54, P55). (Output of write signals is disabled.) [Initial value]
1	Enable output of write strobe signals (WRHX/WRLX or WRX)

In 8-bit external data bus mode, P54 operates as an I/O port regardless of the value of this bit.

The bit is initialized to "0" by a reset.

[Bit 12] HMBS
Specifies the bus size for external access of the area between 800000H and FFFFFFH in 16-bit external data bus mode. The settings are as follows.

Table 2.13.7 Bus Size Selection for the Upper Memory Area

0	16-bit bus size access (Initial value for external vector modes 1 and 3)
1	8-bit bus size access (Initial value for external vector modes 0 and 2)

In external vector modes 0 and 2, the bit is initialized to "1" by a reset. In external vector modes 1 and 3, the bit is initialized to "0" by a reset.

[Bit 11] IOBS
Specifies the bus size for external access of the area between 0000C0H and 0000FFH in 16-bit external data bus mode. The settings are as follows.

Table 2.13.8 Bus Size Selection for the External I/O Area

0	16-bit bus size access [Initial value]
1	8-bit bus size access

The bit is initialized to "0" by a reset.

[Bit 10] HDE
This bit enables or disables I/O for the hold request input (HRQ) and hold acknowledge output (HAKX) pins as follows.

Table 2.13.9 Hold I/O Control Selection

0	Operate as I/O ports (P53, P52). (Disable hold function I/O.) [Initial value]
1	Enable hold request (HRQ) input and hold acknowledge (HAKX) output

The bit is initialized to "0" by a reset.

[Bit 9] RYE
Controls the external ready (RDY) input as follows.

Table 2.13.10 Input Control Selection for the External Ready

0	Operate as an I/O port (P51). (Disable the external RDY input.) [Initial value]
1	Enable the external ready (RDY) input.

The bit is initialized to "0" by a reset.

[Bit 8] CKE
Controls the external clock (CLK) output as follows.

Table 2.13.11 Output Control Selection for the External Clock

0	Operate as an I/O port (P50). (Disable clock output.)
1	Enable output of the clock signal (CLK).

The bit is initialized to "0" by a reset.

2.13.4 Operation

The MB90610A has various modes for different access methods and access areas.
See 3.3 "Memory Access Modes" for details.

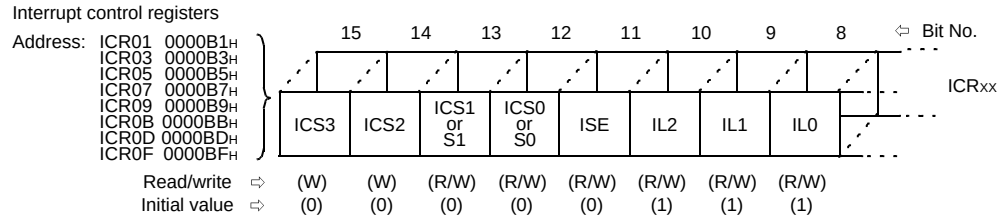
See 3.4 "External Memory Access" for details on the operation of external bus mode.

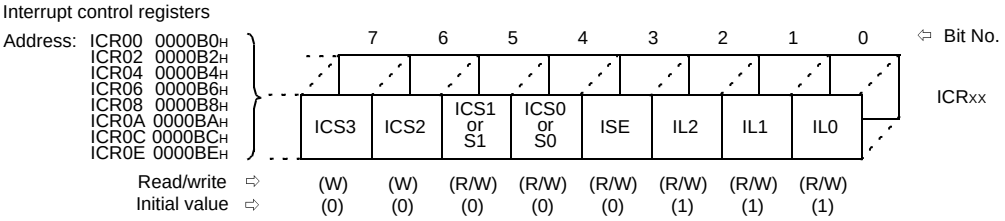
2.14 Interrupt Controller

The interrupt control registers are located in the interrupt controller. An interrupt control register is provided for each I/O with an interrupt function. The registers have the following three functions.

- Set the interrupt level of the corresponding peripheral.
- Select whether to treat interrupts from the corresponding peripheral as standard interrupts or to activate the extended intelligent I/O service.
- Select the extended intelligent I/O service channel.

2.14.1 Registers





Note: Do not access these registers using read-modify-write instructions as this can cause misoperation.

2.14.2 Block Diagram

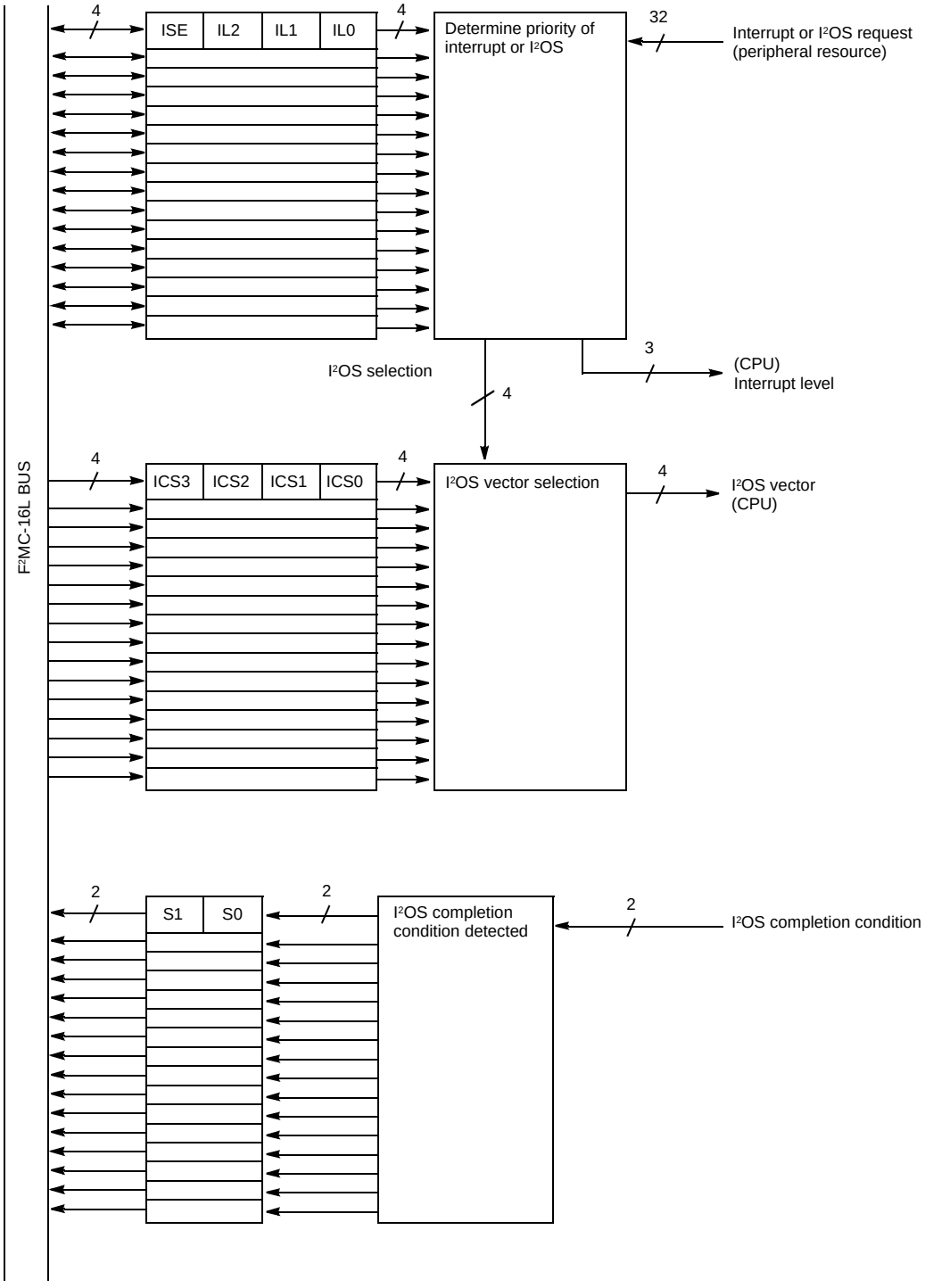
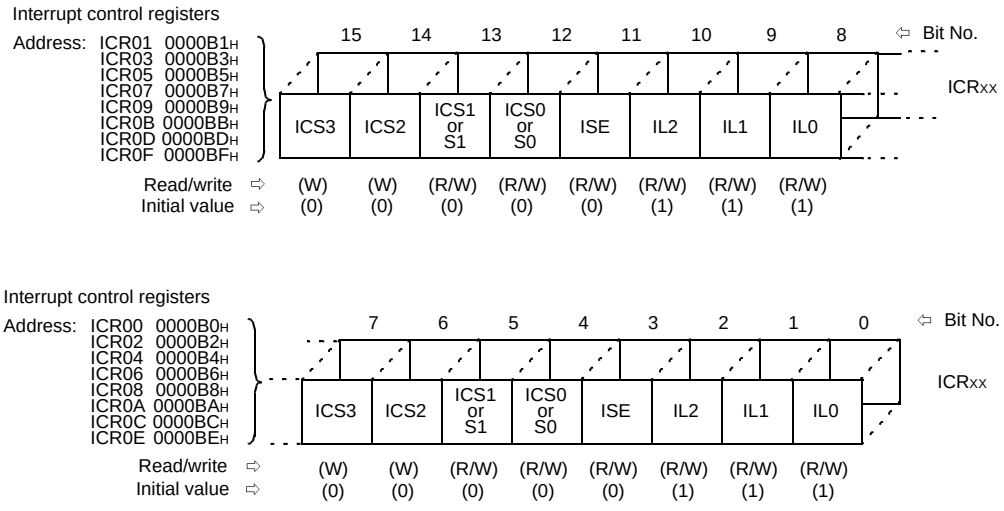


Fig. 2.14.1 Block Diagram of the Interrupt Controller

2.14.3 Register Details

(1) ICR (Interrupt control register)

■ Register layout



Note: ICS3 to 0 are only meaningful when EI²OS is active. Set ISE to "1" when using EI²OS and set ISE to "0" when not using EI²OS. Any values can be set to ICS3 to 0 if EI²OS is not used.
* Reading returns "1".

Note: ICS1 and ICS0 are write-only. S1 and S0 are read-only.

Note: Do not access these registers using read-modify-write instructions as this can cause misoperation.

■ Register contents

(1) Interrupt level setting bits: IL0, IL1, IL2

The interrupt level setting bits specify the interrupt level of the corresponding internal resource. The bits are readable and writable. The bits are initialized to level 7 (no interrupt) by a reset. Table 2.14.1 lists the relationship between the interrupt level setting bits and each interrupt level.

Table 2.14.1 Correspondence Between 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 enable bit: ISE

The processor activates EI²OS if the ISE bit is "1" when an interrupt request occurs. If the ISE bit is "0", the processor activates the interrupt sequence. The bit is readable and writable. The ISE bit changes to "0" when EI²OS completes. If the peripheral does not have an EI²OS function, set the ISE bit to "0" by software.

Initialized to "0" by a reset.

(3) Extended intelligent I/O service channel select bits: ICS3 to 0

These bits specify the EI²OS channel. The bits are write-only. The value set in these bits determines the memory address of the intelligent I/O service descriptor, as described later. ICS is initialized by a reset. Table 2.14.2 lists the correspondence between ICS, the channel number, and the descriptor address.

Table 2.14.2 Correspondence Between ICS, Channel Number, and Descriptor Address

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

(4) Extended intelligent I/O service completion status: S0, S1

When EI²OS completes, these bits can be referenced to determine the completion status. The bits are read-only. The bits are initialized to "00" by a reset.
Table 2.14.3 lists the relationship between the S bits and the completion status.

Table 2.14.3 S Bits and Completion Status

S1	S0	Completion Status
0	0	Reserved
0	1	Stopped due to count completion
1	0	Reserved
1	1	Stopped by request from the internal resource

(5) Interrupt vector allocation

Table 2.14.4 lists the allocation of MB90610A interrupt vectors.

Table 2.14.4 MB90610A Interrupt Vector Allocation

Interrupt	I²OS Support	Interrupt Vector			Interrupt Control Register	
		Number		Address	ICR	Address
Reset	x	#08	08H	FFFFDCH	-	-
INT 9 instruction	x	#09	09H	FFFFD8H	-	-
Exception	x	#10	0AH	FFFFD4H	-	-
External interrupt #0	o	#11	0BH	FFFFD0H	ICR00	0000B0H
External interrupt #1	o	#13	0DH	FFFFC8H	ICR01	0000B1H
External interrupt #2	o	#15	0FH	FFFFC0H	ICR02	0000B2H
External interrupt #3	o	#17	11H	FFFFB8H	ICR03	0000B3H
External interrupt #4	o	#19	13H	FFFFB0H	ICR04	0000B4H
External interrupt #5	o	#21	15H	FFFFA8H	ICR05	0000B5H
External interrupt #6	o	#23	17H	FFFFA0H	ICR06	0000B6H
UART0 transmit complete	o	#24	18H	FFFF9CH		
External interrupt #7	o	#25	19H	FFFF98H	ICR07	0000B7H
UART1 transmit complete	o	#26	1AH	FFFF94H		
PPG #0	x	#27	1BH	FFFF90H	ICR08	0000B8H
PPG #1	x	#28	1CH	FFFF8CH		
16-bit reload timer #0	o	#29	1DH	FFFF88H	ICR09	0000B9H
16-bit reload timer #1	o	#30	1EH	FFFF84H		
A/DC measurement complete	o	#31	1FH	FFFF80H	ICR10	0000BAH
UART2 transmit complete	o	#33	21H	FFFF78H	ICR11	0000BBH
Timebase timer interval interrupt	x	#34	22H	FFFF74H		
UART2 receive complete	◎	#35	23H	FFFF70H	ICR12	0000BCH
UART1 receive complete	◎	#37	25H	FFFF68H	ICR13	0000BDH
UART 0receive complete	◎	#39	27H	FFFF60H	ICR14	0000BEH
Delay interrupt generation module	x	#42	2AH	FFFF54H	ICR15	0000BFH

Note: ○ indicates I²OS support (no stop request), ◎ indicates I²OS support (with stop request), and × indicates no I²OS support.

Do not set I²OS activation in ICRxx for resources that do not support I²OS.

2.14.4 Operation

See 3.3 "Interrupts" for details on the operation of interrupts and EI²OS.

Chapter 3: Operation

3.1 Clock Generator

The clock generator controls operation of the internal clock including the PLL clock multiplier, sleep, timer, and stop functions. The internal clock is called the machine clock and one clock cycle is called a machine cycle. The clock from the source oscillator is called the main clock and the clock from the internal VCO is called the PLL clock.

Note: When operating on 5V, the OSC source oscillation can be in the range 3MHz to 16 MHz. However, the maximum operating frequency for the CPU and peripheral resource circuits is 16 MHz. The device will not operate correctly if the multiplier setting results in this maximum being exceeded. For example, always set the multiplier to 1 if using a 16 MHz source oscillation.

Also, the minimum operating frequency for the VCO is 4 MHz. Do not specify an oscillation less than this frequency.

Figure 3.1.1 shows a block diagram of the clock generator circuit.

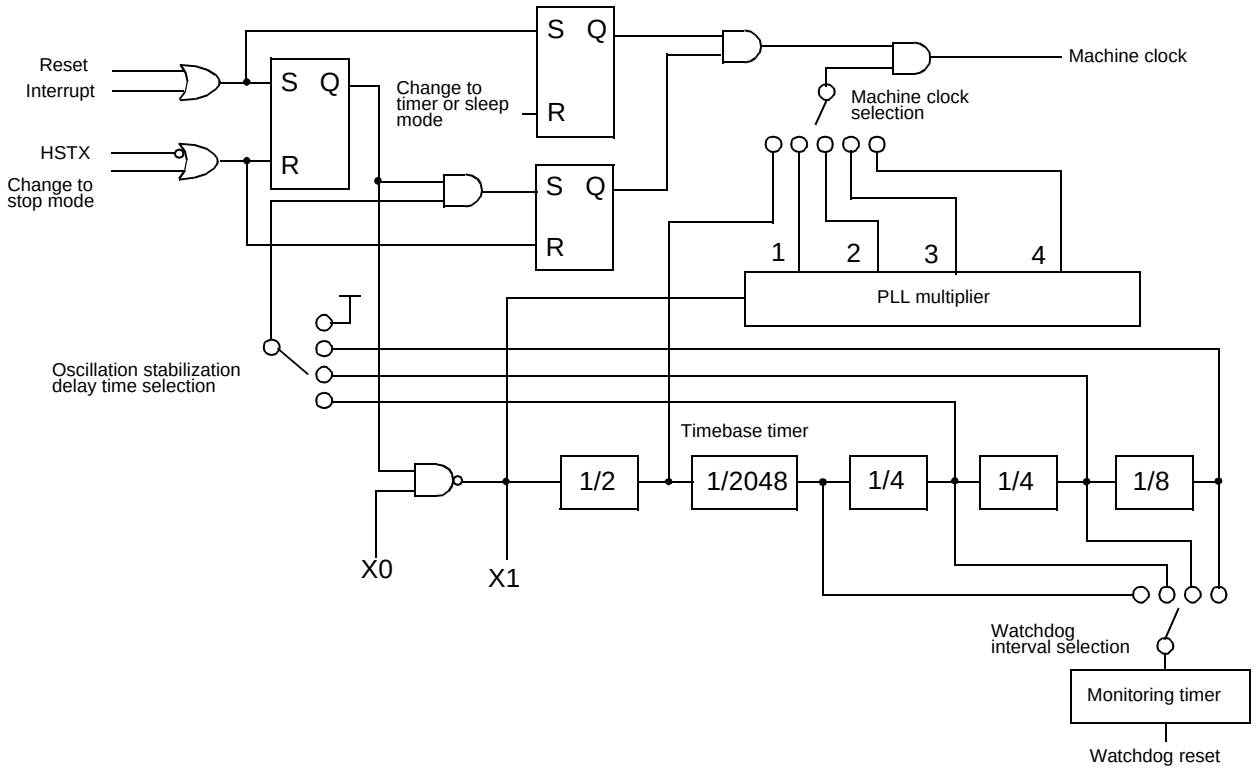


Fig. 3.1.1 Block Diagram of the Clock Generator Circuit

3.2 Resets

3.2.1 Generating a Reset

On generation of a reset source, the F²MC-16L interrupts the currently executing processing and waits for the reset to be cleared. The following events trigger a reset.

- Generation of a power-on reset
- Release of the hardware standby state
- Watchdog timer overflow
- External reset request from the RSTX pin
- Software reset request

After a power-on reset or wake-up from stop mode, operation starts after a delay for the oscillation to stabilize.

Note: Other than in stop mode, sampling of the external reset input is synchronized by the internal clock. Therefore, reset inputs cannot be detected if the externally supplied clock is halted.

When using an external bus, the address generated by the device during a reset is indeterminate. All the external bus access signals (RDX, WRLX, etc.) become inactive.

3.2.2 Operation After the Reset is Cleared

On removal of the reset trigger, the F²MC-16L outputs the address of the reset vector and reads the reset vector and mode data. The reset vector and mode data are located in the 4 bytes from FFFFDCH to FFFFDFH. On release of the reset, the data at these locations are moved by hardware to the registers shown in Figure 3.2.1.

The bus mode after reading the reset vector and mode data is determined by the mode data.

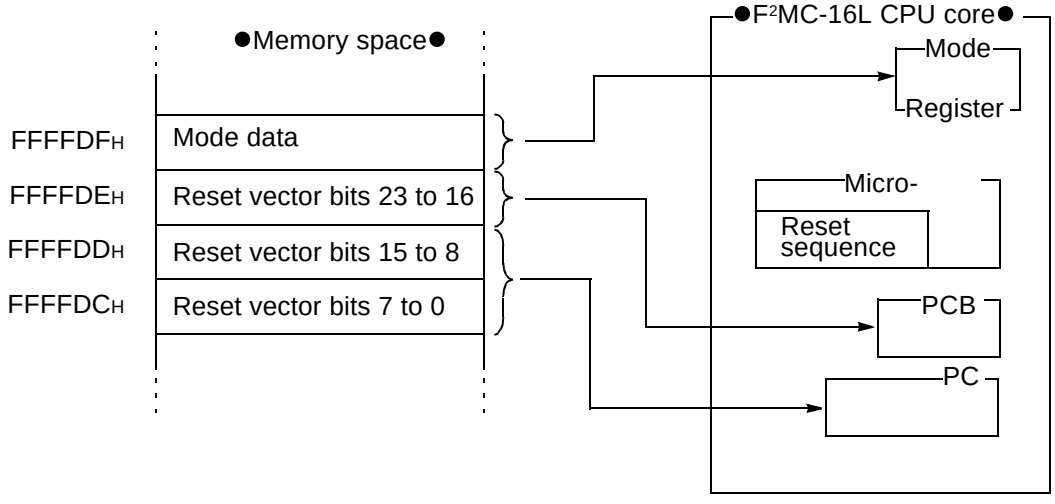


Fig. 3.2.1 Storage Location and Destination of the Reset Vector and Mode Data

Note: The mode register shown in the diagram becomes indeterminate immediately after a reset. Always store the optional mode data in memory so that the data can be set in the register.

3.2.3 Reset Types

Table 3.2.1 lists the five different reset types. The machine clock and watchdog timer initialization depends on the reset type.

The reset type can be determined from the reset type register.

Table 3.2.1 Reset Types

Reset	Reset Cause	Machine Clock	Watchdog Timer	Delay for Oscillation to Stabilize?
Power-on	Power supply turned on	Main clock	Stopped	Yes
Hardware standby	Input of an "L" level to the HSTX pin	Main clock	Stopped	Yes
Watchdog timer	Watchdog timer overflow	Main clock	Stopped	Yes
External pin	Input of an "L" level to the RSTX pin	Holds the previous state	Holds the previous state	No
Software	Writing "0" to the RST bit in the STBYC register	Holds the previous state	Holds the previous state	No

∗ If a reset occurs in stop or hardware standby mode, the device waits for oscillation to stabilize regardless of the reset type.

∗ The delay for oscillation to stabilize after a power-on reset is fixed at 2^{18} source oscillator cycles. For other resets, the delay for oscillation to stabilize depends on CS1 and CS0 in the clock selection register.

Figure 3.2.2 shows the flip-flops for the various reset types. The content of these flip-flops can be read from the watchdog timer control register. Therefore, if it is necessary to determine the reset cause after the reset is cleared, read the watchdog timer control register by software and branch to the appropriate program based on the register content. For reference, Figure 3.2.3 shows the structure of the watchdog timer control register.

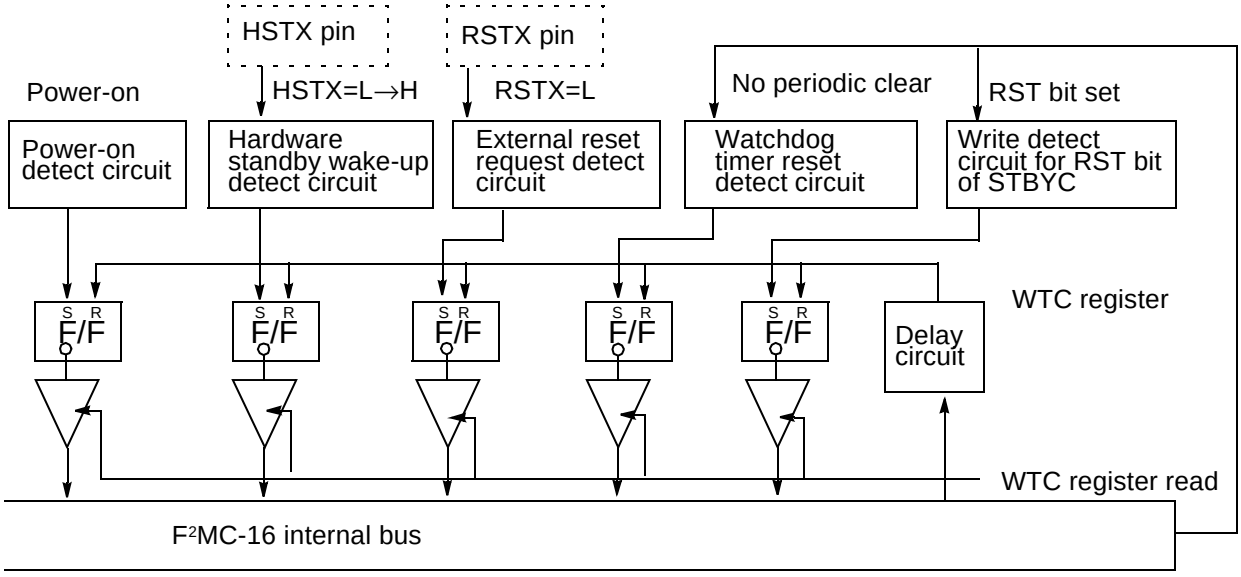


Fig. 3.2.2 Block Diagram of the Reset Type Bits

	7	6	5	4	3	2	1	0	⇦Bit No.
Address: 0000A8H	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)	

Fig. 3.2.3 WDTC (Watchdog Timer Control Register)

If more than one reset type occurs, the reset type bits are set in the watchdog timer control register for each reset type that occurred. Accordingly, if an external reset request and watchdog reset occur simultaneously, both the ERST and WRST bits are set to"1".

However, the power-on reset is an exception to this rule. When the PONR bit is "1", the values of the other bits do not indicate correctly whether their respective reset types occurred or not. Therefore, when developing software, ignore the values of the other reset type bits if the PONR bit is "1".

Table 3.2.2 Value of the Reset Type Bits for Each Reset Type

Reset Type	PONR	STBR	WRST	ERST	SRST
Power-on	1	-	-	-	-
Hardware standby	*	1	*	*	*
Watchdog timer	*	*	1	*	*
External pin	*	*	*	1	*
RST bit	*	*	*	*	1

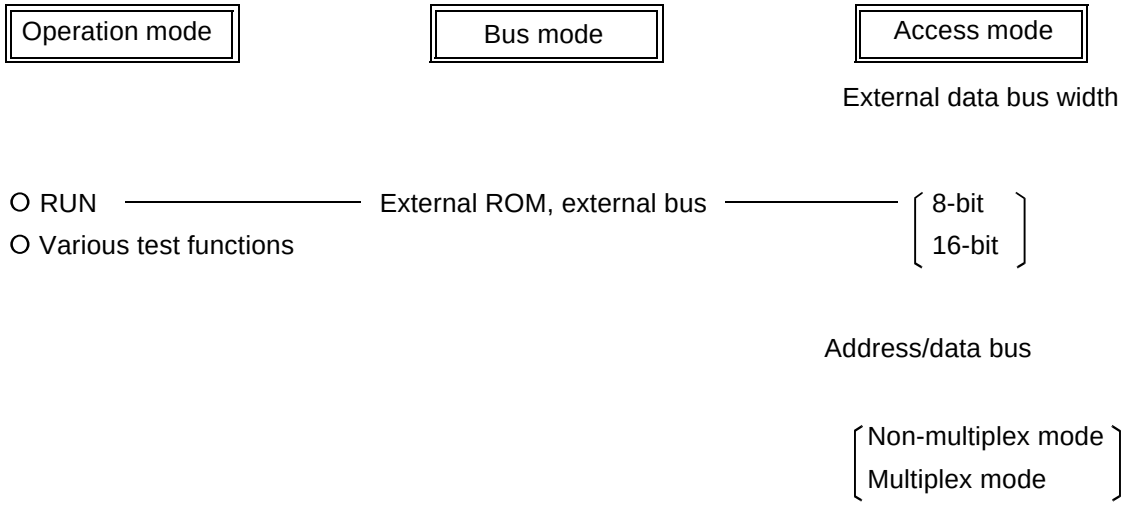
(The values indicated by * hold their previous values.)

The reset type bits are only cleared by reading the watchdog timer control register. Therefore, once a particular reset type occurs, the corresponding reset type bit maintains the value "1", even if a different type of reset subsequently occurs.

3.3 Memory Access Modes

3.3.1 Modes

The F²MC-16L has various modes for the different access methods, access areas, and test operation. The modes for this module are classified as follows.



■ Operation mode

The operation mode controls the device operating status and is set by the mode setting pins (MD×) and the M× bits in the mode data. The operation mode can select normal operation, activation of the internal test program, or activation of the special test functions.

■ Bus mode

The bus mode controls the operation of the external access function. The mode setting pins (MD×) and the M× bits in the mode data determine the bus mode. The mode setting pins (MD×) specify the bus mode for reading the reset vector and mode data. The M× bits in the mode data specify the bus mode for normal operation.

■ Access mode

The access mode specifies the external data bus width and the address/data bus operation. The mode setting pins (MD×) and the S× bit in the mode data determine the access mode. The access mode specifies the external data bus width (8-bit or 16-bit) and whether the address/data bus operates in non-multiplex or multiplex mode.

3.3.2 Mode Pins

Table 3.3.1 lists the operations specified by the different MD2 to MD0 external pin combinations.

Table 3.3.1 Relationship Between Mode Pins and Setting Modes

Mode Pin Settings MD2 MD1 MD0	Mode	Reset Vector Access Area	External Data Bus Width	Address/ Data Bus	Remarks
0 0 0	External vector mode 0	External	8-bit	Multiplex mode	
0 0 1	External vector mode 1	External	16-bit		Reset vector is accessed via a 16-bit bus.
0 1 0	External vector mode 2	External	8-bit	Non-multiplex mode	
0 1 1	External vector mode 3	External	16-bit		Reset vector is accessed via a 16-bit bus.
1 0 0	(Prohibited settings)				
1 0 1					
1 1 0					
1 1 1					

3.3.3 Mode Data

The mode data controls CPU operation and is located at FFFFDFH in main memory. The reset sequence reads the mode data and loads it to the "mode register" in the device. Only the reset sequence can change the content of the mode register.

The settings in the mode register apply after the completion of the reset sequence.

Always set the reserved bits to "0".

Figure 3.3.1 shows the settings for each bit.

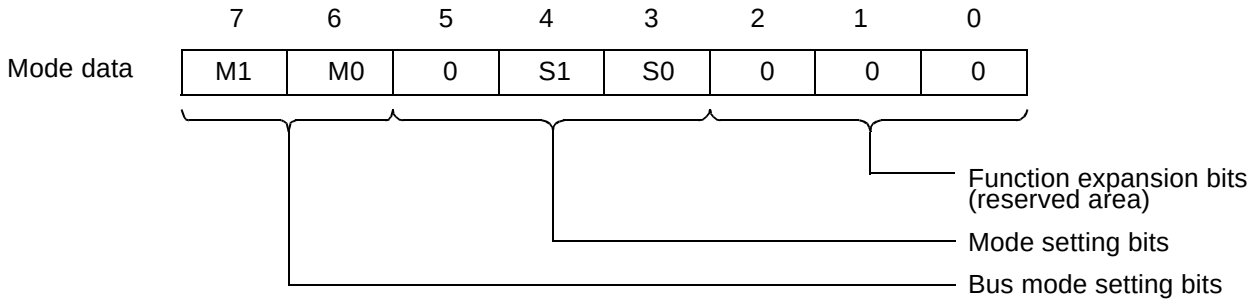


Fig. 3.3.1 FMode Data Structure

■ Mode setting bits

These bits specify the bus mode and access mode after completion of the reset sequence. Table 3.3.3 lists the functions set by the mode setting bits.

Table 3.3.3 Functions Set by the Mode Setting Bits

S1	S0	Function		Remarks
0	0	8-bit external data bus mode	Address/data bus multiplex mode	
0	1	16-bit external data bus mode		
1	0	8-bit external data bus mode	Address/data bus non-multiplex mode	
1	1	16-bit external data bus mode		

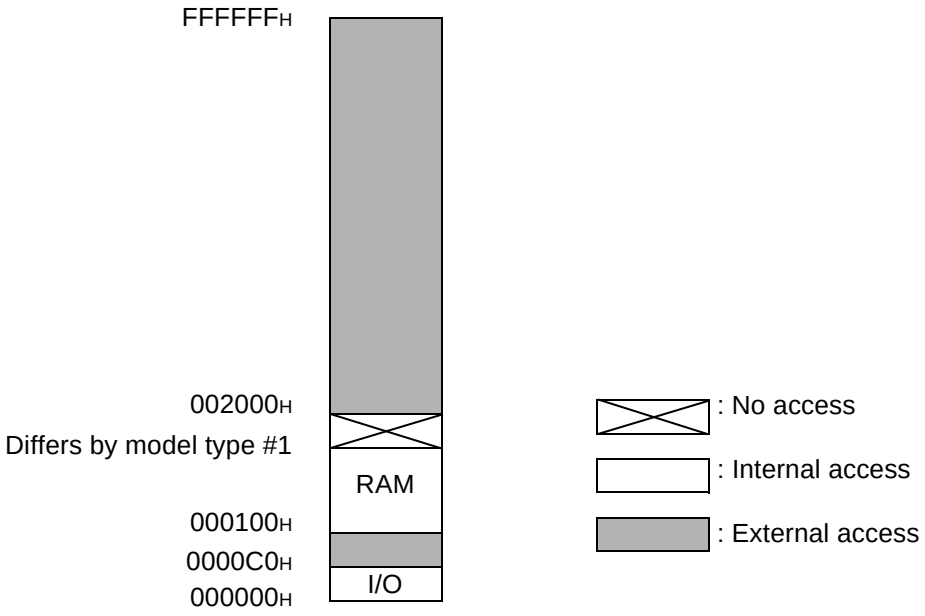
■ Bus mode setting bits

These bits specify the operation mode after completion of the reset sequence. Table 3.3.4 lists the functions set by the bus mode setting bits.

Table 3.3.4 Functions Set by the Bus Mode Setting Bits

M1	M0	Function	Remarks
0	0	(Prohibited setting)	
0	1	(Prohibited setting)	
1	0	External ROM, external bus mode	
1	1	(Prohibited setting)	

Figure 3.3.2 shows the correspondence between the access area and physical address for each bus mode setting.



Note: The addresses marked "Differs by model type" depend on the model type.

Fig. 3.3.2 Relationship Between Access Areas and Physical Addresses for Each Bus Mode

■ Recommended setting example

Table 3.3.5 shows examples of recommended mode pin and mode data settings.

Table 3.3.5 Recommended Mode Pin and Mode Data Settings

Setting Example	MD2	MD1	MD0	M1	M0	S1	S0
External ROM/external bus mode, 16-bit bus, 16-bit bus for vector access (Address/data bus multiplex mode)	0	0	1	1	0	0	1
External ROM/external bus mode, 8-bit bus (Address/data bus multiplex mode)	0	0	0	1	0	0	0
External ROM/external bus mode, 16-bit bus, 16-bit bus for vector access (Address/data bus non-multiplex mode)	0	1	1	1	0	1	1
External ROM/external bus mode, 8-bit bus (Address/data bus non-multiplex mode)	0	1	0	1	0	1	0

Note: In the MB90610A series, a maximum area of 64KB can be accessed when output of the upper address (A23 to A16) is disabled.

The signals input or output via the external pins connected to this module vary depending on the mode. Table 3.3.5 lists the operation of the mode-dependent external pins.

Table 3.3.5 Operation of Mode-Dependent External Pins

Pin	Function							
	Non-Multiplex Mode				Multiplex Mode			
	External Address Control				External Address Control			
	Enabled (Address)		Disabled (Port)		Enabled (Address)		Disabled (Port)	
	External Bus Expansion		External Bus Expansion		External Bus Expansion		External Bus Expansion	
	8-Bit	16-Bit	8-Bit	16-Bit	8-Bit	16-Bit	8-Bit	16-Bit
D07 to 00/ AD07 to 00	D07 to 00				AD07 to 00			
P17 to 10/ D15 to 08 AD15 to 08	Port	D15 to 08	Port	D15 to 08	A15 to 08	AD15 to 08	A15 to 08	AD15 to 08
P27 to 20/ A07 to 00	A07 to 00		A07 to 00		Port			
P37 to 30/ A15 to 08	A15 to 08		A15 to 08					
P47 to 40/ A23 to 16	A23 to 16		Port		A23 to 16		Port	
ALE	ALE				ALE			
RDX	RDX				RDX			
P55/WRLX	WRLX				WRLX			
P54/WRHX	Port	WRHX	Port	WRHX	Port	WRHX	Port	WRHX
P53/HRQ	HRQ				HRQ			
P52/HAKX	HAKX				HAKX			
P51/RDY	RDY				RDY			
P50/CLK	CLK				CLK			

Note: The upper address, WRLX, WRHX, HAKX, HRQ, RDY, and CLK can be set for use as ports by function selection.

3.4 External Memory Access

The F²MC-16L provides the following address, data, and control signals for accessing external memory or peripherals.

- CLK (P50): Outputs the machine cycle clock (KBP)
- RDY (P51): External ready input pin
- WRHX (P54): Write signal for the upper 8 bits of the data bus
- WRLX (P55): Write signal for the lower 8 bits of the data bus
- RDX: Read signal
- ALE: Address latch enable signal (in multiplex mode)

3.4.1 External Memory Access Control Signals

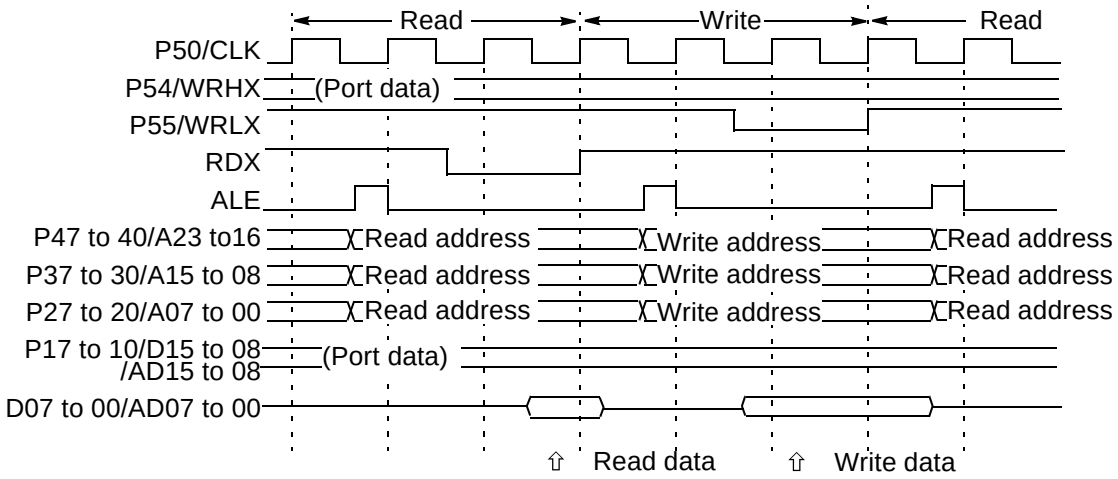
External memory access is performed in 3 cycles if the ready function is not used. Figure 3.4.1 shows an overview of the external access timings.

A function is available to perform 8-bit bus access in 16-bit external bus mode. This allows reading and writing of 8-bit peripheral chips in systems where both 8 and 16-bit peripheral chips are connected to the external bus. As 8-bit bus access uses the lower 8 bits of the data bus, connect 8-bit peripheral chips to the lower 8 bits of the data bus.

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

In multiplex mode, it is possible that only the address output and ALE assert output are performed and, by not asserting RDX, WRLX, and WRHX, actual bus operation is not performed. Do not perform peripheral chip access using the ALE signal only.

❑ External 8-bit bus mode (Non-multiplex mode)



❑ External 8-bit bus mode (Multiplex mode)

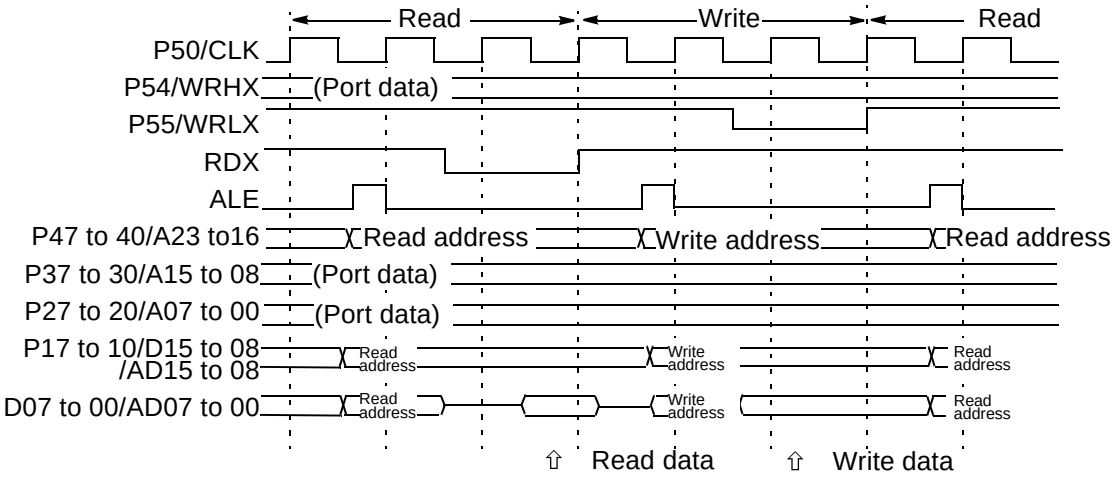
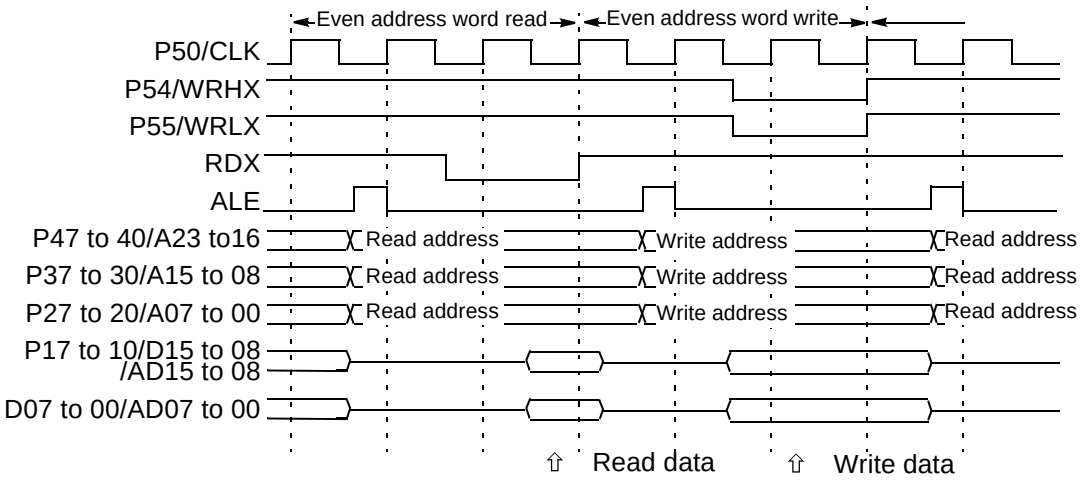


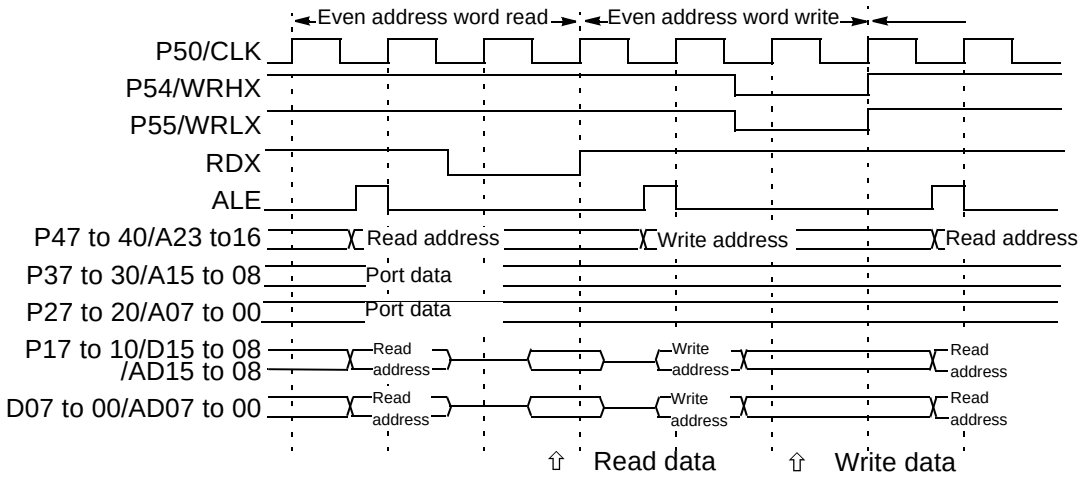
Fig. 3.4.1 Timing Chart for External Memory Access

❑ External 16-bit bus mode (Non-multiplex mode)



∗Design external circuits to always perform word reads.

❑ External 16-bit bus mode (Multiplex mode)



∗Design external circuits to always perform word reads.

Fig. 3.4.1 Timing Chart for External Memory Access

3.4.2 Ready Function

Access to low-speed memory or peripheral circuits can be performed by using the P51/RDY pin and by setting the auto-ready function selection register (ARSR).

When the RYE bit in the bus control signal selection register (EPCR) is set to "1", the access cycle can be extended during access of an external area by treating the period while an "L" level is input to the P51/RDY pin as wait cycles.

❑ Non-multiplex mode

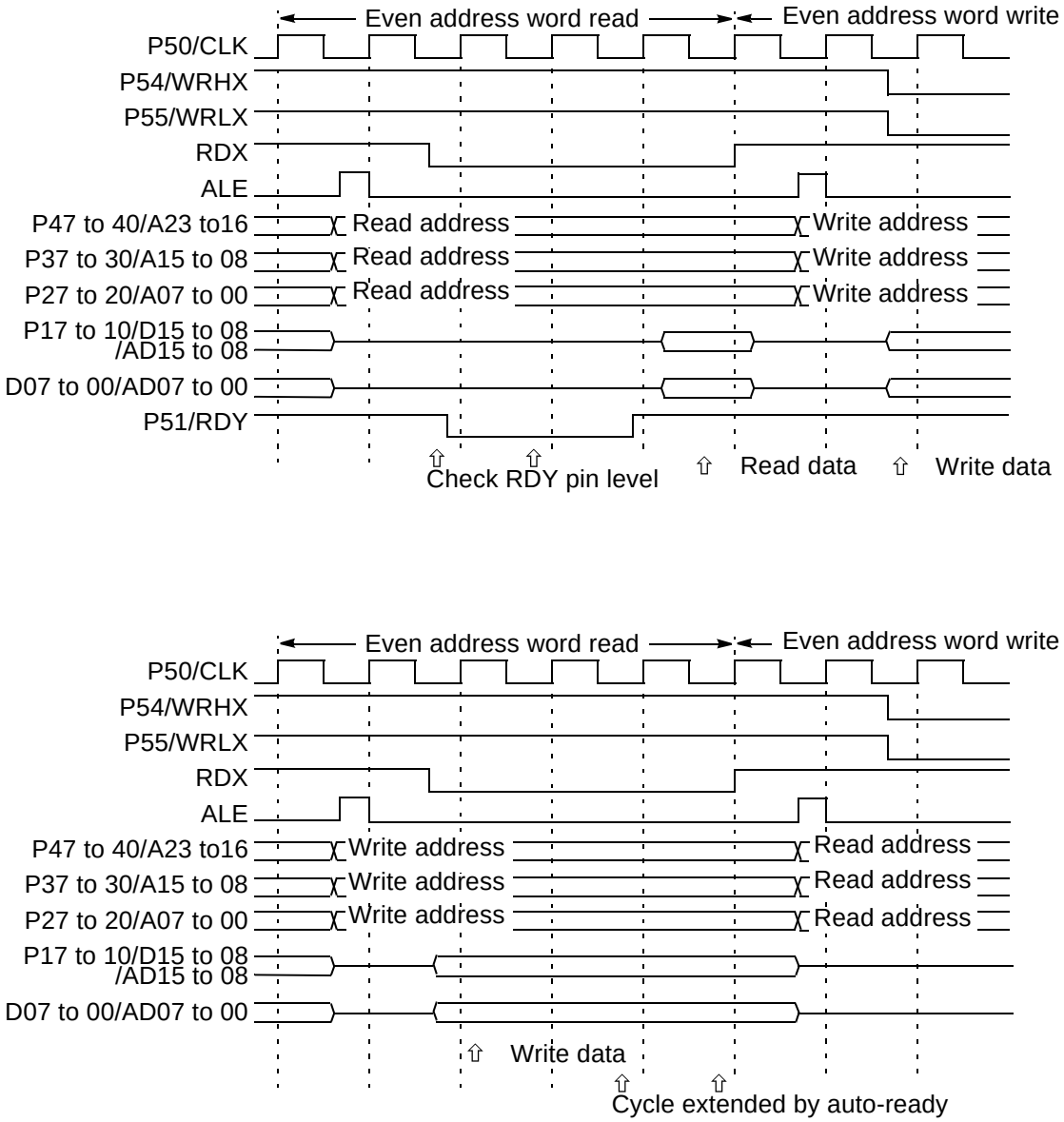


Fig. 3.4.2 Ready Timing Chart

❑ Multiplex mode

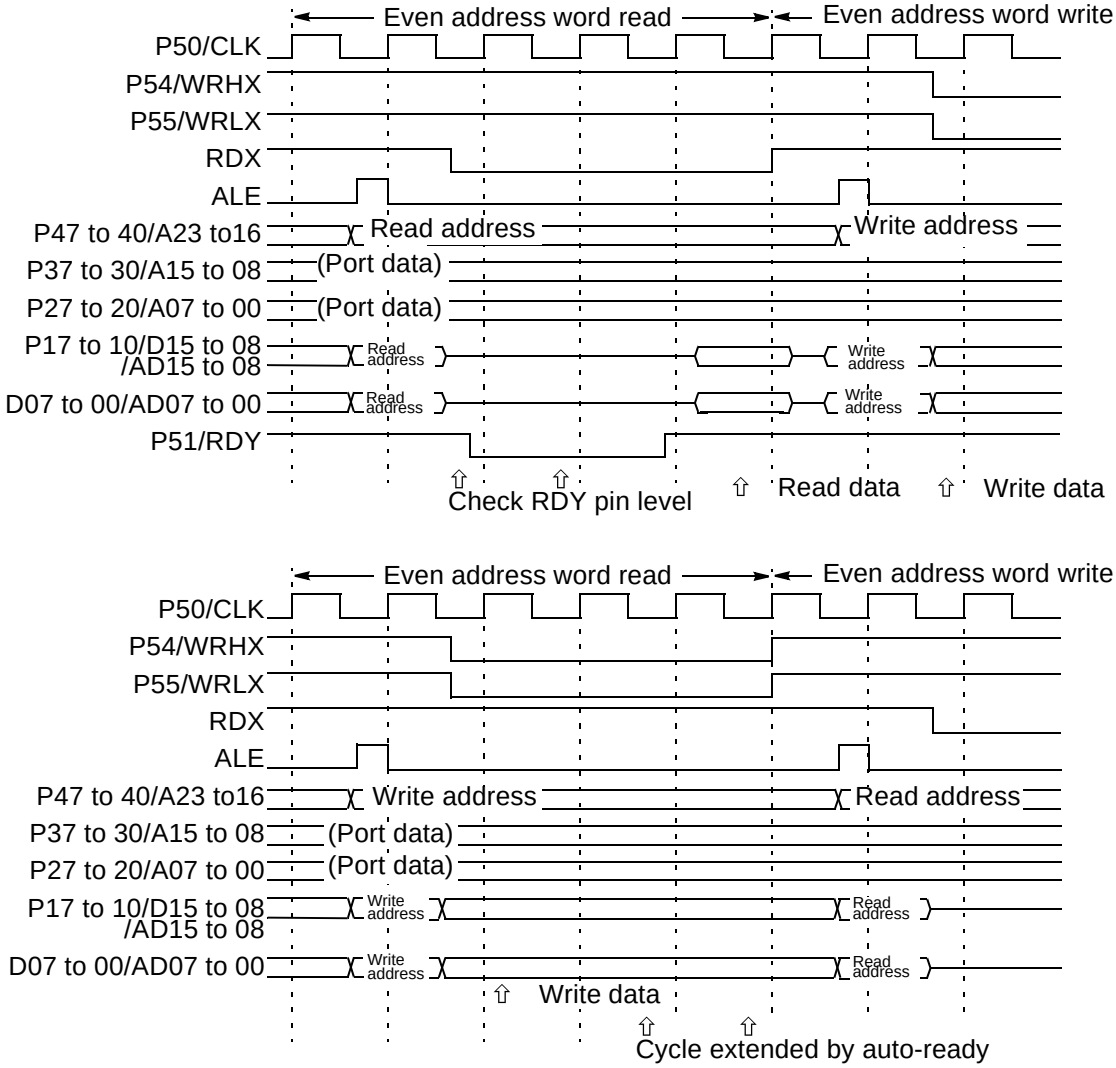


Fig. 3.4.2 Ready Timing Chart

The F²MC-16L has two built-in auto-ready functions for external memory. Auto-ready functions are provided for accessing the lower address external area located between addresses 002000H and 7FFFFFH and for accessing the upper address external area located between addresses 800000H and FFFFFFH. The auto-ready function can extend the access cycle automatically by inserting between 1 and 3 wait cycles, without the need for an external circuit. This function is activated by setting the LMR1 and LMR0 bits (for the lower external address area) and the HMR1 and HMR0 bits (for the upper external address area) in the ARSR register.

The F²MC-16L also has a built-in auto-ready function for external I/O that is independent of the auto-ready function for memory. The auto-ready function can extend the access cycle automatically by inserting between 1 and 3 wait cycles, without the need for an external circuit, when an external area between addresses 0000C0H and 0000FFH is accessed. The function is activated by the IOR1 and IOR0 bits of the ARSR register.

For both the external memory and external I/O auto-ready functions, if the RYE bit in the EPCR register is set to "1", the wait cycles continue after the wait cycles inserted by the auto-ready function are complete for as long as an "L" level is input to the P51/RDY pin.

Figure 3.4.3 shows the structure of the ARSR and EPCR registers.

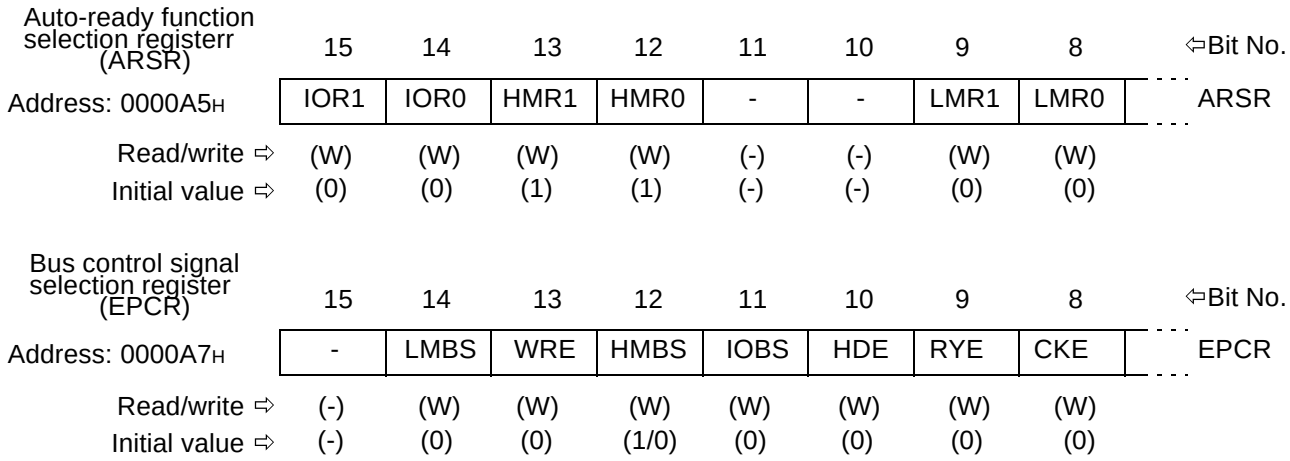


Fig. 3.4.3 Structure of the Auto-Ready Function Selection Register and Bus Control Signal Selection Register

3.4.3 Hold Function

The external bus hold function is activated by the P53/HRQ and P52/HAKX pins if the HDE bit of the EPCR register is set to "1". Inputting an "H" level to the P53/HRQ pin initiates the hold state after completion of the current CPU instruction (or after completing one data element in a string instruction). In the hold state, the device outputs an "L" level from the P52/HAKX pin and sets the following pins to high impedance.

○ Non-multiplex mode
- Address output P47/A23 to P40/A16, P37/A15 to P30/A08, P27/A07 to P20/A00
- Data I/O P17/D15/AD15 to P10/D08/AD08, D07/AD07 to D00/AD00
- Bus control signals RDX, P55/WRLX, P54/WRHX

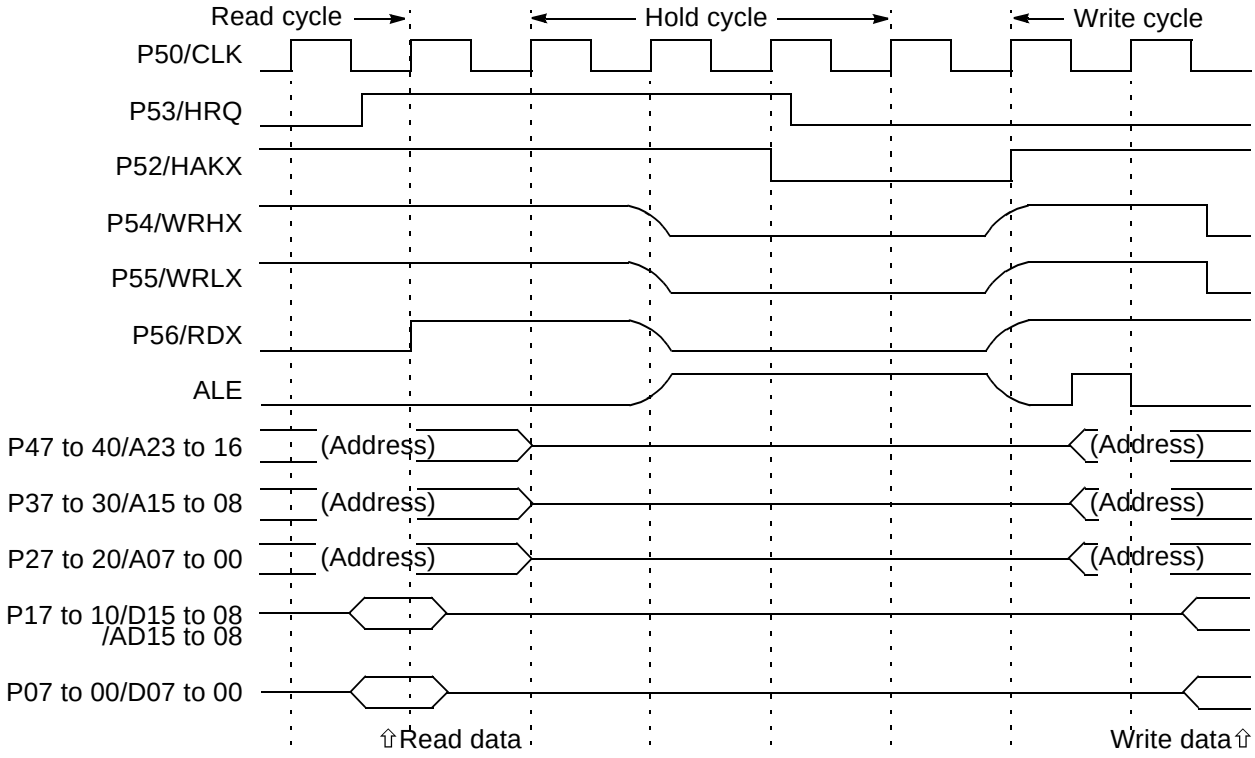
○ Multiplex mode
- Address output P47/A23 to P40/A16
- Address/data I/O P17/D15/AD15 to P10/D08/AD08, D07/AD07 to D00/AD00
- Bus control signals RDX, P55/WRLX, P54/WRHX

This enables external circuits to use the external bus.

Inputting an "L" level to the P53/HRQ pin changes the output of the P52/HAKX pin to the "H" level, restores the external pin statuses, and restarts CPU operation.

Hold request inputs are ignored in stop mode.

❑ Hold timing (external 16-bit bus mode) Non-multiplex mode



❑ Hold timing (external 16-bit bus mode) Multiplex mode

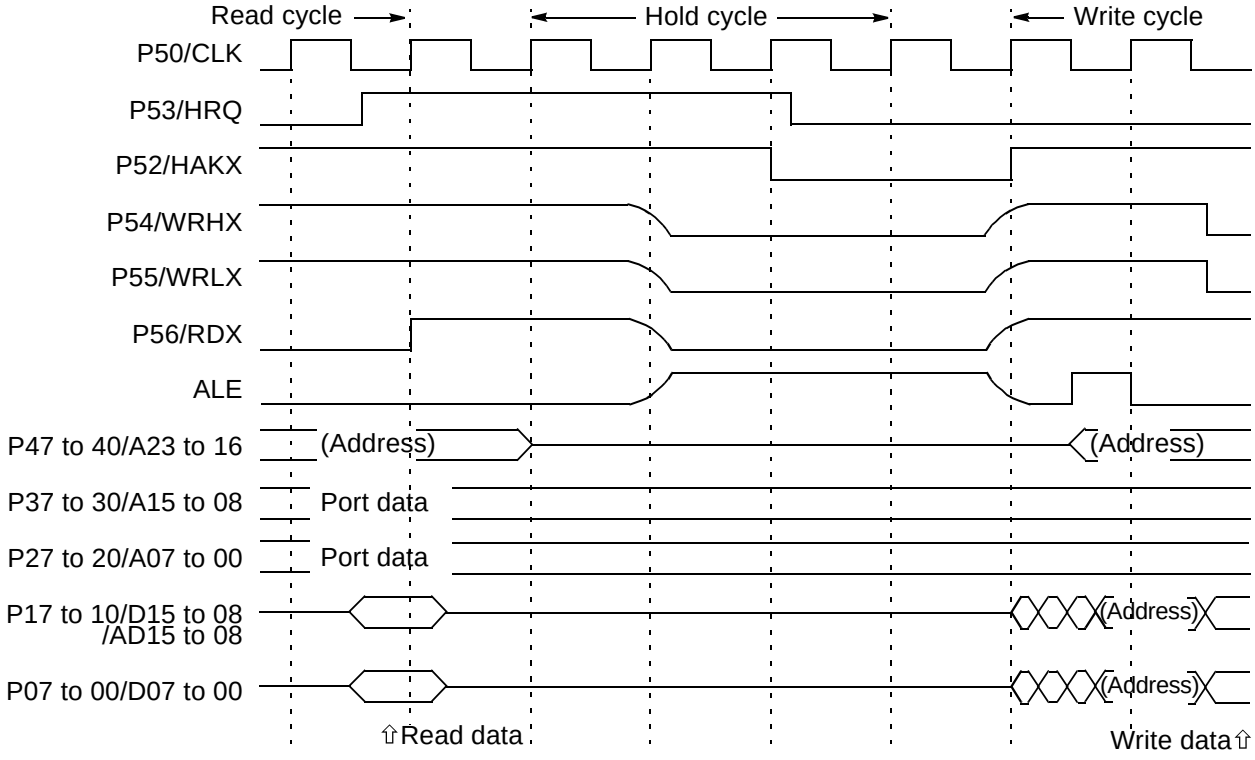


Fig. 3.4.4 Hold Timing

3.5 Low Power Modes

The following operation modes are available: PLL clock mode, PLL sleep mode, timer mode, main clock mode, main sleep mode, stop mode, and hardware standby mode. Operation modes other than PLL clock mode are classified as low power consumption modes.

In main clock mode and main sleep mode, the device operates on the main clock only (OSC oscillator clock). The PLL clock (VCO oscillator) is halted in these modes and the main clock divided by 2 is used as the operating clock. In PLL sleep mode and main sleep mode, the CPU's operating clock only is halted and other elements continue to operate. In timer mode, only the timebase timer operates. Stop mode and hardware standby mode halt the oscillator, maintaining existing data with minimum power consumption.

The CPU intermittent operation function provides an intermittent clock to the CPU when register, internal memory, internal resource, or external bus access is performed. This function reduces power consumption by lowering the CPU execution speed while still providing a high-speed clock to internal resources.

The PLL clock multiplier can be selected as 1, 2, 3, or 4 by the CS1, 0 bits.

Table 3.5.1 shows the state of each chip section in each operation mode.

Table 3.5.1 Operating States in Low Power Modes

State Transition Condition		Oscillation	Clock	CPU	Peripherals	Pins	How to Exit
Main sleep	MCS=1 SLP=1	Operating	Operating	Halted	Operating	Operating	Reset Interrupt
PLL sleep	MCS=0 SLP=1	Operating	Operating	Halted	Operating	Operating	Reset Interrupt
Timer (SPL=0)	MCS=0 STP=1	Operating	Halted	Halted	Halted	Hold current levels	Reset Interrupt
Timer (SPL=1)	MCS=0 STP=1	Operating	Halted	Halted	Halted	HI-Z	Reset Interrupt
Stop (SPL=0)	MCS=1 STP=1	Halted	Halted	Halted	Halted	Hold current levels	Reset Interrupt
Stop (SPL=1)	MCS=1 STP=1	Halted	Halted	Halted	Halted	HI-Z	Reset Interrupt
Hardware standby	HSTX=L	Halted	Halted	Halted	Halted	HI-Z	HSTX=H

Figure 3.5.1 shows the structure of the low power mode control register and clock selection register.

■ LPMCR (Low power mode control register)

Address:	7	6	5	4	3	2	1	0	⇦Bit No.
0000A0H	STP	SLP	SPL	RST	Reserved	CG1	CG0	Reserved	LPMCR
Read/write ⇨	(W)	(W)	(R/W)	(W)	(-)	(R/W)	(R/W)	(-)	
Initial value ⇨	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	

■ CKSCR (Clock selection register)

Address:	15	14	13	12	11	10	9	8	⇦Bit No.
0000A1H	Reserved	MCM	WS1	WS2	Reserved	MCS	CS1	CS0	CKSCR
Read/write ⇨	(-)	(R)	(R/W)	(R/W)	(-)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	

Fig. 3.5.1 LPMCR and CKSCR

Note: Accessing the low power mode control register:

Writing to the low power mode control register changes to the specified low power mode (stop mode or sleep mode). However, when changing to a low power mode, always use one of the instructions listed in Table 3.5.2. Using instructions other than those listed in Table 3.5.2 can cause misoperation. However, any instruction can be used on the low power mode control register when controlling functions other than changing to a low power mode.

Always write to an even-numbered address when performing a word-length write to the low power mode control register. Changing to a low power mode by writing to an odd-numbered address may cause misoperation.

Table 3.5.2 Instructions to Use When Changing to a Low Power Mode

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

The following describes the operation for each mode.

(1) Sleep mode

- Transition to sleep mode

Writing "1" to the SLP bit and "0" to the STP bit in the low power mode control register sets the standby control circuit to sleep mode. In sleep mode, only the clock supplied to the CPU stops. The CPU halts but the peripheral circuits continue to operate.

If an interrupt request is present when "1" is written to the SLP bit, the standby control circuit does not change to sleep mode. Therefore, if the CPU cannot receive the interrupt, the CPU proceeds to execute the next instruction. If the CPU can receive the interrupt, execution immediately branches to the interrupt processing routine.

Sleep mode maintains the contents of internal RAM and the contents of the accumulator and other special registers. The external bus hold function continues to operate in sleep mode. The device goes to the hold state if a hold request is received.

- Wake-up from sleep mode

The standby control circuit exits sleep mode when a reset input or interrupt occurs. When sleep mode is cleared by a reset, the device enters the reset state after waking up from sleep mode.

The standby control circuit exits sleep mode when an interrupt request with a higher priority than level 7 is generated by a peripheral circuit or other source. Normal interrupt processing starts after exiting sleep mode. The CPU executes interrupt processing if the I flag, ILM, and interrupt control register (ICR) settings allow the interrupt to be received. If the CPU cannot receive the interrupt, execution continues from the next instruction after the instruction that entered sleep mode.

Note: When executing interrupt processing, the device normally enters interrupt processing after executing the next instruction after the instruction that entered sleep mode. However, the device may enter interrupt processing before execution of the next instruction if an external bus hold request was received at the same time as the device changed to sleep mode.

(2) Watch mode

- Transition to watch mode

Writing "1" to the STP bit in the low power mode control register when the MCS bit in the clock selection register is "0" sets the standby control circuit to watch mode. In watch mode, all operation halts except for the source oscillator and timebase timer. This halts almost all chip functions.

The SPL bit in the low power mode control register controls whether I/O pins hold their existing states or change to high impedance during watch mode.

If an interrupt request is present when "1" is written to the STP bit, the standby control circuit does not change to watch mode.

Watch mode maintains the contents of internal RAM and the contents of the accumulator and other dedicated registers. The external bus hold function halts in watch mode and hold request inputs are not accepted. It is possible that the bus changes to the Hi-Z state but the HAKX signal does not change to "L" if a hold request is input during transition to watch mode.

● Releasing watch mode

The standby control circuit releases watch mode when a reset input or interrupt occurs. When watch mode is released by a reset, the device enters the reset state after releasing watch mode.

When recovering from watch mode, the standby control circuit first releases watch mode, then delays for the PLL clock oscillation to stabilize. As the MCS bit is not cleared by an external reset, the reset sequence is executed using the main clock if the duration of the reset is shorter than the PLL clock oscillation stabilization delay. As the timebase timer is not cleared, the PLL clock oscillation stabilization delay time will vary between 2^{13} and $3 * 2^{13}$ main clock cycles, depending on the state of the timebase timer.

The standby control circuit releases watch mode when an interrupt request with a higher priority than level 7 is generated by a peripheral circuit or other source. Normal interrupt processing starts after releasing watch mode. The CPU executes interrupt processing if the I flag, ILM, and interrupt control register (ICR) settings allow the interrupt to be received. If the CPU cannot receive the interrupt, execution continues from the next instruction after the instruction that entered watch mode.

Note: When executing interrupt processing, the device normally enters interrupt processing after executing the next instruction after the instruction that entered watch mode. However, the device may enter interrupt processing before execution of the next instruction if an external bus hold request was received at the same time as the device changed to watch mode.

Note: The device enters the PLL clock oscillation stabilization delay state after releasing watch mode. Therefore, if not using the PLL clock, change the MCS bit to "1" immediately after the reset or in the first instruction at the interrupt destination.

(3) Stop mode

● Transition to stop mode

Writing "1" to the STP bit in the low power mode control register when the MCS bit in the clock selection register is "1" sets the standby control circuit to stop mode. Stop mode stops the source oscillator, halting all chip functions. Therefore, this mode can maintain data with minimum power consumption.

The SPL bit in the LPMCR register controls whether I/O pins hold their existing states or change to high impedance during stop mode.

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

Stop mode maintains the contents of internal RAM and the contents of the accumulator and other special registers. The external bus hold function halts in stop mode and hold request inputs are not accepted. It is possible that the bus changes to the Hi-Z state but the HAKX signal does not change to "L" if a hold request is input during transition to stop mode.

● Wake-up from stop mode

The standby control circuit exits stop mode when a reset input or interrupt occurs. When stop mode is cleared by a reset, the device enters the reset state after waking up from stop mode.

When recovering from stop mode, the standby control circuit first delays for the oscillation to stabilize, then exits stop mode. Therefore, if stop mode is exited by a reset, the reset sequence does not execute until after the oscillation stabilization delay.

The standby control circuit clears stop mode when an interrupt request with a higher priority than level 7 is generated by a peripheral circuit or other source. After wake-up from stop mode, normal interrupt processing starts after the delay for the main clock oscillation to stabilize. The duration of the delay is specified in the WS1 and WS0 bits in the CKSCR register. The CPU executes interrupt processing if the I flag, ILM, and interrupt control register (ICR) settings allow the interrupt to be received. If the CPU cannot receive the interrupt, execution continues from the next instruction after the instruction that entered stop mode.

Note: When executing interrupt processing, the device normally enters interrupt processing after executing the next instruction after the instruction that entered stop mode. However, the device may enter interrupt processing before execution of the next instruction if an external bus hold request was received at the same time as the device changed to stop mode.

(4) Hardware standby mode

● Entering hardware standby mode

Applying an "L" level to the HSTX pin sets the standby control circuit to hardware standby mode, regardless of the current state. Hardware standby mode halts the oscillator and sets all I/O pins to high impedance. The mode continues for as long as the HSTX pin is "L" and is unaffected by other states, including resets.

Hardware standby mode maintains the contents of internal RAM, but initializes the accumulator and other special registers.

● Wake-up from hardware standby mode

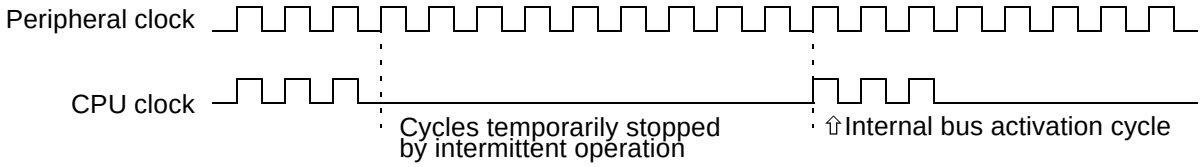
Hardware standby mode can only be cleared by the HSTX pin. When the HSTX pin changes to the "H" level, the standby control circuit clears hardware standby mode, asserts the internal reset signal, and enters the oscillation stabilization delay state. After the oscillation stabilization delay completes, the standby control circuit clears the internal reset and the CPU starts execution from the reset sequence.

(5) CPU intermittent operation function

The CPU intermittent operation function halts the clock to the CPU for a fixed duration when register, internal memory (ROM, RAM, I/O, or resource), or external bus access is performed and delays the start of the internal bus cycle. This function reduces power consumption by lowering the CPU execution speed while still providing a high-speed clock to internal resources. The CG1 and CG0 bits specify the number of clock cycles that the CPU clock is stopped.

External bus operation uses the same clock as the resources.

The instruction execution time when the CPU intermittent operation function is used is calculated by adding a correction value to the normal execution time. The correction value is determined by multiplying the number of cycles that the clock is stopped by the number of register, internal memory, internal resource, or external bus accesses.



(6) Setting the oscillation stabilization delay time

The WS1 and WS0 bits select the oscillation stabilization delay time used on release of stop mode or hardware standby mode, or when a watchdog reset occurs. Set the oscillation stabilization delay time based on the type and characteristics of the oscillator circuit and oscillator element connected to the X0 and X1 pins.

These bits are not initialized by resets other than the power-on reset. The bits are initialized to "11" by a power-on reset. Therefore, the oscillation stabilization delay after power-on is approximately 2^{18} counts of the source oscillator.

(7) Switching the machine clock

● Switching between the main clock and PLL clock

Operation can be switched between the main clock and the PLL clock by writing to the MCS bit in the CKSCR register.

If the MCS bit is changed from "1" to "0", the device switches from the main clock to the PLL clock after the oscillation stabilization delay for the PLL clock (2^{13} machine clocks).

If the MCS bit is changed from "0" to "1", the device switches from the PLL clock to the main clock at the next timing when the PLL and main clock edges match (after between 1 and 8 PLL clocks).

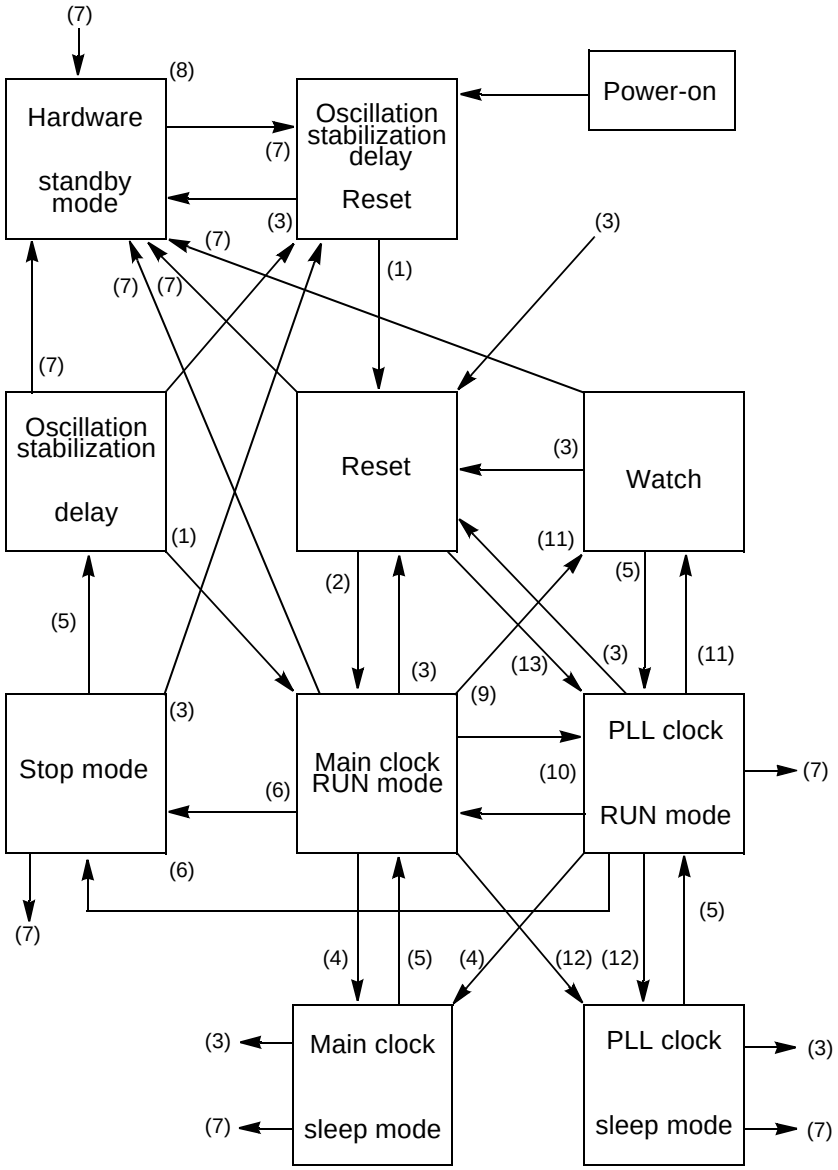
Note that the machine clock does not switch immediately after changing the MCS bit. Therefore, when using resources that depend on the machine clock, check the MCM bit to confirm that the clock switch has occurred before using the resource.

● Initializing the machine clock

The MCS bit is not initialized by external pin and RST bit resets. Other resets initialize MCS to "1".

(8) State transitions

Figure 3.5.2 shows the state transitions for the low power modes.



(1) Completion of the oscillation stabilization delay
(2) Reset released, MCS = "1"
(3) Reset input
(4) SLP = "1", MCS = "1"
(5) Interrupt input
(6) STP = "1", MCS = "1"
(7) Hardware standby input
(8) Wake-up from hardware standby
(9) MCS = "0"
(10) MCS = "1"
(11) STP = "1", MCS = "0"
(12) SLP = "1", MCS = "0"
(13) Reset released, MCS = "0"

Fig. 3.5.2 State Transitions for Low Power Modes

3.6 Pin States During Sleep, Stop, Hold, and Reset

Tables 3.6.1 to 3.6.4 list the pin states for each bus mode during stop, hold, and reset.

Table 3.6.1 Pin States in External Bus/16-Bit Data Bus Mode and Non-Multiplex Mode

<table>
<tr><th rowspan="2">Pin</th><th rowspan="2">Sleep</th><th colspan="2">Stop</th><th rowspan="2">Hold</th><th rowspan="2">Reset</th><th rowspan="2">Hardware Standby</th></tr>
<tr><th>SPL=0</th><th>SPL=1</th></tr>
<tr><td>D07 to D00
D15 to D08</td><td>Input disabled
Hi-Z output</td><td>Input disabled
Hi-Z output</td><td rowspan="12">Input disconnected
Hi-Z output
(Note 6)</td><td rowspan="2">Input disabled
Hi-Z output</td><td>Input disabled
Hi-Z output</td><td rowspan="13">Input disconnected
Hi-Z output
(Note 6)</td></tr>
<tr><td>A07 to A00
A15 to A08</td><td>Output
(Note 1)</td><td>Output
(Note 1)</td><td rowspan="2">Output
(Note 1)</td></tr>
<tr><td>A23 to A16</td><td>Output
(Note 1) (Note 4)</td><td>Output
(Note 1) (Note 4)</td><td>Input disabled
Hi-Z output
(Note 4)</td></tr>
<tr><td>P50 (CLK)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td>Input disabled
Output
(Note 1) (Note 4)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td rowspan="4">Output "H"
(Note 3)</td></tr>
<tr><td rowspan="2">P51 (RDY)</td><td rowspan="3">Hold previous state
(Note 5)</td><td rowspan="3">Hold previous state
(Note 5)</td><td>Input disabled
Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>P52 (HAKX)</td><td>Output "L"</td></tr>
<tr><td>P53 (HRQ)</td><td>Input "1"</td></tr>
<tr><td>P54 (WRHX)
P55 (WRLX)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>RDX</td><td>Output "H"</td><td>Output "H"</td><td rowspan="2">Output "H"
(Note 3)</td><td>Output "H"</td></tr>
<tr><td>ALE</td><td>Output "L"</td><td>Output "L"</td><td>Output enabled
(Note 2)</td></tr>
<tr><td>P67 to P60
P86 to P81
P95 to P90</td><td rowspan="3">Hold previous state
(Note 5)</td><td rowspan="2">Hold previous state
(Note 5)</td><td rowspan="3">Hold previous state
(Note 5)</td><td>Input disabled
Hi-Z output</td></tr>
<tr><td>PA7 to PA1
CS0</td><td>Output "H"
(Note 3)</td></tr>
<tr><td>P76 to P70
P80</td><td colspan="2">Input enabled</td><td>Input disabled
Hi-Z output</td></tr>
</table>

Note 1: The "output" state means that drive is enabled for the pin driver transistor but, as the internal circuit is not operating, the output is fixed at the "H" or "L" level.
The output may change at timings other than during a reset if the internal peripheral circuit is operating and uses the output function. The output does not change during a reset.

Note 2: "Output enabled" means that, as the drive for the pin driver transistor is enabled and operation of the internal circuit is enabled, output from the internal circuit appears at the pin.

Note 3: Operates as a pulled-up output.

Note 4: When used as an output port, the pin holds its previous state.

Note 5: "Hold previous state" means the pin continues to output the state output immediately before entering this mode. For input pins, this means that input is disabled. For output pins belonging to internal peripherals that continue to operate, continuing to output the current output state means that the pin continues to output the value set by the internal peripheral. For port or similar output pins, it means that the pin holds its current output level. "Input disabled" means that, although operation of the input gate for the pin is enabled, the pin value is not recognized internally because the internal circuit is not operating.

Note 6: "Input disconnected" means that operation of the input gate for the pin is disabled. "Hi-Z output" means that drive is enabled for the pin driver transistor and the pin goes to the high impedance state.

Table 3.6.2 Pin States in External Bus/8-Bit Data Bus Mode and Non-Multiplex Mode

<table>
<tr><th rowspan="2">Pin</th><th rowspan="2">Sleep</th><th colspan="2">Stop</th><th rowspan="2">Hold</th><th rowspan="2">Reset</th><th rowspan="2">Hardware Standby</th></tr>
<tr><th>SPL=0</th><th>SPL=1</th></tr>
<tr><td>D07 to D00</td><td>Input disabled
Hi-Z output</td><td>Input disabled
Hi-Z output</td><td rowspan="14">Input disconnected
Hi-Z output
(Note 6)</td><td>Input disabled
Hi-Z output</td><td>Input disable
Hi-Z output</td><td rowspan="14">Input disconnected
Hi-Z output
(Note 6)</td></tr>
<tr><td>A07 to A00
A15 to A08</td><td>Output
(Note 1)</td><td>Output
(Note 1)</td><td>Input disabled
Hi-Z output</td><td rowspan="2">Output
(Note 1)</td></tr>
<tr><td>A23 to A16</td><td>Output
(Note 1) (Note 4)</td><td>Output
(Note 1) (Note 4)</td><td>Input disabled
Hi-Z output
(Note 4)</td></tr>
<tr><td>P50 (CLK)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td>Input disabled
Output
(Note 1) (Note 4)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td rowspan="5">Output "H"
(Note 3)</td></tr>
<tr><td>P51 (RDY)</td><td rowspan="4">Hold previous state
(Note 5)</td><td rowspan="4">Hold previous state
(Note 5)</td><td>Input disabled
Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>P52 (HAKX)</td><td>Output "L"</td></tr>
<tr><td>P53 (HRQ)</td><td>Input "1"</td></tr>
<tr><td>P54 (WRHX)</td><td>Hold previous state
(Note 5)</td></tr>
<tr><td>P55 (WRLX)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>RDX</td><td>Output "H"</td><td>Output "H"</td><td rowspan="2">Output "H"
(Note 3)</td><td>Output "H"</td></tr>
<tr><td>ALE</td><td>Output "L"</td><td>Output "L"</td><td>Output enabled
(Note 2)</td></tr>
<tr><td>P17 to P10
P67 to P60
P86 to P81
P95 to P90</td><td rowspan="3">Hold previous state
(Note 5)</td><td rowspan="2">Hold previous state
(Note 5)</td><td rowspan="3">Hold previous state
(Note 5)</td><td>Input disabled
Hi-Z output</td></tr>
<tr><td>PA7 to PA1
CS0</td><td>Output "H"
(Note 3)</td></tr>
<tr><td>P77 to P70
P80</td><td colspan="2">Input enabled</td><td>Input disabled
Hi-Z output</td></tr>
</table>

Note 1: The "output" state means that drive is enabled for the pin driver transistor but, as the internal circuit is not operating, the output is fixed at the "H" or "L" level.
The output may change at timings other than during a reset if the internal peripheral circuit is operating and uses the output function. The output does not change during a reset.

Note 2: "Output enabled" means that, as the drive for the pin driver transistor is enabled and operation of the internal circuit is enabled, output from the internal circuit appears at the pin.

Note 3: Operates as a pulled-up output.

Note 4: When used as an output port, the pin holds its previous state.

Note 5: "Hold previous state" means the pin continues to output the state output immediately before entering this mode. For input pins, this means that input is disabled. For output pins belonging to internal peripherals that continue to operate, continuing to output the current output state means that the pin continues to output the value set by the internal peripheral. For port or similar output pins, it means that the pin holds its current output level. "Input disabled" means that, although operation of the input gate for the pin is enabled, the pin value is not recognized internally because the internal circuit is not operating.

Note 6: "Input disconnected" means that operation of the input gate for the pin is disabled. "Hi-Z output" means that drive is enabled for the pin driver transistor and the pin goes to the high impedance state.

Table 3.6.3 Pin States in External Bus/16-Bit Data Bus Mode and Multiplex Mode

<table>
<tr><th rowspan="2">Pin</th><th rowspan="2">Sleep</th><th colspan="2">Stop</th><th rowspan="2">Hold</th><th rowspan="2">Reset</th><th rowspan="2">Hardware Standby</th></tr>
<tr><th>SPL=0</th><th>SPL=1</th></tr>
<tr><td>AD07 to AD00
AD15 to AD08</td><td>Input disabled
Hi-Z output</td><td>Input disabled
Hi-Z output</td><td rowspan="11">Input disconnected
Hi-Z output
(Note 6)</td><td>Input disabled
Hi-Z output</td><td>Input disabled
Hi-Z output</td><td rowspan="12">Input disconnected
Hi-Z output
(Note 6)</td></tr>
<tr><td>A23 to A16</td><td>Output
(Note 1) (Note 4)</td><td>Output
(Note 1) (Note 4)</td><td>Input disabled
Hi-Z output
(Note 4)</td><td>Output
(Note 1)</td></tr>
<tr><td>P50 (CLK)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td>Input disabled
Output
(Note 1) (Note 4)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td rowspan="4">Output "H"
(Note 3)</td></tr>
<tr><td>P51 (RDY)</td><td rowspan="3">Hold previous state
(Note 5)</td><td rowspan="3">Hold previous state
(Note 5)</td><td>Input disabled
Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>P52 (HAKX)</td><td>Output "L"</td></tr>
<tr><td>P53 (HRQ)</td><td>Input "1"</td></tr>
<tr><td>P54 (WRHX)
P55 (WRLX)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>RDX</td><td>Output "H"</td><td>Output "H"</td><td rowspan="2">Output "H"
(Note 3)</td><td>Output "H"</td></tr>
<tr><td>ALE</td><td>Output "L"</td><td>Output "L"</td><td>Output enabled
(Note 2)</td></tr>
<tr><td>P27 to P20
P37 to P30
P67 to P60
P86 to P81
P95 to P90</td><td rowspan="3">Hold previous state
(Note 5)</td><td rowspan="2">Hold previous state
(Note 5)</td><td rowspan="3">Hold previous state
(Note 5)</td><td>Input disabled
Hi-Z output</td></tr>
<tr><td>PA7 to PA1
CS0</td><td>Output "H"
(Note 3)</td></tr>
<tr><td>P76 to P70
P80</td><td colspan="2">Input enabled</td><td>Input disabled
Hi-Z output</td></tr>
</table>

Note 1: The "output" state means that drive is enabled for the pin driver transistor but, as the internal circuit is not operating, the output is fixed at the "H" or "L" level.
The output may change at timings other than during a reset if the internal peripheral circuit is operating and uses the output function. The output does not change during a reset.

Note 2: "Output enabled" means that, as the drive for the pin driver transistor is enabled and operation of the internal circuit is enabled, output from the internal circuit appears at the pin.

Note 3: Operates as a pulled-up output.

Note 4: When used as an output port, the pin holds its previous state.

Note 5: "Hold previous state" means the pin continues to output the state output immediately before entering this mode. For input pins, this means that input is disabled. For output pins belonging to internal peripherals that continue to operate, continuing to output the current output state means that the pin continues to output the value set by the internal peripheral. For port or similar output pins, it means that the pin holds its current output level. "Input disabled" means that, although operation of the input gate for the pin is enabled, the pin value is not recognized internally because the internal circuit is not operating.

Note 6: "Input disconnected" means that operation of the input gate for the pin is disabled. "Hi-Z output" means that drive is enabled for the pin driver transistor and the pin goes to the high impedance state.

Table 3.6.4 Pin States in External Bus/8-Bit Data Bus Mode and Multiplex Mode

<table>
<tr><th rowspan="2">Pin</th><th rowspan="2">Sleep</th><th colspan="2">Stop</th><th rowspan="2">Hold</th><th rowspan="2">Reset</th><th rowspan="2">Hardware Standby</th></tr>
<tr><th>SPL=0</th><th>SPL=1</th></tr>
<tr><td>AD07 to AD00</td><td>Input disabled
Hi-Z output</td><td>Input disabled
Hi-Z output</td><td rowspan="15">Input disconnected
Hi-Z output
(Note 6)</td><td rowspan="2">Input disabled
Hi-Z output</td><td>Input disabled
Hi-Z output</td><td rowspan="15">Input disconnected
Hi-Z output
(Note 6)</td></tr>
<tr><td>AD15 to AD08</td><td>Output
(Note 1)</td><td>Output
(Note 1)</td><td rowspan="3">Output
(Note 1)</td></tr>
<tr><td>A23 to A16</td><td>Output
(Note 1) (Note 4)</td><td>Output
(Note 1) (Note 4)</td><td>Input disabled
Hi-Z output
(Note 4)</td></tr>
<tr><td>P50 (CLK)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td>Input disabled
Output
(Note 1) (Note 4)</td><td>Input disabled
Output enabled
(Note 2) (Note 4)</td><td rowspan="6">Output "H"
(Note 3)</td></tr>
<tr><td>P51 (RDY)</td><td rowspan="4">Hold previous state
(Note 5)</td><td rowspan="4">Hold previous state
(Note 5)</td><td>Input disabled
Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>P52 (HAKX)</td><td>Output "L"</td></tr>
<tr><td>P53 (HRQ)</td><td>Input "1"</td></tr>
<tr><td>P54 (WRHX)</td><td>Hold previous state
(Note 5)</td></tr>
<tr><td>P55 (WRLX)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 4)</td><td>Output "H"
(Note 3) (Note 4)</td></tr>
<tr><td>RDX</td><td>Output "H"</td><td>Output "H"</td><td rowspan="2">Output "H"
(Note 3)</td><td>Output "H"</td></tr>
<tr><td>ALE</td><td>Output "L"</td><td>Output "L"</td><td>Output enabled
(Note 2)</td></tr>
<tr><td>P27 to P20
P37 to P30
P67 to P60
P86 to P81
P95 to P90</td><td rowspan="3">Hold previous state
(Note 5)</td><td rowspan="2">Hold previous state
(Note 5)</td><td rowspan="3">Hold previous state
(Note 5)</td><td>Input disabled
Hi-Z output</td></tr>
<tr><td>PA7 to PA1
CS0</td><td>Output "H"
(Note 3)</td></tr>
<tr><td>P77 to P70
P80</td><td colspan="2">Input enabled</td><td>Input disabled
Hi-Z output</td></tr>
</table>

Note 1: The "output" state means that drive is enabled for the pin driver transistor but, as the internal circuit is not operating, the output is fixed at the "H" or "L" level.
The output may change at timings other than during a reset if the internal peripheral circuit is operating and uses the output function. The output does not change during a reset.

Note 2: "Output enabled" means that, as the drive for the pin driver transistor is enabled and operation of the internal circuit is enabled, output from the internal circuit appears at the pin.

Note 3: Operates as a pulled-up output.

Note 4: When used as an output port, the pin holds its previous state.

Note 5: "Hold previous state" means the pin continues to output the state output immediately before entering this mode. For input pins, this means that input is disabled. For output pins belonging to internal peripherals that continue to operate, continuing to output the current output state means that the pin continues to output the value set by the internal peripheral. For port or similar output pins, it means that the pin holds its current output level. "Input disabled" means that, although operation of the input gate for the pin is enabled, the pin value is not recognized internally because the internal circuit is not operating.

Note 6: "Input disconnected" means that operation of the input gate for the pin is disabled. "Hi-Z

Chapter 4: Instructions

4.1 Addressing

In the F^{2}MC-16L, the address format is determined by either the instruction's effective address specification, or by the instruction code itself (implied addressing).

4.1.1 Effective address field

The address formats specified in the effective address field are shown in Table 4.1.1.

Table 4.1.1 Table 4.1.1 Effective Address Field

Code	Notation			Address format	Default bank
00 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct 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

4.1.2 Addressing Details

(1) Immediate value (#imm)

This format specifies the operand value directly.

- #imm4
- #imm8
- #imm6
- #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

* 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 $\begin{pmatrix} @RWi + disp8 \;\; i = 0 \text{ to } 7 \\ @RWj + disp16 \;\; j = 0 \text{ to } 3 \end{pmatrix}$

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 000000H to 0000FFH 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)

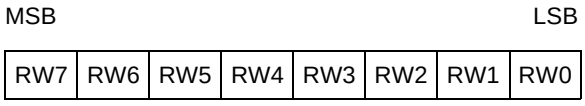
This format directly specifies a bit within a physical address from 000000H to 0000FFH. The bit position is indicated by ":bp", with larger numbers being closer to the MSB and smaller numbers being closer to the LSB.

(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.



A register is selected when the corresponding bit is "1",
and is not selected when the corresponding bit is "0".

Fig. 4.1.1 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.

4.2 Instruction Set

Table 4.2.1 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. This number is used to compensation the correction value when using the CPU clock gear function.
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.

The CPU intermittent operation function halts the clock to the CPU for a specified number of cycles when general purpose register, internal ROM, internal RAM, internal I/O, or external bus access is performed. The CG1 and CG0 bits in the low power mode control register set the number of cycles that the clock is halted.

Therefore, the number of cycles required to execute an instruction when the CPU intermittent operation function is used is calculated by adding a correction value to the normal number of execution cycles. The correction value is the number of accesses multiplied by the number of cycles that the clock is halted.

Table 4.2.2 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 0-15 ad24 16-23	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

Table 4.2.2 Explanation of Symbols in Table of Instructions (Continued)

Symbol	Explanation
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 4.2.3 Effective Address Fields

Code	Notation			Address format	Number of bytes for address extension *
00 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct The ea (effective address) corresponds to the following types (from the left, in order): byte word long word	–
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3			Register indirect	0
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+			Register indirect with post increment	0
10 11 12 13 14 15 16 17	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8 @RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8			Register indirect with 8-bit displacement	1
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16			Register indirect with 16-bit displacement	2
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16			Register indirect with base index Register indirect with base index Program counter indirect with 16-bit displacement Direct address	0 0 2 2

*: The number of bytes in the address extension are added for instructions with “+” in the “#” (number of bytes) column in the instruction set tables.

Table 4.2.4 Number of Execution Cycles for Each Form of Addressing

Code	Operand	(a)* Number of execution cycles for each addressing type	Number of accesses for each form of addressing
00 | 07	Ri RWi RLi	Listed in the instruction set table	Listed in the instruction set table
08 | 0B	@RWj	2	1
0C | 0F	@RWj+	4	2
10 | 17	@RWi+disp8	2	1
18 | 1B	@RWj+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

*: (a) corresponds to (a) in the “~” (number of cycles) column “B” (correction value) colum and in the instructions.

Table 4.2.5 Compensation Values for Number of Cycles Used to Calculate Number of Actual Cycles

Operand	(b) byte		(c)word		(d)long	
	Number of Cycles	Number of Accesses	Number of Cycles	Number of Accesses	Number of Cycles	Number of Accesses
Internal register	+0	1	+0	1	+0	2
Internal 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

Note1: (b), (c), and (d)correspond to the "~" (number of cycles) column in the instructions.

Table 4.2.6 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	–

Note1: 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.

Note2: Because instruction execution is not slowed down by all program fetches in actuality, these compensation values should be used for "worst case" calculations.

4.2.1 F²MC-16L Instruction Set (340 Instructions)

Table 4.2.7 ransfer Instructions (Byte) (41 Instructions)

Table 0.0a Move 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+	7+(a)	0	2×(b)	byte (A) ↔ (eam)	Z	–	–	–	–	–	–	–	–	–
XCH	Ri, ear	2	4	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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.8 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+dips8	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	3	2	0	long (ear) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVL	eam, A	2+	5+(a)	0	(d)	long (eam) ← (A)	–	–	–	–	–	*	*	–	–	–

Note: For an explanation of "(a)" to "(d)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.9 Addition and Subtraction Instructions (Byte/Word/Long-Word) (42 Instructions)

Table 0.0b 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)	Z	–	–	–	–	*	*	*	*	–
SUB	A, #imm8	2	2	0	0	(decimal)	Z	–	–	–	–	*	*	*	*	–
SUB	A, dir	2	5	0	(b)	byte (A) ← (A) - imm8	Z	–	–	–	–	*	*	*	*	–
SUB	A, ear	2	3	1	0	byte (A) ← (A) - (dir)	Z	–	–	–	–	*	*	*	*	–
SUB	A, eam	2+	4+(a)	0	(b)	byte (A) ← (A) - (ear)	Z	–	–	–	–	*	*	*	*	–
SUB	ear, A	2	3	2	0	byte (A) ← (A) - (eam)	–	–	–	–	–	*	*	*	*	–
SUB	eam, A	2+	5+(a)	0	2×(b)	byte (ear) ← (ear) - (A)	–	–	–	–	–	*	*	*	*	*
SUBC	A	1	2	0	0	byte (eam) ← (eam) - (A)	Z	–	–	–	–	*	*	*	*	–
SUBC	A, ear	2	3	1	0	byte (A) ← (AH) - (AL) - (c)	Z	–	–	–	–	*	*	*	*	–
SUBC	A, eam	2+	4+(a)	0	(b)	byte (A) ← (A) - (ear) - (c)	Z	–	–	–	–	*	*	*	*	–
SUBDC	A	1	3	0	0	byte (A) ← (A) - (eam) - (c) byte (A) ← (AH) - (AL) - (c) (decimal)	Z	–	–	–	–	*	*	*	*	–
ADDW	A	1	2	0	0	word (A) ← (AH) + (AL)	–	–	–	–	–	*	*	*	*	–
ADDW	A, ear	2	3	1	0	word (A) ← (A) + (ear)	–	–	–	–	–	*	*	*	*	–
ADDW	A, eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam)	–	–	–	–	–	*	*	*	*	–
ADDW	A, #imm16	3	2	0	0	word (A) ← (A) + imm16	–	–	–	–	–	*	*	*	*	–
ADDW	ear, A	2	3	2	0	word (ear) ← (ear) + (A)	–	–	–	–	–	*	*	*	*	–
ADDW	eam, A	2+	5+(a)	0	2×(c)	word (eam) ← (eam) + (A)	–	–	–	–	–	*	*	*	*	*
ADDCW	A, ear	2	3	1	0	word (A) ← (A) + (ear) + (c)	–	–	–	–	–	*	*	*	*	–
ADDCW	A, eam	2+	4+(a)	0	(c)	word (A) ← (A) + (eam) + (c)	–	–	–	–	–	*	*	*	*	–
SUBW	A	1	2	0	0	word (A) ← (AH) - (AL)	–	–	–	–	–	*	*	*	*	–
SUBW	A. ear	2	2	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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.10 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 (ear) ← (eam) + 1	–	–	–	–	–	*	*	*	–	*
DEC	ear	2	3	2	0	byte (ear) ← (ear) - 1	–	–	–	–	–	*	*	*	–	–
DEC	eam	2+	5+(a)	0	2×(b)	byte (ear) ← (eam) - 1	–	–	–	–	–	*	*	*	–	*
INCW	ear	2	3	2	0	word (ear) ← (ear) + 1	–	–	–	–	–	*	*	*	–	–
INCW	eam	2+	5+(a)	0	2×(c)	word (ear) ← (eam) + 1	–	–	–	–	–	*	*	*	–	*
DECW	ear	2	3	2	0	word(ear) ← (ear) - 1	–	–	–	–	–	*	*	*	–	–
DECW	eam	2+	5+(a)	0	2×(c)	word(ear) ← (eam) -1	–	–	–	–	–	*	*	*	–	*
INCL	ear	2	7	4	0	long (ear) ← (ear) + 1	–	–	–	–	–	*	*	*	–	–
INCL	eam	2+	9+(a)	0	2×(d)	long (ear) ← (eam) + 1	–	–	–	–	–	*	*	*	–	*
DECL	ear	2	7	4	0	long (ear) ← (ear) - 1	–	–	–	–	–	*	*	*	–	–
DECL	eam	2+	9+(a)	0	2×(d)	long (ear) ← (eam) - 1	–	–	–	–	–	*	*	*	–	*

Table 4.2.11 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, #im16	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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.12 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) byte (AL) ← quotient byte (AH) ← remainder	–	–	–	–	–	–	–	*	*	–
DIVU	A ,ear	2	*2	1	0	word (A) ÷ byte (ear) byte (A) ← quotient byte (ear) ← remainder	–	–	–	–	–	–	–	*	*	–
DIVU	A ,eam	2+	*3	0	*6	word (A) ÷ byte (eam) byte (A) ← quotient byte (eam) ← remainder	–	–	–	–	–	–	–	*	*	–
DIVUW	A, ear	2	*4	1	0	long ÷ word (ear) word (A) ← quotient word (ear) ← remainder	–	–	–	–	–	–	–	*	*	–
DIVUW	A, eam	2+	*5	0	*7	long ÷ word (eam) word (A) ← quotient word (eam) ← remainder	–	–	–	–	–	–	–	*	*	–
							–	–	–	–	–	–	–	–	–	–
MUL	A	1	*8	0	0	word (A) ← byte (AH) x byte (AL)	–	–	–	–	–	–	–	–	–	–
MUL	A, ear	2	*9	1	0	word (A) ← byte (A) x byte (ear)	–	–	–	–	–	–	–	–	–	–
MUL	A, eam	2+	*10	0	(b)	word (A) ← byte (A) x byte (eam)										
MULUW	A	1	*11	0	0	long (A) ← word (AH) x word (AL)	–	–	–	–	–	–	–	–	–	–
MULUW	A, ear	2	*12	1	0	long (A) ← word (A) x word (ear)	–	–	–	–	–	–	–	–	–	–
MULUW	A, eam	2+	*13	0	(c)	long (A) ← word (A) x word (earm)	–	–	–	–	–	–	–	–	–	–

*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 x (b) normally.

*7: (c) when dividing into zero or when an overflow occurs, and 2 x (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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.13 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	2	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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.14 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 4.2.15 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	2	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 4.2.16 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		0	long (A) ←Shift until first digit is "1" byte (R0) ← Current shift count	–	–	–	–	–	–	*	–	–	–

*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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.17 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 rotation with carry	–	–	–	–	–	*	*	–	*	–
RORC	A	2	2	0	0	byte (A) ← Left rotation with carry	–	–	–	–	–	*	*	–	*	–
RORC	ear	2	3	2	0	byte (ear) ←Right rotation with carry	–	–	–	–	–	*	*	–	–	–
RORC	eam	2+	5+(a)	0	2×(b)	byte (eam) ← Right rotation with carry	–	–	–	–	–	*	*	–	*	*
ROLC	ear	2	3	2	0	byte (ear) ← Left rotation with carry	–	–	–	–	–	*	*	–	–	–
ROLC	eam	2+	5+(a)	0	2×(b)	byte (eam) ←Left rotation with carry	–	–	–	–	–	*	*	–	*	*
ASR	A, R0	2	*1	1	0	byte (A) ← Arithmetic right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSR	A, R0	2	*1	1	0	byte (A) ← Logical right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSL	A, R0	2	*1	1	0	byte (A) ← Logical left barrel shift (A, R0)	–	–	–	–	–	*	*	–	*	–
ASRW	A	1	2	0	0	word (A) ← Arithmetic right shift (A, 1-bit)	–	–	–	–	*	R	*	–	*	–0
LSRW	A / SHRW	1	2	0	0	word (A) ←Logical right shift (A, 1-bit)	–	–	–	–	–	*	*	–	*	–
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	–	–	–	–	*	*	*	–	*	–
LSRL	A, R0	2	*2	1	0	barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSLL	A, R0	2	*2	1	0	long (A) ←Logical right barrel shift (A, R0) 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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.18 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 if (Z) = 1	–	–	–	–	–	–	–	-	–	–
BNZ / BNE	rel		2	*1	0	0	Branch if (Z) = 0	–	–	–	–	–	–	–	–	–	–
BC / BLO	rel		2	*1	0	0	Branch if (c) = 1	–	–	–	–	–	–	–	–	–	–
BNC / BHS	rel		2	*1	0	0	Branch if (c) = 0	–	–	–	–	–	–	–	–	–	–
BN	rel		2	*1	0	0	Branch if (N) = 1	–	–	–	–	–	–	–	–	–	–
BP	rel		2	*1	0	0	Branch if (N) = 0	–	–	–	–	–	–	–	–	–	–
BV	rel		2	*1	0	0	Branch if (V) = 1	–	–	–	–	–	–	–	–	–	–
BNV	rel		2	*1	0	0	Branch if (V) = 0	–	–	–	–	–	–	–	–	–	–
BT	rel		2	*1	0	0	Branch if (T) = 1	–	–	–	–	–	–	–	–	–	–
BNT	rel		2	*1	0	0	Branch if (T) = 0	–	–	–	–	–	–	–	–	–	–
BLT	rel		2	*1	0	0	Branch if (V) xor (N) = 1	–	–	–	–	–	–	–	–	–	–
BGE	rel		2	*1	0	0	Branch if (V) xor (N) = 0	–	–	–	–	–	–	–	–	–	–
BLE	rel		2	*1	0	0	Branch if ((V) xor (N)) or (Z) = 1	–	–	–	–	–	–	–	–	–	–
BGT	rel		2	*1	0	0	Branch if ((V) xor (N)) or (Z) = 0	–	–	–	–	–	–	–	–	–	–
BLS	rel		2	*1	0	0	Branch if (c) or (Z) = 1	–	–	–	–	–	–	–	–	–	–
BHI	rel		2	*1	0	0	Branch if (c) or (Z) = 0	–	–	–	–	–	–	–	–	–	–
BRA	rel		2	*1	0	0	Branch unconditionally	–	–	–	–	–	–	–	–	–	–
JMP	@A		1	2	0	0	word (PC) ← (A)	–	–	–	–	–	–	–	–	–	–
JMP	addr16		3	3	0	0	word (PC) ← addr16	–	–	–	–	–	–	–	–	–	–
JMP	@ear		2	2	1	0	word (PC) ← (ear)	–	–	–	–	–	–	–	–	–	–
JMP	@eam		2+	4+(a)	0	(c)	word (PC) ← (eam)	–	–	–	–	–	–	–	–	–	–
JMPP	@ear	Note1	2	3	2	0	word (PC) ← (ear), (PCB) ← (ear+2)	–	–	–	–	–	–	–	–	–	–
JMPP	@eam	Note1	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	Note3	2	5	1	(c)	word (PC) ← (ear)	–	–	–	–	–	–	–	–	–	–
CALL	@eam	Note3	2+	7+(a)	0	2×(c)	word (PC) ← (eam)	–	–	–	–	–	–	–	–	–	–
CALL	addr16	Note4	3	6	0	(c)	word (PC) ← addr16	–	–	–	–	–	–	–	–	–	–
CALLV	#vct4	Note4	1	7	0	2×(c)	Vector call instruction	–	–	–	–	–	–	–	–	–	–
CALLP	@ear	Note5	2	8	2	2×(c)	word (PC) ← (ear) 0-15, (PCB) ← (ear)16-23	–	–	–	–	–	–	–	–	–	–
CALLP	@eam	Note5	2+	11+(a)	0	*2	word (PC) ← (ear) 0-15, (PCB) ← (ear)16-23	–	–	–	–	–	–	–	–	–	–
CALLP	addr24	Note6	4	10	0	2×(c)	word (PC) ← addr0-15, (PCB) ← addr16-23	–	–	–	–	–	–	–	–	–	–

*1: 4 when branching, 3 when not branching.

*2: 3 × (c) + (b)

Note1: Read (word) branch address.

Note2: W: Save (word) into stack; R: read (word) branch address.

Note3: Save (word) into stack.

Note4: W: Save (long-word) into W stack; R: read (long-word) R branch address.

Note5: Save (long-word) into stack.

Note: For an explanation of "(a)" to "(d)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.19 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 if byte (A) ≠ imm8	–	–	–	–	*	*	*	*	*	–
CWBNE	A, #imm16, rel	4	*1	0	0	Branch if word (A) ≠ imm16	–	–	–	–	*	*	*	*	*	–
CBNE	ear, #imm8, rel	4	*2	1	0	Branch if byte (ear) ≠ imm8	–	–	–	–	*	*	*	*	*	–
CBNE	eam, #imm8, rel	4+	*3	0	(b)	Branch if byte (eam) ≠ imm8	–	–	–	–	*	*	*	*	*	–
CWBNE	ear, #imm16, rel	5	*4	1	0	Branch if word (ear) ≠ imm16	–	–	–	–	*	*	*	*	*	–
CWBNE	eam, #imm16, rel	5+	*3	0	(c)	Branch if word (eam) ≠ imm16	–	–	–	–	*	*	*	*	*	–
DBNZ	ear, rel	3	*5	2	0	Branch if byte (ear) = (ear)-1, (ear) ≠ 0	–	–	–	–	*	*	*	*	–	–
DBNZ	eam, rel	3+	*6	2	2×(b)	Branch if byte (eam) = (eam)-1, (eam) ≠ 0	–	–	–	–	*	*	*	*	–	*
DWBNZ	ear, rel	3	*5	0	0	Branch if word (ear) = (ear)-1, (ear) ≠ 0	–	–	–	–	*	*	*	*	–	–
DWBNZ	eam, rel	3+	*6	0	2 × (c)	Branch if 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	15	0	6×(c)	Return from interrupt processing	–	–	*	*	*	*	*	*	*	–
LINK	#imm8	2	6	0	(c)	At function entry, saves the old frame pointer to the stack, sets the new frame pointer, and reserves the local pointer area.	–	–	–	–	–	–	–	–	–	–
UNLINK		1	5	0	(c)	At function exit, restores the old frame pointer from the stack.	–	–	–	–	–	–	–	–	–	–
RET	Note1	1	4	0	(c)	Return from subroutine	–	–	–	–	–	–	–	–	–	–
RETP	Note2	1	6	0	(d)	Return from 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

Note1: Return from stack (word)

Note2: Return from stack (long)

Note3: Do not use the RWj+ addressing mode for the CBNE or CWBNE instructions.

Note: For an explanation of "(a)" to "(d)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.20 Other Control Instructions (Byte/Word/Long-Word) (28 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW	A	1	4	0	(c)	word (SP) ← (SP) - 2, ((SP)) ← (A)	–	–	–	–	–	–	–	–	–	–
PUSHW	AH	1	4	0	(c)	word (SP) ← (SP) - 2, ((SP)) ← (AH)	–	–	–	–	–	–	–	–	–	–
PUSHW	PS	1	4	0	(c)	word (SP) ← (SP) - 2, ((SP)) ← (PS)	–	–	–	–	–	–	–	–	–	–
PUSHW	rlst	2	*3	+&	*4	(SP) ← (SP) - 2n, ((SP)) ← (rlst)	–	–	–	–	–	–	–	–	–	–
POPW	A	1	3	0	(c)	word (A) ← ((SP)) , (SP) ← (SP) + 2	–	*	–	–	–	–	–	–	–	–
POPW	AH	1	3	0	(c)	word (AH) ← ((SP)) , (SP) ← (SP) + 2	–	–	–	–	–	–	–	–	–	–
POPW	PS	1	4	0	(c)	word (PS) ← ((SP)) , (SP) ←(SP) + 2	–	–	*	*	*	*	*	*	*	–
POPW	rlst	2	*2	+&	*4	(rlst) ← ((SP)) , (SP) ← (SP)	–	–	–	–	–	–	–	–	–	–
JCTX	@A	1	14	0	6×(c)	Context switch (process switch) instruction	–	–	*	*	*	*	*	*	*	–
AND	CCR, #imm8	2	3	0	0	byte (CCR) ← (CCR) and imm8	–	–	*	*	*	*	*	*	*	–
OR	CCR, #imm8	2	3	0	0	byte (CCR) ← (CCR) or imm8	–	–	*	*	*	*	*	*	*	–
MOV	RP, #imm8	2	2	0	0	byte (RP) ← imm8	–	–	–	–	–	–	–	–	–	–
MOV	ILM, #imm8	2	2	0	0	byte (ILM) ← imm8	–	–	–	–	–	–	–	–	–	–
MOVEA	RWi, ear	2	3	1	0	word (RWi) ← ear	–	–	–	–	–	–	–	–	–	–
MOVEA	RWi, eam	2+	2+(a)	1	0	word (RWi) ← eam	–	–	–	–	–	–	–	–	–	–
MOVEA	A, ear	2	1	0	0	word (A) ← ear	–	*	–	–	–	–	–	–	–	–
MOVEA	A, eam	2+	1+(a)	0	0	word (A) ← eam	–	*	–	–	–	–	–	–	–	–
ADDSP	#imm8	2	3	0	0	word (SP) ← (SP)+ext(imm8)	–	–	–	–	–	–	–	–	–	–
ADDSP	#imm16	3	3	0	0	word (SP) ← (SP)+imm16	–	–	–	–	–	–	–	–	–	–
MOV	A, brg1	2	*1	0	0	byte (A) ← (brg1)	Z	*	–	–	–	*	*	–	–	–
MOV	brg2, A	2	1	0	0	byte (brg2) ← (A)	–	–	–	–	–	*	*	–	–	–
NOP		1	1	0	0	No operation	–	–	–	–	–	–	–	–	–	–
ADB		1	1	0	0	Prefix code for auxiliary AD access	–	–	–	–	–	–	–	–	–	–
DTB		1	1	0	0	Prefix code for data DT access	–	–	–	–	–	–	–	–	–	–
PCB		1	1	0	0	Prefix code for program PC access	–	–	–	–	–	–	–	–	–	–
SPB		1	1	0	0	Prefix code for stack SP access	–	–	–	–	–	–	–	–	–	–
NCC		1	1	0	0	Prefix code for no flag change	–	–	–	–	–	–	–	–	–	–
CMR		1	1	0	0	Prefix for common register bank	–	–	–	–	–	–	–	–	–	–

*1: PCB, ADB, SSB, USB, and SPB: .....1 cycle
DTB, DPR: .....2 cycles

2: 7 + 3(number of POP operations) + 2*(register number of last register POPed) or 7 when RLST = 0 (no move register)

3: 29 + 3(number of PUSH operations) - 3*(register number of last register PUSHed) or 8 when RLST = 0 (no move register)

*4: (number of POP operations) * (c) or (number of PUSH operations) * (c)

Note: For an explanation of "(a)" to "(d)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.21 Bit Manipulation Instructions (21 Instructions)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB	A ,dir: bp	3	5	0	(b)	byte (A) ← (dir:bp)b	Z	*	–	–	–	*	*	–	–	–
MOVB	A ,addr16: bp	4	5	0	(b)	byte (A) ← (addr16:bp)b	Z	*	–	–	–	*	*	–	–	–
MOVB	A, io: bp	3	4	0	(b)	byte (A) ← (io:bp)b	Z	*	–	–	–	*	*	–	–	–
MOVB	dir: bp, A	3	7	0	2×(b)	bit (dir:bp)b ← (A)	–	–	–	–	–	*	*	–	–	*
MOVB	addr16: bp, A	4	7	0	2×(b)	bit (addr16:bp)b ← (A)	–	–	–	–	–	*	*	–	–	*
MOVB	io: bp, A	3	6	0	2×(b)	bit (io:bp)b ← (A)	–	–	–	–	–	*	*	–	–	*
SETB	dir: bp	3	7	0	2×(b)	bit (dir:bp)b ← 1	–	–	–	–	–	–	–	–	–	*
SETB	addr16: bp	4	7	0	2×(b)	bit (addr16:bp)b ← 1	–	–	–	–	–	–	–	–	–	*
SETB	io: bp	3	7	0	2×(b)	bit (io:bp)b ← 1	–	–	–	–	–	–	–	–	–	*
CLRB	dir: bp	3	7	0	2×(b)	bit (dir:bp)b ← 0	–	–	–	–	–	–	–	–	–	*
CLRB	addr16: bp	4	7	0	2×(b)	bit (addr16:bp)b ← 0	–	–	–	–	–	–	–	–	–	*
CLRB	io: bp	3	7	0	2×(b)	bit (io:bp)b ← 0	–	–	–	–	–	–	–	–	–	*
BBC	dir: bp, rel	4	*1	0	(b)	Branch if (dir:bp)b = 0	–	–	–	–	–	–	–	*	–	–
BBC	addr16: bp, rel	5	*1	0	(b)	Branch if (addr16:bp)b = 0	–	–	–	–	–	–	–	*	–	–
BBC	io: bp, rel	4	*2	0	(b)	Branch if (io:bp)b = 0	–	–	–	–	–	–	–	*	–	–
BBS	dir:bp, rel	4	*1	0	(b)	Branch if (dir:bp)b = 1	–	–	–	–	–	–	–	*	–	–
BBS	addr16:bp, rel	5	*1	0	(b)	Branch if (addr16:bp)b = 1	–	–	–	–	–	–	–	*	–	–
BBS	io:bp, rel	4	*2	0	(b)	Branch if (io:bp)b = 1	–	–	–	–	–	–	–	*	–	–
SBBS	addr16:bp, rel	5	*3	0	2×(b)	Branch if (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)", see Table 4.2.3 and Table 4.2.4.

Table 4.2.22 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	–	–	–	–	–	–	–	–	–	–
SWAP / XCHW A,T	1	2	0	0	word (AH) ↔ (AL)	–	*	–	–	–	–	–	–	–	–
EXT	1	1	0	0	Byte sign extension	X	–	–	–	*	*	–	–	–	–
EXTW	1	2	0	0	Word sign extension	–	X	–	–	*	*	–	–	–	–
ZEXT	1	1	0	0	Byte zero extension	Z	–	–	–	R	*	–	–	–	–
ZEXTW	1	1	0	0	Word zero extention	–	Z	–	–	R	*	–	–	–	–

Table 4.2.23 String Instructions (10 Instructions)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	+&	*3	Byte move @AH+ ← @AL+, counter = RW0	–	–	–	–	–	–	–	–	–	–
MOVSD	2	*2	+&	*3	Byte move @AH- ← @AL-, counter = RW0	–	–	–	–	–	–	–	–	–	–
SCEQ / SCEQ1	2	*1	+&	*4	Byte search (@AH+) ~ AL, counter = RW0	–	–	–	–	–	*	*	*	*	–
SCEQD	2	*1	+&	*4	Byte search (@AH-) ~ AL, counter = RW0	–	–	–	–	–	*	*	*	*	–
FILS / FILS1	2	6m+6	+&	*3	Byte fill @AH+ ← AL, counter = RW0	–	–	–	–	–	*	*	–	–	–
MOVSW / MOVSWI	2	*2	+)	*6	Word move @AH+ ← @AL+, counter = RW0	–	–	–	–	–	–	–	–	–	–
MOVSWD	2	*2	+)	*6	Word move @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 the source and destination access different areas, calculate the (b) value separately for each area.

*4: (b) × n

*5: 2 × (RW0)

*6: (c) × (RW0) + (c) × (RW0) When the source and destination access different areas, calculate the (c) value separately for each area.

*7: (c) × n

*8: 2 × (RW0)

m: RW0 value (counter value)

n: Loop count

Note: For an explanation of "(a)" to "(d)", see Table 4.2.3 and Table 4.2.4.

4.3 Instruction Map

Because the F^{2}MC-16L 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.

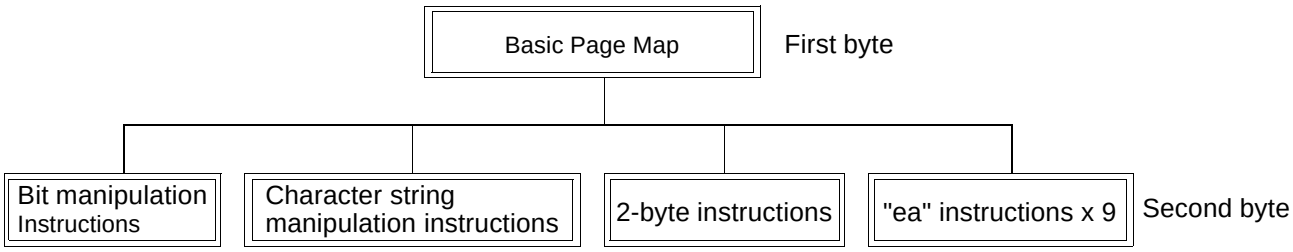
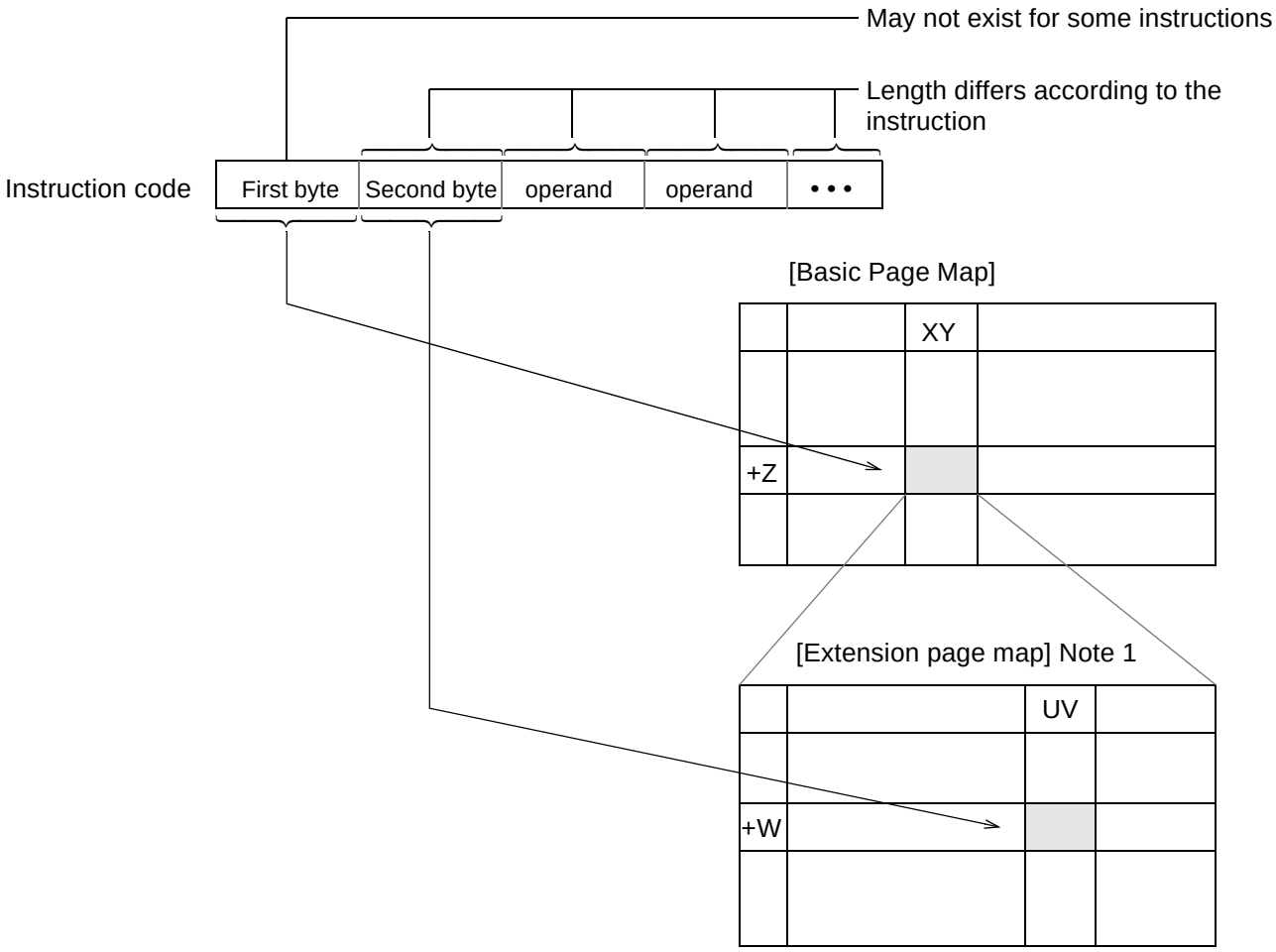


Fig. 4.3.1 Structure of F^{2}MC-16L Instruction Map

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.



Note1: 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. 4.3.2 Correspondence between Actual Instructions and the Instruction Maps

Table 4.3.1 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	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								BNZ/BNE rel
+2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A, #8	MOV A, addr16	JMP addr16	ea instructions								BC /BLO rel
+3	NEG A	JCTX @A	CMP A	CMP A, #8	MOVX A, #8	MOV addr 16, A	JMPP addr24	ea instructions								BNC/BHS rel
+4	PCB	EXT	AND CCR, #8	AND A, #8	MOV dir, #8	MOV io, #8	CALL addr16	ea instructions								BN rel
+5	DTB	ZEXT	OR CCR, #8	OR A, #8	MOVX A, dir	MOVX A, io	CALLP addr24	ea instructions								BP rel
+6	ADB	SWAP	DIVU A	XOR A, #8	MOVW A, SP	MOVW io, #16	RETP	ea instructions								BV rel
+7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP, A	MOVX A, addr16	RET	ea instructions								BNV rel
+8	LINK imm#8	ADDL A, #32	ADDW A	ADDW A, #16	MOVW A, dir	MOVW A, io	INT #vct8	ea instructions	MOVW A, RWi	MOVW RWi, A	MOVW RWi, #16	MOVW A, @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 4.3.2 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 ad16:bp, rel				SBBS addr16:bp
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Table 4.3.3 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 4.3.4 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											
+9																
+A				MOVW @RL1+d8, A	MOVW A, @RL1+d8											
+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 4.3.5 "ea" Instructions) (First byte = 70H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWBNE	CWBNE									CBNE	CBNE
+0	ADDL A, RL0	ADDL A, @RW0+d8	SUBL A, RL0	SUBL A, @RW0+d8	RW0, #16, rel	@RW0+d8, #16, rel	CMPL A, RL0	CMPL A, @RW0+d8	ANDL A, RL0	ANDL A, @RW0+d8	ORL A, RL0	ORL A, @RW0+d8	XORL A, RL0	XORL A, @RW0+d8	R0, #8, rel	@RW0+d8, #8, rel
+1	ADDL A, RL0	ADDL A, @RW1+d8	SUBL A, RL0	SUBL A, @RW1+d8	RW1, #16, rel	@RW1+d8, #16, rel	CMPL A, RL0	CMPL A, @RW1+d8	ANDL A, RL0	ANDL A, @RW1+d8	ORL A, RL0	ORL A, @RW1+d8	XORL A, RL0	XORL A, @RW1+d8	R1, #8, rel	@RW1+d8, #8, rel
+2	ADDL A, RL1	ADDL A, @RW2+d8	SUBL A, RL1	SUBL A, @RW2+d8	RW2, #16, rel	@RW2+d8, #16, rel	CMPL A, RL1	CMPL A, @RW2+d8	ANDL A, RL1	ANDL A, @RW2+d8	ORL A, RL1	ORL A, @RW2+d8	XORL A, RL1	XORL A, @RW2+d8	R2, #8, rel	@RW2+d8, #8, rel
+3	ADDL A, RL1	ADDL A, @RW3+d8	SUBL A, RL1	SUBL A, @RW3+d8	RW3, #16, rel	@RW3+d8, #16, rel	CMPL A, RL1	CMPL A, @RW3+d8	ANDL A, RL1	ANDL A, @RW3+d8	ORL A, RL1	ORL A, @RW3+d8	XORL A, RL1	XORL A, @RW3+d8	R3, #8, rel	@RW3+d8, #8, rel
+4	ADDL A, RL2	ADDL A, @RW4+d8	SUBL A, RL2	SUBL A, @RW4+d8	RW4, #16, rel	@RW4+d8, #16, rel	CMPL A, RL2	CMPL A, @RW4+d8	ANDL A, RL2	ANDL A, @RW4+d8	ORL A, RL2	ORL A, @RW4+d8	XORL A, RL2	XORL A, @RW4+d8	R4, #8, rel	@RW4+d8, #8, rel
+5	ADDL A, RL2	ADDL A, @RW5+d8	SUBL A, RL2	SUBL A, @RW5+d8	RW5, #16, rel	@RW5+d8, #16, rel	CMPL A, RL2	CMPL A, @RW5+d8	ANDL A, RL2	ANDL A, @RW5+d8	ORL A, RL2	ORL A, @RW5+d8	XORL A, RL2	XORL A, @RW5+d8	R5, #8, rel	@RW5+d8, #8, rel
+6	ADDL A, RL3	ADDL A, @RW6+d8	SUBL A,RL3	SUBL A, @RW6+d8	RW6, #16, rel	@RW6+d8, #16, rel	CMPL A, RL3	CMPL A, @RW6+d8	ANDL A, RL3	ANDL A, @RW6+d8	ORL A, RL3	ORL A, @RW6+d8	XORL A, RL3	XORL A, @RW6+d8	R6, #8, rel	@RW6+d8, #8, rel
+7	ADDL A, RL3	ADDL A, @RW7+d8	SUBL A, RL3	SUBL A, @RW7+d8	RW7, #16, rel	@RW7+d8, #16, rel	CMPL A, RL3	CMPL A, @RW7+d8	ANDL A, RL3	ANDL A, @RW7+d8	ORL A, RL3	ORL A, @RW7+d8	XORL A, RL3	XORL A, @RW7+d8	R7, #8, rel	@RW7+d8, #8, rel
+8	ADDL A, @RW0	ADDL A, @RW0+d16	SUBL A, @RW0	SUBL A, @RW0+d16	@ RW0, #16, rel	@RW0+d16, #16, rel	CMPL A, @RW0	CMPL A, @RW0+d16	ANDL A, @RW0	ANDL A, @RW0+d16	ORL A, @RW0	ORL A, @RW0+d16	XORL A, @RW0	XORL A, @RW0+d16	@RW0, #8, rel	@RW0+d16, #8, rel
+9	ADDL A, @RW1	ADDL A, @RW1+d16	SUBL A, @RW1	SUBL A, @RW1+d16	@ RW1, #16, rel	@RW1, d16, #16, rel	CMPL A, @RW1	CMPL A, @RW1+d16	ANDL A, @RW1	ANDL A, @RW1+d16	ORL A, @RW1	ORL A, @RW1+d16	XORL A, @RW1	XORL A, @RW1+d16	@RW1, #8, rel	@RW1+d16, #8, rel
+A	ADDL A, @RW2	ADDL A, @RW2+d16	SUBL A, @RW2	SUBL A, @RW2+d16	@ RW2, #16, rel	@RW2+d16, #16, rel	CMPL A, @RW2	CMPL A, @RW2+d16	ANDL A, @RW2	ANDL A, @RW2+d16	ORL A, @RW2	ORL A, @RW2+d16	XORL A, @RW2	XORL A, @RW2+d16	@RW2, #8, rel	@RW2+d16, #8, rel
+B	ADDL A, @RW3	ADDL A, @RW3+d16	SUBL A, @RW3	SUBL A, @RW3+d16	@ RW3, #16, rel	@RW3+d16, #16, rel	CMPL A, @RW3	CMPL A, @RW3+d16	ANDL A, @RW3	ANDL A, @RW3+d16	ORL A, @RW3	ORL A, @RW3+d16	XORL A, @RW3	XORL A, @RW3+d16	@RW3, #8, rel	@RW3+d16, #8, rel
+C	ADDL A, @RW0+	ADDL A, @RW0+RW7		SUBL A, @RW0+RW7	@ RW0+, #16, rel	@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		@RW0+RW7, #8, rel
+D	ADDL A, @RW1+	ADDL A, @RW1+RW7	Use prohibited	SUBL A, @RW1+RW7	@ RW1+, #16, rel	@RW1+RW7, #16, rel	CMPL A, @RW1+	CMPL A, @RW1+RW7	ANDL A, @RW1+	ANDL A, @RW1+RW7	ORL A, @RW1+	ORL A, @RW1+RW7	XORL A, @RW1+	XORL A, @RW1+RW7	Use prohibited	@RW1+RW7, #8, rel
+E	ADDL A, @RW2+	ADDL A, @PC+d16		SUBL A, @PC+d16	@ RW2+, #16, rel	@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		@PC+d16, #8, rel
+F	ADDL A, @RW3+	ADDL A, addr16		SUBL A, addr16	@ RW3+, #16, rel	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		addr16, #8, rel

Table 4.3.6 "ea" Instructions ((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 @R W0+d8, A	MOV R0, #8	MOV @R W0+d8, #8	MOVEA A, RW0	MOVEA A, @RW0+d8
+1	JMPP @RL0	JMPP @@RW1+d8	CALLP @RL0	CALLP @@RW1+d8	INCL RL0	INCL @RW1+d8	DECL RL0	DECL @RW1+d8	MOVL A, RL0	MOVL A, @RW1+d8	MOVL RL0, A	MOVL @R W1+d8, A	MOV R1, #8	MOV @R W1+d8, #8	MOVEA A, RW1	MOVEA A, @RW1+d8
+2	JMPP @RL1	JMPP @@RW2+d8	CALLP @RL1	CALLP @@RW2+d8	INCL RL1	INCL @RW2+d8	DECL RL1	DECL @RW2+d8	MOVL A, RL1	MOVL A, @RW2+d8	MOVL RL1, A	MOVL @R W2+d8, A	MOV R2, #8	MOV @R W2+d8, #8	MOVEA A, RW2	MOVEA A, @RW2+d8
+3	JMPP @RL1	JMPP @@RW3+d8	CALLP @RL1	CALLP @@RW3+d8	INCL RL1	INCL @RW3+d8	DECL RL1	DECL @RW3+d8	MOVL A, RL1	MOVL A, @RW3+d8	MOVL RL1, A	MOVL @R W3+d8, A	MOV R3, #8	MOV @R W3+d8, #8	MOVEA A, RW3	MOVEA A, @RW3+d8
+4	JMPP @RL2	JMPP @@RW4+d8	CALLP @RL2	CALLP @@RW4+d8	INCL RL2	INCL @RW4+d8	DECL RL2	DECL @RW4+d8	MOVL A, RL2	MOVL A, @RW4+d8	MOVL RL2, A	MOVL @R W4+d8, A	MOV R4, #8	MOV @R W4+d8, #8	MOVEA A, RW4	MOVEA A, @RW4+d8
+5	JMPP @RL2	JMPP @@RW5+d8	CALLP @RL2	CALLP @@RW5+d8	INCL RL2	INCL @RW5+d8	DECL RL2	DECL @RW5+d8	MOVL A, RL2	MOVL A, @RW5+d8	MOVL RL2, A	MOVL @R W5+d8, A	MOV R5, #8	MOV @R W5+d8, #8	MOVEA A, RW5	MOVEA A, @RW5+d8
+6	JMPP @RL3	JMPP @@RW6+d8	CALLP @RL3	CALLP @@RW6+d8	INCL RL3	INCL @RW6+d8	DECL RL3	DECL @RW6+d8	MOVL A, RL3	MOVL A, @RW6+d8	MOVL RL3, A	MOVL @R W6+d8, A	MOV R6, #8	MOV @R W6+d8, #8	MOVEA A, RW6	MOVEA A, @RW6+d8
+7	JMPP @RL3	JMPP @@RW7+d8	CALLP @RL3	CALLP @@RW7+d8	INCL RL3	INCL @RW7+d8	DECL RL3	DECL @RW7+d8	MOVL A, RL3	MOVL A, @RW7+d8	MOVL RL3, A	MOVL @R W7+d8, A	MOV R7, #8	MOV @R W7+d8, #8	MOVEA A, RW7	MOVEA A, @RW7+d8
+8	JMPP @@RW0	JMPP @RW0+d16	CALLP @RW0	CALLP @RW0+d16	INCL @RW0	INCL @RW0+d16	DECL @RW0	DECL @RW0+d16	MOVL A, @RW0	MOVL A, @RW0+d16	MOVL @RW0, A	MOVL @R W0+d16, A	MOV @RW0, #8	MOV @R W0+d16, #8	MOVEA A, @RW0	MOVEA A, @RW0+d16
+9	JMPP @@RW1	JMPP @RW1+d16	CALLP @@RW1	CALLP @RW1+d16	INCL @RW1	INCL @RW1+d16	DECL @RW1	DECL @RW1+d16	MOVL A, @RW1	MOVL A, @RW1+d16	MOVL @RW1, A	MOVL @R W1+d16, A	MOV @RW1, #8	MOV @R W1+d16, #8	MOVEA A, @RW1	MOVEA A, @RW1+d16
+A	JMPP @@RW2	JMPP @RW2+d16	CALLP @@RW2	CALLP @RW2+d16	INCL @RW2	INCL @RW2+d16	DECL @RW2	DECL @RW2+d16	MOVL A, @RW2	MOVL A, @RW2+d16	MOVL @RW2, A	MOVL @R W2+d16, A	MOV @RW2, #8	MOV @R W2+d16, #8	MOVEA A, @RW2	MOVEA A, @RW2+d16
+B	JMPP @@RW3	JMPP @RW3+d16	CALLP @@RW3	CALLP @RW3+d16	INCL @RW3	INCL @RW3+d16	DECL @RW3	DECL @RW3+d16	MOVL A,@ RW3	MOVL A, @RW3+d16	MOVL @RW3, A	MOVL @R W3+d16, A	MOV @RW3, #8	MOV @R W3+d16, #8	MOVEA A, @RW3	MOVEA A, @RW3+d16
+C	JMPP @@RW0+	JMPP @RW0+RW7	CALLP @@RW0+	CALLP @RW0+RW7	INCL @RW0+	INCL @RW0+RW7	DECL @RW0+	DECL @RW0+RW7	MOVL A, @RW0+	MOVL A, @RW0+RW7	MOVL @RW0+, A	MOVL @R W0+RW7, A	MOV @RW0+, #8	MOV @R W0+RW7, #8	MOVEA A, @RW0+	MOVEA A, @RW0+RW7
+D	JMPP @@RW1+	JMPP @RW1+RW7	CALLP @@RW1+	CALLP @RW1+RW7	INCL @RW1+	INCL @RW1+RW7	DECL @RW1+	DECL @RW1+RW7	MOVL A, @RW1+	MOVL A, @RW1+RW7	MOVL @RW1+, A	MOVL @R W1+RW7, A	MOV @RW1+, #8	MOV @R W1+RW7, #8	MOVEA A, @RW1+	MOVEA A, @RW1+RW7
+E	JMPP @@RW2+	JMPP @@PC+d16	CALLP @@RW2+	CALLP @@PC+d16	INCL @RW2+	INCL @PC+d16	DECL @RW2+	DECL @PC+d16	MOVL A, @RW2+	MOVL A, @PC+d16	MOVL @RW2+, A	MOVL @P C+d16, A	MOV @RW2+, #8	MOV @P C+d16, #8	MOVEA A, @RW2+	MOVEA A, @PC+d16
+F	JMPP @@RW3+	JMPP @addr16	CALLP @@RW3+	CALLP @addr16	INCL @RW3+	INCL addr16	DECL @RW3+	DECL addr16	MOVL A, @RW3+	MOVL A, addr16	MOVL @RW3+, A	MOVL addr16, A	MOV @RW3+, #8	MOV addr16, #8	MOVEA A, @RW3+	MOVEA A, addr16

Table 4.3.7 "ea" Instructions) (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 @RW0+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 @RW1+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 @RW2+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 @RW3+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 @RW4+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 @RW5+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 @RW6+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 @RW7+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 @RW0+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 @RW1+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 @RW2+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 @RW3+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 @RW0+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 @RW1+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 @PC+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 4.3.8 "ea" Instructions ((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 @RC+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+	DEC 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 4.3.9 "ea" Instructions) (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 4.3.10 "ea" Instructions < (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 @RW0+RW7, A	SUB @RW0+, A	SUB @RW0+RW7, A	SUBC A, @RW0+	SUBC A, @RW0+RW7	NEG @RW0+	NEG @RW0+RW7	AND @RW0+, A	AND @RW0+RW7, A	OR @RW0+, A	OR @RW0+RW7, A	XOR @RW0+, A	XOR @RW0+RW7, A	NOT @RW0+	NOT @RW0+RW7
+D	ADD @RW1+, A	ADD @RW1+RW7, A	SUB @RW1+, A	SUB @RW1+RW7, A	SUBC A, @RW1+	SUBC A, @RW1+RW7	NEG @RW1+	NEG @RW1+RW7	AND @RW1+, A	AND @RW1+RW7, A	OR @RW1+, A	OR @RW1+RW7, A	XOR @RW1+, A	XOR @RW1+RW7, A	NOT @RW1+	NOT @RW1+RW7
+E	ADD @RW2+, A	ADD C+d16, A	SUB @RW2+, A	SUB C+d16, A	SUBC A, @RW2+	SUBC A, @PC+d16	NEG @RW2+	NEG @PC+d16	AND @RW2+, A	AND C+d16, A	OR @RW2+, A	OR @PC+d16, A	XOR @RW2+, A	XOR C+d16, A	NOT @RW2+	NOT @PC+d16
+F	ADD @RW3+, A	ADD addr16, A	SUB @RW3+, A	SUB addr16, A	SUBC A, @RW3+	SUBC A, addr16	NEG @RW3+	NEG addr16	AND @RW3+, A	AND addr16, A	OR @RW3+, A	OR addr16, A	XOR @RW3+, A	XOR addr16, A	NOT @RW3+	NOT addr16

Table 4.3.11 "ea" Instructions › (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 @RW0+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 @RW1+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 @RW2+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 @RW3+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 @RW0+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 @RW1+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 4.3.12 "ea" Instructions (1) (First byte = 77H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW RW0, A	ADDW @RW0+d8, A	SUBW RW0, A	SUBW @RW0+d8, A	SUBCW A, RW0	SUBCW A, @RW0+d8	NEGW RW0	NEGW @RW0+d8	ANDW RW0, A	ANDW @RW0+d8, A	ORW RW0, A	ORW @RW0+d8, A	XORW RW0, A	XORW @RW0+d8, A	NOTW RW0	NOTW @RW0+d8
+1	ADDW RW1, A	ADDW @RW1+d8, A	SUBW RW1, A	SUBW @RW1+d8, A	SUBCW A, RW1	SUBCW A, @RW1+d8	NEGW RW1	NEGW @RW1+d8	ANDW RW1, A	ANDW @RW1+d8, A	ORW RW1, A	ORW @RW1+d8, A	XORW RW1, A	XORW @RW1+d8, A	NOTW RW1	NOTW @RW1+d8
+2	ADDW RW2, A	ADDW @RW2+d8, A	SUBW RW2, A	SUBW @RW2+d8, A	SUBCW A, RW2	SUBCW A, @RW2+d8	NEGW RW2	NEGW @RW2+d8	ANDW RW2, A	ANDW @RW2+d8, A	OWR RW2, A	ORW @RW2+d8, A	XORW RW2, A	XORW @RW2+d8, A	NOTW RW2	NOTW @RW2+d8
+3	ADDW RW3, A	ADDW @RW3+d8, A	SUBW RW3, A	SUBW @RW3+d8, A	SUBCW A, RW3	SUBCW A, @RW3+d8	NEGW RW3	NEGW @RW3+d8	ANDW RW3, A	ANDW @RW3+d8, A	ORW RW3, A	ORW @RW3+d8, A	XORW RW3, A	XORW @RW3+d8, A	NOTW RW3	NOTW @RW3+d8
+4	ADDW RW4, A	ADDW @RW4+d8, A	SUBW RW4, A	SUBW @RW4+d8, A	SUBCW A, RW4	SUBCW A, @RW4+d8	NEGW RW4	NEGW @RW4+d8	ANDW RW4, A	ANDW @RW4+d8, A	ORW RW4, A	ORW @RW4+d8, A	XORW RW4, A	XORW @RW4+d8, A	NOTW RW4	NOTW @RW4+d8
+5	ADDW RW5, A	ADDW @RW5+d8, A	SUBW RW5, A	SUBW @RW5+d8, A	SUBCW A, RW5	SUBCW A, @RW5+d8	NEGW RW5	NEGW @RW5+d8	ANDW RW5, A	ANDW @RW5+d8, A	ORW RW5, A	ORW @RW5+d8, A	XORW RW5, A	XORW @RW5+d8, A	NOTW RW5	NOTW @RW5+d8
+6	ADDW RW6, A	ADDW @RW6+d8, A	SUBW RW6, A	SUBW @RW6+d8, A	SUBCW A, RW6	SUBCW A, @RW6+d8	NEGW RW6	NEGW @RW6+d8	ANDW RW6, A	ANDW @RW6+d8, A	ORW RW6, A	ORW @RW6+d8, A	XORW RW6, A	XORW @RW6+d8, A	NOTW RW6	NOTW @RW6+d8
+7	ADDW RW7, A	ADDW @RW7+d8, A	SUBW RW7, A	SUBW @RW7+d8, A	SUBCW A,RW7	SUBCW A, @RW7+d8	NEGW RW7	NEGW @RW7+d8	ANDW RW7, A	ANDW @RW7+d8, A	ORW RW7, A	ORW @RW7+d8, A	XORW RW7, A	XORW @RW7+d8, A	NOTW RW7	NOTW @RW7+d8
+8	ADDW @RW0, A	ADDW @RW0+d16, A	SUBW @RW0, A	SUBW @RW0+d16, A	SUBCW A, @RW0	SUBCW A, @RW0+d16	NEGW @RW0	NEGW @RW0+d16	ANDW @RW0, A	ANDW @RW0+d16, A	ORW @RW0, A	ORW @RW0+d16, A	XORW @RW0, A	XORW @RW0+d16, A	NOTW @RW0	NOTW @RW0+d16
+9	ADDW @RW1, A	ADDW @RW1+d16, A	SUBW @RW1, A	SUBW @RW1+d16, A	SUBCW A, @RW1	SUBCW A, @RW1+d16	NEGW @RW1	NEGW @RW1+d16	ANDW @RW1, A	ANDW @RW1+d16, A	ORW @RW1, A	ORW @RW1+d16, A	XORW @RW1, A	XORW @RW1+d16, A	NOTW @RW1	NOTW @RW1+d16
+A	ADDW @RW2, A	ADDW @RW2+d16, A	SUBW @RW2, A	SUBW @RW2+d16, A	SUBCW A, @RW2	SUBCW A, @RW2+d16	NEGW @RW2	NEGW @RW2+d16	ANDW @RW2, A	ANDW @RW2+d16, A	ORW @RW2, A	ORW @RW2+d16, A	XORW @RW2, A	XORW @RW2+d16, A	NOTW @RW2	NOTW @RW2+d16
+B	ADDW @RW3, A	ADDW @RW3+d16, A	SUBW @RW3, A	SUBW @RW3+d16, A	SUBCW A, @RW3	SUBCW A, @RW3+d16	NEGW @RW3	NEGW @RW3+d16	ANDW @RW3, A	ANDW @RW3+d16, A	ORW @RW3, A	ORW @RW3+d16, A	XORW @RW3, A	XORW @RW3+d16, A	NOTW @RW3	NOTW @RW3+d16
+C	ADDW @RW0+, A	ADDW @RW0+RW7	SUBW @RW0+, A	SUBW @RW0+RW7, A	SUBCW A, @RW0+	SUBCW A, @RW0+RW7	NEGW @RW0+	NEGW @RW0+RW7	ANDW @RW0+, A	ANDW @RW0+RW7, A	ORW @RW0+, A	ORW @RW0+RW7, A	XORW @RW0+, A	XORW @RW0+RW7, A	NOTW @RW0+	NOTW @RW0+RW7
+D	ADDW @RW1+, A	ADDW @RW1+RW7	SUBW @RW1+, A	SUBW @RW1+RW7, A	SUBCW A, @RW1+	SUBCW A, @RW1+RW7	NEGW @RW1+	NEGW @RW1+RW7	ANDW @RW1+, A	ANDW @RW1+RW7, A	ORW @RW1+, A	ORW @RW1+RW7, A	XORW @RW1+, A	XORW @RW1+RW7, A	NOTW @RW1+	NOTW @RW1+RW7
+E	ADDW @RW2+, A	ADDW @RPC+d16, A	SUBW @RW2+, A	SUBW @PC+d16, A	SUBCW A, @RW2+	SUBCW A, @PC+d16	NEGW @RW2+	NEGW @PC+d16	ANDW @RW2+, A	ANDW @PC+d16, A	ORW @RW2+, A	ORW @PC+d16, A	XORW @RW2+, A	XORW @PC+d16, A	NOTW @RW2+	NOTW @PC+d16
+F	ADDW @RW3+, A	ADDW @Raddr16, 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 4.3.13 "ea" Instructions ❱ (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					DIVU A, R0	DIVU A, @RW0+d8	DIVUW A, RW0	DIVUW A, @RW0+d8				
+1	MULU A, R1	MULU A, @RW1+d8	MULUW A, RW1	MULUW A, @RW1+d8					DIVU A, R1	DIVU A, @RW1+d8	DIVUW A, RW1	DIVUW A, @RW1+d8				
+2	MULU A, R2	MULU A, @RW2+d8	MULUW A, RW2	MULUW A, @RW2+d8					DIVU A, R2	DIVU A, @RW2+d8	DIVUW A, RW2	DIVUW A, @RW2+d8				
+3	MULU A, R3	MULU A, @RW3+d8	MULUW A, RW3	MULUW A, @RW3+d8					DIVU A, R3	DIVU A, @RW3+d8	DIVUW A, RW3	DIVUW A, @RW3+d8				
+4	MULU A, R4	MULU A, @RW4+d8	MULUW A, RW4	MULUW A, @RW4+d8					DIVU A, R4	DIVU A, @RW4+d8	DIVUW A, RW4	DIVUW A, @RW4+d8				
+5	MULU A, R5	MULU A, @RW5+d8	MULUW A, RW5	MULUW A, @RW5+d8					DIVU A, R5	DIVU A, @RW5+d8	DIVUW A, RW5	DIVUW A, @RW5+d8				
+6	MULU A, R6	MULU A, @RW6+d8	MULUW A, RW6	MULUW A, @RW6+d8					DIVU A, R6	DIVU A, @RW6+d8	DIVUW A, RW6	DIVUW A, @RW6+d8				
+7	MULU A, R7	MULU A, @RW7+d8	MULUW A, RW7	MULUW A, @RW7+d8					DIVU A, R7	DIVU A, @RW7+d8	DIVUW A, RW7	DIVUW A, @RW7+d8				
+8	MULU A, @RW0	MULU A, @RW0+d16	MULUW A, @RW0	MULUW A, @RW0+d16					DIVU A, @RW0	DIVU A, @RW0+d16	DIVUW A, @RW0	DIVUW A, @RW0+d16				
+9	MULU A, @RW1	MULU A, @RW1+d16	MULUW A, @RW1	MULUW A, @RW1+d16					DIVU A, @RW1	DIVU A, @RW1+d16	DIVUW A, @RW1	DIVUW A, @RW1+d16				
+A	MULU A, @RW2	MULU A, @RW2+d16	MULUW A, @RW2	MULUW A, @RW2+d16					DIVU A, @RW2	DIVU A, @RW2+d16	DIVUW A, @RW2	DIVUW A, @RW2+d16				
+B	MULU A, @RW3	MULU A, @RW3+d16	MULUW A, @RW3	MULUW A, @RW3+d16					DIVU A, @RW3	DIVU A, @RW3+d16	DIVUW A, @RW3	DIVUW A, @RW3+d16				
+C	MULU A, @RW0+	MULU A, @RW0+RW7	MULUW A, @RW0+	MULUW A, @RW0+RW7					DIVU A, @RW0+	DIVU A, @RW0+RW7	DIVUW A, @RW0+	DIVUW A, @RW0+RW7				
+D	MULU A, @RW1+	MULU A, @RW1+RW7	MULUW A, @RW1+	MULUW A, @RW1+RW7					DIVU A, @RW1+	DIVU A, @RW1+RW7	DIVUW A, @RW1+	DIVUW A, @RW1+RW7				
+E	MULU A, @RW2+	MULU A, @PC+d16	MULUW A, @RW2+	MULUW A, @PC+d16					DIVU A, @RW2+	DIVU A, @PC+d16	DIVUW A, @RW2+	DIVUW A, @ PC+d16				
+F	MULU A, @RW3+	MULU A, addr16	MULUW A, @RW3+	MULUW A, addr16					DIVU A, @RW3+	DIVU A, addr16	DIVUW A, @RW3+	DIVUW A, addr16				

Table 4.3.14 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 4.3.15 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 @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 4.3.16 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 RW0, @RW0+	MOVW RW0, @RW0+RW7	MOVW RW1, @RW0+	MOVW RW1, @RW0+RW7	MOVW RW2, @RW0+	MOVW RW2, @RW0+RW7	MOVW RW3, @RW0+	MOVW RW3, @RW0+RW7	MOVW RW4, @RW0+	MOVW RW4, @RW0+RW7	MOVW RW5, @RW0+	MOVW RW5, @RW0+RW7	MOVW RW6, @RW0+	MOVW RW6, @RW0+RW7	MOVW RW7, @RW0+	MOVW RW7, @RW0+RW7
+D	MOVW RW0, @RW1+	MOVW RW0, @RW1+RW7	MOVW RW1, @RW1+	MOVW RW1, @RW1+RW7	MOVW RW2, @RW1+	MOVW RW2, @RW1+RW7	MOVW RW3, @RW1+	MOVW RW3, @RW1+RW7	MOVW RW4, @RW1+	MOVW RW4, @RW1+RW7	MOVW RW5, @RW1+	MOVW RW5, @RW1+RW7	MOVW RW6, @RW1+	MOVW RW6, @RW1+RW7	MOVW RW7, @RW1+	MOVW RW7, @RW1+RW7
+E	MOVW RW0, @RW2+	MOVW RW0, @PC+d16	MOVW RW1, @RW2+	MOVW RW1, @PC+d16	MOVW RW2, @RW2+	MOVW RW2, @PC+d16	MOVW RW3, @RW2+	MOVW RW3, @PC+d16	MOVW RW4, @RW2+	MOVW RW4, @PC+d16	MOVW RW5, @RW2+	MOVW RW5, @PC+d16	MOVW RW6, @RW2+	MOVW RW6, @PC+d16	MOVW RW7, @RW2+	MOVW RW7, @PC+d16
+F	MOVW RW0, @RW3+	MOVW RW0, addr16	MOVW RW1, @RW3+	MOVW RW1, addr16	MOVW RW2, @RW3+	MOVW RW2, addr16	MOVW RW3, @RW3+	MOVW RW3, addr16	MOVW RW4, @RW3+	MOVW RW4, addr16	MOVW RW5, @RW3+	MOVW RW5, addr16	MOVW RW6, @RW3+	MOVW RW6, addr16	MOVW RW7, @RW3+	MOVW RW7, addr16

Table 4.3.17 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 0+d16, R0	MOV @RW0, R1	MOV @RW 0+d16, R1	MOV @RW0, R2	MOV @RW 0+d16, R2	MOV @RW0, R3	MOV @RW 0+d16, R3	MOV @RW0, R4	MOV @RW 0+d16, R4	MOV @RW0, R5	MOV @RW 0+d16, R5	MOV @RW0, R6	MOV @RW 0+d16, R6	MOV @RW0, R7	MOV @RW 0+d16, R7
+9	MOV @RW1, R0	MOV @RW 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 2+d16, R0	MOV @RW2, R1	MOV @RW 2+d16, R1	MOV @RW2, R2	MOV @RW 2+d16, R2	MOV @RW2, R3	MOV @RW 2+d16, R3	MOV @RW2, R4	MOV @RW 2+d16, R4	MOV @RW2, R5	MOV @RW 2+d16, R5	MOV @RW2, R6	MOV @RW 2+d16, R6	MOV @RW2, R7	MOV @RW 2+d16, R7
+B	MOV @RW3, R0	MOV @RW 3+d16, R0	MOV @RW3, R1	MOV @RW 3+d16, R1	MOV @RW3, R2	MOV @RW 3+d16, R2	MOV @RW3, R3	MOV @RW 3+d16, R3	MOV @RW3, R4	MOV @RW 3+d16, R4	MOV @RW3, R5	MOV @RW 3+d16, R5	MOV @RW3, R6	MOV @RW 3+d16, R6	MOV @RW3, R7	MOV @RW 3+d16, R7
+C	MOV @RW0+, R0	MOV @RW 0+RW7, R0	MOV @RW0+, R1	MOV @RW 0+RW7, R1	MOV @RW0+, R2	MOV @RW 0+RW7, R2	MOV @RW0+, R3	MOV @RW 0+RW7, R3	MOV @RW0+, R4	MOV @RW 0+RW7, R4	MOV @RW0+, R5	MOV @RW 0+RW7, R5	MOV @RW0+, R6	MOV @RW 0+RW7, R6	MOV @RW0+, R7	MOV @RW 0+RW7, R7
+D	MOV @RW1+, R0	MOV @RW 1+RW7, R0	MOV @RW1+, R1	MOV @RW 1+RW7, R1	MOV @RW1+, R2	MOV @RW 1+RW7, R2	MOV @RW1+, R3	MOV @RW 1+RW7, R3	MOV @RW1+, R4	MOV @RW 1+RW7, R4	MOV @RW1+, R5	MOV @RW 1+RW7, R5	MOV @RW1+, R6	MOV @RW 1+RW7,R6	MOV @RW1+, R7	MOV @RW 1+RW7, R7
+E	MOV @RW2+, R0	MOV 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 4.3.18 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 @RW 0+d8, RW0	MOVW RW0, RW1	MOVW @RW 0+d8, RW1	MOVW RW0, RW2	MOVW @RW 0+d8, RW2	MOVW RW0, RW3	MOVW @RW 0+d8, RW3	MOVW RW0, RW4	MOVW @RW 0+d8, RW4	MOVW RW0, RW5	MOVW @RW 0+d8, RW5	MOVW RW0, RW6	MOVW @RW 0+d8, RW6	MOVW RW0, RW7	MOVW @RW 0+d8, RW7
+1	MOVW RW1, RW0	MOVW @RW 1+d8, RW0	MOVW RW1, RW1	MOVW @RW 1+d8, RW1	MOVW RW1, RW2	MOVW @R 1+d8, RW2	MOVW RW1, RW3	MOVW @RW 1+d8, RW3	MOVW RW1, RW4	MOVW @RW 1+d8, RW4	MOVW RW1, RW5	MOVW @RW 1+d8, RW5	MOVW RW1, RW6	MOVW @RW 1+d8, RW6	MOVW RW1, RW7	MOVW @RW 1+d8, RW7
+2	MOVW RW2, RW0	MOVW @RW 2+d8, RW0	MOVW RW2, RW1	MOVW @RW 2+d8, RW1	MOVW RW2, RW2	MOVW @RW 2+d8, RW2	MOVW RW2, RW3	MOVW @RW 2+d8, RW3	MOVW RW2, RW4	MOVW @RW 2+d8, RW4	MOVW RW2, RW5	MOVW @RW 2+d8, RW5	MOVW RW2, RW6	MOVW @RW 2+d8, RW6	MOVW RW2, RW7	MOVW @RW 2+d8, RW7
+3	MOVW RW3, RW0	MOVW @RW 3+d8, RW0	MOVW RW3, RW1	MOVW @RW 3+d8, RW1	MOVW RW3, RW2	MOVW @RW 3+d8, RW2	MOVW RW3, RW3	MOVW @RW 3+d8, RW3	MOVW RW3, RW4	MOVW @RW 3+d8, RW4	MOVW RW3, RW5	MOVW @RW 3+d8, RW5	MOVW RW3, RW6	MOVW @RW 3+d8, RW6	MOVW RW3, RW7	MOVW @RW 3+d8, RW7
+4	MOVW RW4, RW0	MOVW @RW 4+d8, RW0	MOVW RW4, RW1	MOVW @RW 4+d8, RW1	MOVW RW4, RW2	MOVW @RW 4+d8, RW2	MOVW RW4, RW3	MOVW @RW 4+d8, RW3	MOVW RW4, RW4	MOVW @RW 4+d8, RW4	MOVW RW4, RW5	MOVW @RW 4+d8, RW5	MOVW RW4, RW6	MOVW @RW 4+d8, RW6	MOVW RW4, RW7	MOVW @RW 4+d8, RW7
+5	MOVW RW5, RW0	MOVW @RW 5+d8, RW0	MOVW RW5, RW1	MOVW @RW 5+d8, RW1	MOVW RW5, RW2	MOVW @RW 5+d8, RW2	MOVW RW5, RW3	MOVW @RW 5+d8, RW3	MOVW RW5, RW4	MOVW @RW 5+d8, RW4	MOVW RW5, RW5	MOVW @RW 5+d8, RW5	MOVW RW5, RW6	MOVW @RW 5+d8, RW6	MOVW RW5, RW7	MOVW @RW 5+d8, RW7
+6	MOVW RW6, RW0	MOVW @RW 6+d8, RW0	MOVW RW6, RW1	MOVW @RW 6+d8, RW1	MOVW RW6, RW2	MOVW @RW 6+d8, RW2	MOVW RW6, RW3	MOVW @RW 6+d8, RW3	MOVW RW6, RW4	MOVW @RW 6+d8, RW4	MOVW RW6, RW5	MOVW @RW 6+d8, RW5	MOVW RW6, RW6	MOVW @RW 6+d8, RW6	MOVW RW6, RW7	MOVW @RW 6+d8, RW7
+7	MOVW RW7, RW0	MOVW @RW 7+d8, RW0	MOVW RW7, RW1	MOVW @RW 7+d8, RW1	MOVW RW7, RW2	MOVW @RW 7+d8, RW2	MOVW RW7, RW3	MOVW @RW 7+d8, RW3	MOVW RW7, RW4	MOVW @RW 7+d8, RW4	MOVW RW7, RW5	MOVW @RW 7+d8, RW5	MOVW RW7, RW6	MOVW @RW 7+d8, RW6	MOVW RW7, RW7	MOVW @RW 7+d8, RW7
+8	MOVW @RW0, RW0	MOVW@RW0 +d16, RW0	MOVW @RW0, RW1	MOVW@RW0 +d16, RW1	MOVW @RW0, RW2	MOVW@RW0 +d16, RW2	MOVW @RW0, RW3	MOVW@RW0 +d16, RW3	MOVW @RW0, RW4	MOVW@RW0 +d16, RW4	MOVW @RW0, RW5	MOVW@RW0 +d16, RW5	MOVW @RW0, RW6	MOVW@RW0 +d16, RW6	MOVW @RW0, RW7	MOVW@RW0 +d16, RW7
+9	MOVW @RW1, RW0	MOVW@RW1 +d16, RW0	MOVW @RW1, RW1	MOVW@RW1 +d16, RW1	MOVW @RW1, RW2	MOVW@RW1 +d16, RW2	MOVW @RW1, RW3	MOVW@RW1 +d16, RW3	MOVW @RW1, RW4	MOVW@RW1 +d16, RW4	MOVW @RW1, RW5	MOVW@RW1 +d16, RW5	MOVW @RW1, RW6	MOVW@RW1 +d16, RW6	MOVW @RW1, RW7	MOVW@RW1 +d16, RW7
+A	MOVW @RW2, RW0	MOVW@RW2 +d16, RW0	MOVW @RW2, RW1	MOVW@RW2 +d16, RW1	MOVW @RW2, RW2	MOVW@RW2 +d16, RW2	MOVW @RW2, RW3	MOVW@RW2 +d16, RW3	MOVW @RW2, RW4	MOVW@RW2 +d16, RW4	MOVW @RW2, RW5	MOVW@RW2 +d16, RW5	MOVW @RW2, RW6	MOVW@RW2 +d16, RW6	MOVW @RW2, RW7	MOVW@RW2 +d16, RW7
+B	MOVW @RW3, RW0	MOVW@RW3 +d16, RW0	MOVW @RW3, RW1	MOVW@RW3 +d16, RW1	MOVW @RW3, RW2	MOVW@RW3 +d16, RW2	MOVW @RW3, RW3	MOVW@RW3 +d16, RW3	MOVW @RW3, RW4	MOVW@RW3 +d16, RW4	MOVW @RW3, RW5	MOVW@RW3 +d16, RW5	MOVW @RW3, RW6	MOVW@RW3 +d16, RW6	MOVW @RW3, RW7	MOVW@RW3 +d16, RW7
+C	MOVW @RW0+, RW0	MOVW@RW0 +RW7, RW0	MOVW @ RW0+, RW1	MOVW@RW0 +RW7, RW1	MOVW @ RW0+, RW2	MOVW@RW0 +RW7, RW2	MOVW @ RW0+, RW3	MOVW@RW0 +RW7, RW3	MOVW @ RW0+, RW4	MOVW@RW0 +RW7, RW4	MOVW @ RW0+, RW5	MOVW@RW0 +RW7, RW5	MOVW @ RW0+, RW6	MOVW@RW0 +RW7, RW6	MOVW @ RW0+, RW7	MOVW@RW0 +RW7, RW7
+D	MOVW @RW1+, RW0	MOVW@RW1 +RW7, RW0	MOVW @ RW1+, RW1	MOVW@RW1 +RW7, RW1	MOVW @ RW1+, RW2	MOVW@RW1 +RW7, RW2	MOVW @ RW1+, RW3	MOVW@RW1 +RW7, RW3	MOVW @ RW1+, RW4	MOVW@RW1 +RW7, RW4	MOVW @ RW1+, RW5	MOVW@RW1 +RW7, RW5	MOVW @ RW1+, RW6	MOVW@RW1 +RW7, RW6	MOVW @ RW1+, RW7	MOVW@RW1 +RW7, RW7
+E	MOVW @RW2+, RW0	MOVW @PC+ d16, RW0	MOVW @ RW2+, RW1	MOVW @PC+ d16, RW1	MOVW @ RW2+, RW2	MOVW @PC+ d16, RW2	MOVW @ RW2+, RW3	MOVW @PC+ d16, RW3	MOVW @ RW2+, RW4	MOVW @PC+ d16, RW4	MOVW @ RW2+, RW5	MOVW @PC+ d16, RW5	MOVW @ RW2+, RW6	MOVW @PC+ d16, RW6	MOVW @ RW2+, RW7	MOVW @PC+ d16, RW7
+F	MOVW @RW3+, RW0	MOVW addr 16, RW0	MOVW @ RW3+, RW1	MOVW addr 16, RW1	MOVW @ RW3+, RW2	MOVW addr 16, RW2	MOVW @ RW3+, RW3	MOVW addr 16, RW3	MOVW @ RW3+, RW4	MOVW addr 16, RW4	MOVW @ RW3+, RW5	MOVW addr 16, RW5	MOVW @ RW3+, RW6	MOVW addr 16, RW6	MOVW @ RW3+, RW7	MOVW addr 16, RW7

Table 4.3.19 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, A	XCH R1, @RW2	XCH R1, @RW2+d16, A	XCH R2, @RW2	XCH R2, @RW2+d16, A	XCH R3, @RW2	XCH R3, @RW2+d16, A	XCH R4, @RW2	XCH R4, @RW2+d16, A	XCH R5, @RW2	XCH R5, @RW2+d16, A	XCH R6, @RW2	XCH R6, @RW2+d16, A	XCH R7, @RW2	XCH R7, @RW2+d16, A
+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 4.3.20 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+d8	XCHW RW1, RW0	XCHW RW1, @RW0+d8	XCHW RW2, RW0	XCHW RW2, @RW0+d8	XCHW RW3, RW0	XCHW RW3, @RW0+d8	XCHW RW4, RW0	XCHW RW4, @RW0+d8	XCHW RW5, RW0	XCHW RW5, @RW0+d8	XCHW RW6, RW0	XCHW RW6, @RW0+d8	XCHW RW7, RW0	XCHW RW7, @RW0+d8
+1	XCHW RW0, RW1	XCHW RW0, @RW1+d8	XCHW RW1, RW1	XCHW RW1, @RW1+d8	XCHW RW2, RW1	XCHW RW2, @RW1+d8	XCHW RW3, RW1	XCHW RW3, @RW1+d8	XCHW RW4, RW1	XCHW RW4, @RW1+d8	XCHW RW5, RW1	XCHW RW5, @RW1+d8	XCHW RW6, RW1	XCHW RW6, @RW1+d8	XCHW RW7, RW1	XCHW RW7, @RW1+d8
+2	XCHW RW0, RW2	XCHW RW0, @RW2+d8	XCHW RW1, RW2	XCHW RW1, @RW2+d8	XCHW RW2, RW2	XCHW RW2, @RW2+d8	XCHW RW3, RW2	XCHW RW3, @RW2+d8	XCHW RW4, RW2	XCHW RW4, @RW2+d8	XCHW RW5, RW2	XCHW RW5, @RW2+d8	XCHW RW6, RW2	XCHW RW6, @RW2+d8	XCHW RW7, RW2	XCHW RW7, @RW2+d8
+3	XCHW RW0, RW3	XCHW RW0, @RW3+d8	XCHW RW1, RW3	XCHW RW1, @RW3+d8	XCHW RW2, RW3	XCHW RW2, @RW3+d8	XCHW RW3, RW3	XCHW RW3, @RW3+d8	XCHW RW4, RW3	XCHW RW4, @RW3+d8	XCHW RW5, RW3	XCHW RW5, @RW3+d8	XCHW RW6, RW3	XCHW RW6, @RW3+d8	XCHW RW7, RW3	XCHW RW7, @RW3+d8
+4	XCHW RW0, RW4	XCHW RW0, @RW4+d8	XCHW RW1, RW4	XCHW RW1, @RW4+d8	XCHW RW2, RW4	XCHW RW2, @RW4+d8	XCHW RW3, RW4	XCHW RW3, @RW4+d8	XCHW RW4, RW4	XCHW RW4, @RW4+d8	XCHW RW5, RW4	XCHW RW5, @RW4+d8	XCHW RW6, RW4	XCHW RW6, @RW4+d8	XCHW RW7, RW4	XCHW RW7, @RW4+d8
+5	XCHW RW0, RW5	XCHW RW0, @RW5+d8	XCHW RW1, RW5	XCHW RW1, @RW5+d8	XCHW RW2, RW5	XCHW RW2, @RW5+d8	XCHW RW3, RW5	XCHW RW3, @RW5+d8	XCHW RW4, RW5	XCHW RW4, @RW5+d8	XCHW RW5, RW5	XCHW RW5, @RW5+d8	XCHW RW6, RW5	XCHW RW6, @RW5+d8	XCHW RW7, RW5	XCHW RW7, @RW5+d8
+6	XCHW RW0, RW6	XCHW RW0, @RW6+d8	XCHW RW1, RW6	XCHW RW1, @RW6+d8	XCHW RW2, RW6	XCHW RW2, @RW6+d8	XCHW RW3, RW6	XCHW RW3, @RW6+d8	XCHW RW4, RW6	XCHW RW4, @RW6+d8	XCHW RW5, RW6	XCHW RW5, @RW6+d8	XCHW RW6, RW6	XCHW RW6, @RW6+d8	XCHW RW7, RW6	XCHW RW7, @RW6+d8
+7	XCHW RW0, RW7	XCHW RW0, @RW7+d8	XCHW RW1, RW7	XCHW RW1, @RW7+d8	XCHW RW2, RW7	XCHW RW2, @RW7+d8	XCHW RW3, RW7	XCHW RW3, @RW7+d8	XCHW RW4, RW7	XCHW RW4, @RW7+d8	XCHW RW5, RW7	XCHW RW5, @RW7+d8	XCHW RW6, RW7	XCHW RW6, @RW7+d8	XCHW RW7, RW7	XCHW RW7, @RW7+d8
+8	XCHW RW0, @RW0	XCHW RW0, @RW0+d16	XCHW RW1, @RW0	XCHW RW1, @RW0+d16	XCHW RW2, @RW0	XCHW RW2, @RW0+d16	XCHW RW3, @RW0	XCHW RW3, @RW0+d16	XCHW RW4, @RW0	XCHW RW4, @RW0+d16	XCHW RW5, @RW0	XCHW RW5, @RW0+d16	XCHW RW6, @RW0	XCHW RW6, @RW0+d16	XCHW RW7, @RW0	XCHW RW7, @RW0+d16
+9	XCHW RW0, @RW1	XCHW RW0, @RW1+d16	XCHW RW1, @RW1	XCHW RW1, @RW1+d16	XCHW RW2, @RW1	XCHW RW2, @RW1+d16	XCHW RW3, @RW1	XCHW RW3, @RW1+d16	XCHW RW4, @RW1	XCHW RW4, @RW1+d16	XCHW RW5, @RW1	XCHW RW5, @RW1+d16	XCHW RW6, @RW1	XCHW RW6, @RW1+d16	XCHW RW7, @RW1	XCHW RW7, @RW1+d16
+A	XCHW RW0, @RW2	XCHW RW0, @RW2+d16	XCHW RW1, @RW2	XCHW RW1, @RW2+d16	XCHW RW2, @RW2	XCHW RW2, @RW2+d16	XCHW RW3, @RW2	XCHW RW3, @RW2+d16	XCHW RW4, @RW2	XCHW RW4, @RW2+d16	XCHW RW5, @RW2	XCHW RW5, @RW2+d16	XCHW RW6, @RW2	XCHW RW6, @RW2+d16	XCHW RW7, @RW2	XCHW RW7, @RW2+d16
+B	XCHW RW0, @RW3	XCHW RW0, @RW3+d16	XCHW RW1, @RW3	XCHW RW1, @RW3+d16	XCHW RW2, @RW3	XCHW RW2, @RW3+d16	XCHW RW3, @RW3	XCHW RW3, @RW3+d16	XCHW RW4, @RW3	XCHW RW4, @RW3+d16	XCHW RW5, @RW3	XCHW RW5, @RW3+d16	XCHW RW6, @RW3	XCHW RW6, @RW3+d16	XCHW RW7, @RW3	XCHW RW7, @RW3+d16
+C	XCHW RW0, @RW0+	XCHW RW0, @RW0+RW7	XCHW RW1, @RW0+	XCHW RW1, @RW0+RW7	XCHW RW2, @RW0+	XCHW RW2, @RW0+RW7	XCHW RW3, @RW0+	XCHW RW3, @RW0+RW7	XCHW RW4, @RW0+	XCHW RW4, @RW0+RW7	XCHW RW5, @RW0+	XCHW RW5, @RW0+RW7	XCHW RW6, @RW0+	XCHW RW6, @RW0+RW7	XCHW RW7, @RW0+	XCHW RW7, @RW0+RW7
+D	XCHW RW0, @RW1+	XCHW RW0, @RW1+RW7	XCHW RW1, @RW1+	XCHW RW1, @RW1+RW7	XCHW RW2, @RW1+	XCHW RW2, @RW1+RW7	XCHW RW3, @RW1+	XCHW RW3, @RW1+RW7	XCHW RW4, @RW1+	XCHW RW4, @RW1+RW7	XCHW RW5, @RW1+	XCHW RW5, @RW1+RW7	XCHW RW6, @RW1+	XCHW RW6, @RW1+RW7	XCHW RW7, @RW1+	XCHW RW7, @RW1+RW7
+E	XCHW RW0, @RW2+	XCHW RW0, @PC +d16	XCHW RW1, @RW2+	XCHW RW1, @PC+d16	XCHW RW2, @RW2+	XCHW RW2, @PC+d16	XCHW RW3, @RW2+	XCHW RW3, @PC+d16	XCHW RW4, @RW2+	XCHW RW4, @PC+d16	XCHW RW5, @RW2+	XCHW RW5, @PC+d16	XCHW RW6, @RW2+	XCHW RW6, @PC+d16	XCHW RW7, @RW2+	XCHW RW7, @PC+d16
+F	XCHW RW0, @RW3+	XCHW RW0, addr16	XCHW RW1, @RW3+	XCHW RW1, addr16	XCHW RW2, @RW3+	XCHW RW2, addr16	XCHW RW3, @RW3+	XCHW RW3, addr16	XCHW RW4, @RW3+	XCHW RW4, addr16	XCHW RW5, @RW3+	XCHW RW5, addr16	XCHW RW6, @RW3+	XCHW RW6, addr16	XCHW RW7, @RW3+	XCHW RW7, addr16