

**F<sup>2</sup>MC-16F FAMILY**  
**16-BIT MICROCONTROLLERS**  
**MB90242A SERIES**  
HARDWARE MANUAL

# PREFACE

---

This manual contains important information about your Fujitsu semiconductor product. Please read it through carefully.

The MB90242A Series is a proprietary 16-bit single-chip microcontroller, developed as a general-purpose version of the F<sup>2</sup>MC-16F series of microcontrollers and is capable of use with ASIC (application specific IC) products.

This manual is intended for engineers using this semiconductor product in actual applications, and presents descriptions of MB90242A series functions and operation. Be sure to read the entire manual carefully. For details about instructions used with this product, refer to the "F<sup>2</sup>MC-16F Programming Manual."

\*: F<sup>2</sup>MC is an abbreviation for FUJITSU Flexible Microcontroller, and is a registered trademark.

This manual is organized as follows.

## Section 1. Overview

This section presents the models available in the MB90242A series, with an overview of each model.

## Section 2. Hardware Configuration

This section presents the internal configuration and operating modes of the F<sup>2</sup>MC-16F Family CPU, as well as the specifications of hardware components in the MB90242A series.

## Section 3. Operation

This section describes the use of the MB90242A series including reset sequences, external bus operation and power saving modes.

## Appendix

The appendix describes F<sup>2</sup>MC-16F address notation, and provides a instruction list and instruction maps.

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:	1
1.1 Features	1
1.2 Model Lineup	3
1.3 Blok Diagram	4
1.4 Pin Assignment	5
1.5 External Dimensions	8
1.6 Pin Description	10
1.7 Handling of Semiconductor Devices	19
Chapter 2:	21
2.1 CPU	21
2.2 Maps	66
2.3 Parallel Ports	74
2.4 IIR Filter DSP Unit	79
2.5 UART	93
2.6 SSI (Simple Serial Interface)	109
2.7 16-Bit Reload Timer (With Event Count Function)	114
2.8 16-Bit I/O Timer	124
2.9 A/D Converter	133
2.10 External Interrupts	141
2.11 Delayed Interrupt Generator Module	148
2.12 Watchdog Timer, Timebase Timer Functions	150
Chapter 3:	157
3.1 Clock Generator	157
3.2 Reset	158
3.3 Memory Access Mode	161
3.4 External Memory Access	166
3.5 Power Saving Modes	174
3.6 Pin Status in Sleep, Stop, Hold and Reset Modes	180
APPENDIX	183
APPENDIX A F <sup>2</sup> MC-16F Addressing Specifications	184
A.1 Effective Address Fields	185
A.2 Detailed Addressing Format Specifications	186
APPENDIX B F <sup>2</sup> MC-16F Instruction Lists	200
B.1 Instruction List Heading Descriptions	201
B.2 Instruction List Symbols	203
B.3 Effective Address Fields	205
B.4 Calculation of Execution Cycle Counts	206
B.5 Transfer Instructions	207
B.6 Numerical Calculation Instructions	211
B.7 Logical Calculation Instructions	216

B.8 Shift Instructions .....	218
B.9 Branching Instructions .....	219
B.10 Other Instructions .....	221
B.11 Execution Cycle Counts for Special Operations .....	226
APPENDIX C F <sup>2</sup> MC-16F Instruction Map .....	230
C.1 Basic Map Structure .....	231
C.2 Basic Page Map .....	233
C.3 Bit Operation Instruction Map .....	234
C.4 MOVM Instruction Map .....	235
C.5 Character String Operation Map .....	236
C.6 2-Byte Instruction Map .....	237
C.7 ea Instructions .....	238
C.8 MOVEA RWi, ea .....	247
C.9 MOV Ri, ea .....	248
C.10 MOVW RWi, ea .....	249
C.11 MOV ea, Ri .....	250
C.12 MOVW ea, RWi .....	251
C.13 XCH Ri, ea .....	252
C.14 XCHW RWi, ea .....	253

# Chapter 1:

## Overview

---

The MB90242A family of 16-bit single-chip microcontrollers is optimized for electro-mechanical control of devices such as hard disk drive devices.

The instruction system preserves the AT architecture used in the F<sup>2</sup>MC-16 and 16H series, with the addition of instructions for supporting high-level languages, expanded addressing mode, enhanced multiplication and division instructions and improved bit processing instructions. Also, the addition of a 32-bit accumulator enables processing of long-word data.

Peripheral resources include a sum-of-products calculation unit for easier realization of IIR or FIR digital filter functions. A wealth of additional on-chip components include 6-channel 8/10-bit ADC, UART, 2+1-channel timer, 4-channel input capture, and 4-channel external interrupt.

### 1.1 Features

- F<sup>2</sup>MC-16F CPU Core
  - Minimum instruction execution time ..... 62.5 ns (at crystal oscillator frequency 32 MHz: 5 V±10%)
  - Optimum instruction system for controller applications
  - Instructions enhanced for high-level language (C) and multitasking
  - Improved execution speed ..... 8-byte queuing
  - Powerful interrupt functions (interrupt processing time 1.0µs: at crystal oscillator frequency 32 MHz)
  - Instruction-independent automatic transfer function
  - Expanded intelligent I/O service
- DSP Unit
  - Functions tailored for IIR calculation
  - Adds up to 8-term multiplication results using 16-bit x 16-bit coding
  - The following formula  $Y_k = \sum_{n=0}^N b_n Y_{k-n} + \sum_{m=0}^M a_m X_{k-m}$  execution time 0.625 ns (at crystal oscillator frequency 32 MHz, N=M=3)
  - Up to 3 independent settings for N and M in the above formula
- Internal RAM RAM: 2 Kbytes
  - Mode selection allows RAM data to execute as CPU instructions
- General-Purpose Ports Up to 38 ports
- A/D Converter ..... Analog inputs: 6
  - Resolution: 10 bits
  - Conversion time: 1.25 µs (minimum)
  - 8/10-bit switching available
  - Conversion results storage registers: 4

## 1.1 Features

- 8-Bit UART ..... 1 channel
- 8/16-Bit I/O Simple Serial Interface (max 8 Mbps)... 1 channels
- 16-Bit Free-Run Timer ..... Operating clock: 0.25  $\mu$ s
- 16-Bit Input Capture ..... 4 channels
  - Selection of edge detectors
- 16-Bit Reload Timer ..... 2 channels
- External Interrupts ..... 4 channels
- Timebase Timer ..... 18 bits
- Watchdog Timer
- Clock Gear Function
- Power Saving Modes
  - Sleep mode
  - Stop mode
  - Hardware disk standby mode
- Low Voltage Operation
- Package ..... SQFP-80
- CMOS 0.8  $\mu$ m technology

## 1.2 Model Lineup

Table 1.2 shows the models in the MB90242A series.

**Table 1.2 MB90242A Model lineup**

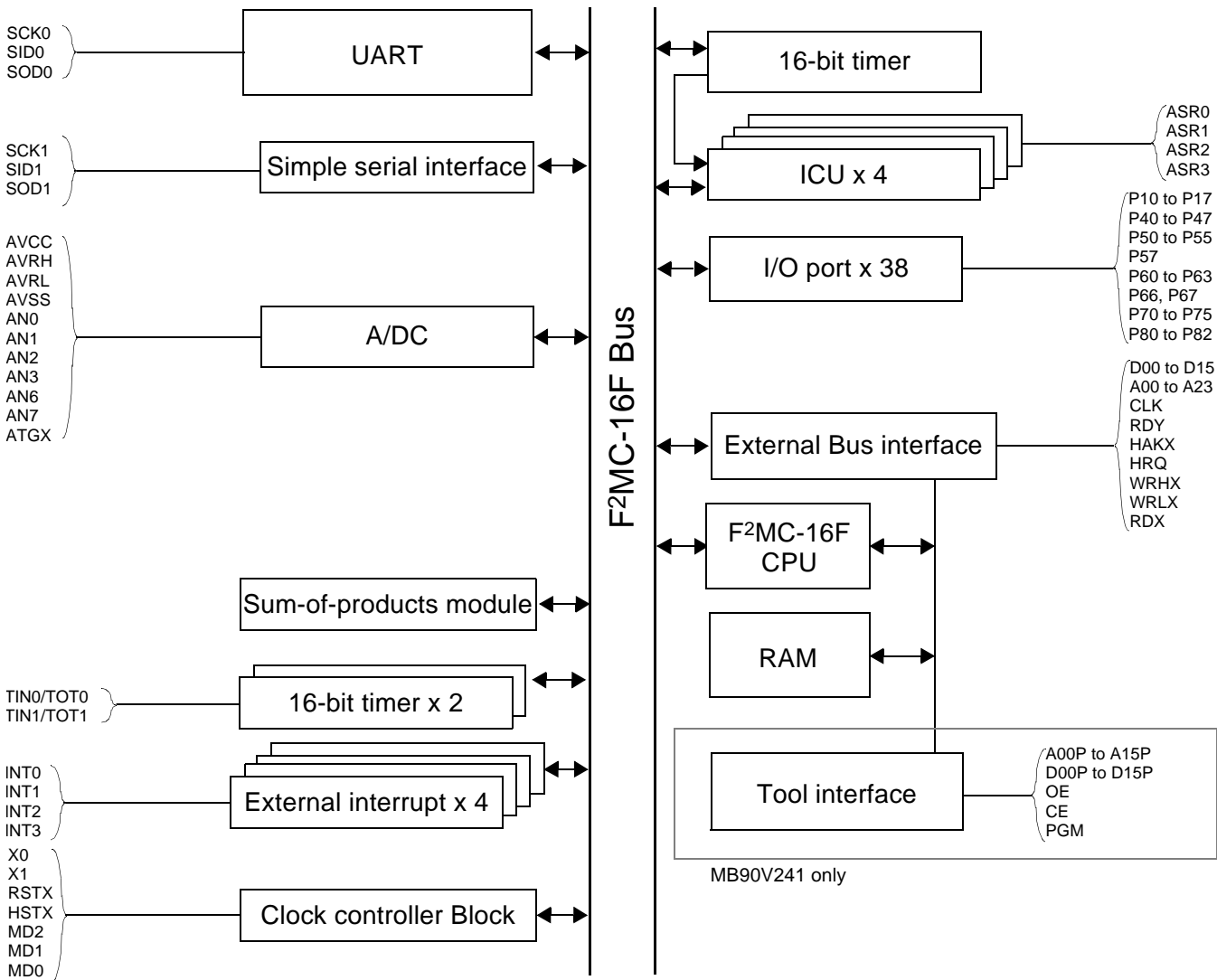
	Package	RAM capacity	Remarks
MB90242A	SQFP-80	2 Kbytes	Mass production device
MB90V241	PGA-256	4Kbytes	Evaluation device

**Note:** RAM capacity shown above is exclusive of 64-byte RAM used in sum-of-products calculation.



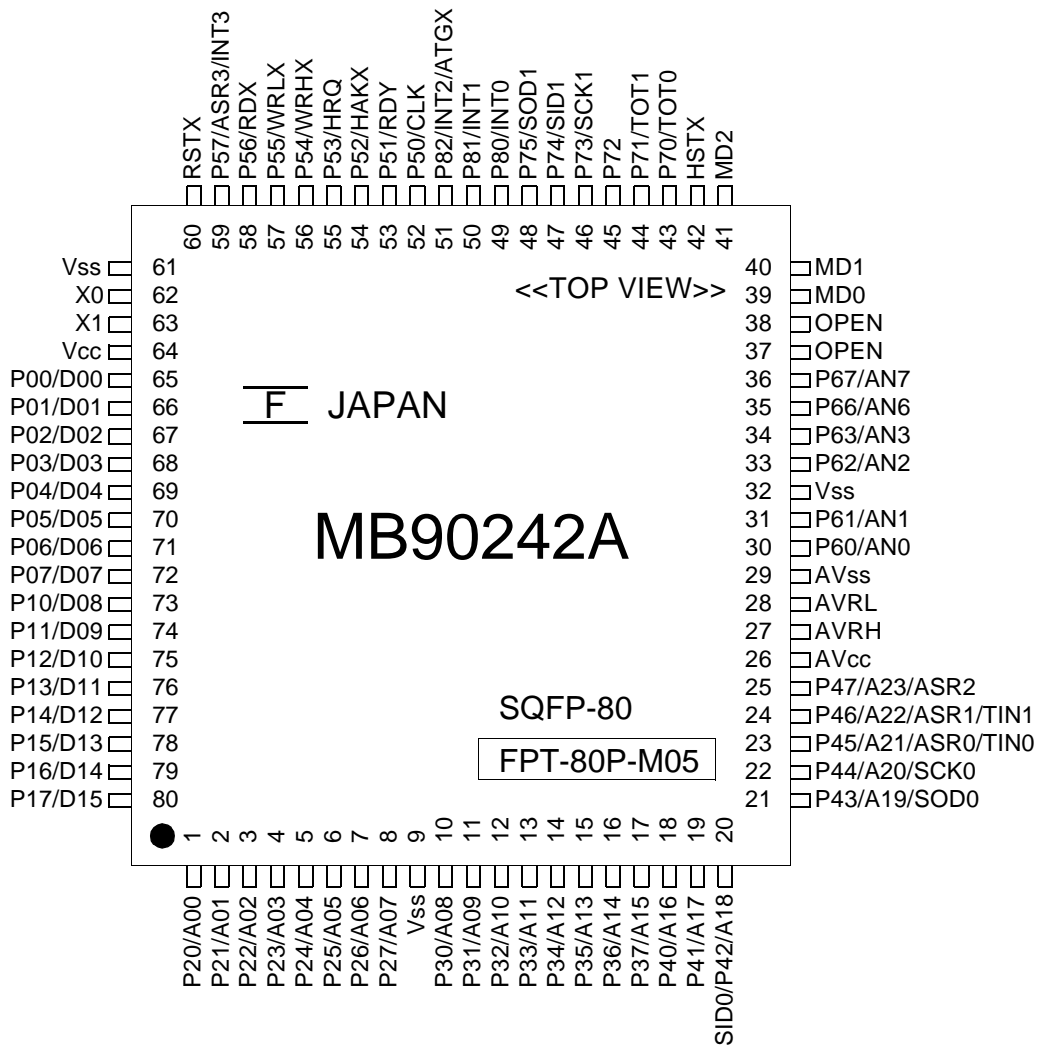
### 1.3 Blok Diagram

## 1.3 Blok Diagram



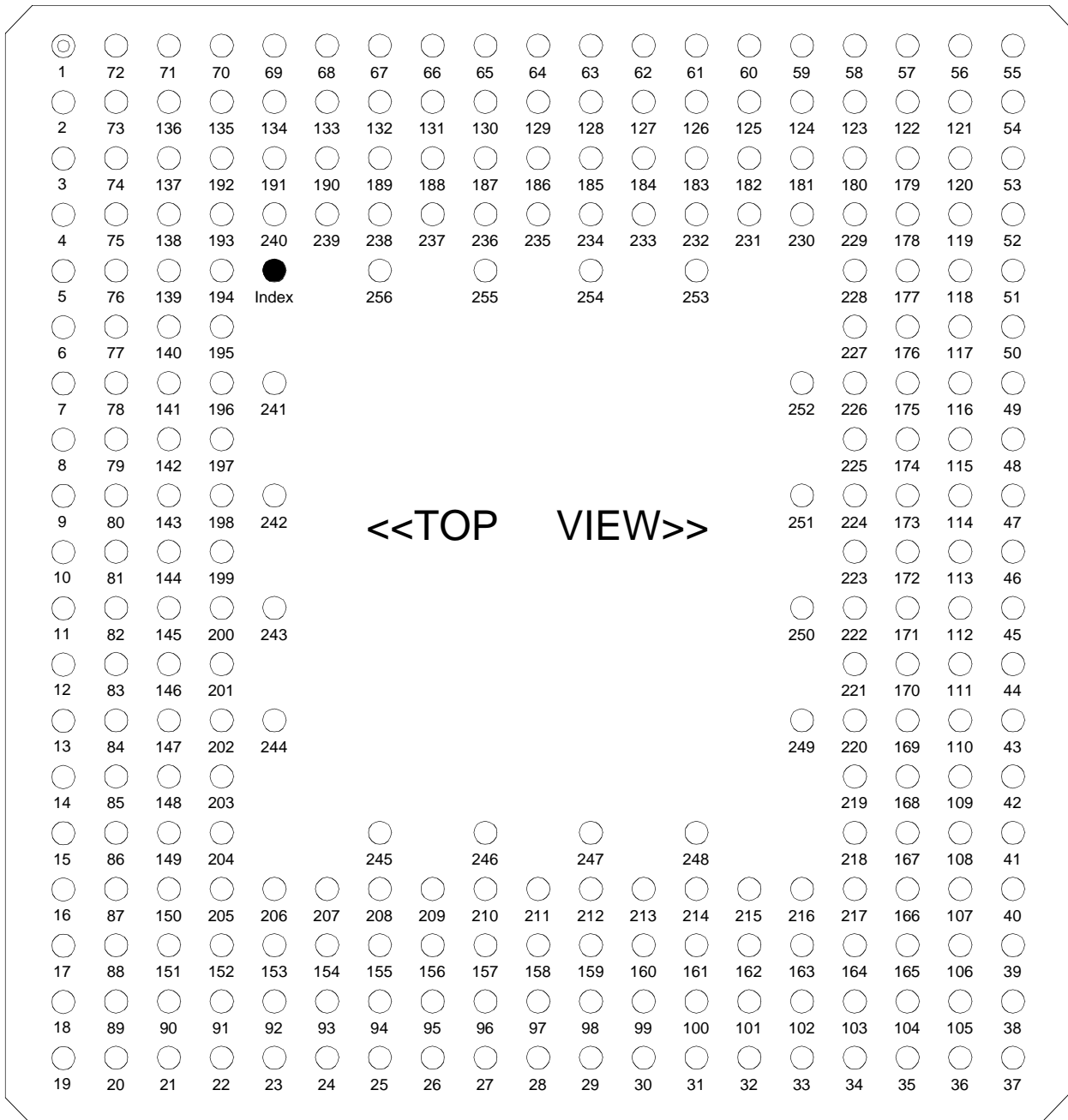
# 1.4 Pin Assignment

## ■ MB90242A



## 1.4 Pin Assignment

### ■ MB90V241



## 1.4 Pin Assignment

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Nam
1	I.C.	38	MD2	75	N.C.	112	I.C.	149	P42	186	I.C.	223	Vss
2	I.C.	39	P82	76	P01	113	I.C.	150	P46	187	I.C.	224	I.C.
3	N.C.	40	P53	77	P04	114	I.C.	151	N.C.	188	AVRH	225	I.C.
4	P00	41	N.C.	78	P07	115	I.C.	152	P74	189	P60	226	Vss
5	P03	42	N.C.	79	P12	116	I.C.	153	P81	190	P66	227	I.C.
6	P05	43	I.C.	80	P16	117	I.C.	154	D01P	191	I.C.	228	Vss
7	P10	44	I.C.	81	P21	118	I.C.	155	D05P	192	I.C.	229	N.C.
8	P13	45	I.C.	82	P23	119	I.C.	156	D08P	193	I.C.	230	Vss
9	P17	46	I.C.	83	P27	120	I.C.	157	D12P	194	Vss	231	I.C.
10	N.C.	47	I.C.	84	P31	121	N.C.	158	A00P	195	X1	232	Vss
11	P22	48	I.C.	85	P34	122	I.C.	159	A02P	196	Vss	233	I.C.
12	P26	49	I.C.	86	P37	123	I.C.	160	A06P	197	N.C.	234	I.C.
13	N.C.	50	I.C.	87	P43	124	I.C.	161	CE	198	P14	235	Vss
14	P33	51	I.C.	88	P45	125	I.C.	162	A11P	199	Vss	236	RSTX
15	P35	52	I.C.	89	N.C.	126	I.C.	163	A15P	200	P25	237	N.C.
16	P40	53	I.C.	90	P73	127	I.C.	164	P.D.	201	P30	238	Vss
17	P44	54	I.C.	91	P80	128	N.C.	165	N.C.	202	Vss	239	DVRL
18	P47	55	I.C.	92	D00P	129	I.C.	166	N.C.	203	P41	240	Vss
19	P71	56	I.C.	93	D03P	130	I.C.	167	P51	204	Vss	241	Vcc
20	P72	57	I.C.	94	D06P	131	AVcc	168	P55	205	P70	242	Vcc
21	P75	58	I.C.	95	D09P	132	AVss	169	N.C.	206	Vss	243	Vcc
22	N.C.	59	I.C.	96	D13P	133	P62	170	I.C.	207	N.C.	244	Vcc
23	D02P	60	I.C.	97	N.C.	134	P67	171	I.C.	208	Vss	245	Vcc
24	D04P	61	I.C.	98	A01P	135	I.C.	172	I.C.	209	D07P	246	Vcc
25	N.C.	62	I.C.	99	A05P	136	I.C.	173	I.C.	210	D11P	247	Vcc
26	D10P	63	N.C.	100	PGM	137	N.C.	174	I.C.	211	Vss	248	Vcc
27	D14P	64	N.C.	101	A09P	138	I.C.	175	N.C.	212	A03P	249	Vcc
28	D15P	65	I.C.	102	A12P	139	X0	176	I.C.	213	P.D.	250	Vcc
29	N.C.	66	N.C.	103	OE	140	P02	177	I.C.	214	Vss	251	Vcc
30	A04P	67	AVRL	104	P.D.	141	P06	178	I.C.	215	A14P	252	Vcc
31	A07P	68	P61	105	VPP	142	P11	179	N.C.	216	Vss	253	Vcc
32	A08P	69	P61	106	HSTX	143	P15	180	I.C.	217	MD0	254	Vcc
33	A10P	70	DVRH	107	P50	144	P20	181	I.C.	218	Vss	255	Vcc
34	A13P	71	I.C.	108	P54	145	P24	182	I.C.	219	P52	256	Vcc
35	P.D.	72	I.C.	109	P56	146	N.C.	183	I.C.	220	Vss		
36	P.D.	73	I.C.	110	P57	147	P32	184	I.C.	221	I.C.		
37	MD1	74	I.C.	111	I.C.	148	P36	185	I.C.	222	I.C.		

P.D. (Pull Down): Requires an external pull-down resistor.

N.C. (Non Connection): Not connected.

I.C. (Internal Connection): Connected to internal pin protection circuit.

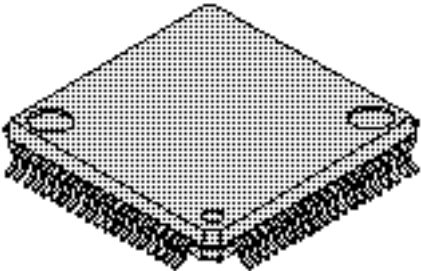
**Fig. 1.4.3 Pin Assignment**

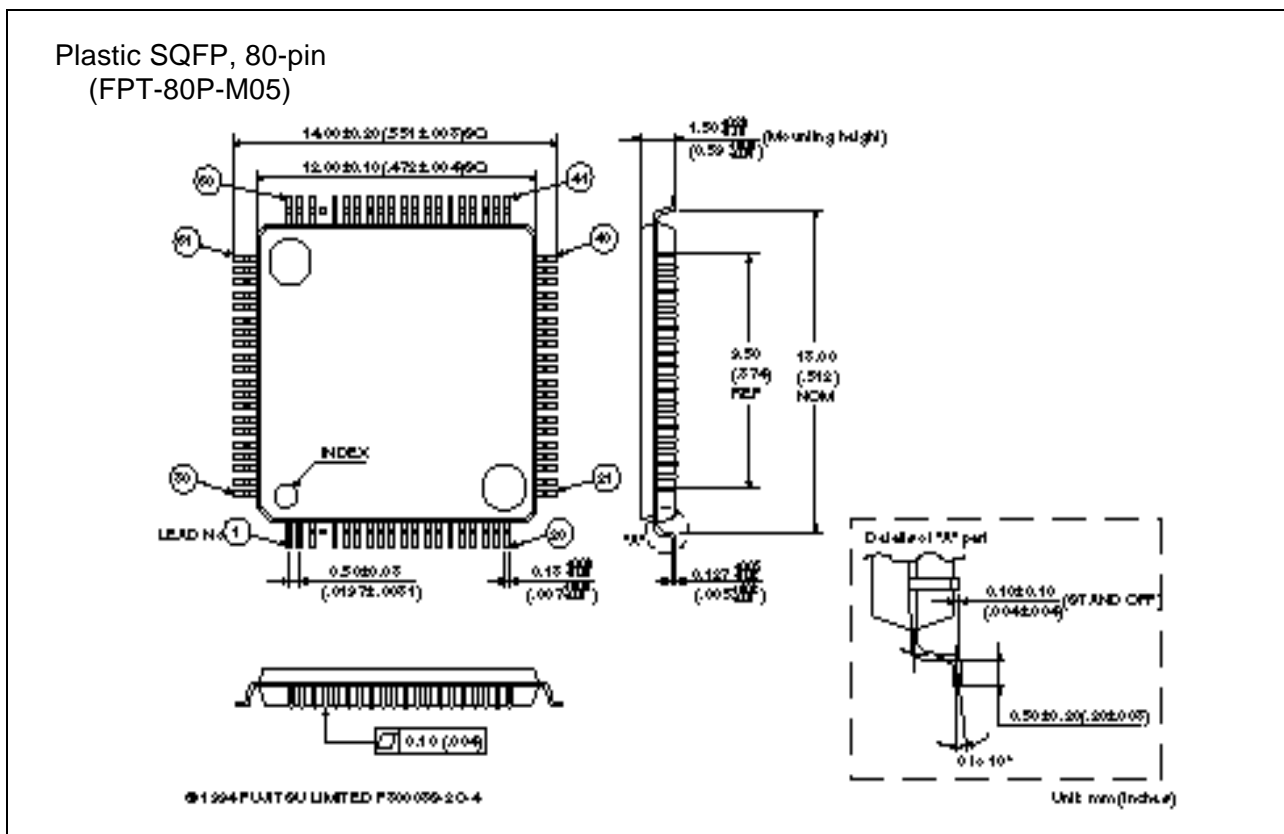
## 1.5 External Dimensions

■ MB90242A

### FPT-80P-M05

EIAJ Code: \*QFP080-P-1212-1

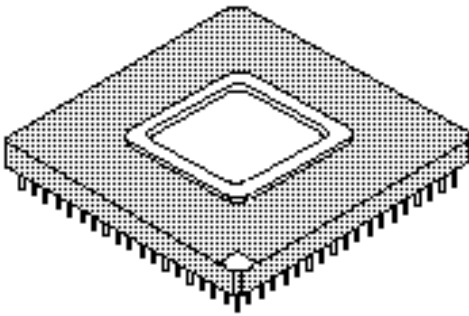
<p>Plastic SQFP, 80-pin</p>  <p>(FPT-80P-M05)</p>	Lead pitch	0.50 mm	
	Package width × package length	2 × 12 mm	
	Lead shape	Gull-wing	
	Sealing method	Plastic molding	



■ MB90V241

**PGA-256C-A02**

EIAJ Code: \*PGA257-C-S19U-2

 <p>Ceramic PGA, 256-pin (PGA-256C-A02)</p>	pin count	257 pins (extra index pin included)
	Lead pitch	100 mil
	Pin matrix	19
	Sealing method	Metal sealing

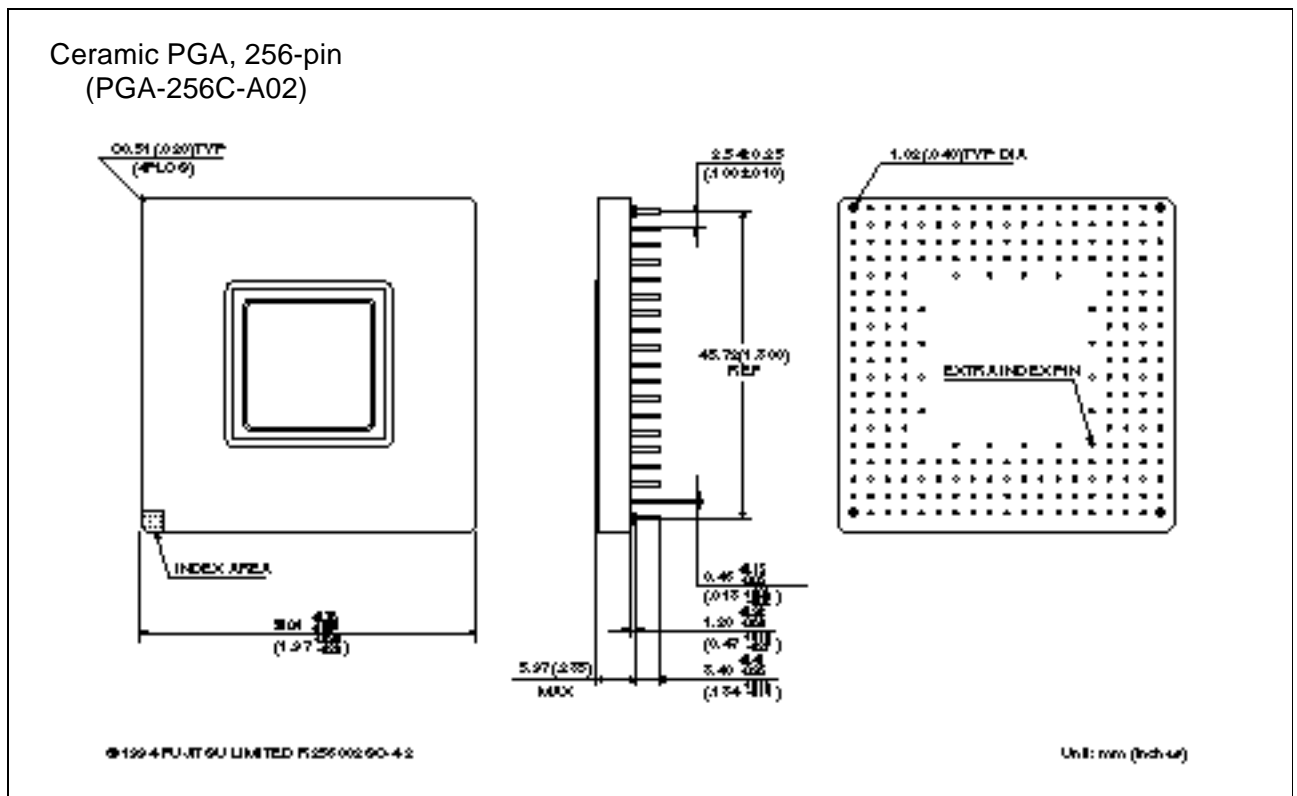


Fig. 1.4.4 PGA-256C-A02

## 1.6 Pin Description

**Table 1.6 Pin Description**

Pin No.	Name	Circuit type	Function
1 to 8	P20/A00 to P27/A07	F	P20 to P27 are not available for use as general-purpose ports. A00 to A07 are the output pins for the lower 8 bits of the external address bus.
9	VSS	power supply	Ground level pin for digital circuits
10 to 17	P30/A08 to P37/A15	F	P30 to P37 are not available for use as general-purpose ports. A08 to A15 are the output pins for the middle 8 bits of the external address bus.
18	P40/A16	F	P40 is a general-purpose I/O port. This function is enabled when the corresponding bit in the upper address control register is set to 'port.' A16 is the output pin for bit 16 of the external address bus. This function is enabled when the corresponding bit in the upper address control register is set to 'address.'
19	P41/A17	F	P41 is a general-purpose I/O port. This function is enabled when the corresponding bit in the upper address control register is set to 'port.' A17 is the output pin for bit 17 of the external address bus. This function is enabled when the corresponding bit in the upper address control register is set to 'address.'
20	P42/A18/SID0	F	P42 is a general-purpose I/O port. This function is enabled when the corresponding bit in the upper address control register is set to 'port.' A18 is the output pin for bit 18 of the external address bus. This function is enabled when the corresponding bit in the upper address control register is set to 'address.' SID0 is the data input pin for UART #0. Because this function is in continual use during UART #0 input operations, I/O access from other functions should not be attempted unless intended.
21	P43/A19/SOD0	F	P43 is a general-purpose I/O port. This function is enabled when UART #0 data output is disabled and the corresponding bit in the upper address control register is set to 'port.' A19 is the output pin for bit 19 of the external address bus. This function is enabled when UART #0 data output is disabled and the corresponding bit in the upper address control register is set to 'address.' SOD0 is the data output pin for UART #0. This function is enabled when UART #0 data output is enabled.

**Table 1.6 Pin Description (Continued)**

Pin No.	Name	Circuit type	Function
22	P44/A20/SCK0	F	<p>P44 is a general-purpose I/O port. This function is enabled when UART #0 clock output is disabled and the corresponding bit in the upper address control register is set to 'port.'</p> <p>A20 is the output pin for bit 20 of the external address bus. This function is enabled when UART #0 clock output is disabled and the corresponding bit in the upper address control register is set to 'address.'</p> <p>SCK0 is the clock signal output pin for UART #0. This function is enabled when UART #0 clock output is enabled.</p>
23	P45/A21/ASR0 /TIN0	F	<p>P45 is a general-purpose I/O port. This function is enabled when the corresponding bit in the upper address control register is set to 'port.'</p> <p>A21 is the output pin for bit 21 of the external address bus. This function is enabled when the corresponding bit in the upper address control register is set to 'address.'</p> <p>ASR0 is the data input pin for input capture #0. This function is used for input at all times when input capture #0 is receiving input, and therefore I/O access from other functions should not be attempted unless intended.</p> <p>TIN0 is the data input pin for 16-bit timer #0. This function is used for input at all times when 16-bit timer #0 is receiving input, and therefore I/O access from other functions should not be attempted unless intended.</p>
24	P46/A22/ASR1 /TIN1	F	<p>P46 is a general-purpose I/O port. This function is enabled the corresponding bit in the upper address control register is set to 'port.'</p> <p>A22 is the output pin for bit 22 of the external address bus. This function is enabled when the corresponding bit in the upper address control register is set to 'address.'</p> <p>ASR1 is the data input pin for input capture #1. This function is used for input at all times when input capture #1 is receiving input, and therefore I/O access from other functions should not be attempted unless intended.</p> <p>TIN1 is the data input pin for 16-bit timer #1. This function is used for input at all times when 16-bit timer #1 is receiving input, and therefore I/O access from other functions should not be attempted unless intended.</p>
25	P47/A23/ASR2	F	<p>P47 is a general-purpose I/O port. This function is enabled when the corresponding bit in the upper address control register is set to 'port.'</p> <p>A23 is the output pin for bit 23 of the external address bus. This function is enabled when the corresponding bit in the address high control register is set to 'address.'</p> <p>ASR2 is the data input pin for input capture #2. This function is used for input at all times when input capture #2 is receiving input, and therefore I/O access from other functions should not be attempted unless intended.</p>



**Table 1.6 Pin Description (Continued)**

Pin No.	Name	Circuit type	Function
26	AVCC	Power supply	Analog circuit power supply. Power must only be switched on or off when a potential greater than AVCC is applied to the VCC terminal.
27	AVRH	Power supply	This is an A/D converter external reference voltage input pin. This pin should only be switched on or off when a potential greater than AVRH is applied to the AVCC pin.
28	AVRL	Power supply	This is an A/D converter external reference voltage input pin.
29	AVSS	Power supply	Analog circuit ground level pin.
30 to 31	P60/AN0 and P61/AN1	H	P60 and P61 are N-channel open drain type I/O ports. When the corresponding bits in the ADER register is set to '0,' the level of these pins is read, resulting from read access to data registers by all read instructions (except for read-modify-write instructions). Values written in data registers are output directly from these pins. AN0 and AN1 are A/D converter analog input pins. To use this function, set the corresponding bit in the ADER register to '1' and set the corresponding bits in the data register to '1.'
32	VSS	Power supply	Digital circuit ground level pin.
33 to 34	P62/AN2 and P63/AN3	H	P62 and P63 are N-channel open drain type I/O ports. When the corresponding bits in the ADER register is set to '0,' the level of these pins is read, resulting from read access to data registers by all read instructions (except for read-modify-write instructions). Values written in data registers are output directly from these pins. AN2 to AN3 are A/D converter analog input pins. To use this function, set the corresponding bit in the ADER register to '1' and set the corresponding bits in the data register to '1.'
35 to 36	P66/AN6 and P67/AN7	H	P66 and P67 are N-channel open drain type I/O ports. When the corresponding bits in the ADER register is set to '0,' the level of these pins is read, resulting from read access to data registers by all read instructions (except for read-modify-write instructions). Values written in data registers are output directly from these pins. AN6 and AN7 are A/D converter analog input pins. To use this function, set the corresponding bit in the ADER register to '1' and set the corresponding bits in the data register to '1.'
37	OPEN		Open terminal. No internal connection.
38	OPEN		Open terminal. No internal connection.
39 to 41	MD0 to MD2	C	Operating mode setting input pins. These pins should be directly connected to Vcc or Vss.
42	HSTX	D	Hardware standby input pin.

**Table 1.6 Pin Description (Continued)**

Pin No.	Name	Circuit type	Function
43 to 45	P70/TOT0, P71/TOT1, P72	I	P70 to P72 are general-purpose I/O ports. This function is enabled when output from 16-bit timers #0 to #1 is also disabled. TOT0 to TOT1 are 16-bit timer output pins. This function is enabled when output from 16-bit timers #0 to #1 is also enabled.
46	P73/SCK1	F	P73 is a general-purpose I/O port. This function is enabled when clock output from SSI #1 is disabled. SCK1 is the SSI #1 clock output. This function is enabled when clock output from SSI #1 is enabled.
47	P74/SID1	F	P74 is a general-purpose I/O port. This function is enabled at all times. SID1 is the data input pins for SSI #1. This function is used for input at all times when SSI #1 is receiving input, and therefore I/O access from other functions should not be attempted unless intended.
48	P75/SOD1	F	P75 is a general-purpose I/O port. This function is enabled when data output from SSI #1 is disabled. SOD0 is the SSI #1 data output. This function is enabled when data output from SSI #1 is enabled.
49 to 50	P80/INT0 and P81/INT1	G	P80 and P81 are general-purpose I/O ports. This function is enabled at all times. INT0 to INT1 are external interrupt input pins. This function is used for input at all times when the external interrupt function is enabled, and therefore I/O access from other functions should not be attempted unless intended.
51	P82/INT2 /ATGX	F	P82 is a general-purpose I/O port. This function is enabled at all times. INT2 is an external interrupt input pin. This function is used for input at all times when the external interrupt function is enabled, and therefore I/O access from other functions should not be attempted unless intended. This pin is held at low level when the CPU is in stop mode, and therefore the signal for recovery from stop mode should be input at INT0 or INT1. ATGX is the A/D converter start trigger input pin. This function is used for input at all times when the A/D converter is in standby mode, and therefore I/O access from other functions should not be attempted unless intended.
52	P50/CLK	F	P50 is a general-purpose I/O port. This function is enabled when the clock signal output setting is disabled. CLK is the clock signal output pin. This function is enabled when clock signal output setting is enabled.
53	P51/RDY	E	P51 is a general-purpose I/O port. This function is enabled when the ready function is disabled. RDY is the ready signal input pin. This function is enabled when the ready function is enabled.

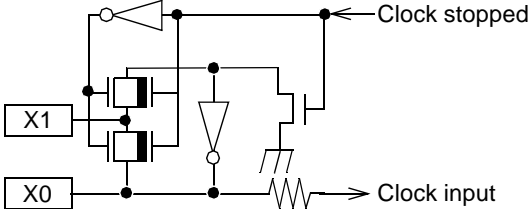
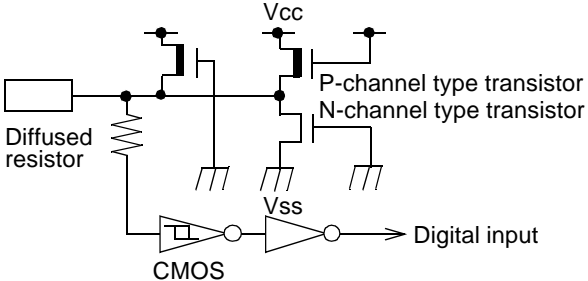
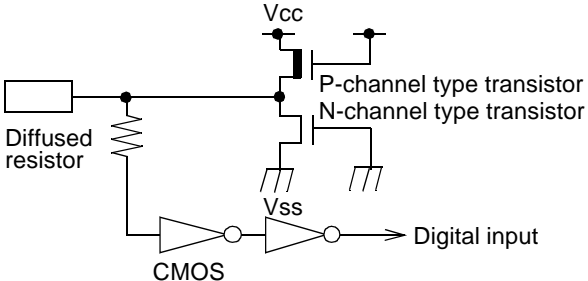
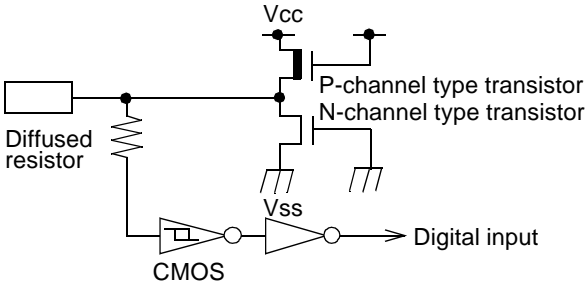
**Table 1.6 Pin Description (Continued)**

Pin No.	Name	Circuit type	Function
54	P52/HAKX	E	P52 is a general-purpose I/O port. This signal is enabled when the hold function is disabled. HAKX is the hold acknowledge signal output pin. This function is enabled when the hold function is enabled.
55	P53/HRQ	E	P53 is a general-purpose I/O port. This signal is enabled when the hold function is disabled. HRQ is the hold request signal input pin. This function is enabled when the hold function is enabled.
56	P54/WRHX	F	P54 is a general-purpose I/O port. This function is enabled when operating in 8-bit external bus mode or when WR pin output is disabled. WRHX is the write strobe signal output pin for the high 8-bit portion of the data bus. This function is enabled when operating in 16-bit external bus mode, or when WR pin output is enabled.
57	P55/WRLX	F	P55 is a general-purpose I/O port. This function is enabled when the WR pin 'output' setting is disabled. WRLX is the write strobe signal output pin for the low 8-bit portion of the data bus. This function is enabled when WR pin output is enabled.
58	P56/RDX	F	P56 is not available for use as a general-purpose port. RDX is the read strobe output signal for the data bus.
59	P57/ASR3/INT3	F	P57 is a general-purpose I/O port. ASR3 is the data input pin for input capture #3. This function is used for input at all times when input capture #3 is enabled for input, and therefore I/O access from other functions should not be attempted unless intended. INT3 is the data input pin for external interrupt #3. This function is used for input at all times when external interrupt #3 is enabled for input, and therefore I/O access from other functions should not be attempted unless intended.
60	RSTX	B	External reset request input.
61	VSS	Power supply	Digital circuit ground level pin.
62 63	X0 X1	A	Crystal oscillator signal pins (32 MHz).
64	VCC	Power supply	Digital circuit power supply.
65 to 72	P00/D00 to P07/D07	E	P00 to P07 are not available for use as general-purpose ports. D00 to D07 are I/O pins for the lower 8 bits of the external data bus.
73 to 80	P10/D08 to P07/D15	E	P10 to P17 are general-purpose I/O ports. This function is enabled when in 8-bit external bus mode. D08 to D15 are I/O pins for the upper 8 bits of the external data bus. This function is enabled when in 16-bit external bus mode.

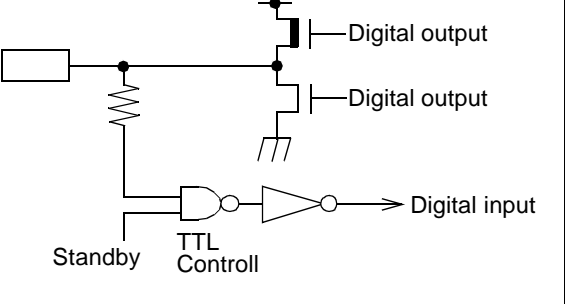
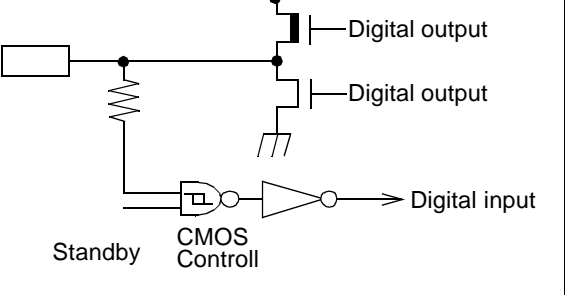
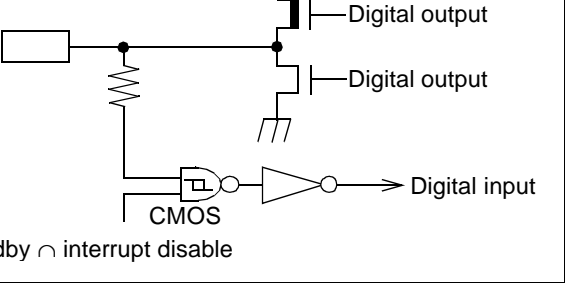
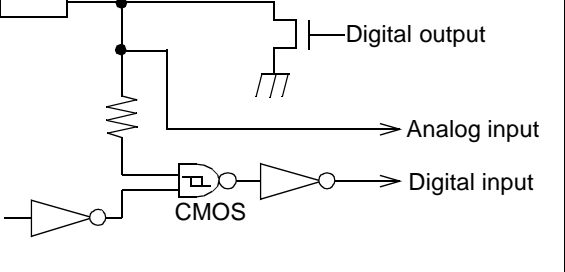
**Table 1.6 Pin Description (Continued)**

Pin No.	Name	Circuit type	Function
–	A15P to A00P	F	Piggyback address pins. ROM connected to these pins is treated as having equivalent operation to internal ROM.
–	D15P to D00P	F	Piggyback address pins. ROM connected to these pins is treated as having equivalent operation to internal ROM.
–	CE, OE, PGM	F	Piggyback address pins. Connect to pins corresponding to ROM.

**Table 1.6.1 Input/Output Circuit Configurations (1)**

Type	Circuit configuration	Remarks
A		<p>-For 32 MHz operation                      -Oscillator feedback resistor:                      approx. 1 MΩ</p>
B		<p>-CMOS level hysteresis input                      No standby control                      Pull-up resistor: approx. 50 kΩ</p>
C		<p>-CMOS level input                      No standby control</p>
D		<p>-CMOS level hysteresis input                      No standby control</p>

**Table 1.6.1 Input/Output Circuit Configurations (2)**

Type	Circuit configuration	Remarks
E		<p>-CMOS level output TTL level input Standby control included</p>
F		<p>-CMOS level output CMOS level hysteresis input Standby control included</p>
G		<p>-CMOS level output CMOS level hysteresis input Standby control included (when interrupt disabled)</p>
H		<p>-N-channel open drain CMOS level output CMOS level hysteresis input Analog input Analog input control included</p>

**Table 1.6.1 Input/Output Circuit Configurations (3)**

Type	Circuit configuration	Remarks
I	<p>The diagram illustrates the internal circuit for Type I. A pin is connected to a resistor. The node between the resistor and the pin is connected to the gates of two PMOS transistors, both labeled 'Digital output'. This node is also connected to an 'Analog output' terminal. A 'Standby' input is connected to the resistor. A 'CMOS Control' block is connected to the resistor and a 'Digital input' terminal.</p>	<ul style="list-style-type: none"> <li>-CMOS level output</li> <li>Analog output</li> <li>CMOS level hysteresis input</li> <li>Standby control included</li> </ul>

## 1.7 Handling of Semiconductor Devices

### (1) Preventing Latch-Up

A phenomenon called latch-up may occur on CMOS IC devices if voltage higher than  $V_{cc}$  or lower than  $V_{ss}$  is applied to input and output pins, or if voltage exceeding the rated voltage is applied between  $V_{cc}$  and  $V_{ss}$ . When latch-up occurs, supply current levels increase rapidly and can result in thermal damage to semiconductor elements. Sufficient care must be taken to avoid exceeding maximum rated values.

For the same reason, care must be taken to ensure that analog power supply levels do not exceed the level of the digital power supply.

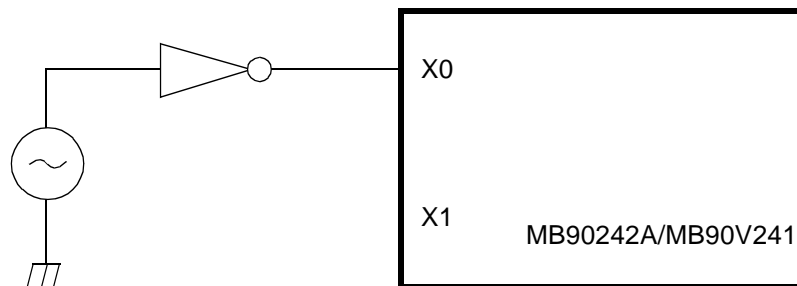
### (2) Handling Unused Input Pins

Unused input pins can cause devices to malfunction if left open, and should therefore be pulled up or down as needed.

### (3) Precautions for Use of External Clock

When an external clock is used, the signal should drive the X0 pin only, the X1 pin should be left open.

Figure 1.7.1 shows an example of an external clock connection.



**Fig. 1.7.1 Example: Use of an External Clock**

### (4) Power Supply Pins

When there are multiple  $V_{cc}$  or  $V_{ss}$  pins, semiconductor device design requires that in order to prevent malfunctions such as latch-up, all internal elements of equivalent potential be connected. Also, to prevent abnormal strobe signal operation due to unwanted lowering of emission or increases in ground level, and to maintain standards for total output current, all elements must be connected to external power supplies and grounds.

In addition it is recommended that  $V_{cc}$  and  $V_{ss}$  of this device be connected with as little impedance as possible from the current supply source.

It is further recommended that an approx. 0.1  $\mu\text{F}$  ceramic capacitor be placed near the device and connected between the  $V_{cc}$  and  $V_{ss}$  terminals as a bypass capacitor.

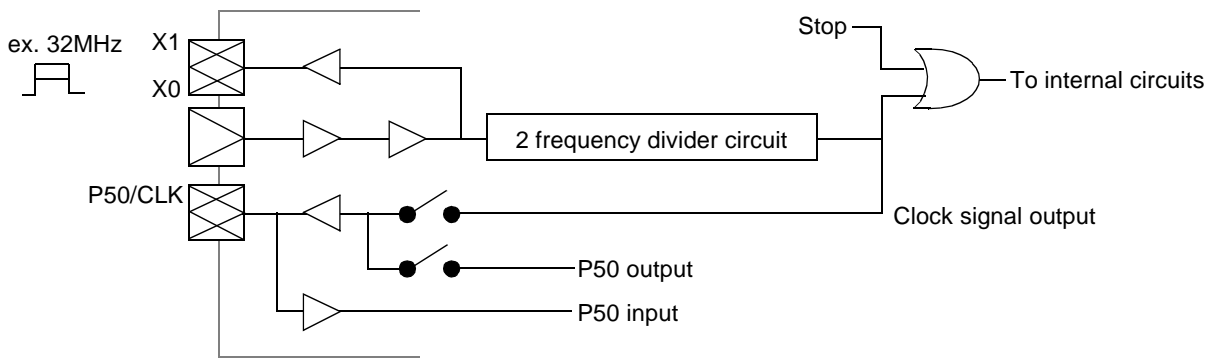
### (5) Crystal Oscillator Circuits

Noise in the vicinity of the X0 and X1 pins can be a cause of abnormal operation in this device. Designers should ensure that the X0 and X1 pins, and the crystal (or ceramic) oscillators as well as the bypass capacitor to ground be placed as close together as possible, and that PC board wiring layouts provide as little interference as possible from other wiring.

Also, PC board artwork can contribute toward stable of operation by surrounding the X0 and X1 pins with ground. This is strongly recommended.



**(6) CLK Signal Pin**



\* In external bus mode, the CLK output from the P50/CLK pin is selected as the initial value.

**(7) HSTX pin**

Make the HSTX pin high upon power-up. If the HSTX pin is used as low level, make the RSTX pin high.

# Chapter 2:

## Hardware Configuration

---

### 2.1 CPU

The F<sup>2</sup>MC-16F CPU core is a high-performance 16-bit CPU designed for applications requiring high-speed, real-time processing such as industrial applications, office automation (OA) products, and automotive devices. The F<sup>2</sup>MC-16F instruction set is designed to be optimized for controller applications, and can handle a wide variety of control functions with high speed and high efficiency. In addition, while the F<sup>2</sup>MC-16F core is designed as a 16-bit data processing CPU, an on-chip 32-bit accumulator is included for handling of 32-bit data. This enables a number of instructions to include 32-bit data processing capability. Memory space can be expanded to a maximum of 16 Mbytes, and can be accessed by either the linear pointer or bank access method. The instruction set, based on F<sup>2</sup>MC-16 architecture, has been enhanced with additional instructions for high level languages, expanded addressing mode and additional coded multiplication and division instructions. The F<sup>2</sup>MC-16F is upwardly compatible with F<sup>2</sup>MC-16 CPUs at object code level.

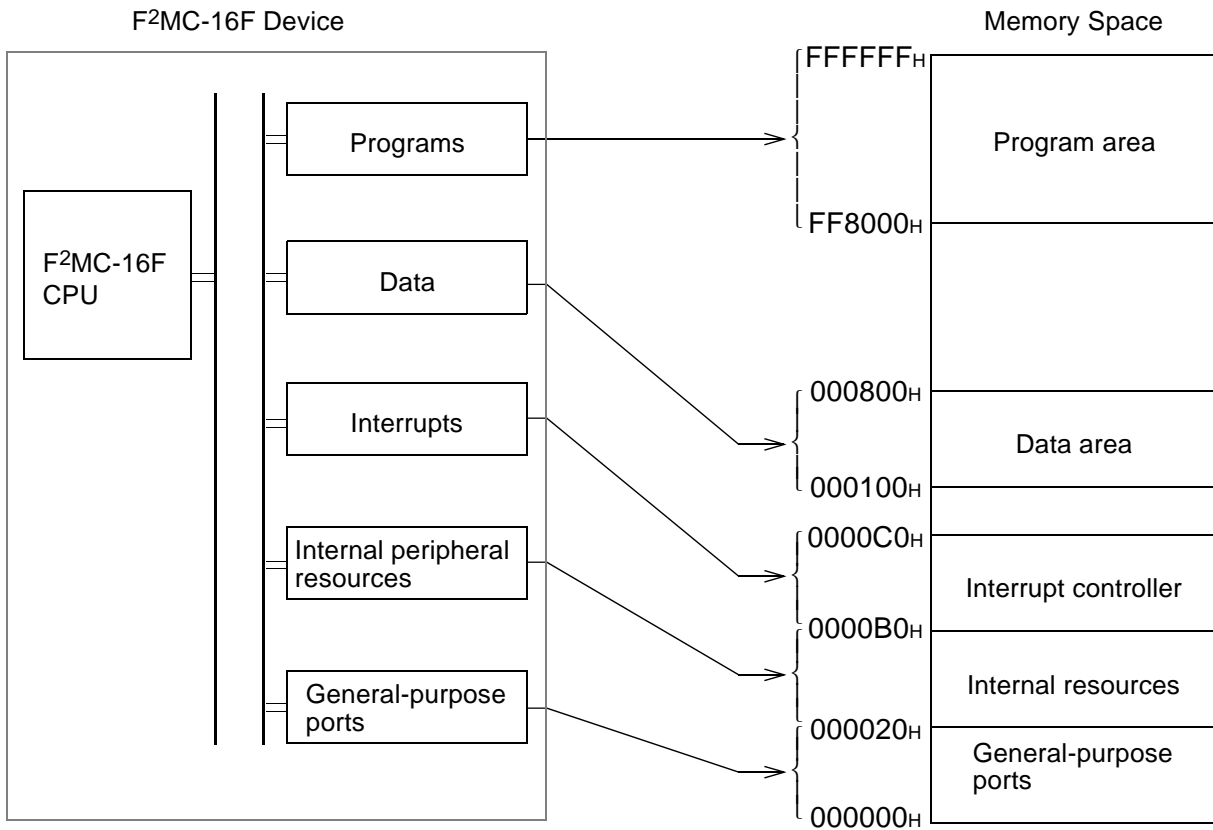
The principal features of the F<sup>2</sup>MC-16F CPU are:

- Minimum instruction execution time ..... 62.5 ns (at 32 MHz source oscillation)
- Memory space ..... 16MB: supports both linear and bank access
  - Instruction set optimized for controller applications
  - Wide variety of data types ..... bit / byte / word / long word
  - Expanded addressing mode..... 25 modes
  - High coding efficiency
  - 32-bit accumulator for higher computational accuracy (32-bit length)
  - Strengthened multiplication/division instructions (coded computation instructions added)
- Powerful interrupt functions
  - Priority levels ..... 8 levels (programmable)
- CPU-independent automatic transfer
  - Expanded intelligent I/O service
    - ..... Expanded and accelerated access area
    - ..... Maximum 15 channels
- Instruction set adapted for high level language (C) and multitasking
  - System stack pointer
  - Wide variety of pointers
  - High-symmetry instruction set
  - Barrel shift instructions
  - Stack check function
- Improved execution speed
  - ..... Instruction operations revised for higher speed
  - ..... 8-byte queuing

### 2.1.1 Memory Space

All data, programs and internal resources controlled by the F<sup>2</sup>MC-16F CPU are stored in the chip's memory area of 16 Mbytes. The CPU is able to access each address in memory as well as each internal resource through the 24-bit address bus (Figure 2.1.1).

For memory space allocation on the MB90242A series, see Section 2.2, "Map."



**Fig. 2.1.1 Memory Map in Relation to F<sup>2</sup>MC-16F Device**

■ Types of Addressing

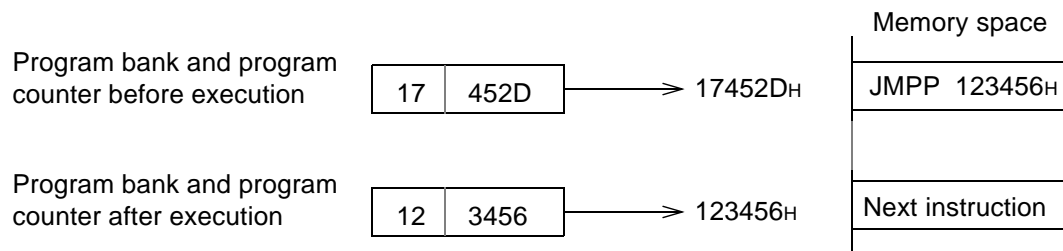
The F<sup>2</sup>MC-16F CPU uses two main methods to generate addresses. In linear addressing, a instruction designates one entire 24-bit address, and in bank addressing, the upper 8 bits of the address designate a bank register for a specific purpose, and the lower 16 bits are used as the addressing operand.

■ Linear Addressing

Two types of linear addressing are used, including direct designation of a 24-bit address as operand, as well as use of the lower 24 bits of a 32-bit general-purpose register or accumulator (Figure 2.1.2).

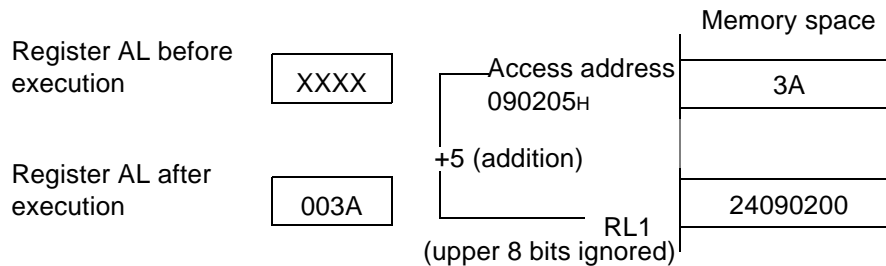
**Example 1.** Linear Addressing: Designation of 24-bit Operand

JMPP 123456H (Instruction branches to address designated by operand.)



**Example 2.** Linear Addressing: Indirect Designation Through 32-Bit Register

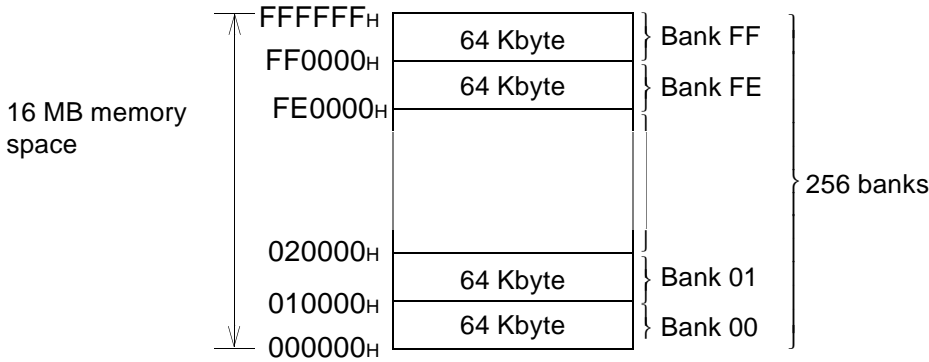
MOV A@RL1+5 (Instruction reads byte-length results of lower 24-bits from RL1 plus 8-bit offset, and stores results in AL.)



**Fig. 2.1.2 Sample Linear Address Designation**

■ Bank Addressing

In bank addressing the 16 Mbytes of memory space is divided into 256 banks of 64 Kbytes each, with bank addresses designated by bank registers. Figure 2.1.3 illustrates the division of the 16 MB memory space into 256 banks.



**Fig. 2.1.3 Schematic Representation of Banks**

There are five types of bank registers. Table 2.1.1 lists bank registers, the space accessed by each register, and principal uses.

**Table 2.1.1 Spaces Accessed by Bank Registers**

Bank register	Space name	Principal uses	Initial value at reset
Program bank register (PCB)	Program (PC) space	Storage of instruction codes, vector tables, and immediate data	FF <sub>H</sub>
Data bank register (DTB)	Data (DT) space	Storage of readable/writable data, access to internal and external peripheral resource control registers and data registers	00 <sub>H</sub>
User stack bank register (USB)	Stack (SP) register	Area used for stack access to registers for saving PUSH/POP instructions and interrupt instructions etc. SSB is used when CCR S=1, USB used when S=0.	00 <sub>H</sub>
System stack bank register (SSB)			00 <sub>H</sub>
Additional bank register (ADB)	Additional (AD) space	Storage of data overflow from data (DT) space	00 <sub>H</sub>

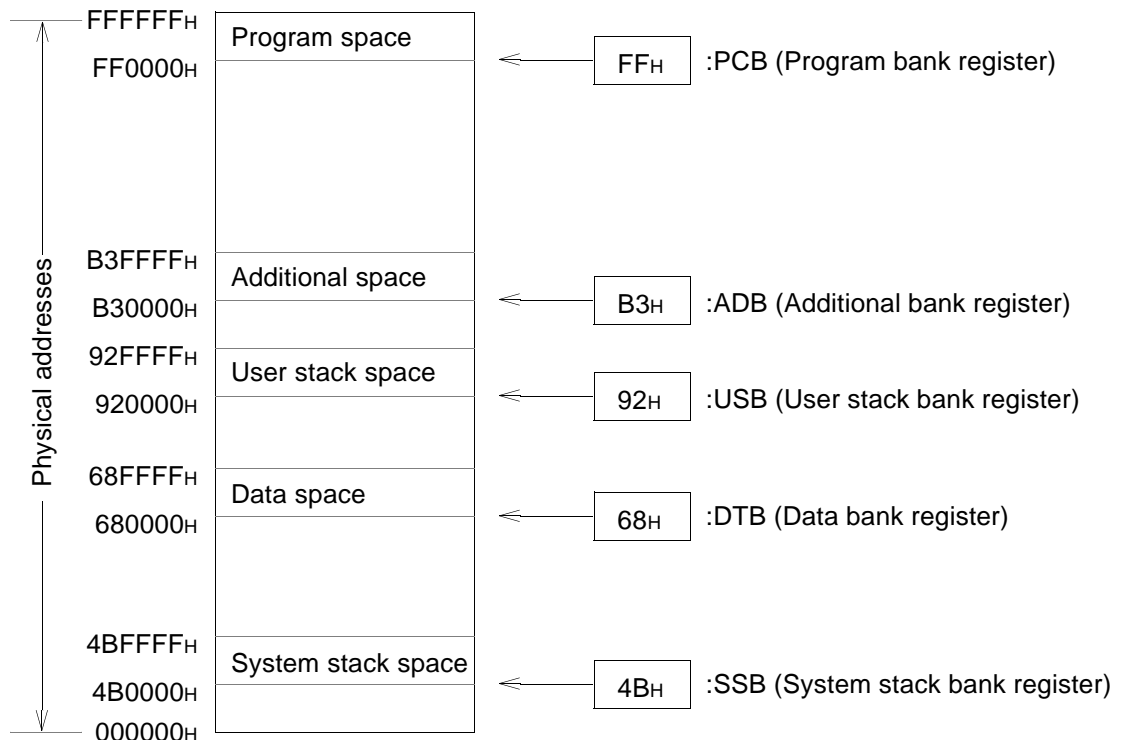
After reset, DT, SP and AD spaces are located in bank 00 (000000H - 00FFFFH), and PC space is located in bank FF (FF0000H - FFFFFFFH).

In order to improve coding efficiency, each instruction has defined default spaces for each type of addressing as shown in Table 2.1.2. To use a space other than its default space in any addressing mode, a prefix code corresponding to the desired bank is designated ahead of the instruction to allow access to the bank space designated by that prefix code.

**Table 2.1.2 Default Spaces**

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

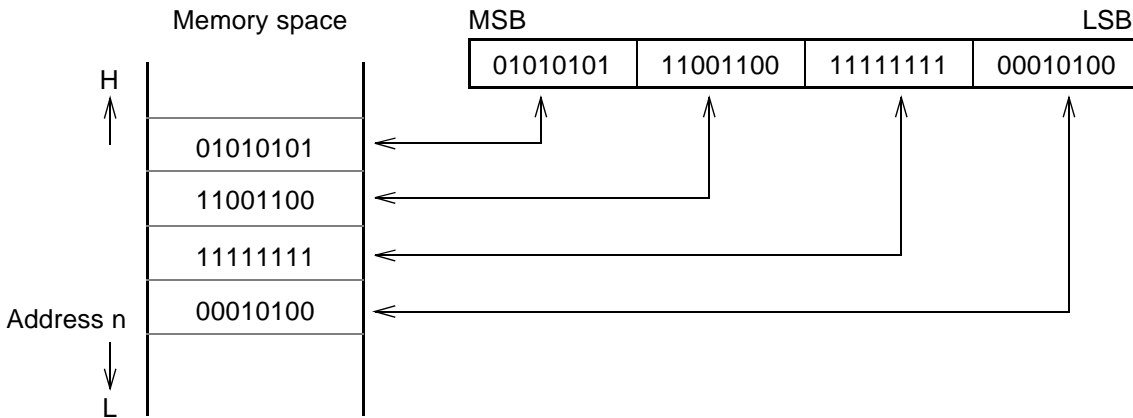
Figure 2.1.4 shows an example of memory space divided into banks, and the corresponding register bank settings.



**Fig. 2.1.4 Example: Bank Space Settings and Physical Addresses**

■ Memory Space Allocation with Multi-Byte Data Lengths

Figure 2.1.5 shows the configuration of data in memory when multi-byte data lengths are used. Data is positioned with the lowest 8 bits at address n, and each subsequent 8 bits at address n+1, n+2, n+3, ...



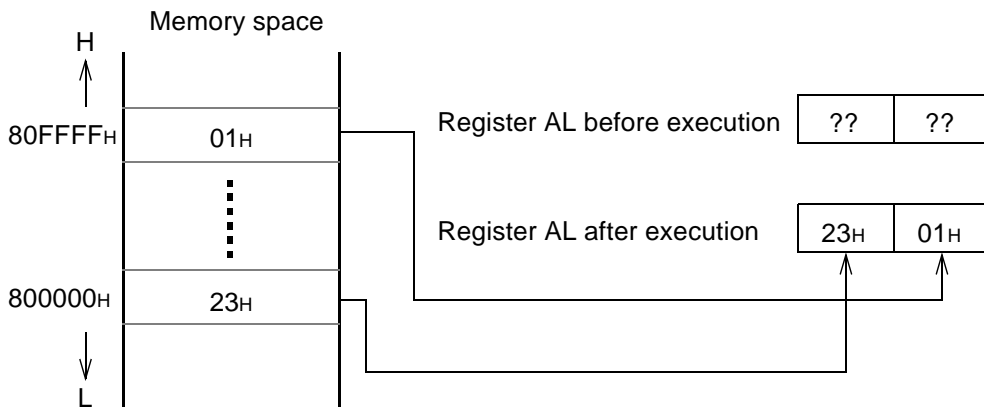
**Fig. 2.1.5 Example: Memory Space Allocation with Multi-Byte Data Lengths**

Writing to memory is executed starting with the lowest addresses first. For 32-bit data, the lower 16 bits are transferred first, followed by the upper 16 bits.

**[CAUTION]** If a reset signal is applied immediately after the lower 16 bits are written, the upper 16 bits may not be written to memory.

■ Access to Multi-Byte Data

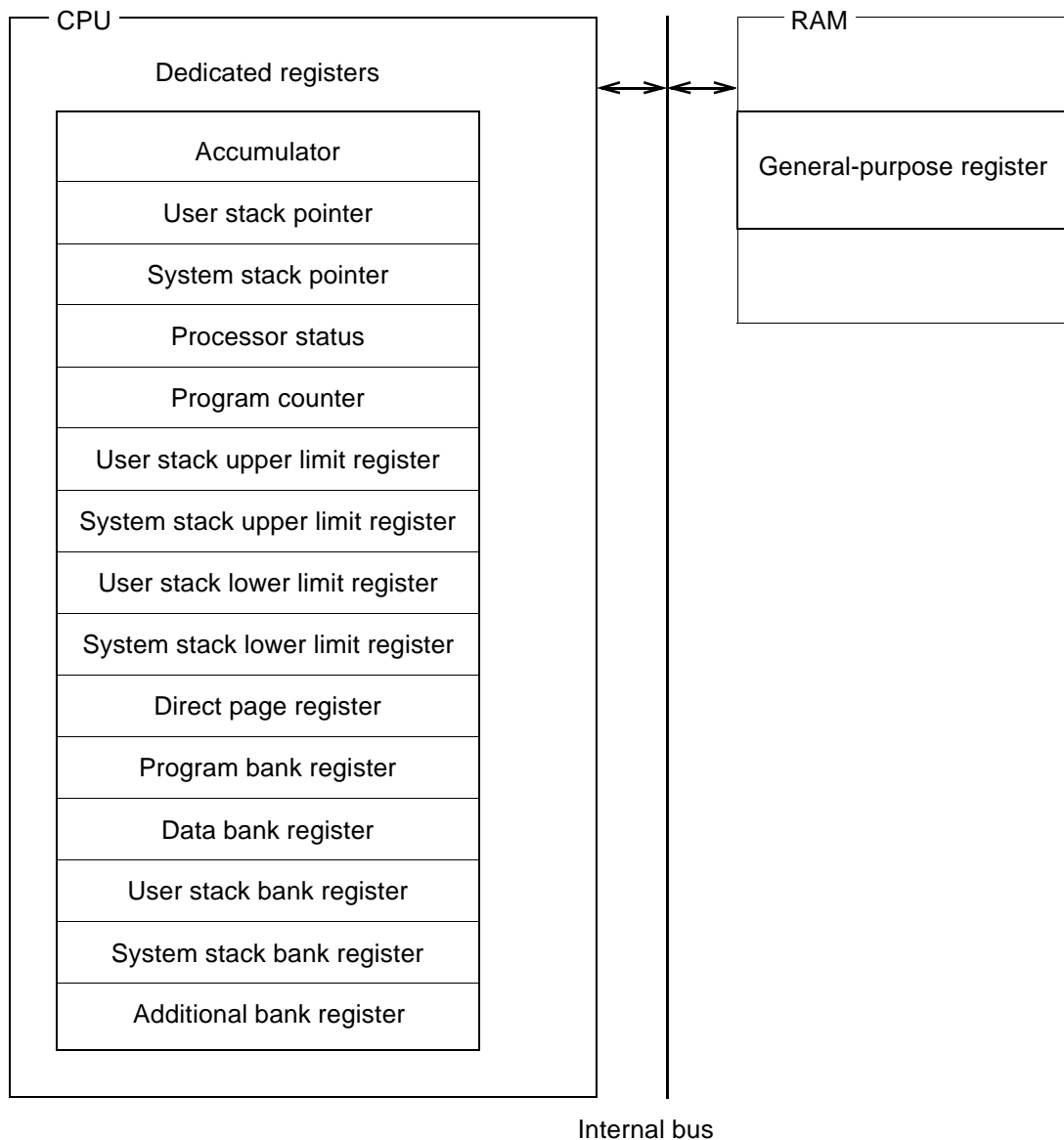
All accesses are based on bank units, so that for instructions accessing multi-byte data at address FFFF<sub>H</sub>, the next byte will be located within the same bank at address 0000<sub>H</sub>. Figure 2.1.6 shows an example of instruction of access to multi-byte data.



**Fig. 2.1.6 Execution of Instruction MOV PW A, 080FFF<sub>H</sub>**

## 2.1.2 Registers

F<sup>2</sup>MC-16F registers are broadly divided into internal dedicated registers on CPU and general-purpose registers in internal RAM. The dedicated registers are dedicated hardware located inside the CPU, and their use is limited by the architecture of the CPU itself. In contrast, the general-purpose registers coexist within RAM in CPU address space, and are similar to the dedicated registers in that they may be accessed without designation of addresses, but differ in that they may be used for user-defined purposes in the same way as ordinary memory. Figure 2.1.7 shows the arrangement of dedicated registers and general-purpose registers within the F<sup>2</sup>MC-16F device.



**Fig. 2.1.7 Dedicated Registers and General-Purpose Registers**



■ Dedicated Registers

Figure 2.1.8 lists the 15 dedicated registers in the F<sup>2</sup>MC-16F CPU.

Configuration	Register name	Function
AH AL	Accumulator	Two 16-bit registers used to store results of calculation, etc. May be concatenated for use as a 32-bit register.
USP	User stack pointer	16-bit pointer indicating the user stack area
SSP	System stack pointer	16-bit pointer indicating the system stack area
PS	Processor status	16-bit register indicating system status
PC	Program counter	16-bit register with the address where the program is stored
USPCU	User stack upper limit register	16-bit register designating the upper limit of the user stack
SSPCU	System stack upper limit register	16-bit register designating the upper limit of the system stack
USPCL	User stack lower limit register	16-bit register designating the lower limit of the user stack
SSPCL	System stack lower limit register	16-bit register designating the lower limit of the system stack
DPR	Direct page register	8-bit register indicating the direct page
PCB	Program bank register	8-bit register indicating program space
DTB	Data bank register	8-bit register indicating the data space
USB	User stack bank register	8-bit register indicating the user stack space
SSB	System stack bank register	8-bit register indicating system stack space
ADB	Additional bank register	8-bit register indicating additional space

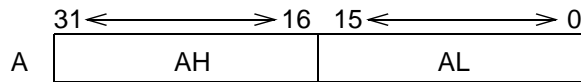
**Fig. 2.1.8 Dedicated Registers**

■ Accumulator (A)

The accumulator (A) consists of two 16-bit registers for arithmetic operation, labeled AH and AL, used to hold the results of calculations, and also used as temporary memory for data transfers. Figure 2.1.9 shows the configuration of the accumulator. The AH and AL registers can be linked for 32-bit data processing, and only the AL register is used for processing of data in 16-bit word units or 8-bit byte units.

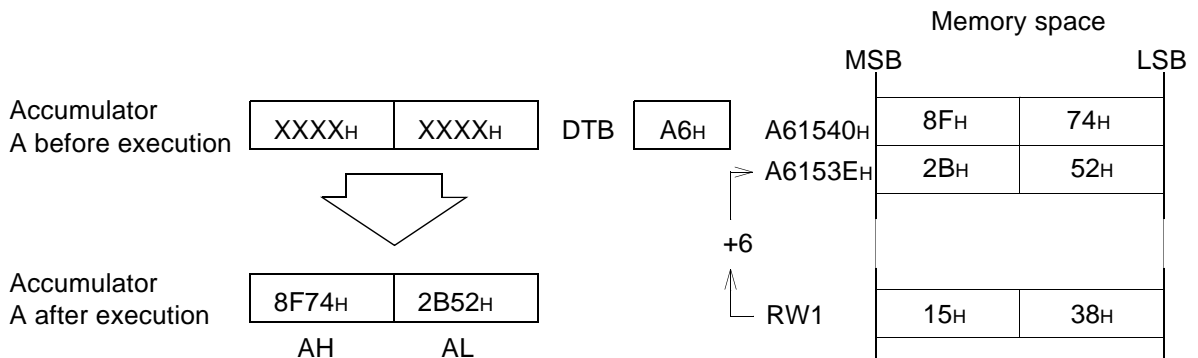
Data stored in the accumulator may be used for calculations involving data in memory/registers (Ri, RWi, RLi). As with the F<sup>2</sup>MC-8/16/16H CPU, the F<sup>2</sup>MC-16F is designed so that when data of word length or less is transferred to the AL register, the previous contents (immediately before transfer) of the AL register are automatically transferred to the AH register. This is called the data preservation function, and increases efficiency by enabling various types of processing using calculations from the contents of the AL and AH registers.

Figure 2.1.10 shows an example of 32-bit data transfer, and Figure 2.1.11 shows an example of data transfer between the AL and AH registers.



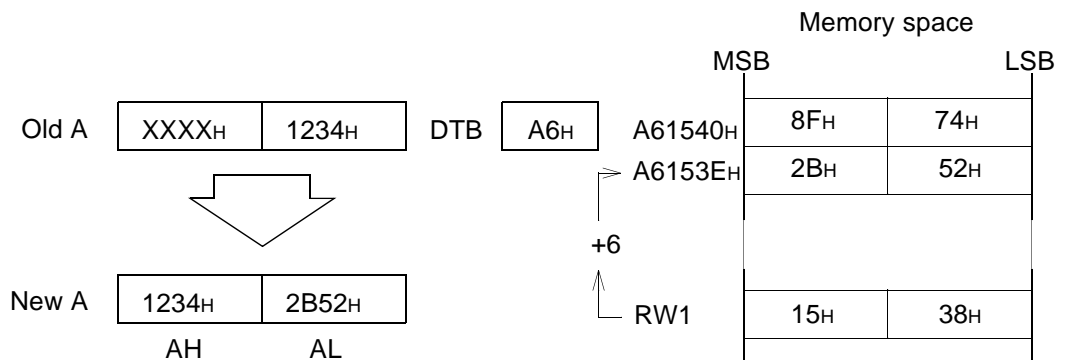
**Fig. 2.1.9 Accumulator (A) Configuration**

MOVL A, @RW1+6 (Instruction: Read in long word format the contents of RW1 + 8-bit offset as an address, and store the resulting contents in A.)



**Fig. 2.1.10 Example of 32-Bit Data Transfer**

MOVW A, @RW1+6 (Instruction: Read in word format the contents of RW1 + 8-bit offset, and store the resulting contents in A.)

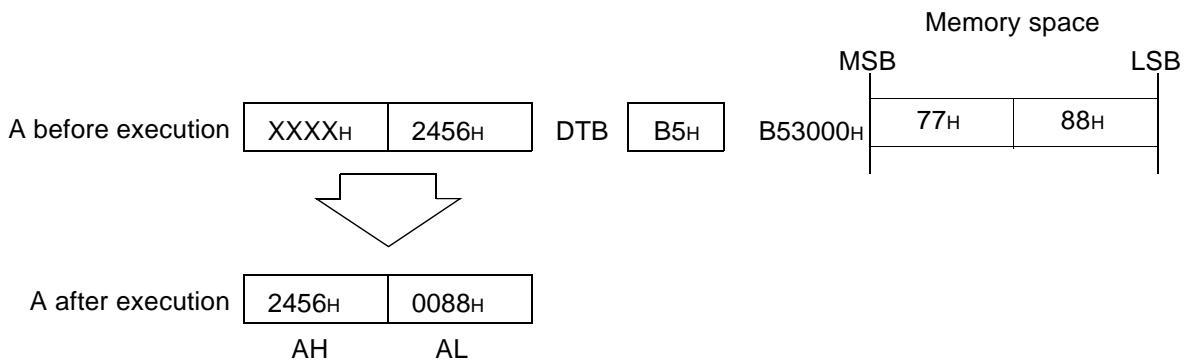


**Fig. 2.1.11 Example of AL - AH Transfer**

## 2.1 CPU

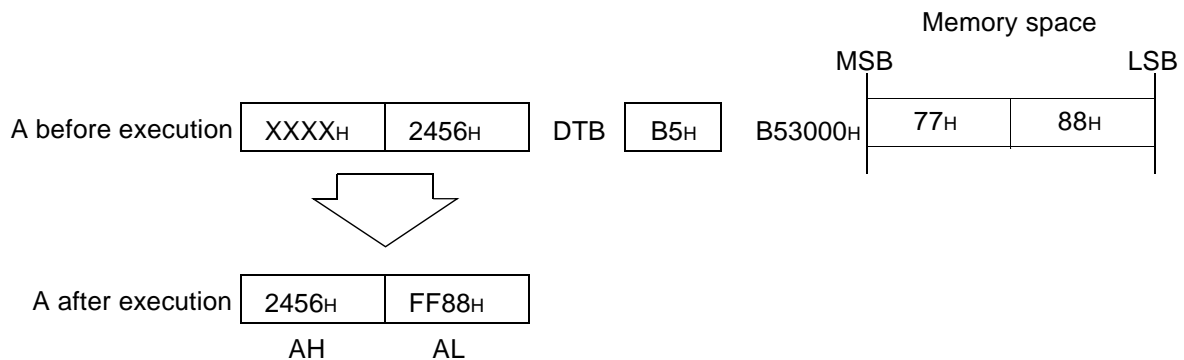
When transferring byte-length or shorter data to register AL, the data is given a coded extension or zero extension and stored in AL in 16-bit length. Also, data in the AL register can be handled in either word length or byte length. When an arithmetic calculation instruction is performed on the contents of AL in byte processing, the upper 8 bits of the value of AL before processing are ignored and the upper 8 bits in the result will all be set to zero.

**MOV A, 3000H** (Instruction: Extend the contents of address 3000H with zeros, and store the results in AL.)



**Fig. 2.1.12 Example of Zero Extension**

**MOVX A, 3000H** (Instruction: Extend the contents of address 3000H with coding, and store the results in AL.)



**Fig. 2.1.13 Example of Coded Extension**

■ User Stack Pointer (USP) and System Stack Pointer (SSP)

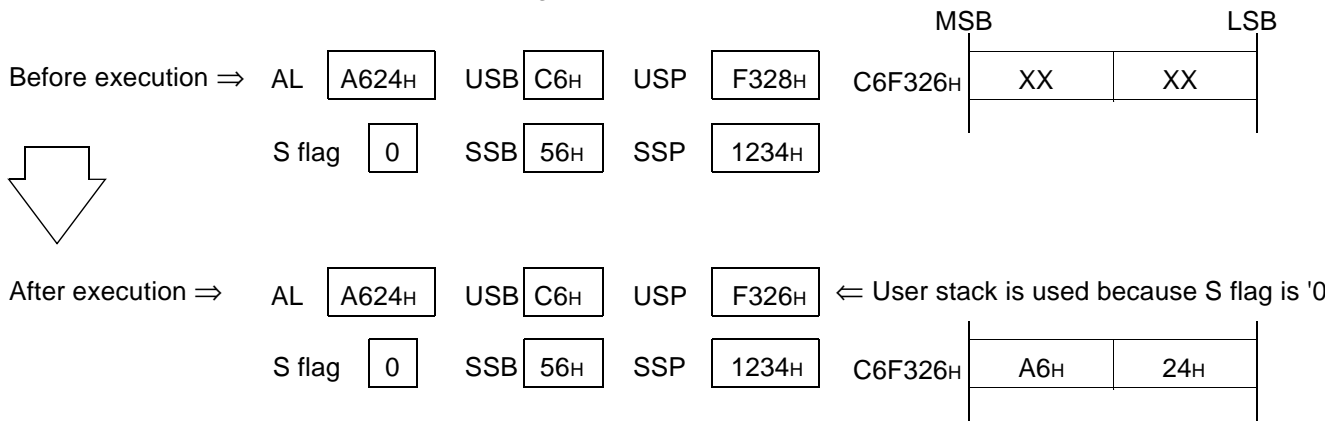
The USP and SSP are 16-bit registers, indicating addresses in memory where data is saved or restored during execution of PUSH/POP instructions and subroutines. The USP and SSP registers are treated in the same way by stack instructions, but the USP register is enabled when the S flag in the processor status register (PS) is set to '0' and the SSP register is enabled when the S flag is set to '1' (see Figure 2.1.14).

Because the S flag is set to '1' when an interrupt is received, saving into a register due to interrupts must be handled in memory areas indicated by the SSP register. Normally stack processing for interrupt routines uses the SSP, and stack processing other than for interrupt routines uses the USP register. When there is no need to divide stack space, the SSP register alone should be used.

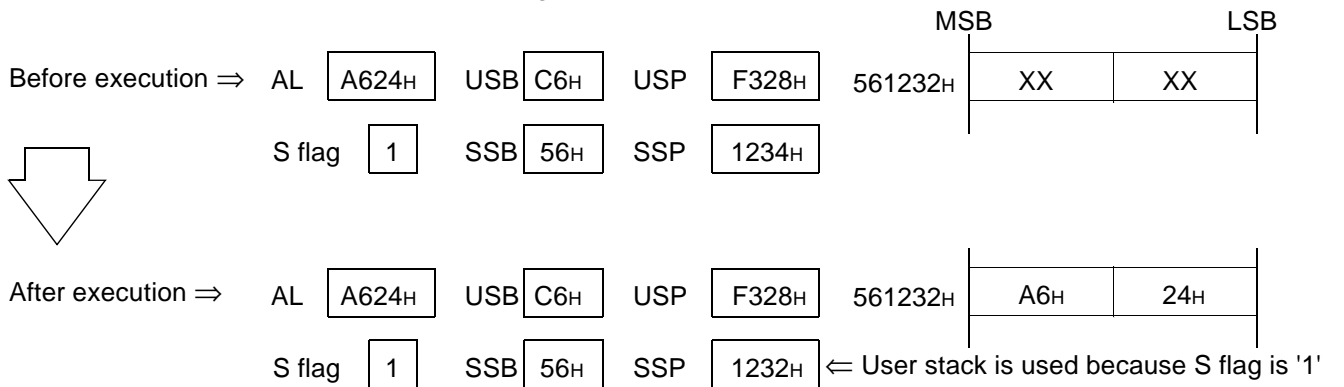
If values set in the stack pointers designate odd-numbered addresses, word access will be broken into two parts, reducing efficiency. Therefore the use of even-numbered addresses is encouraged.

The upper 8 bits of addresses used in stack pointers are stored in the SSB byte in the SSP register, and in the USB byte in the USP register.

**Example 1.** PUSHW A, with S flag set to '0'



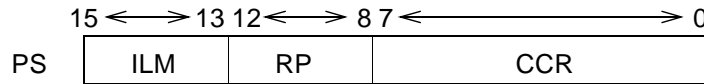
**Example 2.** PUSHW A, with S flag set to '1'



**Fig. 2.1.14 Stack Operation Instructions and Stack Pointers**

■ Processor Status Register (PS)

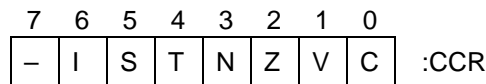
The PS register is configured from bits that perform CPU operating controls and bits that indicate CPU status. As shown in Figure 2.1.15, The upper byte of the PS register is composed of the register bank pointer (RP) which indicates the starting address of the register bank, and the interrupt level mask register (ILM), and the lower byte of PS consists of the condition code register (CCR) which contains flags set to '1' or '0' depending on results of instruction execution and interrupt generation.



**Fig. 2.1.15 PS Register Configuration**

**(1) Condition Code Register (CCR)**

Figure 2.1.16 shows the configuration of the condition code register, and Table 2.1.3 describes its functions.



**Fig. 2.1.16 Condition Code Register Configuration**

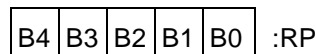
**Table 2.1.3 Flag Functions**

Flag name	Function
I   Interrupt enable flag	The I flag is set to '1' to enable all interrupt requests other than software interrupts. When the flag is '0' interrupts are masked. The reset value is '0.'
S   Stack flag	The S flag is set to '0' to select the USP register as the pointer used for stack operations. When the flag is '1' the SSP register is selected. Following an interrupt or reset, the value is reset to '1.'
T   Sticky bit flag	The T flag is set to '1' when one or more "1s" are contained in the data shifted out from the carry field during execution of logical right-shift or arithmetic right-shift instructions. The value is '0' at all other times. When the shift value is zero places, the bit is set to '0.'
N   Negative flag	This flag is set to '1' if the MSB of the results of arithmetic calculation is '1' and '0' when the MSB is '0.'
Z   Zero flag	The Z flag is set to '1' when the results of arithmetic calculation are all zeros, and is set to '0' at all other times.
V   Overflow flag	The V flag is set to '1' when execution of an arithmetic calculation results in a coded value indicating an overflow, and is set to '0' at all other times.
C   Carry flag	The C flag is set to '1' when an arithmetic calculation requires the MSB to be carried up or down one or more places, and is set to '0' at all other times.

**(2) Register Bank Pointer (RP)**

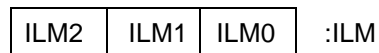
The RP register indicates the relationship between the general purpose registers of the F<sup>2</sup>MC-16F core and internal RAM. The first memory address of the register bank currently in use is indicated using the conversion formula  $[000180_{\text{H}} + (\text{RP}) * 10_{\text{H}}]$ . The RP register has 5-bit configuration and can take values from 00h to 1Fh corresponding to register banks having memory addresses 000180h to 00037Fh. For a description of the relation between the RP register and internal RAM, as well as the configuration of register banks, see the description of general-purpose registers.

Instruction execution can transfer 8-bit immediate data values to the RP register, but only the lower 5 bits can be actually used. Following a reset, this register is initialized to the value 00h.

**Fig. 2.1.17 Register Bank Pointer****(3) Interrupt Level Mask Register (ILM)**

The ILM register has 3-bit configuration, and indicates the level of CPU interrupt masking. To be received by the CPU, interrupt requests must have a stronger (higher) level than the setting of these three bits. The strongest (highest priority) level is 0 and the weakest (lowest priority) level is 7 (see Table 2.1.4). Thus for an interrupt to be received, it must be requested at a level with a smaller value than the current ILM register value. When an interrupt is received, the value of its level is stored in the ILM register, and no interrupts of equal or lower priority will then be received.

Instruction execution can transfer 8-bit immediate data values to the ILM register, but only the lower three bits can be actually used. Following a reset, this register is initialized to the value '0.'

**Fig. 2.1.18 Interrupt Level Register****Table 2.1.4 Relative Strength (Priority) of Levels in the Interrupt Level Mask Register (ILM)**

ILM2	ILM1	ILM0	Level value	Interrupt level enabled
0	0	0	0	Interrupt disabled
0	0	1	1	Level 0 only
0	1	0	2	Level 2 or stronger
0	1	1	3	Level 3 or stronger
1	0	0	4	Level 4 or stronger
1	0	1	5	Level 5 or stronger
1	1	0	6	Level 6 or stronger
1	1	1	7	Level 7 or stronger

■ Program Counter (PC)

The PC register is a 16-bit counter, indicating the lower 16 bits of the memory address containing the instruction code to be executed by the CPU. The upper 8 bits of the address are indicated by the PCB register. The value in the PC register is updated by branching instructions, subroutine call instructions, interrupts and resets. It can also be used as a base pointer for operand access.



**Fig. 2.1.19 Program Counter**

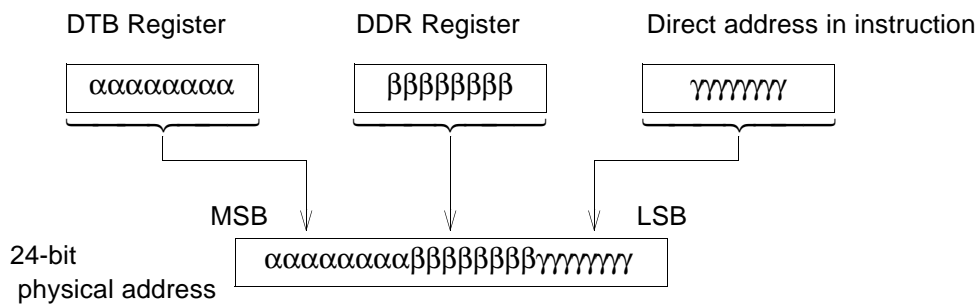
■ User Stack Upper Limit Register (USPCU), System Stack Upper Limit Register (SSPCU), User Stack Lower Limit Register (USPCL), System Stack Lower Limit Register (SSPCL)

These registers are used to indicate when the value in the current stack pointer is outside a given range. Separate upper and lower limits are used for the user stack and system stack, and settings are made using the SPCU and SPCL registers for the current stack as indicated by the S flag in the CCR register. Whenever a subroutine call instruction, interrupt, reset instruction or other procedure causes the top address of the stack to fall outside the range defined by these registers, stack area error exception processing is generated. These values are not initialized by a reset instruction.

For information on the stack check function, see 'Exception Processing.'

■ Direct Page Register (DPR)

The DPR register indicates the values addr8 through addr15 of the operand in direct addressing instructions, as shown in Figure 2.1.20. The DPR register has 8-bit length, and is initialized to '01H' following a reset. Both read and write accesses are enabled.



**Fig. 2.1.20 Generation of Physical Addresses by Direct Addressing**

- Program Counter Bank Register (PCB),  
Data Bank Register (DTB),  
User Stack Bank Register (USB),  
System Stack Bank Register (SSB),  
Additional Bank Register (ADB)

This set of bank registers is used to indicate the respective memory banks allocated in PC space, DT space, SP space (user), SP space (system) and AD space. All these bank registers have 8-bit length. Following a reset, the PCB register is initialized to 'FFH' and all others to '00H.' All except the PCB register are read/write enabled. The PCB register is enabled for read access only. The PCB register is overwritten by processing of software interrupt instructions branching to full 16 MB space including JMPP, CALLP, RETP, RETI, RETIQ, as well as by hardware interrupts and exception processing. The USB and SSB stack pointers are collectively designated in instructions by the term SPB. When the S flag in the CCR register is '0' this term selects the USB pointer, and when the S flag is '1' the SSB pointer is selected.

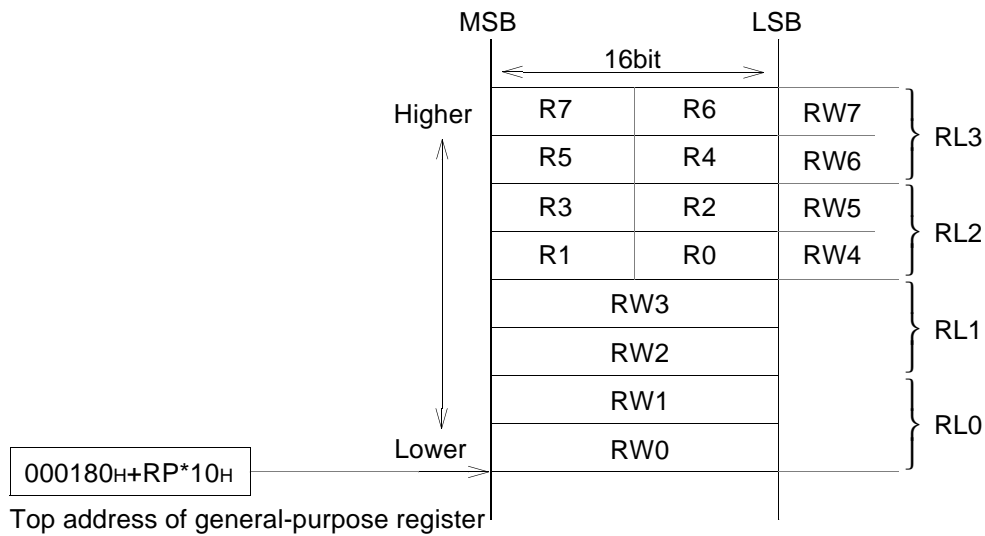
For the operation of each of these registers, see section 2.1.1 "Memory Space."



■ General-Purpose Registers

The F<sup>2</sup>MC-16F CPU core has general-purpose registers located at RAM addresses 000180H-00037H on the memory map. The register bank pointer (RP) is used to indicate which register bank is the currently active area of memory. Each bank contains the following three types of registers. These registers are not independent, and are related as shown in Figure 2.1.21.

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



**Fig. 2.1.21 General-Purpose Registers**

■ Register Banks

Register banks have 16-bit x 8 (16-byte) configuration, and can be used as byte registers R0 to R7, word registers RW0 to RW7, or long-word registers RL0 to RL3 for various calculations. Register banks can also be used as pointers in various instructions, and the RL0 to RL3 registers can be used as linear pointers for direct access to full memory space. Table 2.1.5 lists the functions of each register.

**Table 2.1.5 Register Function**

R0 to R7	Used as operands for various instructions. Note R0 can also be used as a barrel-shift counter or a counter for normalize instructions.
RW0 to RW7	Used as pointers, and as operands for various instructions. Note RW0 can also be used as a counter for string instructions.
RL0 to RL3	Used as long pointers, and as operands for various instructions.

### 2.1.3 Prefix Codes

Prefix codes may be placed before instructions to partially alter the operation of the instruction. There are three types of prefixes: bank select prefixes, common register bank prefixes and flag change suppression prefixes.

#### ■ Bank Select Prefixes

The area of memory space used in a data access operation is determined by addressing. By placing a bank select prefix before an instruction, the area of memory space accessed in that instruction can be specified without regard to the addressing mode. Table 2.1.6 lists bank select prefixes in relation to the areas of memory space selected.

**Table 2.1.6 Bank Select Prefixes**

Bank select prefix	Area selected
PCB	Program space
DTB	Data space
ADB	Additional space
SPB	User stack space when the S flag in the CCR register is '0,' and system stack space when the S flag is '1.'

Note that the instructions listed in Table 2.1.7 ignore bank select prefixes. Also, the effect of bank select prefixes used with instructions listed in Table 2.1.8 is retained until the following instruction.

**Table 2.1.7 Instructions Unaffected by Bank Select Prefixes**

Instruction type	Instruction		Effect of bank select prefix
String instructions	MOVS SCEQ FILS	MOVSW SCWEQ FILSW	Instruction uses bank register designated by operand, regardless of prefix.
Stack operation instructions	PUSHW	POPW	Instruction uses USB if S flag is '0' and SSB if S flag is '1', regardless of prefix.
I/O access instructions	MOV A,io MOVW A,io MOV io,A MOV io,#imm8 MOVB A,io:bp SETB io:bp BBC io:bp,rel WBTC io:bp	MOVX A,io MOVW io,A MOVW io,#imm16 MOVB io:bp,A CLRB io:bp BBS io:bp,rel WBTS io:bp	Space 000000H-0000FFH is accessed, regardless of prefix.
Stack pointer indirect addressing instructions	MOV A,@SP+disp8 MOVW A,@SP+disp8 MOV @SP+disp8,A MOVL @SP+disp8,A	MOVX A,@SP+disp8 MOVL A,@SP+disp8 MOVW @SP+disp8,A	Instruction uses USB if S flag is '0' and SSB if S flag is '1', regardless of prefix.

**Table 2.1.7 Instructions Unaffected by Bank Select Prefixes (Continued)**

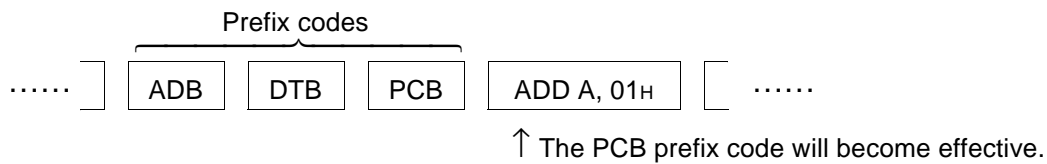
Instruction type	Instruction	Effect of bank select prefix
Long register indirect addressing instructions with displacement	MOV A,@RLi+disp8    MOVX A,@RLi+disp8 MOVW A,@RLi+disp8 MOV @RLi+disp8,A    MOVW @RLi+disp8,A	Instruction accesses space designated by (RLi+disp8) value, regardless of prefix.
Addressing instructions designating 24-bit direct addresses	MOVP A,addr24    MOVXP A,addr24 MOVWP A,addr24    MOVPL A,addr24 MOVP addr24,A    MOVWP addr24,A MOVPL addr24,A	Instruction accesses space designated by addr24 value, regardless of prefix.
Addressing instructions designating accumulator indirect 24-bit addresses	MOVP A,@A    MOVXP A,@A MOVWP A,@A    MOVPL A,@A MOVP @A,Ri    MOVWP @A,RWi MOVPL @A,RLi	Instruction accesses space designated by value of 'A', regardless of prefix.
Interrupt recovery instructions	RETI    RETIQ	Instruction uses SSB value, regardless of prefix.
Multiple data transfer instructions	MOVM    MOVMW	Instruction uses bank registers as determined by the instruction, regardless of prefix.

**Table 2.1.8 Instructions Retaining Bank Select Prefix Effect Until Next Instruction**

Instruction type	Instruction
Flag change instructions	AND CCR,#imm8    OR CCR,#imm8
PS recovery instructions	POPW PS
ILM set instructions	MOV ILM,#imm8

■ Continuous Bank Select Prefix Codes

If bank select prefix codes are continuous, only the last code will become effective.



**Fig. 2.1.22 Continuous Bank Select Prefix Codes**

■ Common Register Bank Prefix (CMR)

To facilitate exchange of data among multiple tasks, it is necessary to have some relatively simple means of accessing the same register bank regardless of the value of the RP at the particular moment. The CMR prefix can be placed before a instruction accessing a general-purpose register, to change all register access for that instruction to common banks in the range 000180H-00018FH (the register bank selected when RP=0).

Note that caution is required when using this prefix with the instructions listed in Table 2.1.9.

**Table 2.1.9 Instructions Requiring Caution When Used with the Common Register Bank Prefix**

Instruction type	Instruction		Effect of bank select prefix
String instructions	MOVS SCEQ FILS	MOVSW SCWEQ FILSW	Do not use the CMR prefix with string instructions.
Flag change instructions	AND CCR,#imm8	OR CCR,#imm8	The effect of the prefix will be retained for the next instruction.
PS recovery instructions	POPW	PS	The effect of the prefix will be retained for the next instruction.
ILM setting instructions	MOV	ILM,#imm8	The effect of the prefix will be retained for the next instruction.

■ Flag Change Suppression Prefix

The flag suppression (or no-change) code (NCC) is used to suppress unwanted flag changes. The NCC is placed before the instruction in which flag changes are to be suppressed, and will suppress all flag changes resulting from that instruction. This prefix affects the T, N, Z, V and C flags.

Note that caution is required when using this prefix with the instructions shown in Table 2.1.10.

**Table 2.1.10 Instructions Requiring Caution When Used with Flag Change Suppression Prefix**

Instruction type	Instruction		Effect of bank select prefix
String instructions	MOVS SCEQ FILS	MOVSW SCWEQ FILSW	Do not use the NCC prefix with string instructions.
Flag change instructions	AND CCR,#imm8	OR CCR,#imm8	The CCR will change according to instruction specifications whether the prefix is used or not. The effect of the prefix will be retained for the next instruction.
PS recovery instructions	POPW	PS	The CCR will change according to instruction specifications whether the prefix is used or not. The effect of the prefix will be retained for the next instruction.
ILM setting instructions	MOV	ILM,#imm8	The effect of the prefix will be retained for the next instruction.

Interrupt instructions, interrupt recovery instructions	INT #vct8 INT addr16 RETI	INT9 INTP addr24 RETIQ	The CCR will change according to instruction specifications whether the prefix is used or not.
Context instructions	JCTX @A		The CCR will change according to instruction specifications whether the prefix is used or not.

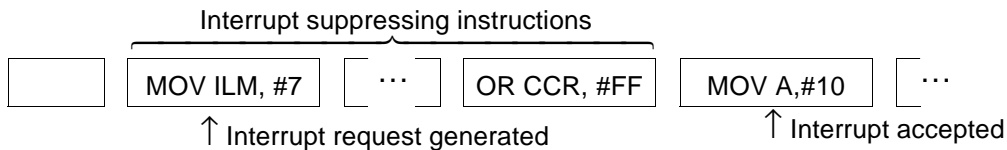
■ About Interrupt Suppressing Instructions

Table 2.1.11 lists ten types of instructions with which generation of hardware interrupt requests is not detected, and interrupt requests are ignored.

**Table 2.1.11 Hardware Interrupt Suppressing Instructions**

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

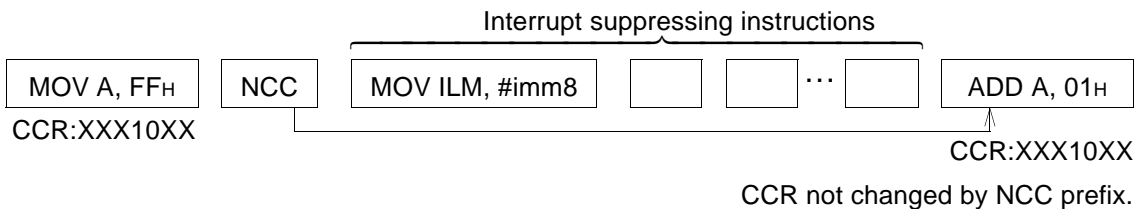
As Figure 2.1.23 shows, when a valid hardware interrupt is generated during execution of one or more of these instructions, interrupt processing will be delayed until after the execution of a instruction that is not one of the above types.



**Fig. 2.1.23 Interrupt Handling during Interrupt Suppressing Instructions**

■ Restrictions on Interrupt Suppressing Instructions Used with Prefix Codes

When a prefix code is placed before an interrupt suppressing instruction, the effect of the prefix code is extended to the first instruction that is not one of the interrupt suppressing instructions. Figure 2.1.24 shows an example.



**Fig. 2.1.24 Interrupt Suppressing Instructions Used with Prefix Codes**

## 2.1.4 Interrupts, Expanded Intelligent I/O Services, and Exceptions

The F<sup>2</sup>MC-16F CPU has four types of functions capable of reacting to the occurrence of a particular event, interrupting the execution of the instruction that is currently being processed, and transferring control to a separately defined program.

- Hardware interrupts ..... Interrupt processing triggered by events occurring in internal resources.
- Software interrupts ..... Interrupt processing triggered by software instructions that generate specific events.
- Expanded intelligent I/O services (EI<sup>2</sup>OS) ..... Transfer processing triggered by events occurring in internal resources.
- Exceptions ..... Processing interruptions triggered by the occurrence of abnormal circumstances.

### ■ Hardware Interrupts

#### (1) Overview

In a hardware interrupt, the CPU reacts to an interrupt request signal from its internal resources circuit, temporarily suspends the execution of the program that it has been executing, and transfers control to an interrupt processing program defined by the user.

Hardware interrupts are initiated when the level of the interrupt request is compared with the interrupt level mask (ILM) register in the CPU processor status (PS) register, and the contents of the I flag in the PS register are referenced by hardware, in order to determine that interrupt conditions exist. When a hardware interrupt is generated, the CPU performs interrupt processing as follows.

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

#### (2) Configuration

Three areas of the F<sup>2</sup>MC-16F core are involved in hardware interrupt processing.

- Internal resources ... The interrupt enable bit and interrupt request bit are used as reference to control interrupt requests from internal resources.
- Interrupt controller ... The ICR register assigns interrupt priority levels and determines priority of interrupts occurring at the same time.
- CPU ... The I and ILM registers compare the level of the interrupt request with the level of the existing requests and identify an interrupt enable status.

The microcode function executes the necessary steps in interrupt processing.

Each of these functions is realized through register settings -- the internal resource control registers for the internal resources, the ICR register for the interrupt controller, and the CCR register for the CPU. Before a hardware interrupt can be used, therefore, settings must be made to these three locations. For information about the ICR register, see 'Interrupt Control Register (ICR)' in the section "Expanded Intelligent I/O Services."

The interrupt vector tables referred to during interrupt processing are located in memory area FFFC00H to FFFFFFFH, and the same tables are used for software interrupts. Table 2.1.12 lists interrupt numbers and interrupt vectors assigned.

**Table 2.1.12 Interrupt Numbers and Interrupt Vectors Assigned**

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 0	FFFFFCH	FFFFFDH	FFFFFEH	Not used	#0	None
⋮	⋮	⋮	⋮	⋮	⋮	⋮
INT 7	FFFFE0H	FFFFE1H	FFFFE2H	Not used	#7	None
INT 8	FFFFDCH	FFFFDDH	FFFFDEH	FFFFDF	#8	(RESET vector)
INT 9	FFFFD8H	FFFFD9H	FFFFDAH	Not used	#9	None
INT 10	FFFFD4H	FFFFD5H	FFFFD6H	Not used	#10	<Exception>
INT 11	FFFFD0H	FFFFD1H	FFFFD2H	Not used	#11	Hardware interrupt #0
INT 12	FFFFCCH	FFFFCDH	FFFFCEH	Not used	#12	Hardware interrupt #1
INT 13	FFFFC8H	FFFFC9H	FFFFCAH	Not used	#13	Hardware interrupt #2
INT 14	FFFFC4H	FFFFC5H	FFFFC6H	Not used	#14	Hardware interrupt #3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
INT 254	FFFC04H	FFFC05H	FFFC06H	Not used	#254	Open
INT 255	FFFC00H	FFFC01H	FFFC02H	Not used	#255	<Stack fault>

**(3) Operation**

Each internal resource with a hardware interrupt function has both an 'interrupt request flag' that indicates whether an interrupt request has been made or not, and an 'interrupt enable flag' used to select whether that circuit will send its interrupt signal to the CPU or not. Each interrupt request flag is set by the occurrence of a particular event within that internal resource, and if the interrupt enable flag has an 'enable' setting, the resulting interrupt request will then be output from the internal resource to the interrupt controller.

The interrupt controller compares individual interrupts simultaneously received with the interrupt levels (IL) in the interrupt control register (ICR), selects the highest-level interrupt (the one with the lowest IL value) and notifies the CPU. If more than one interrupt with the same level is received, the lowest interrupt number is given priority. For the relation between interrupt requests and ICR values, see section 2.2.3 "Interrupt Level Assignments."

The CPU receives the interrupt, compares its level with the ILM field in the processor status (PS) register, and if the value of the interrupt level is less than the ILM setting and the I flag in the PS register has the value '1,' microcoding for interrupt processing will begin as soon as the currently executing instruction is ended.

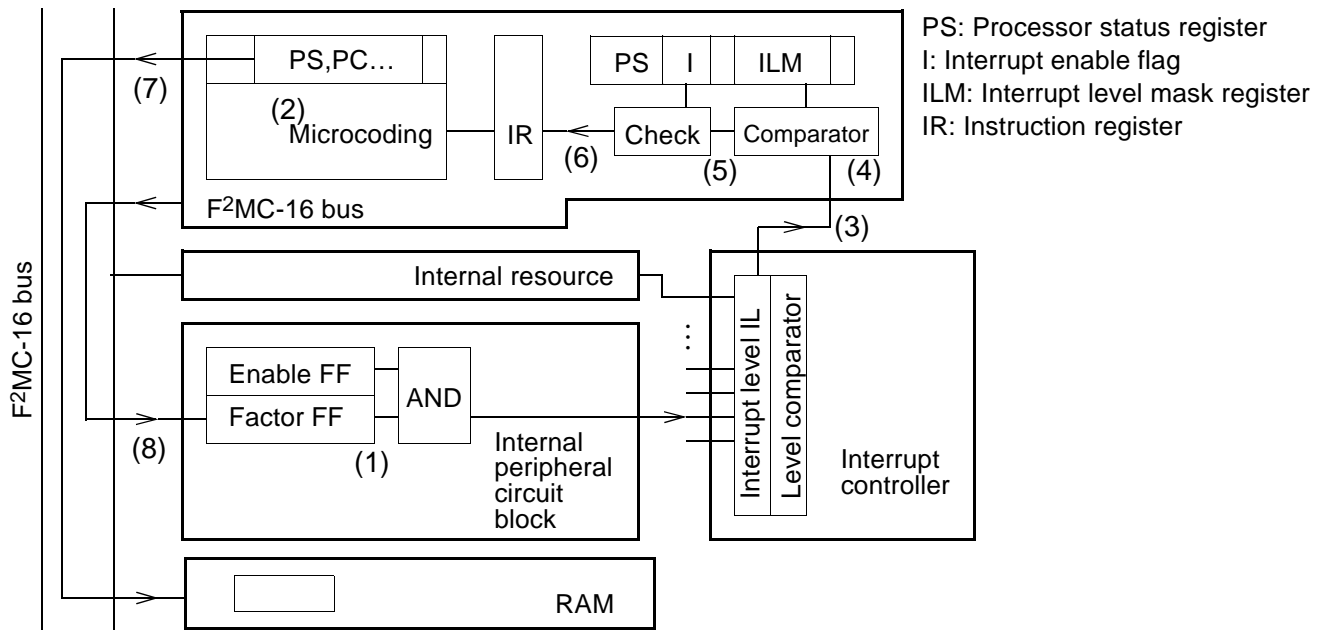
The top of the interrupt processing microcode references the ISE bit in the interrupt controller's ICR register, verifies that the value of that bit is '0' (0=interrupt), and then starts the main sequence of the interrupt processing routine.

In interrupt processing, the 12 bytes in the A, DPR, ADB, DTB, PCB, PC and PS registers are saved to the area of memory designated by the SSB and SSP registers, the contents of the 3-byte interrupt vector is read and loaded into the PC and PCB register, the contents of the ILM field in the PS register are

updated to the level of the currently accepted interrupt request, the S flag is set to '1' and CPU processing branches to the interrupt routine.

Accordingly, the next instruction to be executed will be the interrupt processing program defined by the user.

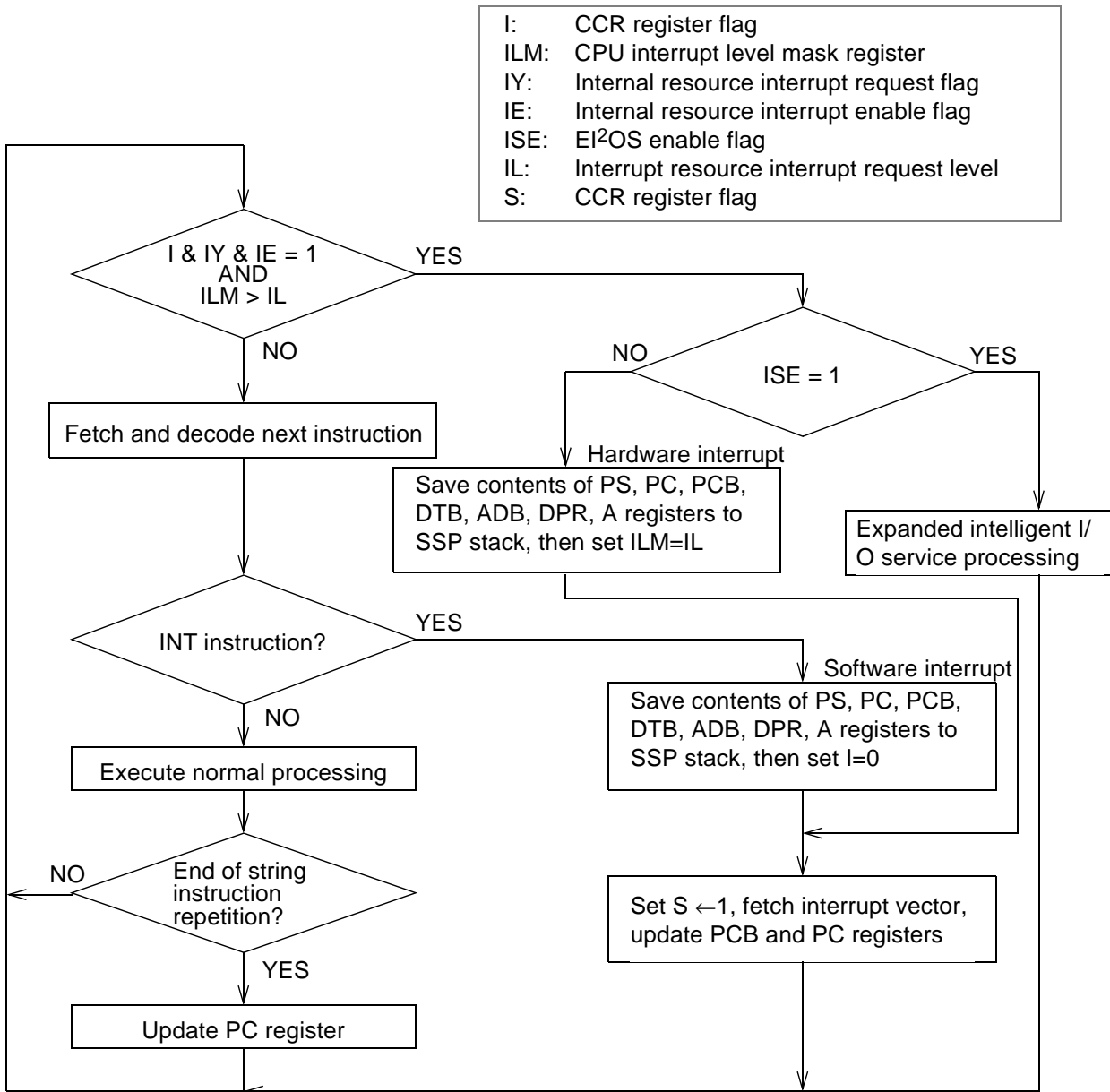
Figure 2.1.25 shows the flow of interrupt processing from the generation of the hardware interrupt, until no more interrupt requests remain in the interrupt request program. Figure 2.1.26 shows the flow of hardware interrupt operations.



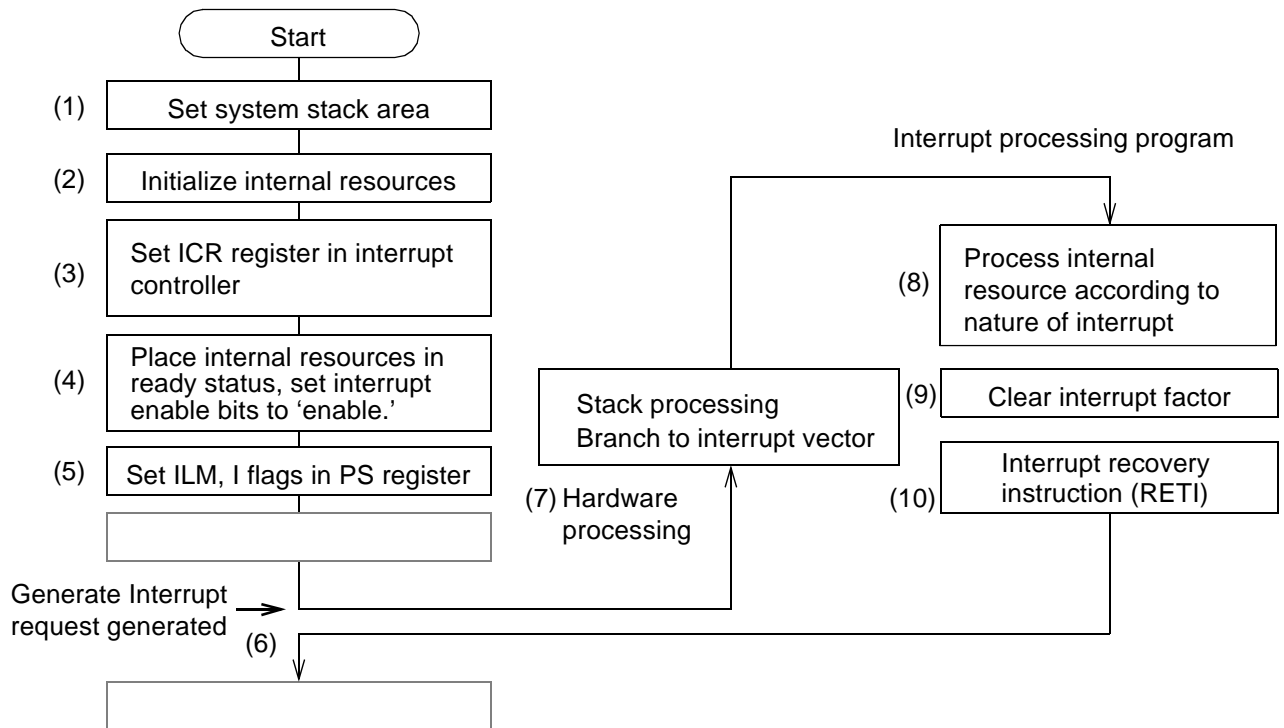
**Fig. 2.1.25 A Hardware Interrupt from Generation to Exit**

- (1) Interrupt factor occurs in internal resource.
- (2) If the interrupt enable bit in that internal resource is set to 'enable,' an interrupt request is generated and sent from the internal resource to the interrupt controller.
- (3) The interrupt controller receives the interrupt request, determines the priority of simultaneously received requests, and transfers it to the CPU with the corresponding interrupt level.
- (4) The CPU receives the interrupt from the interrupt controller, and compares its interrupt level with the value of the IL bit in the processor status (PS) register.
- (5) If the comparison shows a higher priority level than the interrupt level currently being processed, the CPU then checks the value of the I flag in the same processor status (PS) register.
- (6) If the check in step (5) shows that the I flag is set to 'interrupt enabled' status, the processor waits for the end of execution of the instruction that is currently being executed, and then sets the ILM register to the requested level.
- (7) The indicated register settings are saved, and the processor branches, thus transferring control to the interrupt processing routine.
- (8) Software in the user-defined interrupt processing routine clears the interrupt factor that occurred in step (1), and then interrupt processing ends.





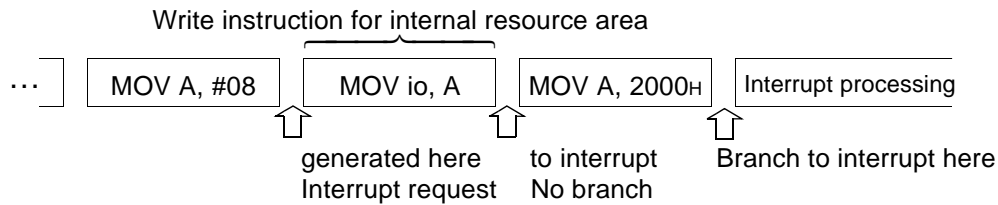
**Fig. 2.1.26 Flow of Interrupt Operations**

**(4) Sample Sequence for Hardware Interrupt Usage****Fig. 2.1.27 Sample Sequence of Hardware Interrupt Usage**

- (1) Set system stack area.
- (2) Initialize internal resources capable of generating interrupt requests.
- (3) Set the ICR register in the interrupt controller.
- (4) Place internal resources in ready status, set interrupt enable bits to 'enable.'
- (5) Set the ILM and I flags in the CPU to enable acceptance of interrupts.
- (6) Interrupt factor occurs in internal resource, generates a hardware interrupt request.
- (7) Interrupt processing hardware saves contents of registers, branches to interrupt processing program.
- (8) Interrupt processing program processes the internal resource that produces the interrupt.
- (9) Clears the interrupt request from the internal resource circuit.
- (10) Execute interrupt recovery instruction, return to the program before branching.

### (5) Hardware Interrupt Requests During Writing to Internal Resource Areas

No hardware interrupt requests will be accepted during writing to internal resource areas. This is in order to avoid abnormal CPU operations that can occur in response to interrupts made during updating of resource interrupt control registers. Internal resource areas are not the I/O addressing areas between 000000H-0000FFH, but the areas allocated to the control registers and data registers of internal resources.



**Fig. 2.1.28 Hardware Interrupt Request During Writing to Internal Resource Areas**

### (6) Interrupt Suppressing Instructions

The F<sup>2</sup>MC-16F core uses certain interrupt suppressing instructions that do not respond to the presence of hardware interrupt requests. See table 2.1.11, "Hardware Interrupt Suppressing Instructions."

### (7) Multiple Interrupts

The F<sup>2</sup>MC-16F core supports multiple interrupts. Thus during processing of one interrupt, when another interrupt of a higher priority level is generated, control is transferred to the higher level interrupt as soon as execution of the current instruction is ended. When processing of the higher level interrupt is completed, control reverts to the first interrupt routine.

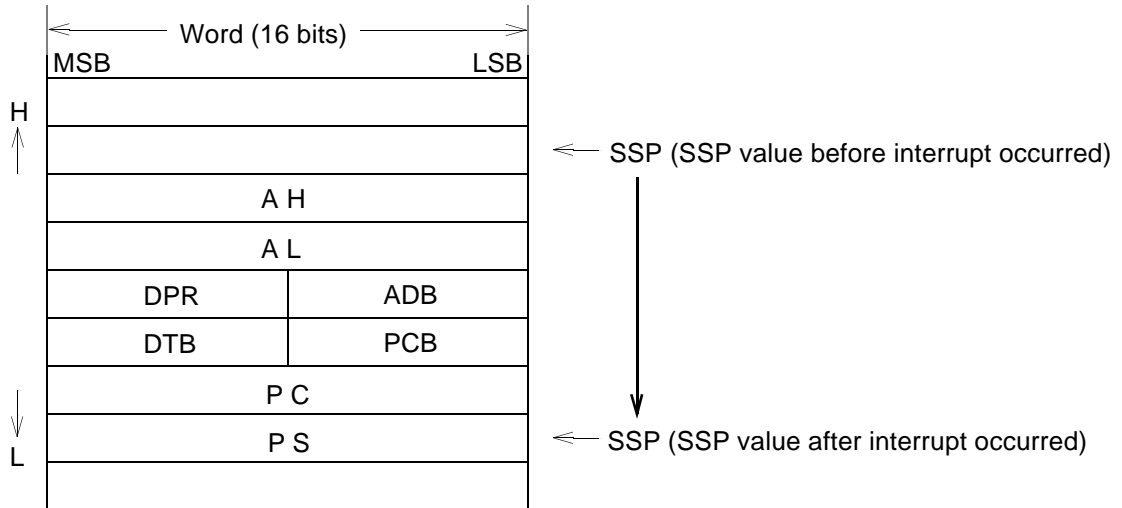
When another interrupt of an equal or lower priority level is received, the new instruction is placed on hold and the current interrupt is processed to completion (unless altered by the ILM register or I flag setting).

Note that multiple extended intelligent I/O services cannot be initiated, so that when one extended intelligent I/O service is being processed, all other interrupt requests and extended intelligent I/O services will be placed on hold.

**(8) Register Saved**

Figure 2.1.29 shows the sequence of saving of registers to the stack.

Register saved in interrupt processing



**Fig. 2.1.29 Registers Saved to Stack**

**(9) Cautionary Information**

In some internal resources, interrupt requests are cleared when the control register or data registers are read. Once an interrupt request is generated, abnormal results may be produced if the interrupt factor is cleared by a read operation before control is passed to the interrupt processing hardware.

For this reason, it is important not to execute register read instructions at the time an interrupt request has occurred when using internal to clear the interrupt request through the register read operation resources.

### ■ Software Interrupts

#### (1) Overview

In a software interrupt, the CPU reacts to the execution of a dedicated instruction, and transfers control from the execution of the program that it has been executing to an interrupt processing program defined by the user. Software interrupts are always initiated by execution of a software interrupt instruction. When a software interrupt is generated, the CPU performs interrupt processing as follows.

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

A software interrupt, or interrupt request by execution of INT instruction in this case, is not provided with an interrupt request flag or enable flag; execution of INT instruction always generates interrupt request.

INT instruction has no interrupt levels; INT instruction, therefore, does not update ILM and the interrupt request that followed is held in retention mode with the I flag set to "0."

#### (2) Configuration

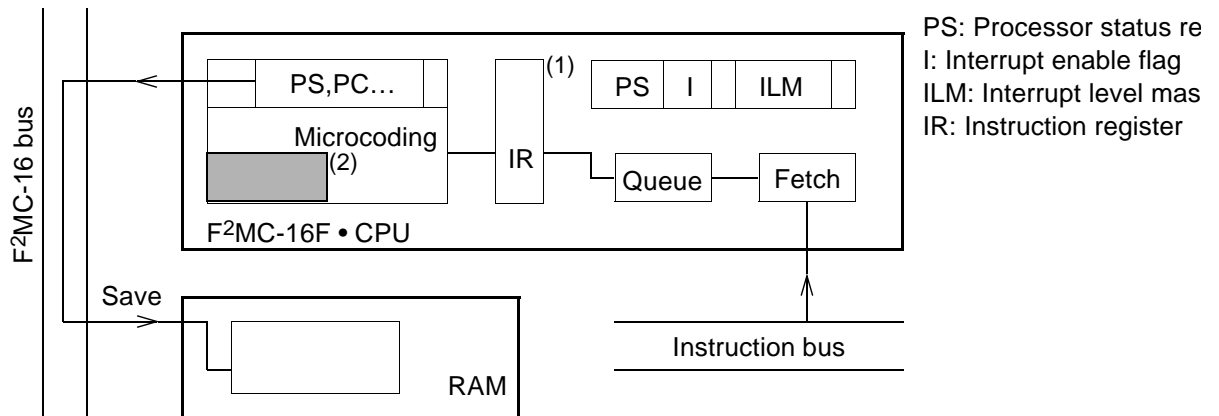
All functions related to software interrupts are contained within the CPU. To use a software interrupt, it is necessary to execute the corresponding instruction.

As shown in Table 2.1.12, interrupt vectors for both hardware interrupts and software interrupts share the same space. For example, interrupt request number INT11 can be used for hardware interrupt #0, and can also be used for software interrupt INT #11. Thus the same interrupt processing subroutine will be called by both hardware interrupt #0 and software interrupt INT #11.

#### (3) Operation

When the CPU fetches and executes a software interrupt command, it activates the software interrupt processing microcoding routine. In software interrupt processing microcoding, the 12 bytes of data contained in memory in the A, DPR, ADB, DTB, PCB, PC and PS registers are saved to the area of memory designated by the SSB and SSP registers, then the contents of the 3-byte interrupt vector is read and loaded into the PC and PCB register, the I flag is set to '0' and the S flag is set to '1,' and CPU processing branches to the interrupt routine.

Figure 2.1.30 shows the flow of interrupt processing from the generation of the software interrupt, until no more interrupt requests remain in the interrupt request program.



**Fig. 2.1.30 A Software Interrupt from Generation to Resolution**

- (1) Software interrupt instruction is executed.
- (2) Contents of dedicated registers saved according to microcoding of software interrupt instruction.
- (3) In user-defined interrupt processing routine, interrupt processing with RETI, RETIQ instruction ends.

**(4) Precaution**

If the program bank register (PCB) is up to FF<sub>H</sub>, vector region for CALLV instruction is overlapped with the table for INT #vct8 instruction. Exercise care for the overlapped address of CALLV instruction and INT #vct8 instruction.

■ Extended Intelligent I/O Service (EI<sup>2</sup>OS)

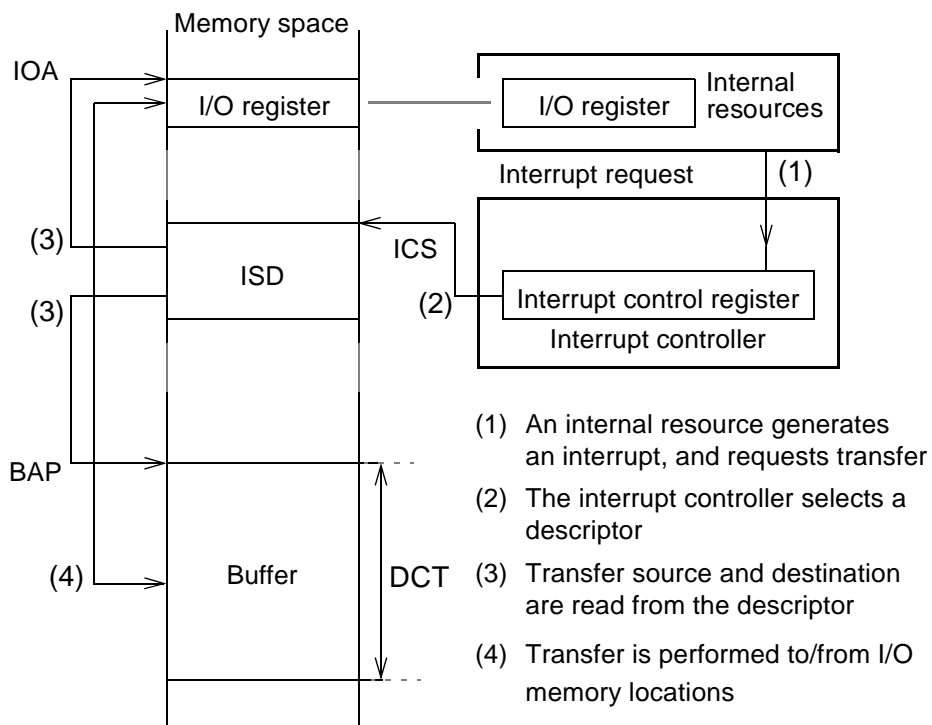
EI<sup>2</sup>OS is a set of automatic data transfers between I/O ports and memory, initiated by interrupts generated from internal resources. These function cause temporary suspension of instruction execution, and allow exchange of data with the I/O ports which are conventionally executed by the interrupt processing program.

This method has the following advantages in comparison with the conventional interrupt processing methods.

- Elimination of need to write transfer programs, allowing reduction of overall program size.
- No internal registers used for transfer, eliminating the need to save register contents and thereby speeding up transfer processing.
- Transfer can be stopped according to I/O status, eliminating unnecessary data transfers.
- Selection of three buffer address handling modes: increment/decrement/no-change.
- Selection of three I/O register address handling modes: increment/decrement/no-change (increment/decrement settings used only when buffer address is changed).

When EI<sup>2</sup>OS processing ends, processing branches automatically to the interrupt processing program as soon as end conditions are set, allowing the user to determine the type of ending conditions used.

Figure 2.1.31 shows an overview of extended intelligent I/O service operation.



**Fig. 2.1.31 Overview of Extended Intelligent I/O Service**

■ Structure of Extended Intelligent I/O Service

Functions related to EI<sup>2</sup>OS may be divided into 4 basic areas.

- ▷ Internal resources ..... Interrupt enable bit, interrupt request bit: Controls interrupt request from internal resources.
- ▷ Interrupt controller ..... ICR: Assigns interrupt level, determines priority of simultaneous interrupt requests, selects E<sup>2</sup>OS operations.
- ▷ CPU ..... I, ILM: Compares requested interrupt level with current level, determines interrupt enable status.  
Microcoding: Executes EI<sup>2</sup>OS processing steps
- RAM..... Descriptor: Writes IE<sup>2</sup>OS transfer data

A description of each of the above registers follows.

■ Interrupt Control Register (ICR)

The interrupt control register (ICR) is located inside the interrupt controller, and supports all I/O resources that have interrupt functions. For the relation between interrupts and the ICR register, see section 2.2.3, "Interrupt Vector Allocation." This register has the following three functions.

- Sets interrupt levels for each related internal resource
- Determines whether interrupts from related internal resources are to be handled as normal interrupts or as extended intelligent I/O services
- Selects channels for extended intelligent I/O services

Caution: Attempted access to this register by read-modify-write instructions (those instructions indicated by an asterisk (\*) in the RMW column in the instruction tables) may result in abnormal operation, and should be avoided.

Figure 2.1.32 shows the bit configuration of the interrupt control register.

7	6	5	4	3	2	1	0	Interrupt control register (ICR) write configuration Reset : 00000111 <sub>B</sub>
ICS3	ICS2	ICS1	ICS0	ISE	IL2	IL1	ILO	
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	Interrupt control register (ICR) read configuration Reset : xx000111 <sub>B</sub>
—	—	S1	S0	ISE	IL2	IL1	ILO	
—	—	R	R	R	R	R	R	

**Note:** Caution: The ICS3 to ICS0 bits are effective only when EI<sup>2</sup>OS has been started. The ISE bit is set to '1' to start EI<sup>2</sup>OS, and otherwise set to '0.' If EI<sup>2</sup>OS has not been started, the ICS0 to ICS3 bits may have any value.

**Fig. 2.1.32 Interrupt Control Register (ICR)**



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

These bits are used to set interrupt levels. Each bit is read/write enabled and sets the interrupt level of the corresponding internal resource. The initial value after reset is '7' (no interrupt). For the relation between interrupt level setting bits and interrupt levels, see Table 2.1.13.

**Table 2.1.13 Relation between Interrupt Level Setting Bits and Interrupt Levels**

IL2	IL1	IL0	Level value
0	0	0	0 (strongest interrupt)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6 (weakest interrupt)
1	1	1	7 (no interrupt)

[Bit 3] ISE: Extended Intelligent I/O Service Enable Bit

This bit is used to enable EI<sup>2</sup>OS. It is read/write enabled, and is set to '1' to initiate EI<sup>2</sup>OS when an interrupt is generated. A setting of '0' will start the interrupt sequence. Also, when EI<sup>2</sup>OS ends (either by count end or internal resource-controlled end), the ISE bit is set to '0.' If the corresponding internal resource has no EI<sup>2</sup>OS function, a software instruction must be used to set the ISE bit to '0.'

The value is initialized to '0' after a reset.

[Bit 7 to 4] ICS3 to ICS0: Extended Intelligent I/O Service Channel Select Bits

These bits are used to select EI<sup>2</sup>OS channels. Each is write-only, and designates an EI<sup>2</sup>OS channel. The values defined by these bits determine the address of an extended intelligent I/O service descriptor (ISD). The ICS bits are initialized to '0000' following a reset.

Table 2.1.14 shows the relation between the ICS bits and corresponding channel numbers and descriptor (ISD) addresses.

**Table 2.1.14 ICS Bit Values, Channel Numbers, and Descriptor Addresses**

ICS3	ICS2	ICS1	ICS0	Selected 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

\* Because these areas are shared with the register save areas for stack area errors (see 'Exceptions'), care should be taken so that one or the other of these areas is used, but not both.

[Bit 5 to 4] S0, S1: Extended Intelligent I/O Service Status

These are the EI<sup>2</sup>OS status bits. They are read-only, and are used to indicate operating status and end status. The value is initialized to '00' after a reset.

Table 2.1.15 indicates the relation between the S bits and EI<sup>2</sup>OS status.

**Table 2.1.15 S Bits and EI<sup>2</sup>OS Status**

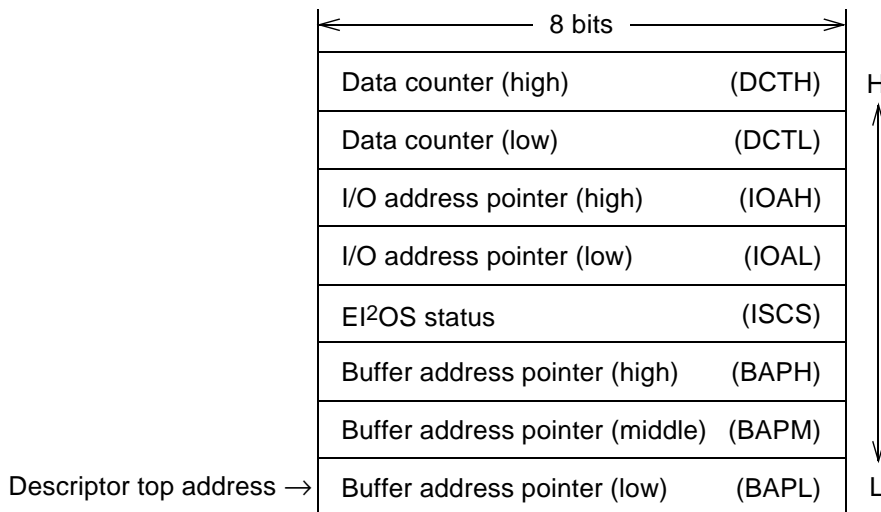
S1	S2	EI <sup>2</sup> OS Status
0	0	EI <sup>2</sup> OS operating or not started
0	1	Stop status caused by count end
1	0	Reserved
1	1	Stop status caused by request from internal resource

■ Extended Intelligent I/O Service Descriptor (ISD)

The extended intelligent I/O service descriptor is located at internal RAM addresses 000100<sub>H</sub>-00017F<sub>H</sub>, and is composed of the following.

- Control data for data transfer
- Status data
- Buffer address pointer

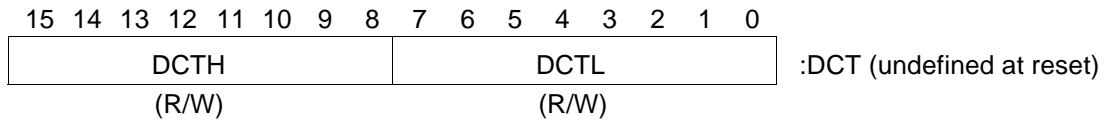
Figure 2.1.33 shows the configuration of the extended intelligent I/O service descriptor.



**Fig. 2.1.33 Configuration of the Extended Intelligent I/O Service Descriptor**

■ Data Counter (DCT)

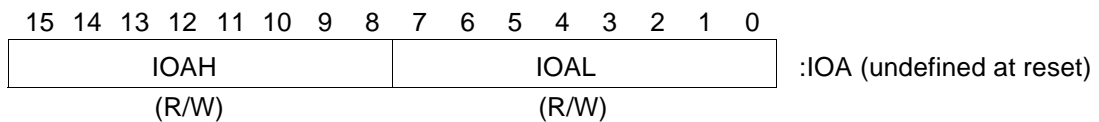
This 16-bit register is used as a counter for transfer data items. The counter value is decremented by 1 after each data transfer. When this counter value reaches zero, EI<sup>2</sup>OS is ended. Figure 2.1.34 shows the configuration of the data counter.



**Fig. 2.1.34 Configuration of Data Counter**

■ I/O Register Address Pointer (IOA)

This 16-bit register is used to indicate 16 bits of the lower address (A15 to A00) of the I/O register transferring data to and from the buffer. The upper address (A23 to A16) is all zeros and can designate any I/O address between 000000H and 00FFFFH. Figure 2.1.35 shows the configuration of the IOA register.



**Fig. 2.1.35 Configuration of I/O Register Address Pointer**

### ■ EI<sup>2</sup>OS Status Register (ISCS)

This 8-bit register is used to set the direction of update movement (increment/decrement) of the buffer address pointer and the I/O register address pointer, transfer data length (byte/word), transfer direction, and update/hold of the buffer address pointer and I/O register address pointer. Figure 2.1.36 shows the configuration of the ISCS register.

7	6	5	4	3	2	1	0	:ISCS (undefined at reset)
IF	BF	–	–	ID	BW	DIR	SE	
(R/W)	(R/W)	(–)	(–)	(R/W)	(R/W)	(R/W)	(R/W)	

Always write '0' to the two empty bits of the ISCS register.

**Fig. 2.1.36 Configuration of the ISCS Register**

The contents of each bit is defined as follows.

[Bit 7] IF: This bit determines whether the I/O register address pointer is updated or held constant.

0: After data transfer, the I/O register address pointer is updated.

1: After data transfer, the I/O register address pointer is held constant.

**[CAUTION]** The pointer cannot be updated when the BF bit is '1.'

[Bit 6] BF: This bit determines whether the buffer address pointer is updated or held constant.

0: After data transfer, the buffer address pointer is updated.

1: After data transfer, the buffer address pointer is held constant.

**[CAUTION]** When updated, only the lower 16 bits of the buffer address pointer are changed.

[Bit 3] ID: This bit determines the buffer address pointer and I/O register address pointer update direction.

0: Increment

1: Decrement

**[CAUTION]** The ID bit has no significance when the BF bit is '1.'

[Bit 2] BW: This bit indicates the transfer data length.

0: Byte

1: Word

[Bit 1] DIR: This bit indicates data transfer direction.

0: I/O address pointer → buffer address pointer

1: Buffer address pointer → I/O address pointer

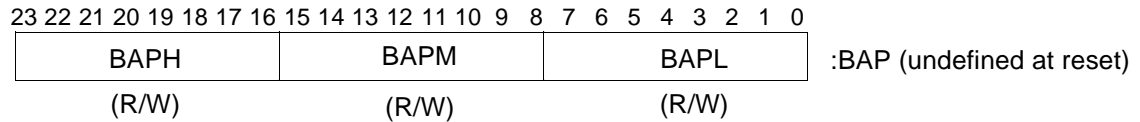
[Bit 0] SE: This bit controls the end of extended intelligent I/O service by request from internal resources.

0: Not ended by request from internal resources

1: Ended by request from internal resources

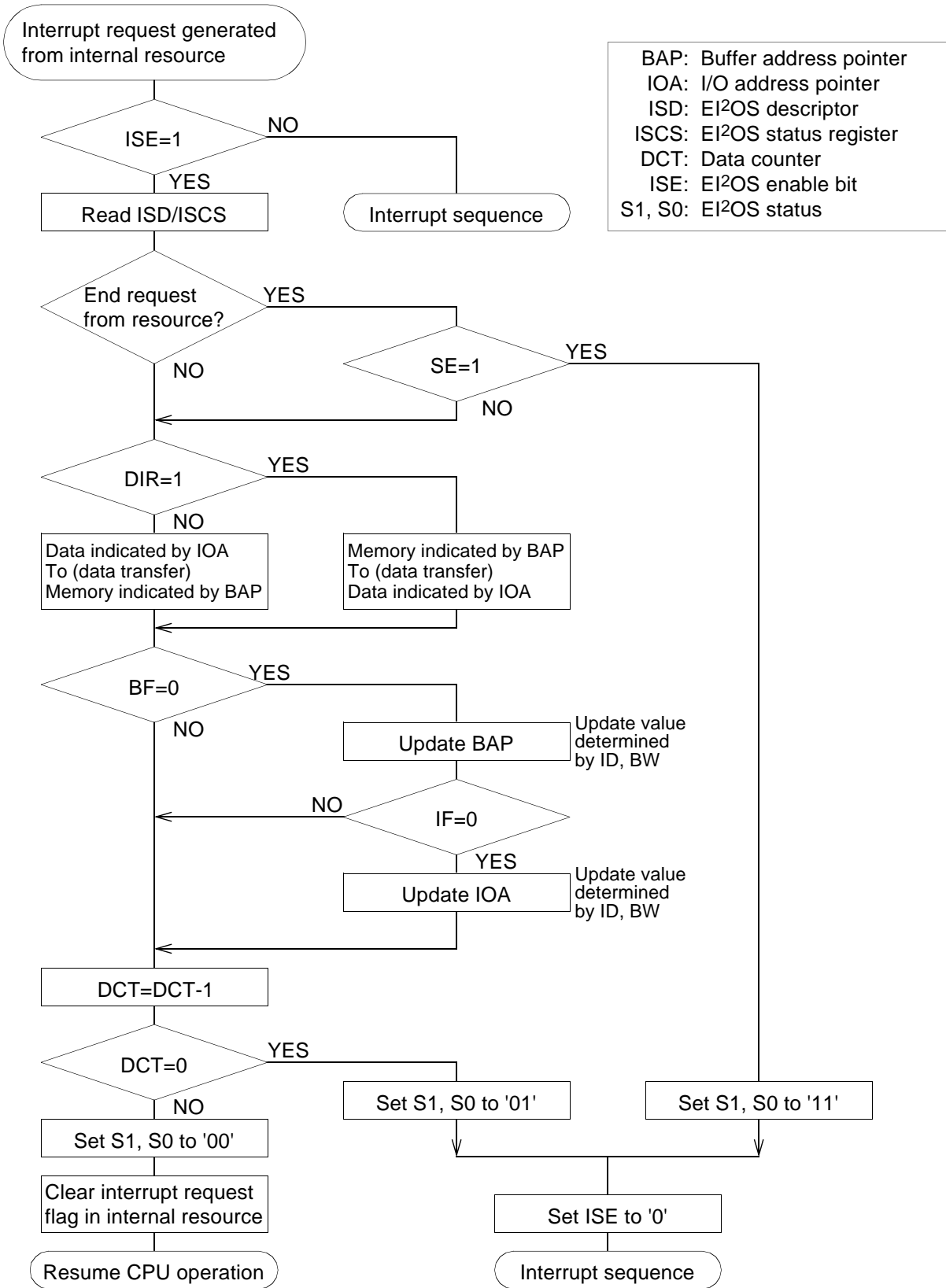
### ■ Buffer Address Pointer (BAP)

This 24-bit register stores addresses to be used for transfer by the next EI<sup>2</sup>OS. A separate BAP register exists for each EI<sup>2</sup>OS channel, so that each EI<sup>2</sup>OS channel can execute transfer to any portion of the entire 16 Mbyte memory space. When the BF bit in the ISCS register is set to 'update enable,' only the lower 16 bits of the BAP register will be updated, and the BAPH field will not be changed. Figure 2.1.37 shows the configuration of the BAP register.



**Fig. 2.1.37 Configuration of Buffer Address Pointer**

Figure 2.1.38 shows the operating flow of EI<sup>2</sup>OS, and Figure 2.1.39 shows the sequence of EI<sup>2</sup>OS operation by the user.



**Fig. 2.1.38 Flow of EI<sup>2</sup>OS Operation**

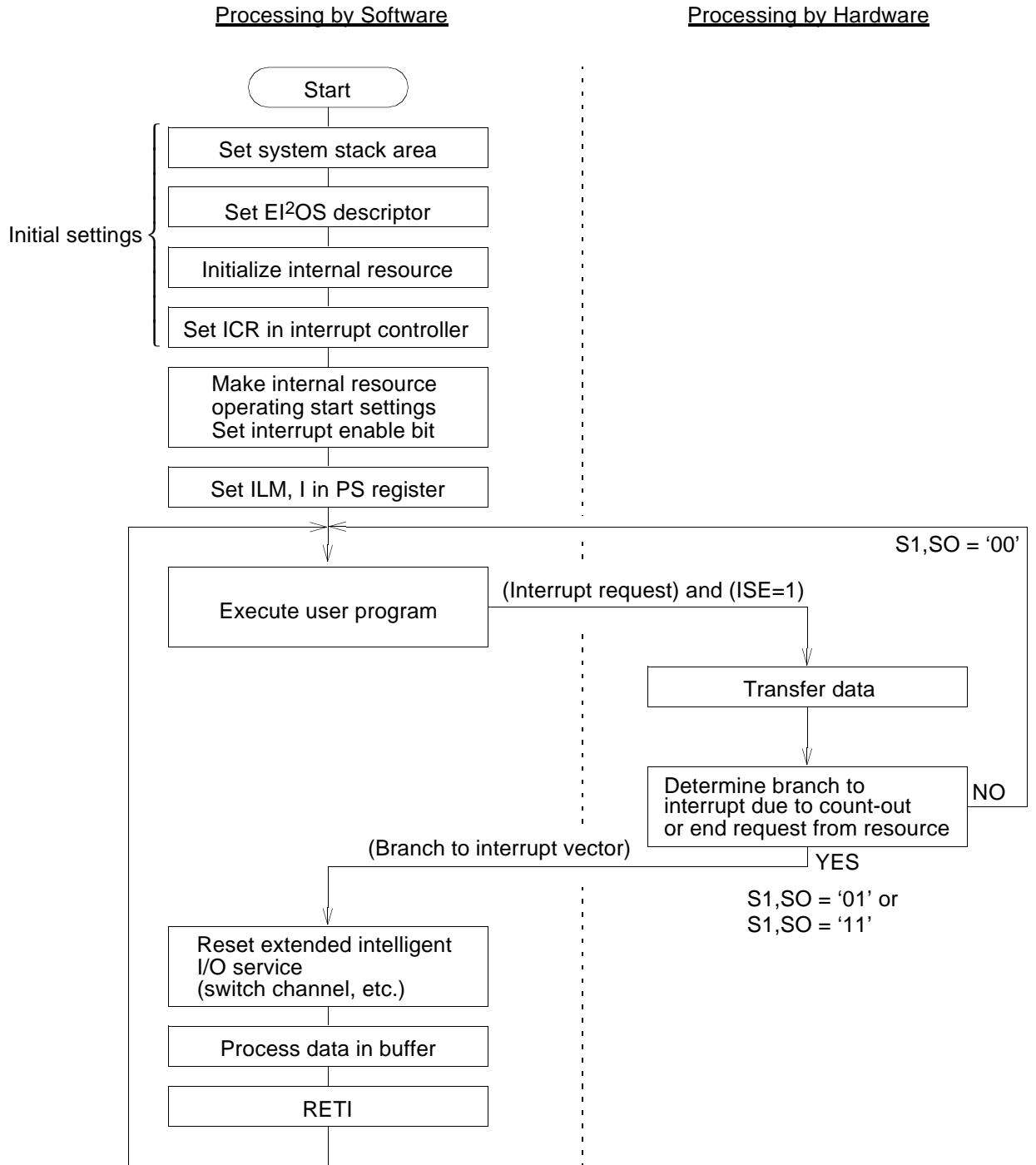


Fig. 2.1.39 Flow of User Operation for EI<sup>2</sup>OS



### ■ Exception Processing

The F<sup>2</sup>MC-16F core provides exception processing for exceptions arising from the following causes.

- (1) Execution of undefined instructions
- (2) Program access address errors (for program access to internal RAM and internal I/O areas)
- (3) Stack area errors due to stack area check functions

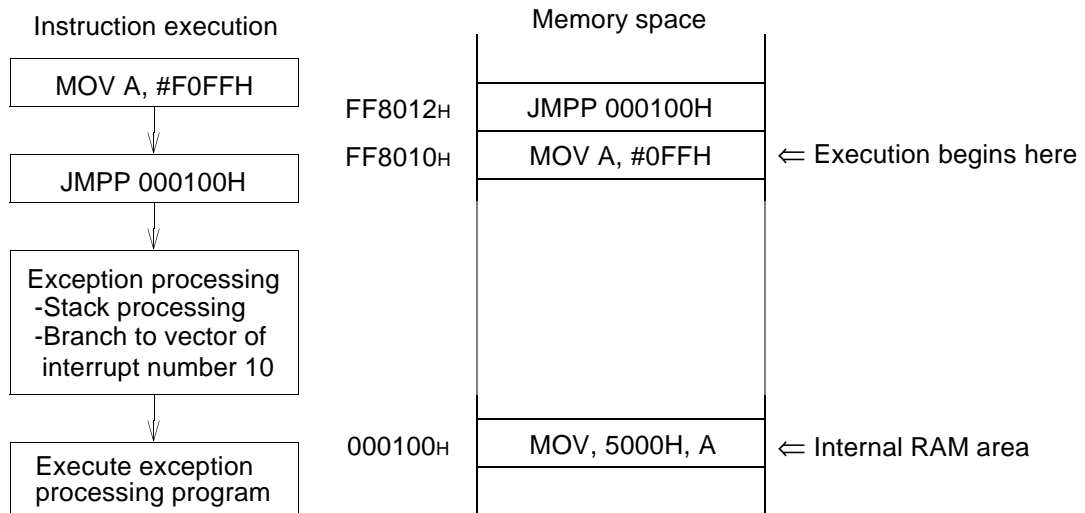
Exception processing is fundamentally the same as interrupt processing, in that exception processing departs from normal processing sequence at the point where the occurrence of an abnormal condition is detected at the boundary between instructions. In general, exception processing occurs as a result of unexpected operation, and its use is recommended only in debugging or for startup of recovery software in emergency situations.

### ■ Exceptions Occurring from Execution of Undefined Instructions

The F<sup>2</sup>MC-16F considers all codes not defined on the instruction map to be undefined instructions. Execution of undefined instructions is handled as the equivalent of the software interrupt instruction 'INT 10.' This means that after the contents of the A, DPR, DTB, ADB, PCB, PC and PS registers are saved to the system stack, the I flag is set to '0' and the S flag to '1' and the program branches to the vector indicated by interrupt number 10. The value of register PC that is placed in the stack will be the address containing the undefined instruction. For this reason, recovery using the RETI and RETIQ instructions is possible but meaningless since another exception will occur.

■ Program Access Address Errors with Internal RAM and Internal I/O Areas

The F<sup>2</sup>MC-16F core treats access to internal RAM or areas defined as internal I/O as the equivalent of the software interrupt instruction 'INT 10.' This means that after the contents of the A, DPR, DTB, ADB, PCB, PC and PS registers are saved to the system stack, the I flag is set to '0' and the S flag to '1' and the program branches to the vector indicated by interrupt number 10. The value of register PC that is placed in the stack will be the address to which program access was attempted resulting in the exceptional condition. For this reason, recovery using the RETI and RETIQ instructions is possible but meaningless since another exception will occur.



**Fig. 2.1.40 Program Access Address Error**

■ Stack Area Errors Due to Stack Area Check Functions

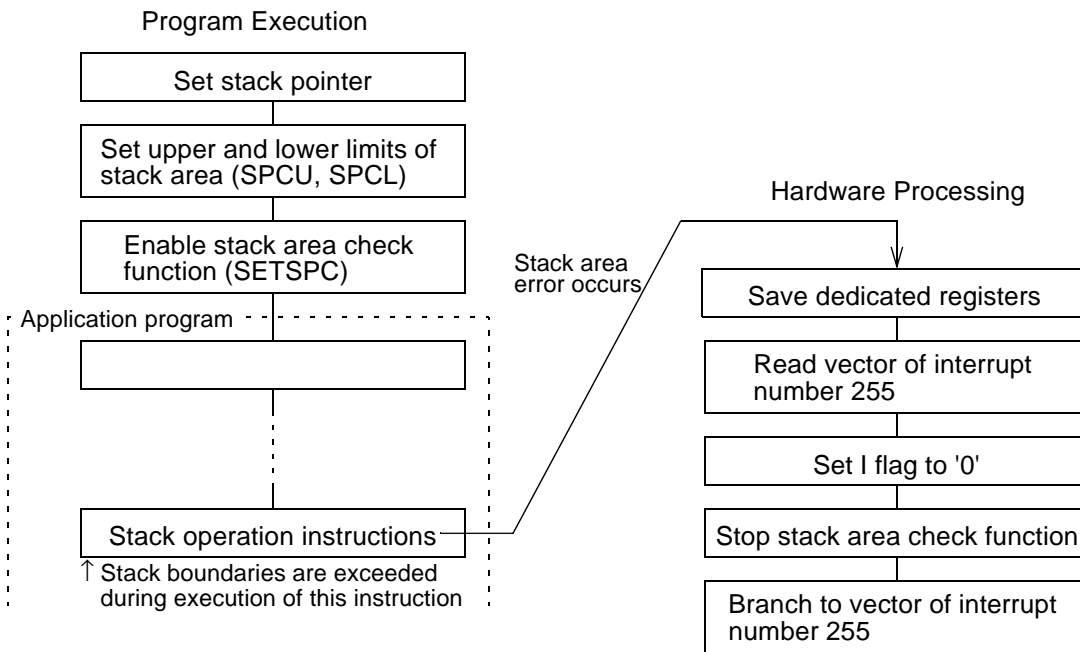
The F<sup>2</sup>MC-16F core has stack area check function, which can detect conditions where the current stack pointer moves outside the area defined by the stack upper/lower limit register. Exception processing as a result of this condition is called a stack area error.

After the stack check function is set to 'enable' status, any time the current stack pointer moves outside of the area defined by the stack upper limit register (SPCU) and stack lower limit register (SPCL), the contents of the A, DPR, DTB, ADB, PCB, PC, and PS registers will be saved to the area 000174H to 00017FH, after which the I flag will be set to '0', the stack check function will stop, and the program will branch to the vector indicated by interrupt number 255. The value of the PC register, saved to address 000176H, will be the address containing the instruction following the instruction that contained the stack access responsible for the exception.

The user is warned that this function is primarily intended for debugging, and therefore it is not possible to return to the original program by using the RETI and RETIQ instructions.

Separate stack upper limit (SPCU) registers and stack lower limit (SPCL) registers are provided for the user stack pointer and the system stack pointer. The MOVW SPCU, #imm16/MOVW SPCL, #imm16 instructions can be used for the user stack area settings when the S flag is '0' and for the system stack area settings when the S flag is '1.'

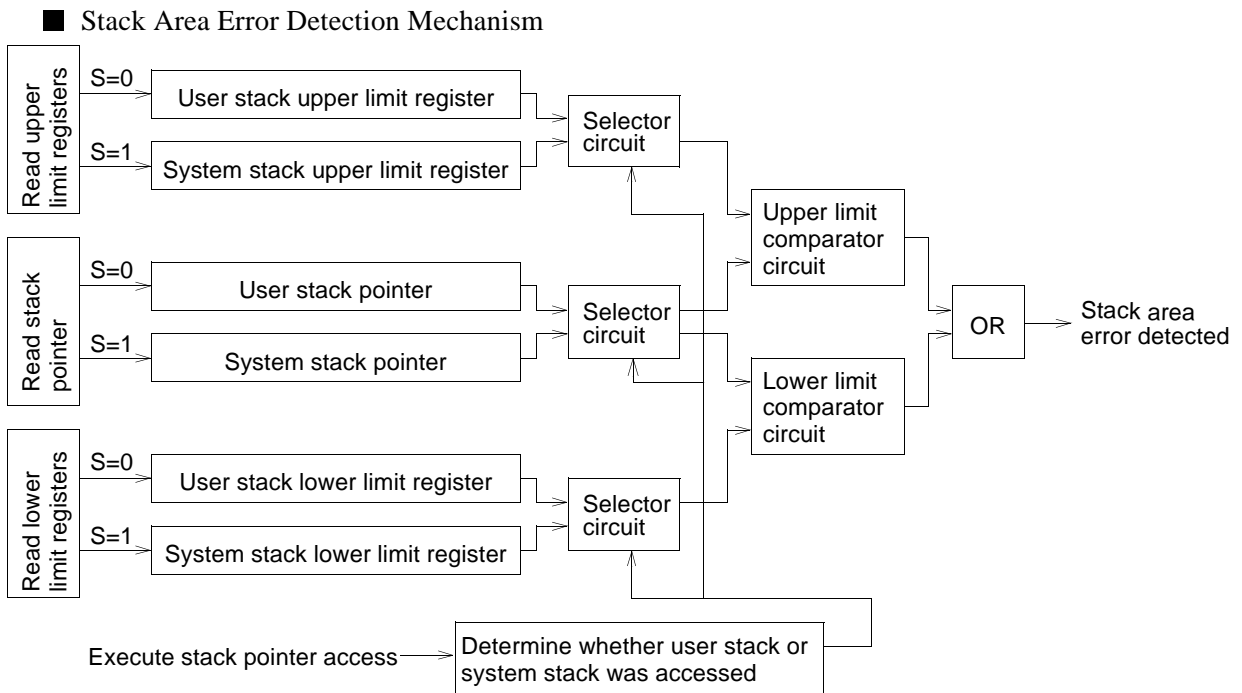
Figure 2.1.41 shows the flow of the stack check function, and Figure 2.1.42 shows the saving of registers when a stack area error occurs.



**Fig. 2.1.41 Flow of Stack Check Function**

	← Word (16 bits) →	
	MSB	LSB
00017EH	AH	
00017CH	AL	
00017AH	DPR	ADB
000178H	DTB	PCB
000176H	PC	
000174H	PS	

**Fig. 2.1.42 Register Saving Due to Stack Area Error**



**Fig. 2.1.43 Block Diagram of Stack Area Error Detection Mechanism**

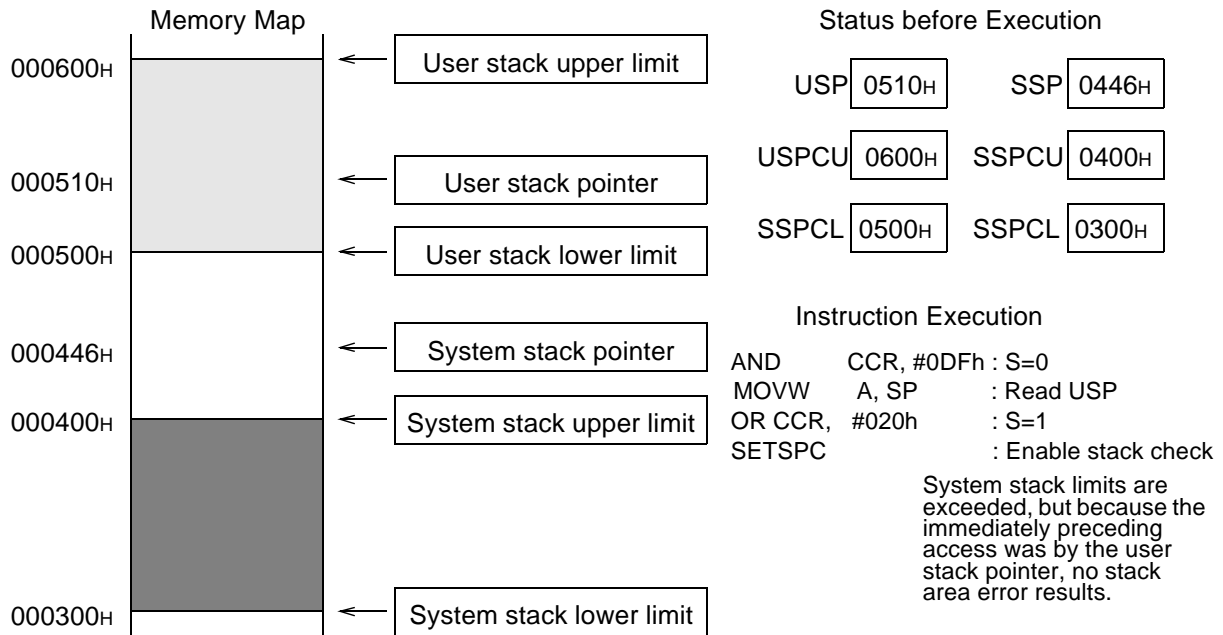
Figure 2.1.43 shows a block diagram of the stack area error detection mechanism.

The S flag in the CCR register alone determines writing to each stack pointer and upper/lower limit register.

The upper limit comparator circuit and lower limit comparator circuit are realized on the same hardware for both the user stack area and system stack area, and switching between the two is governed by the determination "whether user stack or system stack was accessed" with no changes made to the S flag itself. Also, because stack area error detection takes place at the end of each instruction, the following situations may occur.

- (1) When the stack check function is enabled, the current stack (according to the value of the S flag) area may be exceeded, with no stack area error resulted (Figure 2.1.44).
- (2) The S flag and current stack may be changed, without a stack area error occurring in the current stack.
- (3) Stack area limits may be exceeded before a stack operation instruction, but no stack area error may result because the pointer is within the limits after the instruction is executed.

For this reason, when the stack check function is in enabled status, or immediately following a change in the current stack as a result of changes in the S flag, it is necessary to execute the MOVW A, SP instruction to check the stack area.



**Fig. 2.1.44 Example: No Stack Area Error Occurs**

■ Cautionary Information Related to Stack Check Mechanism

- 1) The structure of the stack check mechanism does not enable it to check bank addresses.
- 2) The stack check function is stopped within the stack area error processing hardware, so that it is unnecessary to execute the CLRSPC instruction in the stack area error processing program.
- 3) The value of the S flag and stack pointer will indicate which stack pointer and which limit (upper/lower) was responsible for the error.

### 2.1.5 Standby Control Register Access

The MB90242A series microcontroller can be placed in any of the power saving modes (stop or sleep mode) by writing to the standby control register, provided that the instructions used are those shown in Table 2.1.16. Operation of the MB90242A series microcontroller is not warranted if instructions other than those listed in Table 2.1.16 are used to place the chip in power saving modes. When the standby control register is used to control functions other than change to power saving modes, any instructions may be used.

**Table 2.1.16 Instructions Used to Change To and From Power Saving Modes**

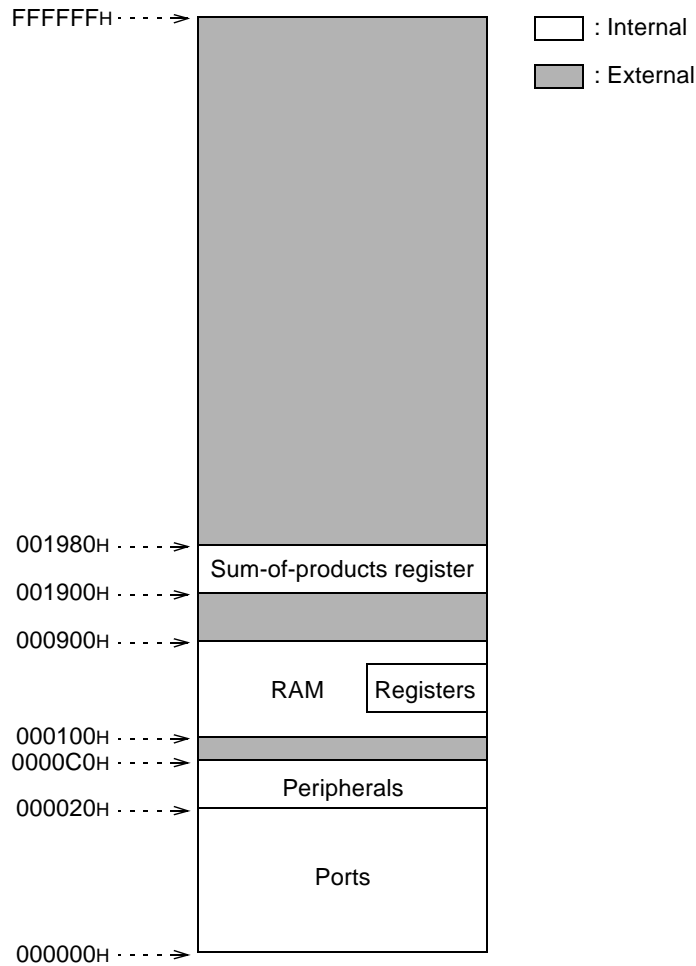
MOV io,#imm8	MOV dir,#imm8	MOV eam,#imm8	MOV eam,Ri
MOV io,A	MOV dir,A	MOV addr16,A	MOV eam,A
MOV @RLi+disp8,A	MOV @SP+disp8,A	MOVP addr24,A	
MOVW io,#imm16	MOVW dir,#imm16	MOVW eam,#imm16	MOVW eam,RWi
MOVW io,A	MOVW dir,A	MOVW addr16,A	MOVW eam,A
MOVW @RLi+disp8,A	MOVW @SP+disp8,A	MOVW addr24,A	
SETB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

## 2.2 Maps

This section describes the allocation of MB90242A series memory space, I/O space and interrupt numbers.

### 2.2.1 Memory Space

Figure 2.2.1 shows MB90242A memory space.



**Fig. 2.2.1 MB90242A Memory Space**

## 2.2.2 I/O Map

The following table provides an I/O map of the MB90242A microcontroller.

**Table 2.2.1 MB90242A I/O Map (1)**

Address	Register	Abbreviation	Access	Resource name	Initial value
000000H	System reserved are	—————	Note 1	—————	—————
000001H	Port 1 data register	PDR1	R/W	Port 1	XXXXXXXX
000002H to 03H	System reserved area	—————	Note 1	—————	—————
000004H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX
000005H	Port 5 data register	PDR5	R/W	Port 5	XXXXXXXX
000006H	Port 6 data register	PDR6	R/W	Port 6	11--1111
000007H	Port 7 data register	PDR7	R/W	Port 7	--XXXXXX
000008H	Port 8 data register	PDR8	R/W	Port 8	----XXX
000009H to 0FH	Vacancy				—————
000010H	System reserved area	—————	Note 1	—————	—————
000011H	Port 1 direction register	DDR1	R/W	Port 1	00000000
000012H to 13H	System reserved area	—————	Note 1	—————	—————
000014H	Port 4 direction register	DDR4	R/W	Port 4	00000000
000015H	Port 5 direction register	DDR5	R/W	Port 5	00000000
000016H	Analog enable register	ADER	R/W	Analog enable	--111111
000017H	Port 7 direction register	DDR7	R/W	Port 7	--000000
000018H	Port 8 direction register	DDR8	R/W	Port 8	----000
000019H to 1FH	Vacancy				—————
000020H	Control status register	SCR1	R/W	I/O simple serial interface ch1	10000000
000021H	Status register	SSR1	R		-----0
000022H 000023H	Serial data register	SDR1L SDR1H	R/W		XXXXXXXX XXXXXXXX
000024H	Vacancy				—————
000025H	Vacancy				—————
000026H 000027H	Vacancy				—————



**Table 2.2.1 MB90242A I/O Map (2)**

Address	Register	Abbreviation	Access	Resource name	Initial value
000028H	Mode control register	UMC0	R/W	UART ch0	00000100
000029H	Status register	USR0	R/W		00010000
00002AH	Input data register/output data register	UIDR0 /UODR0	R/W		XXXXXXXX
00002BH	Rate and data register	URD0	R/W		0000000X
00002CH to 2FH	Vacancy				—————
000030H	Interrupt/DTP enable register	ENIR	R/W	DTP/external inter- rupt	----0000
000031H	Interrupt/DTP factor register	EIRR	R/W		----0000
000032H	Request level setting register	ELVR	R/W		00000000
000033H to 3FH	Vacancy				—————
000040H	Timer control status register #0	TMCSR0	R/W	16-bit timer #0	00000000
000041H			----0000		
000042H	16-bit timer register #0	TMR0	R		XXXXXXXX
000043H					XXXXXXXX
000044H	16-bit timer reload register #0	TMRLR0	R/W		XXXXXXXX
000045H					XXXXXXXX
000046H to 47H	Vacancy				—————
000048H	Timer control status register #1	TMCSR1	R/W	16-bit timer #1	00000000
000049H					----0000
00004AH	16-bit timer register #1	TMR1	R		XXXXXXXX
00004BH					XXXXXXXX
00004CH	16-bit timer reload register #1	TMRLR1	R/W		XXXXXXXX
00004DH					XXXXXXXX
00004EH to 4FH	Vacancy				—————

**Table 2.2.1 MB90242A I/O Map (3)**

Address	Register	Abbreviation	Access	Resource name	Initial value
000050H	Vacancy				_____
000051H					_____
000052H	Vacancy				_____
000053H					_____
000054H	Vacancy				_____
000055H					_____
000056H to 57H	Vacancy				_____
000058H	Vacancy				_____
000059H	Vacancy				_____
00005AH	Vacancy				_____
00005BH					_____
00005CH	Vacancy				_____
00005DH	Vacancy				_____
00005EH	Vacancy				_____
00005FH	Vacancy				_____
000060H	Capture register 0	ICP0	R/W	Input capture 0, 1	XXXXXXXX
000061H					XXXXXXXX
000062H	Capture register 1	ICP1	R/W		XXXXXXXX
000063H					XXXXXXXX
000064H	Control status register 0, 1	ICS0	R/W		00000000
000065H	Vacancy				_____
000066H	Capture register 2	ICP2	R/W	Input capture 2, 3	XXXXXXXX
000067H					XXXXXXXX
000068H	Capture register 3	ICP3	R/W		XXXXXXXX
000069H					XXXXXXXX
00006AH	Control status register 2, 3	ICS1	R/W		00000000
00006BH	Vacancy				_____
00006CH	Data register	TCDT	R	16 bit Free-run timer	00000000
00006DH					00000000
00006EH	Control status register	TCCS	R/W		00000000

## 2.2 Maps

**Table 2.2.1 MB90242A I/O Map (4)**

Address	Register	Abbreviation	Access	Resource name	Initial value
00006FH	Vacancy				————
000070H	A/D control register	ADCS	R/W	A/D converter	000-0000
000071H					-000--00
000072H	Conversion time setting register	ADCT	R/W		XXXXXXXX
000073H					XXXXXXXX
000074H	Conversion data register 0	ADTL0	R		XXXXXXXX
000075H					ADTH0
000076H	Conversion data register 1	ADTL1	R		XXXXXXXX
000077H					ADTH1
000078H	Conversion data register 2	ADTL2	R		XXXXXXXX
000079H					ADTH2
00007AH	Conversion data register 3	ADTL3	R	XXXXXXXX	
00007BH				ADTH3	-----XX
00007CH to 7FH	Vacancy				————
000080H	Sum-of-products control status register	MCSR	R/W	DSP unit	-0XXXXXX
000081H					XXX0XXX0
000082H	Sum-of-products calculation continuous control register	MCCR	R/W		00000000
000083H	System reserved area	————	Note 1		————
000084H	Sum-of-products output register	MDORLL	R	XXXXXXXX	
000085H				MDORLH	XXXXXXXX
000086H				MDORHL	XXXXXXXX
000087H				MDORHH	XXXXXXXX
000088H				MDOROH	XXXXXXXX
000089H to 8FH	Vacancy				————
000090H to 9EH	System reserved area	————	Note 1	————	————
00009FH	Delay interrupt source generate/clear register	DIRR	R/W	Delay interrupt source generator module	-----0
0000A0H	Standby control register	STBYC	R/W	Power saving mode	0001XXXX

**Table 2.2.1 MB90242A I/O Map (5)**

Address	Register	Abbreviation	Access	Resource name	Initial value
0000A4H	Upper address control register	HACR	W	External pins	Note 2
0000A5H	External pin control register	EPCR	W	External pins	Note 2
0000A8H	Watchdog timer control register	TWC	R/W	Watchdog timer	XXXXXXXX
0000A9H	Timebase timer control register	TBTC	R/W	Timebase timer	XXX00000
0000B0H	Interrupt control register 00	ICR00	R/W		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 FFH	External area Note 3	_____	_____	_____	_____

**Note1:** Access prohibited

**Note2:** Initial value varies depending on bus mode.

**Note3:** Of the areas at address 0000FFH and lower, this is the only external access area. All addresses not described in this table are reserved addresses, and access to these addresses is handled as an internal area operation. No access signal for an external bus is generated.

Description of initial values

'0' .....This bit is initialized to value 0.

'1' .....This bit is initialized to value 1.

'X' .....Initial value of this bit is undefined.

'-' .....This bit is not used. Initial value undefined.

### 2.2.3 Interrupt Vector Allocation

The following table shows the allocation of MB90242A interrupt vectors.

**Table 2.2.2 MB90242A Interrupt Vector Allocation**

Interrupt Factor	Relation to EI2OS	Interrupt vector			Interrupt control register	
		Number	Address	Address	ICR	Address
Reset	×	#08	08 <sub>H</sub>	FFFFDC <sub>H</sub>	–	–
INT9 instruction	×	#09	09 <sub>H</sub>	FFFFD8 <sub>H</sub>	–	
Exception	×	#10	0A <sub>H</sub>	FFFFD4 <sub>H</sub>	–	
External interrupt #0	○	#11	0B <sub>H</sub>	FFFFD0 <sub>H</sub>	ICR00	0000B0 <sub>H</sub>
External interrupt #1	○	#13	0D <sub>H</sub>	FFFFC8 <sub>H</sub>	ICR01	0000B1 <sub>H</sub>
ICU #0	○	#15	0F <sub>H</sub>	FFFFC0 <sub>H</sub>	ICR02	0000B2 <sub>H</sub>
ICU #1	Δ	#17	11 <sub>H</sub>	FFFFB8 <sub>H</sub>	ICR03	0000B3 <sub>H</sub>
External interrupt #2	Δ	#19	13 <sub>H</sub>	FFFFB0 <sub>H</sub>	ICR04	0000B4 <sub>H</sub>
ICU #2	Δ	#20	14 <sub>H</sub>	FFFFAC <sub>H</sub>		
External interrupt #3	Δ	#21	15 <sub>H</sub>	FFFFA8 <sub>H</sub>	ICR05	0000B5 <sub>H</sub>
ICU #3	Δ	#22	16 <sub>H</sub>	FFFFA4 <sub>H</sub>		
16-bit timer overflow	○	#23	17 <sub>H</sub>	FFFFA0 <sub>H</sub>	ICR06	0000B6 <sub>H</sub>
Time base timer interval interrupt	○	#25	19 <sub>H</sub>	FFFF98 <sub>H</sub>	ICR07	0000B7 <sub>H</sub>
Reload timer #0 overflow	○	#27	1B <sub>H</sub>	FFFF90 <sub>H</sub>	ICR08	0000B8 <sub>H</sub>
Reload timer #1 overflow	○	#29	1D <sub>H</sub>	FFFF88 <sub>H</sub>	ICR09	0000B9 <sub>H</sub>
–	×	–	–	–	ICR10	0000BA <sub>H</sub>
A/D	○	#33	21 <sub>H</sub>	FFFF78 <sub>H</sub>	ICR11	0000BB <sub>H</sub>
Simple I/O serial #1	○	#35	23 <sub>H</sub>	FFFF70 <sub>H</sub>	ICR12	0000BC <sub>H</sub>
UART #0 send completed	○	#37	25 <sub>H</sub>	FFFF68 <sub>H</sub>	ICR13	0000BD <sub>H</sub>
UART #0 receive completed	⊙	#39	27 <sub>H</sub>	FFFF60 <sub>H</sub>	ICR14	0000BE <sub>H</sub>
Delay interrupt	×	#42	2A <sub>H</sub>	FFFF54 <sub>H</sub>	ICR15	0000BF <sub>H</sub>
Stack fault	×	#255	FF <sub>H</sub>	FFFC00 <sub>H</sub>	–	–

**[CAUTION]** For information about interrupt relation to EI2OS, see Table 2.2.3.

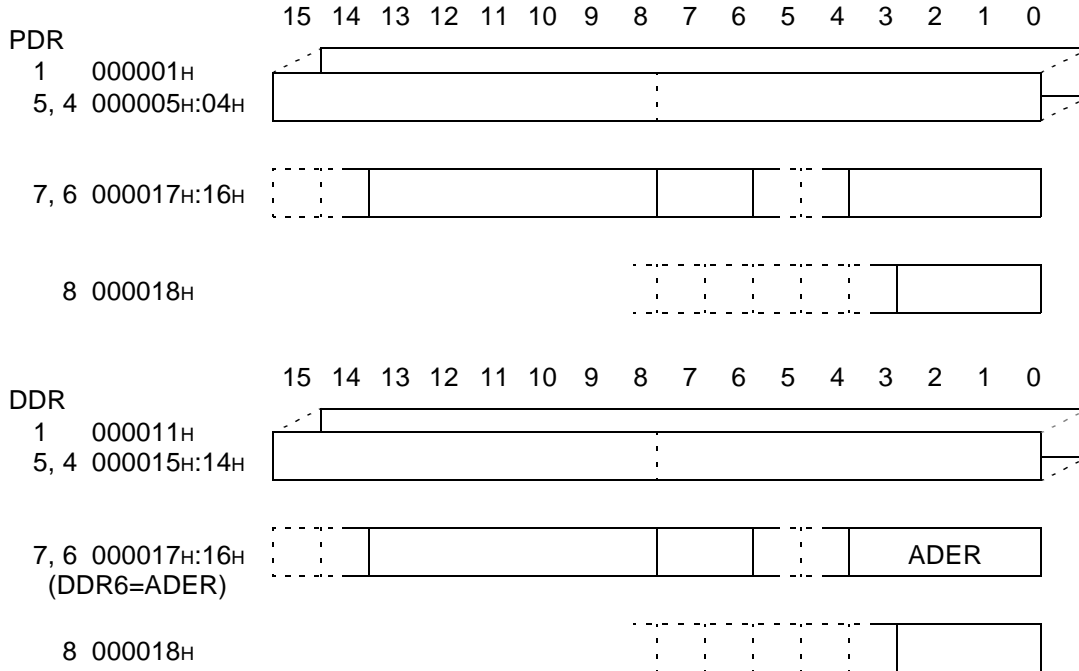
**Table 2.2.3 Description of Interrupt Relation to EI<sup>2</sup>OS**

Symbol	Relation to EI <sup>2</sup> OS	Relation to stop request
⊙	Related	Yes
○	Related	No
Δ	Related, however 2 interrupts are allocated to one ICR, so that when EI <sup>2</sup> OS is used with relation to one interrupt source it is not possible to use both EI <sup>2</sup> OS and a normal interrupt with relation to the other factor.	No
×	No	–

## 2.3 Parallel Ports

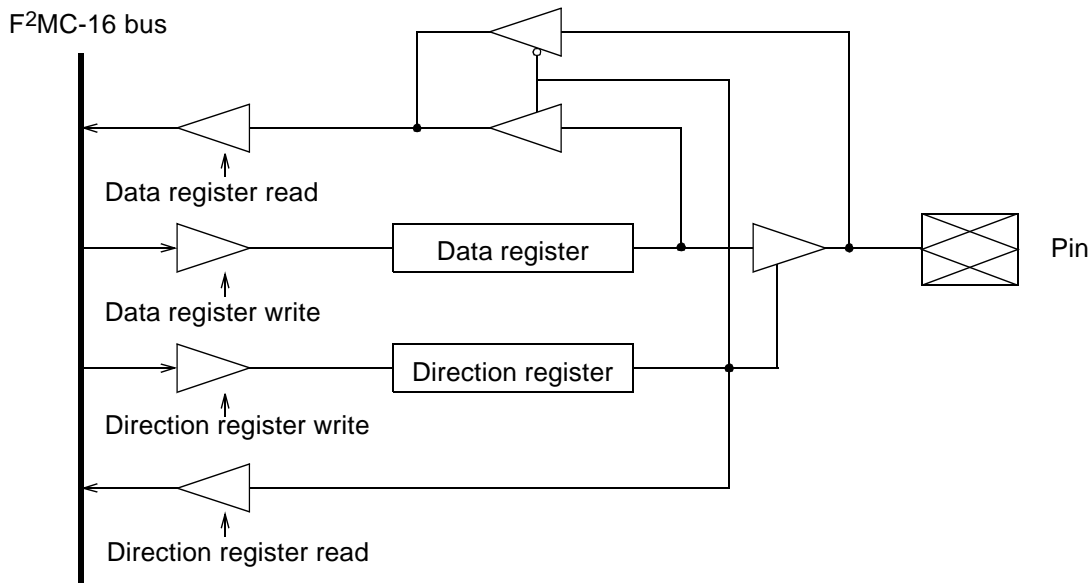
The MB90242A series microcontroller provides 38 I/O ports.

### 2.3.1 Register List



### 2.3.2 Block Diagrams

■ I/O Port



**Fig. 2.3.1 I/O Port Block Diagram**

■ Open Drain Port

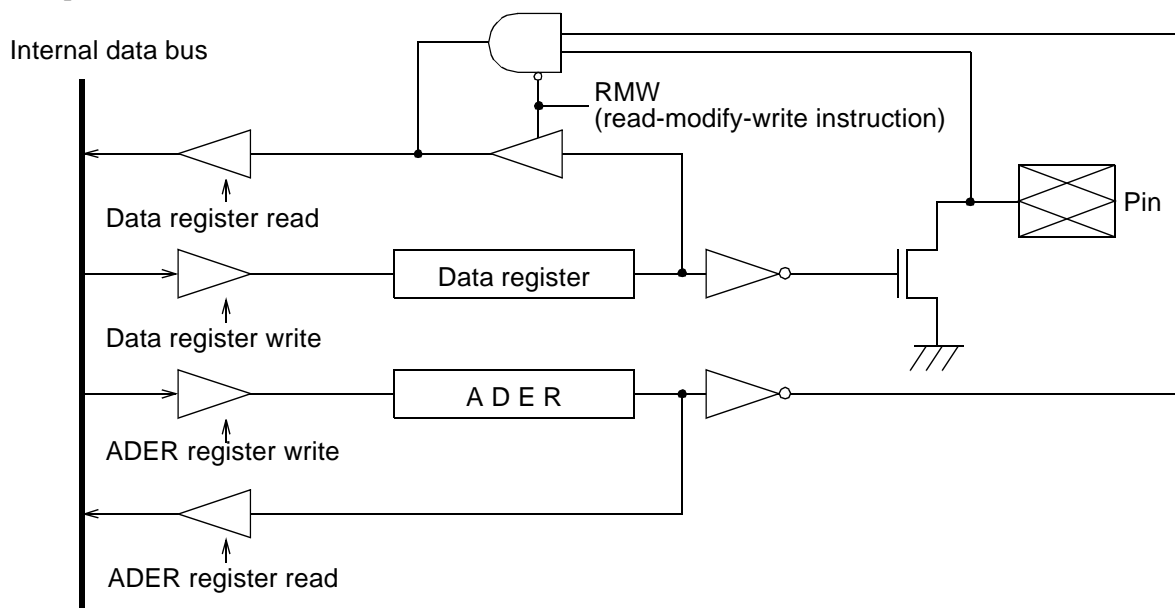


Fig. 2.3.2 Open Drain Port Block Diagram

### 2.3.3 Detailed Register Descriptions

#### (1) PDR 1, 4, 5, 6, 7, 8 (Port data registers)

■ Register Allocation

Port data register

Address : PDR1 000001H	15	14	13	12	11	10	9	8	←Bit no.
PDR5 000005H	PDx7	PDx6	PDx5	PDx4	PDx3	PDx2	PDx1	PDx0	
PDR7 000007H									
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)	

**[CAUTION]** Bits 7 and 6 (PD77, PD76) for port 7 have no register bits.

Port data register

Address : PDR4 000004H	7	6	5	4	3	2	1	0	←Bit no.
PDR6 000006H	PDx7	PDx6	PDx5	PDx4	PDx3	PDx2	PDx1	PDx0	
PDR8 000008H									
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)	
	(X)	(1)	(X)	(X)	(1)	(1)	(1)	(1)	

**[CAUTION]** Bits 5 and 4 (PD65 to PD64) for port 6 have no register bit.

Bits 7 and 6 (PD77 to PD76) for port 7 have no register bit.

Bits 7 to 3 (PD87 to PD83) for port 8 have no register bit.

■ Register Description

All ports other than port 6 are provided with individual direction registers that can be used to set their signal pins to input or output when the output pins for corresponding peripheral resource are not in use. When in input mode, data registers are read from the pin signal levels, while in output mode data registers are read from the latched data register values. This is true when using read-modify-write instructions.

When data registers are read for use as control output, the values read will be the values used for control output regardless of the setting of the direction registers.



## 2.3 Parallel Ports

**[CAUTION]** When the above registers are accessed using read-modify-write instructions (such as bit set instructions), the targeted bit in the command will be set to the designated value, however for any other bits set for input the designated input value of the signal pin will be overwritten with the contents of the corresponding output register. For this reason, whenever pins used for input are switched to output, it is first necessary to write the desired values in the PDR register before setting the DDR register to switch to output.

**[CAUTION]** The process of reading and writing to I/O ports differs from reading and writing to memory in the following ways.

▷ Input Mode

Reading: The read value is the signal level of the corresponding pin.

Writing: The write data is stored in the output latch, and cannot be output to the pin.

▷ Output Mode

Reading: The read value is the value stored in the PDR register.

Writing: The write data is stored in the output latch, and can also be output to the pin.

**[CAUTION]** Note that read/write operations for port 6 differ from those for other ports.

Port 6 (P67, P66, P63 to P60) is an open-drain type general-purpose I/O port which is designed to also function as an analog input signal port. When used as a general-purpose port, the bit(s) corresponding to the ADER register must be set to '0.'

When used as an input port, the contents of the output data register should be set to 'ALL 1' in order to turn off all open-drain output transistors, and external pull-up resistor should be connected. Also, for read access, one of the following two operations should be followed depending on the instruction used.

▷ When reading using read-modify-write instructions

⇒ Read the contents of the output data register. Even when each pin is externally and forcibly driven to '0' there should be no change, even to contents of pins not designated by the instruction.

▷ For reading using all other instructions

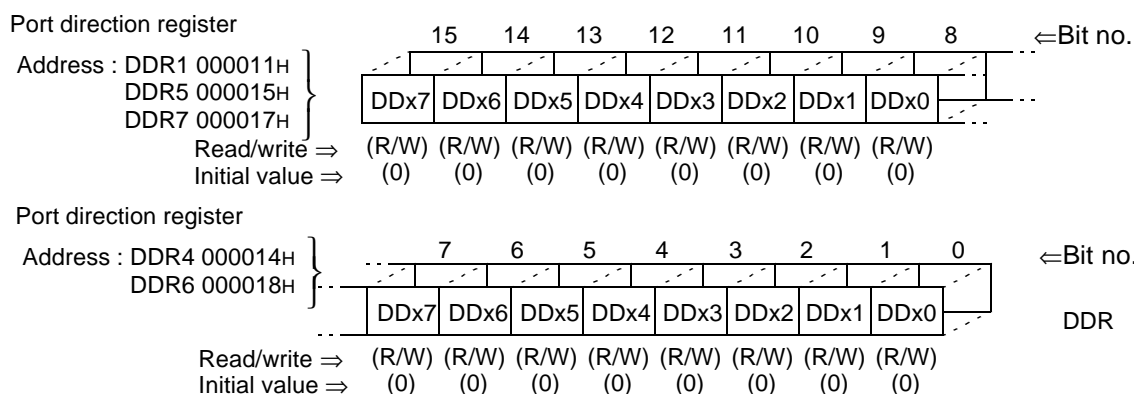
⇒ The level of each signal pin can be read.

When used as an output port, pin values can be changed by writing the desired value to the corresponding output data register.

Also, bits set to '1' in the analog input enable register will always produce the value '0' when read from the corresponding signal pin.

**(2) DDR 1, 4, 5, 7, 8 (Direction registers)**

## ■ Register Allocation



**[CAUTION]** The value '0' must be written to port 5, bit 6 (DD56).  
 Port 7, bits 7 and 6 (DD77 to DD76) have no register bit.  
 Port 8, bits 7 to 3 (DD87 to DD83) have no register bit.

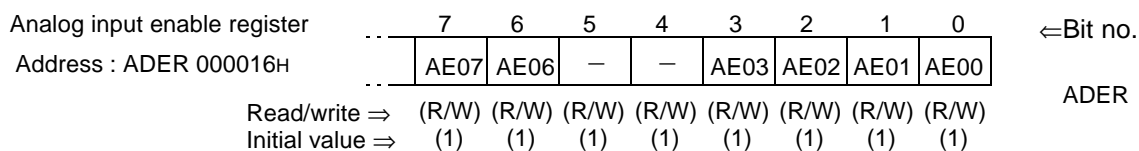
## ■ Register Description

When the corresponding signal pins are functioning as ports, the function of each pin is controlled as follows.

0: Input mode

1: Output mode

DDR registers are initialized to '0' after a reset.

**(3) ADER (Analog input enable register)**

**[CAUTION]** ADER bits 5 and 4 (AE05, AE04) have no register bit.

## ■ Register Description

Port 6 signal pins are controlled as follows.

0: Port input mode

1: Analog input mode

The reset value is '1.'

### 2.3.4 Allocation of Port Pins

In the MB90242A series, the signal pins of ports 0-5 are shared with the external bus. Pin functions can be selected by the bus mode and register settings. Table 2.3.1 shows the allocation of port pins in each mode.

**Table 2.3.1 Port Pin Allocation by Mode**

Pin	Function	
	External data bus 8-bit mode	External data bus 16-bit mode
P07 to P00	D07 to D00	
P17 to P10	Port	D08 to D15
P27 to P20	A07 to 00	
P37 to P30	A15 to 08	
P47 to P40	A23 to A16 Note 1	
P57	Port 57	
P56	RDX	
P55	WRX Note 2	WRLX Note 2
P54	Port	WRHX Note 2
P53	HRQ Note 2	
P52	HAKX Note 2	
P51	RDY Note 2	
P50	CLK Note 2	

- Pins indicated by Note 1 can be used as I/O ports by a setting in the upper address control register.
- Pins indicated by Note 1 (P41 to P47) are shared with peripheral resources (UART, ICU, DTP, TIMER). Users should note that use of each port as resource I/O operations reduces the amount of available address space.
- Pins indicated by Note 2 can be used as I/O ports by setting the external pin control registers.

## 2.4 IIR Filter DSP Unit

This DSP unit provides hardware calculation of sum-of-products calculations ( $\sum B_i * Y_j + \sum A_m * X_n$ ). It can be used for fast, easy IIR filter calculations.

- Features

Coefficients A and B, and variables X and Y are 16-bits x 4 banks.

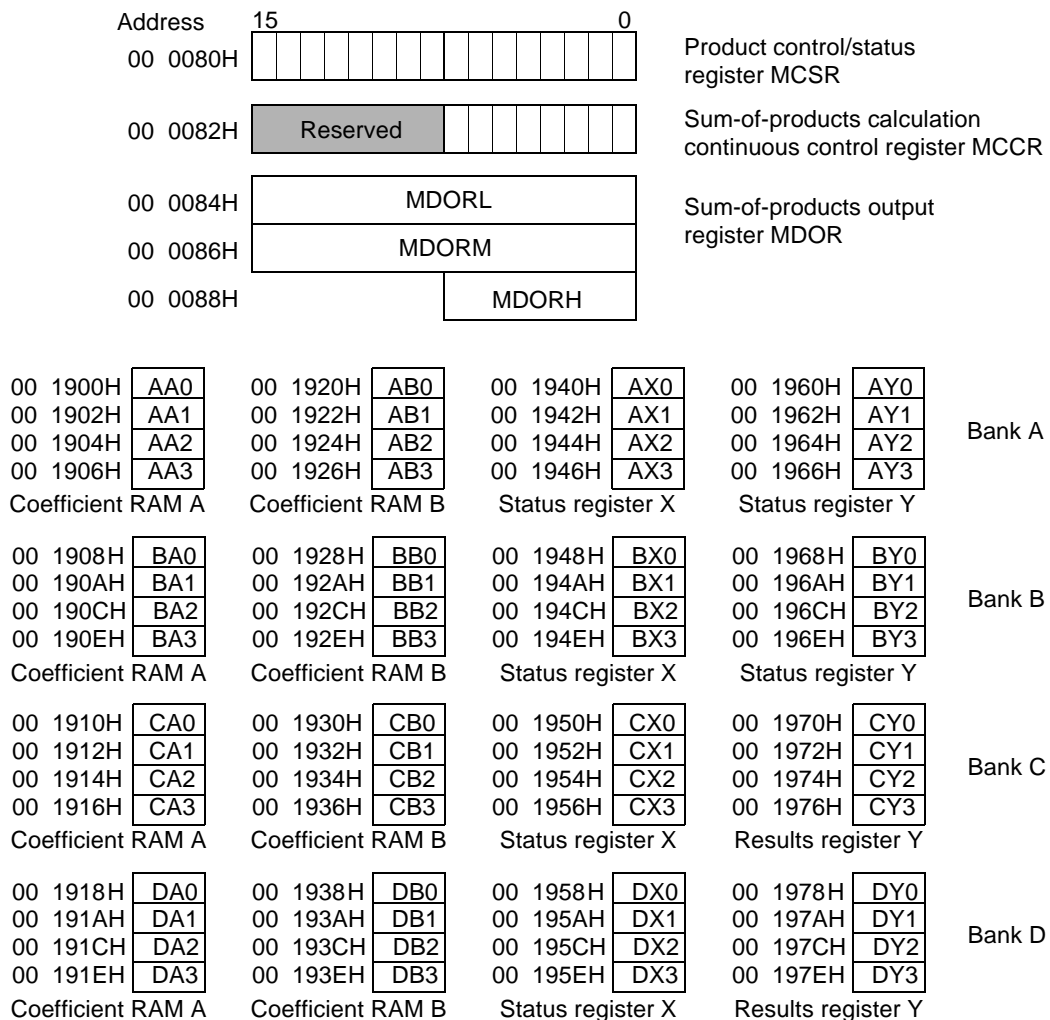
The number of products can be selected in the range (1 to 4) + (1 to 4).

Rounding and clipping functions can be selected at 10-bit or 12-bit length.

Multiple banks can be linked, allowing results of calculations to be transferred into registers in the next bank.

Calculation time is  $(M+N+1) \times B + 1$  machine cycles (where M, N=number of products, B=number of banks).

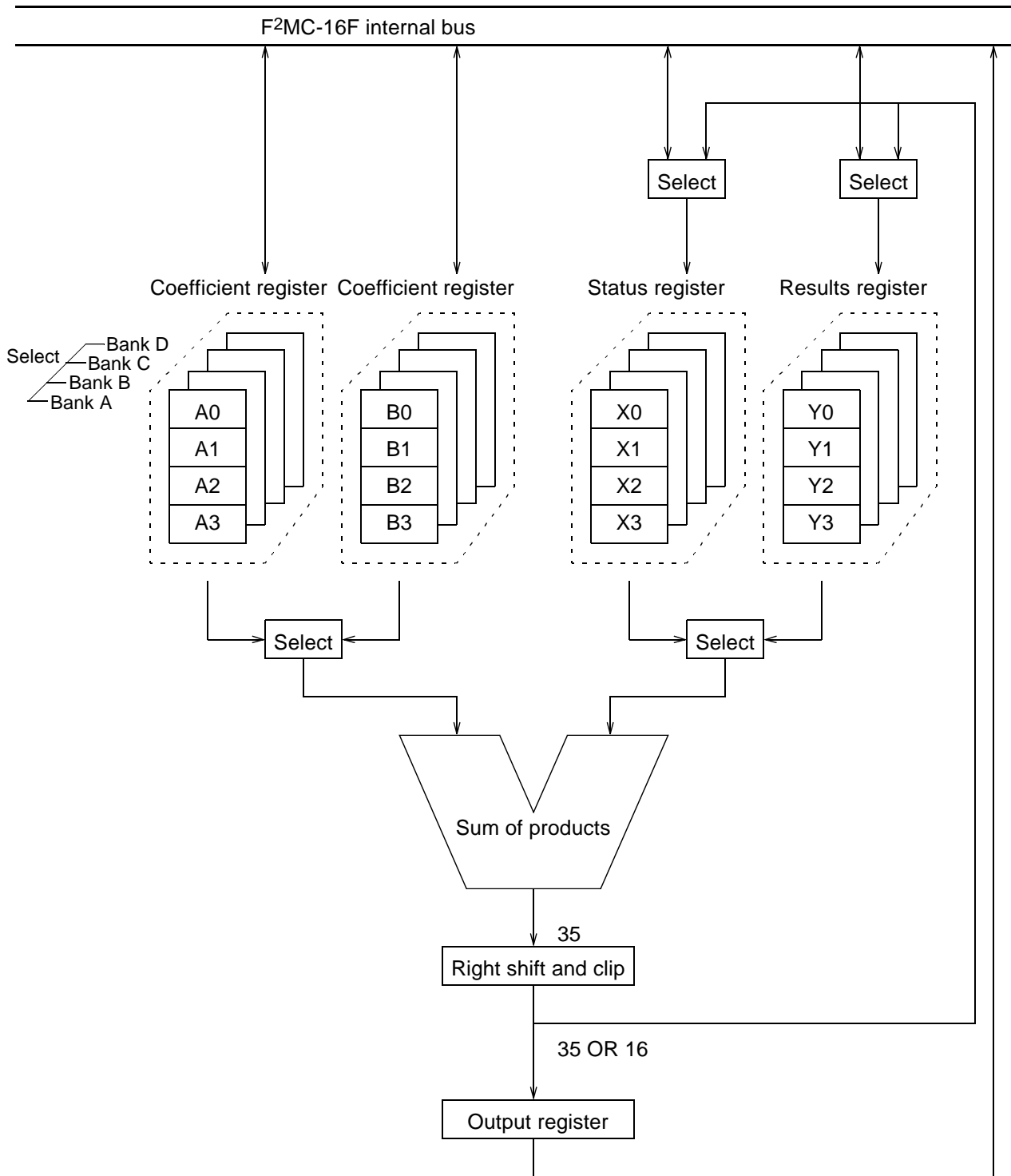
### 2.4.1 Register List



**Fig. 2.4.1 DSP Unit Registers**

**Note:** Addresses in registers X and Y are valid only for reading. Write operations are directed to the 0 registers regardless of address within banks.

**2.4.2 Block Diagram**



**Fig. 2.4.2 DSP Unit Block Diagram**

### 2.4.3 Detailed Register Description

#### (1) MCSR (Sum-of-products control status register)

Address 00 0081 H	15	14	13	12	11	10	9	8	Initial value -XXX XXXX B
	–	WEY	WENY	WENX	N 1	N 0	M 1	M 0	
		R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address 00 0080 H	7	6	5	4	3	2	1	0	Initial value XXX0 XXX0 B
	RND	CLP	DIV	BF	BNK1	BNK0	TRG	MAE	
	R/W	R/W	R/W	R	R/W	R/W	W	R/W	

The MCSR register bits 15 to 05, 03 and 02 can not be changed while the DSP unit is executing calculations (when the BF bit=1).

[Bit 15] This bit is not used. Read values are undefined. Any values written are invalid.

[Bit 14] WEY (Write enable bit for Y0 register)

- This bit specifies that after calculation, data is transferred to the Y0 register in the bank in which calculation is executed.
- When this bit is '1' the 16-bit calculation results in the MDORL register are transferred to the Y0 register. When the value is '0,' no data is transferred.
- The initial value of this bit is undefined. It must always be set prior to calculation.
- This bit cannot be written to while the DSP unit is executing calculations (when the BF bit=1).

[Bit 13] WENY (Write enable bit for next Y0 register)

- This bit specifies that after calculation, data is transferred to the next Y0 register following the bank in which calculation is executed.
- When this bit is '1,' the 16-bit calculation results in the MDORL register are transferred to the Y0 register. When the value is '0,' no data is transferred. (For example, when BNK1,0=01, data is written to the Y0 register in the C bank.)
- The initial value of this bit is undefined. It must always be set prior to calculation.
- This bit cannot be written to while the DSP unit is executing calculations (when the BF bit=1).
- When BNK1,0=11, calculation results are not transferred to the Y0 register in the A bank.

[Bit 12] WENX (Write enable bit for next X0 register)

- This bit specifies that after calculation, data is transferred to the next X0 register following the bank in which calculation is executed.
- When this bit is '1' the 16-bit calculation results in the register are transferred to the X0 register. When the value is '0,' no data is transferred. (For example, when BNK1,0=01, data is written to the X0 register in the D bank.)
- The initial value of this bit is undefined. It must always be set prior to calculation.
- This bit cannot be written to while the DSP unit is executing calculations (when the BF bit=1).
- When BNK1,0=11, the result of calculations is not transferred to the X0 in the bank A.

[Bits 11 to 10] Register for setting the number of N1, N0 terms (number of terms for B\*Y).

[Bits 9 to 8] Register for setting the number of M1, M0 terms (number of terms for A\*X).

- These bit determine the number of product terms for sum-of-products. See Table 2.4.3b, where (the value for N1,0/M1,0) + 1 represents the number of product terms.

- The initial values for this bit are indeterminate. Define it before operation.
- Nothing can be written to this bit while the DSP unit is executing operation (BF=1).

**Table 2.4.1 Number-of-Products Settings (Bit N1,0/M1,0)**

N1	N0	Number of products	M1	M0	Number of products
0	0	1	0	0	1
0	1	2	0	1	2
1	0	3	1	0	3
1	1	4	1	1	4

[Bit 7] RND (Rounding-off bit)

- When this bit is set to 1, the calculation results are stored in 16-bit length integer format in the lower 16 bits (MDORL) of the sum-of-products output register (MDOR). In this case the data stored in the upper bits of the MDOR register (bit 39 to bit 16) has no significance.
- Also in this case, calculation data in integer format stored in the MDORL register is rounded up (when 1; 0 is dropped), with the DIV bit set for rounding in the direction of positive infinity, in order to prevent loss of computational accuracy. This represents binary processing similar to the normal rounding of values in decimal format.
- When this bit is set to 0, rounding-up will not be applied. Also, calculation results will be stored in the MDOR register, in 16-bit length integer format if CLP=1, and in 35-bit length integer format if CLP=0.
- The initial value of this bit is undefined. It must always be set prior to calculation.
- This bit cannot be written to while the DSP unit is executing calculations (when the BF bit=1).

[Bit 6] CLP (Clipping bit)

- When this bit is set to 1, the results of calculation are stored in 16-bit integer format in the lower 16-bits (MDORL) of the sum-of-products output register (MDOR). In this case the data stored in the upper bits of the MDOR register (bit 39 to bit 16) has no significance.
- Also in this case, any overflow condition created in the calculation data in integer format contained in the MDORL register will cause that data to be forcibly replaced by the maximum positive or negative value. Thus either saturation processing or clip processing will be performed on the results of the calculation. The maximum positive value is 7FFF<sub>H</sub>, and the maximum negative value is 8000<sub>H</sub>.
- When this bit is set to 0, clipping processing will not be applied. Also, calculation results will be stored in the MDOR register, in 16-bit length integer format if RND=1, and in 35-bit length integer format if RND=0.
- The initial value of this bit is undefined. It must always be set prior to calculation.
- This bit cannot be written to while the DSP unit is executing calculations (when the BF bit=1).

[Bit 5] DIV (Divide bit)

- This bit indicates the position at which rounding or clipping are applied to calculation results.
- When CLP=1 or RND=1, this bit can be set to 1 to round or clip the LSB value to 12 bits. When this bit is set to 0, the LSB will be rounded or clipped to 10 bits. (When rounded to 10 bits, data having a '1' in the 10th bit only (hex value 0200<sub>H</sub>) will be added to the calculation results.)
- The initial value of this bit is undefined. It must always be set prior to calculation.

- This bit cannot be written to while the DSP unit is executing calculations (when the BF bit=1).

[Bit 4] BF (Busy flag)

- This bit is read-only. No value may be written.
- When the BF=1, the sum-of-products calculator is executing calculations, and no write operations are allowed to MCSR register bits 15 to 5, or to the MCCR register.
- When the BF=0, the sum-of-products register is not operating. Read access to the sum-of-products output register (MDOR) should be made when this status is indicated.
- The BF bit is automatically set to 1 when the MAE and TRG bit are set to 1 to start execution of sum-of-products calculation. It returns to 0 when sum-of-products calculation is ended.
- The BF bit is forcibly cleared to 0 when the MAE bit is set to 0.

[Bit 3, 2] BNK1, BNK0 (Bank bits)

- These bits determine the executing bank. For continuous sum-of-products execution over more than one bank, the value should be that of the first bank.

BNK1,0		Bank used
0	0	A
0	1	B
1	0	C
1	1	D

- The initial value of this bit is undefined. It must always be set prior to calculation.
- This bit cannot be written to while the DSP unit is executing calculations (when the BF bit=1).

[Bit 1] TRG (Trigger bit)

- The TRG bit indicates the start of sum-of-product calculator operation.
- When '1' is written to this bit while in sum-of-products enable status (MAE=1), the DSP unit starts sum-of-products calculation, and the busy flag (BF bit) is set to 1 to indicate that execution is in progress.
- Writing '0' to the TRG bit has no effect.
- When MAE=0, or while sum-of-products calculation is in progress (BF=1), writing '1' to the TRG bit has no effect.
- The TRG bit always has the read value '0.'

[Bit 0] MAE (MAC enable bit)

- When this bit is set to 1 and the sum-of-products calculator is not operating (BF=0), the DSP unit is in sum-of-products standby status. In this status, the DSP unit will start sum-of-products calculation when '1' is written to the TRG bit.
- When this bit is set to 0, the DSP unit is in sum-of-products stop status. It is also possible to forcibly stop sum-of-products calculations even during execution by writing 0 to the MAE bit. This will cause the DSP unit to cut off sum-of-products calculation and clear the BF bit.
- This bit is initialized to 0 after reset.



**(2) MCCR (Sum-of-products control/continue register)**

Address 00 0083 H	15	14	13	12	11	10	9	8	Initial value ---- --00 B
	-	-	-	-	-	-	Reserved	Reserved	
							R/W	R/W	

Address 00 0082 H	7	6	5	4	3	2	1	0	Initial value 0000 0000 B
	OVF	CNTD	CNTC	CNTB	CDRD	CDRC	CDRB	CDRA	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

[Bit15-10] These bits are not used. Read values are undefined. Any values written are invalid.

[Bit 9, 8] These bits are reserved. The reset value is '0.' Write values must always be '0.'

[Bit 7] OVF (OVer Flow bit)

- This is the overflow bit. The OVF bit is set to '1' to indicate overflow in the results of 16-bit (DIV bit) sum-of-products calculation. Note that this bit will not be set to '1' when RND=DLP=0. The value is reset to 0 by executing a reset, or by writing '0' to this bit.
- This bit has a read value of '1' for read-modify-write instructions.

[Bit 6, 5, 4] CNTD, CNTC, CNTB (CoNTinue bits)

- The value '1' is written to these bits to specify that completion of a calculation is followed by continuous calculations.
- Each bit can be used to execute calculations on successive banks.
- The CNTB bit signifies that calculations on A bank will be followed by calculations on B bank.
- The CNTC bit signifies that calculations on B bank will be followed by calculations on C bank.
- The CNTD bit signifies that calculations on C bank will be followed by calculations on D bank.
- Thus, the CTNB, CTNC and CTND bits may be used in combination to designate various types of processing as follows.

A→B→C→D (CNTB=CNTC=CNTD=1)

A→B→C, D (CNTB=CNTC=1, CNTD=0)

A→B, C→D (CNTB=CNTD=1, CNTC=0)

A, B→C→D (CNTB=0, CNTC=CNTD=1)

- The CNTB-CNTD bits are set to 0 by a reset or by writing '0.'
- The CNTB-CNTD bits should not be changed during execution.
- Each of these bits has a read value of '1' for read-modify-write instructions.

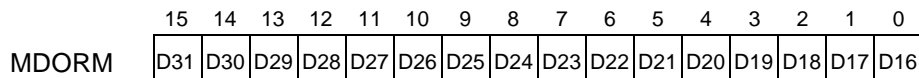
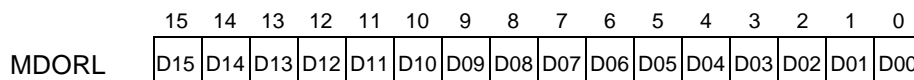
[Bit 3, 2, 1, 0] CDRA-CDRD (Calculated data ready)

- The CDRA-CDRD bits are set to 0 by a reset or by writing '0.'
- The value '1' is not valid when written to the CDRA-CDRD bits. When sum-of-products calculations in a bank are ended, the DSP unit automatically writes '1' to the corresponding bit in the corresponding bank. In continuous operation, each of the CDR bits is set when the corresponding calculation is completed.

**(3) MDOR (Sum-of-products output register)**

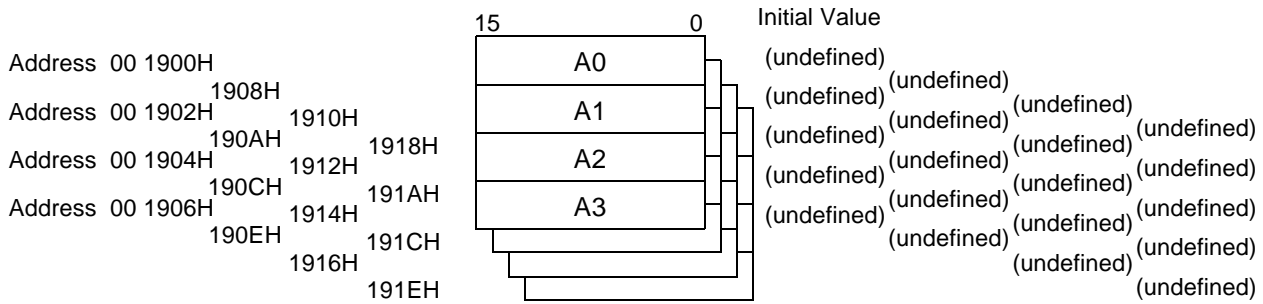
Address 00 0084 H	15	8 7	0	Initial value
	MDORL			(undefined)
Address 00 0086 H	MDORM			(undefined)
Address 00 0088 H	MDORH			(undefined)

- This register contains the results of sum-of-products calculations, and is read-only. All initial values are undefined.
- This register is overwritten by the DSP unit upon completion of sum-of-products calculations, and is used to read the results of calculations after verification of the completed calculation. Read access should be attempted after the corresponding CDR bit (for example, the CDRA bit when BNK1,0 bits=00) is set to 1.
- Sum-of-products calculation results may be either 16-bit or 35-bit length.
- When either the RND bit or CLP bit is set to 1, the results will be in 16-bit integer format. In this case, the calculation results are stored in the MDORL bit (the lower 16-bits of the MDOR register), and the MODRM and MDORH bits will contain invalid data.
- When both the RDN bit and CLP bit are '0,' the results will be in 35-bit length, and will be stored in the MDORL (the lower 16-bits of the MODR register), the MODRM (the middle 16-bits of the MDOR register) and the MDORH (the upper 3-bits of the MODR register). The configuration consists of a 34-bit integer portion and a 6-bit coded portion, as shown below. (D34 has the same value as S flag).



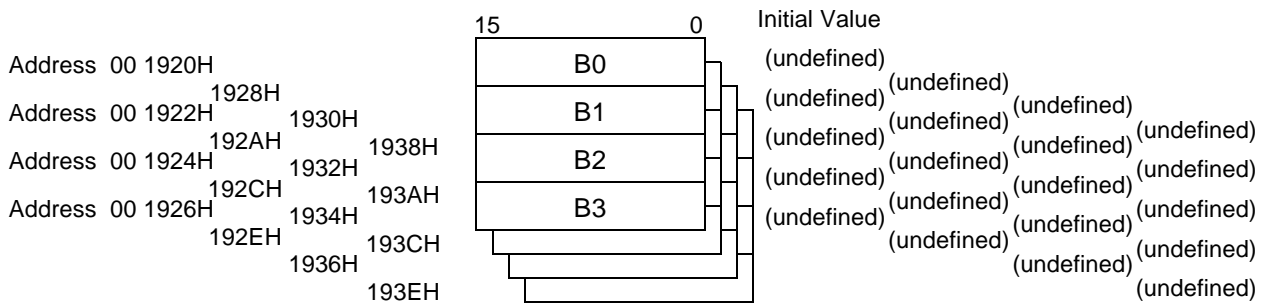
- When the RND or CLP bit is '1', the calculation results can be divided by 1024 or 4096 using the DIV bit, and stored in the MDORL.

**(4) A0-A3 (Coefficient RAM)**



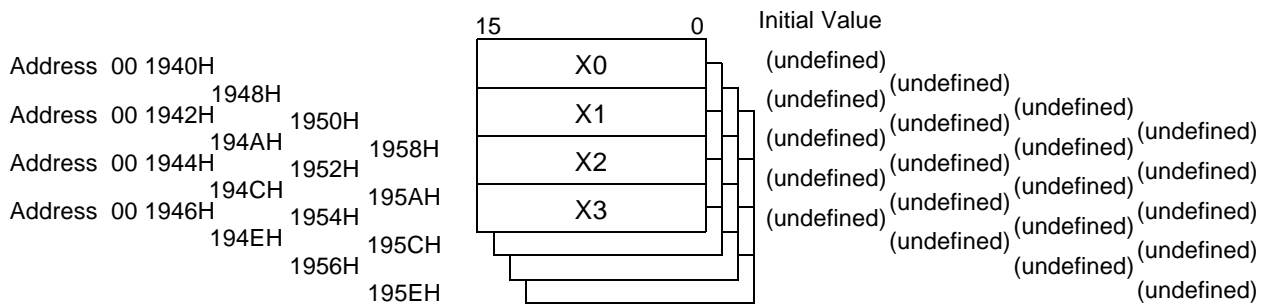
- This register consists of a group of RAM with 4-word configuration from A0 to A3, and contains coefficients for sum-of-products calculation.
- The ARAM register contains coefficient data in 16-bit length integer format.
- The ARAM register has 4 banks of 4 words each, with a bank bit to determine which bank is used in sum-of-products calculation.
- ARAM initial values are undefined.
- This RAM cannot be simultaneously accessed from both the DSP and the CPU, so that reading and writing should not be attempted during execution. If read or write access is attempted during execution, neither the contents of RAM nor the calculation results are assured.
- This RAM is not enabled for byte access. Always access in word units at even-numbered addresses.

**(5) B0 to B3 (Coefficient RAM)**



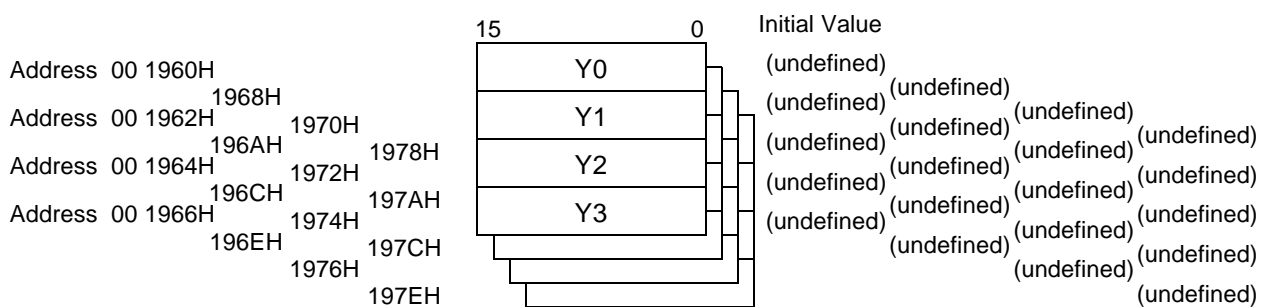
- This register consists of a group of RAM with 4-word configuration from B0 to B3, and contains coefficients for sum-of-products calculation.
- The BRAM register contains coefficient data in 16-bit length integer format.
- The BRAM register has 4 banks of 4 words each, with a bank bit to determine which bank is used in sum-of-products calculation.
- BRAM initial values are undefined.
- This RAM cannot be simultaneously accessed from both the DSP and the CPU, so that reading and writing should not be attempted during execution. If read or write access is attempted during execution, neither the contents of RAM nor the calculation results are assured.
- This RAM is not enabled for byte access. Always access in word units at even-numbered addresses.

**(6) X0-X3 (Input data register)**



- This register consists of a group of registers with 4-word configuration from X0 to X3, and contains input data to be used for sum-of-products calculation.
- The X register contains input data in 16-bit length integer format.
- The X register has 4 banks of 4 words each, with a bank bit to determine which bank is used in sum-of-products calculation.
- X initial value register is undefined.
- Write access to the X register is by means of a shift register, so that only the X0 area is write-enabled. Each time data is written to X0, the previous data is shifted to X1 through X3 in order.
- This register cannot be simultaneously accessed from both the DSP and the CPU, so that reading and writing should not be attempted during execution. If read or write access is attempted during execution, neither the contents of the register nor the calculation results are assured.
- This register is not enabled for byte access. Always access in word units at even-numbered addresses.
- All areas of this register are enabled for read access to each address, however the lower 3 bits of any write values will be ignored (see example).

**(7) Y0 to Y3 (Input data register)**



- This register consists of a group of registers with 4-word configuration from Y0 to Y3, and contains output data to be used for sum-of-products calculation.
- The Y register contains input data in 16-bit length integer format.
- The Y register has 4 banks of 4 words each, with a bank bit to determine which bank is used in sum-of-products calculation.
- Y register initial values are undefined.
- Write access to the Y register is made by means of a shift register, so that only the Y0 area is write-enabled. Each time data is written to Y0, the previous data is shifted to Y1 through Y3 in order.
- This register cannot be simultaneously accessed from both the DSP and the CPU, so that reading

## 2.4 IIR Filter DSP Unit

and writing should not be attempted during execution. If read or write access is attempted during execution, neither the contents of the register nor the calculation results are assured.

- This register is not enabled for byte access. Always access in word units at even-numbered addresses.
- All areas of this register are enabled for read access, however the lower 3 bits of any write values will be ignored (see example).

### Example

Before transfer		After transfer
1940H - - AAAAH		1940H - - 0000H
1942H - - BBBBH	MOVW 1942H, 0000H	1942H - - AAAAH
1944H - - CCCCH		1944H - - BBBBH
1946H - - DDDDH		1946H - - CCCCH

## 2.4.4 Sample Applications

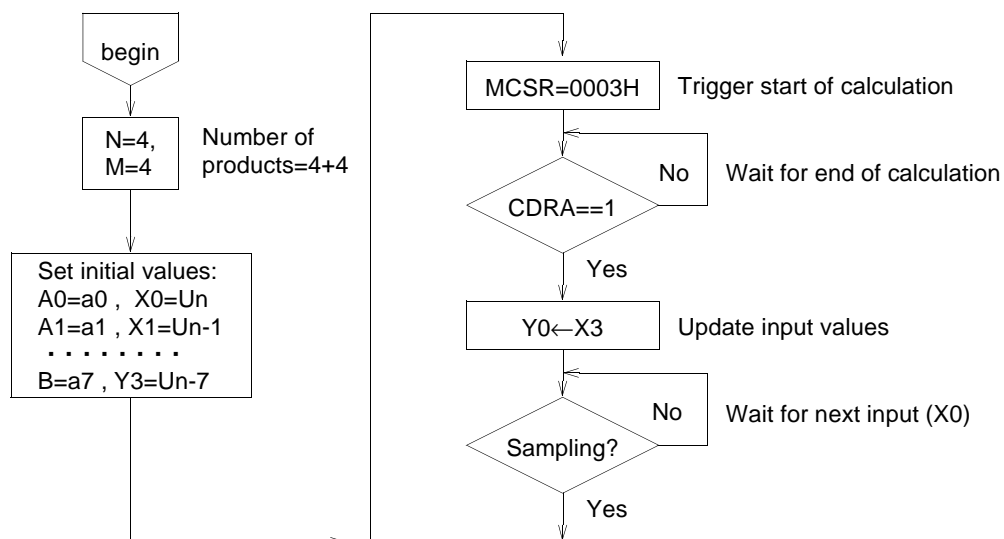
### 2.4.4.1 Eighth Order FIR Filter

The 8-stage FIR filter can be represented by the following formula, in which the input time series is represented by  $[U_{n-2}, U_{n-1}, U_n, \dots]$  and the output time series by  $[V_{n-2}, V_{n-1}, V_n, \dots]$ .

$$V_n = \sum_{i=0}^7 (a_i * U_{n-i})$$

Thus sum-of-products calculation can be performed using coefficient data  $[a_0, a_1, a_2, \dots]$  stored in the coefficient registers  $[A0, A1, A2, A3, B0, B1, B2, B3]$  and input data  $[U_{n-2}, U_{n-1}, U_n, \dots]$  stored in the data registers  $[X0, X1, X2, X3, Y0, Y1, Y2, Y3]$ . (In this case, a maximum of 4 terms is ideal because no shift is made from  $X3 \rightarrow Y0$ .)

The following diagram shows a sample programming flow using the DSP unit.

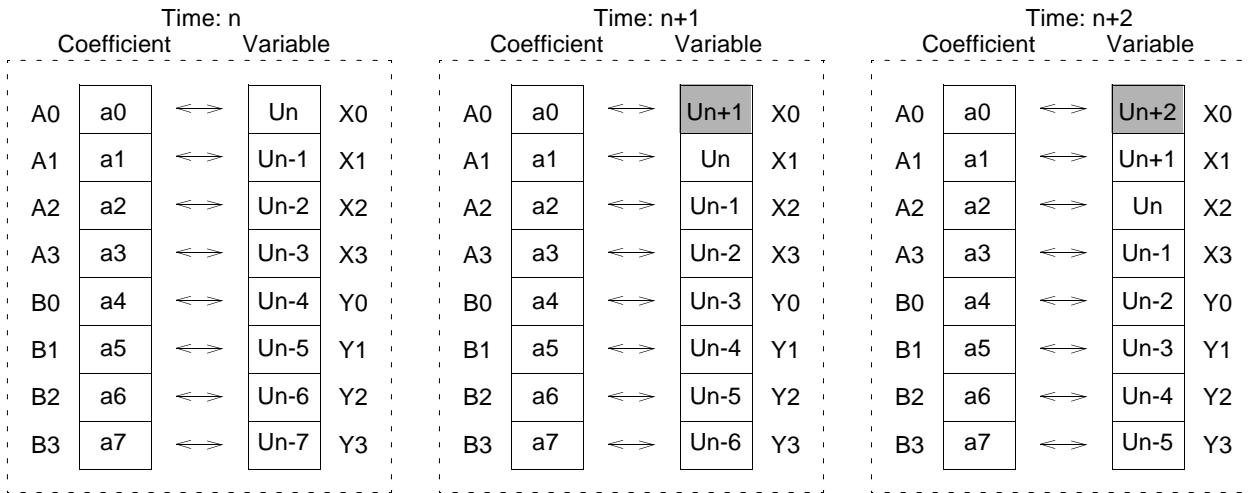


**Fig. 2.4.4a Sample Flow of Calculations in an 8th order FIR Filter**

Further, if we view the contents of the data registers over time during the execution of the above flow of calculations, the result will appear as shown below. This illustration represents a snapshot view of register contents immediately following the end of calculations.

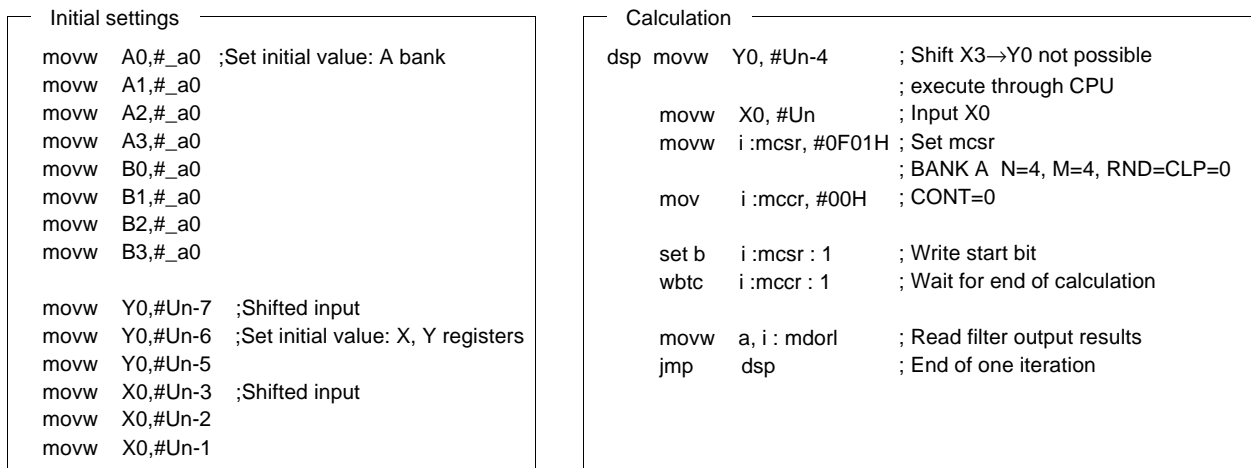
The shaded area of the register shows where the latest contents have been retransferred. Also, the coefficients and variables indicated by the thin two-headed arrows have been multiplied together. Each time the calculation results in the MDOR register are used as output and one transfer is executed, the value in the status register will be shifted by one place.

## 2.4 IIR Filter DSP Unit



**Fig. 2.4.4b Snapshot of Data Registers for 8th order FIR Filter Calculations**

Lastly, the following is an example of a program using the above flow.



**Fig. 2.4.4c Sample Program for 8th order FIR Filter**

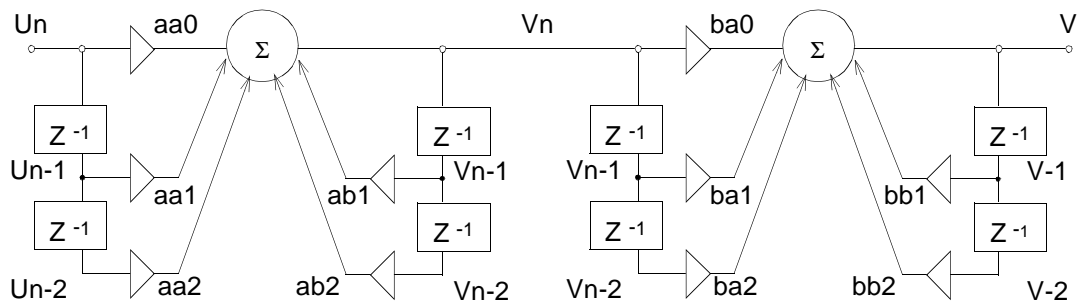
### 2.4.4.2 Biquad Filter (second-order IIR Filter)

Biquad filters are widely used for digital controls. The following sample program illustrates a case of five products (N=2, M=3).

If the input time series is represented by [Un-2, Un-1, Un, ...] and the output time series by [Vn-2, Vn-1, Vn, ...], the biquad filter can be represented by the following formula.

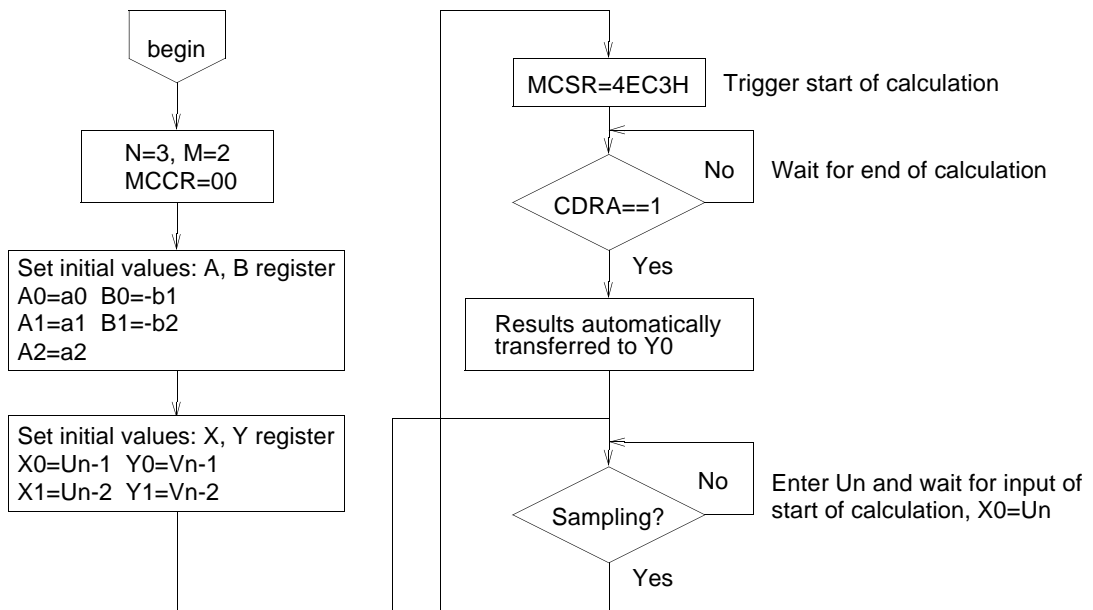
$$V_n = a_0 * U_n + a_1 * U_{n-1} + a_2 * U_{n-2} - b_1 * V_{n-1} - b_2 * V_{n-2}$$

Figure 2.4.4d shows the above formula represented in second order.



**Fig. 2.4.4d Biquad Filter**

Figure 2.4.4e represents the preceding diagram in terms of program flow.

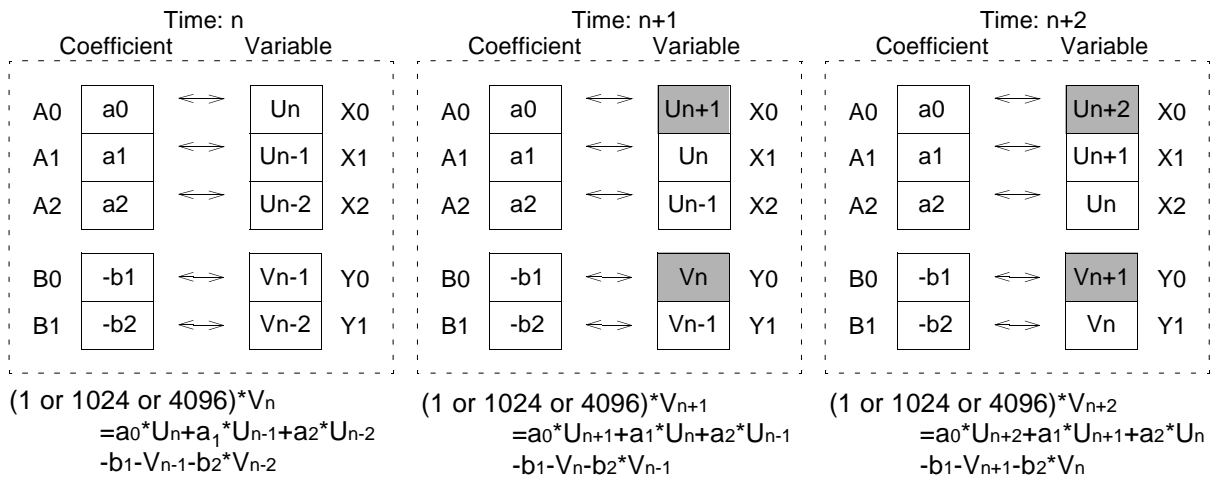


**Fig. 2.4.4e Program Flow for Biquad Filter**

In this case, if we view the contents of the data registers over time during the execution of the above flow of calculations, the result will appear as shown below. This illustration represents a snapshot view of the DSP unit immediately following the end of calculations. The shaded area of the register shows where the latest contents have been retransferred. Also, the coefficients and variables indicated by the thin two-headed arrows have been multiplied together. Each time the calculation results in the MDOR register are used as output and one transfer is executed, the value in the status register will be shifted by one place.

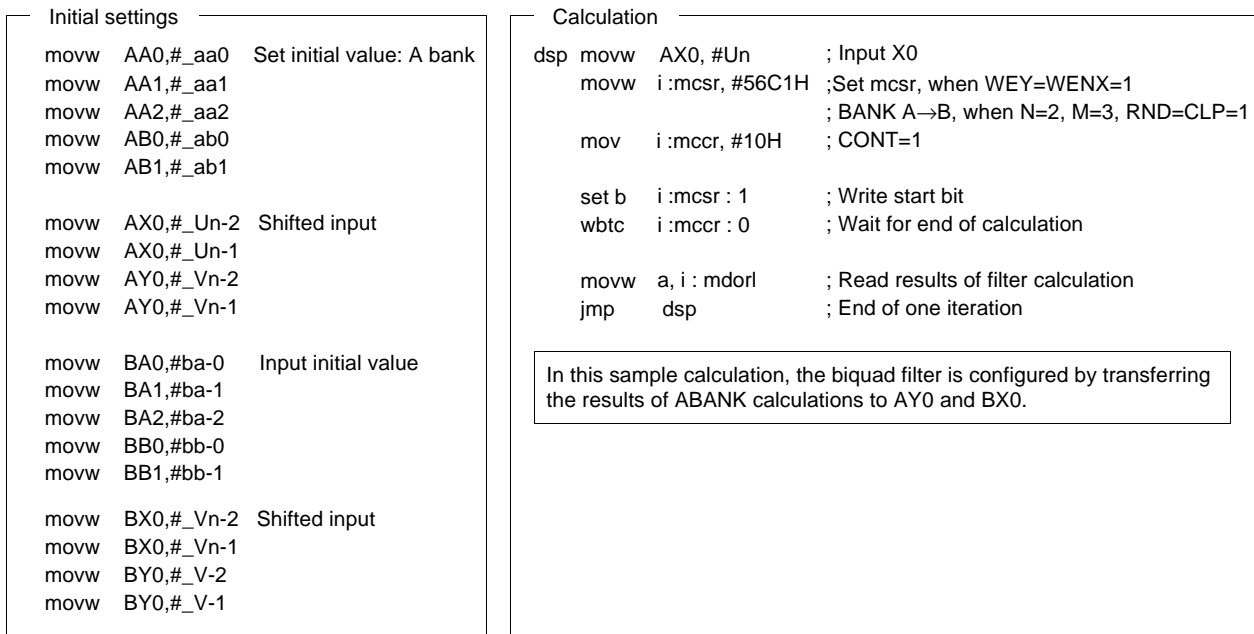


## 2.4 IIR Filter DSP Unit



**Fig. 2.4.4f Snapshot of Biquad Filter Calculations**

Lastly, the following is an example of a program using the above flow.



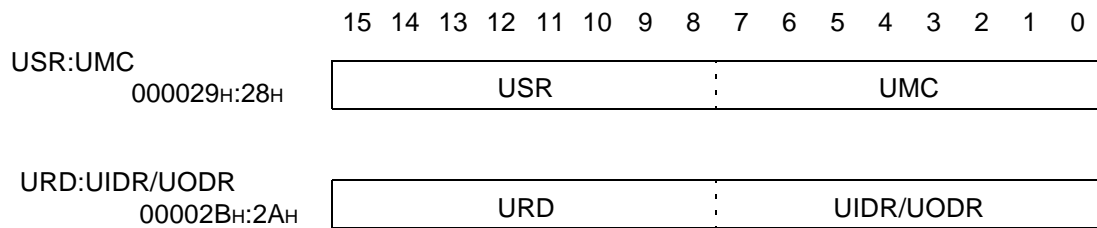
**Fig. 2.4.4g Sample Program for Biquad Filter**

## 2.5 UART

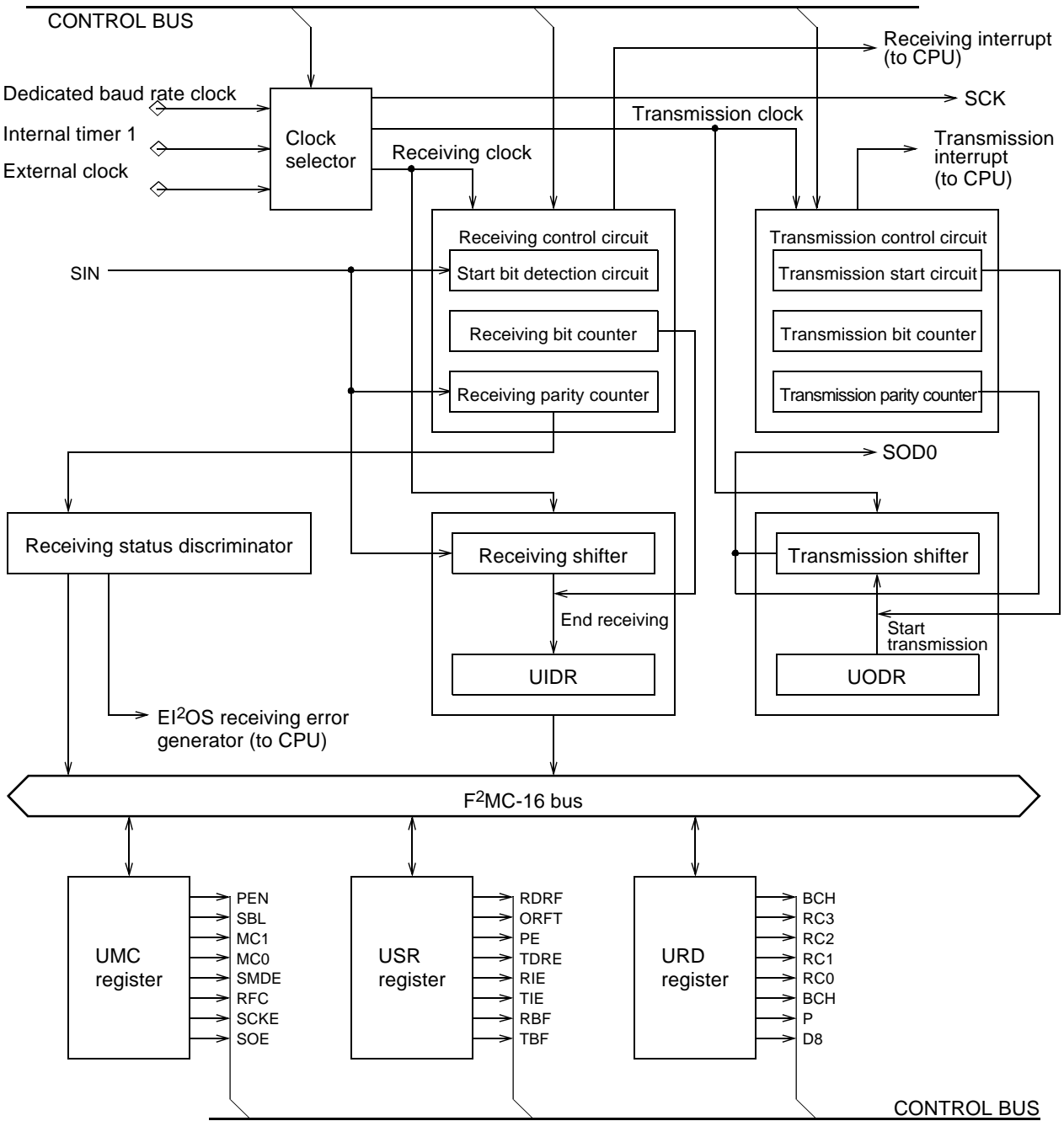
The UART is a serial I/O port for synchronous or asynchronous communication with external circuits, and provides the following features.

- Full-duplex double buffer
- CLK synchronous and CLK asynchronous data transfer capability
- Multiprocessor mode support (mode 2)
- On-chip baud rate generator (12 types)
- Arbitrary baud rate settings from external clock input or internal timer
- Variable data length (7- to 9-bit (no parity), 6- to 8-bit (with parity))
- Error detection (framing, overrun, parity)
- Interrupt functions (2 sources: transmission and receiving)
- NRZ transfer format

### 2.5.1 Register Configuration



### 2.5.2 Block Diagram



**Fig. 2.5.2 Overall Block Diagram**

## 2.5.3 Register Group Descriptions

### (8) UMC (Serial mode control register)

#### ■ Register Allocation

Serial mode control register	7	6	5	4	3	2	1	0	← Bit no.
Address : 000028H	PEN	SBL	MC1	MC0	SMDE	RFC	SCKE	SOE	UMC
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)	(R/W)	(R/W)	
Initial value⇒	(0)	(0)	(0)	(0)	(0)	(1)	(0)	(0)	

#### ■ Register Description

The UMC register sets the operating mode of the UART. Operating mode settings should be entered when the unit is not in operation. The RFC bit, however, may be accessed during operation.

#### ■ Bit Descriptions

[Bit 7] PEN (Parity Enable)

This bit enables the addition (when transmission) or detection (when receiving) of parity bits in serial data input and output. In mode 2 the value should be set to "0."

0: No parity

1: Parity

[Bit 6] SBL (Stop Bit Length)

This bit determines the stop bit length for transmission data. The receiving device always looks only at the first bit of the stop bit, and ignores the second.

0: 1 bit length

1: 2-bit length

[Bits 5, 4] MC1, MC0 (Mode Control)

These bits control the length of transfer data. The two bits in combination are used to select one from four transfer modes (data length) as shown in Table 2.5.1.

**Table 2.5.2 UART Operating Modes**

Mode	MC1	MC0	Data length
0	0	0	7(6)
1	0	1	8(7)
2 Note	1	0	8+1
3	1	1	9(8)

Figures in ( ) indicate data length with parity bit.

**Note:** Mode 2 is used when one host CPU is connected to multiple slave CPUs. In this configuration the receiving parity check function cannot be used, and therefore the PEN bit in the UMC register should be set to "0" (see section 2.5.4 "Operating Description"). Transmission data length is 9 bits, and no parity bit can be attached.

## 2.5UART

### [Bit 3] SMDE (Sync MoDe Enable)

This bit selects the transfer method.

- 0: CLK synchronous transmission
- 1: CLK asynchronous transmission

### [Bit 2] RFC (Receiver Flag Clear)

When 0 is written to this bit, the RDRF, ORFE and PE flags in the USR register are cleared. A write value of "1" is not valid, and the read value is always "1."

### [Bit 1] SCKE (SCLK Enable)

When "1" is written to this bit in CLK synchronous mode, the port signal pins switch to UART serial clock output pins, and output a synchronous clock signal to the outside. In CLK asynchronous mode or in external clock mode, the value "0" should be written to this bit.

- 0: Pins function as general-purpose I/O port, and the serial clock signal output is disabled. With ports in input mode (DDR=0) and the RC3 to RC0 bits set to '1111,' the pins will function as external clock input pins.
- 1: Pins function as UART serial clock output pins.

### [bit 0] SOE (Serial Output Enable)

When 1 is written to this bit, the port pins switch to UART port serial data output pins, and serial data output is enabled.

- 0: Pins function as port pins, and serial data output is disabled.
- 1: Pins function as UART port serial data output pins (SOUT).

## (9) USR (Status register)

### ■ Register Allocation

Status Register	15	14	13	12	11	10	9	8	...	Bit no.
Address : 000029H	RDRF	ORFE	PE	TDRE	RIE	TIE	RBF	TBF	...	USR
Read/write ⇒	(R)	(R)	(R)	(R)	(R/W)	(R/W)	(R)	(R)	...	
Initial value⇒	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	...	

### ■ Register Description

The USR register indicates the current operating status of the UART port.

### ■ Bit Description

#### [bit 15] RDRF (Receiver Data Register Full)

This flag indicates UIDR (input data register) status. It is set when receiving data is loaded into the UIDR register, and is cleared when data is read out from the UIDR register, or when "0" is written to the RFC bit in the UMC register. When the RIE bit is set to 'active' and the RDRF flag is set, an receiving interrupt request is generated.

- 0: Empty
- 1: Data is present

**[Bit 14] ORFE (Over-Run/Framing Error)**

This flag is set when an overrun or framing error occurs during receiving operation, and is cleared when "0" is written to the RFC bit in the UMC register. When this flag is set, data in the UIDR register becomes invalid, and no data can be loaded into the UIDR register from the receiving shifter. When the RIE bit is set to 'active' and the ORFE flag is set, an receiving interrupt request is generated.

0: No error

1: Error

Table 2.5.2 shows the definition of UIDR status when receiving ends using the RDRF and ORFE bits.

**Table 2.5.3 UIDR Register Status at End of Receiving**

RDRF	ORFE	UIDR data status
0	0	Empty
0	1	Framing error
1	0	Normal data
1	1	Overrun error

The occurrence of framing errors or overrun errors invalidates the data in the UIDR register. Normal data receiving can resume after the UIDR register is emptied.

**[Bit 13] PE (Parity Error)**

This bit is set when a parity error occurs during receiving, and is cleared by writing "0" to the RFC bit in the UMC register. When this flag is set, data in the UIDR register becomes invalid, and no data can be loaded into the UIDR register from the receiving shifter. When the RIE bit is set to 'active' and the PE flag is set, a receiving interrupt request is generated.

0: No parity error

1: Parity error

**[Bit 12] TDRE (Transmitter Data Register Empty)**

This flag indicates UODR (output data register) status. It is cleared when transmission data is written to the UODR register, and is set when that data is loaded into the transmission shifter. When the TIE bit is set to 'active' and the TDRE flag is set, a transmission interrupt request is generated.

0: Data present

1: Empty

**[Bit 11] RIE (Receiver Interrupt Enable)**

This bit enables receiver interrupts.

0: Interrupt disabled

1: Interrupt enabled

[Bit 10] TIE (Transmitter Interrupt Enable)

This bit enables transmitter interrupts. When the TDRE bit is set to "1" to enable transmission interrupt requests, interrupts are generated immediately.

0: Interrupt disabled

1: Interrupt enabled

[Bit 9] RBF (Receiver Busy Flag)

This flag indicates that incoming data is being received. It is set when a start bit is detected, and cleared when a stop bit is detected.

0: Receiving stopped

1: Receiving operation in progress

[Bit 8] TBF (Transmitter Busy Flag)

This flag indicates that outgoing data is being transmitted. It is set when transmission data is written to the UODR register, and cleared when transmission ends.

0: Transmission stopped

1: Transmission operation in progress

**(10) UIDR (Input Data Register) / UODR (Output Data Register)**

■ Register Allocation

Input data register / output data register		7	6	5	4	3	2	1	0	⇐ Bit no.
Address :	00002AH	D7	D6	D5	D4	D3	D2	D1	D0	UIDR(read)/ UODR(write)
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)	

■ Register Description

The UIDR (Input data register) is used to contain output serial data. The UODR (output data register) is used to hold output serial data.

When the data length is 6 bits, the upper two bits (D7, D6) contain invalid data, and when the data length is 7 bits, the upper 1 bit (D7) is invalid. The TDRE flag in the USR register should be set to "1" when writing to the UODR register.

**(11) URD (Rate and Data Register)**

■ Register Allocation

Rate and data register	15	14	13	12	11	10	9	8	.. ← Bit no.
Address : 00002BH	BCH	RC3	RC2	RC1	RC0	BCH0	P	D8	URD
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)	..

■ Register Description

The URD register selects the data transfer speed (baud rate) of the UART. When 9-bit data length is used for communication, this register also contains the highest order MSB bit (bit 8). Baud rate selection and parity setting should be made when the unit is not operating.

■ Bit Description

[bits 15, 10] BCH, BCH0 (Baud Rate Clock Change)

These bits switch the baud rate clock in terms of machine rate cycles. (For details, see section 2.5.4 "Operating Description.")

**Table 2.5.4 Clock Input Selection**

BCH	BCH0	Machine cycle
0	0	Not available
0	1	16 MHz
1	0	12 MHz
1	1	10 MHz

[bits 14 - 11] RC3, RC2, RC1, RC0 (Rate Control)

The rate control bits select the UART port clock input signal. (For details, see section 2.5.4 "Operating Description.")

**Table 2.5.4a Clock Input Selection**

RC3 to RC0	Clock input
"0000" to "1011"	Dedicated baud rate generator
"1101"	Internal timer (timer 1)
"1111"	External clock

Note that the settings '1100' and '1110' should not be used.

[bit 9] P

When the parity function is enabled (PEN=1), this bit selects even or odd parity.

0: Even parity

1: Odd parity



[bit 8] D8

When the operating mode is mode 2 or mode 3 (9-bit data length) with no parity, this bit is used as bit 8 in transfer data. When reading data, it is used as bit 8 in the UIDR register, and when writing it is used as bit 8 in the UODR register. In operating modes other than mode 2 or mode 3, this bit is not valid. The TDRE flag in the USR register should be set to "1" when writing to this bit.

## 2.5.4 Operating Description

### (1) Operating Mode

The UART has the operating modes shown in Table 2.5.4. Mode settings can be switched by setting the value in the UMC register.

**Table 2.5.5 UART Operating Modes**

Mode	Parity	Data length	Clock mode	Stop bit length
0	Yes	6	CLK asynchronous or CLK synchronous	1-bit or 2-bit
	No	7		
1	Yes	7		
	No	8		
2	No	8+1		
3	Yes	8		
	No	9		

Note that stop bit length can be set for transmission only. For receiving, the setting is always 1-bit. The unit does not operate in modes other than those shown, and only these settings should be used.

## (2) Clock Selection and Dividing CLK Synchronous/CLK Asynchronous Baud Rate

### a) On-Chip Baud Rate Generator

Twelve baud rate settings can be selected, with CLK synchronous baud rates determined by the following formulas.

$$\text{Baud rate} = \frac{\phi/4}{2^{m-1}} \text{ [bps] (at machine cycle of 16 MHz)}$$

$$\text{Baud rate} = \frac{\phi/3}{2^{m-1}} \text{ [bps] (at machine cycle of 12 MHz)}$$

$$\text{Baud rate} = \frac{\phi/5}{2^{m-1}} \text{ [bps] (at machine cycle of 10 MHz)}$$

Where  $\phi$  represents machine cycle speed, and  $m$  is the decimal equivalent of bits RC3 to RC1 ( $m=2$  to 4).

**Note:** The above formulas do not compute when  $m=0$  or 1.

CLK asynchronous baud rates are transferable in the range -1% to +1%, with baud rates determined by dividing the CLK synchronous baud rate by 8 x 13 or 8 x 12.

Table 2.5.5 shows baud rates at machine cycle of 16 MHz, 12 MHz and 10 MHz. Note that "-" on the table should not be set by the user.

**Table 2.5.6 Baud Rates**

RC3	RC2	RC1	RC0	CLK asynchronous ( $\mu\text{s}/\text{baud}$ )			CLK asynchronous divide ratio	CLK synchronous ( $\mu\text{s}/\text{baud}$ )		
				16MHz	12MHz	10MHz		16MHz	12MHz	10MHz
0	0	0	0	–	–	48/20833	8×12	–	–	–
0	0	0	1	26/38460	26/38460	52/19230	8×13	–	–	–
0	0	1	0	–	–	–	8	–	–	–
0	0	1	1	2/500000	2/500000	4/250000	8	–	–	–
0	1	0	0	48/20833	48/20833	96/10417	8×12	–	–	–
0	1	0	1	52/19230	52/19230	104/9615	8×13	0.5M/2M	0.5M/2M	1M/1M
0	1	1	0	96/10417	96/10417	192/5208	8×12	–	–	–
0	1	1	1	104/9615	104/9615	208/4808	8×13	1M/1M	1M/1M	2/500K
1	0	0	0	192/5208	192/5208	–	8×12	–	–	–
1	0	0	1	208/4808	208/4808	416/2404	8×13	2/500K	2/500K	4/500K
1	0	1	0	–	–	–	8	–	–	–
1	0	1	1	16/62500	16/62500	35/31250	8	–	–	–

b) Internal Timer and External Clock Signals

When bits RC3 to RC0 are set to "1101," the internal timer signal is selected. The reload value setting varies according to the internal timer used. When The RC3 to RC0 bits are set to "1111," the external clock signal is selected.

The CLK asynchronous baud rate is the CLK synchronous baud rate divided by 8. Also, the CLK asynchronous baud rate is transferable in the range of -1% to +1% of the selected baud rate.

Table 2.5.6 shows baud rates with the clock selection set to internal timer. Values on this table are calculated using a machine cycle of 7.3728 MHz. Note that "-" on the table should not be set by the user.

$$\text{Baud rate} = \frac{\phi / X}{8 \times 2(n+1)} \text{ [bps]}$$

$\phi$ : Machine cycle  
 $X$ : Count clock source frequency divide ratio for internal timer used  
 $n$ : Decimal equivalent of reload value

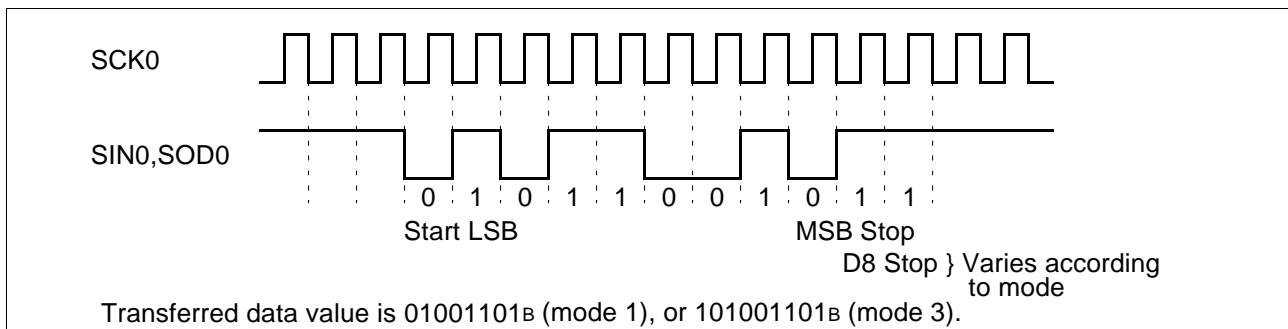
**Table 2.5.7 Baud Rates and Reload Values**

Reload value \ Baud rate	X=2 <sup>1</sup> (machine cycle divided by 2)	X=2 <sup>3</sup> (machine cycle divided by 8)
76800	2	-
38400	5	-
19200	11	2
9600	23	5
4800	47	11
2400	95	23
1200	191	47
600	383	95
300	767	191

The values on this table (in decimal notation) are reload values using internal timers as reload counters.

**(3) Transfer Data Format**

The UART handles only data in NRZ (Non-Return to Zero) format. Figure 2.5.3 shows the relation between the transmit/receive clock and data in CLK synchronous mode.



**Fig. 2.5.3 Transfer Data Format**

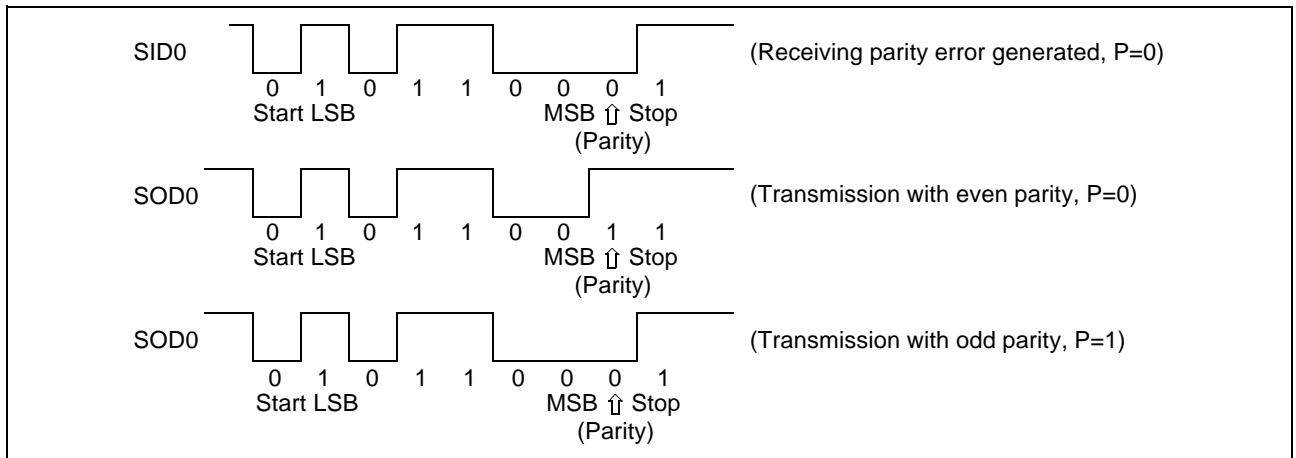
As shown in Figure 2.5.3 "Transfer Data Format," transfer data always begins with a start bit ('L' level data value), then the transfer data at the bit-length designated on LSB-first basis, and ends with a stop bit ('H' level data value). When an external clock signal is selected, the clock should be input at all times. When an internal clock signal (from the dedicated baud rate generator or internal timer) is selected, the clock signal will be output at all times. In CLK synchronous transfer operation, transfer should not begin until the clock has stabilized at the selected baud rate (for two signal periods).

In CLK asynchronous transfer, the SCK0 bit in the UMC register should be set to "0" so that no clock signal is output. The SID0 and SOD0 data transfer formats are the same as shown in Figure 2.5.3 "Transfer Data Format."

**(4) Parity Bit**

When parity is enabled, the P bit in the URD register can be used to designate either even parity or odd parity. Parity is enabled using the PEN bit in the UMC register.

Figure 2.5.4 shows a parity error generated when incoming data at SID0 is received with even parity is designated. Also shown in Figure 2.5.4 is transmission of an outgoing data value of 001101<sub>B</sub> with odd/even parity designated.



**Fig. 2.5.4 Serial Data with Parity Enabled**

**(5) Interrupt Generation and Flag Set Timing**

The UART has 6 flags and 2 interrupt sources. The 6 flags are identified by the names RDRF/ORFE/PE/TDRE/RBF/TBF.

The RDRF flag is set when receiving data is loaded into the UIDR register, and is cleared when data is read out from the UIDR register, or when "0" is written to the RFC bit in the UMC register. The ORFE flag is the overrun/framing error flag, and is set when an overrun or framing error occurs during receiving and cleared when "0" is written to the RFC bit in the UMC register. The PE flag indicates a parity error, and is set when a parity error occurs during receiving, and is cleared by writing "0" to the RFC bit in the UMC register. Note that there is no parity detection function in mode 2. The TDRE flag is set when the UODR register is empty and ready for write, and cleared when transmission data is written to the UODR register.

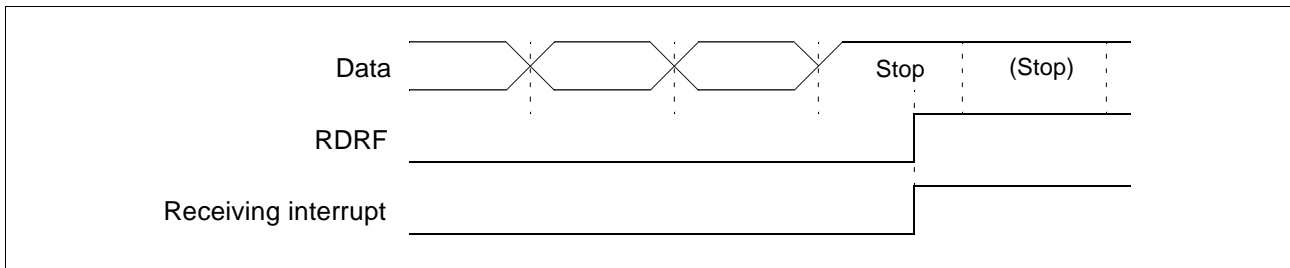
The above 4 flags (RDRF/ORFE/PE/TDRE) function as interrupt factor flags for data transmission and receiving.

The RBF and TBF flags are used to indicate that transmission and receiving are in progress. The RBF flag is active when incoming data is being received, and the TBF flag is active when outgoing data is being transmitted.

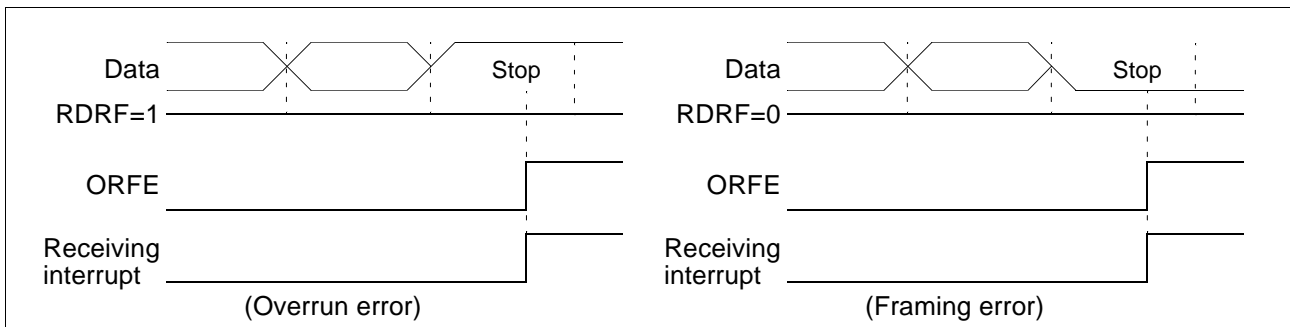
The two interrupt sources are for receiving and transmission, respectively. During receiving, interrupt requests are indicated by the RDRF, ORFE and PE flags. During transmission, the TDRE flag indicates an interrupt request. Flag set timing in each operating mode is shown below.

a) Receiving in Mode 0, Mode 1, Mode 3

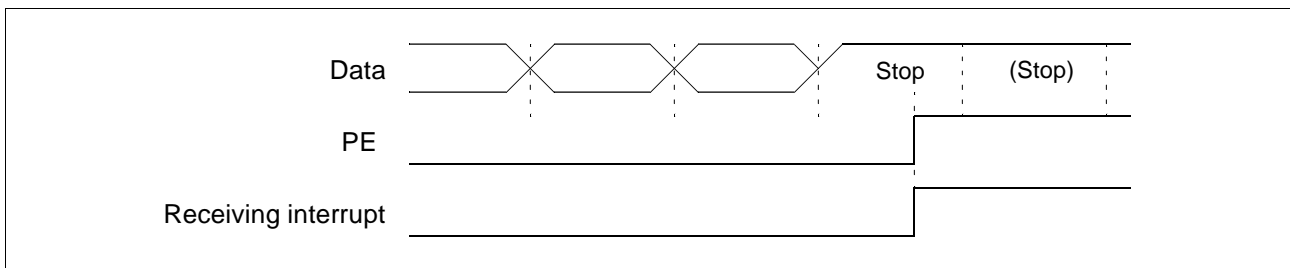
The RDRF, ORFE and PE flags are set, and an interrupt request to the CPU is generated, when receiving transfer has been completed and the last stop bit has been detected. Once the ORFE or PE flag is active, the data in the UIDR register becomes invalid.



**Fig. 2.5.5 RDRF Flag Set Timing (Modes 0, 1, 3)**



**Fig. 2.5.6 ORFE Flag Set Timing (Modes 0, 1, 3)**



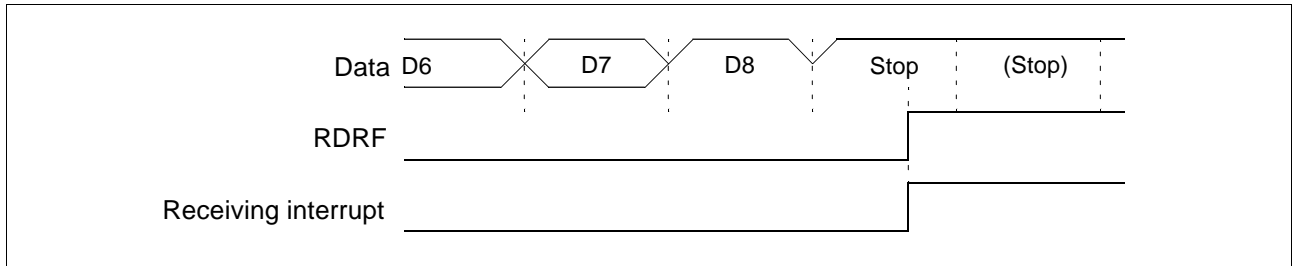
**Fig. 2.5.7 PE Flag Set Timing (Modes 0, 1, 3)**

b) Receiving in Mode 2

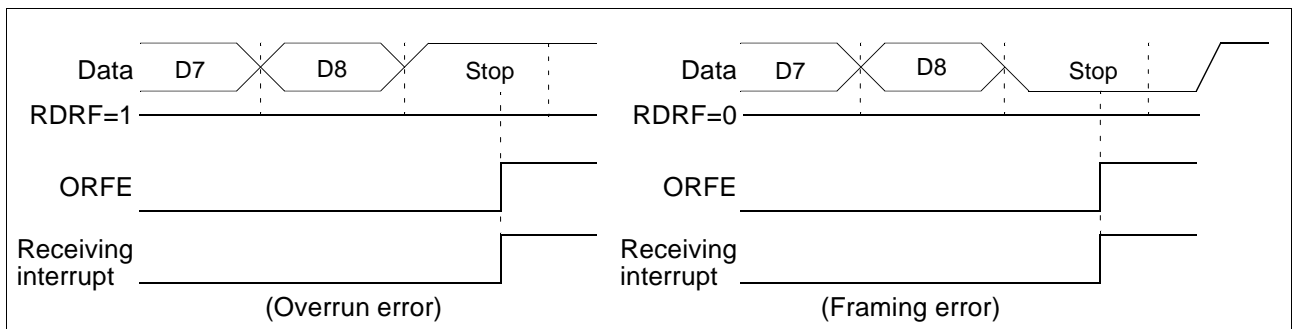
The RDRF flag is set when the last data bit (D8) set to "1" has been received and the last stop bit has been detected.

The ORFE flag is set when the last stop bit has been detected, without regard to the last data bit (D8). Once the ORFE or PE flag is active, the data in the UIDR register becomes invalid.

The interrupt request to the CPU is generated when the flag is set (for use of mode 2 see section 2.5.6. "Sample Applications").



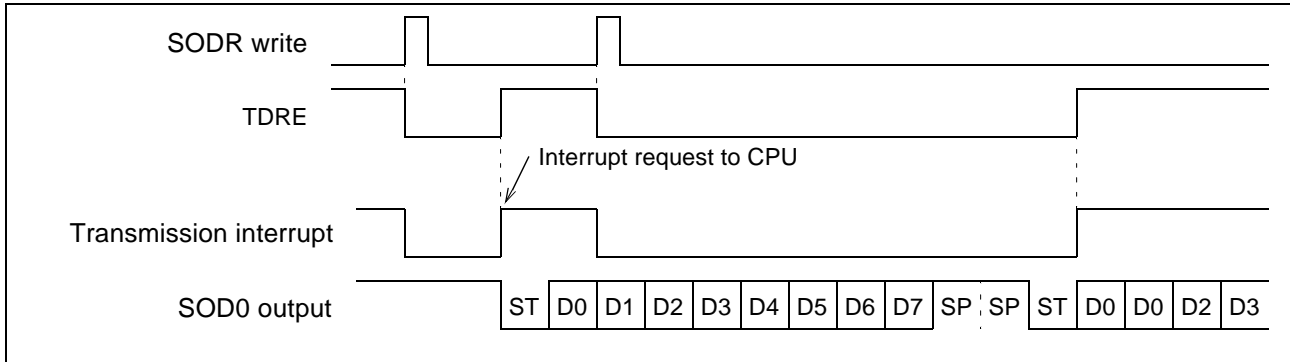
**Fig. 2.5.8 RDRF Flag Set Timing (Mode 2)**



**Fig. 2.5.9 ORFE Flag Set Timing (Mode 2)**

c) Sending

The TDRE flag is set when data written to the UODR register is transferred to the internal shift register, and writing of the next data is enabled. An interrupt request is then generated and sent to the CPU.

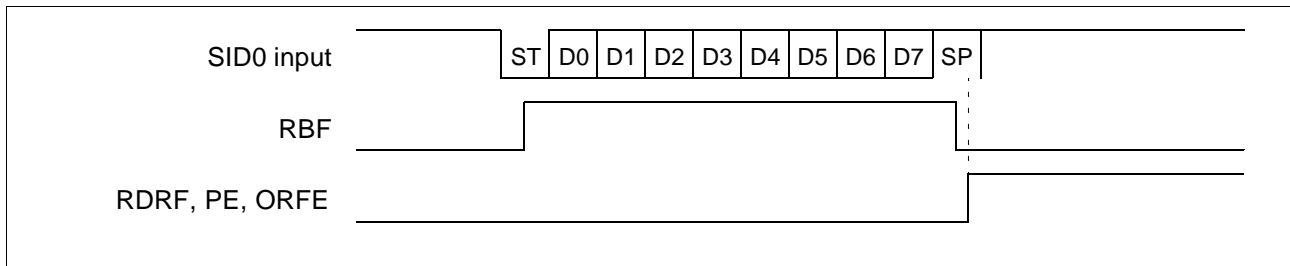


ST: Start bit, D0 to D7: Data bits, SP: Stop bit

**Fig. 2.5.10 TDRE Flag Set Timing (Mode 0)**

d) Status Flag during Transmission

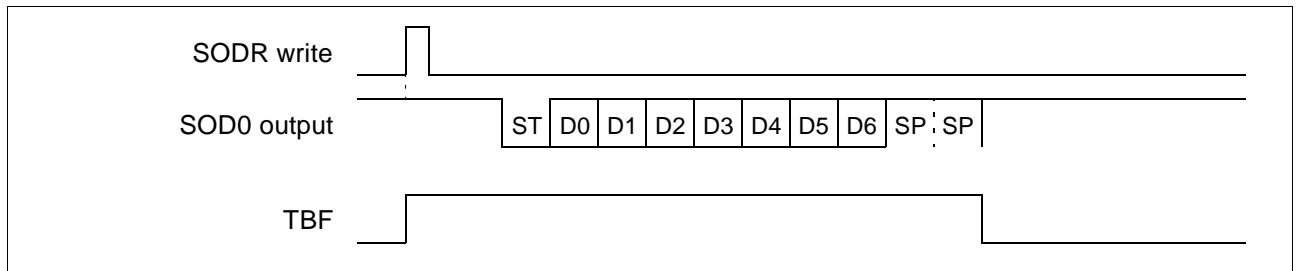
The RBF flag is set when a start bit is detected, and cleared when a stop bit is detected. Once the RBF flag is cleared, transmission data in the UIDR register is no longer correct. At the point that the RDRF flag is set, data in the register is correct data. The relation between the timing of the RBF flag and receiving interrupt flags is shown in figure 2.5.4i.



ST: Start bit, D0 to D7: Data bits, SP: Stop bit

**Fig. 2.5.11 RBF Flag Set Timing (Mode 0)**

The TBF flag is set when transmission data is written to the UODR register, and cleared when transmission ends.



ST: Start bit,                      D0 to D7: Data bits,                      SP: Stop bit

**Fig. 2.5.12 TBF Flag Set Timing (Mode 0)**

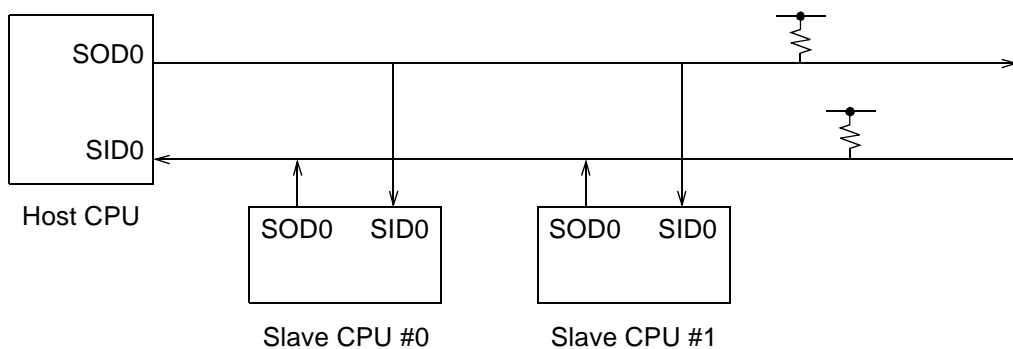
**2.5.5 Notes on Usage**

After release of a reset signal, receiving operations will continue unless the SID0 input pin is fixed to "1." Any receiving flags that were set before determination of the mode setting should therefore be cleared by writing "0" to the RFC bit in the UMC-register.

Transmission mode settings should be made when the RBF/TBF flags in the SSR register are set to "0." Data received during mode settings is not assured.

**2.5.6 Sample Application**

Mode 2 is used when one host CPU is connected to multiple slave CPUs (see Figure 2.5.11 "RBF Flag Set Timing (Mode 0)").

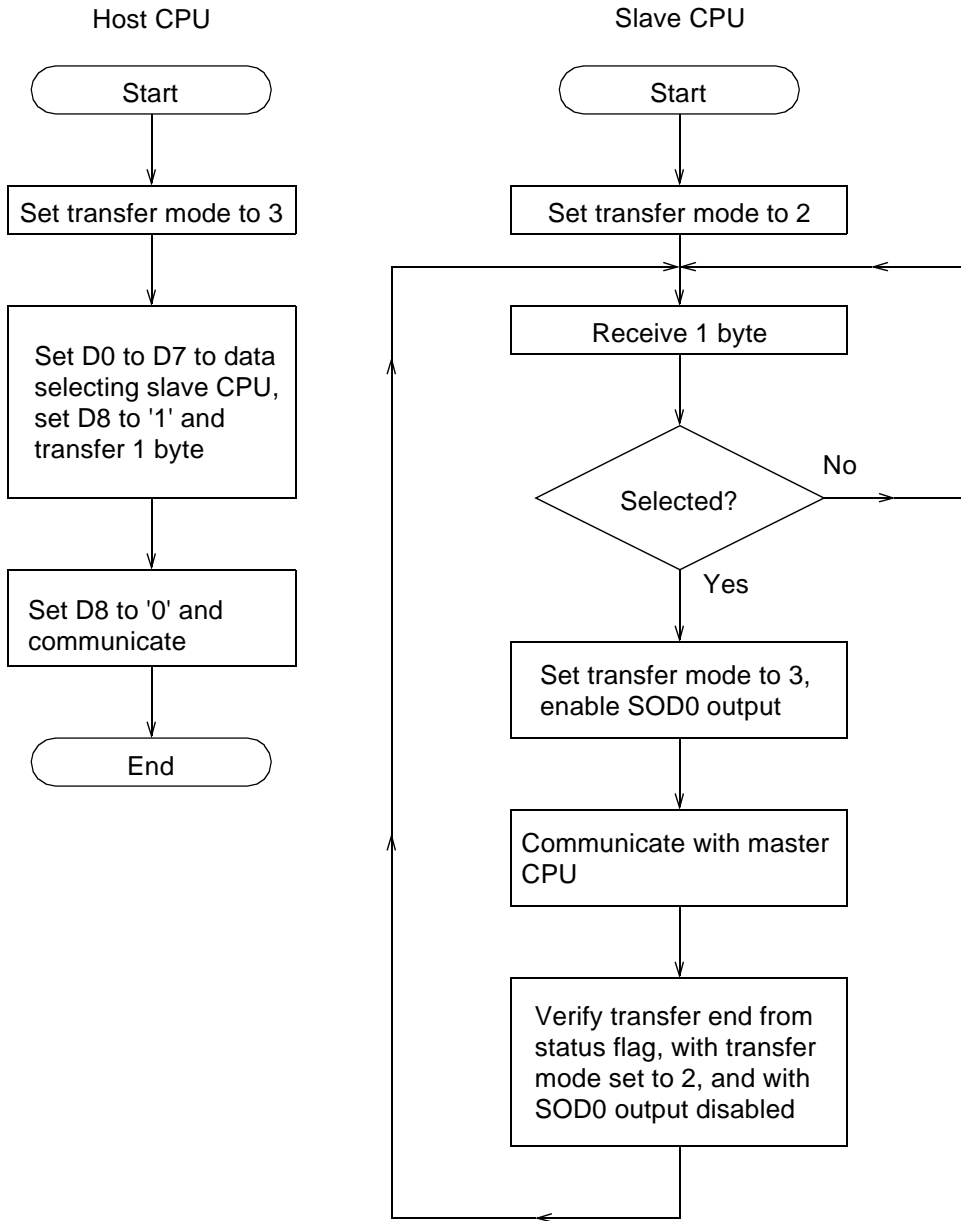


**Fig. 2.5.13 Sample System Configuration Using Mode 2**

Communication begins when the host CPU transfers address data. Address data is data in which the ninth bit (=D8) is set to and '1' to select the slave CPU that is to receive the transmission. The selected slave CPU then communicates with the host CPU according to conventions established by the user. In normal data, the ninth bit is set to "0." The slave CPU that was not selected waits until the next communication begins. Figure 2.5.14 illustrates the flow of this process.

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





**Fig. 2.5.14 Communication Flowchart Using Mode 2**

## 2.6 SSI (Simple Serial Interface)

This block is a serial I/O interface for clock synchronous data transfer using 16/8-bit configuration.

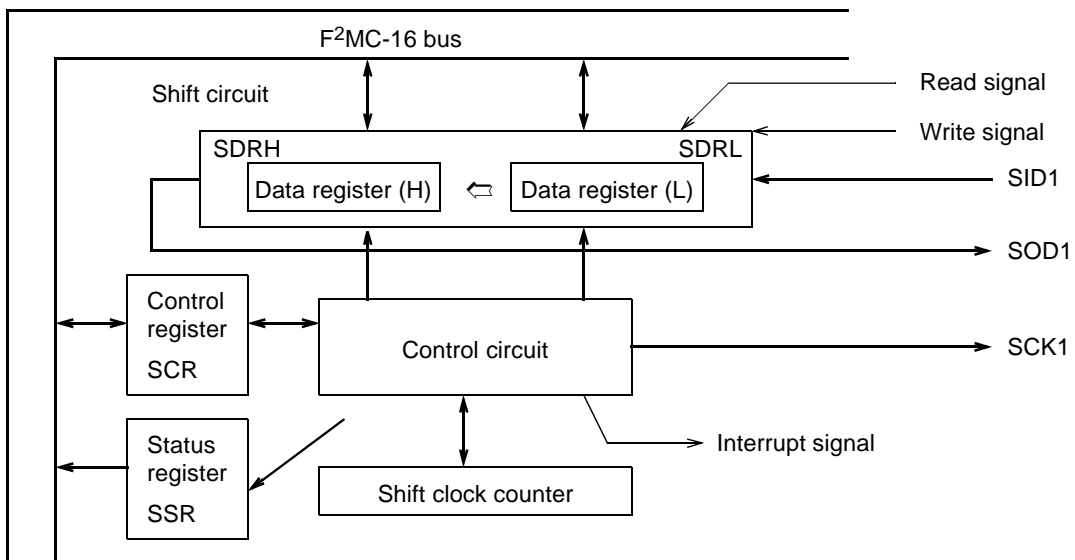
The serial I/O interface provides the following features.

- MSB/LSB-first shift
- 16/8-bit transfer data length
- Maximum shift rate of 8 Mbps (at 32 MHz)
- Shift clock waveform output during data transfer only
- Communications start by writing to SDRH register (SDR word write)
- MB90242A series features 1-channel SSI module

### 2.6.1 Register List

Address:000021H	15	14	13	12	11	10	9	8	Serial status register (SSR)
	-	-	-	-	-	-	BDS	BUSY	
Address:000020H	7	6	5	4	3	2	1	0	Serial control status register (SCR)
	STOP	OCKE	SOE	SIE	SIR	WBS	SMD1	SMD0	
Address:000023H	15	14	13	12	11	10	9	8	Serial data register (high) (SDRH)
	D15	D14	D13	D12	D11	D10	D09	D08	
Address:000022H	7	6	5	4	3	2	1	0	Serial data register (low) (SDRL)
	D07	D06	D05	D04	D03	D02	D01	D00	

### 2.6.2 Block Diagram



## 2.6 SSI (Simple Serial Interface)

### 2.6.3 Register Group Description

#### a) SCR (Serial control status register)

	7	6	5	4	3	2	1	0	
Address:000020H	STOP	OCKE	SOE	SIE	SIR	WBS	SMD1	SMD0	Initial value: 1000 0000 B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

This register controls the serial I/O transfer mode.

#### [Bit 7] Stop bit (STOP)

Transfer operations can be stopped at any time by writing 1 to this bit. Also no transmissions are performed when STOP=1.

0: Normal operation

1: Transfer stop [Initial value]

#### [Bit 6] SCK1 enable bit (OCKE: Output Clock Enable)

This bit enables output from the SCLK (output clock) pins.

0: Pins function as an I/O port [Initial value]

1: Pins function as an SCLK output.

#### [Bit 5] SOD1 enable bit (SOE: Serial Output Enable)

This bit enables output from the Sout (serial clock) pins.

0: Pins function as I/O port [initial value]

1: Pins function as SOUT output.

#### [Bit 4] Transmission interrupt enable bit (SIE: Serial I/O Interrupt Enable)

This bit controls transmission interrupt requests as follows.

0: Serial I/O interrupt disabled [Initial value]

1: Serial I/O interrupt enable

#### [Bit 3] Serial I/O interrupt request bit (SIR: Serial I/O Interrupt Request)

0: No interrupt request. However, with read-modify-write commands, the read value will always be '1.'

1: Set at end of serial data transfer. To enable interrupt (SIE=1). Interrupt request generated to CPU when interrupt is enabled (SIE=1).

Writing '1' to this bit has no significance.

SIR bit clear conditions

- Cleared by reading or writing operations to data registers
- Cleared when STOP=1
- Cleared by a reset
- Cleared by writing '0' to the SIR bit

#### [Bit 2] Word/byte select bit (WBS: Word Byte Select)

This bit switches between 8- and 16-bit data length.

0: 8-bit transfer [Initial value]

1: 16-bit transfer

Writing to this bit during transmission is prohibited.

[Bit 1, 0] Shift clock select bit (SMD1, SMD0: Serial shift clock mode)

This bits are used to select serial clock mode while in source oscillation at 32 MHz operation, as shown in the following table.

SMD1	SMD0	Gear function 1/1	Gear function 1/2	Gear function 1/4	Gear function 1/16
0	0	8Mbps	4Mbps	2Mbps	500Kbps
0	1	4Mbps	2Mbps	1Mbps	250Kbps
1	0	2Mbps	1Mbps	500bps	125Kbps
1	1	1Mbps	500Kbps	250bps	62.5Kbps

- Note:**
- Writing to these bits during transmission is prohibited.
  - Use of the gear function causes changes in communication speeds, and therefore requires setting of the SMD0, SMD1 bits.

b) Serial Status Register (SSR)

	15	14	13	12	11	10	9	8	
Address:000021H	–	–	–	–	–	–	BDS	BUSY	Initial value: ---- ---00B
							R/W	R	

[Bit 15 to 10] Not used.

- Read values are undefined.

[Bit 8] Transfer status bit (BUSY)

- This bit indicates whether the serial transfer is in progress or not.  
0: Stop status [Initial value]  
1: Serial transfer status

[Bit 9] Transfer direction selection bit (BDS)

- This bit determines, on the serial data input/output, whether the bits from the LSB be transferred first or from the MSB first.  
0: MSB first [Initial value]  
1: LSB first
- Updating the bits is prohibited during the transfer.

c) Serial Data Register (SDR)

These registers are used for reading and writing of transfer data.

	15	14	13	12	11	10	9	8	
Address:000023H	D15	D14	D13	D12	D11	D10	D09	D08	Initial value: XXXX XXXXb
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	7	6	5	4	3	2	1	0	
Address:000022H	D07	D06	D05	D04	D03	D02	D01	D00	Initial value: XXXX XXXXb
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

- Writing to these registers during transfer is prohibited.
- Data is transferred by writing to the SDRH register (SDR word write).

### 2.6.4 Operating Description

a) Operating Overview

The SSI block consists of the serial control register, serial status register and serial data register, and is used for input and output of 16/8-bit data. In serial data input/output operation, the contents of the shift register (SDR) are output to the serial output pins (SOUT) in bit series synchronously with the falling edge of the serial clock (SCLK), the signal from the serial input pin (SIN) changes at the falling edge of the serial clock (SCLK) and should be allowed to remain stable until the rising edge of the clock signal. When a transfer ends, an interrupt is generated using the serial I/O interrupt enable bit.

b) Serial I/O Operating Status

There are two types of operating status: transfer status and stop status.

- Transfer status

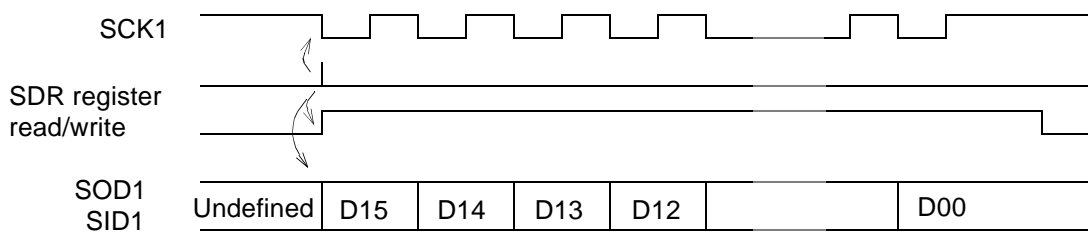
Transfer status is indicated by the value BUSY=1.

- Stop status

Stop status is indicated by the value BUSY=0.

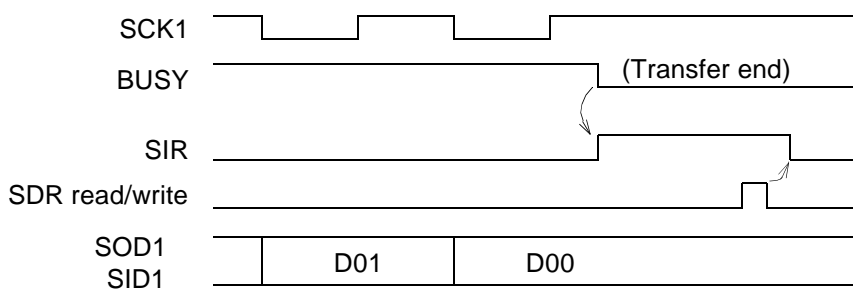
Stop status is initiated by the end of a transfer, or by setting the STOP bit to 1.

c) Shift Operation Start/Stop Bit and I/O Timing



d) Interrupts

The SSI block has the ability to generate interrupt requests to the CPU. When the interrupt flag (SIR bit) is set at the end of a data transfer, if the interrupt enable (SIE) bit is '1', an interrupt request is output to the CPU.



**Fig. 2.6.1 Interrupt Signal Timing**

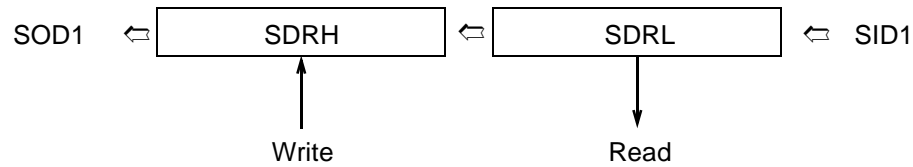
e) SDR Register Read/Write Operation in 8-bit Mode

In 8-bit transfer mode, SDR register read/write operation depends on the data transfer direction.

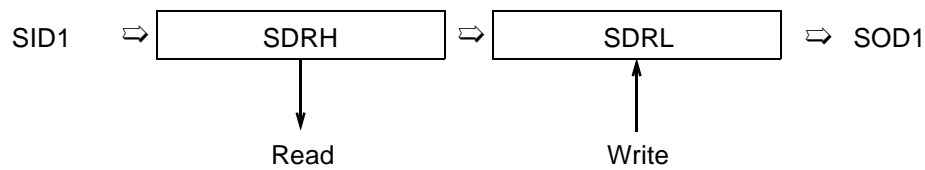
When the MSB first mode is selected, write data into SDH for reading out the data from SDRL.

When the LSB first mode is selected, write data into SDRL for reading out the data from SDRH.

(1) SDR register read/write in the MSB first mode



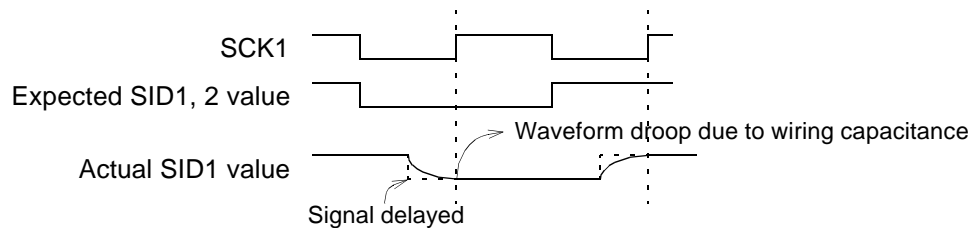
(2) SDR register read/write in the LSB first mode



f) Receiving Serial Data

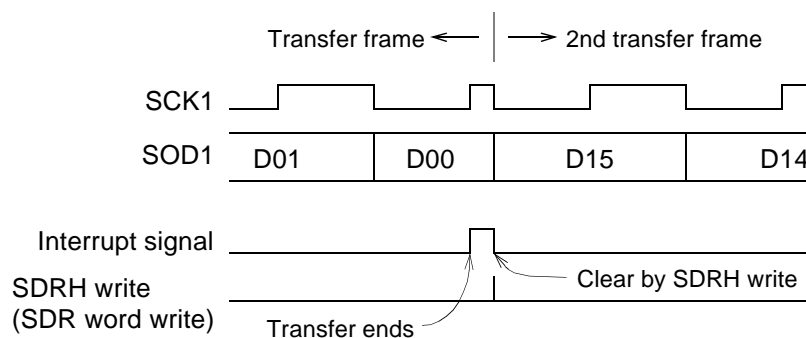
When receiving serial data, dummy write cycles should be used.

g) Expected SIN Signal Value



- The SID1 signals should be held stable until the rising edge of the SCK1 signals. If the value is not stabilized at this time, the shift rate should be reduced.

h) Continuous Transfer



- The SSI module outputs an interrupt signal immediately after the rising edge of the SCK1 signal, so that in continuous transfer operation the 'H' width of the SCK1 signal may be reduced. The SCLK signal may not be picked up for reasons such as use of gear functions at the other end of the communication. Therefore full consideration must be given to possible use of gear functions, or the time interval required between the module interrupt and the next transfer mode.

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

The individual 16-bit timers comprise a 16-bit down-counter, 16-bit reload register, one input pin (TIN0 or TIN1), one output pin (TOT0 or TOT1) and a control register. The choice of input clock signals includes 3 types of internal clock plus an external clock signal. The output pins in reload mode output a toggle-output waveform, and in one-shot mode a square wave to indicate that the count is in progress. The input pins in event count mode are used for event input, and in internal clock mode can be used for trigger input or gate input. The MB90242A series includes three channels of this type of timer.

### 2.7.1 Block Diagram

Figure 2.7.1 shows a block diagram of the 16-bit timer.

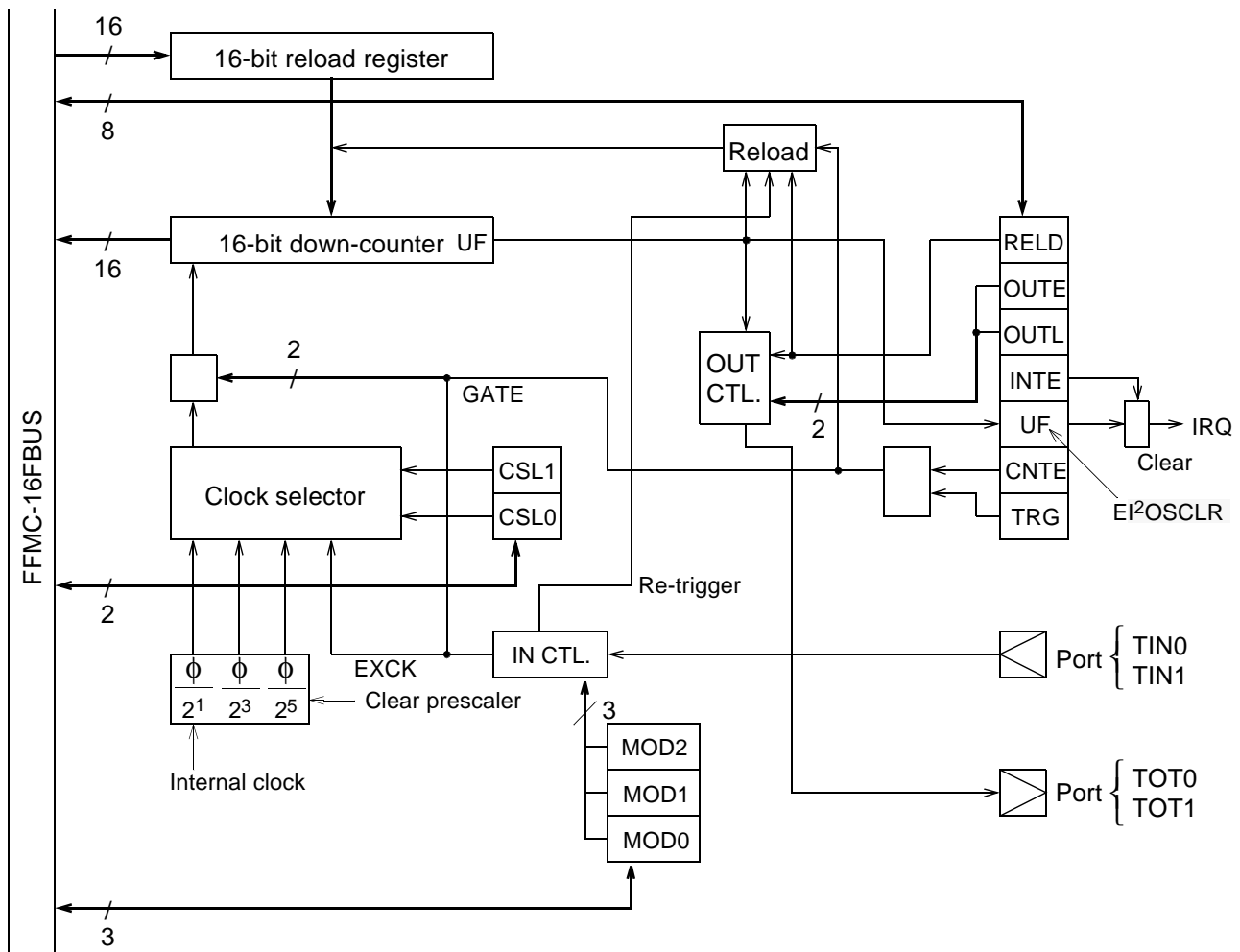
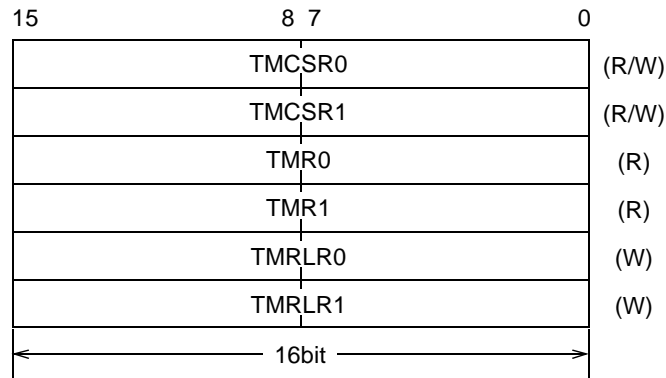
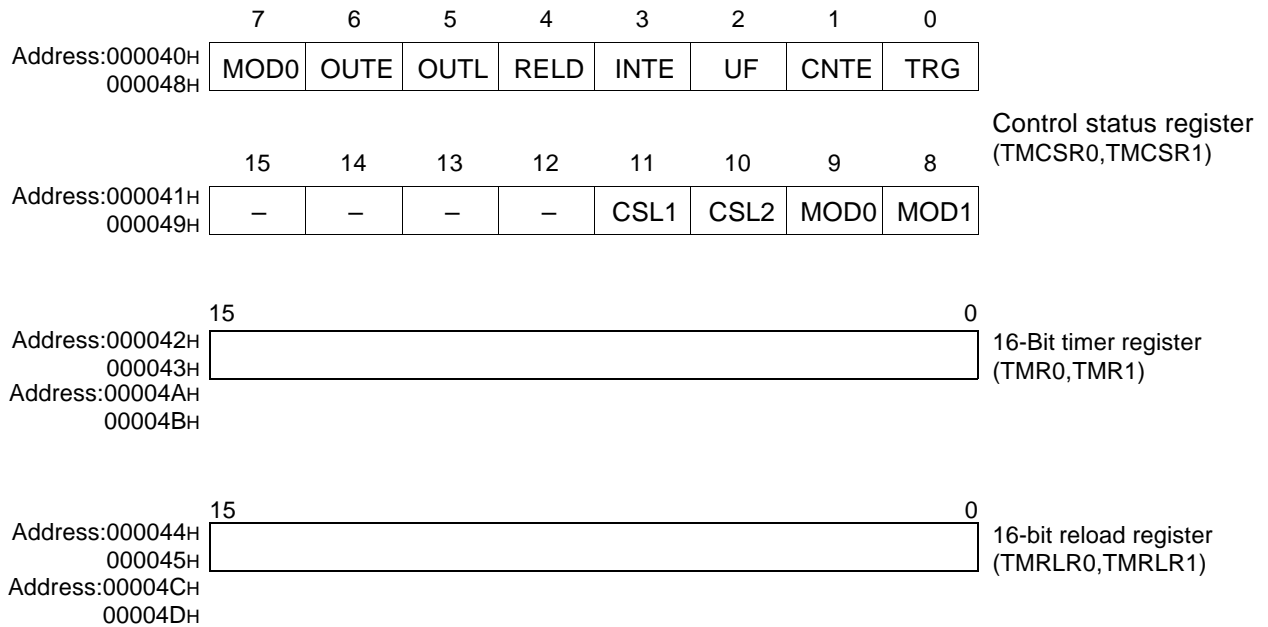


Fig. 2.7.1 16-Bit Timer Block Diagram

### 2.7.2 Register List



**Fig. 2.7.2 16-Bit Timer Register Configuration**



### 2.7.3 Detailed Register Description

#### (1) Control Status Register (TMCSR)

TMCSR	11	10	9	8	7	6	5	4	3	2	1	0
Address:000040H 000041H	CSL1	CSL0	MOD2	MOD1	MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG
Address:000048H 000049H	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

← Read/write  
 ← Initial value

This register controls the 16-bit timer operating mode and interrupts.

Rewriting of bits other than the UF, CNTE, and TRG bits should be done when CNTE=0.

[Bits 11, 10] CSL1, CSL0

Count clock select bits.

Table 2.7.3a shows the clock source selection options. When external event count mode is selected, the valid edge for outing is determined by the MOD1, MOD0 bits.



**Table 2.7.3a Clock Source Setting Using CSL Bits**

CSL1	CSL0	Clock source (machine cycle: $\phi=16$ MHz)
0	0	$\phi / 2^1$ (0.125 $\mu$ s)
0	1	$\phi / 2^3$ (0.5 $\mu$ s)
1	0	$\phi / 2^5$ (2.0 $\mu$ s)
1	1	External event count mode

[Bits 9, 8, 7] MOD2, MOD1, MOD0

These bits determine operating mode as well as I/O pin functions.

The MOD2 bit is used to select input pin functions. When this bit is '0' the input pin is used as a retrigger pin, so that whenever a valid edge is input the contents of the reload register are loaded into the counter and count operation continues. When this bit is '1,' gate count mode is selected, and the signal to the input pin is gated in, meaning that the count will continue only as long as a valid level signal is input.

The MOD1, 0 bits are used to determine the pin functions in each mode. Tables 2.7.3b and 2.7.3c list the combinations of MOD2, 1, 0 bit settings.

**Table 2.7.3b MOD2, 1, 0 bit settings (1)**

Internal clock mode (CSL0, 1=00, 01, 10)

MOD2	MOD1	MOD0	Input pin function	Valid 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	X	0	Gate input	L' level
1	X	1	↑	H' level

**Table 2.7.3c MOD2, 1, 0 bit settings (2)**

Event count mode (CSL0, 1=11)

MOD2	MOD1	MOD0	Input pin function	Valid edge or level
X	0	0	—	—
	0	1	Event input	Rising edge
	1	0	↑	Falling edge
	1	1	↑	Both edges

X : Don't care

**[Bit 6] OUTE**

This is the output enable bit. When the value is '0' the TOT0 (or TOT1) pins become a general-purpose port, and when the value is '1' the TOT0 (or TOT1) pins become timer output pins. The output waveform is a toggle output in reload mode, and in one-shot mode is a square wave output indicating that the count is in progress.

**[Bit 5] OUTL**

This bit sets the output level of the TOT0 (or TOT1) pins. Pin levels are reversed by switching between settings of '0' and '1.'

**Table 2.7.3d OUTE, RELD, OUTL Pin Settings**

OUTE	RELD	OUTL	Output waveform
0	X	X	General-purpose port
1	0	0	H' square wave during count
1	0	1	L' square wave during count
1	1	0	L' toggle output at start of count
1	1	1	H' toggle output at start of count

**[Bit 4] RELD**

This is the reload enable bit. When the value is '1' the timer is in reload mode, and each time the counter value reaches an underflow condition by going from 0000<sub>H</sub> to FFFF<sub>H</sub> the contents of the reload register are loaded into the counter and the count operation continues. If the value is '0,' count operation stops when the counter value reaches an underflow condition by going from 0000<sub>H</sub> to FFFF<sub>H</sub>.

**[Bit 3] INTE**

This is the interrupt enable bit. When the value is '1' an interrupt request is generated each time the UF bit is set to '1.' When the value is '0' no interrupt request is generated.

**[Bit 2] UF**

This is the timer interrupt request bit, and is set to '1' each time the counter value reaches an underflow condition by going from 0000<sub>H</sub> to FFFF<sub>H</sub>. This bit can be cleared by writing '0' or by the intelligent I/O service. Writing '1' to this bit has no effect.

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

With read-modify-write commands, the read value is always '1.'

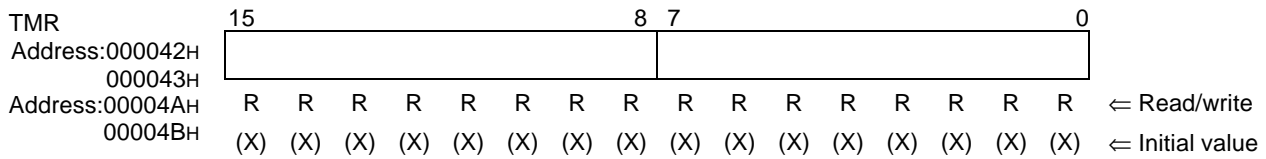
[bit 1] CNTE

This is the timer count enable bit. Writing '1' to this bit enters the state waiting for a trigger for activation. Writing '0' stops the count operation.

[bit 1] TRG

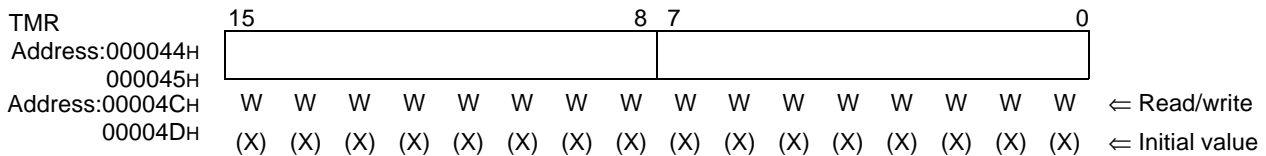
This is the software trigger bit. Writing '1' to this bit applies a software trigger, loading the contents of the reload register into the counter, and starting count operation. Writing '0' to this bit has no effect. The read value is always '0.' Trigger input applied by this register is valid only when the CNTE bit is '1.' Nothing will occur when the CNTE bit is '0.'

### (2) 16-Bit Timer Register (TMR)



This register is able to read the count value from the 16-bit timer. Its initial value is indeterminate. This register should always be read using word transfer commands.

### (3) 16-Bit Reload Register (TMRLR)



The 16-bit reload register saves the initial count value. Its initial value is indeterminate. This register should always be read using word transfer commands.

## 2.7.4 Operating Description

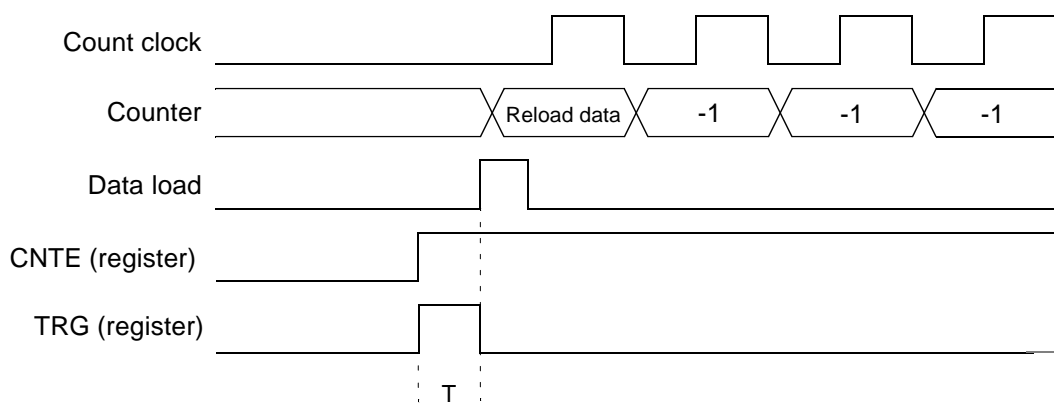
### (1) Internal Clock Operation

A selection of clock sources is provided to enable timers to operate on clock signals that are multiples of the internal clock frequency, by dividing the machine clock period by  $2^1$ ,  $2^3$ , or  $2^5$ . A register setting allows the external input pins to serve as a trigger input or gate input type.

To start count operations simultaneously with the count-enable signal, both the CNTE bit and TRG bit in the control register should be set to '1.' When the timer is in startup status (CNTE=1), trigger input from the TRG bit is valid at all times regardless of operating mode.

Counter startup and counter operation are shown in Figure 2.7.4a.

The time interval T (T= machine cycles) is required between the trigger input that starts the counter, and the loading of data from the reload register into the counter.



**Fig. 2.7.4a Counter Startup and Operation**

### (2) Underflow Operation

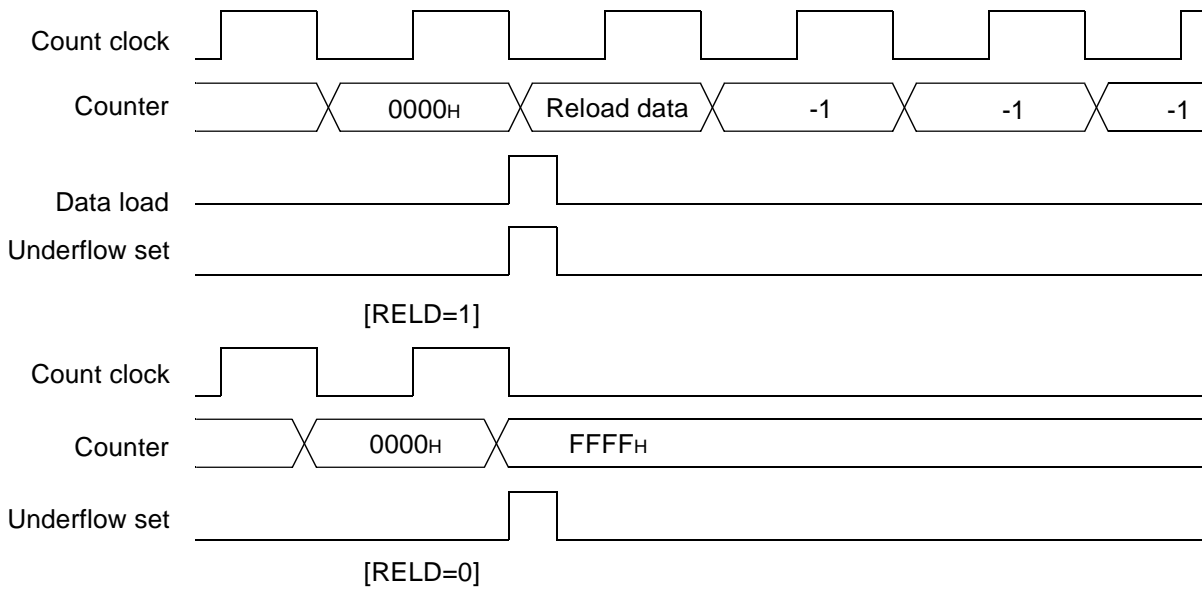
An underflow condition results when the counter value changes from 0000H to FFFFH. Thus an underflow will occur after an interval of "reload register setting value +1" counts.

When an underflow is generated, and the control register RELD bit is '1' the contents of the reload register are loaded into the counter, and count operations continue. When the RELD bit is '0' the count stops at FFFFH.

An underflow condition sets the UF bit in the control register, and if the INTE bit is set to '1' and interrupt request is generated.

Figure 2.7.4b illustrates underflow operation.

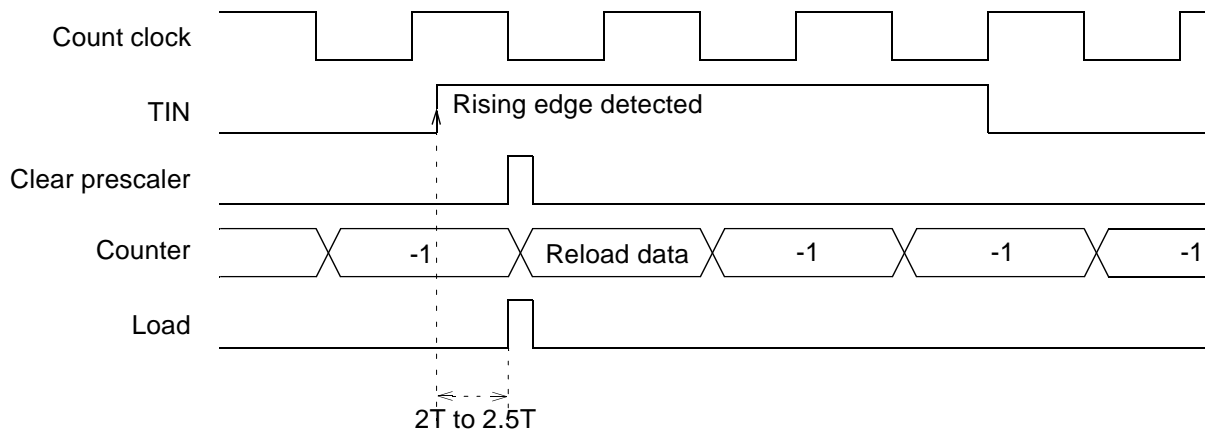
## 2.7 16-Bit Reload Timer (With Event Count Function)



**Fig. 2.7.4b Underflow Operation**

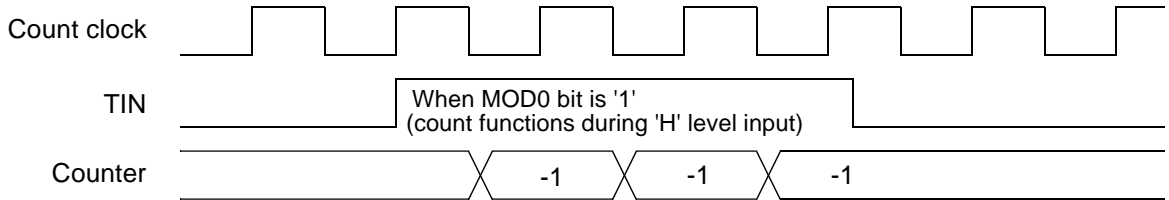
### (3) Input Pin Functions (Internal Clock Mode)

When an internal clock is selected as a clock source, the TIN pin can be used as either a trigger input or a gate input. When the TIN pin is used as a trigger input, the contents of the reload register are loaded into the counter whenever a valid edge is input, and count operation starts after the input prescaler is cleared. The TIN input pulse should be at least  $2 \times T$  ( $T$ = machine cycles). Figure 2.7.4c illustrates trigger input operation.



**Fig. 2.7.4c Trigger Input Operation**

When the TIN pin is used as a gate input, the count functions only as long as the signal input from the TIN pin is at the valid level as determined by the MOD0 bit in the control register. During this time the count clock continues to operate without stopping. In gate mode, software triggers are enabled regardless of gate level. The pulse width at the TIN pin should be  $2 \times t$  ( $T = \text{machine cycles}$ ). Figure 2.7.4d illustrates gate input operation.



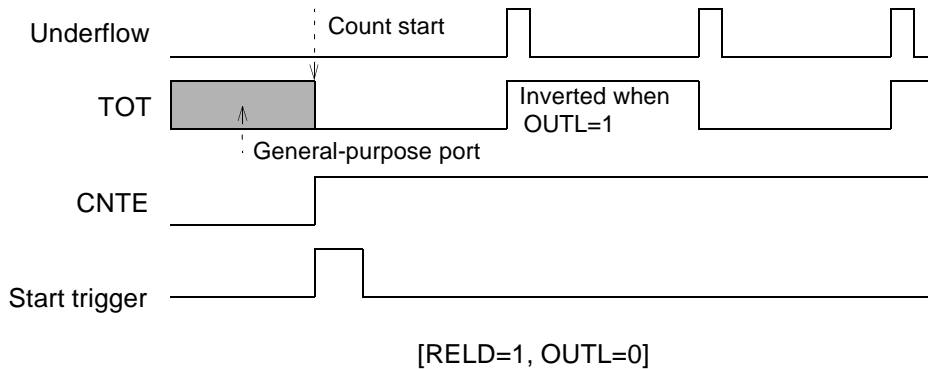
**Fig. 2.7.4d Gate Input Operation**

**(4) External Event Count**

When the external clock source is selected, the TIN pin becomes an external event input pin, and counts valid edges as defined by register setting. The pulse width at the TIN pin should be  $2 \times t$  ( $T = \text{machine cycles}$ ).

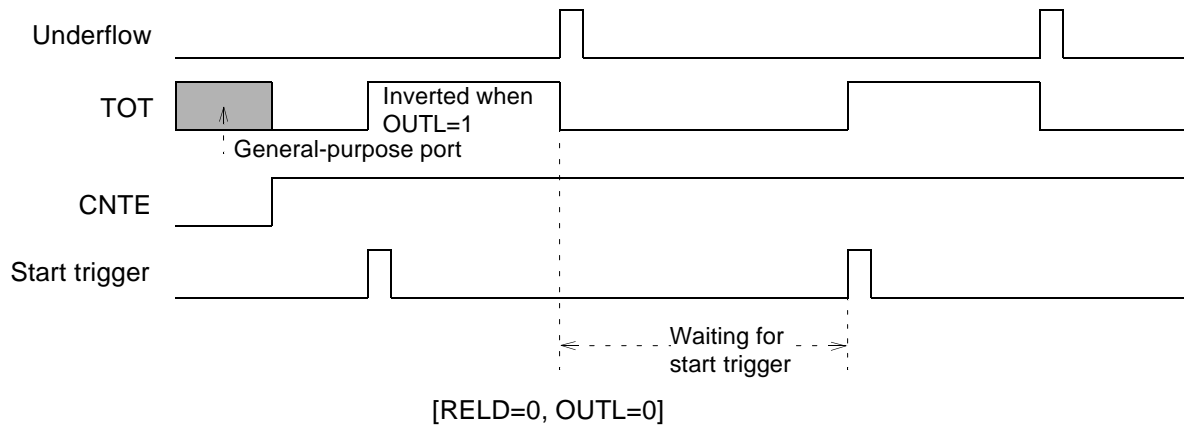
**(5) Output Pin Function**

In reload mode the TOT pin functions as a toggle output inverted by underflow conditions, and in one-shot mode the TOT pin functions as a pulse output indicating that the count is in progress. Output polarity can be set by the DUTL bit, such that when OUTL=0, the toggle output has an initial value of "0," and the one-shot pulse is output as '1' while the count is in progress. When OUTL=1, the output waveform is inverted.



**Fig. 2.7.4e Output Pin Function (1)**

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



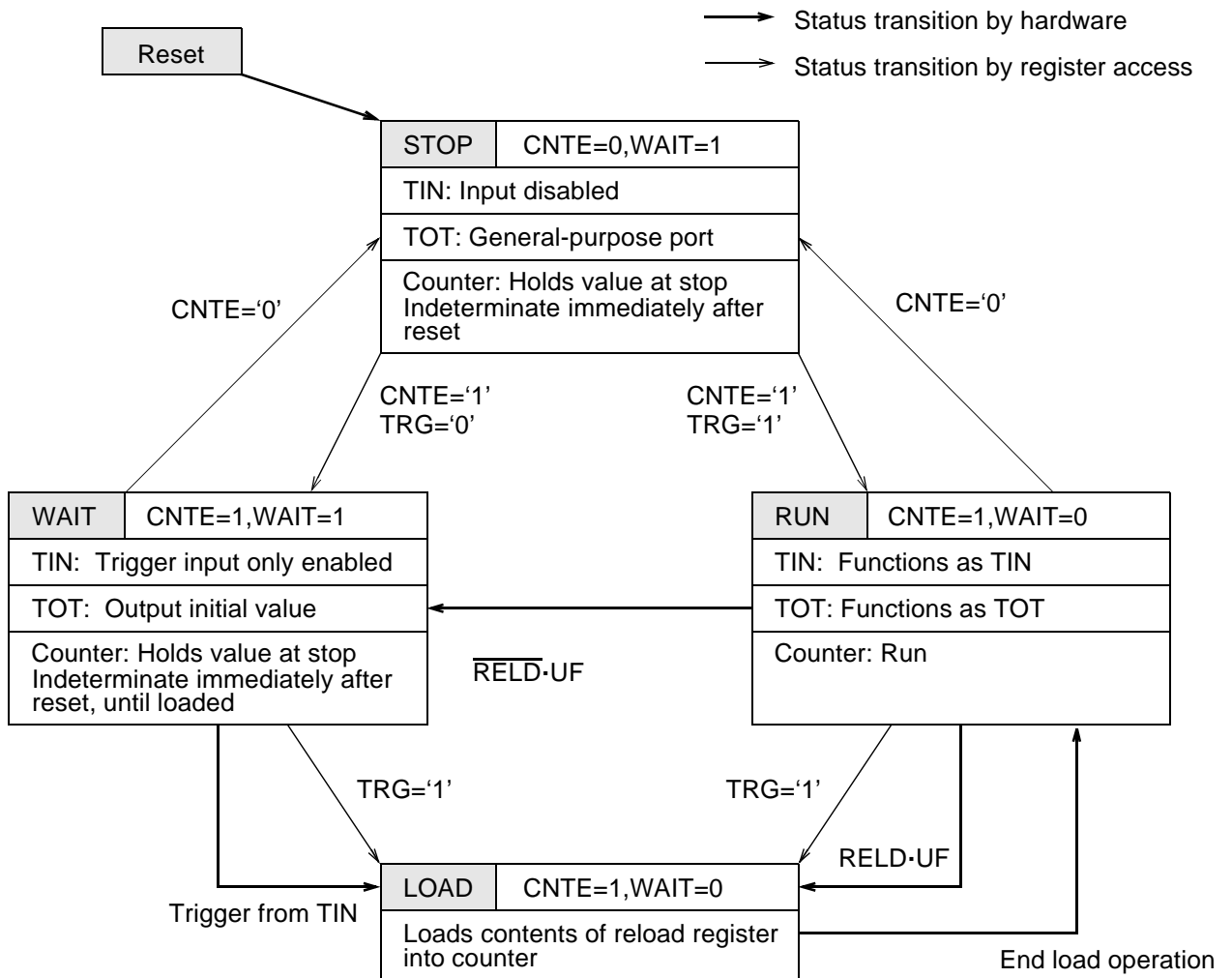
**Fig. 2.7.4f Output Pin Function (2)**

### (6) Extended Intelligent I/O Service (EI<sup>2</sup>OS) Function and Interrupts

This timer circuit has a circuit adapted for EI<sup>2</sup>OS. This means that a timer overflow can be used to start the EI<sup>2</sup>OS operation. The MB90242A series has three timers, and all can be used with EI<sup>2</sup>OS.

**(7) Counter Operating Status**

Counter status is determined by the CNTE bit in the control register and the internal WAIT signal. Available settings include CNTE=0 and WAIT=1 for STOP status, CNTE=1 and WAIT=1 for start trigger WAIT status, and CNTE=1 and WAIT=0 for RUN status. Figure 2.7.4g shows the transitions among these three status.



**Fig. 2.7.4g Counter Status Transition**



## 2.8 16-Bit I/O Timer

The 16-bit I/O timer consists of one 16-bit free-running timer with four input capture modules.

This function can be used to measure input pulse width or external clock cycle by using the 16-bit free-running timer as a base.

### 2.8.1 Functional Overview

#### ■ 16-bit Free Run Timer (x1)

The 16-bit free-running timer consists of a 16-bit up-counter, control register and prescaler. Output values from this timer/counter are used as reference time for the input capture modules.

- One of four counter operation clock signals can be selected.  
Four types of internal clock signals ( $\phi /4$ ,  $\phi /16$ ,  $\phi /32$ ,  $\phi /64$ )
- Interrupts can be generated by counter value overflow conditions.
- Counter values can be initialized to '0000H' by reset or software clear commands.

#### ■ Input Capture Modules (x4)

The input capture module consists of four independent external input pins and corresponding capture registers, and a control register.

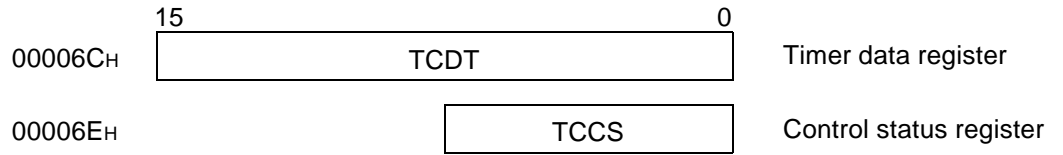
Detection of any given edge of the signals from the external input pins can be used to save the value of the 16-bit free-running timer in a capture register and to simultaneously generate an interrupt.

- Any edge of the external input signal may be selected.  
Rising edge, falling edge, both edges may be selected
- All four input capture modules function independently.
- Interrupts can be generated by the valid edge of the external input signal.

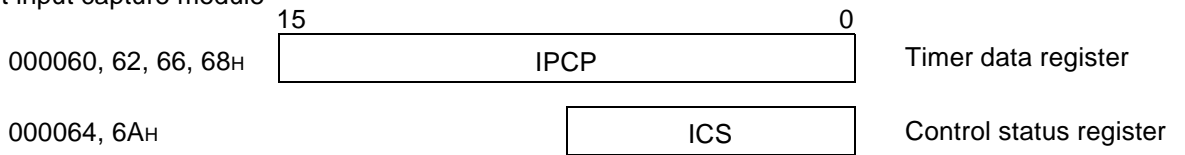
Input capture interrupts can be used to start intelligent I/O services.

### 2.8.2 16-Bit I/O Timer Module Register Configuration

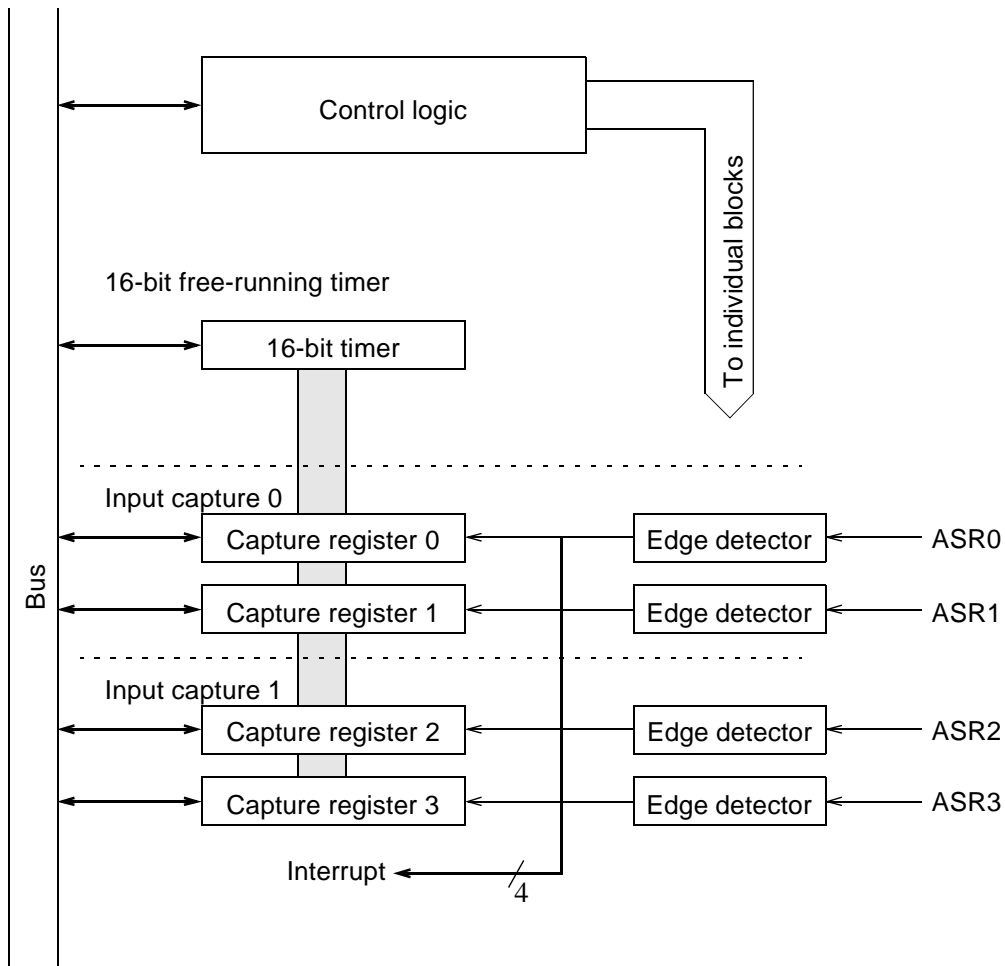
16-bit free-running timer



16-bit input capture module



### 2.8.3 16-bit I/O Timer Module Block Diagram

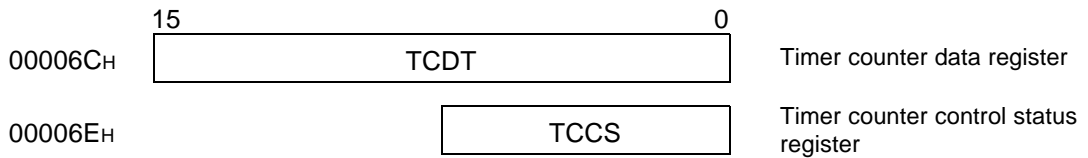


### 2.8.4 16-Bit Free-Running Timer

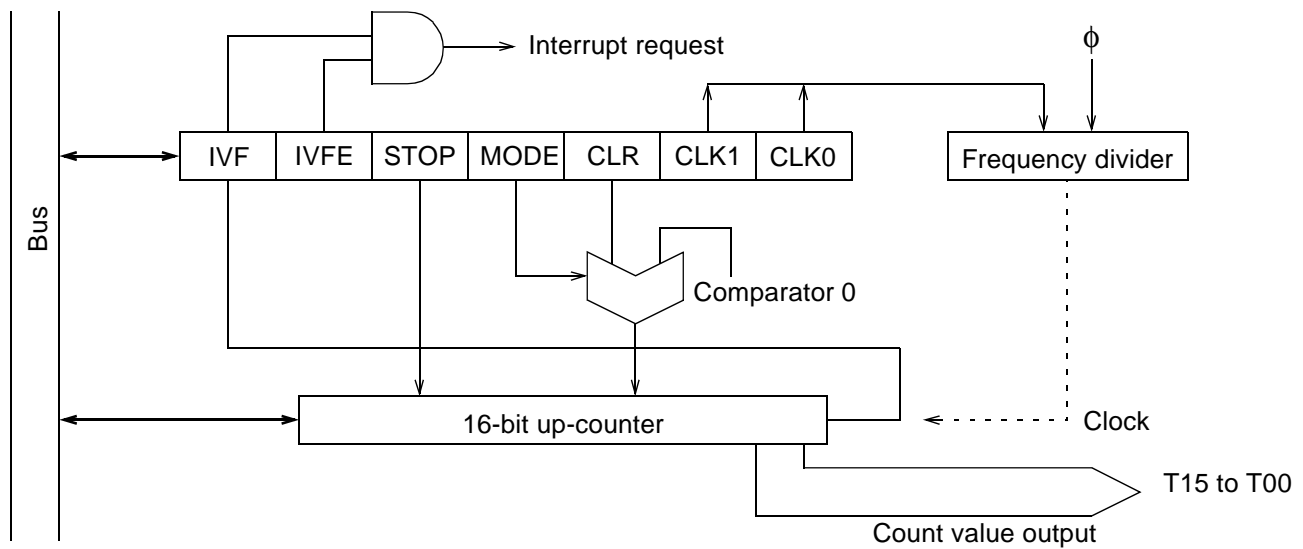
The 16-bit free-running timer consists of a 16-bit up-counter and control status register. Count values from this timer are used as a reference time for the input capture modules.

- One of four counter operation clock signals can be selected.
- Interrupts can be generated by counter value overflow conditions.

### 2.8.5 16-Bit Free-Running Timer Module Register Configuration



### 2.8.6 Block Diagram



## 2.8.7 Detailed Register Description for 16-Bit Free-Running Timer Module

### (1) Data Register

bit	15	14	13	12	11	10	9	8	
00006CH	T15	T14	T13	T12	T11	T10	T09	T08	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	⇐ Attributes
	0	0	0	0	0	0	0	0	⇐ Initial value

bit	7	6	5	4	3	2	1	0	
	T07	T06	T05	T04	T03	T02	T01	T00	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	⇐ Attributes
	0	0	0	0	0	0	0	0	⇐ Initial value

This register can read the count value from the 16-bit free-running counter. The counter value is cleared to '0000' at a reset.

Timer values can be set by writing to this register, however this should always be done when the timer is in stop status (STOP=1).

This register should be accessed by word access.

The 16-bit free-running timer can be initialized by the following two factors.

- Initialization by a reset
- Initialization by the clear bit (CLR) in the control status register

### (2) Control Status Register

bit	7	6	5	4	3	2	1	0	
00006CH	Reserved	IVF	IVFE	STOP	Reserved	CLR	CLK1	CLK0	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	⇐ Attributes
	0	0	0	0	0	0	0	0	⇐ Initial value

[Bit 7] Reserved

Always write '0' to this bit.

[Bit 6] IVF

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

This bit is set to '1' when the 16-bit free-running timer creates an overflow condition.

In this case an interrupt will be generated if the interrupt request enable bit (bit 4: IVFE) has been set. This bit can be cleared by writing '0.' Writing '1' to this bit has no effect.

With read-modify-write commands the read value is always '1. '

0	No interrupt request (initial value)
1	Interrupt request

## 2.8 16-Bit I/O Timer

### [Bit 5] IVFE

This is the 16-bit free-running timer overflow interrupt enable bit.

When this bit is '1' an interrupt will be generated whenever the interrupt flag (bit 5: IVF) is set to '1.'

0	Interrupt disabled (initial value)
1	Interrupt enabled

### [Bit 4] STOP

This bit is used to stop the counting by the 16-bit free-running timer.

When '1' is written to this bit the timer counting is stopped.

When '0' is written to the bit the timer counting is started.

0	Enable count (run) (initial value)
1	Disable Count (stop)

### [Bit 3] Reserved

Always write '0' to this bit.

### [Bit 2] CLR

This bit initializes the 16-bit free-running timer to '0000' during operation.

When '1' is written to this bit, the counter value is initialized to '0000.'

When '0' is written to this bit, there is no effect. The read value is always '0.'

0	No effect (initial value)
1	Initialize counter value to '0000.'

Counter value initialization occurs at the transition of the count value.

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

[Bits 1, 0] CLK1, CLK0

These bits select the 16-bit free-running timer count clock.

The clock signal will switch immediately after writing to these bits, so that changes should only be made when the output compare and input capture functions are in stop status.

External Clock (X0, X1) at 32 MHz

CLK1	CLK0	Count clock	Gear 1/1 $\phi = 16\text{MHz}$	Gear 1/2 $\phi = 8\text{MHz}$	Gear 1/4 $\phi = 4\text{MHz}$	Gear 1/16 $\phi = 1\text{MHz}$
0	0	$\phi / 4$	0.25 $\mu\text{s}$	0.5 $\mu\text{s}$	1 $\mu\text{s}$	4 $\mu\text{s}$
0	1	$\phi / 16$	1 $\mu\text{s}$	2 $\mu\text{s}$	4 $\mu\text{s}$	16 $\mu\text{s}$
1	0	$\phi / 32$	2 $\mu\text{s}$	4 $\mu\text{s}$	8 $\mu\text{s}$	32 $\mu\text{s}$
1	1	$\phi / 64$	4 $\mu\text{s}$	8 $\mu\text{s}$	16 $\mu\text{s}$	64 $\mu\text{s}$

External Clock (X0, X1) at 24 MHz

CLK1	CLK0	Count clock	Gear 1/1 $\phi = 12\text{MHz}$	Gear 1/2 $\phi = 6\text{MHz}$	Gear 1/4 $\phi = 3\text{MHz}$	Gear 1/16 $\phi = 750\text{KHz}$
0	0	$\phi / 4$	0.33 $\mu\text{s}$	0.66 $\mu\text{s}$	1.33 $\mu\text{s}$	5.33 $\mu\text{s}$
0	1	$\phi / 16$	1.33 $\mu\text{s}$	2.67 $\mu\text{s}$	5.33 $\mu\text{s}$	21.3 $\mu\text{s}$
1	0	$\phi / 32$	2.67 $\mu\text{s}$	5.33 $\mu\text{s}$	10.7 $\mu\text{s}$	42.7 $\mu\text{s}$
1	1	$\phi / 64$	5.33 $\mu\text{s}$	10.7 $\mu\text{s}$	21.3 $\mu\text{s}$	85.3 $\mu\text{s}$

### 2.8.8 Input Capture

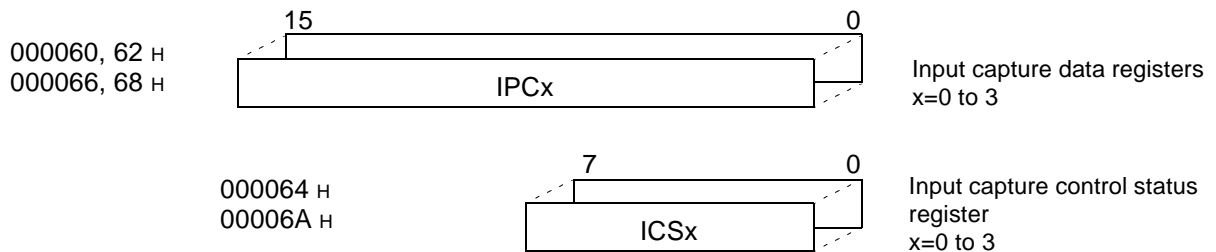
The module is equipped with a function that can detect either the rising or falling edge, or both edges, of a signal input externally, and can save the value of the 16-bit free-running timer at that moment. Interrupts can also be generated at the moment of edge detection.

The input capture module consists of an input capture data register and a control register.

Each input capture module has its own external input pin.

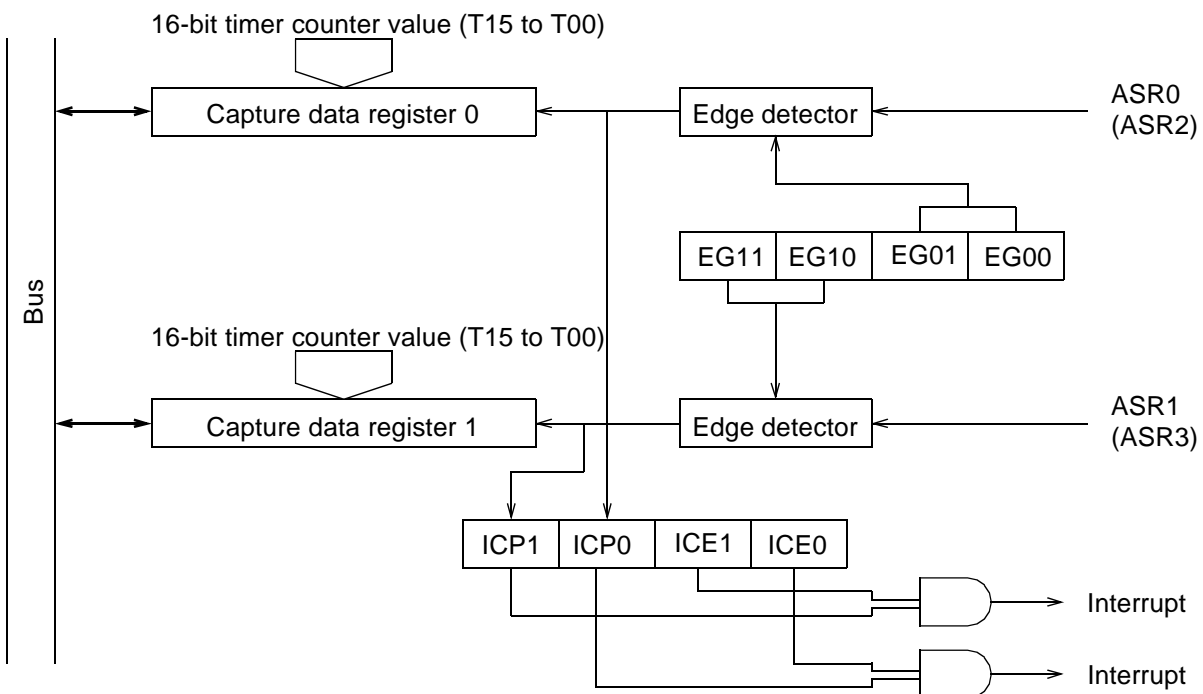
- Three types of valid edge detection can be applied to external signals.
  - Rising edge (↑)
  - Falling edge (↓)
  - Both edges (↑↓)
- Interrupts can be generated at the moment the valid edge is detected in the external signal input.

### 2.8.9 Input Capture Module Register Configuration



### 2.8.10 Input Capture Module Block Diagram

(2 channels: This model contains two of these units, for a total of 4 channels.)



## 2.8.11 Detailed Register Description for Input Capture Module

### (1) Input Capture Data Registers

bit	15	14	13	12	11	10	9	8	
000060, 62H 000066, 68H	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	
	R	R	R	R	R	R	R	R	⇐ Attributes
	X	X	X	X	X	X	X	X	⇐ Initial value

bit	7	6	5	4	3	2	1	0	
	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	
	R	R	R	R	R	R	R	R	⇐ Attributes
	X	X	X	X	X	X	X	X	⇐ Initial value

These registers save the value of the 16-bit free-running timer when the valid edge of the waveform input from the corresponding external pin is detected.

Values at reset are indeterminate.

Access to these registers should be a word access type. Write access is not allowed.

### (2) Control Status Register

bit	7	6	5	4	3	2	1	0	
000064H 00006AH	ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	⇐ Attributes
	0	0	0	0	0	0	0	0	⇐ Initial value

[bits 7, 6] ICP1, ICP0

This is the input capture interrupt flag.

When a valid edge of the external input pin is detected, this bit is set to '1.'

If the interrupt enable bit (ICE0, ICE1) is set an interrupt will be generated when a valid edge is detected.

This bit can be cleared by writing '0.' Writing '1' has no effect. With read-modify-write commands, the read value is '1.'

0	No valid edge detected (initial value)
1	Valid edge detected

The ICP0 bit corresponds to input capture 0

The ICP1 bit corresponds to input capture 1.



## 2.8 16-Bit I/O Timer

[bits 5, 4] ICE1, ICE0

This is the input capture interrupt enable bit.

When this bit is set to '1' an input capture interrupt is generated whenever the interrupt flags (ICP0, ICP1) are set.

0	Interrupt disabled (initial value)
1	Interrupt enabled

The ICE0 bit corresponds to input capture 0

The ICE1 bit corresponds to input capture 1.

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

These bits determine the polarity of the valid edge from the external input pin. They also serve to enable the operation of the input capture module.

EG11 EG01	EG10 EG00	Edge detection polarity	
0	0	No edge detection (stop mode)	(initial value)
0	1	Rising edge detection ↑	
1	0	Falling edge detection ↓	
1	1	Dual edge detection ↑↓	

The EG01, EG00 bits correspond to input capture 0.

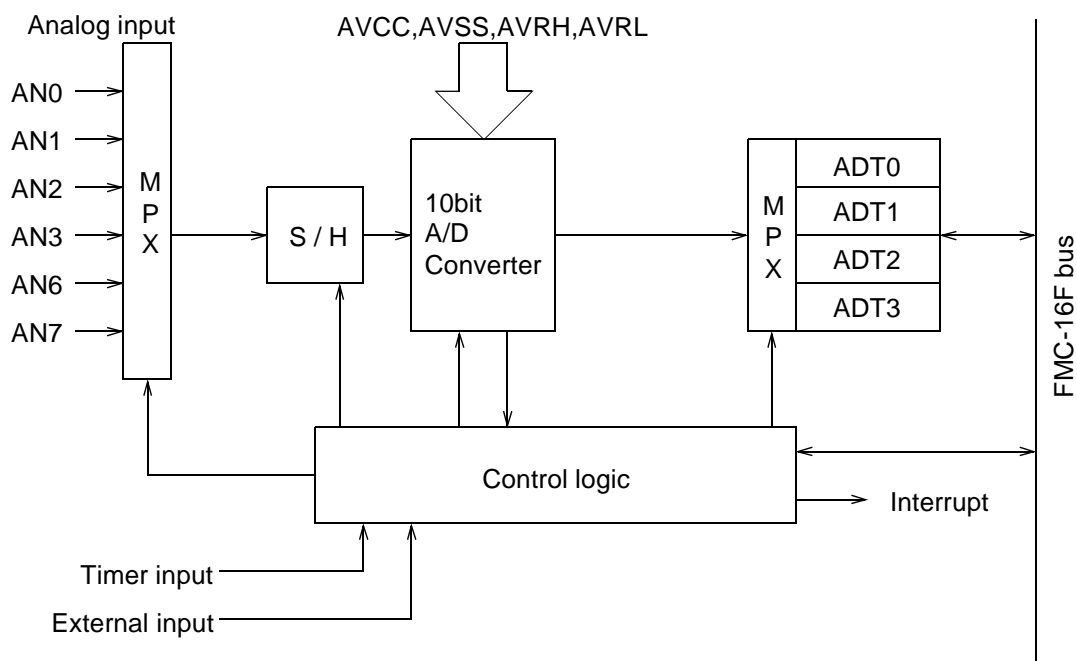
The EG11, EG10 bits correspond to input capture 1.

## 2.9 A/D Converter

The A/D Converter converts analog input voltage into digital values, and provides the following features.

- Conversion time: minimum 1.25 $\mu$ s per channel
- Series-parallel conversion with sample-and-hold circuit
- 10-bit resolution (8/10-bit switching)
- Program selection from 6 analog input channels
- Conversion modes
  - Single conversion mode: one-time conversion of one selected channel
  - Scan conversion mode: scan conversion of maximum of 4 channels
- Conversion data stored in data buffers (total of four data buffers)
- On completing the A/D conversion, the interrupt request that the A/D conversion has ended may be generated to CPU. This interrupt can also be used to start EI<sup>2</sup>OS.
- Startup may be triggerable by software, external trigger (falling edge) or timer (rising edge).

### 2.9.1 Block Diagram



### 2.9.2 Register Configuration

The A/D converter has the following registers.

Address:000070H	bit 15	ADCS2	bit 8 7	ADCS1	bit 0	Control & status register.
Address:000072H		ADCT2		ADCT1		Conversion time setting register
Address:000074H		ADTH0		ADTL0		Conversion data register 0
Address:000076H		ADTH1		ADTL1		Conversion data register 1
Address:000078H		ADTH2		ADTL2		Conversion data register 2
Address:00007AH		ADTH3		ADTL3		Conversion data register 3

### 2.9.3 Detailed Register Description

#### (1) Control & Status Register (ADCS1)

This register provides control and status indicators for the A/D converter.

Address:000070H	bit 7	BUSY	bit 6	INT	bit 5	INTE	bit 4	—	bit 3	STS1	bit 2	STS0	bit 1	STAR	bit 0	Reserved	ADCS1
		0		0		0		—		0		0		0		0	← Initial value
		R/W		R/W		R/W		—		R/W		R/W		R/W		R/W	← Attributes

[Bit 7] BUSY (BUSY):

Read values	0	A/D converter stopped
	1	A/D converter busy
Write values	0	A/D converter forced stop
	1	No effect

Do not execute a software start and forced stop at the same time (STAR=1, BUSY=0).

For RMW commands, the read value is '1.' The initial value at a reset is '0.'

[Bit 6] INT (INTerrupt):

This bit is set by the end of conversion processing (either by the end of 1 conversion cycle in single conversion mode, or by the end of conversion on all channels in scan conversion mode).

0	No interrupt request
1	Interrupt request

This bit can be cleared by writing '0' or starting conversion, or by an EI<sup>2</sup>OS interrupt clear or reset. Writing '1' has no effect.

With read-modify-write commands, the read value is always '1.'

[bit 5] INTE (INTerrupt Enable):

This bit is used to enable/disable interrupts resulting from the end of conversion.

0	Interrupt disabled
1	Interrupt enabled

This bit should be set when using EI<sup>2</sup>OS.

This bit is initialized to '0' at reset.

[Bit 4] Not used

[Bits 3, 2] STS1, STS0 (Start select);

These bits can be set to determine the A/D conversion start factors.

STS1	STS0	Function
0	0	Software start
0	1	External pin-triggered start and software start
1	0	Timer 1 start and software start
1	1	External pin-triggered start, timer 1 start and software start

In modes with more than one start factor, A/D conversion will be started by the first factor to occur. Any start factors occurring during A/D operation (BUSY bit=1) will be ignored (no restart function enabled).

When restarting, stop A/D conversion once (write '0' to the BUSY bit), then restart.

External trigger startups operate on detection of a falling edge, and timer 1 startups on a rising edge.

Initial value at reset is '00.'

[Bit 1] STAR (Start):

Write '1' to this bit to start A/D conversion.

When the A/D converter is operating (BUSY=1), writing '1' has no effect.

Read value is always '0.'

Do not execute a software start and forced stop at the same time (STAR=1, BUSY=0).

[Bit 0] Reserved:

Always write '0' to this bit.

## (2) Control & Status Register (ADCS2)

This register provides control and status indicators for the A/D converter.

bit	15	14	13	12	11	10	9	8	
Address : 000071H	—	ACS2	ACS1	ACS0	—	—	CREG	SCAN	ADCS2
	—	0	0	0	—	—	0	0	← Initial value
	—	R/W	R/W	R/W	—	—	R/W	R/W	← Attributes

[Bits 15] Not used

## 2.9 A/D Converter

[Bits 14, 13, 12] ACS2, ACS1, ACS0 (Analog channel set)

These bits are used to select the A/D conversion channel.

- In single conversion mode, these bits change the channel selection.
- In scan conversion mode, these bits select the scan start channel (see section 2.9.4 "Operating Description").

When AN0 to AN3 are selected, scan conversion will cover the selected channel through AN3.

When AN6 to AN7 are selected, scan conversion will cover the selected channel through AN7.

- Results are stored in different registers depending on channels selected.

Note that the AN2 to AN3 and AN6 to AN7 data registers overlap.

When conversion on AN2 is followed by conversion on AN6, data will be overwritten causing loss of the conversion data on AN2.

ACS2	ACS1	ACS0	Channel	Corresponding data register
0	0	0	AN0	ADTH0, ADTL0
0	0	1	AN1	ADTH1, ADTL1
0	1	0	AN2	ADTH2, ADTL2
0	1	1	AN3	ADTH3, ADTL3
1	0	0	Setting prohibited	—
1	0	1	Setting prohibited	—
1	1	0	AN6	ADTH2, ADTL2
1	1	1	AN7	ADTH3, ADTL3

[Bits 11, 10] Not used

[Bit 9] CREG:

This bit is used to set the number of bits to save from A/D conversion (see section 2.9.3(4)).

0: 10-bit mode

The upper 2 bits of the conversion value are stored in the ADTHx register, and the lower 8 bits in the ADTLx register.

1: 8-bit mode

The upper 8 bits of the conversion value are stored in the ADTLx register.

The initial value at reset is '0.'

[Bit 8] SCAN:

This bit selects the conversion mode.

The initial value after reset is '0.'

	Function
0	Single conversion mode
1	Scan conversion mode

**(3) Conversion time setting registers (ADCT1, ADCT2)**

bit	7	6	5	4	3	2	1	0	
Address:000072H	CV13	CV12	CV11	CV10	CV23	CV22	CV21	CV20	ADCT1
	X	X	X	X	X	X	X	X	← Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Attributes

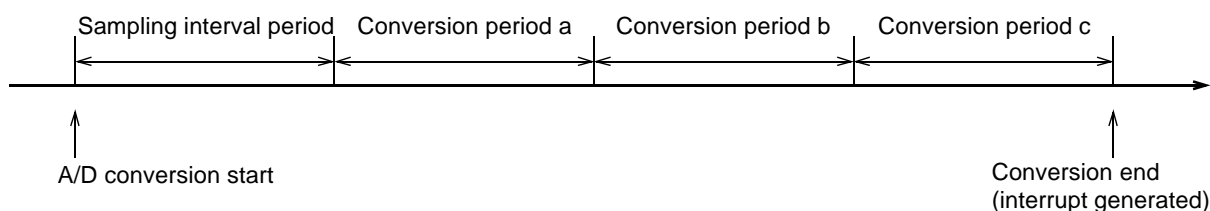
bit	15	14	13	12	11	10	9	8	
Address:000073H	SMP3	SMP2	SMP1	SMP0	CV03	CV02	CV01	CV00	ADCT2
	X	X	X	X	X	X	X	X	← Initial value
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	← Attributes

This register determines the length of the sampling period and conversion interval periods a to c (see Figure 2.9.3). The width of each interval can be expressed by the following formula:

(Register setting value + 1) × 0.1 μs (at machine clock of 10 MHz)

This register should be accessed by word access.

This register is undefined at reset. Settings must always be entered before starting A/D conversion.



**Fig. 2.9.3 Conversion time**

[Bits 15, 14, 13, 12] SMP3, SMP2, SMP1, SMP0 (sampling time):

These bits determine the length of the sampling period.

[Bits 11, 10, 9, 8] CV03, CV02, CV01, CV00 (convert time a):

These bits determine the length of conversion period a.

[Bits 7, 6, 5, 4] CV13, CV12, CV11, CV10 (convert time b):

These bits determine the length of conversion period b.

[Bits 3, 2, 1, 0] CV23, CV22, CV21, CV20 (convert time c):

These bits determine the length of conversion period c.

**Note:** For setup of sampling time and convert time a to c, refer to the electrical characteristics. Total A/D conversion time from its activation to the completion is equivalent to [sampling time + convert time a + convert time b + convert time c + 3 machine clocks].

**(4) Data Registers (ADTHx, ADTLx)**

These registers contain the digital values resulting from A/D conversion.

The ADCS2 register uses different data save formats, depending on the value of the CREG bit. (Switching may be made at any time without regard to A/D operation.)

Values in these registers are updated after the end of each single conversion cycle. The value is normally the last conversion value. The value of these registers is indeterminate at reset.

a) When CREG bit = '0'

bit	7	6	5	4	3	2	1	0	
Address:000074H	D7	D6	D5	D4	D3	D2	D1	D0	ADTLx
000076H	X	X	X	X	X	X	X	X	← Initial value
000078H	R	R	R	R	R	R	R	R	← Attributes

bit	15	14	13	12	11	10	9	8	
Address:000075H	0	0	0	0	0	0	D9	D8	ADTHx
000077H	0	0	0	0	0	0	X	X	← Initial value
000079H	-	-	-	-	-	-	R	R	← Attributes

The ADTHx register corresponds to the upper 2 bits of the conversion value, and the ADTLx register to the lower 8 bits.

Bits 15 to 10 of the ADTLx register have the read value '0.'

b) When CREG bit = '1'

bit	7	6	5	4	3	2	1	0	
Address:000074H	D7	D6	D5	D4	D3	D2	D1	D0	ADTLX
000076H	X	X	X	X	X	X	X	X	← Initial value
000078H	R	R	R	R	R	R	R	R	← Attributes

bit	15	14	13	12	11	10	9	8	
Address:000075H	0	0	0	0	0	0	0	0	ADTHX
000077H	0	0	0	0	0	0	0	0	← Initial value
000079H	-	-	-	-	-	-	-	-	← Attributes

The ADTLx register corresponds to the upper 8 bits.

The ADTHx register has the read value '00H.'

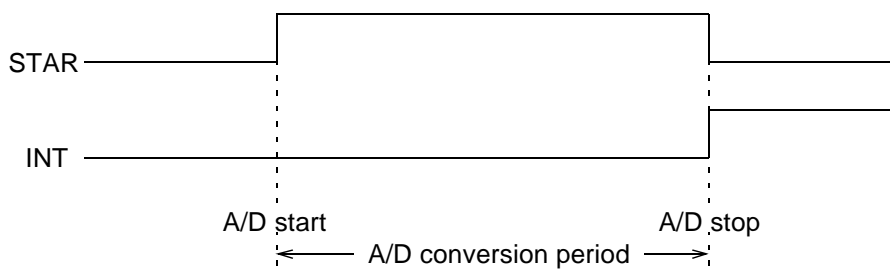
## 2.9.4 Operating Description

The A/D converter uses the 10-bit series-parallel method, and has a minimum conversion time of 1.25 $\mu$ s. This A/D converter provides a selection of single conversion mode and scan conversion mode.

### (1) Single Conversion Mode

When the SCAN bit in the control & status register is set to '0' single conversion mode is selected. A single conversion is performed on the signal from the analog input channel selected by bits ACS2 to ACS0 in the control & status register, and then a single conversion ends.

(Example) ACS=010<sub>B</sub> (AN2 selected)



**Fig. 2.9.4a Single Conversion Mode**

When A/D conversion starts, the analog signal input from AN2 is converted to digital expression, then the operation ends and the INT bit is set.

The interrupt request flag is set after each single conversion.



**(2) Scan Conversion Mode**

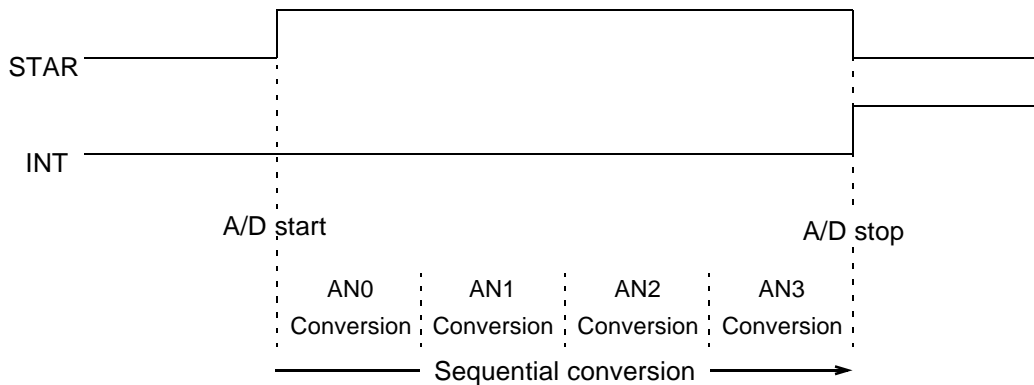
When the SCAN bit in the control & status register is set to '1' scan conversion mode is selected.

Scan conversion begins with the signal from the analog input channel selected by bits ACS2 to ACS0 in the control & status register. The last channel scanned depends on the setting of the ACS2 to ACS0 bits. Up to 4 channels may be scanned. Note, however, that the MB90242A series has only 6 channels, so that AN6 and AN7 can only be scanned in a two-channel scan.

**Table 2.9.4 Channel Settings in Scan Conversion Mode**

ACS2	ACS1	ACS0	Start channel	End channel	Remarks
0	0	0	AN0	AN3	4-channel scan conversion
0	0	1	AN1	AN3	3-channel scan conversion
0	1	0	AN2	AN3	2-channel scan conversion
0	1	1	AN3	AN3	1-channel single conversion
1	1	0	AN6	AN7	2-channel scan conversion
1	1	1	AN7	AN7	1-channel single conversion

(Example) ACS=000<sub>B</sub> (Scan conversion from AN0 to AN3)



**Fig. 2.9.4b Scan Conversion Mode**

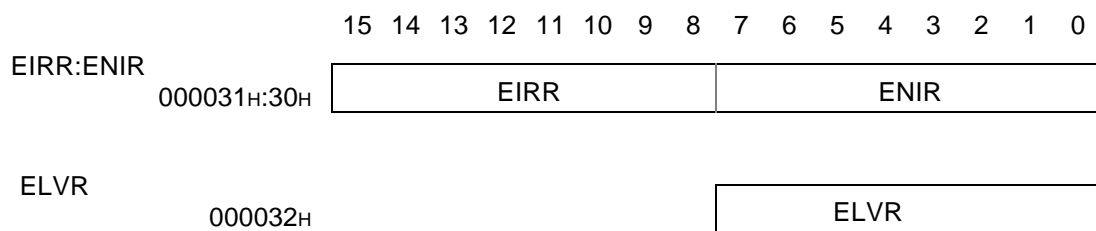
When A/D conversion starts channels AN0 to AN3 are scanned in sequence, then after all scanning is ended the INT bit is set and the operation ends.

The interrupt request flag is set when scan conversion ends.

## 2.10 External Interrupts

The DTP (Data transfer peripheral) module is a peripheral circuit placed between external peripheral resources and the F<sup>2</sup>MC-16F CPU to receive the DMA requests and interrupts generated from the external peripheral resources and pass them to the F<sup>2</sup>MC-16F CPU in order to trigger the extended intelligent I/O services or interrupt processing as necessary. Two request levels (H and L) can be handled for extended intelligent I/O services, and four (H and L, plus rising and falling edges) are handled for external interrupt requests. In addition, this module has the ability to initiate extended intelligent I/O services or interrupt processing. Only factor detection is enabled for signal pins corresponding to other bits.

### 2.10.1 Register List



### 2.10.2 Block Diagram

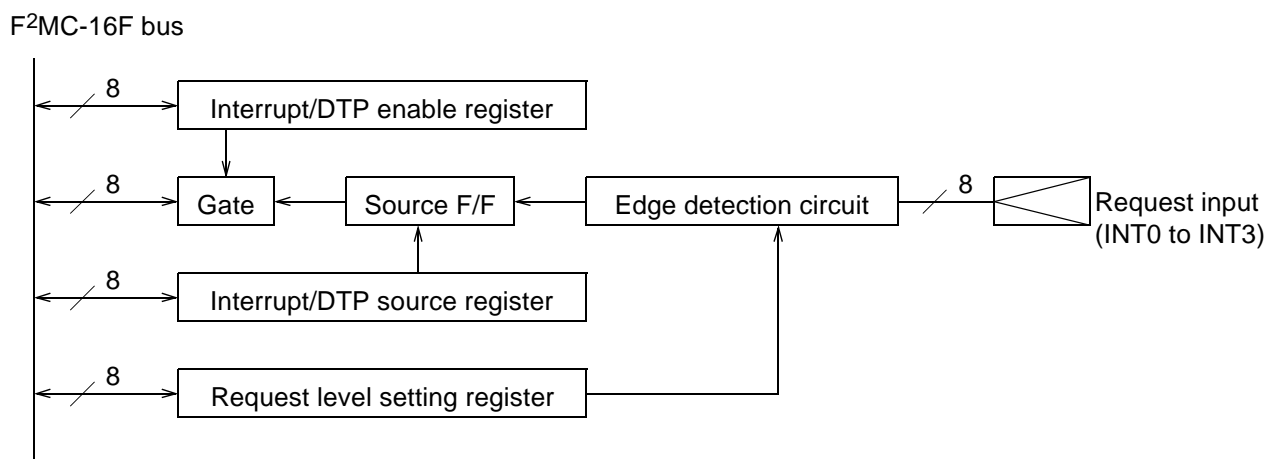


Fig. 2.10.1 Block Diagram

### 2.10.3 Detailed Register Description

#### (1) ENIR (Interrupt/DTP Enable Register)

##### ■ Register Allocation

Interrupt/DTP enable register	7	6	5	4	3	2	1	0	← Bit no.
Address : 000030H	-	-	-	-	EN3	EN2	EN1	EN0	ENIR
Read/write ⇒	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(-)	(0)	(0)	(0)	(0)	

##### ■ Register Description

The ENIR register determines the operation of the external interrupt/DTP request input functions that generate interrupt requests to the interrupt controller. When '1' is written to a bit in this register, the corresponding pin is used as an external interrupt/DTP request input and has the function of generating interrupt requests to the interrupt controller. When '0' is written to a bit, the corresponding pins will retain external interrupt/DTP request input factors, but will not generate requests to the interrupt controller.

#### (2) EIRR (Interrupt/DTP Factor Register)

##### ■ Register Allocation

Interrupt/DTP register	15	14	13	12	11	10	9	8	... ← Bit no.
Address : 000031H	-	-	-	-	ER3	ER2	ER1	ER0	EIRR
Read/write ⇒	(-)	(-)	(-)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(-)	(-)	(-)	(-)	(0)	(0)	(0)	(0)	

##### ■ Register Description

The EIRR register can be read-accessed to show the presence of external interrupt/DTP requests, and can be write-accessed to clear the flip-flop settings that indicate these requests. A read value of '1' indicates that an interrupt/DTP request has been generated at the pin corresponding to that bit. Writing '0' to this register clears the request flip-flop setting at the corresponding bit. Writing '1' has no effect. With read-modify-write commands, the read value is always '1.'

**(3) ELVR (Request Level Setting Register)**

■ Register Allocation

Request level setting register	7	6	5	4	3	2	1	0	← Bit no.
Address : 000032H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA	ELVR
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)	

■ Register Description

The ELVR register is used to select the mode of request detection. 2 bits are assigned to each pin, and are set in combination as shown below. When the request input is in level, clearing the register setting will reset the signal if the input is at active level.

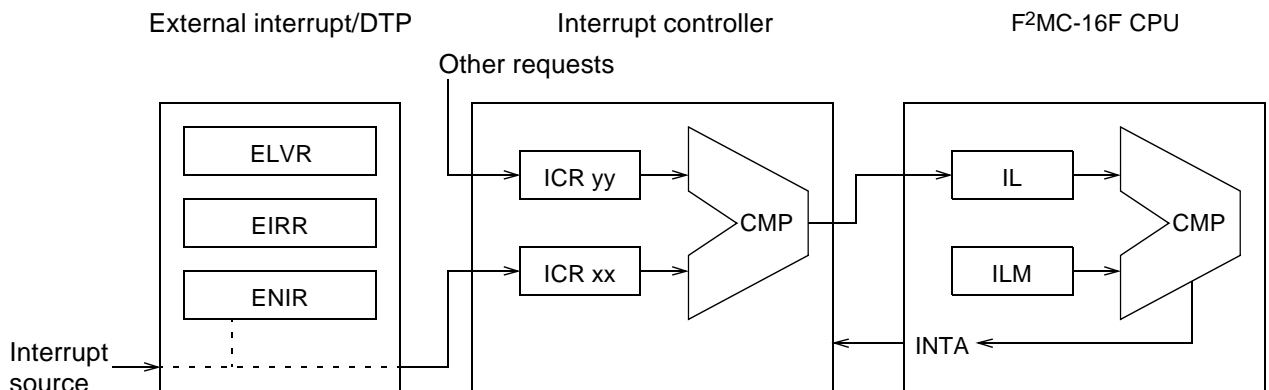
**Table 2.10.1 ELVR Bit Settings**

LBx	LAx	Detection mode
0	0	Interrupt if L level signal
0	1	Interrupt if H level signal
1	0	Interrupt if rising edge
1	1	Interrupt if falling edge

**2.10.4 Operating Description**

**(1) External Interrupt Operations**

When external interrupt requests are used, this resource will generate an interrupt request signal to the interrupt controller whenever an interrupt designated in the ELVR register is received at the corresponding input pin. Interrupts generated simultaneously are assigned priority values by the interrupt controller. If an interrupt from this resource has the current highest priority, the interrupt controller will generate an interrupt request to the F<sup>2</sup>MC-16F CPU, which will compare the interrupt with the ILM bit in its own internal CCR register. If the interrupt has a higher priority level than the ILM bit, the CPU will start the hardware interrupt processing microprogram as soon as the current instruction being executed is terminated.



**Fig. 2.10.2 External Interrupt Operation**

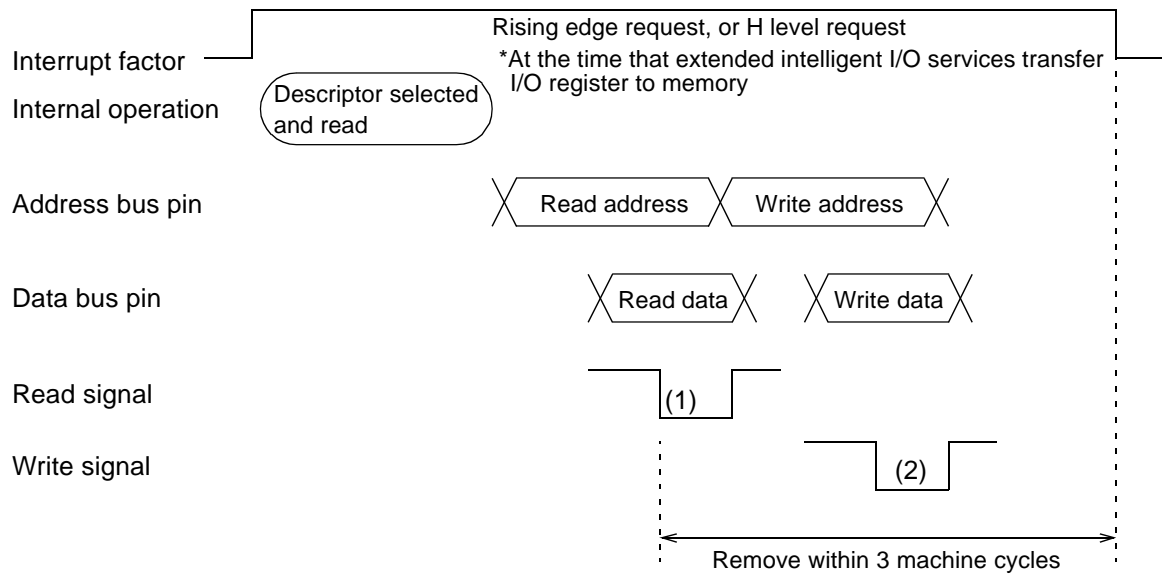
## 2.10 External Interrupts

In the hardware interrupt processing microprogram, the CPU reads the information in the ISE bit from the interrupt microcontroller to determine that the current request is concerned with interrupt processing, and then branches to the interrupt processing microprogram. The interrupt processing microprogram executes the user-defined interrupt processing program by reading interrupt vector areas and generating interrupt acknowledge signals to the interrupt controller on the interrupt microcontroller, and then by transferring to the program counter the jump destination address of the macro command generated by the vectors.

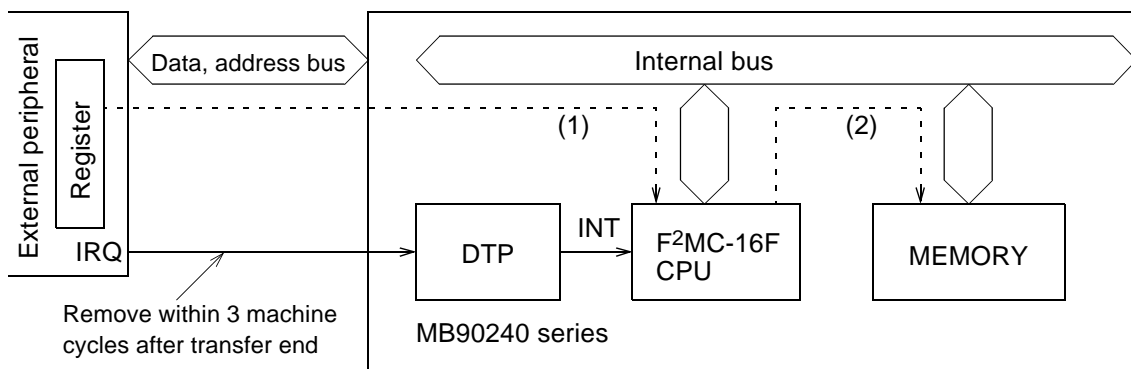
### (2) DTP Operation

Extended intelligent I/O service processing is started with an initialization procedure that sets the I/O address pointer in the extended intelligent I/O service descriptor to the address of the register allocated to addresses 000000<sub>H</sub> to 00FFFF<sub>H</sub>, and at the same time sets the buffer address pointer to the start address of the memory buffer.

The DTP operating sequence is entirely identical to external interrupt processing up to the point that the F<sup>2</sup>MC-16F CPU starts the hardware interrupt processing microprogram. At this point, the value of the ISE bit that the F<sup>2</sup>MC-16F CPU reads within the hardware interrupt processing microprogram indicates DTP processing, and accordingly control is passed to the extended intelligent I/O service processing microprogram. When the extended intelligent I/O service starts up, read or write signals are sent with addressing that designates specific external peripheral resources, and data is transferred to and from the MB90242A chip. The interrupt request to the MB90242A chip should be dismissed within three machine cycles after the transfer to or from the external peripheral resources. Once the transfer is complete, the descriptor is updated, and then a signal for clearing the transfer factor is sent to the interrupt controller. This resource receives the clearing signal, clears the flip-flop setting where the factor has been stored, and awaits the next request from the signal pin. For a detailed description of extended intelligent I/O service processing, see the MB90200 Programming Manual.



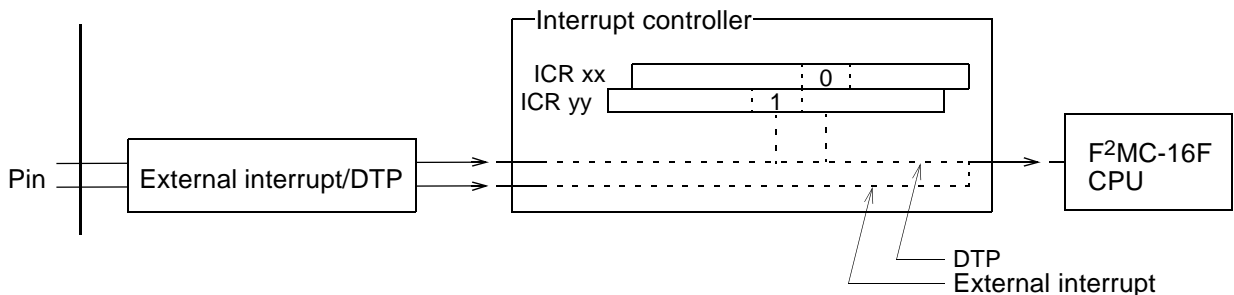
**Fig. 2.10.3 External Interrupt Dismissal Timing at End of DTP Operation**



**Fig. 2.10.4 Simplified Example of Interface and External Peripheral Resource**

**(3) Switching between External Interrupt Requests and DTP Requests**

Switching between external peripheral requests and DTP requests is accomplished by setting the ISE bit in the ICR register that corresponds to this resource. Each signal pin is provided with an individual ICR register, so that DTP can be requested for that signal pin by writing '1' to the ISE bit in the corresponding ICR register, and '0' can be requested by writing '0.'



**Fig. 2.10.5 Switching between External Interrupt Requests and DTP Requests**

## 2.10.5 Notes on Usage

### (1) Conditions for External Peripheral Resources Connected for Use with DTP

External peripheral resources supported by DTP must be able to clear requests automatically after transfers have been completed. In addition, unless transfer requests can be removed within 3 machine cycles (tentative value) after the start of transfer operation, the resource will react as if the next transfer request has been generated.

### (2) Recovery from Standby

When external interrupts are used to recover from standby status in clock stop mode, the input signal should be an H level request. Use of an L level signal may result in abnormal operation. Edge requests cannot be used to recover from standby status in clock stop mode. The MB90242A series has four external interrupt signal pins, however, only CH0 or CH1 should be used to recover from standby status. CH2 and CH3 should be set to Hi-Z status to cut off input signals at standby.

### (3) External Interrupt/DTP Operating Procedure

External interrupt/DTP register settings should be made using the following procedure.

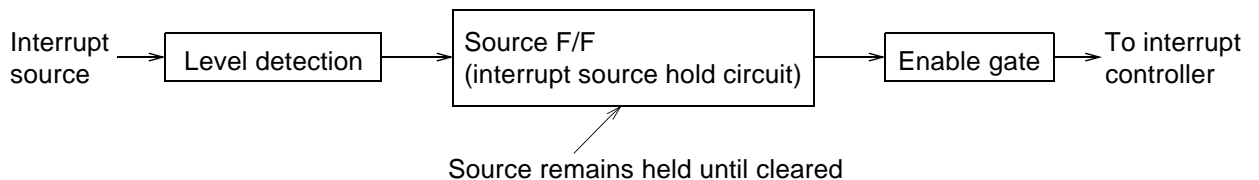
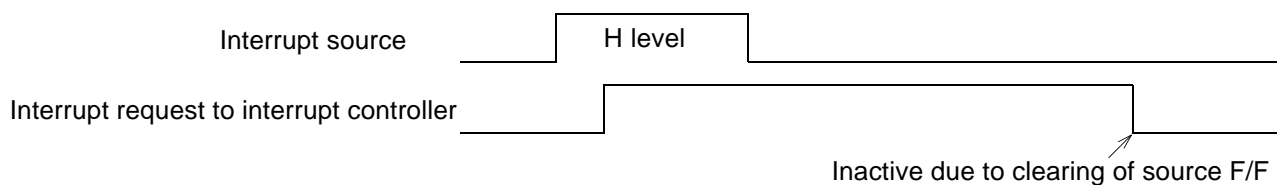
4. Disable the corresponding bits in the enable register.
5. Set the corresponding bits in the request level setting register.
6. Clear the corresponding bits in the interrupt source register.
7. Enable the corresponding bits in the enable register.

(Note that steps 3 and 4 may be done with simultaneous write operations using word access.)

When making settings to registers within this resource, it is first necessary to make a 'disable' setting in the enable register. Also, it is necessary to clear the interrupt source register before returning the enable register to 'enable' status. This is in order to avoid setting erroneous interrupt sources when making register settings or enabling interrupt status.

**(4) External Interrupt Request Levels**

- (1) When the interrupt level uses edge interrupt detection, the pulse width must be at least three machine cycles to allow the edge detection function to operate.
- (2) When the interrupt input level involves a level setting, incoming external interrupt signals are retained, even after removal, in an internal interrupt source hold circuit, and the request to the interrupt controller remains active. Thus to dismiss requests to the interrupt controller, it is necessary to clear the interrupt source hold circuit.

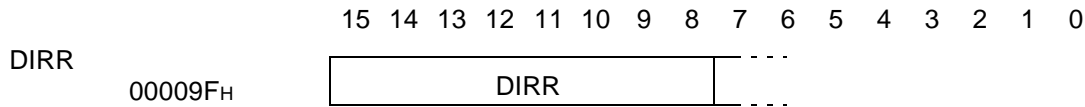
**Fig. 2.10.6 Clearing Interrupt Source Hold Circuit when Level Setting is Used****Fig. 2.10.7 Timing of Interrupt Source and Interrupt Request to Interrupt Controller when Interrupt Enabled**



## 2.11 Delayed Interrupt Generator Module

The Delayed interrupt generator module is used to generate interrupts for task switching. This module can be used to generate or delete software interrupt requests to the F<sup>2</sup>MC-16F CPU.

### 2.11.1 Register List



### 2.11.2 Block Diagram

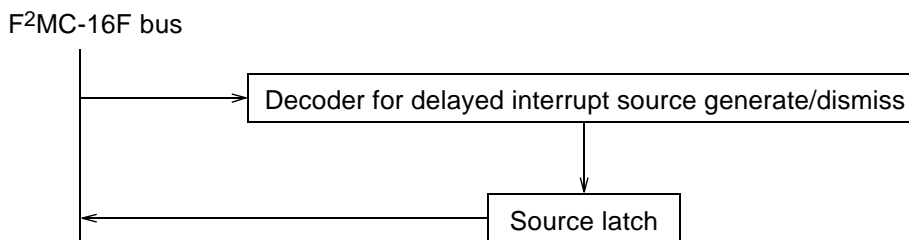


Fig. 2.11.1 Block Diagram

### 2.11.3 Detailed Register Description

#### (1) DIRR (Delayed Interrupt Source Generate/Dismiss Register)

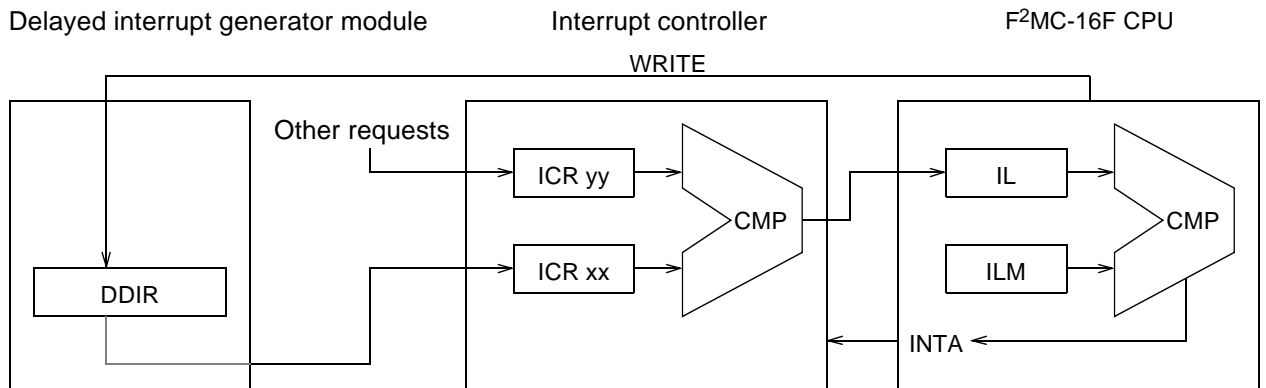
	15	14	13	12	11	10	9	8	...	← Bit no.
Address : 00009FH	-	-	-	-	-	-	-	R0	...	TBTC
Read/write ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(R/W)	...	
Initial value ⇒	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(0)	...	

The DIRR register controls the generation and dismissal of delayed interrupt requests. Writing '1' to this register causes a delayed interrupt request to be generated, and writing '0' dismisses the delayed interrupt request. After a reset, this register is in a state of interrupt source being dismissed. Either '0' or '1' may be written to the reserved bits, however in view of future extension it is recommended that set-bit and clear-bit instruction be used when accessing this register.

## 2.11.4 Operating Description

### (1) Delayed Interrupts

When the CPU uses software commands to write '1' to the corresponding bit in the DIRR register, the request latch is set in the delayed interrupt generator module, and an interrupt request is sent to the interrupt controller. If other interrupt requests have a lower priority, or if there are no other requests, an interrupt is then generated and sent to the F<sup>2</sup>MC-16F CPU by the interrupt controller. Then, the F<sup>2</sup>MC-16F CPU compares the request with the ILM bit in its own internal CCR register. If the request level is higher than the ILM bit, then as soon as the current instruction finishes executing, the hardware interrupt processing microprogram is started. This causes the interrupt processing routine for this interrupt to be executed.



**Fig. 2.11.2 Delayed Interrupt Generator Operation**

During the interrupt processing routine, the source of interrupt is cleared by writing '0' to the corresponding bit in the DDIR register, and by switching tasks.

## 2.11.5 Notes on Usage

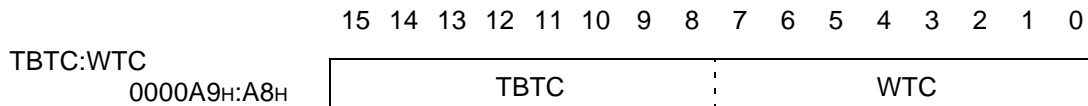
### (1) Delayed Interrupt Request Latch

This latch is set by writing '1' to the corresponding bit in the DIRR register, and cleared by writing '0' to the same bit. Therefore the interrupt processing routine must contain software to clear the source of interrupt within the interrupt processing routine, or else recovery from interrupt processing will only result in the immediate start of another interrupt processing routine. Users should ensure that interrupt processing software is programmed to avoid this problem.

## 2.12 Watchdog Timer, Timebase Timer Functions

The watchdog timer is composed of a 2-bit watchdog counter that uses as a clock source the carry signal of the 18-bit timebase timer, plus a control register and a watchdog reset control unit. The timebase timer consists of an 18-bit timer plus circuits that control interrupt signals at intervals. Figure 2.12.1 shows the configuration of the watchdog timer and timebase timer.

### 2.12.1 Register List



### 2.12.2 Block Diagram

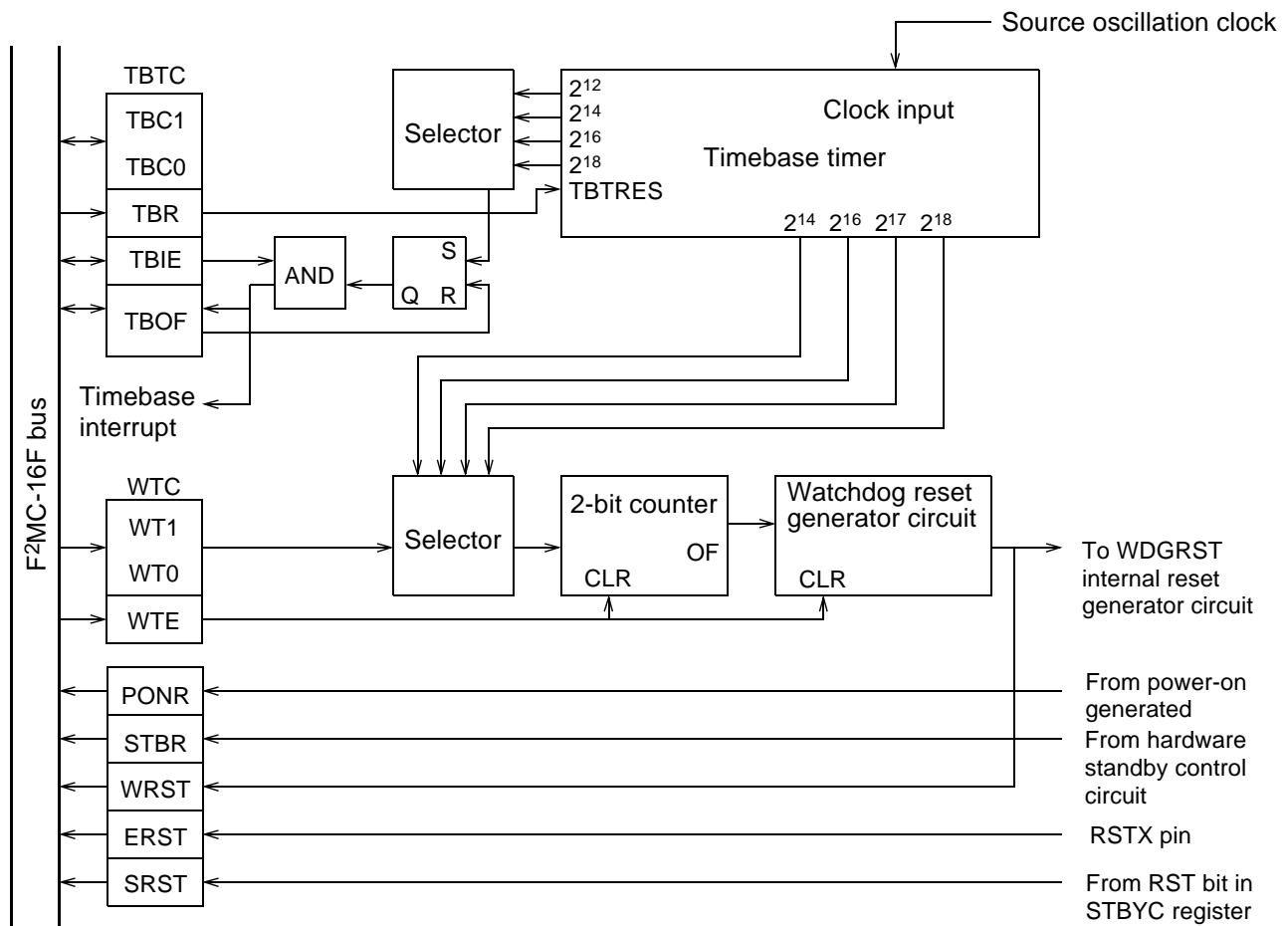


Fig. 2.12.1 Watchdog Timer and Timebase Timer Block Diagram

### 2.12.3 Detailed Register Description

#### (1) WTC (Watchdog timer control register)

##### ■ Register Allocation

Watchdog timer control register	7	6	5	4	3	2	1	0	← Bit no.
Address : 0000A8H	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	WTC
Read/write ⇒	(R)	(R)	(R)	(R)	(R)	(W)	(W)	(W)	
Initial value⇒	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

##### ■ Register Description

This register comprises bits used to control watchdog timer-related functions, plus bits that identify reset sources.

##### ■ Bit Descriptions

[Bits 7 to 3] PONR, STBR, WRST, ERST, SRST

These bits are flags that indicate reset sources. When a reset source occurs, the corresponding bit is set as shown in Table 2.11.1. These bits are all cleared to '0' after a WTC read operation. This register is read-only. Note that at power-on, the value of bits other than the power-on bit is not warranted. Therefore software should be programmed to ignore the value of other bits when the PONR bit is '1.'

**Table 2.12.1 Reset Source Bits and Reset Sources**

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

\* indicates prior value retention.

[Bit 2] WTE

When the watchdog timer is in stop status, it can be returned to operating status by writing '0' to this bit. The second and subsequent writing of '0' will clear the watchdog timer counter. Writing '1' to this bit has no effect.

The watchdog timer can be placed in stop status by power-on, hardware standby or watchdog timer reset. The read value is '1.'

## 2.12 Watchdog Timer, Timebase Timer Functions

[Bit 1, 0] WT1, WT0

These bits select the interval time used for the watchdog timer. To be valid, data must be written to this bit when the watchdog timer starts. All write data written at other times is ignored.

Table 2.13.2 shows interval time settings.

They are write-only bits.

**Table 2.12.2 Watchdog Timer Interval Select Bit**

WT1	WT0	Interval time (machine clock: 16 MHz)	
		Min	Max
0	0	approx. 3.58 ms	approx. 4.61 ms
0	1	approx. 14.33 ms	approx. 18.44 ms
1	0	approx. 28.67 ms	approx. 36.87 ms
1	1	approx. 57.34 ms	approx. 73.73 ms

**Note:** The maximum interval time assumes that the timebase counter is not reset while the watchdog timer is operating.

### (2) TBTC (Timebase timer control register)

#### ■ Register Allocation

Timebase timer control register	15	14	13	12	11	10	9	8	...	← Bit no.
Address : 0000A9H	Reserved	-	-	TBIE	TBOF	TBR	TBC1	TBC0	...	TBTC
Read/write ⇒	(R/W)	(-)	(-)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	...	
Initial value ⇒	(0)	(X)	(X)	(0)	(0)	(0)	(0)	(0)	...	

#### ■ Register Description

This register controls the operation of the timebase timer, as well as the interval interrupt time.

#### ■ Bit Description

[Bit 15] Reserved bit

This is a test bit. The write value should always be '0.'

[Bits 14 to 13] Not used

[Bit 12] TBIE

This bit enables interval interrupts from the timebase timer. Write '1' to enable interrupt, and '0' to disable it. The initial value is '0' after reset. This bit is read/write enabled.

**[Bit 11] TBOF**

This is the timebase timer interrupt request flag. If the TBIE bit has the value '1' then setting this bit to '1' will cause an interrupt request to be generated. This bit will be set to '1' at every interval defined by the TBC1, TBC0 bits. This bit can be cleared by writing '0,' changing into stop mode or hardware standby mode, or by a reset. Writing '1' to this bit is meaningless.

With read-modify-write commands, the read value is always '1.'

**[Bit 10] TBR**

This bit clears all bits in the timebase timer counter to '0.' Write '0' to this bit to clear the timebase timer counter. Writing '1' to this bit is meaningless. The read value is always '1.'

**[Bits 9, 8] TBC1, TBC0**

These bits specify the time interval settings for the timebase timer. Interval settings are shown in Table 2.11.3. The value is initialized to '00B' by a reset. This bit is read/write enabled.

**Table 2.12.3 Timebase Timer Interval Selection**

TBC1	TBC0	Interval time (machine clock at 16 MHz)
0	0	0.256 ms
0	1	1.024 ms
1	0	4.096 ms
1	1	16.384 ms

## 2.12.4 Operating Description

### (1) Watchdog Timer

The watchdog timer function is used to detect program loops. If a program loop condition causes the designated time to elapse before '0' is written to the WTE bit in the watchdog timer, the watchdog timer will generate a watchdog reset request.

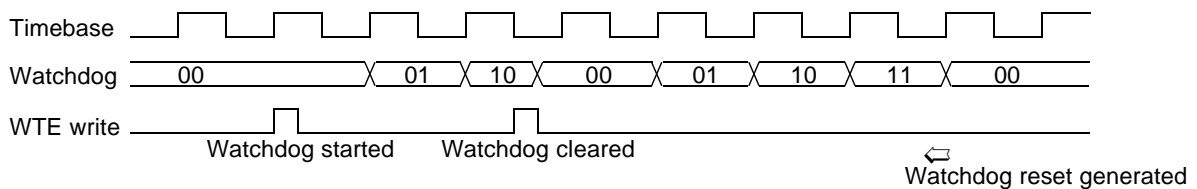
#### ■ Startup

When the watchdog timer is stopped, it can be started by writing '0' to the WTE bit in the WTC register. At this moment, the value of the WT0, WT1 bits is used to determine the watchdog timer reset interval. The interval can only be defined by the data values effective at the time of startup.

#### ■ Watchdog Timer Reset

Once the watchdog timer has been started, the program must clear the 2-bit watchdog counter at regular intervals. Specifically, this means that '0' must be written to the WTE bit in the WTC register at regular intervals. The watchdog counter is configured as a 2-bit counter and uses the carry signal of the timebase counter as its clock source. Thus when the timebase timer is cleared, the watchdog reset generation time may be longer than the setting.

Figure 2.13.2 shows the operation of the watchdog timer.



**Fig. 2.12.2 Watchdog Timer Operation**

#### ■ Watchdog Stop

The watchdog timer is initialized and placed in stop status after startup, by power-on, hardware standby or watchdog reset. The watchdog counter is cleared by a reset from an external pin or software signal, but in these cases the watchdog function does not stop.

#### ■ Other

The watchdog counter may be cleared by writing '0' to the WTE bit, but is also cleared by reset, transition to sleep mode or stop mode, or by a hold acknowledge signal.

## (2) Timebase Timer

The timebase timer functions as clock source for the watchdog counter, timer for oscillation stabilization time, and as an interval timer generating interrupts at designated regular intervals.

### ■ Timebase Timer

The timebase timer consists of an 18-bit counter that counts the source oscillation input produced by the machine clock. The count operation continues as long as the source oscillation signal is input. The timebase timer is cleared by a power-on reset, transition to stop mode or hardware standby mode, or by writing '0' to the TBR bit in the TBTC register.

Clearing the timebase timer affects the watchdog counter and interval interrupt functions that operate using the output from the timebase timer.

### ■ Interval Interrupt Function

This function generates interrupts at regular intervals according to the time base counter carry signal. The TBOF flag is set at an interval determined by the TBC1, TBC0 bits in the TWC register. The setting of this flag is based on the last time that the timebase timer was cleared.

In transition to stop mode or hardware standby mode, the timebase timer is used to time the oscillation stabilization period after recovery. Therefore the TBOF flag is cleared simultaneously with mode transition.



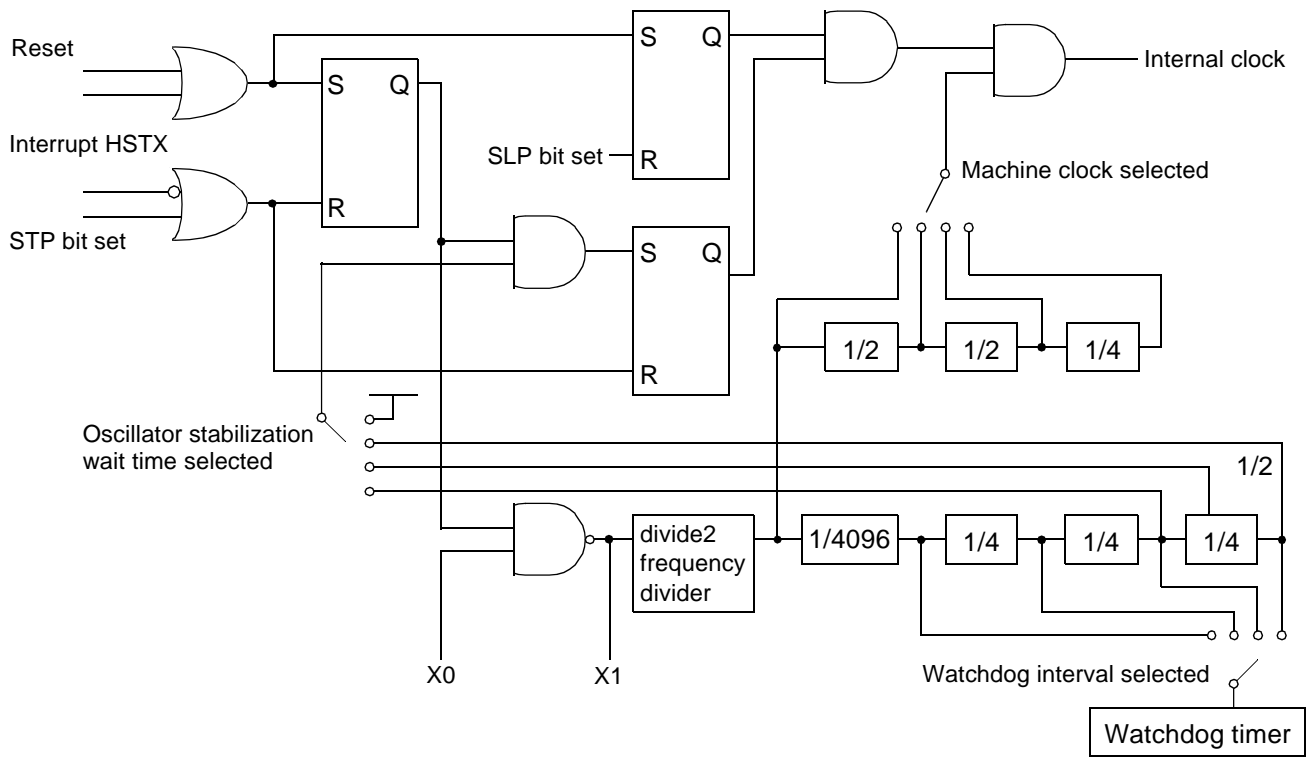


## Chapter 3: Operation

### 3.1 Clock Generator

The clock generator unit controls internal clock operations including sleep mode, stop mode and gear functions. The internal clock is referred to as the machine clock, and one period of this signal is called a machine cycle.

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



**Fig. 3.1.1 Clock Generator Unit Block Diagram**

## 3.2 Reset

### 3.2.1 Reset Factor Generation

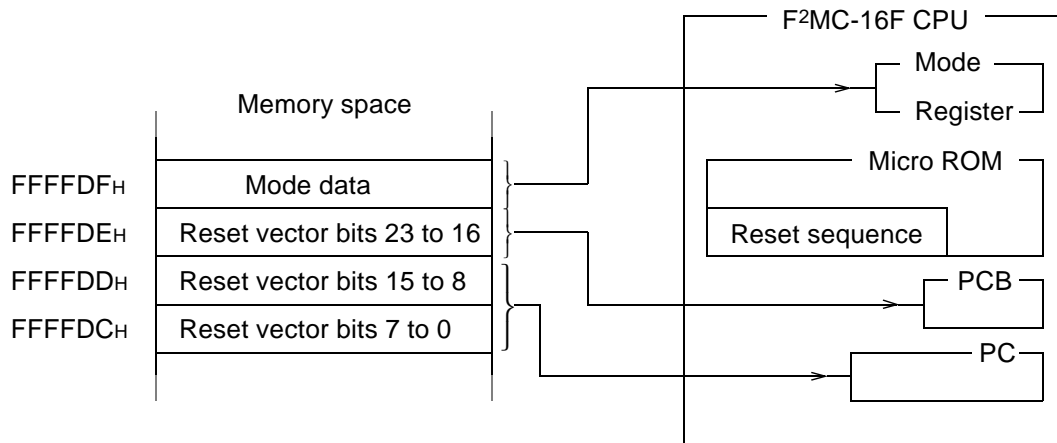
When a reset factor occurs, the reset device immediately stops processing, and goes into reset release standby mode. Resets are generated by the following multiple factors.

- Occurrence of a power-on reset
- Exit from hardware standby status
- Watchdog timer overflow
- Generation of an external reset request from the RSTX pin
- Generation of a reset request by software

When an external bus is used, the device where a reset factor occurs will generate an address that will be indeterminate all signals used for external bus access, such as the RDX and WRX signals, will be inactive.

### 3.2.2 Operation after Reset Release

Once a reset factor is removed, the reset device immediately outputs the address containing the reset vector, and the reset vector and mode data are fetched by the system. Reset vectors and mode data are allocated to the 4 bytes from FFFFDC<sub>H</sub> to FFFFDF<sub>H</sub>, and are transferred after reset release by the register hardware shown in Figure 3.2.1.



**Fig. 3.2.1 Reset Vector and Mode Data Location and Transfer Destinations**

### 3.2.3 Reset Factors

There are five types of reset factor, as listed in Table 3.2.1, each using different machine clock (gear function) and watchdog function default settings.

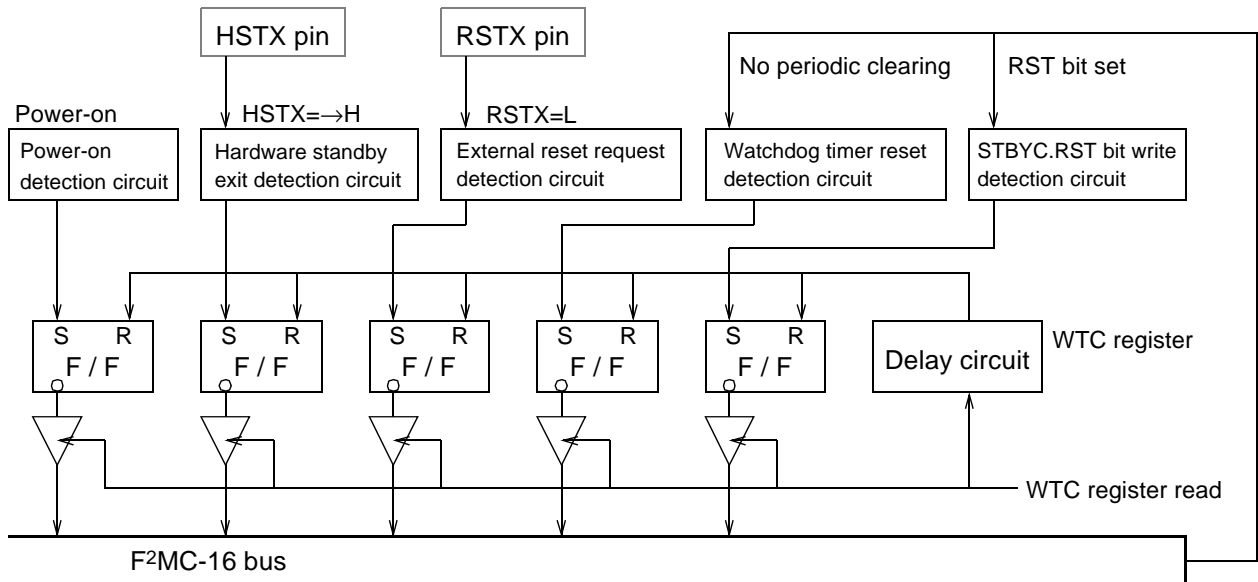
Reset factors can be determined from the reset factor bit in the watchdog control register.

**Table 3.2.4 Reset Factors**

Reset	Originating factor	Machine clock	Watchdog timer	Oscillation stabilization wait
Power-on reset	At power-up	'11'	Stop	Yes
Hardware standby	'L' level input to HSTX pin	'11'	Stop	Yes
Watchdog timer	Watchdog timer overflow	'11'	Stop	No
External pin	'L' level input to RSTX pin	Hold prior status	Hold prior status	No
Software	Write '0' to STBYC register RST bit	Hold prior status	Hold prior status	No

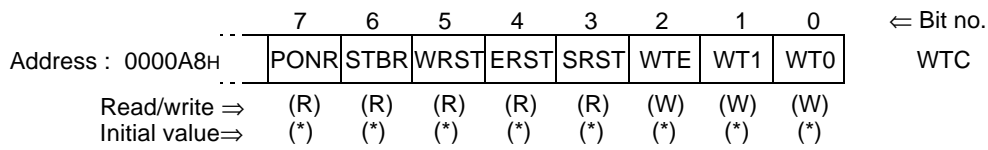
\* When an external pin reset signal is input during stop mode, an oscillation stabilization wait period is applied regardless of reset factor.

Figure 3.2.2 shows flip-flop settings corresponding to each reset factor. Each of these settings can be read from the watchdog timer control register, so that if it is necessary to determine a reset factor after the reset has been released, software commands can be used to process values read from this register and to branch to the appropriate program. The watchdog timer control register is illustrated in Figure 3.2.3.



**Fig. 3.2.2 Reset Factor Bit Block Diagram**

## 3.2 Reset



**Fig. 3.2.3 WTC (Watchdog Timer Control) Register**

The watchdog timer control circuit can handle multiple reset requests, by setting each of the corresponding reset factor bits in the watchdog timer control register. Thus if an external reset request and watchdog reset request occur at the same time, both the ERST and WRST bits will be set to '1.'

Note that an exception occurs with power-on resets, where if the PONR bit is set to '1' all other bits do not indicate their corresponding normal reset factors. For this reason, software should be programmed so that if the value of the PONR bit is '1' all other reset factor bits will be ignored.

**Table 3.2.5 Reset Factor Bit Values and Corresponding Reset Factors**

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

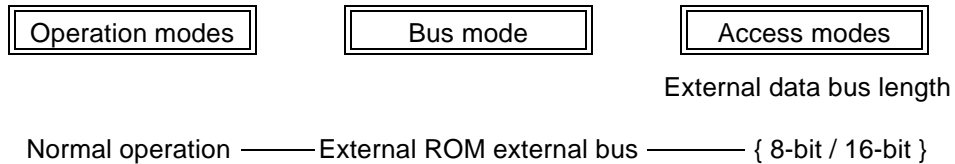
(an asterisk (\*) indicates prior value retained)

Reset factor bits can only be cleared by reading the watchdog timer control register, so that once a reset factor has occurred and the reset factor bit is set to '1' that value will remain '1' when any other reset factors occur.

## 3.3 Memory Access Mode

### 3.3.1 Mode Types

The F<sup>2</sup>MC-16F CPU uses separate modes for access method, access area and operation as shown in the following diagram.



**Fig. 3.3.1 Mode Types**

#### ■ Operating Modes

Operating modes control the operating status of the MB90242A device and are set by the mode setting pins MD2 to MD0.

#### ■ Bus Modes

Bus modes control external ROM operation and external access functions, and on the MB90242A are set for external ROM, external bus operation.

#### ■ Access Modes

Access modes control external data bus width, and are set by the mode setting pins MD2 to MD0 together with the value of the S0 bit in the mode data. The access mode may be selected to specify either 8-bit or 16-bit external data bus width.

### 3.3 Memory Access Mode

#### 3.3.2 Mode Pins

The mode setting pins MD2 to MD0 are three external pins that are used in combination to select the operating mode as shown in Table 3.3.1.

Input to the MD2 to MD0 pins should be connected directly to Vcc or GND.

**Table 3.3.1 Mode Pins and Mode Selection**

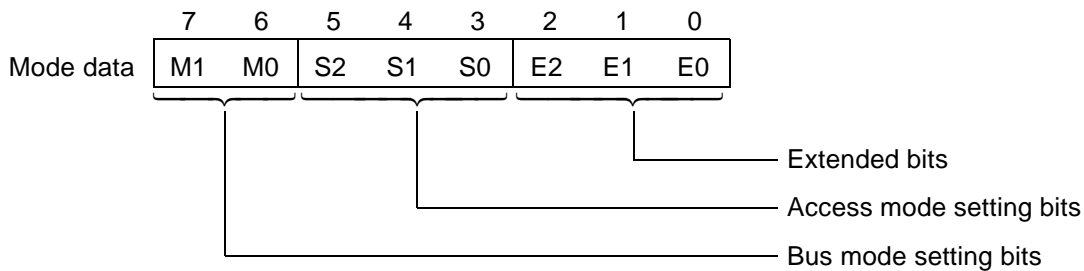
Mode pin setting			Mode name	Reset vector access area	External data bus width	Remarks
MD2	MD1	MD0				
0	0	0	External vector mode 0	External	16-bit	
0	0	1	External vector mode 1	External	8-bit	No upper data byte(s)
0	1	0	(Setting prohibited)			
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

#### 3.3.3 Mode Data

Mode data is located at memory address FFFFDF<sub>H</sub>, and is used to control CPU operation. During reset sequences this data is read and stored in 'Internal mode register on the device.' The contents of the mode register can only be changed by a reset sequence.

Settings made to this register are valid after the reset sequence ends, but some mode pin settings may render a portion of the values in this register invalid.

Figure 3.3.2 shows the values of bits in the mode data register.



**Fig. 3.3.2 Mode Data Structure**

■ Extended Bits (E2 to 0)

**Table 3.3.2 Extended Bit Functions**

E2	E1	E0	Function	Remarks
0	0	0	Internal RAM program use disabled	Attempts to execute programs using internal RAM will cause an exception interrupt.
1	1	0	Internal RAM program use enabled	No exception interrupts generated. Note that this setting will not allow use of tools for debugging of RAM programs.

■ Access Mode Setting Bits

These bits are used to set the access mode after the end of a reset sequence. Table 3.3.2 lists bit values and functions. The S2 and S1 bits should always be set to '00.'

**Table 3.3.2 Access Mode Setting Bits and Their Functions**

S2	S1	S0	Function	Remarks
0	0	0	External data bus 16-bit mode	
0	0	1	External data bus 8-bit mode	

■ Bus Setting Bits

These bits are used to set the operating mode after a reset sequence terminated. Table 3.3.3 lists bit values and functions.

**Table 3.3.3 Bus Mode Setting Bits and Their Functions**

M1	M0	Function	Remarks
0	0	(Setting prohibited)	
0	1	(Setting prohibited)	
1	0	External ROM, external bus mode	
1	1	(Setting prohibited)	

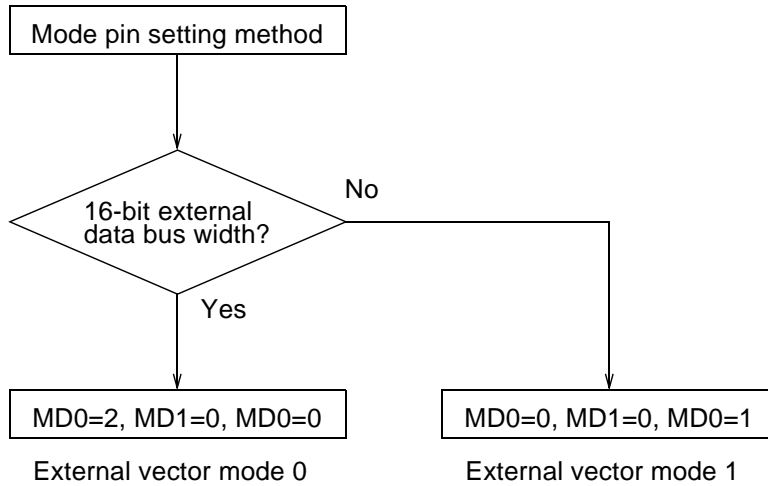


### 3.3.4 Mode Pin and Mode Data Settings

After reset release, both a mode pin setting and a mode data setting are required to allow the MB90242A device to operate in the desired mode.

#### ■ Mode Pin Setting Method

Mode pins are used to determine the method of fetching reset vectors and mode data.



**Fig. 3.3.3 Mode Pin Setting Method**

Mode pin settings result in different initial values for the upper address control register (HACR) and external pin control register (EPCR) after a reset release.

■ Mode Data Setting Method

Mode data is contained in the mode register and is used to determine the operating mode after fetching reset vectors and mode data.

In internal vector mode, mode data setting is made in address FFFFDF<sub>H</sub> in internal ROM. In external mode 0 or 1, mode data setting is made in address FFFFDF<sub>H</sub> in the external memory area.

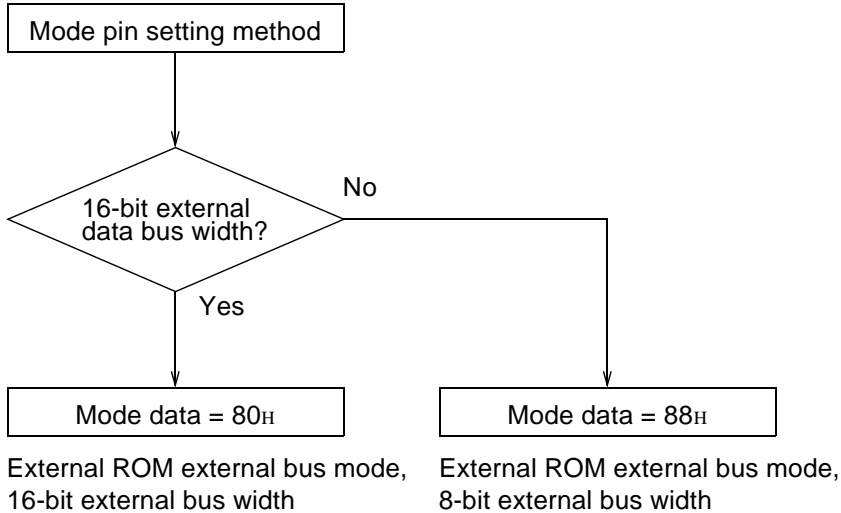


Fig. 3.3.4 Mode Data Setting Method

Memory space	
FFFFDF <sub>H</sub>	Mode data
FFFFDE <sub>H</sub>	Reset vector bits 23 to 16
FFFFDD <sub>H</sub>	Reset vector bits 15 to 8
FFFFDC <sub>H</sub>	Reset vector bits 7 to 0

Fig. 3.3.5 Mode Data and Reset Vector Memory Allocation

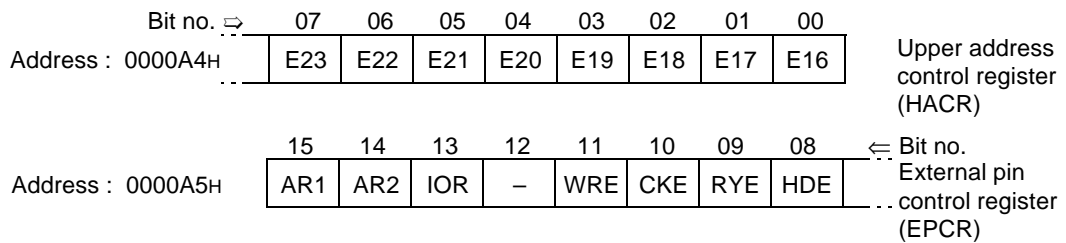
### 3.4 External Memory Access

Access to memory and peripheral devices outside the MB90242A device is enabled by the use of the following address/data/control signals.

- ▷ A23-A00 (P47 to P20) : Address output
- ▷ D15-D00 (P17 to P00) : Data input and output
- ▷ CLK (P50) : Machine cycle clock output
- ▷ RDY (P51) : External ready signal input
- ▷ WRHX (P54) : Data bus upper 8-bit write signal
- ▷ WRLX/WRX (P55) : Data bus lower 8-bit write signal
- ▷ RDX (P56) : Read signal

External memory access takes place by means of the external bus pin control circuit.

#### 3.4.1 Register Configuration



### 3.4.2 Block Diagram

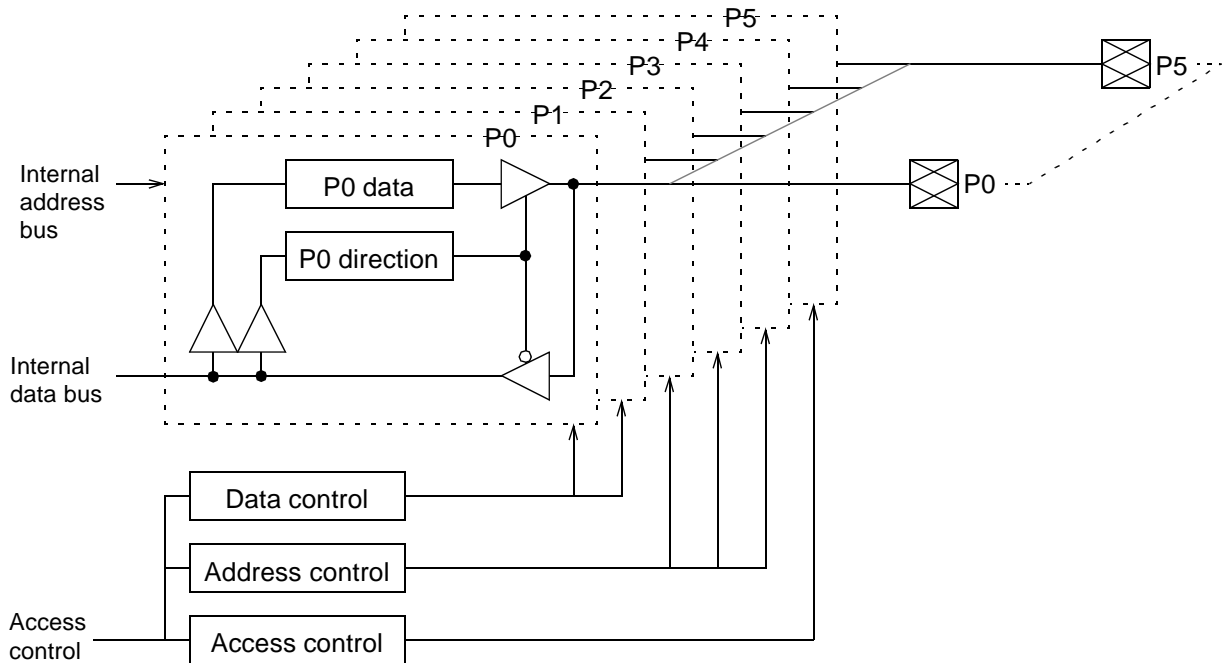


Fig. 3.4.1 Block Diagram

### 3.4.3 Detailed Register Description

#### (1) HACR (Upper address control register)

	07	06	05	04	03	02	01	00	←Bit no.
Address : 0000A4H	E23	E22	E21	E20	E19	E18	E17	E16	HACR (Upper address control register)
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value⇒	(*)	(*)	(*)	(*)	(*)	(*)	(*)	(*)	

This register controls output of addresses A23 to A16 to external circuits. Each bit represents one of the address bits A23 to A16 and controls the corresponding address output pin as follows.

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

All bits in this register are write-only, and have the read value '1' at all times.

The initial values of the bits in this register are determined according to the operating mode of the device at reset, as follows.

When reset vectors are read from internal ROM	1 (I/O port)
When reset vectors are read from external ROM	0 (address output)

**(2) EPCR (External Pin Control Register)**

Address : 0000A5H	15	14	13	12	11	10	09	08	←Bit no.
	AR1	AR0	IOR	Reserved	WRE	CKE	RYE	HDE	EPCR (External Pin Control Register)
Read/write ⇒	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)	
Initial value⇒	(*)	(*)	(0)	(-)	(0)	(*)	(0)	(0)	

This register determines control functions for external bus operation.

This register cannot be accessed when the MB90242A device is operating in single-chip mode. In this situation, all pins function as I/O ports regardless of the values in this register.

All bits in this register are write-only, and have the read value '1' at all times.

[Bits 15, 14] AR1, AR0

This bits determine the automatic wait function for external access to the area 000100H to FFFFFFFH. The two bits are used in combination to make the following settings.

AR1	AR0	Setting
0	0	Automatic wait prohibited
0	1	1-machine cycle automatic wait inserted for external access
1	0	2-machine cycle automatic wait inserted for external access
1	1	3-machine cycle automatic wait inserted for external access

\*: Initial values for these bits are determined by the operating mode of the device at reset, as follows.

When reset vectors are read from internal ROM	00B (automatic wait disabled)
When reset vectors are read from external ROM	11B (3-machine cycle wait inserted)

When the external ready function is enabled, a wait request from an external ready pin will cause the automatic wait period of the number of cycles designated by this setting to be followed by continuous wait status.

[Bit 13]:IOR

This bit will cause an automatic wait of 2 machine cycles to be inserted for external access to the area 0000C0H to 0000FFH.

0	Wait	(initial value)
1	No wait	

This bit is set to '0' by a reset.

When the external ready function is enabled, a wait request from an external ready pin will cause the automatic wait period of 2 machine cycles to be followed by continuous wait status.

[Bit 12]:Reserved

This bit is reserved. The write value must always be '0.'

[Bit 11]:WRE

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

0	I/O port (P55, P54) operation (write signal output disabled) (initial value)
1	Write strobe signal (WRHX/WRLX or WRX) output enabled

When the external data bus in 8-bit mode, this bit can enable use of P54 as an I/O port regardless of the bit value set.

This bit is set to '0' at a reset.

When this bit is set to '1,' it is recommended that the port 5 data register (PDR5) write strobe bit (P55 in 8-bit external data bus mode, and P54 and P55 in 16-bit external data bus mode) be set to '1' due to the possibility of spikes being generated at on the write strobe pin due to occurrence of a reset during system operation.

[Bit 10]:CKE

This bit controls the external clock signal (CLK) output as follows.

0	I/O port (P50) operation (clock output disabled)
1	Clock signal (CLK) output enabled

The initial value of this bit is determined by the operating mode of the device at reset, as follows.

When reset vectors are read from internal ROM	1 (I/O port)
When reset vectors are read from external ROM	0(clock signal output)

[Bit 9]:RYE

This bit controls the external read (RDY) input signal as follows.

0	I/O port (P51) operation (external RDY input disabled) (initial value)
1	External ready (RDY) signal input enabled

This bit is set to 0 at a reset.

### 3.4 External Memory Access

#### [Bit 8]:HDE

This bit enables input/output of signals related to the hold function. The value controls the hold request input signal (HRQ) and hold acknowledge output signal (HAKX) as follows.

0	I/O port (P53, P52) operation (hold function I/O disabled) (initial value)
1	Hold request (HRQ) input/hold acknowledge (HAKX) output enabled

These bits are set to '0' at a reset.

**[CAUTION]** In 16-bit bus mode, if the WRE bit is used to enable the WRHX and WRLX functions, P55 and P54 must not be set to output mode (Bits 5 and 4 in the DDR5 register should be set to '0').

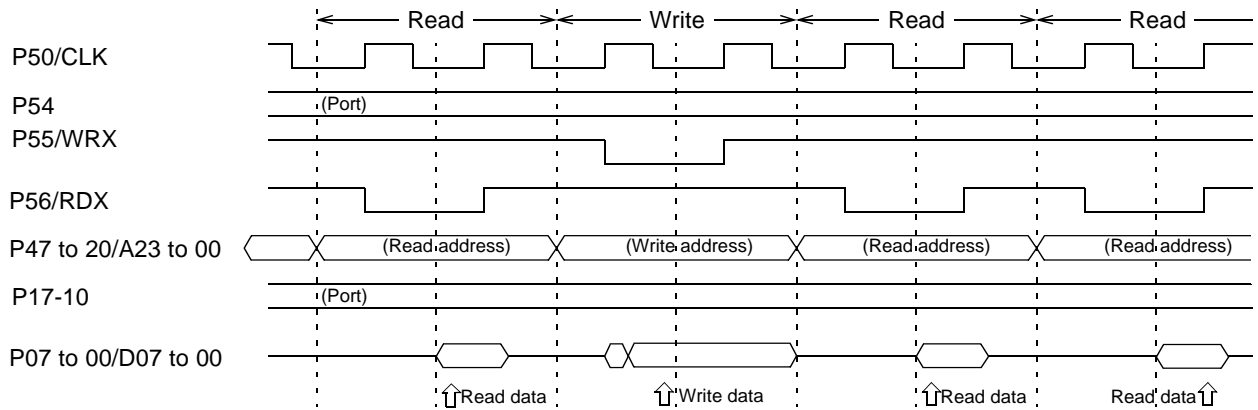
In 8-bit bus mode, if the WRE bit is used to enable the WRX function, P55 must not be set to output mode. (Bit 5 in the DDR5 register should be set to '0'.)

Also, even if the RYE and HDE bits are used to enable the RDY and HRQ input, the corresponding I/O port functions will be enabled. For this reason the corresponding bits in the DDR5 register must be set to '0' (input mode).

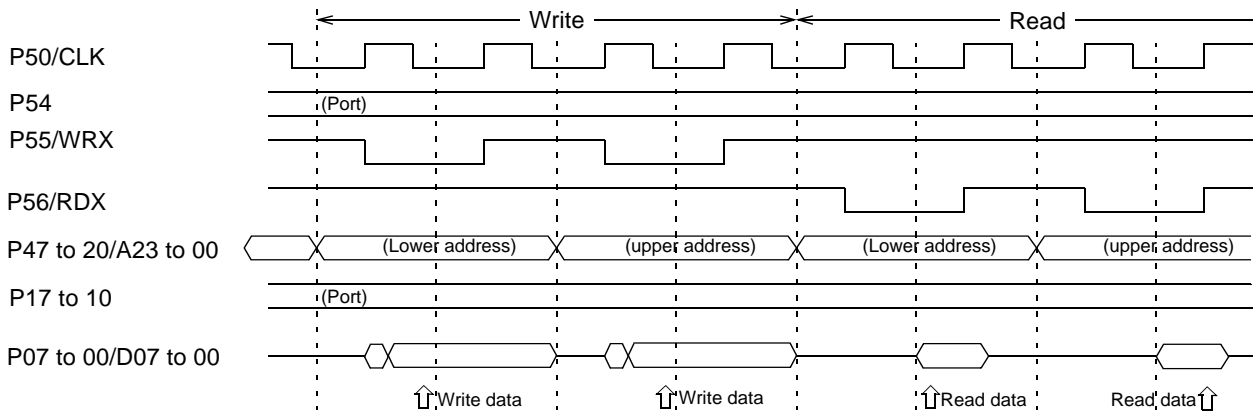
### 3.4.4 External Memory Access Control Signal

Access to external memory requires 2 cycles unless the ready function is used. Figure 3.4.2 presents an overview of the signal timing involved in external access.

#### ■ 8-Bit External Bus Mode

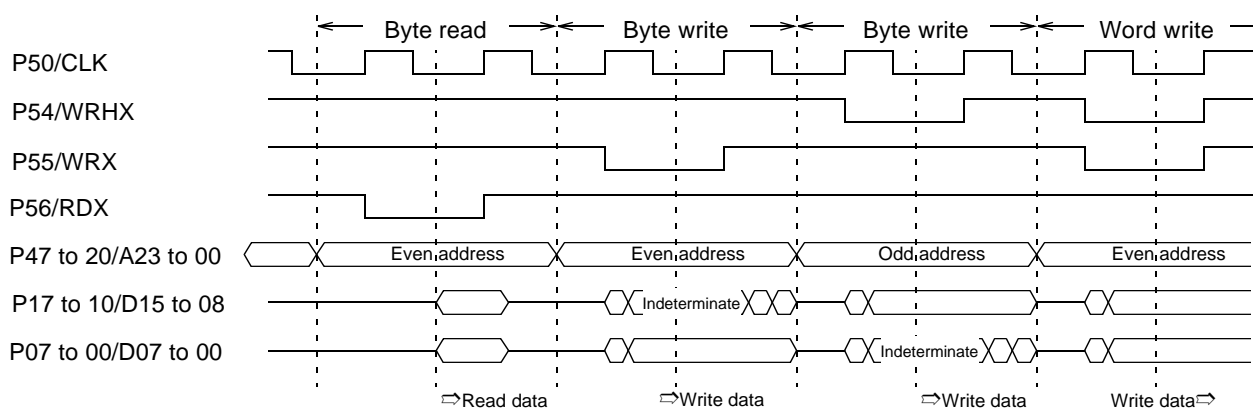


#### Timing for Byte Access



#### Timing for Word Access (the same is true for both even and odd addresses)

#### ■ 16-Bit External Bus Mode



\* Design the external circuit so that the word is always read.

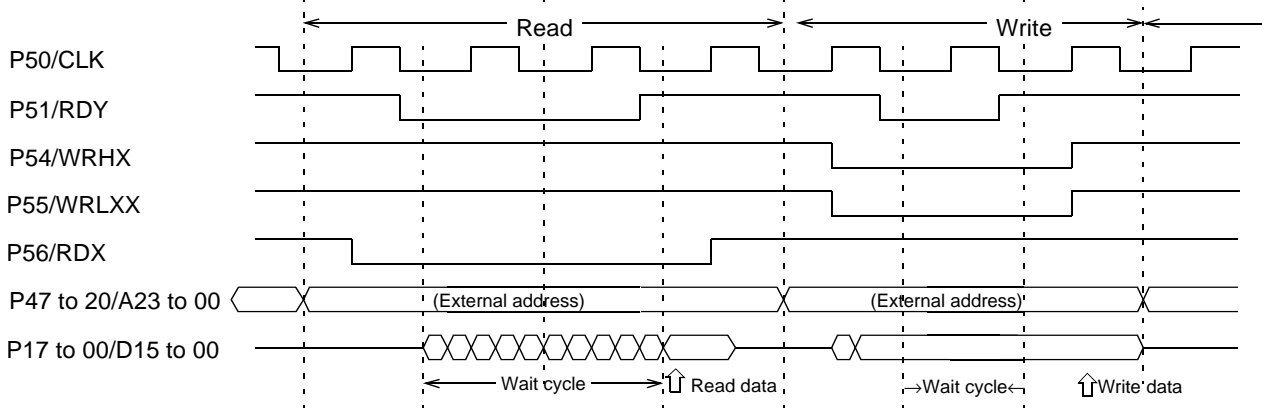
**Fig. 3.4.2 External Memory Access Timing Chart**

The P51/RDY pin or the external pin control register (EPCR) may be set to enable access to low-speed memory or peripheral circuits.



### 3.4.5 Ready Function

The RYE bit in the EPCR register can be set to '1' to extend the access cycle by creating a wait cycle for the period in which the P51/RDY pin is input at 'L' level during access to external circuits.



**Fig. 3.4.3 Ready Function Timing Chart**

The MB90242A device also provides an on-chip auto-ready function for external memory access. The auto ready function operates only for access to external memory allocated to addresses 000100H to FFFFFFFH, and extends the access cycle to 4 machine cycles by automatically inserting 1 to 3 cycles without the use of external circuits. This function is activated by setting the AR1 or AR0 bit in the EPCR register.

In addition, the MB90242A device provides an external I/O auto-ready function that is independent of the memory auto-ready function. When the IOR bit in the EPCR register is set to '0' a wait cycle of 2 machine cycles is automatically inserted for access to external addresses 0000C0H to 0000FFH, creating an access cycle of 4 machine cycles.

When either the external memory auto-ready function or the external I/O auto-ready function is used with the RYE bit in the EPCR register set to '1,' the wait cycle can be continually extended by input of the 'L' level signal at the P51/RDY pin after the end of the wait cycle introduced by the corresponding auto-ready function as described above.

### 3.4.6 Hold Function

When the HDE bit in the EPCR register is set to '1,' the use of the external address hold function is enabled for the P53/HRQ and P52/HAKX pins. When an 'H' level signal is input at the P53/HRQ pin, hold status will be applied at the next break in bus operations, an 'L' level signal will be output from the P52/HAKX pin, and the following pins will be placed in high impedance state.

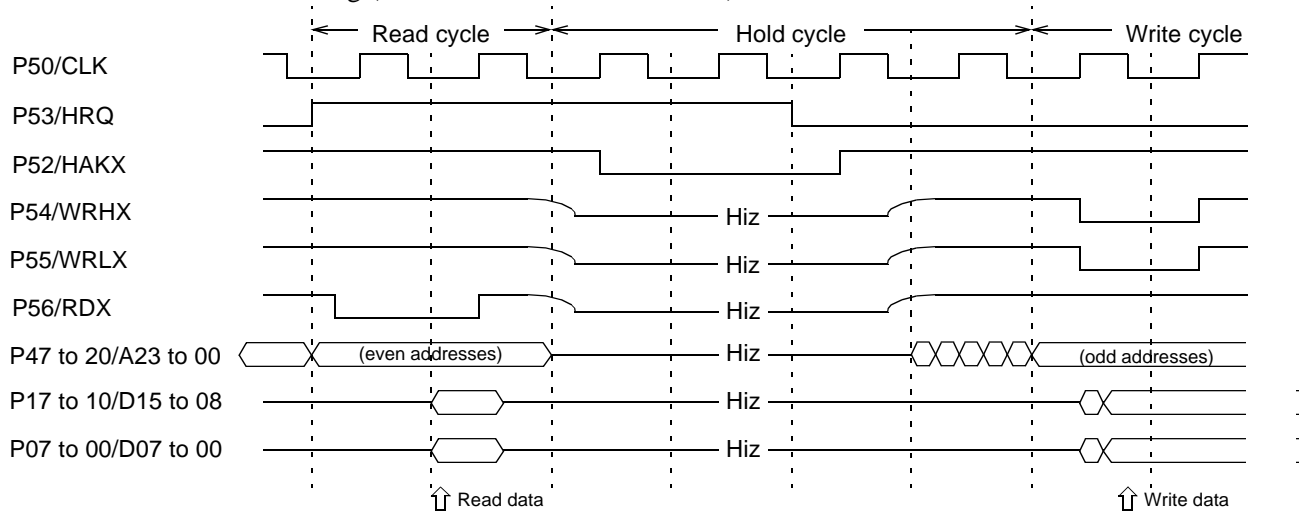
- Address output: P47/A23 to P20/A00
- Data I/O: P17/D15 to P00/D00
- Bus control signal: P56/RDX, P55/WRLX, P54/WRHX

This enables use of the external bus by external circuits.

When an 'L' level signal is input at the P53/HRQ pin, the P52/HAKX pin output becomes 'H' level, the pins revert to external signal pin status, and then bus operation is resumed.

In stop state, no hold request inputs are accepted.

■ Hold Function Timing (in 16-Bit External Bust Mode)



**Fig. 3.4.4 Hold Function Timing**

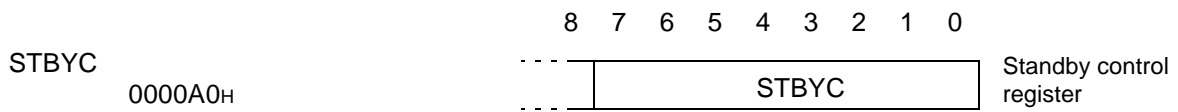
### 3.5 Power Saving Modes

The following power-saving modes are supported: sleep mode, stop mode, hardware standby mode, and gear functions.

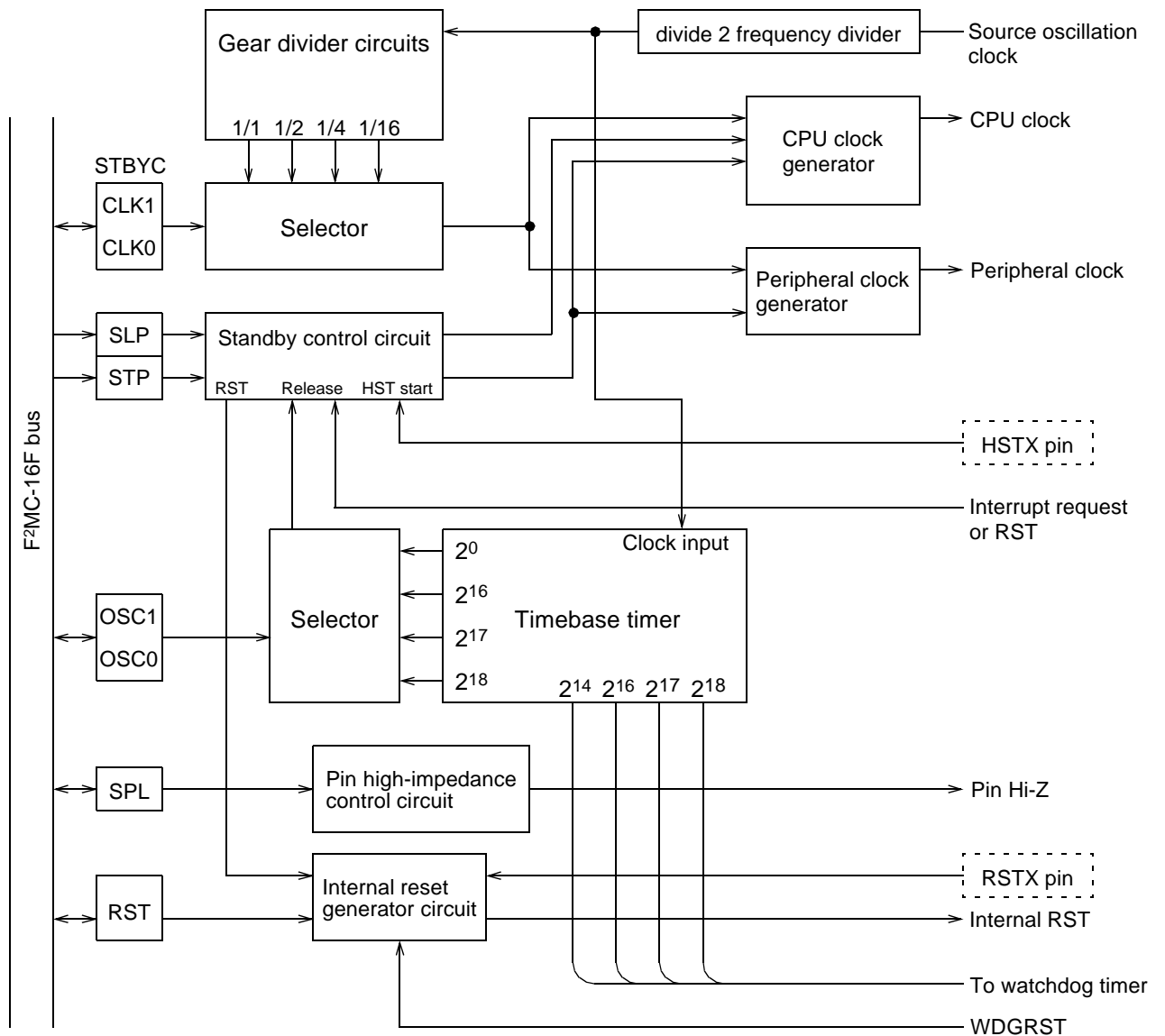
In sleep mode, only the CPU operating clock is stopped and all other functions remain operating. In comparison, oscillator functions stop in both stop mode and hardware standby mode, offering data retention at the lowest level of power consumption. Gear functions allow the external clock to be used as internal clock speeds divided by a choice of 2, 4, or 16, resulting in lower-power operation.

Sleep mode, stop mode and gear functions are controlled by settings in the standby control register (STBYC). Hardware standby mode is controlled by signal input to the HSTX pin.

#### 3.5.1 Register Configuration



### 3.5.2 Block Diagram



**Fig. 3.5.1 Power Saving Mode Control Circuit and Clock Generator**

### 3.5.3 Detailed Register Description

#### (1) STBYC (Standby control register)

	7	6	5	4	3	2	1	0	←Bit no.
Address : 0000A0H	STP	SLP	SPL	RST	OSC1	OSC0	CLK1	CLK0	STBYC
Read/write ⇒	(W)	(W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value⇒	(0)	(0)	(0)	(1)	(*)	(*)	(*)	(*)	

[Bit 7] STP

Write '1' to this bit for transition to stop mode. Writing the value '0' has no effect on operation. The value is set to '0' at a reset or wake-up from stop mode. This bit is write-only, and the read value is always '0.'

### 3.5 Power Saving Modes

#### [Bit 6] SLP

Write '1' to this bit for transition to sleep mode. Writing the value '0' has no effect on operation. This bit is set to '0' at reset, wake-up from sleep mode or stop mode.

When '1' is simultaneously written to both the STP bit and SLP bit, the MB90242A will go into stop mode. This bit is write-only, and the read value is always '0.'

#### [Bit 5] SPL

When this bit is '0,' the level of external pins will be retained in stop mode. When this bit is '1,' external pins will be placed in high-impedance state. The value is set to '0' at reset. This bit is read-write enabled.

#### [Bit 4] RST

When '0' is written to this bit an internal reset signal will be generated for 3 machine cycles. Writing '1' to this bit has no effect. The read value is '1.'

#### [Bits 3, 2] OSC1, OSC0

These bits set the oscillation stabilization wait period at wake-up from stop mode or hardware standby mode.

The initial value at power-on reset is '11,' but these bits are not initialized following resets due to other factors. These bits are read-write enabled.

**Table 3.5.1 OSC Bit Settings**

OSC1	OSC0	Oscillation stabilization wait period (at 32 MHz source oscillation)
0	0	No stabilization wait period
0	1	approx. 2.05 ms ( $2^{15}$ source oscillation count)
1	0	approx. 4.10 ms ( $2^{16}$ source oscillation count)
1	1	approx. 8.19 ms ( $2^{17}$ source oscillation count)

#### [Bits 1, 0] CLK1, CLK0

These bits select the machine clock operating frequency and gear function ratios. They are not initialized following resets from the RSTX pin or RST bit. The initial value is '11' following power-on reset, hardware standby or watchdog reset.

These bits are read-write enabled.

**Table 3.5.2 CLK Bit Settings**

CLK1	CLK0	Machine clock (at 32 MHz source oscillation)	Gear ratio
0	0	16 MHz	1/1
0	1	8 MHz	1/2
1	0	4 MHz	1/4
1	1	1 MHz	1/16

### 3.5.4 Operating Description

#### (1) Overview of Power Saving Modes

Table 3.5.3. lists the status of units in the MB90242A in each operating mode.

**Table 3.5.3 Operating Status in Power Saving Modes**

	Transition condition	Oscillator	Clock	CPU	Internal peripherals	Pins	Exit method
Sleep	SPL=1	Operating	Operating	Stopped	Operating	Operating	Reset Interrupt
Stop (SPL=0)	STP=1	Stopped	Stopped	Stopped	Stopped	Hold	Reset Interrupt
Stop (SPL=1)	STP=1	Stopped	Stopped	Stopped	Stopped	Hi-z	Reset Interrupt
Hardware standby	HSTX=L	Stopped	Stopped	Stopped	Stopped	Hi-z	HSTX=H
Gear function	Write to STBYC register CLK bit	Operating	Operate at divided frequency	Operate at divided frequency	Operate at divided frequency	Operating	Write to STBYC register CLK bit

#### (2) Sleep Mode

##### ■ Transition to Sleep Mode

Transition to sleep mode is initiated by writing '1' to the SLP bit, and '0' to the STP bit in the standby control register (STBYC).

Sleep mode stops only the clock signal supplied to the CPU, stopping the CPU while the internal peripheral resource circuits continue to operate. A number of program fetch operations may be made after writing to the STBYC register and before the CPU stops.

If an interrupt request is generated at the time that '1' is written to the SLP bit, transition to sleep mode will not occur. In this case if the CPU status does not accept the interrupt, it will execute the next command. If the CPU status does accept the interrupt, it will execute the next command following the STBYC write command (as long as it does not hold an interrupt), and then branch to the interrupt processing program after the completion of the next program.

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

### 3.5 Power Saving Modes

#### ■ Wake-up from Sleep Mode

Sleep mode can be exited by input of a reset signal or occurrence of an interrupt.

If a reset is used to release sleep mode, the MB90242A will apply a reset upon wake-up from sleep mode.

Sleep mode will also be released when an interrupt of level 7 or higher (priority level) is generated by an internal resource circuit. After release, the same processing is applied as for a normal interrupt. If the interrupt is accepted according to the set values of the I flag, ILM bit and interrupt control register (ICR), the CPU will execute the next command following the STBYC write instruction (as long as it does not hold an interrupt) and then will execute the interrupt processing after that instruction. If the interrupt is not accepted, execution will continue with the next instruction following the command that caused the transition to sleep mode.

#### (3) Stop Mode

#### ■ Transition to Stop Mode

Transition to stop mode is initiated by writing '1' to the STP bit in the standby control register (STBYC).

In stop mode, the source oscillation is stopped, stopping all MB90242A device functions. This is therefore the mode with the lowest power mode in which data is retained.

Also, the SPL bit in the STBYC register can be used to determine whether external pins are placed in high-impedance state or retain their values at the state immediately preceding the transition to stop mode.

If an interrupt request is generated at the time that '1' is written to the STP bit, transition to stop mode will not occur. In this case if the CPU status does not accept the interrupt, it will execute the next instruction. If the CPU status does accept the interrupt, it will execute the next command following the STBYC write instruction (as long as it does not hold an interrupt), and then branch to the interrupt processing program after the completion of the next instruction.

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

#### ■ Wake-up from Stop Mode

Stop mode can be exited by input of a reset signal or occurrence of an interrupt.

If a reset is used to exit stop mode, the MB90242A will apply a reset upon wake-up from stop mode.

After wake-up from stop mode, the standby control circuit will move first to oscillation stabilization wait mode before releasing the stop mode control. Also, a reset factor is used for wake-up from stop mode, the reset sequence applies an oscillation stabilization wait period before reset.

During stop mode, an interrupt of level 7 or higher (priority level) can be generated using an external interrupt signal to release the MB90242A from stop mode. The wake-up sequence first passes through an oscillation stabilization period defined by the OSC1, OSC0 bits, then follows the normal interrupt processing sequence according to the values of the I flag, ILM bit and interrupt control register (ICR), and the CPU will execute the next instruction following the STBYC write instruction (as long as it does not hold an interrupt) and then will execute the interrupt processing after the completion of the next program. If the interrupt is not accepted, execution will continue with the next instruction following the command that caused the transition to stop mode.

#### (4) Hardware Standby Mode

##### ■ Transition to Hardware Standby Mode

The MB90242A can be placed in hardware standby mode from any mode, by entering an 'L' level signal at the HSTX pin. In hardware standby mode, oscillator signals are stopped and all external pins are placed in high impedance state as long as the HSTX signal remains at 'L' level, regardless of any other modes including reset signals.

In hardware standby mode the contents of RAM are not retained, however a reset may be applied to initialize dedicated registers having initial value settings.

##### ■ Wake-up from Hardware Standby Mode

Hardware standby mode can only be exited by a signal from the HSTX pin. When the HSTX signal changes to 'H' level, hardware standby mode is exited, the internal reset signal is enabled and then the MB90242A moves into oscillation stabilization wait mode. After the oscillation stabilization period has passed, the internal reset is released and the CPU begins operation by executing its own reset sequence.

#### (5) Oscillation Stabilization Wait Period Settings

The OSC1, OSC0 bits are used to select the length of the oscillation stabilization wait period applied after wake-up from stop mode and hardware standby mode. The user should select the appropriate wait period according to the characteristics of oscillator circuits connected to the X0 and X1 pins, as well as the types of oscillator elements used.

These bits are not initialized by any reset signals other than the power-on reset. At a power-on reset, the value is initialized to '11.' As a result, the oscillation stabilization wait period at a power-on reset is approximately  $2^{17}$  counts of the source oscillation.

#### (6) Gear Functions

##### ■ Machine Clock Switching

Machine clock speeds can be switched by writing values to the CLK1, CLK0 bits in the STBYC register. The STBYC register signal resets the machine clock frequency divider and the desired machine clock speed begins with the next machine cycle.

Even when not switching machine clock frequency, unless 16 MHz is selected, the STBYC write cycle can be used to set the machine clock 4 periods longer than the source oscillation.

##### ■ Machine Clock Initialization

The CLK0, CLK1 bits are not initialized by resets from external pins or the RST bit. The initial value following a reset is '11.'



## 3.6 Pin Status in Sleep, Stop, Hold and Reset Modes

Tables 3.6.1 and 3.6.2 show pin status both in bus modes, when the MB90242A is in sleep, stop, hold and reset modes.

**Table 3.6.1 Pin Status in 16-Bit External Bus Mode**

Pin	Sleep mode	Stop mode		Hold mode	Reset
		SPL=0	SPL=1		
P07 to P00	Input disabled Output Hi-Z	Input disabled	Input shut off Output Hi-Z Note 3	Input disabled Output Hi-Z	Input disabled Output Hi-Z
P17 to P10					
P27 to P20 P37 to P30 P47 to P40	Output status Note 4	Output status		Output status	
P50/CLK	Input disabled Output enabled Notes 5 and 6	Input disabled Output status		Input disabled Output enabled Notes 5 and 6	Output enabled
P51/RDY	RDY input Note 6	Immediately prior status retained		RDY input Note 6	Input disabled Output Hi-Z
P52/HAKX	'H' output Note 6			'L' output	
P53/HRQ	HRQ input Note 6			'I' input	
P54/WRHX	'H' output Note 6	'H'		Hi-Z Note 6	
P55/WRLX					
P56/RDX	'H' output	'H'		Output mode 'H' Input mode Hi-Z	'H' output
P57	Immediately prior status retained Note 2	Immediately prior status retained Note 2	Immediately prior status retained Note 2	Input disabled Output Hi-Z	
P67 to P66 P63 to P60					
P75 to P70					
P82					
P81 to P80		Input enabled	Input enabled Note 1		

**Note1:** Same as other ports when used for output port functions. 'Input enabled' means that input functions are available for use. However, input is shut off when the corresponding interrupts are disabled.

**Note2:** In this mode, the immediately preceding output values are retained and output. If in input state, input is disabled. 'Input disabled' means that the first input gate at the pin is enabled for operation, but that internal circuits are not operating so that the signal from the pin is not accepted internally.

**Note3:** 'Input shut off' means that the first input gate at the pin is disabled. 'Output Hi-Z' means that the transistor driving the pin is disabled placing the pin in high-impedance state.

**Note4:** 'Output mode' means that the transistor driving the pin is in drive enabled state, but since the internal circuit operation is stopped, a fixed 'H' or 'L' value is output.

**Note5:** 'Output enabled' means that the transistor driving the pin is in drive enabled state, and because internal circuit operations are enabled, the process of operation can be sensed from the pin.

**Note6:** When used as a general-purpose port, the immediately prior status is retained.

**Table 3.6.2 Pin Status in 8-Bit External Bus Mode**

Pin	Sleep mode	Stop mode		Hold mode	Reset
		SPL=0	SPL=1		
P07 to P00	Input disabled Output Hi-Z	Input disabled	Input shut off Output Hi-Z	Input disabled Output Hi-Z	Input disabled Output Hi-Z
P17 to P10	Immediately prior status retained			Immediately prior status retained	
P27 to P20 P37 to P30 P47 to P40	Output status Note 4	Output status		Input disabled Output Hi-Z	Output status
P50/CLK	Input disabled Output enabled Notes 5 and 6	Output status Note 6		Input disabled Output enabled Notes 5 and 6	Output enabled
P51/RDY	RDY input Note 6	Immediately prior status retained Note 2		RDY input Note 6	Input disabled Output Hi-Z
P52/HAKX	'H' output Note 6			'L' output	
P53/HRQ	HRQ input Note 6			'1' input	
P54/WRHX	Immediately prior status retained			Immediately prior status retained	
P55/WRLX	'H' output Note 6	'H' Note 6		Hi-Z Note 6	
P56/RDX	'H' output	'H'		Output mode 'H' Input mode Hi-Z	'H' output
P57	Immediately prior status retained Note 2	Immediately prior status retained Note 2	Immediately prior status retained Note 2	Immediately prior status retained	Input disabled Output Hi-Z
P67 to P66 P63 to P60					
P75 to P70					
P82					
P81 to P80					

- Note1:** Same as other ports when used for output port functions. 'Input enabled' means that input functions are available for use. However, input is shut off when the corresponding interrupts are disabled.
- Note2:** In this mode, the immediately preceding output values are retained and output. If in input state, input is disabled. 'Input disabled' means that the first input gate at the pin is enabled for operation, but that internal circuits are not operating so that the signal from the pin is not accepted internally.
- Note3:** 'Input shut off' means that the first input gate at the pin is disabled. 'Output Hi-Z' means that the transistor driving the pin is disabled placing the pin in high-impedance state.
- Note4:** 'Output mode' means that the transistor driving the pin is in drive enabled state, but since the internal circuit operation is stopped, a fixed 'H' or 'L' value is output.
- Note5:** 'Output enabled' means that the transistor driving the pin is in drive enabled state, and because internal circuit operations are enabled, the process of operation can be sensed from the pin.
- Note6:** When used as a general-purpose port, the immediately prior status is retained.



# APPENDIX

---

This part of the manual contains F<sup>2</sup>MC-16F addressing specifications, instruction lists and instruction maps.

## Appendix A. F<sup>2</sup>MC-16F Addressing Specifications

- A.1 Effective Address Fields
- A.2 Detailed Addressing Format Specifications

## Appendix B. F<sup>2</sup>MC-16F Instruction Lists

- B.1 Instruction List Heading Descriptions
- B.2 Instruction List Symbols
- B.3 Effective Address Fields
- B.4 Calculation of Execution Cycle Counts
- B.5 Transfer Instructions
- B.6 Numerical Calculation Instructions
- B.7 Logical Calculation Instructions
- B.8 Shift Instructions
- B.9 Branching Instructions
- B.10 Other Instructions
- B.11 Execution Cycle Counts for Special Operations

## Appendix C. F<sup>2</sup>MC-16F Instruction Map

- C.1 Basic Map Structure
- C.2 Basic Page Map
- C.3 Bit Operation Instruction Map
- C.4 MOV<sub>M</sub> Instruction Map
- C.5 Character String Operation Map
- C.6 2-Byte Instruction Map
- C.7 ea Instructions
- C.8 MOVEA Rwi, ea
- C.9 MOV Ri, ea
- C.10 MOVW Rwi, ea
- C.11 MOV ea, Ri
- C.12 MOVW ea, Rwi
- C.13 XCH Ri, ea
- C.14 XCHW Rwi, ea

# APPENDIX A: F<sup>2</sup>MC-16F Addressing Specifications

---

This section of the manual describes addressing specifications for the F<sup>2</sup>MC-16F.

- A.1 Effective Address Fields
- A.2 Detailed Addressing Format Specifications

## A.1 Effective Address Fields

Appendix A.1 shows the address format for designating effective address fields.

**Table A.7 Effective Address Fields**

Code	Notation			Address format	Default Bank
00	R0	RW0	RL0	Register direct ea corresponds to byte, word, long-word formats in order from left.	None
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	DTB
09	@RW1				DTB
0A	@RW2				ADB
0B	@RW3				SPB
0C	@RW0+			Register indirect with post increments	DTB
0D	@RW1+				DTB
0E	@RW2+				ADB
0F	@RW3+				SPB
10	@RW0+disp8			Register indirect with 8-bit displacement	DTB
11	@RW1+disp8				DTB
12	@RW2+disp8				ADB
13	@RW3+disp8				SPB
14	@RW4+disp8			Register indirect with 8-bit displacement	DTB
15	@RW5+disp8				DTB
16	@RW6+disp8				ADB
17	@RW7+disp8				SPB
18	@RW0+disp16			Register indirect with 16-bit displacement	DTB
19	@RW1+disp16				DTB
1A	@RW2+disp16				ADB
1B	@RW3+disp16				SPB
1C	@RW0+RW7			Register indirect with index	DTB
1D	@RW1+RW7			Register indirect with index	DTB
1E	@PC+disp16			PC indirect with 16-bit displacement	PCB
1F	addr16			Direct address	DTB

## A.2 Detailed Addressing Format Specifications

The F<sup>2</sup>MC-16F uses 25 types of addressing formats.

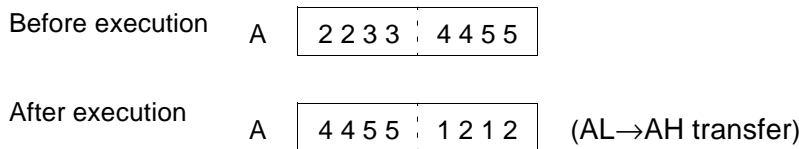
### ■ Immediate addressing (#imm)

Operand values are specified directly.

- #imm4
- #imm8
- #imm16
- #imm32

### ○ Operating Example

MOVW A,#01212H (Instruction: store operand value in A)



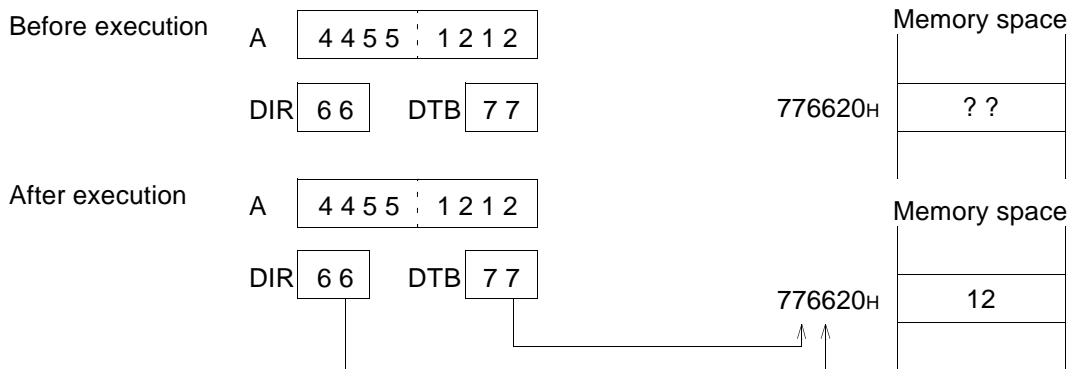
**Fig. A.2a Example of Immediate Addressing**

### ■ Condensed Direct Addressing (dir)

Operands specify the lower 8 bits of the memory address. Addresses bits 8 to 15 are determined by the DPR register. Addresses bits 16 to 23 are set by the DTB register.

### ○ Operating Example

MOV dir 20H, A (Instruction: write lower 8 bits of A in condensed direct addressing)



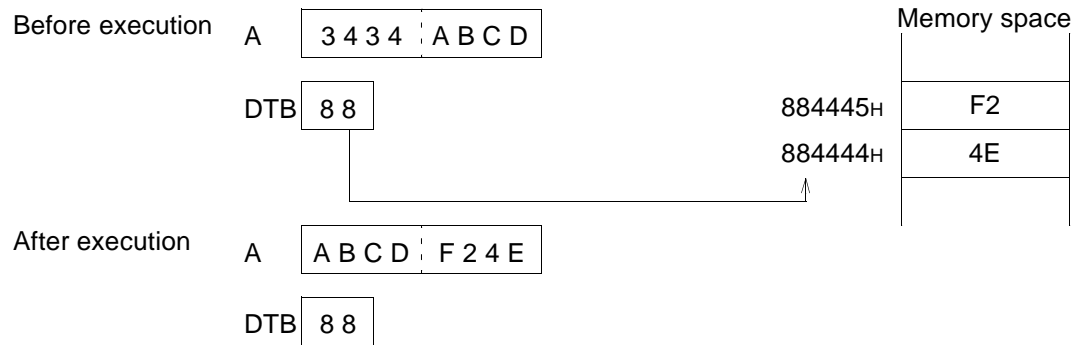
**Fig. A.2b Example of Condensed Direct Addressing**

■ Direct Addressing (addr16)

Operand values specify lower 16 bits of the memory address directly. Addresses bits 16 to 23 are determined by the DTB register.

○ Operating Example

MOVW A, 4444H (Instruction: read 16 bits in direct addressing, and store in A)



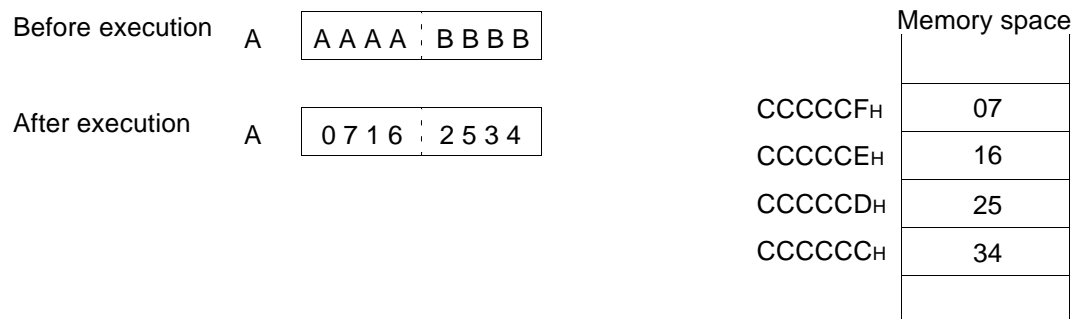
**Fig. A.2c Example of Direct Addressing (addr16)**

■ Direct Addressing (addr24)

Operand values specify a 24-bit memory address directly.

○ Operating Example

MOVPL A, CCCCCCH (Instruction: read 24 bits in direct addressing, and store in A.)



**Fig. A.2d Example of Direct Addressing (addr24)**



## A.2 Detailed Addressing Format Specifications

### ■ Register Direct Addressing

Operand values specify registers directly.

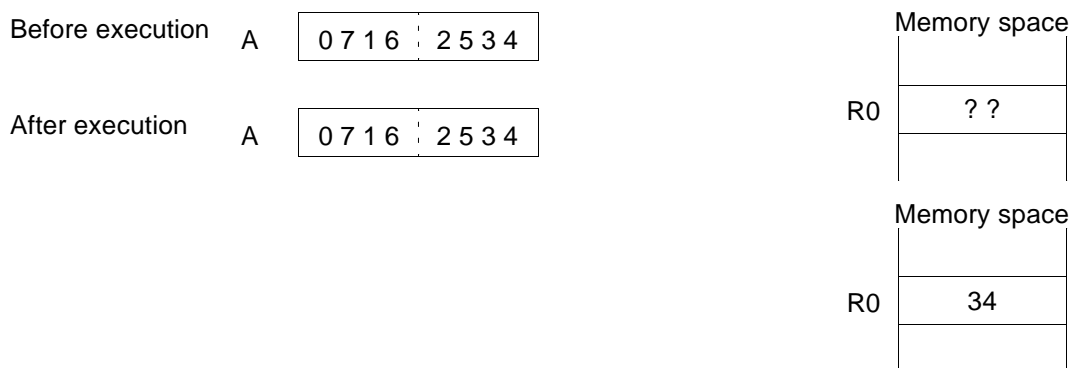
General-purpose registers:	Byte:	R0, R1, R2, R2, R4, R5, R6, R7
	Word:	RW1, RW2, RW3, RW4, RW5, RW6, RW7
	Long-word:	RL0, RL1, RL2, RL3
Dedicated registers:	Accumulators:	A, AL
	Pointer:	SP
	Bank:	PCB, DTB, USB, SSB, ADB
	Page:	DPR
	Control:	PS, CCR, RP, ILM

\* The SP register functions as a USP when the S-bit in the CCR register is '0', and as an SSP when the S-bit is '1.'

The SPB functions as a USB when the S-bit in the CCR register is '0' and as an SSB when the S-bit is '1.'

### ○ Operating Example

MOV R0,A (Instruction to transfer lower 8-bits of A to general-purpose register R.)



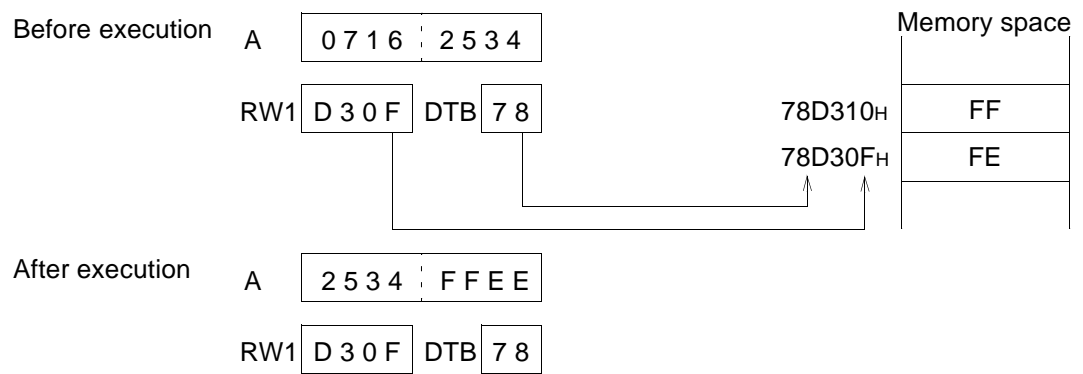
**Fig. A.2e Example of Register Direct Addressing**

■ Register Indirect Addressing (@RWj j=0 to 3)

This type of addressing accesses memory containing the contents of the general-purpose register RWj as address data. Addresses bits 16 to 23 are designated by the DTB register if using RW0 or RW1, by the SPB register if using RW3, and by the ADB register if using RW2.

○ Operating Example

MOVW A,@RW1 (Instruction to read the contents of the register in indirect addressing, and store in A.)



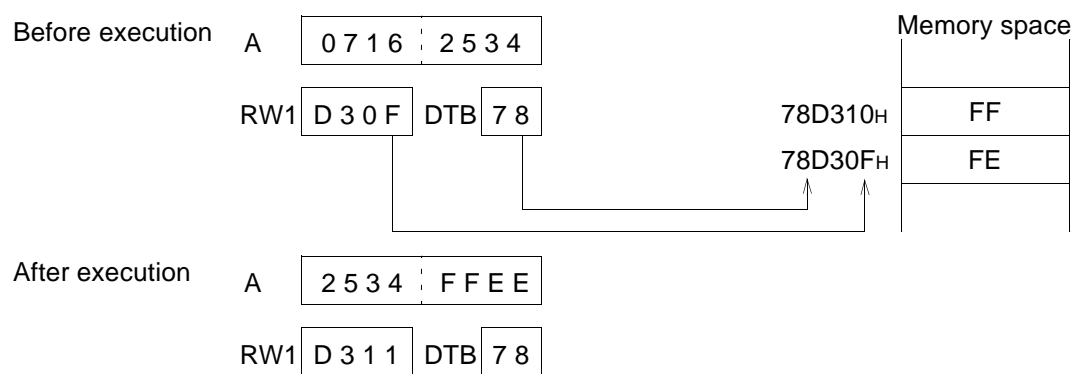
**Fig. A.2f Example of Register Indirect Addressing**

■ Register Indirect Addressing with Post Increments (@RWj+j=0 to 3)

This type of addressing accesses memory containing the contents of the general-purpose register RWj as address data. After manipulating the operand, register RWj is incremented by the operand data length (1 for byte length, 2 for word length, 4 for long-word length). Addresses bits 16 to 23 are designated by the DTB register if using RW0 or RW1, by the SPB register if using RW3, and by the ADB register if using RW2. Note that if the results of the post increment operation are the address of the same register that was set to incremented addressing, the value to be referenced after increment operation will be the incremented value. Also, if a write instruction is used, the write operation will have priority, so that the register that is to receive the increment will become write data.

○ Operating Example

MOVW A,@RW1+ (Instruction to read the contents of the register in indirect addressing for storage in A.)



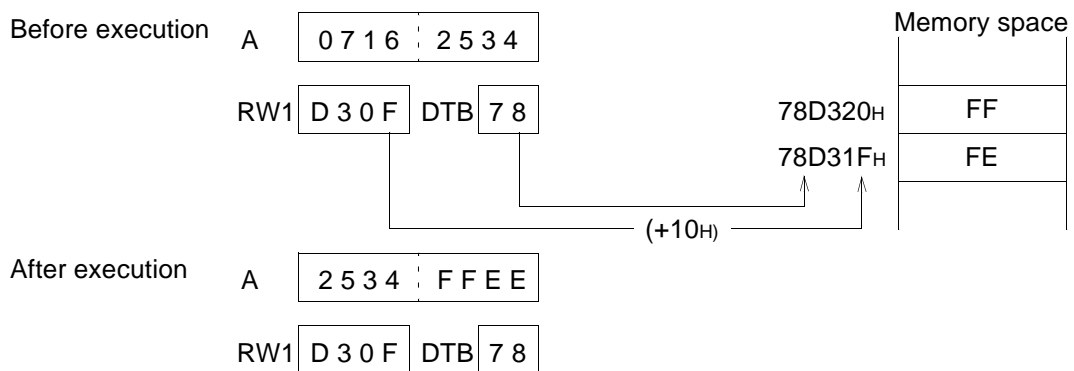
**Fig. A.2g Example of Register Indirect Addressing with Post Increments**

■ Register Indirect Addressing with Displacement ( $@RW_i+disp_8$   $i=0$  to  $7$ ,  $@RW_j+disp_{16}$   $j=0$  to  $3$ )

This type of addressing accesses memory containing addresses derived from the contents of the general-purpose memory  $RW_j$  with a displacement added. Displacements may be either byte or word length, and are added as signed numerical values. Addresses bits 16 to 23 are designated by the DTB register if  $RW_0$ ,  $RW_1$ ,  $RW_4$  or  $RW_5$  is used, by the SPB register if  $RW_3$  or  $RW_7$  is used, and by the ADB if  $RW_2$  or  $RW_6$  is used.

○ Operating Example

MOVW A,@RW1+10H (Instruction to read the contents of the register in indirect addressing with displacement added for storage in A.)



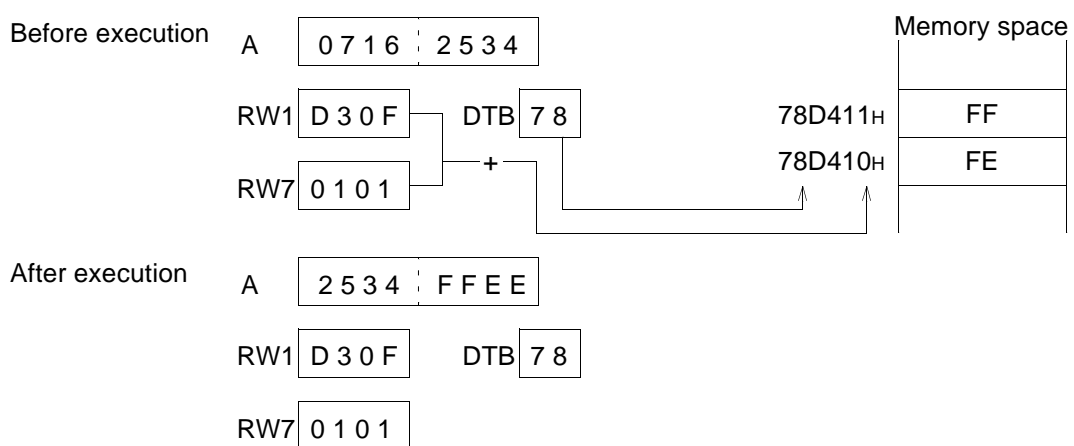
**Fig. A.2h Example of Register Indirect Addressing with Displacement**

■ Register Indirect with Base Index ( $@RW_0+RW_7$ ,  $@RW_1+RW_7$ )

This type of addressing accesses memory containing addresses in which the contents of general-purpose register  $RW_7$  are added with either  $RW_0$  or  $RW_1$ . Addresses bits 16 to 23 are designated by the DTB register.

○ Operating Example

MOVW A,@RW1+RW7 (Instruction to read register in indirect addressing with base index for storage in A.)



**Fig. A.2i Example of Register Indirect Addressing with Base Index**

■ Program Counter Indirect Addressing with Displacement (@PC+disp16)

This type of addressing accesses memory containing addresses calculated with the formula 'instruction address + 4 + disp16'. Displacement is in word length units. Addresses bits 16 to 23 are designated by the PCB register.

The operand address is normally considered to be 'next instruction address + disp16', but note that there are differences when using the following instructions.

- DBNZ eam,rel
- DWBNZ eam,rel
- MOV eam,#imm8
- MOVW eam,#imm16
- CBNE eam,#imm8,rel
- CWBNE eam,#imm16,rel
- MOVM @A,eam#imm8
- MOVWM @A,eam,#imm
- MOVM addr16,eam,#imm8
- MOVMW addr16,eam,#imm8
- MOVM eam,@A,#imm8
- MOVWM eam,@A,#imm8
- MOVM eam,addr16,#imm8
- MOVWM eam,addr16,#imm8

○ Operating Example

MOVW A,@PC+20H (Instruction to read PC in indirect addressing with displacement for storage in A.)

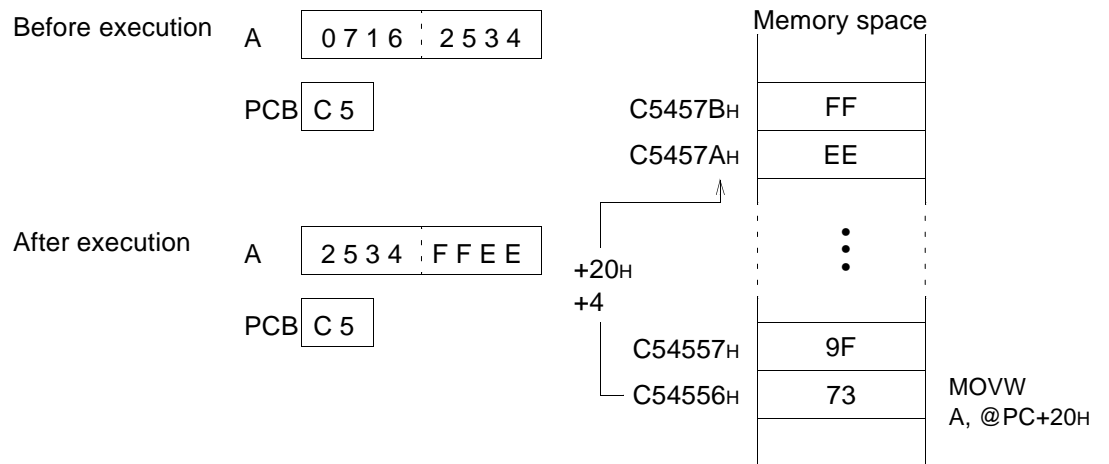


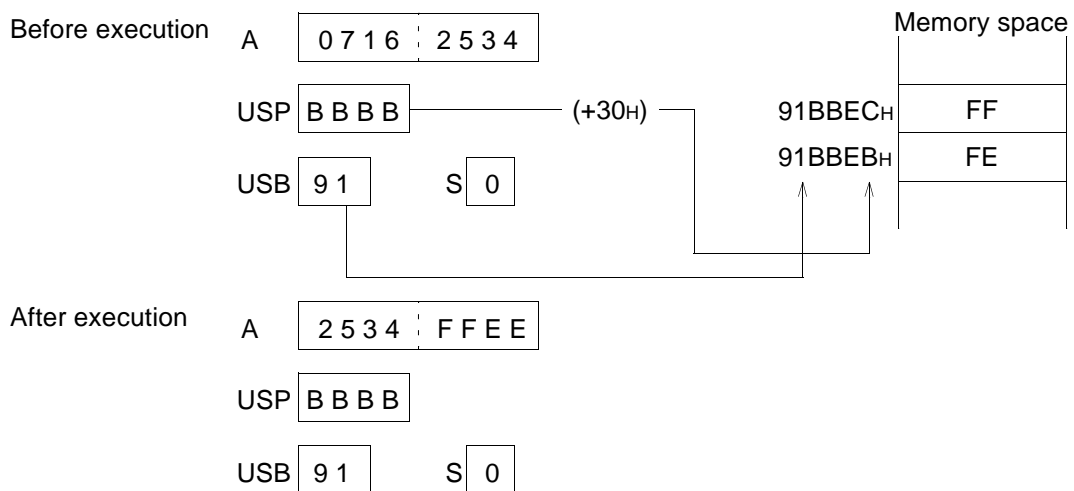
Fig. A.2j Example of Program Counter Indirect Addressing with Displacement

■ Stack Pointer Indirect Addressing with Displacement (@SP+disp8)

This type of addressing accesses memory using addresses calculated with the formula 'stack pointer value + disp8'. Displacement is in byte length units. Addresses bits 16 to 23 are designated by the SPB register.

○ Operating Example

MOVW A,@SP+30H (Instruction to read SP in indirect addressing with displacement for storage in A.)



**Fig. A.2k Example of Stack Pointer Indirect Addressing with Displacement**

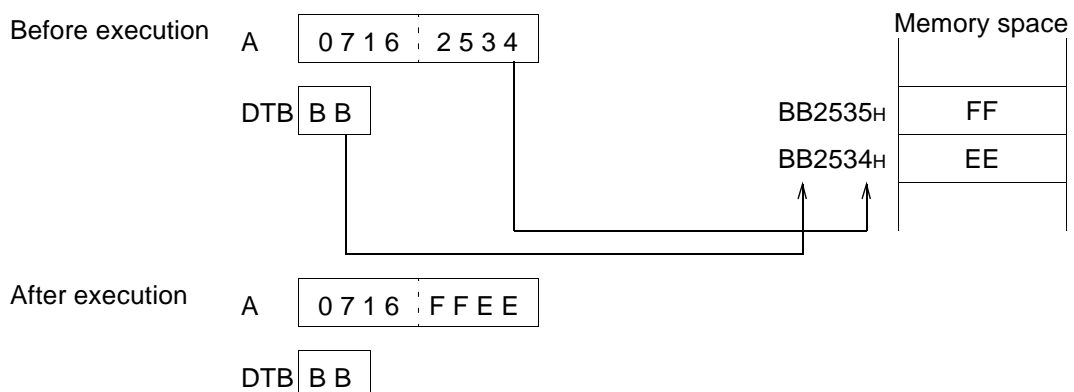
■ Accumulator Indirect Addressing (@A)

There are two types of accumulator indirect addressing; one in which addresses bits 00-15 are designated by the contents of the AL register and address bits 16-23 by the DTB register, and one in which addresses bits 00-23 are designated by the lower 24 bits of the A register.

Mnemonic designations are used.

○ Operating Example

MOVW A,@A (Instruction to read accumulator in indirect addressing for storage in A.)



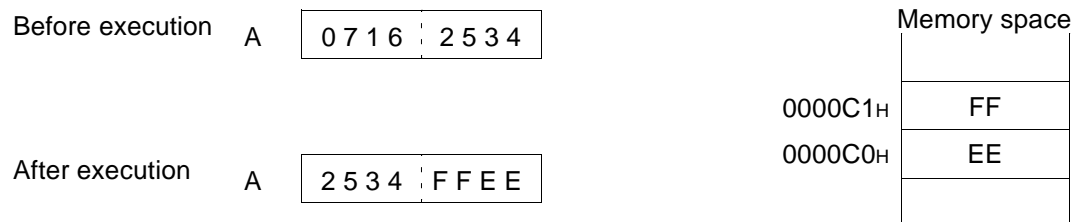
**Fig. A.2l Example of Accumulator Indirect Addressing**

■ I/O Indirect Addressing (io)

This method specifies memory addresses of the operand directly using an 8-bit displacement. Regardless of the values of the DTB and DPR registers, access is to I/O space at physical addresses 000000H to 0000FFH. With this type of addressing, prefix instructions designating access space are invalid.

○ Operating Example

MOVW A,io C0H (Instruction to read I/O directly for storage in A.)



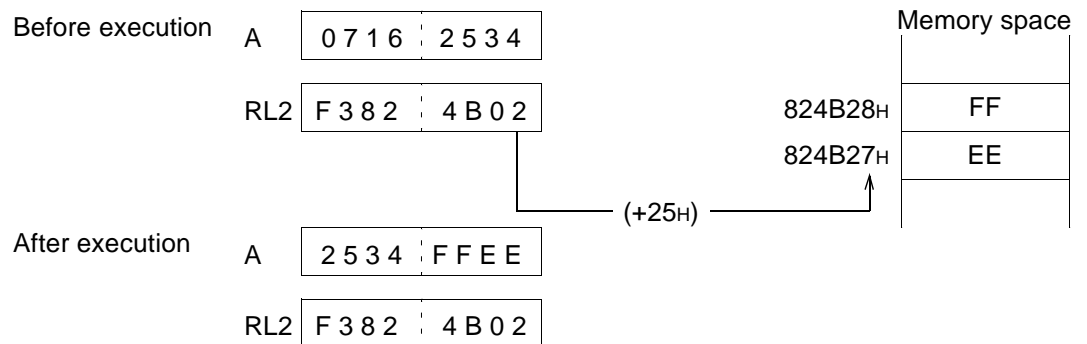
**Fig. A.2m Example of I/O Direct Addressing**

■ Long-Word Register Indirect Addressing with Displacement (@RLi+disp8 i=0 to 3)

This type of addressing accesses memory using addresses derived by using the lower 24 bits of the sum of the general-purpose register RLi plus a displacement. The displacement is 8 bits, and is added as signed numerical values to the RLi contents.

○ Operating Example

MOVW A,@RL2+25H (Instruction to read long-word register in indirect addressing with displacement for storage in A.)



**Fig. A.2n Example of Long-Word Register Indirect Addressing with Displacement**

■ Condensed Direct Bit Addressing (dir:bp)

This method uses the operand to designate the lower 8 bits of the memory address. Addresses bits 8 to 15 are designated by the DPR register, and addresses bits 16 to 23 are designated by the DTB register. The bit location is expressed as bp, with the higher values representing MSB, and the lower values LSB.

○ Operating Example

SETB dir 10H:0 (Instruction to set bit using condensed direct addressing.)



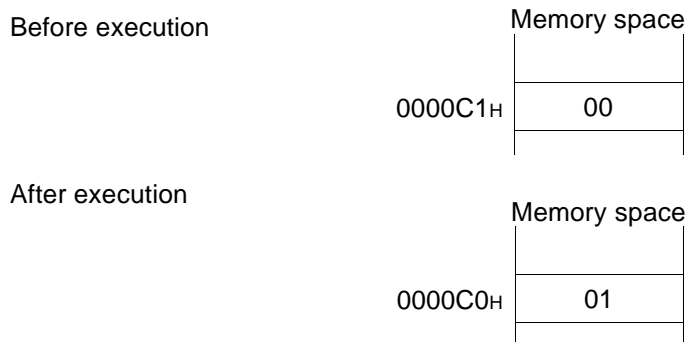
**Fig. A.2o Example of Condensed Direct Bit Addressing**

■ I/O Direct Bit Addressing (io:bp)

This method directly designates bits within the physical address range 000000H to 0000FFH. The bit location is expressed as bp, with the higher values representing MSB, and the lower values LSB.

○ Operating Example

SETB io C1H:0 (Instruction to set bit using I/O direct addressing.)



**Fig. A.2p Example of I/O Direct Bit Addressing**

■ Direct Bit Addressing (addr16:bp)

This method directly designates any bits within the 64-Kbyte area. Addresses bits 16 to 23 are designated by the DTB register. The bit location is expressed as bp, with the higher values representing MSB, and the lower values LSB.

○ Operating Example

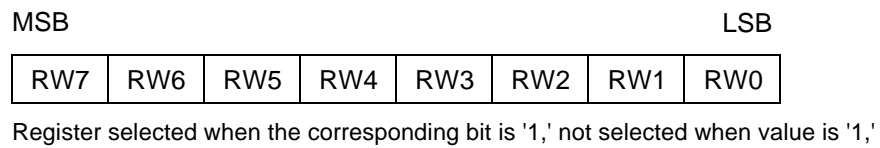
SETB 2222H:0 (Instruction to set a bit using direct addressing.)



**Fig. A.2q Example of Direct Bit Addressing**

■ Register List (rlst)

This method designates the register that is the object of push/pop instructions for a stack.



**Fig. A.2r Register List Configuration**

For operating example, see the 'F2MC-16F MB90200 Series Programming Manual.'

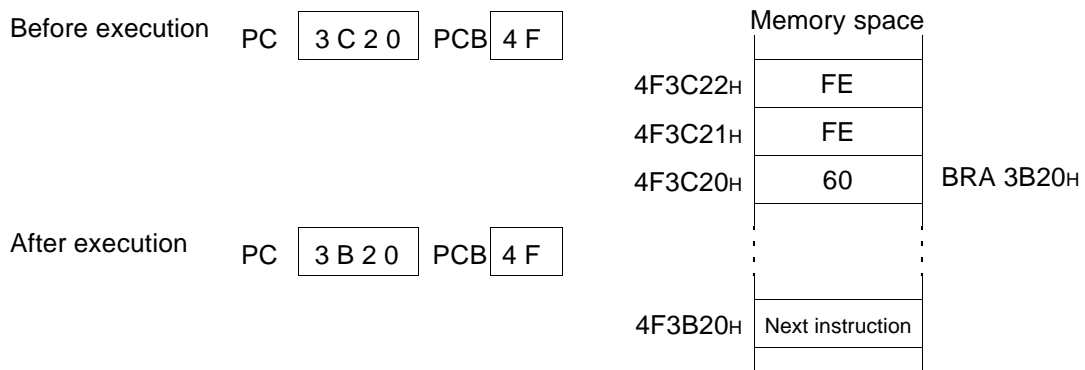


■ Program Counter Relative Branch Addressing

Branch destination addresses are represented as PC values plus 8-bit displacement values. If the result is over 16 bits, no increment/decrement is applied to the bank register and the overflow portion is ignored, so that the result is an address within the 64-Kbyte bank. This method is used for unconditional and conditional branching. Addresses bits 16 to 23 are designated by the PCB register.

○ Operating Example

BRA 3B20H (Instruction to execute unconditional relative branching.)



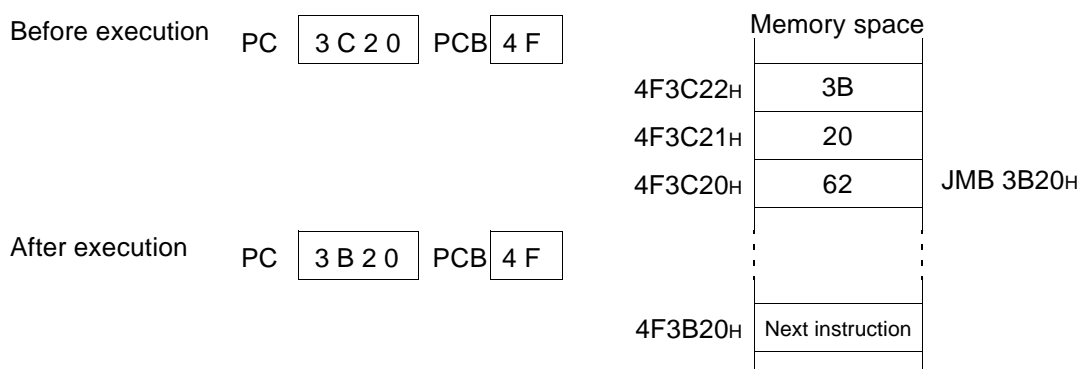
**Fig. A.2s Example of Program Counter Relative Branch Addressing**

■ Direct Branch Addressing (addr16)

Branch destination addresses are designated directly by displacement. The displacement is 16-bit data length and the displacement is used to designate a branch destination within logical space. This method is used for unconditional branching and subroutine call instructions. Addresses bits 16 to 23 are designated by the PCB register.

○ Operating Example

JMP 3B20H (Instruction to perform unconditional branching, with direct address designation within bank.)



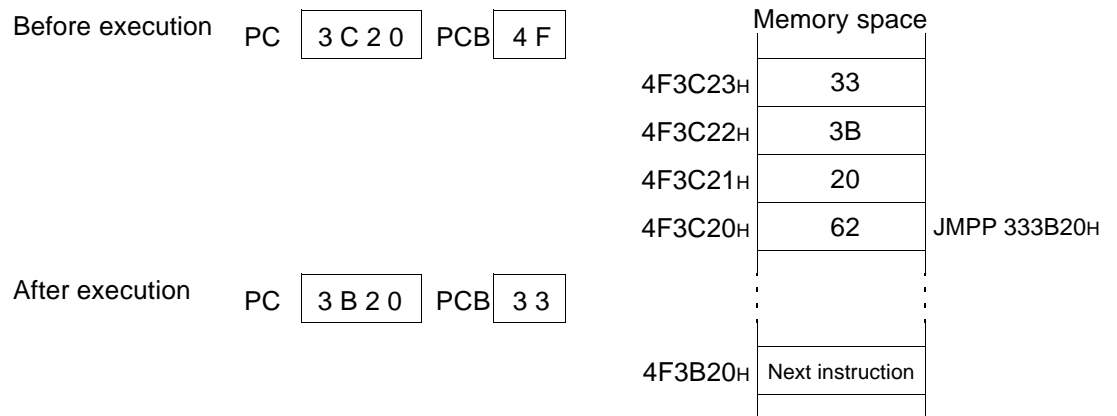
**Fig. A.2t Example of Direct Branch Addressing with 16-Bit Branch Designation**

■ Direct Branch Addressing (addr24)

Branch destination addresses are designated directly by displacement. The displacement data length is 24 bits, and the displacement is used to designate a physical address at the branch destination. This method is used for unconditional branching, subroutine call and software interrupt instructions.

○ Operating Example

JMPP 333B20H (Instruction to perform unconditional branching, with 24-bit direct address designation.)



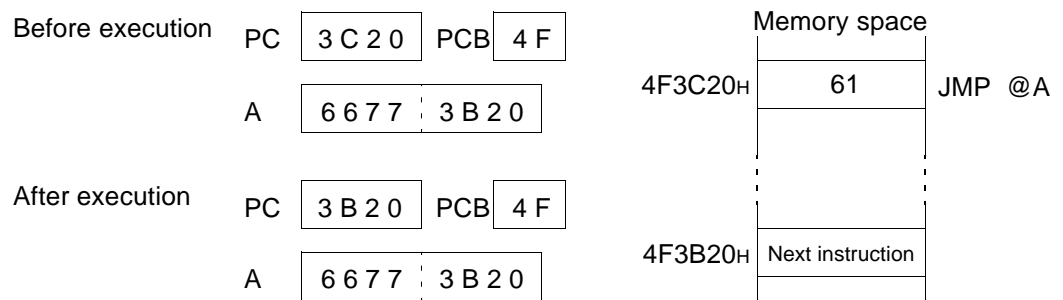
**Fig. A.2u Example of Direct Branch Addressing with 24-Bit Branch Designation**

■ Accumulator Indirect Branch Addressing (@A)

The 16-bit contents of the accumulator AL function as a branch destination address. This data designates the branch destination within bank space, and addresses bits 16 to 23 are designated by the PCB register. However if JCTX is used, the value is determined by the DTB register. This method is used with unconditional branching instructions.

○ Operating Example

JMP @A (Instruction to perform unconditional branching with accumulator indirect addressing.)



**Fig. A.2v Example of Accumulator Indirect Branch Addressing**

## A.2 Detailed Addressing Format Specifications

### ■ Vector Addressing (#vct)

In this method the branch destination address is the contents of the specified vector. Vector number data length may be either 4 bits or 8 bits. This method is used for subroutine call instructions, and software interrupt instructions.

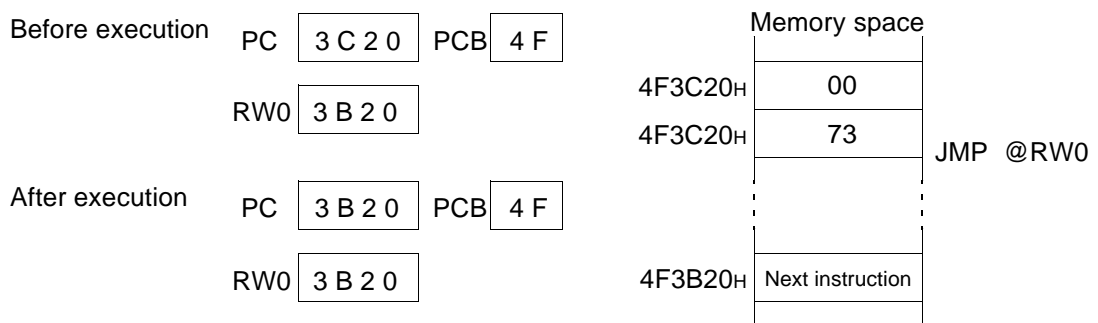
See the "F<sup>2</sup>MC-16F MB90200 Series Programming Manual" for information on of the CALLV #vct4 instruction and INT #vct8 instruction.

### ■ Indirect Designation Branch Addressing (@ear)

In this method the branch destination address is the word data at the address designated by the ear parameter.

#### ○ Operating Example

**JMP @RW0** (Instruction to perform unconditional branching with register indirect addressing.)



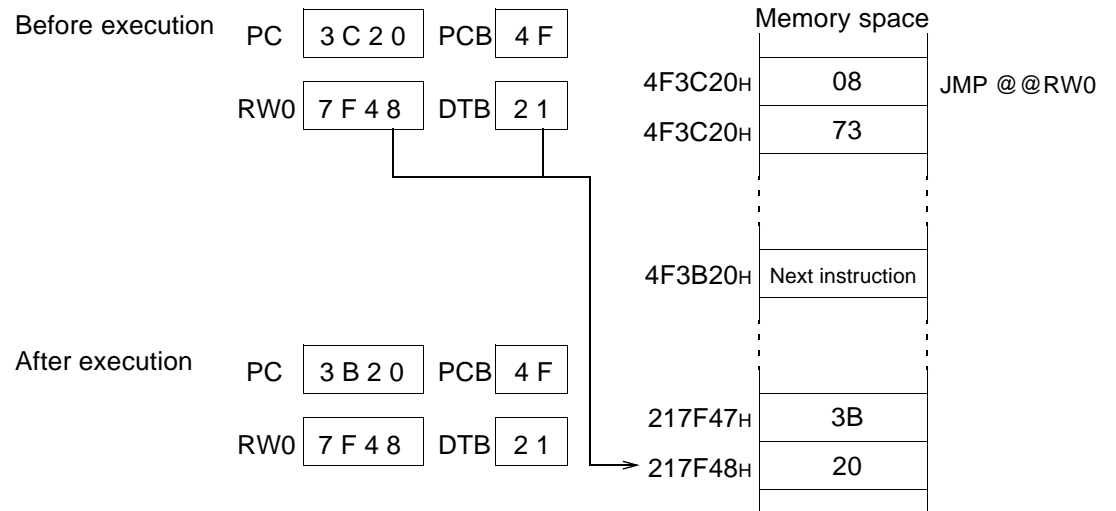
**Fig. A.2w Example of Register Indirect Branch Addressing**

■ Indirect Designation Branch Addressing (@eam)

In this method the branch destination address is the word data at the address designated by the eam parameter.

○ Operating Example

**JMP @@RW0** (Instructions to perform unconditional branching with the register indirect addressing.)



**Fig. A.2x Example of Register Indirect Branch Addressing**

## **APPENDIX B: F<sup>2</sup>MC-16F Instruction Lists**

---

This appendix explains a set of instructions used for an assembler.

- B.1 Instruction List Heading Descriptions
- B.2 Instruction List Symbols
- B.3 Effective Address Fields
- B.4 Calculation of Execution Cycle Counts
- B.5 Transfer Instructions
- B.6 Numerical Calculation Instructions
- B.7 Logical Calculation Instructions
- B.8 Shift Instructions
- B.9 Branching Instructions
- B.10 Other Instructions
- B.11 Execution Cycle Counts for Special Operations

## B.1 Instruction List Heading Descriptions

Appendix B.1 presents descriptions of instruction list headings.

**Table B.1 Instruction List Heading Descriptions**

Heading	Description
Mnemonic	Alphabetic capitals, symbols: shown as they appear in source program. Alphabetic lower case: rewritten in source program notation. Numerals following alphabetic lower case: indicates bit length in instructions.
#	Indicates byte count.
~	Indicates cycle count. For alphabetic characters in headings, see table B.4a.
B	Indicates complement value for calculation of actual cycle count for instruction execution. The actual cycle count is the sum of the values in the '~' column and the B column.
Operation	Describes how the instruction operates.
LH	Indicates special operations with respect to accumulator bits 15 to 08. Z: Transfer zero X: Transfer signed extension -: no transfer
AH	Indicates special operations with respect to the upper 16 bits of the accumulator. *: Transfer from AL to AH -: No transfer Z: Transfer 00 <sub>H</sub> to AH X: According to AL signed extension, transfer 00 <sub>H</sub> or FF <sub>H</sub> to AH.
I	Indicates status of flags: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry) *: Changed as a result of instruction execution -: No change S: Becomes '1' as a result of instruction execution R: Becomes '0' as a result of instruction execution
S	
T	
N	
Z	
V	
C	
RMW	Indicates a read-modify-write instruction (one instruction used to read data from memory, etc. and write the results back to memory). *: Read-modify-write instruction -: Not a read-modify-write instruction <b>Note:</b> This type of instruction cannot be used with addresses having different read/write meanings.

## B.1 Instruction List Heading Descriptions

- Precautionary Information for Calculation of Execution Cycles

The number of cycles (execution cycles) required to execute an instruction is determined by the 'cycle count' value of the instruction, added to the 'compensation value' determined by conditions. However, the actual cycle count for an instruction includes the above cycle count plus a number of cycles required for program reading.

Normally, program reading is performed automatically during empty cycles on the bus when there is an opening in the instruction queue. However, in cases where reading cannot be completed during empty bus cycles, instruction execution will be interrupted and the actual cycle number will be increased to allow program reading.

For calculation of actual cycles, see Appendix B.4 "Calculation of Execution Cycle Counts."

## B.2 Instruction List Symbols

Appendix B.2 lists symbols used in instruction lists and their meanings.

**Table B.2 Instruction List Symbols**

Symbol	Meaning
A	32-bit accumulator Bit length varies according to instruction. Byte: lower 8 bits of AL Word: 16 bits of AL Long word: 32 bits of AL or AH
AH AL	Upper 16 bits of A Lower 16 bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
SPCU SPCL	Stack pointer upper limit register Stack pointer lower limit register
PCB	Program bank register
DTB	Data bank register
ADB	Additional bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir addr16 addr24 ad24 0-15 ad24 16-23	Condensed direct addressing Direct addressing Physical direct addressing ad24 bit0-15 ad24 bit16-23
io	I/O area (000000H to 0000FFH)



**Table B.2 Instruction List Symbols (Continued)**

Symbol	Meaning
#imm4	4-bit immediate data
#imm8	8-bit immediate data
#imm16	16-bit immediate data
#imm32	32-bit immediate data
ext(imm8)	8-bit immediate data with 16-bit data sign extension
disp8	8-bit displacement
disp16	16-bit displacement
bp	Bit offset value
vct4	Vector number (0 to 15)
vct8	Vector number (0 to 255)
( )b	Bit address
rel	PC relative branch addressing
ear	Effective address designation (code 00 to 07)
eam	Effective address designation (code 08 to 1F)
rlst	Register list

## B.3 Effective Address Fields

Table B.3 lists the address formats for designating effective address fields.

**Table B.3 Effective Address Fields**

Code	Notation	Address format	Address extender byte count (Note)
00 01 02 03 04 05 06 07	R0 RW0 RL0 R1 RW1 (RL0) R2 RW2 RL1 R3 RW3 (RL1) R4 RW4 RL2 R5 RW5 (RL2) R6 RW6 RL3 R7 RW7 (RL3)	Register direct ea corresponds to byte, word, long-word formats in order from left.	-
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3	Register indirect	0
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+	Register indirect with post increments	0
10 11 12 13 14 15 16 17	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8 @RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8	Register indirect with 8-bit displacement	1
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16	Register indirect with 16-bit displacement	2
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	0 0 2 2

**Note:** The number of bytes in an address extension corresponds to the '+' indication in the # 'byte count' column in the instruction list.

## B.4 Calculation of Execution Cycle Counts

Appendix B.4a lists cycle counts for each method of addressing, and Appendix B.4b lists compensation values for calculating actual cycle counts.

In the instruction lists, some instructions require calculations that refer to these tables.

**Table B.4a Execution Cycle Counts for Methods Used to Designate Effective Addresses**

Code	Operand	(a)
		Execution cycle count for addressing method
00   07	Ri\RWi\RLi	Given on Instruction List
08   0B	@Wj	1
0C   0F	@RWj+	4
10   17	@RWi+disp8	1
18   1B	@RWj+disp16	1
1C	@RW0+RW7	2
1D	@RW1+RW7	2
1E	@PC+disp16	2
1F	addr16	1

**Note:** (a) is used on the - 'cycle count' column and the B (compensation value) column on the Instruction List.

**Table B.4b Compensation Values for Calculation of Execution Cycle Count**

Operand	(b)	(c)	(d)
	byte	word	long
Internal register	+0	+0	+0
Internal RAM, even address	+0	+0	+0
Internal RAM, odd address	+0	+1	+2
Even address, other than internal RAM	+1	+1	+2
Odd address, other than internal RAM	+1	+3	+6
External data bus 8 bits	+1	+3	+6

**Note:** (b), (c) and (d) are used on the ~ 'cycle count' column and the B (compensation value) column on the Instruction List.

## B.5 Transfer Instructions

**Table B.5a Transfer Instructions (Byte): 50 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV A,dir	2	2	(b)	byte (A) ← (dir)	Z	*	-	-	-	*	*	-	-	-
MOV A,addr16	3	2	(b)	byte (A) ← (addr16)	Z	*	-	-	-	*	*	-	-	-
MOV A,Ri	1	1	0	byte (A) ← (Ri)	Z	*	-	-	-	*	*	-	-	-
MOV A,ear	2	1	0	byte (A) ← (ear)	Z	*	-	-	-	*	*	-	-	-
MOV A,eam	2+	2+(a)	(b)	byte (A) ← (eam)	Z	*	-	-	-	*	*	-	-	-
MOV A,io	2	2	(b)	byte (A) ← (io)	Z	*	-	-	-	*	*	-	-	-
MOV A,#imm8	2	2	0	byte (A) ← (imm8)	Z	*	-	-	-	*	*	-	-	-
MOV A,@A	2	2	(b)	byte (A) ← ((A))	Z	-	-	-	-	*	*	-	-	-
MOV A,@RLi+disp8	3	6	(b)	byte (A) ← ((RLi)+disp8)	Z	*	-	-	-	*	*	-	-	-
MOV A,@SP+disp8	3	3	(b)	byte (A) ← ((SP)+disp8)	Z	*	-	-	-	*	*	-	-	-
MOVP A,addr24	5	3	(b)	byte (A) ← (addr24)	Z	*	-	-	-	*	*	-	-	-
MOVP A,@A	2	2	(b)	byte (A) ← ((A))	Z	-	-	-	-	*	*	-	-	-
MOVN A,#imm4	1	1	0	byte (A) ← imm4	Z	*	-	-	-	<b>R</b>	*	-	-	-
MOVX A,dir	2	2	(b)	byte (A) ← (dir)	X	*	-	-	-	*	*	-	-	-
MOVX A,addr16	3	2	(b)	byte (A) ← (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX A,Ri	2	1	0	byte (A) ← (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX A,ear	2	1	0	byte (A) ← (ear)	X	*	-	-	-	*	*	-	-	-
MOVX A,eam	2+	2+(a)	(b)	byte (A) ← (eam)	X	*	-	-	-	*	*	-	-	-
MOVX A,io	2	2	(b)	byte (A) ← (io)	X	*	-	-	-	*	*	-	-	-
MOVX A,#imm8	2	2	0	byte (A) ← (imm8)	X	*	-	-	-	*	*	-	-	-
MOVX A,@A	2	2	(b)	byte (A) ← ((A))	X	-	-	-	-	*	*	-	-	-
MOVX A,@RWi+disp8	2	3	(b)	byte (A) ← ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX A,@RLi+disp8	3	6	(b)	byte (A) ← ((RLi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX A,@SP+disp8	3	3	(b)	byte (A) ← ((SP)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVXPX A,addr24	5	3	(b)	byte (A) ← (addr24)	X	*	-	-	-	*	*	-	-	-
MOVXPX A,@A	2	2	(b)	byte (A) ← ((A))	X	-	-	-	-	*	*	-	-	-
MOV dir,A	2	2	(b)	byte (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV addr16,A	3	2	(b)	byte (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV Ri,A	1	1	0	byte (Ri) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV ear,A	2	2	0	byte (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV eam,A	2+	2+(a)	(b)	byte (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV io,A	2	2	(b)	byte (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV @RLi+disp8,A	3	6	(b)	byte ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV @SP+disp8,A	3	3	(b)	byte ((SP)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVP addr24,A	5	3	(b)	byte (addr24) ← (A)	-	-	-	-	-	*	*	-	-	-

The notation (a) in the ~ 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.5a Transfer Instructions (Byte): 50 Instructions (continued)**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV Ri,ear	2	2	0	byte (Ri) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOV Ri,eam	2+	3+(a)	(b)	byte (Ri) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVP @A,Ri	2	3	(b)	byte ((A)) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV ear,Ri	2	3	0	byte (ear) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV eam,Ri	2+	3+(a)	(b)	byte (eam) ← (Ri)	-	-	-	-	-	*	*	-	-	-
MOV Ri,#imm8	2	2	0	byte (Ri) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV io,#imm8	3	3	(b)	byte (io) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV dir,#imm8	3	3	(b)	byte (dir) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV ear,#imm8	3	2	0	byte (ear) ← imm8	-	-	-	-	-	*	*	-	-	-
MOV eam,#imm8	3+	2+(a)	(b)	byte (eam) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV @AL,AH / MOV @A,T	2	2	(b)	byte ((A)) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCH A,ear	2	3	0	byte (A) ↔ (ear)	Z	-	-	-	-	-	-	-	-	-
XCH A,eam	2+	3+(a)	2x(b)	byte (A) ↔ (eam)	Z	-	-	-	-	-	-	-	-	-
XCH Ri,ear	2	4	0	byte (Ri) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCH Ri,eam	2+	5+(a)	2x(b)	byte (Ri) ↔ (eam)	-	-	-	-	-	-	-	-	-	-

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

Table B.5b Transfer Instructions (Word): 40 Instructions

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A,dir	2	2	(c)	word (A) ← (dir)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr16	3	2	(c)	word (A) ← (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW A,SP	1	2	0	word (A) ← (SP)	-	*	-	-	-	*	*	-	-	-
MOVW A,RWi	1	1	0	word (A) ← (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW A,ear	2	1	0	word (A) ← (ear)	-	*	-	-	-	*	*	-	-	-
MOVW A,eam	2+	2+(a)	(c)	word (A) ← (eam)	-	*	-	-	-	*	*	-	-	-
MOVW A,io	2	2	(c)	word (A) ← (io)	-	*	-	-	-	*	*	-	-	-
MOVW A,@A	2	2	(c)	word (A) ← ((A))	-	-	-	-	-	*	*	-	-	-
MOVW A,#imm16	3	2	0	word (A) ← imm16	-	*	-	-	-	*	*	-	-	-
MOVW A,@RWi+disp8	2	3	(c)	word (A) ← ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,@RLi+disp8	3	6	(c)	word (A) ← ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,@SP+disp8	3	3	(c)	word (A) ← ((SP)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW A,addr24	5	3	(c)	word (A) ← (addr24)	-	*	-	-	-	*	*	-	-	-
MOVW A,@A	2	2	(c)	word (A) ← ((A))	-	-	-	-	-	*	*	-	-	-
MOVW dir,A	2	2	(c)	word (dir) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW addr16,A	3	2	(c)	word (addr16) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW SP,#imm16	4	2	0	word (SP) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW SP,A	1	2	0	word (SP) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,A	1	1	0	word (RWi) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW ear,A	2	2	0	word (ear) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW eam,A	2+	2+(a)	(c)	word (eam) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW io,A	2	2	(c)	word (io) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RWi+disp8,A	2	3	(c)	word ((RWi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @RLi+disp8,A	3	6	(c)	word ((RLi)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @SP+disp8,A	3	3	(c)	word ((SP)+disp8) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW A,addr24,A	5	3	(c)	word (addr24) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVW @A,RWi	2	3	(c)	word ((A)) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,ear	2	2	0	word (RWi) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,eam	2+	3+(a)	(c)	word (RWi) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVW ear,RWi	2	3	0	word (ear) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW eam,RWi	2+	3+(a)	(c)	word (eam) ← (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW RWi,#imm16	3	2	0	word (RWi) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW io,#imm16	4	3	(c)	word (io) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW ear,#imm16	4	2	0	word (ear) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW eam,#imm16	4+	2+(a)	(c)	word (eam) ← imm16	-	-	-	-	-	*	*	-	-	-
MOVW @AL,AH / MOVW @A,T	2	2	(c)	word (A) ← (AH)	-	-	-	-	-	*	*	-	-	-
XCHW A,ear	2	3	0	word (A) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW A,eam	2+	3+(a)	2x(c)	word (A) ↔ (eam)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,ear	2	4	0	word (RWi) ↔ (ear)	-	-	-	-	-	-	-	-	-	-
XCHW RWi,eam	2+	5+(a)	2x(c)	word (RWi) ↔ (eam)	-	-	-	-	-	-	-	-	-	-

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.5c Transfer Instructions (Long-Word): 11 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVL A,ear	2	2	0	long (A) ← (ear)	-	-	-	-	-	*	*	-	-	-
MOVL A,eam	2+	3+(a)	(d)	long (A) ← (eam)	-	-	-	-	-	*	*	-	-	-
MOVL A,#imm32	5	3	0	long (A) ← imm32	-	-	-	-	-	*	*	-	-	-
MOVL A,@SP+disp8	3	4	(d)	long (A) ← ((SP)+disp8)	-	-	-	-	-	*	*	-	-	-
MOVPL A,addr24	5	4	(d)	long (A) ← (addr24)	-	-	-	-	-	*	*	-	-	-
MOVPL A,@A	2	3	(d)	long (A) ← ((A))	-	-	-	-	-	*	*	-	-	-
MOVPL @A,RLi	2	5	(d)	long ((A)) ← (RLi)	-	-	-	-	-	*	*	-	-	-
MOVL @SP+disp8,A	3	4	(d)	long ((SP)) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVPL addr24,A	5	4	(d)	long (addr24) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVL ear,A	2	2	0	byte (ear1) ← (A)	-	-	-	-	-	*	*	-	-	-
MOVL eam,A	2+	3+(a)	(d)	long (eam1) ← (A)	-	-	-	-	-	*	*	-	-	-

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

## B.6 Numerical Calculation Instructions

**Table B.6a Add/Deduct Instructions (Byte, Word, Long-Word): 42 Instructions (continued)**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD A,#imm8	2	2	0	byte (A) ← (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD A,dir	2	3	(b)	byte (A) ← (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD A,ear	2	2	0	byte (A) ← (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD A,eam	2+	3+(a)	(b)	byte (A) ← (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD ear,A	2	2	0	byte (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	*
ADD eam,A	2+	3+(a)	2x(b)	byte (eam) ← (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC A	1	2	0	byte (A) ← (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,ear	2	2	0	byte (A) ← (A) + (ear) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A,eam	2+	3+(a)	(b)	byte (A) ← (A) + (eam) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC A	1	3	0	byte (A) ← (AH) + (AL) + (C) (hexadecimal)	Z	-	-	-	-	*	*	*	*	-
SUB A,#imm8	2	2	0	byte (A) ← (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB A,dir	2	3	(b)	byte (A) ← (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB A,ear	2	2	0	byte (A) ← (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB A,eam	2+	3+(a)	(b)	byte (A) ← (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB ear,A	2	2	0	byte (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	*
SUB eam,A	2+	3+(a)	2x(b)	byte (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC A	1	2	0	byte (A) ← (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,ear	2	2	0	byte (A) ← (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A,eam	2+	3+(a)	(b)	byte (A) ← (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC A	1	3	0	byte (A) ← (AH) - (AL) - (C) (hexadecimal)	Z	-	-	-	-	*	*	*	*	-
ADDW A	1	2	0	word (A) ← (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW A,ear	2	2	0	word (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW A,eam	2+	3+(a)	(c)	word (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW A,#imm16	3	2	0	word (A) ← (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW ear,A	2	2	0	word (ear) ← (ear) + (A)	-	-	-	-	-	*	*	*	*	*
ADDW eam,A	2+	3+(a)	2x(c)	word (eam) ← (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW A,ear	2	2	0	word (A) ← (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW A,eam	2+	3+(a)	(c)	word (A) ← (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW A	1	2	0	word (A) ← (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW A,ear	2	2	0	word (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW A,eam	2+	3+(a)	(c)	word (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW A,#imm16	3	2	0	word (A) ← (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW ear,A	2	2	0	word (ear) ← (ear) - (A)	-	-	-	-	-	*	*	*	*	*
SUBW eam,A	2+	3+(a)	2x(c)	word (eam) ← (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW A,ear	2	2	0	word (A) ← (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW A,eam	2+	3+(a)	(c)	word (A) ← (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.



**Table B.6a Add/Deduct Instructions (Byte, Word, Long-Word): 42 Instructions (continued)**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADDL A,ear	2	5	0	long (A) ← (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL A,eam	2+	6+(a)	(d)	long (A) ← (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL A,#imm32	5	4	0	long (A) ← (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL A,ear	2	5	0	long (A) ← (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL A,eam	2+	6+(a)	(d)	long (A) ← (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL A,#imm32	5	4	0	long (A) ← (A) - imm32	-	-	-	-	-	*	*	*	*	-

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.6b Increase/Decrease Instructions (Byte, Word, Long-Word): 12 Instructions**

Mnemonic		#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	2	0	byte (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	*
INC	eam	2+	3+(a)	2x(b)	byte (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	2	0	byte (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	*
DEC	eam	2+	3+(a)	2x(b)	byte (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	2	0	word (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	*
INCW	eam	2+	3+(a)	2x(b)	word (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	2	0	word (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	*
DECW	eam	2+	3+(a)	2x(b)	word (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCL	ear	2	4	0	long (ear) $\leftarrow$ (ear) + 1	-	-	-	-	-	*	*	*	-	*
INCL	eam	2+	5+(a)	2x(d)	long (eam) $\leftarrow$ (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	4	0	long (ear) $\leftarrow$ (ear) - 1	-	-	-	-	-	*	*	*	-	*
DECL	eam	2+	5+(a)	2x(d)	long (eam) $\leftarrow$ (eam) - 1	-	-	-	-	-	*	*	*	-	*

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.6c Figure B.6c. Comparison Instructions (Byte, Word, Long-Word): 11 Instructions**

Mnemonic		#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	2	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	2	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	2+(a)	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2	2	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	2	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	2	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	2+(a)	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3	2	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	3	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	4+(a)	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.6d Unsigned Multiply/Divide Instructions (Word, Long-Word): 11 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU A	1	*1	0	word (AH) / byte (AL) Quotient → byte (AL) Remainder → byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU A,ear	2	*2	0	word (A) / byte (ear) Quotient → byte (A) Remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU A,eam	2+	*3	*6	word (A) / byte (eam) Quotient → byte (A) Remainder → byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW A,ear	2	*4	0	long (A) / word (ear) Quotient → word (A) Remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVUW A,eam	2+	*5	*7	long (A) / word (eam) Quotient → word (A) Remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MULU A	1	*8	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,ear	2	*9	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MULU A,eam	2+	*10	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULUW A	1	*11	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,ear	2	*12	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW A,eam	2+	*13	(b)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

- \*1: 3 in case of zero division, 6 in case of overflow, normally 14.
- \*2: 3 in case of zero division, 5 in case of overflow, normally 13.
- \*3: 5 + (a) in case of zero division, 7 + (a) in case of overflow, normally 17 + (a).
- \*4: 3 in case of zero division, 5 in case of overflow, normally 21.
- \*5: 4 + (a) in case of zero division, 7 + (a) in case of overflow, normally 25 + (a).
- \*6: (b) in case of zero division or overflow, normally 2 × (b).
- \*7: (c) in case of zero division or overflow, normally 2 × (c).
- \*8: 3 when byte (AH) is zero, 7 otherwise.
- \*9: 3 when byte (ear) is zero, 7 otherwise.
- \*10: 4 + (a) when byte (eam) is zero, 8 + (a) otherwise
- \*11: 3 when word (AH) is zero, 11 otherwise.
- \*12: 3 when word (ear) is zero, 11 otherwise
- \*13: 4 + (a) when word (eam) is zero, 12 + (a) otherwise

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.6e Signed Multiply/Divide Instructions (Word, Long-Word): 11 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV A	2	*1	0	word (AH) / byte (AL) Quotient → byte (AL) Remainder → byte (AH)	Z	-	-	-	-	-	-	*	*	-
DIV A,ear	2	*2	0	word (A) / byte (ear) Quotient → byte (A) Remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIV A,eam	2+	*3	*6	word (A) / byte (eam) Quotient → byte (A) Remainder → byte (ear)	Z	-	-	-	-	-	-	*	*	-
DIVW A,ear	2	*4	0	long (A) / word (ear) Quotient → word (A) Remainder → word (ear)	-	-	-	-	-	-	-	*	*	-
DIVW A,eam	2+	*5	*7	long (A) / word (eam) Quotient → word (A) Remainder → word (eam)	-	-	-	-	-	-	-	*	*	-
MUL A	2	*8	0	byte (AH) * byte (AL) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,ear	2	*9	0	byte (A) * byte (ear) → word (A)	-	-	-	-	-	-	-	-	-	-
MUL A,eam	2+	*10	(b)	byte (A) * byte (eam) → word (A)	-	-	-	-	-	-	-	-	-	-
MULW A	2	*11	0	word (AH) * word (AL) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,ear	2	*12	0	word (A) * word (ear) → Long (A)	-	-	-	-	-	-	-	-	-	-
MULW A,eam	2+	*13	(b)	word (A) * word (eam) → Long (A)	-	-	-	-	-	-	-	-	-	-

\*1: 3 in case of zero division, 8 or 18 in case of overflow, normally 18.

\*2: 3 in case of zero division, 10 or 21 in case of overflow, normally 22.

\*3: 4 + (a) in case of zero division, 11 + (a) or 22 + (a) in case of overflow, normally 23 + (a).

\*4: For positive dividends: 4 in case of zero division, 10 or 29 in case of overflow, normally 30.  
For negative dividends: 4 in case of zero division, 11 or 30 in case of overflow, normally 31.

\*5: For positive dividends: 4 + (a) in case of zero division, 11 + (a) or 30 + (a) in case of overflow, normally 31 + (a).  
For negative dividends: 4 + (a) in case of zero division, 12 + (a) or 31 + (a) in case of overflow, normally 32 + (a).

\*6: (b) in case of zero division or overflow, normally  $2 \times (b)$ .

\*7: (c) in case of zero division or overflow, normally  $2 \times (c)$ .

\*8: 3 when byte (AH) is zero, 12 when result is positive, 13 when result is negative.

\*9: 3 when byte (ear) is zero, 12 when result is positive, 13 when result is negative.

\*10: 4 + (a) when byte (eam) is zero, 13 + (a) when result is positive, 14 + (a) when result is negative.

\*11: 3 when word (AH) is zero, 12 when result is positive, 13 when result is negative.

\*12: 3 when word (ear) is zero, 16 when result is positive, 19 when result is negative.

\*13: 4 + (a) when word (eam) is zero, 17 + (a) when result is positive, 20 + (a) when result is negative.

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Note:** Two different actual cycle counts are given for DIV instruction and DIVW instructions ending in overflow, representing detection before or after calculation.

When DIV instructions or DIVW instructions result in overflow, the contents of the AL are destroyed.

## B.7 Logical Calculation Instructions

**Table B.7a Logical Instructions (Byte, Word): 39 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND A,#imm8	2	2	0	byte (A) ← (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND A,ear	2	2	0	byte (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND A,eam	2+	3+(a)	(b)	byte (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND ear,A	2	3	0	byte (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	*
AND eam,A	2+	3+(a)	2x(b)	byte (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
OR A,#imm8	2	2	0	byte (A) ← (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR A,ear	2	2	0	byte (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR A,eam	2+	3+(a)	(b)	byte (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR ear,A	2	3	0	byte (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	*
OR eam,A	2+	3+(a)	2x(b)	byte (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XOR A,#imm8	2	2	0	byte (A) ← (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR A,ear	2	2	0	byte (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR A,eam	2+	3+(a)	(b)	byte (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR ear,A	2	3	0	byte (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	*
XOR eam,A	2+	3+(a)	2x(b)	byte (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOT A	1	2	0	byte (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOT ear	2	2	0	byte (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	*
NOT eam	2+	3+(a)	2x(b)	byte (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW A	1	2	0	word (A) ← (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW A,#imm16	3	2	0	word (A) ← (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW A,ear	2	2	0	word (A) ← (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW A,eam	2+	3+(a)	(c)	word (A) ← (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW ear,A	2	3	0	word (ear) ← (ear) and (A)	-	-	-	-	-	*	*	R	-	*
ANDW eam,A	2+	3+(a)	2x(c)	word (eam) ← (eam) and (A)	-	-	-	-	-	*	*	R	-	*
ORW A	1	2	0	word (A) ← (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW A,#imm16	3	2	0	word (A) ← (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW A,ear	2	2	0	word (A) ← (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW A,eam	2+	3+(a)	(c)	word (A) ← (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW ear,A	2	3	0	word (ear) ← (ear) or (A)	-	-	-	-	-	*	*	R	-	*
ORW eam,A	2+	3+(a)	2x(c)	word (eam) ← (eam) or (A)	-	-	-	-	-	*	*	R	-	*
XORW A	1	2	0	word (A) ← (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW A,#imm16	3	2	0	word (A) ← (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW A,ear	2	2	0	word (A) ← (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW A,eam	2+	3+(a)	(c)	word (A) ← (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW ear,A	2	3	0	word (ear) ← (ear) xor (A)	-	-	-	-	-	*	*	R	-	*
XORW eam,A	2+	3+(a)	2x(c)	word (eam) ← (eam) xor (A)	-	-	-	-	-	*	*	R	-	*
NOTW A	1	2	0	word (A) ← not (A)	-	-	-	-	-	*	*	R	-	-
NOTW ear	2	2	0	word (ear) ← not (ear)	-	-	-	-	-	*	*	R	-	*
NOTW eam	2+	3+(a)	2x(c)	word (eam) ← not (eam)	-	-	-	-	-	*	*	R	-	*

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.7b Logical Instructions (Long-Word): 9 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A,ear	2	5	0	long (A) ← (A) and (ear)	-	-	-	-	-	-	*	*	R	-
ANDL A,eam	2+	6+(a)	(d)	long (A) ← (A) and (eam)	-	-	-	-	-	-	*	*	R	-
ORL A,ear	2	5	0	long (A) ← (A) or (ear)	-	-	-	-	-	-	*	*	R	-
ORL A,eam	2+	6+(a)	(d)	long (A) ← (A) or (eam)	-	-	-	-	-	-	*	*	R	-
XORL A,ear	2	5	0	long (A) ← (A) xor (ear)	-	-	-	-	-	-	*	*	R	-
XORL A,eam	2+	6+(a)	(d)	long (A) ← (A) xor (eam)	-	-	-	-	-	-	*	*	R	-

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.7c Sign Inversion Instructions (Long-Word): 9 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG A	1	2	0	byte (A) ← 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG ear	2	2	0	byte (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	*
NEG eam	2+	3+(a)	2x(b)	byte (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW A	1	2	0	word (A) ← 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW ear	2	2	0	word (ear) ← 0 - (ear)	-	-	-	-	-	*	*	*	*	*
NEGW eam	2+	3+(a)	2x(c)	word (eam) ← 0 - (eam)	-	-	-	-	-	*	*	*	*	*

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.7d Absolute Value Instructions (Byte, Word, Long-Word): 3 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ABS A	2	2	0	byte (A) ← Absolute value (A)	Z	-	-	-	-	*	*	*	-	-
ABSW A	2	2	0	word (A) ← Absolute value (A)	-	-	-	-	-	*	*	*	-	-
ABSL A	2	4	0	long (A) ← Absolute value (A)	-	-	-	-	-	*	*	*	-	-

**Table B.7e Normalize Instruction (Long-Word): 1 Instruction**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A,R0	2	*1	0	long (A) ← Shift to the position where 1 was formerly placed byte (R0) ← Number of shifts at that time	-	-	-	-	-	-	*	-	-	-

\*1: 5 when all accumulator are '0,' otherwise 5 + (R0).

## B.8 Shift Instructions

**Table B.8a Shift Instructions (Byte, Word, Long-Word): 27 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	byte (A) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	-
ROLC A	2	2	0	byte (A) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	-
RORC ear	2	2	0	byte (ear) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	*
RORC eam	2+	3+(a)	2x(b)	byte (eam) ← Right rotate with carry	-	-	-	-	-	*	*	-	*	*
ROLC ear	2	2	0	byte (ear) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	*
ROLC eam	2+	3+(a)	2x(b)	byte (eam) ← Left rotate with carry	-	-	-	-	-	*	*	-	*	*
ASR A,RO	2	*1	0	byte (A) ← Arithmetic right barrel shift (A,RO)	-	-	-	-	*	*	*	-	*	-
LSR A,RO	2	*1	0	byte (A) ← Logical right barrel shift (A,RO)	-	-	-	-	*	*	*	-	*	-
LSL A,RO	2	*1	0	byte (A) ← Logical left barrel shift (A,RO)	-	-	-	-	-	*	*	-	*	-
ASR A,#imm8	3	*3	0	byte (A) ← Arithmetic right barrel shift (A,#imm8)	-	-	-	-	*	*	*	-	*	-
LSR A,#imm8	3	*3	0	byte (A) ← Logical right barrel shift (A,#imm8)	-	-	-	-	*	*	*	-	*	-
LSL A,#imm8	3	*3	0	byte (A) ← Logical left barrel shift (A,#imm8)	-	-	-	-	-	*	*	-	*	-
ASRW A	1	2	0	word (A) ← Arithmetic right shift (A,1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW A / SHRW A	1	2	0	word (A) ← Logical right shift (A,1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW A / SHLW A	1	2	0	word (A) ← Logical left shift (A,1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW A,RO	2	*1	0	word (A) ← Arithmetic right barrel shift (A,RO)	-	-	-	-	*	*	*	-	*	-
LSRW A,RO	2	*1	0	word (A) ← Logical right barrel shift (A,RO)	-	-	-	-	*	*	*	-	*	-
LSLW A,RO	2	*1	0	word (A) ← Logical left barrel shift (A,RO)	-	-	-	-	-	*	*	-	*	-
ASRW A,#imm8	3	*3	0	word (A) ← Arithmetic right barrel shift (A,#imm8)	-	-	-	-	*	*	*	-	*	-
LSRW A,#imm8	3	*3	0	word (A) ← Logical right barrel shift (A,#imm8)	-	-	-	-	*	*	*	-	*	-
LSLW A,#imm8	3	*3	0	word (A) ← Logical left barrel shift (A,#imm8)	-	-	-	-	-	*	*	-	*	-
ASRL A,RO	2	*2	0	long (A) ← Arithmetic right barrel shift (A,RO)	-	-	-	-	*	*	*	-	*	-
LSRL A,RO	2	*2	0	long (A) ← Logical right barrel shift (A,RO)	-	-	-	-	*	*	*	-	*	-
LSLL A,RO	2	*2	0	long (A) ← Logical left barrel shift (A,RO)	-	-	-	-	-	*	*	-	*	-
ASRL A,#imm8	3	*4	0	long (A) ← Arithmetic right barrel shift (A,#imm8)	-	-	-	-	*	*	*	-	*	-
LSRL A,#imm8	3	*4	0	long (A) ← Logical right barrel shift (A,#imm8)	-	-	-	-	*	*	*	-	*	-
LSLL A,#imm8	3	*4	0	long (A) ← Logical left barrel shift (A,#imm8)	-	-	-	-	-	*	*	-	*	-

\*1: 3 when R0 is zero, otherwise 3 + (R0)

\*2: 3 when R0 is zero, otherwise 4 + (R0)

\*3: 3 when imm8 is zero, otherwise 3 + imm8

\*4: 3 when imm8 is zero, otherwise 4 + imm8

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

## B.9 Branching Instructions

**Table B.9a Branching Instructions (I): 31 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ / BEQ	rel	2	*1	0	Branch when (Z) = 1	-	-	-	-	-	-	-	-	-
BNZ / BNE	rel	2	*1	0	Branch when (Z) = 0	-	-	-	-	-	-	-	-	-
BC / BLO	rel	2	*1	0	Branch when (C) = 1	-	-	-	-	-	-	-	-	-
BNC / BHS	rel	2	*1	0	Branch when (C) = 0	-	-	-	-	-	-	-	-	-
BN	rel	2	*1	0	Branch when (N) = 1	-	-	-	-	-	-	-	-	-
BP	rel	2	*1	0	Branch when (N) = 0	-	-	-	-	-	-	-	-	-
BV	rel	2	*1	0	Branch when (V) = 1	-	-	-	-	-	-	-	-	-
BNV	rel	2	*1	0	Branch when (V) = 0	-	-	-	-	-	-	-	-	-
BT	rel	2	*1	0	Branch when (T) = 1	-	-	-	-	-	-	-	-	-
BNT	rel	2	*1	0	Branch when (T) = 0	-	-	-	-	-	-	-	-	-
BLT	rel	2	*1	0	Branch when (V) xor (N) = 1	-	-	-	-	-	-	-	-	-
BGE	rel	2	*1	0	Branch when (V) xor (N) = 0	-	-	-	-	-	-	-	-	-
BLE	rel	2	*1	0	Branch when ((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-
BGT	rel	2	*1	0	Branch when ((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-
BLS	rel	2	*1	0	Branch when (C) or (Z) = 1	-	-	-	-	-	-	-	-	-
BHI	rel	2	*1	0	Branch when (C) or (Z) = 0	-	-	-	-	-	-	-	-	-
BRA	rel	2	*1	0	Unconditional branching	-	-	-	-	-	-	-	-	-
JMP	@A	1	2	0	word (PC) ← (A)	-	-	-	-	-	-	-	-	-
JMP	@addr16	3	2	0	word (PC) ← addr16	-	-	-	-	-	-	-	-	-
JMP	@ear	2	3	0	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-
JMP	@eam	2+	4+(a)	(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-
JMPP	@ear *3	2	3	0	word (PC) ← (ear). (PCB) ← (ear+2)	-	-	-	-	-	-	-	-	-
JMPP	@eam *3	2+	4+(a)	(d)	word (PC) ← (eam). (PCB) ← (eam+2)	-	-	-	-	-	-	-	-	-
JMPP	addr24	4	3	0	word (PC) ← ad24 0-15. (PCB) ← ad24 16-23	-	-	-	-	-	-	-	-	-
CALL	@ear *4	2	4	(c)	word (PC) ← (ear)	-	-	-	-	-	-	-	-	-
CALL	@eam *4	2+	5+(a)	2x(c)	word (PC) ← (eam)	-	-	-	-	-	-	-	-	-
CALL	addr16 *5	3	5	(c)	word (PC) ← addr16	-	-	-	-	-	-	-	-	-
CALLV	#vct4 *5	1	5	2x(c)	Vector call instruction	-	-	-	-	-	-	-	-	-
CALLP	@ear *6	2	7	2x(c)	word (PC) ← (ear) 0-15. (PCB) ← (ear)16-23	-	-	-	-	-	-	-	-	-
CALLP	@eam *6	2+	8+(a)	*2	word (PC) ← (eam) 0-15. (PCB) ← (eam)16-23	-	-	-	-	-	-	-	-	-
CALLP	addr24 *7	4	7	2x(c)	word (PC) ← addr0-15. (PCB) ← addr16-23	-	-	-	-	-	-	-	-	-

\*1: 3 when branching occurs, otherwise 2.

\*2:  $3 \times (c) \div (b)$

\*3: Read branch destination address (word)

\*4: Write: Save to stack (word), Read: read branch destination address (word)

\*5: Save to stack (word)

\*6: Write: Save into stack (long-word), Read: read branch destination address (long-word)

\*7: Save to stack (long-word)

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.



**Table B.9b Branching Instructions (II): 20 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE A,#imm8,rel	3	*1	0	Branch when byte (A) ≠ imm8	-	-	-	-	-	*	*	*	*	-
CWBNE A,#imm16,rel	4	*1	0	Branch when word (A)≠ imm16	-	-	-	-	-	*	*	*	*	-
CBNE ear,#imm8,rel	4	*1	0	Branch when byte (ear)≠ imm8	-	-	-	-	-	*	*	*	*	-
CBNE eam,#imm8,rel	4+	*3	(b)	Branch when byte (eam)≠ imm8	-	-	-	-	-	*	*	*	*	-
CWBNE ear,#imm16,rel	5	*1	0	Branch when word (ear)≠ imm16	-	-	-	-	-	*	*	*	*	-
CWBNE eam,#imm16,rel	5+	*3	(c)	Branch when word (eam)≠ imm16	-	-	-	-	-	*	*	*	*	-
DBNZ ear,rel	3	*2	0	Branch when byte (ear)=(ear)-1. (ear)≠ 0	-	-	-	-	-	*	*	*	-	*
DBNZ eam,rel	3+	*4	2x(b)	Branch when byte (eam)=(eam)-1. (eam)≠ 0	-	-	-	-	-	*	*	*	-	*
DWBNZ ear,rel	3	*2	0	Branch when word (ear)=(ear)-1. (ear)≠ 0	-	-	-	-	-	*	*	*	-	*
DWBNZ eam,rel	3+	*4	2x(c)	Branch when word (eam)=(eam)-1. (eam)≠ 0	-	-	-	-	-	*	*	*	-	*
INT #vct8	2	14	8x(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT addr16	3	12	6x(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP addr24	4	13	6x(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9	1	14	8x(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI	1	9	6x(c)	Recovery from interrupt	-	-	*	*	*	*	*	*	*	-
RETIQ *6	2	11	*5	Recovery from interrupt	-	-	*	*	*	*	*	*	*	-
LINK #imm8	2	6	(c)	At the entrance of function, save old frame pointers into a stack, set up new frame pointers, reserve area for local pointers.	-	-	-	-	-	-	-	-	-	-
UNLINK	1	5	(c)	At the exit of function, recover the old frame pointers from the stack.	-	-	-	-	-	-	-	-	-	-
RET *7	1	4	(c)	Recover from the subroutine.	-	-	-	-	-	-	-	-	-	-
RETP *8	1	5	(d)	Recover from the subroutine.	-	-	-	-	-	-	-	-	-	-

\*1: 4 when branching occurs, otherwise 3.

\*2: 5 when branching occurs, otherwise 4.

\*3: 5 + (a) when branching occurs, otherwise 4 + (a).

\*4: 6 + (a) when branching occurs, otherwise 5 + (a).

\*5: 3 x (b) + 2 x (c) when an interrupt request occurs, 6 x (c) for recovery.

\*6: High speed interrupt recovery instruction. If an interrupt request occurs and an interrupt is generated while this instruction is executing, operation will branch to the interrupt vector without a stack operation.

\*7: Recover from stack (word).

\*8: Recover from stack (long-word).

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

## B.10 Other Instructions

**Table B.10a Other Instructions (Byte, Word, Long-Word): 36 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW A	1	3	(c)	word (SP) ← (SP) - 2, ((SP)) ← (A)	-	-	-	-	-	-	-	-	-	-
PUSHW AH	1	3	(c)	word (SP) ← (SP) - 2, ((SP)) ← (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW PS	1	3	(c)	word (SP) ← (SP) - 2, ((SP)) ← (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW rlst	2	*3	*4	(SP) ← (SP) - 2n, ((SP)) ← (rlst)	-	-	-	-	-	-	-	-	-	-
POPW A	1	3	(c)	word (A) ← ((SP)), (SP) ← (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW AH	1	3	(c)	word (AH) ← ((SP)), (SP) ← (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW PS	1	3	(c)	word (PS) ← ((SP)), (SP) ← (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW rlst	2	*2	*4	(rlst) ← ((SP)), (SP) ← (SP)	-	-	-	-	-	-	-	-	-	-
JCTX @a	1	9	6x(c)	Context switching instruction	-	-	*	*	*	*	*	*	*	-
AND CCR,#imm8	2	3	0	byte (CCR) ← (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR CCR,#imm8	2	3	0	byte (CCR) ← (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV RP,#imm8	2	2	0	byte (RP) ← imm8	-	-	-	-	-	-	-	-	-	-
MOV ILM,#imm8	2	2	0	byte (ILM) ← imm8	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,ear	2	3	0	word (RWi) ← ear	-	-	-	-	-	-	-	-	-	-
MOVEA RWi,eam	2+	2+(a)	0	word (RWi) ← eam	-	-	-	-	-	-	-	-	-	-
MOVEA A,ear	2	2	0	word (A) ← ear	-	*	-	-	-	-	-	-	-	-
MOVEA A,eam	2+	1+(a)	0	word (A) ← eam	-	*	-	-	-	-	-	-	-	-
ADDSP #imm8	2	3	0	word (SP) ← ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP #imm16	3	3	0	word (SP) ← imm16	-	-	-	-	-	-	-	-	-	-
MOV A,brg1	2	*1	0	byte (A) ← (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV brg2,A	2	1	0	byte (brg2) ← (A)	-	-	-	-	-	*	*	-	-	-
MOV brg2,#imm8	3	2	0	byte (brg2) ← imm8	-	-	-	-	-	*	*	-	-	-
NOP	1	1	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB	1	1	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB	1	1	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB	1	1	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB	1	1	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC	1	1	0	Prefix code for flag unchange setting	-	-	-	-	-	-	-	-	-	-
CMR	1	1	0	Prefix for common register banks	-	-	-	-	-	-	-	-	-	-

\*1: PCB, ADB, SSB, USB, SPB ... 1  
DTB ... 2  
DPR ... 3

\*2: 3+4\*(POP cycle count)

\*3: 3+4\*(PUSH cycle count)

\*4: (POP cycle) × (c), or (PUSH cycle) × (c)

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

B.10 Other Instructions

**Table B.10a Other Instructions (Byte, Word, Long-Word): 36 Instructions (continued)**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW SPCU,#imm16	4	2	0	word (SPCU) ← (imm16)	-	-	-	-	-	-	-	-	-	-
MOVW SPCL,#imm16	4	2	0	word (SPCL) ← (imm16)	-	-	-	-	-	-	-	-	-	-
SETSPC	2	2	0	Stack check operation enabled	-	-	-	-	-	-	-	-	-	-
CLRSPC	2	2	0	Stack check operation disabled	-	-	-	-	-	-	-	-	-	-
BTSCN A	2	*1	0	byte (A) ← word (A) 1's bit position	Z	-	-	-	-	-	*	-	-	-
BTSCNS A	2	*2	0	byte (A) ← word (A) 1's bit position x 2	Z	-	-	-	-	-	*	-	-	-
BTSCND A	2	*3	0	byte (A) ← word (A) 1's bit position x 4	Z	-	-	-	-	-	*	-	-	-

\*1: 3 when AL is zero, 5 otherwise

\*2: 4 when AL is zero, 6 otherwise

\*3: 5 when AL is zero, 7 otherwise

Table B.10b Bit Operation Instructions: 21 Instructions

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB A,dir:bp	3	3	(b)	byte (A) ← ( dir:bp )b	Z	*	-	-	-	*	*	-	-	-
MOVB A,addr16:bp	4	3	(b)	byte (A) ← ( addr16:bp )b	Z	*	-	-	-	*	*	-	-	-
MOVB A,io:bp	3	3	(b)	byte (A) ← ( io:bp )b	Z	*	-	-	-	*	*	-	-	-
MOVB dir:bp,A	3	4	2x(b)	bit ( dir:bp )b ← (A)	-	-	-	-	-	*	*	-	-	*
MOVB addr16:bp,A	4	4	2x(b)	bit ( addr16:bp )b ← (A)	-	-	-	-	-	*	*	-	-	*
MOVB io:bp,A	3	4	2x(b)	bit ( io:bp )b ← (A)	-	-	-	-	-	*	*	-	-	*
SETB dir:bp	3	4	2x(b)	bit ( dir:bp )b ← 1	-	-	-	-	-	-	-	-	-	*
SETB addr16:bp	4	4	2x(b)	bit ( addr16:bp )b ← 1	-	-	-	-	-	-	-	-	-	*
SETB io:bp	3	4	2x(b)	bit ( io:bp )b ← 1	-	-	-	-	-	-	-	-	-	*
CLRB dir:bp	3	4	2x(b)	bit ( dir:bp )b ← 0	-	-	-	-	-	-	-	-	-	*
CLRB addr16:bp	4	4	2x(b)	bit ( addr16:bp )b ← 0	-	-	-	-	-	-	-	-	-	*
CLRB io:bp	3	4	2x(b)	bit ( io:bp )b ← 0	-	-	-	-	-	-	-	-	-	*
BBC dir:bp,rel	4	*1	(b)	Branch when ( dir:bp )b = 0	-	-	-	-	-	-	*	-	-	-
BBC addr16:bp,rel	5	*1	(b)	Branch when ( addr16:bp )b = 0	-	-	-	-	-	-	*	-	-	-
BBC io:bp,rel	4	*1	(b)	Branch when ( io:bp )b = 0	-	-	-	-	-	-	*	-	-	-
BBS dir:bp,rel	4	*1	(b)	Branch when ( dir:bp )b = 1	-	-	-	-	-	-	*	-	-	-
BBS addr16:bp,rel	5	*1	(b)	Branch when ( addr16:bp )b = 1	-	-	-	-	-	-	*	-	-	-
BBS io:bp,rel	4	*1	(b)	Branch when ( io:bp )b = 1	-	-	-	-	-	-	*	-	-	-
SBBS addr16:bp,rel	5	*2	2x(b)	Branch when ( addr16:bp )b = 1, bit = 1	-	-	-	-	-	-	*	-	-	*
WBTS io:bp	3	*3	*4	Wait until ( io:bp )b = 1	-	-	-	-	-	-	-	-	-	-
WBTC io:bp	3	*3	*4	Wait until ( io:bp )b = 0	-	-	-	-	-	-	-	-	-	-

\*1: 5 when branching occurs, otherwise 4

\*2: 7 when conditions are met, 6 otherwise

\*3: Undefined cycle count

\*4: Until conditions are met

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.10c Accumulator Operation Instructions (Byte, Word): 6 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	byte (A)0-7 $\leftrightarrow$ (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW / XCHW A,T	1	2	0	word (AH) $\leftrightarrow$ (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	byte signed extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	word signed extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	2	0	word zero extension	-	Z	-	-	-	R	*	-	-	-

**Table B.10d String Instructions: 10 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2	*3	byte transfer @AH+ $\leftarrow$ @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2	*3	byte transfer @AH- $\leftarrow$ @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1	*4	byte search @AH+ $\leftarrow$ AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1	*4	byte search @AH- $\leftarrow$ AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	5m+3	*5	byte fill @AH+ $\leftarrow$ AL, counter = RW0	-	-	-	-	-	*	*	-	-	-
MOVSW / MOVSWI	2	*2	*6	word transfer @AH+ $\leftarrow$ @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2	*6	word transfer @AH- $\leftarrow$ @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1	*7	word search @AH+ $\leftarrow$ AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1	*7	word search @AH- $\leftarrow$ AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	5m+3	*8	word fill @AH+ $\leftarrow$ AL, counter = RW0	-	-	-	-	-	*	*	-	-	-

\*1: 3 when RW0 is zero,  $2+6 \times (RW0)$  for count-out,  $6n + 4$  for match

\*2: 4 when RW0 is zero, otherwise  $2 + 6 \times (RW0)$

\*3:  $(b) \times (RW0)$

\*4:  $(b) \times n$

\*5:  $(b) \times (RW0)$

\*6:  $(c) \times (RW0)$

\*7:  $(c) \times n$

\*8:  $(c) \times (RW0)$

The notation 'm' in the - 'cycle count' column is the RW0 (counter) value, and 'n' is the number of loop cycles.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

**Table B.10e Multiple Data Transfer Instructions: 18 Instructions**

Mnemonic	#	~	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVM @A,@Rli,#imm8	3	*1	*3	Multiple data transfer byte ((A) ← ((Rli))	-	-	-	-	-	-	-	-	-	-
MOVM @A,eam,#imm8	3+	*2	*3	Multiple data transfer byte ((A) ← (eam)	-	-	-	-	-	-	-	-	-	-
MOVM addr16,@Rli,#imm8	5	*1	*3	Multiple data transfer byte (addr16) ← ((Rli))	-	-	-	-	-	-	-	-	-	-
MOVM addr16,eam,#imm8	5+	*2	*3	Multiple data transfer byte (addr16) ← (eam)	-	-	-	-	-	-	-	-	-	-
MOV MW @A,@Rli,#imm8	3	*1	*4	Multiple data transfer word ((A) ← ((Rli))	-	-	-	-	-	-	-	-	-	-
MOV MW @A,eam,#imm8	3+	*2	*4	Multiple data transfer word ((A) ← (eam))	-	-	-	-	-	-	-	-	-	-
MOV MW addr16,@Rli,#imm8	5	*1	*4	Multiple data transfer word (addr16) ← ((Rli))	-	-	-	-	-	-	-	-	-	-
MOV MW addr16,eam,#imm8	5+	*2	*4	Multiple data transfer word (addr16) ← ((eam))	-	-	-	-	-	-	-	-	-	-
MOVM @Rli,@A,#imm8	3	*1	*3	Multiple data transfer byte ((Rli) ← ((A))	-	-	-	-	-	-	-	-	-	-
MOVM eam,@A,#imm8	3+	*2	*3	Multiple data transfer byte (eam) ← ((A))	-	-	-	-	-	-	-	-	-	-
MOVM @Rli,addr16,#imm8	5	*1	*3	Multiple data transfer byte ((Rli) ← (addr16)	-	-	-	-	-	-	-	-	-	-
MOVM eam,addr16,#imm8	5+	*2	*3	Multiple data transfer byte (eam) ← (addr16)	-	-	-	-	-	-	-	-	-	-
MOV MW @Rli,@A,#imm8	3	*1	*4	Multiple data transfer word ((Rli) ← ((A))	-	-	-	-	-	-	-	-	-	-
MOV MW eam,@A,#imm8	3+	*2	*4	Multiple data transfer word (eam) ← ((A))	-	-	-	-	-	-	-	-	-	-
MOV MW @Rli,addr16,#imm8	5	*1	*4	Multiple data transfer word ((Rli) ← (addr16)	-	-	-	-	-	-	-	-	-	-
MOV MW eam,addr16,#imm8	5+	*2	*4	Multiple data transfer word (eam) ← (addr16)	-	-	-	-	-	-	-	-	-	-
MOVM bnk:addr16, bnk:addr16,#imm8	*5 7	*1	*3	Multiple data transfer byte (bnk:addr16) ← (bnk:addr16)	-	-	-	-	-	-	-	-	-	-
MOV MW bnk:addr16, bnk:addr16,#imm8	*5 7	*1	*4	Multiple data transfer word (bnk:addr16) ← (bnk:addr16)	-	-	-	-	-	-	-	-	-	-

\*1:  $5 + \text{imm8} \times 5$ , 256 cycles when imm8 is zero.

\*2:  $5 + \text{imm8} \times 5 + (a)$ , 256 cycles when imm8 is zero.

\*3: No. of transfers  $\times (b) \times 2$

\*4: No. of transfer  $\times (c) \times 2$

\*5: The bank register indicated by bnk is the same as for the MOV S instruction.

The notation (a) in the - 'cycle count' column indicates a reference to Appendix B.4a.

The notation '+' in the # 'byte count' column indicates a reference to Appendix B.3.

The notations (b), (c) and (d) in the B (compensation value) column indicate a reference to Appendix B.4b.

## B.11 Execution Cycle Counts for Special Operations

This section describes execution time up to start of interrupt processing, and execution cycle counts for extended intelligent I/O service, and exception handling processing for occurrence of stack area errors and for execution of undefined instructions.

### ■ Execution Time up to Start of Interrupt Processing

- (1) Wait time up to transition to CPU interrupt sequence  
(Do not transition to interrupt sequence during instruction execution.)
- (2) Execution time of interrupt sequence
  - Cycle count: 16
  - Compensation value:  $3 \times (b) + 7 \times (c)$
  - Bus operation:
  - Internal register access: byte access 2 cycles
    - Interrupt vector read: word access 1 cycle, byte access 1 cycle
    - Stack write: word access 6 cycles

\*For compensation values (b) and (c) see Appendix B.4b.

Accordingly, the execution time up to start of interrupt processing is represented by (1) plus (2).

### ■ Extended Intelligent I/O Service Execution Time (1 cycle of transfer time)

- (1) When data transfer occurs normally

**Table B.11a Extended Intelligent I/O Service Execution Time (when ISCS SE bit is '0')**

Buffer address pointer	Fixed	Updated	Updated
I/O address pointer	Fixed	Fixed	Updated
Cycle count			
BAP ⇒ IOA	25	25	28
IOA ⇒ BAP	26	26	29
Compensation value	$5 \times (b) + 6 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)	$5 \times (b) + 6 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)	$5 \times (b) + 7 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)
Bus operation			
Internal register access	Word access: 1 cycle, byte access: 3 cycles	Word access: 1 cycle, byte access: 3 cycles	Word access: 1 cycle, byte access: 3 cycles
ISD access	Word access: 5 cycles, byte access: 2 cycles	Word access: 5 cycles, byte access: 2 cycles	Word access: 6 cycles, byte access: 2 cycles
Transfer operation	Read: 1 cycle, write: 1 cycle	Read: 1 cycle, write: 1 cycle	Read: 1 cycle, write: 1 cycle

\*For compensation values (b) and (c) see Appendix B.4b.

Compensation values for transfer operations should be taken into account.

**Table B.11b Extended Intelligent I/O Service Execution Time (when ISCS SE bit is '1')**

Buffer address pointer	Fixed	Updated	Updated
I/O address pointer	Fixed	Fixed	Updated
Cycle count BAP ⇒ IOA IOA ⇒ BAP	26 27	26 27	29 30
Supplemental value	5x(b)+6x(c) + transfer operation (read: 1 cycle, write: 1 cycle)	5x(b)+6x(c) + transfer operation (read: 1 cycle, write: 1 cycle)	5x(b)+7x(c) + transfer operation (read: 1 cycle, write: 1 cycle)
Bus operation Internal register access ISD access Transfer operation	Word access: 1 cycle, byte access: 3 cycles Word access: 5 cycles, byte access: 2 cycles Read: 1 cycle, write: 1 cycle	Word access: 1 cycle, byte access: 3 cycles Word access: 5 cycles, byte access: 2 cycles Read: 1 cycle, write: 1 cycle	Word access: 1 cycle, byte access: 3 cycles Word access: 6 cycles, byte access: 2 cycles Read: 1 cycle, write: 1 cycle

\*For supplemental values (b) and (c) see Appendix B.4b.

Supplemental values for transfer operations should be taken into account.

(2) When halt occurs due to request from peripheral resources

Execute stack operations, and then branch to interrupt processing program

Cycle count: 28

Supplemental value:  $8 \times (b) + 9 \times (c)$

Bus operation:

Internal register access: word address 1 cycle, byte access 5 cycles

ISD access: word access 1 cycle, byte access 2 cycles

Interrupt vector read: word access 1 cycle, byte access 1 cycle

Stack write: word access 6 cycles

\*For compensation values (b) and (c) see Appendix B.4b.



## B.11 Execution Cycle Counts for Special Operations

(3) At end of count

After transfer operation, execute stack operation and branch to interrupt processing program.

**Table B.11c Extended Intelligent I/O Service Execution Time at End of Count (when ISCS SE bit is '0')**

Buffer address pointer	Fixed	Updated	Updated
I/O address pointer	Fixed	Fixed	Updated
Cycle count BAP ⇒ IOA IOA ⇒ BAP	39 40	39 40	42 43
Compensation value	$7 \times (b) + 13 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)	$7 \times (b) + 13 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)	$7 \times (b) + 14 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)
Bus operation Internal register access	Word access: 1 cycle, byte access: 4 cycles	Word access: 1 cycle, byte access: 4 cycles	Word access: 1 cycle, byte access: 4 cycles
ISD access	Word access: 5 cycles, byte access: 2 cycles	Word access: 5 cycles, byte access: 2 cycles	Word access: 6 cycles, byte access: 2 cycles
Transfer operation	Read: 1 cycle, write: 1 cycle	Read: 1 cycle, write: 1 cycle	Read: 1 cycle, write: 1 cycle
Interrupt vector	Word address: 1 cycle, byte access: 1 cycle	Word address: 1 cycle, byte access: 1 cycle	Word address: 1 cycle, byte access: 1 cycle
Stack write	Word access: 6 cycles	Word access: 6 cycles	Word access: 6 cycles

\*For compensation values (b) and (c) see Appendix B.4b.

Compensation values for transfer operations should be taken into account.

**Table B.11d Extended Intelligent I/O Service Execution Time at End of Count (when ISCS SE bit is '1')**

Buffer address pointer	Fixed	Updated	Updated
I/O address pointer	Fixed	Fixed	Updated
Cycle count BAP ⇒ IOA IOA ⇒ BAP	40 41	40 41	43 44
Compensation value	$7 \times (b) + 13 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)	$7 \times (b) + 13 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)	$7 \times (b) + 14 \times (c)$ + transfer operation (read: 1 cycle, write: 1 cycle)
Bus operation Internal register access	Word access: 1 cycle, byte access: 4 cycles	Word access: 1 cycle, byte access: 4 cycles	Word access: 1 cycle, byte access: 4 cycles
ISD access	Word access: 5 cycles, byte access: 2 cycles	Word access: 5 cycles, byte access: 2 cycles	Word access: 6 cycles, byte access: 2 cycles
Transfer operation	Read: 1 cycle, write: 1 cycle	Read: 1 cycle, write: 1 cycle	Read: 1 cycle, write: 1 cycle
Interrupt vector	Word address: 1 cycle, byte access: 1 cycle	Word address: 1 cycle, byte access: 1 cycle	Word address: 1 cycle, byte access: 1 cycle
Stack write	Word access: 6 cycles	Word access: 6 cycles	Word access: 6 cycles

\*For compensation values (b) and (c) see Appendix B.4b.

Compensation values for transfer operations should be taken into account.



## **APPENDIX C: F<sup>2</sup>MC-16F Instruction Map**

---

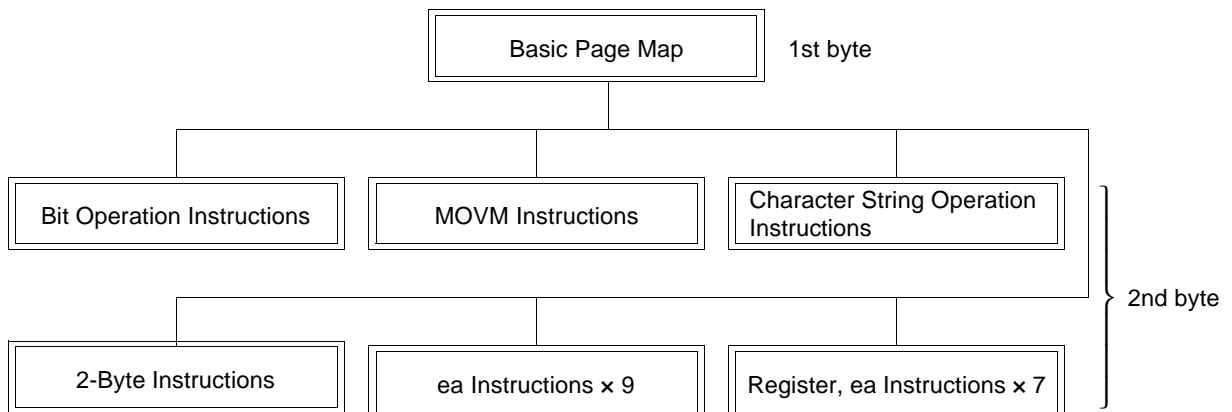
- C.1 Basic Map Structure
- C.2 Basic Page Maps
- C.3 Bit Operation Instruction Maps
- C.4 MOVM Instruction Maps
- C.5 Character String Operation Maps
- C.6 2-Byte Instruction Maps
- C.7 ea Instructions
- C.8 MOVEA R<sub>W</sub><sub>i</sub>, ea
- C.9 MOV R<sub>i</sub>, ea
- C.10 MOVW R<sub>W</sub><sub>i</sub>, ea
- C.11 MOV ea, R<sub>i</sub>
- C.12 MOVW ea, R<sub>W</sub><sub>i</sub>
- C.13 XCH R<sub>i</sub>, ea
- C.14 XCHW R<sub>W</sub><sub>i</sub>, ea

## C.1 Basic Map Structure

F<sup>2</sup>MC-16F instruction codes are composed of 1 or 2 bytes, and accordingly the instruction maps are constructed of several pages in 1- or 2-byte format.

### ■ Instruction Map Structure

Figure C.1a shows the structure of the instruction maps.



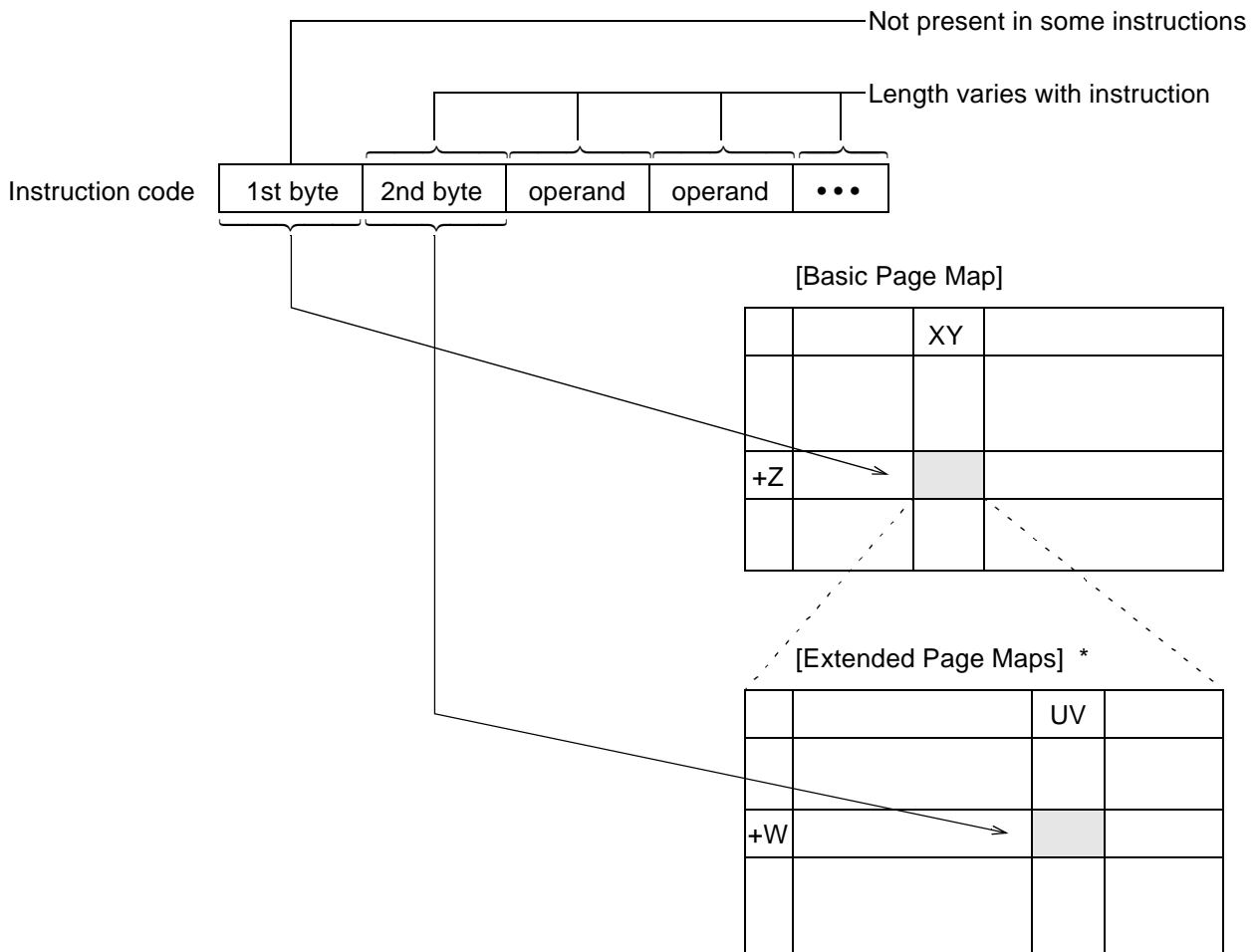
**Fig. C.1a F<sup>2</sup>MC-16F Instruction Map Structure**

For instructions with 1-byte instruction codes (NOP, etc.), the instruction code is described on the basic page map.

For instructions with 2-byte instruction codes (MOVS, etc.), refer to the basic page map, and find the name of the map on which the 2nd byte of the instruction code is described.

Figure C.1b shows the relation between the actual instruction codes and instruction maps.

## C.1 Basic Map Structure



\*: The extended page map is the collective term used for bit operation commands, character string operation commands, 2-byte instructions and ea instructions. Actually several pages are provided to list instructions in each group.

**Fig. C.1b Actual Instruction and Corresponding Map**

# C.2 Basic Page Map

Table C.2 Basic Page


	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 @RWi+d8	MOVW Wi+d8, 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 instructions	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	MULW A	NOTW A	PUSHW r1st	POPW r1st	Two-byte instructions	XCRW RWi, ea	↓	↓	↓	↓	↓	↓	↓	BHI rel

Note: a The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- #4: #imm4 (4-bit immediate data)
- #8: #imm8 (8-bit immediate data)
- #16: #imm16 (16-bit immediate data)
- #32: #imm32 (32-bit immediate data)
- d8: disp8 (8-bit displacement)

b The following instructions perform identical functions.

- LSLW: SHLW
- LSRW: SHRW
- SWAPW: XCHW A, T

 : Instructions with 2-byte instruction codes (continued on extended maps)

### C.3 Bit Operation Instruction Map

Table C.3 Bit Operation Instruction

	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, ad16:bp	MOVB dir: bp,A	MOVB ad16: bp, A	CLRB dir:bp	CLRB _ad16:bp	SETB dir:bp	SETB ad16:bp	BBC dir:bp,rel	BBC ad16:bp, rel	BBS dir:bp, rel	BBS ad16:bp, rel				SBBS ad16:bp, rel
+9																
+A																
+B																
+C																
+D																
+E																
+F																

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- ad16: addr16 (direct address specification)

# C.4 MOVW Instruction Map

Table C.4 MOVW Instruction Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVW @A, @RL0, #8	MOVW @A, @RW0+d8, #8	MOVW addr16, @RL0, #8	MOVW a, ddr16, @R, W0+d8, #8	MOVW @A, @RL0, #8	MOVW @A, @RW0+d8, #8	MOVW addr16, @RL0, #8	MOVW a, ddr16, @R, W0+d8, #8	MOVW @RL0, #8	MOVW @RW0+d8, #8	MOVW @RL0, a, ddr16, #8	MOVW @RW0+d8, a, ddr16, #8	MOVW @RL0, #8	MOVW @RW0+d8, #8	MOVW @RL0, a, ddr16, #8	MOVW @RW0+d8, a, ddr16, #8
+1	MOVW @A, @RL0, #8	MOVW @A, @RW1+d8, #8	MOVW addr16, @RL0, #8	MOVW a, ddr16, @R, W1+d8, #8	MOVW @A, @RL0, #8	MOVW @A, @RW1+d8, #8	MOVW addr16, @RL0, #8	MOVW a, ddr16, @R, W1+d8, #8	MOVW @RL0, #8	MOVW @RW1+d8, #8	MOVW @RL0, a, ddr16, #8	MOVW @RW1+d8, a, ddr16, #8	MOVW @RL0, #8	MOVW @RW1+d8, #8	MOVW @RL0, a, ddr16, #8	MOVW @RW1+d8, a, ddr16, #8
+2	MOVW @A, @RL1, #8	MOVW @A, @RW2+d8, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @R, W2+d8, #8	MOVW @A, @RL1, #8	MOVW @A, @RW2+d8, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @R, W2+d8, #8	MOVW @RL1, #8	MOVW @RW2+d8, #8	MOVW @RL1, a, ddr16, #8	MOVW @RW2+d8, a, ddr16, #8	MOVW @RL1, #8	MOVW @RW2+d8, #8	MOVW @RL1, a, ddr16, #8	MOVW @RW2+d8, a, ddr16, #8
+3	MOVW @A, @RL1, #8	MOVW @A, @RW3+d8, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @R, W3+d8, #8	MOVW @A, @RL1, #8	MOVW @A, @RW3+d8, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @R, W3+d8, #8	MOVW @RL1, #8	MOVW @RW3+d8, #8	MOVW @RL1, a, ddr16, #8	MOVW @RW3+d8, a, ddr16, #8	MOVW @RL1, #8	MOVW @RW3+d8, #8	MOVW @RL1, a, ddr16, #8	MOVW @RW3+d8, a, ddr16, #8
+4	MOVW @A, @RL2, #8	MOVW @A, @RW4+d8, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @R, W4+d8, #8	MOVW @A, @RL2, #8	MOVW @A, @RW4+d8, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @R, W4+d8, #8	MOVW @RL2, #8	MOVW @RW4+d8, #8	MOVW @RL2, a, ddr16, #8	MOVW @RW4+d8, a, ddr16, #8	MOVW @RL2, #8	MOVW @RW4+d8, #8	MOVW @RL2, a, ddr16, #8	MOVW @RW4+d8, a, ddr16, #8
+5	MOVW @A, @RL2, #8	MOVW @A, @RW5+d8, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @R, W5+d8, #8	MOVW @A, @RL2, #8	MOVW @A, @RW5+d8, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @R, W5+d8, #8	MOVW @RL2, #8	MOVW @RW5+d8, #8	MOVW @RL2, a, ddr16, #8	MOVW @RW5+d8, a, ddr16, #8	MOVW @RL2, #8	MOVW @RW5+d8, #8	MOVW @RL2, a, ddr16, #8	MOVW @RW5+d8, a, ddr16, #8
+6	MOVW @A, @RL3, #8	MOVW @A, @RW6+d8, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, @R, W6+d8, #8	MOVW @A, @RL3, #8	MOVW @A, @RW6+d8, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, @R, W6+d8, #8	MOVW @RL3, #8	MOVW @RW6+d8, #8	MOVW @RL3, a, ddr16, #8	MOVW @RW6+d8, a, ddr16, #8	MOVW @RL3, #8	MOVW @RW6+d8, #8	MOVW @RL3, a, ddr16, #8	MOVW @RW6+d8, a, ddr16, #8
+7	MOVW @A, @RL3, #8	MOVW @A, @RW7+d8, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, @R, W7+d8, #8	MOVW @A, @RL3, #8	MOVW @A, @RW7+d8, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, @R, W7+d8, #8	MOVW @RL3, #8	MOVW @RW7+d8, #8	MOVW @RL3, a, ddr16, #8	MOVW @RW7+d8, a, ddr16, #8	MOVW @RL3, #8	MOVW @RW7+d8, #8	MOVW @RL3, a, ddr16, #8	MOVW @RW7+d8, a, ddr16, #8
+8	MOVW @A, @RW0, #8	MOVW @A, @RW0+d16, #8	MOVW addr16, @RL03, #8	MOVW a, ddr16, @RW, W0+d16, #8	MOVW @A, @RW0, #8	MOVW @A, @RW0+d16, #8	MOVW addr16, @RL03, #8	MOVW a, ddr16, @RW, W0+d16, #8	MOVW @RW0, #8	MOVW @RW0+d16, #8	MOVW @RW0, a, ddr16, #8	MOVW @RW0+d16, a, ddr16, #8	MOVW @RW0, #8	MOVW @RW0+d16, #8	MOVW @RW0, a, ddr16, #8	MOVW @RW0+d16, a, ddr16, #8
+9	MOVW @A, @RW1, #8	MOVW @A, @RW1+d16, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @RW, W1+d16, #8	MOVW @A, @RW1, #8	MOVW @A, @RW1+d16, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @RW, W1+d16, #8	MOVW @RW1, #8	MOVW @RW1+d16, #8	MOVW @RW1, a, ddr16, #8	MOVW @RW1+d16, a, ddr16, #8	MOVW @RW1, #8	MOVW @RW1+d16, #8	MOVW @RW1, a, ddr16, #8	MOVW @RW1+d16, a, ddr16, #8
+A	MOVW @A, @RW2, #8	MOVW @A, @RW2+d16, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @RW, W2+d16, #8	MOVW @A, @RW2, #8	MOVW @A, @RW2+d16, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @RW, W2+d16, #8	MOVW @RW2, #8	MOVW @RW2+d16, #8	MOVW @RW2, a, ddr16, #8	MOVW @RW2+d16, a, ddr16, #8	MOVW @RW2, #8	MOVW @RW2+d16, #8	MOVW @RW2, a, ddr16, #8	MOVW @RW2+d16, a, ddr16, #8
+B	MOVW @A, @RW3, #8	MOVW @A, @RW3+d16, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, @RW, W3+d16, #8	MOVW @A, @RW3, #8	MOVW @A, @RW3+d16, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, @RW, W3+d16, #8	MOVW @RW3, #8	MOVW @RW3+d16, #8	MOVW @RW3, a, ddr16, #8	MOVW @RW3+d16, a, ddr16, #8	MOVW @RW3, #8	MOVW @RW3+d16, #8	MOVW @RW3, a, ddr16, #8	MOVW @RW3+d16, a, ddr16, #8
+C	MOVW @A, @RW0, #8	MOVW @A, @RW0+R, W7, #8	MOVW addr16, @RL0, #8	MOVW a, ddr16, @RW, W0+R, W7, #8	MOVW @A, @RW0, #8	MOVW @A, @RW0+R, W7, #8	MOVW addr16, @RL0, #8	MOVW a, ddr16, @RW, W0+R, W7, #8	MOVW @RW0, #8	MOVW @RW0+R, W7, #8	MOVW @RW0, a, ddr16, #8	MOVW @RW0+R, W7, a, ddr16, #8	MOVW @RW0, #8	MOVW @RW0+R, W7, #8	MOVW @RW0, a, ddr16, #8	MOVW @RW0+R, W7, a, ddr16, #8
+D	MOVW @A, @RW1, #8	MOVW @A, @RW1+R, W7, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @RW, W1+R, W7, #8	MOVW @A, @RW1, #8	MOVW @A, @RW1+R, W7, #8	MOVW addr16, @RL1, #8	MOVW a, ddr16, @RW, W1+R, W7, #8	MOVW @RW1, #8	MOVW @RW1+R, W7, #8	MOVW @RW1, a, ddr16, #8	MOVW @RW1+R, W7, a, ddr16, #8	MOVW @RW1, #8	MOVW @RW1+R, W7, #8	MOVW @RW1, a, ddr16, #8	MOVW @RW1+R, W7, a, ddr16, #8
+E	MOVW @A, @RW2, #8	MOVW @A, @RW2+P, C+d16, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @P, C+d16, #8	MOVW @A, @RW2, #8	MOVW @A, @RW2+P, C+d16, #8	MOVW addr16, @RL2, #8	MOVW a, ddr16, @P, C+d16, #8	MOVW @RW2, #8	MOVW @RW2+P, C+d16, #8	MOVW @RW2, a, ddr16, #8	MOVW @RW2+P, C+d16, a, ddr16, #8	MOVW @RW2, #8	MOVW @RW2+P, C+d16, #8	MOVW @RW2, a, ddr16, #8	MOVW @RW2+P, C+d16, a, ddr16, #8
+F	MOVW @A, @RW3, #8	MOVW @A, a, ddr16, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, #8	MOVW @A, @RW3, #8	MOVW @A, @RW3+d16, #8	MOVW addr16, @RL3, #8	MOVW a, ddr16, #8	MOVW @RW3, #8	MOVW @RW3+d16, #8	MOVW @RW3, a, ddr16, #8	MOVW @RW3+d16, a, ddr16, #8	MOVW @RW3, #8	MOVW @RW3+d16, #8	MOVW @RW3, a, ddr16, #8	MOVW @RW3+d16, a, ddr16, #8

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)
- #8 : #imm8 (8-bit immediate data)
- #16: #imm16 (16-bit immediate data)



# C.5 String Operation Instruction Map

Table C.5 String Operation Instruction

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOVSI PCB, PCB	MOVSD	MOVSWI	MOVSWD	MOV#1	MOV#2			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															

\*1: Assembler format: .....MOV# destination bank:addr16,source bank:addr16,#imm8

\*2: Assembler format: .....MOV#M destination bank:addr16,source bank:addr16,#imm8

Note: The following instructions perform identical functions.

MOVSI: MOVS

MOVSWI: MOVSW

SCEQI: SCEQ

SCWEQI: SCWEQ

FILSI: FILS

FILSWI: FILSW

## C.6 2-Byte Instruction Map

Table C.6 2-Byte Instruction

	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	MOV DTB , #imm8	MOV A, @SP+d8	MOVPL @A, RL0	MOVVP @A, R0							
+1	MOV A, ADB	MOV ADB, A	MOVX A, @RL0+d8	MOV @RL0+d8, A	MOV A, @RL0+d8	MOV ADB , #imm8	MOV A, @SP+d8	MOVPL @A, RL0	MOVVP @A, R1							
+2	MOV A, SSB	MOV SSB, A	MOVX A, @RL1+d8	MOV @RL1+d8, A	MOV A, @RL1+d8	MOV SSB , #imm8	MOVW A, @SP+d8	MOVPL @A, RL1	MOVVP @A, R2							
+3	MOV A, USB	MOV USB, A	MOVX A, @RL1+d8	MOV @RL1+d8, A	MOV A, @RL1+d8	MOV USB , #imm8	MOV A, @SP+d8	MOVPL @A, RL1	MOVVP @A, R3							
+4	MOV A, DPR	MOV DPR, A	MOVX A, @RL2+d8	MOV @RL2+d8, A	MOV A, @RL2+d8	MOV DPR , #imm8	MOV @SP+d8, A	MOVPL @A, RL2	MOVVP @A, R4							
+5	MOV A, @A	MOV @AL, AH	MOVX A, @RL2+d8	MOV @RL2+d8, A	MOV A, @RL2+d8	MOV SPB , #imm8	MOVW SP , #imm16	MOVPL @A, RL2	MOVVP @A, R5							
+6	MOV A, PCB	MOVX A, @A	MOVX A, @RL3+d8	MOV @RL3+d8, A	MOV A, @RL3+d8	MOV SPC U, #imm16	MOVW @SP+d8, A	MOVPL @A, RL3	MOVVP @A, R6							
+7	ROL A	ROR A	MOVX A, @RL3+d8	MOV @RL3+d8, A	MOV A, @RL3+d8	MOV SPC L, #imm16	MOV @SP+d8, A	MOVPL @A, RL3	MOVVP @A, R7							
+8	LSLW A, #imm8	LSLL A, #imm8	LSL A, #imm8	MOVW @RL0+d8, A	MOVW A, @RL0+d8	MOV A, @A	MOV A, addr24	MUL A	MOV @A, RW0							
+9	MOV A, SP8	MOV SPB, A	RETIQ	MOVW @RL0+d8, A	MOVW A, @RL0+d8	MOV A, @A	MOV A, addr24	MULW A	MOV @A, RW1							
+A	ASRW A, #imm8	ASRL A, #imm8	ASR A, #imm8	MOVW @RL1+d8, A	MOVW A, @RL1+d8	MOV A, @A	MOV A, addr24	DIV A	MOV @A, RW2							
+B	LSRW A, #imm8	LSRL A, #imm8	LSR A, #imm8	MOVW @RL1+d8, A	MOVW A, @RL1+d8	MOV A, @A	MOV A, addr24	ABS A	MOV @A, RW3							
+C	LSLW A, R0	LSLL A, R0	LSL A, R0	MOVW @RL2+d8, A	MOVW A, @RL2+d8	BTSCN A	MOV addr24, A	ABSL A	MOV @A, RW4							
+D	MOVW A, @A	MOVW @AL, AH	NRML A, R0	MOVW @RL2+d8, A	MOVW A, @RL2+d8	SETPC	CLRSPC		MOV @A, RW5							
+E	ASRW A, R0	ASRL A, R0	ASR A, R0	MOVW @RL3+d8, A	MOVW A, @RL3+d8	BTSCNS A	MOV addr24, A		MOV @A, RW6							
+F	LSRW A, R0	LSRL A, R0	LSR A, R0	MOVW @RL3+d8, A	MOVW A, @RL3+d8	BTSCND A	MOV addr24, A		MOV @A, RW7							

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8: disp8 (8-bit displacement)

## C.7 ea Instructions

Table C.7a ea Instructions [1st byte = 70H]

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDL A, RL0 @RW0+d8	ADDL A, @RW0+d8	SUBL A, RL0 @RW0+d8	SUBL A, @RW0+d8	CWBNE RW0, #16, rel @RW0+d8	CWBNE @RW0+d8, #16, rel	CMPL A, RL0 @RW0+d8	CMPL A, @RW0+d8	ANDL A, RL0 @RW0+d8	ANDL A, @RW0+d8	ORL A, RL0 @RW0+d8	ORL A, @RW0+d8	XORL A, RL0 @RW0+d8	XORL A, @RW0+d8	CBNE R0, #8, rel	CBNE @RW0+d8, #8, rel
+1	ADDL A, RL0 @RW1+d8	ADDL A, @RW1+d8	SUBL A, RL0 @RW1+d8	SUBL A, @RW1+d8	CWBNE RW1, #16, rel @RW1+d8	CWBNE @RW1+d8, #16, rel	CMPL A, RL0 @RW1+d8	CMPL A, @RW1+d8	ANDL A, RL0 @RW1+d8	ANDL A, @RW1+d8	ORL A, RL0 @RW1+d8	ORL A, @RW1+d8	XORL A, RL0 @RW1+d8	XORL A, @RW1+d8	CBNE R1, #8, rel	CBNE @RW1+d8, #8, rel
+2	ADDL A, RL1 @RW2+d8	ADDL A, @RW2+d8	SUBL A, RL1 @RW2+d8	SUBL A, @RW2+d8	CWBNE RW2, #16, rel @RW2+d8	CWBNE @RW2+d8, #16, rel	CMPL A, RL1 @RW2+d8	CMPL A, @RW2+d8	ANDL A, RL1 @RW2+d8	ANDL A, @RW2+d8	ORL A, RL1 @RW2+d8	ORL A, @RW2+d8	XORL A, RL1 @RW2+d8	XORL A, @RW2+d8	CBNE R2, #8, rel	CBNE @RW2+d8, #8, rel
+3	ADDL A, RL1 @RW3+d8	ADDL A, @RW3+d8	SUBL A, RL1 @RW3+d8	SUBL A, @RW3+d8	CWBNE RW3, #16, rel @RW3+d8	CWBNE @RW3+d8, #16, rel	CMPL A, RL1 @RW3+d8	CMPL A, @RW3+d8	ANDL A, RL1 @RW3+d8	ANDL A, @RW3+d8	ORL A, RL1 @RW3+d8	ORL A, @RW3+d8	XORL A, RL1 @RW3+d8	XORL A, @RW3+d8	CBNE R3, #8, rel	CBNE @RW3+d8, #8, rel
+4	ADDL A, RL2 @RW4+d8	ADDL A, @RW4+d8	SUBL A, RL2 @RW4+d8	SUBL A, @RW4+d8	CWBNE RW4, #16, rel @RW4+d8	CWBNE @RW4+d8, #16, rel	CMPL A, RL2 @RW4+d8	CMPL A, @RW4+d8	ANDL A, RL2 @RW4+d8	ANDL A, @RW4+d8	ORL A, RL2 @RW4+d8	ORL A, @RW4+d8	XORL A, RL2 @RW4+d8	XORL A, @RW4+d8	CBNE R4, #8, rel	CBNE @RW4+d8, #8, rel
+5	ADDL A, RL2 @RW5+d8	ADDL A, @RW5+d8	SUBL A, RL2 @RW5+d8	SUBL A, @RW5+d8	CWBNE RW5, #16, rel @RW5+d8	CWBNE @RW5+d8, #16, rel	CMPL A, RL2 @RW5+d8	CMPL A, @RW5+d8	ANDL A, RL2 @RW5+d8	ANDL A, @RW5+d8	ORL A, RL2 @RW5+d8	ORL A, @RW5+d8	XORL A, RL2 @RW5+d8	XORL A, @RW5+d8	CBNE R5, #8, rel	CBNE @RW5+d8, #8, rel
+6	ADDL A, RL3 @RW6+d8	ADDL A, @RW6+d8	SUBL A, RL3 @RW6+d8	SUBL A, @RW6+d8	CWBNE RW6, #16, rel @RW6+d8	CWBNE @RW6+d8, #16, rel	CMPL A, RL3 @RW6+d8	CMPL A, @RW6+d8	ANDL A, RL3 @RW6+d8	ANDL A, @RW6+d8	ORL A, RL3 @RW6+d8	ORL A, @RW6+d8	XORL A, RL3 @RW6+d8	XORL A, @RW6+d8	CBNE R6, #8, rel	CBNE @RW6+d8, #8, rel
+7	ADDL A, RL3 @RW7+d8	ADDL A, @RW7+d8	SUBL A, RL3 @RW7+d8	SUBL A, @RW7+d8	CWBNE RW7, #16, rel @RW7+d8	CWBNE @RW7+d8, #16, rel	CMPL A, RL3 @RW7+d8	CMPL A, @RW7+d8	ANDL A, RL3 @RW7+d8	ANDL A, @RW7+d8	ORL A, RL3 @RW7+d8	ORL A, @RW7+d8	XORL A, RL3 @RW7+d8	XORL A, @RW7+d8	CBNE R7, #8, rel	CBNE @RW7+d8, #8, rel
+8	ADDL A, @RW0 @RW0+d16	ADDL A, @RW0+d16	SUBL A, @RW0 @RW0+d16	SUBL A, @RW0+d16	CWBNE @RW0, #16, rel @RW0+d16	CWBNE @RW0+d16, #16, rel	CMPL A, @RW0 @RW0+d16	CMPL A, @RW0+d16	ANDL A, @RW0 @RW0+d16	ANDL A, @RW0+d16	ORL A, @RW0 @RW0+d16	ORL A, @RW0+d16	XORL A, @RW0 @RW0+d16	XORL A, @RW0+d16	CBNE #8, rel @RW0, #8, rel	CBNE @RW0+d16, #8, rel
+9	ADDL A, @RW1 @RW1+d16	ADDL A, @RW1+d16	SUBL A, @RW1 @RW1+d16	SUBL A, @RW1+d16	CWBNE @RW1, #16, rel @RW1+d16	CWBNE @RW1+d16, #16, rel	CMPL A, @RW1 @RW1+d16	CMPL A, @RW1+d16	ANDL A, @RW1 @RW1+d16	ANDL A, @RW1+d16	ORL A, @RW1 @RW1+d16	ORL A, @RW1+d16	XORL A, @RW1 @RW1+d16	XORL A, @RW1+d16	CBNE #8, rel @RW1, #8, rel	CBNE @RW1+d16, #8, rel
+A	ADDL A, @RW2 @RW2+d16	ADDL A, @RW2+d16	SUBL A, @RW2 @RW2+d16	SUBL A, @RW2+d16	CWBNE @RW2, #16, rel @RW2+d16	CWBNE @RW2+d16, #16, rel	CMPL A, @RW2 @RW2+d16	CMPL A, @RW2+d16	ANDL A, @RW2 @RW2+d16	ANDL A, @RW2+d16	ORL A, @RW2 @RW2+d16	ORL A, @RW2+d16	XORL A, @RW2 @RW2+d16	XORL A, @RW2+d16	CBNE #8, rel @RW2, #8, rel	CBNE @RW2+d16, #8, rel
+B	ADDL A, @RW3 @RW3+d16	ADDL A, @RW3+d16	SUBL A, @RW3 @RW3+d16	SUBL A, @RW3+d16	CWBNE @RW3, #16, rel @RW3+d16	CWBNE @RW3+d16, #16, rel	CMPL A, @RW3 @RW3+d16	CMPL A, @RW3+d16	ANDL A, @RW3 @RW3+d16	ANDL A, @RW3+d16	ORL A, @RW3 @RW3+d16	ORL A, @RW3+d16	XORL A, @RW3 @RW3+d16	XORL A, @RW3+d16	CBNE #8, rel @RW3, #8, rel	CBNE @RW3+d16, #8, rel
+C	ADDL A, @RW0+ @RW0+RW7	ADDL A, @RW0+RW7	SUBL A, @RW0+ @RW0+RW7	SUBL A, @RW0+RW7	CWBNE @RW0+ #16, rel @RW0+RW7	CWBNE @RW0+RW7, #16, rel	CMPL A, @RW0+ @RW0+RW7	CMPL A, @RW0+RW7	ANDL A, @RW0+ @RW0+RW7	ANDL A, @RW0+RW7	ORL A, @RW0+ @RW0+RW7	ORL A, @RW0+RW7	XORL A, @RW0+ @RW0+RW7	XORL A, @RW0+RW7	CBNE #8, rel @RW0+ #8, rel	CBNE @RW0+RW7, #8, rel
+D	ADDL A, @RW1+ @RW1+RW7	ADDL A, @RW1+RW7	SUBL A, @RW1+ @RW1+RW7	SUBL A, @RW1+RW7	CWBNE @RW1+ #16, rel @RW1+RW7	CWBNE @RW1+RW7, #16, rel	CMPL A, @RW1+ @RW1+RW7	CMPL A, @RW1+RW7	ANDL A, @RW1+ @RW1+RW7	ANDL A, @RW1+RW7	ORL A, @RW1+ @RW1+RW7	ORL A, @RW1+RW7	XORL A, @RW1+ @RW1+RW7	XORL A, @RW1+RW7	CBNE #8, rel @RW1+ #8, rel	CBNE @RW1+RW7, #8, rel
+E	ADDL A, @RW2+ @PC+d16	ADDL A, @PC+d16	SUBL A, @RW2+ @PC+d16	SUBL A, @PC+d16	CWBNE @RW2+ #16, rel @PC+d16	CWBNE @PC+d16, #16, rel	CMPL A, @RW2+ @PC+d16	CMPL A, @PC+d16	ANDL A, @RW2+ @PC+d16	ANDL A, @PC+d16	ORL A, @RW2+ @PC+d16	ORL A, @PC+d16	XORL A, @RW2+ @PC+d16	XORL A, @PC+d16	CBNE #8, rel @RW2+ #8, rel	CBNE @PC+d16, #8, rel
+F	ADDL A, @RW3+ addr16	ADDL A, addr16	SUBL A, @RW3+ addr16	SUBL A, addr16	CWBNE @RW3+ #16, rel addr16	CWBNE addr16, #16, rel	CMPL A, @RW3+ addr16	CMPL A, addr16	ANDL A, @RW3+ addr16	ANDL A, addr16	ORL A, @RW3+ addr16	ORL A, addr16	XORL A, @RW3+ addr16	XORL A, addr16	CBNE #8, rel @RW3+ #8, rel	CBNE addr16, #8, rel

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- #8 : #imm8 (8-bit immediate data)
- #16: #imm16 (16-bit immediate data)
- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

**Table C.7b ea Instructions [1st byte = 71H]**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMPP @RL0	JMPP @@RW0+d8	CALLP @RL0	CALLP @@RW0+d8	INCL RL0	INCL @@RW0+d8	DECL RL0	DECL @@RW0+d8	MOVL A, RL0	MOVL A, @RW0+d8	MOVL RL0, A	MOVL @R W0+d8, A	MOV R0, #8	MOV @R W0+d8, #8	MOVEA A, RW0	MOVEA A, @RW0+d8
+1	JMPP @RL0	JMPP @@RW1+d8	CALLP @RL0	CALLP @@RW1+d8	INCL RL0	INCL @@RW1+d8	DECL RL0	DECL @@RW1+d8	MOVL A, RL0	MOVL A, @RW1+d8	MOVL RL0, A	MOVL @R W1+d8, A	MOV R1, #8	MOV @R W1+d8, #8	MOVEA A, RW1	MOVEA A, @RW1+d8
+2	JMPP @RL1	JMPP @@RW2+d8	CALLP @RL1	CALLP @@RW2+d8	INCL RL1	INCL @@RW2+d8	DECL RL1	DECL @@RW2+d8	MOVL A, RL1	MOVL A, @RW2+d8	MOVL RL1, A	MOVL @R W2+d8, A	MOV R2, #8	MOV @R W2+d8, #8	MOVEA A, RW2	MOVEA A, @RW2+d8
+3	JMPP @RL1	JMPP @@RW3+d8	CALLP @RL1	CALLP @@RW3+d8	INCL RL1	INCL @@RW3+d8	DECL RL1	DECL @@RW3+d8	MOVL A, RL1	MOVL A, @RW3+d8	MOVL RL1, A	MOVL @R W3+d8, A	MOV R3, #8	MOV @R W3+d8, #8	MOVEA A, RW3	MOVEA A, @RW3+d8
+4	JMPP @RL2	JMPP @@RW4+d8	CALLP @RL2	CALLP @@RW4+d8	INCL RL2	INCL @@RW4+d8	DECL RL2	DECL @@RW4+d8	MOVL A, RL2	MOVL A, @RW4+d8	MOVL RL2, A	MOVL @R W4+d8, A	MOV R4, #8	MOV @R W4+d8, #8	MOVEA A, RW4	MOVEA A, @RW4+d8
+5	JMPP @RL2	JMPP @@RW5+d8	CALLP @RL2	CALLP @@RW5+d8	INCL RL2	INCL @@RW5+d8	DECL RL2	DECL @@RW5+d8	MOVL A, RL2	MOVL A, @RW5+d8	MOVL RL2, A	MOVL @R W5+d8, A	MOV R5, #8	MOV @R W5+d8, #8	MOVEA A, RW5	MOVEA A, @RW5+d8
+6	JMPP @RL3	JMPP @@RW6+d8	CALLP @RL3	CALLP @@RW6+d8	INCL RL3	INCL @@RW6+d8	DECL RL3	DECL @@RW6+d8	MOVL A, RL3	MOVL A, @RW6+d8	MOVL RL3, A	MOVL @R W6+d8, A	MOV R6, #8	MOV @R W6+d8, #8	MOVEA A, RW6	MOVEA A, @RW6+d8
+7	JMPP @RL3	JMPP @@RW7+d8	CALLP @RL3	CALLP @@RW7+d8	INCL RL3	INCL @@RW7+d8	DECL RL3	DECL @@RW7+d8	MOVL A, RL3	MOVL A, @RW7+d8	MOVL RL3, A	MOVL @R W7+d8, A	MOV R7, #8	MOV @R W7+d8, #8	MOVEA A, RW7	MOVEA A, @RW7+d8
+8	JMPP @@RW0	JMPP @ @RW0+d16	CALLP @RW0	CALLP @ @RW0+d16	INCL @RW0	INCL @@RW0+d16	DECL @RW0	DECL @@RW0+d16	MOVL A, @RW0	MOVL A, @RW0+d16	MOVL @RW0, A	MOVL @R W0+d16, A	MOV @RW0, #8	MOV @R W0+d16, #8	MOVEA A, @RW0	MOVEA A, @RW0+d16
+9	JMPP @@RW1	JMPP @ @RW1+d16	CALLP @RW1	CALLP @ @RW1+d16	INCL @RW1	INCL @@RW1+d16	DECL @RW1	DECL @@RW1+d16	MOVL A, @RW1	MOVL A, @RW1+d16	MOVL @RW1, A	MOVL @R W1+d16, A	MOV @RW1, #8	MOV @R W1+d16, #8	MOVEA A, @RW1	MOVEA A, @RW1+d16
+A	JMPP @@RW2	JMPP @ @RW2+d16	CALLP @RW2	CALLP @ @RW2+d16	INCL @RW2	INCL @@RW2+d16	DECL @RW2	DECL @@RW2+d16	MOVL A, @RW2	MOVL A, @RW2+d16	MOVL @RW2, A	MOVL @R W2+d16, A	MOV @RW2, #8	MOV @R W2+d16, #8	MOVEA A, @RW2	MOVEA A, @RW2+d16
+B	JMPP @@RW3	JMPP @ @RW3+d16	CALLP @RW3	CALLP @ @RW3+d16	INCL @RW3	INCL @@RW3+d16	DECL @RW3	DECL @@RW3+d16	MOVL A, @RW3	MOVL A, @RW3+d16	MOVL @RW3, A	MOVL @R W3+d16, A	MOV @RW3, #8	MOV @R W3+d16, #8	MOVEA A, @RW3	MOVEA A, @RW3+d16
+C	JMPP @RW0+	JMPP @ @RW0+RW7	CALLP @RW0+	CALLP @ @RW0+RW7	INCL @RW0+	INCL @RW0+RW7	DECL @RW0+	DECL @RW0+RW7	MOVL A, @RW0+	MOVL A, @RW0+RW7	MOVL @RW0+, A	MOVL @R W0+RW7, A	MOV @RW0+, #8	MOV @R W0+RW7, #8	MOVEA A, @RW0+	MOVEA A, @RW0+RW7
+D	JMPP @RW1+	JMPP @ @RW1+RW7	CALLP @RW1+	CALLP @ @RW1+RW7	INCL @RW1+	INCL @RW1+RW7	DECL @RW1+	DECL @RW1+RW7	MOVL A, @RW1+	MOVL A, @RW1+RW7	MOVL @RW1+, A	MOVL @R W1+RW7, A	MOV @RW1+, #8	MOV @R W1+RW7, #8	MOVEA A, @RW1+	MOVEA A, @RW1+RW7
+E	JMPP @RW2+	JMPP @ @PC+d16	CALLP @RW2+	CALLP @ @PC+d16	INCL @RW2+	INCL @PC+d16	DECL @RW2+	DECL @PC+d16	MOVL A, @RW2+	MOVL A, @PC+d16	MOVL @RW2+, A	MOVL @P C+d16, A	MOV @RW2+, #8	MOV @P C+d16, #8	MOVEA A, @RW2+	MOVEA A, @PC+d16
+F	JMPP @RW3+	JMPP @addr16	CALLP @RW3+	CALLP @addr16	INCL @RW3+	INCL addr16	DECL @RW3+	DECL addr16	MOVL A, @RW3+	MOVL A, addr16	MOVL @RW3+, A	MOVL addr16, A	MOV @RW3+, #8	MOV addr16, #8	MOVEA A, @RW3+	MOVEA A, addr16

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

#8 : #imm8 (8-bit immediate data)

d8 : disp8 (8-bit displacement)

d16: disp16 (16-bit displacement)

**Table C.7c ea Instructions [1st byte = 72H]**

	00	01	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ROL R0	ROL @RW0+d8	ROR R0	ROR @RW0+d8	INC R0	INC @RW0+d8	DEC R0	DEC @RW0+d8	MOV A, R0	MOV @RW0+d8	MOV A, R0	MOV @RW0+d8	MOV @R W0+d8, A	MOVX A, R0	MOVX @RW0+d8	XCH A, R0	XCH @RW0+d8
+1	ROL R1	ROL @RW1+d8	ROR R1	ROR @RW1+d8	INC R1	INC @RW1+d8	DEC R1	DEC @RW1+d8	MOV A, R1	MOV @RW1+d8	MOV A, R1	MOV @RW1+d8	MOV @R W1+d8, A	MOVX A, R1	MOVX @RW1+d8	XCH A, R1	XCH @RW1+d8
+2	ROL R2	ROL @RW2+d8	ROR R2	ROR @RW2+d8	INC R2	INC @RW2+d8	DEC R2	DEC @RW2+d8	MOV A, R2	MOV @RW2+d8	MOV A, R2	MOV @RW2+d8	MOV @R W2+d8, A	MOVX A, R2	MOVX @RW2+d8	XCH A, R2	XCH @RW2+d8
+3	ROL R3	ROL @RW3+d8	ROR R3	ROR @RW3+d8	INC R3	INC @RW3+d8	DEC R3	DEC @RW3+d8	MOV A, R3	MOV @RW3+d8	MOV A, R3	MOV @RW3+d8	MOV @R W3+d8, A	MOVX A, R3	MOVX @RW3+d8	XCH A, R3	XCH @RW3+d8
+4	ROL R4	ROL @RW4+d8	ROR R4	ROR @RW4+d8	INC R4	INC @RW4+d8	DEC R4	DEC @RW4+d8	MOV A, R4	MOV @RW4+d8	MOV A, R4	MOV @RW4+d8	MOV @R W4+d8, A	MOVX A, R4	MOVX @RW4+d8	XCH A, R4	XCH @RW4+d8
+5	ROL R5	ROL @RW5+d8	ROR R5	ROR @RW5+d8	INC R5	INC @RW5+d8	DEC R5	DEC @RW5+d8	MOV A, R5	MOV @RW5+d8	MOV A, R5	MOV @RW5+d8	MOV @R W5+d8, A	MOVX A, R5	MOVX @RW5+d8	XCH A, R5	XCH @RW5+d8
+6	ROL R6	ROL @RW6+d8	ROR R6	ROR @RW6+d8	INC R6	INC @RW6+d8	DEC R6	DEC @RW6+d8	MOV A, R6	MOV @RW6+d8	MOV A, R6	MOV @RW6+d8	MOV @R W6+d8, A	MOVX A, R6	MOVX @RW6+d8	XCH A, R6	XCH @RW6+d8
+7	ROL R7	ROL @RW7+d8	ROR R7	ROR @RW7+d8	INC R7	INC @RW7+d8	DEC R7	DEC @RW7+d8	MOV A, R7	MOV @RW7+d8	MOV A, R7	MOV @RW7+d8	MOV @R W7+d8, A	MOVX A, R7	MOVX @RW7+d8	XCH A, R7	XCH @RW7+d8
+8	ROL @RW0	ROL @RW0+d16	ROR @RW0	ROR @RW0+d16	INC @RW0	INC @RW0+d16	DEC @RW0	DEC @RW0+d16	MOV @RW0, A	MOV @RW0+d16	MOV @RW0, A	MOV @RW0+d16	MOV @R W0+d16, A	MOVX @RW0, A	MOVX @RW0+d16	XCH @RW0, A	XCH @RW0+d16
+9	ROL @RW1	ROL @RW1+d16	ROR @RW1	ROR @RW1+d16	INC @RW1	INC @RW1+d16	DEC @RW1	DEC @RW1+d16	MOV @RW1, A	MOV @RW1+d16	MOV @RW1, A	MOV @RW1+d16	MOV @R W1+d16, A	MOVX @RW1, A	MOVX @RW1+d16	XCH @RW1, A	XCH @RW1+d16
+A	ROL @RW2	ROL @RW2+d16	ROR @RW2	ROR @RW2+d16	INC @RW2	INC @RW2+d16	DEC @RW2	DEC @RW2+d16	MOV @RW2, A	MOV @RW2+d16	MOV @RW2, A	MOV @RW2+d16	MOV @R W2+d16, A	MOVX @RW2, A	MOVX @RW2+d16	XCH @RW2, A	XCH @RW2+d16
+B	ROL @RW3	ROL @RW3+d16	ROR @RW3	ROR @RW3+d16	INC @RW3	INC @RW3+d16	DEC @RW3	DEC @RW3+d16	MOV @RW3, A	MOV @RW3+d16	MOV @RW3, A	MOV @RW3+d16	MOV @R W3+d16, A	MOVX @RW3, A	MOVX @RW3+d16	XCH @RW3, A	XCH @RW3+d16
+C	ROL @RW0+	ROL @RW0+RW7	ROR @RW0+	ROR @RW0+RW7	INC @RW0+	INC @RW0+RW7	DEC @RW0+	DEC @RW0+RW7	MOV @RW0+, A	MOV @RW0+RW7	MOV @RW0+, A	MOV @RW0+RW7	MOV @R W0+RW7, A	MOVX @RW0+, A	MOVX @RW0+RW7	XCH @RW0+, A	XCH @RW0+RW7
+D	ROL @RW1+	ROL @RW1+RW7	ROR @RW1+	ROR @RW1+RW7	INC @RW1+	INC @RW1+RW7	DEC @RW1+	DEC @RW1+RW7	MOV @RW1+, A	MOV @RW1+RW7	MOV @RW1+, A	MOV @RW1+RW7	MOV @R W1+RW7, A	MOVX @RW1+, A	MOVX @RW1+RW7	XCH @RW1+, A	XCH @RW1+RW7
+E	ROL @RW2+	ROL @PC+d16	ROR @RW2+	ROR @PC+d16	INC @RW2+	INC @PC+d16	DEC @RW2+	DEC @PC+d16	MOV @RW2+, A	MOV @PC+d16	MOV @RW2+, A	MOV @PC+d16	MOV @R C+d16, A	MOVX @RW2+, A	MOVX @PC+d16	XCH @RW2+, A	XCH @PC+d16
+F	ROL @RW3+	ROL @RW3+ addr16	ROR @RW3+	ROR @RW3+ addr16	INC @RW3+	INC @RW3+ addr16	DEC @RW3+	DEC @RW3+ addr16	MOV @RW3+, A	MOV @RW3+ addr16	MOV @RW3+, A	MOV @RW3+ addr16	MOV @R W3+ addr16, A	MOVX @RW3+, A	MOVX @RW3+ addr16	XCH @RW3+, A	XCH @RW3+ addr16

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

d8 : disp8 (8-bit displacement)

d16: disp16 (16-bit displacement)

**Table C.7d ea Instructions [1st byte = 73H]**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	JMP @RW0	JMP @RW0+d8	CALL @RW0	CALL @RW0+d8	INCW RW0	INCW @RW0+d8	DECW RW0	DECW @RW0+d8	MOVW A, RW0	MOVW A, @RW0+d8	MOVW RW0, A	MOVW @RW0+d8, A	MOVW RW0, #16	MOVW @RW0+d8, #16	XCHW A, RW0	XCHW A, @RW0+d8
+1	JMP @RW1	JMP @RW1+d8	CALL @RW1	CALL @RW1+d8	INCW RW1	INCW @RW1+d8	DECW RW1	DECW @RW1+d8	MOVW A, RW1	MOVW A, @RW1+d8	MOVW RW1, A	MOVW @RW1+d8, A	MOVW RW1, #16	MOVW @RW1+d8, #16	XCHW A, RW1	XCHW A, @RW1+d8
+2	JMP @RW2	JMP @RW2+d8	CALL @RW2	CALL @RW2+d8	INCW RW2	INCW @RW2+d8	DECW RW2	DECW @RW2+d8	MOVW A, RW2	MOVW A, @RW2+d8	MOVW RW2, A	MOVW @RW2+d8, A	MOVW RW2, #16	MOVW @RW2+d8, #16	XCHW A, RW2	XCHW A, @RW2+d8
+3	JMP @RW3	JMP @RW3+d8	CALL @RW3	CALL @RW3+d8	INCW RW3	INCW @RW3+d8	DECW RW3	DECW @RW3+d8	MOVW A, RW3	MOVW A, @RW3+d8	MOVW RW3, A	MOVW @RW3+d8, A	MOVW RW3, #16	MOVW @RW3+d8, #16	XCHW A, RW3	XCHW A, @RW3+d8
+4	JMP @RW4	JMP @RW4+d8	CALL @RW4	CALL @RW4+d8	INCW RW4	INCW @RW4+d8	DECW RW4	DECW @RW4+d8	MOVW A, RW4	MOVW A, @RW4+d8	MOVW RW4, A	MOVW @RW4+d8, A	MOVW RW4, #16	MOVW @RW4+d8, #16	XCHW A, RW4	XCHW A, @RW4+d8
+5	JMP @RW5	JMP @RW5+d8	CALL @RW5	CALL @RW5+d8	INCW RW5	INCW @RW5+d8	DECW RW5	DECW @RW5+d8	MOVW A, RW5	MOVW A, @RW5+d8	MOVW RW5, A	MOVW @RW5+d8, A	MOVW RW5, #16	MOVW @RW5+d8, #16	XCHW A, RW5	XCHW A, @RW5+d8
+6	JMP @RW6	JMP @RW6+d8	CALL @RW6	CALL @RW6+d8	INCW RW6	INCW @RW6+d8	DECW RW6	DECW @RW6+d8	MOVW A, RW6	MOVW A, @RW6+d8	MOVW RW6, A	MOVW @RW6+d8, A	MOVW RW6, #16	MOVW @RW6+d8, #16	XCHW A, RW6	XCHW A, @RW6+d8
+7	JMP @RW7	JMP @RW7+d8	CALL @RW7	CALL @RW7+d8	INCW RW7	INCW @RW7+d8	DECW RW7	DECW @RW7+d8	MOVW A, RW7	MOVW A, @RW7+d8	MOVW RW7, A	MOVW @RW7+d8, A	MOVW RW7, #16	MOVW @RW7+d8, #16	XCHW A, RW7	XCHW A, @RW7+d8
+8	JMP @RW0	JMP @RW0+d16	CALL @RW0	CALL @RW0+d16	INCW @RW0	INCW @RW0+d16	DECW @RW0	DECW @RW0+d16	MOVW A, @RW0	MOVW A, @RW0+d16	MOVW @RW0, A	MOVW @RW0+d16, A	MOVW @RW0, #16	MOVW @RW0+d16, #16	XCHW A, @RW0	XCHW A, @RW0+d16
+9	JMP @RW1	JMP @RW1+d16	CALL @RW1	CALL @RW1+d16	INCW @RW1	INCW @RW1+d16	DECW @RW1	DECW @RW1+d16	MOVW A, @RW1	MOVW A, @RW1+d16	MOVW @RW1, A	MOVW @RW1+d16, A	MOVW @RW1, #16	MOVW @RW1+d16, #16	XCHW A, @RW1	XCHW A, @RW1+d16
+A	JMP @RW2	JMP @RW2+d16	CALL @RW2	CALL @RW2+d16	INCW @RW2	INCW @RW2+d16	DECW @RW2	DECW @RW2+d16	MOVW A, @RW2	MOVW A, @RW2+d16	MOVW @RW2, A	MOVW @RW2+d16, A	MOVW @RW2, #16	MOVW @RW2+d16, #16	XCHW A, @RW2	XCHW A, @RW2+d16
+B	JMP @RW3	JMP @RW3+d16	CALL @RW3	CALL @RW3+d16	INCW @RW3	INCW @RW3+d16	DECW @RW3	DECW @RW3+d16	MOVW A, @RW3	MOVW A, @RW3+d16	MOVW @RW3, A	MOVW @RW3+d16, A	MOVW @RW3, #16	MOVW @RW3+d16, #16	XCHW A, @RW3	XCHW A, @RW3+d16
+C	JMP @RW0+	JMP @RW0+RW7	CALL @RW0+	CALL @RW0+RW7	INCW @RW0+	INCW @RW0+RW7	DECW @RW0+	DECW @RW0+RW7	MOVW A, @RW0+	MOVW A, @RW0+RW7	MOVW @RW0+, A	MOVW @RW0+RW7, A	MOVW @RW0+, #16	MOVW @RW0+RW7, #16	XCHW A, @RW0+	XCHW A, @RW0+RW7
+D	JMP @RW1+	JMP @RW1+RW7	CALL @RW1+	CALL @RW1+RW7	INCW @RW1+	INCW @RW1+RW7	DECW @RW1+	DECW @RW1+RW7	MOVW A, @RW1+	MOVW A, @RW1+RW7	MOVW @RW1+, A	MOVW @RW1+RW7, A	MOVW @RW1+, #16	MOVW @RW1+RW7, #16	XCHW A, @RW1+	XCHW A, @RW1+RW7
+E	JMP @RW2+	JMP @PC+d16	CALL @RW2+	CALL @PC+d16	INCW @RW2+	INCW @PC+d16	DECW @RW2+	DECW @PC+d16	MOVW A, @RW2+	MOVW A, @PC+d16	MOVW @RW2+, A	MOVW @PC+d16, A	MOVW @RW2+, #16	MOVW @PC+d16, #16	XCHW A, @RW2+	XCHW A, @PC+d16
+F	JMP @RW3+	JMP @addr16	CALL @RW3+	CALL @addr16	INCW @RW3+	INCW @addr16	DECW @RW3+	DEC @addr16	MOVW A, @RW3+	MOVW @addr16	MOVW @RW3+, A	MOVW @addr16, A	MOVW @RW3+, #16	MOVW @addr16, #16	XCHW A, @RW3+	XCHW A, @addr16

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- #16: #imm16 (16-bit immediate data)
- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

**Table C.7e ea Instructions [1st byte = 74H]**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD A, R0 @RW0+d8	SUB A, R0 @RW0+d8	SUB A, R0 @RW0+d8	SUB A, R0 @RW0+d8	ADDC A, R0 @RW0+d8	ADDC A, R0 @RW0+d8	CMP A, R0 @RW0+d8	CMP A, R0 @RW0+d8	AND A, R0 @RW0+d8	AND A, R0 @RW0+d8	OR A, R0 @RW0+d8	OR A, R0 @RW0+d8	XOR A, R0 @RW0+d8	XOR A, R0 @RW0+d8	DBNZ R0, r	DBNZ @RW0+d8, r
+1	ADD A, R1 @RW1+d8	SUB A, R1 @RW1+d8	SUB A, R1 @RW1+d8	SUB A, R1 @RW1+d8	ADDC A, R1 @RW1+d8	ADDC A, R1 @RW1+d8	CMP A, R1 @RW1+d8	CMP A, R1 @RW1+d8	AND A, R1 @RW1+d8	AND A, R1 @RW1+d8	OR A, R1 @RW1+d8	OR A, R1 @RW1+d8	XOR A, R1 @RW1+d8	XOR A, R1 @RW1+d8	DBNZ R1, r	DBNZ @RW1+d8, r
+2	ADD A, R2 @RW2+d8	SUB A, R2 @RW2+d8	SUB A, R2 @RW2+d8	SUB A, R2 @RW2+d8	ADDC A, R2 @RW2+d8	ADDC A, R2 @RW2+d8	CMP A, R2 @RW2+d8	CMP A, R2 @RW2+d8	AND A, R2 @RW2+d8	AND A, R2 @RW2+d8	OR A, R2 @RW2+d8	OR A, R2 @RW2+d8	XOR A, R2 @RW2+d8	XOR A, R2 @RW2+d8	DBNZ R2, r	DBNZ @RW2+d8, r
+3	ADD A, R3 @RW3+d8	SUB A, R3 @RW3+d8	SUB A, R3 @RW3+d8	SUB A, R3 @RW3+d8	ADDC A, R3 @RW3+d8	ADDC A, R3 @RW3+d8	CMP A, R3 @RW3+d8	CMP A, R3 @RW3+d8	AND A, R3 @RW3+d8	AND A, R3 @RW3+d8	OR A, R3 @RW3+d8	OR A, R3 @RW3+d8	XOR A, R3 @RW3+d8	XOR A, R3 @RW3+d8	DBNZ R3, r	DBNZ @RW3+d8, r
+4	ADD A, R4 @RW4+d8	SUB A, R4 @RW4+d8	SUB A, R4 @RW4+d8	SUB A, R4 @RW4+d8	ADDC A, R4 @RW4+d8	ADDC A, R4 @RW4+d8	CMP A, R4 @RW4+d8	CMP A, R4 @RW4+d8	AND A, R4 @RW4+d8	AND A, R4 @RW4+d8	OR A, R4 @RW4+d8	OR A, R4 @RW4+d8	XOR A, R4 @RW4+d8	XOR A, R4 @RW4+d8	DBNZ R4, r	DBNZ @RW4+d8, r
+5	ADD A, R5 @RW5+d8	SUB A, R5 @RW5+d8	SUB A, R5 @RW5+d8	SUB A, R5 @RW5+d8	ADDC A, R5 @RW5+d8	ADDC A, R5 @RW5+d8	CMP A, R5 @RW5+d8	CMP A, R5 @RW5+d8	AND A, R5 @RW5+d8	AND A, R5 @RW5+d8	OR A, R5 @RW5+d8	OR A, R5 @RW5+d8	XOR A, R5 @RW5+d8	XOR A, R5 @RW5+d8	DBNZ R5, r	DBNZ @RW5+d8, r
+6	ADD A, R6 @RW6+d8	SUB A, R6 @RW6+d8	SUB A, R6 @RW6+d8	SUB A, R6 @RW6+d8	ADDC A, R6 @RW6+d8	ADDC A, R6 @RW6+d8	CMP A, R6 @RW6+d8	CMP A, R6 @RW6+d8	AND A, R6 @RW6+d8	AND A, R6 @RW6+d8	OR A, R6 @RW6+d8	OR A, R6 @RW6+d8	XOR A, R6 @RW6+d8	XOR A, R6 @RW6+d8	DBNZ R6, r	DBNZ @RW6+d8, r
+7	ADD A, R7 @RW7+d8	SUB A, R7 @RW7+d8	SUB A, R7 @RW7+d8	SUB A, R7 @RW7+d8	ADDC A, R7 @RW7+d8	ADDC A, R7 @RW7+d8	CMP A, R7 @RW7+d8	CMP A, R7 @RW7+d8	AND A, R7 @RW7+d8	AND A, R7 @RW7+d8	OR A, R7 @RW7+d8	OR A, R7 @RW7+d8	XOR A, R7 @RW7+d8	XOR A, R7 @RW7+d8	DBNZ R7, r	DBNZ @RW7+d8, r
+8	ADD A, @RW0 @RW0+d16	SUB A, @RW0 @RW0+d16	SUB A, @RW0 @RW0+d16	SUB A, @RW0 @RW0+d16	ADDC A, @RW0 @RW0+d16	ADDC A, @RW0 @RW0+d16	CMP A, @RW0 @RW0+d16	CMP A, @RW0 @RW0+d16	AND A, @RW0 @RW0+d16	AND A, @RW0 @RW0+d16	OR A, @RW0 @RW0+d16	OR A, @RW0 @RW0+d16	XOR A, @RW0 @RW0+d16	XOR A, @RW0 @RW0+d16	DBNZ @RW0, r	DBNZ @RW0+d16, r
+9	ADD A, @RW1 @RW1+d16	SUB A, @RW1 @RW1+d16	SUB A, @RW1 @RW1+d16	SUB A, @RW1 @RW1+d16	ADDC A, @RW1 @RW1+d16	ADDC A, @RW1 @RW1+d16	CMP A, @RW1 @RW1+d16	CMP A, @RW1 @RW1+d16	AND A, @RW1 @RW1+d16	AND A, @RW1 @RW1+d16	OR A, @RW1 @RW1+d16	OR A, @RW1 @RW1+d16	XOR A, @RW1 @RW1+d16	XOR A, @RW1 @RW1+d16	DBNZ @RW1, r	DBNZ @RW1+d16, r
+A	ADD A, @RW2 @RW2+d16	SUB A, @RW2 @RW2+d16	SUB A, @RW2 @RW2+d16	SUB A, @RW2 @RW2+d16	ADDC A, @RW2 @RW2+d16	ADDC A, @RW2 @RW2+d16	CMP A, @RW2 @RW2+d16	CMP A, @RW2 @RW2+d16	AND A, @RW2 @RW2+d16	AND A, @RW2 @RW2+d16	OR A, @RW2 @RW2+d16	OR A, @RW2 @RW2+d16	XOR A, @RW2 @RW2+d16	XOR A, @RW2 @RW2+d16	DBNZ @RW2, r	DBNZ @RW2+d16, r
+B	ADD A, @RW3 @RW3+d16	SUB A, @RW3 @RW3+d16	SUB A, @RW3 @RW3+d16	SUB A, @RW3 @RW3+d16	ADDC A, @RW3 @RW3+d16	ADDC A, @RW3 @RW3+d16	CMP A, @RW3 @RW3+d16	CMP A, @RW3 @RW3+d16	AND A, @RW3 @RW3+d16	AND A, @RW3 @RW3+d16	OR A, @RW3 @RW3+d16	OR A, @RW3 @RW3+d16	XOR A, @RW3 @RW3+d16	XOR A, @RW3 @RW3+d16	DBNZ @RW3, r	DBNZ @RW3+d16, r
+C	ADD A, @RW0+ @RW0+RW7	SUB A, @RW0+ @RW0+RW7	SUB A, @RW0+ @RW0+RW7	SUB A, @RW0+ @RW0+RW7	ADDC A, @RW0+ @RW0+RW7	ADDC A, @RW0+ @RW0+RW7	CMP A, @RW0+ @RW0+RW7	CMP A, @RW0+ @RW0+RW7	AND A, @RW0+ @RW0+RW7	AND A, @RW0+ @RW0+RW7	OR A, @RW0+ @RW0+RW7	OR A, @RW0+ @RW0+RW7	XOR A, @RW0+ @RW0+RW7	XOR A, @RW0+ @RW0+RW7	DBNZ @RW0+, r	DBNZ @RW0+RW7, r
+D	ADD A, @RW1+ @RW1+RW7	SUB A, @RW1+ @RW1+RW7	SUB A, @RW1+ @RW1+RW7	SUB A, @RW1+ @RW1+RW7	ADDC A, @RW1+ @RW1+RW7	ADDC A, @RW1+ @RW1+RW7	CMP A, @RW1+ @RW1+RW7	CMP A, @RW1+ @RW1+RW7	AND A, @RW1+ @RW1+RW7	AND A, @RW1+ @RW1+RW7	OR A, @RW1+ @RW1+RW7	OR A, @RW1+ @RW1+RW7	XOR A, @RW1+ @RW1+RW7	XOR A, @RW1+ @RW1+RW7	DBNZ @RW1+, r	DBNZ @RW1+RW7, r
+E	ADD A, @RW2+ @PC+d16	SUB A, @RW2+ @PC+d16	SUB A, @RW2+ @PC+d16	SUB A, @RW2+ @PC+d16	ADDC A, @RW2+ @PC+d16	ADDC A, @RW2+ @PC+d16	CMP A, @RW2+ @PC+d16	CMP A, @RW2+ @PC+d16	AND A, @RW2+ @PC+d16	AND A, @RW2+ @PC+d16	OR A, @RW2+ @PC+d16	OR A, @RW2+ @PC+d16	XOR A, @RW2+ @PC+d16	XOR A, @RW2+ @PC+d16	DBNZ @RW2+, r	DBNZ @PC+d16, r
+F	ADD A, @RW3+ addr16	SUB A, @RW3+ addr16	SUB A, @RW3+ addr16	SUB A, @RW3+ addr16	ADDC A, @RW3+ addr16	ADDC A, @RW3+ addr16	CMP A, @RW3+ addr16	CMP A, @RW3+ addr16	AND A, @RW3+ addr16	AND A, @RW3+ addr16	OR A, @RW3+ addr16	OR A, @RW3+ addr16	XOR A, @RW3+ addr16	XOR A, @RW3+ addr16	DBNZ @RW3+, r	DBNZ @addr16, r

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

**Table C.7f ea Instructions [1st byte = 75H]**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADD R0, A	ADD @R W0+d8, A	SUB R0, A	SUB @R W0+d8, A	SUBC A, R0	SUBC @RW0+d8	NEG R0	NEG @RW0+d8	AND R0, A	AND @R W0+d8, A	OR R0, A	OR @RW0+d8, A	XOR R0, A	XOR @R W0+d8, A	NOT R0	NOT @RW0+d8
+1	ADD R1, A	ADD @R W1+d8, A	SUB R1, A	SUB @R W1+d8, A	SUBC A, R1	SUBC @RW1+d8	NEG R1	NEG @RW1+d8	AND R1, A	AND @R W1+d8, A	OR R1, A	OR @RW1+d8, A	XOR R1, A	XOR @R W1+d8, A	NOT R1	NOT @RW1+d8
+2	ADD R2, A	ADD @R W2+d8, A	SUB R2, A	SUB @R W2+d8, A	SUBC A, R2	SUBC @RW2+d8	NEG R2	NEG @RW2+d8	AND R2, A	AND @R W2+d8, A	OR R2, A	OR @RW2+d8, A	XOR R2, A	XOR @R W2+d8, A	NOT R2	NOT @RW2+d8
+3	ADD R3, A	ADD @R W3+d8, A	SUB R3, A	SUB @R W3+d8, A	SUBC A, R3	SUBC @RW3+d8	NEG R3	NEG @RW3+d8	AND R3, A	AND @R W3+d8, A	OR R3, A	OR @RW3+d8, A	XOR R3, A	XOR @R W3+d8, A	NOT R3	NOT @RW3+d8
+4	ADD R4, A	ADD @R W4+d8, A	SUB R4, A	SUB @R W4+d8, A	SUBC A, R4	SUBC @RW4+d8	NEG R4	NEG @RW4+d8	AND R4, A	AND @R W4+d8, A	OR R4, A	OR @RW4+d8, A	XOR R4, A	XOR @R W4+d8, A	NOT R4	NOT @RW4+d8
+5	ADD R5, A	ADD @R W5+d8, A	SUB R5, A	SUB @R W5+d8, A	SUBC A, R5	SUBC @RW5+d8	NEG R5	NEG @RW5+d8	AND R5, A	AND @R W5+d8, A	OR R5, A	OR @RW5+d8, A	XOR R5, A	XOR @R W5+d8, A	NOT R5	NOT @RW5+d8
+6	ADD R6, A	ADD @R W6+d8, A	SUB R6, A	SUB @R W6+d8, A	SUBC A, R6	SUBC @RW6+d8	NEG R6	NEG @RW6+d8	AND R6, A	AND @R W6+d8, A	OR R6, A	OR @RW6+d8, A	XOR R6, A	XOR @R W6+d8, A	NOT R6	NOT @RW6+d8
+7	ADD R7, A	ADD @R W7+d8, A	SUB R7, A	SUB @R W7+d8, A	SUBC A, R7	SUBC @RW7+d8	NEG R7	NEG @RW7+d8	AND R7, A	AND @R W7+d8, A	OR R7, A	OR @RW7+d8, A	XOR R7, A	XOR @R W7+d8, A	NOT R7	NOT @RW7+d8
+8	ADD @RW0, A	ADD @R W0+d16, A	SUB @RW0, A	SUB @R W0+d16, A	SUBC A, @RW0	SUBC @RW0+d16	NEG @RW0	NEG @RW0+d16	AND @RW0, A	AND @R W0+d16, A	OR @RW0, A	OR @RW0+d16, A	XOR @RW0, A	XOR @R W0+d16, A	NOT @RW0	NOT @RW0+d16
+9	ADD @RW1, A	ADD @R W1+d16, A	SUB @RW1, A	SUB @R W1+d16, A	SUBC A, @RW1	SUBC @RW1+d16	NEG @RW1	NEG @RW1+d16	AND @RW1, A	AND @R W1+d16, A	OR @RW1, A	OR @RW1+d16, A	XOR @RW1, A	XOR @R W1+d16, A	NOT @RW1	NOT @RW1+d16
+A	ADD @RW2, A	ADD @R W2+d16, A	SUB @RW2, A	SUB @R W2+d16, A	SUBC A, @RW2	SUBC @RW2+d16	NEG @RW2	NEG @RW2+d16	AND @RW2, A	AND @R W2+d16, A	OR @RW2, A	OR @RW2+d16, A	XOR @RW2, A	XOR @R W2+d16, A	NOT @RW2	NOT @RW2+d16
+B	ADD @RW3, A	ADD @R W3+d16, A	SUB @RW3, A	SUB @R W3+d16, A	SUBC A, @RW3	SUBC @RW3+d16	NEG @RW3	NEG @RW3+d16	AND @RW3, A	AND @R W3+d16, A	OR @RW3, A	OR @RW3+d16, A	XOR @RW3, A	XOR @R W3+d16, A	NOT @RW3	NOT @RW3+d16
+C	ADD @RW0+, A	ADD @R W0+RW7, A	SUB @RW0+, A	SUB @R W0+RW7, A	SUBC A, @RW0+	SUBC @RW0+RW7	NEG @RW0+	NEG @RW0+RW7	AND @RW0+, A	AND @R W0+RW7, A	OR @RW0+, A	OR @R W0+RW7, A	XOR @RW0+, A	XOR @R W0+RW7, A	NOT @RW0+	NOT @RW0+RW7
+D	ADD @RW1+, A	ADD @R W1+RW7, A	SUB @RW1+, A	SUB @R W1+RW7, A	SUBC A, @RW1+	SUBC @RW1+RW7	NEG @RW1+	NEG @RW1+RW7	AND @RW1+, A	AND @R W1+RW7, A	OR @RW1+, A	OR @R W1+RW7, A	XOR @RW1+, A	XOR @R W1+RW7, A	NOT @RW1+	NOT @RW1+RW7
+E	ADD @RW2+, A	ADD @P C+d16, A	SUB @RW2+, A	SUB @P C+d16, A	SUBC A, @RW2+	SUBC @PC+d16	NEG @RW2+	NEG @PC+d16	AND @RW2+, A	AND @P 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 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

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)



Table C.7g ea Instructions [1st byte = 76H]

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW A, RW0	ADDW @RW0+d8	SUBW A, RW0	SUBW @RW0+d8	ADDCW A, RW0	ADDCW @RW0+d8	CMPW A, RW0	CMPW @RW0+d8	ANDW A, RW0	ANDW @RW0+d8	ORW A, RW0	ORW @RW0+d8	XORW A, RW0	XORW @RW0+d8	DWBZ RW0, r	DWBZ @RW0+d8, r
+1	ADDW A, RW1	ADDW @RW1+d8	SUBW A, RW1	SUBW @RW1+d8	ADDCW A, RW1	ADDCW @RW1+d8	CMPW A, RW1	CMPW @RW1+d8	ANDW A, RW1	ANDW @RW1+d8	ORW A, RW1	ORW @RW1+d8	XORW A, RW1	XORW @RW1+d8	DWBZ RW1, r	DWBZ @RW1+d8, r
+2	ADDW A, RW2	ADDW @RW2+d8	SUBW A, RW2	SUBW @RW2+d8	ADDCW A, RW2	ADDCW @RW2+d8	CMPW A, RW2	CMPW @RW2+d8	ANDW A, RW2	ANDW @RW2+d8	ORW A, RW2	ORW @RW2+d8	XORW A, RW2	XORW @RW2+d8	DWBZ RW2, r	DWBZ @RW2+d8, r
+3	ADDW A, RW3	ADDW @RW3+d8	SUBW A, RW3	SUBW @RW3+d8	ADDCW A, RW3	ADDCW @RW3+d8	CMPW A, RW3	CMPW @RW3+d8	ANDW A, RW3	ANDW @RW3+d8	ORW A, RW3	ORW @RW3+d8	XORW A, RW3	XORW @RW3+d8	DWBZ RW3, r	DWBZ @RW3+d8, r
+4	ADDW A, RW4	ADDW @RW4+d8	SUBW A, RW4	SUBW @RW4+d8	ADDCW A, RW4	ADDCW @RW4+d8	CMPW A, RW4	CMPW @RW4+d8	ANDW A, RW4	ANDW @RW4+d8	ORW A, RW4	ORW @RW4+d8	XORW A, RW4	XORW @RW4+d8	DWBZ RW4, r	DWBZ @RW4+d8, r
+5	ADDW A, RW5	ADDW @RW5+d8	SUBW A, RW5	SUBW @RW5+d8	ADDCW A, RW5	ADDCW @RW5+d8	CMPW A, RW5	CMPW @RW5+d8	ANDW A, RW5	ANDW @RW5+d8	ORW A, RW5	ORW @RW5+d8	XORW A, RW5	XORW @RW5+d8	DWBZ RW5, r	DWBZ @RW5+d8, r
+6	ADDW A, RW6	ADDW @RW6+d8	SUBW A, RW6	SUBW @RW6+d8	ADDCW A, RW6	ADDCW @RW6+d8	CMPW A, RW6	CMPW @RW6+d8	ANDW A, RW6	ANDW @RW6+d8	ORW A, RW6	ORW @RW6+d8	XORW A, RW6	XORW @RW6+d8	DWBZ RW6, r	DWBZ @RW6+d8, r
+7	ADDW A, RW7	ADDW @RW7+d8	SUBW A, RW7	SUBW @RW7+d8	ADDCW A, RW7	ADDCW @RW7+d8	CMPW A, RW7	CMPW @RW7+d8	ANDW A, RW7	ANDW @RW7+d8	ORW A, RW7	ORW @RW7+d8	XORW A, RW7	XORW @RW7+d8	DWBZ RW7, r	DWBZ @RW7+d8, r
+8	ADDW A, @RW0	ADDW @RW0+d16	SUBW A, @RW0	SUBW @RW0+d16	ADDCW A, @RW0	ADDCW @RW0+d16	CMPW A, @RW0	CMPW @RW0+d16	ANDW A, @RW0	ANDW @RW0+d16	ORW A, @RW0	ORW @RW0+d16	XORW A, @RW0	XORW @RW0+d16	DWBZ @RW0, r	DWBZ @RW0+d16, r
+9	ADDW A, @RW1	ADDW @RW1+d16	SUBW A, @RW1	SUBW @RW1+d16	ADDCW A, @RW1	ADDCW @RW1+d16	CMPW A, @RW1	CMPW @RW1+d16	ANDW A, @RW1	ANDW @RW1+d16	ORW A, @RW1	ORW @RW1+d16	XORW A, @RW1	XORW @RW1+d16	DWBZ @RW1, r	DWBZ @RW1+d16, r
+A	ADDW A, @RW2	ADDW @RW2+d16	SUBW A, @RW2	SUBW @RW2+d16	ADDCW A, @RW2	ADDCW @RW2+d16	CMPW A, @RW2	CMPW @RW2+d16	ANDW A, @RW2	ANDW @RW2+d16	ORW A, @RW2	ORW @RW2+d16	XORW A, @RW2	XORW @RW2+d16	DWBZ @RW2, r	DWBZ @RW2+d16, r
+B	ADDW A, @RW3	ADDW @RW3+d16	SUBW A, @RW3	SUBW @RW3+d16	ADDCW A, @RW3	ADDCW @RW3+d16	CMPW A, @RW3	CMPW @RW3+d16	ANDW A, @RW3	ANDW @RW3+d16	ORW A, @RW3	ORW @RW3+d16	XORW A, @RW3	XORW @RW3+d16	DWBZ @RW3, r	DWBZ @RW3+d16, r
+C	ADDW A, @RW0+	ADDW @RW0+RW7	SUBW A, @RW0+	SUBW @RW0+RW7	ADDCW A, @RW0+	ADDCW @RW0+RW7	CMPW A, @RW0+	CMPW @RW0+RW7	ANDW A, @RW0+	ANDW @RW0+RW7	ORW A, @RW0+	ORW @RW0+RW7	XORW A, @RW0+	XORW @RW0+RW7	DWBZ @RW0+, r	DWBZ @RW0+RW7, r
+D	ADDW A, @RW1+	ADDW @RW1+RW7	SUBW A, @RW1+	SUBW @RW1+RW7	ADDCW A, @RW1+	ADDCW @RW1+RW7	CMPW A, @RW1+	CMPW @RW1+RW7	ANDW A, @RW1+	ANDW @RW1+RW7	ORW A, @RW1+	ORW @RW1+RW7	XORW A, @RW1+	XORW @RW1+RW7	DWBZ @RW1+, r	DWBZ @RW1+RW7, r
+E	ADDW A, @RW2+	ADDW @PC+d16	SUBW A, @RW2+	SUBW @PC+d16	ADDCW A, @RW2+	ADDCW @PC+d16	CMPW A, @RW2+	CMPW @PC+d16	ANDW A, @RW2+	ANDW @PC+d16	ORW A, @RW2+	ORW @PC+d16	XORW A, @RW2+	XORW @PC+d16	DWBZ @RW2+, r	DWBZ @PC+d16, r
+F	ADDW A, @RW3+	ADDW addr16	SUBW A, @RW3+	SUBW addr16	ADDCW A, @RW3+	ADDCW addr16	CMPW A, @RW3+	CMPW addr16	ANDW A, @RW3+	ANDW addr16	ORW A, @RW3+	ORW addr16	XORW A, @RW3+	XORW addr16	DWBZ @RW3+, r	DWBZ addr16, r

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

**Table C.7h ea Instructions 1st byte = 77H]**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	ADDW RW0, A	ADDW @R W0+d8, A	SUBW @R RW0, A	SUBW @R W0+d8, A	SUBCW A, RW0	SUBCW A, @RW0+d8	NEGW RW0	NEGW @RW0+d8	ANDW RW0, A	ANDW @R W0+d8, A	ORW RW0, A	ORW @R W0+d8, A	XORW RW0, A	XORW @R W0+d8, A	NOTW RW0	NOTW @RW0+d8
+1	ADDW RW1, A	ADDW @R W1+d8, A	SUBW @R RW1, A	SUBW @R W1+d8, A	SUBCW A, RW1	SUBCW A, @RW1+d8	NEGW RW1	NEGW @RW1+d8	ANDW RW1, A	ANDW @R W1+d8, A	ORW RW1, A	ORW @R W1+d8, A	XORW RW1, A	XORW @R W1+d8, A	NOTW RW1	NOTW @RW1+d8
+2	ADDW RW2, A	ADDW @R W2+d8, A	SUBW @R RW2, A	SUBW @R W2+d8, A	SUBCW A, RW2	SUBCW A, @RW2+d8	NEGW RW2	NEGW @RW2+d8	ANDW RW2, A	ANDW @R W2+d8, A	ORW RW2, A	ORW @R W2+d8, A	XORW RW2, A	XORW @R W2+d8, A	NOTW RW2	NOTW @RW2+d8
+3	ADDW RW3, A	ADDW @R W3+d8, A	SUBW @R RW3, A	SUBW @R W3+d8, A	SUBCW A, RW3	SUBCW A, @RW3+d8	NEGW RW3	NEGW @RW3+d8	ANDW RW3, A	ANDW @R W3+d8, A	ORW RW3, A	ORW @R W3+d8, A	XORW RW3, A	XORW @R W3+d8, A	NOTW RW3	NOTW @RW3+d8
+4	ADDW RW4, A	ADDW @R W4+d8, A	SUBW @R RW4, A	SUBW @R W4+d8, A	SUBCW A, RW4	SUBCW A, @RW4+d8	NEGW RW4	NEGW @RW4+d8	ANDW RW4, A	ANDW @R W4+d8, A	ORW RW4, A	ORW @R W4+d8, A	XORW RW4, A	XORW @R W4+d8, A	NOTW RW4	NOTW @RW4+d8
+5	ADDW RW5, A	ADDW @R W5+d8, A	SUBW @R RW5, A	SUBW @R W5+d8, A	SUBCW A, RW5	SUBCW A, @RW5+d8	NEGW RW5	NEGW @RW5+d8	ANDW RW5, A	ANDW @R W5+d8, A	ORW RW5, A	ORW @R W5+d8, A	XORW RW5, A	XORW @R W5+d8, A	NOTW RW5	NOTW @RW5+d8
+6	ADDW RW6, A	ADDW @R W6+d8, A	SUBW @R RW6, A	SUBW @R W6+d8, A	SUBCW A, RW6	SUBCW A, @RW6+d8	NEGW RW6	NEGW @RW6+d8	ANDW RW6, A	ANDW @R W6+d8, A	ORW RW6, A	ORW @R W6+d8, A	XORW RW6, A	XORW @R W6+d8, A	NOTW RW6	NOTW @RW6+d8
+7	ADDW RW7, A	ADDW @R W7+d8, A	SUBW @R RW7, A	SUBW @R W7+d8, A	SUBCW A, RW7	SUBCW A, @RW7+d8	NEGW RW7	NEGW @RW7+d8	ANDW RW7, A	ANDW @R W7+d8, A	ORW RW7, A	ORW @R W7+d8, A	XORW RW7, A	XORW @R W7+d8, A	NOTW RW7	NOTW @RW7+d8
+8	ADDW @RW0, A	ADDW @R W0+d16, A	SUBW @R @RW0, A	SUBW @R W0+d16, A	SUBCW A, @RW0	SUBCW A, @RW0+d16	NEGW @RW0	NEGW @RW0+d16	ANDW @RW0, A	ANDW @R W0+d16, A	ORW @RW0, A	ORW @R W0+d16, A	XORW @RW0, A	XORW @R W0+d16, A	NOTW @RW0	NOTW @RW0+d16
+9	ADDW @RW1, A	ADDW @R W1+d16, A	SUBW @R @RW1, A	SUBW @R W1+d16, A	SUBCW A, @RW1	SUBCW A, @RW1+d16	NEGW @RW1	NEGW @RW1+d16	ANDW @RW1, A	ANDW @R W1+d16, A	ORW @RW1, A	ORW @R W1+d16, A	XORW @RW1, A	XORW @R W1+d16, A	NOTW @RW1	NOTW @RW1+d16
+A	ADDW @RW2, A	ADDW @R W2+d16, A	SUBW @R @RW2, A	SUBW @R W2+d16, A	SUBCW A, @RW2	SUBCW A, @RW2+d16	NEGW @RW2	NEGW @RW2+d16	ANDW @RW2, A	ANDW @R W2+d16, A	ORW @RW2, A	ORW @R W2+d16, A	XORW @RW2, A	XORW @R W2+d16, A	NOTW @RW2	NOTW @RW2+d16
+B	ADDW @RW3, A	ADDW @R W3+d16, A	SUBW @R @RW3, A	SUBW @R W3+d16, A	SUBCW A, @RW3	SUBCW A, @RW3+d16	NEGW @RW3	NEGW @RW3+d16	ANDW @RW3, A	ANDW @R W3+d16, A	ORW @RW3, A	ORW @R W3+d16, A	XORW @RW3, A	XORW @R W3+d16, A	NOTW @RW3	NOTW @RW3+d16
+C	ADDW @RW0+, A	ADDW @R W0+RW7	SUBW @R @RW0+, A	SUBW @R W0+RW7, A	SUBCW A, @RW0+	SUBCW A, @RW0+RW7	NEGW @RW0+	NEGW @RW0+RW7	ANDW @RW0+, A	ANDW @R W0+RW7, A	ORW @RW0+, A	ORW @R W0+RW7, A	XORW @RW0+, A	XORW @R W0+RW7, A	NOTW @RW0+	NOTW @RW0+RW7
+D	ADDW @RW1+, A	ADDW @R W1+RW7	SUBW @R @RW1+, A	SUBW @R W1+RW7, A	SUBCW A, @RW1+	SUBCW A, @RW1+RW7	NEGW @RW1+	NEGW @RW1+RW7	ANDW @RW1+, A	ANDW @R W1+RW7, A	ORW @RW1+, A	ORW @R W1+RW7, A	XORW @RW1+, A	XORW @R W1+RW7, A	NOTW @RW1+	NOTW @RW1+RW7
+E	ADDW @RW2+, A	ADDW @P C+d16, A	SUBW @R @RW2+, A	SUBW @P @PC+d16, A	SUBCW A, @RW2+	SUBCW A, @PC+d16	NEGW @RW2+	NEGW @PC+d16	ANDW @RW2+, A	ANDW @P C+d16, A	ORW @RW2+, A	ORW @P C+d16, A	XORW @RW2+, A	XORW @P C+d16, A	NOTW @RW2+	NOTW @PC+d16
+F	ADDW @RW3+, A	ADDW addr16, A	SUBW @R @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

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

**Table C.7i ea Instructions [1st byte = 78H]**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MULU A, R0	MULU @RW0+d8	MULUW A, RW0	MULUW @RW0+d8	MUL A, R0	MUL @RW0+d8	MULW A, RW0	MULW @RW0+d8	DIVU A, R0	DIVU @RW0+d8	DIVUW A, RW0	DIVUW @RW0+d8	DIV A, R0	DIV @RW0+d8	DIVW A, RW0	DIVW @RW0+d8
+1	MULU A, R1	MULU @RW1+d8	MULUW A, RW1	MULUW @RW1+d8	MUL A, R1	MUL @RW1+d8	MULW A, RW1	MULW @RW1+d8	DIVU A, R1	DIVU @RW1+d8	DIVUW A, RW1	DIVUW @RW1+d8	DIV A, R1	DIV @RW1+d8	DIVW A, RW1	DIVW @RW1+d8
+2	MULU A, R2	MULU @RW2+d8	MULUW A, RW2	MULUW @RW2+d8	MUL A, R2	MUL @RW2+d8	MULW A, RW2	MULW @RW2+d8	DIVU A, R2	DIVU @RW2+d8	DIVUW A, RW2	DIVUW @RW2+d8	DIV A, R2	DIV @RW2+d8	DIVW A, RW2	DIVW @RW2+d8
+3	MULU A, R3	MULU @RW3+d8	MULUW A, RW3	MULUW @RW3+d8	MUL A, R3	MUL @RW3+d8	MULW A, RW3	MULW @RW3+d8	DIVU A, R3	DIVU @RW3+d8	DIVUW A, RW3	DIVUW @RW3+d8	DIV A, R3	DIV @RW3+d8	DIVW A, RW3	DIVW @RW3+d8
+4	MULU A, R4	MULU @RW4+d8	MULUW A, RW4	MULUW @RW4+d8	MUL A, R4	MUL @RW4+d8	MULW A, RW4	MULW @RW4+d8	DIVU A, R4	DIVU @RW4+d8	DIVUW A, RW4	DIVUW @RW4+d8	DIV A, R4	DIV @RW4+d8	DIVW A, RW4	DIVW @RW4+d8
+5	MULU A, R5	MULU @RW5+d8	MULUW A, RW5	MULUW @RW5+d8	MULU A, R5	MULU @RW5+d8	MULW A, RW5	MULW @RW5+d8	DIVU A, R5	DIVU @RW5+d8	DIVUW A, RW5	DIVUW @RW5+d8	DIV A, R5	DIV @RW5+d8	DIVW A, RW5	DIVW @RW5+d8
+6	MULU A, R6	MULU @RW6+d8	MULUW A, RW6	MULUW @RW6+d8	MUL A, R6	MUL @RW6+d8	MULW A, RW6	MULW @RW6+d8	DIVU A, R6	DIVU @RW6+d8	DIVUW A, RW6	DIVUW @RW6+d8	DIV A, R6	DIV @RW6+d8	DIVW A, RW6	DIVW @RW6+d8
+7	MULU A, R7	MULU @RW7+d8	MULUW A, RW7	MULUW @RW7+d8	MUL A, R7	MUL @RW7+d8	MULW A, RW7	MULW @RW7+d8	DIVU A, R7	DIVU @RW7+d8	DIVUW A, RW7	DIVUW @RW7+d8	DIV A, R7	DIV @RW7+d8	DIVW A, RW7	DIVW @RW7+d8
+8	MULU A, @RW0	MULU @RW0+d16	MULUW A, @RW0	MULUW @RW0+d16	MUL A, @RW0	MUL @RW0+d16	MULW A, @RW0	MULW @RW0+d16	DIVU A, @RW0	DIVU @RW0+d16	DIVUW A, @RW0	DIVUW @RW0+d16	DIV A, @RW0	DIV @RW0+d16	DIVW A, @RW0	DIVW @RW0+d16
+9	MULU A, @RW1	MULU @RW1+d16	MULUW A, @RW1	MULUW @RW1+d16	MUL A, @RW1	MUL @RW1+d16	MULW A, @RW1	MULW @RW1+d16	DIVU A, @RW1	DIVU @RW1+d16	DIVUW A, @RW1	DIVUW @RW1+d16	DIV A, @RW1	DIV @RW1+d16	DIVW A, @RW1	DIVW @RW1+d16
+A	MULU A, @RW2	MULU @RW2+d16	MULUW A, @RW2	MULUW @RW2+d16	MUL A, @RW2	MUL @RW2+d16	MULW A, @RW2	MULW @RW2+d16	DIVU A, @RW2	DIVU @RW2+d16	DIVUW A, @RW2	DIVUW @RW2+d16	DIV A, @RW2	DIV @RW2+d16	DIVW A, @RW2	DIVW @RW2+d16
+B	MULU A, @RW3	MULU @RW3+d16	MULUW A, @RW3	MULUW @RW3+d16	MUL A, @RW3	MUL @RW3+d16	MULW A, @RW3	MULW @RW3+d16	DIVU A, @RW3	DIVU @RW3+d16	DIVUW A, @RW3	DIVUW @RW3+d16	DIV A, @RW3	DIV @RW3+d16	DIVW A, @RW3	DIVW @RW3+d16
+C	MULU A, @RW0+	MULU @RW0+RW7	MULUW A, @RW0+	MULUW @RW0+RW7	MUL A, @RW0+	MUL @RW0+RW7	MULW A, @RW0+	MULW @RW0+RW7	DIVU A, @RW0+	DIVU @RW0+RW7	DIVUW A, @RW0+	DIVUW @RW0+RW7	DIV A, @RW0+	DIV @RW0+RW7	DIVW A, @RW0+	DIVW @RW0+RW7
+D	MULU A, @RW1+	MULU @RW1+RW7	MULUW A, @RW1+	MULUW @RW1+RW7	MUL A, @RW1+	MUL @RW1+RW7	MULW A, @RW1+	MULW @RW1+RW7	DIVU A, @RW1+	DIVU @RW1+RW7	DIVUW A, @RW1+	DIVUW @RW1+RW7	DIV A, @RW1+	DIV @RW1+RW7	DIVW A, @RW1+	DIVW @RW1+RW7
+E	MULU A, @RW2+	MULU @PC+d16	MULUW A, @RW2+	MULUW @PC+d16	MUL A, @RW2+	MUL @PC+d16	MULW A, @RW2+	MULW @PC+d16	DIVU A, @RW2+	DIVU @PC+d16	DIVUW A, @RW2+	DIVUW @PC+d16	DIV A, @RW2+	DIV @PC+d16	DIVW A, @RW2+	DIVW @PC+d16
+F	MULU A, @RW3+	MULU addr16	MULUW A, @RW3+	MULUW addr16	MUL A, @RW3+	MUL addr16	MULW A, @RW3+	MULW addr16	DIVU A, @RW3+	DIVU addr16	DIVUW A, @RW3+	DIVUW addr16	DIV A, @RW3+	DIV addr16	DIVW A, @RW3+	DIVW addr16

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

# C.8 MOVEA RWi, ea

Table C.8 MOVEA RWi, ea [Frist 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, RW1	MOVEA RW1, @RW0+d8	MOVEA RW2, RW2	MOVEA RW2, @RW0+d8	MOVEA RW3, RW3	MOVEA RW3, @RW0+d8	MOVEA RW4, RW4	MOVEA RW4, @RW0+d8	MOVEA RW5, RW5	MOVEA RW5, @RW0+d8	MOVEA RW6, RW6	MOVEA RW6, @RW0+d8	MOVEA RW7, RW7	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, @RW1	MOVEA RW1, @RW1+d16	MOVEA RW2, @RW2	MOVEA RW2, @RW2+d16	MOVEA RW3, @RW3	MOVEA RW3, @RW3+d16	MOVEA RW4, @RW4	MOVEA RW4, @RW4+d16	MOVEA RW5, @RW5	MOVEA RW5, @RW5+d16	MOVEA RW6, @RW6	MOVEA RW6, @RW6+d16	MOVEA RW7, @RW7	MOVEA RW7, @RW7+d16
+9	MOVEA RW0, @RW1	MOVEA RW0, @RW1+d16	MOVEA RW1, @RW1	MOVEA RW1, @RW1+d16	MOVEA RW2, @RW2	MOVEA RW2, @RW2+d16	MOVEA RW3, @RW3	MOVEA RW3, @RW3+d16	MOVEA RW4, @RW4	MOVEA RW4, @RW4+d16	MOVEA RW5, @RW5	MOVEA RW5, @RW5+d16	MOVEA RW6, @RW6	MOVEA RW6, @RW6+d16	MOVEA RW7, @RW7	MOVEA RW7, @RW7+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, @RW3	MOVEA RW3, @RW3+d16	MOVEA RW4, @RW4	MOVEA RW4, @RW4+d16	MOVEA RW5, @RW5	MOVEA RW5, @RW5+d16	MOVEA RW6, @RW6	MOVEA RW6, @RW6+d16	MOVEA RW7, @RW7	MOVEA RW7, @RW7+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

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

## C.9 MOV Ri, ea

Table C.9 MOV Ri, ea [First byte = 7Ah]

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0	MOV R0, @RW0+d8	MOV R1, R0	MOV R1, @RW0+d8	MOV R2, R0	MOV R2, @RW0+d8	MOV R3, R0	MOV R3, @RW0+d8	MOV R4, R0	MOV R4, @RW0+d8	MOV R5, R0	MOV R5, @RW0+d8	MOV R6, R0	MOV R6, @RW0+d8	MOV R7, R0	MOV R7, @RW0+d8
+1	MOV R0, R1	MOV R0, @RW1+d8	MOV R1, R1	MOV R1, @RW1+d8	MOV R2, R1	MOV R2, @RW1+d8	MOV R3, R1	MOV R3, @RW1+d8	MOV R4, R1	MOV R4, @RW1+d8	MOV R5, R1	MOV R5, @RW1+d8	MOV R6, R1	MOV R6, @RW1+d8	MOV R7, R1	MOV R7, @RW1+d8
+2	MOV R0, R2	MOV R0, @RW2+d8	MOV R1, R2	MOV R1, @RW2+d8	MOV R2, R2	MOV R2, @RW2+d8	MOV R3, R2	MOV R3, @RW2+d8	MOV R4, R2	MOV R4, @RW2+d8	MOV R5, R2	MOV R5, @RW2+d8	MOV R6, R2	MOV R6, @RW2+d8	MOV R7, R2	MOV R7, @RW2+d8
+3	MOV R0, R3	MOV R0, @RW3+d8	MOV R1, R3	MOV R1, @RW3+d8	MOV R2, R3	MOV R2, @RW3+d8	MOV R3, R3	MOV R3, @RW3+d8	MOV R4, R3	MOV R4, @RW3+d8	MOV R5, R3	MOV R5, @RW3+d8	MOV R6, R3	MOV R6, @RW3+d8	MOV R7, R3	MOV R7, @RW3+d8
+4	MOV R0, R4	MOV R0, @RW4+d8	MOV R1, R4	MOV R1, @RW4+d8	MOV R2, R4	MOV R2, @RW4+d8	MOV R3, R4	MOV R3, @RW4+d8	MOV R4, R4	MOV R4, @RW4+d8	MOV R5, R4	MOV R5, @RW4+d8	MOV R6, R4	MOV R6, @RW4+d8	MOV R7, R4	MOV R7, @RW4+d8
+5	MOV R0, R5	MOV R0, @RW5+d8	MOV R1, R5	MOV R1, @RW5+d8	MOV R2, R5	MOV R2, @RW5+d8	MOV R3, R5	MOV R3, @RW5+d8	MOV R4, R5	MOV R4, @RW5+d8	MOV R5, R5	MOV R5, @RW5+d8	MOV R6, R5	MOV R6, @RW5+d8	MOV R7, R5	MOV R7, @RW5+d8
+6	MOV R0, R6	MOV R0, @RW6+d8	MOV R1, R6	MOV R1, @RW6+d8	MOV R2, R6	MOV R2, @RW6+d8	MOV R3, R6	MOV R3, @RW6+d8	MOV R4, R6	MOV R4, @RW6+d8	MOV R5, R6	MOV R5, @RW6+d8	MOV R6, R6	MOV R6, @RW6+d8	MOV R7, R6	MOV R7, @RW6+d8
+7	MOV R0, R7	MOV R0, @RW7+d8	MOV R1, R7	MOV R1, @RW7+d8	MOV R2, R7	MOV R2, @RW7+d8	MOV R3, R7	MOV R3, @RW7+d8	MOV R4, R7	MOV R4, @RW7+d8	MOV R5, R7	MOV R5, @RW7+d8	MOV R6, R7	MOV R6, @RW7+d8	MOV R7, R7	MOV R7, @RW7+d8
+8	MOV R0, @RW0	MOV R0, @RW0+d16	MOV R1, @RW0	MOV R1, @RW0+d16	MOV R2, @RW0	MOV R2, @RW0+d16	MOV R3, @RW0	MOV R3, @RW0+d16	MOV R4, @RW0	MOV R4, @RW0+d16	MOV R5, @RW0	MOV R5, @RW0+d16	MOV R6, @RW0	MOV R6, @RW0+d16	MOV R7, @RW0	MOV R7, @RW0+d16
+9	MOV R0, @RW1	MOV R0, @RW1+d16	MOV R1, @RW1	MOV R1, @RW1+d16	MOV R2, @RW1	MOV R2, @RW1+d16	MOV R3, @RW1	MOV R3, @RW1+d16	MOV R4, @RW1	MOV R4, @RW1+d16	MOV R5, @RW1	MOV R5, @RW1+d16	MOV R6, @RW1	MOV R6, @RW1+d16	MOV R7, @RW1	MOV R7, @RW1+d16
+A	MOV R0, @RW2	MOV R0, @RW2+d16	MOV R1, @RW2	MOV R1, @RW2+d16	MOV R2, @RW2	MOV R2, @RW2+d16	MOV R3, @RW2	MOV R3, @RW2+d16	MOV R4, @RW2	MOV R4, @RW2+d16	MOV R5, @RW2	MOV R5, @RW2+d16	MOV R6, @RW2	MOV R6, @RW2+d16	MOV R7, @RW2	MOV R7, @RW2+d16
+B	MOV R0, @RW3	MOV R0, @RW3+d16	MOV R1, @RW3	MOV R1, @RW3+d16	MOV R2, @RW3	MOV R2, @RW3+d16	MOV R3, @RW3	MOV R3, @RW3+d16	MOV R4, @RW3	MOV R4, @RW3+d16	MOV R5, @RW3	MOV R5, @RW3+d16	MOV R6, @RW3	MOV R6, @RW3+d16	MOV R7, @RW3	MOV R7, @RW3+d16
+C	MOV R0, @RW0+	MOV R0, @RW0+RW7	MOV R1, @RW0+	MOV R1, @RW0+RW7	MOV R2, @RW0+	MOV R2, @RW0+RW7	MOV R3, @RW0+	MOV R3, @RW0+RW7	MOV R4, @RW0+	MOV R4, @RW0+RW7	MOV R5, @RW0+	MOV R5, @RW0+RW7	MOV R6, @RW0+	MOV R6, @RW0+RW7	MOV R7, @RW0+	MOV R7, @RW0+RW7
+D	MOV R0, @RW1+	MOV R0, @RW1+RW7	MOV R1, @RW1+	MOV R1, @RW1+RW7	MOV R2, @RW1+	MOV R2, @RW1+RW7	MOV R3, @RW1+	MOV R3, @RW1+RW7	MOV R4, @RW1+	MOV R4, @RW1+RW7	MOV R5, @RW1+	MOV R5, @RW1+RW7	MOV R6, @RW1+	MOV R6, @RW1+RW7	MOV R7, @RW1+	MOV R7, @RW1+RW7
+E	MOV R0, @RW2+	MOV R0, @PC+d16	MOV R1, @RW2+	MOV R1, @PC+d16	MOV R2, @RW2+	MOV R2, @PC+d16	MOV R3, @RW2+	MOV R3, @PC+d16	MOV R4, @RW2+	MOV R4, @PC+d16	MOV R5, @RW2+	MOV R5, @PC+d16	MOV R6, @RW2+	MOV R6, @PC+d16	MOV R7, @RW2+	MOV R7, @PC+d16
+F	MOV R0, @RW3+	MOV R0, @addr16	MOV R1, @RW3+	MOV R1, @addr16	MOV R2, @RW3+	MOV R2, @addr16	MOV R3, @RW3+	MOV R3, @addr16	MOV R4, @RW3+	MOV R4, @addr16	MOV R5, @RW3+	MOV R5, @addr16	MOV R6, @RW3+	MOV R6, @addr16	MOV R7, @RW3+	MOV R7, @addr16

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

# C.10 MOVW RWi, ea

Table C.10 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

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

## C.11 MOV ea, Ri

Table C.11 MOV ea, Ri [First byte = 7CH]

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV R0, R0	MOV @R, W0+d8, R0	MOV R0, R1	MOV @R, W0+d8, R1	MOV R0, R2	MOV @R, W0+d8, R2	MOV R0, R3	MOV @R, W0+d8, R3	MOV R0, R4	MOV @R, W0+d8, R4	MOV R0, R5	MOV @R, W0+d8, R5	MOV R0, R6	MOV @R, W0+d8, R6	MOV R0, R7	MOV @R, W0+d8, R7
+1	MOV R1, R0	MOV @R, W1+d8, R0	MOV R1, R1	MOV @R, W1+d8, R1	MOV R1, R2	MOV @R, W1+d8, R2	MOV R1, R3	MOV @R, W1+d8, R3	MOV R1, R4	MOV @R, W1+d8, R4	MOV R1, R5	MOV @R, W1+d8, R5	MOV R1, R6	MOV @R, W1+d8, R6	MOV R1, R7	MOV @R, W1+d8, R7
+2	MOV R2, R0	MOV @R, W2+d8, R0	MOV R2, R1	MOV @R, W2+d8, R1	MOV R2, R2	MOV @R, W2+d8, R2	MOV R2, R3	MOV @R, W2+d8, R3	MOV R2, R4	MOV @R, W2+d8, R4	MOV R2, R5	MOV @R, W2+d8, R5	MOV R2, R6	MOV @R, W2+d8, R6	MOV R2, R7	MOV @R, W2+d8, R7
+3	MOV R3, R0	MOV @R, W3+d8, R0	MOV R3, R1	MOV @R, W3+d8, R1	MOV R3, R2	MOV @R, W3+d8, R2	MOV R3, R3	MOV @R, W3+d8, R3	MOV R3, R4	MOV @R, W3+d8, R4	MOV R3, R5	MOV @R, W3+d8, R5	MOV R3, R6	MOV @R, W3+d8, R6	MOV R3, R7	MOV @R, W3+d8, R7
+4	MOV R4, R0	MOV @R, W4+d8, R0	MOV R4, R1	MOV @R, W4+d8, R1	MOV R4, R2	MOV @R, W4+d8, R2	MOV R4, R3	MOV @R, W4+d8, R3	MOV R4, R4	MOV @R, W4+d8, R4	MOV R4, R5	MOV @R, W4+d8, R5	MOV R4, R6	MOV @R, W4+d8, R6	MOV R4, R7	MOV @R, W4+d8, R7
+5	MOV R5, R0	MOV @R, W5+d8, R0	MOV R5, R1	MOV @R, W5+d8, R1	MOV R5, R2	MOV @R, W5+d8, R2	MOV R5, R3	MOV @R, W5+d8, R3	MOV R5, R4	MOV @R, W5+d8, R4	MOV R5, R5	MOV @R, W5+d8, R5	MOV R5, R6	MOV @R, W5+d8, R6	MOV R5, R7	MOV @R, W5+d8, R7
+6	MOV R6, R0	MOV @R, W6+d8, R0	MOV R6, R1	MOV @R, W6+d8, R1	MOV R6, R2	MOV @R, W6+d8, R2	MOV R6, R3	MOV @R, W6+d8, R3	MOV R6, R4	MOV @R, W6+d8, R4	MOV R6, R5	MOV @R, W6+d8, R5	MOV R6, R6	MOV @R, W6+d8, R6	MOV R6, R7	MOV @R, W6+d8, R7
+7	MOV R7, R0	MOV @R, W7+d8, R0	MOV R7, R1	MOV @R, W7+d8, R1	MOV R7, R2	MOV @R, W7+d8, R2	MOV R7, R3	MOV @R, W7+d8, R3	MOV R7, R4	MOV @R, W7+d8, R4	MOV R7, R5	MOV @R, W7+d8, R5	MOV R7, R6	MOV @R, W7+d8, R6	MOV R7, R7	MOV @R, W7+d8, R7
+8	MOV @RW0, R0	MOV @RW, W0+d16, R0	MOV @RW0, R1	MOV @RW, 0+d16, R1	MOV @RW0, R2	MOV @RW, 0+d16, R2	MOV @RW0, R3	MOV @RW, 0+d16, R3	MOV @RW0, R4	MOV @RW, 0+d16, R4	MOV @RW0, R5	MOV @RW, 0+d16, R5	MOV @RW0, R6	MOV @RW, 0+d16, R6	MOV @RW0, R7	MOV @RW, 0+d16, R7
+9	MOV @RW1, R0	MOV @RW, W1+d16, R0	MOV @RW1, R1	MOV @RW, 1+d16, R1	MOV @RW1, R2	MOV @RW, 1+d16, R2	MOV @RW1, R3	MOV @RW, 1+d16, R3	MOV @RW1, R4	MOV @RW, 1+d16, R4	MOV @RW1, R5	MOV @RW, 1+d16, R5	MOV @RW1, R6	MOV @RW, 1+d16, R6	MOV @RW1, R7	MOV @RW, 1+d16, R7
+A	MOV @RW2, R0	MOV @RW, W2+d16, R0	MOV @RW2, R1	MOV @RW, 2+d16, R1	MOV @RW2, R2	MOV @RW, 2+d16, R2	MOV @RW2, R3	MOV @RW, 2+d16, R3	MOV @RW2, R4	MOV @RW, 2+d16, R4	MOV @RW2, R5	MOV @RW, 2+d16, R5	MOV @RW2, R6	MOV @RW, 2+d16, R6	MOV @RW2, R7	MOV @RW, 2+d16, R7
+B	MOV @RW3, R0	MOV @RW, W3+d16, R0	MOV @RW3, R1	MOV @RW, 3+d16, R1	MOV @RW3, R2	MOV @RW, 3+d16, R2	MOV @RW3, R3	MOV @RW, 3+d16, R3	MOV @RW3, R4	MOV @RW, 3+d16, R4	MOV @RW3, R5	MOV @RW, 3+d16, R5	MOV @RW3, R6	MOV @RW, 3+d16, R6	MOV @RW3, R7	MOV @RW, 3+d16, R7
+C	MOV @RW0+, R0	MOV @RW, W0+RW7, R0	MOV @RW0+, R1	MOV @RW, 0+RW7, R1	MOV @RW0+, R2	MOV @RW, 0+RW7, R2	MOV @RW0+, R3	MOV @RW, 0+RW7, R3	MOV @RW0+, R4	MOV @RW, 0+RW7, R4	MOV @RW0+, R5	MOV @RW, 0+RW7, R5	MOV @RW0+, R6	MOV @RW, 0+RW7, R6	MOV @RW0+, R7	MOV @RW, 0+RW7, R7
+D	MOV @RW1+, R0	MOV @RW, W1+RW7, R0	MOV @RW1+, R1	MOV @RW, 1+RW7, R1	MOV @RW1+, R2	MOV @RW, 1+RW7, R2	MOV @RW1+, R3	MOV @RW, 1+RW7, R3	MOV @RW1+, R4	MOV @RW, 1+RW7, R4	MOV @RW1+, R5	MOV @RW, 1+RW7, R5	MOV @RW1+, R6	MOV @RW, 1+RW7, R6	MOV @RW1+, R7	MOV @RW, 1+RW7, R7
+E	MOV @RW2+, R0	MOV @RW, PC+d16, R0	MOV @RW2+, R1	MOV @RW, PC+d16, R1	MOV @RW2+, R2	MOV @RW, PC+d16, R2	MOV @RW2+, R3	MOV @RW, PC+d16, R3	MOV @RW2+, R4	MOV @RW, PC+d16, R4	MOV @RW2+, R5	MOV @RW, PC+d16, R5	MOV @RW2+, R6	MOV @RW, PC+d16, R6	MOV @RW2+, R7	MOV @RW, PC+d16, R7
+F	MOV @RW3+, R0	MOV @RW, a addr16, R0	MOV @RW3+, R1	MOV @RW, a addr16, R1	MOV @RW3+, R2	MOV @RW, a addr16, R2	MOV @RW3+, R3	MOV @RW, a addr16, R3	MOV @RW3+, R4	MOV @RW, a addr16, R4	MOV @RW3+, R5	MOV @RW, a addr16, R5	MOV @RW3+, R6	MOV @RW, a addr16, R6	MOV @RW3+, R7	MOV @RW, a addr16, R7

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

# C.12 MOVW ea, RWi

Table C.12 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 @R 1+d8, RW1	MOVW RW1, RW2	MOVW @R 1+d8, RW2	MOVW RW1, RW3	MOVW @RW 1+d8, RW3	MOVW RW1, RW4	MOVW @RW 1+d8, RW4	MOVW RW1, RW5	MOVW @RW 1+d8, RW5	MOVW RW1, RW6	MOVW @RW 1+d8, RW6	MOVW RW1, RW7	MOVW @RW 1+d8, RW7
+2	MOVW RW2, RW0	MOVW @RW 2+d8, RW0	MOVW RW2, RW1	MOVW @RW 2+d8, RW1	MOVW RW2, RW2	MOVW @RW 2+d8, RW2	MOVW RW2, RW3	MOVW @RW 2+d8, RW3	MOVW RW2, RW4	MOVW @RW 2+d8, RW4	MOVW RW2, RW5	MOVW @RW 2+d8, RW5	MOVW RW2, RW6	MOVW @RW 2+d8, RW6	MOVW RW2, RW7	MOVW @RW 2+d8, RW7
+3	MOVW RW3, RW0	MOVW @RW 3+d8, RW0	MOVW RW3, RW1	MOVW @RW 3+d8, RW1	MOVW RW3, RW2	MOVW @RW 3+d8, RW2	MOVW RW3, RW3	MOVW @RW 3+d8, RW3	MOVW RW3, RW4	MOVW @RW 3+d8, RW4	MOVW RW3, RW5	MOVW @RW 3+d8, RW5	MOVW RW3, RW6	MOVW @RW 3+d8, RW6	MOVW RW3, RW7	MOVW @RW 3+d8, RW7
+4	MOVW RW4, RW0	MOVW @RW 4+d8, RW0	MOVW RW4, RW1	MOVW @RW 4+d8, RW1	MOVW RW4, RW2	MOVW @RW 4+d8, RW2	MOVW RW4, RW3	MOVW @RW 4+d8, RW3	MOVW RW4, RW4	MOVW @RW 4+d8, RW4	MOVW RW4, RW5	MOVW @RW 4+d8, RW5	MOVW RW4, RW6	MOVW @RW 4+d8, RW6	MOVW RW4, RW7	MOVW @RW 4+d8, RW7
+5	MOVW RW5, RW0	MOVW @RW 5+d8, RW0	MOVW RW5, RW1	MOVW @RW 5+d8, RW1	MOVW RW5, RW2	MOVW @RW 5+d8, RW2	MOVW RW5, RW3	MOVW @RW 5+d8, RW3	MOVW RW5, RW4	MOVW @RW 5+d8, RW4	MOVW RW5, RW5	MOVW @RW 5+d8, RW5	MOVW RW5, RW6	MOVW @RW 5+d8, RW6	MOVW RW5, RW7	MOVW @RW 5+d8, RW7
+6	MOVW RW6, RW0	MOVW @RW 6+d8, RW0	MOVW RW6, RW1	MOVW @RW 6+d8, RW1	MOVW RW6, RW2	MOVW @RW 6+d8, RW2	MOVW RW6, RW3	MOVW @RW 6+d8, RW3	MOVW RW6, RW4	MOVW @RW 6+d8, RW4	MOVW RW6, RW5	MOVW @RW 6+d8, RW5	MOVW RW6, RW6	MOVW @RW 6+d8, RW6	MOVW RW6, RW7	MOVW @RW 6+d8, RW7
+7	MOVW RW7, RW0	MOVW @RW 7+d8, RW0	MOVW RW7, RW1	MOVW @RW 7+d8, RW1	MOVW RW7, RW2	MOVW @RW 7+d8, RW2	MOVW RW7, RW3	MOVW @RW 7+d8, RW3	MOVW RW7, RW4	MOVW @RW 7+d8, RW4	MOVW RW7, RW5	MOVW @RW 7+d8, RW5	MOVW RW7, RW6	MOVW @RW 7+d8, RW6	MOVW RW7, RW7	MOVW @RW 7+d8, RW7
+8	MOVW @RW0, RW0	MOVW@RW0 +d16, RW0	MOVW @RW0, RW1	MOVW@RW0 +d16, RW1	MOVW @RW0, RW2	MOVW@RW0 +d16, RW2	MOVW @RW0, RW3	MOVW@RW0 +d16, RW3	MOVW @RW0, RW4	MOVW@RW0 +d16, RW4	MOVW @RW0, RW5	MOVW@RW0 +d16, RW5	MOVW @RW0, RW6	MOVW@RW0 +d16, RW6	MOVW @RW0, RW7	MOVW@RW0 +d16, RW7
+9	MOVW @RW1, RW0	MOVW@RW1 +d16, RW0	MOVW @RW1, RW1	MOVW@RW1 +d16, RW1	MOVW @RW1, RW2	MOVW@RW1 +d16, RW2	MOVW @RW1, RW3	MOVW@RW1 +d16, RW3	MOVW @RW1, RW4	MOVW@RW1 +d16, RW4	MOVW @RW1, RW5	MOVW@RW1 +d16, RW5	MOVW @RW1, RW6	MOVW@RW1 +d16, RW6	MOVW @RW1, RW7	MOVW@RW1 +d16, RW7
+A	MOVW @RW2, RW0	MOVW@RW2 +d16, RW0	MOVW @RW2, RW1	MOVW@RW2 +d16, RW1	MOVW @RW2, RW2	MOVW@RW2 +d16, RW2	MOVW @RW2, RW3	MOVW@RW2 +d16, RW3	MOVW @RW2, RW4	MOVW@RW2 +d16, RW4	MOVW @RW2, RW5	MOVW@RW2 +d16, RW5	MOVW @RW2, RW6	MOVW@RW2 +d16, RW6	MOVW @RW2, RW7	MOVW@RW2 +d16, RW7
+B	MOVW @RW3, RW0	MOVW@RW3 +d16, RW0	MOVW @RW3, RW1	MOVW@RW3 +d16, RW1	MOVW @RW3, RW2	MOVW@RW3 +d16, RW2	MOVW @RW3, RW3	MOVW@RW3 +d16, RW3	MOVW @RW3, RW4	MOVW@RW3 +d16, RW4	MOVW @RW3, RW5	MOVW@RW3 +d16, RW5	MOVW @RW3, RW6	MOVW@RW3 +d16, RW6	MOVW @RW3, RW7	MOVW@RW3 +d16, RW7
+C	MOVW @RW0+, RW0	MOVW@RW0 +RW7, RW0	MOVW @ RW0+, RW1	MOVW@RW0 +RW7, RW1	MOVW @ RW0+, RW2	MOVW@RW0 +RW7, RW2	MOVW @ RW0+, RW3	MOVW@RW0 +RW7, RW3	MOVW @ RW0+, RW4	MOVW@RW0 +RW7, RW4	MOVW @ RW0+, RW5	MOVW@RW0 +RW7, RW5	MOVW @ RW0+, RW6	MOVW@RW0 +RW7, RW6	MOVW @ RW0+, RW7	MOVW@RW0 +RW7, RW7
+D	MOVW @RW1+, RW0	MOVW@RW1 +RW7, RW0	MOVW @ RW1+, RW1	MOVW@RW1 +RW7, RW1	MOVW @ RW1+, RW2	MOVW@RW1 +RW7, RW2	MOVW @ RW1+, RW3	MOVW@RW1 +RW7, RW3	MOVW @ RW1+, RW4	MOVW@RW1 +RW7, RW4	MOVW @ RW1+, RW5	MOVW@RW1 +RW7, RW5	MOVW @ RW1+, RW6	MOVW@RW1 +RW7, RW6	MOVW @ RW1+, RW7	MOVW@RW1 +RW7, RW7
+E	MOVW @RW2+, RW0	MOVW @PC+ d16, RW0	MOVW @ RW2+, RW1	MOVW @PC+ d16, RW1	MOVW @ RW2+, RW2	MOVW @PC+ d16, RW2	MOVW @ RW2+, RW3	MOVW @PC+ d16, RW3	MOVW @ RW2+, RW4	MOVW @PC+ d16, RW4	MOVW @ RW2+, RW5	MOVW @PC+ d16, RW5	MOVW @ RW2+, RW6	MOVW @PC+ d16, RW6	MOVW @ RW2+, RW7	MOVW @PC+ d16, RW7
+F	MOVW @RW3+, RW0	MOVW addr 16, RW0	MOVW @ RW3+, RW1	MOVW addr 16, RW1	MOVW @ RW3+, RW2	MOVW addr 16, RW2	MOVW @ RW3+, RW3	MOVW addr 16, RW3	MOVW @ RW3+, RW4	MOVW addr 16, RW4	MOVW @ RW3+, RW5	MOVW addr 16, RW5	MOVW @ RW3+, RW6	MOVW addr 16, RW6	MOVW @ RW3+, RW7	MOVW addr 16, RW7

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)



## C.13 XCH Ri, ea

Table C.13 XCH 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, @RW0+d8	XCH R0, R0, @RW0+d8	XCH R1, R0, @RW0+d8	XCH R1, R1, @RW0+d8	XCH R2, R0, @RW0+d8	XCH R2, R1, @RW0+d8	XCH R3, R0, @RW0+d8	XCH R3, R1, @RW0+d8	XCH R4, R0, @RW0+d8	XCH R4, R1, @RW0+d8	XCH R5, R0, @RW0+d8	XCH R5, R1, @RW0+d8	XCH R6, R0, @RW0+d8	XCH R6, R1, @RW0+d8	XCH R7, R0, @RW0+d8	XCH R7, R1, @RW0+d8
+1	XCH R0, R1, @RW1+d8	XCH R0, R1, @RW1+d8	XCH R1, R1, @RW1+d8	XCH R1, R2, @RW1+d8	XCH R2, R1, @RW1+d8	XCH R2, R2, @RW1+d8	XCH R3, R1, @RW1+d8	XCH R3, R2, @RW1+d8	XCH R4, R1, @RW1+d8	XCH R4, R2, @RW1+d8	XCH R5, R1, @RW1+d8	XCH R5, R2, @RW1+d8	XCH R6, R1, @RW1+d8	XCH R6, R2, @RW1+d8	XCH R7, R1, @RW1+d8	XCH R7, R2, @RW1+d8
+2	XCH R0, R2, @RW2+d8	XCH R0, R2, @RW2+d8	XCH R1, R2, @RW2+d8	XCH R1, R3, @RW2+d8	XCH R2, R2, @RW2+d8	XCH R2, R3, @RW2+d8	XCH R3, R2, @RW2+d8	XCH R3, R3, @RW2+d8	XCH R4, R2, @RW2+d8	XCH R4, R3, @RW2+d8	XCH R5, R2, @RW2+d8	XCH R5, R3, @RW2+d8	XCH R6, R2, @RW2+d8	XCH R6, R3, @RW2+d8	XCH R7, R2, @RW2+d8	XCH R7, R3, @RW2+d8
+3	XCH R0, R3, @RW3+d8	XCH R0, R3, @RW3+d8	XCH R1, R3, @RW3+d8	XCH R1, R4, @RW3+d8	XCH R2, R3, @RW3+d8	XCH R2, R4, @RW3+d8	XCH R3, R3, @RW3+d8	XCH R3, R4, @RW3+d8	XCH R4, R3, @RW3+d8	XCH R4, R4, @RW3+d8	XCH R5, R3, @RW3+d8	XCH R5, R4, @RW3+d8	XCH R6, R3, @RW3+d8	XCH R6, R4, @RW3+d8	XCH R7, R3, @RW3+d8	XCH R7, R4, @RW3+d8
+4	XCH R0, R4, @RW4+d8	XCH R0, R4, @RW4+d8	XCH R1, R4, @RW4+d8	XCH R1, R5, @RW4+d8	XCH R2, R4, @RW4+d8	XCH R2, R5, @RW4+d8	XCH R3, R4, @RW4+d8	XCH R3, R5, @RW4+d8	XCH R4, R4, @RW4+d8	XCH R4, R5, @RW4+d8	XCH R5, R4, @RW4+d8	XCH R5, R5, @RW4+d8	XCH R6, R4, @RW4+d8	XCH R6, R5, @RW4+d8	XCH R7, R4, @RW4+d8	XCH R7, R5, @RW4+d8
+5	XCH R0, R5, @RW5+d8	XCH R0, R5, @RW5+d8	XCH R1, R5, @RW5+d8	XCH R1, R6, @RW5+d8	XCH R2, R5, @RW5+d8	XCH R2, R6, @RW5+d8	XCH R3, R5, @RW5+d8	XCH R3, R6, @RW5+d8	XCH R4, R5, @RW5+d8	XCH R4, R6, @RW5+d8	XCH R5, R5, @RW5+d8	XCH R5, R6, @RW5+d8	XCH R6, R5, @RW5+d8	XCH R6, R6, @RW5+d8	XCH R7, R5, @RW5+d8	XCH R7, R6, @RW5+d8
+6	XCH R0, R6, @RW6+d8	XCH R0, R6, @RW6+d8	XCH R1, R6, @RW6+d8	XCH R1, R7, @RW6+d8	XCH R2, R6, @RW6+d8	XCH R2, R7, @RW6+d8	XCH R3, R6, @RW6+d8	XCH R3, R7, @RW6+d8	XCH R4, R6, @RW6+d8	XCH R4, R7, @RW6+d8	XCH R5, R6, @RW6+d8	XCH R5, R7, @RW6+d8	XCH R6, R6, @RW6+d8	XCH R6, R7, @RW6+d8	XCH R7, R6, @RW6+d8	XCH R7, R7, @RW6+d8
+7	XCH R0, R7, @RW7+d8	XCH R0, R7, @RW7+d8	XCH R1, R7, @RW7+d8	XCH R1, @RW0, @RW0+d16	XCH R2, R7, @RW7+d8	XCH R2, @RW0, @RW0+d16	XCH R3, R7, @RW7+d8	XCH R3, @RW0, @RW0+d16	XCH R4, R7, @RW7+d8	XCH R4, @RW0, @RW0+d16	XCH R5, R7, @RW7+d8	XCH R5, @RW0, @RW0+d16	XCH R6, R7, @RW7+d8	XCH R6, @RW0, @RW0+d16	XCH R7, R7, @RW7+d8	XCH R7, @RW0, @RW0+d16
+8	XCH R0, @RW1	XCH R0, @RW1+d16	XCH R1, @RW1	XCH R1, @RW1+d16	XCH R2, @RW1	XCH R2, @RW1+d16	XCH R3, @RW1	XCH R3, @RW1+d16	XCH R4, @RW1	XCH R4, @RW1+d16	XCH R5, @RW1	XCH R5, @RW1+d16	XCH R6, @RW1	XCH R6, @RW1+d16	XCH R7, @RW1	XCH R7, @RW1+d16
+A	XCH R0, @RW2	XCH R0, @RW2+d16	XCH R1, @RW2	XCH R1, @RW2+d16	XCH R2, @RW2	XCH R2, @RW2+d16	XCH R3, @RW2	XCH R3, @RW2+d16	XCH R4, @RW2	XCH R4, @RW2+d16	XCH R5, @RW2	XCH R5, @RW2+d16	XCH R6, @RW2	XCH R6, @RW2+d16	XCH R7, @RW2	XCH R7, @RW2+d16
+B	XCH R0, @RW3	XCH R0, @RW3+d16	XCH R1, @RW3	XCH R1, @RW3+d16	XCH R2, @RW3	XCH R2, @RW3+d16	XCH R3, @RW3	XCH R3, @RW3+d16	XCH R4, @RW3	XCH R4, @RW3+d16	XCH R5, @RW3	XCH R5, @RW3+d16	XCH R6, @RW3	XCH R6, @RW3+d16	XCH R7, @RW3	XCH R7, @RW3+d16
+C	XCH R0, @RW0+	XCH R0, @RW0+RW7	XCH R1, @RW0+	XCH R1, @RW0+RW7	XCH R2, @RW0+	XCH R2, @RW0+RW7	XCH R3, @RW0+	XCH R3, @RW0+RW7	XCH R4, @RW0+	XCH R4, @RW0+RW7	XCH R5, @RW0+	XCH R5, @RW0+RW7	XCH R6, @RW0+	XCH R6, @RW0+RW7	XCH R7, @RW0+	XCH R7, @RW0+RW7
+D	XCH R0, @RW1+	XCH R0, @RW1+RW7	XCH R1, @RW1+	XCH R1, @RW1+RW7	XCH R2, @RW1+	XCH R2, @RW1+RW7	XCH R3, @RW1+	XCH R3, @RW1+RW7	XCH R4, @RW1+	XCH R4, @RW1+RW7	XCH R5, @RW1+	XCH R5, @RW1+RW7	XCH R6, @RW1+	XCH R6, @RW1+RW7	XCH R7, @RW1+	XCH R7, @RW1+RW7
+E	XCH R0, @RW2+	XCH R0, @RW2+PC+d16	XCH R1, @RW2+	XCH R1, @RW2+PC+d16	XCH R2, @RW2+	XCH R2, @RW2+PC+d16	XCH R3, @RW2+	XCH R3, @RW2+PC+d16	XCH R4, @RW2+	XCH R4, @RW2+PC+d16	XCH R5, @RW2+	XCH R5, @RW2+PC+d16	XCH R6, @RW2+	XCH R6, @RW2+PC+d16	XCH R7, @RW2+	XCH R7, @RW2+PC+d16
+F	XCH R0, @RW3+	XCH R0, @RW3+addr16	XCH R1, @RW3+	XCH R1, @RW3+addr16	XCH R2, @RW3+	XCH R2, @RW3+addr16	XCH R3, @RW3+	XCH R3, @RW3+addr16	XCH R4, @RW3+	XCH R4, @RW3+addr16	XCH R5, @RW3+	XCH R5, @RW3+addr16	XCH R6, @RW3+	XCH R6, @RW3+addr16	XCH R7, @RW3+	XCH R7, @RW3+addr16

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)

# C.14 XCHW RWi, ea

Table C.14 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 R W0, @RW0+	XCHW RW0 @RW0+RW7	XCHW R W1, @RW0+	XCHW RW1 @RW0+RW7	XCHW R W2, @RW0+	XCHW RW2 @RW0+RW7	XCHW R W3, @RW0+	XCHW RW3 @RW0+RW7	XCHW R W4, @RW0+	XCHW RW4 @RW0+RW7	XCHW R W5, @RW0+	XCHW RW5 @RW0+RW7	XCHW R W6, @RW0+	XCHW RW6 @RW0+RW7	XCHW R W7, @RW0+	XCHW RW7 @RW0+RW7
+D	XCHW R W0, @RW1+	XCHW RW0 @RW1+RW7	XCHW R W1, @RW1+	XCHW RW1 @RW1+RW7	XCHW R W2, @RW1+	XCHW RW2 @RW1+RW7	XCHW R W3, @RW1+	XCHW RW3 @RW1+RW7	XCHW R W4, @RW1+	XCHW RW4 @RW1+RW7	XCHW R W5, @RW1+	XCHW RW5 @RW1+RW7	XCHW R W6, @RW1+	XCHW RW6 @RW1+RW7	XCHW R W7, @RW1+	XCHW RW7 @RW1+RW7
+E	XCHW R W0, @RW2+	XCHW RW0 @PC+d16	XCHW R W1, @RW2+	XCHW RW1 @PC+d16	XCHW R W2, @RW2+	XCHW RW2 @PC+d16	XCHW R W3, @RW2+	XCHW RW3 @PC+d16	XCHW R W4, @RW2+	XCHW RW4 @PC+d16	XCHW R W5, @RW2+	XCHW RW5 @PC+d16	XCHW R W6, @RW2+	XCHW RW6 @PC+d16	XCHW R W7, @RW2+	XCHW RW7 @PC+d16
+F	XCHW R W0, @RW3+	XCHW RW0 addr16	XCHW R W1, @RW3+	XCHW RW1 addr16	XCHW R W2, @RW3+	XCHW RW2 addr16	XCHW R W3, @RW3+	XCHW RW3 addr16	XCHW R W4, @RW3+	XCHW RW4 addr16	XCHW R W5, @RW3+	XCHW RW5 addr16	XCHW R W6, @RW3+	XCHW RW6 addr16	XCHW R W7, @RW3+	XCHW RW7 addr16

Note: The following lists the correspondence between the symbols in the above table and the symbols in the instruction set tables.

- d8 : disp8 (8-bit displacement)
- d16: disp16 (16-bit displacement)