FUJITSU SEMICONDUCTOR
CONTROLLER MANUAL

CM44-10105-2E

F^2MC-16LX FAMILY

16-BIT MICROCONTROLLER

MB90590 SERIES HARDWARE MANUAL

FUJITSU

F^2MC-16LX FAMILY

16-BIT MICROCONTROLLER

MB90590 SERIES

HARDWARE MANUAL

FUJITSU LIMITED

PREFACE

■ Objectives and Intended Reader

Thank you very much for your continued patronage of Fujitsu semiconductor products.

The MB90590 series has been developed as a general-purpose version of the $F^2MC^{®}$-16LX series, which is an original 16-bit single-chip microcontroller compatible with the Application Specific IC (ASIC).

This manual explains the functions and operation of the MB90590 series for designers who actually use the MB90590 series to design products. Read this manual first.

■ Trademarks

F^2MC stands for FUJITSU Flexible Microcontroller and is a registered trademark of Fujitsu Limited.

■ Structure of This Manual

Chapter 1 Overview

Chapter 1 explains the advantages and basic specifications of the MB90590 series.

Chapter 2 CPU

Chapter 2 explains the memory layout of the MB90590 series.

Chapter 3 Interrupts

Chapter 3 explains the interrupt functions of the MB90590 series and also explains the functions and operation of the extended intelligent I/O service (EI^2OS).

Chapter 4 Delayed Interrupts

Chapter 4 explains the delayed interrupt functions and operation.

Chapter 5 Clock and Reset

Chapter 5 explains the clock and reset functions and operation.

Chapter 6 Low-Power Control Circuit

Chapter 6 explains the functions and operation of the low-power control circuit.

Chapter 7 Memory Access Modes

Chapter 7 explains the functions and operation of the memory access modes.

Chapter 8 I/O Ports

Chapter 8 explains the I/O port functions and operation.

Chapter 9 Time Base Timer

This chapter explains the time base timer functions and operation.

Chapter 10 Watchdog Timer

Chapter 10 explains the watchdog timer functions and operation.

Chapter 11 16-bit I/O Timer

Chapter 11 explains the 16-bit I/O timer functions and operation.

Chapter 12 16-bit Reload Timer (with Event Count Function)

Chapter 12 explains functions and operation of the 16-bit reload timer (with the event count function).

Chapter 13 Time Base Timer

Chapter 13 explains the time base timer functions and operation.

Chapter 14 8/16-bit PPG

Chapter 14 explains the 8/16-bit PPG functions and operation.

Chapter 15 DTP/External Interrupts

Chapter 15 explains the DTP/external interrupt functions and operation.

Chapter 16 A/D Converter

Chapter 16 explains the A/D converter functions and operation.

Chapter 17 UART0

Chapter 17 explains the UART0 functions and operation.

Chapter 18 Serial I/O

Chapter 18 explains the serial I/O functions and operation.

Chapter 19 CAN Controller

Chapter 19 explains the CAN controller functions and operation.

Chapter 20 Stepping Motor Controller

Chapter 20 explains the functions and operation of the stepping motor controller.

Chapter 21 Sound Generator

Chapter 21 explains sound generator functions and operation.

Chapter 22 ROM Correction

Chapter 22 explains the ROM correction functions and operation.

Chapter 23 ROM Mirroring Function Selection Module

Chapter 23 explains the functions and operation of the ROM mirroring function selection module.

Chapter 24 Two-megabit Flash Memory

Chapter 24 explains the functions and operation of the 2-megabit flash memory.

Chapter 25 Example of Connection F^2MC-16LX MB90F594A for Serial Writing

Chapter 25 explains the connection of F^2MC-16LX MB90F594A for serial writing, using an example.

Appendix

Appendix explains instructions, provides I/O maps and timing diagrams in flash memory mode, and lists MB90590 interrupt vectors.

1. The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.
2. The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended to be incorporated in devices for actual use. Also, FUJITSU is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.
3. The contents of this document may not be reproduced or copied without the permission of FUJITSU LIMITED.
4. FUJITSU semiconductor devices are intended for use in standard applications (computers, office automation and other office equipments, industrial, communications, and measurement equipments, personal or household devices, etc.).
   CAUTION:
   Customers considering the use of our products in special applications where failure or abnormal operation may directly affect human lives or cause physical injury or property damage, or where extremely high levels of reliability are demanded (such as aerospace systems, atomic energy controls, sea floor repeaters, vehicle operating controls, medical devices for life support, etc.) are requested to consult with FUJITSU sales representatives before such use. The company will not be responsible for damages arising from such use without prior approval.
5. Any semiconductor devices have inherently a certain rate of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
6. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Control Law of Japan, the prior authorization by Japanese government should be required for export of those products from Japan.

©1999 FUJITSU LIMITED Printed in Japan

READING THIS MANUAL

■ Page organization

Each section in this document contains a summary of the section. Reading only the summaries will give you an overview of the product.

In addition, the title of the section is also appears in subsections so that you always know which section you are reading.

CONTENT

CHAPTER 1 OVERVIEW 1

1.1 Product Overview 2
1.2 Features 3
1.3 Block Diagram 5
1.4 Pin Assignment 6
1.5 Package Dimensions 7
1.6 Pin Functions 8
1.7 Input-Output Circuits 12
1.8 Handling the Device 15

CHAPTER 2 CPU 17

2.1 Outline of CPU 18
2.2 Memory Space 19
2.3 Memory Space Map 20
2.4 Linear Addressing 21
2.5 Bank Addressing Types 22
2.6 Multi-byte Data in Memory Space 24
2.7 Registers 25
2.7.1 Accumulator (A) 27
2.7.2 User Stack Pointer (USP) and System Stack Pointer (SSP) 28
2.7.3 Processor Status (PS) 29
2.7.4 Program Counter (PC) 32
2.8 Register Bank 33
2.9 Prefix Codes 35
2.10 Interrupt Disable Instructions 37

CHAPTER 3 INTERRUPTS 39

3.1 Outline of Interrupts 40
3.2 Interrupt Vector 43
3.3 Interrupt Control Registers (ICR) 44
3.4 Interrupt Flow 47
3.5 Hardware Interrupts 49
3.5.1 Hardware Interrupt Operation 50
3.5.2 Occurrence and Release of Hardware Interrupt 51
3.5.3 Multiple interrupts 53
3.6 Software Interrupts 54
3.7 Extended Intelligent I/O Service (EI^{2}OS) 56
3.7.1 Extended Intelligent I/O Service Descriptor (ISD) 58
3.7.2 EI^{2}OS Status Register (ISCS) 60
3.8 Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI^{2}OS) 61
3.9 Exceptions 64

CHAPTER 4 DELAYD INTERRUPT 65

4.1 Outline of Delayed Interrupt Module 66

4.2 Delayed Interrupt Register 67
4.3 Delayed Interrupt Operation 68

CHAPTER 5 CLOCK AND RESET 69
5.1 Clock Generator 70
5.2 Reset Cause Occurrence 71
5.3 Reset Causes 73

CHAPTER 6 LOW-POWER CONTORL CIRCUIT 75
6.1 Outline of Low-Power Control Circuit 76
6.2 Registers 78
6.2.1 Low Power Mode Control Register (LPMCR) 79
6.2.2 Clock Selection Register (CKSCR) 81
6.3 Low-Power Mode Operation 83
6.3.1 Sleep Mode 85
6.3.2 Timer Mode 86
6.3.3 Stop Mode 87
6.3.4 Hardware Standby Mode 89
6.4 Intermittent CPU Operation 90
6.5 Switching Machine Clocks 91
6.6 Status Transition of Clock Selection 92

CHAPTER 7 MEMORY ACCESS MODES 93
7.1 Outline of Memory Access Modes 94
7.2 Mode Pins 95
7.3 Mode Data 96

CHAPTER 8 I/O PORTS 99
8.1 I/O Ports 100
8.2 I/O Port Registers 101
8.2.1 Port Data Register 102
8.2.2 Port Direction Register 103
8.2.3 Analog Input Enable Register 104

CHAPTER 9 TIME BASE TIMER 105
9.1 Outline of Time Base Timer 106
9.2 Time Base Timer Control Register 107
9.3 Operations of Time Base Timer 109

CHAPTER 10 WATCH-DOG TIMER 111
10.1 Outline of Watch-Dog Timer 112
10.2 Watch-dog Timer Operation 115

CHAPTER 11 16-BIT I/O TIMER 117
11.1 Outline of 16-Bit I/O Timer 118
11.2 16-Bit I/O Timer Registers 120
11.3 16-bit Free-running Timer 121
11.3.1 Data Register 122

11.3.2 Control Status Register 123
11.3.3 16-bit Free-running TimerOperation 125
11.4 Output Compare 127
11.4.1 Output Compare Register 128
11.4.2 Control Status Register of Output Compare 129
11.4.3 16-bit Output Compare Operation 132
11.5 Input Capture 135
11.5.1 Input Capture Register Details 136
11.5.2 16-bit Input Capture Operation 138

CHAPTER 12 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION) 141
12.1 Outline of 16-Bit Reload Timer (with Event Count Function) 142
12.2 16-Bit Reload Timer (with Event Count Function) 144
12.2.1 Timer Control Status Register (TMCSR) 145
12.2.2 Register Layout of 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR) 148
12.3 Internal Clock and External Clock Operations of 16-bit Reload Timer 149
12.4 Underflow Operation of 16-bit Reload Timer 152
12.5 Output Pin Functions of 16-bit Reload Timer 153
12.6 Counter Operation State 154

CHAPTER 13 WATCH-DOG TIMER 155
13.1 Outline of Watch Timer 156
13.2 Watch-dog Timer Registers 157
13.2.1 Timer Control Register 158
13.2.2 Sub-second Registers 160
13.2.3 Second/Minute/Hour Registers 161

CHAPTER 14 8/16-BIT PPG 163
14.1 Outline of 8/16-bit PPG 164
14.2 Block Diagram of 8/16-Bit PPG 165
14.3 8/16-Bit PPG Registers 167
14.3.1 PPG0 Operation Mode Control Register (PPGC0) 168
14.3.2 PPG1 Operation Mode Control Register (PPGC1) 170
14.3.3 PPG0, 1 Output Control Register (PPG01) 172
14.3.4 Reload Register (PRLL/PRLH) 174
14.4 Operations of 8/16-bit PPG 175
14.5 Selecting a Count Clock for 8/16-Bit PPG 177
14.6 Controlling Pin Output of 8/16-bit PPG Pulses 178
14.7 8/16-bit PPG Interrupts 179
14.8 Initial Values of 8/16-bit PPG Hardware 180

CHAPTER 15 DTP/EXTERNAL INTERRUPTS 183
15.1 Outline of DTP/External Interrupts 184
15.2 DTP/External Interrupt Registers 186
15.3 Operations of DTP/External Interrupts 188
15.4 Switching between External Interrupt and DTP Requests 190
15.5 Notes on Using DTP/External Interrupts 191

CHAPTER 16 A/D Converter 193
16.1 Features of A/D Converter 194
16.2 Block Diagram of A/D Converter 196
16.3 A/D Converter Registers 197
16.3.1 Control Status Registers (ADCS0) 198
16.3.2 Control Status Register (ADCS1) 201
16.3.3 Data Registers (ADCR1 and ADCR0) 204
16.4 Operations of A/D Converter 206
16.5 Conversion Using EI^{2}OS 208
16.5.1 Starting EI^{2}OS in Single Mode 209
16.5.2 Starting EI^{2}OS in Continuous Mode 211
16.5.3 Starting EI^{2}OS in Stop Mode 213
16.6 Conversion Data Protection 215

CHAPTER 17 UART0 217
17.1 Feature of UART0 218
17.2 UART Block Diagram 219
17.3 UART Registers 220
17.3.1 Serial Mode Control Register (UMC) 221
17.3.2 Status Register (USR) 223
17.3.3 Input Data Register (UIDR) and Output Data Register (UODR) 225
17.3.4 Rate and Data Register (URD) 226
17.4 UART0 Operation 228
17.5 Baud Rate 229
17.6 Internal and External Clock 232
17.7 Transfer Data Format 233
17.8 Parity Bit 234
17.9 Interrupt Generation and Flag Set Timings 235
17.9.1 Flag Set Timings for a Receive Operation (in Mode 0, 1, or 3) 236
17.9.2 Flag set timings for a receive operation (in mode 2) 237
17.9.3 Flag set Timings for a Transmit Operation 238
17.9.4 Status Flag During Transmit and Receive Operation 239
17.10 UART0 Application Example 241

CHAPTER 18 SERIAL I/O 243
18.1 Outline of Serial I/O 244
18.2 Serial I/O Registers 245
18.2.1 Serial Mode Control Status Register (SMCS) 246
18.2.2 Serial Shift Data Register (SDR) 250
18.3 Serial I/O Prescaler (CDCR) 251
18.4 Serial I/O Operation 252
18.4.1 Shift Clock 253
18.4.2 Serial I/O Operation 254
18.4.3 Shift Operation Start/Stop Timing 256
18.4.4 Interrupt Function of the Extended Serial I/O Interface 258
18.5 Negative Clock Operation 259

CHAPTER 19 CAN CONTROLLER 261

19.1 Features of CAN Controller 262
19.2 Block Diagram of CAN Controller 263
19.3 List of Overall Control Registers 264
19.4 List of Message Buffers (ID Registers) 266
19.5 List of Message Buffers (DLC Registers and Data Registers) 269
19.6 Classifying the CAN Controller Registers 273
19.6.1 Control Status Register (CSR) 274
19.6.2 Bus Operation Stop Bit (HALT = 1) 277
19.6.3 Last Event Indicator Register (LEIR) 278
19.6.4 Receive and Transmit Error Counters (RTEC) 280
19.6.5 Bit Timing Register (BTR) 281
19.6.6 Message Buffer Valid Register (BVALR) 283
19.6.7 IDE register (IDER) 284
19.6.8 Transmission Request Register (TREQR) 285
19.6.9 Transmission RTR Register (TRTRR) 286
19.6.10 Remote Frame Receiving Wait Register (RFWTR) 287
19.6.11 Transmission Cancel Register (TCANR) 288
19.6.12 Transmission Complete Register (TCR) 289
19.6.13 Transmission Interrupt Enable Register (TIER) 290
19.6.14 Reception Complete Register (RCR) 291
19.6.15 Remote Request Receiving Register (RRTRR) 292
19.6.16 Receive Overrun Register (ROVRR) 293
19.6.17 Reception Interrupt Enable Register (RIER) 294
19.6.18 Acceptance Mask Select Register (AMSR) 295
19.6.19 Acceptance Mask Registers 0 and 1 (AMR0 and AMR1) 297
19.6.20 Message Buffers 299
19.6.21 ID Register x (x = 0 to 15) (IDRx) 300
19.6.22 DLC Register x (x = 0 to 15) (DLCRx) 302
19.6.23 Data Register x (x = 0 to 15) (DTRx) 303
19.7 Transmission of CAN Controller 305
19.8 Reception of CAN Controller 307
19.9 Reception Flowchart of CAN Controller 310
19.10 How to Use the CAN Controller 311
19.11 Procedure for Transmission by Message Buffer (x) 313
19.12 Procedure for Reception by Message Buffer (x) 315
19.13 Setting Configuration of Multi-level Message Buffer 317

CHAPTER 20 STEPPING MOTOR CONTROLLER 319
20.1 Outline of Stepping Motor Controller 320
20.2 Stepping Motor Controller Registers 321
20.2.1 PWM Control 0 register 322
20.2.2 PWM1&2 Compare Registers 323
20.3 PWM1&2 Select Registers 324

CHAPTER 21 SOUND GENERATOR 327
21.1 Outline of Sound Generator 328
21.2 Sound Generator Registers 329
21.2.1 Sound Control Register 330

21.2.2 Frequency Data register 332
21.2.3 Amplitude Data Register 333
21.2.4 Decrement Grade Register 334
21.2.5 Tone Count Register 335

CHAPTER 22 ROM CORRECTION 337
22.1 Outline of ROM Correction 338
22.2 Application Example of ROM Correction 340
22.2.1 Correction Example of Program Errors 341
22.2.2 Example of Correction Processing 342

CHAPTER 23 ROM MIRRORING MODULE 345
23.1 Outline of ROM Mirroring Module 346
23.2 ROM Mirroring Register (ROMM) 347

CHAPTER 24 2M/3M-BIT FLASH MEMORY 349
24.1 Outline of 2M/3M-Bit Flash Memory 350
24.2 Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory 351
24.3 Write/Erase Modes 353
24.4 Flash Memory Control Status Register (FMCS) 355
24.5 Starting the Flash Memory Automatic Algorithm 357
24.6 Confirming the Automatic Algorithm Execution State 358
24.6.1 Data Polling Flag (DQ7) 360
24.6.2 Toggle Bit Flag (DQ6) 362
24.6.3 Timing Limit Exceeded Flag (DQ5) 363
24.6.4 Sector Erase Timer Flag (DQ3) 364
24.7 Detailed Explanation of Writing to and Erasing Flash Memory 365
24.7.1 Setting The Read/Reset State 366
24.7.2 Writing Data 367
24.7.3 Erasing All Data (Erasing Chips) 369
24.7.4 Erasing Optional Data (Erasing Sectors) 370
24.7.5 Suspending Sector Erase 372
24.7.6 Restarting Sector Erase 373
24.8 Notes on using 2M-Bit Flash Memory 374
24.9 Reset Vector Address in Flash Memory 376
24.10 Example of Programming 2M/3M-Bit Flash Memory 377

CHAPTER 25 EXAMPLES OF F2MC-16LX MB90F591/MB90F594A SERIAL WRITE CONNECTION 381
25.1 Basic Configuration of F2MC-16LX MB90F591/MB90F594A Serial Write Connection 382
25.2 Example of Serial Write Connection (User Power Supply Used) 385
25.3 Example of Serial Write Connection (Power Supplied from the Writer) 387
25.4 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used) 389
25.5 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Writer) 391

APPENDIX 393
APPENDIX A I/O Maps 394
APPENDIX B INSTRUCTIONS 404
B.1 Instruction Types 405
B.2 Addressing 406
B.3 Direct Addressing 408
B.4 Indirect Addressing 413
B.5 Execution Cycle Count 419
B.6 Effective Address Field 422
B.7 How to Read the Instruction List 424
B.8 F2MC-16LX Instruction List 427
B.9 Instruction Map 438
APPENDIX C Timing Diagrams in Flash Memory Mode 460
APPENDIX D List of MB90590 Interrupt Vectors 465

INDEX 469

FIGURES

Figure 1.3-1 Block Diagram 5
Figure 1.4-1 Pin Assignment 6
Figure 1.5-1 Package Dimensions 7
Figure 1.8-1 Using External Clock 15
Figure 1.8-2 Power Supply Pins (Vcc/Vss) 16
Figure 2.2-1 Sample Relationship between F2MC-16LX System and Memory Map 19
Figure 2.3-1 Memory Space Map 20
Figure 2.4-1 Example of Linear Method (24-bit Register Operand Specification) 21
Figure 2.4-2 Example of Linear Method (32-bit Register Indirect Specification) 21
Figure 2.5-1 Physical Addresses of Each Space 23
Figure 2.6-1 Sample Allocation of Multi-byte Data in Memory 24
Figure 2.6-2 Execution of MOVW A, 080FFFFH 24
Figure 2.7-1 Special Registers 26
Figure 2.7-2 General-purpose Registers 26
Figure 2.7-3 32-bit Data Transfer 27
Figure 2.7-4 AL-AH Transfer 27
Figure 2.7-5 Stack Manipulation Instruction and Stack Pointer 28
Figure 2.7-6 Processor Status (PS) Structure 29
Figure 2.7-7 Condition Code Register (CCR) Configuration 29
Figure 2.7-8 Register Bank Pointer (RP) 30
Figure 2.7-9 Interrupt Level Register (ILM) 30
Figure 2.7-10 Program Counter 32
Figure 2.8-1 Generating a Physical address in Direct Addressing Mode 34
Figure 2.10-1 Interrupt Disable Instruction 37
Figure 2.10-2 Interrupt Disable Instructions and Prefix Codes 37
Figure 2.10-3 Consecutive Prefix Codes 37
Figure 3.1-1 Overview of Hardware Interrupts 40
Figure 3.1-2 Overview of Software Interrupts 41
Figure 3.1-3 Overview of the Extended Intelligent I/O Service (EI^2OS) 42
Figure 3.3-1 Interrupt Control Register (ICR) 44
Figure 3.4-1 Interrupt Flow 47
Figure 3.4-2 Register Saving during Interrupt Processing 48
Figure 3.5-1 Occurrence and Release of Hardware Interrup 51
Figure 3.5-2 Registers Saved in Stack 53
Figure 3.6-1 Occurrence and Release of Software Interrupt 55

Figure 3.7-1 Outline of Extended Intelligent I/O Service 57
Figure 3.7-2 Extended Intelligent I/O Service Descriptor Configuration 58
Figure 3.7-3 Data Counter Configuration 58
Figure 3.7-4 I/O Register Address Pointer Configuration 59
Figure 3.7-5 ISCS Configuration 60
Figure 3.8-1 EI^{2}OS Operation Flow 61
Figure 3.8-2 EI^{2}OS Use Flow 62
Figure 4.1-1 Block Diagram 66
Figure 4.3-1 Delayed Interrupt Issuance 68
Figure 5.1-1 Clock Generator Circuit Block Diagram 70
Figure 5.2-1 Source and Destination of Reset Vector and Mode Data 72
Figure 5.3-1 Reset Cause bit Block Diagram 74
Figure 5.3-2 WDTC (Watch-Dog Timer Control) Register 74
Figure 6.1-1 Low-power Control Circuit and Clock Generator 77
Figure 6.4-1 Intermittent CPU Operation 90
Figure 6.6-1 Status Transition of Clock Selection 92
Figure 7.3-1 Mode Data Structure 96
Figure 7.3-2 Access Areas and Physical Addresses in each Bus Mode 97
Figure 8.1-1 I/O Port Block Diagram 100
Figure 8.2-1 I/O Port Registers 101
Figure 8.2-2 Port Data Registers 102
Figure 8.2-3 Port Direction Registers 103
Figure 8.2-4 Analog Input Enable Register 104
Figure 9.1-1 Block Diagram of Time Base Timer 106
Figure 10.1-1 Watch-dog Timer Block Diagram 112
Figure 10.2-1 Watch-dog Timer Operation 115
Figure 11.1-1 Block Diagram of 16-bit I/O Timer 119
Figure 11.3-1 16-bit Free-running Timer Block Diagram 121
Figure 11.4-1 Output Compare Block Diagram 127
Figure 11.4-2 Sample of Output Waveform when Compare Registers 0 and 1 are Used 132
Figure 11.4-3 Sample of a Output Waveform with Two Compare Registers (The Initial Output Value is '0') 133
Figure 11.5-1 Input Capture Block Diagram 135
Figure 11.5-2 Sample of Input Capture Fetch Timing 138
Figure 12.1-1 Block Diagram of 16-bit Reload Timer 143
Figure 12.3-1 Activation and Operation of 16-bit Reload Timer Counter 149
Figure 12.3-2 Trigger Input Operation of 16-bit Reload Timer 150
Figure 12.3-3 Gate Input Operation of 16-bit Reload Timer 150
Figure 12.4-1 Underflow Operation of 16-bit Reload Timer 152

Figure 12.5-1 Output Pin Function of 16-bit Reload Timer (1) 153
Figure 12.5-2 Output Pin Function of 16-bit Reload Timer (2) 153
Figure 12.6-1 Counter State Transitions 154
Figure 13.1-1 Block Diagram of Watch-dog Timer 156
Figure 14.2-1 8-bit PPG ch0 Block Diagram 165
Figure 14.2-2 8-bit PPG ch1 Block Diagram 166
Figure 14.4-1 PPG Output Operation, Output Waveform 176
Figure 14.6-1 8+8 PPG Output Operation Waveform 178
Figure 14.8-1 Write Timing for 8/16-bit PPG Reload Registers (PRLL and PRLH) 180
Figure 14.8-2 PRL Write Operation Block Diagram 181
Figure 15.1-1 Block Diagram of DTP/External Interrupts 184
Figure 15.3-1 External Interrupt 188
Figure 15.3-2 Timing to Cancel the External Interrupt at the End of DTP Operation 189
Figure 15.3-3 Sample Interface to the External Peripheral 189
Figure 15.4-1 Switching Between External Interrupt and DTP Requests 190
Figure 15.5-1 Clearing the Cause Hold Circuit Upon Level Set 192
Figure 15.5-2 Interrupt Cause and Interrupt Request to the Interrupt Controller While Interrupts are Enabled 192
Figure 16.2-1 Block Diagram of A/D Converter 196
Figure 16.3-1 A/D Converter Register Configuration 197
Figure 16.5-1 A/D conversion processing flow from the start to converted data transfer (in continuous mode) 208
Figure 17.2-1 UART Block Diagram 219
Figure 17.7-1 Transfer Data Format 233
Figure 17.8-1 Serial Data with Parity Enabled 234
Figure 17.9-1 RDRF Set Timing (Mode 0, 1, or 3) 236
Figure 17.9-2 ORFE Set Timing (Mode 0, 1, or 3) 236
Figure 17.9-3 PE Set Timing (Mode 0, 1, or 3) 236
Figure 17.9-4 RDRF Set Timing (Mode 2) 237
Figure 17.9-5 ORFE Set Timing (Mode 2) 237
Figure 17.9-6 TDRE Set Timing (Mode 0) 238
Figure 17.9-7 RBF Set Timing (Mode 0) 239
Figure 17.9-8 TBF Set Timing (Mode 0) 239
Figure 17.10-1 RBF Set Timing (mode 0) 241
Figure 17.10-2 Example System Configuration Using Mode 2 241
Figure 17.10-3 Communication Flowchart for Mode 2 Operation 242
Figure 18.1-1 Extended Serial I/O Interface Block Diagram 244
Figure 18.4-1 Extended I/O Serial Interface Operation Transitions 255
Figure 18.4-2 Serial Data Register Read/write 255

Figure 18.4-3 Shift Operation Start/Stop Timing (Internal Clock) 256
Figure 18.4-4 Shift Operation Start/Stop Timing (External Clock) 256
Figure 18.4-5 Shift Operation Start/Stop Timing (External Shift Clock Mode with Instruction Shift) 257
Figure 18.4-6 Stop Timing when '1' is Written to the STOP Bit 257
Figure 18.4-7 Serial Data I/O Shift Timing 257
Figure 18.4-8 Interrupt Signal Output Timing of the Extended Serial I/O Interface 258
Figure 19.2-1 Block Diagram of CAN Controller 263
Figure 19.6-1 Node Status Transition Diagram 275
Figure 19.6-2 Bit Time Segment in CAN Specification 282
Figure 19.6-3 Bit Time Segment in CAN Controller 282
Figure 19.7-1 Transmission Flowchart of the CAN Controller 306
Figure 19.8-1 Flowchart Determining Message Buffer (x) where Received Messages Stored 308
Figure 19.9-1 Reception Flowchart of the CAN Controller 310
Figure 19.12-1 Example of Receive Interrupt Handling 316
Figure 19.13-1 Examples of Operation of Multi-level Message Buffer 318
Figure 20.1-1 Block Diagram of Stepping Motor Controllerr 320
Figure 21.1-1 Block Diagram of Sound Generator 328
Figure 21.2-1 Relationship between Tone Signal and Register Value 332
Figure 21.2-2 Relationship between Register Value and PWM Pulse 333
Figure 22.1-1 Block Diagram of ROM Correction 338
Figure 22.2-1 System Structure Example 340
Figure 22.2-2 ROM Correction Processing Example 341
Figure 22.2-3 Processing Flow of ROM Correction 342
Figure 22.2-4 ROM Correction Processing Flow Diagram 343
Figure 23.1-1 Block Diagram of ROM Mirroring Module 346
Figure 24.2-1 Block Diagram of the Entire Flash Memory 351
Figure 24.2-2 Sector Configuration of the 2M/3M-Bit Flash Memory 352
Figure 24.7-1 Example of the Flash Memory Write Procedure 368
Figure 24.7-2 Example of the Flash Memory Sector Erase Procedure 371
Figure 25.2-1 Example of Serial Write Connection for MB90F591/MB90F594A Internal Vector Modes (User Power Supply Used) 385
Figure 25.3-1 Example of Serial Write Connection for MB90F591/MB90F594A Internal Vector Modes (Power Supplied from the Writer) 387
Figure 25.4-1 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used) 389
Figure 25.5-1 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Writer) 391
Figure B.3-1 Example of immediate addressing (#imm) 408
Figure B.3-2 Example of Register Direct Addressing 409

Figure B.3-3 Example of Direct Branch Addressing (addr16) 409
Figure B.3-4 Example of Direct Branch Addressing (addr24) 409
Figure B.3-5 Example of I/O Direct Addressing (io) 410
Figure B.3-6 Example of Abbreviated Direct Addressing (dir) 410
Figure B.3-7 Example of Direct Addressing (addr16) 410
Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp) 411
Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp) 411
Figure B.3-10 Example of Direct Bit addressing (addr16:bp) 411
Figure B.3-11 Example of Vector Addressing (#vct) 412
Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3) 413
Figure B.4-2 Example of Register Indirect Addressing with Post Increment (@RWj + j = 0 to 3) 413
Figure B.4-3 Example of Register Indirect Addressing with Offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3) 414
Figure B.4-4 Example of Long Register Indirect Addressing with Offset (@RLi + disp8 i = 0 to 3) 414
Figure B.4-5 Example of Program Counter Indirect Addressing with Offset (@PC + disp16) 415
Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7) 415
Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel) 416
Figure B.4-8 Configuration of the Register List 416
Figure B.4-9 Example of Register List (rlist) 416
Figure B.4-10 Example of Accumulator Indirect Addressing (@A) 417
Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A) 417
Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear) 418
Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam) 418
Figure B.9-1 Structure of Instruction Map 438
Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map 439
Figure C-1 Timing Diagram for Read Access 460
Figure C-2 Write Data polling Read (WE control) 460
Figure C-3 Timing Diagram for Write Access (CE Control) 461
Figure C-4 Timing Diagram for Write Access (Chip Erasing/Sector Erasing) 461
Figure C-5 Timing Diagram for Data Polling 462
Figure C-6 Timing Diagram for Toggle Bit 462
Figure C-7 Timing Diagram for Output of RY/BY Signal during Writing/Erasing 462
Figure C-8 Timing Diagram for Output of RY/BY Signal at Hardware Reset 463
Figure C-9 Enable Sector Protect/Verify Sector Protect 463
Figure C-10 Temporary Sector Protect Cancellation 464

TABLES

Table 1.1-1 Product Overview 2
Table 1.2-1 MB90590 Features 3
Table 1.6-1 Pin Functions 8
Table 1.7-1 Input-output Circuits 12
Table 2.5-1 Default Space 22
Table 2.7-1 Levels Indicated by the Interrupt Level Mask (ILM) Register 30
Table 2.8-1 Register Functions 33
Table 2.8-2 Relationship between Registers 33
Table 2.9-1 Bank Select Prefix 35
Table 3.2-1 Interrupt Vectors 43
Table 3.3-1 Interrupt Level Setting Bits and Interrupt Levels 44
Table 3.3-2 ICS bits, Channel Numbers, and Descriptor Addresses 46
Table 3.3-3 S Bits and End Conditions 46
Table 3.5-1 Compensation Values for Interrupt Processing Cycle Count 52
Table 3.8-1 Execution Time when the Extended EI^{2}OS Continues 62
Table 3.8-2 Data Transfer Compensation Values for Extended EI^{2}OS Execution Time 62
Table 5.3-1 Reset Causes 73
Table 5.3-2 Reset Cause Bits 74
Table 6.2-1 CG Bit Setting 80
Table 6.2-2 WS Bit Setting 81
Table 6.2-3 CS Bit Setting 82
Table 6.3-1 Low-power mode status 83
Table 6.3-2 List of Instructions Used for Transition to Low-power Mode 84
Table 7.2-1 Mode Pins and Modes 95
Table 7.3-1 Mode Setting Bits and Functions 96
Table 7.3-2 Bus Mode Setting Bits and Functions 96
Table 7.3-3 Sample Recommended Setting of Mode Pins and Mode Data 97
Table 9.2-1 Selecting the Time Base Timer Interval 107
Table 10.1-1 Reset Cause Registers 113
Table 10.1-2 Watch-dog Timer Interval Selection Bit 113
Table 12.2-1 Clock Sources for CSL Bit Settings 145
Table 12.2-2 MOD2, 1, 0 Bit Settings (1) 146
Table 12.2-3 MOD2, 1, 0 Bit Settings (2) 146
Table 12.2-4 OUTE, RELD, and OUTL Settings 146
Table 14.4-1 Reload Operation and Pulse Output 175

Table 15.2-1 Interrupt Request Detection Factor for LBx and LAx Pins 187
Table 16.3-1 Operation Mode Setting 198
Table 17.3-1 UART Operation Modes 221
Table 17.3-2 UIDR State after Receive Completion 223
Table 17.3-3 Clock Input Selection 226
Table 17.3-4 Clock Input Selection 226
Table 17.4-1 UART0 Operating Modes 228
Table 17.5-1 Baud Rate 230
Table 17.6-1 Baud Rate and Reload Value 232
Table 18.2-1 Setting the Serial Mode Selection Bit 246
Table 18.2-2 Setting the Transfer Direction Selection Bit 246
Table 18.2-3 Setting the Serial Output Enable Bit 247
Table 18.2-4 Setting the Shift Clock Output Enable Bit 247
Table 18.2-5 Setting the Serial Shift Clock Mode 247
Table 18.2-6 Setting the Interrupt Request Enable Bit 248
Table 18.2-7 Setting the Transfer Status Bit 248
Table 18.2-8 Setting the Stop Bit 249
Table 18.3-1 Machine Clock Division Ratio 251
Table 18.5-1 Setting the NEG Bit 259
Table 19.3-1 List of Overall Control Registers 264
Table 19.4-1 List of Message Buffers (ID Registers) 266
Table 19.5-1 List of Message Buffers (DLC Registers and Data Registers) 269
Table 19.5-2 List of Message Buffers (Data Registers) 271
Table 19.6-1 Correspondence between NS1 and NS0 and Node Status 275
Table 19.6-2 Selection of Acceptance Mask 295
Table 24.3-1 Flash Memory Control Signals 353
Table 24.5-1 Command Sequence Table 357
Table 24.6-1 Bit Assignments of Hardware Sequence Flags 358
Table 24.6-2 Hardware Sequence Flag Functions 359
Table 24.6-3 Data Polling Flag State Transitions (State Change for Normal Operation) 360
Table 24.6-4 Data Polling Flag State Transitions (State Change for Abnormal Operation) 360
Table 24.6-5 Toggle Bit Flag State Transitions (State Change for Normal Operation) 362
Table 24.6-6 Toggle Bit Flag State Transitions (State Change for Abnormal Operation) 362
Table 24.6-7 Timing Limit Exceeded Flag State Transitions (State Change for Normal Operation) 363
Table 24.6-8 Timing Limit Exceeded Bit Flag State Transitions (State Change for Abnormal Operation) . 363
Table 24.6-9 Sector Erase Timer Flag State Transitions (State Change for Normal Operation) 364
Table 24.6-10 Sector Erase Timer Flag State Transitions (State Change for Abnormal Operation) 364
Table 25.1-1 Pins Used for Fujitsu Standard Serial Onboard Writing 382

Table 25.1-2 AF200 Flash Microcomputer Programmer System Configuration (Manufactured by Yokogawa Digital Computer Ltd.) 383
Table A-1 I/O Map 394
Table A-2 I/O Map (19XX Address) 399
Table B.2-1 Effective Address Field 407
Table B.3-1 Direct Addressing Registers 408
Table B.3-2 CALLV Vector List 412
Table B.5-1 Execution Cycle Counts in Each Addressing Mode 419
Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles 420
Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles 420
Table B.6-1 Effective Address Field 422
Table B.7-1 Description of Items in the Instruction List 424
Table B.7-2 Explanation on Symbols in the Instruction List 425
Table B.8-1 41 Transfer Instructions (byte) 427
Table B.8-2 38 Transfer Instructions (byte) 428
Table B.8-3 42 Addition/subtraction Instructions (byte, word, long word) 429
Table B.8-4 12 Increment/decrement Instructions (byte, word, long word) 429
Table B.8-5 11 Compare Instructions (byte, word, long word) 430
Table B.8-6 11 unsigned multiplication/division instructions (word, long word) 430
Table B.8-7 39 Logic 1 Instructions (byte, word) 431
Table B.8-8 Six Logic 2 Instructions (long word) 432
Table B.8-9 Six Sign Inversion Instructions (byte, word) 432
Table B.8-10 One Normalization Instruction (long word) 432
Table B.8-11 18 Shift Instructions (byte, word, long word) 432
Table B.8-12 31 Branch 1 Instructions 433
Table B.8-13 19 Branch 2 Instructions 434
Table B.8-14 31 28 Other Control Instructions (byte, word, long word) 435
Table B.8-15 21 Bit Operand Instructions 435
Table B.8-16 Six Accumulator Operation Instructions (byte, word) 436
Table B.8-17 Ten String Instructions 436
Table B.9-1 Example of an Instruction Code 439
Table B.9-2 Basic Page Map 440
Table B.9-3 Bit Operation Instruction Map (first byte = 6CH) 441
Table B.9-4 Character String Operation Instruction Map (first byte = 6EH) 442
Table B.9-5 2-byte Instruction Map (first byte = 6FH) 443
Table B.9-6 ea Instruction 1 (first byte = 70H) 444
Table B.9-7 ea Instruction 2 (first byte = 71H) 445
Table B.9-8 ea Instruction 3 (first byte = 72H) 446

Table B.9-9 ea Instruction 4 (first byte = 73H) 447
Table B.9-10 ea Instruction 5 (first byte = 74H) 448
Table B.9-11 ea Instruction 6 (first byte = 75H) 449
Table B.9-12 ea Instruction 7 (first byte = 76H) 450
Table B.9-13 ea Instruction 8 (first byte = 77H) 451
Table B.9-14 ea Instruction 9 (first byte = 78H) 452
Table B.9-15 MOVEA RWi, ea Instruction (first byte = 79H) 453
Table B.9-16 MOV Ri, ea Instruction (first byte = 7AH) 454
Table B.9-17 MOVW RWi, ea Instruction (first byte = 7BH) 455
Table B.9-18 MOV ea, Ri Instruction (first byte = 7CH) 456
Table B.9-19 MOVW ea, Rwi Instruction (first byte = 7DH) 457
Table B.9-20 XCH Ri, ea Instruction (first byte = 7EH) 458
Table B.9-21 XCHW RWi, ea Instruction (first byte = 7FH) 459
Table D-1 MB90590 Interrupt Vectors 465
Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers 467

CHAPTER 1 OVERVIEW

The MB90590 Series is a family menber of the F^{2}MC-16LX micro controllers.

1.1 Product Overview

1.2 Features

1.3 Block Diagram

1.4 Pin Assignment

1.5 Package Dimensions

1.6 Pin Functions

1.7 Circuit

1.8 Handling the Device

1.1 Product Overview

Table 1.1-1 provides a quick outlook of the MB90590 Series.

■ **Product Overview**

Table 1.1-1 Product Overview

Features	MB90V590A	MB90F594A/F591	MB90594/591
Product type	Evaluation sample	Flash version	Mask ROM version
CPU	F^2MC-16LX CPU		
System clock	On-chip PLL clock multiplier (x1, x2, x3, x4, 1/2 when PLL stop) Minimum instruction execution time: 62.5 ns (4 MHz osc. PLL x4)		
ROM/Flash memory	External	Boot-block Flash memory 256K/384K bytes with Hard-wired reset vector	Mask ROM 256K/384K bytes
RAM	8Kbytes	6K/8K bytes	
Technology	0.5 µm CMOS with on-chip voltage regulator for internal power supply	0.5 µm CMOS + Flash memory with on-chip voltage regulator for internal power supply and on-chip charge pump for programming voltage	0.5 µm CMOS with on-chip voltage regulator for internal power supply
Package	PGA-256	QFP100	

1.2 Features

Table 1.2-1 lists the features of the MB90590 series.

■ Features

Table 1.2-1 MB90590 Features

Function	Feature
UART (3 channels)	Full duplex double buffer Supports asynchronous/synchronous(with start/stop bit) transfer Baud rate : 4808/5208/9615/10417/19230/38460/62500/500000bps (asynchronous) 500K/1M/2Mbps (synchronous) at System clock = 16MHz
Serial IO	Transfer can be started from MSB or LSB Supports internal clock synchronized transfer and external clock synchronized transfer Supports positive-edge and negative-edge clock synchronization Baud rate : 31.25K/62.5K/125K/500K/1Mbps at System clock = 16MHz
A/D Converter	10 or 8-bit resolution 8 input channels Conversion time : 26.3us (per one channel)
16-bit Reload Timer (2 channels)	Operation clock frequency : fsys/2^1, fsys/2^3, fsys/2^5 (fsys = System clock frequency) Supports External Event Count function
Watch Timer	Directly operates with the oscillation clock Facility to correct oscillation deviation Read/Write accessible Second/Minute/Hour registers Signals interrupts
16-bit IO Timer	Signals an interrupt when overflow Supports Timer Clear when a match with Output Compare(Channel 0) Operation clock frequency : fsys/2^2, fsys/2^4,fsys/2^6, fsys/2^8(fsys = System clock frequency)
16-bit Output Compare (6 channels)	Signals an interrupt when a match with 16-bit IO Timer Six 16-bit compare registers A pair of compare registers can be used to generate an output signal
16-bit Input Capture (6 channels)	Rising edge, falling edge or rising & falling edge sensitive Six 16-bit Capture registers Signals an interrupt upon external event

Table 1.2-1 MB90590 Features (Continued)

Function	Feature
8/16-bit Programmable Pulse Generator (6 channels)	Supports 8-bit and 16-bit operation modes Twelve 8-bit reload counters Twelve 8-bit reload registers for L pulse width Twelve 8-bit reload registers for H pulse width A pair of 8-bit reload counters can be configured as one 16-bit reload counter or as 8-bit prescaler plus 8-bit reload counter 6 output pins Operation clock frequency : fsys, fsys/2^1, fsys/2^2, fsys/2^3, fsys/2^4 or 128us@fosc=4MHz (fsys = System clock frequency, fosc = Oscillation clock frequency)
CAN Interface (2 channels)	Conforms to CAN Specification Version 2.0 Part A and B Automatic re-transmission in case of error Automatic transmission responding to Remote Frame Prioritized 16 message buffers for data and ID's Supports multiple messages Flexible configuration of acceptance filtering : Full bit compare / Full bit mask / Two partial bit masks Supports up to 1Mbps
Stepping Motor Controller (4 channels)	Four high current outputs for each channel Synchronized two 8-bit PWM*s for each channel Succeeds to MB89940 design resource
External Interrupt (8 channels)	Can be programmed edge sensitive or level sensitive
Sound Generator	8-bit PWM signal is mixed with tone frequency from 8-bit reload counter PWM frequency : 62.5K, 31.2K, 15.6K, 7.8KHz at System clock = 16MHz Tone frequency : PWM frequency / 2 / (reload value + 1)
IO Ports	Virtually all external pins can be used as general purpose IO All push-pull outputs and schmitt trigger inputs Bit-wise programmable as input/output or peripheral signal
Flash Memory	Supports automatic programming, Embedded AlgorithmTM *1 Write/Erase/Erase-Suspend/Resume commands A flag indicating completion of the algorithm Hard-wired reset vector available in order to point to a fixed boot sector in Flash Memory Flash Writer from Minato Electronics Inc. Boot block configuration Erase can be performed on each block

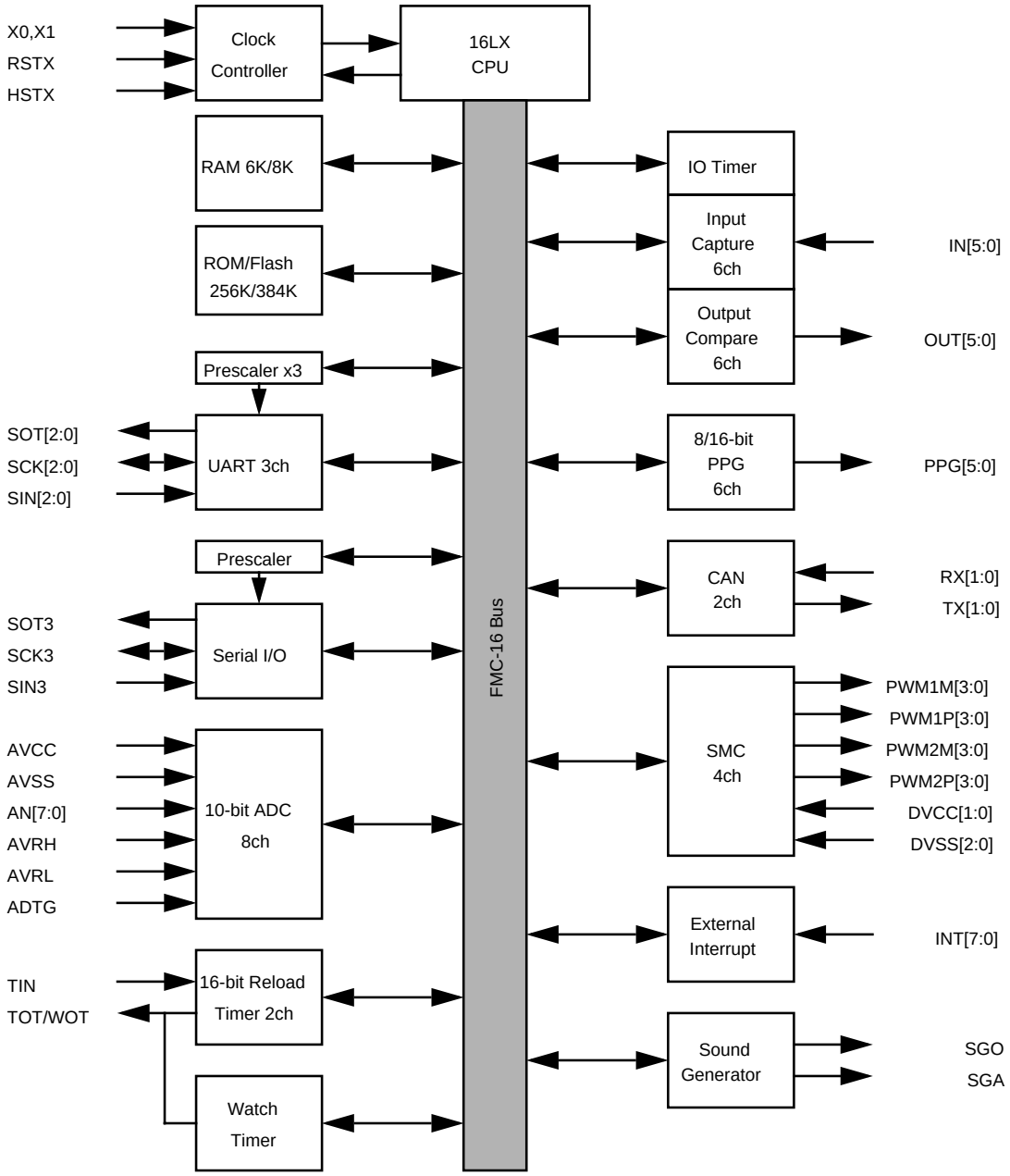
*1: Embedded Algorithm is a trade mark of Advanced Micro Devices Inc.

1.3 Block Diagram

Figure 1.3-1 shows a block diagram of the MB90590 series.

■ Block Diagram

Figure 1.3-1 Block Diagram

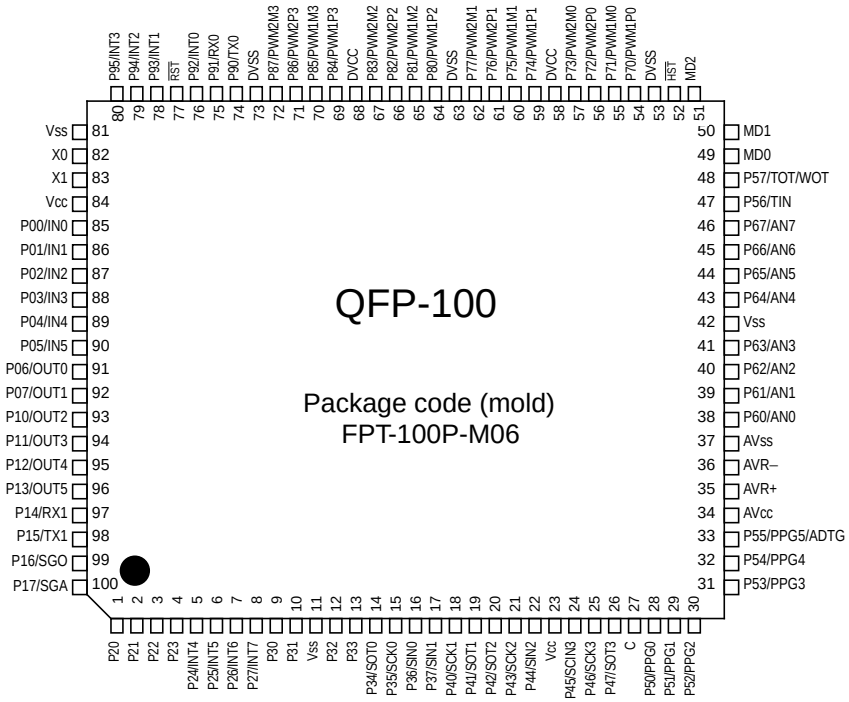


1.4 Pin Assignment

Figure 1.4-1 shows the pin assignments for the MB90590 series.

■ Pin Assignment

Figure 1.4-1 Pin Assignment



1.5 Package Dimensions

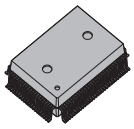
Figure 1.5-1 shows the package dimensions of the MB90590 series.

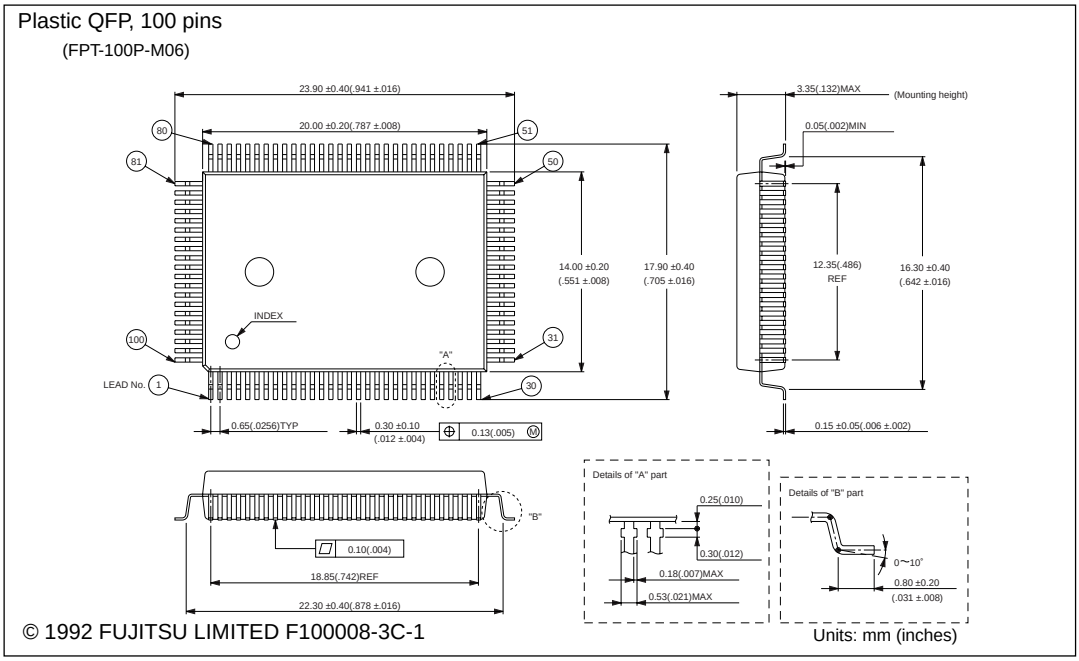
■ Package Dimensions

Figure 1.5-1 Package Dimensions

FPT-100P-M06

EIAJ Code: ∗ QFP100-P-1420-4

Plastic QFP, 100 pins  (FPT-100P-M06)	Lead pitch	0.65mm
	Package width × package length	14 × 20mm
	Lead configuration	Gull wing
	Sealing	Plastic mold
	Flat portion length	0.80mm



1.6 Pin Functions

Table 1.6-1 describes the pin functions of the MB90590 series.

■ Pin Functions

Table 1.6-1 Pin Functions

No.	Pin name	Circuit type	Function
82	X0	A	Oscillation input
83	X1		Oscillation output
77	$\overline{\text{RST}}$	B	Reset input
52	$\overline{\text{HST}}$	C	Hardware standby input
85 to 90	P00 to P05	D	General purpose IO
	IN0 to IN5		Inputs for the Input Captures
91 to 96	P06 to P07 P10 to P13	D	General purpose IO
	OUT0 to OUT5		Outputs for the Output Compares. To enable the signal outputs, the corresponding bits of the Port Direction registers should be set to "1".
97	P14	D	General purpose IO
	RX1		RX input for CAN Interface 1
98	P15	D	General purpose IO
	TX1		TX output for CAN Interface 1. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
99	P16	D	General purpose IO
	SGO		SGO output for the Sound Generator. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
100	P17	D	General purpose IO
	SGA		SGA output for the Sound Generator. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
1 to 4	P20 to P23	D	General purpose IO
5 to 8	P24 to P27	D	General purpose IO
	INT4 to INT7		External interrupt input for INT4 to INT7 These pin functions are not supported by MB90V590

Table 1.6-1 Pin Functions (Continued)

No.	Pin name	Circuit type	Function
9 to 10	P30 to P31	D	General purpose IO
12 to 13	P32 to P33	D	General purpose IO
14	P34	D	General purpose IO
	SOT0		SOT output for UART 0. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
15	P35	D	General purpose IO
	SCK0		SCK input/output for UART 0. To enable the signal output, the corresponding bit of the Port Direction register should be set to "1".
16	P36	D	General purpose IO
	SIN0		SIN input for UART 0
17	P37	D	General purpose IO
	SIN1		SIN input for UART 1
18	P40	D	General purpose IO
	SCK1		SCK input/output for UART 1
19	P41	D	General purpose IO
	SOT1		SOT output for UART 1
20	P42	D	General purpose IO
	SOT2		SOT output for UART 2
21	P43	D	General purpose IO
	SCK2		SCK input/output for UART 2
22	P44	D	General purpose IO
	SIN2		SIN input for UART 2
24	P45	D	General purpose IO
	SIN3		SIN input for the Serial IO
25	P46	D	General purpose IO
	SCK3		SCK input/output for the Serial IO
26	P47	D	General purpose IO
	SOT3		SOT output for the Serial IO
28 to 33	P50 to P55	D	General purpose IO
	PPG0 to PPG5, ADTG		Outputs for the Programmable Pulse Generators. Pin number 33 is also shared with ADTG input for the external trigger of the A/D Converter.

Table 1.6-1 Pin Functions (Continued)

No.	Pin name	Circuit type	Function
38 to 41	P60 to P63	E	General purpose IO
	AN0 to AN3		Inputs for the A/D Converter
43 to 46	P64 to P67	E	General purpose IO
	AN4 to AN7		Inputs for the A/D Converter
47	P56	D	General purpose IO
	TIN		TIN input for the 16-bit Reload Timers
48	P57	D	General purpose IO
	TOT/WOT		TOT output for the 16-bit Reload Timers and WOT output for the Watch Timer. Only one of three output enable flags in these pheripheral blocks can be set at a time. Otherwise the output signal has no meaning.
54 to 57	P70 to P73	F	General purpose IO
	PWM1P0 PWM1M0 PWM2P0 PWM2M0		Output for Stepping Motor Controller channel 0.
59 to 62	P74 to P77	F	General purpose IO
	PWM1P1 PWM1M1 PWM2P1 PWM2M1		Output for Stepping Motor Controller channel 1.
64 to 67	P80 to P83	F	General purpose IO
	PWM1P2 PWM1M2 PWM2P2 PWM2M2		Output for Stepping Motor Controller channel 2.
69 to 72	P84 to P87	F	F, GGeneral purpose IO
	PWM1P3 PWM1M3 PWM2P3 PWM2M3		Output for Stepping Motor Controller channel 3.
74	P90	D	General purpose IO
	TX0		TX output for CAN Interface 0
75	P91	D	General purpose IO
	RX0		RX input for CAN Interface 0
76	P92	D	General purpose IO
	INT0		External interrupt input for INT0

Table 1.6-1 Pin Functions (Continued)

No.	Pin name	Circuit type	Function
78	P93	D	General purpose IO
	INT1		External interrupt input for INT1
79	P94	D	General purpose IO
	INT2		External interrupt input for INT2
80	P95	D	General purpose IO
	INT3		External interrupt input for INT3
58 68	DVCC		Dedicated power supply pins for the high current output buffers (Pin No. 54 to 72)
53 63 73	DVSS		Dedicated ground pins for the high current output buffers (Pin No. 54 to 72)
34	AVCC		Dedicated power supply pin for the A/D Converter
37	AVSS		Dedicated ground pin for the A/D Converter
35	AVR+		Upper referance voltage input for the A/D Converter
36	AVR-		Lower reference voltage input for the A/D Converter
49 50	MD0 MD1	C	Test mode inputs. These pins should be connected to VCC
51	MD2	H	Test mode input. This pin should be connected to VSS
27	C		External capacitor pin. A capacitor of 0.1μF should be connected to this pin and VSS.
23 84	VCC		Power supply pins
11 42 81	VSS		Ground pins

1.7 Input-Output Circuits

Table 1.7-1 lists the input-output circuits.

■ Input-output Circuits

Table 1.7-1 Input-output Circuits

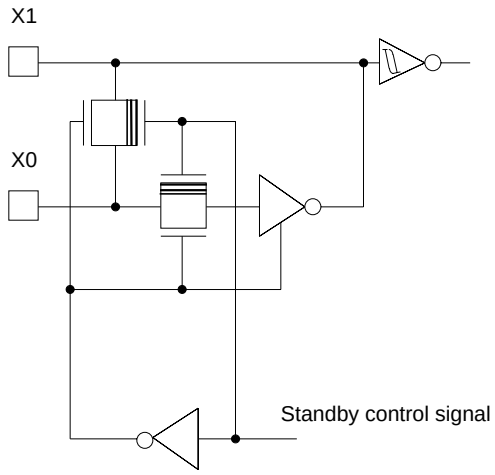
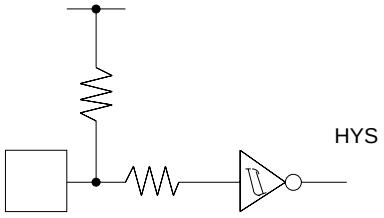
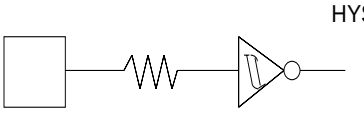
Class	Circuit	Remarks
A		• 2Oscillation feedback resistor: 1 Mohm approx.
B		• Hysteresis input with pull-up Resistor: 50 Kohm approx.
C		• Hysteresis input

Table 1.7-1 Input-output Circuits (Continued)

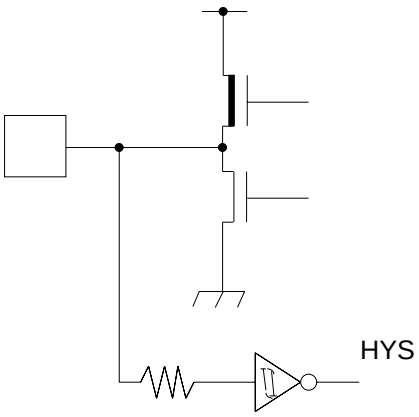
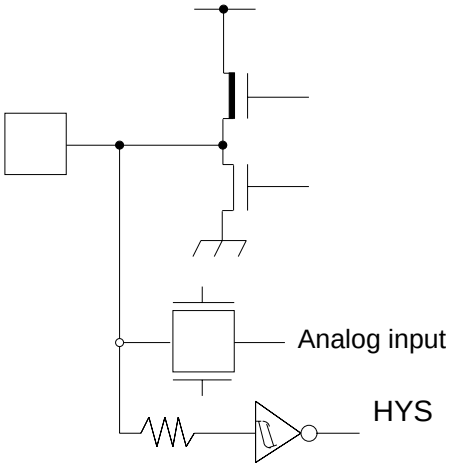
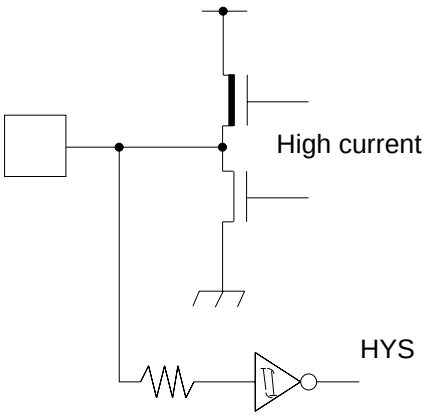
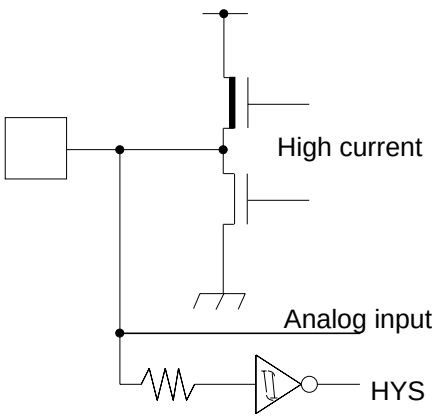
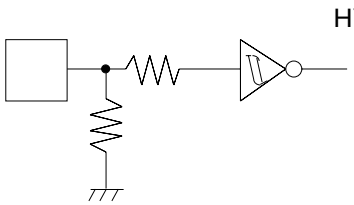
Class	Circuit	Remarks
D		• CMOS output • Hysterisis input
E		• CMOS output • Hysterisis input • Analog input
F		• CMOS high current output • Hysterisis input

Table 1.7-1 Input-output Circuits (Continued)

Class	Circuit	Remarks
G		• CMOS high current output • Hysterisis input • Analog input
H		• Hysteresis input with pull-down Resistor: 50 Kohm approx. Flash version does not have pull-down register.

1.8 Handling the Device

Special care is required for the following when handling the device:

- **Preventing latch-up**
- **Handling unused input pins**
- **Using external clock**
- **Power supply pins (Vcc/Vss)**
- **Pull-up/down resistors**

■ Handling the Device

❍ Preventing latch-up

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

- A voltage higher than Vcc or lower than Vss is applied to an input or output pin.
- A voltage higher than the rated voltage is applied between Vcc and Vss.
- The AVcc power supply is applied before the Vcc voltage.

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

❍ Handling unused input pins

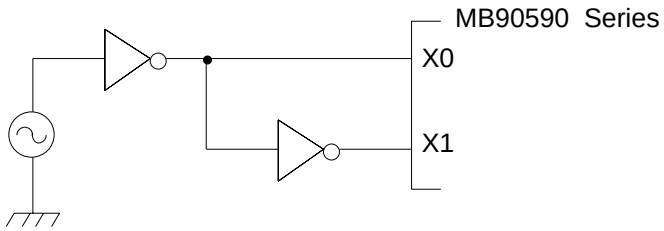
Do not leave unused input pins open, as doing so may cause misoperation of the device. Use a pull-up or pull-down resistor.

❍ Using external clock

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

Figure 1.8-1 is a diagram of how to use external clock.

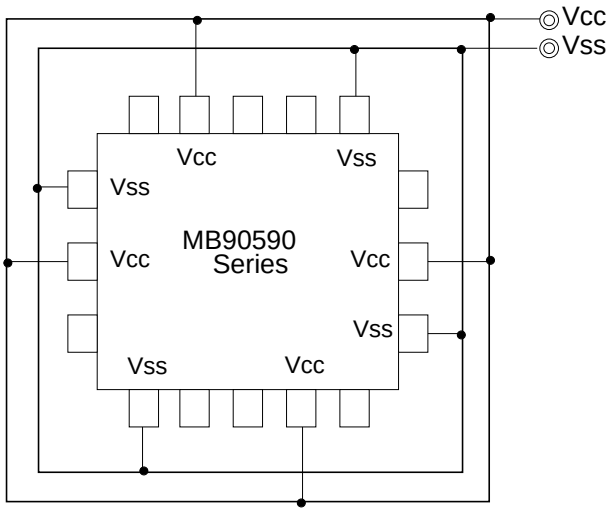
Figure 1.8-1 Using External Clock



❍ Power supply pins (Vcc/Vss)

Ensure that all Vcc-level power supply pins are at the same potential. In addition, ensure the same for all Vss-level power supply pins. (See the figure 1.8-2.) If there are more than one Vcc or Vss system, the device may operate incorrectly even within the guaranteed operating range.

Figure 1.8-2 Power Supply Pins (Vcc/Vss)



❍ **Pull-up/down resistors**

The MB90590 Series does not support internal pull-up/down resistors. Use external components where needed.

CHAPTER 2 CPU

This chapter explains the memory space of the MB90590.

2.1 Outline of CPU

2.2 Memory Space

2.3 Memory Space Map

2.4 Linear Addressing

2.5 Bank Addressing Types

2.6 Multi-byte Data in Memory Space

2.7 Registers

2.8 Register Bank

2.9 Prefix Codes

2.10 Interrupt Disable Instructions

2.1 Outline of CPU

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

■ Outline of CPU

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

- **Minimum instruction execution time: 62.5 ns (at 4-MHz oscillation, 4 times clock multiplication)**
- **Maximum memory space: 16 Mbytes, accessed in linear or bank mode**
- **Instruction set optimized for controller applications**
  - Rich data types: Bit, byte, word, long word
  - Extended addressing modes: 23 types
  - High-precision operation (32-bit length) based on 32-bit accumulator
- **Powerful interrupt functions**

  Eight priority levels (programmable)
- **CPU-independent automatic transfer**

  Up to 16 channels of the extended intelligent I/O service
- **Instruction set compatible with high-level language (C)/multitasking**

  System stack pointer/instruction set symmetry/barrel-shift instructions
- **Improved execution speed: 4-byte queue**

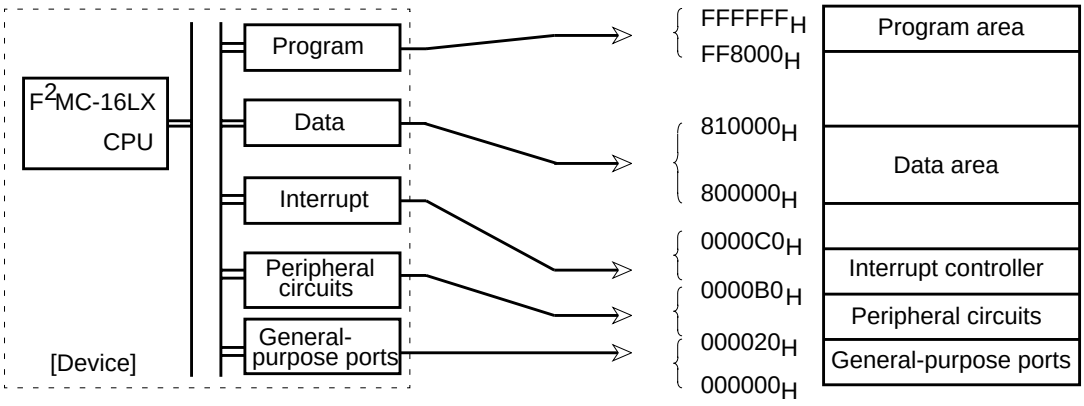
2.2 Memory Space

An F²MC-16LX CPU has a 16-Mbyte memory space. All data program input and output managed by the F²MC-16LX CPU are located in this 16-Mbyte memory space. The CPU accesses the resources by indicating their addresses using a 24-bit address bus.

■ Outline of CPU Memory Space

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

Figure 2.2-1 Sample Relationship between F²MC-16LX System and Memory Map



■ Address Generation Types

The F²MC-16LX has the following two addressing:

❍ **Linear addressing**

An entire 24-bit address is specified by an instruction.

❍ **Bank addressing**

The eight high-order bits of an address are specified by an appropriate bank register, and the remaining 16 low-order bits are specified by an instruction.

2.3 Memory Space Map

The memory space of the MB90590 Series is shown in Figure 2.3-1.

■ Memory Space Map

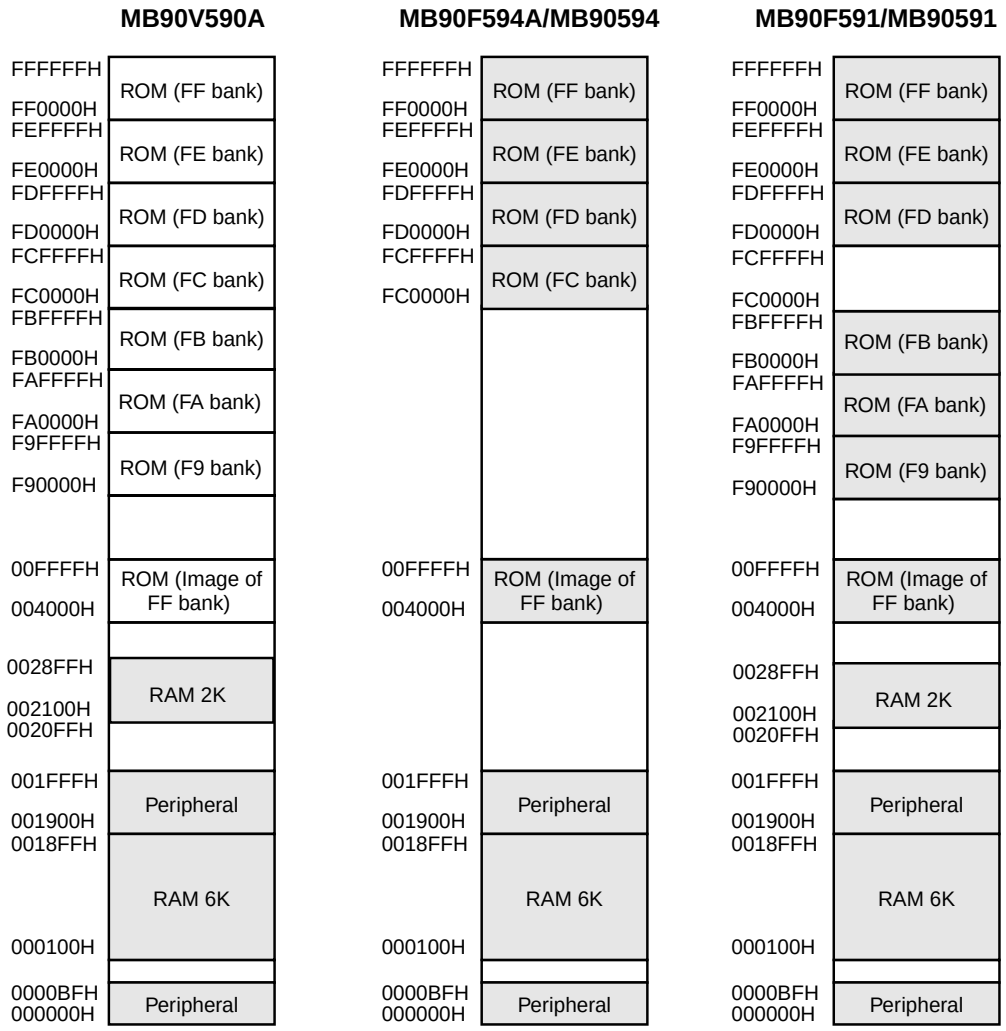
The high-order portion of bank 00 gives the image of the FF bank ROM to make the small model of the C compiler effective. Since the low-order 16 bits are the same, the table in ROM can be referenced without using the far specification in the pointer declaration.

For example, an attempt to access $00C000_H$ accesses the value at $FFC000_H$ in ROM.

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

The image between $FF4000_H$ and $FFFFFF_H$ is visible in bank 00, while the image between $FF0000_H$ and $FF3FFF_H$ is visible only in bank FF.

Figure 2.3-1 Memory Space Map



2.4 Linear Addressing

There are two types of linear addressing:

- **24-bit operand specification: Directly specifies a 24-bit address using operands.**
- **32-bit register indirect specification: Indirectly specifies the 24 low-order bits of a 32-bit general-purpose register value as the address.**

■ 24-bit Operand Specification

Figure 2.4-1 shows an example of 24-bit operand specification. Figure 2.4-2 shows an example of 32-bit register indirect specification.

Figure 2.4-1 Example of Linear Method (24-bit Register Operand Specification)

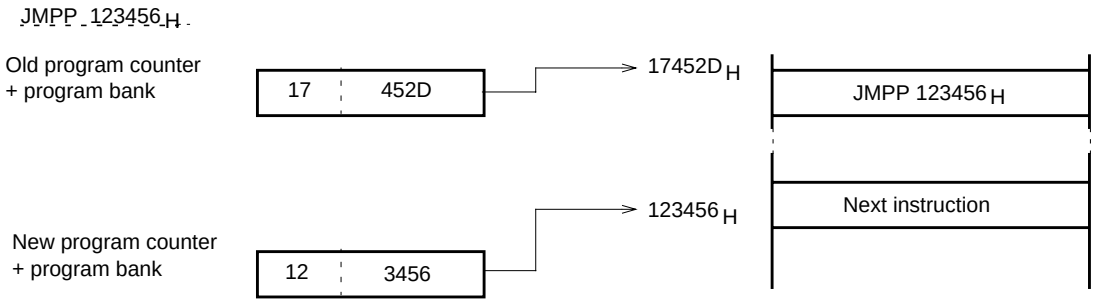
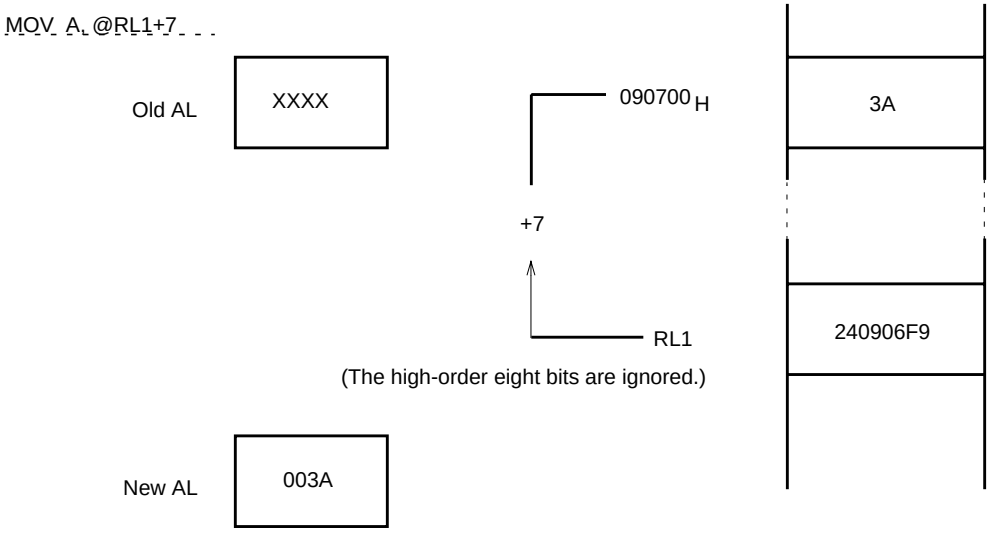


Figure 2.4-2 Example of Linear Method (32-bit Register Indirect Specification)



2.5 Bank Addressing Types

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

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

■ Bank Addressing Types

❍ Program bank register (PCB)

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

❍ Data bank register (DTB)

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

❍ User stack bank register (USB)/system stack bank register (SSB)

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

❍ Additional bank register (ADB)

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

Table 2.5-1 lists the default spaces used in each addressing mode, which are pre-determined to improve instruction coding efficiency. To use a non-default space for an addressing mode, specify a prefix code corresponding to a bank before the instruction. This enables access to the bank space corresponding to the specified prefix code.

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

Table 2.5-1 Default Space

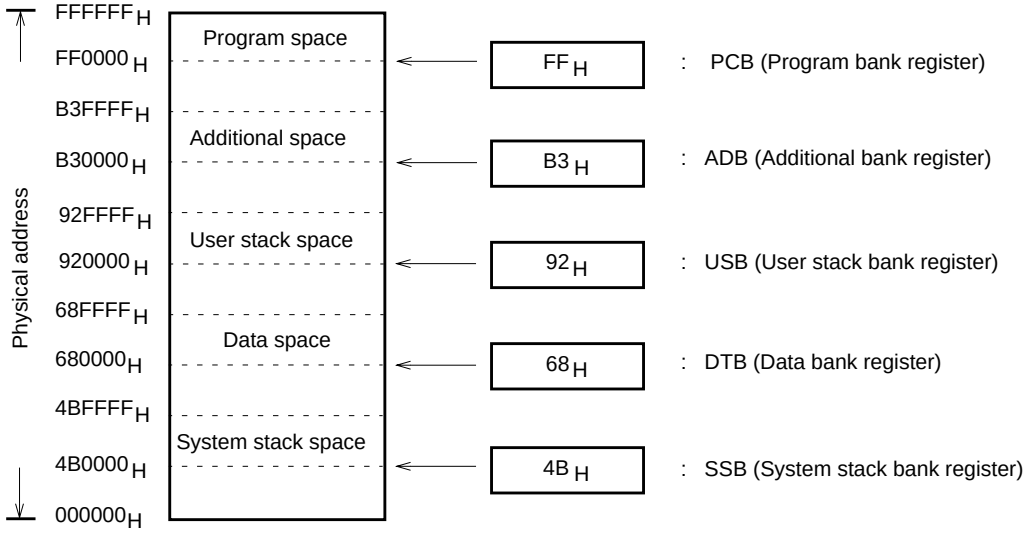
Default space	Addressing mode
Program space	PC indirect, program access, branch
Data space	Addressing mode using @RW0, @RW1, @RW4, or @RW5, @A, addr16, and dir

Table 2.5-1 Default Space (Continued)

Default space	Addressing mode
Stack space	Addressing mode using PUSHW, POPW, @RW3, or @RW7
Additional space	Addressing mode using @RW2 or @RW6

Figure 2.5-1 is an example of a memory space divided into register banks.

Figure 2.5-1 Physical Addresses of Each Space



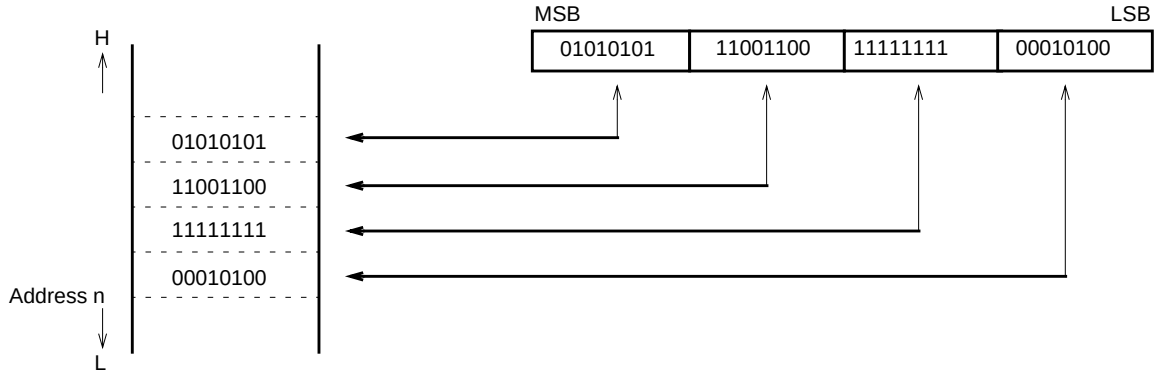
2.6 Multi-byte Data in Memory Space

Data is written to memory from the low-order addresses. Therefore, for a 32-bit data item, the low-order 16 bits are transferred before the high-order 16 bits.
If a reset signal is input immediately after the low-order bits are written, the high-order bits might not be written.

■ Multi-byte Data Allocation in Memory Space

Figure 2.6-1 is a diagram of multi-byte data configuration in memory. The low-order eight bits of a data item are stored at address n, then address n+1, address n+2, address n+3, etc.

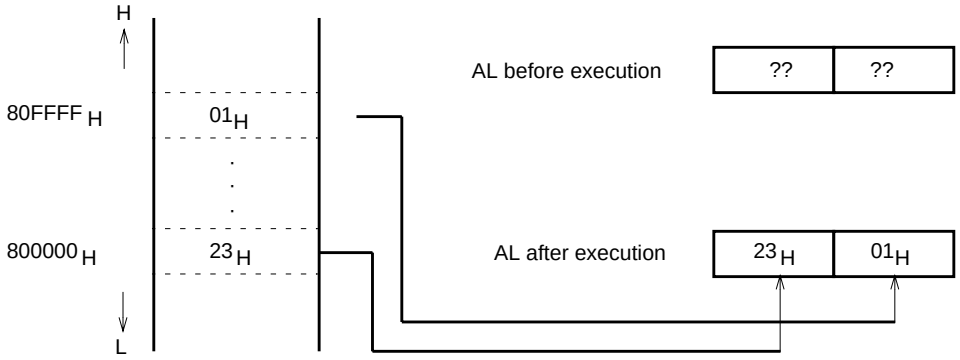
Figure 2.6-1 Sample Allocation of Multi-byte Data in Memory



■ Accessing Multi-byte Data

Fundamentally, accesses are made within a bank. For an instruction accessing a multi-byte data item, address FFFFH is followed by address 0000H of the same bank. Figure 2.6-2 is an example of an instruction accessing multi-byte data.

Figure 2.6-2 Execution of MOVW A, 080FFFFH



2.7 Registers

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

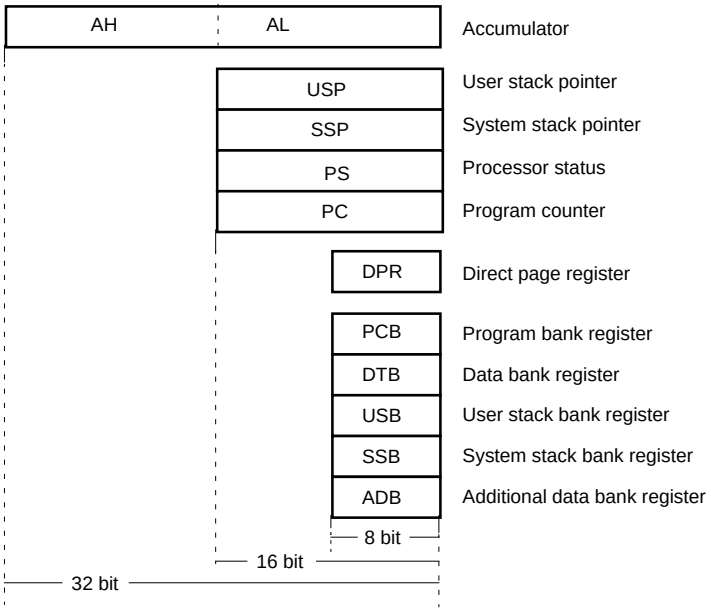
■ Special Registers

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

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

Figure 2.7-1 is a diagram of the special registers.

Figure 2.7-1 Special Registers

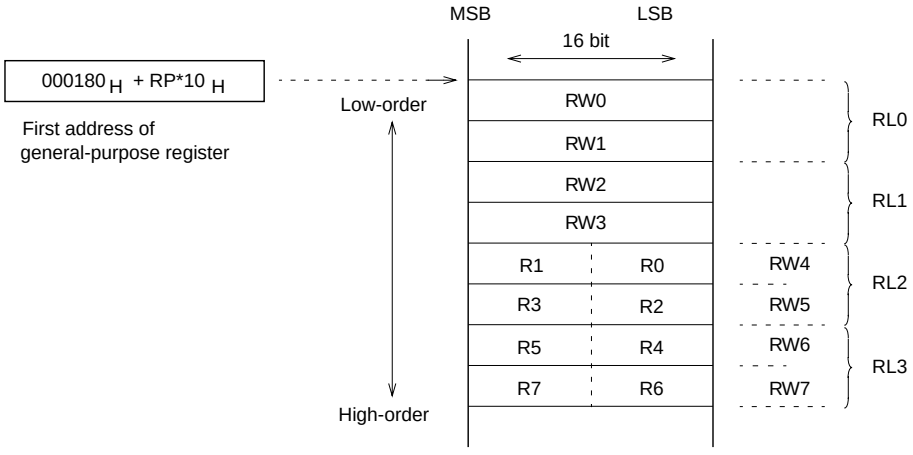


■ General-purpose Registers

The F^{2}MC-16LX general-purpose registers are located from addresses 000180H to 00037FH (maximum configuration) of main storage. The register bank pointer (RP) indicates which of the above addresses are currently being used as a register bank. Each bank has the following three types of registers. These registers are mutually dependent as described in Figure 2.7-2.

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

Figure 2.7-2 General-purpose Registers



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

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

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

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

2.7.1 Accumulator (A)

The accumulator (A) register consists of two 16-bit arithmetic operation registers (AH and AL), and is used as a temporary storage for operation results and transfer data.

■ Accumulator (A)

The A register consists of two 16-bit arithmetic operation registers (AH and AL). The A register is used as a temporary storage for operation results and transfer data. During 32-bit data processing, AH and AL are used together. Only AL is used for word processing in 16-bit data processing mode or for byte processing in 8-bit data processing mode (see Figures 2.7-3 and 2.7-4). The data stored in the A register can be operated upon with the data in memory or registers (Ri, Rwi, or Rli). In the same manner as with the F^2MC-8L, when a word or shorter data item is transferred to AL, the previous data item in AL is automatically sent to AH (data preservation function). The data preservation function and operation between AL and AH help improve processing efficiency.

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

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

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

Figure 2.7-3 32-bit Data Transfer

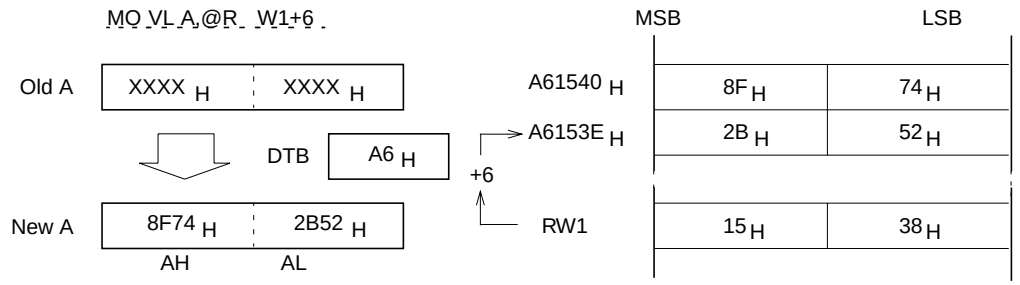
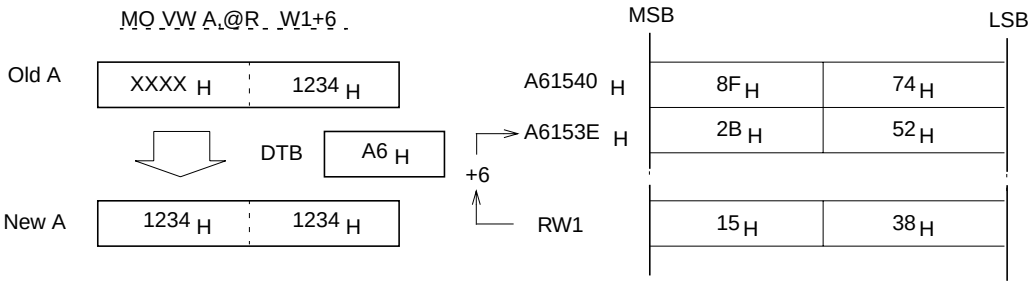


Figure 2.7-4 AL-AH Transfer



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

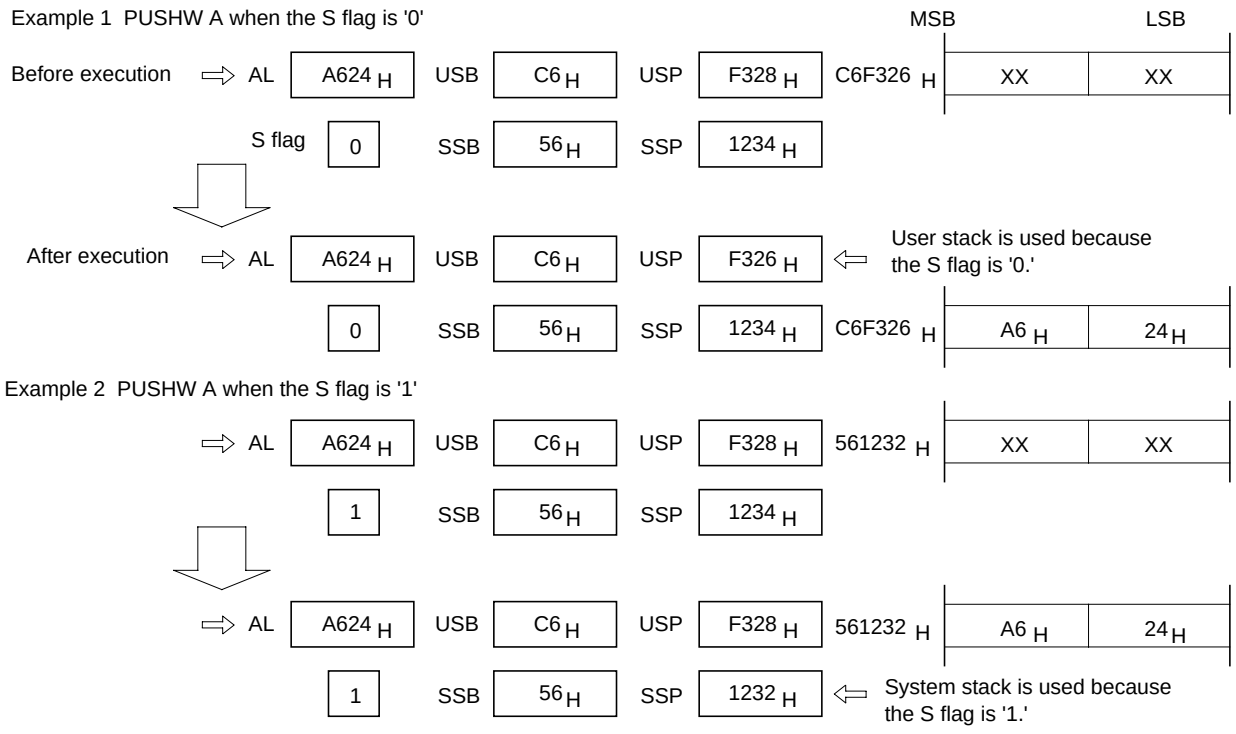
USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data when a push/pop instruction or subroutine is executed.

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

USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data in the event of a push/pop instruction or subroutine execution. The USP and SSP registers are used by stack instructions. The USP register is enabled when the S flag in the processor status register is '0,' and the SSP register is enabled when the S flag is '1' (see Figure 2.7-5). Since the S flag is set when an interrupt is accepted, register values are always saved in the memory area indicated by SSP during interrupt processing. SSP is used for stack processing in an interrupt routine, while USP is used for stack processing outside an interrupt routine. If the stack space is not divided, use only the SSP.

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

Figure 2.7-5 Stack Manipulation Instruction and Stack Pointer



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

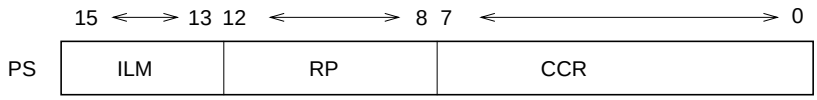
2.7.3 Processor Status (PS)

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

■ Processor Status (PS)

As shown in Figure 2.7-6, the high-order byte of the PS register consists of a register bank pointer (RP) and an interrupt level mask register (ILM). The RP indicates the start address of a register bank. The low-order byte of the PS register is a condition code register (CCR), containing the flags to be set or reset depending on the results of instruction execution or interruptoccurrences.

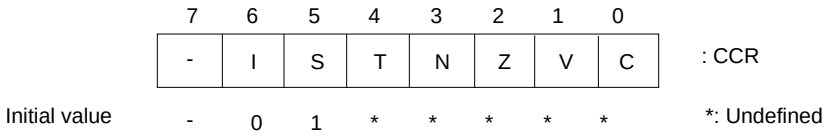
Figure 2.7-6 Processor Status (PS) Structure



■ Condition Code Register (CCR)

Figure 2.7-7 is a diagram of condition code register configuration.

Figure 2.7-7 Condition Code Register (CCR) Configuration



❍ **I: Interrupt enable flag:**

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

❍ **S: Stack flag:**

When the S flag is 0, USP is enabled as the stack manipulation pointer.

When the S flag is 1, SSP is enabled as the stack manipulation pointer.

The S flag is set by an interrupt reception or a reset.

❍ **T: Sticky bit flag:**

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

❍ **N: Negative flag:**

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

❍ **Z: Zero flag:**

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

❍ **V: Overflow flag:**

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

❍ **C: Carry flag:**

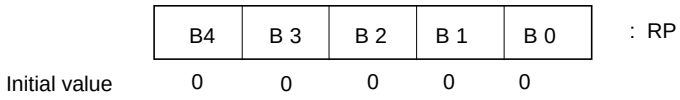
The C flag is set when a carry-up or carry-down from the MSB occurs as a result of operation execution, and is otherwise cleared.

■ Register Bank Pointer (RP)

The RP register indicates the relationship between the general-purpose registers of the F^{2}MC-16LX and the internal RAM addresses. Specifically, the RP register indicates the first memory address of the currently used register bank in the following conversion expression: [00180H + (RP)*10H] (see Figure 2.7-8). The RP register consists of five bits, and can take a value between 00H and 1FH. Register banks can be allocated at addresses from 000180H to 00037H in memory.

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

Figure 2.7-8 Register Bank Pointer (RP)



■ Interrupt Level Mask Register (ILM)

The ILM register consists of three bits, indicating the CPU interrupt masking level. An interrupt request is accepted only when the level of the interrupt is higher than that indicated by these three bits. Level 0 is the highest priority interrupt, and level 7 is the lowest priority interrupt (see Table 2.7-9). Therefore, for an interrupt to be accepted, its level value must be smaller than the current ILM value. When an interrupt is accepted, the level value of that interrupt is set in ILM. Thus, an interrupt of the same or lower level cannot be accepted subsequently. ILM is initialized to all zeroes by a reset. An instruction may transfer an eight-bit immediate value to the ILM register, but only the low-order three bits of that data are used.

Figure 2.7-9 Interrupt Level Register (ILM)

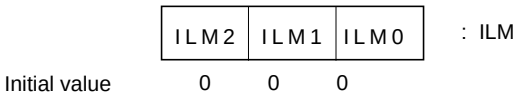


Table 2.7-1 Levels Indicated by the Interrupt Level Mask (ILM) Register

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

Table 2.7-1 Levels Indicated by the Interrupt Level Mask (ILM) Register (Continued)

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

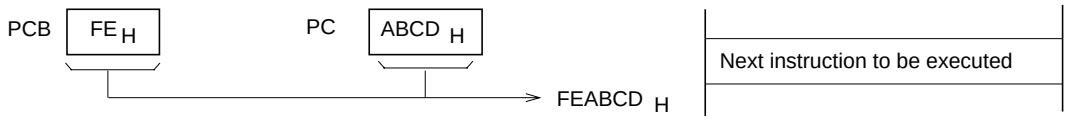
2.7.4 Program Counter (PC)

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

■ Program Counter (PC)

Figure 2.7-10 shows the program counter.

Figure 2.7-10 Program Counter



2.8 Register Bank

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

■ Register Bank

Table 2.8-1 lists the functions of the registers. Table 2.8-2 indicates the relationship between the registers.

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

Table 2.8-1 Register Functions

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

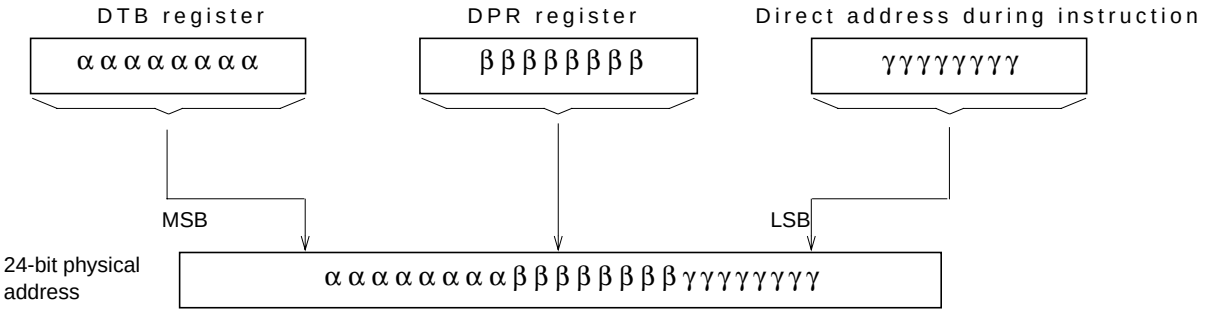
Table 2.8-2 Relationship between Registers

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

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

DPR specifies addr8 to addr15 of the instruction operands in direct addressing mode as shown in Figure 2.8-1. DPR is eight bits long, and is initialized to 01H by a reset. DPR can be read or written to by an instruction.

Figure 2.8-1 Generating a Physical address in Direct Addressing Mode



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

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

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

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

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

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

PCB is updated when the JMPP, CALLP, RETP, RETIQ, or RETF instruction branching to the entire 16-Mbyte space is executed or when an interrupt occurs. For operation of each register, see Section 2.2 "Memory space."

2.9 Prefix Codes

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

■ Bank Select Prefix

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

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

Table 2.9-1 lists the bank select prefixes and the corresponding memory spaces.

Table 2.9-1 Bank Select Prefix

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

Use the following instructions with care:

❍ **String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)**

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

❍ **Stack manipulation instructions (PUSHW, POPW)**

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

❍ **I/O access instructions**

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

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

❍ **Flag change instructions (AND CCR,#imm8, OR CCR,#imm8)**

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

❍ **POPW PS**

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

❍ **MOV ILM,#imm8**

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

❍ **RETI**

SSB is used regardless of the prefix.

■ Common Register Bank Prefix (CMR)

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

❍ **String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)**

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

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

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

❍ **MOV ILM,#imm8**

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

■ Flag Change Disable Prefix (NCC)

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

❍ **String instructions (MOVS, MOVSW, SCEQ, SCWEQ, FILS, FILSW)**

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

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

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

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

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

❍ **JCTX @A**

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

❍ **MOV ILM,#imm8**

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

2.10 Interrupt Disable Instructions

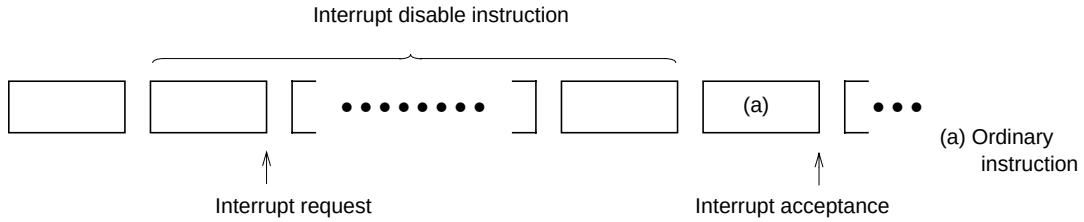
Interrupt requests are not sampled for the following ten instructions:

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

■ Interrupt Disable Instructions

If a valid interrupt request occurs during execution of any of the above instructions, the interrupt can be processed only when an instruction other than the above is executed. For details, see Figure 2.10-1.

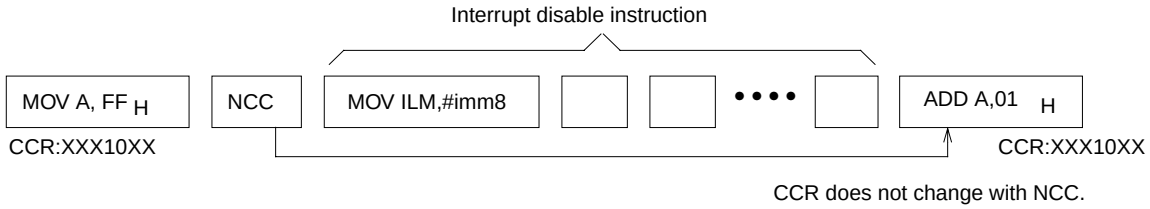
Figure 2.10-1 Interrupt Disable Instruction



■ Restrictions on Interrupt Disable Instructions and Prefix Instructions

When a prefix code is placed before an interrupt disable instruction, the prefix code affects the first instruction after the code other than the interrupt disable instruction. For details, see Figure 2.10-2.

Figure 2.10-2 Interrupt Disable Instructions and Prefix Codes

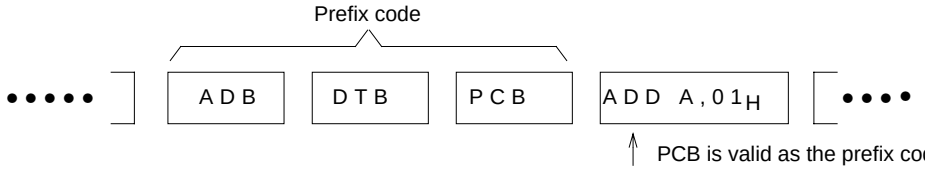


■ Consecutive Prefix Codes

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

In the figure below, competitive prefix codes are PCB, ADB, DTB, and SPB. For details, see Figure 2.10-3.

Figure 2.10-3 Consecutive Prefix Codes



CHAPTER 3 INTERRUPTS

This chapter explains the interrupt functions and operations.

3.1 Outline of Interrupts

3.2 Interrupt Vector

3.3 Interrupt Control Registers (ICR)

3.4 Interrupt Flow

3.5 Hardware Interrupts

3.6 Software Interrupts

3.7 Extended Intelligent I/O Service (EI^2OS)

3.8 Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI^2OS)

3.9 Exceptions

3.1 Outline of Interrupts

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

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

■ Hardware Interrupts

A hardware interrupt is activated by an interrupt request from an internal resource. A hardware interrupt request occurs when both the interrupt request flag and the interrupt enable flag in an internal resource are set. Therefore, an internal resource must have an interrupt request flag and interrupt enable flag to issue a hardware interrupt request.

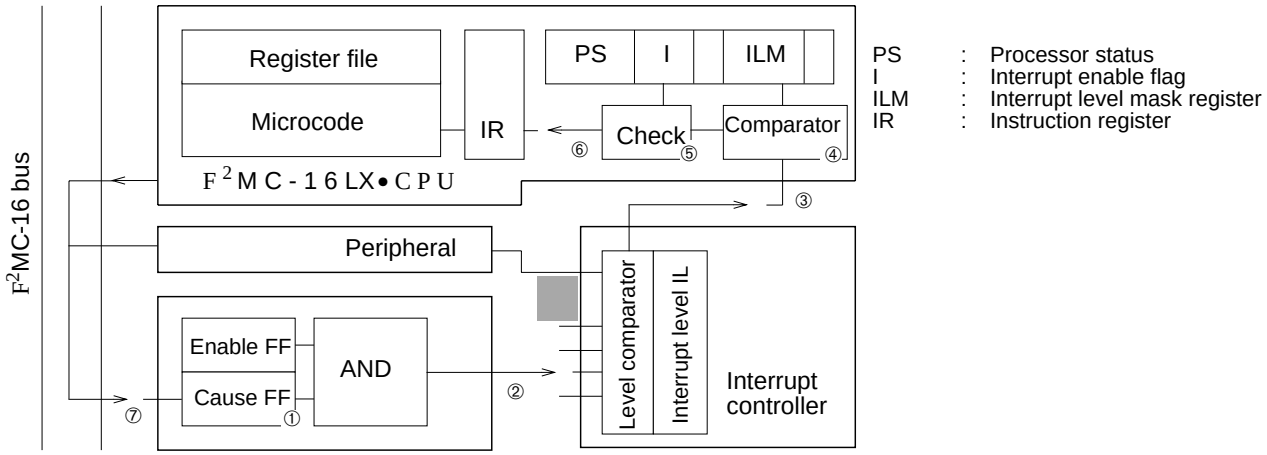
❍ Specifying an interrupt level

An interrupt level can be specified for the hardware interrupt. To specify an interrupt level, use the level setting bits (IL0, IL1, and IL2) of the interrupt controller.

❍ Masking a hardware interrupt request

A hardware interrupt request can be masked by using the I flag of the processor status register (PS) in the CPU and the ILM bits (IL0, IL1, and IL2). When an unmasked interrupt request occurs, the CPU saves 12 bytes of data that consists of registers PS, PC, PCB, DTB, ADB, DPR, and A in the memory area indicated by the SSB and SSP registers.

Figure 3.1-1 Overview of Hardware Interrupts



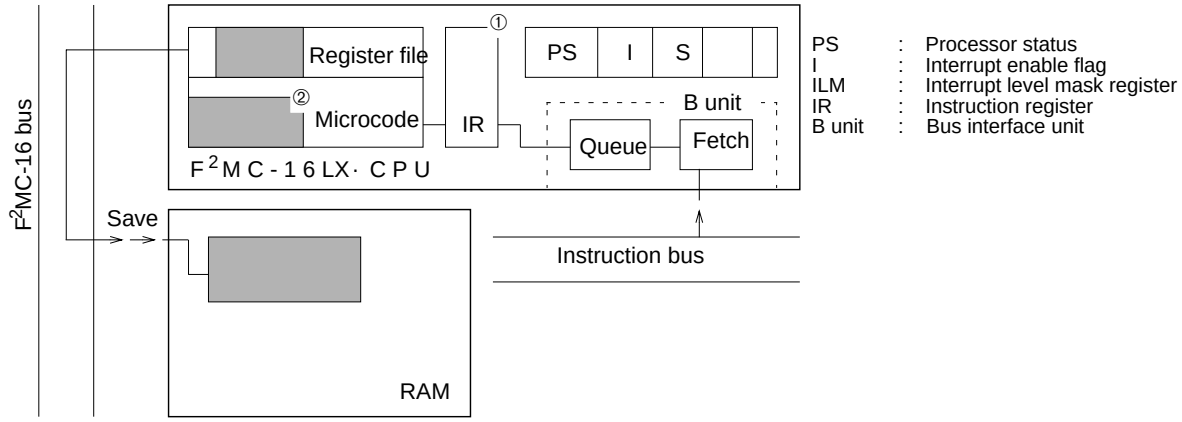
■ Software Interrupts

Interrupts requested by executing the INT instruction are software interrupts. An interrupt

request by the INT instruction does not have an interrupt request or enable flag. An interrupt request is issued always by executing the INT instruction.

No interrupt level is assigned to the INT instruction. Therefore, ILM is not updated when the INT instruction is used. Instead, the I flag is cleared and the continuing interrupt requests are suspended.

Figure 3.1-2 Overview of Software Interrupts



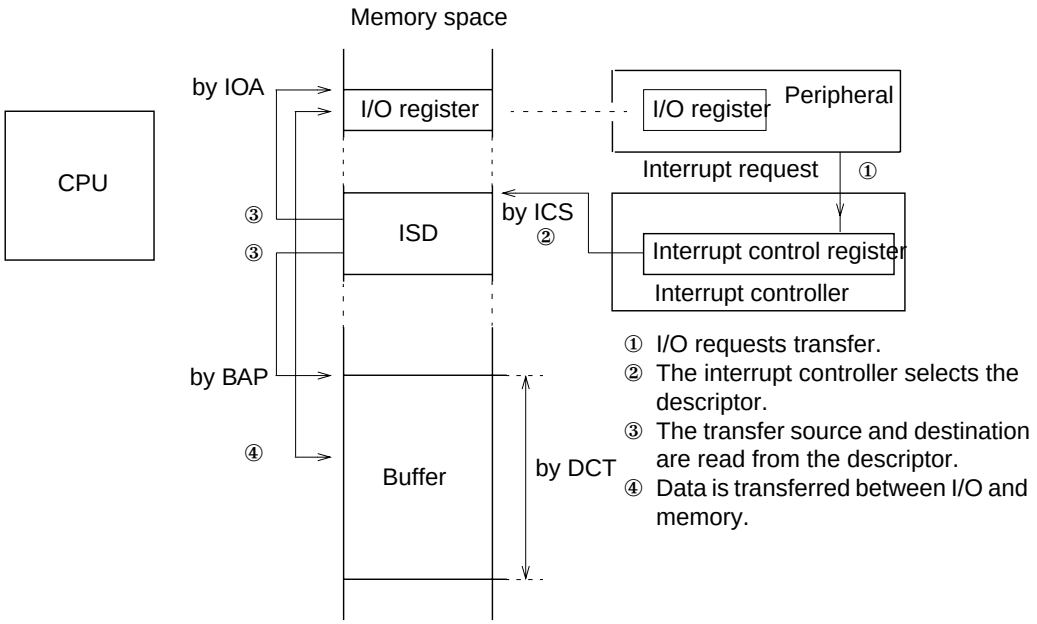
■ Extended Intelligent I/O Service (EI^{2}OS)

The extended intelligent I/O service automatically transfers data between an internal resource and memory. This processing is traditionally performed by an interrupt processing program, but the EI^{2}OS enables data to be transferred in a manner similar to a DMA (direct memory access) operation.

To activate the extended intelligent I/O service function from an internal resource, the interrupt control register (ICR) of the interrupt controller must have an extended intelligent I/O service enable flag (ISE).

The extended intelligent I/O service is started when an interrupt request occurs with 1 specified in the ISE flag. To generate a normal interrupt using a hardware interrupt request, set the ISE flag to 0.

Figure 3.1-3 Overview of the Extended Intelligent I/O Service (EI^{2}OS)



■ Exceptions

Exception processing is basically the same as interrupt processing. When an exception is detected between instructions, exception processing is performed. In general, exception processing occurs as a result of an unexpected operation. Therefore, use exception processing only for debugging programs or for activating recovery software in an emergency.

3.2 Interrupt Vector

An interrupt vector uses the same area for both hardware and software interrupts. For example, interrupt request number INT42 is used for a delayed hardware interrupt and for software interrupt INT #42. Therefore, the delayed interrupt and INT #42 call the same interrupt processing routine. Interrupt vectors are allocated between addresses FFFC00H and FFFFFFH as shown in Table 3.2-1.

■ Interrupt Vector

Table 3.2-1 Interrupt Vectors

Interrupt request	Vector address L	Vector address H	Vector address bank	Mode register
INT 0*1	$FFFFFC_H$	$FFFFFD_H$	$FFFFFE_H$	Unused
INT 1*1	$FFFFF8_H$	$FFFFF9_H$	$FFFFFA_H$	Unused
. . .	. . .	. . .	. . .	. . .
INT 7*1	$FFFFE0_H$	$FFFFE1_H$	$FFFFE2_H$	Unused
INT 8*2	$FFFFDC_H$	$FFFFDD_H$	$FFFFDE_H$	$FFFFDF_H$
INT 9	$FFFFD8_H$	$FFFFD9_H$	$FFFFDA_H$	Unused
INT 10*3	$FFFFD4_H$	$FFFFD5_H$	$FFFFD6_H$	Unused
INT 11	$FFFFD0_H$	$FFFFD1_H$	$FFFFD2_H$	Unused
. . .	. . .	. . .	. . .	. . .
INT 254	$FFFC04_H$	$FFFC05_H$	$FFFC06_H$	Unused
INT 255	$FFFC00_H$	$FFFC01_H$	$FFFC02_H$	Unused

***1: When PCB is FFH, the vector area for the CALLV instruction is the same as that for INT #vct8(#0 to #7). Care must be taken when using the vector for the CALLV instruction.**

***2: The vector is a reset vector.**

***3: The vector is an exception processing vector.**

■ Listing of Interrupt Vectors

See Table C-1 in Appendix C for a list of the MB90590 interrupt vectors.

3.3 Interrupt Control Registers (ICR)

The interrupt control registers are in the interrupt controller. Each interrupt control register has a corresponding I/O that has an interrupt function. The interrupt control registers have the following three functions:

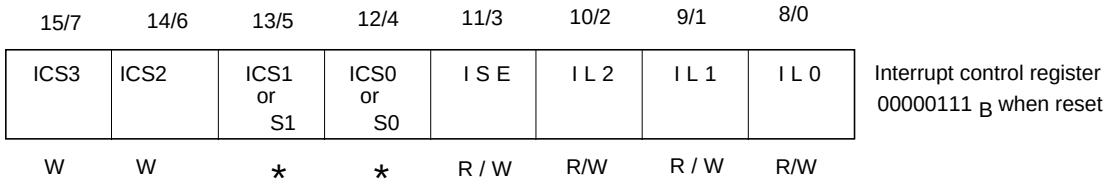
- **Setting an interrupt level for corresponding peripherals**
- **Selecting whether to use an ordinary interrupt or extended intelligent I/O service for the corresponding peripherals**
- **Selecting the extended intelligent I/O service channel**

Do not access an interrupt control register by using a read-modify-write instruction, as doing so causes a misoperation.

■ Interrupt Control Register (ICR)

Figure 3.3-1 is a diagram of the bit configuration of an interrupt control register.

Figure 3.3-1 Interrupt Control Register (ICR)



Note:

ICS3 to ICS0 are valid only when EI^{2}OS is activated. Set '1' in ISE to activate EI^{2}OS, and set '0' in ISE not to activate it. When EI^{2}OS is not to be activated, any value can be set in ICS3 to ICS0. * '1' is read always.

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

[bits 10 to 8] [bits 2 to 0]: IL0, IL1, and IL2 (interrupt level setting bits)

These bits are readable and writable, and specify the interrupt level of the corresponding internal resources. Upon a reset, these bits are initialized to level 7 (no interrupt). Table 3.3-1 describes the relationship between the interrupt level setting bits and interrupt levels.

Table 3.3-1 Interrupt Level Setting Bits and Interrupt Levels

ILM2	ILM1	ILM0	Level
0	0	0	0 (Strongest)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5

Table 3.3-1 Interrupt Level Setting Bits and Interrupt Levels (Continued)

ILM2	ILM1	ILM0	Level
1	1	0	6 (Weakest)
1	1	1	7 (No interrupt)

[bit 11] [bit 3]: ISE (extended intelligent I/O service enable bits)

The ISE bit is readable and writable. In response to an interrupt request, EI^{2}OS is activated when '1' is set in the ISE bit and an interrupt sequence is activated when '0' is set in the ISE bit. Upon completion of EI^{2}OS, the ISE bit is cleared to a zero. If the corresponding peripheral does not have the EI^{2}OS function, the ISE bit must be set to '0' on the software side.

Upon a reset, the ISE bit is initialized to '0.'

[bits 15 to 12] [bits 7 to 4]: ICS 3 to 0 (extended intelligent I/O service channel select bits)

ICS3 to ICS0 are write-only bits. These bits specify the EI^{2}OS channel. The values set in these bits determined the intelligent I/O service descriptor addresses in memory, which is explained later. The ICS bits are initialized by a reset.

Table 3.3-2 describes the correspondence between the ICS bits, channel numbers, and descriptor addresses.

Table 3.3-2 ICS bits, Channel Numbers, and Descriptor Addresses

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

[bits 13 and 12] [bits 5 and 4]: S0 and S1 (extended intelligent I/O service status)

S0 and S1 are read-only bits. The values set in these bits indicate the end condition of EI^{2}OS. These bits are initialized to '00' upon a reset.

Table 3.3-3 shows the relationship between the S bits and the end conditions.

Table 3.3-3 S Bits and End Conditions

S1	S0	End condition
0	0	EI^{2}OS running or not activated
0	1	Termination by count
1	0	Reserved
1	1	Termination by request from resource

3.4 Interrupt Flow

Figure 3.4-1 shows the interrupt flow.

■ **Interrupt Flow**

Figure 3.4-1 Interrupt Flow

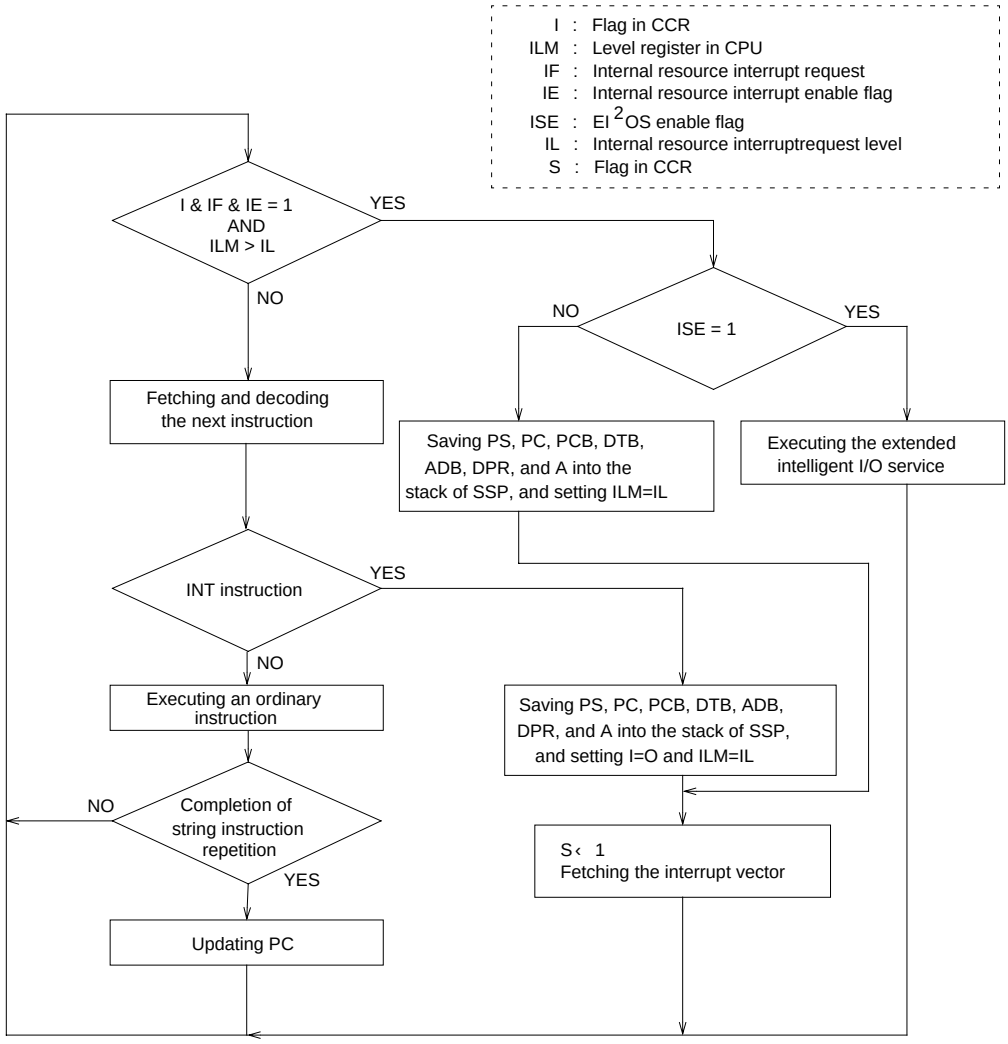
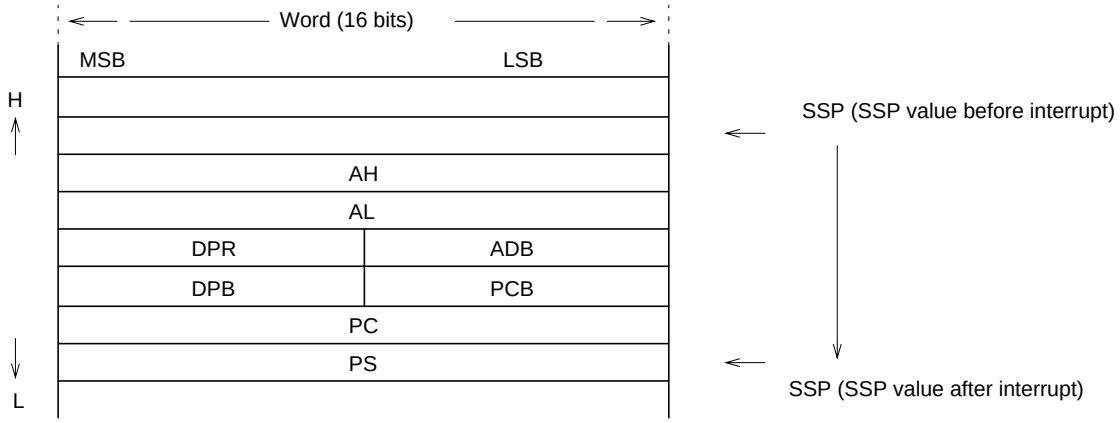


Figure 3.4-2 Register Saving during Interrupt Processing



3.5 Hardware Interrupts

In response to an interrupt request signal from an internal resource, the CPU pauses current program execution and transfers control to the interrupt processing program defined by the user. This function is called the hardware interrupt function.

■ Hardware Interrupts

A hardware interrupt occurs when the relevant conditions are satisfied as a result of two operations: comparison between the interrupt request level and the value in the interrupt level mask register (ILM) of PS in the CPU, and hardware reference to the I flag value of PS.

The CPU performs the following processing when a hardware interrupt occurs:

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

■ Structure of Hardware Interrupt

Hardware interrupts are handled by the following three sections:

❍ **Internal resources**

Interrupt enable and request bits: Used to control interrupt requests from resources.

❍ **Interrupt controller**

ICR: Assigns interrupt levels and determines the priority levels of simultaneously requested interrupts.

❍ **CPU**

I and ILM: Used to compare the requested and current interrupt levelsand to identify the interrupt enable status.

Microcode: Interrupt processing step

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

The interrupt vector table referenced during interrupt processing is assigned to addresses $FFFC00_H$ to $FFFFFF_H$ in memory. These addresses are shared with software interrupts.

3.5.1 Hardware Interrupt Operation

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

■ Hardware Interrupt Operation

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

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

The interrupt processing body saves 12 bytes (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP, fetches three bytes of interrupt vector and loads them onto PC and PCB, updates the ILM of PS to a level value of the received interrupt, sets the S flag, then performs branch processing. As a result, the interrupt processing program defined by the user is executed next.

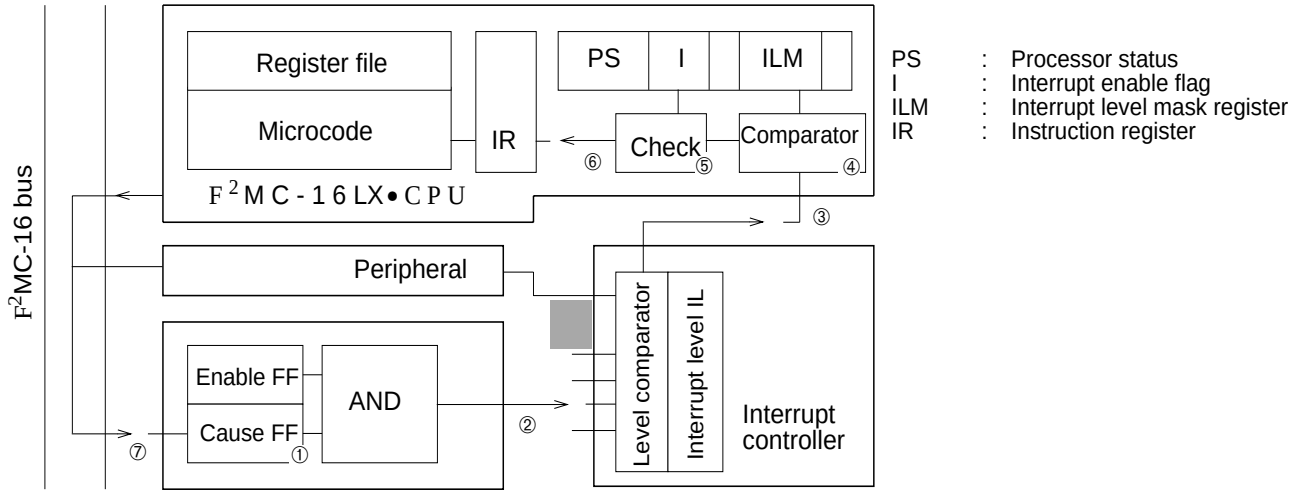
Figure 3.5-1 illustrates the flow from the occurrence of a hardware interrupt until there is no interrupt request in the interrupt processing program.

3.5.2 Occurrence and Release of Hardware Interrupt

Figure 3.5-1 shows the processing flow from occurrence of a hardware interrupt to release of the interrupt request in an interrupt processing program.

■ Occurrence and Release of Hardware Interrupt

Figure 3.5-1 Occurrence and Release of Hardware Interrup



1. An interrupt cause occurs in a peripheral.
2. The interrupt enable bit in the peripheral is referenced. If interrupts are enabled, the peripheral issues an interrupt request to the interrupt controller.
3. Upon reception of the interrupt request, the interrupt controller determines the priority levels of simultaneously requested interrupts. Then, the interrupt controller transfers the interrupt level of the corresponding interrupt to the CPU.
4. The CPU compares the interrupt level requested by the interrupt controller with the ILM bit of the processor status register.
5. If the comparison shows that the requested level is higher than the current interrupt processing level, the I flag value of the same processor status register is checked.
6. If the check in step 5. shows that the I flag indicates interrupt enable status, the requested level is written to the ILM bit. Interrupt processing is performed as soon as the currently executing instruction is completed, then control is transferred to the interrupt processing routine.
7. When the interrupt cause of step 1. is cleared by software in the user interrupt processing routine, the interrupt request is completed.

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

Interrupt start: 24 + 6 × (Table 3.3.2 machine cycles)

Interrupt return: 15 + 6 × (Table 3.3.2 machine cycles) RETI instruction

Table 3.5-1 Compensation Values for Interrupt Processing Cycle Count

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

3.5.3 Multiple interrupts

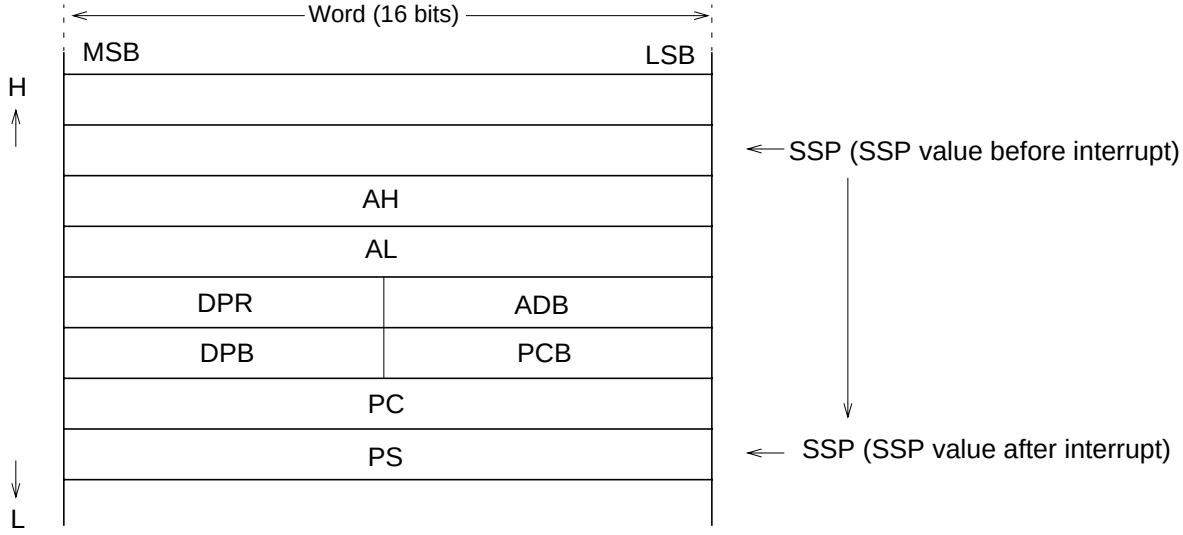
As a special case, no hardware interrupt request can be accepted while data is being written to the I/O area. This is intended to prevent the CPU from operating falsely because of an interrupt request issued while an interrupt control register for a resource is being updated.
If an interrupt occurs during interrupt processing, a higher-level interrupt is processed first.

■ Multiple Interrupts

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

Figure 3.5-2 shows the order of the registers saved in the stack.

Figure 3.5-2 Registers Saved in Stack



3.6 Software Interrupts

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

■ Software Interrupts

The CPU performs the following processing when a software interrupt occurs:

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

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

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

■ Structure of Software Interrupts

Software interrupts are handled within the CPU:

CPU.....Microcode: Interrupt processing step

■ List of MB90590 Interrupt Vectors

Table D-1 lists the interrupt vectors of the MB90590 series.

As shown in Table D-1, software interrupts share the same interrupt vector area with hardware interrupts.

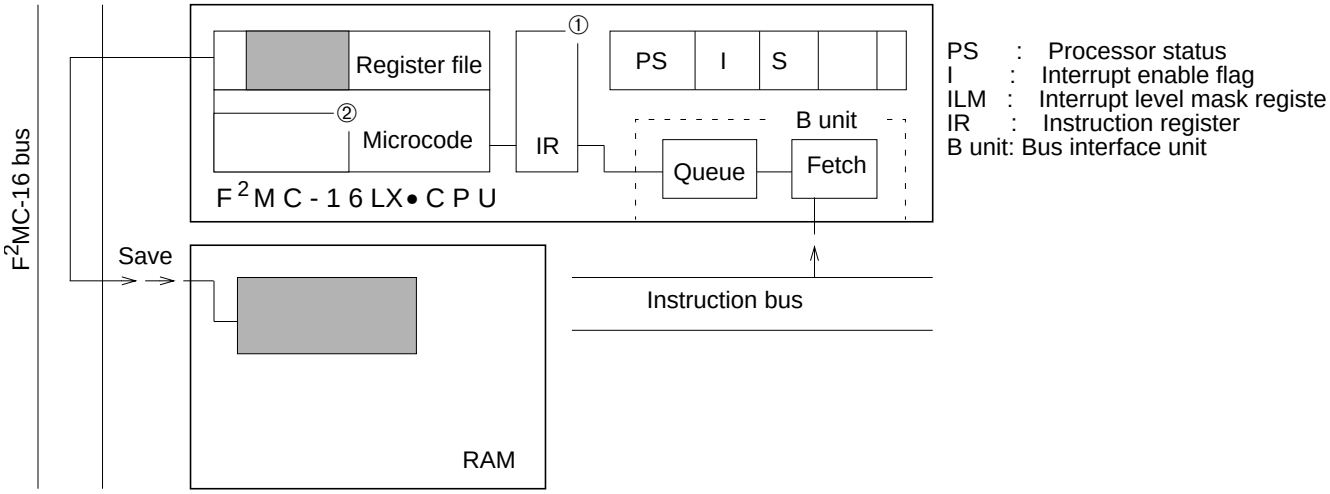
For example, interrupt request number INT 13 is used for external interrupt #0 of a hardware interrupt as well as for INT #13 of a software interrupt. Therefore, external interrupt #0 and INT #13 call the same interrupt processing routine.

■ Software Interrupt Operation

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

Figure 3.6-1 illustrates the flow from the occurrence of a software interrupt until there is no interrupt request in the interrupt processing program.

Figure 3.6-1 Occurrence and Release of Software Interrupt



1. The software interrupt instruction is executed.
2. Special CPU registers in the register file are saved according to the microcode corresponding to the software interrupt instruction.
3. The interrupt processing is completed with the RETI instruction in the user interrupt processing routine.

■ Others

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

Table D-2 shows the relationship of interrupt cause, interrupt vector, and interrupt control register in the MB90590 series.

3.7 Extended Intelligent I/O Service (EI^{2}OS)

The EI^{2}OS function automatically transfers data between input and output and memory. An interrupt processing program was conventionally used for such processing, but EI^{2}OS enables data transfer to be performed like DMA (direct memory access).

■ Extended Intelligent I/O Service (EI^{2}OS)

EI^{2}OS has the following advantages over the conventional method:

- The program size can be small because it is not necessary to write a transfer program.
- No internal register is used for transfer, eliminating the need for register saving and increasing the transfer speed.
- Transfer can be terminated from I/O, preventing unnecessary data from being transferred.
- The buffer address may either be incremented or left unupdated.
- The I/O register address may either be incremented or left unupdated.

At the end of EI^{2}OS, processing automatically branches to an interrupt processing routine after the end condition is set. Thus, the user can identify the end condition.

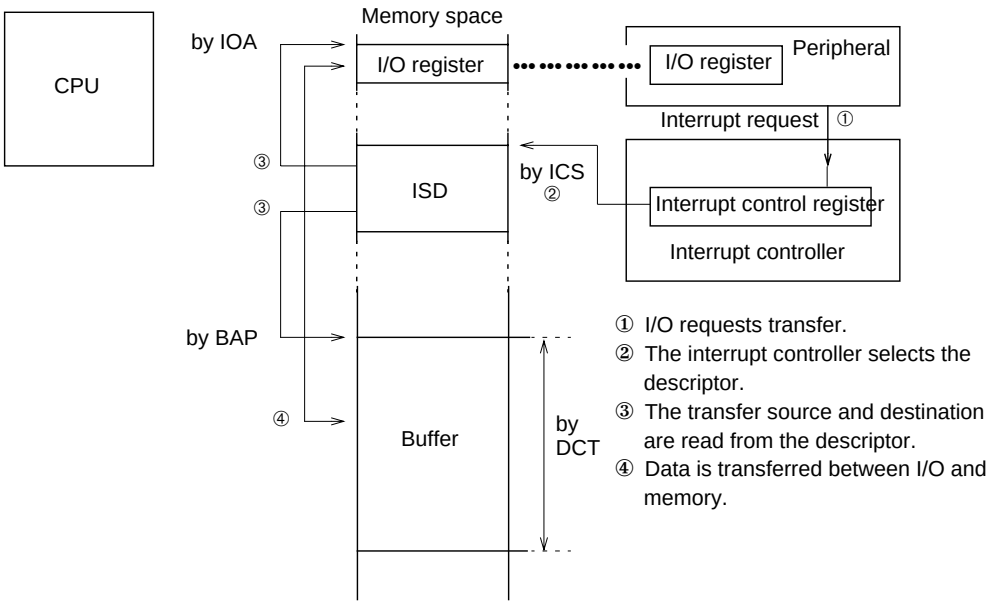
To implement EI^{2}OS, the hardware is distributed in two blocks. Each block has the following registers and descriptors.

❍ **Interrupt control register: Exists in the interrupt controller and indicates the ISD address.**

❍ **Extended intelligent I/O service descriptor (ISD): Exists in RAM and holds the transfer mode, I/O address, number of transfers, and buffer address.**

Figure 3.7-1 outlines the extended intelligent I/O service.

Figure 3.7-1 Outline of Extended Intelligent I/O Service



Note:

The area that can be specified by IOA is between 000000_H and $00FFFF_H$.

The area that can be specified by BAP is between 000000_H and $FFFFFF_H$.

The maximum transfer count that can be specified by DCT is 65,536.

■ Structure

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

Internal resources

Interrupt enable and request bits: Used to control interrupt requests from resources.

Interrupt controller

ICR: Assigns interrupt levels, determines the priority levels of simultaneously requested interrupts, and selects the EI^{2}OS operation.

CPU

I and ILM: Used to compare the requested and current interrupt levels and to identify the interrupt enable status

Microcode: EI^{2}OS processing step

RAM

Descriptor: Describes the EI^{2}OS transfer information.

.

3.7.1 Extended Intelligent I/O Service Descriptor (ISD)

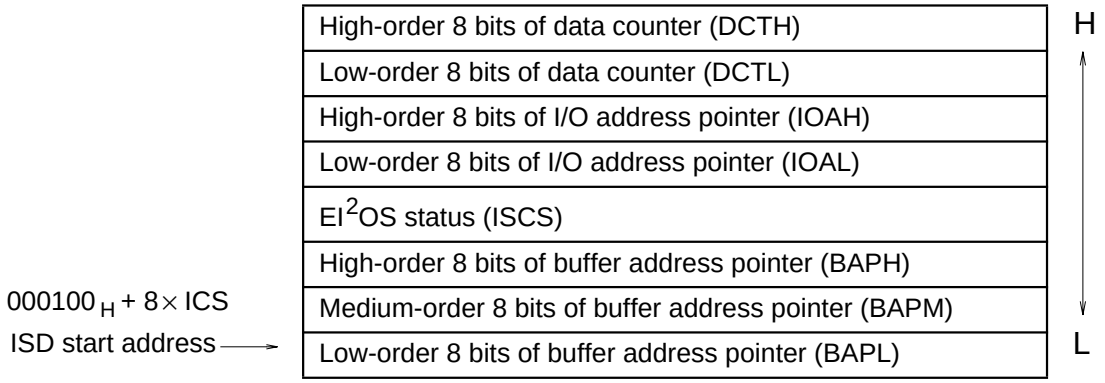
The extended intelligent I/O service descriptor exists between 000100_H and $00017F_H$ in internal RAM, and consists of the following items:

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

■ Extended Intelligent I/O Service Descriptor (ISD)

Figure 3.7-2 shows the configuration of the extended intelligent I/O service descriptor.

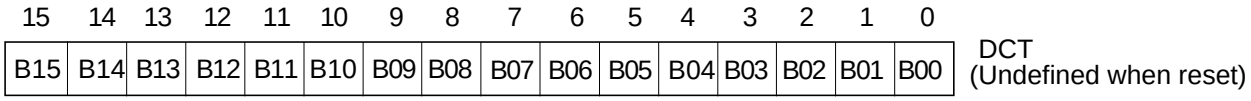
Figure 3.7-2 Extended Intelligent I/O Service Descriptor Configuration



■ Data Counter (DCT)

This is a 16-bit register that works as a counter corresponding to the number of data items transferred. This counter is decremented by one before data transfer. EI^{2}OS is terminated when this counter reaches 0. Figure 3.7-3 is a diagram of the data counter configuration.

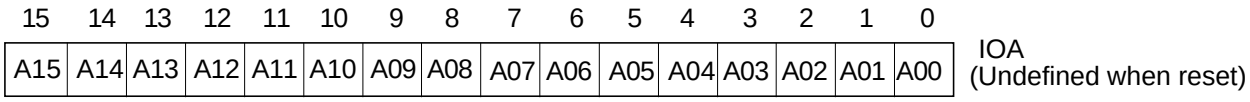
Figure 3.7-3 Data Counter Configuration



■ I/O Register Address Pointer (IOA)

This is a 16-bit register that indicates the low-order address (A15 to A0) of the buffer and I/O register used for data transfer. The high-order address (A23 to A16) are all zeroes, and any I/O between addresses 000000_H and $00FFFF_H$ can be specified. Figure 3.7-4 is a diagram of the IOA configuration.

Figure 3.7-4 I/O Register Address Pointer Configuration



■ **Buffer Address Pointer (BAP)**

This 24-bit register holds the address used for the next EI^{2}OS transfer. BAP exists for each EI^{2}OS channel. Therefore, each EI^{2}OS channel can be used for transfer with anywhere in the 16-Mbyte space. If the BF bit of ISCS is set to '0' (update enabled), only the low-order 16 bits of BAP changes and BAPH does not change.

3.7.2 EI^{2}OS Status Register (ISCS)

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

■ EI^{2}OS Status Register (ISCS)

Figure 3.7-5 is a diagram of the ISCS configuration.

Figure 3.7-5 ISCS Configuration

7	6	5	4	3	2	1	0	
Reserved	Reserved	Reserved	IF	BW	BF	DIR	SE	ISCS (Undefined when reset)

* Always write 0 to bits 7 to 5 of ISCS.

Each bit is described below.

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

0 : The I/O register address pointer is updated after data transfer.

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

<Note> Only increment is allowed.

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

0 : Byte

1 : Word

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

0 : The buffer address pointer is updated after data transfer.

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

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

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

0 : I/O --> Buffer

1 : Buffer --> I/O

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

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

1 : The extended intelligent I/O service is terminated by a resource request.

3.8 Operation Flow of and Procedure for Using the Extended Intelligent I/O Service (EI^{2}OS)

Figure 3.8-1 is a diagram of the EI^{2}OS operation flow. Figure 3.8-2 is a diagram of the EI^{2}OS use procedure.

■ EI^{2}OS Operation Flow

Figure 3.8-1 EI^{2}OS Operation Flow

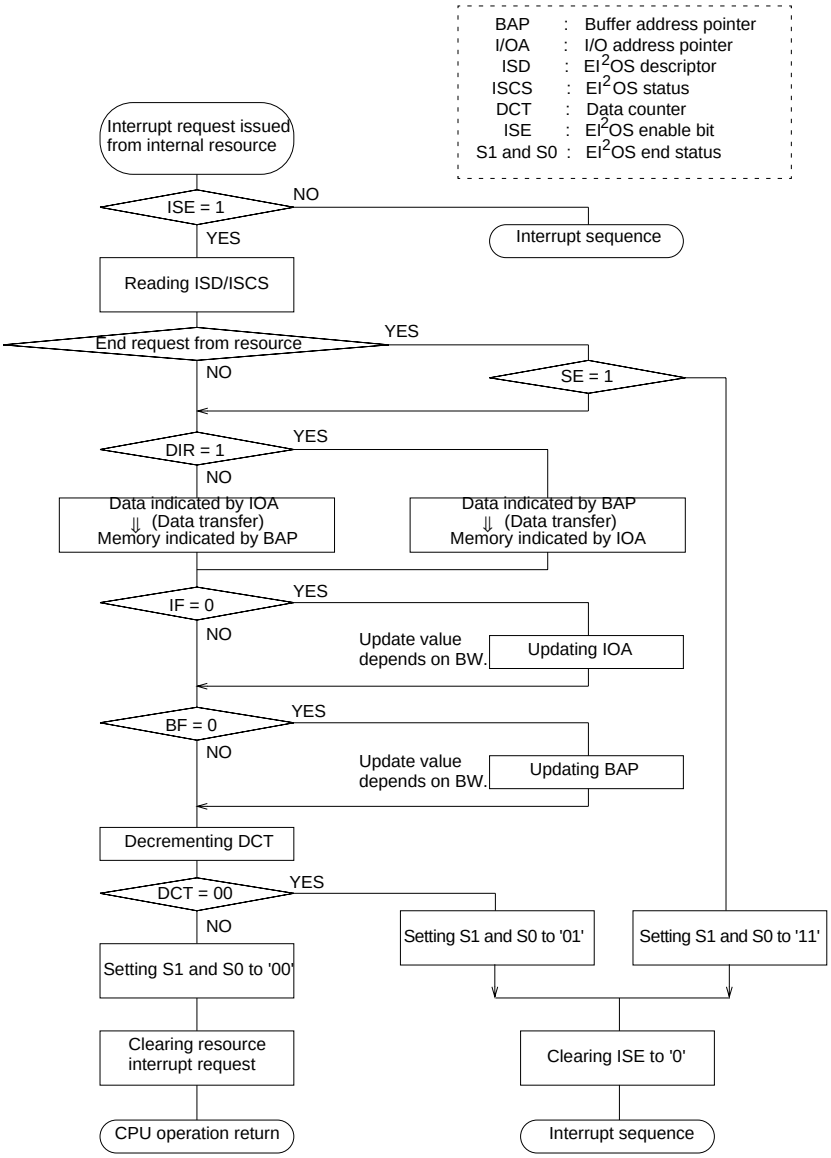
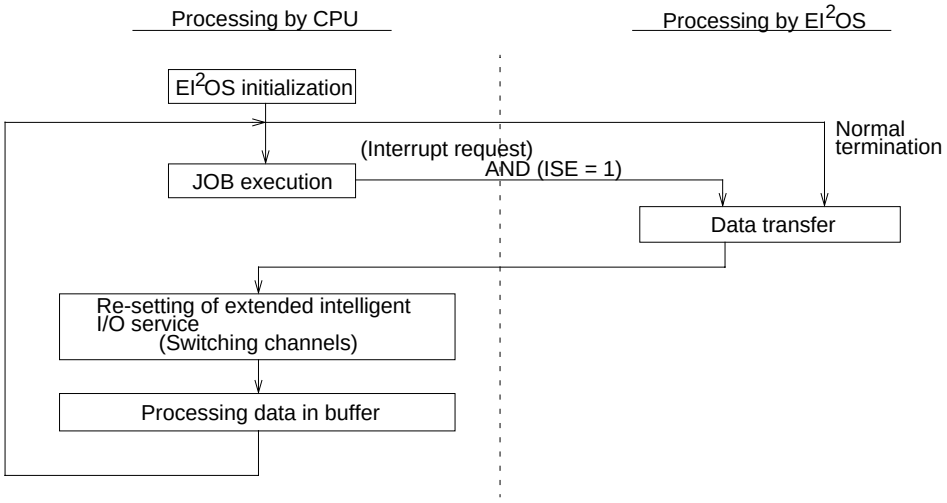


Figure 3.8-2 EI^{2}OS Use Flow



The extended EI^{2}OS execution time for each flow is described below.

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

(Table 3.8-1 + Table 3.8-2) machine cycles

❍ **When a stop request is issued from a resource**

(36 + 6 × Table 3.D-2) machine cycles

❍ **When the counting is completed**

(Table 3.8-1 + Table 3.8-2 + (21 + 6 × Table 3.D-2)) machine cycles

Table 3.8-1 Execution Time when the Extended EI^{2}OS Continues

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

Table 3.8-2 Data Transfer Compensation Values for Extended EI^{2}OS Execution Time

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

B: Byte data transfer

E: Even address word transfer

8: 8-bit external bus word transfer

O: Odd address word transfer

3.9 Exceptions

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

■ Execution of an Undefined Instruction

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

■ Exception due to Execution of an Undefined Instruction

The F^{2}MC-16LX handles all codes that are not defined in the instruction map as undefined instructions. When an undefined instruction is executed, processing equivalent to the INT 10 software interrupt instruction is performed. Specifically, the AL, AH, DPR, DTB, ADB, PCB, PC, and PS values are saved into the system stack, and processing branches to the routine indicated by the interrupt number 10 vector. In addition, the I flag is cleared and the S flag is set. The PC value saved in the stack is the address at which the undefined instruction is stored. Processing can be restored by the RETI instruction, but is of no use, however, because the same exception occurs again.

CHAPTER 4 DELAYD INTERRUPT

This chapter explains the functions and operations of the delayed interrupt.

4.1 Outline of Delayed Interrupt Module

4.2 Delayed Interrupt Register

4.3 Delayed Interrupt Operation

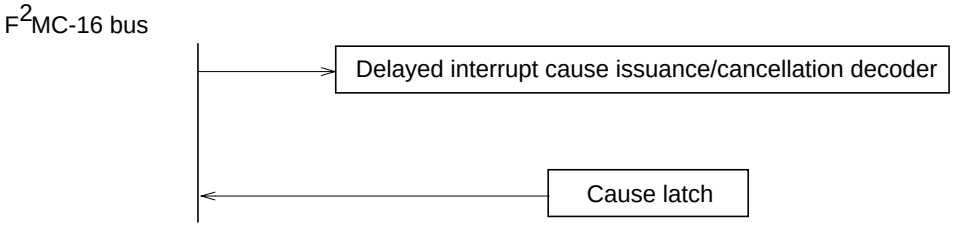
4.1 Outline of Delayed Interrupt Module

The delayed interrupt source module is used to generate interrupts for switching tasks. Using this module, interrupt requests to the F^{2}MC-16LX CPU can be issued and canceled by software.

■ Block Diagram of Delayed Interrupt

Figure 4.1-1 is a block diagram of the delayed interrupt source module.

Figure 4.1-1 Block Diagram



■ Notes on Operation

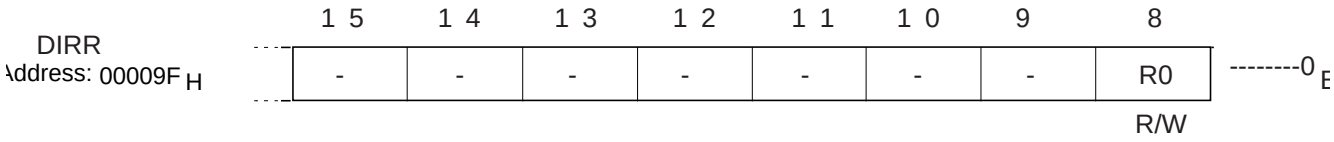
This lock is set by writing '1' to the corresponding bit of DIRR, and is cleared by writing '0' to the same bit. Therefore, interrupt processing is reactivated immediately after control returns from interrupt processing, unless the software is designed so that the cause of the interrupt is cleared within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

4.2 Delayed Interrupt Register

DIRR controls issuance and cancellation of delayed interrupt requests. Writing "1" to this register issues a delayed interrupt request, and writing "0" cancels the delayed interrupt request. Upon a reset, the request is canceled.

■ **Delayed Interrupt Cause Issuance/Cancellation Register (DIRR: Delayed Interrupt Request Register)**

In DIRR, either "0" or "1" can be written to the reserved bit area. However, it is recommended that a set bit or clear bit instruction be used to access this register for future expansions.



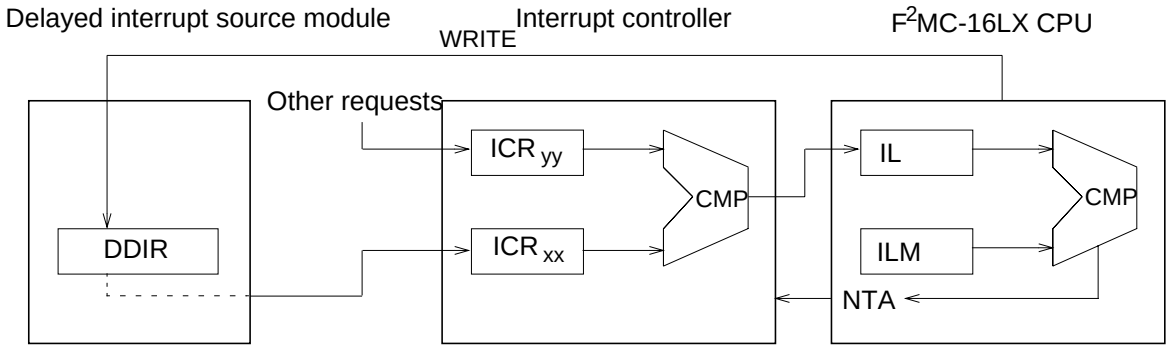
4.3 Delayed Interrupt Operation

When the CPU writes "1" to the relevant bit of DIRR by software, the request latch in the delayed interrupt source module is set and an interrupt request is issued to the interrupt controller.

■ Delayed Interrupt Occurrence

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

Figure 4.3-1 Delayed Interrupt Issuance



CHAPTER 5 CLOCK AND RESET

This chapter explains the functions and operations of clocks and resets.

5.1 Clock Generator

5.2 Reset Cause Occurrence

5.3 Reset Causes

5.1 Clock Generator

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

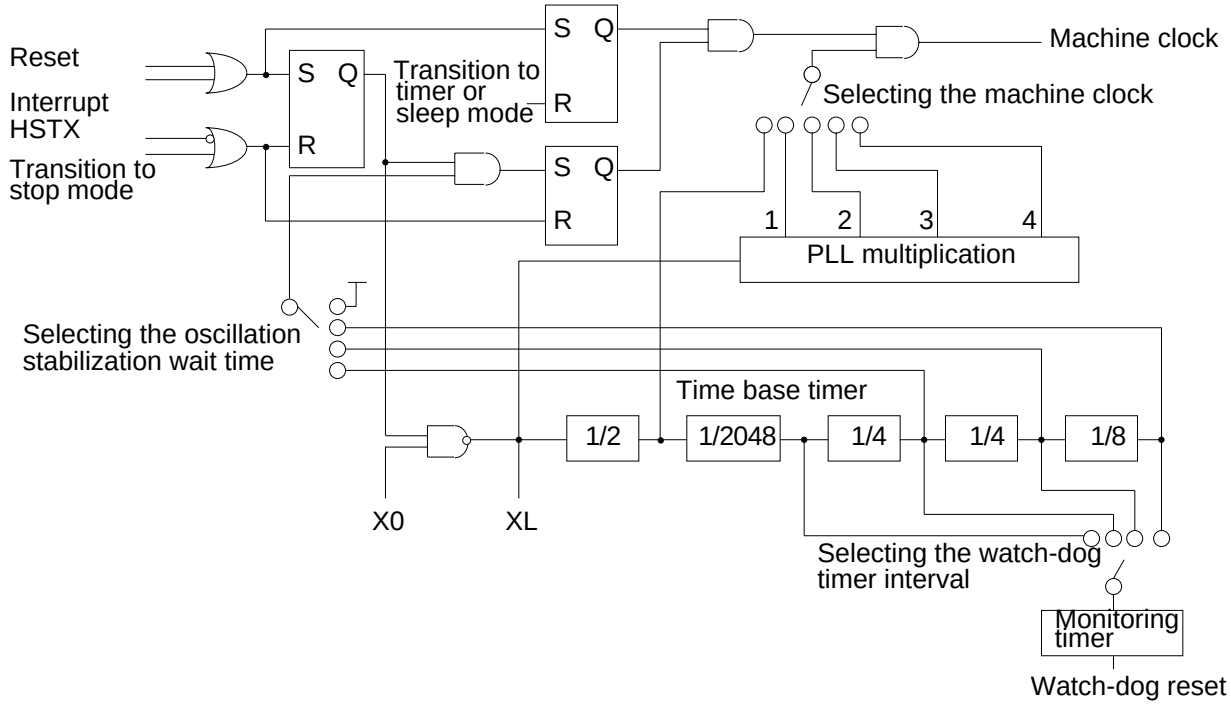
■ Notes on Clock Generator

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

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

Figure 5.1-1 is a block diagram of the clock generator circuit.

Figure 5.1-1 Clock Generator Circuit Block Diagram



5.2 Reset Cause Occurrence

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

■ Reset Cause Occurrence

A reset is caused by the following factors:

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

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

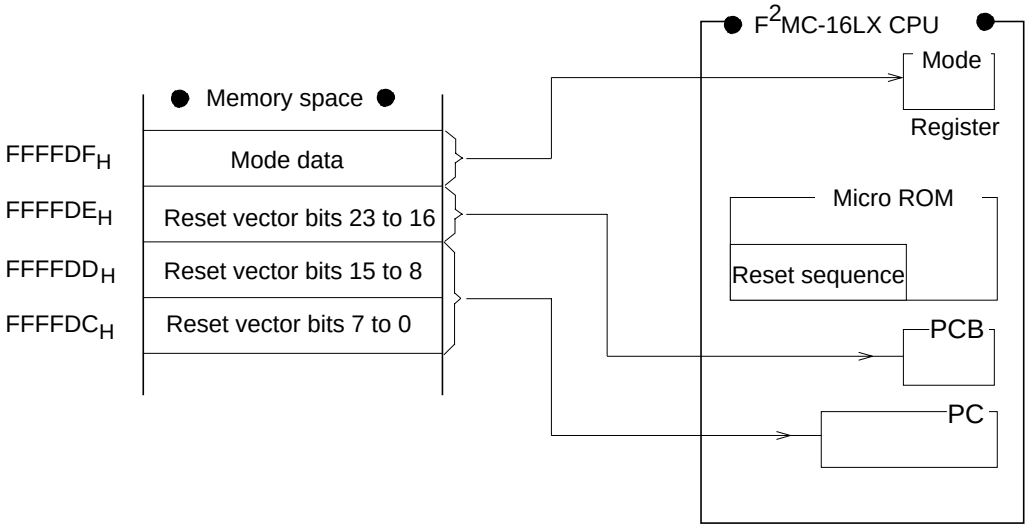
■ Operation after Reset Release

When a reset cause is removed, the F^{2}MC-16LX immediately outputs the address in which the reset vector is stored, then fetches the reset vector and mode data. The reset vector and mode data are assigned to the four bytes between FFFFDC$_H$ and FFFFDF$_H$. After reset is released, the reset vector and mode data are transferred to the registers by the hardware as described in Figure 5.2-1.

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

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

Figure 5.2-1 Source and Destination of Reset Vector and Mode Data



Note:

For MB90F594, the reset vector and mode data have predetermined values by the hard-wired logic.

For more information, refer to "Reset Vector Addesses in Flash Memory".

5.3 Reset Causes

Table 5.3-1 lists the five reset causes. The machine clock and watch-dog function are initialized differently for each reset cause.
The reset cause register indicates the reset cause.

■ Reset Causes

Table 5.3-1 Reset Causes

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

* In stop or hardware standby mode, a reset input allows for oscillation stabilization time regardless of the reset cause.

* The oscillation stabilization time for a power-on reset is fixed to 2^{18} cycles of source oscillation. For other types of reset, the oscillation stabilization wait time is determined by CS1 and CS0 of the clock selection register.

As shown in Figure 5.3-1, each reset cause has a corresponding flip-flop. The contents of the flip-flop can be obtained by reading the watch-dog timer control register. If identifying the reset cause is requiredafter the reset is released, ensure that the value read from the watch-dog timer control register is processed by software and processing branches to an appropriate program. Figure 5.3-2 is a diagram of the watch-dog timer control register.

Figure 5.3-1 Reset Cause bit Block Diagram

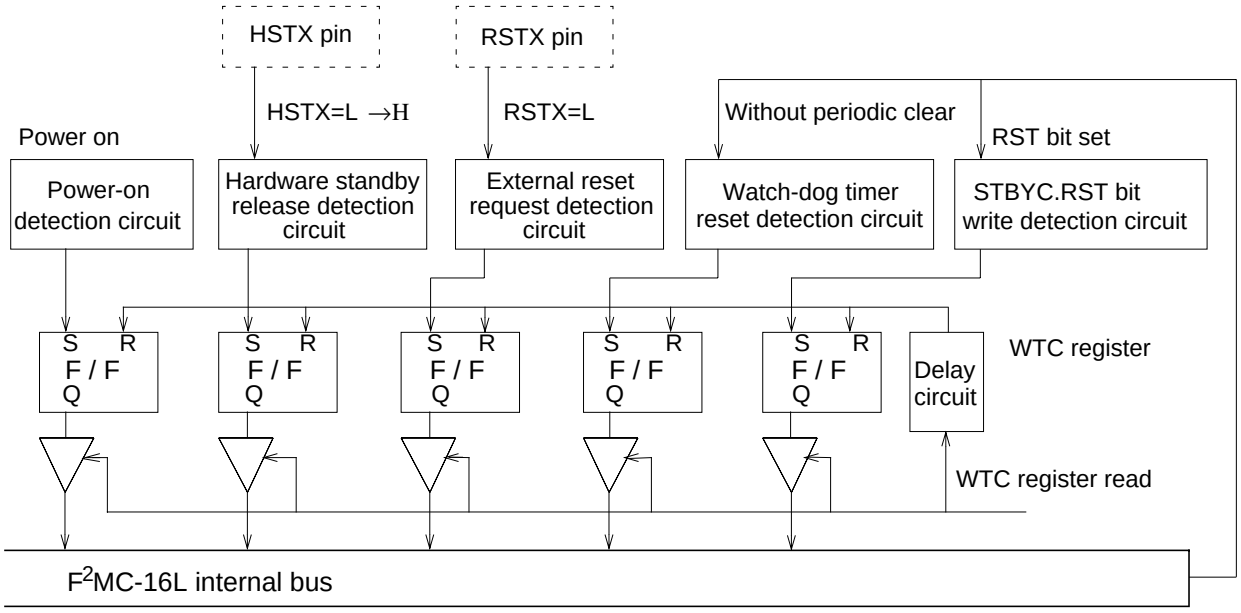


Figure 5.3-2 WDTC (Watch-Dog Timer Control) Register

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: 0000A8$_H$	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	WDTC
Read/write ⇨	(R)	(R)	(R)	(R)	(R)	(W)	(W)	(W)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

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

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

Table 5.3-2 Reset Cause Bits

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

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

CHAPTER 6 LOW-POWER CONTORL CIRCUIT

This chapter explains the functions and operations of the low-power control circuits.

6.1 Outline of Low-Power Control Circuit

6.2 Registers

6.3 Low-Power Mode Operation

6.4 Intermittent CPU Operation

6.5 Switching Machine Clocks

6.6 Status Transition of Clock Selection

6.1 Outline of Low-Power Control Circuit

The MB90590 Series supports various operation modes to help reduce the power dissipation.
The operation modes include PLL clock mode, PLL sleep mode, timer mode, main clock mode, main sleep mode, stop mode, and hardware standby mode. Modes other than PLL clock mode are classified as low-power modes.

■ Outline of Lower-power Control Circuit

In main clock mode or main sleep mode, the main clock (OSC oscillation clock) is used. The operation clock is generated by dividing the main clock by two, and the PLL clock (VCO oscillation clock) is stopped.

In PLL sleep mode or main sleep mode, only the CPU operation clock is stopped. All other clocks are in operation.

In timer mode, only the time base timer is in operation.In stop mode or hardware standby mode, oscillation is stopped. The data can be maintained at the lowest power consumption possible.

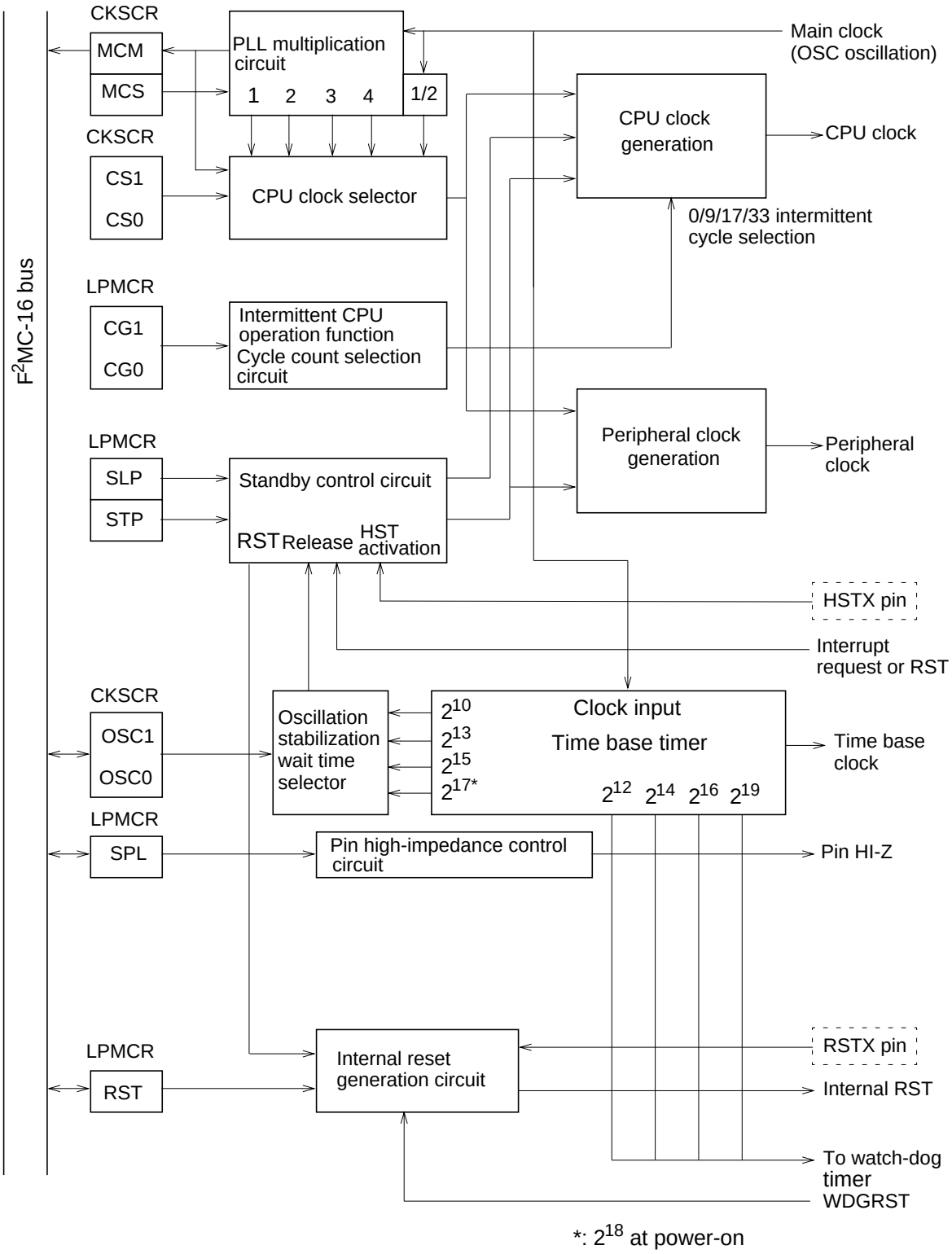
The intermittent CPU operation function is used to intermittently enable the clock supplied to the CPU when a register, internal memory, internal resource, or external bus is accessed. CPU execution is slowed while high-speed clock is supplied to the internal resources, enabling processing at low power consumption.

The PLL clock multiplication factor can be selected from 1, 2, 3, and 4 by setting the CS1 and CS0 bits.

The oscillation stabilization wait time for the main clock upon release of stop or hardware standby mode can be set by the WS1 and WS0 bits.

■ Block Diagram

Figure 6.1-1 Low-power Control Circuit and Clock Generator



6.2 Registers

A low-power control circuit has the following two registers:

- **Low-power mode control register (LPMCR)**
- **Clock selection register (CKSCR)**

■ **Low Power Mode Control Register**

Address:	7	6	5	4	3	2	1	0 ⇐ Bit No.	
$0000A0_H$	STP	SLP	SPL	RST	Reserved	CG1	CG0	Reserved	LPMCR
Read/write ⇨	(W)	(W)	(R/W)	(W)	(-)	(R/W)	(R/W)	(-)	
Initial value ⇨	(0)	(0)	(0)	(1)	(1)	(0)	(0)	(0)	

Clock selection register

Address:	15	14	13	12	11	10	9	8 ⇐ Bit No.	
$0000A1_H$	Reserved	MCM	WS1	WS0	Reserved	MCS	CS1	CS0	CKSCR
Read/write ⇨	(-)	(R)	(R/W)	(R/W)	(-)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(1)	(1)	(1)	(1)	(1)	(1)	(0)	(0)	

6.2.1 Low Power Mode Control Register (LPMCR)

In association with the clock selection register, the low-power mode control register sets various operation modes to reduce power consumption.

■ **Low Power Mode Control Register (LPMCR)**

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

[bit 7] STP

Writing '1' to this bit starts the timer mode (CKSCR.MCS=0) or stop mode (CKSCR.MCS=1). Writing '0' performs no operation. This bit is cleared to '0' upon a reset, timer mode release, or stop mode release. This is a write-only bit. '0' is always read from this bit.

[bit 6] SLP

Writing '1' to this bit starts sleep mode. Writing '0' performs no operation. This bit is cleared to '0' upon a reset, clock release, or stop release.

Writing '1' to the STP and SLP bits simultaneously starts clock or stop mode. This is a write-only bit. '0' is always read from this bit.

[bit 5] SPL

When '0' is written to this bit, the external pin level in timer mode or stop mode is maintained. When '1' is written to this bit, the external pin in clock or stop mode is set to high impedance. This bit is cleared to '0' upon a reset. This bit is readable and writable.

It is important to note that when SPL is set to 0 and the microcontroller is in the stop mode (STP=1 and MCS=1), all inputs must be provided with stable digital values. Otherwise it results in current consumption at the input buffers. (A/D analog inputs are exception)

Generally it is recommended to set the SPL bit to 1 when the microcontroller is in the stop mode inorder to disable all input buffers.

[bit 4] RST

Writing '0' to this bit generates internal reset signals for three machine cycles. Writing '1' performs no operation. '1' is always read from this bit.

[bits 2 and 1] CG1 and CG0

These bits are used to set the clock pause cycle count during intermittent CPU operation.

These bits are initialized to '00' upon a reset by power-on, hardware standby, or watch-dog. These bits are not initialized by any other type of reset. These bits are readable and writable.

The intermittent CPU operation function pauses the clock to the CPU when a register, internal memory, internal resource, or external bus is accessed, thus delaying the activation of the internal bus cycle. CPU execution is slowed while high-speed clock is supplied to an

internal resource, enabling processing at low power consumption.

Table 6.2-1 CG Bit Setting

CG1	CG0	CPU clock pause cycle count
0	0	0 cycle (CPU clock = Resource clock)
0	1	9 cycles (CPU clock: Resource clock = 1:3 to 4 approx.)
1	0	17 cycles (CPU clock: Resource clock = 1:5 to 6 approx.)
1	1	33 cycles (CPU clock: Resource clock = 1:9 to 10 approx.)

6.2.2 Clock Selection Register (CKSCR)

The clock selection register sets and controls the CPU machine clock, and sets the oscillation stabilization wait time when power is turned on or oscillation is restored.

■ **Clock Selection Register (CKSCR)**

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

[bit 14] MCM

This bit indicates whether the main clock or PLL clock is selected as the machine clock. '0' indicates that the PLL clock is selected, and '1' indicates that the main clock is selected. When MCS=0 and MCM=1, the system is waiting for the PLL clock oscillation to stabilize. The PLL clock oscillation stabilization wait time is fixed to 2^{13} main clock cycles.

[bits 13 and 12] WS1 and WS0

These bits are used to set the main clock oscillation stabilization wait time upon release of stop or hardware standby mode.

These bits are initialized to '11' upon a power-on reset. These bits are not initialized by any other type of reset. These bits are readable and writable.

Table 6.2-2 WS Bit Setting

WS1	WS0	Oscillation stabilization wait time (at 4 MHz source oscillation)
0	0	Approx. 256μs (2^{10} counts of source oscillation)
0	1	Approx. 2.05 ms (2^{13} counts of source oscillation)
1	0	Approx. 8.19 ms (2^{15} counts of source oscillation)
1	1	Approx. 32.77 ms (2^{17} counts of source oscillation)

*: Approx. 65.54ms (2^{18} counts of source oscillation) at power-on.

More stabilization time of 2^{17} is added to the default duration only upon with the power-on reset. Therefor, after power-on there will be about 65.54ms of the stabilization time.

[bit 10] MCS

This bit is used to select the main clock or PLL clock as the machine clock. Writing '0' selects the PLL clock and writing '1' selects the main clock. When this bit is updated from '1' to '0,' the PLL clock oscillation stabilization wait period is created by automatically clearing the time base timer and the TBOF bit of the time base timer control register. The oscillation stabilization wait time for the PLL clock is fixed to 2^{13} main clock cycles. (The oscillation wait time is about 2 ms at 4 MHz source oscillation.)

When the main clock is selected, the operation clock is generated by dividing the main clock by two. (The operation clock is 2 MHz at 4 MHz source oscillation.)

This bit is initialized to '1' by the power-on reset, hardware standby, or watch-dog reset. But it is not initialized by the external reset from the RSTX pin or by the software reset (the RST bit in the LPMCR register).

Note:

When updating the MCS bit from '1' to '0,' ensure that the time base timer interrupt is masked by the TBIE bit or the ILM bit of the CPU.

[bits 9 and 8] CS1 and CS0

These bits are used to select the multiplication factor of the PLL clock.

These bits are initialized to '00' upon a power-on reset. These bits are not initialized by any other type of reset.

Write is disabled when '0' is written to the MCS bit. To update the CS bit, set '1' in the MCS bit (to start main clock mode).

These bits are readable and writable.

Table 6.2-3 CS Bit Setting

CS1	CS0	Machine clock (at 4 MHz source oscillation)
0	0	4 MHz (Operation frequency = OSC oscillation frequency)
0	1	8 MHz (Operation frequency = OSC oscillation frequency *2)
1	0	12 MHz (Operation frequency = OSC oscillation frequency *3)
1	1	16 MHz (Operation frequency = OSC oscillation frequency *4)

Note:

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

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

6.3 Low-Power Mode Operation

Table 6.3-1 lists the chip status in each operation mode.

■ Low-power Mode Operation

Table 6.3-1 Low-power mode status

	Transition condition	Oscillation & T.B.T	PLL	Regulator	CPU	Watch Timer	Other Peripheral	Pin	Release method
Main sleep	MCS=1 SLP=1	Operat-ing	Stopped	Operat-ing	Stopped	Operat-ing	Operat-ing	Operat-ing	Reset Interrupt
PLL sleep	MCS=0 SLP=1	Operat-ing	Operat-ing	Operat-ing	Stopped	Operat-ing	Operat-ing	Operat-ing	Reset Interrupt
Timer (SPL=0)	MCS=0 STP=1	Operat-ing	Stopped	Operat-ing	Stopped	Operat-ing	Stopped	Held	Reset Interrupt
Timer (SPL=1)	MCS=0 STP=1	Operat-ing	Stopped	Operat-ing	Stopped	Operat-ing	Stopped	HI-Z	Reset Interrupt
Stop (SPL=0)	MCS=1 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	Stopped	Held*	Reset Interrupt
Stop (SPL=1)	MCS=1 STP=1	Stopped	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	Reset Interrupt
Hardware standby	HSTX=L	Stopped	Stopped	Stopped	Stopped	Stopped	Stopped	HI-Z	HSTX=H

Note: When the SPL is set to 0 in the stop mode, all inputs must be provided with stable digital values. Otherwise it results in current consumption at the input buffers. (A/D analog inputs are exception)

■ **Note: Low-power Mode Control Register Access**

Writing data to the low-power mode control register starts low-power mode (stop or sleep mode). In this case, use an instruction shown in Table 6.3-2. If any other instruction is used to start low-power mode, misoperation may result. Any instruction can be used to control functions other than transition of the low-power mode control register to low-power mode.

To write data to the low-power mode control register in word length, ensure that the data is written to an even-number address. If low-power mode is started by writing data to an odd-number address, misoperation may result.

Table 6.3-2 List of Instructions Used for Transition to Low-power Mode

MOV io,#imm8
MOV io,A
MOV RLi+dip8,A
MOVW io,#imm16
MOVW io,A
MOVW RLi+dip8,A
	MOV dir,#imm8
	MOV dir,A
	MOVP addr24,A
	MOVW dir,#imm16
	MOVW dir,A
	MOPW addr24,A
		MOV eam,#imm8
		MOV addr16,A
		
		MOVW eam,#imm16
		MOVW addr16,A
			MOV eam,#immRi
			MOV eam,A
			
			MOVW eam,RWi
			MOVW eam,A

SETB io:bp	SETB dir:bp	SETB addr16:bp	

6.3.1 Sleep Mode

In sleep mode, only the clock supplied to the CPU is stopped. As a result, the CPU terminates while peripheral circuits keep operating.

■ Transition to Sleep Mode

The standby control circuit is set in sleep mode by writing '1' to the SLP bit and '0' to the STP bit of the low power mode control register. In sleep mode, only the clock supplied to the CPU is stopped. The CPU stops, and the peripheral circuits continue operation.

If an interrupt request has been issued when '1' is written to the SLP bit, the standby control circuit does not enter sleep mode. Therefore, the CPU executes the next instruction if the interrupt cannot be accepted, or immediately branches to the interrupt processing routine if the interrupt can be accepted.

In sleep mode, the values of special registers such as the accumulator and the internal RAM are maintained. The external bus hold function works even in sleep mode. If there is a hold request, the hold status is entered.

■ Releasing Sleep Mode

The standby control circuit releases sleep mode in the event of a reset input or an interrupt. If sleep mode is released by a reset, the reset status takes effect after sleep mode is released.

If a peripheral circuit or similar issues an interrupt request of a higher interrupt level than 7 in sleep mode, the standby control circuit releases sleep mode. After sleep mode is released, processing is handled as normal interrupt processing. The CPU executes interrupt processing if the interrupt can be accepted according to the I flag, ILM, and the interrupt control register (ICR). If the interrupt cannot be accepted, processing continues from the instruction following the instruction that was being executed before the transition to sleep mode.

Note:

Usually, interrupt processing is started after the instruction following the instruction that was executed during the transition to sleep mode. If, however, transition to sleep mode and acceptance of the external bus hold request are simultaneous, interrupt processing may start before the next instruction is executed.

6.3.2 Timer Mode

Timer mode stops operations other than the source oscillation, time base timer, and watch-dog timer, resulting in almost all functions of the chip being stopped.

■ Transition to Timer Mode

The standby control circuit is set to timer mode when the MCS bit of the clock selection register is '0' and '1' is written to the STP bit of the low-power mode control register. In timer mode, all operations are stopped except for the source oscillation and time base timer. Most functions of the chip stop.

Using the SPL bit of the low-power mode control register, the I/O pin may be maintained at the immediately preceding status or at high impedance in timer mode.

If an interrupt request has been issued when '1' is written to the STP bit, the standby control circuit does not enter timer mode.

In timer mode, the values of special registers such as the accumulator and the internal RAM are maintained. The external bus hold function is disabled in timer mode. No hold request is accepted. If a hold request is input during transition to timer mode, the HAKX signal may not become 'L' while the bus is Hi-Z.

■ Releasing Timer Mode

The standby control circuit releases timer mode in the event of a reset input or an interrupt. If timer mode is released by a reset, the reset status takes effect after timer mode is released.

To return from timer mode, the standby control circuit initially releases timer mode, then enters the PLL clock oscillation stabilization wait state. The MCS bit is not cleared by an external reset, so the reset sequence is performed using the main clock if the reset period is shorter than the PLL clock oscillation stabilization wait period. The PLL clock oscillation stabilization wait period is 2^{13} to $3*2^{13}$ main clock cycles depending on the time base timer status, because the time base timer is not cleared.

If a peripheral circuit or similar issues an interrupt request of a higher interrupt level than 7 in timer mode, the standby control circuit releases timer mode. After the timer mode is released, processing is handled as normal interrupt processing. The CPU executes interrupt processing if the interrupt can be accepted according to the I flag, ILM, and the interrupt control register (ICR). If the interrupt cannot be accepted, processing continues from the instruction following the instruction that was being executed during transition to timer mode.

Note:

Usually, interrupt processing is started after the instruction following the instruction that was being executed during the transition to timer mode. If, however, transition to timer mode and acceptance of the external bus hold request are simultaneous, interrupt processing may start before the next instruction is executed.

The standby control circuit enters PLL clock oscillation stabilization wait status when timer mode is released. If the PLL clock is not used, write '1' to the MCS bit by an instruction immediately following the reset or interrupt.

6.3.3 Stop Mode

Stop mode stops the source oscillation, resulting in all functions of the chip being stopped. Data can be maintained at the lowest power consumption possible.

■ Transition to Stop Mode

The standby control circuit is set to stop mode when the MCS bit of the clock selection register is '1' and '1' is written to the STP bit of the low-power mode control register. In stop mode, the source oscillation is stopped and all functions of the chip are stopped. Therefore, data can be maintained at the lowest power consumption possible.

Using the SPL bit of the LPMCR, the I/O pins can be maintained at the immediately preceding status or at high impedance in stop mode. When the SPL bit is set to 0, all inputs must be provided with stable digital values. Otherwise it results in current consumption at the input buffers. (A/D analog inputs are exception)

If an interrupt request has been issued when '1' is written to the STP bit, the standby control circuit does not enter the stop mode.

In stop mode, the values of special registers such as the accumulator and the internal RAM are maintained. The external bus hold function is disabled in stop mode. No hold request is accepted. If a hold request is input during the transition to stop mode, the HAKX signal may not become 'L' while the bus is Hi-Z.

■ Releasing Stop Mode

The standby control circuit releases stop mode in the event of a reset input or an interrupt. If stop mode is released by a reset, the reset status takes effect after stop mode is released.

To return from stop mode, the standby control circuit initially enters the PLL clock oscillation stabilization wait status, and then releases stop mode. Even if stop mode is released by a reset, the reset sequence is executed after the oscillation stabilization wait period.

If a peripheral circuit or similar issues an interrupt request of a higher interrupt level than 7 in stop mode, the standby control circuit releases stop mode. After stop mode is released, the processing is handled as normal interrupt processing after the main clock oscillation stabilization wait period specified by the WS1 and WS0 bits of CKSCR. The CPU executes interrupt processing if the interrupt can be accepted according to the I flag, ILM, and the interrupt control register (ICR). If the interrupt cannot be accepted, processing continues from the instruction following the instruction that was being executed during transition to stop mode.

■ Setting the Oscillation Stabilization Wait Time

Use the WS1 and WS0 bits to specify the oscillation stabilization wait time when stop mode or hardware standby mode is released. Specify the oscillation stabilization wait time according to the types and characteristics of the oscillator circuit and oscillator device connected to the X0 and X1 pins.

These bits are not initialized upon a reset, except for a power-on reset. Upon a power-on reset, these bits are initialized to '11.' Therefore, at power-on, the oscillation stabilization wait time is about 2^{17} counts of source oscillation.

Note:

Usually, interrupt processing is started after the instruction following the instruction that was being executed during the transition to stop mode. If, however, transition to stop mode and

acceptance of the external bus hold request are simultaneous, interrupt processing may start before the next instruction is executed.

6.3.4 Hardware Standby Mode

In the hardware standby mode, oscillation is stopped and all I/O pins are set to high impedance while the HSTX pin is at "L" level, regardless of other statuses (including reset).

■ Transition to Hardware Standby Mode

The standby control circuit can be set in hardware standby mode from any status by setting the HSTX pin at 'L' level. In hardware standby mode, oscillation is stopped and all I/O pins are set to high impedance while the HSTX pin is at 'L' level, regardless of other status including reset.

In hardware standby mode, the internal RAM contents are maintained but the special registers such as the accumulator are initialized.

■ Releasing Hardware Standby Mode

Hardware standby mode can be released only by the HSTX pin. When the HSTX pin is set at 'H' level, the standby control circuit releases hardware standby mode, enables the internal reset signal, and enters oscillation stabilization wait status. After the oscillation stabilization wait period, the standby control circuit releases the internal reset, and consequently the CPU starts execution from the reset sequence.

■ Setting the Oscillation Stabilization Wait time

Use the WS1 and WS0 bits to specify the oscillation stabilization wait time when stop mode or hardware standby mode is released. Specify the oscillation stabilization wait time according to the types and characteristics of the oscillator circuit and oscillator device connected to the X0 and X1 pins.

These bits are not initialized upon a reset, except for a power-on reset. Upon a power-on reset, these bits are initialized to '11.' Therefore, at power-on, the oscillation stabilization wait time is about 2^{17} counts of source oscillation.

6.4 Intermittent CPU Operation

The intermittent CPU operation function pauses the clock supplied to the CPU when a register, internal memory (ROM, RAM, I/O, or resource), or external bus is accessed, delaying the activation of the internal bus cycle. The CPU execution speed is decreased while a high-speed clock is supplied to internal resources, thus enabling processing at low power consumption.

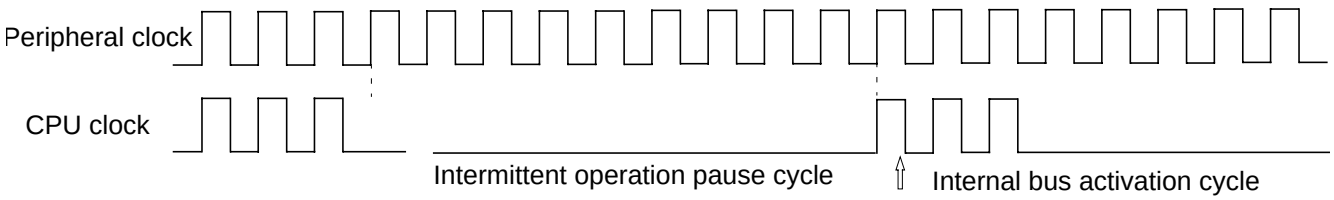
■ Intermittent CPU Operation

Figure 6.4-1 is a diagram of intermittent CPU operation. For intermittent CPU operation, the CG1 and CG0 bits are used to select the cycle count for clock pausing.

The external bus operation itself is performed using the same clock as that used for the resources.

An instruction execution time using the intermittent CPU operation function can be obtained by adding a compensation value to the ordinary execution time. The compensation value is obtained by multiplying the number of accesses to a register, internal memory, internal resource, or external bus by the cycle count for pausing.

Figure 6.4-1 Intermittent CPU Operation



6.5 Switching Machine Clocks

Writing to the MCS bit in the CKSCR register switches the machine clock from the main clock to the PLL clock.

■ Switching between Main Clock and PLL Clock

Write data to the MCS bit of the CKSCR register to switch between the main clock and PLL clock.

When the MCS bit is changed from '1' to '0,' the PLL clock takes over the main clock after the PLL clock oscillation stabilization wait time (2^{13} machine clock cycles).

When the MCS bit is changed from '0' to '1,' the main clock takes over the PLL clock when the edges of the PLL and main clocks match (after about 1 to 8 PLL clock cycles).

Writing to the MCS bit does not change the machine clock immediately. To manipulate a resource that depends on the machine clock, always reference the MCM bit before hand to check that the machine clock has been switched.

■ Initializing the Machine Clock

The MCS bit is not initialized by a reset using an external pin or RST bit. The MCS bit is initialized to '1' by any other reset.

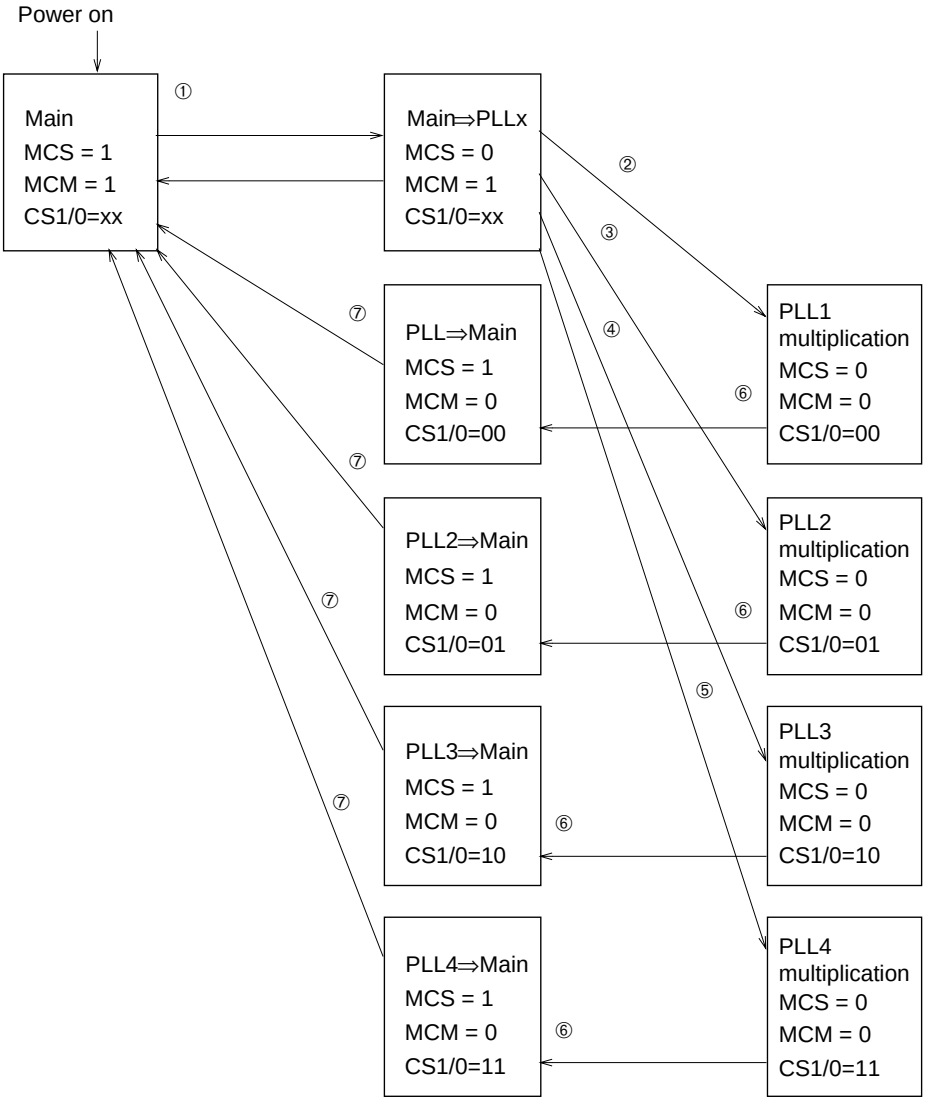
6.6 Status Transition of Clock Selection

The oscillation stabilization wait time for the PLL clock is fixed at 2^{13} main clock cycles. (The oscillation wait time is about 2 ms at a source oscillation of 4 MHz.)

■ Status Transition of Clock Selection

Figure 6.6-1 is a diagram of status transition of clock selection.

Figure 6.6-1 Status Transition of Clock Selection



① MCS bit clear
② End of PLL clock oscillation stabilization wait & CS1/0=00
③ End of PLL clock oscillation stabilization wait & CS1/0=01
④ End of PLL clock oscillation stabilization wait & CS1/0=10
⑤ End of PLL clock oscillation stabilization wait & CS1/0=11
⑥ MCS bit set (including hardware standby and watch-dog reset)
⑦ Synchronization timing between PLL clock and main clock

CHAPTER 7 MEMORY ACCESS MODES

This chapter explains the functions and operations of the memory access modes.

7.1 Outline of Memory Access Modes

7.2 Mode Pins

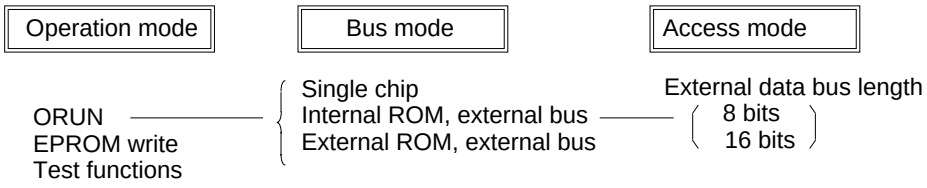
7.3 Mode Data

7.1 Outline of Memory Access Modes

In the F^{2}MC-16LX, the following three memory access modes are provided for each of the access methods, access areas, and tests:

- **Operation mode**
- **Bus mode**
- **Access mode**

■ **Memory Access Modes**



For the MB90590 Series, the external bus function is not supported. Therefor the following part of this document is not fully supported. In user applications, please use the MB90590 Series in the single chip mode.

To set the MB90590 Series into the signle chip mode, the mode inputs (MD2 to 0) should be "011" and the most significant two bits of the mode data (M1 and M0) should be "00".

❍ **Operation mode**

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

❍ **Bus mode**

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

❍ **Access mode**

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

7.2 Mode Pins

Table 7.2-1 describes the operations specified by combinations of the MD2 to MD0 external pins.

■ Mode pins

Table 7.2-1 Mode Pins and Modes

Mode pin setting MD2	MD1	MD0	Mode name	Reset vector access area	External data bus width	Remarks
0	0	0	External vector mode 0	External	8 bits	
0	0	1	External vector mode 1	External	16 bits	Reset vector, 16-bit bus width access
0	1	0	External vector mode 2	External	16 bits	Reset vector, 8-bit bus width access
0	1	1	Internal vector mode	Internal	(Mode data)	Reset sequence and later segments are controlled based on mode data.
1	0	0	Reserved			
1	0	1				
1	1	0				
1	1	1	Flash memory	--	--	

In External vector mode 2, the HMBS bit of the bus control signal selection register is set to '1,' and the bus width for external access to the area between 800000_H and $FFFFFF_H$ is 8 bits. Use External vector mode 2 when the bus width for the ROM is 8 bits but the bus width for the RAM or other components is 16 bits. In External vector mode 1, the HMBS bit is set to '0' and the access bus width becomes 16 bits.

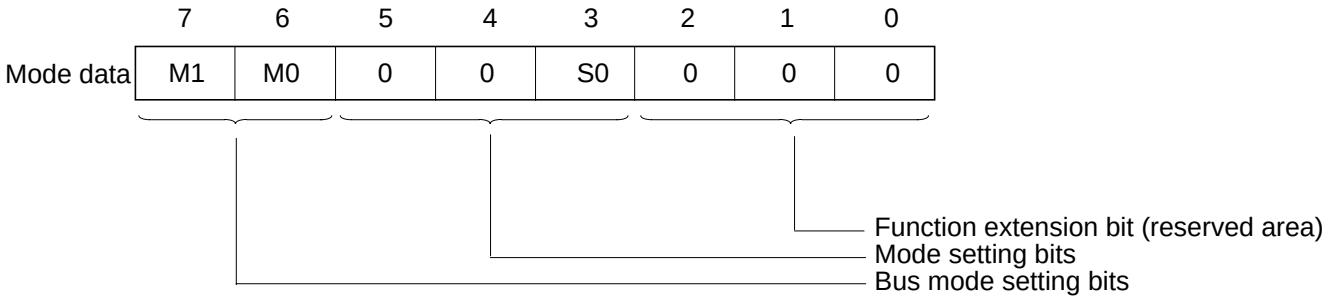
7.3 Mode Data

Mode data is stored at FFFFDF$_H$ of main memory and used for controlling the CPU operation. This data is fetched during a reset sequence and stored in the mode register inside the device. The mode register value can be changed only by a reset sequence.
The setting of this register is valid after the reset sequence.
Always set the reserved bits to '0.'

■ Mode Data

Figure 7.3-1 is a diagram of the setting of the bits.

Figure 7.3-1 Mode Data Structure



■ Mode Setting BitsMode Setting Bits

These bits are used to specify the bus mode or access mode after the reset sequence is completed. Table 7.3-1 shows the relationship between the bits and the functions.

Table 7.3-1 Mode Setting Bits and Functions

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

■ Bus Mode Setting Bits

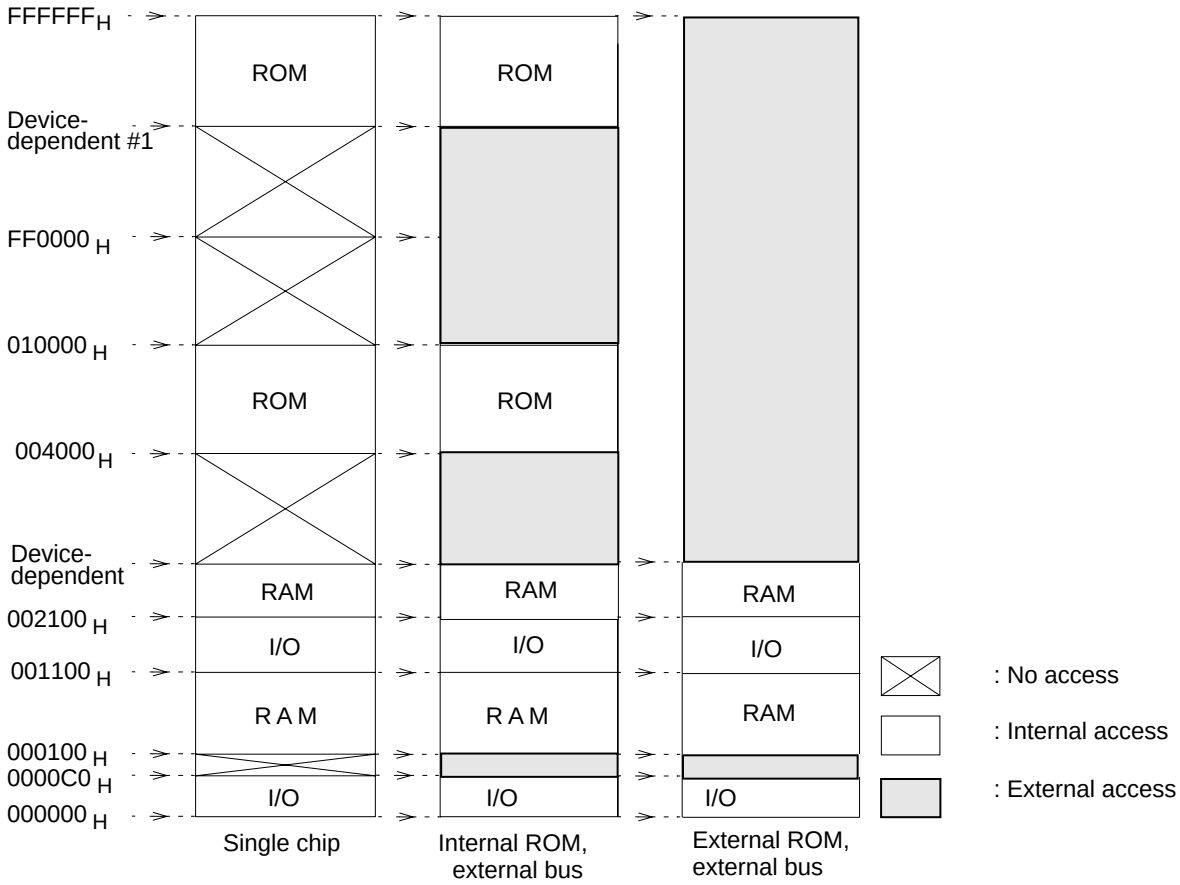
These bits are used to specify the operation mode after the reset sequence is completed. Table 7.3-2 shows the relationship between the bits and the functions.

Table 7.3-2 Bus Mode Setting Bits and Functions

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

Figure 7.3-2 is a diagram of the correspondence between the access areas and physical addresses for each bus mode.

Figure 7.3-2 Access Areas and Physical Addresses in each Bus Mode



Note:

"Device-dependent" means an address that is determined depending on the device.

■ Recommended Setting

Table 7.3-3 lists a sample recommended setting of mode pins and mode data.

Table 7.3-3 Sample Recommended Setting of Mode Pins and Mode Data

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

Table 7.3-3 Sample Recommended Setting of Mode Pins and Mode Data (Continued)

Sample setting	MD2	MD1	MD0	M1	M0	S0
External ROM and external bus mode, 8-bit bus	0	0	0	1	0	0

CHAPTER 8 I/O PORTS

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

8.1 I/O Port

8.2 I/O Port Registers

8.1 I/O Ports

Each pin of the ports can be specified as input or output using the direction register if the corresponding peripheral does not use the pin. When a pin is specified as input, the logic level at the pin is read. When a pin is specified as output, the data register value is read. The above also applies to a read operation for the read-modify-write instructions.
Only for Port 0, Port 1, Port 2 and Port 3, the corresponding bits of the Port Direction registers should be set to "1" in oder to enable peripheral signal outputs.

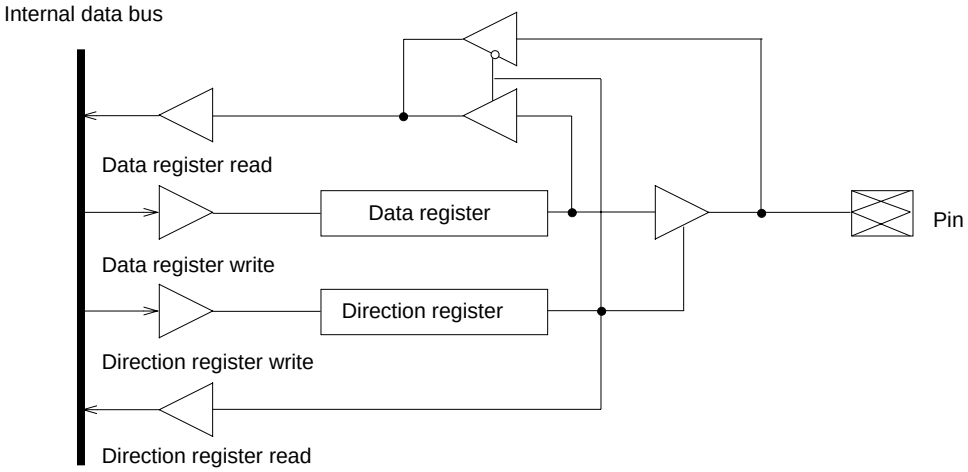
■ I/O Ports

When a pin is used as an output of other peripheral function, the peripheral output value is read regardless of the direction register value.

It is generally recommended that the read-modify-write instructions should not be used for setting the data register prior to setting the port as an output. This is because the read-modify-write instruction in this case results reading the logic level at the port rather than the register value.

Figure 8.1-1 is a block diagram of the I/O ports.

Figure 8.1-1 I/O Port Block Diagram



8.2 I/O Port Registers

There are three types of I/O port registers:

- **Port data register (PDR0 to 9)**
- **Port direction register (DDR0 to 9)**
- **Analog input enable register (ADER)**

■ I/O Port Registers

Figure 8.2-1 shows the I/O port registers.

Figure 8.2-1 I/O Port Registers

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000000 H	P07	P06	P05	P04	P03	P02	P01	P00	Port 0 data register (PDR0)
Address : 000001 H	P17	P16	P15	P14	P13	P12	P11	P10	Port 1 data register (PDR1)
Address : 000002 H	P27	P26	P25	P24	P23	P22	P21	P20	Port 2 data register (PDR2)
Address : 000003 H	P37	P36	P35	P34	P33	P32	P31	P30	Port 3 data register (PDR3)
Address : 000004 H	P47	P46	P45	P44	P43	P42	P41	P40	Port 4 data register (PDR4)
Address : 000005 H	P57	P56	P55	P54	P53	P52	P51	P50	Port 5 data register (PDR5)
Address : 000006 H	P67	P66	P65	P64	P63	P62	P61	P60	Port 6 data register (PDR6)
Address : 000007 H	P77	P76	P75	P74	P73	P72	P71	P70	Port 7 data register (PDR7)
Address : 000008 H	P87	P86	P85	P84	P83	P82	P81	P80	Port 8 data register (PDR8)
Address : 000009 H	—	—	P95	P94	P93	P92	P91	P90	Port 9 data register (PDR9)

Bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0	
Address : 000010 H	D07	D06	D05	D04	D03	D02	D01	D00	Port 0 direction register (DDR0)
Address : 000011 H	D17	D16	D15	D14	D13	D12	D11	D10	Port 1 direction register (DDR1)
Address : 000012 H	D27	D26	D25	D24	D23	D22	D21	D20	Port 2 direction register (DDR2)
Address : 000013 H	D37	D36	D35	D34	D33	D32	D31	D30	Port 3 direction register (DDR3)
Address : 000014 H	D47	D46	D45	D44	D43	D42	D41	D40	Port 4 direction register (DDR4)
Address : 000015 H	D57	D56	D55	D54	D53	D52	D51	D50	Port 5 direction register (DDR5)
Address : 000016 H	D67	D66	D65	D64	D63	D62	D61	D60	Port 6 direction register (DDR6)
Address : 000017 H	D77	D76	D75	D74	D73	D72	D71	D70	Port 7 direction register (DDR7)
Address : 000018 H	D87	D86	D85	D84	D83	D82	D81	D80	Port 8 direction register (DDR8)
Address : 000019 H	—	—	D95	D94	D93	D92	D91	D90	Port 9 direction register (DDR9)

Bit	15	14	13	12	11	10	9	8	
Address : 00001B H	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Port 6 analog input enable register (ADER)

8.2.1 Port Data Register

Note that R/W for I/O ports differ from R/W for memory in the following points:
Input mode
Read: The level at the corresponding pin is read.
Write: Data is written to an output latch.
Output mode
Read: The data register latch value is read.
Write: Data is written to an output latch and output to the corresponding pin.

■ **Port data Register**

Figure 8.2-2 shows the port data registers.

Figure 8.2-2 Port Data Registers

	7	6	5	4	3	2	1	0	Initial value	Access	
PDR0 Address: 000000 H	P07	P06	P05	P04	P03	P02	P01	P00	Undefined	R/W	***1**

	15	14	13	12	11	10	9	8			
PDR1 Address: 000001 H	P17	P16	P15	P14	P13	P12	P11	P10	Undefined	R/W	*1

	7	6	5	4	3	2	1	0			
PDR2 Address: 000002 H	P27	P26	P25	P24	P23	P22	P21	P20	Undefined	R/W	*1

	15	14	13	12	11	10	9	8			
PDR3 Address: 000003 H	P37	P36	P35	P34	P33	P32	P31	P30	Undefined	R/W	*1

	7	6	5	4	3	2	1	0			
PDR4 Address: 000004 H	P47	P46	P45	P44	P43	P42	P41	P40	Undefined	R/W	*1

	15	14	13	12	11	10	9	8			
PDR5 Address: 000005 H	P57	P56	P55	P54	P53	P52	P51	P50	Undefined	R/W	*1

	7	6	5	4	3	2	1	0			
PDR6 Address: 000006 H	P67	P66	P65	P64	P63	P62	P61	P60	Undefined	R/W	*1

	15	14	13	12	11	10	9	8			
PDR7 Address: 000007 H	P77	P76	P75	P74	P73	P72	P71	P70	Undefined	R/W	*1

	7	6	5	4	3	2	1	0			
PDR8 Address: 000008 H	P87	P86	P85	P84	P83	P82	P81	P80	Undefined	R/W	*1

	15	14	13	12	11	10	9	8			
PDR9 Address: 000009 H	—	—	P95	P94	P93	P92	P91	P90	Undefined	R/W	*1

8.2.2 Port Direction Register

When a pin is used as a port, the corresponding pin is controlled as described below:
0: Input mode
1: Output mode

■ Port Direction Register

Figure 8.2-3 shows the port direction registers.

Figure 8.2-3 Port Direction Registers

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

	15	14	13	12	11	10	9	8	Initial value	Access
DDR1 Address: 000011 H	D17	D16	D15	D14	D13	D12	D11	D10	00000000 B	R/W

	7	6	5	4	3	2	1	0	Initial value	Access
DDR2 Address: 000012 H	D27	D26	D25	D24	D23	D22	D21	D20	00000000 B	R/W

	15	14	13	12	11	10	9	8	Initial value	Access
DDR3 Address: 000013 H	D37	D36	D35	D34	D33	D32	D31	D30	00000000 B	R/W

	7	6	5	4	3	2	1	0	Initial value	Access
DDR4 Address: 000014 H	D47	D46	D45	D44	D43	D42	D41	D40	00000000 B	R/W

	15	14	13	12	11	10	9	8	Initial value	Access
DDR5 Address: 000015 H	D57	D56	D55	D54	D53	D52	D51	D50	00000000 B	R/W

	7	6	5	4	3	2	1	0	Initial value	Access
DDR6 Address: 000016 H	D67	D66	D65	D64	D63	D62	D61	D60	00000000 B	R/W

	15	14	13	12	11	10	9	8	Initial value	Access
DDR7 Address: 000017 H	D77	D76	D75	D74	D73	D72	D71	D70	00000000 B	R/W

	7	6	5	4	3	2	1	0	Initial value	Access
DDR8 Address: 000018 H	D87	D86	D85	D84	D83	D82	D81	D80	00000000 B	R/W

	15	14	13	12	11	10	9	8	Initial value	Access
DDR9 Address: 000019 H	—	—	D95	D94	D93	D92	D91	D90	__000000 B	R/W

8.2.3 Analog Input Enable Register

This register controls the port 6 pins as described below:
0: Port input/output mode
1: Analog input mode
If an external pin is used as an analog input for the A/D converter, the corresponding bit should be set to "1".

■ **Analog Input Enable Register**

Figure 8.2-4 shows the analog input enable register.

Figure 8.2-4 Analog Input Enable Register



CHAPTER 9 TIME BASE TIMER

This chapter explains the functions and operations of the time base timer.

9.1 Outline of Time Base Timer

9.2 Time Base Timer Control Register

9.3 Operations of Time Base Timer

9.1 Outline of Time Base Timer

The time base timer consists of an 18-bit time base counter and a control register. The 18-bit time base counter divides the system clock. The time base timer issues interrupts at specified intervals based on carry signals of the time base counter.

■ Outline of Time Base Timer

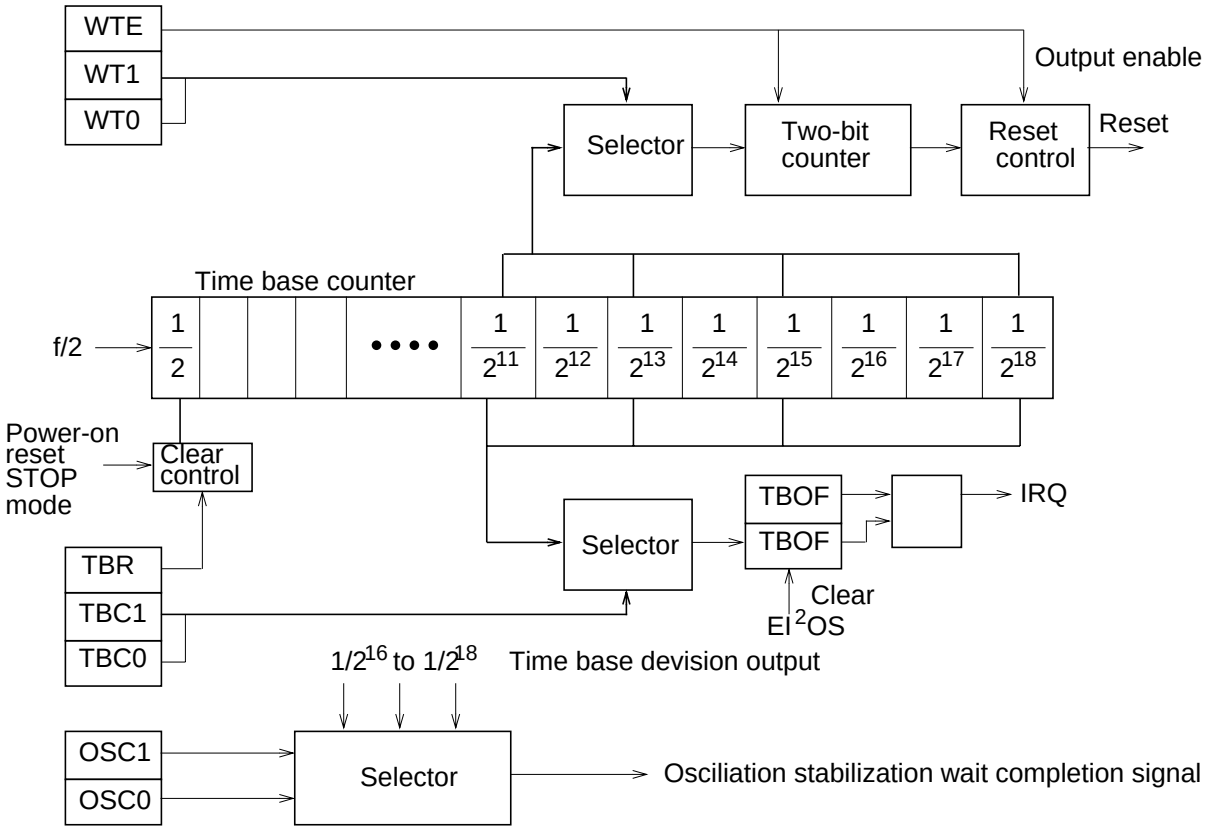
When the power is turned on, the time base counter can be cleared to all zeroes by setting the stop mode or by software (writing '0' to the TBR bit). The time base counter is incremented while the source oscillation is input.

The time base counter can be used as a timer for supplying clock to the watch-dog timer or for waiting for the oscillation to stabilize.

■ Block Diagram of Time Base Timer

Figure 9.1-1 shows a block diagram of the time base timer.

Figure 9.1-1 Block Diagram of Time Base Timer



9.2 Time Base Timer Control Register

The time base timer control register controls interrupts of the time base timer and can clear the time base counter.

■ Time Base Timer Control Register (TBTC)

	bit	15	14	13	12	11	10	9	8	Initial value
TBTC		Reserved	—	—	TBIE	TBOF	TBR	TBC1	TBC0	
Address: 0000A9$_H$		W			R/W	R/W	W	R/W	R/W	1- -00100$_B$

[bit 15] Reserved

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

[bit 12] TBIE

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

[bit 11] TBOF

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

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

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

[bit 10] TBR

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

Writing '0' clears the time base counter.

Writing '1' has no effect.

'1' is always read from this bit.

[bits 9 and 8] TBC1 and TBC0

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

Table 9.2-1 lists the specifiable intervals.

Table 9.2-1 Selecting the Time Base Timer Interval

TBC1	TBC0	Interval at 4 MHz source oscillation
0	0	1.024 ms
0	1	4.096 ms

Table 9.2-1 Selecting the Time Base Timer Interval (Continued)

TBC1	TBC0	Interval at 4 MHz source oscillation
1	0	16.384 ms
1	1	131.072 ms

9.3 Operations of Time Base Timer

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

■ Time Base Counter

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

■ Interval Interrupt Function

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

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

CHAPTER 10 WATCH-DOG TIMER

This chapter explains the functions and operations of the watch-dog timer.

10.1 Outline of Watch-Dog Timer

10.2 Watch-dog Timer Operations

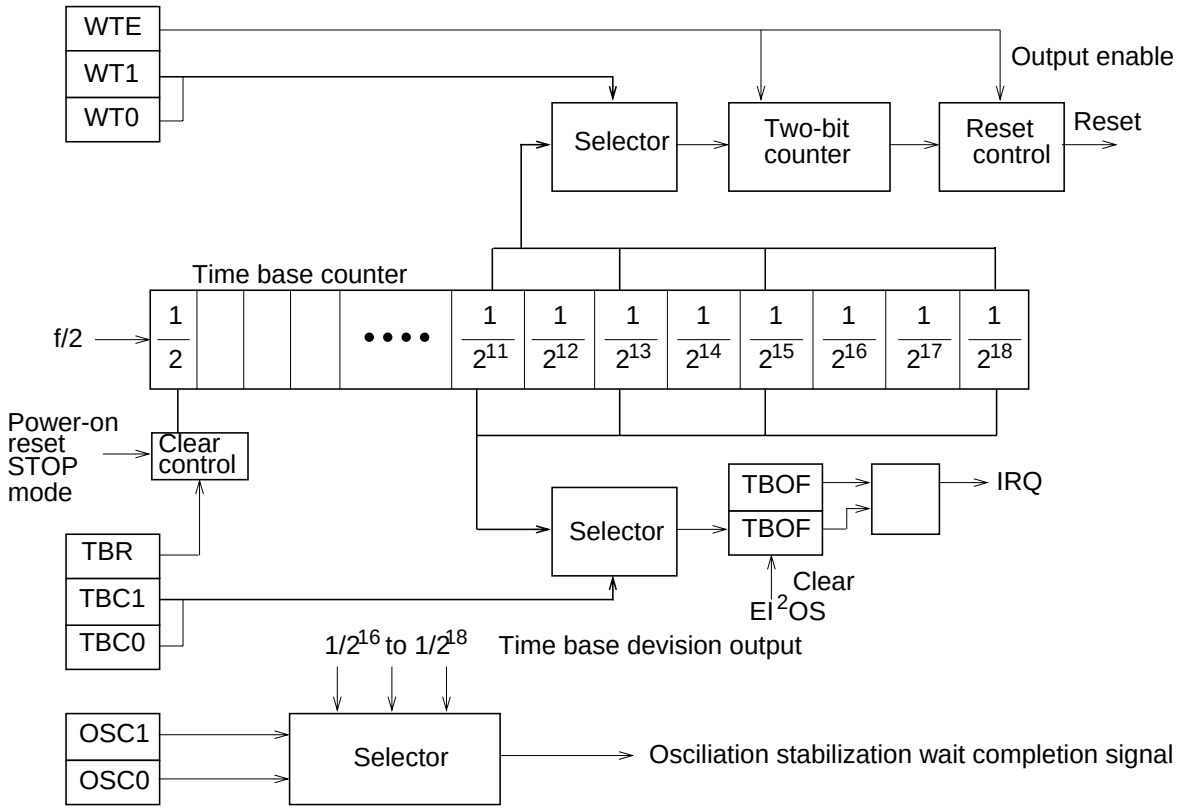
10.1 Outline of Watch-Dog Timer

The watch-dog timer consists of a two-bit watch-dog counter, control register, and watch-dog reset controller. The two-bit watch-dog counter uses the carry signals of an 18-bit time base counter as a clock source.

■ Watch-dog Timer Block Diagram

Figure 10.1-1 is a diagram of the configuration of the watch-dog timer.

Figure 10.1-1 Watch-dog Timer Block Diagram



■ Watch-dog Timer Control Register (WDTC)

	Bit	7	6	5	4	3	2	1	0	Initial value
WDTC Address : 0000A8$_H$		PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	XXXXX111$_H$
		R	R	R	R	R	W	W	W	

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

These flags indicate the reset causes. The flags are set upon a reset as described in Table 10.1-1.

All bits are cleared to '0' after the WDTC register is read. These bits are read-only bits. For details, see Section 5.2 "Reset".

Table 10.1-1 Reset Cause Registers

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

(*: The previous value is maintained.)

[bit 2] WTE

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

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

[bits 1 and 0] WT1 and WT0

These bits are used to select the watch-dog timer interval. Only the data items written during watch-dog timer activation are valid. Data items that are written outside watch-dog timer activation are ignored. Table 10.1-2 lists the interval settings.

These bits are write-only bits.

Table 10.1-2 Watch-dog Timer Interval Selection Bit

WT1	WT0	Interval (at a source oscillation of 4 MHz)		Main clock cycle count
		Minimum	Maximum	
0	0	approx. 3.58 ms	approx. 4.61 ms	2^{14} plus or minus 2^{11} cycles
0	1	approx. 14.33 ms	approx. 18.43 ms	2^{16} plus or minus 2^{13} cycles
1	0	approx. 57.23 ms	approx. 73.73 ms	2^{18} plus or minus 2^{15} cycles

Table 10.1-2 Watch-dog Timer Interval Selection Bit (Continued)

WT1	WT0	Interval (at a source oscillation of 4 MHz)		Main clock cycle count
		Minimum	Maximum	
1	1	approx. 458.7 ms	approx. 589.82 ms	2^{21} plus or minus 2^{18} cycles

Note: The interval becomes the maximum when the time base counter is not reset during watch-dog timer operation.

10.2 Watch-dog Timer Operation

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

■ Activation

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

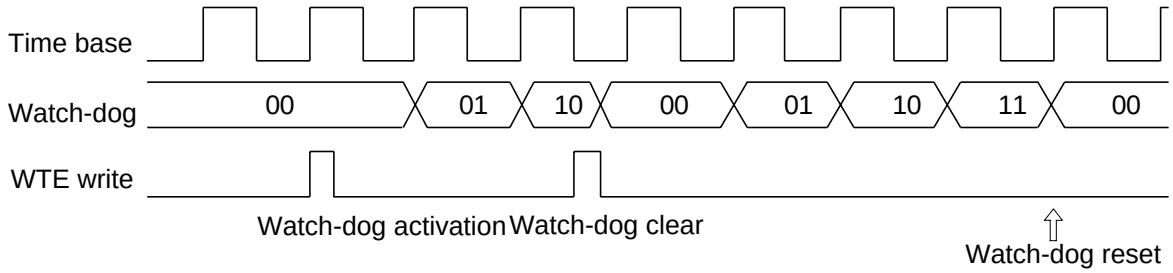
■ Watch-dog Counter

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

The watch-dog counter is cleared not only by writing to the WTE bit but also by a reset, transition to the sleep or stop mode, and a hold acknowledge signal. (The watch-dog counter is not cleared by transition to timer mode.)

Figure 10.2-1 is a diagram of the watch-dog timer operation.

Figure 10.2-1 Watch-dog Timer Operation



■ Watch-dog Stop

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

CHAPTER 11 16-BIT I/O TIMER

This chapter explains the functions and operations of the 16-bit I/O timer.

11.1 Outline of 16-Bit I/O Timer

11.2 16-Bit I/O Timer Registers

11.3 16-bit Free-running Timer

11.4 Output Compare

11.5 Input Capture

11.1 Outline of 16-Bit I/O Timer

The MB90590 Series contains one 16-bit free-running timer module, three output compare modules, and three input capture modules and supports six input channels and six output channels. The following sections only describes the 16-bit free-running timer, Output Compare 0/1 and Input Capture 0/1.
The remaining modules have the identical functions and the register addresses should be found in the I/O map.

■ **16-bit Free-running Timer**

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

- **Four counter clocks are available.**

  Internal clock:ϕ/4, ϕ/16, ϕ/64, ϕ/256

- **An interrupt can be generated upon a counter overflow or a match with compare register 0.**

- **The counter value can be initialized to '0000H' upon a reset, software clear, or match with compare register 0.**

■ **Output Compare (2 Channels per One Module)**

The output compare module consists of two 16-bit compare registers, compare output latch, and control register.

When the 16-bit free-running timer value matches the compare register value, the output level is reversed and an interrupt is issued.

- **The two compare registers can be used independently.**

  Output pins and interrupt flags corresponding to compare registers

- **Output pins can be controlled based on pairs of the two compare registers.**

  Output pins can be reversed by using the two compare registers.

- **Initial values for output pins can be set.**

- **Interrupts can be generated upon a compare match.**

■ **Input Capture (2 Channels per one Module)**

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

- **The detection edge of an external input signal can be specified.**

  Rising, falling, or both edges

- **Two input channels can operate independently.**

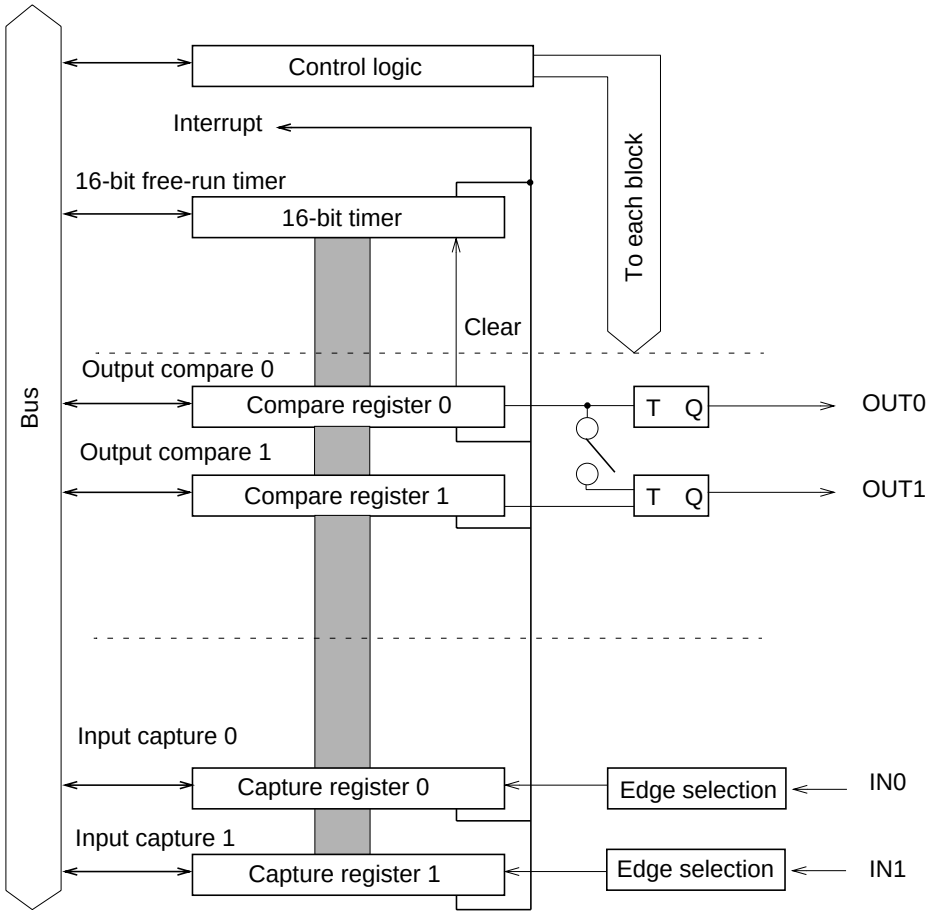
- **An interrupt can be issued upon a valid edge of an external input signal.**

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

■ Block Diagram of 16-bit I/O Timer

Figure 11.1-1 shows a block diagram of the 16-bit I/O timer.

Figure 11.1-1 Block Diagram of 16-bit I/O Timer

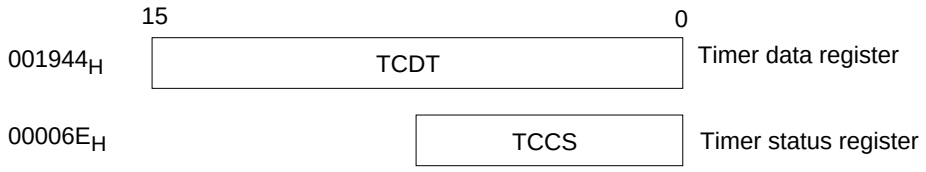


11.2 16-Bit I/O Timer Registers

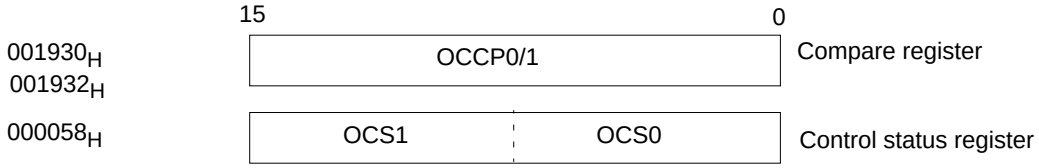
The 16-bit I/O timer has the following three registers:

- **16-bit free-running timer register**
- **16-bit output compare register**
- **16-bit input capture register**

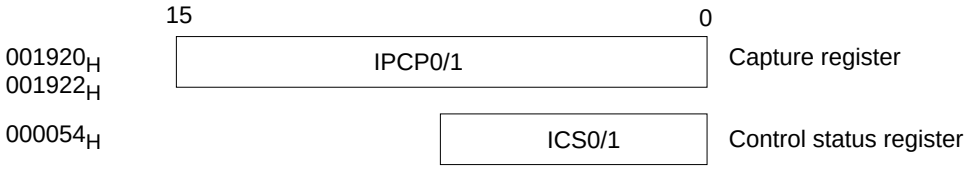
■ **16-bit Free-runningTimer**



■ **16-bit Output Compare**



■ **16-bit Input Capture**



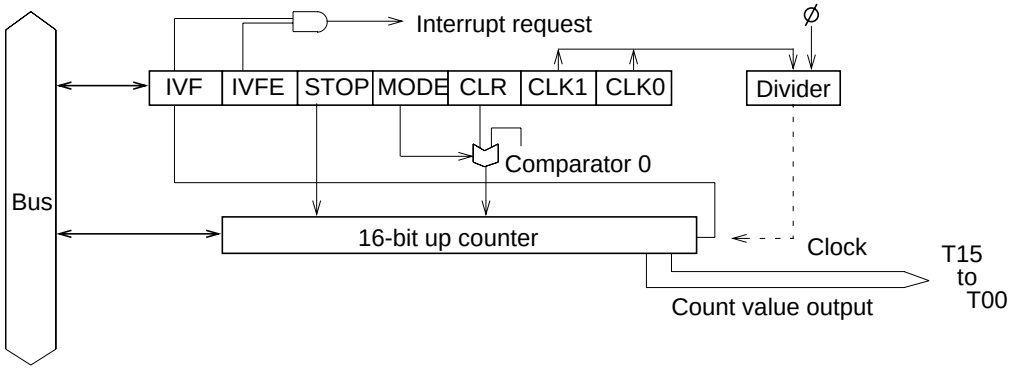
11.3 16-bit Free-running Timer

The 16-bit free-running timer consists of a 16-bit up counter and a control status register. The count values of this timer are used as the base timer for the output compares and input captures.

- **Four counter clock frequencies are available.**
- **An interrupt can be generated upon a counter value overflow.**
- **The counter value can be initialized upon a match with compare register 0, depending on the mode.**

■ 16-bit Free-running Timer Block Diagram

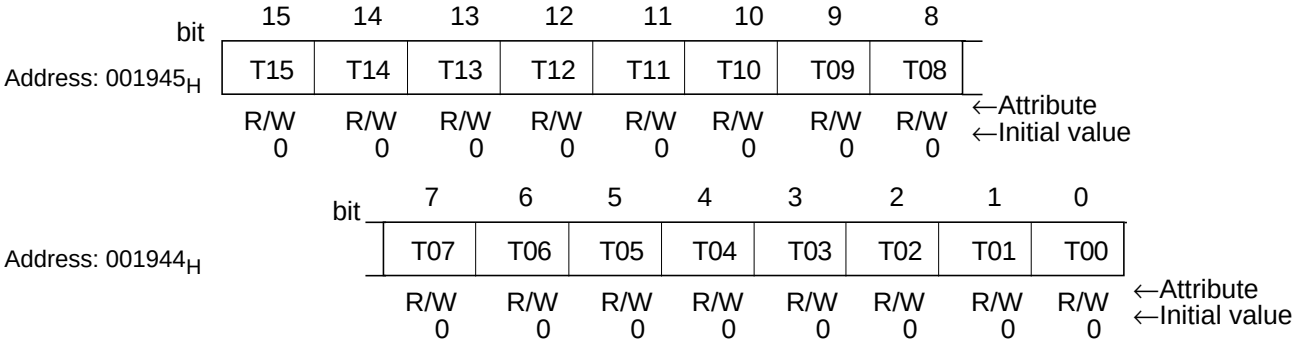
Figure 11.3-1 16-bit Free-running Timer Block Diagram



11.3.1 Data Register

The data register can read the count value of the 16-bit free-running timer. The counter value is cleared to '0000' upon a reset. The timer value can be set by writing a value to this register. However, ensure that the value is written while the operation is stopped (STOP=1).
The data register must be accessed by the word access instructions.

■ **Data Register**



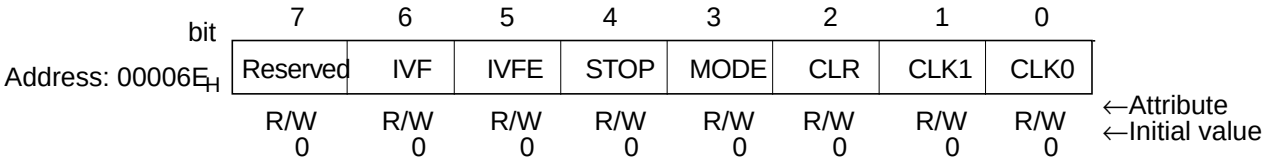
The 16-bit free-running timer is initialized upon the following factors:

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

11.3.2 Control Status Register

The control status register sets the operation mode of the 16-bit free-running timer, starts and stops the 16-bit free-running timer, and controls interrupts.

■ **Control Status Rgister**



[bit 7] Reserved bit

Always write '0' to this bit.

[bit 6] IVF

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

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

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

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

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

0	No interrupt request (initial value)
1	Interrupt request

[bit 5] IVFE

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

0	Interrupt disabled (initial value)
1	Interrupt enabled

[bit 4] STOP

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

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

0	Counter enabled (operation) (initial value)
1	Counter disabled (stop)

Note:

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

[bit 3] MODE

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

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

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

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

Note:

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

[bit 2] CLR

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

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

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

Note:

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

[bits 1 and 0] CLK1 and CLK0

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

CLK1	CLK0	Count clock	ϕ=16 MHz	ϕ=8 MHz	ϕ=4 MHz	ϕ=1 MHz
0	0	ϕ/4	0.25 µs	0.5 µs	1 µs	4 µs
0	1	ϕ/16	1 µs	2 µs	4 µs	16 µs
1	0	ϕ/64	4 µs	8 µs	16 µs	32 µs
1	1	ϕ/256	16 µs	32 µs	64 µs	128 µs

ϕ = Machine clock

11.3.3 16-bit Free-running TimerOperation

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

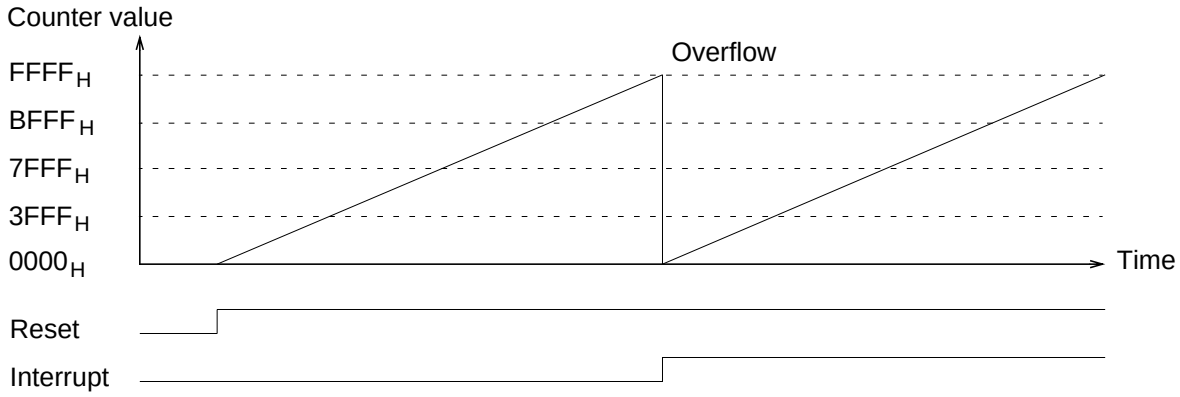
■ 16-bit Free-running Timer Operation

The counter value is cleared in the following conditions:

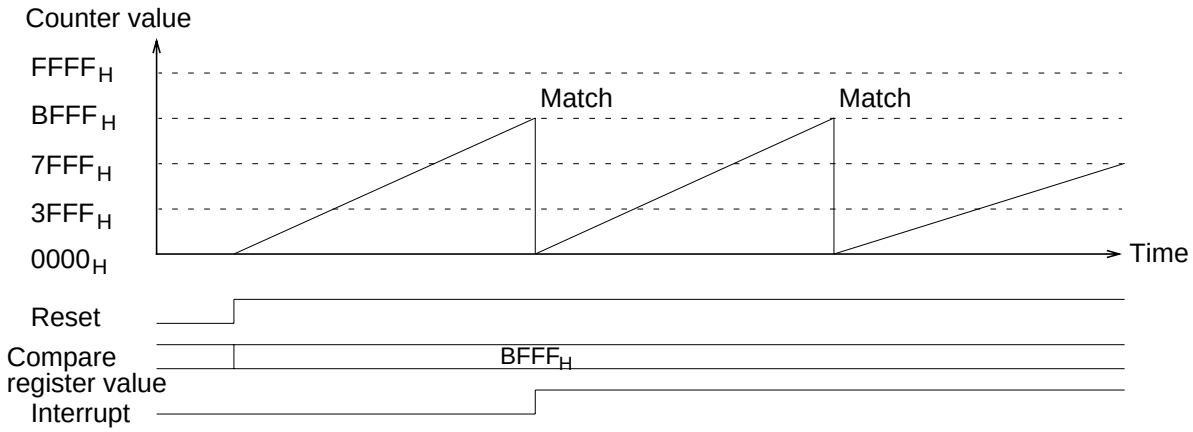
- When an overflow occurs.
- When a match with the output compare register 0 occurs. (This depends on the mode.)
- When '1' is written to the CLR bit of the TCCS register during operation.
- When '0000' is written to the TCDC register during stop.
- Reset

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

■ Clearing the Counter by an Overflow



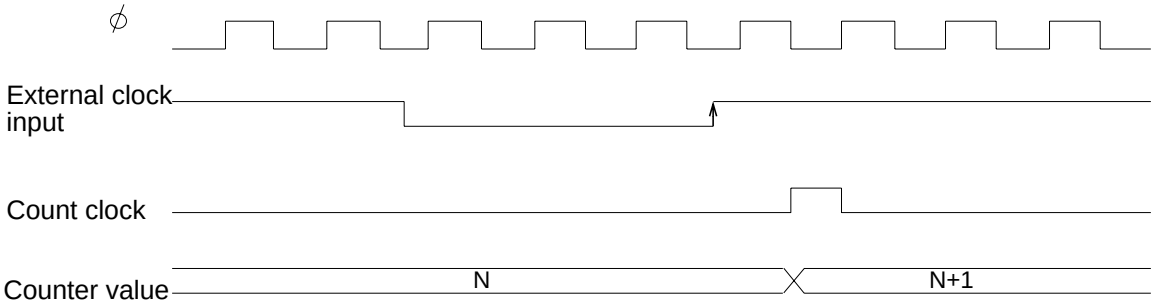
■ Clearing the Counter upon a Match with Output Compare Register 0



■ 16-bit Free-running Timer Timing

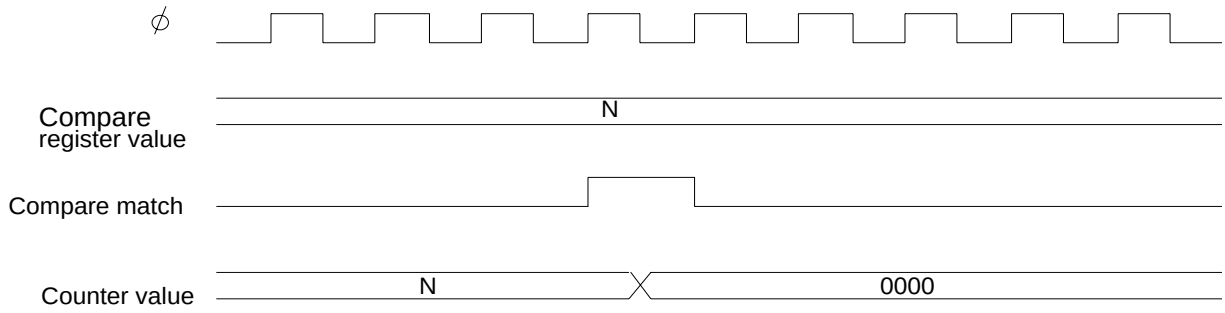
❍ 16-bit free-running timer count timing

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



❍ 16-bit free-running timer clear timing (match with the compare register 0)

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



11.4 Output Compare

The output compare module consists of two 16-bit compare registers, two compare output pins, and control register. If the value written to the compare register of this module matches the 16-bit free-running timer value, the output level of the pin can be reversed and an interrupt can be issued.

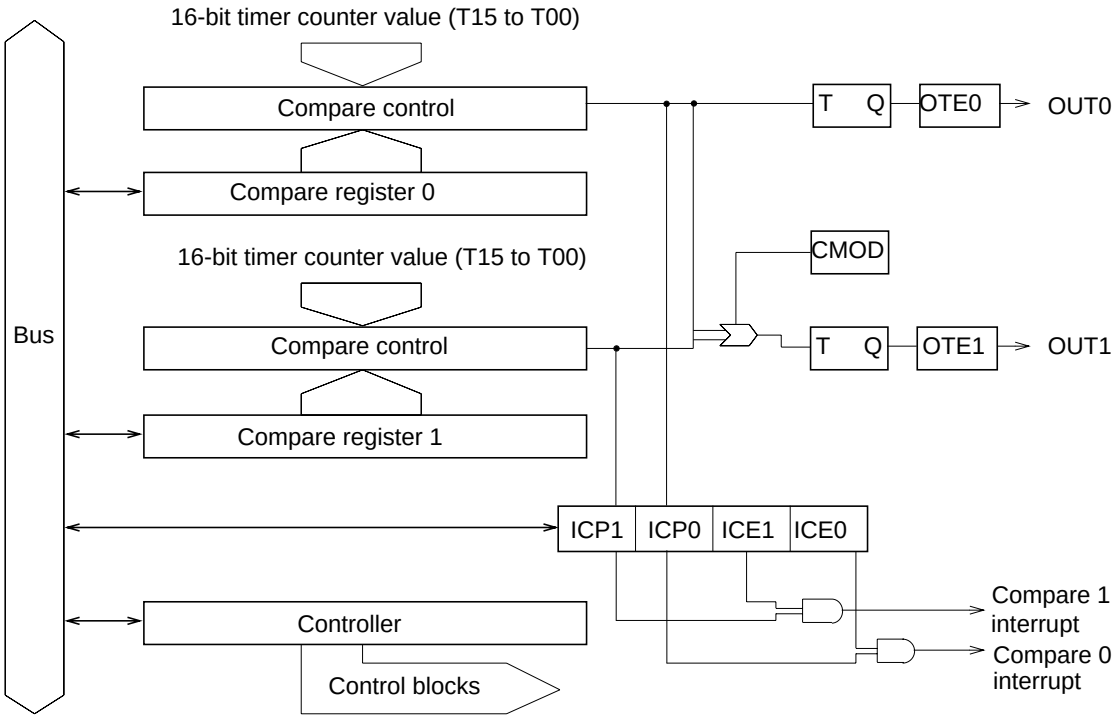
■ Output Compare

- Two compare registers exist that can be used independently. Depending on the setting, the two compare registers can be used to control pin outputs.
- The initial value for the pin output can be specified.
- An interrupt can be issued upon a match as a result of comparison.

■ Output Compare Block Diagram

Figure 11.4-1 shows a block diagram of output compare.

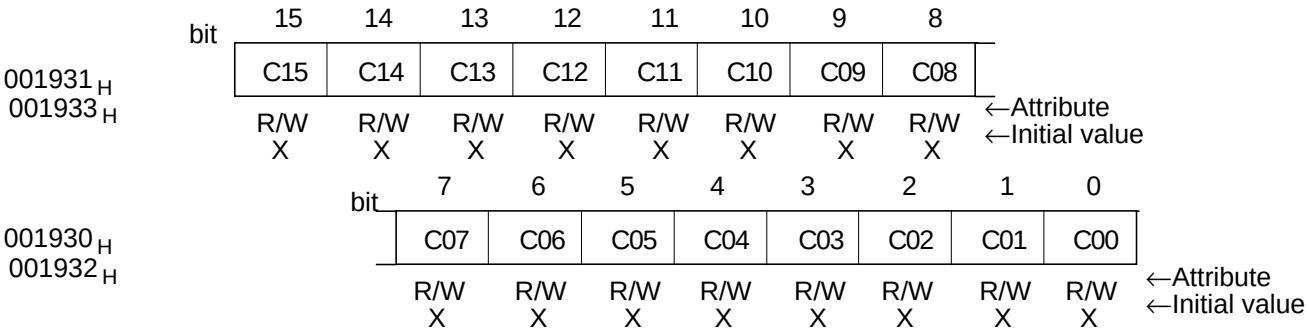
Figure 11.4-1 Output Compare Block Diagram



11.4.1 Output Compare Register

These 16-bit compare registers are compared with the 16-bit free-running timer. Since the initial register values are undefined, set appropriate value before enabling the operation. These registers must be accessed by the word access instructions. When the value of the register matches that of the 16-bit free-running timer, a compare signal is generated and the output compare interrupt flag is set. If output is enabled, the output level corresponding to the compare register is reversed.

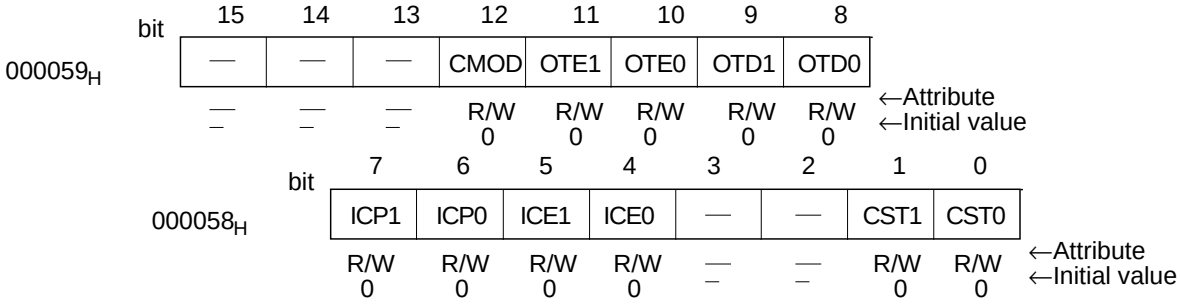
■ **Output Compare Register**



11.4.2 Control Status Register of Output Compare

The control status register sets the operation mode of output compare, starts and stops output compare, controls interrupts, and sets the external output pins.

■ **Control Status Register of Output Compare**



[bits 15, 14, and 13] Unused bits

[bit 12] CMOD

CMOD is used to switch the pin output level reverse mode upon a match while pin output is enabled (OTE1=1 or OTE0=1).

- When CMOD=0 (initial value), the output level of the pin corresponding to the compare register is reversed.
  - OUT0: The level is reversed upon a match with compare register 0.
  - OUT1: The level is reversed upon a match with compare register 1.
- When CMOD=1, the output level is reversed for the compare register 0 in the same manner as for CMOD=0. The output level of the pin corresponding to compare register 1 (OUT1), however, is reversed upon a match with compare register 0 or 1. If compare registers 0 and 1 have the same value, the same operation as with a single compare register is performed.
  - OUT0: The level is reversed upon a match with compare register 0.
  - OUT1: The level is reversed upon a match with compare register 0 or 1.

[bits 11 and 10] OTE1 and OTE0

These bits are used to enable the output compare output pins. The initial value for these bits is '0.'

0	General-purpose port (initial value)
1	Output compare pin output

Note:

OTE1: Corresponds to output compare 1 (OUT1).

OTE0: Corresponds to output compare 0 (OUT0).

When they are specified as outputs, the corresponding bits of the Port Direction registers should also be set to "1".

[bits 9 and 8] OTD1 and OTD0

These bits are used to change the pin output level when the output compare pin output is enabled. The initial value of the compare pin output is '0.' Ensure that the compare operation is stopped before a value is written. When read, these bits indicate the output compare pin output value.

0	Sets '0' for the compare pin output. (initial value)
1	Sets '1' for the compare pin output.

Note:

OTD1: Corresponds to output compare 1.

OTD0: Corresponds to output compare 0.

[bits 7 and 6] ICP1 and ICP0

These bits are used as output compare interrupt flags. '1' is set to these bits when the compare register value matches the 16-bit free-run timer value. While the interrupt request bits (ICE1 and ICE0) are enabled, an output compare interrupt occurs when the ICP1 and ICP0 bits are set. These bits are cleared by writing '0.'

Writing '1' has no effect. '1' is always read by a read-modify-write instruction.

0	No compare match (initial value)
1	Compare match

Note:

ICP1: Corresponds to output compare 1.

ICP0: Corresponds to output compare 0.

[bits 5 and 4] ICE1 and ICE0

These bits are used as output compare interrupt enable flags. While the '1' is written to these bits, an output compare interrupt occurs when an interrupt flag (ICP1 or ICP0) is set.

0	Output compare interrupt disabled (initial value)
1	Output compare interrupt enabled

Note:

ICE1: Corresponds to output compare 1.

ICE0: Corresponds to output compare 0.

[bits 3 and 2] Unused bits

[bits 1 and 0] CST1 and CST0

These bits are used to enable the comparison with 16-bit free-run timer.

0	Compare operation disabled (initial value)
1	Compare operation enabled

Ensure that a value is written to the compare register before the compare operation is enabled.

Note:

CST1: Corresponds to output compare 1.

CST0: Corresponds to output compare 0.

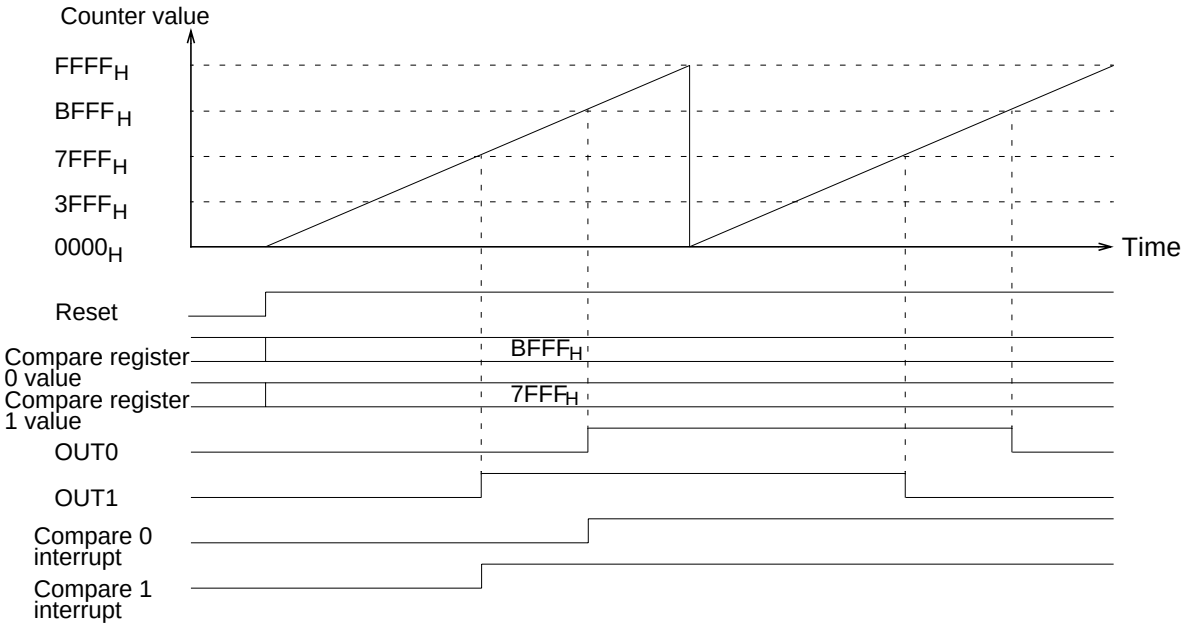
Since output compare is synchronized with the 16-bit free-running timer clock, stopping the 16-bit free-running timer stops compare operation.

11.4.3 16-bit Output Compare Operation

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

■ **Sample of Output Waveform when Compare Registers 0 and 1 are Used (The Initial Output Value is 0.)**

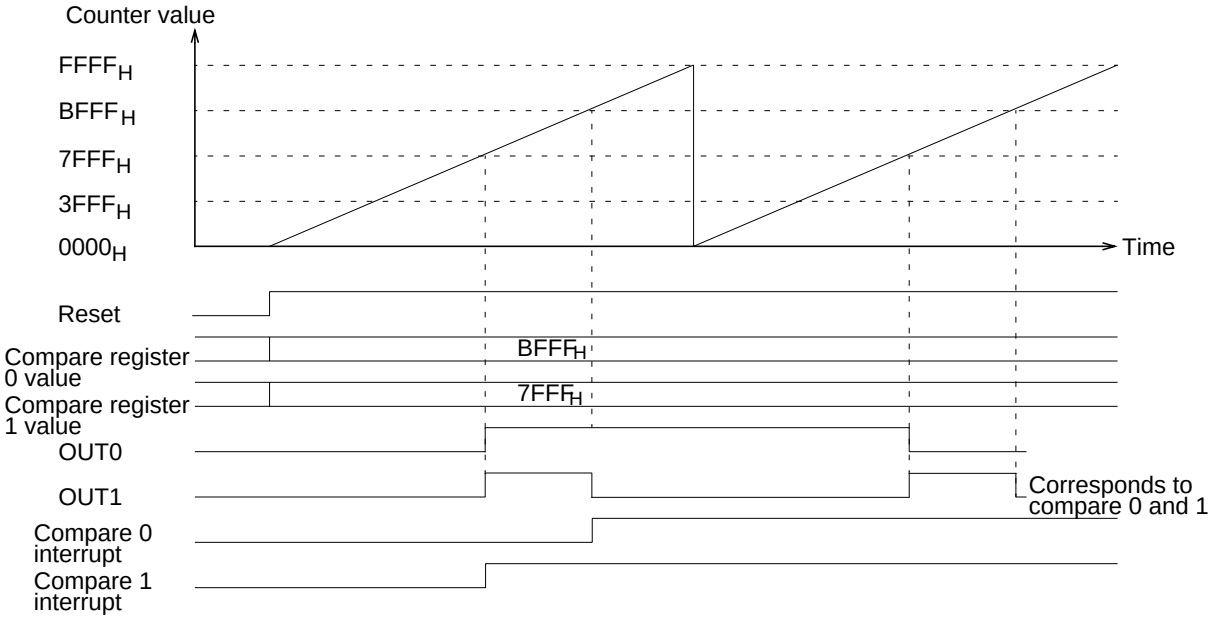
Figure 11.4-2 Sample of Output Waveform when Compare Registers 0 and 1 are Used



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

■ Sample of a Output Waveform with Two Compare Registers (The Initial Output Value is '0.')

Figure 11.4-3 Sample of a Output Waveform with Two Compare Registers (The Initial Output Value is '0')

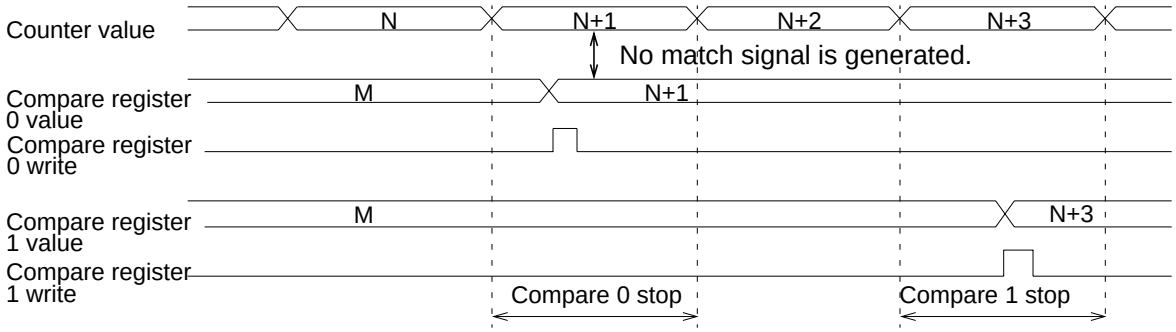


■ Output Compare Timing

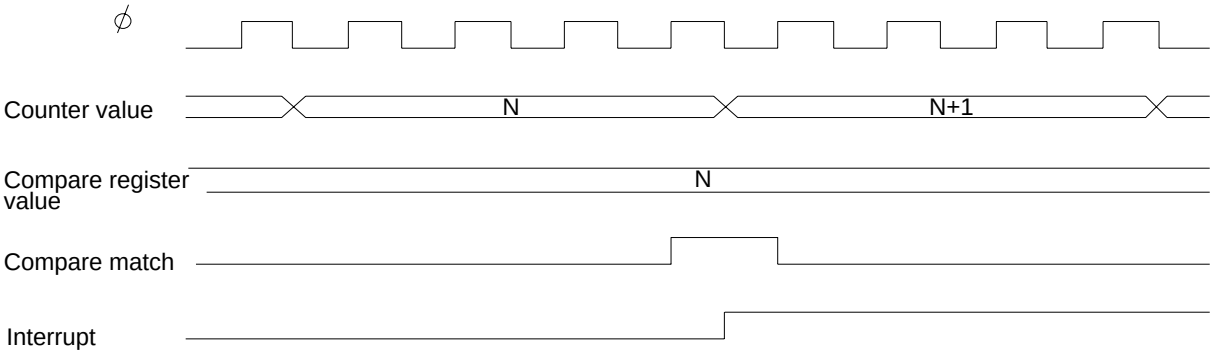
In output compare operation, a compare match signal is generated when the free-running timer value matches the specified compare register value. The output value can be reversed and an interrupt can be issued. The output reverse timing upon a compare match is synchronized with the counter count timing.

❍ Compare operation upon update of compare register

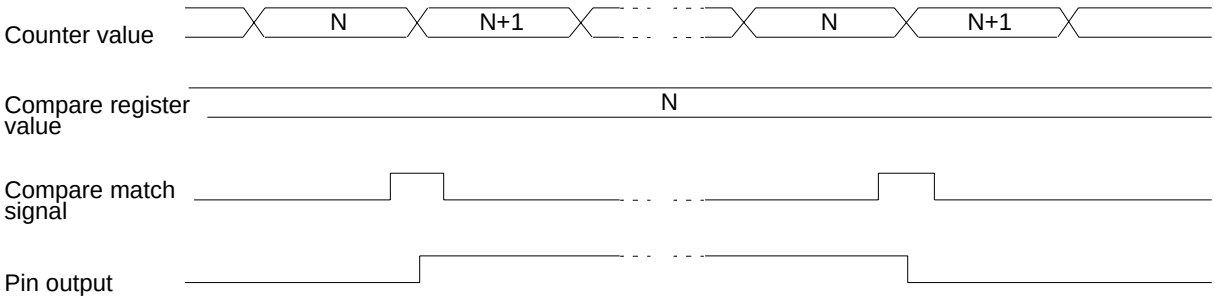
When the compare register is updated, comparison with the counter value is not performed.



❍ **Interrupt timing**



❍ **Output pin change timing**



11.5 Input Capture

Input capture detects a rising or falling edge or both edges of an external input signal and stores a 16-bit free-running timer value at that time in a register. In addition, input capture can generate an interrupt upon detection of an edge. Input capture consists of an input capture data register and a control register.

■ Input Capture

Each input capture has a corresponding external input pin.

❍ **The valid edge of an external input can be selected from the following three types:**

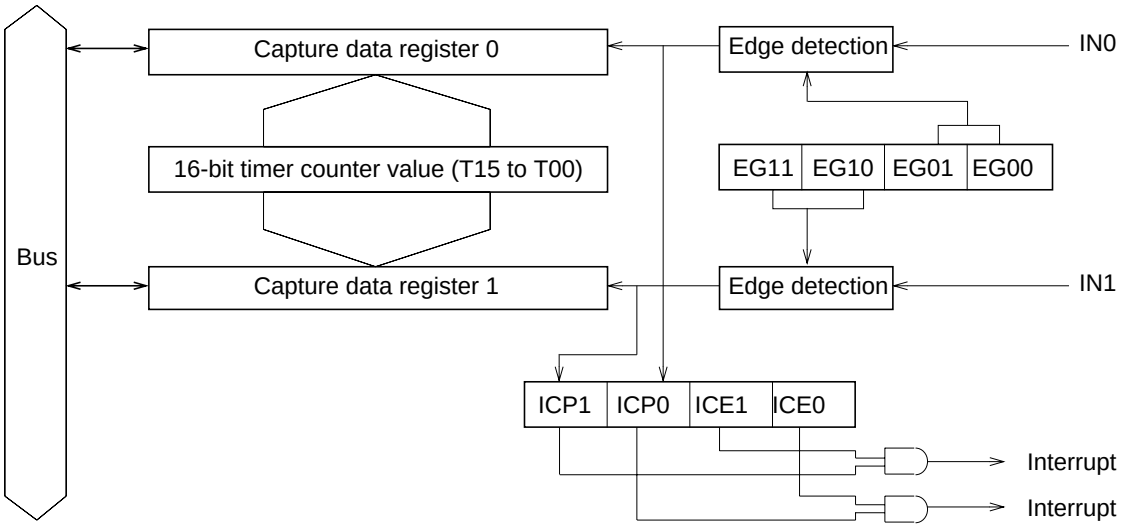
Rising edge	↑
Falling edge	↓
Both edges	↑ ↓

❍ **An interrupt can be generated upon detection of a valid edge of an external input.**

■ Input Capture Block Diagram

Figure 11.5-1 shows a block diagram of input capture.

Figure 11.5-1 Input Capture Block Diagram

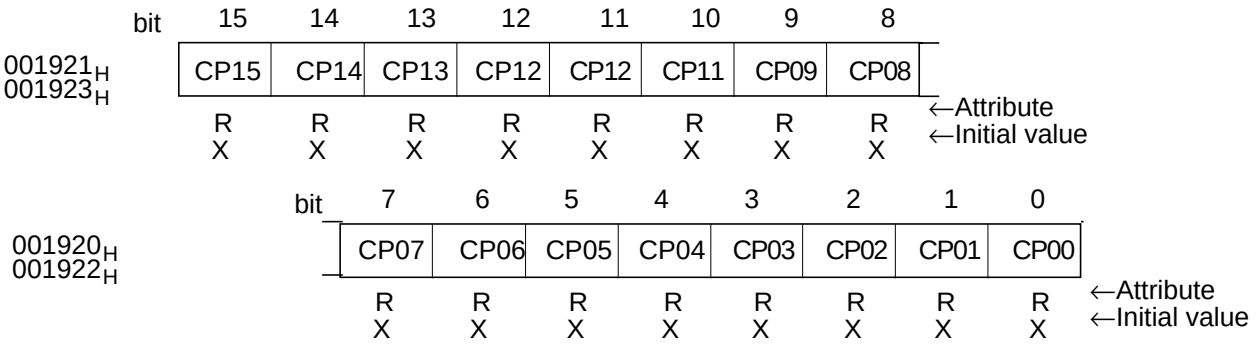


11.5.1 Input Capture Register Details

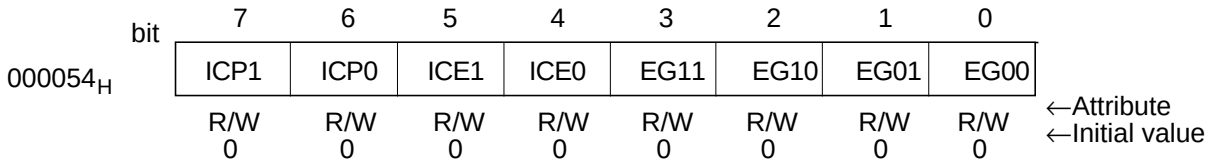
Input capture has the two registers listed. These registers store a value from the 16-bit free running timer when a valid edge of the corresponding external pin input waveform is detected. (The registers must be accessed in word mode. No values can be written to the registers.)

- **Input capture data register**
- **Input capture control register**

■ **Input Capture Data Register**



■ **Control Status Register**



[bits 7 and 6] ICP1 and ICP0

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

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

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

Note:

ICP0: Corresponds to input capture 0.

ICP1: Corresponds to input capture 1.

[bits 5 and 4] ICE1 and ICE0

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

0	Interrupt disabled (initial value)
1	Interrupt enabled

Note:

ICE0: Corresponds to input capture 0.

ICE1: Corresponds to input capture 1.

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

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

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

Note:

EG01 and EG00: Correspond to input capture 0.

EG11 and EG10: Correspond to input capture 1.

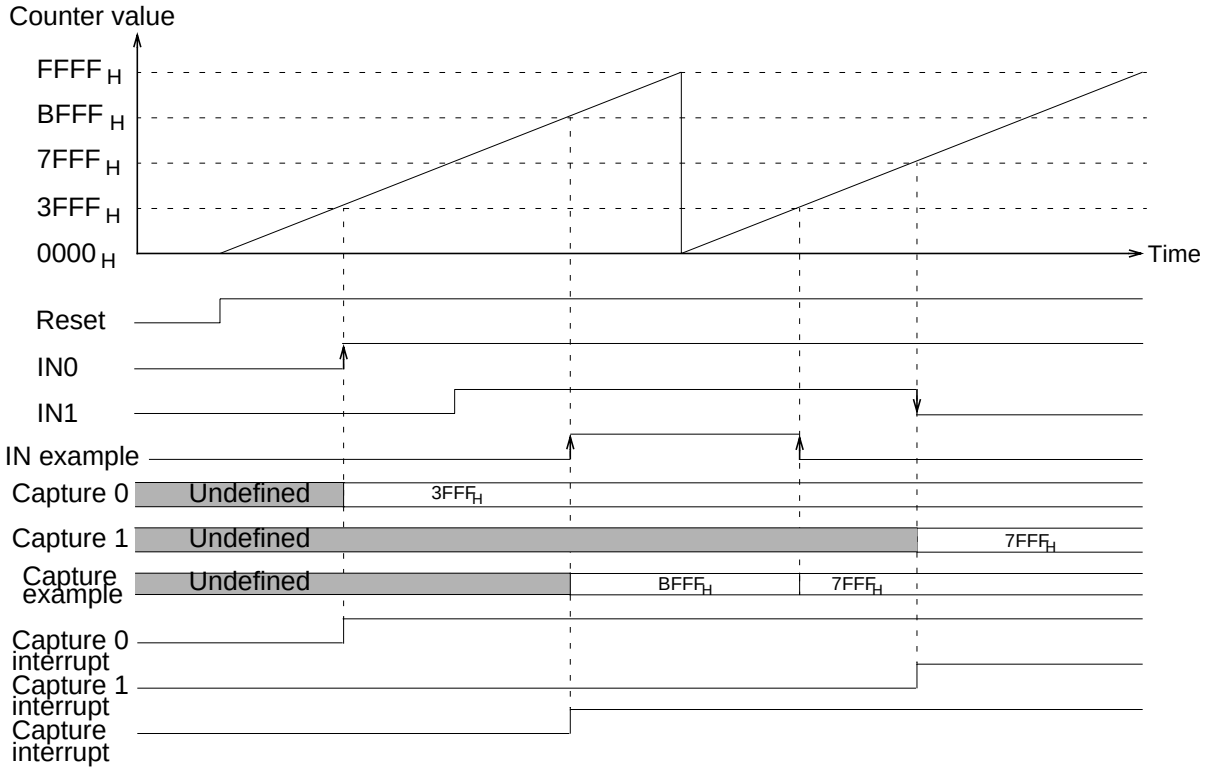
11.5.2 16-bit Input Capture Operation

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

■ **Sample of Input Capture Fetch Timing**

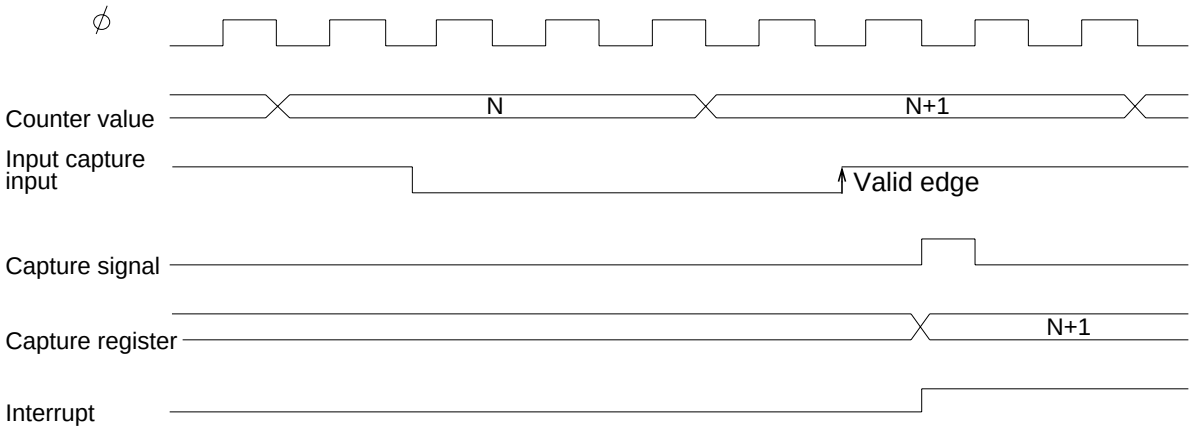
- Capture 0: Rising edge
- Capture 1: Falling edge
- Capture example: Both edges

Figure 11.5-2 Sample of Input Capture Fetch Timing



■ Input Capture Input Timing

❍ Capture timing for input signals



CHAPTER 12 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)

This chapter explains the functions and operations of the 16-bit reload timer (with the event count function).

12.1 Outline of 16-Bit Reload Timer (with Event Count Function)

12.2 16-Bit Reload Timer (with Event Count Function)

12.3 Internal Clock and External Clock Operations of 16-Bit Reload Timer

12.4 Underflow Operation of 16-Bit Reload Timer

12.5 Output Pin Functions of 16-Bit Reload Timer

12.6 Counter Operation State

12.1 Outline of 16-Bit Reload Timer (with Event Count Function)

The 16-bit reload timer consists of a 16-bit down-counter, a 16-bit reload register, one input pin (TIN) and one output pin (TOUT), and a control register. The input clock can be selected from one external clock and three types of internal clock.

■ Outline of 16-bit Reload Timer (with Event Count Function)

The output pin (TOUT) outputs a toggle output waveform in reload mode and outputs a square waveform indicating counting in one-shot mode. The input pin (TIN) is used for event input in event count mode, and can be used for trigger input or gate input in internal clock mode.

The MB90590 Series has two 16-bit reload timers. However the TIN input and TOUT output external pins are shared between the two timers.

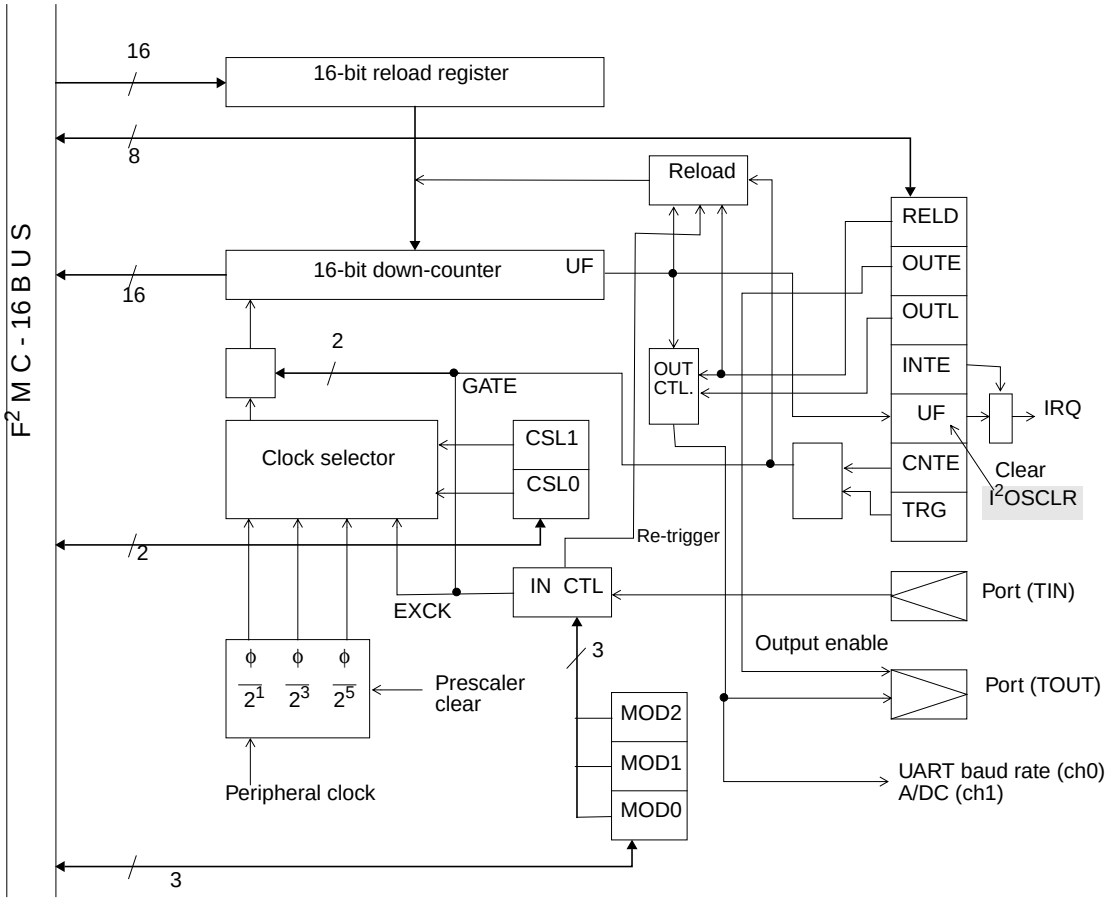
■ Intelligent I/O Service (EI^{2}OS) Function and Interrupts

The timer includes a circuit that supports EI^{2}OS. The timer can activate EI^{2}OS when an underflow occurs. EI^{2}OS can be used with both timers on this product. However, as both timers (ch0 and ch1) are connected to the same interrupt control register (ICRx) in the interrupt controller, ch0 and ch1 cannot be assigned to different EI^{2}OS services. Also, as the two timers have different interrupt vectors, they can be assigned to two different interrupt services. However, as ch0 and ch1 share an interrupt control register as described above, the same interrupt level applies to both channels.

■ Block Diagram of 16-bit Reload Timer

Figure 12.1-1 shows a block diagram of the 16-bit reload timer.

Figure 12.1-1 Block Diagram of 16-bit Reload Timer

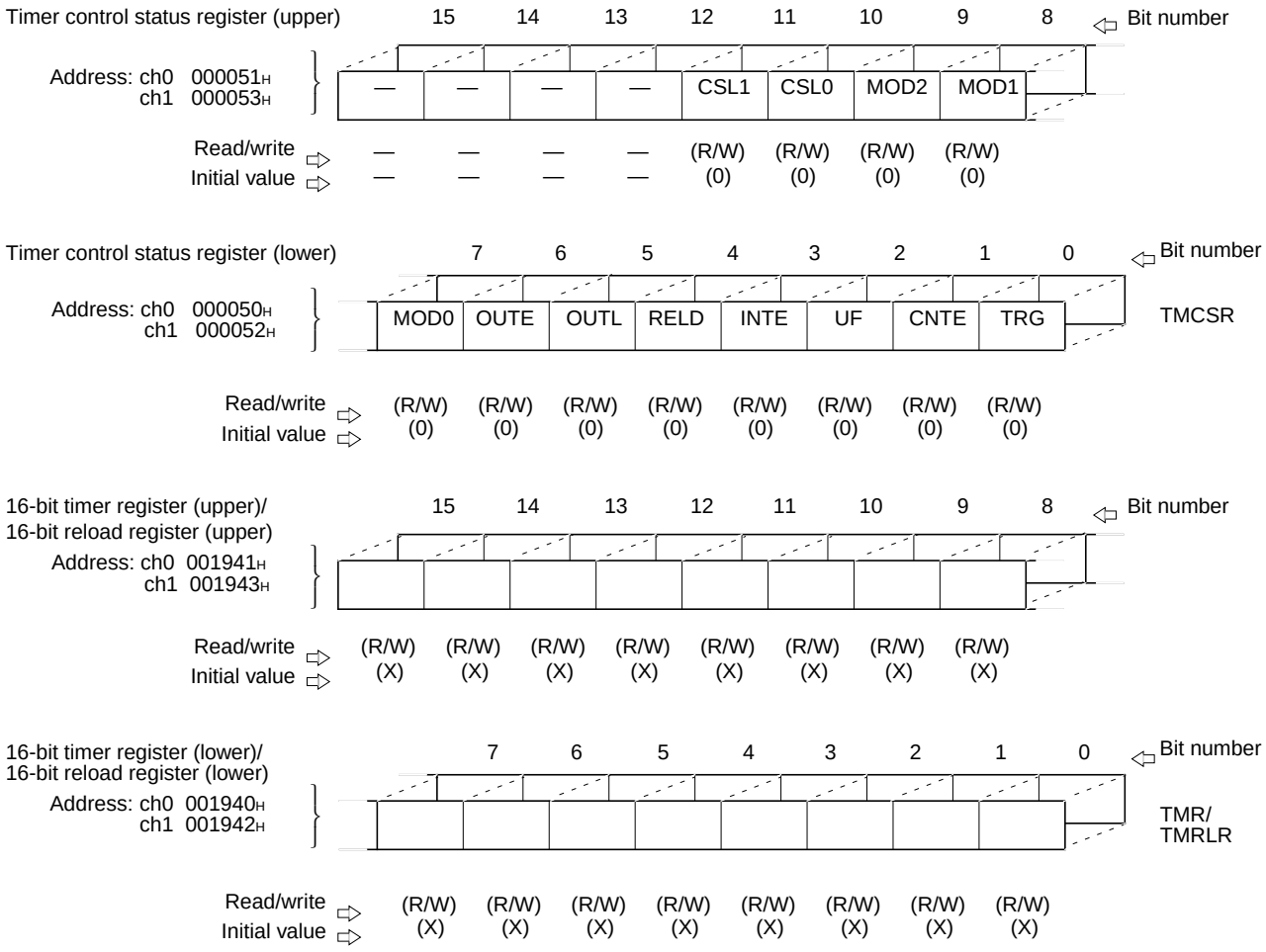


12.2 16-Bit Reload Timer (with Event Count Function)

The 16-bit reload timer has the following two types of registers:

- **Timer control register (TMCSR)**
- **16-bit timer register (TMR)/16-bit reload register (TMRLR)**

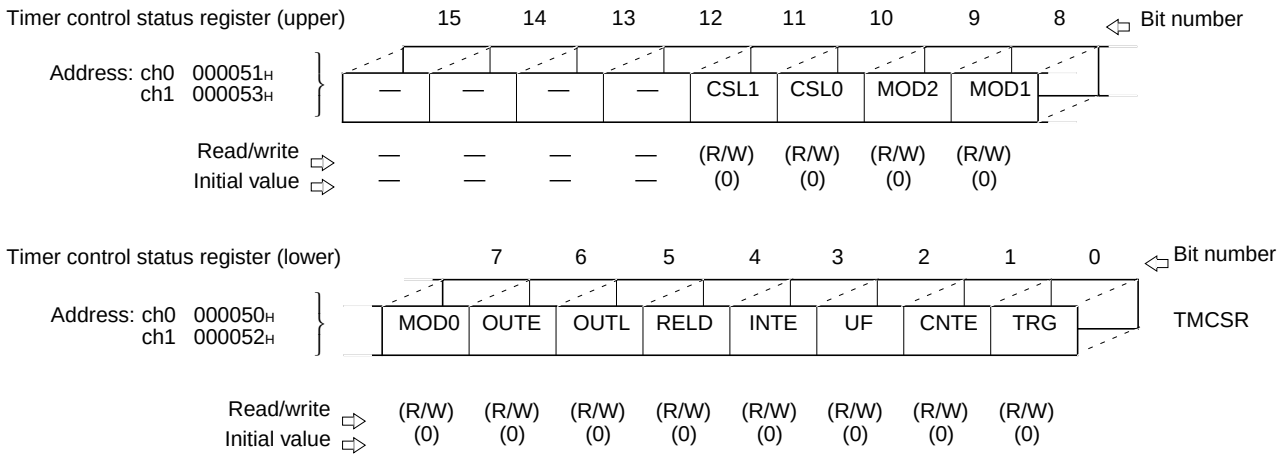
■ 16-bit Reload Timer Register



12.2.1 Timer Control Status Register (TMCSR)

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

■ Register Layout of Timer Control Register (TMCSR)



■ Register Contents of Timer Control Register (TMCSR)

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

The count clock select bits. Table 12.2-1 lists the selected clock sources.

Table 12.2-1 Clock Sources for CSL Bit Settings

CSL1	CSL0	Clock Source (Machine cycle ϕ = 16 MHz)
0	0	$\phi/2^1$ (0.125 µs)
0	1	$\phi/2^3$ (0.5 µs)
1	0	$\phi/2^5$ (2.0 µs)
1	1	External event count mode

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

These bits set the operation mode and I/O pin functions.

The MOD2 bit selects the I/O functions. When MOD2 = "0", the input pin functions as a trigger input. In this case, the reload register contents is loaded to the counter when an active edge is input to the input pin and count operation proceeds. When MOD2 = "1", the timer operates in gate counter mode and the input pin functions as a gate input. In this mode, the counter only counts while an active level is input to the input pin.

The MOD1 and 0 bits set the pin functions for each mode. Tables 12.2-2 and 12.2-3 list the

MOD2, 1, 0 bit settings.

Table 12.2-2 MOD2, 1, 0 Bit Settings (1)

MOD2	MOD1	MOD0	Input Pin Function	Active Edge or Level
0	0	0	Trigger disabled	-
0	0	1	Trigger input	Rising edge
0	1	0	↑	Falling edge
0	1	1	↑	Both edges
1	×	0	Gate input	"L" level
1	×	1	↑	"H" level

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

Table 12.2-3 MOD2, 1, 0 Bit Settings (2)

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

- Event counter mode (CSL0,1 = "11")
- Bits marked as × in the table can be set to any value.

[Bit 6] OUTE

Output enable bit. The TOUT pin functions as a general-purpose port when this bit is "0" and as the timer output pin when this bit is "1". In reload mode, the output waveform toggles. In one-shot mode, TOUT outputs a square waveform that indicates that counting is in progress.

[Bit 5] OUTL

This bit sets the output level for the TOUT pin.

Table 12.2-4 OUTE, RELD, and OUTL Settings

OUTE	RELD	OUTL	Output Waveform
0	×	×	General-purpose port

Table 12.2-4 OUTE, RELD, and OUTL Settings (Continued)

OUTE	RELD	OUTL	Output Waveform
1	0	0	Output an "H" level square waveform during counting.
1	0	1	Output an "L" level square waveform during counting.
1	1	0	Toggle output. Starts with "L" level output.
1	1	1	Toggle output. Starts with "H" level output.

[Bit 4] RELD (Reload)

This bit enables reload operations. When RELD is "1", the timer operates in reload mode. In this mode, the timer loads the reload register contents into the counter and continues counting whenever an underflow occurs (when the counter value changes from 0000_H to $FFFF_H$). When RELD is "0", the timer operates in one-shot mode. In this mode, the count operation stops when an underflow occurs due to the counter value changing from 0000_H to $FFFF_H$.

[Bit 3] INTE (Interrupt enable)

Timer interrupt request enable bit. When INTE is "1", an interrupt request is generated when the UF bit changes to "1". When INTE is "0", no interrupt request is generated, even when the UF bit changes to "1".

[Bit 2] UF (Underflow)

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

[Bit 1] CNTE (Count enable)

Timer count enable bit. Writing "1" to CNTE sets the timer to wait for a trigger. Writing "0" stops count operation.

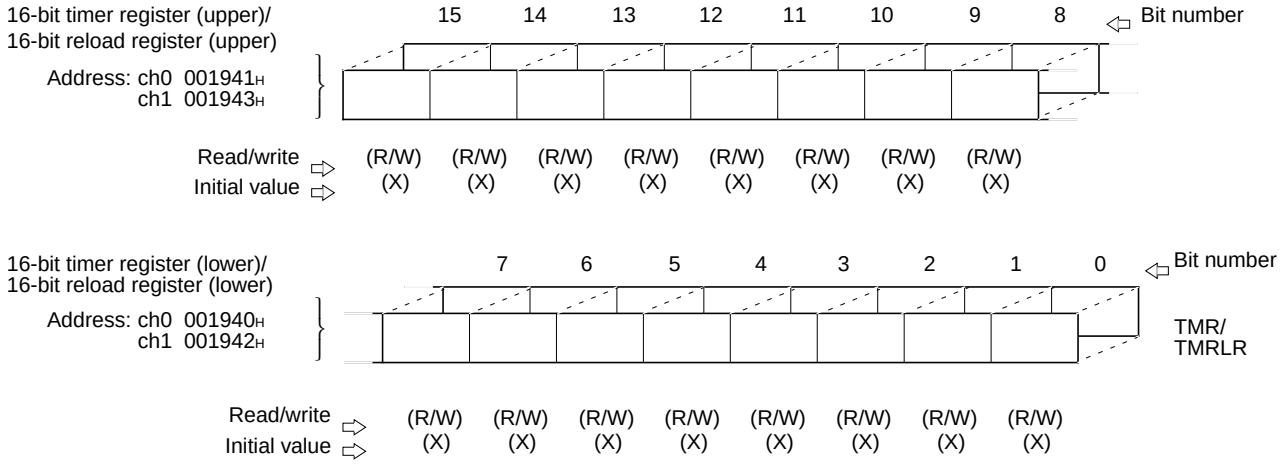
[Bit 0] TRG (Trigger)

Software trigger bit. Writing "1" to TRG applies a software trigger, causing the timer to load the reload register contents to the counter and start counting. Writing "0" has no meaning. Reading always returns "0". Applying a trigger using this register is only valid when CNTE = "1". Writing "1" has no effect if CNTE = "0".

12.2.2 Register Layout of 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR)

TMR contents (for reading)
Reading this register reads the count value of the 16-bit timer. The initial value is undefined. Always read this register using the word access instructions.
TMRLR contents (for writing)
The 16-bit reload register holds the initial count value. The initial value is undefined. Always write to this register using the word access instructions.

■ **Register Layout of 16-bit Timer Register (TMR)/16-bit Reload Register (TMRLR)**



12.3 Internal Clock and External Clock Operations of 16-bit Reload Timer

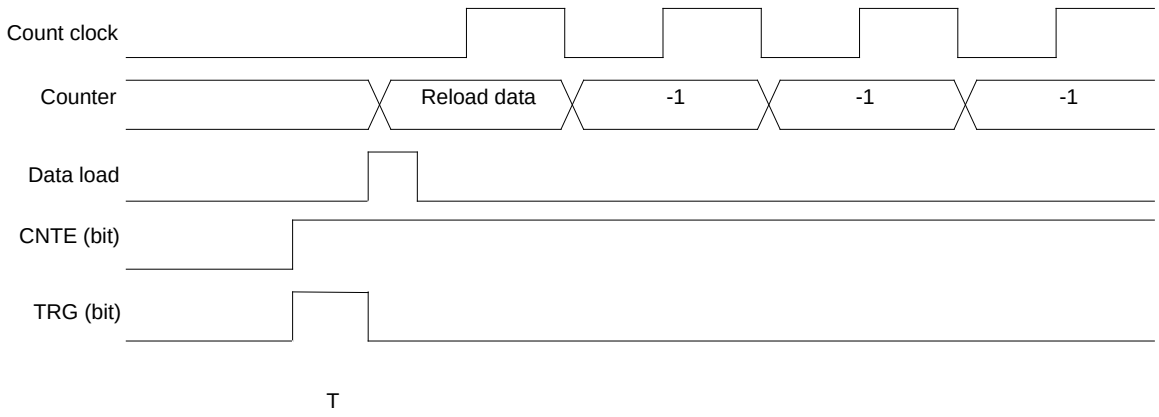
The machine clock divided by 2^1, 2^3, or 2^5 can be selected as the clock sources for operating the timer from an internal divide clock. The external input pin can be selected as either a trigger input or gate input by a register setting.
If an external clock is selected, the TIN pin functions as an external event input pin to count the number of valid edges set in the register.

■ Internal Clock Operation of 16-bit Reload Timer

Writing "1" to both the CNTE and TRG bits in the control register enables and starts counting at one time. Using the TRG bit as a trigger input is always available when the timer is enabled (CNTE = "1"), regardless of the operation mode.

Figure 12.3-1 shows counter activation and counter operation. A time period T (T: machine cycle) is required from the counter start trigger being input until the reload register data is loaded into counter.

Figure 12.3-1 Activation and Operation of 16-bit Reload Timer Counter

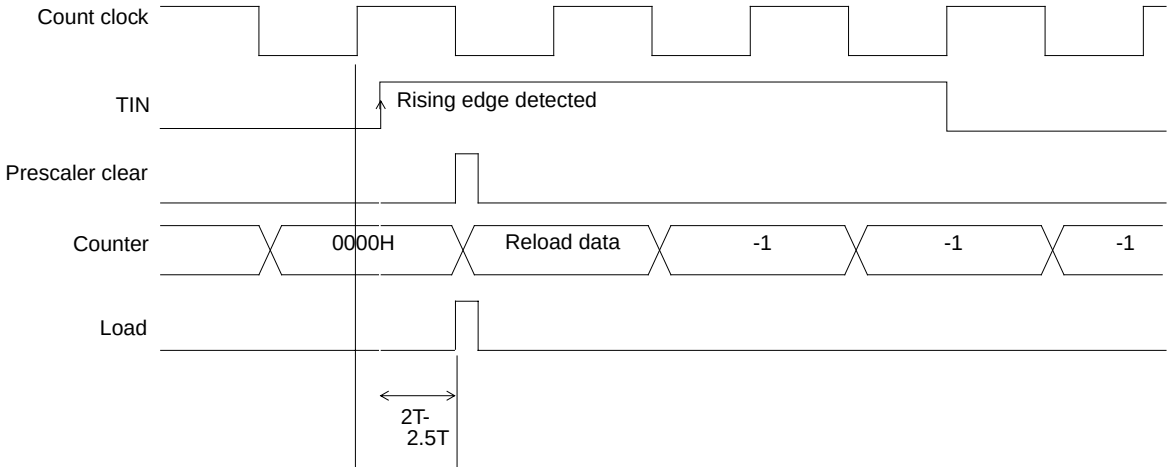


■ Input Pin Functions of 16-bit Reload Timer (in Internal Clock Mode)

The TIN pin can be used as either a trigger input or a gate input when an internal clock is selected as the clock source. When used as a trigger input, input of an active edge causes the timer to load the reload register contents to the counter and then start count operation after clearing the internal prescaler. Input a pulse width of at least 2T (T is the machine cycle) to TIN.

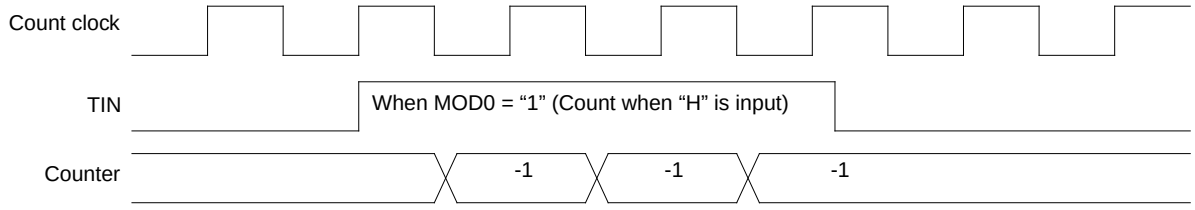
Figure 12.3-2 shows the operation of trigger input.

Figure 12.3-2 Trigger Input Operation of 16-bit Reload Timer



When used as a gate input, the counter only counts while the active level specified by the MOD0 bit of the control register is input to the TIN pin. In this case, the count clock continues to operate unless stopped. The software trigger can be used in gate mode, regardless of the gate level. Input a pulse width of at least 2T (T is the machine cycle) to the TIN pin. Figure 12.3-3 shows the operation of gate input.

Figure 12.3-3 Gate Input Operation of 16-bit Reload Timer



■ External Event Counter

The TIN pin functions as an external event input pin when an external clock is selected. The counter counts on the active edge specified in the register. Input a pulse width of at least 4T (T is the machine cycle) to the TIN pin.

12.4 Underflow Operation of 16-bit Reload Timer

An underflow is defined for this timer as the time when the counter value changes from 0000_H to $FFFF_H$. Therefore, an underflow occurs after (reload register setting + 1) counts.

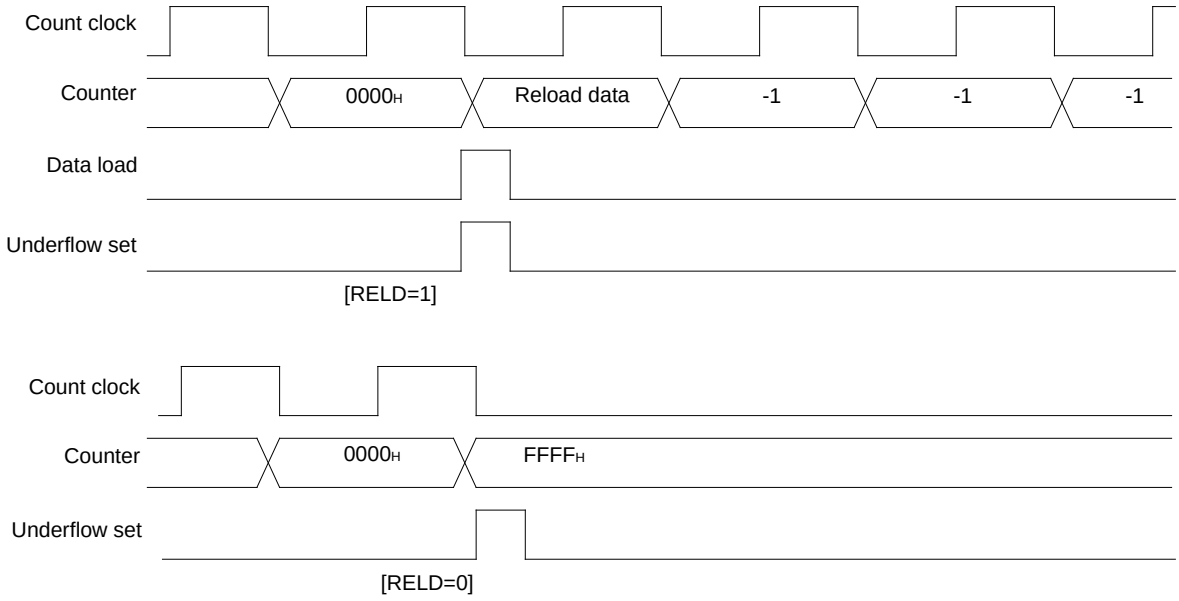
■ Underflow Operation of 16-bit Reload Timer

If the RELD bit in the control register is "1" when the underflow occurs, the contents of the reload register is loaded into the counter and counting continues. When RELD is "0", counting stops with the counter at $FFFF_H$.

The UF bit in the control register is set when the underflow occurs. If the INTE bit is "1" at this time, an interrupt request is generated.

Figure 12.4-1 shows the operation when an underflow occurs.

Figure 12.4-1 Underflow Operation of 16-bit Reload Timer



12.5 Output Pin Functions of 16-bit Reload Timer

In reload mode, the TOUT pin performs toggle output (inverts at each underflow). In one-shot mode, the TOUT pin functions as a pulse output that outputs a particular level while the count is in progress.

■ Output Pin Functions of 16-bit Reload Timer

The OUTL bit of the control register sets the output polarity. When OUTL = "0", the initial value for toggle output is "0" and the one-shot pulse output is "1" while the count is in progress. The output waveforms are opposite when OUTL = "1".

Figures 12.5-1 and 12.5-2 show the output pin functions.

Figure 12.5-1 Output Pin Function of 16-bit Reload Timer (1)

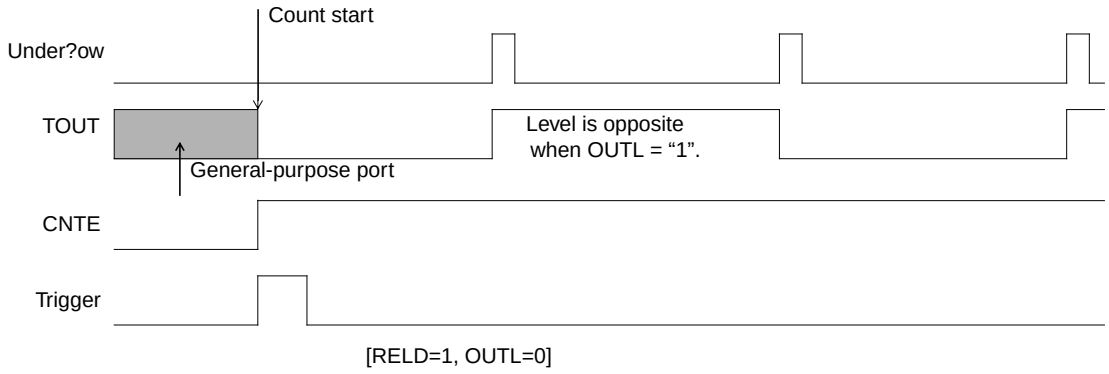
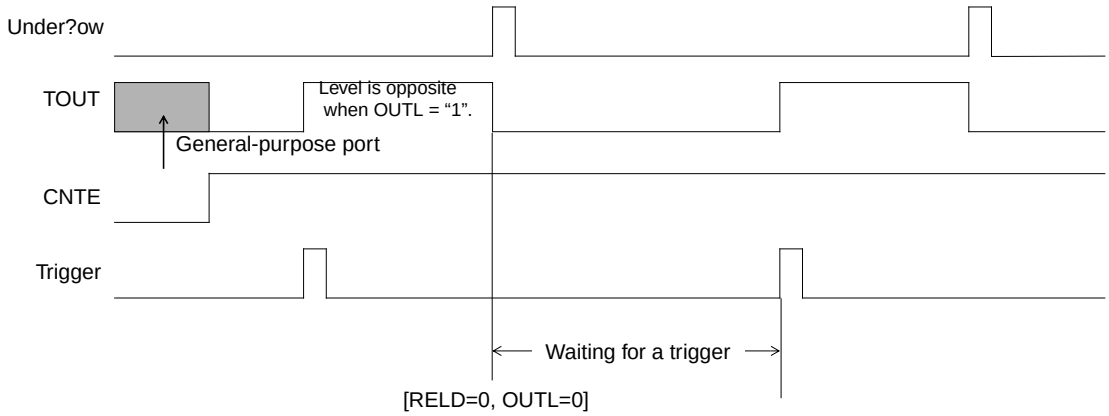


Figure 12.5-2 Output Pin Function of 16-bit Reload Timer (2)



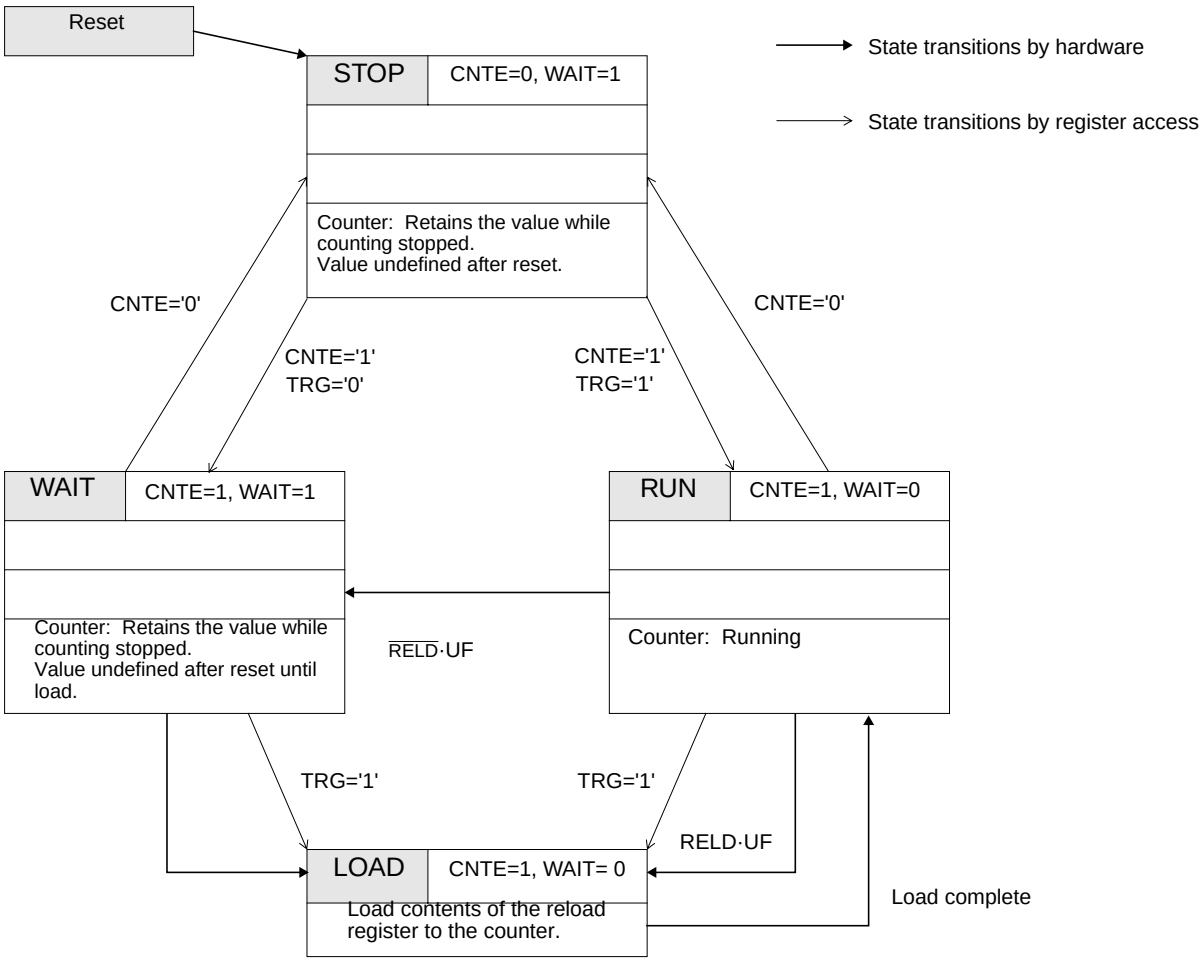
12.6 Counter Operation State

The counter state is determined by the CNTE bit in the control register and the internal WAIT signal. Available states are: CNTE = "0" and WAIT = "1" (STOP state), CNTE = "1" and WAIT = "1" (WAIT state for trigger), and CNTE = "1" and WAIT = "0" (RUN state).

■ Counter Operation State

Figure 12.6-1 shows the transitions between each state.

Figure 12.6-1 Counter State Transitions



CHAPTER 13 WATCH-DOG TIMER

This chapter explains the functions and operations of the watch-dog timer.

13.1 Outline of Watch Timer

13.2 Watch-dog Timer Registers

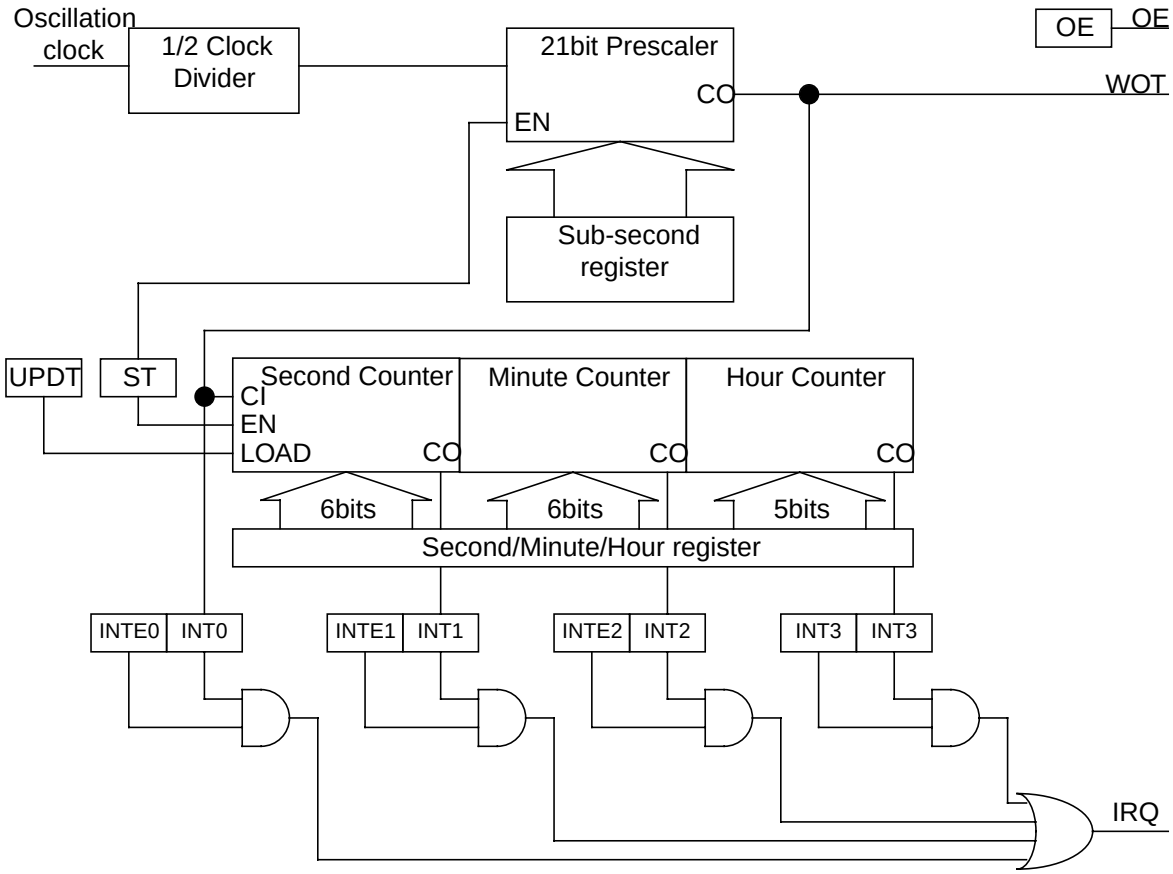
13.1 Outline of Watch Timer

The Watch Timer consists of the Timer Control register, Sub-second register, Second/ Minute/Hour registers, 1/2 clock divider, 21bit prescaler and Second/Minute/Hour counters. The oscillation frequency of the MCU is assumed to be at 4MHz for the aimed operation of the Watch Timer. The Watch Timer operates as the real-world timer and provides the real-world time information.

■ Block Diagram of Watch-dog Timer

Figure 13.1-1 shows a block diagram of the watch-dog timer.

Figure 13.1-1 Block Diagram of Watch-dog Timer

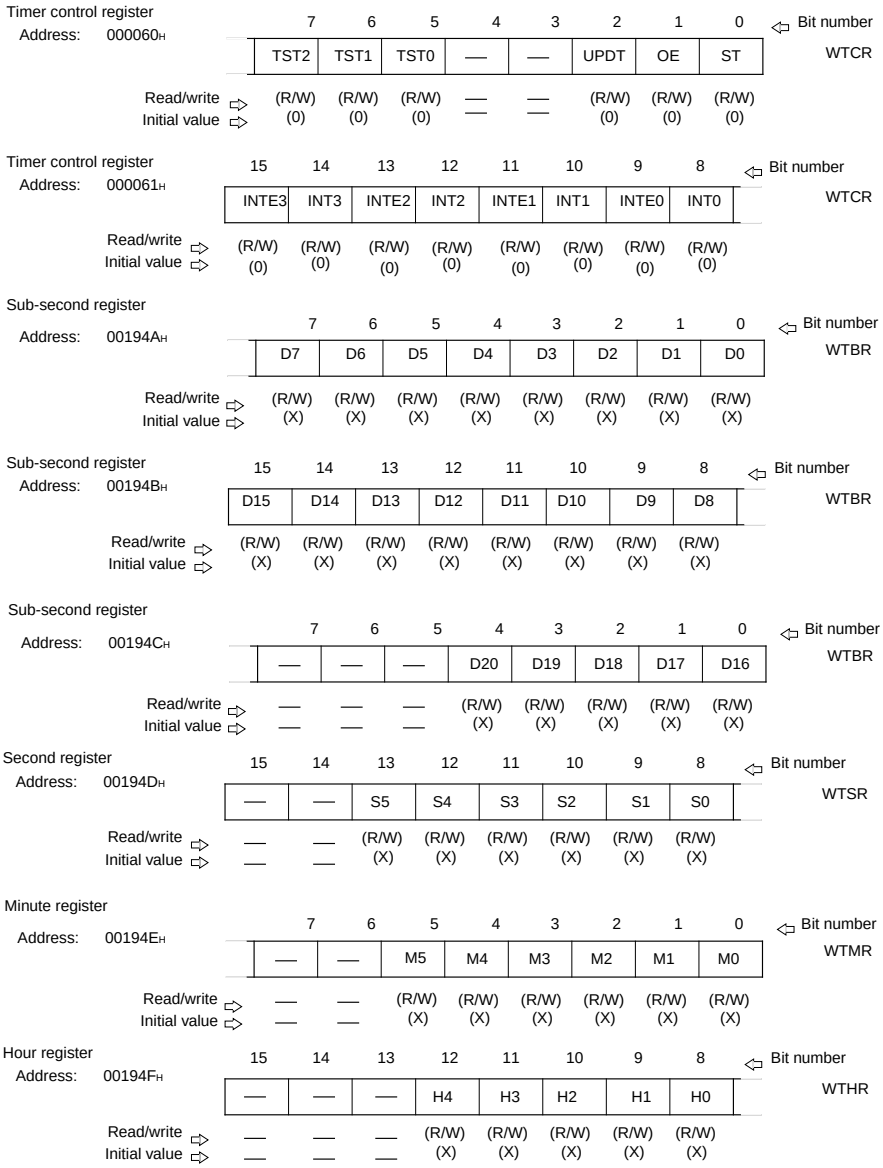


13.2 Watch-dog Timer Registers

The watch-dog timer has the following five types of registers:

- **Timer control register (WTCR)**
- **Subsecond register (WTBR)**
- **Second register (WTSR)**
- **Minute register (WTMR)**
- **Hour register (WTHR)**

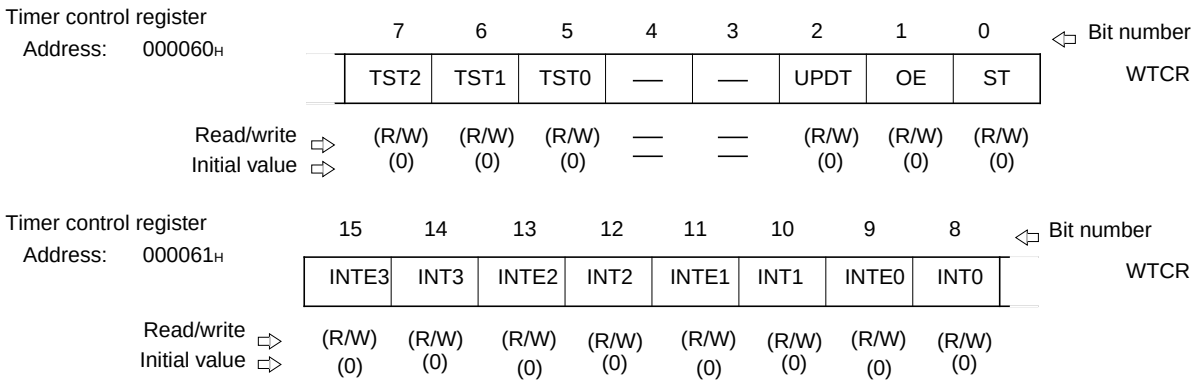
■ Watch-dog Timer Registers



13.2.1 Timer Control Register

The timer control register starts and stops the watch-dog timer, controls interrupts, and sets the external output pins.

■ **Timer Control Register**



[bits 15 to 8] INT3 to 0, INTE3 to 0 : Interrupt flags and Interrupt enable flags

INT0 to INT3 are the interrupt flags. They are set when the second counter, minute counter and hour counter overflow respectively. If a INT bit is set while the corresponding INTE bit is "1", the Watch Timer signals an interrupt. These flags are intended to signal an interrupt every second/minute/hour/day.

Writing "0" to the INT bits clears the flags and writing "1" does not have any effect. Any read-modify-write instruction performed on the INT bit results reading "1".

[bits 7 to 5] TST2 to 0 : Test bits

These bits are prepared for the device test. In any user applications, they should be set to "000".

[bit 2] UPDT : Update bit

The UPDT bit is prepared for modifying the Second/Minute/Hour counter values.

To modify the counter values, write the modified data in the Second/Minute/Hour registers. Then set the UPDT bit to "1". The register values are loaded to the counter at the next CO signal from the 21-bit prescaler. The UPDT bit is reset by the hardware when the counter values are updated. However, if the set operation by software and the reset operation by hardware occur at the same time, the UPDT bit will not be reset.

Note:

If this bit is set during "59 second", normal up count operation is executed and this bit is reset to "0" without reflecting the Second/Minute/Hour register values.

Writing "0" to the UPDT bit has no effect and a read-modify-write instruction results in reading "0".

[bit 1] OE : Output enable bit

When the OE bit is set to "1", the WOT external pin serves as the output for the Watch

Timer. Otherwise it can be used as a general purpose I/O or for another peripheral block.

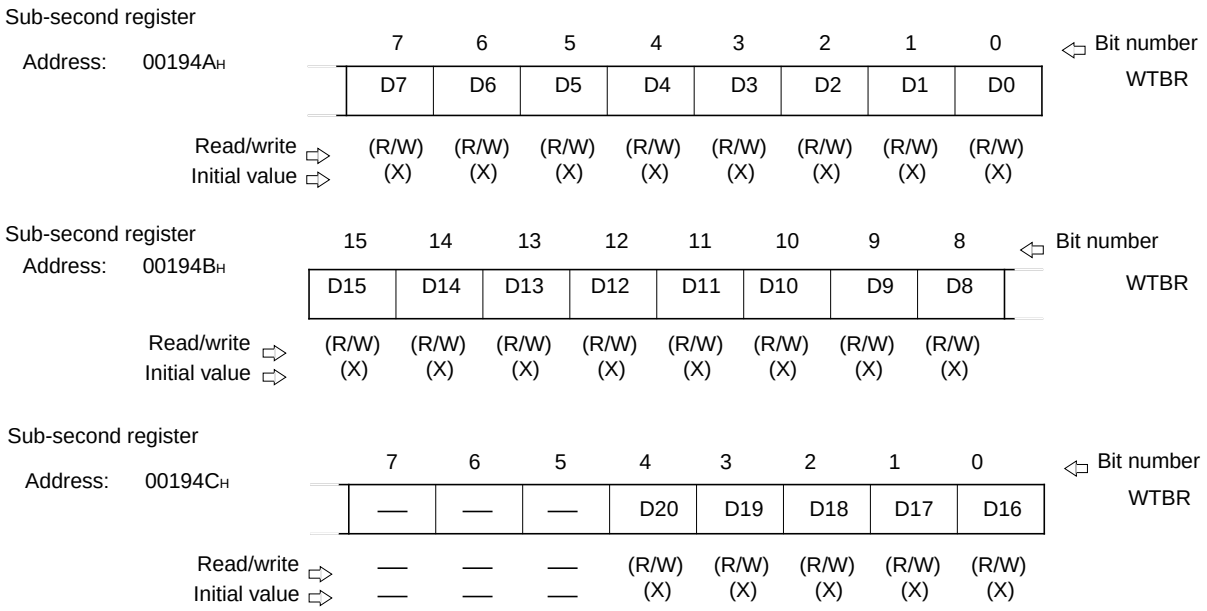
[bit 0] ST : Start bit

When the ST bit is set to "1", the Watch Timer loads Second/Minute/Hour values from the registers and starts its operation. When it is reset to "0", all the counters and the prescalers are reset to "0" and halts.

13.2.2 Sub-second Registers

The subsecond register stores a reload value for the 21-bit prescaler that divides the oscillation clock. The reload value is usually set so that the 21-bit prescaler will output exactly within a one-second cycle.

■ Sub-second Register



[bit 20 to 0] D20 to D0

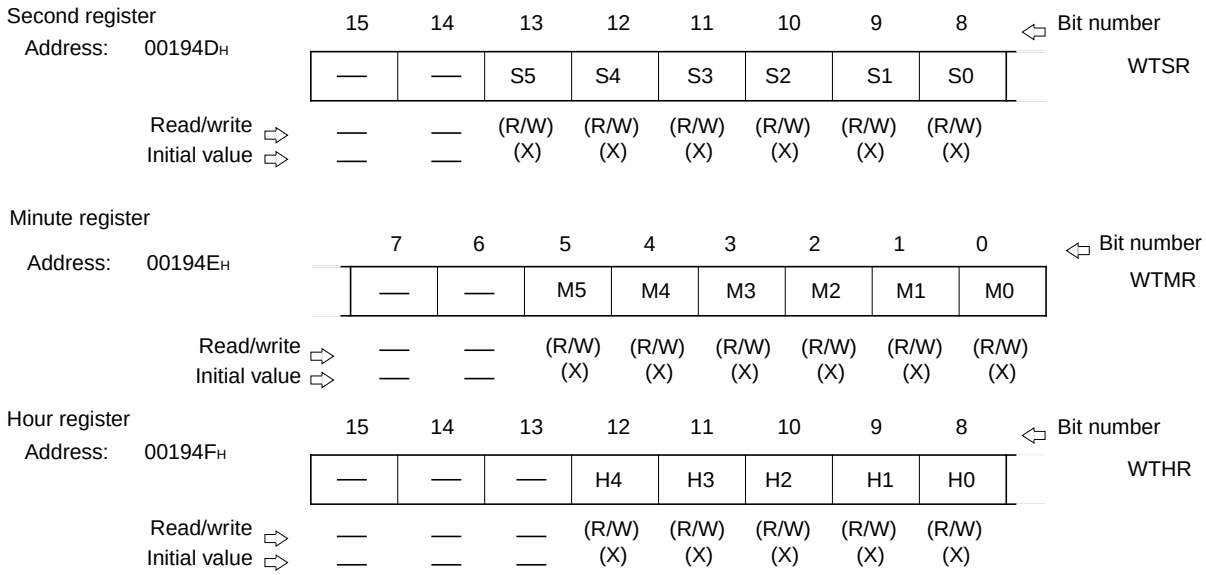
The Sub-second register stores the reload value for the 21bit prescaler. This value is reloaded after the reload counter reaches "0". Note that when modifying the all three bytes, make sure the reload operation will not be performed in between the write instructions. Otherwise the 21-bit prescaler loads the incorrect value of the combination of new data and old data bytes. It is generally recommended that the Sub-Second register are updated while the ST bit is "0". If the sub-second registers are set to "0", the 21-bit prescaler does not operate at all.

The input clock frequency always equals the oscillation clock frequency and it is intended to be 4MHz. The reload value of the 21bit prescaler is typically set to Hex1E847F which equals to "$2^7 * 5^6$-1". Therefore the combination of these two prescalers is intended to provide a clock signal of exact one second.

13.2.3 Second/Minute/Hour Registers

The Second/Minute/Hour registers stores the time information. It is a binary representation of the second, minute and hour.
Reading these registers simply returns the counter values. These registers are write associable however, the written data is loaded in the counters after the UPDT bit is set to "1".

■ Second/Minute/Hour Registers



Since there are three byte-registers, make sure the obtained values from the registers are consistent.

i.e. Obtained value of "1 hour, 59 minute, 59 second" could be "0 hour 59 minute, 59 second" or "1 hour, 0 minute, 0 second" or "2 hour, 0 minute, 0 second".

Also when the operation clock of the MCU is the half of the oscillation clock (When the PLL is stopped), the read values from these registers may be corrupt. This is due to the synchronization of the read operation and the count operation. Therefore it is recommended is use a second interrupt to trigger the read instructions.

CHAPTER 14 8/16-BIT PPG

This chapter explains the 8/16-bit PPG and explains its functions.

14.1 Outline of 8/16-Bit PPG

14.2 Block Diagram of 8/16-Bit PPG

14.3 8/16-Bit PPG Registers

14.4 Operations of 8/16-Bit PPG

14.5 Selecting a Count Clock for 8/16-Bit PPG

14.6 Controlling Pin Output of 8/16-Bit PPG Pulses

14.7 8/16-Bit PPG Interrupts

14.8 Initial Values of 8/16-Bit PPG Hardware

14.1 Outline of 8/16-bit PPG

The 8/16-bit Programable Pulse Generator (PPG) consists of two eight-bit down counters, four eight-bit reload registers, one 16-bit control register, two external pulse output signals, and two interrupt outputs. The following functions are implemented:

■ Function of 8/16-bit PPG

❍ **8-bit PPG output, 2-channel independent operation mode:**

Two independent channels of PPG output operation are implemented.

❍ **16-bit PPG output operation mode:**

One channel of 16-bit PPG output operation is implemented.

❍ **8+8-bit PPG output operation mode:**

8-bit PPG output operation is implemented at specifies intervals, using channel 0 output as channel 1 clock input.

❍ **PPG output operation:**

Pulse waves are output at specified intervals and duty ratio. With an external circuit, this module can be used as a D/A converter.

The MB90590 Series contains six PPG's. The follwoing sections only describe the functionality of the PPG 0/1. The remaining PPG's have the identical function and the register addresses should be found in the I/O map.

14.2 Block Diagram of 8/16-Bit PPG

Figure 14.2-1 shows a block diagram of the 8/16-bit PPG (ch0). Figure 14.2-2 shows a block diagram of the 8/16-bit PPG (ch1).

■ Block Diagram of 8/16-bit PPG

Figure 14.2-1 8-bit PPG ch0 Block Diagram

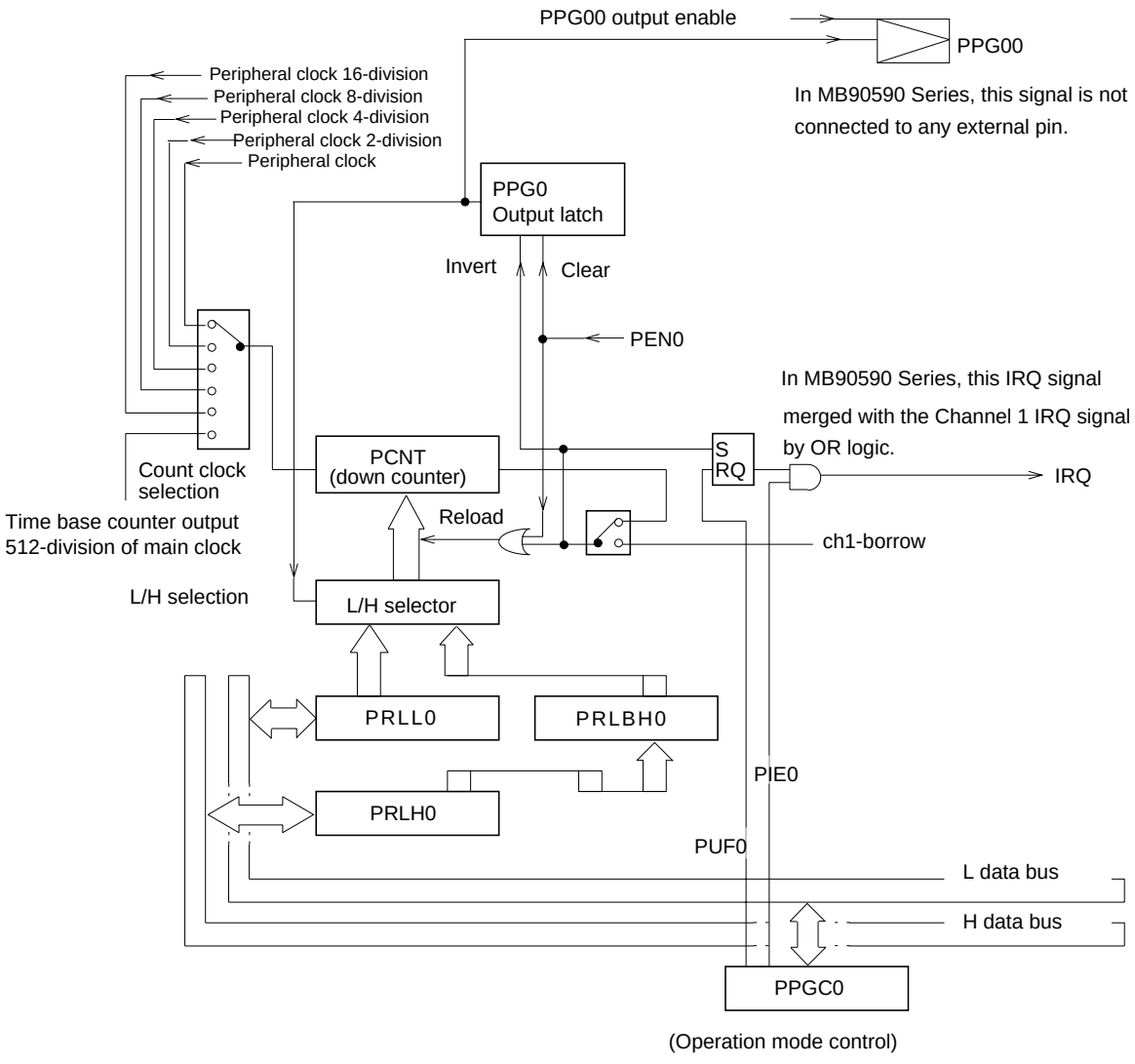
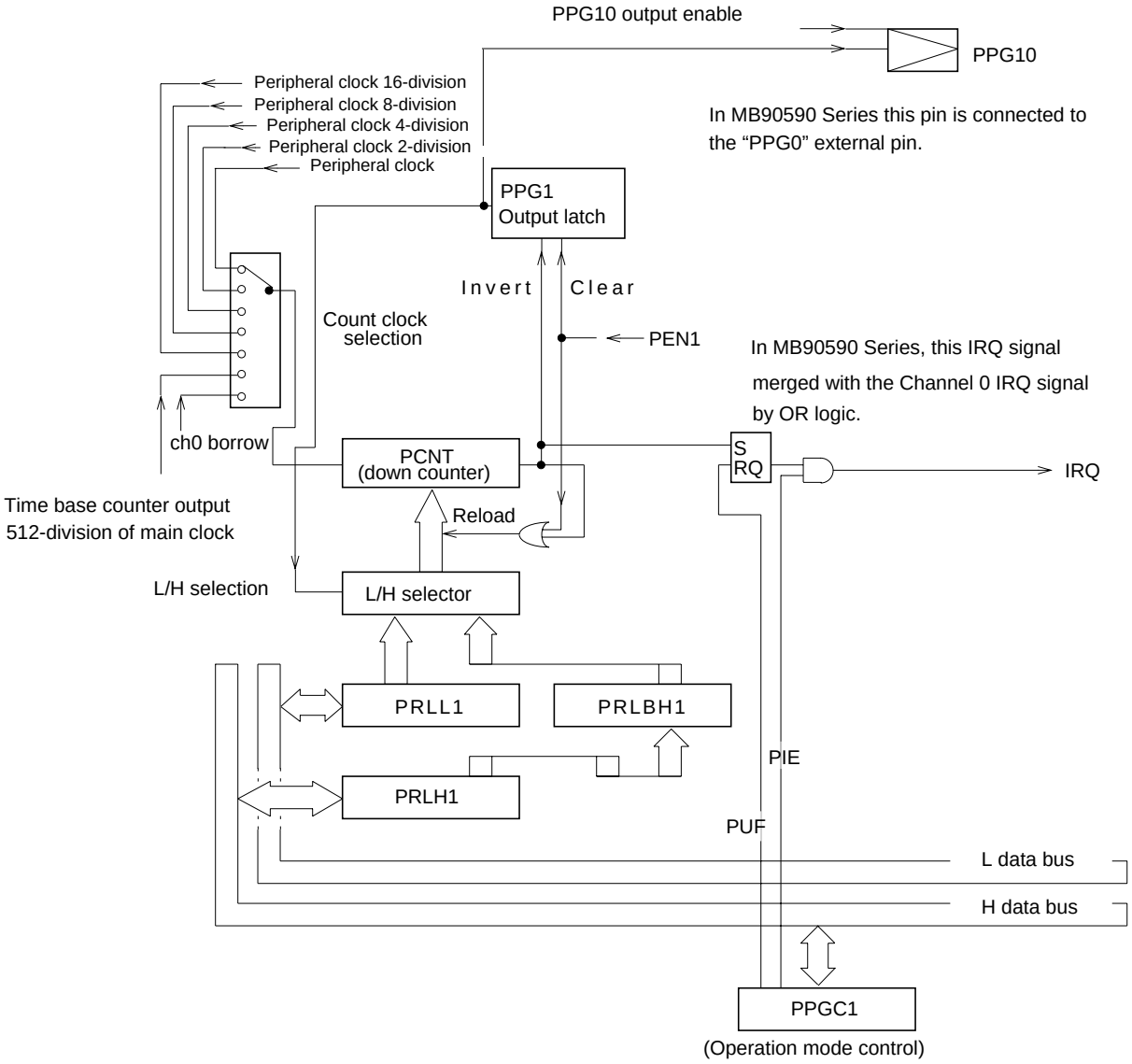


Figure 14.2-2 8-bit PPG ch1 Block Diagram



14.3 8/16-Bit PPG Registers

The 8/16-bit PPG has the following five types of registers:

- **PPG0 operation mode control register**
- **PPG1 operation mode control register**
- **PPG0, 1 operation mode control register**
- **Reload register H**
- **Reload register L**

■ 8/16-bit PPG Registers

PPG0 operation mode control register

Address: ch0 000038H

Bit No.	7	6	5	4	3	2	1	0
	PEN0	—	PE00	PIE0	PUF0	—	—	Reserved
Read/write	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(-)	(-)	(W)
Initial value	(0)	(-)	(0)	(0)	(0)	(-)	(-)	(1)

PPGC0

PPG1 operation mode control register

Address: ch0 000039H

Bit No.	15	14	13	12	11	10	9	8
	PEN1	—	PE10	PIE1	PUF1	MD1	MD0	Reserved
Read/write	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)
Initial value	(0)	(-)	(0)	(0)	(0)	(0)	(0)	(1)

PPGC1

PPG0,1 output control register

Address: ch0 1 003AH

Bit No.	7	6	5	4	3	2	1	0
	PCS2	PCS1	PCS0	PCM2	PCM1	PCM0	—	—
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(-)	(-)
Initial value	(0)	(0)	(0)	(0)	(0)	(0)	(-)	(-)

PPG01

Reload register H

Address: ch0 001901H

ch1 001903H

Bit No.	15	14	13	12	11	10	9	8
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

PRLH

Reload register L

Address: ch0 001900H

ch1 001902H

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

PRLL

14.3.1 PPG0 Operation Mode Control Register (PPGC0)

PPGC0 is a five-bit control register that selects the operation mode of the block, controls pin outputs, selects count clock, and controls triggers.

■ PPG0 Operation Mode Control Register (PPGC0)

PPG0 operation mode control register

Address: ch0, 000038H

	7	6	5	4	3	2	1	0 ⇐ Bit No.
	PEN0	-	PE00	PIE0	PUF0	-	-	Reserved
Read/write ⇨	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(-)	(-)	(W)
Initial value ⇨	(0)	(-)	(0)	(0)	(0)	(-)	(-)	(1)

PPGC0

[bit 7] PEN0 (PPG enable): Operation enable bit

This bit enables the counter operation of the PPG.

PEN0	Operation
0	Stop ('L' level output maintained)
1	PPG operation enabled

Setting this bit to 1 enables the counter operation.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 5] PE00 (PPG output enable 00): PPG00 pin output enable bit

This bit controls the PPG00 pulse output external pin as described below.

0	General-purpose port pin (pulse output disabled)
1	PPG00 = pulse output pin (pulse output enabled)

This bit is initialized to '0' upon a reset. This bit is readable and writable.

For MB90590 Series, this bit should always be set to "0".

[bit 4] PIE0 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

0	Interrupt disabled
1	Interrupt enabled

While this bit is "1", an interrupt request is issued as soon as PUF0 is set to "1". No interrupt request is issued while this bit is set to "0".

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 3] PUF0 (PPG underflow flag): PPG counter underflow bit

This bit indicates the PPG counter underflow as described below.

0	PPG counter underflow is not detected.
1	PPG counter underflow is detected.

In 8-bit PPG 2-channel mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the ch0 counter value becoming from 00H to FFH. In 16-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the Channel 0 and 1 counter value becoming from 0000H to FFFFH. To set this bit to '0,' write '0.' Writing '1' to this bit has not effect. Upon a read operation during a read-modify-write instruction, '1' is read.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 0]

This is a reserved bit. When setting PPGC0, always set this bit to 1.

14.3.2 PPG1 Operation Mode Control Register (PPGC1)

PPGC0 is a seven-bit control register that selects the operation mode of the block, controls pin outputs, selects count clock, and controls triggers.

■ **PPG1 Operation Mode Control Register (PPGC1)**

PPG1 operation mode control register
Address: ch1 000039H

bit	15	14	13	12	11	10	9	8
	PEN1	-	PE10	PIE1	PUF1	MD1	MD0	Reserved
Read/write	(R/W)	(-)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(W)
Initial value	(0)	(-)	(0)	(0)	(0)	(0)	(0)	(1)

⇐ Bit No.

PPGC1

[bit 15] PEN1 (PPG enable): Operation enable bit

This bit enables the counter operation of the PPG.

PEN1	Operation
0	Stop ('L' level output maintained)
1	PPG operation enabled

Setting this bit to 1 enables the counter operation.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 13] PE10 (PPG output enable 10): PPG10 pin output enable bit

This bit controls the PPG10 pulse output external pin as described below.

0	General-purpose port pin (pulse output disabled)
1	PPG10 = pulse output pin (pulse output enabled)

This bit is initialized to '0' upon a reset. This bit is readable and writable.

For MB90590 Series , the pulse signal is output to the "PPG0" external pin.

[bit 12] PIE1 (PPG interrupt enable): PPG interrupt enable bit

This bit controls PPG interrupt as described below.

0	Interrupt disabled
1	Interrupt enabled

While this bit is "1", an interrupt request is issued as soon as PUF1 is set to "1". No interrupt request is issued while this bit is set to "0".

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 11] PUF1 (PPG underflow flag): PPG counter underflow bit

This bit indicates the PPG counter underflow as described below.

0	PPG counter underflow is not detected.
1	PPG counter underflow is detected.

In 8-bit PPG 2-channel mode or 8-bit prescaler + 8-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the Channel 1 counter value becoming from 00H to FFH. In 16-bit PPG mode, this bit is set to "1" when an underflow occurs as a result of the Channel 0 and 1 counter value becoming from 0000H to FFFFH. To set this bit to '0,' write '0.' Writing '1' to this bit has not effect. Upon a read operation during a read-modify-write instruction, '1' is read.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 10, 9] MD1, 0 (PPG count mode): Operation mode selection bit

This bit selects the PPG timer operation mode as described below.

MD1	MD0	Operation mode
0	0	8-bit PPG 2ch independent mode
0	1	8-bit prescaler + 8-bit PPG 1ch mode
0	0	Reserved
0	1	16-bit PPG 1ch mode

This bit is initialized to '00' upon a reset. This bit is readable and writable.

Note:

Do not set '10' in this bit.

To write '01' to this bit, ensure that '01' is not written to the PEN0 bit of PPGC0 or PEN1 bit of PPGC1. Write '11' or '00' in both the PEN0 and PEN1 bits simultaneously.

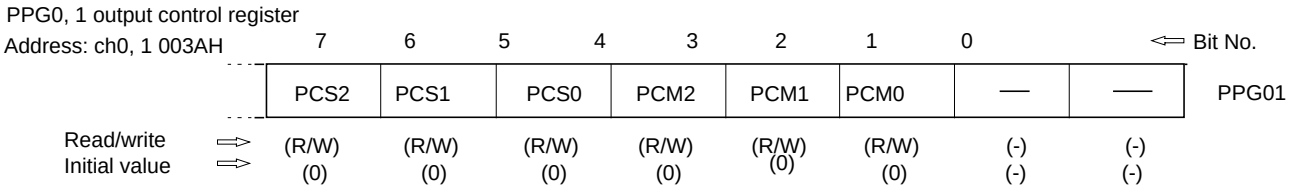
To write '11' to this bit, update PPGC0 and PPGC1 by word transfer and write '11' or '00' to both the PEN0 and PEN1 bits simultaneously.

[bit 8] This is a reserved bit. When setting PPGC1, always write 1 to this bit.

14.3.3 PPG0, 1 Output Control Register (PPG01)

The PPG0, 1 output control register (PPG01) is an 8-bit control register that controls the pin output of the 8/16-bit PPG.

■ PPG0, 1 Clock Select Register (PPG01)



[bits 7 to 5] PCS2 to 0 (PPG count select): Count clock selection bit

These bits select the operation clock for the down counter of Channel 1 as described below.

PCS2	PCS1	PCS0	Operation mode
0	0	0	Peripheral clock (62.5-ns machine clock, 16 MHz)
0	0	1	Peripheral clock/2 (125-ns machine clock, 16 MHz)
0	1	0	Peripheral clock/4 (250-ns machine clock, 16 MHz)
0	1	1	Peripheral clock/8 (500-ns machine clock, 16 MHz)
1	0	0	Peripheral clock/16 (1-µs machine clock, 16 MHz)
1	0	1	Clock input fromthe time base timer (128-us, 4-MHz source oscillation)

This bit is initialized to '000' upon a reset. This bit is readable and writable.

Note:

In 8-bit prescaler + 8-bit PPG mode or in 16-bit PPG mode, ch1 PPG operates in response to a counter clock from ch0. Therefore, the setting in these bits has no effect.

[bits 4 to 2] PCM2 to 0 (PPG count mode): Count clock selection bit

These bits select the operation clock for the down counter of Channel 0 as described below.

PCS2	PCS1	PCS0	Operation mode
0	0	0	Peripheral clock (62.5-ns machine clock, 16 MHz)
0	0	1	Peripheral clock/2 (125-ns machine clock, 16 MHz)
0	1	0	Peripheral clock/4 (250-ns machine clock, 16 MHz)
0	1	1	Peripheral clock/8 (500-ns machine clock, 16 MHz)

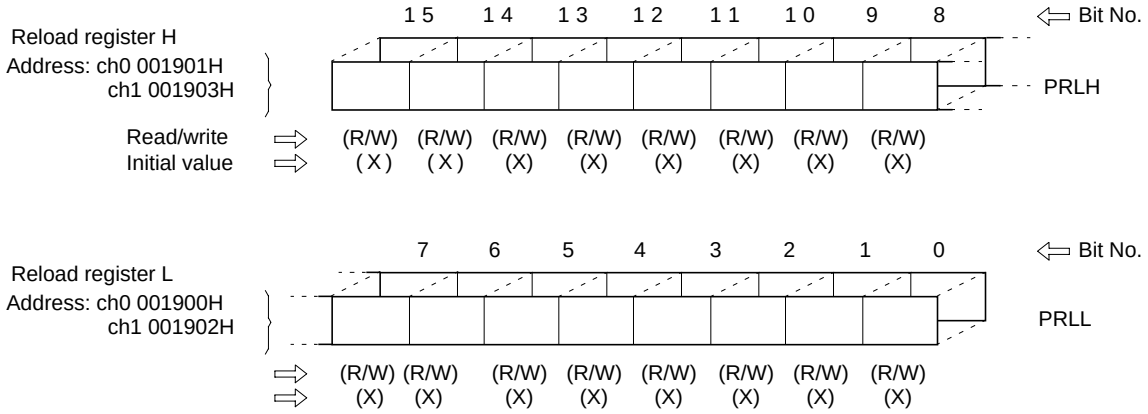
PCS2	PCS1	PCS0	Operation mode
1	0	0	Peripheral clock/16 (1-µs machine clock, 16 MHz)
1	0	1	Clock input from the time base timer (128-us, 4-MHz source oscillation)

This bit is initialized to '000' upon a reset. This bit is readable and writable.

14.3.4 Reload Register (PRLL/PRLH)

The reload registers (PRLL and PRLH) are 8-bit registers that store reload values for the PCNT down counters. The PRLL and PRLH registers are readable and writable.

■ **Reload Register (PRLL/PRLH)**



Register name	Function
0	Holds the L side reload value.
1	Holds the H side reload value.

Note:

In 8-bit prescaler + 8-bit PPG mode, different values in PRLL and PRLH of Channel 0 may cause the PPG waveform of ch1 to vary in each cycle. Write the same value to PRLL and PRLH of ch0.

14.4 Operations of 8/16-bit PPG

One 8/16-bit PPG consists of two channels of 8-bit PPG units. These two channels can be used in three modes: independent two-channel mode, 8-bit prescaler + 8-bit PPG mode, and single-channel 16-bit PPG mode.

■ Operations of 8/16-bit PPG

Each of the 8-bit PPG units has two eight-bit reload registers. One reload register is for the L pulse width (PRLL) and the other is for the H pulse width (PRLH). The values stored in these registers are reloaded into the 8-bit down counter (PCNT), from the PRLL and PRLH in turn. The pin output value is inverted upon a reload caused by counter borrow. This operation results in the pulses of the specified L pulse width and H pulse width.

Table 14.4-1 lists the relationship between the reload operation and pulse outputs.

Table 14.4-1 Reload Operation and Pulse Output

Reload operation	Pin output change		
PRLH => PCNT	PPG0/1 [0 => 1]	↑	Rise
PRLL => PCNT	PPG0/1 [1 => 0]	↓	Fall

When 1 is set in bit 4 (PIE0) of PPGC0 or in bit 12 (PIE1) of PPGC1, an interrupt request is output upon a borrow from 00 to FF (from 0000 to FFFF in 16-bit PPG mode) of each counter.

■ Operation Modes of 8/16-bit PPG

This block can be used in three modes: independent two-channel mode, 8-bit prescaler + 8-bit PPG mode, and single-channel 16-bit PPG mode.

❍ Independent two-channel mode

The two channels of 8-bit PPG units operate independently. The PPG00 pin is connected to the ch0 PPG output, while the PPG10 pin is connected to the ch1 PPG output.

❍ 8-bit prescaler + 8-bit PPG mode

ch0 is used as an 8-bit prescaler while the count in ch1 is based on borrow outputs from ch0. Thus, 8-bit PPG waveforms can be output with arbitrary length of cycle time. The PPG00 pin is connected to the ch0 prescaler output, while the PPG10 pin is connected to the ch1 PPG output.

❍ 16-bit PPG 1ch mode

ch0 and ch1 are connected and used as a single 16-bit PPG. The PPG00 and PPG10 pins are connected to the 16-bit PPG output.

For the MB90590 Series, the output signal from the Channel 0 PPG is not connected to any external pin.

■ 8/16-bit PPG Output Operation

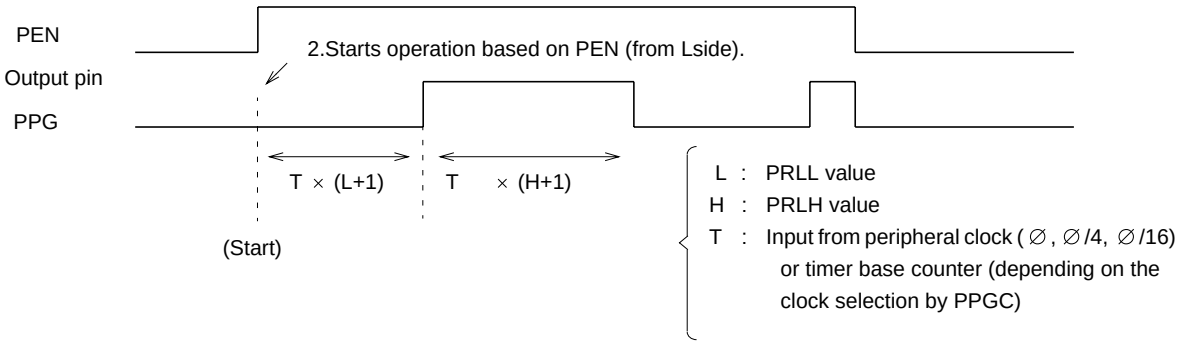
In this block, the ch0 PPG is activated to start counting when '1' is written to bit 7 (PEN0) of the PPGC0 (PWM operation mode control) register. Similarly, the ch1 PPG is activated to start counting when '1' is written to bit 15 (PEN1) of the PPGC1 register. Once the operation has started, counting is terminated by writing '0' to bit 7 (PEN0) of PPGC0 or in bit 15 (PEN1) of PPGC1. Once the counting is terminated, the output is maintained at the L level.

For the MB90590 Series, the output signal from the Channel 0 PPG is not connected to any external pin.

In 8-bit prescaler + 8-bit PPG mode, do not set ch1 to be in operation while ch0 operation is stopped.

In 16-bit PPG mode, ensure that bit 7 (PEN0) of PPGC0 register and bit 15 (PEN1) of PPGC1 register are started or stopped simultaneously. The figure below is a diagram of PPG output operation. During PPG operation, a pulse wave is continuously output at a frequency and duty ratio (the ratio of the H-level period of the pulse wave to the L-level period). PPG continues operation until stop is specified explicitly.

Figure 14.4-1 PPG Output Operation, Output Waveform



■ Relationship Between 8/16-bit PPG Reload Value and Pulse Width

The width of the output pulse is determined by adding 1 to the reload register value and multiplying it by the count clock cycle. Note that when the reload register value is 00_H during 8-bit PPG operation or 0000_H during 16-bit PPG operation, the pulse width is equivalent to one count clock cycle. In addition, note that when the reload register value is FF_H during 8-PPG operation, the pulse width is equivalent to 256 count clock cycles. When the reload register value is $FFFF_H$ during 16-bit PPG operation, the pulse width is equivalent to 65536 count clock cycles.

P1=T × (L+1)
Ph=T × (H+1)

L : PRLL value
H : value
T : Input clock cycle
Ph : High pulse width
Pl : Low pulse width

14.5 Selecting a Count Clock for 8/16-Bit PPG

The count clock used for the operation is supplied from the peripheral clock or the time base timer. The count clock can be selected from six choices.

■ Selecting a Count Clock for 8/16-bit PPG

Select ch0 clock at bit 4 to 2 (PCM2 to 0) of the PPG01 register, and ch1 clock at bit 7 to 5 (PCS2 to 0) of the PPG01 register.

The clock is selected from a peripheral clock 1/16 to 1 times higher than a machine clock or an input clock from the time base timer.

In 8-bit prescaler + 8-bit PPG mode or 16-bit PPG mode, however, the setting in the PCS2 to 0 has no effect.

When the time base timer input is used, the first count cycle after a trigger or a stop may be shifted. The cycle may also be shifted if the time base counter is cleared during operation of this module.

In 8-bit prescaler + 8-bit PPG mode, if ch1 is activated while ch0 is in operation and ch1 is stopped, the first count cycle may be shifted.

14.6 Controlling Pin Output of 8/16-bit PPG Pulses

The pulses generated by this module can be output from external pins PPG00 and PPG10.

■ Controlling Pin Output of 8/16-bit PPG Pulses

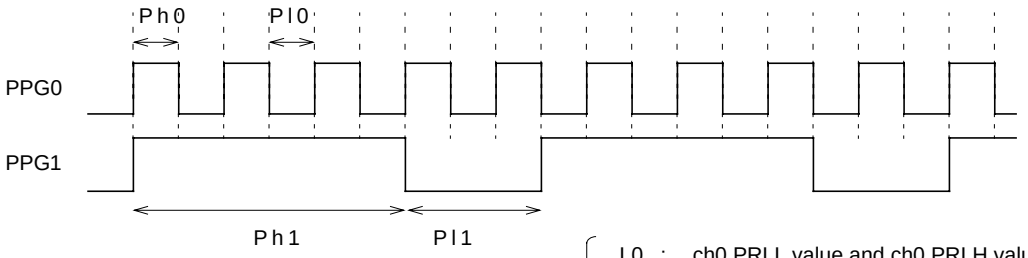
To output the pulses from an external pin, write '1' to the bit corresponding to each pin. When '0' is written to these bits (default), the pulses are not output from the corresponding external pins; the pins work as general-purpose ports.

In 16-bit PPG mode, the same waveform is output from PPG00 and PPG10. Thus, the same output can be obtained by enabling both external pin.

In 8-bit prescaler + 8-bit PPG mode, the 8-bit prescaler toggle output waveform is output from PPG00, while the 8-bit PPG waveform is output from PPG10. Figure 14.6-1 is a diagram of output waveforms in this mode.

For the MB90590 Series, the output signal from the Channel 0 PPG is not connected to any external pin.

Figure 14.6-1 8+8 PPG Output Operation Waveform



Pl0 = T × (L0+1)
Ph0 = T × (L0+1)
Pl1 = T × (L0+1) × (Ll+1)
Ph1 = T × (L0+1) × (Hl+1)

L0 : ch0 PRLL value and ch0 PRLH value
L1 : ch1 PRLL value
H1 : ch1 PRLH value
T : Input clock cycle
Ph0 : PPG00 and 01 high pulse width
Pl0 : PPG00 and 01 low pulse width
Ph1 : PPG10 and 11 high pulse width
Pl1 : PPG10 and 11 low pulse width

Note:

Set the same value in ch0 PRLL and ch0 PRLH.

14.7 8/16-bit PPG Interrupts

For the 8/16-bit PPG, an interrupt becomes active when the reload value counts out and a borrow occurs.

■ 8/16-bit PPG Interrupts

In 8-bit PPG 2ch mode or 8-bit prescaler + 8-bit PPG mode, an interrupt is requested by a borrow in each counter. In 16-bit PPG mode, PUG0 and PUF1 are simultaneously set by a borrow in the 16-bit counter. Therefore, enable only PIE0 or PIE1 to unify the interrupt causes. In addition, simultaneously clear the interrupt flags for PUF0 and PUF1.

14.8 Initial Values of 8/16-bit PPG Hardware

The hardware components of this block are initialized to the following values when reset:

■ Initial Values of 8/16-bit PPG Hardware

❍ **<Registers>**

- PPGC0 -> 0X000001B
- PPGC1 -> 00000001B
- PPG10 -> XXXXXX00B

❍ **<Pulse outputs>**

- PPG00 -> 'L'
- PPG10 -> 'L'
- PE00 -> PPG00 output disabled
- PE10 -> PPG10 output disabled

❍ **<Interrupt requests>**

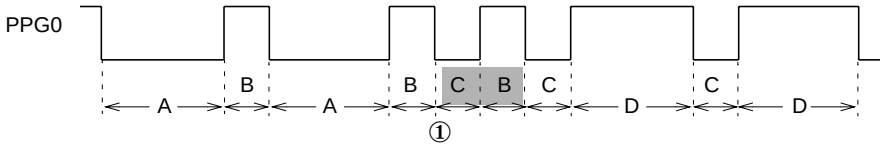
- IRQ0 -> 'L'
- IRQ1 -> 'L'

Hardware components other than the above are not initialized.

Note:

In a mode other than 16-bit PPG mode, it is recommended to use a word transfer instruction to write data in reload registers PRLL and PRLH. If two byte transfer instructions are used to write a data item to these registers, a pulse of unexpected cycle time may be output depending on the timing.

Figure 14.8-1 Write Timing for 8/16-bit PPG Reload Registers (PRLL and PRLH)

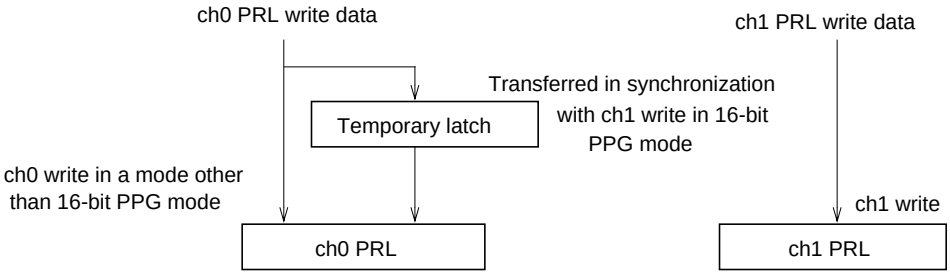


Assume that PRLL is updated from A to C before point 1 in the time chart above, and PRLH is updated from B to D after point 1. Since the PRL values at point 1 are PRLL=C and PRLH=B, a pulse of L side count value C and H side count value B is output only once.

Similarly, to write data in PRL of ch0 and ch1 in 16-bit PPG mode, use a long word transfer instruction, or use word transfer instructions in the order of ch0 and then ch1. In this mode, the data is only temporarily written to ch0 PRL. Then, the data is actually written into ch0 PRL when the ch1 PRL is written to.

In a mode other than 16-bit PPG mode, ch0 and ch1 PRL are written independently.

Figure 14.8-2 PRL Write Operation Block Diagram



CHAPTER 15 DTP/EXTERNAL INTERRUPTS

This chapter explains the functions and operations of the DTP/external interrupts.

15.1 Outline of DTP/External Interrupts

15.2 DTP/External Interrupt Registers

15.3 Operations of DTP/External Interrupts

15.4 Switching Between External Interrupt and DTP Requests

15.5 Notes on Using DTP/External Interrupts

15.1 Outline of DTP/External Interrupts

The data transfer peripheral (DTP) is located between an external peripheral and the F^{2}MC-16LX CPU. The DTP receives a DMA request or interrupt request from the external peripheral, transfers the request to the F^{2}MC-16LX CPU to activate the intelligent I/O service or interrupt processing.

■ Outline of DTP/External Interrupts

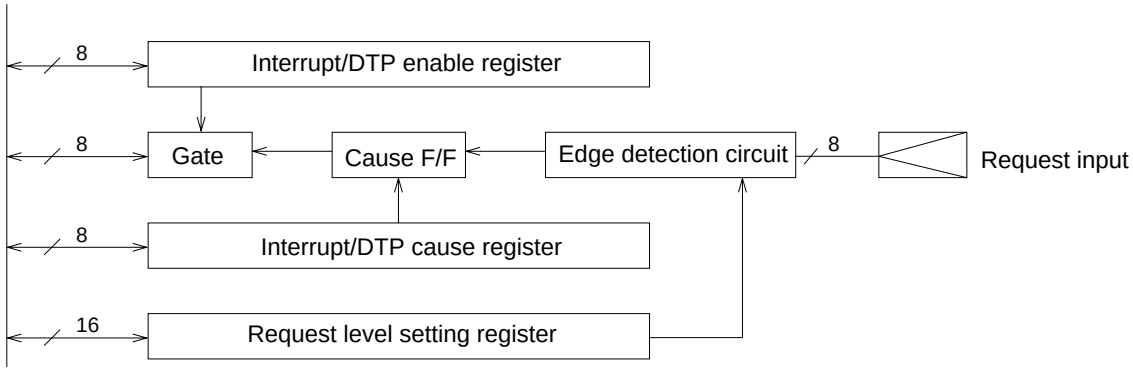
For the intelligent I/O service, 'H' and 'L' request levels are available. For an external interrupt request, four request levels are available: 'H,' 'L,' rising edge, and falling edge.

For the MB90590 Series, the external bus interface is not supported. Therefore the DTP/ External Interrupt can not serve as the data transfer peripheral. It can be only used as the External Interrupt.

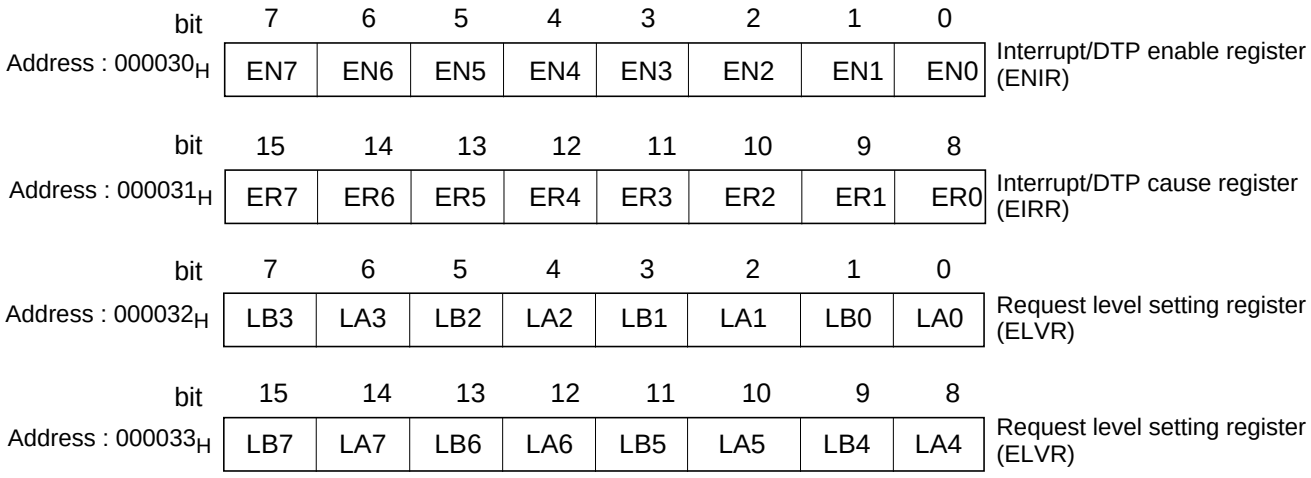
For MB90V590, there are only four external pins assigned to this block. Therefor the external interrupt channel 4 to 7 are not supported. These external interrupts should be disabled.

■ Block Diagram of DTP/External Interrupts

Figure 15.1-1 Block Diagram of DTP/External Interrupts



■ DTP/external Interrupts Registers



15.2 DTP/External Interrupt Registers

The DTP/external interrupts has the following three types of registers:

- **Interrupt/DTP enable register (ENIR: Interrupt request enable register)**
- **Interrupt/DTP flag (EIRR: External interrupt request register)**
- **Request level setting register (ELVR: External level register)**

■ Interrupt/DTP Enable Register (ENIR: Interrupt request enable register)

	7	6	5	4	3	2	1	0	Initial value
ENIR Address : 000030_H	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	00000000_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

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

■ Interrupt/DTP Flags (EIRR: External interrupt request register)

	15	14	13	12	11	10	9	8	Initial value
EIRR Address : 000031_H	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	XXXXXXXX $_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	·····The objects differ for R and W.

The EIRR indicates the presence of external interrupt/DTP requests at the pins corresponding to the '1' bits of this register. Writing '0' to a bit of this register clears the corresponding request flag. Writing '1' has no effect. '1' is always read from this register by a read-modify-write instruction.

■ Request Level Setting Register (ELVR: External level register)

	7	6	5	4	3	2	1	0	Initial value
Address : 000032 H	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	00000000_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

	7	6	5	4	3	2	1	0	Initial value
Address : 000033 H	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	00000000_B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

ELVR defines the request event at the external pin. Each pin is assigned two bits as described in Table 15.2-1. If a request is detected by the input level, the interrupt flag is set as long as the input is at the specified level even after the flag is reset by software.

Table 15.2-1 Interrupt Request Detection Factor for LBx and LAx Pins

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

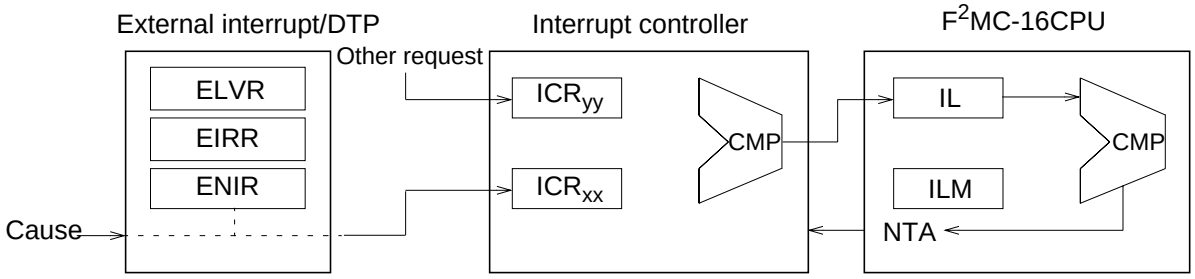
15.3 Operations of DTP/External Interrupts

When the interrupt flag is set, this block signals an interrupt to the interrupt controller. The interrupt controller judges the priority levels of the simultaneous interrupts, and issues an interrupt request to the F^{2}MC-16LX CPU if the interrupt from this block has the highest priority. The F^{2}MC-16LX CPU compares the ILM bits of its internal CCR register and the interrupt request. If the interrupt level of the request is higher than that indicated by the ILM bits, the F^{2}MC-16LX CPU activates the hardware interrupt processing microprogram as soon as the currently executing instruction is terminated.

■ External Interrupt Operation

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

Figure 15.3-1 External Interrupt



■ DTP operation

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

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

Figure 15.3-2 Timing to Cancel the External Interrupt at the End of DTP Operation

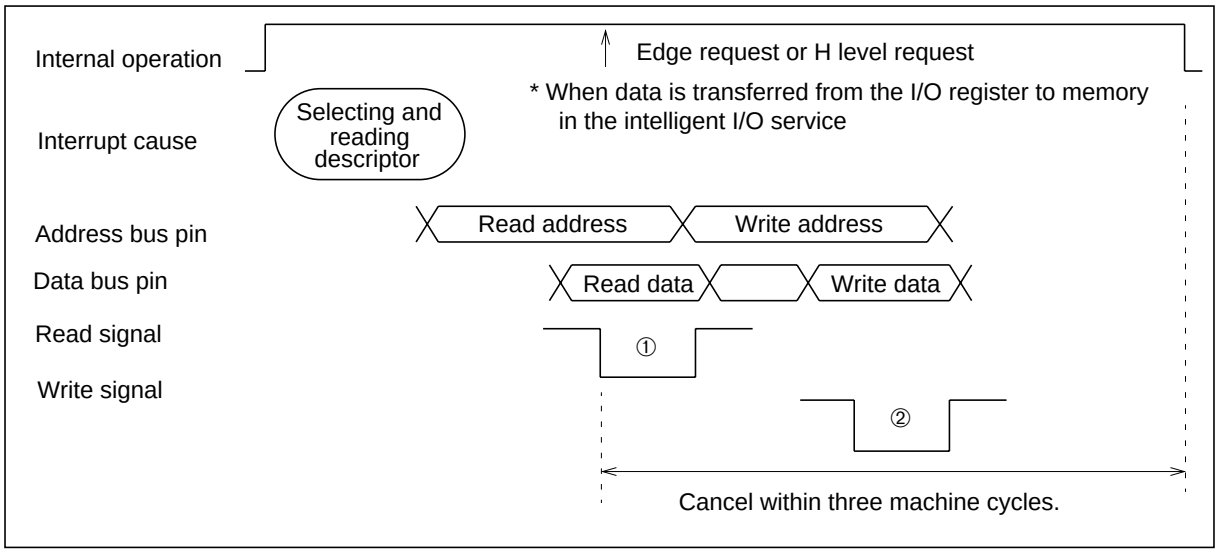
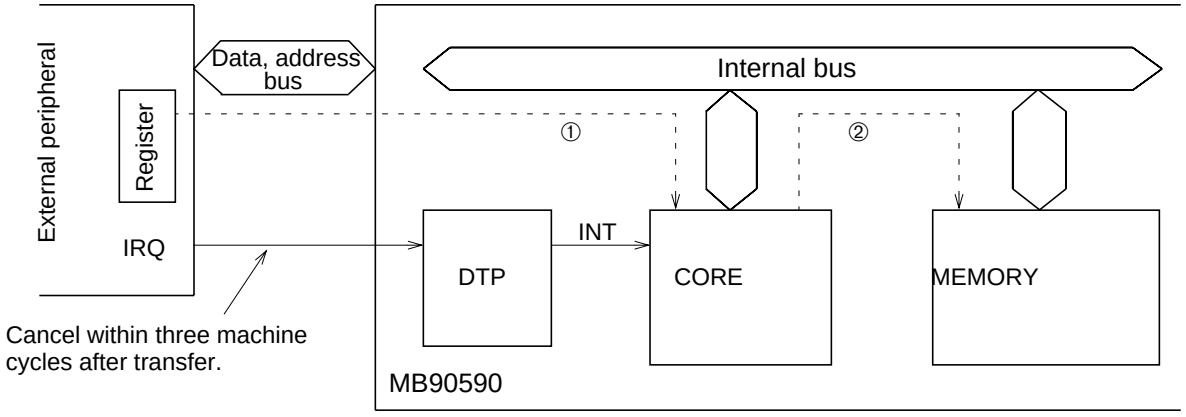


Figure 15.3-3 Sample Interface to the External Peripheral

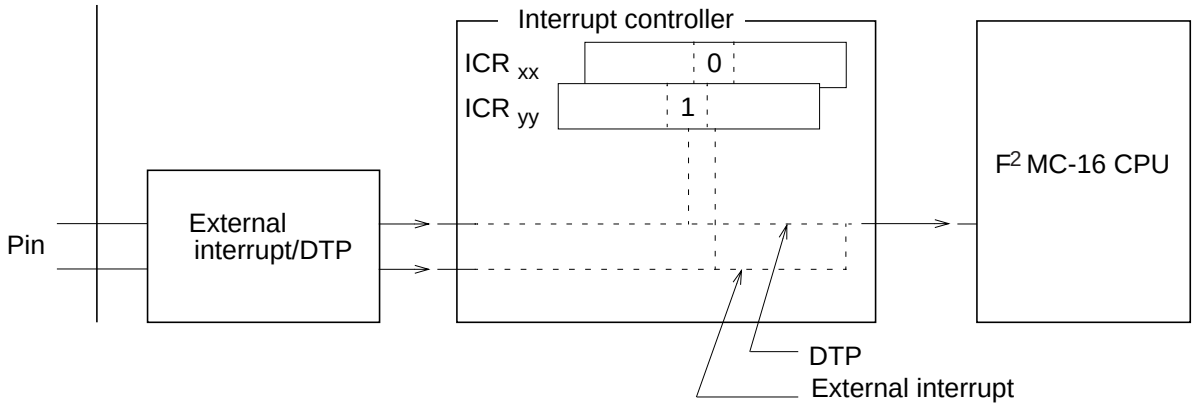


15.4 Switching between External Interrupt and DTP Requests

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

■ Switching Between External Interrupt and DTP Requests

Figure 15.4-1 Switching Between External Interrupt and DTP Requests



15.5 Notes on Using DTP/External Interrupts

Note carefully the following items when using DTP/external interrupts:

- **Conditions on the externally connected peripheral when DTP is used**
- **Recovery from standby**
- **External interrupt/DTP operation procedure**
- **External interrupt request level**

■ Notes on Using DTP/External Interrupts

❍ Conditions on the externally connected peripheral when DTP is used

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

❍ Recovery from standby

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

❍ External interrupt/DTP operation procedure

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

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

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

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

❍ External interrupt request level

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

As shown in Figure 15.5-1, when the request input level is related to the level setting, a request that is input from an external device to the interrupt controller is kept active even if the request is later canceled because a cause hold circuit has been installed. To cancel the request to the interrupt controller, the cause hold circuit must be cleared as shown in Figure 15.5-2.

Figure 15.5-1 Clearing the Cause Hold Circuit Upon Level Set

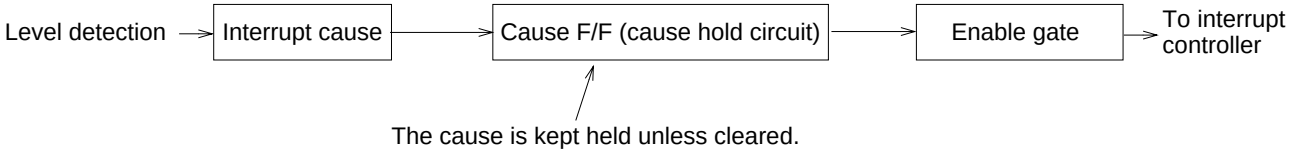
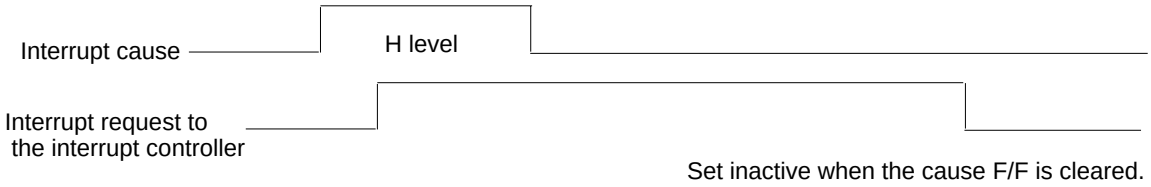


Figure 15.5-2 Interrupt Cause and Interrupt Request to the Interrupt Controller While Interrupts are Enabled



CHAPTER 16 A/D Converter

This chapter explains the functions and operations of the A/D converter.

16.1 Features of A/D Converter

16.2 Block Diagram of A/D Converter

16.3 A/D Converter Registers

16.4 Operations of A/D Converter

16.5 Conversion Using EI^{2}OS

16.6 Conversion Data Protection

16.1 Features of A/D Converter

The A/D converter converts analog input voltages into digital values. The A/D converter has the following features:

■ Features of A/D converter

- **Conversion time:**

  26.3 µs min. per channel (at 16 MHz machine clock)

- **RC sequential compare conversion with sample and hold circuit**

- **10-bit resolution**

- **Analog input selected from eight channels by programming**

  Single conversion mode: One channel is selected for conversion.

  Scan conversion mode: Voltages in multiple consecutive channels are converted. Up to eight channels can be programmed.

  Continuous conversion mode: Voltages at the specified channel are converted repeatedly.

  Stop conversion mode: Voltages at the specified channel are converted, then the system pauses and stands by for the next activation. (The conversion start points can be synchronized.)

- **Interrupt request**

  At the end of A/D conversion, a relevant interrupt request can be issued to the CPU. This interrupt can be used to activate the EI^2OS, which automatically transfers A/D conversion result to memory. This feature is suitable for continuous processing.

- **Selectable activation cause**

  The activation can be done by software, external trigger (falling edge), or timer (rising edge).

■ Analog Input Enable Register

Always write '1' to the ADER bit corresponding to a pin used as analog input.

bit	15	14	13	12	11	10	9	8	
Address: 00001B$_H$	ADE7	ADE6	ADE5	ADE4	ADE3	ADE2	ADE1	ADE0	Initial value 11111111$_B$
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Port 6 pins are controlled as described below.

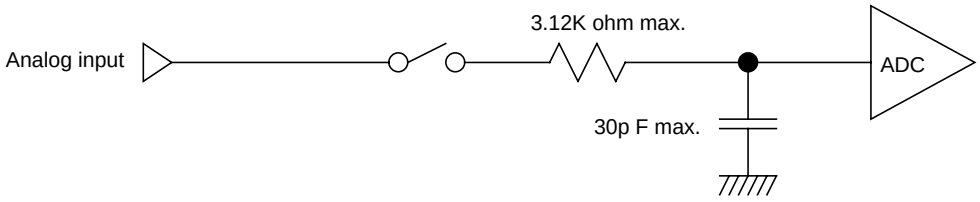
0: Port input/output mode

1: Analog input mode

'1' is set upon a reset.

■ Input Impedance

The sampling circuit of the A/D Converter can be represented with the equivalent circuit shown below.



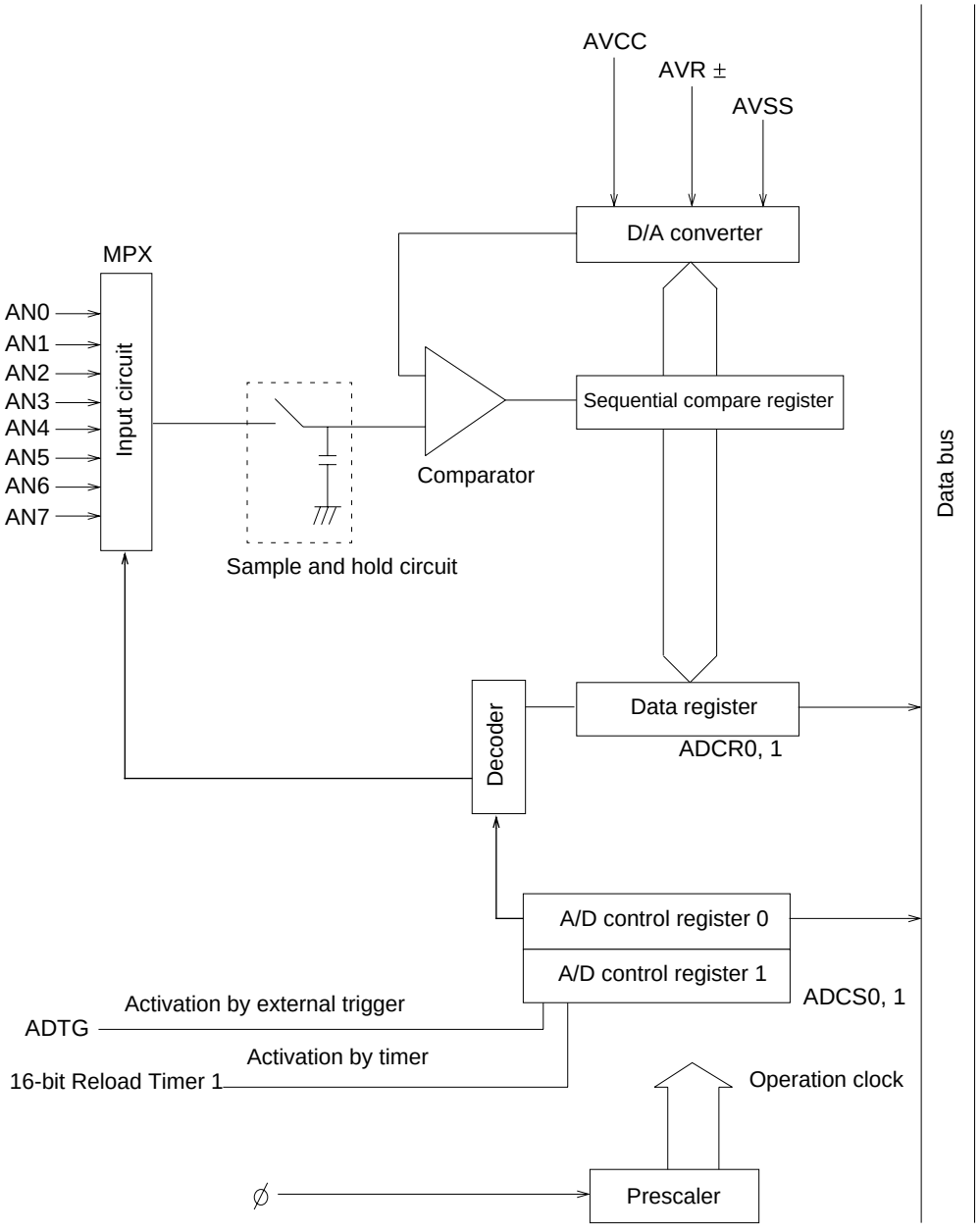
Driving impedance to an analog input should be lower than 15.5K ohm when the sampling time is set to 4μs (ST=0 and ST0=0 at 16MHz machine clock). Otheriwse the conversion accuracy will be worsened. If this is the case, set the sampling time longer (ST1=1 and ST0=1) or add external capacitor in order to compensate the driving impedance.

16.2 Block Diagram of A/D Converter

Figure 16.2-1 shows a block diagram of the A/D converter.

■ Block Diagram of A/D Converter

Figure 16.2-1 Block Diagram of A/D Converter



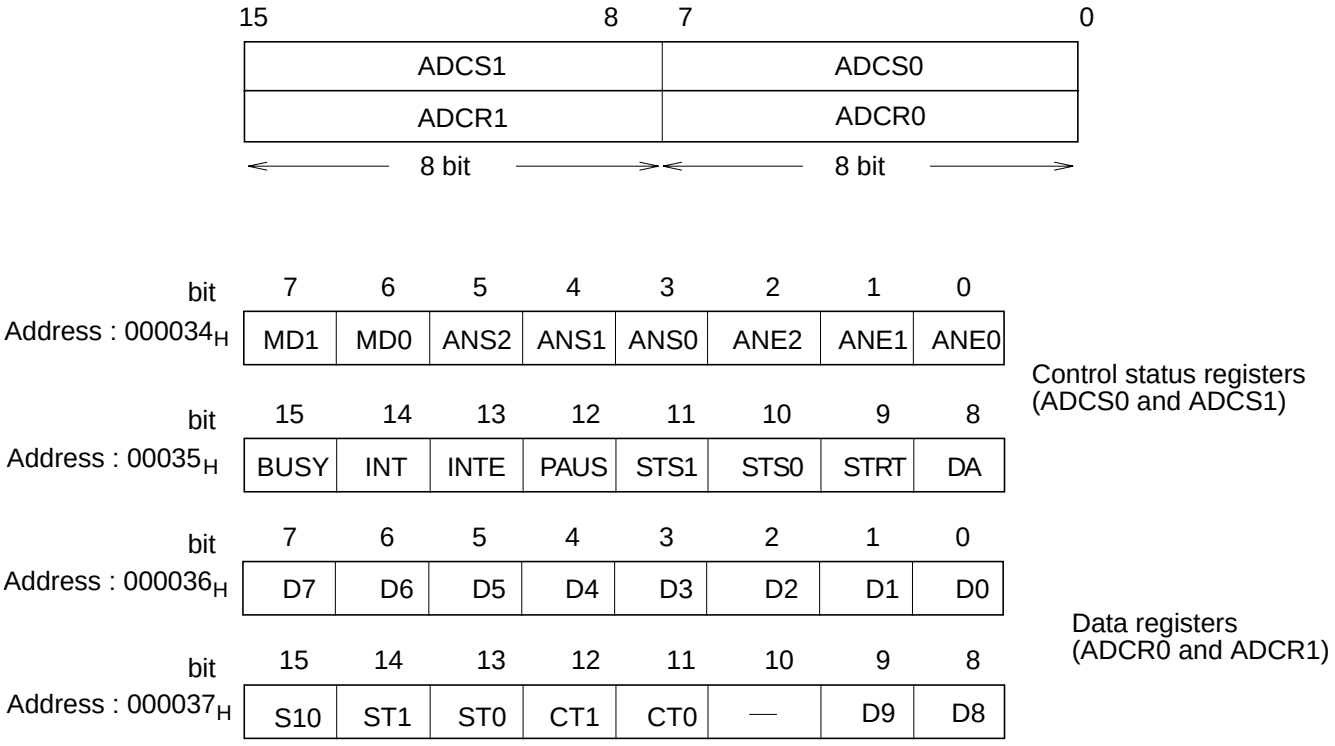
16.3 A/D Converter Registers

The A/D converter has the following two types of registers:

- **Control status resister**
- **Data register**

■ A/D Converter Registers

Figure 16.3-1 A/D Converter Register Configuration



16.3.1 Control Status Registers (ADCS0)

The control status register (ADCS0) controls the A/D converter and indicates the status. Do not rewrite ADCS0 during A/D conversion.

■ **Control Status Registers (ADCS0)**

	7	6	5	4	3	2	1	0	
ADCS0 Address: 000034$_H$	MD1	MD0	ANS2	ANS1	ANS0	ANE2	ANE1	ANE0	
	0 R/W	0 R/W	0 R/W	0 R/W	0 R/W	0 R/W	0 R/W	0 R/W	←Initial value ←Bit attribute

[bits 7 and 6] MD1 and MD0 (A/D converter mode set):

Table 16.3-1 Operation Mode Setting

MD1	MD0	Operation mode
0	0	Single mode. Reactivation during operation is allowed.
0	1	Single mode. Reactivation during operation is not allowed.
1	0	Continuous mode. Reactivation during operation is not allowed.
1	1	Stop mode. Reactivation during operation is not allowed.

❍ **Single mode:**

A/D conversion is continuously performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0. The conversion stops once it has been done for all these channels.

❍ **Continuous mode:**

A/D conversion is repeatedly performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0.

❍ **Stop mode:**

A/D conversion is performed from the channel specified with ANS2 to ANS0 to the channel specified with ANE2 to ANE0, pausing for each channel. The A/D conversion is resumed upon an activation.

Upon a reset, these bits are initialized to '00.'

Note:

When activated in the continuous or stop mode, A/D conversion continues until it is stopped by the BUSY bit.

The conversion is stopped by writing '0' to the BUSY bit.

Reactivation disabled in single mode, continuous mode, and stop mode applies to all kinds

of activation by software, an external trigger, and a timer.

[bits 5, 4, and 3] ANS2, ANS1, and ANS0 (Analog start channel set):

Use these bits to specify the start channel for A/D conversion.

When the A/D converter is activated, A/D conversion starts from the channel selected with these bits.

ANS2	ANS1	ANS0	Start channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

* Read

During A/D conversion, the current conversion channel is read from these bits. If the system is stopped in the stop mode, the last conversion channel is read.

* Upon a reset, these bits are initialized to '000.'

[bits 2, 1, and 0] ANE2, ANE1, and ANE0 (Analog end channel set):

Use these bits to set the A/D conversion end channel.

ANS2	ANS1	ANS0	End channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

Note:

When the same channel is written to ANE2 to ANE0 and ANS2 to ANS0, conversion is performed for one channel only (single conversion).

In the continuous or stop mode, operation returns to the start channel specified in ANS2 to ANS0 after the conversion is completed for the channel specified in ANE2 to ANE0.

If the ANS value is greater than the ANE value, conversion starts from the ANS channel. Then, once conversion is complete up to channel 7, operation returns to channel 0 and

conversion is performed up to the ANE channel.

Upon a reset, these bits are initialized to '000.'

Example: ANS=6, ANE=3, single mode

Conversion is performed in the following sequence: CH6, CH7, CH0, CH1, CH2, CH3

16.3.2 Control Status Register (ADCS1)

The control status register (ADCS1) controls the A/D converter and indicates the status.

■ **Control Status Register (ADCS1)**

	15	14	13	12	11	10	9	8	
ADCS1 Address: 000035_H	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	DA	
	0 R/W	0 R/W	0 R/W	0 R/W	0 R/W	0 R/W	0 W	0 R/W	←Initial value ←Bit attribute

[bit 15] BUSY (busy flag and stop):

- Read

This bit indicates the A/D converter operation.

This bit is set when A/D conversion starts and is cleared when the conversion ends.

- Write

Writing "0" to this bit during A/D conversion forces the conversion to terminate.

The above feature is used for forced stop in continuous or stop mode.

"1" cannot be written to the BUSY bit. With a read-modify-write (RMW) instruction, "1" is read from this bit. In single mode, this bit is cleared at the end of A/D conversion.

In continuous or stop mode, this bit is not cleared until conversion is stopped by writing "0."

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

Do not perform a forced stop and activation by software simultaneously (BUSY = 0, STRT = 1).

[bit 14] INT (Interrupt):

This bit is set when conversion data is written to ADCR.

An interrupt request is issued if this bit is set while bit 5 (INTE) is '1.' In addition, the EI^2OS is activated if it is enabled. Writing '1' has no effect.

This bit is cleared by writing '0' or by the EI^2OS interrupt clear signal.

Note: To clear this bit by writing '0,' ensure that A/D conversion is not in progress.

This bit initialized to '0' upon a reset.

[bit 13] INTE (Interrupt enable):

This bit is used to enable or disable interrupts at the end of conversion.

- 0: Interrupts are disabled.
- 1: Interrupts are enabled.

Set this bit when using the EI^2OS. The EI^2OS is activated when an interrupt request is

issued.

Upon a reset, this bit is initialized to '0.'

[bit 12] PAUS (A/D conversion pause):

This bit is set when the A/D conversion is paused.

Only one register is available for storing the A/D conversion result. Therefore, unless the conversion results are transferred by the EI^{2}OS, the result data would be continuously updated and destroyed in continuous conversion.

To prevent the above condition, the system is designed so that a data register value must be transferred by the EI^{2}OS before the next conversion data is saved. A/D conversion pauses during that period. A/D conversion is resumed at the end of transfer by the EI^{2}OS.

This register is valid only when the EI^{2}OS is used.

Note:

For the conversion data protection function, see Section 16.4, "Operations."

Upon a reset, this bit is initialized to '0.'

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

Upon a reset, these bits are initialized to '00.'

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

STS1	STS0	Function
0	0	Activation by software
0	1	Activation by external pin trigger and software
1	0	Activation by timer and software
1	1	Activation by external pin trigger, timer, and software

In a mode allowing two or more activation factors, A/D conversion is activated by the souce that occures first.

The activation source setting changes as soon as it is updated. Thus, take care when updating it during A/D conversion.

Note:

The external pin trigger is detected by the falling edge. If this bit is updated to external trigger activation while the external trigger input level is 'L,' A/D may be activated at once.

When timer is selected, the 16-bit Reload Timer 1 is selected.

[bit 9] STRT (Start):

A/D conversion is activated when '1' is written to this bit.

To reactivate A/D conversion, write '1' to this bit again.

Upon a reset, this bit is initialized to '0.'

In the stop mode, a reactivation during the operation is not supported. Check the BUSY bit before writing '1'.

Do not perform a forced stop and activation by software simultaneously. (BUSY=0, STRT=1)

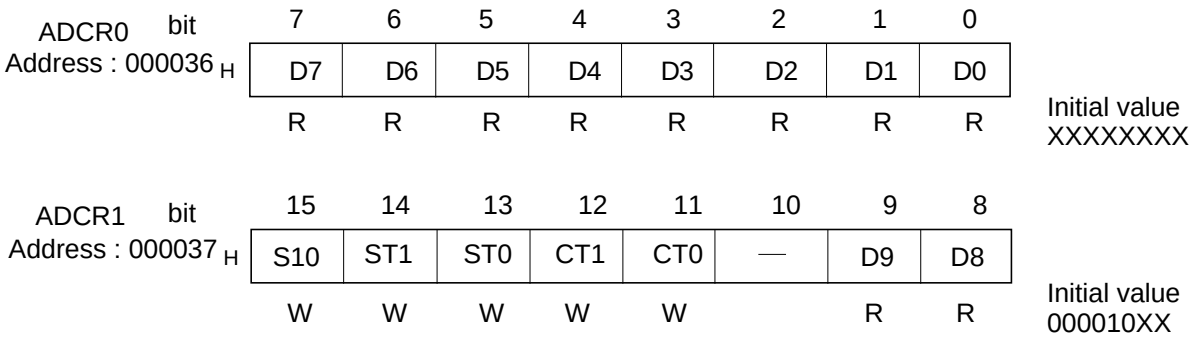
[bit 8] DA

This is a test bit. Always write '0' to this bit.

16.3.3 Data Registers (ADCR1 and ADCR0)

These registers are used to store the digital values produced as a result of the conversion. ADCR1 stores the most significant two bits of the conversion result, while ADCR0 stores the lower eight bits. These register values are updated each time conversion is completed. Usually, the final conversion value is stored in these bits.

■ **Data Registers (ADCR1 and ADCR0)**



'0' is always read from the bits 10 to 15 of ADCR1.

The conversion data protection function is available. See Section 2.7.4, "Operations."

Ensure that no data is written to these registers during A/D conversion.

[bits 15] S10

This bit specifies the resolution of the conversion. When it is set to "0", the 10-bit A/D convertion is performed. Otherwise the 8-bit A/D conversion is performed and the result is stored in the D7 to D0.

Reading this bit always returns "0".

[bits 14 and 13] ST1 and ST0 (Sampling time):

ST1	ST0	Function
0	0	64 machine cycles (4μs at 16MHz)
0	1	Reserved
1	0	Reserved
1	1	4096 machine cycles (256μs at 16MHz)

These bits determins the duration of the voltage sampling time at the inuput.

Reading these bits always returns "00".

[bits 12 and 11] CT1 and CT0 (Compare time):

CT1	CT0	Function
0	0	176 machine cycles (22µs at 8MHz)
0	1	352 machine cycles (22µs at 16MHz)
1	0	Reserved
1	1	Reserved

These bits determins the duration of the compare operation time.

Do not set to '00' unless the machine clock is 8MHz. Otherwise the conversion accuracy is not guranteed.

Reading these bits always returns "00".

16.4 Operations of A/D Converter

The A/D converter operates employs the sequential compare technique, and has a 10-bit resolution.
Since the A/D converter has only one register (16 bits) for storing the conversion result, the conversion data registers (ADCR0 and ADCR1) are updated each time conversion is completed. Thus, the A/D converter alone must not be used for the continuous conversion. Use the F^{2}MC-16 intelligent I/O service (EI^{2}OS) function to transfer converted data to memory while conversion is in progress.
The operation modes are explained below.

■ Single Mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits. The converter stops operation after the conversion is completed for the end channel specified with the ANE bits. If the start and end channels are the same (ANS=ANE), conversion is performed only for one channel.

Example:

ANS = 0 0 0 , ANE = 0 1 1

Start -> AN0 -> AN1 -> AN2 -> AN3 -> End

ANS = 0 1 0 , ANE = 0 1 0

Start -> AN2 -> End

■ Continuous Mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and ANE bits. After the conversion is completed for the end channel specified with the ANE bits, conversion is repeated from the analog inputs of the ANS. If the start and end channels are the same (ANS=ANE), conversion for one channel is repeated.

Example:

ANS = 0 0 0 , ANE = 0 1 1

Start -> AN0 -> AN1 -> AN2 -> AN3 -> AN0 -> Repeat

ANS = 0 1 0 , ANE = 0 1 0

Start -> AN2 -> AN2 -> AN2 -> Repeat

In continuous mode, conversion is repeated until '0' is written to the BUSY bit. (Writing '0' to the BUSY bit forces the operation to end.) If the operation is terminated forcibly, conversion stops before conversion is completed. (Upon a forced stop, the conversion register stores the last data that has been converted completely.)

■ Stop Mode

In this mode, the converter sequentially converts the analog inputs specified with the ANS and

ANE bits, pausing each time conversion for one channel is completed. To release pausing, activate the converter again.

After the conversion is completed for the end channel specified with the ANE bits, conversion is repeated from the analog inputs of the ANS. If the start and end channels are the same (ANS=ANE), conversion is performed only for one channel.

Example:

ANS = 0 0 0 , ANE = 0 1 1

Start -> AN0 -> End -> Restart -> AN1 -> End -> Restarte -> AN2 -> End ->

-> Restart -> AN3 -> End -> Restart --->AN0 Repeat

ANS = 0 1 0 , ANE = 0 1 0

Start -> AN2 -> End -> Restart -> AN2 -> End -> Restarte -> AN2 Repeat

Only the activation sources specified with STS1 and STS0 are used.

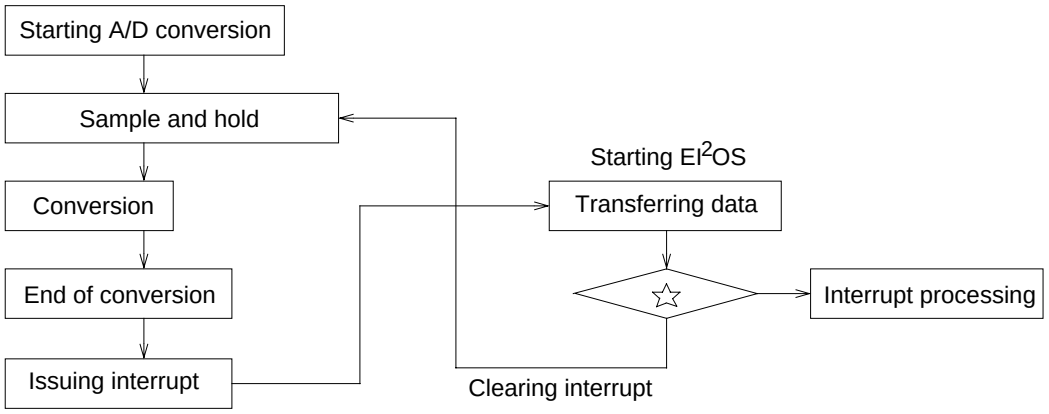
Using this mode, start of conversion can be synchronized with the activation source.

16.5 Conversion Using EI^{2}OS

Figure 16.5-1 shows the processing flow from the start of A/D conversion to the transfer of converted data (in continuous mode).

■ Conversion Using EI^{2}OS

Figure 16.5-1 A/D conversion processing flow from the start to converted data transfer (in continuous mode)



The portion indicated by the star (☆) is determined according to the EI^{2}OS setting.

16.5.1 Starting EI^{2}OS in Single Mode

Follow the steps below to start the EI^{2}OS in single mode.

- **To terminate conversion after analog inputs AN1 to AN3 are converted**
- **To transfer conversion data sequentially to addresses 200H to 206H**
- **To start conversion by software**
- **To use the highest interrupt level**

■ **Starting EI^{2}OS in Single Mode**

Settings	Sample program	Function
EI^{2}OS setting	MOV ICR3 #08H	Specifies the highest interrupt level, EI^{2}OS activation upon an interrupt, and the descriptor address.
	MOV BAPL, #00H	Specifies the transfer destination address of converted data.
	MOV BAPM, #02H	
	MOV BAPH, #00H	
	MOV ISCS, #08H	Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is terminated in response to a request from a resource.
	MOV I / OA, #38H	
	MOV DCT, #03H	EI^{2}OS transfer is performed three times. This count is the same as the conversion count.
A/D converter setting	MOV ADCS0 #0BH	Specifies single mode, start channel AN1, and end channel AN3.
	MOV ADCS1 #A2H	Specifies activation by software and start of A/D conversion.
Interrupt sequence	RET	Specifies return from an interrupt.

ICR3: Interrupt control register

BAPL: Buffer address pointer, low-order

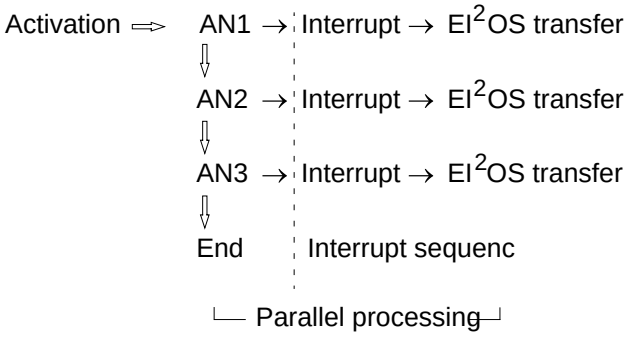
BAPM: Buffer address pointer, medium-order

BAPH: Buffer address pointer, high-order

ISCS: EI^{2}OS status register

I/OA: I/O address counter

DCT : Data counter



16.5.2 Starting EI^{2}OS in Continuous Mode

Follow the steps below to start the EI^{2}OS in continuous mode.

- **To convert analog inputs AN3 to AN5 and obtain two conversion data items for each channel**
- **To transfer conversion data sequentially to addresses 600H to 60CH**
- **To start conversion by external edge input**
- **To use the highest interrupt level**

■ Starting EI^{2}OS in Continuous Mode

Settings	Sample program	Function
EI^{2}OS setting	MOV ICR3 #08H	Specifies the highest interrupt level, EI^{2}OS activation upon an interrupt, and the descriptor address.
	MOV BAPL, #00H	Specifies the transfer destination address of converted data.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #08H	Specifies word data transfer. The transfer destination address is incremented after transfer. Data is transferred from I/O to memory. Transfer is terminated in response to a request from a resource.
	MOV I / OA, #38H	Transfer source address
	MOV DCT, #06H	EI^{2}OS transfer is performed six times. Data is transferred for three channels × 2.
A/D converter setting	MOV ADCS0 #9DH	Specifies continuous mode, start channel AN3, and end channel AN5.
	MOV ADCS1 #A4H	Specifies activation by external edge and start of A/D conversion.
Interrupt sequence	MOV ADCS1 #00H	Specifies return from an interrupt.
	RET	

ICR3 : Interrupt control register

BAPL : Buffer address pointer, low-order

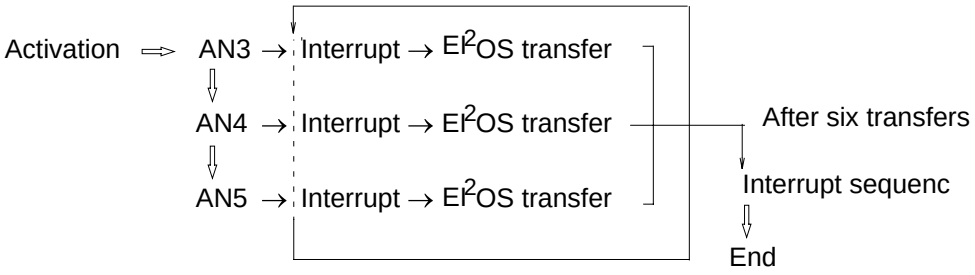
BAPM : Buffer address pointer, medium-order

BAPH : Buffer address pointer, high-order

ISCS : EI^{2}OS status register

I/OA : I/O address counter

DCT : Data counter



16.5.3 Starting EI^{2}OS in Stop Mode

Follow the steps below to start the EI^{2}OS in stop mode.

- **To convert analog input AN3 12 times at fixed intervals**
- **To transfer conversion data sequentially to addresses 600H to 618H**
- **To start conversion by external edge input**
- **To use the highest interrupt level**

■ Starting EI^{2}OS in Stop Mode

Settings	Sample program	Function
EI^{2}OS setting	MOV ICR3 #08H	Specifies the highest interrupt level, EI^{2}OS activation upon an interrupt, and the descriptor address.
	MOV BAPL, #00H	Specifies the transfer destination address of converted data.
	MOV BAPM, #06H	
	MOV BAPH, #00H	
	MOV ISCS, #08H	Specifies word data transfer. The transfer destination address is incremented after transfer.
	MOV I / OA, #38H	Data is transferred from I/O to memory. Transfer is terminated in response to a request from a resource.
	MOV DCT, #0CH	Transfer source address
A/D converter setting	MOV ADCS0 #DBH	EI^{2}OS transfer is performed 12 times.
	MOV ADCS1 #A4H	Specifies continuous mode, start channel AN3, and end channel AN3 (one-channel conversion).
Interrupt sequence	MOV ADCS1 #00H	Specifies activation by external edge and start of A/D conversion.
	RET	Specifies return from an interrupt.

ICR3 : Interrupt control register

BAPL : Buffer address pointer, low-order

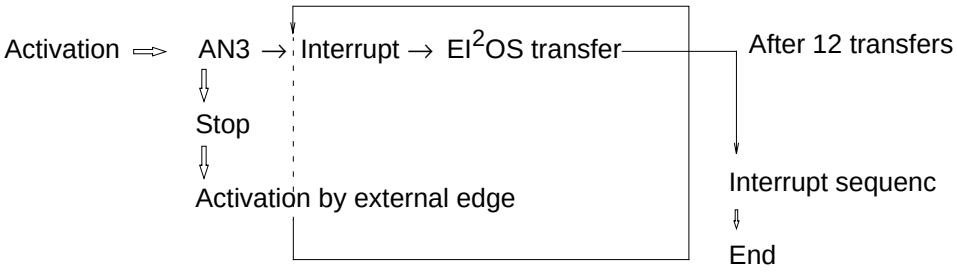
BAPM : Buffer address pointer, medium-order

BAPH : Buffer address pointer, high-order

ISCS : EI^{2}OS status register

I/OA : I/O address counter

DCT : Data counter



16.6 Conversion Data Protection

The A/D converter has a conversion data protection function that enables continuous conversion and preservation of multiple data items using EI^{2}OS.
Since there is only one conversion data register, its value is updated each time conversion is completed. Thus, continuous data conversion results in the loss of the previous data due to storage of the new data. To prevent this situation, the A/D converter pauses after conversion if the previous data item has not been transferred to memory by EI^{2}OS. The converted data is not saved until the previous data is transferred to memory.

■ Conversion Data Protection

The pause is released after data is transferred to memory by EI^{2}OS.

If the previous data has been transferred to memory, the A/D converter continues operation without pausing.

Note:

This function is related to the INT and INTE bits of ADCS1.

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

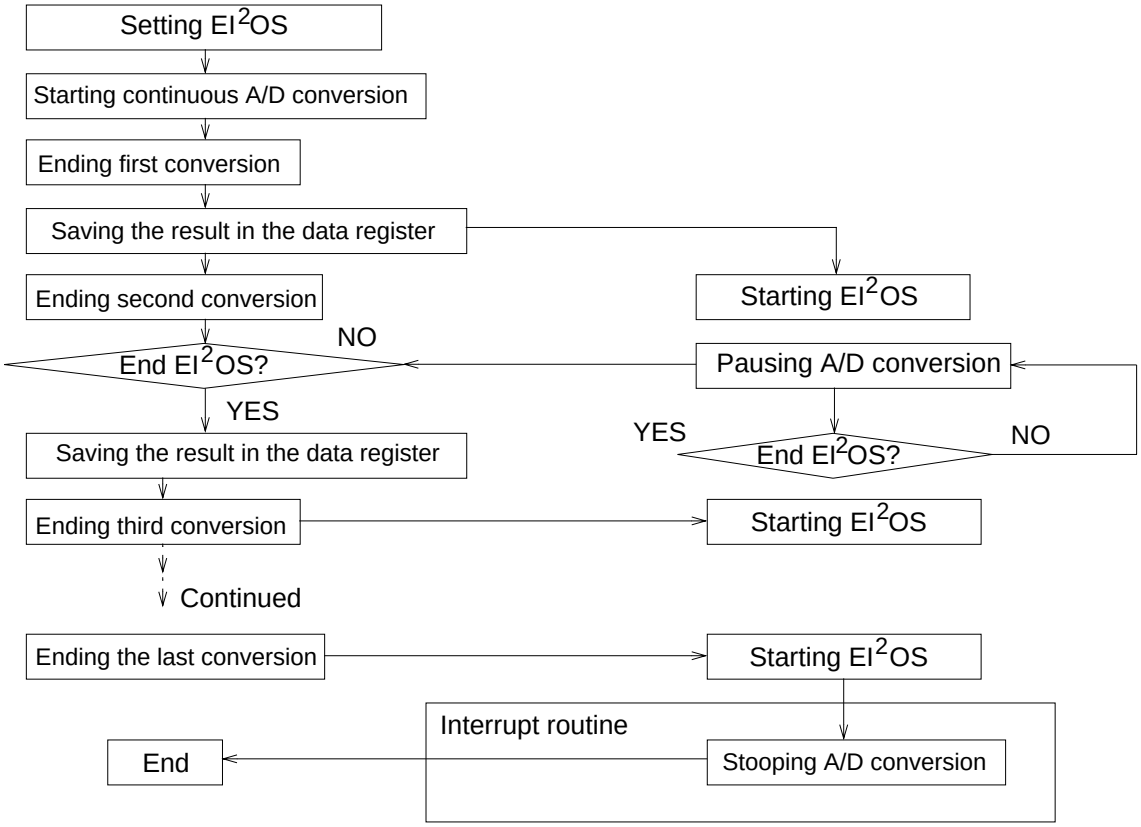
If interrupts are disabled (INTE=0), this function is disabled. Continuous A/D conversion results in loss of previous data, since the converted data items are saved to the register one after another.

If EI^{2}OS is not used while interrupts are enabled (INTE=1), the INT bit is not cleared. Thus, the data protection function works and the A/D converter pauses. In this case, clearing the INT bit in the interrupt sequence releases the pause.

If the A/D converter is pausing during EI^{2}OS operation, disabling interrupts may restart the A/D converter. In this case, the value in the conversion data register may be changed without being transferred.

Restarting the A/D converter while it is pausing destroys the standby data.

■ Flow of Data Protection Function (When EI^{2}OS is Used)



■ Notes on using the conversion data protection function

To start the A/D converter upon an external trigger or internal timer, A/D activation factor bits STS1 and STS0 of the ADCS1 register are used. Ensure that the input values of the external trigger or internal timer are inactive. If the values are active, A/D conversion may start immediately.

When setting STS1 and STS0, ensure that '1' (input) is specified for ADTG and '0' (output) is specified for the internal timer (timer 2).

CHAPTER 17 UART0

This chapter explains the UART0 functions and operations.

17.1 Feature of UART0

17.2 UART Block Diagram

17.3 UART Registers

17.4 UART0 Operation

17.5 Baud Rate

17.6 Internal and External Clock

17.7 Transfer Data Format

17.8 Parity Bit

17.9 Interrupt Generation and Flag Set Timings

17.10 UART0 Application Example

17.1 Feature of UART0

The UART is a serial I/O port for asynchronous or CLK synchronous communication. The MB90590 Series contains three UART's. The follwoing sections only describe the functionality of the UART 0. The remaining UART's have the identical function and the register addresses should be found in the I/O map.

■ Feature of UART0

UART0 has the follwoing features.

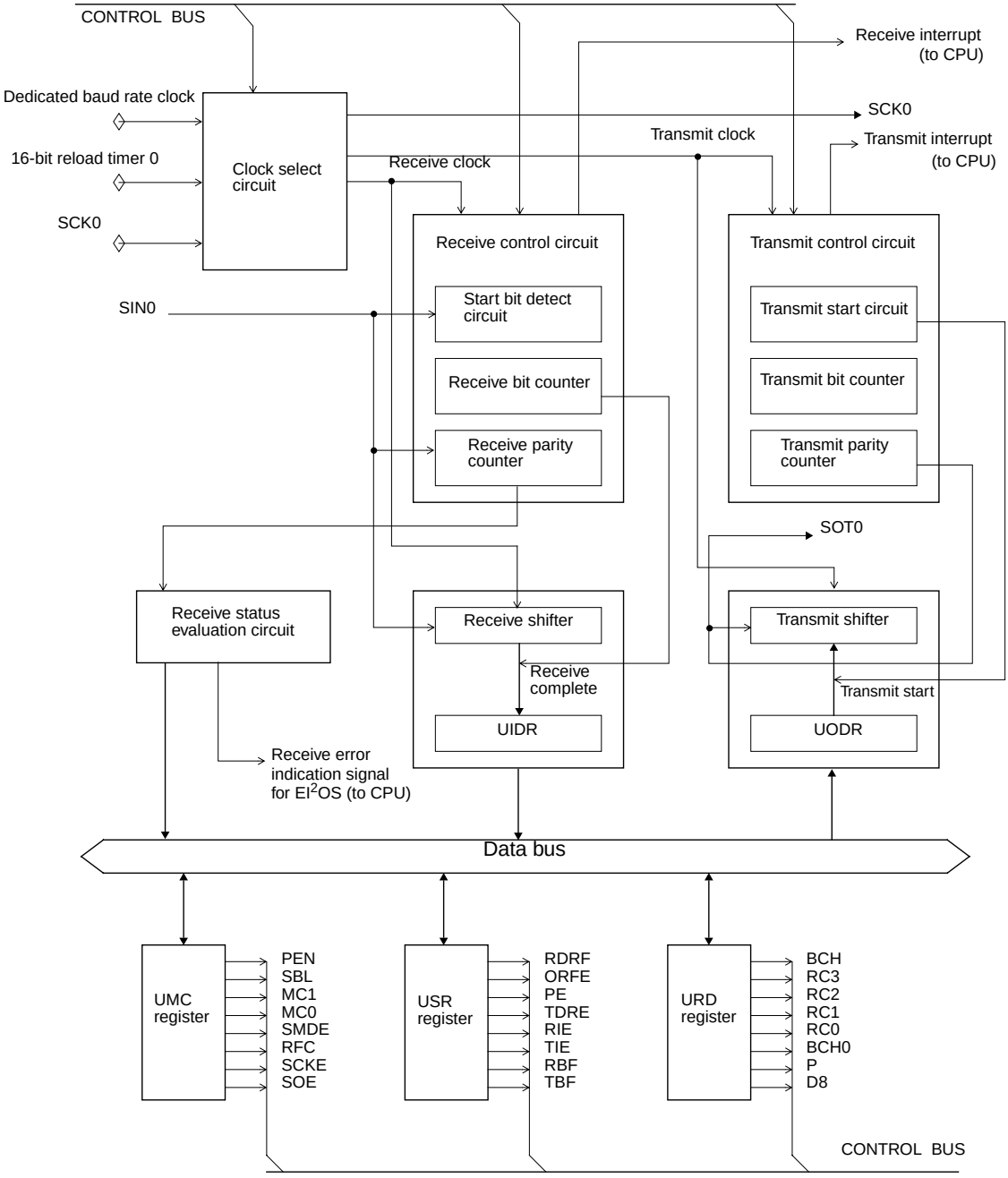
- Full duplex double buffer
- Supports CLK synchronous and CLK asynchronous start-stop data transfer.
- Multiprocessor mode support (mode 2)
- Internally dedicated baud rate generator (12 types)
- Supports flexible baud rate setting using an external clock input or internal timer.
- Variable data length (7 to 9 bits, [no parity]; 6 to 8 bits [with parity]).
- Error detect function (framing, overrun, and parity)
- Interrupt function (receive and transmit interrupts)Error detect function (framing, overrun, and parity)
- NRZ type transfer format

17.2 UART Block Diagram

Figure 17.2-1 shows a block diagram of the UART.

■ UART Block Diagram

Figure 17.2-1 UART Block Diagram

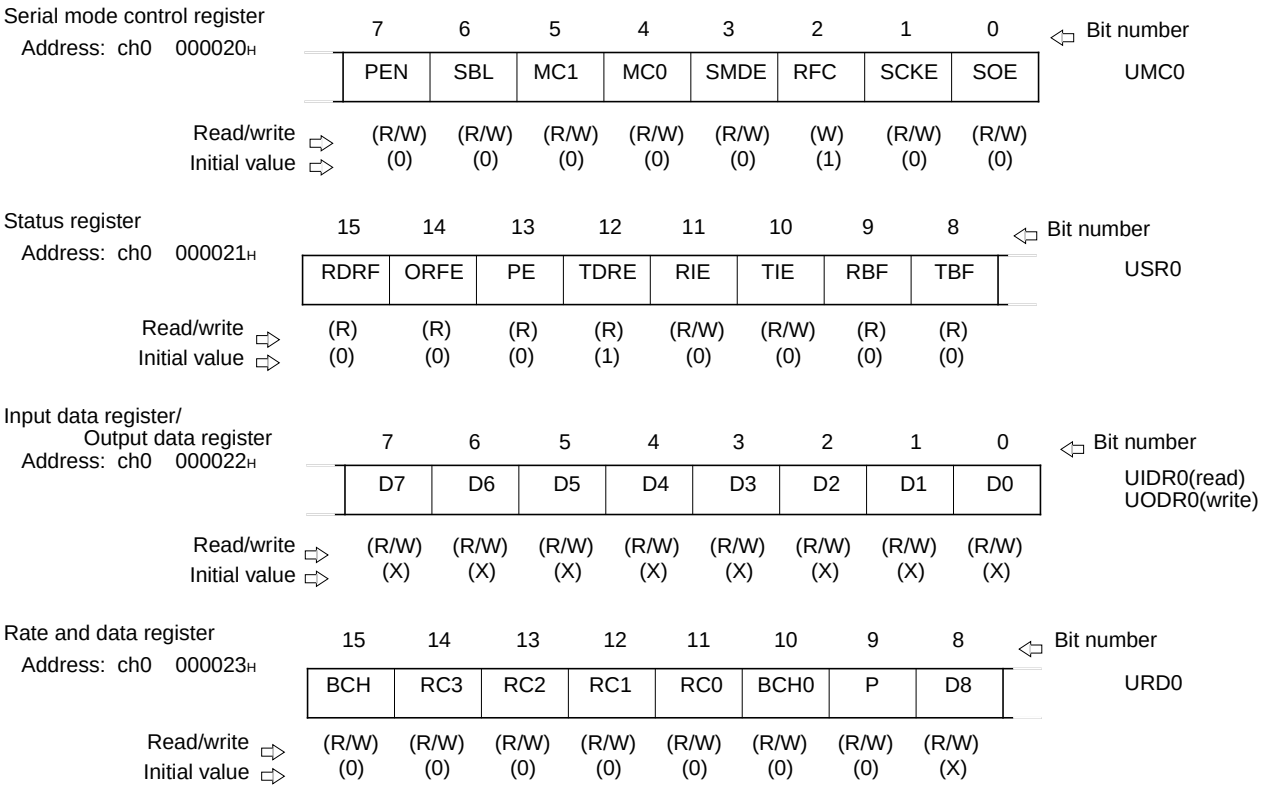


17.3 UART Registers

The UART has the following four registers:

- **Serial mode control register**
- **Status register**
- **Input data register/output data register**
- **Rate and data register**

■ UART Registers



17.3.1 Serial Mode Control Register (UMC)

UMC specifies the operation mode of UART0. Set the operation mode while operation is halted. However, the RFC bit can be accessed during operation.

■ Layout of Serial Mode Control Register (UMC)

Serial mode control register
Address: ch0 000020_H

Bit number	7	6	5	4	3	2	1	0
UMC0	PEN	SBL	MC1	MC0	SMDE	RFC	SCKE	SOE
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)

■ Serial Mode Control Register (UMC) Contents

[Bit 7] PEN (Parity enable)

Specifies whether to add (for transmit) or detect (for receive) a parity bit in serial data I/O. Set to "0" in mode 2.

0: Do not use parity

1: Use parity

[Bit 6] SBL (Stop bit length)

Specifies the number of stop bits for transmit data. For receive data, the first stop bit only is recognized and any second stop bit is ignored.

0: 1 bit length

1: 2 bits length

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

These bits control the length of the transferred data. Table 17.3-1 lists the four transfer modes (data lengths) selectable by these bits.

Table 17.3-1 UART Operation Modes

Mode	MC1	MC0	Data Length*1
0	0	0	7 (6)
1	0	1	8 (7)
2*2	1	0	8 + 1
3	1	1	9 (8)

*1: The figures enclosed in parentheses indicate the data length with parity.

*2: Mode 2 is used when a number of slave CPUs are connected to a single host CPU. As the receive parity check function cannot be used, set PEN in the UMC register to "0" (see Section 17.4 "Operation" for details). The transmit data length is 9 bits and no parity bit can be added.

[Bit 3] SMDE (Synchro mode enable)

This bit selects the transfer method.

0:Start-stop CLK synchronous transfer (clocked synchronous transfer using start and stop bits.)

1:Start-stop CLK asynchronous transfer

[Bit 2] RFC (Receiver flag clear)

Writing "0" to this bit clears the RDRF, ORFE, and PE flags in the USR register. Writing "1" has no effect. Reading always returns "1".

Note:

When receive interrupts are enabled during UART0 operation, only write "0" to RFC when either RDRF, ORFE, or PE is "1".

[Bit 1] SCKE (SCLK enable)

Writing "1" to this bit in CLK synchronous mode switches the port pin to the UART0 serial clock output pin and outputs the synchronizing clock. Set to 0 in CLK asynchronous mode or external clock mode.

0: The pin functions as a general purpose I/O port and does not output the serial clock. The pin functions as the external clock input pin when the port is set to input mode (DDR=0) and RC3 to 0 are set to "1111".

1: The pin functions as the UART0 serial clock output pin.

Note:

The corresponding bit of the Port Direction register should be set to "1" when the port pin is used as the clock output. This is for UART0 only.

[Bit 0] SOE (Serial output enable)

Writing "1" to this bit switches the port pin to the UART0 serial data output pin and enables serial output.

0: The pin functions as a port pin and does not output serial data.

1: The pin functions as the UART0 serial data output pin (SOT).

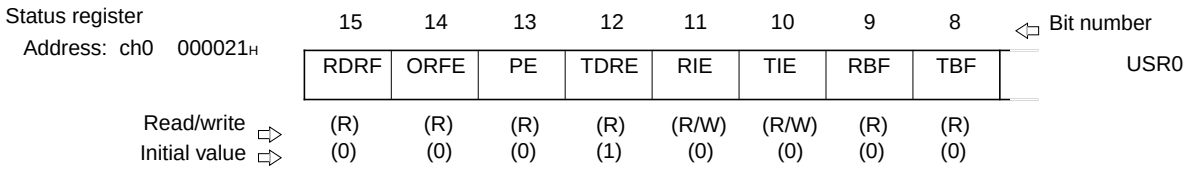
Note:

The corresponding bit of the Port Direction register should be set to "1" when the port pin is used as the serial output. This is for UART0 only.

17.3.2 Status Register (USR)

USR indicates the current state of the UART0 port.

■ Status Register (USR) Layout



■ Status Register (USR) Contents

[Bit 15] RDRF (Receiver data register full)

This flag indicates the state of the UIDR0 (input data register). The flag is set when the receive data is loaded into UIDR0. Reading UIDR0 or writing "0" to RFC in the UMC0 register clears the flag. If RIE is active, a receive interrupt request is generated when RDRF is set.

0: No data in UIDR0

1: Data present in UIDR0

[Bit 14] ORFE (Over-run/framing error)

The flag is set when an overrun or framing error occurs in receiving. Writing "0" to RFC in the UMC0 register clears the flag. When this flag is set, the data in UIDR0 is invalid and the load from the receive shifter to UIDR0 is not performed. If RIE is active, a receive interrupt request is generated when ORFE is set.

0: No error

1: Error

Table 17.3-2lists the UIDR0 states after receive completion by RDRF or ORFE.

Table 17.3-2 UIDR State after Receive Completion

RDRF	ORFE	UIDR0 Data State
0	0	Empty
0	1	Framing error
1	0	Valid data
1	1	Overrun error

The data in UIDR is invalid if an overrun or framing error has occurred. Next data can be received after clearing the flag(s).

[Bit 13] PE (Parity error)

The flag is set when a receive parity error occurs. Writing "0" to RFC in the UMC register clears the flag. When this flag is set, the data in UIDR0 is invalid and the load from the receive shifter to UIDR0 is not performed. If RIE is active, a receive interrupt request is generated when PE is set.

0: No parity error

1: Parity error

[Bit 12] TDRE (Transmitter data register empty)

This flag indicates the state of the UODR0 (output data register). Writing transmit data to the UODR0 register clears the flag. The flag is set when the data is loaded to the transmit shifter and the transmission is started. If TIE is active, a transmit interrupt request is generated when TDRE is set.

0: Data present in UODR0

1: No data in UODR0

[Bit 11] RIE (Receiver interrupt enable)

Enables receive interrupt requests.

0: Disable interrupts.

1: Enable interrupts.

[Bit 10] TIE (Transmitter interrupt enable)

Enables transmit interrupt requests. A transmit interrupt is generated immediately if transmit interrupts are enabled when TDRE is "1".

0: Disable interrupts.

1: Enable interrupts.

[Bit 9] RBF (Receiver busy flag)

This flag indicates that UART0 is receiving input data. The flag is set when the start bit is detected and cleared when the stop bit is detected.

0: Receiver idle

1: Receiver busy

[Bit 8] TBF (Transmitter busy flag)

This flag indicates that UART0 is transmitting input data. The flag is set when transmit data is written to the UODR0 register and cleared when transmission completes.

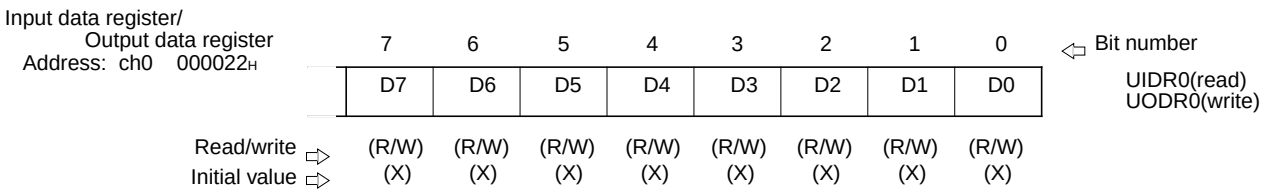
0: Transmitter idle

1: Transmitter busy

17.3.3 Input Data Register (UIDR) and Output Data Register (UODR)

UIDR (input data register) is the serial data input register. UODR (output data register) is the serial data output register.
The most significant two bits (D7 and D6) are ignored if the data length is 6 bits and the most significant bit (D7) is ignored if the data length is 7 bits. Write to UODR only when TDRE = "1" in the USR register. Read UIDR only when RDRF = "1" in the USR register.

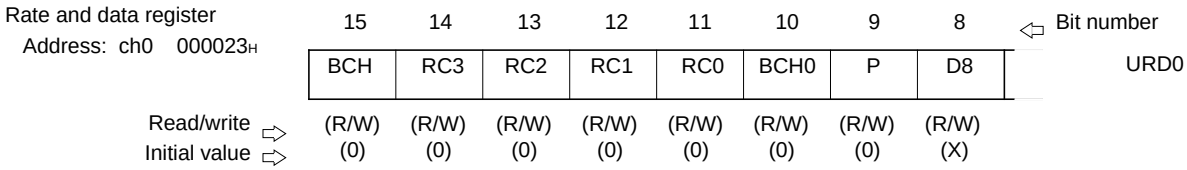
■ Input Data Register (UIDR) and Output Data Register (UODR)



17.3.4 Rate and Data Register (URD)

URD selects the data transfer speed (baud rate) for UART0. The register also holds the most significant bit (bit 8) of the data when the transmit data length is 9 bits. Set the baud rate and parity when UART0 is halted.

■ Layout of Rate and Data Register (URD)



■ Rate and Data Register (URD) Contents

[Bits 15, 10] BCH, BCH0 (Baud rate clock change)

Specifies the machine cycles for the baud rate clock (see Section 17.4 "Operation" for details).

Table 17.3-3 Clock Input Selection

BCH	BCH0	Divider ratio	Setting Example for Each Machine Cycle
0	0	-	- Prohibited setting -
0	1	Divide by 4	For a 16-MHz machine cycle: 16/4 = 4 MHz
1	0	Divide by 3	For a 12-MHz machine cycle: 12/3 = 4 MHz
1	1	Divide by 5	For a 10-MHz machine cycle: 10/5 = 2 MHz

Note:

Do not set BCH and BCH0 to "00".

[Bits 14 to 11] RC3, RC2, RC1, RC0 (Rate control)

Selects the clock input for the UART0 port (see Section 17.4 "Operation" for details).

Table 17.3-4 Clock Input Selection

RC3 to RC0	Clock Input
"0000" to "1011"	Dedicated baud rate generator
"1101"	16-bit Reload Timer 0
"1111"	External clock

Note:

Do not set the rate control bits to "1100" "1110".

[Bit 9] P

Sets even or odd parity when parity is active (PEN = "1").

0: Even parity

1: Odd parity

[Bit 8] D8

Holds the bit 8 of the transfer data in mode 2 or 3 (9-bit data length) and no parity. Treated as bit 8 of the UIDR0 register for reading. Treated as bit 8 of the UODR register for writing. The bit has no meaning in the other modes. Write to D8 only when TDRE = "1" in the USR0 register.

17.4 UART0 Operation

Table 17.4-1 lists the operating modes for UART0. Set the UMC register to switch between modes.

■ UART0 Operation Modes

Table 17.4-1 UART0 Operating Modes

Mode	Parity	Data Length	Clock Mode	Length of Stop Bits*
0	On	6	CLK asynchronous or CLK synchronous	1 bit or 2 bits
	Off	7		
1	On	7		
	Off	8		
2	Off	8+1		
3	On	8		
	Off	9		

*: The number of stop bits can only be set for transmission. The number of receive stop bits is always set to one. Do not set modes other than those listed above. UART0 does not operate if an invalid mode is set.

Note:

UART0 uses start-stop clock synchronous transfer. Therefore, a start and stop bit are added to the data even in clock synchronous transfer.

17.5 Baud Rate

When the dedicated baud rate generator is used, the following two types of baud rates are available:

- **CLK synchronous baud rate**
- **CLK asynchronous baud rate**

■ CLK Synchronous Baud Rate

The five URD register bits: BCH, BCH0 and RC3, RC2, RC1 select the baud rate for CLK synchronous transfer.

First select the machine clock divider ratio using BCH and BCH0.

BCH	BCH0		
0	1	=>	Divide by 4 [For example, at 16 MHz: 16/4 = 4 MHz]
1	0	=>	Divide by 3 [For example, at 12 MHz: 12/3 = 4 MHz]
1	1	=>	Divide by 5 [For example, at 10 MHz: 10/5 = 2 MHz]

Then, set the division ratio for the clock selected above in RC3, RC2, and RC1. The following three settings are available for CLK synchronous transfer. Other settings are prohibited.

RC3	RC2	RC1		
0	1	0	=>	Divide by 2 [For example, at 4 MHz: 4/2 = 2.0 M (bps)]
0	1	1	=>	Divide by 4 [For example, at 4 MHz: 4/4 = 1.0 M (bps)]
1	0	0	=>	Divide by 8 [For example, at 4 MHz: 4/8 = 0.5 M (bps)]
				(At 2 MHz, the speed becomes half the above examples.)

■ CLK Asynchronous Baud Rate

The six URD register bits: BCH, BCH0 and RC3, RC2, RC1, RC0 select the baud rate for CLK asynchronous transfer.

First select the machine clock divider ratio using BCH and BCH0.

BCH	BCH0		
0	1	=>	Divide by 4 [For example, at 16 MHz: 16/4 = 4 MHz]
1	0	=>	Divide by 3 [For example, at 12 MHz: 12/3 = 4 MHz]
1	1	=>	Divide by 5 [For example, at 10 MHz: 10/5 = 2 MHz]

Then, set the asynchronous transfer clock division ratio for the clock selected above in RC3, RC2, RC1, and RC0. The following settings are available.

RC3	RC2	RC1			RC0	
0	0	0	⇒ Divide by 8 × 1	×	0	⇒ Divide by 12
0	1	0	⇒ Divide by 8 × 2		1	⇒ Divide by 13
0	1	1	⇒ Divide by 8 × 4			
1	0	0	⇒ Divide by 8 × 8			
0	0	1	⇒ Not divided	×	0	⇒ Prohibited setting
1	0	1	⇒ Divide by 8		1	⇒ Divide by 8

The above 12 baud rates can be selected. The following formula shows how to calculate the CLK synchronous baud rate.

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

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

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

where ϕ is a machine cycle and m is in decimal notation for RC3 to 1.

Note:

The above formula for m=0 or m=1 cannot be calculated.

Data transfer is possible if the CLK asynchronous baud rate is in the range -1% to +1%. The baud rate is the CLK synchronous baud rate divided by 8 × 13, 8 × 12, or 8.

Table 17.5-1 shows examples for 16 MHz, 12 MHz, and 10 MHz machine cycles. However, do not use the settings marked as '_' in the table.

Table 17.5-1 Baud Rate

				CLK asynchronous (μs/Baud)			CLK asynchronous divider ratio	CLK synchronous (μs/Baud)		
				16 MHz	12 MHz	10 MHz		16 MHz	12 MHz	10 MHz
RC3	RC2	RC1	RC0	BCH/ 0=01	BCH/ 0=10	BCH/ 0=11		BCH/ 0=01	BCH/ 0=10	BCH/ 0=11
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	0.5 / 2M	0.5 / 2M	1 / 1M
0	1	1	0	96/10417	96/10417	192/ 5208	8 × 12	-	-	-

Table 17.5-1 Baud Rate (Continued)

RC3	RC2	RC1	RC0	CLK asynchronous (µs/Baud)			CLK asynchronous divider ratio	CLK synchronous (µs/Baud)		
				16 MHz	12 MHz	10 MHz		16 MHz	12 MHz	10 MHz
				BCH/ 0=01	BCH/ 0=10	BCH/ 0=11		BCH/ 0=01	BCH/ 0=10	BCH/ 0=11
0	1	1	1	104/ 9615	104/ 9615	208/ 4808	8 × 13	1 / 1M	1 / 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 / 250K
1	0	1	0	-	-	-	8			
1	0	1	1	16/ 62500	16/ 62500	32/ 31250	8	-	-	-

17.6 Internal and External Clock

Setting RC3 to 0 to "1101" selects the clock signal from the 16-bit Reload Timer.
Setting RC3 to 0 to "1111" selects the external clock.

■ Internal and External Clock

The CLK asynchronous baud rate is the CLK synchronous baud rate divided by 8. Also, data transfer is possible if the CLK asynchronous baud rate is in the range -1% to +1% of the selected baud rate. Table 17.6-1 lists the baud rates when the internal timer is selected as the clock. The values in this table are calculated for a machine cycle of 7.3728 MHz. However, do not use the settings marked as '_' in the table.

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

- ϕ: Machine cycle
- X: Divider ratio for the count clock source for the internal timer
- n: Reload value (decimal)

Table 17.6-1 Baud Rate and Reload Value

Baud Rate	Reload Value	
	X = 2^1 (divide machine cycle by 2)	**X = 2^3 (divide machine cycle 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

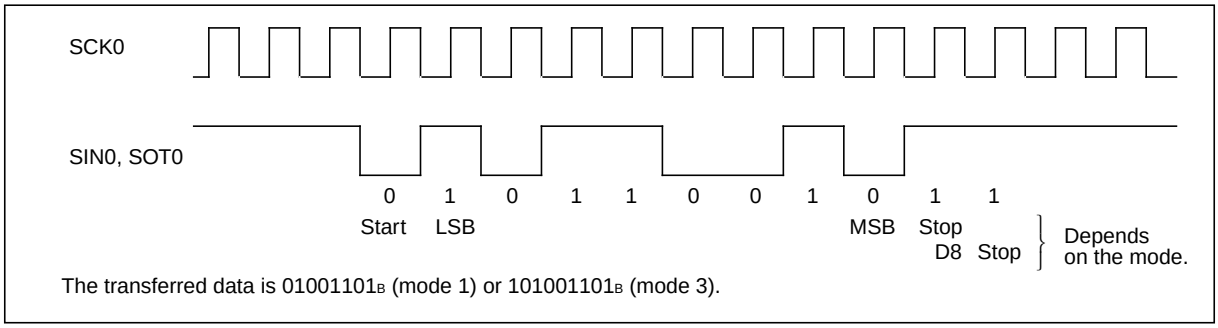
The values in the table are the reload values (decimal) for reload count operation of the 16-bit Reload Timer.

17.7 Transfer Data Format

UART0 only handles NRZ (non-return-to-zero) type data. Figure 17.7-1 shows the relationship between the transmit/receive clock and the data for CLK synchronous mode.

■ Transfer Data Format

Figure 17.7-1 Transfer Data Format



As shown in Figure 17.7-1 "Transfer Data Format", the transfer data always starts with the start bit (L level data), the specified number of data bits are transmitted with the LSB first, then transmission ends with the stop bit ('H' level data). Always input a clock if external clock operation is selected. When an internal clock (the dedicated baud rate generator or 16-bit Reload Timer) is selected, the clock is output continuously. When using CLK synchronous transfer, do not start data transfer until the selected baud rate clock has stabilized (for two baud rate clock cycles).

When using CLK asynchronous transfer, set the SCKE bit in the UMC0 register to "0" to disable clock output. The transfer data format of SIN0 and SOUT0 is the same as shown in Figure 17.7-1 "Transfer Data Format".

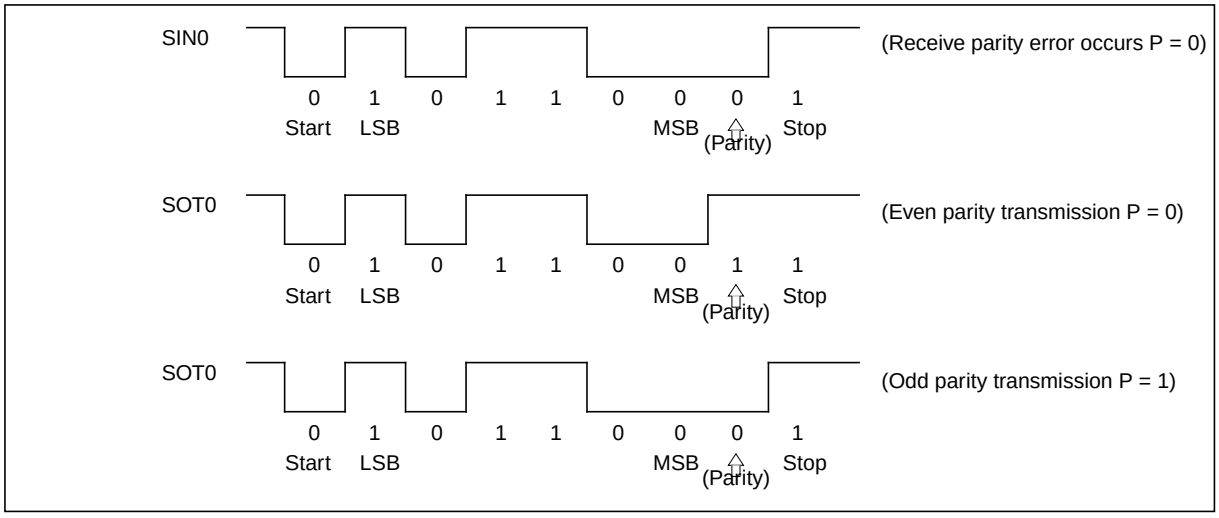
17.8 Parity Bit

The P bit in the URD0 register specifies whether to use even or odd parity when parity is enabled. The PEN bit in the UMC0 register enables parity.

■ Parity Bit

Inputting the data shown in Figure 17.8-1 to SIN when even parity is set causes a receive parity error. Figure 17.8-1 also shows the data transmitted when sending 001101_B with even parity and odd parity.

Figure 17.8-1 Serial Data with Parity Enabled



17.9 Interrupt Generation and Flag Set Timings

UART0 has two interrupt causes and six flags. The two interrupt causes are the receive and transmit interrupts. The six flags are RDRF, ORFE, PE, TDRE, RBF, and TBF. For reception, the RDRF, ORFE, and PE flags request an interrupt. For transmission, the TDRE flag requests an interrupt.

■ Set Timings of the Six Flags

❍ RDRF flag

The RDRF flag is set when receive data is loaded into the UIDR register. The flag is cleared by writing "0" to RFC in the UMC register or by reading the UIDR0 register.

❍ ORFE flag

The ORFE flag is an overrun or framing error flag. The flag is set when a receive error occurs and is cleared by writing "0" to RFC in the UMC0 register.

❍ PE flag

The PE flag is a reception parity error flag. The flag is set when a receive parity error occurs and is cleared by writing "0" to RFC in the UMC0 register. Note that the parity detect function is not available in mode 2.

❍ TDRE flag

The TDRE flag is set when the UODR0 register becomes empty and is available for writing. The flag is cleared by writing to the UODR0 register. The above four flags (RDRF, ORFE, PE, and TDRE) trigger transmit or receive interrupts.

❍ RBF and TBF flags

The RBF and TBF flags indicate that reception or transmission is in progress. The RBF flag becomes active during reception, and the TBF flag becomes active during transmission.

17.9.1 Flag Set Timings for a Receive Operation (in Mode 0, 1, or 3)

The RDRF, ORFE, and PE flags are set and an interrupt request to the CPU generated when the final stop bit is detected indicating the end of reception transfer. The data in UIDR0 is invalid when either the ORFE or PE bit is active.

■ **Flag set Timings for a Receive Operation (in Mode 0, 1, or 3)**

Figures 17.9-1, 17.9-2, and 17.9-3 show the set timings of the RDRF, ORFE, and PE flags respectively.

Figure 17.9-1 RDRF Set Timing (Mode 0, 1, or 3)

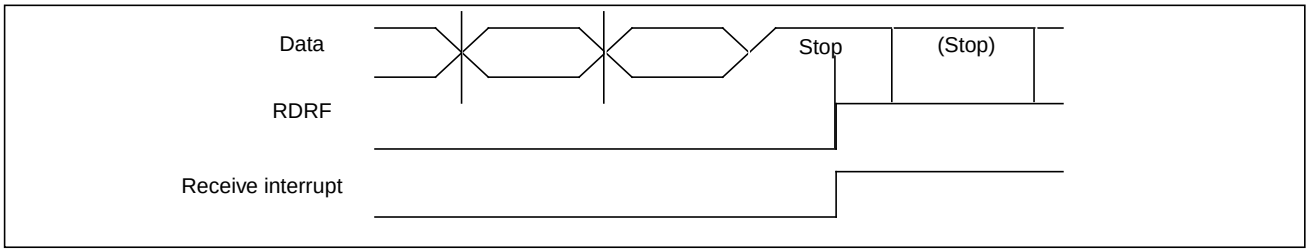


Figure 17.9-2 ORFE Set Timing (Mode 0, 1, or 3)

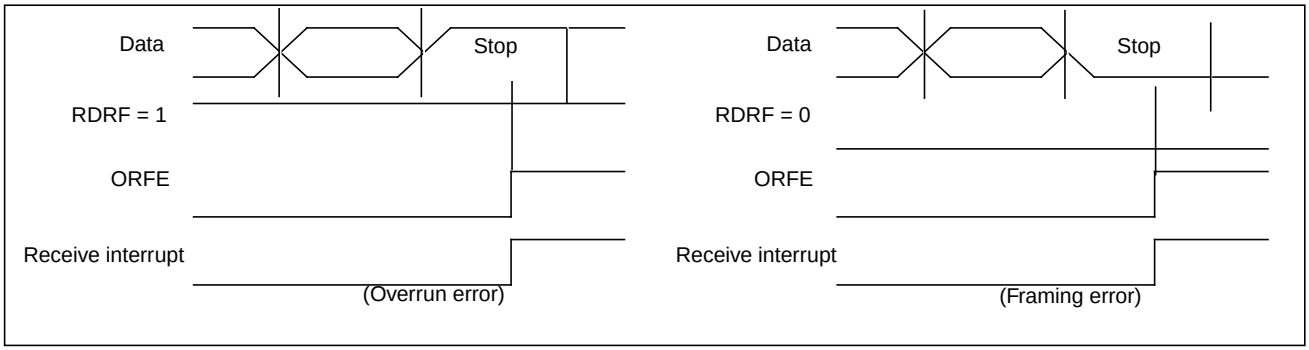
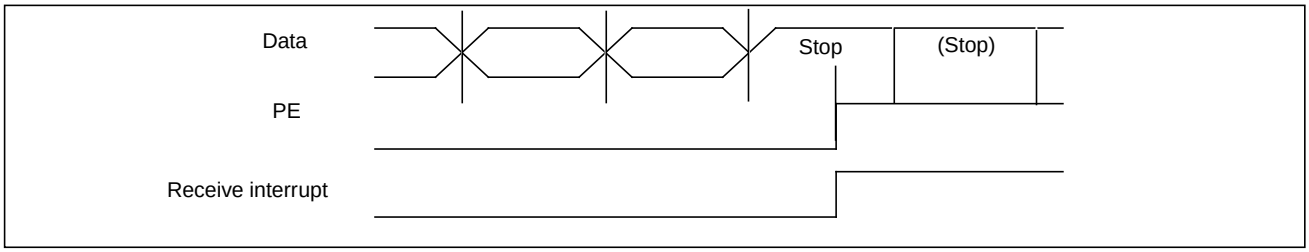


Figure 17.9-3 PE Set Timing (Mode 0, 1, or 3)



17.9.2 Flag set timings for a receive operation (in mode 2)

The RDRF flag is set when the final stop bit is detected and reception transfer ends with the last data bit (D8) having the value "1".
The ORFE flag is set when the final stop bit is detected, irrespective of the value of the last data bit (D8). The data in UIDR0 is invalid when the ORFE bit is active.
The interrupt request to the CPU is generated when either of the flags are set (see Section 17.10 "Application Example" for details on using mode 2).

■ **Flag Set Timings for a Receive Operation (in Mode 2)**

Figure 17.9-4 RDRF Set Timing (Mode 2)

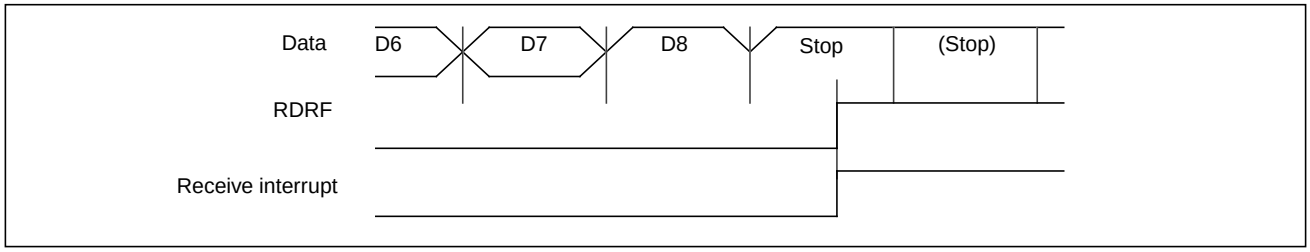
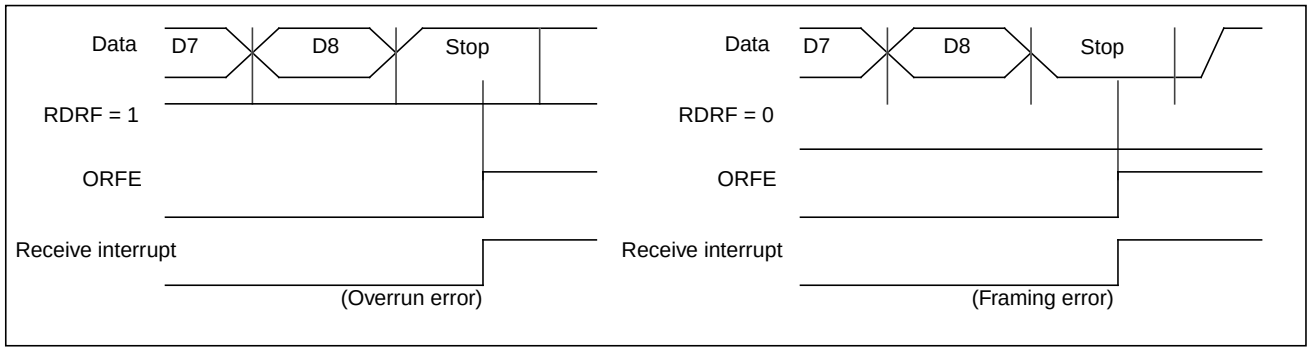


Figure 17.9-5 ORFE Set Timing (Mode 2)

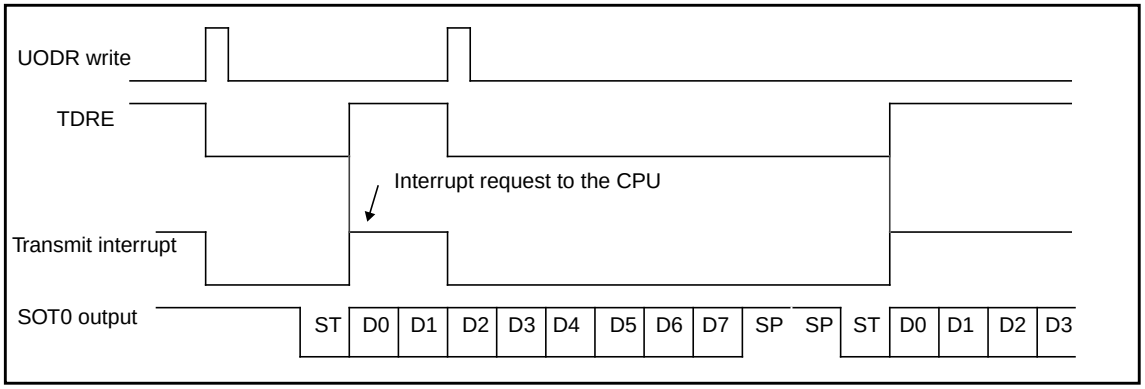


17.9.3 Flag set Timings for a Transmit Operation

TDRE is set and an interrupt request to the CPU is generated when the data written in UODR0 register is transferred to the internal shift register and the next data can be written to UODR0.

■ **Flag Set Timings for a Transmit Operation**

Figure 17.9-6 TDRE Set Timing (Mode 0)



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

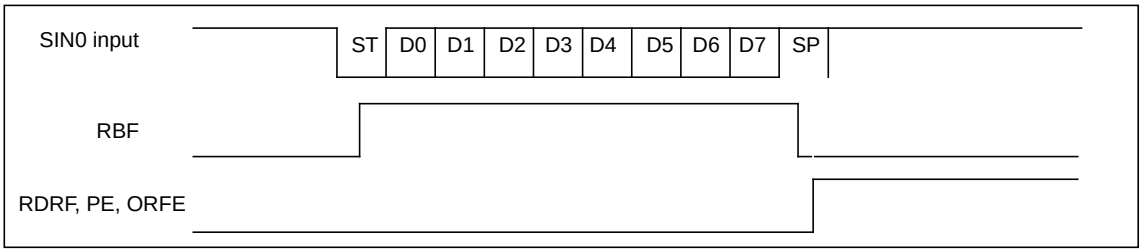
17.9.4 Status Flag During Transmit and Receive Operation

RBF is set when the start bit is detected and cleared when a stop bit is detected. The receive data in UIDR0 at the RBF clear timing is not yet valid. The data in UIDR0 becomes valid at the RDRF set timing.

■ Status Flag during Transmit and Receive Operation

Figure 17.9-7 shows the relationship between the RBF and receive interrupt flag timing.

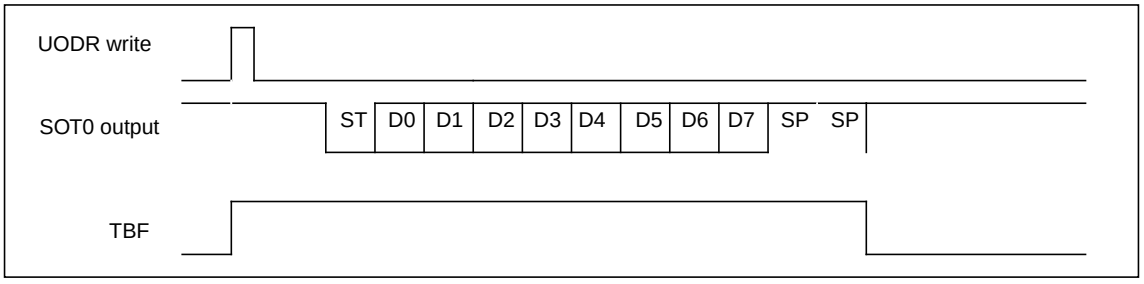
Figure 17.9-7 RBF Set Timing (Mode 0)



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

Writing the transmission data to UODR0 sets TBF. TBF is cleared when transmission completes.

Figure 17.9-8 TBF Set Timing (Mode 0)



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

Note:

Receive operation starts after releasing a reset unless the SIN input pin is fixed at "1". Therefore, before setting the mode, write "0" to RFC in the UMC0 register to clear any receive flags that have been set.

Set the communication mode when the RBF and TBF flags in the USR0 register are "0". The data transmitted and received during mode setting cannot be guaranteed.

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

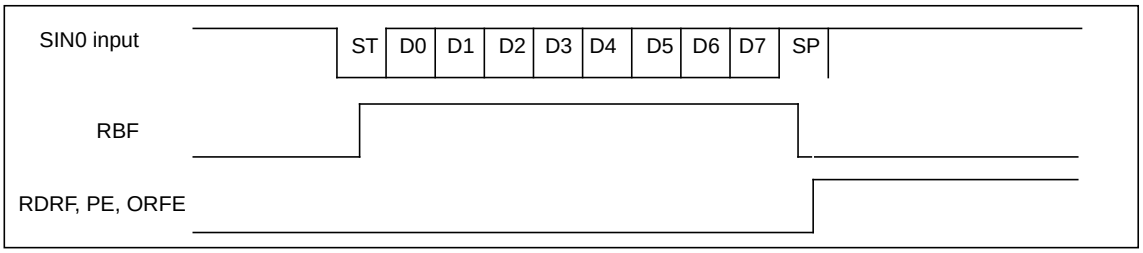
See the Section 3.7 "Extended intelligent I/O service (EI^{2}OS)" for details on EI^{2}OS.

17.10 UART0 Application Example

Mode 2 is used when a number of slave CPUs are connected to a host CPU (see Figure 17.10-1.)

■ Application Example

Figure 17.10-1 RBF Set Timing (mode 0)



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

As shown in Figure 17.10-2, communication starts with the host CPU transmitting address data. The ninth bit (D8) of the address data is set to "1". The address selects the slave CPU with which communication will be established. The selected slave CPU communicates with the host CPU using a protocol determined by the user. In normal data, D8 is set to "0". Unselected slave CPUs wait in standby until the next communication session starts. Figure 17.10-3 shows a flowchart of operation in this mode.

Because the parity check function is not available in this mode, set the PEN bit in the UMC0 register to "0".

Figure 17.10-2 Example System Configuration Using Mode 2

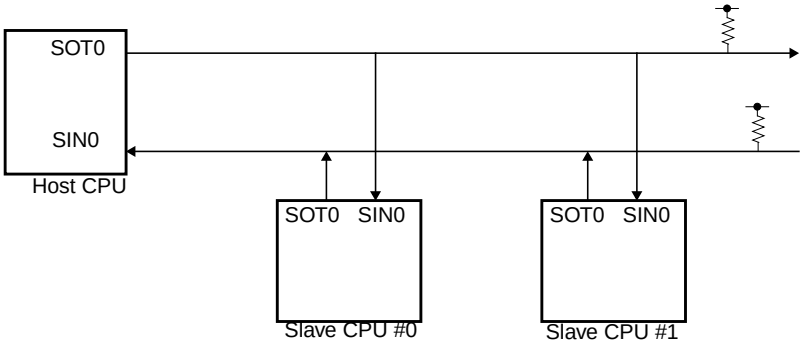
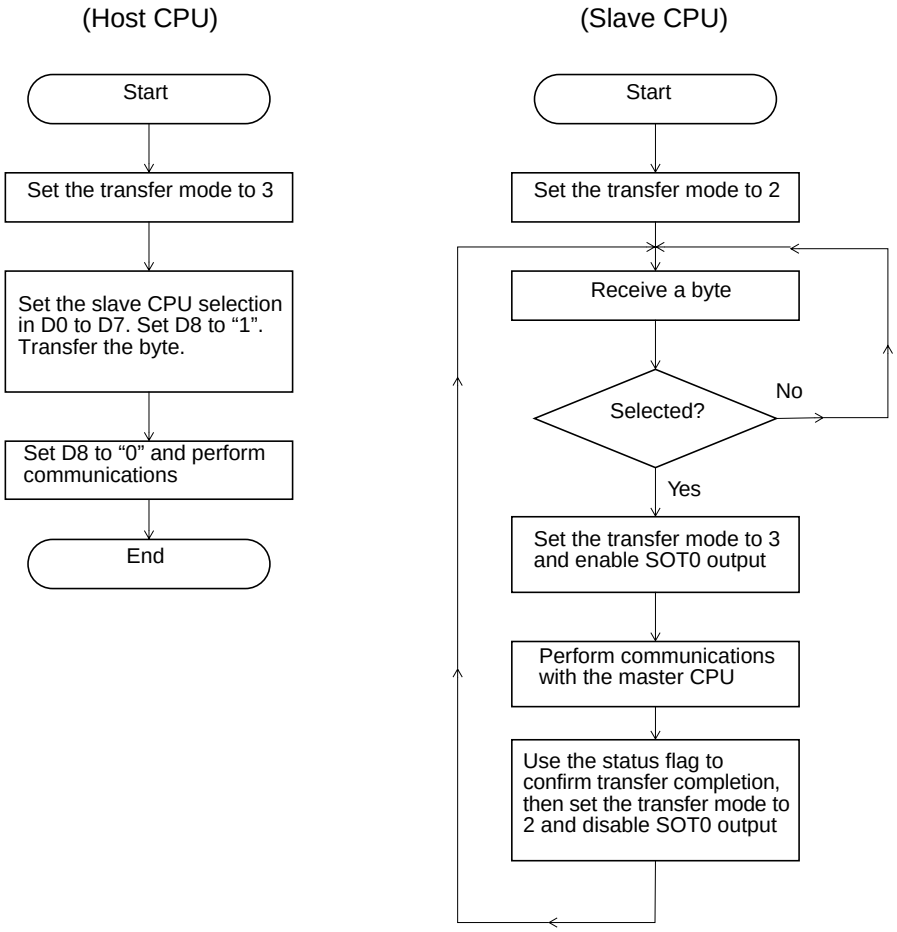


Figure 17.10-3 Communication Flowchart for Mode 2 Operation



CHAPTER 18 SERIAL I/O

This chapter explains the functions and operations of the serial I/O.

18.1 Outline of Serial I/O

18.2 Serial I/O Registers

18.3 Serial I/O Prescaler (CDCR)

18.4 Serial I/O Operation

18.5 Negative Clock Operation

18.1 Outline of Serial I/O

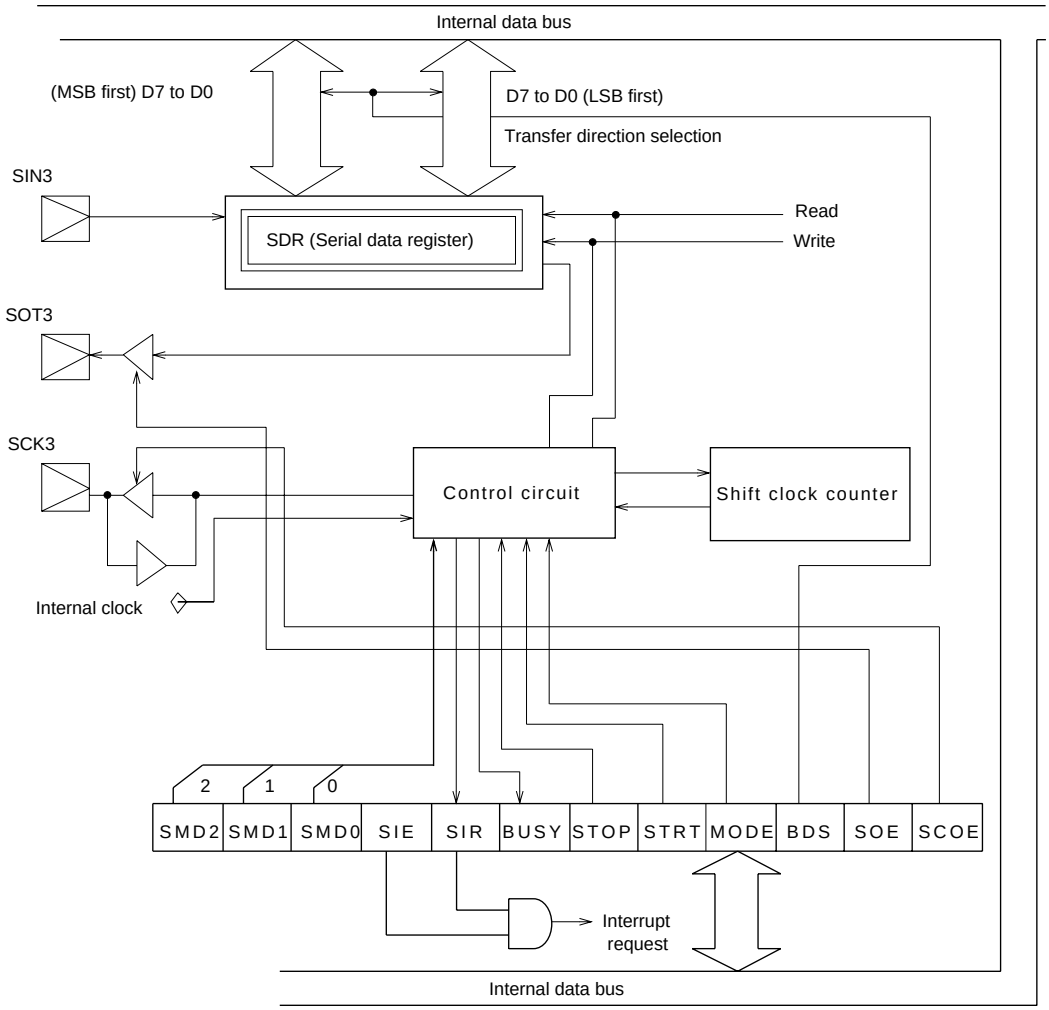
The serial I/O interface operates in two modes:

- **Internal shift clock mode: Data is transferred in synchronization with the internal clock.**
- **External shift clock mode: Data is transferred in synchronization with the clock supplied via the external pin (SCK3). By manipulating the general-purpose port sharing the external pin (SCK3), data can also be transferred by a CPU instruction in this mode.**

■ Serial I/O Block Diagram

This block is a serial I/O interface that allows data transfer using clock synchronization. The interface consists of a single eight-bit channel. Data can be transferred from the LSB or MSB.

Figure 18.1-1 Extended Serial I/O Interface Block Diagram



18.2 Serial I/O Registers

The serial I/O has the following two registers:

- **Serial mode control status register (SMCS)**
- **Serial data register (SDR)**

■ **Serial I/O Resisters**

	15	14	13	12	11	10	9	8	
Address : 00002D$_H$	SMD2	SMD1	SMD0	SIE	SIR	BUSY	STOP	STRT	Serial mode control status register (SMCS)

	7	6	5	4	3	2	1	0
Address : 00002C$_H$	—	—	—	—	MODE	BDS	SOE	SCOE

	7	6	5	4	3	2	1	0	
Address : 00002E$_H$	D7	D6	D5	D4	D3	D2	D1	D0	Serial data register (SDR)

18.2.1 Serial Mode Control Status Register (SMCS)

The serial mode control status register (SMCS) controls the serial I/O transfer mode.

■ **Serial Mode Control Status Register (SMCS)**

	15	14	13	12	11	10	9	8	
SMCS Address: 00002D$_H$	SMD2	SMD1	SMD0	SIE	SIR	BUSY	STOP	STRT	Initial value 00000010$_B$
	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	
					*1			*2	

	7	6	5	4	3	2	1	0	
SMCS Address: 00002C$_H$	—	—	—	—	MODE	BDS	SOE	SCOE	Initial value ----0000$_B$
					R/W	R/W	R/W	R/W	

*1: Only '0' can be written.
*2: Only '1' can be written. '0' is always read.

■ **Bit functions of Serial Mode Control Status Register (SMCS)**

[bit 3] Serial mode selection bit (MODE)

The serial mode selection bit is used to select the conditions to start the transfer operation from the stop state. This bit must not be updated during operation.

Table 18.2-1 Setting the Serial Mode Selection Bit

MODE	Operation
0	Transfer starts when STRT=1. [Default]
1	Transfer starts when the serial data register is read or written to.

This bit is initialized to a '0' upon a reset, and can be read or written to. To activate the intelligent I/O service, ensure that '1' is written to this bit.

[bit 2] Bit direction select bit (BDS)

When serial data is input or output, this bit determines from which bit data is to be transferred first, the least significant bit (LSB first) or the most significant bit (MSB first), as shown in Table 18.2-2.

Table 18.2-2 Setting the Transfer Direction Selection Bit

0	LSB first [default]
1	MSB first

Note:

Specify the bit ordering before any data is written to SDR.

[bit 1] Serial output enable bit (SOE: Serial out enable)

This bit controls the output from the serial I/O output external pins (SOT1 and 2) as shown in Table 18.2-3.

Table 18.2-3 Setting the Serial Output Enable Bit

0	General-purpose port pin [default]
1	Serial data output

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 0] Shift clock output enable bit (SCOE: SCK1 output enable)

This bit controls the output from the shift clock I/O output external pins (SCK1 and 2) as shown in Table 18.2-4.

Table 18.2-4 Setting the Shift Clock Output Enable Bit

0	General-purpose port pin, transfer for each instruction [default]
1	Shift clock output pin

Ensure that '0' is written to this bit when data is transferred for each instruction in external shift clock mode.

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[Bits 15, 14, and 13] Shift clock selection bits (SMD2, SMD1, SMD0: Serial shift clock mode)

These bits are used to select the serial shift clock mode, as shown in Table 18.2-5.

Table 18.2-5 Setting the Serial Shift Clock Mode

SMD2	SMD1	SMD0	ϕ=16MHz div=8	ϕ=8MHz div=4	ϕ=4MHz div=4
0	0	0	1 MHz	1 MHz	500 kHz
0	0	1	500 kHz	500 kHz	250 kHz
0	1	0	125 kHz	125 kHz	62.5 kHz
0	1	1	62.5 kHz	62.5 kHz	31.25 kHz
1	0	0	31.25 kHz	31.25 kHz	5.625 kHz
1	0	1	External shift clock mode		
1	1	0	Reserved		
1	1	1	Reserved		

div	M1	DIV3	DIV2	DIV1	DIV0	Recommended machine cycle
3	1	1	1	0	1	6 MHz
4	1	1	1	0	0	8 MHz
5	1	1	0	1	1	10 MHz

div	M1	DIV3	DIV2	DIV1	DIV0	Recommended machine cycle
6	1	1	0	1	0	12 MHz
7	1	1	0	0	1	14 MHz
8	1	1	0	0	0	16 MHz

Setting of the Serial I/O prescaler (CDCR)

* For details, see 2.6.4 "Serial I/O Prescaler".

These bits are initialized to '000' upon a reset. These bits must not be updated during data transfer.

Five types of internal shift clock and an external shift clock are available. Do not set 110 or 111 in SMD2, SMD1, and SMD0 as these values are reserved.

Shift operation can be performed for each instruction by specifying SCOE =0 during clock selection and by using the ports that share the SCK1 and SCK2 pins.

[bit 12] Serial I/O interrupt enable bit (SIE: Serial I/O interrupt enable)

This bit controls the serial I/O interrupt request as shown in Table 18.2-6.

Table 18.2-6 Setting the Interrupt Request Enable Bit

0	Serial I/O interrupt disabled [initial value]
1	Serial I/O interrupt enabled

This bit is initialized to '0' upon a reset. This bit is readable and writable.

[bit 11] Serial I/O interrupt request bit (SIR: Serial I/O interrupt request)

When serial data transfer is completed, '1' is set to this bit. If this bit is set while interrupts are enabled (SIE=1), an interrupt request is issued to the CPU. The clear condition varies with the MODE bit.

When '0' is written to the MODE bit, the SIR bit is cleared by writing '0'. When '1' is written to the MODE bit, the SIR bit is cleared by reading or writing to SDR. When the system is reset or '1' is written to the STOP bit, the SIR bit is cleared regardless of the MODE bit value.

Writing '1' to the SIR bit has no effect. '1' is always read by a read operation of a read-modify-write instruction.

[bit 10] Transfer status bit (BUSY)

The transfer status bit indicates whether serial transfer is being executed.

Table 18.2-7 Setting the Transfer Status Bit

BUSY	Operating
0	Stopped, or standing by for serial data register R/W [default]
1	Serial transfer

This bit is initialized to '0' upon a reset. This is a read-only bit.

[bit 9] Stop bit (STOP)

The stop bit forcibly terminates serial transfer. When '1' is written to this bit, the transfer is

stopped.

Table 18.2-8 Setting the Stop Bit

STOP	Operating
0	Normal operation
1	Transfer stop by STOP=1 [initial value]

This bit is initialized to '1' upon a reset. This bit is readable and writable.

[bit 8] Start bit (STRT: Start)

The start bit activates serial transfer. Writing '1' to this bit starts the data transfer when the MODE bit is set to 0. When the MODE bit is set to 1 and the STRT bit is set to 1, writing the data into serial data register starts the transfer.

Writing '1' is ignored while the system is performing serial transfer or standing by for a serial shift register read or write. Writing '0' has no effect.'0' is always read.

18.2.2 Serial Shift Data Register (SDR)

This serial data register stores the serial I/O transfer data. During transfer, the SDR must not be read or written to.

■ **Serial Shift Data Register (SDR)**

SDR
Address : $00002E_H$

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value XX_H
(undefined)

18.3 Serial I/O Prescaler (CDCR)

The Serial I/O Prescaler provides the shift clock for the Serial I/O.
The operation clock for the Serial I/O is obtained by dividing the machine clock. The Serial I/O is designed so that a constant baud rate can be obtained for a variety of machine clocks by the user of the communication prescaler. The CDCR register controls the machine clock division.

■ Serial I/O Prescaler (CDCR)

	15	14	13	12	11	10	9	8	Initial value
CDCR Address: 00006D$_H$	MD	—	—	—	DIV3	DIV2	DIV1	DIV0	0---1111$_B$
	R/W				R/W	R/W	R/W	R/W	

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

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

0: The Serial I/O Prescaler is disabled.

1: The Serial I/O Prescaler is enabled.

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

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

Table 18.3-1 Machine Clock Division Ratio

DIV3 to 0	Division ratio
1101_B	3
1100_B	4
1011_B	5
1010_B	6
1001_B	7
1000_B	8

Note:

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

18.4 Serial I/O Operation

The extended serial I/O consists of the serial mode control status register (SMCS) and shift register (SDR), and is used for input and output of 8-bit serial data.

■ Serial I/O Operation

The bits in the shift register are serially output via the serial output pin (SOT1 pin) at the falling edge of the serial shift clock (external clock or internal clock). The bits are serially input to the shift register (SDR) via the serial input pin (SIN1 pin) at the rising edge of the serial shift clock. The shift direction (transfer from MSB or LSB) is specified by the direction specification bit (BDS) of the serial mode control status register (SMCS).

At the end of serial data transfer, this block is stopped or stands by for a read or write of the data register according to the MODE bit of the serial mode control status register (SMCS). To start transfer from the stop or standby state, follow the procedure below.

- **To resume operation from the stop state, write '0' to the STOP bit and '1' to the STRT bit. (The STOP and STRT bits can be set simultaneously.)**
- **To resume operation from the serial shift data register R/W standby state, read or write to the data register.**

18.4.1 Shift Clock

There are two modes of shift clock: internal or external shift clock. These two modes are selected by setting the SMCS. To switch the modes, ensure that serial I/O transfer is stopped. To check whether the serial I/O transfer is stopped, read the BUSY bit.

■ Internal Shift Clock Mode

In internal shift clock mode, data transfer is based on the internal clock. As a synchronization timing output, a shift clock of 50% duty ratio can be output from the SCK pin. Data is transferred at one bit per clock. The transfer speed is expressed as follows:

$$\text{Transfer speed (s)} = \frac{A}{\text{/Internal clock machine cycle (Hz)}}$$

"A" is the division ratio indicated by the SMD bits of SMCS. The value can be 10, 20, 80, 160, or 320.

■ External Shift Clock Mode

In external shift clock mode, the data transfer is based on the external clock supplied via the SCK3 pin. Data is transferred at one bit per clock.

The transfer speed can be between DC and 1/(5 machine cycles). For example, the transfer speed can be up to 2 MHz when 1 machine cycle is equal to 0.1 ms.

A data bit can also be transferred by software, which is enabled as described below.

Select external shift clock mode, and write '0' to the SCOE bit of SMCS. Then, write '1' to the direction register for the port sharing the SCK3 pin, and place the port in output mode. Then, when '1' and '0' are written to the data register (PDR) of the port, the port value output via the SCK3 pin is fetched as the external clock and transfer starts. Ensure that the shift clock starts from 'H.'

Note:

The SMCS or SDR must not be written to during serial I/O operation.

18.4.2 Serial I/O Operation

There are four serial I/O operation statuses:

- **STOP**
- **Halt**
- **SDR R/W standby**
- **Transfer**

■ **Serial I/O Operation**

❍ **STOP**

The STOP state is initiated upon RESET or when '1' is written to the STOP bit of SMCS. The shift counter is initialized, and '0' is written to SIR.

To resume operation from the STOP state, write '0' to STOP and '1' to STRT. (These two bits can be written to simultaneously.) Since the STOP bit overrides the STRT bit, transfer cannot be started by writing '1' to STRT while '1' is written to STOP.

❍ **Halt**

When transfer is completed while the MODE bit is '0,' '0' is set to BUSY and '1' is set to SIR of the SMCS, the counter is initialized, and the system stops. To resume operation from the stop state, write '1' to STRT.

❍ **Serial data register R/W standby**

When transfer is completed while the MODE bit is '1,' '0' is set to BUSY and '1' is set to SIR of the SMCS, and the system enters the serial data register R/W standby state. If the interrupt enable flag is set, an interrupt signal is output from this block.

To resume operation from R/W standby state, read or write to the serial data register. This sets the BUSY bit to '1' and starts data transfer.

❍ **Transfer**

'1' is set to the BUSY bit and serial transfer is being performed. According to the MODE bit, the halt state or R/W standby state comes next.

Figures 18.4-1 is diagrams of the operation transitions.

Figure 18.4-1 Extended I/O Serial Interface Operation Transitions

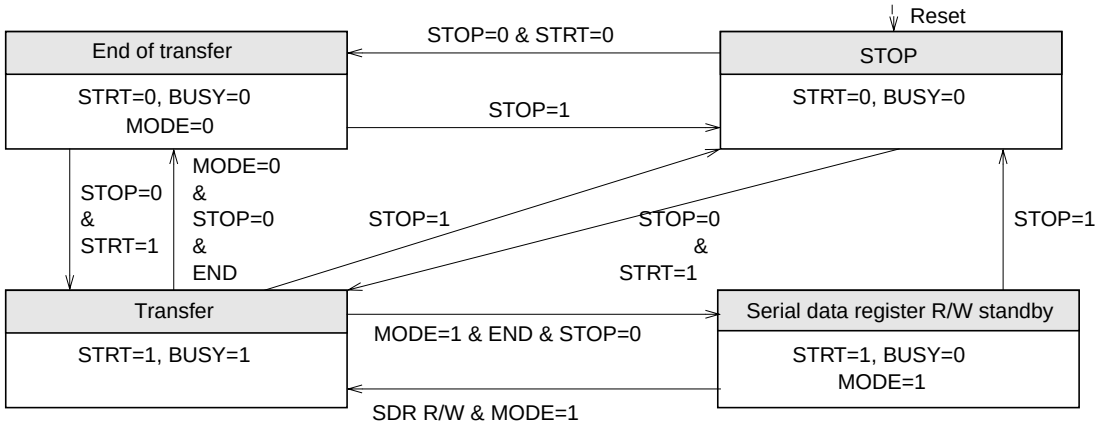
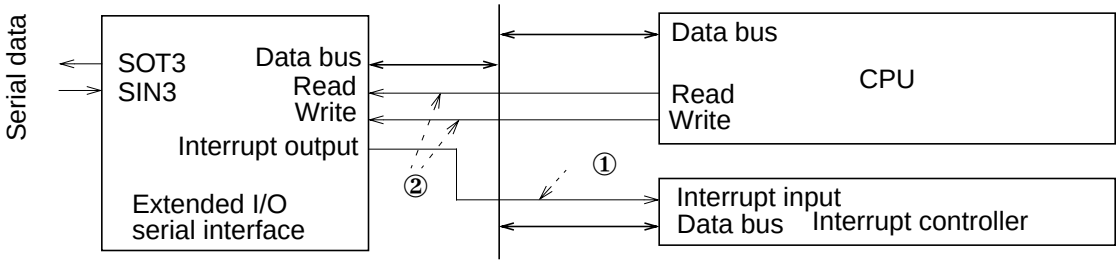


Figure 18.4-2 Serial Data Register Read/write



1. If '1' is written to MODE, transfer ends according to the shift clock counter. The read/write standby state starts when '1' is written to SIR. If '1' is written to the SIE bit, an interrupt signal is generated. No interrupt signal is generated when SIE is inactive or transfer has been terminated by writing '1' to STOP.
2. Reading or writing to the serial data register clears the interrupt request and starts serial transfer.

18.4.3 Shift Operation Start/Stop Timing

To start the shift operation, set the STOP bit to "0" and the STRT bit to "1" in SMCS. The system may stop the shift operation at the end of transfer or when "1" is set in the STOP bit.

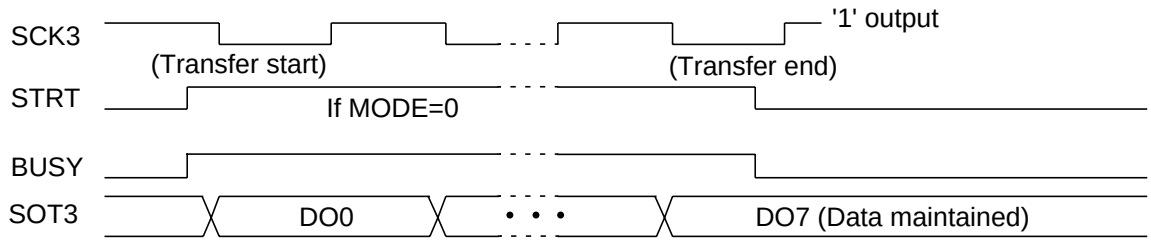
- **Stop by STOP=1 -> The system stops with SIR=0 regardless of the MODE bit.**
- **Stop by end of transfer -> The system stops with SIR=1 regardless of the MODE bit.**

Regardless of the MODE bit, the BUSY bit becomes '1' during serial transfer and becomes '0' during stop or R/W standby state. To check the transfer status, read this bit.

■ **Shift Operation Start/Stop Timing**

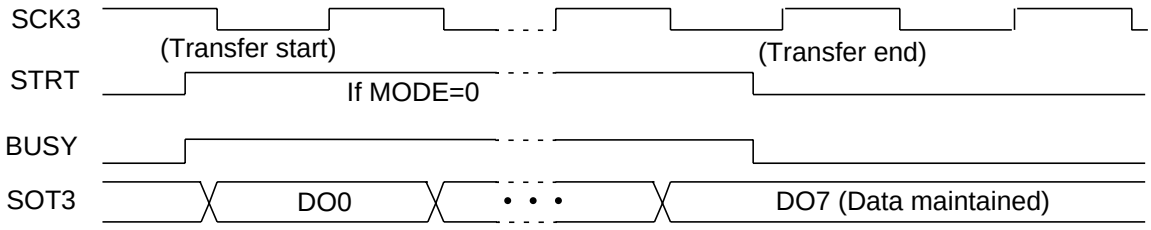
❍ **Internal shift clock mode (LSB first)**

Figure 18.4-3 Shift Operation Start/Stop Timing (Internal Clock)



❍ **External shift clock mode (LSB first)**

Figure 18.4-4 Shift Operation Start/Stop Timing (External Clock)



❍ **External shift clock mode with instruction shift (LSB first)**

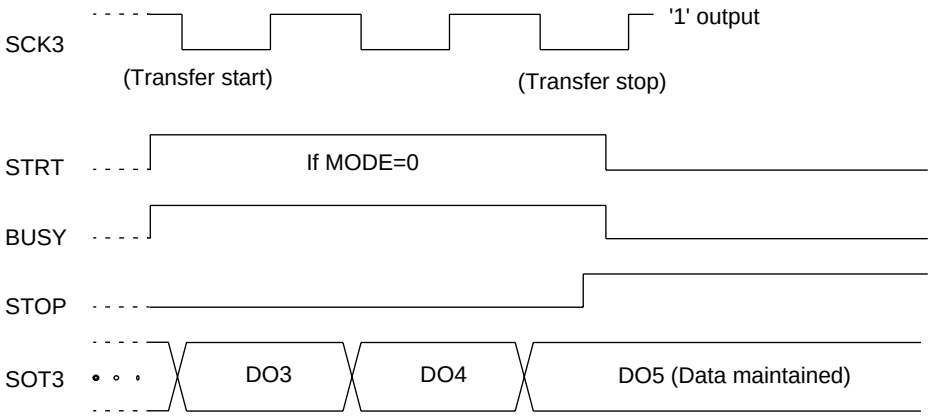
Figure 18.4-5 Shift Operation Start/Stop Timing (External Shift Clock Mode with Instruction Shift)



* For an instruction shift, 'H' is output when '1' is written to the bit corresponding to SCK of PDR, and 'L' is output when '0' is written. (When SCOE=0 in external shift clock mode)

❍ **Stop by STOP=1 (LSB first, internal clock)**

Figure 18.4-6 Stop Timing when '1' is Written to the STOP Bit

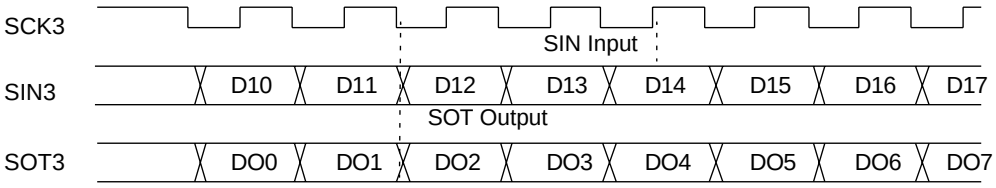


Note: DO7 to DO0 indicate output data.

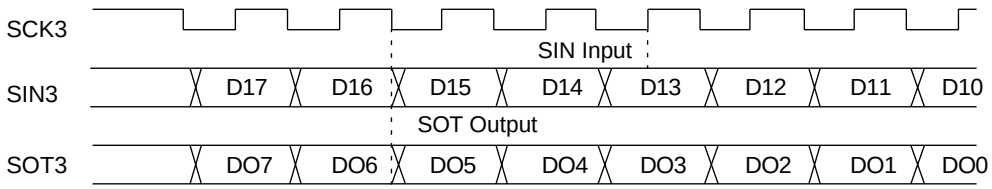
During serial data transfer, data is output from the serial output pin (SOT2) at the falling edge of the shift clock, and input from the serial input pin (SIN) at the rising edge.

Figure 18.4-7 Serial Data I/O Shift Timing

❍ LSB first (When the BDS bit is '0')



❍ MSB first (When the BDS bit is '1')

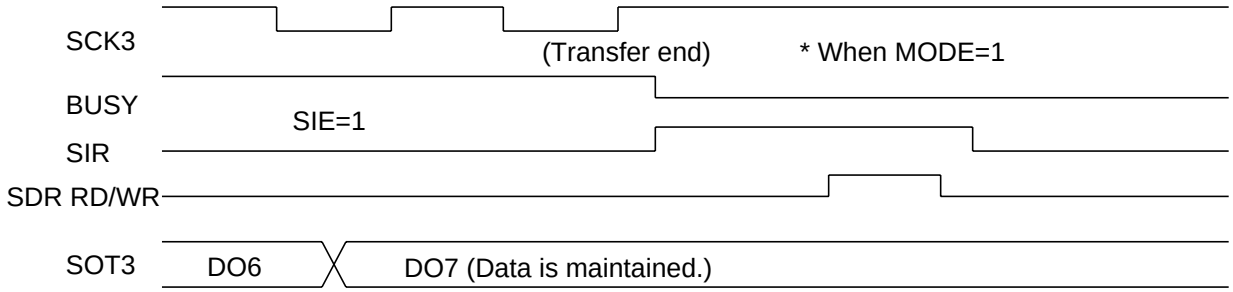


18.4.4 Interrupt Function of the Extended Serial I/O Interface

This block can issue an interrupt request to the CPU. At the end of data transfer, the SIR bit is set as an interrupt flag. When '1' is written to the interrupt enable bit (SIE bit) of SMCS, an interrupt request is issued to the CPU.

■ **Interrupt Function of the Extended Serial I/O Interface**

Figure 18.4-8 Interrupt Signal Output Timing of the Extended Serial I/O Interface



18.5 Negative Clock Operation

The MB90590 Series supports the negative clock operation of the Serial I/O. In this opearation, the shift clock signal is simply negated by a inverter. Therefore the definition of the shift clock signal in the preceeding sections of the Serial I/O is inversed from the logic low level to logic high level, from the negative edge to the positive edge and vise-versa. This is the same for both the serial clock input and output.

■ Negative Clock Operation

SES
Address : 00002F$_H$

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	NEG
							R/W

Initial value _____0$_B$

Table 18.5-1 Setting the NEG Bit

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

CHAPTER 19 CAN CONTROLLER

This chapter explains the functions and operations of the CAN controller.

19.1 Features of CAN Controller

19.2 Block Diagram of CAN Controller

19.3 List of Overall Control Registers

19.4 List of Message Buffers (ID Registers)

19.5 List of Message Buffers (DLC Registers and Data Registers)

19.6 Classifying the CAN Controller Registers

19.7 Transmission of CAN Controller

19.8 Reception of CAN Controller

19.9 Reception Flowchart of CAN Controller

19.10 How to Use the CAN Controller

19.11 Procedure for Transmission by Message Buffer (x)

19.12 Procedure for Reception by Message Buffer (x)

19.13 Setting Configuration of Multi-level Message Buffer

19.1 Features of CAN Controller

The CAN controller is a module built into a 16-bit microcontroller (F^2MC-16LX). The CAN (Controller Area Network) is the standard protocol for serial communication between automobile controllers and is widely used in industrial applications.

■ Features of CAN Controller

The CAN controller has the following features:

- **Conforms to CAN Specification Version 2.0 Part A and B**

  Supports transmission/reception in standard frame and extended frame formats

- **Supports transmitting of data frames by receiving remote frames**

- **16 transmitting/receiving message buffers**

  29-bit ID and 8-byte data

  Multi-level message buffer configuration

- **Supports full-bit comparison, full-bit mask and partial bit mask filtering.**

  Two acceptance mask registers in either standard frame format or extended frame formats

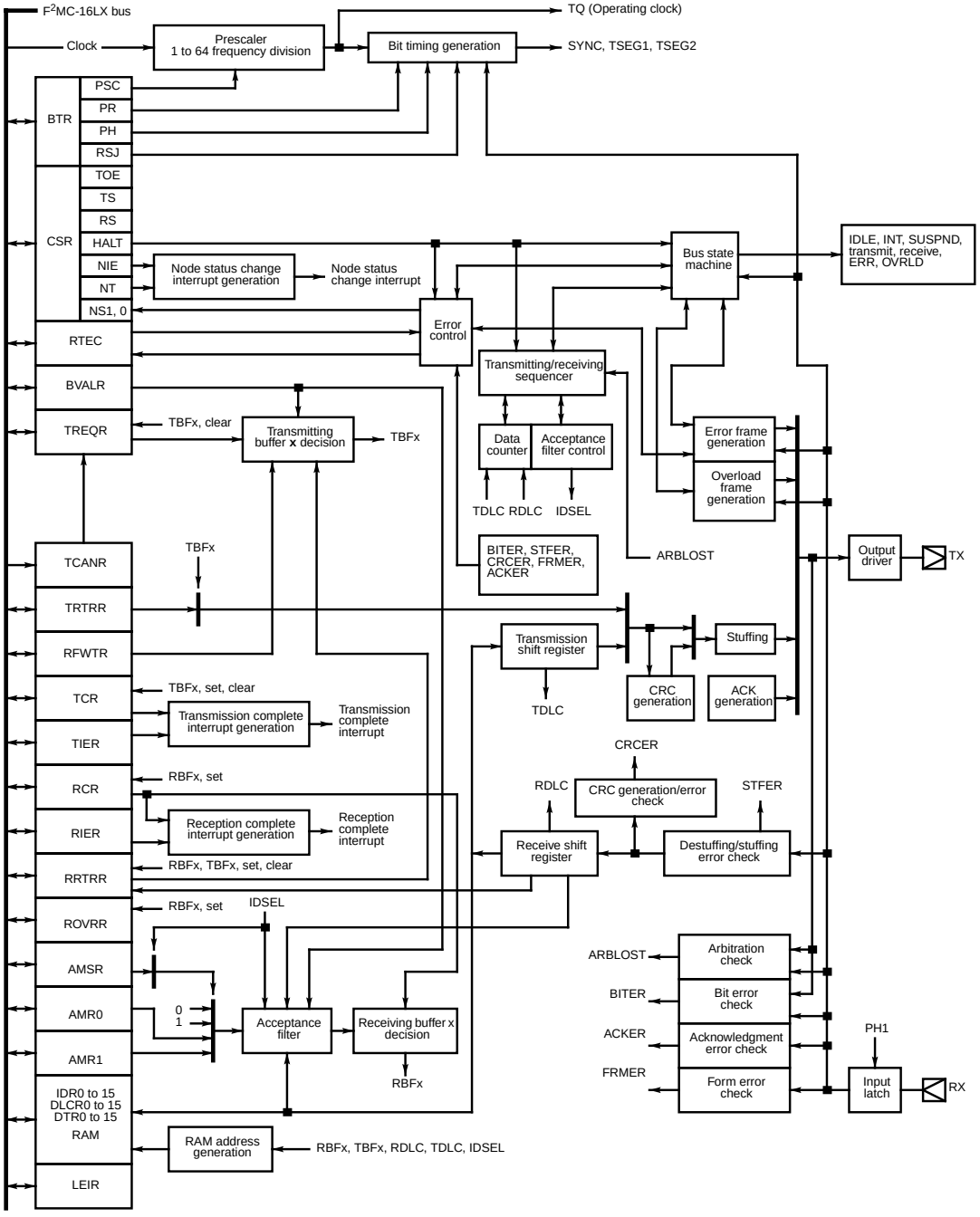
- **Bit rate programmable from 10 Kbits/s to 1 Mbits/s (when input clock is at 16 MHz)**

19.2 Block Diagram of CAN Controller

Figure 19.2-1 shows a block diagram of the CAN controller.

■ Block Diagram of CAN Controller

Figure 19.2-1 Block Diagram of CAN Controller



19.3 List of Overall Control Registers

Table 19.3-1 lists overall control registers.

■ **List of Overall Control Registers**

Table 19.3-1 List of Overall Control Registers

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
000070H	000080_H	Message buffer valid register	BVALR	R/W	00000000 00000000
000071H	000081_H				
000072_H	000082_H	Transmit request register	TREQR	R/W	00000000 00000000
000073_H	000083_H				
000074_H	000084_H	Transmit cancel register	TCANR	W	00000000 00000000
000075_H	000085_H				
000076_H	000086_H	Transmit complete register	TCR	R/W	00000000 00000000
000077_H	000087_H				
000078_H	000088_H	Receive complete register	RCR	R/W	00000000 00000000
000079_H	000089_H				
$00007A_H$	$00008A_H$	Remote request receiving register	RRTRR	R/W	00000000 00000000
$00007B_H$	$00008B_H$				
$00007C_H$	$00008C_H$	Receive overrun register	ROVRR	R/W	00000000 00000000
$00007D_H$	$00008D_H$				
$00007E_H$	$00008E_H$	Receive interrupt enable register	RIER	R/W	00000000 00000000
$00007F_H$	$00008F_H$				
$001C00_H$	$001D00_H$	Control status register	CSR	R/W, R	00---000 0----001
$001C01_H$	$001D01_H$				
$001C02_H$	$001D02_H$	Last event indicator register	LEIR	R/W	-------- 000-0000
$001C03_H$	$001D03_H$				
$001C04_H$	$001D04_H$	Receive/transmit error counter	RTEC	R	00000000 00000000
$001C05_H$	$001D05_H$				
$001C06_H$	$001D06_H$	Bit timing register	BTR	R/W	-1111111 11111111
$001C07_H$	$001D07_H$				

Table 19.3-1 List of Overall Control Registers (Continued)

Address		Register	Abbreviation	Access	Initial Value
CAN0	CAN1				
$001C08_H$	$001D08_H$	IDE register	IDER	R/W	XXXXXXXX XXXXXXXX
$001C09_H$	$001D09_H$				
$001C0A_H$	$001D0A_H$	Transmit RTR register	TRTRR	R/W	00000000 00000000
$001C0B_H$	$001D0B_H$				
$001C0C_H$	$001D0C_H$	Remote frame receive waiting register	RFWTR	R/W	XXXXXXXX XXXXXXXX
$001C0D_H$	$001D0D_H$				
$001C0E_H$	$001D0E_H$	Transmit interrupt enable register	TIER	R/W	00000000 00000000
$001C0F_H$	$001D0F_H$				
$001C10_H$	$001D10_H$	Acceptance mask select register	AMSR	R/W	XXXXXXXX XXXXXXXX
$001C11_H$	$001D11_H$				
$001C12_H$	$001D12_H$				XXXXXXXX XXXXXXXX
$001C13_H$	$001D13_H$				
$001C14_H$	$001D14_H$	Acceptance mask register 0	AMR0	R/W	XXXXXXXX XXXXXXXX
$001C15_H$	$001D15_H$				
$001C16_H$	$001D16_H$				XXXXX--- XXXXXXXX
$001C17_H$	$001D17_H$				
$001C18_H$	$001D18_H$	Acceptance mask register 1	AMR1	R/W	XXXXXXXX XXXXXXXX
$001C19_H$	$001D19_H$				
$001C1A_H$	$001D1A_H$				XXXXX--- XXXXXXXX
$001C1B_H$	$001D1B_H$				

19.4 List of Message Buffers (ID Registers)

Table 19.4-1 lists message buffers (ID registers).

■ **List of Message Buffers (ID registers)**

Table 19.4-1 List of Message Buffers (ID Registers)

Address		Register	Abbrevia-tion	Access	Initial Value
CAN0	CAN1				
$001A00_H$ to $001A1F_H$	$001B00_H$ to $001B1F_H$	General-purpose RAM	--	R/W	XXXXXXXX to XXXXXXXX
$001A20_H$	$001B20_H$	ID register 0	IDR0	R/W	XXXXXXXX XXXXXXXX
$001A21_H$	$001B21_H$				
$001A22_H$	$001B22_H$				XXXXX--- XXXXXXXX
$001A23_H$	$001B23_H$				
$001A24_H$	$001B24_H$	ID register 1	IDR1	R/W	XXXXXXXX XXXXXXXX
$001A25_H$	$001B25_H$				
$001A26_H$	$001B26_H$				XXXXX--- XXXXXXXX
$001A27_H$	$001B27_H$				
$001A28_H$	$001B28_H$	ID register 2	IDR2	R/W	XXXXXXXX XXXXXXXX
$001A29_H$	$001B29_H$				
$001A2A_H$	$001B2A_H$				XXXXX--- XXXXXXXX
$001A2B_H$	$001B2B_H$				
$001A2C_H$	$001B2C_H$	ID register 3	IDR3	R/W	XXXXXXXX XXXXXXXX
$001A2D_H$	$001B2D_H$				
$001A2E_H$	$001B2E_H$				XXXXX--- XXXXXXXX
$001A2F_H$	$001B2F_H$				
$001A30_H$	$001B30_H$	ID register 4	IDR4	R/W	XXXXXXXX XXXXXXXX
$001A31_H$	$001B31_H$				
$001A32^H$	$001B32^H$				XXXXX--- XXXXXXXX
$001A33_H$	$001B33_H$				

Table 19.4-1 List of Message Buffers (ID Registers) (Continued)

Address		Register	Abbrevia -tion	Access	Initial Value
CAN0	CAN1				
$001A34_H$	$001B34_H$	ID register 5	IDR5	R/W	XXXXXXXX XXXXXXXX
$001A35_H$	$001B35_H$				
$001A36_H$	$001B36_H$				XXXXX--- XXXXXXXX
$001A37_H$	$001B37_H$				
$001A38_H$	$001B38_H$	ID register 6	IDR6	R/W	XXXXXXXX XXXXXXXX
$001A39_H$	$001B39_H$				
$001A3A_H$	$001B3A_H$				XXXXX--- XXXXXXXX
$001A3B_H$	$001B3B_H$				
$001A3C_H$	$001B3C_H$	ID register 7	IDR7	R/W	XXXXXXXX XXXXXXXX
$001A3D_H$	$001B3D_H$				
$001A3E_H$	$001B3E_H$				XXXXX--- XXXXXXXX
$001A3F_H$	$001B3F_H$				
$001A40_H$	$001B40_H$	ID register 8	IDR8	R/W	XXXXXXXX XXXXXXXX
001A41H	001B41H				
001A42H	001B42H				XXXXX--- XXXXXXXX
001A43H	001B43H				
$001A44_H$	$001B44_H$	ID register 9	IDR9	R/W	XXXXXXXX XXXXXXXX
$001A45_H$	$001B45_H$				
$001A46_H$	$001B46_H$				XXXXX--- XXXXXXXX
$001A47_H$	$001B47_H$				
$001A48_H$	$001B48_H$	ID register 10	IDR10	R/W	XXXXXXXX XXXXXXXX
$001A49_H$	$001B49_H$				
$001A4A_H$	$001B4A_H$				XXXXX--- XXXXXXXX
$001A4B_H$	$001B4B_H$				
$001A4C_H$	$001B4C_H$	ID register 11	IDR11	R/W	XXXXXXXX XXXXXXXX
$001A4D_H$	$001B4D_H$				
$001A4E_H$	$001B4E_H$				XXXXX--- XXXXXXXX
$001A4F_H$	$001B4F_H$				

Table 19.4-1 List of Message Buffers (ID Registers) (Continued)

Address		Register	Abbrevia -tion	Access	Initial Value
CAN0	CAN1				
$001A50_H$	$001B50_H$	ID register 12	IDR12	R/W	XXXXXXXX XXXXXXXX
$001A51_H$	$001B51_H$				
$001A52_H$	$001B52_H$				XXXXX--- XXXXXXXX
$001A53_H$	$001B53_H$				
$001A54_H$	001B54H	ID register 13	IDR13	R/W	XXXXXXXX XXXXXXXX
$001A55_H$	001B55H				
$001A56_H$	$001B56_H$				XXXXX--- XXXXXXXX
$001A57_H$	$001B57_H$				
$001A58_H$	$001B58_H$	ID register 14	IDR14	R/W	XXXXXXXX XXXXXXXX
$001A59_H$	$001B59_H$				
$001A5A_H$	$001B5A_H$				XXXXX--- XXXXXXXX
$001A5B_H$	$001B5B_H$				
$001A5C_H$	$001B5C_H$	ID register 15	IDR15	R/W	XXXXXXXX XXXXXXXX
$001A5D_H$	$001B5D_H$				
$001A5E_H$	$001B5E_H$				XXXXX--- XXXXXXXX
$001A5F_H$	$001B5F_H$				

19.5 List of Message Buffers (DLC Registers and Data Registers)

Table 19.5-1 lists message buffers (DLC registers), and Table 19.5-2 lists message buffers (data registers).

■ List of Message Buffers (DLC Registers and Data Registers)

Table 19.5-1 List of Message Buffers (DLC Registers and Data Registers)

Address		Register	Abbrevia-tion	Access	Initial Value
CAN0	CAN1				
$001A60_H$	$001B60_H$	DLC register 0	DLCR0	R/W	----XXXX
001A61H	$001B61_H$				
$001A62_H$	$001B62_H$	DLC register 1	DLCR1	R/W	----XXXX
$001A63_H$	$001B63_H$				
$001A64_H$	$001B64_H$	DLC register 2	DLCR2	R/W	----XXXX
$001A65_H$	$001B65_H$				
$001A66_H$	$001B66_H$	DLC register 3	DLCR3	R/W	----XXXX
$001A67_H$	$001B67_H$				
$001A68_H$	$001B68_H$	DLC register 4	DLCR4	R/W	----XXXX
$001A69_H$	$001B69_H$				
$001A6A_H$	$001B6A_H$	DLC register 5	DLCR5	R/W	----XXXX
$001A6B_H$	$001B6B_H$				
$001A6C_H$	$001B6C_H$	DLC register 6	DLCR6	R/W	----XXXX
$001A6D_H$	$001B6D_H$				
$001A6E_H$	$001B6E_H$	DLC register 7	DLCR7	R/W	----XXXX
$001A6F_H$	$001B6F_H$				
$001A70_H$	$001B70_H$	DLC register 8	DLCR8	R/W	----XXXX
$001A71_H$	$001B71_H$				
$001A72_H$	$001B72_H$	DLC register 9	DLCR9	R/W	----XXXX
$001A73_H$	$001B73_H$				
$001A74_H$	$001B74_H$	DLC register 10	DLCR10	R/W	----XXXX
$001A75_H$	$001B75_H$				

Table 19.5-1 List of Message Buffers (DLC Registers and Data Registers) (Continued)

Address		Register	Abbrevia-tion	Access	Initial Value
CAN0	CAN1				
$001A76_H$	$001B76_H$	DLC register 11	DLCR11	R/W	----XXXX
$001A77_H$	$001B77_H$				
$001A78_H$	$001B78_H$	DLC register 12	DLCR12	R/W	----XXXX
$001A79_H$	$001B79_H$				
$001A7A_H$	$001B7A_H$	DLC register 13	DLCR13	R/W	----XXXX
$001A7B_H$	$001B7B_H$				
$001A7C_H$	$001B7C_H$	DLC register 14	DLCR14	R/W	----XXXX
$001A7D_H$	$001B7D_H$				
$001A7E_H$	$001B7E_H$	DLC register 15	DLCR15	R/W	----XXXX
$001A7F_H$	$001B7F_H$				

■ List of Message Buffers (Data Registers)

Table 19.5-2 List of Message Buffers (Data Registers)

Address		Register	Abbrevia -tion	Access	Initial Value
CAN0	CAN1				
$001A80_H$ to $001A87_H$	$001B80_H$ to $001B87_H$	Data register 0 (8 bytes)	DTR0	R/W	XXXXXXXX to XXXXXXXX
$001A88_H$ to $001A8F_H$	$001B88_H$ to $001B8F_H$	Data register 1 (8 bytes)	DTR1	R/W	XXXXXXXX to XXXXXXXX
$001A90_H$ to $001A97_H$	$001B90_H$ to $001B97_H$	Data register 2 (8 bytes)	DTR2	R/W	XXXXXXXX to XXXXXXXX
$001A98_H$ to $001A9F_H$	$001B98_H$ to $001B9F_H$	Data register 3 (8 bytes)	DTR3	R/W	XXXXXXXX to XXXXXXXX
001AA0H to $001AA7_H$	001BA0H to $001BA7_H$	Data register 4 (8 bytes)	DTR4	R/W	XXXXXXXX to XXXXXXXX
001AA8H to $001AAF_H$	001BA8H to $001BAF_H$	Data register 5 (8 bytes)	DTR5	R/W	XXXXXXXX to XXXXXXXX
001AB0H to $001AB7_H$	001BB0H to $001BB7_H$	Data register 6 (8 bytes)	DTR6	R/W	XXXXXXXX to XXXXXXXX
001AB8H to $001ABF_H$	001BB8H to $001BBF_H$	Data register 7 (8 bytes)	DTR7	R/W	XXXXXXXX to XXXXXXXX
001AC0H to $001AC7_H$	001BC0H to $001BC7_H$	Data register 8 (8 bytes)	DTR8	R/W	XXXXXXXX to XXXXXXXX
001AC8H to $001ACF_H$	001BC8H to $001BCF_H$	Data register 9 (8 bytes)	DTR9	R/W	XXXXXXXX to XXXXXXXX
001AD0H to $001AD7_H$	001BD0H to $001BD7_H$	Data register 10 (8 bytes)	DTR10	R/W	XXXXXXXX to XXXXXXXX
$001AD8_H$ to $001ADF_H$	$001BD8_H$ to $001BDF_H$	Data register 11 (8 bytes)	DTR11	R/W	XXXXXXXX to XXXXXXXX
$001AE0_H$ to $001AE7_H$	$001BE0_H$ to $001BE7_H$	Data register 12 (8 bytes)	DTR12	R/W	XXXXXXXX to XXXXXXXX

Table 19.5-2 List of Message Buffers (Data Registers) (Continued)

Address		Register	Abbrevia -tion	Access	Initial Value
CAN0	CAN1				
$001AE8_H$ to $001AEF_H$	$001BE8_H$ to $001BEF_H$	Data register 13 (8 bytes)	DTR13	R/W	XXXXXXXX to XXXXXXXX
$001AF0_H$ to $001AF7_H$	$001BF0_H$ to $001BF7_H$	Data register 14 (8 bytes)	DTR14	R/W	XXXXXXXX to XXXXXXXX
001AF8H to 001AFFH	001BF8H to 001BFFH	Data register 15 (8 bytes)	DTR15	R/W	XXXXXXXX to XXXXXXXX

19.6 Classifying the CAN Controller Registers

There are three types of CAN controller registers:

- **Overall control registers**
- **Message buffer control registers**
- **Message buffers**

■ Overall Control Registers

The overall control registers are the following four registers:

- Control status register (CSR)
- Last event indicator register (LEIR)
- Receive and transmit error counter (RTEC)
- Bit timing register (BTR)

■ Message Buffer Control Registers

The message buffer control registers are the following 14 registers:

- Message buffer valid register (BVALR)
- IDE register (IDER)
- Transmission request register (TREQR)
- Transmission RTR register (TRTRR)
- Remote frame receiving wait register (RFWTR)
- Transmission cancel register (TCANR)
- Transmission complete register (TCR)
- Transmission interrupt enable register (TIER)
- Reception complete register (RCR)
- Remote request receiving register (RRTRR)
- Receive overrun register (ROVRR)
- Reception interrupt enable register (RIER)
- Acceptance mask select register (AMSR)
- Acceptance mask registers 0 and 1 (AMR0 and AMR1)

■ Message Buffers

The message buffers are the following three registers:

- ID register x (x = 0 to 15) (IDRx)
- DLC register x (x = 0 to 15) (DLCRx)
- Data register x (x = 0 to 15) (DTRx)

19.6.1 Control Status Register (CSR)

Control status register (CSR) is prohibited from executing any bit manipulation instructions (Read-Modify-Write instructions).

■ **Control Status Register (CSR)**

	15	14	13	12	11	10	9	8
Address: $001C01_H$ (CAN0) $001D01_H$ (CAN1)	TS	RS	—	—	—	NT	NS1	NS0
Read/write:	(R)	(R)	(—)	(—)	(—)	(R/W)	(R)	(R)
Initial value:	(0)	(0)	(—)	(—)	(—)	(0)	(0)	(0)

	7	6	5	4	3	2	1	0
Address: $001C00_H$ (CAN0) $001D00_H$ (CAN1)	TOE	—	—	—	—	NIE	Reserved	HALT
Read/write :	(R/W)	(—)	(—)	(—)	(—)	(R/W)	(W)	(R/W)
Initial value:	(0)	(—)	(—)	(—)	(—)	(0)	(0)	(1)

[Bit 15] TS: Transmit status bit

This bit indicates whether a message is being transmitted.

0: Message not being transmitted

1: Message being transmitted

This bit is 0 even while error and overload frames are transmitted.

[Bit 14] RS: Receive status bit

This bit indicates whether a message is being received.

0: Message not being received

1: Message being received

While a message is on the bus, this bit becomes 1. Therefore, this bit is also 1 while a message is being transmitted. This bit does not necessarily indicates whether a receiving message passes through the acceptance filter.

As a result, when this bit is 0, it implies that the bus operation is stopped (HALT = 0); the bus is in the intermission/bus idle or a error/overload frame is on the bus.

[Bit 10] NT: Node status transition flag

If the node status is changed to increment, or from Bus Off to Error Active, this bit is set to 1.

In other words, the NT bit is set to 1 if the node status is changed from Error Active (00) to Warning (01), from Warning (01) to Error Passive (10), from Error Passive (10) to Bus Off (11), and from Bus Off (11) to Error Active (00). Numbers in parentheses indicate the values of NS1 and NS0 bits.

When the node status transition interrupt enable bit (NIE) is 1, an interrupt is generated. Writing 0 sets the NT bit to 0. Writing 1 to the NT bit is ignored. 1 is read when a Read Modify Write instruction is performed.

[Bits 9 to 8] NS1 and NS0: Node status bits 1 and 0

These bits indicate the current node status.

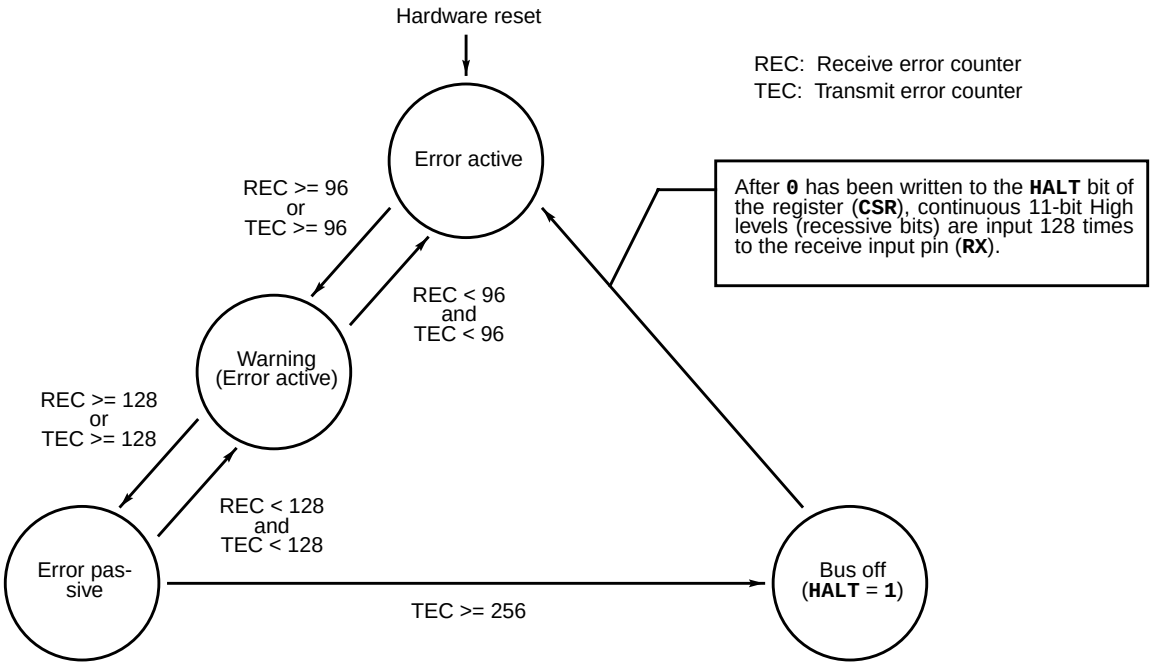
Table 19.6-1 Correspondence between NS1 and NS0 and Node Status

NS1	NS0	Node Status
0	0	Error active
0	1	Warning (error active)
1	0	Error passive
1	1	Bus off

Note:

Warning (error active) is included in the error active in CAN Specification 2.0B for the node status, however, indicates that the transmit error counter or receive error counter has exceeded 96. The node status change diagram is shown in Figure 19.6-1.

Figure 19.6-1 Node Status Transition Diagram



[Bit 7] TOE: Transmit output enable bit

Writing 1 to this bit switches from a general-purpose port pin to a transmit pin of the CAN controller.

0: General-purpose port pin

1: Transmit pin of CAN controller

[Bit 2] NIE: Node status transition interrupt enable bit

This bit enables or disables a node status transition interrupt (when NT = 1).

0: Node status transition interrupt disabled

1: Node status transition interrupt enabled

[Bit 1] Reserved

The is a reserved bit. Do not write "1" to this bit.

[Bit 0] HALT: Bus operation stop bit

This bit sets or cancels bus operation stop, or displays its state.

19.6.2 Bus Operation Stop Bit (HALT = 1)

The bus operation stop bit sets or cancels stopping of bus operation, or indicates its status

■ Conditions for Setting Bus Operation Stop (HALT=1)

There are three conditions for setting bus operation stop (HALT = 1):

- After hardware reset
- When node status changed to bus off
- By writing 1 to HALT

Note:

The bus operation should be stopped by writing 1 to HALT before the F^2MC-16LX is changed in low-power consumption mode (stop mode, timer mode, and hardware stand-by mode).

If transmission is in progress when 1 is written to HALT, the bus operation is stopped (HALT = 1) after transmission is terminated. If reception is in progress when 1 is written to HALT, the bus operation is stopped immediately (HALT = 1). If received messages are being stored in the message buffer (x), stop the bus operation (HALT = 1) after storing the messages.

To check whether the bus operation has stopped, always read the HALT bit.

■ Conditions for Canceling Bus Operation Stop (HALT = 0)

- By writing 0 to HALT

Note:

Canceling the bus operation stop after hardware reset or by writing 1 to HALT as above conditions is performed after 0 is written to HALF and continuous 11-bit High levels (recessive bits) have been input to the receive input pin (RX) (HALT = 0).

Canceling the bus operation stop when the node status is changed to bus off as above conditions is performed after 0 is written to HALF and continuous 11-bit High levels (recessive bits) have been input 128 times to the receive input pin (RX) (HALT = 0). Then, the values of both transmit and receive error counters reach 0 and the node status is changed to error active.

■ State during Bus Operation Stop (HALT = 1)

- The bus does not perform any operation, such as transmission and reception.
- The transmit output pin (TX) outputs a High level (recessive bit).
- The values of other registers and error counters are not changed.

Note:

The bit timing register (BTR) should be set during bus operation stop (HALT = 1).

19.6.3 Last Event Indicator Register (LEIR)

This register indicates the last event.
The NTE, TCE, and RCE bits are exclusive. When the corresponding bit of the last event is set to 1, other bits are set to 0s.

■ **Last Event Indicator Register (LEIR)**

	7	6	5	4	3	2	1	0
Address: 001C02$_H$ (CAN0) 001D02$_H$ (CAN1)	NTE	TCE	RCE	—	MBP3	MBP2	MBP1	MBP0
Read/write :	(R/W)	(R/W)	(R/W)	(—)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(—)	(0)	(0)	(0)	(0)

[Bit 7] NTE: Node status transition event bit

When this bit is 1, node status transition is the last event.

This bit is set to 1 at the same time the NT bit of the control status register (CSR) is set.

This bit is also set to 1 irrespective of the setting of the node status transition interrupt enable bit (NIE) of CSR.

Writing 0 to this bit sets the NTE bit to 0. Writing 1 to this bit is ignored.

1 is read when a Read Modify Write instruction is executed.

[Bit 6] TCE: Transmit completion event bit

When this bit is 1, it indicates that transmit completion is the last event.

This bit is set to 1 at the same time as any one of the bits of the transmit completion register (TCR). This bit is also set to 1, irrespective of the settings of the bits of the transmit interrupt enable register (TIER).

Writing 0 sets this bit to 0. Writing 1 to this bit is ignored.

1 is read when a Read Modify Write instruction is performed.

When this bit is set to 1, the MBP3 to MBP0 bits are used to indicate the message buffer number completing the transmit operation.

[Bit 5] RCE: Receive completion event bit

When this bit is 1, it indicates that receive completion is the last event.

This bit is set to 1 at the same time as any one of the bits of the receive complete register (RCR). This bit is also set to 1 irrespective of the settings of the bits of the receive interrupt enable register (RIER).

Writing 0 sets this bit to 0. Writing 1 to this bit is ignored.

1 is read when a Read Modify Write instruction is performed.

When this bit is set to 1, the MBP3 to MBP0 bits are used to indicate the message buffer number completing the receive operation.

[Bits 3 to 0] MBP3 to MBP0: Message buffer pointer bits

When the TCE or RCE bit is set to 1, these bits indicate the corresponding numbers of the message buffers (0 to 15). If the NTE bit is set to 1, these bits have no meaning.

Writing 0 sets these bits to 0s. Writing 1 to these bits is ignored.

1s are read when a Read Modify Write instruction is performed.

If LEIR is accessed within an CAN interrupt handler, the event causing the interrupt is not neccessarily the same as indicated by LEIR. In the time from interrupt request to the LEIR access by the interrupt handler there may occur other CAN events.

19.6.4 Receive and Transmit Error Counters (RTEC)

The receive and transmit error counters indicate the counts for transmission errors and reception errors defined in the CAN specifications. These registers can only be read.

■ **Receive and Transmit Error Counters (RTEC)**

	15	14	13	12	11	10	9	8
Address: $001C05_H$ (CAN0) $001D05_H$ (CAN1)	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
Read/write:	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

	7	6	5	4	3	2	1	0
Address: $001C04_H$ (CAN0) $001D04_H$ (CAN1)	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
Read/write:	(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

[Bits 15 to 8] TEC7 to TEC0: Transmit error counter

These are transmit error counters.

TEC7 to TEC0 values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Error Passive is indicated for the node status (NS1 and NS0 of control status register CSR = 11).

[Bits 7 to 0] REC7 to REC0: Receive error counter

These are receive error counters.

REC7 to REC0 values indicate 0 to 7 when the counter value is more than 256, and the subsequent increment is not counted for counter value. In this case, Bus Off is indicated for the node status (NS1 and NS0 of control status register CSR = 10).

19.6.5 Bit Timing Register (BTR)

Bit timing register (BTR) stores the prescaler and bit timing setting.

■ **Bit Timing Register (BTR)**

	15	14	13	12	11	10	9	8
Address: 001C07$_H$ (CAN0) 001D07$_H$ (CAN1)	—	TS2.2	TS2.1	TS2.0	TS1.3	TS1.2	TS1.1	TS1.0
Read/write:	(—)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(—)	(1)	(1)	(1)	(1)	(1)	(1)	(1)

	7	6	5	4	3	2	1	0
Address: 001C06$_H$ (CAN0) 001D06$_H$ (CAN1)	RSJ1	RSJ0	PSC5	PSC4	PSC3	PSC2	PSC1	PSC0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)

Note

This register should be set during bus operation stop (HALT = 1).

[Bits 14 to 12] TS2.2 to TS2.0: Time segment 2 setting bits 2 to 0

These bits define the number of the time quanta (TQ's) for the time segment 2 (TSEG2). The time segment 2 is equal to the phase buffer segment 2 (PHASE_SEG2) in the CAN specification.

[Bits 11 to 8] TS1.3 to TS1.0: Time segment 1 setting bits 3 to 0

These bits define the number of the time quanta (TQ's) for the time segment 1 (TSEG1). The time segment 1 is equal to the propagation segment (PROP_SEG) + phase buffer segment 1 (PHASE_SEG1) in the CAN specification.

[Bits 7 and 6] RSJ1 and RSJ0: Resynchronization jump width setting bits 1 and 0

These bits define the number of the time quanta (TQ's) for the resynchronization jump width.

[Bits 5 to 0] PSC5 to PSC0: Prescaler setting bits 5 to 0

These bits define the time quanta (TQ) of the CAN controller.

The bit time segments defined in the CAN specification, and the CAN controller are shown in Figures 19.6-2 and 19.6-3 respectively.

Figure 19.6-2 Bit Time Segment in CAN Specification

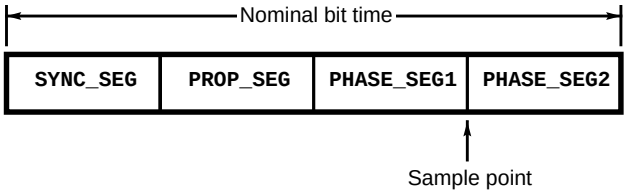
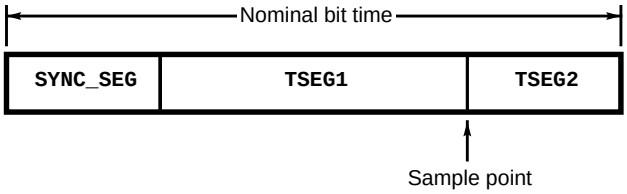


Figure 19.6-3 Bit Time Segment in CAN Controller



The relationship between PSC = PSC5 to PSC0, TSI = TS1.3 to TS1.0, TS2 = TS2.2 to TS1.0, and RSJ = RSJ1 and RSJ0 when the input clock (CLK), time quanta (TQ), bit time (BT), synchronous segment (SYNC_SEG), time segment 1 and 2 (TSEG1 and TSEG2), and resynchronization jump width [(RSJ1 and RSJ0) +1] frequency division is shown below.

The input clock is supplied with the machine clock.

TQ = (PSC + 1) × CLK

BT = SYNC_SEG + TSEG1 + TSEG2

(1 + (TS1 + 1) + (TS2 + 1)) × TQ

= (3 + TS1 + TS2) TQ

RSJW = (RSJ + 1) × TQ

For correct operation, the following conditions should be met.

BT >= 8TQ

TSEG2 >= RSJW + 2TQ*1

TSEG1 >= delay time*2 + RSJW

*1) 2TQ: Data processing time

*2) Delay time: Twice as long as the sum of the bus propagation, input comparator and output driver delay

19.6.6 Message Buffer Valid Register (BVALR)

Message buffer valid register (BVALR) stores the validity of the message buffers or displays their state.

■ Message Buffer Valid Register (BVALR)

	15	14	13	12	11	10	9	8
Address: 000071_H (CAN0) 000081_H (CAN1)	BVAL15	BVAL14	BVAL13	BVAL12	BVAL11	BVAL10	BVAL9	BVAL8
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)

	7	6	5	4	3	2	1	0
Address: 000070_H (CAN0) 000080_H (CAN1)	BVAL7	BVAL6	BVAL5	BVAL4	BVAL3	BVAL2	BVAL1	BVAL0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

0: Message buffer (x) invalid

1: Message buffer (x) valid

If the message buffer (x) is set to invalid, it will not transmit or receive messages.

If the buffer is set to invalid during transmission operating, it becomes invalid (BVALx = 0) after the transmission is completed or terminated by an error.

If the buffer is set to invalid during reception operating, it immediately becomes invalid (BVALx = 0). If received messages are stored in a message buffer (x), the message buffer (x) is invalid after storing the messages.

Note:

x indicates a message buffer number (x = 0 to 15).

When invaliding a message buffer (x) by writing 0 to a bit (BVALx), execution of a bit manipulation instruction is prohibited until the bit is set to 0.

19.6.7 IDE register (IDER)

This register stores the frame format used by the message buffers (x) during transmission/reception.

■ **IDE Register (IDER)**

	15	14	13	12	11	10	9	8
Address: $001C09_H$ (CAN0) $001D09_H$ (CAN1)	IDE15	IDE14	IDE13	IDE12	IDE11	IDE10	IDE9	IDE8
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

	7	6	5	4	3	2	1	0
Address: $001C08_H$ (CAN0) $001D08_H$ (CAN1)	IDE7	IDE6	IDE5	IDE4	IDE3	IDE2	IDE1	IDE0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

0: The standard frame format (ID11 bit) is used for the message buffer (x).

1: The extended frame format (ID29 bit) is used for the message buffer (x).

Note:

This register should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) = 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

19.6.8 Transmission Request Register (TREQR)

Transmission request register (TREQR) stores transmission requests to the message buffers (x) or displays their state.

■ Transmission Request Register (TREQR)

	15	14	13	12	11	10	9	8
Address: 000073_H (CAN0) 000083_H (CAN1)	TREQ15	TREQ14	TREQ13	TREQ12	TREQ11	TREQ10	TREQ9	TREQ8
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)

	7	6	5	4	3	2	1	0
Address: 000072_H (CAN0) 000082_H (CAN1)	TREQ7	TREQ6	TREQ5	TREQ4	TREQ3	TREQ2	TREQ1	TREQ0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

When 1 is written to TREQx, transmission to the message buffer (x) starts. If RFWTx of the remote frame receiving wait register (RFWTR)*1 is 0, transmission starts immediately. However, if RFWTx = 1, transmission starts after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR)*1 becomes 1). Transmission starts*2 immediately even when RFWTx = 1, if RRTRx is already 1 when 1 is written to TREQx.

*1: For RFWTR and TRTRR, see 19.6.9 and 19.6.10.

*2: For cancellation of transmission, see 19.6.11 and 19.6.12.

Writing 0 to TREQx is ignored.

0 is read when a Read Modify Write instruction is performed.

If clearing (to 0) at completion of the transmit operation and setting by writing 1 are concurrent, clearing is preferred.

If 1 is written to more than one bit, transmission is performed, starting with the lower-numbered message buffer (x).

TREQx is 1 while transmission is pending, and becomes 0 when transmission is completed or canceled.

19.6.9 Transmission RTR Register (TRTRR)

This register stores the RTR (Remote Transmission Request) bits for the message buffers (x).

■ **Transmission RTR Register (TRTRR)**

	15	14	13	12	11	10	9	8
Address: 001C0B$_H$ (CAN0) 001D0B$_H$ (CAN1)	TRTR15	TRTR14	TRTR13	TRTR12	TRTR11	TRTR10	TRTR9	TRTR8
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)

	7	6	5	4	3	2	1	0
Address: 001C0A$_H$ (CAN0) 001D0A$_H$ (CAN1)	TRTR7	TRTR6	TRTR5	TRTR4	TRTR3	TRTR2	TRTR1	TRTR0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

0: Data frame

1: Remote frame

19.6.10 Remote Frame Receiving Wait Register (RFWTR)

Remote frame receiving wait register (RFWTR) stores the conditions for starting transmission when a request for data frame transmission is set (TREQx of the transmission request register (TREQR) is 1 and TRTRx of the transmitting RTR register (TRTRR) is 0).

- **0: Transmission starts immediately**
- **1: Transmission starts after waiting until remote frame received (RRTRx of remote request receiving register (RRTRR) becomes 1)**

■ **Remote Frame Receiving Wait Register (RFWTR)**

	15	14	13	12	11	10	9	8
Address: $001C0D_H$ (CAN0) $001D0D_H$ (CAN1)	RFWT15	RFWT14	RFWT13	RFWT12	RFWT11	RFWT10	RFWT9	RFWT8
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

	7	6	5	4	3	2	1	0
Address: $001C0C_H$ (CAN0) $001D0C_H$ (CAN1)	RFWT7	RFWT6	RFWT5	RFWT4	RFWT3	RFWT2	RFWT1	RFWT0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

Note:

Transmission starts immediately if RRTRx is already 1 when a request for transmission is set.

For remote frame transmission, do not set RFWTx to 1.

19.6.11 Transmission Cancel Register (TCANR)

When 1 is written to TCANx, this register cancels a pending request for transmission to the message buffer (x).
At completion of cancellation, TREQx of the transmission request register (TREQR) becomes 0. Writing 0 to TCANx is ignored.
This is a write-only register and its read value is always 0.

■ **Transmission Cancel Register (TCANR)**

	15	14	13	12	11	10	9	8
Address: 000075_H (CAN0) 000085_H (CAN1)	TCAN15	TCAN14	TCAN13	TCAN12	TCAN11	TCAN10	TCAN9	TCAN8
Read/write:	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

	7	6	5	4	3	2	1	0
Address: 000074_H (CAN0) 000084_H (CAN1)	TCAN7	TCAN6	TCAN5	TCAN4	TCAN3	TCAN2	TCAN1	TCAN0
Read/write:	(W)	(W)	(W)	(W)	(W)	(W)	(W)	(W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

19.6.12 Transmission Complete Register (TCR)

At completion of transmission by the message buffer (x), the corresponding TCx becomes 1.
If TIEx of the transmission complete interrupt enable register (TIER) is 1, an interrupt occurs.

■ Transmission Complete Register (TCR)

	15	14	13	12	11	10	9	8
Address: 000077_H (CAN0) 000087_H (CAN1)	TC15	TC14	TC13	TC12	TC11	TC10	TC9	TC8
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)

	7	6	5	4	3	2	1	0
Address: 000076_H (CAN0) 000086_H (CAN1)	TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

❍ Conditions for TCx = 0

- Write 0 to TCx.
- Write 1 to TREQx of the transmission request register (TREQR).

After the completion of transmission, write 0 to TCx to set it to 0. Writing 1 to TCx is ignored.

1 is read when a Read Modify Write instruction is performed.

Note:

If setting to 1 by completion of the transmit operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.13 Transmission Interrupt Enable Register (TIER)

This register enables or disables the transmission interrupt by the message buffer (x). The transmission interrupt is generated at transmission completion (when TCx of the transmission complete register (TCR) is 1).

■ **Transmission Interrupt Enable Register (TIER)**

	15	14	13	12	11	10	9	8
Address: 001C0F$_H$ (CAN0) 001D0F$_H$ (CAN1)	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8
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)

	7	6	5	4	3	2	1	0
Address: 001C0E$_H$ (CAN0) 001D0E$_H$ (CAN1)	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

0: Transmission interrupt disabled

1: Transmission interrupt enabled

19.6.14 Reception Complete Register (RCR)

At completion of storing received message in the message buffer (x), RCx becomes 1. If RIEx of the reception complete interrupt enable register (RIER) is 1, an interrupt occurs.

■ **Reception Complete Register (RCR)**

	15	14	13	12	11	10	9	8
Address: 000079_H (CAN0) 000089_H (CAN1)	RC15	RC14	RC13	RC12	RC11	RC10	RC9	RC8
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)

	7	6	5	4	3	2	1	0
Address: 000078_H (CAN0) 000088_H (CAN1)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

❍ **Conditions for RCx = 0**

Write 0 to RCx.

After completion of handling received message, write 0 to RCx to set it to 0. Writing 1 to RCx is ignored.

1 is read when a Read Modify Write instruction is perforrmed.

Note:

If setting to 1 by completion of the receive operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.15 Remote Request Receiving Register (RRTRR)

After a remote frame is stored in the message buffer (x), RRTRx becomes 1 (at the same time as RCx setting to 1).

■ **Remote Request Receiving Register (RRTRR)**

	15	14	13	12	11	10	9	8
Address: $00007B_H$ (CAN0) $00008B_H$ (CAN1)	RRTR15	RRTR14	RRTR13	RRTR12	RRTR11	RRTR10	RRTR9	RRTR8
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)

	7	6	5	4	3	2	1	0
Address: $00007A_H$ (CAN0) $00008A_H$ (CAN1)	RRTR7	RRTR6	RRTR5	RRTR4	RRTR3	RRTR2	RRTR1	RRTR0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

❍ **Conditions for RRTRx = 0**

- Write 0 to RRTRx.
- After a received data frame is stored in the message buffer (x) (at the same time as RCx setting to 1).
- Transmission by the message buffer (x) is completed (TCx of the transmission complete register (TCR) is 1).

Writing 1 to RRTRx is ignored.

1 is read when a Read Modify Write instruction is performed.

Note:

If setting to 1 by completion of the recieve operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.16 Receive Overrun Register (ROVRR)

If RCx of the reception complete register (RCR) is 1 when completing storing of a received message in the message buffer (x), ROVRx becomes 1, indicating that reception has overrun.

■ **Receive Overrun Register (ROVRR)**

	15	14	13	12	11	10	9	8
Address: 00007D$_H$ (CAN0) 00008D$_H$ (CAN1)	ROVR15	ROVR14	ROVR13	ROVR12	ROVR11	ROVR10	ROVR9	ROVR8
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)

	7	6	5	4	3	2	1	0
Address: 00007C$_H$ (CAN0) 00008C$_H$ (CAN1)	ROVR7	ROVR6	ROVR5	ROVR4	ROVR3	ROVR2	ROVR1	ROVR0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

Writing 0 to ROVRx results in ROVRx = 0. Writing 1 to ROVRx is ignored. After checking that reception has overrun, write 0 to ROVRx to set it to 0.

1 is read when a Read Modify Write instruction is performed.

Note:

If setting to 1 by completion of the recieve operation and clearing to 0 by writing occur at the same time, the bit is set to 1.

19.6.17 Reception Interrupt Enable Register (RIER)

Reception interrupt enable register (RIER) enables or disables the reception interrupt by the message buffer (x).
The reception interrupt is generated at reception completion (when RCx of the reception completion register (RCR) is 1).

■ Reception Interrupt Enable Register (RIER)

	15	14	13	12	11	10	9	8
Address: 00007F$_H$ (CAN0) 00008F$_H$ (CAN1)	RIE15	RIE14	RIE13	RIE12	RIE11	RIE10	RIE9	RIE8
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)

	7	6	5	4	3	2	1	0
Address: 00007E$_H$ (CAN0) 00008E$_H$ (CAN1)	RIE7	RIE6	RIE5	RIE4	RIE3	RIE2	RIE1	RIE0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

0: Reception interrupt disabled

1: Reception interrupt enabled

19.6.18 Acceptance Mask Select Register (AMSR)

This register selects masks (acceptance mask) for comparison between the received message ID's and the message buffer ID's.

■ Acceptance Mask Select Register (AMSR)

BYTE0

	7	6	5	4	3	2	1	0
Address: 001C10$_H$ (CAN0) 001D10$_H$ (CAN1)	AMS3.1	AMS3.0	AMS2.1	AMS2.0	AMS1.1	AMS1.0	AMS0.1	AMS0.0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE1

	15	14	13	12	11	10	9	8
Address: 001C11$_H$ (CAN0) 001D11$_H$ (CAN1)	AMS7.1	AMS7.0	AMS6.1	AMS6.0	AMS5.1	AMS5.0	AMS4.1	AMS4.0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE2

	7	6	5	4	3	2	1	0
Address: 001C12$_H$ (CAN0) 001D12$_H$ (CAN1)	AMS11.1	AMS11.0	AMS10.1	AMS10.0	AMS9.1	AMS9.0	AMS8.1	AMS8.0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE3

	15	14	13	12	11	10	9	8
Address: 001C13$_H$ (CAN0) 001D13$_H$ (CAN1)	AMS15.1	AMS15.0	AMS14.1	AMS14.0	AMS13.1	AMS13.0	AMS12.1	AMS12.0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

Table 19.6-2 Selection of Acceptance Mask

AMSx.1	AMSx.0	Acceptance Mask
0	0	Full-bit comparison
0	1	Full-bit mask
1	0	Acceptance mask register 0 (AMR0)

Table 19.6-2 Selection of Acceptance Mask (Continued)

AMSx.1	AMSx.0	Acceptance Mask
1	1	Acceptance mask register 1 (AMR1)

Note:

AMSx.1 and AMSx.0 should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored

19.6.19 Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)

There are two acceptance mask registers, AMR0 and AMR1, both of which are available either in the standard frame format or extended frame format.
AM28 to AM18 (11 bits) are used for acceptance masks in the standard frame format and AM28 to AM0 (29 bits) are used for acceptance masks in the extended format.

■ Acceptance Mask Registers 0 and 1 (AMR0 and AMR1)

AMR0 BYTE0	7	6	5	4	3	2	1	0
Address: $001C14_H$ (CAN0) $001D14_H$ (CAN1)	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR0 BYTE1	15	14	13	12	11	10	9	8
Address: $001C15_H$ (CAN0) $001D15_H$ (CAN1)	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR0 BYTE2	7	6	5	4	3	2	1	0
Address: $001C16_H$ (CAN0) $001D16_H$ (CAN1)	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR0 BYTE3	15	14	13	12	11	10	9	8
Address: $001C17_H$ (CAN0) $001D17_H$ (CAN1)	AM4	AM3	AM2	AM1	AM0	—	—	—
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

AMR1 BYTE0

	7	6	5	4	3	2	1	0
Address: 001C18$_H$ (CAN0) 001D18$_H$ (CAN1)	AM28	AM27	AM26	AM25	AM24	AM23	AM22	AM21
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR1 BYTE1

	15	14	13	12	11	10	9	8
Address: 001C19$_H$ (CAN0) 001D19$_H$ (CAN1)	AM20	AM19	AM18	AM17	AM16	AM15	AM14	AM13
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR1 BYTE2

	7	6	5	4	3	2	1	0
Address: 001C1A$_H$ (CAN0) 001D1A$_H$ (CAN1)	AM12	AM11	AM10	AM9	AM8	AM7	AM6	AM5
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

AMR1 BYTE3

	15	14	13	12	11	10	9	8
Address: 001C1B$_H$ (CAN0) 001D1B$_H$ (CAN1)	AM4	AM3	AM2	AM1	AM0	—	—	—
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

❍ **0: Compare**

Compare the bit of the acceptance code (ID register IDRx for comparing with the received message ID) corresponding to this bit with the bit of the received message ID. If there is no match, no message is received.

❍ **1: Mask**

Mask the bit of the acceptance code ID register (IDRx) corresponding to this bit. No comparison is made with the bit of the received message ID.

Note:

AMR0 and AMR1 should be set when all the message buffers (x) selecting AMR0 and AMR1 are invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffers are valid (BVALx = 1) may cause unnecessary received messages to be stored.

19.6.20 Message Buffers

There are 16 message buffers. Message buffer x (x = 0 to 15) consists of an ID register (IDRx), DLC register (DLCRx), and data register (DTRx).

■ Message Buffers

- ❍ The message buffer (x) is used both for transmission and reception.
- ❍ The lower-numbered message buffers are assigned higher priority.
  - At transmission, when a request for transmission is made to more than one message buffer, transmission is performed, starting with the lowest-numbered message buffer (See 19.7).
  - At reception, when the received message ID passes through the acceptance filter (mechanism for comparing the acceptance-masked ID of received message and message buffer) of more than one message buffer, the received message is stored in the lowest-numbered message buffer (See 19.8).
- ❍ When the same acceptance filter is set in more than one message buffer, the message buffers can be used as a multi-level message buffer. This provides allowance for receiving time (See 19.12).

Note:

A write operation to message buffers and general-purpose RAM areas should be performed in words to even addresses only. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

When the BVALx bit of the message buffer valid register (BVALR) is 0 (Invalid), the message buffers x (IDRx, DLCRx, and DTRx) can be used as general-purpose RAM.

During the receive/transmit operation of the CAN controller, the CAN Controller write/read to/from the message buffers. If the CPU tries to write/read to/from the message buffers in this period, the CPU has to wait a maximum time of 64 machine cycles.

This is also true for the general-purpose RAM.

19.6.21 ID Register x (x = 0 to 15) (IDRx)

IDRegister x (x = 0 to 15) (IDRx) is the ID register for message buffer (x).

■ **ID Register x (x = 0 to 15) (IDRx)**

BYTE0	7	6	5	4	3	2	1	0
Address: $001A20_H$ + 4**x** (CAN0) $001B20_H$ + 4**x** (CAN1)	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE1	15	14	13	12	11	10	9	8
Address: $001A21_H$ + 4**x** (CAN0) $001B21_H$ + 4**x** (CAN1)	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE2	7	6	5	4	3	2	1	0
Address: $001A22_H$ + 4**x** (CAN0) $001B22_H$ + 4**x** (CAN1)	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE3	15	14	13	12	11	10	9	8
Address: $001A23_H$ + 4**x** (CAN0) $001B23_H$ + 4**x** (CAN1)	ID4	ID3	ID2	ID1	ID0	—	—	—
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(—)
Initial value:	(X)	(X)	(X)	(X)	(X)	(—)	(—)	(—)

When using the message buffer (x) in the standard frame format (IDEx of the IDE register (IDER) = 0), use 11 bits of ID28 to ID18. When using the buffer in the extended frame format (IDEx = 1), use 29 bits of ID28 to ID0.

ID28 to ID0 have the following functions:

- Set acceptance code (ID for comparing with the received message ID).
- Set transmitted message ID.

Note: In the standard frame format, setting 1s to all bits of ID28 to ID22 is prohibited).

- Store the received message ID.

Note: All received message ID bits are stored (even if bits are masked). In the standard frame

format, ID17 to ID0 stores image of old message left in the receive shift register.

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

This register should be set when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

19.6.22 DLC Register x (x = 0 to 15) (DLCRx)

DLC Register x (x = 0 to 15) (DLCRx) is the DLC register for message buffer x.

■ DLC Register x (x = 0 to 15) (DLCRx)

	7	6	5	4	3	2	1	0
Address: $001A60_H$ + 2**x** (CAN0) $001B60_H$ + 2**x** (CAN1)	—	—	—	—	DLC3	DLC2	DLC1	DLC0
Read/write:	(—)	(—)	(—)	(—)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(—)	(—)	(—)	(—)	(X)	(X)	(X)	(X)

❍ Transmission

- Set the data length (byte count) of a transmitted message when a data frame is transmitted (TRTRx of the transmitting RTR register (TRTRR) is 0).
- Set the data length (byte count) of a requested message when a remote frame is transmitted (TRTRx = 1).

Note:

Setting other than 0000 to 1000 (0 to 8 bytes) is prohibited.

❍ Reception

- Store the data length (byte count) of a received message when a data frame is received (RRTRx of the remote frame request receiving register (RRTRR) is 0).
- Store the data length (byte count) of a requested message when a remote frame is received (RRTRx = 1).

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

19.6.23 Data Register x (x = 0 to 15) (DTRx)

Data register x (x = 0 to 15) (DTRx) is the data register for message buffer (x). This register is used only in transmitting and receiving a data frame but not in transmitting and receiving a remote frame.

■ **Data Register x (x = 0 to 15) (DTRx)**

BYTE0

	7	6	5	4	3	2	1	0
Address: $001A80_H$ + 8**x** (CAN0) $001B80_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE1

	15	14	13	12	11	10	9	8
Address: $001A81_H$ + 8**x** (CAN0) $001B81_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE2

	7	6	5	4	3	2	1	0
Address: $001A82_H$ + 8**x** (CAN0) $001B82_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE3

	15	14	13	12	11	10	9	8
Address: $001A83_H$ + 8**x** (CAN0) $001B83_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE4

	7	6	5	4	3	2	1	0
Address: $001A84_H$ + 8**x** (CAN0) $001B84_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE5

	15	14	13	12	11	10	9	8
Address: $001A85_H$ + 8**x** (CAN0) $001B85_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE6

	7	6	5	4	3	2	1	0
Address: $001A86_H$ + 8**x** (CAN0) $001B86_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

BYTE7

	15	14	13	12	11	10	9	8
Address: $001A87_H$ + 8**x** (CAN0) $001B87_H$ + 8**x** (CAN1)	D7	D6	D5	D4	D3	D2	D1	D0
Read/write:	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value:	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

❍ **Sets transmitted message data (any of 0 to 8 bytes).**

Data is transmitted in the order of BYTE0, BYTE1, ..., BYTE7, starting with the MSB.

❍ **Stores received message data.**

Data is stored in the order of BYTE0, BYTE1, ..., BYTE7, starting with the MSB.

Even if the received message data is less than 8 bytes, the remaining bytes of the data register (DTRx), to which data are stored, are undefined.

Note:

A write operation to this register should be performed in words. A write operation in bytes causes undefined data to be written to the upper byte at writing to the lower byte. Writing to the upper byte is ignored.

19.7 Transmission of CAN Controller

When 1 is written to TREQx of the transmission request register (TREQR), transmission by the message buffer (x) starts. At this time, TREQx becomes 1 and TCx of the transmission complete register (TCR) becomes 0.

■ Starting Transmission of the CAN Controller

If RFWTx of the remote frame receiving wait register (RFWTR) is 0, transmission starts immediately. If RFWTx is 1, transmission starts after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR) becomes 1).

If a request for transmission is made to more than one message buffer (more than one TREQx is 1), transmission is performed, starting with the lowest-numbered message buffer.

Message transmission to the CAN bus (by the transmit output pin TX) starts when the bus is idle.

If TRTRx of the transmission RTR register (TRTRR) is 0, a data frame is transmitted. If TRTRx is 1, a remote frame is transmitted.

If the message buffer competes with other CAN controllers on the CAN bus for transmission and arbitration fails, or if an error occurs during transmission, the message buffer waits until the bus is idle and repeats retransmission until it is successful.

■ Canceling a Transmission Request from the CAN Controller

❍ Canceling by transmission cancel register (TCANR)

A transmission request for message buffer (x) having not executed transmission during transmission pending can be canceled by writing 1 to TCANx of the transmission cancel register (TCANR). At completion of cancellation, TREQx becomes 0.

❍ Canceling by storing received message

The message buffer (x) having not executed transmission despite transmission request also performs reception.

If the message buffer (x) has not executed transmission despite a request for transmission of a data frame (TRTRx = 0 or TREQx = 1), the transmission request is canceled after storing received data frames passing through the acceptance filter (TREQx = 0).

Note:

A transmission request is not canceled by storing remote frames (TREQx = 1 remains unchanged).

If the message buffer (x) has not executed transmission despite a request for transmission of a remote frame (TRTRx = 1 or TREQx = 1), the transmission request is canceled after storing received remote frames passing through the acceptance filter (TREQx = 0).

Note:

The transmission request is canceled by storing either data frames or remote frames.

■ Completing Transmission of the CAN Controller

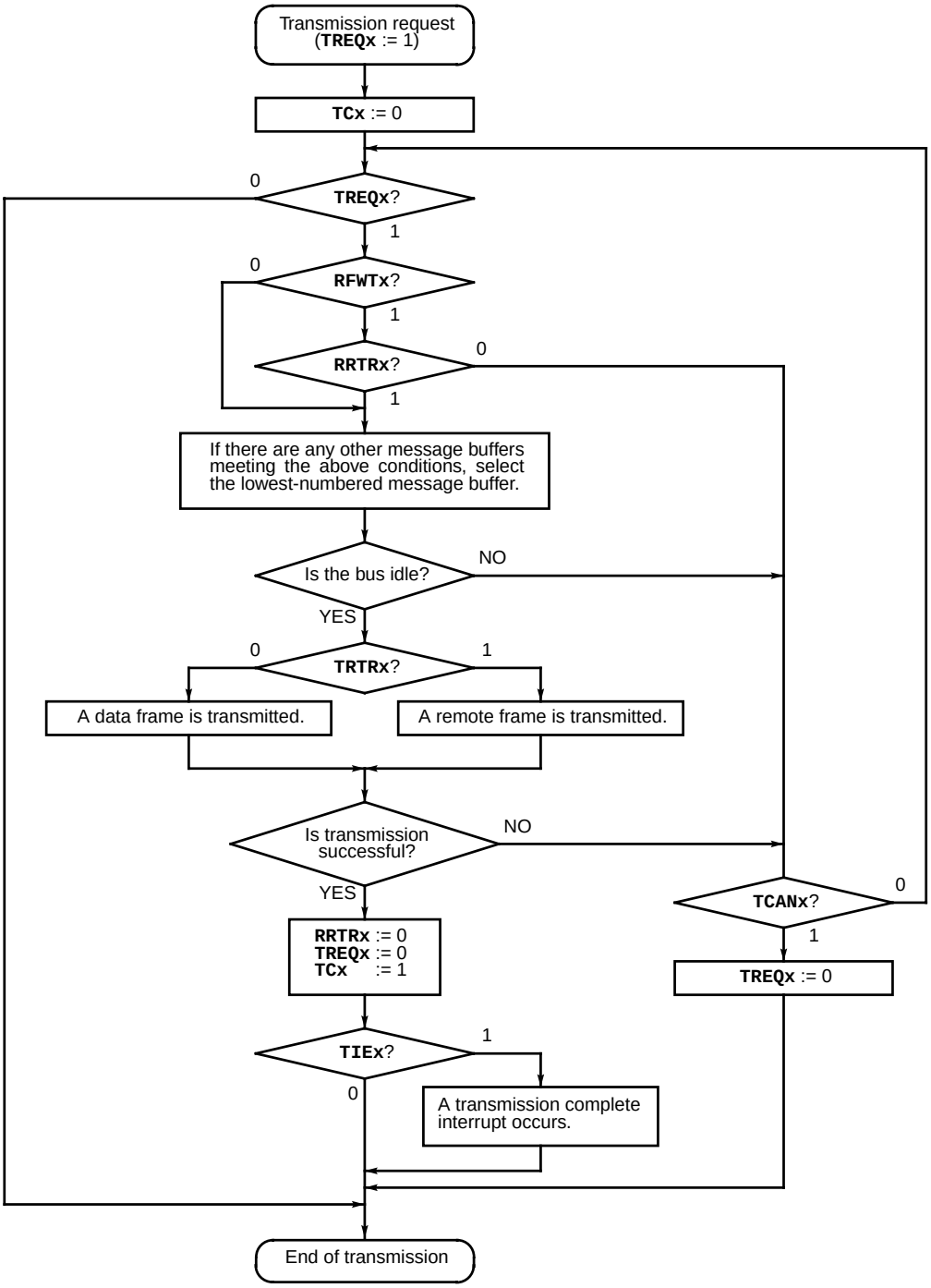
When transmission is successful, RRTRx becomes 0, TREQx becomes 0, and TCx of the transmission complete register (TCR) becomes 1.

If the transmission complete interrupt is enabled (TIEx of the transmission complete interrupt enable register (TIER) is 1), an interrupt occurs.

■ Transmission Flowchart of the CAN Controller

Figure 19.7-1 shows a transmission flowchart of the CAN controller.

Figure 19.7-1 Transmission Flowchart of the CAN Controller



19.8 Reception of CAN Controller

Reception starts when the start of data frame or remote frame (SOF) is detected on the CAN bus.

■ Acceptance Filtering

The received message in the standard frame format is compared with the message buffer (x) set in the standard frame format (IDEx of the IDE register (IDER) is 0). The received message in the extended frame format is compared with the message buffer (x) set (IDEx is 1) in the extended frame format.

If all the bits set to Compare by the acceptance mask agree after comparison between the received message ID and acceptance code (ID register (IDRx) for comparing with the received message ID), the received message passes to the acceptance filter of the message buffer (x).

■ Storing Received Message

When the receive operation is successful, received messages are stored in a message buffer x including IDs passed through the acceptance filter.

When receiving data frames, received messages are stored in the ID register (IDRx), DLC register (DLCRx), and data register (DTRx).

Even if received message data is less than 8 bytes, some data is stored in the remaining bytes of the DTRx and its value is undefined.

When receiving remote frames, received messages are stored only in the IDRx and DLCRx, and the DTRx remains unchanged.

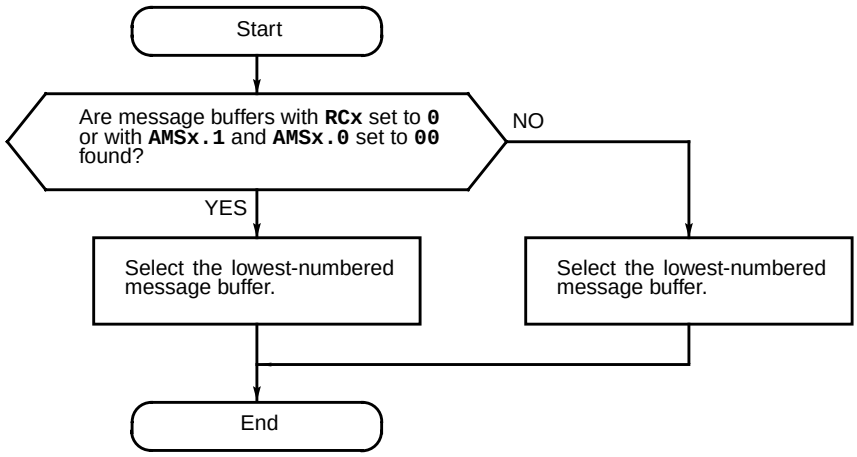
If there is more than one message buffer including IDs passed through the acceptance filter, the message buffer x in which received messages are to be stored is determined according to the following rules.

- The order of priority of the message buffer x (x = 0 to 15) rises as its number lower; in other words, message buffer 0 is given the highest and the message buffer 15 is given the lowest priority.
- Basically, message buffers with the RCx bit of 0 in the receive completion register (RCR) are preferred in storing received messages.
- If the bits of the acceptance mask select register (AMSR) are set to All Bits Compare (for message buffers with the AMSx.1 and AMSx.0 bits set to 00), received messages are stored irrespective of the value of the RCx bit of the RCR.
- If there are message buffers with the RCx bit of the RCR set to 0, or with the bits of the AMSR set to All Bits Compare, received messages are stored in the lowest-number (highest-priority) message buffer x.
- If there are no message buffers above-mentioned, received messages are stored in a lower-number message buffer x.
- Message buffers should be arranged in ascending numeric order. The lowest message buffers should be with All Bits Compare, then AMR0 or AMR1 masks. And The highest message buffers should be with All Bits Mask.

Figure 19.8-1 shows a flowchart for determining the message buffer (x) where received messages are to be stored. It is recommended that message buffers be arranged in the

following order: message buffers in which each AMSR bit is set to All Bits Compare, message buffers using AMR0 or AMR1, and message buffers in which each AMSR bit is set to All Bits Mask.

Figure 19.8-1 Flowchart Determining Message Buffer (x) where Received Messages Stored



■ Receive Overrun

When a message is stored in the message buffer with the corresponding RCx being already set to 1, it will results in receive overrun. In this case, the corresponding ROVRx bit in the receive overrun register ROVRR is set to 1.

■ Processing for Reception of Data Frame and Remote Frame

❍ Processing for reception of data frame

RRTRx of the remote request receiving register (RRTRR) becomes 0.

TREQx of the transmission request register (TREQR) becomes 0 (immediately before storing the received message). A transmission request for message buffer (x) having not executed transmission will be canceled.

Note:

A request for transmission of either a data frame or remote frame is canceled.

❍ Processing for reception of remote frame

RRTRx becomes 1.

If TRTRx of the transmitting RTR register (TRTRR) is 1, TREQx becomes 0. As a result, the request for transmitting remote frame to message buffer having not executed transmission will be canceled.

Note:

A request for data frame transmission is not canceled.

For cancellation of a transmission request, see Figure 19.7.

■ Completing Reception

RCx of the reception complete register (RCR) becomes 1 after storing the received message.

If a reception interrupt is enabled (RIEx of the reception interrupt enable register (RIER) is 1), an

interrupt occurs.

Note:

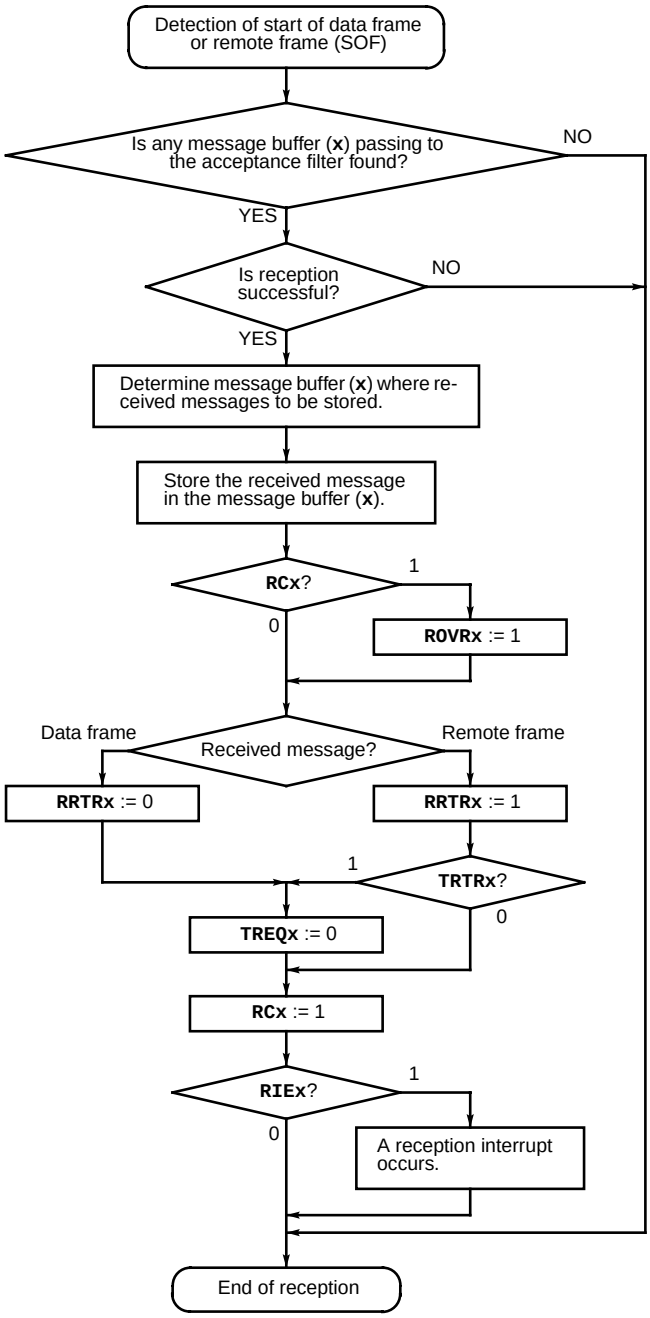
This CAN controller will not receive any messages transmitted by itself.

19.9 Reception Flowchart of CAN Controller

Figure 19.9-1 shows a reception flowchart of the CAN controller.

■ Reception Flowchart of the CAN Controller

Figure 19.9-1 Reception Flowchart of the CAN Controller



19.10 How to Use the CAN Controller

The following settings are required to use the CAN controller:

- **Bit timing**
- **Frame format**
- **ID**
- **Acceptance filter**
- **Low-power consumption mode**

■ Setting Bit Timing

The bit timing register (BTR) should be set during bus operation stop (when the bus operation stop bit (HALT) of the control status register (CSR) is 1).

After the setting completion, write 0 to HALT to cancel bus operation stop.

■ Setting Frame Format

Set the frame format used by the message buffer (x). When using the standard frame format, set IDEx of the IDE register (IDER) to 0. When using the extended frame format, set IDEx to 1.

This setting should be made when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

■ Setting ID

Set the message buffer (x) ID to ID28 to ID0 of ID register (IDRx). The message buffer (x) ID need not be set to ID11 to ID0 in the standard frame format. The message buffer (x) ID is used as a transmission message at transmission and is used as an acceptance code at reception.

This setting should be made when the message buffer (x) is invalid (BVALx of the message buffer valid register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

■ Setting Acceptance Filter

The acceptance filter of the message buffer (x) is set by an acceptance code and acceptance mask set. It should be set when the acceptance message buffer (x) is invalid (BVALx of the message buffer enable register (BVALR) is 0). Setting when the buffer is valid (BVALx = 1) may cause unnecessary received messages to be stored.

Set the acceptance mask used in each message buffer (x) by the acceptance mask select register (AMSR). The acceptance mask registers (AMR0 and AMR1) should also be set if used (For the setting details, see 19.6.18 and 19.6.19).

The acceptance mask should be set so that a transmission request may not be canceled when unnecessary received messages are stored. For example, it should be set to a full-bit comparison if only one specific ID is used for the transmission.

■ Setting Low-power Consumption Mode

To set the F^2MC-16LX in a low-power consumption mode (Stop, Watch, Hardware Standby,

etc.), write 1 to the bus operation stop bit (HALT) of the control status register (CSR), and then check that the bus operation has stopped (HALT = 1).

19.11 Procedure for Transmission by Message Buffer (x)

After setting the bit timing, frame format, ID, and acceptance filter, set BVALx to 1 to activate the message buffer (x).

■ Procedure for Transmission by Message Buffer (x)

❍ Setting transmit data length code

Set the transmit data length code (byte count) to DLC3 to DLC0 of the DLC register (DLCRx).

For data frame transmission (when TRTRx of the transmission RTR register (TRTRR) is 0), set the data length of the transmitted message.

For remote frame transmission (when TRTRx = 1), set the data length (byte count) of the requested message.

Note:

Setting other than 0000 to 1000 (0 to 8 bytes) is prohibited.

❍ Setting transmit data (only for transmission of data frame)

For data frame transmission (when TRTRx of the transmission register (TRTRR) is 0), set data as the count of byte transmitted in the data register (DTRx).

Note:

Transmit data should be rewritten while the TREQx bit of the transmission request register (TREQR) set to 0. There is no need for setting the BVALx bit of the message buffer valid register (BVALR) to 0. Setting the BVALx bit to 0 may cause incoming remote frame to be lost.

❍ Setting transmission RTR register

For data frame transmission, set TRTRx of the transmission RTR register (TRTRR) to 0.

For remote frame transmission, set TRTRx to 1.

❍ Setting conditions for starting transmission (only for transmission of data frame)

Set RFWTx of the remote frame receiving wait register (RFWTR) to 0 to start transmission immediately after a request for data frame transmission is set (TREQx of the transmission request register (TREQR) is 1 and TRTRx of the transmission RTR register (TRTRR) is 0).

Set RFWTx to 1 to start transmission after waiting until a remote frame is received (RRTRx of the remote request receiving register (RRTRR) becomes 1) after a request for data frame transmission is set (TREQx = 1 and TRTRx = 0).

Note:

Remote frame transmission can not be made, if RFWTx is set to 1.

❍ Setting transmission complete interrupt

When generating a transmission complete interrupt, set TIEx of the transmission complete interrupt enable register (TIER) to 1.

When not generating a transmission complete interrupt, set TIEx to 0.

❍ **Setting transmission request**

For a transmission request, set TREQx of the transmission request register (TREQR) to 1.

❍ **Canceling transmission request**

When canceling a pending request for transmission to the message buffer (x), write 1 to TCANx of the transmission cancel register (TCANR).

Check TREQx. For TREQx = 0, transmission cancellation is terminated or transmission is completed. Check TCx of the transmission complete register (TCR). For TCx = 0, transmission cancellation is terminated. For TCx = 1, transmission is completed.

❍ **Processing for completion of transmission**

If transmission is successful, TCx of the transmission complete register (TCR) becomes 1.

If the transmission complete interrupt is enabled (TIEx of the transmission complete interrupt enable register (TIER) is 1), an interrupt occurs.

After checking the transmission completion, write 0 to TCx to set it to 0. This cancels the transmission complete interrupt.

In the following cases, the pending transmission request is canceled by receiving and storing a message.

- Request for data frame transmission by reception of data frame
- Request for remote frame transmission by reception of data frame
- Request for remote frame transmission by reception of remote frame

Request for data frame transmission is not canceled by receiving and storing a remote frame. ID and DLC, however, are changed by the ID and DLC of the received remote frame. Note that the ID and DLC of data frame to be transmitted become the value of received remote frame.

19.12 Procedure for Reception by Message Buffer (x)

After setting the bit timing, frame format, ID, and acceptance filter, make the settings described below.

■ Procedure for Reception by Message Buffer (x)

❍ Setting reception interrupt

To enable reception interrupt, set RIEx of the reception interrupt enable register (RIER) to 1.

To disable reception interrupt, set RIEx to 0.

❍ Starting reception

When starting reception after setting, set BVALx of the message buffer valid register (BVALR) to 1 to make the message buffer (x) valid.

❍ Processing for reception completion

If reception is successful after passing to the acceptance filter, the received message is stored in the message buffer (x) and RCx of the reception complete register (RCR) becomes 1. For data frame reception, RRTRx of the remote request receiving register (RRTRR) becomes 0. For remote frame reception, RRTRx becomes 1.

If a reception interrupt is enabled (RIEx of the reception interrupt enable register (RIER) is 1), an interrupt occurs.

After checking the reception completion (RCx = 1), process the received message.

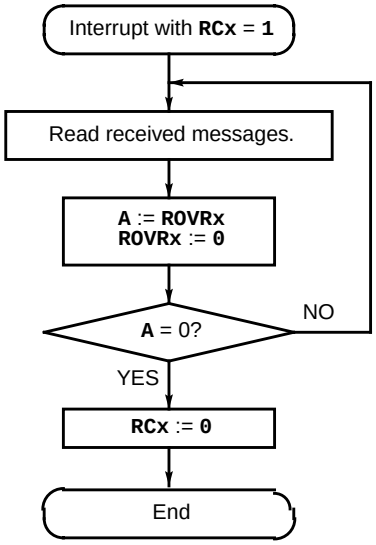
After completion of processing the received message, check ROVRx of the reception overrun register (ROVRR).

If ROVRx = 0, the processed received message is valid. Write 0 to RCRx to set it to 0 (the reception complete interrupt is also canceled) to terminate reception.

If ROVRx = 1, a reception overrun occurred and the next message may have overwritten the processed message. In this case, received messages should be processed again after setting the ROVRx bit to 0 by writing 0 to it.

Figure 19.12-1 shows an example of receive interrupt handling.

Figure 19.12-1 Example of Receive Interrupt Handling



19.13 Setting Configuration of Multi-level Message Buffer

If the receptions are performed frequently, or if several different ID's of messages are received, in other words, if there is insufficient time for handling messages, more than one message buffer can be combined into a multi-level message buffer to provide allowance for processing time of the received message by CPU.

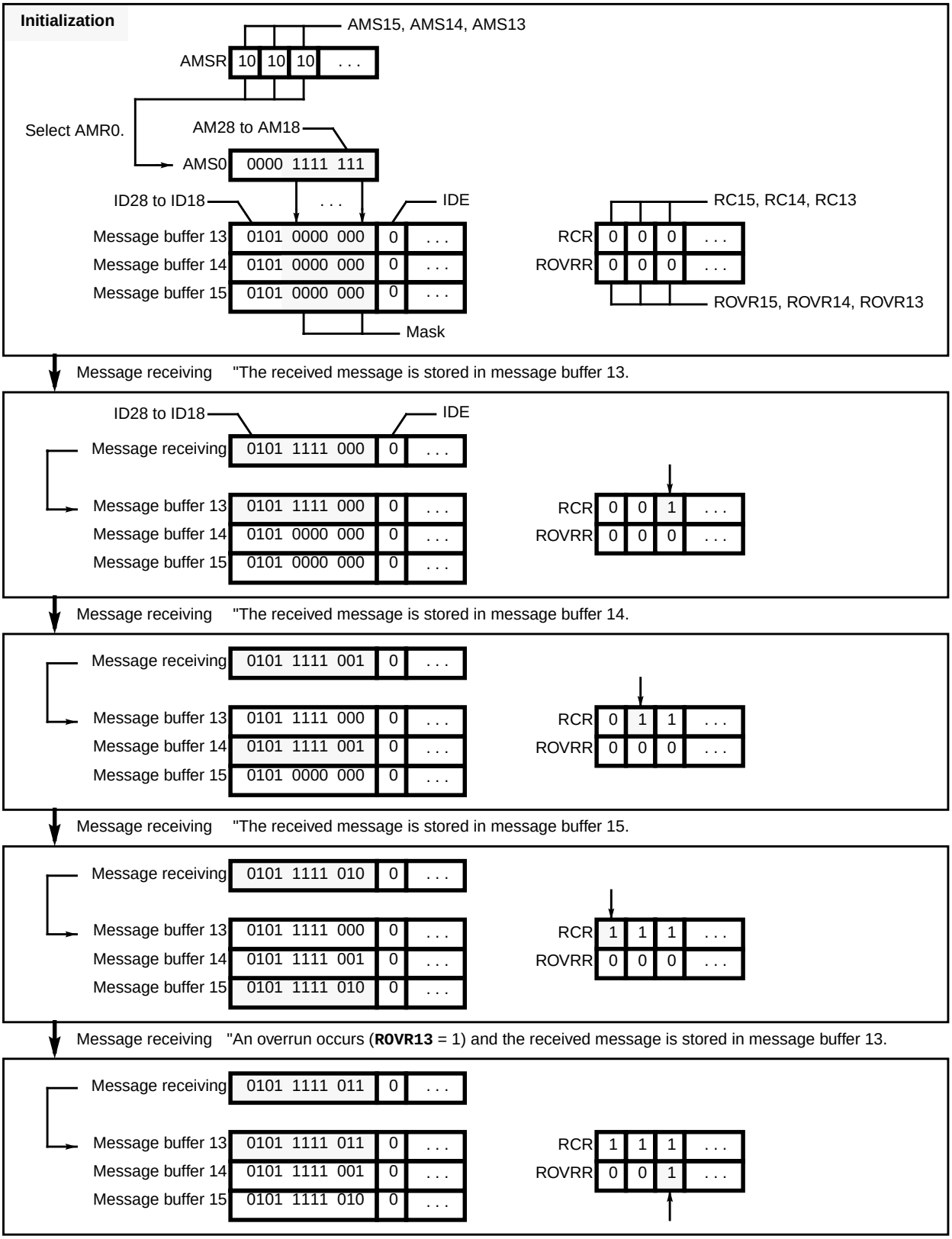
■ Setting Configuration of Multi-level Message Buffer

To provide a multi-level message buffer, the same acceptance filter must be set in the combined message buffers.

If the bits of the acceptance mask select register (AMSR) are set to All Bits Compare ((AMSx.1, AMSx.0) = (0, 0)), multi-level message configuration of message buffers is not allowed. This is because All Bits Compare causes received messages to be stored irrespective of the value of the RCx bit of the receive completion register (RCR), so received messages are always stored in lower-numbered (lower-priority) message buffers even if All Bits Compare and identical acceptance code (ID register (IDRx)) are specified for more than one message buffer. Therefore, All Bits Compare and identical acceptance code should not be specified for more than one message buffer.

Figure 19.13-1 shows operational examples of multi-level message buffers.

Figure 19.13-1 Examples of Operation of Multi-level Message Buffer



Note:

Four messages are received with the same acceptance filter set in message buffers 13, 14 and 15.

CHAPTER 20 STEPPING MOTOR CONTROLLER

This chapter explains the functions and operations of the stepping motor controller.

20.1 Outline of Stepping Motor Controller

20.2 Stepping Motor Controller Registers

20.3 PWM 1&2 Select

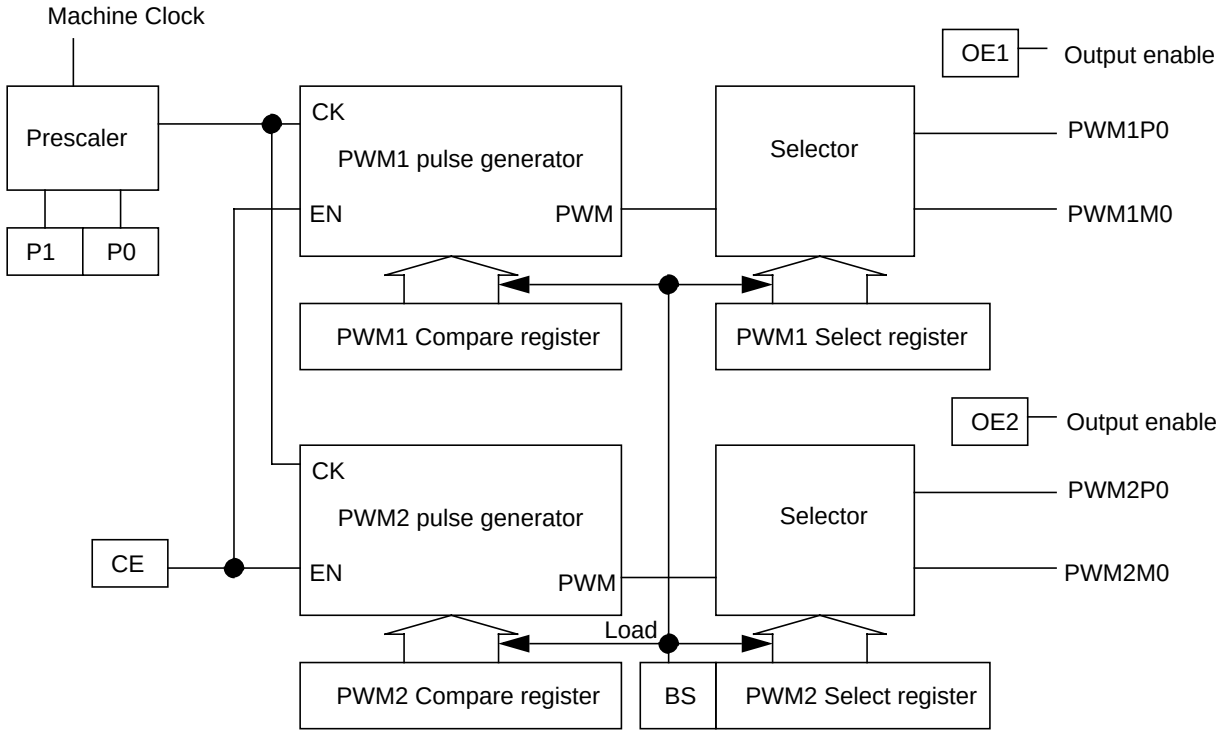
20.1 Outline of Stepping Motor Controller

The Stepping Motor Controller consists of two PWM Pulse Generators, four motor drivers, drivers and Selector Logic.
The four motor drivers have high output drive capabilities and they can be directly connected to the four ends of two motor coils. The combination of the PWM Pulse Generators and Selector Logic is designed to control the rotation of the motor. A synchronization mechanism assures the synchronous operations of the two PWMs. The following sections describe the Stepping Motor Controller 0 only. The other controllers have the same function. The register addresses are found in the I/O map.

■ Block Diagram of Stepping Motor Controller

Figure 20.1-1 shows a block diagram of the stepping motor controller.

Figure 20.1-1 Block Diagram of Stepping Motor Controllerr

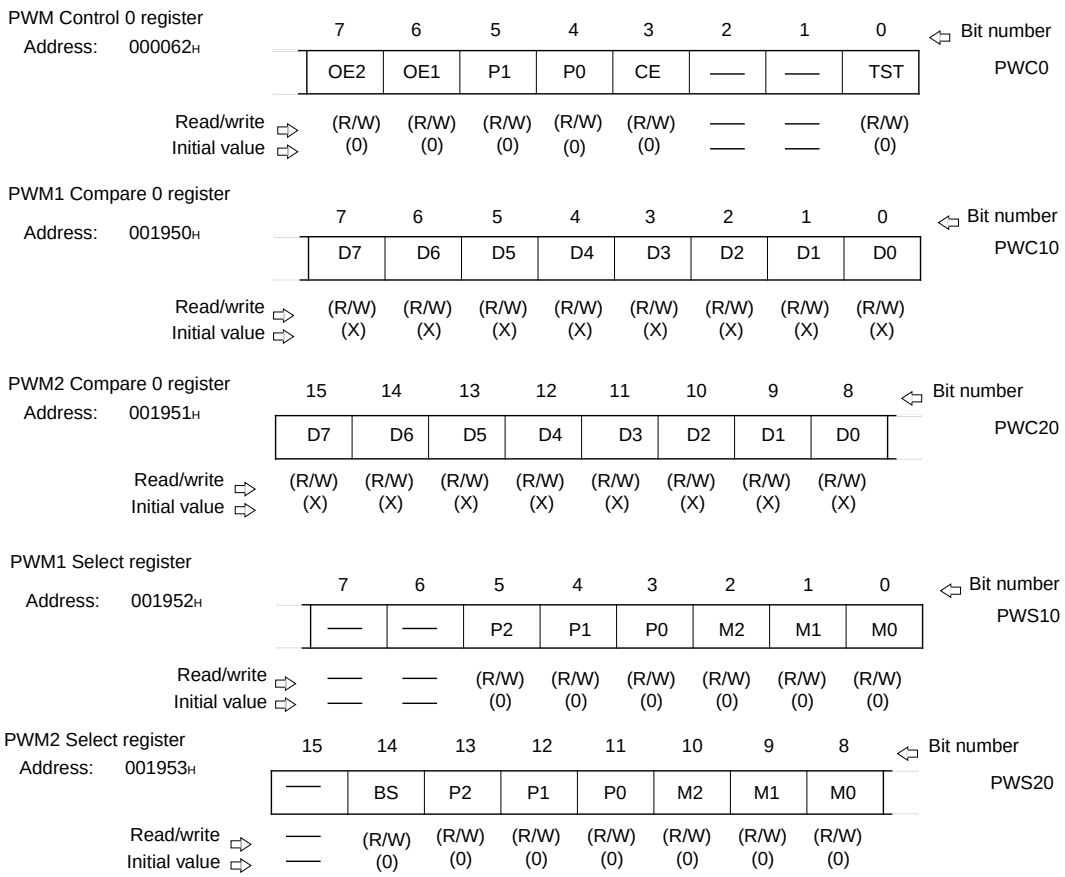


20.2 Stepping Motor Controller Registers

The stepping motor controller has the following five types of registers:

- **PWM control 0 register (PWMC0)**
- **PWM1 compare 0 register (PWC10)**
- **PWM2 compare 0 register (PWC20)**
- **PWM1 select register (PWS10)**
- **PWM2 select register (PWS20)**

■ Stepping Motor Controller Registers



20.2.1 PWM Control 0 register

The PWM control 0 register starts and stops the stepping motor controller, controls interrupts, and sets the external output pins.

■ PWM Control 0 Register

PWM Control 0 register
Address: 000062H

	7	6	5	4	3	2	1	0	⇦ Bit number
	OE2	OE1	P1	P0	CE	——	——	TST	PWC0
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	——	——	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	——	——	(0)	

[bits 7] OE2 : Output enable bit

When this bit is set to "1", the external pins are assigned as PWM2P0 and PWM2M0 outputs. Otherwise they can be used as general purpose IO.

[bits 6] OE1 : Output enable bit

When this bit is set to "1", the external pins are assigned as PWM1P0 and PWM1M0 outputs. Otherwise they can be used as general purpose IO.

[bits 5 to 4] P1 to P0 : Operation clock select bits

These bits specify the clock input signal for the PWM pulse generators.

P1	P0	Clock input
0	0	Machine clock
0	1	1/2 Machine clock
1	0	1/4 Machine clock
1	1	1/8 Machine clock

[bits 3] CE : Count enable bit

This bit enables the operation of the PWM pulse generators. When it is set to "1", the PWM pulse generators start their operation. Note that the PWM2 pulse generator starts the operation one machine clock cycle after the PWM1 pulse generators is started. This is to help reduce the switching noise from the output drivers.

[bits 0] TST : Test bit

This bit is for the device test. In user applications, it should always be set to "0".

20.2.2 PWM1&2 Compare Registers

The contents of the two 8-bit compare registers determine the widths of PWM pulses. The stored value of "00H" represents the PWM duty of 0% and "FFH" represents the duty of 99.6%.

■ PWM1&2 Compare Registers

PWM1&2 compare registers are accessible at any time, however the modified values are reflected to the pulse width at the end of the current PWM cycle after the BS bit of the PWM2 Select register is set to "1".

PWM1 Compare 0 register

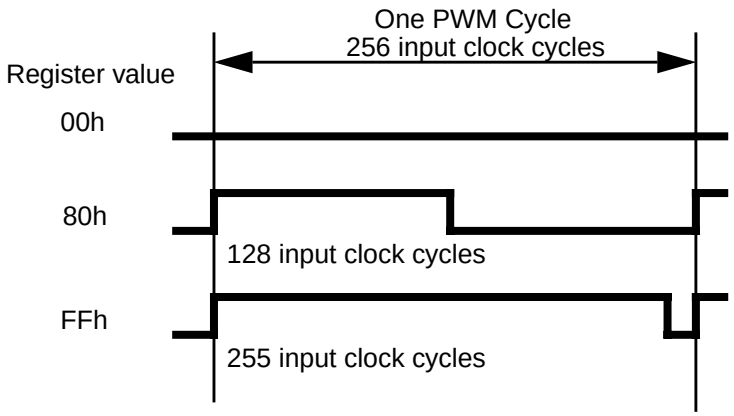
Address: 001950H

	7	6	5	4	3	2	1	0	⇦ Bit number
	D7	D6	D5	D4	D3	D2	D1	D0	PWC10
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)	

PWM2 Compare 0 register

Address: 001951H

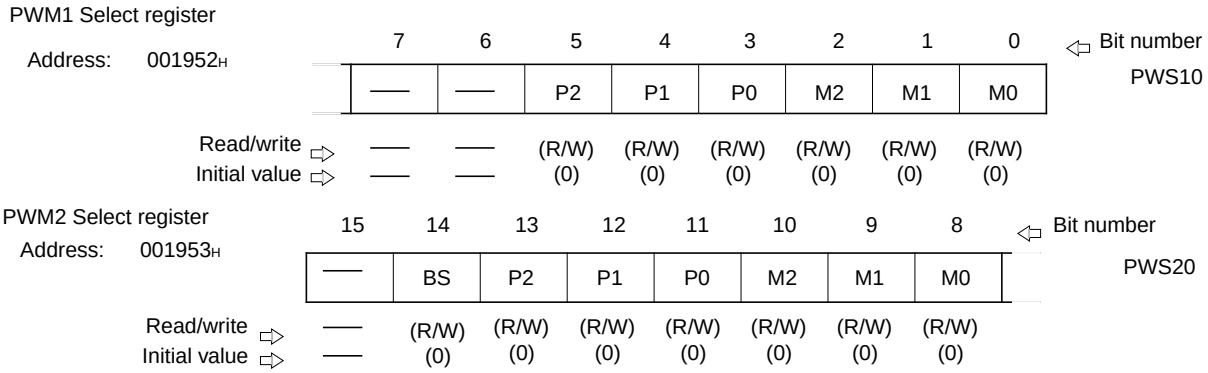
	15	14	13	12	11	10	9	8	⇦ Bit number
	D7	D6	D5	D4	D3	D2	D1	D0	PWC20
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	



20.3 PWM1&2 Select Registers

The PWM1 and PWM2 select registers select 0, 1, the PWM pulse, or high impedance for the external pin output of the stepping motor controller.

■ PWM1&2 Select Registers



[bits 14] BS : Update bit

This bit is prepared to synchronize the settings for the PWM outputs. Any modifications in the two compare registers and two select registers are not reflected to the output signals until this bit is set.

When this bit is set to "1", the PWM pulse generators and selectors load the register contents at the end of the current PWM cycle. The BS bit is reset to "0" automatically at the beginning of the next PWM cycle. If the BS bit is set to "1" by software at the same time as this automatic reset, the BS bit is set to "1" (or remains unchanged) and the automatic reset is cancelled.

[bits 13 to 11] P2 to P0 : Output Select bits

These bits selects the output signal at PWM2P0.

[bits 10 to 8] M2 to M0 : Output Select bits

These bits selects the output signal at PWM2M0.

[bits 5 to 3] P2 to P0 : Output Select bits

These bits selects the output signal at PWM1P0.

[bits 2 to 0] M2 to M0 : Output Select bits

These bits selects the output signal at PWM1M0.

The following table shows the relationship between the output levels and select bits.

P2	P1	P0	PWMnP0	M2	M1	M0	PWMnM0
0	0	0	L	0	0	0	L
0	0	1	H	0	0	1	H

P2	P1	P0	PWMnP0	M2	M1	M0	PWMnM0
0	1	X	PWM pulses	0	1	X	PWM pulses
1	X	X	High impedance	1	X	X	High impedance

CHAPTER 21 SOUND GENERATOR

This chapter explains the functions and operations of the sound generator.

21.1 Outline of Sound Generator

21.2 Sound Generator Registers

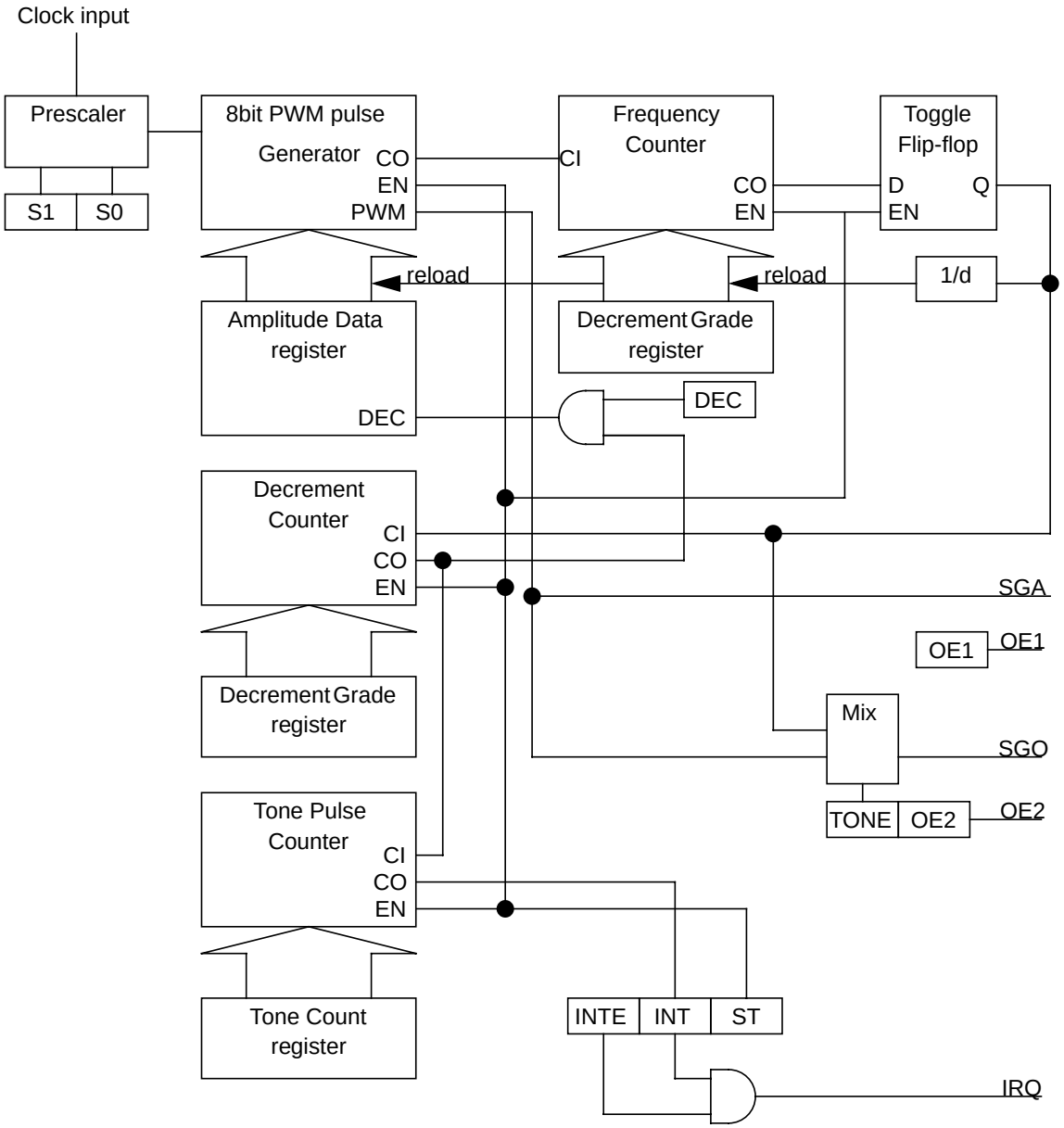
21.1 Outline of Sound Generator

The Sound Generator consists of the Sound Control register, Frequency Data register, Amplitude Data register, Decrement Grade register, Tone Count register, PWM pulse generator, Frequency counter, Decrement counter and Tone Pulse counter.

■ Block Diagram of Sound Generator

Figure 21.1-1 shows a block diagram of the sound generator.

Figure 21.1-1 Block Diagram of Sound Generator

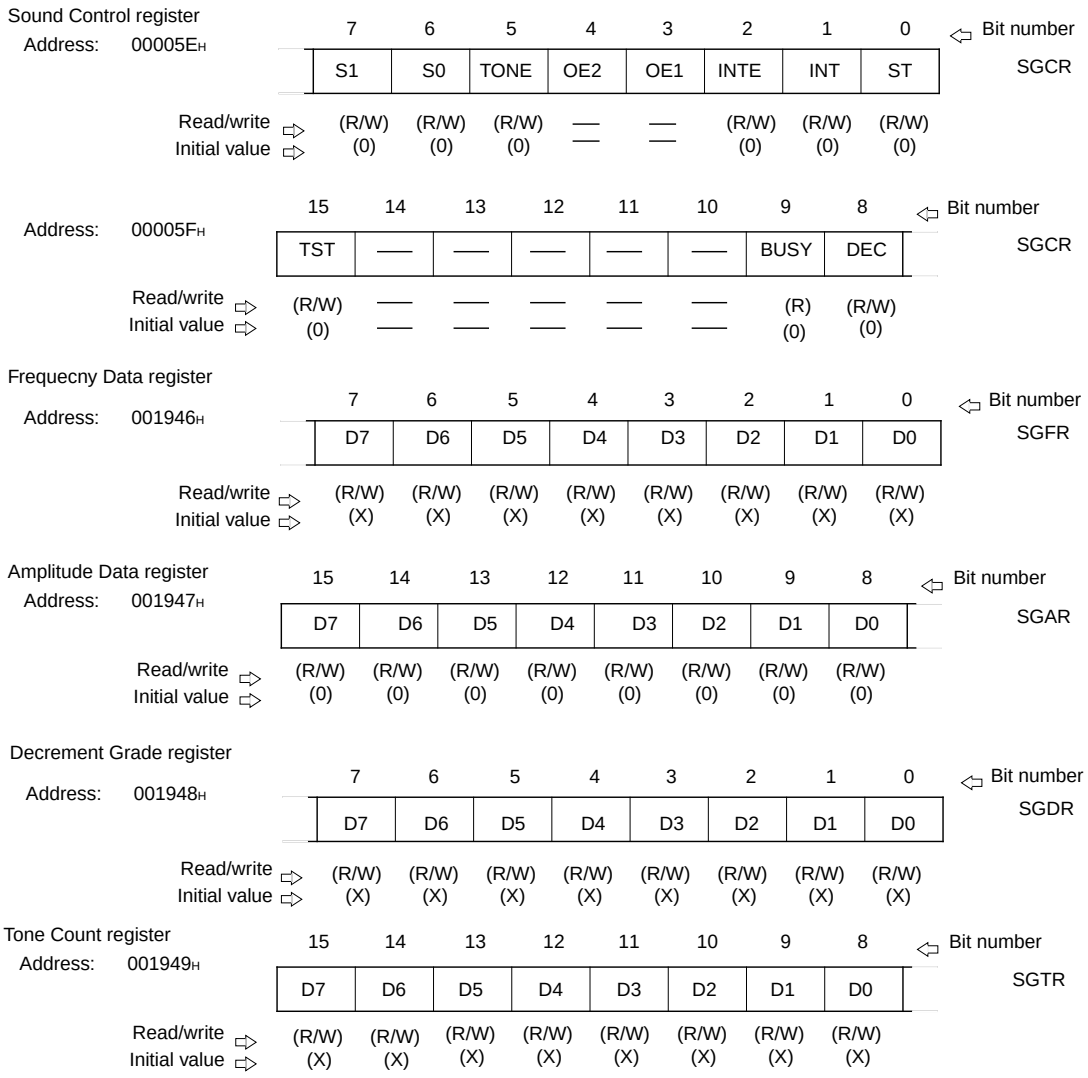


21.2 Sound Generator Registers

The sound generator has the following types of registers:

- **Sound control register (SGCR)**
- **Frequency data register (SGFR)**
- **Amplitude data register (SGAR)**
- **Decrement grade register (SGDR)**
- **Tone count register (SGTR)**

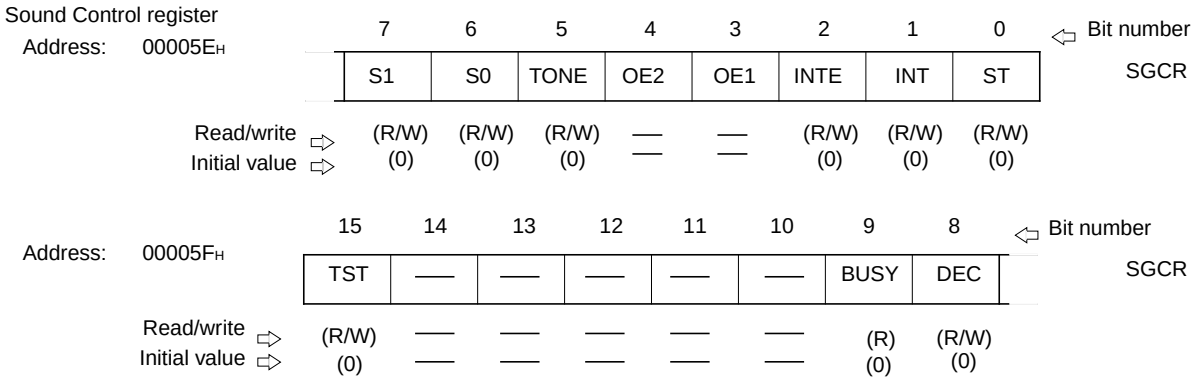
■ Sound Generator Registers



21.2.1 Sound Control Register

The sound control register controls the operation status of the sound generator by controlling interrupts and setting the external output pins.

■ Sound Control Register



[bits 15] TST : Test bit

This bit is prepared for the device test. In any user applications, it should be set to "0".

[bits 9] BUSY : Busy bit

This bit indicates whether the Sound Generator is in operation. This bit is set to "1" upon the ST bit is set to "1". It is reset to "0" when the ST bit is reset to "0" and the operation is completed at the end of one tone cycle. Any write instructions performed on this bit has no effect.

[bits 8] DEC : Auto-decrement enable bit

The DEC bit is prepared for an automatic de-gradation of the sound in conjunction with the Decrement Grade register.

If this bit is set to "1", the stored value in the Amplitude Data register is decremented by 1(one), every time when the Decrement counter counts the number of tone pulses from the toggle flip-flop specified by the Decrement Grade register.

[bits 7 to 6] S1 to S0 : Operation clock select bits

These bits specify the clock input signal for the Sound Generator.

S1	S0	Clock input
0	0	Machine clock
0	1	1/2 Machine clock
1	0	1/4 Machine clock
1	1	1/8 Machine clock

[bits 5] TONE : Tone output bit

When this bit is set to "1", the SGO signal becomes a simple square-waveform (tone pulses) from the toggle flip-flop. Otherwise it is the mixed (AND logic) signal of the tone and PWM pulses.

[bits 4] OE2 : Sound output enable bit

When this bit is set to "1", the external pin is assigned as the SGO output. Otherwise the pin can be used as a general purpose IO. To enable the SGO output, the corresponding bit of the Port Direction register should also be set to "1".

[bits 3] OE1 : Amplitude output enable bit

When this bit is set to "1", the external pin is assigned as the SGA output. Otherwise the pin can be used as a general purpose IO. To enable the SGA output, the corresponding bit of the Port Direction register should also be set to "1".

The SGA signal is the PWM pulses from the PWM pulse generator representing the amplitude of the sound.

[bits 2] INTE : Interrupt enable bit

This bit enables the interrupt signal of the Sound Generator. When this bit is "1" and the INT bit is set to "1", the Sound Generator signals an interrupt.

[bits 1] INT : Interrupt bit

This bit is set to "1" when the Tone Pulse counter counts the number of the tone pulses specified by the Tone Count register and Decrement Grade register.

This bit is reset to "0" by writing "0". Writing "1" has no effect and Read-Modify-Write instructions always result in reading "1".

[bits 0] ST : Start bit

This bit is for starting the operation of the Sound Generator. While this bit is "1", the Sound Generator perform its operation.

When this bit is reset to "0", the Sound Generator stops its operation at the end of the current tone cycle. The BUSY bit indicates whether the Sound Generator is fully stopped.

21.2.2 Frequency Data register

The Frequency Data register stores the reload value for the Frequency counter. The stored value represents the frequency of the sound (or the tone signal from the toggle flip-flop). The register value is reloaded into the counter at every transition of the toggle signal.
The following figure shows the relationship between the tone signal and the register value.

■ **Frequency Data Register**

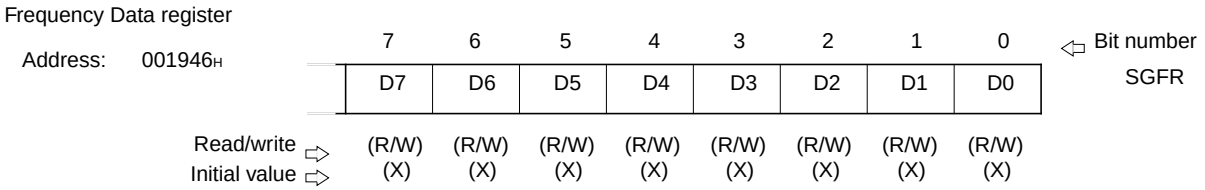
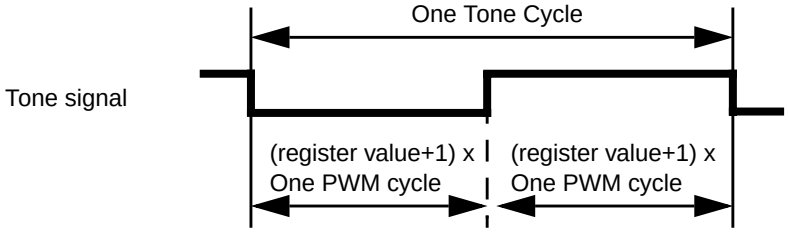


Figure 21.2-1 shows the relationship between a tone signal and a register value.

Figure 21.2-1 Relationship between Tone Signal and Register Value



It should be noted that modifications of the register value while operation may alter the duty cycle of 50% depending on the timing of the modification.

21.2.3 Amplitude Data Register

The Amplitude Data register stores the reload value for the PWM pulse generator. The register value represents the amplitude of the sound. The register value is reloaded into the PWM pulse generator at the end of every tone cycle.

■ **Amplitude Data Register**

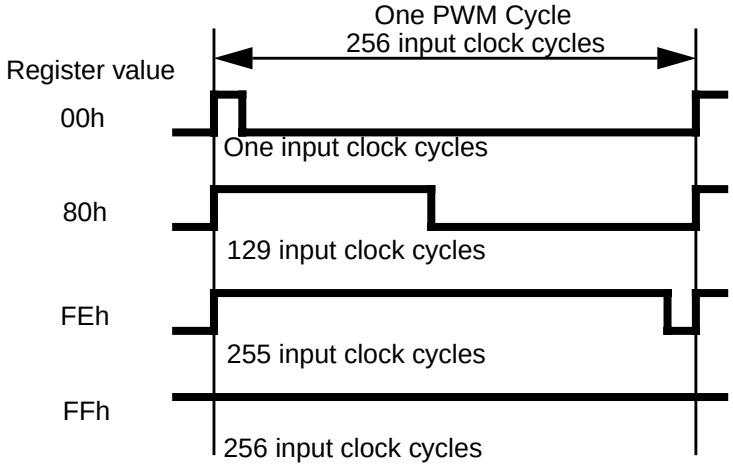
Amplitude Data register
Address: 001947H

Bit number	15	14	13	12	11	10	9	8
SGAR	D7	D6	D5	D4	D3	D2	D1	D0
Read/write	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

When the DEC bit is "1" and the Decrement counter reaches its reload value, this register value is decremented by 1(one). And when the register value reaches "00", further decrements are not performed. However the sound generator continues its operation until the ST bit is cleared.

Figure 12.2-2 shows the relationship between the register value and the PWM pulse.

Figure 21.2-2 Relationship between Register Value and PWM Pulse



When the register value is set to "FF", the PWM signal is always "1".

21.2.4 Decrement Grade Register

The Decrement Grade register stores the reload value for the Decrement counter. They are prepared to automatically decrement the stored value in the Amplitude Data register.

■ Decrement Grade Register

Decrement Grade register

Address: 001948_H

	7	6	5	4	3	2	1	0	⇦ Bit number
	D7	D6	D5	D4	D3	D2	D1	D0	SGDR
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	

When the DEC bit is "1" and the Decrement counter counts the number of tone pulses up to the reload value, the stored value in the Amplitude Data register is decremented by 1(one) at the end of the tone cycle.

This operation realizes automatic de-gradation of the sound with fewer number of CPU interventions.

It should be noted that the number of the tone pulses specified by this register equals to "register value +1". When the Decrement Grade register is set to "00", the decrement operation is performed every tone cycle.

21.2.5 Tone Count Register

The Tone Count register stores the reload value for the Tone Pulse counter. The Tone Pulse counter accumulate the number of tone pulses (or number of decrement operations) and when it reaches the reload value it sets the INT bit. They are intended to reduce the frequency of interrupts.

■ Tone Count Register

Tone Count register
Address: 001949_H

Bit number	15	14	13	12	11	10	9	8
SGTR	D7	D6	D5	D4	D3	D2	D1	D0
Read/write ⇨	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)
Initial value ⇨	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

The count input of the Tone Pulse counter is connected to the carry-out signal from the Decrement counter. And when the Tone count register is set to "00", the Tone Pulse counter sets the INT bit every carry-out from the Decrement counter. Thus the number of accumulated tone pulses is;

((Decrement Grade register) +1) x ((Tone Count register) +1)

i.e. When the both registers are set to "00", the INT bit is set every tone cycle.

CHAPTER 22 ROM CORRECTION

This chapter explains the functions and operations of ROM correction.

22.1 Outline of ROM Correction

22.2 Application Example of ROM Correction

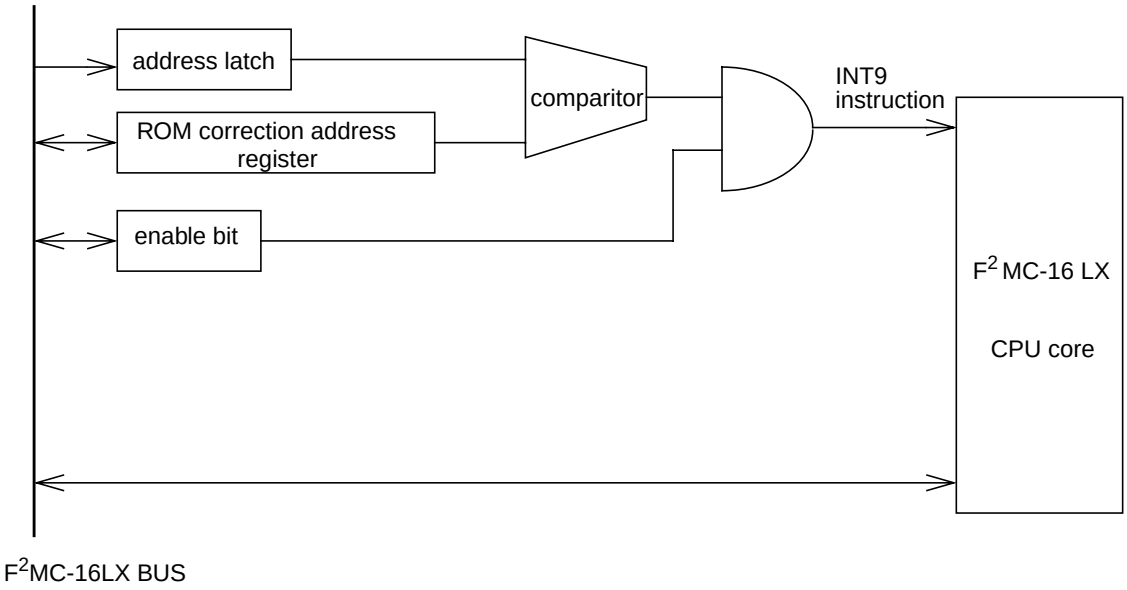
22.1 Outline of ROM Correction

When the setting of the address is the same as the ROM Correction Address registers, the INT9 instruction will be executed. By processing the INT9 interrupt service routine, the ROM correction function can be achieved.
There are two address registers, in each containing a compare enable bit. When the address register and the program counter are in agreement, and when the compare enable bit is at '1', then the CPU will be forced to execute INT9 instruction.

■ Block Diagram of ROM Correction

Figure 22.1-1 shows a block diagram of the ROM correction.

Figure 22.1-1 Block Diagram of ROM Correction



■ ROM Correction Address Register 0/1 (PADR0/PADR1)

These registers hold the addresses for the comparison with program counter. If there is an agreement and when the corresponding ADCSR compare enable bit is at '1', this module demands the CPU to execute the INT9 instruction.

If the corresponding compare enalble bit is '0', nothing will occur even there is a match.

	byte	byte	byte	access	initial value
PADR0 1FF2H/1FF1H/1FF0H				R/W	undefined
PADR1 1FF5H/1FF4H/1FF3H				R/W	undefined

The correspondance to the PACSR will be as follows.

ROM Correction Address register	Compare enable bit
PADR0	PADR1
AD0E	AD1E

■ ROM Correction Control Register (PACSR)

The correspondence with PACSR is shown below.

ROM Correction Control Register

Address : 009E$_H$

	7	6	5	4	3	2	1	0	
	Reserved	Reserved	Reserved	Reserved	AD1E	Reserved	AD0E	Reserved	PACSR
Read/write ⇨	(–)	(–)	(–)	(–)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇨	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)	

⇦ Bit number

This register controls operation of the address detect function and indicates its status.

[bit 7~4]

These are the reserved bits, be sure to write '0'.

[bit 3]: AD1E (Compare Enable 1)

This is the PADR1 enable bit.

When this bit is at '1', this module compares the PADR1 register and the program counter. If there is an agreement, the INT9 instruction is sent to the CPU.

[bit 2]:

Reserved bit.

[bit 1]: AD0E (Compare Enable 0)

This is the PADR0 enable bit.

When this bit is at '1', this module compares the PADR0 register and the program counter. If there is an agreement, the INT9 instruction is sent to the CPU.

[bit 0]:

Reserved bit.

■ Operations of ROM Correction

When the program counter indicates the same address as the ROM Correction Address register, the INT9 instruction will be executed. By processing the INT9 interrupt service routine, the ROM correction function can be achieved.

There are two address registers, in each containing a compare enable bit. When the address register and the program counter are in agreement, and when the compare enable bit is at '1', then the CPU will be forced to execute INT9 instruction.

Note:

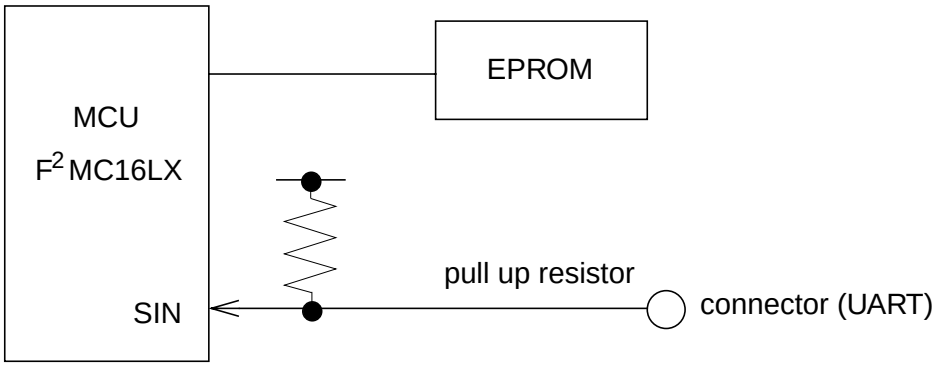
When the address register and the program counter are in agreement, the internal data bus content will be forced to be '01H', so INT9 instruction will be executed. Before changing the content of the address register, make sure the compare enable bit is at '0'. If it is changed while the compare enable bit is at '1', there will occur an error.

22.2 Application Example of ROM Correction

The ROM correction function is enabled by externally providing an EEPROM and by storing information on corrections and a patch program in the EEPROM. Based on the correction information stored in the EEPROM, the CPU sets addresses requiring corrections in the ROM correction module and transfers the patch program to RAM. The processing can be transferred to the patch program by executing the INT9 instruction after address match detection.

■ System Structure

Figure 22.2-1 System Structure Example



■ EEPROM Memory Map

address: content

0000H: number of bytes of the corrected program No. 0 (0 implies no ROM correction)

0001H: bit 7-0 program address No. 0

0002H: bit 15-8 program address No. 0

0003H: bit 24-16 program address No. 0

0004H: number of bytes of the corrected program No. 1 (0 implies no ROM correction)

0005H: bit 7-0 program address No. 1

0006H: bit 15-8 program address No. 1

0007H: bit 24-16 program address No. 1

0010H~: corrected program No. 0/1 body

■ Initial Condition

EEPROM all at '0'.

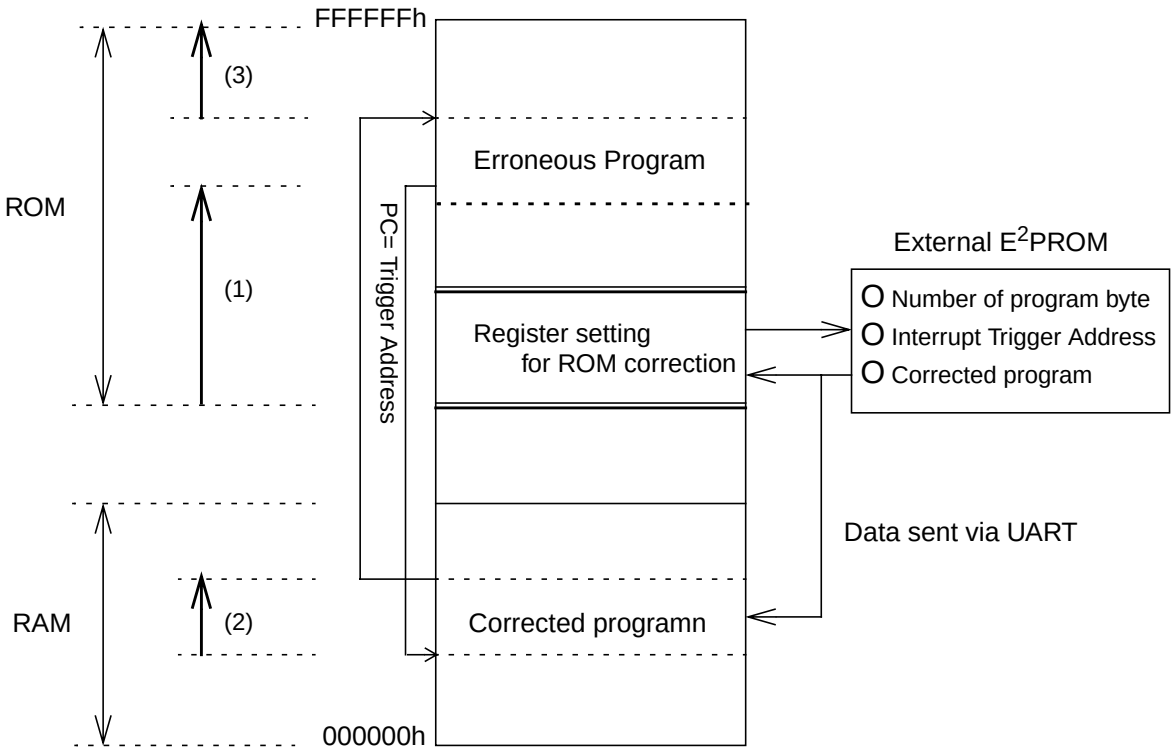
22.2.1 Correction Example of Program Errors

Send the body of the corrected program and the program address to the MCU through the connector (UART). MCU then writes that information into the EEPROM.

■ **When a Program Error Occurs**

Figure 22.2-2 shows an example of ROM correction processing when a program error occurs.

Figure 22.2-2 ROM Correction Processing Example



22.2.2 Example of Correction Processing

After resetting, the MCU reads the contents of the EEPROM. If the byte number of the corrected program is not '0', the body of the corrected program is read from the EEPROM and written in the RAM. Then the MCU sets the correction address either on PADR0 or on PADR1 and sets the compare enable bit. First address of the corrected program can written in the user-defined location of the RAM if a relocatable correction program is desired. In this case the INT9 service routine look for this user-defined location to jump to the corrected program.

■ **Reset Sequence**

Figure 22.2-3 Processing Flow of ROM Correction

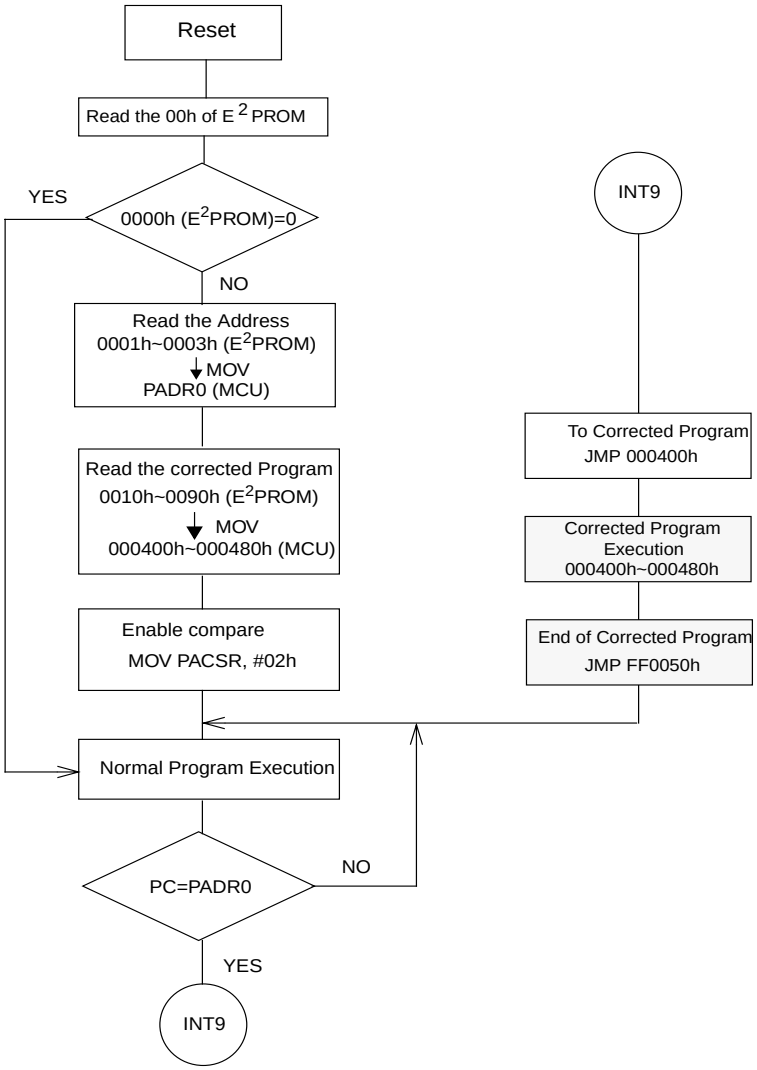
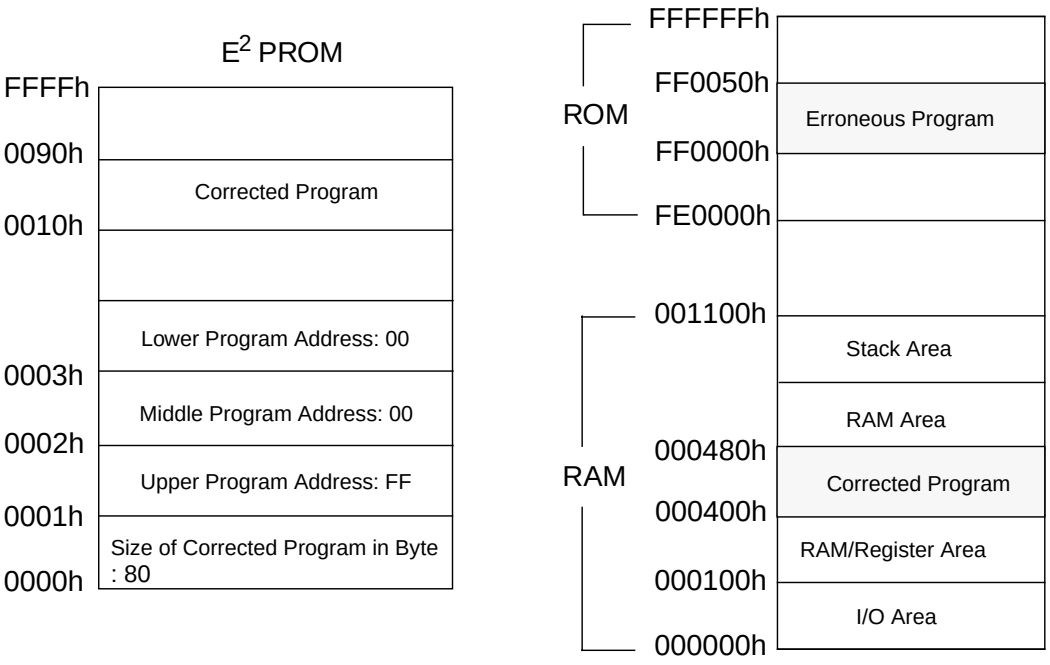


Figure 22.2-4 ROM Correction Processing Flow Diagram



■ **INT9 Interrupt**

In the interrupt routine, the address that produces the interrupt can be known by checking the stacked program counter value. The information stacked during interrupt will be discarded.

CHAPTER 23 ROM MIRRORING MODULE

This chapter explains the ROM mirroring module.

23.1 Outline of ROM Mirroring Module

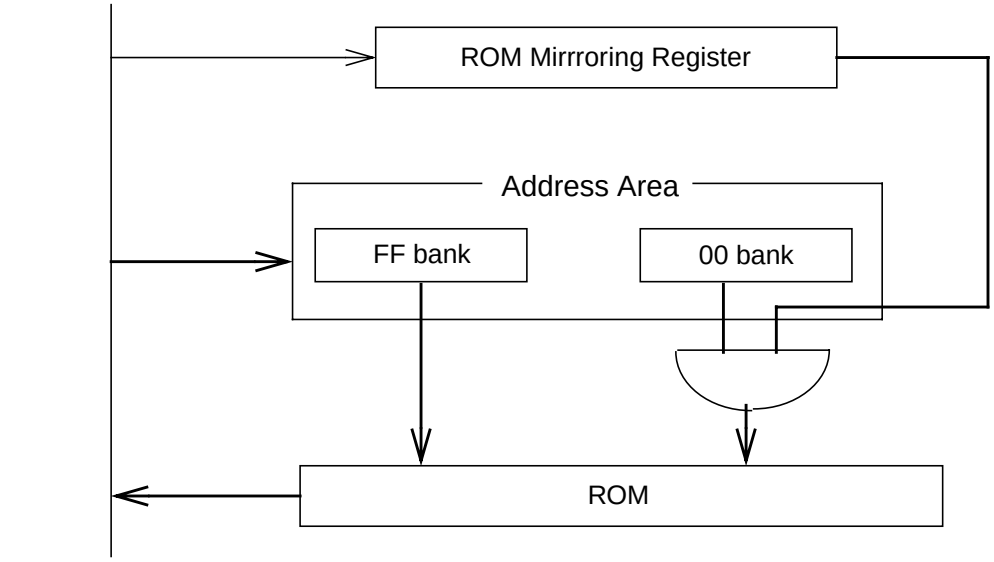
23.2 ROM Mirroring Register (ROMM)

23.1 Outline of ROM Mirroring Module

The ROM Mirroring module switches whether to mirror the image of the FF bank of the ROM to the 00 bank.

■ Block Diagram of ROM Mirroring Module

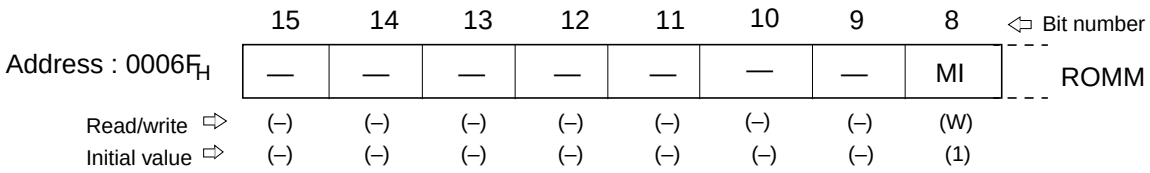
Figure 23.1-1 Block Diagram of ROM Mirroring Module



23.2 ROM Mirroring Register (ROMM)

Do not access the ROM mirroring register (ROMM) when addresses 004000H to 00FFFFH are being accessed.

■ ROM Mirroring Register (ROMM)



[bit 8] : MI

The image of the ROM data in the FF bank can also be found in the 00 bank when '1' is written to this bit. However, this memory mapping will not be done when this bit is written to '0'. This bit is write only.

Note:

Only FF4000~FFFFFF is mirrorred to 004000~00FFFF when ROM mirroring functing is activated. Therefore, addresses FFF000~FF3FFF will not be mirrorred to 00 bank.

CHAPTER 24 2M/3M-BIT FLASH MEMORY

This chapter describes the functions and operation of the 2M/3M-bit flash memory. The following three methods are available for writing data to and erasing data from the flash memory:

- **Parallel writer (Minato Electronics Model 1890A)**
- **Serial writer (Yokogawa Digital Computer Model AF-200)**
- **Executing programs to write/erase data**

This chapter elaborates on Chapter 3, "Executing Programs to Write/Erase Data."

24.1 Outline of 2M/3M-Bit Flash Memory

24.2 Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory

24.3 Write/Erase Modes

24.4 Flash Memory Control Status Register (FMCS)

24.5 Starting the Flash Memory Automatic Algorithm

24.6 Confirming the Automatic Algorithm Execution State

24.7 Detailed Explanation of Writing to and Erasing Flash Memory

24.8 Notes on Using 2M/3M-Bit Flash Memory

24.9 Reset Vector Address in Flash Memory

24.10 Example of Programming 2M/3M-Bit Flash Memory

24.1 Outline of 2M/3M-Bit Flash Memory

The 2M/3M-bit flash memory is mapped to the FC to FF bank in the CPU memory map. The functions of the flash memory interface circuit enable read-access and program-access from the CPU in the same way as mask ROM. Instructions from the CPU can be used via the flash memory interface circuit to write data to and erase data from the flash memory. Internal CPU control therefore enables rewriting of the flash memory while it is mounted. As a result, improvements in programs and data can be performed efficiently.
Selector operations such as enable selector protect cannot be used.

■ 2M/3M-bit Flash Memory Features

- Use of automatic program algorithm (Embedded Algorithm: Equivalent to MBM29F400TA)
- Erase pause/restart functions provided
- Detection of completion of writing/erasing using data polling or toggle bit functions
- Detection of completion of writing/erasing using CPU interrupts
- Compatible with JEDEC standard commands
- Sector erase function (any combination of sectors)
- Minimum of 10,000 write/erase operations

Embedded Algorithm is a trademark of Advanced Micro Device, Inc.

■ Writing to/Erasing Flash Memory

The flash memory cannot be written to and read at the same time. That is, when data is written to or erased data from the flash memory, the program in the flash memory must first be copied to RAM. The entire process is then executed in RAM so that data is simply written to the flash memory. This eliminates the need for the program to access the flash memory from the flash memory itself.

■ Flash Memory Register

❍ Flash Memory Control Status Register (FMCS)

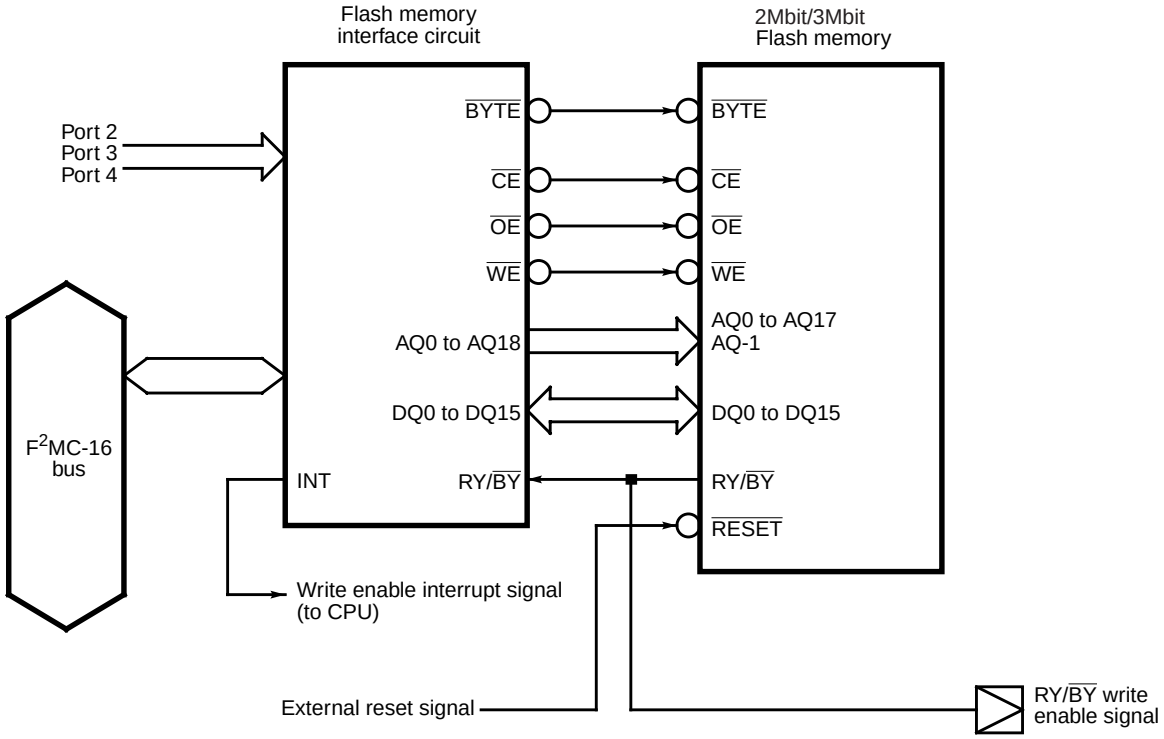
	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: $0000AE_H$	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(X)	(0)	(0)	(0)	(0)	

24.2 Block Diagram of the Entire Flash Memory and Sector Configuration of the Flash Memory

Figure 24.2-1 shows a block diagram of the entire flash memory with the flash memory interface circuit included. Figure 24.2-2 shows the sector configuration of the flash memory.

■ Block Diagram of the Entire Flash Memory

Figure 24.2-1 Block Diagram of the Entire Flash Memory



■ Sector Configuration of the 2M/3M-Bit Flash Memory

Figure 24.2-2 shows the sector configuration of the 2M/3M-bit flash memory. The addresses in the figure indicate the high-order and low-order addresses of each sector.

Figure 24.2-2 Sector Configuration of the 2M/3M-Bit Flash Memory

MB90F594A

	Writer address	CPU address
SA6 (16K bytes)	$7FFFF_H$	$FFFFFF_H$
SA5 (8K bytes)	$7BFFF_H$	$FFBFFF_H$
SA4 (8K bytes)	$79FFF_H$	$FF9FFF_H$
SA3 (32K bytes)	$77FFF_H$	$FF7FFF_H$
SA2 (64K bytes)	$6FFFF_H$	$FEFFFF_H$
SA1 (64K bytes)	$5FFFF_H$	$FDFFFF_H$
SA0 (64K bytes)	$4FFFF_H$	$FCFFFF_H$
	40000_H	$FC0000_H$

MB90F591

	Writer address	CPU address
SA11 (16K bytes)	$7FFFF_H$	$FFFFFF_H$
SA10 (8K bytes)	$7BFFF_H$	$FFBFFF_H$
SA8 (8K bytes)	$79FFF_H$	$FF9FFF_H$
SA8 (32K bytes)	$77FFF_H$	$FF7FFF_H$
SA7 (64K bytes)	$6FFFF_H$	$FEFFFF_H$
SA6 (64K bytes)	$5FFFF_H$	$FDFFFF_H$
Unused	$4FFFF_H$	$FCFFFF_H$
SA5 (16K bytes)	$3FFFF_H$	$FBFFFF_H$
SA4 (8K bytes)	$3BFFF_H$	$FBBFFF_H$
SA3 (8K bytes)	$39FFF_H$	$FB9FFF_H$
SA2 (32K bytes)	$37FFF_H$	$FB7FFF_H$
SA1(64K bytes)	$2FFFF_H$	$FAFFFF_H$
SA0 (64K bytes)	$1FFFF_H$	$F9FFFF_H$
Unused	$0FFFF_H$	$F8FFFF_H$
	00000_H	$F80000_H$

*: The writer address is equivalent to the CPU address when data is written to the flash memory using a parallel writer. When a general writer is used for writing/erasing, this address is used for writing/erasing.

24.3 Write/Erase Modes

The flash memory can be accessed in two different ways: Flash memory mode and alternative mode. Flash memory mode enables data to be directly written to or erased from the external pins. Alternative mode enables data to be written to or erased from the CPU via the internal bus. Use the mode external pins to select the mode.

■ Flash Memory Mode

The CPU stops when the mode pins are set to 111 while the reset signal is asserted. The flash memory interface circuit is connected directly to ports 0, 2, 3, and 4, enabling direct control from the external pins. This mode makes the MCU seem like a standard flash memory to the external pins, and write/erase can be performed using a flash memory programmer.

In flash memory mode, all operations supported by the flash memory automatic algorithm can be used.

■ Alternative Mode

The flash memory is located in the FC to FF banks in the CPU memory space, and like ordinary mask ROM, can be read-accessed and program-accessed from the CPU via the flash memory interface circuit.

Since writing/erasing the flash memory is performed by instructions from the CPU via the flash memory interface circuit, this mode allows rewriting even when the MCU is soldered on the target board.

Sector protect operations cannot be performed in these modes.

■ Flash Memory Control Signals

Table 24.3-1 lists the flash memory control signals in flash memory mode.

There is almost a one-to-one correspondence between the flash memory control signals and the external pins of the MBM29F400TA. The V_{ID} (12 V) pins required by the sector protect operations are MD0, MD1, and MD2 instead of A9, $\overline{\text{RESET}}$, and $\overline{\text{OE}}$ for the MBM29F400TA.

Since the memory capacity of the MB90F594A is half of the MBM29F400TA, the AQ18 pin corresponding to the address signal A17 of the MBM29F400TA is redundant. These pins should always be set to 1.

In flash memory mode, the external data bus signal width is limited to 8 bits, enabling only one-byte access. The DQ15 to DQ18 pins are not supported. The $\overline{\text{BYTE}}$ pin should always be set to 0.

Table 24.3-1 Flash Memory Control Signals

MB90F594A/MB90F591			MBM29F400TA
Pin number	Normal function	Flash memory mode	
1 to 8	P20 to P27	AQ0 to AQ7	A-1, A0 to A6
9	P30	AQ16	A15
10	P31	$\overline{\text{CE}}$	$\overline{\text{CE}}$

Table 24.3-1 Flash Memory Control Signals (Continued)

MB90F594A/MB90F591			MBM29F400TA
Pin number	**Normal function**	**Flash memory mode**	
12	P32	$\overline{OE}$	$\overline{OE}$
13	P33	$\overline{WE}$	$\overline{WE}$
14 to 15	P34 to P35	AQ17 to AQ18	A16 to A17
16	P36	$\overline{BYTE}$	$\overline{BYTE}$
17	P37	$\overline{RY/BY}$	$\overline{RY/BY}$
18 to 22	P40 to P44	AQ8 to AQ12	A7 to A11
24 to 26	P45 to P47	AQ13 to AQ15	A12 to A14
49	MD0	MDO	A9 (V_{ID})
50	MD1	MD1	$\overline{RESET}$ (V_{ID})
51	MD2	MD2	OE (V_{ID})
85 to 92	P00 to P07	DQ0 to DQ7	DQ0 to DQ7
77	$\overline{RST}$	$\overline{RESET}$	$\overline{RESET}$
Not supported			DQ8 to DQ15

24.4 Flash Memory Control Status Register (FMCS)

The flash memory control status register (FMCS), together with the flash memory interface circuit, is used to write data to and erase data from the flash memory.

■ Flash Memory Control Status Register (FMCS)

	7	6	5	4	3	2	1	0	⇐ Bit No.
Address: $0000AE_H$	INTE	RDYINT	WE	RDY	Reserved	LPM1	Reserved	LPM0	FMCS
Read/write ⇒	(R/W)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R/W)	
Initial value ⇒	(0)	(0)	(0)	(1)	(0)	(0)	(0)	(0)	

❍ Explanation of bits

[Bit 7] INTE (interrupt enable)

This bit generates an interrupt to the CPU when flash memory write/erase terminates.

An interrupt to the CPU is generated when the INTE and RDYINT bits are 1. No interrupt is generated when the INTE bit is 0.

- 0: Disables interrupts when write/erase terminates.
- 1: Enables interrupts when write/erase terminates.

[Bit 6] RDYINT (ready interrupt)

This bit indicates the operating state of the flash memory.

This bit is set to 1 when flash memory write/erase terminates. Data cannot be written to or erased from the flash memory while this bit is 0 after a flash memory write/erase. Flash memory write/erase is enabled when write/erase terminates and this bit is set to 1.

Writing 0 clears this bit to 0. Writing 1 is ignored. This bit is set to 1 at the termination timing of the flash memory automatic algorithm (see Section 1.4, "Starting the Flash Memory Automatic Algorithm"). When the read-modify-write (RMW) instruction is used, 1 is always read.

- 0: Write/erase is being executed.
- 1: Write/erase has terminated (interrupt request generated).

[Bit 5] WE (write enable)

This bit enables writing to the flash memory area.

When this bit is 1, writing after the command sequence (see Section 1.4, "Starting the Flash Memory Automatic Algorithm") is issued to the FC to FF bank writes to the flash memory area. When this bit is 0, the write/erase signal is not generated. This bit is used when the flash memory Write/Erase command is started.

If write/erase is not performed, it is recommended that this bit be set to 0 to prevent data from being mistakenly written to the flash memory.

- 0: Disables flash memory write/erase.

- 1: Enables flash memory write/erase.

[Bit 4] RDY (ready)

This bit enables flash memory write/erase.

Flash memory write/erase is disabled while this bit is 0. However, Suspend commands, such as the Read/Reset command and Sector Erase Suspend command, can be accepted even if this bit is 0.

- 0: Write/erase is being executed.
- 1: Write/erase has terminated (next data write/erase enabled).

[Bits 3 and 1] Reserved bits

These bits are reserved for testing. During regular use, they should always be set to 0.

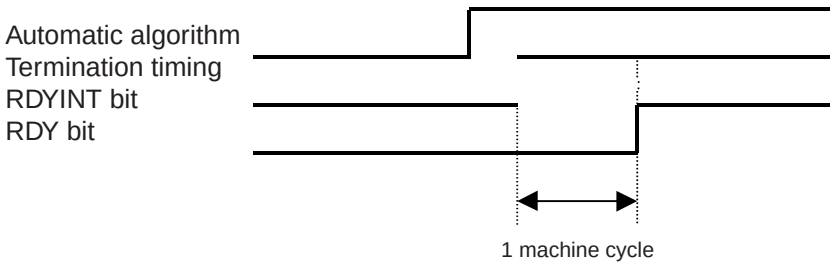
[Bits 2 and 0] LPM1 and LPM0 (low power mode)

These bits control the current consumed by the flash memory when the flash memory is accessed. Since the access time to the flash memory from the CPU is largely dependent on this setting, select a setting value based on the operating frequency of the CPU.

- 01: Low power consumption mode (Operates at an internal operating frequency up to 4 MHz.)
- 10: Low power consumption mode (Operates at an internal operating frequency up to 8 MHz.)
- 11: Low power consumption mode (Operates at an internal operating frequency up to 10 MHz.)
- 00: Regular power consumption mode (Operates at an internal operating frequency up to 16 MHz.)

Note:

The RDYINT and RDY bits cannot be changed at the same time. Create a program so that decisions are made using one or the other of these bits.



24.5 Starting the Flash Memory Automatic Algorithm

Four types of commands are available for starting the flash memory automatic algorithm: Read/Reset, Write, and Chip Erase. Control of suspend and restart is enabled for sector erase.

■ Command Sequence Table

Table 24.5-1 lists the commands used for flash memory write/erase. All of the data written to the command register is in bytes, but use word access to write. The data of the high-order bytes at this time is ignored.

Table 24.5-1 Command Sequence Table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/Reset (*1)	1	FxXXXX	XXF0	-	-	-	-	-	-	-	-	-	-
Read/Reset (*1)	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXF0	RA	RD	-	-	-	-
Write program	4	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XXA0	PA (even)	PD (word)	-	-	-	-
Chip Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX10
Sector Erase	6	FxAAAA	XXAA	Fx5554	XX55	FxAAAA	XX80	FxAAAA	XXAA	Fx5554	XX55	SA (even)	XX30
Sector Erase Suspend		Entering address FxXXXX data (xxBOH) suspends erasing during sector erase.											
Sector Erase Restart		Entering address FxXXXX data (xx3OH) restarts erasing after erasing is suspended during sector erase.											

Notes:

- The addresses Fx in the table mean FF, FE, FD, and FC. Use these addresses as the access target bank values for operations.
- The addresses in the table are the values in the CPU memory map. All addresses and data are represented using hexadecimal notation. However, the letter X is an optional value.
- RA: Read address
- PA: Write address. Only even addresses can be specified.
- SA: Sector address. See Section 1.2, "2M/3M-Bit Flash Memory Sector Configuration."
- RD: Read data
- PD: Write data. Only word data can be specified.

*1 Both of the two types of Read/Reset commands can reset the flash memory to read mode.

24.6 Confirming the Automatic Algorithm Execution State

Because the write/erase flow of the flash memory is controlled using the automatic algorithm, the flash memory has hardware for posting its internal operating state and completion of operation. This automatic algorithm enables confirmation of the operating state of the built-in flash memory using the following hardware sequences.

■ Hardware Sequence Flags

The hardware sequence flags are configured from the four-bit output of DQ7, DQ6, DQ5, and DQ3. The functions of these bits are those of the data polling flag (DQ7), toggle bit flag (DQ6), timing limit exceeded flag (DQ5), and sector erase timer flag (DQ3). The hardware sequence flags can therefore be used to confirm that writing or chip sector erase has been completed or that erase code write is valid.

The hardware sequence flags can be accessed by read-accessing the addresses of the target sectors in the flash memory after setting of the command sequence (see Table 24.5-1 in Section 24.5, "Starting the Flash Memory Automatic Algorithm"). Table 24.6-1 lists the bit assignments of the hardware sequence flags.

Table 24.6-1 Bit Assignments of Hardware Sequence Flags

Bit No.	7	6	5	4	3	2	1	0
Hardware sequence flag	DQ7	DQ6	DQ5	-	DQ3	-	-	-

To determine whether automatic writing or chip sector erase is being executed, the hardware sequence flags can be checked or the status can be determined from the RDY bit of the flash memory control register (FMCS) that indicates whether writing has been completed. After writing/erasing has terminated, the state returns to the read/reset state. When creating a program, use one of the flags to confirm that automatic writing/erasing has terminated. Then, perform the next processing operation, such as data read. In addition, the hardware sequence flags can be used to confirm whether the second or subsequent sector erase code write is valid. The following sections describe each hardware sequence flag separately. Table 24.6-2 lists the functions of the hardware sequence flags.

Table 24.6-2 Hardware Sequence Flag Functions

State		DQ7	DQ6	DQ5	DQ3
State change for normal operation	Write --> Write completed (write address specified)	$\overline{DQ7}$ --> DATA:7	Toggle --> DATA:6	0 --> DATA:5	0 --> DATA:3
	Chip/sector erase --> Erase completed	0 --> 1	Toggle --> Stop	0 --> 1	1
	Sector erase wait --> Erase started	0	Toggle	0	0 --> 1
	Erase --> Sector erase suspended (sector being erased)	0 --> 1	Toggle --> 1	0	1 --> 0
	Sector erase suspend --> Erase restarted (sector being erased)	1 --> 0	1 --> Toggle	0	0 --> 1
	Sector erase suspended (sector not being erased)	DATA:7	DATA:6	DATA:5	DATA:3
Abnormal operation	Write	$\overline{DQ7}$	Toggle	1	0
	Chip/sector erase	0	Toggle	1	1

24.6.1 Data Polling Flag (DQ7)

The data polling flag (DQ7) uses the data polling function to post that the automatic algorithm is being executed or has terminated

■ Data Polling Flag (DQ7)

Tables 24.6-3 and 24.6-4 list the state transitions of the data polling flag.

Table 24.6-3 Data Polling Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ7	$\overline{DQ7}$ -->	0 --> 1	0	0 --> 1	1 --> 0	DATA:7

Table 24.6-4 Data Polling Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ7	$\overline{DQ7}$	0

❍ Write

Read-access during execution of the automatic write algorithm causes the flash memory to output the opposite data of bit 7 last written, regardless of the value at the address specified by the address signal. Read-access at the end of the automatic write algorithm causes the flash memory to output bit 7 of the read value of the address specified by the address signal.

❍ Chip/sector erase

For a sector erase, read-access during execution of the chip erase/sector erase algorithm causes the flash memory to output 0 from the sector currently being erased. For a chip erase, read-access causes the flash memory to output 0 regardless of the value at the address specified by the address signal. Read-access at the end of the automatic write algorithm causes the flash memory to output 1 in the same way.

❍ Sector erase suspend

Read-access during a sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 7 (DATA: 7) of the read value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased. Referencing this flag together with the toggle bit flag (DQ6) enables a decision to be made on whether the flash memory is in the erase suspended state and which sector is being erased.

Note:

When the automatic algorithm is being started, read-access to the specified address is ignored. Since termination of the data polling flag (DQ7) can be accepted for a data read

and other bits output, data read after the automatic algorithm has terminated should be performed after read-access has confirmed that data polling has terminated.

24.6.2 Toggle Bit Flag (DQ6)

Like the data polling flag, the toggle bit flag (DQ6) uses the toggle bit function to post that the automatic algorithm is being executed or has terminated.

■ Toggle Bit Flag (DQ6)

Tables 24.6-5 and 24.6-6 list the state transitions of the toggle bit flag.

Table 24.6-5 Toggle Bit Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ6	Toggle --> DATA:6	Toggle --> Stop	Toggle	Toggle --> 1	1 --> Toggle	DATA:6

Table 24.6-6 Toggle Bit Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ6	Toggle	Toggle

❍ **Write/chip sector erase**

Continuous read-access during execution of the automatic write algorithm and chip/sector erase algorithm causes the flash memory to toggle the 1 or 0 state for every read cycle, regardless of the value at the address specified by the address signal. Continuous read-access at the end of the automatic write algorithm and chip/sector erase algorithm causes the flash memory to stop toggling bit 6 and output bit 6 (DATA: 6) of the read value of the address specified by the address signal.

❍ **Sector erase suspend**

Read-access during a sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 6 (DATA: 6) of the read value at the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

<Note>

For a write, if the sector where data is to be written is rewrite-protected, the toggle bit terminates the toggle operation after approximately 2µs without any data being rewritten. For an erase, if all of the selected sectors are write-protected, the toggle bit performs toggling for approximately 100µs and then returns to the read/reset state without any data being rewritten.

24.6.3 Timing Limit Exceeded Flag (DQ5)

The timing limit exceeded flag (DQ5) is used to post that execution of the automatic algorithm has exceeded the time (internal pulse count) prescribed in the flash memory.

■ Timing Limit Exceeded Flag (DQ5)

Tables 24.6-7 and 24.6-8 list the state transitions of the timing limit exceeded flag.

Table 24.6-7 Timing Limit Exceeded Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ5	0 --> DATA:5	0 --> 1	0	0	0	DATA:5

Table 24.6-8 Timing Limit Exceeded Bit Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ5	1	1

❍ Write/chip sector erase

Read-access after write or chip/sector erase automatic algorithm activation causes the flash memory to output 0 if the time is within the prescribed time (time required for write/erase) or to output 1 if the prescribed time has been exceeded. Because this is done regardless of whether the automatic algorithm is being executed or has terminated, it is possible to determine whether write/erase was successful or unsuccessful. That is, when this flag outputs 1, writing can be determined to have been unsuccessful if the automatic algorithm is still being executed by the data polling function or toggle bit function.

For example, writing 1 to a flash memory address where 0 has been written will cause the fail state to occur. In this case, the flash memory will lock and execution of the automatic algorithm will not terminate. As a result, valid data will not be output from the data polling flag (DQ7). In addition, the toggle bit flag (DQ6) will exceed the time limit without stopping the toggle operation and the timing limit exceeded flag (DQ5) will output 1. Note that this state indicates that the flash memory is not faulty, but has been used correctly. When this state occurs, execute the Reset command.

24.6.4 Sector Erase Timer Flag (DQ3)

The sector erase timer flag (DQ3) is used to post whether the automatic algorithm is being executed during the sector erase wait period after the Sector Erase command has been started.

■ **Sector Erase Timer Flag (DQ3)**

Tables 24.6-9 and 24.6-10 list the state transitions of the sector erase timer flag.

Table 24.6-9 Sector Erase Timer Flag State Transitions (State Change for Normal Operation)

Operating state	Write --> Completed	Chip/sector erase --> Completed	Sector erase wait --> Started	Sector erase --> Erase suspend (sector being erased)	Sector erase suspend --> Restarted (sector being erased)	Sector erase suspended (sector not being erased)
DQ3	0 --> DATA:3	1	0 --> 1	1 --> 0	0 --> 1	DATA:3

Table 24.6-10 Sector Erase Timer Flag State Transitions (State Change for Abnormal Operation)

Operating state	Write	Chip/sector erase
DQ3	0	1

❍ **Sector erase**

Read-access after the Sector Erase command has been started causes the flash memory to output 0 if the automatic algorithm is being executed during the sector erase wait period, regardless of the value at the address specified by the address signal of the sector that issued the command. The flash memory outputs 1 if the sector erase wait period has been exceeded.

If the data polling function or toggle bit function indicates that the erase algorithm is being executed, internally controlled erase has already started if this flag is 1. Continuous write of the sector erase codes or commands other than the Sector Erase Suspend command will be ignored until erase is terminated.

If this flag is 0, the flash memory will accept write of additional sector erase codes. To confirm this, it is recommended that the state of this flag be checked before continuing to write sector erase codes. If this flag is 1 after the second state check, it is possible that additional sector erase codes may not be accepted.

❍ **Sector erase**

Read-access during execution of sector erase suspend causes the flash memory to output 1 if the address specified by the address signal belongs to the sector being erased. The flash memory outputs bit 3 (DATA: 3) of the read value of the address specified by the address signal if the address specified by the address signal does not belong to the sector being erased.

24.7 Detailed Explanation of Writing to and Erasing Flash Memory

This section describes each operation procedure of flash memory Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend, and Sector Erase Restart when a command that starts the automatic algorithm is issued.

■ Detailed Explanation of Flash Memory Write/Erase

The flash memory executes the automatic algorithm by issuing a command sequence (see Table 24.5-1 in Section 24.5, "Starting the Flash Memory Automatic Algorithm") for a write cycle to the bus to perform Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend, or Sector Erase Restart operations. Each bus write cycle must be performed continuously. In addition, whether the automatic algorithm has terminated can be determined using the data polling or other function. At normal termination, the flash memory is returned to the read/reset state.

Each operation of the flash memory is described in the following order:

- Setting the read/reset state
- Writing data
- Erasing all data (erasing chips)
- Erasing optional data (erasing sectors)
- Suspending sector erase
- Restarting sector erase

24.7.1 Setting The Read/Reset State

This section describes the procedure for issuing the Read/Reset command to set the flash memory to the read/reset state.

■ Setting the Flash Memory to the Read/Reset State

The flash memory can be set to the read/reset state by sending the Read/Reset command in the command sequence table (see Table 24.5-1 in Section 24.5, "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Read/Reset command has two types of command sequences that execute the first and third bus operations. However, there are no essential differences between these command sequences.

The read/reset state is the initial state of the flash memory. When the power is turned on and when a command terminates normally, the flash memory is set to the read/reset state. In the read/reset state, other commands wait for input.

In the read/reset state, data is read by regular read-access. As with the mask ROM, program access from the CPU is enabled. The Read/Reset command is not required to read data by a regular read. The Read/Reset command is mainly used to initialize the automatic algorithm in such cases as when a command does not terminate normally.

24.7.2 Writing Data

This section describes the procedure for issuing the Write command to write data to the flash memory.

■ Writing Data to the Flash Memory

The data write automatic algorithm of the flash memory can be started by sending the Write command in the command sequence table (see Table 24.5-1 in Section 24.5, "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory. When data write to the target address is completed in the fourth cycle, the automatic algorithm and automatic write are started.

❍ Specifying addresses

Only even addresses can be specified as the write addresses specified in a write data cycle. Odd addresses cannot be written correctly. That is, writing to even addresses must be done in units of word data.

Writing can be done in any order of addresses or even if the sector boundary is exceeded. However, the Write command writes only data of one word for each execution.

❍ Notes on writing data

Writing cannot return data 0 to data 1. When data 1 is written to data 0, the data polling algorithm (DQ7) or toggle operation (DQ6) does not terminate and the flash memory elements are determined to be faulty. If the time prescribed for writing is thus exceeded, the timing limit exceeded flag (DQ6) is determined to be an error. Otherwise, the data is viewed as if dummy data 1 had been written. However, when data is read in the read/reset state, the data remains 0. Data 0 can be set to data 1 only by erase operations.

All commands are ignored during execution of the automatic write algorithm. If a hardware reset is started during writing, the data of the written addresses will be unpredictable.

■ Writing to the Flash Memory

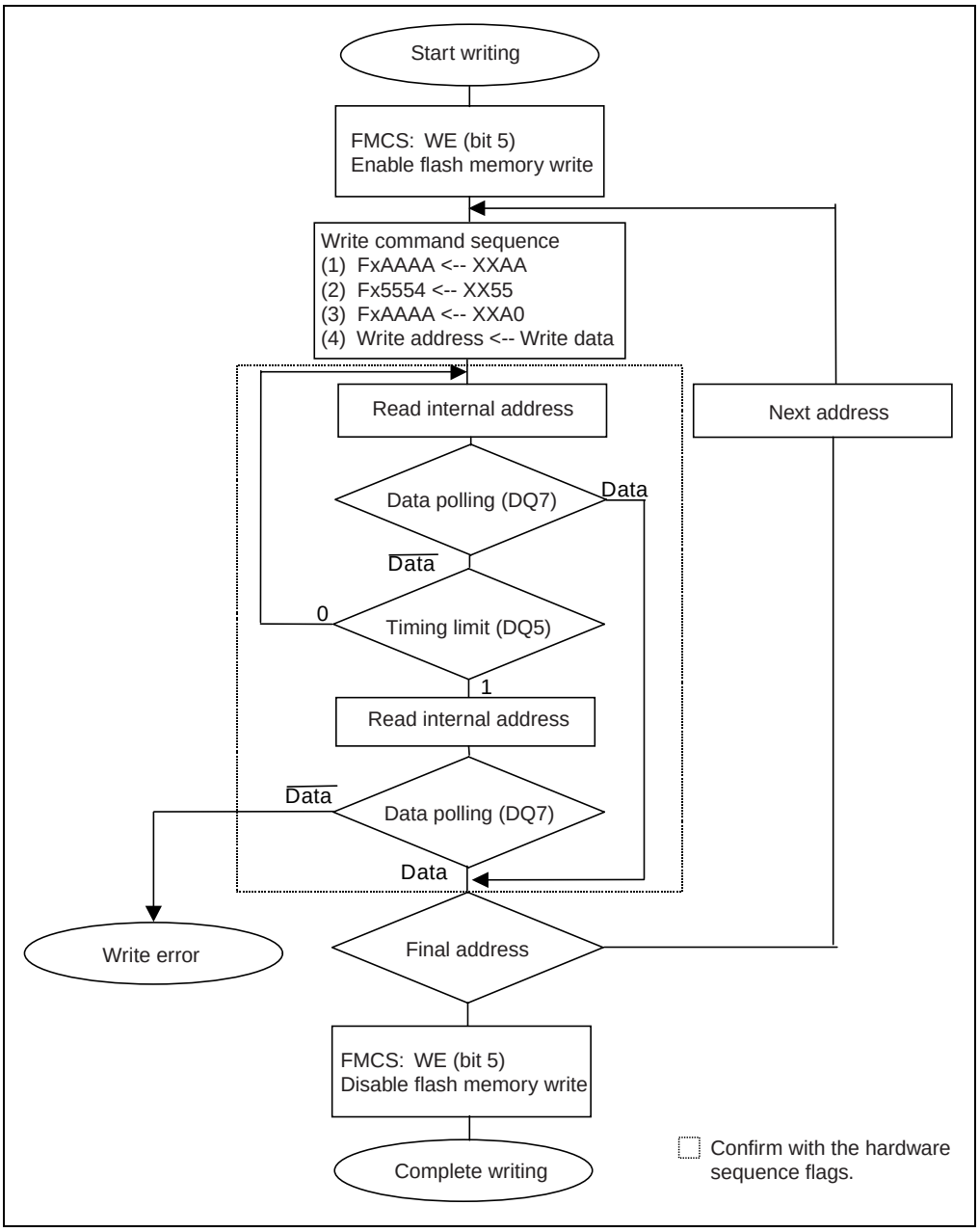
Figure 24.7-1 is an example of the procedure for writing to the flash memory. The hardware sequence flags (see Section 24.6, "Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Here, the data polling flag (DQ7) is used to confirm that writing has terminated.

The data read to check the flag is read from the address written to last.

The data polling flag (DQ7) changes at the same time that the timing limit exceeded flag (DQ5) changes. For example, even if the timing limit exceeded flag (DQ5) is 1, the data polling flag bit (DQ7) must be rechecked.

Also for the toggle bit flag (DQ6), the toggle operation stops at the same time that the timing limit exceeded flag bit (DQ5) changes to 1. The toggle bit flag (DQ6) must therefore be rechecked.

Figure 24.7-1 Example of the Flash Memory Write Procedure



24.7.3 Erasing All Data (Erasing Chips)

This section describes the procedure for issuing the Chip Erase command to erase all data in the flash memory.

■ Erasing all Data in the Flash Memory (Erasing Chips)

All data can be erased from the flash memory by sending the Chip Erase command in the command sequence table (see Table 24.5-1 in Section 24.5, "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Chip Erase command is executed in six bus operations. When writing of the sixth cycle is completed, the chip erase operation is started. For chip erase, the user need not write to the flash memory before erasing. During execution of the automatic erase algorithm, the flash memory writes 0 for verification before all of the cells are erased automatically.

24.7.4 Erasing Optional Data (Erasing Sectors)

This section describes the procedure for issuing the Sector Erase command to erase optional data (erase sector) in the flash memory. Individual sectors can be erased. Multiple sectors can also be specified at one time.

■ Erasing Optional Data (Erasing Sectors) in the Flash Memory

Optional sectors in the flash memory can be erased by sending the Sector Erase command in the command sequence table (see Table 24.5-1 in Section 24.5, "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

❍ Specifying sectors

The Sector Erase command is executed in six bus operations. Sector erase wait of 50µs is started by writing the sector erase code (30h) to an accessible even-numbered address in the target sector in the sixth cycle. To erase multiple sectors, write the erase code (30h) to the addresses in the target sectors after the above processing operation.

❍ Notes on specifying multiple sectors

Erase is started when the sector erase wait period of 50µs terminates after the final sector erase code has been written. That is, to erase multiple sectors at one time, an erase code (sixth cycle of the command sequence) must be written within 50µs of writing of the address of a sector and the address of the next sector must be written within 50µs of writing of the previous erase code. Otherwise, the address and erase code may not be accepted. The sector erase timer (hardware sequence flag DQ3) can be used to check whether writing of the subsequent sector erase code is valid. At this time, specify so that the address used for reading the sector erase timer indicates the sector to be erased.

■ Erasing Sectors in the Flash Memory

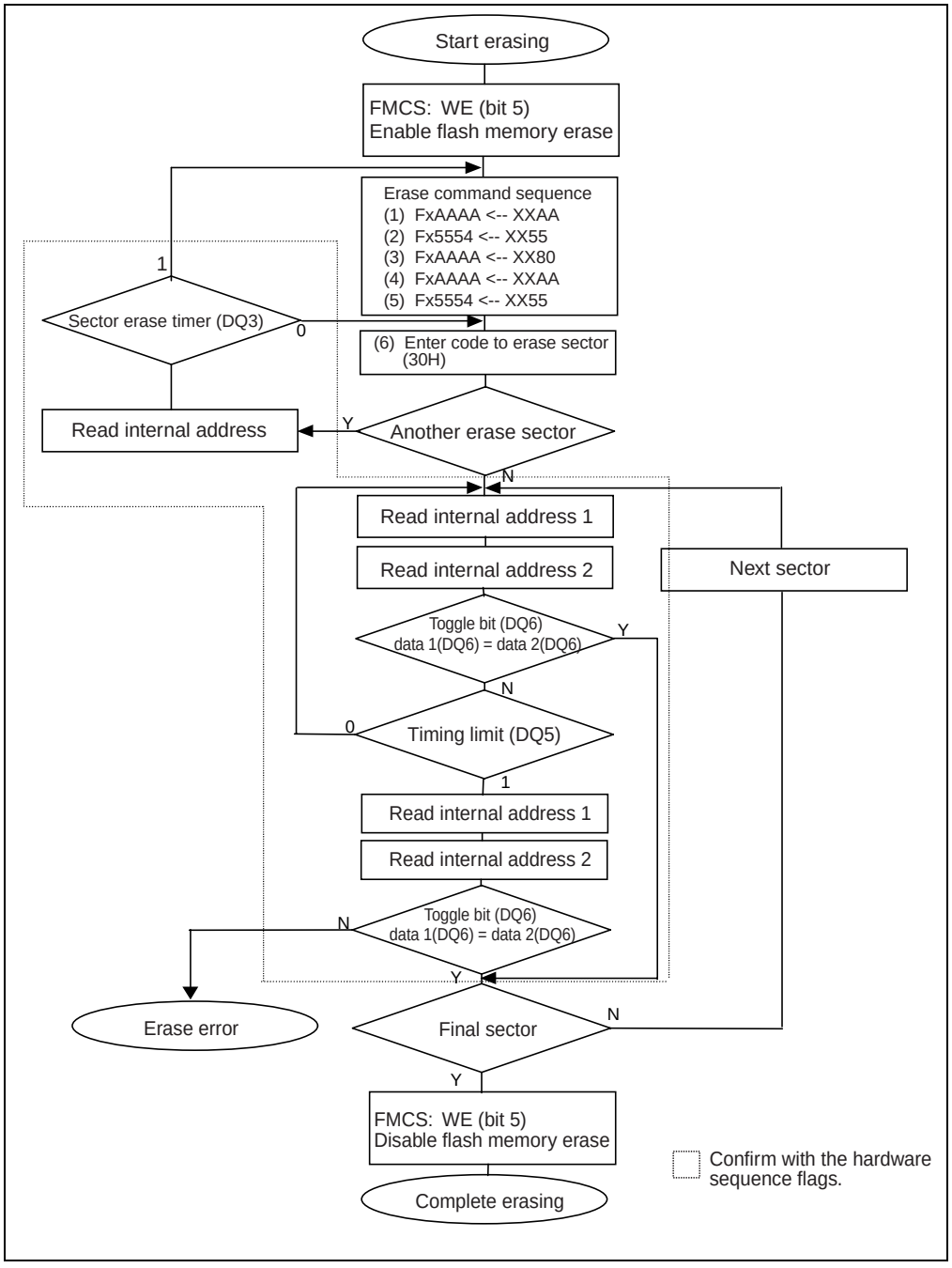
The hardware sequence flags (see Section 1.5, "Referencing the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Figure 24.7-2 is an example of the procedure for erasing sectors in the flash memory. Here, the toggle bit flag (DQ6) is used to confirm that erasing has terminated.

The data that is read to check the flag is read from the sector to be erased.

The toggle bit flag (DQ6) stops the toggle operation at the same time that the timing limit exceeded flag (DQ5) is changed to 1. For example, even if the timing limit exceeded flag (DQ5) is 1, the toggle bit flag (DQ6) must be rechecked.

The data polling flag (DQ7) also changes at the same time that the timing limit exceeded flag bit (DQ5) changes. As a result, the data polling flag (DQ7) must be rechecked.

Figure 24.7-2 Example of the Flash Memory Sector Erase Procedure



24.7.5 Suspending Sector Erase

This section describes the procedure for issuing the Sector Erase Suspend command to suspend erasing of flash memory sectors. Data can be read from sectors that are not being erased.

■ Suspending Erasing of Flash Memory Sectors

Erasing of flash memory sectors can be suspended by sending the Sector Erase Suspend command in the command sequence table (see Table 1 in Section 1.4, "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Sector Erase Suspend command suspends the sector erase operation being executed and enables data to be read from sectors that are not being erased. In this state, only reading is enabled; data cannot be written. This command is valid only during sector erase operations that include the erase wait time. The command will be ignored during chip erase or write operations.

This command is implemented by writing the erase suspend code (B0h). At this time, specify an optional address in the flash memory for the address. An Erase Suspend command issued again during erasing of sectors will be ignored.

Entering the Sector Erase Suspend command during the sector erase wait period will immediately terminate sector erase wait, cancel the erase operation, and set the erase stop state. Entering the Erase Suspend command during the erase operation after the sector erase wait period has terminated will set the erase suspend state after a maximum period of 15μs has elapsed.

24.7.6 Restarting Sector Erase

This section describes the procedure for issuing the Sector Erase Restart command to restart suspended erasing of flash memory sectors.

■ Restarting Erasing of Flash Memory Sectors

Suspended erasing of flash memory sectors can be restarted by sending the Sector Erase Restart command in the command sequence table (see Table 24.5-1 in Section 24.5, "Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Sector Erase Restart command is used to restart erasing of sectors from the sector erase suspend state set using the Sector Erase Suspend command. The Sector Erase Restart command is implemented by writing the erase restart code (30h). At this time, specify an optional address in the flash memory area for the address.

If a Sector Erase Restart command is issued during sector erase, the command will be ignored.

24.8 Notes on using 2M-Bit Flash Memory

This section contains notes on using 2M-bit flash memory.

■ Notes on using flash memory

❍ Input of a hardware reset ($\overline{RST}$)

To input a hardware reset when the automatic algorithm has not been started and reading is in progress, a minimum low-level width of 500 ns must be maintained. In this case, a maximum of 500 ns is required until data can be read from the flash memory after a hardware reset has been activated.

Similarly, to input a hardware reset when the automatic algorithm has been activated and writing or erasing is in progress, a minimum low-level width of 50 ns must be maintained. In this case, 20 (s are required until data can be read after the operation for initializing the flash memory has terminated.

A hardware reset during writing the data being written to be undefined. A hardware reset during erasing may make the sector being erased unusable.

❍ Canceling of a software reset, watchdog timer reset, and hardware standby

When the flash memory is being written to or erased with CPU access and if reset conditions occur while the automatic algorithm is active, the CPU may run out of control. This occurs because these reset conditions cause the automatic algorithm to continue without initializing the flash memory unit, possibly preventing the flash memory unit from entering the read state when the CPU starts the sequence after the reset has been deasserted. These reset conditions must be disabled during writing to or erasing of the flash memory.

❍ Program access to flash memory

When the automatic algorithm is operating, read access to the flash memory is disabled. With the memory access mode of the CPU set to internal ROM mode, writing or erasing must be started after the program area is switched to another area such as RAM. In this case, when sectors (SA6) containing interrupt vectors are erased, writing or erasing interrupt processing cannot be executed. For the same reason, all interrupt sources other than the flash memory are disabled while the automatic algorithm is operating.

Also, while the automatic algorithm is being executed, all interrupt sources except flash memory are disabled.

❍ Hold function

When the CPU accepts a hold request, the Write signal $\overline{WE}$ of the flash memory unit may be skewed, causing erroneous writing or erasing due to an erroneous write. When the acceptance of a hold request is enabled (HDE bit of EPCR set to 1), ensure that the WE bit of the control status register (FMCS) is 0.

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

Because write and erase interrupts issued to the CPU from the flash memory interface circuit cannot be accepted by the EI^{2}OS, they should not be used.

❍ **Applying V_{ID}**

Applying V_{ID} required for the sector protect operation should always be started and terminated when the supply voltage is on.

24.9 Reset Vector Address in Flash Memory

The MB90F594A supports a hard-wired reset vector.
When the addresses FFFFDC$_H$ to FFFFDF$_H$ are accessed for reading data in internal vector mode, the values that have been determined by the hard-wired logic in advance are read. However, in flash memory mode, as mentioned in the previous chapter, all addresses can be accessed.
Consequently, it is meaningless to write data to these addresses. Especially when programming flash memory from the CPU (that is, not in flash memory mode), do not read these addresses for software polling. Otherwise, the flash memory returns a fixed reset vector instead of the hardware sequence flag value.

■ **Reset vector address in flash memory**

The following table shows the reset vector and mode data values determined in advance.

Reset vector	FFA000$_H$
Mode data	00$_H$

24.10 Example of Programming 2M/3M-Bit Flash Memory

This section presents a programming example of 2M/3M-bit flash memory.

■ Programming example of 2M/3M-bit flash memory

Flash Memory Sample Program

```
                    NAME   FLASHWE
                    TITLE  FLASHWE
;------------------------------------------------------------------------------
;2M/3M-bit-FLASH test program
;
;1: Transmits the program (address: FFBC00H, sector: SA6) from FLASH to RAM
;   (address: 001500H).
;2: Executes the program on RAM.
;3: Writes the PDR1 value to FLASH (address: FI0000H, sector: SA1).
;4: Reads the written value (address: FD0000H, sector: SA1) and outputs it to PDR2.
;5: Erases the written sector (SA1).
;6: Checks and outputs erase data.
;Conditions
;  - Number of bytes transmitted to RAM: 100H (256B)
;  - Write/erase termination judgment
;       Judgment according to DQ5 (timing limit excess flag)
;       Judgment according to DQ6 (toggle bit flag)
;       Judgment according to RDY (FMCS)
;  - Error handling
;       Hi output to P00 to P07
;       Reset command issuance
;------------------------------------------------------------------
;
RESOUS  IOSEG   ABS=00          ;"RESOUS" I/O segment definition
        ORG     0000H
PDR0    RB      1
PDR1    RB      1
PDR2    RB      1
PDR3    RB      1
        ORG     0010H
DDR0    RB      1
DDR1    RB      1
DDR2    RB      1
DDR3    RB      1
        ORG     00A1H
CKSCR   RB      1
        ORG     00AEH
FMCS    RB      1
        ORG     006FH
ROMM    RB      1
RESOUS  ENDS
;
SSTA    SSEG
        RW      0127H
STA_T   RW      1
SSTA    ENDS
;
DATA    DSEG    ABS=0FFH        ;FLASH command address
        ORG     5554H
COMADR2 RW      1
        ORG     0AAAAH
COMADR1 RW      1
DATA    ENDS
;////////////////////////////////////////////////////////////
```

```
;Main program (FFA000H)
;////////////////////////////////////////////////////////
CODE    CSEG
START:
;       ////////////////////////////////////////////////
;       Initialization
;       ////////////////////////////////////////////////
        MOV     CKSCR,#0BAH     ;3-multiple setting
        MOV     RP,#0
        MOV     A,#!STA_T
        MOV     SSB,A
        MOVW    A,#STA_T
        MOVW    SP,A
        MOV     ROMM,#00H       ;Mirror OFF
        MOV     PDR0,#00H       ;For error check
        MOV     DDR0,#0FFH
        MOV     PDR1,#00H       ;Port for data input
        MOV     DDR1,#00H
        MOV     PDR2,#00H       ;Port for data output
        MOV     DDR2,#0FFH
;       ////////////////////////////////////////////////////////
;       Transfer of "FLASH write erase program (FFBC00H)" to RAM (1500H address)
;       ////////////////////////////////////////////////////////
        MOVW    A,#1500H        ;Transfer destination RAM area
        MOVW    A,#0BC00H       ;Transfer source address (program position)
        MOVW    RW0,#100H       ;Number of bytes to be transferred
        MOVS    ADB,PCB         ;Transfer of 100H from FFBC00H to 001500H
        CALLP   001500H         ;Jump to the address containing the transferred
;                                program
;       ////////////////////////////////////////////////
;       Data output
;       ////////////////////////////////////////////////
OUT     MOV     A,#0FDH
        MOV     ADB,A
        MOVW    RW2,#0000H
        MOVW    A,@RW2+00
        MOV     PDR2,A
END     JMP     *
CODE    ENDS
;////////////////////////////////////////////////////////
;FLASH write erase program (SA6)
;////////////////////////////////////////////////////////
RAMPRG  CSEG    ABS=0FFH
        ORG     0BC00H
;       ////////////////////////////////////////
        Initialization
;       ////////////////////////////////////////
        MOVW    RW0,#0500H      ;RW0:RAM space for input data acquisition  00:0500~
        MOVW    RW2,#0000H      ;RW2:Flash memory write address           FD:0000~
        MOV     A,#00H          ;DTB modification
        MOV     DTB,A           ;Bank specification for @RW0
        MOV     A,#0FDH         ;ADB modification 1
        MOV     ADB,A           ;Bank specification for write mode specification
;                                address
        MOV     PDR3,#00H       ;Switch initialization
        MOV     DDR3,#00H
;
WAIT1   BBC     PDR3:0,WAIT1   ;PDR3: 0(write start at high level)
;
;////////////////////////////////////////////
;Write (SA1)
;////////////////////////////////////////////
        MOV     A,PDR1
        MOVW    @RW0+00,A               ;PDR1 data allocation to RAM
        MOV     FMCS,#20H               ;Write mode setting
        MOVW    ADB:COMADR1,#00AAH      ;Flash write command 1
        MOVW    ADB:COMADR2,#0055H      ;Flash write command 2
        MOVW    ADB:COMADR1,#00A0H      ;Flash write command 3
```

```
;
        MOVW    A,@RW0+00               ;Input data (RW0) write to flash memory (RW2)
        MOVW    @RW2+00,A
WRITE   ;Wait time check
;       ////////////////////////////////////////////////////////////////
;       ERROR when the time limit excess check flag is set and toggle operation is
;       in progress
;       ////////////////////////////////////////////////////////////////
        MOVW    A,@RW2+00
        AND     A,#20H                  ;DQ5 time limit check
        BZ      NTOW                    ;Time limit over
        MOVW    A,@RW2+00               ;AH
        MOVW    A,@RW2+00               ;AL
        XORW    A                       ;XOR of AH and AL (1 when the values differ)
        AND     A,#40H                  ;Is the DQ6 toggle bit different?
        BNZ     ERROR                   ;To ERROR when the DQ6 toggle bit is different
;       ////////////////////////////////////
;       Write termination check (FMCS-RDY)
;       ////////////////////////////////////
;       ////////////////////////////////////
NTOW    MOVW    A,FMCS
        AND     A,#10H                  ;Extraction of FMCS RDY bit (bit 4)
        BZ      WRITE                   ;End of write?
        MOV     FMCS,#00H               ;Write mode release
;       ////////////////////////////////////////////////
;       Write data output
;       ////////////////////////////////////////////////
        MOVW    RW2,#0000H              ;Write data output
        MOVW    A,@RW2+00
        MOV     PDR2,A
;
WAIT2   BBC     PDR3:1,WAIT2            ;PDR3: 1(sector erase start at high level)
;
;/////////////////////////////////////////
;Sector erase (SA1)
;/////////////////////////////////////////
        MOV     @RW2+00,#0000H          ;Address initialization
        MOV     FMCS,#20H               ;Erase mode setting
        MOVW    ADB:COMADR1,#00AAH      ;Flash erase command 1
        MOVW    ADB:COMADR2,#0055H      ;Flash erase command 2
        MOVW    ADB:COMADR1,#0080H      ;Flash erase command 3
        MOVW    ADB:COMADR1,#00AAH      ;Flash erase command 4
        MOVW    ADB:COMADR2,#0055H      ;Flash erase command 5
        MOV     @RW2+00,#0030H          ;Issuance of erase command 6 to the sector
                                         to be erased
ELS     ;Wait time check
;       ////////////////////////////////////////////////////////////////
;       ERROR when the time limit excess check flag is set and toggle operation is
;       in progress
;       ////////////////////////////////////////////////////////////////
        MOVW    A,@RW2+00
        AND     A,#20H                  ;DQ5 time limit check
        BZ      NTOE                    ;Time limit over
        MOVW    A,@RW2+00               ;AH  High and Low are alternately output from
        MOVW    A,@RW2+00               ;AL  DQ6 per read during write operation.
        XORW    A                       ;XOR of AH and AL (If the DQ6 value differs,
;                                        write operation is in progress (1)).
        AND     A,#40H                  ;Is the DQ6 toggle bit High?
        BNZ     ERROR                   ;ERROR when the DQ6 toggle bit is High
;       ////////////////////////////////////
;       Erase termination check (FMCS-RDY)
;       ////////////////////////////////////
NTOE    MOVW    A,FMCS                  ;
        AND     A,#10H                  ;Extraction of FMCS RDY bit (bit 4)
        BZ      ELS                     ;End of sector erase?
        MOV     FMCS,#00H               ;FLASH erase mode release
        RETP                            ;Return to the main program
;/////////////////////////////////////////
```

```
;Error
;//////////////////////////////////////////
ERROR   MOV     FMCS,#00H               ;FLASH mode release
        MOV     PDR0,#0FFH              ;Error handling check
        MOV     ADB:COMADR1,#0F0H       ;Reset command (read is enabled)
        RETP                            ;Return to the main program
RAMPRG  ENDS
;//////////////////////////////////////////
VECT    CSEG    ABS=0FFH
        ORG     0FFDCH
        DSL     START
        DB      00H
VECT    ENDS
;
```

CHAPTER 25 EXAMPLES OF F^{2}MC-16LX MB90F591/ MB90F594A SERIAL WRITE CONNECTION

This chapter provides examples of F^{2}MC-16LX MB90F591/MB90F594A serial write connection.

25.1 Basic Configuration of F^{2}MC-16LX MB90F591/MB90F594A Serial Write Connection

25.2 Example of Serial Write Connection (User Power Supply Used)

25.3 Example of Serial Write Connection (Power Supplied from the Writer)

25.4 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)

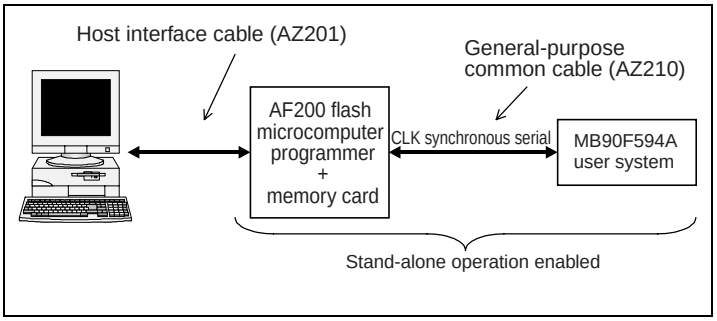
25.5 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Writer)

25.1 Basic Configuration of F^{2}MC-16LX MB90F591/MB90F594A Serial Write Connection

The MB90F591/MB90F594A supports flash ROM serial onboard writing (Fujitsu standard). This section describes the specifications.

■ Basic Configuration of F^{2}MC-16LX MB90F591/MB90F594A Serial Write Connection

The AF200 flash microcomputer programmer from Yokogawa Digital Computer Ltd. is used for Fujitsu standard serial onboard writing.



Note:

Ask the company representative from Yokogawa Digital Computer Ltd. for details about the functions and operations of the AF200 flash microcomputer programmer, general-purpose common cable for connection (AZ210), and connectors.

Table 25.1-1 Pins Used for Fujitsu Standard Serial Onboard Writing

Pin	Function	Additional information
MD2, MD1 MD0	Mode pins	Controls write mode from the flash microcomputer programmer.
X0, X1	Oscillation pins	In write mode, the CPU internal operation clock signal is one multiple of the PLL clock signal frequency. Therefore, because the oscillation clock frequency becomes the internal operation clock signal, the resonator used for serial rewriting is 1 MHz to 16 MHz.
P00, P01	Write program activation pins	-
RSTX	Reset pin	-
SIN3	Serial data input pin	Serial input-output is used.
SOT3	Serial data output pin	
SCK3	Serial clock signal input pin	

Table 25.1-1 Pins Used for Fujitsu Standard Serial Onboard Writing (Continued)

Pin	Function	Additional information
C	C pin	This external capacitor pin is used to stabilize the power supply. Connect a ceramic capacitor of approximately 0.1μF to the outside.
VCC	Supply voltage pin	If the write voltage (5 V±10%) is supplied from the user system, the flash microcomputer programmer need not be connected. Connect so that the power supply of the user side is not short-circuited.
VSS	GND pin	Common to the ground of the flash microcomputer programmer.
HSTX	Hardware standby pin	Input high level during serial write mode.

Even if the P00, SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial writing.

Sections 25.2 to 25.5 present examples the following four types of serial write connection. See each Section as required.

- Serial write connection (user power supply used)
- Serial write connection (power supplied from the writer)
- Minimum connection to the flash microcomputer programmer (user power supply used)
- Minimum connection to the flash microcomputer programmer (power supplied from the writer)

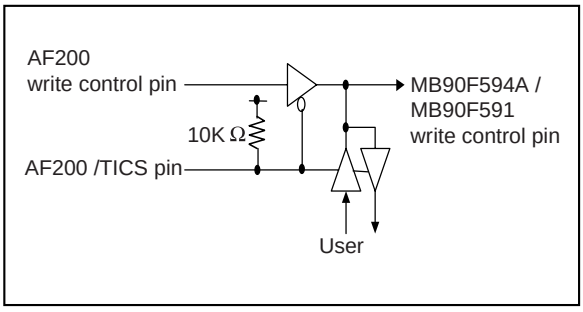


Table 25.1-2 AF200 Flash Microcomputer Programmer System Configuration (Manufactured by Yokogawa Digital Computer Ltd.)

Model	Function
AF200/ACP	Flash microcomputer programmer and 100 V power adapter
AF200/AC2P	Flash microcomputer programmer and power adapter complying with overseas specifications
AZ201	PC/AT RS232C cable
AZ210	Standard target probe (a) length: 1 m
FF001	Fujitsu F^2MC-16LX flash microcomputer control module

Table 25.1-2 AF200 Flash Microcomputer Programmer System Configuration (Manufactured by Yokogawa Digital Computer Ltd.) (Continued)

Model	Function
FF001/P2	2MB PC Card (Option)
FF001/P4	4MB PC Card (Option)

Inquiries: Yokogawa Digital Computer Ltd., Sales Department of Machinery Business Center

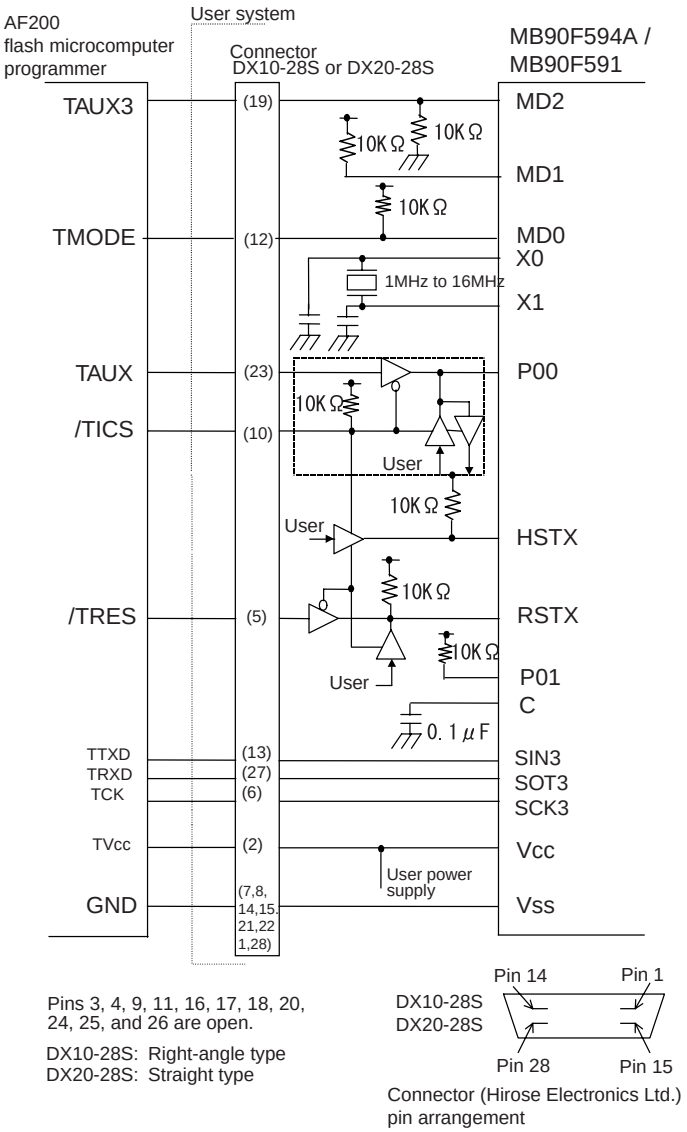
Telephone number: 042-333-6224

25.2 Example of Serial Write Connection (User Power Supply Used)

Figure 25.2-1 is an example of a serial write connection for internal vector modes (single-chip mode and internal ROM external bus mode) when the user power supply is used.
The mode pins MD2, MD1, and MD0 are set to 011.

■ Example of Serial Write Connection (User Power Supply Used)

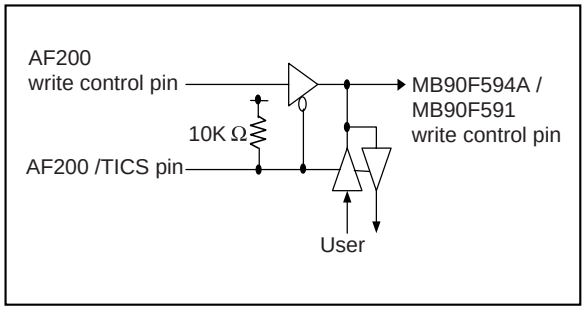
Figure 25.2-1 Example of Serial Write Connection for MB90F591/MB90F594A Internal Vector Modes (User Power Supply Used)



- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit

shown in the figure below is required in the same way that it is for P00. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial writing.

- Connect the AF200 while the user power is off.

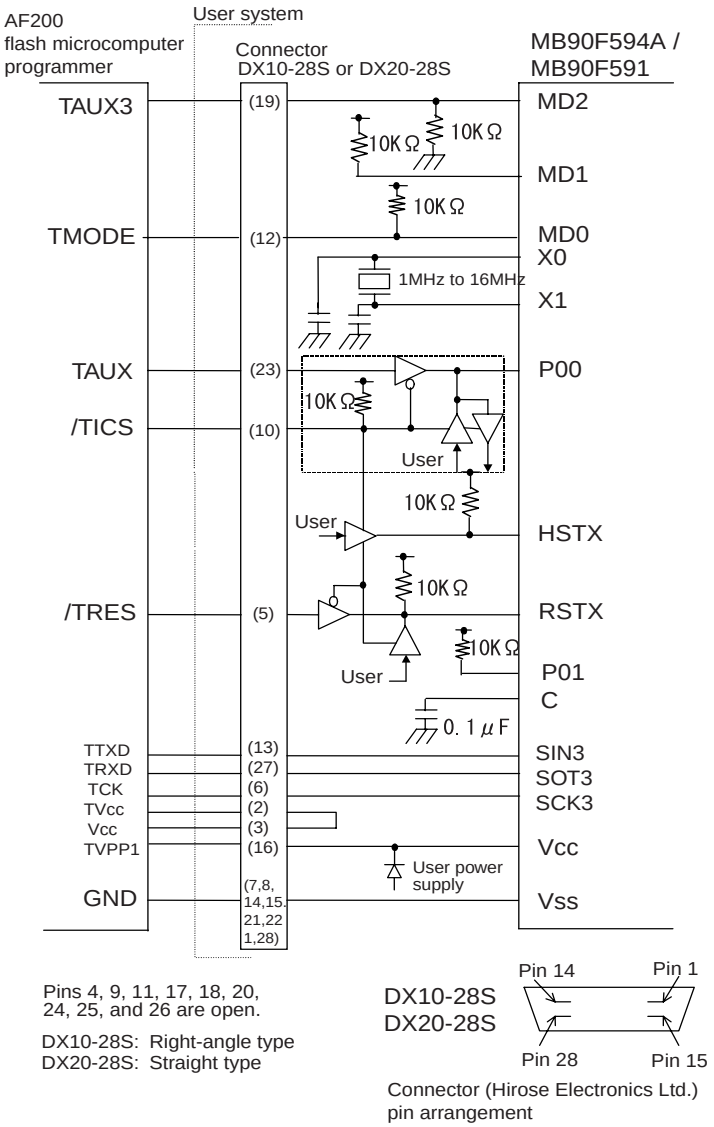


25.3 Example of Serial Write Connection (Power Supplied from the Writer)

Figure 25.3-1 is an example of a serial write connection for internal vector modes (single-chip mode and internal ROM external bus mode) when power is supplied from the writer.
The mode pins MD2, MD1, and MD0 are set to 011.

■ Example of Serial Write Connection (Power Supplied from the Writer)

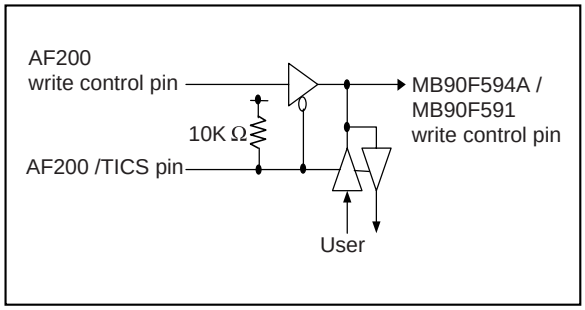
Figure 25.3-1 Example of Serial Write Connection for MB90F591/MB90F594A Internal Vector Modes (Power Supplied from the Writer)



- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit

shown in the figure below is required in the same way that it is for P00. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial writing.

- Connect the AF200 while the user power is off.
- When the write power is supplied from the AF200, be careful not to short-circuit the user power supply.



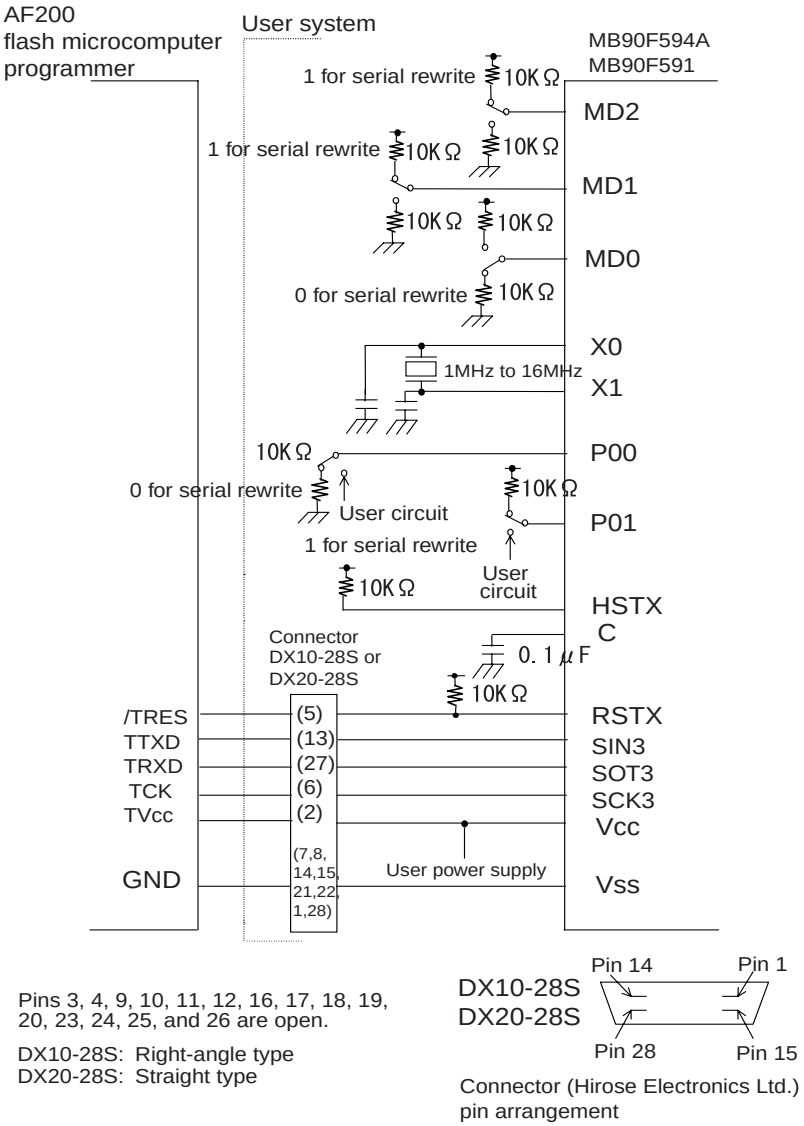
25.4 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)

Figure 25.4-1 is an example of the minimum connection to the flash microcomputer programmer when the user power supply is used.

■ Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)

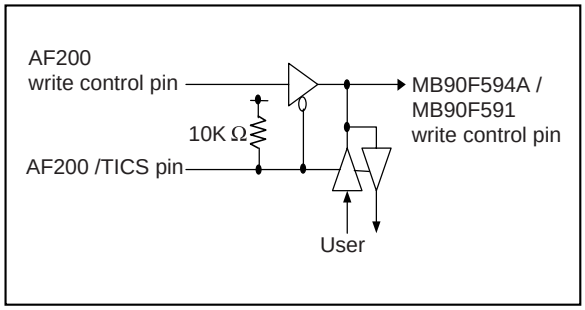
For a flash memory write, the MD2, MD1, MD0, and P00 pins and flash microcomputer programmer need not be connected if the pins are set as described below.

Figure 25.4-1 Example of Minimum Connection to the Flash Microcomputer Programmer (User Power Supply Used)



- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial writing.

- Connect the AF200 while the user power is off.



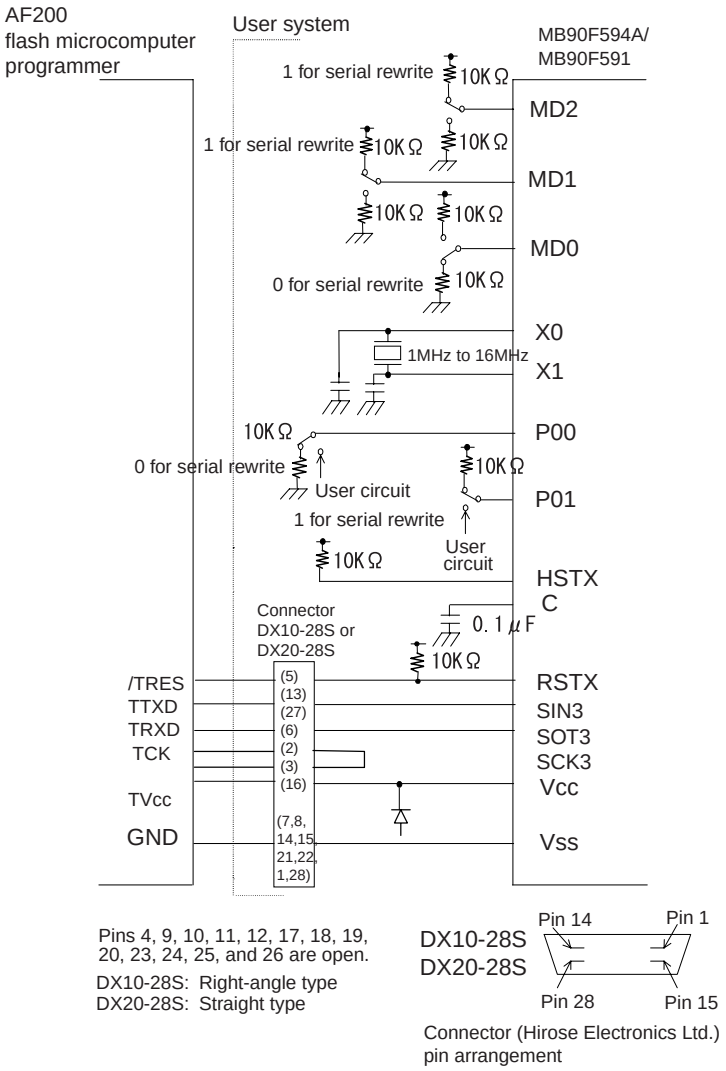
25.5 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Writer)

Figure 25.5-1 is an example of the minimum connection to the flash microcomputer programmer when power is supplied from the writer.

■ Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Writer)

For a flash memory write, the MD2, MD1, MD0, and P00 pins and flash microcomputer programmer need not be connected if the pins are set as described below.

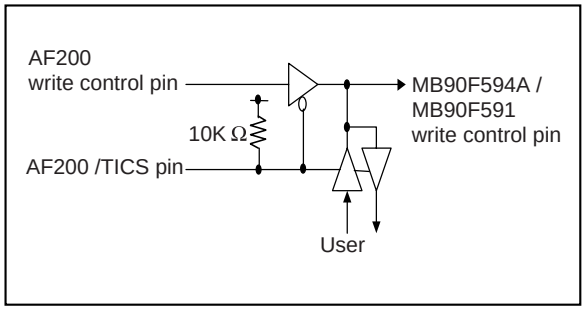
Figure 25.5-1 Example of Minimum Connection to the Flash Microcomputer Programmer (Power Supplied from the Writer)



- Even if the SIN3, SOT3, and SCK3 pins are used for the user system, the control circuit

shown in the figure below is required. The /TICS signal of the flash microcomputer programmer can be used to disconnect the user circuit during serial writing.

- Connect the AF200 while the user power is off.
- When the write power is supplied from the AF200, be careful not to short-circuit the user power supply.



APPENDIX

The appendixes provide I/O maps, instructions, and other information.

A I/O Maps

B Instructions

C Timing Diagram in Flash Memory Mode

D List of MB90590 Interrupt Vectors

APPENDIX A I/O Maps

Table A-1 lists addresses to be assigned to the registers in the peripheral blocks.

■ I/O Maps

Table A-1 I/O Map

Address	Register	Abbrevia -tion	Access	Peripheral	Initial value
00 H	Port 0 data register	PDR0	R/W	Port 0	XXXXXXXX
01 H	Port 1 data register	PDR1	R/W	Port 1	XXXXXXXX
02 H	Port 2 data register	PDR2	R/W	Port 2	XXXXXXXX
03 H	Port 3 data register	PDR3	R/W	Port 3	XXXXXXXX
04 H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX
05 H	Port 5 data register	PDR5	R/W	Port 5	XXXXXXXX
06 H	Port 6 data register	PDR6	R/W	Port 6	XXXXXXXX
07 H	Port 7 data register	PDR7	R/W	Port 7	---XXXXX
08 H	Port 8 data register	PDR8	R/W	Port 8	XXXXXXXX
09 H	Port 9 data register	PDR9	R/W	Port 9	XXXXXXXX
0A to 0F H	Use prohibited				
10 H	Port 0 direction register	DDR0	R/W	Port 0	00000000
11 H	Port 1 direction register	DDR1	R/W	Port 1	00000000
12 H	Port 2 direction register	DDR2	R/W	Port 2	00000000
13 H	Port 3 direction register	DDR3	R/W	Port 3	00000000
14 H	Port 4 direction register	DDR4	R/W	Port 4	00000000
15 H	Port 5 direction register	DDR5	R/W	Port 5	00000000
16 H	Port 6 direction register	DDR6	R/W	Port 6	00000000
17 H	Port 7 direction register	DDR7	R/W	Port 7	00000000
18 H	Port 8 direction register	DDR8	R/W	Port 8	00000000
19 H	Port 9 direction register	DDR9	R/W	Port 9	--000000
1A H	Use prohibited				
1B H	Analog input enable	ADER	R/W	Port 6, A/D	11111111
1C to 1F H	Use prohibited				

Table A-1 I/O Map (Continued)

Address	Register	Abbrevia-tion	Access	Peripheral	Initial value
20 H	Serial mode control 0	UMC0	R/W	UART0	00000100
21 H	Status 0	USR0	R/W		00010000
22 H	Input/output data 0	UIDR0/ UODR0	R/W		XXXXXXXX
23 H	Rate/data 0	URD0	R/W		0000000X
24 H	Serial mode control 1	UMC1	R/W	UART1	00000100
25 H	Status 1	USR1	R/W		00010000
26 H	Input/output data 1	UIDR1/ UODR1	R/W		XXXXXXXX
27 H	Rate/data 1	URD1	R/W		0000000X
28 H	Serial mode control 2	UMC2	R/W	UART2	00000100
29 H	Status 2	USR2	R/W		00010000
2A H	Input/output data 2	UIDR2/ UODR2	R/W		XXXXXXXX
2B H	Rate/data 2	URD2	R/W		0000000X
2C H	Serial mode control	SMCS	R/W	Serial I/O	----0000
2D H	Serial mode control	SMCS	R/W		00000010
2E H	Serial data	SDR	R/W		XXXXXXXX
2F H	Edge selector	SES	R/W		-------0
30 H	External interrupt enable	ENIR	R/W	External interrupt	00000000
31 H	External interrupt request	EIRR	R/W		XXXXXXXX
32 H	External interrupt level	ELVR	R/W		00000000
33 H	External interrupt level	ELVR	R/W		00000000
34 H	A/D control status 0	ADCS0	R/W	A/D converter	00000000
35 H	A/D control status 1	ADCS1	R/W		00000000
36 H	A/D data 0	ADCR0	R		XXXXXXXX
37 H	A/D data 1	ADCR1	R/W		000010XX
38 H	PPG0 operation mode control register	PPGC0	R/W	16-bit programmable pulse generator 0/1	0-000--1
39 H	PPG1 operation mode control register	PPGC1	R/W		0-000001
3A H	PPG0/PPG1 clock select register	PPC01	R/W		00000000
3B H	Use prohibited				

Table A-1 I/O Map (Continued)

Address	Register	Abbrevia -tion	Access	Peripheral	Initial value
3C $_H$	PPG2 operation mode control register	PPGC2	R/W	16-bit programmable pulse generator 2/3	0-000--1
3D $_H$	PPG3 operation mode control register	PPGC3	R/W		0-000001
3E $_H$	PPG2/PPG3 clock select register	PPG23	R/W		00000000
3F $_H$	Use prohibited				
40 $_H$	PPG4 operation mode control register	PPGC4	R/W	16-bit programmable pulse generator 4/5	0-000--1
41 $_H$	PPG5 operation mode control register	PPGC5	R/W		0-000001
42 $_H$	PPG4/PPG5 clock select register	PPG45	R/W		00000000
43 $_H$	Use prohibited				
44 $_H$	PPG6 operation mode control register	PPGC6	R/W	16-bit programmable pulse generator 6/7	0-000--1
45 $_H$	PPG7 operation mode control register	PPGC7	R/W		0-000001
46 $_H$	PPG6/PPG7 clock select register	PPG67	R/W		00000000
47 $_H$	Use prohibited				
48 $_H$	PPG8 operation mode control register	PPGC8	R/W	16-bit programmable pulse generator 8/9	0-000--1
49 $_H$	PPG9 operation mode control register	PPGC9	R/W		0-000001
4A $_H$	PPG8/PPG9 clock select register	PPG89	R/W		00000000
4B $_H$	Use prohibited				
4C $_H$	PPGA operation mode control register	PPGCA	R/W	16-bit programmable pulse generator A/B	0-000--1
4D $_H$	PPGB operation mode control register	PPGCB	R/W		0-000001
4E $_H$	PPGA/PPGB clock select register	PPGAB	R/W		00000000
4F $_H$	Use prohibited				
50 $_H$	Timer control status 0	TMCSR0	R/W	16-bit reload timer 0	00000000
51 $_H$	Timer control status 0	TMCSR0	R/W		----0000
52 $_H$	Timer control status 1	TMCSR1	R/W	16-bit reload timer 1	00000000
53 $_H$	Timer control status 1	TMCSR1	R/W		----0000

Table A-1 I/O Map (Continued)

Address	Register	Abbrevia-tion	Access	Peripheral	Initial value
54 H	Input capture control status 0/1	ICS01	R/W	Input capture 0/1	00000000
55 H	Input capture control status 2/3	ICS23	R/W	Input capture 2/3	00000000
56 H	Input capture control status 4/5	ICS45	R/W	Input capture 4/5	00000000
57 H	Use prohibited				
58 H	Output compare control status 0	OCS0	R/W	Output compare 0/1	0000--00
59 H	Output compare control status 1	OCS1	R/W		---00000
5A H	Output compare control status 2	OCS2	R/W	Output compare 2/3	0000--00
5B H	Output compare control status 3	OCS3	R/W		---00000
5C H	Output compare control status 4	OCS4	R/W	Output compare 4/5	0000--00
5D H	Output compare control status 5	OCS5	R/W		---00000
5E H	Sound control	SGCR	R/W	Sound generator	00000000
5F H	Sound control	SGCR	R/W		0------0
60 H	Watch-dog timer control	WTCR	R/W	Watch-dog timer	000--000
61 H	Watch-dog timer control	WTCR	R/W		00000000
62 H	PWM control 0	PWC0	R/W	Stepping motor controller 0	00000--0
63 H	Use prohibited				
64 H	PWM control 1	PWC1	R/W	Stepping motor controller 1	00000--0
65 H	Use prohibited				
66 H	PWM control 2	PWC2	R/W	Stepping motor controller 2	00000--0
67 H	Use prohibited				
68 H	PWM control 3	PWC3	R/W	Stepping motor controller 3	00000--0
69 H	Use prohibited				
6A to 6C H	Use prohibited				
6D H	Serial I/O prescaler	CDCR	R/W	Prescaler (Serial I/O)	0XXX1111

Table A-1 I/O Map (Continued)

Address	Register	Abbrevia -tion	Access	Peripheral	Initial value
6E $_H$	Timer control	TCCS	R/W	I/O timer	00000000
6F $_H$	ROM mirror	ROMM	W	ROM mirror	XXXXXXX1
70 to 8F $_H$	Reserved for CAN interface 0/1. See the "CAN Controller Hardware Manual."				
90 to 9D $_H$	Use prohibited				
9E $_H$	ROM correction control status	PACSR	R/W	ROM correction	00000000
9F $_H$	Delayed interrupt/release	DIRR	R/W	Delayed interrupt	-------0
A0 $_H$	Low-power mode	LPMCR	R/W	Low-power control circuit	00011000
A1 $_H$	Clock selection	CKSCR	R/W		11111100
A2 to A7$_H$	Use prohibited				
A8 $_H$	Watch-dog control	WDTC	R/W	Watch-dog timer	XXXXX111
A9 $_H$	Time base timer control register	TBTC	R/W	Time base timer	1--00100
AA to AD $_H$	Use prohibited				
AE $_H$	Flash control status (only for MB90F594. Use prohibited for other controllers.)	FMCS	R/W	Flash memory	000X0--0
AF $_H$	Use prohibited				

Table A-1 I/O Map (Continued)

Address	Register	Abbrevia-tion	Access	Peripheral	Initial value
B0 $_{H}$	Interrupt control register 00	ICR00	R/W	Interrupt controller	00000111
B1 $_{H}$	Interrupt control register 01	ICR01	R/W		00000111
B2 $_{H}$	Interrupt control register 02	ICR02	R/W		00000111
B3 $_{H}$	Interrupt control register 03	ICR03	R/W		00000111
B4 $_{H}$	Interrupt control register 04	ICR04	R/W		00000111
B5 $_{H}$	Interrupt control register 05	ICR05	R/W		00000111
B6 $_{H}$	Interrupt control register 06	ICR06	R/W		00000111
B7 $_{H}$	Interrupt control register 07	ICR07	R/W		00000111
B8 $_{H}$	Interrupt control register 08	ICR08	R/W		00000111
B9 $_{H}$	Interrupt control register 09	ICR09	R/W		00000111
BA $_{H}$	Interrupt control register 10	ICR10	R/W		00000111
BB $_{H}$	Interrupt control register 11	ICR11	R/W		00000111
BC $_{H}$	Interrupt control register 12	ICR12	R/W		00000111
BD $_{H}$	Interrupt control register 13	ICR13	R/W		00000111
BE $_{H}$	Interrupt control register 14	ICR14	R/W		00000111
BF $_{H}$	Interrupt control register 15	ICR15	R/W		00000111
C0 to FF $_{H}$	Use prohibited				

Table A-2 I/O Map (19XX Address)

Address	Register	Abbrevia-tion	Access	Peripheral	Initial value
1900 $_{H}$	Reload L	PRLL0	R/W	16-bit programmable pulse generator 0/1	XXXXXXXX
1901 $_{H}$	Reload H	PRLH0	R/W		XXXXXXXX
1902 $_{H}$	Reload L	PRLL1	R/W		XXXXXXXX
1903 $_{H}$	Reload H	PRLH1	R/W		XXXXXXXX
1904 $_{H}$	Reload L	PRLL2	R/W	16-bit programmable pulse generator 2/3	XXXXXXXX
1905 $_{H}$	Reload H	PRLH2	R/W		XXXXXXXX
1906 $_{H}$	Reload L	PRLL3	R/W		XXXXXXXX
1907 $_{H}$	Reload H	PRLH3	R/W		XXXXXXXX

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1908 H	Reload L	PRLL4	R/W	16-bit programmable pulse generator 4/5	XXXXXXXX
1909 H	Reload H	PRLH4	R/W		XXXXXXXX
190A H	Reload L	PRLL5	R/W		XXXXXXXX
190B H	Reload H	PRLH5	R/W		XXXXXXXX
190C H	Reload L	PRLL6	R/W	16-bit programmable pulse generator 6/7	XXXXXXXX
190D H	Reload H	PRLH6	R/W		XXXXXXXX
190E H	Reload L	PRLL7	R/W		XXXXXXXX
190F H	Reload H	PRLH7	R/W		XXXXXXXX
1910 H	Reload L	PRLL8	R/W	16-bit programmable pulse generator 8/9	XXXXXXXX
1911 H	Reload H	PRLH8	R/W		XXXXXXXX
1912 H	Reload L	PRLL9	R/W		XXXXXXXX
1913 H	Reload H	PRLH9	R/W		XXXXXXXX
1914 H	Reload L	PRLLA	R/W	16-bit programmable pulse generator A/B	XXXXXXXX
1915 H	Reload H	PRLHA	R/W		XXXXXXXX
1916 H	Reload L	PRLLB	R/W		XXXXXXXX
1917 H	Reload H	PRLHB	R/W		XXXXXXXX
1918 to 191F H	Use prohibited				
1920 H	Input capture 0	IPCP0	R	Input capture 0/1	XXXXXXXX
1921 H	Input capture 0	IPCP0	R		XXXXXXXX
1922 H	Input capture 1	IPCP1	R		XXXXXXXX
1923 H	Input capture 1	IPCP1	R		XXXXXXXX
1924 H	Input capture 2	IPCP2	R	Input capture 2/3	XXXXXXXX
1925 H	Input capture 2	IPCP2	R		XXXXXXXX
1926 H	Input capture 3	IPCP3	R		XXXXXXXX
1927 H	Input capture 3	IPCP3	R		XXXXXXXX
1928 H	Input capture 4	IPCP4	R	Input capture 4/5	XXXXXXXX
1929 H	Input capture 4	IPCP4	R		XXXXXXXX
192A H	Input capture 5	IPCP5	R		XXXXXXXX
192B H	Input capture 5	IPCP5	R		XXXXXXXX
192D to 192F H	Use prohibited				

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
1930 H	Output compare 0	OCCP0	R/W	Output compare 0/1	XXXXXXXX
1931 H	Output compare 0	OCCP0	R/W		XXXXXXXX
1932 H	Output compare 1	OCCP1	R/W		XXXXXXXX
1933 H	Output compare 1	OCCP1	R/W		XXXXXXXX
1934 H	Output compare 2	OCCP2	R/W	Output compare 2/3	XXXXXXXX
1935 H	Output compare 2	OCCP2	R/W		XXXXXXXX
1936 H	Output compare 3	OCCP3	R/W		XXXXXXXX
1937 H	Output compare 3	OCCP3	R/W		XXXXXXXX
1938 H	Output compare 4	OCCP4	R/W	Output compare 4/5	XXXXXXXX
1939 H	Output compare 4	OCCP4	R/W		XXXXXXXX
193A H	Output compare 5	OCCP5	R/W		XXXXXXXX
193B H	Output compare 5	OCCP5	R/W		XXXXXXXX
193D to 193F H	Use prohibited				
1940 H	Timer 0/reload 0	TMR0/ TMRLR0	R/W	16-bit reload timer 0	XXXXXXXX
1941 H	Timer 0/reload 0	TMR0/ TMRLR0	R/W		XXXXXXXX
1942 H	Timer 1/reload 1	TMR1/ TMRLR1	R/W	16-bit reload timer 1	XXXXXXXX
1943 H	Timer 1/reload 1	TMR1/ TMRLR1	R/W		XXXXXXXX
1944 H	Timer data	TCDT	R/W	I/O timer	00000000
1945 H	Timer data	TCDT	R/W		00000000
1946 H	Frequency data	SGFR	R/W	Sound generator	XXXXXXXX
1947 H	Amplitude data	SGAR	R/W		XXXXXXXX
1948 H	Decrement grade	SGDR	R/W		XXXXXXXX
1949 H	Tone count	SGTR	R/W		XXXXXXXX

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbreviation	Access	Peripheral	Initial value
194A $_H$	Subsecond data	WTBR	R/W	Watchdog timer	XXXXXXXX
194B $_H$	Subsecond data	WTBR	R/W		XXXXXXXX
194C $_H$	Subsecond data	WTBR	R/W		---XXXXX
194D $_H$	Second data	WTSR	R/W		--000000
194E $_H$	Minute data	WTMR	R/W		--000000
194F $_H$	Hour data	WTHR	R/W		---00000
1950 $_H$	PWM1 compare 0	PWC10	R/W	Stepping motor controller 0	XXXXXXXX
1951 $_H$	PWM2 compare 0	PWC20	R/W		XXXXXXXX
1952 $_H$	PWM1 select 0	PWS10	R/W		--000000
1953 $_H$	PWM2 select 0	PWS20	R/W		-0000000
1954 $_H$	PWM1 compare 1	PWC11	R/W	Stepping motor controller 1	XXXXXXXX
1955 $_H$	PWM2 compare 1	PWC21	R/W		XXXXXXXX
1956 $_H$	PWM1 select 1	PWS11	R/W		--000000
1957 $_H$	PWM2 select 1	PWS21	R/W		-0000000
1958 $_H$	PWM1 compare 2	PWC12	R/W	Stepping motor controller 2	XXXXXXXX
1959 $_H$	PWM2 compare 2	PWC22	R/W		XXXXXXXX
195A $_H$	PWM1 select 2	PWS12	R/W		--000000
195B $_H$	PWM2 select 2	PWS22	R/W		-0000000
195C $_H$	PWM1 compare 3	PWC13	R/W	Stepping motor controller 3	XXXXXXXX
195D $_H$	PWM2 compare 3	PWC23	R/W		XXXXXXXX
195E $_H$	PWM1 select 3	PWS13	R/W		--000000
195F $_H$	PWM2 select 3	PWS23	R/W		-0000000
1960 to 19FF $_H$	Used prohibited				
1A00 to 1AFF $_H$	Reserved for CAN interface 0. See the "CAN Controller Hardware Manual."				
1B00 to 1BFF $_H$	Reserved for CAN interface 1. See the "CAN Controller Hardware Manual."				
1C00 to 1CFF $_H$	Reserved for CAN interface 0. See the "CAN Controller Hardware Manual."				
1D00 to 1DFF $_H$	Reserved for CAN interface 1. See the "CAN Controller Hardware Manual."				
1E00 to 1EFF $_H$	Use prohibited				

Table A-2 I/O Map (19XX Address) (Continued)

Address	Register	Abbrevia-tion	Access	Peripheral	Initial value
1EF0 $_H$	ROM correction address 0	PADR0	R/W	ROM correction	XXXXXXXX
1EF1 $_H$	ROM correction address 1	PADR0	R/W		XXXXXXXX
1EF2 $_H$	ROM correction address 2	PADR0	R/W		XXXXXXXX
1EF3 $_H$	ROM correction address 3	PADR1	R/W		XXXXXXXX
1EF4 $_H$	ROM correction address 4	PADR1	R/W		XXXXXXXX
1EF5 $_H$	ROM correction address 5	PADR1	R/W		XXXXXXXX
1EF6 to 1FFF $_H$	Use prohibited				

- Initial value "?" indicates an unused bit, and "X" indicates an undefined value.
- The addresses between 0000_H and $00FF_H$, which are not listed, have been reserved for the main functions of the MCU. The result of read access to these reserved addresses is "X." Write access to these addresses is not allowed.

❍ **Explanation of write and read**

R/W: Both read and write enabled

R: Only read enabled

W: Only write enabled

❍ **Explanation of initial values**

0: The initial value of this bit is "0".

1: The initial value of this bit is "1".

X: The initial value of this bit is undefined.

-: This bit is not used, and the initial value is undefined.

APPENDIX B INSTRUCTIONS

Appendix B describes the instructions used by the F^{2}MC-16LX.

B.1 Instruction Types
B.2 Addressing
B.3 Direct Addressing
B.4 Indirect Addressing
B.5 Number of Execution Cycles
B.6 Effective Address Field
B.7 How to Read the Instruction List
B.8 F^{2}MC-16LX Instruction List
B.9 Instruction Map

B.1 Instruction Types

The F^{2}MC-16LX supports 351 types of instructions. Addressing is enabled by using an effective address field of each instruction or using the instruction code itself.

■ Instruction Types

The F^{2}MC-16LX supports the following 351 types of instructions:

- 41 transfer instructions (byte)
- 38 transfer instructions (word or long word)
- 42 addition/subtraction instructions (byte, word, or long word)
- 12 increment/decrement instructions (byte, word, or long word)
- 11 comparison instructions (byte, word, or long word)
- 11 unsigned multiplication/division instructions (word or long word)
- 11 signed multiplication/division instructions (word or long word)
- 39 logic instructions (byte or word)
- 6 logic instructions (long word)
- 6 sign inversion instructions (byte or word)
- 1 normalization instruction (long word)
- 18 shift instructions (byte, word, or long word)
- 50 branch instructions
- 6 accumulator operation instructions (byte or word)
- 28 other control instructions (byte, word, or long word)
- 21 bit operation instructions
- 10 string instructions

B.2 Addressing

With the F^{2}MC-16LX, the address format is determined by the instruction effective address field or the instruction code itself (implied). When the address format is determined by the instruction code itself, specify an address in accordance with the instruction code used. Some instructions permit the user to select several types of addressing.

■ Addressing

The F^{2}MC-16LX supports the following 23 types of addressing:

- Immediate (#imm)
- Register direct
- Direct branch address (addr16)
- Physical direct branch address (addr24)
- I/O direct (io)
- Abbreviated direct address (dir)
- Direct address (addr16)
- I/O direct bit address (io:bp)
- Abbreviated direct bit address (dir:bp)
- Direct bit address (addr16:bp)
- Vector address (#vct)
- Register indirect (@RWj j = 0 to 3)
- Register indirect with post increment (@RWj+ j = 0 to 3)
- Register indirect with displacement (@RWi + disp8 i = 0 to 7, @RWj+ disp16 j = 0 to 3)
- Long register indirect with displacement (@RLi + disp8 i = 0 to 3)
- Program counter indirect with displacement (@PC + disp16)
- Register indirect with base index (@RW0 + RW7, @RW1 + RW7)
- Program counter relative branch address (rel)
- Register list (rlst)
- Accumulator indirect (@A)
- Accumulator indirect branch address (@A)
- Indirectly-specified branch address (@ear)
- Indirectly-specified branch address (@eam)

■ Effective Address Field

Table B.1 lists the address formats specified by the effective address field.

Table B.2-1 Effective Address Field

Code	Representation			Address format	Default bank
00 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	None
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3			Register indirect	DTB DTB ADB SPB
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+			Register indirect with post increment	DTB DTB ADB SPB
10 11 12 13	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8			Register indirect with 8-bit displacement	DTB DTB ADB SPB
14 15 16 17	@RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8			Register indirect with 8-bit displacement	DTB DTB ADB SPB
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16			Register indirect with 16-bit displacement	DTB DTB ADB SPB
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16			Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	DTB DTB PCB DTB

B.3 Direct Addressing

An operand value, register, or address is specified explicitly in direct addressing mode.

■ Direct Addressing

❍ Immediate addressing (#imm)

Specify an operand value explicitly (#imm4/ #imm8/ #imm16/ #imm32).

Figure B.3-1 Example of immediate addressing (#imm)

MOVW A, #01212H (This instruction stores the operand value in A.)

Before execution	A	2233 4455	
After execution	A	4455 1212	(Some instructions transfer AL to AH.)

❍ Register direct addressing

Specify a register explicitly as an operand. Table B.3-1 lists the registers that can be specified. Figure B.3-2 shows an example of register direct addressing.

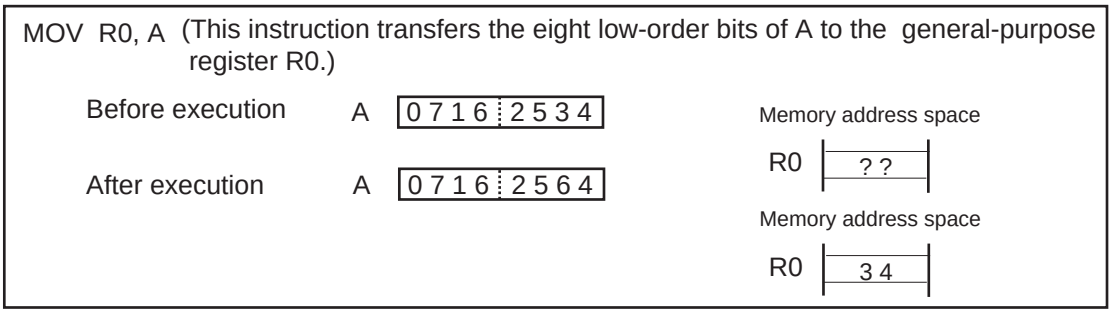
Table B.3-1 Direct Addressing Registers

General-purpose register	Byte	R0, R1, R2, R3, R4, R5, R6, R7
	Word	RW0, RW1, RW2, RW3, RW4, R5W, RW6, RW7
	Long word	RL0, RL1, RL2, RL3
Special-purpose register	Accumulator	A, AL
	Pointer	SP*
	Bank	PCB, DTB, USB, SSB, ADB
	Page	DPR
	Control	PS, CCR, RP, ILM

*1 One of the user stack pointer (USP) and system stack pointer (SSP) is selected and used depending on the value of the S flag bit in the condition code register (CCR). For branch instructions, the program counter (PC) is not specified in an instruction operand but is specified

implicitly.

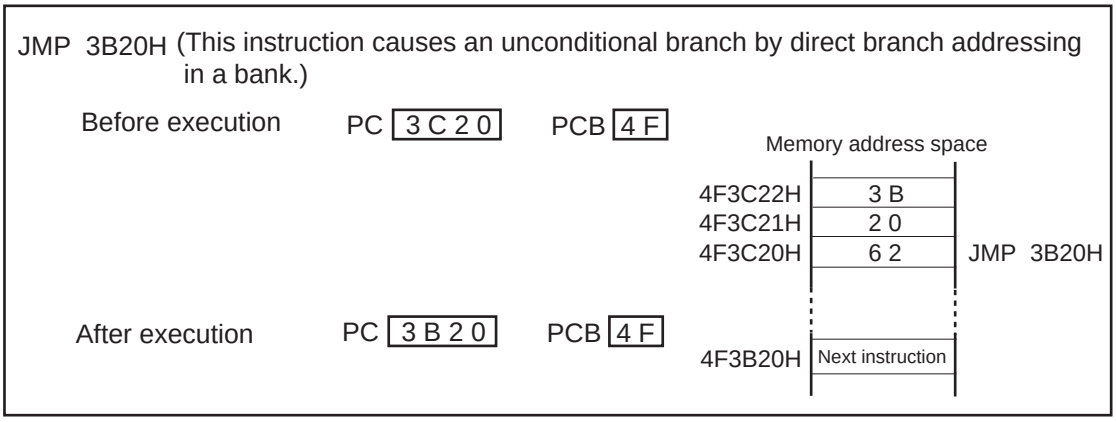
Figure B.3-2 Example of Register Direct Addressing



❍ **Direct branch addressing (addr16)**

Specify an offset explicitly for the branch destination address. The size of the offset is 16 bits, which indicates the branch destination in the logical address space. Direct branch addressing is used for an unconditional branch, subroutine call, or software interrupt instruction. Bits 23 to 16 of the address are specified by the program bank register (PCB).

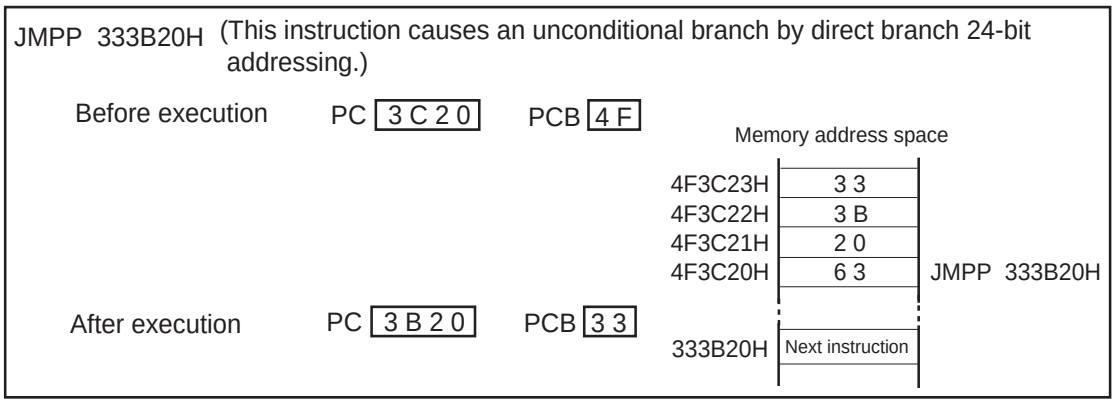
Figure B.3-3 Example of Direct Branch Addressing (addr16)



❍ **Physical direct branch addressing (addr24)**

Specify an offset explicitly for the branch destination address. The size of the offset is 24 bits. Physical direct branch addressing is used for unconditional branch, subroutine call, or software interrupt instruction.

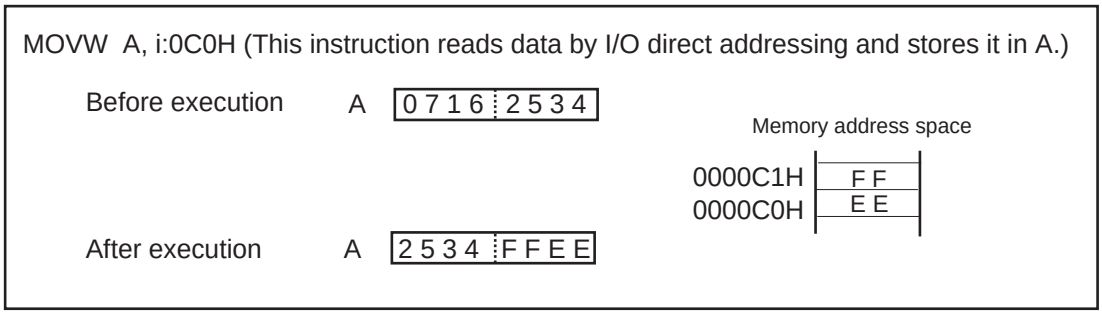
Figure B.3-4 Example of Direct Branch Addressing (addr24)



❍ I/O direct addressing (io)

Specify an 8-bit offset explicitly for the memory address in an operand. The I/O address space in the physical address space from 000000H to 0000FFH is accessed regardless of the data bank register (DTB) and direct page register (DPR). A bank select prefix for bank addressing is invalid if specified before an instruction using I/O direct addressing.

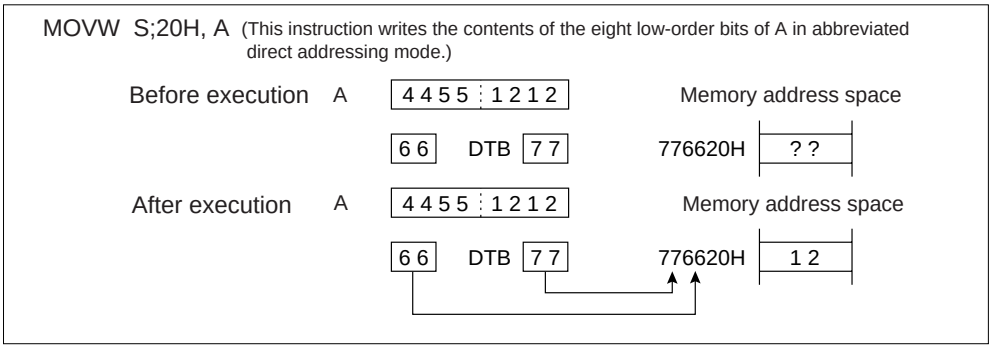
Figure B.3-5 Example of I/O Direct Addressing (io)



❍ Abbreviated direct addressing (dir)

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB).

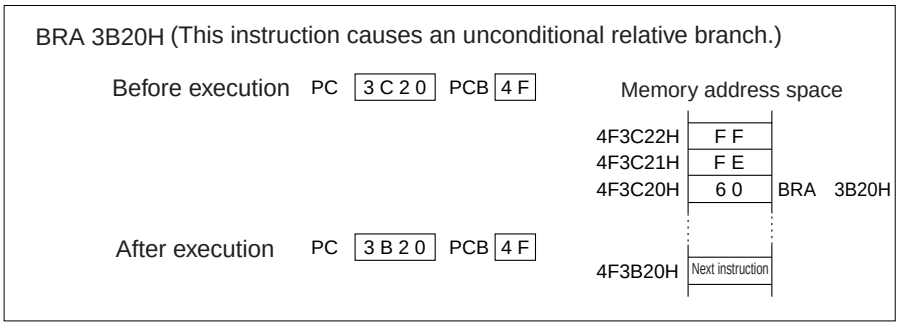
Figure B.3-6 Example of Abbreviated Direct Addressing (dir)



❍ Direct addressing (addr16)

Specify the 16 low-order bits of a memory address explicitly in an operand. Address bits 16 to 23 are specified by the data bank register (DTB). A prefix instruction for access space addressing is invalid for this mode of addressing.

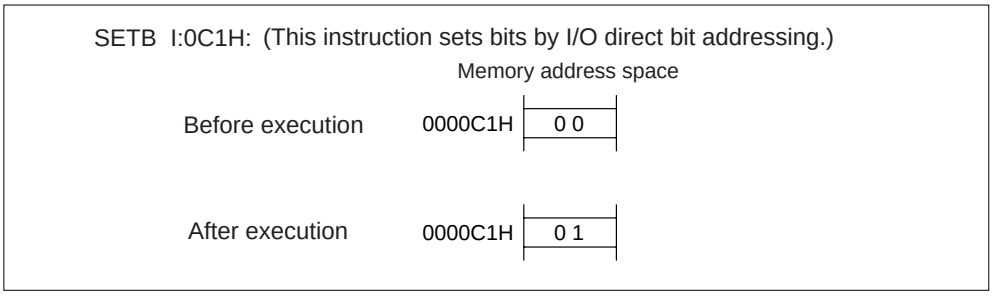
Figure B.3-7 Example of Direct Addressing (addr16)



❍ **I/O direct bit addressing (io:bp)**

Specify bits in physical addresses 000000H to 0000FFH explicitly. Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

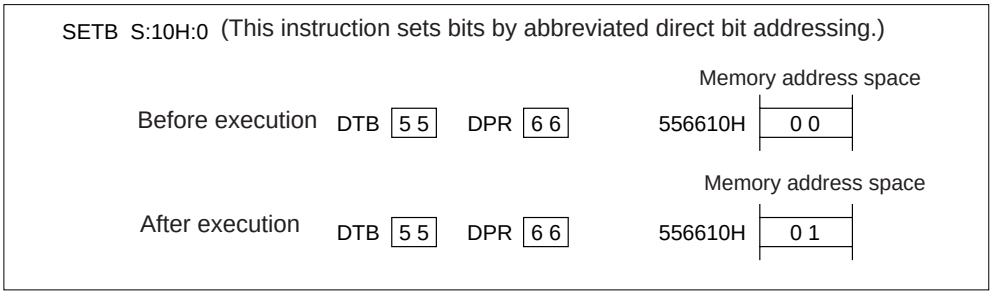
Figure B.3-8 Example of I/O Direct Bit Addressing (io:bp)



❍ **Abbreviated direct bit addressing (dir:bp)**

Specify the eight low-order bits of a memory address explicitly in an operand. Address bits 8 to 15 are specified by the direct page register (DPR). Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

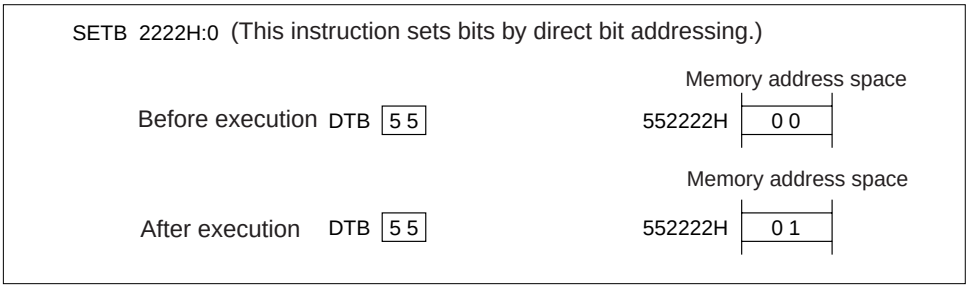
Figure B.3-9 Example of Abbreviated Direct Bit Addressing (dir:bp)



❍ **Direct bit addressing (addr16:bp)**

Specify arbitrary bits in 64 kilobytes explicitly. Address bits 16 to 23 are specified by the data bank register (DTB). Bit positions are indicated by ":bp", where the larger number indicates the most significant bit (MSB) and the lower number indicates the least significant bit (LSB).

Figure B.3-10 Example of Direct Bit addressing (addr16:bp)



❍ **Vector Addressing (#vct)**

Specify vector data in an operand to indicate the branch destination address. There are two

sizes for vector numbers: 4 bits and 8 bits. Vector addressing is used for a subroutine call or software interrupt instruction.

Figure B.3-11 Example of Vector Addressing (#vct)

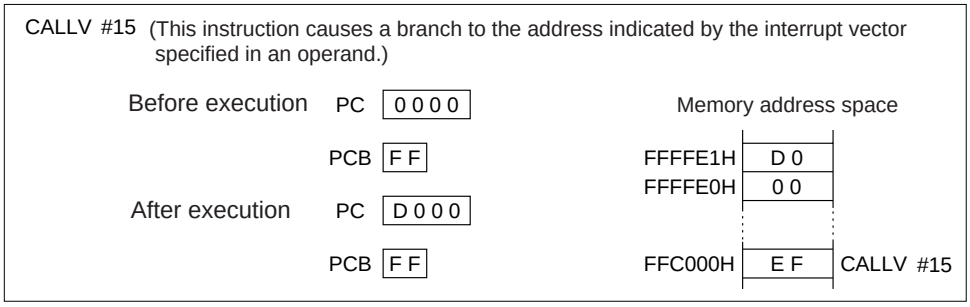


Table B.3-2 CALLV Vector List

Instruction	Vector address L	Vector address H
CALLV #0	XFFFEH	XXFFFFH
CALLV #1	XFFFCH	XXFFFDH
CALLV #2	XFFFAH	XXFFFBH
CALLV #3	XFFF8H	XXFFF9H
CALLV #4	XFFF6H	XXFFF7H
CALLV #5	XFFF4H	XXFFF5H
CALLV #6	XFFF2H	XXFFF3H
CALLV #7	XFFF0H	XXFFF1H
CALLV #8	XFFEEH	XXFFEFH
CALLV #9	XFFECH	XXFFEDH
CALLV #10	XFFEAH	XXFFEBH
CALLV #11	XFFE8H	XXFFE9H
CALLV #12	XFFE6H	XXFFE7H
CALLV #13	XFFE4H	XXFFE5H
CALLV #14	XFFE2H	XXFFE3H
CALLV #15	XFFE0H	XXFFE1H

Note: A PCB register value is set in XX.

Note:

When the program bank register (PCB) is FF_H, the vector area overlaps the vector area of INT #vct8 (#0 to #7). Use vector addressing carefully (see Table B.3).

B.4 Indirect Addressing

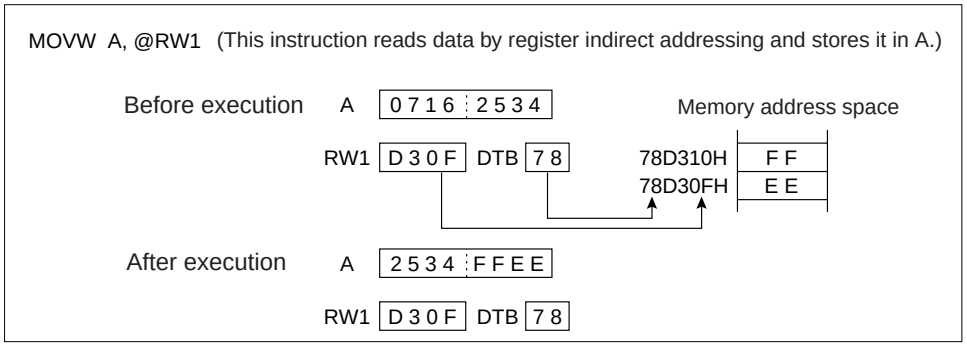
In indirect addressing mode, an address is specified indirectly by the address data of an operand.

■ Indirect Addressing

❍ Register indirect addressing (@RWj j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

Figure B.4-1 Example of Register Indirect Addressing (@RWj j = 0 to 3)



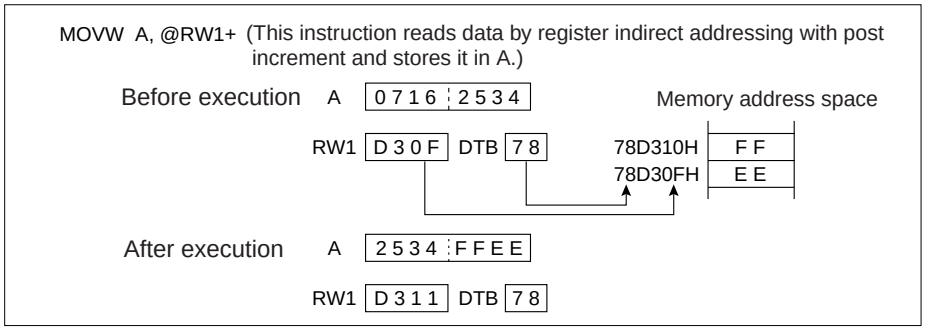
❍ Register indirect addressing with post increment (@RWj+ j = 0 to 3)

Memory is accessed using the contents of general-purpose register RWj as an address. After operand operation, RWj is incremented by the operand size (1 for a byte, 2 for a word, or 4 for a long word). Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0 or RW1 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 is used, or additional data bank register (ADB) when RW2 is used.

If the post increment results in the address of the register that specifies the increment, the incremented value is referenced after that. In this case, if the next instruction is a write instruction, priority is given to writing by an instruction and, therefore, the register that would be incremented becomes write data.

Figure B.4-2 Example of Register Indirect Addressing with Post Increment (@RWj + j = 0

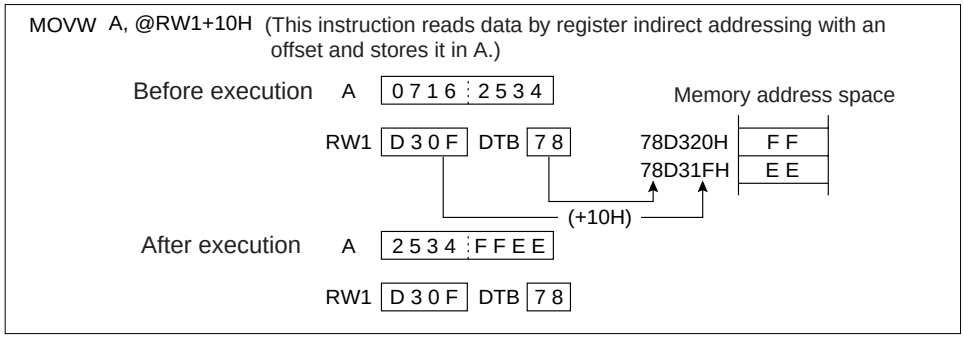
to 3)



❍ **Register indirect addressing with offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)**

Memory is accessed using the address obtained by adding an offset to the contents of general-purpose register RWj. Two types of offset, byte and word offsets, are used. They are added as signed numeric values. Address bits 16 to 23 are indicated by the data bank register (DTB) when RW0, RW1, RW4, or RW5 is used, system stack bank register (SSB) or user stack bank register (USB) when RW3 or RW7 is used, or additional data bank register (ADB) when RW2 or RW6 is used.

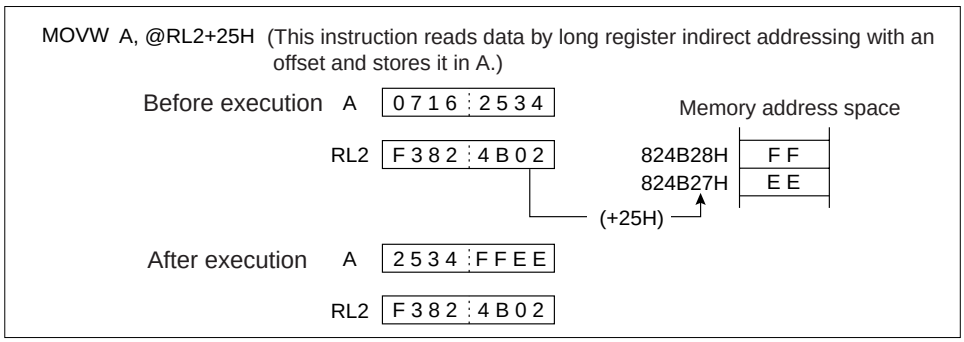
Figure B.4-3 Example of Register Indirect Addressing with Offset (@RWi + disp8 i = 0 to 7, @RWj + disp16 j = 0 to 3)



❍ **Long register indirect addressing with offset (@RLi + disp8 i = 0 to 3)**

Memory is accessed using the address that is the 24 low-order bits obtained by adding an offset to the contents of general-purpose register RLi. The offset is 8-bits long and is added as a signed numeric value.

Figure B.4-4 Example of Long Register Indirect Addressing with Offset (@RLi + disp8 i = 0 to 3)

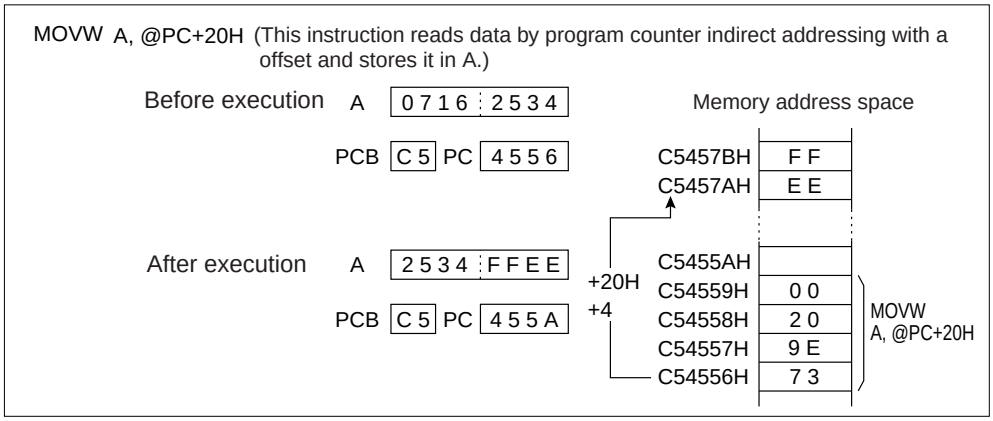


❍ **Program counter indirect addressing with offset (@PC + disp16)**

Memory is accessed using the address indicated by (instruction address + 4 + disp16). The offset is one word long. Address bits 16 to 23 are specified by the program bank register (PCB). Note that the operand address of each of the following instructions is not deemed to be (next instruction address + disp16):

- DBNZ eam, rel DWBNZ eam, rel
- CBNE eam, #imm8, rel CWBNE eam, #imm16, rel
- MOV eam, #imm8 MOVW eam, #imm16

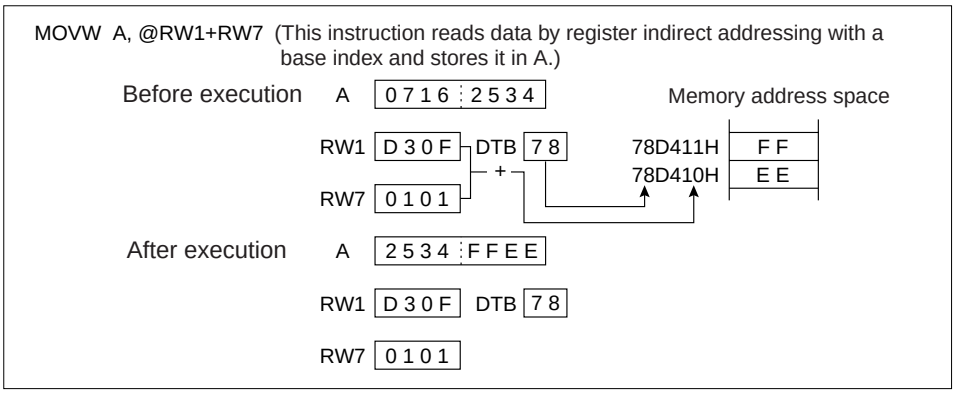
Figure B.4-5 Example of Program Counter Indirect Addressing with Offset (@PC + disp16)



❍ **Register indirect addressing with base index (@RW0 + RW7, @RW1 + RW7)**

Memory is accessed using the address determined by adding RW0 or RW1 to the contents of general-purpose register RW7. Address bits 16 to 23 are indicated by the data bank register (DTB).

Figure B.4-6 Example of Register Indirect Addressing with Base Index (@RW0 + RW7, @RW1 + RW7)

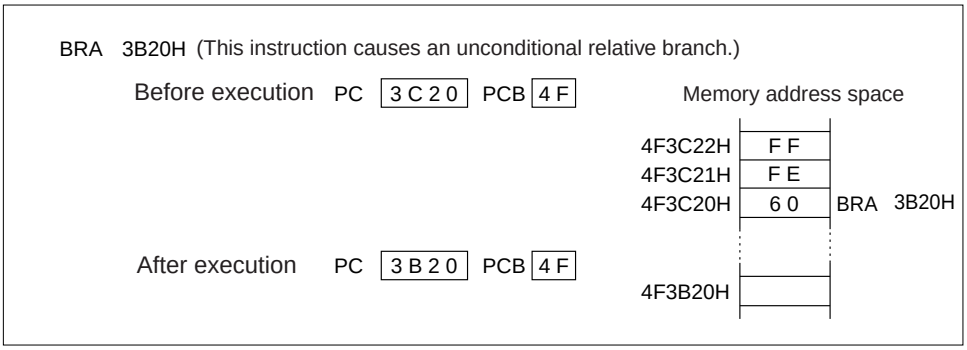


❍ **Program counter relative branch addressing (rel)**

The address of the branch destination is a value determined by adding an 8-bit offset to the program counter (PC) value. If the result of addition exceeds 16 bits, bank register incrementing or decrementing is not performed and the excess part is ignored, and therefore the address is contained within a 64-kilobyte bank. This addressing is used for both conditional and unconditional branch instructions. Address bits 16 to 23 are indicated by the program bank

register (PCB).

Figure B.4-7 Example of Program Counter Relative Branch Addressing (rel)



❍ **Register list (rlst)**

Specify a register to be pushed onto or popped from a stack.

Figure B.4-8 Configuration of the Register List

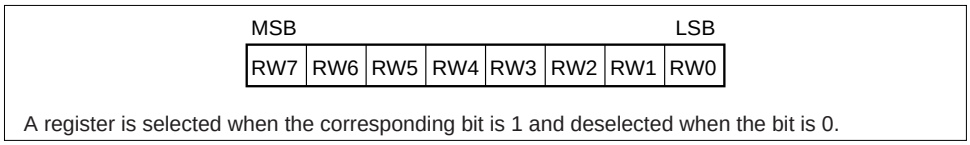
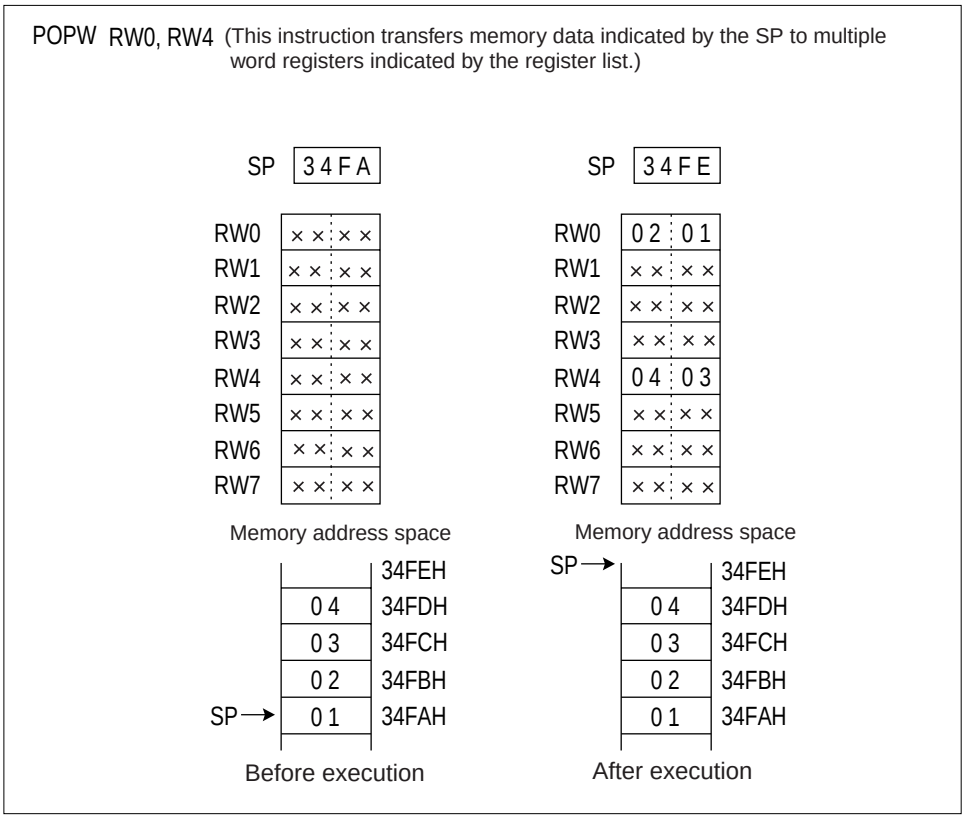


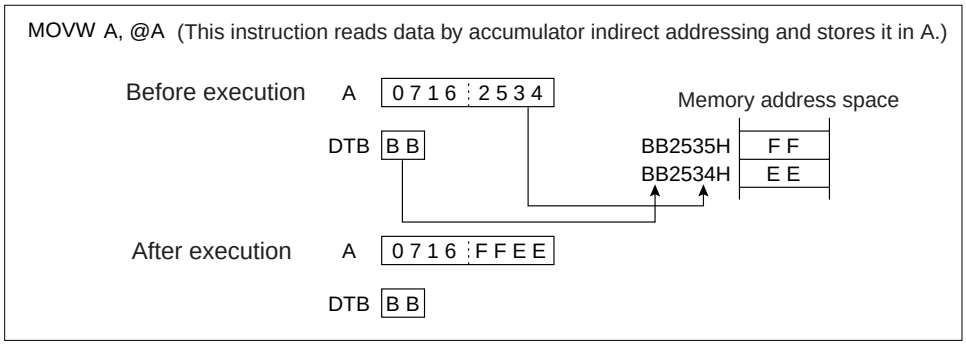
Figure B.4-9 Example of Register List (rlist)



❍ **Accumulator indirect addressing (@A)**

Memory is accessed using the address indicated by the contents of the low-order bytes (16 bits) of the accumulator (AL). Address bits 16 to 23 are specified by a mnemonic in the data bank register (DTB).

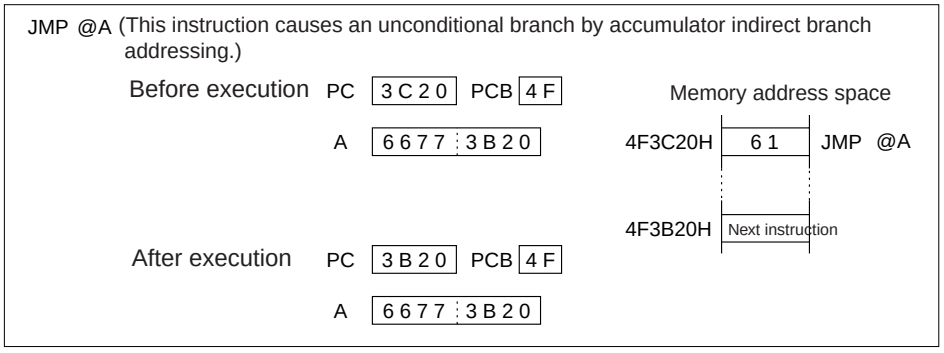
Figure B.4-10 Example of Accumulator Indirect Addressing (@A)



❍ **Accumulator indirect branch addressing (@A)**

The address of the branch destination is the content (16 bits) of the low-order bytes (AL) of the accumulator. It indicates the branch destination in the bank address space. Address bits 16 to 23 are specified by the program bank register (PCB). For the Jump Context (JCTX) instruction, however, address bits 16 to 23 are specified by the data bank register (DTB). This addressing is used for unconditional branch instructions.

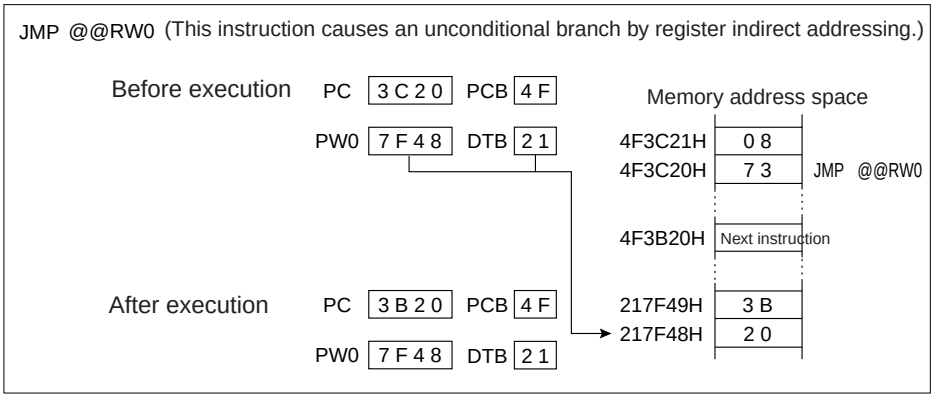
Figure B.4-11 Example of Accumulator Indirect Branch Addressing (@A)



❍ **Indirect specification branch addressing (@ear)**

The address of the branch destination is the word data at the address indicated by ear.

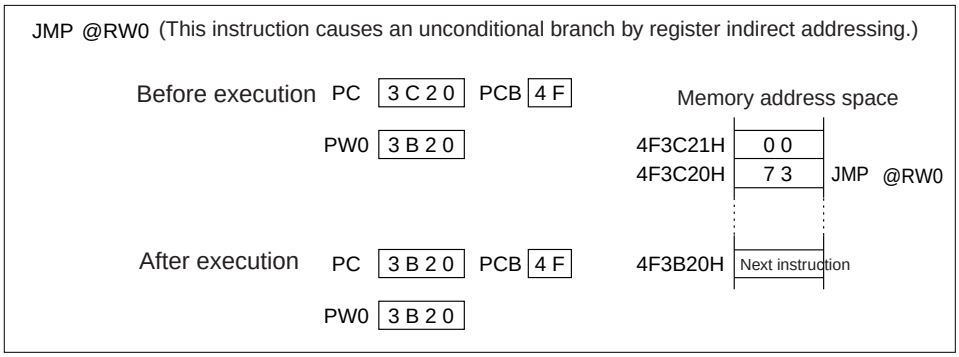
Figure B.4-12 Example of Indirect Specification Branch Addressing (@ear)



❍ **Indirect specification branch addressing (@eam)**

The address of the branch destination is the word data at the address indicated by eam.

Figure B.4-13 Example of Indirect Specification Branch Addressing (@eam)



B.5 Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch.

■ Execution Cycle Count

The number of cycles required for instruction execution (execution cycle count) is obtained by adding the number of cycles required for each instruction, "correction value" determined by the condition, and the number of cycles for instruction fetch. In the mode of fetching an instruction from memory such as internal ROM connected to a 16-bit bus, the program fetches the instruction being executed in word increments. Therefore, intervening in data access increases the execution cycle count.

Similarly, in the mode of fetching an instruction from memory connected to an 8-bit external bus, the program fetches every byte of an instruction being executed. Therefore, intervening in data access increases the execution cycle count. In CPU intermittent operation mode, access to a general-purpose register, internal ROM, internal RAM, internal I/O, or external data bus causes the clock to the CPU to halt for the cycle count specified by the CG0 and CG1 bits of the low power consumption mode control register. Therefore, for the cycle count required for instruction execution in CPU intermittent operation mode, add the "access count x cycle count for the halt" as a correction value to the normal execution count.

■ Calculating the Execution Cycle Count

Table B.5-1 lists execution cycle counts and Tables B.5-2 and B.5-3 summarize correction value data.

Table B.5-1 Execution Cycle Counts in Each Addressing Mode

Code	Operand	(a)*	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
00 | 07	Ri Rwi RLi	See the instruction list.	See the instruction list.
08 | 0B	@RWj	2	1
0C | 0F	@RWj+	4	2
10 | 17	@RWi+disp8	2	1

Table B.5-1 Execution Cycle Counts in Each Addressing Mode (Continued)

Code	Operand	(a)*	Register access count in each addressing mode
		Execution cycle count in each addressing mode	
18 | 1B	@RWi+disp16	2	1
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16	4 4 2 1	2 2 0 0

*: (a) is used for ~ (cycle count) and B (correction value) in B-8, "F^{2}MC-16LX Instruction List."

Table B.5-2 Cycle Count Correction Values for Counting Execution Cycles

Operand	(b) byte*1		(c) word*1		(d) long*1	
	Cycle count	Access count	Cycle count	Access count	Cycle count	Access count
Internal register	+0	1	+0	1	+0	2
Internal memory Even address	+0	1	+0	1	+0	2
Internal memory Odd address	+0	1	+2	2	+4	4
External data bus 16-bit even address	+1	1	+1	1	+2	2
External data bus 16-bit odd address	+1	1	+4	2	+8	4
External data bus (*2) 28 bits	+1	1	+4	2	+8	4

*1: (b), (c), and (d) are used for ~ (cycle count) and B (correction value) in B.8, "F^{2}MC-16LX Instruction List."

*2: When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

Table B.5-3 Cycle Count Correction Values for Counting Instruction Fetch Cycles

Instruction	Byte boundary	Word boundary
Internal memory	—	+2
External data bus 16 bits	—	+3
External data bus 8 bits	+3	—

Notes:

1. When an external data bus is used, the cycle counts during which an instruction is made to wait by ready input or automatic ready must also be added.

2. Actually, instruction execution is not delayed by every instruction fetch. Therefore, use the correction values to calculate the worst case.

B.6 Effective Address Field

Table B.6-1 shows the effective address field.

■ **Effective Address Field**

Table B.6-1 Effective Address Field

Code	Representation			Address format	Byte count of extended address part (*1)
00 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct: Individual parts correspond to the byte, word, and long word types in order from the left.	—
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3			Register indirect	0
0C 0D 0E 0F	@RW0+ @RW1+ @RW2+ @RW3+			Register indirect with post increment	0
10 11 12 13 14 15 16 17	@RW0+disp8 @RW1+disp8 @RW2+disp8 @RW3+disp8 @RW4+disp8 @RW5+disp8 @RW6+disp8 @RW7+disp8			Register indirect with 8-bit displacement	1
18 19 1A 1B	@RW0+disp16 @RW1+disp16 @RW2+disp16 @RW3+disp16			Register indirect with 16-bit displacement	2
1C 1D 1E 1F	@RW0+RW7 @RW1+RW7 @PC+disp16 addr16			Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	0 0 2 2

*1 Each byte count of the extended address part applies to + in the # (byte count) column in the

"F^{2}MC-16LX Instruction List" in Appendix B.8.

B.7 How to Read the Instruction List

Table B.7-1describes the items used in the F^{2}MC-16LX Instruction List, and Table B.7-2 describes the symbols used in the same list.

■ Description of instruction presentation items and symbols

Table B.7-1 Description of Items in the Instruction List

Item	Description
Mnemonic	Uppercase, symbol: Represented as is in the assembler. Lowercase: Rewritten in the assembler. Number following lowercase: Indicates bit length in the instruction.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table B.2a for the alphabetical letters in items.
RG	Indicates the number of times a register access is performed during instruction execution. The number is used to calculate the correction value for CPU intermittent operation.
B	Indicates the correction value used to calculate the actual number of cycles during instruction execution. The actual number of cycles during instruction execution can be determined by adding the value in the ~ column to this value.
Operation	Indicates the instruction operation.
LH	Indicates the special operation for bits 15 to 08 of the accumulator. Z: Transfers 0. X: Transfers after sign extension. -: No transfer
AH	Indicates the special operation for the 16 high-order bits of the accumulator. *: Transfers from AL to AH. -: No transfer Z: Transfers 00 to AH. X: Transfers 00H or FFH to AH after AL sign extension.

Table B.7-1 Description of Items in the Instruction List (Continued)

Item	Description
I	Each indicates the state of each flag: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), C (carry). *: Changes upon instruction execution. -: No change Z: Set upon instruction execution. X: Reset upon instruction execution.
S	
T	
N	
Z	
V	
C	
RMW	Indicates whether the instruction is a Read Modify Write instruction (reading data from memory by the I instruction and writing the result to memory). *: Read Modify Write instruction -: Not Read Modify Write instruction Note: Cannot be used for an address that has different meanings between read and write operations.

Table B.7-2 Explanation on Symbols in the Instruction List

Symbol	Explanation
A	The bit length used varies depending on the 32-bit accumulator instruction. low-order bits of byte AL 16 bits of word AL long AL: 32 bits of AH
AH AL	16 high-order bits of A 16 low-order bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	Program bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7

Table B.7-2 Explanation on Symbols in the Instruction List (Continued)

Symbol	Explanation
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	Abbreviated direct addressing Direct addressing Physical direct addressing Bits 0 to 15 of addr24 Bits 16 to 23 of addr24
io	I/O area (000000H to 0000FFH)
#imm4 #imm8 #imm16 #imm32 ext (imm8)	4-bit immediate data 8-bit immediate data 16-bit immediate data 32-bit immediate data 16-bit data obtained by sign extension of 8-bit immediate data
disp8 disp16	8-bit displacement 16-bit displacement
bp	Bit offset
vct4 vct8	Vector number (0 to 15) Vector number (0 to 255)
( ) b	Bit address
rel car eam	PC relative branch effective addressing (code 00 to 07) Effective addressing (code 08 to 1F)
rlst	Register list

B.8 F²MC-16LX Instruction List

Tables B.8-1 to B.9-19 list the instructions used by the F²MC-16LX.

Table B.8-1 41 Transfer Instructions (byte)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV	A,dir	2	3	0	(b)	byte (A) <-- (dir)	Z	*	-	-	*	*	-	-	-	-
MOV	A,addr16	3	4	0	(b)	byte (A) <-- (addr16)	Z	*	-	-	*	*	-	-	-	-
MOV	A,Ri	1	2	1	0	byte (A) <-- (Ri)	Z	*	-	-	*	*	-	-	-	-
MOV	A,ear	2	2	1	0	byte (A) <-- (ear)	Z	*	-	-	*	*	-	-	-	-
MOV	A,eam	2+	3+(a)	0	(b)	byte (A) <-- (eam)	Z	*	-	-	*	*	-	-	-	-
MOV	A,io	2	3	0	(b)	byte (A) <-- (io)	Z	*	-	-	*	*	-	-	-	-
MOV	A,#imm8	2	2	0	0	byte (A) <-- imm8	Z	*	-	-	*	*	-	-	-	-
MOV	A,@A	2	3	0	(b)	byte (A) <-- ((A))	Z	-	-	-	*	*	-	-	-	-
MOV	A,@RLi+disp8	3	10	2	(b)	byte (A) <-- ((RLi)+disp8)	Z	*	-	-	*	*	-	-	-	-
MOVN	A,#imm4	1	1	0	0	byte (A) <-- imm4	Z	*	-	-	R	*	-	-	-	-
MOVX	A,dir	2	3	0	(b)	byte (A) <-- (dir)	X	*	-	-	-	*	*	-	-	-
MOVX	A,addr16	3+	4	0	(b)	byte (A) <-- (addr16)	X	*	-	-	-	*	*	-	-	-
MOVX	A,Ri	2	2	1	0	byte (A) <-- (Ri)	X	*	-	-	-	*	*	-	-	-
MOVX	A,ear	2	2	1	0	byte (A) <-- (ear)	X	*	-	-	-	*	*	-	-	-
MOVX	A,eam	2+	3+(a)	0	(b)	byte (A) <-- (eam)	X	*	-	-	-	*	*	-	-	-
MOVX	A,io	2	3	0	(b)	byte (A) <-- (io)	X	*	-	-	-	*	*	-	-	-
MOVX	A,#imm8	2	2	0	0	byte (A) <-- imm8	X	*	-	-	-	*	*	-	-	-
MOVX	A,@A	2	3	0	(b)	byte (A) <-- ((A))	X	-	-	-	-	*	*	-	-	-
MOVX	A,@RWi+disp8	2	5	1	(b)	byte (A) <-- ((RWi)+disp8)	X	*	-	-	-	*	*	-	-	-
MOVX	A,@RLi+disp8	3	10	2	(b)	byte (A) <-- ((RLi)+disp8	X	*	-	-	-	*	*	-	-	-
MOV	dir,A	2	3	0	(b)	byte (dir) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	addr16,A	3	4	0	(b)	byte (addr16) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,A	1	2	1	0	byte (Ri) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	ear,A	2	2	1	0	byte (ear) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	eam,A	2+	3+(a)	0	(b)	byte (eam) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	io,A	2	3	0	(b)	byte (io) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	RLi+disp8,A	3	10	2	(b)	byte ((RLi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,ear	2	3	2	0	byte (Ri) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,eam	2+	4+(a)	1	(b)	byte (Ri) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOV	ear,Ri	2	4	2	0	byte (ear) <-- (Ri)	-	-	-	-	-	*	*	-	-	-
MOV	eam,Ri	2+	5+(a)	1	(b)	byte (eam) <-- (Ri)	-	-	-	-	-	*	*	-	-	-
MOV	Ri,#imm8	2	2	1	0	byte (Ri) <-- imm8	-	-	-	-	-	*	*	-	-	-
MOV	io,#imm8	3	5	0	(b)	byte (io) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	dir,#imm8	3	5	0	(b)	byte (dir) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	ear,#imm8	3	2	1	0	byte (ear) <-- imm8	-	-	-	-	-	*	*	-	-	-
MOV	eam,#imm8	3+	4+(a)	0	(b)	byte (eam) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	@AL,AH	2	3	0	(b)	byte ((A)) <-- (AH)	-	-	-	-	-	*	*	-	-	-
XCH	A,ear	2	4	2	0	byte (A) <--> (ear)	Z	-	-	-	-	-	-	-	-	-
XCH	A,eam	2+	5+(a)	0	2×(b)	byte (A) <--> (eam)	Z	-	-	-	-	-	-	-	-	-
XCH	Ri,ear	2	7	4	0	byte (Ri) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCH	Ri,eam	2+	9+(a)	2	2×(b)	byte (Ri) <--> (eam)	-	-	-	-	-	-	-	-	-	-

See Tables B.5-1 and B.5 -2 for information on (a) to (d) in the table.

Table B.8-2 38 Transfer Instructions (byte)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW	A,dir	2	3	0	(c)	word (A) <-- (dir)	-	*	-	-	-	*	*	-	-	-
MOVW	A,addr16	3	4	0	(c)	word (A) <-- (addr16)	-	*	-	-	-	*	*	-	-	-
MOVW	A,SP	1	1	0	0	word (A) <-- (SP)	-	*	-	-	-	*	*	-	-	-
MOVW	A,RWi	1	2	1	0	word (A) <-- (RWi)	-	*	-	-	-	*	*	-	-	-
MOVW	A,ear	2	2	1	0	word (A) <-- (ear)	-	*	-	-	-	*	*	-	-	-
MOVW	A,eam	2+	3+(a)	0	(c)	word (A) <-- (eam)	-	*	-	-	-	*	*	-	-	-
MOVW	A,io	2	3	0	(c)	word (A) <-- (io)	-	*	-	-	-	*	*	-	-	-
MOVW	A,@A	2	3	0	(c)	word (A) <-- ((A))	-	-	-	-	-	*	*	-	-	-
MOVW	A,#imm16	3	2	0	0	word (A) <-- imm16	-	*	-	-	-	*	*	-	-	-
MOVW	A,@RWi+disp8	2	5	1	(c)	word (A) <-- ((RWi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW	A,@RLi+disp8	3	10	2	(c)	word (A) <-- ((RLi)+disp8)	-	*	-	-	-	*	*	-	-	-
MOVW	dir,A	2	3	0	(c)	word (dir) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	addr16,A	3	4	0	(c)	word (addr16) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	SP,A	1	1	0	0	word (SP) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	RWi,A	1	2	1	0	word (RWi) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	ear,A	2	2	1	0	word (ear) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	eam,A	2+	3+(a)	0	(c)	word (eam) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	io,A	2	3	0	(c)	word (io) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	@RWi+disp8,A	2	5	1	(c)	word ((RWi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	@RLi+disp8,A	3	10	2	(c)	word ((RLi)+disp8) <-- (A)	-	-	-	-	-	*	*	-	-	-
MOVW	RWi,ear RWi,eam	2	3	2	0	word (RWi) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOVW	ear,Rwi	2+	4+(a)	1	(c)	word (RWi) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVW	eam,Rwi	2	4	2	0	word (ear) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW	RWi,#imm16	2+	5+(a)	1	(c)	word (eam) <-- (RWi)	-	-	-	-	-	*	*	-	-	-
MOVW	io,#imm16	3	2	1	0	word (RWi) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW	ear,#imm16	4	5	0	(c)	word (io) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOVW	eam,#imm16	4	2	1	0	word (ear) <-- imm16	-	-	-	-	-	*	*	-	-	-
MOVW	@AL,AH/MOVW	4+	4+(a)	0	(c)	word (eam) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOVW	@A,T	2	3	0	(c)	word ((A)) <-- (AH)	-	-	-	-	-	*	*	-	-	-
XCHW	A,ear	2	4	2	0	word (A) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCHW	A,eam	2+	5+(a)	0	2 x (c)	word (A) <-- >(eam)	-	-	-	-	-	-	-	-	-	-
XCHW	RWi, ear	2	7	4	0	word (RWi) <--> (ear)	-	-	-	-	-	-	-	-	-	-
XCHW	RWi, eam	2+	9+(a)	2	2 x (c)	word (RWi) <--> (eam)	-	-	-	-	-	-	-	-	-	-
MOVL	A,ear	2	4	2	0	long (A) <-- (ear)	-	-	-	-	-	*	*	-	-	-
MOVL	A,eam	2+	5+(a)	0	(d) 0	long (A) <-- (eam)	-	-	-	-	-	*	*	-	-	-
MOVL	A,#imm32	5	3	0		long (A) <-- imm32	-	-	-	-	-	*	*	-	-	-
MOVL	ear,A	2	4	2	(d)	long (ear1) <-- (A) long	-	-	-	-	-	*	*	-	-	-
MOVL	eam,A	2+	5+(a)	0		(eam1) <-- (A)	-	-	-	-	-	*	*	-	-	-

See Tables B.5-1 and B.5 -2 for information on (a) to (d) in the table.

Table B.8-3 42 Addition/subtraction Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD	A,#imm8	2	2	0	0	byte (A) <-- (A) + imm8	Z	-	-	-	-	*	*	*	*	-
ADD	A,dir	2	5	0	(b)	byte (A) <-- (A) + (dir)	Z	-	-	-	-	*	*	*	*	-
ADD	A,ear	2	3	1	0	byte (A) <-- (A) + (ear)	Z	-	-	-	-	*	*	*	*	-
ADD	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) + (eam)	Z	-	-	-	-	*	*	*	*	-
ADD	ear,A	2	3	2	0	byte (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADD	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam) + (A)	Z	-	-	-	-	*	*	*	*	*
ADDC	A	1	2	0	0	byte (A) <-- (AH) + (AL) + (C)	Z	-	-	-	-	*	*	*	*	-
ADDC	A,ear	2	3	1	0	byte (A) <-- (A) + (ear)+ (C)	Z	-	-	-	-	*	*	*	*	-
ADDC	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) + (eam)+ (C)	Z	-	-	-	-	*	*	*	*	-
ADDDC	A	1	3	0	0	byte (A) <-- (AH) + (AL) + (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
SUB	A,#imm8	2	2	0	0	byte (A) <-- (A) - imm8	Z	-	-	-	-	*	*	*	*	-
SUB	A,dir	2	5	0	(b)	byte (A) <-- (A) - (dir)	Z	-	-	-	-	*	*	*	*	-
SUB	A,ear	2	3	1	0	byte (A) <-- (A) - (ear)	Z	-	-	-	-	*	*	*	*	-
SUB	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) - (eam)	Z	-	-	-	-	*	*	*	*	-
SUB	ear,A	2	3	2	0	byte (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUB	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBC	A	1	2	0	0	byte (A) <-- (AH) - (AL) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC	A,ear	2	3	1	0	byte (A) <-- (A) - (ear) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBC	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) - (eam) - (C)	Z	-	-	-	-	*	*	*	*	-
SUBDC	A	1	3	0	0	byte (A) <-- (AH) - (AL) - (C) (decimal)	Z	-	-	-	-	*	*	*	*	-
ADDW	A	1	2	0	0	word (A) <-- (AH) + (AL)	-	-	-	-	-	*	*	*	*	-
ADDW	A,ear	2	3	1	0	word (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDW	A,#imm16	3	2	0	0	word (A) <-- (A) + imm16	-	-	-	-	-	*	*	*	*	-
ADDW	ear,A	2	3	2	0	word (ear) <-- (ear) + (A)	-	-	-	-	-	*	*	*	*	-
ADDW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) + (A)	-	-	-	-	-	*	*	*	*	*
ADDCW	A,ear	2	3	1	0	word (A) <-- (A) + (ear) + (C)	-	-	-	-	-	*	*	*	*	-
ADDCW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) + (eam) + (C)	-	-	-	-	-	*	*	*	*	-
SUBW	A	1	2	0	0	word (A) <-- (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
SUBW	A,ear	2	3	1	0	word (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBW	A,#imm16	3	2	0	0	word (A) <-- (A) - imm16	-	-	-	-	-	*	*	*	*	-
SUBW	ear,A	2	3	2	0	word (ear) <-- (ear) - (A)	-	-	-	-	-	*	*	*	*	-
SUBW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) - (A)	-	-	-	-	-	*	*	*	*	*
SUBCW	A,ear	2	3	1	0	word (A) <-- (A) - (ear) - (C)	-	-	-	-	-	*	*	*	*	-
SUBCW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) - (eam) - (C)	-	-	-	-	-	*	*	*	*	-
ADDL	A,ear	2	6	2	0	long (A) <-- (A) + (ear)	-	-	-	-	-	*	*	*	*	-
ADDL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) + (eam)	-	-	-	-	-	*	*	*	*	-
ADDL	A,#imm32	5	4	0	0	long (A) <-- (A) + imm32	-	-	-	-	-	*	*	*	*	-
SUBL	A,ear	2	6	2	0	long (A) <-- (A) - (ear)	-	-	-	-	-	*	*	*	*	-
SUBL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) - (eam)	-	-	-	-	-	*	*	*	*	-
SUBL	A,#imm32	5	4	0	0	long (A) <-- (A) - imm32	-	-	-	-	-	*	*	*	*	-

See Tables B.5-1 and for information on (a) to (d) in the table.

Table B.8-4 12 Increment/decrement Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC	ear	2	3	2	0	byte (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INC	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DEC	ear	2	3	2	0	byte (ear) <-- (ear) - 1 byte	-	-	-	-	-	*	*	*	-	-
DEC	eam	2+	5+(a)	0	2×(b)	(eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*
INCW	ear	2	3	2	0	word (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECW	ear	2	3	2	0	word (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*

Table B.8-4 12 Increment/decrement Instructions (byte, word, long word) (Continued)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INCL	ear	2	7	4	0	long (ear) <-- (ear) + 1	-	-	-	-	-	*	*	*	-	-
INCL	eam	2+	9+(a)	0	2×(d)	long (eam) <-- (eam) + 1	-	-	-	-	-	*	*	*	-	*
DECL	ear	2	7	4	0	long (ear) <-- (ear) - 1	-	-	-	-	-	*	*	*	-	-
DECL	eam	2+	9+(a)	0	2×(d)	long (eam) <-- (eam) - 1	-	-	-	-	-	*	*	*	-	*

See Tables B.5-1 and B.5 -2for information on (a) to (d) in the table.

Table B.8-5 11 Compare Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP	A	1	1 2	0	0	byte (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMP	A,ear	2	3+(a)	1	0	byte (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMP	A,eam	2+	2	0	(b)	byte (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMP	A,#imm8	2		0	0	byte (A) - imm8	-	-	-	-	-	*	*	*	*	-
CMPW	A	1	1 2	0	0	word (AH) - (AL)	-	-	-	-	-	*	*	*	*	-
CMPW	A,ear	2	3+(a)	1	0	word (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPW	A,eam	2+	2	0	(c)	word (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPW	A,#imm16	3		0	0	word (A) - imm16	-	-	-	-	-	*	*	*	*	-
CMPL	A,ear	2	6	2	0	long (A) - (ear)	-	-	-	-	-	*	*	*	*	-
CMPL	A,eam	2+	7+(a)	0	(d)	long (A) - (eam)	-	-	-	-	-	*	*	*	*	-
CMPL	A,#imm32	5	3	0	0	long (A) - imm32	-	-	-	-	-	*	*	*	*	-

See Tables B.5-1and B.5 -2for information on (a) to (d) in the table.

Table B.8-6 11 unsigned multiplication/division instructions (word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU	A	1	*1	0	0	word (AH) / byte (AL) quotient --> byte (AL) remainder --> byte (AH)	-	-	-	-	-	-	-	*	*	-
DIVU	A,ear	2	*2	1	0	word (A) / byte (ear) quotient --> byte (A) remainder --> byte (ear)	-	-	-	-	-	-	-	*	*	-
DIVU	A,eam	2+	*3	0	*6	word (A) / byte (eam) quotient --> byte (A) remainder --> byte (eam)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,ear	2	*4	1	0	long (A) / word (ear) quotient --> word(A) remainder --> word(ear)	-	-	-	-	-	-	-	*	*	-
DIVUW	A,eam	2+	*5	0	*7	long (A) / word (eam) quotient --> word(A) remainder --> word(eam)	-	-	-	-	-	-	-	*	*	-
MULU	A	1	*8	0	0	byte (AH) * byte (AL) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,ear	2	*9	1	0	byte (A) * byte (ear) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULU	A,eam	2+	*10	0	(b)	byte (A) * byte (eam) --> word (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A	1	*11	0	0	word (AH) * word (AL) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,ear	2	*12	1	0	word (A) * word (ear) --> Long (A)	-	-	-	-	-	-	-	-	-	-
MULUW	A,eam	2+	*13	0	(c)	word (A) * word (eam) --> Long (A)										

*1 3: Division by 0 7: Overflow 15: Normal

*2 4: Division by 0 8: Overflow 16: Normal

*3 6+(a): Division by 0 9+(a): Overflow 19+(a): Normal

*4 4: Division by 0 7: Overflow 22: Normal

*5 6+(a): Division by 0 8+(a): Overflow 26+(a): Normal

*6 (b): Division by 0 or overflow 2x(b): Normal

*7 (c): Division by 0 or overflow 2x(c): Normal

*8 3: Byte (AH) is 0. 7: Byte (AH) is not 0.

*9 4: Byte (ear) is 0. 8: Byte (ear) is not 0.

*10 5+(a): Byte (eam) is 0. 9+(a): Byte (eam) is not 0.

*11 3: Word (AH) is 0. 11: Word (AH) is not 0

*12 4: Word (ear) is 0. 12: Word (ear) is not 0.

*13 5+(a): Word (eam) is 0. 13+(a): Word (eam) is not 0.

Note:

See Tables B.5-1 and B.5-2 for information on (a) to (d) in the table.

Table B.8-7 39 Logic 1 Instructions (byte, word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND	A,#imm8	2	2	0	0	byte (A) <-- (A) and imm8	-	-	-	-	-	*	*	R	-	-
AND	A,ear	2	3	1	0	byte (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
AND	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
AND	ear,A	2	3	2	0	byte (ear) <-- (ear)and (A)	-	-	-	-	-	*	*	R	-	-
AND	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam)and (A)	-	-	-	-	-	*	*	R	-	*
OR	A,#imm8	2	2	0	0	byte (A) <-- (A) or imm8	-	-	-	-	-	*	*	R	-	-
OR	A,ear	2	3	1	0	byte (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
OR	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
OR	ear,A	2	3	2	0	byte (ear) <-- (ear)or (A)	-	-	-	-	-	*	*	R	-	-
OR	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam)or (A)	-	-	-	-	-	*	*	R	-	*
XOR	A,#imm8	2	2	0	0	byte (A) <-- (A) xor imm8	-	-	-	-	-	*	*	R	-	-
XOR	A,ear	2	3	1	0	byte (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XOR	A,eam	2+	4+(a)	0	(b)	byte (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XOR	ear,A	2	3	2	0	byte (ear) <-- (ear)xor (A)	-	-	-	-	-	*	*	R	-	-
XOR	eam,A	2+	5+(a)	0	2×(b)	byte (eam) <-- (eam)xor (A)	-	-	-	-	-	*	*	R	-	*
NOT	A	1	2	0	0	byte (A) <-- not (A)	-	-	-	-	-	*	*	R	-	-
NOT	ear	2	3	2	0	byte (ear) <-- not (ear)	-	-	-	-	-	*	*	R	-	-
NOT	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- not (eam)	-	-	-	-	-	*	*	R	-	*
ANDW	A	1	2	0	0	word (A) <-- (AH) and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	A,#imm16	3	2	0	0	word (A) <-- (A) and imm16	-	-	-	-	-	*	*	R	-	-
ANDW	A,ear	2	3	1	0	word (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ANDW	ear,A	2	3	2	0	word (ear) <-- (ear)and (A)	-	-	-	-	-	*	*	R	-	-
ANDW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam)and (A)	-	-	-	-	-	*	*	R	-	*
ORW	A	1	2	0	0	word (A) <-- (AH) or (A)	-	-	-	-	-	*	*	R	-	-
ORW	A,#imm16	3	2	0	0	word (A) <-- (A) or imm16	-	-	-	-	-	*	*	R	-	-
ORW	A,ear	2	3	1	0	word (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
ORW	ear,A	2	3	2	0	word (ear) <-- (ear)or (A)	-	-	-	-	-	*	*	R	-	-
ORW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam)or (A)	-	-	-	-	-	*	*	R	-	*
XORW	A	1	2	0	0	word (A) <-- (AH) xor (A)	-	-	-	-	-	*	*	R	-	-
XORW	A,#imm16	3	2	0	0	word (A) <-- (A) xor imm16	-	-	-	-	-	*	*	R	-	-
XORW	A,ear	2	3	1	0	word (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORW	A,eam	2+	4+(a)	0	(c)	word (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-
XORW	ear,A	2	3	2	0	word (ear) <-- (ear)xor (A)	-	-	-	-	-	*	*	R	-	*
XORW	eam,A	2+	5+(a)	0	2×(c)	word (eam) <-- (eam)xor (A)	-	-	-	-	-	*	*	R	-	-
NOTW	A	1	2	0	0	word (A) <-- not (A)	-	-	-	-	-	*	*	R	-	-
NOTW	ear	2	3	2	0	word (ear) <-- not (ear)	-	-	-	-	-	*	*	R	-	*
NOTW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- not (eam)										

See Tables B.5-1 and B.5-2 for information on (a) to (d) in the table.

Table B.8-8 Six Logic 2 Instructions (long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL	A,ear	2	6	2	0	long (A) <-- (A) and (ear)	-	-	-	-	-	*	*	R	-	-
ANDL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) and (eam)	-	-	-	-	-	*	*	R	-	-
ORL	A,ear	2	6	2	0	long (A) <-- (A) or (ear)	-	-	-	-	-	*	*	R	-	-
ORL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) or (eam)	-	-	-	-	-	*	*	R	-	-
XORL	A,ear	2	6	2	0	long (A) <-- (A) xor (ear)	-	-	-	-	-	*	*	R	-	-
XORL	A,eam	2+	7+(a)	0	(d)	long (A) <-- (A) xor (eam)	-	-	-	-	-	*	*	R	-	-

See Tables B.5-1 and B.5-2 for information on (a) to (d) in the table.

Table B.8-9 Six Sign Inversion Instructions (byte, word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG	A	1	2	0	0	byte (A) <-- 0 - (A)	X	-	-	-	-	*	*	*	*	-
NEG	ear	2	3	2	0	byte (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEG	eam	2+	5+(a)	0	0×(b)	byte (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*
NEGW	A	1	2	0	0	word (A) <-- 0 - (A)	-	-	-	-	-	*	*	*	*	-
NEGW	ear	2	3	2	0	word (ear) <-- 0 - (ear)	-	-	-	-	-	*	*	*	*	-
NEGW	eam	2+	5+(a)	0	2×(c)	word (eam) <-- 0 - (eam)	-	-	-	-	-	*	*	*	*	*

See Tables B.5-1 and B.5-2 for information on (a) to (d) in the table.

Table B.8-10 One Normalization Instruction (long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML	A,R0	2	*1	1	0	long (A) <-- Shifts to the position where '1' is set for the first time. byte (RD) <-- Shift count at that time	-	-	-	-	-	-	*	-	-	-

See Tables B.5-1 and B.5-2 for information on (a) to (d) in the table.

Table B.8-11 18 Shift Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC	A	2	2	0	0	byte (A) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	A	2	2	0	0	byte (A) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	ear	2	3	2	0	byte (ear) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	-
RORC	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- With right rotation carry	-	-	-	-	-	*	*	-	*	*
ROLC	ear	2	3	2	0	byte (ear) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	-
ROLC	eam	2+	5+(a)	0	2×(b)	byte (eam) <-- With left rotation carry	-	-	-	-	-	*	*	-	*	*
ASR	A,R0	2	*1	1	0	byte (A) <-- Arithmetic right shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
LSR	A,R0	2	*1	1	0	byte (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
LSL	A,R0	2	*1	1	0	byte (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

Table B.8-11 18 Shift Instructions (byte, word, long word) (Continued)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ASRW	A	1	2	0	0	word (A) <-- Arithmetic right shift (A, 1 bit)	-	-	-	-	*	*	*	-	*	-
LSRW	A/SHRW A	1	2	0	0	word (A) <-- Logical right shift (A, 1 bit)	-	-	-	-	*	R	*	-	*	-
LSLW	A/SHLW A	1	2	0	0	word (A) <-- Logical left shift (A, 1 bit)	-	-	-	-	-	*	*	-	*	-
ASRW	A,R0	2	*1	1	0	word (A) <-- Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRW	A,R0	2	*1	1	0	word (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLW	A,R0	2	*1	1	0	word (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-
ASRL	A,R0	2	*2	1	0	long (A) <-- Arithmetic right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSRL	A,R0	2	*2	1	0	long (A) <-- Logical right barrel shift (A, R0)	-	-	-	-	*	*	*	-	*	-
LSLL	A,R0	2	*2	1	0	long (A) <-- Logical left barrel shift (A, R0)	-	-	-	-	-	*	*	-	*	-

*1 6 when R0 is 0; otherwise, 5 + (R0)

*2 6 when R0 is 0; otherwise, 5 + (R0)

See Tables B.5-1 and B.5-2 for information on (a) to (d) in the table.

Table B.8-12 31 Branch 1 Instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ/BEQ	rel	2	*1	0	0	Branch on (Z) = 1	-	-	-	-	-	-	-	-	-	-
BNZ/BNE	rel	2	*1	0	0	Branch on (Z) = 0	-	-	-	-	-	-	-	-	-	-
BC/BLOB	rel	2	*1	0	0	Branch on (C) = 1	-	-	-	-	-	-	-	-	-	-
NC/BHS	rel	2	*1	0	0	Branch on (C) = 0	-	-	-	-	-	-	-	-	-	-
BN	rel	2	*1	0	0	Branch on (N) = 1	-	-	-	-	-	-	-	-	-	-
BP	rel	2	*1	0	0	Branch on (N) = 0	-	-	-	-	-	-	-	-	-	-
BV	rel	2	*1	0	0	Branch on (V) = 1	-	-	-	-	-	-	-	-	-	-
BNV	rel	2	*1	0	0	Branch on (V) = 0	-	-	-	-	-	-	-	-	-	-
BT	rel	2	*1	0	0	Branch on (T) = 1	-	-	-	-	-	-	-	-	-	-
BNT	rel	2	*1	0	0	Branch on (T) = 0	-	-	-	-	-	-	-	-	-	-
BLT	rel	2	*1	0	0	Branch on (V) xor (N) = 1	-	-	-	-	-	-	-	-	-	-
BGE	rel	2	*1	0	0	Branch on (V) xor (N) = 0	-	-	-	-	-	-	-	-	-	-
BLE	rel	2	*1	0	0	((V) xor (N)) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BGT	rel	2	*1	0	0	((V) xor (N)) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BLS	rel	2	*1	0	0	Branch on (C) or (Z) = 1	-	-	-	-	-	-	-	-	-	-
BHI	rel	2	*1	0	0	Branch on (C) or (Z) = 0	-	-	-	-	-	-	-	-	-	-
BRA	rel	2	*1	0	0	Unconditional branch	-	-	-	-	-	-	-	-	-	-
JMP	@A	1	2	0	0	word (PC) <-- (A)	-	-	-	-	-	-	-	-	-	-
JMP	addr16	3	3	0	0	word (PC) <-- addr16	-	-	-	-	-	-	-	-	-	-
JMP	@ear	2	3	1	0	word (PC) <-- (ear)	-	-	-	-	-	-	-	-	-	-
JMP	@eam	2+	4+(a)	0	(c)	word (PC) <-- (eam)	-	-	-	-	-	-	-	-	-	-
JMPP	@ear *3	2	5	2	0	word (PC) <-- (ear), (PCB) <-- (ear+2)	-	-	-	-	-	-	-	-	-	-
JMPP	@eam *3	2+	6+(a)	0	(d)	word (PC) <-- (eam), (PCB) <-- (eam+2)	-	-	-	-	-	-	-	-	-	-
JMPP	addr24	4	4	0	0	word(PC) <-- ad24 0-15,(PCB) <-- ad24 16-23	-	-	-	-	-	-	-	-	-	-
CALL	@ear *4	2	6	1	(c)	word (PC) <-- (ear)	-	-	-	-	-	-	-	-	-	-
CALL	@eam *4	2+	7+(a)	0	2×(c)	word (PC) <-- (eam)	-	-	-	-	-	-	-	-	-	-
CALL	addr16 *5	3	6	0	(c)	word (PC) <-- addr16	-	-	-	-	-	-	-	-	-	-
CALLV	#vct4 *5	1	7	0	2×(c)	Vector call instruction	-	-	-	-	-	-	-	-	-	-
CALLP	@ear *6	2	10	2	2×(c)	word(PC) <-- (ear)0-15,(PCB) <-- (ear)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	@eam *6	2+	11+(a)	0	*2	word(PC) <-- (eam)0-15,(PCB) <-- (eam)16-23	-	-	-	-	-	-	-	-	-	-
CALLP	addr24 *7	4	10	0	2×(c)	word(PC) <-- addr0-15, (PCB) <-- addr16-23	-	-	-	-	-	-	-	-	-	-

*1 4 when a branch is made; otherwise, 3

*2 3 x (c) + (b)

*3 Read (word) of branch destination address

*4 W: Save to stack (word) R: Read (word) of branch destination address

*5 Save to stack (word)

*6 W: Save to stack (long word) R: Read (long word) of branch destination address

*7 Save to stack (long word)

Note:

See Tables B.5-1and B.5-2 for information on (a) to (d) in the table.

Table B.8-13 19 Branch 2 Instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE	A,#imm8,rel	3	*1	0	0	Branch on byte (A) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	A,#imm16,rel	4	*1	0	0	Branch on word (A) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CBNE	ear,#imm8,rel	4	*2	1	0	Branch on byte (ear) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CBNE	eam,#imm8,rel *9	4+	*3	0	(b)	Branch on byte (eam) not equal to imm8	-	-	-	-	-	*	*	*	*	-
CWBNE	ear,#imm16,rel	5	*4	1	0	Branch on word (ear) not equal to imm16	-	-	-	-	-	*	*	*	*	-
CWBNE	eam,#imm16,rel*9	5+	*3	0	(c)	Branch on word (eam) not equal to imm16	-	-	-	-	-	*	*	*	*	-
DBNZ	ear,rel	3	*5	2	0	Branch on byte (ear) = (ear) - 1, (ear)not equal to 0	-	-	-	-	-	*	*	*	-	-
DBNZ	eam,rel	3+	*6	2	2×(b)	Branch on byte (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
DWBNZ	ear,rel	3	*5	2	0	Branch on word (ear) = (ear) - 1, (ear) not equal to 0	-	-	-	-	-	*	*	*	-	-
DWBNZ	eam,rel	3+	*6	2	2×(c)	Branch on word (eam) = (eam) - 1, (eam) not equal to 0	-	-	-	-	-	*	*	*	-	*
INT	#vct8	2	20	0	8×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT	addr16	3	16	0	6×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INTP	addr24	4	17	0	6×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
INT9		1	20	0	8×(c)	Software interrupt	-	-	R	S	-	-	-	-	-	-
RETI		1	15	0	6×(c)	Return from interrupt	-	-	*	*	*	*	*	*	*	-
ILINK		2	6	0	(c)	Saves the old frame pointer in the stack upon entering the function, then sets the new frame pointer and reserves the local pointer area.	-	-	-	-	-	-	-	-	-	-
IUNLINK		1	5	0	(c)	Recovers the old frame pointer from the stack upon exiting the function.	-	-	-	-	-	-	-	-	-	-
RET	*7	1	4	0	(c)	Return from subroutine	-	-	-	-	-	-	-	-	-	-
RETP	*8	1	6	0	(d)	Return from subroutine	-	-	-	-	-	-	-	-	-	-

*1 5 when a branch is made; otherwise, 4

*2 13 when a branch is made; otherwise, 12

*3 7+(a) when a branch is made; otherwise, 6+(a)

*4 8 when a branch is made; otherwise, 7

*5 7 when a branch is made; otherwise, 6

*6 8+(a) when a branch is made; otherwise, 7+(a)

*7 Return from stack (word)

*8 Return from stack (long word)

*9 Do not use RWj+ addressing mode with the CBNE or CWBNE instruction.

Note:

See Tables B.4 and B.5 for information on (a) to (d) in the table.

See Tables B.5-1and B.5-2 for information on (a) to (d) in the table.

Table B.8-14 31 28 Other Control Instructions (byte, word, long word)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW	A	1	4	0	(c)	word (SP) <-- (SP) - 2 , ((SP)) <-- (A)	-	-	-	-	-	-	-	-	-	-
PUSHW	AH	1	4	0	(c)	word (SP) <-- (SP) - 2 , ((SP)) <-- (AH)	-	-	-	-	-	-	-	-	-	-
PUSHW	PS	1	4	0	(c)	word (SP) <-- (SP) - 2 , ((SP)) <-- (PS)	-	-	-	-	-	-	-	-	-	-
PUSHW	rlst	2	*3		*4	(SP) <-- (SP) - 2n , ((SP)) <-- (rlst)	-	-	-	-	-	-	-	-	-	-
POPW	A	1	3	0	(c)	word (A) <-- ((SP)) , (SP) <-- (SP) + 2	-	*	-	-	-	-	-	-	-	-
POPW	AH	1	3	0	(c)	word (AH) <-- ((SP)) , (SP) <-- (SP) + 2	-	-	-	-	-	-	-	-	-	-
POPW	PS	1	4	0	(c)	word (PS) <-- ((SP)) , (SP) <-- (SP) + 2	-	-	*	*	*	*	*	*	*	-
POPW	rlst	2	*2		*4	(rlst) <-- ((SP)) , (SP) <-- (SP)	-	-	-	-	-	-	-	-	-	-
JCTX	@A	1	14	0	6×(c)	Context switch instruction	-	-	*	*	*	*	*	*	*	-
AND	CCR,#imm8	2	3	0	0	byte (CCR) <-- (CCR) and imm8	-	-	*	*	*	*	*	*	*	-
OR	CCR,#imm8	2	3	0	0	byte(CCR) <-- (CCR) or imm8	-	-	*	*	*	*	*	*	*	-
MOV	RP,#imm8	2	2	0	0	byte (RP) <-- imm8	-	-	-	-	-	-	-	-	-	-
MOV	ILM,#imm8	2	2	0	0	byte (ILM)Â© imm8	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,ear	2	3	1	0	word (RWi) <-- ear	-	-	-	-	-	-	-	-	-	-
MOVEA	RWi,eam	2+	2+(a)	1	0	word (RWi) <-- eam	-	-	-	-	-	-	-	-	-	-
MOVEA	A,ear	2	1	0	0	word (A) <-- ear	-	*	-	-	-	-	-	-	-	-
MOVEA	A,eam	2+	1+(a)	0	0	word (A) <-- eam	-	*	-	-	-	-	-	-	-	-
ADDSP	#imm8	2	3	0	0	word (SP) <-- ext(imm8)	-	-	-	-	-	-	-	-	-	-
ADDSP	#imm16	3	3	0	0	word (SP) <-- imm16	-	-	-	-	-	-	-	-	-	-
MOV	A,brg1	2	*1	0	0	byte (A) <-- (brg1)	Z	*	-	-	-	*	*	-	-	-
MOV	brg2,A	2	1	0	0	byte (brg2) <-- (A)	-	-	-	-	-	*	*	-	-	-
NOP		1	1	0	0	No operation	-	-	-	-	-	-	-	-	-	-
ADB		1	1	0	0	Prefix code for AD space access	-	-	-	-	-	-	-	-	-	-
DTB		1	1	0	0	Prefix code for DT space access	-	-	-	-	-	-	-	-	-	-
PCB		1	1	0	0	Prefix code for PC space access	-	-	-	-	-	-	-	-	-	-
SPB		1	1	0	0	Prefix code for SP space access	-	-	-	-	-	-	-	-	-	-
NCC		1	1	0	0	Prefix code for flag no-change	-	-	-	-	-	-	-	-	-	-
CMR		1	1	0	0	Prefix code for common register bank	-	-	-	-	-	-	-	-	-	-

*1 PCB, ADB, SSB, USB: 1, DTB, DPR: 2

*2 7 + 3×(POP count) + 2×(POP last register number), 7 when RLST = 0 (no transfer register)

*3 29 + 3×(PUSH count) - 3×(PUSH last register number), 8 when RLST = 0 (no transfer register)

*4 (POP count)×(c) or (PUSH count)×(c)

*5 (POP count) or (PUSH count)

Note:

See Tables B.5-1 and B.5-2 for information on (a) to (d) in the table.

Table B.8-15 21 Bit Operand Instructions

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB	A,dir:bp	3	5	0	(b)	byte (A) <-- (dir:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,addr16:bp	4	5	0	(b)	byte (A) <-- (addr16:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	A,io:bp	3	4	0	(b)	byte (A) <-- (io:bp)b	Z	*	-	-	-	*	*	-	-	-
MOVB	dir:bp,A	3	7	0	2×(b)	bit (dir:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB	addr16:bp,A	4	7	0	2×(b)	bit (addr16:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*
MOVB	io:bp,A	3	6	0	2×(b)	bit (io:bp)b <-- (A)	-	-	-	-	-	*	*	-	-	*

Table B.8-15 21 Bit Operand Instructions (Continued)

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SETB	dir:bp	3	7	0	2×(b)	bit (dir:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB	addr16:bp	4	7	0	2×(b)	bit (addr16:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
SETB	io:bp	3	7	0	2×(b)	bit (io:bp)b <-- 1	-	-	-	-	-	-	-	-	-	*
CLRB	dir:bp	3	7	0	2×(b)	bit (dir:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB	addr16:bp	4	7	0	2×(b)	bit (addr16:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
CLRB	io:bp	3	7	0	2×(b)	bit (io:bp)b <-- 0	-	-	-	-	-	-	-	-	-	*
BBC	dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC	addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBC	io:bp,rel	4	*2	0	(b)	Branch on (io:bp) b = 0	-	-	-	-	-	-	*	-	-	-
BBS	dir:bp,rel	4	*1	0	(b)	Branch on (dir:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS	addr16:bp,rel	5	*1	0	(b)	Branch on (addr16:bp) b = 1	-	-	-	-	-	-	*	-	-	-
BBS	io:bp,rel	4	*1	0	(b)	Branch on (io:bp) b = 1	-	-	-	-	-	-	*	-	-	-
SBBS	addr16:bp,rel	5	*3	0	2×(b)	Branch on (addr16:bp) b = 1, bit = 1	-	-	-	-	-	-	*	-	-	*
WBTS	io:bp	3	*4	0	*5	Waits until (io:bp) b = 1	-	-	-	-	-	-	-	-	-	-
WBTC	io:bp	3	*4	0	*5	Waits until (io:bp) b = 0	-	-	-	-	-	-	-	-	-	-

*1 8 when a branch is made; otherwise, 7

*2 7 when a branch is made; otherwise, 6

*3 10 when the condition is met; otherwise 9

*4 Undefined count

*5 Until the condition is met(dir:bp)b

Table B.8-16 Six Accumulator Operation Instructions (byte, word)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A)0-7 <--> (A)8-15	-	-	-	-	-	-	-	-	-	-
SWAPW/XCHW A,T	1	2	0	0	word (AH) <--> (AL)	-	*	-	-	-	-	-	-	-	-
EXT	1	1	0	0	Byte sign extension	X	-	-	-	-	*	*	-	-	-
EXTW	1	2	0	0	Word sign extension	-	X	-	-	-	*	*	-	-	-
ZEXT	1	1	0	0	Byte zero extension	Z	-	-	-	-	R	*	-	-	-
ZEXTW	1	1	0	0	Word zero extensionbyte	-	-	Z	-	-	R	*	-	-	-

See Tables B.5-1and B.5-2 for information on (a) to (d) in the table.

Table B.8-17 Ten String Instructions

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS / MOVSI	2	*2		*3	byte transfer @AH+ <-- @AL+, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSD	2	*2		*3	byte transfer @AH- <-- @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCEQ / SCEQI	2	*1		*4	byte search @AH+ <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
SCEQD	2	*1		*4	byte search @AH- <-- AL, counter RW0	-	-	-	-	-	*	*	*	*	-
FILS / FILSI	2	6m+6		*3	byte fill @AH+ <-- AL, counter RW0	-	-	-	-	-	*	*	-	-	-

Table B.8-17 Ten String Instructions (Continued)

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVSW / MOVSW	2	*2		*6	word transfer @AH+ <-- @AL+, counter = RW0 word transfer @AH- <-- @AL-, counter = RW0	-	-	-	-	-	-	-	-	-	-
MOVSWD	2	*2		*6	word search @AH+ Å| AL, counter = RW0 word search @AH- Å| AL, counter = RW0	-	-	-	-	-	-	-	-	-	-
SCWEQ / SCWEQI	2	*1		*7	word fill @AH+ <-- AL, counter = RW0	-	-	-	-	-	*	*	*	*	-
SCWEQD	2	*1		*7		-	-	-	-	-	*	*	*	*	-
FILSW / FILSWI	2	6m+6		*6		-	-	-	-	-	*	*	-	-	-

*1 5 when RW0 is 0, 4 + 7 × (RW0) when the counter expires, or 7n + 5 when a match occurs

*2 5 when RW0 is 0; otherwise, 4 + 8 × (RW0)

*3 (b) × (RW0) + (b) × (RW0) When the source and destination access different areas, calculate the (b) item individually.

*4 (b) × n

*5 2 × (R × W0)

*6 (c) × (RW0) + (c) × (RW0) When the source and destination access different areas, calculate the (c) item individually.

*7 (c) × n

*8 2 × 0(RW0)

m: RW0 value (counter value), n: Loop count

Note:

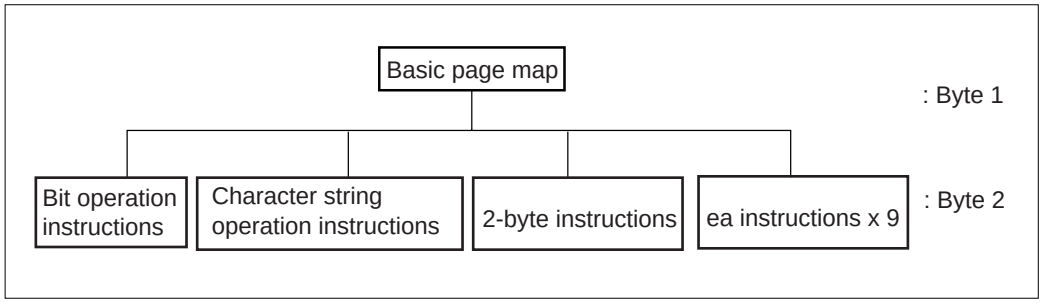
See Tables B.5-1and B.5-2 for information on (a) to (d) in the table.

B.9 Instruction Map

Each F^{2}MC-16LX instruction code consists of 1 or 2 bytes. Therefore, the instruction map consists of multiple pages. Tables B.9-2 to B.9-21 summarize the F^{2}MC-16LX instruction map.

■ Structure of Instruction Map

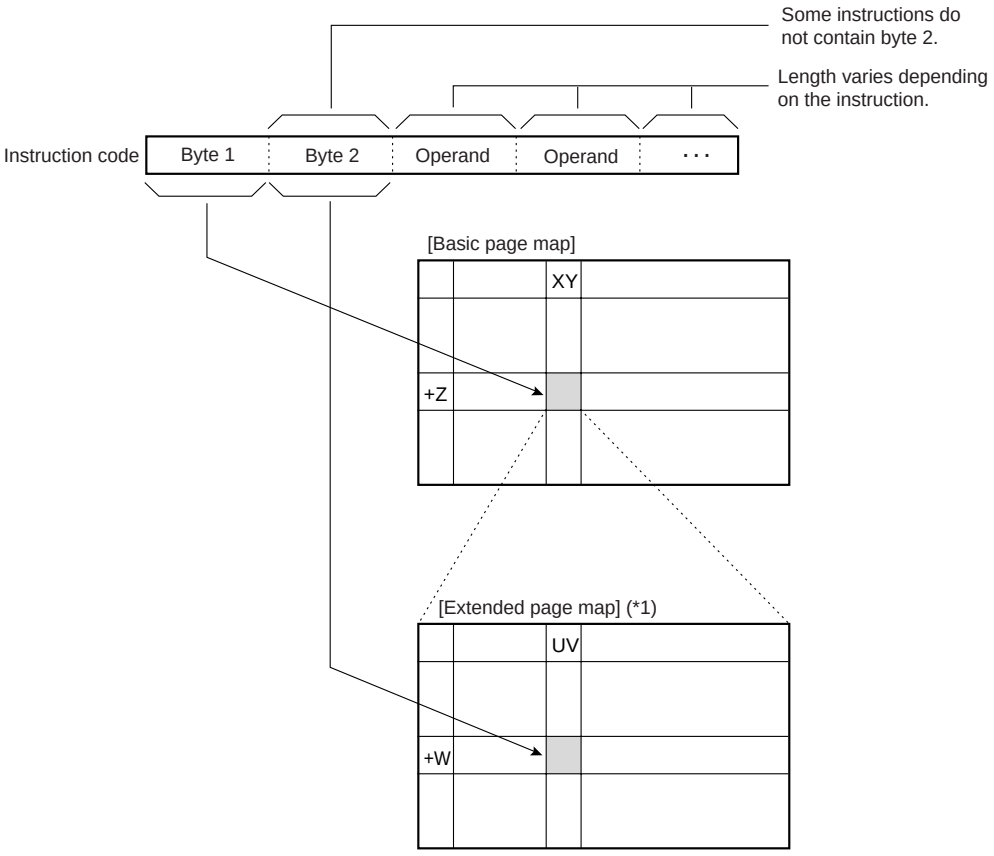
Figure B.9-1 Structure of Instruction Map



An instruction such as the NOP instruction that ends in one byte is completed within the basic page. An instruction such as the MOVS instruction that requires two bytes recognizes the existence of byte 2 when it references byte 1, and can check the following one byte by referencing the map for byte 2. Figure B.9-2 shows the correspondence between an actual

instruction code and instruction map.

Figure B.9-2 Correspondence between Actual Instruction Code and Instruction Map



*1 The extended page map is a generic name of maps for bit operation instructions, character string operation instructions, 2-byte instructions, and ea instructions. Actually, there are multiple extended page maps for each type of instructions.

An example of an instruction code is shown in Table B.9-1.

Table B.9-1 Example of an Instruction Code

Instruction	Byte 1 (from basic page map)	Byte 2 (from extended page map)
NOP	00 +0=00	-
AND A, #8	30 +4=34	-
MOV A, ADB	60 +F=6F	00 +0=00
@RW2+d8, rel	70 +0=70	F0 +2=F2

Table B.9-2 Basic Page Map

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	NOP	CMR	ADD A,dir	ADD A,#8	MOV A,dir	MOV A,io	BRA rel	ea instruction 1	MOV A,Ri	MOV Ri,A	MOV Ri,#8	MOVX A,Ri	MOVX A, @RWi+d8	MOVN A,#4	CALLV #4	BZ /BEQ rel
+ 1	INT9	NCC	SUB A,dir	SUB A,#8	MOV dir,A	MOV io,A	JMP @A	ea instruction 2								BNZ/BNE rel
+ 2	ADDDC A	SUBDC A	ADDC A	SUBC A	MOV A,#8	MOV A,addr16	JMP addr16	ea instruction 3								BC /BLO rel
+ 3	NEG A	JCTX @A	CMP A	CMP A,#8	MOVX A,#8	MOV addr16,A	JMPP addr24	ea instruction 4								BNC /BHS rel
+ 4	PCB	EXT	AND CCR,#8	AND A,#8	MOV dir,#8	MOV io,#8	CALL addr16	ea instruction 5								BN rel
+ 5	DTB	ZEXT	OR CCR,#8	OR A,#8	MOVX A,dir	MOVX A,io	CALLP addr24	ea instruction 6								BP rel
+ 6	ADB	SWAP	DIVU A	XOR A,#8	MOVW A,SP	MOVW io,#16	RETP	ea instruction 7								BV rel
+ 7	SPB	ADDSP #8	MULU A	NOT A	MOVW SP,A	MOVX A,addr16	RET	ea instruction 8								BNV rel
+ 8	LINK #imm8	ADDL A,#32	ADDW A	ADDW A,#16	MOVW A,dir	MOVW A,io	INT #vct8	ea instruction 9	MOVW A,RWi	MOVW RWi,A	MOVW RWi,#16	MOVW A, @RWi+d8	MOVW @RWi+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								BLT rel
+ B	NEGW	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 instruction	MOV ea,Ri								BLE rel
+ D		ZEXTW	ORW A	ORW A,#16	PUSHW AH	POPW AH		MOVW ea,RWi								BGT rel
+ E	ASRW A	SWAPW	XORW A	XORW A,#16	PUSHW PS	POPW PS	Character string operation instruction	XCH Ri,ea								BLS rel
+ F	LSRW A	ADDSP #16	MULUW A	NOTW A	PUSHW rlst	POPW rlst	2-byte instruction	XCHW RWi,ea								BHI rel

Table B.9-3 Bit Operation Instruction Map (first byte = 6CH)

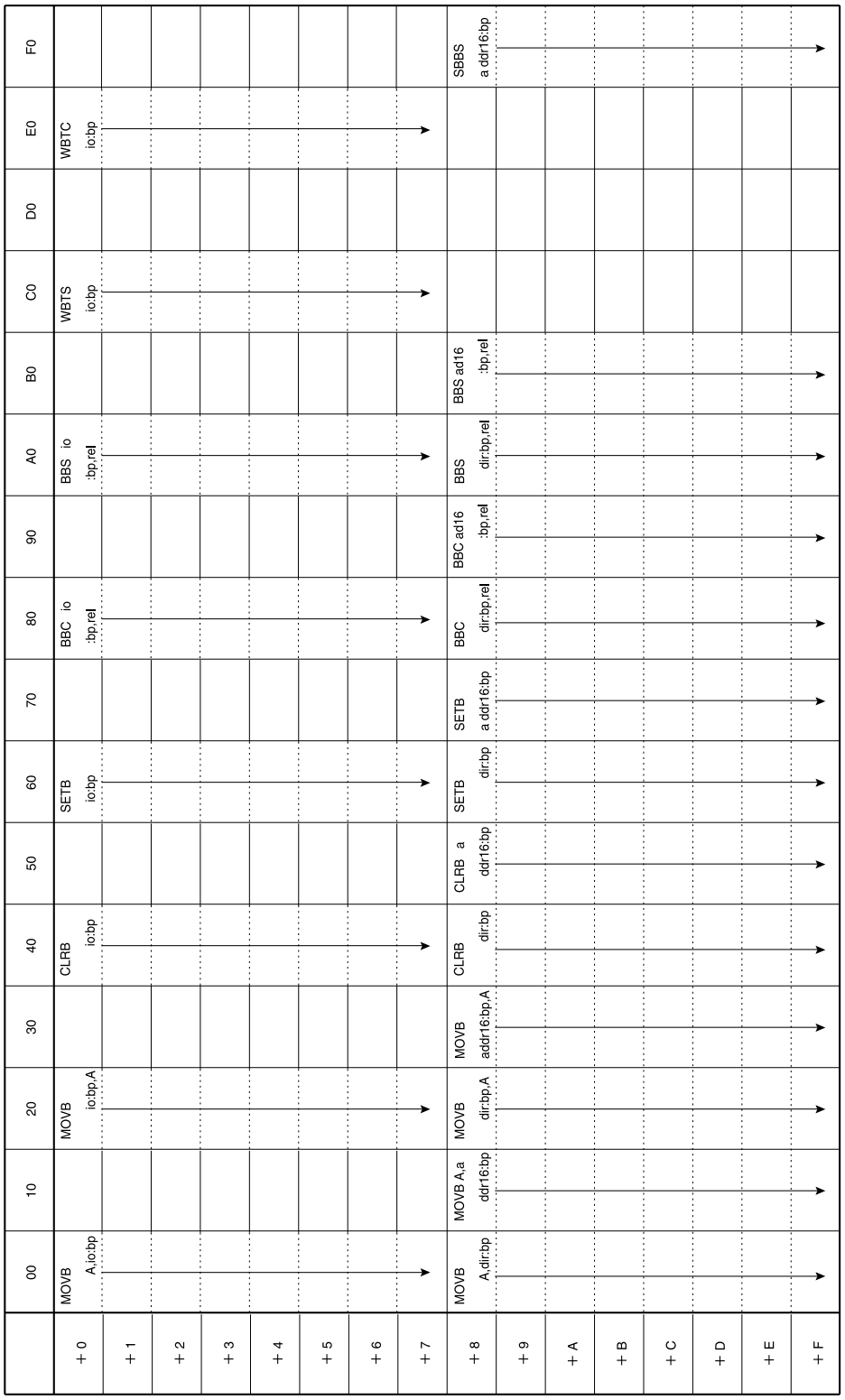


Table B.9-4 Character String Operation Instruction Map (first byte = $6E_H$)

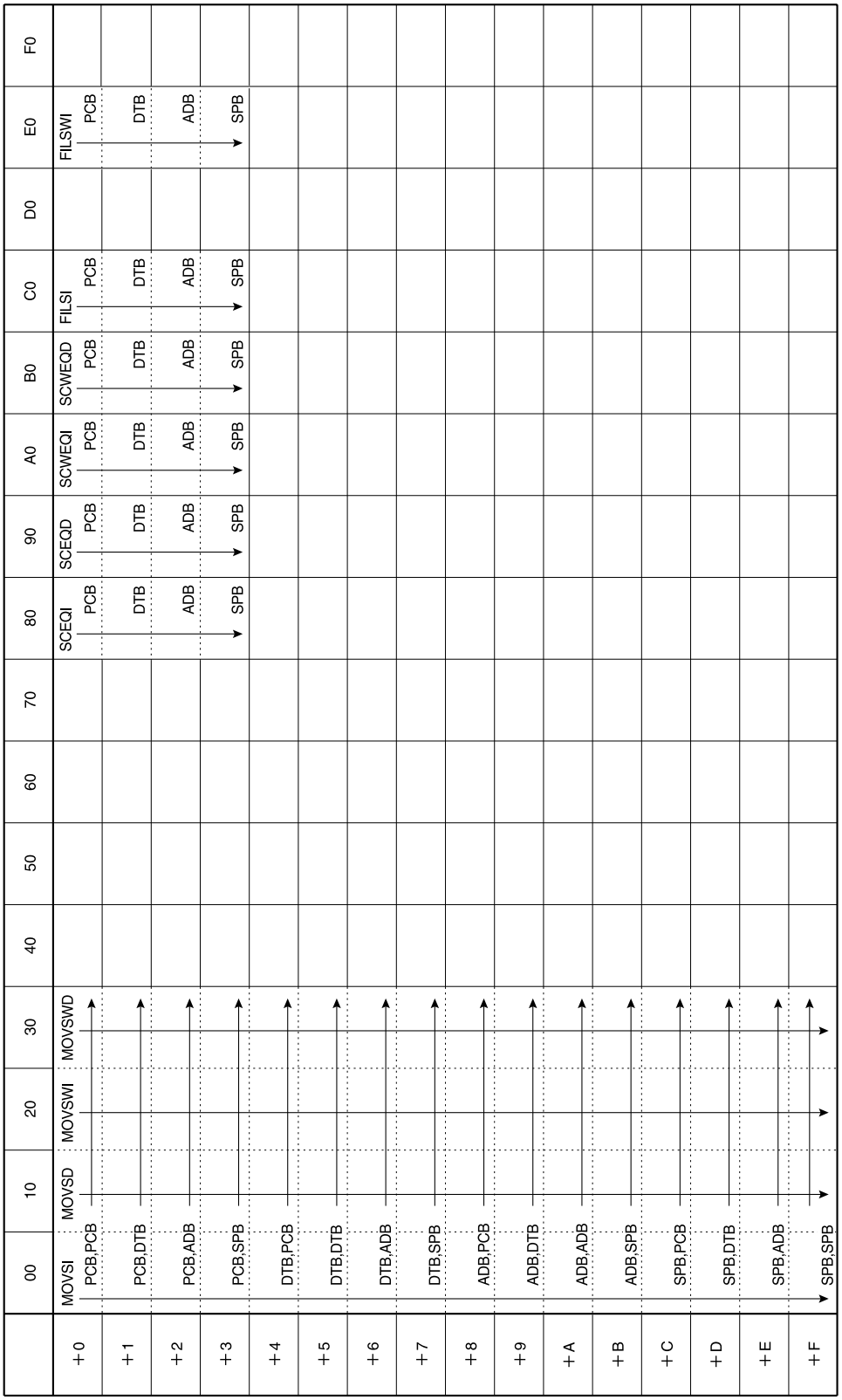


Table B.9-5 2-byte Instruction Map (first byte = 6FH)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+0	MOV A,DTB	MOV DTB,A	MOVX A, @RL0+d8	MOV @RL0 +d8,A	MOV A, @RL0+d8											
+1	MOV A,ADB	MOV ADB,A														
+2	MOV A,SSB	MOV SSB,A	MOVX A, @RL1+d8	MOV @RL1 +d8,A	MOV A, @RL1+d8											
+3	MOV A,USB	MOV USB,A														
+4	MOV A,DPR	MOV DPR,A	MOVX A, @RL2+d8	MOV @RL2 +d8,A	MOV A, @RL2+d8											
+5	MOV A,@A	MOV @AL,AH														
+6	MOV A,PCB	MOVX A,@A	MOVX A, @RL3+d8	MOV @RL3 +d8,A	MOV A, @RL3+d8											
+7	ROLC A	RORC A														
+8				MOVW @RL 0+d8,A	MOVW A, @RL0+d8											
+9																
+A				MOVW @RL 1+d8,A	MOVW A, @RL1+d8											
+B																
+C	LSLW A,R0	LSLL A,R0	LSL A,R0	MOVW @RL 2+d8,A	MOVW A, @RL2+d8											
+D	MOVW A,@A	MOVW @AL,AH	NRML A,R0													
+E	ASRW A,R0	ASRL A,R0	ASR A,R0	MOVW @RL 3+d8,A	MOVW A, @RL3+d8											
+F	LSRW A,R0	LSRL A,R0	LSR A,R0													

Table B.9-6 ea Instruction 1 (first byte = 70_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
					CWBNE ,↓	CWBNE ,↓									CBNE ,↓	CBNE ,↓
+ 0	ADDL A, A,RL0	ADDL A, @RW0+d8	SUBL A,RL0	SUBL A, @RW0+d8	RW0, #16,rel	@RW0+d8, #16,rel	CMPL A,RL0	CMPL A, @RW0+d8	ANDL A,RL0	ANDL A, @RW0+d8	ORL A,RL0	ORL A, @RW0+d8	XORL A,RL0	XORL A, @RW0+d8	R0, #8,rel	@RW0+d8, #8,rel
+ 1	ADDL A, A,RL0	ADDL A, @RW1+d8	SUBL A,RL0	SUBL A, @RW1+d8	RW1, #16,rel	@RW1+d8, #16,rel	CMPL A,RL0	CMPL A, @RW1+d8	ANDL A,RL0	ANDL A, @RW1+d8	ORL A,RL0	ORL A, @RW1+d8	XORL A,RL0	XORL A, @RW1+d8	R1, #8,rel	@RW1+d8, #8,rel
+ 2	ADDL A, A,RL1	ADDL A, @RW2+d8	SUBL A,RL1	SUBL A, @RW2+d8	RW2, #16,rel	@RW2+d8, #16,rel	CMPL A,RL1	CMPL A, @RW2+d8	ANDL A,RL1	ANDL A, @RW2+d8	ORL A,RL1	ORL A, @RW2+d8	XORL A,RL1	XORL A, @RW2+d8	R2, #8,rel	@RW2+d8, #8,rel
+ 3	ADDL A, A,RL1	ADDL A, @RW3+d8	SUBL A,RL1	SUBL A, @RW3+d8	RW3, #16,rel	@RW3+d8, #16,rel	CMPL A,RL1	CMPL A, @RW3+d8	ANDL A,RL1	ANDL A, @RW3+d8	ORL A,RL1	ORL A, @RW3+d8	XORL A,RL1	XORL A, @RW3+d8	R3, #8,rel	@RW3+d8, #8,rel
+ 4	ADDL A, A,RL2	ADDL A, @RW4+d8	SUBL A,RL2	SUBL A, @RW4+d8	RW4, #16,rel	@RW4+d8, #16,rel	CMPL A,RL2	CMPL A, @RW4+d8	ANDL A,RL2	ANDL A, @RW4+d8	ORL A,RL2	ORL A, @RW4+d8	XORL A,RL2	XORL A, @RW4+d8	R4, #8,rel	@RW4+d8, #8,rel
+ 5	ADDL A, A,RL2	ADDL A, @RW5+d8	SUBL A,RL2	SUBL A, @RW5+d8	RW5, #16,rel	@RW5+d8, #16,rel	CMPL A,RL2	CMPL A, @RW5+d8	ANDL A,RL2	ANDL A, @RW5+d8	ORL A,RL2	ORL A, @RW5+d8	XORL A,RL2	XORL A, @RW5+d8	R5, #8,rel	@RW5+d8, #8,rel
+ 6	ADDL A, A,RL3	ADDL A, @RW6+d8	SUBL A,RL3	SUBL A, @RW6+d8	RW6, #16,rel	@RW6+d8, #16,rel	CMPL A,RL3	CMPL A, @RW6+d8	ANDL A,RL3	ANDL A, @RW6+d8	ORL A,RL3	ORL A, @RW6+d8	XORL A,RL3	XORL A, @RW6+d8	R6, #8,rel	@RW6+d8, #8,rel
+ 7	ADDL A, A,RL3	ADDL A, @RW7+d8	SUBL A,RL3	SUBL A, @RW7+d8	RW7, #16,rel	@RW7+d8, #16,rel	CMPL A,RL3	CMPL A, @RW7+d8	ANDL A,RL3	ANDL A, @RW7+d8	ORL A,RL3	ORL A, @RW7+d8	XORL A,RL3	XORL A, @RW7+d8	R7, #8,rel	@RW7+d8, #8,rel
+ 8	ADDL A, A,@RW0	ADDL A, @RW0+d16	SUBL A,@RW0	SUBL A, @RW0+d16	@RW0, #16,rel	@RW0+d16 , #16,rel	CMPL A,@RW0	CMPL A, @RW0+d16	ANDL A,@RW0	ANDL A, @RW0+d16	ORL A,@RW0	ORL A, @RW0+d16	XORL A,@RW0	XORL A, @RW0+d16	@RW0, #8,rel	@RW0+d16 #8,rel
+ 9	ADDL A, A,@RW1	ADDL A, @RW1+d16	SUBL A,@RW1	SUBL A, @RW1+d16	@RW1, #16,rel	@RW1+d16 , #16,rel	CMPL A,@RW1	CMPL A, @RW1+d16	ANDL A,@RW1	ANDL A, @RW1+d16	ORL A,@RW1	ORL A, @RW1+d16	XORL A,@RW1	XORL A, @RW1+d16	@RW1, #8,rel	@RW1+d16 #8,rel
+ A	ADDL A, A,@RW2	ADDL A, @RW2+d16	SUBL A,@RW2	SUBL A, @RW2+d16	@RW2, #16,rel	@RW2+d16 , #16,rel	CMPL A,@RW2	CMPL A, @RW2+d16	ANDL A,@RW2	ANDL A, @RW2+d16	ORL A,@RW2	ORL A, @RW2+d16	XORL A,@RW2	XORL A, @RW2+d16	@RW2, #8,rel	@RW2+d16 #8,rel
+ B	ADDL A, A,@RW3	ADDL A, @RW3+d16	SUBL A,@RW3	SUBL A, @RW3+d16	@RW3, #16,rel	@RW3+d16 , #16,rel	CMPL A,@RW3	CMPL A, @RW3+d16	ANDL A,@RW3	ANDL A, @RW3+d16	ORL A,@RW3	ORL A, @RW3+d16	XORL A,@RW3	XORL A, @RW3+d16	@RW3, #8,rel	@RW3+d16 #8,rel
+ C	ADDL A, A,@RW0+	ADDL A, @RW0+RW7	SUBL A,@RW0+	SUBL A, @RW0+RW7	Use prohibited	@RW0+RW7 , #16,rel	CMPL A,@RW0+	CMPL A, @RW0+RW7	ANDL A,@RW0+	ANDL A, @RW0+RW7	ORL A,@RW0+	ORL A, @RW0+RW7	XORL A,@RW0+	XORL A, @RW0+RW7	Use prohibited	@RW0+RW7 #8,rel
+ D	ADDL A, A,@RW1+	ADDL A, @RW1+RW7	SUBL A,@RW1+	SUBL A, @RW1+RW7	Use prohibited	@RW1+RW7 , #16,rel	CMPL A,@RW1+	CMPL A, @RW1+RW7	ANDL A,@RW1+	ANDL A, @RW1+RW7	ORL A,@RW1+	ORL A, @RW1+RW7	XORL A,@RW1+	XORL A, @RW1+RW7	Use prohibited	@RW1+RW7 #8,rel
+ E	ADDL A, @PC+d16	ADDL A, A,@RW2+	SUBL @PC+d16	SUBL A, A,@RW2+	Use prohibited	@PC+d16, #16,rel	CMPL A,@RW2+	CMPL A, @PC+d16	ANDL A,@RW2+	ANDL A, @PC+d16	ORL A,@RW2+	ORL A, @PC+d16	XORL A,@RW2+	XORL A, @PC+d16	Use prohibited	@PC+d16, #8,rel
+ F	ADDL A, A,@RW3+	ADDL A, addr16	SUBL A,@RW3+	SUBL A, addr16	Use prohibited	addr16, #16,rel	CMPL A,@RW3+	CMPL A, addr16	ANDL A,@RW3+	ANDL A, addr16	ORL A,@RW3+	ORL A, addr16	XORL A,@RW3+	XORL A, addr16	Use prohibited	addr16, #8,rel

Table B.9-7 ea Instruction 2 (first byte = 71_H)

	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 ,A	MOV R0,#8	MOV @R ,#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 ,A	MOV R1,#8	MOV @R ,#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 ,A	MOV R2,#8	MOV @R ,#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 ,A	MOV R3,#8	MOV @R ,#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 ,A	MOV R4,#8	MOV @R ,#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 ,A	MOV R5,#8	MOV @R ,#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 ,A	MOV R6,#8	MOV @R ,#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 ,A	MOV R7,#8	MOV @R ,#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 ,A	MOV @RW0,#8	MOV @RW 0+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 ,A	MOV @RW1,#8	MOV @RW 1+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 ,A	MOV @RW2,#8	MOV @RW 2+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 ,A	MOV @RW3,#8	MOV @RW 3+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 ,A	MOV @RW0+,#8	MOV @RW 0+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 ,A	MOV @RW1+,#8	MOV @RW 1+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 ,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 a ,A	MOV @RW3+,#8	MOV a @RW 3+,#8	MOVEA A,@RW3+	MOVEA A, @addr16

Table B.9-8 ea Instruction 3 (first byte = 72_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ROLC R0	ROLC @RW0+d8	RORC R0	RORC @RW0+d8	INC R0	INC @RW0+d8	DEC R0	DEC @RW0+d8	MOV A,R0	MOV A, @RW0+d8	MOV R0,A	MOV @R W0+d8,A	MOVX A,R0	MOVX A, @RW0+d8	XCH A,R0	XCH A, @RW0+d8
+ 1	ROLC R1	ROLC @RW1+d8	RORC R1	RORC @RW1+d8	INC R1	INC @RW1+d8	DEC R1	DEC @RW1+d8	MOV A,R1	MOV A, @RW1+d8	MOV R1,A	MOV @R W1+d8,A	MOVX A,R1	MOVX A, @RW1+d8	XCH A,R1	XCH A, @RW1+d8
+ 2	ROLC R2	ROLC @RW2+d8	RORC R2	RORC @RW2+d8	INC R2	INC @RW2+d8	DEC R2	DEC @RW2+d8	MOV A,R2	MOV A, @RW2+d8	MOV R2,A	MOV @R W2+d8,A	MOVX A,R2	MOVX A, @RW2+d8	XCH A,R2	XCH A, @RW2+d8
+ 3	ROLC R3	ROLC @RW3+d8	RORC R3	RORC @RW3+d8	INC R3	INC @RW3+d8	DEC R3	DEC @RW3+d8	MOV A,R3	MOV A, @RW3+d8	MOV R3,A	MOV @R W3+d8,A	MOVX A,R3	MOVX A, @RW3+d8	XCH A,R3	XCH A, @RW3+d8
+ 4	ROLC R4	ROLC @RW4+d8	RORC R4	RORC @RW4+d8	INC R4	INC @RW4+d8	DEC R4	DEC @RW4+d8	MOV A,R4	MOV A, @RW4+d8	MOV R4,A	MOV @R W4+d8,A	MOVX A,R4	MOVX A, @RW4+d8	XCH A,R4	XCH A, @RW4+d8
+ 5	ROLC R5	ROLC @RW5+d8	RORC R5	RORC @RW5+d8	INC R5	INC @RW5+d8	DEC R5	DEC @RW5+d8	MOV A,R5	MOV A, @RW5+d8	MOV R5,A	MOV @R W5+d8,A	MOVX A,R5	MOVX A, @RW5+d8	XCH A,R5	XCH A, @RW5+d8
+ 6	ROLC R6	ROLC @RW6+d8	RORC R6	RORC @RW6+d8	INC R6	INC @RW6+d8	DEC R6	DEC @RW6+d8	MOV A,R6	MOV A, @RW6+d8	MOV R6,A	MOV @R W6+d8,A	MOVX A,R6	MOVX A, @RW6+d8	XCH A,R6	XCH A, @RW6+d8
+ 7	ROLC R7	ROLC @RW7+d8	RORC R7	RORC @RW7+d8	INC R7	INC @RW7+d8	DEC R7	DEC @RW7+d8	MOV A,R7	MOV A, @RW7+d8	MOV R7,A	MOV @R W7+d8,A	MOVX A,R7	MOVX A, @RW7+d8	XCH A,R7	XCH A, @RW7+d8
+ 8	ROLC @RW0	ROLC @RW0+d16	RORC @RW0	RORC @RW0+d16	INC @RW0	INC @RW0+d16	DEC @RW0	DEC @RW0+d16	MOV A,@RW0	MOV A, @RW0+d16	MOV @RW0,A	MOV @R W0+d16,A	MOVX A,@RW0	MOVX A, @RW0+d16	XCH A,@RW0	XCH A, @RW0+d16
+ 9	ROLC @RW1	ROLC @RW1+d16	RORC @RW1	RORC @RW1+d16	INC @RW1	INC @RW1+d16	DEC @RW1	DEC @RW1+d16	MOV A,@RW1	MOV A, @RW1+d16	MOV @RW1,A	MOV @R W1+d16,A	MOVX A,@RW1	MOVX A, @RW1+d16	XCH A,@RW1	XCH A, @RW1+d16
+ A	ROLC @RW2	ROLC @RW2+d16	RORC @RW2	RORC @RW2+d16	INC @RW2	INC @RW2+d16	DEC @RW2	DEC @RW2+d16	MOV A,@RW2	MOV A, @RW2+d16	MOV @RW2,A	MOV @R W2+d16,A	MOVX A,@RW2	MOVX A, @RW2+d16	XCH A,@RW2	XCH A, @RW2+d16
+ B	ROLC @RW3	ROLC @RW3+d16	RORC @RW3	RORC @RW3+d16	INC @RW3	INC @RW3+d16	DEC @RW3	DEC @RW3+d16	MOV A,@RW3	MOV A, @RW3+d16	MOV @RW3,A	MOV @R W3+d16,A	MOVX A,@RW3	MOVX A, @RW3+d16	XCH A,@RW3	XCH A, @RW3+d16
+ C	ROLC @RW0+	ROLC @RW0+RW7	RORC @RW0+	RORC @RW0+RW7	INC @RW0+	INC @RW0+RW7	DEC @RW0+	DEC @RW0+RW7	MOV A,@RW0+	MOV A, @RW0+RW7	MOV @RW0+,A	MOV @R W0+RW7,A	MOVX A,@RW0+	MOVX A, @RW0+RW7	XCH A,@RW0+	XCH A, @RW0+RW7
+ D	ROLC @RW1+	ROLC @RW1+RW7	RORC @RW1+	RORC @RW1+RW7	INC @RW1+	INC @RW1+RW7	DEC @RW1+	DEC @RW1+RW7	MOV A,@RW1+	MOV A, @RW1+RW7	MOV @RW1+,A	MOV @R W1+RW7,A	MOVX A,@RW1+	MOVX A, @RW1+RW7	XCH A,@RW1+	XCH A, @RW1+RW7
+ E	ROLC @RW2+	ROLC @PC+d16	RORC @RW2+	RORC @PC+d16	INC @RW2+	INC @PC+d16	DEC @RW2+	DEC @PC+d16	MOV A,@RW2+	MOV A, @PC+d16	MOV @RW2+,A	MOV @P C+d16,A	MOVX A,@RW2+	MOVX A, @PC+d16	XCH A,@RW2+	XCH A, @PC+d16
+ F	ROLC @RW3+	ROLC addr16	RORC @RW3+	RORC addr16	INC @RW3+	INC addr16	DEC @RW3+	DEC addr16	MOV A,@RW3+	MOV A, addr16	MOV @RW3+,A	MOV addr16,A	MOVX A,@RW3+	MOVX A, addr16	XCH A,@RW3+	XCH A, addr16

Table B.9-9 ea Instruction 4 (first byte = 73_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	JMP @RW0	JMP @@RW0+d8	CALL @RW0	CALL @@RW0+d8	INCW RW0	INCW @RW0-d8	DECW RW0	DECW @RW0+d8	MOVW A,RW0	MOVW A,@RW0+d8	MOVW RW0,A	MOVW @R W0+d8,A	MOVW RW0,#16	MOVW @RW 0+d8,#16	XCHW A,RW0	XCHW A,@RW0+d8
+ 1	JMP @RW1	JMP @@RW1+d8	CALL @RW1	CALL @@RW1+d8	INCW RW1	INCW @RW1+d8	DECW RW1	DECW @RW1+d8	MOVW A,RW1	MOVW A,@RW1+d8	MOVW RW1,A	MOVW @R W1+d8,A	MOVW RW1,#16	MOVW @RW 1+d8,#16	XCHW A,RW1	XCHW A,@RW1+d8
+ 2	JMP @RW2	JMP @@RW2+d8	CALL @RW2	CALL @@RW2+d8	INCW RW2	INCW @RW2-d8	DECW RW2	DECW @RW2+d8	MOVW A,RW2	MOVW A,@RW2+d8	MOVW RW2,A	MOVW @R W2+d8,A	MOVW RW2,#16	MOVW @RW 2+d8,#16	XCHW A,RW2	XCHW A,@RW2+d8
+ 3	JMP @RW3	JMP @@RW3+d8	CALL @RW3	CALL @@RW3+d8	INCW RW3	INCW @RW3-d8	DECW RW3	DECW @RW3+d8	MOVW A,RW3	MOVW A,@RW3+d8	MOVW RW3,A	MOVW @R W3+d8,A	MOVW RW3,#16	MOVW @RW 3+d8,#16	XCHW A,RW3	XCHW A,@RW3+d8
+ 4	JMP @RW4	JMP @@RW4+d8	CALL @RW4	CALL @@RW4+d8	INCW RW4	INCW @RW4-d8	DECW RW4	DECW @RW4+d8	MOVW A,RW4	MOVW A,@RW4+d8	MOVW RW4,A	MOVW @R W4+d8,A	MOVW RW4,#16	MOVW @RW 4+d8,#16	XCHW A,RW4	XCHW A,@RW4+d8
+ 5	JMP @RW5	JMP @@RW5+d8	CALL @RW5	CALL @@RW5+d8	INCW RW5	INCW @RW5-d8	DECW RW5	DECW @RW5+d8	MOVW A,RW5	MOVW A,@RW5+d8	MOVW RW5,A	MOVW @R W5+d8,A	MOVW RW5,#16	MOVW @RW 5+d8,#16	XCHW A,RW5	XCHW A,@RW5+d8
+ 6	JMP @RW6	JMP @@RW6+d8	CALL @RW6	CALL @@RW6+d8	INCW RW6	INCW @RW6-d8	DECW RW6	DECW @RW6+d8	MOVW A,RW6	MOVW A,@RW6+d8	MOVW RW6,A	MOVW @R W6+d8,A	MOVW RW6,#16	MOVW @RW 6+d8,#16	XCHW A,RW6	XCHW A,@RW6+d8
+ 7	JMP @RW7	JMP @@RW7+d8	CALL @RW7	CALL @@RW7+d8	INCW RW7	INCW @RW7-d8	DECW RW7	DECW @RW7+d8	MOVW A,RW7	MOVW A,@RW7+d8	MOVW RW7,A	MOVW @R W7+d8,A	MOVW RW7,#16	MOVW @RW 7+d8,#16	XCHW A,RW7	XCHW A,@RW7+d8
+ 8	JMP @@RW0	JMP @ @RW0+d16	CALL @@RW0	CALL @ @RW0+d16	INCW @RW0	INCW @RW0-d16	DECW @RW0	DECW @RW0+d16	MOVW A,@RW0	MOVW A,@RW0+d16	MOVW @RW0,A	MOVW @R W0+d16,A	MOVW @RW0,#16	MOVW@RW0 +d16,#16	XCHW A,@RW0	XCHW A,@RW0+d16
+ 9	JMP @@RW1	JMP @ @RW1+d16	CALL @@RW1	CALL @ @RW1+d16	INCW @RW1	INCW @RW1+d16	DECW @RW1	DECW @RW1+d16	MOVW A,@RW1	MOVW A,@RW1+d16	MOVW @RW1,A	MOVW @R W1+d16,A	MOVW @RW1,#16	MOVW@RW1 +d16,#16	XCHW A,@RW1	XCHW A,@RW1+d16
+ A	JMP @@RW2	JMP @ @RW2+d16	CALL @@RW2	CALL @ @RW2+d16	INCW @RW2	INCW @RW2-d16	DECW @RW2	DECW @RW2+d16	MOVW A,@RW2	MOVW A,@RW2+d16	MOVW @RW2,A	MOVW @R W2+d16,A	MOVW @RW2,#16	MOVW@RW2 +d16,#16	XCHW A,@RW2	XCHW A,@RW2+d16
+ B	JMP @@RW3	JMP @ @RW3+d16	CALL @@RW3	CALL @ @RW3+d16	INCW @RW3	INCW @RW3-d16	DECW @RW3	DECW @RW3+d16	MOVW A,@RW3	MOVW A,@RW3+d16	MOVW @RW3,A	MOVW @R W3+d16,A	MOVW @RW3,#16	MOVW@RW3 +d16,#16	XCHW A,@RW3	XCHW A,@RW3+d16
+ C	JMP @@RW0+	JMP @ @RW0+RW7	CALL @@RW0+	CALL @ @RW0+RW7	INCW @RW0+	INCW @RW0+RW7	DECW @RW0+	DECW @RW0+RW7	MOVW A,@RW0+	MOVW A,@RW0+RW7	MOVW @RW0+,A	MOVW @R W0+RW7,A	MOVW @ RW0+,#16	MOVW@RW0 +RW7,#16	XCHW A,@RW0+	XCHW A,@RW0+RW7
+ D	JMP @@RW1+	JMP @ @RW1+RW7	CALL @@RW1+	CALL @ @RW1+RW7	INCW @RW1+	INCW @RW1+RW7	DECW @RW1+	DECW @RW1+RW7	MOVW A,@RW1+	MOVW A,@RW1+RW7	MOVW @RW1+,A	MOVW @R W1+RW7,A	MOVW @ RW1+,#16	MOVW@RW1 +RW7,#16	XCHW A,@RW1+	XCHW A,@RW1+RW7
+ E	JMP @@RW2+	JMP @@PC+d16	CALL @@RW2+	CALL @@PC+d16	INCW @RW2+	INCW @PC+d16	DECW @RW2+	DECW @PC+d16	MOVW A,@RW2+	MOVW A,@PC+d16	MOVW @RW2+,A	MOVW @P C+d16,A	MOVW @ RW2+,#16	MOVW @PC +d16,#16	XCHW A,@RW2+	XCHW A,@PC+d16
+ F	JMP @@RW3+	JMP @addr16	CALL @@RW3+	CALL @addr16	INCW @RW3+	INCW addr16	DECW @RW3+	DECW addr16	MOVW A,@RW3+	MOVW A,addr16	MOVW @RW3+,A	MOVW @P C+d16,A	MOVW @ RW3+,#16	MOVW ad dr16,#16	XCHW A,@RW3+	XCHW A,addr16

Table B.9-10 ea Instruction 5 (first byte = 74_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADD A, A,R0	ADD A, @RW0+d8	SUB A, A,R0	SUB A, @RW0+d8	ADDC A, A,R0	ADDC A, @RW0+d8	CMP A, A,R0	CMP A, @RW0+d8	AND A, A,R0	AND A, @RW0+d8	OR A, A,R0	OR A, @RW0+d8	XOR A, A,R0	XOR A, @RW0+d8	DBNZ R0,r	DBNZ @ RW0+d8,r
+ 1	ADD A, A,R1	ADD A, @RW1+d8	SUB A, A,R1	SUB A, @RW1+d8	ADDC A, A,R1	ADDC A, @RW1+d8	CMP A, A,R1	CMP A, @RW1+d8	AND A, A,R1	AND A, @RW1+d8	OR A, A,R1	OR A, @RW1+d8	XOR A, A,R1	XOR A, @RW1+d8	DBNZ R1,r	DBNZ @ RW1+d8,r
+ 2	ADD A, A,R2	ADD A, @RW2+d8	SUB A, A,R2	SUB A, @RW2+d8	ADDC A, A,R2	ADDC A, @RW2+d8	CMP A, A,R2	CMP A, @RW2+d8	AND A, A,R2	AND A, @RW2+d8	OR A, A,R2	OR A, @RW2+d8	XOR A, A,R2	XOR A, @RW2+d8	DBNZ R2,r	DBNZ @ RW2+d8,r
+ 3	ADD A, A,R3	ADD A, @RW3+d8	SUB A, A,R3	SUB A, @RW3+d8	ADDC A, A,R3	ADDC A, @RW3+d8	CMP A, A,R3	CMP A, @RW3+d8	AND A, A,R3	AND A, @RW3+d8	OR A, A,R3	OR A, @RW3+d8	XOR A, A,R3	XOR A, @RW3+d8	DBNZ R3,r	DBNZ @ RW3+d8,r
+ 4	ADD A, A,R4	ADD A, @RW4+d8	SUB A, A,R4	SUB A, @RW4+d8	ADDC A, A,R4	ADDC A, @RW4+d8	CMP A, A,R4	CMP A, @RW4+d8	AND A, A,R4	AND A, @RW4+d8	OR A, A,R4	OR A, @RW4+d8	XOR A, A,R4	XOR A, @RW4+d8	DBNZ R4,r	DBNZ @ RW4+d8,r
+ 5	ADD A, A,R5	ADD A, @RW5+d8	SUB A, A,R5	SUB A, @RW5+d8	ADDC A, A,R5	ADDC A, @RW5+d8	CMP A, A,R5	CMP A, @RW5+d8	AND A, A,R5	AND A, @RW5+d8	OR A, A,R5	OR A, @RW5+d8	XOR A, A,R5	XOR A, @RW5+d8	DBNZ R5,r	DBNZ @ RW5+d8,r
+ 6	ADD A, A,R6	ADD A, @RW6+d8	SUB A, A,R6	SUB A, @RW6+d8	ADDC A, A,R6	ADDC A, @RW6+d8	CMP A, A,R6	CMP A, @RW6+d8	AND A, A,R6	AND A, @RW6+d8	OR A, A,R6	OR A, @RW6+d8	XOR A, A,R6	XOR A, @RW6+d8	DBNZ R6,r	DBNZ @ RW6+d8,r
+ 7	ADD A, A,R7	ADD A, @RW7+d8	SUB A, A,R7	SUB A, @RW7+d8	ADDC A, A,R7	ADDC A, @RW7+d8	CMP A, A,R7	CMP A, @RW7+d8	AND A, A,R7	AND A, @RW7+d8	OR A, A,R7	OR A, @RW7+d8	XOR A, A,R7	XOR A, @RW7+d8	DBNZ R7,r	DBNZ @ RW7+d8,r
+ 8	ADD A, A,@RW0	ADD A, @RW0+d16	SUB A, A,@RW0	SUB A, @RW0+d16	ADDC A, A,@RW0	ADDC A, @RW0+d16	CMP A, A,@RW0	CMP A, @RW0+d16	AND A, A,@RW0	AND A, @RW0+d16	OR A, A,@RW0	OR A, @RW0+d16	XOR A, A,@RW0	XOR A, @RW0+d16	DBNZ @RW0,r	DBNZ @R W0+d16,r
+ 9	ADD A, A,@RW1	ADD A, @RW1+d16	SUB A, A,@RW1	SUB A, @RW1+d16	ADDC A, A,@RW1	ADDC A, @RW1+d16	CMP A, A,@RW1	CMP A, @RW1+d16	AND A, A,@RW1	AND A, @RW1+d16	OR A, A,@RW1	OR A, @RW1+d16	XOR A, A,@RW1	XOR A, @RW1+d16	DBNZ @RW1,r	DBNZ @R W1+d16,r
+ A	ADD A, A,@RW2	ADD A, @RW2+d16	SUB A, A,@RW2	SUB A, @RW2+d16	ADDC A, A,@RW2	ADDC A, @RW2+d16	CMP A, A,@RW2	CMP A, @RW2+d16	AND A, A,@RW2	AND A, @RW2+d16	OR A, A,@RW2	OR A, @RW2+d16	XOR A, A,@RW2	XOR A, @RW2+d16	DBNZ @RW2,r	DBNZ @R W2+d16,r
+ B	ADD A, A,@RW3	ADD A, @RW3+d16	SUB A, A,@RW3	SUB A, @RW3+d16	ADDC A, A,@RW3	ADDC A, @RW3+d16	CMP A, A,@RW3	CMP A, @RW3+d16	AND A, A,@RW3	AND A, @RW3+d16	OR A, A,@RW3	OR A, @RW3+d16	XOR A, A,@RW3	XOR A, @RW3+d16	DBNZ @RW3,r	DBNZ @R W3+d16,r
+ C	ADD A, A,@RW0+	ADD A, @RW0+RW7	SUB A, A,@RW0+	SUB A, @RW0+RW7	ADDC A, A,@RW0+	ADDC A, @RW0+RW7	CMP A, A,@RW0+	CMP A, @RW0+RW7	AND A, A,@RW0+	AND A, @RW0+RW7	OR A, A,@RW0+	OR A, @RW0+RW7	XOR A, A,@RW0+	XOR A, @RW0+RW7	DBNZ @RW0+,r	DBNZ @R W0+RW7,r
+ D	ADD A, A,@RW1+	ADD A, @RW1+RW7	SUB A, A,@RW1+	SUB A, @RW1+RW7	ADDC A, A,@RW1+	ADDC A, @RW1+RW7	CMP A, A,@RW1+	CMP A, @RW1+RW7	AND A, A,@RW1+	AND A, @RW1+RW7	OR A, A,@RW1+	OR A, @RW1+RW7	XOR A, A,@RW1+	XOR A, @RW1+RW7	DBNZ @RW1+,r	DBNZ @R W1+RW7,r
+ E	ADD A, A,@RW2+	ADD A, @PC+d16	SUB A, A,@RW2+	SUB A, @PC+d16	ADDC A, A,@RW2+	ADDC A, @PC+d16	CMP A, A,@RW2+	CMP A, @PC+d16	AND A, A,@RW2+	AND A, @PC+d16	OR A, A,@RW2+	OR A, @PC+d16	XOR A, A,@RW2+	XOR A, @PC+d16	DBNZ @RW2+,r	DBNZ @ PC+d16,r
+ F	ADD A, A,@RW3+	ADD A, addr16	SUB A, A,@RW3+	SUB A, addr16	ADDC A, A,@RW3+	ADDC A, addr16	CMP A, A,@RW3+	CMP A, addr16	AND A, A,@RW3+	AND A, addr16	OR A, A,@RW3+	OR A, addr16	XOR A, A,@RW3+	XOR A, addr16	DBNZ @RW3+,r	DBNZ addr16,r

Table B.9-11 ea Instruction 6 (first byte = 75_H)

	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 A, @RW0+d8	NEG R0	NEG @RW0+d8	AND R0,A	AND @R W0+d8,A	OR R0,A	OR @R W0+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 A, @RW1+d8	NEG R1	NEG @RW1+d8	AND R1,A	AND @R W1+d8,A	OR R1,A	OR @R W1+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 A, @RW2+d8	NEG R2	NEG @RW2+d8	AND R2,A	AND @R W2+d8,A	OR R2,A	OR @R W2+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 A, @RW3+d8	NEG R3	NEG @RW3+d8	AND R3,A	AND @R W3+d8,A	OR R3,A	OR @R W3+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 A, @RW4+d8	NEG R4	NEG @RW4+d8	AND R4,A	AND @R W4+d8,A	OR R4,A	OR @R W4+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 A, @RW5+d8	NEG R5	NEG @RW5+d8	AND R5,A	AND @R W5+d8,A	OR R5,A	OR @R W5+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 A, @RW6+d8	NEG R6	NEG @RW6+d8	AND R6,A	AND @R W6+d8,A	OR R6,A	OR @R W6+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 A, @RW7+d8	NEG R7	NEG @RW7+d8	AND R7,A	AND @R W7+d8,A	OR R7,A	OR @R W7+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 A, @RW0+d16	NEG @RW0	NEG @RW0+d16	AND @RW0,A	AND @R W0+d16,A	OR @RW0,A	OR @R W0+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 A, @RW1+d16	NEG @RW1	NEG @RW1+d16	AND @RW1,A	AND @R W1+d16,A	OR @RW1,A	OR @R W1+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 A, @RW2+d16	NEG @RW2	NEG @RW2+d16	AND @RW2,A	AND @R W2+d16,A	OR @RW2,A	OR @R W2+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 A, @RW3+d16	NEG @RW3	NEG @RW3+d16	AND @RW3,A	AND @R W3+d16,A	OR @RW3,A	OR @R W3+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 A, @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 A, @RW1+RW7	NEG @RW1+	NEG @RW1+RW7	AND @RW1+,A	AND @R W1+RW7,A	OR @RW1+,A	OR @R W1+RW7,A	XOR @RW1+,A	XOR @R W1+RW7,A	NOT @RW1+	NOT @RW1+RW7
+ E	ADD @RW2+,A	ADD @P C+d16,A	SUB @RW2+,A	SUB @P C+d16,A	SUBC A,@RW2+	SUBC A, @PC+d16	NEG @RW2+	NEG @PC+d16	AND @RW2+,A	AND @P C+d16,A	OR @RW2+,A	OR @P C+d16,A	XOR @RW2+,A	XOR @P C+d16,A	NOT @RW2+	NOT @PC+d16
+ F	ADD @RW3+,A	ADD addr16,A	SUB @RW3+,A	SUB addr16,A	SUBC A,@RW3+	SUBC A, addr16	NEG @RW3+	NEG addr16	AND @RW3+,A	AND addr16,A	OR @RW3+,A	OR addr16,A	XOR @RW3+,A	XOR addr16,A	NOT @RW3+	NOT addr16

Table B.9-12 ea Instruction 7 (first byte = 76_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADDW A,@RW0	ADDW A,@RW0+d8	SUBW A,@RW0	SUBW A,@RW0+d8	ADDCW A,@RW0	ADDCW A,@RW0+d8	CMPW A,@RW0	CMPW A,@RW0+d8	ANDW A,@RW0	ANDW A,@RW0+d8	ORW A,@RW0	ORW A,@RW0+d8	XORW A,@RW0	XORW A,@RW0+d8	DWBNZ RW0,r	DWBNZ @RW0+d8,r
+ 1	ADDW A,@RW1	ADDW A,@RW1+d8	SUBW A,@RW1	SUBW A,@RW1+d8	ADDCW A,@RW1	ADDCW A,@RW1+d8	CMPW A,@RW1	CMPW A,@RW1+d8	ANDW A,@RW1	ANDW A,@RW1+d8	ORW A,@RW1	ORW A,@RW1+d8	XORW A,@RW1	XORW A,@RW1+d8	DWBNZ RW1,r	DWBNZ @RW1+d8,r
+ 2	ADDW A,@RW2	ADDW A,@RW2+d8	SUBW A,@RW2	SUBW A,@RW2+d8	ADDCW A,@RW2	ADDCW A,@RW2+d8	CMPW A,@RW2	CMPW A,@RW2+d8	ANDW A,@RW2	ANDW A,@RW2+d8	ORW A,@RW2	ORW A,@RW2+d8	XORW A,@RW2	XORW A,@RW2+d8	DWBNZ RW2,r	DWBNZ @RW2+d8,r
+ 3	ADDW A,@RW3	ADDW A,@RW3+d8	SUBW A,@RW3	SUBW A,@RW3+d8	ADDCW A,@RW3	ADDCW A,@RW3+d8	CMPW A,@RW3	CMPW A,@RW3+d8	ANDW A,@RW3	ANDW A,@RW3+d8	ORW A,@RW3	ORW A,@RW3+d8	XORW A,@RW3	XORW A,@RW3+d8	DWBNZ RW3,r	DWBNZ @RW3+d8,r
+ 4	ADDW A,@RW4	ADDW A,@RW4+d8	SUBW A,@RW4	SUBW A,@RW4+d8	ADDCW A,@RW4	ADDCW A,@RW4+d8	CMPW A,@RW4	CMPW A,@RW4+d8	ANDW A,@RW4	ANDW A,@RW4+d8	ORW A,@RW4	ORW A,@RW4+d8	XORW A,@RW4	XORW A,@RW4+d8	DWBNZ RW4,r	DWBNZ @RW4+d8,r
+ 5	ADDW A,@RW5	ADDW A,@RW5+d8	SUBW A,@RW5	SUBW A,@RW5+d8	ADDCW A,@RW5	ADDCW A,@RW5+d8	CMPW A,@RW5	CMPW A,@RW5+d8	ANDW A,@RW5	ANDW A,@RW5+d8	ORW A,@RW5	ORW A,@RW5+d8	XORW A,@RW5	XORW A,@RW5+d8	DWBNZ RW5,r	DWBNZ @RW5+d8,r
+ 6	ADDW A,@RW6	ADDW A,@RW6+d8	SUBW A,@RW6	SUBW A,@RW6+d8	ADDCW A,@RW6	ADDCW A,@RW6+d8	CMPW A,@RW6	CMPW A,@RW6+d8	ANDW A,@RW6	ANDW A,@RW6+d8	ORW A,@RW6	ORW A,@RW6+d8	XORW A,@RW6	XORW A,@RW6+d8	DWBNZ RW6,r	DWBNZ @RW6+d8,r
+ 7	ADDW A,@RW7	ADDW A,@RW7+d8	SUBW A,@RW7	SUBW A,@RW7+d8	ADDCW A,@RW7	ADDCW A,@RW7+d8	CMPW A,@RW7	CMPW A,@RW7+d8	ANDW A,@RW7	ANDW A,@RW7+d8	ORW A,@RW7	ORW A,@RW7+d8	XORW A,@RW7	XORW A,@RW7+d8	DWBNZ RW7,r	DWBNZ @RW7+d8,r
+ 8	ADDW A,@RW0	ADDW A,@RW0+d16	SUBW A,@RW0	SUBW A,@RW0+d16	ADDCW A,@RW0	ADDCW A,@RW0+d16	CMPW A,@RW0	CMPW A,@RW0+d16	ANDW A,@RW0	ANDW A,@RW0+d16	ORW A,@RW0	ORW A,@RW0+d16	XORW A,@RW0	XORW A,@RW0+d16	DWBNZ @RW0,r	DWBNZ @R0+d16,r
+ 9	ADDW A,@RW1	ADDW A,@RW1+d16	SUBW A,@RW1	SUBW A,@RW1+d16	ADDCW A,@RW1	ADDCW A,@RW1+d16	CMPW A,@RW1	CMPW A,@RW1+d16	ANDW A,@RW1	ANDW A,@RW1+d16	ORW A,@RW1	ORW A,@RW1+d16	XORW A,@RW1	XORW A,@RW1+d16	DWBNZ @RW1,r	DWBNZ @R1+d16,r
+ A	ADDW A,@RW2	ADDW A,@RW2+d16	SUBW A,@RW2	SUBW A,@RW2+d16	ADDCW A,@RW2	ADDCW A,@RW2+d16	CMPW A,@RW2	CMPW A,@RW2+d16	ANDW A,@RW2	ANDW A,@RW2+d16	ORW A,@RW2	ORW A,@RW2+d16	XORW A,@RW2	XORW A,@RW2+d16	DWBNZ @RW2,r	DWBNZ @R2+d16,r
+ B	ADDW A,@RW3	ADDW A,@RW3+d16	SUBW A,@RW3	SUBW A,@RW3+d16	ADDCW A,@RW3	ADDCW A,@RW3+d16	CMPW A,@RW3	CMPW A,@RW3+d16	ANDW A,@RW3	ANDW A,@RW3+d16	ORW A,@RW3	ORW A,@RW3+d16	XORW A,@RW3	XORW A,@RW3+d16	DWBNZ @RW3,r	DWBNZ @R3+d16,r
+ C	ADDW A,@RW0+	ADDW A,@RW0+RW7	SUBW A,@RW0+	SUBW A,@RW0+RW7	ADDCW A,@RW0+	ADDCW A,@RW0+RW7	CMPW A,@RW0+	CMPW A,@RW0+RW7	ANDW A,@RW0+	ANDW A,@RW0+RW7	ORW A,@RW0+	ORW A,@RW0+RW7	XORW A,@RW0+	XORW A,@RW0+RW7	DWBNZ @RW0+,r	DWBNZ @R0+RW7,r
+ D	ADDW A,@RW1+	ADDW A,@RW1+RW7	SUBW A,@RW1+	SUBW A,@RW1+RW7	ADDCW A,@RW1+	ADDCW A,@RW1+RW7	CMPW A,@RW1+	CMPW A,@RW1+RW7	ANDW A,@RW1+	ANDW A,@RW1+RW7	ORW A,@RW1+	ORW A,@RW1+RW7	XORW A,@RW1+	XORW A,@RW1+RW7	DWBNZ @RW1+,r	DWBNZ @R1+RW7,r
+ E	ADDW A,@RW2+	ADDW A,@PC+d16	SUBW A,@RW2+	SUBW A,@PC+d16	ADDCW A,@RW2+	ADDCW A,@PC+d16	CMPW A,@RW2+	CMPW A,@PC+d16	ANDW A,@RW2+	ANDW A,@PC+d16	ORW A,@RW2+	ORW A,@PC+d16	XORW A,@RW2+	XORW A,@PC+d16	DWBNZ @RW2+,r	DWBNZ @PC+d16,r
+ F	ADDW A,@RW3+	ADDW A,addr16	SUBW A,@RW3+	SUBW A,addr16	ADDCW A,@RW3+	ADDCW A,addr16	CMPW A,@RW3+	CMPW A,addr16	ANDW A,@RW3+	ANDW A,addr16	ORW A,@RW3+	ORW A,addr16	XORW A,@RW3+	XORW A,addr16	DWBNZ @RW3+,r	DWBNZ addr16,r

Table B.9-13 ea Instruction 8 (first byte = 77_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW0,A	W0+d8,A	RW0,A	W0+d8,A	A,RW0	@RW0+d8	RW0	@RW0+d8	RW0,A	W0+d8,A	RW0,A	W0+d8,A	RW0,A	W0+d8,A	RW0	@RW0+d8
+ 1	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW1,A	W1+d8,A	RW1,A	W1+d8,A	A,RW1	@RW1+d8	RW1	@RW1+d8	RW1,A	W1+d8,A	RW1,A	W1+d8,A	RW1,A	W1+d8,A	RW1	@RW1+d8
+ 2	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW2,A	W2+d8,A	RW2,A	W2+d8,A	A,RW2	@RW2+d8	RW2	@RW2+d8	RW2,A	W2+d8,A	RW2,A	W2+d8,A	RW2,A	W2+d8,A	RW2	@RW2+d8
+ 3	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW3,A	W3+d8,A	RW3,A	W3+d8,A	A,RW3	@RW3+d8	RW3	@RW3+d8	RW3,A	W3+d8,A	RW3,A	W3+d8,A	RW3,A	W3+d8,A	RW3	@RW3+d8
+ 4	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW4,A	W4+d8,A	RW4,A	W4+d8,A	A,RW4	@RW4+d8	RW4	@RW4+d8	RW4,A	W4+d8,A	RW4,A	W4+d8,A	RW4,A	W4+d8,A	RW4	@RW4+d8
+ 5	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW5,A	W5+d8,A	RW5,A	W5+d8,A	A,RW5	@RW5+d8	RW5	@RW5+d8	RW5,A	W5+d8,A	RW5,A	W5+d8,A	RW5,A	W5+d8,A	RW5	@RW5+d8
+ 6	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW6,A	W6+d8,A	RW6,A	W6+d8,A	A,RW6	@RW6+d8	RW6	@RW6+d8	RW6,A	W6+d8,A	RW6,A	W6+d8,A	RW6,A	W6+d8,A	RW6	@RW6+d8
+ 7	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	RW7,A	W7+d8,A	RW7,A	W7+d8,A	A,RW7	@RW7+d8	RW7	@RW7+d8	RW7,A	W7+d8,A	RW7,A	W7+d8,A	RW7,A	W7+d8,A	RW7	@RW7+d8
+ 8	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW0,A	W0+d16,A	@RW0,A	W0+d16,A	A,@RW0	@RW0+d16	@RW0	@RW0+d16	@RW0,A	W0+d16,A	@RW0,A	W0+d16,A	@RW0,A	W0+d16,A	@RW0	@RW0+d16
+ 9	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW1,A	W1+d16,A	@RW1,A	W1+d16,A	A,@RW1	@RW1+d16	@RW1	@RW1+d16	@RW1,A	W1+d16,A	@RW1,A	W1+d16,A	@RW1,A	W1+d16,A	@RW1	@RW1+d16
+ A	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW2,A	W2+d16,A	@RW2,A	W2+d16,A	A,@RW2	@RW2+d16	@RW2	@RW2+d16	@RW2,A	W2+d16,A	@RW2,A	W2+d16,A	@RW2,A	W2+d16,A	@RW2	@RW2+d16
+ B	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW3,A	W3+d16,A	@RW3,A	W3+d16,A	A,@RW3	@RW3+d16	@RW3	@RW3+d16	@RW3,A	W3+d16,A	@RW3,A	W3+d16,A	@RW3,A	W3+d16,A	@RW3	@RW3+d16
+ C	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW0+,A	W0+RW7,A	@RW0+,A	W0+RW7,A	A,@RW0+	@RW0+RW7	@RW0+	@RW0+RW7	@RW0+,A	W0+RW7,A	@RW0+,A	W0+RW7,A	@RW0+,A	W0+RW7,A	@RW0+	@RW0+RW7
+ D	ADDW	ADDW @R	SUBW	SUBW @R	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @R	ORW	ORW @R	XORW	XORW @R	NOTW	NOTW
	@RW1+,A	W1+RW7,A	@RW1+,A	W1+RW7,A	A,@RW1+	@RW1+RW7	@RW1+	@RW1+RW7	@RW1+,A	W1+RW7,A	@RW1+,A	W1+RW7,A	@RW1+,A	W1+RW7,A	@RW1+	@RW1+RW7
+ E	ADDW	ADDW @P	SUBW	SUBW @P	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW @P	ORW	ORW @P	XORW	XORW @P	NOTW	NOTW
	@RW2+,A	C+d16,A	@RW2+,A	C+d16,A	A,@RW2+	@PC+d16	@RW2+	@PC+d16	@RW2+,A	C+d16,A	@RW2+,A	C+d16,A	@RW2+,A	C+d16,A	@RW2+	@PC+d16
+ F	ADDW	ADDW	SUBW	SUBW	SUBCW	SUBCW A,	NEGW	NEGW	ANDW	ANDW	ORW	ORW	XORW	XORW	NOTW	NOTW
	@RW3+,A	addr16,A	@RW3+,A	addr16,A	A,@RW3+	addr16	@RW3+	addr16	@RW3+,A	addr16,A	@RW3+,A	addr16,A	@RW3+,A	addr16,A	@RW3+	addr16

Table B.9-14 ea Instruction 9 (first byte = 78_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MULU A,R0	MULU A,@RW0+d8	MULUW A,@RW0	MULUW A,@RW0+d8					DIVU A,R0	DIV A,@RW0+d8	DIVUW A,@RW0	DIVUW A,@RW0+d8				
+ 1	MULU A,R1	MULU A,@RW1+d8	MULUW A,@RW1	MULUW A,@RW1+d8					DIVU A,R1	DIV A,@RW1+d8	DIVUW A,@RW1	DIVUW A,@RW1+d8				
+ 2	MULU A,R2	MULU A,@RW2+d8	MULUW A,@RW2	MULUW A,@RW2+d8					DIVU A,R2	DIV A,@RW2+d8	DIVUW A,@RW2	DIVUW A,@RW2+d8				
+ 3	MULU A,R3	MULU A,@RW3+d8	MULUW A,@RW3	MULUW A,@RW3+d8					DIVU A,R3	DIV A,@RW3+d8	DIVUW A,@RW3	DIVUW A,@RW3+d8				
+ 4	MULU A,R4	MULU A,@RW4+d8	MULUW A,@RW4	MULUW A,@RW4+d8					DIVU A,R4	DIV A,@RW4+d8	DIVUW A,@RW4	DIVUW A,@RW4+d8				
+ 5	MULU A,R5	MULU A,@RW5+d8	MULUW A,@RW5	MULUW A,@RW5+d8					DIVU A,R5	DIV A,@RW5+d8	DIVUW A,@RW5	DIVUW A,@RW5+d8				
+ 6	MULU A,R6	MULU A,@RW6+d8	MULUW A,@RW6	MULUW A,@RW6+d8					DIVU A,R6	DIV A,@RW6+d8	DIVUW A,@RW6	DIVUW A,@RW6+d8				
+ 7	MULU A,R7	MULU A,@RW7+d8	MULUW A,@RW7	MULUW A,@RW7+d8					DIVU A,R7	DIV A,@RW7+d8	DIVUW A,@RW7	DIVUW A,@RW7+d8				
+ 8	MULU A,@RW0	MULU A,@RW0+d16	MULUW A,@RW0	MULUW A,@RW0+d16					DIVU A,@RW0	DIV A,@RW0+d16	DIVUW A,@RW0	DIVUW A,@RW0+d16				
+ 9	MULU A,@RW1	MULU A,@RW1+d16	MULUW A,@RW1	MULUW A,@RW1+d16					DIVU A,@RW1	DIV A,@RW1+d16	DIVUW A,@RW1	DIVUW A,@RW1+d16				
+ A	MULU A,@RW2	MULU A,@RW2+d16	MULUW A,@RW2	MULUW A,@RW2+d16					DIVU A,@RW2	DIV A,@RW2+d16	DIVUW A,@RW2	DIVUW A,@RW2+d16				
+ B	MULU A,@RW3	MULU A,@RW3+d16	MULUW A,@RW3	MULUW A,@RW3+d16					DIVU A,@RW3	DIV A,@RW3+d16	DIVUW A,@RW3	DIVUW A,@RW3+d16				
+ C	MULU A,@RW0+	MULU A,@RW0+RW7	MULUW A,@RW0+	MULUW A,@RW0+RW7					DIVU A,@RW0+	DIV A,@RW0+RW7	DIVUW A,@RW0+	DIVUW A,@RW0+RW7				
+ D	MULU A,@RW1+	MULU A,@RW1+RW7	MULUW A,@RW1+	MULUW A,@RW1+RW7					DIVU A,@RW1+	DIV A,@RW1+RW7	DIVUW A,@RW1+	DIVUW A,@RW1+RW7				
+ E	MULU A,@RW2+	MULU A,@PC+d16	MULUW A,@RW2+	MULUW A,@PC+d16					DIVU A,@RW2+	DIV A,@PC+d16	DIVUW A,@RW2+	DIVUW A,@PC+d16				
+ F	MULU A,@RW3+	MULU A,addr16	MULUW A,@RW3+	MULUW A,addr16					DIVU A,@RW3+	DIV A,addr16	DIVUW A,@RW3+	DIVUW A,addr16				

Table B.9-15 MOVEA RWi, ea Instruction (first byte = 79_H)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVEA RW0,RW0	MOVEA RW0 ,@RW0+d8	MOVEA RW1,RW0	MOVEA RW1 ,@RW0+d8	MOVEA RW2,RW0	MOVEA RW2 ,@RW0+d8	MOVEA RW3,RW0	MOVEA RW3 ,@RW0+d8	MOVEA RW4,RW0	MOVEA RW4 ,@RW0+d8	MOVEA RW5,RW0	MOVEA RW5 ,@RW0+d8	MOVEA RW6,RW0	MOVEA RW6 ,@RW0+d8	MOVEA RW7,RW0	MOVEA RW7 ,@RW0+d8
+ 1	MOVEA RW0,RW1	MOVEA RW0 ,@RW1+d8	MOVEA RW1,RW1	MOVEA RW1 ,@RW1+d8	MOVEA RW2,RW1	MOVEA RW2 ,@RW1+d8	MOVEA RW3,RW1	MOVEA RW3 ,@RW1+d8	MOVEA RW4,RW1	MOVEA RW4 ,@RW1+d8	MOVEA RW5,RW1	MOVEA RW5 ,@RW1+d8	MOVEA RW6,RW1	MOVEA RW6 ,@RW1+d8	MOVEA RW7,RW1	MOVEA RW7 ,@RW1+d8
+ 2	MOVEA RW0,RW2	MOVEA RW0 ,@RW2+d8	MOVEA RW1,RW2	MOVEA RW1 ,@RW2+d8	MOVEA RW2,RW2	MOVEA RW2 ,@RW2+d8	MOVEA RW3,RW2	MOVEA RW3 ,@RW2+d8	MOVEA RW4,RW2	MOVEA RW4 ,@RW2+d8	MOVEA RW5,RW2	MOVEA RW5 ,@RW2+d8	MOVEA RW6,RW2	MOVEA RW6 ,@RW2+d8	MOVEA RW7,RW2	MOVEA RW7 ,@RW2+d8
+ 3	MOVEA RW0,RW3	MOVEA RW0 ,@RW3+d8	MOVEA RW1,RW3	MOVEA RW1 ,@RW3+d8	MOVEA RW2,RW3	MOVEA RW2 ,@RW3+d8	MOVEA RW3,RW3	MOVEA RW3 ,@RW3+d8	MOVEA RW4,RW3	MOVEA RW4 ,@RW3+d8	MOVEA RW5,RW3	MOVEA RW5 ,@RW3+d8	MOVEA RW6,RW3	MOVEA RW6 ,@RW3+d8	MOVEA RW7,RW3	MOVEA RW7 ,@RW3+d8
+ 4	MOVEA RW0,RW4	MOVEA RW0 ,@RW4+d8	MOVEA RW1,RW4	MOVEA RW1 ,@RW4+d8	MOVEA RW2,RW4	MOVEA RW2 ,@RW4+d8	MOVEA RW3,RW4	MOVEA RW3 ,@RW4+d8	MOVEA RW4,RW4	MOVEA RW4 ,@RW4+d8	MOVEA RW5,RW4	MOVEA RW5 ,@RW4+d8	MOVEA RW6,RW4	MOVEA RW6 ,@RW4+d8	MOVEA RW7,RW4	MOVEA RW7 ,@RW4+d8
+ 5	MOVEA RW0,RW5	MOVEA RW0 ,@RW5+d8	MOVEA RW1,RW5	MOVEA RW1 ,@RW5+d8	MOVEA RW2,RW5	MOVEA RW2 ,@RW5+d8	MOVEA RW3,RW5	MOVEA RW3 ,@RW5+d8	MOVEA RW4,RW5	MOVEA RW4 ,@RW5+d8	MOVEA RW5,RW5	MOVEA RW5 ,@RW5+d8	MOVEA RW6,RW5	MOVEA RW6 ,@RW5+d8	MOVEA RW7,RW5	MOVEA RW7 ,@RW5+d8
+ 6	MOVEA RW0,RW6	MOVEA RW0 ,@RW6+d8	MOVEA RW1,RW6	MOVEA RW1 ,@RW6+d8	MOVEA RW2,RW6	MOVEA RW2 ,@RW6+d8	MOVEA RW3,RW6	MOVEA RW3 ,@RW6+d8	MOVEA RW4,RW6	MOVEA RW4 ,@RW6+d8	MOVEA RW5,RW6	MOVEA RW5 ,@RW6+d8	MOVEA RW6,RW6	MOVEA RW6 ,@RW6+d8	MOVEA RW7,RW6	MOVEA RW7 ,@RW6+d8
+ 7	MOVEA RW0,RW7	MOVEA RW0 ,@RW7+d8	MOVEA RW1,RW7	MOVEA RW1 ,@RW7+d8	MOVEA RW2,RW7	MOVEA RW2 ,@RW7+d8	MOVEA RW3,RW7	MOVEA RW3 ,@RW7+d8	MOVEA RW4,RW7	MOVEA RW4 ,@RW7+d8	MOVEA RW5,RW7	MOVEA RW5 ,@RW7+d8	MOVEA RW6,RW7	MOVEA RW6 ,@RW7+d8	MOVEA RW7,RW7	MOVEA RW7 ,@RW7+d8
+ 8	MOVEA RW0,@RW0	MOVEA RW0 ,@RW0+d16	MOVEA RW1,@RW0	MOVEA RW1 ,@RW0+d16	MOVEA RW2,@RW0	MOVEA RW2 ,@RW0+d16	MOVEA RW3,@RW0	MOVEA RW3 ,@RW0+d16	MOVEA RW4,@RW0	MOVEA RW4 ,@RW0+d16	MOVEA RW5,@RW0	MOVEA RW5 ,@RW0+d16	MOVEA RW6,@RW0	MOVEA RW6 ,@RW0+d16	MOVEA RW7,@RW0	MOVEA RW7 ,@RW0+d16
+ 9	MOVEA RW0,@RW1	MOVEA RW0 ,@RW1+d16	MOVEA RW1,@RW1	MOVEA RW1 ,@RW1+d16	MOVEA RW2,@RW1	MOVEA RW2 ,@RW1+d16	MOVEA RW3,@RW1	MOVEA RW3 ,@RW1+d16	MOVEA RW4,@RW1	MOVEA RW4 ,@RW1+d16	MOVEA RW5,@RW1	MOVEA RW5 ,@RW1+d16	MOVEA RW6,@RW1	MOVEA RW6 ,@RW1+d16	MOVEA RW7,@RW1	MOVEA RW7 ,@RW1+d16
+ A	MOVEA RW0,@RW2	MOVEA RW0 ,@RW2+d16	MOVEA RW1,@RW2	MOVEA RW1 ,@RW2+d16	MOVEA RW2,@RW2	MOVEA RW2 ,@RW2+d16	MOVEA RW3,@RW2	MOVEA RW3 ,@RW2+d16	MOVEA RW4,@RW2	MOVEA RW4 ,@RW2+d16	MOVEA RW5,@RW2	MOVEA RW5 ,@RW2+d16	MOVEA RW6,@RW2	MOVEA RW6 ,@RW2+d16	MOVEA RW7,@RW2	MOVEA RW7 ,@RW2+d16
+ B	MOVEA RW0,@RW3	MOVEA RW0 ,@RW3+d16	MOVEA RW1,@RW3	MOVEA RW1 ,@RW3+d16	MOVEA RW2,@RW3	MOVEA RW2 ,@RW3+d16	MOVEA RW3,@RW3	MOVEA RW3 ,@RW3+d16	MOVEA RW4,@RW3	MOVEA RW4 ,@RW3+d16	MOVEA RW5,@RW3	MOVEA RW5 ,@RW3+d16	MOVEA RW6,@RW3	MOVEA RW6 ,@RW3+d16	MOVEA RW7,@RW3	MOVEA RW7 ,@RW3+d16
+ C	MOVEA R W0,@RW0+	MOVEA RW0 ,@RW0+RW7	MOVEA R W1,@RW0+	MOVEA RW1 ,@RW0+RW7	MOVEA R W2,@RW0+	MOVEA RW2 ,@RW0+RW7	MOVEA R W3,@RW0+	MOVEA RW3 ,@RW0+RW7	MOVEA R W4,@RW0+	MOVEA RW4 ,@RW0+RW7	MOVEA R W5,@RW0+	MOVEA RW5 ,@RW0+RW7	MOVEA R W6,@RW0+	MOVEA RW6 ,@RW0+RW7	MOVEA R W7,@RW0+	MOVEA RW7 ,@RW0+RW7
+ D	MOVEA R W0,@RW1+	MOVEA RW0 ,@RW1+RW7	MOVEA R W1,@RW1+	MOVEA RW1 ,@RW1+RW7	MOVEA R W2,@RW1+	MOVEA RW2 ,@RW1+RW7	MOVEA R W3,@RW1+	MOVEA RW3 ,@RW1+RW7	MOVEA R W4,@RW1+	MOVEA RW4 ,@RW1+RW7	MOVEA R W5,@RW1+	MOVEA RW5 ,@RW1+RW7	MOVEA R W6,@RW1+	MOVEA RW6 ,@RW1+RW7	MOVEA R W7,@RW1+	MOVEA RW7 ,@RW1+RW7
+ E	MOVEA R W0,@RW2+	MOVEA RW0 ,@PC+d16	MOVEA R W1,@RW2+	MOVEA RW1 ,@PC+d16	MOVEA R W2,@RW2+	MOVEA RW2 ,@PC+d16	MOVEA R W3,@RW2+	MOVEA RW3 ,@PC+d16	MOVEA R W4,@RW2+	MOVEA RW4 ,@PC+d16	MOVEA R W5,@RW2+	MOVEA RW5 ,@PC+d16	MOVEA R W6,@RW2+	MOVEA RW6 ,@PC+d16	MOVEA R W7,@RW2+	MOVEA RW7 ,@PC+d16
+ F	MOVEA R W0,@RW3+	MOVEA RW0 ,addr16	MOVEA R W1,@RW3+	MOVEA RW1 ,addr16	MOVEA R W2,@RW3+	MOVEA RW2 ,addr16	MOVEA R W3,@RW3+	MOVEA RW3 ,addr16	MOVEA R W4,@RW3+	MOVEA RW4 ,addr16	MOVEA R W5,@RW3+	MOVEA RW5 ,addr16	MOVEA R W6,@RW3+	MOVEA RW6 ,addr16	MOVEA R W7,@RW3+	MOVEA RW7 ,addr16

Table B.9-16 MOV Ri, ea Instruction (first byte = $7A_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV R0, R0,R0	MOV R0, @RW0+d8	MOV R1,R0	MOV R1, @RW0+d8	MOV R2,R0	MOV R2, @RW0+d8	MOV R3,R0	MOV R3, @RW0+d8	MOV R4,R0	MOV R4, @RW0+d8	MOV R5,R0	MOV R5, @RW0+d8	MOV R6,R0	MOV R6, @RW0+d8	MOV R7,R0	MOV R7, @RW0+d8
+ 1	MOV R0,R1	MOV R0, @RW1+d8	MOV R1,R1	MOV R1, @RW1+d8	MOV R2,R1	MOV R2, @RW1+d8	MOV R3,R1	MOV R3, @RW1+d8	MOV R4,R1	MOV R4, @RW1+d8	MOV R5,R1	MOV R5, @RW1+d8	MOV R6,R1	MOV R6, @RW1+d8	MOV R7,R1	MOV R7, @RW1+d8
+ 2	MOV R0,R2	MOV R0, @RW2+d8	MOV R1,R2	MOV R1, @RW2+d8	MOV R2,R2	MOV R2, @RW2+d8	MOV R3,R2	MOV R3, @RW2+d8	MOV R4,R2	MOV R4, @RW2+d8	MOV R5,R2	MOV R5, @RW2+d8	MOV R6,R2	MOV R6, @RW2+d8	MOV R7,R2	MOV R7, @RW2+d8
+ 3	MOV R0,R3	MOV R0, @RW3+d8	MOV R1,R3	MOV R1, @RW3+d8	MOV R2,R3	MOV R2, @RW3+d8	MOV R3,R3	MOV R3, @RW3+d8	MOV R4,R3	MOV R4, @RW3+d8	MOV R5,R3	MOV R5, @RW3+d8	MOV R6,R3	MOV R6, @RW3+d8	MOV R7,R3	MOV R7, @RW3+d8
+ 4	MOV R0,R4	MOV R0, @RW4+d8	MOV R1,R4	MOV R1, @RW4+d8	MOV R2,R4	MOV R2, @RW4+d8	MOV R3,R4	MOV R3, @RW4+d8	MOV R4,R4	MOV R4, @RW4+d8	MOV R5,R4	MOV R5, @RW4+d8	MOV R6,R4	MOV R6, @RW4+d8	MOV R7,R4	MOV R7, @RW4+d8
+ 5	MOV R0,R5	MOV R0, @RW5+d8	MOV R1,R5	MOV R1, @RW5+d8	MOV R2,R5	MOV R2, @RW5+d8	MOV R3,R5	MOV R3, @RW5+d8	MOV R4,R5	MOV R4, @RW5+d8	MOV R5,R5	MOV R5, @RW5+d8	MOV R6,R5	MOV R6, @RW5+d8	MOV R7,R5	MOV R7, @RW5+d8
+ 6	MOV R0,R6	MOV R0, @RW6+d8	MOV R1,R6	MOV R1, @RW6+d8	MOV R2,R6	MOV R2, @RW6+d8	MOV R3,R6	MOV R3, @RW6+d8	MOV R4,R6	MOV R4, @RW6+d8	MOV R5,R6	MOV R5, @RW6+d8	MOV R6,R6	MOV R6, @RW6+d8	MOV R7,R6	MOV R7, @RW6+d8
+ 7	MOV R0,R7	MOV R0, @RW7+d8	MOV R1,R7	MOV R1, @RW7+d8	MOV R2,R7	MOV R2, @RW7+d8	MOV R3,R7	MOV R3, @RW7+d8	MOV R4,R7	MOV R4, @RW7+d8	MOV R5,R7	MOV R5, @RW7+d8	MOV R6,R7	MOV R6, @RW7+d8	MOV R7,R7	MOV R7, @RW7+d8
+ 8	MOV R0,@RW0	MOV R0, @RW0+d16	MOV R1,@RW0	MOV R1, @RW0+d16	MOV R2,@RW0	MOV R2, @RW0+d16	MOV R3,@RW0	MOV R3, @RW0+d16	MOV R4,@RW0	MOV R4, @RW0+d16	MOV R5,@RW0	MOV R5, @RW0+d16	MOV R6,@RW0	MOV R6, @RW0+d16	MOV R7,@RW0	MOV R7, @RW0+d16
+ 9	MOV R0,@RW1	MOV R0, @RW1+d16	MOV R1,@RW1	MOV R1, @RW1+d16	MOV R2,@RW1	MOV R2, @RW1+d16	MOV R3,@RW1	MOV R3, @RW1+d16	MOV R4,@RW1	MOV R4, @RW1+d16	MOV R5,@RW1	MOV R5, @RW1+d16	MOV R6,@RW1	MOV R6, @RW1+d16	MOV R7,@RW1	MOV R7, @RW1+d16
+ A	MOV R0,@RW2	MOV R0, @RW2+d16	MOV R1,@RW2	MOV R1, @RW2+d16	MOV R2,@RW2	MOV R2, @RW2+d16	MOV R3,@RW2	MOV R3, @RW2+d16	MOV R4,@RW2	MOV R4, @RW2+d16	MOV R5,@RW2	MOV R5, @RW2+d16	MOV R6,@RW2	MOV R6, @RW2+d16	MOV R7,@RW2	MOV R7, @RW2+d16
+ B	MOV R0,@RW3	MOV R0, @RW3+d16	MOV R1,@RW3	MOV R1, @RW3+d16	MOV R2,@RW3	MOV R2, @RW3+d16	MOV R3,@RW3	MOV R3, @RW3+d16	MOV R4,@RW3	MOV R4, @RW3+d16	MOV R5,@RW3	MOV R5, @RW3+d16	MOV R6,@RW3	MOV R6, @RW3+d16	MOV R7,@RW3	MOV R7, @RW3+d16
+ C	MOV R0, @RW0+	MOV R0, @RW0+RW7	MOV R1, @RW0+	MOV R1, @RW0+RW7	MOV R2, @RW0+	MOV R2, @RW0+RW7	MOV R3, @RW0+	MOV R3, @RW0+RW7	MOV R4, @RW0+	MOV R4, @RW0+RW7	MOV R5, @RW0+	MOV R5, @RW0+RW7	MOV R6, @RW0+	MOV R6, @RW0+RW7	MOV R7, @RW0+	MOV R7, @RW0+RW7
+ D	MOV R0, @RW1+	MOV R0, @RW1+RW7	MOV R1, @RW1+	MOV R1, @RW1+RW7	MOV R2, @RW1+	MOV R2, @RW1+RW7	MOV R3, @RW1+	MOV R3, @RW1+RW7	MOV R4, @RW1+	MOV R4, @RW1+RW7	MOV R5, @RW1+	MOV R5, @RW1+RW7	MOV R6, @RW1+	MOV R6, @RW1+RW7	MOV R7, @RW1+	MOV R7, @RW1+RW7
+ E	MOV R0, @RW2+	MOV R0, @PC+d16	MOV R1, @RW2+	MOV R1, @PC+d16	MOV R2, @RW2+	MOV R2, @PC+d16	MOV R3, @RW2+	MOV R3, @PC+d16	MOV R4, @RW2+	MOV R4, @PC+d16	MOV R5, @RW2+	MOV R5, @PC+d16	MOV R6, @RW2+	MOV R6, @PC+d16	MOV R7, @RW2+	MOV R7, @PC+d16
+ F	MOV R0, @RW3+	MOV R0, addr16	MOV R1, @RW3+	MOV R1, addr16	MOV R2, @RW3+	MOV R2, addr16	MOV R3, @RW3+	MOV R3, addr16	MOV R4, @RW3+	MOV R4, addr16	MOV R5, @RW3+	MOV R5, addr16	MOV R6, @RW3+	MOV R6, addr16	MOV R7, @RW3+	MOV R7, addr16

Table B.9-17 MOVW RWi, ea Instruction (first byte = $7B_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVW RW0,RW0	MOVW RW0,@RW0+d8	MOVW RW1,RW0	MOVW RW1,@RW0+d8	MOVW RW2,RW0	MOVW RW2,@RW0+d8	MOVW RW3,RW0	MOVW RW3,@RW0+d8	MOVW RW4,RW0	MOVW RW4,@RW0+d8	MOVW RW5,RW0	MOVW RW5,@RW0+d8	MOVW RW6,RW0	MOVW RW6,@RW0+d8	MOVW RW7,RW0	MOVW RW7,@RW0+d8
+ 1	MOVW RW0,RW1	MOVW RW0,@RW1+d8	MOVW RW1,RW1	MOVW RW1,@RW1+d8	MOVW RW2,RW1	MOVW RW2,@RW1+d8	MOVW RW3,RW1	MOVW RW3,@RW1+d8	MOVW RW4,RW1	MOVW RW4,@RW1+d8	MOVW RW5,RW1	MOVW RW5,@RW1+d8	MOVW RW6,RW1	MOVW RW6,@RW1+d8	MOVW RW7,RW1	MOVW RW7,@RW1+d8
+ 2	MOVW RW0,RW2	MOVW RW0,@RW2+d8	MOVW RW1,RW2	MOVW RW1,@RW2+d8	MOVW RW2,RW2	MOVW RW2,@RW2+d8	MOVW RW3,RW2	MOVW RW3,@RW2+d8	MOVW RW4,RW2	MOVW RW4,@RW2+d8	MOVW RW5,RW2	MOVW RW5,@RW2+d8	MOVW RW6,RW2	MOVW RW6,@RW2+d8	MOVW RW7,RW2	MOVW RW7,@RW2+d8
+ 3	MOVW RW0,RW3	MOVW RW0,@RW3+d8	MOVW RW1,RW3	MOVW RW1,@RW3+d8	MOVW RW2,RW3	MOVW RW2,@RW3+d8	MOVW RW3,RW3	MOVW RW3,@RW3+d8	MOVW RW4,RW3	MOVW RW4,@RW3+d8	MOVW RW5,RW3	MOVW RW5,@RW3+d8	MOVW RW6,RW3	MOVW RW6,@RW3+d8	MOVW RW7,RW3	MOVW RW7,@RW3+d8
+ 4	MOVW RW0,RW4	MOVW RW0,@RW4+d8	MOVW RW1,RW4	MOVW RW1,@RW4+d8	MOVW RW2,RW4	MOVW RW2,@RW4+d8	MOVW RW3,RW4	MOVW RW3,@RW4+d8	MOVW RW4,RW4	MOVW RW4,@RW4+d8	MOVW RW5,RW4	MOVW RW5,@RW4+d8	MOVW RW6,RW4	MOVW RW6,@RW4+d8	MOVW RW7,RW4	MOVW RW7,@RW4+d8
+ 5	MOVW RW0,RW5	MOVW RW0,@RW5+d8	MOVW RW1,RW5	MOVW RW1,@RW5+d8	MOVW RW2,RW5	MOVW RW2,@RW5+d8	MOVW RW3,RW5	MOVW RW3,@RW5+d8	MOVW RW4,RW5	MOVW RW4,@RW5+d8	MOVW RW5,RW5	MOVW RW5,@RW5+d8	MOVW RW6,RW5	MOVW RW6,@RW5+d8	MOVW RW7,RW5	MOVW RW7,@RW5+d8
+ 6	MOVW RW0,RW6	MOVW RW0,@RW6+d8	MOVW RW1,RW6	MOVW RW1,@RW6+d8	MOVW RW2,RW6	MOVW RW2,@RW6+d8	MOVW RW3,RW6	MOVW RW3,@RW6+d8	MOVW RW4,RW6	MOVW RW4,@RW6+d8	MOVW RW5,RW6	MOVW RW5,@RW6+d8	MOVW RW6,RW6	MOVW RW6,@RW6+d8	MOVW RW7,RW6	MOVW RW7,@RW6+d8
+ 7	MOVW RW0,RW7	MOVW RW0,@RW7+d8	MOVW RW1,RW7	MOVW RW1,@RW7+d8	MOVW RW2,RW7	MOVW RW2,@RW7+d8	MOVW RW3,RW7	MOVW RW3,@RW7+d8	MOVW RW4,RW7	MOVW RW4,@RW7+d8	MOVW RW5,RW7	MOVW RW5,@RW7+d8	MOVW RW6,RW7	MOVW RW6,@RW7+d8	MOVW RW7,RW7	MOVW RW7,@RW7+d8
+ 8	MOVW RW0,@RW0	MOVW RW0,@RW0+d16	MOVW RW1,@RW0	MOVW RW1,@RW0+d16	MOVW RW2,@RW0	MOVW RW2,@RW0+d16	MOVW RW3,@RW0	MOVW RW3,@RW0+d16	MOVW RW4,@RW0	MOVW RW4,@RW0+d16	MOVW RW5,@RW0	MOVW RW5,@RW0+d16	MOVW RW6,@RW0	MOVW RW6,@RW0+d16	MOVW RW7,@RW0	MOVW RW7,@RW0+d16
+ 9	MOVW RW0,@RW1	MOVW RW0,@RW1+d16	MOVW RW1,@RW1	MOVW RW1,@RW1+d16	MOVW RW2,@RW1	MOVW RW2,@RW1+d16	MOVW RW3,@RW1	MOVW RW3,@RW1+d16	MOVW RW4,@RW1	MOVW RW4,@RW1+d16	MOVW RW5,@RW1	MOVW RW5,@RW1+d16	MOVW RW6,@RW1	MOVW RW6,@RW1+d16	MOVW RW7,@RW1	MOVW RW7,@RW1+d16
+ A	MOVW RW0,@RW2	MOVW RW0,@RW2+d16	MOVW RW1,@RW2	MOVW RW1,@RW2+d16	MOVW RW2,@RW2	MOVW RW2,@RW2+d16	MOVW RW3,@RW2	MOVW RW3,@RW2+d16	MOVW RW4,@RW2	MOVW RW4,@RW2+d16	MOVW RW5,@RW2	MOVW RW5,@RW2+d16	MOVW RW6,@RW2	MOVW RW6,@RW2+d16	MOVW RW7,@RW2	MOVW RW7,@RW2+d16
+ B	MOVW RW0,@RW3	MOVW RW0,@RW3+d16	MOVW RW1,@RW3	MOVW RW1,@RW3+d16	MOVW RW2,@RW3	MOVW RW2,@RW3+d16	MOVW RW3,@RW3	MOVW RW3,@RW3+d16	MOVW RW4,@RW3	MOVW RW4,@RW3+d16	MOVW RW5,@RW3	MOVW RW5,@RW3+d16	MOVW RW6,@RW3	MOVW RW6,@RW3+d16	MOVW RW7,@RW3	MOVW RW7,@RW3+d16
+ C	MOVW R0,@RW0+	MOVW RW0,@RW0+RW7	MOVW R1,@RW0+	MOVW RW1,@RW0+RW7	MOVW R2,@RW0+	MOVW RW2,@RW0+RW7	MOVW R3,@RW0+	MOVW RW3,@RW0+RW7	MOVW R4,@RW0+	MOVW RW4,@RW0+RW7	MOVW R5,@RW0+	MOVW RW5,@RW0+RW7	MOVW R6,@RW0+	MOVW RW6,@RW0+RW7	MOVW R7,@RW0+	MOVW RW7,@RW0+RW7
+ D	MOVW R0,@RW1+	MOVW RW0,@RW1+RW7	MOVW R1,@RW1+	MOVW RW1,@RW1+RW7	MOVW R2,@RW1+	MOVW RW2,@RW1+RW7	MOVW R3,@RW1+	MOVW RW3,@RW1+RW7	MOVW R4,@RW1+	MOVW RW4,@RW1+RW7	MOVW R5,@RW1+	MOVW RW5,@RW1+RW7	MOVW R6,@RW1+	MOVW RW6,@RW1+RW7	MOVW R7,@RW1+	MOVW RW7,@RW1+RW7
+ E	MOVW R0,@RW2+	MOVW RW0,@PC+d16	MOVW R1,@RW2+	MOVW RW1,@PC+d16	MOVW R2,@RW2+	MOVW RW2,@PC+d16	MOVW R3,@RW2+	MOVW RW3,@PC+d16	MOVW R4,@RW2+	MOVW RW4,@PC+d16	MOVW R5,@RW2+	MOVW RW5,@PC+d16	MOVW R6,@RW2+	MOVW RW6,@PC+d16	MOVW R7,@RW2+	MOVW RW7,@PC+d16
+ F	MOVW R0,@RW3+	MOVW RW0,0,addr16	MOVW R1,@RW3+	MOVW RW1,1,addr16	MOVW R2,@RW3+	MOVW RW2,2,addr16	MOVW R3,@RW3+	MOVW RW3,3,addr16	MOVW R4,@RW3+	MOVW RW4,4,addr16	MOVW R5,@RW3+	MOVW RW5,5,addr16	MOVW R6,@RW3+	MOVW RW6,6,addr16	MOVW R7,@RW3+	MOVW RW7,7,addr16

Table B.9-18 MOV ea, Ri Instruction (first byte = $7C_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOV R0,R0	MOV @R W0+d8,R0	MOV R0,R1	MOV @R W0+d8,R1	MOV R0,R2	MOV @R W0+d8,R2	MOV R0,R3	MOV @R W0+d8,R3	MOV R0,R4	MOV @R W0+d8,R4	MOV R0,R5	MOV @R W0+d8,R5	MOV R0,R6	MOV @R W0+d8,R6	MOV R0,R7	MOV @R W0+d8,R7
+ 1	MOV R1,R0	MOV @R W1+d8,R0	MOV R1,R1	MOV @R W1+d8,R1	MOV R1,R2	MOV @R W1+d8,R2	MOV R1,R3	MOV @R W1+d8,R3	MOV R1,R4	MOV @R W1+d8,R4	MOV R1,R5	MOV @R W1+d8,R5	MOV R1,R6	MOV @R W1+d8,R6	MOV R1,R7	MOV @R W1+d8,R7
+ 2	MOV R2,R0	MOV @R W2+d8,R0	MOV R2,R1	MOV @R W2+d8,R1	MOV R2,R2	MOV @R W2+d8,R2	MOV R2,R3	MOV @R W2+d8,R3	MOV R2,R4	MOV @R W2+d8,R4	MOV R2,R5	MOV @R W2+d8,R5	MOV R2,R6	MOV @R W2+d8,R6	MOV R2,R7	MOV @R W2+d8,R7
+ 3	MOV R3,R0	MOV @R W3+d8,R0	MOV R3,R1	MOV @R W3+d8,R1	MOV R3,R2	MOV @R W3+d8,R2	MOV R3,R3	MOV @R W3+d8,R3	MOV R3,R4	MOV @R W3+d8,R4	MOV R3,R5	MOV @R W3+d8,R5	MOV R3,R6	MOV @R W3+d8,R6	MOV R3,R7	MOV @R W3+d8,R7
+ 4	MOV R4,R0	MOV @R W4+d8,R0	MOV R4,R1	MOV @R W4+d8,R1	MOV R4,R2	MOV @R W4+d8,R2	MOV R4,R3	MOV @R W4+d8,R3	MOV R4,R4	MOV @R W4+d8,R4	MOV R4,R5	MOV @R W4+d8,R5	MOV R4,R6	MOV @R W4+d8,R6	MOV R4,R7	MOV @R W4+d8,R7
+ 5	MOV R5,R0	MOV @R W5+d8,R0	MOV R5,R1	MOV @R W5+d8,R1	MOV R5,R2	MOV @R W5+d8,R2	MOV R5,R3	MOV @R W5+d8,R3	MOV R5,R4	MOV @R W5+d8,R4	MOV R5,R5	MOV @R W5+d8,R5	MOV R5,R6	MOV @R W5+d8,R6	MOV R5,R7	MOV @R W5+d8,R7
+ 6	MOV R6,R0	MOV @R W6+d8,R0	MOV R6,R1	MOV @R W6+d8,R1	MOV R6,R2	MOV @R W6+d8,R2	MOV R6,R3	MOV @R W6+d8,R3	MOV R6,R4	MOV @R W6+d8,R4	MOV R6,R5	MOV @R W6+d8,R5	MOV R6,R6	MOV @R W6+d8,R6	MOV R6,R7	MOV @R W6+d8,R7
+ 7	MOV R7,R0	MOV @R W7+d8,R0	MOV R7,R1	MOV @R W7+d8,R1	MOV R7,R2	MOV @R W7+d8,R2	MOV R7,R3	MOV @R W7+d8,R3	MOV R7,R4	MOV @R W7+d8,R4	MOV R7,R5	MOV @R W7+d8,R5	MOV R7,R6	MOV @R W7+d8,R6	MOV R7,R7	MOV @R W7+d8,R7
+ 8	MOV @RW0,R0	MOV @RW 0+d16,R0	MOV @RW0,R1	MOV @RW 0+d16,R1	MOV @RW0,R2	MOV @RW 0+d16,R2	MOV @RW0,R3	MOV @RW 0+d16,R3	MOV @RW0,R4	MOV @RW 0+d16,R4	MOV @RW0,R5	MOV @RW 0+d16,R5	MOV @RW0,R6	MOV @RW 0+d16,R6	MOV @RW0,R7	MOV @RW 0+d16,R7
+ 9	MOV @RW1,R0	MOV @RW 1+d16,R0	MOV @RW1,R1	MOV @RW 1+d16,R1	MOV @RW1,R2	MOV @RW 1+d16,R2	MOV @RW1,R3	MOV @RW 1+d16,R3	MOV @RW1,R4	MOV @RW 1+d16,R4	MOV @RW1,R5	MOV @RW 1+d16,R5	MOV @RW1,R6	MOV @RW 1+d16,R6	MOV @RW1,R7	MOV @RW 1+d16,R7
+ A	MOV @RW2,R0	MOV @RW 2+d16,R0	MOV @RW2,R1	MOV @RW 2+d16,R1	MOV @RW2,R2	MOV @RW 2+d16,R2	MOV @RW2,R3	MOV @RW 2+d16,R3	MOV @RW2,R4	MOV @RW 2+d16,R4	MOV @RW2,R5	MOV @RW 2+d16,R5	MOV @RW2,R6	MOV @RW 2+d16,R6	MOV @RW2,R7	MOV @RW 2+d16,R7
+ B	MOV @RW3,R0	MOV @RW 3+d16,R0	MOV @RW3,R1	MOV @RW 3+d16,R1	MOV @RW3,R2	MOV @RW 3+d16,R2	MOV @RW3,R3	MOV @RW 3+d16,R3	MOV @RW3,R4	MOV @RW 3+d16,R4	MOV @RW3,R5	MOV @RW 3+d16,R5	MOV @RW3,R6	MOV @RW 3+d16,R6	MOV @RW3,R7	MOV @RW 3+d16,R7
+ C	MOV @RW0+,R0	MOV @RW 0+RW7,R0	MOV @RW0+,R1	MOV @RW 0+RW7,R1	MOV @RW0+,R2	MOV @RW 0+RW7,R2	MOV @RW0+,R3	MOV @RW 0+RW7,R3	MOV @RW0+,R4	MOV @RW 0+RW7,R4	MOV @RW0+,R5	MOV @RW 0+RW7,R5	MOV @RW0+,R6	MOV @RW 0+RW7,R6	MOV @RW0+,R7	MOV @RW 0+RW7,R7
+ D	MOV @RW1+,R0	MOV @RW 1+RW7,R0	MOV @RW1+,R1	MOV @RW 1+RW7,R1	MOV @RW1+,R2	MOV @RW 1+RW7,R2	MOV @RW1+,R3	MOV @RW 1+RW7,R3	MOV @RW1+,R4	MOV @RW 1+RW7,R4	MOV @RW1+,R5	MOV @RW 1+RW7,R5	MOV @RW1+,R6	MOV @RW 1+RW7,R6	MOV @RW1+,R7	MOV @RW 1+RW7,R7
+ E	MOV @RW2+,R0	MOV P C+d16,R0	MOV @RW2+,R1	MOV P C+d16,R1	MOV @RW2+,R2	MOV P C+d16,R2	MOV @RW2+,R3	MOV P C+d16,R3	MOV @RW2+,R4	MOV P C+d16,R4	MOV @RW2+,R5	MOV P C+d16,R5	MOV @RW2+,R6	MOV P C+d16,R6	MOV @RW2+,R7	MOV P C+d16,R7
+ F	MOV @RW3+,R0	MOV a ddr16,R0	MOV @RW3+,R1	MOV a ddr16,R1	MOV @RW3+,R2	MOV a ddr16,R2	MOV @RW3+,R3	MOV a ddr16,R3	MOV @RW3+,R4	MOV a ddr16,R4	MOV @RW3+,R5	MOV a ddr16,R5	MOV @RW3+,R6	MOV a ddr16,R6	MOV @RW3+,R7	MOV a ddr16,R7

Table B.9-19 MOVW ea, Rwi Instruction (first byte = $7D_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	MOVW RW0,RW0	MOVW @RW 0+d8,RW0	MOVW RW0,RW1	MOVW @RW 0+d8,RW1	MOVW RW0,RW2	MOVW @RW 0+d8,RW2	MOVW RW0,RW3	MOVW @RW 0+d8,RW3	MOVW RW0,RW4	MOVW @RW 0+d8,RW4	MOVW RW0,RW5	MOVW @RW 0+d8,RW5	MOVW RW0,RW6	MOVW @RW 0+d8,RW6	MOVW RW0,RW7	MOVW @RW 0+d8,RW7
+ 1	MOVW RW1,RW0	MOVW @RW 1+d8,RW0	MOVW RW1,RW1	MOVW @RW 1+d8,RW1	MOVW RW1,RW2	MOVW @RW 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	WOVW RW7,RW4	MOVW @RW 7+d8,RW4	WOVW RW7,RW5	MOVW @RW 7+d8,RW5	WOVW RW7,RW6	MOVW @RW 7+d8,RW6	WOVW RW7,RW7	MOVW @RW 7+d8,RW7
+ 8	MOVW @RW0,RW0	MOVW @RW0 @16,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 @16,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 @16,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 @16,RW0	MOVW @RW3,RW1	MOVW @RW3 +d16,RW1	MOVW @RW3,RW2	MOVW @RW3 +d16,RW2	MOVW @RW3,RW3	MOVW @RW3 +d16,RW3	MOVW @RW3,RW4	MOVW @RW3 +d16,RW4	MOVW @RW3,RW5	MOVW @RW3 +d16,RW5	MOVW @RW3,RW6	MOVW @RW3 +d16,RW6	MOVW @RW3,RW7	MOVW @RW3 +d16,RW7
+ C	MOVW @ RW0+,RW0	MOVW @RW0 +RW7,RW0	MOVW @ RW0+,RW1	MOVW @RW0 +RW7,RW1	MOVW @ RW0+,RW2	MOVW @RW0 +RW7,RW2	MOVW @ RW0+,RW3	MOVW @RW0 +RW7,RW3	MOVW @ RW0+,RW4	MOVW @RW0 +RW7,RW4	MOVW @ RW0+,RW5	MOVW @RW0 +RW7,RW5	MOVW @ RW0+,RW6	MOVW @RW0 +RW7,RW6	MOVW @ RW0+,RW7	MOVW @RW0 +RW7,RW7
+ D	MOVW @ RW1+,RW0	MOVW @RW1 +RW7,RW0	MOVW @ RW1+,RW1	MOVW @RW1 +RW7,RW1	MOVW @ RW1+,RW2	MOVW @RW1 +RW7,RW2	MOVW @ RW1+,RW3	MOVW @RW1 +RW7,RW3	MOVW @ RW1+,RW4	MOVW @RW1 +RW7,RW4	MOVW @ RW1+,RW5	MOVW @RW1 +RW7,RW5	MOVW @ RW1+,RW6	MOVW @RW1 +RW7,RW6	MOVW @ RW1+,RW7	MOVW @RW1 +RW7,RW7
+ E	MOVW @ RW2+,RW0	MOVW @PC+ d16,RW0	MOVW @ RW2+,RW1	MOVW @PC+ d16,RW1	MOVW @ RW2+,RW2	MOVW @PC+ d16,RW2	MOVW @ RW2+,RW3	MOVW @PC+ d16,RW3	MOVW @ RW2+,RW4	MOVW @PC+ d16,RW4	MOVW @ RW2+,RW5	MOVW @PC+ d16,RW5	MOVW @ RW2+,RW6	MOVW @PC+ d16,RW6	MOVW @ RW2+,RW7	MOVW @PC+ d16,RW7
+ F	MOVW @ RW3+,RW0	MOVW addr 16,RW0	MOVW @ RW3+,RW1	MOVW addr 16,RW1	MOVW @ RW3+,RW2	MOVW addr 16,RW2	MOVW @ RW3+,RW3	MOVW addr 16,RW3	MOVW @ RW3+,RW4	MOVW addr 16,RW4	MOVW @ RW3+,RW5	MOVW addr 16,RW5	MOVW @ RW3+,RW6	MOVW addr 16,RW6	MOVW @ RW3+,RW7	MOVW addr 16,RW7

Table B.9-20 XCH Ri, ea Instruction (first byte = $7E_H$)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	XCH R0, R0,R0	XCH R0, @RW0+d8	XCH R1, R1,R0	XCH R1, @RW0+d8	XCH R2, R2,R0	XCH R2, @RW0+d8	XCH R3, R3,R0	XCH R3, @RW0+d8	XCH R4, R4,R0	XCH R4, @RW0+d8	XCH R5, R5,R0	XCH R5, @RW0+d8	XCH R6, R6,R0	XCH R6, @RW0+d8	XCH R7, R7,R0	XCH R7, @RW0+d8
+ 1	XCH R0, R0,R1	XCH R0, @RW1+d8	XCH R1, R1,R1	XCH R1, @RW1+d8	XCH R2, R2,R1	XCH R2, @RW1+d8	XCH R3, R3,R1	XCH R3, @RW1+d8	XCH R4, R4,R1	XCH R4, @RW1+d8	XCH R5, R5,R1	XCH R5, @RW1+d8	XCH R6, R6,R1	XCH R6, @RW1+d8	XCH R7, R7,R1	XCH R7, @RW1+d8
+ 2	XCH R0, R0,R2	XCH R0, @RW2+d8	XCH R1, R1,R2	XCH R1, @RW2+d8	XCH R2, R2,R2	XCH R2, @RW2+d8	XCH R3, R3,R2	XCH R3, @RW2+d8	XCH R4, R4,R2	XCH R4, @RW2+d8	XCH R5, R5,R2	XCH R5, @RW2+d8	XCH R6, R6,R2	XCH R6, @RW2+d8	XCH R7, R7,R2	XCH R7, @RW2+d8
+ 3	XCH R0, R0,R3	XCH R0, @RW3+d8	XCH R1, R1,R3	XCH R1, @RW3+d8	XCH R2, R2,R3	XCH R2, @RW3+d8	XCH R3, R3,R3	XCH R3, @RW3+d8	XCH R4, R4,R3	XCH R4, @RW3+d8	XCH R5, R5,R3	XCH R5, @RW3+d8	XCH R6, R6,R3	XCH R6, @RW3+d8	XCH R7, R7,R3	XCH R7, @RW3+d8
+ 4	XCH R0, R0,R4	XCH R0, @RW4+d8	XCH R1, R1,R4	XCH R1, @RW4+d8	XCH R2, R2,R4	XCH R2, @RW4+d8	XCH R3, R3,R4	XCH R3, @RW4+d8	XCH R4, R4,R4	XCH R4, @RW4+d8	XCH R5, R5,R4	XCH R5, @RW4+d8	XCH R6, R6,R4	XCH R6, @RW4+d8	XCH R7, R7,R4	XCH R7, @RW4+d8
+ 5	XCH R0, R0,R5	XCH R0, @RW5+d8	XCH R1, R1,R5	XCH R1, @RW5+d8	XCH R2, R2,R5	XCH R2, @RW5+d8	XCH R3, R3,R5	XCH R3, @RW5+d8	XCH R4, R4,R5	XCH R4, @RW5+d8	XCH R5, R5,R5	XCH R5, @RW5+d8	XCH R6, R6,R5	XCH R6, @RW5+d8	XCH R7, R7,R5	XCH R7, @RW5+d8
+ 6	XCH R0, R0,R6	XCH R0, @RW6+d8	XCH R1, R1,R6	XCH R1, @RW6+d8	XCH R2, R2,R6	XCH R2, @RW6+d8	XCH R3, R3,R6	XCH R3, @RW6+d8	XCH R4, R4,R6	XCH R4, @RW6+d8	XCH R5, R5,R6	XCH R5, @RW6+d8	XCH R6, R6,R6	XCH R6, @RW6+d8	XCH R7, R7,R6	XCH R7, @RW6+d8
+ 7	XCH R0, R0,R7	XCH R0, @RW7+d8	XCH R1, R1,R7	XCH R1, @RW7+d8	XCH R2, R2,R7	XCH R2, @RW7+d8	XCH R3, R3,R7	XCH R3, @RW7+d8	XCH R4, R4,R7	XCH R4, @RW7+d8	XCH R5, R5,R7	XCH R5, @RW7+d8	XCH R6, R6,R7	XCH R6, @RW7+d8	XCH R7, R7,R7	XCH R7, @RW7+d8
+ 8	XCH R0, R0,@RW0	XCH R0, @RW0+d16	XCH R1, R1,@RW0	XCH R1, @RW0+d16	XCH R2, R2,@RW0	XCH R2, @RW0+d16	XCH R3, R3,@RW0	XCH R3, @RW0+d16	XCH R4, R4,@RW0	XCH R4, @RW0+d16	XCH R5, R5,@RW0	XCH R5, @RW0+d16	XCH R6, R6,@RW0	XCH R6, @RW0+d16	XCH R7, R7,@RW0	XCH R7, @RW0+d16
+ 9	XCH R0, R0,@RW1	XCH R0, @RW1+d16	XCH R1, R1,@RW1	XCH R1, @RW1+d16	XCH R2, R2,@RW1	XCH R2, @RW1+d16	XCH R3, R3,@RW1	XCH R3, @RW1+d16	XCH R4, R4,@RW1	XCH R4, @RW1+d16	XCH R5, R5,@RW1	XCH R5, @RW1+d16	XCH R6, R6,@RW1	XCH R6, @RW1+d16	XCH R7, R7,@RW1	XCH R7, @RW1+d16
+ A	XCH R0, R0,@RW2	XCH R0, W2+d16,A	XCH R1, R1,@RW2	XCH R1, W2+d16,A	XCH R2, R2,@RW2	XCH R2, W2+d16,A	XCH R3, R3,@RW2	XCH R3, W2+d16,A	XCH R4, R4,@RW2	XCH R4, W2+d16,A	XCH R5, R5,@RW2	XCH R5, W2+d16,A	XCH R6, R6,@RW2	XCH R6, W2+d16,A	XCH R7, R7,@RW2	XCH R7, W2+d16,A
+ B	XCH R0, R0,@RW3	XCH R0, @RW3+d16	XCH R1, R1,@RW3	XCH R1, @RW3+d16	XCH R2, R2,@RW3	XCH R2, @RW3+d16	XCH R3, R3,@RW3	XCH R3, @RW3+d16	XCH R4, R4,@RW3	XCH R4, @RW3+d16	XCH R5, R5,@RW3	XCH R5, @RW3+d16	XCH R6, R6,@RW3	XCH R6, @RW3+d16	XCH R7, R7,@RW3	XCH R7, @RW3+d16
+ C	XCH R0, R0,@RW0+	XCH R0, @RW0+RW7	XCH R1, R1,@RW0+	XCH R1, @RW0+RW7	XCH R2, R2,@RW0+	XCH R2, @RW0+RW7	XCH R3, R3,@RW0+	XCH R3, @RW0+RW7	XCH R4, R4,@RW0+	XCH R4, @RW0+RW7	XCH R5, R5,@RW0+	XCH R5, @RW0+RW7	XCH R6, R6,@RW0+	XCH R6, @RW0+RW7	XCH R7, R7,@RW0+	XCH R7, @RW0+RW7
+ D	XCH R0, R0,@RW1+	XCH R0, @RW1+RW7	XCH R1, R1,@RW1+	XCH R1, @RW1+RW7	XCH R2, R2,@RW1+	XCH R2, @RW1+RW7	XCH R3, R3,@RW1+	XCH R3, @RW1+RW7	XCH R4, R4,@RW1+	XCH R4, @RW1+RW7	XCH R5, R5,@RW1+	XCH R5, @RW1+RW7	XCH R6, R6,@RW1+	XCH R6, @RW1+RW7	XCH R7, R7,@RW1+	XCH R7, @RW1+RW7
+ E	XCH R0, R0,@RW2+	XCH R0, @PC+d16	XCH R1, R1,@RW2+	XCH R1, @PC+d16	XCH R2, R2,@RW2+	XCH R2, @PC+d16	XCH R3, R3,@RW2+	XCH R3, @PC+d16	XCH R4, R4,@RW2+	XCH R4, @PC+d16	XCH R5, R5,@RW2+	XCH R5, @PC+d16	XCH R6, R6,@RW2+	XCH R6, @PC+d16	XCH R7, R7,@RW2+	XCH R7, @PC+d16
+ F	XCH R0, R0,@RW3+	XCH R0, addr16	XCH R1, R1,@RW3+	XCH R1, addr16	XCH R2, R2,@RW3+	XCH R2, addr16	XCH R3, R3,@RW3+	XCH R3, addr16	XCH R4, R4,@RW3+	XCH R4, addr16	XCH R5, R5,@RW3+	XCH R5, addr16	XCH R6, R6,@RW3+	XCH R6, addr16	XCH R7, R7,@RW3+	XCH R7, addr16

Table B.9-21 XCHW RWi, ea Instruction (first byte = $7F_H$)

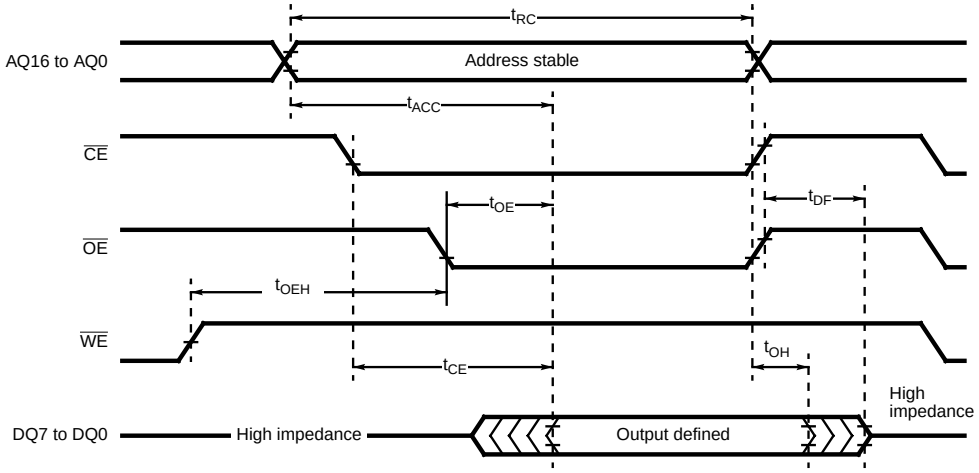
	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+ 0	XCHW RW0, RW0,RW0	XCHW RW0, @RW0+d8	XCHW RW1, RW0,RW0	XCHW RW1, @RW0+d8	XCHW RW2, RW0,RW0	XCHW RW2, @RW0+d8	XCHW RW3, RW0,RW0	XCHW RW3, @RW0+d8	XCHW RW4, RW0,RW0	XCHW RW4, @RW0+d8	XCHW R5,RW0	XCHW RW5, @RW0+d8	XCHW R6,RW0	XCHW RW6, @RW0+d8	XCHW R7,RW0	XCHW RW7, @RW0+d8
+ 1	XCHW RW0, RW0,RW1	XCHW RW0, @RW1+d8	XCHW RW1, RW0,RW1	XCHW RW1, @RW1+d8	XCHW RW2, RW0,RW1	XCHW RW2, @RW1+d8	XCHW RW3, RW0,RW1	XCHW RW3, @RW1+d8	XCHW RW4, RW0,RW1	XCHW RW4, @RW1+d8	XCHW R5,RW1	XCHW RW5, @RW1+d8	XCHW R6,RW1	XCHW RW6, @RW1+d8	XCHW R7,RW1	XCHW RW7, @RW1+d8
+ 2	XCHW RW0, RW0,RW2	XCHW RW0, @RW2+d8	XCHW RW1, RW1,RW2	XCHW RW1, @RW2+d8	XCHW RW2, RW2,RW2	XCHW RW2, @RW2+d8	XCHW RW3, RW3,RW2	XCHW RW3, @RW2+d8	XCHW RW4, RW4,RW2	XCHW RW4, @RW2+d8	XCHW R5,RW2	XCHW RW5, @RW2+d8	XCHW R6,RW2	XCHW RW6, @RW2+d8	XCHW R7,RW2	XCHW RW7, @RW2+d8
+ 3	XCHW RW0, RW0,RW3	XCHW RW0, @RW3+d8	XCHW RW1, RW1,RW3	XCHW RW1, @RW3+d8	XCHW RW2, RW2,RW3	XCHW RW2, @RW3+d8	XCHW RW3, RW3,RW3	XCHW RW3, @RW3+d8	XCHW RW4, RW4,RW3	XCHW RW4, @RW3+d8	XCHW R5,RW3	XCHW RW5, @RW3+d8	XCHW R6,RW3	XCHW RW6, @RW3+d8	XCHW R7,RW3	XCHW RW7, @RW3+d8
+ 4	XCHW RW0, RW0,RW4	XCHW RW0, @RW4+d8	XCHW RW1, RW1,RW4	XCHW RW1, @RW4+d8	XCHW RW2, RW2,RW4	XCHW RW2, @RW4+d8	XCHW RW3, RW3,RW4	XCHW RW3, @RW4+d8	XCHW RW4, RW4,RW4	XCHW RW4, @RW4+d8	XCHW R5,RW4	XCHW RW5, @RW4+d8	XCHW R6,RW4	XCHW RW6, @RW4+d8	XCHW R7,RW4	XCHW RW7, @RW4+d8
+ 5	XCHW RW0, RW0,RW5	XCHW RW0, @RW5+d8	XCHW RW1, RW1,RW5	XCHW RW1, @RW5+d8	XCHW RW2, RW2,RW5	XCHW RW2, @RW5+d8	XCHW RW3, RW3,RW5	XCHW RW3, @RW5+d8	XCHW RW4, RW4,RW5	XCHW RW4, @RW5+d8	XCHW R5,RW5	XCHW RW5, @RW5+d8	XCHW R6,RW5	XCHW RW6, @RW5+d8	XCHW R7,RW5	XCHW RW7, @RW5+d8
+ 6	XCHW RW0, RW0,RW6	XCHW RW0, @RW6+d8	XCHW RW1, RW1,RW6	XCHW RW1, @RW6+d8	XCHW RW2, RW2,RW6	XCHW RW2, @RW6+d8	XCHW RW3, RW3,RW6	XCHW RW3, @RW6+d8	XCHW RW4, RW4,RW6	XCHW RW4, @RW6+d8	XCHW R5,RW6	XCHW RW5, @RW6+d8	XCHW R6,RW6	XCHW RW6, @RW6+d8	XCHW R7,RW6	XCHW RW7, @RW6+d8
+ 7	XCHW RW0, RW0,RW7	XCHW RW0, @RW7+d8	XCHW RW1, RW1,RW7	XCHW RW1, @RW7+d8	XCHW RW2, RW2,RW7	XCHW RW2, @RW7+d8	XCHW RW3, RW3,RW7	XCHW RW3, @RW7+d8	XCHW RW4, RW4,RW7	XCHW RW4, @RW7+d8	XCHW R5,RW7	XCHW RW5, @RW7+d8	XCHW R6,RW7	XCHW RW6, @RW7+d8	XCHW R7,RW7	XCHW RW7, @RW7+d8
+ 8	XCHW RW0, RW0,R0	XCHW RW0, @RW0+d16	XCHW RW1, RW1,@RW0	XCHW RW1, @RW0+d16	XCHW RW2, RW2,@RW0	XCHW RW2, @RW0+d16	XCHW RW3, RW3,@RW0	XCHW RW3, @RW0+d16	XCHW RW4, RW4,@RW0	XCHW RW4, @RW0+d16	XCHW R5,@RW0	XCHW RW5, @RW0+d16	XCHW R6,@RW0	XCHW RW6, @RW0+d16	XCHW R7,@RW0	XCHW RW7, @RW0+d16
+ 9	XCHW RW0, RW0,@RW1	XCHW RW0, @RW1+d16	XCHW RW1, RW1,@RW1	XCHW RW1, @RW1+d16	XCHW RW2, RW2,@RW1	XCHW RW2, @RW1+d16	XCHW RW3, RW3,@RW1	XCHW RW3, @RW1+d16	XCHW RW4, RW4,@RW1	XCHW RW4, @RW1+d16	XCHW R5,@RW1	XCHW RW5, @RW1+d16	XCHW R6,@RW1	XCHW RW6, @RW1+d16	XCHW R7,@RW1	XCHW RW7, @RW1+d16
+ A	XCHW RW0, RW0,@RW2	XCHW RW0, @RW2+d16	XCHW RW1, RW1,@RW2	XCHW RW1, @RW2+d16	XCHW RW2, RW2,@RW2	XCHW RW2, @RW2+d16	XCHW RW3, RW3,@RW2	XCHW RW3, @RW2+d16	XCHW RW4, RW4,@RW2	XCHW RW4, @RW2+d16	XCHW R5,@RW2	XCHW RW5, @RW2+d16	XCHW R6,@RW2	XCHW RW6, @RW2+d16	XCHW R7,@RW2	XCHW RW7, @RW2+d16
+ B	XCHW RW0, RW0,@RW3	XCHW RW0, @RW3+d16	XCHW RW1, RW1,@RW3	XCHW RW1, @RW3+d16	XCHW RW2, RW2,@RW3	XCHW RW2, @RW3+d16	XCHW RW3, RW3,@RW3	XCHW RW3, @RW3+d16	XCHW RW4, RW4,@RW3	XCHW RW4, @RW3+d16	XCHW R5,@RW3	XCHW RW5, @RW3+d16	XCHW R6,@RW3	XCHW RW6, @RW3+d16	XCHW R7,@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 WO,@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 WO,@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 WO,@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

APPENDIX C Timing Diagrams in Flash Memory Mode

Each timing diagram for the external pins of the MB90F594 in the Flash Memory mode is shown below.

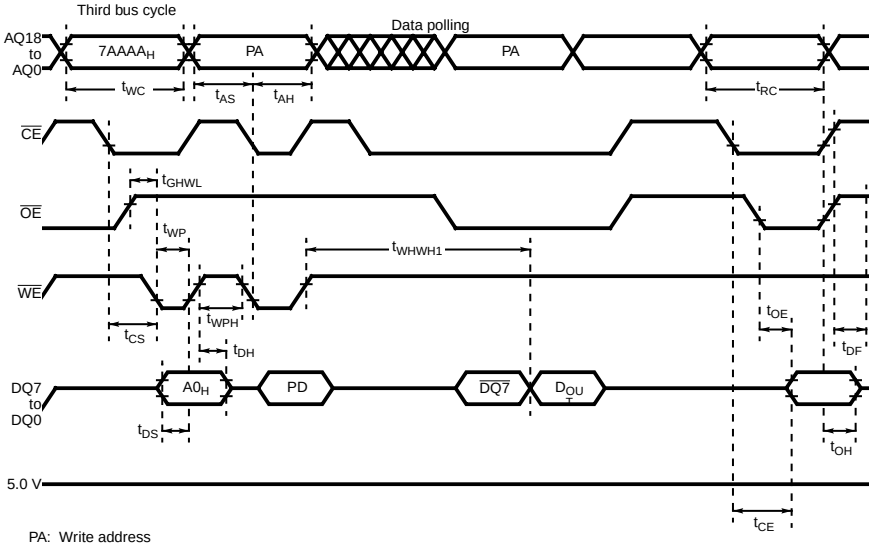
■ Data read by Read Access

Figure C-1 Timing Diagram for Read Access



■ Write, Data polling, Read ($\overline{WE}$ control)

Figure C-2 Write Data polling Read (WE control)

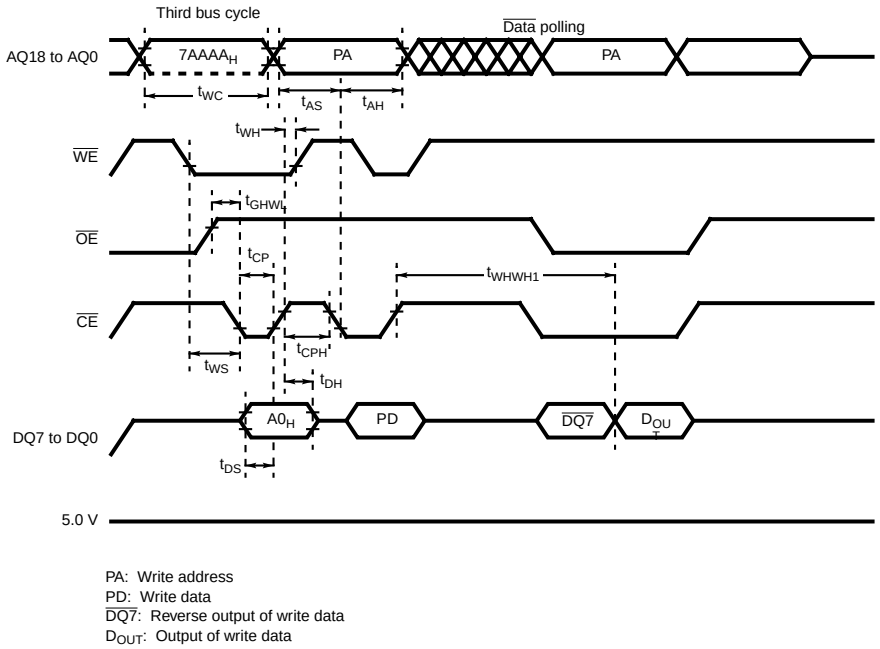


PA: Write address
PD: Write data
$\overline{DQ7}$: Reverse output of write data
D_{OUT}: Output of write data

Note: The last two bus cycle sequences out of the four are described.

■ Write Data Polling Read ($\overline{CE}$ control)

Figure C-3 Timing Diagram for Write Access ($\overline{CE}$ Control)

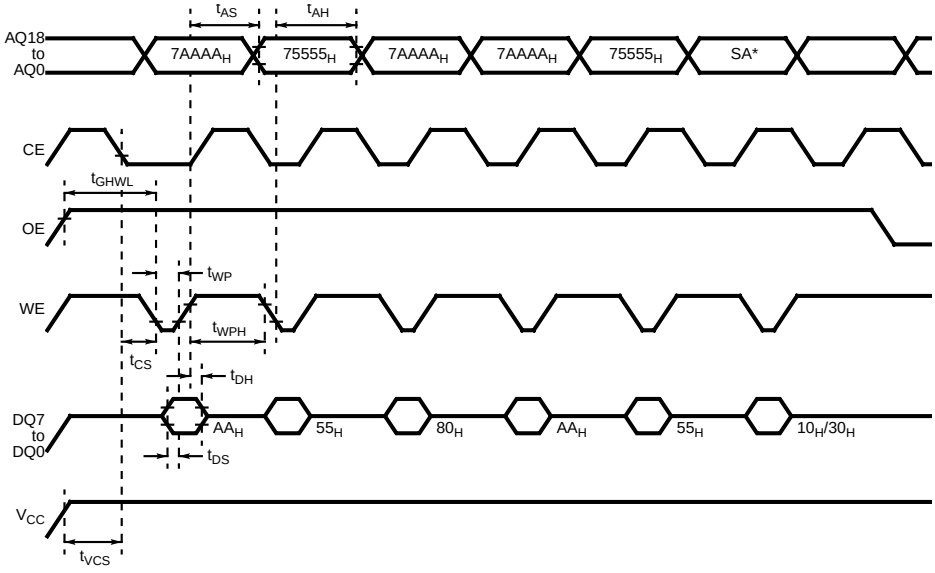


PA: Write address
PD: Write data
$\overline{DQ7}$: Reverse output of write data
D_{OUT}: Output of write data

Note: The last two bus cycle sequences out of the four are described.

■ Chip Erase/sector Erase Command Sequence

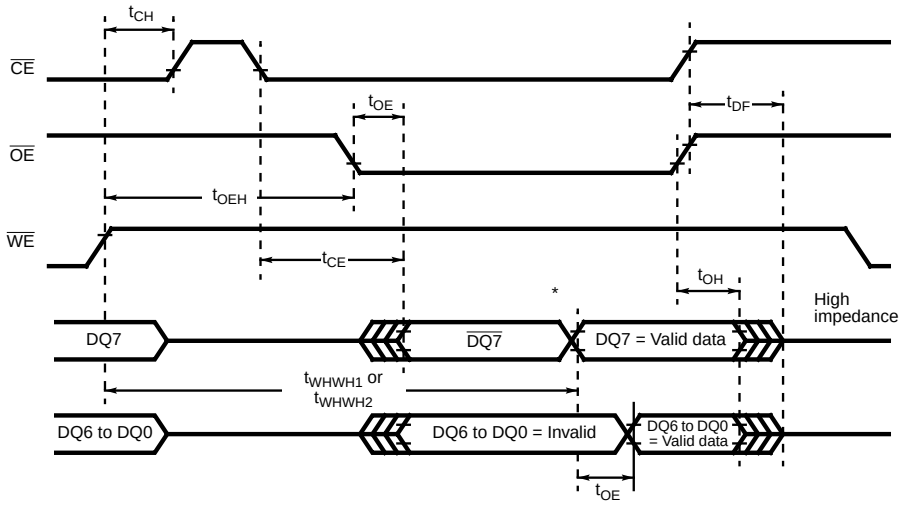
Figure C-4 Timing Diagram for Write Access (Chip Erasing/Sector Erasing)



Note*: SA is the sector address at sector erasing. 7AAAA$_H$ (or 6AAAA$_H$) is the address at chip erasing.

■ Data Polling

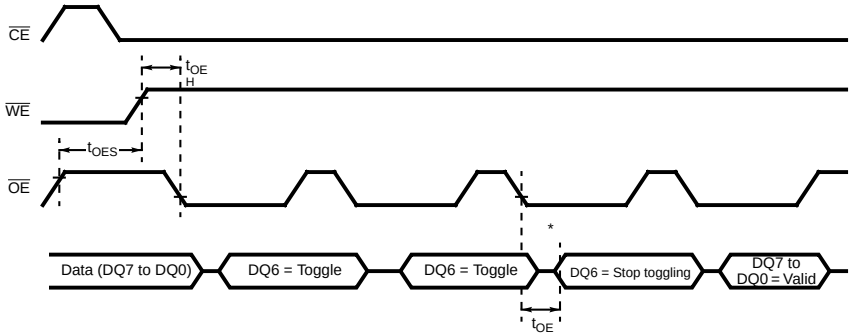
Figure C-5 Timing Diagram for Data Polling



Note*: DQ7 is valid data (The device terminates automatic operation).

■ Toggle Bit

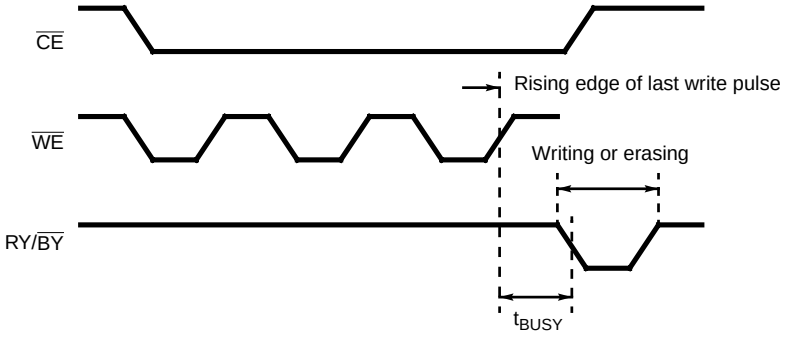
Figure C-6 Timing Diagram for Toggle Bit



Note*: DQ6 stops toggling (The device terminates automatic operation).

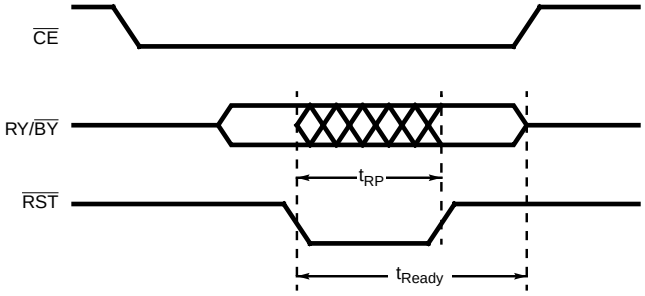
■ RY/BY Timing During Writing/erasing

Figure C-7 Timing Diagram for Output of RY/$\overline{BY}$ Signal during Writing/Erasing



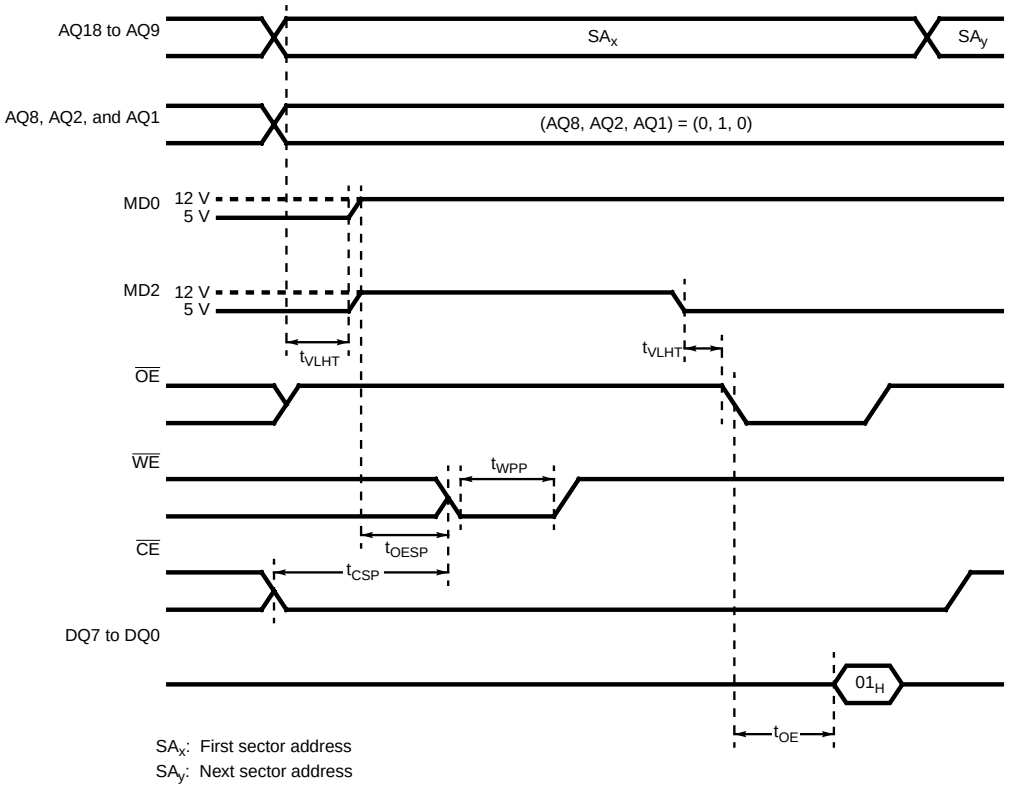
■ $\overline{\text{RST}}$ and $\overline{\text{RY/BY}}$ timing

Figure C-8 Timing Diagram for Output of RY/BY Signal at Hardware Reset



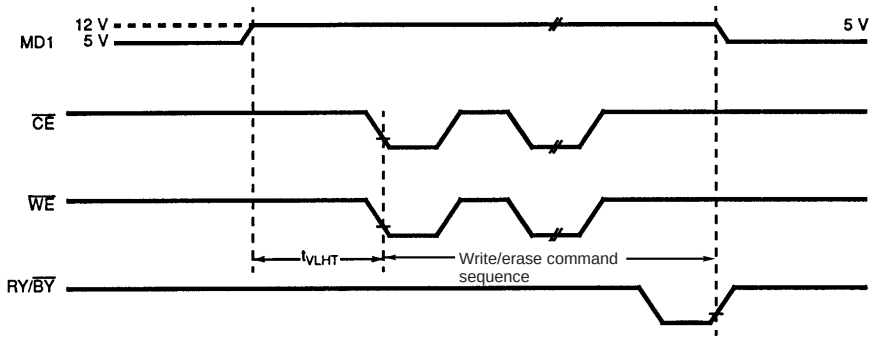
■ Enable Sector Protect/verify Sector Protect

Figure C-9 Enable Sector Protect/Verify Sector Protect



■ Temporary Sector Protect Cancellation

Figure C-10 Temporary Sector Protect Cancellation



APPENDIX D List of MB90590 Interrupt Vectors

The interrupt vector table to be referenced for interrupt processing is allocated to $FFFC00_H$ to $FFFFFF_H$ in the memory area and also used for software interrupts.

■ List of MB90590 Interrupt Vectors

Table D-1 lists the interrupt vectors for the MB90590 series.

Table D-1 MB90590 Interrupt Vectors

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 0	$FFFFEC_H$	$FFFFED_H$	$FFFFEE_H$	Unused	#0	None
. . .	. . .	. . .	. . .	. . .	. . .	. . .
INT 7	$FFFFE0_H$	$FFFFE1_H$	$FFFFE2_H$	Unused	#7	None
INT 8	$FFFFDC_H$	$FFFFDD_H$	$FFFFDE_H$	FFFFDF	#8	(RESET vector)
INT 9	$FFFFD8_H$	$FFFFD9_H$	$FFFFDA_H$	Unused	#9	ROM correction
INT 10	$FFFFD4_H$	$FFFFD5_H$	$FFFFD6_H$	Unused	#10	<Exception>
INT 11	$FFFFD0_H$	$FFFFD1_H$	$FFFFD2_H$	Unused	#11	Time base timer
INT 12	$FFFFCC_H$	$FFFFCD_H$	$FFFFCE_H$	Unused	#12	External interrupt (INT0 to INT7)
INT 13	$FFFFC8_H$	$FFFFC9_H$	$FFFFCA_H$	Unused	#13	CAN 0 RX
INT 14	$FFFFC4_H$	$FFFFC5_H$	$FFFFC6_H$	Unused	#14	CAN 0 TX/NS
INT 15	$FFFFC0_H$	$FFFFC1_H$	$FFFFC2_H$	Unused	#15	CAN 1 RX
INT 16	$FFFFBC_H$	$FFFFBD_H$	$FFFFBE_H$	Unused	#16	CAN 1 TX/NS
INT 17	$FFFFB8_H$	$FFFFB9_H$	$FFFFBA_H$	Unused	#17	PPG 0/1
INT 18	$FFFFB4_H$	$FFFFB5_H$	$FFFFB6_H$	Unused	#18	PPG 2/3
INT 19	$FFFFB0_H$	$FFFFB1_H$	$FFFFB2_H$	Unused	#19	PPG 4/5
INT 20	$FFFFAC_H$	$FFFFAD_H$	$FFFFAE_H$	Unused	#20	PPG 6/7
INT 21	$FFFFA8_H$	$FFFFA9_H$	$FFFFAA_H$	Unused	#21	PPG 8/9
INT 22	$FFFFA4_H$	$FFFFA5_H$	$FFFFA6_H$	Unused	#22	PPG A/B
INT 23	$FFFFA0_H$	$FFFFA1_H$	$FFFFA2_H$	Unused	#23	16-bit reload timer 0
INT 24	$FFFF9C_H$	$FFFF9D_H$	$FFFF9E_H$	Unused	#24	16-bit reload timer 1
INT 25	$FFFF98_H$	$FFFF99_H$	$FFFF9A_H$	Unused	#25	Input capture 0/1

Table D-1 MB90590 Interrupt Vectors (Continued)

Software interrupt instruction	Vector address L	Vector address M	Vector address H	Mode register	Interrupt No.	Hardware interrupt
INT 26	$FFFF94_H$	$FFFF95_H$	$FFFF96_H$	Unused	#26	Output compare 0/1
INT 27	FFFF90H	FFFF91H	FFFF92H	Unused	#27	Input capture 2/3
INT 28	$FFFF8C_H$	$FFFF8D_H$	$FFFF8E_H$	Unused	#28	Output compare 2/3
INT 29	$FFFF88_H$	$FFFF89_H$	$FFFF8A_H$	Unused	#29	Input capture 4/5
INT 30	$FFFF84_H$	$FFFF85_H$	$FFFF86_H$	Unused	#30	Output compare 4/5
INT 31	$FFFF80_H$	$FFFF81_H$	$FFFF82_H$	Unused	#31	A/D converter
INT 32	$FFFF7C_H$	$FFFF7D_H$	$FFFF7E_H$	Unused	#32	I/O timer/watch-dog timer
INT 33	$FFFF78_H$	$FFFF79_H$	$FFFF7A_H$	Unused	#33	Serial I/O
INT 34	$FFFF74_H$	$FFFF75_H$	$FFFF76_H$	Unused	#34	Sound generator
INT 35	$FFFF70_H$	$FFFF71_H$	$FFFF72_H$	Unused	#35	UART 0 RX
INT 36	$FFFF6C_H$	$FFFF6D_H$	$FFFF6E_H$	Unused	#36	UART 0 TX
INT 37	$FFFF68_H$	$FFFF69_H$	$FFFF6A_H$	Unused	#37	UART 1 RX
INT 38	$FFFF64_H$	$FFFF65_H$	$FFFF66_H$	Unused	#38	UART 1 TX
INT 39	FFFF60H	$FFFF61_H$	$FFFF62_H$	Unused	#39	UART 2 RX
INT 40	$FFFF5C_H$	$FFFF5D_H$	$FFFF5E_H$	Unused	#40	UART 2 TX
INT 41	$FFFF58_H$	$FFFF59_H$	$FFFF5A_H$	Unused	#41	Flash Memory
INT 42	$FFFF54_H$	$FFFF55_H$	$FFFF56_H$	Unused	#42	Delayed interrupt
INT 43	$FFFF50_H$	$FFFF51_H$	$FFFF52_H$	Unused	#43	None
. . .	. . .	. . .	. . .	. . .	. . .	. . .
INT 254	$FFFC04_H$	$FFFC05_H$	$FFFC06_H$	Unused	#254	None
INT 255	$FFFC00_H$	$FFFC01_H$	$FFFC02_H$	Unused	#255	None

■ Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Table D-2 summarizes the relationships among the interrupt causes, interrupt vectors, and

interrupt control registers of the MB90595 series.

Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Interrupt cause	EI^{2}OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Reset	N	#08	FFFFDC$_H$	—	—
INT9 instruction	N	#09	FFFFD8$_H$	—	—
Exception	N	#10	FFFFD4$_H$	—	—
Time base timer	N	#11	FFFFD0$_H$	ICR00	0000B0$_H$
External interrupt (INT0 to INT7)	Y1	#12	FFFFCC$_H$		
CAN 0 RX	N	#13	FFFFC8$_H$	ICR01	0000B1$_H$
CAN 0 TX/NS	N	#14	FFFFC4$_H$		
CAN 1 RX	N	#15	FFFFC0$_H$	ICR02	0000B2$_H$
CAN 1 TX/NS	N	#16	FFFFBC$_H$		
PPG 0/1	N	#17	FFFFB8$_H$	ICR03	0000B3$_H$
PPG 2/3	N	#18	FFFFB4$_H$		
PPG 4/5	N	#19	FFFFB0$_H$	ICR04	0000B4$_H$
PPG 6/7	N	#20	FFFFAC$_H$		
PPG 8/9	N	#21	FFFFA8$_H$	ICR05	0000B5$_H$
PPG A/B	N	#22	FFFFA4$_H$		
16-bit reload timer 0	Y1	#23	FFFFA0$_H$	ICR06	0000B6$_H$
16-bit reload timer 1	Y1	#24	FFFF9C$_H$		
Input capture 0/1	Y1	#25	FFFF98$_H$	ICR07	0000B7$_H$
Output compare 0/1	Y1	#26	FFFF94$_H$		
Input capture 2/3	Y1	#27	FFFF90$_H$	ICR08	0000B8$_H$
Output compare 2/3	Y1	#28	FFFF8C$_H$		
Input capture 4/5	Y1	#29	FFFF88$_H$	ICR09	0000B9$_H$
Output compare 4/5	Y1	#30	FFFF84$_H$		
A/D converter	Y1	#31	FFFF80$_H$	ICR10	0000BA$_H$
I/O timer/watchdog time	N	#32	FFFF7CH		
Serial I/O	Y1	#33	FFFF78$_H$	ICR11	0000BB$_H$
Sound generator	N	34	FFFF74$_H$		
UART 0 RX	Y2	35	FFFF70$_H$	ICR12	0000BC$_H$
UART 0 TX	Y1	36	FFFF6C$_H$		

Table D-2 Interrupt Causes, Interrupt Vectors, and Interrupt Control Registers

Interrupt cause	EI^{2}OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
UART 1 RX	Y2	37	FFFF68$_H$	ICR13	0000BD$_H$
UART 1 TX	Y1	38	FFFF64$_H$		
UART 2 RX	Y2	39	FFFF60H	ICR14	0000BEH
UART 2 TX	Y1	40	FFFF5CH		
Flash memory	N	41	FFFF58H	ICR15	0000BFH
Delayed interrupt	N	42	FFFF54H		

Y1: An EI^{2}OS interrupt clear signal or EI^{2}OS register read access clears the interrupt request flag.

Y2: An EI^{2}OS interrupt clear signal or EI^{2}OS register read access clears the interrupt request flag. A stop request is issued.

N: An EI^{2}OS interrupt clear signal does not clear the interrupt request flag.

Notes:

For a peripheral module having two interrupt causes for one interrupt number, an EI^{2}OS interrupt clear signal clears both interrupt request flags.

When EI^{2}OS ends, an EI^{2}OS clear signal is sent to every interrupt flag assigned to each interrupt number.

EI^{2}OS is activated when one of two interrupts assigned to an interrupt control register (ICR) is caused while EI^{2}OS is enabled. This means that an EI^{2}OS descriptor that should essentially be specific to each interrupt cause is shared by two interrupts. Therefore, while one interrupt is enabled, the other interrupt must be disabled.

INDEX

The index follows on the next page.
This is listed in alphabetic order.

Index

Numerics

16-bit free-running timer 118
16-bit free-running timer block diagram 121
16-bit free-running timer operation 125
16-bit free-running timer timing 126
16-bit I/O timer, block diagram of 119
16-bit reload timer (in internal clock mode), input pin function of 150
16-bit reload timer (with event count function), outline of 142
16-bit reload timer register 144
16-bit reload timer, block diagram of 143
16-bit reload timer, internal clock operation of 149
16-bit reload timer, output pin function of 153
16-bit reload timer, underflow operation of 152
16-bit timer register (TMR)/16-bit reload register (TMRLR), register layout of 148
24-bit operand specification 21
2M-bit flash memory feature 350
8/16-bit PPG hardware, initial value of 180
8/16-bit PPG, function of 164
8/16-bit PPG, selecting count clock for 177

A

A/D converter, block diagram of 196
acceptance filter, setting 311
acceptance filtering 307
acceptance mask registers 0 and 1 (AMR0 and AMR1) 297
acceptance mask select register (AMSR) 295
accumulator (A) 27
activation 115
address generation type 19
addressing 406
amplitude data register 333
analog input enable register 195

B

bank addressing type 22
bank select prefix 35
bit timing register (BTR) 281
bit timing, setting 311
block diagram 5, 77
buffer address pointer (BAP) 59
bus mode setting bit 96
bus operation stop (HALT = 0), condition for canceling 277
bus operation stop (HALT = 1), state during 277
bus operation stop (HALT=1), condition for setting 277

C

CAN controller, block diagram of 263
CAN controller, canceling transmission request from 305
CAN controller, completing transmission of 306
CAN controller, feature of 262
CAN controller, reception flowchart of 310
CAN controller, starting transmission of 305
CAN controller, transmission flowchart of 306
CE control 461
chip erase/sector erase command sequence 461
CLK asynchronous baud rate 229
CLK synchronous baud rate 229
clock generator, note on 70
clock selection register (CKSCR) 81
clock selection, status transition of 92
command sequence table 357
common register bank prefix (CMR) 36
compare registers 0 and 1 being used, output waveform sample when 132
compare registers, output waveform sample with two 133
condition code register (CCR) 29
continuous mode 206
continuous mode, starting EI^{2}OS in 211
control status register 136
control status register (ADCS0) 198
control status register (ADCS1) 201
control status register (CSR) 274
control status register (FMCS), flash memory 355
control status rgister 123
conversion data protection 215
counter operation state 154
CPU memory space, outline of 19

D

data counter (DCT) 58

data frame and remote frame, processing for reception of 308
data polling 462
data polling flag (DQ7) 360
data register 122
data register x (x = 0 to 15) (DTRx) 303
data registers (ADCR1 and ADCR0) 204
decrement grade register 334
delayed interrupt cause issuance/cancellation register (DIRR) 67
delayed interrupt occurrence 68
delayed interrupt, block diagram of 66
direct addressing 408
DLC register x (x = 0 to 15) (DLCRx) 302
DTP operation 189
DTP request, switching between external interrupt and 190
DTP/external interrupt, note on using 191

E

effective address field 407, 422
EI^{2}OS operation flow 61
EI^{2}OS status register (ISCS) 60
EI^{2}OS, conversion using 208
enable sector protect/verify sector protect 463
EPROM memory map 340
erasing chip 369
erasing flash memory 350
erasing sector 370
erasing sector in flash memory 370
execution cycle count 419
execution cycle count, calculating 419
extended intelligent I/O service (EI^{2}OS) 41, 56
extended intelligent I/O service descriptor (ISD) 58
extended serial I/O interface, interrupt function of 258
external clock 232
external event counter 151
external interrupt operation 188
external interrupt request 190
external shift clock mode 253

F

flag change disable prefix (NCC) 36
flash memory control signal 353
flash memory mode 353
flash memory register 350
flash memory write/erase, detailed explanation of 365
flash memory, block diagram of entire 351
flash memory, writing to 367
flash microcomputer programmer (power supplied from writer), example of minimum connection to 391
flash microcomputer programmer (user power supply used), example of minimum connection to .389
frame format, setting 311
frequency data register 332

G

general-purpose register 26

H

hardware interrupt 40, 49
hardware interrupt operation 50
hardware interrupt, occurrence and release of 51
hardware interrupt, structure of 49
hardware sequence flag 358
hardware standby mode, releasing 89
hardware standby mode, transition to 89

I

I/O map 394
I/O port 100
I/O port register 101
I/O register address pointer (IOA) 58
ID register x (x = 0 to 15) (IDRx) 300
IDE register (IDER) 284
indirect addressing 413
initial condition 340
input capture 135
input capture (2 channels per one module) 118
input capture block diagram 135
input capture data register 136
input capture fetch timing, sample of 138
input capture input timing 139
input data register (UIDR) and output data register (UODR) 225
input impedance 195
input-output circuit 12
instruction map, structure of 438
instruction presentation item and symbol, description of 424
INT9 interrupt 343
intelligent I/O service (EI^{2}OS) function and interrupt 142
intermittent CPU operation 90
internal and external clock 232
internal shift clock mode 253

Interrupt cause, interrupt vector, and interrupt control register 466
interrupt control register 466
interrupt control register (ICR) 44
interrupt disable instruction 37
interrupt flow 47
interrupt level mask register (ILM) 30
interrupt vector 43, 466
interrupt, 8/16-bit PPG 179
interrupt, intelligent I/O service (EI^{2}OS) function and 142
interrupt/DTP enable register 186
interval interrupt function 109

L

last event indicator register (LEIR) 278
layout of rate and data register (URD) 226
low power mode control register 78
low power mode control register (LPMCR) 79
lower-power control circuit, outline of 76
low-power consumption mode, setting 311
low-power mode control register access, note of ... 84
low-power mode operation 83

M

machine clock, initializing 91
main clock and PLL clock, switching between 91
MB90595 interrupt vector, list of 465
memory access mode 94
memory space map 20
memory space, multi-byte data allocation in 24
message buffer 299
message buffer (data register), list of 271
message buffer (DLC register and data register), list of 269
message buffer (ID register), list of 266
message buffer (x), procedure for reception by ... 315
message buffer (x), procedure for transmission by 313
message buffer valid register (BVALR) 283
mode data 96
mode pin 95
mode setting bit 96
multi-byte data, accessing 24
multi-level message buffer, setting configuration of 317
multiple interrupt 53

N

negative clock operation 259

O

operation, note on 66
oscillation stabilization wait time, setting 87, 89
output compare 127
output compare (2 channels per one module) 118
output compare block diagram 127
output compare register 128
output compare register 0, clearing counter upon match with 126
output compare timing 133
output compare, control status register of 129
output data register (UODR) 225
overflow, clearing counter by 125

P

package dimension 7
parity bit 234
pin assignment 6
pin function 8
PLL clock and main clock , switching between 91
port data register 102
port direction register 103
PPG0 operation mode control register (PPGC0) . 168
PPG0, 1 clock select register (PPG01) 172
PPG1 operation mode control register (PPGC1) . 170
prefix code 37
prefix code, consecutive 37
prefix instruction 37
prefix instruction, restriction on interrupt disable instruction and 37
processor status (PS) 29
program counter (PC) 32
PWM control 0 register 322
PWM1&2 compare register 323
PWM1&2 select register 324

R

rate and data register (URD) content 226
read access, data read by 460
read state, setting flash memory to 366
receive and transmit error counter (RTEC) 280
receive overrun 308
receive overrun register (ROVRR) 293
received message, storing 307
reception complete register (RCR) 291

reception interrupt enable register (RIER) 294
recommended setting 97
register bank 33
register bank pointer (RP) 30
reload value and pulse width, relationship between 8/16-bit PPG 176
remote frame receiving wait register (RFWTR) ... 287
remote frame, processing for reception of 308
remote request receiving register (RRTRR) 292
request level setting register 187
reset cause 73
reset cause occurrence 71
reset release, operation after 71
reset sequence 342
reset state, setting flash memory to 366
ROM correction address register 0/1 (PADR0/PADR1) 338
ROM correction control register (PACSR) 339
ROM correction, block diagram of 338
ROM correction, operations of 339
ROM mirroring module, block diagram of 346
ROM mirroring register (ROMM) 347
RST and RY/BY timing 463
RY/BY timing during writing/erasing 462

S

sector configuration 351
sector erase timer flag (DQ3) 364
sector, restarting erasing of flash memory 373
sector, suspending erasing of flash memory 372
serial I/O operation 252, 254
serial I/O prescaler (CDCR) 251
serial mode control register (UMC) content 221
serial mode control register (UMC), layout of 221
serial mode control status register (SMCS) 246
serial shift data register (SDR) 250
serial write connection (power supplied from writer), example of 387
serial write connection (user power supply used), example of 385
serial write connection, basic configuration of 382
set timing of six flags 235
setting ID 311
single mode 206
single mode, starting EI^{2}OS in 209
sleep mode, releasing 85
sleep mode, transition to 85
software interrupt 40, 54
software interrupt operation 54
sound control register 330
sound generator register 329
sound generator, block diagram of 328
special register 25
status flag during transmit and receive operation 239
status register (USR) content 223
status register (USR) layout 223
stepping motor controller register 321
stepping motor controller, block diagram of 320
stop mode 206
stop mode, releasing 87
stop mode, starting EI^{2}OS in 213
stop mode, transition to 87
structure 57
sub-second register 160
system stack pointer (SSP), user stack pointer (USP) and 28

T

time base counter 109
time base timer control register (TBTC) 107
time base timer, block diagram of 106
time base timer, outline of 106
timer control register 158
timer control register (TMSCR), register content of 145
timer control register (TMSCR), register layout of 145
timer mode, releasing 86
timer mode, transition to 86
timing limit exceeded flag (DQ5) 363
toggle bit 462
toggle bit flag (DQ6) 362
tone count register 335
transfer data format 233
transmission cancel register (TCANR) 288
transmission complete register (TCR) 289
transmission interrupt enable register (TIER) 290
transmission request register (TREQR) 285
transmission RTR register (TRTRR) 286

U

undefined instruction, exception due to execution of 64
undefined instruction, execution of 64
user power supply 385, 389
user stack pointer (USP) and system stack pointer (SSP) 28

W

watch-dog counter 115
watch-dog stop 115
watch-dog timer block diagram 112
watch-dog timer control register (WDTC) 113
watch-dog timer register 157
watch-dog timer, block diagram of 156
WE control 460
write data polling read (CE control) 461
write, data polling, read (WE control) 460
writer 387, 391
writing to flash memory 350

CM44-10105-2E

FUJITSU SEMICONDUCTOR • CONTROLLER MANUAL

F^2MC-16LX FAMILY
16-BIT MICROCONTROLLER
Type MB90590 SERIES
HARDWARE MANUAL

November 1999 the second edition

Published **FUJITSU LIMITED** Electronic Devices

Edited Technical Communication Dept.

FUJITSU

CM44-10105-2E