

# Hitachi Single-Chip Microcomputer

H8/539F-ZTAT™

HD64F5398

HD64F5398S

## Hardware Manual

ADE-602-108A

Rev. 2.0

3/23/98

Hitachi, Ltd.



# Preface

The H8/539F is an F-ZTAT™\*1 microcontroller with on-chip flash memory that can be reprogrammed onboard, offering even better field-programmability than ZTAT™\*2 microcontrollers with user-programmable on-chip ROM.

The H8/539F is an original Hitachi high-performance single-chip microcontroller with a high-speed 16-bit H8/500 CPU core and extensive on-chip peripheral functions. It is suitable for controlling a wide range of medium-scale office and industrial equipment and consumer products.

The general-register architecture and highly orthogonal, optimized instruction set of the H8/500 CPU enable even programs coded in the high-level C language to be compiled into efficient object code.

Many of the peripheral functions needed in microcontroller application systems are provided on-chip, including large RAM and ROM, a powerful set of timers, a serial interface, a high-precision A/D converter, and I/O ports. Compact, high-performance systems can thus be implemented easily.

Additionally, the on-chip flash memory makes this microcontroller suitable for high-speed data transfer and fast arithmetic/logic operations.

This document describes the H8/539F hardware. For further details about the H8/500 CPU instruction set, refer to the *H8/500 Series Programming Manual*.

Notes: 1. F-ZTAT™ (Flexible-ZTAT) is a trademark of Hitachi, Ltd.

2. ZTAT™ (Zero Turn-Around Time) is a registered trademark of Hitachi, Ltd.

# Contents

Section 1	Overview .....	1
1.1	Features .....	1
1.2	Block Diagram .....	5
1.3	Pin Descriptions .....	6
1.3.1	Pin Arrangement .....	6
1.3.2	Pin Functions .....	8
Section 2	CPU .....	27
2.1	Overview .....	27
2.1.1	Features .....	27
2.1.2	Address Space .....	28
2.1.3	Programming Model .....	30
2.2	General Registers .....	31
2.2.1	Overview .....	31
2.2.2	Register Configuration .....	31
2.2.3	Stack Pointer .....	31
2.2.4	Frame Pointer .....	31
2.3	Control Registers .....	32
2.3.1	Overview .....	32
2.3.2	Register Configuration .....	32
2.3.3	Program Counter .....	32
2.3.4	Status Register .....	33
2.4	Page Registers .....	36
2.4.1	Overview .....	36
2.4.2	Register Configuration .....	37
2.4.3	Code Page Register .....	37
2.4.4	Data Page Register .....	38
2.4.5	Extended Page Register .....	38
2.4.6	Stack Page Register .....	38
2.5	Base Register .....	39
2.5.1	Overview .....	39
2.5.2	Register Configuration .....	39
2.6	Data Formats .....	40
2.6.1	Data Formats in General Registers .....	40
2.6.2	Data Formats in Memory .....	41
2.6.3	Stack Data Formats .....	41
2.7	Addressing Modes and Effective Address Calculation .....	42
2.7.1	Addressing Modes .....	42
2.7.2	Effective Address Calculation .....	46

2.8	Operating Modes .....	48
2.8.1	Minimum Mode .....	48
2.8.2	Maximum Mode .....	48
2.9	Basic Operational Timing .....	48
2.9.1	Overview .....	49
2.9.2	Access to On-Chip Memory .....	49
2.9.3	Access to Two-State-Access Address Space .....	50
2.9.4	Access to On-Chip Supporting Modules .....	51
2.9.5	Access to Three-State-Access Address Space .....	52
2.10	CPU States .....	54
2.10.1	Overview .....	54
2.10.2	Program Execution State .....	55
2.10.3	Exception-Handling State .....	55
2.10.4	Bus-Released State .....	56
2.10.5	Reset State .....	64
2.10.6	Power-Down State .....	64
<b>Section 3 MCU Operating Modes .....</b>		<b>65</b>
3.1	Overview .....	65
3.1.1	Selection of Operating Mode .....	65
3.1.2	Register Configuration .....	66
3.2	Mode Control Register .....	67
3.3	Operating Mode Descriptions .....	68
3.3.1	Mode 1 (Expanded Minimum Mode) .....	68
3.3.2	Mode 2 (Expanded Minimum Mode) .....	68
3.3.3	Mode 3 (Expanded Maximum Mode) .....	68
3.3.4	Mode 4 (Expanded Maximum Mode) .....	68
3.3.5	Modes 5 and 6 .....	68
3.3.6	Mode 7 (Single-Chip Mode) .....	68
3.4	Pin Functions in Each Operating Mode .....	69
3.5	Memory Map in Each Mode .....	70
3.6	Notes on Use of Externally Expanded Modes .....	73
3.6.1	H8/539F (Dual Power Model) .....	73
3.7	Notes on H8/539F S-Mask Model (Single Power Source Model) .....	74
<b>Section 4 Exception Handling .....</b>		<b>75</b>
4.1	Overview .....	75
4.1.1	Exception Handling Types and Priority .....	75
4.1.2	Exception Handling Operation .....	76
4.1.3	Exception Sources and Vector Table .....	77

4.2	Reset .....	79
4.2.1	Overview .....	79
4.2.2	Reset Sequence .....	79
4.2.3	Interrupts after Reset .....	82
4.3	Address Error .....	82
4.3.1	Address Error in Instruction Prefetch .....	83
4.3.2	Address Error in Word Data Access .....	83
4.3.3	Address Error in Single-Chip Mode .....	84
4.4	Trace .....	86
4.5	Interrupts .....	86
4.6	Invalid Instructions .....	87
4.7	Trap Instructions and Zero Divide .....	88
4.8	Cases in which Exception Handling is Deferred .....	89
4.8.1	Instructions that Disable Exception Handling .....	89
4.8.2	Disabling of Exceptions Immediately after a Reset .....	90
4.8.3	Disabling of Interrupts after a Data Transfer Cycle .....	90
4.9	Stack Status after Completion of Exception Handling .....	91
4.9.1	PC Value Pushed on Stack for Trace, Interrupts, Trap Instructions, and Zero Divide Exceptions .....	92
4.9.2	PC Value Pushed on Stack for Address Error and Invalid Instruction Exceptions...	92
4.10	Notes on Use of the Stack .....	92
<b>Section 5 H8 Multiplier (H8MULT).....</b>		<b>93</b>
5.1	Overview.....	93
5.1.1	Features .....	93
5.1.2	Block Diagram .....	94
5.1.3	Register Configuration.....	95
5.2	Register Descriptions.....	96
5.2.1	MULT Control Register .....	96
5.2.2	MULT Base Address Register.....	98
5.2.3	MULT Multiplier Address Register .....	98
5.2.4	MULT Multiplicand Address Register.....	98
5.2.5	MULT Multiplier Register A .....	99
5.2.6	MULT Multiplier Register B.....	99
5.2.7	MULT Multiplier Register C.....	99
5.2.8	MULT Immediate Multiplier Register .....	100
5.2.9	MULT Immediate Multiplicand Register.....	100
5.2.10	MULT Result Register, Extended High Word .....	101
5.2.11	MULT Result Register, High Word.....	101
5.2.12	MULT Result Register, Low Word .....	101

5.3	Operation .....	102
5.3.1	Initialization of MULT Result Registers .....	102
5.3.2	Writing to MULT Multiplier Registers .....	103
5.3.3	Bus-Stealing Function.....	103
5.3.4	Multiply and Multiply-Accumulate Functions .....	106
<b>Section 6</b>	<b>Interrupt Controller .....</b>	<b>115</b>
6.1	Overview .....	115
6.1.1	Features .....	115
6.1.2	Block Diagram .....	116
6.1.3	Register Configuration .....	117
6.2	Interrupt Sources .....	118
6.2.1	NMI Interrupt.....	120
6.2.2	IRQ0 Interrupt.....	121
6.2.3	IRQ1 to IRQ3 Interrupt.....	121
6.2.4	Internal Interrupts .....	124
6.3	Register Descriptions .....	125
6.3.1	Interrupt Priority Registers A to F .....	125
6.3.2	Timing of Priority Changes .....	126
6.4	Interrupt Operations .....	127
6.4.1	Operations up to Interrupt Acceptance .....	127
6.4.2	Interrupt Exception Handling .....	129
6.4.3	Interrupt Exception Handling Sequence .....	131
6.4.4	Stack after Interrupt Exception Handling .....	133
6.5	Interrupts during DTC Operation .....	134
6.6	Interrupt Response Time .....	135
<b>Section 7</b>	<b>Data Transfer Controller .....</b>	<b>137</b>
7.1	Overview .....	137
7.1.1	Features .....	137
7.1.2	Block Diagram .....	138
7.1.3	Register Configuration .....	139
7.2	Register Descriptions .....	140
7.2.1	Data Transfer Mode Register .....	140
7.2.2	Data Transfer Source Address Register .....	141
7.2.3	Data Transfer Destination Address Register .....	141
7.2.4	Data Transfer Count Register .....	142
7.2.5	Data Transfer Enable Registers A to F .....	142
7.2.6	Note on Timing of DTE Modifications .....	143
7.3	Operation .....	144
7.3.1	DTC Operations .....	144
7.3.2	DTC Vector Table .....	146

7.3.3	Location of Register Information in Memory .....	149
7.3.4	Number of States per Data Transfer .....	150
7.4	Procedure for Using DTC .....	152
7.5	Example .....	153
<b>Section 8</b>	<b>Wait-State Controller .....</b>	<b>157</b>
8.1	Overview .....	157
8.1.1	Features .....	157
8.1.2	Block Diagram .....	158
8.1.3	Register Configuration .....	158
8.2	Wait Control Register .....	159
8.3	Operation .....	160
8.3.1	Programmable Wait Mode .....	161
8.3.2	Pin Wait Mode .....	162
8.3.3	Pin Auto-Wait Mode .....	163
<b>Section 9</b>	<b>Clock Pulse Generator .....</b>	<b>165</b>
9.1	Overview .....	165
9.1.1	Block Diagram .....	165
9.2	Oscillator Circuit .....	166
9.2.1	Connecting a Crystal Resonator .....	166
9.2.2	External Clock Input .....	168
9.3	Duty Adjustment Circuit .....	170
<b>Section 10</b>	<b>I/O Ports .....</b>	<b>171</b>
10.1	Overview .....	171
10.2	Port 1 .....	174
10.2.1	Overview .....	174
10.2.2	Register Descriptions .....	175
10.2.3	Pin Functions in Each Mode .....	176
10.2.4	Port 1 Read/Write Operations .....	178
10.3	Port 2 .....	180
10.3.1	Overview .....	180
10.3.2	Register Descriptions .....	181
10.3.3	Pin Functions in Each Mode .....	182
10.3.4	Port 2 Read/Write Operations .....	183
10.4	Port 3 .....	185
10.4.1	Overview .....	185
10.4.2	Register Descriptions .....	186
10.4.3	Pin Functions in Each Mode .....	187
10.4.4	Port 3 Read/Write Operations .....	188

10.5	Port 4 .....	190
10.5.1	Overview .....	190
10.5.2	Register Descriptions .....	191
10.5.3	Pin Functions in Each Mode .....	192
10.5.4	Port 4 Read/Write Operations .....	192
10.6	Port 5 .....	195
10.6.1	Overview .....	195
10.6.2	Register Descriptions .....	196
10.6.3	Pin Functions in Each Mode .....	197
10.6.4	Port 5 Read/Write Operations .....	198
10.7	Port 6 .....	201
10.7.1	Overview .....	201
10.7.2	Register Descriptions .....	202
10.7.3	Pin Functions in Each Mode .....	203
10.7.4	Port 6 Read/Write Operations .....	203
10.8	Port 7 .....	208
10.8.1	Overview .....	208
10.8.2	Register Descriptions .....	209
10.8.3	Pin Functions in Each Mode .....	210
10.8.4	Port 7 Read/Write Operations .....	211
10.9	Port 8 .....	216
10.9.1	Overview .....	216
10.9.2	Register Descriptions .....	216
10.9.3	Port 8 Read Operation.....	217
10.10	Port 9 .....	218
10.10.1	Overview .....	218
10.10.2	Register Descriptions .....	218
10.10.3	Port 9 Read Operation .....	219
10.11	Port A .....	220
10.11.1	Overview .....	220
10.11.2	Register Descriptions .....	221
10.11.3	Pin Functions in Each Mode .....	222
10.11.4	Port A Read/Write Operations .....	226
10.12	Port B .....	233
10.12.1	Overview .....	233
10.12.2	Register Descriptions .....	234
10.12.3	Pin Functions in Each Mode .....	235
10.12.4	Built-In Pull-Up Transistors .....	237
10.12.5	Port B Read/Write Operations .....	237



10.13	Port C .....	241
10.13.1	Overview .....	241
10.13.2	Register Descriptions .....	242
10.13.3	Pin Functions in Each Mode .....	243
10.13.4	Built-In MOS Pull-Up Transistors .....	245
10.13.5	Port C Read/Write Operations .....	245
10.14	ø Pin .....	249
10.14.1	Overview .....	249
10.14.2	Register Description.....	249
Section 11 16-Bit Integrated-Timer Pulse Unit .....		251
11.1	Overview .....	251
11.1.1	Features .....	251
11.1.2	Block Diagram .....	252
11.1.3	Input/Output Pins .....	253
11.2	Timer Counters and Compare/Capture Registers .....	254
11.3	Channel 1 Registers .....	255
11.3.1	Register Configuration .....	256
11.3.2	Timer Control Register (High) .....	258
11.3.3	Timer Control Register (Low) .....	260
11.3.4	Timer Status Register (High) .....	264
11.3.5	Timer Status Register (Low) .....	268
11.3.6	Timer Output Enable Register .....	272
11.4	Channel 2 to 5 Registers .....	277
11.4.1	Register Configuration .....	278
11.4.2	Timer Control Register (Low) .....	282
11.4.3	Timer Status Register (High) .....	284
11.4.4	Timer Status Register (Low) .....	286
11.4.5	Timer Output Enable Register .....	288
11.5	Channel 6 and 7 Registers .....	291
11.5.1	Register Configuration .....	292
11.5.2	Timer Status Register (High) .....	294
11.5.3	Timer Status Register (Low) .....	296
11.5.4	Timer Output Enable Register .....	298
11.6	IPU Register Descriptions .....	300
11.6.1	Timer Mode Register A .....	300
11.6.2	Timer Mode Register B .....	303
11.6.3	Timer Start Register .....	306
11.7	H8/500 CPU Interface .....	308
11.7.1	16-Bit Accessible Registers .....	308
11.7.2	Eight-Bit Accessible Registers .....	311

11.8	Examples of Timer Operation .....	314
11.8.1	Examples of Counting .....	314
11.8.2	Selection of Output Level .....	317
11.8.3	Input Capture Function .....	320
11.8.4	Counter Clearing Function .....	324
11.8.5	PWM Output Mode .....	326
11.8.6	Synchronizing Mode .....	330
11.8.7	External Event Counting .....	333
11.8.8	Programmed Periodic Counting Mode .....	336
11.8.9	Phase Counting Mode .....	339
11.9	Interrupts .....	345
11.9.1	Interrupt Timing .....	345
11.9.2	Interrupt Sources and DTC Interrupts .....	347
11.10	Notes and Precautions .....	349
<b>Section 12</b>	<b>PWM Timers .....</b>	<b>361</b>
12.1	Overview.....	361
12.1.1	Features .....	361
12.1.2	Block Diagram .....	362
12.1.3	Pin Configuration.....	363
12.1.4	Register Configuration.....	363
12.2	Register Descriptions.....	364
12.2.1	Timer Counter (TCNT).....	364
12.2.2	Duty Register (DTR).....	364
12.2.3	Timer Control Register (TCR).....	365
12.3	PWM Timer Operation .....	367
12.4	Usage Notes .....	369
<b>Section 13</b>	<b>Watchdog Timer .....</b>	<b>371</b>
13.1	Overview .....	371
13.1.1	Features .....	371
13.1.2	Block Diagram .....	372
13.1.3	Register Configuration .....	372
13.2	Register Descriptions .....	373
13.2.1	Timer Counter .....	373
13.2.2	Timer Control/Status Register .....	374
13.2.3	Reset Control/Status Register .....	376
13.2.4	Notes on Register Access .....	377
13.3	Operation .....	379
13.3.1	Watchdog Timer Operation .....	379
13.3.2	Interval Timer Operation .....	380

13.3.3	Operation in Software Standby Mode .....	381
13.3.4	Timing of Setting of Overflow Flag (OVF) .....	382
13.3.5	Timing of Setting of Watchdog Timer Reset Bit (WRST) .....	383
13.4	Usage Notes .....	384
<b>Section 14 Serial Communication Interface .....</b>		<b>385</b>
14.1	Overview .....	385
14.1.1	Features .....	385
14.1.2	Block Diagram .....	386
14.1.3	Input/Output Pins .....	387
14.1.4	Register Configuration .....	387
14.2	Register Descriptions .....	389
14.2.1	Receive Shift Register .....	389
14.2.2	Receive Data Register .....	389
14.2.3	Transmit Shift Register .....	390
14.2.4	Transmit Data Register .....	390
14.2.5	Serial Mode Register .....	391
14.2.6	Serial Control Register .....	395
14.2.7	Serial Status Register .....	399
14.2.8	Bit Rate Register .....	404
14.3	Operation .....	411
14.3.1	Overview .....	411
14.3.2	Operation in Asynchronous Mode .....	413
14.3.3	Clocked Synchronous Operation .....	423
14.3.4	Multiprocessor Communication .....	433
14.4	Interrupts and DTC .....	441
14.5	Usage Notes .....	441
<b>Section 15 A/D Converter .....</b>		<b>445</b>
15.1	Overview .....	445
15.1.1	Features .....	445
15.1.2	Block Diagram .....	446
15.1.3	Input/Output Pins .....	447
15.1.4	Register Configuration .....	448
15.2	Register Descriptions .....	449
15.2.1	A/D Data Registers 0 to B .....	449
15.2.2	A/D Control Status Register .....	450
15.2.3	A/D Control Register .....	454
15.2.4	A/D Trigger Register .....	455
15.3	H8/500 CPU Interface .....	457

15.4	Operation .....	459
15.4.1	Single Mode .....	459
15.4.2	Scan Mode .....	462
15.4.3	Analog Input Sampling and A/D Conversion Time .....	465
15.4.4	External Triggering of A/D Conversion .....	467
15.4.5	Starting A/D Conversion by IPU .....	467
15.5	Interrupts and DTC .....	468
15.6	Usage Notes .....	468
<b>Section 16</b>	<b>Bus Controller .....</b>	<b>473</b>
16.1	Overview .....	473
16.1.1	Features .....	473
16.1.2	Block Diagram .....	474
16.1.3	Register Configuration .....	475
16.2	Register Descriptions .....	475
16.2.1	Byte Area Top Register .....	475
16.2.2	Three-State Area Top Register .....	476
16.2.3	Bus Control Register .....	477
16.3	Operation .....	481
16.3.1	Operation after Reset in Each Mode .....	481
16.3.2	Timing of Changes in Bus Areas and Bus Size .....	488
16.3.3	I/O Port Expansion Function .....	490
16.4	Usage Notes .....	491
<b>Section 17</b>	<b>RAM .....</b>	<b>499</b>
17.1	Overview .....	499
17.1.1	Block Diagram .....	499
17.1.2	Register Configuration .....	500
17.2	RAM Control Register .....	500
17.3	Operation .....	501
17.3.1	Expanded Modes (Modes 1 to 6) .....	501
17.3.2	Single-Chip Mode (Mode 7) .....	501
<b>Section 18</b>	<b>Flash Memory (H8/539F S-Mask model) .....</b>	<b>503</b>
18.1	Overview .....	503
18.1.1	Block Diagram .....	503
18.1.2	Register Configuration .....	504
18.2	RAM Control Register .....	504
18.3	Operation .....	505
18.3.1	Expanded Modes (Modes1 to 6) .....	505
18.3.2	Single-Chip Mode (Mode7) .....	505

Section 19	Flash Memory (H8/539F Dual power source system ( $V_{PP} = 12\text{ V}$ ))	507
19.1	Overview	507
19.1.1	Flash Memory Overview	507
19.1.2	Mode Origramming and Flash Memory Address Space	508
19.1.3	Features	508
19.1.4	Block Diagram	510
19.1.5	Input/Output Pins	511
19.1.6	Register Configuration	511
19.2	Register Descriptions	512
19.2.1	Flash Memory Control Register (FLMCR)	512
19.2.2	Erase Block Register 1 (EBR1)	514
19.2.3	Erase Block Register 2 (EBR2)	515
19.2.4	RAM Control Register (RAMCR)	517
19.2.5	Flash Memory Emulation Register (FLMER)	518
19.2.6	Flash Memory Status Register (FLMSR)	519
19.3	On-Board Programming Mods	523
19.3.1	Boot Mode	523
19.3.2	User Program Mode	529
19.4	Programming and Erasing Flash Memory	531
19.4.1	Program Mode	531
19.4.2	Program-Verify Mode	531
19.4.3	Programming Flowchart and Sample Program	533
19.4.4	Erase Mode	538
19.4.5	Erase-Verify Mode	538
19.4.6	Erasing Flowchart and Sample Program	539
19.4.7	Prewrite-Verify Mode	554
19.4.8	Protect Modes	554
19.4.9	NMI Input Masking	557
19.5	Flash Memory Emulation by RAM	558
19.6	PROM Mode	563
19.6.1	PROM Mode Setting	563
19.6.2	Socket Adapter and Memory Map	563
19.6.3	Operation in PROM Mode	564
19.7	Flash Memory Programming and Erasing Precautions	572
19.8	Notes on Mounting Board Development—Handling of $V_{pp}$ and Mode MD2 Pins	578
Section 20	Flash Memory (H8/539F S-Mask Model: Single Power Source)	581
20.1	Overview	581
20.1.1	Notes on S-Mask Model (Single Power Source)	581
20.1.2	Mode Pin Settings and ROM Space	582
20.1.3	Features	583

20.1.4	Block Diagram .....	584
20.1.5	Pin Configuration .....	585
20.1.6	Register Configuration .....	585
20.2	Register Descriptions .....	586
20.2.1	Flash Memory Control Register (FLMCR) .....	586
20.2.2	Erase Block Register 1 (EBR1) .....	590
20.2.3	RAM Control Register (RAMCR) .....	592
20.2.4	Flash Memory Emulation Register (FLMER) .....	593
20.2.5	Flash Memory Status Register (FLMSR) .....	594
20.3	On-Board Programming Mods .....	598
20.3.1	Boot Mode .....	598
20.3.2	User Program Mode .....	604
20.4	Programming and Erasing Flash Memory .....	606
20.4.1	Program Mode .....	606
20.4.2	Program-Verify Mode .....	607
20.4.3	Sample 32-byte Programming Flowchart .....	609
20.4.4	Erase Mode .....	616
20.4.5	Erase-Verify Mode .....	616
20.4.6	Sample Flowchart for Erasing One Block .....	617
20.4.7	Protection Modes .....	626
20.4.8	NMI Input Disabling Conditions .....	629
20.5	Flash Memory Emulation in RAM .....	631
20.6	PROM Mode .....	636
20.6.1	PROM Mode Setting .....	636
20.6.2	Socket Adapter and Memory Map .....	636
20.6.3	PROM Mode Operation .....	637
20.6.4	Memory Read Mode .....	640
20.6.5	Auto-Program Mode .....	644
20.6.6	Auto-Erase Mode .....	646
20.6.7	Status Read Mode .....	647
20.6.8	PROM Mode Transition Time .....	649
20.6.9	Notes on Memory Programming .....	650
20.7	Flash Memory Programming and Erasing Precautions .....	652
Section 21 Power-Down State .....		657
21.1	Overview .....	657
21.2	Sleep Mode .....	658
21.2.1	Transition to Sleep Mode .....	658
21.2.2	Exit from Sleep Mode .....	658
21.3	Software Standby Mode .....	659
21.3.1	Transition to Software Standby Mode .....	659

21.3.2	Software Standby Control Register.....	659
21.3.3	Exit from Software Standby Mode .....	660
21.3.4	Sample Application of Software Standby Mode .....	661
21.3.5	Note .....	661
21.4	Hardware Standby Mode .....	662
21.4.1	Transition to Hardware Standby Mode .....	662
21.4.2	Recovery from Hardware Standby Mode .....	662
21.4.3	Timing for Hardware Standby Mode .....	663
<b>Section 22 Electrical Characteristics (H8/539F).....</b>		<b>665</b>
22.1	Absolute Maximum Ratings .....	665
22.2	Electrical Characteristics .....	666
22.2.1	DC Characteristics .....	666
22.2.2	AC Characteristics .....	670
22.2.3	A/D Conversion Characteristics .....	676
22.2.4	Flash Memory Characteristics .....	677
22.3	Operational Timing .....	678
22.3.1	Bus Timing .....	678
22.3.2	Control Signal Timing .....	682
22.3.3	Clock Timing .....	685
22.3.4	I/O Port Timing .....	686
22.3.5	PWM Timing Output Timing .....	686
22.3.6	IPU Timing .....	687
22.3.7	SCI Input/Output Timing .....	688
22.3.8	Flash Memory Read Timing .....	689
<b>Section 23 Electrical Characteristics (H8/539F S Mask model).....</b>		<b>691</b>
23.1	Absolute Maximum Ratings .....	691
23.2	Electrical Characteristics .....	692
23.2.1	DC Characteristics .....	692
23.2.2	AC Characteristics .....	697
23.2.3	A/D Conversion Characteristics .....	702
23.2.4	Flash Memory Characteristics .....	703
23.3	Operational Timing .....	705
23.3.1	Bus Timing .....	705
23.3.2	Control Signal Timing .....	709
23.3.3	Clock Timing .....	711
23.3.4	I/O Port Timing .....	712
23.3.5	PWM Timing Output Timing .....	712
23.3.6	IPU Timing .....	713
23.3.7	SCI Input/Output Timing .....	714

Appendix A	Instruction Set .....	715
A.1	Instruction List .....	715
A.2	Machine-Language Instruction Codes .....	722
A.3	Operation Code Map.....	734
A.4	Number of States Required for Execution .....	739
A.5	Instruction Set .....	749
A.5.1	Features .....	749
A.5.2	Instruction Types .....	749
A.5.3	Basic Instruction Formats .....	750
A.5.4	Data Transfer Instructions .....	751
A.5.5	Arithmetic Instructions .....	755
A.5.6	Logic Instructions .....	762
A.5.7	Shift Instructions .....	764
A.5.8	Bit Manipulation Instructions .....	766
A.5.9	Branch Instructions .....	769
A.5.10	System Control Instructions .....	777
A.5.11	Short-Format Instructions .....	784
Appendix B	Initial Values of CPU Registers .....	785
Appendix C	On-Chip Registers .....	786
C.1	H8/539F Dual power source model .....	786
C.2	H8/539F S Mask model .....	806
Appendix D	Pin Function Selection .....	826
D.1	Port 3 Function Selection .....	826
D.2	Port 4 Function Selection .....	827
D.3	Port 5 Function Selection .....	829
D.4	Port 6 Function Selection .....	831
D.5	Port 7 Function Selection .....	832
D.6	Port A Function Selection .....	834
Appendix E	I/O Port Block Diagrams .....	839
Appendix F	Memory Maps .....	860
Appendix G	Pin States .....	861
G.1	States of I/O Ports .....	861
G.2	Pin States at Reset .....	863
Appendix H	Package Dimensions.....	868



# Section 1 Overview

## 1.1 Features

The H8/539F is a CMOS microcomputer unit (MCU) with an original Hitachi architecture. It consists of an H8/500 CPU core plus supporting functions required in system configurations.

The H8/500 CPU features a highly orthogonal instruction set that permits addressing modes and data sizes to be specified independently in each instruction. An internal 16-bit architecture and 16-bit, two-state access to both on-chip memory and external memory enhance the CPU's data-processing capability and provide the speed needed for realtime control applications.

The on-chip supporting functions include RAM, ROM, timers, a serial communication interface (SCI), A/D converter, and I/O ports. An on-chip data transfer controller (DTC) provides an efficient way to transfer data in either direction between memory and I/O without using the CPU.

A ZTAT™\* (Zero Turn-Around Time) version of the H8/539 is already available, with on-chip ROM that can be freely programmed by the user. However, the PROM in the ZTAT version can be programmed once only. Flash memory, on the other hand, is electrically programmable and erasable, so that it can be reprogrammed while mounted on the circuit board. Moreover, the single-transistor structure of flash memory—in contrast to the two-transistor structure of EEPROM—makes it suitable for large-capacity applications.

Use of the H8/539F with on-chip flash memory allows the program and data to be modified even after embedding in the application system, offering QTAT capability for small-lot, multiple-model production, and the possibility of optimization tuning on an individual product basis, as well as version upgrading and maintenance after shipment.

Note: \*ZTAT™ is a trademark of Hitachi, Ltd.

Table 1-1 lists the main features of the H8/539F.

**Table 1-1 Features**

<b>Feature</b>	<b>Description</b>
H8/500 CPU	<p>General-register machine</p> <ul style="list-style-type: none"> <li>• Eight 16-bit general registers</li> <li>• Five 8-bit and two 16-bit control registers</li> </ul> <p>High-speed operation</p> <ul style="list-style-type: none"> <li>• Maximum clock rate : 16 MHz (oscillator frequency: 16 MHz)</li> </ul> <p>Two operating modes</p> <ul style="list-style-type: none"> <li>• Minimum mode: maximum 64-kbyte address space</li> <li>• Maximum mode: maximum 1-Mbyte address space</li> </ul> <p>Highly orthogonal instruction set</p> <ul style="list-style-type: none"> <li>• Addressing modes and data size can be specified independently for each instruction</li> </ul> <p>Register and memory addressing modes</p> <ul style="list-style-type: none"> <li>• Register-register operations</li> <li>• Register-memory (or memory-register) operations</li> </ul> <p>Instruction set optimized for C language</p> <ul style="list-style-type: none"> <li>• Special short formats for frequently-used instructions and addressing modes</li> </ul>
Memory	<ul style="list-style-type: none"> <li>• RAM : 4-kbyte high-speed on-chip RAM</li> </ul> <hr/> <ul style="list-style-type: none"> <li>• ROM : 128-kbyte flash memory (Eight large-block divisions, eight small-block divisions) Dual power source system <math>V_{PP} = 12\text{ V}</math></li> </ul> <hr/> <ul style="list-style-type: none"> <li>• ROM : 128-kbyte flash memory 8 block divisions (32 kB × 3, 28 kB × 1, 1 kB × 4) S Mask model single power source</li> </ul>

**Table 1-1 Features (cont)**

<b>Feature</b>	<b>Description</b>												
16-bit integrated-timer pulse unit (IPU)	Pulse unit with seven 16-bit timer channels												
	<table border="1"> <thead> <tr> <th><b>Channel</b></th> <th><b>Compare Registers</b></th> <th><b>Compare/Capture Registers</b></th> </tr> </thead> <tbody> <tr> <td>Channel 1</td> <td>4</td> <td>4</td> </tr> <tr> <td>Channels 2 to 5</td> <td>2 each</td> <td>2 each</td> </tr> <tr> <td>Channels 6 &amp; 7</td> <td>—</td> <td>2 each</td> </tr> </tbody> </table>	<b>Channel</b>	<b>Compare Registers</b>	<b>Compare/Capture Registers</b>	Channel 1	4	4	Channels 2 to 5	2 each	2 each	Channels 6 & 7	—	2 each
	<b>Channel</b>	<b>Compare Registers</b>	<b>Compare/Capture Registers</b>										
	Channel 1	4	4										
Channels 2 to 5	2 each	2 each											
Channels 6 & 7	—	2 each											
Clock source can be selected independently for each channel													
<ul style="list-style-type: none"> <li>• Thirteen internal clock sources</li> <li>• Three external clock sources</li> </ul>													
Two counting modes	<ul style="list-style-type: none"> <li>• Free-running timer</li> <li>• Interval timer</li> </ul>												
Three types of pulse output	<ul style="list-style-type: none"> <li>• One-shot output</li> <li>• Toggle output</li> <li>• PWM output</li> </ul>												
Automatic measurement functions	<ul style="list-style-type: none"> <li>• Programmable period counting</li> <li>• Phase counting</li> </ul>												
Synchronization function	<ul style="list-style-type: none"> <li>• Counters on different channels can be synchronized</li> </ul>												
Serial communication interface (SCI)	<ul style="list-style-type: none"> <li>• Asynchronous or clocked synchronous mode (selectable)</li> <li>• Full duplex: can send and receive simultaneously</li> <li>• Dedicated on-chip baud rate generator</li> <li>• Multiprocessor communication function (asynchronous mode)</li> </ul>												
A/D converter	<ul style="list-style-type: none"> <li>• Ten-bit resolution</li> <li>• Twelve channels, single mode or scan mode selectable</li> <li>• Can be triggered externally, or by IPU compare match</li> <li>• Selectable analog conversion voltage range</li> </ul>												
I/O ports	<ul style="list-style-type: none"> <li>• 74 input/output pins</li> <li>• 12 input-only pins</li> </ul>												
Interrupt controller (INTC)	<ul style="list-style-type: none"> <li>• Five external interrupt pins (NMI, <math>\overline{IRQ_0}</math> to <math>\overline{IRQ_3}</math>)</li> <li>• Thirty-nine internal interrupt sources</li> <li>• Eight programmable priority levels</li> </ul>												

**Table 1-1 Features (cont)**

<b>Feature</b>	<b>Description</b>												
Data transfer controller (DTC)	<ul style="list-style-type: none"> <li>• Can transfer data in both directions between memory and I/O without using the CPU</li> </ul>												
Wait-state controller (WSC)	<ul style="list-style-type: none"> <li>• Can insert wait states (<math>T_W</math>) in access to external I/O or memory</li> </ul>												
Bus controller (BSC)	<ul style="list-style-type: none"> <li>• Address space can be partitioned into 16-bit-bus and 8-bit-bus areas</li> <li>• Address space can be partitioned into two-state-access and three-state-access areas</li> <li>• I/O ports can be expanded and reconfigured</li> </ul>												
Operating modes	<p>Seven operating modes</p> <ol style="list-style-type: none"> <li>1. High-speed 16-bit bus modes, starting in 2-state 16-bit mode at reset <ul style="list-style-type: none"> <li>• Expanded minimum mode (mode 1)</li> <li>• Expanded maximum modes (modes 3 and 4)</li> </ul> </li> <li>2. Low-speed 16-bit bus modes, starting in 3-state 8-bit mode on reset release <ul style="list-style-type: none"> <li>• Expanded minimum mode (mode 6)</li> <li>• Expanded maximum mode (mode 5)</li> </ul> </li> <li>3. Low-speed 8-bit bus mode <ul style="list-style-type: none"> <li>• Expanded minimum mode (mode 2)</li> </ul> </li> <li>4. Single-chip mode <ul style="list-style-type: none"> <li>• Maximum mode (mode 7)</li> </ul> </li> </ol>												
Power-down state	<p>Three power-down modes</p> <ul style="list-style-type: none"> <li>• Sleep mode</li> <li>• Software standby mode</li> <li>• Hardware standby mode</li> </ul>												
Watchdog timer (WDT)	<ul style="list-style-type: none"> <li>• Timer overflow can generate reset output (H8/539F dual power source model only)</li> <li>• Also usable as an interval timer</li> </ul>												
PWM timer	<ul style="list-style-type: none"> <li>• Duty cycle: 0% to 100%</li> <li>• Resolution: 1/250</li> </ul>												
Multiplier (MULT)	<ul style="list-style-type: none"> <li>• 16 bit <math>\times</math> 16 bit signed or unsigned multiplication</li> <li>• Multiply-accumulate: 32 bits (saturating); 42 bits (non-saturating)</li> </ul>												
Other features	<ul style="list-style-type: none"> <li>• On-chip clock oscillator</li> </ul>												
Product lineup	<table border="1"> <thead> <tr> <th></th> <th><b>Model</b></th> <th><b>Package</b></th> <th><b>ROM</b></th> </tr> </thead> <tbody> <tr> <td>Dual power source system</td> <td>HD64F5398F</td> <td>112-pin plastic QFP (FP-112)</td> <td>Flash memory (<math>V_{PP} = 12\text{ V}</math>)</td> </tr> <tr> <td>Single power source system (S mask model)</td> <td>HD64F5398SF</td> <td>112-pin plastic QFP (FP-112)</td> <td>Flash memory (single power source)</td> </tr> </tbody> </table>		<b>Model</b>	<b>Package</b>	<b>ROM</b>	Dual power source system	HD64F5398F	112-pin plastic QFP (FP-112)	Flash memory ( $V_{PP} = 12\text{ V}$ )	Single power source system (S mask model)	HD64F5398SF	112-pin plastic QFP (FP-112)	Flash memory (single power source)
	<b>Model</b>	<b>Package</b>	<b>ROM</b>										
Dual power source system	HD64F5398F	112-pin plastic QFP (FP-112)	Flash memory ( $V_{PP} = 12\text{ V}$ )										
Single power source system (S mask model)	HD64F5398SF	112-pin plastic QFP (FP-112)	Flash memory (single power source)										

## 1.2 Block Diagram

Figure 1-1 shows a block diagram of the H8/539F.

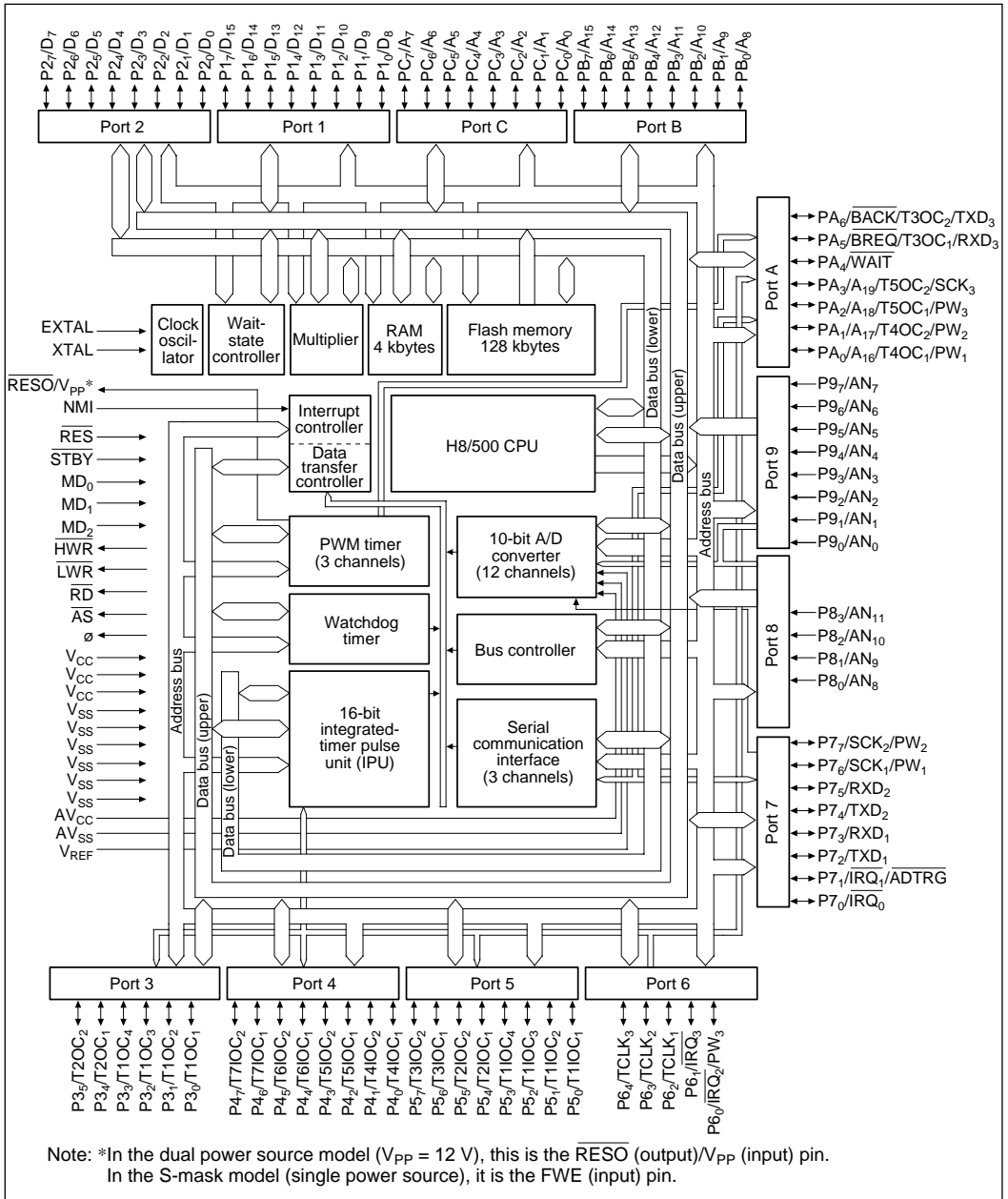


Figure 1-1 H8/539F Block Diagram

# 1.3 Pin Descriptions

## 1.3.1 Pin Arrangement

Figure 1-2 (a) shows the pin arrangement of the H8/539F (FP-112 package).

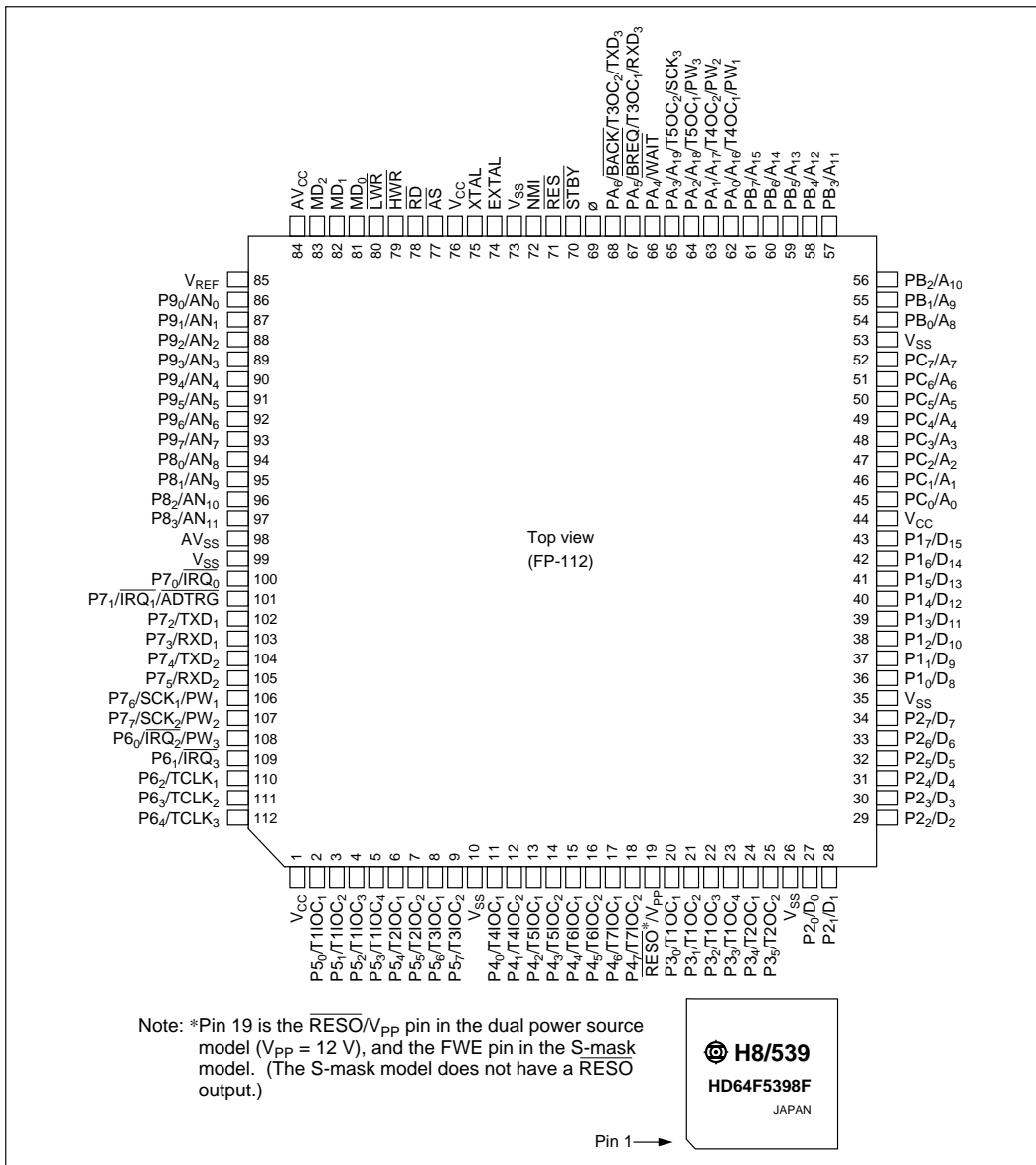
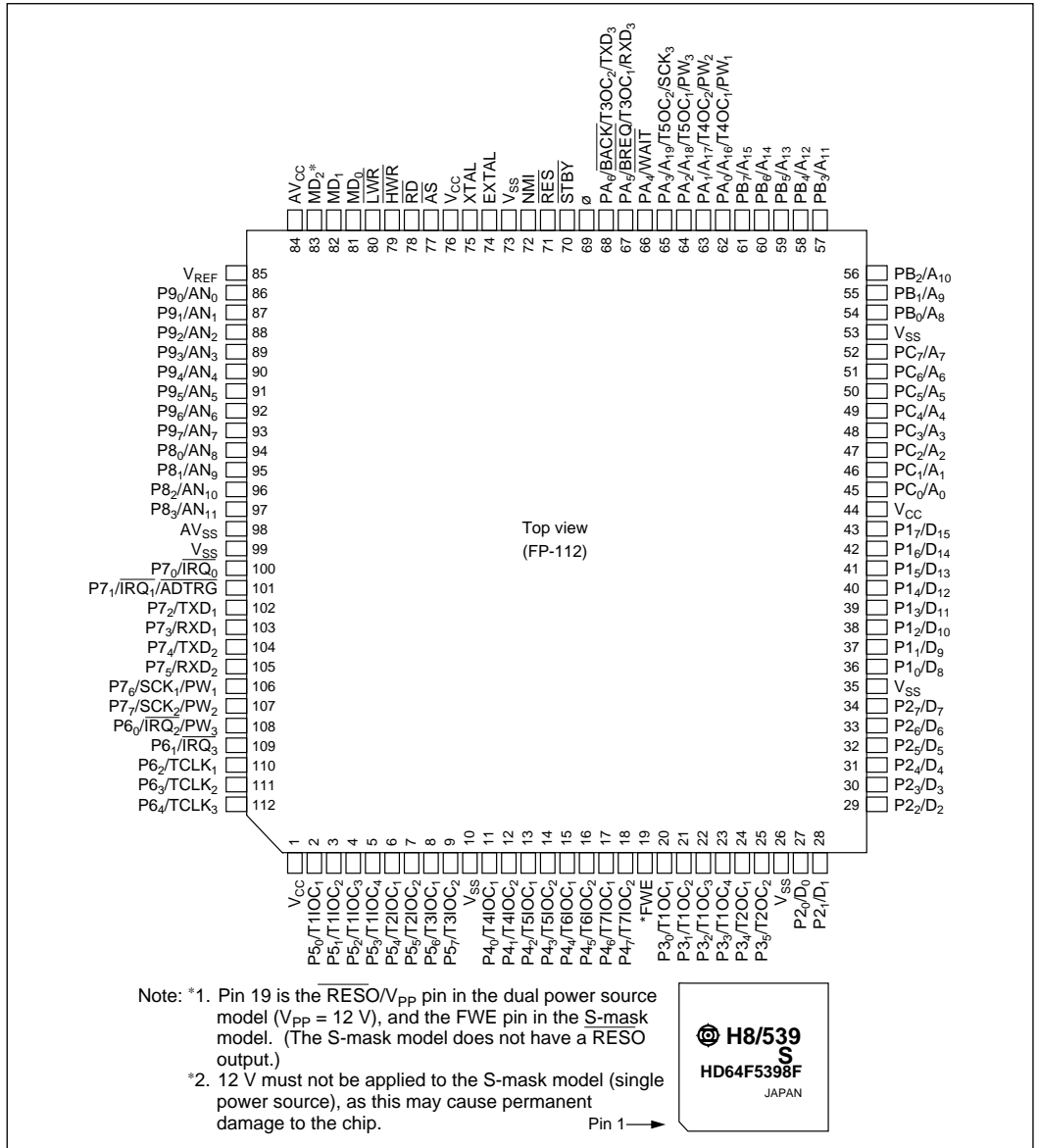


Figure 1-2 (a) H8/539F Pin Arrangement (FP-112, Top View)

Figure 1-2 (b) shows the pin arrangement of the H8/539F S-mask model (FP-112 package). The pin arrangement is the same for the S-mask model (single power source) and dual power source model, but the function of the FWE pin (pin 19) is different.



**Figure 1-2 (b) H8/539F S-Mask Model Pin Arrangement (FP-112, Top View)**

### 1.3.2 Pin Functions

(1) **Pin Assignments in Each Operating Mode:** Table 1-2 lists the assignments of the pins of the FP-112 package in each operating mode.

**Table 1-2 Pin Assignments in Each Operating Mode (FP-112)**

No.	Expanded Minimum Modes		Expanded Maximum Modes		Single-Chip Mode	Notes
	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7	
1	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	
2	P5 <sub>0</sub> /T1IOC <sub>1</sub>	P5 <sub>0</sub> /T1IOC <sub>1</sub>	P5 <sub>0</sub> /T1IOC <sub>1</sub>	P5 <sub>0</sub> /T1IOC <sub>1</sub>	P5 <sub>0</sub> /T1IOC <sub>1</sub>	
3	P5 <sub>1</sub> /T1IOC <sub>2</sub>	P5 <sub>1</sub> /T1IOC <sub>2</sub>	P5 <sub>1</sub> /T1IOC <sub>2</sub>	P5 <sub>1</sub> /T1IOC <sub>2</sub>	P5 <sub>1</sub> /T1IOC <sub>2</sub>	
4	P5 <sub>2</sub> /T1IOC <sub>3</sub>	P5 <sub>2</sub> /T1IOC <sub>3</sub>	P5 <sub>2</sub> /T1IOC <sub>3</sub>	P5 <sub>2</sub> /T1IOC <sub>3</sub>	P5 <sub>2</sub> /T1IOC <sub>3</sub>	
5	P5 <sub>3</sub> /T1IOC <sub>4</sub>	P5 <sub>3</sub> /T1IOC <sub>4</sub>	P5 <sub>3</sub> /T1IOC <sub>4</sub>	P5 <sub>3</sub> /T1IOC <sub>4</sub>	P5 <sub>3</sub> /T1IOC <sub>4</sub>	
6	P5 <sub>4</sub> /T2IOC <sub>1</sub>	P5 <sub>4</sub> /T2IOC <sub>1</sub>	P5 <sub>4</sub> /T2IOC <sub>1</sub>	P5 <sub>4</sub> /T2IOC <sub>1</sub>	P5 <sub>4</sub> /T2IOC <sub>1</sub>	
7	P5 <sub>5</sub> /T2IOC <sub>2</sub>	P5 <sub>5</sub> /T2IOC <sub>2</sub>	P5 <sub>5</sub> /T2IOC <sub>2</sub>	P5 <sub>5</sub> /T2IOC <sub>2</sub>	P5 <sub>5</sub> /T2IOC <sub>2</sub>	
8	P5 <sub>6</sub> /T3IOC <sub>1</sub>	P5 <sub>6</sub> /T3IOC <sub>1</sub>	P5 <sub>6</sub> /T3IOC <sub>1</sub>	P5 <sub>6</sub> /T3IOC <sub>1</sub>	P5 <sub>6</sub> /T3IOC <sub>1</sub>	
9	P5 <sub>7</sub> /T3IOC <sub>2</sub>	P5 <sub>7</sub> /T3IOC <sub>2</sub>	P5 <sub>7</sub> /T3IOC <sub>2</sub>	P5 <sub>7</sub> /T3IOC <sub>2</sub>	P5 <sub>7</sub> /T3IOC <sub>2</sub>	
10	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	
11	P4 <sub>0</sub> /T4IOC <sub>1</sub>	P4 <sub>0</sub> /T4IOC <sub>1</sub>	P4 <sub>0</sub> /T4IOC <sub>1</sub>	P4 <sub>0</sub> /T4IOC <sub>1</sub>	P4 <sub>0</sub> /T4IOC <sub>1</sub>	
12	P4 <sub>1</sub> /T4IOC <sub>2</sub>	P4 <sub>1</sub> /T4IOC <sub>2</sub>	P4 <sub>1</sub> /T4IOC <sub>2</sub>	P4 <sub>1</sub> /T4IOC <sub>2</sub>	P4 <sub>1</sub> /T4IOC <sub>2</sub>	
13	P4 <sub>2</sub> /T5IOC <sub>1</sub>	P4 <sub>2</sub> /T5IOC <sub>1</sub>	P4 <sub>2</sub> /T5IOC <sub>1</sub>	P4 <sub>2</sub> /T5IOC <sub>1</sub>	P4 <sub>2</sub> /T5IOC <sub>1</sub>	
14	P4 <sub>3</sub> /T5IOC <sub>2</sub>	P4 <sub>3</sub> /T5IOC <sub>2</sub>	P4 <sub>3</sub> /T5IOC <sub>2</sub>	P4 <sub>3</sub> /T5IOC <sub>2</sub>	P4 <sub>3</sub> /T5IOC <sub>2</sub>	
15	P4 <sub>4</sub> /T6IOC <sub>1</sub>	P4 <sub>4</sub> /T6IOC <sub>1</sub>	P4 <sub>4</sub> /T6IOC <sub>1</sub>	P4 <sub>4</sub> /T6IOC <sub>1</sub>	P4 <sub>4</sub> /T6IOC <sub>1</sub>	
16	P4 <sub>5</sub> /T6IOC <sub>2</sub>	P4 <sub>5</sub> /T6IOC <sub>2</sub>	P4 <sub>5</sub> /T6IOC <sub>2</sub>	P4 <sub>5</sub> /T6IOC <sub>2</sub>	P4 <sub>5</sub> /T6IOC <sub>2</sub>	
17	P4 <sub>6</sub> /T7IOC <sub>1</sub>	P4 <sub>6</sub> /T7IOC <sub>1</sub>	P4 <sub>6</sub> /T7IOC <sub>1</sub>	P4 <sub>6</sub> /T7IOC <sub>1</sub>	P4 <sub>6</sub> /T7IOC <sub>1</sub>	
18	P4 <sub>7</sub> /T7IOC <sub>2</sub>	P4 <sub>7</sub> /T7IOC <sub>2</sub>	P4 <sub>7</sub> /T7IOC <sub>2</sub>	P4 <sub>7</sub> /T7IOC <sub>2</sub>	P4 <sub>7</sub> /T7IOC <sub>2</sub>	
19	$\overline{\text{RESO}}/V_{\text{PP}}$	$\overline{\text{RESO}}/V_{\text{PP}}$	$\overline{\text{RESO}}/V_{\text{PP}}$	$\overline{\text{RESO}}/V_{\text{PP}}$	$\overline{\text{RESO}}/V_{\text{PP}}$	Dual power source (V <sub>PP</sub> =12V)
	FWE*1	FWE*1	FWE*1	FWE*1	FWE*1	S Mask model (single power source)
20	P3 <sub>0</sub> /T1OC <sub>1</sub>	P3 <sub>0</sub> /T1OC <sub>1</sub>	P3 <sub>0</sub> /T1OC <sub>1</sub>	P3 <sub>0</sub> /T1OC <sub>1</sub>	P3 <sub>0</sub> /T1OC <sub>1</sub>	
21	P3 <sub>1</sub> /T1OC <sub>2</sub>	P3 <sub>1</sub> /T1OC <sub>2</sub>	P3 <sub>1</sub> /T1OC <sub>2</sub>	P3 <sub>1</sub> /T1OC <sub>2</sub>	P3 <sub>1</sub> /T1OC <sub>2</sub>	
22	P3 <sub>2</sub> /T1OC <sub>3</sub>	P3 <sub>2</sub> /T1OC <sub>3</sub>	P3 <sub>2</sub> /T1OC <sub>3</sub>	P3 <sub>2</sub> /T1OC <sub>3</sub>	P3 <sub>2</sub> /T1OC <sub>3</sub>	
23	P3 <sub>3</sub> /T1OC <sub>4</sub>	P3 <sub>3</sub> /T1OC <sub>4</sub>	P3 <sub>3</sub> /T1OC <sub>4</sub>	P3 <sub>3</sub> /T1OC <sub>4</sub>	P3 <sub>3</sub> /T1OC <sub>4</sub>	

Notes: 1. For the PROM mode, see section 19, "Flash Memory (H8/539F)" and section 20, "Flash



Memory (H8/539F S Mask model)".

- \*1: 12 V must not be applied to the S-mask model (single power source), as this will permanently damage the device.

**Table 1-2 Pin Assignments in Each Operating Mode (FP-112) (cont)**

No.	Expanded Minimum Modes		Expanded Maximum Modes		Single-Chip Mode	Notes
	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7	
24	P3 <sub>4</sub> /T2OC <sub>1</sub>	P3 <sub>4</sub> /T2OC <sub>1</sub>	P3 <sub>4</sub> /T2OC <sub>1</sub>	P3 <sub>4</sub> /T2OC <sub>1</sub>	P3 <sub>4</sub> /T2OC <sub>1</sub>	
25	P3 <sub>5</sub> /T2OC <sub>2</sub>	P3 <sub>5</sub> /T2OC <sub>2</sub>	P3 <sub>5</sub> /T2OC <sub>2</sub>	P3 <sub>5</sub> /T2OC <sub>2</sub>	P3 <sub>5</sub> /T2OC <sub>2</sub>	
26	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	
27	D <sub>0</sub> <sup>*2</sup>	P2 <sub>0</sub>	D <sub>0</sub> <sup>*2</sup>	D <sub>0</sub> <sup>*2</sup>	P2 <sub>0</sub>	
28	D <sub>1</sub> <sup>*2</sup>	P2 <sub>1</sub>	D <sub>1</sub> <sup>*2</sup>	D <sub>1</sub> <sup>*2</sup>	P2 <sub>1</sub>	
29	D <sub>2</sub> <sup>*</sup>	P2 <sub>2</sub>	D <sub>2</sub> <sup>*2</sup>	D <sub>2</sub>	P2 <sub>2</sub>	
30	D <sub>3</sub> <sup>*</sup>	P2 <sub>3</sub>	D <sub>3</sub> <sup>*2</sup>	D <sub>3</sub>	P2 <sub>3</sub>	
31	D <sub>4</sub> <sup>*2</sup>	P2 <sub>4</sub>	D <sub>4</sub> <sup>*2</sup>	D <sub>4</sub>	P2 <sub>4</sub>	
32	D <sub>5</sub> <sup>*2</sup>	P2 <sub>5</sub>	D <sub>5</sub> <sup>*2</sup>	D <sub>5</sub>	P2 <sub>5</sub>	
33	D <sub>6</sub> <sup>*2</sup>	P2 <sub>6</sub>	D <sub>6</sub> <sup>*2</sup>	D <sub>6</sub>	P2 <sub>6</sub>	
34	D <sub>7</sub> <sup>*2</sup>	P2 <sub>7</sub>	D <sub>7</sub> <sup>*2</sup>	D <sub>7</sub>	P2 <sub>7</sub>	
35	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	
36	D <sub>8</sub>	D <sub>8</sub>	D <sub>8</sub>	D <sub>8</sub>	P1 <sub>0</sub>	
37	D <sub>9</sub>	D <sub>9</sub>	D <sub>9</sub>	D <sub>9</sub>	P1 <sub>1</sub>	
38	D <sub>10</sub>	D <sub>10</sub>	D <sub>10</sub>	D <sub>10</sub>	P1 <sub>2</sub>	
39	D <sub>11</sub>	D <sub>11</sub>	D <sub>11</sub>	D <sub>11</sub>	P1 <sub>3</sub>	
40	D <sub>12</sub>	D <sub>12</sub>	D <sub>12</sub>	D <sub>12</sub>	P1 <sub>4</sub>	
41	D <sub>13</sub>	D <sub>13</sub>	D <sub>13</sub>	D <sub>13</sub>	P1 <sub>5</sub>	
42	D <sub>14</sub>	D <sub>14</sub>	D <sub>14</sub>	D <sub>14</sub>	P1 <sub>6</sub>	
43	D <sub>15</sub>	D <sub>15</sub>	D <sub>15</sub>	D <sub>15</sub>	P1 <sub>7</sub>	
44	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	
45	A <sub>0</sub>	PC <sub>0</sub> /A <sub>0</sub>	A <sub>0</sub>	PC <sub>0</sub> /A <sub>0</sub>	PC <sub>0</sub>	
46	A <sub>1</sub>	PC <sub>1</sub> /A <sub>1</sub>	A <sub>1</sub>	PC <sub>1</sub> /A <sub>1</sub>	PC <sub>1</sub>	
47	A <sub>2</sub>	PC <sub>2</sub> /A <sub>2</sub>	A <sub>2</sub>	PC <sub>2</sub> /A <sub>2</sub>	PC <sub>2</sub>	
48	A <sub>3</sub>	PC <sub>3</sub> /A <sub>3</sub>	A <sub>3</sub>	PC <sub>3</sub> /A <sub>3</sub>	PC <sub>3</sub>	
49	A <sub>4</sub>	PC <sub>4</sub> /A <sub>4</sub>	A <sub>4</sub>	PC <sub>4</sub> /A <sub>4</sub>	PC <sub>4</sub>	
50	A <sub>5</sub>	PC <sub>5</sub> /A <sub>5</sub>	A <sub>5</sub>	PC <sub>5</sub> /A <sub>5</sub>	PC <sub>5</sub>	

Notes: 1. For the PROM mode, see section 19, “Flash Memory (H8/539F)” and section 20, “Flash Memory (H8/539F S Mask model)”.

- \*2. In modes 5 and 6, the external bus space has a 16-bit bus width, but an 8-bit bus width is set after a reset. In this case, the upper half of the data bus (D15 to D8) is enabled, and the lower half (D7 to D0) is disabled. The bus width can be changed to 16 bits

(D15 to D0) by software after setting the BCRE bit to 1 in the bus control register (BCR). In modes 1, 3, and 4, the external bus space has a 16-bit bus width (D15 to D0) after a reset, but this can be changed to 8 bits by a byte area top register (ARBT) setting. In this case, the upper half of the data bus (D15 to D8) is enabled, and the lower half (D7 to D0) is disabled. For details of the settings, see section 16, Bus Controller.

**Table 1-2 Pin Assignments in Each Operating Mode (FP-112) (cont)**

No.	Expanded Minimum Modes		Expanded Maximum Modes		Single-Chip Mode	Notes
	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7	
51	A <sub>6</sub>	PC <sub>6</sub> /A <sub>6</sub>	A <sub>6</sub>	PC <sub>6</sub> /A <sub>6</sub>	PC <sub>6</sub>	
52	A <sub>7</sub>	PC <sub>7</sub> /A <sub>7</sub>	A <sub>7</sub>	PC <sub>7</sub> /A <sub>7</sub>	PC <sub>7</sub>	
53	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	
54	A <sub>8</sub>	PB <sub>0</sub> /A <sub>8</sub>	A <sub>8</sub>	PB <sub>0</sub> /A <sub>8</sub>	PB <sub>0</sub>	
55	A <sub>9</sub>	PB <sub>1</sub> /A <sub>9</sub>	A <sub>9</sub>	PB <sub>1</sub> /A <sub>9</sub>	PB <sub>1</sub>	
56	A <sub>10</sub>	PB <sub>2</sub> /A <sub>10</sub>	A <sub>10</sub>	PB <sub>2</sub> /A <sub>10</sub>	PB <sub>2</sub>	
57	A <sub>11</sub>	PB <sub>3</sub> /A <sub>11</sub>	A <sub>11</sub>	PB <sub>3</sub> /A <sub>11</sub>	PB <sub>3</sub>	
58	A <sub>12</sub>	PB <sub>4</sub> /A <sub>12</sub>	A <sub>12</sub>	PB <sub>4</sub> /A <sub>12</sub>	PB <sub>4</sub>	
59	A <sub>13</sub>	PB <sub>5</sub> /A <sub>13</sub>	A <sub>13</sub>	PB <sub>5</sub> /A <sub>13</sub>	PB <sub>5</sub>	
60	A <sub>14</sub>	PB <sub>6</sub> /A <sub>14</sub>	A <sub>14</sub>	PB <sub>6</sub> /A <sub>14</sub>	PB <sub>6</sub>	
61	A <sub>15</sub>	PB <sub>7</sub> /A <sub>15</sub>	A <sub>15</sub>	PB <sub>7</sub> /A <sub>15</sub>	PB <sub>7</sub>	
62	PA <sub>0</sub> /T4OC <sub>1</sub> / PW <sub>1</sub>	PA <sub>0</sub> /T4OC <sub>1</sub> / PW <sub>1</sub>	A <sub>16</sub>	PA <sub>0</sub> /A <sub>16</sub> / PW <sub>1</sub>	PA <sub>0</sub> /T4OC <sub>1</sub> / PW <sub>1</sub>	
63	PA <sub>1</sub> /T4OC <sub>2</sub> / PW <sub>2</sub>	PA <sub>1</sub> /T4OC <sub>2</sub> / PW <sub>2</sub>	A <sub>17</sub>	PA <sub>1</sub> /A <sub>17</sub> / PW <sub>2</sub>	PA <sub>1</sub> /T4OC <sub>2</sub> / PW <sub>2</sub>	
64	PA <sub>2</sub> /T5OC <sub>1</sub> / PW <sub>3</sub>	PA <sub>2</sub> /T5OC <sub>1</sub> / PW <sub>3</sub>	A <sub>18</sub>	PA <sub>2</sub> /A <sub>18</sub> / PW <sub>3</sub>	PA <sub>2</sub> /T5OC <sub>1</sub> / PW <sub>3</sub>	
65	PA <sub>3</sub> /T5OC <sub>2</sub> / SCK <sub>3</sub>	PA <sub>3</sub> /T5OC <sub>2</sub> / SCK <sub>3</sub>	A <sub>19</sub>	PA <sub>3</sub> /A <sub>19</sub> / SCK <sub>3</sub>	PA <sub>3</sub> /T5OC <sub>2</sub> / SCK <sub>3</sub>	
66	PA <sub>4</sub> /WAIT	PA <sub>4</sub> /WAIT	PA <sub>4</sub> /WAIT	PA <sub>4</sub> /WAIT	PA <sub>4</sub>	
67	PA <sub>5</sub> /BREQ/ T3OC <sub>1</sub> /RXD <sub>3</sub>	PA <sub>5</sub> /BREQ/ T3OC <sub>1</sub> /RXD <sub>3</sub>	PA <sub>5</sub> /BREQ/ T3OC <sub>1</sub> /RXD <sub>3</sub>	PA <sub>5</sub> /BREQ/ T3OC <sub>1</sub> /RXD <sub>3</sub>	PA <sub>5</sub> /T3OC <sub>1</sub> / RXD <sub>3</sub>	
68	PA <sub>6</sub> /BACK/ T3OC <sub>2</sub> /TXD <sub>3</sub>	PA <sub>6</sub> /BACK/ T3OC <sub>2</sub> /TXD <sub>3</sub>	PA <sub>6</sub> /BACK/ T3OC <sub>2</sub> /TXD <sub>3</sub>	PA <sub>6</sub> /BACK/ T3OC <sub>2</sub> /TXD <sub>3</sub>	PA <sub>6</sub> /T3OC <sub>2</sub> / TXD <sub>3</sub>	
69	∅	∅	∅	∅	∅	
70	STBY	STBY	STBY	STBY	STBY	
71	RES	RES	RES	RES	RES	
72	NMI	NMI	NMI	NMI	NMI	

Notes: 1. For the PROM mode, see section 19, “Flash Memory (H8/539F)” and section 20, “Flash Memory (H8/539F S Mask model)”.

2. Pins marked NC should be left unconnected.

**Table 1-2 Pin Assignments in Each Operating Mode (FP-112) (cont)**

No.	Expanded Minimum Modes		Expanded Maximum Modes		Single-Chip Mode	Notes
	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7	
73	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	
74	EXTAL	EXTAL	EXTAL	EXTAL	EXTAL	
75	XTAL	XTAL	XTAL	XTAL	XTAL	
76	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	V <sub>CC</sub>	
77	$\overline{AS}$	$\overline{AS}$	$\overline{AS}$	$\overline{AS}$	$\overline{AS}$	
78	$\overline{RD}$	$\overline{RD}$	$\overline{RD}$	$\overline{RD}$	$\overline{RD}$	
79	$\overline{HWR}$	$\overline{HWR}$	$\overline{HWR}$	$\overline{HWR}$	$\overline{HWR}$	
80	$\overline{LWR}$	$\overline{LWR}$	$\overline{LWR}$	$\overline{LWR}$	$\overline{LWR}$	
81	MD <sub>0</sub>	MD <sub>0</sub>	MD <sub>0</sub>	MD <sub>0</sub>	MD <sub>0</sub>	
82	MD <sub>1</sub>	MD <sub>1</sub>	MD <sub>1</sub>	MD <sub>1</sub>	MD <sub>1</sub>	
83	MD <sub>2</sub> * <sup>1</sup>	MD <sub>2</sub> * <sup>1</sup>	MD <sub>2</sub> * <sup>1</sup>	MD <sub>2</sub> * <sup>1</sup>	MD <sub>2</sub> * <sup>1</sup>	
84	AV <sub>CC</sub>	AV <sub>CC</sub>	AV <sub>CC</sub>	AV <sub>CC</sub>	AV <sub>CC</sub>	
85	V <sub>REF</sub>	V <sub>REF</sub>	V <sub>REF</sub>	V <sub>REF</sub>	V <sub>REF</sub>	
86	P9 <sub>0</sub> /AN <sub>0</sub>	P9 <sub>0</sub> /AN <sub>0</sub>	P9 <sub>0</sub> /AN <sub>0</sub>	P9 <sub>0</sub> /AN <sub>0</sub>	P9 <sub>0</sub> /AN <sub>0</sub>	
87	P9 <sub>1</sub> /AN <sub>1</sub>	P9 <sub>1</sub> /AN <sub>1</sub>	P9 <sub>1</sub> /AN <sub>1</sub>	P9 <sub>1</sub> /AN <sub>1</sub>	P9 <sub>1</sub> /AN <sub>1</sub>	
88	P9 <sub>2</sub> /AN <sub>2</sub>	P9 <sub>2</sub> /AN <sub>2</sub>	P9 <sub>2</sub> /AN <sub>2</sub>	P9 <sub>2</sub> /AN <sub>2</sub>	P9 <sub>2</sub> /AN <sub>2</sub>	
89	P9 <sub>3</sub> /AN <sub>3</sub>	P9 <sub>3</sub> /AN <sub>3</sub>	P9 <sub>3</sub> /AN <sub>3</sub>	P9 <sub>3</sub> /AN <sub>3</sub>	P9 <sub>3</sub> /AN <sub>3</sub>	
90	P9 <sub>4</sub> /AN <sub>4</sub>	P9 <sub>4</sub> /AN <sub>4</sub>	P9 <sub>4</sub> /AN <sub>4</sub>	P9 <sub>4</sub> /AN <sub>4</sub>	P9 <sub>4</sub> /AN <sub>4</sub>	
91	P9 <sub>5</sub> /AN <sub>5</sub>	P9 <sub>5</sub> /AN <sub>5</sub>	P9 <sub>5</sub> /AN <sub>5</sub>	P9 <sub>5</sub> /AN <sub>5</sub>	P9 <sub>5</sub> /AN <sub>5</sub>	
92	P9 <sub>6</sub> /AN <sub>6</sub>	P9 <sub>6</sub> /AN <sub>6</sub>	P9 <sub>6</sub> /AN <sub>6</sub>	P9 <sub>6</sub> /AN <sub>6</sub>	P9 <sub>6</sub> /AN <sub>6</sub>	
93	P9 <sub>7</sub> /AN <sub>7</sub>	P9 <sub>7</sub> /AN <sub>7</sub>	P9 <sub>7</sub> /AN <sub>7</sub>	P9 <sub>7</sub> /AN <sub>7</sub>	P9 <sub>7</sub> /AN <sub>7</sub>	
94	P8 <sub>0</sub> /AN <sub>8</sub>	P8 <sub>0</sub> /AN <sub>8</sub>	P8 <sub>0</sub> /AN <sub>8</sub>	P8 <sub>0</sub> /AN <sub>8</sub>	P8 <sub>0</sub> /AN <sub>8</sub>	
95	P8 <sub>1</sub> /AN <sub>9</sub>	P8 <sub>1</sub> /AN <sub>9</sub>	P8 <sub>1</sub> /AN <sub>9</sub>	P8 <sub>1</sub> /AN <sub>9</sub>	P8 <sub>1</sub> /AN <sub>9</sub>	
96	P8 <sub>2</sub> /AN <sub>10</sub>	P8 <sub>2</sub> /AN <sub>10</sub>	P8 <sub>2</sub> /AN <sub>10</sub>	P8 <sub>2</sub> /AN <sub>10</sub>	P8 <sub>2</sub> /AN <sub>10</sub>	
97	P8 <sub>3</sub> /AN <sub>11</sub>	P8 <sub>3</sub> /AN <sub>11</sub>	P8 <sub>3</sub> /AN <sub>11</sub>	P8 <sub>3</sub> /AN <sub>11</sub>	P8 <sub>3</sub> /AN <sub>11</sub>	
98	AV <sub>SS</sub>	AV <sub>SS</sub>	AV <sub>SS</sub>	AV <sub>SS</sub>	AV <sub>SS</sub>	
99	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	V <sub>SS</sub>	

Notes: 1. For the PROM mode, see section 19, “Flash Memory (H8/539F)” and section 20, “Flash Memory (H8/539F S Mask model)”.

2. Pins marked NC should be left unconnected.

\*1: 12 V must not be applied to the S-mask model (single power source), as this will permanently damage the device.

**Table 1-2 Pin Assignments in Each Operating Mode (FP-112) (cont)**

No.	Expanded Minimum Modes		Expanded Maximum Modes		Single-Chip Mode	
	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7	Notes
100	P7 <sub>0</sub> /IRQ <sub>0</sub>	P7 <sub>0</sub> /IRQ <sub>0</sub>	P7 <sub>0</sub> /IRQ <sub>0</sub>	P7 <sub>0</sub> /IRQ <sub>0</sub>	P7 <sub>0</sub> /IRQ <sub>0</sub>	
101	P7 <sub>1</sub> /IRQ <sub>1</sub> / ADTRG	P7 <sub>1</sub> /IRQ <sub>1</sub> / ADTRG	P7 <sub>1</sub> /IRQ <sub>1</sub> / ADTRG	P7 <sub>1</sub> /IRQ <sub>1</sub> / ADTRG	P7 <sub>1</sub> /IRQ <sub>1</sub> / ADTRG	
102	P7 <sub>2</sub> /TXD <sub>1</sub>	P7 <sub>2</sub> /TXD <sub>1</sub>	P7 <sub>2</sub> /TXD <sub>1</sub>	P7 <sub>2</sub> /TXD <sub>1</sub>	P7 <sub>2</sub> /TXD <sub>1</sub>	
103	P7 <sub>3</sub> /RXD <sub>1</sub>	P7 <sub>3</sub> /RXD <sub>1</sub>	P7 <sub>3</sub> /RXD <sub>1</sub>	P7 <sub>3</sub> /RXD <sub>1</sub>	P7 <sub>3</sub> /RXD <sub>1</sub>	
104	P7 <sub>4</sub> /TXD <sub>2</sub>	P7 <sub>4</sub> /TXD <sub>2</sub>	P7 <sub>4</sub> /TXD <sub>2</sub>	P7 <sub>4</sub> /TXD <sub>2</sub>	P7 <sub>4</sub> /TXD <sub>2</sub>	
105	P7 <sub>5</sub> /RXD <sub>2</sub>	P7 <sub>5</sub> /RXD <sub>2</sub>	P7 <sub>5</sub> /RXD <sub>2</sub>	P7 <sub>5</sub> /RXD <sub>2</sub>	P7 <sub>5</sub> /RXD <sub>2</sub>	
106	P7 <sub>6</sub> /SCK <sub>1</sub> / PW <sub>1</sub>	P7 <sub>6</sub> /SCK <sub>1</sub> / PW <sub>1</sub>	P7 <sub>6</sub> /SCK <sub>1</sub> / PW <sub>1</sub>	P7 <sub>6</sub> /SCK <sub>1</sub> / PW <sub>1</sub>	P7 <sub>6</sub> /SCK <sub>1</sub> / PW <sub>1</sub>	
107	P7 <sub>7</sub> /SCK <sub>2</sub> / PW <sub>2</sub>	P7 <sub>7</sub> /SCK <sub>2</sub> / PW <sub>2</sub>	P7 <sub>7</sub> /SCK <sub>2</sub> / PW <sub>2</sub>	P7 <sub>7</sub> /SCK <sub>2</sub> / PW <sub>2</sub>	P7 <sub>7</sub> /SCK <sub>2</sub> / PW <sub>2</sub>	
108	P6 <sub>0</sub> /IRQ <sub>2</sub> / PW <sub>3</sub>	P6 <sub>0</sub> /IRQ <sub>2</sub> / PW <sub>3</sub>	P6 <sub>0</sub> /IRQ <sub>2</sub> / PW <sub>3</sub>	P6 <sub>0</sub> /IRQ <sub>2</sub> / PW <sub>3</sub>	P6 <sub>0</sub> /IRQ <sub>2</sub> / PW <sub>3</sub>	
109	P6 <sub>1</sub> /IRQ <sub>3</sub>	P6 <sub>1</sub> /IRQ <sub>3</sub>	P6 <sub>1</sub> /IRQ <sub>3</sub>	P6 <sub>1</sub> /IRQ <sub>3</sub>	P6 <sub>1</sub> /IRQ <sub>3</sub>	
110	P6 <sub>2</sub> /TCLK <sub>1</sub>	P6 <sub>2</sub> /TCLK <sub>1</sub>	P6 <sub>2</sub> /TCLK <sub>1</sub>	P6 <sub>2</sub> /TCLK <sub>1</sub>	P6 <sub>2</sub> /TCLK <sub>1</sub>	
111	P6 <sub>3</sub> /TCLK <sub>2</sub>	P6 <sub>3</sub> /TCLK <sub>2</sub>	P6 <sub>3</sub> /TCLK <sub>2</sub>	P6 <sub>3</sub> /TCLK <sub>2</sub>	P6 <sub>3</sub> /TCLK <sub>2</sub>	
112	P6 <sub>4</sub> /TCLK <sub>3</sub>	P6 <sub>4</sub> /TCLK <sub>3</sub>	P6 <sub>4</sub> /TCLK <sub>3</sub>	P6 <sub>4</sub> /TCLK <sub>3</sub>	P6 <sub>4</sub> /TCLK <sub>3</sub>	

- Notes: 1. For the PROM mode, see section 19, “Flash Memory (H8/539F)” and section 20, “Flash Memory (H8/539F S Mask model)”.
2. Pins marked NC should be left unconnected.

(2) **Pin Functions:** Table 1-3 indicates the function of each pin.

**Table 1-3 Pin Functions**

Type	Symbol	Pin No.	I/O	Name and Function
Power	$V_{CC}$	1, 44, 76	Input	<b>Power:</b> Connected to the power supply (4.5 V to 5.5 V). Connect all $V_{CC}$ pins to the system power supply (4.5 V to 5.5 V). The chip will not operate if any $V_{CC}$ pin is left unconnected.
	$V_{SS}$	10, 26, 35, 53, 73, 99	Input	<b>Ground:</b> Connected to ground (0 V). Connect all $V_{SS}$ pins to the 0-V system power supply. The chip will not operate if any $V_{SS}$ pin is left unconnected.
Clock	XTAL	75	Input	<b>Crystal:</b> Connected to a crystal resonator. The frequency should be equal to the desired system clock frequency ( $\phi$ ). If an external clock is input at the EXTAL pin, input a complementary clock at XTAL.
	EXTAL	74	Input	<b>Crystal/external clock:</b> Connected to a crystal resonator or external clock. The frequency should be equal to the desired system clock frequency ( $\phi$ ). An external clock can also be input from the EXTAL pin. See section 8.2, "Oscillator Circuit" for examples of connections at XTAL and EXTAL.
	$\phi$	69	Output	<b>System clock:</b> Supplies the system clock ( $\phi$ ) to peripheral devices.
System control	$\overline{BACK}$	68	Output	<b>Bus request acknowledge:</b> Indicates that the bus right has been granted to an external device. A device requesting the bus sends a $\overline{BREQ}$ signal to the microcontroller. The microcontroller replies with a $\overline{BACK}$ signal.
	$\overline{BREQ}$	67	Input	<b>Bus request:</b> Sent by an external device to the microcomputer chip to request the bus right. Bus acquisition should be confirmed with the $\overline{BACK}$ signal.
	$\overline{STBY}$	70	Input	<b>Standby:</b> Input pin for transition to the hardware standby mode (a power-down state).
	$\overline{RES}$	71	Input	<b>Reset:</b> Input pin for transition to the reset state.

**Table 1-3 Pin Functions (cont)**

Type	Symbol	Pin No.	I/O	Name and Function
Address bus	A <sub>19</sub> –A <sub>0</sub>	65–54, 52–45	Output	<b>Address bus:</b> Address output pins.
Data bus	D <sub>15</sub> –D <sub>0</sub> *	43–36, 34–27	Input/ Output	<b>Data bus:</b> Sixteen-bit bidirectional data bus.
Bus control signals	$\overline{\text{WAIT}}$	66	Input	<b>Wait:</b> Requests insertion of wait states (T <sub>W</sub> ) in external-device access cycles by the CPU; used for interfacing to low-speed external devices.
	$\overline{\text{AS}}$	77	Output	<b>Address strobe:</b> Indicates valid address output on the address bus during external-device access.
	$\overline{\text{RD}}$	78	Output	<b>Read:</b> Indicates reading of data from the data bus during external-device access. The CPU latches read data at the rising edge of $\overline{\text{RD}}$ .
	$\overline{\text{HWR}}$	79	Output	<b>High write:</b> Indicates output of data on the upper data bus (D <sub>15</sub> to D <sub>8</sub> ) during external-device access.
	$\overline{\text{LWR}}$	80	Output	<b>Low write:</b> Indicates output of data on the lower data bus (D <sub>7</sub> to D <sub>0</sub> ) during external-device access.
Interrupt signals	NMI	72	Input	<b>Nonmaskable interrupt:</b> Nonmaskable interrupt request signal. The input edge can be selected in the NMI control register (NMICR).
	$\overline{\text{IRQ}}_0$	100	Input	<b>Interrupt request 0 to 3:</b> Maskable interrupt request signals. The type of input can be selected in the IRQ control register (IRQCR).
	$\overline{\text{IRQ}}_1$	101		
	$\overline{\text{IRQ}}_2$	108		
	$\overline{\text{IRQ}}_3$	109		

Notes: \* When the external bus space uses an 8-bit bus width, D15 to D8 are enabled and D7 to D0 are disabled.



**Table 1-3 Pin Functions (cont)**

Type	Symbol	Pin No.	I/O	Name and Function																																																																		
Operating mode control	MD <sub>2</sub> *1	83	Input	<b>Mode 2 to mode 0:</b> Input pins for setting the operating mode. The following table lists the operating modes and bus widths.																																																																		
	MD <sub>1</sub>	82																																																																				
	MD <sub>0</sub>	81																																																																				
<table border="1"> <thead> <tr> <th colspan="3">Pin Settings</th> <th rowspan="2">Operating Mode</th> <th rowspan="2">H8/500 CPU Operating Mode</th> <th rowspan="2">On-Chip ROM</th> <th rowspan="2">External Bus</th> </tr> <tr> <th>MD<sub>2</sub>*1</th> <th>MD<sub>1</sub></th> <th>MD<sub>0</sub></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>Do not use</td> <td></td> <td></td> <td></td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>Mode 1</td> <td>Expanded minimum</td> <td>Disabled</td> <td>16 bits</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>Mode 2</td> <td>Expanded minimum</td> <td>Enabled</td> <td>8 bits</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>Mode 3</td> <td>Expanded maximum</td> <td>Disabled</td> <td>16 bits</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>Mode 4</td> <td>Expanded maximum</td> <td>Enabled</td> <td>16 bits</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>Mode 5</td> <td>Expanded maximum</td> <td>Disabled</td> <td>16 bits</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>Mode 6</td> <td>Expanded minimum</td> <td>Disabled</td> <td>16 bits</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>Mode 7</td> <td>Single chip maximum</td> <td>Enabled</td> <td>—</td> </tr> </tbody> </table>					Pin Settings			Operating Mode	H8/500 CPU Operating Mode	On-Chip ROM	External Bus	MD <sub>2</sub> *1	MD <sub>1</sub>	MD <sub>0</sub>	0	0	0	Do not use				0	0	1	Mode 1	Expanded minimum	Disabled	16 bits	0	1	0	Mode 2	Expanded minimum	Enabled	8 bits	0	1	1	Mode 3	Expanded maximum	Disabled	16 bits	1	0	0	Mode 4	Expanded maximum	Enabled	16 bits	1	0	1	Mode 5	Expanded maximum	Disabled	16 bits	1	1	0	Mode 6	Expanded minimum	Disabled	16 bits	1	1	1	Mode 7	Single chip maximum	Enabled	—
Pin Settings			Operating Mode	H8/500 CPU Operating Mode	On-Chip ROM	External Bus																																																																
MD <sub>2</sub> *1	MD <sub>1</sub>	MD <sub>0</sub>																																																																				
0	0	0	Do not use																																																																			
0	0	1	Mode 1	Expanded minimum	Disabled	16 bits																																																																
0	1	0	Mode 2	Expanded minimum	Enabled	8 bits																																																																
0	1	1	Mode 3	Expanded maximum	Disabled	16 bits																																																																
1	0	0	Mode 4	Expanded maximum	Enabled	16 bits																																																																
1	0	1	Mode 5	Expanded maximum	Disabled	16 bits																																																																
1	1	0	Mode 6	Expanded minimum	Disabled	16 bits																																																																
1	1	1	Mode 7	Single chip maximum	Enabled	—																																																																
Serial communication interface (SCI)	TXD <sub>1</sub>	102	Output	<b>Transmit data 1, 2, and 3:</b> Serial transmit data output pins for SCI1, SCI2, and SCI3.																																																																		
	TXD <sub>2</sub>	104																																																																				
	TXD <sub>3</sub>	68																																																																				
	RXD <sub>1</sub>	103	Input	<b>Receive data 1, 2, and 3:</b> Serial receive data input pins for SCI1, SCI2, and SCI3.																																																																		
	RXD <sub>2</sub>	105																																																																				
	RXD <sub>3</sub>	67																																																																				
	SCK <sub>1</sub>	106	Input/ Output	<b>Serial clock 1, 2, and 3:</b> Serial clock input/output pins for SCI1, SCI2, and SCI3. Used for input and output of the serial clock in clocked synchronous mode, and of the SCI operating clock in asynchronous mode.																																																																		
	SCK <sub>2</sub>	107																																																																				
	SCK <sub>3</sub>	65																																																																				
PWM timer	PW <sub>1</sub>	62	Output	<b>PWM1, PWM2, and PWM3 output:</b> Output pins for PWM1, PWM2, and PWM3.																																																																		
		106																																																																				
	PW <sub>2</sub>	63 107	Output																																																																			
	PW <sub>3</sub>	64 108	Output																																																																			

\*1: 12 V must not be applied to the S-mask model (single power source), as this will permanently damage the device.

**Table 1-3 Pin Functions (cont)**

Type	Symbol	Pin No.	I/O	Name and Function
16-bit integrated-timer pulse unit (IPU)	T1IOC <sub>1</sub>	2	Input/ Output	<b>Input capture/output compare 1 to 4 (channel 1):</b> Input capture input or output compare output pins for IPU channel 1.
	T1IOC <sub>2</sub>	3		
	T1IOC <sub>3</sub>	4		
	T1IOC <sub>4</sub>	5		
	T1OC <sub>1</sub>	20	Output	<b>Output compare 1 to 4 (channel 1):</b> Dedicated output compare output pins for IPU channel 1.
	T1OC <sub>2</sub>	21		
	T1OC <sub>3</sub>	22		
	T1OC <sub>4</sub>	23		
	T2IOC <sub>1</sub>	6	Input/ Output	<b>Input capture/output compare 1 and 2 (channel 2):</b> Input capture input or output compare output pins for IPU channel 2.
	T2IOC <sub>2</sub>	7		
	T2OC <sub>1</sub>	24	Output	<b>Output compare 1 and 2 (channel 2):</b> Dedicated output compare output pins for IPU channel 2.
	T2OC <sub>2</sub>	25		
	T3IOC <sub>1</sub>	8	Input/ Output	<b>Input capture/output compare 1 and 2 (channel 3):</b> Input capture input or output compare output pins for IPU channel 3.
	T3IOC <sub>2</sub>	9		
	T3OC <sub>1</sub>	67	Output	<b>Output compare 1 and 2 (channel 3):</b> Dedicated output compare output pins for IPU channel 3.
	T3OC <sub>2</sub>	68		
	T4IOC <sub>1</sub>	11	Input/ Output	<b>Input capture/output compare 1 and 2 (channel 4):</b> Input capture input or output compare output pins for IPU channel 4.
	T4IOC <sub>2</sub>	12		
	T4OC <sub>1</sub>	62	Output	<b>Output compare 1 and 2 (channel 4):</b> Dedicated output compare output pins for IPU channel 4.
	T4OC <sub>2</sub>	63		
	T5IOC <sub>1</sub>	13	Input/ Output	<b>Input capture/output compare 1 and 2 (channel 5):</b> Input capture input or output compare output pins for IPU channel 5.
	T5IOC <sub>2</sub>	14		
	T5OC <sub>1</sub>	64	Output	<b>Output compare 1 and 2 (channel 5):</b> Dedicated output compare output pins for IPU channel 5.
	T5OC <sub>2</sub>	65		
	T6IOC <sub>1</sub>	15	Input/ Output	<b>Input capture/output compare 1 and 2 (channel 6):</b> Input capture input or output compare output pins for IPU channel 6.
	T6IOC <sub>2</sub>	16		
	T7IOC <sub>1</sub>	17	Input/ Output	<b>Input capture/output compare 1 and 2 (channel 7):</b> Input capture input or output compare output pins for IPU channel 7.
	T7IOC <sub>2</sub>	18		
	TCLK <sub>1</sub>	110	Input	<b>Timer clock 1 to 3 (all channels):</b> IPU external clock input pins. All channels can select these clock inputs.
	TCLK <sub>2</sub>	111		
	TCLK <sub>3</sub>	112		

**Table 1-3 Pin Functions (cont)**

Type	Symbol	Pin No.	I/O	Name and Function
A/D converter	AN <sub>11</sub> –AN <sub>0</sub>	97–86	Input	<b>Analog input 11 to 0:</b> Analog input pins for the A/D converter.
	V <sub>REF</sub>	85	Input	<b>Reference power supply (V<sub>REF</sub> ≤ AV<sub>CC</sub>):</b> Input pin for the A/D converter's reference voltage. Apply a voltage corresponding to the A/D conversion full-scale value.
	AV <sub>CC</sub>	84	Input	<b>Analog power supply:</b> Power supply pin for analog circuits in the A/D converter. Connect to a stable +5-V analog power supply separate from the other power supply pins.
	AV <sub>SS</sub>	98	Input	<b>Analog ground:</b> Ground pin for analog circuits in the A/D converter. Connect to a stable 0-V analog power supply separate from the other power supply pins.
	$\overline{\text{ADTRG}}$	101	Input	<b>A/D trigger:</b> Trigger input for starting A/D conversion. The A/D conversion start time is specified by the falling edge of $\overline{\text{ADTRG}}$ .
Watchdog timer	$\overline{\text{RESO}}$	19	Output	<b>Reset output:</b> If reset output is selected, a low pulse is output for 132 cycles when the watchdog timer overflows. $\overline{\text{RESO}}$ is an open-drain output pin and should be pulled up to V <sub>CC</sub> (+5 V) externally, regardless of whether reset output is selected or not. The S mask model does not have this function.
Flash Memory	Dual power source type V <sub>PP</sub>	19	Input	<b>Program power supply:</b> The flash memory programming power supply pin. Connect to the programming power supply (12 V) when programming and erasing.
	Single power source type (S-mask model) FWE* <sup>1</sup>			<b>Flash write enable:</b> Enables program mode setting.
I/O ports	P <sub>17</sub> – P <sub>10</sub>	43 – 36	Input/ Output	<b>Port 1:</b> 8-bit input/output port. The direction of each bit can be selected in the port 1 data direction register (P1DDR).
	P <sub>27</sub> – P <sub>20</sub>	34 – 27	Input/ Output	<b>Port 2:</b> 8-bit input/output port. Input or output can be set for each bit in the port 2 data direction register (P2DDR).
	P <sub>35</sub> – P <sub>30</sub>	25 – 20	Input/ Output	<b>Port 3:</b> 6-bit input/output port. Input or output can be set for each bit in the port 3 data direction register (P3DDR). LEDs can be driven directly (10-mA sink).

\*1: 12 V must not be applied to the S-mask model (single power source), as this will permanently damage the device.

**Table 1-3 Pin Functions (cont)**

Type	Symbol	Pin No.	I/O	Name and Function
I/O ports	P4 <sub>7</sub> – P4 <sub>0</sub>	18 – 11	Input/ Output	<b>Port 4:</b> 8-bit input/output port with Schmitt-trigger inputs. Input or output can be set for each bit in the port 4 data direction register (P4DDR).
	P5 <sub>7</sub> – P5 <sub>0</sub>	9 – 2	Input/ Output	<b>Port 5:</b> 8-bit input/output port with Schmitt-trigger inputs. Input or output can be set for each bit in the port 5 data direction register (P5DDR). LEDs can be driven directly (10-mA sink).
	P6 <sub>4</sub> – P6 <sub>0</sub>	112 – 108	Input/ Output	<b>Port 6:</b> 5-bit input/output port. Input or output can be set for each bit in the port 6 data direction register (P6DDR).
	P7 <sub>7</sub> – P7 <sub>0</sub>	107 – 100	Input/ Output	<b>Port 7:</b> 8-bit input/output port. Input or output can be set for each bit in the port 7 data direction register (P7DDR).
	P8 <sub>3</sub> – P8 <sub>0</sub>	97 – 94	Input	<b>Port 8:</b> 4-bit input port.
	P9 <sub>7</sub> – P9 <sub>0</sub>	93 – 86	Input	<b>Port 9:</b> 8-bit input port.
	PA <sub>6</sub> – PA <sub>0</sub>	68 – 62	Input/ Output	<b>Port A:</b> 7-bit input/output port. Input or output can be set for each bit in the port A data direction register (PADDDR).
	PB <sub>7</sub> – PB <sub>0</sub>	61 – 54	Input/ Output	<b>Port B:</b> 8-bit input/output port with MOS input pull-up transistors. Input or output can be set for each bit in the port B data direction register (PBDDR).
	PC <sub>7</sub> – PC <sub>0</sub>	52 – 45	Input/ Output	<b>Port C:</b> 8-bit input/output port with MOS input pull-up transistors. Input or output can be set for each bit in the port C data direction register (PCDDR).

## 1.4 Notes on S-Mask Model (Single Power Source)

There are two models of the H8/539F with on-chip flash memory: a dual power source model and a single power source (S-mask) model. Points to be noted when using the H8/539F single power source S-mask model are given below.

### 1.4.1 Voltage Application

12 V must not be applied to the S-mask model (single power source), as this will permanently damage the device.

The flash memory programming power source for the S-mask model (single power source) is  $V_{CC}$ .

The programming power source for the dual power source model was the  $V_{PP}$  pin (12 V), but there is no  $V_{PP}$  pin in the single power source model. In the S-mask model the FWE pin is provided at the same pin position as the  $V_{PP}$  pin in the dual power source model, but FWE is not a power source pin—it is used to control flash memory write enabling.

Also, in boot mode, 12 V must be applied to the  $MD_2$  pin in the dual power source model, but this is not necessary in the S-mask model (single power source).

The maximum rating of the FWE and  $MD_2$  pins in the S-mask model (single power source) is  $V_{CC} + 0.3$  V. Applying a voltage in excess of the maximum rating will permanently damage the device.



Do not select the HN28F101 programmer setting for the S-mask model (single power source). If this setting is made by mistake, 12.0 V may be applied to the FWE pin, causing permanent damage to the device.

When using a PROM programmer to program the on-chip flash memory in the S-mask model (single power source), use a PROM programmer that supports Hitachi microcomputer device types with 128-kbyte on-chip flash memory.

### 1.4.2 Product Type Names and Markings

Table 1-4 shows examples of product type names and markings for the H8/539F (dual power source model) and H8/539F S-mask model (single power source), and the differences in flash memory programming power source.

**Table 1-4 Differences in H8/539F and H8/539F S-Mask Model Markings**

	Dual Power Source Model: H8/539F	Single Power Source Model: H8/539F S-Mask Model (Preliminary)
Product type name	HD64F5398F16	HD64F5398SF16
Sample markings		
		"S" is printed above the type name
Flash memory programming power source	$V_{PP}$ power source (12.0 $\pm$ 0.6 V)	$V_{CC}$ power source (5.0 $\pm$ 10%)

### 1.4.3 Differences in S-Mask Model

Table 1-5 shows the differences between the H8/539F (dual power source model) and H8/539F S-mask model (single power source model).

**Table 1-5 Differences between H8/539F and H8/539F S-Mask Model**

Item	Dual Power Source Model: H8/539F	Single Power Source Model: H8/539F S-Mask Model (Preliminary)
Program/erase voltage	12 V must be applied to $V_{PP}$ power source pin from off-chip	12 V application not required. $V_{CC}$ single power source programming
$V_{PP}$ (FWE) pin function	Dual function as $V_{PP}$ power source and RESO output	FWE pin function only. RESO pin function eliminated
Programming modes	<ul style="list-style-type: none"> <li>PROM mode</li> <li>On-board: Boot mode</li> <li>User program mode</li> </ul>	←
Operating modes allowing on-board programming	Modes 4 and 7	←
On-board programming unit	Byte-unit programming	32-byte-unit programming
Programming with PROM programmer	Select Hitachi stand-alone flash memory HN28F101 setting	Special programming mode setting required. Use of PROM programmer that supports Hitachi microcomputer device types with 128-kbyte on-chip flash memory (128-byte-unit fast page programming)

**Table 1-5 Differences between H8/539F and H8/539F S-Mask Model (cont)**

Item	Dual Power Source Model: H8/539F	Single Power Source Model: H8/539F S-Mask Model (Preliminary)															
Boot mode setting method	MD <sub>2</sub> = V <sub>PP</sub> /RESO = 12 V Reset release	FWE = 1 input Mode 4: Set mode 5 Mode 7: Set mode 6 Reset release															
		Pin															
		<table border="1"> <thead> <tr> <th>Mode</th> <th>MD2</th> <th>MD1</th> <th>MD0</th> <th>FWE</th> </tr> </thead> <tbody> <tr> <td>Mode 4</td> <td>"1"</td> <td>"0"</td> <td style="background-color: #cccccc;">"1"</td> <td style="background-color: #cccccc;">"1"</td> </tr> <tr> <td>Mode 7</td> <td>"1"</td> <td>"1"</td> <td style="background-color: #cccccc;">"0"</td> <td style="background-color: #cccccc;">"1"</td> </tr> </tbody> </table>	Mode	MD2	MD1	MD0	FWE	Mode 4	"1"	"0"	"1"	"1"	Mode 7	"1"	"1"	"0"	"1"
Mode	MD2	MD1	MD0	FWE													
Mode 4	"1"	"0"	"1"	"1"													
Mode 7	"1"	"1"	"0"	"1"													
User program mode setting method	V <sub>PP</sub> /RESO = 12 V	FWE = "1"															
Programming mode timing																	
Prewrite processing	Required before erasing	Not required															
Multiple-block erase	Multiple blocks can be erased simultaneously (verification performed block by block, only unerased blocks re-erased)	Block-unit erasing. Multiple blocks cannot be erased simultaneously (Erase procedure is also different)															
Programming processing	Block corresponding to programming address must be set in EBR1/2 registers before programming	Settings at left not required (Programming procedure is also different)															

**Table 1-5 Differences between H8/539F and H8/539F S-Mask Model (cont)**

Item	Dual Power Source Model: H8/539F	Single Power Source Model: H8/539F S-Mask Model (Preliminary)
EBR register configuration	EBR1, EBR2 (See section 19 for bit functions)	EBR1 (See section 20 for bit functions)
Memory map (block configuration) ■ Shadow area	<p>Page 0: LB7 (12 KB), SB0 (512 B), SB1 (512 B), SB2 (512 B), SB3 (512 B), SB4 (512 B), SB5 (512 B), SB6 (512 B), SB7 (512 B) — 16 KB</p> <p>Page 1 to Page 2: LB6 (16 KB), LB5 (16 KB), LB4 (16 KB), LB3 (16 KB), LB2 (16 KB), LB1 (16 KB), LB0 (16 KB) — 128 KB</p>	<p>Page 0: EB0 (1 KB), EB1 (1 KB), EB2 (1 KB), EB3 (1 KB), EB4 (12 KB) — 16 KB</p> <p>Page 1 to Page 2: EB5 (32 KB), EB6 (32 KB), EB7 (32 KB) — 128 KB</p>
	<p>Block configuration: 16 blocks            16 KB x 7: LB0 to LB6            12 KB x 1: LB7            512 B x 8: SB0 to SB7</p>	<p>Block configuration: 8 blocks            32 KB x 3: EB5 to EB7            28 KB x 1: EB4            1 KB x 4: EB0 to EB3</p>
Reset during operation	<p>Drive <math>\overline{\text{RES}}</math> pin low for at least 6 system clock cycles (<math>6\phi</math>)            (<math>\overline{\text{RES}}</math> pulse width <math>t_{\text{RESW}} = \text{min. } 6.0 \text{ tcy}</math>)</p>	<p>Drive <math>\overline{\text{RES}}</math> pin low for at least 20 system clock cycles (<math>20\phi</math>)            (<math>\overline{\text{RES}}</math> pulse width <math>t_{\text{RESW}} = \text{min. } 20 \text{ tcy}</math>)</p>



**Table 1-5 Differences between H8/539F and H8/539F S-Mask Model (cont)**

Item	Dual Power Source Model: H8/539F	Single Power Source Model: H8/539F S-Mask Model (Preliminary)																																
FLMCR register	<table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>V<sub>PP</sub></td><td>V<sub>PP</sub>E</td><td>—</td><td>—</td><td>EV</td><td>PV</td><td>E</td><td>P</td> </tr> </table> <p>(See section 19 for bit functions)</p>	7	6	5	4	3	2	1	0	V <sub>PP</sub>	V <sub>PP</sub> E	—	—	EV	PV	E	P	<table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>FWE</td><td>SWE</td><td>ESU</td><td>PSU</td><td>EV</td><td>PV</td><td>E</td><td>P</td> </tr> </table> <p>(See section 20 for bit functions)</p>	7	6	5	4	3	2	1	0	FWE	SWE	ESU	PSU	EV	PV	E	P
7	6	5	4	3	2	1	0																											
V <sub>PP</sub>	V <sub>PP</sub> E	—	—	EV	PV	E	P																											
7	6	5	4	3	2	1	0																											
FWE	SWE	ESU	PSU	EV	PV	E	P																											
RAM emulation block configuration	<p>On-chip RAM</p> <p>Flash memory</p>	<p>On-chip RAM</p> <p>Flash memory</p>																																
FLMER register	<table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>OVLP</td><td>—</td><td>—</td><td>—</td><td>A11E</td><td>A10E</td><td>A9E</td><td>—</td> </tr> </table> <p>Bit 1 = A9E bit (See section 19 for bit functions)</p>	7	6	5	4	3	2	1	0	OVLP	—	—	—	A11E	A10E	A9E	—	<table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>OVLP</td><td>—</td><td>—</td><td>—</td><td>—</td><td>A10E</td><td>A9E</td><td>—</td> </tr> </table> <p>Bit 1 = Reserved (See section 20 for bit functions)</p>	7	6	5	4	3	2	1	0	OVLP	—	—	—	—	A10E	A9E	—
7	6	5	4	3	2	1	0																											
OVLP	—	—	—	A11E	A10E	A9E	—																											
7	6	5	4	3	2	1	0																											
OVLP	—	—	—	—	A10E	A9E	—																											
RAMCR register	<table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>RAME1</td><td>—</td><td>RAME2</td><td>—</td><td>—</td><td>RAM2</td><td>RAM1</td><td>RAM0</td> </tr> </table> <p>Bit 0 = RAM0 bit (See sections 17 and 19 for bit functions)</p>	7	6	5	4	3	2	1	0	RAME1	—	RAME2	—	—	RAM2	RAM1	RAM0	<table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td>RAME1</td><td>—</td><td>RAME2</td><td>—</td><td>—</td><td>RAM2</td><td>RAM1</td><td>—</td> </tr> </table> <p>Bit 0 = Reserved (See sections 18 and 20 for bit functions)</p>	7	6	5	4	3	2	1	0	RAME1	—	RAME2	—	—	RAM2	RAM1	—
7	6	5	4	3	2	1	0																											
RAME1	—	RAME2	—	—	RAM2	RAM1	RAM0																											
7	6	5	4	3	2	1	0																											
RAME1	—	RAME2	—	—	RAM2	RAM1	—																											
Details concerning flash memory	See section 19, Flash Memory (H8/539F: Dual Power System) (V <sub>PP</sub> = 12 V)	See section 20, Flash Memory (H8/539F, S-Mask Model, Single Power Source) (Programming/erasing procedure, etc., is also different)																																
Electrical characteristics	See section 22, Electrical Characteristics (H8/539F)	See section 23, Electrical Characteristics (H8/539F S-Mask Model) (Wait times during programming/erasing, etc., are also different)																																
List of registers	See Appendix C, Registers, (1) H8/539F	See Appendix C, Registers, (2) H8/539F S-Mask Model																																

# Section 2 CPU

## 2.1 Overview

The H8/539F has the H8/500 CPU, which is common to all chips in the H8/500 Family. The H8/500 CPU is a high-speed central processing unit that is designed for realtime control and supports a large address space. Its architecture features eight general registers, 16-bit internal data paths, and an optimized instruction set.

The H8/500 CPU is suitable for control of a wide range of medium-scale office and industrial equipment.

Section 2 summarizes the CPU architecture, instruction set, and operation.

### 2.1.1 Features

The main features of the H8/500 CPU are listed below.

- General-register machine
  - Eight 16-bit general registers
  - Seven control registers (two 16-bit registers, five 8-bit registers)
- High-speed operation: 16 MHz maximum clock rate
  - At 16 MHz a register-register add operation takes only 125 ns.
- Maximum address space: 1 Mbyte
  - Managed in 64-kbyte pages
  - Four pages available simultaneously: code page, stack page, data page, and extended page.

The CPU architecture supports up to 16 Mbytes, but the chip has only enough pins to address 1 Mbyte.

- Two CPU operating modes
  - Minimum mode: 64-kbyte address space
  - Maximum mode: 1-Mbyte address space
- Highly orthogonal instruction set

Addressing modes and data sizes can be specified independently within each instruction.

- Register and memory addressing modes

Register-register and register-memory (or memory-register) operations are supported.

- Instruction set optimized for C language

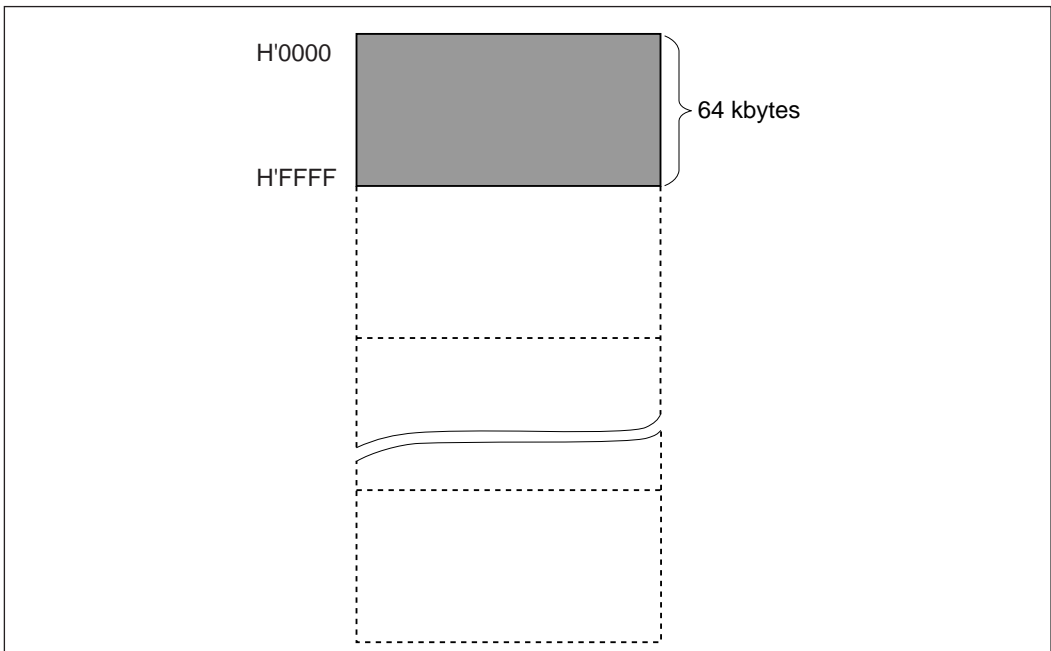
In addition to the general registers and orthogonal instruction set, the CPU has special short formats for frequently-used instructions and addressing modes.

### 2.1.2 Address Space

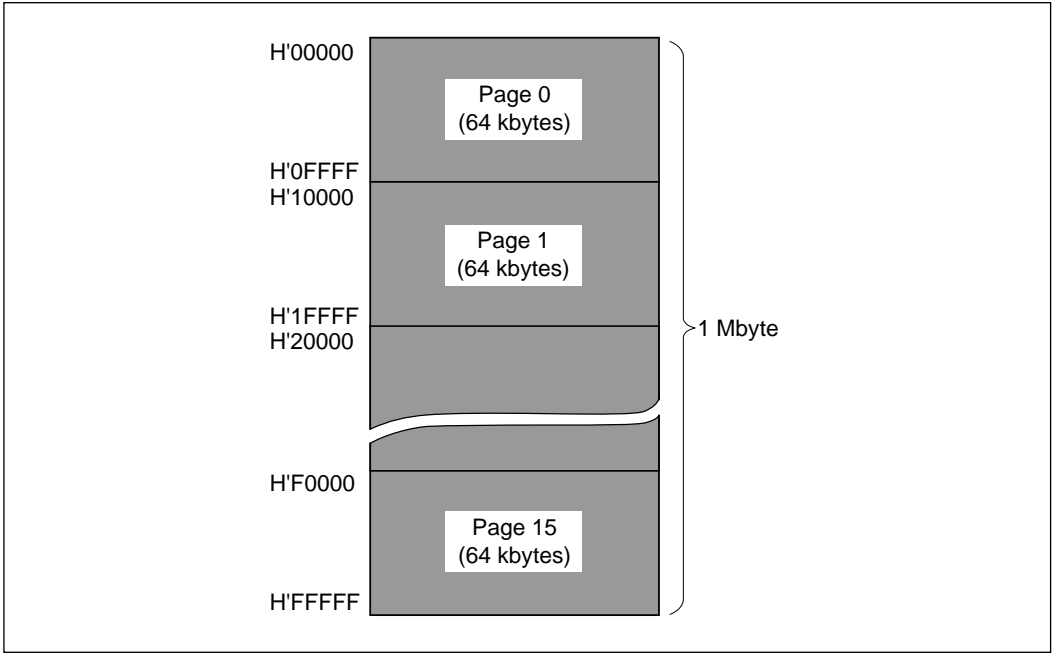
The H8/500 CPU has different address spaces in its two operating modes, the minimum mode and maximum mode. The CPU operating mode is selected by the input at the mode pins ( $MD_2$  to  $MD_0$ ) at a reset. Table 2-1 summarizes the CPU operating modes. Figure 2-1 shows a memory map for the minimum mode. Figure 2-2 shows a memory map for the maximum mode.

**Table 2-1 CPU Operating Modes**

Operating Mode	Features
Minimum mode	Maximum combined size of program area and data area: 64 kbytes
Maximum mode	Maximum combined size of program area and data area: 1 Mbyte



**Figure 2-1 Memory Map in Minimum Mode**



**Figure 2-2 Memory Map in Maximum Mode**

### 2.1.3 Programming Model

Figure 2-3 shows a programming model of the H8/500 CPU.

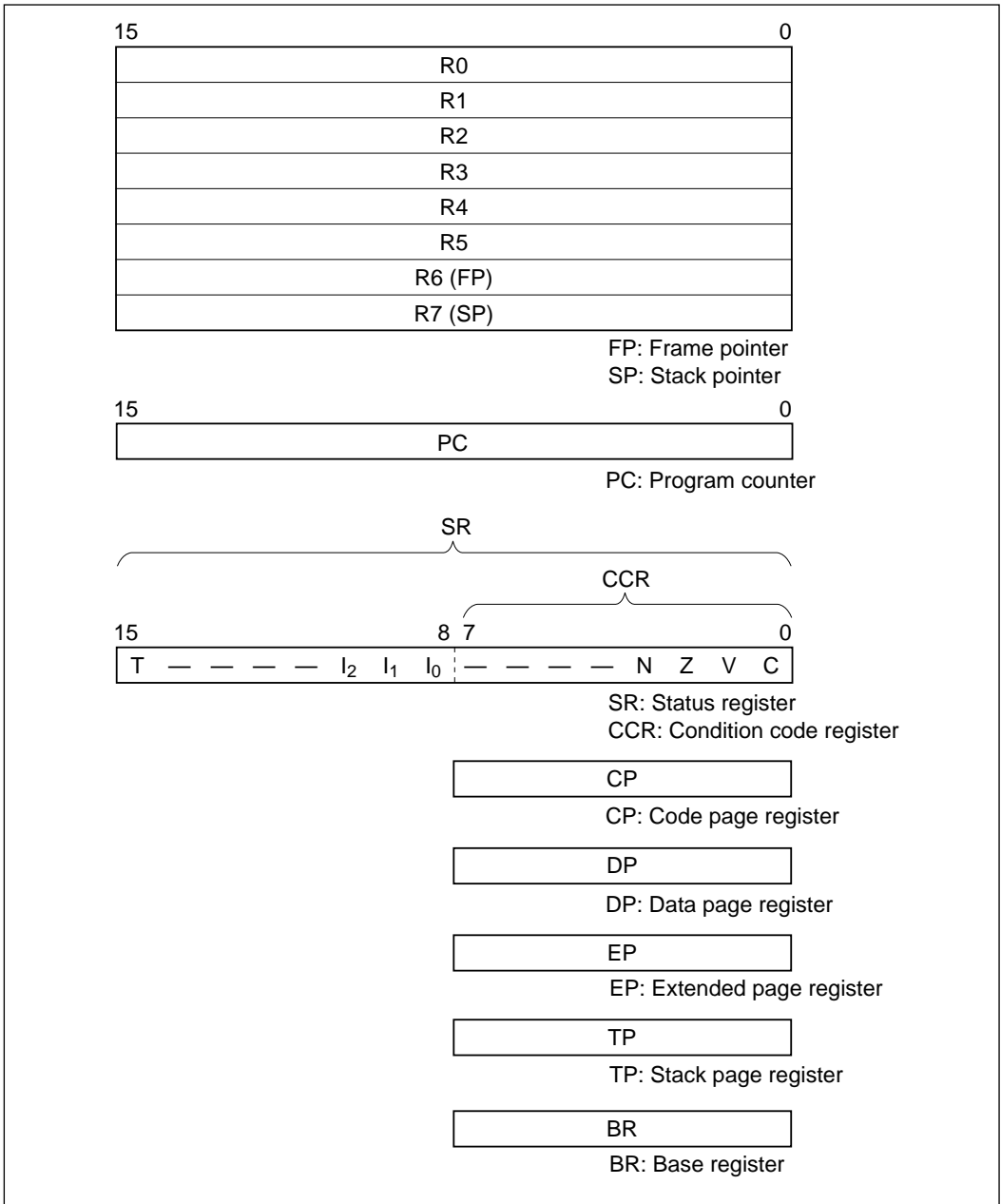


Figure 2-3 Programming Model

## 2.2 General Registers

The H8/500 CPU has eight 16-bit general registers.

The general registers are described next.

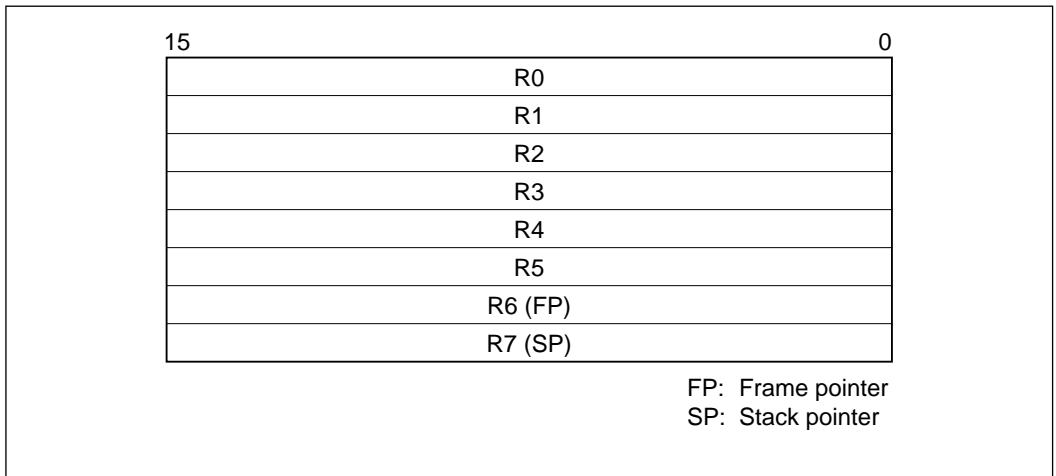
### 2.2.1 Overview

All eight of the general registers are functionally alike; there is no distinction between data registers and address registers. When these registers are accessed as data registers, either byte or word size can be selected.

When these registers are accessed as address registers, word size is implicitly assumed.

### 2.2.2 Register Configuration

Figure 2-4 shows the general register configuration.



**Figure 2-4 General Register Configuration**

### 2.2.3 Stack Pointer

R7 functions as the stack pointer (SP), and is used implicitly in exception handling and subroutine calls. It is also used implicitly in pre-decrement or post-increment mode by the LDM and STM instructions, which load and store multiple registers on the stack.

### 2.2.4 Frame Pointer

R6 functions as a frame pointer (FP). The LINK and UNLK instructions use R6 implicitly to reserve or release a stack frame.

## 2.3 Control Registers

The H8/500 CPU has two control registers.

The control registers are described next.

### 2.3.1 Overview

The control registers include a 16-bit program counter and a 16-bit status register.

The program counter and status register are described next.

### 2.3.2 Register Configuration

Figure 2-5 illustrates the program counter and status register.

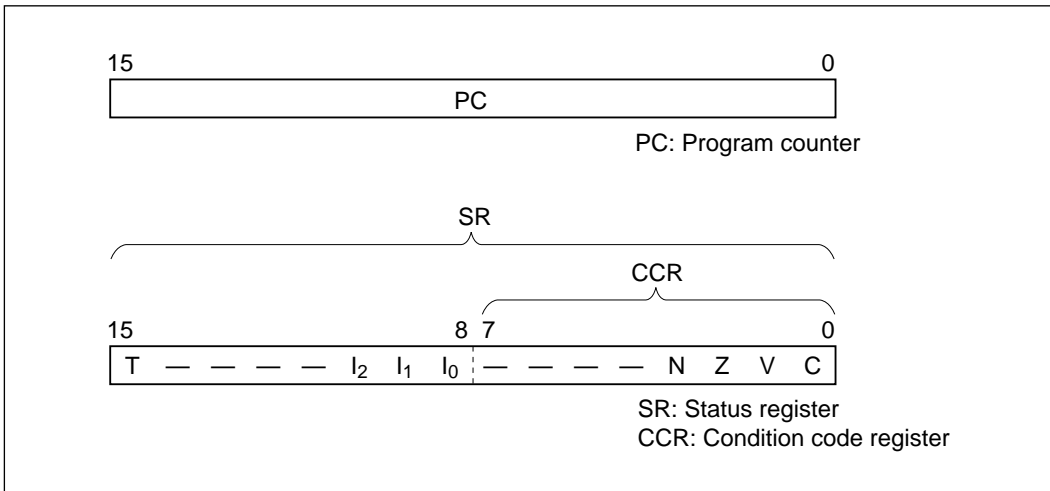


Figure 2-5 Program Counter and Status Register

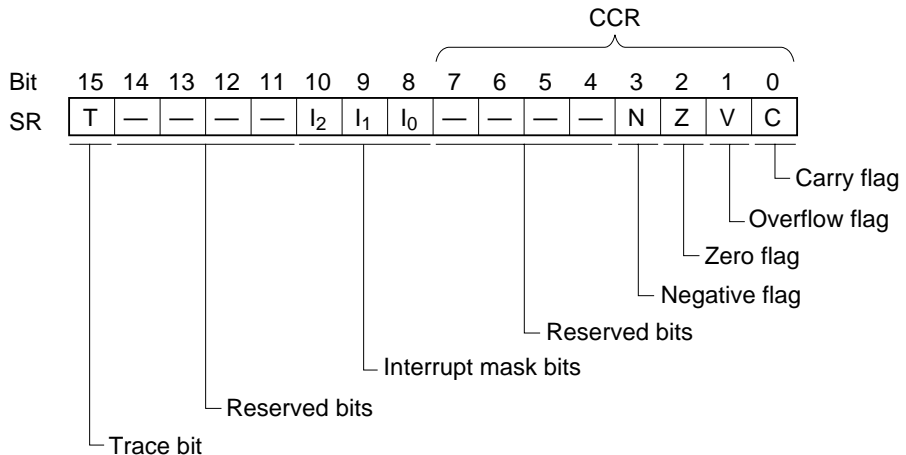
### 2.3.3 Program Counter

The 16-bit program counter (PC) indicates the address of the next instruction the CPU will execute.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC																

### 2.3.4 Status Register

The 16-bit status register (SR) contains status flags that indicate the internal state of the CPU.



The lower eight bits of the status register are referred to as the condition code register (CCR). Byte access to the CCR is possible.

(1) **Bit 15—Trace (T):** Selects trace mode.

#### Bit 15

T	Description
0	Instructions are executed in succession (initial mode after reset)
1	Trace exception handling starts after each instruction (trace mode)

For information about trace exception handling, see section 4.4, “Trace.”

(2) **Bits 14 to 11—Reserved:** Read-only bits, always read as 0.

(3) **Bits 10 to 8—Interrupt mask (I<sub>2</sub>, I<sub>1</sub>, I<sub>0</sub>):** These bits indicate the interrupt request mask level (0 to 7) of the program that is currently executing. Table 2-2 explains the interrupt request mask levels.



**Table 2-2 Interrupt Mask Levels**

Interrupt Mask			Level	Priority	Acceptable Interrupts
I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>			
1	1	1	7	↑ High  ↓ Low	NMI
1	1	0	6		Level 7 and NMI
1	0	1	5		Levels 6, 7 and NMI
1	0	0	4		Levels 5 to 7 and NMI
0	1	1	3		Levels 4 to 7 and NMI
0	1	0	2		Levels 3 to 7 and NMI
0	0	1	1		Levels 2 to 7 and NMI
0	0	0	0		Levels 1 to 7 and NMI

The CPU accepts only interrupts higher than the interrupt mask level. NMI (level 8) is accepted at any interrupt mask level\*. After accepting an interrupt, the H8/500 CPU updates I<sub>2</sub>, I<sub>1</sub>, and I<sub>0</sub> to the level of the interrupt. Table 2-3 indicates the values of the interrupt mask bits after an interrupt is accepted. A reset sets all three interrupt mask bits to 1.

Note: \* The exception is when programming or erasing flash memory, in which case NMI input is disabled. See section 19.4.9, “NMI Input Masking” and section 20.4.8, “NMI Input Masking” for details.

**Table 2-3 Interrupt Mask Bits (I<sub>2</sub>, I<sub>1</sub>, I<sub>0</sub>) after an Interrupt is Accepted**

Level of Interrupt Accepted	Interrupt Mask		
	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>
NMI (8)	1	1	1
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1

- (4) **Bits 7 to 4—Reserved:** Read-only bits, always read as 0.
- (5) **Bit 3—Negative (N):** The most significant data bit, regarded as a sign bit.
- (6) **Bit 2—Zero (Z):** Set to 1 to indicate zero data and cleared to 0 at other times.
- (7) **Bit 1—Overflow (V):** Set to 1 when an arithmetic overflow occurs and cleared to 0 at other times.
- (8) **Bit 0—Carry (C):** Set to 1 when a carry or borrow occurs at the most significant data bit and cleared to 0 at other times.

The specific changes that occur in the condition code bits when each instruction is executed are listed in Appendix A.1 “Instruction Tables.” See the *H8/500 Series Programming Manual* for further details.

## 2.4 Page Registers

The H8/500 CPU has four page registers.

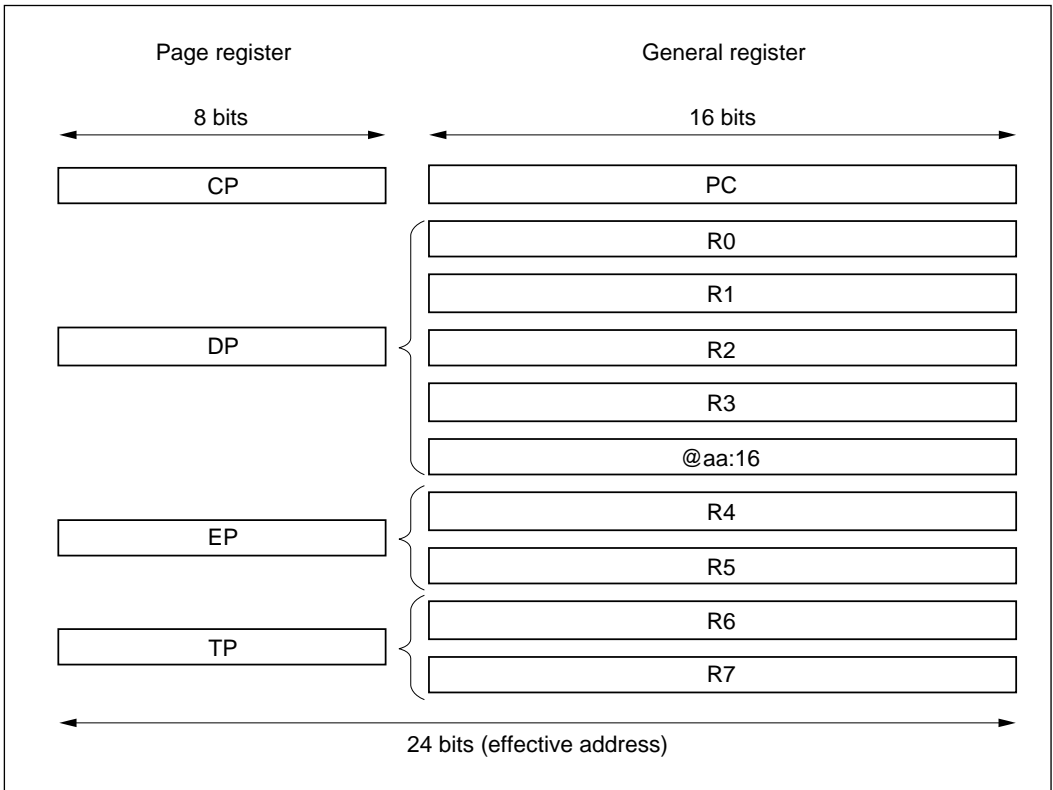
The page registers are described next.

### 2.4.1 Overview

All page registers are eight-bit registers.

The four page registers are the code page register (CP), data page register (DP), extended page register (EP), and stack page register (TP).

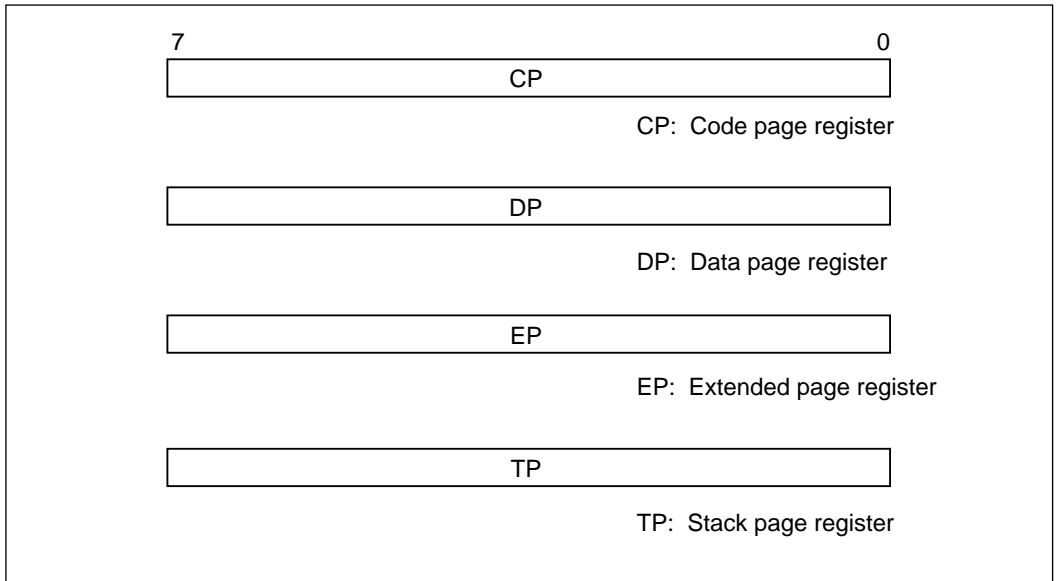
The page registers are not used to calculate effective addresses in minimum mode. In maximum mode, the page registers combine with the program counter and general registers to generate 24-bit effective addresses as shown in figure 2-6, thereby expanding the program area, data area, and stack area.



**Figure 2-6 Combinations of Page Registers with PC and General Registers**

## 2.4.2 Register Configuration

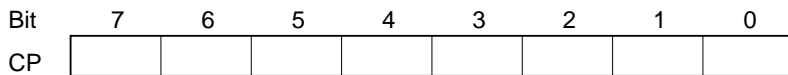
Figure 2-7 shows the page registers.



**Figure 2-7 Page Registers**

## 2.4.3 Code Page Register

The code page register (CP) combines with the program counter to generate a 24-bit program code address. CP contains the upper eight bits of the address.

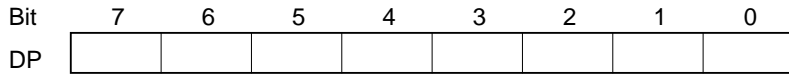


In maximum mode, CP is initialized at a reset to a value loaded from the vector table, and CP and PC are both saved and restored in exception handling.

The LDC instruction can be used to modify the CP contents.

### 2.4.4 Data Page Register

The data page register (DP) combines with general registers R0 to R3 to generate a 24-bit effective address. DP contains the upper eight bits of the address.

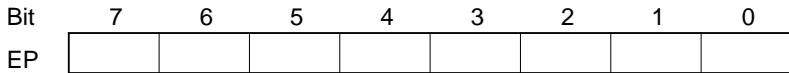


DP is used to calculate effective addresses in register indirect addressing mode using R0 to R3, and in absolute addressing mode (but not short absolute addressing mode).

The LDC instruction can be used to modify the DP contents.

### 2.4.5 Extended Page Register

The extended page register (EP) combines with general register R4 or R5 to generate a 24-bit operand address. EP contains the upper eight bits of the address.

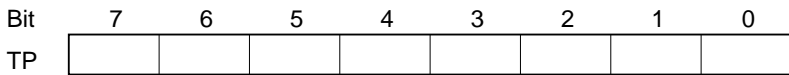


EP is used to calculate effective addresses in register indirect addressing mode using R4 or R5.

The LDC instruction can be used to modify the EP contents.

### 2.4.6 Stack Page Register

The stack page register (TP) combines with R6 (SP) or R7 (FP) to generate a 24-bit stack address. TP contains the upper eight bits of the address.



TP is used to calculate effective addresses in the register indirect addressing mode using R6 or R7, in exception handling, and in subroutine calls.

The LDC instruction can be used to modify the TP contents.

## 2.5 Base Register

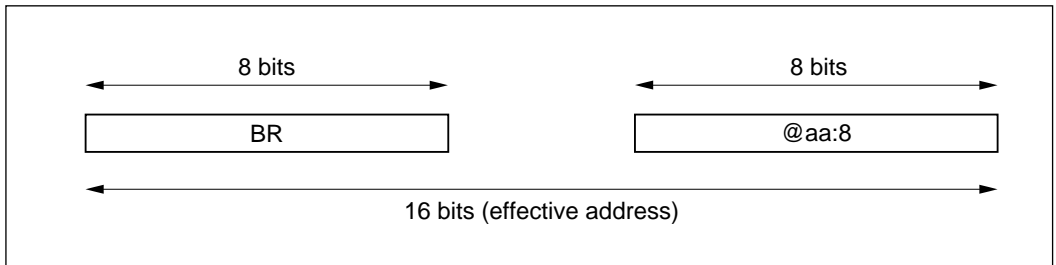
The H8/500 CPU has one 8-bit base register.

The base register is described next.

### 2.5.1 Overview

The eight-bit base register (BR) stores the base address used in short absolute addressing mode (representing the upper eight bits of an address in page 0). Figure 2-8 illustrates the base register and short absolute addressing mode. In this addressing mode a 16-bit effective address is generated by using the BR contents as the upper eight bits and an address given in the instruction code as the lower eight bits. The short absolute addressing mode always addresses page 0.

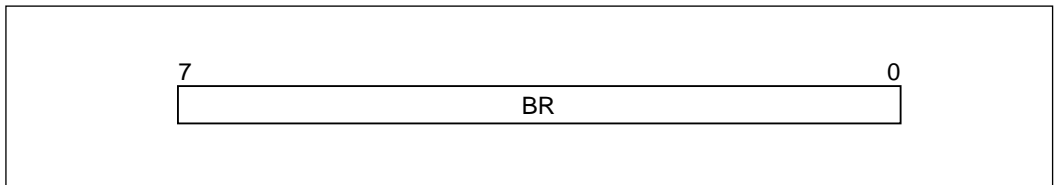
The LDC instruction can be used to modify the BR contents.



**Figure 2-8 Short Absolute Addressing Mode and Base Register**

### 2.5.2 Register Configuration

Figure 2-9 shows the base register.



**Figure 2-9 Base Register**

## 2.6 Data Formats

The H8/500 CPU can process five types of data: one-bit data, four-bit BCD data, eight-bit (byte) data, 16-bit (word) data, and 32-bit (longword) data. Bit manipulation instructions operate on one-bit data. Decimal arithmetic instructions operate on four-bit BCD data. All instructions except certain arithmetic and data transfer instructions can operate on byte and word data. Multiply and divide instructions operate on longword data.

The data formats are described next.

### 2.6.1 Data Formats in General Registers

Table 2-4 indicates the data formats in general registers. All sizes of data can be stored: one-bit data, four-bit BCD data, eight-bit (byte) data, 16-bit (word) data, and 32-bit (longword) data.

In addressing of one-bit data, bit 15 is the most significant bit and bit 0 is the least significant bit. BCD and byte data are stored in the lower eight bits of a general register. All 16 bits of a general register are used to store word data. Two general registers are used for longword data: the upper 16 bits are stored in  $R_n$  ( $n$  must be an even number); the lower 16 bits are stored in  $R_{n+1}$ .

Operations performed on BCD data or byte data do not alter the upper eight bits of the register.

**Table 2-4 General Register Data Formats**

Data Type	Register No.	Data Structure																																		
One bit	$R_n$	<div style="display: flex; justify-content: space-between; width: 100%;"> <span>15</span> <span>0</span> </div> <table border="1" style="width: 100%; text-align: center;"> <tr> <td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> </table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																					
BCD	$R_n$	<div style="display: flex; justify-content: space-between; width: 100%;"> <span>7</span> <span>4 3</span> <span>0</span> </div> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="8">Don't care</td> <td colspan="3">Upper digit</td> <td colspan="3">Lower digit</td> </tr> </table>	Don't care								Upper digit			Lower digit																						
Don't care								Upper digit			Lower digit																									
Byte	$R_n$	<div style="display: flex; justify-content: space-between; width: 100%;"> <span>7</span> <span>0</span> </div> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="8">Don't care</td> <td colspan="3">MSB</td> <td colspan="3">LSB</td> </tr> </table>	Don't care								MSB			LSB																						
Don't care								MSB			LSB																									
Word	$R_n$	<div style="display: flex; justify-content: space-between; width: 100%;"> <span>15</span> <span>0</span> </div> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="8">MSB</td> <td colspan="8">LSB</td> </tr> </table>	MSB								LSB																									
MSB								LSB																												
Longword*	$R_n$ $R_{n+1}$	<div style="display: flex; justify-content: space-between; width: 100%;"> <span>31</span> <span>16</span> </div> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="8">MSB</td> <td colspan="8">Upper 16 bits</td> </tr> <tr> <td colspan="8"></td> <td colspan="8">Lower 16 bits</td> <td colspan="2">LSB</td> </tr> </table> <div style="display: flex; justify-content: space-between; width: 100%;"> <span>15</span> <span>0</span> </div>	MSB								Upper 16 bits																Lower 16 bits								LSB	
MSB								Upper 16 bits																												
								Lower 16 bits								LSB																				

Note: \* For longword data  $n$  must be even (0, 2, 4, or 6).

## 2.6.2 Data Formats in Memory

Table 2-5 indicates the data formats in memory.

Instructions that access bit data in memory have byte or word operands. The instruction specifies a bit number to indicate a specific bit in the operand.

Access to word data in memory must always begin at an even address. Access to word data starting at an odd address causes an address error. The upper eight bits of word data are stored in address  $n$  (where  $n$  is an even number); the lower eight bits are stored in address  $n + 1$ .

**Table 2-5 Data Formats in Memory**

Data Type	Data Format																
One bit (in byte operand data)	<p>Address <math>n</math></p> <table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> </table>	7	6	5	4	3	2	1	0								
7	6	5	4	3	2	1	0										
One bit (in word operand data)	<p>Even address</p> <table border="1"> <tr> <td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td> </tr> </table> <p>Odd address</p> <table border="1"> <tr> <td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> </table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8										
7	6	5	4	3	2	1	0										
Byte	<p>Address <math>n</math></p> <table border="1"> <tr> <td>MSB</td><td>LSB</td> </tr> </table>	MSB	LSB														
MSB	LSB																
Word	<p>Even address</p> <table border="1"> <tr> <td>MSB</td><td>Upper 8 bits</td> </tr> </table> <p>Odd address</p> <table border="1"> <tr> <td>Lower 8 bits</td><td>LSB</td> </tr> </table>	MSB	Upper 8 bits	Lower 8 bits	LSB												
MSB	Upper 8 bits																
Lower 8 bits	LSB																

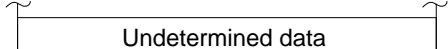

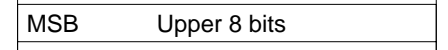
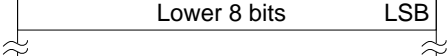
## 2.6.3 Stack Data Formats

Table 2-6 shows the data formats on the stack.

When the stack is accessed in exception processing (to save or restore the program counter, code page register, or status register), word access is always performed, regardless of the actual data size. Similarly, when the stack is accessed by an instruction using the pre-decrement or post-increment register indirect addressing mode specifying  $R7$  ( $@-R7$  or  $@R7+$ ), which is the stack pointer, word access is performed regardless of the operand size specified in the instruction. Programs should be coded so that the stack pointer always indicates an even address. An address error will occur if the stack pointer indicates an odd address.



**Table 2-6 Data Formats on the Stack**

<b>Data Type</b>	<b>Data Format</b>
Byte data on stack	Even address 
	Odd address 
Word data on stack	Even address 
	Odd address 

## 2.7 Addressing Modes and Effective Address Calculation

The H8/500 CPU supports seven addressing modes.

These modes and the corresponding effective address calculations are described next.

### 2.7.1 Addressing Modes

The seven addressing modes supported by the H8/500 CPU are:

1. Register direct
2. Register indirect
3. Register indirect with displacement
4. Register indirect with pre-decrement or post-increment
5. Immediate
6. Absolute
7. PC-relative

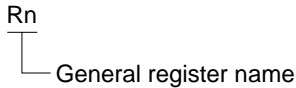
Due to the highly orthogonal nature of the instruction set, most instructions having operands can use any applicable addressing mode from 1 through 6. The PC-relative mode 7 is used by branching instructions.

In most instructions, the addressing mode is specified in the effective address (EA) field and effective address extension (if present).

Table 2-7 indicates how the addressing mode is specified in the effective address field.

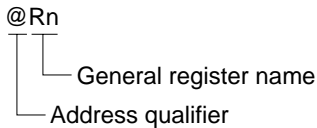
**(1) Register Direct Addressing Mode:** The contents of a general register Rn are used directly as operand data. This addressing mode is specified by giving the general register name.

#### Register direct addressing mode



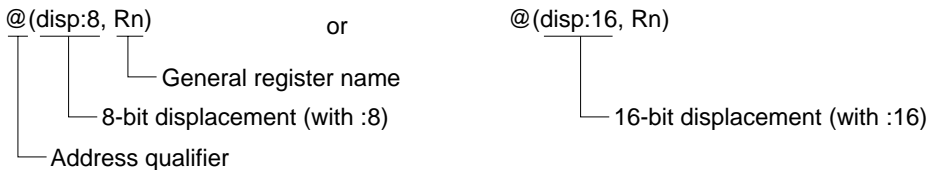
**(2) Register Indirect Addressing Mode:** The contents of a general register Rn are used as a memory address, and data access is performed at that memory address. This addressing mode is specified by giving the general register name with an address qualifier (@).

#### Register indirect addressing mode



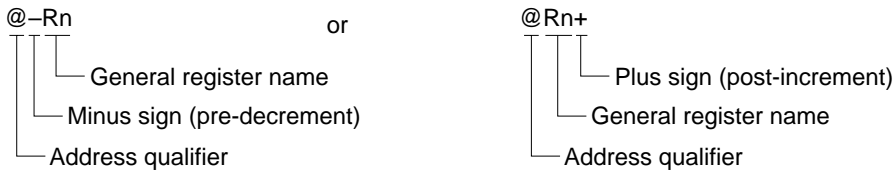
**(3) Register Indirect Addressing Mode with Displacement:** A displacement value is added to the contents of a general register Rn, the sum is used as a memory address, and data access is performed at that memory address. This addressing mode is specified by giving the general register name with the address qualifier (@) and an 8-bit or 16-bit displacement value.

#### Register indirect addressing mode with displacement



**(4) Register Indirect Addressing Mode with Pre-Decrement or Post-Increment:** In register indirect addressing mode with pre-decrement, a general register value is first decremented by  $-1$  or  $-2$ , then the result is used as a memory address and data access is performed at that memory address. In register indirect addressing mode with post-increment, a general register value is used as a memory address and data access is performed at that memory address, then the register value is incremented by 1 or 2. This addressing mode is specified by giving the general register name with the address qualifier (@) and a plus or minus sign (+ or -).

## Register indirect addressing mode with pre-decrement or post-increment



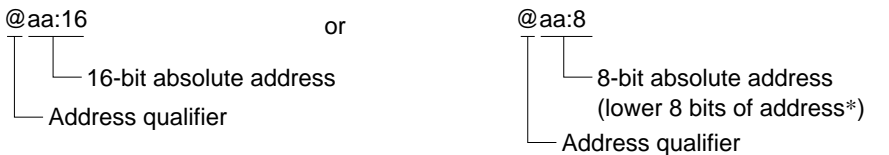
**(5) Immediate Addressing Mode:** Eight-bit or 16-bit immediate data given in the instruction are used directly as the operand data. This addressing mode is specified by giving the immediate data with a data qualifier (#).

### Immediate addressing mode



**(6) Absolute Addressing Mode:** Data access is performed at a memory address given as a 16-bit absolute address in the instruction, or given as an eight-bit absolute address in the instruction and combined with the base register (BR) value. This addressing mode is specified by giving the absolute address with an address qualifier.

### Absolute addressing mode



\* Upper 8 bits are specified by BR

**(7) PC-Relative Addressing Mode:** An eight-bit or 16-bit displacement value given in the instruction is added to the program counter value, the sum is used as a memory address, and this memory address is moved into the program counter. This addressing mode is specified by giving the displacement value.

### PC-relative addressing mode



**Table 2-7 Addressing Modes**

No.	Addressing Mode	Mnemonic	EA Field	EA Extension																
1	Register direct	Rn	<table border="1"> <tr> <td>1</td><td>0</td><td>1</td><td>0</td><td>Sz</td><td>r</td><td>r</td><td>r</td> </tr> <tr> <td colspan="5"></td> <td>*1</td><td colspan="2">*2</td> </tr> </table>	1	0	1	0	Sz	r	r	r						*1	*2		None
1	0	1	0	Sz	r	r	r													
					*1	*2														
2	Register indirect	@Rn	<table border="1"> <tr> <td>1</td><td>1</td><td>0</td><td>1</td><td>Sz</td><td>r</td><td>r</td><td>r</td> </tr> </table>	1	1	0	1	Sz	r	r	r	None								
1	1	0	1	Sz	r	r	r													
3	Register indirect with displacement	@(d:8,Rn) @(d:16,Rn)	<table border="1"> <tr> <td>1</td><td>1</td><td>1</td><td>0</td><td>Sz</td><td>r</td><td>r</td><td>r</td> </tr> <tr> <td>1</td><td>1</td><td>1</td><td>1</td><td>Sz</td><td>r</td><td>r</td><td>r</td> </tr> </table>	1	1	1	0	Sz	r	r	r	1	1	1	1	Sz	r	r	r	Displacement (1 byte) Displacement (2 bytes)
1	1	1	0	Sz	r	r	r													
1	1	1	1	Sz	r	r	r													
4	Register indirect with pre-decrement Register indirect with post-increment	@-Rn @Rn+	<table border="1"> <tr> <td>1</td><td>0</td><td>1</td><td>1</td><td>Sz</td><td>r</td><td>r</td><td>r</td> </tr> <tr> <td>1</td><td>1</td><td>0</td><td>0</td><td>Sz</td><td>r</td><td>r</td><td>r</td> </tr> </table>	1	0	1	1	Sz	r	r	r	1	1	0	0	Sz	r	r	r	None
1	0	1	1	Sz	r	r	r													
1	1	0	0	Sz	r	r	r													
5	Immediate	#xx:8 #xx:16	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td> </tr> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td> </tr> </table>	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	Immediate data (1 byte) Immediate data (2 bytes)
0	0	0	0	0	1	0	0													
0	0	0	0	1	1	0	0													
6	Absolute (@aa:8 is short absolute)	@aa:8 @aa:16	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>Sz</td><td>1</td><td>0</td><td>1</td> </tr> <tr> <td>0</td><td>0</td><td>0</td><td>1</td><td>Sz</td><td>1</td><td>0</td><td>1</td> </tr> </table>	0	0	0	0	Sz	1	0	1	0	0	0	1	Sz	1	0	1	1-byte absolute address (offset from BR) 2-byte absolute address
0	0	0	0	Sz	1	0	1													
0	0	0	1	Sz	1	0	1													
7	PC-relative	disp	No EA field. Addressing mode is specified in op-code.	1- or 2-byte displacement																

Notes: 1. Sz specifies the operand size.

2. rrr specifies a general register.

Sz	Operand Size
0	Byte
1	Word

rrr	General Register
000	R0
001	R1
010	R2
011	R3
100	R4
101	R5
110	R6
111	R7

## 2.7.2 Effective Address Calculation

Table 2-8 explains how an effective address is calculated in each addressing mode. The page registers are not used to calculate effective addresses in minimum mode.

**Table 2-8 Effective Address Calculation**

No.	Addressing Mode Mnemonic	Effective Address Calculation	Effective Address
1	Register direct Rn	—	Operand is contents of Rn.
2	Register indirect @Rn	—	
3	Register indirect with displacement @(d:8,Rn)		
	@(d:16,Rn)		
4	Register indirect with pre-decrement @-Rn		
	Register indirect with post-increment @Rn+	Rn is incremented by 1 or 2 after instruction execution.	

Notes: 1. 1 for a byte operand, 2 for a word operand, and always 2 for R7 in register indirect mode with pre-decrement or post-increment, even if byte size is specified.

2. Register Indirect	Page Register
R7, R6	TP
R5, R4	EP
R3–R0	DP

**Table 2-8 Effective Address Calculation (cont)**

No.	Addressing Mode Mnemonic EA Field	Effective Address Calculation	Effective Address														
5	Absolute @aa:8	—	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">23</td> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">H'00</td> <td style="text-align: center;">BR</td> <td style="text-align: center;">EA extension data</td> </tr> </table>	23	15	0	H'00	BR	EA extension data								
	23	15	0														
H'00	BR	EA extension data															
	<div style="border: 1px solid black; padding: 2px; display: inline-block;">0 0 0 0 Sz 1 0 1</div>	—															
5	@aa:16	—	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">23</td> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">DP</td> <td colspan="2" style="text-align: center;">EA extension data</td> </tr> </table>	23	15	0	DP	EA extension data									
	23	15	0														
DP	EA extension data																
	<div style="border: 1px solid black; padding: 2px; display: inline-block;">0 0 0 1 Sz 1 0 1</div>	—															
6	Immediate #xx:8	—	Operand is 1-byte EA extension data.														
	<div style="border: 1px solid black; padding: 2px; display: inline-block;">0 0 0 0 0 1 0 0</div>	—															
6	#xx:16	—	Operand is 2-byte EA extension data.														
	<div style="border: 1px solid black; padding: 2px; display: inline-block;">0 0 0 0 1 1 0 0</div>	—															
7	PC-relative d:8 No EA field. Specified in op-code.	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">PC</td> <td style="text-align: center;">0</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">Displacement</td> <td style="text-align: center;">0</td> </tr> </table> <p style="text-align: center;">16 bits (8 bits with sign extension)</p>	15	0	PC	0	15	0	Displacement	0	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">23</td> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">CP</td> <td colspan="2" style="text-align: center;">Result</td> </tr> </table>	23	15	0	CP	Result	
	15	0															
PC	0																
15	0																
Displacement	0																
23	15	0															
CP	Result																
7	d:16 No EA field. Specified in op-code.	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">PC</td> <td style="text-align: center;">0</td> </tr> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">Displacement</td> <td style="text-align: center;">0</td> </tr> </table>	15	0	PC	0	15	0	Displacement	0	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: center;">23</td> <td style="width: 50%; text-align: center;">15</td> <td style="width: 50%; text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">CP</td> <td colspan="2" style="text-align: center;">Result</td> </tr> </table>	23	15	0	CP	Result	
	15	0															
PC	0																
15	0																
Displacement	0																
23	15	0															
CP	Result																

## 2.8 Operating Modes

The H8/500 CPU has two operating modes: minimum mode and maximum mode. The mode is selected by the mode pins ( $MD_2$  to  $MD_0$ ).

The operating modes are described next.

### 2.8.1 Minimum Mode

Minimum mode supports an address space of up to 64 kbytes. The page registers are ignored. Instructions that branch across page boundaries (PJMP, PJSR, PRTS, PRTD) are invalid.

### 2.8.2 Maximum Mode

In maximum mode the page registers are valid, expanding the maximum address space to 1 Mbyte. It is possible to move from one page to another with branching instructions (PJMP, PJSR, PRTS, PRTD) and when branching to interrupt-handling routines.

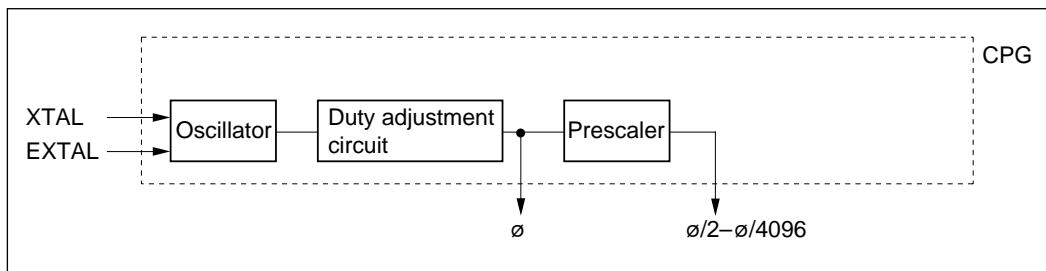
When data access crosses a page boundary, the program must rewrite the page register before it can access the data in the next page.

For further information on the operating modes, see section 3, “MCU Operating Modes.”

## 2.9 Basic Operational Timing

When an external clock signal is fed to the EXTAL pin or a crystal resonator is connected across the XTAL and EXTAL pins, supplying a signal of the same frequency as the system clock ( $\phi$ ), duty adjustment is performed by the duty adjustment circuit to generate the  $\phi$  clock. Figure 2-10 shows a block diagram of the clock oscillator.

The basic operational timing of the H8/500 CPU is described next.



**Figure 2-10 Block Diagram of Clock Oscillator**

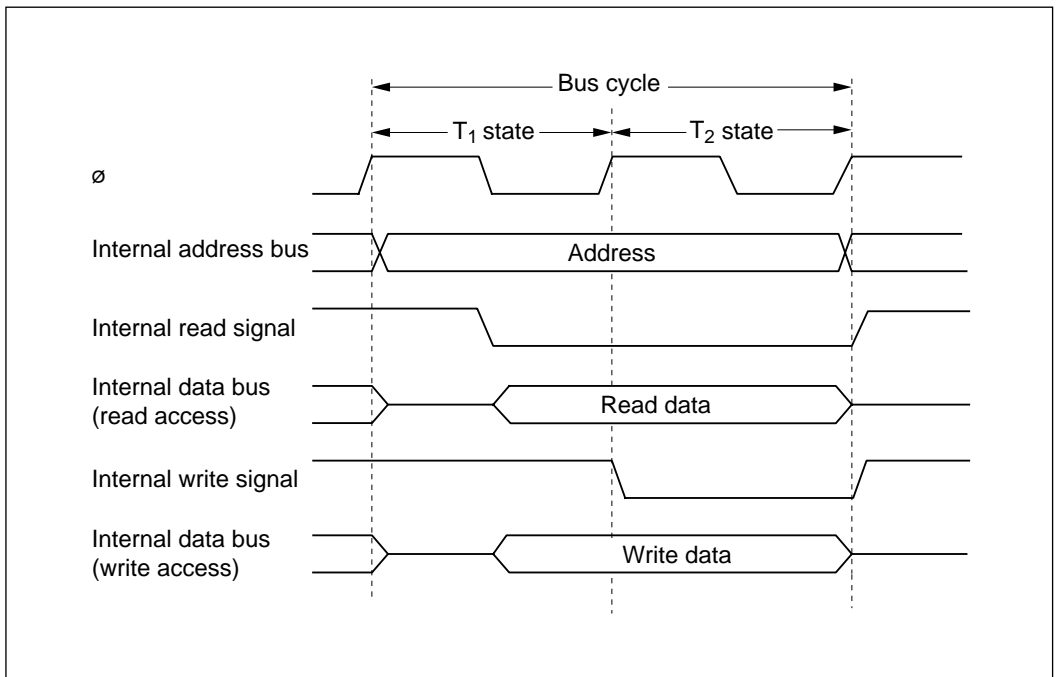
## 2.9.1 Overview

The system clock ( $\phi$ ) produced by duty adjustment of the clock ( $f_{OSC}$ ) supplied from the clock oscillator is the H8/500 CPU's time base. One cycle of the system clock is referred to as a "state." The H8/500 CPU's bus cycle consists of two or three states. The CPU uses different methods to access on-chip memory, on-chip supporting modules, and external devices.

These access methods are described next.

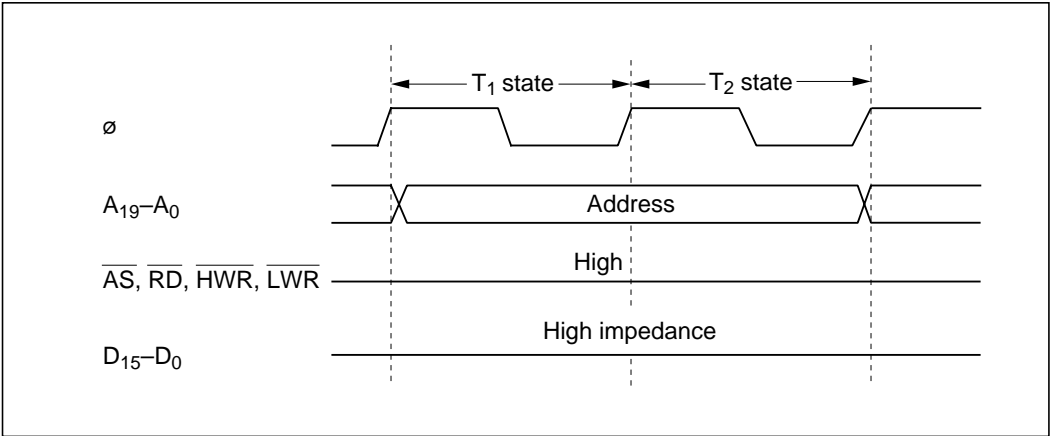
## 2.9.2 Access to On-Chip Memory

On-chip memory is accessed in two states using a 16-bit bus. Figure 2-11 shows the on-chip memory access cycle. Figure 2-12 shows the pin states during on-chip memory access.



**Figure 2-11 On-Chip Memory Access Cycle**

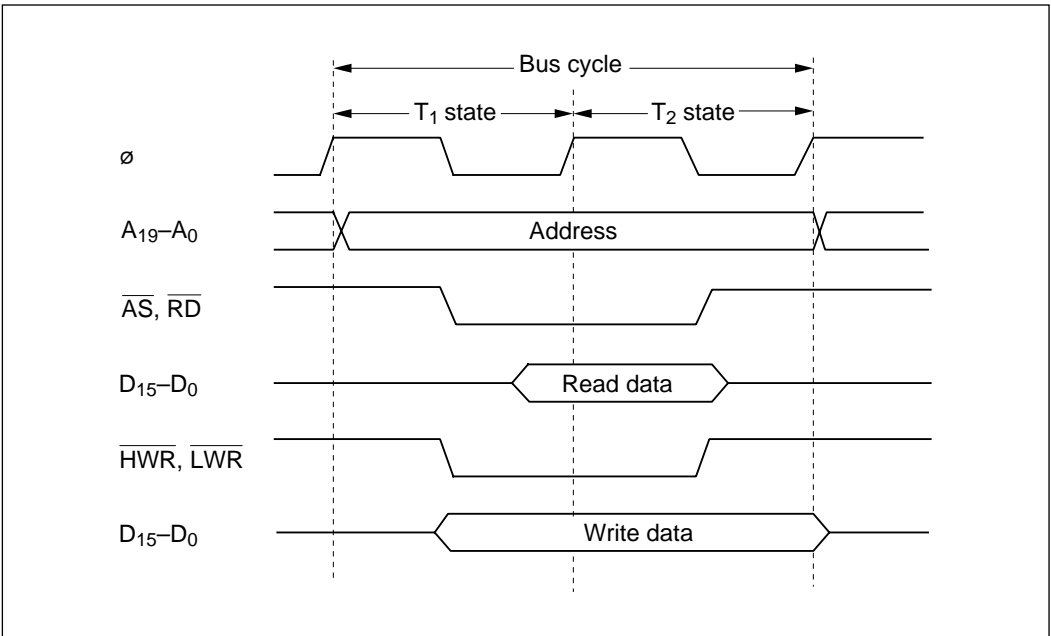




**Figure 2-12 Pin States during Access to On-Chip Memory**

### 2.9.3 Access to Two-State-Access Address Space

Two-state access permits high-speed processing. No wait states ( $T_W$ ) can be inserted in access to the two-state-access address space. The external two-state-access address space is accessed via a 16-bit bus. Figure 2-13 shows the access cycle for the external two-state-access address space.

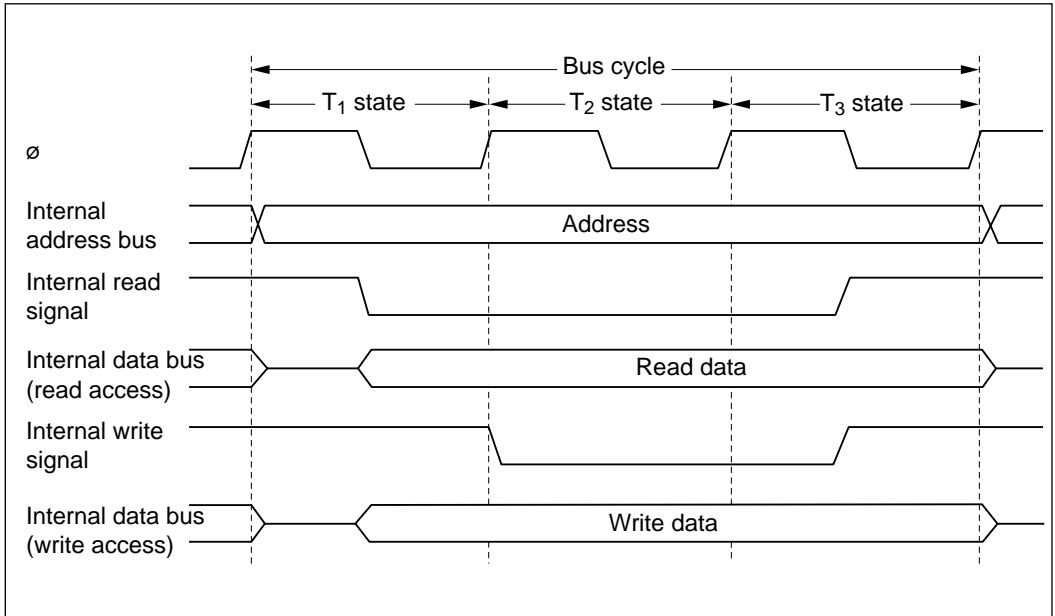


**Figure 2-13 Access Cycle for External Two-State-Access Address Space**

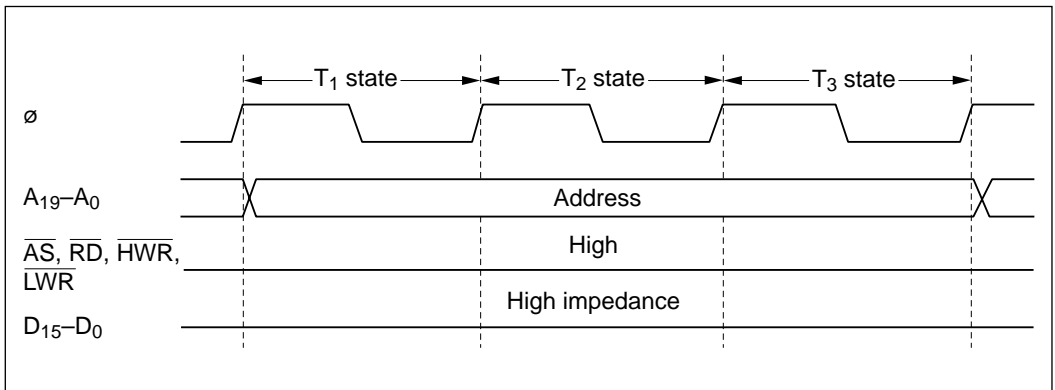
## 2.9.4 Access to On-Chip Supporting Modules

The on-chip supporting modules are always accessed in three states. The data bus is eight bits wide, except that some of the registers in the 16-bit integrated-timer pulse unit (IPU) are accessed via a 16-bit data bus.

Figure 2-14 shows the on-chip supporting module access cycle. Figure 2-15 indicates the pin states during access to an on-chip supporting module.



**Figure 2-14 Access Cycle for On-Chip Supporting Modules**



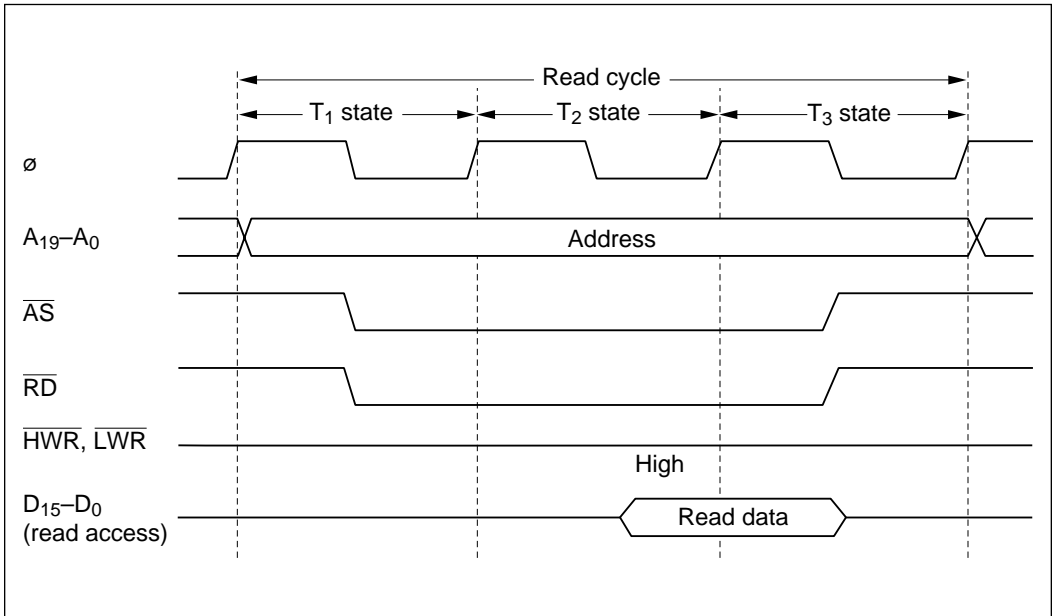
**Figure 2-15 Pin States during Access to On-Chip Supporting Modules**

## 2.9.5 Access to Three-State-Access Address Space

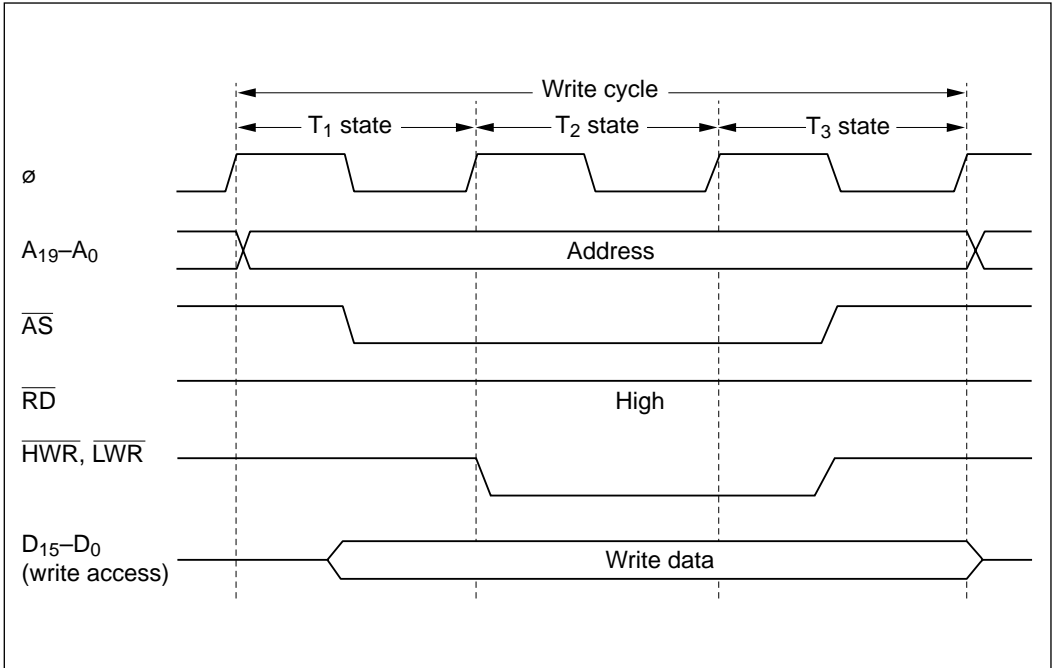
Three-state access is used for interfacing to low-speed devices.

The wait-state controller (WSC) can insert wait states ( $T_W$ ) in access to the three-state-access address space.

Figure 2-16 shows the three-state read access cycle. Figure 2-17 shows the three-state write access cycle.



**Figure 2-16 Read Access Cycle for Three-State-Access Address Space**



**Figure 2-17 Write Access Cycle for Three-State-Access Address Space**

## 2.10 CPU States

The H8/500 CPU has five processing states.

These states are described next.

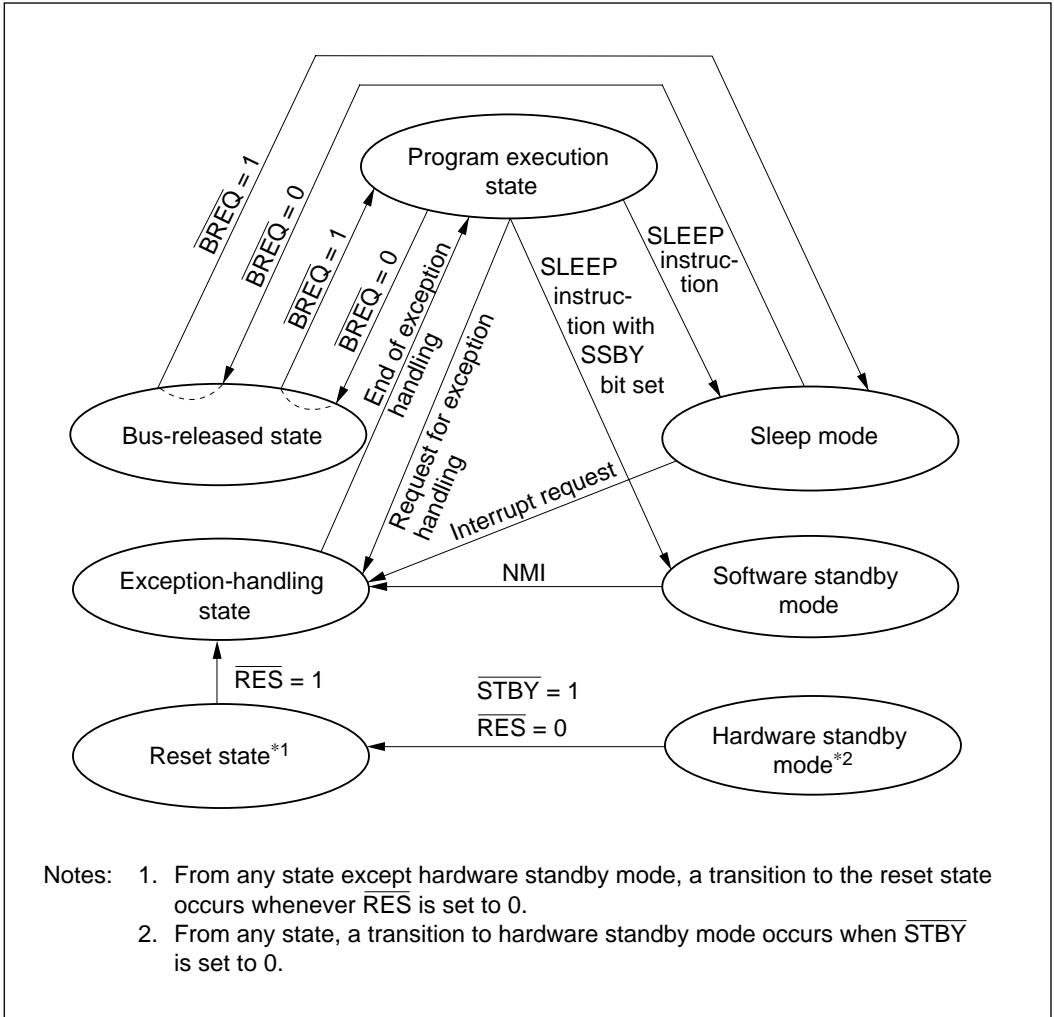
### 2.10.1 Overview

The five processing states of the H8/500 CPU are the program execution state, exception-handling state, bus-released state, reset state, and power-down state.

The power-down state is further divided into a sleep mode, software standby mode, and hardware standby mode. Table 2-9 summarizes these states. Figure 2-18 shows a map of the state transitions.

**Table 2-9 Processing States**

<b>State</b>	<b>Description</b>
Program execution state	The H8/500 CPU executes program instructions in sequence.
Exception-handling state	A transient state in which the H8/500 CPU executes a hardware sequence (saving the program counter and status register, fetching a vector, etc.) triggered by a reset, interrupt, or other exception.
Bus-released state	The H8/500 CPU has released the external bus in response to an external bus request signal.
Reset state	The H8/500 CPU and all on-chip supporting modules have been initialized and are stopped.
Power-down state	Sleep mode Software standby mode Hardware standby mode



**Figure 2-18 State Transitions**

### 2.10.2 Program Execution State

In this state the H8/500 CPU executes program instructions in normal sequence.

### 2.10.3 Exception-Handling State

The exception-handling state is a transient state that occurs when the H8/500 CPU alters the normal program flow due to an interrupt, trap instruction, address error, or other exception.

See section 4, “Exception Handling” for further information on the exception-handling state.

## 2.10.4 Bus-Released State

When so requested, the H8/500 CPU can grant control of the external bus to an external device. While an external device has the bus right, the H8/500 CPU is said to be in the bus-released state.

Granting of the bus is controlled by the  $\overline{\text{BREQ}}$  and  $\overline{\text{BACK}}$  signals. Bus requests are input at the  $\overline{\text{BREQ}}$  pin. When the bus has been released, an acknowledging signal is output at the  $\overline{\text{BACK}}$  pin.

Figure 2-19 illustrates the procedure for releasing the bus.

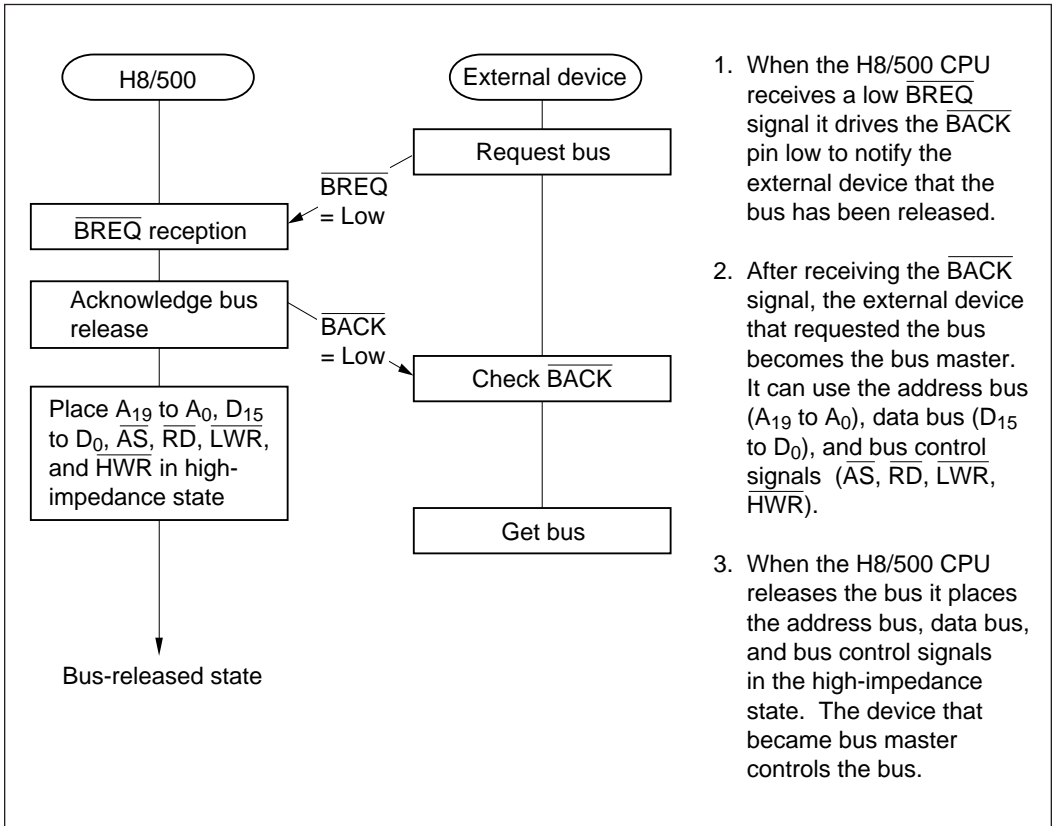


Figure 2-19 Bus Release Procedure

**Bus Release Control Register (Address H'FF1B):** This register (BRCR) enables and disables  $\overline{\text{BREQ}}$  input and  $\overline{\text{BACK}}$  output. BRCR is initialized to H'FE by a reset and in hardware standby mode. It is not initialized in software standby mode. The BRCR bit structure is shown next.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	BRLE
Initial value	1	1	1	1	1	1	1	0
R/W	—	—	—	—	—	—	—	R/W

Bus release enable bit  
 Selects port A functions

**Bits 7 to 1—Reserved:** Read-only bits, always read as 1.

**Bit 0—Bus Release Enable Bit (BRLE):** Selects the functions of pins PA<sub>6</sub> and PA<sub>5</sub>.

**Bit 0**

BRLE	Description
0	PA <sub>6</sub> and PA <sub>5</sub> are used for general-purpose input and output
1	PA <sub>6</sub> is used for $\overline{\text{BACK}}$ output; PA <sub>5</sub> is used for $\overline{\text{BREQ}}$ input



### (1) Case in which $\overline{\text{BREQ}}$ is Acknowledged at End of Bus Cycle

Figure 2-20 shows the timing when the H8/500 CPU acknowledges the  $\overline{\text{BREQ}}$  signal at the end of a bus cycle.

The  $\overline{\text{BREQ}}$  signal is sampled during every instruction fetch cycle and data read or write cycle. In word data access by means of two successive byte accesses, first to the upper byte, then to the lower byte (access to the eight-bit-bus-access address space or an on-chip supporting module), the H8/500 CPU does not release the bus right until it has accessed the lower byte.

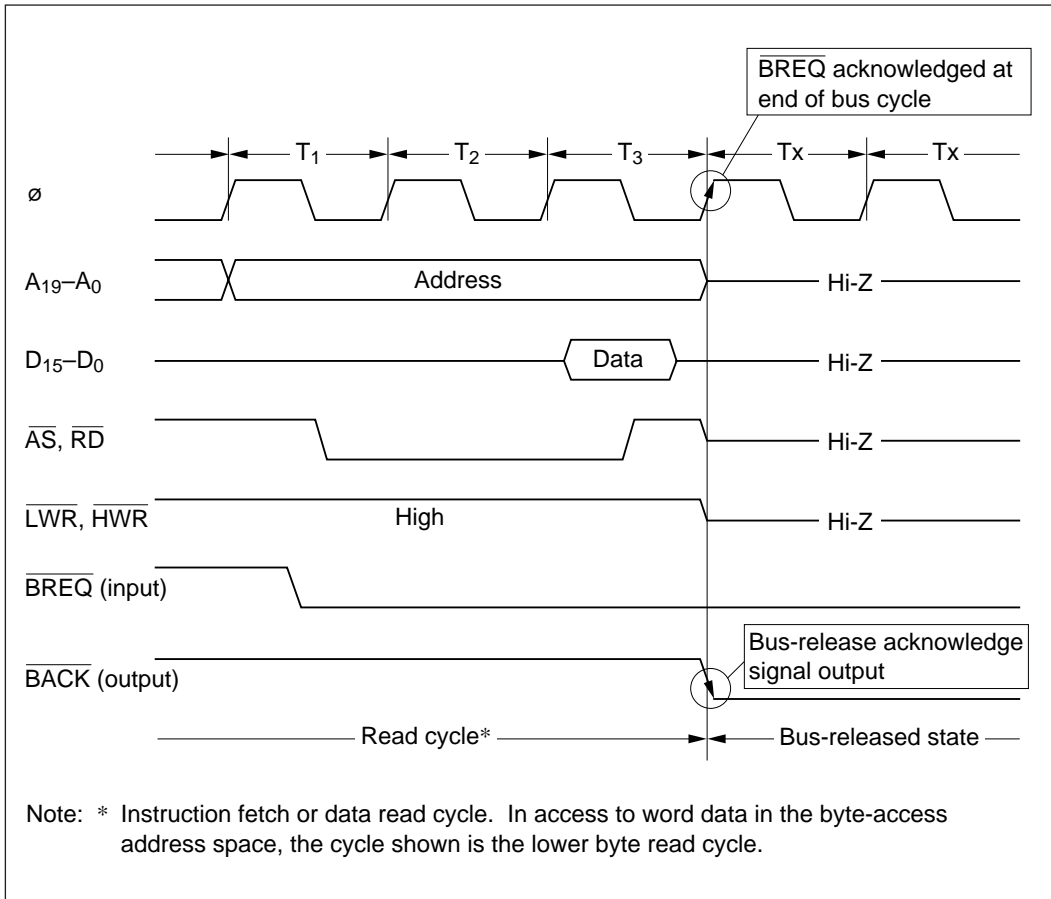
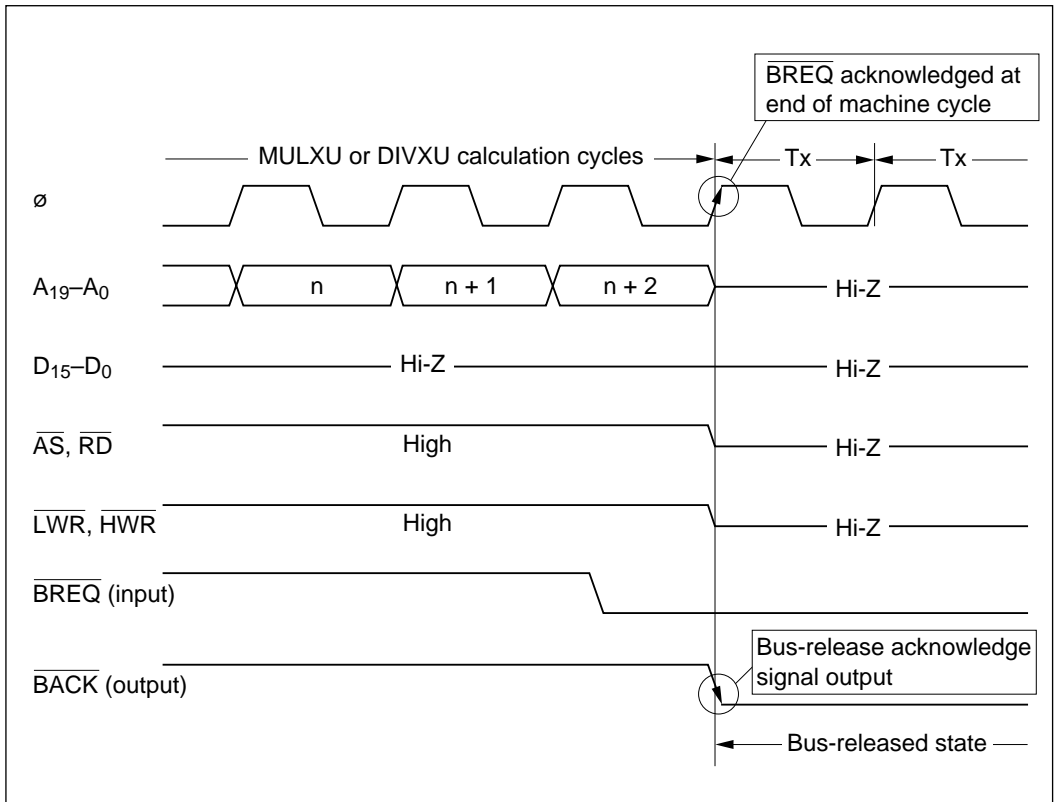


Figure 2-20 Case of  $\overline{\text{BREQ}}$  Acknowledged at End of Bus Cycle (Read Cycle Example)

**(2) Case in which  $\overline{\text{BREQ}}$  is Acknowledged at End of Machine Cycle**

Figure 2-21 shows the timing when the H8/500 CPU acknowledges the  $\overline{\text{BREQ}}$  signal at the end of a machine cycle.

The H8/500 CPU acknowledges the  $\overline{\text{BREQ}}$  signal at the end of machine cycles during execution of the MULXU or DIVXU instruction.



**Figure 2-21 Case of  $\overline{\text{BREQ}}$  Acknowledged at End of Machine Cycle (During Execution of MULXU or DIVXU Instruction)**

### (3) Case in which $\overline{\text{BREQ}}$ is Acknowledged in Sleep Mode

Figure 2-22 shows the timing when the H8/500 CPU acknowledges the  $\overline{\text{BREQ}}$  signal in sleep mode.

The H8/500 CPU acknowledges the  $\overline{\text{BREQ}}$  signal at any time during sleep mode.

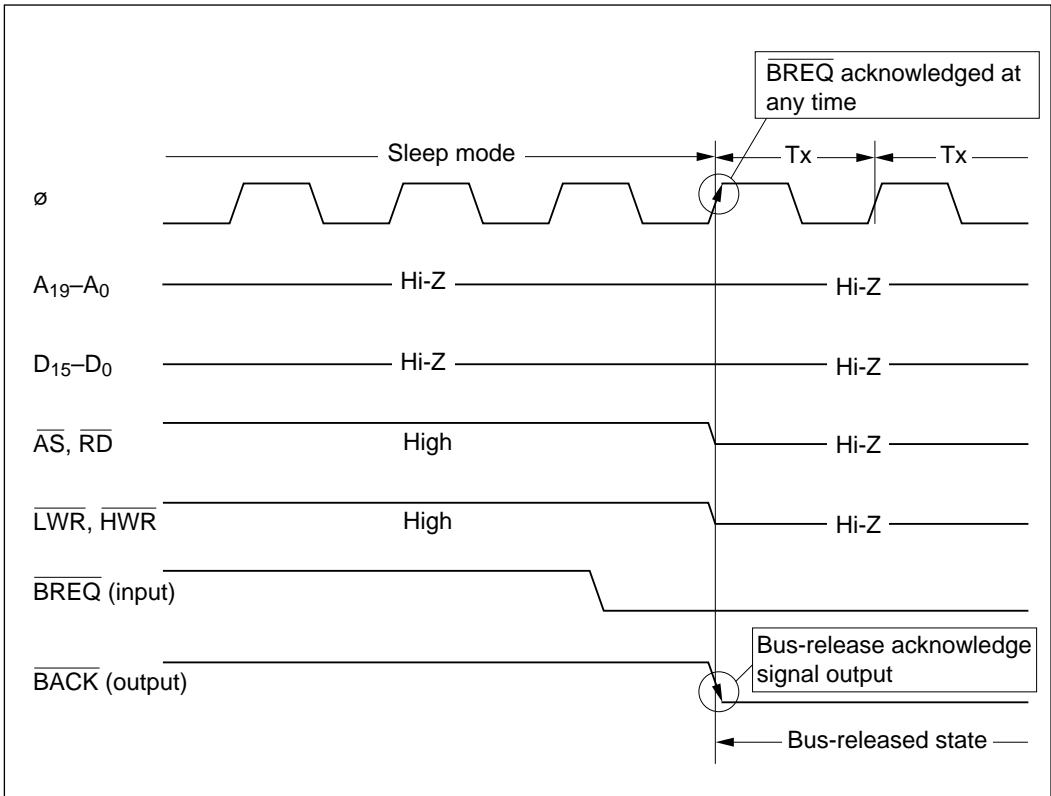


Figure 2-22 Case of  $\overline{\text{BREQ}}$  Acknowledged in Sleep Mode

#### (4) Bus-Release Operation during Two-State Access

Figure 2-23 shows the timing when the bus is requested during a two-state access cycle.

When an external device requests the bus during two-state access, the H8/500 CPU enters the bus-released state as follows:

- (1) The  $\overline{\text{BREQ}}$  pin is sampled at the start of the  $T_1$  state. If  $\overline{\text{BREQ}}$  is low, at the end of the bus cycle the H8/500 CPU halts and enters the bus-released state.
- (2) In the case of two-state access, at the end of the  $T_2$  state the  $\overline{\text{BACK}}$  signal goes low to indicate that the bus-released state has been entered. The address bus ( $A_{19}$  to  $A_0$ ), data bus ( $D_{15}$  to  $D_0$ ), and bus control signals ( $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{LWR}}$ ,  $\overline{\text{HWR}}$ ) are placed in the high-impedance state.
- (3) While the bus is released, the H8/500 CPU constantly samples the  $\overline{\text{BREQ}}$  pin (at each  $T_x$  state) and remains in the bus-released state while  $\overline{\text{BREQ}}$  is low.
- (4) When  $\overline{\text{BREQ}}$  is high during a  $T_x$  state, at the end of the next state the H8/500 CPU drives the  $\overline{\text{BACK}}$  signal high to indicate that it has regained possession of the bus (and that CPU cycles will resume).
- (5) CPU cycles resume at the end of the next state after  $\overline{\text{BACK}}$  goes high.

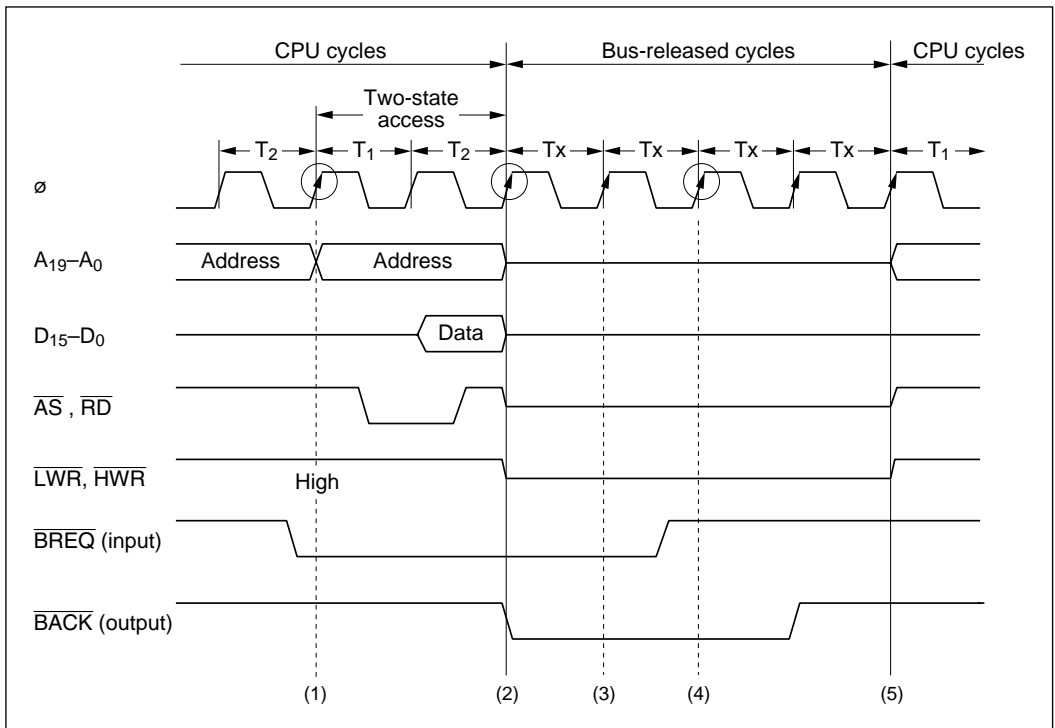


Figure 2-23 Bus Release during Two-State Access (Read Cycle Example)

## (5) Bus-Release Operation during Three-State Access

Figure 2-24 shows the timing when the bus is requested during a three-state access cycle.

When an external device requests the bus during three-state access, the H8/500 CPU enters the bus-released state as follows:

- (1) The  $\overline{\text{BREQ}}$  pin is sampled at the start of the  $T_1$ ,  $T_2$ , and  $T_W$  states. If  $\overline{\text{BREQ}}$  is low, at the end of the bus cycle the H8/500 CPU halts and enters the bus-released state.
- (2) In the case of three-state access, at the end of the  $T_3$  state the  $\overline{\text{BACK}}$  signal goes low to indicate that the bus-released state has been entered. The address bus ( $A_{19}$  to  $A_0$ ), data bus ( $D_{15}$  to  $D_0$ ), and bus control signals ( $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{LWR}}$ ,  $\overline{\text{HWR}}$ ) are placed in the high-impedance state.
- (3) When  $\overline{\text{BREQ}}$  is high during a  $T_x$  state, at the end of the next state the H8/500 CPU drives the  $\overline{\text{BACK}}$  signal high to indicate that it has regained possession of the bus (and that CPU cycles will resume).
- (4) CPU cycles resume at the end of the next state after  $\overline{\text{BACK}}$  goes high.

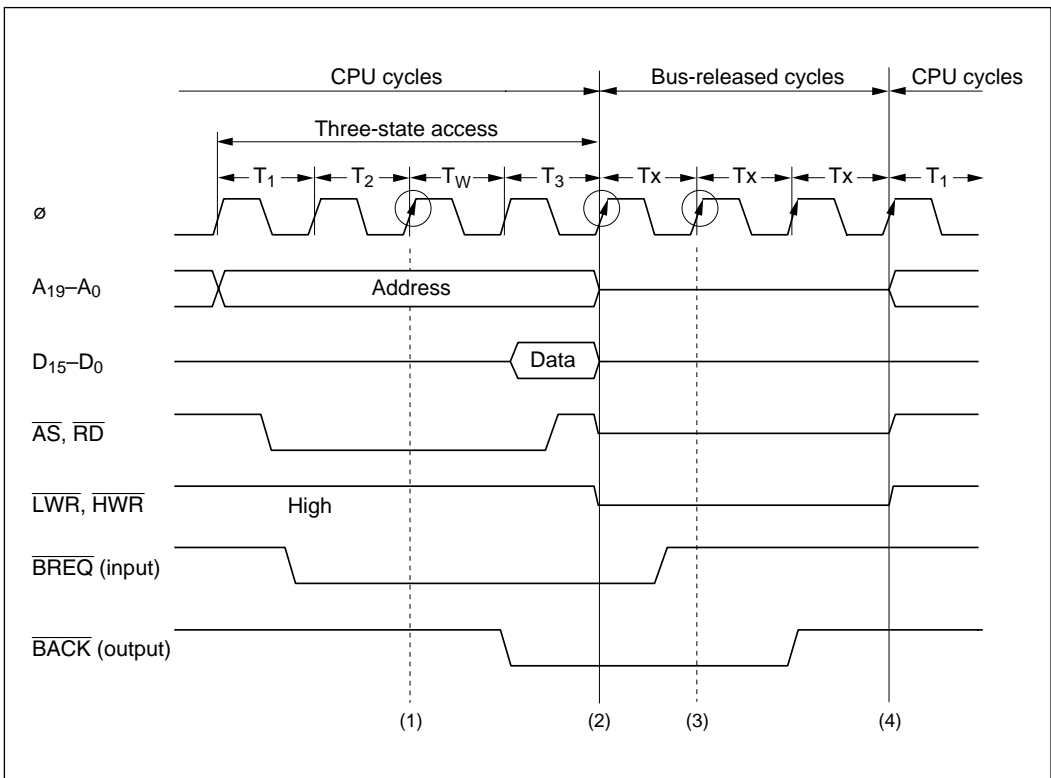


Figure 2-24 Bus Release during Three-State Access (Read Cycle Example)

## (6) Bus-Release Operation during Internal CPU Operations

Figure 2-25 shows the timing when the bus is requested during internal CPU operations.

When an external device requests the bus during internal CPU operations, the H8/500 CPU enters the bus-released state as follows:

- (1) The  $\overline{\text{BREQ}}$  pin is sampled at the start of the  $T_1$  state. If  $\overline{\text{BREQ}}$  is low, at the end of the internal cycle the H8/500 CPU halts and enters the bus-released state.
- (2) In the case of internal CPU operations, at the end of a  $T_1$  state the  $\overline{\text{BACK}}$  signal goes low to indicate that the bus-released state has been entered. The address bus ( $A_{19}$  to  $A_0$ ), data bus ( $D_{15}$  to  $D_0$ ), and bus control signals ( $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{LWR}}$ ,  $\overline{\text{HWR}}$ ) are placed in the high-impedance state.
- (3) When  $\overline{\text{BREQ}}$  is high during a  $T_x$  state, at the end of the next state the H8/500 CPU drives the  $\overline{\text{BACK}}$  signal high to indicate that it has regained possession of the bus (and that CPU cycles will resume).
- (4) CPU cycles resume at the end of the next state after  $\overline{\text{BACK}}$  goes high.

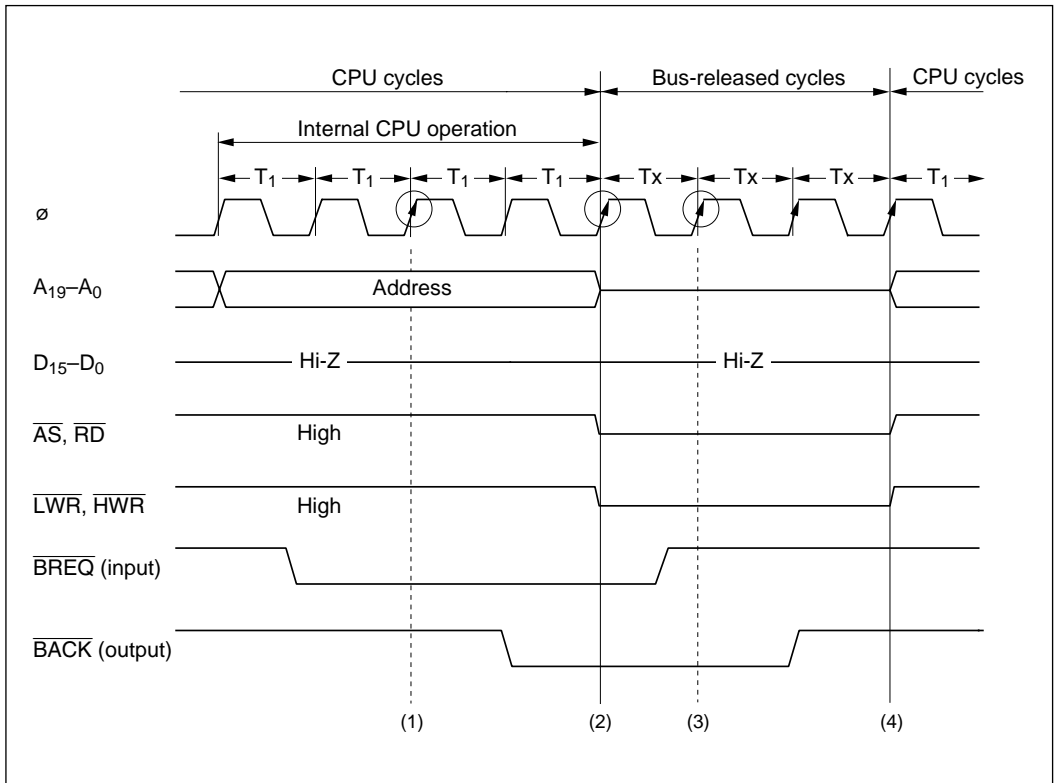


Figure 2-25 Bus Release during Internal CPU Operation

## (7) Notes

- The H8/500 CPU does not accept interrupts while in the bus-released state.
- The  $\overline{\text{BREQ}}$  signal must be held low until  $\overline{\text{BACK}}$  goes low. If  $\overline{\text{BREQ}}$  returns to the high level before  $\overline{\text{BACK}}$  goes low, the bus release operation may be executed incorrectly.

### 2.10.5 Reset State

In the reset state, the H8/500 CPU and all on-chip supporting modules are initialized and placed in the stopped state. The H8/500 CPU enters the reset state whenever the  $\overline{\text{RES}}$  pin goes low, unless the H8/500 CPU is currently in the hardware standby mode.

See section 4.2, “Reset” for further information on the reset state.

### 2.10.6 Power-Down State

The power-down state comprises three power-down modes: sleep mode, software standby mode, and hardware standby mode.

See section 21, “Power-Down State” for further information.

# Section 3 MCU Operating Modes

## 3.1 Overview

### 3.1.1 Selection of Operating Mode

The H8/539F has seven operating modes (modes 1 to 7).

Modes 1 to 6 are externally expanded modes in which external memory and peripheral devices can be accessed. Modes 1, 2, and 6 are expanded minimum modes, supporting a 64-kbyte address space. Modes 3, 4, and 5 are expanded maximum modes, supporting a maximum 1-Mbyte address space. When using modes 1 to 6, refer to section 3.6, Notes on Use of Externally Expanded Modes. See section 3.7, Notes on H8/539F S-Mask Model (Single Power Source Model), concerning all modes (1 to 7) in the H8/539F S-mask model (single power source model). Mode 7 is a single-chip mode: all ports are available for general-purpose input and output, but external addresses cannot be used.

Mode 0 is reserved for future use and must not be selected in the H8/539F.

Both the pin functions and address space vary depending on the mode. Table 3-1 summarizes the selection of operating modes.

**Table 3-1 Operating Mode Selection**

MCU Operating Mode	MD <sub>2</sub>	MD <sub>1</sub>	MD <sub>0</sub>	Description	CPU Operating Mode	On-Chip RAM	On-Chip ROM	Data Bus Width
Mode 0	0	0	0	—	—	—	—	—
Mode 1	0	0	1	Expanded minimum mode	Minimum mode	Enabled*1	Disabled	16 bits
Mode 2*3	0	1	0	Expanded minimum mode	Minimum mode	Enabled*1	Enabled	8 bits
Mode 3	0	1	1	Expanded maximum mode	Maximum mode	Enabled*1	Disabled	16 bits
Mode 4*3	1	0	0	Expanded maximum mode	Maximum mode	Enabled*1	Enabled	16 bits
Mode 5	1	0	1	Expanded maximum mode	Maximum mode	Enabled*1	Disabled	16 bits*2



**Table 3-1 Operating Mode Selection (cont)**

MCU Operating Mode	MD <sub>2</sub>	MD <sub>1</sub>	MD <sub>0</sub>	Description	CPU Operating Mode	On-Chip RAM	On-Chip ROM	Data Bus Width
Mode 6	1	1	0	Expanded minimum mode	Minimum mode	Enabled*1	Disabled	16 bits*2
Mode 7*3	1	1	1	Single-chip mode	Minimum mode	Enabled	Enabled	—

**Legend**

0: Low

1: High

—: Not available

- Notes:
1. If RAM enable bits 1 and 2 (RAME1 and RAME2) in the RAM control register (RAMCR) are cleared to 0, these addresses become external addresses.
  2. Eight-bit three-state-access address space after a reset.
  3. In the dual power model ( $V_{PP} = 12\text{ V}$ ), when pin settings are made for mode 2, 4, or 7 and 12 V is applied to the  $V_{PP}$  pin, flash memory can be programmed or erased. See section 19, “Flash Memory (H8/539F)” for details.

**S-Mask Model (Single Power Source):** Programming/erasing is enabled by making pin settings for mode 4 or 7, and setting the FLMCR register with the FWE pin in the high-level input state. For details, see section 20, Flash Memory (H8/539F S-Mask Model).

**3.1.2 Register Configuration**

The MCU operating mode can be monitored in the mode control register (MDCR). Table 3-2 summarizes this register.

**Table 3-2 Register Configuration**

Address	Name	Abbreviation	R/W	Initial Value
H'FF19	Mode control register	MDCR	R	Undetermined

### 3.2 Mode Control Register

The mode control register (MDCR) is an eight-bit register that indicates the current operating mode of the H8/539F. The MDCR bit structure is shown next.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	MDS2	MDS1	MDS0
Initial value	1	1	0	0	0	—*	—*	—*
R/W	—	—	—	—	—	R	R	R

Reserved bits
Mode select 2 to 0  
Bits indicating the current operating mode

Note: \* Determined by pins MD<sub>2</sub> to MD<sub>0</sub>. MDCR latches the inputs at the mode pins (MD<sub>2</sub> to MD<sub>0</sub>) at the rise of the RES signal.

- (1) **Bits 7 and 6—Reserved:** Read-only bits, always read as 1.
- (2) **Bits 5 to 3—Reserved:** Read-only bits, always read as 0.
- (3) **Bits 2 to 0—Mode Select 2 to 0 (MDS2 to MDS0):** These bits indicate the values of pins MD<sub>2</sub> to MD<sub>0</sub> latched at the rise of the  $\overline{\text{RES}}$  signal (the current operating mode).

MDS2 to MDS0 correspond to pins MD<sub>2</sub> to MD<sub>0</sub>.

MDS2 to MDS0 are read-only bits.

## **3.3 Operating Mode Descriptions**

### **3.3.1 Mode 1 (Expanded Minimum Mode)**

In mode 1 the data bus is 16 bits wide. The bus controller's byte area register (ARBT) is enabled in mode 1, so part of the address space can be accessed with an eight-bit bus width. The maximum address space supported in mode 1 is 64 kbytes.

The on-chip ROM is disabled in mode 1.

### **3.3.2 Mode 2 (Expanded Minimum Mode)**

In mode 2 the data bus is eight bits wide. The on-chip ROM is enabled.

The maximum address space supported in mode 2 is 64 kbytes.

The bus controller's byte-area register (ARBT) is disabled in mode 2.

### **3.3.3 Mode 3 (Expanded Maximum Mode)**

In mode 3 the data bus is 16 bits wide. The bus controller's byte area register (ARBT) is enabled in mode 3, so part of the address space can be accessed with an eight-bit bus width. The maximum address space supported in mode 3 is 1 Mbyte.

The on-chip ROM is disabled in mode 3.

### **3.3.4 Mode 4 (Expanded Maximum Mode)**

In mode 4 the data bus is 16 bits wide. The bus controller's byte area register (ARBT) is enabled in mode 4, so part of the address space can be accessed with an eight-bit bus width. The maximum address space supported in mode 4 is 1 Mbyte. The on-chip ROM is enabled.

### **3.3.5 Modes 5 and 6**

Mode 5 is functionally identical to mode 3, and mode 6 is functionally identical to mode 1. When the chip comes out of reset, however, the bus controller's byte area register (ARBT) is disabled in modes 5 and 6 and eight-bit, three-state access is performed throughout the address space. The byte area register can be enabled by setting the BCRE bit to 1 in the bus control register (BCR).

### **3.3.6 Mode 7 (Single-Chip Mode)**

The external address space cannot be accessed.

### 3.4 Pin Functions in Each Operating Mode

The pin functions of the I/O ports vary depending on the operating mode. Table 3-3 summarizes the functions in each mode. Selection of pin functions is described in section 10, “I/O Ports.”

**Table 3-3 Pin Functions in Each Mode**

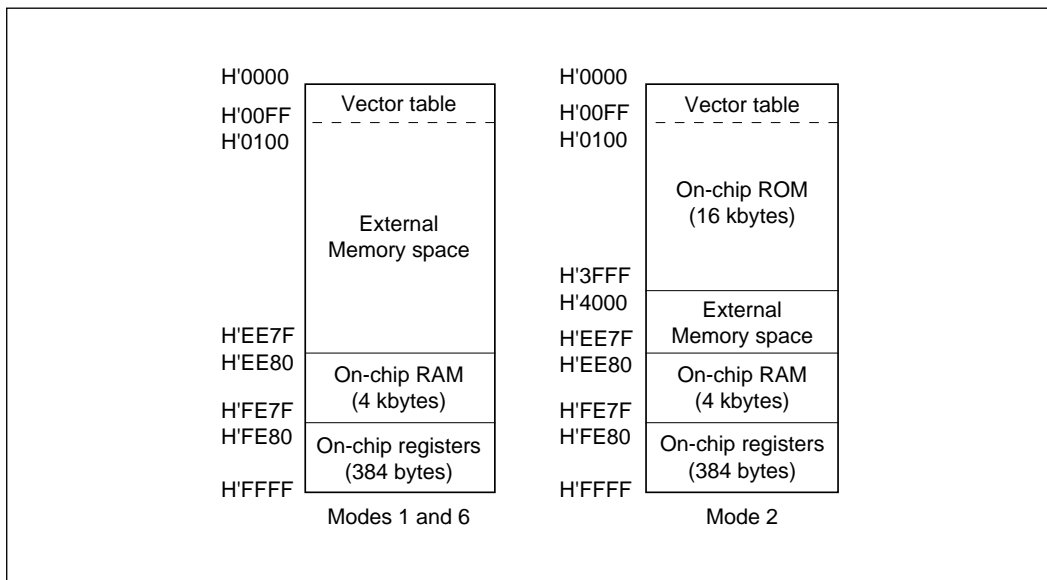
Port	Expanded Minimum Modes		Expanded Maximum Modes		Single-Chip Mode
	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7
Port 1	Data bus* <sup>6</sup> (D <sub>15</sub> to D <sub>8</sub> )	Data bus (D <sub>15</sub> to D <sub>8</sub> )	Data bus* <sup>6</sup> (D <sub>15</sub> to D <sub>8</sub> )	Data bus (D <sub>15</sub> to D <sub>8</sub> )	Input/output port
Port 2	Data bus (D <sub>7</sub> to D <sub>0</sub> )	Input/output port	Data bus (D <sub>7</sub> to D <sub>0</sub> )	Data bus (D <sub>7</sub> to D <sub>0</sub> )	Input/output port
Port 3	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>
Port 4	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>
Port 5	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>	Input/output port* <sup>1</sup>
Port 6	Input/output port* <sup>4</sup> IRQ <sub>2</sub> , IRQ <sub>3</sub>	Input/output port* <sup>4</sup> IRQ <sub>2</sub> , IRQ <sub>3</sub>	Input/output port* <sup>4</sup> IRQ <sub>2</sub> , IRQ <sub>3</sub>	Input/output port* <sup>4</sup> IRQ <sub>2</sub> , IRQ <sub>3</sub>	Input/output port* <sup>4</sup> IRQ <sub>2</sub> , IRQ <sub>3</sub>
Port 7	Input/output port* <sup>5</sup> IRQ <sub>0</sub> , IRQ <sub>1</sub> , ADTRG	Input/output port* <sup>5</sup> IRQ <sub>0</sub> , IRQ <sub>1</sub> , ADTRG	Input/output port* <sup>5</sup> IRQ <sub>0</sub> , IRQ <sub>1</sub> , ADTRG	Input/output port* <sup>5</sup> IRQ <sub>0</sub> , IRQ <sub>1</sub> , ADTRG	Input/output port* <sup>5</sup> IRQ <sub>0</sub> , IRQ <sub>1</sub> , ADTRG
Port 8	Input port* <sup>3</sup>	Input port* <sup>3</sup>	Input port* <sup>3</sup>	Input port* <sup>3</sup>	Input port* <sup>3</sup>
Port 9	Input port* <sup>3</sup>	Input port* <sup>3</sup>	Input port* <sup>3</sup>	Input port* <sup>3</sup>	Input port* <sup>3</sup>
Port A	Input/output port* <sup>2, *4</sup> BREQ, BACK, WAIT	Input/output port* <sup>2, *4</sup> BREQ, BACK, WAIT	Input/output port* <sup>1, *2</sup> BREQ, BACK, WAIT, address bus (A <sub>19</sub> to A <sub>16</sub> )	Input/output port* <sup>2, *4</sup> BREQ, BACK, WAIT, address bus (A <sub>19</sub> to A <sub>16</sub> )	Input/output port* <sup>2, *4</sup>
Port B	Address bus (A <sub>15</sub> to A <sub>8</sub> )	Input port/ address bus (A <sub>15</sub> to A <sub>8</sub> )	Address bus (A <sub>15</sub> to A <sub>8</sub> )	Input port/ address bus (A <sub>15</sub> to A <sub>8</sub> )	Input/output port
Port C	Address bus (A <sub>7</sub> to A <sub>0</sub> )	Input port/ address bus (A <sub>7</sub> to A <sub>0</sub> )	Address bus (A <sub>7</sub> to A <sub>0</sub> )	Input port/ address bus (A <sub>7</sub> to A <sub>0</sub> )	Input/output port

Notes on next page.

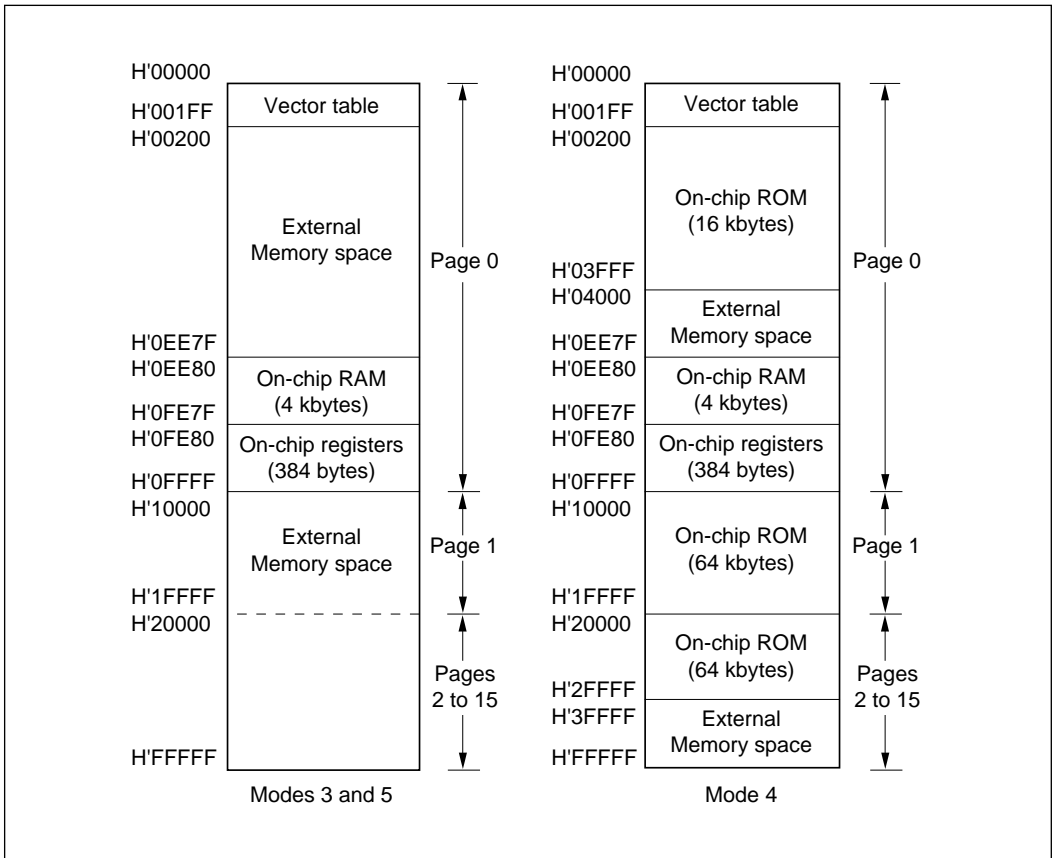
- Notes:
1. Also used for timer input/output.
  2. Also used for serial communication interface.
  3. Also used for A/D conversion.
  4. Also used for timer output and PWM timer output.
  5. Also used for serial communication interface and PWM timer output.
  6. In modes 5 and 6, the external bus space has a 16-bit bus width, but an 8-bit bus width is set after a reset. In this case, the upper half of the data bus (D15 to D8) is enabled, and the lower half (D7 to D0) is disabled. After the BCRE bit in the bus control register (BCR) has been set to 1 by software, the bus width can be changed to 16 bits (D15 to D0) by a byte area top register (ARBT) setting. In modes 1, 3, and 4, the external bus space has a 16-bit bus width (D15 to D0) after a reset, but this can be changed to 8 bits by an ARBT setting. In this case, the upper half of the data bus (D15 to D8) is enabled, and the lower half (D7 to D0) is disabled. For details of the settings, see section 16, Bus Controller.

### 3.5 Memory Map in Each Mode

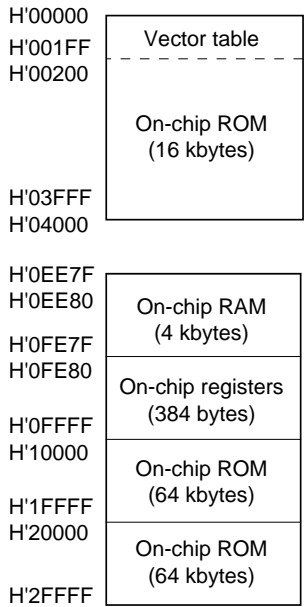
Figure 3-1 shows a memory map for the expanded minimum modes (modes 1, 6, and 2). Figure 3-2 shows a memory map for the expanded maximum modes (modes 3, 5, and 4). Figure 3-3 shows a memory map for single-chip mode (mode 7).



**Figure 3-1 Memory Map in Expanded Minimum Modes**



**Figure 3-2 Memory Map in Expanded Maximum Modes**



Mode 7

**Figure 3-3 Memory Map in Single-Chip Mode**

## 3.6 Notes on Use of Externally Expanded Modes

### 3.6.1 H8/539F (Dual Power Source Model)

The following points must be observed when using the H8/539F in an externally expanded mode (modes 1 to 6).

- Initialize the  $\phi$ OE bit \*1 to 1 before an external space access.

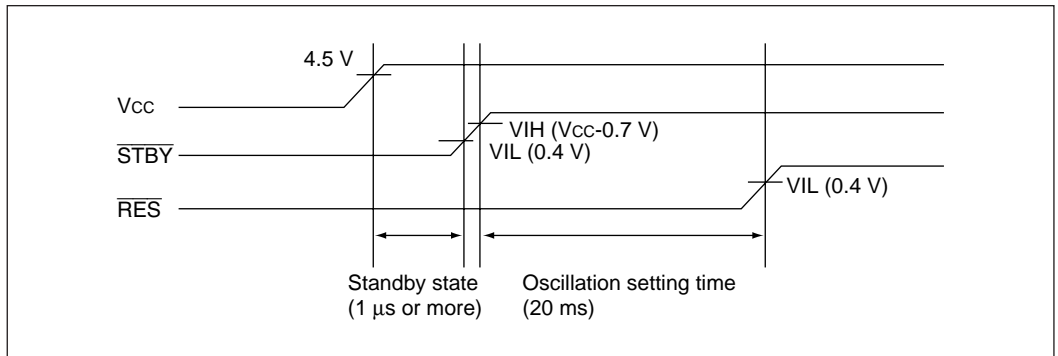
When using the H8/539F in an externally expanded mode with on-chip ROM enabled (mode 2 or 4), the  $\phi$ OE bit in the  $\phi$  control register ( $\phi$ CR) must be initialized to 1 before accessing the external bus space.

- In modes with on-chip ROM disabled, enter standby mode at power-on.

When using the H8/539F in an externally expanded mode with on-chip ROM disabled (mode 1, 3, 5, or 6), hardware standby mode must be entered at power-on. Figure 3-4 shows the power-on timing.

Note that when  $\phi$  output is inhibited, the  $\phi$  pin goes to the high-impedance state and external space accesses will not function correctly .

See section 3.7, Notes on H8/539F S-Mask Model (Single Power Source Model), for further information concerning the H8/539F S-mask model (single power source model).



**Figure 3-4 Power-On Timing**

Note: 1. For details, see section 10.14,  $\phi$  Pin.



### 3.7 Notes on H8/539F S-Mask Model (Single Power Source Model)

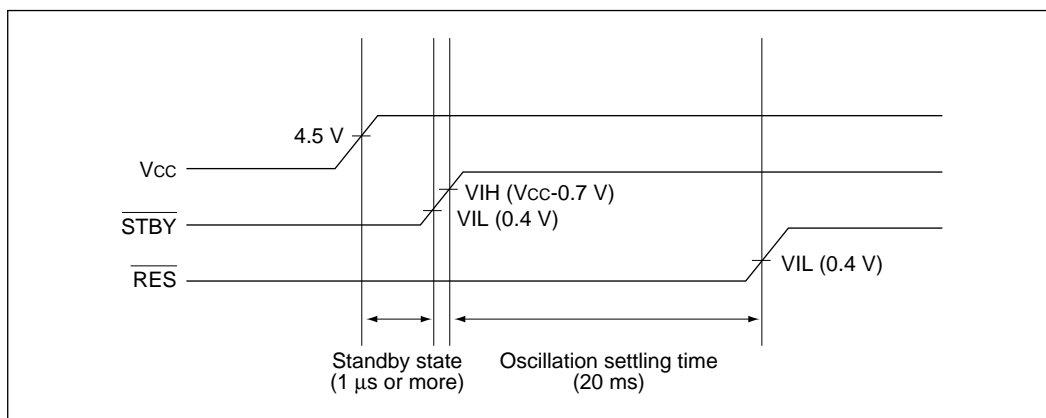
The following points should be noted when using the H8/539F S-mask model (single power source model) in any of modes 1 to 7.

At power-on, set hardware standby mode in all modes (1 to 7).

With the H8/539F S-mask model (single power source model), hardware standby mode must be set at power-on. Figure 3-5 shows the power-on timing.

Note that when  $\phi$  output is disabled the  $\phi$  pin goes to the high-impedance state, and external space access will not function correctly.

For information concerning the H8/539F (dual power source model), see section 3.6, Notes on Use of Externally Expanded Mode.



**Figure 3-5 Power-On Timing**

# Section 4 Exception Handling

## 4.1 Overview

There are five types of exceptions: reset, address error, trace, interrupt, and instruction exceptions. There are three types of instruction exceptions: invalid instruction, trap instruction, and DIVXU instruction with zero divisor.

Handling of these exceptions is described next.

### 4.1.1 Exception Handling Types and Priority

Table 4-1 lists the types of exception handling for exceptions other than instruction exceptions, and indicates their priority. The system assigns a reserved priority to each of these exception types. If two or more exceptions occur simultaneously, they are accepted and handled in priority order.

Table 4-2 lists the types of instruction exception handling. Instruction exceptions cannot occur simultaneously, so there is no priority order.

**Table 4-1 Exception Types and Priority**

Priority	Exception Type	Source	Start of Exception Handling
High ↑ ↓ Low	Reset	RES input	Rising edge of $\overline{\text{RES}}$ signal
	Address error	Invalid access (address error)	End of instruction execution
	Trace	Trace bit (T) = 1 in SR	End of instruction execution
	Interrupt	External or internal interrupt request	End of instruction execution or end of exception handling

**Table 4-2 Instruction Exceptions**

Exception Type	Source	Start of Exception Handling
Invalid instruction	Fetching of undefined code	Start of execution of instruction with undefined code
Trap instruction	Trap instruction	Start of execution of trap instruction
Zero divide	DIVXU instruction	Start of execution of DIVXU instruction with zero divisor

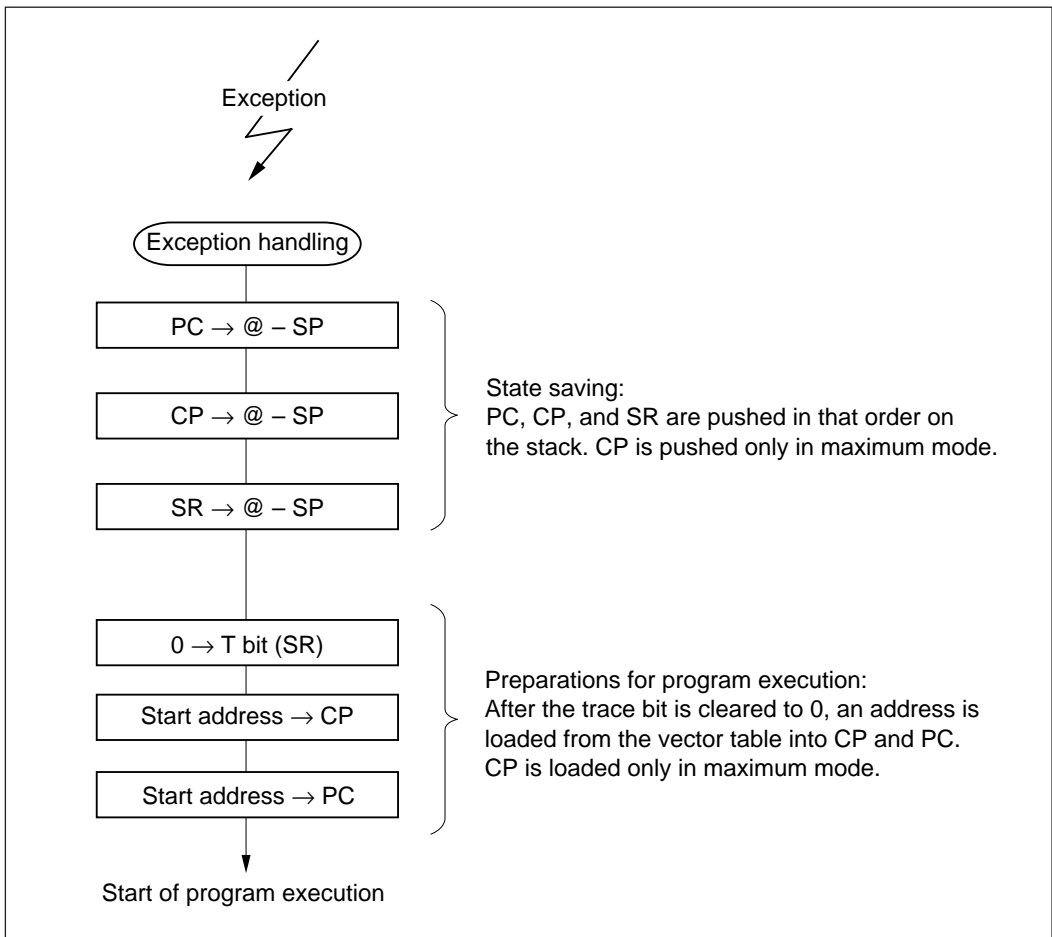
## 4.1.2 Exception Handling Operation

Exception handling can originate from a variety of sources.

Exception handling other than reset exception handling is described next. For reset exception handling, see section 4.2, “Reset.”

Figure 4-1 is a flowchart of the handling of exceptions other than a reset.

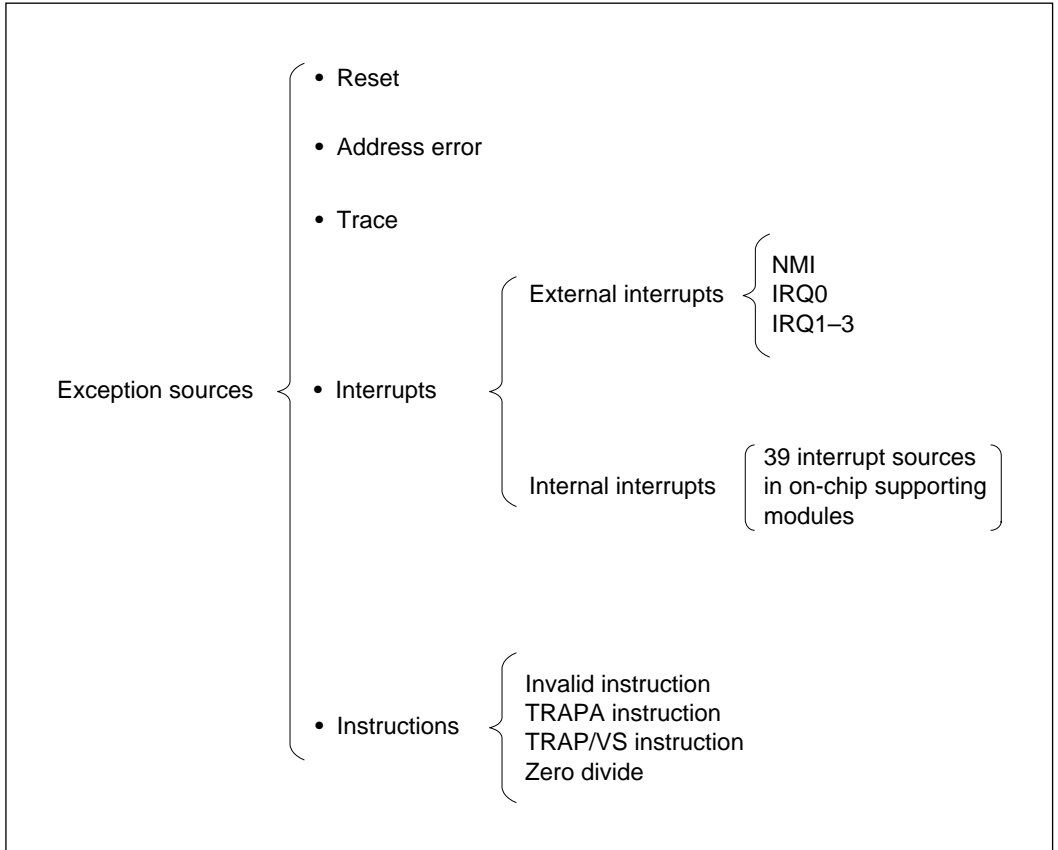
In minimum mode, the program counter (PC) and status register (SR) are saved on the stack. In maximum mode the code page register (CP), PC, and SR are saved on the stack. Next the T bit in the status register is cleared to 0, the start address corresponding to the exception source is read from the exception vector table, and program execution begins from the indicated address.



**Figure 4-1 Exception Handling Flowchart**

### 4.1.3 Exception Sources and Vector Table

Figure 4-2 classifies the exception sources. Table 4-3 shows the exception vector table. The vector addresses differ between minimum and maximum modes. In maximum mode the vector table is located in page 0. For internal interrupt vectors, see table 6-3, “Interrupt Priorities and Vector Addresses.”



**Figure 4-2 Classification of Exception Sources**

**Table 4-3 Exception Vector Table**

<b>Exception Source</b>	<b>Vector Address</b>	
	<b>Minimum Mode</b>	<b>Maximum Mode</b>
Reset (initial PC value)	H'0000–H'0001	H'0000–H'0003
(Reserved for system)	H'0002–H'0003	H'0004–H'0007
Invalid instruction	H'0004–H'0005	H'0008–H'000B
DIVXU instruction (zero divisor)	H'0006–H'0007	H'000C–H'000F
TRAP/VS instruction	H'0008–H'0009	H'0010–H'0013
(Reserved for system)	H'000A–H'000B ⋮	H'0014–H'0017 ⋮
	H'000E–H'000F	H'001C–H'001F
Address error	H'0010–H'0011	H'0020–H'0023
Trace	H'0012–H'0013	H'0024–H'0027
(Reserved for system)	H'0014–H'0015	H'0028–H'002B
External interrupt: NMI	H'0016–H'0017	H'002C–H'002F
(Reserved for system)	H'0018–H'0019 ⋮	H'0030–H'0033 ⋮
	H'001E–H'001F	H'003C–H'003F
TRAPA instruction (16 sources)	H'0020–H'0021 ⋮	H'0040–H'0043 ⋮
	H'003E–H'003F	H'007C–H'007F
External interrupt: IRQ0	H'0040–H'0041	H'0080–H'0083
WDT interval interrupt	H'0042–H'0043	H'0084–H'0087
External interrupts: IRQ1	H'0048–H'0049	H'0090–H'0093
IRQ2	H'004A–H'004B	H'0094–H'0097
IRQ3	H'004C–H'004D	H'0098–H'009B
Internal interrupts	H'0044–H'0045 H'0050–H'0051 ⋮	H'0088–H'008B H'00A0–H'00A3 ⋮
	H'009E–H'009F	H'013C–H'013F

## 4.2 Reset

### 4.2.1 Overview

A reset has the highest exception priority.

Reset exception handling is described below.

When the  $\overline{\text{RES}}$  pin goes low, all processing halts and the chip enters the reset state. A reset initializes the internal state of the H8/500 CPU and the registers of on-chip supporting modules. When the  $\overline{\text{RES}}$  pin rises from low to high, the H8/500 CPU begins reset exception handling.

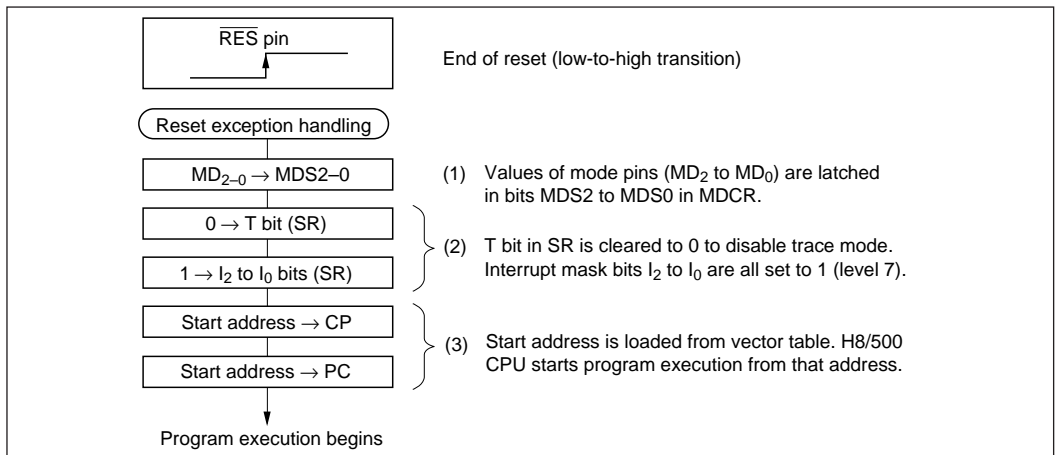
### 4.2.2 Reset Sequence

The chip enters the reset state when the  $\overline{\text{RES}}$  pin goes low.

To ensure that the chip is reset, the  $\overline{\text{RES}}$  pin should be held low for at least 20 ms at power-up. To reset the chip during operation, the  $\overline{\text{RES}}$  pin should be held low for at least six system clock cycles ( $6\phi$ ) in the H8/539F. In the H8/539F S-mask model, the  $\overline{\text{RES}}$  pin should be held low for at least 20 system clock cycles ( $20\phi$ ). When an external clock is used, the  $\overline{\text{RES}}$  pin should be held low for at least the external clock output setting delay time ( $t_{\text{DEXT}}$ ) at power-up and in a reset start from the standby state.

See appendix G, “Pin States” for the states of the pins in the reset state.

When the  $\overline{\text{RES}}$  pin rises to the high level after being held low for the necessary time, the H8/500 CPU begins reset exception handling. Figure 4-3 shows the sequence of operations at the end of the reset state.

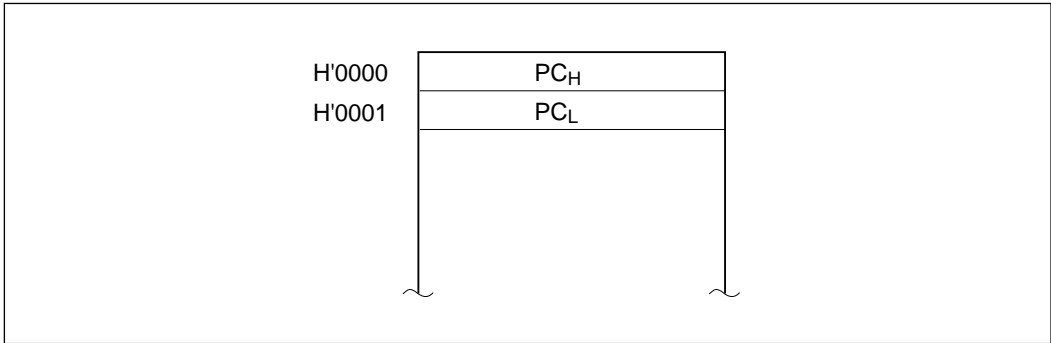


**Figure 4-3 Reset Exception Handling Flowchart**

The vector table contents differs between minimum and maximum mode. The vector table contents in each mode are described next.

**(1) Minimum Mode:** Figure 4-4 shows the reset vector in minimum mode.

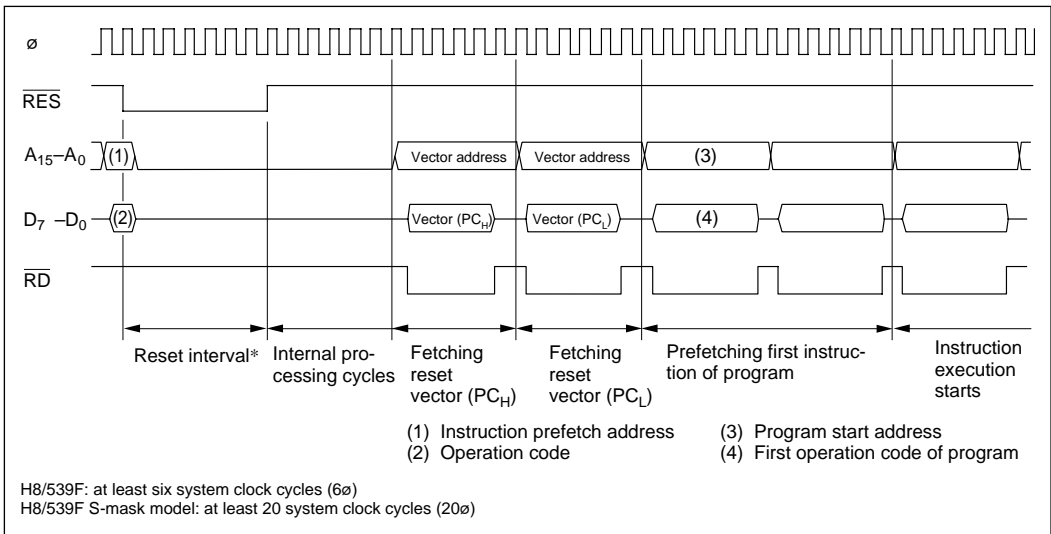
In minimum mode the reset vector is located at addresses H'0000 and H'0001. When exception handling begins, the H8/500 CPU copies the reset vector into the program counter (PC). Program execution then starts from the PC address.



**Figure 4-4 Reset Vector in Minimum Mode**

Figure 4-5 shows the reset sequence in minimum mode.

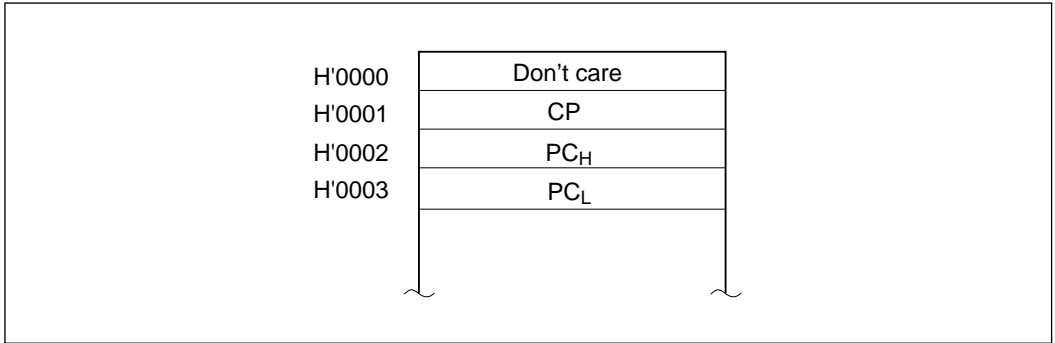
Figure 4-5 shows the case in which the program area and stack area are both located in the eight-bit-bus three-state-access address space.



**Figure 4-5 Reset Sequence in Minimum Mode**

(2) **Maximum Mode:** Figure 4-6 shows the reset vector in maximum mode.

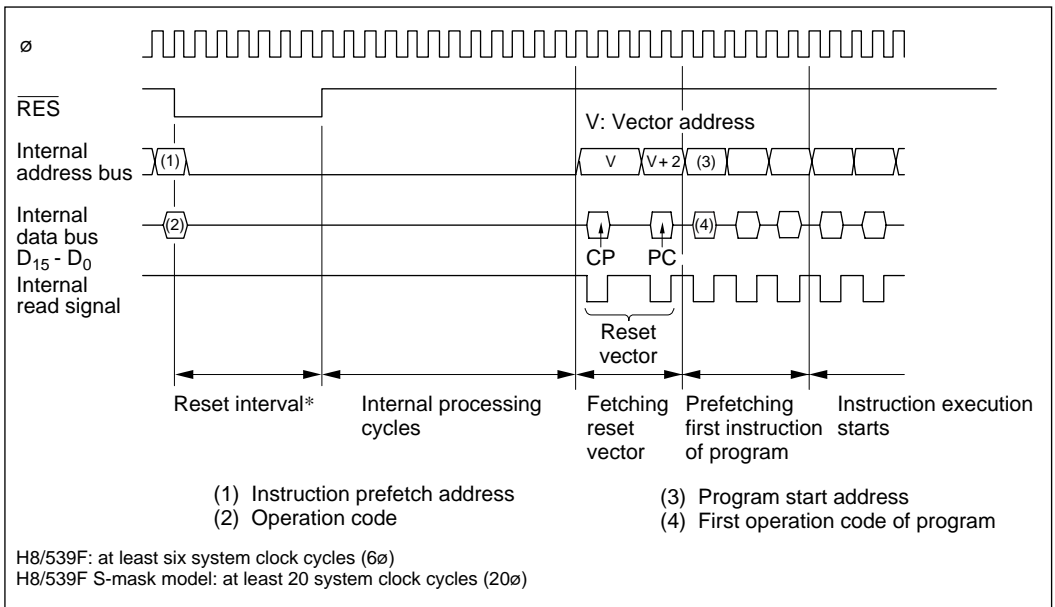
In maximum mode the reset vector is located at addresses H'0000 to H'0003. When exception handling begins, the H8/500 CPU copies the reset vector into the code page register (CP) and program counter (PC), ignoring the vector data at H'0000. Program execution then starts from the CP and PC address.



**Figure 4-6 Reset Vector in Maximum Mode**

Figure 4-7 shows the reset sequence in maximum mode.

Figure 4-7 shows the case in which the program area and stack area are both located in the 16-bit-bus two-state-access address space.



**Figure 4-7 Reset Sequence in Maximum Mode**



### 4.2.3 Interrupts after Reset

If an interrupt is accepted after a reset but before the stack pointer (SP) is initialized, the program counter and status register will not be saved correctly, leading to a program crash. This danger can be avoided as explained next.

When the chip comes out of the reset state all interrupts, including NMI, are disabled, so the first instruction is always executed. Crashes can be avoided by using this first instruction to initialize SP. In minimum mode, the first instruction after a reset should initialize SP. In maximum mode, the first instruction after a reset should initialize the stack page register (TP), and the next instruction should initialize SP.

#### Examples:

1. Minimum mode

```
.ORG H'0000
MOV.W  #H'FE80, SP
.
```

2. Maximum mode

```
.ORG H'0000
LDC.B  #H'00, TP
MOV.W  #H'FE80, SP
.
```

## 4.3 Address Error

An address error occurs when invalid access is attempted. There are three types of address errors:

1. Address error in instruction prefetch
2. Address error in word data access
3. Address error in single-chip mode

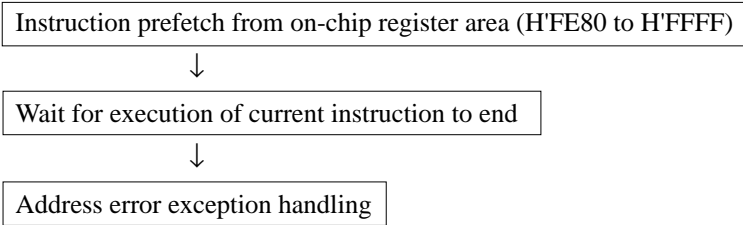
When an address error occurs, the H8/500 CPU begins address error exception handling and clears the T bit of the status register to 0. The interrupt mask level in bits I2 to I0 is not changed.

Each type of address error is described next.

### 4.3.1 Address Error in Instruction Prefetch

An attempt to prefetch an instruction from the on-chip registers at addresses H'FE80 to H'FFFF causes an address error.

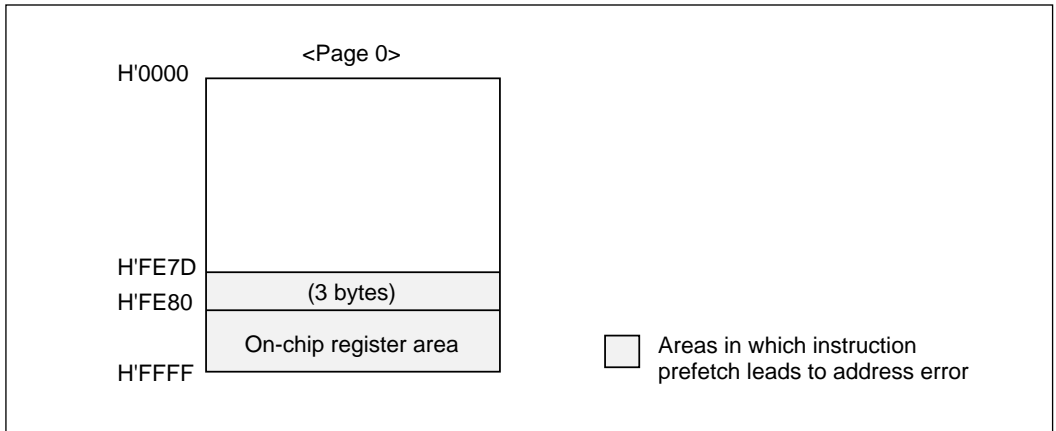
The address error exception handling sequence for this case is:



The PC value pushed on the stack is the start address of the instruction immediately following the last instruction executed.

Program code should not be located in addresses H'FE7D to H'FE7E. If program code is located in these addresses, instruction prefetch will be attempted in the on-chip register area, causing an address error.

Figure 4-8 shows the areas in which instruction prefetch leads to an address error.

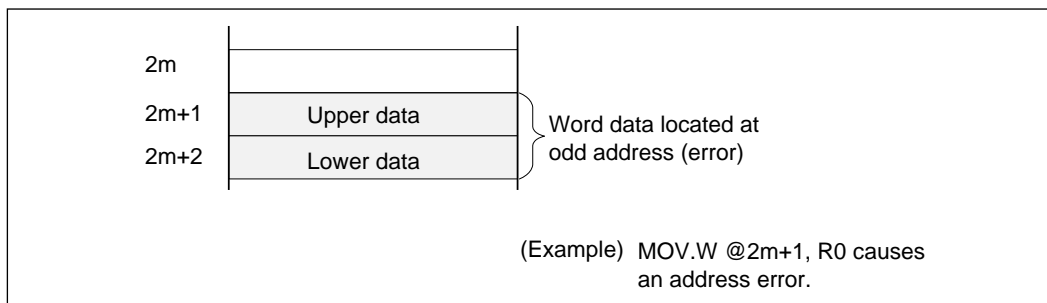


**Figure 4-8 Areas in which Instruction Prefetch Leads to Address Error**

### 4.3.2 Address Error in Word Data Access

An address error occurs if an attempt is made to access word data starting at an odd address. The PC value pushed on the stack is the address of the next instruction after the instruction that attempted to access word data at an odd address.

Figure 4-9 shows an example of illegal location of word data.



**Figure 4-9 Example of Illegal Location of Word Data**

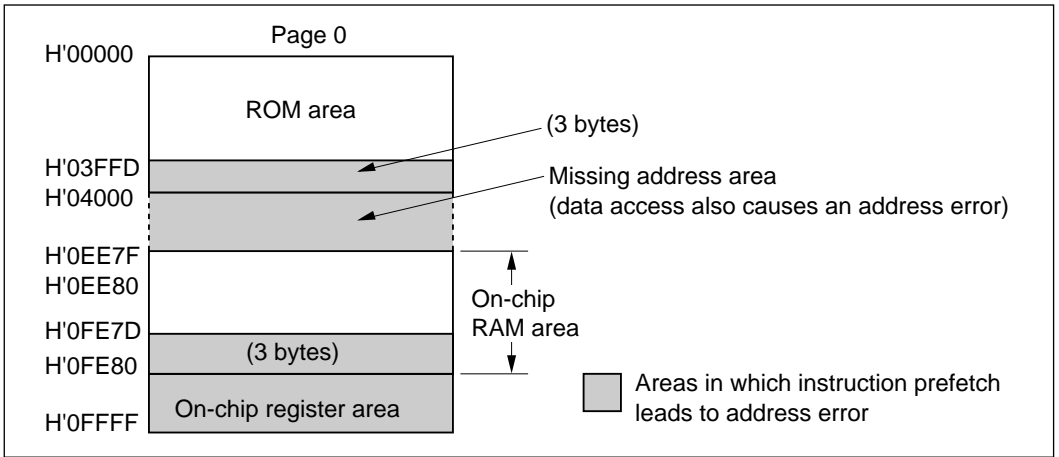
### 4.3.3 Address Error in Single-Chip Mode

In single-chip mode there is no external memory, so in addition to the word access address errors described in section 4.3.2, Address Error in Word Data Access, address errors can occur due to access to missing areas in the address space.

**(1) Access to Addresses H'04000 to H'0EE7F and H'30000 to H'FFFFFF:** In single-chip mode these addresses form a missing address area; they are assigned neither to on-chip memory nor to on-chip registers.

Instruction prefetch, byte data access, or word data access in the missing address area causes an address error. An address error also occurs if an instruction is located in the last three bytes of on-chip ROM in page 0, because the H8/500 CPU will attempt to prefetch the next instruction from addresses H'04000 to H'04002 in the missing address area.

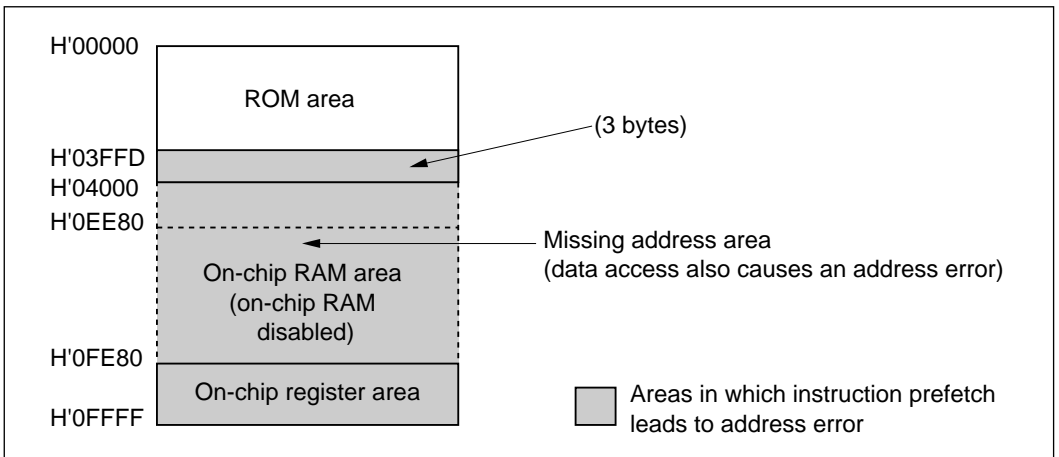
The same type of error will not occur if an instruction is located in the last three bytes of on-chip ROM in page 1 or page 2.



**Figure 4-10 Areas in which Instruction Prefetch Leads to Address Error (Single-Chip Mode)**

(2) **Access to Disabled RAM Area:** When the on-chip RAM area is disabled in single-chip mode, addresses H'04000 to H'0FE7F are also a missing area. Instruction prefetch, byte data access, or word data access in this missing address area causes an address error. An address error also occurs if an instruction is located in the last three bytes of on-chip ROM in page 0, because the H8/500 CPU will attempt to prefetch the next instruction from addresses H'04000 to H'04002 in the missing address area.

The same type of error will not occur if an instruction is located in the last three bytes of on-chip ROM in page 1 or page 2.



**Figure 4-11 Areas in which Instruction Prefetch Leads to Address Error (Single-Chip Mode with On-Chip RAM Disabled)**

## 4.4 Trace

Trace mode can be used by a debug program, for example, to monitor the execution of a program under test.

**(1) Trace Mode:** When the trace bit (T bit) in the status register (SR) is set to 1, the H8/500 CPU operates in trace mode. A trace exception occurs at the completion of each instruction.

In trace exception handling the T bit in SR is cleared to 0 to disable trace mode. The interrupt mask level in bits I2 to I0 is not changed, however; interrupts are accepted during trace exception handling.

The trace exception-handling routine should end with an RTE instruction. When the trace routine returns with the RTE instruction, the status register is popped from the stack and trace mode resumes.

**(2) Contention with Address Error Exception Handling:** Address error exception handling occurs at the end of a bus cycle, so it does not normally conflict with trace exception handling. One instruction is always executed after returning from exception handling, however, so contention may occur at this point, requiring special consideration.

If address error and trace exceptions both occur at the end of an instruction, because of the priority relationship between these exceptions, address error exception handling is carried out. Trace mode is disabled during execution of the instruction that caused the address error and during the address error exception handling routine. After return from address error exception handling, one instruction is executed, then trace mode resumes.

## 4.5 Interrupts

There are five external sources of interrupt exception handling ( $\overline{\text{NMI}}$ ,  $\overline{\text{IRQ}}_0$ ,  $\overline{\text{IRQ}}_1$ ,  $\overline{\text{IRQ}}_2$ ,  $\overline{\text{IRQ}}_3$ ) and 39 sources in the on-chip supporting modules. Table 4-4 classifies the interrupt sources. The on-chip supporting modules that can request interrupts are the 16-bit integrated timer pulse unit (IPU), serial communication interfaces 1 and 2 (SCI1 and SCI2), A/D converter, and watchdog timer (WDT).

NMI is the highest-priority interrupt and is always accepted\*. The other 43 interrupt sources are controlled by the interrupt controller. The interrupt controller arbitrates between simultaneous interrupts by means of internal registers in which interrupt priorities are assigned to each module.

The interrupt priorities are set in interrupt priority registers A to F (IPRA to IPRF) in the interrupt controller. An interrupt priority level from 7 to 0 can be assigned to IRQ0. A single priority level from 7 to 0 can be assigned collectively to IRQ1, IRQ2, and IRQ3. Independent priority levels from 7 to 0 can also be assigned to each of the on-chip supporting modules.

Note : \* The exception is when programming or erasing flash memory, in which case NMI input is disabled. See section 19.4.9, “NMI Input Masking” and section 20.4.8, “NMI Input Masking” for details.

The interrupt controller also controls the starting of the data transfer controller (DTC) in response to an interrupt. The DTC can transfer data in either direction between memory and I/O without using the CPU.

Whether to start the DTC can be selected on an individual interrupt basis in data transfer enable registers A to F (DTEA to DTEF) in the interrupt controller. The DTC is started if the corresponding bit in DTEA to DTEF is set to 1. If this bit is cleared to 0, interrupt exception handling is carried out. A few interrupts, including NMI, cannot start the DTC. The CPU halts during DTC operation.

For details of DTC interrupts, see section 7, “Data Transfer Controller.” Interrupt controller functions are detailed in section 6, “Interrupt Controller.”

**Table 4-4 Interrupt Sources**

<b>Interrupt Category</b>		<b>Number of Sources</b>
External interrupts	NMI	1
	IRQ0	1
	IRQ1–IRQ3	3
Internal interrupts	IPU	29
	SCI1	4
	SCI2/SCI3	4
	A/D converter	1
	WDT	1

## 4.6 Invalid Instructions

An invalid instruction is an instruction with an undefined operation code or illegal addressing mode. If an attempt is made to execute an invalid instruction, the H8/500 CPU starts invalid instruction exception handling. The PC value pushed on the stack is the value of the program counter when the invalid instruction code was detected.

In the invalid instruction exception-handling sequence the T bit of the status register is cleared to 0, but the interrupt mask level (I2 to I0) is not changed.

## 4.7 Trap Instructions and Zero Divide

When the TRAPA or TRAP/VS instruction is executed, the H8/500 CPU starts trap exception handling. If an attempt is made to execute a DIVXU instruction with a zero divisor, the H8/500 CPU starts zero divide exception handling.

In the exception-handling sequences for these exceptions the T bit of the status register is cleared to 0, but the interrupt mask level (I2 to I0) is not changed.

If a normal interrupt is requested during execution of a trap or zero-divide instruction, interrupt handling begins after the exception-handling sequence for the trap or zero-divide instruction has been executed.

**(1) TRAPA Instruction:** When the TRAPA instruction is executed, the H8/500 CPU starts exception handling according to the CPU operating mode.

The TRAPA instruction includes a vector number from 0 to 15. The start address is read from the corresponding location in the vector table.

**(2) TRAP/VS Instruction:** When the TRAP/VS instruction is executed, the H8/500 CPU starts exception handling if the overflow (V) flag in the condition code register (CCR) is set to 1.

If the V flag is cleared to 0, no exception occurs and the next instruction is executed.

**(3) DIVXU Instruction with Zero Divisor:** The H8/500 CPU starts exception handling if an attempt is made to divide by zero in a DIVXU instruction.

## 4.8 Cases in which Exception Handling is Deferred

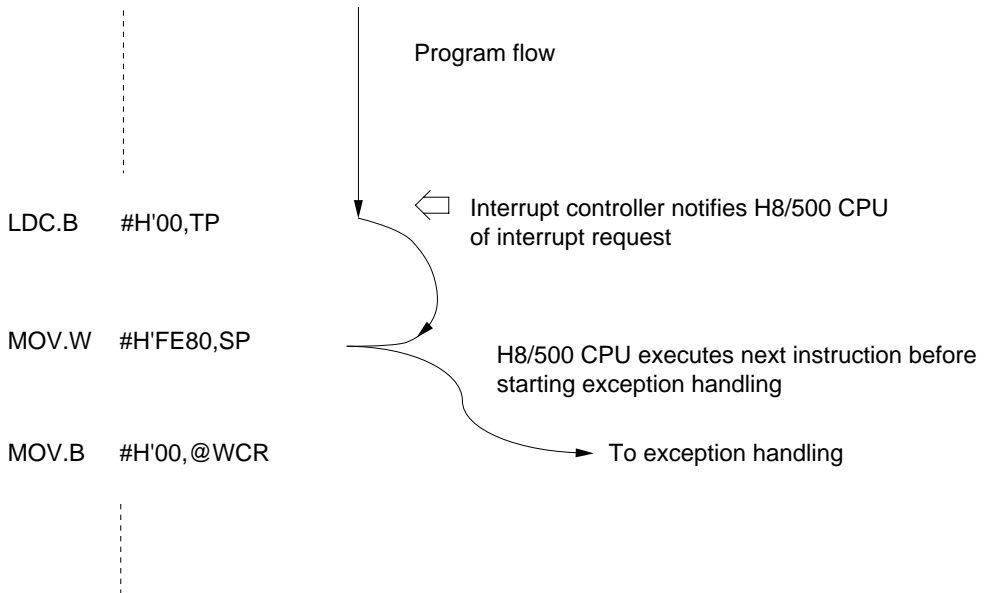
Exception handling of address errors, trace exceptions, external interrupt requests (NMI,  $\overline{\text{IRQ}}_0$ ,  $\overline{\text{IRQ}}_1$ ,  $\overline{\text{IRQ}}_2$ ,  $\overline{\text{IRQ}}_3$ ), and internal interrupt requests (39 sources) is not carried out immediately after execution of an interrupt-disabling instruction, reset exception, or data transfer cycle, but is deferred until after the next instruction has been executed.

### 4.8.1 Instructions that Disable Exception Handling

Interrupts are disabled immediately after the execution of five instructions: XORC, ORC, ANDC, LDC, and RTE.

After executing one of these instructions, the H8/500 CPU always executes the next instruction. If the next instruction is also one of these five, the next instruction after that is executed too. Exception handling starts after the next instruction that is not one of these five has been executed. See the following example.

#### Example:





### 4.8.2 Disabling of Exceptions Immediately after a Reset

After carrying out reset exception handling, the H8/500 CPU always executes the initial instruction.

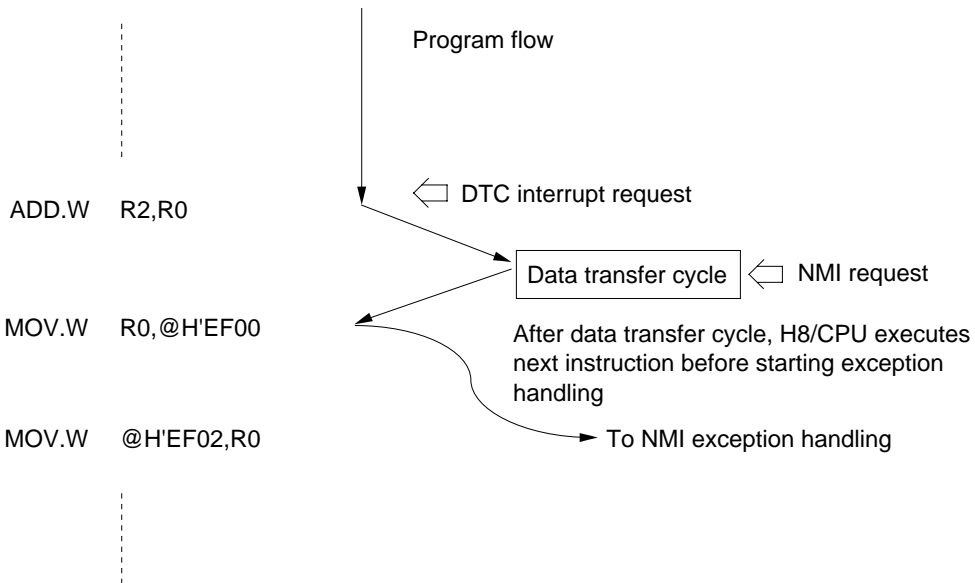
If an interrupt is accepted after a reset but before SP is initialized, the program counter and status register will not be saved correctly, leading to a program crash. To prevent this, in minimum mode the first instruction after a reset should initialize SP. In maximum mode, the first instruction after a reset should be an LDC instruction initializing TP, and the next instruction should initialize SP.

### 4.8.3 Disabling of Interrupts after a Data Transfer Cycle

If an interrupt starts the data transfer controller and a second interrupt is requested during the data transfer cycle, when the data transfer cycle ends, the H8/500 CPU always executes the next instruction before handling the second interrupt.

Even if a nonmaskable interrupt (NMI) occurs during a data transfer cycle, it is not accepted until the next instruction has been executed. An example is shown next.

#### Example:



## 4.9 Stack Status after Completion of Exception Handling

The status of the stack after exception handling is described next.

Table 4-5 shows the stack after completion of exception handling for various types of exceptions in minimum and maximum modes.

**Table 4-5 Stack after Exception Handling**

Exception Source	Minimum Mode	Maximum Mode
Trace, interrupt, trap instruction, DIVXU (zero divide)		
Note: The RTE instruction returns to the next instruction after the instruction being executed when the exception occurred.		
Invalid instruction		
Note: The CP and PC values pushed on the stack are not necessarily the address of the first byte of the invalid instruction.		
Address error		
Note: The CP and PC values pushed on the stack are the address of the next instruction after the last instruction executed.		

### **4.9.1 PC Value Pushed on Stack for Trace, Interrupts, Trap Instructions, and Zero Divide Exceptions**

The PC value pushed on the stack for a trace, interrupt, trap, or zero divide exception is the address of the next instruction at the time when the interrupt was accepted.

### **4.9.2 PC Value Pushed on Stack for Address Error and Invalid Instruction Exceptions**

The PC value pushed on the stack for an address error or invalid instruction exception differs depending on the conditions when the exception occurs.

## **4.10 Notes on Use of the Stack**

When using the stack, pay attention to the following points. Mistakes may lead to address errors when the stack is accessed, or may cause system crashes.

1. Always set SP on an even address.

If SP indicates an odd address, an address error will occur when the H8/500 CPU accesses the stack during interrupt handling or for a subroutine call. To keep SP pointing to an even address, always use word data size when saving or restoring register data or other data to or from the stack.

2. @-SP and @SP+ addressing modes

To keep SP pointing to an even address, in the @-SP and @SP+ addressing modes the H8/500 CPU performs word access even if the instruction specifies byte size.

This is not true in the @-Rn (pre-decrement) and @Rn+ (post-increment) addressing modes when Rn is a register from R0 to R6.

## Section 5 H8 Multiplier (H8MULT)

### 5.1 Overview

The on-chip multiplier module (H8MULT) can perform 16-bit  $\times$  16-bit signed or unsigned multiply and multiply-accumulate operations. These operations can be speeded up by a bus-stealing function.

#### 5.1.1 Features

Features of the H8MULT module are listed below.

- 16-bit  $\times$  16-bit multiplication executed in two clock cycles

Signed or unsigned multiplication can be selected. Up to three multiplier values can be designated in advance.

- Multiply-and-accumulate operations can be executed in three clock cycles

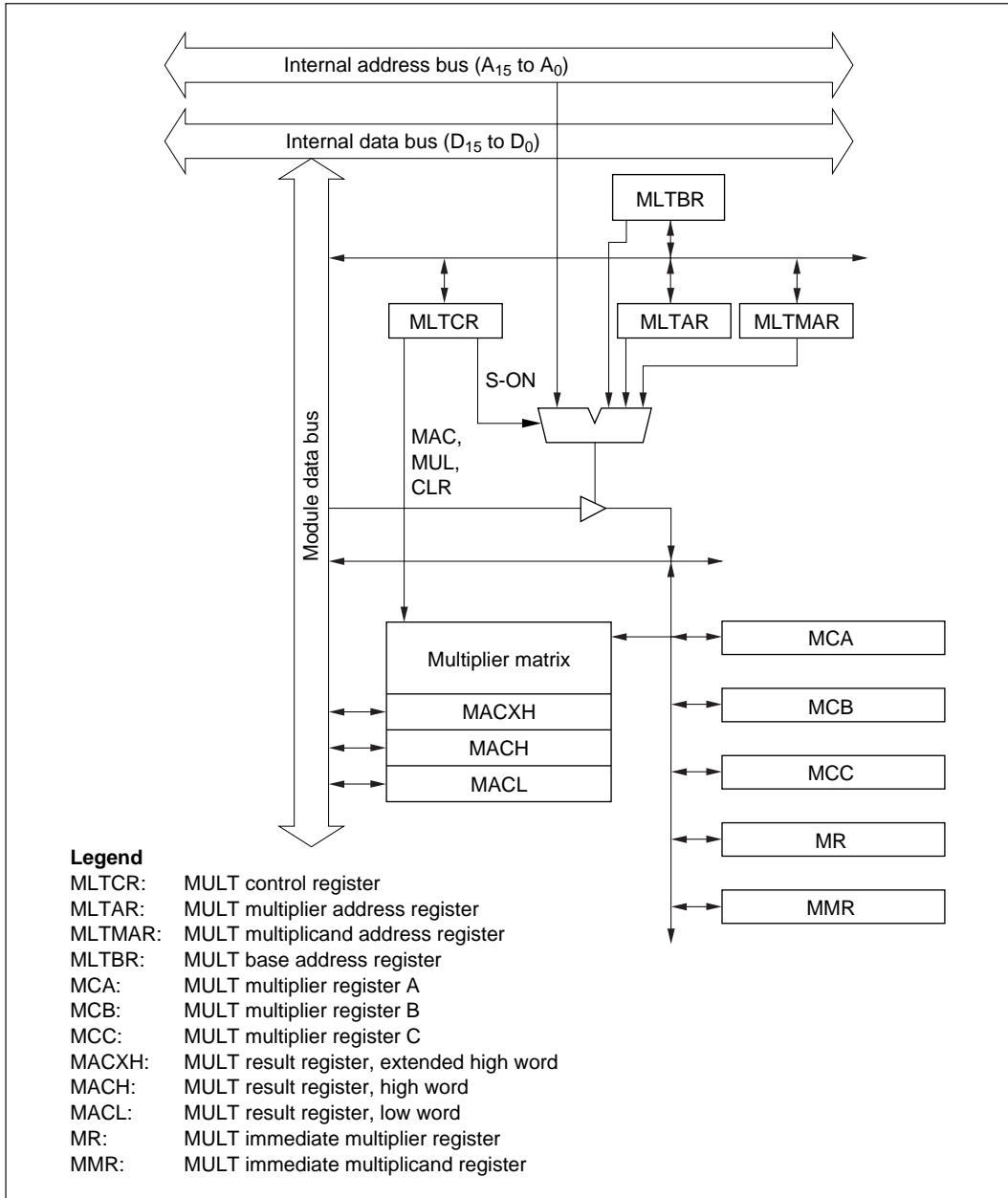
Saturating or non-saturating operation can be selected. The results of non-saturating multiply-accumulate operations are stored in 42-bit form. The results of saturating multiply-accumulate operation are stored in 32-bit form. Up to three multiplier values can be designated in the H8MULT registers in advance, an arrangement ideally suited for second-order digital filtering.

- Built-in bus-stealing function

For higher-speed operation, the bus-stealing function enables multipliers and multiplicands to be loaded into H8MULT while the CPU is reading memory.

## 5.1.2 Block Diagram

Figure 5-1 shows a block diagram of the H8MULT module.



**Figure 5-1 H8MULT Block Diagram**

### 5.1.3 Register Configuration

Table 5-1 summarizes the internal registers of the H8MULT module. The type of operation (multiply or multiply-accumulate, signed or unsigned) and the bus-stealing function can be selected by register settings.

**Table 5-1 H8MULT Registers**

Type	Address	Name	Abbreviation	R/W	Initial Value
Control registers	H'FFA0	MULT control register	MLTCR	R/W	H'38
	H'FFA1	MULT base address register	MLTBR	R/W	H'00
	H'FFA2	MULT multiplier address register	MLTAR	R/W	H'00
	H'FFA3	MULT multiplicand address register	MLTMAR	R/W	H'00
Arithmetic registers*1	H'FFB0	MULT multiplier register A	MCA	R/W	H'0000
	H'FFB2	MULT multiplier register B	MCB	R/W	H'0000
	H'FFB4	MULT multiplier register C	MCC	R/W	H'0000
	H'FFB6	MULT result register, extended high word	MACXH	R/W	Undetermined
	H'FFB8	MULT result register, high word	MACH	R/W*2	Undetermined
	H'FFBA	MULT result register, low word	MACL	R/W*2	Undetermined
	H'FFBC	MULT immediate multiplier register	MR	W	Undetermined
	H'FFBE	MULT immediate multiplicand register	MMR	W	Undetermined

- Notes: 1. The arithmetic registers require word-size access. Byte-size access is not supported. If byte-size access is attempted, subsequent results may be incorrect.  
 2. MULT result registers MACH and MACL cannot be modified independently. Write access to MACH must be immediately followed by write access to MACL, so that the modification takes place 32 bits at a time.

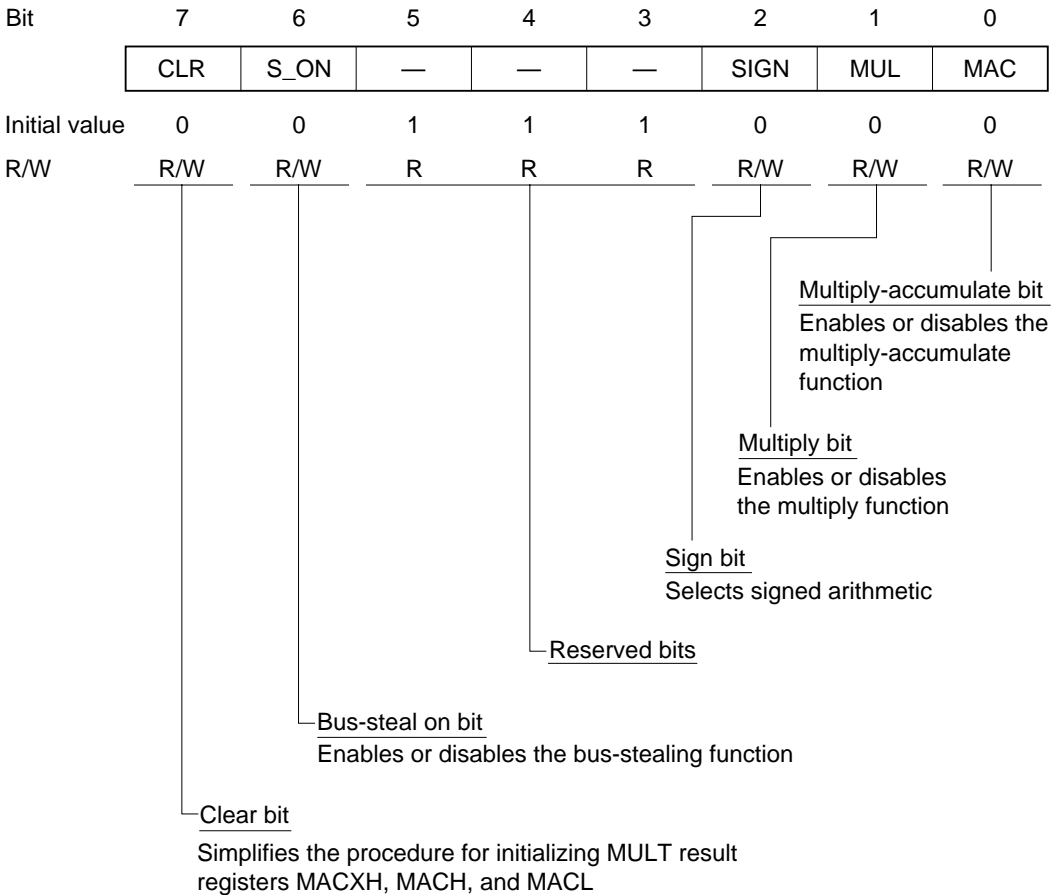
Example: MDV.W #aa:16, @MACH } These instructions must be executed  
 MDV.W #aa:16, @MACL } consecutively.

## 5.2 Register Descriptions

This section describes the H8MULT registers.

### 5.2.1 MULT Control Register

The MULT control register (MLTCR) is an eight-bit readable/writable register that clears the MULT result registers, selects the type of multiplication operation, and selects the bus-stealing function. The bit structure of MLTCR is shown next.



**(1) Bit 7—Clear (CLR):** The purpose of this bit is to simplify the procedure for initializing MULT result registers MACXH, MACH, and MACL. If the CLR bit is set to 1, when a write access is made to one of these three registers (MACXH, MACH, or MACL), regardless of the value of the write data, the other two registers are initialized to H'0000.

**(2) Bit 6—Bus-Steal On (S\_ON):** Enables or disables the bus-stealing function. If the S\_ON bit is set to 1, data can be set in the MULT registers at the same time as the CPU accesses memory. If the S\_ON bit is cleared to 0, this bus-stealing function is disabled.

For further information, see section 5.3.3 “Bus-Stealing Function.”

**(3) Bits 5 to 3—Reserved:** These bits are reserved for future expansion. They are always read as 1 and cannot be modified.

**(4) Bit 2—Sign (SIGN):** Specifies signed arithmetic. The multiplication is performed in signed mode if the SIGN bit is set to 1, and in unsigned mode if the SIGN bit is cleared to 0. When a multiply-accumulate operation is executed, the operation is performed in non-saturating mode or saturating mode. The results of saturating multiply-accumulate operations are stored in 32-bit form of MACH and MACL registers. In this case, MACXH register is not used. When an overflow occurs, set bit 0 in the MACXH register to 1. The results of non-saturating multiply-accumulate operations are stored in 42-bit form of MACXH, MACH, and MACL registers. In this case, an overflow is not detected.

For details on the SIGN bit and the operation contents, see section 5.3.4 “Multiply and Multiply-Accumulate Functions.”

**(5) Bit 1—Multiply (MUL):** Enables or disables the multiply function. The multiply function is enabled when the MUL bit is set to 1. Do not set both the MUL bit and MAC bit (bit 0) to 1 at the same time. If both bits are set to 1, neither function is enabled.

**(6) Bit 0—Multiply-Accumulate (MAC):** Enables or disables the multiply-accumulate function. The multiply-accumulate function is enabled when the MAC bit is set to 1. Do not set both the MAC bit and MUL bit (bit 1) to 1 at the same time. If both bits are set to 1, neither function is enabled.



### 5.2.2 MULT Base Address Register

The MULT base address register (MLTBR) is a readable/writable register that specifies the upper eight bits of the memory address of the multiplier or multiplicand in multiply or multiply-accumulate operations when the bus-stealing function is enabled.

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

### 5.2.3 MULT Multiplier Address Register

The MULT multiplier address register (MLTAR) is a readable/writable register that specifies the lower eight bits of the memory address of the multiplier in multiply or multiply-accumulate operations when the bus-stealing function is enabled.

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

### 5.2.4 MULT Multiplicand Address Register

The MULT multiplicand address register (MLTMAR) is a readable/writable register that specifies the lower eight bits of the memory address of the multiplicand in multiply or multiply-accumulate operations when the bus-stealing function is enabled.

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

### 5.2.5 MULT Multiplier Register A

MULT multiplier register A (MCA) is a 16-bit readable/writable register that stores a multiplier for use in multiply or multiply-accumulate operations.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: MCA requires word-size access.

### 5.2.6 MULT Multiplier Register B

MULT multiplier register B (MCB) is a 16-bit readable/writable register that stores a multiplier for use in multiply or multiply-accumulate operations.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: MCB requires word-size access.

### 5.2.7 MULT Multiplier Register C

MULT multiplier register C (MCC) is a 16-bit readable/writable register that stores a multiplier for use in multiply or multiply-accumulate operations.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: MCC requires word-size access.

### 5.2.8 MULT Immediate Multiplier Register

The MULT immediate multiplier register (MR) is a 16-bit write-only register into which a multiplier value can be loaded for use in multiply or multiply-accumulate operations.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

MR is a write-only register. When read, it always returns H'FFFF.

Note: MR requires word-size access.

### 5.2.9 MULT Immediate Multiplicand Register

The MULT immediate multiplicand register (MMR) is a 16-bit write-only register into which a multiplicand value can be loaded for use in multiply or multiply-accumulate operations.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
R/W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

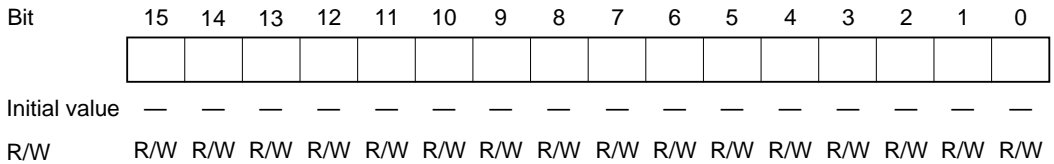
MMR is a write-only register. When read, it always returns H'FFFF.

Note: MMR requires word-size access.

### 5.2.10 MULT Result Register, Extended High Word

The MULT result register (MACXH) is a 16-bit readable/writable register that stores the upper 10 bits of the 42-bit result of a non-saturating multiply-accumulate operation. The sign-extended value of bit 9 is set in bits 15 to 10 of MACXH.

MACXH is not used in multiply operations, and bits 15 to 1 are not used in saturating multiply-accumulate operations.

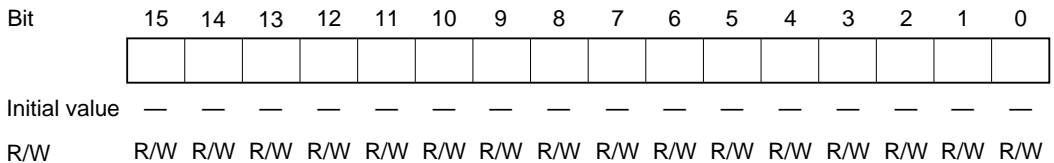


Bit 0 of the MACXH register is an overflow flag (OVF) that is set to 1 when the result of a saturating multiply-accumulate operation overflows.

Note: MACXH requires word-size access.

### 5.2.11 MULT Result Register, High Word

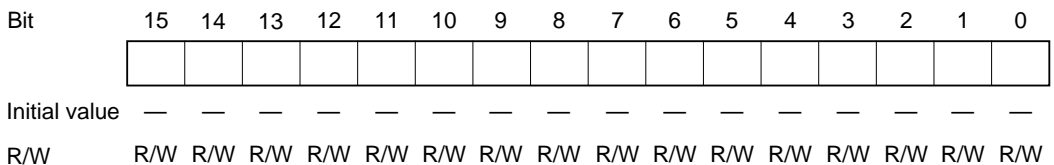
The MULT result register, high word (MACH) is a 16-bit readable/writable register that stores bits 31 to 16 of a non saturating multiply-accumulate operation.



Note: MACH requires word-size access.

### 5.2.12 MULT Result Register, Low Word

The MULT result register, low word (MACL) is a 16-bit readable/writable register that stores bits 15 to 0 of a non saturating multiply-accumulate operation.



Note: MACL requires word-size access.

## 5.3 Operation

The operation of the H8/539F's on-chip multiplier module will be described in the following order: initialization of MULT result registers; register write; bus-stealing function; then multiply and multiply-accumulate operations.

### 5.3.1 Initialization of MULT Result Registers

MULT result registers MACXH, MACH, and MACL are not initialized by a reset. In a multiply-accumulate operation, in which the multiplication result is added to the value in the MULT result registers, the MULT result registers must be initialized before use, either by clearing them or by writing the necessary values in them ahead of time. Initialization is not necessary when these registers are only used for multiplication.

Initialization should be performed by one of the following methods.

**(1) Individual Register Initialization:** The registers can be initialized by writing to them individually. The MACH and MACL registers must be written to consecutively.

Example:

```
MOV.W #H'0000, @MACXH
MOV.W #H'0000, @MACH ; Do not change the order of these two instructions
MOV.W #H'0000, @MACL
```

or

```
CLR.W @MACXH
CLR.W @MACH ; Do not change the order of these two instructions
CLR.W @MACL
```

**(2) One-Step Initialization:** All three registers can be initialized at once. MACXH, MACH, and MACL are all initialized to #H'0000, regardless of the write data.

Example:

```
BSET.B #7, @MLTCR ; Set CLR bit in MLTCR
MOV.W #aa:16, @MACXH ; Destination can be @MACH or @MACL instead
(BCLR.B #7, @MLTCR)
```

The one-step initialization function operates at a write access to MACXH, MACH, or MACL. It does not operate at a read access, or a write access to any other register, so the CLR bit does not necessarily have to be cleared to 0 after one-step initialization.

### 5.3.2 Writing to MULT Multiplier Registers

The MULT multiplier registers (MCA, MCB, MCC) can be loaded by writing to them directly, or by bus stealing. The bus-stealing function and direct writing are performed independently for MCA, MCB, and MCC, so both types of loading can be used together.

**(1) Direct Writing:** This method writes to MCA, MCB, or MCC by direct addressing. Specify the address of MCA, MCB, or MCC as the destination operand in a write instruction. Be sure to use a word-size instruction.

Example:

```
MOV.W #aa:16, @MCA      ; Write 16-bit data #aa in MCA
```

**(2) Loading Data by Bus Stealing:** When the CPU accesses its memory address space, the data on the data bus can be loaded automatically into a MULT multiplier register (bus stealing). Bus stealing is performed only for particular addresses, which are specified in the MULT multiplier address register (MLTAR) and MULT base address register (MLTBR).

For further information, see section 5.3.3 “Bus-Stealing Function.”

Example:

```
BSET.B #6, @MLTCR      ;  
MOV.B #H'FE, @MLTBR    ; Set up bus-stealing function  
MOV.B #H'80, @MLTAR    ;  
  
MOV.W #aa:16, @FE80    ; Write data #aa:16 to @FE80 and load same data into MCA  
.  
.  
.  
MOV.W @FE80, R0        ; Read data from @FE80 and load same data into MCA  
.  
.  
.  
                        ; TST.W @FE80 instruction would do the same
```

### 5.3.3 Bus-Stealing Function

The bus-stealing function loads the value on the data bus into the H8MULT module when the CPU accesses its memory address space. The bus-stealing function can be used to multiply or multiply-and-accumulate two values stored in memory.

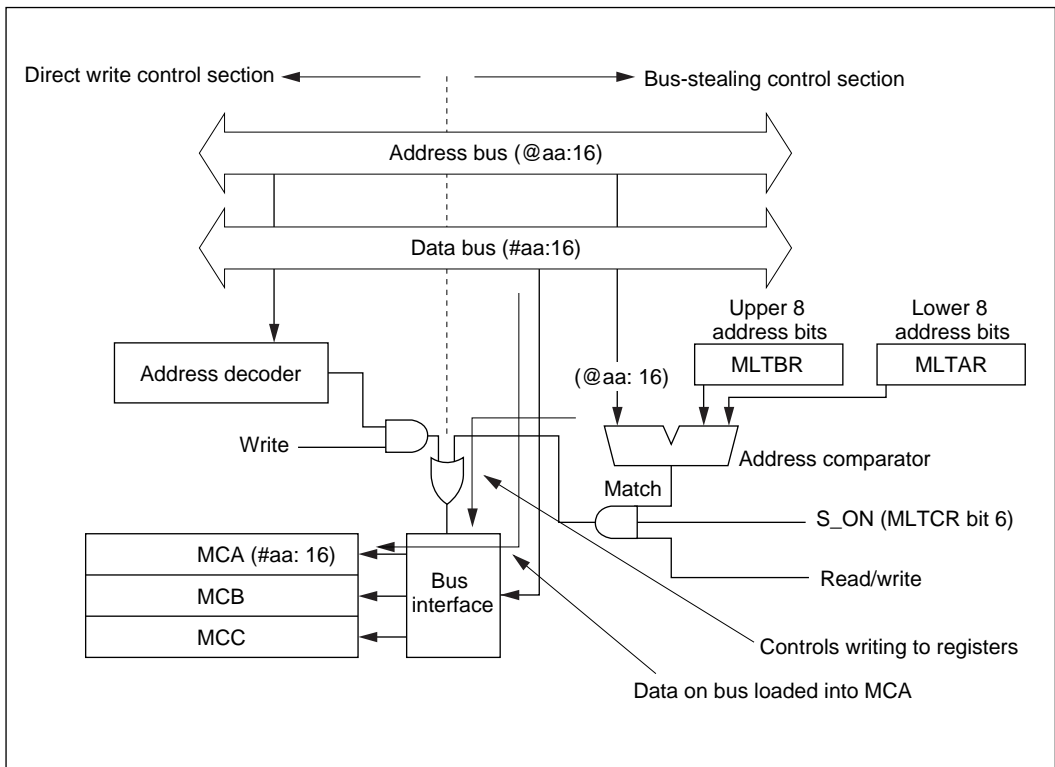
The bus-stealing function can be enabled or disabled by bit 6 (S\_ON) in the MULT control register (MLTCR).

**(1) Loading of Multiplier by Bus Stealing:** Figure 5-2 shows the loading of data into register MCA by bus stealing. If the S\_ON bit is set to 1, the H8MULT module monitors the address bus when the CPU accesses its memory address space, and compares the address on the bus with the MULT base address register (MLTBR) and MULT multiplier address register (MLTAR).

If MLTBR (upper 8 bits) and MLTAR (lower 8 bits) = @aa:16, the data on the data bus is loaded into MULT multiplier register A (MCA).

If MLTBR (upper 8 bits) and MLTAR (lower 8 bits) + 2 = @aa:16, the data on the data bus is loaded into MULT multiplier register B (MCB).

If MLTBR (upper 8 bits) and MLTAR (lower 8 bits) + 4 = @aa:16, the data on the data bus is loaded into MULT multiplier register C (MCC).



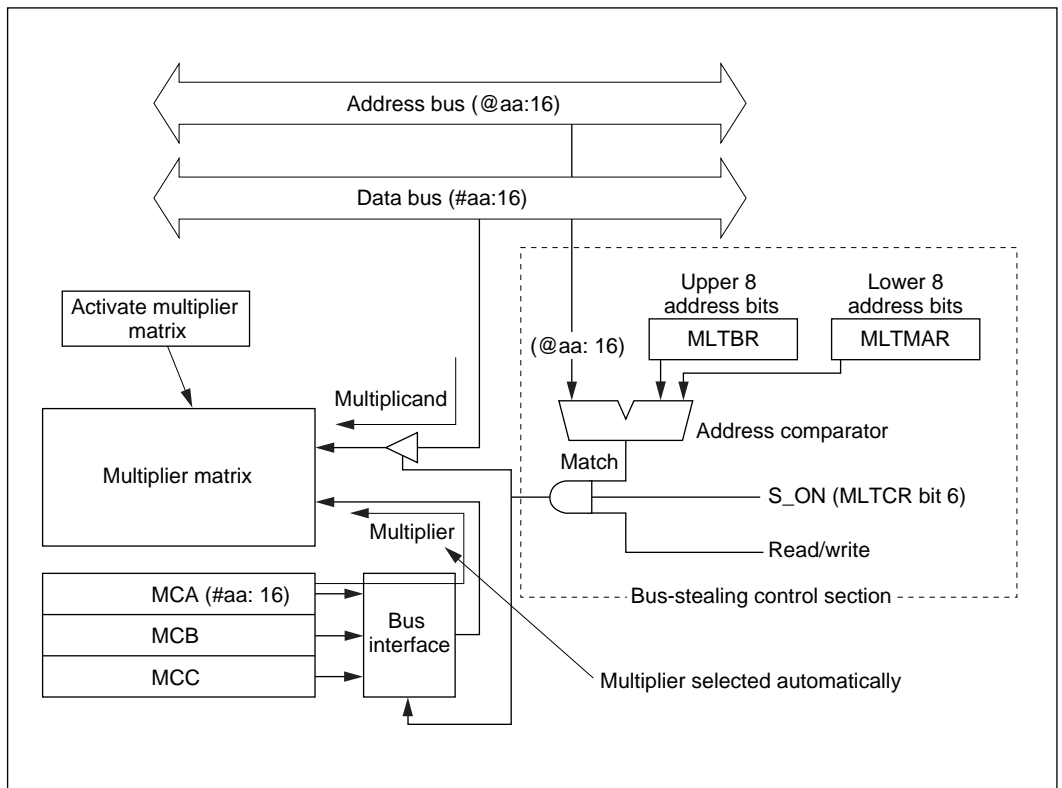
**Figure 5-2 Loading of Data into Register MCA by Bus Stealing**

**(2) Loading of Multiplicand and Activation of Multiplier by Bus Stealing:** Figure 5-3 shows the loading of the multiplicand and automatic selection of the multiplier register by bus stealing. If the S\_ON bit is set to 1, the H8MULT module monitors the address bus when the CPU accesses its memory address space, and compares the address on the bus with the MULT base address register (MLTBR) and MULT multiplicand address register (MLTMAR).

If MLTBR (upper 8 bits) and MLTMAR (lower 8 bits) = @aa:16, the data on the data bus is loaded as the multiplicand, the multiplier is fetched from MULT multiplier register A (MCA), and these values are multiplied, or multiplied and accumulated.

If MLTBR (upper 8 bits) and MLTMAR (lower 8 bits) + 2 = @aa:16, the multiplier is fetched from MULT multiplier register B (MCB).

If MLTBR (upper 8 bits) and MLTMAR (lower 8 bits) + 4 = @aa:16, the multiplier is fetched from MULT multiplier register C (MCC).



**Figure 5-3 Loading of Multiplicand and Activation of Multiplier by Bus Stealing**



### 5.3.4 Multiply and Multiply-Accumulate Functions

The H8MULT module can execute  $16 \times 16$ -bit multiplication, and accumulate products up to a data length of 42 bits. The multiplier and multiplicand on which arithmetic is carried out can be specified in two ways. They can be loaded directly into the H8MULT module, or data in memory can be loaded into the H8MULT module by the bus-stealing function. Multiply and multiply-accumulate operations are described below.

**(1) Multiply: Direct Loading of Multiplier and Multiplicand:** The procedure is given next.

(a) Select the multiply function.

- Unsigned multiplication: Set bits 2 to 0 (SIGN, MUL, MAC) in the MULT control register (MLTCR) to 010.
- Signed multiplication: Set MLTCR bits 2 to 0 (SIGN, MUL, MAC) to 110.

Three results are stored in 42-bit form of MACXH, MACH and MACL registers. In this case, an overflow is not detected.

(b) Set the multiplier and multiplicand.

Load the multiplier into the MULT immediate multiplier register (MR), then load the multiplicand into the MULT immediate multiplicand register (MMR). The multiplier matrix is activated automatically when the multiplicand is loaded.

Be sure to use word-size data transfer instructions to load the multiplier and multiplicand. The instruction that loads MMR must be executed immediately after the instruction that loads MR. A coding example is given next.

Example: signed multiplication, #AAAA  $\times$  #BBBB

```
MOV.B #06, @MLTCR    ; SIGN = 1, MUL = 1
MOV.W #AAAA, @MR     ; Load multiplier
MOV.W #BBBB, @MMR    ; Load multiplicand and start multiplying
```

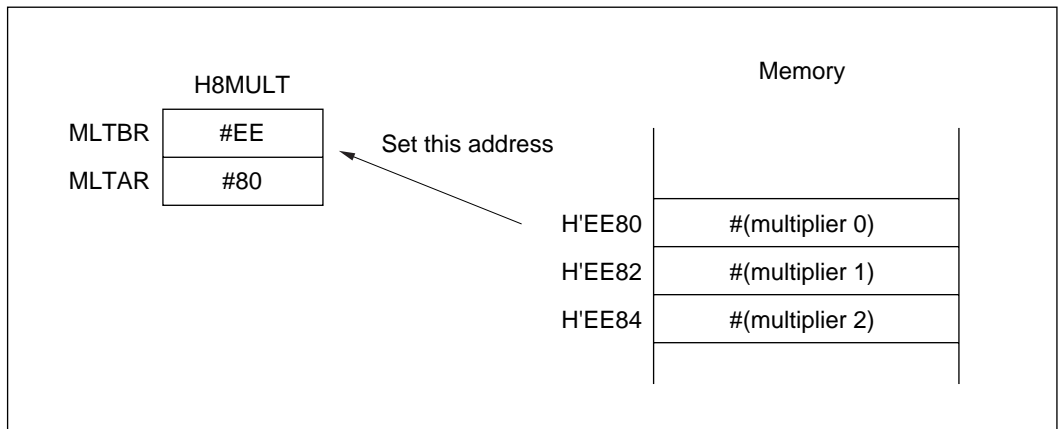
**(2) Multiply: Multiplier Loaded by Bus Stealing, Multiplicand Loaded Directly:** The procedure is given next.

(a) Select the multiply function.

See under (1).

(b) Select the bus-stealing function.

Set bit 6 (S\_ON) to 1 in the MULT control register (MLTCR), and specify the address at which the multiplier will be located in the MULT base address register (MLTBR) and MULT multiplier address register (MLTAR). The multiplier can be located in any of three words starting at the specified address. Place the upper eight bits of the address in MLTBR and the lower eight bits in MLTAR. See the example in figure 5-4.



**Figure 5-4 MLTBR and MLTAR Settings**

(c) Set the multiplier and multiplicand

The multiplicand must be set immediately after the multiplier. First access the multiplier in memory, then load the multiplicand into MMR. The multiplier matrix is activated automatically when the multiplicand is loaded.

Be sure to use word-size instructions to access the multiplier in memory and load the multiplicand. These instructions must be executed consecutively.

A coding example is given next. Figure 5-5 shows the data flow.

Example: Unsigned multiplication, multiplier  $\times$  #BBBB, multiplier loaded from @EE80 on memory by bus stealing

```
MOV.B #42, @MLTCR    ; S_ON = 1, MUL = 1
MOV.B #EE, @MLTBR
MOV.B #80, @MLTAR    ; Multiplier address = #EE80

MOV.W @EE80, R0      ; Access multiplier address
                       ; Bus-stealing function loads multiplier into MCA
MOV.W #BBBB, @MMR    ; Load multiplicand to start multiplying multiplier  $\times$  #BBBB
```

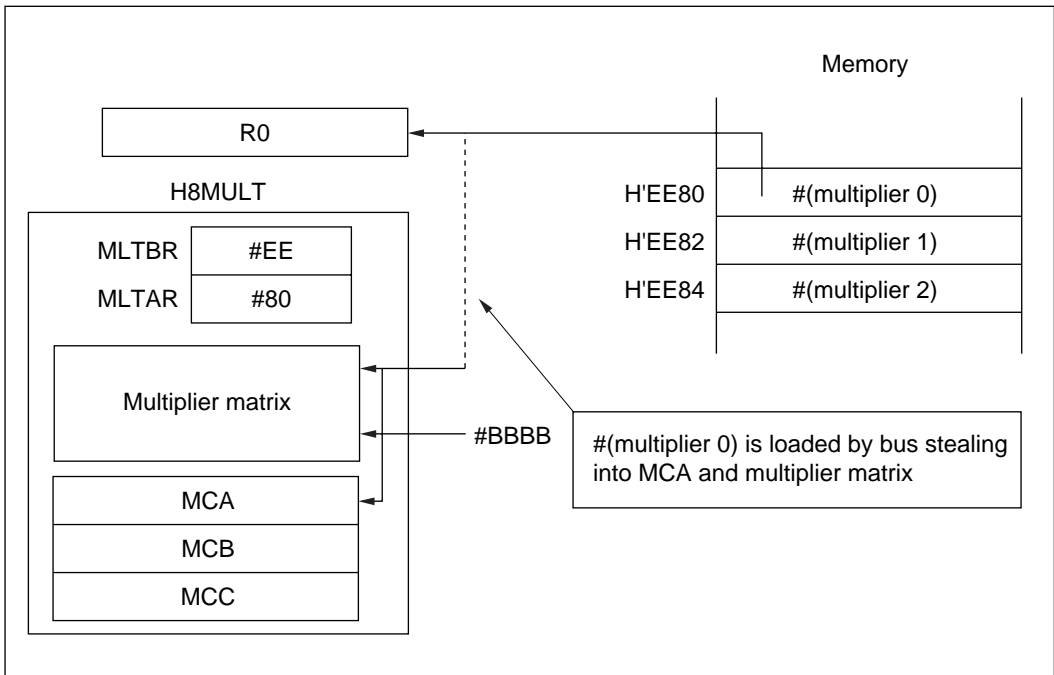


Figure 5-5 Multiplication Data Flow

### (3) Multiply: Multiplier and Multiplicand Loaded by Bus Stealing

(a) Select the multiply function.

See under (1).

(b) Select the bus-stealing function.

See under (2) (b).

To load the multiplicand by bus stealing, in addition to the steps in (2) (b), set the lower eight bits of the address where the multiplicand will be located in the MULT multiplicand address register (MLTMAR). See the example in figure 5-6.

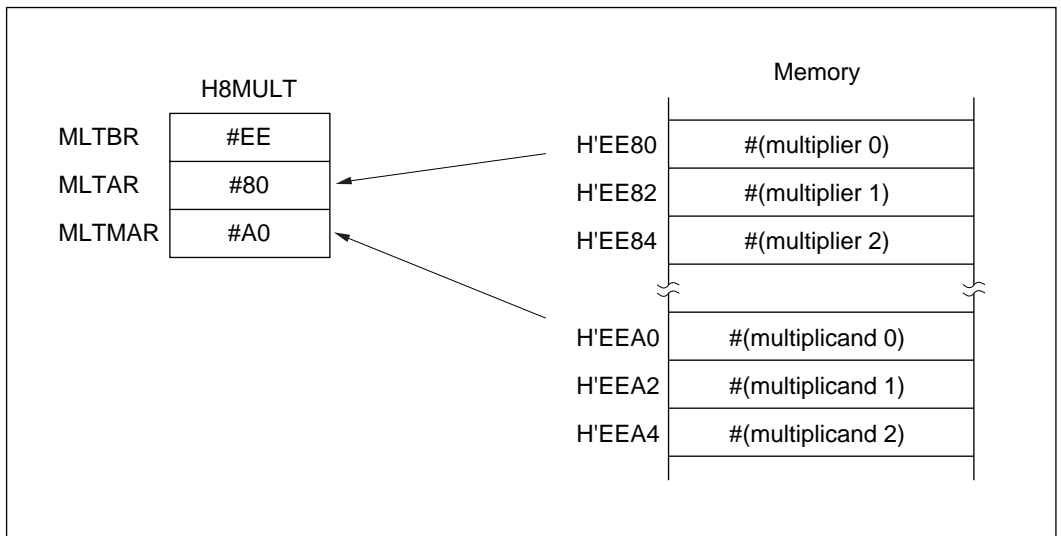


Figure 5-6 MLTBR, MLTAR, and MLTMAR Settings

(c) Set the multiplier and multiplicand

Access the multiplier, then the multiplicand. The two accesses do not have to be consecutive. When the multiplier is accessed in memory, it is temporarily stored in one of the MULT multiplier registers (MCA, MCB, or MCC) by the bus-stealing function. After that, when the multiplicand is accessed, the multiplier is fetched from MCA, MCB, or MCC, the multiplier and multiplicand are both loaded into the H8MULT module, and multiplication begins.

The register from which the multiplier is fetched is determined by the multiplicand address. For details see section 5.3.3 “Bus-Stealing Function.” A coding example is given next. Figure 5-7 shows the data flow.

Example: Unsigned multiplication, multiplier (@EE80) × multiplicand (@EEA0), loaded from memory by bus stealing

```
MOV.B #42, @MLTCR      ; S_ON = 1, MUL = 1
MOV.B #EE, @MLTBR
MOV.B #80, @MLTAR      ; Multiplier address = #EE80
MOV.B #A0, @MLTMAR     ; Multiplicand address = #EEA0

MOV.W @EE80, R0        ; Access multiplier address
                       ; Bus-stealing function loads multiplier into MCA

MOV.W @EEA0, R0        ; Access multiplicand address; H8MULT loads multiplicand
                       ; by bus-stealing function, gets multiplier from MCA,
                       ; loads multiplier into multiplier matrix, and starts
                       ; multiplying
```

**(4) Multiply and Accumulate: Direct Loading of Multiplier and Multiplicand:** The procedure is given next.

- (a) Select the multiply-accumulate function.
  - Saturating accumulation: Set bits 2 to 0 (SIGN, MUL, MAC) in the MULT control register (MLTCR) to 001. The results are stored in 32-bit form in the MACH and MACL registers. When an overflow occurs, bit 0 in the MACXH register is set to 1.
  - Non-saturating accumulation: Set bits 2 to 0 (SIGN, MUL, MAC) in the MULT control register (MLTCR) to 101. The results are stored in 42-bit form in the MACXH, MACH, and MACL registers. In this case, an overflow is not detected.

(b) Set a constant and specify the multiplier and multiplicand.

First set a constant in the MULT result registers (MACXH, MACH, MACL), or clear these registers. Next load the multiplier into the MULT immediate multiplier register (MR), then load the multiplicand into the MULT immediate multiplicand register (MMR). The multiplier matrix is activated automatically when the multiplicand is loaded.

The operation performed is  $(\text{multiplier}) \times (\text{multiplicand}) + (\text{constant})$ .

Be sure to use word-size data transfer instructions to load the multiplier and multiplicand. The instruction that loads MMR must be executed immediately after the instruction that loads MR. A coding example is given next.

Example: Non-saturating multiply-accumulate, #AAAA  $\times$  #BBBB + #CCCC

```
BSET.B #7, @MLTCR      ; CLR = 1
CLR.W @MACXH           ; Initialize MACXH, MACH, and MACL
BCLR.B #7, @MLTCR      ; CLR = 0

MOV.W #0000, @MACH
MOV.W #CCCC, @MACL     ; Set 32-bit constant
MOV.B #05, @MLTCR     ; SIGN = 1, MUL = 1
MOV.W #AAAA, @MR       ; Load multiplier
MOV.W #BBBB, @MMR      ; Load multiplicand and start multiplying
```

**(5) Multiply and Accumulate: Multiplier Loaded by Bus Stealing, Multiplicand Loaded Directly:** The procedure is given next.

- (a) Select the multiply-accumulate function.

See under (4).

- (b) Select the bus-stealing function.

Set bit 6 (S\_ON) to 1 in the MULT control register (MLTCR), and specify the address at which the multiplier will be located in the MULT base address register (MLTBR) and MULT multiplier address register (MLTAR). The multiplier can be located in any of three words starting at the specified address. Place the upper eight bits of the address in MLTBR and the lower eight bits in MLTAR.

- (c) Set the multiplier and multiplicand

The multiplicand must be set immediately after the multiplier. First access the multiplier in memory, then load the multiplicand into MMR. The multiplier matrix is activated automatically when the multiplicand is loaded.

Be sure to use word-size instructions to access the multiplier in memory and load the multiplicand. These instructions must be executed consecutively.

A coding example is given next.

Example: Saturating accumulation, multiplier  $\times$  #BBBB + #CCCC, multiplier loaded from @EE80 on memory by bus stealing

```
BSET.B #7, @MLTCR      ; CLR = 1
CLR.W @MACXH
BCLR.B #7, @MLTCR      ; CLR = 0

MOV.W #0000, @MACH
MOV.W #CCCC, @MACL
MOV.B #41, @MLTCR      ; S_ON = 1, MAC = 1
MOV.B #EE, @MLTBR
MOV.B #80, @MLTAR      ; Multiplier address = #EE80

MOV.W @EE80, R0        ; Access multiplier address
                        ; Bus-stealing function loads multiplier into MCA
MOV.W #BBBB, @MMR      ; Load multiplicand to start multiply-accumulate operation
                        ; multiplier  $\times$  #BBBB + #CCCC.
```

## (6) Multiply and Accumulate: Multiplier and Multiplicand Loaded by Bus Stealing

- (a) Select the multiply-accumulate function.

See under (4).

- (b) Select the bus-stealing function.

See under (5) (b).

To load the multiplicand by bus stealing, in addition to the steps in (5) (b), set the lower eight bits of the address where the multiplicand will be located in the MULT multiplicand address register (MLTMAR).

- (c) Set the multiplier and multiplicand

Access the multiplier, then the multiplicand. The two accesses do not have to be consecutive. When the multiplier is accessed in memory, it is temporarily stored in one of the MULT multiplier registers (MCA, MCB, or MCC) by the bus-stealing function. After that, when the multiplicand is accessed, the multiplier is fetched from MCA, MCB, or MCC, the multiplier and multiplicand are both loaded into the H8MULT module, and multiplication begins.

The register from which the multiplier is fetched is determined by the multiplicand address. For details see section 5.3.3 “Bus-stealing function.” A coding example is given next.

Example: Saturating multiplication and accumulation, bus stealing,  
multiplier (@EE80) × multiplicand (@EEA0) + #CCCC

```
BSET.B #7, @MLTCR      ; CLR = 1
```

```
CLR.W @MACXH
```

```
BCLR.B #7, @MLTCR     ; CLR = 0
```

```
MOV.W #0000, @MACH
```

```
MOV.W #CCCC, @MACL
```

```
MOV.B #41, @MLTCR     ; S_ON = 1, MAC = 1
```

```
MOV.B #EE, @MLTBR
```

```
MOV.B #80, @MLTAR     ; Multiplier address = #EE80
```

```
MOV.B #A0, @MLTAR     ; Multiplicand address = #EEA0
```

```
MOV.W @EE80, R0       ; Access multiplier address  
                       ; Bus-stealing function loads multiplier into MCA
```

```
MOV.W @EEA0, R0       ; Access multiplicand address; H8MULT loads multiplicand  
                       ; by bus-stealing function, fetches multiplier from MCA,  
                       ; loads multiplier into multiplier matrix, and starts multiplying
```



# Section 6 Interrupt Controller

## 6.1 Overview

The interrupt controller decides when to start interrupt exception handling and when to start the data transfer controller (DTC), and arbitrates between competing interrupts. This section describes the interrupts and the functions, features, internal structure, and registers of the interrupt controller.

For details of data transfers performed by the DTC, see section 7, “Data Transfer Controller.”

### 6.1.1 Features

The features of the interrupt controller are:

- Six interrupt priority registers (IPR)

Priority levels from 7 to 0 can be assigned to IRQ0, IRQ1 to IRQ3, and each of the on-chip supporting modules, covering all interrupts except NMI.

- Default priority order for simultaneous interrupts on the same level

Lower-priority interrupts remain pending until higher-priority interrupts have been handled. NMI has the highest priority level (8) and cannot be masked.\*

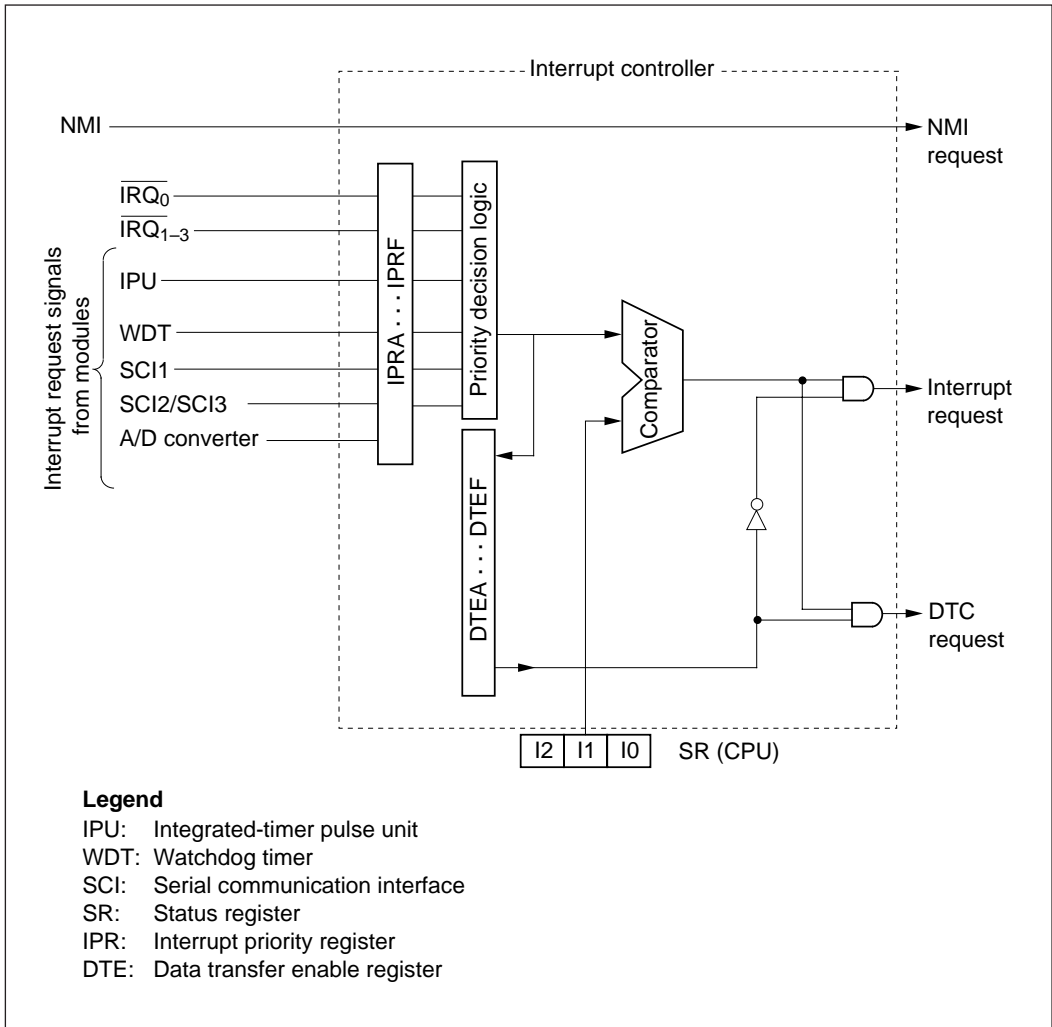
- Six data transfer enable (DTE) registers

Software can select which interrupts (other than NMI) to have served by the DTC.

Note: \* The exception is when programming or erasing flash memory, in which case NMI input is disabled. See section 19.4.9, “NMI Input Masking” and section 20.4.8, “NMI Input Masking” for details.

## 6.1.2 Block Diagram

Figure 6-1 shows a block diagram of the interrupt controller.



**Figure 6-1 Block Diagram of Interrupt Controller**

### 6.1.3 Register Configuration

The interrupt controller has six interrupt priority registers (IPRA to IPRF) and six data transfer enable registers (DTEA to DTEF). See section 7.2.5, “Data Transfer Enable Registers A to F” for details of DTEA to DTEF.

Table 6-1 summarizes these registers.

**Table 6-1 Interrupt Controller Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FF00	Interrupt priority register A	IPRA	R/W	H'00
H'FF01	Interrupt priority register B	IPRB	R/W	H'00
H'FF02	Interrupt priority register C	IPRC	R/W	H'00
H'FF03	Interrupt priority register D	IPRD	R/W	H'00
H'FF04	Interrupt priority register E	IPRE	R/W	H'00
H'FF05	Interrupt priority register F	IPRF	R/W	H'00
H'FF08	Data transfer enable register A	DTEA	R/W	H'00
H'FF09	Data transfer enable register B	DTEB	R/W	H'00
H'FF0A	Data transfer enable register C	DTEC	R/W	H'00
H'FF0B	Data transfer enable register D	DTED	R/W	H'00
H'FF0C	Data transfer enable register E	DTEE	R/W	H'00
H'FF0D	Data transfer enable register F	DTEF	R/W	H'00

Table 6-2 summarizes the NMI control register (NMICR), IRQ control register (IRQCR), and IRQ flag register (IRQFR).

**Table 6-2 Interrupt Controller Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FF1C	NMI control register	NMICR	R/W	H'FE
H'FF1D	IRQ control register	IRQCR	R/W	H'F0
H'FEDE	IRQ flag register	IRQFR	R/W	H'F1

## 6.2 Interrupt Sources

There are two types of interrupts: external interrupts (NMI, IRQ0, and IRQ1 to IRQ3.), and internal interrupts (39 sources). Table 6-3 indicates the default priority order and vector addresses of these interrupts.

When multiple interrupts occur simultaneously, the interrupt with the highest priority is served first. Using IPRA to IPRF, software can assign priorities to interrupts on a module basis. Relative priorities within the same module are fixed. If the same priority is assigned to two or more modules, simultaneous interrupt requests from those modules are served in the priority order shown in table 6-3.

After a reset, all interrupts except NMI are assigned priority 0 and are disabled.

**Table 6-3 Interrupt Priorities and Vector Addresses**

Interrupt Source	Assignable Priority Levels (initial value)	Corresponding IPR Bits	Priority within Module	Vector Table Entry Address		Priority among Interrupts on Same Level														
				Minimum Mode	Maximum Mode															
NMI	8 (8)	—	—	H'0016–0017	H'002C–002F															
IRQ0 Interval timer	7–0 (0)	IPRA upper	2 1	H'0040–0041 H'0042–0043	H'0080–0083 H'0084–0087															
A/D converter		4 bits	0	H'0044–0045	H'0088–008B															
IRQ1	7–0 (0)	IPRA lower	2 1	H'0048–0049 H'004A–004B	H'0090–0093 H'0094–0097															
IRQ2		4 bits	0	H'004C–004D	H'0098–009B															
IPU channel 1	IMI1 IMI2 CMI1/CMI2 OVI	7–0 (0)	IPRB upper 4 bits 0	3 2 1 0	H'0050–0051 H'0052–0053 H'0054–0055 H'0056–0057		H'00A0–00A3 H'00A4–00A7 H'00A8–00AB H'00AC–00AF													
								IMI3 IMI4 CMI3/CMI4	7–0 (0)	IPRB lower 4 bits 0	2 1 0	H'0058–0059 H'005A–005B H'005C–005D	H'00B0–00B3 H'00B4–00B7 H'00B8–00BB							
														IPU channel 2	IMI1 IMI2 CMI1/CMI2 OVI	7–0 (0)	IPRC upper 4 bits 0	3 2 1 0	H'0060–0061 H'0062–0063 H'0064–0065 H'0066–0067	H'00C0–00C3 H'00C4–00C7 H'00C8–00CB H'00CC–00CF
IPU channel 4	IMI1 IMI2 CMI1/CMI2 OVI	7–0 (0)	IPRD upper 4 bits 0	3 2 1 0	H'0070–0071 H'0072–0073 H'0074–0075 H'0076–0077		H'00E0–00E3 H'00E4–00E7 H'00E8–00EB H'00EC–00EF													
								IPU channel 5	IMI1 IMI2 CMI1/CMI2 OVI	7–0 (0)	IPRD lower 4 bits 0	3 2 1 0	H'0078–0079 H'007A–007B H'007C–007D H'007E–007F	H'00F0–00F3 H'00F4–00F7 H'00F8–00FB H'00FC–00FF						
IPU channel 6	IMI1 IMI2 OVI	7–0 (0)	IPRE upper 4 bits 0	2 1 0	H'0080–0081 H'0082–0083 H'0086–0087		H'0100–0103 H'0104–0107 H'010C–010F													
								IPU channel 7	IMI1 IMI2 OVI	7–0 (0)	IPRE lower 4 bits 0	2 1 0	H'0088–0089 H'008A–008B H'008E–008F	H'0110–0113 H'0114–0117 H'011C–011F						
SCI1	ERI1 RI1 TI1 TE11	7–0 (0)	IPRF upper 4 bits 0	3 2 1 0	H'0090–0091 H'0092–0093 H'0094–0095 H'0096–0097		H'0120–0123 H'0124–0127 H'0128–012B H'012C–012F													
								SCI2/ SCI3	ERI2/ERI3 RI2/RI3 TI2/TI3 TE12/TE13	7–0 (0)	IPRF lower 4 bits 0	3 2 1 0	H'0098–0099 H'009A–009B H'009C–009D H'009E–009F	H'0130–0133 H'0134–0137 H'0138–013B H'013C–013F						

The five external interrupts are NMI and IRQ0 to IRQ3.

Each external interrupt is described below.

### 6.2.1 NMI Interrupt

NMI has the highest interrupt priority level (8) and cannot be masked\*. Input at the NMI pin is edge-sensed. Either the rising edge or falling edge can be selected by setting or clearing the nonmaskable interrupt edge bit (NMIEG) in the NMI control register (NMICR).

In NMI exception handling the T bit in the status register (SR) is cleared to 0 and I2 to I0 are all set to 1, thereby setting the interrupt mask level to 7.

Note: \* The exception is when programming or erasing flash memory, in which case NMI input is disabled. See section 19.4.9, “NMI Input Masking” and section 20.4.8, “NMI Input Masking” for details.

**NMI Control Register (Address H'FF1C):** The NMI control register (NMICR) selects the sensitive edge of the NMI input. NMICR is initialized to H'FE by a reset and in hardware standby mode. It is not initialized in software standby mode. The NMICR bit structure is shown next.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	NMIEG
Initial value	1	1	1	1	1	1	1	0
R/W	—	—	—	—	—	—	—	R/W

Reserved bits

Nonmaskable interrupt edge  
 Selects sensitive edge of NMI input

(1) **Bits 7 to 1—Reserved:** Read-only bits, always read as 1.

(2) **Bit 0—Nonmaskable Interrupt Edge (NMIEG):** Selects the sensitive edge of the NMI input.

#### Bit 0

NMIEG	Description
0	NMI is requested on falling edge of NMI input (Initial value)
1	NMI is requested on rising edge of NMI input

## 6.2.2 IRQ0 Interrupt

An IRQ0 interrupt can be requested by an interrupt signal from the  $\overline{\text{IRQ}}_0$  pin or an interrupt signal from the watchdog timer (WDT). These two interrupt sources have different vectors.

The interrupt from the  $\overline{\text{IRQ}}_0$  pin is level-sensed. A  $\overline{\text{IRQ}}_0$  input requests an IRQ0 interrupt if the interrupt request enable 0 bit (IRQ0E) in the IRQ control register (IRQCR) is set to 1. A WDT overflow requests an IRQ0 interrupt when the WDT is set to interval timer mode. The WDT then requests an IRQ0 interrupt each time the timer counter (TCNT) overflows.

A priority level from 7 to 0 can be assigned to IRQ0 in the upper four bits of IPRA. If bit 4 in DTEA is set to 1, IRQ0 is served by the DTC.

In IRQ0 exception handling the T bit in SR is cleared to 0 and the interrupt mask level is set to the value selected in the four upper bits of IPRA.

## 6.2.3 IRQ1 to IRQ3 Interrupt

Interrupts IRQ1 to IRQ3 are requested by interrupt signals from the  $\overline{\text{IRQ}}_1$  to  $\overline{\text{IRQ}}_3$  pins. The  $\overline{\text{IRQ}}_1$  to  $\overline{\text{IRQ}}_3$  inputs are sensed on the falling edge. The falling edge generates an  $\overline{\text{IRQ}}_1$  to  $\overline{\text{IRQ}}_3$  interrupt request if the interrupt request enable 1, 2, or 3 bit (IRQ1E, IRQ2E, or IRQ3E) in the IRQ control register (IRQCR) is set to 1.

A priority level from 7 to 0 can be assigned to IRQ1, IRQ2, and IRQ3 collectively in the lower four bits of IPRA. If bits 2 to 0 in DTEA are set, these interrupts are served by the DTC.

In IRQ1, IRQ2, and IRQ3 exception handling the T bit in SR is cleared to 0 and the interrupt mask level is set to the value selected in the lower four bits of IPRA.

**IRQ Control Register (Address H'FF1D):** The IRQ control register (IRQCR) enables and disables inputs at  $\overline{\text{IRQ}}_1$  to  $\overline{\text{IRQ}}_3$ , and  $\overline{\text{IRQ}}_0$ . IRQCR is initialized to H'F0 by a reset and in hardware standby mode. It is not initialized in software standby mode. The bit structure of IRQCR is shown next.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	IRQ3E	IRQ2E	IRQ1E	IRQ0E
Initial value	1	1	1	1	0	0	0	0
R/W	—	—	—	—	R/W	R/W	R/W	R/W

Reserved bits
Interrupt request enable bits  
 These bits select functions of ports 6 and 7

(1) **Bits 7 to 4—Reserved:** Read-only bits, always read as 1.

(2) **Bit 3—Interrupt Request 3 Enable (IRQ3E):** Selects the function of pin P6<sub>1</sub>.

**Bit 3**

IRQ3E	Description
0	P6 <sub>1</sub> is used for general-purpose input and output (Initial value)
1	P6 <sub>1</sub> is used for $\overline{\text{IRQ}}_3$ input

(3) **Bit 2—Interrupt Request 2 Enable (IRQ2E):** Selects the function of pin P6<sub>0</sub>.

**Bit 2**

IRQ2E	Description
0	P6 <sub>0</sub> is used for general-purpose input and output (Initial value)
1	P6 <sub>0</sub> is used for $\overline{\text{IRQ}}_2$ input

(4) **Bit 1—Interrupt Request 1 Enable (IRQ1E):** Selects the function of pin P7<sub>1</sub>.

**Bit 1**

IRQ1E	Description
0	P7 <sub>1</sub> is used for general-purpose input and output (Initial value)
1	P7 <sub>1</sub> is used for $\overline{\text{IRQ}}_1$ input

(5) **Bit 0—Interrupt Request 0 Enable (IRQ0E):** Selects the function of pin P7<sub>0</sub>.

**Bit 0**

IRQ0E	Description
0	P7 <sub>0</sub> is used for general-purpose input and output (Initial value)
1	P7 <sub>0</sub> is used for $\overline{\text{IRQ}}_0$ input



**IRQ Flag Register (Address H'FEDE):** The IRQ flag register (IRQFR) indicates the presence of IRQ1 to IRQ3 interrupt requests. When an IRQ1 to IRQ3 interrupt is requested by external input, the H8/500 CPU sets the interrupt request 1, 2, or 3 flag (IRQ1F, IRQ2F, or IRQ3F) to 1. The interrupt request can be cleared by reading this flag after it has been set to 1, then writing 0. The H8/500 CPU clears IRQ1F, IRQ2F, or IRQ3F to 0 when it outputs the interrupt vector.

IRQFR is initialized to H'F1 by a reset and in hardware standby mode. It is not initialized in software standby mode. The bit structure of IRQFR is shown next.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	IRQ3F	IRQ2F	IRQ1F	—
Initial value	1	1	1	1	0	0	0	1
R/W	—	—	—	—	R/W*	R/W*	R/W*	—

Reserved bit

Interrupt request flags  
These bits indicate interrupt request input

Reserved bits

Note: \* Software can only write 0 to clear the flag.

(1) **Bits 7 to 4—Reserved:** Read-only bits, always read as 1.

(2) **Bit 3—Interrupt Request 3 Flag (IRQ3F):** Indicates that interrupt request 3 (IRQ3) has been input.

### Bit 3

<b>IRQ3F</b>	<b>Description</b>
0	Interrupt request 3 (IRQ3) has not been input (Initial value)
1	Interrupt request 3 (IRQ3) has been input and is waiting for interrupt service [Clearing conditions] <ul style="list-style-type: none"> <li>• Cleared to 0 automatically when the H8/500 CPU accepts IRQ3 and the interrupt vector is output</li> <li>• Can also be cleared by reading 1, then writing 0, in which case the pending IRQ3 interrupt request is deleted</li> </ul>

(3) **Bit 2—Interrupt Request 2 Flag (IRQ2F):** Indicates that interrupt request 2 (IRQ2) has been input.

#### Bit 2

IRQ2F	Description
0	Interrupt request 2 (IRQ2) has not been input (Initial value)
1	Interrupt request 2 (IRQ2) has been input and is waiting for interrupt service [Clearing conditions] <ul style="list-style-type: none"><li>• Cleared to 0 automatically when the H8/500 CPU accepts IRQ2 and the interrupt vector is output</li><li>• Can also be cleared by reading 1, then writing 0, in which case the pending IRQ2 interrupt request is deleted</li></ul>

(4) **Bit 1—Interrupt Request 1 Flag (IRQ1F):** Indicates that interrupt request 1 (IRQ1) has been input.

#### Bit 1

IRQ1F	Description
0	Interrupt request 1 (IRQ1) has not been input (Initial value)
1	Interrupt request 1 (IRQ1) has been input and is waiting for interrupt service [Clearing conditions] <ul style="list-style-type: none"><li>• Cleared to 0 automatically when the H8/500 CPU accepts IRQ1 and the interrupt vector is output</li><li>• Can also be cleared by reading 1, then writing 0, in which case the pending IRQ1 interrupt request is deleted</li></ul>

(5) **Bit 0—Reserved:** Read-only bit, always read as 1.

### 6.2.4 Internal Interrupts

There are 39 internal interrupt sources in the on-chip supporting modules. A different interrupt vector address is assigned to each source, so the interrupt handling routine does not have to determine which interrupt has occurred.

Priority levels from 7 to 0 are assigned to each module in IPRA to IPRF. DTEA to DTEF indicate which interrupts in each module are served by the DTC.

When an internal interrupt request is accepted, the T bit in SR is cleared to 0 and the interrupt mask level in I2 to I1 is set to the value selected in IPRA to IPRF.

## 6.3 Register Descriptions

### 6.3.1 Interrupt Priority Registers A to F

The six interrupt priority registers (IPRA to IPRF) assign priority levels from 7 to 0 to interrupt sources other than NMI. A reset initializes IPRA to IPRF to H'00.

The bit structure of IPRA to IPRF is shown next.

Bit	7	6	5	4	3	2	1	0
	0				0			
Initial value	0	0	0	0	0	0	0	0
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
	└─ <u>Upper four bits</u>				└─ <u>Lower four bits</u>			

**(1) Bits 7 to 4—Interrupt Priority, Upper Four Bits:** These bits select an interrupt priority level. Bit 7 must always be cleared to 0.

**(2) Bits 3 to 0—Interrupt Priority, Lower Four Bits:** These bits select an interrupt priority level. Bit 3 must always be cleared to 0.

The on-chip supporting modules are mapped onto the interrupt priority registers as shown in table 6-4. Each interrupt priority register is assigned two on-chip supporting modules. The upper four bits of the interrupt priority register specify the priority level of one module; the lower four bits specify the priority of the other module.

Table 6-5 indicates how priority levels are set in the interrupt priority registers. For example, to assign level 7 to SC11, set bits 6 to 4 in IPRF to 111.

**Table 6-4 On-Chip Supporting Modules and Interrupt Priority Registers**

Register	Bits 6 to 4	Bits 2 to 0
	On-Chip Supporting Module	On-Chip Supporting Module
IPRA	$\overline{\text{IRQ}}_0$ , WDT, A/D converter	$\overline{\text{IRQ}}_1$ to $\overline{\text{IRQ}}_3$
IPRB	IPU channel 1	IPU channel 1
IPRC	IPU channel 2	IPU channel 3
IPRD	IPU channel 4	IPU channel 5
IPRE	IPU channel 6	IPU channel 7
IPRF	SCI1	SCI2/SCI3

**Table 6-5 Interrupt Priority Settings in IPRH and IPRL**

Bits 6 to 4 or Bits 2 to 0	Interrupt Priority Level
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

### 6.3.2 Timing of Priority Changes

The interrupt controller requires two system clock cycles ( $2\phi$ ) to determine the priority level of an interrupt. Therefore, when an instruction modifies an instruction priority register (IPRA to IPRF), the new priority takes effect starting from the third state after that instruction has been executed.

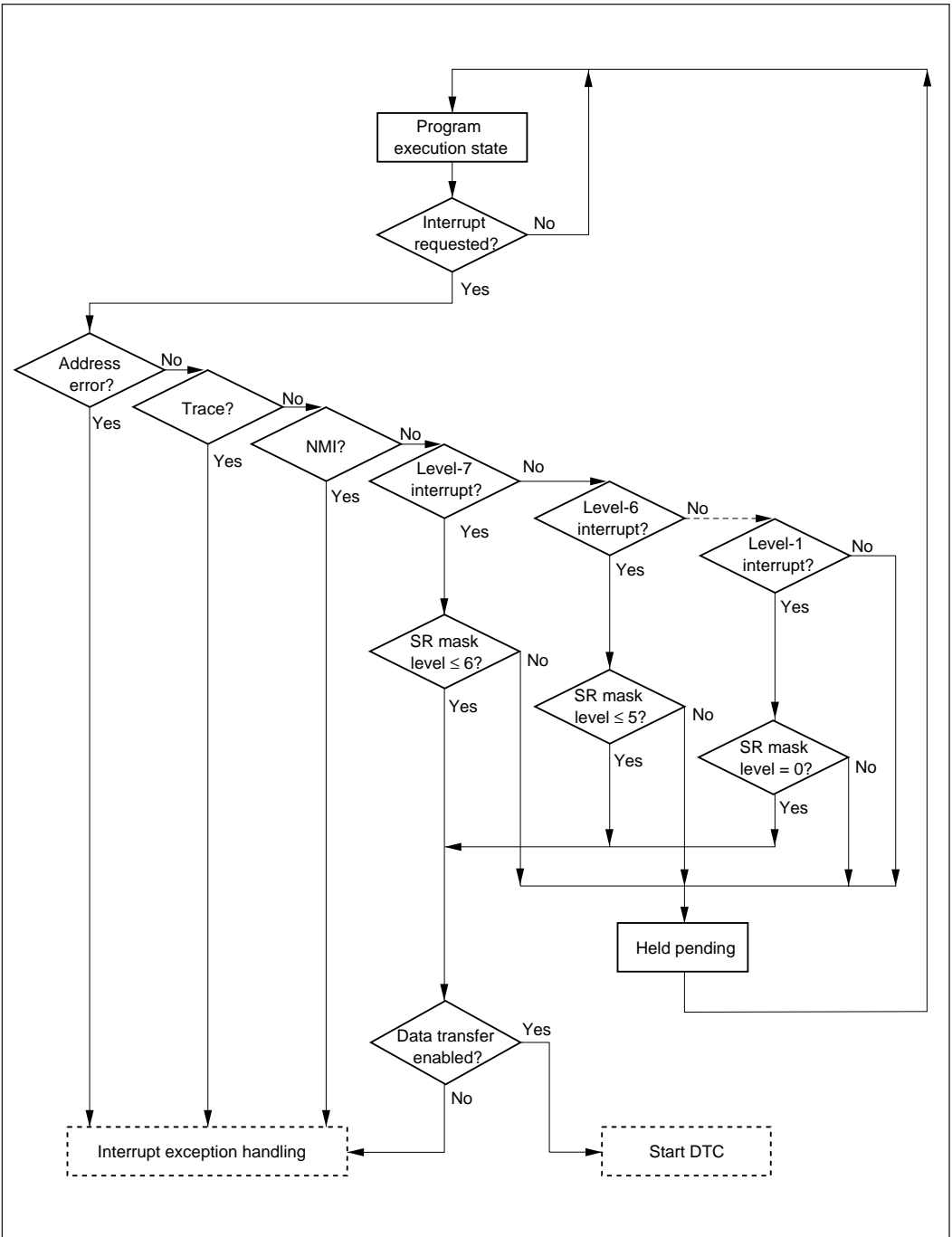
## 6.4 Interrupt Operations

Interrupt operations are described next.

### 6.4.1 Operations up to Interrupt Acceptance

Figure 6-2 is a flowchart of the interrupt sequence up to the point at which an interrupt is accepted.

1. The interrupt controller receives interrupt request signals from one or more on-chip supporting modules or external interrupt sources.
2. The interrupt controller checks the interrupt priorities assigned in IPRA to IPRF and selects the interrupt with the highest priority level. Interrupts with lower priorities remain pending. Among interrupts with the same assigned level, the interrupt controller determines priority as explained in table 6-3.
3. The interrupt controller compares the priority level of the selected interrupt request with the mask level in SR bits I2 to I0. If the priority level is equal to or less than the mask level, the interrupt request remains pending. If the priority level is higher than the mask level, the interrupt controller accepts the interrupt request.
4. After accepting an interrupt, the interrupt controller checks the corresponding bit in DTEA to DTEF. If this bit is set to 1, the data transfer controller is started. If it is cleared to 0, interrupt exception handling is started.

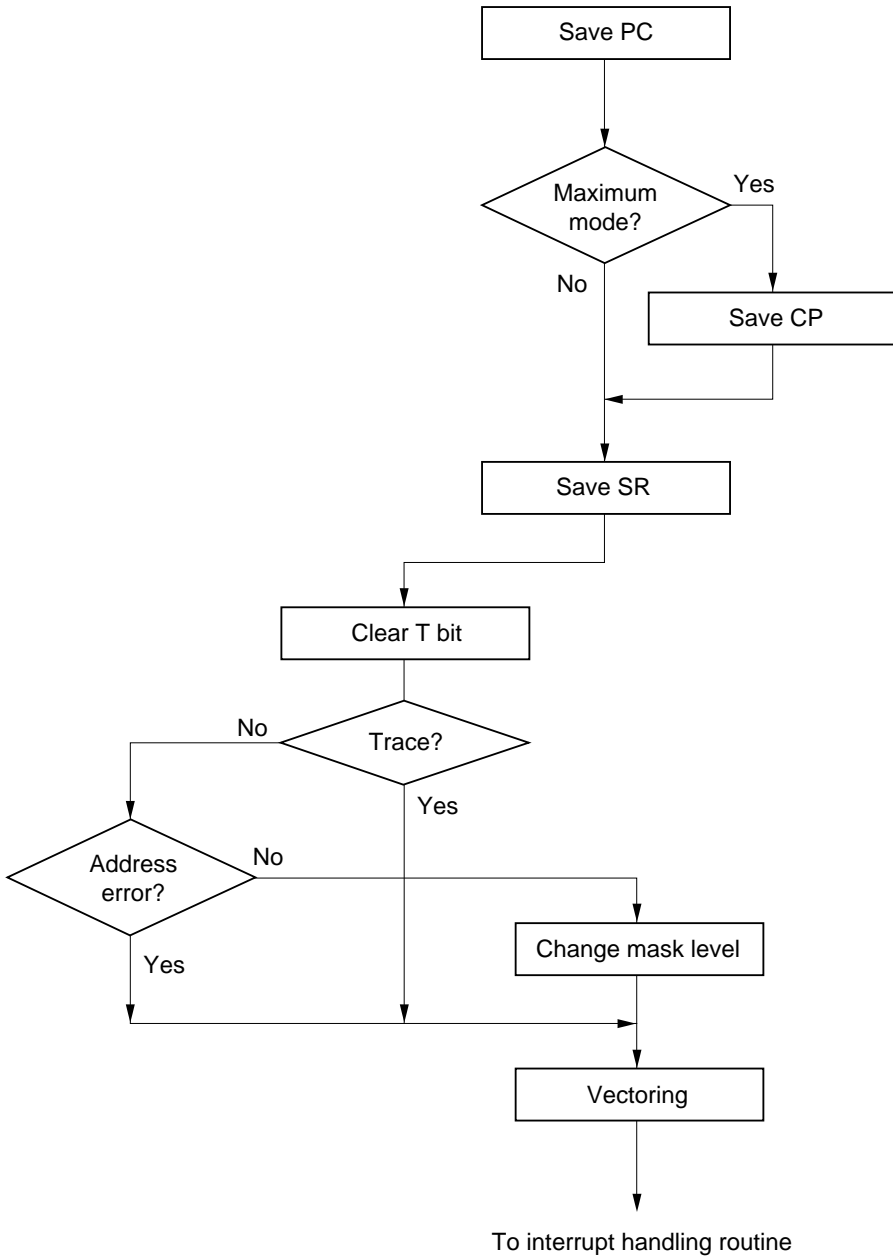


**Figure 6-2 Flowchart up to Interrupt Acceptance**

## 6.4.2 Interrupt Exception Handling

Interrupt exception handling is described below. Figure 6-3 shows a flowchart. For DTC operations, see section 7, “Data Transfer Controller.”

1. When the interrupt controller accepts an interrupt, after the H8/500 CPU finishes executing the current instruction, PC and SR (in minimum mode) or PC, CP, and SR (in maximum mode) are pushed on the stack, leaving the stack in the condition shown in section 6.4.4, “Stack after Interrupt Exception Handling.”
2. The interrupt controller clears the T bit in SR to 0, and sets the interrupt mask level (I2 to I0) to the priority level of the interrupt.
3. In minimum mode, the interrupt controller reads a one-word vector address corresponding to the accepted interrupt from the vector table and copies this word into PC. Execution of the interrupt handling routine then starts from the PC address. In maximum mode, the interrupt controller reads a two-word vector address corresponding to the accepted interrupt from the vector table, copies the lower byte of the first word into CP, and copies the second word into PC. Execution of the interrupt handling routine then starts from the address indicated by CP and PC.

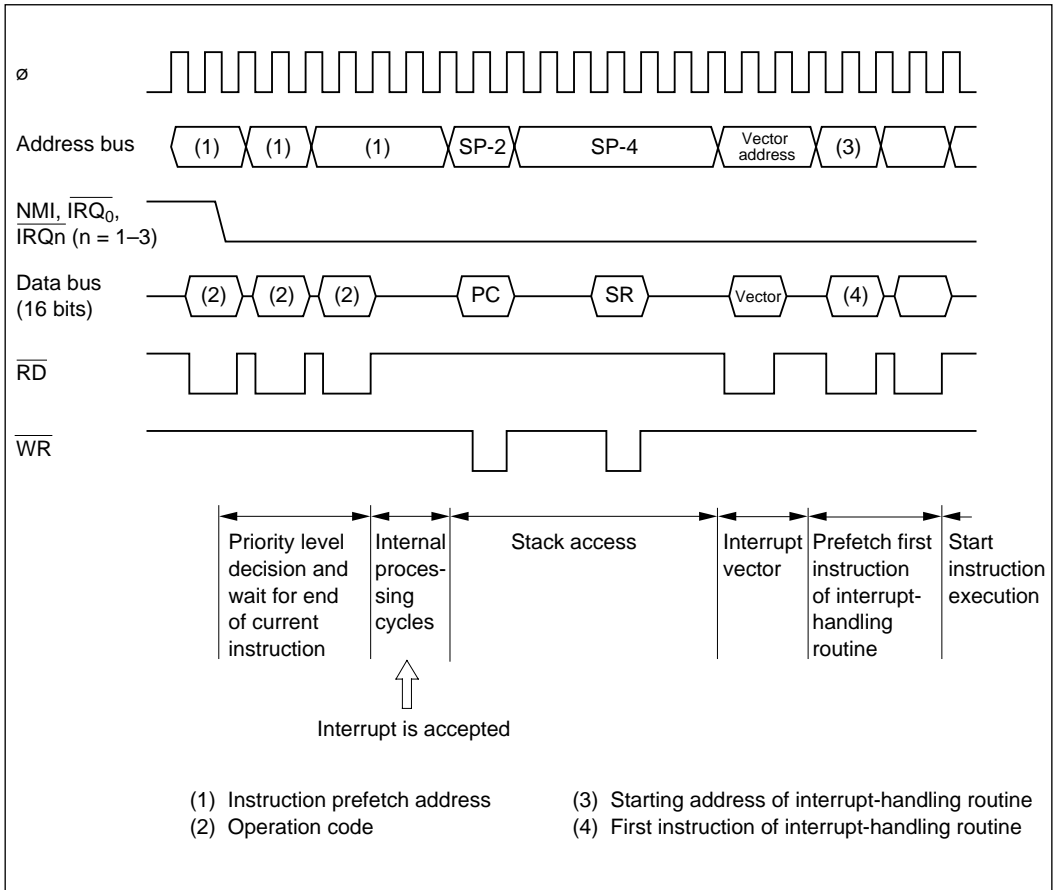


**Figure 6-3 Interrupt Exception Handling Flowchart**



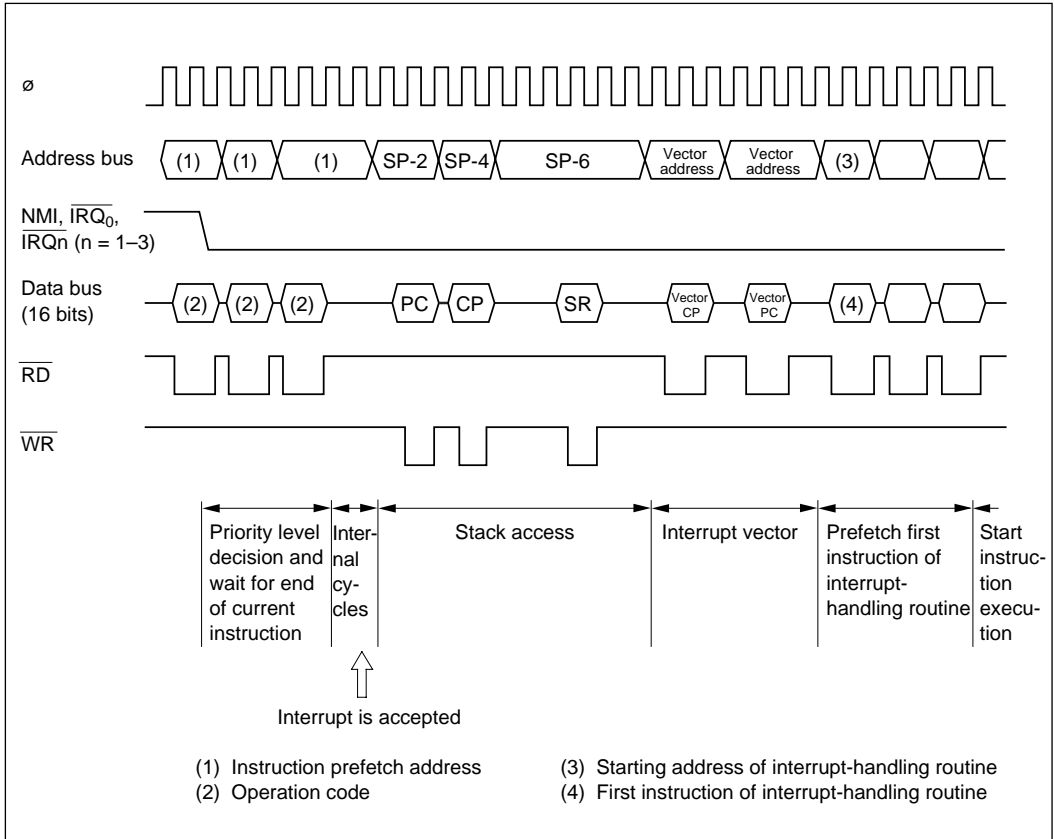
### 6.4.3 Interrupt Exception Handling Sequence

Figure 6-4 is a timing diagram of the interrupt sequence in minimum mode, for the case in which the interrupt handling routine starts at an even address and the program area and stack area are in the external 16-bit-bus two-state-access address space.



**Figure 6-4 Interrupt Sequence in Minimum Mode**

Figure 6-5 is a timing diagram of the interrupt sequence in maximum mode, for the case in which the interrupt handling routine starts at an even address and the program area and stack area are in the external 16-bit-bus two-state-access address space.

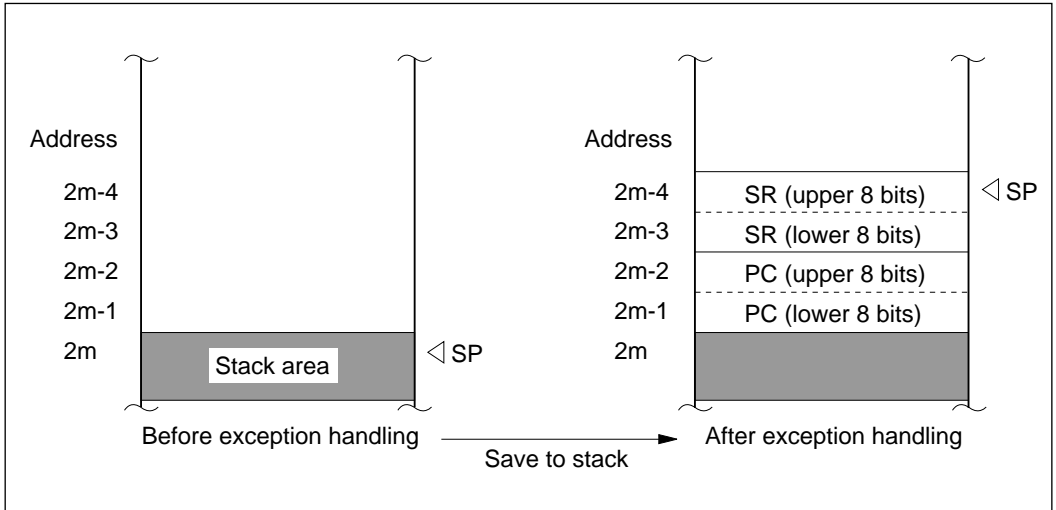


**Figure 6-5 Interrupt Sequence in Maximum Mode**

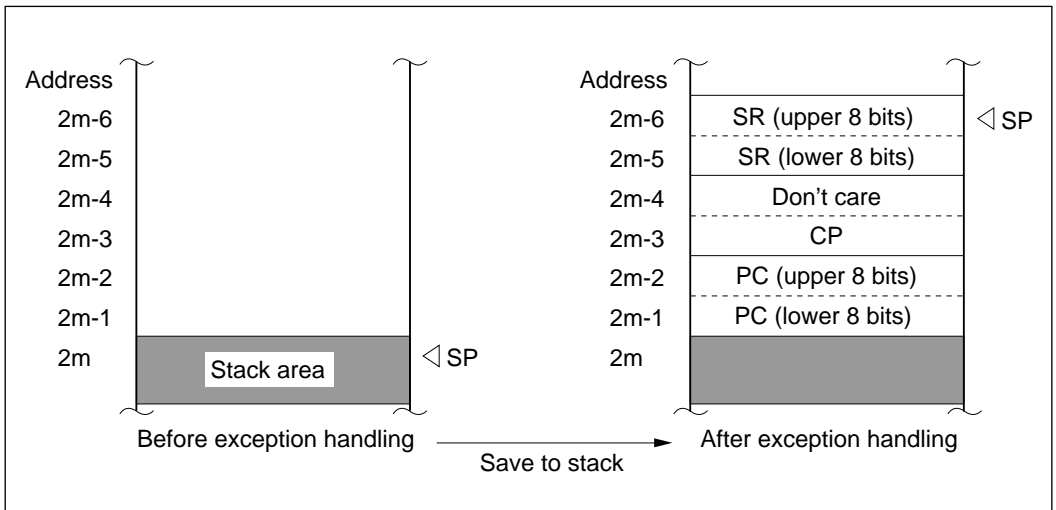
## 6.4.4 Stack after Interrupt Exception Handling

Figure 6-6 shows the stack before and after interrupt exception handling in minimum mode. Figure 6-7 shows the stack before and after interrupt exception handling in maximum mode. The PC value saved on the stack is the address of the next instruction to be executed.

SP must always point to an even address. If an odd address is set in SP, an address error will occur when the stack is accessed.



**Figure 6-6 Stack before and after Interrupt Exception Handling in Minimum Mode**

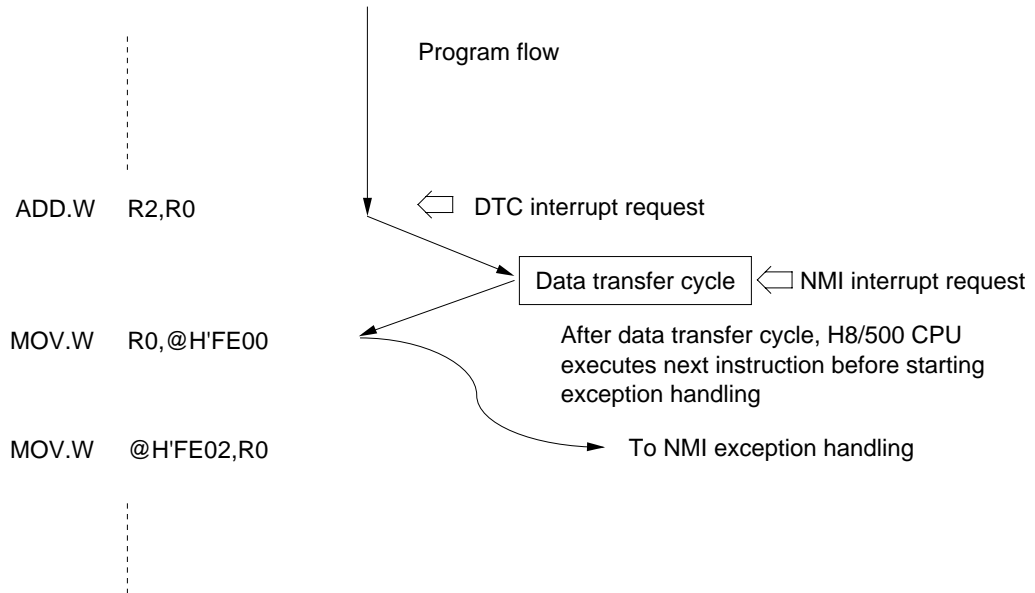


**Figure 6-7 Stack before and after Interrupt Exception Handling in Maximum Mode**

## 6.5 Interrupts during DTC Operation

If an interrupt is requested during a DTC data transfer cycle, the interrupt controller holds the interrupt pending until the data transfer cycle has been completed and the next instruction has been executed. An example is shown below.

### Example:



## 6.6 Interrupt Response Time

The H8/539F can access a memory area in two states via a 16-bit bus. Fastest interrupt service is obtained by placing the program and stack in this area. Table 6-6 indicates the interrupt response time in minimum mode. The maximum number of states occurs when the LDM instruction is executed with all registers specified.

**Table 6-6 Number of States before Interrupt Service in Minimum Mode**

Reason for Wait	Number of States			
	Stack Area: 16* <sup>1</sup>		Stack Area: 8* <sup>2</sup>	
	Instruction: 16* <sup>3</sup>	Instruction: 8* <sup>4</sup>	Instruction: 16* <sup>3</sup>	Instruction: 8* <sup>4</sup>
Interrupt priority decision and comparison with SR mask level	2	2	2	2
Maximum number of states to completion of current instruction	38	—	38	—
Saving of PC and SR	16	16	—	—
	—	—	28 + 6 m	28 + 6 m
Total number of states	56	92 + 16 m	68 + 6 m	104 + 22 m

Notes: 1. Stack area in 16-bit-bus two-state-access address space  
 2. Stack area in 8-bit-bus three-state-access address space  
 3. Instruction in 16-bit-bus two-state-access address space  
 4. Instruction in 8-bit-bus three-state-access address space  
 m: Number of wait states inserted in memory access

Table 6-7 indicates the interrupt response time in maximum mode. The maximum number of states occurs when the LDM instruction is executed with all registers specified.

**Table 6-7 Number of States before Interrupt Service in Maximum Mode**

Reason for Wait	Number of States			
	Stack Area: 16* <sup>1</sup>		Stack Area: 8* <sup>2</sup>	
	Instruction: 16* <sup>3</sup>	Instruction: 8* <sup>4</sup>	Instruction: 16* <sup>3</sup>	Instruction: 8* <sup>4</sup>
Interrupt priority decision and comparison with SR mask level	2	2	2	2
Maximum number of states to completion of current instruction	38	74 + 16 m	38	74 + 16 m
Saving of PC, CP, and SR	21	21	41 + 10 m	41 + 10 m
Total number of states	61	97 + 16 m	81 + 10 m	117 + 26 m

- Notes:
1. Stack area in 16-bit-bus two-state-access address space
  2. Stack area in 8-bit-bus three-state-access address space
  3. Instruction in 16-bit-bus two-state-access address space
  4. Instruction in 8-bit-bus three-state-access address space
- m: Number of wait states inserted in memory access

# Section 7 Data Transfer Controller

## 7.1 Overview

An interrupt-triggered data transfer controller (DTC) is included on-chip. The DTC can transfer data between memory and I/O, memory and memory, or I/O and I/O without using the CPU. For example, the DTC can set data in the registers of an on-chip supporting module or send data to an I/O port or serial communication interface (SCI) independently of program execution. The H8/500 CPU halts while the DTC is operating.

### 7.1.1 Features

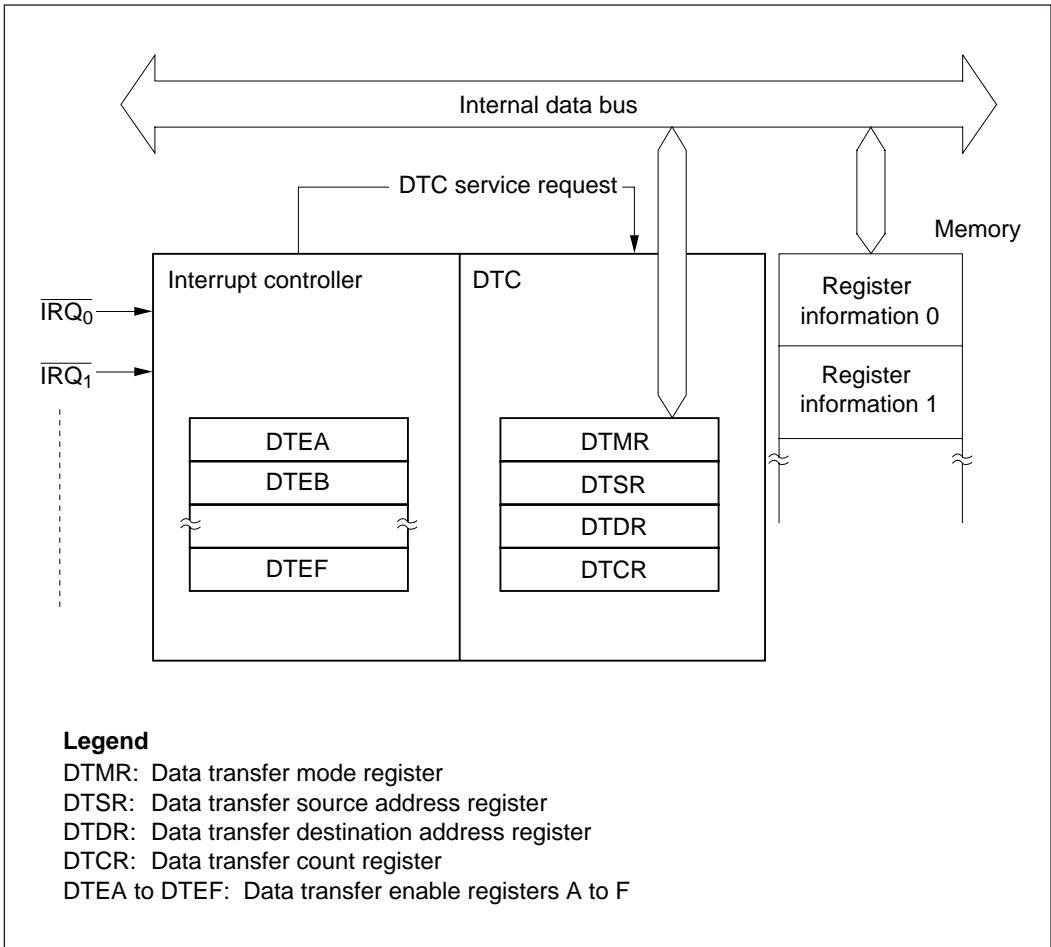
The features of the DTC are:

- The source address and destination address can be set anywhere in the 64-kbyte address space of page 0.
- The source address and destination address can be incremented or left unchanged after a data transfer.
- The DTC can be programmed to transfer one byte or one word of data per interrupt.
- A data transfer count of up to 65,536 bytes or words can be set in the data transfer counter register (DTCR).
- After a data transfer, if the data transfer count is zero, the interrupt request that started the DTC is transferred to the H8/500 CPU. The H8/500 CPU then starts normal interrupt exception handling.

## 7.1.2 Block Diagram

Figure 7-1 shows a block diagram of the data transfer controller.

When DTC service is requested, the DTC loads its control registers from memory with information corresponding to the interrupt source, transfers a byte or word of data, and writes any altered register information back to memory.



**Figure 7-1 Block Diagram of Data Transfer Controller**



### 7.1.3 Register Configuration

Table 7-1 summarizes the DTC control registers.

**Table 7-1 DTC Registers**

<b>Name</b>	<b>Abbreviation</b>	<b>R/W</b>
Data transfer mode register	DTMR	—
Data transfer source address register	DTSR	—
Data transfer destination address register	DTDR	—
Data transfer count register	DTCR	—

These registers cannot be accessed directly. To set information in the DTC control registers, software should alter the information in memory.

Starting of the DTC is controlled by the interrupt controller's data transfer enable registers.

Table 7-2 summarizes these registers.

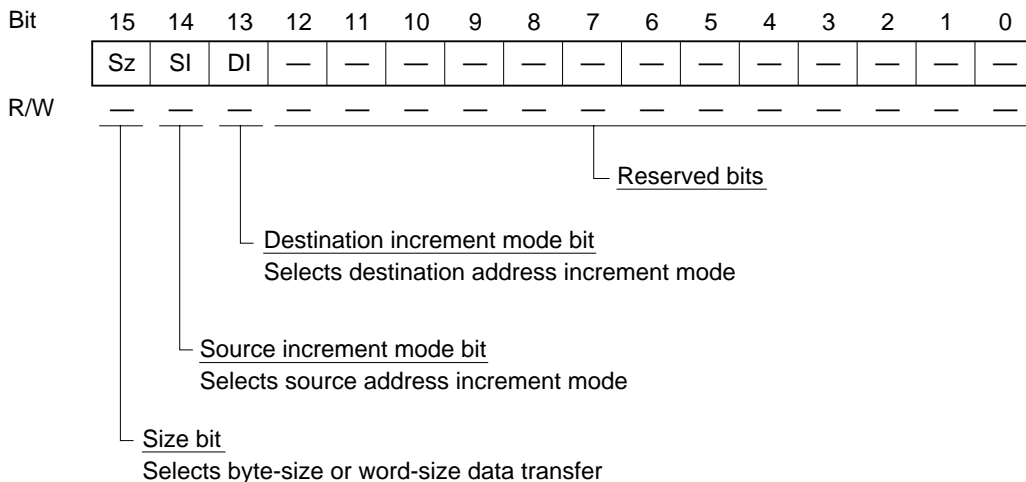
**Table 7-2 Data Transfer Enable Registers**

<b>Address</b>	<b>Name</b>	<b>Abbreviation</b>	<b>R/W</b>	<b>Initial Value</b>
H'FF08	Data transfer enable register A	DTEA	R/W	H'00
H'FF09	Data transfer enable register B	DTEB	R/W	H'00
H'FF0A	Data transfer enable register C	DTEC	R/W	H'00
H'FF0B	Data transfer enable register D	DTED	R/W	H'00
H'FF0C	Data transfer enable register E	DTEE	R/W	H'00
H'FF0D	Data transfer enable register F	DTEF	R/W	H'00

## 7.2 Register Descriptions

### 7.2.1 Data Transfer Mode Register

The data transfer mode register (DTMR) is a 16-bit register that selects the data size and specifies whether to increment the source and destination addresses. The DTMR bit structure is shown next.



(1) **Bit 15—Size (Sz):** Selects byte-size or word-size data transfer.

<b>Bit 15</b>	
<b>Sz</b>	<b>Description</b>
0	Byte transfer
1	Word (two-byte) transfer*

Note: \* For word transfer, DTSR and DTDR must indicate even addresses.

(2) **Bit 14—Source Increment Mode (SI):** Specifies whether to increment the source address.

<b>Bit 14</b>	
<b>SI</b>	<b>Description</b>
0	Not incremented
1	1. If Sz = 0: incremented by 1 after each data transfer 2. If Sz = 1: incremented by 2 after each data transfer

(3) **Bit 13—Destination Increment Mode (DI):** Specifies whether to increment the destination address.

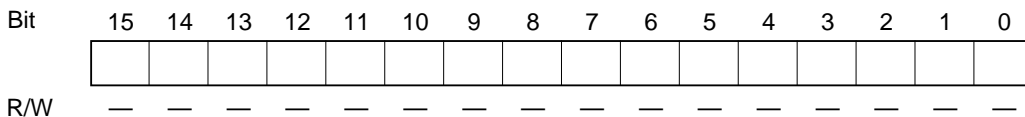
**Bit 13**

DI	Description
0	Not incremented
1	1. If Sz = 0: incremented by 1 after each data transfer 2. If Sz = 1: incremented by 2 after each data transfer

(4) **Bits 12 to 0—Reserved:** Reserved bits.

**7.2.2 Data Transfer Source Address Register**

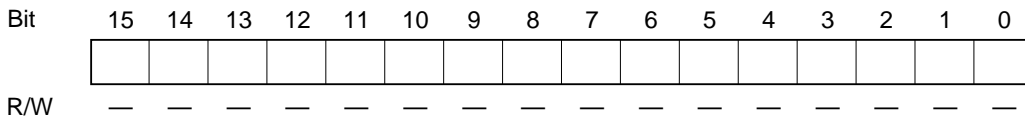
The data transfer source address register (DTSR) is a 16-bit register that designates the data transfer source address. The DTSR bit structure is shown next.



For word transfer the source address must be even. In maximum mode, the source address is implicitly located in page 0.

**7.2.3 Data Transfer Destination Address Register**

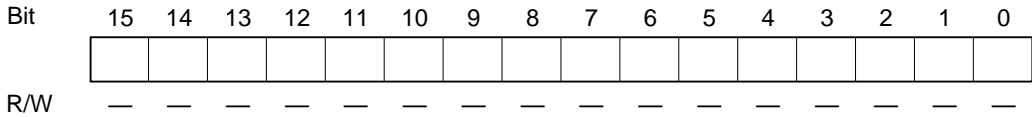
The data transfer destination address register (DTDR) is a 16-bit register that designates the data transfer destination address. The DTDR bit structure is shown next.



For word transfer the destination address must be even. In maximum mode, the destination address is implicitly located in page 0.

### 7.2.4 Data Transfer Count Register

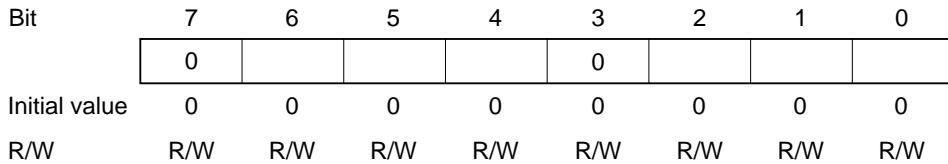
The data transfer count register (DTCR) is a 16-bit register that designates the number of bytes or words to be transferred. The initial count can be set from 1 to 65,536. A register value of 0 designates an initial count of 65,536. The DTCR bit structure is shown next.



The data transfer count register is decremented automatically after each byte or word is transferred. When the count reaches 0, indicating that the designated number of bytes or words have been transferred, the DTC sends the H8/500 CPU an interrupt request with the same interrupt source that started the data transfer.

### 7.2.5 Data Transfer Enable Registers A to F

The six data transfer enable registers (DTEA to DTEF) specify whether an interrupt starts the DTC. (Certain interrupts, such as NMI, cannot start the DTC.) The bit structure of DTEA to DTEF is shown next.



The bits in these registers are assigned to interrupts as indicated in table 7-3.

If the bit for a certain interrupt is set to 1, that interrupt is regarded as a request for DTC service. If the bit is cleared to 0, the interrupt is regarded as an H8/500 CPU interrupt request.

Only the interrupts indicated in table 7-3 can request DTC service in the H8/539F. DTE bits not assigned to any interrupt (indicated by “—” in table 7-3) must be cleared to 0.

**Table 7-3 Bit Assignments of Data Transfer Enable Registers**

Register	On-Chip Supporting Module	Bits 7 to 4				On-Chip Supporting Module	Bits 3 to 0			
		7	6	5	4		3	2	1	0
DTEA	$\overline{\text{IRQ}}_0$ , ADI	—	ADI	(IRQ0)	IRQ0	$\overline{\text{IRQ}}_{1-3}$	—	IRQ3	IRQ2	IRQ1
DTEB	IPU (CH1)	—	CMI1,2	IMI2	IMI1	IPU (CH1)	—	CMI3,4	IMI4	IMI3
DTEC	IPU (CH2)	—	CMI1,2	IMI2	IMI1	IPU (CH3)	—	CMI1,2	IMI2	IMI1
DTED	IPU (CH4)	—	CMI1,2	IMI2	IMI1	IPU (CH5)	—	CMI1,2	IMI2	IMI1
DTEE	IPU (CH6)	—	—	IMI2	IMI1	IPU (CH7)	—	—	IMI2	IMI1
DTEF	SCI1	—	TI	RI	—	SCI2/SCI3	—	TI	RI	—

### 7.2.6 Note on Timing of DTE Modifications

The interrupt controller requires two system clock cycles ( $2\phi$ ) to determine the priority level of an interrupt. When an instruction modifies one of registers DTEA to DTEF, the new setting takes effect starting from the third state after the instruction has been executed.

## 7.3 Operation

DTC operations are described next.

### 7.3.1 DTC Operations

Data transfer operations when the DTC is activated are described below. Figure 7-2 is a flowchart of the DTC operations.

Figure 7-2 is a flowchart of the data transfer operations performed by the DTC. For operations from the occurrence of an interrupt until the DTC is activated, see section 6.4.1, “Operations up to Interrupt Acceptance.”

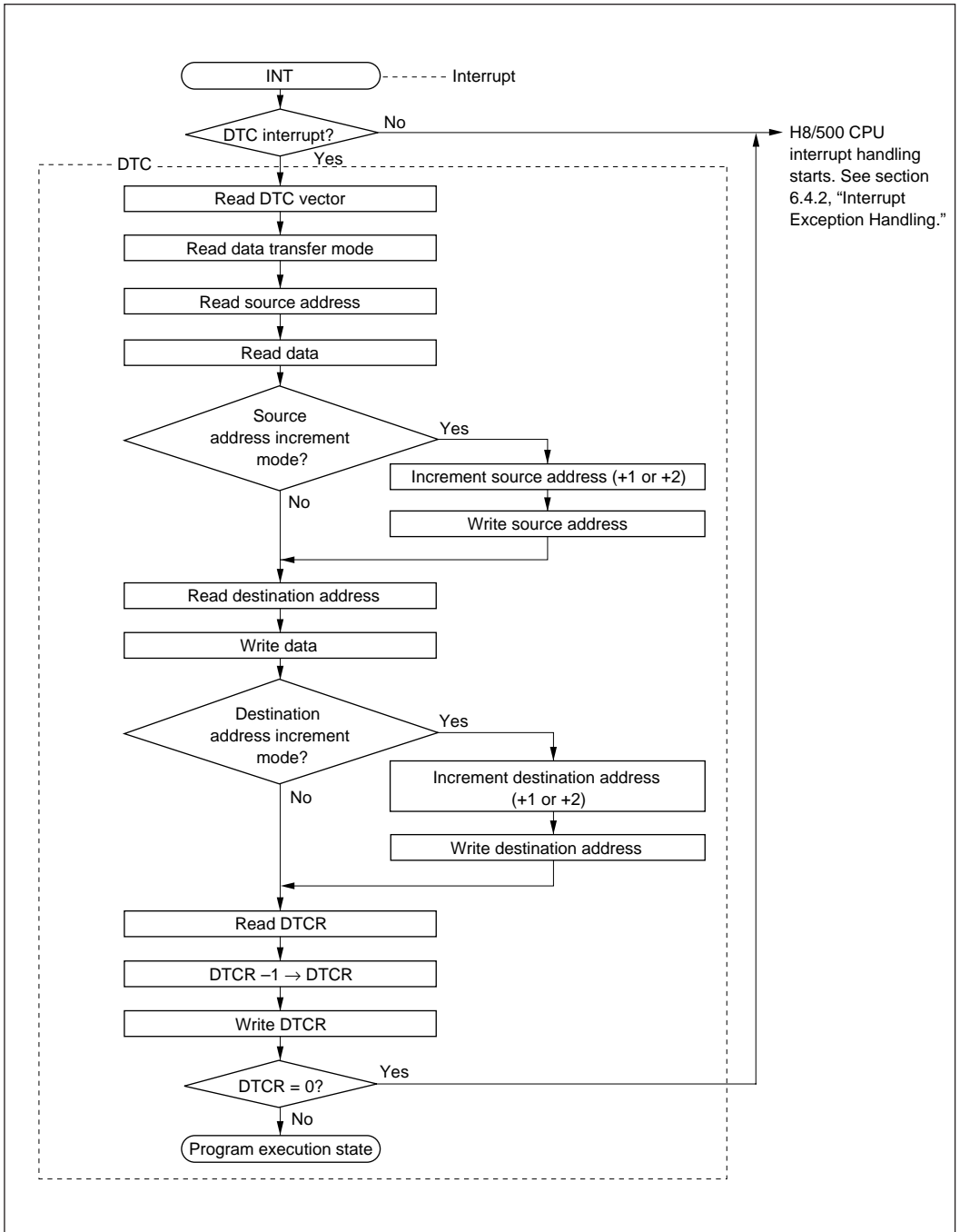
1. From the DTC vector table, the DTC reads the address at which the register information for the interrupt is stored in memory and loads the stored information into its control registers.

When the DTC is activated, the interrupt source that activated the DTC is cleared, except for interrupts from the serial communication interface.

2. The DTC transfers the data and increments the source and destination addresses as required, then decrements DTCCR.

If the DTC was activated by an interrupt from the serial communication interface, the interrupt source is cleared when the DTC accesses the transmit data register (TDR) or receive data register (RDR).

3. The DTC writes updated register information back to memory.
4. If the DTCCR value is 0, the H8/500 CPU starts interrupt exception handling for the interrupt that activated the DTC.



**Figure 7-2 Flowchart of DTC Operations**

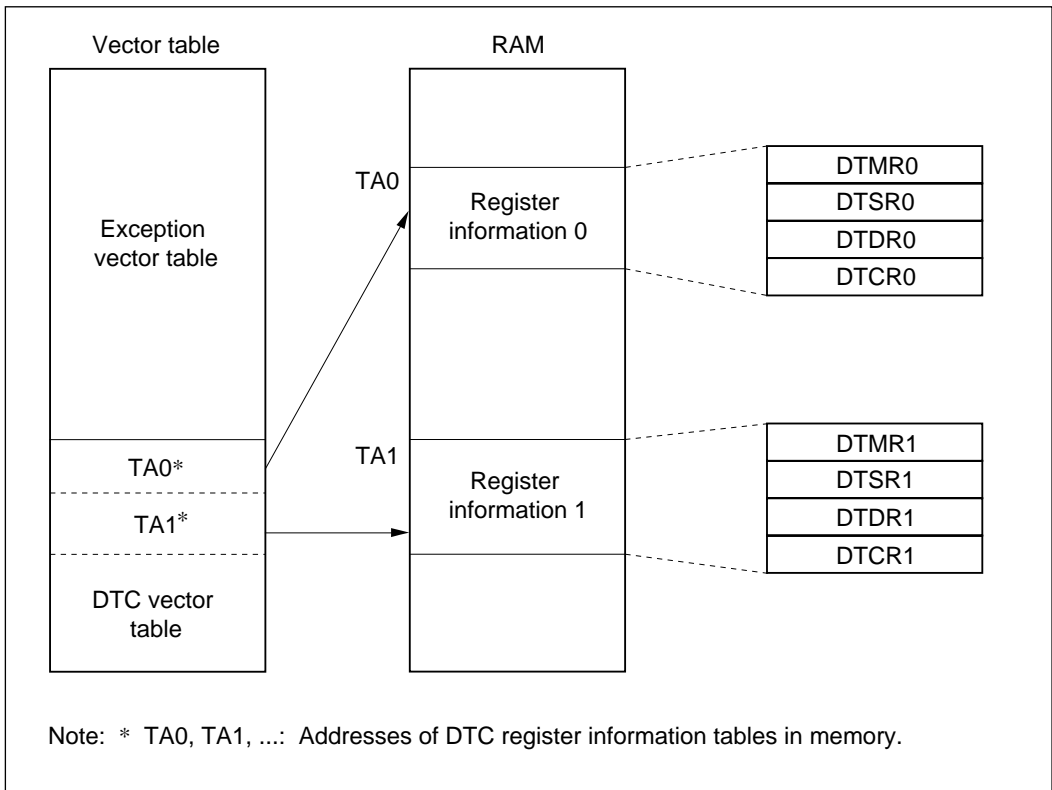
### 7.3.2 DTC Vector Table

Figure 7-3 shows how the DTC vector table works.

For each interrupt that can request DTC service, the DTC vector table provides a pointer to an address in memory where the DTC control register information for that interrupt is stored.

Register information tables can be placed in any available locations in page 0.

Figure 7-3 shows an example in which the register information is located in RAM. Register information can also be stored in ROM if there is no need to update the information after each transfer (if the source and destination addresses are not incremented and the desired data transfer count is one).

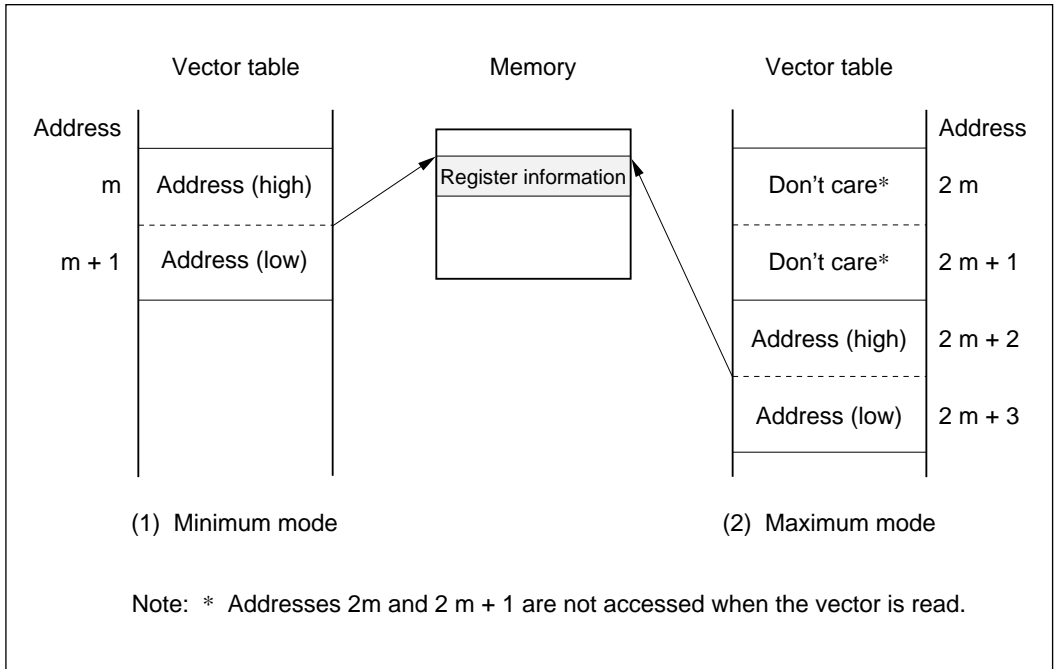


**Figure 7-3 DTC Vector Table**

The DTC vector table structure differs between minimum and maximum modes. In maximum mode there is no page specification: page 0 is assumed implicitly.

Figure 7-4 shows a DTC vector table entry in minimum and maximum mode.





**Figure 7-4 DTC Vector Table Entry**

Table 7-4 lists the address of the entry in the DTC vector table for each interrupt source.

**Table 7-4 Addresses of DTC Vectors**

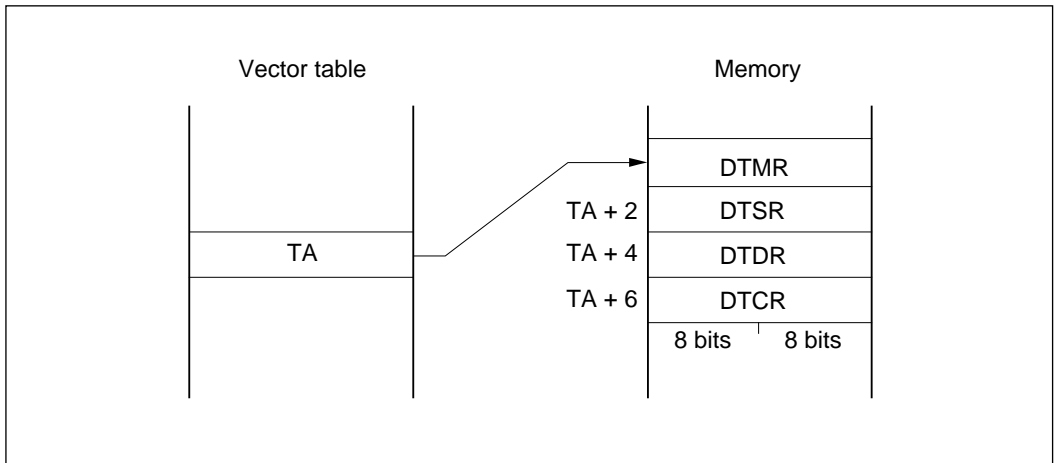
Interrupt Source	Address of Vector Table Entry		
	Minimum Mode	Maximum Mode	
IRQ0		H'00C0–00C1	H'0180–0183
Interval timer		H'00C2–00C3	H'0184–0187
AD converter	ADI	H'00C4–00C5	H'0188–018B
IRQ1		H'00C8–00C9	H'0190–0193
IRQ2		H'00CA–00CB	H'0194–0197
IRQ3		H'00CC–00CD	H'0198–019B
IPU channel 1	IMI1	H'00D0–00D1	H'01A0–01A3
	IMI2	H'00D2–00D3	H'01A4–01A7
	CMI1/CMI2	H'00D4–00D5	H'01A8–01AB
	—	—	—
IPU channel 1	IMI3	H'00D8–00D9	H'01B0–01B3
	IMI4	H'00DA–00DB	H'01B4–01B7
	CMI3/CMI4	H'00DC–00DD	H'01B8–01BB
	—	—	—
IPU channel 2	IMI1	H'00E0–00E1	H'01C0–01C3
	IMI2	H'00E2–00E3	H'01C4–01C7
	CMI1/CMI2	H'00E4–00E5	H'01C8–01CB
	—	—	—
IPU channel 3	IMI1	H'00E8–00E9	H'01D0–01D3
	IMI2	H'00EA–00EB	H'01D4–01D7
	CMI1/CMI2	H'00EC–00ED	H'01D8–01DB
	—	—	—
IPU channel 4	IMI1	H'00F0–00F1	H'01E0–01E3
	IMI2	H'00F2–00F3	H'01E4–01E7
	CMI1/CMI2	H'00F4–00F5	H'01E8–01EB
	—	—	—
IPU channel 5	IMI1	H'00F8–00F9	H'01F0–01F3
	IMI2	H'00FA–00FB	H'01F4–01F7
	CMI1/CMI2	H'00FC–00FD	H'01F8–01FB
	—	—	—
IPU channel 6	IMI1	H'00A0–00A1	H'0140–0143
	IMI2	H'00A2–00A3	H'0144–0147
	—	—	—
IPU channel 7	IMI1	H'00A8–00A9	H'0150–0153
	IMI2	H'00AA–00AB	H'0154–0157
	—	—	—

**Table 7-4 Addresses of DTC Vectors (H8/539) (cont)**

Interrupt Source	Address of Vector Table Entry	
	Minimum Mode	Maximum Mode
SCI1	—	—
	RI1	H'00B2–00B3
	TI1	H'00B4–00B5
	—	—
SCI2/SCI3	—	—
	RI2/RI3	H'00BA–00BB
	TI2/TI3	H'00BC–00BD
	—	—

**7.3.3 Location of Register Information in Memory**

For each interrupt, the DTC control register information is stored in memory in the order shown in figure 7-5.



**Figure 7-5 Order of Register Information in Memory**

### 7.3.4 Number of States per Data Transfer

Table 7-5 lists the number of states required per data transfer, assuming that the DTC control register information is stored in the 16-bit-bus two-state-access address space.

**Table 7-5 Number of States per Data Transfer**

Increment Mode		16-Bit-Bus 2-State-Access ↔ Address Space	On-Chip Supporting Module	8-Bit-Bus 3-State-Access ↔ Address Space	On-Chip Supporting Module
Source (SI)	Destination (DI)	Byte Transfer	Word Transfer	Byte Transfer	Word Transfer
0	0	31	34	32	38
0	1	33	36	34	40
1	0	33	36	34	40
1	1	35	38	36	42

Note: Numbers in the table are the number of states.

The values in table 7-5 are calculated from the formula:

$$N = 26 + 2 \times SI + 2 \times DI + M_S + M_D$$

Where  $M_S$  and  $M_D$  have the following meanings:

$M_S$ : Number of states for reading source data

$M_D$ : Number of states for writing data to destination

The values of  $M_S$  and  $M_D$  depend on the data location as follows:

1. Byte or word data in 16-bit-bus two-state-access address space: 2 states
2. Byte data in eight-bit-bus three-state-access address space or on-chip supporting module: 3 states
3. Word data in eight-bit-bus three-state-access address space or on-chip supporting module: 6 states

If the DTC control register information is stored in the eight-bit-bus three-state-access address space,  $20 + 4 \times SI + 4 \times DI$  must be added to the values in table 7-5.

Table 7-6 indicates the number of additional states between the occurrence of an interrupt request and the starting of the DTC (states during which the interrupt controller checks priority and waits for execution of the current instruction to end). At maximum, this number of states is the sum of

the values indicated for items No. 1 and 2 in table 7-4.

If the data transfer count is 0 at the end of a data transfer cycle, the number of states from the end of the data transfer cycle until the first instruction of the interrupt-handling routine is executed is the value given for item No. 3 in table 7-4. The maximum number of states in table 7-6 occurs when the LDM instruction is executed with all registers specified.

**Table 7-6 Number of States before Interrupt Service**

No.	Reason for Wait	Number of States		
		Minimum Mode	Maximum Mode	
1	Interrupt priority decision and comparison with mask level in SR	2		
2	Number of states to completion of current instruction	Instruction is in 16-bit-bus two-state-access address space	(LDM instruction specifying all registers) 38	
		Instruction is in 8-bit-bus three-state-access address space	(LDM instruction specifying all registers) 74 + 16 m	
3	Number of states from saving of PC and SR or PC, CP, and SR until prefetching of first instruction of interrupt-handling routine	Instruction is in 16-bit-bus two-state-access address space	16	21
		Instruction is in 8-bit-bus three-state-access address space	28 + 6 m	41 + 10 m

**Notation**

m: Number of wait states inserted in memory access

## 7.4 Procedure for Using DTC

The procedure for using the DTC is explained next. Figure 7-6 shows the flowchart.

### Procedure for Using the DTC

1. DTC register setup: Set the appropriate DTMR, DTSR, DTDR, and DTCR register information in the memory location indicated in the DTC vector table.
2. DTE<sub>n</sub>, IPR<sub>n</sub> (n = A to F), and SR setup: Set the data transfer enable bit of the pertinent interrupt to 1, and set the priority of the interrupt source (in the interrupt priority register) and the interrupt mask level (in the CPU status register) so that the interrupt can be accepted.
3. Interrupt enabling: Set the interrupt enable bit for the interrupt source in the control register of the on-chip supporting module (or IRQ control register).

Following these preparations, the DTC will be started each time the interrupt occurs.

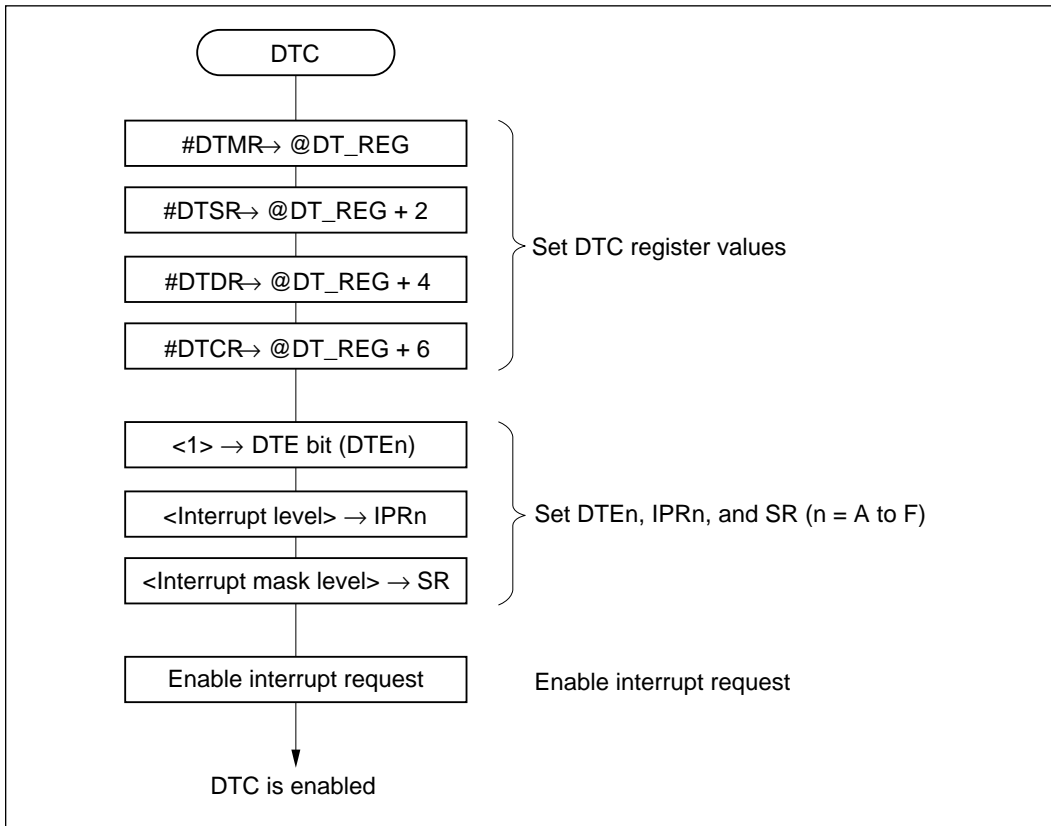


Figure 7-6 Procedure for Using DTC

## 7.5 Example

(1) **Purpose:** To receive 128 bytes of serial data via serial communication interface.

(2) **Conditions:**

- Operating mode: minimum mode.
- Receive data is to be stored in consecutive addresses starting at H'FC00.
- The DTC vector table contains H'F6 at address H'00B2 and H'80 at address H'00B3.
- The desired interrupt mask level in the CPU status register is 4, and the desired SCI interrupt priority level is 5.

Table 7-7 shows the DTC control register information to be set in RAM.

**Table 7-7 DTC Control Register Information Set in RAM**

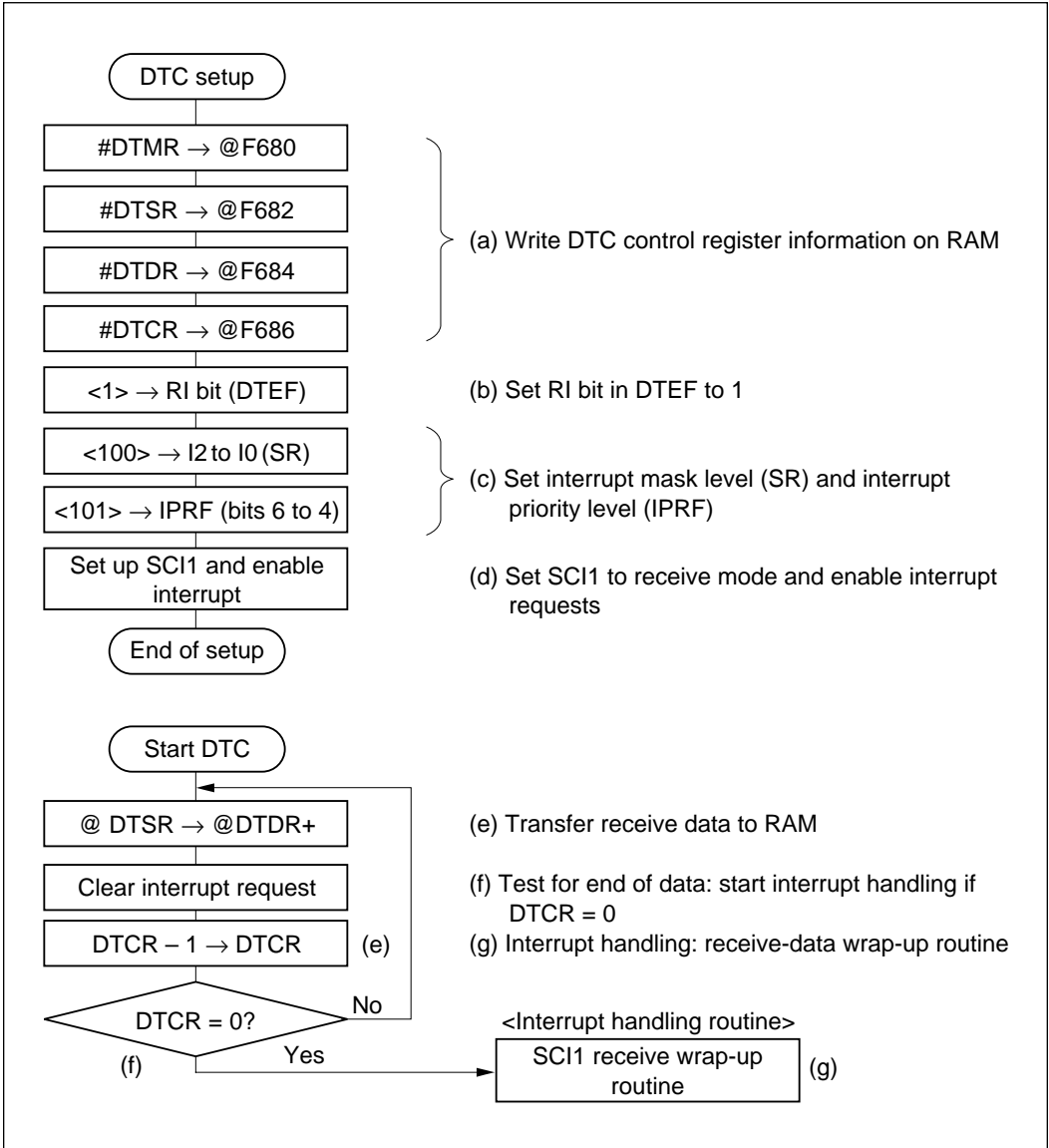
Register	Setting	Value
DTMR	Byte transfer Source address fixed Destination address incremented	H'2000
DTSR	Address of SCI1 receive data register	H'FECD
DTDR	Address H'FC00	H'FC00
DTCR	Transfer count (128)	H'0080

(3) **Operation**

- (a) Software sets DTMR, DTSR, DTDR, and DTCR information in RAM addresses H'F680 to H'F687 as shown in table 7-7.
- (b) Software sets the RI (SCI1 Receive Interrupt) bit in data transfer enable register F (DTEF) to 1.
- (c) Software sets the interrupt mask level in SR bits I2 to I0 to 4, and the SCI1 interrupt priority level in the upper four bits of interrupt priority register F (IPRF) to 0101 (5).
- (d) Software sets SCI1 to the appropriate receive mode, and sets the receive interrupt enable bit (RIE) in the serial control register (SCR) to 1 to enable receive interrupts.
- (e) Thereafter, each time SCI1 receives one byte of data, the DTC is activated and transfers the byte of receive data into RAM. The DTC automatically clears the SCI1 receive interrupt request.

- (f) When 128 bytes have been transferred (DTCR = 0), SCI1 receive interrupt exception handling begins.
- (g) The interrupt-handling routine executes a receive wrap-up routine.

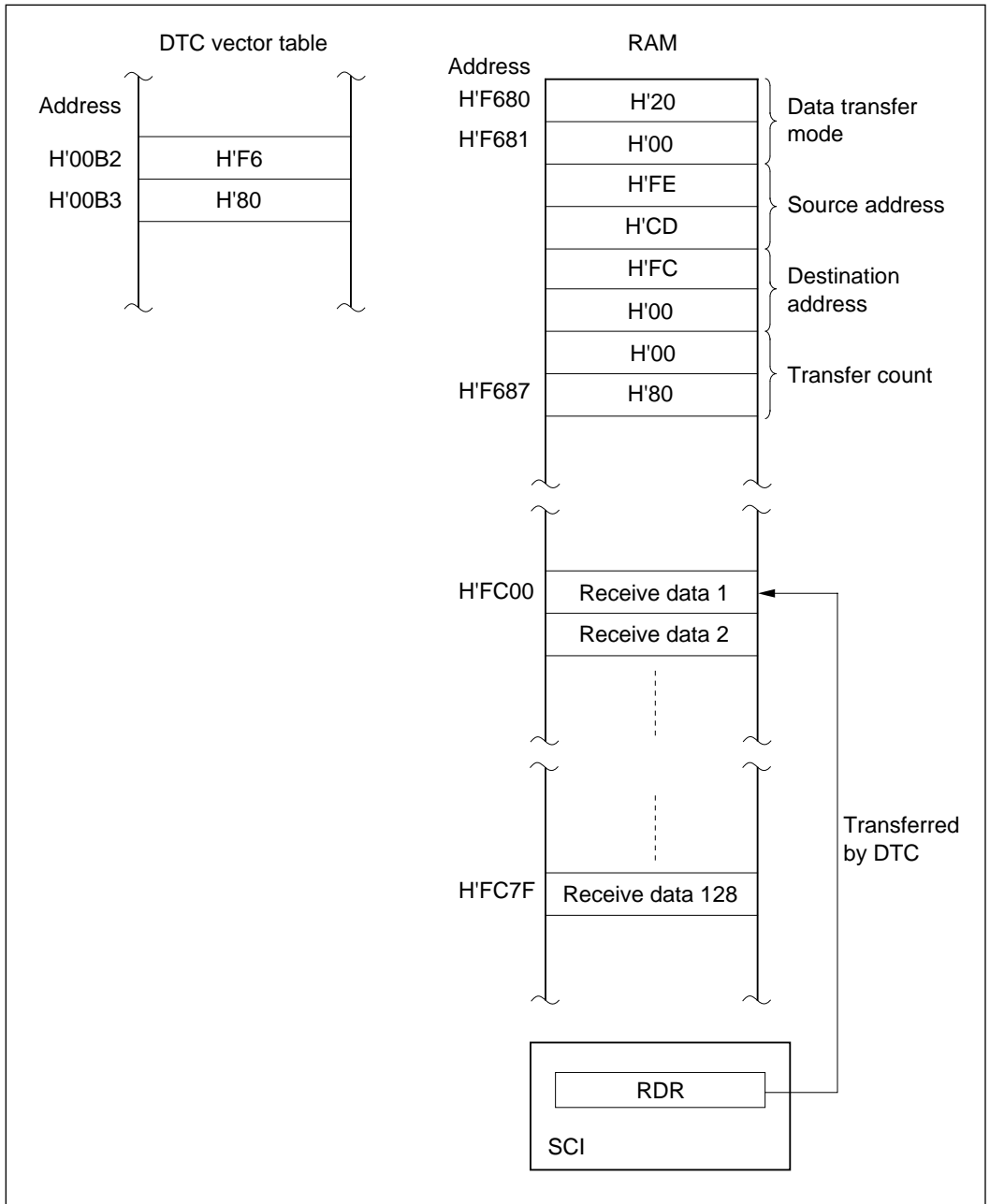
Figure 7-7 is a flowchart for this example.



**Figure 7-7 Flowchart for DTC Example**



Figure 7-8 shows an example of the use of the DTC for continuous SCI reception.



**Figure 7-8 Example of Use of DTC to Receive Continuous Serial Data**

# Section 8 Wait-State Controller

## 8.1 Overview

For interfacing to low-speed external devices, an on-chip wait-state controller (WSC) can insert wait states ( $T_W$ ) into bus cycles. The wait function can be used in CPU and DTC access cycles to the external three-state-access address space. It is not used in access to the two-state-access address space or the on-chip register area (H'FE80 to H'FFFF).

Wait states are inserted between the  $T_2$  state and  $T_3$  state in the bus cycle. The number of wait states can be selected by a value set in the wait control register (WCR), or by holding the  $\overline{\text{WAIT}}$  pin low for the required interval.

### 8.1.1 Features

The features of the wait-state controller are:

- Selection of three operating modes

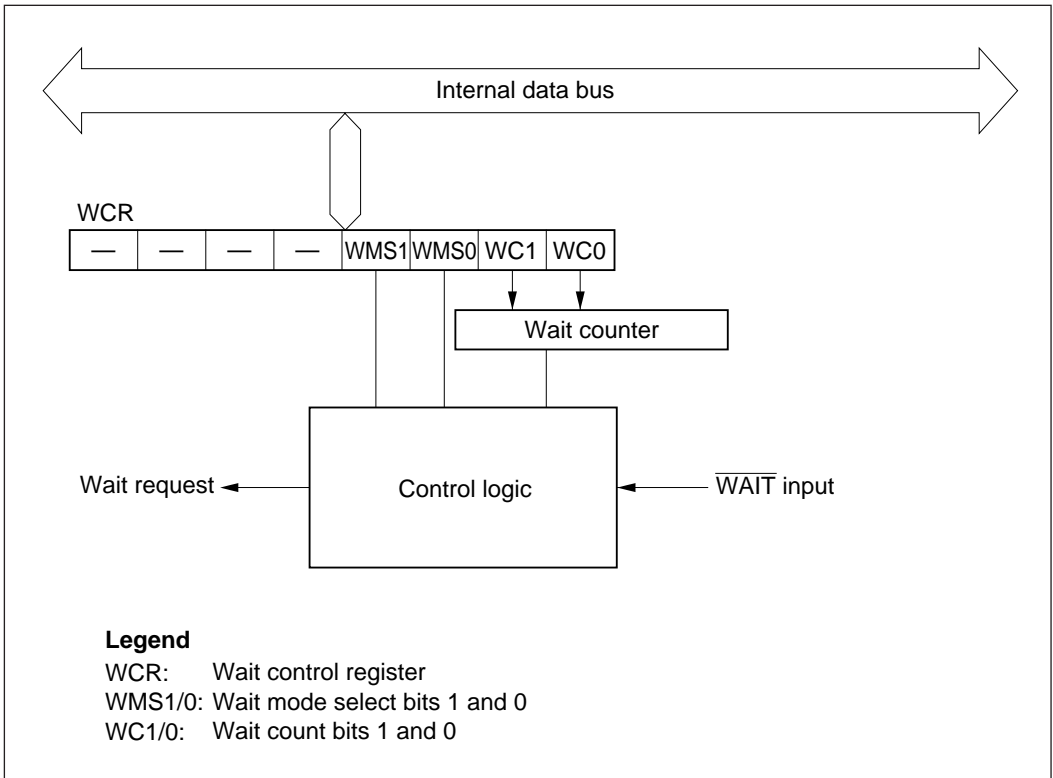
Programmable wait mode, pin wait mode, or pin auto-wait mode

- Selection of number of wait states

0, 1, 2, or 3 wait states can be inserted, and 4 or more wait states can be inserted in pin wait mode by holding the  $\overline{\text{WAIT}}$  pin low.

### 8.1.2 Block Diagram

Figure 8-1 shows a block diagram of the wait-state controller.



**Figure 8-1 Block Diagram of Wait State Controller**

### 8.1.3 Register Configuration

Table 8-1 summarizes the wait control register.

**Table 8-1 Wait Control Register**

Address	Name	Abbreviation	R/W	Initial Value
H'FF14	Wait control register	WCR	R/W	H'F3

## 8.2 Wait Control Register

The wait control register (WCR) is an eight-bit register that specifies the wait mode and the number of wait states to be inserted. The WCR bit structure is shown next.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	WMS1	WMS0	WC1	WC0
Initial value	1	1	1	1	0	0	1	1
R/W	—	—	—	—	R/W	R/W	R/W	R/W

Wait count 1 and 0  
 These bits indicate the number of wait states to be inserted

Wait mode select 1 and 0  
 These bits select the wait mode

Reserved bits

WCR is initialized to H'F3 by a reset and in hardware standby mode. WCR is not initialized in software standby mode.

(1) **Bits 7 to 4—Reserved:** Read-only bits, always read as 1.

(2) **Bits 3 and 2—Wait Mode Select 1 and 0 (WMS1 and WMS0):** These bits select the wait mode.

Bit 3	Bit 2	Description
WMS1	WMS0	
0	0	Programmable wait mode (Initial value)
0	1	No wait states (Tw) inserted, regardless of wait count
1	0	Pin wait mode
1	1	Pin auto-wait mode

**(3) Bits 1 and 0—Wait Count 1 and 0 (WC1 and WC0):** These bits specify the number of wait states to be inserted. Wait states ( $T_W$ ) are inserted only in bus cycles in which the CPU or DTC accesses the external three-state-access address space.

Bit 1	Bit 0	Description
WC1	WC0	
0	0	No programmable wait states ( $T_W$ ) inserted
0	1	1 wait state inserted
1	0	2 wait states inserted
1	1	3 wait states inserted (Initial value)

## 8.3 Operation

Table 8-2 summarizes the operation of the three wait modes.

**Table 8-2 Wait Modes**

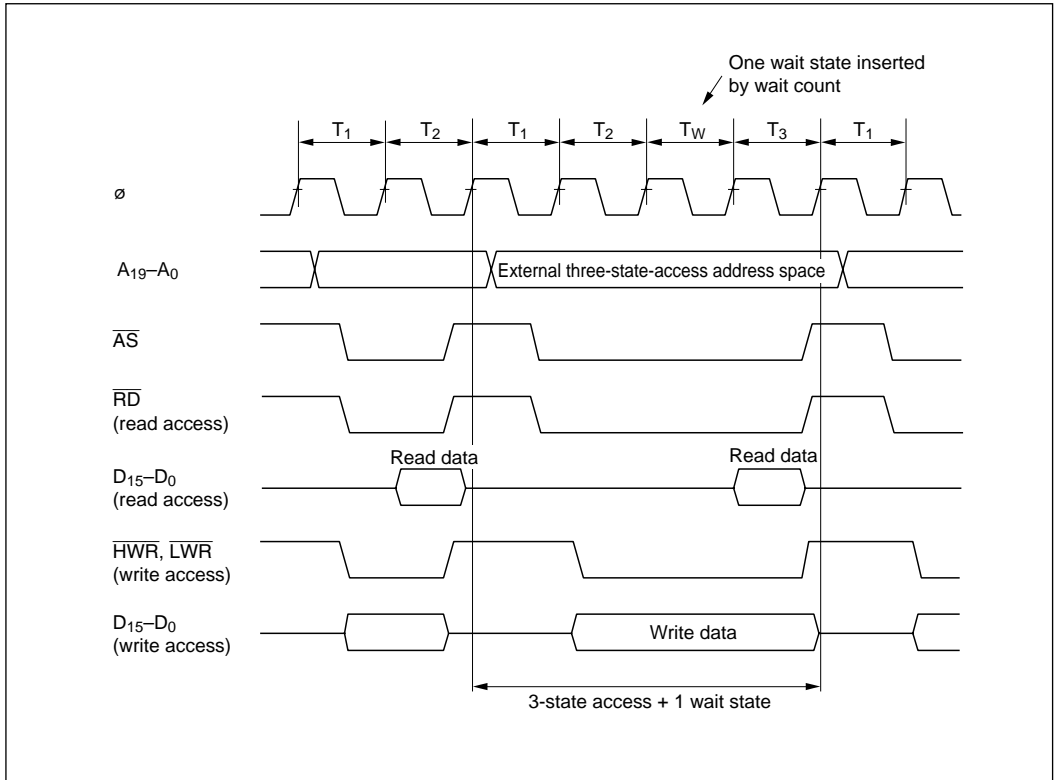
Mode	Description		
	$\overline{\text{WAIT}}$ Pin Function	Insertion Conditions	Number of Wait States Inserted
Programmable wait mode WMS1 = 0 WMS0 = 0	Disabled	Inserted in access to external three-state-access address space	1 to 3 states are inserted as specified by bits WC0 and WC1
Pin wait mode WMS1 = 1 WMS0 = 0	Enabled	Inserted in access to external three-state-access address space	<ul style="list-style-type: none"> <li>0 to 3 states are inserted as specified by bits WC0 and WC1</li> <li>Additional states can be inserted by driving the <math>\overline{\text{WAIT}}</math> signal low</li> </ul>
Pin auto-wait mode WMS1 = 1 WMS0 = 1	Enabled	Inserted in access to external three-state-access address space if $\overline{\text{WAIT}}$ is low	1 to 3 states are inserted as specified by bits WC0 and WC1

### 8.3.1 Programmable Wait Mode

Programmable wait mode is selected when  $WMS1 = 0$  and  $WMS0 = 0$ .

Whenever the CPU or DTC accesses the external three-state-access address space, the number of wait states selected by bits  $WC1$  and  $WC0$  are inserted. The  $PA_4/\overline{WAIT}$  pin is not used for wait control; it is available for general-purpose input or output.

Figure 8-2 shows the timing of operation in this mode when the wait count is 1 ( $WC1 = 0$ ,  $WC0 = 1$ ).



**Figure 8-2 Programmable Wait Mode**  
**(Example of External 16-Bit-Bus, Three-State-Access Address Space)**

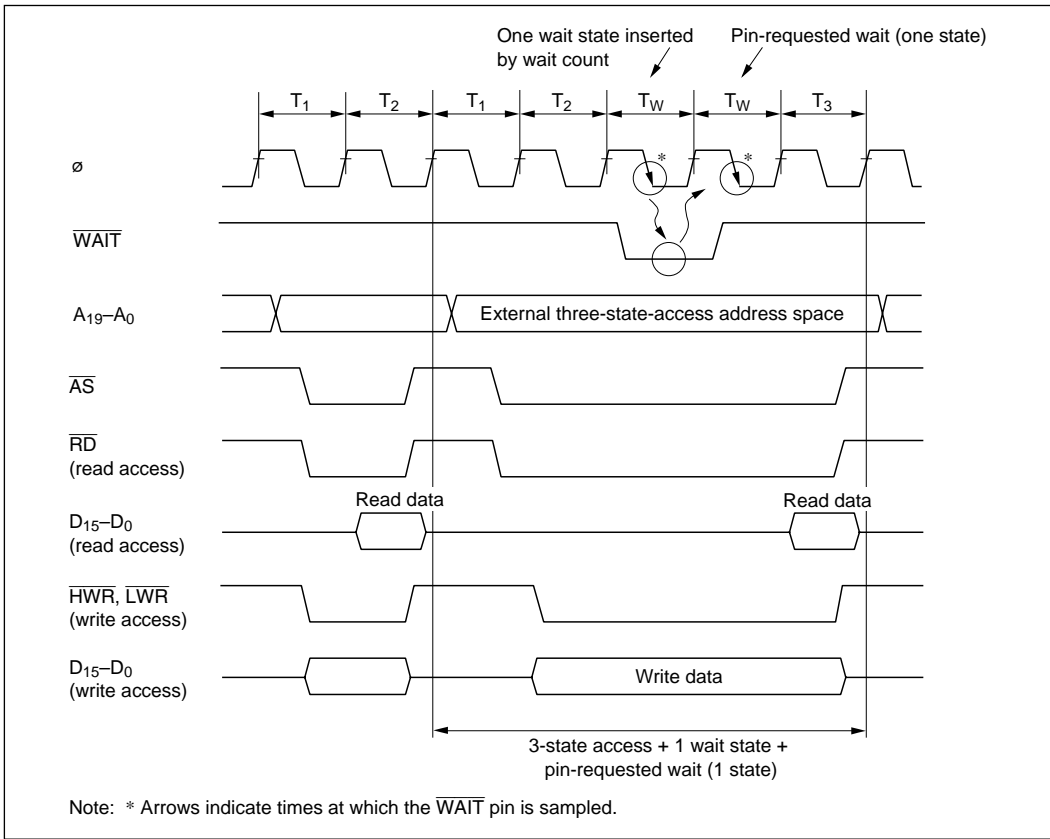
### 8.3.2 Pin Wait Mode

Pin wait mode is selected when  $WMS1 = 1$  and  $WMS0 = 0$ . In this mode the  $\overline{WAIT}$  function of the  $PA_4/\overline{WAIT}$  pin is used automatically.

The number of wait states indicated by wait count bits  $WC1$  and  $WC0$  are inserted into any bus cycle in which the CPU or DTC accesses the external three-state-access address space. In addition, wait states are inserted if the  $\overline{WAIT}$  signal is driven low, even if the wait count is 0. Wait states continue to be inserted until the  $\overline{WAIT}$  signal goes high.

This mode is useful for inserting four or more wait states, or when different external devices require different numbers of wait states.

Figure 8-3 shows the timing of operation in this mode when the wait count is 1 ( $WC1 = 0$ ,  $WC0 = 1$ ) and the  $\overline{WAIT}$  signal is held low to insert one additional wait state.



**Figure 8-3 Pin Wait Mode**  
(Example of External 16-Bit-Bus, Three-State-Access Address Space)

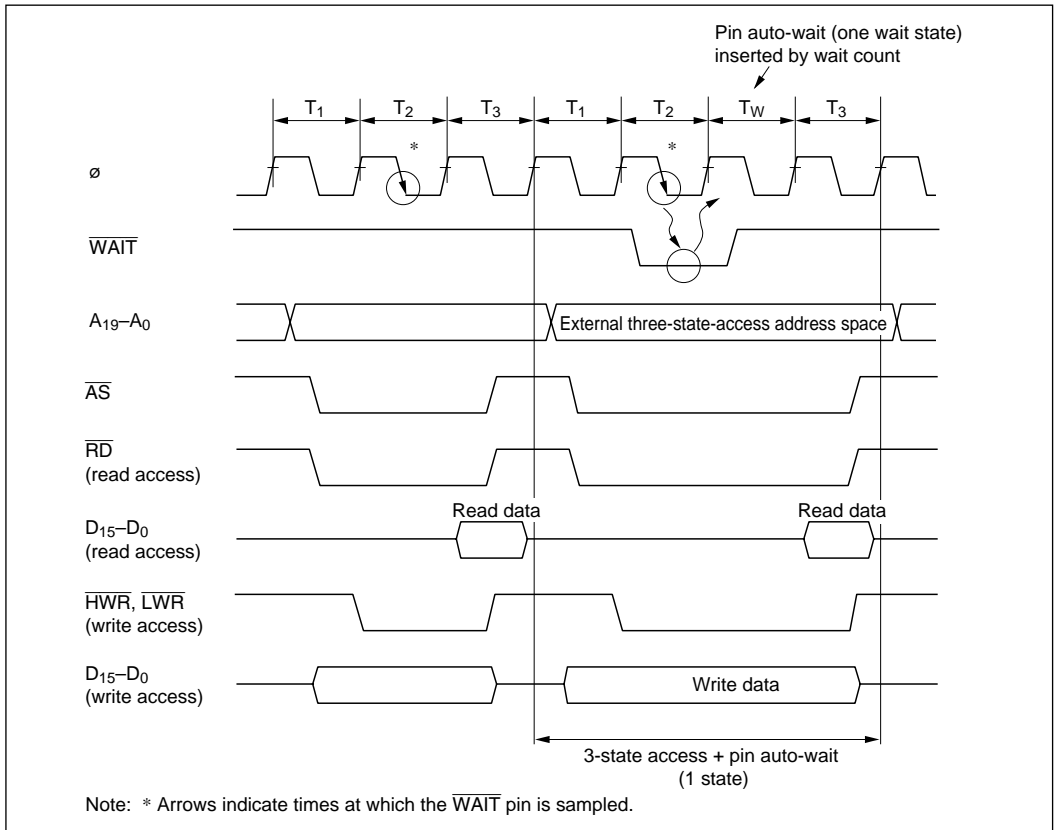
### 8.3.3 Pin Auto-Wait Mode

Pin auto-wait mode is selected when  $WMS1 = 1$  and  $WMS0 = 1$ . In this mode the  $\overline{WAIT}$  function of the  $PA_4/\overline{WAIT}$  pin is used automatically. When the CPU or DTC accesses the external three-state-access address space, if the  $\overline{WAIT}$  pin is low the number of wait states indicated by bits  $WC1$  and  $WC0$  are inserted.

This mode offers a simple way to interface a low-speed device: wait states can be inserted by routing the address strobe signal ( $\overline{AS}$ ) and a decoded address signal to the  $\overline{WAIT}$  pin.

Figure 8-4 shows the timing of operation in this mode when the wait count is 1 ( $WC1 = 0, WC0 = 1$ ).

In pin auto-wait mode the  $\overline{WAIT}$  pin is sampled only once, on the falling edge of the system clock ( $\phi$ ) in the  $T_2$  state. If the  $\overline{WAIT}$  signal is low at this time, the wait-state controller inserts the number of wait states indicated by bits  $WC1$  and  $WC0$ . The  $\overline{WAIT}$  pin is not sampled during the  $T_W$  and  $T_3$  states, so no additional wait states are inserted even if the  $\overline{WAIT}$  signal continues to be held low.



**Figure 8-4 Pin Auto-Wait Mode**  
(Example of External 16-Bit-Bus, Three-State-Access Address Space)



# Section 9 Clock Pulse Generator

## 9.1 Overview

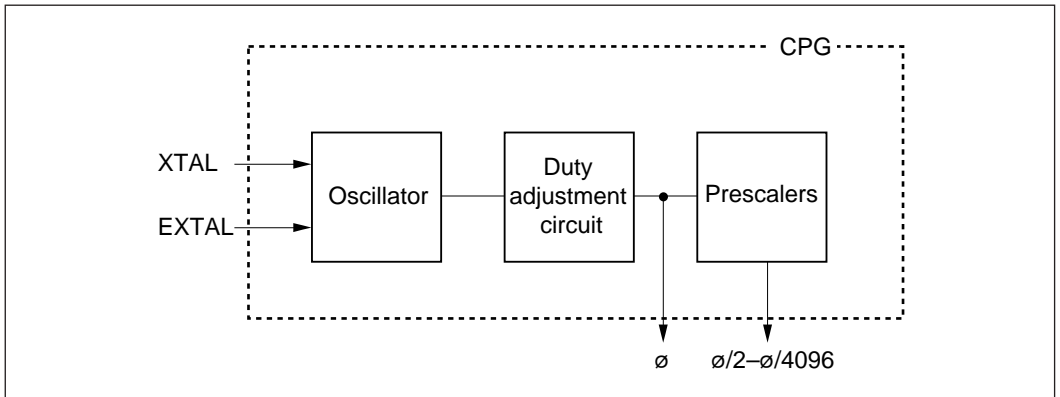
This LSI has an on-chip clock pulse generator (CPG). The clock pulse generator consists of an oscillator, system clock frequency divider, and frequency dividers (prescalers) for the clock signals of the on-chip supporting modules.

The H8/539F has an on-chip 1:1 clock pulse generator.

The 1:1 clock pulse generator does not include a frequency divider, but includes a circuit for adjusting the duty of the input clock.

### 9.1.1 Block Diagram

Figure 9-1 shows the configuration of the 1:1 clock pulse generator.



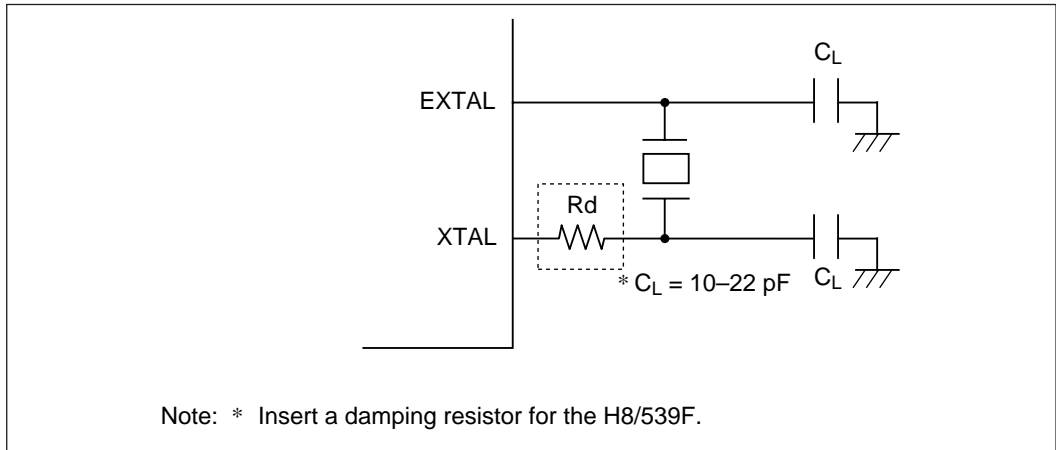
**Figure 9-1 Block Diagram of 1:1 Clock Pulse Generator**

## 9.2 Oscillator Circuit

Clock pulses can be generated by connecting a crystal resonator to the clock oscillator circuit, or by supplying an external clock signal. These two methods are described next.

### 9.2.1 Connecting a Crystal Resonator

(1) **Circuit Configuration:** A crystal resonator can be connected as in the example in figure 9-2. An AT-cut parallel resonating crystal should be used. For the 1:1 clock pulse generator, insert a damping resistor as listed in table 9-1.

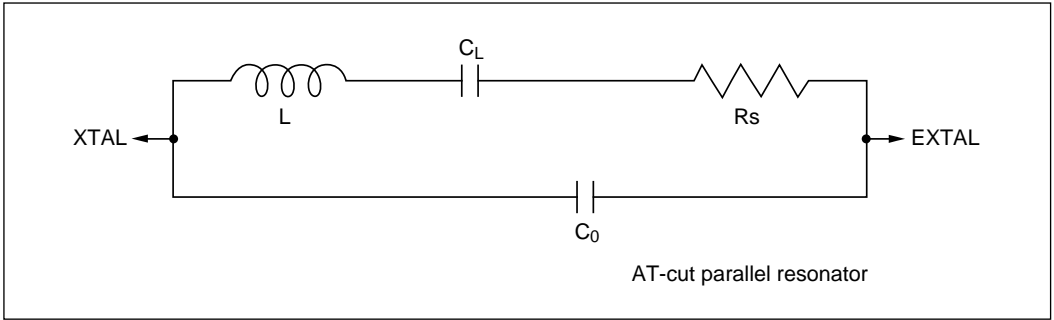


**Figure 9-2 Example of Crystal Resonator Connection**

**Table 9-1 Damping Resistance (Examples)**

Frequency (MHz)	2	4	8	10	12	16
Rd max ( $\Omega$ )	1 k	500	200	0	0	0

(2) **Crystal Resonator:** Figure 9-3 shows an equivalent circuit of the crystal resonator. The crystal resonator should have the characteristics listed in table 9-2. Use a crystal resonator with a frequency equal to the system clock frequency ( $\phi$ ).



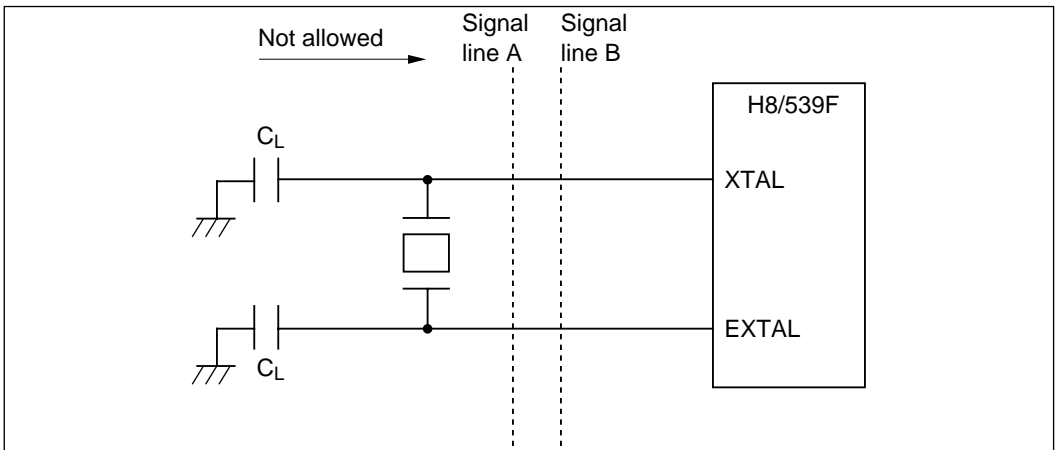
**Figure 9-3 Crystal Resonator Equivalent Circuit**

**Table 9-2 Crystal Resonator Parameters**

Frequency (MHz)	2	4	8	10	12	16
Rs max ( $\Omega$ )	500	120	80	70	60	50
C <sub>0</sub> max (pF)	7					

**(3) Notes on Board Design:** When a crystal resonator is connected, the following points should be noted:

- Other signal lines should be routed away from the oscillator circuit to prevent induction from interfering with correct oscillation. See figure 9-4.
- When the board is designed, the crystal resonator and its load capacitors should be placed as close as possible to the XTAL and EXTAL pins.

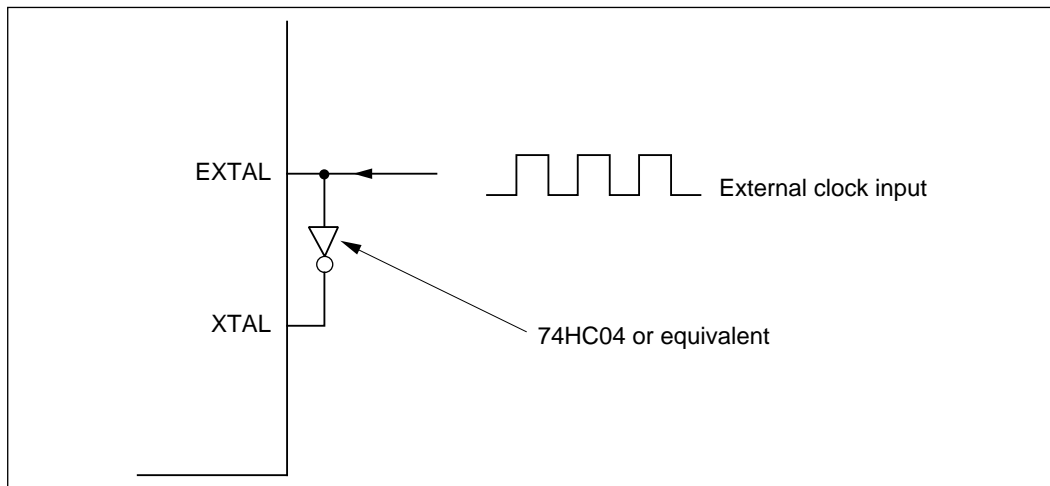


**Figure 9-4 Example of Incorrect Board Design**

## 9.2.2 External Clock Input

(1) **Circuit Configuration:** An external clock signal can be input at the EXTAL pin as shown in the example in figure 9-5. A reverse-phase clock should be input at the XTAL pin.

When the circuit configuration in figure 9-5 is used, the external clock should be held high in standby mode.



**Figure 9-5 Example of External Clock Input**

Note: The H8/539F can be driven with the XTAL pin left open if the stray capacitance at the XTAL pin does not exceed 10 pF and the clock input can be held high in standby mode.

### (2) External Clock

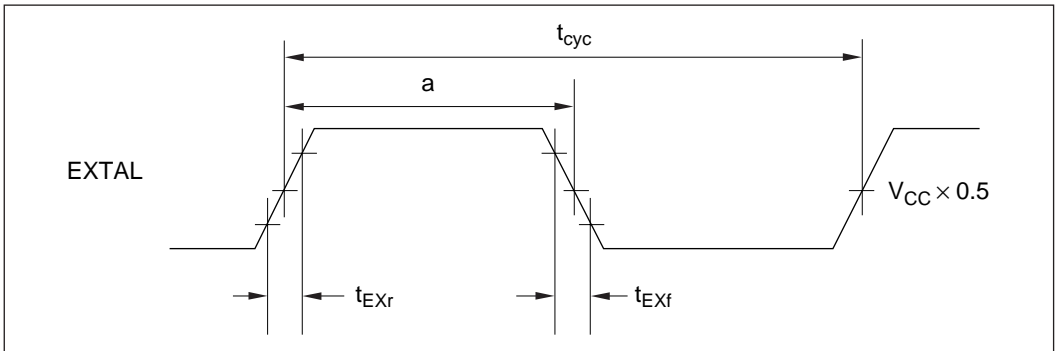
Table 9-3 and figure 9-6 indicate the required clock timing.

The external clock output settling delay time is shown in table 22-5 in section 22.2.2, “AC Characteristics” and in table 23-5 in section 23.2.2 “AC Characteristics”. The external clock output settling delay timing is shown in figure 22-2 in section 22.3.3, “Clock Timing”, and in table 23-11 in section 23.3.3, “Clock Timing”.

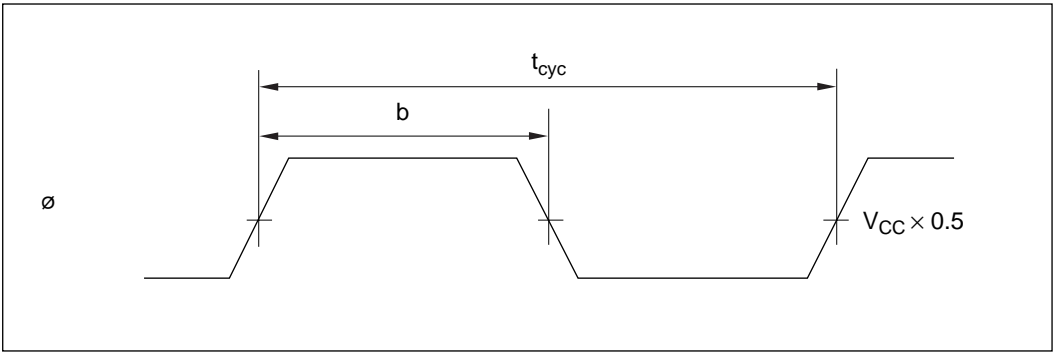
When the specified clock is input at the EXTAL pin, internal clock signal output settles after the elapse of the external clock output settling delay time ( $t_{DEXT}$ ). As the clock signal output remains unsettled during the  $t_{DEXT}$  period, the reset signal should be driven low to retain the reset state.

**Table 9-3 Clock Timing**

Item	Symbol	$V_{CC} = 5.0\text{ V} \pm 10\%$		Unit	Test Conditions	
		Min	Max			
External clock rise time	$t_{EXr}$	—	5	ns		Figure 9-6
External clock fall time	$t_{EXf}$	—	5	ns		
External clock input duty (a/ $t_{cyc}$ )	—	30	70	%	$\phi \geq 5\text{ MHz}$	Figure 9-6
		40	60	%	$\phi < 5\text{ MHz}$	
Clock duty cycle (b/ $t_{cyc}$ )	—	40	60	%		Figure 9-7



**Figure 9-6 External Clock Input Timing**



**Figure 9-7  $\phi$  Clock Output Timing**

### 9.3 Duty Adjustment Circuit

When the external clock frequency is 5 MHz or higher, the duty adjustment circuit adjusts the duty cycle to create the system clock ( $\phi$ ).

# Section 10 I/O Ports

## 10.1 Overview

The H8/539F has twelve I/O ports. Ports 1, 2, 4, 5, 7, B, and C are eight-bit input/output ports. Port 3 is a six-bit input/output port. Port 6 is a five-bit input/output port. Port A is a seven-bit input/output port. Port 8 is a four-bit input port. Port 9 is an eight-bit input port.

These ports are multiplexed with inputs and outputs of the on-chip supporting modules. The functions of ports 1, 2, A, B, and C also differ depending on the operating mode.

Each port has a data direction register (DDR) for selecting input or output, and a data register (DR) for holding output data. In addition to DR and DDR, port A has a bus release control register (BRCR), and ports B and C have MOS input pull-up transistor control registers (PBPCR and PCPCR).

Ports 1, 2, A, B, and C can drive one TTL load and a 90-pF capacitive load. Ports 3 to 7 can drive one TTL load and a 30-pF capacitive load. Ports 3 and 5 can drive LEDs (with 10-mA current sink). Ports 4 and 5 have Schmitt-trigger input circuits.

PWM output pin functions have been added to ports 6 and 7 of the H8/539F, and both serial communication input/output and PWM output pin functions have been added to port A.

Table 10-1 summarizes ports 1 to C of the H8/539F, giving the pin names and functions in each mode.

**Table 10-1 Ports 1 to C, Pin Names, and Functions in Each Mode**

Port	Description	Pins	Expanded Minimum		Expanded Maximum		Mode 7 (Single-Chip Mode)
			Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	
Port 1	8-bit input/output port	P1 <sub>7</sub> –P1 <sub>0</sub> / D <sub>15</sub> –D <sub>8</sub>	Data bus (D <sub>15</sub> to D <sub>8</sub> )				General-purpose input/output
Port 2	8-bit input/output port	P2 <sub>7</sub> –P2 <sub>0</sub> / D <sub>7</sub> –D <sub>0</sub>	Data bus (D <sub>7</sub> to D <sub>0</sub> )	General-purpose input/output	Data bus (D <sub>7</sub> to D <sub>0</sub> )	Data bus (D <sub>7</sub> to D <sub>0</sub> )	General-purpose input/output
Port 3	6-bit input/output port	P3 <sub>5</sub> –P3 <sub>0</sub> / T2OC <sub>2</sub> , T2OC <sub>1</sub> , T1OC <sub>4</sub> –T1OC <sub>1</sub>	Output (T2OC <sub>2/1</sub> , T1OC <sub>4/3/2/1</sub> ) from 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output				
Port 4	8-bit input/output port	P4 <sub>7</sub> /T7IOC <sub>2</sub> , P4 <sub>6</sub> /T7IOC <sub>1</sub> , P4 <sub>5</sub> /T6IOC <sub>2</sub> , P4 <sub>4</sub> /T6IOC <sub>1</sub> , P4 <sub>3</sub> /T5IOC <sub>2</sub> , P4 <sub>2</sub> /T5IOC <sub>1</sub> , P4 <sub>1</sub> /T4IOC <sub>2</sub> , P4 <sub>0</sub> /T4IOC <sub>1</sub>	Input and output (T7IOC <sub>2/1</sub> , T6IOC <sub>2/1</sub> , T5IOC <sub>2/1</sub> , T4IOC <sub>2/1</sub> ) for 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output				
Port 5	8-bit input/output port	P5 <sub>7</sub> –P5 <sub>0</sub> / T3IOC <sub>2</sub> , T3IOC <sub>1</sub> , T2IOC <sub>2</sub> , T2IOC <sub>1</sub> , T1IOC <sub>4</sub> –T1IOC <sub>1</sub>	Input and output (T3IOC <sub>2/1</sub> , T2IOC <sub>2/1</sub> , T1IOC <sub>4/3/2/1</sub> ) for 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output				
Port 6	5-bit input/output port	P6 <sub>4</sub> /TCLK <sub>3</sub> , P6 <sub>3</sub> /TCLK <sub>2</sub> , P6 <sub>2</sub> /TCLK <sub>1</sub> , P6 <sub>1</sub> /IRQ <sub>3</sub> , P6 <sub>0</sub> /IRQ <sub>2</sub> /PW <sub>3</sub>	Clock input (TCLK <sub>3/2/1</sub> ) for 16-bit integrated-timer pulse unit (IPU), external interrupt input (IRQ <sub>3/2</sub> ), PWM timer output (PW <sub>3</sub> ), and general-purpose input/output				
Port 7	8-bit input/output port	P7 <sub>7</sub> /SCK <sub>2</sub> /PW <sub>2</sub> , P7 <sub>6</sub> /SCK <sub>1</sub> /PW <sub>1</sub> , P7 <sub>5</sub> /RXD <sub>2</sub> , P7 <sub>4</sub> /TXD <sub>2</sub> , P7 <sub>3</sub> /RXD <sub>1</sub> , P7 <sub>2</sub> /TXD <sub>1</sub> , P7 <sub>1</sub> /IRQ <sub>1</sub> / ADTRG <sub>1</sub> , P7 <sub>0</sub> /IRQ <sub>0</sub>	Input and output (SCK <sub>2/1</sub> , TXD <sub>2/1</sub> , RXD <sub>2/1</sub> ) for serial communication interfaces 1 and 2 (SCI1/2), external interrupt input (IRQ <sub>1/0</sub> ), A/D converter trigger input (ADTRG <sub>1</sub> ), PWM timer output (PW <sub>2/1</sub> ), and general-purpose input/output				
Port 8	4-bit input port	P8 <sub>3</sub> –P8 <sub>0</sub> / AN <sub>11</sub> –AN <sub>8</sub>	Analog input for A/D converter (AN <sub>11</sub> to AN <sub>8</sub> ) and general-purpose input				
Port 9	8-bit input port	P9 <sub>7</sub> –P9 <sub>0</sub> / AN <sub>7</sub> –AN <sub>0</sub>	Analog input for A/D converter (AN <sub>7</sub> to AN <sub>0</sub> ) and general-purpose input				



**Table 10-1 Ports 1 to C, Pin Names, and Functions in Each Mode (cont)**

Port	Description	Pins	Expanded Minimum		Expanded Maximum		Mode 7 (Single-Chip Mode)
			Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	
Port A	7-bit input/output port	PA <sub>6</sub> /T3OC <sub>2</sub> / BACK/TXD <sub>3</sub> , PA <sub>5</sub> /T3OC <sub>1</sub> / BREQ/RXD <sub>3</sub> , PA <sub>4</sub> /WAIT	Output from 16-bit integrated-timer pulse unit (IPU), input and output (TXD <sub>3</sub> , RXD <sub>3</sub> ) for serial communication interface 3 (SCI3), general-purpose input/output, and BACK, BREQ, and WAIT input and output if enabled by settings in bus release control register (BRCR), wait control register (WCR), and port A control register (PACR)				16-bit integrated-timer pulse unit (IPU) output, serial communication interface 3 (SCI3) input and output (TXD <sub>3</sub> , RXD <sub>3</sub> ), and general-purpose input/output (PA <sub>4</sub> : general-purpose input/output only)
		PA <sub>3</sub> /A <sub>19</sub> / T5OC <sub>2</sub> /SCK <sub>3</sub> , PA <sub>2</sub> /A <sub>18</sub> / T5OC <sub>1</sub> /PW <sub>3</sub> , PA <sub>1</sub> /A <sub>17</sub> / T4OC <sub>2</sub> /PW <sub>2</sub> , PA <sub>0</sub> /A <sub>16</sub> / T4OC <sub>1</sub> /PW <sub>1</sub>	Output (T5OC <sub>2/1</sub> , T4OC <sub>2/1</sub> ) from 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output	Page address output (A <sub>19</sub> to A <sub>16</sub> )	Page address output (A <sub>19</sub> to A <sub>16</sub> ), serial communication interface 3 (SCI3) input/output (SCK <sub>3</sub> ), output (PW <sub>1/2/3</sub> ) from PWM timers (PW <sub>1/2/3</sub> ), and general-purpose input/output	Page address output (A <sub>19</sub> to A <sub>16</sub> ), serial communication interface 3 (SCI3) input/output (SCK <sub>3</sub> ), output (PW <sub>1/2/3</sub> ) from PWM timers (PW <sub>1/2/3</sub> ), and general-purpose input/output	Page address output (A <sub>19</sub> to A <sub>16</sub> ), serial communication interface 3 (SCI3) input/output (SCK <sub>3</sub> ), output (PW <sub>1/2/3</sub> ) from PWM timers (PW <sub>1/2/3</sub> ), and general-purpose input/output
Port B	8-bit input/output port	PB <sub>7</sub> –PB <sub>0</sub> / A <sub>15</sub> –A <sub>8</sub>	Address output (A <sub>15</sub> to A <sub>0</sub> )	Address output (A <sub>15</sub> to A <sub>0</sub> ) when DDR = 1,	Address output (A <sub>15</sub> to A <sub>0</sub> )	Address output (A <sub>15</sub> to A <sub>0</sub> ) when DDR = 1,	General-purpose input/output
Port C	8-bit input/output port	PC <sub>7</sub> –PC <sub>0</sub> / A <sub>7</sub> –A <sub>0</sub>		general-purpose input when DDR = 0		general-purpose input when DDR = 0	

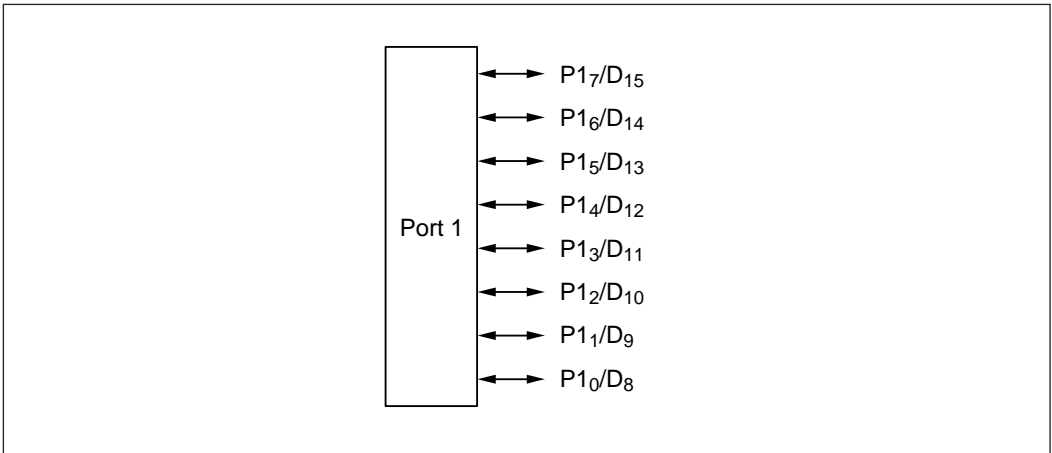
## 10.2 Port 1

### 10.2.1 Overview

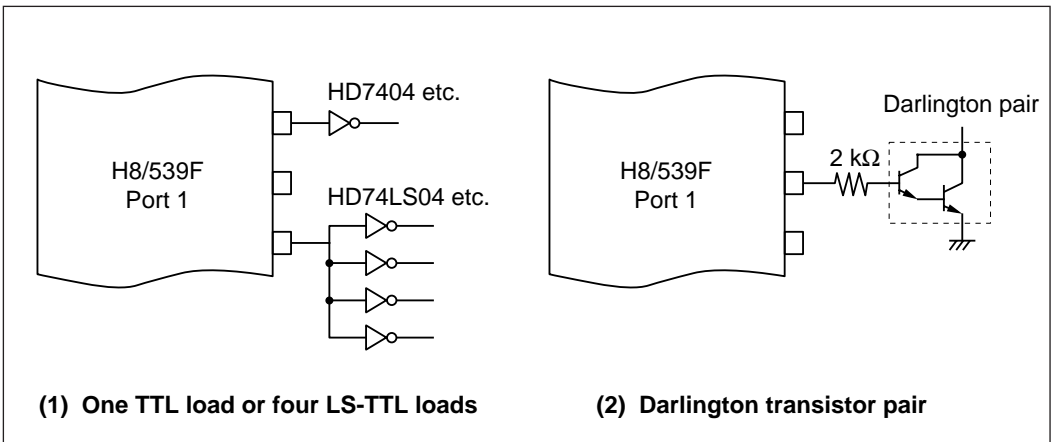
Port 1 is an eight-bit general-purpose input/output port in mode 7. In modes 1 to 6, port 1 is a data bus ( $D_{15}$  to  $D_8$ ).

Pins in port 1 can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair.

Figure 10-1 summarizes the pin functions. Figure 10-2 shows examples of output loads for port 1.



**Figure 10-1 Port 1 Pin Functions**



**Figure 10-2 Examples of Port 1 Output Loads**

## 10.2.2 Register Descriptions

Table 10-2 summarizes the registers of port 1.

**Table 10-2 Port 1 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE80	Port 1 data direction register	P1DDR	W	H'00
H'FE82	Port 1 data register	P1DR	R/W	H'00

**(1) Port 1 Data Direction Register:** The port 1 data direction register (P1DDR) is an eight-bit register. Each bit selects input or output for one pin in port 1. These input/output designations are valid only in mode 7.

Bit	7	6	5	4	3	2	1	0
	P <sub>7</sub> DDR	P <sub>6</sub> DDR	P <sub>5</sub> DDR	P <sub>4</sub> DDR	P <sub>3</sub> DDR	P <sub>2</sub> DDR	P <sub>1</sub> DDR	P <sub>0</sub> DDR
Initial value	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W

A pin in port 1 becomes an output pin if the corresponding P1DDR bit is set to 1, and an input pin if this bit is cleared to 0. P1DDR is a write-only register. All bits always return the value 1 when read.

P1DDR is initialized to H'00 by a reset and in hardware standby mode. P1DDR is not initialized in software standby mode.

**(2) Port 1 Data Register:** The port 1 data register (P1DR) is an eight-bit register that stores data for pins P<sub>0</sub> to P<sub>7</sub>. P1DR is used only in mode 7. In modes 1 to 6, the bit values in P1DR cannot be modified and always read 1.

Bit	7	6	5	4	3	2	1	0
	P <sub>7</sub>	P <sub>6</sub>	P <sub>5</sub>	P <sub>4</sub>	P <sub>3</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>0</sub>
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in P1DDR is set to 1, the corresponding P1DR bit value is output at the corresponding pin. If port 1 is read the value in P1DR is returned, regardless of the actual state of the pin.

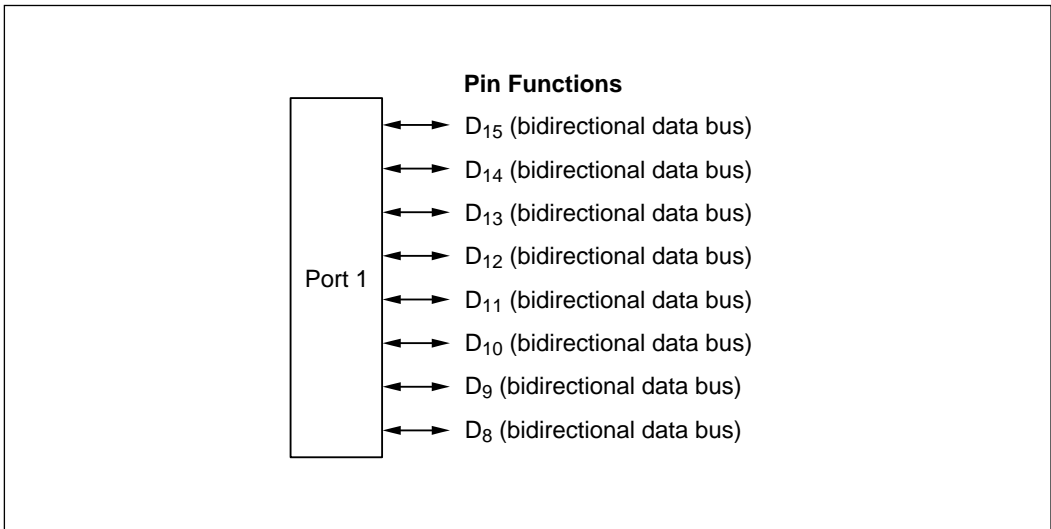
When a bit in P1DDR is cleared to 0, it is possible to write to the corresponding P1DR bit but the value is not output at the pin. If P1DR is read the value at the pin is returned, regardless of the value written in P1DR.

P1DR is initialized to H'00 by a reset and in hardware standby mode. P1DR is not initialized in software standby mode.

### 10.2.3 Pin Functions in Each Mode

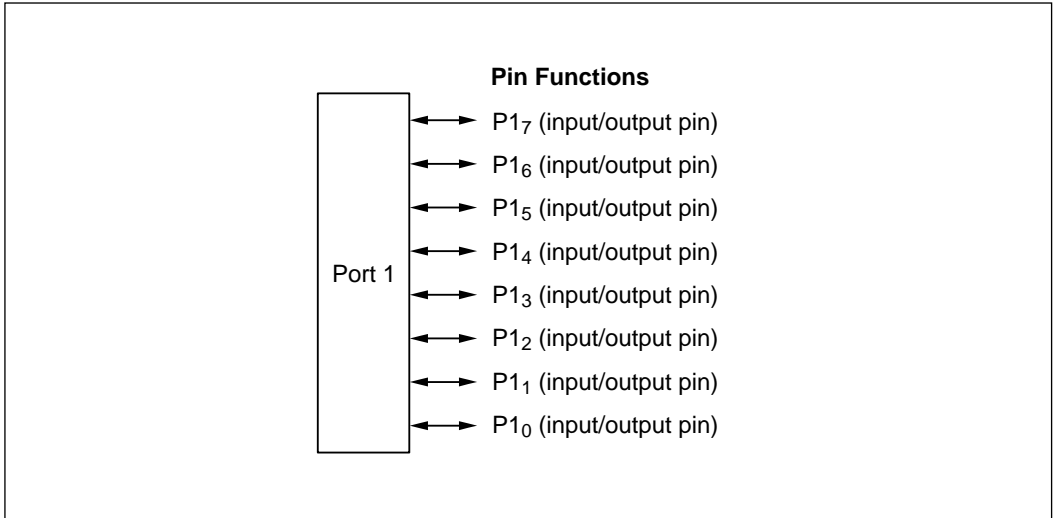
The functions of port 1 differ between the externally expanded modes (modes 1 to 6) and single-chip mode (mode 7). The pin functions in each mode are described below.

**(1) Pin Functions in Externally Expanded Modes (Modes 1 to 6):** The settings in P1DDR are ignored. Port 1 automatically becomes a bidirectional data bus. Figure 10-3 shows the pin functions in modes 1 to 6.



**Figure 10-3 Pin Functions in Modes 1 to 6**

**(2) Pin Functions in Single-Chip Mode (Mode 7):** Port 1 consists of general-purpose input/output pins. Input or output can be selected separately for each pin. A pin becomes an output pin if the corresponding P1DDR bit is set to 1 and an input pin if this bit is cleared to 0. Figure 10-4 shows the pin functions in mode 7.



**Figure 10-4 Pin Functions in Mode 7**

**(3) Software Standby Mode:** Transition to software standby does not change the pin functions in single-chip mode. In the externally expanded modes, port 1 is in the high-impedance state during software standby.

## 10.2.4 Port 1 Read/Write Operations

P1DR and P1DDR have different read/write functions depending on whether port 1 is used as a data bus ( $D_{15}$  to  $D_8$ ) or for general-purpose input or output ( $P1_7$  to  $P1_0$ ). The operating states and functions of port 1 are described next.

**(1) Data Bus (Modes 1 to 6):** Figure 10-5 shows a block diagram illustrating the data-bus function. Table 10-3 indicates register read/write data. When port 1 operates as a data bus, the values in the port 1 data register (P1DR) have no effect on the bus lines. When read, P1DR returns all 1s.

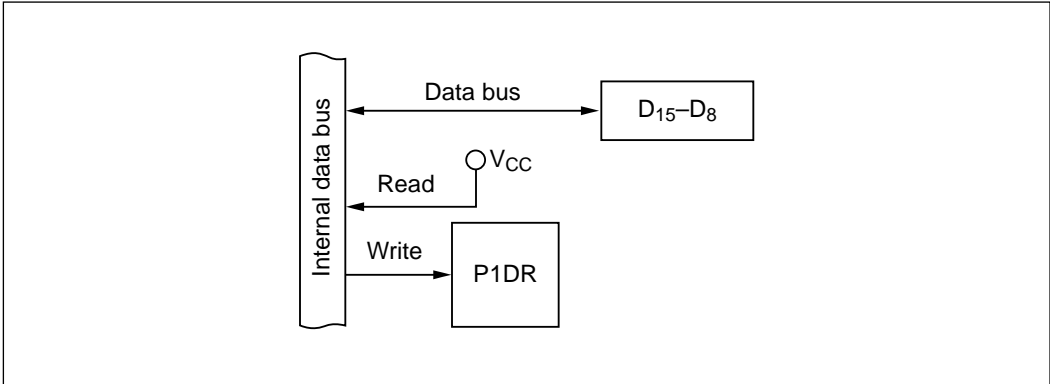


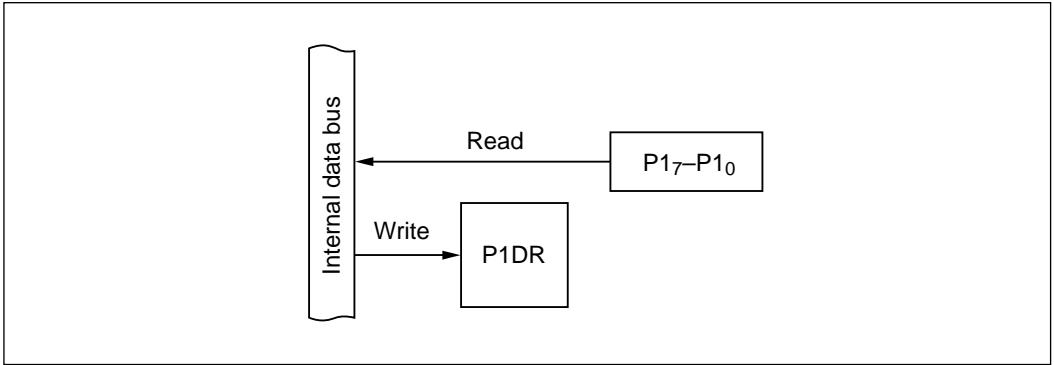
Figure 10-5 Data Bus:  $D_{15}$  to  $D_8$  (Modes 1 to 6)

Table 10-3 Register Read/Write Data

	Read	Write
P1DR	Always 1	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(2) Input Port (Mode 7):** Figure 10-6 shows a block diagram illustrating the general-purpose input function. Table 10-4 indicates register read/write data. When port 1 operates as an input pin, values written in the port 1 data register (P1DR) have no effect on general-purpose input lines. When read, P1DR returns the value at the pin.



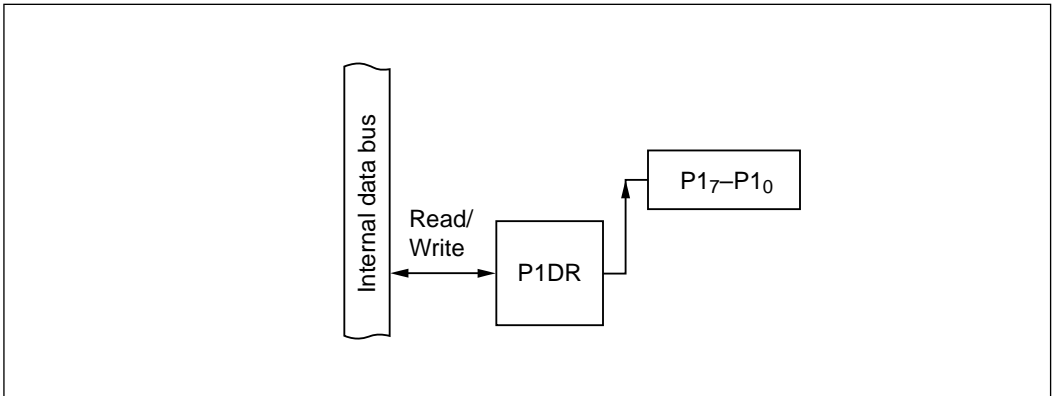
**Figure 10-6 Input Port (Mode 7)**

**Table 10-4 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
P1DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(3) Output Port (Mode 7):** Figure 10-7 shows a block diagram illustrating the general-purpose output function. Table 10-5 indicates register read/write data. The value written in the port 1 data register (P1DR) is output at the pin. When read, P1DR returns the value written in P1DR.



**Figure 10-7 Output Port (Mode 7)**

**Table 10-5 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
P1DR	P1DR value	Value output at pin

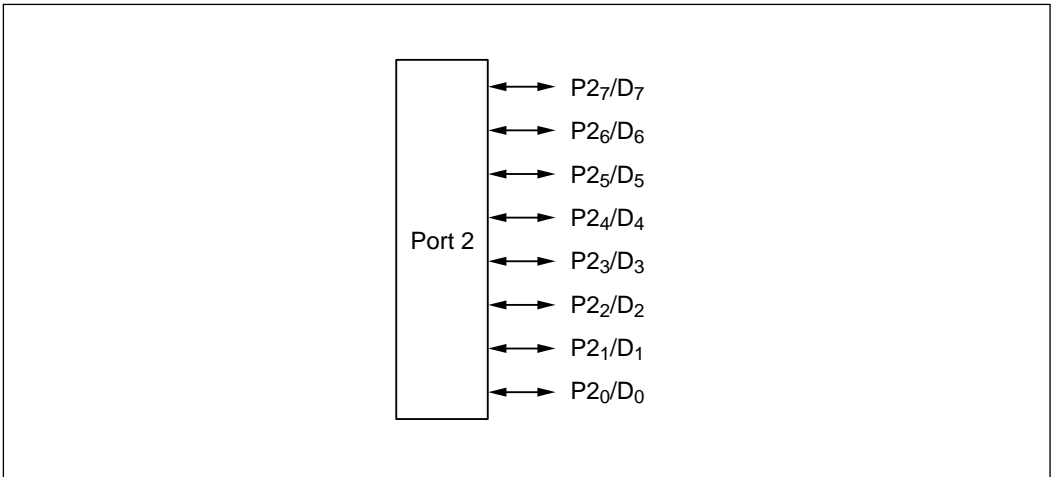
## 10.3 Port 2

### 10.3.1 Overview

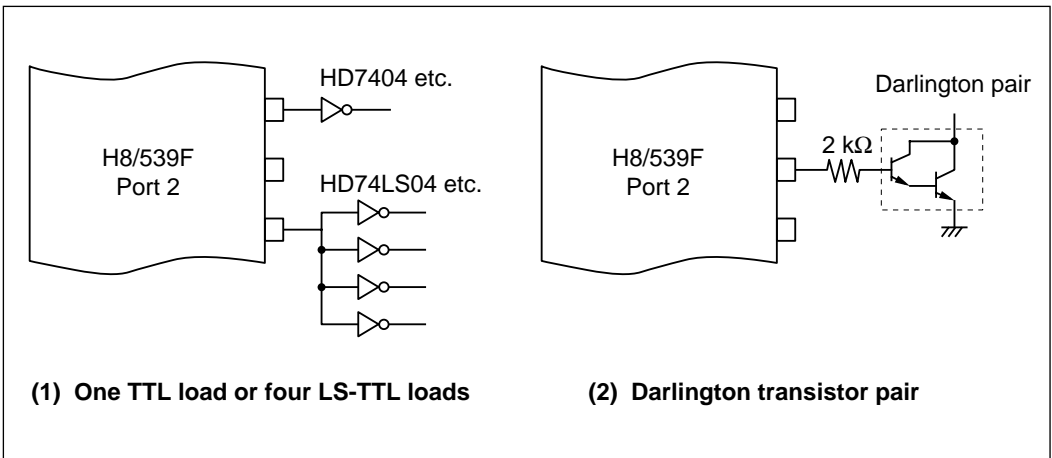
Port 2 is an eight-bit general-purpose input/output port in modes 2 and 7. In modes 1, 3, 4, 5, and 6, port 2 is a data bus (D<sub>7</sub> to D<sub>0</sub>).

Pins in port 2 can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair.

Figure 10-8 summarizes the pin functions. Figure 10-9 shows examples of output loads for port 2.



**Figure 10-8 Port 2 Pin Functions**



**Figure 10-9 Examples of Port 2 Output Loads**



### 10.3.2 Register Descriptions

Table 10-6 summarizes the registers of port 2.

**Table 10-6 Port 2 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE81	Port 2 data direction register	P2DDR	W	H'00
H'FE83	Port 2 data register	P2DR	R/W	H'00

**(1) Port 2 Data Direction Register:** The port 2 data direction register (P2DDR) is an eight-bit register. Each bit selects input or output for one pin in port 2. These input/output designations are valid only in modes 2 and 7.

Bit	7	6	5	4	3	2	1	0
	P2 <sub>7</sub> DDR	P2 <sub>6</sub> DDR	P2 <sub>5</sub> DDR	P2 <sub>4</sub> DDR	P2 <sub>3</sub> DDR	P2 <sub>2</sub> DDR	P2 <sub>1</sub> DDR	P2 <sub>0</sub> DDR
Initial value	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W

A pin in port 2 becomes an output pin if the corresponding P2DDR bit is set to 1, and an input pin if this bit is cleared to 0. P2DDR is a write-only register. All bits always return the value 1 when read.

P2DDR is initialized to H'00 by a reset and in hardware standby mode. P2DDR is not initialized in software standby mode.

**(2) Port 2 Data Register:** The port 2 data register (P2DR) is an eight-bit register that stores data for pins P2<sub>7</sub> to P2<sub>0</sub>. P2DR is used only in modes 2 and 7. In modes 1, 3, 4, 5, and 6, the bit values in P2DR cannot be modified and always read 1.

Bit	7	6	5	4	3	2	1	0
	P2 <sub>7</sub>	P2 <sub>6</sub>	P2 <sub>5</sub>	P2 <sub>4</sub>	P2 <sub>3</sub>	P2 <sub>2</sub>	P2 <sub>1</sub>	P2 <sub>0</sub>
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in P2DDR is set to 1, the corresponding P2DR bit value is output at the corresponding pin. If port 2 is read the value in P2DR is returned, regardless of the actual state of the pin.

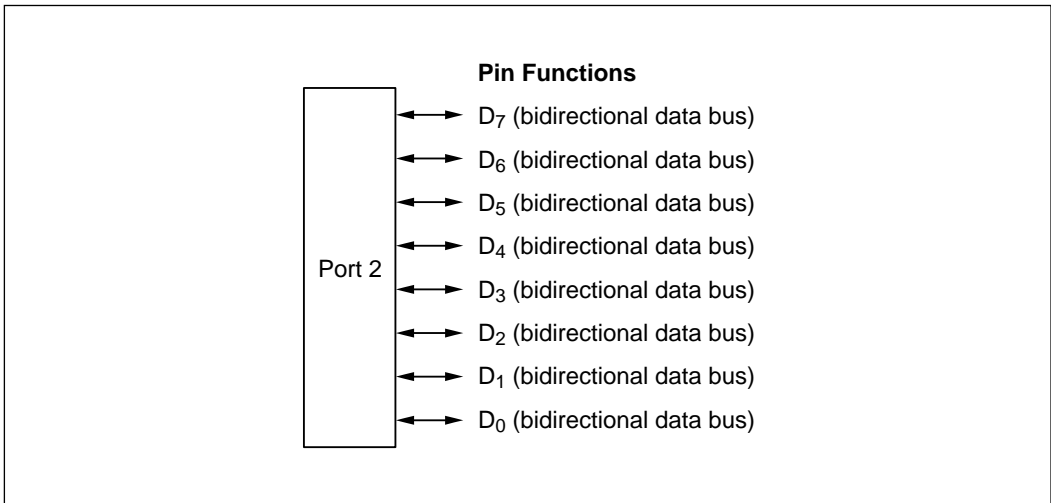
When a bit in P2DDR is cleared to 0, it is possible to write to the corresponding P2DR bit but the value is not output at the pin. If P2DR is read the value at the pin is returned, regardless of the value written in P2DR.

P2DR is initialized to H'00 by a reset and in hardware standby mode. P2DR is not initialized in software standby mode.

### 10.3.3 Pin Functions in Each Mode

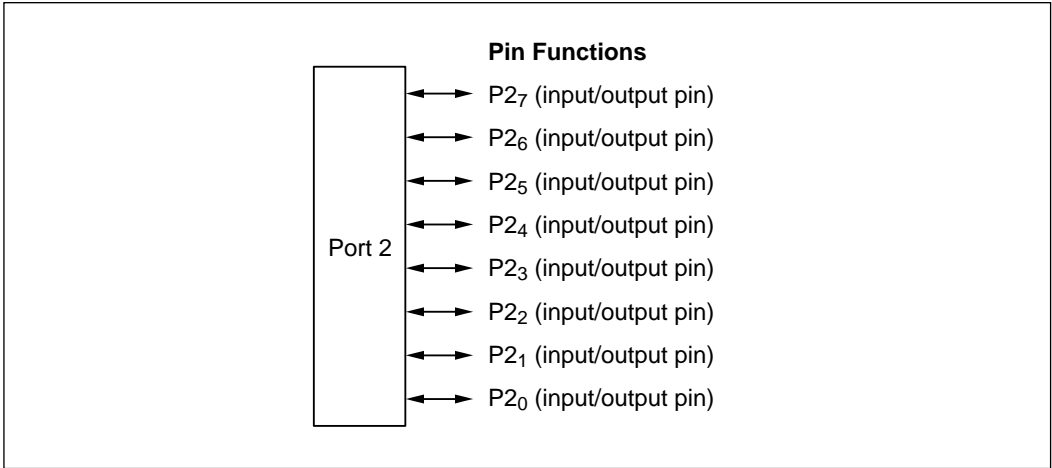
The functions of port 2 differ between modes 1, 3, 4, 5, and 6 on the one hand, and modes 2 and 7 on the other hand. The pin functions in each mode group are described below.

**(1) Pin Functions in Modes 1, 3, 4, 5, and 6:** The settings in P2DDR are ignored. Port 2 automatically becomes a bidirectional data bus. Figure 10-10 shows the pin functions in modes 1, 3, 4, 5, and 6.



**Figure 10-10 Pin Functions in Modes 1, 3, 4, 5, and 6**

**(2) Pin Functions in Modes 2 and 7:** Port 2 consists of general-purpose input/output pins. Input or output can be selected separately for each pin. A pin becomes an output pin if the corresponding P2DDR bit is set to 1 and an input pin if this bit is cleared to 0. Figure 10-11 shows the pin functions in modes 2 and 7.



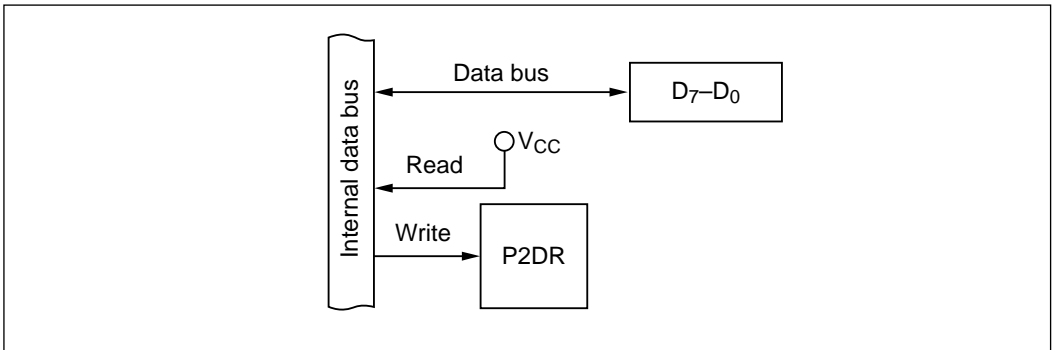
**Figure 10-11 Pin Functions in Modes 2 and 7**

**(3) Software Standby Mode:** Transition to software standby does not change the pin functions in modes 2 and 7. In the externally expanded modes, port 2 is in the high-impedance state during software standby.

### 10.3.4 Port 2 Read/Write Operations

P2DR and P2DDR have different read/write functions depending on whether port 2 is used as a data bus (D<sub>7</sub> to D<sub>0</sub>) or for general-purpose input or output (P2<sub>7</sub> to P2<sub>0</sub>). The operating states and functions of port 2 are described next.

**(1) Data Bus (Modes 1, 3, 4, 5, and 6):** Figure 10-12 shows a block diagram illustrating the data-bus function. Table 10-7 indicates register read/write data. When port 2 operates as a data bus, the values in the port 2 data register (P2DR) have no effect on the bus lines. When read, P2DR returns all 1s.



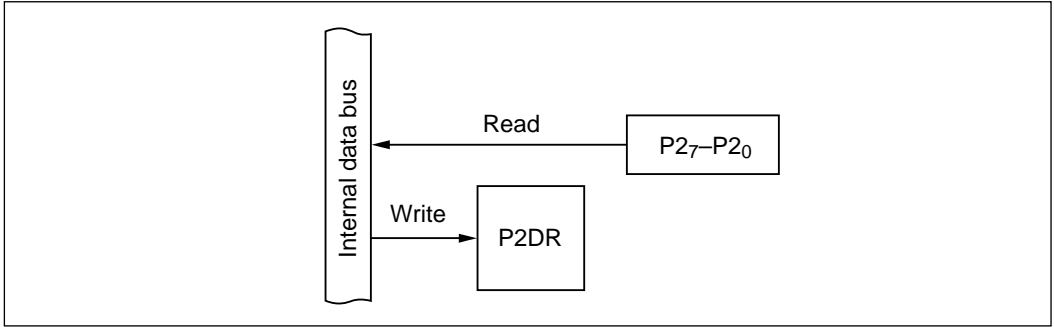
**Figure 10-12 Data Bus: D<sub>7</sub> to D<sub>0</sub> (Modes 1, 3, 4, 5, and 6)**

**Table 10-7 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
P2DR	Always 1	Don't care*

Note: The register can be written to, but the value is not output at the pins.

(2) **Input Port (Modes 2 and 7):** Figure 10-13 shows a block diagram illustrating the general-purpose input function. Table 10-8 indicates register read/write data. Values written in the port 2 data register (P2DR) have no effect on general-purpose input lines. When read, P2DR returns the value at the pin.



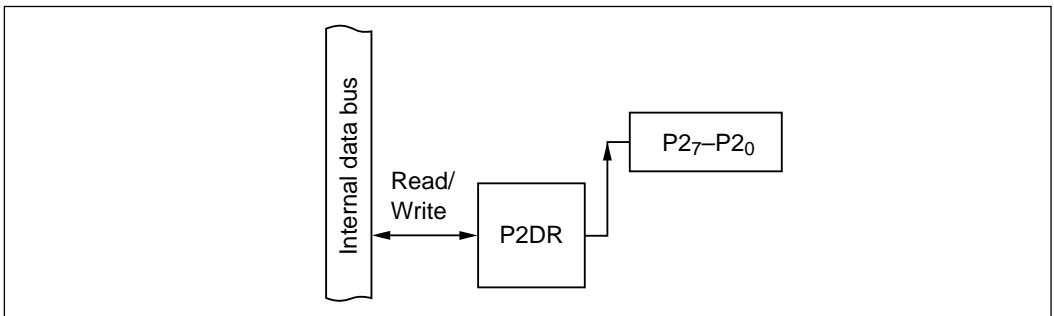
**Figure 10-13 Input Port (Modes 2 and 7)**

**Table 10-8 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
P2DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

(3) **Output Port (Modes 2 and 7):** Figure 10-14 shows a block diagram illustrating the general-purpose output function. Table 10-9 indicates register read/write data. The value written in the port 2 data register (P2DR) is output at the pin. When read, P2DR returns the value written in P2DR.



**Figure 10-14 Output Port (Modes 2 and 7)**

**Table 10-9 Register Read/Write Data**

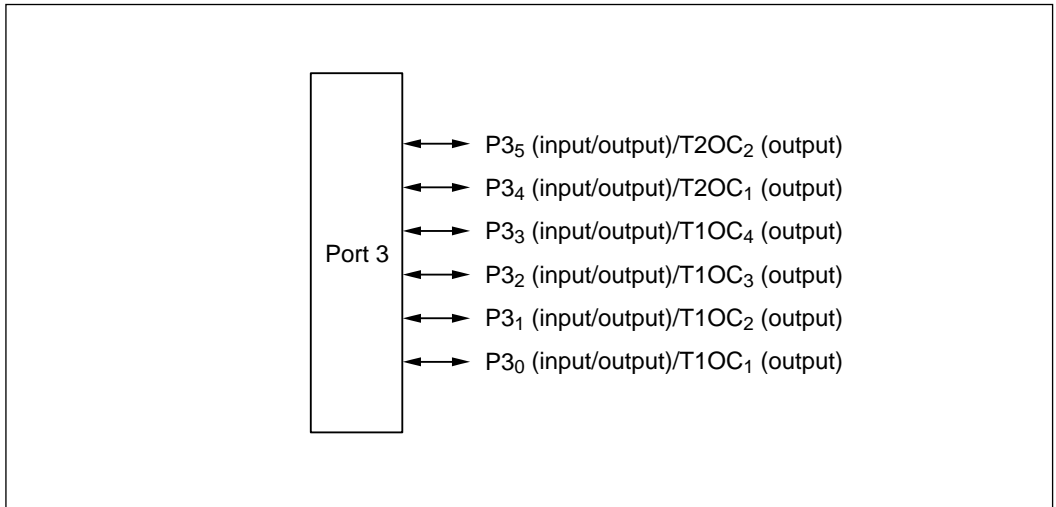
	<b>Read</b>	<b>Write</b>
P2DR	P2DR value	Value output at pin

## 10.4 Port 3

### 10.4.1 Overview

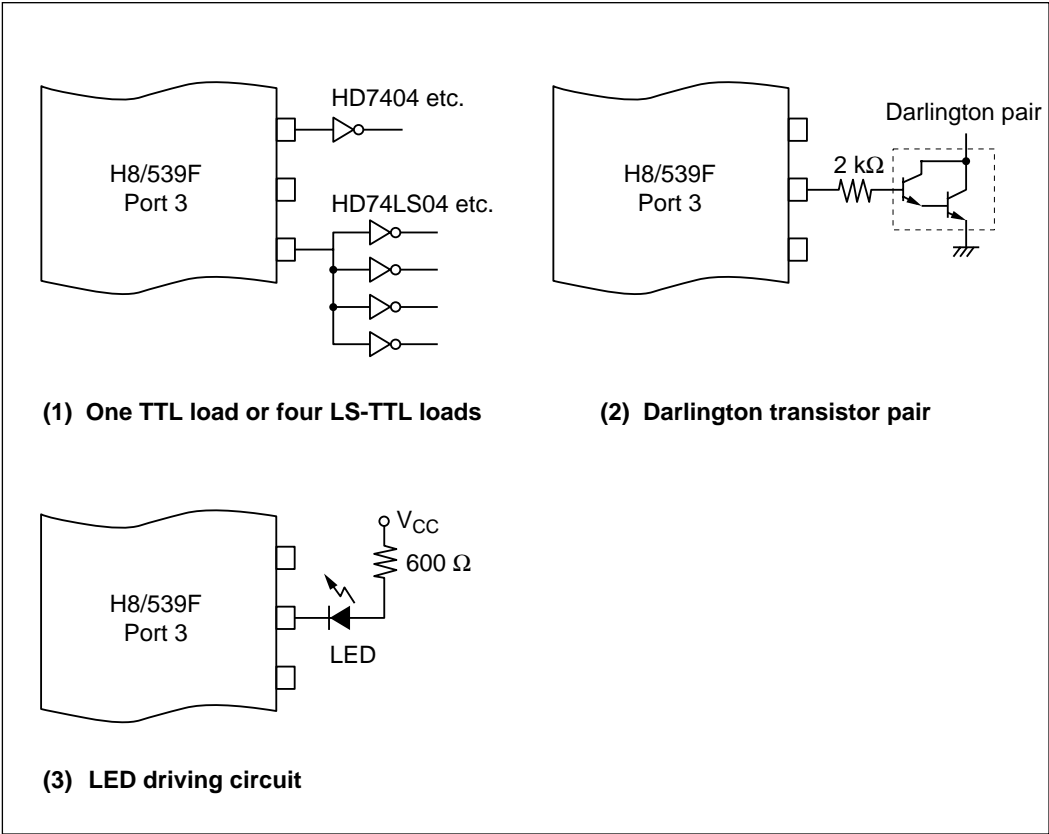
Port 3 is a six-bit input/output port that is multiplexed with output compare pins (T2OC<sub>2</sub>, T2OC<sub>1</sub>, T1OC<sub>4</sub> to T1OC<sub>1</sub>) of the 16-bit integrated-timer pulse unit (IPU). Figure 10-15 summarizes the pin functions.

Pins in port 3 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair or LED (with 10-mA current sink).



**Figure 10-15 Port 3 Pin Functions**

Figure 10-16 shows examples of output loads for port 3.



**Figure 10-16 Examples of Port 3 Output Loads**

### 10.4.2 Register Descriptions

Table 10-10 summarizes the registers of port 3.

**Table 10-10 Port 3 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE84	Port 3 data direction register	P3DDR	W	H'C0
H'FE86	Port 3 data register	P3DR	R/W	H'C0

**(1) Port 3 Data Direction Register:** The port 3 data direction register (P3DDR) is an eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	—	—	P3 <sub>5</sub> DDR	P3 <sub>4</sub> DDR	P3 <sub>3</sub> DDR	P3 <sub>2</sub> DDR	P3 <sub>1</sub> DDR	P3 <sub>0</sub> DDR
Initial value	1	1	0	0	0	0	0	0
R/W	—	—	W	W	W	W	W	W

A pin in port 3 becomes an output pin if the corresponding P3DDR bit is set to 1, and an input pin if this bit is cleared to 0. P3DDR is a write-only register. All bits always return the value 1 when read.

P3DDR is initialized to H'C0 by a reset and in hardware standby mode. P3DDR is not initialized in software standby mode.

**(2) Port 3 Data Register:** The port 3 data register (P3DR) is an eight-bit register that stores data for pins P3<sub>5</sub> to P3<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	—	—	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Initial value	1	1	0	0	0	0	0	0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in P3DDR is set to 1, the corresponding P3DR bit value is output at the corresponding pin. If port 3 is read the value in P3DR is returned, regardless of the actual state of the pin.

When a bit in P3DDR is cleared to 0, it is possible to write to the corresponding P3DR bit but the value is not output at the pin. If P3DR is read the value at the pin is returned, regardless of the value written in P3DR.

P3DR is initialized to H'C0 by a reset and in hardware standby mode. P3DR is not initialized in software standby mode.

### 10.4.3 Pin Functions in Each Mode

In all modes port 3 can be used for general-purpose input or output, or for the output compare function of the 16-bit integrated-timer pulse unit (IPU).

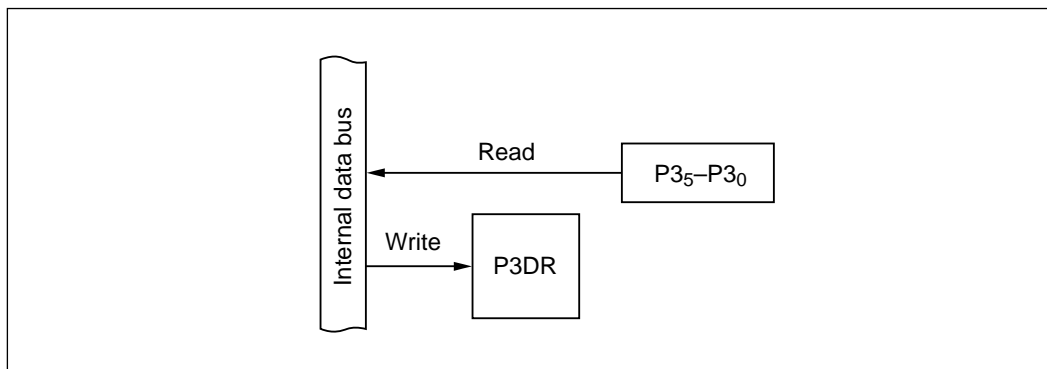
**(1) Pin Functions in Modes 1 to 7:** When a pin is used for IPU output, the setting in P3DDR is ignored. T1OC<sub>1</sub> to T1OC<sub>4</sub>, T2OC<sub>1</sub>, or T2OC<sub>2</sub> output is selected automatically. For methods of selecting pin functions, see appendix D “Pin Function Selection.”

**(2) Software Standby Mode:** Transition to software standby mode initializes the on-chip supporting modules, so port 3 becomes an input or output port according to P3DDR and P3DR.

#### 10.4.4 Port 3 Read/Write Operations

P3DR and P3DDR have different read/write functions depending on whether port 3 is used for the output compare function (T1OC<sub>1</sub> to T1OC<sub>4</sub>, T2OC<sub>1</sub>, T2OC<sub>2</sub>) of the 16-bit integrated-timer pulse unit (IPU) or general-purpose input or output (P3<sub>5</sub> to P3<sub>0</sub>). The operating states and functions of port 3 are described next.

**(1) Input Port (Modes 1 to 7):** Figure 10-17 shows a block diagram illustrating the general-purpose input function. Table 10-11 indicates register read/write data. Values written in the port 3 data register (P3DR) have no effect on general-purpose input lines. When read, P3DR returns the value at the pin.



**Figure 10-17 Input Port (Modes 1 to 7)**

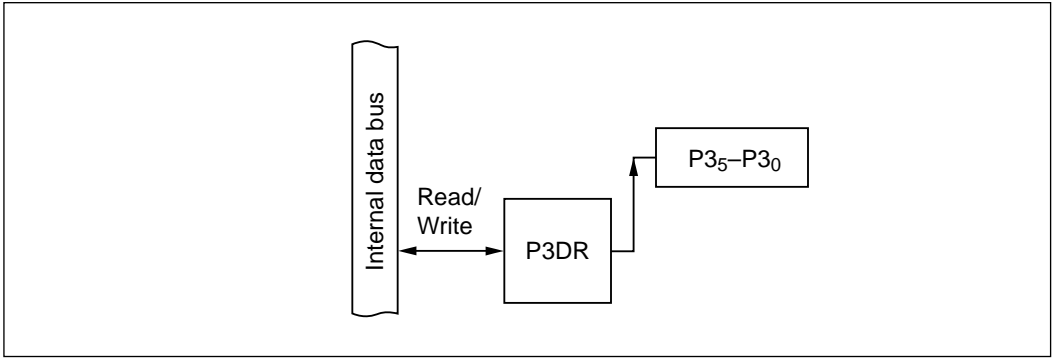
**Table 10-11 Register Read/Write Data**

	Read	Write
P3DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(2) Output Port (Modes 1 to 7):** Figure 10-18 shows a block diagram illustrating the general-purpose output function. Table 10-12 indicates register read/write data. The value written in the port 3 data register (P3DR) is output at the pin. When read, P3DR returns the value written in P3DR.



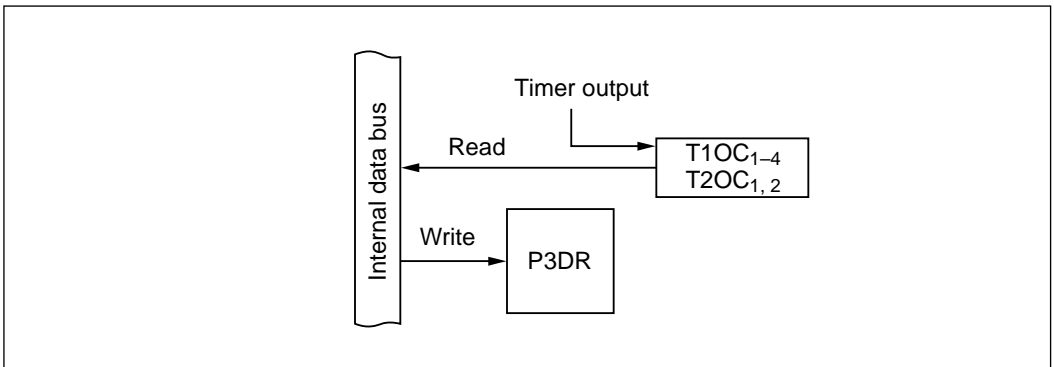


**Figure 10-18 Output Port (Modes 1 to 7)**

**Table 10-12 Register Read/Write Data**

	Read	Write
P3DR	P3DR value	Value output at pin

**(3) Timer Output Pins (Modes 1 to 7):** Figure 10-19 shows a block diagram illustrating the output function using the output compare output pins. Table 10-13 indicates register read/write data. When a pin in port 3 is used as an output compare output pin, the setting in the port 3 data direction register (P3DDR) is ignored. The value in the port 3 data register (P3DR) has no effect on the timer output. When read, P3DR returns the timer output level (T1OC<sub>1</sub> to T1OC<sub>4</sub>, T2OC<sub>1</sub>, or T2OC<sub>2</sub>).



**Figure 10-19 Timer Output Pins (Modes 1 to 7)**

**Table 10-13 Register Read/Write Data**

	Read	Write
P3DR	Pin value	Don't care*

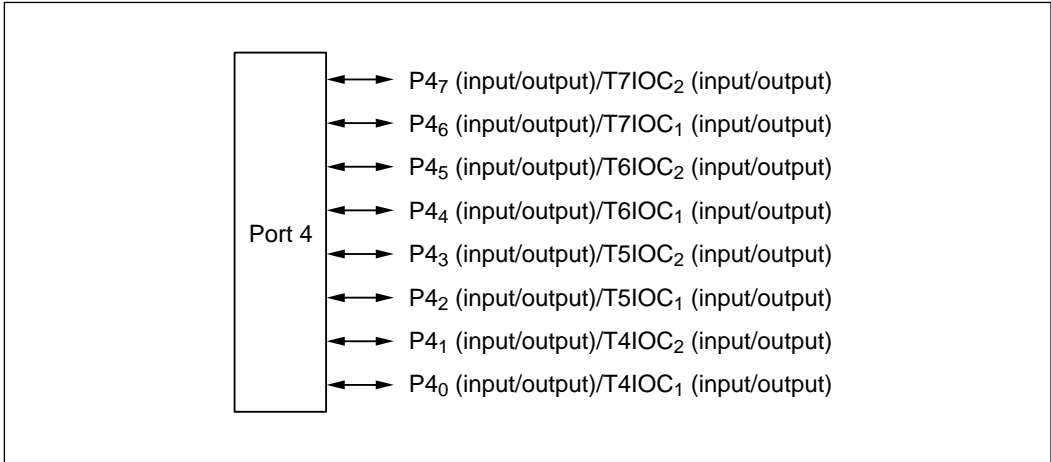
Note: The register can be written to, but the value is not output at the pins.

## 10.5 Port 4

### 10.5.1 Overview

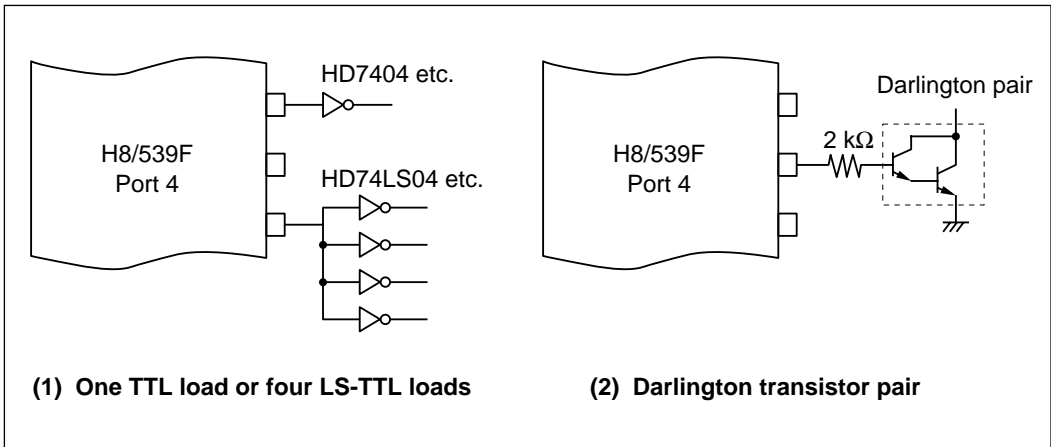
Port 4 is an eight-bit input/output port that is multiplexed with output compare and input capture pins (T7IOC<sub>2/1</sub>, T6IOC<sub>2/1</sub>, T5IOC<sub>2/1</sub>, T4IOC<sub>2/1</sub>) of the 16-bit integrated-timer pulse unit (IPU). Figure 10-20 summarizes the pin functions.

Pins in port 4 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair. P<sub>47</sub> to P<sub>40</sub> have Schmitt-trigger input circuits.



**Figure 10-20 Port 4 Pin Functions**

Figure 10-21 shows examples of output loads for port 4.



**Figure 10-21 Examples of Port 4 Output Loads**

## 10.5.2 Register Descriptions

Table 10-14 summarizes the registers of port 4.

**Table 10-14 Port 4 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE85	Port 4 data direction register	P4DDR	W	H'00
H'FE87	Port 4 data register	P4DR	R/W	H'00

**(1) Port 4 Data Direction Register:** The port 4 data direction register (P4DDR) is an eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	P4 <sub>7</sub> DDR	P4 <sub>6</sub> DDR	P4 <sub>5</sub> DDR	P4 <sub>4</sub> DDR	P4 <sub>3</sub> DDR	P4 <sub>2</sub> DDR	P4 <sub>1</sub> DDR	P4 <sub>0</sub> DDR
Initial value	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W

A pin in port 4 becomes an output pin if the corresponding P4DDR bit is set to 1, and an input pin if this bit is cleared to 0. P4DDR is a write-only register. All bits always return the value 1 when read.

P4DDR is initialized to H'00 by a reset and in hardware standby mode. P4DDR is not initialized in software standby mode.

**(2) Port 4 Data Register:** The port 4 data register (P4DR) is an eight-bit register that stores data for pins P4<sub>7</sub> to P4<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	P4 <sub>7</sub>	P4 <sub>6</sub>	P4 <sub>5</sub>	P4 <sub>4</sub>	P4 <sub>3</sub>	P4 <sub>2</sub>	P4 <sub>1</sub>	P4 <sub>0</sub>
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in P4DDR is set to 1, the corresponding P4DR bit value is output at the corresponding pin. If port 4 is read the value in P4DR is returned, regardless of the actual state of the pin.

When a bit in P4DDR is cleared to 0, it is possible to write to the corresponding P4DR bit but the value is not output at the pin. If P4DR is read the value at the pin is returned, regardless of the value written in P4DR.

P4DR is initialized to H'00 by a reset and in hardware standby mode. P4DR is not initialized in software standby mode.

### 10.5.3 Pin Functions in Each Mode

In all modes port 4 can be used for general-purpose input or output, or for the input capture and output compare functions of the 16-bit integrated-timer pulse unit (IPU).

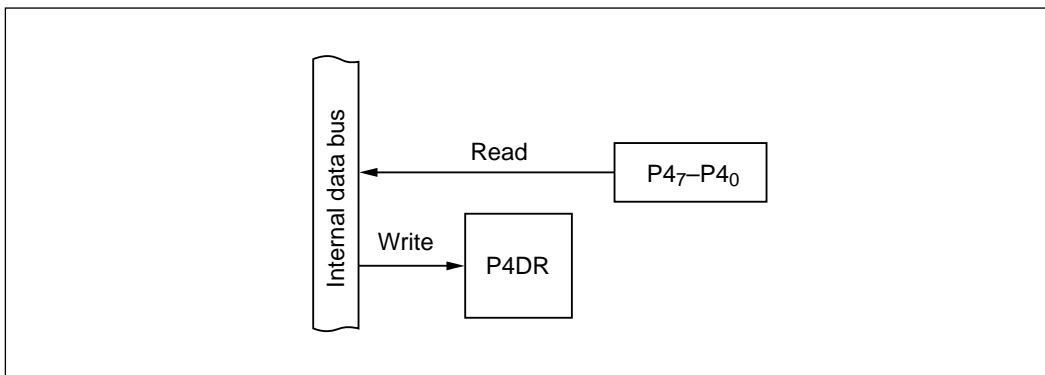
(1) **Pin Functions in Modes 1 to 7:** When a pin is used for the IPU output-compare function, the setting in P4DDR has no effect. T4IOC<sub>1</sub>, T4IOC<sub>2</sub>, T5IOC<sub>1</sub>, T5IOC<sub>2</sub>, T6IOC<sub>1</sub>, T6IOC<sub>2</sub>, T7IOC<sub>1</sub>, or T7IOC<sub>2</sub> output is selected automatically. When the IPU input capture function is selected, the P4DDR setting is valid and the pin can simultaneously function as a general-purpose input or output port. For methods of selecting pin functions, see appendix D “Pin Function Selection.”

(2) **Software Standby Mode:** Transition to software standby mode initializes the on-chip supporting modules, so port 4 becomes an input or output port according to P4DDR and P4DR.

### 10.5.4 Port 4 Read/Write Operations

P4DR and P4DDR have different read/write functions depending on whether port 4 is used for the input capture or output compare function (T4IOC<sub>1/2</sub>, T5IOC<sub>1/2</sub>, T6IOC<sub>1/2</sub>, T7IOC<sub>1/2</sub>) of the 16-bit integrated-timer pulse unit (IPU) or for general-purpose input or output (P4<sub>7</sub> to P4<sub>0</sub>). The operating states and functions of port 4 are described next.

(1) **Input Port (Modes 1 to 7):** Figure 10-22 shows a block diagram illustrating the general-purpose input function. Table 10-15 indicates register read/write data. Values written in the port 4 data register (P4DR) have no effect on general-purpose input lines. When read, P4DR returns the value at the pin.



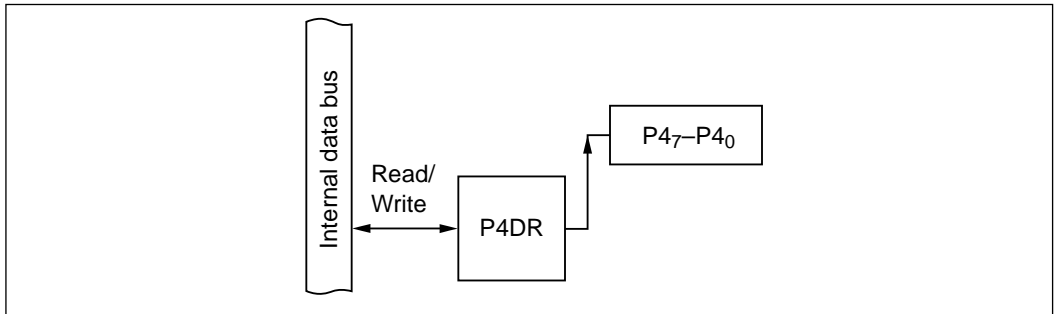
**Figure 10-22 Input Port (Modes 1 to 7)**

**Table 10-15 Register Read/Write Data**

	Read	Write
P4DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

(2) **Output Port (Modes 1 to 7):** Figure 10-23 shows a block diagram illustrating the general-purpose output function. Table 10-16 indicates register read/write data. The value written in the port 4 data register (P4DR) is output at the pin. When read, P4DR returns the value written in P4DR.

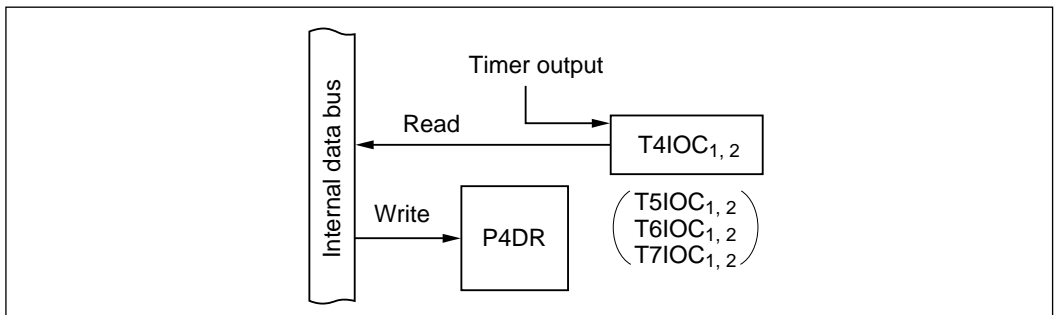


**Figure 10-23 Output Port (Modes 1 to 7)**

**Table 10-16 Register Read/Write Data**

	Read	Write
P4DR	P4DR value	Value output at pin

(3) **Timer Output Pins (Modes 1 to 7):** Figure 10-24 shows a block diagram illustrating the output compare function. Table 10-17 indicates register read/write data. When a pin in port 4 is used for output compare, the value in the port 4 data register (P4DR) has no effect on the timer output. When read, P4DR returns the timer output level (T4IOC<sub>1,2</sub>, T5IOC<sub>1,2</sub>, T6IOC<sub>1,2</sub>, T7IOC<sub>1,2</sub>).



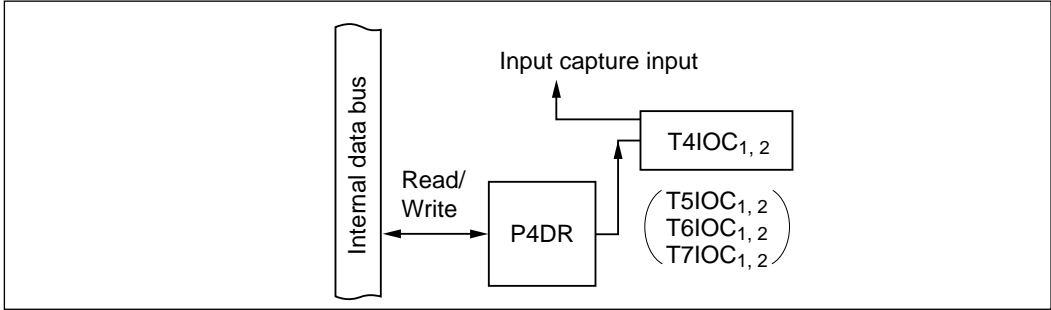
**Figure 10-24 Output Compare Pins (Modes 1 to 7)**

**Table 10-17 Register Read/Write Data**

	Read	Write
P4DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

**(4) Timer Input Combined with General-Purpose Output (Modes 1 to 7):** Figure 10-25 shows a block diagram illustrating the input capture function when combined with general-purpose output. Table 10-18 indicates register read/write data. An input capture pin can also function as an output port, in which case the output value is input to the timer.

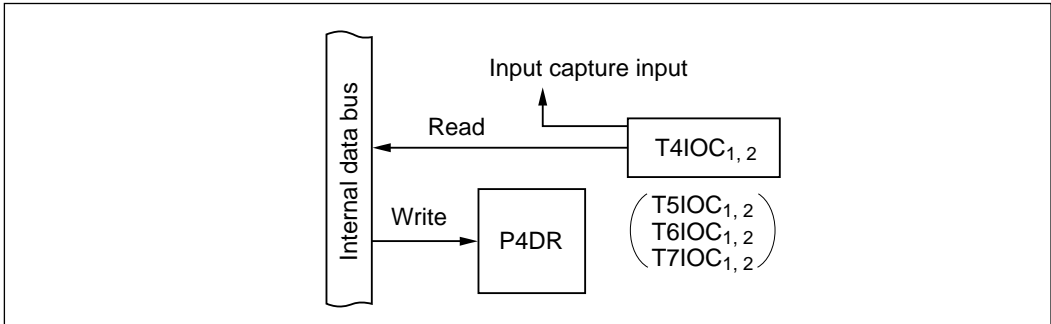


**Figure 10-25 Input Capture Combined with General-Purpose Output (Modes 1 to 7)**

**Table 10-18 Register Read/Write Data**

	Read	Write
P4DR	P4DR value	Value output at pin (timer input)

**(5) Timer Input Combined with General-Purpose Input (Modes 1 to 7):** Figure 10-26 shows a block diagram illustrating the input capture function when combined with general-purpose input. Table 10-19 indicates register read/write data. An input capture pin can also be read as an input port, to monitor the timer input level at T4IOC<sub>1</sub>, T4IOC<sub>2</sub>, T5IOC<sub>1</sub>, T5IOC<sub>2</sub>, T6IOC<sub>1</sub>, T6IOC<sub>2</sub>, T7IOC<sub>1</sub>, or T7IOC<sub>2</sub>.



**Figure 10-26 Input Capture Combined with General-Purpose Input (Modes 1 to 7)**

**Table 10-19 Register Read/Write Data**

	Read	Write
P4DR	Timer input	Don't care*

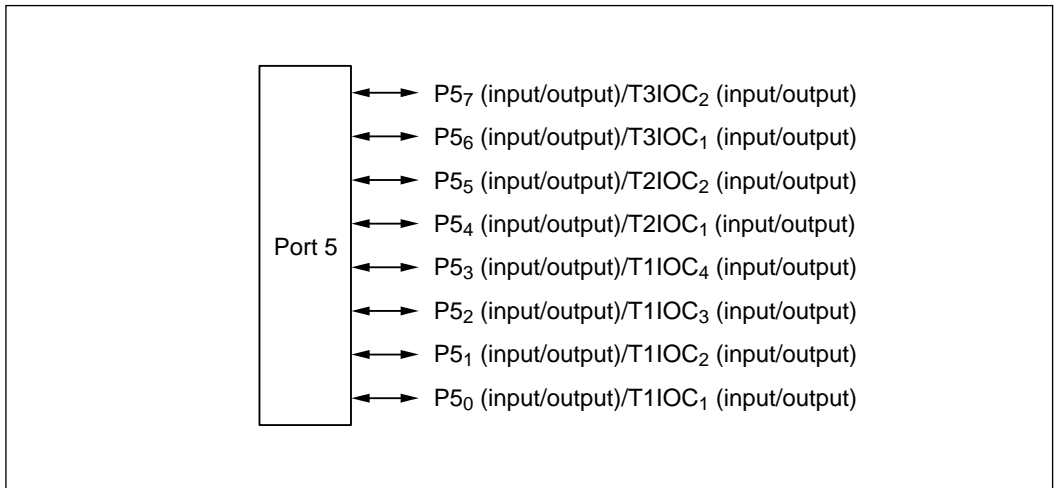
Note: The register can be written to, but the value is not output at the pins.

## 10.6 Port 5

### 10.6.1 Overview

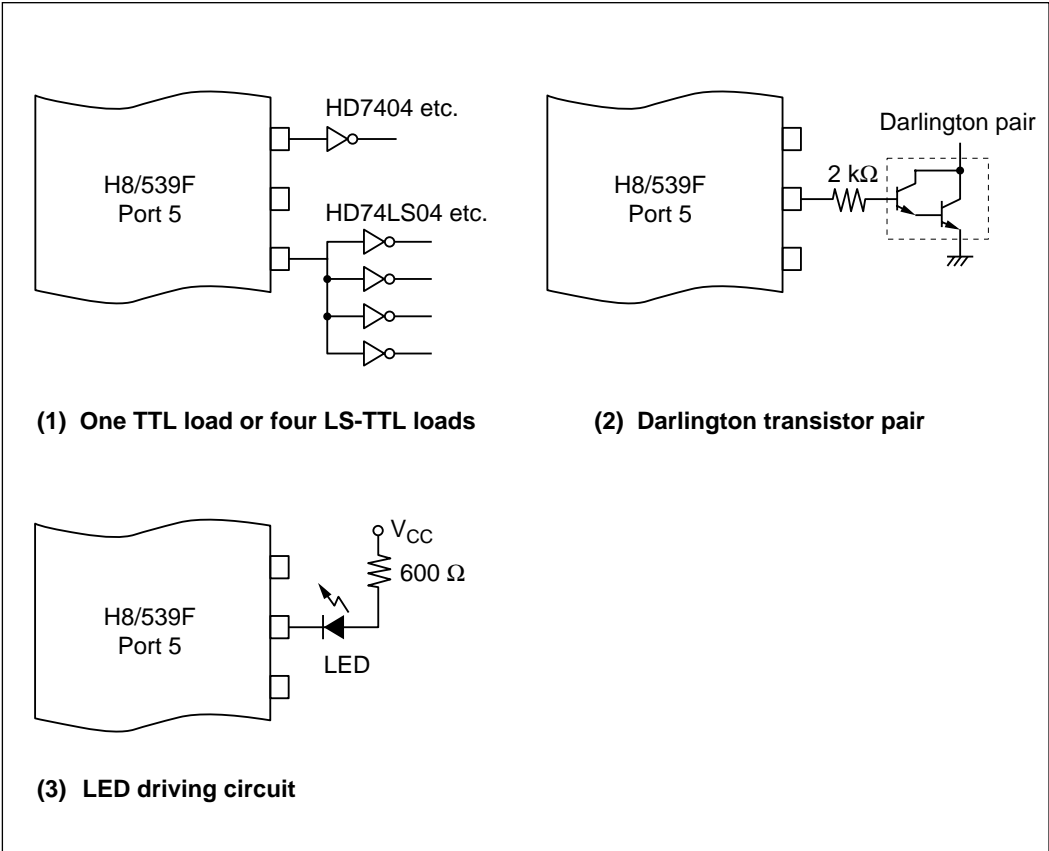
Port 5 is an eight-bit input/output port that is multiplexed with output compare and input capture pins (T3IOC<sub>2/1</sub>, T2IOC<sub>2/1</sub>, T1IOC<sub>4/3/2/1</sub>) of the 16-bit integrated-timer pulse unit (IPU). Figure 10-27 summarizes the pin functions.

Pins in port 5 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair or LED (with 10-mA current sink). Inputs are Schmitt-triggered.



**Figure 10-27 Port 5 Pin Functions**

Figure 10-28 shows examples of output loads for port 5.



**Figure 10-28 Examples of Port 5 Output Loads**

### 10.6.2 Register Descriptions

Table 10-20 summarizes the registers of port 5.

**Table 10-20 Port 5 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE88	Port 5 data direction register	P5DDR	W	H'00
H'FE8A	Port 5 data register	P5DR	R/W	H'00



**(1) Port 5 Data Direction Register:** The port 5 data direction register (P5DDR) is an eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	P5 <sub>7</sub> DDR	P5 <sub>6</sub> DDR	P5 <sub>5</sub> DDR	P5 <sub>4</sub> DDR	P5 <sub>3</sub> DDR	P5 <sub>2</sub> DDR	P5 <sub>1</sub> DDR	P5 <sub>0</sub> DDR
Initial value	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W

A pin in port 5 becomes an output pin if the corresponding P5DDR bit is set to 1, and an input pin if this bit is cleared to 0. P5DDR is a write-only register. All bits always return the value 1 when read.

P5DDR is initialized to H'00 by a reset and in hardware standby mode. P5DDR is not initialized in software standby mode.

**(2) Port 5 Data Register:** The port 5 data register (P5DR) is an eight-bit register that stores data for pins P5<sub>7</sub> to P5<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	P5 <sub>7</sub>	P5 <sub>6</sub>	P5 <sub>5</sub>	P5 <sub>4</sub>	P5 <sub>3</sub>	P5 <sub>2</sub>	P5 <sub>1</sub>	P5 <sub>0</sub>
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in P5DDR is set to 1, the corresponding P5DR bit value is output at the corresponding pin. If port 5 is read the value in P5DR is returned, regardless of the actual state of the pin.

When a bit in P5DDR is cleared to 0, it is possible to write to the corresponding P5DR bit but the value is not output at the pin. If P5DR is read the value at the pin is returned, regardless of the value written in P5DR.

P5DR is initialized to H'00 by a reset and in hardware standby mode. P5DR is not initialized in software standby mode.

### 10.6.3 Pin Functions in Each Mode

In all modes port 5 can be used for general-purpose input or output, or for the input capture and output compare functions of the 16-bit integrated-timer pulse unit (IPU).

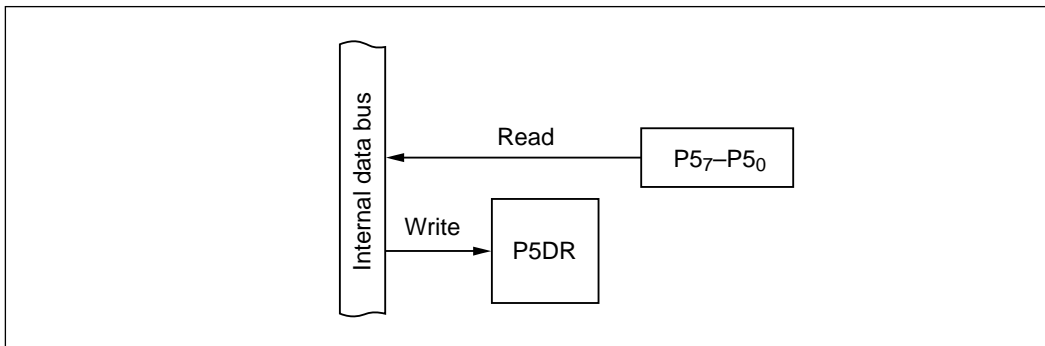
**(1) Pin Functions in Modes 1 to 7:** When a pin is used for the IPU output compare function, the setting in P5DDR is ignored. T1IOC<sub>1</sub> to T1IOC<sub>4</sub>, T2IOC<sub>1</sub>, T2IOC<sub>2</sub>, T3IOC<sub>1</sub>, or T3IOC<sub>2</sub> output is selected automatically. When the IPU input capture function is selected, the P5DDR setting is valid and the pin can simultaneously function as a general-purpose input or output port. For methods of selecting pin functions, see appendix D “Pin Function Selection.”

**(2) Software Standby Mode:** Transition to software standby mode initializes the on-chip supporting modules, so port 5 becomes an input or output port according to P5DDR and P5DR.

#### 10.6.4 Port 5 Read/Write Operations

P5DR and P5DDR have different read/write functions depending on whether port 5 is used for the input capture or output compare function (T1IOC<sub>1/2/3/4</sub>, T2IOC<sub>1/2</sub>, T3IOC<sub>1/2</sub>) of the 16-bit integrated-timer pulse unit (IPU) or for general-purpose input or output. The operating states and functions of port 5 are described next.

**(1) Input Port (Modes 1 to 7):** Figure 10-29 shows a block diagram illustrating the general-purpose input function. Table 10-21 indicates register read/write data. Values written in the port 5 data register (P5DR) have no effect on general-purpose input lines. When read, P5DR returns the value at the pin.



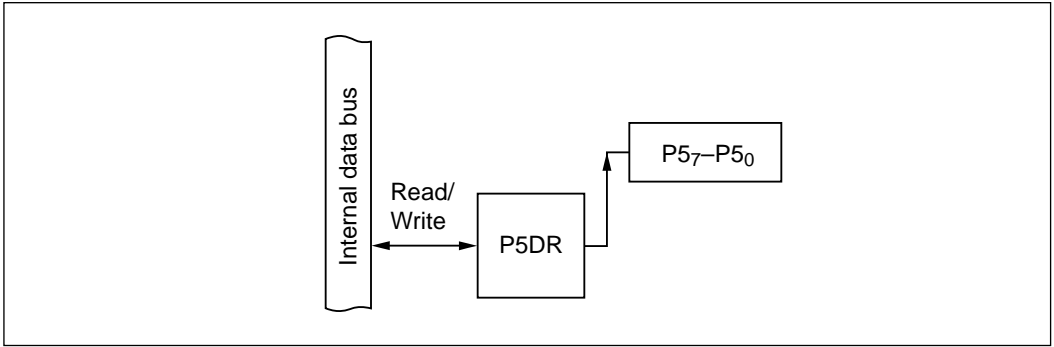
**Figure 10-29 Input Port (Modes 1 to 7)**

**Table 10-21 Register Read/Write Data**

	Read	Write
P5DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(2) Output Port (Modes 1 to 7):** Figure 10-30 shows a block diagram illustrating the general-purpose output function. Table 10-22 indicates register read/write data. The value written in the port 5 data register (P5DR) is output at the pin. When read, P5DR returns the value written in P5DR.

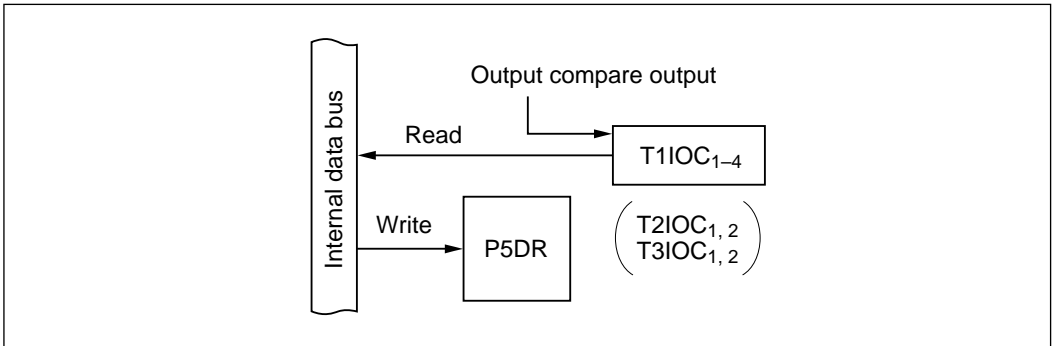


**Figure 10-30 Output Port (Modes 1 to 7)**

**Table 10-22 Register Read/Write Data**

	Read	Write
P5DR	P5DR value	Value output at pin

**(3) Timer Output Pins (Modes 1 to 7):** Figure 10-31 shows a block diagram illustrating the output compare function. Table 10-23 indicates register read/write data. When a pin in port 5 is used for output compare, the value in the port 5 data register (P5DR) has no effect on the timer output. P5DR can be read to monitor the timer output level (T1IOC<sub>1</sub> to T1IOC<sub>4</sub>, T2IOC<sub>1</sub>, T2IOC<sub>2</sub>, T3IOC<sub>1</sub>, T3IOC<sub>2</sub>).



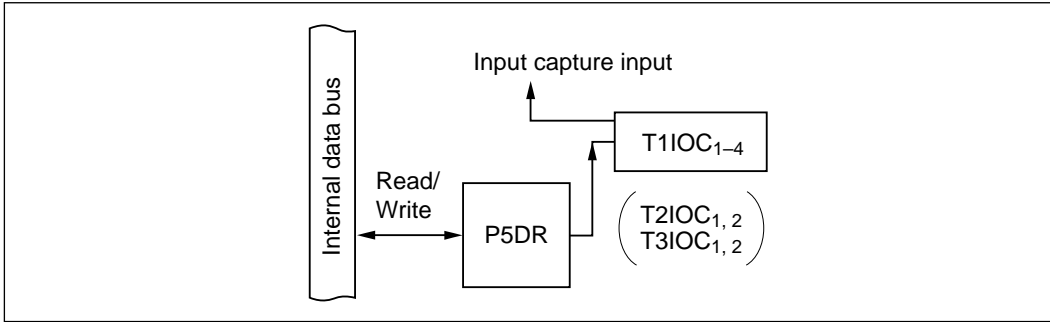
**Figure 10-31 Output Compare Pins (Modes 1 to 7)**

**Table 10-23 Register Read/Write Data**

	Read	Write
P5DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(4) Timer Input Combined with General-Purpose Output (Modes 1 to 7):** Figure 10-32 shows a block diagram illustrating the input capture function when combined with general-purpose output. Table 10-24 indicates register read/write data. An input capture pin can also function as an output port, in which case the output value is input to the timer.

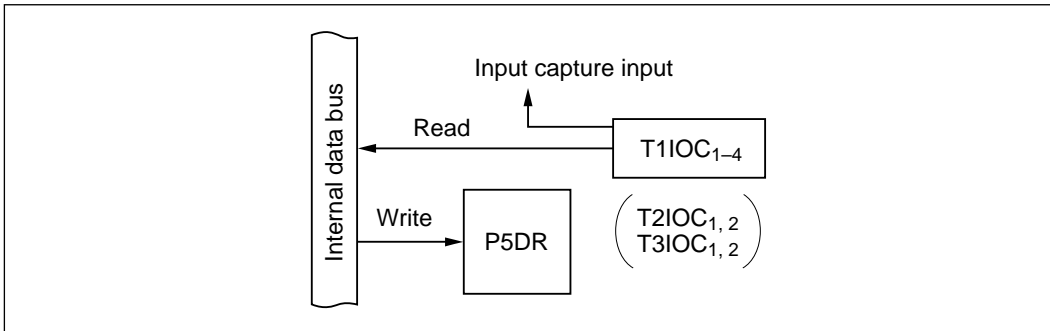


**Figure 10-32 Input Capture Combined with General-Purpose Output (Modes 1 to 7)**

**Table 10-24 Register Read/Write Data**

	Read	Write
P5DR	P5DR value	Value output at pin (Timer input)

**(5) Timer Input Combined with General-Purpose Input (Modes 1 to 7):** Figure 10-33 shows a block diagram illustrating the input capture function when combined with general-purpose input. Table 10-25 indicates register read/write data. An input capture pin can also be read as an input port, to monitor the timer input level at T1IOC<sub>1</sub> to T1IOC<sub>4</sub>, T2IOC<sub>1</sub>, T2IOC<sub>2</sub>, T3IOC<sub>1</sub>, or T3IOC<sub>2</sub>.



**Figure 10-33 Input Capture Combined with General-Purpose Input (Modes 1 to 7)**

**Table 10-25 Register Read/Write Data**

	Read	Write
P5DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

## 10.7 Port 6

### 10.7.1 Overview

Port 6 is a five-bit input/output port that is multiplexed with the external clock pins ( $\overline{\text{TCLK}}_{3/2/1}$ ) of the 16-bit integrated-timer pulse unit (IPU), with external interrupt pins ( $\overline{\text{IRQ}}_3$  and  $\overline{\text{IRQ}}_2$ ), and with a PWM timer output pin ( $\text{PW}_3$ ). Figure 10-34 summarizes the pin functions.

Pins in port 6 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair.

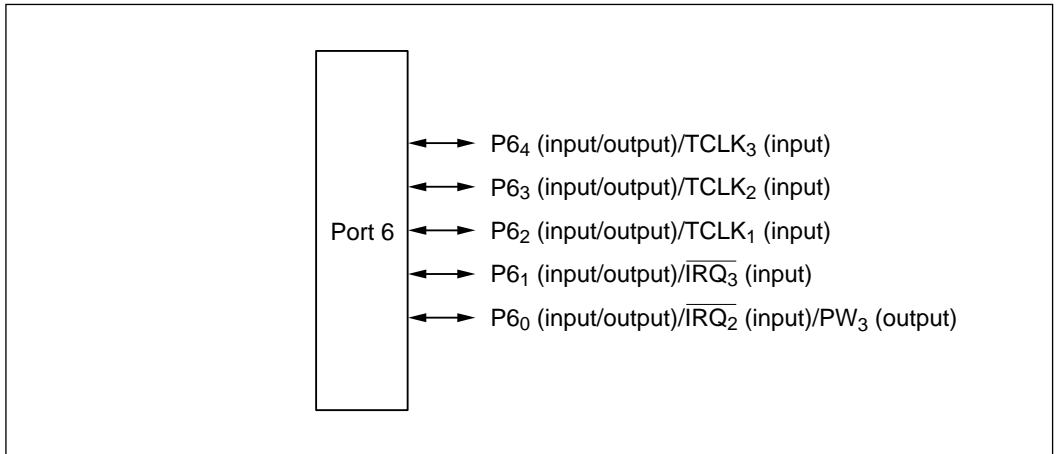


Figure 10-34 Port 6 Pin Functions

Figure 10-35 shows examples of output loads for port 6.

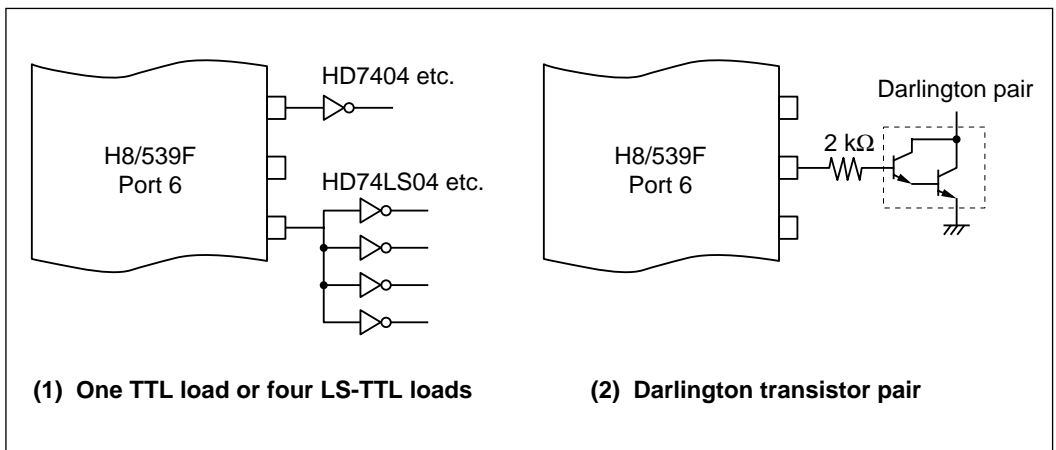


Figure 10-35 Examples of Port 6 Output Loads

## 10.7.2 Register Descriptions

Table 10-26 summarizes the registers of port 6.

**Table 10-26 Port 6 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE89	Port 6 data direction register	P6DDR	W	H'E0
H'FE8B	Port 6 data register	P6DR	R/W	H'E0
H'FEDB	Port 6/7 control register	P67CR	R/W	H'3E

**(1) Port 6 Data Direction Register:** The port 6 data direction register (P6DDR) is an eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	—	—	—	P6 <sub>4</sub> DDR	P6 <sub>3</sub> DDR	P6 <sub>2</sub> DDR	P6 <sub>1</sub> DDR	P6 <sub>0</sub> DDR
Initial value	1	1	1	0	0	0	0	0
R/W	—	—	—	W	W	W	W	W

A pin in port 6 becomes an output pin if the corresponding P6DDR bit is set to 1, and an input pin if this bit is cleared to 0. P6DDR is a write-only register. All bits always return the value 1 when read.

P6DDR is initialized to H'E0 by a reset and in hardware standby mode. P6DDR is not initialized in software standby mode.

**(2) Port 6 Data Register:** The port 6 data register (P6DR) is an eight-bit register that stores data for pins P6<sub>4</sub> to P6<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	—	—	—	P6 <sub>4</sub>	P6 <sub>3</sub>	P6 <sub>2</sub>	P6 <sub>1</sub>	P6 <sub>0</sub>
Initial value	1	1	1	0	0	0	0	0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W

When a bit in P6DDR is set to 1, the corresponding P6DR bit value is output at the corresponding pin.

When a bit in P6DDR is cleared to 0, it is possible to write to the corresponding P6DR bit but the value is not output at the pin. If P6DR is read the value at the pin is returned, regardless of the value written in P6DR.

P6DR is initialized to H'E0 by a reset and in hardware standby mode. P6DR is not initialized in software standby mode.

**(3) Port 6/7 Control Register:** The port 6/7 control register (P67CR) is an eight-bit register that controls the functions of pin P6<sub>0</sub> in port 6 and pins P7<sub>7</sub> and P7<sub>6</sub> in port 7.

Bit	7	6	5	4	3	2	1	0
	PW2E	PW1E	—	—	—	—	—	PW3E
Initial value	0	0	1	1	1	1	1	0
R/W	R/W	R/W	R	R	R	R	R	R/W

**Bits 7 and 6—PW<sub>2</sub> Enable and PW<sub>1</sub> Enable (PW2E, PW1E):** These bits control the PWM output function of pins P7<sub>7</sub>/SCK<sub>2</sub>/PW<sub>2</sub> and P7<sub>6</sub>/SCK<sub>1</sub>/PW<sub>1</sub> in port 7. When bits PW2E and PW1E are set to 1, these pins can be used for PW<sub>2</sub> and PW<sub>1</sub> output and cannot be used for SCK<sub>2</sub> and SCK<sub>1</sub> output.

**Bit 0—PW<sub>3</sub> Enable (PW3E):** Controls the PWM output function of pin P6<sub>0</sub>/IRQ<sub>2</sub>/PW<sub>3</sub> in port 6. When bit PW3E is set to 1, this pin can be used for PW<sub>3</sub> output.

### 10.7.3 Pin Functions in Each Mode

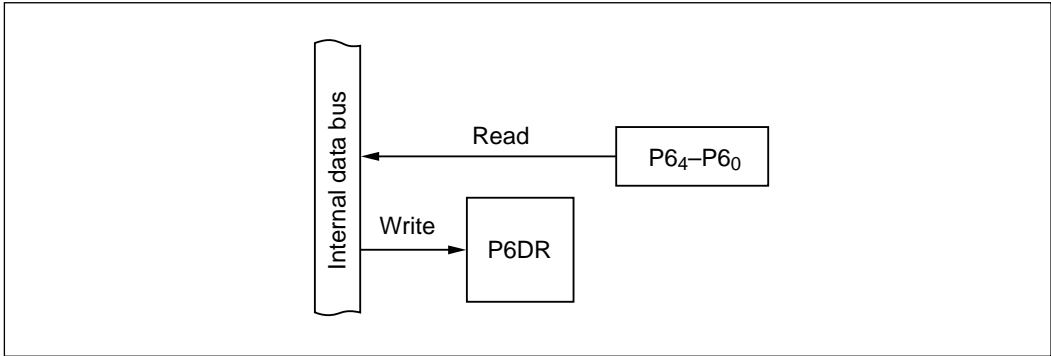
**(1) Pin Functions in Modes 1 to 7:** When a pin is used for IPU external clock input (TCLK<sub>3/2/1</sub>) or external interrupt input (IRQ<sub>3/2</sub>), it can simultaneously function as a general-purpose input or output port. When a pin is used for PWM timer output (PW<sub>3</sub>), the P6DDR setting is disregarded and the PW<sub>3</sub> function is selected. For methods of selecting pin functions, see appendix D “Pin Function Selection.”

**(2) Software Standby Mode:** Transition to software standby mode initializes the on-chip supporting modules, so port 6 becomes an input or output port according to P6DDR and P6DR.

### 10.7.4 Port 6 Read/Write Operations

P6DR and P6DDR have different read/write functions depending on whether port 6 is used for external clock input (TCLK<sub>3/2/1</sub>) to the 16-bit integrated-timer pulse unit (IPU), external interrupt input (IRQ<sub>3/2</sub>), PWM timer output (PW<sub>3</sub>), or general-purpose input or output (P6<sub>4</sub> to P6<sub>0</sub>). The operating states and functions of port 6 are described next.

**(1) Input Port (Modes 1 to 7):** Figure 10-36 shows a block diagram illustrating the general-purpose input function. Table 10-27 indicates register read/write data. Values written in the port 6 data register (P6DR) have no effect on general-purpose input lines. When read, P6DR returns the value at the pin.



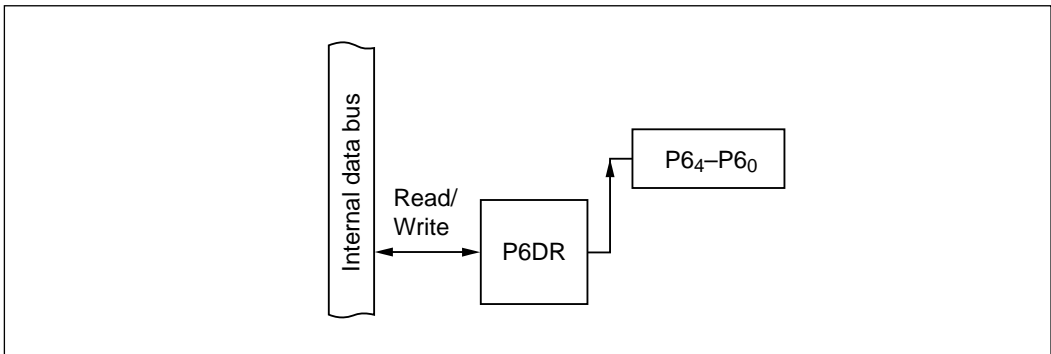
**Figure 10-36 Input Port (Modes 1 to 7)**

**Table 10-27 Register Read/Write Data**

	Read	Write
P6DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(2) Output Port (Modes 1 to 7):** Figure 10-37 shows a block diagram illustrating the general-purpose output function. Table 10-28 indicates register read/write data. The value written in the port 6 data register (P6DR) is output at the pin. When read, P6DR returns the value written in P6DR.



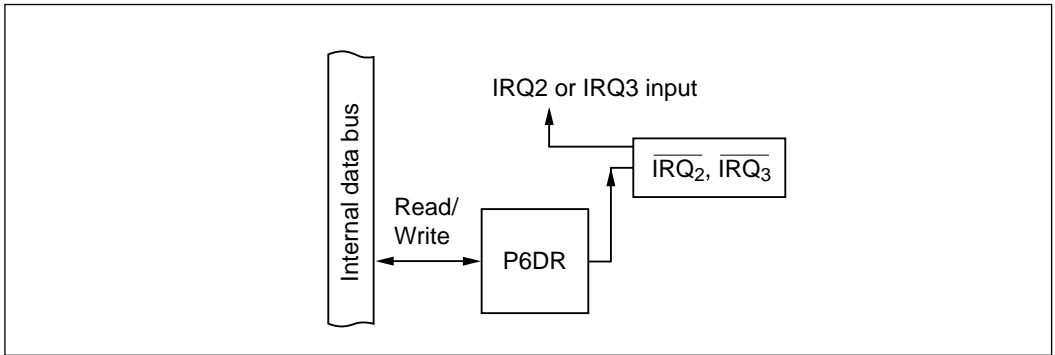
**Figure 10-37 Output Port (Modes 1 to 7)**



**Table 10-28 Register Read/Write Data**

	Read	Write
P6DR	P6DR value	Value output at pin

**(3)  $\overline{\text{IRQ}}_3$  or  $\overline{\text{IRQ}}_2$  Input Combined with General-Purpose Output (P6<sub>1</sub>, P6<sub>0</sub>: modes 1 to 7):**  
 Figure 10-38 shows a block diagram illustrating the  $\overline{\text{IRQ}}_3$  and  $\overline{\text{IRQ}}_2$  input function of P6<sub>1</sub> and P6<sub>0</sub> when combined with general-purpose output. Table 10-29 indicates register read/write data. When P6<sub>1</sub> and P6<sub>0</sub> are used for  $\overline{\text{IRQ}}_3$  and  $\overline{\text{IRQ}}_2$  input they can also function as general-purpose output ports. If the general-purpose output function is used, however, output of a falling edge will cause an interrupt.

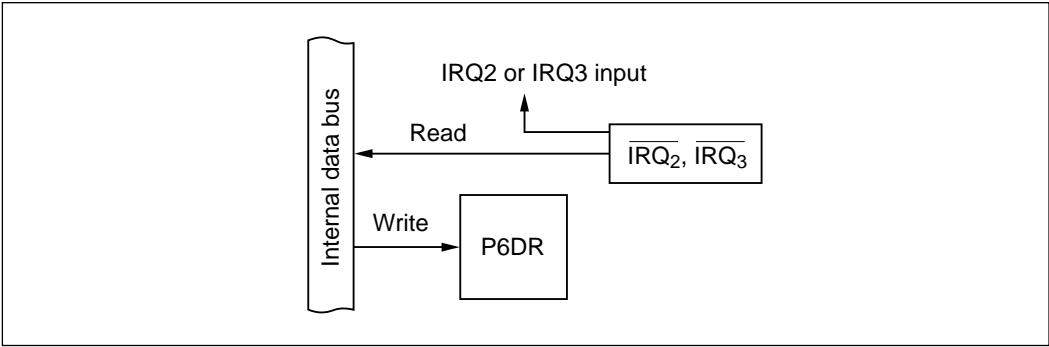


**Figure 10-38  $\overline{\text{IRQ}}_3$  or  $\overline{\text{IRQ}}_2$  Input Combined with General-Purpose Output (Modes 1 to 7)**

**Table 10-29 Register Read/Write Data**

	Read	Write
P6DR	P6DR value	Value output at pin

**(4)  $\overline{\text{IRQ}}_3$  or  $\overline{\text{IRQ}}_2$  Input Combined with General-Purpose Input (P6<sub>1</sub>, P6<sub>0</sub>: Modes 1 to 7):**  
 Figure 10-39 shows a block diagram illustrating the  $\overline{\text{IRQ}}_3$  and  $\overline{\text{IRQ}}_2$  input function when combined with general-purpose input. Table 10-30 indicates register read/write data. When P6<sub>1</sub> and P6<sub>0</sub> are used for  $\overline{\text{IRQ}}_3$  and  $\overline{\text{IRQ}}_2$  input they can also be read as general-purpose input ports, to monitor the input level at  $\overline{\text{IRQ}}_3$  or  $\overline{\text{IRQ}}_2$ .



**Figure 10-39**  $\overline{\text{IRQ}}_3$  or  $\overline{\text{IRQ}}_2$  Input Combined with General-Purpose Input (Modes 1 to 7)

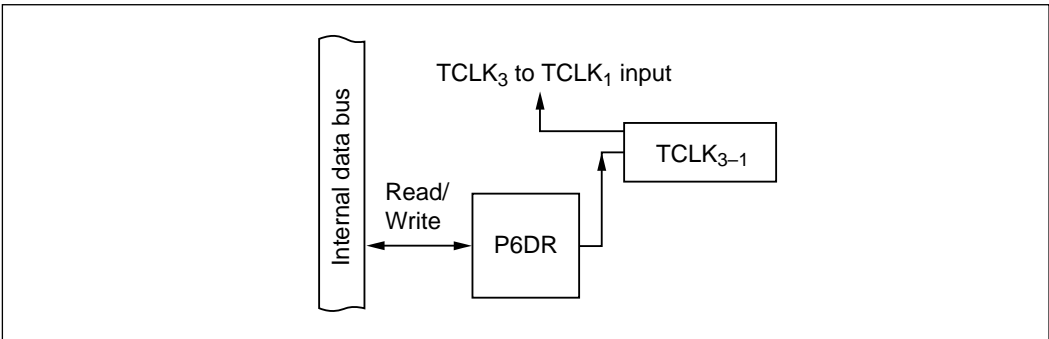
**Table 10-30** Register Read/Write Data

	Read	Write
P6DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(5) Timer Clock Input Combined with General-Purpose Output (P<sub>64</sub> to P<sub>62</sub>: Modes 1 to 7):**

Figure 10-40 shows a block diagram illustrating the TCLK<sub>3</sub> to TCLK<sub>1</sub> input function of P<sub>64</sub> to P<sub>62</sub> when combined with general-purpose output. Table 10-31 indicates register read/write data. When P<sub>64</sub> to P<sub>62</sub> are used for TCLK<sub>3</sub>, TCLK<sub>2</sub>, and TCLK<sub>1</sub> input they can also function as general-purpose output ports.

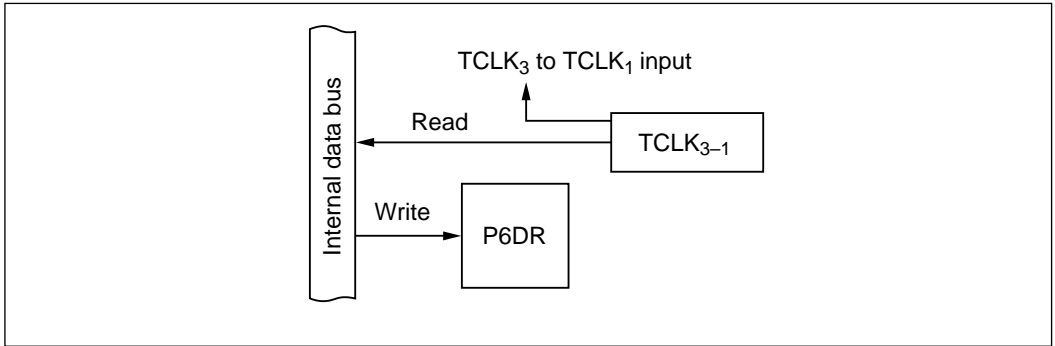


**Figure 10-40** TCLK<sub>3</sub> to TCLK<sub>1</sub> Input Combined with General-Purpose Output (Modes 1 to 7)

**Table 10-31** Register Read/Write Data

	Read	Write
P6DR	P6DR value	Value output at pin

**(6) Timer Clock Input Combined with General-Purpose Input (P6<sub>4</sub> to P6<sub>2</sub>: Modes 1 to 7):** Figure 10-41 shows a block diagram illustrating the TCLK<sub>3</sub> to TCLK<sub>1</sub> input function of P6<sub>4</sub> to P6<sub>2</sub> when combined with general-purpose input. Table 10-32 indicates register read/write data. When P6<sub>4</sub> to P6<sub>2</sub> are used for TCLK<sub>3</sub>, TCLK<sub>2</sub>, and TCLK<sub>1</sub> input they can also be read as general-purpose input ports, to monitor the input level at TCLK<sub>3</sub> to TCLK<sub>1</sub>.



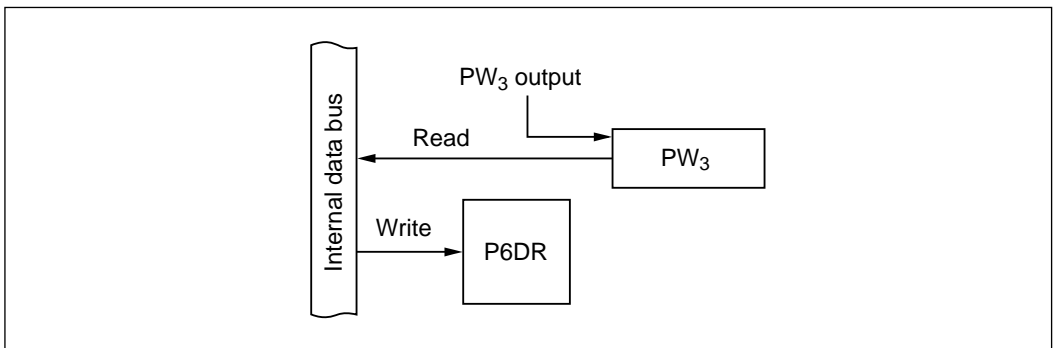
**Figure 10-41 TCLK<sub>3</sub> to TCLK<sub>1</sub> Input Combined with General-Purpose Input (Modes 1 to 7)**

**Table 10-32 Register Read/Write Data**

	Read	Write
P6DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(7) PW<sub>3</sub> Output Combined with General-Purpose Input (P6<sub>0</sub>: Modes 1 to 7):** Figure 10-42 shows a block diagram illustrating the PW<sub>3</sub> output function of P6<sub>0</sub> when combined with general-purpose input. Table 10-33 indicates register read/write data. When P6<sub>0</sub> is used for PW<sub>3</sub> output it can also be read as a general-purpose input port, to monitor the state of the PW<sub>3</sub> pin.



**Figure 10-42 PW<sub>3</sub> Output Combined with General-Purpose Input (Modes 1 to 7)**

**Table 10-33 Register Read/Write Data**

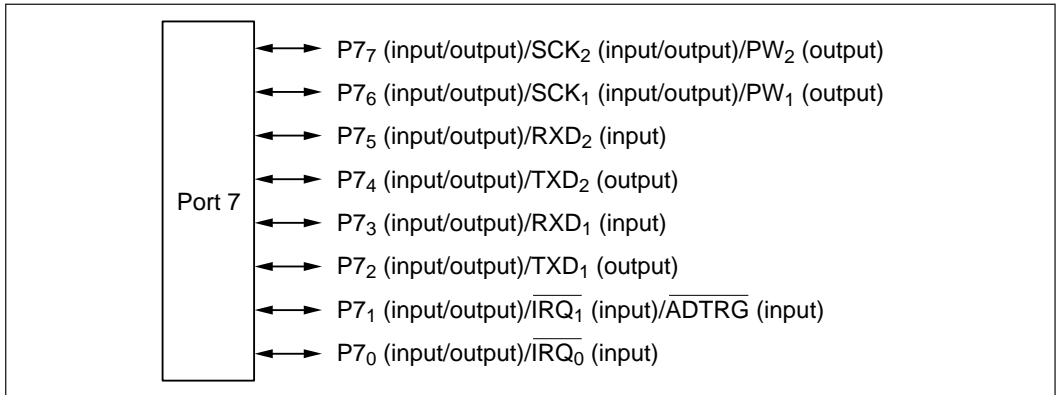
	<b>Read</b>	<b>Write</b>
P6DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

## 10.8 Port 7

### 10.8.1 Overview

Port 7 is an eight-bit input/output port that is multiplexed with the serial clock input/output pins (SCK<sub>2</sub> and SCK<sub>1</sub>), transmit data output pins (TXD<sub>2</sub> and TXD<sub>1</sub>), and receive data input pins (RXD<sub>2</sub> and RXD<sub>1</sub>) of the serial communication interface (SCI), with PWM timer output pins (PW<sub>1</sub> and PW<sub>2</sub>), with external interrupt pins (IRQ<sub>1</sub> and IRQ<sub>0</sub>), and with the external trigger pin (ADTRG) of the A/D converter. Figure 10-43 summarizes the pin functions. Pins in port 7 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair.



**Figure 10-43 Port 7 Pin Functions**

Figure 10-44 shows examples of output loads for port 7.

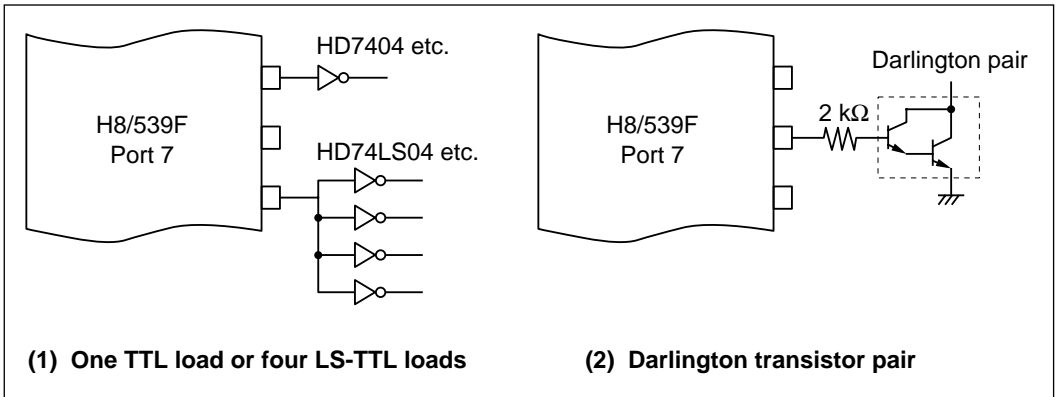


Figure 10-44 Examples of Port 7 Output Loads

### 10.8.2 Register Descriptions

Table 10-34 summarizes the registers of port 7.

Table 10-34 Port 7 Registers

Address	Name	Abbreviation	R/W	Initial Value
H'FE8C	Port 7 data direction register	P7DDR	W	H'00
H'FE8E	Port 7 data register	P7DR	R/W	H'00
H'FEDE	Port 6/7 control register	P67CR	R/W	H'3E

**(1) Port 7 Data Direction Register:** The port 7 data direction register (P7DDR) is an eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	P7 <sub>7</sub> DDR	P7 <sub>6</sub> DDR	P7 <sub>5</sub> DDR	P7 <sub>4</sub> DDR	P7 <sub>3</sub> DDR	P7 <sub>2</sub> DDR	P7 <sub>1</sub> DDR	P7 <sub>0</sub> DDR
Initial value	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W

A pin in port 7 becomes an output pin if the corresponding P7DDR bit is set to 1, and an input pin if this bit is cleared to 0. P7DDR is a write-only register. All bits always return the value 1 when read.

P7DDR is initialized to H'00 by a reset and in hardware standby mode. P7DDR is not initialized in software standby mode.

**(2) Port 7 Data Register:** The port 7 data register (P7DR) is an eight-bit register that stores data for pins P7<sub>7</sub> to P7<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	P7 <sub>7</sub>	P7 <sub>6</sub>	P7 <sub>5</sub>	P7 <sub>4</sub>	P7 <sub>3</sub>	P7 <sub>2</sub>	P7 <sub>1</sub>	P7 <sub>0</sub>
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in P7DDR is set to 1, the corresponding P7DR bit value is output at the corresponding pin. If port 7 is read the value in P7DR is returned, regardless of the actual state of the pin.

When a bit in P7DDR is cleared to 0, it is possible to write to the corresponding P7DR bit but the value is not output at the pin. If P7DR is read the value at the pin is returned, regardless of the value written in P7DR.

P7DR is initialized to H'00 by a reset and in hardware standby mode. P7DR is not initialized in software standby mode.

**(3) Port 6/7 Control Register:** The port 6/7 control register (P67CR) is an eight-bit register that controls the functions of pin P6<sub>0</sub> in port 6 and pins P7<sub>7</sub> and P7<sub>6</sub> in port 7.

Bit	7	6	5	4	3	2	1	0
	PW2E	PW1E	—	—	—	—	—	PW3E
Initial value	0	0	1	1	1	1	1	0
R/W	R/W	R/W	R	R	R	R	R	R/W

**Bits 7 and 6—PW<sub>2</sub> Enable and PW<sub>1</sub> Enable (PW2E, PW1E):** These bits control the PWM output function of pins P7<sub>7</sub>/SCK<sub>2</sub>/PW<sub>2</sub> and P7<sub>6</sub>/SCK<sub>1</sub>/PW<sub>1</sub> in port 7. When bits PW2E and PW1E are set to 1, these pins can be used for PW<sub>2</sub> and PW<sub>1</sub> output and cannot be used for SCK<sub>2</sub> and SCK<sub>1</sub> output.

**Bit 0—PW<sub>3</sub> Enable (PW3E):** Controls the PWM output function of pin P6<sub>0</sub>/IRQ<sub>2</sub>/PW<sub>3</sub> in port 6. When bit PW3E is set to 1, this pin can be used for PW<sub>3</sub> output.

### 10.8.3 Pin Functions in Each Mode

**(1) Pin Functions in Modes 1 to 7:** When a pin is used for input or output by the serial communication interface (SCI) or a PWM timer, the P7DDR setting is disregarded and the pin is used for serial clock input or output (SCK<sub>2/1</sub>), transmit data output (TXD<sub>2/1</sub>), receive data input (RXD<sub>2/1</sub>), or PWM timer output (PW<sub>1/2</sub>).

When P7<sub>1</sub> and P7<sub>0</sub> are used for external interrupt input ( $\overline{\text{IRQ}}_1$  and  $\overline{\text{IRQ}}_0$ ), they can simultaneously function as general-purpose input or output ports. P7<sub>1</sub> can also function as the external trigger signal ( $\overline{\text{ADTRG}}$ ) for the A/D converter.

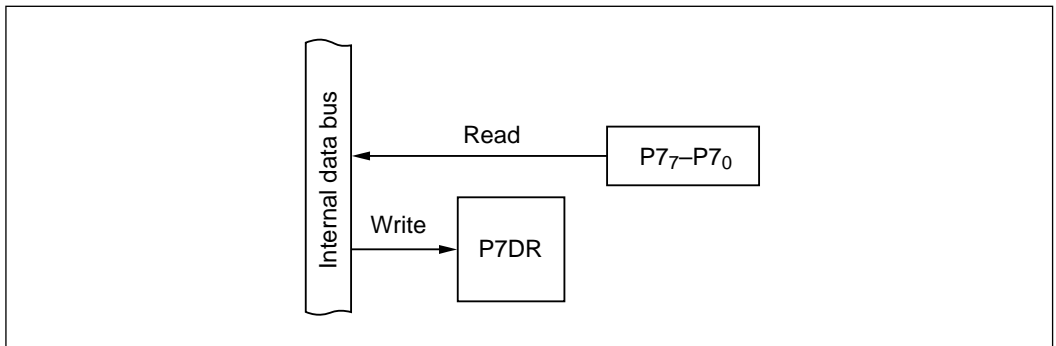
For methods of selecting pin functions, see appendix D “Pin Function Selection.”

**(2) Software Standby Mode:** Transition to software standby mode initializes the on-chip supporting modules, so port 7 becomes an input or output port according to P7DDR and P7DR.

### 10.8.4 Port 7 Read/Write Operations

P7DR and P7DDR have different read/write functions depending on whether port 7 is used for output of transmit data (TXD<sub>1/2</sub>), input of receive data (RXD<sub>1/2</sub>), input or output of serial clocks (SCK<sub>1/2</sub>) for the serial communication interface, PWM timer output (PW<sub>2/1</sub>), external interrupt input ( $\overline{\text{IRQ}}_{1/0}$ ), or general-purpose input or output. The operating states and functions of port 7 are described next.

**(1) Input Port (Modes 1 to 7):** Figure 10-45 shows a block diagram illustrating the general-purpose input function. Table 10-35 indicates register read/write data. Values written in the port 7 data register (P7DR) have no effect on general-purpose input lines. When read, P7DR returns the value at the pin.



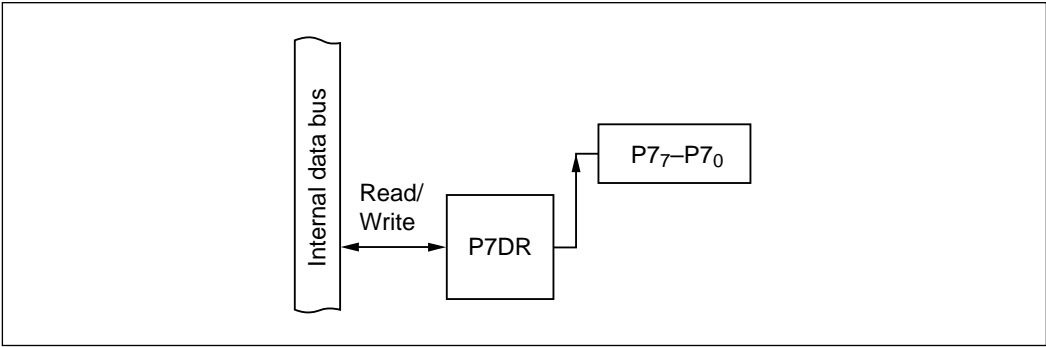
**Figure 10-45 Input Port (Modes 1 to 7)**

**Table 10-35 Register Read/Write Data**

	Read	Write
P7DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(2) Output Port (Modes 1 to 7):** Figure 10-46 shows a block diagram illustrating the general-purpose output function. Table 10-36 indicates register read/write data. The value written in the port 7 data register (P7DR) is output at the pin. When read, P7DR returns the value written in P7DR.



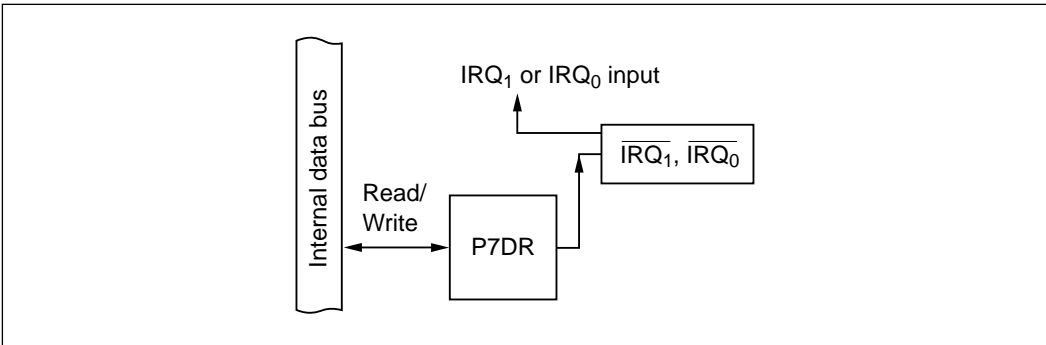
**Figure 10-46 Output Port (Modes 1 to 7)**

**Table 10-36 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
P7DR	P7DR value	Value output at pin

**(3)  $\overline{\text{IRQ}}_1$  or  $\overline{\text{IRQ}}_0$  Input Combined with General-Purpose Output (P7<sub>1</sub>, P7<sub>0</sub>: Modes 1 to 7):**

Figure 10-47 shows a block diagram illustrating the  $\overline{\text{IRQ}}_1$  and  $\overline{\text{IRQ}}_0$  input function when combined with general-purpose output. Table 10-37 indicates register read/write data. When P7<sub>1</sub> and P7<sub>0</sub> are used for  $\overline{\text{IRQ}}_1$  and  $\overline{\text{IRQ}}_0$  input they can also function as general-purpose output ports. If the general-purpose output function is used, however, output of a falling edge will cause an interrupt.



**Figure 10-47  $\overline{\text{IRQ}}_1$  or  $\overline{\text{IRQ}}_0$  Input Combined with General-Purpose Output (Modes 1 to 7)**

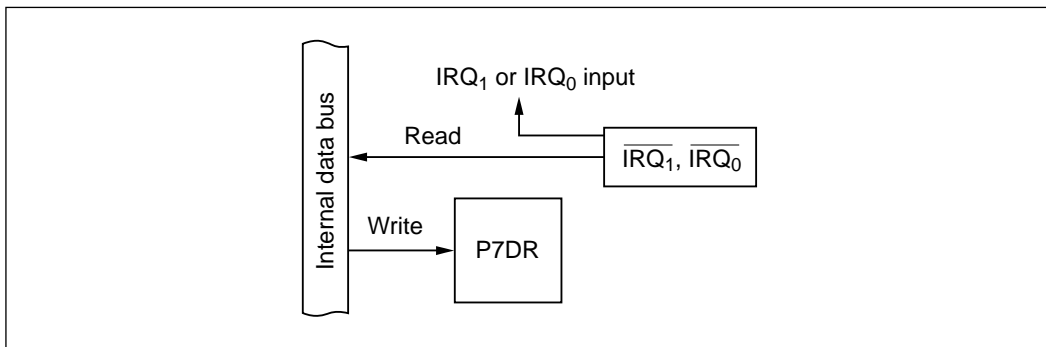
**Table 10-37 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
P7DR	P7DR value	Value output at pin



**(4)  $\overline{\text{IRQ}}_1$  or  $\overline{\text{IRQ}}_0$  Input Combined with General-Purpose Input (P7<sub>1</sub>, P7<sub>0</sub>: Modes 1 to 7):**

Figure 10-48 shows a block diagram illustrating the  $\overline{\text{IRQ}}_1$  and  $\overline{\text{IRQ}}_0$  input function when combined with general-purpose input. Table 10-38 indicates register read/write data. When P7<sub>1</sub> and P7<sub>0</sub> are used for  $\overline{\text{IRQ}}_1$  and  $\overline{\text{IRQ}}_0$  input they can also be read as general-purpose input ports, to monitor the input level at  $\overline{\text{IRQ}}_1$  or  $\overline{\text{IRQ}}_0$ .



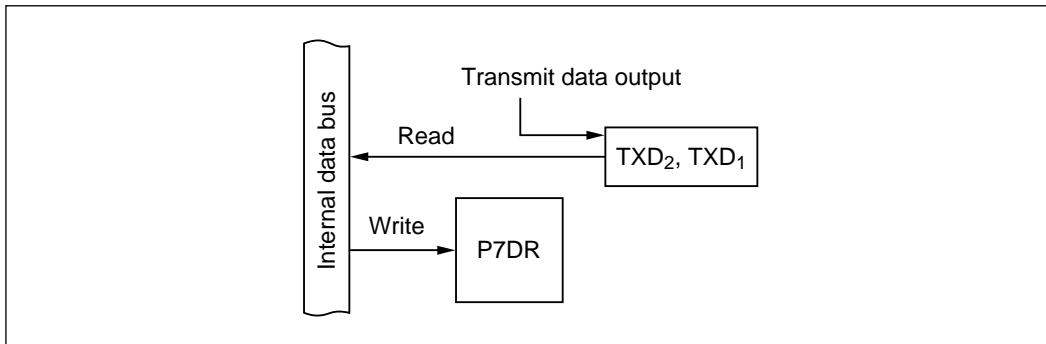
**Figure 10-48  $\overline{\text{IRQ}}_1$  or  $\overline{\text{IRQ}}_0$  Input Combined with General-Purpose Input (Modes 1 to 7)**

**Table 10-38 Register Read/Write Data**

	Read	Write
P7DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

**(5) TXD<sub>2</sub> and TXD<sub>1</sub> Output (P7<sub>4</sub> and P7<sub>2</sub>: Modes 1 to 7):** Figure 10-49 shows a block diagram illustrating the TXD<sub>2</sub> and TXD<sub>1</sub> output function. Table 10-39 indicates register read/write data. When P7<sub>4</sub> and P7<sub>2</sub> are used for TXD<sub>2</sub> and TXD<sub>1</sub> output, the value written in P7DR is ignored, but P7DR can be read to monitor the levels at the TXD<sub>2</sub> and TXD<sub>1</sub> pins.



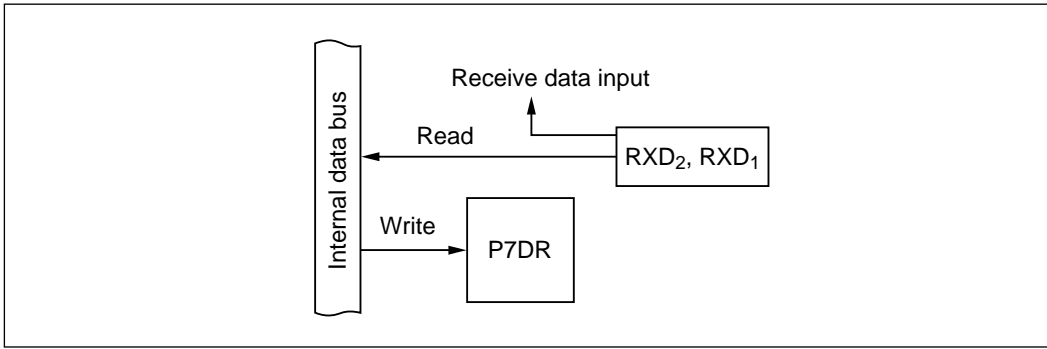
**Figure 10-49 TXD<sub>2</sub> and TXD<sub>1</sub> Output (Modes 1 to 7)**

**Table 10-39 Register Read/Write Data**

	Read	Write
P7DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

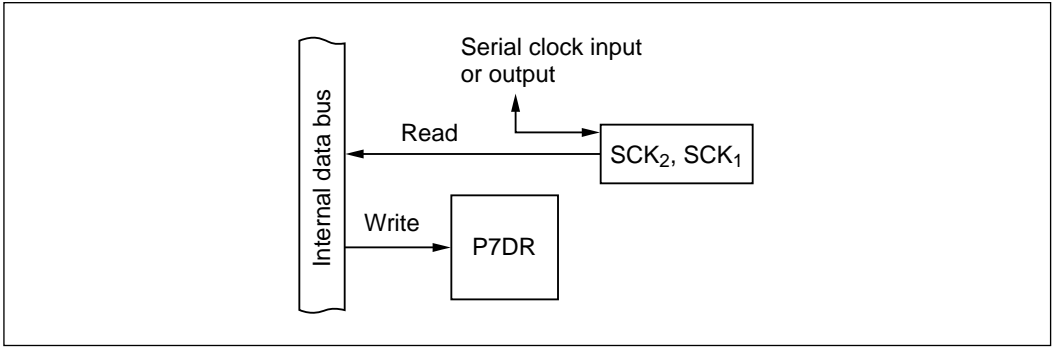
(6) **RXD<sub>2</sub> and RXD<sub>1</sub> Input (P7<sub>5</sub> and P7<sub>3</sub>: Modes 1 to 7)**: Figure 10-50 shows a block diagram illustrating the RXD<sub>2</sub> and RXD<sub>1</sub> input function. Table 10-40 indicates register read/write data. When P7<sub>5</sub> and P7<sub>3</sub> are used for RXD<sub>2</sub> and RXD<sub>1</sub> input, the value written in P7DR is ignored, but P7DR can be read to monitor the levels at the RXD<sub>2</sub> and RXD<sub>1</sub> pins (to detect the line break state, for example).

**Figure 10-50 RXD<sub>2</sub> and RXD<sub>1</sub> Input (Modes 1 to 7)****Table 10-40 Register Read/Write Data**

	Read	Write
P7DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

(7) **SCK<sub>2</sub> and SCK<sub>1</sub> Pins (P7<sub>6</sub> and P7<sub>4</sub>: Modes 1 to 7)**: Figure 10-51 shows a block diagram illustrating the SCK<sub>2</sub> and SCK<sub>1</sub> input/output function. Table 10-41 indicates register read/write data. When P7<sub>6</sub> and P7<sub>4</sub> are used for SCK<sub>2</sub> and SCK<sub>1</sub> input or output, the value written in P7DR is ignored, but P7DR can be read to monitor the levels at the SCK<sub>2</sub> and SCK<sub>1</sub> pins.



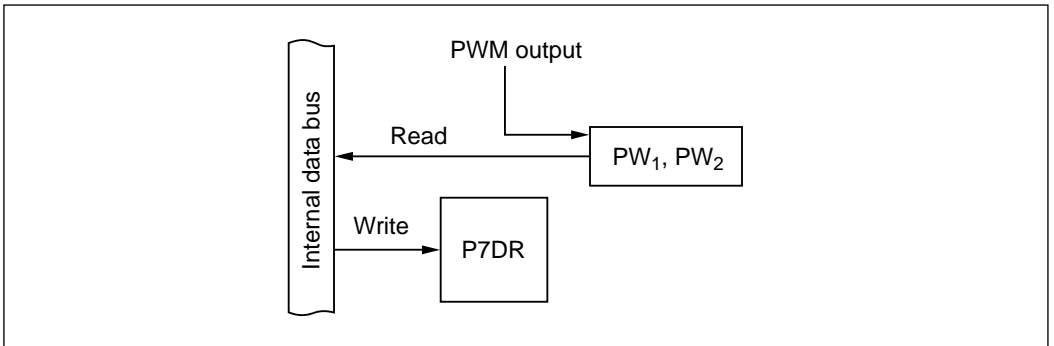
**Figure 10-51 SCK<sub>2</sub> and SCK<sub>1</sub> Pins (Modes 1 to 7)**

**Table 10-41 Register Read/Write Data**

	Read	Write
P7DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

**(8) PW<sub>2</sub> and PW<sub>1</sub> Output (P7<sub>7</sub> and P7<sub>6</sub>: Modes 1 to 7):** Figure 10-52 shows a block diagram illustrating the PWM output function. Table 10-42 indicates register read/write data. When P7<sub>7</sub> and P7<sub>6</sub> function as PW<sub>2</sub> and PW<sub>1</sub>, data written in the port 7 data register (P7DR) is not output at the pins, but P7DR can be read to monitor the levels of the PW<sub>2</sub> and PW<sub>1</sub> pins.



**Figure 10-52 PW<sub>2</sub> and PW<sub>1</sub> Output (Modes 1 to 7)**

**Table 10-42 Register Read/Write Data**

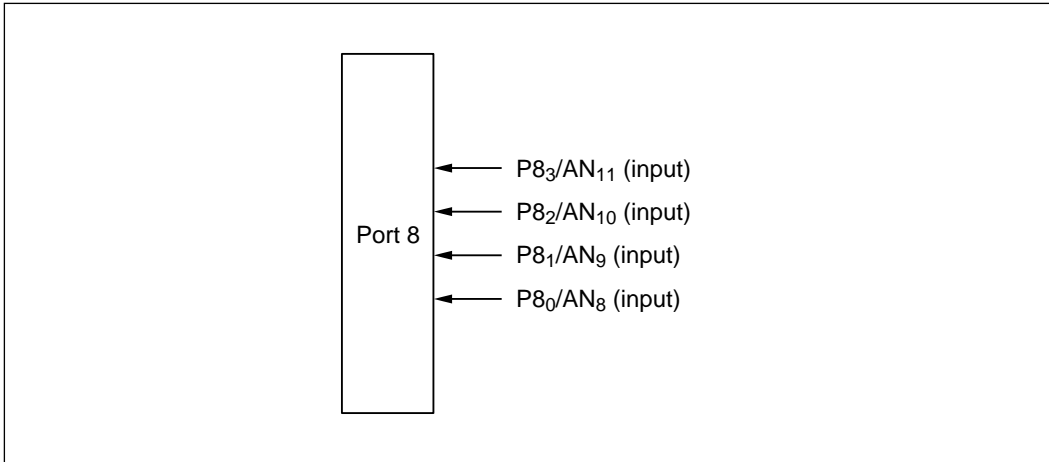
	Read	Write
P7DR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

## 10.9 Port 8

### 10.9.1 Overview

Port 8 is a four-bit input port that is multiplexed with analog input pins of the A/D converter. Figure 10-53 summarizes the pin functions.



**Figure 10-53 Port 8 Pin Functions**

### 10.9.2 Register Descriptions

Table 10-43 summarizes the registers of port 8. Since port 8 is used only for input, there is no data direction register.

**Table 10-43 Port 8 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE8F	Port 8 data register	P8DR	R	Undetermined

**(1) Port 8 Data Register:** The port 8 data register (P8DR) is an eight-bit register that indicates the values of pins P8<sub>3</sub> to P8<sub>0</sub>.

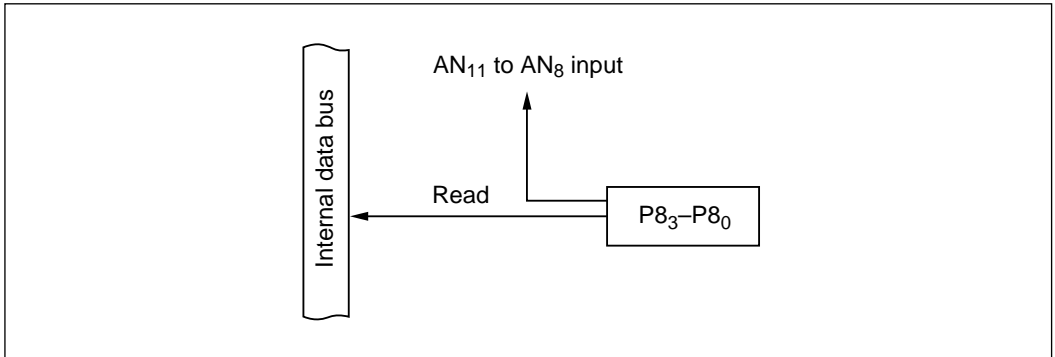
Bit	7	6	5	4	3	2	1	0
	—	—	—	—	P8 <sub>3</sub>	P8 <sub>2</sub>	P8 <sub>1</sub>	P8 <sub>0</sub>
Initial value	1	1	1	1	—	—	—	—
R/W	—	—	—	—	R	R	R	R

P8DR is a read-only register. It cannot be written. The upper four bits of P8DR are reserved bits that always return the value 1 when read.

### 10.9.3 Port 8 Read Operation

Figure 10-54 shows a block diagram of port 8.

While being used for analog input, port 8 can also function as a general-purpose input port. When read, P8DR returns the values at the pins. If P8DR is read when the A/D converter is sampling an analog input, however, the pin being sampled is read as 1.

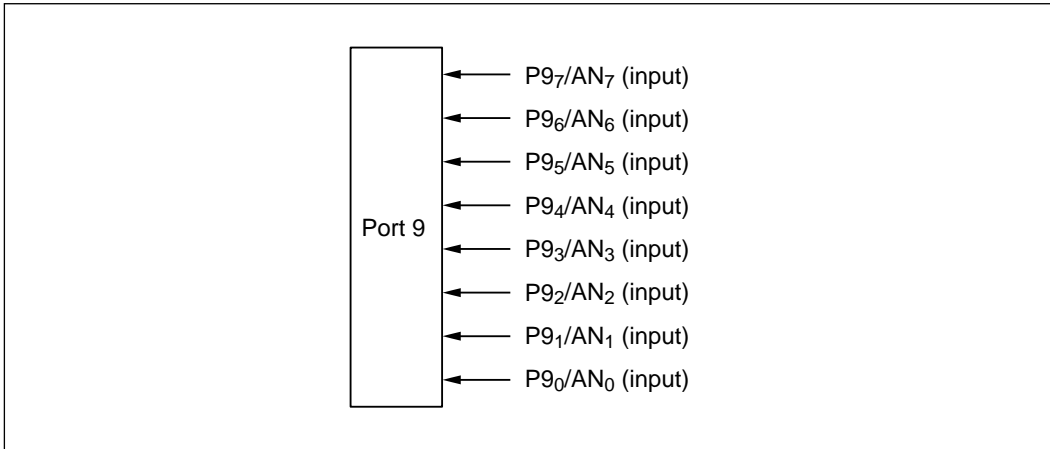


**Figure 10-54 Analog Input and General-Purpose Input (Modes 1 to 7)**

## 10.10 Port 9

### 10.10.1 Overview

Port 9 is an eight-bit input port that is multiplexed with analog input pins of the A/D converter. Figure 10-55 summarizes the pin functions.



**Figure 10-55 Port 9 Pin Functions**

### 10.10.2 Register Descriptions

Table 10-44 summarizes the registers of port 9. Since port 9 is used only for input, there is no data direction register.

**Table 10-44 Port 9 Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE92	Port 9 data register	P9DR	R	Undetermined

**(1) Port 9 Data Register:** The port 9 data register (P9DR) is an eight-bit register that indicates the values of pins P9<sub>7</sub> to P9<sub>0</sub>.

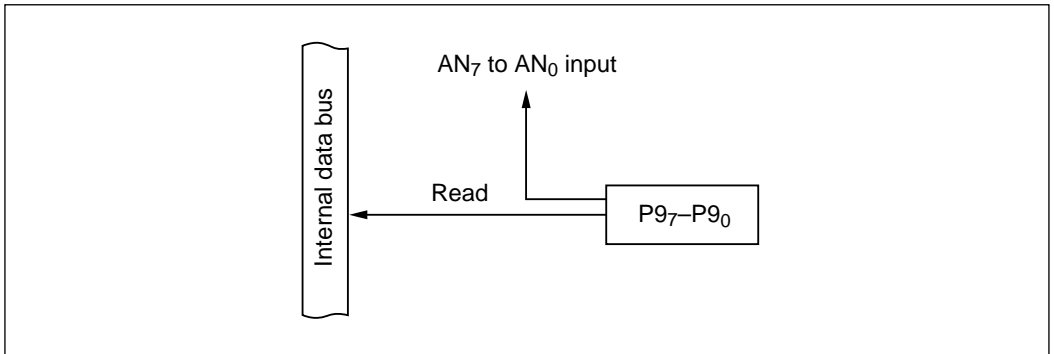
Bit	7	6	5	4	3	2	1	0
	P9 <sub>7</sub>	P9 <sub>6</sub>	P9 <sub>5</sub>	P9 <sub>4</sub>	P9 <sub>3</sub>	P9 <sub>2</sub>	P9 <sub>1</sub>	P9 <sub>0</sub>
Initial value	—	—	—	—	—	—	—	—
R/W	R	R	R	R	R	R	R	R

P9DR is a read-only register. It cannot be written.

### 10.10.3 Port 9 Read Operation

Figure 10-56 shows a block diagram of port 9.

While being used for analog input, port 9 can also function as a general-purpose input port. When read, P9DR returns the values at the pins. If P9DR is read when the A/D converter is sampling an analog input, however, the pin being sampled is read as 1.



**Figure 10-56** Analog Input and General-Purpose Input (Modes 1 to 7)

## 10.11 Port A

### 10.11.1 Overview

Port A is a seven-bit input/output port that is multiplexed with output compare pins ( $T5OC_{2/1}$ ,  $T4OC_{2/1}$ ,  $T3OC_{2/1}$ ) of the 16-bit integrated-timer pulse unit (IPU), pins for the  $\overline{BREQ}$ ,  $\overline{BACK}$ , and  $\overline{WAIT}$  signals, PWM timer output pins ( $PW_{1/2/3}$ ), serial communication interface 3 input and output pins ( $TXD_3$ ,  $RXD_3$ ,  $SCK_3$ ), and the page address bus ( $A_{19}$  to  $A_{16}$ ). Figure 10-57 summarizes the pin functions. Pins in port A can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair.

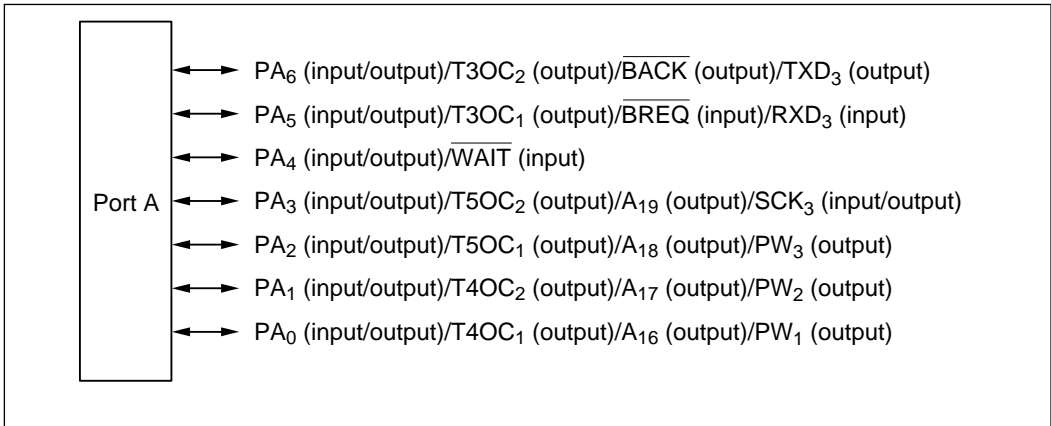


Figure 10-57 Port A Pin Functions

Figure 10-58 shows examples of output loads for port A.

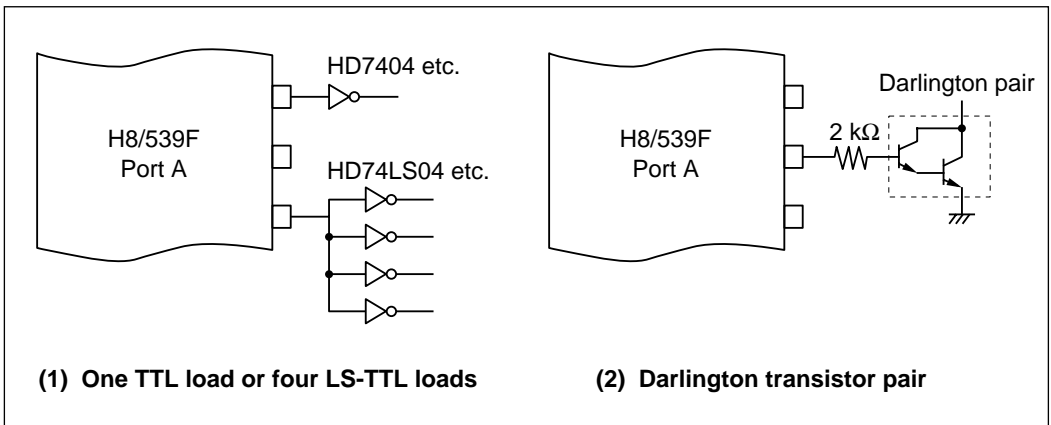


Figure 10-58 Examples of Port A Output Loads



## 10.11.2 Register Descriptions

Table 10-45 summarizes the registers of port A.

**Table 10-45 Port A Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE91	Port A data direction register	PADDR	W	H'80
H'FE93	Port A data register	PADR	R/W	H'80
H'FEDA	Port A control register	PACR	R/W	H'90

**(1) Port A Data Direction Register:** The port A data direction register (PADDR) is an eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	—	PA <sub>6</sub> DDR	PA <sub>5</sub> DDR	PA <sub>4</sub> DDR	PA <sub>3</sub> DDR	PA <sub>2</sub> DDR	PA <sub>1</sub> DDR	PA <sub>0</sub> DDR
Initial value	1	0	0	0	0	0	0	0
R/W	—	W	W	W	W	W	W	W

A pin in port A becomes an output pin if the corresponding PADDR bit is set to 1, and an input pin if this bit is cleared to 0. PADDR is a write-only register. All bits always return the value 1 when read.

PADDR is initialized to H'80 by a reset and in hardware standby mode. PADDR is not initialized in software standby mode.

**(2) Port A Data Register:** The port A data register (PADR) is an eight-bit register that stores data for pins PA<sub>6</sub> to PA<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	—	PA <sub>6</sub>	PA <sub>5</sub>	PA <sub>4</sub>	PA <sub>3</sub>	PA <sub>2</sub>	PA <sub>1</sub>	PA <sub>0</sub>
Initial value	1	0	0	0	0	0	0	0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in PADDR is set to 1, the corresponding PADR bit value is output at the corresponding pin. If port A is read the value in PADR is returned, regardless of the actual state of the pin.

When a bit in PADDR is cleared to 0, it is possible to write to the corresponding PADR bit but the value is not output at the pin. If PADR is read the value at the pin is returned, regardless of the value written in PADR.

PADR is initialized to H'80 by a reset and in hardware standby mode. PADR is not initialized in software standby mode.

**(3) Port A Control Register:** The port A control register (PACR) is an eight-bit register that controls the functions of pin PA<sub>6</sub> to PA<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	—	TXD3E	RXD3E	—	SCK3E	PW3E	PW2E	PW1E
Initial value	1	0	0	1	0	0	0	0
R/W	R	R/W	R/W	R	R/W	R/W	R/W	R/W

**Bits 6, 5, and 3—TXD<sub>3</sub> Enable, RXD<sub>3</sub> Enable, and SCK<sub>3</sub> Enable (TXD3E, RXD3E, SCK3E):** These bits control the TXD<sub>3</sub>, RXD<sub>3</sub>, and SCK<sub>3</sub> functions of pins PA<sub>6</sub>/T3OC<sub>2</sub>/ $\overline{\text{BACK}}$ /TXD<sub>3</sub>, PA<sub>5</sub>/T3OC<sub>1</sub>/ $\overline{\text{BREQ}}$ /RXD<sub>3</sub>, and PA<sub>3</sub>/T5OC<sub>2</sub>/A<sub>19</sub>/SCK<sub>3</sub> in port A. When bits TXD3E, RXD3E, and SCK3E are set to 1, pins PA<sub>6</sub>, PA<sub>5</sub>, and PA<sub>3</sub> can be used for TXD<sub>3</sub> output, RXD<sub>3</sub> input, and SCK<sub>3</sub> input or output.

**Bits 2 to 0—PW<sub>3</sub> Enable, PW<sub>2</sub> Enable, and PW<sub>1</sub> Enable (PW3E, PW2E, PW1E):** These bits control the PW<sub>3/2/1</sub> functions of pins PA<sub>2</sub>/T5OC<sub>1</sub>/A<sub>18</sub>/PW<sub>3</sub>, PA<sub>1</sub>/T4OC<sub>2</sub>/A<sub>17</sub>/PW<sub>2</sub>, and PA<sub>0</sub>/T4OC<sub>1</sub>/A<sub>16</sub>/PW<sub>1</sub> in port A. When bits PW3E, PW2E, and PW1E are set to 1, these pins can be used for PW<sub>3</sub> output, PW<sub>2</sub> output, and PW<sub>1</sub> output.

### 10.11.3 Pin Functions in Each Mode

Port A has different functions in different operating modes. A description for each mode is given next.

**(1) Pin Functions in Modes 1, 2, and 6:** Port A can be used for the output-compare function (T3OC<sub>2/1</sub>, T4OC<sub>2/1</sub>, T5OC<sub>2/1</sub>) of the 16-bit integrated-timer pulse unit (IPU), bus control ( $\overline{\text{BREQ}}$  and  $\overline{\text{BACK}}$ ), serial communication interface 3 input and output (SCK<sub>3</sub>, TXD<sub>3</sub>, RXD<sub>3</sub>), PWM timer output (PW<sub>3</sub>, PW<sub>2</sub>, PW<sub>1</sub>), wait signal input (WAIT), and general-purpose output.

When a pin is used for output compare, bus control, serial communication interface 3 input or output, PWM timer output, or wait signal input, the PADDR setting is ignored.

The priority of pin functions for PA<sub>5</sub>/T3OC<sub>1</sub>/ $\overline{\text{BREQ}}$ /RXD<sub>3</sub> and PA<sub>6</sub>/T3OC<sub>2</sub>/ $\overline{\text{BACK}}$ /TXD<sub>3</sub> is:

Bus control > TXD<sub>3</sub>, RXD<sub>3</sub> > output compare > general-purpose output

The TXD<sub>3</sub> and RXD<sub>3</sub> pin functions are available when bits TXD3E and RXD3E are set to 1 in the port A control register (PACR). When these bits are set to 1, the corresponding pins cannot be used for output compare.

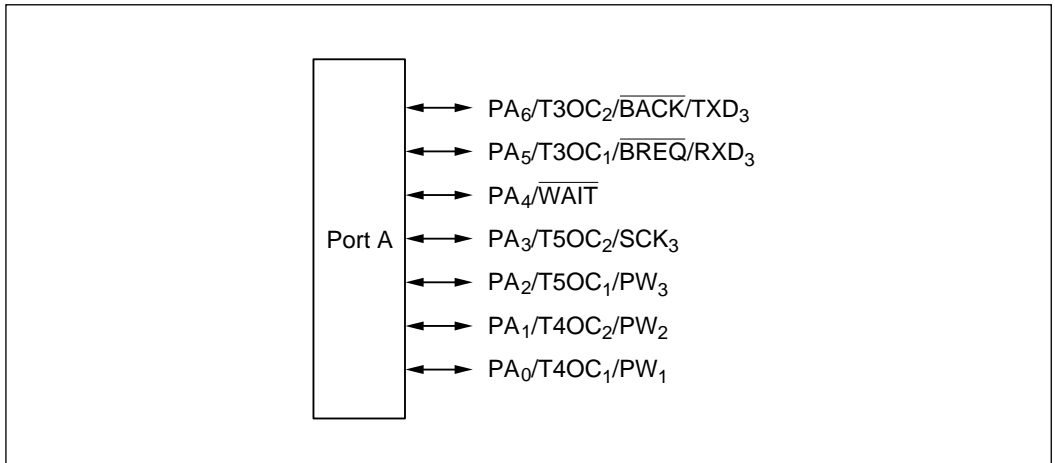
The priority of pin functions for PA<sub>3</sub>/T5OC<sub>2</sub>/SCK<sub>3</sub>, PA<sub>2</sub>/T5OC<sub>1</sub>/PW<sub>3</sub>, PA<sub>1</sub>/T4OC<sub>2</sub>/PW<sub>2</sub>, and PA<sub>0</sub>/T4OC<sub>1</sub>/PW<sub>1</sub> is:

PWM output pins, serial communication interface pins > output compare output pins > output portst

The SCK<sub>3</sub>, PW<sub>3</sub>, PW<sub>2</sub>, and PW<sub>1</sub> pin functions are available when bits SCK3E, PW3E, PW2E, and PW1E, respectively, are set to 1 in PACR. When these bits are set to 1, the corresponding pins cannot be used as output compare pins.

For methods of selecting pin functions, see appendix D “Pin Function Selection.”

Figure 10-59 shows the functions of port A in modes 1, 2, and 6.



**Figure 10-59 Port A Pin Functions in Modes 1, 2, and 6**

**(2) Pin Functions in Modes 3 and 5:** Port A has pins that can be used for the output compare function (T3OC<sub>2/1</sub>) of the 16-bit integrated-timer pulse unit (IPU), bus control ( $\overline{\text{BREQ}}$  and  $\overline{\text{BACK}}$ ), serial communication interface 3 input and output (TXD<sub>3</sub>, RXD<sub>3</sub>), wait signal input ( $\overline{\text{WAIT}}$ ), or general-purpose input or output, and pins that are used for page address output (A<sub>19</sub> to A<sub>16</sub>). When a pin is used for output compare, bus control, serial communication input/output, or wait signal input, the PADDR setting is ignored.

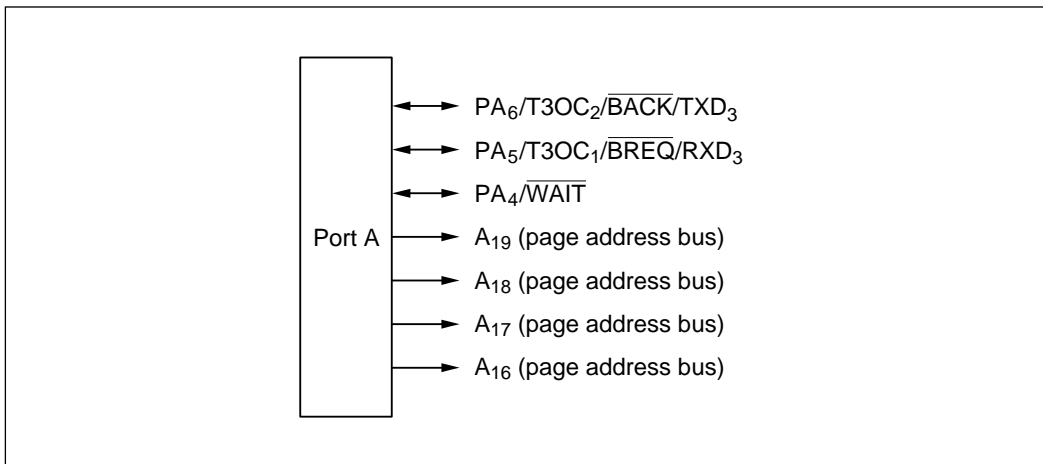
The priority of pin functions for PA<sub>5</sub>/T3OC<sub>1</sub>/ $\overline{\text{BREQ}}$ /RXD<sub>3</sub> and PA<sub>6</sub>/T3OC<sub>2</sub>/ $\overline{\text{BACK}}$ /TXD<sub>3</sub> is:

TXD<sub>3</sub>, RXD<sub>3</sub> > bus control > output compare > general-purpose output

The TXD<sub>3</sub> and RXD<sub>3</sub> pin functions are available when bits TXD3E and RXD3E are set to 1 in the port A control register (PACR). When these bits are set to 1, the corresponding pins cannot be used for output compare.

For methods of selecting pin functions, see appendix D “Pin Function Selection.”

Figure 10-60 shows the functions of port A in modes 3 and 5.



**Figure 10-60 Port A Pin Functions in Modes 3 and 5**

**(3) Pin Functions in Mode 4:** Port A has pins that can be used for the output compare function ( $T3OC_{2/1}$ ) of the 16-bit integrated-timer pulse unit (IPU), bus control ( $\overline{BREQ}$  and  $\overline{BACK}$ ), serial communication interface 3 input and output ( $SCK_3$ ,  $TXD_3$ ,  $RXD_3$ ), PWM timer output ( $PW_1$ ,  $PW_2$ ,  $PW_3$ ), wait signal input ( $\overline{WAIT}$ ), page address output ( $A_{19}$  to  $A_{16}$ ), and general-purpose input or output. When a pin is used for output compare, bus control, serial communication input/output, PWM timer output, or wait signal input, the PADDR setting is ignored.

The  $SCK_3$ ,  $PW_3$ ,  $PW_2$ , and  $PW_1$  functions of pins  $PA_3$  to  $PA_0$  are available when bits  $SCK3E$ ,  $PW3E$ ,  $PW2E$ , and  $PW1E$  are set to 1 in the port A control register (PACR). When these bits are set to 1, the corresponding pins cannot be used for page address output. When bits  $SCK3E$ ,  $PW3E$ ,  $PW2E$ , and  $PW1E$  are cleared to 0 in PACR, these pins are used for page address output if the corresponding PADDR bit is set to 1, and for general-purpose input if the corresponding PADDR bit is cleared to 0.

The priority of pin functions for  $PA_5/T3OC_1/\overline{BREQ}/RXD_3$  and  $PA_6/T3OC_2/\overline{BACK}/TXD_3$  is:

Bus control >  $TXD_3$ ,  $RXD_3$  > output compare > general-purpose output

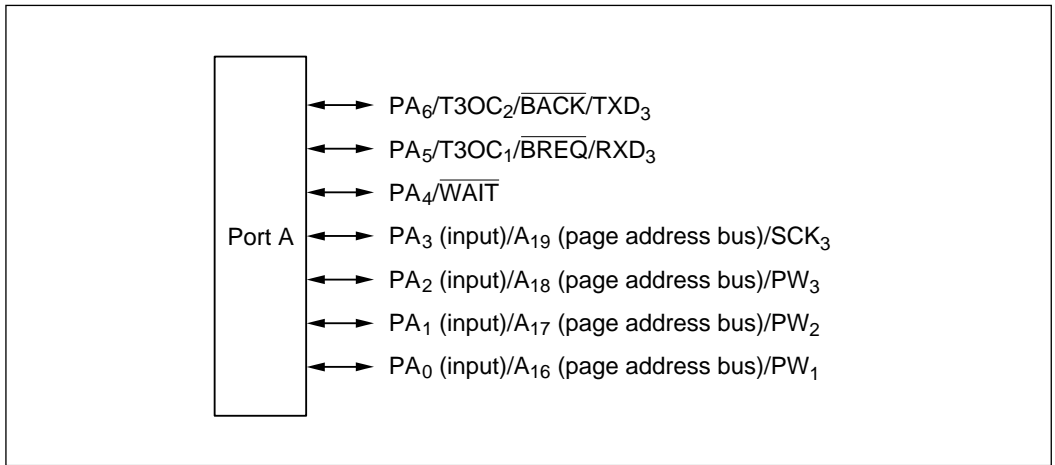
The  $TXD_3$  and  $RXD_3$  pin functions are available when bits  $TXD3E$  and  $RXD3E$  are set to 1 in the port A control register (PACR). When these bits are set to 1, the corresponding pins cannot be used for output compare.

The priority of pin functions for  $PA_3/A_{19}/SCK_3$ ,  $PA_2/A_{18}/PW_3$ ,  $PA_1/A_{17}/PW_2$ , and  $PA_0/A_{16}/PW_1$  is:

$SCK_3, PW_{3/2/1} > \text{address bus} > \text{general-purpose input}$

For methods of selecting pin functions, see appendix D “Pin Function Selection.”

Figure 10-61 shows the functions of port A in mode 4.



**Figure 10-61 Port A Pin Functions in Mode 4**

**(4) Pin Functions in Mode 7:** Port A can be used for the output compare function (T3OC<sub>2/1</sub>, T4OC<sub>2/1</sub>, T5OC<sub>2/1</sub>) of the 16-bit integrated-timer pulse unit (IPU), serial communication interface 3 input and output (SCK<sub>3</sub>, TXD<sub>3</sub>, RXD<sub>3</sub>), PWM timer output (PW<sub>1</sub>, PW<sub>2</sub>, PW<sub>3</sub>), and general-purpose input or output. When a pin is used for serial communication interface 3 input or output, PWM timer output, or output compare, the PADDR setting is ignored.

The priority of pin functions for PA<sub>6</sub>/T3OC<sub>2</sub>/TXD<sub>3</sub> and PA<sub>5</sub>/T3OC<sub>1</sub>/RXD<sub>3</sub> is:

$TXD_3, RXD_3 > \text{output compare} > \text{general-purpose output}$

The TXD<sub>3</sub> and RXD<sub>3</sub> pin functions are available when bits TXD3E and RXD3E are set to 1 in the port A control register (PACR). When these bits are set to 1, the corresponding pins cannot be used for output compare.

The priority of pin functions for PA<sub>3</sub>/T5OC<sub>2</sub>/SCK<sub>3</sub>, PA<sub>2</sub>/T5OC<sub>1</sub>/PW<sub>3</sub>, PA<sub>1</sub>/T4OC<sub>2</sub>/PW<sub>2</sub>, and PA<sub>0</sub>/T4OC<sub>1</sub>/PW<sub>1</sub> is:

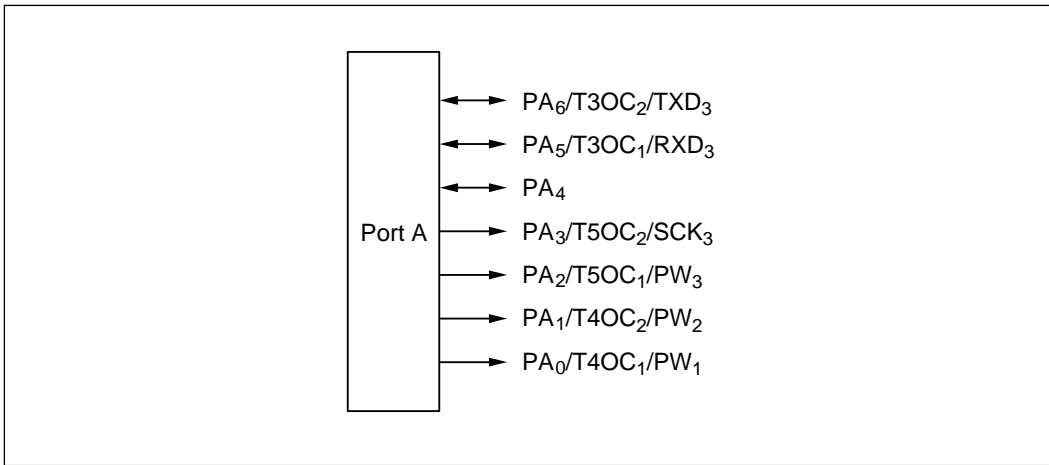
$\text{PWM output pins, serial communication interface pins} > \text{output compare output pins} > \text{output ports}$

The SCK<sub>3</sub>, PW<sub>3</sub>, PW<sub>2</sub>, and PW<sub>1</sub> pin functions are available when bits SCK3E, PW3E, PW2E, and PW1E, respectively, are set to 1 in PACR. When these bits are set to 1, these pins cannot be used

as output compare pins.

For methods of selecting pin functions, see appendix D “Pin Function Selection.”

Figure 10-62 shows the functions of port A in mode 7.

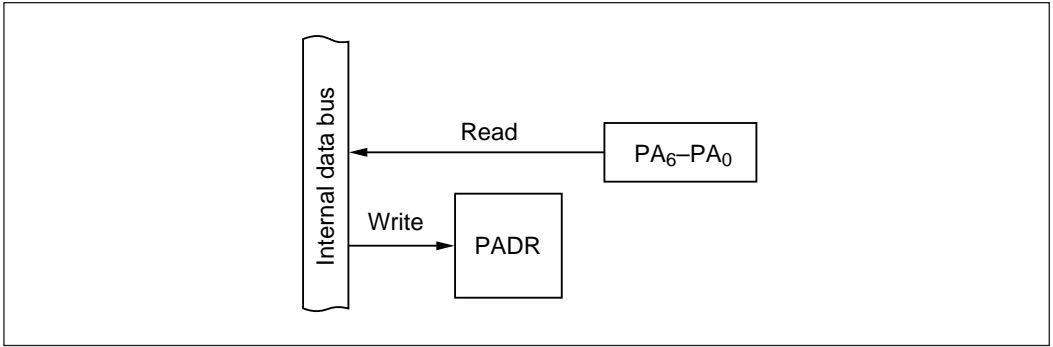


**Figure 10-62 Port A Pin Functions in Mode 7**

#### 10.11.4 Port A Read/Write Operations

PADR and PADDR have different read/write functions depending on whether port A is used for bus control ( $\overline{\text{BREQ}}$ ,  $\overline{\text{BACK}}$ ), wait signal input ( $\overline{\text{WAIT}}$ ), the output compare function (T5OC<sub>2/1</sub>, T4OC<sub>2/1</sub>, T3OC<sub>2/1</sub>) of the 16-bit integrated-timer pulse unit (IPU), serial communication interface 3 input or output (SCK<sub>3</sub>, TXD<sub>3</sub>, RXD<sub>3</sub>), or general-purpose input or output. The operating states and functions of port A are described next.

**(1) Input Port (PA<sub>6</sub> to PA<sub>4</sub> in Modes 1 to 7; PA<sub>3</sub> to PA<sub>0</sub> in Modes 1, 2, 4, 6, and 7):** Figure 10-63 shows a block diagram illustrating the general-purpose input function. Table 10-46 indicates register read/write data. Values written in the port A data register (PADR) have no effect on general-purpose input lines. When read, PADR returns the value at the pin.



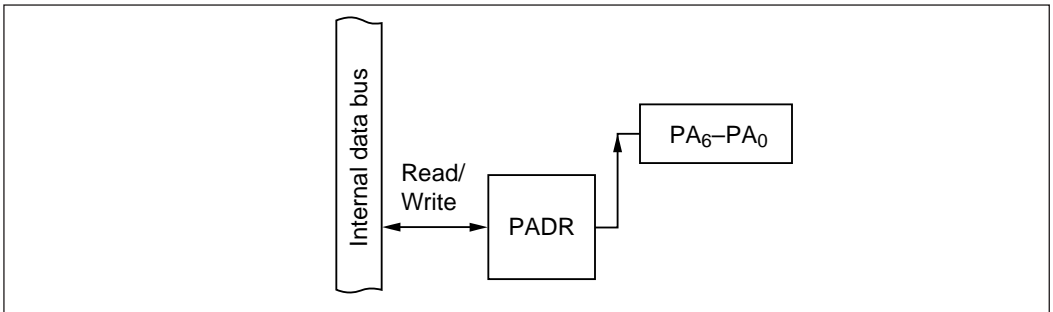
**Figure 10-63 Input Port (Modes 1 to 7)**

**Table 10-46 Register Read/Write Data**

	Read	Write
PADR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

(2) **Output Port (PA<sub>6</sub> to PA<sub>4</sub> in Modes 1 to 7; PA<sub>3</sub> to PA<sub>0</sub> in Modes 1, 2, 4, 6, and 7):** Figure 10-64 shows a block diagram illustrating the general-purpose output function. Table 10-47 indicates register read/write data. The value written in the port A data register (PADR) is output at the pin. When read, PADR returns the value written in PADR.

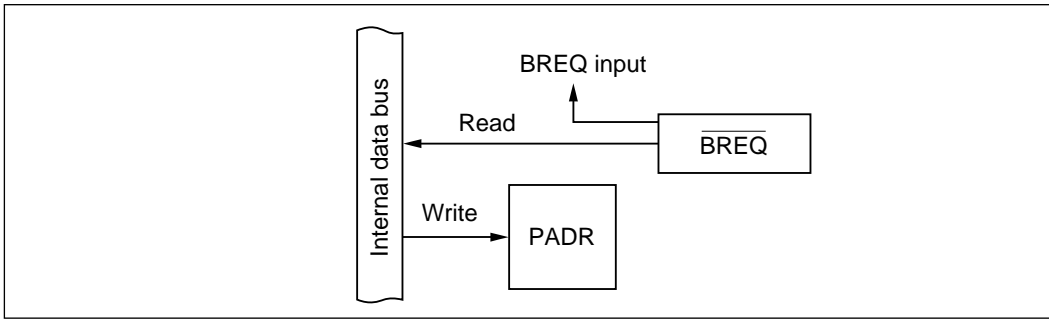


**Figure 10-64 Output Port (Modes 1 to 7)**

**Table 10-47 Register Read/Write Data**

	Read	Write
PADR	PADR value	Value output at pin

(3)  **$\overline{\text{BREQ}}$  Pin (PA<sub>5</sub>: Modes 1 to 6):** Figure 10-65 shows a block diagram illustrating the  $\overline{\text{BREQ}}$  function. Table 10-48 indicates register read/write data. When PA<sub>5</sub> is used for  $\overline{\text{BREQ}}$  input, the value written in the port A data register (PADR) has no effect.



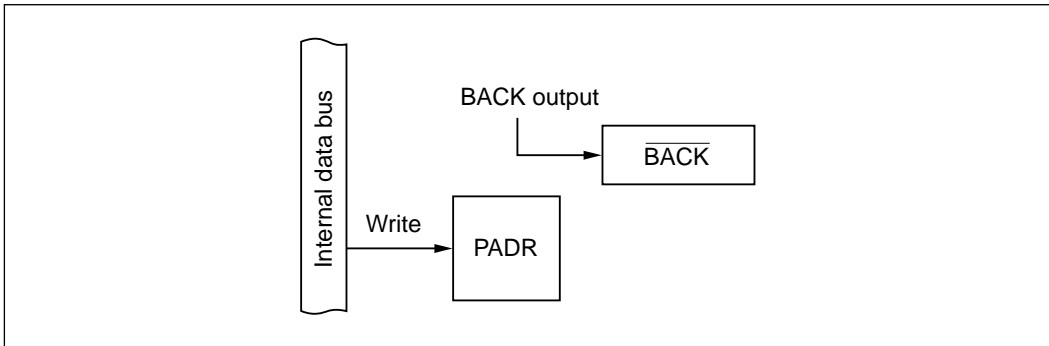
**Figure 10-65**  $\overline{\text{BREQ}}$  Input Pin (Modes 1 to 6)

**Table 10-48** Register Read/Write Data

	Read	Write
PADR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

(4)  **$\overline{\text{BACK}}$  Pin ( $\text{PA}_6$ : Modes 1 to 6):** Figure 10-66 shows a block diagram illustrating the  $\overline{\text{BACK}}$  function. Table 10-49 indicates register read/write data. When  $\text{PA}_6$  is used for  $\overline{\text{BACK}}$  output, the value written in the port A data register (PADR) has no effect. When read, PADR returns an undetermined value.



**Figure 10-66**  $\overline{\text{BACK}}$  Output Pin (Modes 1 to 6)

**Table 10-49** Register Read/Write Data

	Read	Write
PADR	Undetermined value	Don't care*

Note: The register can be written to, but the value is not output at the pines.



(5) **WAIT Pin (PA<sub>4</sub>: Modes 1 to 6):** Figure 10-67 shows a block diagram illustrating the  $\overline{\text{WAIT}}$  function. Table 10-50 indicates register read/write data. When PA<sub>4</sub> is used for  $\overline{\text{WAIT}}$  input, the value written in the port A data register (PADR) has no effect.

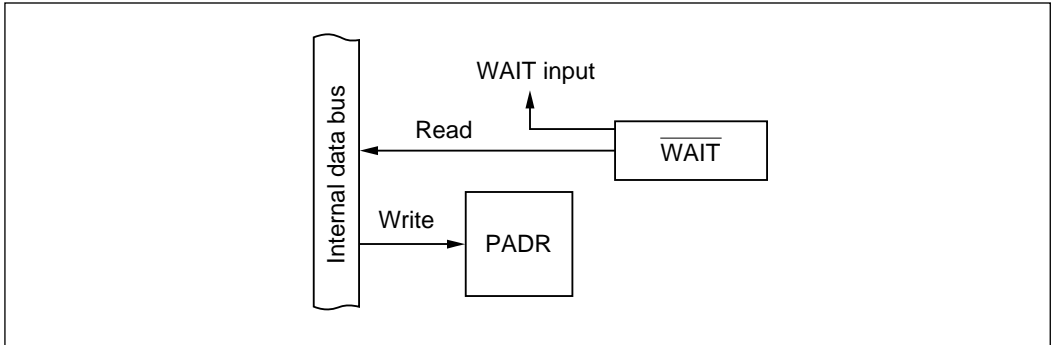


Figure 10-67  $\overline{\text{WAIT}}$  Input Pin (Modes 1 to 6)

Table 10-50 Register Read/Write Data

	Read	Write
PADR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

(6) **Timer Output Pins (PA<sub>6</sub>, PA<sub>5</sub>, PA<sub>3</sub> to PA<sub>0</sub>: Modes 1 to 7):** Figure 10-68 shows a block diagram illustrating the timer output function. Table 10-51 indicates register read/write data. When PA<sub>6</sub>, PA<sub>5</sub>, and PA<sub>3</sub> to PA<sub>0</sub> are used for T3OC<sub>2</sub>, T3OC<sub>1</sub>, T5OC<sub>2</sub>, T5OC<sub>1</sub>, T4OC<sub>2</sub>, and T4OC<sub>1</sub> output, values written in the port A data register (PADR) have no effect on the timer output. PADR can be read to monitor the timer output level (T3OC<sub>2</sub>, T3OC<sub>1</sub>, T5OC<sub>2</sub>, T5OC<sub>1</sub>, T4OC<sub>2</sub>, T4OC<sub>1</sub>).

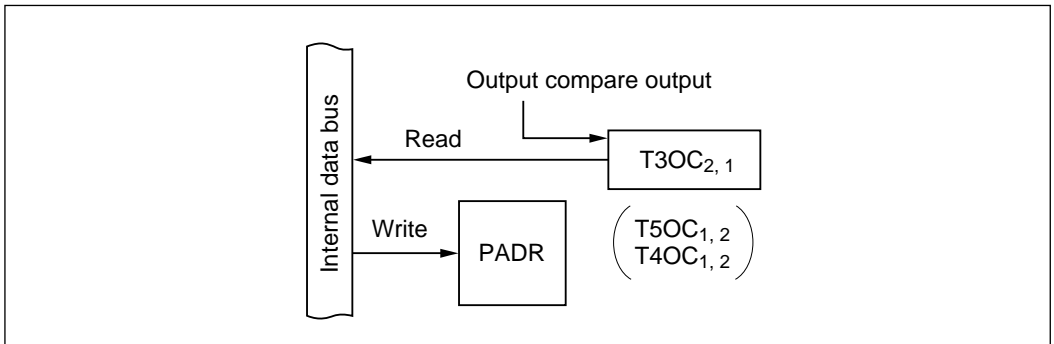


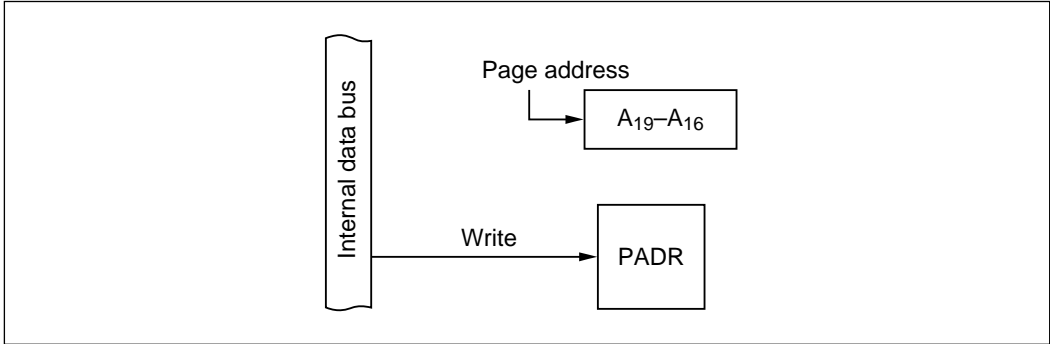
Figure 10-68 Output Compare Pins (Modes 1 to 7)

**Table 10-51 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
PADR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(7) Page Address Bus (PA<sub>3</sub> to PA<sub>0</sub>: Modes 3 to 5):** Figure 10-69 shows a block diagram illustrating the page-address-bus function. Table 10-52 indicates register read/write data. When PA<sub>3</sub> to PA<sub>0</sub> are used for A<sub>19</sub> to A<sub>16</sub> output, values written in the port A data register (PADR) have no effect. When read, PADR returns an undetermined value.



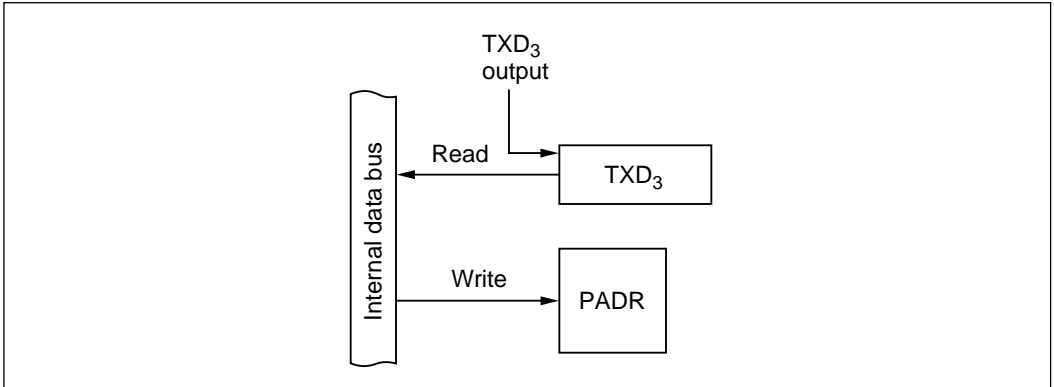
**Figure 10-69 Page Address Bus (Modes 3 to 5)**

**Table 10-52 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
PADR	Undetermined value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(8) TXD<sub>3</sub> Output (PA<sub>6</sub>: Modes 1, 2, 4, 6, and 7):** Figure 10-70 shows a block diagram illustrating the TXD<sub>3</sub> output function. Table 10-53 indicates register read/write data. When PA<sub>6</sub> is used for TXD<sub>3</sub> output, the value written in PADR is ignored, but PADR can be read to monitor the level at the TXD<sub>3</sub> pin.



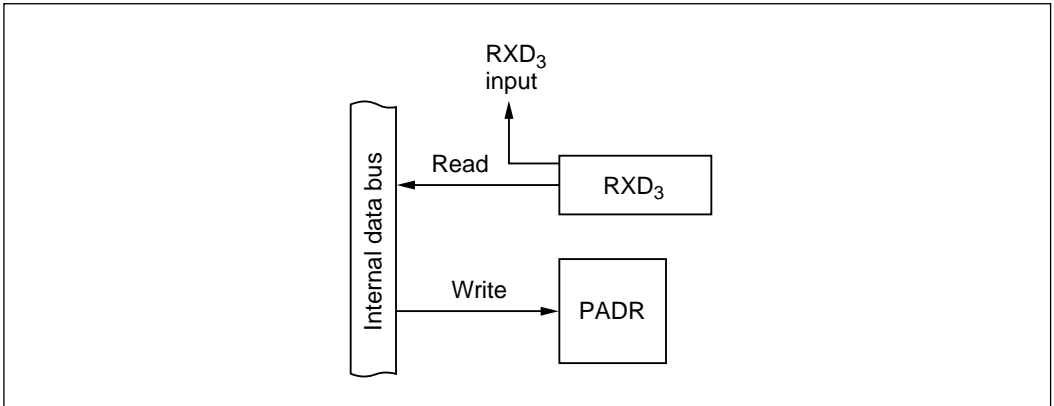
**Figure 10-70 TXD<sub>3</sub> Output (Modes 1, 2, 4, 6, and 7)**

**Table 10-53 Register Read/Write Data**

	Read	Write
PADR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(9) RXD<sub>3</sub> Input (PA<sub>5</sub>; Modes 1, 2, 4, 6, and 7):** Figure 10-71 shows a block diagram illustrating the RXD<sub>3</sub> input function. Table 10-54 indicates register read/write data. When PA<sub>5</sub> is used for RXD<sub>3</sub> input, the value written in PADR is ignored, but PADR can be read to monitor the level at the RXD<sub>3</sub> pin.



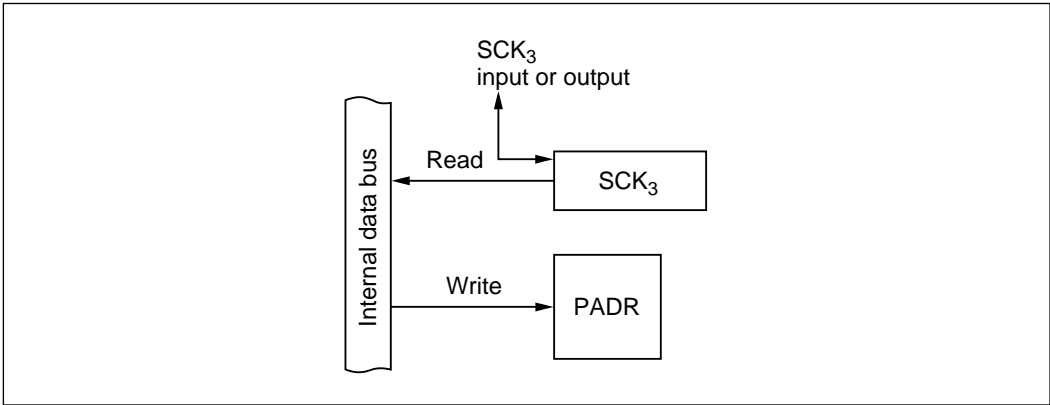
**Figure 10-71 RXD<sub>3</sub> Input (Modes 1, 2, 4, 6, and 7)**

**Table 10-54 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
PADR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

**(10) SCK<sub>3</sub> Pin (PA<sub>3</sub>: Modes 1, 2, 4, 6, and 7):** Figure 10-72 shows a block diagram illustrating the SCK<sub>3</sub> input/output function. Table 10-55 indicates register read/write data. When PA<sub>3</sub> is used for SCK<sub>3</sub> input or output, the value written in PADR is ignored, but PADR can be read to monitor the level at the SCK<sub>3</sub> pin.



**Figure 10-72 SCK<sub>3</sub> Pins (Modes 1, 2, 4, 6, and 7)**

**Table 10-55 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
PADR	Pin value	Don't care*

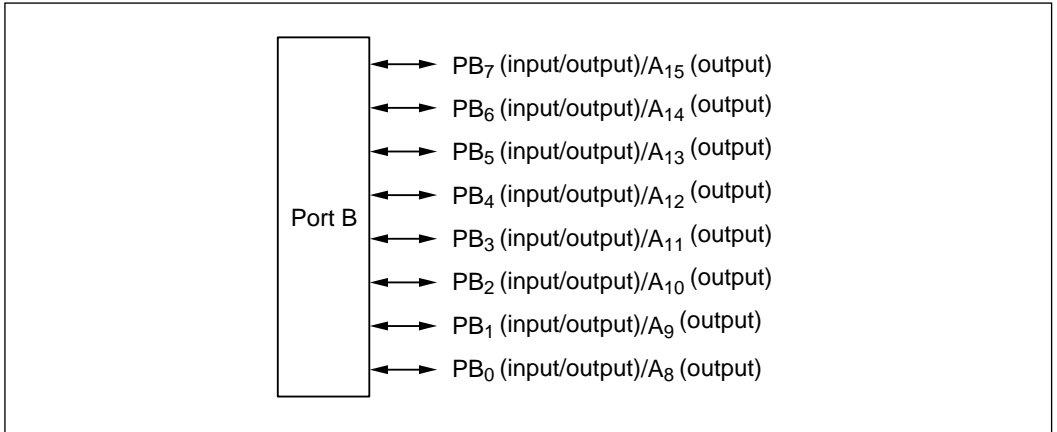
Note: The register can be written to, but the value is not output at the pines.

## 10.12 Port B

### 10.12.1 Overview

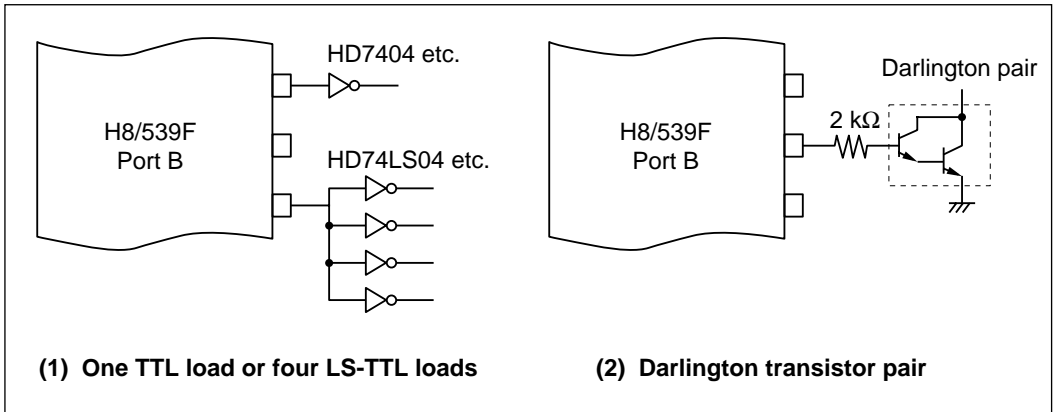
Port B is an eight-bit input/output port. Figure 10-73 summarizes the pin functions.

Pins in port B can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair. They have software-programmable built-in MOS pull-up transistors.



**Figure 10-73 Port B Pin Functions**

Figure 10-74 shows examples of output loads for port B.



**Figure 10-74 Examples of Port B Output Loads**

## 10.12.2 Register Descriptions

Table 10-56 summarizes the registers of port B.

**Table 10-56 Port B Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE94	Port B data direction register	PBDDR	W	H'00
H'FE96	Port B data register	PBDR	R/W	H'00
H'FE98	Port B pull-up transistor control register	PBPCR	R/W	H'00

**(1) Port B Data Direction Register:** The port B data direction register (PBDDR) is an-eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	PB <sub>7</sub> DDR	PB <sub>6</sub> DDR	PB <sub>5</sub> DDR	PB <sub>4</sub> DDR	PB <sub>3</sub> DDR	PB <sub>2</sub> DDR	PB <sub>1</sub> DDR	PB <sub>0</sub> DDR
Initial value	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W

A pin in port B becomes an output pin if the corresponding PBDDR bit is set to 1, and an input pin if this bit is cleared to 0. PBDDR is a write-only register. All bits always return the value 1 when read.

PBDDR is initialized to H'00 by a reset and in hardware standby mode. PBDDR is not initialized in software standby mode.

**(2) Port B Data Register:** The port B data register (PBDR) is an-eight-bit register that stores data for pins PB<sub>7</sub> to PB<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	PB <sub>7</sub>	PB <sub>6</sub>	PB <sub>5</sub>	PB <sub>4</sub>	PB <sub>3</sub>	PB <sub>2</sub>	PB <sub>1</sub>	PB <sub>0</sub>
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in PBDDR is set to 1, the corresponding PBDR bit value is output at the corresponding pin. If port B is read the value in PBDR is returned, regardless of the actual state of the pin.

When a bit in PBDDR is cleared to 0, it is possible to write to the corresponding PBDR bit but the value is not output at the pin. If PBDR is read the value at the pin is returned, regardless of the

value written in PBDR.

PBDR is initialized to H'00 by a reset and in hardware standby mode. PBDR is not initialized in software standby mode.

**(3) Port B Pull-Up Transistor Control Register:** The port B pull-up transistor control register (PBPCR) is an-eight-bit register that turns the MOS pull-up transistors of PB<sub>7</sub> to PB<sub>0</sub> on and off. PBPCR is ignored in modes 1 to 6 and used only in mode 7.

Bit	7	6	5	4	3	2	1	0
	PB <sub>7</sub> PON	PB <sub>6</sub> PON	PB <sub>5</sub> PON	PB <sub>4</sub> PON	PB <sub>3</sub> PON	PB <sub>2</sub> PON	PB <sub>1</sub> PON	PB <sub>0</sub> PON
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

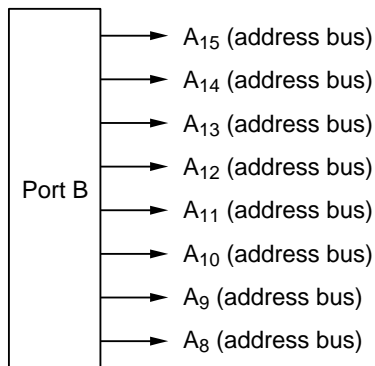
When a PBDDR bit is cleared to 0, if the corresponding PBPCR bit is set to 1, the built-in pull-up transistor is turned on.

PBPCR is initialized to H'00 by a reset and in hardware standby mode. PBPCR is not initialized in software standby mode.

### 10.12.3 Pin Functions in Each Mode

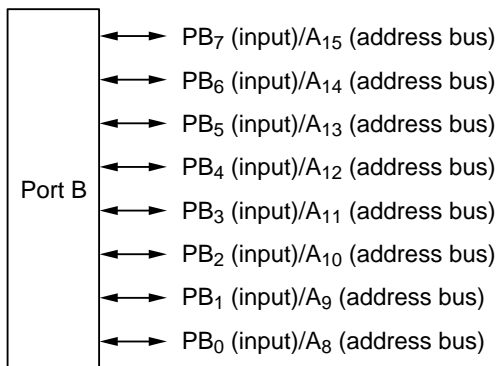
Port B has one set of functions in modes 1, 3, 5, and 6, another set of functions in modes 2 and 4, and another set of functions in mode 7. A description for each mode group is given next.

**(1) Pin Functions in Modes 1, 3, 5, and 6:** Port B is used for address output (A<sub>15</sub> to A<sub>8</sub>). The PBDDR settings are ignored. Figure 10-75 shows the pin functions in modes 1, 3, 5, and 6.



**Figure 10-75 Pin Functions in Modes 1, 3, 5, and 6**

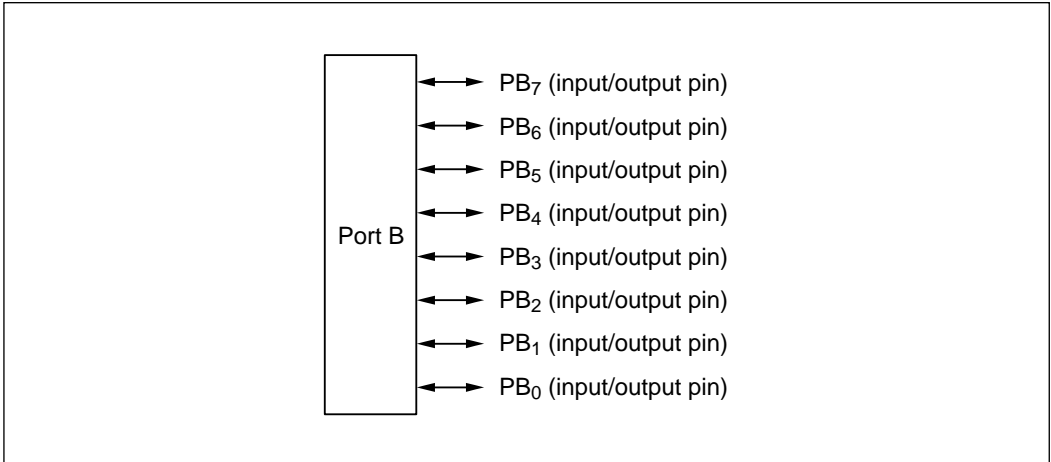
**(2) Pin Functions in Modes 2 and 4:** Port B can be used for address output ( $A_{15}$  to  $A_8$ ) or general-purpose input. A pin is used for address output if the corresponding PBDDR bit is set to 1, and for general-purpose input if this bit is cleared to 0. Figure 10-76 shows the pin functions in modes 2 and 4.



**Figure 10-76 Pin Functions in Modes 2 and 4**

**(3) Pin Functions in Mode 7:** Port B consists of general-purpose input/output pins. Input or output can be selected separately for each pin. A pin becomes an output pin if the corresponding PBDDR bit is set to 1 and an input pin if this bit is cleared to 0. Figure 10-77 shows the pin functions in mode 7.





**Figure 10-77 Pin Functions in Mode 7**

**10.12.4 Built-In Pull-Up Transistors**

Port B has built-in MOS pull-up transistors that can be controlled by software. To turn an input pull-up transistor on, clear its PBDDR bit to 0 and set its PBPCR bit to 1. The input pull-up transistors are turned off by a reset and in hardware standby mode. Table 10-57 summarizes the states of the input pull-ups in each mode.

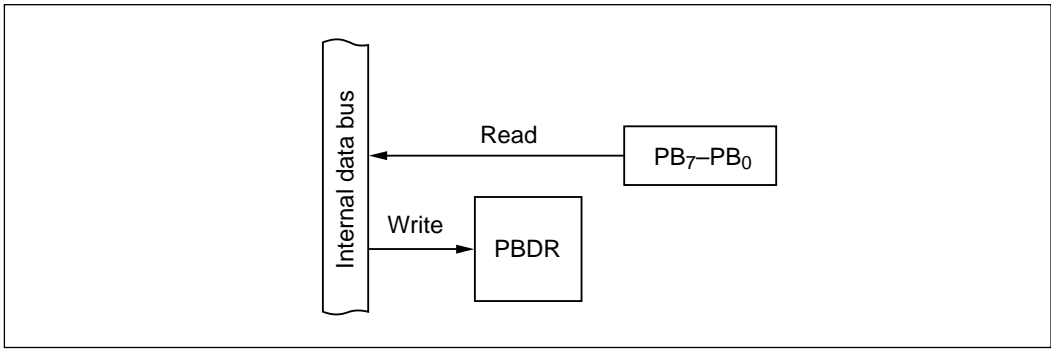
**Table 10-57 Pull-Up Transistor States in Each Mode**

Mode	Reset	Hardware Standby Mode	Other Modes (Including Software Standby Mode)
1–6	Off	Off	Off
7			On/Off

**10.12.5 Port B Read/Write Operations**

PBDR and PBDDR have different read/write functions depending on whether port B is used for address output (A<sub>15</sub> to A<sub>8</sub>) or general-purpose input or output. The operating states and functions of port B are described next.

**(1) Input Port (Modes 2 and 4):** Figure 10-78 shows a block diagram illustrating the general-purpose input function. Table 10-58 indicates register read/write data. Values written in the port B data register (PBDR) have no effect on general-purpose input lines. When read, PBDR returns the value at the pin.



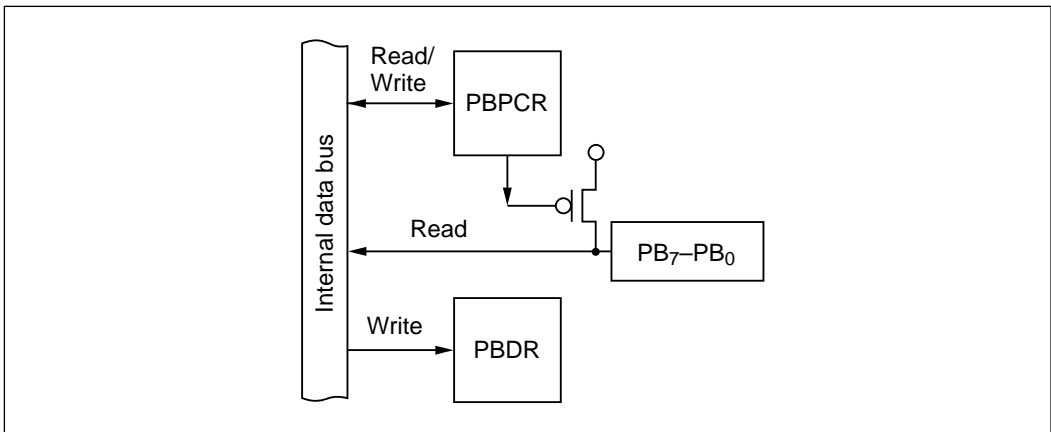
**Figure 10-78 Input Port (Modes 2 and 4)**

**Table 10-58 Register Read/Write Data**

	Read	Write
PBDR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

**(2) Input Port with Internal Pull-Up (Mode 7):** Figure 10-79 shows a block diagram illustrating the general-purpose input function and built-in input pull-up transistors. Table 10-59 indicates register read/write data. Values written in the port B data register (PBDR) have no effect on general-purpose input lines. When read, PBDR returns the value at the pin. When a bit in the port B pull-up transistor control register (PBPCR) is set to 1, the corresponding PBDR bit always reads 1.



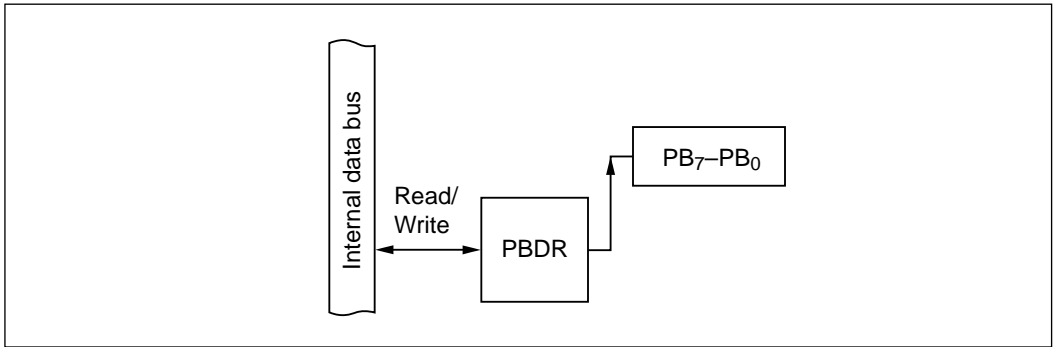
**Figure 10-79 Input Port with Built-In Pull-Up Transistors (Mode 7)**

**Table 10-59 Register Read/Write Data**

	Read	Write
PBDR	Pin value, or always 1*1	Don't care*2
PBPCR	PBPCR value	0/1*1

Note: \*1 If set to 1, the corresponding PBDR bit always reads 1.  
 \*2 The register can be written to, but the value is not output at the pines.

**(3) Output Port (Mode 7):** Figure 10-80 shows a block diagram illustrating the general-purpose output function. Table 10-60 indicates register read/write data. The value written in the port B data register (PBDR) is output at the pin. When read, PBDR returns the value written in PBDR.

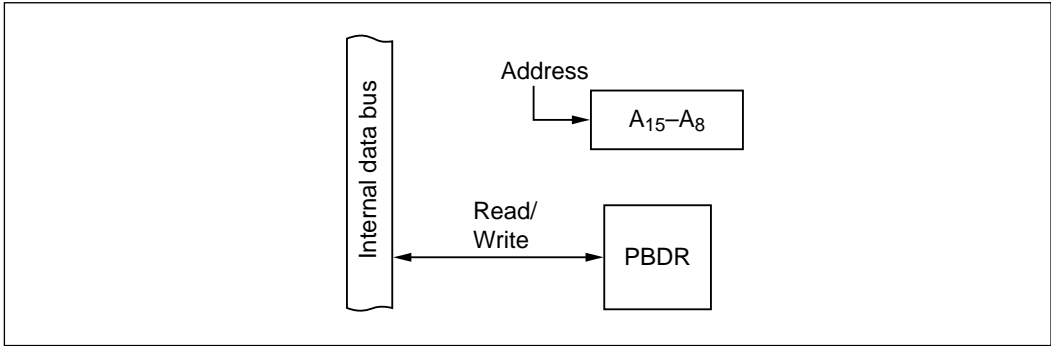


**Figure 10-80 Output Port (Mode 7)**

**Table 10-60 Register Read/Write Data**

	Read	Write
PBDR	PBDR value	Value output at pin

**(4) Address Bus (Modes 1 to 6):** Figure 10-81 shows a block diagram illustrating the address-bus function. Table 10-61 indicates register read/write data. When port B is used as an address bus, values written in the port B data register (PBDR) have no effect on the bus lines. When read, PBDR returns the value written in PBDR.



**Figure 10-81 Address Bus (Modes 1 to 6)**

**Table 10-61 Register Read/Write Data**

	Read	Write
PBDR	PBDR value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

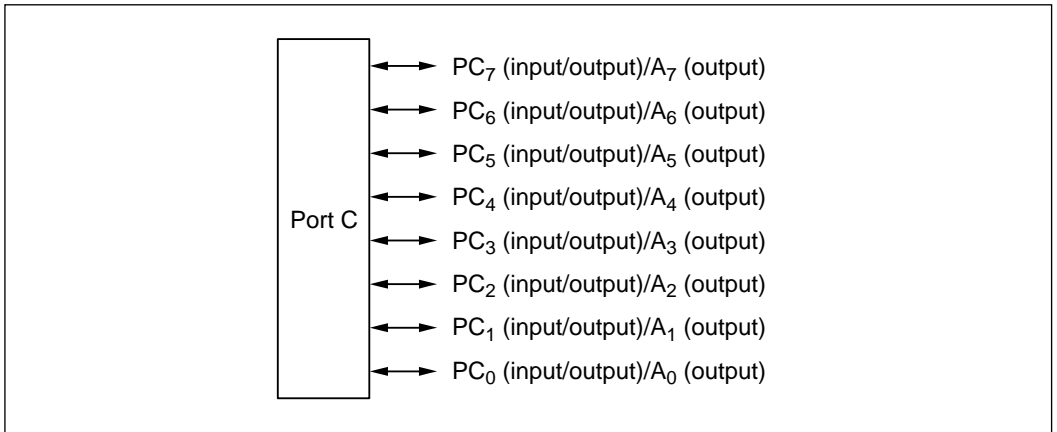
## 10.13 Port C

### 10.13.1 Overview

Port C is an eight-bit input/output port. Figure 10-82 summarizes the pin functions.

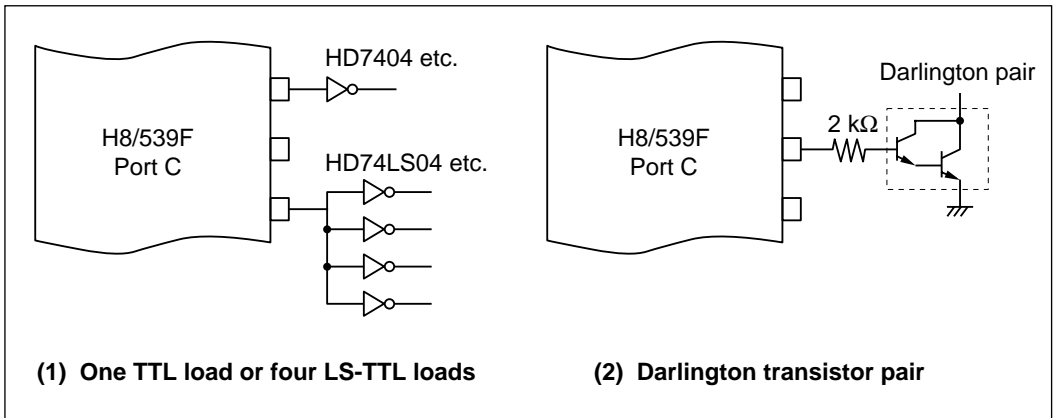
Port C is an address bus ( $A_7$  to  $A_0$ ) in modes 1, 3, 5, and 6. In modes 2 and 4 port C can be used for address output ( $A_7$  to  $A_0$ ) or general-purpose input. In mode 7 port C is a general-purpose input/output port.

Pins in port C can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair. They have software-programmable built-in MOS pull-up transistors.



**Figure 10-82 Port C Pin Functions**

Figure 10-83 shows examples of output loads for port C.



**Figure 10-83 Examples of Port C Output Loads**

### 10.13.2 Register Descriptions

Table 10-62 summarizes the registers of port C.

**Table 10-62 Port C Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE95	Port C data direction register	PCDDR	W	H'00
H'FE97	Port C data register	PCDR	R/W	H'00
H'FE99	Port C pull-up transistor control register	PCPCR	R/W	H'00

**(1) Port C Data Direction Register:** The port C data direction register (PCDDR) is an-eight-bit register. Each bit selects input or output for one pin.

Bit	7	6	5	4	3	2	1	0
	PC <sub>7</sub> DDR	PC <sub>6</sub> DDR	PC <sub>5</sub> DDR	PC <sub>4</sub> DDR	PC <sub>3</sub> DDR	PC <sub>2</sub> DDR	PC <sub>1</sub> DDR	PC <sub>0</sub> DDR
Initial value	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W

A pin in port C becomes an output pin if the corresponding PCDDR bit is set to 1, and an input pin if this bit is cleared to 0. PCDDR is a write-only register. All bits always return the value 1 when read.

PCDDR is initialized to H'00 by a reset and in hardware standby mode. PCDDR is not initialized in software standby mode.

**(2) Port C Data Register:** The port C data register (PCDR) is an-eight-bit register that stores data for pins PC<sub>7</sub> to PC<sub>0</sub>.

Bit	7	6	5	4	3	2	1	0
	PC <sub>7</sub>	PC <sub>6</sub>	PC <sub>5</sub>	PC <sub>4</sub>	PC <sub>3</sub>	PC <sub>2</sub>	PC <sub>1</sub>	PC <sub>0</sub>
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When a bit in PCDDR is set to 1, the corresponding PCDR bit value is output at the corresponding pin. If port C is read the value in PCDR is returned, regardless of the actual state of the pin.

When a bit in PCDDR is cleared to 0, it is possible to write to the corresponding PCDR bit but the value is not output at the pin. If PCDR is read the value at the pin is returned, regardless of the

value written in PCDR.

PCDR is initialized to H'00 by a reset and in hardware standby mode. PCDR is not initialized in software standby mode.

**(3) Port C Pull-Up Transistor Control Register:** The port C pull-up transistor control register (PCPCR) is an-eight-bit register that turns the MOS pull-up transistors of PC<sub>7</sub> to PC<sub>0</sub> on and off. PCPCR is ignored in modes 1 to 6 and used only in mode 7.

Bit	7	6	5	4	3	2	1	0
	PC <sub>7</sub> PON	PC <sub>6</sub> PON	PC <sub>5</sub> PON	PC <sub>4</sub> PON	PC <sub>3</sub> PON	PC <sub>2</sub> PON	PC <sub>1</sub> PON	PC <sub>0</sub> PON
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

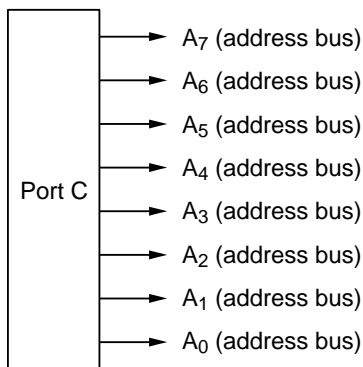
When a PCDDR bit is cleared to 0, if the corresponding PCPCR bit is set to 1, the built-in pull-up transistor is turned on.

PCPCR is initialized to H'00 by a reset and in hardware standby mode. PCPCR is not initialized in software standby mode.

### 10.13.3 Pin Functions in Each Mode

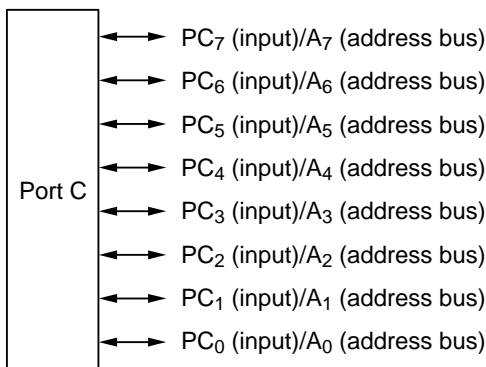
Port C has one set of functions in modes 1, 3, 5, and 6, another set of functions in modes 2 and 4, and another set of functions in mode 7. A description for each mode group is given next.

**(1) Pin Functions in Modes 1, 3, 5, and 6:** Port C is used for address output (A<sub>7</sub> to A<sub>0</sub>). The PCDDR settings are ignored. Figure 10-84 shows the pin functions in modes 1, 3, 5, and 6.



**Figure 10-84 Pin Functions in Modes 1, 3, 5, and 6**

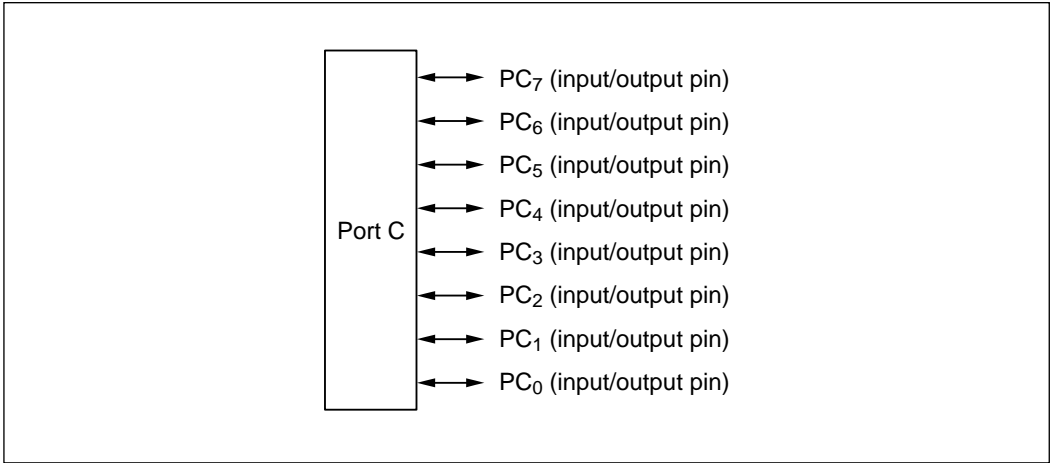
**(2) Pin Functions in Modes 2 and 4:** Port C can be used for address output ( $A_7$  to  $A_0$ ) or general-purpose input. A pin is used for address output if the corresponding PCDDR bit is set to 1, and for general-purpose input if this bit is cleared to 0. Figure 10-85 shows the pin functions in modes 2 and 4.



**Figure 10-85 Pin Functions in Modes 2 and 4**

**(3) Pin Functions in Mode 7:** Port C consists of general-purpose input/output pins. Input or output can be selected separately for each pin. A pin becomes an output pin if the corresponding PCDDR bit is set to 1 and an input pin if this bit is cleared to 0. Figure 10-86 shows the pin functions in mode 7.





**Figure 10-86 Pin Functions in Mode 7**

### 10.13.4 Built-In MOS Pull-Up Transistors

Port C has built-in MOS pull-up transistors that can be controlled by software. To turn an input pull-up transistor on, clear its PCDDR bit to 0 and set its PCPCR bit to 1. The input pull-up transistors are turned off by a reset and in hardware standby mode. Table 10-63 summarizes the states of the input pull-ups in each mode.

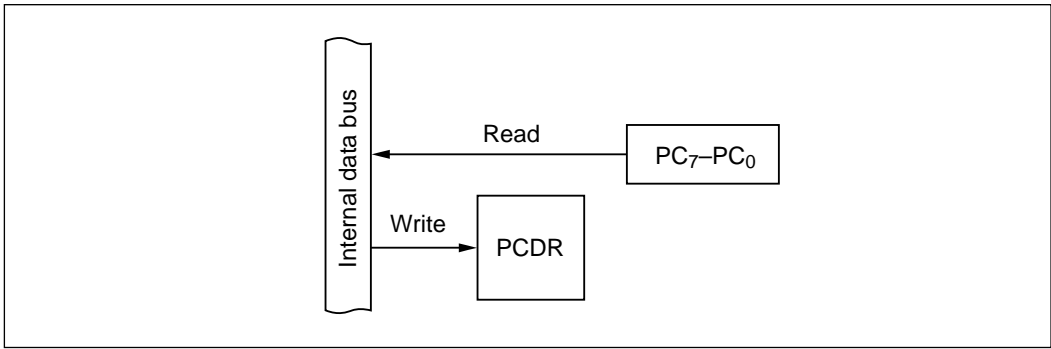
**Table 10-63 Pull-Up Transistor States in Each Mode**

Mode	Reset	Hardware Standby Mode	Other Modes (Including Software Standby Mode)
1–6	Off	Off	Off
7			On/Off

### 10.13.5 Port C Read/Write Operations

PCDR and PCDDR have different read/write functions depending on whether port C is used for address output (A<sub>7</sub> to A<sub>0</sub>) or general-purpose input or output. The operating states and functions of port C are described next.

**(1) Input Port (Modes 2 and 4):** Figure 10-87 shows a block diagram illustrating the general-purpose input function. Table 10-64 indicates register read/write data. Values written in the port C data register (PCDR) have no effect on general-purpose input lines. When read, PCDR returns the value at the pin.



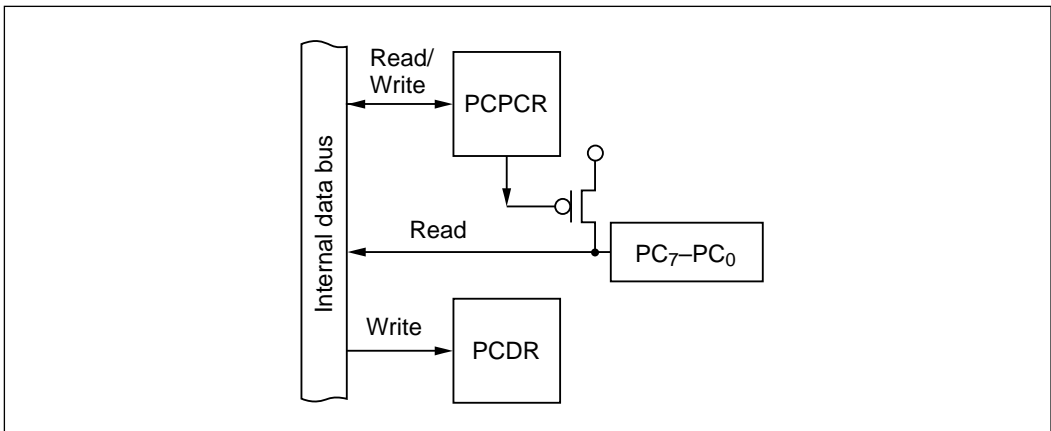
**Figure 10-87 Input Port (Modes 2 and 4)**

**Table 10-64 Register Read/Write Data**

	Read	Write
PCDR	Pin value	Don't care*

Note: The register can be written to, but the value is not output at the pins.

**(2) Input Port with Internal Pull-Up (Mode 7):** Figure 10-88 shows a block diagram illustrating the general-purpose input function of port C using the built-in input pull-up transistors. Table 10-65 indicates register read/write data. Values written in the port C data register (PCDR) have no effect on general-purpose input lines. When read, PCDR returns the value at the pin. When a bit in the port C pull-up transistor control register (PCPCR) is set to 1, the corresponding PCDR bit always reads 1.



**Figure 10-88 Input Port with Built-In Pull-Up Transistors (Mode 7)**

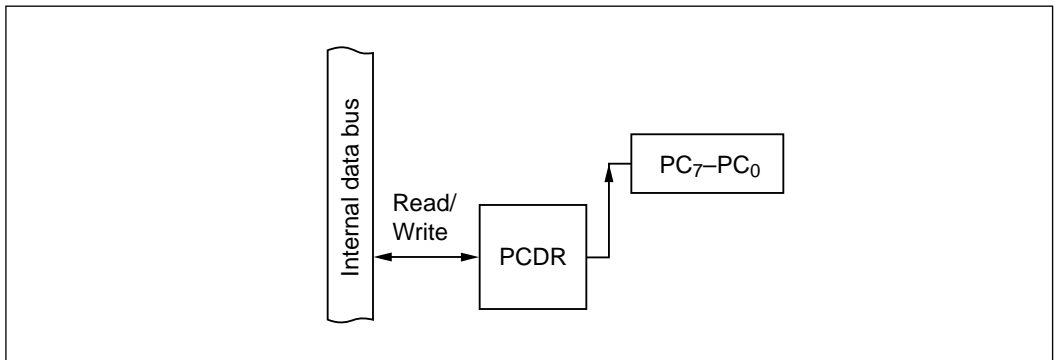
**Table 10-65 Register Read/Write Data**

	Read	Write
PCDR	Pin value, or always 1*1	Don't care*2
PCPCR	PCPCR value	0/1*1

Note: \*1 If set to 1, the corresponding PCDR bit always reads 1.

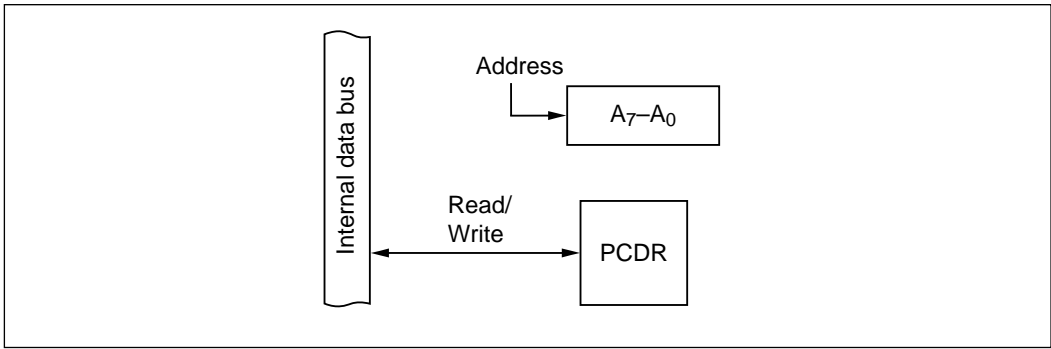
\*2 The register can be written to, but the value is not output at the pines.

**(3) Output Port (Mode 7):** Figure 10-89 shows a block diagram illustrating the general-purpose output function. Table 10-66 indicates register read/write data. The value written in the port C data register (PCDR) is output at the pin. When read, PCDR returns the value written in PCDR.

**Figure 10-89 Output Port (Mode 7)****Table 10-66 Register Read/Write Data**

	Read	Write
PCDR	PCDR value	Value output at pin

**(4) Address Bus (Modes 1 to 6):** Figure 10-90 shows a block diagram illustrating the address-bus function. Table 10-67 indicates register read/write data. When port C is used as an address bus, values written in the port C data register (PCDR) have no effect on the bus lines. When read, PCDR returns the value written in PCDR.



**Figure 10-90 Address Bus (Modes 1 to 6)**

**Table 10-67 Register Read/Write Data**

	<b>Read</b>	<b>Write</b>
PCDR	PCDR value	Don't care*

Note: The register can be written to, but the value is not output at the pines.

## 10.14 $\emptyset$ Pin

### 10.14.1 Overview

The  $\emptyset$  pin outputs the system clock. The  $\emptyset$  pin can drive one TTL load and a 90-pF capacitive load.

### 10.14.2 Register Description

Table 10-68 summarizes the  $\emptyset$  pin control register.

**Table 10-68  $\emptyset$  Pin Registers**

Address	Name	Abbreviation	R/W	Initial Value
H'FE9A	$\emptyset$ control register	$\emptyset$ CR	R/W	Undefined*1

Note: \*1 In standby mode,  $\emptyset$ CR is initialized to H'FF.

**(1)  $\emptyset$  Control Register:** The  $\emptyset$  control register ( $\emptyset$ CR) is an eight-bit register that enables or disables output of the system clock ( $\emptyset$ ).

Bit	7	6	5	4	3	2	1	0
	$\emptyset$ OE	—	—	—	—	—	—	—
Initial value	Undefined*2	1	1	1	1	1	1	1
R/W	R/W	R	R	R	R	R	R	R

Note: \*2 The  $\emptyset$ OE bit is initialized to 1 in standby mode. It is not initialized by a reset.

**Bit 7— $\emptyset$  Output Enable ( $\emptyset$ OE):** Enables or disables output of the system clock ( $\emptyset$ ). When the  $\emptyset$ OE bit is cleared to 0, the  $\emptyset$  pin goes to the high-impedance state.

**Caution:** Do not disable system clock output except in single-chip mode (mode 7). When using a mode with on-chip ROM disabled (mode 1, 3, 5, or 6), standby mode must be entered at power-on, so that the  $\emptyset$ OE bit is set to 1. Also note that the  $\emptyset$ OE bit must be set to 1 before accessing the external space. If system clock ( $\emptyset$ ) output is disabled in an expanded mode (modes 1-6), external data input and output will not be performed correctly. For details, see section 3.6, Notes on Use of Externally Expanded Modes.

#### Bit 7

$\emptyset$ OE	Description
0	System clock ( $\emptyset$ ) output is disabled
1	System clock ( $\emptyset$ ) output is enabled

# Section 11 16-Bit Integrated-Timer Pulse Unit

## 11.1 Overview

The built-in 16-bit integrated-timer pulse unit (IPU) has seven channels and three types of timers. The IPU can output 28 independent waveforms, or output 12 waveforms and process 16 pulse inputs or outputs. It can also provide six-phase PWM output, automatically measure pulse widths and periods, count input from a two-phase encoder, and start the A/D converter.

### 11.1.1 Features

The IPU features are listed below.

- Twelve waveform outputs and sixteen pulse inputs or outputs
- Sixteen registers with software-assignable output compare or input capture functions
- Twenty-eight independent comparators

Channel	Output Compare Registers	Output Compare/Input Capture Registers
CH1	4	4
CH2–5	2	2
CH6, 7	—	2

- Selection of sixteen counter clock sources (external clock sources are shared by all channels):

$\phi$ ,  $\phi/2$ ,  $\phi/4$ ,  $\phi/8$ ,  $\phi/16$ ,  $\phi/32$ ,  $\phi/64$ ,  $\phi/128$ ,  $\phi/256$ ,  $\phi/512$ ,  $\phi/1024$ ,  $\phi/2048$ ,  $\phi/4096$ , TCLK1, TCLK2, TCLK3

- Input capture function

Rising edge, falling edge, or both edges

- Pulse output

One-shot, toggle, or PWM output

- Counter synchronization function

Software can write to two or more timer counters simultaneously. Counters can be cleared simultaneously by compare match or input capture.

- PWM output mode

One-phase, two-phase, or three-phase PWM output (up to nine-phase PWM output using the counter synchronization function)

- Auto-measure function

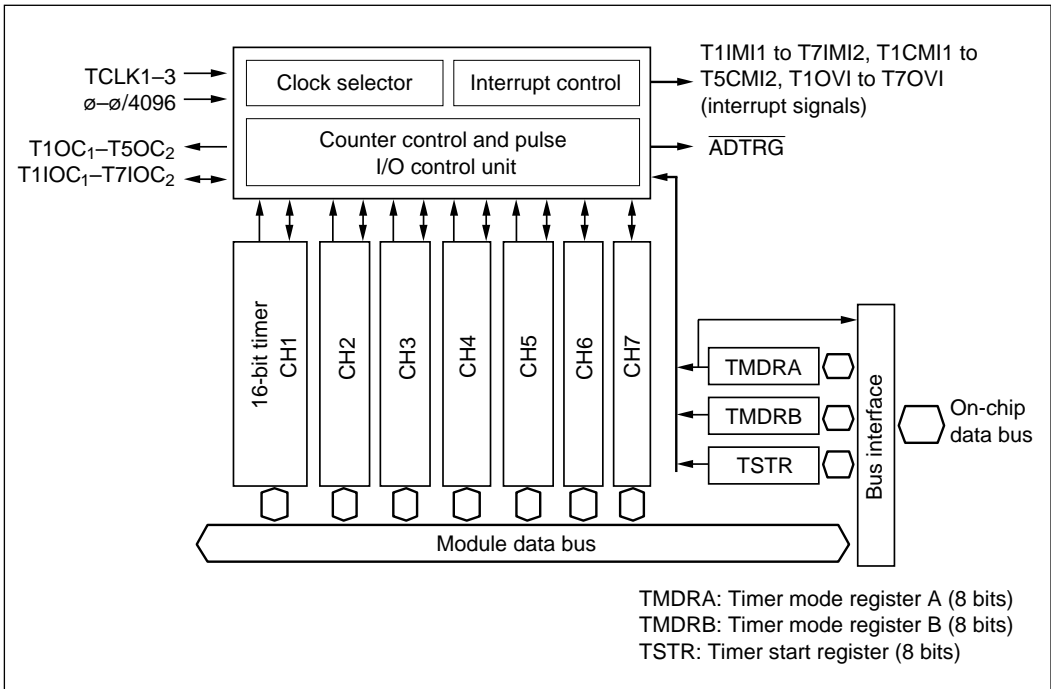
Two timer channels can be coordinated for automatic measurement of pulse width or frequency and for two-phase encoder counting

- Thirty-five interrupt sources

16 compare match/input capture interrupts, 12 compare match interrupts, and 7 overflow interrupts: total 35 sources. The compare match/input capture interrupts and overflow interrupts are independently vectored. The compare match interrupts have one interrupt vector per two interrupt sources. The compare match/input capture interrupts and compare match interrupts can start the data transfer controller (DTC) to transfer data.

### 11.1.2 Block Diagram

Figure 11-1 shows a block diagram of the IPU.



**Figure 11-1 IPU Block Diagram**

### 11.1.3 Input/Output Pins

Table 11-1 summarizes the IPU pins.

**Table 11-1 IPU Pins**

Channel	Pin Name	Input/Output	Function
1	T1IOC <sub>1</sub>	Input/Output	T1GR1 output compare/input capture pin (multiplexed with PWM output)
	T1IOC <sub>2</sub>	Input/Output	T1GR2 output compare/input capture pin (multiplexed with PWM output)
	T1OC <sub>1</sub>	Output	T1DR1 output compare pin (multiplexed with PWM output)
	T1OC <sub>2</sub>	Output	T1DR2 output compare pin
	T1IOC <sub>3</sub>	Input/Output	T1GR3 output compare/input capture pin
	T1IOC <sub>4</sub>	Input/Output	T1GR4 output compare/input capture pin
	T1OC <sub>3</sub>	Output	T1DR3 output compare pin
	T1OC <sub>4</sub>	Output	T1DR4 output compare pin
2	T2IOC <sub>1</sub>	Input/Output	T2GR1 output compare/input capture pin (multiplexed with PWM output)
	T2IOC <sub>2</sub>	Input/Output	T2GR2 output compare/input capture pin (multiplexed with PWM output)
	T2OC <sub>1</sub>	Output	T2DR1 output compare pin
	T2OC <sub>2</sub>	Output	T2DR2 output compare pin
3	T3IOC <sub>1</sub>	Input/Output	T3GR1 output compare/input capture pin (multiplexed with PWM output)
	T3IOC <sub>2</sub>	Input/Output	T3GR2 output compare/input capture pin (multiplexed with PWM output)
	T3OC <sub>1</sub>	Output	T3DR1 output compare pin
	T3OC <sub>2</sub>	Output	T3DR2 output compare pin
4	T4IOC <sub>1</sub>	Input/Output	T4GR1 output compare/input capture pin
	T4IOC <sub>2</sub>	Input/Output	T4GR2 output compare/input capture pin
	T4OC <sub>1</sub>	Output	T4DR1 output compare pin
	T4OC <sub>2</sub>	Output	T4DR2 output compare pin



**Table 11-1 IPU Pins (cont)**

Channel	Pin Name	Input/Output	Function
5	T5IOC <sub>1</sub>	Input/Output	T5GR1 output compare/input capture pin
	T5IOC <sub>2</sub>	Input/Output	T5GR2 output compare/input capture pin
	T5OC <sub>1</sub>	Output	T5DR1 output compare pin
	T5OC <sub>2</sub>	Output	T5DR2 output compare pin
6	T6IOC <sub>1</sub>	Input/Output	T6GR1 output compare/input capture pin (multiplexed with PWM output)
	T6IOC <sub>2</sub>	Input/Output	T6GR2 output compare/input capture pin
7	T7IOC <sub>1</sub>	Input/Output	T7GR1 output compare/input capture pin (multiplexed with PWM output)
	T7IOC <sub>2</sub>	Input/Output	T7GR2 output compare/input capture pin
External clock	TCLK <sub>1</sub>	Input	External clock 1 input pin (A phase input for phase measurement mode)
	TCLK <sub>2</sub>	Input	External clock 2 input pin (B phase input for phase measurement mode)
	TCLK <sub>3</sub>	Input	External clock 3

## 11.2 Timer Counters and Compare/Capture Registers

The IPU has seven 16-bit timer counters (TCNTs), one for each channel. Each counter can be accessed 16 bits at a time.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCNT	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Each of the seven channels has 16-bit capture and compare registers. A capture register latches the TCNT value when an external capture signal is received or an event occurs. An interrupt can also be requested at this time compare register contents are compared with the TCNT value at all times, and a compare match signal and/or interrupt is generated when the two match. The configuration of each channel will be described next.

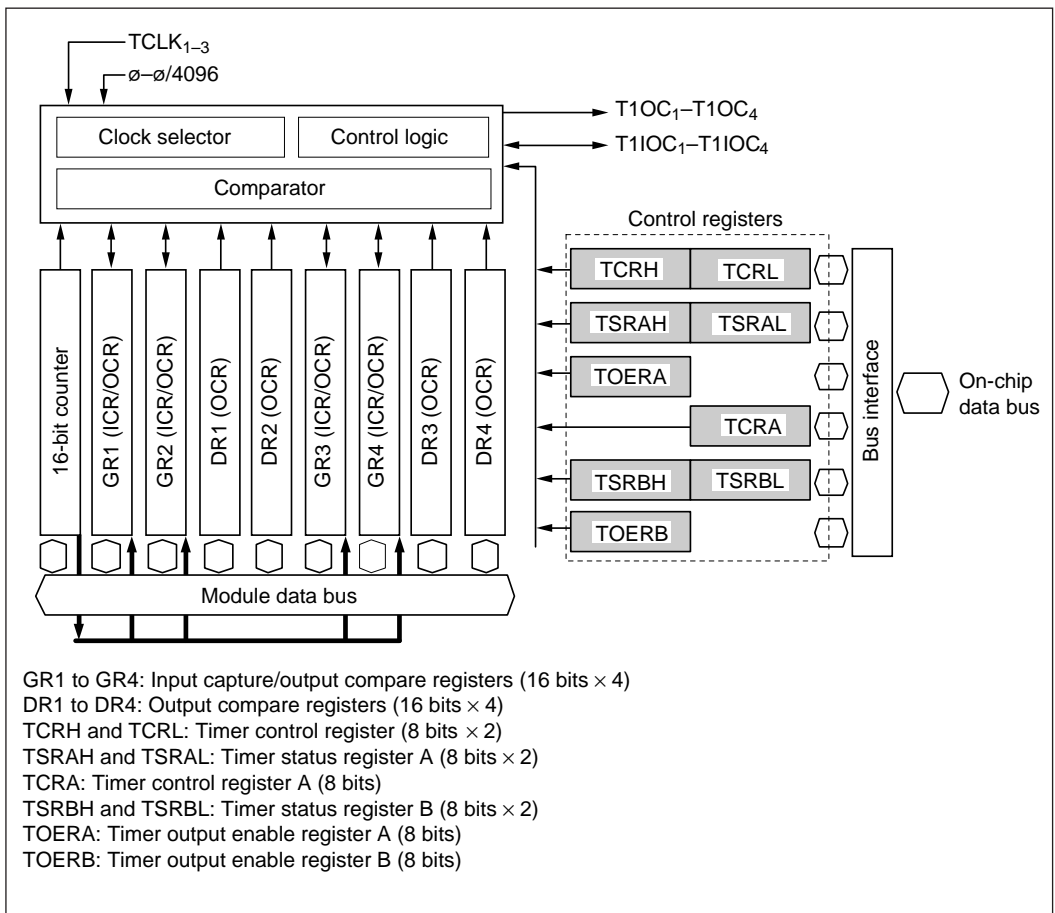
## 11.3 Channel 1 Registers

Channel 1 has four general registers used for both input capture and output compare, and four dedicated registers used only for output compare.

The input capture/output compare registers function as output compare registers after a reset. They can be switched over to input capture by setting bits IEG41 to IEG10 in the timer control registers.

Channel 1 can simultaneously generate a maximum of eight waveforms, or can simultaneously generate four waveforms and measure four waveforms. Three-phase PWM output is possible in PWM mode. See section 11.8, “Examples of Timer Operation” for details.

Figure 11-2 shows a block diagram of channel 1.



**Figure 11-2 Channel 1 Block Diagram**

### 11.3.1 Register Configuration

Table 11-2 summarizes the channel 1 registers.

**Table 11-2 Channel 1 Registers**

Chan- nel	Address	Name	Abbe- viation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
1	H'FF20	Timer control register (high)	T1CRH	R/W	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'C0
	H'FF21	Timer control register (low)	T1CRL	R/W	—	CCLR2	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'80
	H'FF22	Timer status register A (high)	T1SRAH	R/W	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
	H'FF23	Timer status register A (low)	T1SRAL	R/W	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
	H'FF24	Timer output enable register A	T1OERA	R/W	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
	H'FF25	Timer mode register A	TMDRA	R/W	MD6-7	MD4-7	MD3-5	MD2-6	SYNC3	SYNC2	SYNC1	SYNC0	H'00
	H'FF26	Timer counter register (high)	T1CNTH	R/W									H'00
	H'FF27	Timer counter register (low)	T1CNTL	R/W									H'00
	H'FF28	General register 1 (high)	T1GR1H	R/W									H'FF
	H'FF29	General register 1 (low)	T1GR1L	R/W									H'FF
	H'FF2A	General register 2 (high)	T1GR2H	R/W									H'FF
	H'FF2B	General register 2 (low)	T1GR2L	R/W									H'FF
	H'FF2C	Dedicated register 1 (high)	T1DR1H	R/W									H'FF
	H'FF2D	Dedicated register 1 (low)	T1DR1L	R/W									H'FF
	H'FF2E	Dedicated register 2 (high)	T1DR2H	R/W									H'FF
	H'FF2F	Dedicated register 2 (low)	T1DR2L	R/W									H'FF
	H'FF30	Timer start register	TSTR	R/W	—	STR7	STR6	STR5	STR4	STR3	STR2	STR1	H'80
	H'FF31	Timer control register A	T1CRA	R/W	—	—	—	—	IEG41	IEG40	IEG31	IEG30	H'F0

**Table 11-2 Channel 1 Registers (cont)**

Chan- nel	Address	Name	Abbre- viation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
	H'FF32	Timer status register B (high)	T1SRBH	R/W	—	—	—	—	CMIE4	CMIE3	IMIE4	IMIE3	H'F0
	H'FF33	Timer status register B (low)	T1SRBL	R/W	—	—	—	—	CMF4	CMF3	IMF4	IMF3	H'F0
	H'FF34	Timer output enable register B	T1OERB	R/W	DOE41	DOE40	DOE31	DOE30	GOE41	GOE40	GOE31	GOE30	H'00
	H'FF35	Timer mode register B	TMDRB	R/W	—	—	MDF	PWM4	PWM3	PWM2	PWM1	PWM0	H'C0
	H'FF38	General register 3 (high)	T1GR3H	R/W									H'FF
	H'FF39	General register 3 (low)	T1GR3L	R/W									H'FF
	H'FF3A	General register 4 (high)	T1GR4H	R/W									H'FF
	H'FF3B	General register 4 (low)	T1GR4L	R/W									H'FF
	H'FF3C	Dedicated register 3 (high)	T1DR3H	R/W									H'FF
	H'FF3D	Dedicated register 3 (low)	T1DR3L	R/W									H'FF
	H'FF3E	Dedicated register 4 (high)	T1DR4H	R/W									H'FF
	H'FF3F	Dedicated register 4 (low)	T1DR4L	R/W									H'FF

### 11.3.2 Timer Control Register (High)

Timer control register high (TCRH) is an eight-bit readable/writable register that selects the timer clock source. Each channel has one TCRH. The bit structure of TCRH in channel 1 is shown next.

Bit	7	6	5	4	3	2	1	0
TCRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0
Initial value	1	1	0	0	0	0	0	0
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Reserved bits  
Clock edge 1/0  
 These bits select the external clock edge  
Timer prescaler 3–0  
 These bits select the clock source

(1) **Bits 7 and 6—Reserved:** Read-only bits, always read as 1.

(2) **Bits 5 and 4—Clock Edge 1/0 (CKEG1/0):** These bits select the external clock edge.

Bit 5	Bit 4	Description
CKEG1	CKEG0	
0	0	Increment on rising edge (Initial value)
0	1	Increment on falling edge
1	0	Increment on both edges
1	1	

CKEG1/0 can be set to increment the count on the rising edge, falling edge, or both edges of the external clock. When TPSC3 to TPSC0 are set so as not to select an external clock source, CKEG1 and CKEG0 are ignored.

For further details, see section 11.8.7, “External Event Counting.”

**(3) Bits 3 to 0—Timer Prescaler (TPSC3 to TPSC0):** These bits select the clock source. One of 16 clock sources can be selected, as listed next.

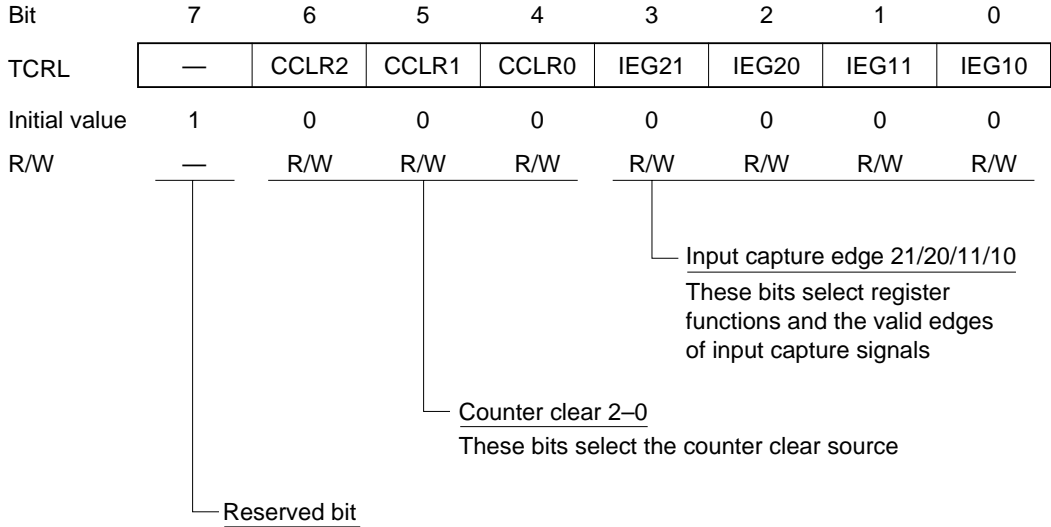
Bit 3	Bit 2	Bit 1	Bit 0	Description	
TPSC3	TPSC2	TPSC1	TPSC0		
0	0	0	0	$\emptyset$	(100 ns)* (Initial value)
0	0	0	1	$\emptyset/2$	(200 ns)*
0	0	1	0	$\emptyset/4$	(400 ns)*
0	0	1	1	$\emptyset/8$	(800 ns)*
0	1	0	0	$\emptyset/16$	(1.6 $\mu$ s)*
0	1	0	1	$\emptyset/32$	(3.2 $\mu$ s)*
0	1	1	0	$\emptyset/64$	(6.4 $\mu$ s)*
0	1	1	1	$\emptyset/128$	(12.8 $\mu$ s)*
1	0	0	0	$\emptyset/256$	(25.6 $\mu$ s)*
1	0	0	1	$\emptyset/512$	(51.2 $\mu$ s)*
1	0	1	0	$\emptyset/1024$	(102.4 $\mu$ s)*
1	0	1	1	$\emptyset/2048$	(204.8 $\mu$ s)*
1	1	0	0	$\emptyset/4096$	(409.6 $\mu$ s)*
1	1	0	1	External clock (TCLK <sub>1</sub> )	
1	1	1	0	External clock (TCLK <sub>2</sub> )	
1	1	1	1	External clock (TCLK <sub>3</sub> )	

Note: \* Values in parentheses are resolution values for a 10-MHz clock rate.

### 11.3.3 Timer Control Register (Low)

Timer control register low (TCRL) is an eight-bit readable/writable register that selects input capture edges, and selects the timer counter clear source.

Channel 1 has two timer control registers (low), designated TCRL and TCRA. The bit structure of TCRL in channel 1 is shown next.



(1) **Bit 7 —Reserved:** Read-only bit, always read as 1.

**(2) Bits 6 to 4—Counter Clear 2 to 0 (CCLR2/1/0):** These bits select the counter clear source.

Bit 6	Bit 5	Bit 4	Description
CCLR2	CCLR1	CCLR0	
0	0	0	Counter not cleared (Initial value)
0	0	1	Synchronized counter clearing enabled
0	1	0	
0	1	1	
1	0	0	Counter cleared on GR1 compare match or capture
1	0	1	Counter cleared on DR2 compare match
1	1	0	Counter cleared on GR3 compare match or capture
1	1	1	Counter cleared on DR4 compare match

When CCLR2 is 0 and either CCLR1 or CCLR0 is set to 1, or both CCLR1 and CCLR0 are set to 1, the counter is cleared in synchronization with the clearing of a timer pair selected in timer mode register A (TMDA).

If GR1 or GR3 is used as a compare register the counter is cleared by compare match. If GR1 or GR3 is used as a capture register the counter is cleared by input capture.

For further details, see section 11.8.4, “Counter Clearing Function” and section 11.8.6, “Synchronizing Mode.”

**(3) Bits 3 and 2—Input Capture Edge 21/20 (IEG21/20):** These bits select the function of GR2 and the valid edge of the input capture signal.

Bit 3	Bit 2	Description
IEG21	IEG20	
0	0	GR2 is not used for input capture (Initial value)*
0	1	Capture in GR2 on rising edge of input capture signal
1	0	Capture in GR2 on falling edge of input capture signal
1	1	Capture in GR2 on both edges of input capture signal

Note: \* GR2 becomes an output compare register.

A reset clears bits IEG21 and IEG20 to 0, disabling input capture and making GR2 an output compare register. If IEG21 or IEG20 is set to 1, or both IEG21 and IEG20 are set to 1, GR2 becomes an input capture register.

For further details, see section 11.8.3, “Input Capture Function.”



**(4) Bits 1 and 0—Input Capture Edge 11/10 (IEG11/10):** These bits select the function of GR1 and the valid edge of the input capture signal.

Bit 1	Bit 0	Description
IEG11	IEG10	
0	0	GR1 is not used for input capture (Initial value)*
0	1	Capture in GR1 on rising edge of input capture signal
1	0	Capture in GR1 on falling edge of input capture signal
1	1	Capture in GR1 on both edges of input capture signal

Note: \* GR1 becomes an output compare register.

A reset clears bits IEG11 and IEG10 to 0, disabling input capture and making GR1 an output compare register. If IEG11 or IEG10 is set to 1, or both IEG11 and IEG10 are set to 1, GR1 becomes an input capture register.

For further details, see section 11.8.3, “Input Capture Function.”

TCRA is an eight-bit readable/writable register. The bit structure of TCRA in channel 1 is shown next.

Bit	7	6	5	4	3	2	1	0
TCRA	—	—	—	—	IEG41	IEG40	IEG31	IEG30
Initial value	1	1	1	1	0	0	0	0
R/W	—	—	—	—	R/W	R/W	R/W	R/W

Reserved bits

Input capture edge 41/40/31/30  
 These bits select register functions and the valid edges of input capture signals

**(1) Bits 7 to 4 —Reserved:** Read-only bits, always read as 1.

**(2) Bits 3 and 2—Input Capture Edge 41/40 (IEG41/40):** These bits select the function of GR4 and the valid edge of the input capture signal.

<b>Bit 3</b>	<b>Bit 2</b>	<b>Description</b>
<b>IEG41</b>	<b>IEG40</b>	
0	0	GR4 is not used for input capture (Initial value)*
0	1	Capture in GR4 on rising edge of input capture signal
1	0	Capture in GR4 on falling edge of input capture signal
1	1	Capture in GR4 on both edges of input capture signal

Note: \* GR4 becomes an output compare register.

A reset clears bits IEG41 and IEG40 to 0, disabling input capture and making GR4 an output compare register. If IEG41 or IEG40 is set to 1, or both IEG41 and IEG40 are set to 1, GR4 becomes an input capture register.

For further details, see section 11.8.3, “Input Capture Function.”

**(3) Bits 1 and 0—Input Capture Edge 31/30 (IEG31/30):** These bits select the function of GR3 and the valid edge of the input capture signal.

<b>Bit 1</b>	<b>Bit 0</b>	<b>Description</b>
<b>IEG31</b>	<b>IEG30</b>	
0	0	GR3 is not used for input capture (Initial value)*
0	1	Capture in GR3 on rising edge of input capture signal
1	0	Capture in GR3 on falling edge of input capture signal
1	1	Capture in GR3 on both edges of input capture signal

Note: \* GR3 becomes an output compare register.

A reset clears bits IEG31 and IEG30 to 0, disabling input capture and making GR3 an output compare register. If IEG31 or IEG30 is set to 1, or both IEG31 and IEG30 are set to 1, GR3 becomes an input capture register.

For further details, see section 11.8.3, “Input Capture Function.”

### 11.3.4 Timer Status Register (High)

Timer status register high (TSRH) is an eight-bit readable/writable register that enables and disables timer interrupts.

After OVIE, CMIE2, CMIE1, IMIE2, or IMIE1 is set to 1 in TSRH, an interrupt is requested when OVF, CMF2, CMFI, IMF2, or IMF1 is set to 1 in TSRL.

Channel 1 has two timer status registers (high), designated TSRAH and TSRBH. Channels 2 to 7 have one TSRH each. The bit structure of TSRAH in channel 1 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRAH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1
Initial value	1	1	1	0	0	0	0	0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W

Reserved bits

Overflow interrupt enable  
Enables or disables counter overflow interrupts

Compare match interrupt enable 2/1  
These bits enable and disable DR2 and DR1 compare match interrupts

Input capture/Compare match interrupt enable 2/1  
These bits enable and disable GR2 and GR1 compare match and input capture interrupts

(1) **Bits 7 to 5—Reserved:** Read-only bits, always read as 1.

(2) **Bit 4—Overflow Interrupt Enable (OVIE):** Enables or disables the counter overflow interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 4**

<b>OVIE</b>	<b>Description</b>	
0	Counter overflow interrupt is disabled	(Initial value)
1	Counter overflow interrupt is enabled	

(3) **Bit 3—Compare Match Interrupt Enable 2 (CMIE2):** Enables or disables the DR2 compare match interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 3**

<b>CMIE2</b>	<b>Description</b>	
0	DR2 compare match interrupt is disabled	(Initial value)
1	DR2 compare match interrupt is enabled	

(4) **Bit 2—Compare Match Interrupt Enable 1 (CMIE1):** Enables or disables the DR1 compare match interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 2**

<b>CMIE1</b>	<b>Description</b>	
0	DR1 compare match interrupt is disabled	(Initial value)
1	DR1 compare match interrupt is enabled	

(5) **Bit 1—Input Capture/Compare Match Interrupt Enable 2 (IMIE2):** Enables or disables the GR2 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

<b>IMIE2</b>	<b>Description</b>	
0	GR2 compare match or input capture interrupt is disabled	(Initial value)
1	GR2 compare match or input capture interrupt is enabled	

**(6) Bit 0—Input Capture/Compare Match Interrupt Enable 1 (IMIE1):** Enables or disables the GR1 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

IMIE1	Description
0	GR1 compare match or input capture interrupt is disabled (Initial value)
1	GR1 compare match or input capture interrupt is enabled

TSRBH is an eight-bit readable/writable register. The bit structure of TSRBH in channel 1 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRBH	—	—	—	—	CMIE4	CMIE3	IMIE4	IMIE3
Initial value	1	1	1	1	0	0	0	0
R/W	—	—	—	—	R/W	R/W	R/W	R/W

Reserved bits

Input capture/Compare match interrupt enable 4/3  
These bits enable and disable GR4 and GR3 compare match and input capture interrupts

Compare match interrupt enable 4/3  
These bits enable and disable DR4 and DR3 compare match interrupts

**(1) Bits 7 to 4—Reserved:** Read-only bits, always read as 1.

(2) **Bit 3—Compare Match Interrupt Enable 4 (CMIE4):** Enables or disables the DR4 compare match interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 3**

<b>CMIE4</b>	<b>Description</b>	
0	DR4 compare match interrupt is disabled	(Initial value)
1	DR4 compare match interrupt is enabled	

(3) **Bit 2—Compare Match Interrupt Enable 3 (CMIE3):** Enables or disables the DR3 compare match interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 2**

<b>CMIE3</b>	<b>Description</b>	
0	DR3 compare match interrupt is disabled	(Initial value)
1	DR3 compare match interrupt is enabled	

(4) **Bit 1—Input Capture/Compare Match Interrupt Enable 4 (IMIE4):** Enables or disables the GR4 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

<b>IMIE4</b>	<b>Description</b>	
0	GR4 compare match or input capture interrupt is disabled	(Initial value)
1	GR4 compare match or input capture interrupt is enabled	

(5) **Bit 0—Input Capture/Compare Match Interrupt Enable 3 (IMIE3):** Enables or disables the GR3 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

<b>IMIE3</b>	<b>Description</b>	
0	GR3 compare match or input capture interrupt is disabled	(Initial value)
1	GR3 compare match or input capture interrupt is enabled	

### 11.3.5 Timer Status Register (Low)

Timer status register low (TSRL) is an eight-bit readable/writable register that indicates timer status. Writing to TSRL is restricted to clearing a flag to 0 after reading the 1 value of that flag.

After OVIE, CMIE2, CMIE1, IMIE2, or IMIE1 is set to 1 in TSRH, an interrupt is requested when OVF, CMF2, CMF1, IMF2, or IMF1 is set to 1 in TSRL.

Channel 1 has two timer status registers (low), designated TSRAL and TSRBL. Channels 2 to 7 have one TSRL each. The bit structure of TSRAL in channel 1 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRAL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1
Initial value	1	1	1	0	0	0	0	0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W

Input capture/ Compare match flag 2/1  
 Flags indicating GR2 and GR1 compare match or input capture

Compare match flag 2/1  
 Flags indicating DR2 and DR1 compare match

Overflow flag  
 Flag indicating counter overflow

Reserved bits

(1) **Bits 7 to 5—Reserved:** Read-only bits, always read as 1.

(2) **Bit 4—Overflow Flag (OVF):** Set to 1 when the counter overflows from H'FFFF to H'0000. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 4**

<b>OVF</b>	<b>Description</b>
0	Cleared by reading OVF after OVF is set to 1, then writing 0 in OVF (Initial value)
1	Set when counter overflow occurs

(3) **Bit 3—Compare Match Flag 2 (CMF2):** Set to 1 when the counter value matches the DR2 value. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 3**

<b>CMF2</b>	<b>Description</b>
0	1. Cleared by reading CMF2 after CMF2 is set to 1, then writing 0 in CMF2 (Initial value) 2. Cleared when the DTC is activated by a CMI2 interrupt
1	Set when DR2 compare match occurs

(4) **Bit 2—Compare Match Flag 1 (CMF1):** Set to 1 when the counter value matches the DR1 value. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 2**

<b>CMF1</b>	<b>Description</b>
0	1. Cleared by reading CMF1 after CMF1 is set to 1, then writing 0 in CMF1 (Initial value) 2. Cleared when the DTC is activated by a CMI1 interrupt
1	Set when DR1 compare match occurs

(5) **Bit 1—Input Capture/Compare Match Flag 2 (IMF2):** Set to 1 when the counter value matches the GR2 value, or the counter value is captured in GR2. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

<b>IMF2</b>	<b>Description</b>
0	1. Cleared by reading IMF2 after IMF2 is set to 1, then writing 0 in IMF2 (Initial value) 2. Cleared when the DTC is activated by an IMI2 interrupt
1	Set when GR2 input capture or compare match occurs



**(6) Bit 0—Input Capture/Compare Match Flag 1 (IMF1):** Set to 1 when the counter value matches the GR1 value, or the counter value is captured in GR1. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

IMF1	Description
0	1. Cleared by reading IMF1 after IMF1 is set to 1, then writing 0 in IMF1 (Initial value) 2. Cleared when the DTC is activated by an IMI1 interrupt
1	Set when GR1 input capture or compare match occurs

TSRBL is an eight-bit readable/writable register. The bit structure of TSRBL in channel 1 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRBL	—	—	—	—	CMF4	CMF3	IMF4	IMF3
Initial value	1	1	1	1	0	0	0	0
R/W	—	—	—	—	R/W	R/W	R/W	R/W

Reserved bits

Compare match flag 4/3  
Flags indicating DR4 and DR3 compare match

Input capture/  
Compare match  
flag 4/3  
Flags indicating  
GR4 and GR3  
compare match  
or input capture

**(1) Bits 7 to 4—Reserved:** Read-only bits, always read as 1.

**(2) Bit 3—Compare Match Flag 4 (CMF4):** Set to 1 when the counter value matches the DR4 value. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 3**

<b>CMF4</b>	<b>Description</b>
0	1. Cleared by reading CMF4 after CMF4 is set to 1, then writing 0 in CMF4 (Initial value) 2. Cleared when the DTC is activated by a CMI4 interrupt
1	Set when DR4 compare match occurs

**(3) Bit 2—Compare Match Flag 3 (CMF3):** Set to 1 when the counter value matches the DR3 value. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 2**

<b>CMF3</b>	<b>Description</b>
0	1. Cleared by reading CMF3 after CMF3 is set to 1, then writing 0 in CMF3 (Initial value) 2. Cleared when the DTC is activated by a CMI3 interrupt
1	Set when DR3 compare match occurs

**(4) Bit 1—Input Capture/Compare Match Flag 4 (IMF4):** Set to 1 when the counter value matches the GR4 value, or the counter value is captured in GR4. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

<b>IMF4</b>	<b>Description</b>
0	1. Cleared by reading IMF4 after IMF4 is set to 1, then writing 0 in IMF4 (Initial value) 2. Cleared when the DTC is activated by an IMI4 interrupt
1	Set when GR4 input capture or compare match occurs

**(5) Bit 0—Input Capture/Compare Match Flag 3 (IMF3):** Set to 1 when the counter value matches the GR3 value, or the counter value is captured in GR3. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

<b>IMF3</b>	<b>Description</b>
0	1. Cleared by reading IMF3 after IMF3 is set to 1, then writing 0 in IMF3 (Initial value) 2. Cleared when the DTC is activated by an IMI3 interrupt
1	Set when GR3 input capture or compare match occurs

### 11.3.6 Timer Output Enable Register

The timer output enable register (TOER) is an eight-bit readable/writable register that enables or disables output of compare match signals and selects the output level.

Channel 1 has two timer output enable registers, designated TOERA and TOERB. Channels 2 to 7 have one TOER each. The bit structure of TOERA in channel 1 is shown next.

For the selection of general register (GR) functions, see section 11.3.3, “Timer Control Register (Low).”

Bit	7	6	5	4	3	2	1	0
TOERA	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

<p><u>Dedicated register output enable 21/20</u> These bits enable and disable output of the counter-DR2 compare match signal, and select the output level</p>	<p><u>Dedicated register output enable 11/10</u> These bits enable and disable output of the counter-DR1 compare match signal, and select the output level</p>	<p><u>General register output enable 21/20</u> These bits enable and disable output of the counter-GR2 compare match signal, and select the output level</p>	<p><u>General register output enable 11/10</u> These bits enable and disable output of the counter-GR1 compare match signal, and select the output level</p>
--	--	--	--

**(1) Bits 7 and 6—Dedicated Register Output Enable 21/20 (DOE21/20):** These bits enable and disable output of the counter-DR2 compare match signal, and select the output level. For further details, see section 11.8.2, “Selection of Output Level.”

<b>Bit 7</b>	<b>Bit 6</b>	<b>Description</b>
<b>DOE21</b>	<b>DOE20</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

**(2) Bits 5 and 4—Dedicated Register Output Enable 11/10 (DOE11/10):** These bits enable and disable output of the counter-DR1 compare match signal, and select the output level. For further details, see section 11.8.2, “Selection of Output Level.”

<b>Bit 5</b>	<b>Bit 4</b>	<b>Description</b>
<b>DOE11</b>	<b>DOE10</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

**(3) Bits 3 and 2—General Register Output Enable 21/20 (GOE21/20):** These bits enable and disable output of the counter-GR2 compare match signal, and select the output level.

<b>Bit 3</b>	<b>Bit 2</b>	<b>Description</b>
<b>GOE21</b>	<b>GOE20</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

When GR2 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE21 and GOE20. Bits 3 and 2 are thus ignored except when IEG21 = IEG20 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

**(4) Bits 1 and 0—General Register Output Enable 11/10 (GOE11/10):** These bits enable and disable output of the counter-GR1 compare match signal, and select the output level.

Bit 1	Bit 0	Description
GOE11	GOE10	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

When GR1 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE11 and GOE10. Bits 1 and 0 are thus ignored except when IEG11 = IEG10 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

TOERB is an eight-bit readable/writable register. The bit structure of TOERB in channel 1 is shown next.

For the selection of general register (GR) functions, see section 11.3.3, “Timer Control Register (Low).”

Bit	7	6	5	4	3	2	1	0
TOERB	DOE41	DOE40	DOE31	DOE30	GOE41	GOE40	GOE31	GOE30
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	<u>Dedicated register output enable 41/40</u> These bits enable and disable output of the counter-DR4 compare match signal, and select the output level		<u>Dedicated register output enable 31/30</u> These bits enable and disable output of the counter-DR3 compare match signal, and select the output level		<u>General register output enable 41/40</u> These bits enable and disable output of the counter-GR4 compare match signal, and select the output level		<u>General register output enable 31/30</u> These bits enable and disable output of the counter-GR3 compare match signal, and select the output level	

**(1) Bits 7 and 6—Dedicated Register Output Enable 41/40 (DOE41/40):** These bits enable and disable output of the counter-DR4 compare match signal, and select the output level. For further details, see section 11.8.2, “Selection of Output Level.”

<b>Bit 7</b>	<b>Bit 6</b>	<b>Description</b>
<b>DOE41</b>	<b>DOE40</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

**(2) Bits 5 and 4—Dedicated Register Output Enable 31/30 (DOE31/30):** These bits enable and disable output of the counter-DR3 compare match signal, and select the output level. For further details, see section 11.8.2, “Selection of Output Level.”

<b>Bit 5</b>	<b>Bit 4</b>	<b>Description</b>
<b>DOE31</b>	<b>DOE30</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

**(3) Bits 3 and 2—General Register Output Enable 41/40 (GOE41/40):** These bits enable and disable output of the counter-GR4 compare match signal, and select the output level.

<b>Bit 3</b>	<b>Bit 2</b>	<b>Description</b>
<b>GOE41</b>	<b>GOE40</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

When GR4 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE41 and GOE40. Bits 3 and 2 are thus ignored except when IEG41 = IEG40 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

**(4) Bits 1 and 0—General Register Output Enable 31/30 (GOE31/30):** These bits enable and disable output of the counter-GR3 compare match signal, and select the output level.

<b>Bit 1</b>	<b>Bit 0</b>	
<b>GOE31</b>	<b>GOE30</b>	<b>Description</b>
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

When GR3 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE31 and GOE30. Bits 1 and 0 are thus ignored except when IEG31 = IEG30 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

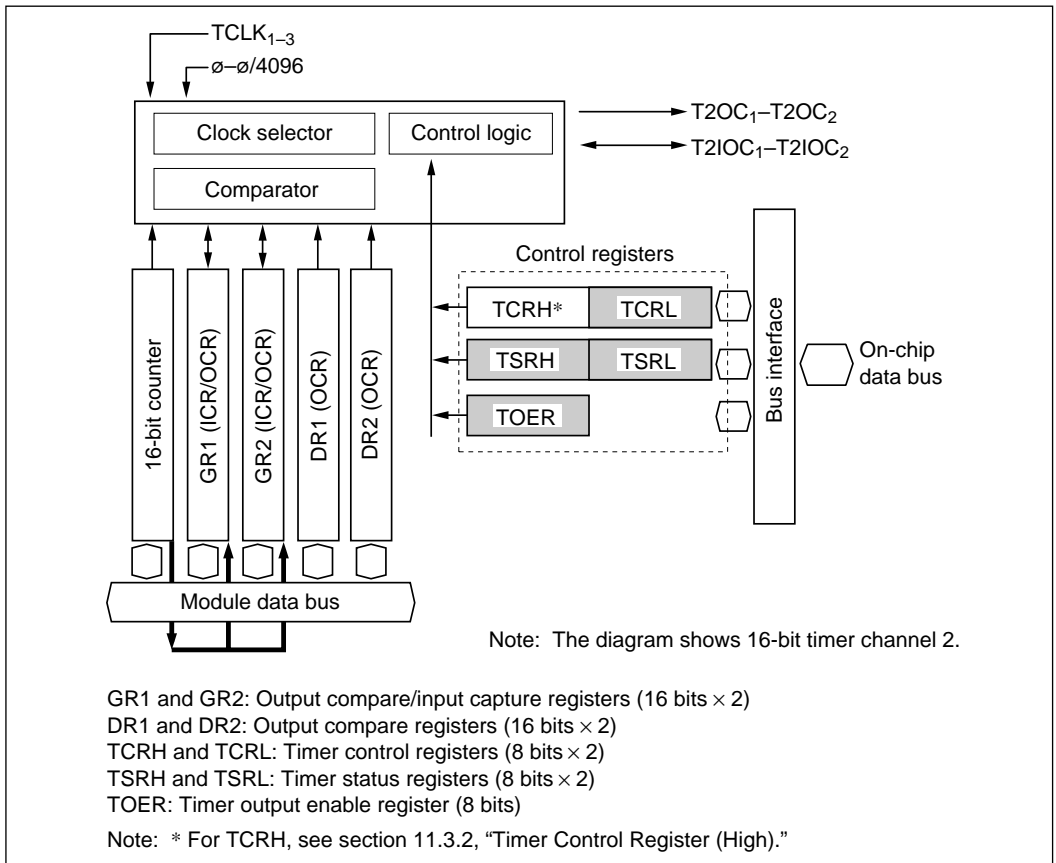
## 11.4 Channel 2 to 5 Registers

Channels 2 to 5 each have two general registers used for output compare and input capture, and two dedicated registers used only for output compare.

The general registers function as output compare registers after a reset. They can be switched over to input capture by setting IEG21 to IEG10 in the timer control registers.

Each of channels 2 to 5 can simultaneously generate a maximum of four waveforms, or can simultaneously generate two waveforms and measure two waveforms. In programmed periodic counting mode, channels 2 to 4 are used for setting the measurement period, and channel 5 is used to measure the waveform. Channels 2 and 3 can provide two-phase PWM output. See section 11.8, “Examples of Timer Operation” for details.

Figure 11-3 shows a block diagram of channels 2 to 5.



**Figure 11-3 Block Diagram of Channels 2 to 5**



## 11.4.1 Register Configuration

Table 11-3 summarizes the registers of channels 2 and 3.

**Table 11-3 Registers of Channels 2 and 3**

Channel	Address	Name	Abbreviation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
2	H'FF40	Timer control register (high)	T2CRH	R/W	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
	H'FF41	Timer control register (low)	T2CRL	R/W	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
	H'FF42	Timer status register (high)	T2SRH	R/W	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
	H'FF43	Timer status register (low)	T2SRL	R/W	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
	H'FF44	Timer output enable register	T2OER	R/W	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
	H'FF46	Timer counter register (high)	T2CNTH	R/W									H'00
	H'FF47	Timer counter register (low)	T2CNTL	R/W									H'00
	H'FF48	General register 1 (high)	T2GR1H	R/W									H'FF
	H'FF49	General register 1 (low)	T2GR1L	R/W									H'FF
	H'FF4A	General register 2 (high)	T2GR2H	R/W									H'FF
	H'FF4B	General register 2 (low)	T2GR2L	R/W									H'FF
	H'FF4C	Dedicated register 1 (high)	T2DR1H	R/W									H'FF
	H'FF4D	Dedicated register 1 (low)	T2DR1L	R/W									H'FF
	H'FF4E	Dedicated register 2 (high)	T2DR2H	R/W									H'FF
	H'FF4F	Dedicated register 2 (low)	T2DR2L	R/W									H'FF

**Table 11-3 Registers of Channels 2 and 3 (cont)**

Chan- nel	Address	Name	Abbre- viation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
3	H'FF50	Timer control register (high)	T3CRH	R/W	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
	H'FF51	Timer control register (low)	T3CRL	R/W	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
	H'FF52	Timer status register (high)	T3SRH	R/W	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
	H'FF53	Timer status register (low)	T3SRL	R/W	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
	H'FF54	Timer output enable register	T3OER	R/W	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
	H'FF56	Timer counter register (high)	T3CNTH	R/W									H'00
	H'FF57	Timer counter register (low)	T3CNTL	R/W									H'00
	H'FF58	General register 1 (high)	T3GR1H	R/W									H'FF
	H'FF59	General register 1 (low)	T3GR1L	R/W									H'FF
	H'FF5A	General register 2 (high)	T3GR2H	R/W									H'FF
	H'FF5B	General register 2 (low)	T3GR2L	R/W									H'FF
	H'FF5C	Dedicated register 1 (high)	T3DR1H	R/W									H'FF
	H'FF5D	Dedicated register 1 (low)	T3DR1L	R/W									H'FF
	H'FF5E	Dedicated register 2 (high)	T3DR2H	R/W									H'FF
	H'FF5F	Dedicated register 2 (low)	T3DR2L	R/W									H'FF

Table 11-4 summarizes the registers of channels 4 and 5.

**Table 11-4 Registers of Channels 4 and 5**

Chan- nel	Address	Name	Abbre- viation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
4	H'FF60	Timer control register (high)	T4CRH	R/W	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
	H'FF61	Timer control register (low)	T4CRL	R/W	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
	H'FF62	Timer status register (high)	T4SRH	R/W	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
	H'FF63	Timer status register (low)	T4SRL	R/W	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
	H'FF64	Timer output enable register	T4OER	R/W	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
	H'FF66	Timer counter register (high)	T4CNTH	R/W									H'00
	H'FF67	Timer counter register (low)	T4CNTL	R/W									H'00
	H'FF68	General register 1 (high)	T4GR1H	R/W									H'FF
	H'FF69	General register 1 (low)	T4GR1L	R/W									H'FF
	H'FF6A	General register 2 (high)	T4GR2H	R/W									H'FF
	H'FF6B	General register 2 (low)	T4GR2L	R/W									H'FF
	H'FF6C	Dedicated register 1 (high)	T4DR1H	R/W									H'FF
	H'FF6D	Dedicated register 1 (low)	T4DR1L	R/W									H'FF
	H'FF6E	Dedicated register 2 (high)	T4DR2H	R/W									H'FF
	H'FF6F	Dedicated register 2 (low)	T4DR2L	R/W									H'FF

**Table 11-4 Registers of Channels 4 and 5 (cont)**

Chan- nel	Address	Name	Abbre- viation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
5	H'FF70	Timer control register (high)	T5CRH	R/W	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
	H'FF71	Timer control register (low)	T5CRL	R/W	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
	H'FF72	Timer status register (high)	T5SRH	R/W	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
	H'FF73	Timer status register (low)	T5SRL	R/W	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
	H'FF74	Timer output enable register	T5OER	R/W	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
	H'FF76	Timer counter register (high)	T5CNTH	R/W									H'00
	H'FF77	Timer counter register (low)	T5CNTL	R/W									H'00
	H'FF78	General register 1 (high)	T5GR1H	R/W									H'FF
	H'FF79	General register 1 (low)	T5GR1L	R/W									H'FF
	H'FF7A	General register 2 (high)	T5GR2H	R/W									H'FF
	H'FF7B	General register 2 (low)	T5GR2L	R/W									H'FF
	H'FF7C	Dedicated register 1 (high)	T5DR1H	R/W									H'FF
	H'FF7D	Dedicated register 1 (low)	T5DR1L	R/W									H'FF
	H'FF7E	Dedicated register 2 (high)	T5DR2H	R/W									H'FF
	H'FF7F	Dedicated register 2 (low)	T5DR2L	R/W									H'FF

### 11.4.2 Timer Control Register (Low)

Timer control register low (TCRL) is an eight-bit readable/writable register. For timer control register high (TCRH), see section 11.3.2, “Timer Control Register (High).” The bit structure of TCRL in channels 2 to 5 is shown next.

Bit	7	6	5	4	3	2	1	0
TCRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10
Initial value	1	1	0	0	0	0	0	0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W

Reserved bits  
 These bits select the counter clear source

Counter clear 1/0  
 These bits select the counter clear source

Input capture edge 21/20/11/10  
 These bits select register functions and the valid edges of input capture signals

(1) **Bits 7 and 6 —Reserved:** Read-only bits, always read as 1.

(2) **Bits 5 and 4—Counter Clear 1 and 0 (CCLR1/0):** These bits select the counter clear source.

Bit 5	Bit 4	Description
CCLR1	CCLR0	
0	0	Counter not cleared (Initial value)
0	1	Counter cleared on GR1 compare match or capture
1	0	Counter cleared on DR2 compare match*
1	1	Synchronous clearing of counter enabled

Note: \* In channels 6 and 7 the counter is cleared on GR2 compare match or capture.

When CCLR1 = CCLR0 = 1, the counter is cleared in synchronization with the clearing of the paired timer selected in timer mode register A.

If GR1 is used as a compare register the counter is cleared by compare match. If GR1 is used as a capture register the counter is cleared by input capture.

For further details, see section 11.8.4, “Counter Clearing Function” and section 11.8.6, “Synchronizing Mode.”

**(3) Bits 3 and 2—Input Capture Edge 21/20 (IEG21/20):** These bits select the function of GR2 and the valid edge of the input capture signal.

<b>Bit 3</b>	<b>Bit 2</b>	<b>Description</b>
<b>IEG21</b>	<b>IEG20</b>	
0	0	GR2 is not used for input capture (Initial value)*
0	1	Capture in GR2 on rising edge of input capture signal
1	0	Capture in GR2 on falling edge of input capture signal
1	1	Capture in GR2 on both edges of input capture signal

Note: \* GR2 becomes an output compare register.

A reset clears bits IEG21 and IEG20 to 0, disabling input capture and making GR2 an output compare register. If IEG21 or IEG20 is set to 1, or both IEG21 and IEG20 are set to 1, GR2 becomes an input capture register.

For further details, see section 11.8.3, “Input Capture Function.”

**(4) Bits 1 and 0—Input Capture Edge 11/10 (IEG11/10):** These bits select the function of GR1 and the valid edge of the input capture signal.

<b>Bit 1</b>	<b>Bit 0</b>	<b>Description</b>
<b>IEG11</b>	<b>IEG10</b>	
0	0	GR1 is not used for input capture (Initial value)*
0	1	Capture in GR1 on s rising edge of input capture signal
1	0	Capture in GR1 on falling edge of input capture signal
1	1	Capture in GR1 on both edges of input capture signal

Note: \* GR1 becomes an output compare register.

A reset clears bits IEG11 and IEG10 to 0, disabling input capture and making GR1 an output compare register. If IEG11 or IEG10 is set to 1, or both IEG11 and IEG10 are set to 1, GR1 becomes an input capture register.

For further details, see section 11.8.3, “Input Capture Function.”

### 11.4.3 Timer Status Register (High)

Timer status register high (TSRH) is an eight-bit readable/writable register. After OVIE, CMIE2, CMIE1, IMIE2, or IMIE1 is set to 1 in TSRH, an interrupt is requested when OVF, CMF2, CMF1, IMF2, or IMF1 is set to 1 in TSRL. The bit structure of TSRH in channels 2 to 5 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1
Initial value	1	1	1	0	0	0	0	0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W

Reserved bits

Overflow interrupt enable  
Enables or disables counter overflow interrupts

Compare match interrupt enable 2/1  
These bits enable and disable DR2 and DR1 compare match interrupts

Input capture/  
Compare match  
interrupt enable  
2/1  
These bits enable and disable GR2 and GR1 compare match and input capture interrupts

(1) **Bits 7 to 5—Reserved:** Read-only bits, always read as 1.

(2) **Bit 4—Overflow Interrupt Enable (OVIE):** Enables or disables the counter overflow interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

#### Bit 4

OVIE	Description
0	Counter overflow interrupt is disabled (Initial value)
1	Counter overflow interrupt is enabled

(3) **Bit 3—Compare Match Interrupt Enable 2 (CMIE2):** Enables or disables the DR2 compare match interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 3**

<b>CMIE2</b>	<b>Description</b>
0	DR2 compare match interrupt is disabled (Initial value)
1	DR2 compare match interrupt is enabled

(4) **Bit 2—Compare Match Interrupt Enable 1 (CMIE1):** Enables or disables the DR1 compare match interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 2**

<b>CMIE1</b>	<b>Description</b>
0	DR1 compare match interrupt is disabled (Initial value)
1	DR1 compare match interrupt is enabled

(5) **Bit 1—Input Capture/Compare Match Interrupt Enable 2 (IMIE2):** Enables or disables the GR2 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

<b>IMIE2</b>	<b>Description</b>
0	GR2 input capture or compare match interrupt is disabled (Initial value)
1	GR2 input capture or compare match interrupt is enabled

(6) **Bit 0—Input Capture/Compare Match Interrupt Enable 1 (IMIE1):** Enables or disables the GR1 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

<b>IMIE1</b>	<b>Description</b>
0	GR1 input capture or compare match interrupt is disabled (Initial value)
1	GR1 input capture or compare match interrupt is enabled



### 11.4.4 Timer Status Register (Low)

Timer status register low (TSRL) is an eight-bit readable/writable register. After OVIE, CMIE2, CMIE1, IMIE2, or IMIE1 is set to 1 in TSRH, an interrupt is requested when OVF, CMF2, CMF1, IMF2, or IMF1 is set to 1 in TSRL. Writing to TSRL is restricted to clearing a flag to 0 after reading the 1 value of that flag. The bit structure of TSRL in channels 2 to 5 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1
Initial value	1	1	1	0	0	0	0	0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W

Reserved bits

Overflow flag  
Flag indicating counter overflow

Compare match flag 2/1  
Flags indicating DR2 and DR1 compare match

Input capture/  
Compare match  
flag 2/1  
Flags indicating GR2 and GR1 compare match or input capture

(1) **Bits 7 to 5—Reserved:** Read-only bits, always read as 1.

(2) **Bit 4—Overflow Flag (OVF):** Set to 1 when the counter overflows from H'FFFF to H'0000. For further details, see section 11.9.1, “Interrupt Timing.”

#### Bit 4

OVF	Description
0	Cleared by reading OVF after OVF is set to 1, then writing 0 in OVF (Initial value)
1	Set when counter overflow occurs

**(3) Bit 3—Compare Match Flag 2 (CMF2):** Set to 1 when the counter value matches the DR2 value. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 3**

CMF2	Description
0	1. Cleared by reading CMF2 after CMF2 is set to 1, then writing 0 in CMF2 (Initial value) 2. Cleared when the DTC is activated by a CMI2 interrupt
1	Set when DR2 compare match occurs

**(4) Bit 2—Compare Match Flag 1 (CMF1):** Set to 1 when the counter value matches the DR1 value. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 2**

CMF1	Description
0	1. Cleared by reading CMF1 after CMF1 is set to 1, then writing 0 in CMF1 (Initial value) 2. Cleared when the DTC is activated by a CMI1 interrupt
1	Set when DR1 compare match occurs

**(5) Bit 1—Input Capture/Compare Match Flag 2 (IMF2):** Set to 1 when the counter value matches the GR2 value, or the counter value is captured in GR2. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

IMF2	Description
0	1. Cleared by reading IMF2 after IMF2 is set to 1, then writing 0 in IMF2 (Initial value) 2. Cleared when the DTC is activated by an IMI2 interrupt
1	Set when GR2 input capture or compare match occurs

**(6) Bit 0—Input Capture/Compare Match Flag 1 (IMF1):** Set to 1 when the counter value matches the GR1 value, or the counter value is captured in GR1. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

IMF1	Description
0	1. Cleared by reading IMF1 after IMF1 is set to 1, then writing 0 in IMF1 (Initial value) 2. Cleared when the DTC is activated by an IMI1 interrupt
1	Set when GR1 input capture or compare match occurs

### 11.4.5 Timer Output Enable Register

The timer output enable register (TOER) is an eight-bit readable/writable register. The bit structure of TOER in channels 2 to 5 is shown next.

For the selection of general register (GR) functions, see section 11.3.3, “Timer Control Register (Low).”

Bit	7	6	5	4	3	2	1	0
TOER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

<p><u>Dedicated register output enable 21/20</u> These bits enable and disable output of the counter-DR2 compare match signal, and select the output level</p>	<p><u>Dedicated register output enable 11/10</u> These bits enable and disable output of the counter-DR1 compare match signal, and select the output level</p>	<p><u>General register output enable 21/20</u> These bits enable and disable output of the counter-GR2 compare match signal, and select the output level</p>	<p><u>General register output enable 11/10</u> These bits enable and disable output of the counter-GR1 compare match signal, and select the output level</p>
--	--	--	--

(1) **Bits 7 and 6—Dedicated Register Output Enable 21/20 (DOE21/20):** These bits enable and disable output of the counter-DR2 compare match signal, and select the output level. For further details, see section 11.8.2, “Selection of Output Level.”

Bit 7	Bit 6	Description
DOE21	DOE20	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	Toggle on compare match*

Note: \* Channels 2 and 3 do not have an output toggle function. If these bits are set to 11, the output goes to 1 on compare match.

**(2) Bits 5 and 4—Dedicated Register Output Enable 11/10 (DOE11/10):** These bits enable and disable output of the counter-DR1 compare match signal, and select the output level. For further details, see section 11.8.2, “Selection of Output Level.”

<b>Bit 5</b>	<b>Bit 4</b>	<b>Description</b>
<b>DOE11</b>	<b>DOE10</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	Toggle on compare match*

Note: \* Channels 2 and 3 do not have an output toggle function. If these bits are set to 11, the output goes to 1 on compare match.

**(3) Bits 3 and 2—General Register Output Enable 21/20 (GOE21/20):** These bits enable and disable output of the counter-GR2 compare match signal, and select the output level.

<b>Bit 3</b>	<b>Bit 2</b>	<b>Description</b>
<b>GOE21</b>	<b>GOE20</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	Toggle on compare match*

Note: \* Channels 2 and 3 do not have an output toggle function. If these bits are set to 11, the timer outputs 1 on compare match.

When GR2 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE21 and GOE20. Bits 3 and 2 are thus ignored except when IEG21 = IEG20 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

**(4) Bits 1 and 0—General Register Output Enable 11/10 (GOE11/10):** These bits enable and disable output of the counter-GR1 compare match signal, and select the output level.

<b>Bit 1</b>	<b>Bit 0</b>	<b>Description</b>
<b>GOE11</b>	<b>GOE10</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	Toggle on compare match *

Note: \* Channels 2 and 3 do not have an output toggle function. If these bits are set to 11, the timer outputs 1 on compare match.

When GR1 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE11 and GOE10. Bits 1 and 0 are thus ignored except when IEG11 = IEG10 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

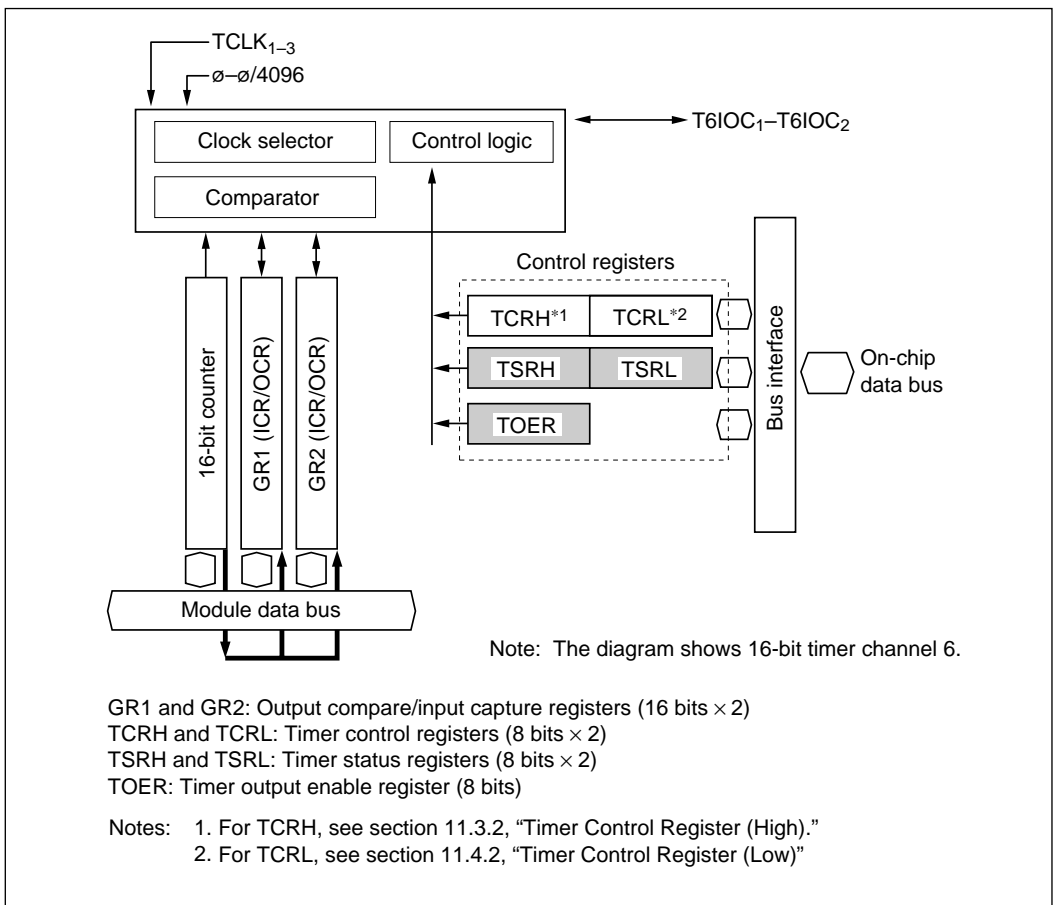
## 11.5 Channel 6 and 7 Registers

Channels 6 and 7 each have two general registers used for output compare and input capture.

The general registers function as output compare registers after a reset. They can be switched over to input capture by setting IEG21 to IEG10 in the timer control registers.

Each of channels 6 and 7 can simultaneously measure two waveforms and generate one waveform. Channels 6 and 7 can each be used to measure waveforms in programmed periodic counting mode. The timer counter in channel 7 can count up or down according to the phase of two external clock signals in phase counting mode. Channels 6 and 7 can provide single-phase PWM output in PWM output mode. See section 11.8, “Examples of Timer Operation” for details.

Figure 11-4 shows a block diagram of channels 6 and 7.



**Figure 11-4 Block Diagram of Channels 6 and 7**

## 11.5.1 Register Configuration

Table 11-5 summarizes the registers of channels 6 and 7.

**Table 11-5 Registers of Channels 6 and 7**

Chan- nel	Address	Name	Abbre- viation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
6	H'FF80	Timer control register (high)	T6CRH	R/W	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'C0
	H'FF81	Timer control register (low)	T6CRL	R/W	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'C0
	H'FF82	Timer status register (high)	T6SRH	R/W	—	—	—	—	—	OVIE	IMIE2	IMIE1	H'F8
	H'FF83	Timer status register (low)	T6SRL	R/W	—	—	—	—	—	OVF	IMF2	IMF1	H'F8
	H'FF84	Timer output enable register	T6OER	R/W	—	—	—	—	GOE21	GOE20	GOE11	GOE10	H'F0
	H'FF86	Timer counter register (high)	T6CNTH	R/W									H'00
	H'FF87	Timer counter register (low)	T6CNTL	R/W									H'00
	H'FF88	General register 1 (high)	T6GR1H	R/W									H'FF
	H'FF89	General register 1 (low)	T6GR1L	R/W									H'FF
	H'FF8A	General register 2 (high)	T6GR2H	R/W									H'FF
	H'FF8B	General register 2 (low)	T6GR2L	R/W									H'FF

**Table 11-5 Registers of Channels 6 and 7 (cont)**

Chan- nel	Address	Name	Abbre- viation	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Initial Value
7	H'FF90	Timer control register (high)	T7CRH	R/W	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
	H'FF91	Timer control register (low)	T7CRL	R/W	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
	H'FF92	Timer status register (high)	T7SRH	R/W	—	—	—	—	—	OVIE	IMIE2	IMIE1	H'F8
	H'FF93	Timer status register (low)	T7SRL	R/W	—	—	—	—	—	OVF	IMF2	IMF1	H'F8
	H'FF94	Timer output enable register	T7OER	R/W	—	—	—	—	GOE21	GOE20	GOE11	GOE10	H'F0
	H'FF96	Timer counter register (high)	T7CNTH	R/W									H'00
	H'FF97	Timer counter register (low)	T7CNTL	R/W									H'00
	H'FF98	General register 1 (high)	T7GR1H	R/W									H'FF
	H'FF99	General register 1 (low)	T7GR1L	R/W									H'FF
	H'FF9A	General register 2 (high)	T7GR2H	R/W									H'FF
	H'FF9B	General register 2 (low)	T7GR2L	R/W									H'FF



## 11.5.2 Timer Status Register (High)

Timer status register high (TSRH) is an eight-bit readable/writable register. After OVIE, IMIE2, or IMIE1 is set to 1 in TSRH, an interrupt is requested when OVF, IMF2, or IMF1 is set to 1 in TSRL. For timer control register high and low, see section 11.3.2, “Timer Control Register (High)” and section 11.4.2, “Timer Control Register (Low).” The bit structure of TSRH in channels 6 and 7 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRH	—	—	—	—	—	OVIE	IMIE2	IMIE1
Initial value	1	1	1	1	1	0	0	0
R/W	—	—	—	—	—	R/W	R/W	R/W

Input capture/  
Compare match  
interrupt enable  
2/1  
These bits enable  
and disable  
compare match  
and input capture  
interrupts

Overflow interrupt enable  
Enables or disables counter  
overflow interrupts

Reserved bits

(1) **Bits 7 to 3—Reserved:** Read-only bits, always read as 1.

(2) **Bit 2—Overflow Interrupt Enable (OVIE):** Enables or disables the counter overflow interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

### Bit 2

OVIE	Description
0	Counter overflow interrupt is disabled (Initial value)
1	Counter overflow interrupt is enabled

**(3) Bit 1—Input Capture/Compare Match Interrupt Enable 2 (IMIE2):** Enables or disables the GR2 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

<b>IMIE2</b>	<b>Description</b>
0	GR2 input capture or compare match interrupt is disabled (Initial value)
1	GR2 input capture or compare match interrupt is enabled

**(4) Bit 0—Input Capture/Compare Match Interrupt Enable 1 (IMIE1):** Enables or disables the GR1 compare match or input capture interrupt. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

<b>IMIE1</b>	<b>Description</b>
0	GR1 input capture or compare match interrupt is disabled (Initial value)
1	GR1 input capture or compare match interrupt is enabled

### 11.5.3 Timer Status Register (Low)

Timer status register low (TSRL) is an eight-bit readable/writable register. After OVIE, IMIE2, or IMIE1 is set to 1 in TSRH, an interrupt is requested when OVF, IMF2, or IMF1 is set to 1 in TSRL. Writing to TSRL is restricted to clearing a flag to 0 after reading the 1 value of that flag. The bit structure of TSRL in channels 6 and 7 is shown next.

Bit	7	6	5	4	3	2	1	0
TSRL	—	—	—	—	—	OVF	IMF2	IMF1
Initial value	1	1	1	1	1	0	0	0
R/W	—	—	—	—	—	R/W	R/W	R/W

Input capture/  
Compare match  
interrupt enable  
2/1

Flags indicating  
GR2 and GR1  
compare match  
or input capture

Overflow flag  
Flag indicating counter  
overflow

Reserved bits

(1) **Bits 7 to 3—Reserved:** Read-only bits, always read as 1.

(2) **Bit 2—Overflow Flag (OVF):** Set to 1 when the counter overflows from H'FFFF to H'0000 or when the counter in channel 7 underflows from H'0000 to H'FFFF in phase counting mode. For further details, see section 11.9.1, “Interrupt Timing,” and section 11.8.9, “Phase counting Mode.”

#### Bit 2

OVF	Description
0	Cleared by reading OVF after OVF is set to 1, then writing 0 in OVF (Initial value)
1	Set when counter overflow occurs

**(3) Bit 1—Input Capture/Compare Match Flag 2 (IMF2):** Set to 1 when the counter value matches the GR2 value, or the counter value is captured in GR2. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 1**

<b>IMF2</b>	<b>Description</b>
0	1. Cleared by reading IMF2 after IMF2 is set to 1, then writing 0 in IMF2 (Initial value) 2. Cleared when the DTC is activated by an IMI2 interrupt
1	Set when GR2 input capture or compare match occurs

**(4) Bit 0—Input Capture/Compare Match Flag 1 (IMF1):** Set to 1 when the counter value matches the GR1 value, or the counter value is captured in GR1. For further details, see section 11.9.1, “Interrupt Timing.”

**Bit 0**

<b>IMF1</b>	<b>Description</b>
0	1. Cleared by reading IMF1 after IMF1 is set to 1, then writing 0 in IMF1 (Initial value) 2. Cleared when the DTC is activated by an IMI1 interrupt
1	Set when GR1 input capture or compare match occurs

### 11.5.4 Timer Output Enable Register

The timer output enable register (TOER) is an eight-bit readable/writable register. The bit structure of TOER in channels 6 and 7 is shown next.

For the selection of general register (GR) functions, see section 11.3.3, “Timer Control Register (Low).”

Bit	7	6	5	4	3	2	1	0
TOER	—	—	—	—	GOE21	GOE20	GOE11	GOE10
Initial value	1	1	1	1	0	0	0	0
R/W	—	—	—	—	R/W	R/W	R/W	R/W

— Reserved bits

— General register output enable 21/20  
 These bits enable and disable output of the counter-GR2 compare match signal, and select the output level

— General register output enable 11/10  
 These bits enable and disable output of the counter-GR1 compare match signal, and select the output level

(1) **Bits 7 to 4—Reserved:** Read-only bits, always read as 1.

**(2) Bits 3 and 2—General Register Output Enable 21/20 (GOE21/20):** These bits enable and disable output of the counter-GR2 compare match signal, and select the output level.

<b>Bit 3</b>	<b>Bit 2</b>	<b>Description</b>
<b>GOE21</b>	<b>GOE20</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

When GR2 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE21 and GOE20. Bits 3 and 2 are thus ignored except when IEG21 = IEG20 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

**(3) Bits 1 and 0—General Register Output Enable 11/10 (GOE11/10):** These bits enable and disable output of the counter-GR1 compare match signal, and select the output level.

<b>Bit 1</b>	<b>Bit 0</b>	<b>Description</b>
<b>GOE11</b>	<b>GOE10</b>	
0	0	Compare match signal output is disabled (Initial value)
0	1	Output 0 on compare match
1	0	Output 1 on compare match
1	1	

When GR1 is used for input capture, however, compare match signal output is disabled regardless of the setting of GOE11 and GOE10. Bits 1 and 0 are thus ignored except when IEG11 = IEG10 = 0.

For further details, see section 11.8.2, “Selection of Output Level.”

## 11.6 IPU Register Descriptions

### 11.6.1 Timer Mode Register A

Timer mode register A (TMDRA) is an eight-bit readable/writable register that selects timer synchronizing and operating modes. The bit structure of TMDRA is shown next.

Bit	7	6	5	4	3	2	1	0
TMDRA	MD6-7	MD4-7	MD3-5	MD2-6	SYNC3	SYNC2	SYNC1	SYNC0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Timer mode 6-7, 4-7, 3-5, 2-6  
 These bits operate two timers  
 in programmed periodic counting mode

Timer synchronizing bits 3-0  
 These bits synchronize  
 two timers

(1) **Bit 7—Timer Mode 6-7 (MD6-7):** Operates channels 6 and 7 in programmed periodic counting mode.

#### Bit 7

MD6-7	Description
0	Timers 6 and 7 operate normally (Initial value)
1	Timers 6 and 7 operate in programmed periodic counting mode

The counter value in channel 7 is captured in GR2 in channel 7 at intervals set in GR2 in channel 6. If channel 7 is externally clocked, the number of external events occurring in regular intervals timed by channel 6 can be counted. For further details see section 11.8.8, “Programmed Periodic Counting Mode.”

(2) **Bit 6—Timer Mode 4-7 (MD4-7):** Operates channels 4 and 7 in programmed periodic counting mode.

**Bit 6**

<b>MD4-7</b>	<b>Description</b>
0	Timers 4 and 7 operate normally (Initial value)
1	Timers 4 and 7 operate in programmed periodic counting mode

The counter value in channel 7 is captured in GR1 in channel 7 at intervals set in DR2 in channel 4. If channel 7 is externally clocked, the number of external events occurring in regular intervals timed by channel 4 can be counted. For further details see section 11.8.8, “Programmed Periodic Counting Mode.”

(3) **Bit 5—Timer Mode 3-5 (MD3-5):** Operates channels 3 and 5 in programmed periodic counting mode.

**Bit 5**

<b>MD3-5</b>	<b>Description</b>
0	Timers 3 and 5 operate normally (Initial value)
1	Timers 3 and 5 operate in programmed periodic counting mode

The counter value in channel 5 is captured in GR1 in channel 5 at intervals set in DR2 in channel 3. If channel 5 is externally clocked, the number of external events occurring in regular intervals timed by channel 3 can be counted. For further details see section 11.8.8, “Programmed Periodic Counting Mode.”

(4) **Bit 4—Timer Mode 2-6 (MD2-6):** Operates channels 2 and 6 in programmed periodic counting mode.

**Bit 4**

<b>DM2-6</b>	<b>Description</b>
0	Timers 2 and 6 operate normally (Initial value)
1	Timers 2 and 6 operate in programmed periodic counting mode

The counter value in channel 6 is captured in GR1 in channel 6 at intervals set in DR2 in channel 2. If channel 6 is externally clocked, the number of external events occurring in regular intervals timed by channel 2 can be counted. For further details see section 11.8.8, “Programmed Periodic Counting Mode.”



(5) **Bit 3—Timer Synchronizing Bit 3 (SYNC3):** Synchronizes two timer channels.

**Bit 3**

<b>SYNC3</b>	<b>Description</b>
0	Timer counters in channels 6 and 7 operate independently (Initial value)
1	Timer counters in channels 6 and 7 are synchronized

When SYNC3 = 1, timer counters can be preset and cleared in synchronization. If two or more bits among SYNC3, SYNC2, SYNC1, and SYNC0 are set to 1 simultaneously, all selected timer counters are synchronized. For further details, see section 11.8.6 “Synchronizing Mode.”

(6) **Bit 2—Timer Synchronizing Bit 2 (SYNC2):** Synchronizes two timer channels.

**Bit 2**

<b>SYNC2</b>	<b>Description</b>
0	Timer counters in channels 4 and 5 operate independently (Initial value)
1	Timer counters in channels 4 and 5 are synchronized

When SYNC2 = 1, timer counters can be preset and cleared in synchronization. If two or more bits among SYNC3, SYNC2, SYNC1, and SYNC0 are set to 1 simultaneously, all selected timer counters are synchronized. For further details, see section 11.8.6 “Synchronizing Mode.”

(7) **Bit 1—Timer Synchronizing Bit 1 (SYNC1):** Synchronizes two timer channels.

**Bit 1**

<b>SYNC1</b>	<b>Description</b>
0	Timer counters in channels 2 and 3 operate independently (Initial value)
1	Timer counters in channels 2 and 3 are synchronized

When SYNC1 = 1, timer counters can be preset and cleared in synchronization. If two or more bits among SYNC3, SYNC2, SYNC1, and SYNC0 are set to 1 simultaneously, all selected timer counters are synchronized. For further details, see section 11.8.6 “Synchronizing Mode.”

(8) **Bit 0—Timer Synchronizing Bit 0 (SYNC0):** Synchronizes the timer counters in channel 1 and other channels.

**Bit 0**

<b>SYNC0</b>	<b>Description</b>
0	Timer counters in channel 1 and other channels operate independently (Initial value)
1	Timer counters in channel 1 and other channels are synchronized

When SYNC0 = 1, timer counters can be preset and cleared in synchronization. If two or more bits among SYNC3, SYNC2, SYNC1, and SYNC0 are set to 1 simultaneously, all selected timer counters are synchronized. For further details, see section 11.8.6 “Synchronizing Mode.”

### 11.6.2 Timer Mode Register B

Timer mode register B (TMDRB) is an eight-bit readable/writable register that selects timer operating modes. The bit structure of TMDRB is shown next.

Bit	7	6	5	4	3	2	1	0
TMDRB	—	—	MDF	PWM4	PWM3	PWM2	PWM1	PWM0
Initial value	1	1	0	0	0	0	0	0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W

Phase counting mode  
Operates channel 7 in phase counting mode

PWM timer mode 4-0  
These bits operate channels 7, 6, 3, 2, and 1 as pulse-width modulators

Reserved bits

(1) **Bits 7 and 6—Reserved:** Read-only bits, always read as 1.

(2) **Bit 5—Phase Counting Mode (MDF):** Operates channel 7 in phase counting mode. For further details see section 11.8.9, “Phase Counting Mode.”

#### Bit 5

MDF	Description
0	Channel 7 operates normally <span style="float: right;">(Initial value)</span>
1	Channel 7 operates in phase counting mode

(3) **Bit 4—PWM Timer Mode 4 (PWM4):** Operates channel 7 as a PWM timer.

**Bit 4**

<b>PWM4</b>	<b>Description</b>	
0	Channel 7 operates normally	(Initial value)
1	Channel 7 operates as a PWM timer	

Channel 7 operates as a PWM timer with independent period and duty cycle, providing one PWM output. When PWM4 = 1, settings of GOE11 and GOE10 in the channel 7 timer output enable register (TOER) are ignored. For further details, see section 11.8.5 “PWM Output Mode.”

(4) **Bit 3—PWM Timer Mode 3 (PWM3):** Operates channel 6 as a PWM timer.

**Bit 3**

<b>PWM3</b>	<b>Description</b>	
0	Channel 6 operates normally	(Initial value)
1	Channel 6 operates as a PWM timer	

Channel 6 operates as a PWM timer with independent period and duty cycle, providing one PWM output. When PWM3 = 1, settings of GOE11 and GOE10 in the channel 6 timer output enable register (TOER) are ignored. For further details, see section 11.8.5 “PWM Output Mode.”

(5) **Bit 2—PWM Timer Mode 2 (PWM2):** Operates channel 3 as a PWM timer.

**Bit 2**

<b>PWM2</b>	<b>Description</b>	
0	Channel 3 operates normally	(Initial value)
1	Channel 3 operates as a PWM timer	

Channel 3 operates as a PWM timer with independent period and duty cycle. Channel 3 can provide two-phase PWM output. When PWM2 = 1, settings of GOE21, GOE20, GOE11, and GOE10 in the channel 3 timer output enable register (TOER) are ignored. For further details, see section 11.8.5 “PWM Output Mode.”

(6) **Bit 1—PWM Timer Mode 1 (PWM1):** Operates channel 2 as a PWM timer.

**Bit 1**

<b>PWM1</b>	<b>Description</b>	
0	Channel 2 operates normally	(Initial value)
1	Channel 2 operates as a PWM timer	

Channel 2 operates as a PWM timer with independent period and duty cycle. Channel 2 can provide two-phase PWM output. When PWM1 = 1, settings of GOE21, GOE20, GOE11, and GOE10 in the channel 2 timer output enable register (TOER) are ignored. For further details, see section 11.8.5 “PWM Output Mode.”

(7) **Bit 0—PWM Timer Mode 0 (PWM0):** Operates channel 1 as a PWM timer.

**Bit 0**

<b>PWM0</b>	<b>Description</b>	
0	Channel 1 operates normally	(Initial value)
1	Channel 1 operates as a PWM timer	

Channel 1 operates as a PWM timer with independent period and duty cycle. Channel 1 can provide three-phase PWM output. When PWM0 = 1, settings of DOE11, DOE10, GOE21, GOE20, GOE11, and GOE10 in the channel 1 timer output enable register (TOER) are ignored. For further details, see section 11.8.5 “PWM Output Mode.”

### 11.6.3 Timer Start Register

The timer start register (TSTR) is an eight-bit readable/writable register that starts and stops the counters. The bit structure of TSTR is shown next.

Bit	7	6	5	4	3	2	1	0
TSTR	—	STR7	STR6	STR5	STR4	STR3	STR2	STR1
Initial value	1	0	0	0	0	0	0	0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Reserved bit
Counter start 7 to 1  
These bits start and stop the  
counters

(1) **Bit 7—Reserved:** Read-only bit, always read as 1.

(2) **Bit 6—Counter Start 7 (STR7):** Starts and stops the counter in channel 7.

#### Bit 6

STR7	Description	
0	Timer counter 7 is halted	(Initial value)
1	Timer counter 7 is counting	

(3) **Bit 5—Counter Start 6 (STR6):** Starts and stops the counter in channel 6.

#### Bit 5

STR6	Description	
0	Timer counter 6 is halted	(Initial value)
1	Timer counter 6 is counting	

(4) **Bit 4—Counter Start 5 (STR5):** Starts and stops the counter in channel 5.

#### Bit 4

STR5	Description	
0	Timer counter 5 is halted	(Initial value)
1	Timer counter 5 is counting	

(5) **Bit 3—Counter Start 4 (STR4):** Starts and stops the counter in channel 4.

**Bit 3**

<b>STR4</b>	<b>Description</b>	
0	Timer counter 4 is halted	(Initial value)
1	Timer counter 4 is counting	

(6) **Bit 2—Counter Start 3 (STR3):** Starts and stops the counter in channel 3.

**Bit 2**

<b>STR3</b>	<b>Description</b>	
0	Timer counter 3 is halted	(Initial value)
1	Timer counter 3 is counting	

(7) **Bit 1—Counter Start 2 (STR2):** Starts and stops the counter in channel 2.

**Bit 1**

<b>STR2</b>	<b>Description</b>	
0	Timer counter 2 is halted	(Initial value)
1	Timer counter 2 is counting	

(8) **Bit 0—Counter Start 1 (STR1):** Starts and stops the counter in channel 1.

**Bit 0**

<b>STR1</b>	<b>Description</b>	
0	Timer counter 1 is halted	(Initial value)
1	Timer counter 1 is counting	

## 11.7 H8/500 CPU Interface

Some IPU registers can be accessed 16 bits at a time, while others are limited to eight-bit access. These two types of registers differ in their write timing, as explained next.

### 11.7.1 16-Bit Accessible Registers

The timer counters (TCNT), general registers (GR), and dedicated registers (DR) are 16-bit registers. The H8/500 CPU can access these registers a word at a time using a 16-bit data bus. Byte access is also possible.

Figure 11-5 shows an example of word write timing to a timer counter. Figure 11-6 shows an example of byte write timing to a timer counter.

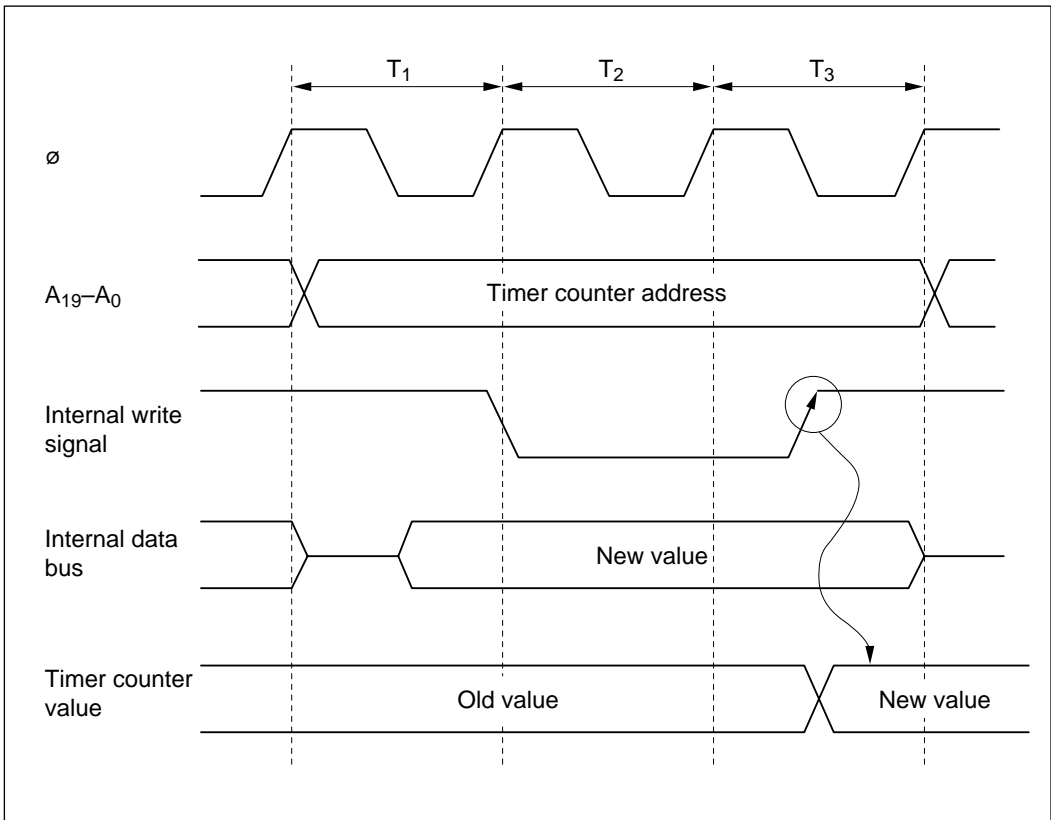
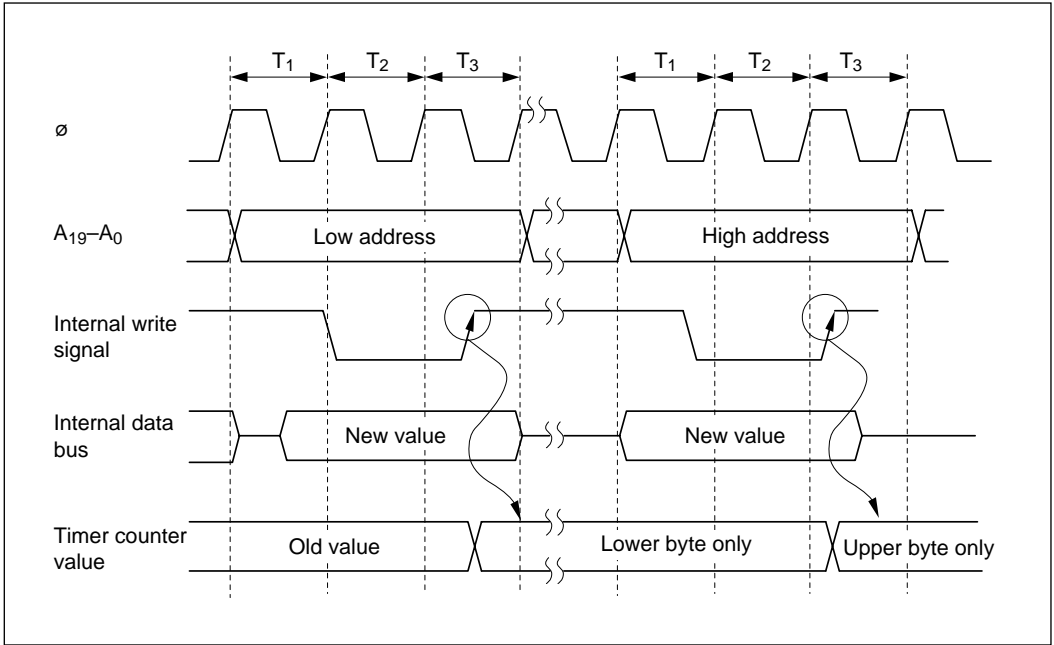
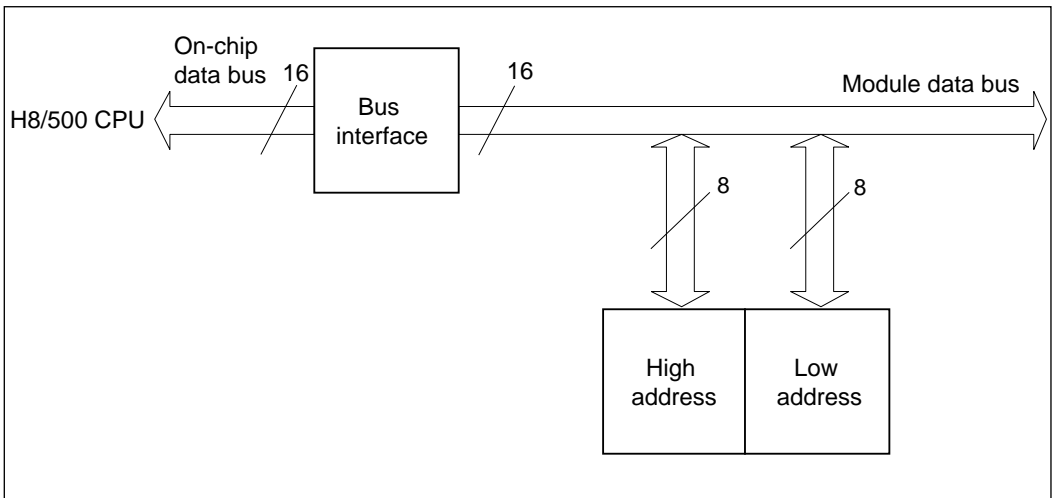


Figure 11-5 Example of Word Write Timing for Timer Counter



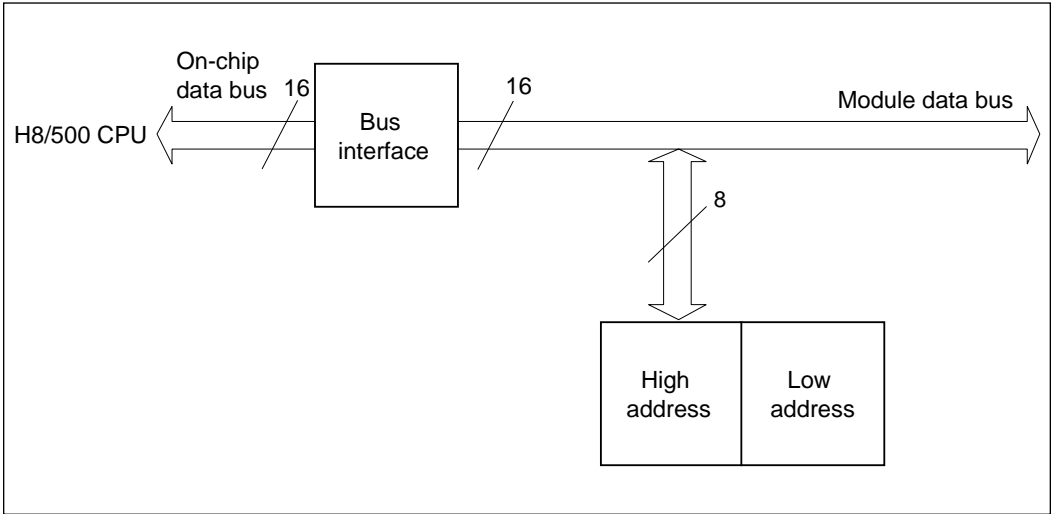
**Figure 11-6 Example of Byte Write Timing for Timer Counter**

- Read and Write Operations:** Timer counters, general registers, and dedicated registers can be written and read a word at a time or a byte at a time. Figure 11-7 illustrates word read/write operations. Figure 11-8 illustrates upper byte read/write operations. Figure 11-9 illustrates lower byte read/write operations.

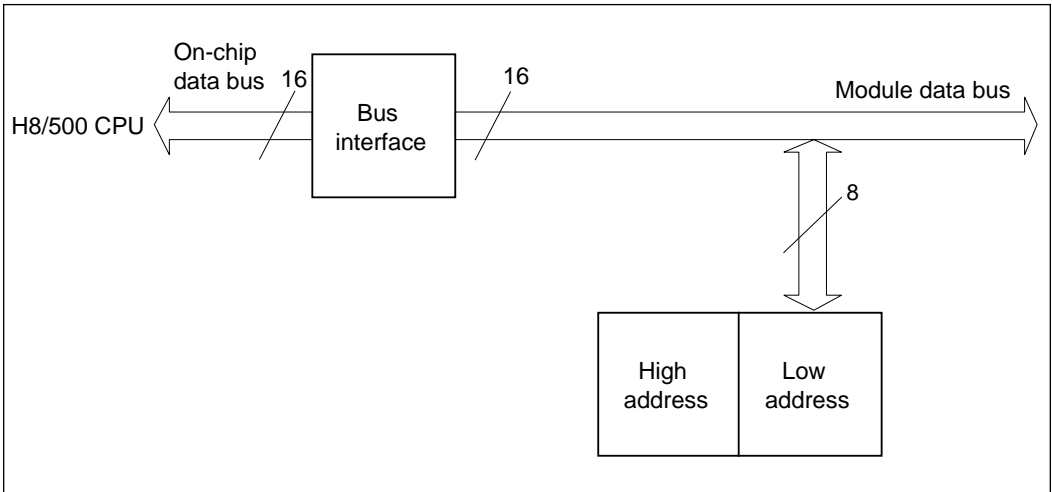


**Figure 11-7 Word Read/Write Operations**





**Figure 11-8 Upper Byte Read/Write Operations**

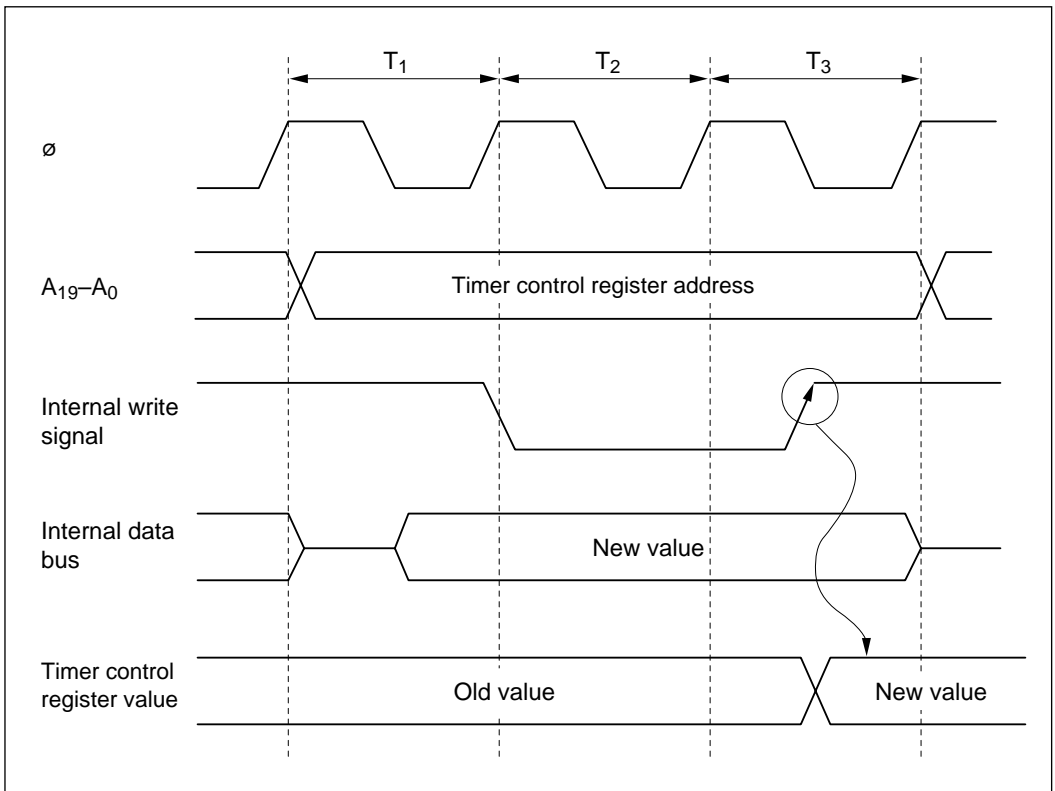


**Figure 11-9 Lower Byte Read/Write Operations**

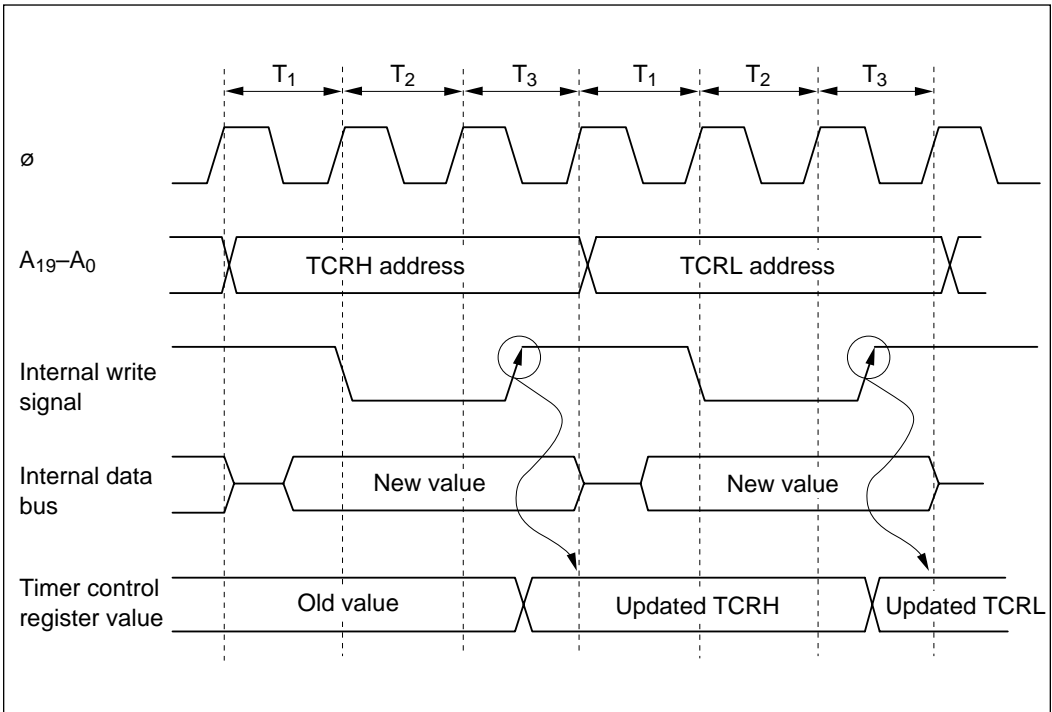
### 11.7.2 Eight-Bit Accessible Registers

The IPU's timer control registers (TCRH and TCRL), timer status registers (TSRH and TSRL), timer output enable registers (TOER), timer mode register A (TMDA), timer mode register B (TMDB), and timer start register (TSTR) are eight-bit registers. The H8/500 CPU accesses these registers a byte at a time using an eight-bit data bus. If an instruction specifies word size, two registers are accessed at consecutive addresses, upper byte (even address) first and lower byte (odd address) second.

Figure 11-10 shows an example of byte write timing to a timer control register. Figure 11-11 shows an example of write timing to a timer control register by an instruction specifying word operand size.



**Figure 11-10 Example of Byte Write Timing for Timer Control Register**

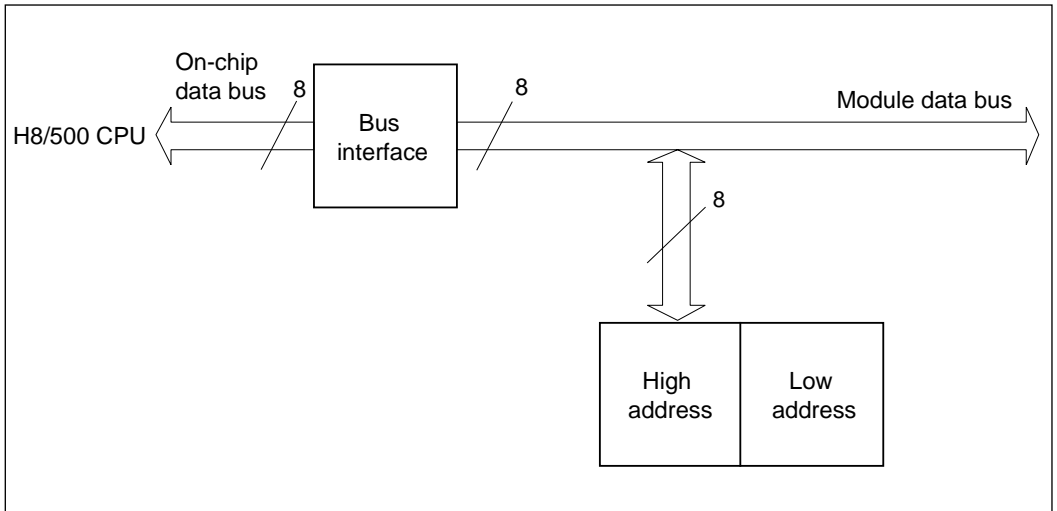


**Figure 11-11 Example of Write Timing for Timer Control Register by Instruction Specifying Word Operand Size**

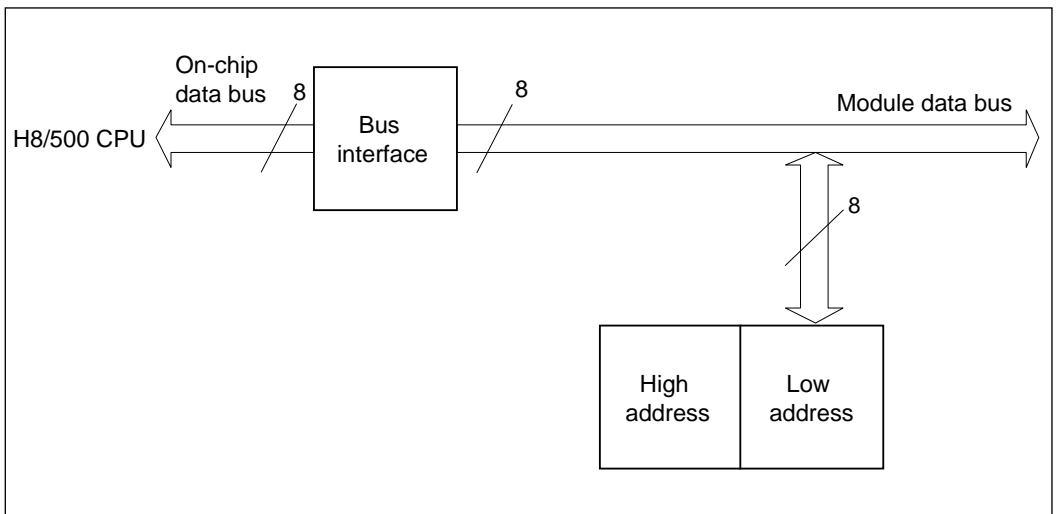
- **Read and Write Operations:** Table 11-6 lists the byte-accessed registers. Figure 11-12 illustrates upper byte read/write operations. Figure 11-13 illustrates lower byte read/write operations.

**Table 11-6 Eight-Bit Access Registers**

Name	Abbreviation		
	Byte Access	Word Access	
Timer control registers (high)	TCRH	TCR	Upper
Timer control registers (low)	TCRL		Lower
Timer status registers (high)	TSRH	TSR	Upper
Timer status registers (low)	TSRL		Lower
Timer output enable registers	TOER	TOER	Upper
Timer mode registers	TMDR	TMDR	Lower
Timer start registers	TSTR	TSTR	Upper
		T1CRB	Lower



**Figure 11-12 Upper Byte Read/Write Operations**



**Figure 11-13 Lower Byte Read/Write Operations**

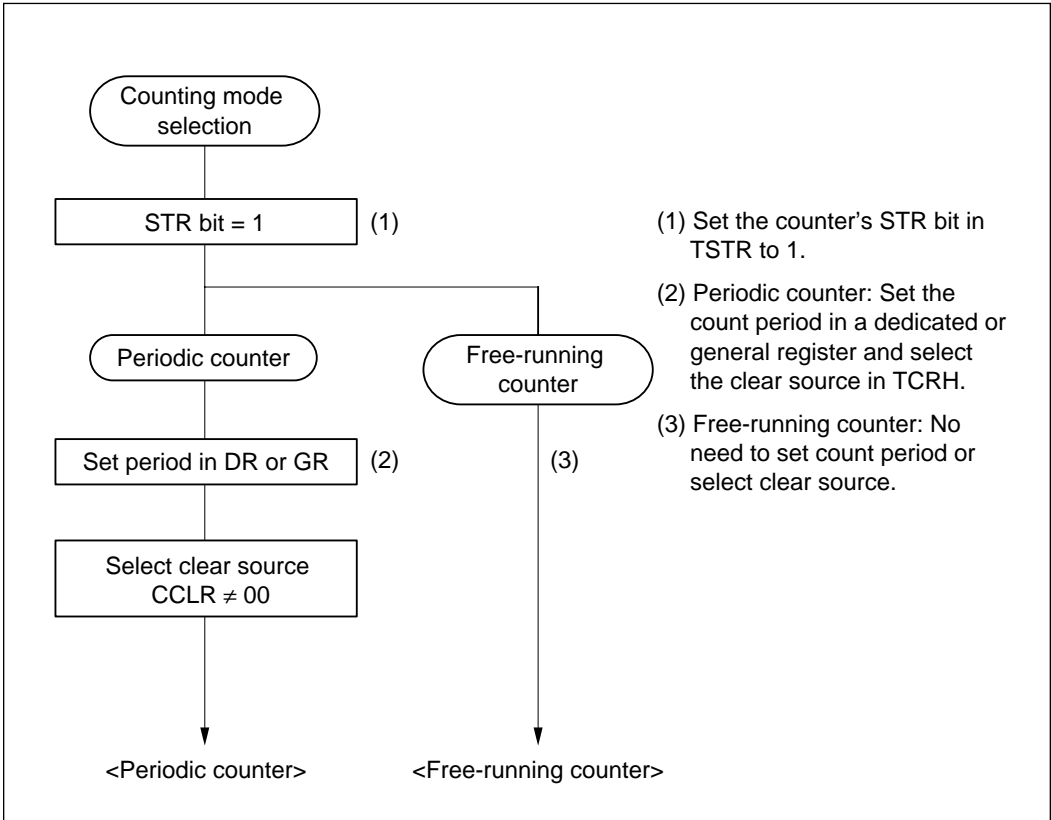
## 11.8 Examples of Timer Operation

The 16-bit integrated-timer pulse unit (IPU) has several application-oriented operating modes. These are outlined and examples are given below.

### 11.8.1 Examples of Counting

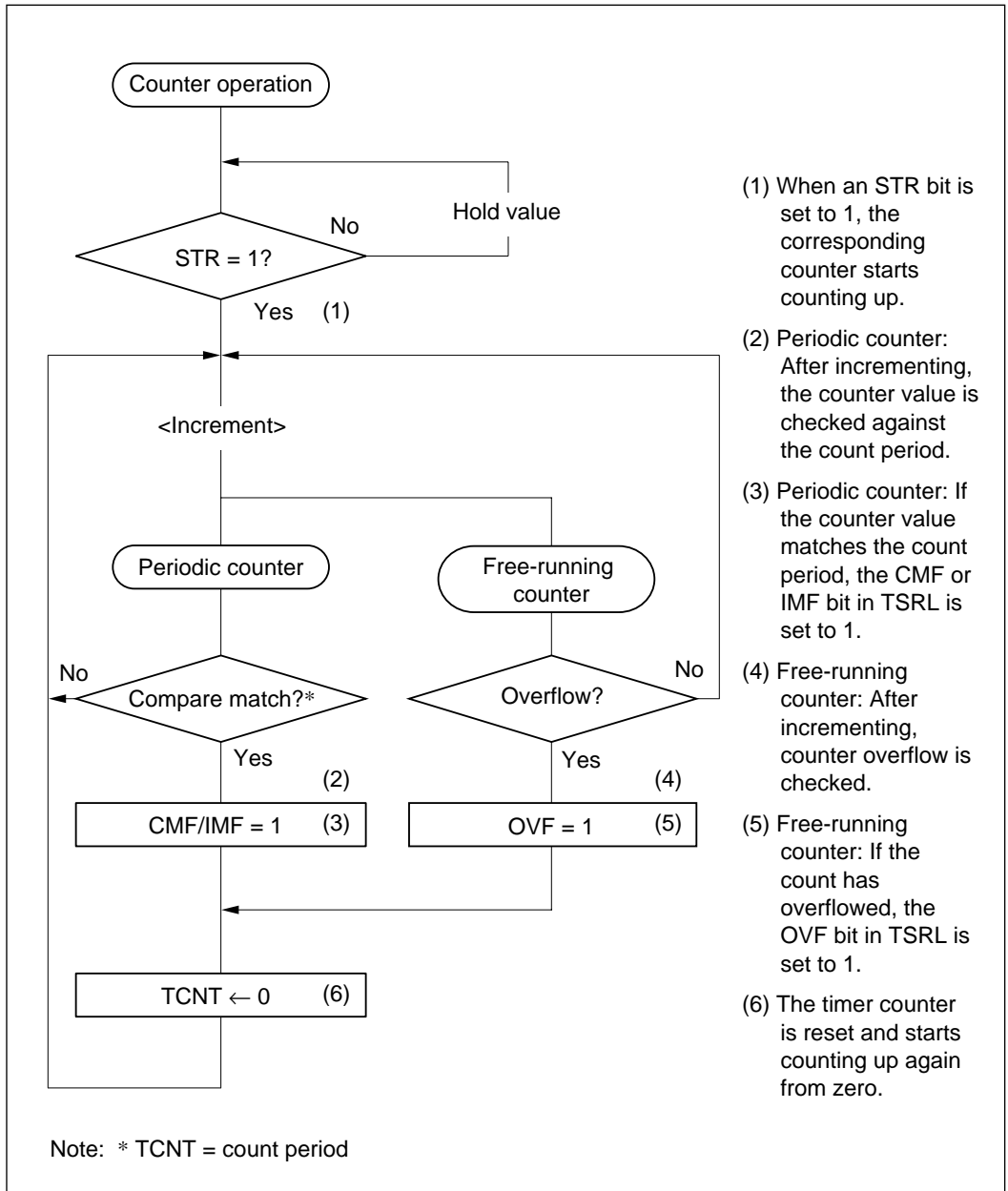
When a start (STR) bit in the timer start register (TSTR) is set to 1, the corresponding counter starts counting from H'0000. There are two counting modes: a free-running mode and a periodic mode. Figure 11-14 shows the procedure for selecting the counting mode.

#### Procedure for Selecting Counting Mode



**Figure 11-14 Procedure for Selecting Counting Mode**

**Counter Operation:** Figure 11-15 illustrates counter operations.



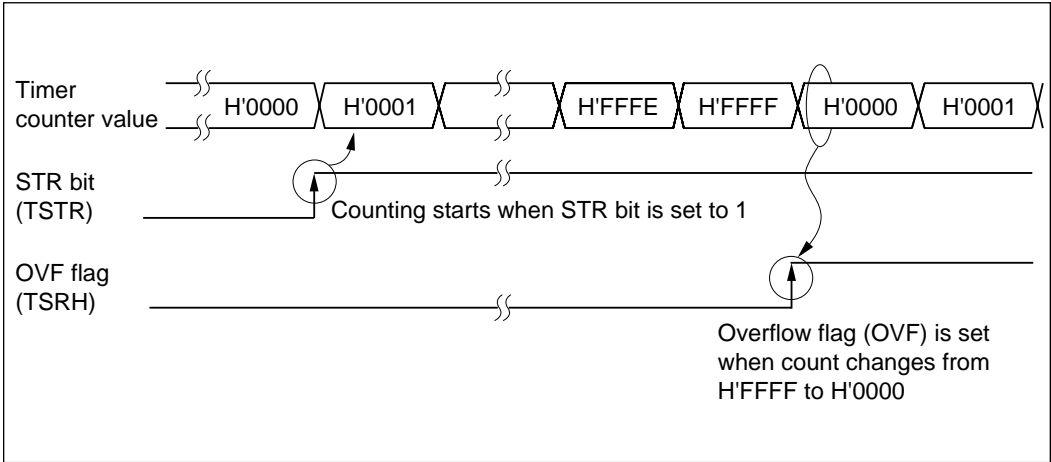
- (1) When an STR bit is set to 1, the corresponding counter starts counting up.
- (2) Periodic counter: After incrementing, the counter value is checked against the count period.
- (3) Periodic counter: If the counter value matches the count period, the CMF or IMF bit in TSRL is set to 1.
- (4) Free-running counter: After incrementing, counter overflow is checked.
- (5) Free-running counter: If the count has overflowed, the OVF bit in TSRL is set to 1.
- (6) The timer counter is reset and starts counting up again from zero.

Note: \* TCNT = count period

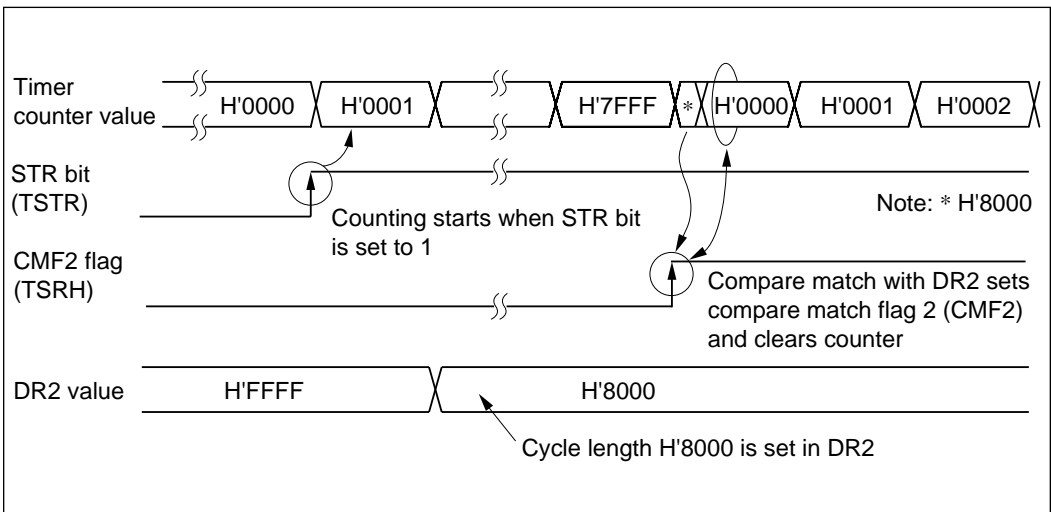
**Figure 11-15 Counter Operation**

A reset leaves the IPU in free-running mode. Figure 11-16 shows an example of free-running counting. The counter starts from H'0000, counts up to H'FFFF, then returns to H'0000, at which point the OVF flag is set in timer status register high (TSRH). Counting then continues from H'0000.

If compare match is selected as a counter clear source, the IPU operates in periodic counting mode. Figure 11-17 shows an example of periodic counting. The counter starts from H'0000 and counts up to H'8000. At this point a compare match with DR2 occurs, so the CMF2 flag in TSRH is set to 1 and the counter is automatically cleared. Counting then continues from H'0000.



**Figure 11-16 Free-Running Counter Operation**

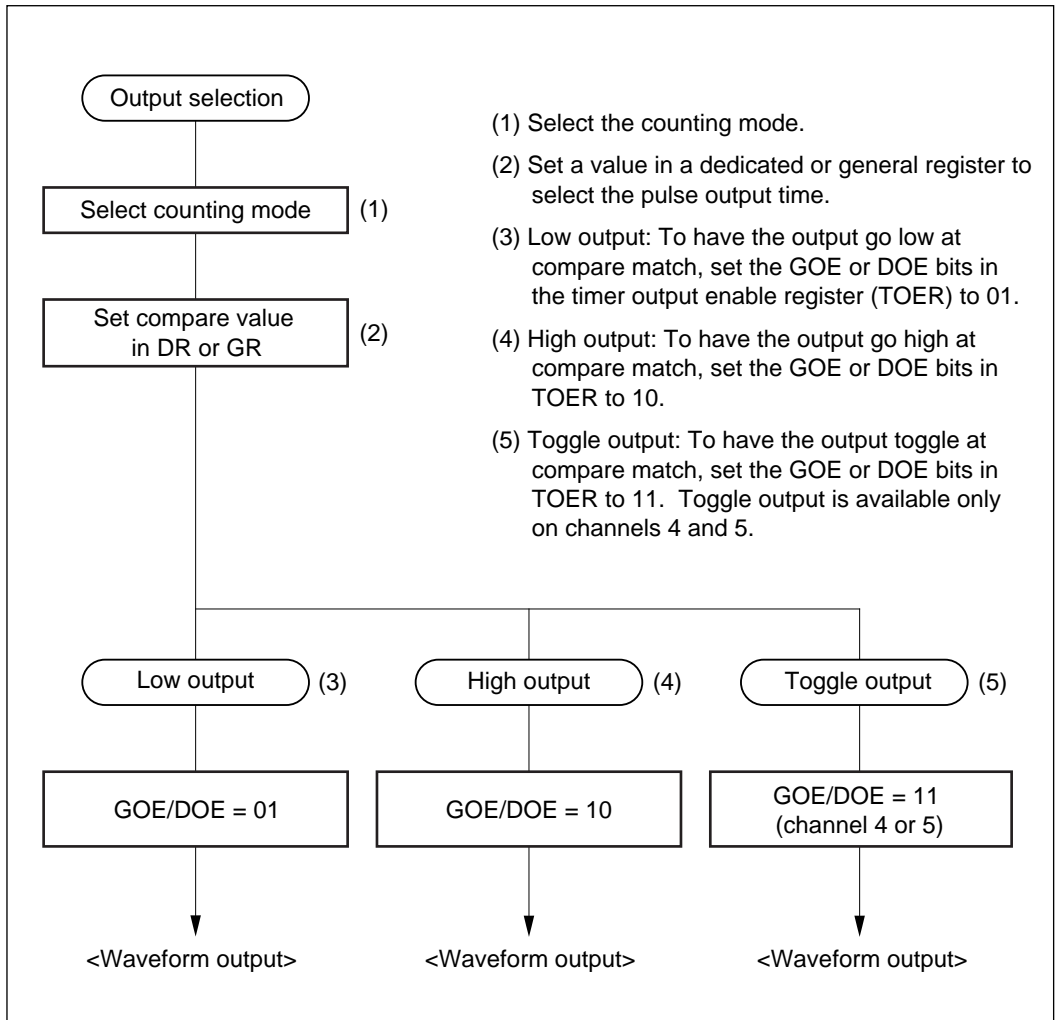


**Figure 11-17 Periodic Counter Operation**

## 11.8.2 Selection of Output Level

Compare match signals can be output in three modes: high, low, or toggle. Figure 11-18 shows the procedure for selecting the output level.

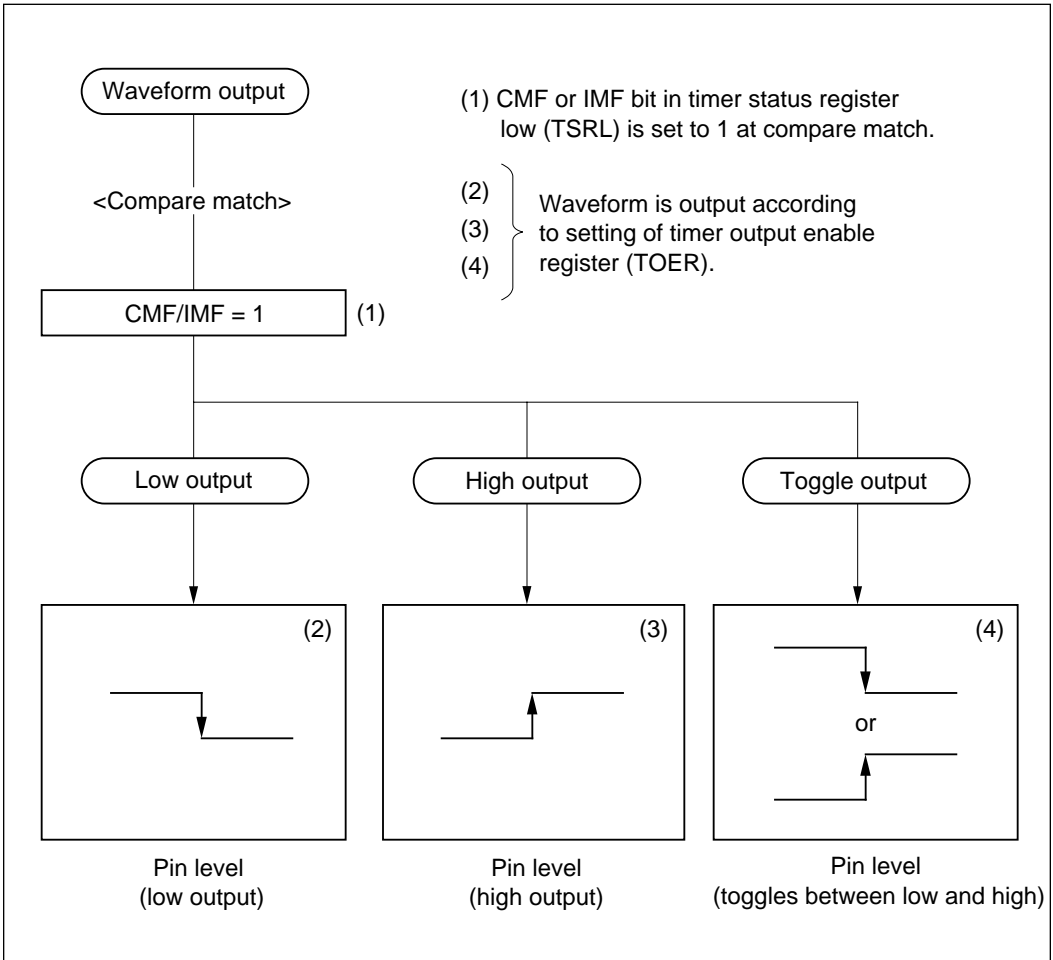
### Procedure for Selecting Output Level



**Figure 11-18 Procedure for Selecting Output Level**



**Waveform Output Operation:** Figure 11-19 illustrates waveform output operations.



**Figure 11-19 Waveform Output**

Figure 11-20 shows examples of waveform output on channel 4. High output is selected from T4IOC<sub>1</sub>, low output from T4IOC<sub>2</sub>, and toggle output from T4OC<sub>1</sub>.

High output is selected by setting bits GOE11 and GOE10 to 10 in the channel 4 timer output enable register (TOER). The IPU drives T4IOC<sub>1</sub> high when the counter matches the value in GR1 (H'0001). Low output is selected by setting bits GOE21 and GOE20 to 01 in the channel 4 TOER. The IPU drives T4IOC<sub>2</sub> low when the counter matches the value in GR2 (H'0003). Toggle output is selected by setting bits GOE11 and GOE10 to 11 in the channel 4 TOER. The IPU toggles T4OC<sub>1</sub> when the counter matches the value in DR1 (H'0004). The counter is cleared when the count matches the value in DR2 (H'00FF).

If high or low output is selected, when compare match occurs, and if the pin is already at the selected output level, the output level does not change.

- Settings
  - TOER (channel 4): H'36
  - TCRL (channel 4): H'E0 (clear on T4DR2 compare match)

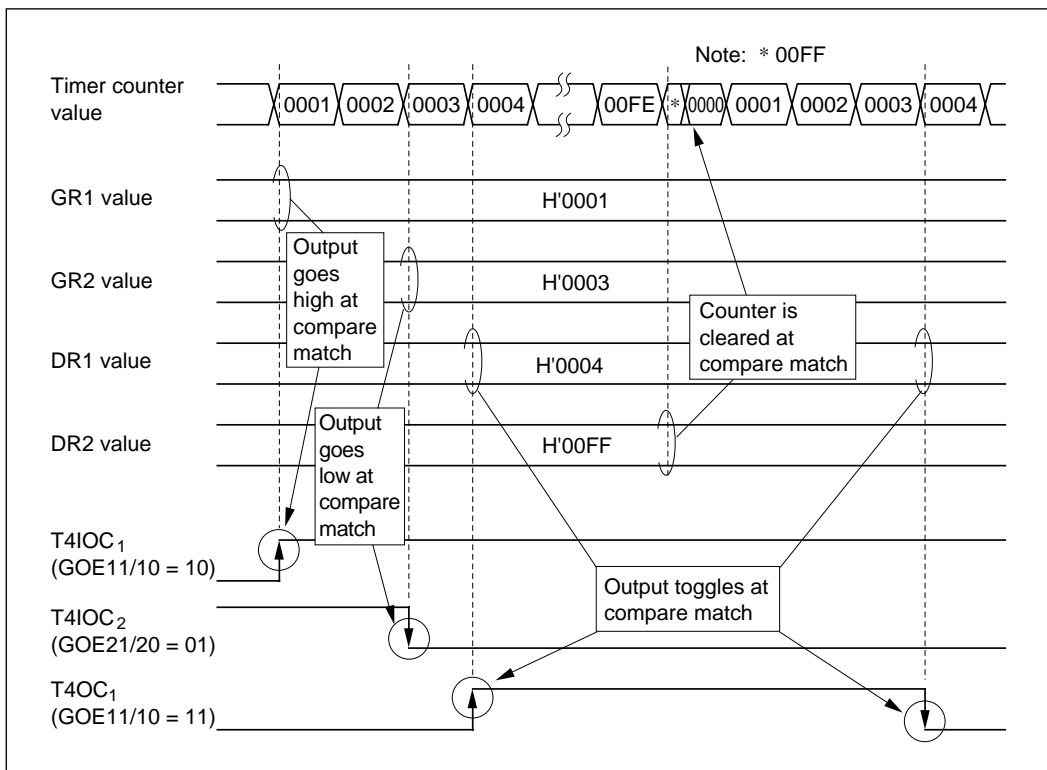


Figure 11-20 Example of Waveform Output on Channel 4

### 11.8.3 Input Capture Function

The counter value can be captured into a register when a transition occurs at an input capture pin. Capture can take place on the rising edge, falling edge, or both edges. Figure 11-21 shows the procedure for selecting the input capture function.

#### Procedure for Selecting Input Capture Mode

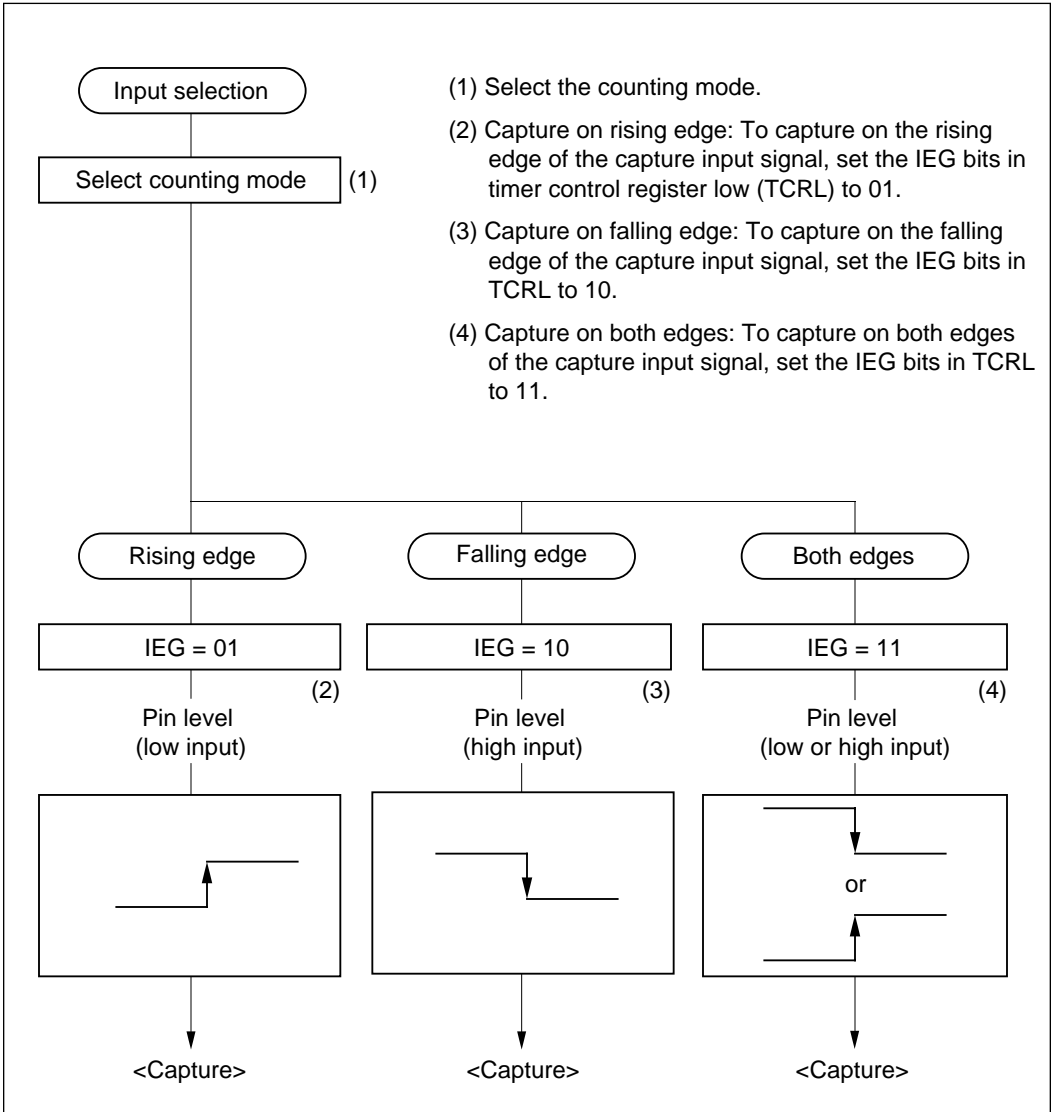
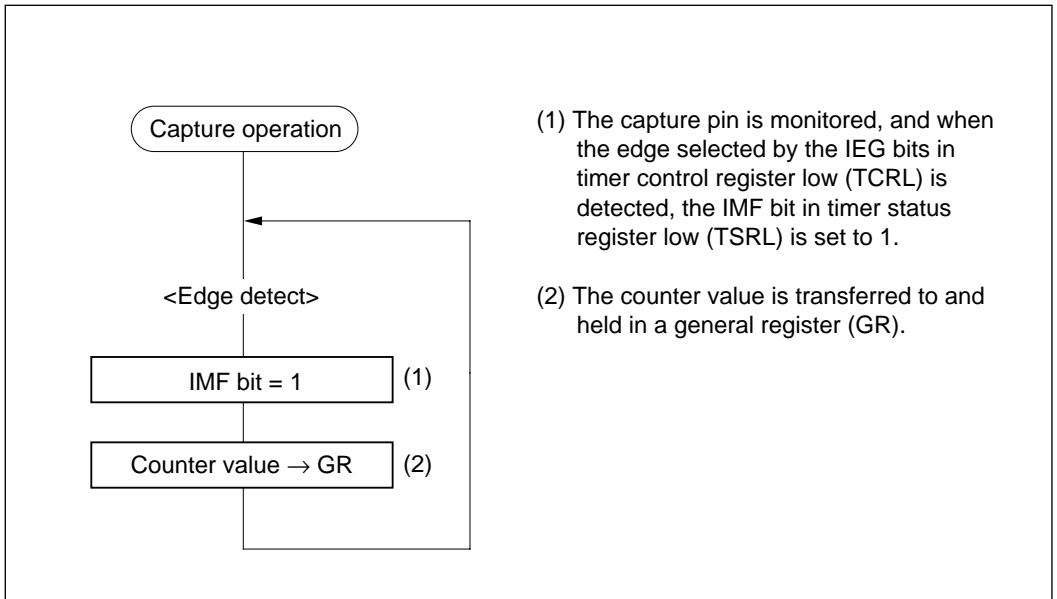


Figure 11-21 Procedure for Selecting Capture Input Mode

**Capture Operation:** Figure 11-22 illustrates input capture operations.



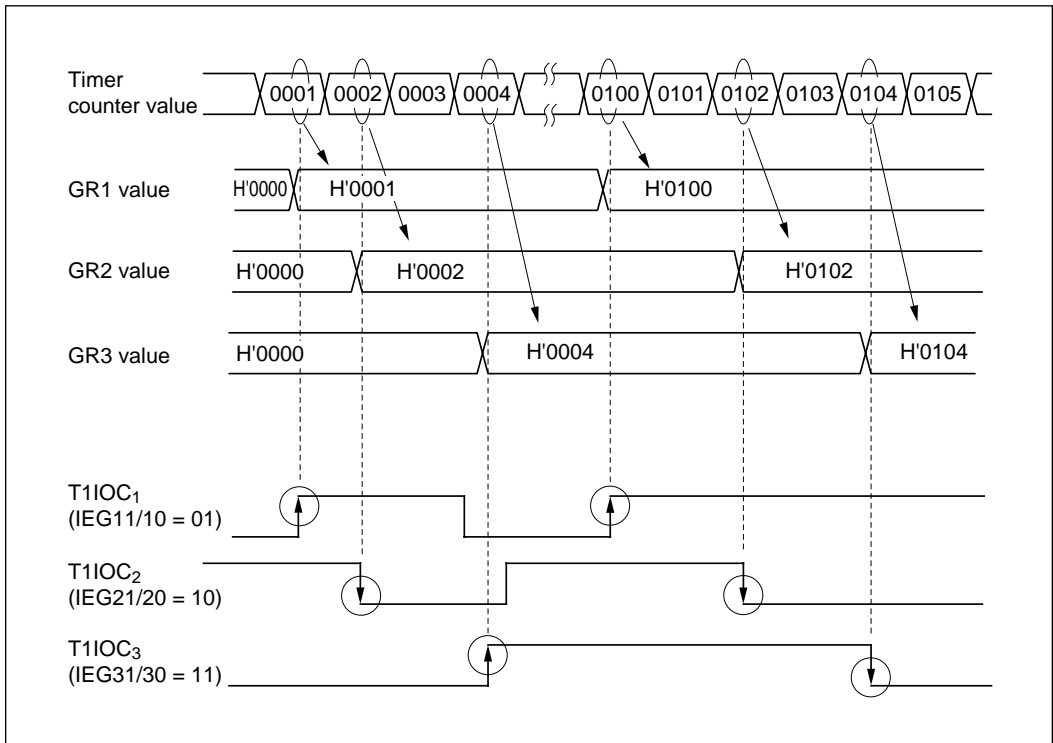
**Figure 11-22 Capture Mode Operation**

Figure 11-23 shows an example of input capture on channel 1.

The rising edge of T1IOC<sub>1</sub> is selected by setting bits IEG11 and IEG10 to 01 in channel 1 timer control register low (TCRL). The IPU transfers the counter value (H'0001 and H'0100) to GR1 on the rising edge of the T1IOC<sub>1</sub> input. The falling edge of T1IOC<sub>2</sub> is selected by setting bits IEG21 and IEG20 in channel 1 TCRL to 10. The IPU transfers the counter value (H'0002 and H'0102) to GR2 on the falling edge of the T1IOC<sub>2</sub> input. The rising and falling edges of T1IOC<sub>3</sub> are selected by setting bits IEG31 and IEG30 in channel 1 timer control register A (TCRA) to 11. The IPU transfers counter value H'0004 on the rising edge and value H'0104 on the falling edge of the T1IOC<sub>1</sub> input, to GR3.

- Settings

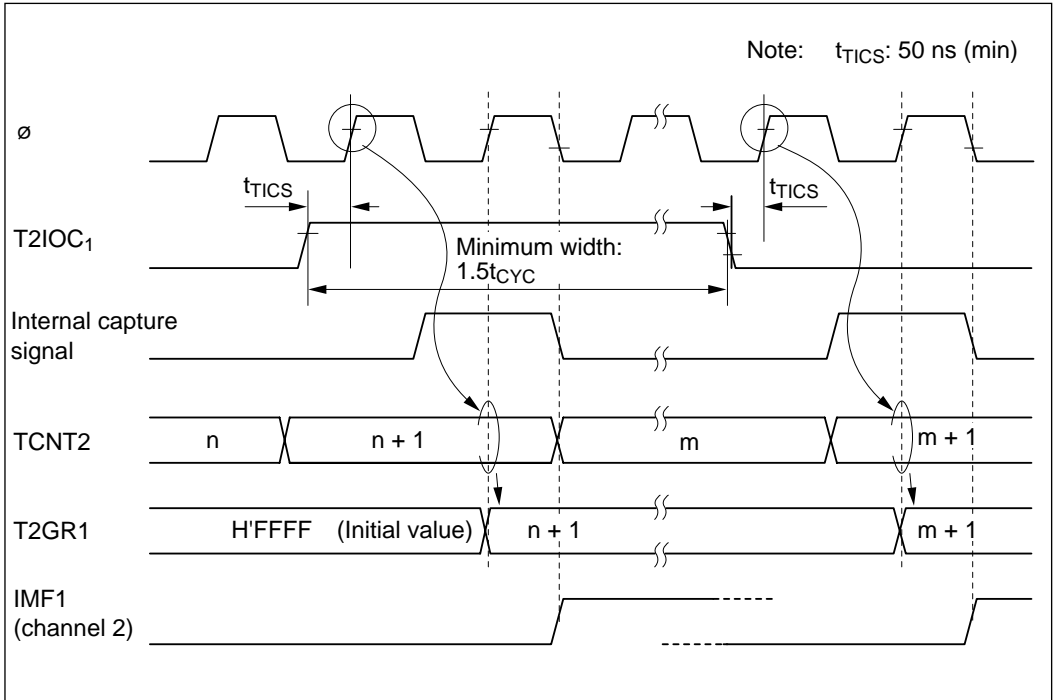
- TCRL: H'89
- TCRA: HF3



**Figure 11-23 Example of Input Capture on Channel 1**

Figure 11-24 shows an example of input capture timing on channel 2. The IPU latches the input capture signal input at the T2IOC<sub>1</sub> pin on the rising edge of the system clock ( $\phi$ ). One system clock cycle ( $1.0t_{CYC}$ ) after the input capture signal is latched, the counter value ( $n + 1$ ) is transferred to T2GR1. The IMF1 flag in timer status register low (TSRL) is set  $1.5t_{CYC}$  after the input capture signal is latched.

The pulse width of the input capture signal must be at least  $1.5t_{CYC}$ .

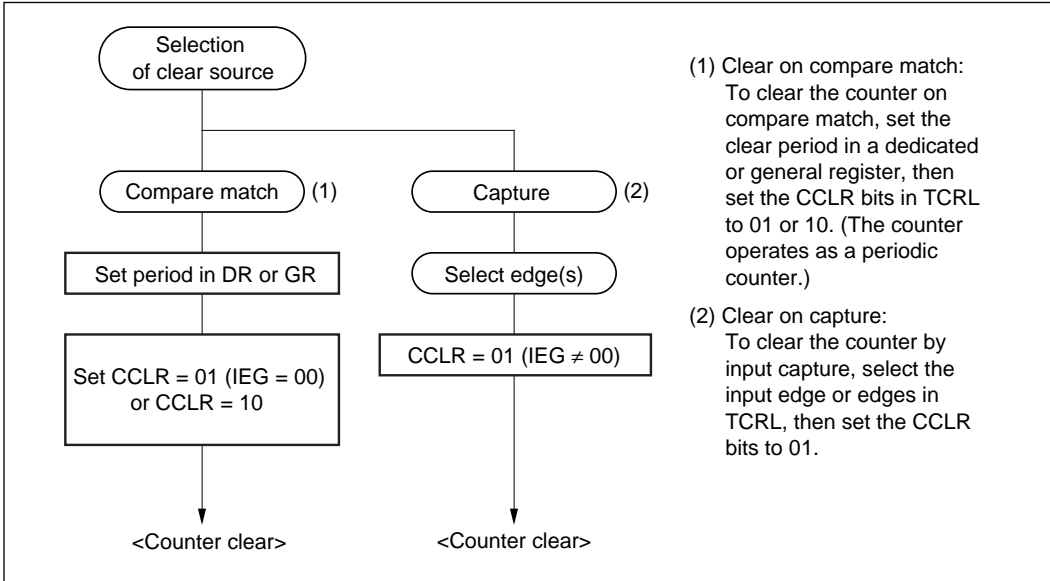


**Figure 11-24 Capture Input Timing**

### 11.8.4 Counter Clearing Function

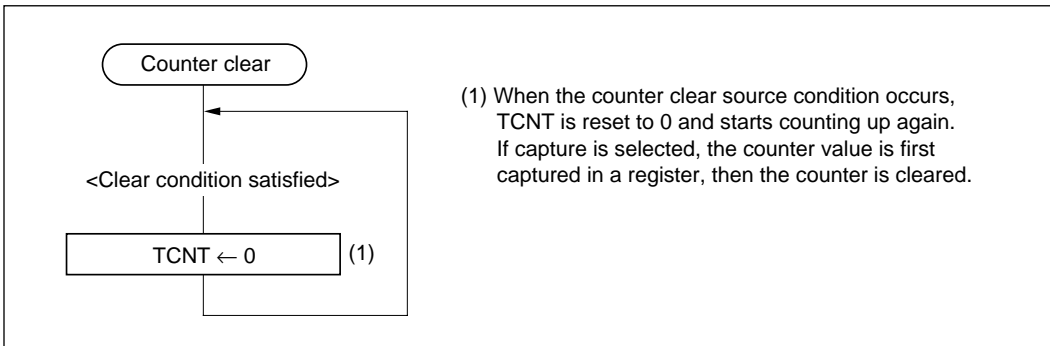
A counter can be cleared by input capture or compare match. When compare match is selected as a counter clear source, the count repeats cyclically from H'0000 to the value in the compare register. When input capture is selected as a counter clear source, the counter can be cleared at intervals determined by external events. Figure 11-25 shows the procedure for selecting the counter clear source.

#### Procedure for Selecting Counter Source



**Figure 11-25 Procedure for Selecting Counter Clear Source**

**Counter Clear Operation:** Figure 11-26 illustrates the counter clear operation.



**Figure 11-26 Counter Clearing Operation**

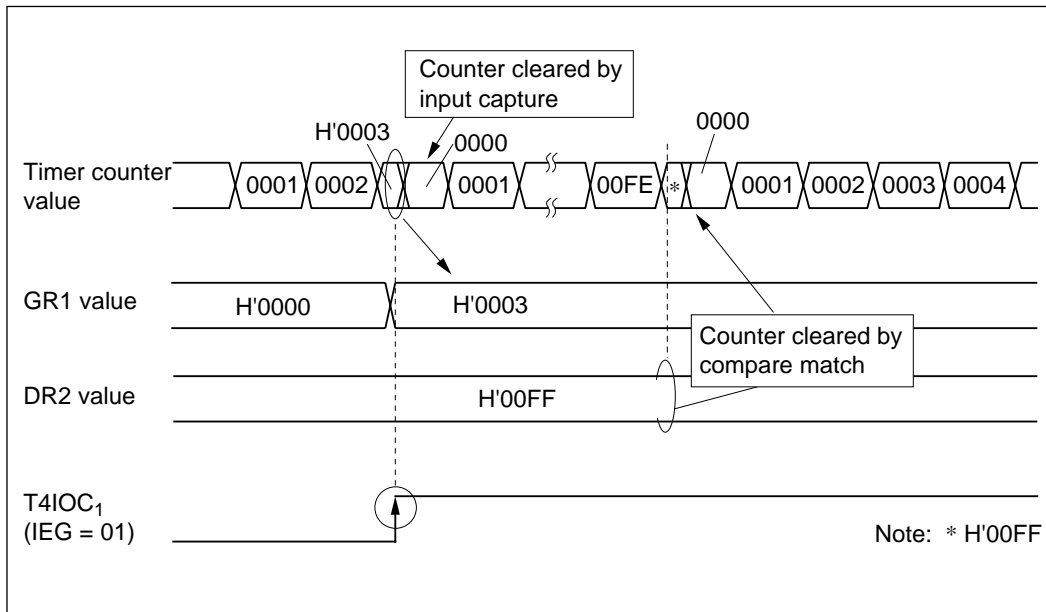
Figure 11-27 shows an example of counter clearing on channel 4.

In this example the channel-4 counter is cleared by input capture at  $T4IOC_1$ . This clear condition is selected by setting  $CCLR1$  and  $CCLR0$  in channel 4 timer control register low (TCRL) to 01. The rising edge is selected by setting  $IEG11$  and  $IEG10$  to 01. The IPU transfers the counter value (H'0003) on the rising edge of the  $T4IOC_1$  input to GR1, then clears the counter.

To clear the counter on DR2 compare match, set  $CCLR1$  and  $CCLR0$  to 10 in TCRL.

- Settings

- TCRL (channel 4): H'D4 (to clear on input capture in T4GR1)
- TCRL (channel 4): H'E0 (to clear on compare match with T4DR2)



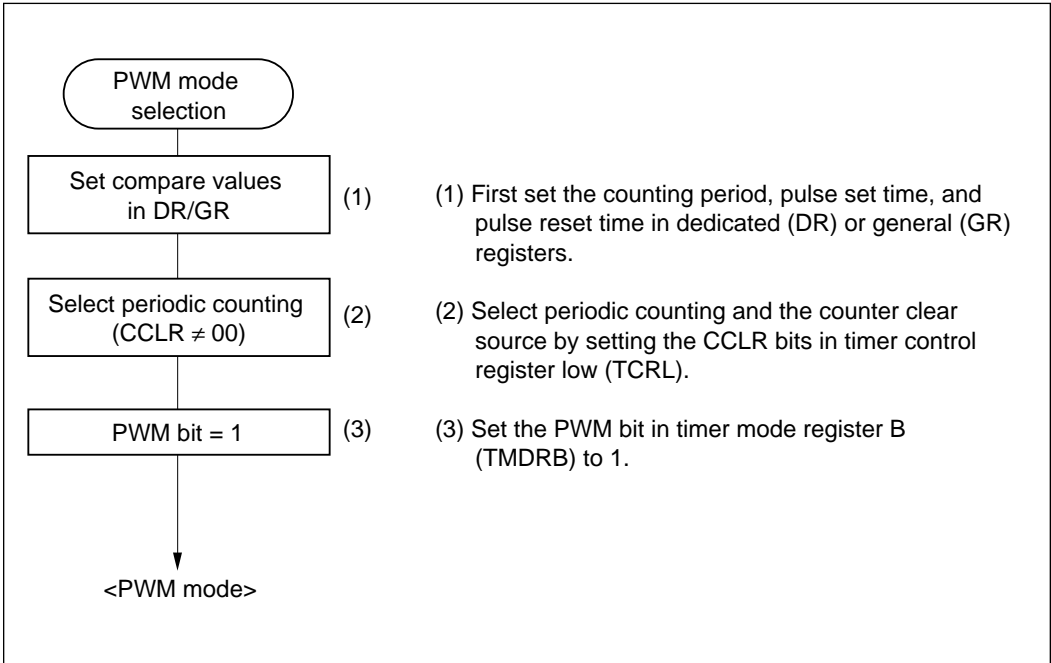
**Figure 11-27 Example of Input Counter Clearing on Channel 4**



### 11.8.5 PWM Output Mode

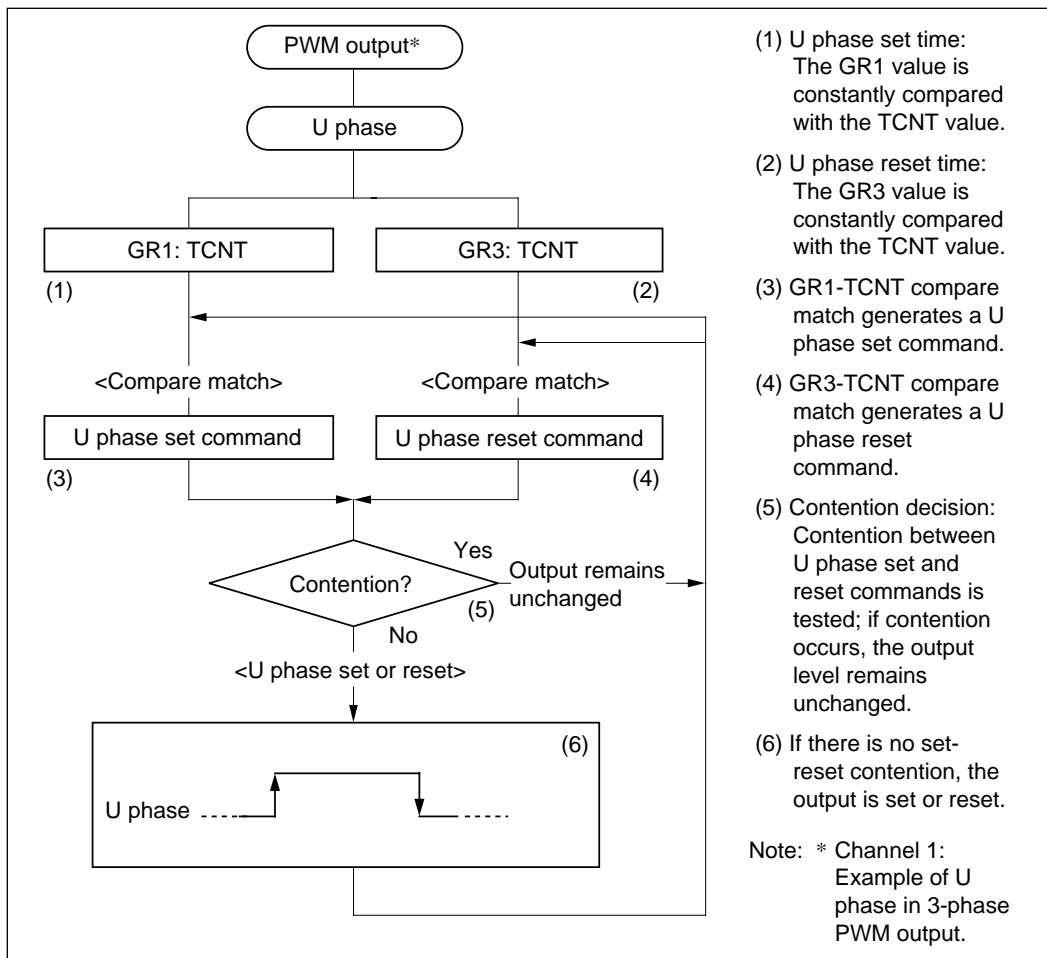
Channels 1, 2, 3, 6, and 7 can be used as PWM timers. Channel 1 can provide three-phase PWM output, channels 2 and 3 can provide two-phase PWM output, and channels 6 and 7 can provide single-phase PWM output. Figure 11-28 shows the procedure for selecting PWM output mode.

#### Procedure for Selecting PWM Mode



**Figure 11-28 Procedure for Selecting PWM Output Mode**

**PWM Output Operation:** Figure 11-29 illustrates PWM output operations.



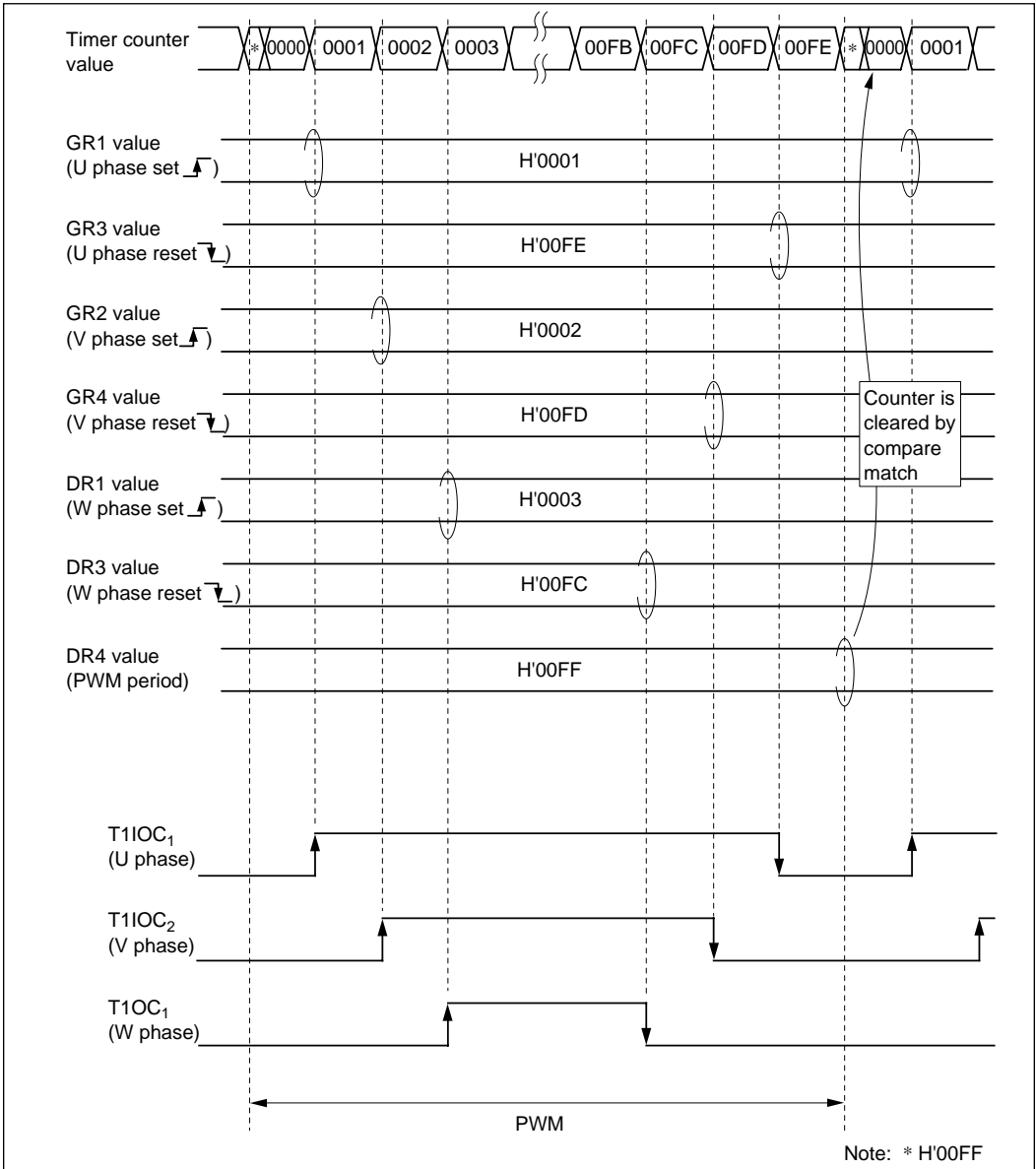
**Figure 11-29 PWM Output Operation**

Figure 11-30 shows an example of three-phase PWM output on channel 1.

The U phase is output at the T1IOC<sub>1</sub> pin. The V phase is output at the T1IOC<sub>2</sub> pin. The W phase is output at the T1IOC<sub>3</sub> pin. The IPU sets T1IOC<sub>1</sub> when the timer counter matches GR1 (H'0001), and resets T1IOC<sub>1</sub> when the timer counter matches GR3 (H'00FE). The IPU sets T1IOC<sub>2</sub> when the timer counter matches GR2 (H'0002), and resets T1IOC<sub>2</sub> when the timer counter matches GR4 (H'00FD). The IPU sets T1IOC<sub>3</sub> when the timer counter matches DR1 (H'0003), and resets T1IOC<sub>3</sub> when the timer counter matches DR3 (H'00FC).

The IPU clears the counter when the timer counter matches DR4 (H'00FF).

- Settings
  - TMDRB: H'C1 (PWM output on channel 1)
  - TCRL: H'F0 (clear on T1DR4 compare match)
  - TCRA: H'F0



**Figure 11-30 Example of Three-Phase PWM Output on Channel 1**

In PWM mode the compare registers are paired: one register sets the pulse; the other register resets the pulse. The counter should be set to periodic counting mode. Table 11-7 indicates the register pair assigned to each output pin.

**Table 11-7 Output Pins and Register Pairs**

Channel	Output Pin	Set	Reset	PWM Period
1	T1IOC <sub>1</sub>	GR1	GR3	DR2, GR3, DR4
	T1IOC <sub>2</sub>	GR2	GR4	
	T1OC <sub>1</sub>	DR1	DR3	
2	T2IOC <sub>1</sub>	GR1	DR1	DR2
	T2IOC <sub>2</sub>	GR2	DR2	
3	T3IOC <sub>1</sub>	GR1	DR1	DR2
	T3IOC <sub>2</sub>	GR2	DR2	
6	T6IOC <sub>1</sub>	GR1	GR2	GR2
7	T7IOC <sub>1</sub>	GR1	GR2	GR2

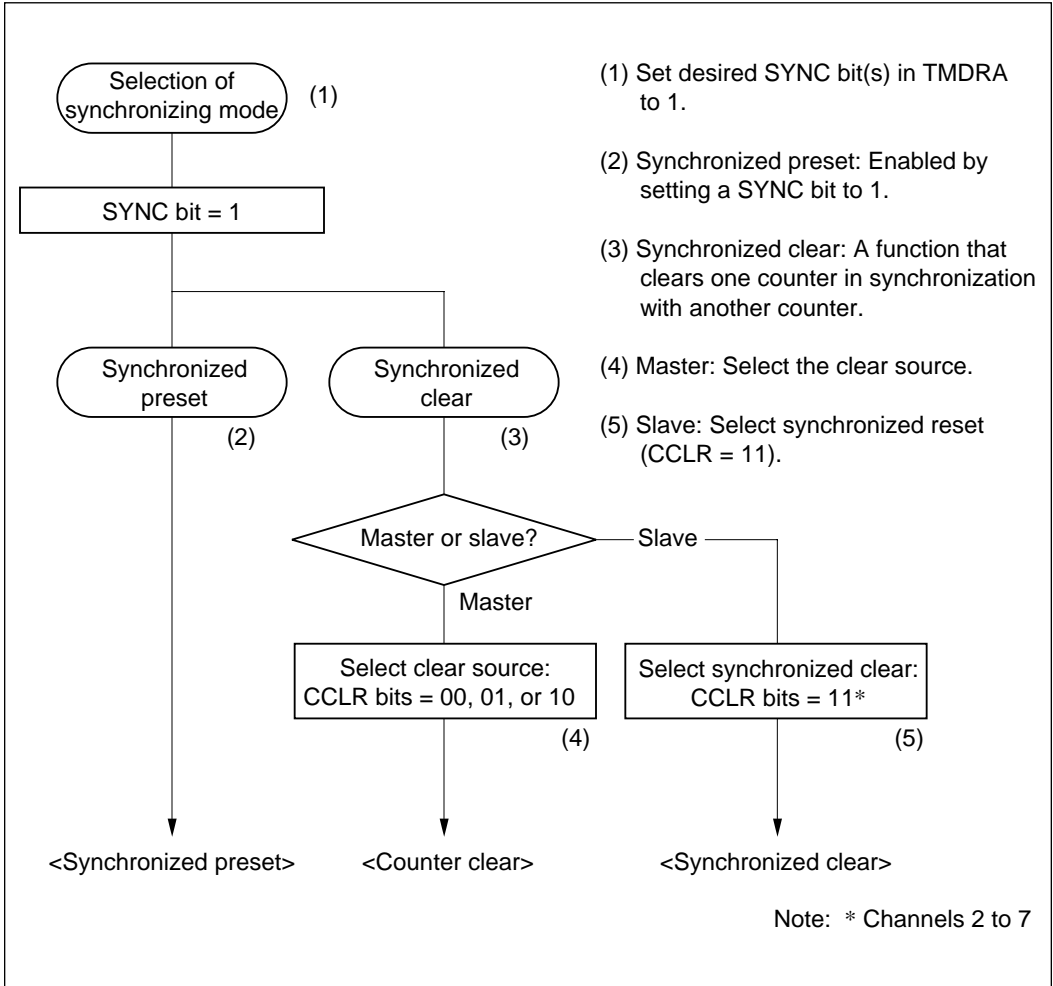
### Usage Notes

1. In PWM output mode, the output levels of PWM output pins cannot be set in the timer output enable register (TOER). Any output level settings made will be ignored.
2. Settings of the IEG bits in timer control register low (TCRL) are valid in PWM output mode. The IEG bits must be cleared to 0.
3. In PWM output mode, periodic counting should be used by selecting a counter clear source in TCRL. Table 11-7 lists the registers that can set the PWM period in each channel.

## 11.8.6 Synchronizing Mode

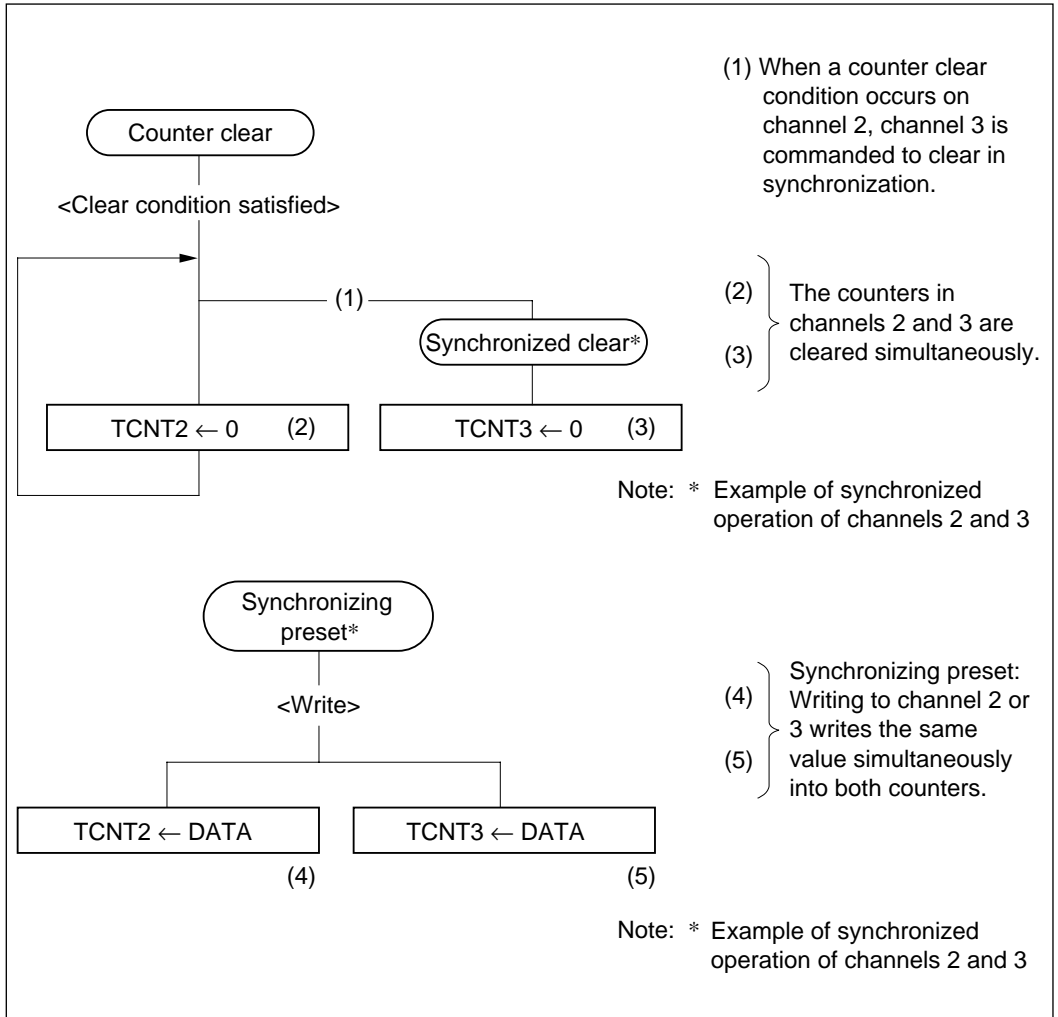
In synchronizing mode two or more timer counters can be rewritten or cleared simultaneously. Figure 11-31 shows the procedure for selecting synchronizing mode.

### Procedure for Selecting Synchronizing Mode



**Figure 11-31 Procedure for Selecting Synchronizing Mode**

**Synchronized Operation:** Figure 11-32 shows an example of synchronized operation of channels 2 and 3.



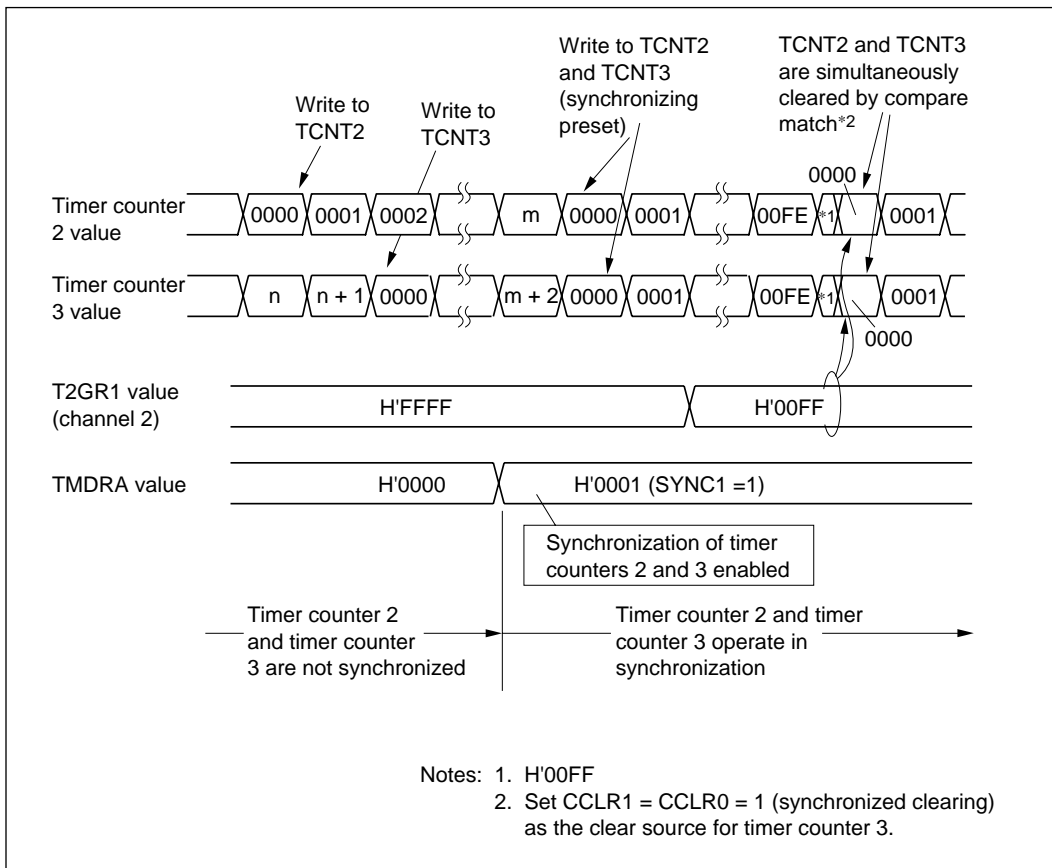
**Figure 11-32 Example of Synchronized Operation of Channels 2 and 3**

Figure 11-33 shows an example of the synchronization of timer counters 2 and 3.

Timer counters 2 and 3 are synchronized by setting the SYNC1 bit in timer mode register A (TMDRA) to 1. The timer counters are synchronously preset by writing a new value to either timer counter 2 or 3; the IPU simultaneously writes the same value in the other timer counter. Synchronized clearing is selected by setting CCLR0 = CCLR1 = 1 in timer control register low (TCRL) as the clear source for timer counter 3. The IPU clears timer counters 2 and 3 simultaneously when timer counter 2 matches T2GR1 (H'00FF).

- Settings

- T2GR1: H'00FF
- TMDRA: H'02 (SYNC1 = 1)
- TCRL (channel 2): H'D0 (clear at compare match with T2GR1)
- TCRL (channel 3): H'F0 (enabling synchronized clearing)

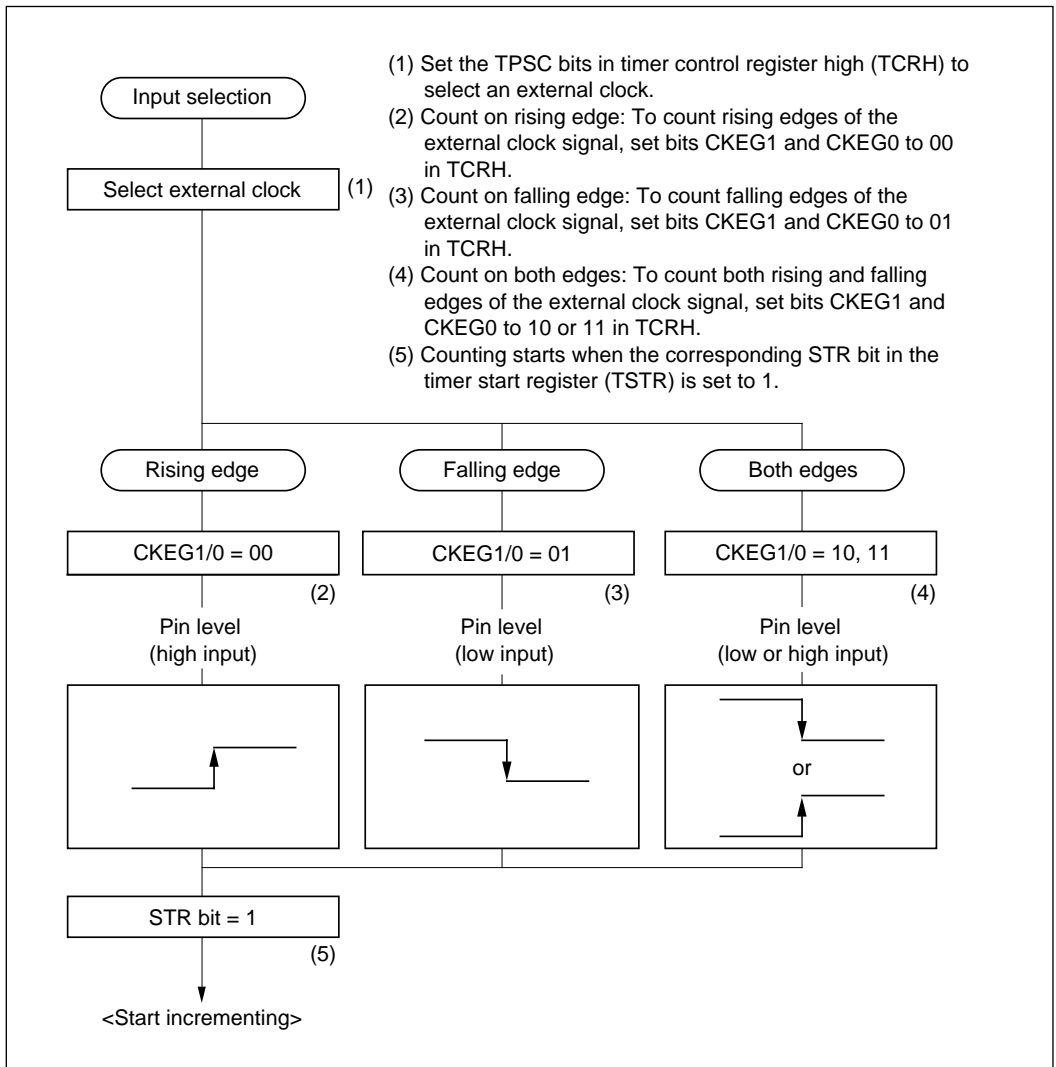


**Figure 11-33 Example of Synchronization of Timer Counters 2 and 3**

## 11.8.7 External Event Counting

The IPU has three external clock input pins. If external event signals are input at these external clock input pins, external events can be counted. The counter can be set to increment on the rising or falling edge, or on both edges of the external clock signal. The value of an externally clocked counter can be captured at regular intervals to measure external event frequencies. Figure 11-34 shows the procedure for selecting external event counting mode.

### Procedure for Selecting External Event Counting Mode



**Figure 11-34 Procedure for Selecting External Event Counting Mode**

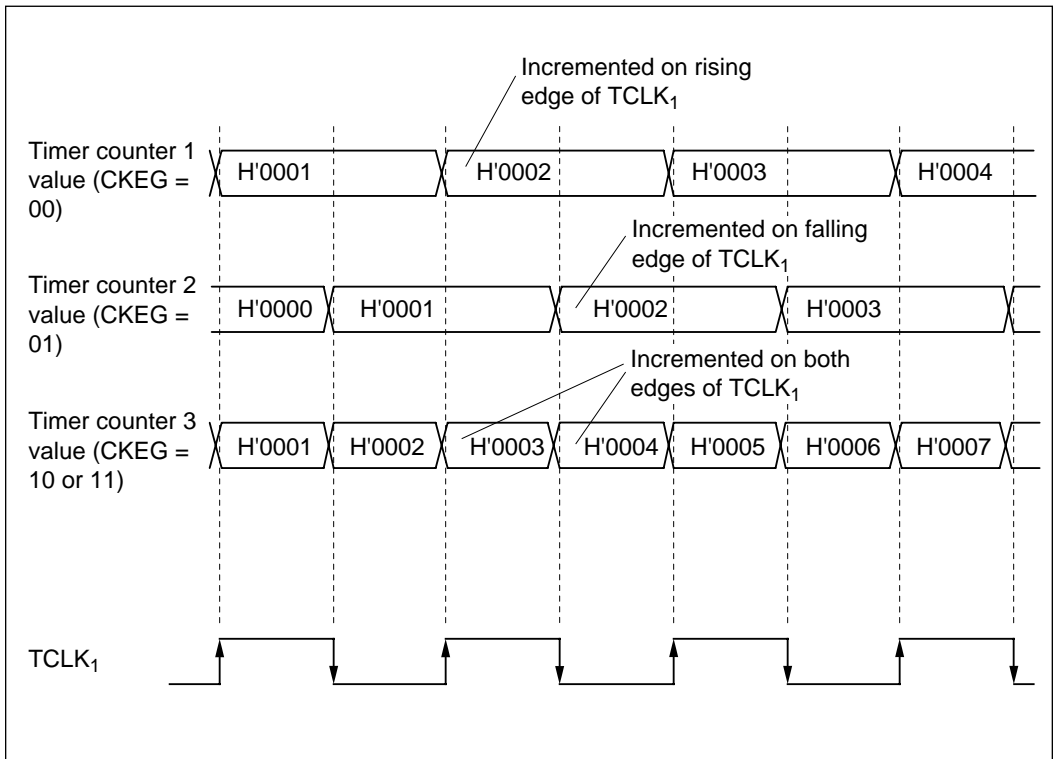


**External Event Counting Operation:** Counting operations are the same as for an internal clock. For details, see section 11.8.1, “Examples of Counting.”

Figure 11-35 shows an example of external event counting.

In this example timer counters 1, 2, and 3 count external event inputs at  $TCLK_1$ . In channel 1, the rising edge of  $TCLK_1$  is selected by setting the CKEG1 and CKEG0 bits in TCRH to 00. The IPU counts rising edges of  $TCLK_1$ . In channel 2, the falling edge of  $TCLK_1$  is selected by setting the CKEG1 and CKEG0 bits in TCRH to 01. The IPU counts falling edges of  $TCLK_1$ . In channel 3, both edges of  $TCLK_1$  are selected by setting the CKEG1 and CKEG0 bits in TCRH to 10 or 11. The IPU counts both rising and falling edges of  $TCLK_1$ .

- Settings
  - TCRH (channel 1): H'CD (count rising edges)
  - TCRH (channel 2): H'DD (count falling edges)
  - TCRH (channel 3): H'ED or H'FD (count both rising and falling edges)



**Figure 11-35 Example of External Event Counting**

Figure 11-36 shows an example of external clock input timing.

The IPU latches external clock signals (TCLK<sub>1</sub> to TCLK<sub>3</sub>) on the rising edge of the system clock ( $\phi$ ). TCNT2 is incremented 1.5 system clock cycles ( $1.5t_{CYC}$ ) after the external clock is latched. The pulse width of the external clock signal must be at least  $1.5t_{CYC}$ .

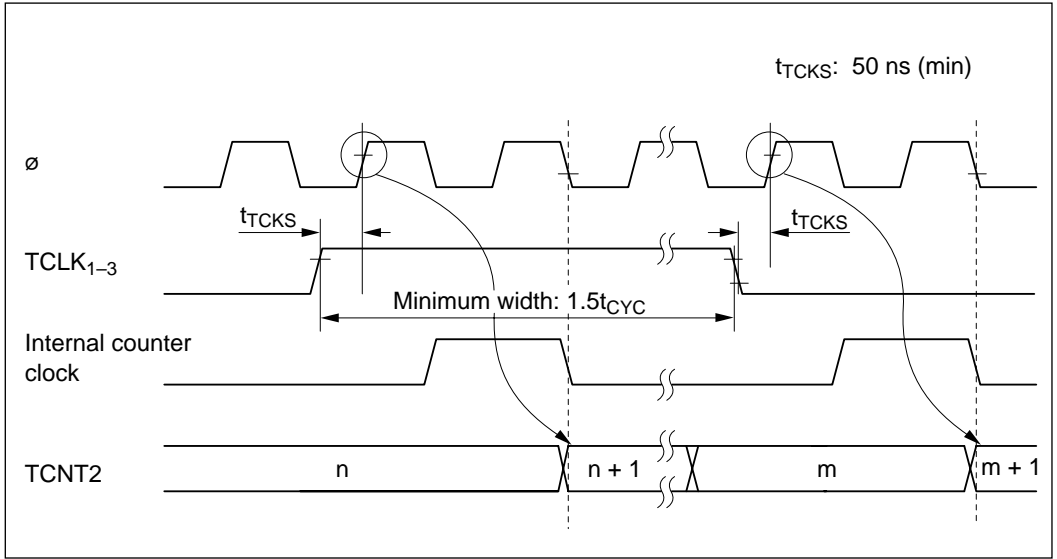
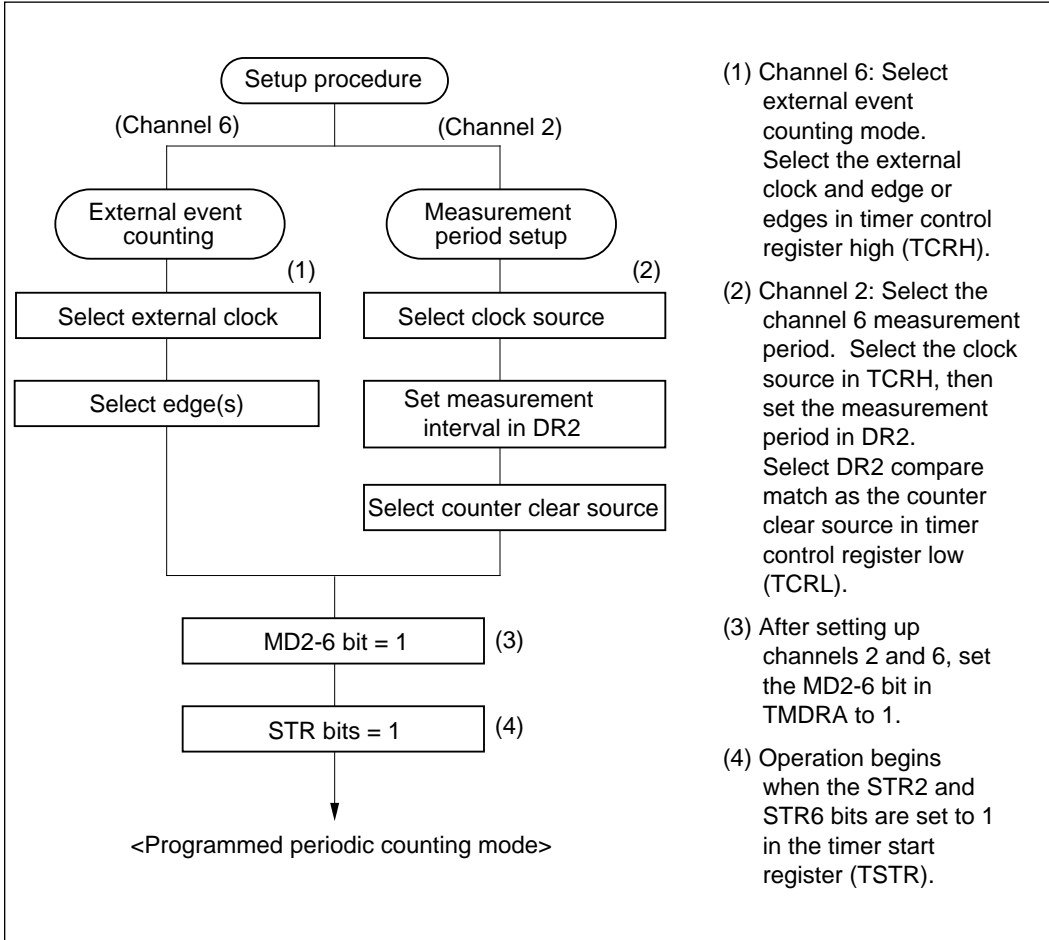


Figure 11-36 External Clock Input Timing

## 11.8.8 Programmed Periodic Counting Mode

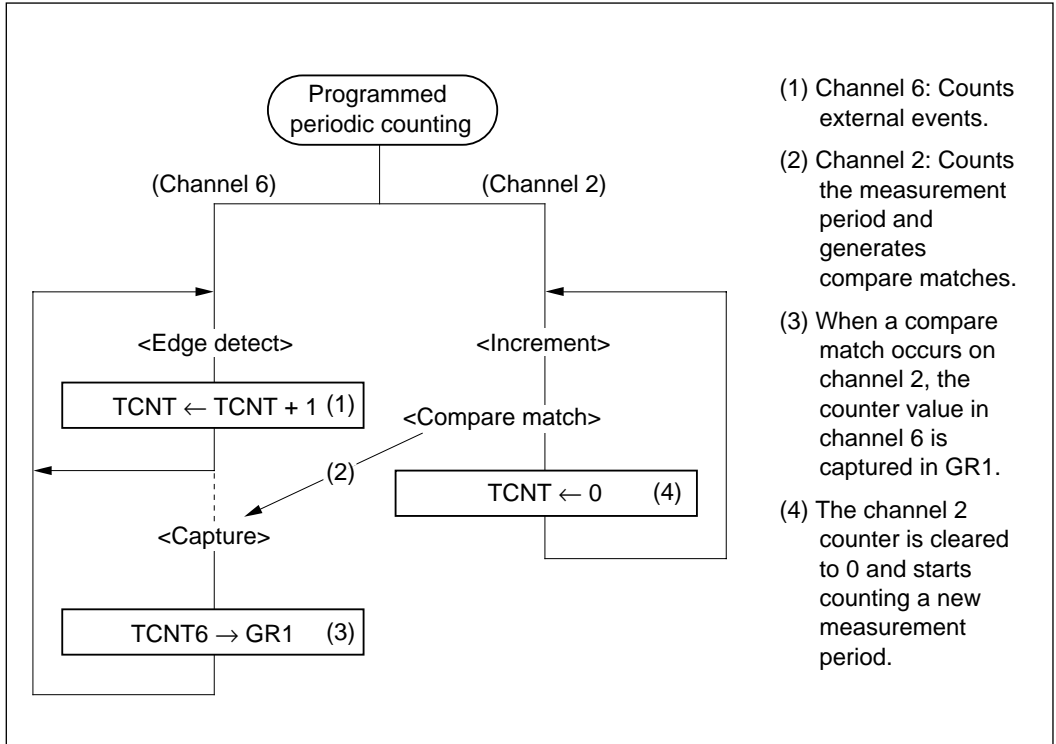
In programmed periodic counting mode, the value of an externally clocked counter is captured into a general register by compare match on a different channel. No external input capture signal is needed. Figure 11-37 shows the procedure for selecting programmed periodic counting mode.

**Procedure for Selecting Programmed Periodic Counting Mode :** Example when bit MD2-6 = 1 in timer mode register A (TMDRA)



**Figure 11-37 Procedure for Selecting Programmed Periodic Counting Mode**

**Programmed Periodic Counting Operation:** Figure 11-38 shows the programmed periodic counting operation.



- (1) Channel 6: Counts external events.
- (2) Channel 2: Counts the measurement period and generates compare matches.
- (3) When a compare match occurs on channel 2, the counter value in channel 6 is captured in GR1.
- (4) The channel 2 counter is cleared to 0 and starts counting a new measurement period.

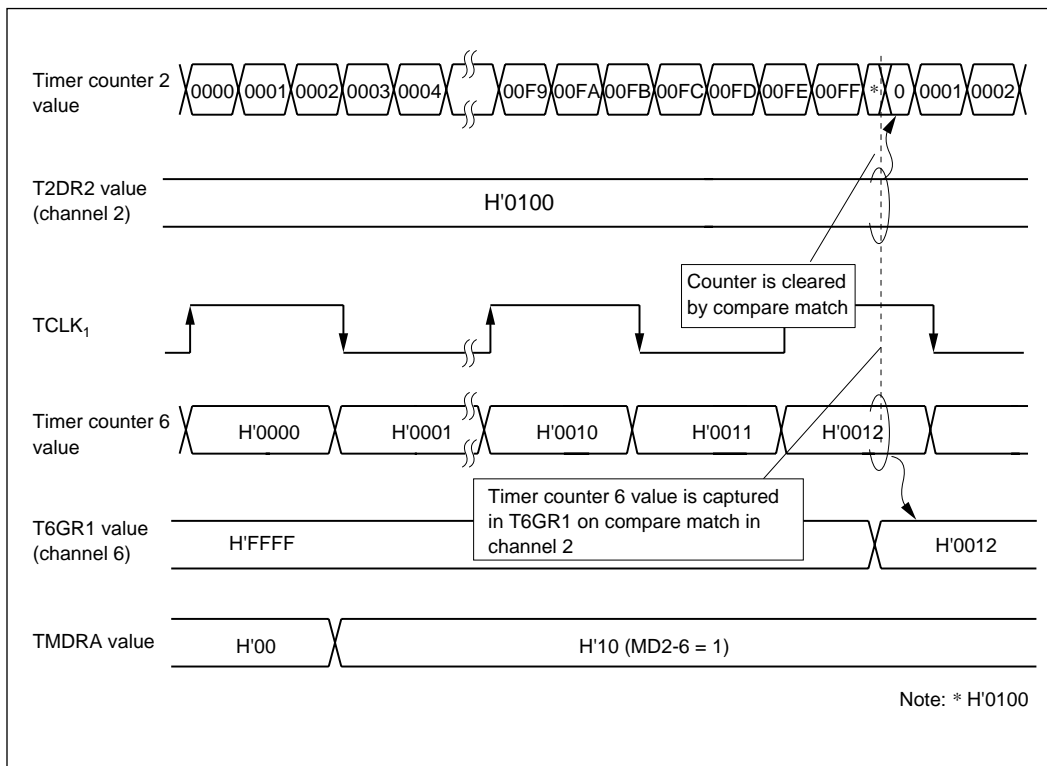
**Figure 11-38 Operation in Programmed Periodic Counting Mode**

Figure 11-39 shows an example of programmed periodic counting. Table 11-8 lists the possible combinations of compare-match channels and capture channels.

In this example external events are counted over a programmed period using channels 2 and 6. The IPU automatically transfers the value of timer counter 6 (H'0012) to T6GR1 when timer counter 2 matches T2DR2 (H'0100). Timer counter 2 is set to be cleared by compare match with T2DR2.

- Settings

- TCRL (channel 2): H'E0 (cleared by compare match with T2DR2)
- TCRH (channel 6): H'ED or H'FD (increment on both rising and falling edges)
- TMDRA: H'10 (capture in T6GR1 on compare match with T2DR2)



**Figure 11-39 Example of Programmed Periodic Counting**

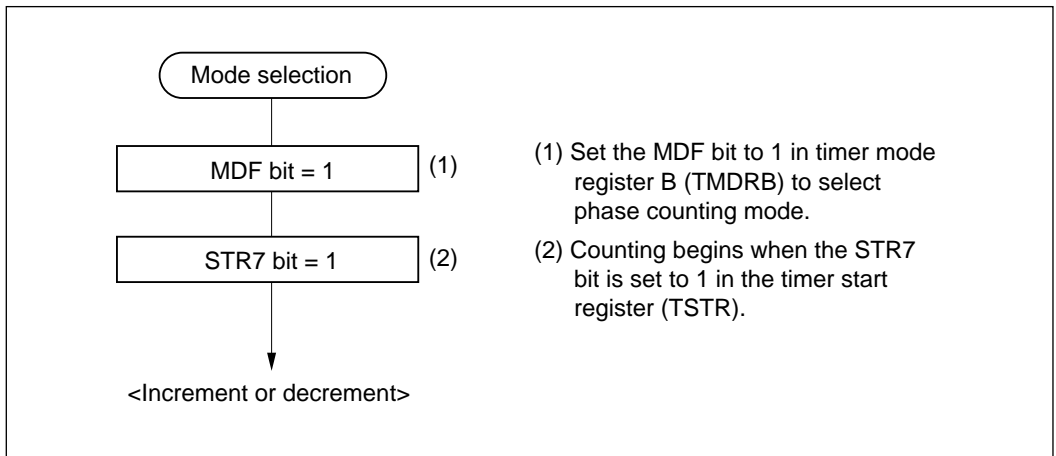
**Table 11-8 Combinations of Compare Match Channels and Capture Channels**

	Compare Match Channel		Capture Channel	
	Channel No.	Register	Channel No.	Register
MD2-6	Channel 2	DR2	Channel 6	GR1
MD3-5	Channel 3	DR2	Channel 5	GR1
MD4-7	Channel 4	DR2	Channel 7	GR1
MD6-7	Channel 6	GR2	Channel 7	GR2

### 11.8.9 Phase Counting Mode

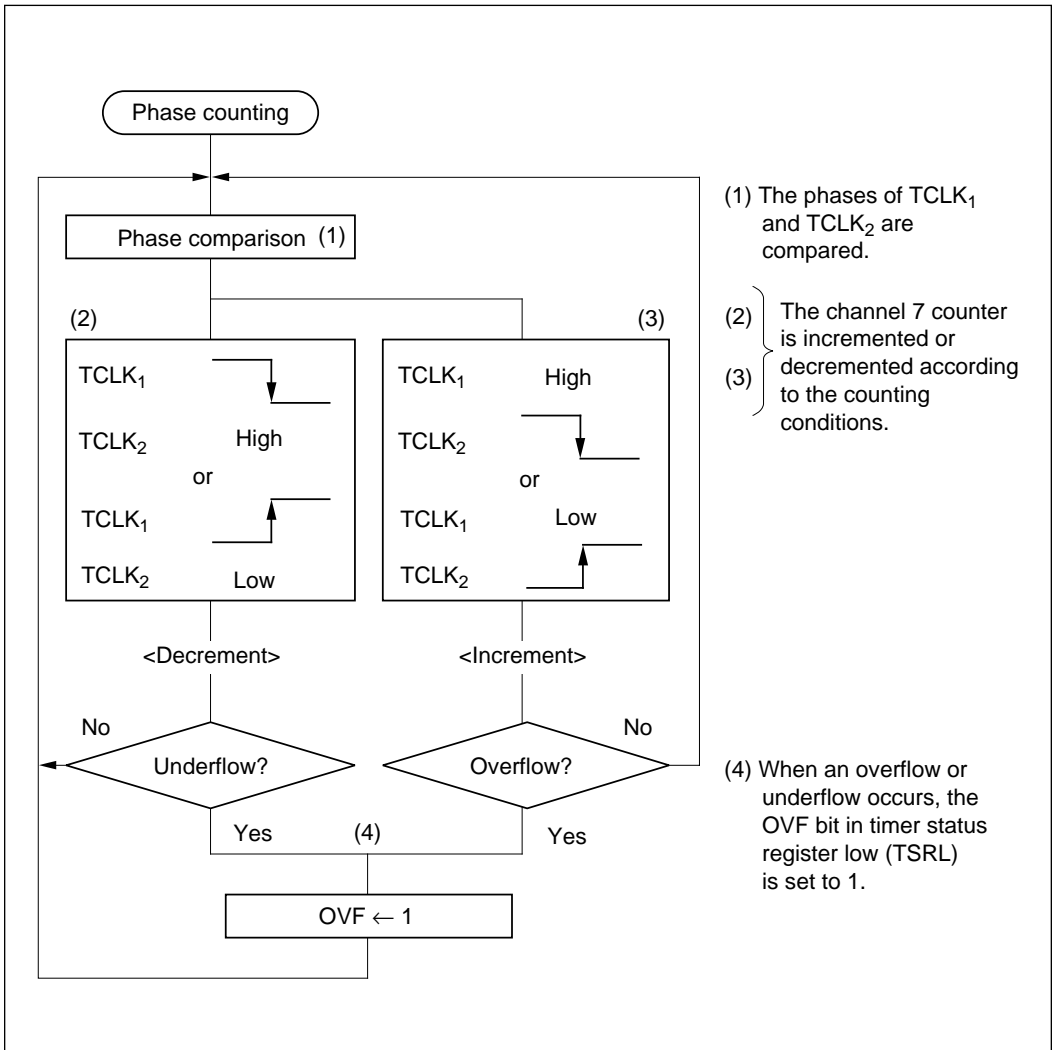
One application of phase counting mode is control of an AC servo motor. If the output of a two-phase encoder is fed to two external clock pins, the phase relationship between the two clock signals is detected and the counter is incremented or decremented accordingly. Phase counting is available only on channel 7. Figure 11-40 shows the procedure for selecting phase counting mode.

#### Procedure for Selecting Phase Counting Mode



**Figure 11-40 Procedure for Selecting Phase Counting Mode**

**Phase Counting Operation:** Figure 11-41 shows the phase counting operation.



(1) The phases of TCLK<sub>1</sub> and TCLK<sub>2</sub> are compared.

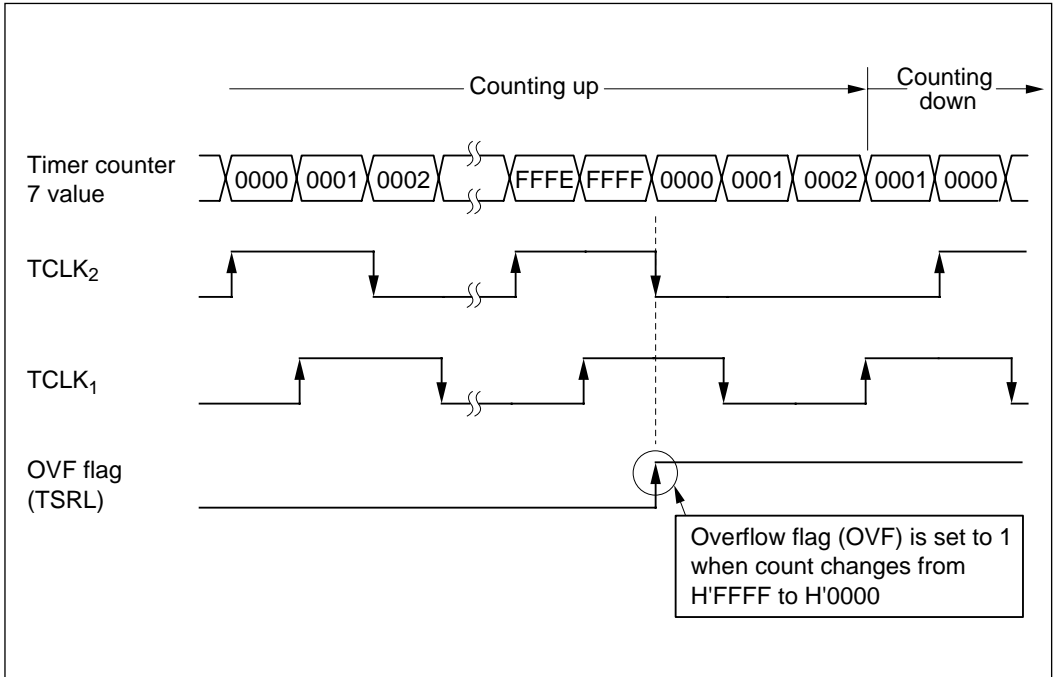
(2) } The channel 7 counter is incremented or decremented according to the counting conditions.  
 (3) }

(4) When an overflow or underflow occurs, the OVF bit in timer status register low (TSRL) is set to 1.

**Figure 11-41 Operation in Phase Counting Mode**

Figure 11-42 shows an example in which the counter counts up, overflows, then counts down.

In up-counting, the counter counts repeatedly from H'0000 to H'FFFF. The IPU sets the overflow flag (OVF) in timer status register low (TSRL) when the count returns from H'FFFF to H'0000. For the up/down counting conditions, see figure 11-44 “Counting Conditions” and table 11-9 “Up/Down-Counting Conditions.”

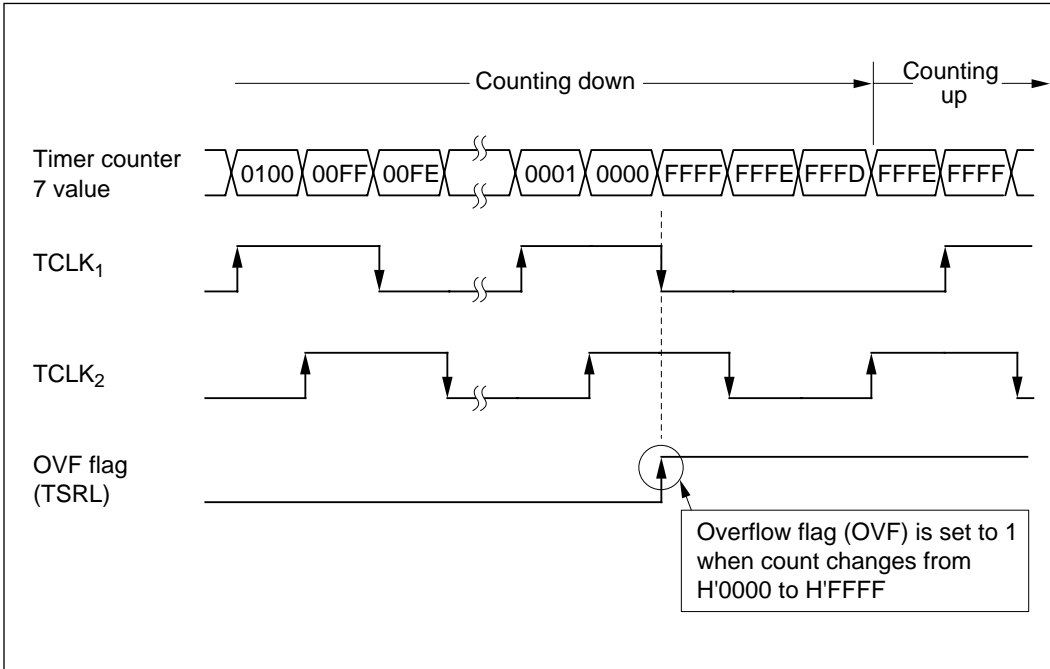


**Figure 11-42 Example of Up-Counting, Overflow, and Down-Counting**



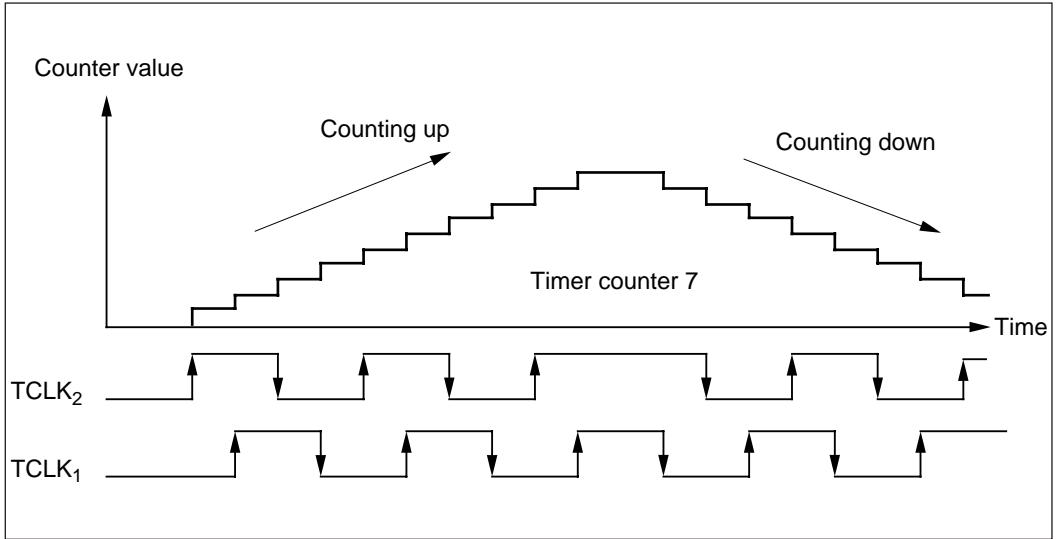
Figure 11-43 shows an example in which the counter counts down, underflows, then counts up.

In down-counting, the counter counts repeatedly from H'FFFF to H'0000. The IPU sets the overflow flag (OVF) in timer status register low (TSRL) when the count returns from H'0000 to H'FFFF. For the up/down counting conditions, see figure 11-44 “Counting Conditions” and table 11-9, “Up/Down-Counting Conditions.”



**Figure 11-43 Example of Down-Counting, Underflow, and Up-Counting**

Figure 11-44 shows the counting conditions. Table 11-9 indicates the up- and down-counting conditions. The IPU counts all edges of  $TCLK_1$  and  $TCLK_2$ .



**Figure 11-44 Counting Conditions**

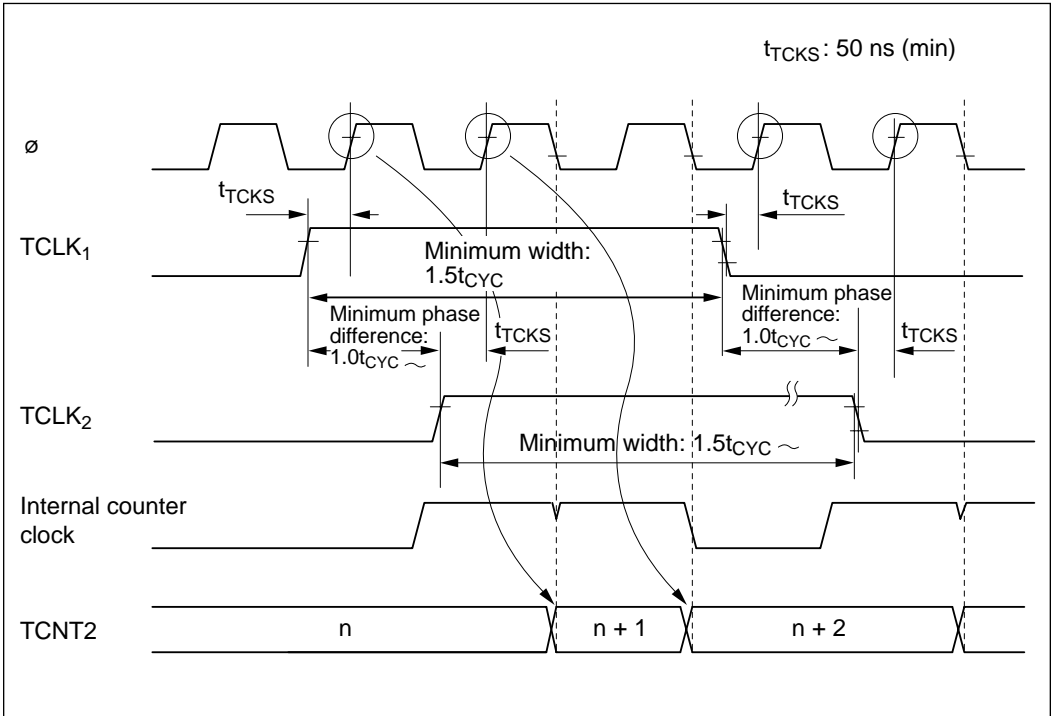
**Table 11-9 Up/Down-Counting Conditions**

Counting Direction	Up-Counting				Down-Counting			
	$TCLK_2$		High		Low		Low	
$TCLK_1$	Low		High		High		Low	

Figure 11-45 shows the external clock input timing in phase counting mode.

The IPU latches the external clock signals on the rising edge of the system clock ( $\emptyset$ ). The counter is incremented 1.5 system clock cycles ( $1.5t_{CYC}$ ) after the external clock is latched.

The external clock pulse width must be at least 1.5 system clock cycles ( $1.5t_{CYC}$ ). The phase difference between  $TCLK_1$  and  $TCLK_2$  must be at least  $1.0t_{CYC}$ .



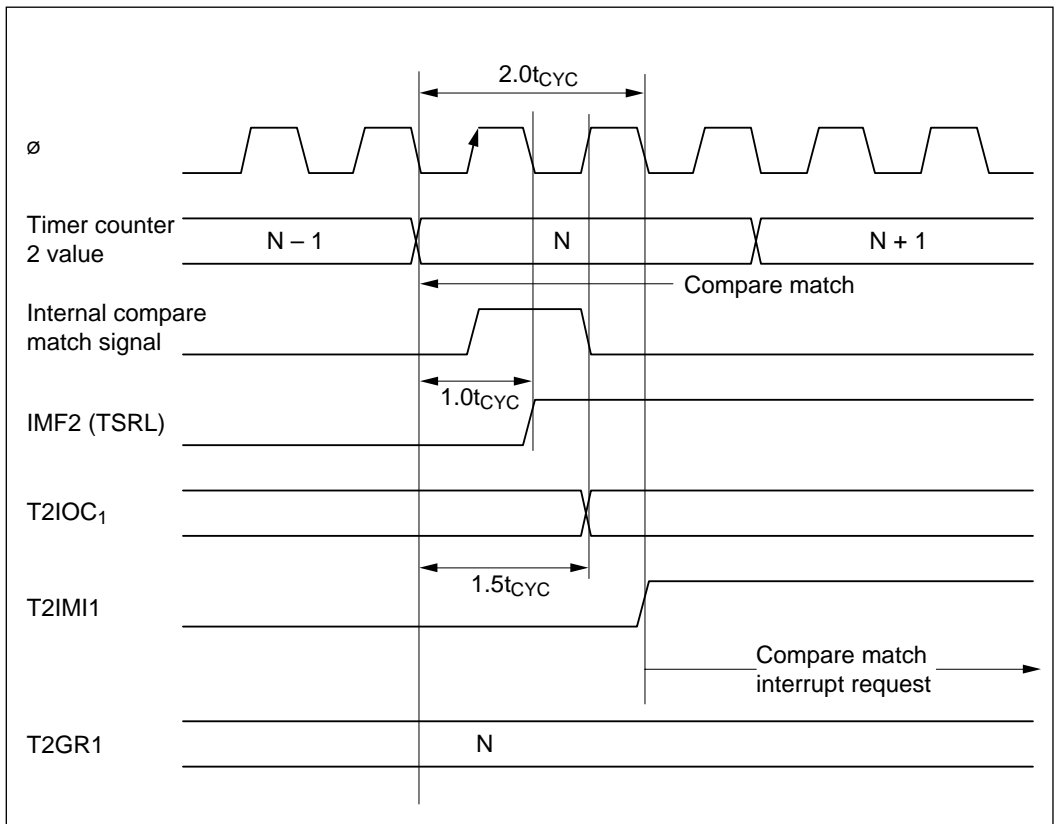
**Figure 11-45 External Clock Input Timing in Phase Counting Mode**

## 11.9 Interrupts

The IPU can request three types of interrupts: compare match, input capture, and overflow. The timing of each type of interrupt request is described next.

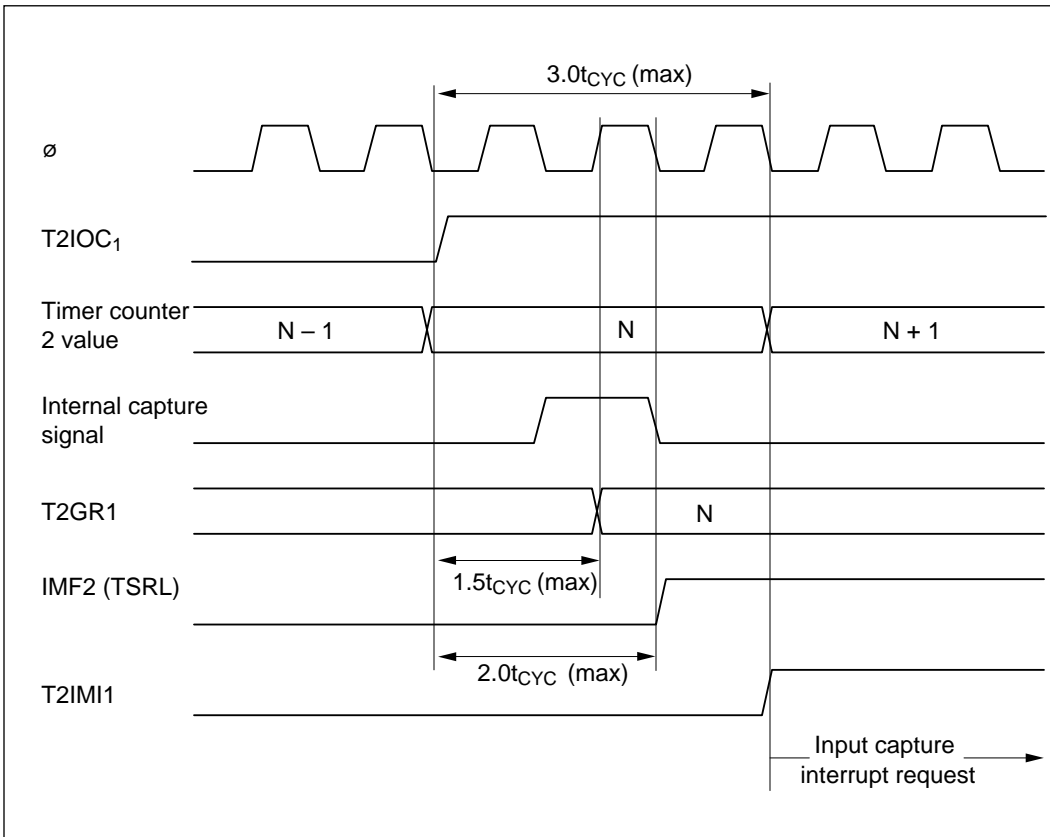
### 11.9.1 Interrupt Timing

**(1) Output Compare Timing:** Figure 11-46 shows the timing from counter incrementation to generation of a compare match interrupt request. One system clock cycle ( $1.0t_{CYC}$ ) after timer counter 2 matches the T2GR1 value (N), the IPU sets the input capture/compare match flag (IMF). A compare match signal ( $T2IOC_1$ ) is output  $0.5t_{CYC}$  after IMF is set. The interrupt request ( $T2IMI1$ ) is generated  $0.5t_{CYC}$  after the  $T2IOC_1$  output. The  $T2IMI1$  interrupt request therefore comes  $2.0t_{CYC}$  after the counter is incremented to N.



**Figure 11-46** Timing from Incrementation to Compare Match Interrupt Request

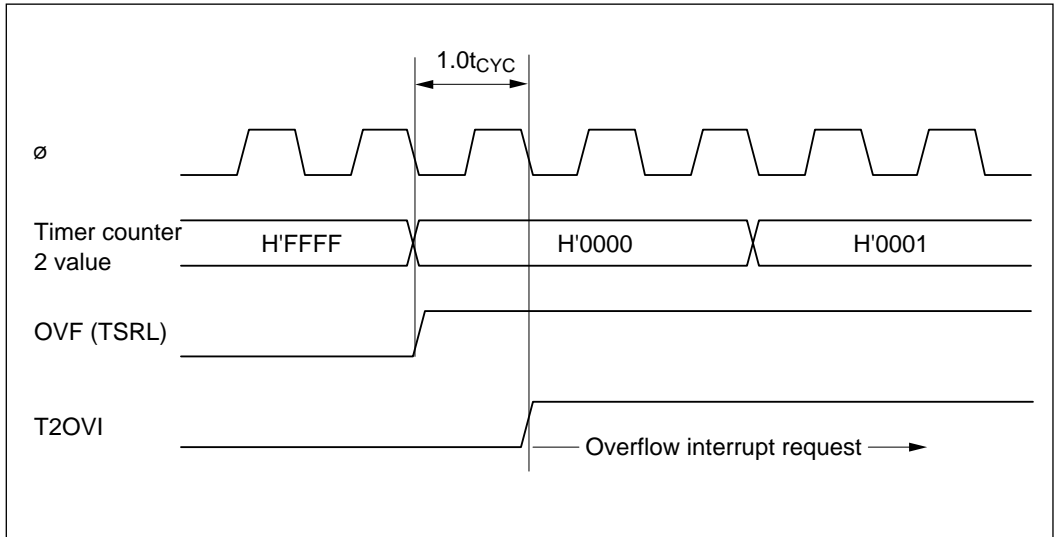
**(2) Input Capture Timing:** Figure 11-47 shows the timing from capture signal input to generation of an input capture interrupt request. A maximum  $1.5t_{CYC}$  after input of the capture signal, the IPU transfers the timer counter value (N) to T2GR1. The input capture/compare match flag (IMF) is set  $0.5t_{CYC}$  after the input capture transfer. The interrupt request (T2IMI1) is generated  $1.0t_{CYC}$  after IMF is set. The T2IMI1 interrupt request therefore comes a maximum  $3.0t_{CYC}$  after the counter value becomes N.



**Figure 11-47 Timing from Capture Input to Input Capture Interrupt Request**

**(3) Overflow Timing:** Figure 11-48 shows the timing from counter incrementation to generation of an overflow interrupt request. When the value of timer counter 2 returns from H'FFFF to H'0000 the IPU sets the overflow flag (OVF). The interrupt request (T2OVI) is generated  $1.0t_{CYC}$  after OVF is set.

In phase counting mode, the IPU sets the overflow flag (OVF) when the timer counter value returns from H'0000 to H'FFFF. For usage in phase counting mode, see section 11.8.9 “Phase Counting Mode.”



**Figure 11-48 Timing from Counter Incrementation to Overflow Interrupt Request**

### 11.9.2 Interrupt Sources and DTC Interrupts

The IPU has 35 interrupt sources. Of these, the compare match interrupt sources and the compare match/input capture interrupt sources can start the data transfer controller (DTC) to transfer data. Table 11-10 lists the interrupt sources and indicates which can start the DTC.

The exclusive compare match interrupt sources (such as TICMI<sub>1</sub> and TICMI<sub>2</sub>) are paired. Both sources in each pair share the same vector. Data transfer should not be enabled for both interrupt sources at the same time.

**Table 11-10 Interrupt Sources and DTC Interrupts**

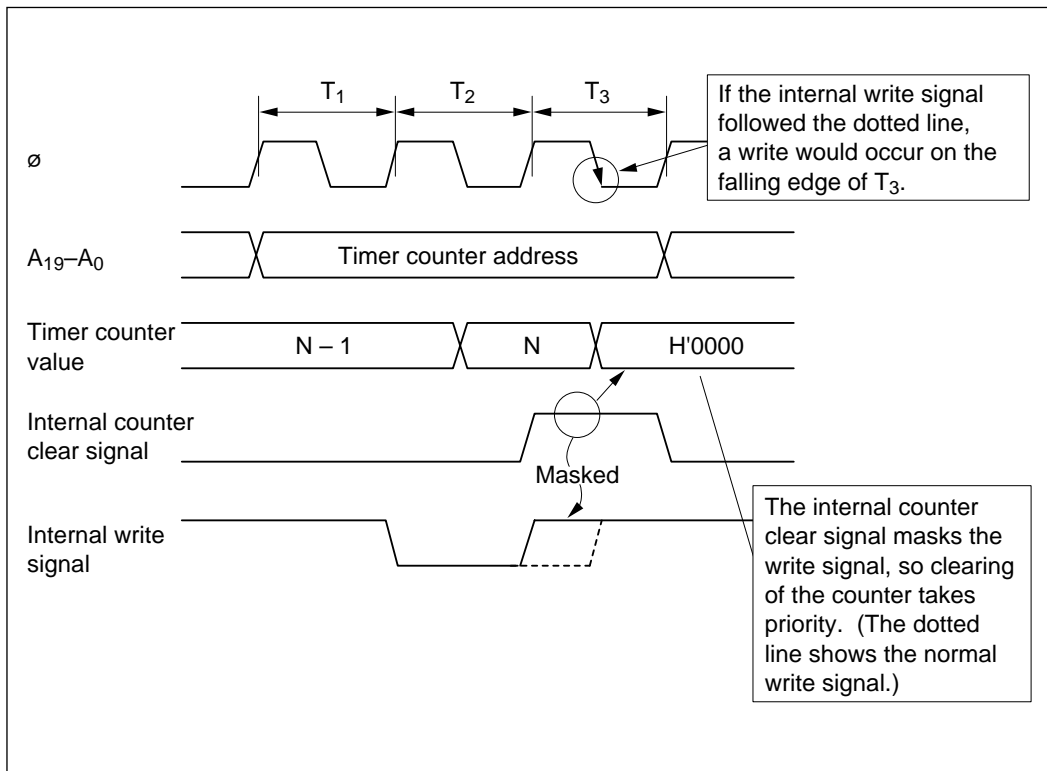
Channel	Interrupt Source	Description	DTC Activation Possible	Priority Order		
1	T1IMI <sub>1</sub>	GR1 compare match or input capture	Yes	High ↑		
	T1IMI <sub>2</sub>	GR2 compare match or input capture	Yes			
	T1CMI <sub>1</sub> / T1CMI <sub>2</sub>	DR1 or DR2 compare match	Yes			
	T1OVI	Timer counter 1 overflow	No			
	T1IMI <sub>3</sub>	GR3 compare match or input capture	Yes			
	T1IMI <sub>4</sub>	GR4 compare match or input capture	Yes			
	T1CMI <sub>3</sub> / T1CMI <sub>4</sub>	DR3 or DR4 compare match	Yes			
2	T2IMI <sub>1</sub>	GR1 compare match or input capture	Yes	↑		
	T2IMI <sub>2</sub>	GR2 compare match or input capture	Yes			
	T2CMI <sub>1</sub> / T2CMI <sub>2</sub>	DR1 or DR2 compare match	Yes			
	T2OVI	Timer counter 2 overflow	No			
3	T3IMI <sub>1</sub>	GR1 compare match or input capture	Yes		↑	
	T3IMI <sub>2</sub>	GR2 compare match or input capture	Yes			
	T3CMI <sub>1</sub> / T3CMI <sub>2</sub>	DR1 or DR2 compare match	Yes			
	T3OVI	Timer counter 3 overflow	No			
4	T4IMI <sub>1</sub>	GR1 compare match or input capture	Yes			↑
	T4IMI <sub>2</sub>	GR2 compare match or input capture	Yes			
	T4CMI <sub>1</sub> / T4CMI <sub>2</sub>	DR1 or DR2 compare match	Yes			
	T4OVI	Timer counter 4 overflow	No			
5	T5IMI <sub>1</sub>	GR1 compare match or input capture	Yes	↑		
	T5IMI <sub>2</sub>	GR2 compare match or input capture	Yes			
	T5CMI <sub>1</sub> / T5CMI <sub>2</sub>	DR1 or DR2 compare match	Yes			
	T5OVI	Timer counter 5 overflow	No			
6	T6IMI <sub>1</sub>	GR1 compare match or input capture	Yes		↑	
	T6IMI <sub>2</sub>	GR2 compare match or input capture	Yes			
	T6OVI	Timer counter 6 overflow	No			
7	T7IMI <sub>1</sub>	GR1 compare match or input capture	Yes			↑
	T7IMI <sub>2</sub>	GR2 compare match or input capture	Yes			
	T7OVI	Timer counter 7 overflow	No			
				Low ↓		

## 11.10 Notes and Precautions

This section describes contention between the compare registers and various IPU operations, and other matters requiring special attention.

### (1) Contention between Counter Read/Write by the H8/500 CPU and IPU Operations

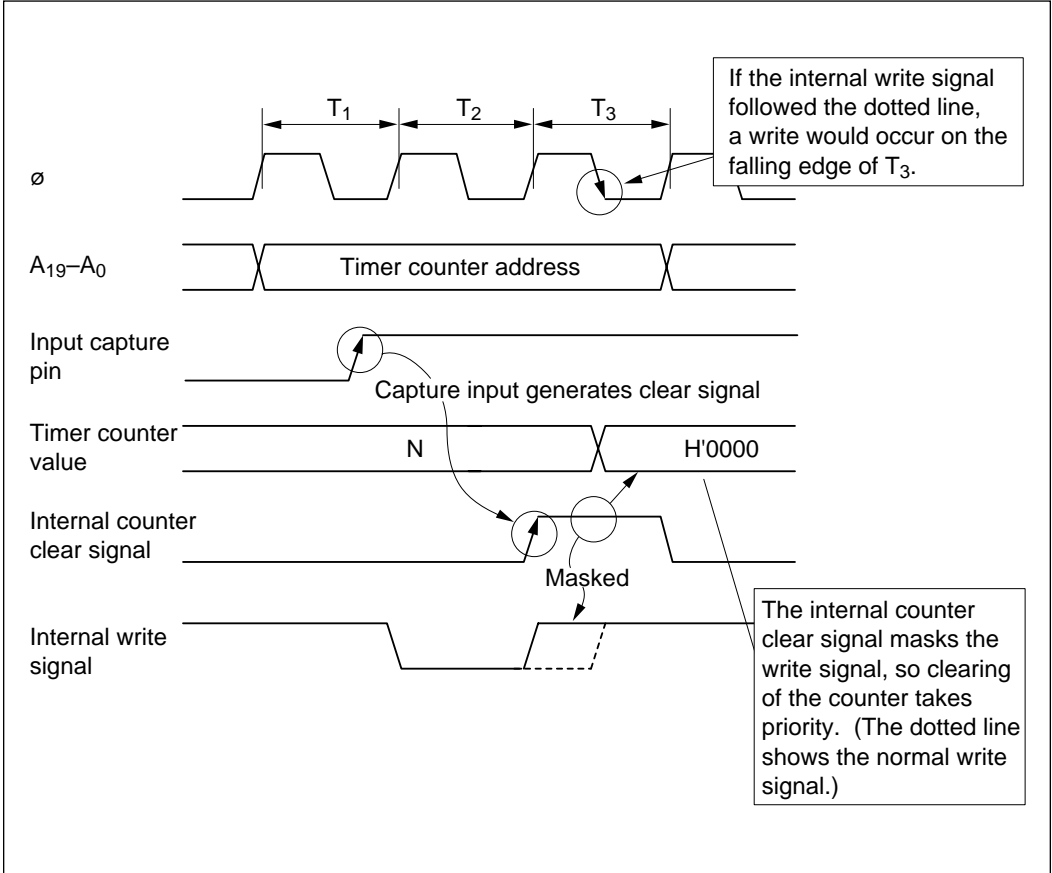
**Contention between Writing to Timer Counter by H8/500 CPU ( $T_3$ ) and Clearing by Compare Match:** Clearing the counter has priority.



**Figure 11-49 Contention between Writing to Timer Counter by H8/500 CPU ( $T_3$ ) and Clearing by Compare Match**

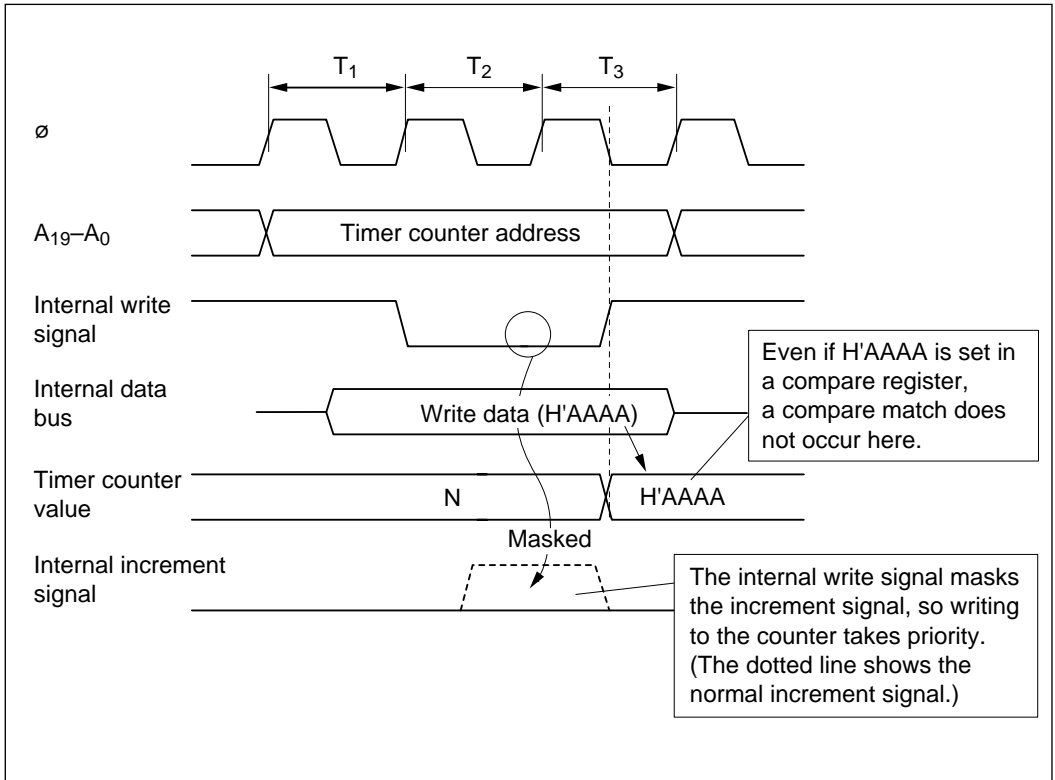


**Contention between Writing to Timer Counter by H8/500 CPU ( $T_3$ ) and Clearing by Capture Input:** Clearing the counter has priority.



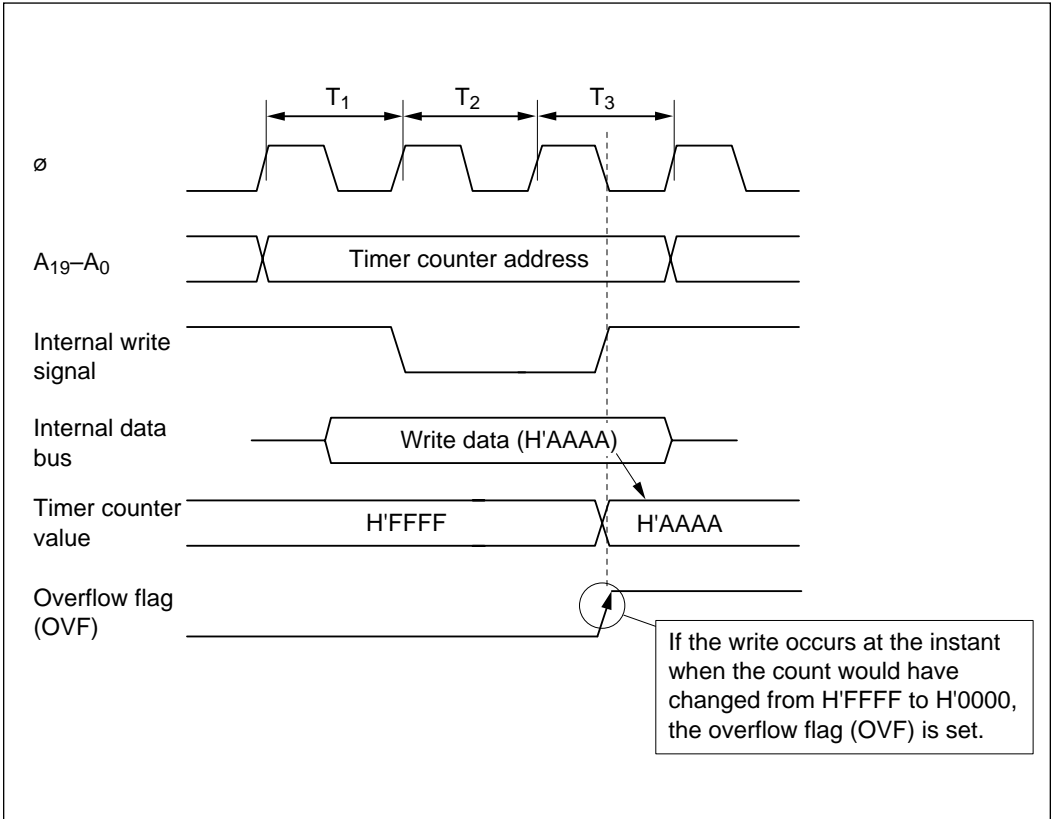
**Figure 11-50 Contention between Writing to Timer Counter by H8/500 CPU ( $T_3$ ) and Clearing by Capture Input**

**Contention between Timer Counter Write ( $T_3$ ) and Increment: Writing has priority.**



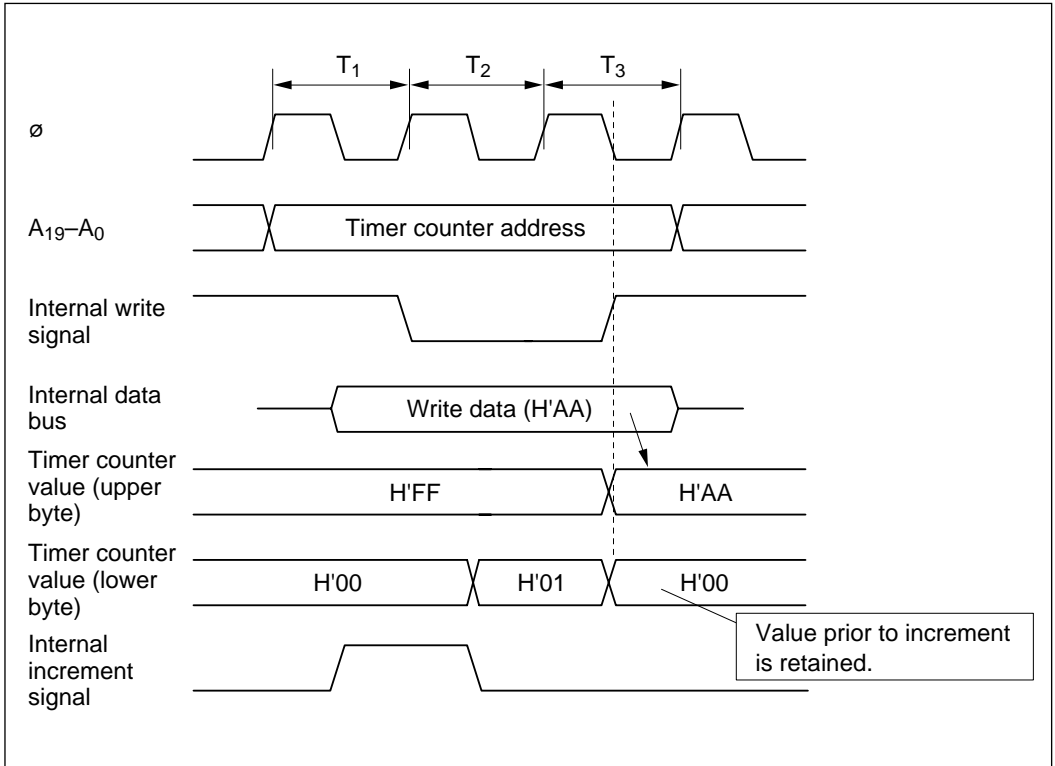
**Figure 11-51 Contention between Timer Counter Write ( $T_3$ ) by H8/500 CPU and Increment**

**Contention between Timer Counter Write ( $T_3$ ) and Setting of Overflow Flag:** Setting the overflow flag has priority.



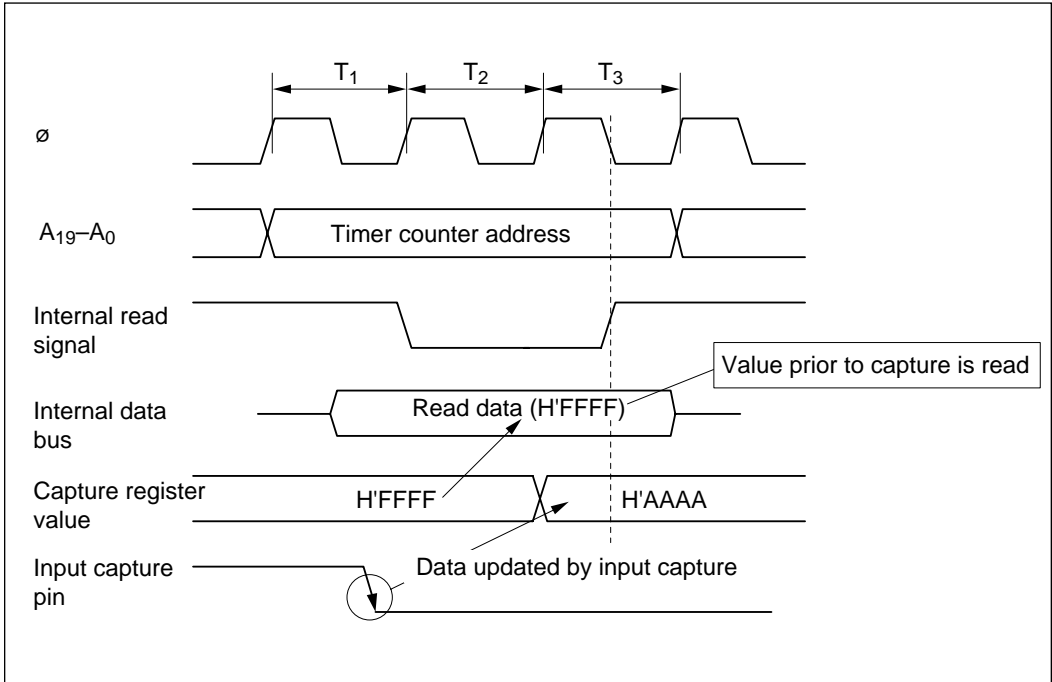
**Figure 11-52 Contention between Timer Counter Write ( $T_3$ ) by H8/500 CPU and Setting of Overflow Flag**

**Contention between Timer Counter Byte Write (T<sub>2</sub>) and Increment:** If the write is to the upper byte, the new value is written in the upper byte and the lower byte retains its old value. If the write is to the lower byte, the new value is written in the lower byte and the upper byte retains its old value. If the contention occurs at T<sub>3</sub>, however, the byte that is not written is incremented.



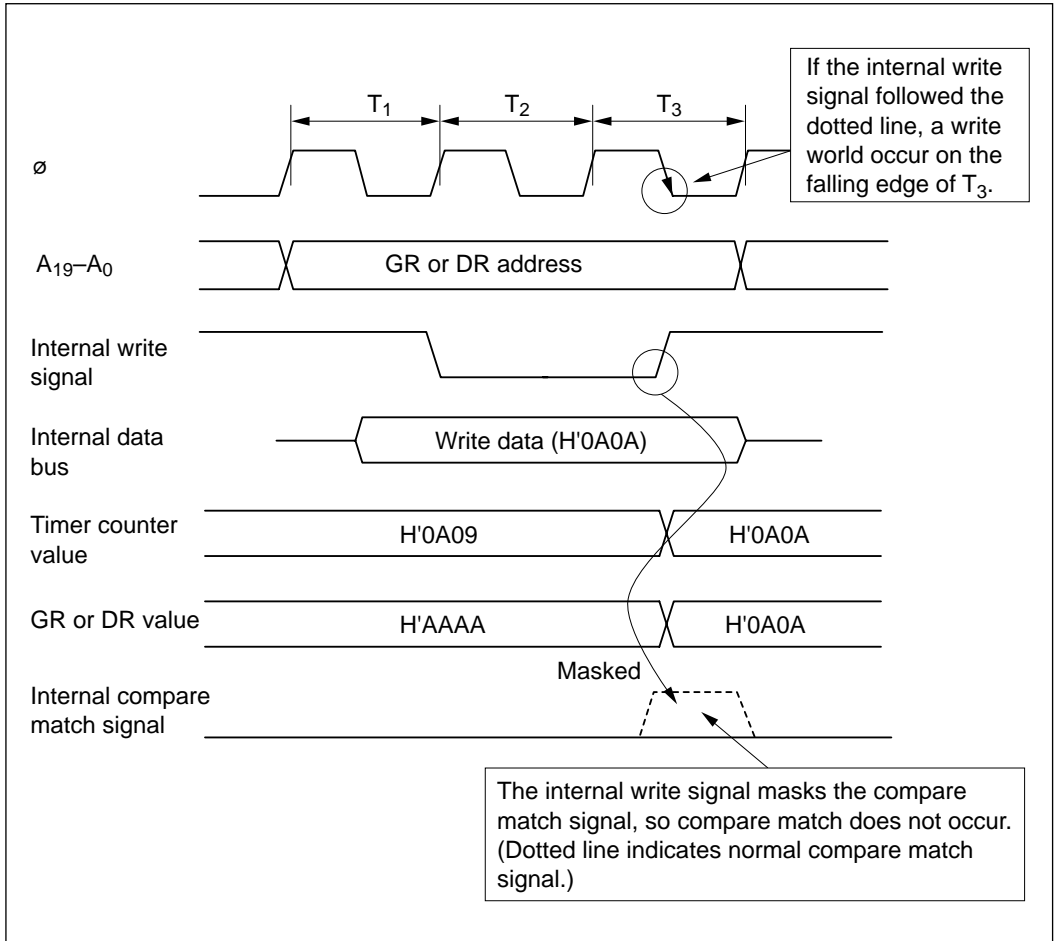
**Figure 11-53 Contention between Timer Counter Byte Write (T<sub>2</sub>) by H8/500 CPU and Increment**

**Contention between Capture Register Read ( $T_3$ ) and Input Capture:** The H8/500 CPU reads the data prior to capture.



**Figure 11-54 Contention between Capture Register Read ( $T_3$ ) by H8/500 CPU and Input Capture**

**Contention between Writing to General Register or Dedicated Register by H8/500 CPU (T<sub>3</sub>) and Compare Match:** Compare match does not occur.



**Figure 11-55 Contention between Writing to General Register or Dedicated Register by H8/500 CPU (T<sub>3</sub>) and Compare Match**

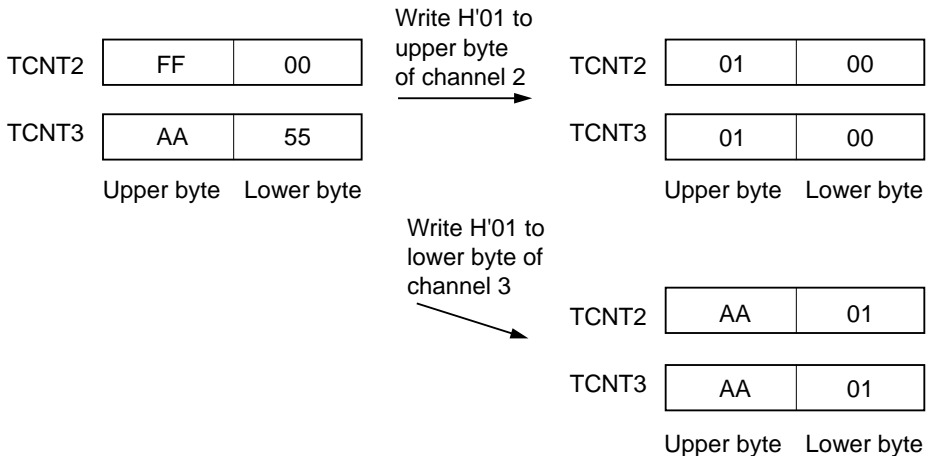
**(2) Note on Writing in Synchronizing Mode:** After a write in synchronizing mode, all 16 bits of all specified counters have the same value as the counter that was written. This is true regardless of the operand size (word or byte).

**Example:** When channels 2 and 3 are synchronized

- Word write to channel 2 or word write to channel 3



- Byte write to channel 2 or byte write to channel 3



**(3) Note on Compare Register Setting:** The compare match frequency differs depending on whether the timer counter clock source is the system clock ( $\phi$ ) or another source.

When the counter increments on the system clock as in figure 11-56, the compare match frequency is:

$$T = \phi / (N + 1)$$

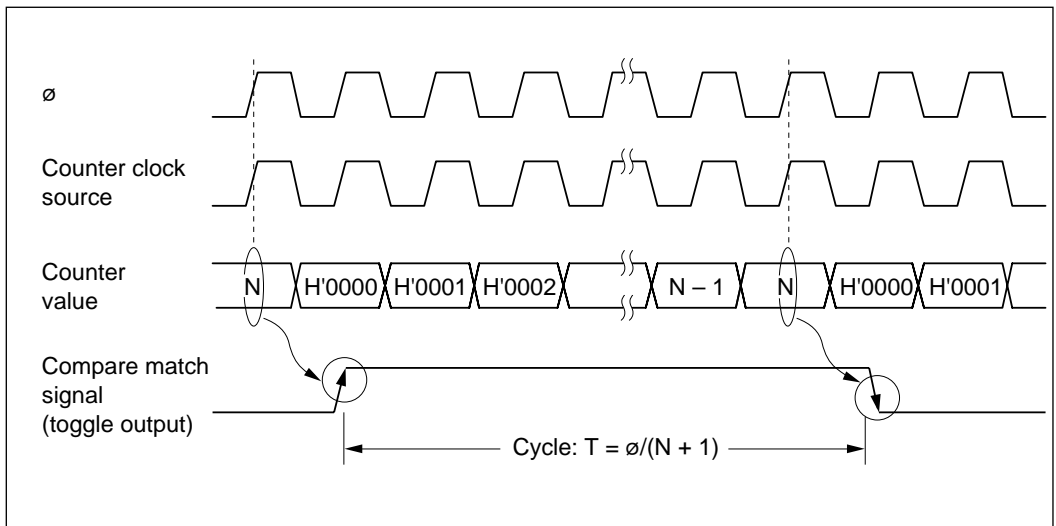
(T: compare match frequency.  $\phi$ : system clock frequency. N: value set in compare register.)

When the counter increments on a clock source other than the system clock as in figure 11-57, the compare match frequency is:

$$T = \phi / (D * N) \quad * \text{ Example: If the counter clock source is } \phi/2, \text{ then } D = 2.$$

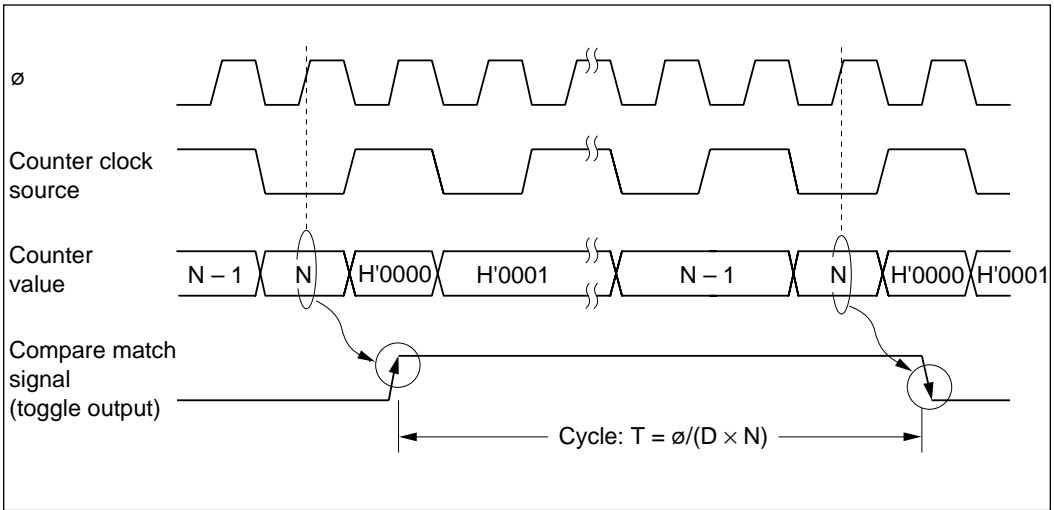
(T: compare match frequency.  $\phi$ : system clock frequency. D: frequency ratio of system clock to counter clock source. N: value set in compare register.)

In this case, if H'0000 is set in the compare register, compare match does not occur.



**Figure 11-56 Compare Match Frequency when Clock Source is System Clock**





**Figure 11-57 Compare Match Frequency when Clock Source is not System Clock**

**Note on Rewriting the Compare Match Register:** To generate a compare match after rewriting the register, the following condition must be satisfied. Note that even if the counter value when rewriting the register and the register value after rewriting the register do match, a compare match will not be generated.

1. Slowing down compare match timing

$$\text{Reg} \leq \text{Count} < \text{Reg}' \dots\dots\dots (1)$$

However, if  $\text{Reg} \approx \text{TCNT}$ , the following condition must be met:

$$\text{Count} < \text{Reg}' \dots\dots\dots (1')$$

Where   Reg:   Register value before rewriting  
           Count: Register value during rewriting  
           Reg':   Register value after rewriting  
            $t_{\text{cyc}}$ : Counter refresh cycle or overflow cycle

2. Speeding up compare match timing

$$\text{Reg} \leq \text{Count} \leq t_{\text{cyc}} \dots\dots\dots (2)$$

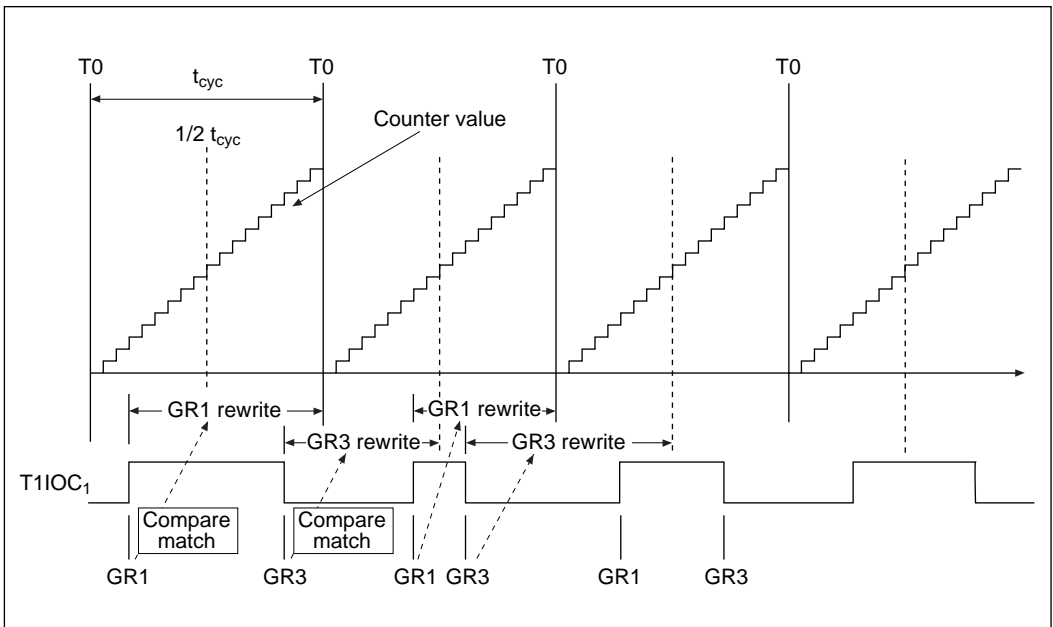
Where   Reg:   Register value before rewriting  
           Count: Register value during rewriting  
            $t_{\text{cyc}}$ : Counter refresh cycle or overflow cycle

**Rewriting the Compare Match Register in PWM Mode:** In PWM mode, to shorten the pulse width, two register values must be rewritten within the same cycle. Restrictions regarding writing to the register are as described previously. Refer to figure 11-58 for a timing diagram of actual rewrite processing (updating).

- Example: PWM pulse output on channel 1
  - Pulse set: GR1
  - Pulse reset: GR3
- Setting range
  - GR1: Between 0 and  $1/2 t_{cyc}$ . (Between 1 and  $1/2 t_{cyc}$  when  $\emptyset$  is selected as the clock source.)
  - GR3: Between  $1/2 t_{cyc}$  and  $t_{cyc}$ .

Here,  $t_{cyc}$  refers to one cycle of the counter.

- Rewriting register to shorten pulse width
  - GR1 rewrite: At GR1, or while  $1/2 t_{cyc} < \text{count} \leq t_{cyc}$ .
  - GR3 rewrite: At GR3 or while  $0 < \text{count} \leq t_{cyc}$  ( $1 < \text{count} < 1/2 t_{cyc}$  if  $\emptyset$  is the clock source).



**Figure 11-58 Example of Register Rewrite Timing in PWM Mode**

# Section 12 PWM Timers

## 12.1 Overview

The H8/539F has a built-in pulse-width modulation (PWM) timer module with three independent channels. Each PWM timer has an eight-bit timer counter (TCNT) and an eight-bit duty register (DTR). DTR settings can provide pulse output with any duty cycle from 0% to 100%.

### 12.1.1 Features

The PWM timer features are:

- Selection of eight counter clock sources
- Selection of duty cycles from 0% to 100% with 1/250 resolution
- Selection of direct or inverted PWM output

## 12.1.2 Block Diagram

Figure 12-1 shows a block diagram of one PWM timer.

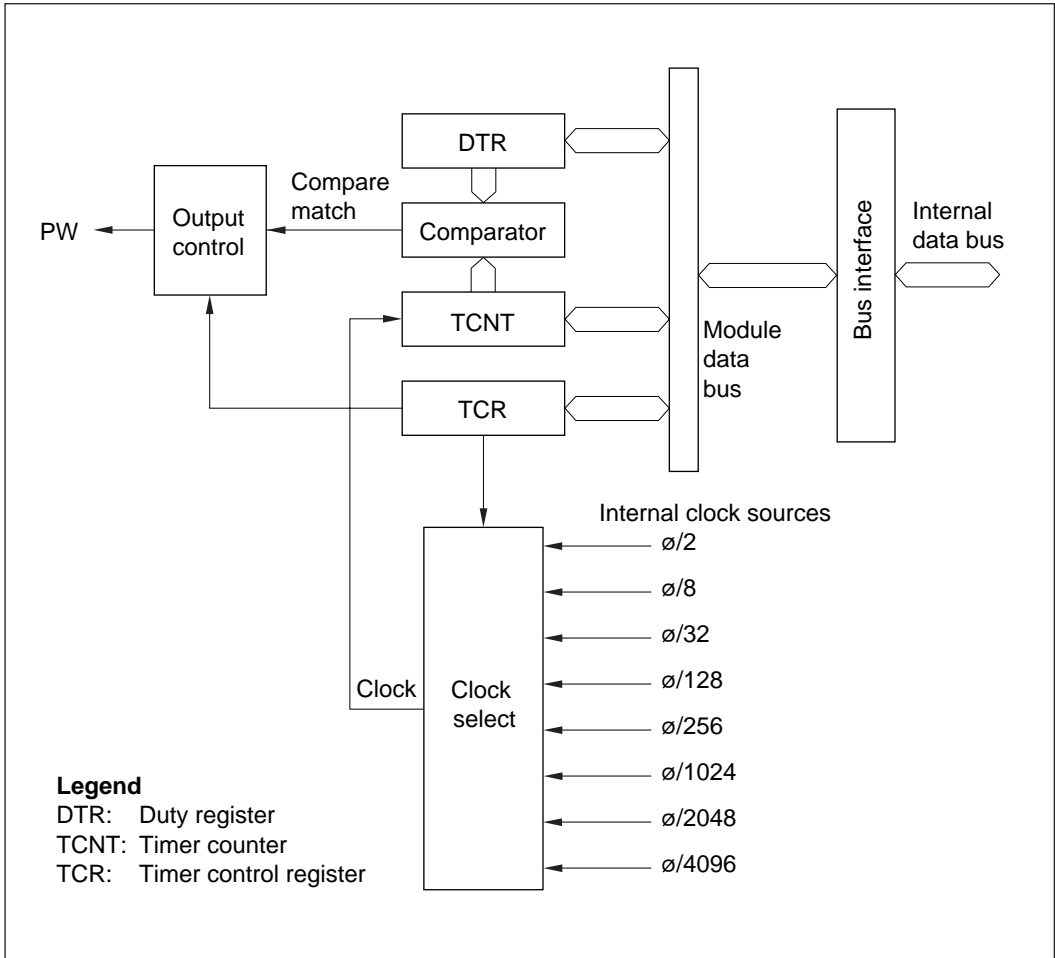


Figure 12-1 Block Diagram of PWM Timer

### 12.1.3 Pin Configuration

Table 12-1 summarizes the PWM timer output pins.

**Table 12-1 PWM Timer Pins**

Name	Abbreviation	I/O	Function
PWM1 output pin	PW <sub>1</sub>	Output	PWM timer 1 pulse output
PWM2 output pin	PW <sub>2</sub>	Output	PWM timer 2 pulse output
PWM3 output pin	PW <sub>3</sub>	Output	PWM timer 3 pulse output

### 12.1.4 Register Configuration

Table 12-2 summarizes the internal registers of the PWM timers.

**Table 12-2 PWM Timer Registers**

Channel	Name	Abbreviation	R/W	Initial Value	Address
1	Timer control register	TCR	R/W	H'38	H'FEF0
	Duty register	DTR	R/W	H'FF	H'FEF1
	Timer counter	TCNT	R/(W)*	H'00	H'FEF2
2	Timer control register	TCR	R/W	H'38	H'FEF4
	Duty register	DTR	R/W	H'FF	H'FEF5
	Timer counter	TCNT	R/(W)*	H'00	H'FEF6
3	Timer control register	TCR	R/W	H'38	H'FEF8
	Duty register	DTR	R/W	H'FF	H'FEF9
	Timer counter	TCNT	R/(W)*	H'00	H'FEFA

Note: \* Can be written and read, but the write function is for test purposes only. Do not write to these registers during normal operation.

## 12.2 Register Descriptions

### 12.2.1 Timer Counter (TCNT)

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
R/W	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)

The timer counter (TCNT) is an eight-bit up-counter. When the output enable bit (OE) is set to 1 in TCR, TCNT starts counting pulses of the internal clock selected by clock select bits 2 to 0 (CKS2 to CKS0). After counting from H'00 to H'F9, the count repeats from H'00.

TCNT can be written to and read, but the write function is for test purposes only. Do not write to TCNT during normal operation, because this may have unpredictable effects.

TCNT is initialized to H'00 by a reset and in standby mode, and when the OE bit is cleared to 0.

### 12.2.2 Duty Register (DTR)

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The duty register (DTR) specifies the duty cycle of the output pulse. Any duty cycle from 0% to 100% can be output by setting the corresponding value in DTR. The resolution is 1/250. Writing 0 (H'00) in DTR gives a 0% duty cycle. Writing 125 (H'7D) gives a 50% duty cycle. Writing 250 (H'FA) gives a 100% duty cycle.

The DTR and TCNT values are constantly compared. When the values match, the PWM output is placed in the 0 state. When the TCNT value changes from H'00 to H'01, the PWM output is placed in the 1 state, unless the DTR value is H'00, in which case the duty cycle is 0% and the PWM output remains in the 0 state.

DTR is double-buffered. A new value written in DTR does not become valid until after the timer count changes from H'F9 to H'00. While the OE bit is cleared to 0 in TCR, however, new values written in DTR become valid immediately. When DTR is read, the value read is the currently valid value.

DTR is initialized to H'FF by a reset and in standby mode.

### 12.2.3 Timer Control Register (TCR)

Bit	7	6	5	4	3	2	1	0
	OE	OS	—	—	—	CKS2	CKS1	CKS0
Initial value	0	0	1	1	1	0	0	0
R/W	R/W	R/W	—	—	—	R/W	R/W	R/W

The timer control register (TCR) is an eight-bit readable/writable register that selects the clock input to TCNT and controls PWM output.

TCR is initialized to H'38 by a reset and in standby mode.

**Bit 7—Output Enable (OE):** Starts or stops TCNT and controls PWM output.

#### Bit 7

OE	Description
0	PWM output is disabled and the TCNT value is cleared to 0 (Initial value)
1	PWM output is enabled and TCNT is counting

**Bit 6—Output Select (OS):** Selects direct or inverted PWM output.

#### Bit 6

OS	Description
0	Direct PWM output (Initial value)
1	Inverted PWM output

**Bits 5 to 3—Reserved:** Read-only bits, always read as 1.



**Bits 2 to 0—Clock Select (CKS2 to CKS0):** These bits select one of eight internal clock sources, obtained by dividing the system clock ( $\phi$ ), for input to TCNT.

Bit 2	Bit 1	Bit 0	Description
CKS2	CKS1	CKS0	
0	0	0	$\phi/2$ (Initial value)
0	0	1	$\phi/8$
0	1	0	$\phi/32$
0	1	1	$\phi/128$
1	0	0	$\phi/256$
1	0	1	$\phi/1024$
1	1	0	$\phi/2048$
1	1	1	$\phi/4096$

The PWM resolution, period, and frequency can be calculated as follows from the frequency of the selected internal clock source.

$$\text{Resolution} = 1/(\text{internal clock frequency})$$

$$\text{PWM period} = \text{resolution} \times 250$$

$$\text{PWM frequency} = 1/(\text{PWM period})$$

Table 12-3 lists the resolution, PWM period, and PWM frequency for each clock source when the system clock frequency ( $\phi$ ) is 10 MHz.

**Table 12-3 PWM Period and Resolution**

Internal Clock Frequency	Resolution	PWM Period	PWM Frequency
$\phi/2$	200 ns	50 $\mu$ s	20 kHz
$\phi/8$	800 ns	200 $\mu$ s	5 kHz
$\phi/32$	3.2 $\mu$ s	800 $\mu$ s	1.25 kHz
$\phi/128$	12.8 $\mu$ s	3.2 ms	312.5 Hz
$\phi/256$	25.6 $\mu$ s	6.4 ms	156.3 Hz
$\phi/1024$	102.4 $\mu$ s	25.6 ms	39.1 Hz
$\phi/2048$	204.8 $\mu$ s	51.2 ms	19.5 Hz
$\phi/4096$	409.6 $\mu$ s	102.4 ms	9.8 Hz

## 12.3 PWM Timer Operation

PWM timer operation is described below. Figure 12-2 shows a timing diagram.

### (1) Direct Output (OS = 0)

- When OE = 0 [(a) in figure 12-2]

The timer count is held at H'00 and PWM output is disabled. The pin state depends on the port data register (DR) and data direction register (DDR) settings. A value (N) written in DTR becomes valid immediately.

- When OE is set to 1

— TCNT begins counting up, and the PWM output goes to the 1 state. [(b) in figure 12-2]

— When the count reaches the DTR value, the PWM output goes to the 0 state. [(c) in figure 12-2]

— If the DTR value is changed (by writing M), the new value becomes valid after TCNT changes from H'F9 to H'00. [(d) in figure 12-2]

### (2) Inverted Output (OS = 1): The PWM output is inverted. [(e) in figure 12-2]

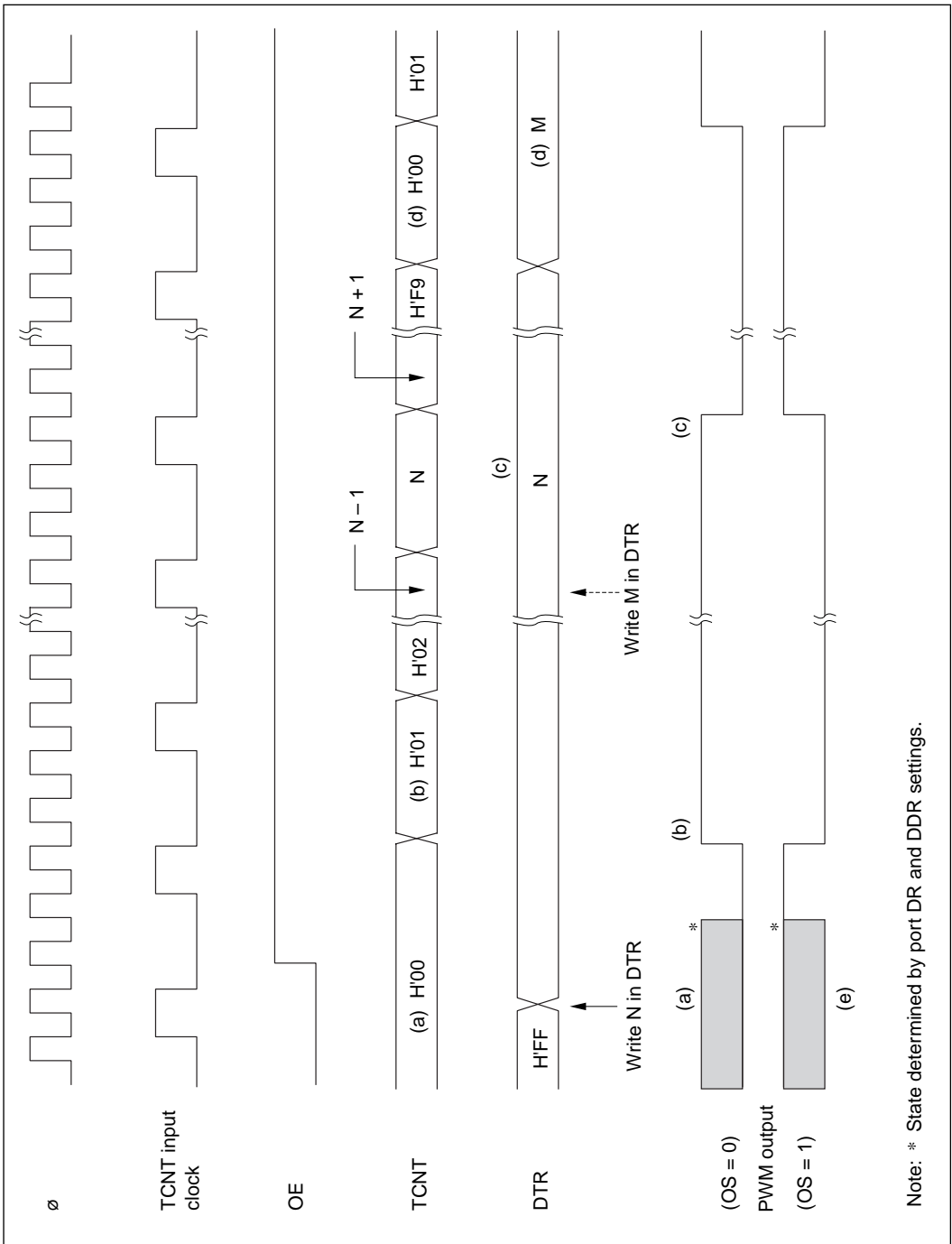


Figure 12-2 PWM Operation Timing

Note: \* State determined by port DR and DDR settings.

## 12.4 Usage Notes

When using the PWM timers, note the following points.

To use port 6, 7, or A for PWM output, first set the appropriate bit (PWM1E, PWM2E, or PWM3E) to 1 in P67CR or PACR. Each of these bits can be set independently.

The H8/538 has no PWM timers.

- Any necessary changes to bits CKS2 to CKS0 and OS should be made before the OE bit is set to 1.
- If the DTR value is H'00, the duty cycle is 0% (always 0). If the DTR value is H'FA to H'FF, the duty cycle is 100% (always 1).

For inverted output, these output levels are inverted.

# Section 13 Watchdog Timer

## 13.1 Overview

System operation can be monitored by the on-chip watchdog timer (WDT, one channel). The WDT can generate a reset signal for the entire chip if a system crash allows the timer counter (TCNT) to overflow. When this watchdog function is not needed, the WDT can be used as an interval timer. In interval timer operation, an IRQ0 interrupt is requested at each counter overflow. The WDT is also used in recovering from software standby mode.

### 13.1.1 Features

WDT features are listed below.

- Selection of eight counter clock sources
- Interval timer option
- Timer counter overflow generates a reset signal or interrupt

The reset signal is generated in watchdog timer operation. An IRQ0 interrupt is requested in interval timer operation.

- Overflow reset signal resets the entire chip internally, and can also be output externally

The reset signal generated by timer counter overflow during watchdog timer operation resets the entire chip internally. If enabled by the reset output enable bit, an external reset signal can be output to reset other system devices simultaneously, in the H8/539F (dual power source model).

The H8/539F S-mask model (single power source) does not have a reset external output function.

### 13.1.2 Block Diagram

Figure 13-1 shows a block diagram of the WDT.

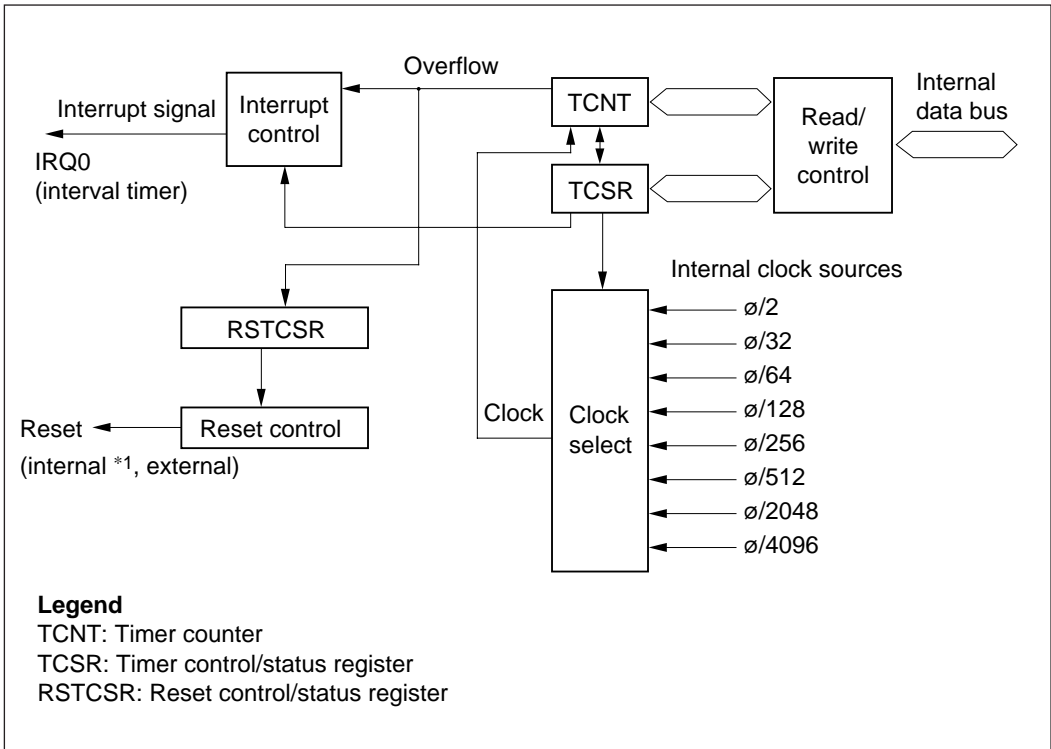


Figure 13-1 WDT Block Diagram

### 13.1.3 Register Configuration

Table 13-1 summarizes the WDT registers.

Table 13-1 WDT Registers

Address		Name	Abbreviation	R/W	Initial Value
Write	Read				
H'FF10	H'FF10	Timer control/status register	TCSR	R/(W)*2	H'18
	H'FF11	Timer counter	TCNT	R/W	H'00
H'FF1F		Reset control/status register	RSTCSR	R/(W)*2	H'3F

Note: 1. The H8/539F dual power source model has an external reset output function, but the H8/539F S-mask model (single power source) does not.

2. Software can write 0 in bit 7 to clear the flag, but cannot write 1.

## 13.2 Register Descriptions

The watchdog timer has three registers, which are described next.

### 13.2.1 Timer Counter

The timer counter (TCNT) is an eight-bit readable and writable\* up-counter. The TCNT bit structure is shown next.

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When the timer enable bit (TME) in the timer control/status register (TCSR) is set to 1, the timer counter starts counting pulses of an internal clock source selected by clock select bits 2 to 0 (CKS2 to CKS0) in TCSR. When the count overflows (changes from H'FF to H'00), an overflow flag (OVF) in TCSR is set to 1. The timer count is initialized to H'00 by a reset and when the TME bit is cleared to 0.

Note: \* TCNT is write-protected by a password. See section 13.2.4, “Notes on Register Access” for details.

### 13.2.2 Timer Control/Status Register

The timer control/status register (TCSR) is an eight-bit readable and partly writable\*1 register. Its functions include selecting the timer mode and clock source. The TCSR bit structure is shown next.

Bit	7	6	5	4	3	2	1	0
	OVF	WT/IT	TME	—	—	CKS2	CKS1	CKS0
Initial value	0	0	0	1	1	0	0	0
R/W	R/(W)*2	R/W	R/W	—	—	R/W	R/W	R/W

Clock select bits  
 These bits select the TCNT clock source

Reserved bits

Timer enable bit  
 Enables and disables the timer

Timer mode select bit  
 Selects the mode

Overflow flag  
 Status flag indicating overflow

Bits 7 to 5 are initialized to 0 by a reset, in hardware standby mode, and in software standby mode. Bits 2 to 0 are initialized to 0 by a reset and in hardware standby mode, but retain their values in software standby mode.

- Notes: 1. TCSR is write-protected by a password. See section 13.2.4 “Notes on Register Access” for details.
2. Software can write 0 in bit 7 to clear the flag, but cannot write 1.

**(1) Bit 7—Overflow Flag (OVF):** This status flag indicates that the timer counter has overflowed from H'FF to H'00 in interval timer mode. When OVF = 1, an IRQ0 interrupt is requested.

#### Bit 7

OVF	Description
0	Cleared by reading OVF after it has been set to 1, then writing 0 in OVF (Initial value)
1	Set when TCNT overflows



**(2) Bit 6—Timer Mode Select (WT/IT̄):** Selects whether to use the WDT as a watchdog timer or interval timer. If used as an interval timer (WT/IT̄ = 0), the WDT generates an IRQ0 interrupt request when the timer counter (TCNT) overflows. If used as a watchdog timer (WT/IT̄ = 1), the WDT generates a reset when the timer counter (TCNT) overflows.

**Bit 6**

WT/IT̄	Description
0	Interval timer: IRQ0 interrupt request (Initial value)
1	Watchdog timer: reset request

**(3) Bit 5—Timer Enable (TME):** Enables or disables the timer counter (TCNT). Always clear TME to 0 before entering software standby mode.

**Bit 5**

TME	Description
0	Timer disabled: TCNT is initialized to H'00 and stopped. (Initial value)
1	Timer enabled: TCNT starts counting.

**(4) Bits 4 and 3—Reserved:** Read-only bits, always read as 1.

**(5) Bits 2 to 0—Clock Select 2 to 0 (CKS2/1/0):** These bits select one of eight internal clock sources for input to TCNT. The clock signals are obtained by prescaling the system clock ( $\phi$ ). The overflow interval listed in the following table is the time from when TCNT begins counting from H'00 until an overflow occurs. When the WDT operates as an interval timer, IRQ0 interrupts are requested at this interval. Set CKS2 to CKS0 to the clock settling time before entering software standby mode.

Bit 2	Bit 1	Bit 0	Description	
CKS2	CKS1	CKS0	Clock Source	Overflow Interval ( $\phi = 10 \text{ MHz}$ )
0	0	0	$\phi/2$	51.2 $\mu\text{s}$ (Initial value)
0	0	1	$\phi/32$	819.2 $\mu\text{s}$
0	1	0	$\phi/64$	1.6 ms
0	1	1	$\phi/128$	3.3 ms
1	0	0	$\phi/256$	6.6 ms
1	0	1	$\phi/512$	13.1 ms
1	1	0	$\phi/2048$	52.4 ms
1	1	1	$\phi/4096$	104.9 ms

### 13.2.3 Reset Control/Status Register

The H8/539F S-mask model (single power source) does not have a  $\overline{\text{RESO}}$  pin or a reset external output function. RSTOE in the RSTCSR register can be set and read, but a reset external output operation is not performed.

The reset control/status register (RSTCSR) is an eight-bit readable and partly writable\*<sup>1</sup> register that indicates when a reset signal has been generated by WDT overflow, and controls external output of this reset signal.

Bit	7	6	5	4	3	2	1	0
	WRST	RSTOE	—	—	—	—	—	—
Initial value	0	0	1	1	1	1	1	1
R/W	R/(W)* <sup>2</sup>	R/W	—	—	—	—	—	—

Bits 7 and 6 are initialized by input of a reset signal at the  $\overline{\text{RES}}$  pin. They are not initialized by a reset signal generated by the WDT.

- Notes:
1. RSTCSR is write-protected by a password. See section 13.2.4, “Notes on Register Access” for details
  2. Only a 0 can be written in bit 7 to clear the flag, after first reading the RSTCSR register.

**(1) Bit 7—Watchdog Timer Reset (WRST):** Indicates that the watchdog timer counter has overflowed and generated a reset signal. This reset signal resets the entire chip. If the reset output enable bit (RSTOE) is set to 1, the reset signal is also output (low) at the  $\overline{\text{RESO}}$  pin to initialize external system devices.

#### Bit 7

WRST	Description
0	Cleared to 0 by reset signal input at $\overline{\text{RES}}$ pin, when 0 is written in WRST after reading WRST = 1 by software (Initial value)
1	Set by TCNT overflow when WDT is used as a watchdog timer, generating a reset signal

Note: If 0 is written in WRST without reading WRST = 1, the WRST bit will not be cleared.

**(2) Bit 6—Reset Output Enable (RSTOE):** Enables or disables external output at the  $\overline{\text{RESO}}$  pin\* of the reset signal generated if the timer counter (TCNT) overflows when the WDT is used as a watchdog timer.

**Bit 6**

<b>RSTOE</b>	<b>Description</b>	
0	Reset signal generated by TCNT overflow is not output externally	(Initial value)
1	Reset signal generated by TCNT overflow is output externally	

Note: \* H8/539F (Dual power source model):

The  $\overline{\text{RESO}}$  pin is an open-drain output pin. Regardless of whether reset output is used, the  $\overline{\text{RESO}}$  pin should be pulled up to Vcc externally. A sample circuit is shown in figure 19-24 in section 19.7, “Flash Memory Programming and Erasing Precautions”, and figure 19-28, section 19.8 “Notes on Mounting Board Development-Handling of Vpp and Mode MD2 Pins.” The  $\overline{\text{RESO}}$  output is multiplexed with Vpp input (the flash memory power supply), and therefore reset output off-chip must be disabled (by clearing RSTOE to 0) when 12 V is applied to the Vpp pin. For cautions concerning the  $\overline{\text{RESO}}$ /Vpp pin, see notes 5 and 6 in section 19.7, “Flash Memory Programming and Erasing Precautions.”

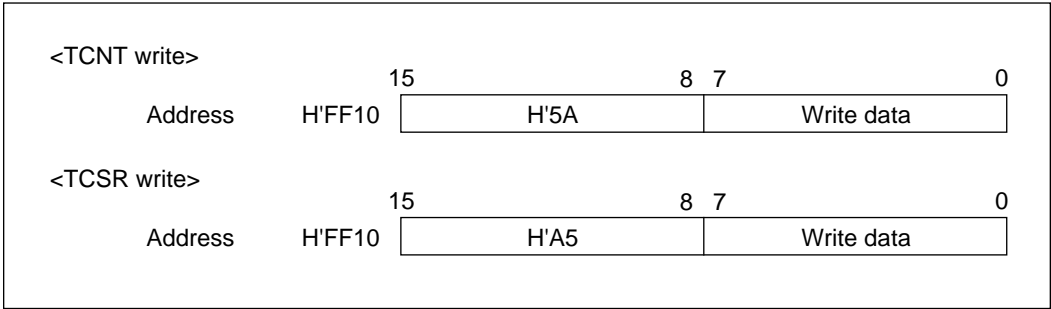
The H8/539F S mask model (single power source) does not have  $\overline{\text{RESO}}$  pin function. When set to RSTOE, it is readable but there is no external output reset operation.

**(3) Bits 5 to 0—Reserved:** Read-only bits, always read as 1.

**13.2.4 Notes on Register Access**

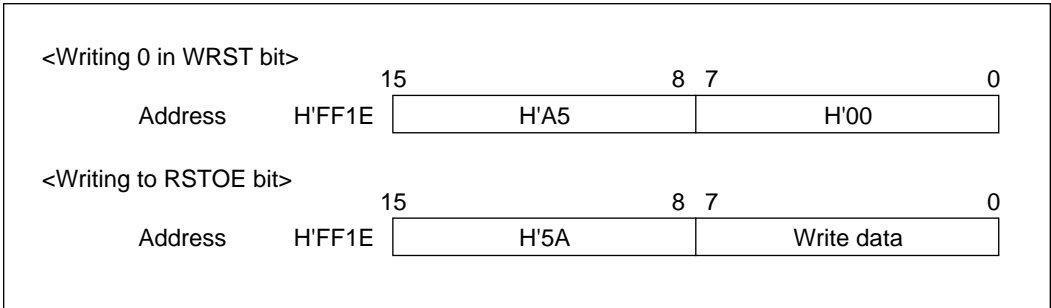
The watchdog timer’s TCNT, TCSR, and RSTCSR registers differ from other registers in being more difficult to write. The procedures for writing and reading these registers are given below.

**(1) Writing to TCNT and TCSR:** These registers must be written by word access. They cannot be written by byte instructions. Figure 13-2 shows the format of data written to TCNT and TCSR. TCNT and TCSR both have the same write address. The write data must be contained in the lower byte of the written word. The upper byte must contain H'5A (password for TCNT) or H'A5 (password for TCSR). This transfers the write data from the lower byte to TCNT or TCSR.



**Figure 13-2 Format of Data Written to TCNT and TCSR**

**(2) Writing to RSTCSR:** RSTCSR must be written by word access. It cannot be written by byte instructions. Figure 13-3 shows the format of data written to RSTCSR. To write 0 in the WRST bit, the write data must have H'A5 in the upper byte and H'00 in the lower byte. The H'00 in the lower byte clears the WRST bit in RSTCSR to 0. To write to the RSTOE bit, the upper byte must contain H'5A and the lower byte must contain the write data. Writing this word transfers a write data value into the RSTOE bit.



**Figure 13-3 Format of Data Written to RSTCSR**

**(3) Reading TCNT, TCSR, and RSTCSR:** These registers are read like other registers. Byte access instructions can be used. The read addresses are H'FF10 for TCSR, H'FF11 for TCNT, and H'FF1F for RSTCSR, as listed in table 13-2.

**Table 13-2 Read Addresses of TCNT, TCSR, and RSTCSR**

Address	Register
H'FF10	TCSR
H'FF11	TCNT
H'FF1F	RSTCSR

## 13.3 Operation

This section describes operations when the WDT is used as a watchdog timer and as an interval timer, and the WDT's function in software standby mode.

### 13.3.1 Watchdog Timer Operation

Figure 13-4 illustrates watchdog timer operation. To use the WDT as a watchdog timer, set the WT/IT and TME bits to 1. Software must prevent TCNT overflow by rewriting the TCNT value (normally by writing H'00) before overflow occurs. If TCNT fails to be rewritten and overflows due to a system crash etc., the chip is internally reset for 518 system clock cycles (518 $\phi$ ).

The watchdog reset signal can be externally output from the  $\overline{\text{RESO}}^*$  pin to reset external system devices. The reset signal is output externally for 132 system clock cycles (132 $\phi$ ). External output can be enabled or disabled by the RSTOE bit in RSTCSR.

A watchdog reset has the same vector as a reset generated by input at the  $\overline{\text{RES}}$  pin. Software can distinguish a  $\overline{\text{RES}}$  reset from a watchdog reset by checking the WRST bit in RSTCSR.

If a  $\overline{\text{RES}}$  reset and a watchdog reset occur simultaneously, the  $\overline{\text{RES}}$  reset always takes priority.

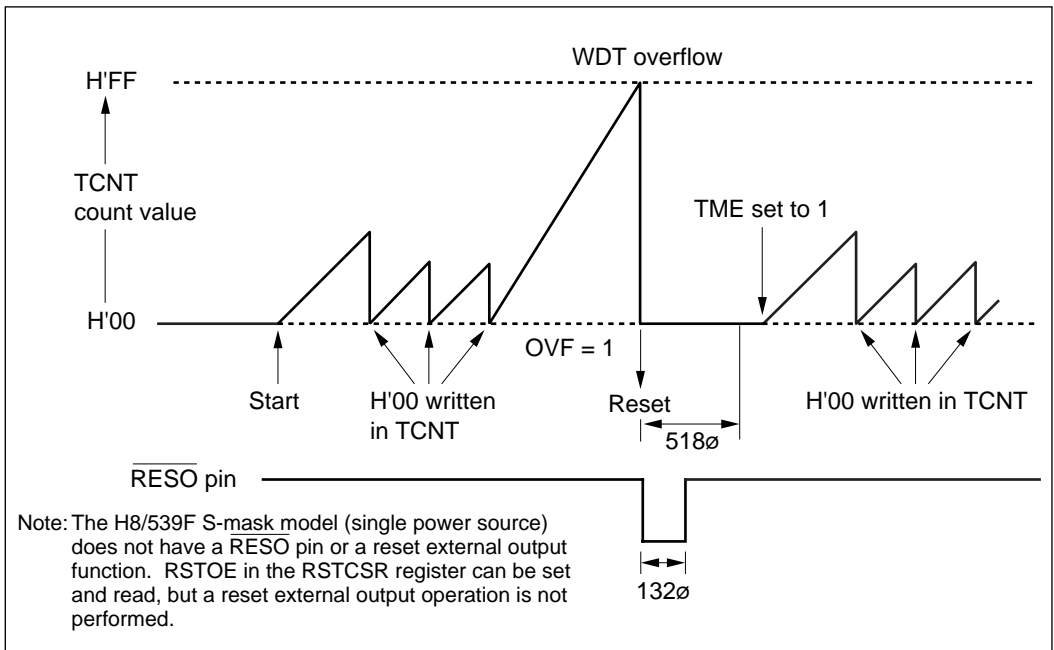
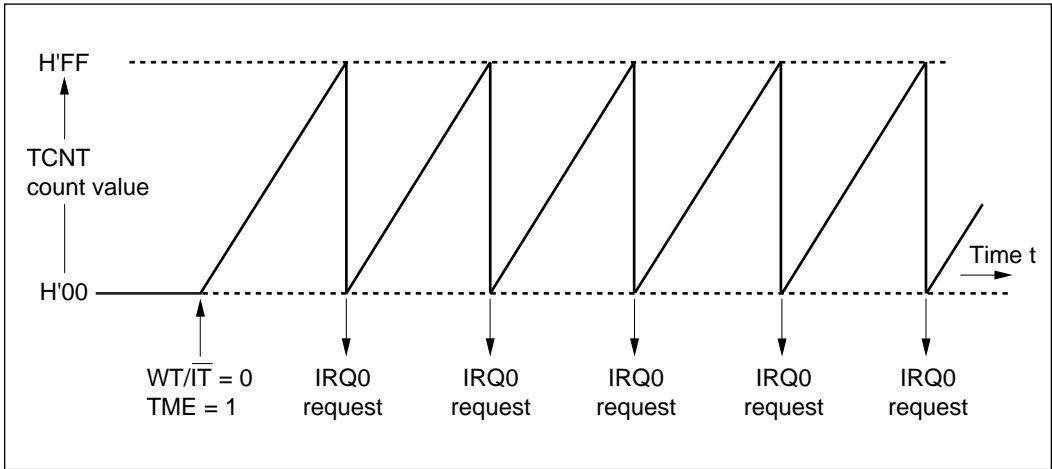


Figure 13-4 Watchdog Timer Operation

### 13.3.2 Interval Timer Operation

Figure 13-5 illustrates interval timer operation. To use the WDT as an interval timer, clear  $\overline{WT/IT}$  to 0 and set  $TME$  to 1. An  $\overline{IRQ0}$  request is generated each time the timer count overflows. This function can be used to generate  $\overline{IRQ0}$  requests at regular intervals.

This  $\overline{IRQ0}$  interrupt has a different vector from the interrupt requested by  $\overline{IRQ0}$  input. Software does not have to check whether the interrupt request came from the  $\overline{IRQ0}$  pin or the interval timer.



**Figure 13-5 Interval Timer Operation**

### 13.3.3 Operation in Software Standby Mode

The watchdog timer has a special function in recovery from software standby mode. WDT settings required when software standby mode is used are described next.

**(1) Before Transition to Software Standby Mode:** The TME bit in the timer control/status register (TCSR) must be cleared to 0 to stop the watchdog timer counter before execution of the SLEEP instruction. The chip cannot enter software standby mode while the TME bit is set to 1. Before entering software standby mode, software should also set bits CKS2 to CKS0 in TCSR so that the overflow interval is equal to or greater than the settling time of the clock oscillator ( $t_{OSC2}$ )\*.

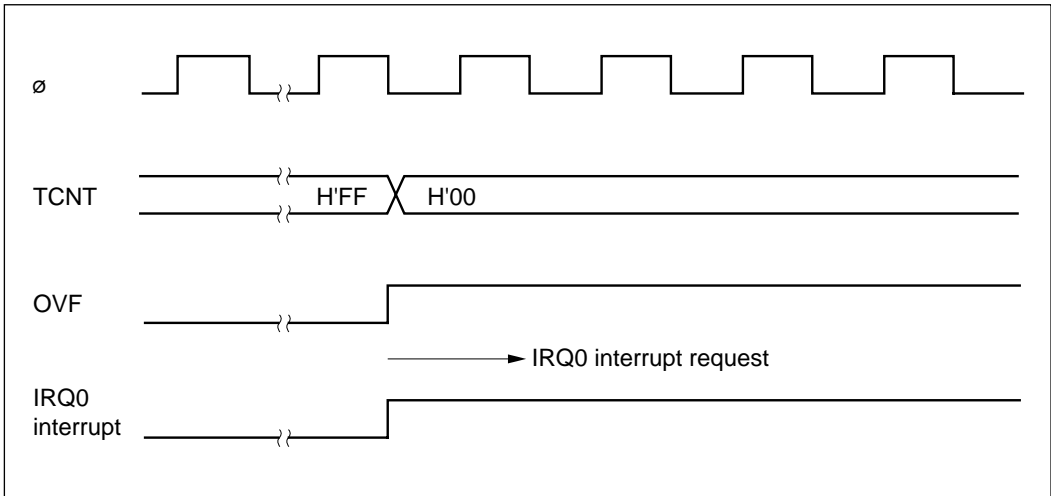
**(2) Recovery from Software Standby Mode:** In recovery from software standby mode the WDT operates as follows.

When an NMI request signal is received, the clock oscillator starts running and the timer counter (TCNT) starts counting at the rate selected by bits CKS2 to CKS0 in TCSR before software standby mode was entered. When TCNT overflows (changes from H'FF to H'00), the system clock ( $\phi$ ) is presumed to be stable and usable, clock signals are supplied to the entire chip, software standby mode ends, and the NMI interrupt-handling routine starts executing. This timer overflow does not set the OVF flag in TCSR to 1, and the TME bit remains cleared to 0.

Note: \* When using an external clock, make a WDT timer control/status register (TCSR) setting that will secure the external clock output settling delay time ( $t_{DEXT}$ ).

### 13.3.4 Timing of Setting of Overflow Flag (OVF)

Figure 13-6 shows the timing of setting of the OVF flag in the timer control/status register (TCSR). The OVF flag is set to 1 when the timer counter overflows. When OVF is set to 1, an IRQ0 interrupt is requested simultaneously.

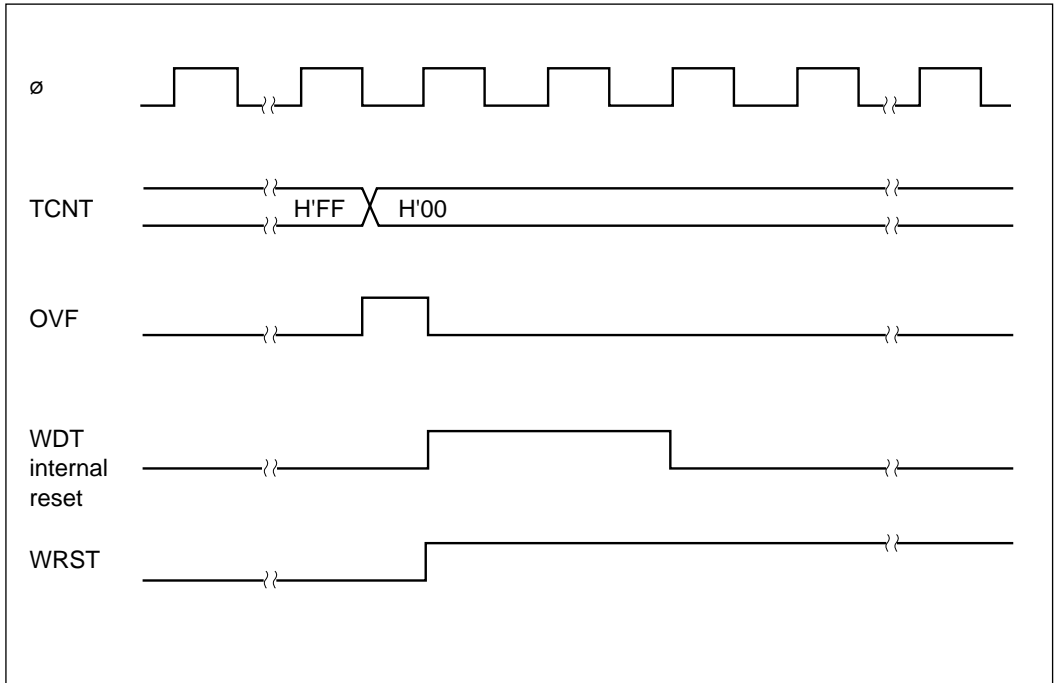


**Figure 13-6 Timing of Setting of OVF**



### 13.3.5 Timing of Setting of Watchdog Timer Reset (WRST)

The WRST bit in the reset control/status register (RSTCSR) is valid when  $WT/\overline{IT} = 1$  and  $TME = 1$ . Figure 13-7 shows the timing of setting of WRST and the internal reset timing. The WRST bit is set to 1 when the timer count overflows and OVF is set to 1. At the same time an internal reset signal is generated for the entire chip. This internal reset signal clears OVF, but the WRST bit remains set to 1. The reset routine must therefore contain an instruction that clears the WRST bit.

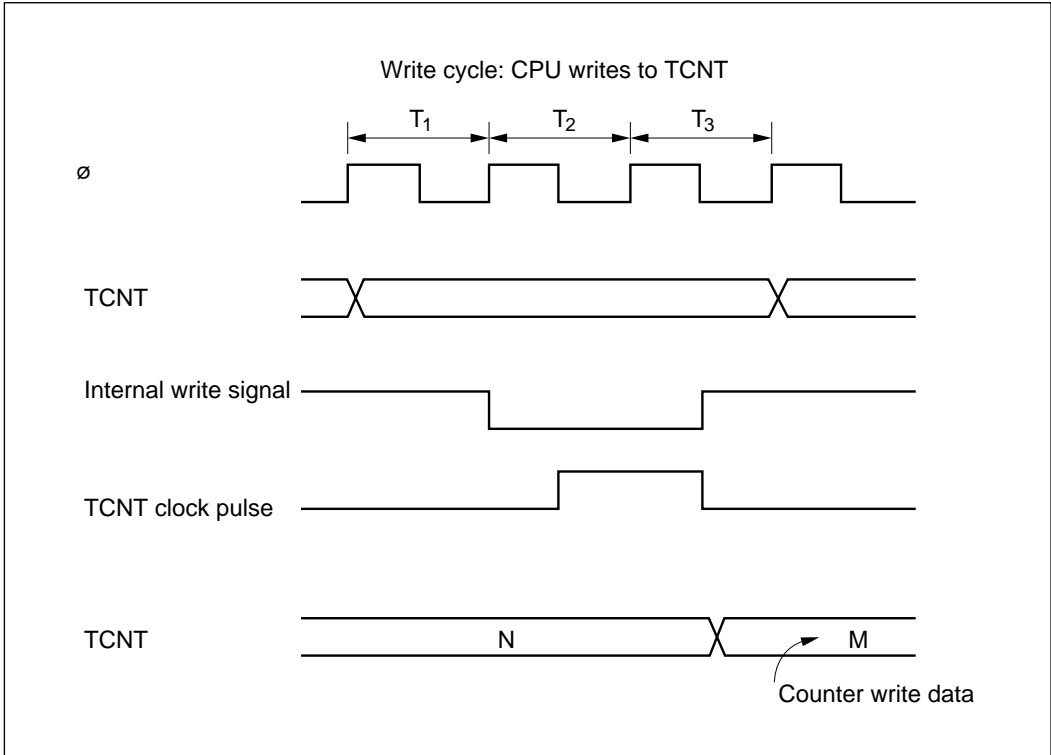


**Figure 13-7 Timing of Setting of WRST Bit and Internal Reset**

## 13.4 Usage Notes

Note the following points when using the watchdog timer.

**(1) Contention between Timer Counter (TCNT) Write and Increment:** If a timer counter clock pulse is generated during the  $T_3$  state of a write cycle to the timer counter, the write takes priority and the timer counter is not incremented. See figure 13-8.



**Figure 13-8 Contention between TCNT Write and Increment**

**(2) Changing CKS2 to CKS0 Values:** Software should stop the watchdog timer (by clearing the TME bit to 0) before changing the values of bits CKS2 to CKS0 in the timer control/status register (TCSR).

# Section 14 Serial Communication Interface

## 14.1 Overview

The H8/539F has an on-chip serial communication interface (SCI) with three independent channels. All channels are functionally identical. The SCI supports both asynchronous and clocked synchronous serial communication. It also has a multiprocessor communication function for serial communication among two or more processors.

### 14.1.1 Features

SCI features are listed below.

- Selection of asynchronous or synchronous mode

#### a. Asynchronous mode

Serial data communication is performed using the start/stop method, in which synchronization is established character by character.

The SCI can communicate with a UART (Universal Asynchronous Receiver/Transmitter), ACIA (Asynchronous Communication Interface Adapter), or other chip that employs standard asynchronous serial communication. It can also communicate with two or more other processors using the multiprocessor communication function. There are twelve selectable serial data communication formats.

- Data length: seven or eight bits
- Stop bit length: one or two bits
- Parity: even, odd, or none
- Multiprocessor bit: one or none
- Receive error detection: parity, overrun, and framing errors
- Break detection: by reading the RXD level directly when a framing error occurs

#### b. Clocked synchronous mode

Serial data communication is synchronized with a clock signal. The SCI can communicate with other chips having a clocked synchronous communication function. There is one serial data communication format.

- Data length: eight bits
- Receive error detection: overrun errors

- Full duplex communication

The transmitting and receiving sections are independent, so the SCI can transmit and receive simultaneously. Both sections use double buffering, so continuous data transfer is possible in both the transmit and receive directions.

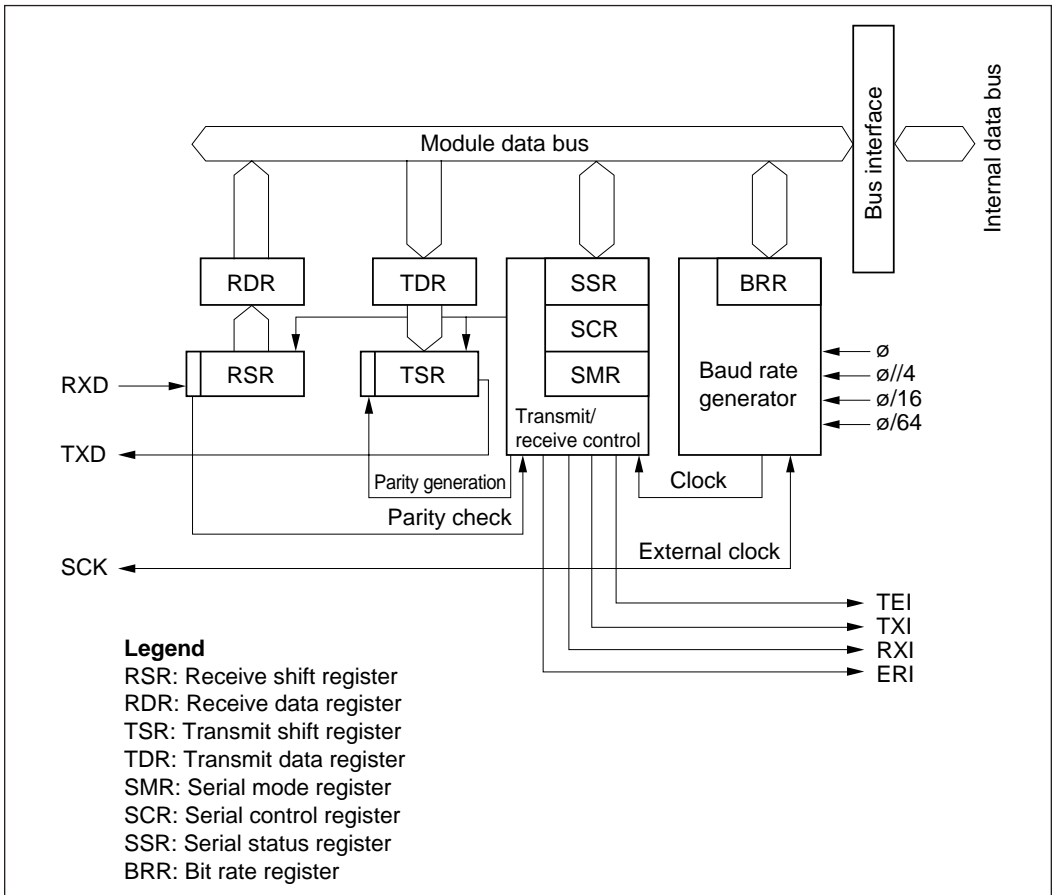
- Built-in baud rate generator with selectable bit rates
- Internal or external transmit/receive clock source: baud rate generator or SCK pin
- Four types of interrupts

Transmit-data-empty, transmit-end, receive-data-full, and receive-error interrupts are requested independently. The transmit-data-empty and receive-data-full interrupts can be served by the on-chip data transfer controller (DTC) to transfer data.

In the H8/539F, SCI2 and SCI3 have the same interrupt vectors.

### 14.1.2 Block Diagram

Figure 14-1 shows a block diagram of the SCI.



**Figure 14-1 SCI Block Diagram**

### 14.1.3 Input/Output Pins

Table 14-1 summarizes the serial communication pins for each SCI channel.

**Table 14-1 SCI Pins**

Channel	Pin Name	Abbreviation	Input/Output	Function
1	Serial clock pin	SCK <sub>1</sub>	Input/output	SCI1 clock input/output
	Receive data pin	RXD <sub>1</sub>	Input	SCI1 receive data input
	Transmit data pin	TXD <sub>1</sub>	Output	SCI1 transmit data output
2	Serial clock pin	SCK <sub>2</sub>	Input/output	SCI2 clock input/output
	Receive data pin	RXD <sub>2</sub>	Input	SCI2 receive data input
	Transmit data pin	TXD <sub>2</sub>	Output	SCI2 transmit data output
3	Serial clock pin	SCK <sub>3</sub>	Input/output	SCI3 clock input/output
	Receive data pin	RXD <sub>3</sub>	Input	SCI3 receive data input
	Transmit data pin	TXD <sub>3</sub>	Output	SCI3 transmit data output

### 14.1.4 Register Configuration

Table 14-2 summarizes the SCI registers. These registers select the communication mode (asynchronous or clocked synchronous), specify the data format and bit rate, and control the transmitter and receiver sections.

**Table 14-2 Channel 1 Registers**

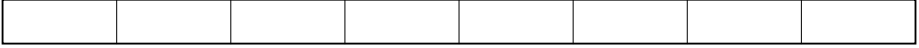
<b>Channel</b>	<b>Address</b>	<b>Name</b>	<b>Abbreviation</b>	<b>R/W</b>	<b>Initial Value</b>
1	H'FEC8	Serial mode register	SMR	R/W	H'00
	H'FEC9	Bit rate register	BRR	R/W	H'FF
	H'FECA	Serial control register	SCR	R/W	H'00
	H'FECB	Transmit data register	TDR	R/W	H'FF
	H'FECC	Serial status register	SSR	R/(W)*	H'84
	H'FECD	Receive data register	RDR	R	H'00
2	H'FED0	Serial mode register	SMR	R/W	H'00
	H'FED1	Bit rate register	BRR	R/W	H'FF
	H'FED2	Serial control register	SCR	R/W	H'00
	H'FED3	Transmit data register	TDR	R/W	H'FF
	H'FED4	Serial status register	SSR	R/(W)*	H'84
	H'FED5	Receive data register	RDR	R	H'00
3	H'FEC0	Serial mode register	SMR	R/W	H'00
	H'FEC1	Bit rate register	BRR	R/W	H'FF
	H'FEC2	Serial control register	SCR	R/W	H'00
	H'FEC3	Transmit data register	TDR	R/W	H'FF
	H'FEC4	Serial status register	SSR	R/(W)*	H'84
	H'FEC5	Receive data register	RDR	R	H'00

Note: \* Software can write 0 to clear flags, but cannot write 1.

## 14.2 Register Descriptions

### 14.2.1 Receive Shift Register


The receive shift register (RSR) receives serial data.

Bit	7	6	5	4	3	2	1	0
								
R/W	—	—	—	—	—	—	—	—

Data input at the RXD pin are loaded into RSR in the order received, LSB (bit 0) first. In this way the SCI converts received data to parallel form. When one byte has been received, it is automatically transferred to the receive data register (RDR). The H8/500 CPU cannot read or write RSR directly.

### 14.2.2 Receive Data Register

The receive data register (RDR) stores serial receive data.

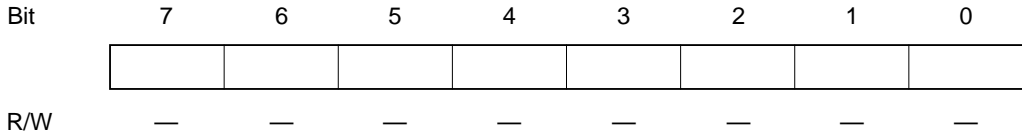
Bit	7	6	5	4	3	2	1	0
								
Initial value	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R

The SCI completes the reception of one byte of serial data by moving the received data from the receive shift register (RSR) into RDR for storage. RSR is then ready to receive the next data. This double buffering allows the SCI to receive data continuously.

The H8/500 CPU can read but not write RDR. RDR is initialized to H'00 by a reset and in the standby modes.

### 14.2.3 Transmit Shift Register

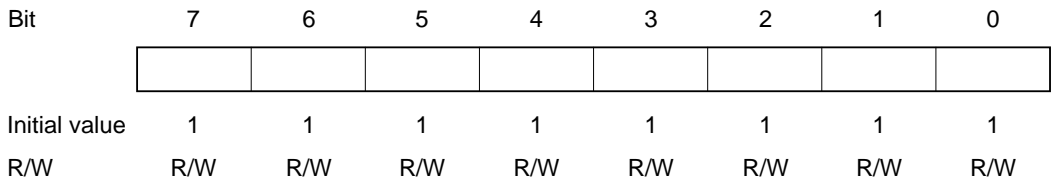
The transmit shift register (TSR) transmits serial data.



The SCI loads transmit data from the transmit data register (TDR) into TSR, then transmits the data serially from the TXD pin, LSB (bit 0) first. After transmitting one data byte, the SCI automatically loads the next transmit data from TDR into TSR and starts transmitting again. If TDRE is set to 1, however, the SCI does not load the TDR contents into TSR. The H8/500 CPU cannot read or write TSR directly.

### 14.2.4 Transmit Data Register

The transmit data register (TDR) is an eight-bit register that stores data for serial transmission.



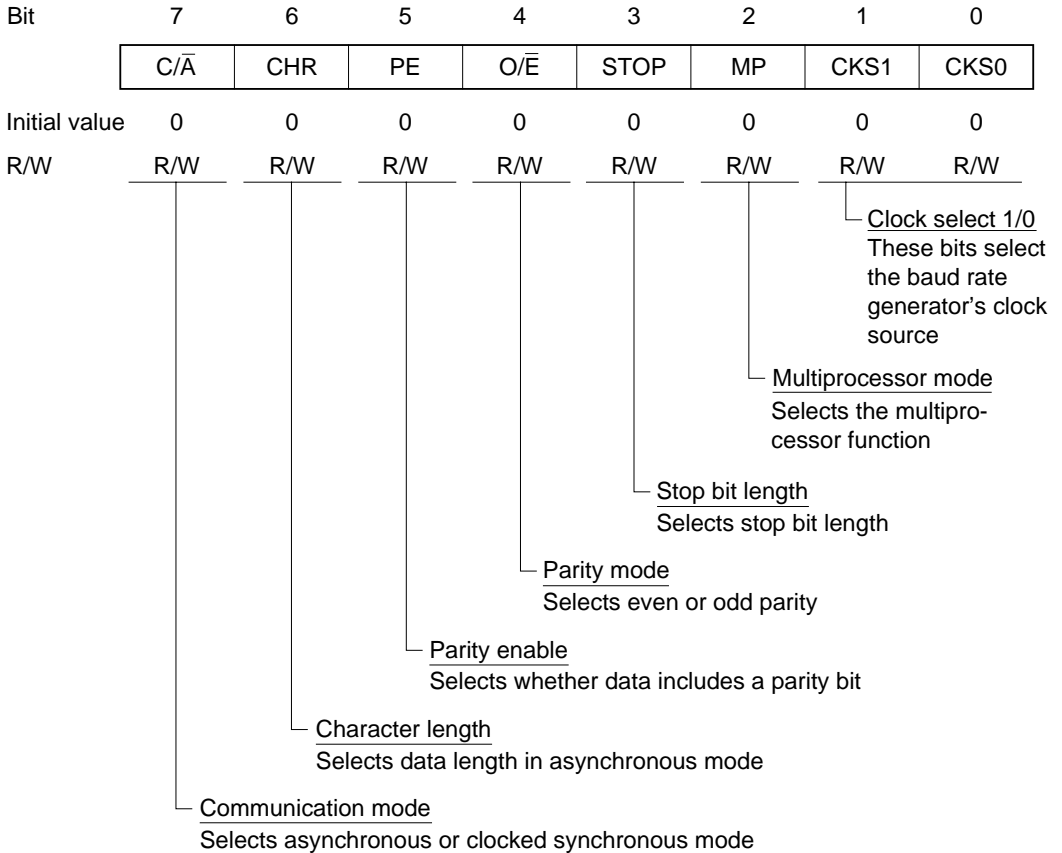
When the SCI detects that the transmit shift register (TSR) is empty, it moves transmit data written in TDR into TSR and starts serial transmission. Continuous serial transmission is possible by writing the next transmit data in TDR during serial transmission from TSR.

The H8/500 CPU can always read and write TDR. TDR is initialized to H'FF by a reset and in the standby modes.



### 14.2.5 Serial Mode Register

The serial mode register (SMR) is an eight-bit register that specifies the SCI serial communication format and selects the clock source for the baud rate generator.



The H8/500 CPU can always read and write SMR. SMR is initialized to H'00 by a reset and in the standby modes.

(1) **Bit 7—Communication Mode (C/A):** Selects whether the SCI operates in asynchronous or clocked synchronous mode.

**Bit 7**

C/A	Description
0	Asynchronous mode (Initial value)
1	Clocked synchronous mode

(2) **Bit 6—Character Length (CHR):** Selects seven-bit or eight-bit data in asynchronous mode. In clocked synchronous mode the data length is always eight bits, regardless of the CHR setting.

**Bit 6**

CHR	Description
0	Eight-bit data (Initial value)
1	Seven-bit data*

Note: \* When seven-bit data is selected, the MSB of the transmit data register (bit 7) is not transmitted.

(3) **Bit 5—Parity Enable (PE):** Selects whether to add a parity bit to transmit data and check parity of receive data, in asynchronous mode. In clocked synchronous mode the parity bit is neither added nor checked, regardless of the PE setting.

**Bit 5**

PE	Description
0	Parity bit not added or checked (Initial value)
1	Parity bit added and checked*

Note: \* When PE is set to 1 an even or odd parity bit is added to transmit data, depending on the parity mode (O/E) setting. Receive data parity is checked according to the even/odd (O/E) mode setting.

**(4) Bit 4—Parity Mode ( $O/\bar{E}$ ):** Selects even or odd parity when parity bits are added and checked. The  $O/\bar{E}$  setting is used only in asynchronous mode and only when the parity enable bit (PE) is set to 1 to enable parity generation and checking. The  $O/\bar{E}$  setting is ignored in clocked synchronous mode, or in asynchronous mode when parity is disabled.

**Bit 4**

$O/\bar{E}$	Description
0	Even parity* <sup>1</sup> (Initial value)
1	Odd parity* <sup>2</sup>

- Notes:
1. If even parity is selected, the parity bit added to transmit data makes an even number of 1s in the transmitted character and parity bit combined. Receive data must have an even number of 1s in the received character and parity bit combined.
  2. If odd parity is selected, the parity bit added to transmit data makes an odd number of 1s in the transmitted character and parity bit combined. Receive data must have an odd number of 1s in the received character and parity bit combined.

**(5) Bit 3—Stop Bit Length (STOP):** Selects one or two bits as the stop bit length in asynchronous mode. This setting is used only in asynchronous mode. It is ignored in clocked synchronous mode because no stop bits are added.

**Bit 3**

STOP	Description
0	One stop bit* <sup>1</sup> (Initial value)
1	Two stop bits* <sup>2</sup>

- Notes:
1. In transmitting, a single 1 bit (Stop bit) is added at the end of each transmitted character.
  2. In transmitting, two 1 bits (Stop bit) are added at the end of each transmitted character.

In receiving, only the first stop bit is checked, regardless of the STOP bit setting. If the second stop bit is 1 it is treated as a stop bit. If the second stop bit is 0 it is treated as the start bit of the next incoming character.

**(6) Bit 2—Multiprocessor Mode (MP):** Selects a multiprocessor format. When a multiprocessor format is selected, settings of the parity enable (PE) and parity mode (O/ $\bar{E}$ ) bits are ignored. The MP bit setting is used only in asynchronous mode; it is ignored in clocked synchronous mode. For the multiprocessor communication function, see section 14.3.4, “Multiprocessor Communication.”

**Bit 2**

<b>MP</b>	<b>Description</b>
0	Multiprocessor function disabled (Initial value)
1	Multiprocessor format selected

**(7) Bits 1 and 0—Clock Select 1 and 0 (CKS1/0):** These bits select the internal clock source of the on-chip baud rate generator. Four clock sources are available:  $\emptyset$ ,  $\emptyset/4$ ,  $\emptyset/16$ , and  $\emptyset/64$ . For further information on the clock source, bit rate register settings, and bit rate, see section 14.2.8, “Bit Rate Register.”

<b>Bit 1</b>	<b>Bit 0</b>	<b>Description</b>
<b>CKS1</b>	<b>CKS0</b>	
0	0	System clock ( $\emptyset$ ) (Initial value)
0	1	$\emptyset/4$
1	0	$\emptyset/16$
1	1	$\emptyset/64$

## 14.2.6 Serial Control Register

The serial control register (SCR) enables the SCI transmitter and receiver, selects serial clock output in asynchronous mode, enables and disables interrupts, and selects the transmit/receive clock.

Bit	7	6	5	4	3	2	1	0
	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- Bit 7: Transmit interrupt enable  
Enables and disables transmit-data-empty interrupts (TXI)
- Bit 6: Receive interrupt enable  
Enables and disables receive-data-full interrupts (RXI) and receive error interrupts (ERI)
- Bit 5: Transmit enable  
Enables and disables the transmitter
- Bit 4: Receive enable  
Enables and disables the receiver
- Bit 3: Multiprocessor interrupt enable  
Enables and disables multiprocessor interrupts
- Bit 2: Transmit end interrupt enable  
Enables and disables transmit-end interrupts (TEI)
- Bit 1: Clock enable 1/0  
Selects the SCI clock source
- Bit 0: Clock enable 1/0  
Selects the SCI clock source

The H8/500 CPU can always read and write SCR. SCR is initialized to H'00 by a reset and in the standby modes.

**(1) Bit 7—Transmit Interrupt Enable (TIE):** Enables or disables the transmit-data-empty interrupt (TXI) requested when the transmit data register empty bit (TDRE) in the serial status register (SSR) is set to 1 due to transfer of serial transmit data from TDR to TSR.

#### Bit 7

TIE	Description
0	Transmit-data-empty interrupt request (TXI) is disabled* (Initial value)
1	Transmit-data-empty interrupt request (TXI) is enabled

Note: \* The TXI interrupt request can be cleared by reading TDRE after it has been set to 1, then clearing TDRE to 0, or by clearing TIE to 0.

**(2) Bit 6—Receive Interrupt Enable (RIE):** Enables or disables the receive-data-full interrupt (RXI) requested when the receive data register full bit (RDRF) in the serial status register (SSR) is set to 1 due to transfer of serial receive data from RSR to RDR. Also enables or disables receive-error interrupt (ERI) requests.

#### Bit 6

RIE	Description
0	Receive-data-full interrupt (RXI) and receive-error interrupt (ERI) requests are disabled* (Initial value)
1	Receive-data-full interrupt (RXI) and receive-error interrupt (ERI) requests are enabled

Note: \* RXI and ERI interrupt requests can be cleared by reading the RDRF flag or error flag (FER, PER, or ORER) after it has been set to 1, then clearing the flag to 0, or by clearing RIE to 0.

**(3) Bit 5—Transmit Enable (TE):** Enables or disables the SCI transmitter.

#### Bit 5

TE	Description
0	Transmitter disabled* <sup>1</sup> , TXD pin available for general-purpose I/O (Initial value)
1	Transmitter enabled* <sup>2</sup> , TXD used for transmit data output

Notes: 1. The transmit data register empty bit (TDRE) in the serial status register (SSR) is locked at 1.  
 2. Serial transmitting starts when the transfer data register empty (TDRE) bit in the serial status register (SSR) is cleared to 0 after writing of transmit data into TDR. Select the transmit format in SMR before setting TE to 1.

(4) **Bit 4—Receive Enable (RE):** Enables or disables the SCI receiver.

**Bit 4**

RE	Description
0	Receiver disabled* <sup>1</sup> , RXD pin available for general-purpose I/O (Initial value)
1	Receiver enabled* <sup>2</sup> , RXD used for receive data input

- Notes:
1. Clearing RE to 0 does not affect the receive flags (RDRF, FER, PER, ORER). These flags retain their previous values.
  2. Serial receiving starts when a start bit is detected in asynchronous mode, or serial clock input is detected in clocked synchronous mode. Select the receive format in SMR before setting RE to 1.

(5) **Bit 3—Multiprocessor Interrupt Enable (MPIE):** Enables or disables multiprocessor interrupts. The MPIE setting is used only in asynchronous mode, and only if the multiprocessor mode bit (MP) in the serial mode register (SMR) is set to 1. The MPIE setting is ignored in clocked synchronous mode or when the MP bit is cleared to 0.

**Bit 3**

MPIE	Description
0	Multiprocessor interrupts are disabled (normal receive operation) (Initial value)
1	Multiprocessor interrupts are enabled.* Until data is received in which the multiprocessor bit set to 1 receive-data-full interrupt requests (RXI), receive-error interrupt requests (ERI), and setting of the RDRF, FER, and ORER status flags in the serial status register (SSR) are disabled. MPIE is cleared to 0 when: <ol style="list-style-type: none"><li>1. MPIE is cleared to 0, or</li><li>2. Multiprocessor bit (MPB) is set to 1 in receive data.</li></ol>

Note: \* The SCI does not transfer receive data from RSR to RDR, does not detect receive errors, and does not set the RDRF, FER, and ORER flags in the serial status register (SSR). When it receives data with the multiprocessor bit (MPB) set to 1, the SCI automatically clears MPIE to 0, enables RXI and ERI interrupts (if the TIE and RIE bits in SCR are set to 1), and allows FER and ORER to be set.

**(6) Bit 2—Transmit-End Interrupt Enable (TEIE):** Enables or disables the transmit-end interrupt (TEI) requested if TDR does not contain effective transmit data when the MSB is transmitted.

## Bit 2

TEIE	Description
0	Transmit-end interrupt (TEI) requests are disabled* (Initial value)
1	Transmit-end interrupt (TEI) requests are enabled*

Note: \* The TEI request can be cleared by reading the TDRE bit in the serial status register (SSR) after it has been set to 1, then clearing TDRE to 0, thereby clearing the transmit end (TEND) bit to 0; or by clearing the TEIE bit to 0.

**(7) Bits 1 and 0—Clock Enable 1 and 0 (CKE1/0):** These bits select the SCI clock source and enable or disable clock output from the SCK pin. Depending on the combination of CKE1 and CKE0, the SCK pin can be used for general-purpose input/output, serial clock output, or serial clock input.

The CKE0 setting is valid only in asynchronous mode, and only when the SCI is internally clocked (CKE1 = 0). The CKE0 setting is ignored in clocked synchronous mode, or when an external clock source is selected (CKE1 = 1). Select the SCI operating mode in the serial mode register (SMR) before setting CKE1 and CKE0. For further details on selection of the SCI clock source, see table 14-9 in section 14.3, “Operation.”

## Bit 1 Bit 0

CKE1	CKE0	Description	
0	0	Asynchronous mode	Internal clock, SCK pin available for general-purpose input/output* <sup>1</sup>
		Clocked synchronous mode	Internal clock, SCK pin used for serial clock output* <sup>1</sup>
0	1	Asynchronous mode	Internal clock, SCK pin used for clock output* <sup>2</sup>
		Clocked synchronous mode	Internal clock, SCK pin used for serial clock output
1	0	Asynchronous mode	External clock, SCK pin used for clock input* <sup>3</sup>
		Clocked synchronous mode	External clock, SCK pin used for serial clock input
1	1	Asynchronous mode	External clock, SCK pin used for clock input* <sup>3</sup>
		Clocked synchronous mode	External clock, SCK pin used for serial clock input

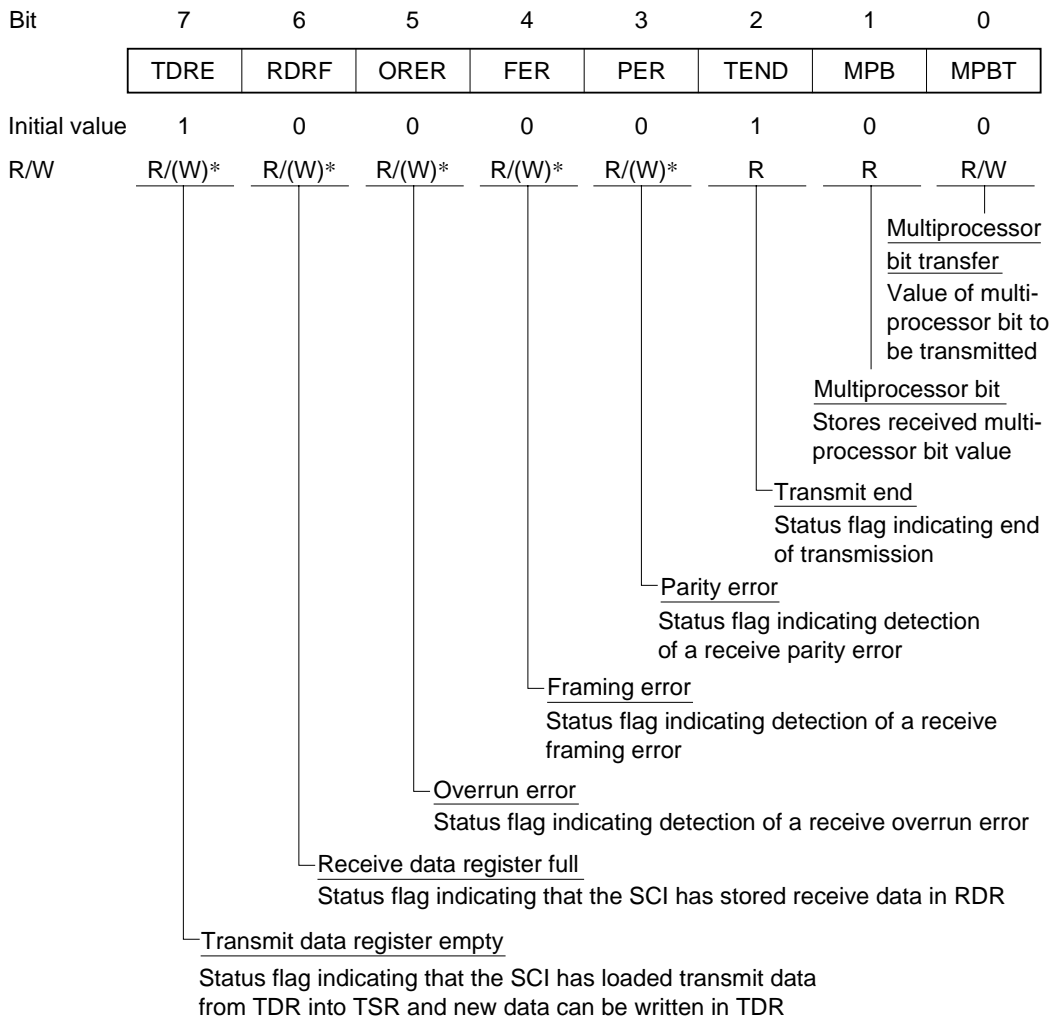
Notes: 1. Initial value  
 2. The output clock frequency is the same as the bit rate.  
 3. The input clock frequency is 16 times the bit rate.



## 14.2.7 Serial Status Register

The serial status register (SSR) is an eight-bit register containing multiprocessor bit values, and status flags that indicate SCI operating status.

The H8/500 CPU can always read and write SSR, but cannot write 1 in the status flags (TDRE, RDRF, ORER, PER, and FER). These flags can be cleared to 0 only if they have first been read after being set to 1. Bits 2 (TEND) and 1 (MPB) are read-only bits and cannot be written.



Note: \* Software can write 0 to clear the flag, but cannot write 1.

SSR is initialized to H'84 by a reset and in the standby modes.

**(1) Bit 7—Transmit Data Register Empty (TDRE):** Indicates that the SCI has loaded transmit data from TDR into TSR and new data can be written in TDR.

#### Bit 7

TDRE	Description
0	TDR contains valid transmit data TDRE is cleared to 0 when: 1. Software reads TDRE after it has been set to 1, then writes 0 in TDRE 2. The DTC writes data in TDR
1	TDR does not contain valid transmit data (Initial value) TDRE is set to 1 when: 1. The chip is reset or enters standby mode 2. The TE bit in the serial control register (SCR) is cleared to 0, or 3. TDR contents are loaded into TSR, so new data can be written in TDR

**(2) Bit 6—Receive Data Register Full (RDRF):** Indicates that RDR contains new receive data.

#### Bit 6

RDRF	Description
0	RDR does not contain new receive data (Initial value) RDRF is cleared to 0 when: 1. The chip is reset or enters standby mode 2. Software reads RDRF after it has been set to 1, then writes 0 in RDRF 3. The DTC reads data from RDR
1	RDR contains new receive data RDRF is set to 1 when serial data are received normally and transferred from RSR to RDR.

Note: RDR and RDRF are not affected by detection of receive errors or by clearing of the RE bit to 0 in the serial control register. They retain their previous contents. If RDRF is still set to 1 when reception of the next data ends, an overrun error (ORER) occurs and receive data is lost.

**(3) Bit 5—Overrun Error (ORER):** Indicates that data reception ended abnormally due to an overrun error.

#### Bit 5

ORER	Description
0	Receiving is in progress or has ended normally (Initial value)*1 ORER is cleared to 0 when: 1. The chip is reset or enters standby mode 2. Software reads ORER after it has been set to 1, then writes 0 in ORER
1	A receive overrun error occurred*2 ORER is set to 1 if reception of the next serial data ends when RDRF is set to 1

Notes: 1. Clearing the RE bit to 0 in the serial control register does not affect the ORER bit, which retains its previous value.  
2. RDR continues to hold the receive data before the overrun error, so subsequent receive data are lost. Serial receiving cannot continue while ORER is set to 1. In clocked synchronous mode, serial transmitting is also disabled.

**(4) Bit 4—Framing Error (FER):** Indicates that data reception ended abnormally due to a framing error in asynchronous mode.

#### Bit 4

FER	Description
0	Receiving is in progress or has ended normally (Initial value)*1 FER is cleared to 0 when: 1. The chip is reset or enters standby mode 2. Software reads FER after it has been set to 1, then writes 0 in FER
1	A receive framing error occurred FER is set to 1 if the stop bit at the end of receive data is checked and found to be 0*2.

Notes: 1. Clearing the RE bit to 0 in the serial control register does not affect the FER bit, which retains its previous value.  
2. When the stop bit length is two bits, only the first bit is checked. The second stop bit is not checked. When a framing error occurs the SCI transfers the receive data into RDR but does not set RDRF. Serial receiving cannot continue while FER is set to 1. In clocked synchronous mode, serial transmitting is also disabled.

**(5) Bit 3—Parity Error (PER):** Indicates that data reception ended abnormally due to a parity error in asynchronous mode.

### Bit 3

PER	Description
0	Receiving is in progress or has ended normally*1 (Initial value) PER is cleared to 0 when: 1. The chip is reset or enters standby mode 2. Software reads PER after it has been set to 1, then writes 0 in PER
1	A receive parity error occurred*2 PER is set to 1 if the number of 1s in receive data, including the parity bit, does not match the even or odd parity setting of the parity mode bit (O/E) in the serial mode register (SMR).

Notes: 1. Clearing the RE bit to 0 in the serial control register does not affect the PER bit, which retains its previous value.  
2. When a parity error occurs the SCI transfers the receive data into RDR but does not set RDRF. Serial receiving cannot continue while PER is set to 1. In clocked synchronous mode, serial transmitting is also disabled.

**(6) Bit 2—Transmit End (TEND):** Indicates that when the last bit of a serial character was transmitted TDR did not contain new transmit data, so transmission has ended. TEND is a read-only bit and cannot be written.

### Bit 2

TEND	Description
0	Transmission is in progress TEND is cleared to 0 when: 1. Software reads TDRE after it has been set to 1, then writes 0 in TDRE 2. The DTC writes data in TDR
1	End of transmission (Initial value) TEND is set to 1 when: 1. The chip is reset or enters standby mode 2. TE is cleared to 0 in the serial control register (SCR) 3. TDRE is 1 when the last bit of a serial character (1 byte) is transmitted

**(7) Bit 1—Multiprocessor Bit (MPB):** Stores the value of the multiprocessor bit in receive data when a multiprocessor format is used in asynchronous mode. MPB is a read-only bit and cannot be written.

**Bit 1**

<b>MPB</b>	<b>Description</b>
0	Multiprocessor bit value in receive data is 0* (Initial value)
1	Multiprocessor bit value in receive data is 1

Note: \* If RE is cleared to 0 when a multiprocessor format is selected, MPB retains its previous value.

**(8) Bit 0—Multiprocessor Bit Transfer (MPBT):** Stores the value of the multiprocessor bit added to transmit data when a multiprocessor format is selected for transmitting in asynchronous mode. The MPBT setting is ignored in clocked synchronous mode, when a multiprocessor format is not selected, or when the SCI is not transmitting.

**Bit 0**

<b>MPBT</b>	<b>Description</b>
0	Multiprocessor bit value in transmit data is 0 (Initial value)
1	Multiprocessor bit value in transmit data is 1

### 14.2.8 Bit Rate Register

The bit rate register (BRR) is an eight-bit register that, together with the CKS1 and CKS0 bits in the serial mode register (SMR) that select the baud rate generator clock source, determines the serial transmit/receive bit rate.

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The H8/500 CPU can always read and write BRR. BRR is initialized to H'FF by a reset and in the standby modes. SCI1 and SCI2 have independent baud rate generator control, so different values can be set in the two channels.

Table 14-3 shows examples of BRR settings in asynchronous mode.

**Table 14-3 Examples of Bit Rates and BRR Settings in Asynchronous Mode (1)**

Bit Rate (bits/s)	$\phi$ (MHz)											
	1			1.2288			2			2.097152		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	1	70	+0.03	1	86	+0.31	1	141	+0.03	1	148	-0.04
150	0	207	+0.16	0	255	0	1	103	+0.16	1	108	+0.21
300	0	103	+0.16	0	127	0	0	207	+0.16	0	217	+0.21
600	0	51	+0.16	0	63	0	0	103	+0.16	0	108	+0.21
1200	0	25	+0.16	0	31	0	0	51	+0.16	0	54	-0.70
2400	0	12	+0.16	0	15	0	0	25	+0.16	0	26	+1.14
4800	—	—	—	0	7	0	0	12	+0.16	0	13	-2.48
9600	—	—	—	0	3	0	—	—	—	—	—	—
19200	—	—	—	0	1	0	—	—	—	—	—	—
31250	0	0	0.00	—	—	—	0	1	0	—	—	—
38400	—	—	—	0	0	0	—	—	—	—	—	—

**Table 14-3 Examples of Bit Rates and BRR Settings in Asynchronous Mode (2)**

Bit Rate (bits/s)	$\phi$ (MHz)											
	2.4576			3			3.6864			4		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	1	174	-0.26	1	212	+0.03	2	64	+0.70	2	70	+0.03
150	1	127	0	1	155	+0.16	1	191	0	1	207	+0.16
300	0	255	0	1	77	+0.16	1	95	0	1	103	+0.16
600	0	127	0	0	155	+0.16	0	191	0	0	207	+0.16
1200	0	63	0	0	77	+0.16	0	95	0	0	103	+0.16
2400	0	31	0	0	38	+0.16	0	47	0	0	51	+0.16
4800	0	15	0	0	19	-2.34	0	23	0	0	25	+0.16
9600	0	7	0	0	9	-2.34	0	11	0	0	12	+0.16
19200	0	3	0	0	4	-2.34	0	5	0	—	—	—
31250	—	—	—	0	2	0	—	—	—	0	3	0
38400	0	1	0	—	—	—	0	2	0	—	—	—

**Table 14-3 Examples of Bit Rates and BRR Settings in Asynchronous Mode (3)**

Bit Rate (bits/s)	$\phi$ (MHz)											
	4.9152			5			6			6.144		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	86	+0.31	2	88	-0.25	2	106	-0.44	2	108	+0.08
150	1	255	0	2	64	+0.16	2	77	0	2	79	0
300	1	127	0	1	129	+0.16	1	155	0	1	159	0
600	0	255	0	1	64	+0.16	1	77	0	1	79	0
1200	0	127	0	0	129	+0.16	0	155	+0.16	0	159	0
2400	0	63	0	0	64	+0.16	0	77	+0.16	0	79	0
4800	0	31	0	0	32	-1.36	0	38	+0.16	0	39	0
9600	0	15	0	0	15	+1.73	0	19	-2.34	0	19	0
19200	0	7	0	0	7	+1.73	—	—	—	0	9	0
31250	0	4	-1.70	0	4	0	0	5	0	0	5	+2.40
38400	0	3	0	0	3	+1.73	—	—	—	0	4	0

**Table 14-3 Examples of Bit Rates and BRR Settings in Asynchronous Mode (4)**

Bit Rate (Bits/s)	$\phi$ (MHz)											
	7.3728			8			9.8304			10		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	130	-0.07	2	141	+0.03	2	174	-0.26	3	43	+0.88
150	2	95	0	2	103	+0.16	2	127	0	2	129	+0.16
300	1	191	0	1	207	+0.16	1	255	0	2	64	+0.16
600	1	95	0	1	103	+0.16	1	127	0	1	129	+0.16
1200	0	191	0	0	207	+0.16	0	255	0	1	64	+0.16
2400	0	95	0	0	103	+0.16	0	127	0	0	129	+0.16
4800	0	47	0	0	51	+0.16	0	63	0	0	64	+0.16
9600	0	23	0	0	25	+0.16	0	31	0	0	32	-1.36
19200	0	11	0	0	12	+0.16	0	15	0	0	15	+1.73
31250	—	—	—	0	7	0	0	9	-1.70	0	9	0
38400	0	5	0	—	—	—	0	7	0	0	7	+1.73
307200	—	—	—	—	—	—	0	0	0	—	—	—
312500	—	—	—	—	—	—	—	—	—	0	0	0

**Table 14-3 Examples of Bit Rates and BRR Settings in Asynchronous Mode (5)**

Bit Rate (Bits/s)	$\phi$ (MHz)											
	12			12.288			14			14.7456		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	212	0.03	2	217	0.08	2	248	-0.17	3	64	0.07
150	2	155	0.16	2	159	0.00	2	181	0.16	2	191	0.00
300	2	77	0.16	2	79	0.00	2	90	0.16	2	95	0.00
600	1	155	0.16	1	159	0.00	1	181	0.16	1	191	0.00
1200	1	77	0.16	1	79	0.00	1	90	0.16	1	95	0.00
2400	0	155	0.16	0	159	0.00	0	181	0.16	0	191	0.00
4800	0	77	0.16	0	79	0.00	0	90	0.16	0	95	0.00
9600	0	38	0.16	0	39	0.00	0	45	-0.93	0	47	0.00
19200	0	19	-2.34	0	19	0.00	0	22	-0.93	0	23	0.00
31250	0	11	0.00	0	11	2.40	0	13	0.00	0	14	-1.70
38400	0	9	-2.34	0	9	0.00	0	10	3.57	0	11	0.00



**Table 14-3 Examples of Bit Rates and BRR Settings in Asynchronous Mode (6)**

Bit Rate (Bits/s)	ø (MHz)		
	16		
	n	N	Error (%)
110	3	70	0.03
150	2	207	0.16
300	2	103	0.16
600	1	207	0.16
1200	1	103	0.16
2400	0	207	0.16
4800	0	103	0.16
9600	0	51	0.16
19200	0	25	0.16
31250	0	15	0.00
38400	0	12	0.16

- Notes: 1. Settings with an error of 1% or less are recommended.  
 2. The BRR setting is calculated as follows:

$$N = \lceil \frac{\text{ø}}{64 \times 2^{2n-1} \times B} \rceil \times 10^6 - 1$$

B: Bit rate

N: BRR setting for baud rate generator ( $0 \leq N \leq 255$ )

ø: Operation frequency (MHz)

n: Baud rate generator clock source ( $n = 0, 1, 2, 3$ )

(For the clock sources and values of n, see table 14-4.)

**Table 14-4 Clock Sources and n**

n	Clock Source	SMR Settings	
		CKS1	CKS0
0	ø	0	0
1	ø/4	0	1
2	ø/16	1	0
3	ø/64	1	1

3. Error is calculated as follows:

$$\text{Error (\%)} = \left\{ \frac{\text{ø}}{[(N + 1) \times B \times 64^{2n-1}] \times 10^6 - 1} \right\} \times 100$$

Tables 14-5 and 14-6 indicate the maximum bit rates in asynchronous mode for various system clock frequencies.

**Table 14-5 Maximum Bit Rates for Various Frequencies (Asynchronous Mode)**

$\phi$ (MHz)	Maximum Bit Rate (Bits/s)	Settings	
		n	N
1	31250	0	0
1.2288	38400	0	0
2	62500	0	0
2.097152	65536	0	0
2.4576	76800	0	0
3	93750	0	0
3.6864	115200	0	0
4	125000	0	0
4.9152	153600	0	0
5	156250	0	0
6	187500	0	0
6.144	192000	0	0
7.3728	230400	0	0
8	250000	0	0
9.8304	307200	0	0
10	312500	0	0
12	375000	0	0
12.288	384000	0	0
14	437500	0	0
14.7456	460800	0	0
16	500000	0	0
17.2032	537600	0	0
18	562500	0	0
19.6608	614400	0	0
20	625000	0	0

**Table 14-6 Maximum Bit Rates with External Clock Input (Asynchronous Mode)**

<b>ø (MHz)</b>	<b>External Clock Input (MHz)</b>	<b>Maximum Bit Rate (Bits/s)</b>
1	0.2500	15625
1.2288	0.3072	19200
2	0.5000	31250
2.097152	0.5243	32768
2.4576	0.6144	38400
3	0.7500	46875
3.6864	0.9216	57600
4	1.0000	62500
4.9152	1.2288	76800
5	1.2500	78125
6	1.5000	93750
6.144	1.5360	96000
7.3728	1.8432	115200
8	2.0000	125000
9.8304	2.4576	153600
10	2.5000	156250
12	3.0000	187500
12.288	3.0720	192000
14	3.5000	218750
14.7456	3.6864	230400
16	4.0000	250000
17.2032	4.3008	268800
18	4.5000	281250
19.6608	4.9152	307200
20	5.0000	312500

Table 14-7 shows examples of settings in clocked synchronous mode.

**Table 14-7 Examples of Bit Rates and BRR Settings in Synchronous Mode**

Bit Rate (Bits/s)	$\varnothing$ (MHz)											
	1		2		4		8		10		16	
	n	N	n	N	n	N	n	N	n	N	n	N
110	—	—	3	70	—	—	—	—	—	—	—	—
250	1	249	2	124	2	249	3	124	—	—	3	249
500	1	124	1	249	2	124	2	249	—	—	3	124
1 k	0	249	1	124	1	249	1	124	—	—	2	249
2.5 k	0	99	0	199	1	99	1	199	1	249	2	99
5 k	0	49	0	99	0	199	1	99	1	124	1	199
10 k	0	24	0	49	0	99	0	199	0	249	1	99
25 k	0	9	0	19	0	39	0	79	0	99	0	159
50 k	0	4	0	9	0	19	0	39	0	49	0	79
100 k	—	—	0	4	0	9	0	19	0	24	0	39
250 k	0	0*	0	1	0	3	0	7	0	9	0	15
500 k			0	0*	0	1	0	3	0	4	0	7
1 M					0	0*	0	1	—	—	0	3
2.5 M							—	—	0	0*	—	—

Blank: No setting available

—: Setting possible, but error occurs

\* : Continuous transmit/receive not possible

Note: The BRR setting is calculated as follows:

$$N = \lceil \varnothing / (8 \times 2^{2n-1} \times B) \rceil \times 10^6 - 1$$

B: Bit rate

N: BRR setting for baud rate generator ( $0 \leq N \leq 255$ )

$\varnothing$ : Operation frequency (MHz)

n: Baud rate generator clock source ( $n = 0, 1, 2, 3$ )

(For the clock sources and values of n, see table 14-8.)

**Table 14-8 Clock Sources and n**

n	Clock Source	SMR Settings	
		CKS1	CKS0
0	$\varnothing$	0	0
1	$\varnothing/4$	0	1
2	$\varnothing/16$	1	0
3	$\varnothing/64$	1	1

## 14.3 Operation

### 14.3.1 Overview

The SCI has an asynchronous mode in which characters are synchronized individually, and a clocked synchronous mode in which communication is synchronized with clock pulses. Serial communication is possible in either mode. Asynchronous or clocked synchronous mode and the communication format are selected in the serial mode register (SMR), as shown in table 14-9. The SCI clock source is selected by the  $\overline{C/A}$  bit in the serial mode register (SMR) and the CKE1 and CKE0 bits in the serial control register (SCR), as shown in table 14-10.

#### (1) Asynchronous Mode

- Data length is selectable: seven or eight bits.
- Parity and multiprocessor bits are selectable. So is the stop bit length (one or two bits). The foregoing selections constitute the communication format (The combination of these items determines the communication format and the character length.).
- In receiving, it is possible to detect framing errors (FER), parity errors (PER), overrun errors (ORER), and the break state.
- An internal or external clock can be selected as the SCI clock source.
  - When an internal clock is selected, the SCI operates using the built-in baud rate generator, and can output a serial clock signal with a frequency matching the bit rate.
  - When an external clock is selected, the external clock input must have a frequency 16 times the bit rate. (The built-in baud rate generator is not used.)

#### (2) Clocked Synchronous Mode

- The communication format has a fixed eight-bit data length.
- In receiving, it is possible to detect overrun errors (ORER).
- An internal or external clock can be selected as the SCI clock source.
  - When an internal clock is selected, the SCI operates using the built-in baud rate generator, and outputs a serial clock signal to external devices.
  - When an external clock is selected, the SCI operates on the input serial clock. The built-in baud rate generator is not used.

**Table 14-9 Serial Mode Register Settings and SCI Communication Formats**

SMR Settings					SCI Communication Format							
Bit 7	Bit 6	Bit 5	Bit 2	Bit 3	Mode	Data Length	Parity Bit	Multi-processor Bit	Stop Bit Length			
C/A	CHR	PE	MP	STOP								
0	0	0	0	0	Asynchronous mode	8-bit data	Absent	Absent	1 bit			
				1					2 bits			
		1		0					1 bit			
				1					2 bits			
1	0			0					7-bit data	Absent		1 bit
				1					2 bits			
				1			Present	1 bit				
				1				2 bits				
0	*	1	0	0	Asynchronous mode (multiprocessor format)	8-bit data	Absent	Present	1 bit			
	*			1					2 bits			
1	*			0					7-bit data			1 bit
	*			1					2 bits			
1	*	*	*	*	Clocked synchronous mode	8-bit data		Absent	None			

Note: Asterisks (\*) in the table indicate don't-care bits.

**Table 14-10 SMR and SCR Settings and SCI Clock Source Selection**

SMR			SCR Settings		SCI Transmit/Receive Clock	
Bit 7	Bit 1	Bit 0	Mode	Clock Source	SCK Pin Function	
C/A	CKE1	CKE0				
0	0	0	Asynchronous mode	Internal	General-purpose input/output (SCI does not use the SCK pin)	
		1			Outputs a clock with frequency matching the bit rate	
		0			External	Inputs a clock with frequency 16 times the bit rate
1						
1	0	0	Clocked synchronous mode	Internal	Outputs the serial clock	
		1			External	Inputs the serial clock
		0				
		1				

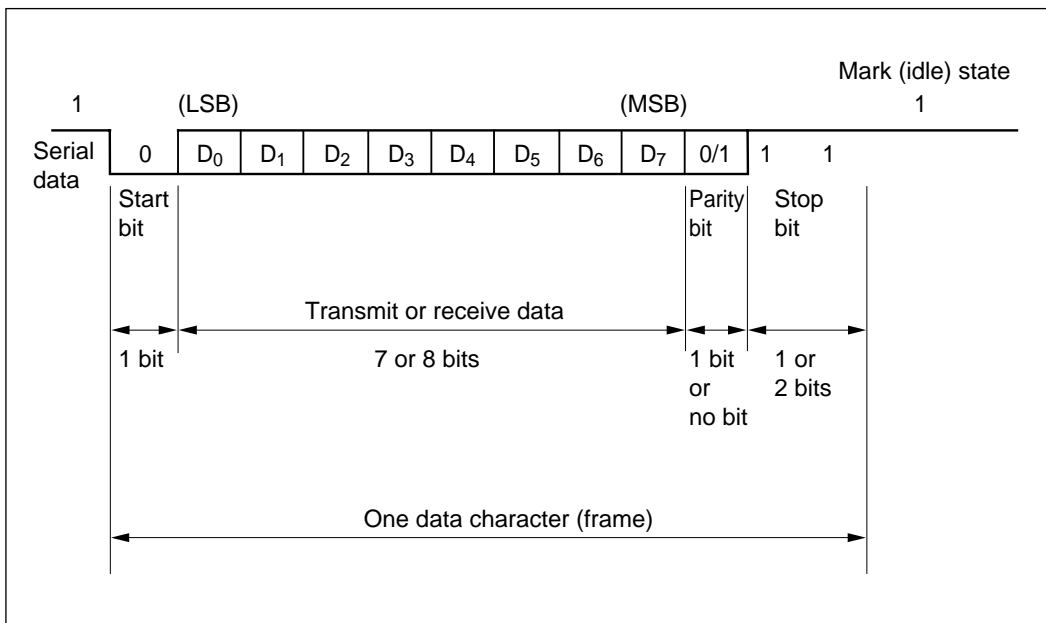
### 14.3.2 Operation in Asynchronous Mode

In asynchronous mode each transmitted or received character begins with a start bit and ends with a stop bit. Serial communication is synchronized one character at a time.

The transmitting and receiving sections of the SCI are independent, so full duplex communication is possible. The transmitter and receiver are both double buffered, so data can be written and read while transmitting and receiving are in progress, enabling continuous transmitting and receiving.

Figure 14-2 shows the general format of asynchronous serial communication. In asynchronous serial communication the communication line is normally held in the mark (high) state. The SCI monitors the line and starts serial communication when the line goes to the space (low) state, indicating a start bit. One serial character consists of a start bit (low), data (LSB first), parity bit (high or low), and stop bit (high), in that order.

When receiving in asynchronous mode, the SCI synchronizes on the falling edge of the start bit. The SCI samples each data bit on the eighth pulse of a clock with a frequency 16 times the bit rate. Receive data is latched at the center of each bit.



**Figure 14-2 Data Format in Asynchronous Communication  
(Example: 8-Bit Data with Parity and Two Stop Bits)**

**(1) Transmit/Receive Formats:** Table 14-11 shows the 12 communication formats that can be selected in asynchronous mode. The format is selected by settings in the serial mode register (SMR).

**Table 14-11 Serial Communication Formats (Asynchronous Mode)**

SMR Settings				Serial Communication Format and Frame Length											
CHR	PE	MP	STOP	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	S				8-bit data						STOP	
0	0	0	1	S				8-bit data					STOP	STOP	
0	1	0	0	S				8-bit data					P	STOP	
0	1	0	1	S				8-bit data					P	STOP	STOP
1	0	0	0	S				7-bit data						STOP	
1	0	0	1	S				7-bit data					STOP	STOP	
1	1	0	0	S				7-bit data					P	STOP	
1	1	0	1	S				7-bit data					P	STOP	STOP
0	*	1	0	S				8-bit data					MPB	STOP	
0	*	1	1	S				8-bit data					MPB	STOP	STOP
1	*	1	0	S				7-bit data					MPB	STOP	
1	*	1	1	S				7-bit data					MPB	STOP	STOP

SMR: serial mode register      P: parity bit  
 S: start bit                      MPB: multiprocessor bit  
 STOP: stop bit

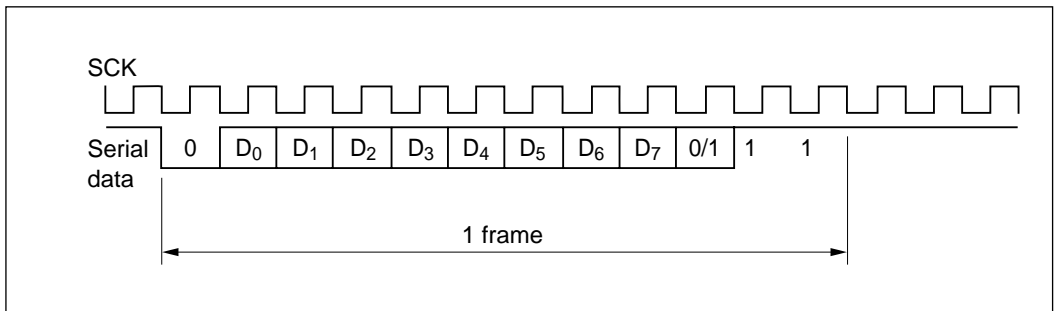
Note: Asterisks (\*) in the table indicate don't-care bits.



**(2) Clock:** An internal clock generated by the on-chip baud rate generator or an external clock input from the SCK pin can be selected as the SCI transmit/receive clock. The clock source is selected by the  $\overline{C/A}$  bit in the serial mode register (SMR) and bits CKE1 and CKE0 in the serial control register (SCR). See table 14-10.

When an external clock is input at the SCK pin, it must have a frequency equal to 16 times the desired bit rate.

When the SCI operates on an internal clock, it can output a clock signal at the SCK pin. The frequency of this output clock is equal to the bit rate. The phase is aligned as in figure 14-3 so that the rising edge of the clock occurs at the center of each transmit data bit.



**Figure 14-3 Phase Relationship between Output Clock and Serial Data (Asynchronous Mode)**

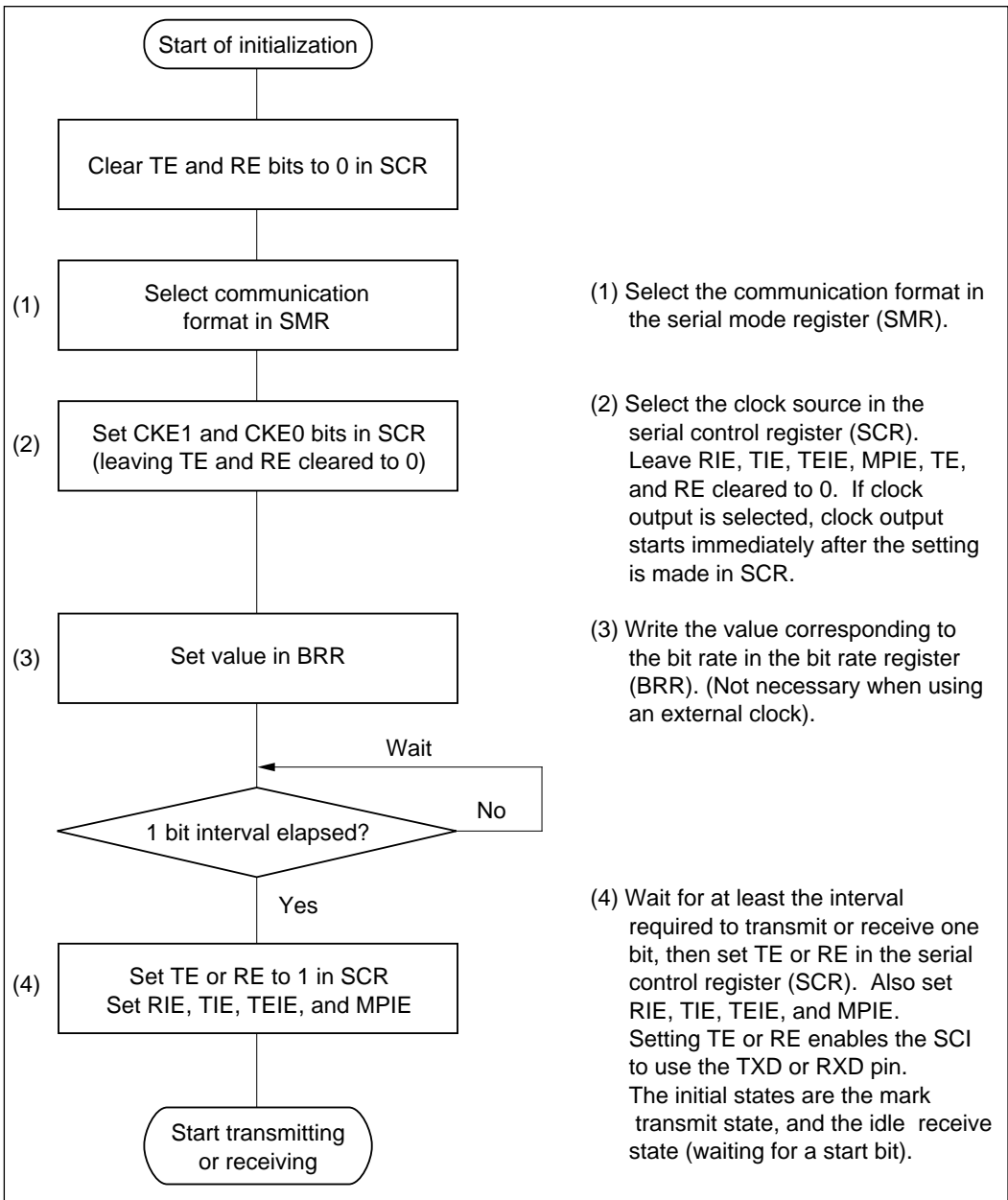
### **(3) Transmitting and Receiving Data**

**SCI Initialization (Asynchronous Mode):** Before transmitting or receiving, software must clear the TE and RE bits to 0 in the serial control register (SCR), then initialize the SCI as follows.

When changing the communication mode or format, always clear the TE and RE bits to 0 before following the procedure given below. Clearing TE to 0 sets TDRE to 1 and initializes the transmit shift register (TSR). Clearing RE to 0, however, does not initialize the RDRF, PER, FER, and OER flags and receive data register (RDR), which retain their previous contents.

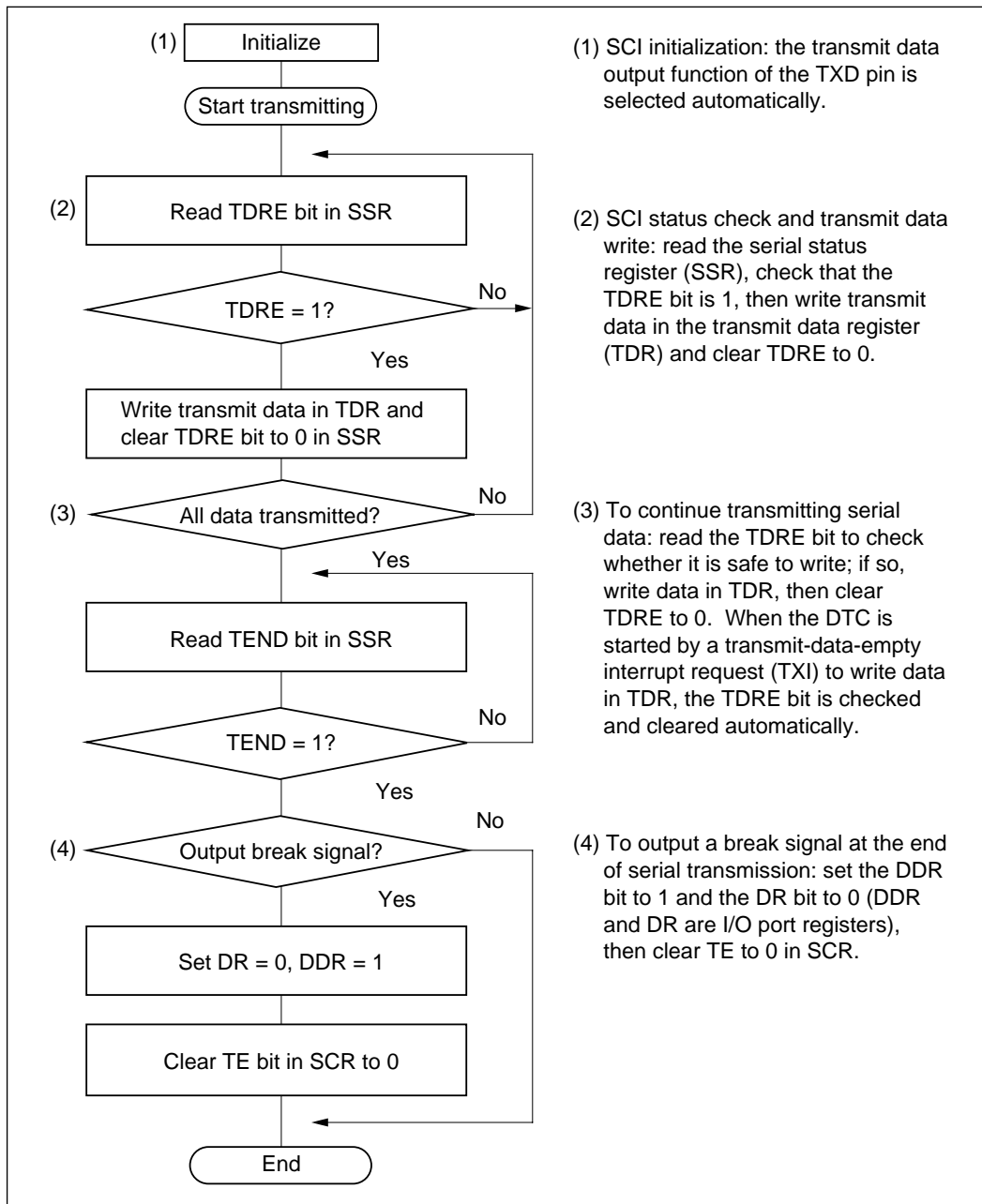
When an external clock is used, the clock should not be stopped during initialization or subsequent operation. SCI operation becomes unreliable if the clock is stopped.

Figure 14-4 is a sample flowchart for initializing the SCI.



**Figure 14-4 Sample Flowchart for SCI Initialization**

**Transmitting Serial Data (Asynchronous Mode):** Figure 14-5 shows a sample flowchart for transmitting serial data and indicates the procedure to follow.



**Figure 14-5 Sample Flowchart for Transmitting Serial Data**

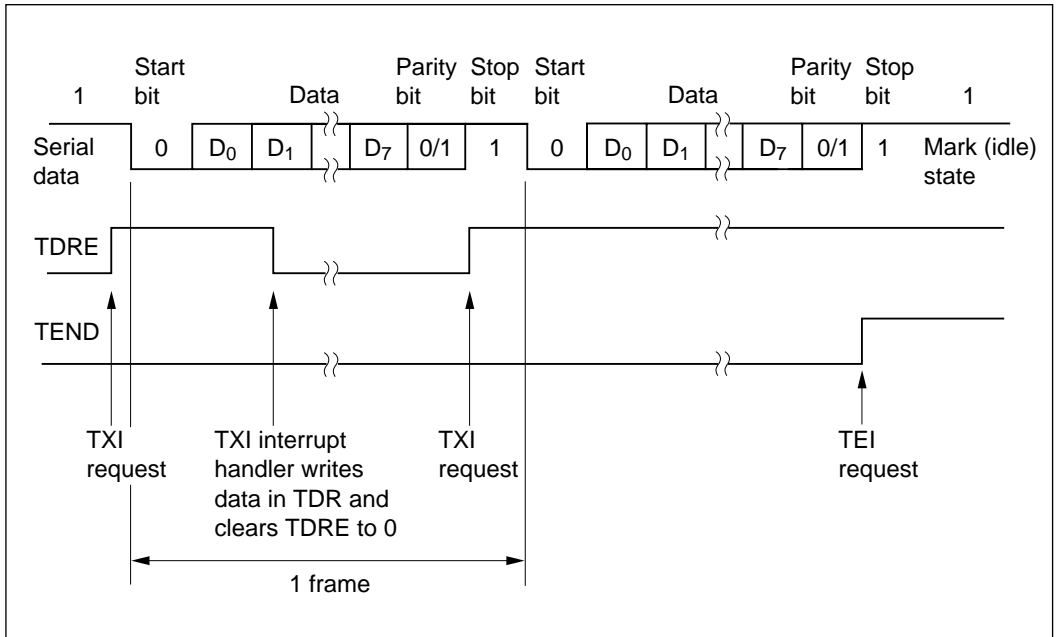
In transmitting serial data, the SCI operates as follows.

1. The SCI monitors the TDRE bit in SSR. When TDRE is cleared to 0 the SCI recognizes that the transmit data register (TDR) contains new data, and loads this data from TDR into the transmit shift register (TSR).
2. After loading the data from TDR into TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) is set to 1 in SCR, the SCI requests a transmit-data-empty interrupt (TXI) at this time.

Serial transmit data is transmitted in the following order from the TXD pin:

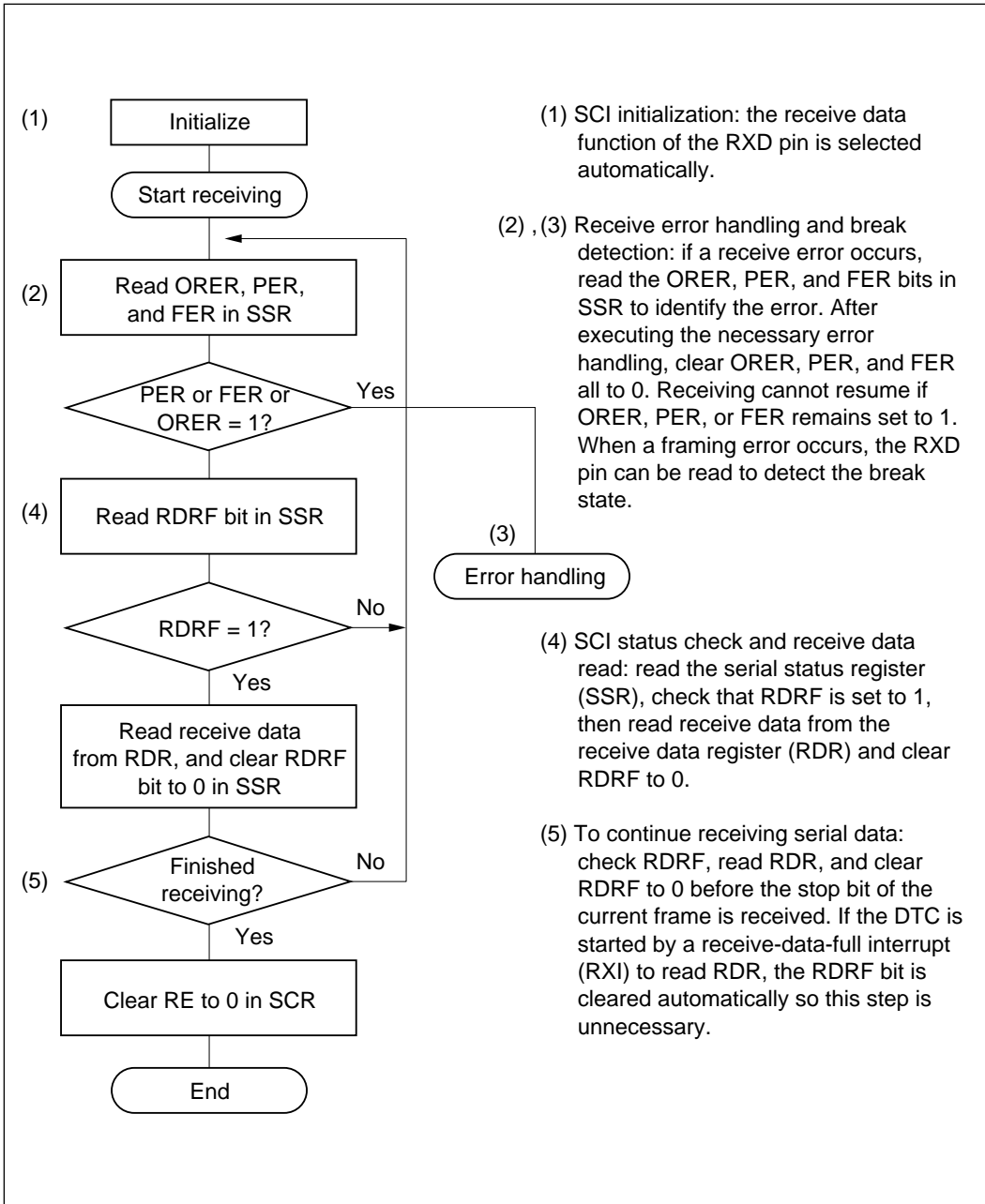
- a. Start bit: one 0 bit is output.
  - b. Transmit data: seven or eight bits are output, LSB first.
  - c. Parity bit or multiprocessor bit: one parity bit (even or odd parity) or one multiprocessor bit is output. Formats in which neither a parity bit nor a multiprocessor bit is output can also be selected.
  - d. Stop bit: one or two 1 bits (stop bits) are output.
  - e. Mark state: output of 1 bits continues until the start bit of the next transmit data.
3. The SCI checks the TDRE bit when it outputs the stop bit. If TDRE is 0, the SCI loads new data from TDR into TSR, outputs the stop bit, then begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit to 1 in SSR, outputs the stop bit, then continues output of 1 bits in the mark state. If the transmit-end interrupt enable bit (TEIE) in SCR is set to 1, a transmit-end interrupt (TEI) is requested.

Figure 14-6 shows an example of SCI transmit operation in asynchronous mode.

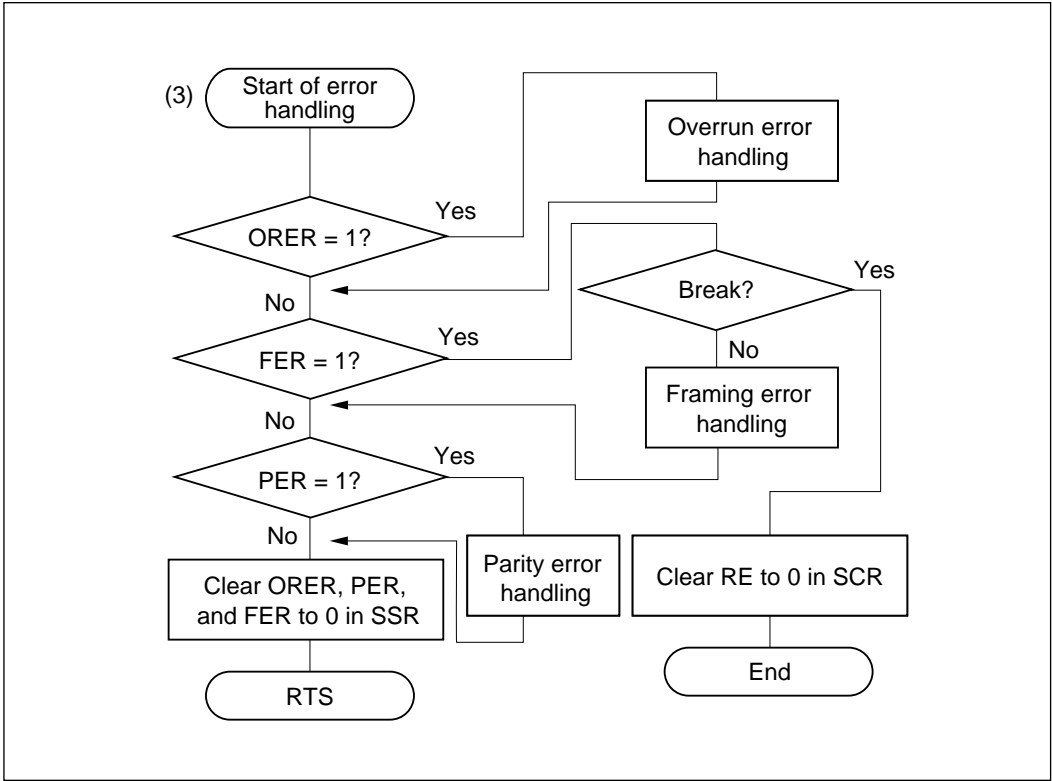


**Figure 14-6 Example of SCI Transmit Operation  
(8-Bit Data with Parity and One Stop Bit)**

**Receiving Serial Data (Asynchronous Mode):** Figure 14-7 shows a sample flowchart for receiving serial data and indicates the procedure to follow.



**Figure 14-7 Sample Flowchart for Receiving Serial Data  
(Continued on Next Page)**



**Figure 14-7 Sample Flowchart for Receiving Serial Data (cont)**

In receiving, the SCI operates as follows.

1. The SCI monitors the receive data line. When it detects a start bit 0, the SCI synchronizes internally and starts receiving.
2. Receive data is shifted into RSR in order from LSB to MSB.
3. The parity bit and stop bit are received.

After receiving these bits, the SCI makes the following checks:

- a. Parity check: the number of 1s in the receive data must match the even or odd parity setting of the  $O/\bar{E}$  bit in SMR.
- b. Stop bit check: the stop bit value must be 1. If there are two stop bits, only the first stop bit is checked.
- c. Status check: RDRF must be 0 so that receive data can be loaded from RSR into RDR.

If these checks all pass, the SCI sets RDRF to 1 and stores the received data in RDR. If one of the checks fails (receive error), the SCI operates as indicated in table 14-12.

Note: When a receive error flag is set, further receiving is disabled. When receiving resumes after an error flag was set, the RDRF bit is not set to 1.

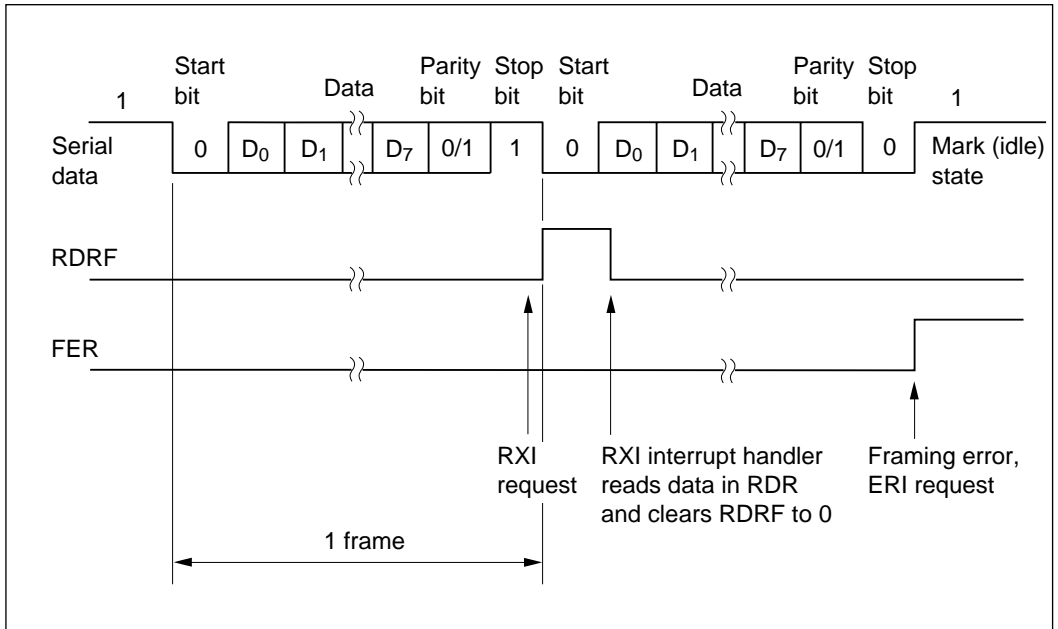
4. After setting RDRF to 1, if the receive-data-full interrupt enable bit (RIE) is set to 1 in SCR, the SCI requests a receive-data-full interrupt (RXI). If one of the error flags (ORER, PER, or FER) is set to 1 and the receive-data-full interrupt enable bit (RIE) in SCR is also set to 1, the SCI requests a receive-error interrupt (ERI).

Figure 14-8 shows an example of SCI receive operation in asynchronous mode.

**Table 14-12 Receive Error Conditions and SCI Operation**

<b>Receive Error</b>	<b>Abbreviation</b>	<b>Condition</b>	<b>Data Transfer</b>
Overrun error	ORER	Receiving of next data ends while RDRF is still set to 1 in SSR	Receive data not loaded from RSR into RDR
Framing error	FER	Stop bit is 0	Receive data loaded from RSR into RDR
Parity error	PER	Parity of receive data differs from even/odd parity setting in SMR	Receive data loaded from RSR into RDR





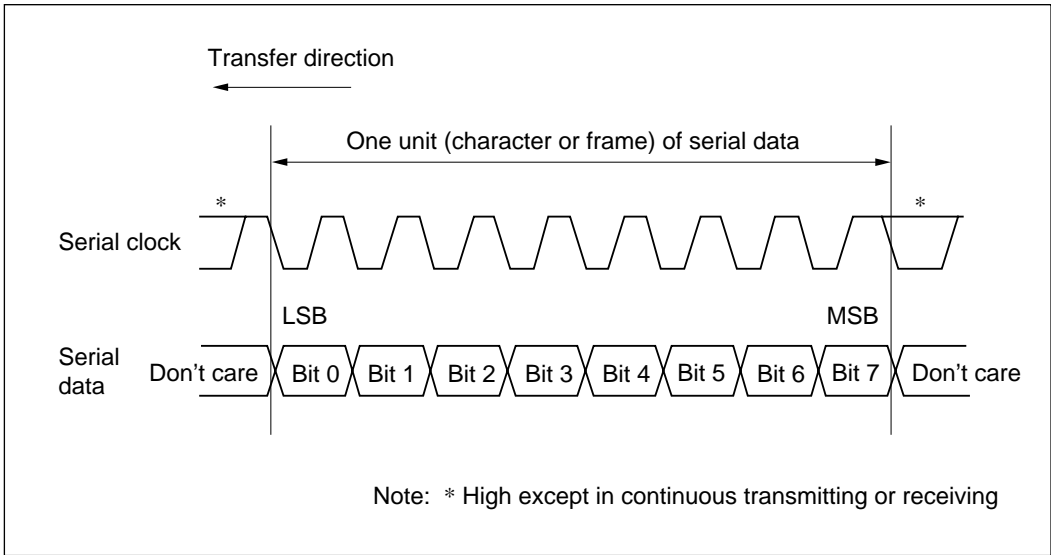
**Figure 14-8 Example of SCI Receive Operation  
(8-Bit Data with Parity and One Stop Bit)**

### 14.3.3 Clocked Synchronous Operation

In clocked synchronous mode, the SCI transmits and receives data in synchronization with clock pulses. This mode is suitable for high-speed serial communication.

The SCI transmitter and receiver share the same clock but are otherwise independent, so full duplex communication is possible. The transmitter and receiver are also double buffered, so continuous transmitting or receiving is possible by reading or writing data while transmitting or receiving is in progress.

Figure 14-9 shows the general format in clocked synchronous serial communication.



**Figure 14-9 Data Format in Clocked Synchronous Communication**

In clocked synchronous serial communication, each data bit is placed on the communication line from one falling edge of the serial clock to the next. Data is guaranteed valid at the rising edge of the serial clock. In each character, the serial data bits are transmitted in order from LSB (first) to MSB (last). After output of the MSB, the communication line remains in the state of the MSB. In clocked synchronous mode the SCI receives data by synchronizing with the rising edge of the serial clock.

**(1) Communication Format:** The data length is fixed at eight bits. No parity bit or multiprocessor bit can be added.

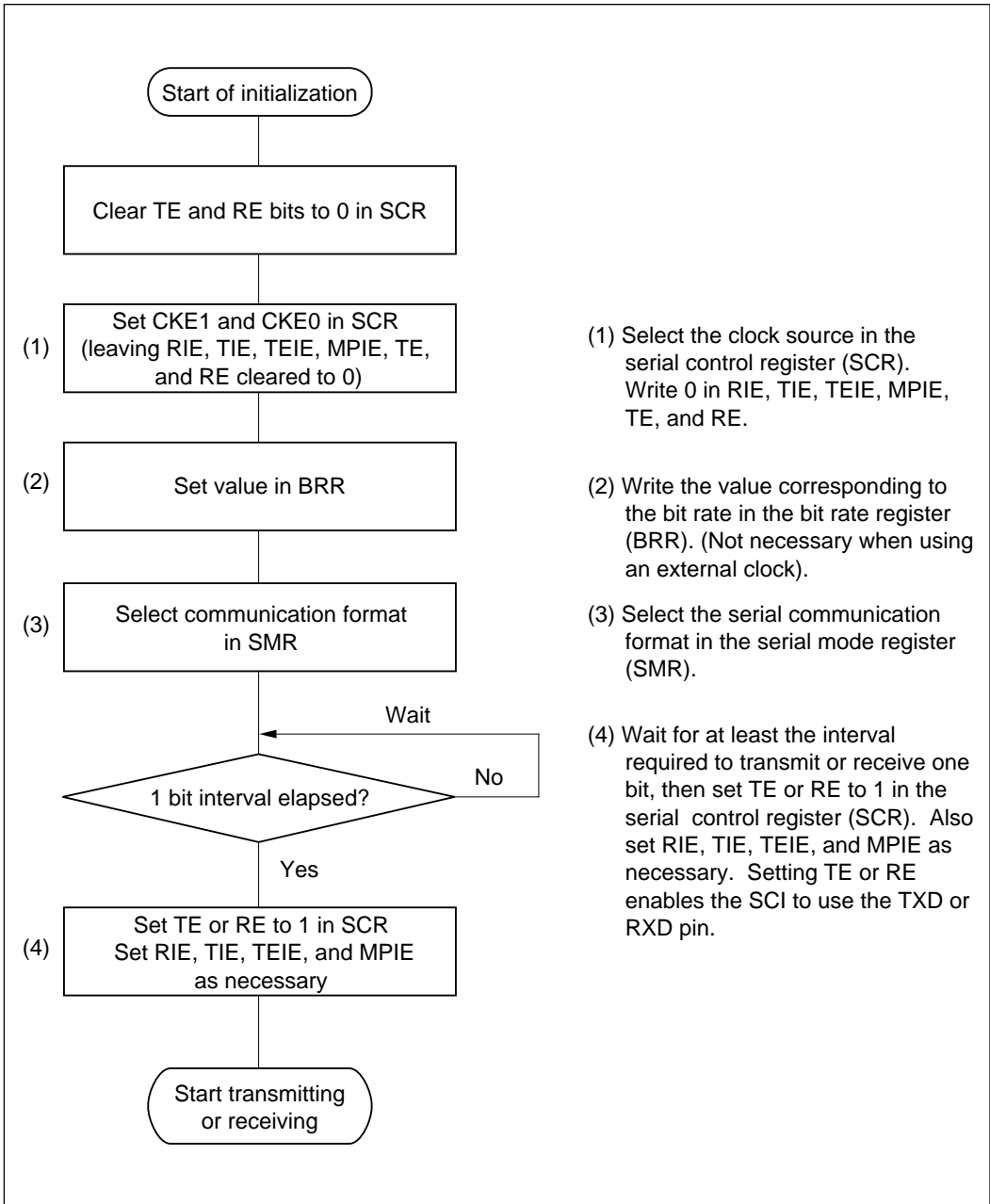
**(2) Clock:** An internal clock generated by the on-chip baud rate generator or an external clock input from the SCK pin can be selected according to the setting of the  $C/\bar{A}$  bit in the serial mode register (SMR) and the CKE1 and CKE0 bits in the serial control register (SCR). See table 14-9. When the SCI operates on an internal clock, it outputs the clock signal at the SCK pin. Eight clock pulses are output per transmitted or received character. When the SCI is not transmitting or receiving, the clock signal remains in the high state.

### (3) Transmitting and Receiving Data

**SCI Initialization (Clocked Synchronous Mode):** Before transmitting or receiving, software must clear the TE and RE bits to 0 in the serial control register (SCR), then initialize the SCI as follows.

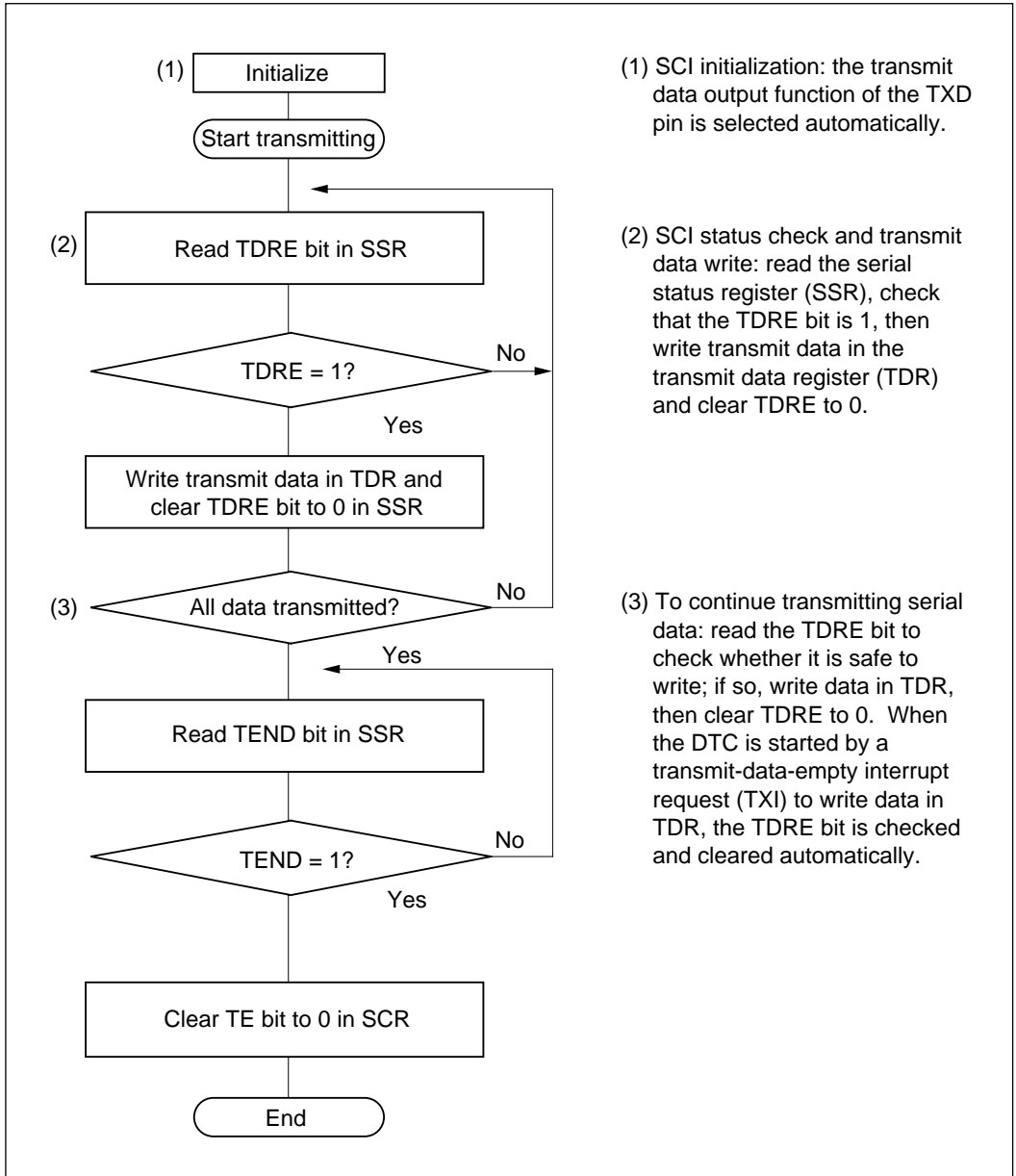
When changing the communication mode or format, always clear the TE and RE bits to 0 before following the procedure given below. Clearing TE to 0 sets TDRE to 1 and initializes the transmit shift register (TSR). Clearing RE to 0, however, does not initialize the RDRF, PER, FER, and ORER flags and receive data register (RDR), which retain their previous contents.

Figure 14-10 is a sample flowchart for initializing the SCI.



**Figure 14-10 Sample Flowchart for SCI Initialization**

**Transmitting Serial Data (Clocked Synchronous Mode):** Figure 14-11 shows a sample flowchart for transmitting serial data and indicates the procedure to follow.



**Figure 14-11 Sample Flowchart for Serial Transmitting**

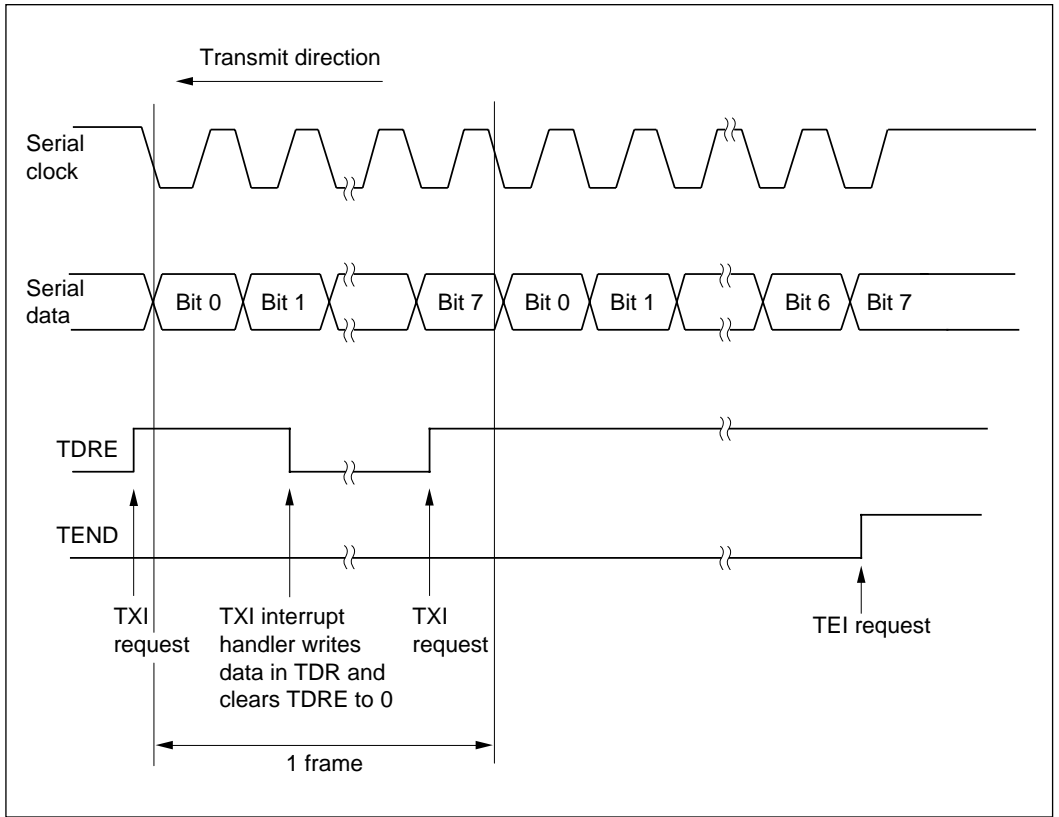
In transmitting serial data, the SCI operates as follows.

1. The SCI monitors the TDRE bit in SSR. When TDRE is cleared to 0 the SCI recognizes that the transmit data register (TDR) contains new data, and loads this data from TDR into the transmit shift register (TSR).
2. After loading the data from TDR into TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) in SCR is set to 1, the SCI requests a transmit-data-empty interrupt (TXI) at this time.

If clock output is selected, the SCI outputs eight serial clock pulses. If an external clock source is selected, the SCI outputs data in synchronization with the input clock. Data is output from the TXD pin in order from LSB (bit 0) to MSB (bit 7).

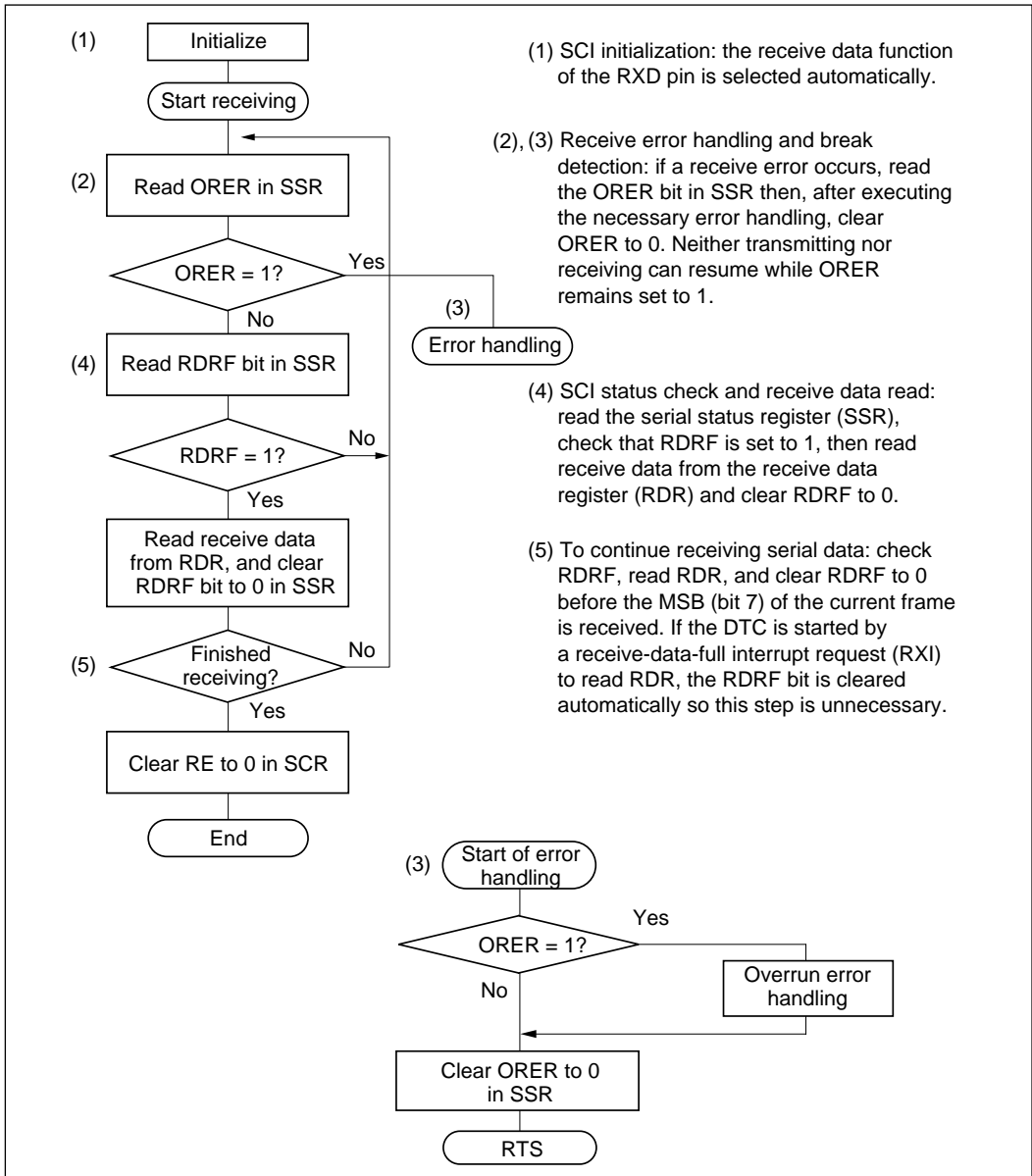
3. The SCI checks the TDRE bit when it outputs the MSB (bit 7). If TDRE is 0, the SCI loads data from TDR into TSR and begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit in SSR to 1, and after transmitting the MSB, holds the transmit data pin (TXD) in the MSB state. If the transmit-end interrupt enable bit (TEIE) in SCR is set to 1, a transmit-end interrupt (TEI) is requested at this time.
4. After the end of serial transmission, the SCK pin is held in the high state.

Figure 14-12 shows an example of SCI transmit operation.



**Figure 14-12 Example of SCI Transmit Operation**

**Receiving Serial Data (Clocked Synchronous Mode):** Figure 14-13 shows a sample flowchart for receiving serial data and indicates the procedure to follow. When switching from asynchronous mode to clocked synchronous mode, make sure that ORER, PER, and FER are cleared to 0. If ORER, PER, or FER is set to 1 the RDRF bit will not be set and both transmitting and receiving will be disabled.



**Figure 14-13 Sample Flowchart for Serial Receiving**



In receiving, the SCI operates as follows.

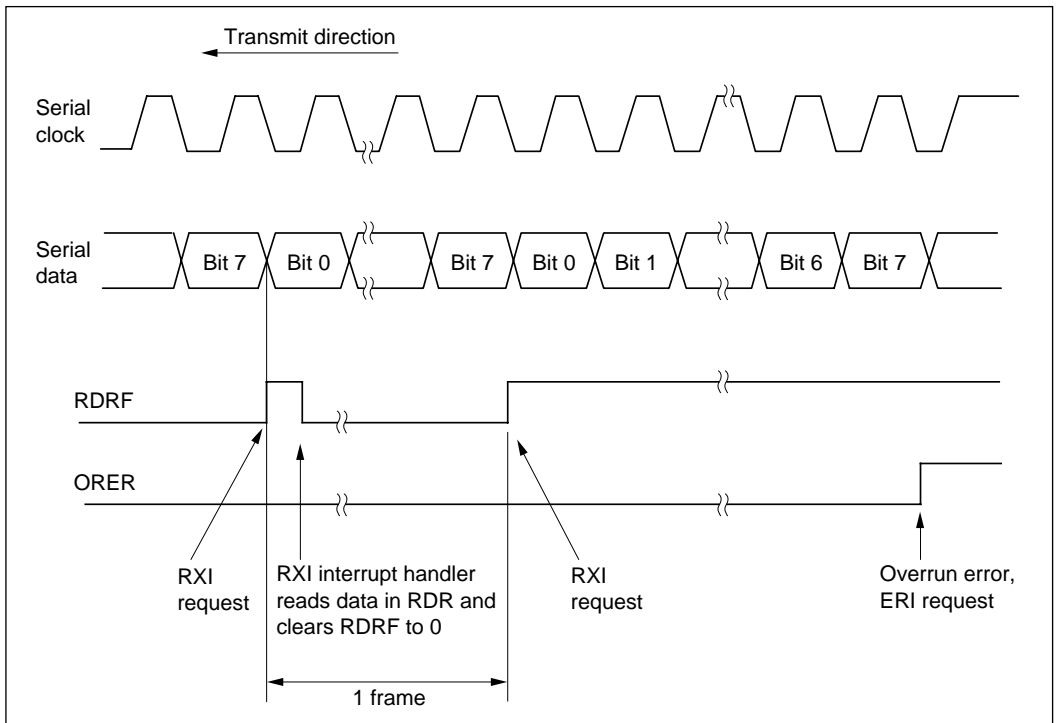
1. The SCI synchronizes with serial clock input or output and initializes internally.
2. Receive data is shifted into RSR in order from LSB to MSB.

After receiving the data, the SCI checks that RDRF is 0 so that receive data can be loaded from RSR into RDR. If this check passes, the SCI sets RDRF to 1 and stores the received data in RDR. If the check does not pass (receive error), the SCI operates as indicated in table 14-12.

Note: Both transmitting and receiving are disabled while a receive error flag is set. The RDRF bit is not set to 1. Be sure to clear the error flag.

3. After setting RDRF to 1, if the receive-data-full interrupt enable bit (RIE) is set to 1 in SCR, the SCI requests a receive-data-full interrupt (RXI). If the ORER bit is set to 1 and the receive-data-full interrupt enable bit (RIE) in SCR is also set to 1, the SCI requests a receive-error interrupt (ERI).

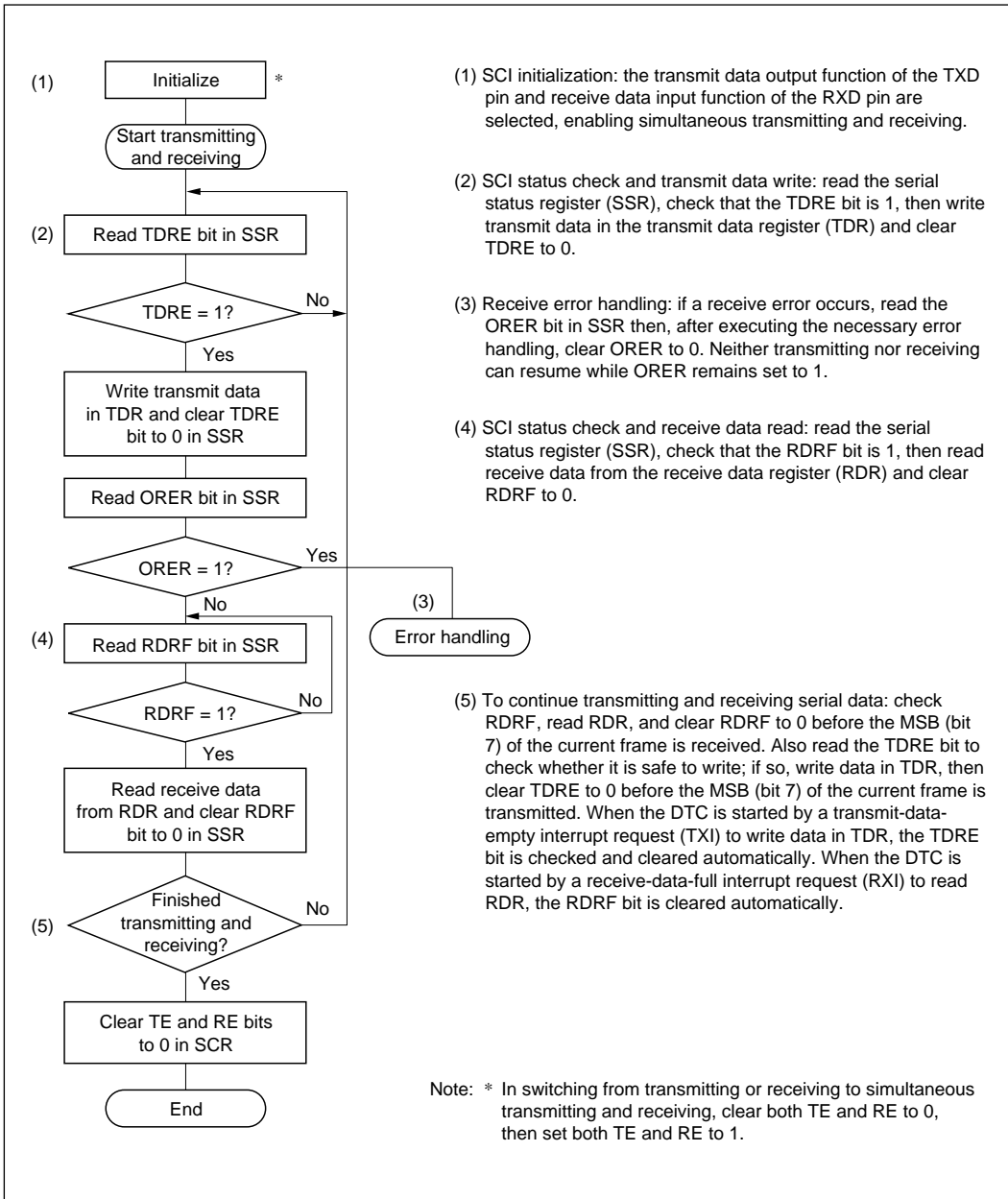
Figure 14-14 shows an example of SCI receive operation.



**Figure 14-14 Example of SCI Receive Operation**

## Transmitting and Receiving Serial Data Simultaneously (Clocked Synchronous Mode):

Figure 14-15 shows a sample flowchart for transmitting and receiving serial data simultaneously and indicates the procedure to follow.



**Figure 14-15 Sample Flowchart for Simultaneous Transmitting and Receiving**

### 14.3.4 Multiprocessor Communication

The multiprocessor communication function enables several processors to share a single serial communication line. The processors communicate in asynchronous mode using a format with an additional multiprocessor bit (multiprocessor format).

In multiprocessor communication, each receiving processor is addressed by an ID. A serial communication cycle consists of an ID-sending cycle that identifies the receiving processor, and a data-sending cycle. The multiprocessor bit distinguishes ID-sending cycles from data-sending cycles.

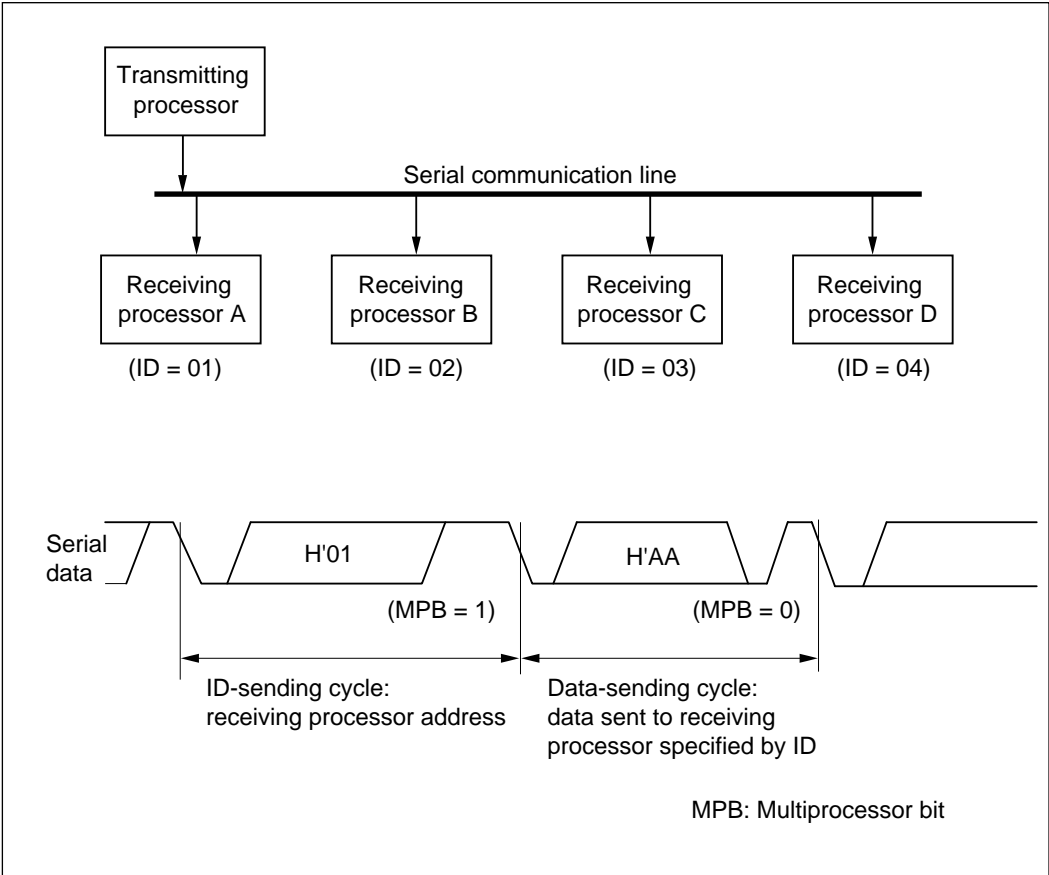
The transmitting processor should start by sending the ID of the receiving processor with which it wants to communicate as data with the multiprocessor bit set to 1. Next the transmitting processor should send transmit data with the multiprocessor bit cleared to 0.

When a receiving processor receives data with the multiprocessor bit set to 1, if multiprocessor interrupts are enabled, an interrupt is requested. The interrupt-handling routine should compare the data with the processor's own ID. If the ID matches, the processor should continue to receive data. If the ID does not match, the processor should skip further incoming data until it again receives data with the multiprocessor bit set to 1. Multiple processors can send and receive data in this way.

Figure 14-16 shows an example of communication among different processors using a multiprocessor format.

**(1) Communication Formats:** Four formats are available. Parity-bit settings are ignored when a multiprocessor format is selected. For details see table 14-9.

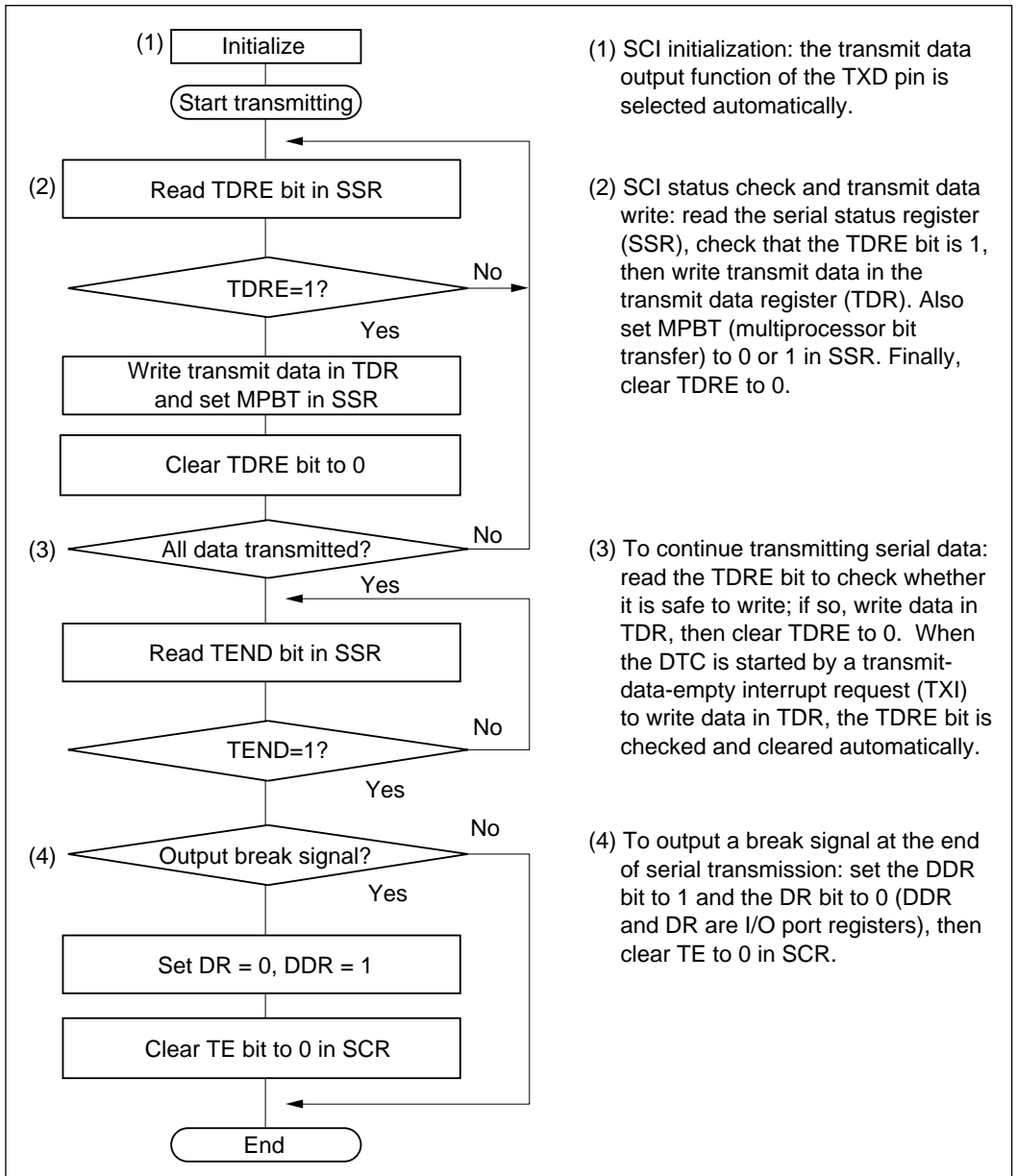
**(2) Clock:** See the description of asynchronous mode.



**Figure 14-16 Example of Communication among Processors using Multiprocessor Format (Sending Data H'AA to Receiving Processor A)**

### (3) Transmitting and Receiving Data

**Transmitting Multiprocessor Serial Data:** Figure 14-17 shows a sample flowchart for transmitting multiprocessor serial data and indicates the procedure to follow.



**Figure 14-17 Sample Flowchart for Transmitting Multiprocessor Serial Data**

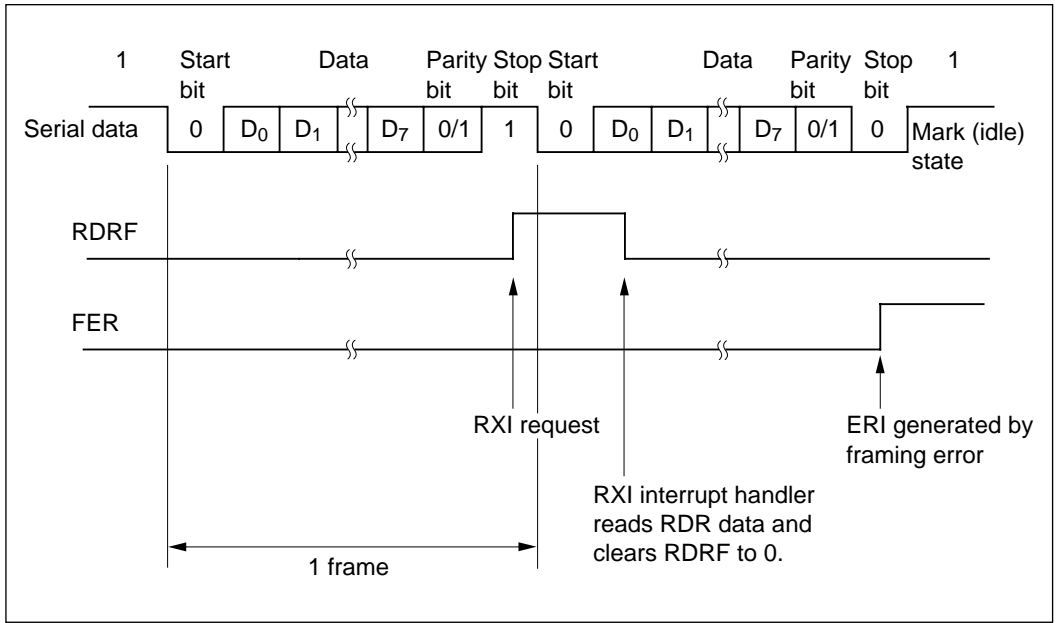
In transmitting serial data, the SCI operates as follows.

1. The SCI monitors the TDRE bit in SSR. When TDRE is cleared to 0 the SCI recognizes that the transmit data register (TDR) contains new data, and loads this data from TDR into the transmit shift register (TSR).
2. After loading the data from TDR into TSR, the SCI sets the TDRE bit to 1 and starts transmitting. If the transmit-data-empty interrupt enable bit (TIE) in SCR is set to 1, the SCI requests a transmit-data-empty interrupt (TXI) at this time.

Serial transmit data is transmitted in the following order from the TXD pin:

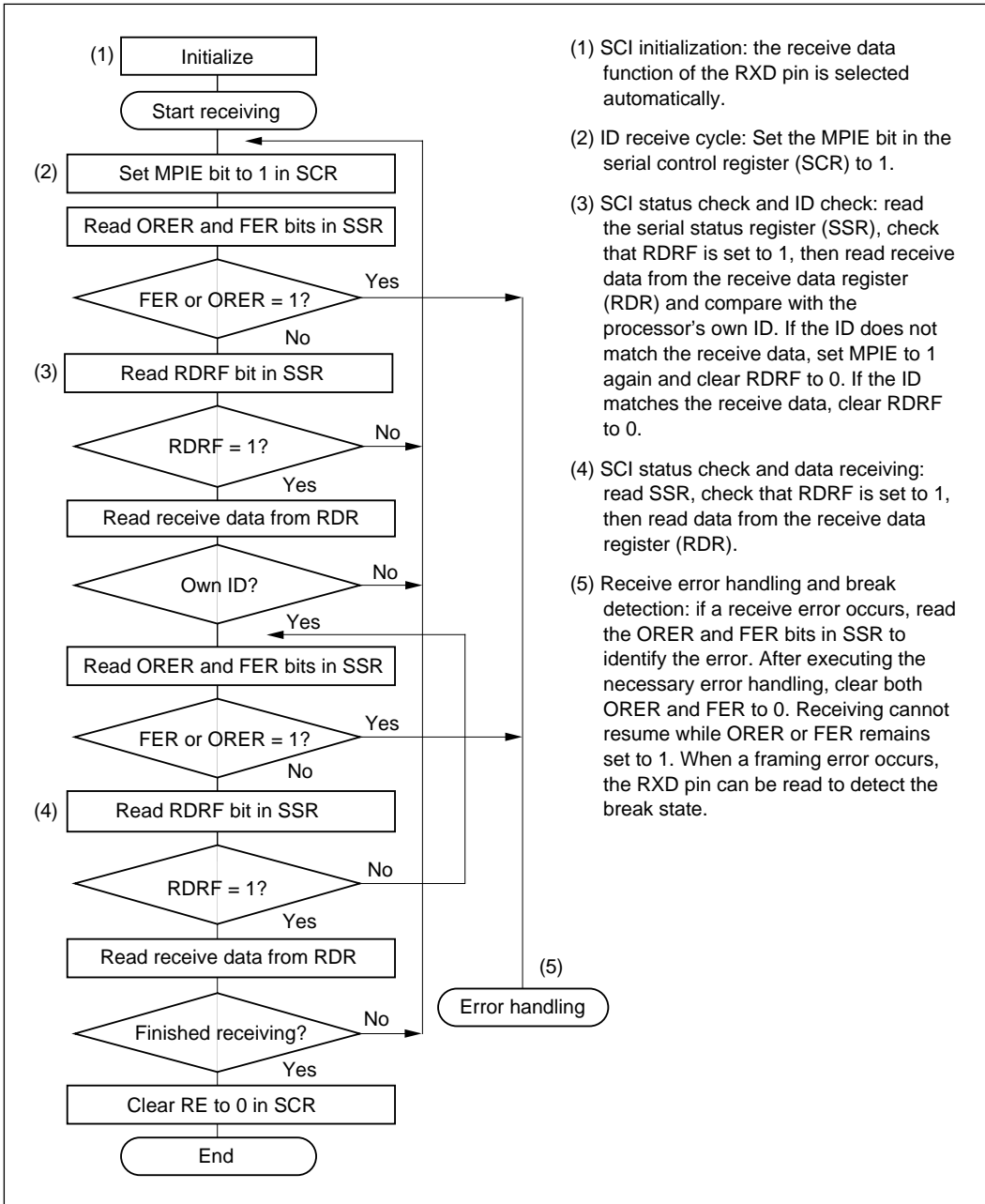
- a. Start bit: one 0 bit is output.
  - b. Transmit data: seven or eight bits are output, LSB first.
  - c. Multiprocessor bit: one multiprocessor bit (MPBT value) is output.
  - d. Stop bit: one or two 1 bits (stop bits) are output.
  - e. Mark state: output of 1 bits continues until the start bit of the next transmit data.
3. The SCI checks the TDRE bit when it outputs the stop bit. If TDRE is 0, the SCI loads data from TDR into TSR, outputs the stop bit, then begins serial transmission of the next frame. If TDRE is 1, the SCI sets the TEND bit in SSR to 1, outputs the stop bit, then continues output of 1 bits in the mark state. If the transmit-end interrupt enable bit (TEIE) in SCR is set to 1, a transmit-end interrupt (TEI) is requested at this time.

Figure 14-18 shows an example of SCI transmit operation using a multiprocessor format.



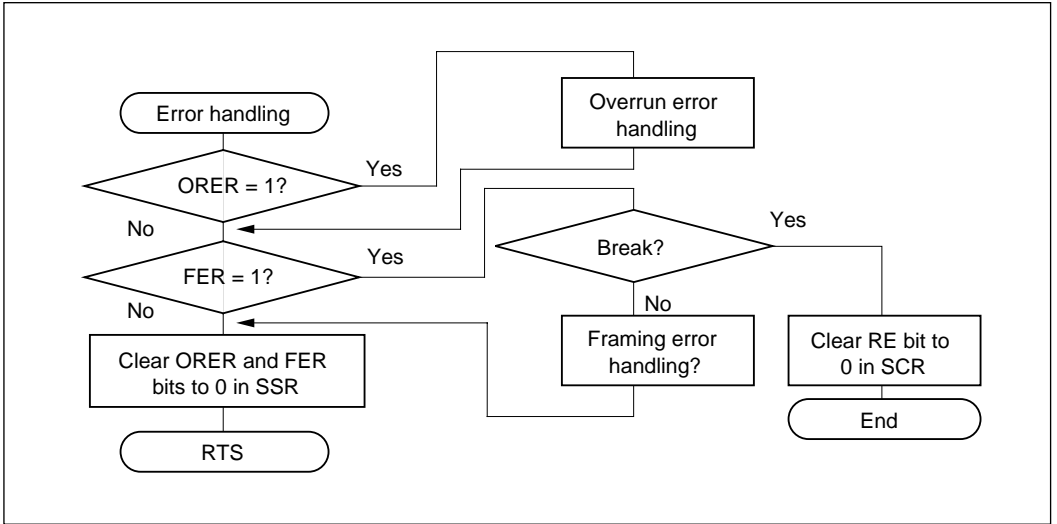
**Figure 14-18 Example of SCI Transmit Operation (8-Bit Data with Multiprocessor Bit and One Stop Bit)**

**Receiving Multiprocessor Serial Data:** Figure 14-19 shows a sample flowchart for receiving multiprocessor serial data and indicates the procedure to follow.



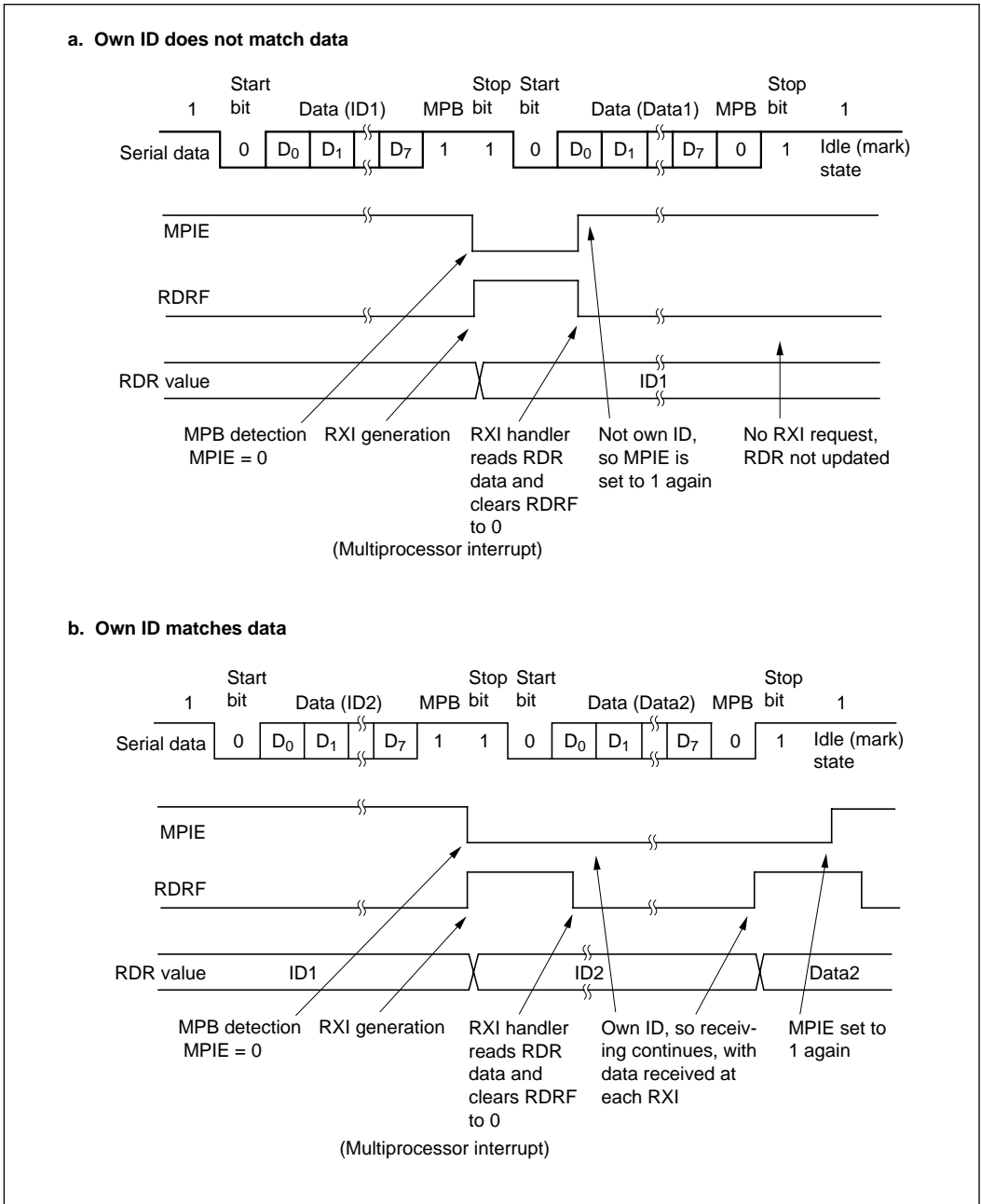
**Figure 14-19 Sample Flowchart for Receiving Multiprocessor Serial Data (Continued on Next Page)**





**Figure 14-19 Sample Flowchart for Receiving Multiprocessor Serial Data (cont)**

Figure 14-20 shows an example of SCI receive operation using a multiprocessor format.



**Figure 14-20 Example of SCI Receive Operation  
(Eight-Bit Data with Multiprocessor Bit and One Stop Bit)**

## 14.4 Interrupts and DTC

The SCI has four interrupt sources in each channel: transmit-end (TEI), receive-error (ERI), receive-data-full (RXI), and transmit-data-empty (TXI). Table 14-13 lists the interrupt sources and indicates their priority.

**Table 14-13 SCI Interrupt Sources**

Interrupt Source	Description	DTC Availability	Priority
ERI	Receive error (ORER, PER, or FER)	No	High
RXI	Receive data register full (RDRF)	Yes	↑ ↓
TXI	Transmit data register empty (TDRE)	Yes	
TEI	Transmit end (TEND)	No	Low

These interrupts can be enabled and disabled by the TIE, RIE bits in the serial control register (SCR). Each interrupt request is sent separately to the interrupt controller. TXI is requested when the TDRE bit in SSR is set to 1. TEI is requested when the TEND bit in SSR is set to 1. TXI can start the data transfer controller (DTC) to transfer data. TDRE is automatically cleared to 0 when the DTC executes the data transfer. TEI cannot start the DTC.

RXI is requested when the RDRF bit in SSR is set to 1. ERI is requested when the ORER, PER, or FER bit in SSR is set to 1. RXI can start the DTC to transfer data. RDRF is automatically cleared to 0 when the DTC executes the data transfer. ERI cannot start the DTC.

## 14.5 Usage Notes

Note the following points when using the SCI.

**(1) TDR Write and TDRE:** The TDRE bit in the serial status register (SSR) is a status flag indicating loading of transmit data from TDR into TSR. The SCI sets TDRE to 1 when it transfers data from TDR to TSR.

Data can be written into TDR regardless of the state of TDRE. If new data is written in TDR when TDRE is 0, the old data stored in TDR will be lost because this data has not yet been transferred to TSR. Before writing transmit data to TDR, be sure to check that TDRE is set to 1.

**(2) Simultaneous Multiple Receive Errors:** Table 14-14 indicates the state of SSR status flags when multiple receive errors occur simultaneously. When an overrun error occurs the RSR contents are not transferred to RDR, so receive data is lost.

**Table 14-14 SSR Status Flags and Transfer of Receive Data**

SSR Status Flags				Receive Data Transfer		Receive Errors
RDRF	ORER	FER	PER	RSR → RDR		
1	1	0	0	×		Overrun error
0	0	1	0	○		Framing error
0	0	0	1	○		Parity error
1	1	1	0	×		Overrun error + framing error
1	1	0	1	×		Overrun error + parity error
0	0	1	1	○		Framing error + parity error
1	1	1	1	×		Overrun error + framing error + parity error

○ : Receive data is transferred from RSR to RDR.

× : Receive data is not transferred from RSR to RDR.

**(3) Break Detection and Processing:** Break signals can be detected by reading the RXD pin directly when a framing error (FER) is detected. In the break state the input from the RXD pin consists of all 0s, so FER is set and the parity error flag (PER) may also be set. In the break state the SCI receiver continues to operate, so if the FER bit is cleared to 0 it will be set to 1 again.

**(4) Sending a Break Signal:** When TE is cleared to 0 the TXD pin becomes an I/O port, the level and direction (input or output) of which are determined by the DR and DDR bits. This feature can be used to send a break signal.

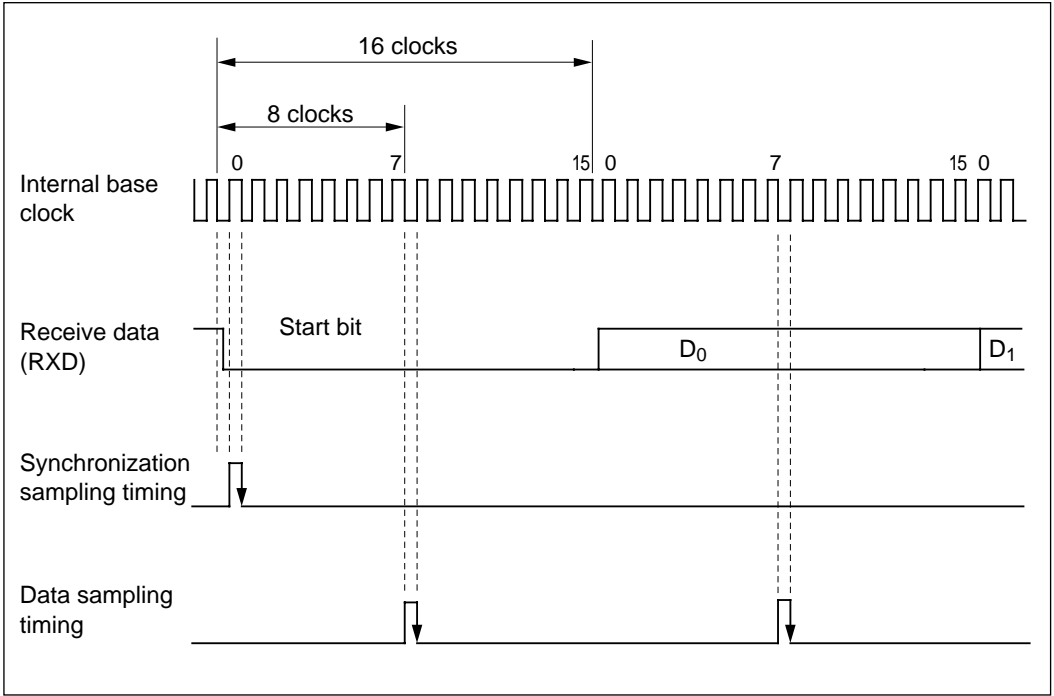
After the serial transmitter is initialized, the DR value substitutes for the mark state until TE is set to 1 (the TXD pin function is not selected until TE is set to 1). The DDR and DR bits should therefore both be set to 1 beforehand.

To send a break signal during serial transmission, clear the DR bit to 0, then clear TE to 0. When TE is cleared to 0 the transmitter is initialized, regardless of its current state, so the TXD pin becomes I/O port outputting the value 0.

**(5) Receive Error Flags and Transmitter Operation (Clocked Synchronous Mode Only):**

When a receive error flag (ORER, PER, or FER) is set to 1, the SCI will not start transmitting even if TDRE is set to 1. Be sure to clear the receive error flags to 0 when starting to transmit. Note that clearing RE to 0 does not clear the receive error flags.

**(6) Receive Data Sampling Timing in Asynchronous Mode and Receive Margin:** In asynchronous mode the SCI operates on an base clock with 16 times the bit rate frequency. In receiving, the SCI synchronizes internally with the falling edge of the start bit, which it samples on the base clock. Receive data is latched on the rising edge of the eighth base clock pulse. See figure 14-21.



**Figure 14-21 Receive Data Sampling Timing in Asynchronous Mode**

The receive margin in asynchronous mode can therefore be expressed as in equation (1).

$$M = \left\{ \left( 0.5 - \frac{1}{2N} \right) - \left( L - 0.5 - \frac{1}{2N} \right) F - \frac{|D - 0.5|}{N} (1 + F) \right\} \times 100\% \quad \dots\dots\dots (1)$$

- M: Receive margin (%)
- N: Ratio of clock frequency to bit rate (N = 16)
- D: Clock duty cycle (D = 0 to 1.0)
- L: Frame length (L = 9 to 12)
- F: Absolute deviation of clock frequency

From equation (1), if F = 0 and D = 0.5 the receive margin is 46.875%, as given by equation (2).

$$\begin{aligned}
 D &= 0.5, F = 0 \\
 M &= \left( 0.5 - \frac{1}{2 \times 16} \right) \times 100\% \\
 &= 46.875\% \quad \dots\dots\dots (2)
 \end{aligned}$$

This is a theoretical value. A reasonable margin to allow in system designs is 20 to 30%.

**(7) SCI Channel 3:** Use of pins for this channel must be enabled by setting bits 6, 5, and 3 in the port A control register (PACR).

# Section 15 A/D Converter

## 15.1 Overview

The H8/539F includes a 10-bit successive-approximations A/D converter. Software can select a maximum of 12 analog input channels.

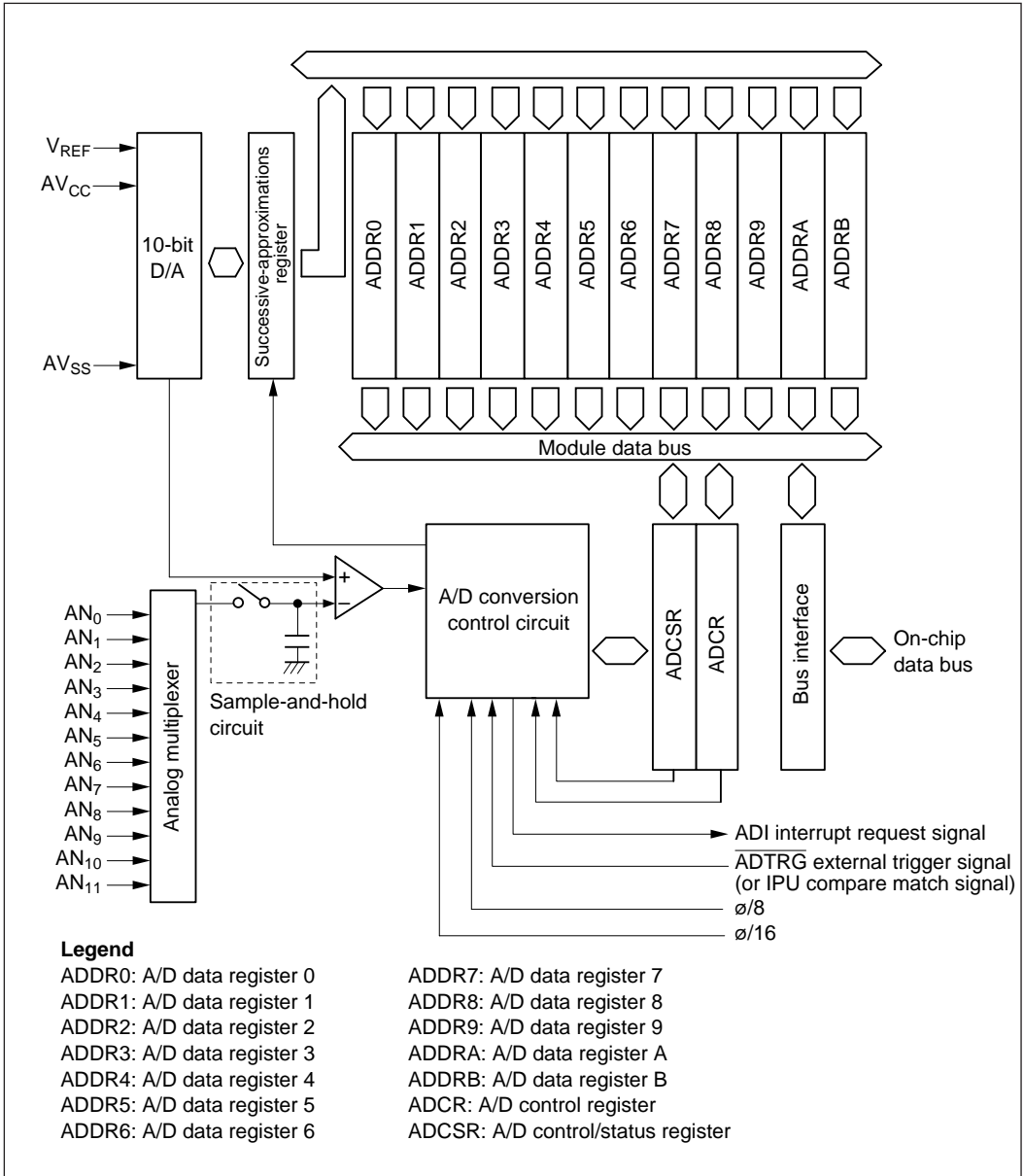
### 15.1.1 Features

A/D converter features are listed below.

- Ten-bit resolution  
Number of input channels: 12
- High-speed conversion  
Conversion time: minimum 8.3  $\mu$ s per channel ( $\phi$  = 16-MHz system clock)
- Two conversion modes  
Single mode: A/D conversion of one channel  
Scan mode: continuous conversion on one to 12 channels
- Twelve 10-bit A/D data registers  
A/D conversion results are transferred for storage into 12 A/D data registers. Each channel has its own A/D data register.
- Built-in sample-and-hold function  
A sample-and-hold circuit is built into the A/D converter, permitting a simplified external analog input circuit.
- A/D conversion interrupt with DTC (data transfer controller) support  
At the end of A/D conversion, an A/D end interrupt request (ADI) can be sent to the H8/500 CPU. The ADI interrupt can also be served by the DTC.
- External triggering  
A/D conversion can be started by an external trigger signal.
- Selectable analog conversion voltage range  
The analog voltage conversion range can be set by input at the  $V_{REF}$  pin.
- A/D conversion can also be started by the IPU.

## 15.1.2 Block Diagram

Figure 15-1 shows a block diagram of the A/D converter.



**Figure 15-1 A/D Converter Block Diagram**

### 15.1.3 Input/Output Pins

Table 15-1 summarizes the A/D converter's input pins. The 12 analog input pins (AN<sub>0</sub> to AN<sub>11</sub>) are divided into three groups: AN<sub>0</sub> to AN<sub>3</sub> (group 0), AN<sub>4</sub> to AN<sub>7</sub> (group 1), and AN<sub>8</sub> to AN<sub>11</sub> (group 2). The ADTRG pin can trigger the start of A/D conversion externally. The A/D converter starts A/D conversion when a low pulse is applied to this pin. AV<sub>CC</sub> and AV<sub>SS</sub> are the power supply for the analog circuits in the A/D converter. V<sub>REF</sub> is a conversion reference voltage.

To protect the reliability of the chip, AV<sub>CC</sub>, AV<sub>SS</sub>, V<sub>CC</sub>, and V<sub>SS</sub> should be related as follows: AV<sub>CC</sub> = V<sub>CC</sub> ± 10%; AV<sub>SS</sub> = V<sub>SS</sub>. AV<sub>CC</sub> and AV<sub>SS</sub> must not be left open, even if the A/D converter is not used (including hardware/software standby mode). Voltages applied to the analog input pins should be in the range AV<sub>SS</sub> ≤ AN<sub>n</sub> ≤ V<sub>REF</sub>.

**Table 15-1 A/D Converter Pins**

Pin Name	Abbreviation	Input/Output	Function
Analog power supply	AV <sub>CC</sub>	Input	Analog power supply
Analog ground	AV <sub>SS</sub>	Input	Analog ground and reference voltage
Reference voltage	V <sub>REF</sub>	Input	Analog reference voltage
Analog input 0	AN <sub>0</sub>	Input	Analog input pins 0 to 3 (analog group 0)
Analog input 1	AN <sub>1</sub>	Input	
Analog input 2	AN <sub>2</sub>	Input	
Analog input 3	AN <sub>3</sub>	Input	
Analog input 4	AN <sub>4</sub>	Input	Analog input pins 4 to 7 (analog group 1)
Analog input 5	AN <sub>5</sub>	Input	
Analog input 6	AN <sub>6</sub>	Input	
Analog input 7	AN <sub>7</sub>	Input	
Analog input 8	AN <sub>8</sub>	Input	Analog input pins 8 to 11 (analog group 2)
Analog input 9	AN <sub>9</sub>	Input	
Analog input 10	AN <sub>10</sub>	Input	
Analog input 11	AN <sub>11</sub>	Input	
A/D trigger	ADTRG	Input	External trigger pin for A/D conversion



## 15.1.4 Register Configuration

Table 15-2 summarizes the A/D converter's registers.

**Table 15-2 A/D Converter Registers**

<b>Address</b>	<b>Name</b>	<b>Abbreviation</b>	<b>R/W</b>	<b>Initial Value</b>
H'FEA0	A/D data register 0 (high/low)	ADDR0(H/L)	R	H'0000
H'FEA2	A/D data register 1 (high/low)	ADDR1(H/L)	R	H'0000
H'FEA4	A/D data register 2 (high/low)	ADDR2(H/L)	R	H'0000
H'FEA6	A/D data register 3 (high/low)	ADDR3(H/L)	R	H'0000
H'FEA8	A/D data register 4 (high/low)	ADDR4(H/L)	R	H'0000
H'FEAA	A/D data register 5 (high/low)	ADDR5(H/L)	R	H'0000
H'FEAC	A/D data register 6 (high/low)	ADDR6(H/L)	R	H'0000
H'FEAE	A/D data register 7 (high/low)	ADDR7(H/L)	R	H'0000
H'FEB0	A/D data register 8 (high/low)	ADDR8(H/L)	R	H'0000
H'FEB2	A/D data register 9 (high/low)	ADDR9(H/L)	R	H'0000
H'FEB4	A/D data register A (high/low)	ADDRA(H/L)	R	H'0000
H'FEB6	A/D data register B (high/low)	ADDRB(H/L)	R	H'0000
H'FEB8	A/D control/status register	ADCSR	R/W*	H'00
H'FEB9	A/D control register	ADCR	R/W	H'1F
H'FEDC	A/D trigger register	ADTRGR	R/W	H'FF

Note: \* Software can write 0 in bit 7 of the A/D control/status register (ADCSR) to clear the flag, but cannot write 1.

## 15.2 Register Descriptions

### 15.2.1 A/D Data Registers 0 to B

A/D data registers 0 to B (ADDR0 to ADDR<sub>B</sub>) are 16-bit read-only registers that store the results of A/D conversion of the analog inputs. There are 12 registers, corresponding to analog inputs 0 to 11 (AN<sub>0</sub> to AN<sub>11</sub>). The A/D data registers are initialized to H'0000 by a reset and in the standby modes.

Bit	7	6	5	4	3	2	1	0
ADDRnH (upper byte)	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
Initial value	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Bit	7	6	5	4	3	2	1	0
ADDRnL (lower byte)	AD1	AD0	—	—	—	—	—	—
Initial value	0	0	0	0	0	0	0	0
R/W (n = 0 to B)	R	R	R	R	R	R	R	R

The on-chip A/D converter converts the analog inputs to 10-bit digital values. The upper eight of the 10 bits are stored in the upper byte of the A/D data register of the selected channel. The lower two bits are stored in the lower byte of the A/D data register. Only the two upper bits of the lower byte of an A/D data register are valid. Table 15-3 indicates the pairings of analog input channels and A/D data registers.

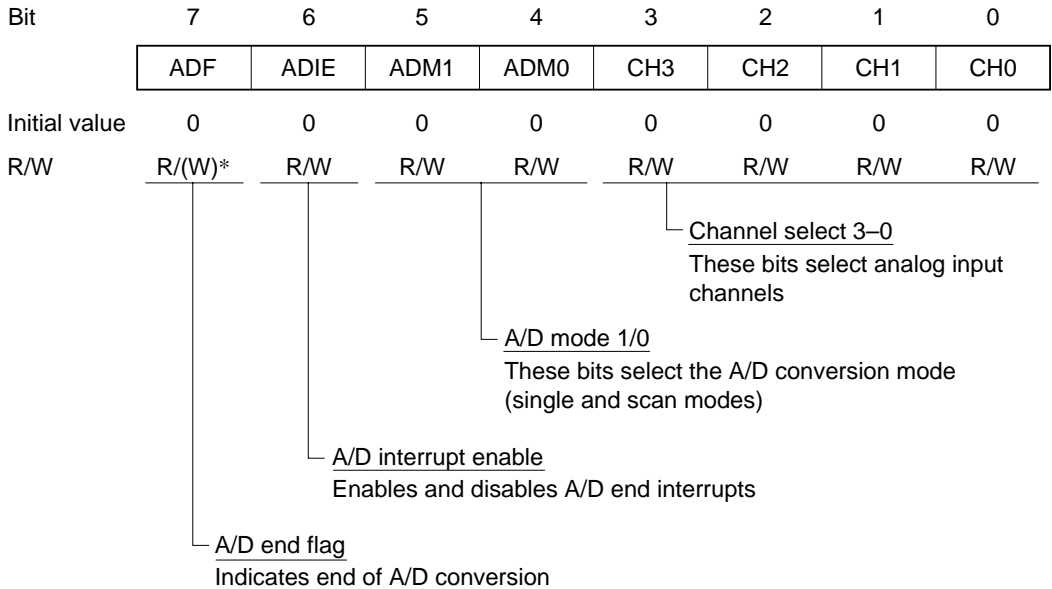
The H8/500 CPU can always read and write the A/D data registers. The upper byte must always be read before the lower byte. It is possible to read only the upper byte of an A/D data register, but it is not possible to read only the lower byte. For further details see section 15.3, “H8/500 CPU Interface.” Bits 5 to 0 of the A/D data registers are reserved bits that cannot be modified and always read 0.

**Table 15-3 Analog Input Channels and A/D Data Registers**

Analog Input Channel	A/D Data Register	Analog Input Channel	A/D Data Register	Analog Input Channel	A/D Data Register
AN <sub>0</sub>	ADDR0	AN <sub>4</sub>	ADDR4	AN <sub>8</sub>	ADDR8
AN <sub>1</sub>	ADDR1	AN <sub>5</sub>	ADDR5	AN <sub>9</sub>	ADDR9
AN <sub>2</sub>	ADDR2	AN <sub>6</sub>	ADDR6	AN <sub>10</sub>	ADDRA
AN <sub>3</sub>	ADDR3	AN <sub>7</sub>	ADDR7	AN <sub>11</sub>	ADDRB

**15.2.2 A/D Control Status Register**

The A/D control status register (ADCSR) is an eight-bit readable/writable register that selects the A/D conversion mode. ADCSR is initialized to H'00 by a reset and in the standby modes.



Note: \* Software can write 0 to clear the flag, but cannot write 1.

(1) **Bit 7—A/D End Flag (ADF):** Indicates the end of A/D conversion. ADF is initialized to 0 by a reset and in the standby modes.

#### Bit 7

ADF	Description
0	A/D conversion is in progress or the A/D converter is idle (Initial value) ADF is cleared to 0 when: 1. Software reads ADF after it has been set to 1, then writes 0 in ADF 2. The DTC is started by ADI
1	A/D conversion has ended and a digital value has been loaded into one or more A/D data registers ADF is set to 1 when: 1. A/D conversion ends in single mode 2. All conversion in one selected analog group ends

After ADF is set to 1, the A/D converter operates differently in single mode and scan mode. In single mode, after loading a digital value into an A/D data register, the A/D converter sets ADF to 1 then goes into the idle state. In scan mode, after completing all conversion in one selected analog group, the A/D converter sets ADF to 1 then continues converting.

Software cannot write 1 in ADF.

(2) **Bit 6—A/D Interrupt Enable (ADIE):** Enables or disables the A/D end interrupt (ADI). ADIE is initialized to 0 by a reset and in the standby modes.

#### Bit 6

ADIE	Description
0	A/D end interrupt (ADI) is disabled (Initial value)
1	A/D end interrupt (ADI) is enabled

When A/D conversion ends and the ADF bit in ADCSR is set to 1, if ADIE is also set to 1 an A/D end interrupt (ADI) is requested. The ADI interrupt request can be cleared by clearing ADF to 0 or clearing ADIE to 0.

**(3) Bits 5 and 4—A/D Mode 1/0 (ADM1/0):** These bits select single mode, four-channel scan mode, eight-channel scan mode, or 12-channel scan mode as the A/D conversion mode. ADM1 and ADM0 are cleared to 00 by a reset and in the standby modes, selecting single mode. To ensure correct operation, always clear ADST to 0 before changing the conversion mode.

<b>Bit 5</b>	<b>Bit 4</b>	<b>Description</b>
<b>ADM1</b>	<b>ADM0</b>	
0	0	Single mode (Initial value)
0	1	Four-channel scan mode (analog group 0, 1, or 2)
1	0	Eight-channel scan mode (analog groups 0 and 1)
1	1	Twelve-channel scan mode (analog groups 0, 1, and 2)

When ADM1 and ADM0 are cleared to 00, single mode is selected. In single mode one analog channel is converted once. The channel is selected by bits CH3 to CH0 in ADCSR.

Setting ADM1 and ADM0 to 01 selects four-channel scan mode. In scan mode, one or more channels are converted continuously. The channels converted in scan mode are selected by bits CH3 to CH0 in ADCSR. In four-channel scan mode, A/D conversion is performed in the four channels in analog group 0 (AN<sub>0</sub> to AN<sub>3</sub>), analog group 1 (AN<sub>4</sub> to AN<sub>7</sub>), or analog group 2 (AN<sub>8</sub> to AN<sub>11</sub>).

Setting ADM1 and ADM0 to 10 selects eight-channel scan mode. A/D conversion is performed in the eight channels in analog group 0 (AN<sub>0</sub> to AN<sub>3</sub>) and analog group 1 (AN<sub>4</sub> to AN<sub>7</sub>).

Setting ADM1 and ADM0 to 11 selects 12-channel scan mode. A/D conversion is performed in the 12 channels in analog group 0 (AN<sub>0</sub> to AN<sub>3</sub>), analog group 1 (AN<sub>4</sub> to AN<sub>7</sub>), and analog group 2 (AN<sub>8</sub> to AN<sub>11</sub>).

For further details on operation in single and scan modes, see section 15.4, “Operation.”

**(4) Bits 3 to 0—Channel Select 3 to 0 (CH3 to CH0):** These bits and ADM1 and ADM0 select the analog input channels. CH3 to CH0 are initialized to 0000 by a reset and in the standby modes. To ensure correct operation, always clear ADST to 0 in the A/D control register (ADCR) before changing the analog input channel selection.

Bit 3	Bit 2	Bit 1	Bit 0	Analog Input Channels	
				Single Mode	Four-Channel Scan Mode
CH3	CH2	CH1	CH0		
0	0	0	0	AN <sub>0</sub> (Initial value)	AN <sub>0</sub>
		0	1	AN <sub>1</sub>	AN <sub>0,1</sub>
		1	0	AN <sub>2</sub>	AN <sub>0-2</sub>
		1	1	AN <sub>3</sub>	AN <sub>0-3</sub>
	1	0	0	AN <sub>4</sub>	AN <sub>4</sub>
		0	1	AN <sub>5</sub>	AN <sub>4,5</sub>
		1	0	AN <sub>6</sub>	AN <sub>4-6</sub>
		1	1	AN <sub>7</sub>	AN <sub>4-7</sub>
1	0*1	0	0	AN <sub>8</sub>	AN <sub>8</sub>
		0	1	AN <sub>9</sub>	AN <sub>8,9</sub>
		1	0	AN <sub>10</sub>	AN <sub>8-10</sub>
		1	1	AN <sub>11</sub>	AN <sub>8-11</sub>

Bit 3	Bit 2	Bit 1	Bit 0	Analog Input Channels	
				Eight-Channel Scan Mode	12-Channel Scan Mode
CH3	CH2	CH1	CH0		
0	0	0	0	AN <sub>0,4</sub>	AN <sub>0,4,8</sub>
		0	1	AN <sub>0,1,4,5</sub>	AN <sub>0,1,4,5,8,9</sub>
		1	0	AN <sub>0-2,4-6</sub>	AN <sub>0-2,4-6,8-10</sub>
		1	1	AN <sub>0-7</sub>	AN <sub>0-11</sub>
	1	0	0	AN <sub>0,4</sub>	AN <sub>0,4,8</sub>
		0	1	AN <sub>0,1,4,5</sub>	AN <sub>0,1,4,5,8,9</sub>
		1	0	AN <sub>0-2,4-6</sub>	AN <sub>0-2,4-6,8-10</sub>
		1	1	AN <sub>0-7</sub>	AN <sub>0-11</sub>
1	0*1	0	0	Reserved*2	AN <sub>0,4,8</sub>
		0	1		AN <sub>0,1,4,5,8,9</sub>
		1	0		AN <sub>0-2,4-6,8-10</sub>
		1	1		AN <sub>0-11</sub>

Notes: 1. Must be cleared to 0.

2. Reserved for future expansion. Must not be used.

### 15.2.3 A/D Control Register

The A/D control register (ADCR) is an eight-bit readable/writable register that controls the start of A/D conversion and selects the A/D clock. ADCR is initialized to H'1F by a reset and in the standby modes. Bits 4 to 0 of ADCR are reserved for future expansion. They cannot be modified and always read 1.

Bit	7	6	5	4	3	2	1	0
	TRGE	CKS	ADST	—	—	—	—	—
Initial value	0	0	0	1	1	1	1	1
R/W	R/W	R/W	R/W	—	—	—	—	—

└─ Trigger enable  
Enables and disables external triggering of A/D conversion

└─ Clock select  
Selects the A/D conversion time

└─ A/D start  
Starts and stops A/D conversion

└─ Reserved bits

**(1) Bit 7—Trigger Enable (TRGE):** Enables or disables external triggering of A/D conversion. When TRGE is set to 1, P7<sub>1</sub> automatically becomes the  $\overline{\text{ADTRG}}$  input pin. TRGE is initialized to 0 by a reset and in the standby modes.

#### Bit 7

TRGE	Description
0	A/D conversion cannot be externally triggered (Initial value)
1	A/D conversion can be externally triggered (P7 <sub>1</sub> is the $\overline{\text{ADTRG}}$ pin)

After TRGE is set to 1, if a low pulse is input at the  $\overline{\text{ADTRG}}$  pin, the A/D converter detects the falling edge of the pulse and sets the ADST bit in ADCR to 1. Subsequent operation is the same as if software had set the ADST bit to 1. External triggering operates only when the ADST bit is cleared to 0.

When the external trigger function is used, the low pulse input at the  $\overline{\text{ADTRG}}$  pin must have a width of at least 1.5 system clocks (1.5 $\phi$ ). For further details see section 15.4.4, “External Triggering of A/D Conversion.”

**(2) Bit 6—Clock Select (CKS):** Selects the A/D conversion time. A/D conversion is performed in 266 states when CKS is cleared to 0, or in 134 states when CKS is set to 1. CKS is initialized to 0 by a reset and in the standby modes. To ensure correct operation, always clear ADST to 0 before changing the A/D conversion time.

#### Bit 6

CKS	Description
0	Conversion time = 266 states (maximum) (Initial value)
1	Conversion time = 134 states (maximum)

**(3) Bit 5—A/D Start (ADST):** Starts and stops A/D conversion. A/D conversion starts when ADST is set to 1 and stops when ADST is cleared to 0. ADST is initialized to 0 by a reset and in the standby modes.

#### Bit 5

ADST	Description
0	A/D conversion is stopped (Initial value)
1	A/D conversion is in progress Clearing conditions: 1. Single mode: cleared to 0 automatically at the end of A/D conversion 2. Scan mode: check that ADF is set to 1 in ADCSR, then write 0 in ADST

The ADST bit operates differently in single and scan modes. In single mode, ADST is cleared to 0 automatically after A/D conversion of one channel. In scan mode, after all selected analog inputs have been converted A/D conversion of all these channels begins again, so ADST remains set to 1. When the conversion time or analog input channel selection is changed in scan mode, the ADST bit should first be cleared to 0 to halt A/D conversion.

Before changing the A/D conversion time (CKS bit in ADCR), operating mode (ADM1/0 bits in ADCSR), or analog input channel selection (bits CH3 to CH0 in ADCSR), always check that the A/D converter is stopped (ADST = 0). Making these changes while the A/D converter is operating (ADST = 1) may produce incorrect values in the A/D data registers.

**(4) Bits 4 to 0—Reserved:** These bits are reserved for future expansion. They cannot be modified and always read 1.

### 15.2.4 A/D Trigger Register

The A/D trigger register (ADTRGR) is used to switch the A/D external trigger. The A/D external trigger can be selected from the ADTRG pin or an IPU channel 1 DR3 compare match. ADTRGR is set to H'FF in standby mode and by a reset.



Bit	7	6	5	4	3	2	1	0
	EXTRG	—	—	—	—	—	—	—
Initial value	1	1	1	1	1	1	1	1
R/W	R/W	—	—	—	—	—	—	—

Reserved bits  
External trigger source select  
Selects an IPU channel compare match or the ADTRG pin as the A/D external trigger

**(1) Bit 7—External Trigger Source Select (EXTRG):** This bit selects the A/D external trigger from an IPU channel 1 compare match or the ADTRG pin. The A/D external trigger source is the ADTRG pin when EXTRG is set to 1, and an IPU channel 1 DR3 compare match when EXTRG is cleared to 0. EXTRG is initialized to 1 by a reset and in standby mode.

#### Bit 7

EXTRG	Description
0	A TPU channel 1 DR3 compare match is set as the A/D external trigger source
1	The ADTRG pin is set as the A/D external trigger source (Initial value)

**(2) Bit 6—Bits 6 to 0:** Reserved: These bits are reserved for future expansion. They are always read as 1 and cannot be modified.

For a description of a sample operation, see section 15.4.5 “Starting the A/D Converter with the IPU.”

### 15.3 H8/500 CPU Interface

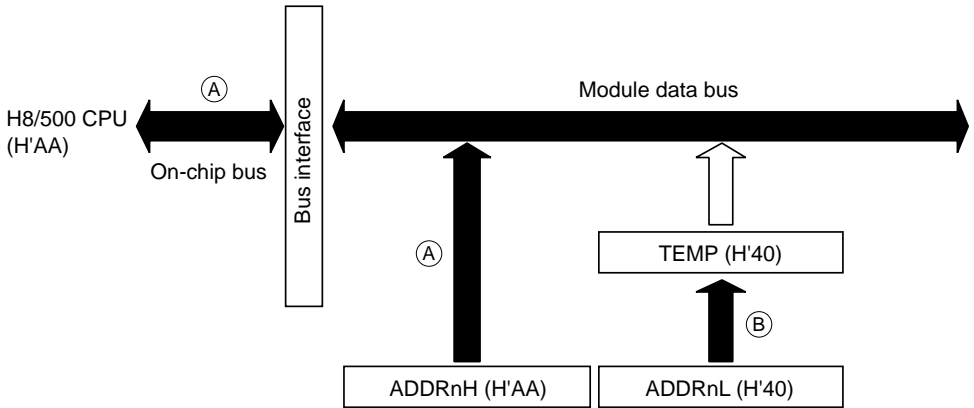
A/D data registers 0 to B (ADDR0 to ADDR B) are 16-bit registers, but they are connected to the H8/500 CPU via an eight-bit on-chip data bus. The upper and lower bytes of an A/D data register are necessarily read separately. To prevent data from changing between the reading of the upper and lower bytes of an A/D data register, the lower byte is read using a temporary register (TEMP). The upper byte can be read directly.

An A/D data register is read as follows. The upper byte must be read first. The H8/500 CPU receives the upper-byte data directly at this time. At the same time, the A/D converter transfers the lower-byte data internally into TEMP. Next, when the lower byte is read, the H8/500 CPU receives the contents of TEMP.

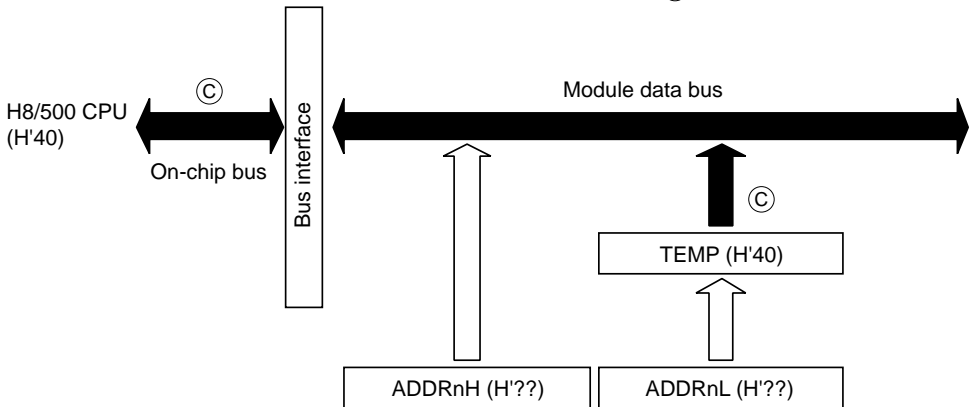
When reading an A/D data register using byte operand size, always read the upper byte before the lower byte. It is possible to read only the upper byte, but if only the lower byte is read incorrect data may be obtained. When an A/D data register is read using word operand size, the upper byte will automatically be read before the lower byte.

Figure 15-2 shows the data flow when an A/D data register is read. In the example shown, the upper byte of the A/D data register contains H'AA and the lower byte contains H'40. First the H8/500 CPU reads H'AA directly from the upper byte while H'40 is transferred to TEMP in the A/D converter. Next, when the H8/500 CPU reads the lower byte of the A/D data register, it obtains the TEMP contents.

(1) ADDRnH (upper byte) read: ADDRnH [H'AA] → H8/500 CPU [H'AA] (A)  
 ADDRnL [H'40] → TEMP [H'40] (B)



(2) ADDRnL (lower byte) read: ADDRnH [H'??] → Don't care  
 ADDRnL [H'??] → Don't care  
 TEMP [H'40] → H8/500 CPU [H'40] (C)



**Figure 15-2 A/D Data Register Read Operation (Reading H'AA40)**

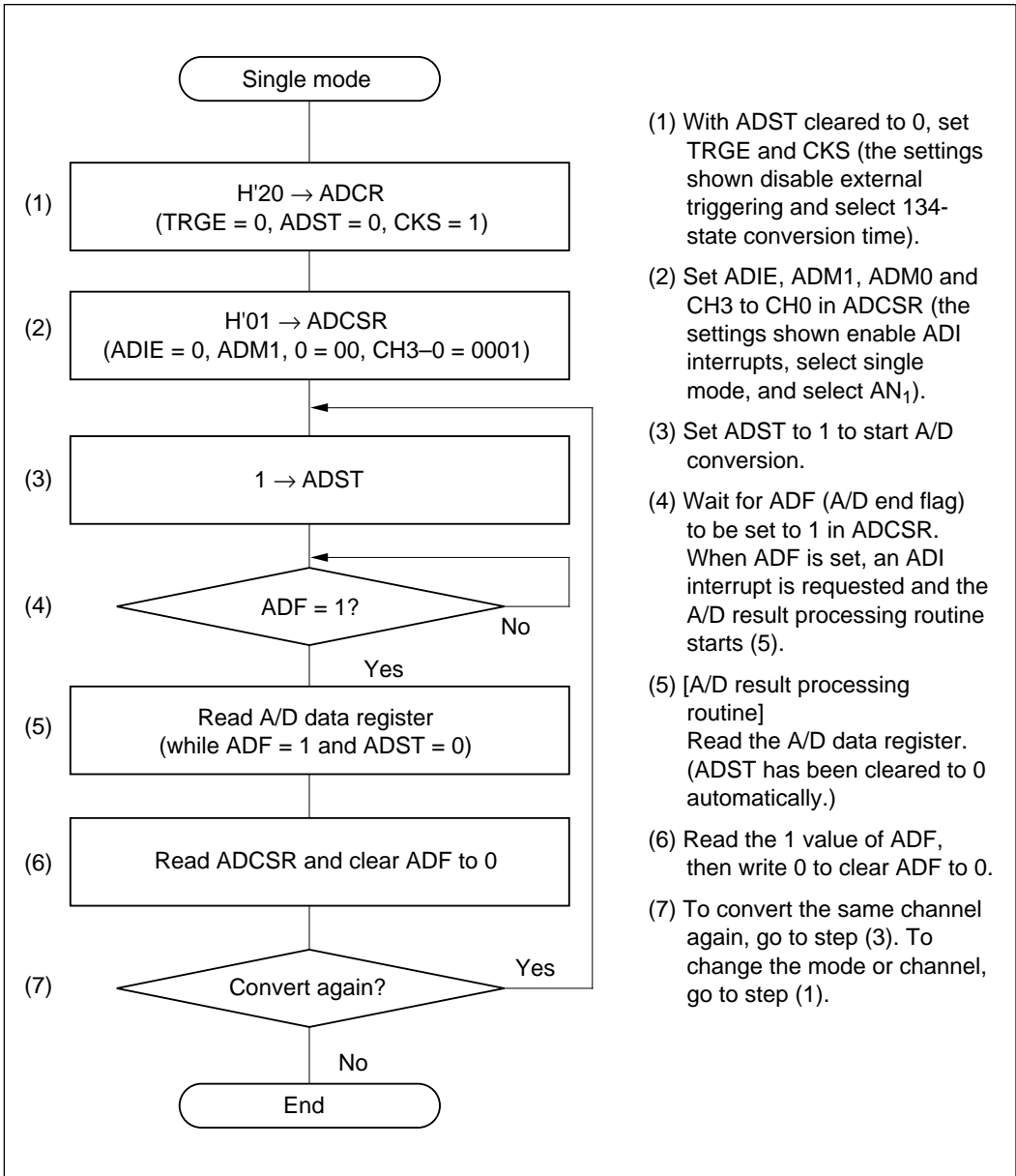
## 15.4 Operation

The A/D converter operates by successive approximations with 10-bit resolution. It has two operating modes: single mode and scan mode. In single mode, one selected channel is converted once. In scan mode, one or more selected channels are converted repeatedly until the ADST bit in the A/D control register (ADCR) is cleared to 0.

### 15.4.1 Single Mode

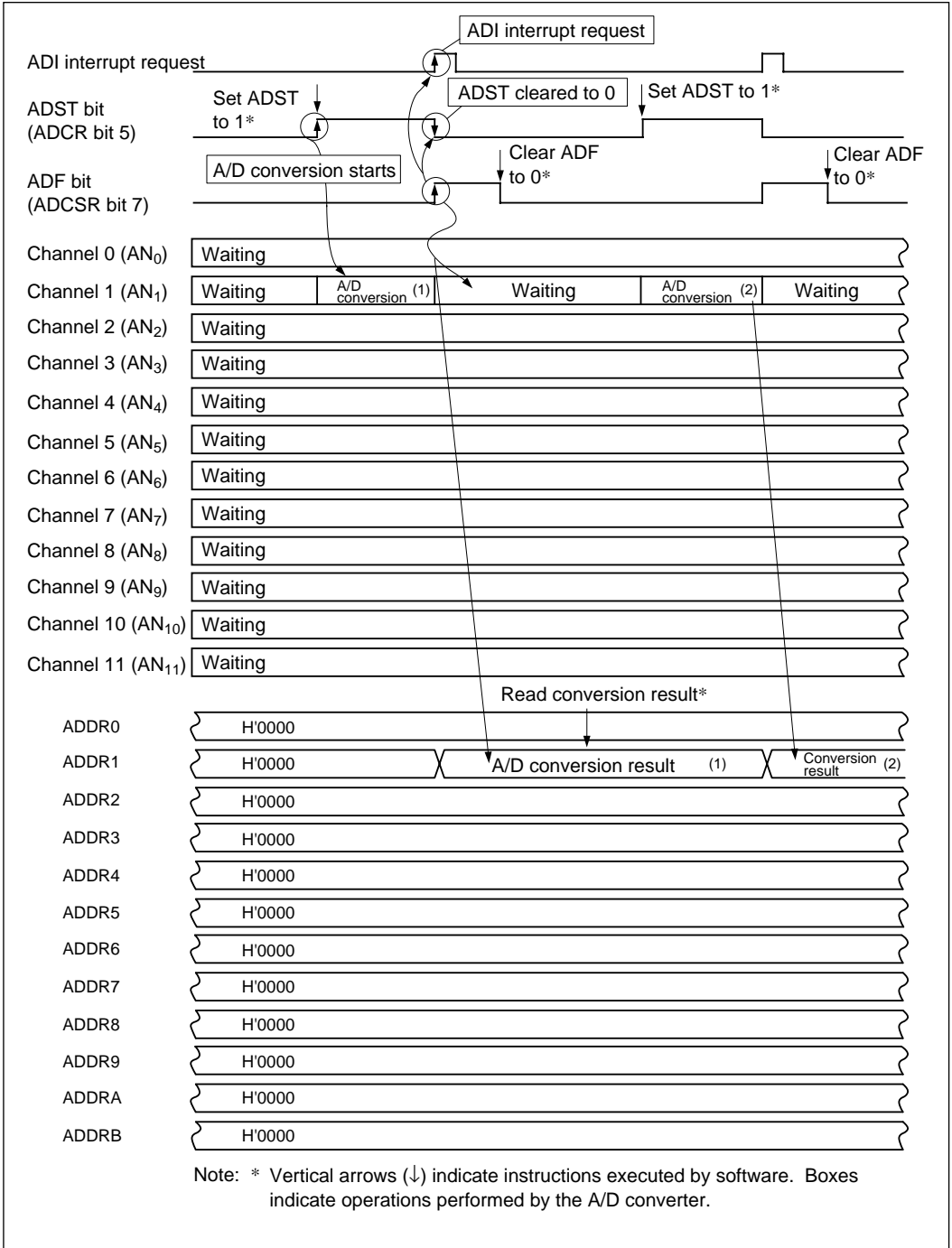
Single mode can be selected to perform one A/D conversion on one channel. Single mode is selected by clearing bits ADM1 and ADM0 to 00 in the A/D control/status register (ADCSR). A/D conversion then starts when the ADST bit is set to 1 in ADCR. The ADST bit remains set to 1 during A/D conversion and is automatically cleared to 0 when conversion ends. When conversion ends the ADF bit is set to 1 in ADCSR. If the ADIE bit is also set to 1, an ADI interrupt is requested. To clear ADF to 0, first read ADF after ADF has been set to 1, then write 0 in ADF. If the ADI interrupt is served by the data transfer controller (DTC), however, ADF is cleared to 0 automatically.

Figure 15-3 shows a flowchart for selecting analog input channel 1 ( $AN_1$ ) and performing A/D conversion in single mode. Figure 15-4 is a timing diagram.



- (1) With ADST cleared to 0, set TRGE and CKS (the settings shown disable external triggering and select 134-state conversion time).
- (2) Set ADIE, ADM1, ADM0 and CH3 to CH0 in ADCSR (the settings shown enable ADI interrupts, select single mode, and select AN<sub>1</sub>).
- (3) Set ADST to 1 to start A/D conversion.
- (4) Wait for ADF (A/D end flag) to be set to 1 in ADCSR. When ADF is set, an ADI interrupt is requested and the A/D result processing routine starts (5).
- (5) [A/D result processing routine]  
Read the A/D data register. (ADST has been cleared to 0 automatically.)
- (6) Read the 1 value of ADF, then write 0 to clear ADF to 0.
- (7) To convert the same channel again, go to step (3). To change the mode or channel, go to step (1).

**Figure 15-3 Flowchart for Single Mode**



**Figure 15-4 Example of A/D Converter Operation (Single Mode, Channel 1 Selected)**

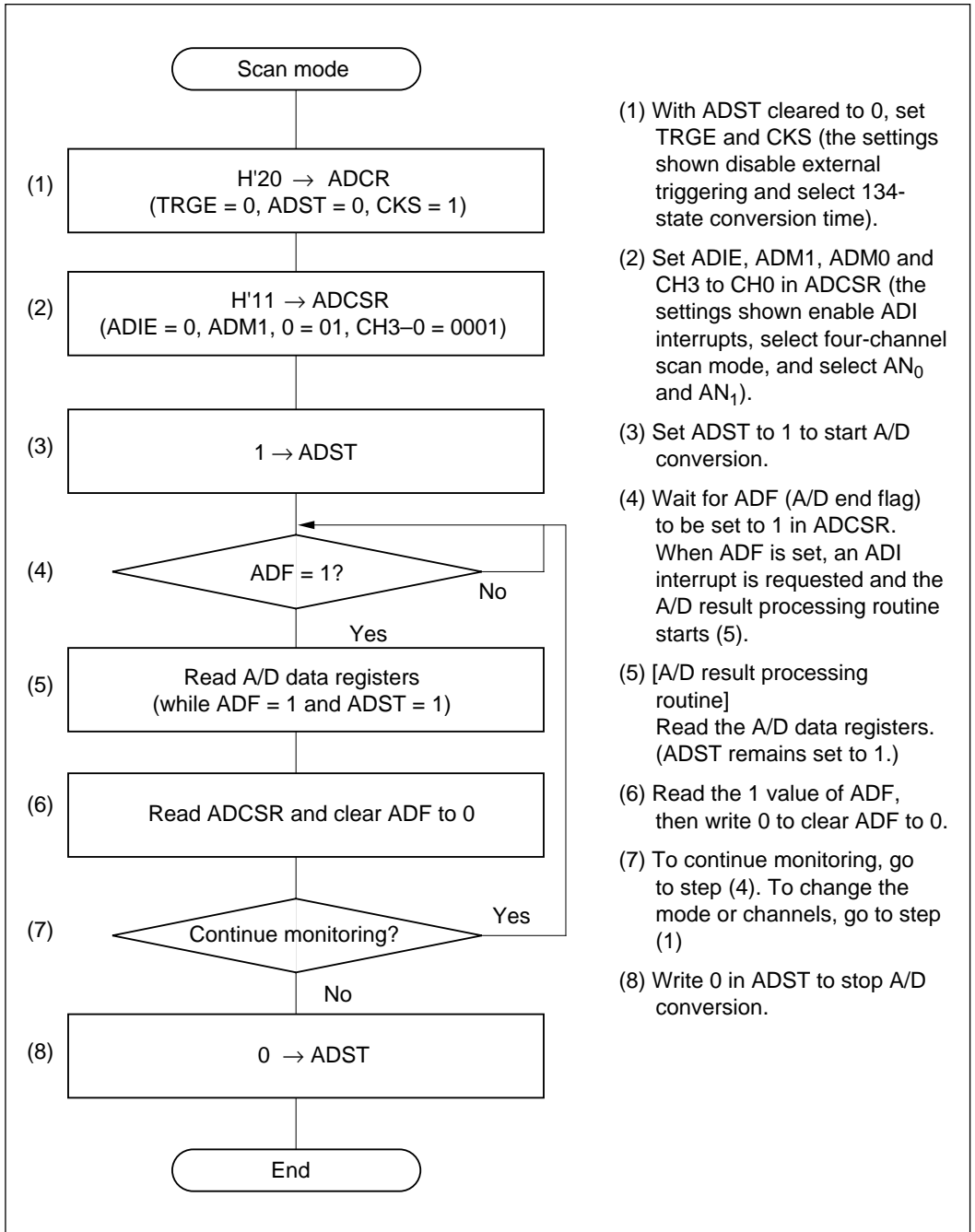
## 15.4.2 Scan Mode

Scan mode can be selected to perform A/D conversion on one or more channels repeatedly (to monitor the channels continuously, for example). Scan mode is selected by setting bits ADM1 and ADM0 in the A/D control/status register (ADCSR) to 01, 10, or 11. The 01 setting selects four-channel scan mode. The 10 setting selects eight-channel scan mode. The 11 setting selects 12-channel scan mode. A/D conversion starts when the ADST bit in ADCR is set to 1.

In scan mode the channels are converted in ascending order of channel number ( $AN_0$ ,  $AN_1$ , ...,  $AN_{11}$ ). The ADST bit remains set to 1 until software clears it to 0.

When all conversion in one selected analog group is completed, the ADF bit in ADCSR is set to 1, then A/D conversion is performed again. If the ADIE bit in ADCSR is set to 1, then when ADF is set to 1 an ADI interrupt is requested. To clear ADF to 0, first read ADF after it has been set to 1, then write 0 in ADF. If the ADI interrupt is served by the data transfer controller (DTC), however, ADF is cleared to 0 automatically.

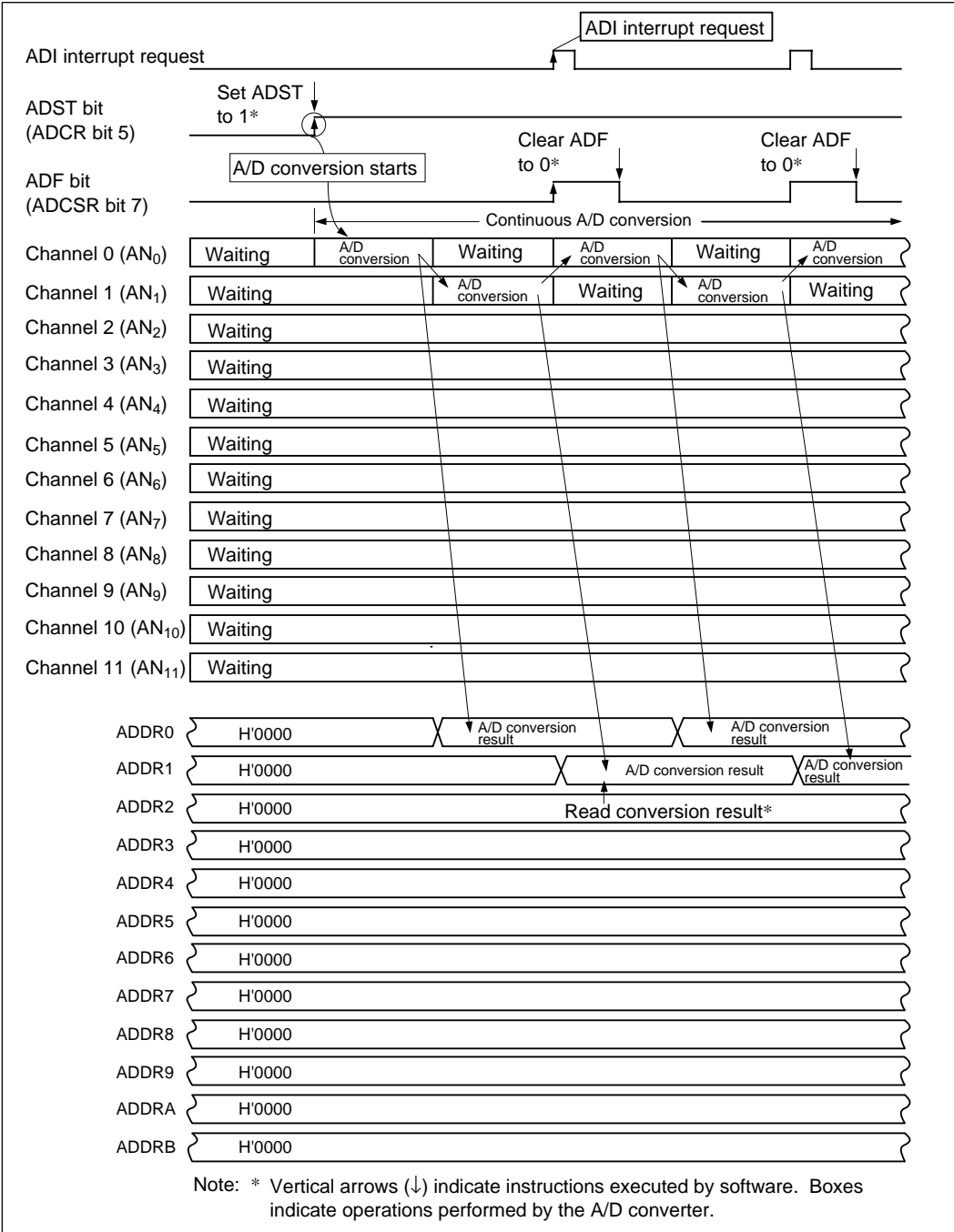
Figure 15-5 shows a flowchart for selecting analog input channels 0 and 1 ( $AN_0$  and  $AN_1$ ) and performing A/D conversion in four-channel scan mode. Figure 15-6 is a timing diagram.



- (1) With ADST cleared to 0, set TRGE and CKS (the settings shown disable external triggering and select 134-state conversion time).
- (2) Set ADIE, ADM1, ADM0 and CH3 to CH0 in ADCSR (the settings shown enable ADI interrupts, select four-channel scan mode, and select AN<sub>0</sub> and AN<sub>1</sub>).
- (3) Set ADST to 1 to start A/D conversion.
- (4) Wait for ADF (A/D end flag) to be set to 1 in ADCSR. When ADF is set, an ADI interrupt is requested and the A/D result processing routine starts (5).
- (5) [A/D result processing routine]  
Read the A/D data registers. (ADST remains set to 1.)
- (6) Read the 1 value of ADF, then write 0 to clear ADF to 0.
- (7) To continue monitoring, go to step (4). To change the mode or channels, go to step (1)
- (8) Write 0 in ADST to stop A/D conversion.

**Figure 15-5 Flowchart for Scan Mode**





**Figure 15-6 Example of A/D Converter Operation (Four-Channel Scan Mode, Channels 0 and 1 Selected)**

### 15.4.3 Analog Input Sampling and A/D Conversion Time

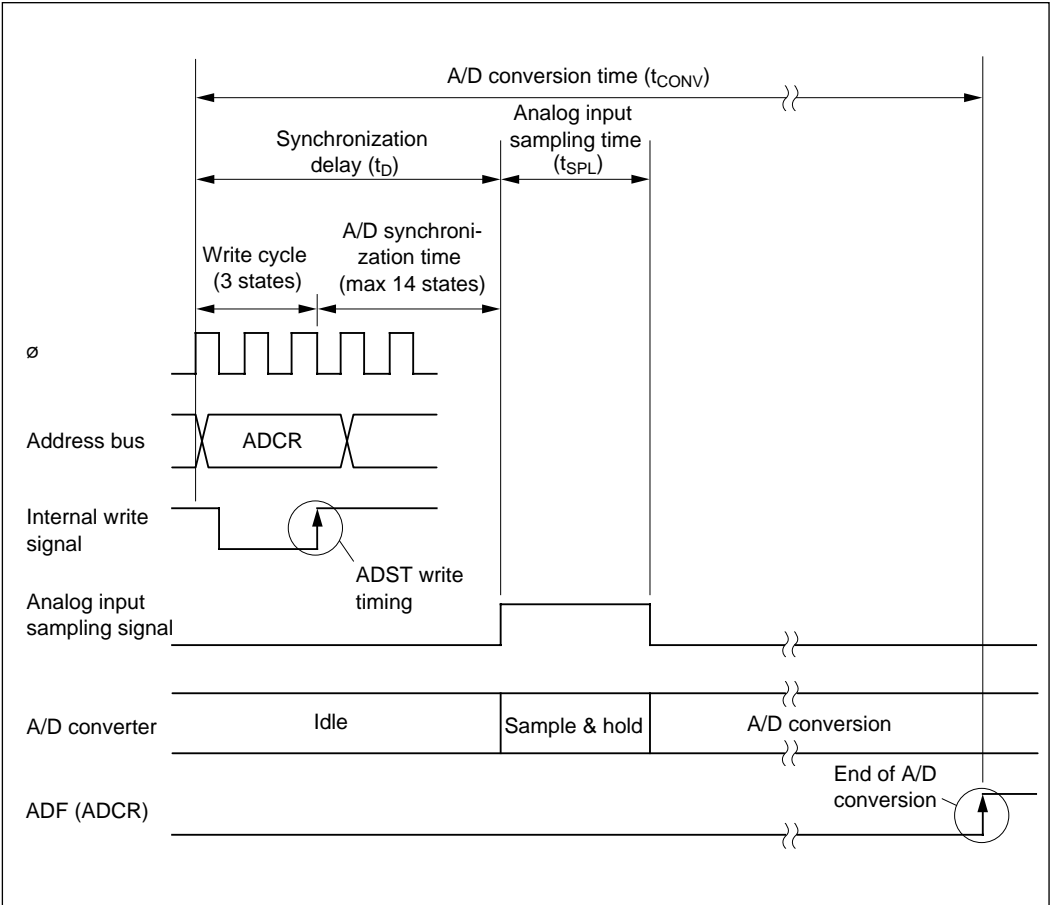
The A/D converter has a built-in sample-and-hold circuit. The A/D converter starts sampling the analog inputs at a time  $t_D$  (synchronization delay) after the ADST bit is set to 1 in the A/D control register (ADCR). Figure 15-7 shows the sampling timing.

The A/D conversion time ( $t_{CONV}$ ) includes  $t_D$  and the analog input sampling time ( $t_{SPL}$ ). The length of  $t_D$  varies because it includes time needed to synchronize the A/D converter. The total conversion time therefore varies within the ranges indicated in table 15-4.

In scan mode, the  $t_{CONV}$  values given in table 15-4 apply to the first conversion. In the second and subsequent conversions there is no  $t_D$ , and  $t_{CONV}$  is fixed at 256 states when  $CKS = 0$  or 128 states when  $CKS = 1$ .

**Table 15-4 A/D Conversion Time (Single Mode)**

Item	Symbol	CKS = 0			CKS = 1			Unit
		Min	Typ	Max	Min	Typ	Max	
Synchronization delay	$t_D$	10	—	17	6	—	9	States
Input sampling time	$t_{SPL}$	—	80	—	—	40	—	
A/D conversion time	$t_{CONV}$	259	—	266	131	—	134	

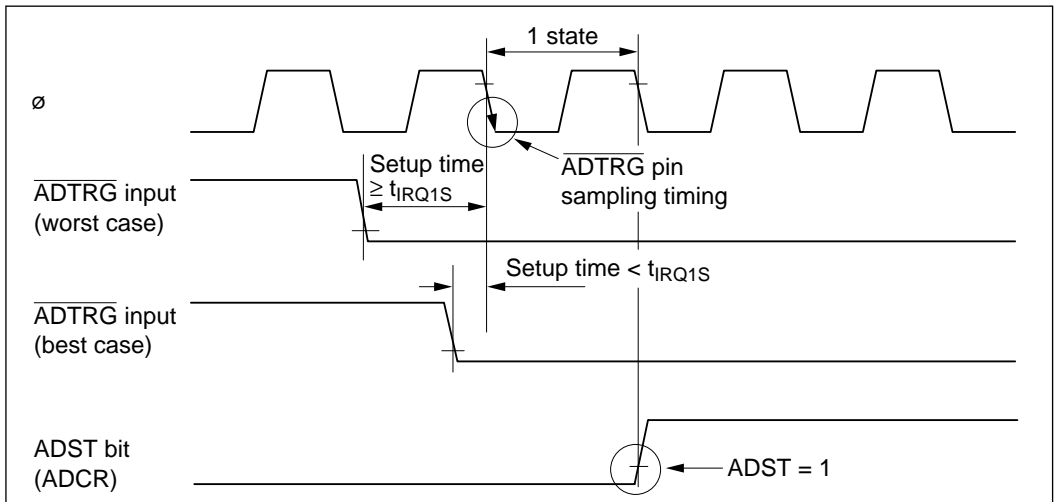


**Figure 15-7 A/D Conversion Timing**

#### 15.4.4 External Triggering of A/D Conversion

A/D conversion can be started by input of an external trigger signal. External triggering is enabled by setting the TRGE bit to 1 in the A/D control register. When the TRGE bit is set to 1, P7<sub>1</sub> automatically becomes the  $\overline{\text{ADTRG}}$  input pin. If a low pulse is input at the  $\overline{\text{ADTRG}}$  pin in this state, the A/D converter detects the falling edge of the pulse and sets the ADST bit to 1. Figure 15-8 shows the external trigger input timing.

The ADST bit is set to 1 one state after the A/D converter samples the falling edge of the  $\overline{\text{ADTRG}}$  signal. The time from when the ADST bit is set to 1 until A/D conversion begins is the same as when software writes 1 in ADST.



**Figure 15-8 External Trigger Input Timing**

#### 15.4.5 Starting A/D Conversion by IPU

In the H8/539F, A/D conversion can be started by a compare match in the on-chip integrated-timer pulse unit (IPU). To start A/D conversion by IPU compare match, follow the procedure given next.

1. Set bits DOE21 and DOE20 (bits 7 and 6) to 1, 0 in IPU channel 1 timer output enable register A (TOERA).
2. Set the starting time of the A/D converter in IPU channel 1 dedicated register 2 (DR2).
3. Set the TRGE bit (bit 7) in the A/D control register (ADCR) to 1.
4. Clear bit 7 of the ADTRGR register at address H'FEDC to 0.

After these settings, A/D conversion will start when the IPU channel 1 timer counter value matches DR2. In this case A/D conversion cannot be started by input at the  $\overline{\text{ADTRG}}$  pin. When the IPU starts A/D conversion, the timing is the same as if the T1OC<sub>2</sub> pin were externally connected to the  $\overline{\text{ADTRG}}$  pin. See the relevant timing diagrams for these pins.

## 15.5 Interrupts and DTC

The A/D converter can request an A/D end interrupt (ADI) at the end of conversion. ADI is enabled when the ADIE bit is set to 1 in the A/D control/status register (ADCSR), and disabled when ADIE is cleared to 0.

If the ADI bit in the interrupt controller's data transfer enable register A (DTEA) is set to 1, the ADI interrupt is served by the data transfer controller (DTC). When the DTC is started by ADI to perform a data transfer, the ADF bit in ADCSR is automatically cleared to 0. For further details on the DTC, see section 7, "Data Transfer Controller."

## 15.6 Usage Notes

When using the A/D converter, note the following points:

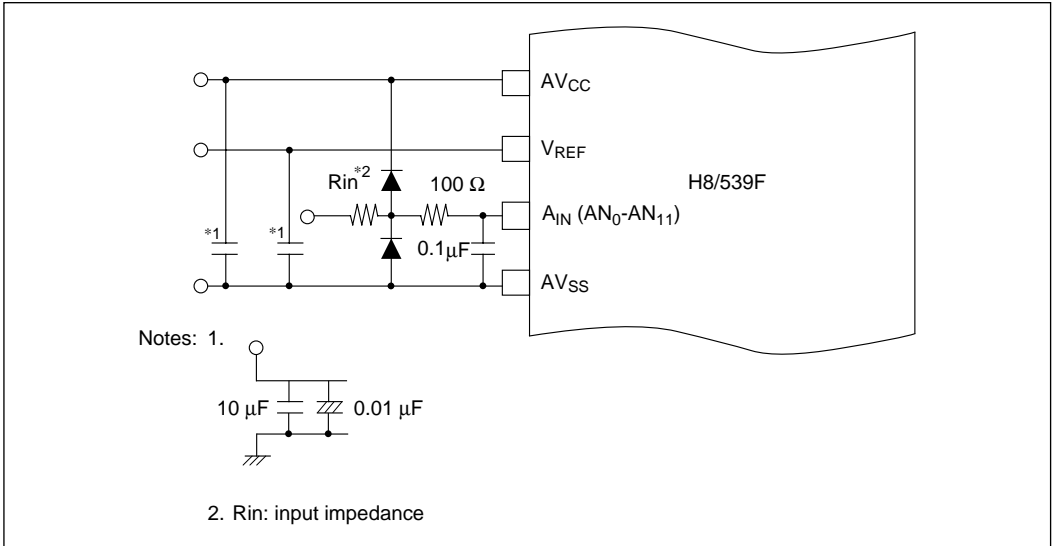
- (1) Analog Input Voltage Range:** During A/D conversion, the voltages input to the analog input pins should be in the range  $AV_{SS} \leq A_{IN} \leq V_{REF}$ .
- (2) Relationships of  $AV_{CC}$  and  $AV_{SS}$  to  $V_{CC}$  and  $V_{SS}$ :**  $AV_{CC}$ ,  $AV_{SS}$ ,  $V_{CC}$ , and  $V_{SS}$  should be related as follows:  $AV_{CC} = V_{CC} \pm 10\%$ ;  $AV_{SS} = V_{SS}$ .  $AV_{CC}$  and  $AV_{SS}$  must not be left open, even if the A/D converter is not used (include hardware/software stand-by mode).
- (3)  $V_{REF}$  Input Range:** The reference voltage input at the  $V_{REF}$  pin should be in the range  $V_{REF} \leq AV_{CC}$ .

Failure to observe points (1), (2), and (3) above may degrade chip reliability.

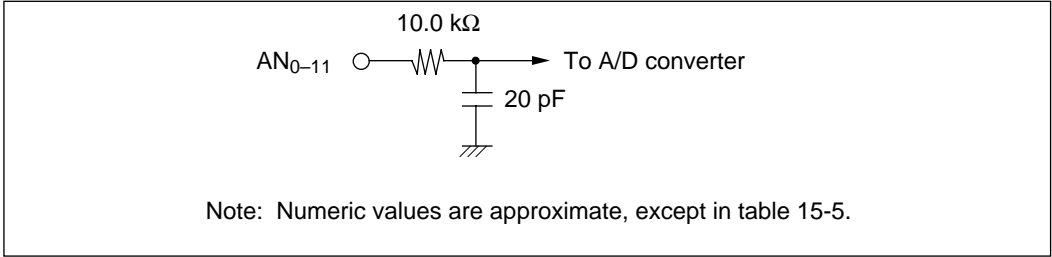
- (4) Note on Board Design:** In board layout, separate the digital circuits from the analog circuits as much as possible. Particularly avoid layouts in which the signal lines of digital circuits cross or closely approach the signal lines of analog circuits. Induction and other effects may cause the analog circuits to operate incorrectly, or may adversely affect the accuracy of A/D conversion.

The analog input signals ( $A_{IN}$ ), analog reference voltage ( $V_{REF}$ ), and analog supply voltage ( $AV_{CC}$ ) must be separated from digital circuits by the analog ground ( $AV_{SS}$ ). The analog ground ( $AV_{SS}$ ) should be connected to a stable digital ground ( $V_{SS}$ ) at one point on the board.

- (5) Note on Noise:** To prevent damage from surges and other abnormal voltages at the analog input pins ( $A_{IN}$ ) and analog reference voltage pin ( $V_{REF}$ ), connect a protection circuit like the one in figure 15-9 between  $AV_{CC}$  and  $AV_{SS}$ . The bypass capacitors connected to  $AV_{CC}$  and  $V_{REF}$  and the filter capacitors connected to  $A_{IN}$  must be connected to  $AV_{SS}$ . If filter capacitors like those in figure 15-9 are connected, the voltage values input to the analog input pins ( $A_{IN}$ ) will be smoothed, which may give rise to error. Also, when A/D conversion is frequently performed with the A/D converter in scan mode, etc., if the current charged or discharged by the analog input capacitance of the H8/539F exceeds the current input via the input impedance  $R_{in}$ , an error will arise in the filter capacitor voltage. The circuit constants should therefore be selected carefully.



**Figure 15-9 Example of Analog Input Protection Circuit**



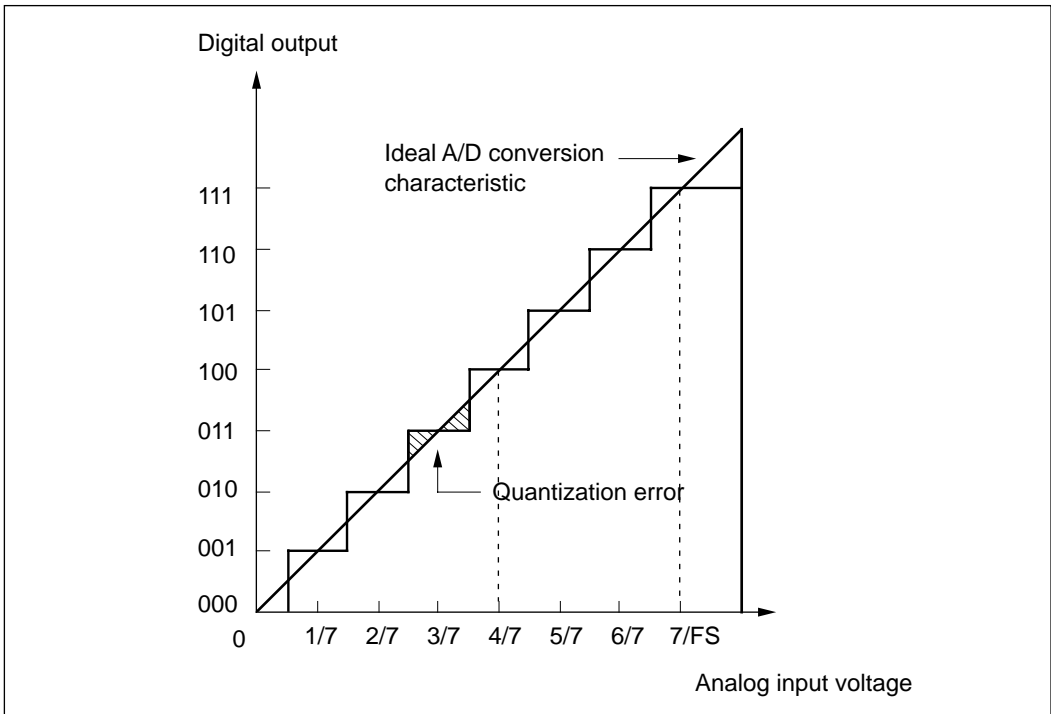
**Figure 15-10 Analog Input Pin Equivalent Circuit**

**Table 15-5 Analog Input Pin Ratings**

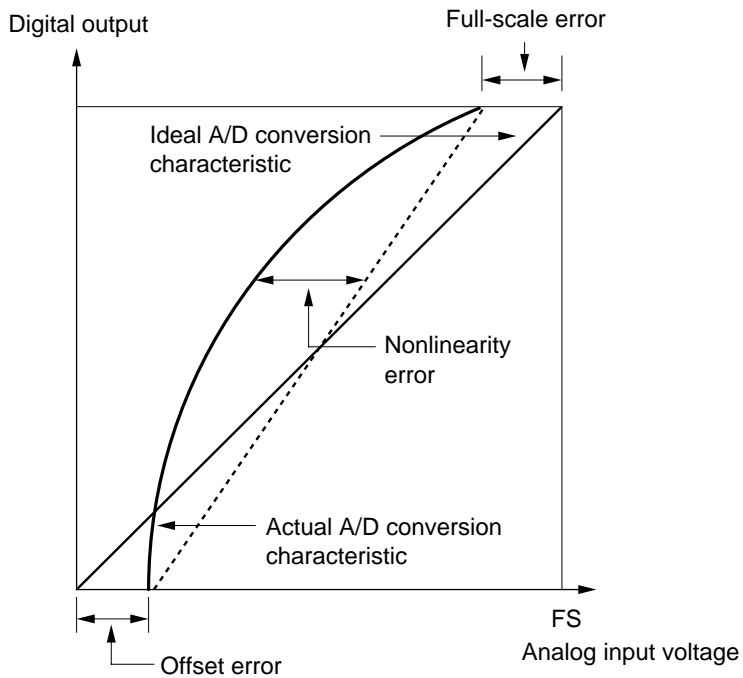
Item		Min	Max	Unit
Analog input capacitance		—	20	pF
Allowable signal-source impedance	$4.5\text{ V} \leq AV_{CC} \leq 5.5\text{ V}$	—	10	kΩ

**(6) A/D Conversion Accuracy Definitions:** A/D conversion accuracy in the H8/539F is defined as follows:

- Resolution: digital output code length of A/D converter
- Offset error: deviation from ideal A/D conversion characteristic of analog input voltage required to raise digital output from minimum voltage value 0000000000 to 0000000001, excluding quantization error (figure 15-12)
- Full-scale error: deviation from ideal A/D conversion characteristic of analog input voltage required to raise digital output from 1111111110 to 1111111111, excluding quantization error (figure 15-12)
- Quantization error: intrinsic error of the A/D converter; 0.5 LSB (figure 15-11)
- Nonlinearity error: deviation from ideal A/D conversion characteristic in range from zero volts to full scale, excluding of offset error, full-scale error, and quantization error.
- Absolute accuracy: deviation of digital value from analog input value, including offset error, full-scale error, quantization error, and nonlinearity error.



**Figure 15-11 A/D Converter Accuracy Definitions (1)**



**Figure 15-12 A/D Converter Accuracy Definitions (2)**



# Section 16 Bus Controller

## 16.1 Overview

The on-chip bus controller (BSC) can dynamically alter the bus width and the length of the bus cycle. When a 16-bit bus mode is selected by the inputs at the mode pins, the bus controller can reserve part of the address space as a byte access area accessed via an eight-bit bus, switch another part from a three-state bus cycle to a high-speed two-state bus cycle, and switch the eight-bit-bus area to 16-bit access.

### 16.1.1 Features

Bus controller features are listed below.

- An eight-bit access area can be defined in the 16-bit bus modes (modes 1, 3, 4, 5\*, and 6\*)

The eight-bit access area consists of addresses greater than the value set in the byte area top register (ARBT). (This area does not include the address set in ARBT, which is the boundary of the word area.) When an address greater than the ARBT value is accessed, only the upper data bus ( $D_{15}$  to  $D_8$ ) is valid. The access is performed with eight-bit bus width. The ARBT setting does not change the bus width of the on-chip ROM, on-chip RAM, and on-chip register areas.

Note: \* Modes 5 and 6 have a 16-bit bus, but when the chip comes out of reset the ARBT and AR3T settings are ignored: the entire external address space is accessed in three states via an eight-bit bus. Software can enable the ARBT and AR3T settings by altering a value in the bus control register (BCR).

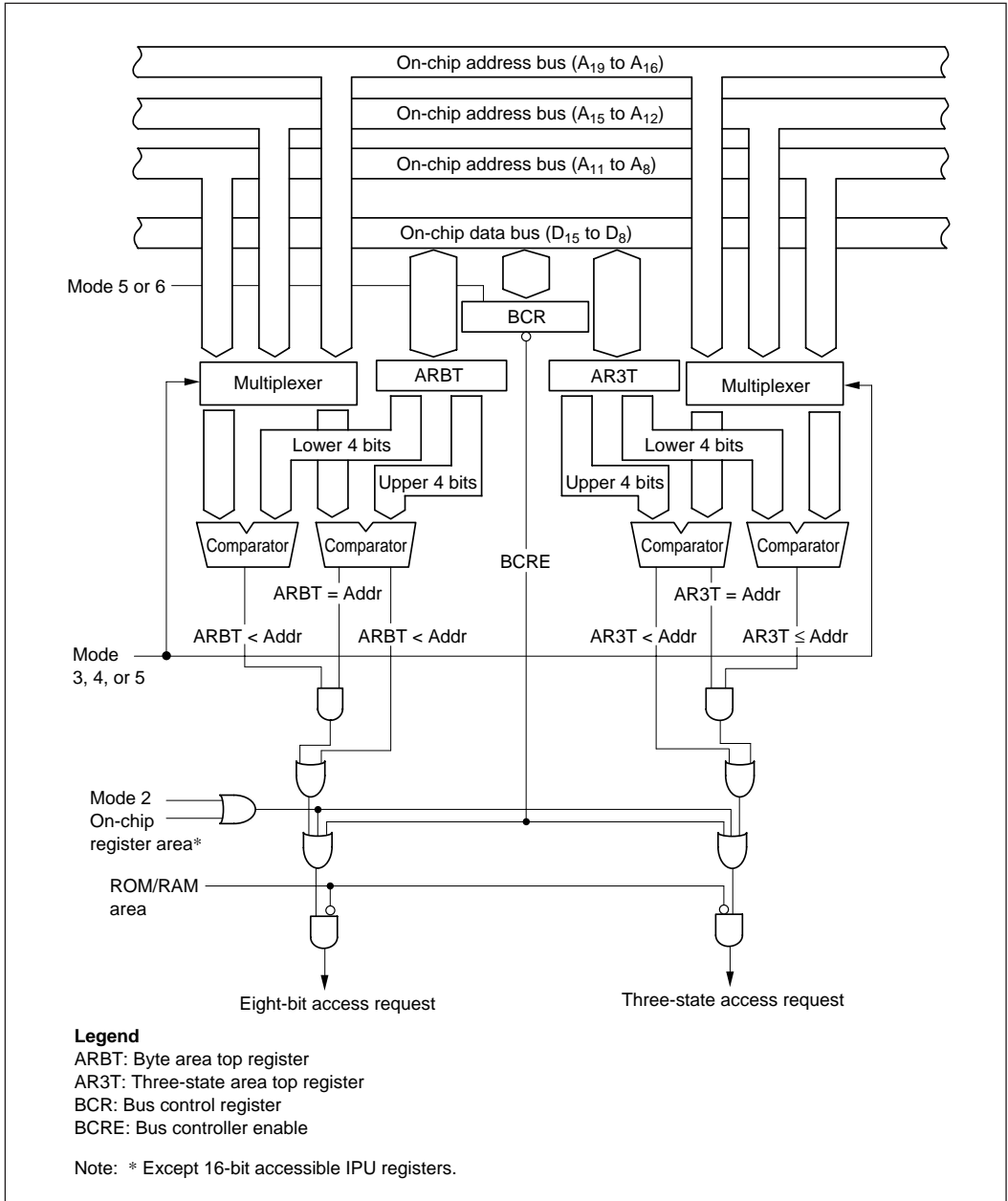
- Two-state access area can be defined

The three-state-access area consists of addresses equal to or greater than the value set in the three-state area top register (AR3T). (The address set in AR3T is included as the boundary of the three-state area.) When addresses less than the AR3T value are accessed, the bus cycle consists of two states. Wait states ( $T_W$ ) cannot be inserted in two-state access. The AR3T setting does not change the bus cycle length of the on-chip ROM, on-chip RAM, and on-chip register areas.

- Areas can be defined in steps of 256 bytes in minimum mode, or 4 kbytes in maximum mode.

## 16.1.2 Block Diagram

Figure 16-1 shows a block diagram of the bus controller.



**Figure 16-1 Bus Controller Block Diagram**

### 16.1.3 Register Configuration

Table 16-1 summarizes the bus controller's registers. The bus controller has three 8-bit registers: a byte area top register (ARBT) that designates the boundary of the word area; a three-state area top register (AR3T) that designates the boundary of the three-state-access address space; and a bus control register (BCR) used to switch the bus width in modes 5 and 6. The H8/500 CPU can always read and write ARBT, AR3T, and BCR.

**Table 16-1 Bus Controller Registers**

Address	Register Name	Abbreviation	R/W	Initial Value
H'FF16	Byte area top register	ARBT	R/W	H'FF
H'FF17	Three-state area top register	AR3T	R/W	H'EE (H'0E)*1
H'FEDF	Bus control register	BCR	R/W	H'BF (H'3F)*2

Notes: 1. H'0E is the initial value in modes 3, 4, and 5. In modes 1, 2, 6, and 7 the initial value is H'EE.  
 2. H'3F is the initial value in modes 5 and 6. In modes 1 to 4 and 7 the initial value is H'BF.

## 16.2 Register Descriptions

### 16.2.1 Byte Area Top Register

The byte area top register (ARBT) specifies the boundary address that separates the area accessed with 16-bit bus width from the area accessed using only the upper eight bits of the 16-bit bus. The address set in ARBT is the word area boundary: the last address accessed with 16-bit bus width.

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bus controller controls the H8/500 CPU so that external addresses exceeding the ARBT value are accessed with eight-bit bus width.

In expanded maximum mode, the ARBT value is treated as bits  $A_{19}$  to  $A_{12}$  (the upper eight bits) of the word area boundary address. The word area boundary can be set in minimum 4-kbyte steps. In expanded maximum mode, addresses H'00000 to H'00FFF are always a word access area.

In expanded minimum mode, the ARBT value is treated as bits  $A_{15}$  to  $A_8$  (the upper eight bits) of the word area boundary address. The word area boundary can be set in minimum 256-byte steps. In expanded minimum mode, addresses H'0000 to H'00FF are always a word access area.

The ARBT setting applies only to external addresses. It cannot change the bus width of the on-chip ROM or RAM or on-chip register areas. In mode 2 the ARBT setting is ignored: the external address bus has a fixed eight-bit width. In modes 5 and 6 the ARBT setting is ignored until the BCRE bit is set to 1 in the bus control register (BCR).

ARBT is initialized to H'FF by a reset and in hardware standby mode. ARBT is not initialized in software standby mode.

### 16.2.2 Three-State Area Top Register

The three-state area top register (AR3T) specifies the boundary address that separates the area accessed in two states from the area accessed in three states. The address set in AR3T is the three-state area boundary: the first address accessed in three states.

Bit	7	6	5	4	3	2	1	0
Initial value	1 (0)*	1 (0)*	1 (0)*	0	1	1	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: \* Modes 3 to 5

The bus controller controls the H8/500 CPU so that external addresses equal to or greater than the ARBT value are accessed in three states. Wait states cannot be inserted into the two-state-access area.

In expanded maximum mode, the AR3T value is treated as bits A<sub>19</sub> to A<sub>12</sub> (the upper eight bits) of the three-state area boundary address. The three-state area boundary can be set in minimum 4-kbyte steps. In expanded maximum mode, addresses H'FF000 to H'FFFFFF are always a three-state-access area.

In expanded minimum mode, the AR3T value is treated as bits A<sub>15</sub> to A<sub>8</sub> (the upper eight bits) of the three-state area boundary address. The three-state area boundary can be set in minimum 256-byte steps. In expanded minimum mode, addresses H'FF00 to H'FFFF are always a three-state-access area.

The AR3T setting applies only to external addresses. It cannot change the bus cycle length of the on-chip ROM or RAM or on-chip register areas. In mode 2 the AR3T setting is ignored: the external address space is always a three-state-access area. In modes 5 and 6 the AR3T setting is ignored until the BCRE bit is set to 1 in the bus control register (BCR).

AR3T is initialized to H'EE (modes 1, 2, 6, and 7) or H'0E (modes 3 to 5) by a reset and in hardware standby mode. ARBT is not initialized in software standby mode.

### 16.2.3 Bus Control Register

The bus control register (BCR) enables or disables the bus controller's bus control functions in modes 5 and 6, and enables or disables on-chip I/O port functions.

Bit	7	6	5	4	3	2	1	0
	BCRE	0P3T	—	P9AE	EXIOP	PCRE	PBCE	P12E
Initial value	0 (1)*	0	1	1	1	1	1	1
R/W	R/W (R)*	R/W	—	R/W	R/W	R/W	R/W	R/W

<p>Ports 1 and 2 enable Enables and disables reading and writing of ports 1 and 2</p>	<p>Ports B and C enable Enables and disables reading and writing of ports B and C</p>
<p><u>Reserved bit</u></p>	<p><u>Pull-up transistor control register enable</u> Enables and disables reading and writing of port B and C pull-up transistor control registers</p>
<p><u>Expanded I/O ports</u> Allocates H'0FE9C to H'0FE9F as external addresses</p>	<p><u>Ports 9 and A enable</u> Enables and disables reading and writing of ports 9 and A</p>
<p><u>Zero page three-state</u> Forces three-state access to all addresses in page 0</p>	<p><u>Bus controller enable</u> Enables and disables bus control functions of the bus controller</p>

Note: \* In modes 1, 2, 3, 4, and 7.

When the bus controller enable bit (BCRE) is set to 1, the bus controller controls the bus according to the values in ARBT and AR3T. As an exception, when the zero page three-state bit (0P3T; bit 6) is set to 1, all external addresses in page 0 are placed in the three-state-access area regardless of the AR3T setting.

Bits 4, 2, 1, and 0 enable or disable reading and writing of on-chip I/O ports. If one of these bits is cleared to 0, the corresponding on-chip I/O ports cannot be accessed. The port addresses become part of the external eight-bit three-state-access area instead.

Bit 3 is for I/O port expansion. When this bit is cleared to 0, H'0FE9C to H'0FE9F become part of the external eight-bit three-state-access area.

For precautions on modifying the BCR value, see section 16.4, “Usage Notes.”

**(1) Bit 7—Bus Controller Enable (BCRE):** Enables or disables bus control functions using the values in ARBT and AR3T in modes 5 and 6.

#### Bit 7

BCRE	Description
0	The H8/500 CPU accesses all external addresses in three states using an eight-bit bus* (Initial value in modes 5 and 6) This bit cannot be cleared to 0 in modes 1 to 4 and 7.
1	The H8/500 CPU accesses external addresses according to the ARBT and AR3T settings (Initial value in modes 1 to 4 and 7; cannot be cleared to 0)

Note: \* Access is performed using only the upper eight bits (D<sub>15</sub> to D<sub>8</sub>) of the 16-bit bus.

**(2) Bit 6—Zero Page Three-State (0P3T):** Selects three-state access for all external addresses in page 0, regardless of the AR3T setting.

#### Bit 6

0P3T	Description
0	The H8/500 CPU accesses external addresses according to the ARBT and AR3T settings (Initial value)
1	The H8/500 CPU accesses external addresses according to the ARBT and AR3T settings except in page 0, where three-state access is selected regardless of the AR3T setting*

Note: \* In mode 7 there is no external address space, so the 0P3T value has no meaning.

**(3) Bit 5—Reserved:** Read-only bit, always read as 1. Reserved for future expansion.

**(4) Bit 4—Port 9 and A Enable (P9AE):** Enables or disables reading and writing of ports 9 and A, allowing these I/O ports to be reconfigured off-chip.

**Bit 4**

<b>P9AE</b>	<b>Description</b>
0	On-chip I/O ports 9 and A cannot be written or read The DR and DDR addresses of ports 9 and A (H'0FE90 to H'0FE93) become part of the external eight-bit three-state-access area.*
1	On-chip I/O ports 9 and A can be written and read (Initial value)

Note: \* Cannot be cleared to 0 in mode 7.

For details see section 16.3.3, “I/O Port Expansion Function.”

**(5) Bit 3—Expanded I/O Ports (EXIOP):** Enables or disables expansion of I/O ports, allowing I/O ports to be configured off-chip.

**Bit 3**

<b>EXIOP</b>	<b>Description</b>
0	External I/O ports can be written and read H'0FE9C to H'0FE9F become part of the external eight-bit three-state-access area.*
1	External I/O ports cannot be written or read (Initial value)

Note: \* Cannot be cleared to 0 in mode 7.

For details see section 16.3.3, “I/O Port Expansion Function.”

**(6) Bit 2—Pull-Up Transistor Control Register Enable (PCRE):** Enables or disables reading and writing of port B and C pull-up transistor control registers (PBPCR and PCPCR).

**Bit 2**

<b>PCRE</b>	<b>Description</b>
0	Port B and C pull-up transistor control registers (PBPCR and PCPCR) cannot be written or read PBPCR and PCPCR addresses (H'0FE98 to H'0FE9B) become part of the external eight-bit three-state-access area.*
1	Port B and C pull-up transistor control registers (PBPCR and PCPCR) can be written and read (Initial value)

Note: \* Cannot be cleared to 0 in mode 7.

For details see section 16.3.3, “I/O Port Expansion Function.”

**(7) Bit 1—Port B and C Enable (PBCE):** Enables or disables reading and writing of ports B and C, allowing these I/O ports to be reconfigured off-chip.

**Bit 1**

<b>PBCE</b>	<b>Description</b>
0	On-chip I/O ports B and C cannot be written or read The DR and DDR addresses of ports B and C (H'0FE94 to H'0FE97) become part of the external eight-bit three-state-access area.*
1	On-chip I/O ports B and C can be written and read (Initial value)

Note: \* Cannot be cleared to 0 in mode 7.

For details see section 16.3.3, “I/O Port Expansion Function.”

**(8) Bit 0—Port 1 and 2 Enable (P12E):** Enables or disables reading and writing of ports 1 and 2, allowing these I/O ports to be reconfigured off-chip.

**Bit 0**

<b>P12E</b>	<b>Description</b>
0	On-chip I/O ports 1 and 2 cannot be written or read The DR and DDR addresses of ports 1 and 2 (H'0FE80 to H'0FE83) become part of the external eight-bit three-state-access area.*
1	On-chip I/O ports 1 and 2 can be written and read (Initial value)

Note: \* Cannot be cleared to 0 in mode 7.

For details see section 16.3.3, “I/O Port Expansion Function.”

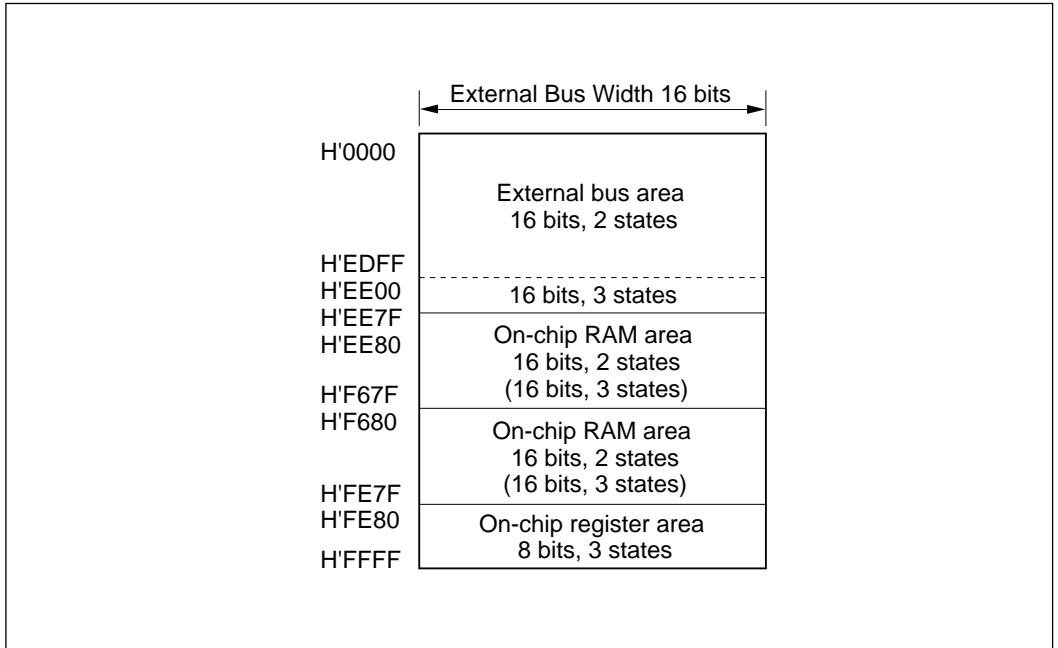


## 16.3 Operation

### 16.3.1 Operation after Reset in Each Mode

Figures 16-2 to 16-8 illustrate operation in each mode after a reset.

(1) **Mode 1:** The external data bus space. H'0000 to H'EDFF are a 16-bit two-state-access area. H'EE00 to H'FE7F are a 16-bit three-state-access area. When the on-chip RAM is enabled, however, the on-chip RAM area is a 16-bit two-state-access area.

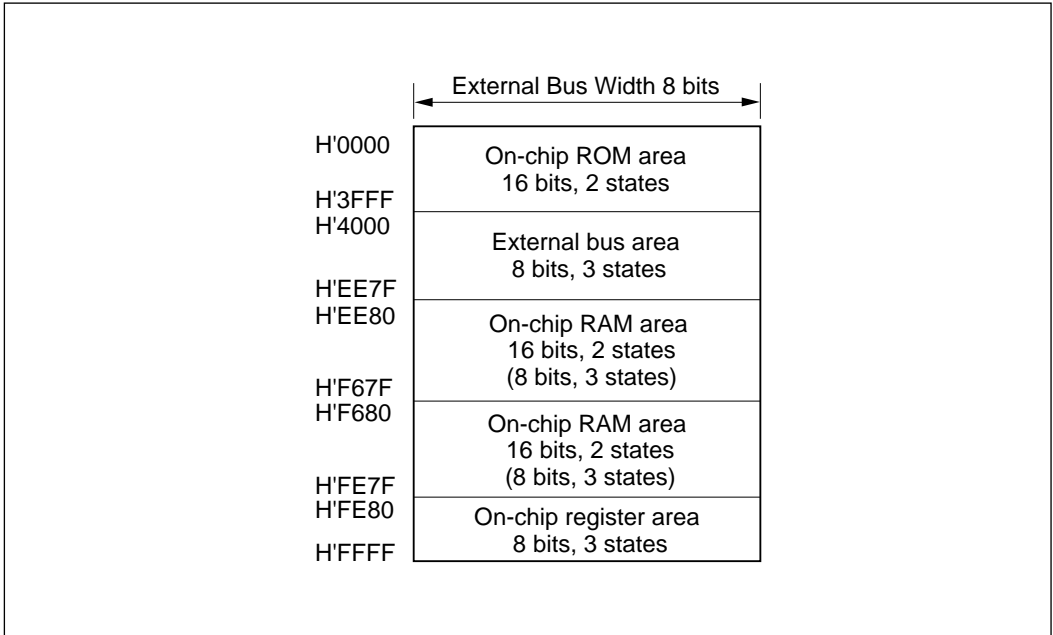


**Figure 16-2 Bus Width and Bus Cycle Length after Reset (Mode 1)**

## (2) Mode 2

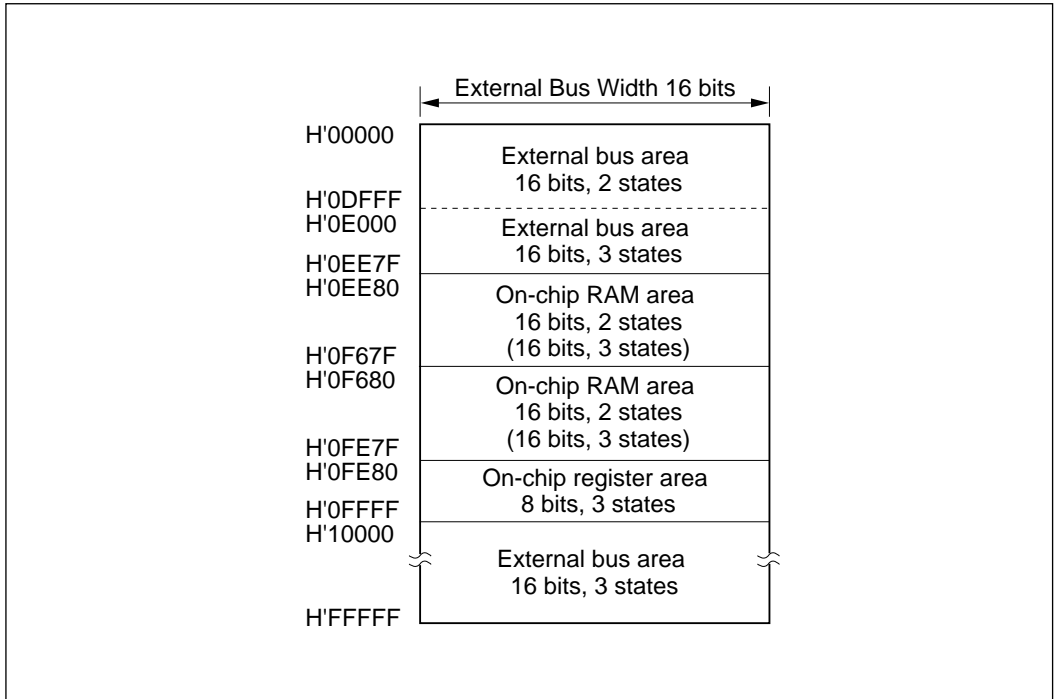
The external data bus space is eight bits wide. H'0000 to H'3FFF (on-chip ROM) are a 16-bit two-state-access area. H'4000 to H'FE7F are an eight-bit three-state-access area.

When the on-chip RAM is enabled, however, the on-chip RAM area is a 16-bit two-state-access area.



**Figure 16-3 Bus Width and Bus Cycle Length after Reset (Mode 2)**

**(3) Mode 3:** The external data bus space. H'00000 to H'0DFFF are a 16-bit two-state-access area. H'0E000 to H'0FE7F and H'10000 to H'FFFFFF are a 16-bit three-state-access area. When the on-chip RAM is enabled, however, the on-chip RAM area is a 16-bit two-state-access area.

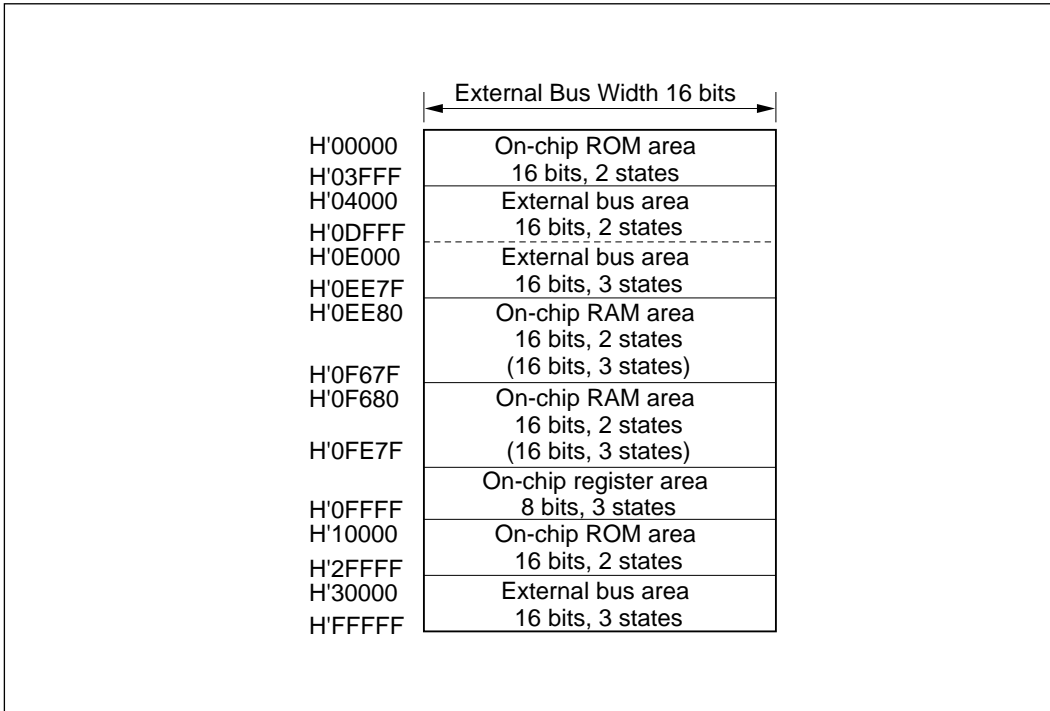


**Figure 16-4 Bus Width and Bus Cycle Length after Reset (Mode 3)**

#### (4) Mode 4

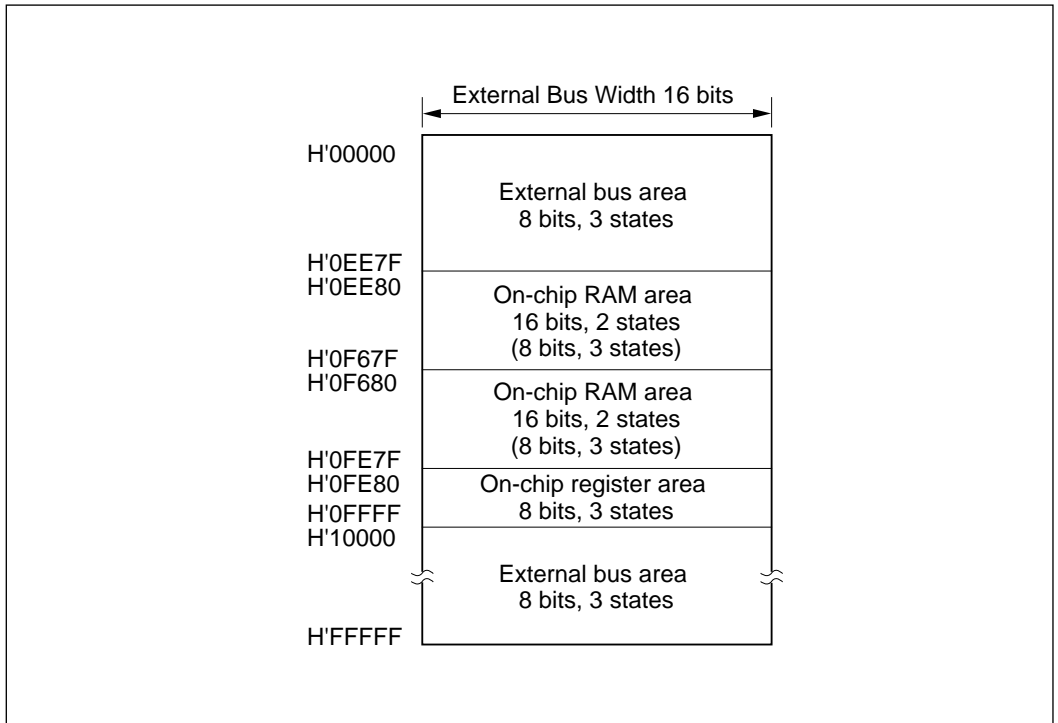
The external data bus space is 16 bits wide. H'00000 to H'03FFF and H'10000 to H'2FFFF (on-chip ROM) are 16-bit two-state-access areas. H'04000 to H'0DFFF is a 16-bit two-state access area. H'0E000 to H'0FE7F and H'30000 to H'FFFFFF are a 16-bit three-state-access area.

When the on-chip RAM is enabled, however, the on-chip RAM area is a 16-bit two-state-access area.



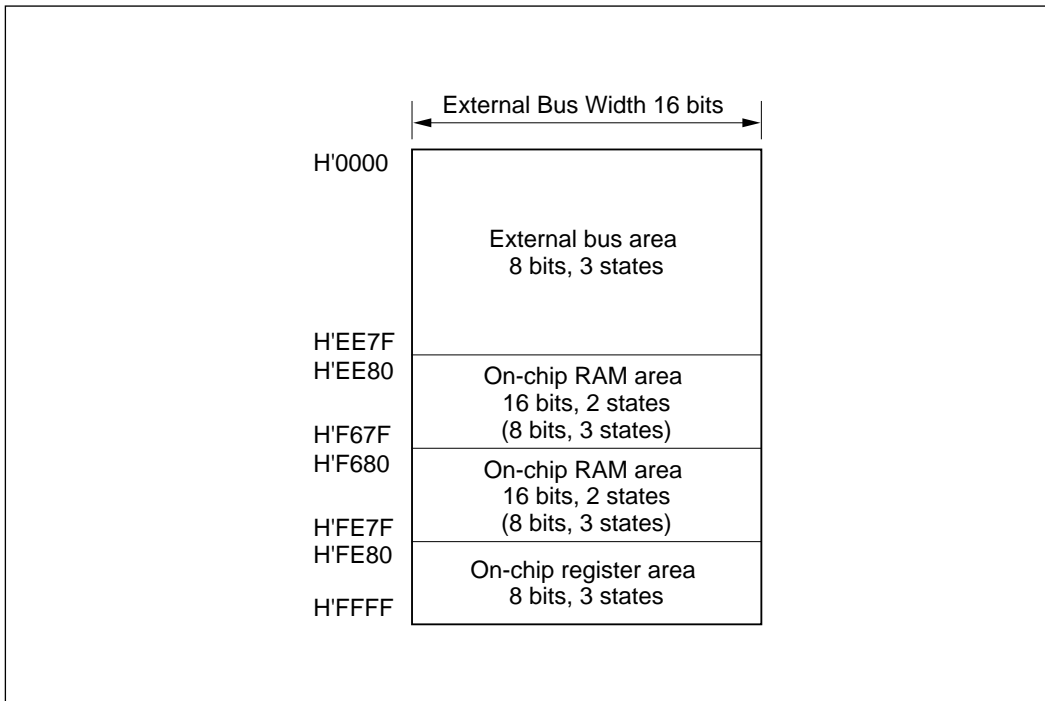
**Figure 16-5 Bus Width and Bus Cycle Length after Reset (Mode 4)**

**(5) Mode 5:** The external data bus space uses a 16-bit bus width. After a reset, H'00000 to H'FFFFFF are an eight-bit three-state-access area because BCRE = 0 in the bus control register (BCR). In this case, the upper half of the data bus (D15 to D8) is enabled (see Table 16-2 (2)). When the on-chip RAM is enabled, however, the on-chip RAM area is a 16-bit two-state-access area.



**Figure 16-6 Bus Width and Bus Cycle Length after Reset (Mode 5)**

**(6) Mode 6:** The external data bus space. H'0000 to H'FE80 are an eight-bit three-state-access area (BCRE = 0 in BCR). In this case, the upper half of the data bus (D15 to D8) is enabled (see Table 16-2 (2)). When the on-chip RAM is enabled, however, the on-chip RAM area is a 16-bit two-state-access area.

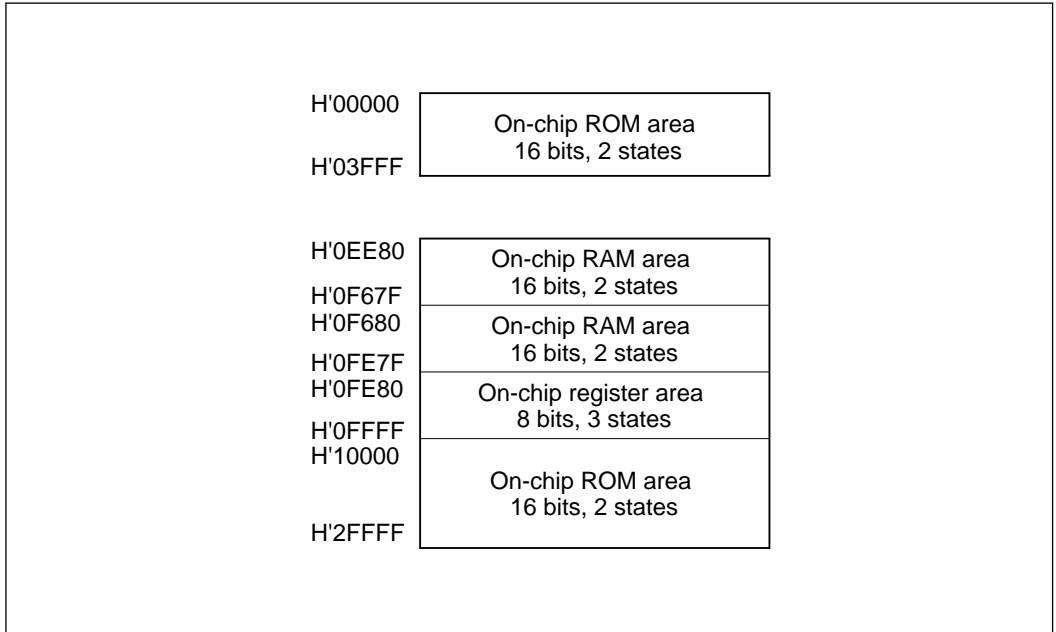


**Figure 16-7 Bus Width and Bus Cycle Length after Reset (Mode 6)**

(7) **Mode 7:** There is no external bus.

H'00000 to H'03FFF and H'10000 to H'2FFFF (on-chip ROM) are a 16-bit two-state-access area.

When the on-chip RAM is enabled, the on-chip RAM area is also a 16-bit two-state-access area.



**Figure 16-8 Bus Width and Bus Cycle Length after Reset (Mode 7)**

### 16.3.2 Timing of Changes in Bus Areas and Bus Size

Changes in the bus areas and bus size take effect in the next bus cycle after the write cycle to ARBT or AR3T.

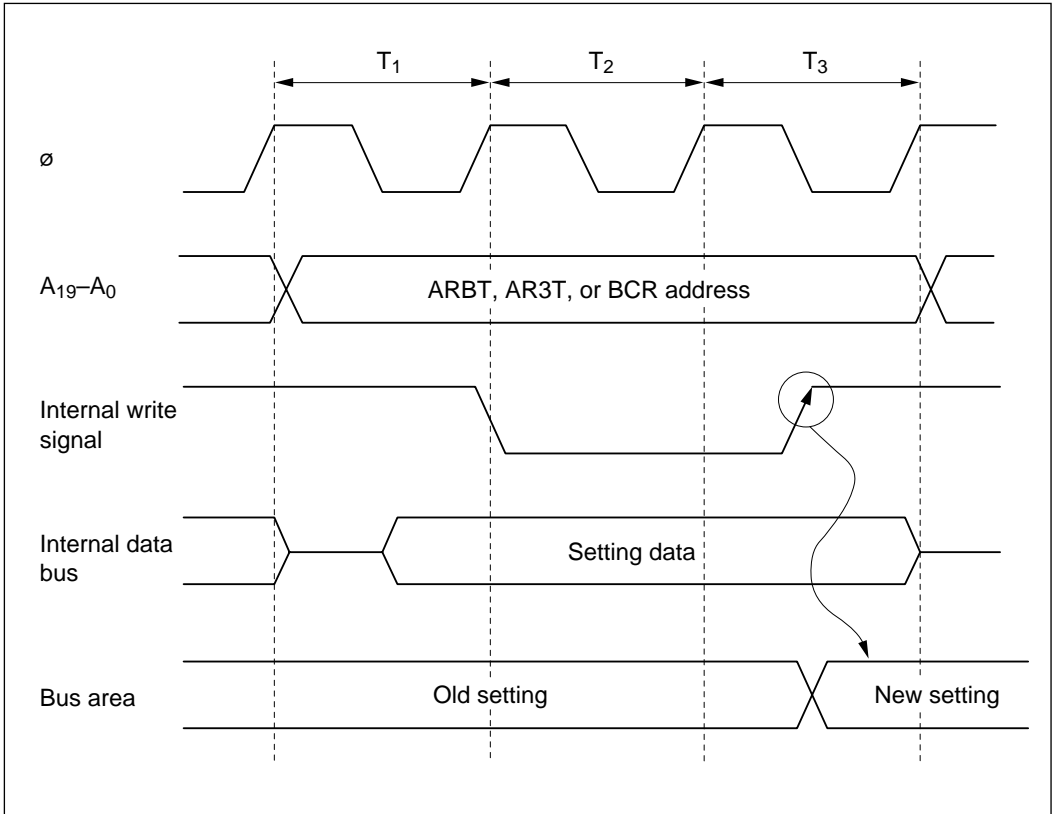
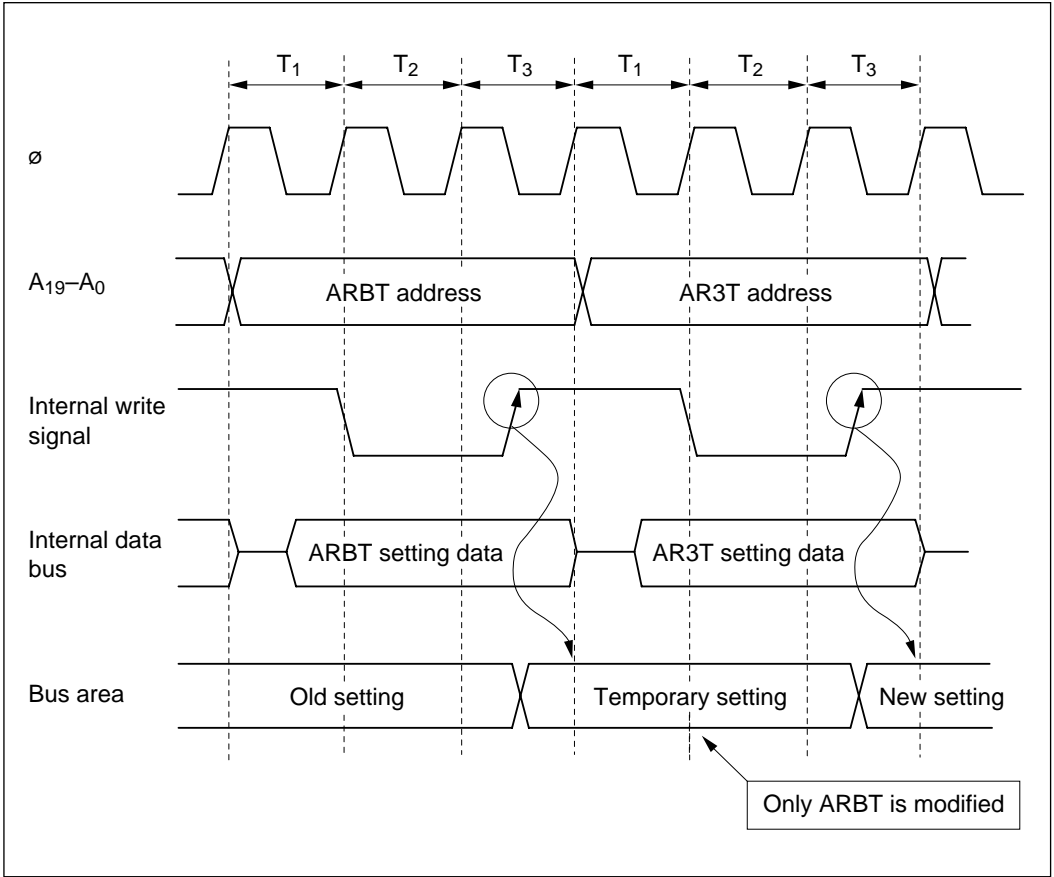


Figure 16-9 Timing of Changes in Bus Controller Settings (Byte Write)





**Figure 16-10 Timing of Changes in Bus Controller Settings (Word Write)**

### 16.3.3 I/O Port Expansion Function

Bus control register bits 4 to 0 can be set for I/O port expansion. This function enables ports 1, 2, A, B, and C, which cannot be used in the bus expansion modes (modes 1 to 6), to be reconfigured off-chip. Figure 16-11 shows an example of I/O port reconfiguration.

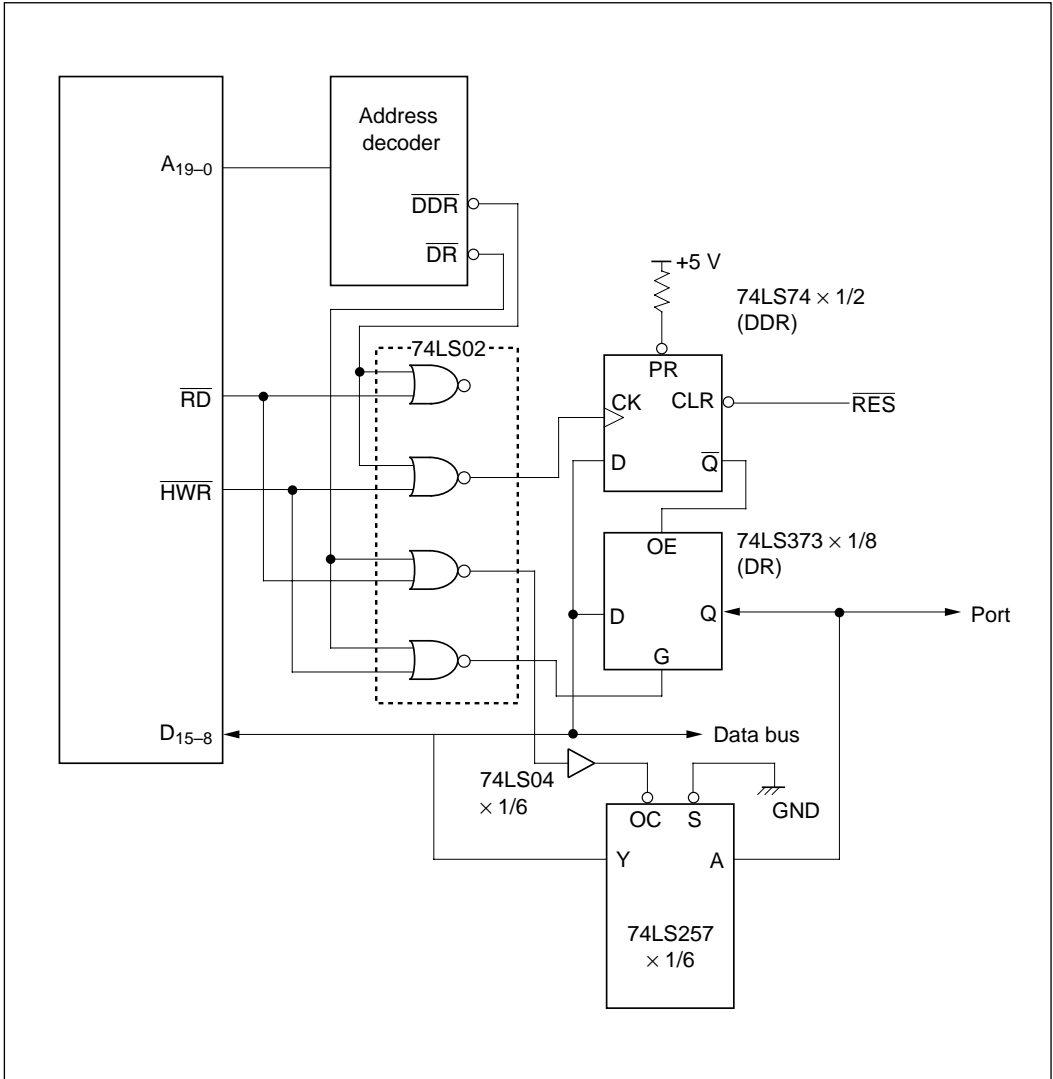


Figure 16-11 Example of I/O Port Reconfiguration (1 Bit)

## 16.4 Usage Notes

When using the bus controller, note the following points:

**(1) Restrictions on AR3T and ARBT Settings:** AR3T and ARBT settings should satisfy equation (1).

$$AR3T \leq ARBT + 1 \quad \dots\dots(1)$$

No eight-bit, two-state-access area is defined for the H8/539F. If  $AR3T > ARBT + 1$ , eight-bit three-state access is performed.

**(2) Possible Partitionings of the Address Space:** The address space can be partitioned in eight ways as follows:

1. Two areas: 16 bits, two states; 16 bits, three states
2. Two areas: 16 bits, two states; eight bits, three states
3. Two areas: 16 bits, three states; eight bits, three states
4. Three areas: 16 bits, two states; 16 bits, three states; eight bits, three states
5. One area: eight bits, three states \*1
6. Three areas: 16 bits, three states (page 0)\*2; 16 bits, two states; 16 bits, three states
7. Three areas: 16 bits, three states (page 0)\*2; 16 bits, two states; 8 bits, three states
8. Four areas: 16 bits, three states (page 0)\*2; 16 bits, two states; 16 bits, three states; eight bits, three states

Notes: 1. Possible only in modes 5 and 6 when  $BCRE = 0$  in the bus control register (BCR).  
2. Set by the  $OP3T$  bit in BCR.

**(3) Modification of ARBT, AR3T, and BCR:** When ARBT, AR3T, and BCR settings are modified, an invalid bus area may be created temporarily. This may prevent normal program execution. Crashes can be avoided by one of the following methods:

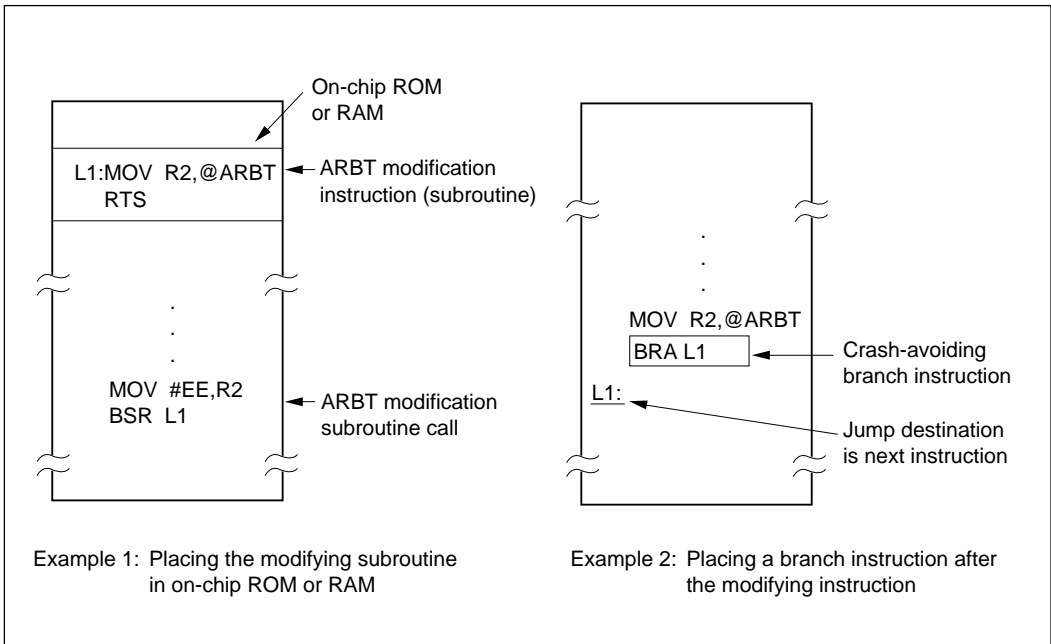
1. Place routines that modify ARBT, AR3T, and BCR in on-chip ROM or RAM.

Perform the modification in an area that is not affected by the ARBT, AR3T, and BCR settings. The modification can be followed by a jump to any area without crashing. (Example 1)

2. Place a branch instruction after the instruction that modifies ARBT, AR3T, or BCR.

After the write to ARBT, AR3T, or BCR,\* the instruction fetch from the temporary invalid bus area is cleared by execution of the branch instruction, thus preventing a crash. (Example 2)

Note: \* To modify both ARBT and AR3T simultaneously, a word access instruction is recommended.



**Figure 16-12 Example of Program Structure for Modifying ARBT, AR3T, and BCR**

**(4) Access Types and Operation of Data Bus and Control Signals:** Table 16-2 indicates how the data bus and control signals operate in various types of access.

**Table 16-2 (1) Data Bus and Control Signal Operation in Various Types of Access (Mode 2)**

No.	External Bus Width	Instruction Designations			Data Bus			Control Signals		
		Operand Address	Operand Size	Access Direction	A <sub>0</sub>	D <sub>15</sub> to D <sub>8</sub>	D <sub>7</sub> to D <sub>0</sub>	RD	HWR	LWR
1	8 bits	Byte area	Byte	Write	0	Output	Not used (port)	H	L	H
2	Write			1	Output	H		L	H	
3	Read			0	Input	L		H	H	
4	Read			1	Input	L		H	H	
5		Word	Word	Write	0	Output	H	L	H	
	Write			1	Output	H	L	H		
6	Read			0	Input	L	H	H		
	Read			1	Input	L	H	H		

Notes: 1. How to read the table:

- 1) External Bus width: external bus width determined by the operating mode.
  - 2) Operand address: area containing the operand address specified in the instruction.  
Examples: ARBT > operand address: byte area  
ARBT < operand address: word area
  - 3) Operand size: size of operand specified in the instruction.  
Examples: MOV.B: byte size  
MOV.W: word size
  - 4) Access direction: as below.  
Examples: MOV.B Rn, <EA>: write (CPU → <EA>)  
MOV.B <EA>, Rn: read (<EA> → CPU)
2. When a byte area is addressed by an instruction with word operand size, the CPU accesses memory twice, accessing the even byte first, then the odd byte. Instructions that specify word-size operands should always specify an even operand address.

**Table 16-2 (2) Data Bus and Control Signal Operation in Various Types of Access  
(Modes 1, 3, 4, 5, 6)**

No.	External Bus Width	Instruction Designations			Data Bus			Control Signals			
		Operand Address	Operand Size	Access Direction	A <sub>0</sub>	D <sub>15</sub> to D <sub>8</sub>	D <sub>7</sub> to D <sub>0</sub>	$\overline{\text{RD}}$	$\overline{\text{HWR}}$	$\overline{\text{LWR}}$	
1	16 bits* Byte area* (Modes 5 and 6, after reset)		Byte	Write	0	Output	High impedance	H	L	H	
2					1	Output	High impedance	H	L	H	
3				Read	0	Input	Don't care	L	H	H	
4					1	Input	Don't care	L	H	H	
5				Word	Write	0	Output	High impedance	H	L	H
						1	Output	High impedance	H	L	H
6	Read	0	Input	Don't care	L	H	H				
		1	Input	Don't care	L	H	H				

Notes: 1. How to read the table:

- 1) External bus width: external bus width determined by the operating mode.
  - 2) Operand address: area containing the operand address specified in the instruction.  
Examples: ARBT > operand address: byte area  
ARBT < operand address: word area
  - 3) Operand size: size of operand specified in the instruction.  
Examples: MOV.B: byte size  
MOV.W: word size
  - 4) Access direction: as below.  
Examples: MOV.B Rn, <EA>: write (CPU → <EA>)  
MOV.B <EA>, Rn: read (<EA> → CPU)
2. When a byte area is addressed by an instruction with word operand size, the CPU accesses memory twice, accessing the even byte first, then the odd byte. Instructions that specify word-size operands should always specify an even operand address.
- \* Modes 5 and 6 have a 16-bit bus width, but an 8-bit bus width is set after a reset.

**Table 16-2 (3) Data Bus and Control Signal Operation in Various Types of Access  
(Modes 1, 3, 4, 5, 6)**

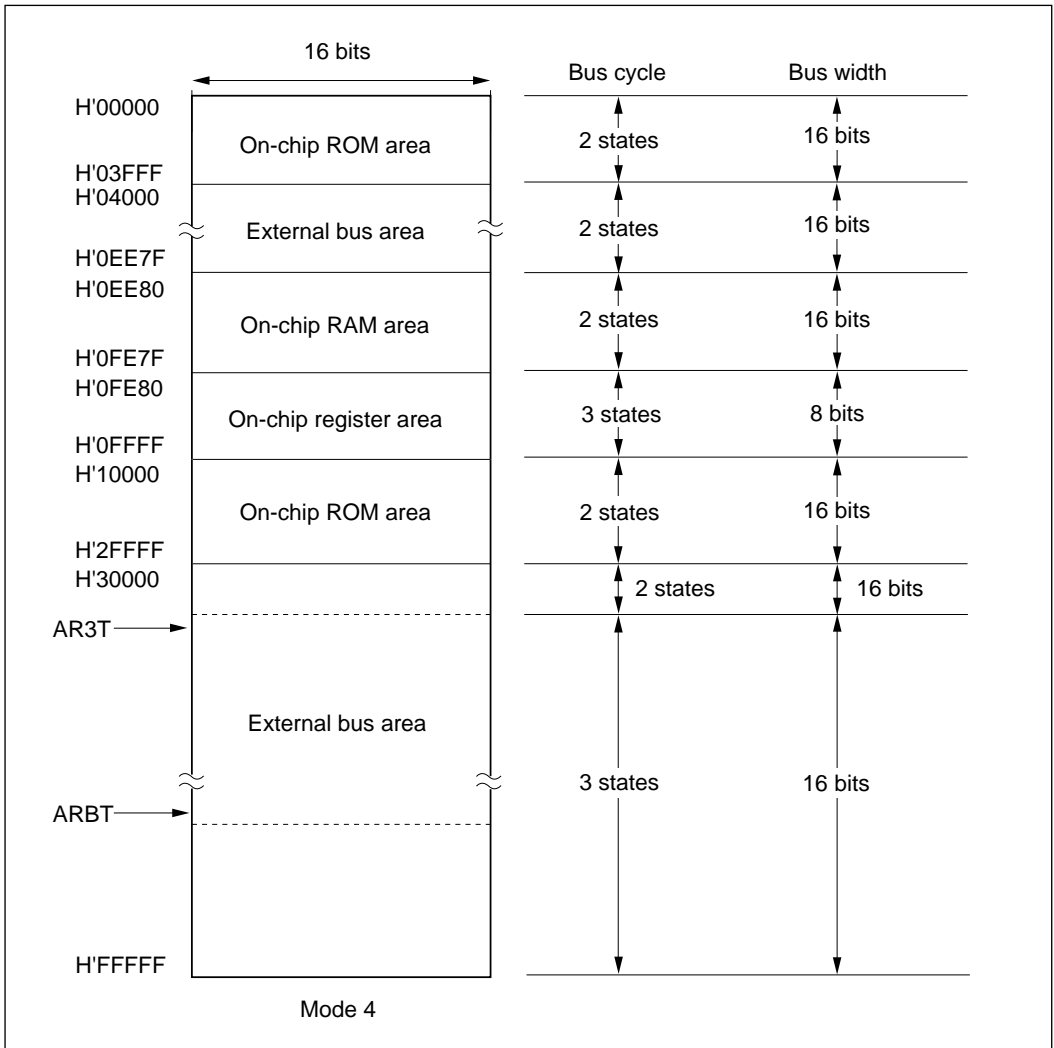
No.	External Bus Width	Instruction Designations			Data Bus			Control Signals		
		Operand Address	Operand Size	Access Direction	A <sub>0</sub>	D <sub>15</sub> to D <sub>8</sub>	D <sub>7</sub> to D <sub>0</sub>	$\overline{\text{RD}}$	$\overline{\text{HWR}}$	$\overline{\text{LWR}}$
1	16 bits	Word area	Byte	Write	0	Output	Dummy data	H	L	H
2	1				Dummy data	Output	H	H	L	
3				Read	0	Input	Don't care	L	H	H
4					1	Don't care	Input	L	H	H
5		Word	Write	0	Output	Output	H	L	L	
				1	—	—	—	—	—	
6				Read	0	Input	Input	L	H	H
					1	—	—	—	—	—

Notes: 1. How to read the table:

- 1) External bus width: external bus width determined by the operating mode.
  - 2) Operand address: area containing the operand address specified in the instruction.  
Examples: ARBT > operand address: byte area  
ARBT < operand address: word area
  - 3) Operand size: size of operand specified in the instruction.  
Examples: MOV.B: byte size  
MOV.W: word size
  - 4) Access direction: as below.  
Examples: MOV.B Rn, <EA>: write (CPU → <EA>)  
MOV.B <EA>, Rn: read (<EA> → CPU)
2. Instructions that specify word-size operands should always specify an even operand address.

Figures 16-13 and 16-14 show examples of usage of the bus controller in mode 4.

1.  $AR3T \leq ARBT + 1$



**Figure 16-13 Example of Use of Bus Controller (Mode 4)**



2.  $AR3T > ARBT + 1$

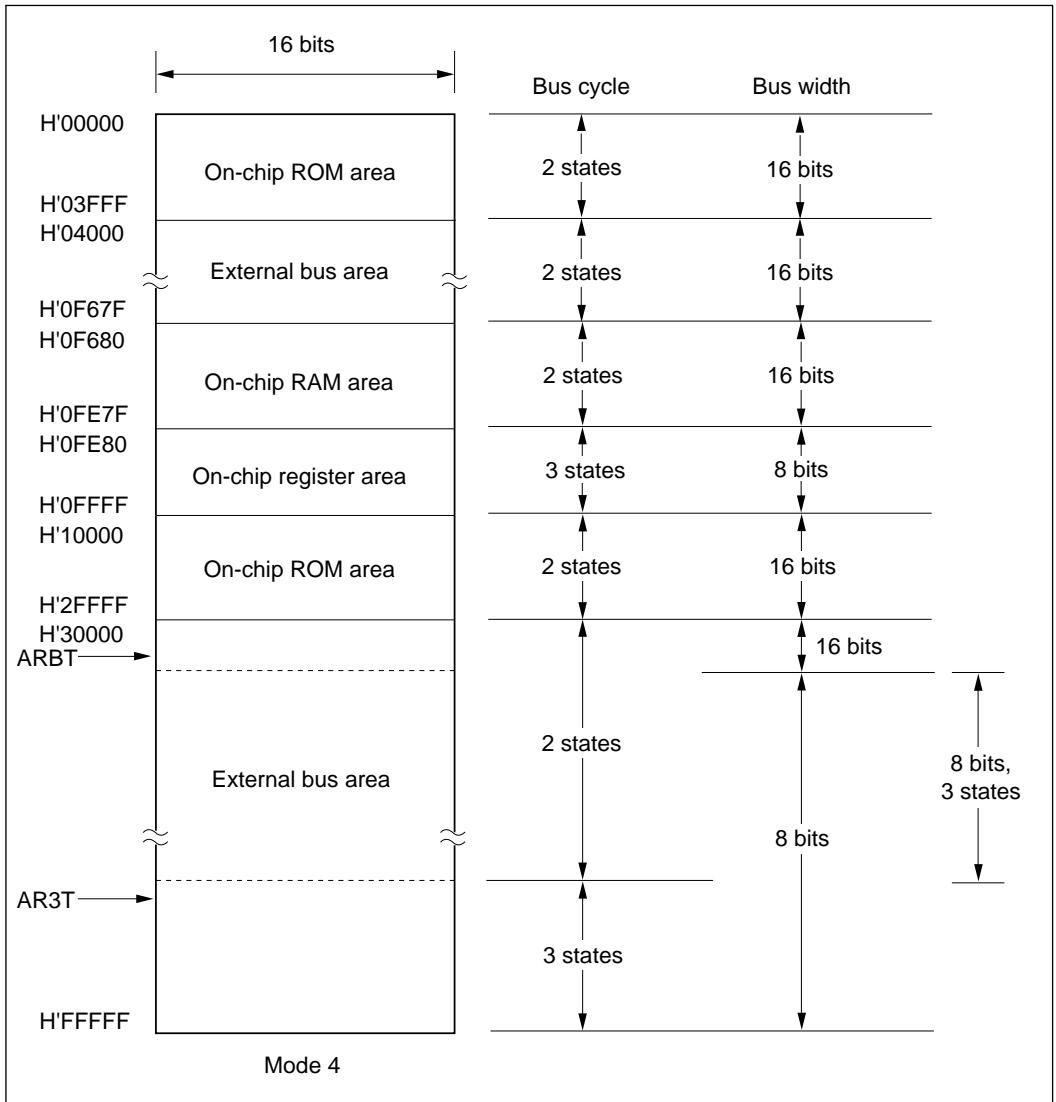


Figure 16-14 Example of Use of Bus Controller (Mode 4)

# Section 17 RAM (H8/539F)

## 17.1 Overview

The H8/539F has 4 kbytes of on-chip static RAM. The RAM is connected to the H8/500 CPU by a 16-bit data bus. The H8/500 CPU accesses both byte data and word data in two states, making the RAM suitable for rapid data transfer and high-speed computation.

The on-chip RAM is assigned to addresses H'EE80 to H'FE7F. The RAM control register (RAMCR) enables this area to be switched between on-chip RAM and external memory.

### 17.1.1 Block Diagram

Figure 17-1 shows a block diagram of the on-chip RAM.

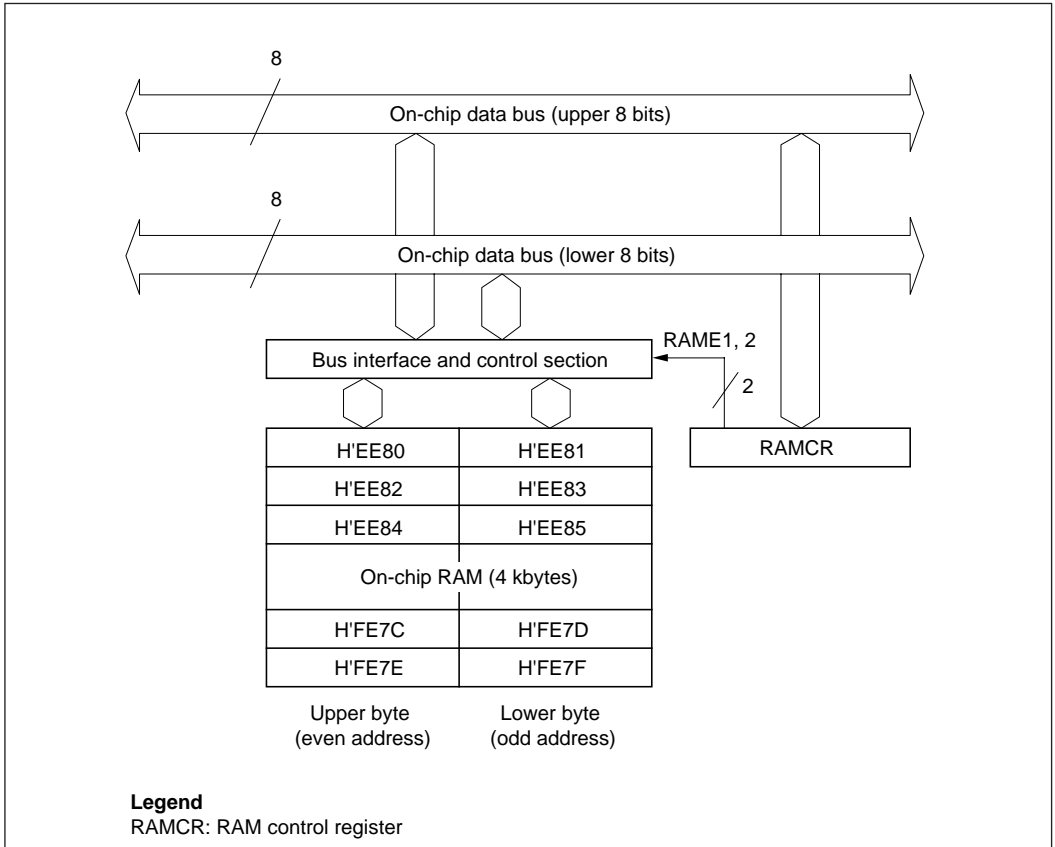


Figure 17-1 RAM Block Diagram

### 17.1.2 Register Configuration

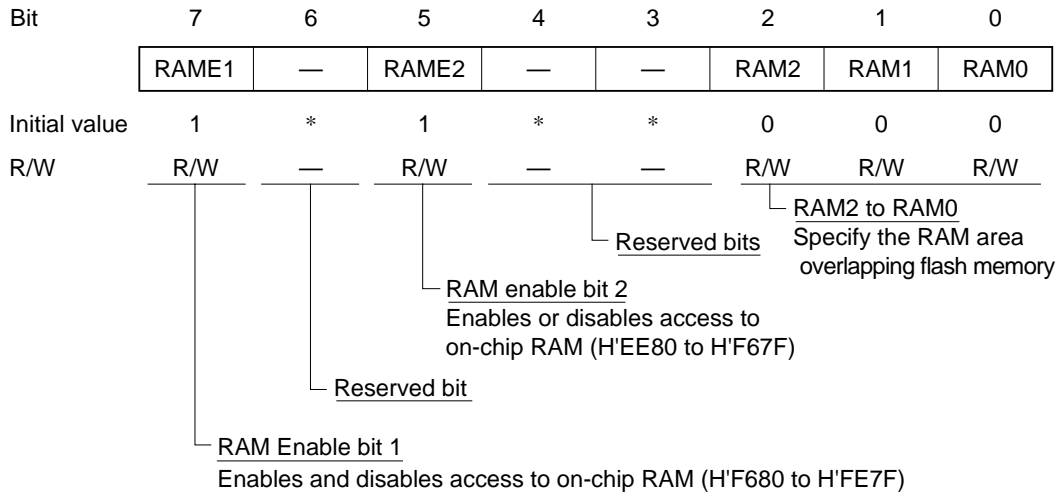
The RAM is controlled by the RAM control register (RAMCR). Table 17-1 gives the address and initial value of RAMCR.

**Table 17-1 RAM Control Register**

Address	Register Name	Abbreviation	R/W	Initial Value
H'FF15	RAM control register	RAMCR	R/W	Undetermined

### 17.2 RAM Control Register

The RAM control register (RAMCR) enables or disables access to the on-chip RAM and controls RAM area overlapping. For details of RAM area overlapping, see section 18.2.4, “RAM Control Register (RAMCR).”



Note: \* Bit 6, 4 and 3 have undetermined values when read.

**(1) Bits 7 and 5—RAM Enable 1 and 2 (RAME1, RAME2):** These bits enable or disable access to on-chip RAM.

#### Bit 7

RAME1	Description
0	On-chip RAM (H'F680 to H'FE7F) cannot be accessed
1	On-chip RAM (H'F680 to H'FE7F) can be accessed (Initial value)

## Bit 5

RAME2	Description
0	On-chip RAM (H'EE80 to H'F67F) cannot be accessed
1	On-chip RAM (H'EE80 to H'F67F) can be accessed (Initial value)

The RAME1 and RAME2 bits are initialized on the rising edge of the reset signal. They are not initialized in software standby mode. In modes 1 to 6, when the RAME1 and RAME2 bits are cleared to 0 to disable access to on-chip RAM, addresses H'F680 to H'FE7F and H'EE80 to H'F67F become an external memory area.

**(2) Bits 6, 4, and 3—Reserved:** Bit 6, 4 and 3 have undetermined values when read, and cannot be modified.

**(3) Bits 2 to 0: RAM2 to RAM0:** Bits 2 to 0 are used in RAM emulation of flash memory. For details, see section 19.2.4, RAM Control Register (RAMCR).

## 17.3 Operation

### 17.3.1 Expanded Modes (Modes 1 to 6)

In the expanded modes (modes 1 to 6), when bits RAME1 and RAME2 are set to 1, accesses to addresses H'F680 to H'FE7F and H'EE80 to H'F67F are directed to the on-chip RAM. When bits RAME1 and RAME2 are cleared to 0, accesses to addresses H'F680 to H'FE7F and H'EE80 to H'F67F are directed to off-chip memory.

### 17.3.2 Single-Chip Mode (Mode 7)

In single-chip mode (mode 7), when bits RAME1 and RAME2 are set to 1, accesses to addresses H'F680 to H'FE7F and H'EE80 to H'F67F are directed to the on-chip RAM. When bits RAME1 and RAME2 are cleared to 0, any type of access to addresses H'F680 to H'FE7F and H'EE80 to H'F67F (instruction fetch or data read/write) causes an address error. For the exception handling when an address error occurs, see section 4, “Exception Handling.”

# Section 18 RAM (H8/539F S Mask model)

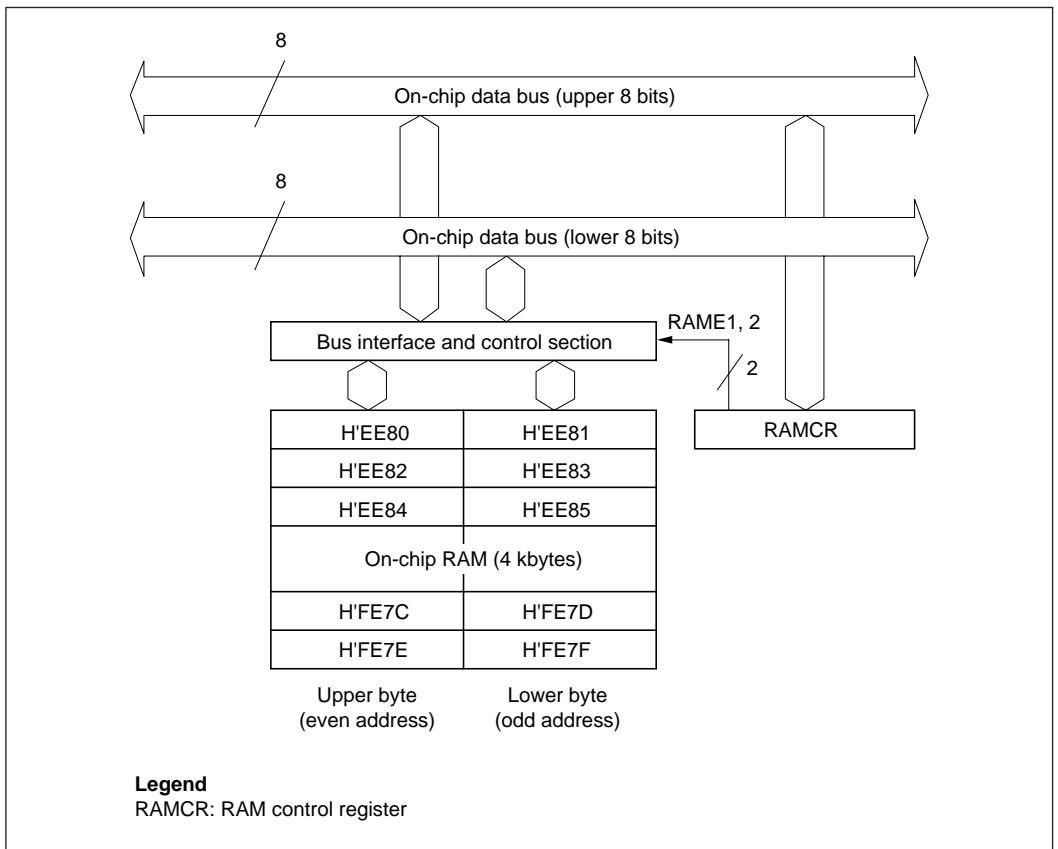
## 18.1 Overview

The H8/539F S Mask model has 4 kbytes of on-chip static RAM. The RAM is connected to the H8/500 CPU by a 16-bit data bus. The H8/500 CPU accesses both byte data and word data in two states, making the RAM suitable for rapid data transfer and high-speed computation.

The on-chip RAM is assigned to addresses H'EE80 to H'FE7F. The RAM control register (RAMCR) enables this area to be switched between on-chip RAM and external memory.

### 18.1.1 Block Diagram

Figure 18-1 shows a block diagram of the on-chip RAM.



**Figure 18-1 RAM Block Diagram**

## 18.1.2 Register Configuration

The RAM is controlled by the RAM control register (RAMCR). Table 18-1 gives the address and initial value of RAMCR.

**Table 18-1 RAM Control Register**

Address	Register Name	Abbreviation	R/W	Initial Value
H'FF15	RAM control register	RAMCR	R/W	Undetermined

## 18.2 RAM Control Register

The RAM control register (RAMCR) enables or disables access to the on-chip RAM and controls RAM area overlapping. For details of RAM area overlapping, see section 20.2.4, “RAM Control Register (RAMCR).”

Bit	7	6	5	4	3	2	1	0
	RAME1	—	RAME2	—	—	RAM2	RAM1	—
Initial value	1	*	1	*	*	0	0	*
R/W	R/W	—	R/W	—	—	R/W	R/W	—

RAM Enable bit 1  
 Enables and disables access to on-chip RAM (H'F680 to H'FE7F)

RAM enable bit 2  
 Enables or disables access to on-chip RAM (H'EE80 to H'F67F)

Reserved bit

Reserved bits

RAM2 and RAM1  
 Specify the RAM area overlapping flash memory

Reserved bit

Note: \* Bit 6, 4, 3 and 0 have undetermined values when read.

**(1) Bits 7 and 5—RAM Enable 1 and 2 (RAME1, RAME2):** These bits enable or disable access to on-chip RAM.

### Bit 7

RAME1	Description
0	On-chip RAM (H'F680 to H'FE7F) cannot be accessed
1	On-chip RAM (H'F680 to H'FE7F) can be accessed (Initial value)

## Bit 5

RAME2	Description
0	On-chip RAM (H'EE80 to H'F67F) cannot be accessed
1	On-chip RAM (H'EE80 to H'F67F) can be accessed (Initial value)

The RAME1 and RAME2 bits are initialized on the rising edge of the reset signal. They are not initialized in software standby mode. In modes 1 to 6, when the RAME1 and RAME2 bits are cleared to 0 to disable access to on-chip RAM, addresses H'F680 to H'FE7F and H'EE80 to H'F67F become an external memory area.

(2) **Bits 6, 4, 3 and 0—Reserved:** Bit 6, 4, 3 and 0 are reserved, and have undetermined values when read. They cannot be modified.

(3) **Bits 2 and 1—RAM2 and RAM1:** Bits 2 and 1 are used in RAM emulation of flash memory. For details, see section 20.2.3, RAM Control Register (RAMCR).

## 18.3 Operation

### 18.3.1 Expanded Modes (Modes 1 to 6)

In the expanded modes (modes 1 to 6), when bits RAME1 and RAME2 are set to 1, accesses to addresses H'F680 to H'FE7F and H'EE80 to H'F67F are directed to the on-chip RAM. When bits RAME1 and RAME2 are cleared to 0, accesses to addresses H'F680 to H'FE7F and H'EE80 to H'F67F are directed to off-chip memory.

### 18.3.2 Single-Chip Mode (Mode 7)

In single-chip mode (mode 7), when bits RAME1 and RAME2 are set to 1, accesses to addresses H'F680 to H'FE7F and H'EE80 to H'F67F are directed to the on-chip RAM. When bits RAME1 and RAME2 are cleared to 0, any type of access to addresses H'F680 to H'FE7F and H'EE80 to H'F67F (instruction fetch or data read/write) causes an address error. For the exception handling when an address error occurs, see section 4, “Exception Handling.”

# Section 19 Flash Memory

## (H8/539F Dual power source system ( $V_{PP} = 12\text{ V}$ ))

### 19.1 Overview

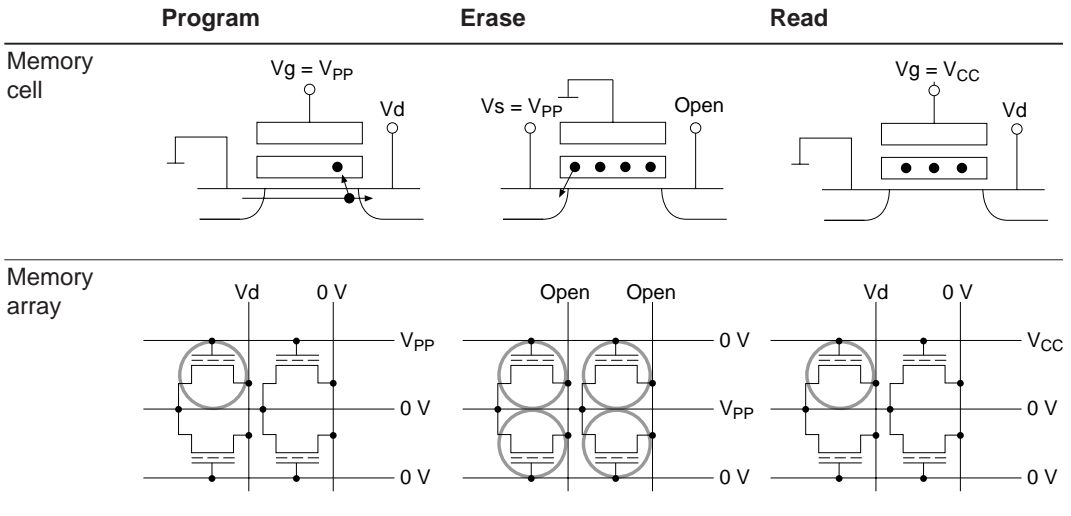
#### 19.1.1 Flash Memory Overview

Table 19-1 illustrates the principle of operation of the dual power source system H8/539F's on-chip flash memory.

Like EPROM, flash memory is programmed by applying a high gate-to-drain voltage that draws hot electrons generated in the vicinity of the drain into a floating gate. The threshold voltage of a programmed memory cell is therefore higher than that of an erased cell. Cells are erased by grounding the gate and applying a high voltage to the source, causing the electrons stored in the floating gate to tunnel out. After erasure, the threshold voltage drops. A memory cell is read like an EPROM cell, by driving the gate to the high level and detecting the drain current, which depends on the threshold voltage. Erasing must be done carefully, because if a memory cell is overerased, its threshold voltage may become negative, causing the cell to operate incorrectly.

Section 19.4.6 shows an optimal erase control flowchart and sample program.

**Table 19-1 Principle of Memory Cell Operation**





## 19.1.2 Mode Programming and ROM Address Space

As its on-chip ROM, the H8/539F has 128 kbytes of flash memory. ROM is connected to the CPU by a 16-bit data bus. The CPU accesses both byte data and word data in two states.

The flash memory is assigned to addresses H'00000 to H'03FFF and H'10000 to H'2FFFF on the memory map. The contents of flash memory addresses H'00000 to H'03FFF are the same as the contents of addresses H'10000 to H'13FFF. The mode pins enable either on-chip flash memory or external memory to be selected for this area. Table 19-2 summarizes the mode pin settings and usage of the flash memory area.

**Table 19-2 Mode Pin Settings and Flash Memory Area**

Mode	Mode Pin Setting			Flash Memory Area Usage
	MD <sub>2</sub>	MD <sub>1</sub>	MD <sub>0</sub>	
Mode 0	0	0	0	Illegal setting
Mode 1	0	0	1	External memory area
Mode 2*	0	1	0	On-chip flash memory area
Mode 3	0	1	1	External memory area
Mode 4	1	0	0	On-chip flash memory area
Mode 5	1	0	1	External memory area
Mode 6	1	1	0	External memory area
Mode 7	1	1	1	On-chip flash memory area

Note: \* Boot mode cannot be used in mode 2 in the dual power source H8/539F. Neither boot mode nor user program mode can be used in mode 2 in the single power source H8/539F (S-mask model). For details of the single power source system, see section 20, "Flash Memory (H8/539F S-Mask Model, Single Power Source)".

## 19.1.3 Features

Features of the flash memory are listed below.

- Five flash memory operating modes

The flash memory has five operating modes: program mode, program-verify mode, erase mode, erase-verify mode, and prewrite-verify mode.

- Block erase designation

Blocks to be programmed or erased in the flash memory address space can be selected by bit settings. The address space includes a large-block area (seven 16-kbyte blocks and one 12-kbyte block) and a small-block area (eight 512-byte blocks ).

- Program and erase time

Programming one byte of flash memory typically takes 50  $\mu$ s. Erasing typically takes 1 s.

- Erase-program cycles

Flash memory contents can be erased and reprogrammed up to 100 times.

- On-board programming modes

These modes can be used to program, erase, and verify flash memory contents. There are two modes: boot mode, and user programming mode.

- Automatic bit-rate alignment

In boot-mode data transfer, the H8/539F aligns its bit rate automatically to the host bit rate (9600 bps, 4800 bps, 2400 bps).

- Flash memory emulation by RAM

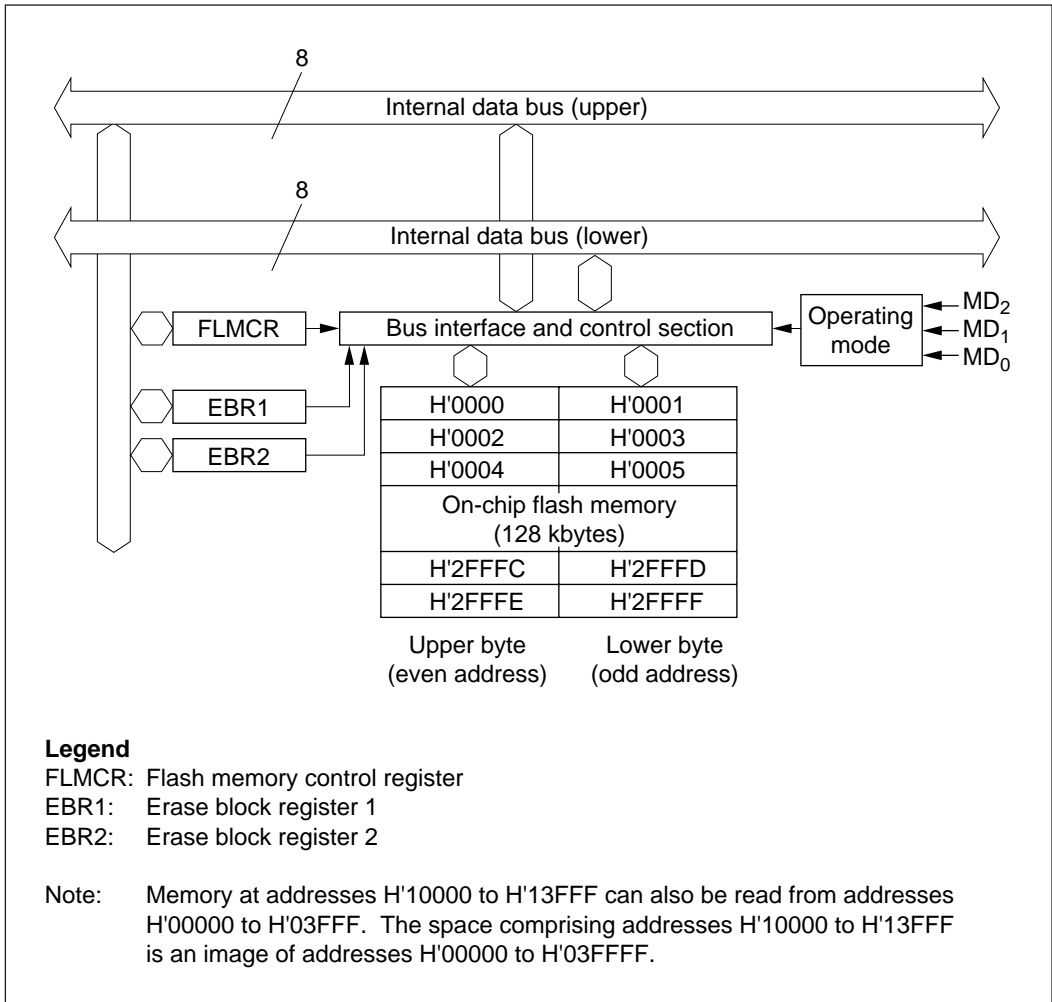
Part of the RAM area can be overlapped onto flash memory, to emulate flash memory updates in real time.

- PROM mode

As an alternative to on-board programming, the flash memory can be programmed and erased in PROM mode, using a general-purpose PROM programmer.

## 19.1.4 Block Diagram

Figure 19-1 shows a block diagram of the flash memory.



**Figure 19-1 Flash Memory Block Diagram**

### 19.1.5 Input/Output Pins

Flash memory is controlled by the pins listed in table 19-3.

**Table 19-3 Flash Memory Pins**

Pin Name	Abbreviation	Input/Output	Function
Programming power	V <sub>PP</sub>	Power supply	Apply 12.0 V
Mode 2	MD <sub>2</sub>	Input	H8/539F operating mode programming
Mode 1	MD <sub>1</sub>	Input	H8/539F operating mode programming
Mode 0	MD <sub>0</sub>	Input	H8/539F operating mode programming
Transmit data	TXD <sub>1</sub>	Output	Serial transmit data output
Receive data	RXD <sub>1</sub>	Input	Serial receive data input

The transmit data and receive data pins are used in boot mode.

### 19.1.6 Register Configuration

The flash memory is controlled by the registers listed in table 19-4.

**Table 19-4 Flash Memory Registers\*3**

Name	Abbreviation	R/W	Initial Value	Address
Flash memory control register	FLMCR	R/W	H'00 or H'80	H'FEE0
Erase block register 1	EBR1	R/W	H'00	H'FEE2
Erase block register 2	EBR2	R/W	H'00	H'FEE3
RAM control register*1	RAMCR	R/W	Undetermined*2	H'FF15
Flash memory emulation register	FLMER	R/W	H'71	H'FEEC
Flash memory status register	FLMSR	R	H'7F	H'FEED

- Notes:
1. The RAM control register enables or disables access to the on-chip RAM, but it is also used in this chapter for RAM area setting in on-board programming mode.
  2. Bits 6, 4, and 3 are reserved for system use. They cannot be written to, and will return an undetermined value if read.
  3. These registers and bits (bits 2 to 0 of RAMCR) are used only for flash memory control, and are not present in mask and ZTAT versions. Registers and bits exclusively for flash memory use should not be accessed in mask and ZTAT versions; if they are, a read will return H'FF (or 1 in the case of RAMCR bits 2 to 0), and writes will be invalid.

## 19.2 Register Descriptions

### 19.2.1 Flash Memory Control Register (FLMCR)

The flash memory control register (FLMCR) is an eight-bit register that controls the flash memory operating modes. Transitions to program mode, erase mode, program-verify mode, and erase-verify mode are made by setting bits in this register. FLMCR is initialized to H'00 by a reset, in the standby modes, and when 12 V is not applied to  $V_{pp}$ . When 12 V is applied to  $V_{pp}$ , a reset or entry to a standby mode initializes FLMCR to H'80. The FLMCR bit structure is shown next. It is not possible to set the EV, PV, E, or P bit, or any bit in the EBR1 or EBR2 register, to 1, until the  $V_{ppE}$  bit is set.

Bit	7	6	5	4	3	2	1	0
	$V_{pp}$	$V_{ppE}$	—	—	EV	PV	E	P
Initial value	0	0	0	0	0	0	0	0
R/W	R	R/W	—	—	R/W	R/W	R/W	R/W

Programming power  
 Status flag indicating that 12 V is applied to  $V_{pp}$

$V_{pp}$  enable  
 Designates enabling or disabling of 12 V application to  $V_{pp}$

Reserved bits

Erase-verify mode  
 Designates transition to or exit from erase-verify mode

Program-verify mode  
 Designates transition to or exit from program-verify mode

Erase mode  
 Designates transition to or exit from erase mode

Program mode  
 Designates transition to or exit from program mode

(1) **Bit 7—Programming Power (V<sub>PP</sub>):** The programming power supply bit (V<sub>PP</sub>) indicates the voltage level detected at the V<sub>PP</sub> pin as 1 or 0. For the threshold value, see high voltage application criterion level V<sub>H</sub> in section 22.2.1, DC Characteristics. For further information, see note 3 in section 19.7, “Flash Memory Programming and Erasing Precautions.”

**Bit 7**

V <sub>PP</sub>	Description	
0	Cleared when 12 V is not applied to V <sub>PP</sub>	(Initial value)
1	Set when 12 V is applied to V <sub>PP</sub>	

(2) **Bit 6—V<sub>PP</sub> Enable (V<sub>PP</sub>E)\*1\*2\*3:** Selects enabling or disabling of 12 V application to the V<sub>PP</sub> pin. When programming and erasing, a wait of at least 5 μs is necessary after setting this bit to allow on-chip oscillation to settle. The V<sub>PP</sub>E bit should only be cleared after the other bits (the P, E, PV, and EV bits) have been cleared.

**Bit 6**

V <sub>PP</sub> E	Description	
0	On-chip power supply disabled	(Initial value)
1	On-chip power supply enabled	

(3) **Bits 5 to 4—Reserved:** Read-only bits, always read as 0.

(4) **Bit 3—Erase-Verify Mode (EV):\*1** Selects transition to or exit from erase-verify mode.

**Bit 3**

EV	Description	
0	Exit from erase-verify mode	(Initial value)
1	Transition to erase-verify mode	

(5) **Bit 2—Erase-Verify Mode (PV):\*1** Selects transition to or exit from program-verify mode.

**Bit 2**

PV	Description	
0	Exit from program-verify mode	(Initial value)
1	Transition to program-verify mode	

(6) **Bit 1—Erase Mode (E):**\*1\*4 Selects transition to or exit from erase mode.

#### Bit 1

E	Description
0	Exit from erase mode (Initial value)
1	Transition to erase mode

(7) **Bit 0—Program Mode (P):**\*1\*4 Selects transition to or exit from program mode.

#### Bit 0

P	Description
0	Exit from program mode (Initial value)
1	Transition to program mode

Notes: \*1. Do not set two or more of these bits simultaneously.

Do not cut the  $V_{CC}$  or  $V_{PP}$  power while any of these bits is set.

\*2. Do not or clear the  $V_{PP}E$  bit at the same time as another bit (the P, E, PV, or EV bit).

\*3. Do not set or clear the  $V_{PP}E$  bit during execution of a program in flash memory. For details, see item (3) in section 19.7, “Flash Memory Programming and Erasing Precautions.”

\*4. Set the P and E bits in accordance with the programming and erasing algorithms shown in section 19.4, “Programming and Erasing Flash Memory.” When one of these bits is set, watchdog timer setting should be carried out beforehand to provide for the possibility of program runaway. See section 19.7, “Flash Memory Programming and Erasing Precautions” for notes on the use of these bits.

### 19.2.2 Erase Block Register 1 (EBR1)

Erase block register 1 (EBR1) is an eight-bit register that designates large flash-memory blocks for erasure. EBR1 is initialized to H'00 by a reset, in the standby modes, and when 12 V is not applied to  $V_{PP}$ . When a bit in EBR1 is set to 1, the corresponding block is selected and can be erased. Figure 19-2 shows a block map. No bits in the EBR1 or EBR2 register can be set to 1 until the  $V_{PP}E$  bit is set to 1 in the FLMCR register.

Bit	7	6	5	4	3	2	1	0
	LB7	LB6	LB5	LB4	LB3	LB2	LB1	LB0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**(1) Bits 7 to 0—Large Block 7 to 0 (LB7 to LB0):** These bits select large blocks (LB7 to LB0) to be erased.

**Bits 7 to 0**

<b>LB7 to LB0</b>	<b>Description</b>
0	Block LB7 to LB0 is not selected (Initial value)
1	Block LB7 to LB0 is selected

**19.2.3 Erase Block Register 2 (EBR2)**

Erase block register 2 (EBR2) is an eight-bit register that selects small flash-memory blocks for programming and erasure. EBR2 is initialized to H'00 by a reset, in the standby modes, and when 12 V is not applied to V<sub>PP</sub>. When a bit in EBR2 is set to 1, the corresponding block is selected for programming and erasure. Figure 19-2 shows a block map.

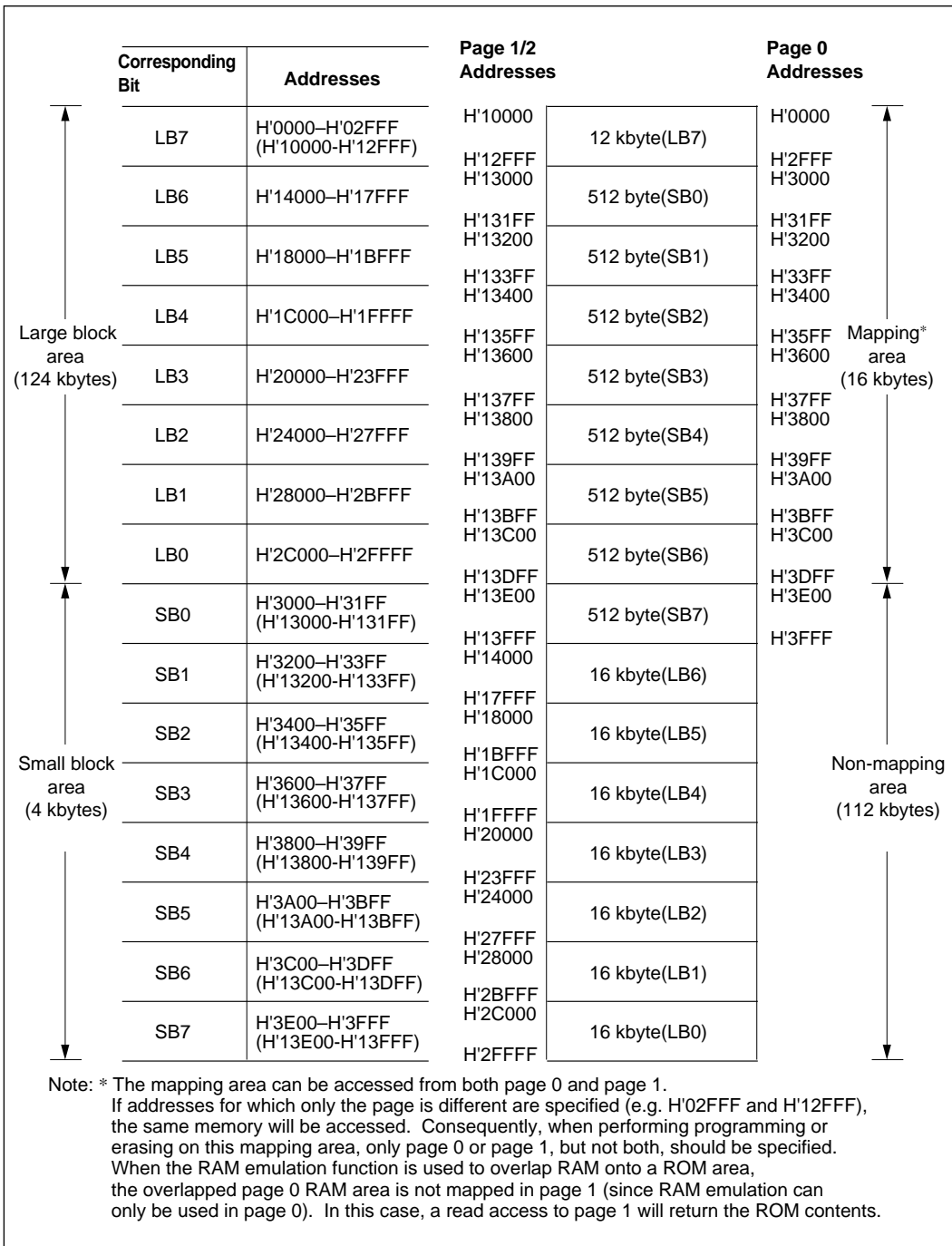
Bit	7	6	5	4	3	2	1	0
	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**(1) Bits 7 to 0—Small Block 7 to 0 (SB7 to SB0):** These bits select small blocks (SB7 to SB0) to be programmed and erased.

**Bits 7 to 0**

<b>SB7 to SB0</b>	<b>Description</b>
0	Block SB7 to SB0 is not selected (Initial value)
1	Block SB7 to SB0 is selected



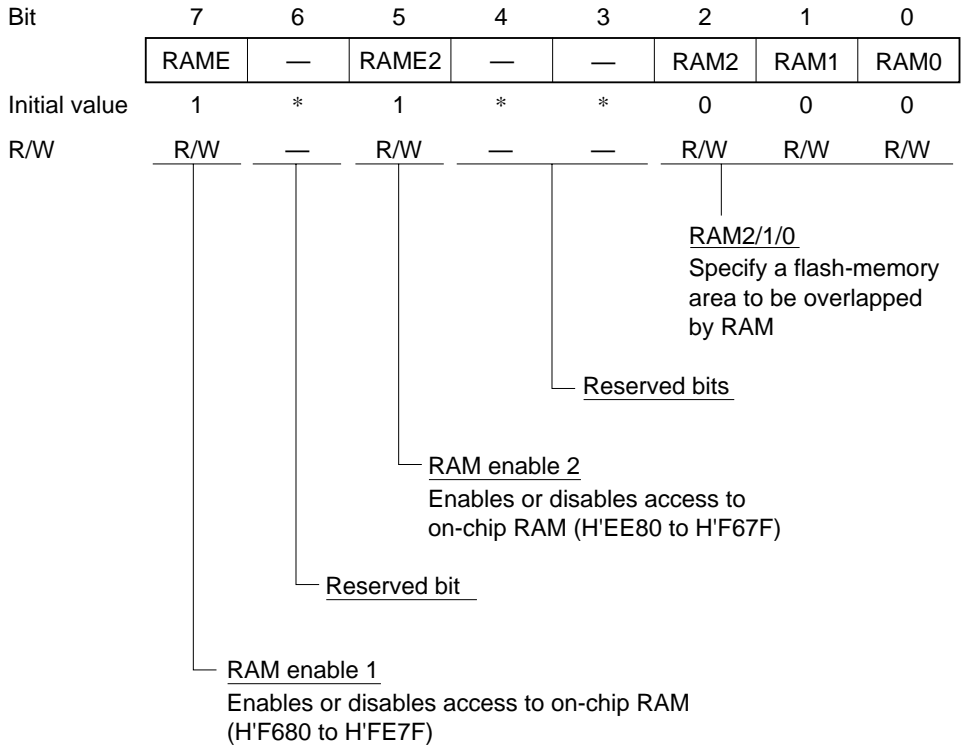


Note: \* The mapping area can be accessed from both page 0 and page 1. If addresses for which only the page is different are specified (e.g. H'02FFF and H'12FFF), the same memory will be accessed. Consequently, when performing programming or erasing on this mapping area, only page 0 or page 1, but not both, should be specified. When the RAM emulation function is used to overlap RAM onto a ROM area, the overlapped page 0 RAM area is not mapped in page 1 (since RAM emulation can only be used in page 0). In this case, a read access to page 1 will return the ROM contents.

**Figure 19-2 Erase Block Map**

## 19.2.4 RAM Control Register (RAMCR)

The RAM control register (RAMCR) enables or disables access to the on-chip RAM and controls RAM area overlapping.



Note: \* Bit 6, 4, and 3 have undetermined values when read.

(1) **Bits 7 and 5—RAM Enable 1 and 2 (RAME1, RAME2):** When bit 7 or 5 is cleared to 0, access to the respective on-chip RAM area is disabled. For details see section 17.2, “RAM Control Register.”

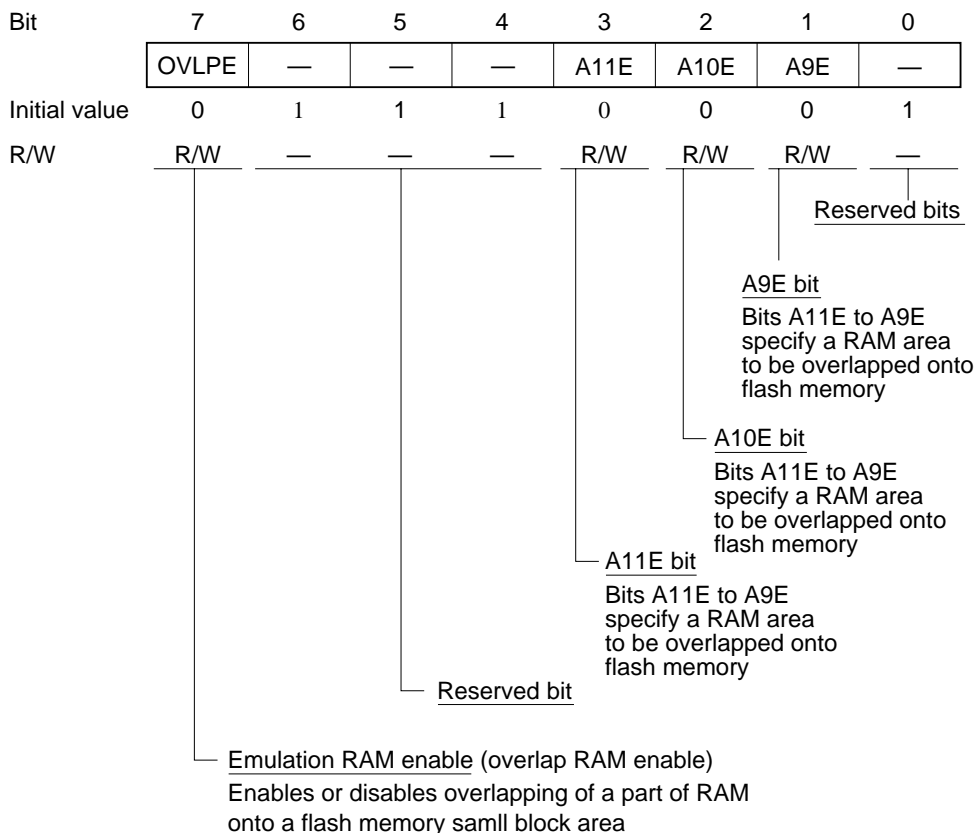
(2) **Bits 6, 4, and 3—Reserved:** Bit 6, 4, 3 have undetermined values when read. They cannot be modified.

**(3) Bits 2 to 0—RAM2 to RAM0:** Bits 2 to 0 are used together with bits 7, 3, 2, and 1 of the flash memory emulation register (FLMER) to specify a ROM area for which overlapping is to be performed (see table 19-5). In modes 2, 4, and 7 (with on-chip flash memory enabled), the initial value of these bits is 0, but they can be modified by writing 1\*1. In other modes these bits cannot be modified and always read 0. They are initialized by a reset and in hardware standby mode. They are not initialized in software standby mode.

Note: 1. Bits 2 to 0 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7. (In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{PP}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{PP}$  voltage.)

### 19.2.5 Flash Memory Emulation Register (FLMER)

The flash memory emulation register (FLMER) performs enabling and disabling of flash memory RAM emulation and RAM area modification when RAM emulation is started.



**(1) Bit 7—Emulation RAM Enable (OVLPE):** Used with 3 to 1 to specify a RAM area (see table 19-5). When bit 7 is set, all flash memory blocks are protected from programming and erasing, regardless of the values of bits 3 to 1. This state is referred to as emulation protection\*<sup>1</sup>. In this state the flash memory will not enter program mode or erase mode even if the P or E bit is set in the flash memory control register (FLMCR). Only transitions to verify modes are possible. Bit 7 must be cleared to 0 to enable flash memory to be actually programmed or erased.

In modes 2, 4, and 7 (with on-chip flash memory enabled), the initial value of this bit is 0, but it can be modified by writing 1\*<sup>2</sup>. In other modes this bit cannot be modified and always reads 0. It is initialized by a reset and in hardware standby mode. It is not initialized in software standby mode.

**(2) Bits 3 to 1—A11E to A9E:** Bits 3 to 1 select the RAM area to be overlapped onto ROM when performing RAM emulation of flash memory (see table 19-6).

In modes 2, 4, and 7 (with on-chip flash memory enabled), the initial value of these bits is 0, but they can be modified by writing. In other modes, these bits cannot be modified and always read 0. They are initialized by a reset and in hardware standby mode. They are not initialized in software standby mode.

Notes: 1. For details of emulation protection, see section 19.4.8, “Protect Modes.”  
2. Bits 7, 3, 2, and 1 of the flash memory emulation register (FLMER) and bits 2 to 0 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7.  
(In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{PP}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{PP}$  voltage.)

## 19.2.6 Flash Memory Status Register (FLMSR)

The flash memory status register (FLMSR) is used to detect a flash memory error.

Bit	7	6	5	4	3	2	1	0
	FLER	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
R/W	R	—	—	—	—	—	—	—

Flash memory error  
Status flag indicating that an error was detected during programming or erasing

Reserved bit

**(1) Bit 7—Flash Memory Error (FLER):** Indicates that an error occurred while flash was being programmed or erased. When bit 7 is set, flash memory is placed in an error-mode.\*1

Bits 7	
FLER	Description
0	Flash memory is not write/erase-protected (Initial value) (is not in error protect mode*1) [Clearing conditions] Reset by $\overline{\text{RES}}$ pin*3 or hardware standby mode
1	Indicates that an error occurred while flash memory was being programmed or erased, and error protection*1 is in effect [Setting conditions] <ul style="list-style-type: none"> <li>•Flash memory was read*2 while being programmed or erased (including vector read or instruction fetch, but not including reading of a RAM area overlapped onto flash memory).</li> <li>•A hardware exception-handling sequence (other than a reset, invalid instruction, trap instruction, or zero-divide exception) was executed just before programming erasing.*4</li> <li>•The SLEEP instruction (for transition to sleep mode or software standby mode) was executed during programming or erasing.</li> </ul>

- Note:
1. For details, see section 19.4.8, “Protect Modes.”
  2. The read data has undetermined values.
  3. In the H8/538F a watchdog timer reset is included in the FLER bit clearing conditions, but in the H8/539F only  $\overline{\text{RES}}$  pin input is applicable.
  4. This is before stack and vector reads are performed in exception handling.

(2) **Bits 6 to 0—Reserved:** Read-only bits, always read as 1.

**Table 19-5 ROM Area Setting**

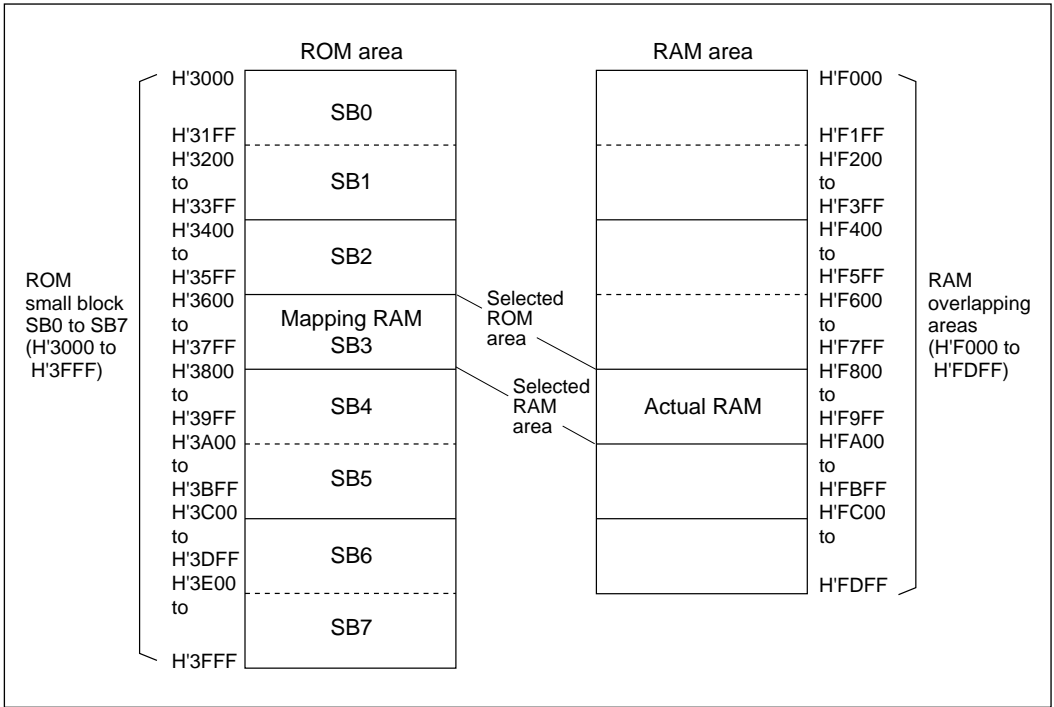
ROM Area (Mapping RAM Area)	FLMER Register Bit 7 <sup>*1</sup>	RAMCR Register			Overlap Function	Program /Erase Protection
		Bit 2 <sup>*1</sup>	Bit 1 <sup>*1</sup>	Bit 0 <sup>*1</sup>		
		OVLPE	RAM2	RAM1		
—	0	0/1	0/1	0/1	Disabled	Enabled
H'3000-H'31FF	1	0	0	0	Enabled	Enabled
H'3200-H'33FF	1	0	0	1	Enabled	Enabled
H'3400-H'35FF	1	0	1	0	Enabled	Enabled
H'3600-H'37FF	1	0	1	1	Enabled	Enabled
H'3800-H'39FF	1	1	0	0	Enabled	Enabled
H'3A00-H'3BFF	1	1	0	1	Enabled	Enabled
H'3C00-H'3DFF	1	1	1	0	Enabled	Enabled
H'3E00-H'3FFF	1	1	1	1	Enabled	Enabled

**Table 19-6 RAM Area<sup>\*2</sup> Setting**

RAM Area <sup>*2</sup> (Mapping RAM Area)	FLMER Register			RAMCR Register	
	Bit 3 <sup>*1</sup>	Bit 2 <sup>*1</sup>	Bit 1 <sup>*1</sup>	Bit 7 <sup>*1</sup>	Bit 5 <sup>*1</sup>
	A11E	A10E	A9E	RAME1	RAME2
H'F000-H'F1FF (512 bytes)	0	0	0	1	1
H'F200-H'F3FF (512 bytes)	0	0	1	1	1
H'F400-H'F5FF (512 bytes)	0	1	0	1	1
H'F600-H'F7FF (512 bytes)	0	1	1	1	1
H'F800-H'F9FF (512 bytes)	1	0	0	1	1
H'FA00-H'FBFF (512 bytes)	1	0	1	1	1
H'FC00-H'FDFF (512 bytes)	1	1	0	1	1
Use prohibited <sup>*3</sup>	1	1	1	1	1
Use prohibited <sup>*4</sup>	0/1	0/1	0/1	0/1 <sup>*4</sup>	0/1 <sup>*4</sup>

- Notes: 1. Bits 7, 3, 2, and 1 of the flash memory emulation register (FLMER) and bits 2 to 0 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7.  
(In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{PP}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{PP}$  voltage.)
2. RAM area overlapped onto flash memory
  3. When A11E and A10E are both set to 1, A9E is always cleared to 0.
  4. Use prohibited when RAME1=0 or RAME2=0. (Can be used when RAME1=RAME2=1.)

## Example of Overlapping of ROM and RAM Areas



## 19.3 On-Board Programming Modes

When an on-board programming mode is selected, the on-chip flash memory can be programmed, erased, and verified. There are two on-board programming modes: boot mode, and user program mode. These modes are selected by inputs at the mode pins (MD<sub>2</sub> to MD<sub>0</sub>) and V<sub>PP</sub> pin. Table 19-7 indicates how to select the on-board programming modes. Boot mode cannot be used in the H8/539F's mode 2 (on-chip ROM enabled). For information about turning V<sub>PP</sub> on and off, and V<sub>PP</sub> and MD<sub>2</sub> pin circuit examples, see section 19.7, Flash Memory Programming and Erasing Precautions, and section 19.8, Notes on Mounting Board Development—Handling of V<sub>PP</sub> and MD<sub>2</sub> Pins.

**Table 19-7 On-Board Programming Mode Selection**

Mode Selections		V <sub>PP</sub>	MD <sub>2</sub>	MD <sub>1</sub>	MD <sub>0</sub>	Notes
Boot mode	Mode 4	12 V *	12 V *	0	0	0:V <sub>IL</sub> 1:V <sub>IH</sub>
	Mode 7		12 V *	1	1	
User program mode	Mode 2		0	1	0	
	Mode 4		1	0	0	
	Mode 7		1	1	1	

Note: \*(1) For the timing of turning V<sub>PP</sub> on, see notes 6 to 8 in “Notes on Use of Boot Mode.”

Set the 12 V application level to 12.0 ± 0.6 V.

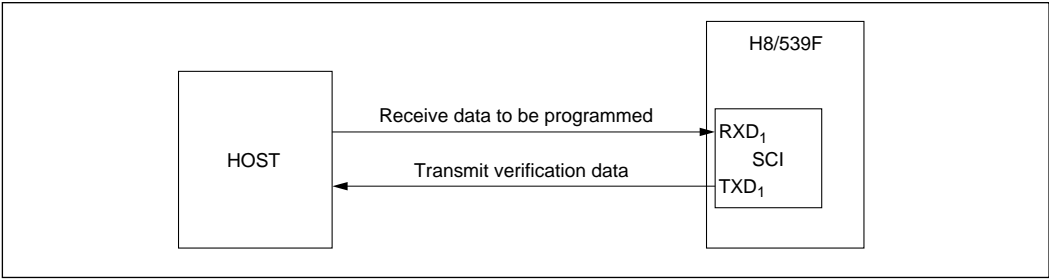
(2) In boot mode, the mode control register (MDCR) can monitor the mode 4 or 7 status in the same way as in normal mode.

### 19.3.1 Boot Mode

To use boot mode, a user program for programming and erasing the flash memory must be provided in advance on the host machine (which may be a personal computer). Serial communication interface channel 1 is used in asynchronous mode. If the H8/539F is placed in boot mode, after it comes out of reset, a built-in boot program is activated. This program starts by measuring the low period of data transmitted from the host and setting the bit rate register (BRR) accordingly. The H8/539F's built-in serial communication interface (SCI) can then be used to download the user program from the host machine. The user program is stored in on-chip RAM.

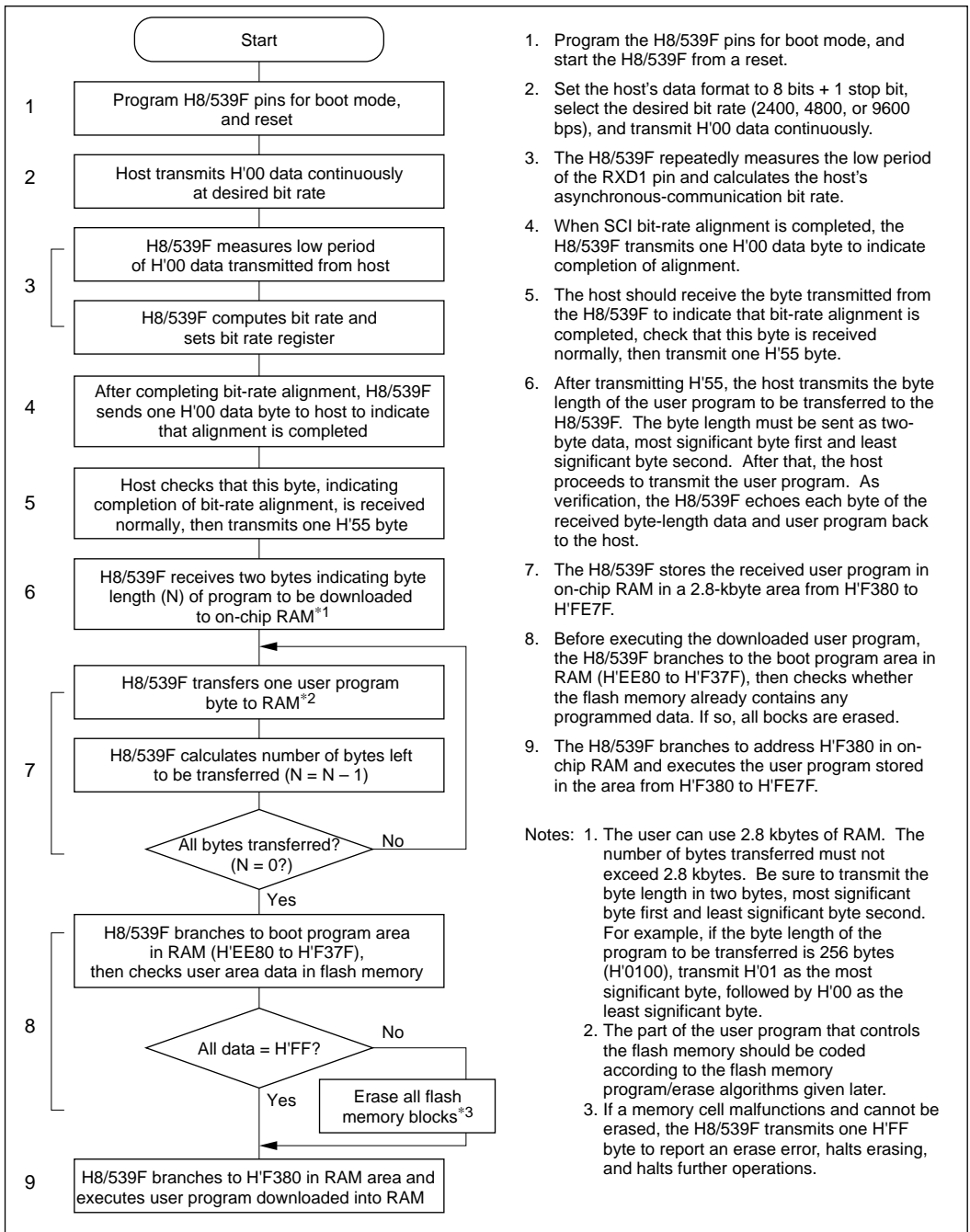
After the program has been stored, execution branches to the start address of the user program in on-chip RAM (H'F380), and the program stored on RAM is executed to program and erase the flash memory. Figure 19-4 shows the boot-mode execution procedure.





**Figure 19-3 Boot-Mode System Configuration**

**Boot-Mode Execution Procedure:** Figure 18-4 shows the boot-mode execution procedure.

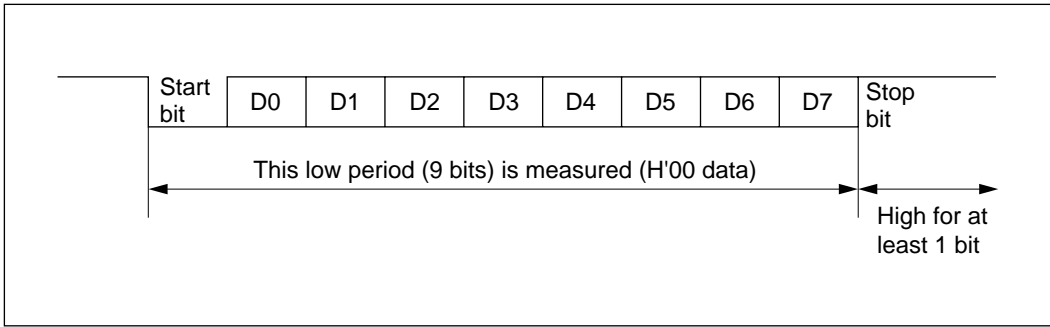


1. Program the H8/539F pins for boot mode, and start the H8/539F from a reset.
2. Set the host's data format to 8 bits + 1 stop bit, select the desired bit rate (2400, 4800, or 9600 bps), and transmit H'00 data continuously.
3. The H8/539F repeatedly measures the low period of the RXD1 pin and calculates the host's asynchronous-communication bit rate.
4. When SCI bit-rate alignment is completed, the H8/539F transmits one H'00 data byte to indicate completion of alignment.
5. The host should receive the byte transmitted from the H8/539F to indicate that bit-rate alignment is completed, check that this byte is received normally, then transmit one H'55 byte.
6. After transmitting H'55, the host transmits the byte length of the user program to be transferred to the H8/539F. The byte length must be sent as two-byte data, most significant byte first and least significant byte second. After that, the host proceeds to transmit the user program. As verification, the H8/539F echoes each byte of the received byte-length data and user program back to the host.
7. The H8/539F stores the received user program in on-chip RAM in a 2.8-kbyte area from H'F380 to H'FE7F.
8. Before executing the downloaded user program, the H8/539F branches to the boot program area in RAM (H'EE80 to H'F37F), then checks whether the flash memory already contains any programmed data. If so, all blocks are erased.
9. The H8/539F branches to address H'F380 in on-chip RAM and executes the user program stored in the area from H'F380 to H'FE7F.

- Notes:
1. The user can use 2.8 kbytes of RAM. The number of bytes transferred must not exceed 2.8 kbytes. Be sure to transmit the byte length in two bytes, most significant byte first and least significant byte second. For example, if the byte length of the program to be transferred is 256 bytes (H'0100), transmit H'01 as the most significant byte, followed by H'00 as the least significant byte.
  2. The part of the user program that controls the flash memory should be coded according to the flash memory program/erase algorithms given later.
  3. If a memory cell malfunctions and cannot be erased, the H8/539F transmits one H'FF byte to report an erase error, halts erasing, and halts further operations.

**Figure 19-4 Boot Mode Flowchart**

## Automatic Alignment of SCI Bit Rate



**Figure 19-5 Measurement of Low Period in Data Transmitted from Host**

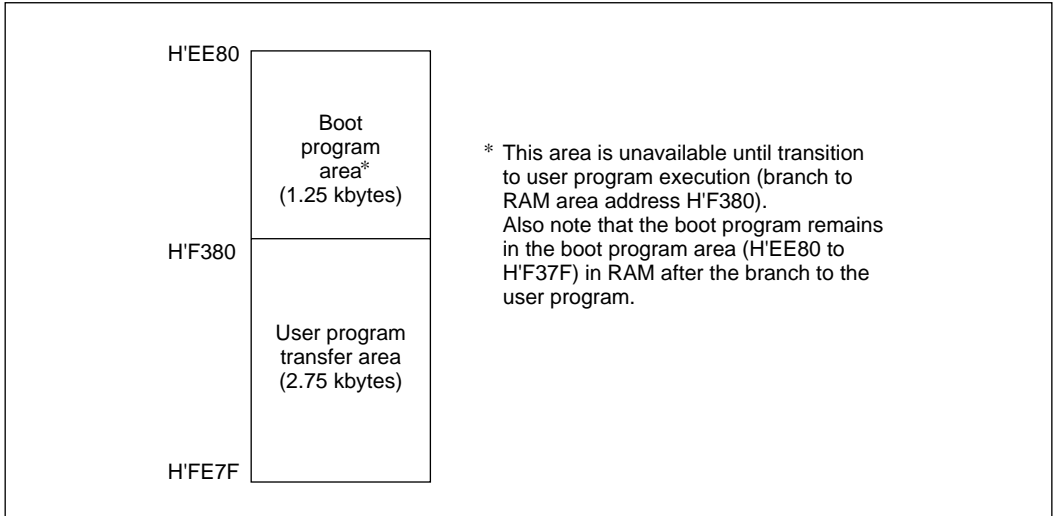
When started in boot mode, the H8/539F measures the low period in asynchronous SCI data transmitted from the host (figure 19-5). The data format is eight data bits, one stop bit, and no parity bit. From the measured low period (nine bits), the H8/539F computes the host's bit rate. After aligning its own bit rate, the H8/539F sends the host one byte of H'00 data to indicate that bit-rate alignment is completed. The host should check that this alignment-completed indication is received normally, then transmit one H'55 byte. If the host does not receive a normal alignment-completed indication, the H8/539F should be reset, then restarted in boot mode to measure the low period again. There may be some alignment error between the host's and H8/539F's bit rates, depending on the host's bit rate and the H8/539F's system clock frequency. To have the SCI operate normally, set the host's bit rate to 2400, 4800, or 9600 bps.\*<sup>1</sup> Table 19-8 lists typical host bit rates and indicates the clock-frequency ranges over which the H8/539F can align its bit rate automatically. Boot mode should be used within these frequency ranges.\*<sup>2</sup>

**Table 19-8 System Clock Frequencies Permitting Automatic Bit-Rate Alignment by H8/539F**

Host Bit Rate	System Clock Frequencies Permitting Automatic Bit-Rate Alignment by H8/539F
9600 bps	8 MHz to 16 MHz
4800 bps	4 MHz to 16 MHz
2400 bps	2 MHz to 16 MHz

- Notes:
1. The host's bit rate should be set only to 2400, 4800, or 9600 bps. Do not set it to other values.
  2. Though host bit rates and system clock frequencies which are not listed in table 18-8 may permit automatic bit-rate alignment by the H8/539F, they cause some alignment error between the host's and H8/539F's bit rates. Therefore, the SCI cannot operate normally after such bit-rate alignment. Boot mode should be used within those host bit rate and system clock frequency ranges listed in table 19-7.

**RAM Area Allocation in Boot Mode:** In boot mode, the 1280 bytes from H'EE80 to H'F37F are reserved for use by the boot program. The user program is transferred into the area from H'F380 to H'FE7F (2.75 kbytes). The boot program area is used during the transition to execution of the user program transferred into RAM.



**Figure 19-6 RAM Areas in Boot Mode**

### Notes on Use of Boot Mode

1. When the H8/539F comes out of reset in boot mode, it measures the low period of the input at the SCI's RXD<sub>1</sub> pin. The reset should end with RXD<sub>1</sub> high. After the reset ends, it takes about 100 states for the H8/539F to get ready to measure the low period of the RXD<sub>1</sub> input.
2. In boot mode, if any data has been programmed into the flash memory (if all data are not H'FF), all flash memory blocks are erased. Boot mode is for use when user program mode is unavailable, e.g. the first time on-board programming is performed, or if the update program activated in user program mode is accidentally erased.
3. Interrupts cannot be used while the flash memory is being programmed or erased.
4. The RXD<sub>1</sub> and TXD<sub>1</sub> lines should be pulled up on-board.
5. Before branching to the user program (at address H'F380 in the RAM area), the H8/539F terminates transmit and receive operations by the on-chip SCI (by clearing the RE and TE bits to 0 in the serial control register (SCR)), but the auto-aligned bit rate remains set in bit rate register BRR1. The transmit data pin (TXD<sub>1</sub>) is in the high output state (in port 7, the P7<sub>2</sub>DDR and P7<sub>2</sub>DR bits are set to 1).

When the branch to the user program occurs, the contents of general registers in the CPU are undetermined. After the branch, the user program should begin by initializing general registers, especially the stack pointer (SP), which is used implicitly in subroutine calls and at other times. The stack pointer must be set to provide a stack area for use by the user program. The other on-chip registers do not have specific initialization requirements.

6. In a transition to boot mode, a reset start can be performed after applying 12 V to the MD<sub>2</sub> and V<sub>PP</sub> pins in accordance with the mode setting conditions in table 19-6. The H8/539F latches the mode pin state internally when the reset is cleared (low-to-high transition)\*<sup>1</sup>, and the boot mode state is retained.

Boot mode can be exited by a reset clearance\*<sup>1</sup> after clearing 12 V application to the MD<sub>2</sub> and V<sub>PP</sub> pins, but the following points need to be noted.

- (a) When switching from boot mode to normal mode (V<sub>PP</sub> ≠ 12 V, MD<sub>2</sub> ≠ 12 V), the microcontroller's internal boot mode state must first be cleared by reset input using the  $\overline{\text{RES}}$  pin.

In this case, the interval required between cutting V<sub>pp</sub> and reset clearance is the flash memory read setup time (t<sub>FRS</sub>)\*<sup>2</sup>.

- (b) If application of 12 V to the MD<sub>2</sub> pin is cleared during boot mode, the microcontroller's internal boot mode state will be retained and boot mode will continue unless reset input is performed using the  $\overline{\text{RES}}$  pin.

If a watchdog timer reset occurs in the boot mode state, the microcontroller's internal mode state will not be cleared, and the on-chip boot program will be restarted regardless of the mode pin state.

- (c) When switching to boot mode (when reset is cleared) and during a boot mode operation, ensure that the program voltage V<sub>PP</sub> stays within the range 12 V ± 0.6 V. Boot mode execution will not be performed correctly outside this range. Also, do not cut V<sub>PP</sub> during boot program execution, or during flash memory programming or erasing\*<sup>2</sup>.

Notes: 1. With regard to mode pin input, the mode programming setup time (t<sub>MDS</sub>) must be satisfied with respect to reset clearance timing. When 12 V is applied/cut at the MD<sub>2</sub> pin, there will be a delay in the rise and fall waveforms due to the effect of the pull-up/pull-down resistor, etc., connected at the MD<sub>2</sub> pin. This delay must be confirmed in practice in the design process. For the MD<sub>2</sub> pin threshold value, see high voltage application criterion level V<sub>H</sub> in section 22.2.1, DC Characteristics.

2. See note 3 in section 19.7. “Flash Memory Programming and Erasing Precautions” for notes on  $V_{PP}$  application/cutoff. For the  $V_{PP}$  pin threshold value, see high voltage application criterion level  $V_H$  in section 22.2.1, DC Characteristics.
7. If the  $MD_2$  pin is changed from 0 V to 12 V or from 12 V to 0 V during a reset (while a low level is being input at the  $\overline{RES}$  pin), the microcontroller operating mode is changed by the momentary transition to the 5 V input level. Since the state of multiplexed address/port pins and bus control output signals ( $\overline{AS}$ ,  $\overline{RD}$ ,  $\overline{HWR}$ ,  $\overline{LWR}$ ) changes as a result, use of these pins as output signals during a reset must be inhibited outside the microcontroller.
8. When applying 12 V to the  $V_{PP}$  and  $MD_2$  pins, ensure that the peak overshoot does not exceed the maximum rating of 13 V. Also be sure to connect decoupling capacitors to the  $V_{PP}$  and  $MD_2$  pins. Circuit examples are shown in figure 19-24 in section 19.7, Flash Memory Programming and Erasing Precautions, and in section 19.8, Notes on Mounting Board Development—Handling of  $V_{PP}$  and  $MD_2$  Pins.

### 19.3.2 User Program Mode

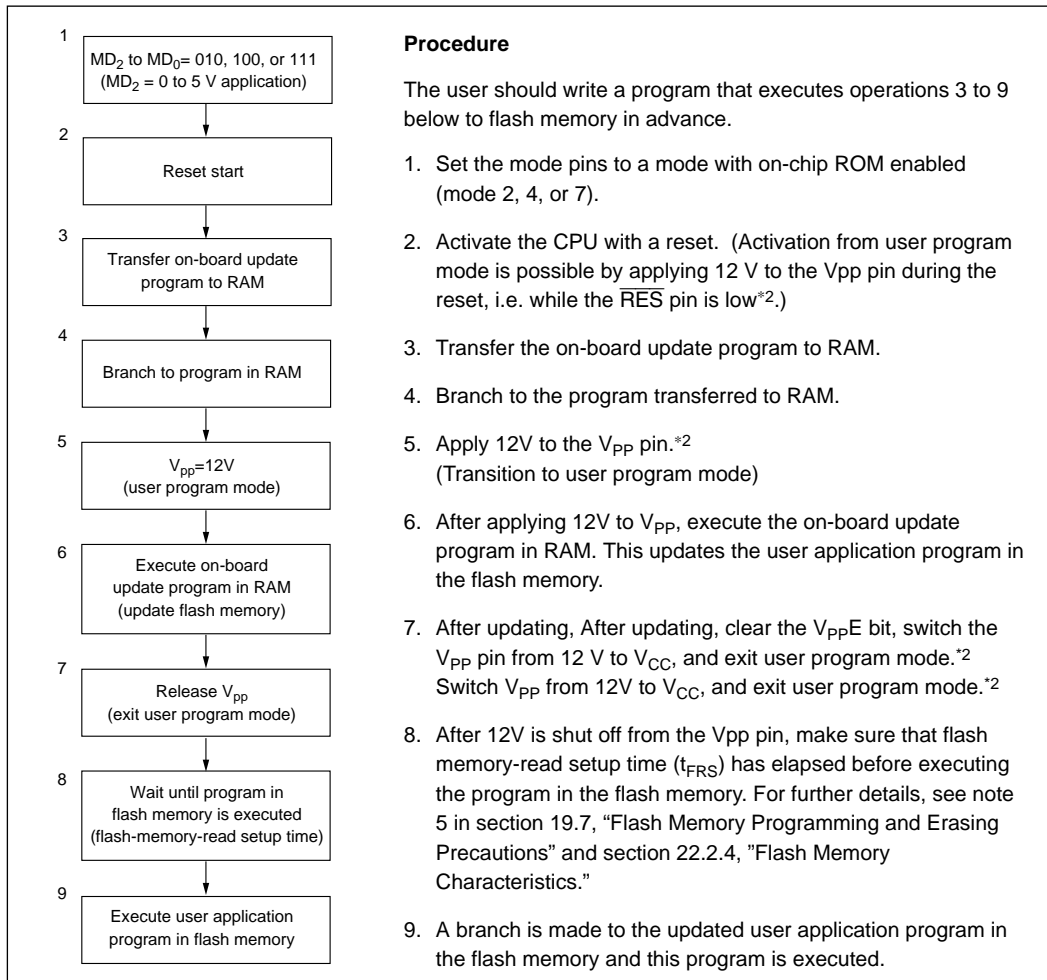
When set to user program mode, the H8/539F can erase and program its flash memory by executing a user program. On-board updates of the on-chip flash memory can be carried out by providing on-board circuits for supplying  $V_{PP}$  and data, and storing an update program in part of the program area.

To select user program mode, select a mode that enables the on-chip ROM (mode 2, 4, or 7) and apply 12 V to the  $V_{PP}$  pin. In this mode, the on-chip peripheral modules operate as they normally would in mode 2, 4, or 7, except for the flash memory. A watchdog timer overflow, however, cannot output a reset signal while 12 V is applied to  $V_{PP}$ . The watchdog timer’s reset output enable bit (RSTOE) should not be set to 1.

The flash memory cannot be read while being erased, so the update program must either be stored in external memory, or transferred temporarily to the RAM area and executed in RAM.

**Example of User Program Mode Execution Procedure\*1:** Figure 19-7 shows the procedure for user program mode execution in RAM.

In the case of a reset start, activation is possible from user program mode.



**Figure 19-7 User Program Mode Operation (Example)**

- Notes: 1. Do not apply 12 V to the V<sub>pp</sub> pin during normal operation. To prevent microcontroller errors caused by accidental programming or erasing, apply 12 V to V<sub>pp</sub> only when the flash memory is programmed or erased, or when flash memory is emulated by RAM. Overprogramming or overerasing due to program runaway, etc., may prevent memory cells from operating normally. While 12 V is applied, the watchdog timer should be running and enabled to halt runaway program execution, so that program runaway will not lead to overprogramming or overerasing.
2. For further information about turning V<sub>pp</sub> on and off, see note 5 in section 19.7, "Flash Memory Programming and Erasing Precautions."

## 19.4 Programming and Erasing Flash Memory

The H8/539F's on-chip flash memory is programmed and erased by software, using the CPU. The flash memory can operate in program mode, erase mode, program-verify mode, erase-verify mode, or prewrite-verify mode. Transitions to these modes can be made by setting the P, E, PV, and EV bits in the flash memory control register (FLMCR). A wait time of at least 5  $\mu$ s should be left after setting the VppE bit when making a transition to an operating mode. The area to be programmed in flash memory (target block is specified by means of erase block registers 1 and 2 (EBR1, EBR2)).

The flash memory cannot be read while being programmed or erased. The program that controls the programming and erasing of the flash memory must be stored and executed in on-chip RAM or in external memory. A description of each mode is given below, with recommended flowcharts and sample programs for programming and erasing. Recommended programming and erasing flowcharts adopt an algorithm which doubles the programming or erasing time successively. This algorithm can decrease the number of repetitions and shorten the verify time, enabling high-speed programming and erasing. The high-speed algorithm is specially effective, when the H8/539F is used at a low clock frequency.

Section 19.7, "Flash Memory Programming and Erasing Precautions," gives further notes on programming and erasing. See section 22.2.4, Flash Memory Characteristics, for the wait time after a bit is set or cleared in the flash memory control register (FLMCR).

### 19.4.1 Program Mode

To write data into the flash memory, follow the programming algorithm shown in figure 19-8. This programming algorithm can write data without subjecting the device to voltage stress or impairing the reliability of programmed data.

To program data, first write the data to the address to be programmed, as in writing to RAM. The flash memory latches the address and data in an address latch and data latch. Next set the P bit in FLMCR, selecting program mode. The programming duration is the time during which the P bit is set. Programming duration should be set to increase by  $2^{n-1}$  times ( $n=1, 2, 3, 4, 5, 6$ ) of the initial setting value. A software timer should be used to provide an initial setting value of 10 to 15.8  $\mu$ s. Set  $n$  so that the total programming time does not exceed 1ms. Programming for too long a time, due to program runaway for example, can cause device damage. Before selecting program mode, set up the watchdog timer so as to prevent overprogramming.

### 19.4.2 Program-Verify Mode

In program-verify mode, after data has been programmed in program mode, the data is read to check that it has been programmed correctly.

After the programming time has elapsed, exit programming mode (clear the P bit to 0) and select



program-verify mode (set the PV bit to 1). In program-verify mode, a program-verify voltage is applied to the memory cells at the latched address. If the flash memory is read in this state, the data at the latched address will be read. After selecting program-verify mode, wait 4  $\mu$ s or more before reading, then compare the programmed data with the verify data. If they agree, exit program-verify mode and program the next address. If they do not agree, select program mode again and repeat the same program and program-verify sequence. When repeating the program and program-verify sequence for the same bit, set the total programming time to a maximum of 1ms.

### 19.4.3 Programming Flowchart and Sample Program

#### Flowchart for Programming One Byte

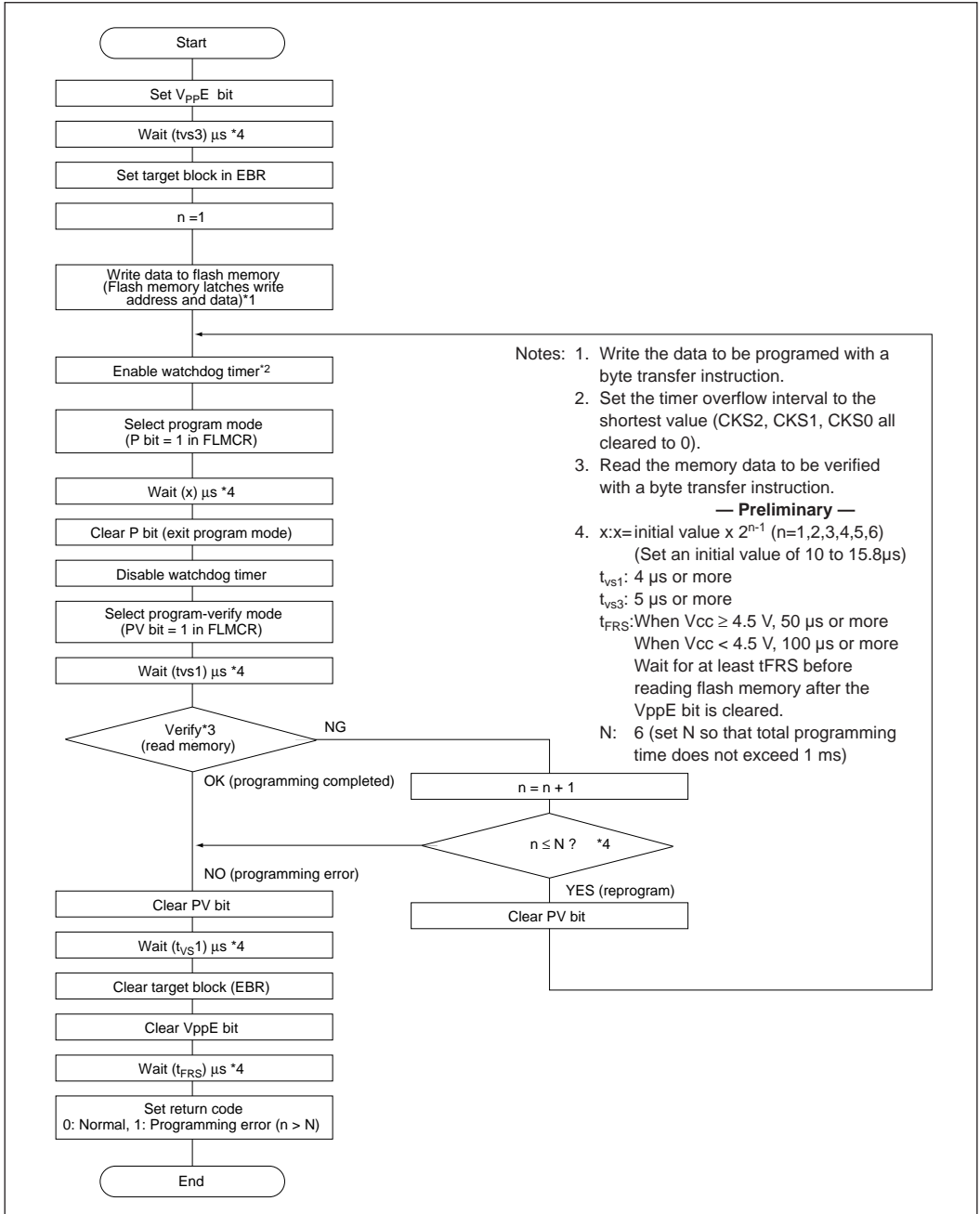


Figure 19-8 Programming Flowchart

**Sample Program for Programming One Byte:** This program uses the following registers:

- R0: Specifies program data as byte data./ Return value (0: Normal, 1: Write error).
- R1: Used to specify the program target block.
- R2: Used for program address page specification.
- R3: Used for program address specification.

After the values of R0 (program data), R1 (program target block), R2 (program address page), and R3 (program address) have been specified, arbitrary data can be programmed at an arbitrary address by calling the fwrite subroutine.

The wait time due to software looping after bit setting depends on the operating frequency. The relevant operating frequency can be specified by setting the MHZ symbol value. In this program the wait time (number of loops) is calculated on the assumption that the scb/f instruction is located at an even address in two-state access space (on-chip RAM).

The read setup time (tFRS) after clearing the V<sub>ppE</sub> bit is the value for the case where V<sub>CC</sub> ≥ 4.5 V.

See section 22.2.4, “Flash Memory Characteristics,” for the wait time after setting a bit in the flash memory control register (FLMCR).

```

0001:; *****
0002:; * fwrite, src (Ver. 0.10) - Preliminary -
0003:; * Sample program for programming one byte of H8/539F flash memory
0004:; *
0005:; *****
0006:;
0007:;
0008: MHZ .equ d'10 ; Depends on operating frequency (10 MHz)
0009: RAMSTR .equ H'EE80 ; Program transfer destination RAM address
0010:; Register addresses
0011: FLMCR .equ H'FEE0 ; Flash memory control register
0012: EBR .equ H'FEE2 ; Target block specification register
0013: TCSR .equ H'FF10 ; Timer control/status register
0014: WCR .equ H'FF14 ; Wait control register
0015:;
0016:;
0017: .align H'2
0018: main: .equ $
0019: ldc.b #H'00:8,tp ; Stack page register setting
0020: mov.w #H'FE80,sp ; Stack pointer setting
0021: ldc.b #H'00:8,ep ; Page register initialization
0022: ldc.b #H'00:8,dp ; Page register initialization
0023: ;
0024: mov.w #prog_start,R0 ; Transfer start address
0025: mov.w #prog_stop,R1 ; Transfer end address
0026: bsr tensou:16 ; Program transfer to RAM
0027:;
0028: ; Argument setting and subroutine call
0029: jsr @RAMSTR ; JMP SUB to RAM area program (prog_start)
0030: ; (All-mat constant write example)
0031:;
0032: main_end: ; End of write

```

```

0033:    bra    main_end
0034:;
0035:;
0036:; *****
0037:; * tensou SUB
0038:; * Copy RAM execution program to RAM
0039:; *****
0040:    .align    H'2
0041: tensou:equ $
0042:                ;Arguments R0 Transfer start address
0043:                ;          R1 Transfer end address
0044:    stm (R2-R3), @-sp ; save used registers
0045:                ; R2 Transfer destination RAM address
0046:                ; R3 Transfer data work
0047:    mov:i #RAMSTR,R2 ; Transfer destination address setting
0048:tensou01:
0049:;    mov.w @R0+,R3    ; ROM PROG DATA → R3
0050:;    mov.w R3,@R2+    ; R3 → RAM WRITE
0051:;    cmp.w R1,R0      ; R1:END R0:INCREASED ADDR.
0052:;    blt tensou01     ; R0=R1 → NEXT INSTRUCTION.
0053:;    ldm @spt,(R2-R3) ; Restre used registers
0054:;    rts              ; Subroutine return
0055:;
0056:;
0057:; *****
0058:; ** Start of program for transfer to RAM          **
0059:; *****
0060:    .align H'2
0061:prog_start:.equ $          ;start of program for transfer to RAM
0062:;
0063:;
0064:;
0065:;
0066:;
0067:; *****
0068:; all0_write SUB
0069:; Flash memory ALL H'00 write
0070:; *****
0071:    .align H'2
0072:all0_write: .equ $
0073:                ; Arguments R0 Return code 0: Normal
0074:                ;          1: Write error
0075:    stm(R1-R5),@-sp ; Save used registers
0076:                ; R0 Program data
0077:                ; R1 Target block specification
0078:                ; R2 Program address page specification
0079:                ; R3 Program address specification
0080:                ; Wait loop counts
0081:;
0082:    clr.b@WCR      ; No wait state insertion
0083:    ldc.w #H'0700,SR ; Disable interrupts during programming/erasing
0084:;
0085:    mov.w #(d'5*MHZ/d'8),R3
0086:                ; Set VPPE wait loop counter
0087:    mov.b #H'40,@FLMCR; Set VPPE bit
0088:all0_w01:
0089:scb/f R3,all0_w01 ; VPPE wait (5 µs or more)
0090:;
0091:                ; Argument setting and subroutine call
0092:    mov.w #H'FFFF,R1 ; Target block specification
0093:    mov.b #H'01,R2   ; Program address page specification
0094:all0_w02:
0095:    mov.w #H'0000,R3 ; Program address

```

```

0096:all0_w03:
0097:      mov.b #H'00,R0      ; Program data
0098:      bsr fwrite          ; 1-byte program
0099:      cmp.w #H'0000,R0    ; Return code check
0100:      bne all0_w05        ; If write error, end
0101:;
0102:      cmp.w #H'FFFF,R3    ; Last address of page?
0103:      beq all0_w04        ; If last address, next page
0104:      add.w #H'01,R3      ; Increment program address
0105:      bra allo_w03        ;
0106:all0_w04:
0107:      cmp.t #H'02,R2      ; Last page?
0108:      beg all0_w05        ; If last page, end
0109:      add.b #H'01,R2:     ; Page address + 1
0110:      bra all0_w02        ;
0111:all0_w05:
0112:      mov.w #(d'50*MHZ/d'8), R3
0113:      ; Set VPPE clear wait counter
0114:      clr.b @FLMCR       ; Clear VPPE bit
0115:all0_w06:
0116:      scb/f R3,all0_w06  ; VPPE clear wait (50 µs or more)
0117:;
0118:      ldm @sp+,(R1-R5)   ; Restore used registers
0119:      rts                ; Subroutine return
0120:all0_write_end: .equ $
0121:;
0122:;
0123:; *****
0124:; * fwrite SUB *
0125:; * To program one byte of flash memory (SUB) *
0126:; *****
0127:      .align H'2
0128:fwrite: .equ $
0129:      ; Arguments R0 Program data/return code
0130:      ; Return code 0: Normal
0131:      ; 1: Write error
0132:      ;R1 Target block
0133:      ;R2 Program address page
0134:      ;R3 Program address
0135:      stm(R2-R5),@-sp    ; Save used registers
0136:      ; R2 Decision count counter
0137:      ; R3 Wait loop counts
0138:      ; R4 Program address
0139:      ; R5 P time loop count
0140:      stc.b ep,@-sp     ; Save used page registers
0141:;
0142:      ldc.b R2,ep        ; Program address page
0143:      mov.w R3,R4        ; Program address
0144:;
0145:      mov.w R1,@EBR      ; Set target block in EBR1, EBR2
0146:;
0147:      mov.w #H'0001,R2   ; Decision count counter = 1
0148:      mov.w #((d'16*MHZ-d'4)/d'8),R5
0149:;      ; P time loop count initial value (approx. 16 µs)
0150:      mov.b R0,@R4      ; Dummy write (latch)
0151:fwrite 01:
0152:      mov.w R5,R3        ; Set loop counter
0153:      add.w R5,R5        ; P bit wait × 2
0154:      mov.w #H'A57A,@TCSR ; Set watchdog timer
0155:;      nop                ; Adjust so that scb/f wait is at even address
0156:      ; If odd, insert NOP
0157:      mov.b #H'41,@FLMCR ; Set P bit
0158:      ; Adjust so that scb/f wait is at even address

```

```

0159:fwrite 02:
0160:      scb/f R3,fwrite 02 ; P wait (initially 16 µs) programming time wait
0161:      mov.b #H'40,@FLMCR ; Clear P bit
0162:      mov.w #H'A500,@TCSR; Stop watchdog timer
0163;;
0164:      mov.w #(d'4*MHZ/d'8)R3
0165:      ; Set PV wait loop counter
0166:      mov.b #H'44,@FLMCR ; Set PV bit
0167:fwrite 03:
0168:      scb/f R3,fwrite 03 ; PV wait (4 µs or more)
0169:      cmp.b @R4,R0      ; Compare with programmed data
0170:      beg fwrite 05      ; Decision: If the same go to end processing
0171;;
0172:      add.w #H'01,R2      ; R2 = R2 + 1
0173:      cmp.w #H'0006,R2    ; Count decision (6 times)
0174:      bhi fwrite 05      ; If count > 6, end
0175:      mov.w #(d'4*MHZ/d'8),R3
0176:      ; Set post-PV-clearing wait counter
0177:      mov.b #H'40,@FLMCR ; Clear PV bit
0178:fwrite 04:
0179:      scb/f R3,fwrite 04 ; Post-PV-clearing wait (4 µs or more)
0180:      bra fwrite 01
0181;;
0182:fwrite 05:
0183:      mov.w #(d'4*MHZ/d'8),R3
0184:      ; Set post-PV-clearing wait counter
0185:      mov.b #H'40,@FLMCR ; Clear PV bit
0186:fwrite 06:
0187:      sob/f R3,fwrite 06 ; Post-PV-clearing wait (4 µs or more)
0188:      clr.w @EBR        ; Clear target block
0189;;
0190:      clr.w R0           ; Set return value (R0) = OK
0191:      cmp.w #H'0006,R2    ; Count decision
0192:      bls fwrite 07      ; No count overflow
0193:      mov.w #H'01, R0    ; Set return value (R0) = NG
0194:fwrite 07:
0195;;
0196:      ldc.b @sp+,ep      ; Restore used page register
0197:      ldm @sp+,(R2-R5)  ; Restore used registers
0198:      rts              ; Subroutine return
0199;;
0200;; *****
0201;;
0202:      ;
0203:      ;
0204:      ;
0205:prog_stop: .equ $      ;End of program for transfer to RAM
0206;;
0207;;
0208:      .end
0209:

```

#### 19.4.4 Erase Mode

To erase the flash memory, follow the erasing algorithm shown in figure 19-9. This erasing algorithm can erase data without subjecting the device to voltage stress or impairing the reliability of programmed data.

To erase flash memory, before starting to erase, first place all memory data in all blocks to be erased in the programmed state (program all memory data to H'00). If all memory data is not in the programmed state, follow the sequence described later to program the memory data to H'00. Select the flash memory areas to be erased with erase block registers 1 and 2 (EBR1 and EBR2). Next set the E bit in FLMCR, selecting erase mode. The erase time is the time during which the E bit is set. To prevent overerasing, divide the first three times into 6.25 ms, 12.5 ms, and 25 ms intervals, followed by 50 ms intervals repeated a maximum of 599 times, so that the total erase time does not exceed 30s. Overerasing, due to program runaway for example, can give memory cells a negative threshold voltage and cause them to operate incorrectly. Before selecting erase mode, set up the watchdog timer so as to prevent overerasing.

#### 19.4.5 Erase-Verify Mode

In program-verify mode, after data has been erased, it is read to check that it has been erased correctly. After the erase time has elapsed, exit erase mode (clear the E bit to 0) and select erase-verify mode (set the EV bit to 1). Before reading data in erase-verify mode, write H'FF dummy data to the address to be read. As a result of this dummy write, the erase-verify voltage is applied to the memory cells at the latched address. When the flash memory is read in this state, the data at the latched address is read. After selecting erase-verify mode, wait at least 4  $\mu$ s, plus at least 2  $\mu$ s for the dummy write to each address, before reading. If the read data has been successfully erased, perform the dummy write and erase-verify for the next address. If the read data has not been erased, select erase mode again and repeat the same erase and erase-verify sequence through the last address, until all memory data has been erased H'FF. Do not repeat the erase and erase-verify sequence more than 602 times, however.

Flowchart for Erasing One Block (ferase)

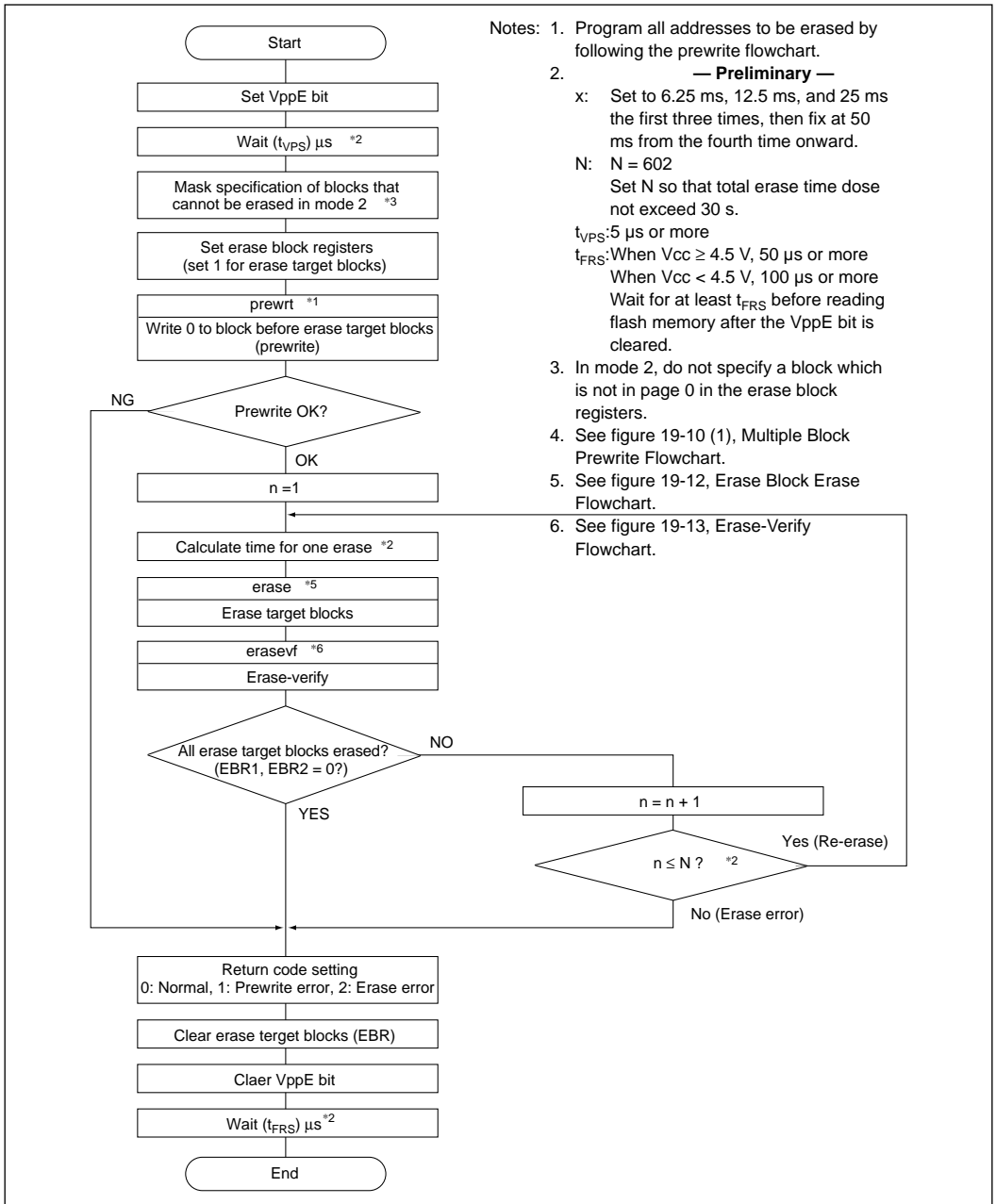


Figure 19-9 Multiple-Block Erasing Flowchart



### Flowchart for Prewriting Multiple Blocks (prewrt)

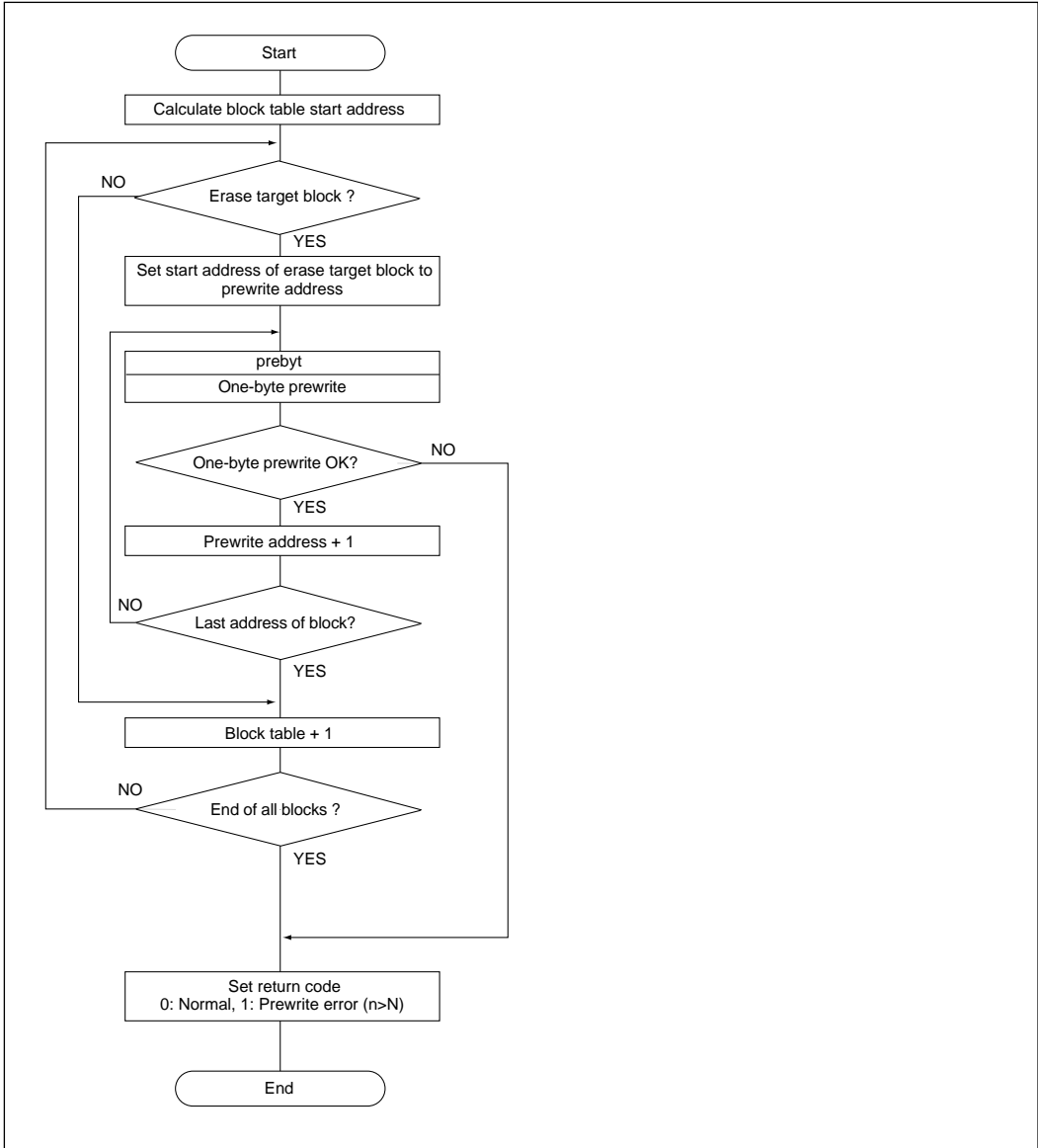
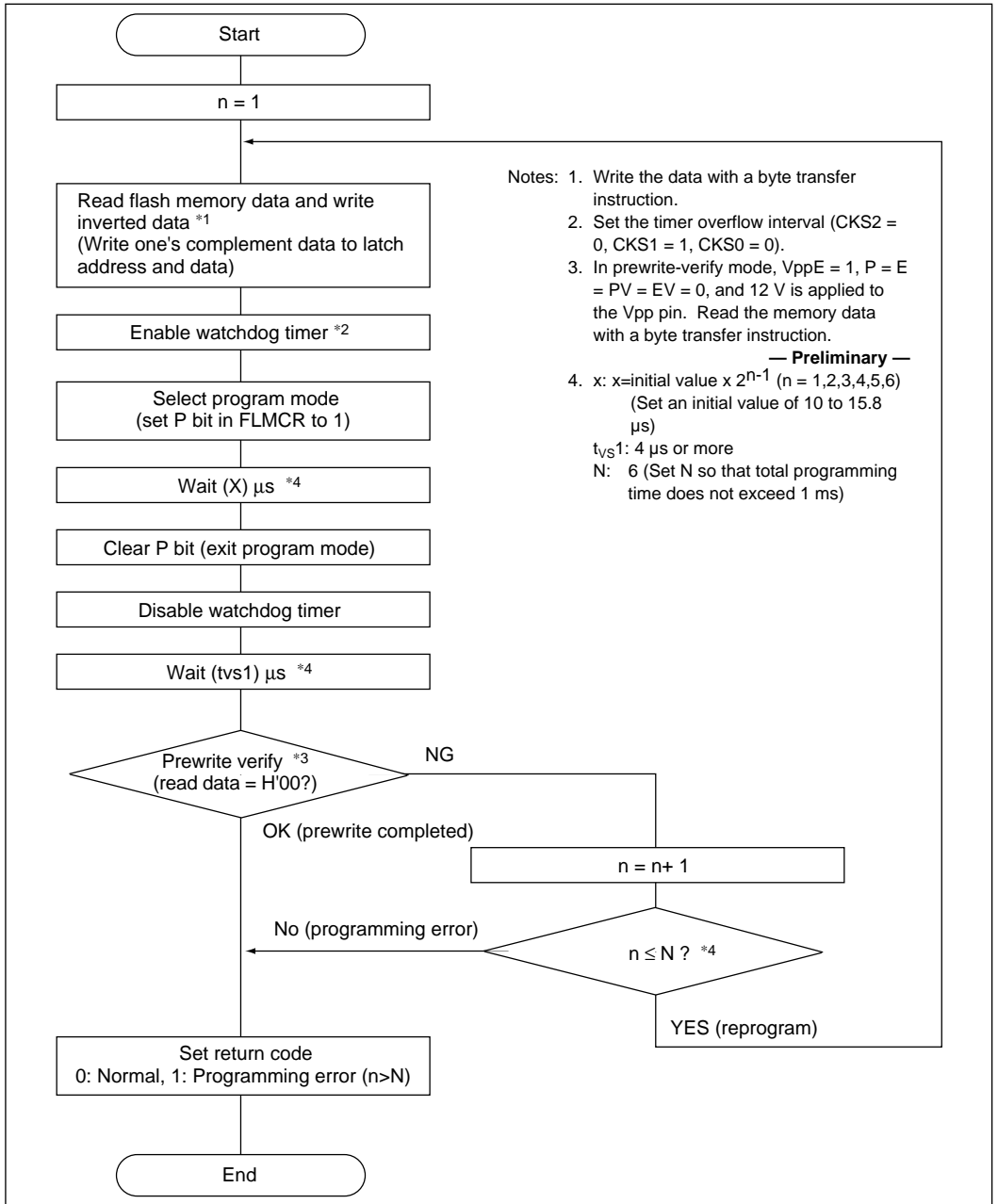


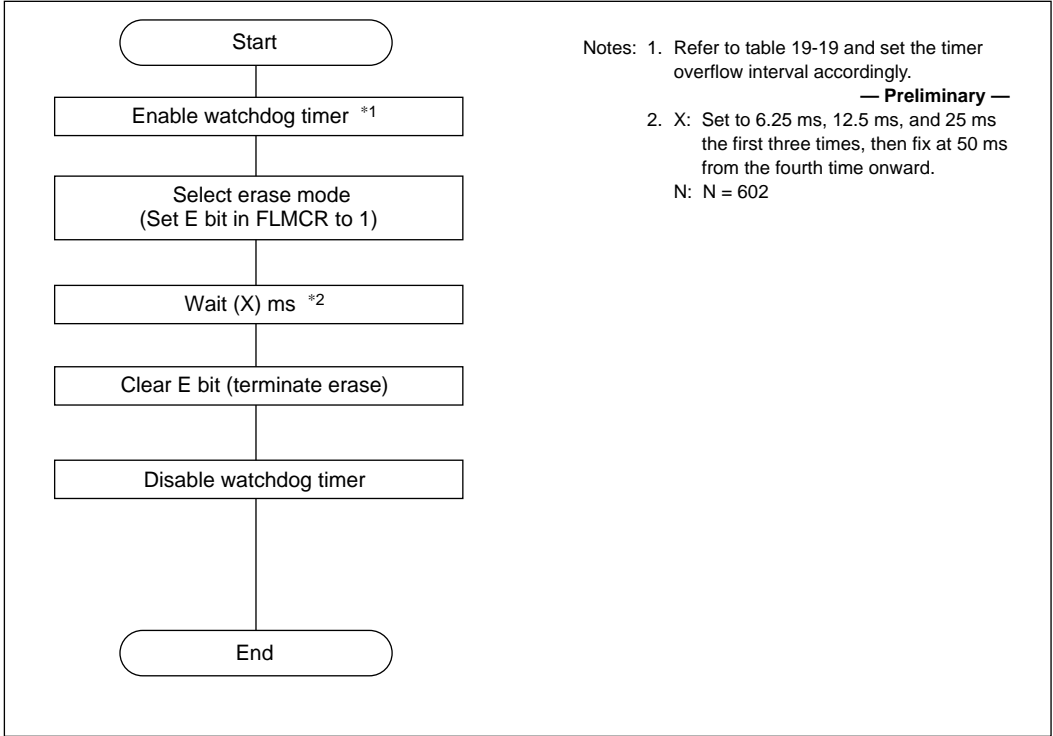
Figure 19-10 Multiple-Block Prewrite Flowchart

**Flowchart for Prewriting One Byte (prebyt)**



**Figure 19-11 One Byte-Block Prewrite Flowchart**

### Flowchart for Erasing Erase Target Block (erase)



**Figure 19-12 Erase Target Block Erase Flowchart**

### Erase-Verify Flowchart (erasevf)

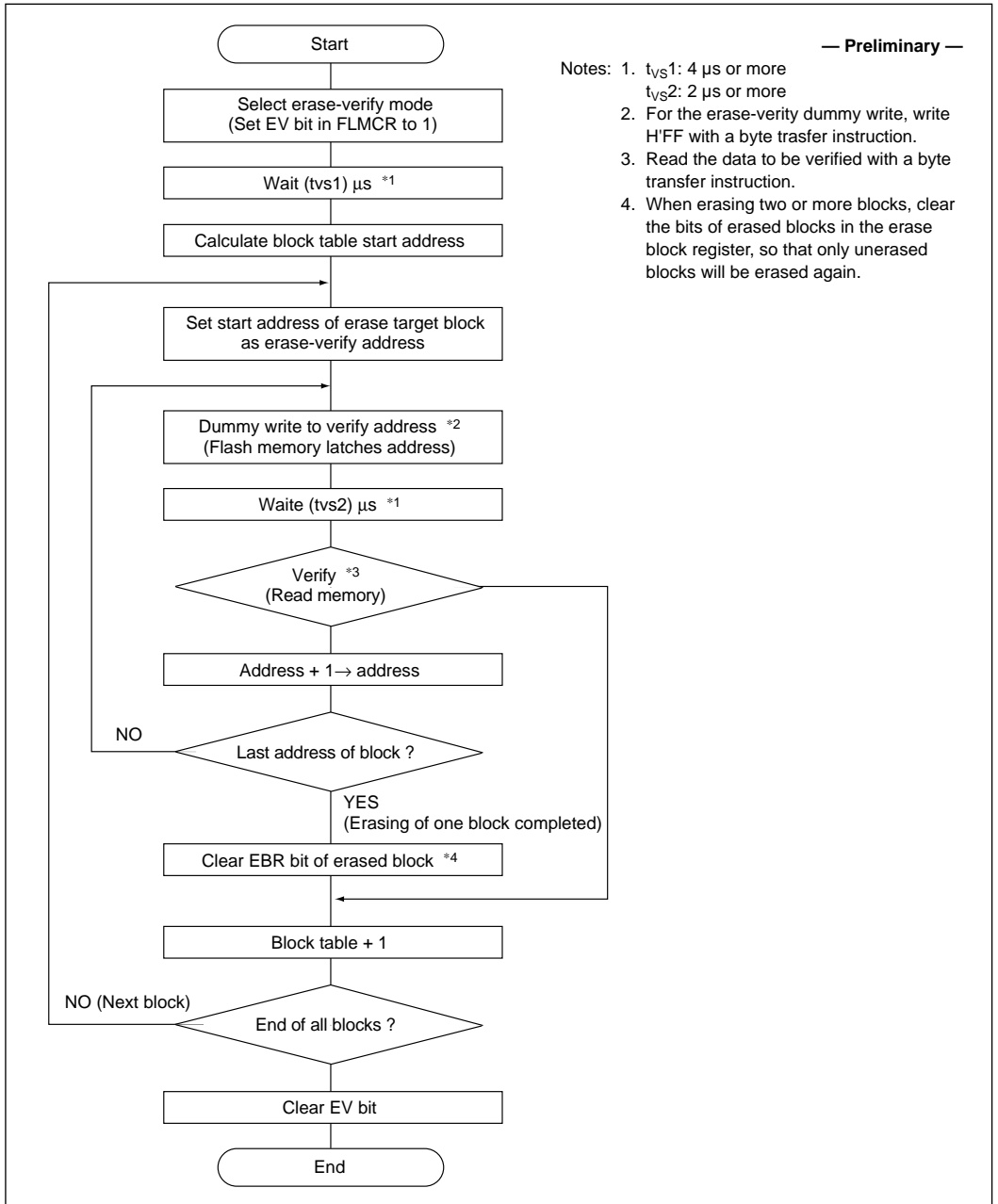


Figure 19-13 Multiple-BlockPrewrite Flowchart

**Loop Counter Values in Programs and Watchdog Timer Overflow Settings:** A wait time is necessary after a bit is set or cleared in the flash memory control register (FLMCR). In the program examples, wait times are provided by means of software loops. The software loop counter value depends on the operating frequency, and whether the scb/f instruction is located at an even address and in two-state access space, and whether wait state insertion is disabled. In these program examples, the calculation assumes an even address, two-state access space, and no wait state insertion. Examples of counter values for typical frequencies are shown in table 19-9.

The value set in TCSR to provide the watchdog timer overflow interval setting when erase mode is selected depends on the operating frequency. TCSR set values for different operating frequencies are shown in table 19-10.

As software loops are used, there is intrinsic error in the wait times, and the calculated value and actual time may not be the same. Therefore, initial values should be set so that the total programming time does not exceed 1 ms, and the total erase time does not exceed 30 s.

The set value for the watchdog timer is calculated on the basis of the number of instructions including the programming time and erase time from the time the watchdog timer is started until it stops. Therefore, no other instructions should be added between starting and stopping of the watchdog timer in these program examples.

The loop counter value for each frequency is calculated as shown below.

**Formulas:** Formulas for calculating loop counter value in program

**(1) Program time (P bit set) and calculation formula:** When the scb/f instruction is in two-state access space, and the branch destination is at an even address, the processing time is 4 states when the register value = 0 (no branch) and 8 states when the register value  $\geq 1$  (branch). Thus the calculation formula, with truncation of the decimal part, is as follows.

Loop counter value = ((Wait time ( $\mu$ s)  $\times$  operating frequency (MHz)) states-4 states)/8 states

**(2) Erase time (E bit set) calculation formula:** For the same access space as in (1) above, the calculation formula is as follows.

Loop counter value = ((Wait time ( $\mu$ s)  $\times$  operating frequency (MHz)) states - 14 states) / 18 states.

**(3) Wait time (after PV setting:  $t_{vs1}$ ), (after EV setting:  $t_{vs1}$ ), (after latching:  $t_{vs2}$ ) (after  $V_{ppE}$  clearing:  $t_{vps}$ ) calculation formula:** With the same number of states as in (1) above, the calculation formula, with rounding of the decimal part, is as follows.

Loop counter value = ((Wait time (μs) × operating frequency (MHz)) states-4 +4 states)/8 states

**Table 19-9 Example of Sample Software Loop Counter Values for Typical Operating Frequencies**

		Operating Frequency			
		f = 16 MHz	f = 10 MHz	f = 8 MHz	f = 2 MHz
		Counter Set Value	Counter Set Value	Counter Set Value	Counter Set Value
Program time (initial set value)	15.8 μs	H'001E	H'0013	H'000F	H'0003
t <sub>VS1</sub>	4 μs	H'0008	H'0005	H'0004	H'0001
t <sub>VS2</sub>	2 μs	H'0004	H'0002	H'0002	H'0001
t <sub>VPS</sub>	5 μs	H'000A	H'0006	H'0005	H'0001
Erase time (initial set value)	6.25ms	H'15B2	H'0D8F	H'0AD9	H'02B5

**Table 19-10 Watchdog Timer Overflow Interval Settings when Erase Mode is set**

Operating Frequency [MHz]	Variable
	TCSR Set Value
10 MHz ≤ frequency ≤ 16 MHz	H'A57F
2 MHz ≤ frequency < 10 MHz	H'A57E

**Sample Multiple-Block Erase Program:** This program uses the following registers.

R0: Specifies blocks to be erased (set as explained below)

/Return value (0: Normal, 1: Prewrite error, 2: Erase error)

R1: -/NG block (0: Normal, Other: NG block)

After a value is specified in R0 (erase block), an arbitrary block can be erased by calling the `ferase` subroutine. After the `ferase` subroutine ends, the return value is returned in R0 and an NG block in R1.

The wait time due to software looping after bit setting depends on the operating frequency. The relevant operating frequency can be specified by setting the `MHZ` symbol value. In this program the wait time (number of loops) is calculated on the assumption that the `scb/f` instruction is located at an even address in two-state access space (on-chip RAM). The read setup time ( $t_{FRS}$ ) after `VppE` bit clearing used here is for the case where  $V_{cc} \geq 4.5$  V. See section 22.2.4, Flash Memory Characteristics, for the wait time after a bit is set in the flash memory control register (FLMCR).

Each bit in R0 corresponds to a bit in the erase block registers (EBR1, EBR2). A bit map of R0 and an example of the method of calling the subroutine are shown below.

A bit map of R0 and an example setting for erasing specific blocks are shown next.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R0	LB7	LB6	LB5	LB4	LB3	LB2	LB1	LB0	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0
	Corresponds to EBR1								Corresponds to EBR2							

Example: to erase blocks LB2, SB7, and SB0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R0	LB7	LB6	LB5	LB4	LB3	LB2	LB1	LB0	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0
	Corresponds to EBR1								Corresponds to EBR2							

Setting 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1

`Ferase` subroutine calling is performed as shown for the `all_erase` subroutine in list 1.

## List 1: Sample Block Erase Program

```

0001:;*****
0002:;* ferase, src          (Ver. 0.13)    - Preliminary -          *
0003:;* Sample program for H8/539F flash memory block erasing      *
0004:;*                                                                *
0005:;*****
0006:;
0007:;
0008:MHZ .equ d'16          ; Depends on operating frequency (16 MHz)
0009:RAMSTR .equ H'EE80    ; Program transfer destination RAM address
0010:; Register addresses
0011:FLMCR .equ H'FEE0    ; Flash memory control register
0012:EBR .equ H'FEE2      ; Target block specification register
0013:TCSR .equ H'FE10     ; Timer control/status register
0014:WCR .equ H'FE14     ; Wait control register
0015:MDCR .equ H'FE19    ; Mode control register
0016:;
0017:;
0018:; .align H'2
0019:main:.equ $
0020:    ldc.b #H'00:B,tp    ; Stack page register setting
0021:    mov.w #H'FE80,sp   ; Stack pointer setting
0022:    ldc.b #H'00:8,ep   ; Page register initialization
0023:    ldc.b #H'00:8,dp   ; Page register initialization
0024:;
0025:    mov.w #prog_start,R0 ; Transfer start address
0026:    mov.w #prog_stop,R1 ; Transfer end address
0027:    bsr tensou:16      ; Program transfer to RAM
0028:;
0029:;                               ; Argument setting and subroutine call
0030:    jsr @RAMSTR        ; JMP SUB to RAM area program (prog_start)
0031:;                               ; (All-mat erase example)
0032:;
0033:main_end:                ; End of erase
0034:    bra main_end
0035:;
0036:;
0037:;*****
0038:;* tensou SUB                                                    *
0039:;* Copy RAM execution program to RAM                            *
0040:;*****
0041:; .align H'2
0042:tensou:.equ $
0043:;                               ; Arguments  R0 Transfer start address
0044:;                               ; R1 Transfer end address
0045:    stm(R2-R3),@-sp      ; Save used registers
0046:;                               ; R2 Transfer destination RAM address
0047:;                               ; R3 Transfer data work
0048:    mov:i #RAMSTR       ; Transfer destination address setting
0049:tensou 01:
0050:    mov.w @R0+,R3        ; RAM PROG DATA → R3
0051:    mov.w R3,@R2+        ; R3 → RAM WRITE
0052:    cmp.w R1,R0         ; R1:END R0:INCREASED ADDR.
0053:    blt tensou 01       ; R0=R1 → NEXT INSTRUCTION.
0054:    ldm @sp+,(R2-R3)    ; Restore used registers
0055:    rts                 ; Subroutine return
0056:;
0057:;
0058:;*****
0059:; **          Start of program for transfer to RAM          **

```



```

0060:;*****
0061:    .align H'2
0062:prog_start:.equ $          ; Start of program for transfer to RAM
0063:    ;
0064:    ;
0065:    ;
0066:;
0067:;
0068:;*****
0069:**  all_erase  SUB          *
0070:;    Flash memory all-mat erase          *
0071:;*****
0072:    .align H'2
0073:all_erase:.equ $
0074:    ; Arguments
0075:    ; R0-/Return code 0: Normal
0076:    ;                1: Prewrite error
0077:    ;                2: Erase error
0078:    ; R1-/NG target block
0079:stm(R2-R5),@-sp          ; Save used registers
0080:    ; R0 Erase-verify target block
0081:    ; R2 Wait loop counts
0082:;
0083:    clr.b @WCR          ; No wait state insertion
0084:    ldc.w #H'0700,sr    ; Disable interrupts during programming/erasing
0085:;
0086:    mov.w #(d'5*MHZ/d'8),R2
0087:    ; Set VPPE wait loop counter
0088:    mov.b #H'40,@FLMCR ; Set VPPE bit
0089:all_e01:
0090:    scb/f R2,all_e01    ; VPPE wait (5 µs or more)
0091:;
0092:    mov.w #H'FFFF,@EBR ; Erase target block specification
0093:    bsr erasevf:16      ; Erase-verify
0094:    mov.w @EBR,R0      ;
0095:    tst.w R0           ; Unerased block check
0096:    beq all_e02        ; If there is unerased block,
0097:    ; Argument setting and subroutine call
0098:    mov.w @EBR,R0      ; Specify unerased block as target block
0099:    bsr ferase         ; Erase block
0100:    ; Return R0 Return code (0, 1, 2)
0101:    ; R1 NG target block
0102:all_e02
0103:    mov.w #(d'50*MHZ/d'8),R2
0104:    ; Set VPPE clear wait counter
0105:    clr.b@FLMCR        ; Clear VPPE bit
0106:all_e03
0107:    scb/f R2,all_e03    ; VPPE clear wait (50 µs or more)
0108:;
0109:    ldm @sp+,(R2-R5)    ; Restore used registers
0110:    rts                ; Subroutine return
0111:all_erase_end: .equ $
0112:;
0113:;
0114:;*****
0115:* ferase SUB
0116:*  Flash memory block erase (SUB)
0117:;*****
0118:    .align H'2
0119:ferase:.equ $
0120:    ; Arguments      R0 Erase target block/return code
0121:    ; Return code   0: Normal
0122:    ;                1: Prewrite error

```

```

0123:                ;                2: Erase error
0124:                ; R1 - / NG target block
0125:    stm(R2-R5),@-sp    ; Save used registers
0126:                ; R0 E time loop count (lower)
0127:                ; R1 E time loop count (upper)
0128:                ; R2 Erase loop counter
0129:    stc.b ep,@-sp    ; Save used page register
0130:;;
0131:    cmp.b #H'c2,@MDCR    ; Mode check
0132:    bne ferase 01    ; If mode 2,
0133:    and.w #H'80FF,R0    ; mask target blocks except page 0
0134:ferase 01:
0135:    mov.w R0,@EBR    ; Set target block in EBR1, EBR2
0136:;;
0137:    bsr prewrt:16    ; Target block prewrite subroutine
0138:    tst.w R0    ; Return code check
0139:    bne ferase 05
0140:;;
0141:    mov.w #((d'6250*MHZ-d'14)/d'18),R0
0142:                ; E time loop count initial value (6.25 ms)
0143:    clr.w R1    ; E time loop count (upper)
0144:    mov.w #H'0001,R2
0145:ferase 02:
0146:    bsr erase:16    ; Erase subroutine
0147:    bsr erasevf:16    ; Erase-verify subroutine
0148:;;
0149:    tst.w @EBR
0150:    beq ferase 04    ; If @EBR = 0, end
0151:    add.w #H'01,R2
0152:;;
0153:    cmp.w #H'04,R2    ; Set loop counter
0154:    bhs ferase 03
0155:    add.w R0,R0    ; E time loop count (lower) x 2
0156:    addx.w R1,R1    ; E time loop count (upper) x 2
0157:ferase 03:
0158:;;
0159:    cmp.w #d'602,R2
0160:    bls ferase 02    ; If erase count > 602, NG end
0161:                ; If erase count is NG (>602),
0162:    mov.w #H'0002,R0    ; return code = erase error
0163:    mov.w @EBR,R1    ; R1 = NG block
0164:    bra ferase 05
0165:ferase 04:
0166:    clr.w R0    ; Return code = OK
0167:    clr.w R1    ; No NG block
0168:ferase 05
0169:;;
0170:    clr.w @EBR    ; Clear target block
0171:    ldc.b @sp+,ep    ; Restore used page register
0172:    ldm @sp+,(R2-R5)    ; Restore used registers
0173:    rts    ; Subroutine return
0174:;;
0175:;;
0176:;*****
0177:; * prewrt SUB    *
0178:; * Target block prewrite (SUB)    *
0179:;*****
0180:    .align2
0181:prewrt: .equ $
0182:                ; Arguments    @EBR Erase target block
0183:                ; R0    - /return code
0184:                ; Return code 0: Normal
0185:                ;                1: Prewrite error

```

```

0186:                                ; R1          - / NG target block
0187:    stm(R2-R4),@-sp              ; Save used registers
0188:                                ; R0  Argument - prewrite address page
0189:                                ; R1  Argument - prewrite address
0190:                                ; R2  Block table ADR
0191:                                ; R3  Target block bit number
0192:                                ; R4  Prewrite address
0193:    clr.w R0
0194:    mov.w #RAMSTR,R2              ; RAM program start address
0195:    add.w #blockadr,R2           ; + block table start address
0196:    sub.w #prog_start,R2        ; -> block table start address
0197:    clr.w R3                      ; Target block bit number
0198:prewrt 01:
0199:    btst.w R3,@EBR
0200:    beq prewrt 04                ; If R3 bit of @EBR= B'1, execute the following
0201:    mov.w @R2,R4
0202:    ldc.b R4,ep                  ; Set block start address page in ep
0203:    mov.w @(H'02,R2),R4         ; Set block start address in R4
0204:prewrt 02:
0205:    stc.b ep,R0                  ; Set prewrite address page in argument R0
0206:    mov.w R4,R1                  ; Set prewrite address in argument R1
0207:    bsr prebyt                  ; Prewrite (one byte) subroutine
0208:;
0209:    tst.w R0
0210:    bne prewrt 03              ; If prebyt return ≠ 0, end
0211:    cmp.w @(H'04,R2),R4
0212:    bhs prewrt 03              ; If R4 ≥ block end ADR, end
0213:    add.w #H'01,R4              ; R4 = R4 + 1
0214:    bra prewrt 02              ;
0215:prewrt 03:
0216:;
0217:prewrt 04:
0218:    add.w #H'0006,R2           ; Block table next line address
0219:;
0220:    tst.w R0
0221:    bne prewrt 05              ; If prebyt return ≠ 0, end
0222:    add.w #H'01,R3
0223:    cmp.w #H'0F,R3
0224:    bls prewrt 01              ; If target block bit number > 15, end
0225:prewrt 05:
0226:    clr.w R1
0227:    tst.w R0
0228:    beq prewrt 06              ; If prebyt return ≠ 0, execute the following
0229:    bset.w R3,R1                ; Set 1 in R3 bit of R1
0230:prewrt 06:
0231:;
0232:    ldm @sp+,(R2-R4)           ; Restore used registers
0233:    rts                        ; Subroutine return
0234:;
0235:;
0236:;*****
0237:; * prebyt SUB *
0238:; * One-byte prewrite (SUB) *
0239:;*****
0240:    .align 2
0241:prebyt:.equ $
0242:                                ; Arguments R0 Prewrite address page/return code
0243:                                ; Return code 0: Normal
0244:                                ; 1: Prewrite error
0245:                                ; R1 Prewrite address
0246:    stm (R2-R5),@-sp           ; Save used registers
0247:                                ; R2 Decision count counter
0248:                                ; R3 Wait loop counts

```

```

0249:                                ; R4  Prewrite address
0250:                                ; R5  P time loop count
0251:    stc.b ep,@-sp                ; Save used page register
0252:;
0253:    ldc.b R0,ep                    ; Prewrite address page
0254:    mov.w R1,R4                    ; Prewrite address
0255:;
0256:    mov.w #H'0001,R2                ; Decision count counter = 1
0257:    mov.w #((d'16*MHZ-d'4)/d'8),R5 ; P time loop count initial value (approx. 16 µs)
0258:;
0259:prebyt 01:
0260:    clr.b @R4                      ; Dummy write (H'00 compulsory latch)
0261:    mov.w R5,R3                    ; Set loop counter
0262:    add.w R5,R5                    ; P bit wait × 2
0263:    mov.w #H'A57A,@TCSR            ; Set watchdog timer
0264:    nop                            ; Adjust so that scb/f wait is at even address
0265:    mov.b #H'41,@FLMCR            ; Set P bit
0266:;                                ; Adjust so that scb/f wait is at even address
0267:prebyt 02:
0268:    scb/f R3,prebyt 02            ; P wait (initially 16 µs) prewrite time wait
0269:    mov.b #H'40,@FLMCR            ; Reset P bit
0270:    mov.w #H'A500,@TCSR            ; Stop watchdog timer
0271:;
0272:    mov.w #(d'4*MHZ/d'8),R3        ;
0273:;                                ; Set P reset wait loop counter
0274:prebyt 03:
0275:    scb/f R3,prebyt 03            ; P bit reset wait (4 µs or more)
0276:    tst.b@R4
0277:    beq prebyt 04
0278:    add.w #H'01,R2                ; R2 = R2 + 1
0279:    cmp.w #H'0006,R2              ; Count decision (6 times)
0280:    bhi prebyt 04                ; If count > 6, end
0281:    bra prebyt 01
0282:;
0283:prebyt 04:
0284:;
0285:    clr.w R0                      ; Set return value (R0) = OK
0286:    cmp.w #H'0006,R2              ; Count decision
0287:    bls prebyt 05                ; If count overflow, execute the following
0288:    mov.w #H'01,R0                ; Set return value (R0) = prewrite NG
0289:prebyt 05:
0290:;
0291:    ldc.b @sp+,ep                ; Restore used page register
0292:    ldm @sp+,(R2-R5)              ; Restore used registers
0293:    rts                            ; Subroutine return
0294:;
0295:;
0296:;*****
0297:; * erase      SUB                                *
0298:; * Flash memory erase (SUB)                    *
0299:;*****
0300:    .align 2
0301:erase: .equ $
0302:                                ; Arguments      @EBR          Erase target block
0303:                                ; R0 Erase wait loop count (lower)
0304:                                ; R1 Erase wait loop count (upper)
0305:    stm(R2-R5),@-sp              ; Save used registers
0306:                                ; R2 Erase wait loop count (lower)
0307:                                ; R3 Erase wait loop count (upper)
0308:;
0309:    mov.w R0,R2                  ; Erase wait loop count (lower)
0310:    mov.w R1,R3                  ; Erase wait loop count (upper)
0311:    mov.w #H'A57F,@TCSR          ; Set watchdog timer

```

```

0312:      nop                ; Adjust so that erase01: is at even address
0313:      mov.b #H'42,@FLMCR  ; Set E bit
0314:                ; Adjust so that erase01: is at even address
0315:erase 01:
0316:      nop                ; nop instructions to increase wait time
0317:      nop
0318:      nop
0319:      nop
0320:      nop
0321:      scb/f R2,erase 01   ; Erase time wait (initially 6.25 ms)
0322:      scb/f R3,erase 01
0323:      mov.b #H'40,@FLMCR  ; Reset E bit
0324:      mov.w #H'A500,@TCSR ; Stop watchdog timer
0325:      mov.w #(d'4*MHZ/d'8),R2
0326:                ; Set E reset wait loop counter
0327:erase 02:
0328:      scb/f R2,erase 02   ; E bit reset wait (4 µs or more)
0329:;
0330:      ldm @sp+,(R2-R5)    ; Restore used registers
0331:      rts                ; Subroutine return
0332:;
0333:;
0334:;*****
0335:;* erasevf  SUB                                *
0336:;* Erase-verify (SUB)                        *
0337:;*****
0338:      .align H'2
0339:erasevf:.equ $
0340:                ; Arguments @EBR Erase target block/unerased block
0341:      stm(R2-R5),@-sp    ; Save used registers
0342:                ; R2 Block table ADR
0343:                ; R3 Target block bit number
0344:                ; R4 Verify address
0345:                ; R5 Wait loop counts
0346:      stc.b ep,@-sp     ; Save used page register
0347:;
0348:      mov.w #(d'4*MHZ/d'8),R5
0349:                ; Set EV wait loop counter
0350:      mov.b #H'48,@FLMCR ; Set EV bit
0351:erasevf 01:
0352:      scb/f R5,erasevf 01 ; EV wait (4 µs or more)
0353:;
0354:      mov.w #RAMSTR,R2   ; RAM program start
0355:      add.w #blockadr,R2 ; + block table relative address
0356:      sub.w #prog_start,R2 ; -> block table start address
0357:      clr.w R3          ; Target block bit number
0358:erasevf 02:
0359:      btst.w R3,@ EBR
0360:      beq erasevf 07     ; If R3 bit of @EBR= B'1, execute the following
0361:      mov.w @R2,R4
0362:      ldc.b R4,ep       ; Set block start address page in ep
0363:      mov.w @(H'02,R2),R4 ; Set block start address in R4
0364:erasevf 03:
0365:      mov.w #(d'2 *MHZ/d'8),R5
0366:                ; Set post-latch wait loop counter
0367:      mov.b #H'FF@R4    ; Dummy write (address latch)
0368:erasevf 04:
0369:      scb/f R5,erasevf 04 ; Post-latch wait (2 µs or more)
0370:;
0371:      cmp.b #H'FF,@R4   ; Verify
0372:      bne erasevf 06     ; If target address is unerased, end
0373:      cmp.w @(H'04,R2),R4
0374:      bhs erasevf 05     ; If R4 >= block end ADR, end

```

```

0375:      add.w #H'01,R4      ; R4 = R4 + 1
0376:      bra erasevf 03      ;
0377:erasevf 05:
0378:      bclr.w R3,@EBR      ; Clear @EBR target block bit
0379:erasevf 06:
0380:;;
0381:erasevf 07:
0382:      add.w #H'0006,R2    ; Block table next line address
0383:
0384:      add.w #H'01,R3
0385:      cmp.b #H'0F,R3
0386:      bls erasevf 02      ; If target block bit number > 15, end
0387:;;
0388:      mov.w #(d'4*MHZ/d'8),R5
0389:      ; Set post-EV-clearing wait loop counter
0390:      mov.b #H'40,@FLMCR ; Clear EV bit
0391:erasevf 08:
0392:      scb/f R5,erasevf 08 ; Post-EV-clearing wait (4 µs or more)
0393:;;
0394:      ldc.b @sp+,ep        ; Restore used page register
0395:      ldm @sp+,(R2-R5)    ; Restore used registers
0396:      rts                  ; Subroutine return
0397:;;
0398:;;
0399:;*****
0400:; * blockadr  DATA *
0401:; *      Flash Memory Block Addresses *
0402:;*****
0403:      .align H'2
0404:blockadr: .equ $
0405:      .data.w H'0001,H'3000,H'31FF      ;SB0
0406:      .data.w H'0001,H'3200,H'33FF      ;SB1
0407:      .data.w H'0001,H'3400,H'35FF      ;SB2
0408:      .data.w H'0001,H'3600,H'37FF      ;SB3
0409:      .data.w H'0001,H'3800,H'39FF      ;SB4
0410:      .data.w H'0001,H'3A00,H'3BFF      ;SB5
0411:      .data.w H'0001,H'3C00,H'3DFF      ;SB6
0412:      .data.w H'0001,H'3E00,H'3FFF      ;SB7
0413:;;
0414:      .data.w H'0002,H'C000,H'FFFF      ;LB0
0415:      .data.w H'0002,H'8000,H'BFFF      ;LB1
0416:      .data.w H'0002,H'4000,H'7FFF      ;LB2
0417:      .data.w H'0002,H'0000,H'3FFF      ;LB3
0418:      .data.w H'0001,H'C000,H'FFFF      ;LB4
0419:      .data.w H'0001,H'8000,H'BFFF      ;LB5
0420:      .data.w H'0001,H'4000,H'7FFF      ;LB6
0421:      .data.w H'0001,H'0000,H'2FFF      ;LB7
0422:;;
0423:;*****
0424:;
0425:      ;
0426:      ;
0427:      ;
0428:prog_stop: .equ $      ; End of program for transfer to RAM
0429:;;
0430:;;
0431:      .end
0432:

```

## 19.4.7 Prewrite-Verify Mode

Prewrite-verify mode is a verify mode used after zeroizing all bits to equalize their threshold voltages before erasing them.

To program all bits, use the one-byte prewrite algorithm shown in figure 19-10. Use this procedure to set all flash memory data to H'00 after programming. After the necessary programming time has elapsed, exit program mode (by clearing the P bit to 0) and select prewrite-verify mode (leave the P, E, PV, and EV bits all cleared to 0). In prewrite-verify mode, a prewrite-verify voltage is applied to the memory cells at the read address. If the flash memory is read in this state, the data at the read address will be read. After selecting prewrite-verify mode, wait 4  $\mu$ s before reading.

Note: For a sample prewriting program, see the prewrite subroutine in the sample erasing program.

## 19.4.8 Protect Modes

Flash memory can be protected from programming and erasing by software or hardware methods. These two protection modes are described below.

**Software Protection:** Prevents transitions to program mode and erase mode even if the P or E bit is set in the flash memory control register (FLMCR). Details are as follows.

Protection	Description	Function		
		Program	Erase	Verify*1
Block protect	Individual blocks can be program/erase-protected by the erase block registers (EBR1 and EBR2). If EBR1 and EBR2 are both set to H'00, all blocks are program/erase-protected.	Disabled	Disabled	Enabled
Emulation protect*2	When the OVLPE bit is set in the flash memory emulation register (FMLER), all blocks are protected from both programming and erasing.	Disabled	Disabled*3	Enabled

**Hardware Protection:** Suspends or disables the programming and erasing of flash memory, and resets the flash memory control register (FLMCR) and erase block registers (EBR1 and EBR2). The error-protect function permits the P and E bits to be set, but prevents transitions to program mode and erase mode. Details of hardware protection are as follows.

Protection	Description	Function		
		Program	Erase	Verify*1
Programing voltage ( $V_{PP}$ ) protect	When $V_{PP}$ is not applied, FLMCR, EBR1, and EBR2 are initialized, disabling programming and erasing. To obtain this protection, $V_{PP}$ should not exceed $V_{CC}$ .*4	Disabled	Disabled*3	Disabled
Reset and standby protect	When a reset occurs (including a watchdog timer reset) or standby mode is entered, FLMCR, EBR1, and EBR2 are initialized, disabling programming and erasing. Note that $\overline{RES}$ input does not ensure a reset unless the $\overline{RES}$ pin is held low for at least 20 ms at power-up (to enable the oscillator to settle), or at least six system clock cycles ( $6\phi$ ) during operation.*5	Disabled	Disabled*3	Disabled
Error protect	If an operational error is detected during programming or erasing of flash memory (FLER = 1), the FLMCR, EBR1, and EBR2 settings are preserved, but programming or erasing is aborted immediately. This type of protection can be cleared only by a reset by means of the $\overline{RES}$ pin*6 or hardware standby.	Disabled	Disabled*3	Enabled

- Notes:
1. Three modes: program-verify, erase-verify, and prewrite-verify.
  2. Except in RAM areas overlapped onto flash memory.
  3. All blocks are erase-disabled. It is not possible to specify individual blocks.
  4. For details, see section 19.7, "Flash Memory Programming and Erasing Precautions."
  5. See section 4.2.2, "Reset Sequence" and section 19.7, "Flash Memory Programming and Erasing Precautions."
  6. In the H8/538F, this includes the FLER bit clearing conditions and a watchdog timer reset, but in the H8/539F only  $\overline{RES}$  pin reset input is applicable.

**Error Protect:** This protection mode is entered if one of the error conditions that set the FLER bit in FLMSR\*1 is detected while flash memory is being programmed or erased (while the P bit or E bit is set in FLMCR\*3). These conditions can occur if microcontroller operations do not follow the programming or erasing algorithm. Error protect is a flash-memory state. It does not affect other microcontroller operations.

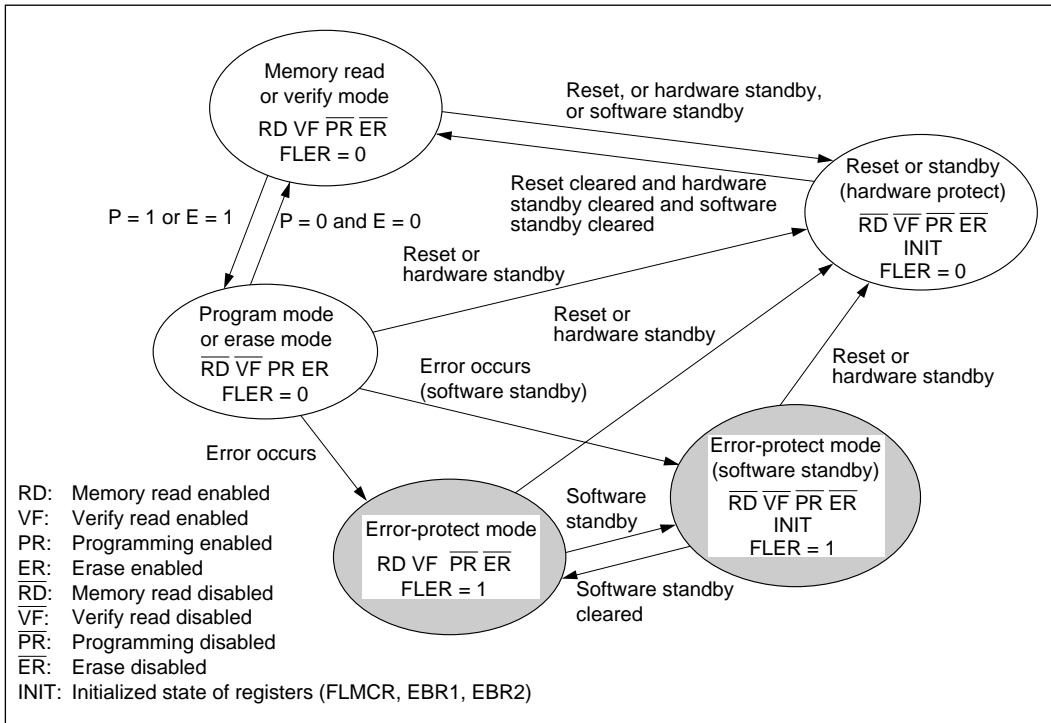
In this state the settings of the flash memory control register (FLMCR) and erase block registers (EBR1 and EBR2) are preserved,\*2 but program mode or erase mode is terminated as soon as the



error is detected. While the FLER bit is set, it is not possible to enter program mode or erase mode, even by setting the P bit or E bit in FLMCR again. The PV and EV bits in FLMCR remain valid, however. Transitions to verify modes are possible in the error-protect state.

The error-protect state can be cleared only by a reset by means of the  $\overline{\text{RES}}$  pin or entry to hardware standby mode.

- Notes:
1. For the detailed conditions that set the FLER bit, see section 19.2.4, “RAM Control Register (RAMCR).”
  2. It is possible to write to the FLMCR, EBR1, and EBR2 registers. Note that a transition to software standby mode in the error protect state initializes these registers.
  3. Note that NMI input is disabled when the P bit or E bit is set. For details, see section 19.4.9, “NMI Input Masking.”



**Figure 19-14 Flash Memory State Transitions**

**[In Modes 2, 4, and 7 (On-Chip ROM Enabled) when Programming Voltage ( $V_{pp}$ ) is Applied]**

The purpose of error-protect mode is to prevent overprogramming or overerasing damage to flash memory by detecting abnormal conditions that occur if the programming or erasing algorithm is not followed, or if a program crashes while the flash memory is being programmed or erased.

This protection function does not cover abnormal conditions other than the setting conditions of the flash memory error bit (FLER), however. Also, if too much time elapses before the error-protect state is reached, the flash memory may already have been damaged. This function accordingly does not offer foolproof protection from damage to flash memory.

To prevent abnormal operations, when programming voltage ( $V_{PP}$ ) is applied, follow the programming and erasing algorithms correctly, and keep microcontroller operations under constant internal and external supervision, using the watchdog timer for example. If a transition to error-protect mode occurs, the flash memory may contain incorrect data due to errors in programming or erasing, or it may contain data that has been insufficiently programmed or erased because of the suspension of these operations. Boot mode should be used to recover to a normal state.

Overprogramming or overerasing may prevent normal start-up of boot mode.

### 19.4.9 NMI Input Masking

NMI input is disabled when flash memory is being programmed or erased (when the P or E bit is set in FLMCR). NMI input is also disabled while the boot program is executing in boot mode, until the branch to the on-chip RAM area takes place.\*<sup>1</sup> There are three reasons for this.

1. NMI input during programming or erasing might cause a violation of the programming or erasing algorithm. Normal operation could not be assured.
2. In the NMI exception-handling sequence during programming or erasing, the vector would not be read correctly.\*<sup>2</sup> The result might be a program runaway.
3. If NMI input occurred during boot program execution, the normal boot-mode sequence could not be executed.

For these reasons, under certain conditions the H8/539F masks the normally nonmaskable NMI input. This masking does not, however, ensure normal programming, erasing, and other microcontroller operations. NMI requests should be disabled externally whenever  $V_{PP}$  is applied.

NMI input is also disabled in the error-protect state and while the P or E bit remains set in the flash memory control register (FLMCR) during emulation of flash memory using RAM.

Notes: 1. The disabled state lasts until the branch to the boot program area in on-chip RAM (addresses H'EE80 to HF37F) that takes place as soon as the transfer of the user program is completed. After the branch to the RAM area, NMI input is enabled except during programming or erasing. NMI interrupt requests must therefore be disabled externally until the user program has completed initial programming (including the vector table and the NMI interrupt-handling program).

2. In this case, the vector may not be read correctly for the following two reasons.
  - a. If flash memory is read while being programmed or erased (while the P or E bit is set in FLMCR), correct read data will not be obtained. Undetermined values are returned.
  - b. If the NMI entry in the vector table has not been programmed yet, NMI exception handling will not be executed correctly.

## 19.5 Flash Memory Emulation by RAM

Erasing and programming flash memory takes time, which can make it difficult to tune parameters and other data in real time. If necessary, real-time updates of flash memory can be emulated by overlapping the small-block flash-memory area (H'3000 to H'3FFF) with part of the RAM. This RAM reassignment is performed using bits 7, 3, 2, and 1 of the flash memory emulation register (FLMER) and bits 2 to 0 of the RAM control register (RAMCR).

After a flash memory area has been overlapped by RAM, it can be accessed from two address areas: the overlapped flash memory area (mapping RAM area), and the original RAM area (the actual RAM area). Bits 7, 3, 2, and 1 of FLMER and bits 2 to 0 of RAMCR are valid in Modes 2, 4, and 7. In other modes, they always read 0 and the RAM area cannot be reassigned. When the flash memory emulation function is used, bits 7 and 5 of RAMCR should both be set (RAME1 = RAME2 = 1). Table 19-10 indicates how to assign a mapping RAM area. Bits 7 and 5 of RAMCR should both be set (RAME1 = RAME2 = 1). Table 19-10 indicates how to assign a mapping RAM area.

### Flash Memory Emulation Register (FLMER)

Bit	7	6	5	4	3	2	1	0
	OVLPE	—	—	—	A11E	A10E	A9E	—
Initial value	0	1	1	1	0	0	0	1
R/W	R/W	—	—	—	R/W	R/W	R/W	—

**Table 19-11 ROM Area Setting**

ROM Area (Mapping RAM Area)	FLMER Register Bit 7*1	RAMCR Register			Overlap Function	Program /Erase Protection
	OVLPE	Bit 2*1	Bit 1*1	Bit 0*1		
		RAM2	RAM1	RAM0		
—	0	0/1	0/1	0/1	Disabled	Disabled
H'3000-H'31FF	1	0	0	0	Enabled	Enabled
H'3200-H'33FF	1	0	0	1	Enabled	Enabled
H'3400-H'35FF	1	0	1	0	Enabled	Enabled
H'3600-H'37FF	1	0	1	1	Enabled	Enabled
H'3800-H'39FF	1	1	0	0	Enabled	Enabled
H'3A00-H'3BFF	1	1	0	1	Enabled	Enabled
H'3C00-H'3DFF	1	1	1	0	Enabled	Enabled
H'3E00-H'3FFF	1	1	1	1	Enabled	Enabled

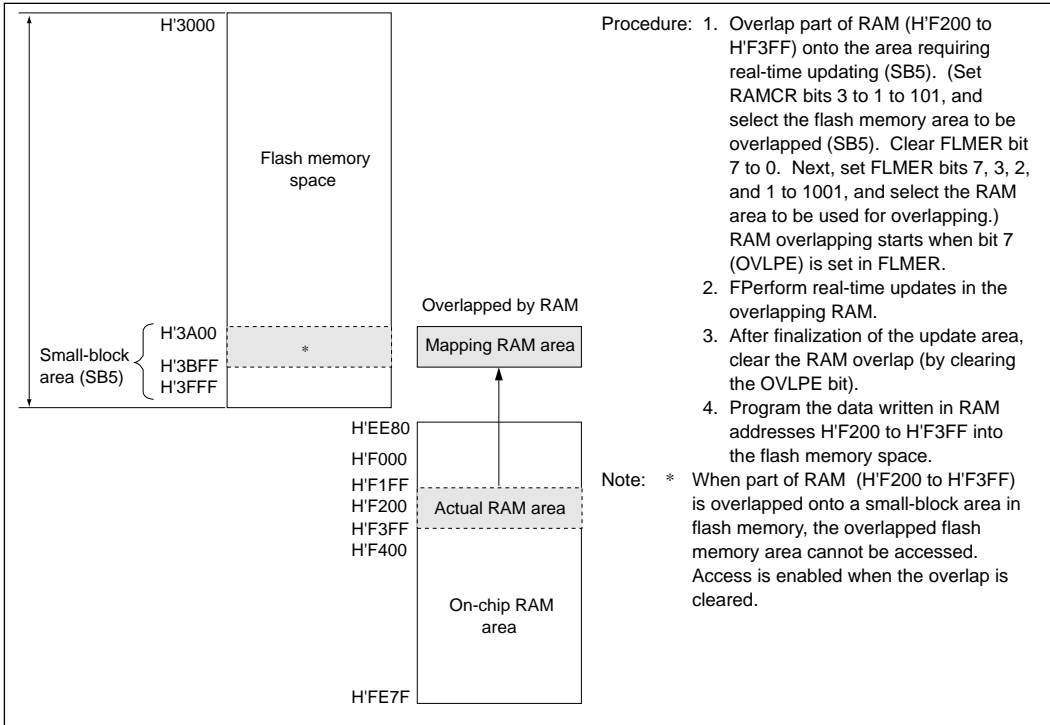
**Table 19-12 RAM Area\*2 Setting**

RAM Area*2 (Actual RAM Area)	FLMER Register			RAMCR Register	
	Bit 3*1	Bit 2*1	Bit 1*1	Bit 7*1	Bit 5*1
	A11E	A10E	A9E	RAME2	RAME1
H'F000-H'F1FF (512 bytes)	0	0	0	1	1
H'F200-H'F3FF (512 bytes)	0	0	1	1	1
H'F400-H'F5FF (512 bytes)	0	1	0	1	1
H'F600-H'F7FF (512 bytes)	0	1	1	1	1
H'F800-H'F9FF (512 bytes)	1	0	0	1	1
H'FA00-H'FBFF (512 bytes)	1	0	1	1	1
H'FC00-H'FDFF (512 bytes)	1	1	0	1	1
Use prohibited *3	1	1	1	1	1
Use prohibited *4	0/1	0/1	0/1	0/1 *4	0/1 *4

- Notes: 1. Bits 7, 3, 2, and 1 of the flash memory emulation register (FLMER) and bits 2 to 0 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7.  
(In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{PP}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{pp}$  voltage.)
2. RAM area overlapped onto flash memory
  3. When A11E and A10E are both set to 1, A9E is always cleared to 0.
  4. Use prohibited when RAME1 = 0 or RAME2 = 0. (Can be used when RAME1 = RAME2 = 1.)

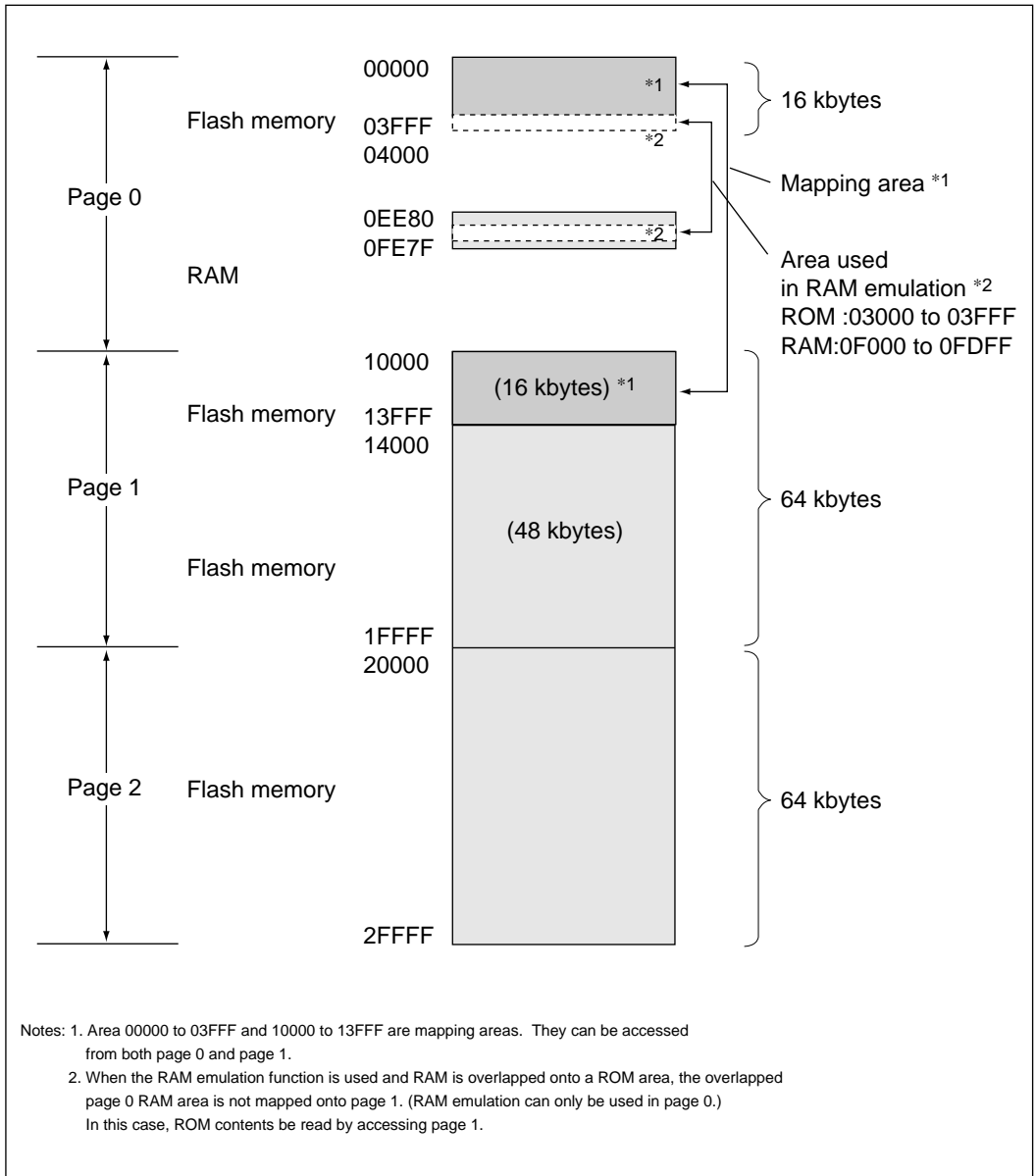
## Example of Emulation of Real-Time Flash-Memory Updating

In the following example, RAM area H'F200 to H'F3FF is overlapped onto the SB5 flash memory area (H'3A00 to H'3BFF).



**Figure 19-15 Example of RAM Overlapping**

## Notes on emulation RAM access



**Figure 19-16 Notes on Emulation RAM Access**

- Notes on applying and releasing the programming voltage ( $V_{PP}$ )

As in on-board program mode, care is required when applying and releasing  $V_{PP}$  to prevent erroneous programming or erasing. To prevent erroneous programming and erasing due to program runaway during  $V_{PP}$  application, in particular, the watchdog timer should be set when the P and E bits in the flash memory control register (FLMCR) are set, even while the emulation function is being used.

For details, see section 19.7, “Flash Memory Programming and Erasing Precautions.”

- NMI input disabling conditions

When the emulation function is used, NMI input is disabled when the P bit or E bit in the flash memory control register (FLMCR) is set, in the same way as with normal programming and erasing.

The P and E bits are cleared by a reset (including a watchdog timer reset), in standby mode, and when 12 V is not being applied to  $V_{PP}$ .

## 19.6 PROM Mode

### 19.6.1 PROM Mode Setting

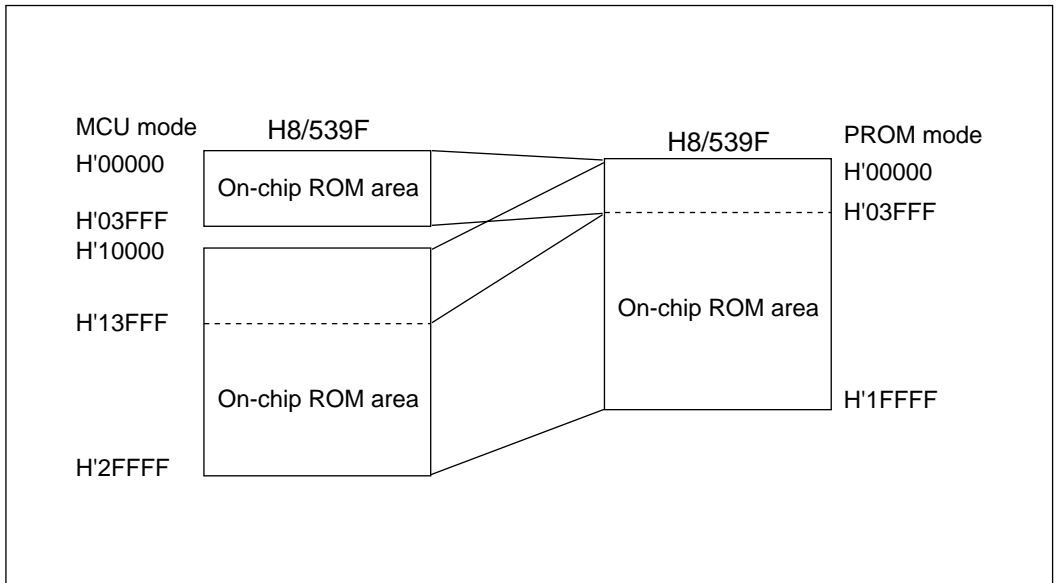
The on-chip flash memory of the H8/539F can be programmed and erased not only in the on-board programming modes but also in PROM mode, using a general-purpose PROM programmer. In PROM mode, make sure that the socket adapter listed in table 19-13 is used.

### 19.6.2 Socket Adapter and Memory Map

Programs can be written and verified by attaching a special 112-pin/32-pin socket adapter to the general-purpose PROM programmer. Table 19-13 gives ordering information for the socket adapter. Figure 19-18 shows a memory map in PROM mode.

**Table 19-13** Socket Adapter

Microcontroller	Package	Socket Adapter
HD64F5389F	112-pin plastic QFP (FP-112)	HS539FESH01H



**Figure 19-17** Memory Map in PROM Mode

**Note:** Use an appropriate tool when inserting the device in the IC socket and removing it. For example, the tool shown in table 19-14 can be used.



**Table 19-14 Example of Tool**

Manufacturer	Part Number
ENPLAS Corporation	HP-100 (vacuum pen)

**19.6.3 Operation in PROM Mode**

The program/erase/verify specifications in PROM mode are the same as for the standard HN28F101 flash memory. The H8/539F does not have a device recognition code, so the programmer cannot read the device name automatically. Table 19-15 indicates how to select the various operating modes in PROM mode. Table 19-15 indicates how to select the various operating modes in PROM mode, and table 19-16 lists the commands used in PROM mode.

**Table 19-15 Operating Mode Selection in PROM Mode**

Mode		Pins						A <sub>16</sub> to A <sub>0</sub>
		V <sub>PP</sub>	V <sub>CC</sub>	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	D <sub>7</sub> to D <sub>0</sub>	
Read	Read	V <sub>CC</sub>	V <sub>CC</sub>	L	L	H	Data output	Address input
	Output disable	V <sub>CC</sub>	V <sub>CC</sub>	L	H	H	High impedance	
	Standby	V <sub>CC</sub>	V <sub>CC</sub>	H	X	X	High impedance	
Command write	Read	V <sub>PP</sub>	V <sub>CC</sub>	L	L	H	Data output	
	Output disable	V <sub>PP</sub>	V <sub>CC</sub>	L	H	H	High impedance	
	Standby	V <sub>PP</sub>	V <sub>CC</sub>	H	X	X	High impedance	
	Write	V <sub>PP</sub>	V <sub>CC</sub>	L	H	L	Data input	

**Legend**

L: Low level  
H: High level  
V<sub>PP</sub>: V<sub>PP</sub> level  
V<sub>CC</sub>: V<sub>CC</sub> level  
X: Don't care

**Table 19-16 PROM Mode Commands**

Command	Cycles	1st Cycle			2nd Cycle		
		Mode	Address	Data	Mode	Address	Data
Memory read	1	Write	X	H'00	Read	RA	Dout
Erase setup/erase	2	Write	X	H'20	Write	X	H'20
Erase-verify	2	Write	EA	H'A0	Read	X	EVD
Auto-erase setup/ auto-erase	2	Write	X	H'30	Write	X	H'30
Program setup/ program	2	Write	X	H'40	Write	PA	PD
Program-verify	2	Write	X	H'C0	Read	X	PVD
Reset	2	Write	X	H'FF	Write	X	H'FF

PA: Program address

EA: Erase-verify address

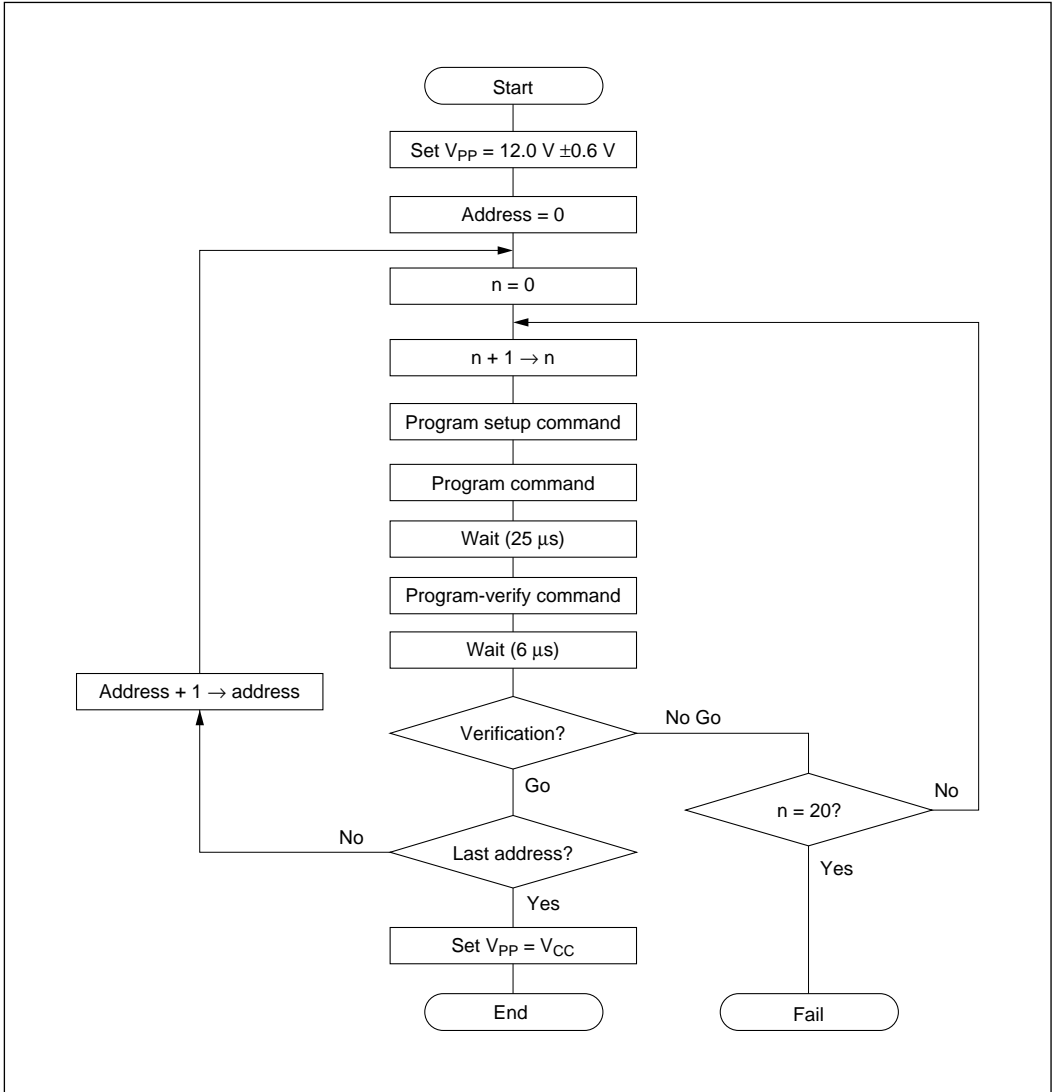
RA: Read address

PD: Program data

PVD: Program-verify output data

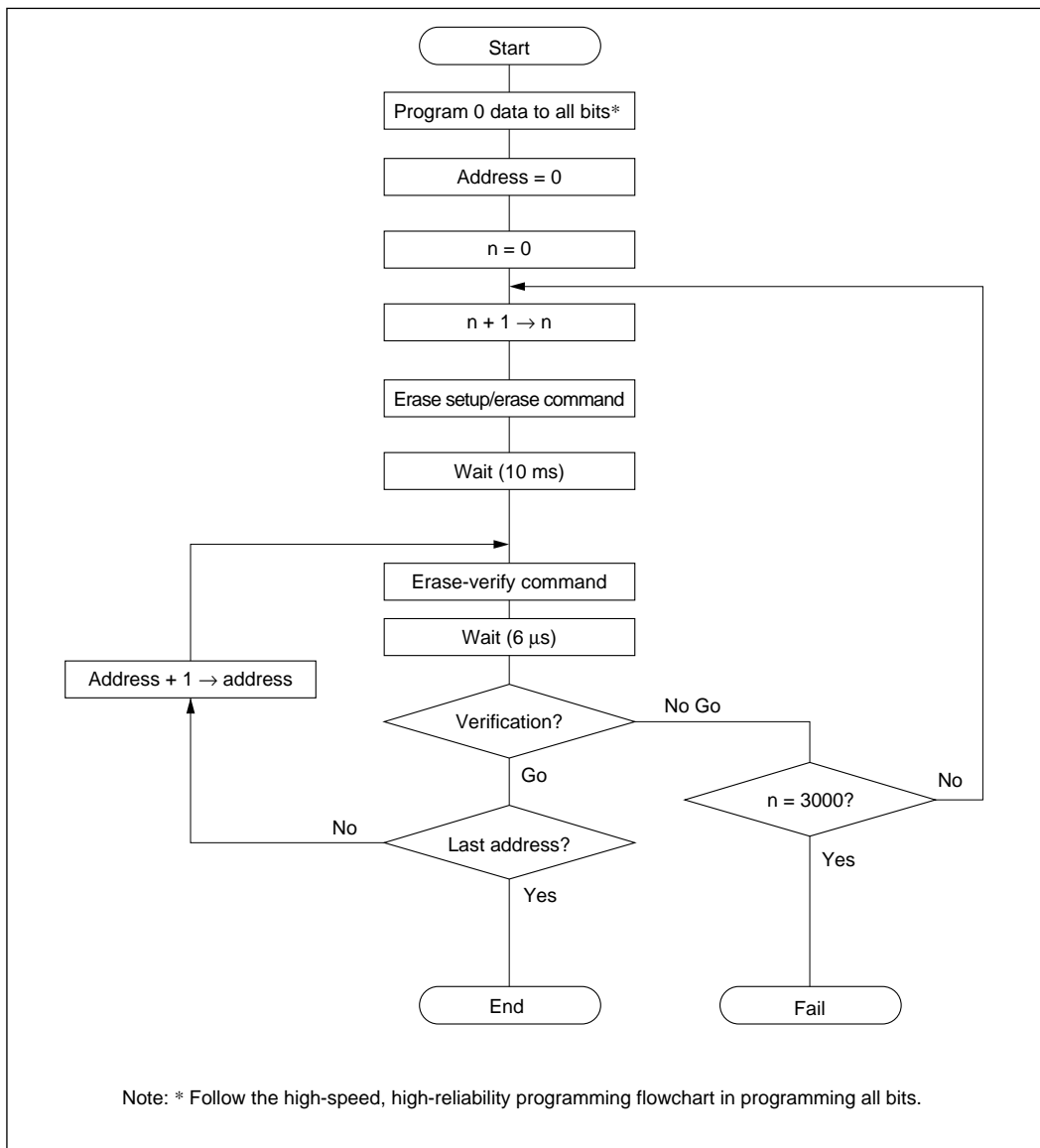
EVD: Erase-verify output data

**High-Speed, High-Reliability Programming:** Unused areas of the H8/539F flash memory contain H'FF data (initial value). The H8/539F flash memory uses a high-speed, high-reliability programming procedure. This procedure provides enhanced programming speed without subjecting the device to voltage stress and without sacrificing the reliability of programmed data. Figure 19-19 shows the basic high-speed, high-reliability programming flowchart. Tables 19-17 and 19-18 list the electrical characteristics during programming.



**Figure 19-18 High-Speed, High-Reliability Programming**

**High-Speed, High-Reliability Erasing:** The H8/539F flash memory uses a high-speed, high-reliability erasing procedure. This procedure provides enhanced erasing speed without subjecting the device to voltage stress and without sacrificing data reliability. Figure 19-20 shows the basic high-speed, high-reliability erasing flowchart. Tables 19-17 and 19-18 list the electrical characteristics during programming.



**Figure 19-19 High-Speed, High-Reliability Erasing**

**Table 19-17 DC Characteristics in PROM Mode**(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{PP} = 12.0\text{ V} \pm 0.6\text{ V}$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item		Symbol	Min	Typ	Max	Unit	Test Conditions
Input high voltage	$O_7$ to $O_0$ , $A_{16}$ to $A_0$ , $\overline{OE}$ , $\overline{CE}$ , $\overline{WE}$	$V_{IH}$	2.2	—	$V_{CC} + 0.3$	V	
Input low voltage	$O_7$ to $O_0$ , $A_{16}$ to $A_0$ , $\overline{OE}$ , $\overline{CE}$ , $\overline{WE}$	$V_{IL}$	-0.3	—	0.8	V	
Output high voltage	$O_7$ to $O_0$	$V_{OH}$	2.4	—	—	V	$I_{OH} = -200\ \mu\text{A}$
Output low voltage	$O_7$ to $O_0$	$V_{OL}$	—	—	0.45	V	$I_{OL} = 1.6\text{ mA}$
Input leakage current	$O_7$ to $O_0$ , $A_{16}$ to $A_0$ , $\overline{OE}$ , $\overline{CE}$ , $\overline{WE}$	$ I_{LI} $	—	—	2	$\mu\text{A}$	$V_{in} = 0$ to $V_{CC}$ V
$V_{CC}$ current	Read	$I_{CC}$	—	40	80	mA	
	Program	$I_{CC}$	—	40	80	mA	
	Erase	$I_{CC}$	—	40	80	mA	
$V_{PP}$ current	Read	$I_{PP}$	—	—	10	$\mu\text{A}$	$V_{PP} = 5.0\text{ V}$
			—	10	20	mA	$V_{PP} = 12.6\text{ V}$
	Program	$I_{PP}$	—	20	40	mA	
	Erase	$I_{PP}$	—	20	40	mA	

Caution: For the absolute maximum ratings, see section 22.1, “Absolute Maximum Ratings.”

Permanent damage to the chip may result if absolute maximum ratings are exceeded.

Set  $V_{PP}$  below 13V taking into account the peak overshoot.

**Table 19-18 AC Characteristics in PROM Mode**(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{PP} = 12.0\text{ V} \pm 0.6\text{ V}$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

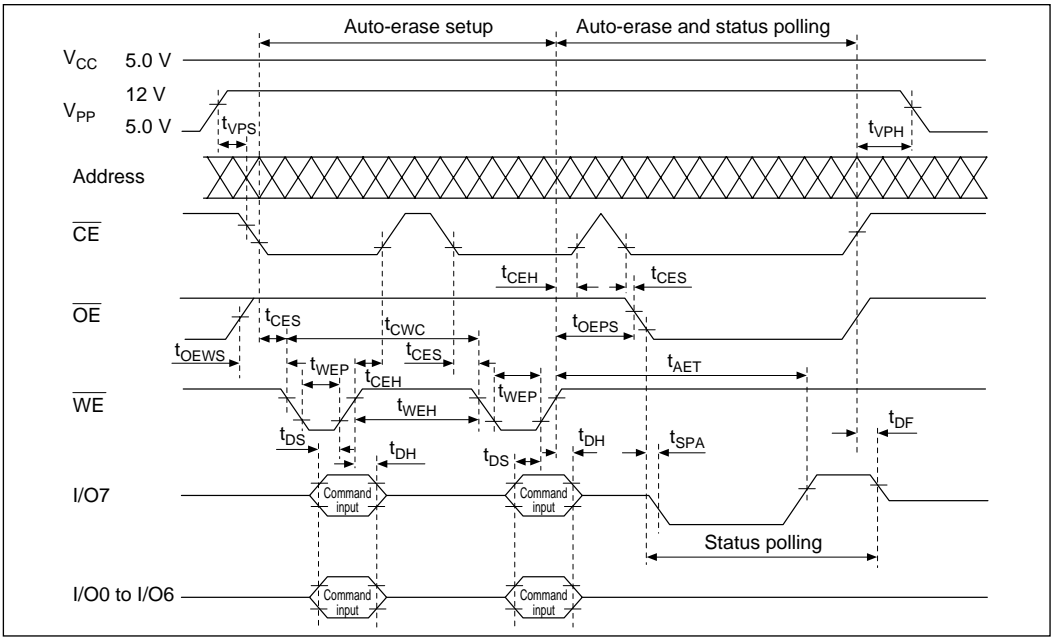
Item	Symbol	Min	Typ	Max	Unit	Test Conditions
Command write cycle	$t_{CWC}$	120	—	—	ns	Figure 19-18
Address setup time	$t_{AS}$	0	—	—	ns	Figure 19-19 *
Address hold time	$t_{AH}$	60	—	—	ns	Figure 19-20
Data setup time	$t_{DS}$	50	—	—	ns	
Data hold time	$t_{DH}$	10	—	—	ns	
$\overline{CE}$ setup time	$t_{CES}$	0	—	—	ns	
$\overline{CE}$ hold time	$t_{CEH}$	0	—	—	ns	
$V_{PP}$ setup time	$t_{VPS}$	100	—	—	ns	
$V_{PP}$ hold time	$t_{VPH}$	100	—	—	ns	
$\overline{WE}$ programming pulse width	$t_{WEP}$	70	—	—	ns	
$\overline{WE}$ programming pulse high time	$t_{WEH}$	40	—	—	ns	
$\overline{OE}$ setup time before command write	$t_{OEWS}$	0	—	—	ns	
$\overline{OE}$ setup time before verify	$t_{OERS}$	6	—	—	$\mu\text{s}$	
Verify access time	$t_{VA}$	—	—	500	ns	
$\overline{OE}$ setup time before status polling	$t_{OEPS}$	120	—	—	ns	
Status polling access time	$t_{SPA}$	—	—	120	ns	
Program wait time	$t_{PPW}$	25	—	—	ns	
Erase wait time	$t_{ET}$	9	—	11	ms	
Output disable time	$t_{DF}$	0	—	40	ns	
Total auto-erase time	$t_{AET}$	0.5	—	30	s	

Note:  $\overline{CE}$ ,  $\overline{OE}$ , and  $\overline{WE}$  should be high during transitions of  $V_{PP}$  from 5 V to 12 V and from 12 V to 5 V.

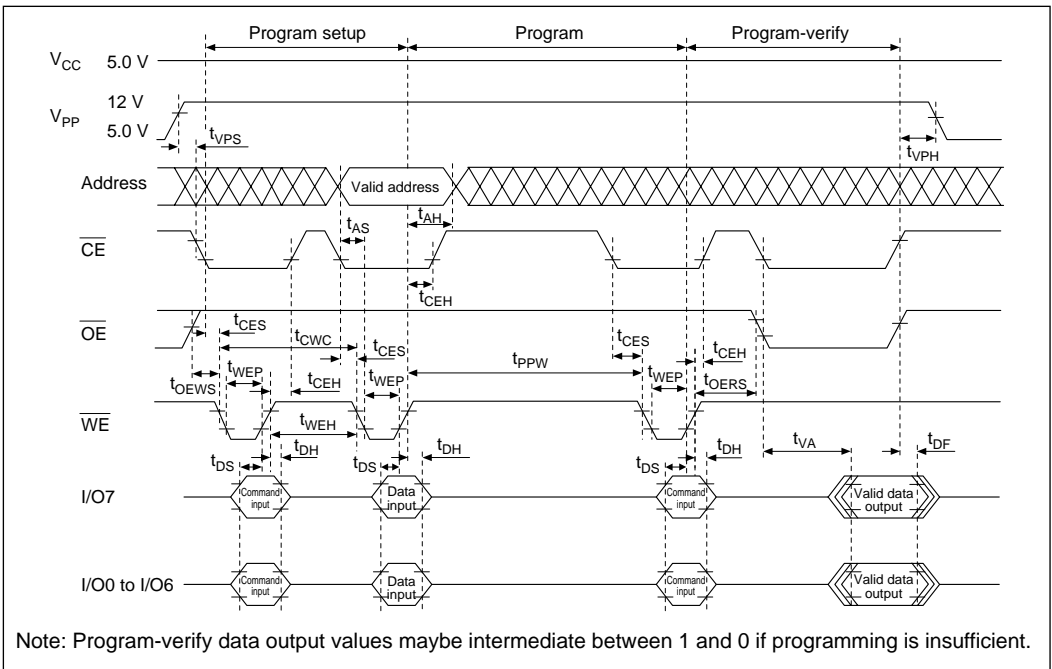
\* Input pulse level: 0.45 V to 2.4 V

Input rise time and fall time  $\leq 10\text{ ns}$

Timing reference levels: 0.8 V and 2.0 V for input; 0.8 V and 2.0 V for output

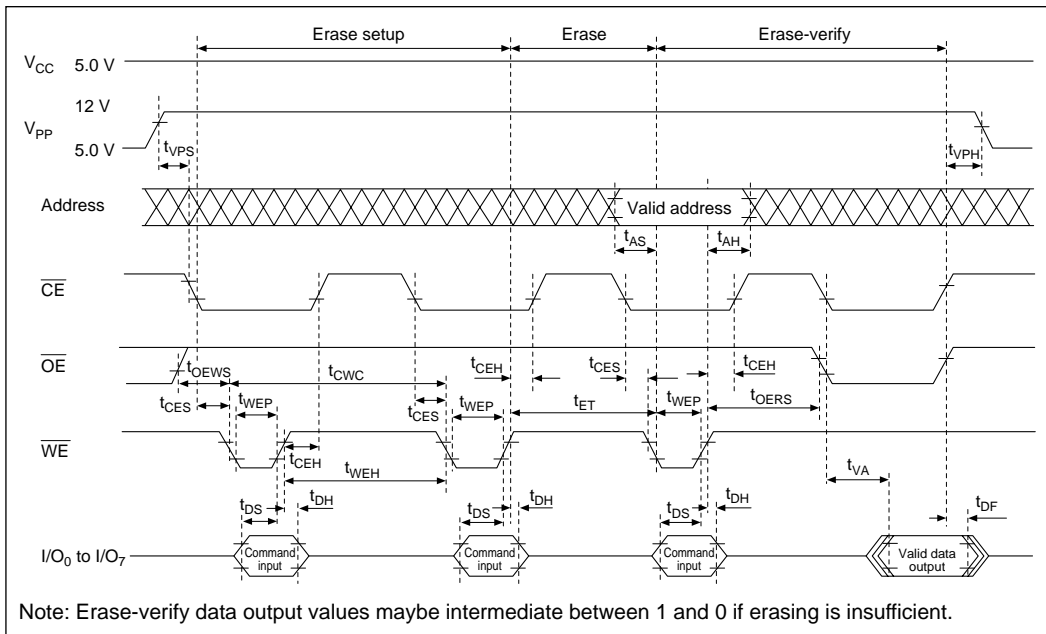


**Figure 19-20 Auto-Erase Timing**



Note: Program-verify data output values may be intermediate between 1 and 0 if programming is insufficient.

**Figure 19-21 High-Speed, High-Reliability Programming Timing**



**Figure 19-22 Erase Timing**



## 19.7 Flash Memory Programming and Erasing Precautions

The following points must be noted when using on-board programming mode, the emulation function with RAM, and PROM mode.

**(1) Program with the specified voltages and timing. The rated programming voltage ( $V_{PP}$ ) of the flash memory is 12.0 V.**

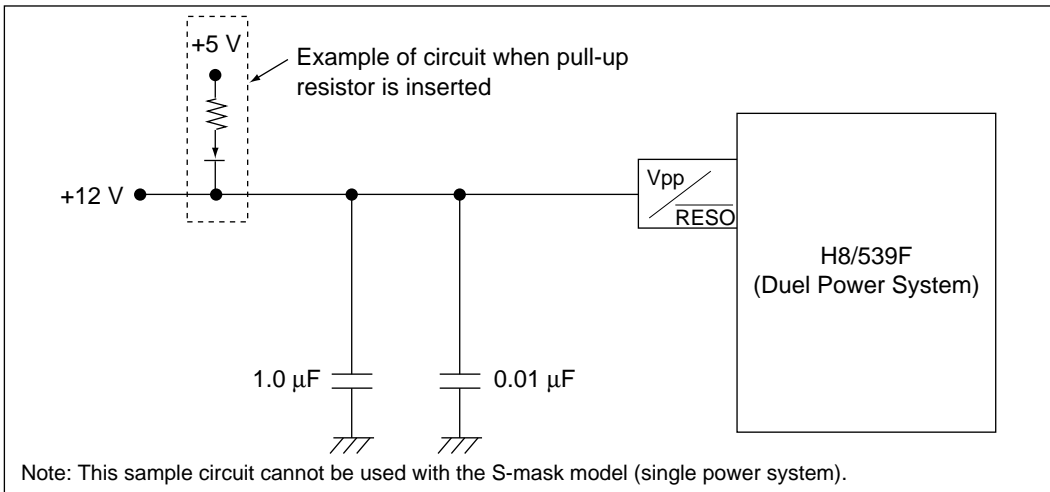
Applied voltages in excess of the rating can permanently damage the device. In particular, ensure that the peak overshoot does not exceed the maximum rating of 13 V at the  $V_{PP}$  and MD2 pins.

In PROM mode,  $V_{PP}$  can be set to 12.0 V if HN28F101 is selected as the PROM programmer setting. 12 V must not be applied to the S-mask model (single power source model), as this will permanently damage the device.

For information concerning the use of the S-mask model, see section 20, Flash Memory (H8/539F, S-Mask Model, Single Power Source).

**(2) Design a current margin into the programming voltage ( $V_{PP}$ ) power supply. Bypass capacitors must be connected to the  $V_{PP}$  pin. (See figures 19-24 and 19-28.)** Ensure that  $V_{PP}$  remains within the range  $12.0 \pm 0.6$  V (11.4 V to 12.6 V) during programming and erasing. Programming and erasing may become impossible outside this range. Connect decoupling capacitors as close to the  $V_{PP}$  pin as possible. When boot mode is used, also, decoupling capacitors should be connected to the MD2 pin in the same way.

For details, see section 19.8, “Notes on Mounting Board Development—Handling of  $V_{PP}$  and Mode MD2 Pins.”



**Figure 19-23 Example of  $V_{PP}$  Power Supply Circuit Design**

### **(3) Precautions in applying and releasing the programming voltage ( $V_{PP}$ ) (See figures 19-22 to 19-24)**

**(a) Apply the programming voltage ( $V_{PP}$ ) after  $V_{CC}$  has stabilized, and shut off  $V_{PP}$  before  $V_{CC}$ .** To avoid programming or erasing flash memory by mistake,  $V_{PP}$  should only be applied and released when the microcontroller is in a “stable operating condition” as described below.

#### **Microcontroller stable operating condition**

- The  $V_{CC}$  voltage must be within the rated voltage range ( $V_{CC} = 4.5\text{ V to }5.5\text{ V}$ ).  
If the  $V_{CC}$  voltage is turned on or off while  $V_{CC}$  is not within its rated voltage range ( $V_{CC} = 4.5\text{ V to }5.5\text{ V}$ ), since the microcontroller is unstable, the flash memory may be programmed or erased by mistake. This can occur even if  $V_{CC} = 0\text{ V}$ . Adequate power supply measures should be taken, such as the insertion of a decoupling capacitor, to prevent fluctuation of the  $V_{CC}$  power supply when  $V_{PP}$  is applied.
- Oscillation must have stabilized (following the elapse of the oscillation settling time).  
When the  $V_{CC}$  power is turned on, hold the  $\overline{RES}$  pin low for the duration of the oscillation settling time ( $t_{OSC1} = 20\text{ ms}$ ) before applying  $V_{PP}$ . Do not apply or release  $V_{PP}$  when oscillation has been stopped or is unstable.
- In boot mode,  $V_{PP}$  should be applied and released during a reset.  
In boot mode, release a reset after the  $V_{PP}$  and MD2 voltages have been stabilized at the programming voltage level ( $12.0\text{ V} \pm 0.6\text{ V}$ ).  
For a reset during operation, apply or release  $V_{PP}$  only after the  $\overline{RES}$  pin has been held low for at least six system clock cycles ( $6\phi$ ).
- The  $V_{PP}E$ , P, E, PV, and EV bits must be cleared in the flash memory control register (FLMCR).  
When applying or releasing  $V_{PP}$ , make sure that the  $V_{PP}E$ , P, E, PV, or EV bit is not set by mistake.
- There must be no program runaway.

When  $V_{PP}$  is applied, program execution must be supervised, e.g. by the watchdog timer.

These power-on and power-off timing requirements for  $V_{CC}$  and  $V_{PP}$  should also be satisfied in the event of a power failure and in recovery from a power failure. If these requirements are not satisfied, overprogramming or overerasing may occur due to program runaway, etc., which could cause memory cells to malfunction.

**(b) The  $V_{PP}$  flag is set and cleared by a threshold decision on the voltage applied to the  $V_{PP}$  pin. The threshold level is approximately in the range from  $V_{CC} + 2\text{ V}$  to  $11.4\text{ V}$ .**

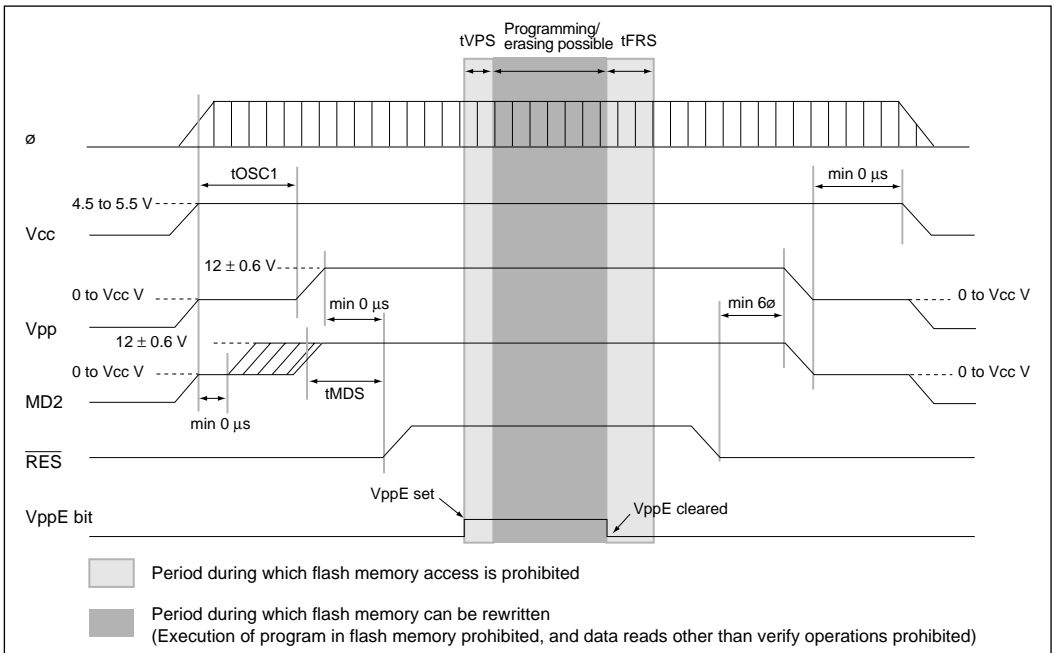
When this flag is set, it becomes possible to set the  $V_{PP}$  enable bit ( $V_{PP}E$ ) in the flash memory control register (FLMCR), even though the  $V_{PP}$  voltage may not yet have settled in the

programming voltage range of  $12.0\text{ V} \pm 0.6\text{ V}$  after  $V_{PP}$  power is turned on. Do not actually program or erase the flash memory until  $V_{PP}$  has reached the programming voltage range. The programming voltage range for programming and erasing flash memory is  $12.0\text{ V} \pm 0.6\text{ V}$  ( $11.4\text{ V}$  to  $12.6\text{ V}$ ). Programming and erasing cannot be performed correctly outside this range. When not programming or erasing the flash memory, ensure that the  $V_{PP}$  voltage does not exceed the  $V_{CC}$  voltage. This will prevent unintentional programming and erasing.

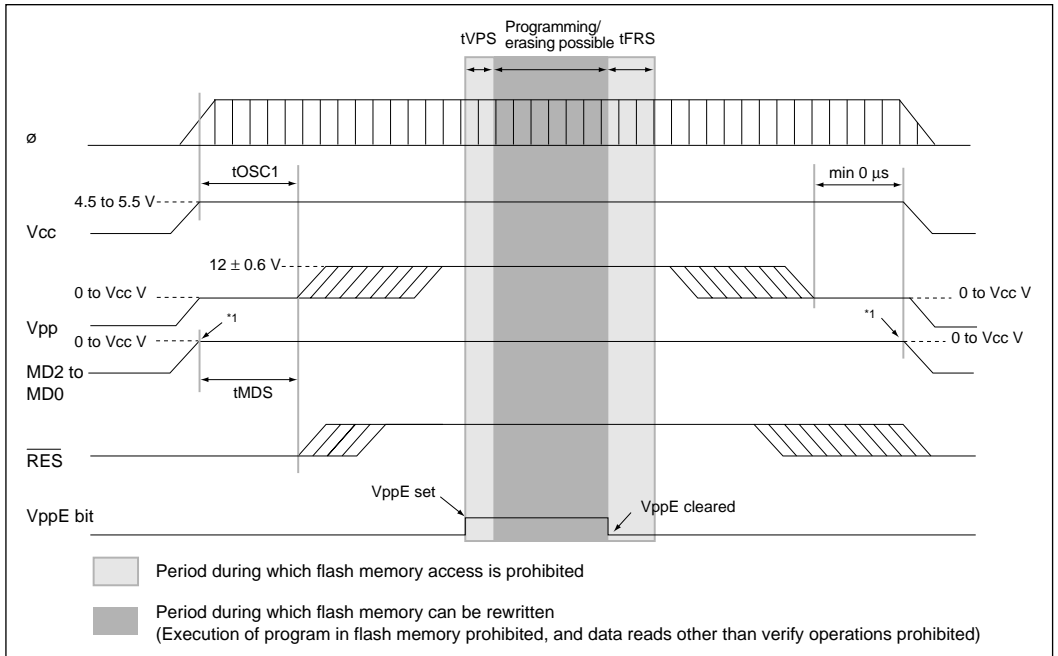
**(c) After the programming voltage ( $V_{PP}$ ) is shut off, make sure that the flash memory read setup time ( $t_{FRS}$ )\* has elapsed before reading the flash memory.** When switching from boot mode or user program mode to normal mode ( $V_{PP} \neq 12\text{ V}$ ,  $MD \neq 12\text{ V}$ ), this setup time is required as the interval before reading flash memory after clearing the  $V_{PP}E$  bit. When switching from boot mode to another mode, the mode programming setup time ( $t_{MDS}$ ) is required with respect to the  $\overline{RES}$  release timing.

Note: \* The flash memory read setup time stipulates the period, from clearing of the  $V_{PP}E$  bit until flash memory is read (figure 22-21). Also, when using an external clock (EXTAL input), after powering on and when returning from standby mode, the flash memory read setup time must be allowed to elapse before flash memory is read (figure 22-22).

**(d) Set the  $V_{PP}$  enable bit ( $V_{PP}E$ ) before programming or erasing.** Setting the  $V_{PP}E$  bit makes it possible to write to the flash memory control register (FLMCR) and the erase block registers (EBR1 and EBR2).

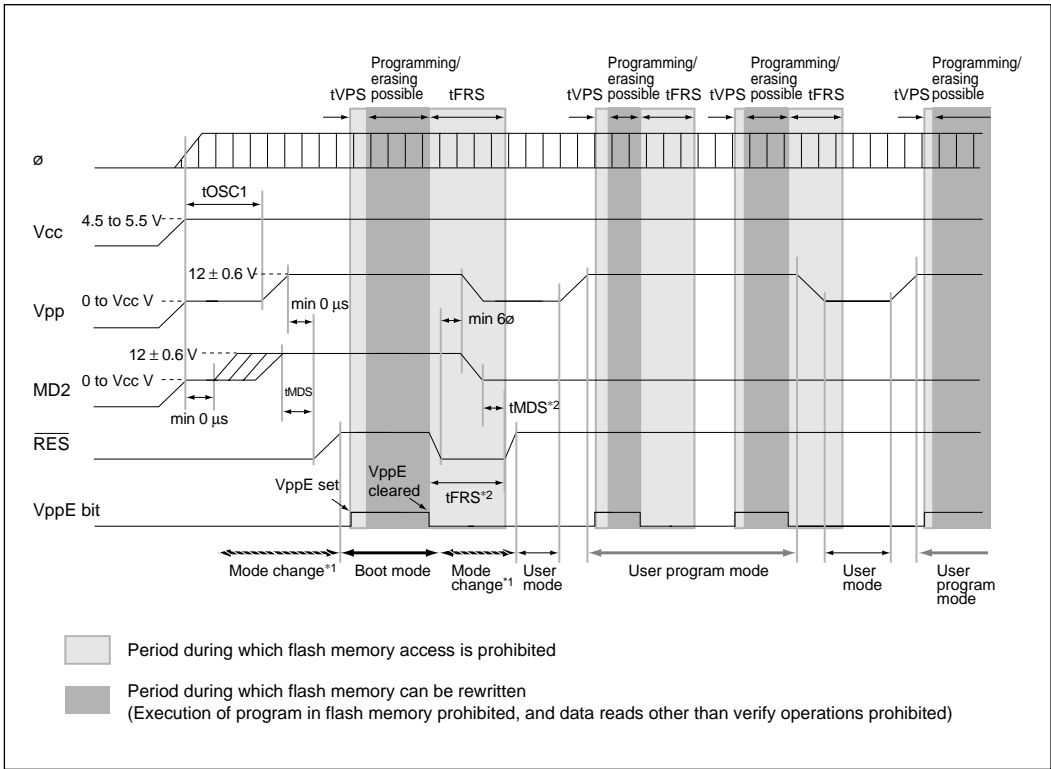


**Figure 19-24 Power-On and Power-Off Timing (Boot Mode)**



Notes: 1. The level of the mode pins (MD2 to MD0) must be fixed from power-on to power-off by pulling the pins up or down.

**Figure 19-25 Power-On and Power-Off Timing (User Program Mode)**



- Notes:
- When entering boot mode or making a transition from boot mode to another mode, mode switching must be carried out by means of  $\overline{RES}$  input. The states of ports multiplexed as address pins and bus control output signals ( $\overline{AS}$ ,  $\overline{RD}$ ,  $\overline{HWR}$ ,  $\overline{LWR}$ ) change during this switchover interval (the interval during which the  $\overline{RES}$  pin input is low), and therefore these pins should not be used as output signals during this time.
  - When making a transition from boot mode to another mode, the flash memory read setup time  $t_{FRS}$  and mode programming setup  $t_{MDS}$  must be satisfied with respect to  $\overline{RES}$  clearance timing.

**Figure 19-26 Mode Transition Timing**  
**(Example: Boot Mode  $\Rightarrow$  User Mode  $\Leftrightarrow$  User Program Mode)**

**(4) Do not apply 12 V to the  $V_{pp}$  pin during normal operation.** To prevent program runaway caused by accidental programming or erasing, apply 12 V to  $V_{pp}$  only when the flash memory is programmed or erased, or when flash memory is emulated by RAM. Overprogramming or overerasing due to program runaway, etc., may prevent memory cells from operating normally. While 12 V is applied, the watchdog timer should be running and enabled to halt runaway program execution, so that program runaway will not lead to overprogramming or overerasing.

**(5) Disable watchdog timer reset output ( $\overline{\text{RESO}}$ ) while the programming voltage ( $V_{\text{PP}}$ ) is applied.** If 12 V is applied during watchdog timer reset output (while the  $\overline{\text{RESO}}$  pin is low), overcurrent flow will permanently destroy the reset output circuit. When 12 V is applied to the  $V_{\text{PP}}/\overline{\text{RESO}}$  pin, the reset output enable bit (RESOE) in the watchdog timer reset control/status register (RSTCSR) should not be set to 1.

If a Vcc pull-up resistor is connected to the  $V_{\text{PP}}/\overline{\text{RESO}}$  pin externally, a diode must be inserted to prevent reverse current to the Vcc side when  $V_{\text{PP}}$  is applied (figure 19-21).

**(6) When the  $V_{\text{PP}}/\overline{\text{RESO}}$  pin is used as the watchdog timer reset output (when 12 V is not applied), the rise and fall of the reset output waveform will be delayed by any decoupling capacitors connected to the  $V_{\text{PP}}/\overline{\text{RESO}}$  pin, the Vcc pull-up resistor, etc.**

**(7) Follow the recommended algorithm for programming and erasing flash memory.** This algorithm enables programming and erasing to be performed without subjecting the device to voltage stress or sacrificing programmed data reliability.

When setting the program (P) bit or erase (E) bit in the flash memory control register (FLMCR), the watchdog timer should be set beforehand to prevent program runaway.

**(8) Do not set or clear the  $V_{\text{PP}}\text{E}$  bit while a program is running in flash memory.** Flash memory data cannot be read normally when the  $V_{\text{PP}}\text{E}$  bit is set or cleared. Also, although flash memory data can be rewritten after waiting for the elapse of the  $V_{\text{PP}}$  enable setup time ( $t_{\text{VPS}}$ ), flash memory can only be accessed for the purpose of verification (verification during the program, erase, or prewrite flow). Flash memory program execution and data reading should only be performed after the elapse of the flash memory setup time after the  $V_{\text{PP}}\text{E}$  bit is cleared.

In the same way, when using the function for emulation by RAM with 12 V applied to the  $V_{\text{PP}}$  pin, the flash memory read setup time must be provided after clearing the  $V_{\text{PP}}\text{E}$  bit when executing a program or reading data in flash memory. However, read and write accesses can be carried out in flash memory space and the overlapped RAM area regardless of whether the  $V_{\text{PP}}\text{E}$  bit is set or cleared.

**(9) Do not use interrupts while programming or erasing flash memory.** When  $V_{\text{PP}}$  is applied, disable all interrupt requests, including NMI, to give the programming or erase operation (including emulation by RAM) the highest priority.

**(10) Before programming, check that the chip is correctly mounted in the PROM programmer.** Overcurrent damage to the device can result if the index marks on the PROM programmer socket, socket adapter, and chip are not correctly aligned.

**(11) Do not touch the socket adapter or chip while programming.** Touching either of these can cause contact faults and write errors.

## 19.8 Notes on Mounting Board Development—Handling of $V_{PP}$ and Mode MD2 Pins

**(1) The standard 12 V high voltage is applied to the  $V_{PP}$  and mode MD2 pins when erasing or programming flash memory.** The voltage at these pins also includes overshoot and noise, and the following points should be noted to ensure that the 13 V maximum rated voltage is not exceeded. 12 V must not be applied to the S-mask model (single power source), as this may cause permanent damage to the chip.

For information concerning the use of the S-mask model, see section 20, Flash Memory (H8/539F, S-Mask Model, Single Power Source).

**(a) Bypass capacitors should be inserted to eliminate overshoot and noise.** These should be positioned as close as possible to the chip's  $V_{PP}$  and mode MD2 pins.

1.0  $\mu$ F: Stabilizes fluctuations in the power supply low-frequency components, such as power supply ripple.

0.01  $\mu$ F: Bypasses high-frequency components such as induction noise.

**(b) The  $V_{PP}$  and mode MD2 pin wiring should be kept as short as possible to suppress induction noise.** When designing a new board, in particular, noise may be increased by jumper wires, etc. In this case too, the power supply waveform should be monitored and measures taken to prevent the maximum rating from being exceeded.

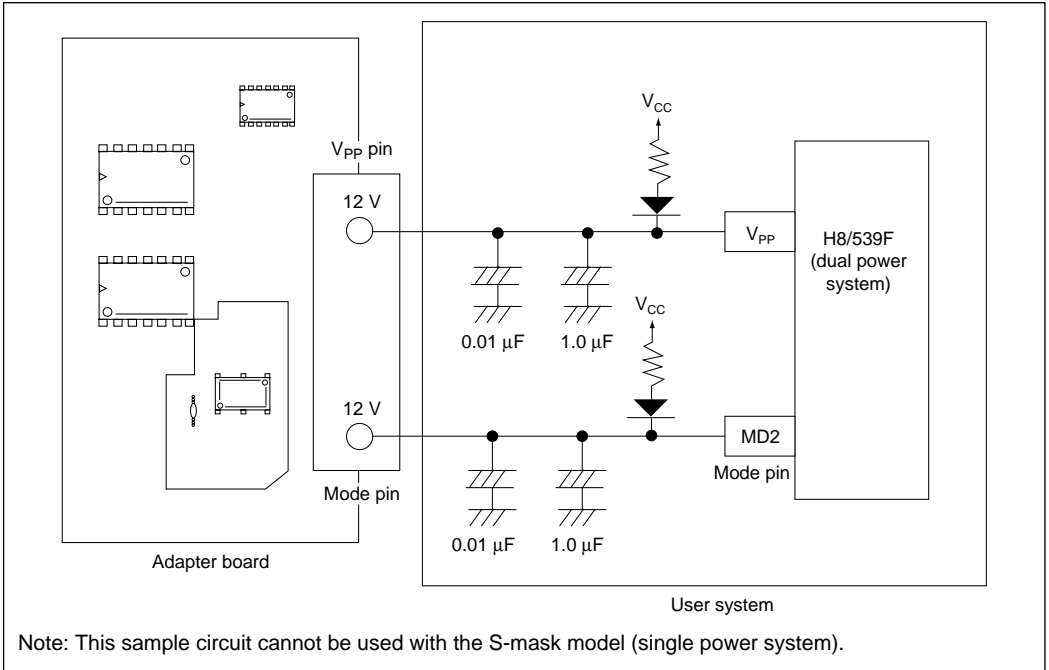
**(c) The maximum rated voltage is based on the potential of the  $V_{SS}$  pin.** If the potential of this pin oscillates due to current fluctuations, etc., the voltage of the  $V_{PP}$  and mode MD2 pins may exceed the maximum rated voltage. Careful attention must therefore be paid to stabilizing the reference potential.

Note: When the user system's 12 V power supply is connected, attention must be paid to the current capacity. A power supply with a small current capacity will not be able to handle fluctuations in the chip's operating voltage, resulting in voltage drops and rises or oscillation that may make it impossible to obtain the rated operating voltage. Caution is required if the power supply has a large current capacity, or if the 12 V voltage is turned on abruptly by means of a switch, etc., since a voltage exceeding the maximum rating may be generated due to the inductance component of the power supply wiring or the power supply characteristics.

Before using the power supply, check the power supply waveform to ensure that the above problems will not arise.

**(2) 12 V is applied to the  $V_{PP}$  and mode MD2 pins when erasing or programming flash memory.** When these pins are pulled up to the  $V_{CC}$  line in normal operation, diodes should be inserted to prevent reverse current from flowing to the  $V_{CC}$  line when 12 V is applied.

Note: If the mode MD2 pin to which 12 V is applied is to be set to 0 in normal operation, it should be pulled down with a resistor.



**Figure 19-27 Example of Mounting Board Design for H8/539F (dual power system)  
(Connection to Adapter Board—When V<sub>PP</sub> Pin and Mode Pin Settings are 1)**



# Section 20 Flash Memory (H8/539F, S-Mask Model: Single Power Source)

## 20.1 Overview

### 20.1.1 Notes on S-Mask Model (Single Power Source)

There are two models of the H8/539F with on-chip flash memory: a dual power source model and a single power source (S-mask) model. Points to be noted when using the H8/539F single power source S-mask model are given below.

For the differences between the dual power source model and single power source (S-mask) model, see section 1.4.3, Differences in S-Mask Model.

#### (1) Voltage application

12 V must not be applied to the S-mask model (single power source), as this will permanently damage the device.

The flash memory programming power source for the S-mask model (single power source) is  $V_{CC}$ .

The programming power source for the dual power source model was the  $V_{PP}$  pin (12 V), but there is no  $V_{PP}$  pin in the single power source model. In the S-mask model the FWE pin is provided at the same pin position as the  $V_{PP}$  pin in the dual power source model, but FWE is not a power source pin—it is used to control flash memory write enabling.

Also, in boot mode, 12 V must be applied to the MD2 pin in the dual power source model, but this is not necessary in the S-mask model (single power source).

The maximum rating of the FWE and MD2 pins in the S-mask model (single power source) is  $V_{CC} + 0.3$  V. Applying a voltage in excess of the maximum rating will permanently damage the device.

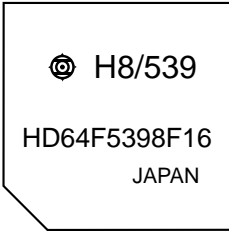
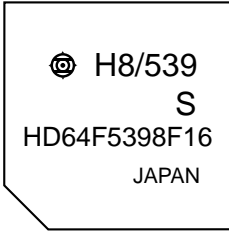
Do not select the HN28F101 programmer setting for the S-mask model (single power source). If this setting is made by mistake, 12.0 V will be applied to the FWE pin, permanently damaging the device.

When using a PROM programmer to program the on-chip flash memory in the S-mask model (single power source), use a PROM programmer that supports Hitachi microcomputer device types with 128-kbyte on-chip flash memory.

## (2) Product type names and markings

Table 20-1 shows examples of product type names and markings for the H8/539F (dual power source model) and H8/539F S-mask model (single power source), and the differences in flash memory programming power source.

**Table 20-1 Differences between H8/539F and H8/539F S-Mask Model Markings**

	Dual Power Source Model: H8/539F	Single Power Source Model: H8/539F S-Mask Model (Preliminary)
Product type name	HD64F5398F16	HD64F5398SF16
Sample markings		 <p>“S” is printed above the type name</p>
Flash memory programming power source	$V_{PP}$ power source (12.0 $\pm$ 0.6 V)	$V_{CC}$ power source (5.0 $\pm$ 10%)

### 20.1.2 Mode Pin Settings and ROM Space

The H8/539F has 128 kbytes of on-chip flash memory. The ROM is connected to the CPU by a 16-bit data bus. The CPU accesses flash memory in two states for both byte-size and word-size instructions.

The flash memory is allocated to addresses H'00000—H'03FFF and H'10000—H'2FFFF in the memory map. The contents of flash memory addresses H'00000—H'03FFF are the same as those of addresses H'10000—H'13FFF. This space can be switched between on-chip flash memory space and external memory space with a mode pin setting. Mode pin settings and corresponding flash memory space designations are shown in table 20-1.

**Table 20-1 Mode Pin Settings and ROM Space**

Mode	Mode Pin Setting			ROM Space Designation
	MD <sub>2</sub>	MD <sub>1</sub>	MD <sub>0</sub>	
Mode 0	0	0	0	Setting prohibited
Mode 1	0	0	1	External memory space
Mode 2*	0	1	0	On-chip flash memory space
Mode 3	0	1	1	External memory space
Mode 4	1	0	0	On-chip flash memory space
Mode 5	1	0	1	External memory space
Mode 6	1	1	0	External memory space
Mode 7	1	1	1	On-chip flash memory space

Note: \* Do not perform flash memory programming or erasing in mode 2. When mode 2 is set, the FWE pin must be driven low.

### 20.1.3 Features

The features of the flash memory are summarized below.

**Four flash memory operating states** There are four flash memory operating states: program mode, program-verify mode, erase mode, and erase-verify mode.

**Erase block specification** The flash memory space block to be erased can be specified by setting the corresponding register bit. The flash memory is divided into three 32-kbyte blocks, one 28-kbyte block, and four 1-kbyte blocks.

**Programming/erase times** The flash memory programming time is 10 ms (typ.) for simultaneous 32-byte programming, equivalent to 300  $\mu$ s (typ.) per byte, and the erase time for one block is 100 ms (typ.).

**Reprogramming capability** The flash memory can be reprogrammed up to 100 times.

**On-board programming modes** There are two modes in which flash memory can be programmed, erased, and verified on-board: boot mode and user program mode.

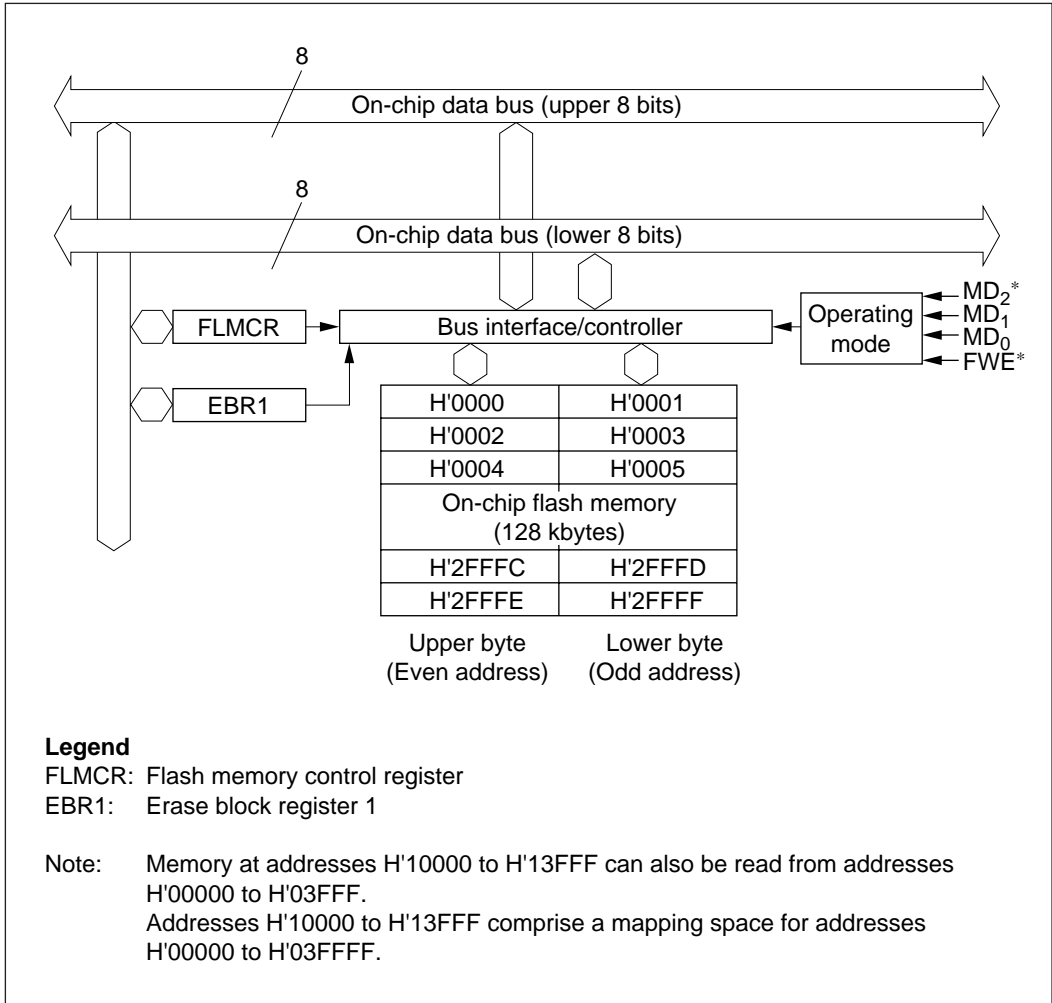
**Automatic bit rate adjustment** For data transfer in boot mode, the H8/539F's bit rate can be automatically adjusted to match the transfer bit rate of the host (9600, 4800, or 2400 bps).

**Flash memory emulation in RAM** Flash memory programming can be emulated in real time by overlapping a part of RAM onto flash memory.

**PROM mode** Flash memory can be programmed and erased in PROM mode, using a PROM programmer, as well as in on-board programming mode.

## 20.1.4 Block Diagram

Figure 20-1 shows a block diagram of the flash memory.



**Figure 20-1 Block Diagram of Flash Memory**

**Note:** \* 12 V must on no account be applied to the S-mask model (single power source), as this will permanently damage the chip.

### 20.1.5 Pin Configuration

The flash memory is controlled by means of the pins shown in table 20-2.

**Table 20-2 Flash Memory Pins**

Pin Name	Abbreviation	I/O	Function
Flash write enable	FWE*	Input	Sets program mode
Mode 2	MD <sub>2</sub> *	Input	Sets H8/539F operating mode
Mode 1	MD <sub>1</sub>	Input	Sets H8/539F operating mode
Mode 0	MD <sub>0</sub>	Input	Sets H8/539F operating mode
Transmit data	TXD <sub>1</sub>	Output	Serial transmit data output
Receive data	RXD <sub>1</sub>	Input	Serial receive data input

The transfer data pin and receive data pin are used in boot mode.

Note: \* 12 V must on no account be applied to the S-mask model (single power source), as this will permanently damage the chip.

### 20.1.6 Register Configuration

The registers used to control the on-chip flash memory are shown in table 20-3.

**Table 20-3 Flash Memory Registers\*<sup>2</sup>**

Register Name	Abbreviation	R/W	Initial Value	Address
Flash memory control register	FLMCR	R/W	H'00 or H'08	H'FEE0
Erase block register 1	EBR1	R/W	H'00	H'FEE2
RAM control register* <sup>2</sup>	RAMCR	R/W	Undefined	H'FF15
Flash memory emulation register	FLMER	R/W	H'73	H'FEEC
Flash memory status register	FLMSR	R	H'7F	H'FEED

Note: 1. The function of the RAM control register is to enable or disable access to the on-chip RAM, but in this section it is used for RAM area setting in on-board programming mode.

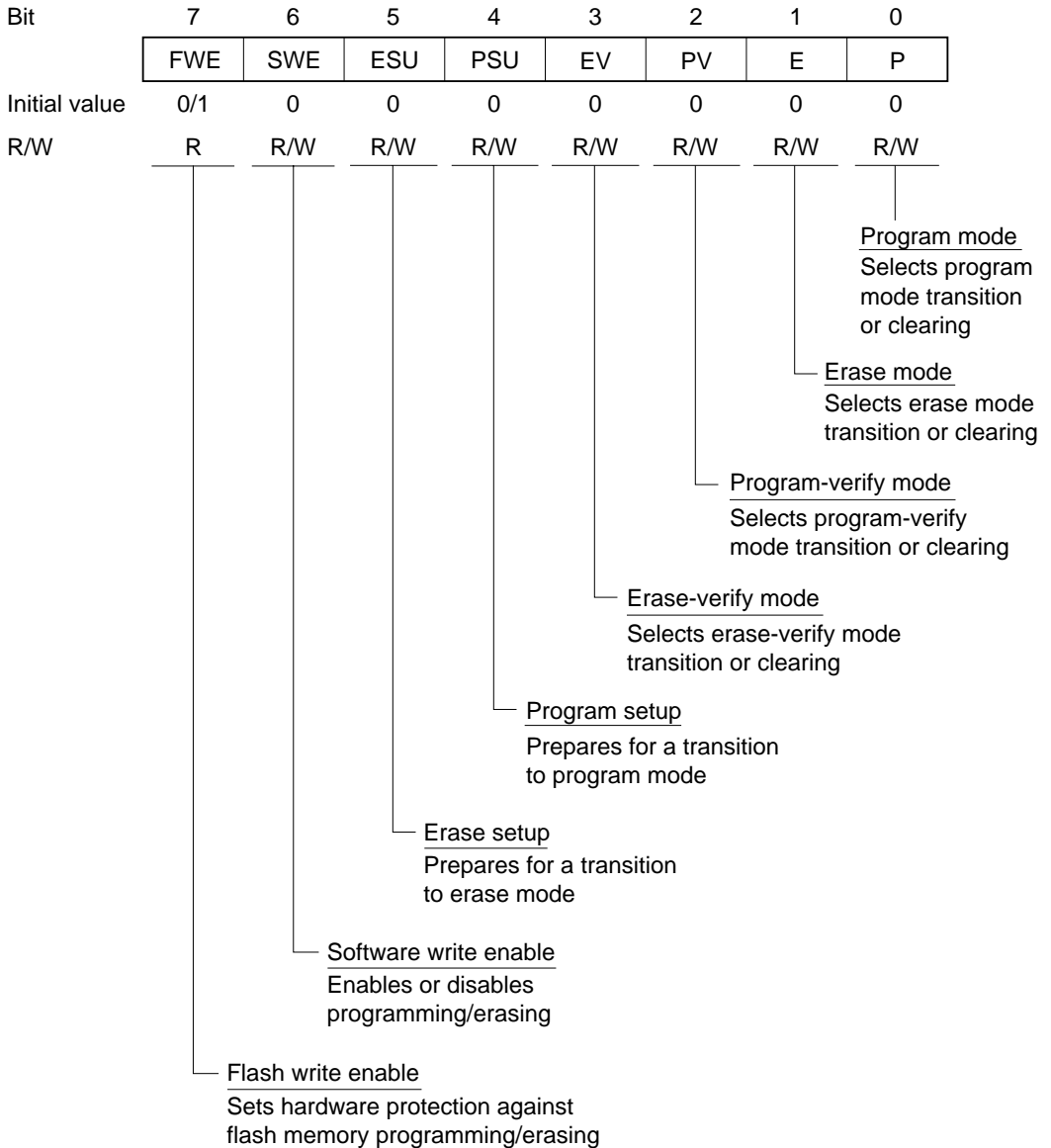
2. These registers and bits (bits 2 to 0 of RAMCR) are used only for flash memory control, and are not present in mask and ZTAT versions. Registers and bits exclusively for flash memory use should not be accessed in mask and ZTAT versions; if they are, a read will return H'FF (or 1 in the case of RAMCR bits 2 to 0), and writes will be invalid.

## 20.2 Register Descriptions

### 20.2.1 Flash Memory Control Register (FLMCR)

FLMCR is an 8-bit register used for flash memory operating mode control. Program-verify mode or erase-verify mode is entered by setting SWE to 1 when FWE = 1, then setting the corresponding bit. Program mode is entered by setting SWE to 1 when FWE = 1, then setting the PSU bit, and finally setting the P bit. Erase mode is entered by setting SWE to 1 when FWE = 1, then setting the ESU bit, and finally setting the E bit. FLMCR is initialized by a reset and in standby mode. Its initial value is H'80 when a high level is input to the FWE pin, and H'00 when a low level is input. When on-chip flash memory is disabled, a read will return H'00, and writes are invalid.

Writes to bits ESU, PSU, EV, and PV in FLMCR are enabled only when FWE = 1 and SWE = 1; writes to the E bit only when FWE = 1, SWE = 1, and ESU = 1; and writes to the P bit only when FWE = 1, SWE = 1, and PSU = 1.



(1) **Bit 7—Flash Write Enable (FWE):** Sets hardware protection against flash memory programming/erasing. See section 20.7, Flash Memory Programming and Erasing Precautions, for more information on the use of this bit.

#### Bit 7

FWE	Description
0	When a low level is applied to the FWE pin (hardware-protected state)
1	When a high level is applied to the FWE pin

(2) **Bit 6—Software Write Enable (SWE)\*<sup>1</sup>, \*<sup>2</sup>:** Enables or disables flash memory programming and erasing. This bit should be set before setting FLMCR bits 5 to 0 and EBR bits 7 to 0. (Do not set the ESU, PSU, EV, PV, E, or P bit at the same time.)

#### Bit 6

SWE	Description
0	Programming/erasing disabled (Initial value)
1	Programming/erasing enabled [Setting condition] When FWE = 1

(3) **Bit 5—Erase Setup (ESU)\*<sup>1</sup>:** Prepares for a transition to erase mode. (Do not set the SWE, PSU, EV, PV, E, or P bit at the same time.)

#### Bit 5

ESU	Description
0	Erase setup cleared (Initial value)
1	Erase setup [Setting condition] When FWE = 1 and SWE = 1

(4) **Bit 4—Program Setup (PSU)\*<sup>1</sup>:** Prepares for a transition to program mode. (Do not set the SWE, ESU, EV, PV, E, or P bit at the same time.)

#### Bit 4

PSU	Description
0	Program setup cleared (Initial value)
1	Program setup [Setting condition] When FWE = 1 and SWE = 1



(5) **Bit 3—Erase-Verify Mode (EV)\*1:** Selects erase-verify mode transition or clearing. (Do not set the SWE, ESU, PSU, PV, E, or P bit at the same time.)

**Bit 3**

EV	Description
0	Erase-verify mode cleared (Initial value)
1	Transition to erase-verify mode [Setting condition] When FWE = 1 and SWE = 1

(6) **Bit 2—Program-Verify Mode (PV)\*1:** Selects program-verify mode transition or clearing. (Do not set the SWE, ESU, PSU, EV, E, or P bit at the same time.)

**Bit 2**

PV	Description
0	Program-verify mode cleared (Initial value)
1	Transition to program-verify mode [Setting condition] When FWE = 1 and SWE = 1

(7) **Bit 1—Erase Mode (E)\*1, \*3:** Selects erase mode transition or clearing. (Do not set the SWE, ESU, PSU, EV, PV, or P bit at the same time.)

**Bit 1**

E	Description
0	Erase mode cleared (Initial value)
1	Transition to erase mode [Setting condition] When FWE = 1, SWE = 1, and ESU = 1

(8) **Bit 0—Program Mode (P)\*1, \*3:** Selects program mode transition or clearing. (Do not set the SWE, ESU, PSU, EV, PV, or E bit at the same time.)

**Bit 0**

P	Description
0	Program mode cleared (Initial value)
1	Transition to program mode [Setting condition] When FWE = 1, SWE = 1, and PSU = 1

- Notes:
1. Do not set multiple bits simultaneously. Do not cut  $V_{CC}$  when a bit is set.
  2. The SWE bit must not be set or cleared at the same time as other bits (ESU, PSU, EV, PV, E, or P).
  3. P bit and E bit setting should be carried out in accordance with the program/erase algorithm shown in section 20.4, Flash Memory Programming/Erasing. Before setting either of these bits, a watchdog timer setting should be made to prevent program runaway. See section 20.7, Flash Memory Programming and Erasing Precautions, for more information on the use of these bits.

## 20.2.2 Erase Block Register 1 (EBR1)

EBR1 is an 8-bit register that specifies the flash memory block to be erased. EBR1 is initialized to H'00 by a reset, in standby mode, when a high level is not being input to the FWE pin, and when a high level is input to the FWE pin while the SWE bit in FLMCR is cleared to 0. When a bit in EBR1 is set to 1, the corresponding block can be erased. Other blocks are erase-protected. Only one bit in EBR1 can be set at a time (do not set 1 simultaneously for multiple erase blocks). An EBR1 bit cannot be set to 1 until the SWE bit is set to 1 in FLMCR. The erase block configuration is shown in table 20-2. Erasing is performed one block at a time. To erase the entire flash memory, individual blocks must be erased in succession.

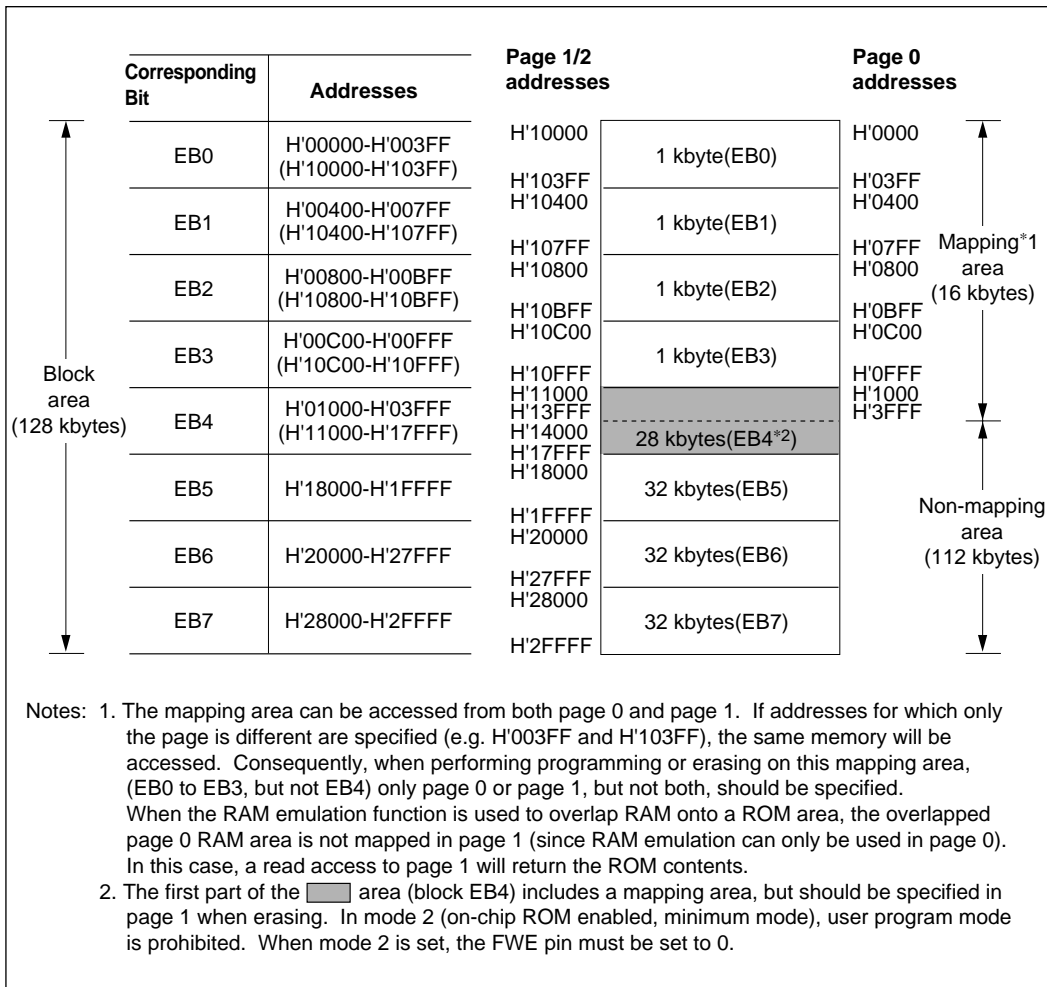
Bit	7	6	5	4	3	2	1	0
	EB7	EB6	EB5	EB4	EB3	EB2	EB1	EB0
Initial value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**(1) Bits 7 to 0—Block 7 to Block 0 (EB7 to EB0):** Setting one of these bits specifies the corresponding block (EB7 to EB0) for erasure.

### Bit 7 to 0

<b>EB7 to EB0</b>	<b>Description</b>
-------------------	--------------------

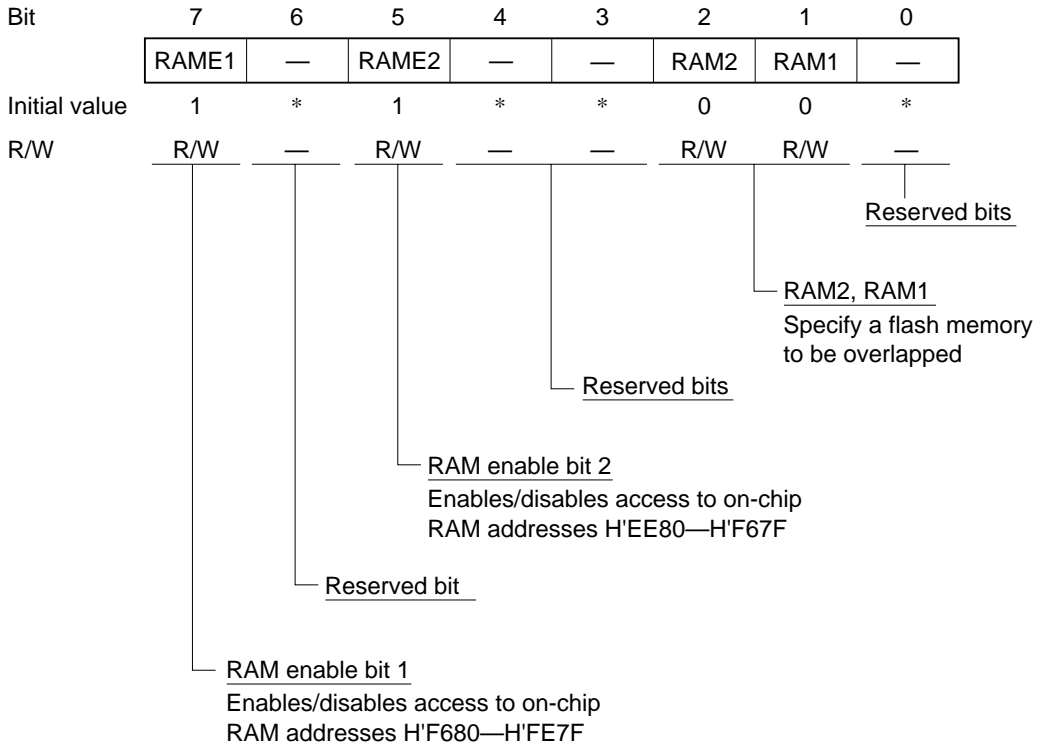
0	Corresponding block (EB7 to EB0) not selected (Initial value)
1	Corresponding block (EB7 to EB0) selected



**Figure 20-2 Flash Memory Erase Blocks**

### 20.2.3 RAM Control Register (RAMCR)

RAMCR controls enabling and disabling of access to on-chip RAM, and RAM area overlapping.



Note: \* The value of bits 6, 4, 3, and 0 is undefined when RAMCR is read.

(1) **Bits 7 and 5: RAM Enable 1, RAM Enable 2 (RAME1, RAME2):** When bits 7 and 5 are cleared to 0, access to on-chip RAM is disabled. For details, see section 18.2, RAM Control Register (RAMCR).

(2) **Bits 6, 4, 3, 0—Reserved:** These bits cannot be modified and will return an undefined value if read.

(3) **Bits 2 and 1—RAM2, RAM1:** Used together with bits 7, 3, and 2 in the flash memory emulation register (FLMER) to select the ROM area to be overlapped (see table 20-4.) In modes 2, 4, and 7 (on-chip flash memory enabled), these bits have an initial value of 0 and can be modified<sup>\*1</sup>. In other modes they cannot be modified and are always read as 0. These bits are initialized by a reset and in hardware standby mode. They are not initialized in software standby mode.

Notes: 1. Bits 2 to 1 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7. (In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{PP}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{PP}$  voltage.)  
 The H8/539F S-mask model uses a single 5 V power source, and does not have a  $V_{PP}$  pin. As with the regular H8/539F, the H8/539F S-mask model can perform RAM emulation in modes 2, 4, and 7 (on-chip flash memory enabled). Flash memory programming must not be performed in mode 2.

### 20.2.4 Flash Memory Emulation Register (FLMER)

FLMER performs enabling and disabling of flash memory RAM emulation and RAM area modification when RAM emulation is started.

Bit	7	6	5	4	3	2	1	0
	OVLPE	—	—	—	A11E	A10E	—	—
Initial value	0	1	1	1	0	0	1	1
R/W	R/W	—	—	—	R/W	R/W	—	—

Reserved bits  
 A10E bit  
 Bits A11E and A10E specify a RAM area to be overlapped onto flash memory  
 A11E bit  
 Bits A11E and A10E specify a RAM area to be overlapped onto flash memory  
Reserved bit  
 Emulation RAM enable (overlap RAM enable)  
 Enables or disables overlapping of a part of RAM onto a flash memory small block area

**(1) Bit 7—Emulation RAM Enable (OVLPE):** Used with bits 3 and 2 to specify a RAM area (see table 20-4). When bit 7 is set, all flash memory blocks are protected from programming and erasing, regardless of the values of bits 3 and 2. This state is referred to as emulation protection<sup>\*1</sup>. In this state the flash memory will not enter program mode or erase mode even if the P or E bit is set in the flash memory control register (FLMCR). Only transitions to verify modes are possible. Bit 7 must be cleared to 0 to enable flash memory to be actually programmed or erased.

In modes 2, 4, and 7 (with on-chip flash memory enabled), the initial value of this bit is 0, but it can be modified by writing 1\*2. In other modes this bit cannot be modified and always reads 0. It is initialized by a reset and in hardware standby mode. It is not initialized in software standby mode. Flash memory programming must not performed in mode 2.

**(2) Bits 3 and 2—A11E, A10E:** These bits specify a RAM area to be overlapped onto ROM when performing flash memory emulation in RAM (see table 20-5).

In modes 2, 4, and 7 (with on-chip flash memory enabled), the initial value of these bits is 0, but they can be modified. In other modes these bits cannot be modified and always read 0. They are initialized by a reset and in hardware standby mode. They are not initialized in software standby mode. Flash memory programming must not performed in mode 2.

- Notes: 1. For details of emulation protection, see section 20.4.8, Protection Modes.  
 2. Bits 7, 3, and 2 of the flash memory emulation register (FLMER) and bits 2 and 1 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7.  
 (In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{PP}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{PP}$  voltage. The H8/539F S-mask model uses a single 5 V power source, and does not have a  $V_{PP}$  pin. As with the regular H8/539F, the H8/539F S-mask model can perform RAM emulation in modes 2, 4, and 7 (on-chip flash memory enabled).)  
 Flash memory programming must not performed in mode 2.

### 20.2.5 Flash Memory Status Register (FLMSR)

FLMSR is used to detect flash memory errors.

Bit	7	6	5	4	3	2	1	0
	FLER	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
R/W	R	—	—	—	—	—	—	—

Flash memory error (FLER)  
 Status flag indicating detection of an error during flash memory programming or erasing

**(1) Bit 7—Flash Memory Error (FLER):** Indicates that an error has occurred during flash memory programming or erasing. When FLER is set to 1, flash memory enters error protection mode.

**Bits 7**

<b>FLER</b>	<b>Description</b>
0	Flash memory program/erase protection (error protection*1) is disabled (Initial value) [Clearing condition] $\overline{RES}$ pin reset*3 or hardware standby mode
1	An error has occurred during flash memory programming/erasing, and error protection*1 has been enabled [Setting conditions] <ul style="list-style-type: none"> <li>•When flash memory is read*2 during programming/erasing (including a vector read or instruction fetch, but excluding a read in a RAM area overlapped onto flash memory space)</li> <li>•Immediately after the start of exception handling during programming/erasing (excluding reset, illegal instruction, trap instruction, and division-by-zero exception handling)*4</li> <li>•When a SLEEP instruction (including software standby) is executed during programming/erasing</li> <li>•When the bus is released during programming/erasing*5</li> </ul>

- Notes:
1. For details of error protection, see section 20.4.8, Protection Modes.
  2. An undefined value will be read in this case.
  3. In the H8/538F, a watchdog timer reset is included in the FLER bit clearing conditions, but only  $\overline{RES}$  pin reset input is applicable in the case of the H8/539F.
  4. Before a stack or vector read is performed in exception handling.
  5. Applies to the H8/539F S-mask model. The bus release condition does not apply to the H8/538F or H8/539F.

Bits 6 to 0 are reserved. They cannot be modified and are always read as 1.

**Table 20-4 ROM Area Setting**

ROM Area (Mapping RAM Area)	FLMER Register	RAMCR Register		Overlap Function	Program /Erase Protection
	Bit 7*1	Bit 2*1	Bit 1*1		
	OVLPE	RAM2	RAM1		
—	0	0/1	0/1	Disabled	Disabled
H'0000-H'03FF	1	0	0	Enabled	Enabled
H'0400-H'07FF	1	0	1	Enabled	Enabled
H'0800-H'0BFF	1	1	0	Enabled	Enabled
H'0C00-H'0FFF	1	1	1	Enabled	Enabled

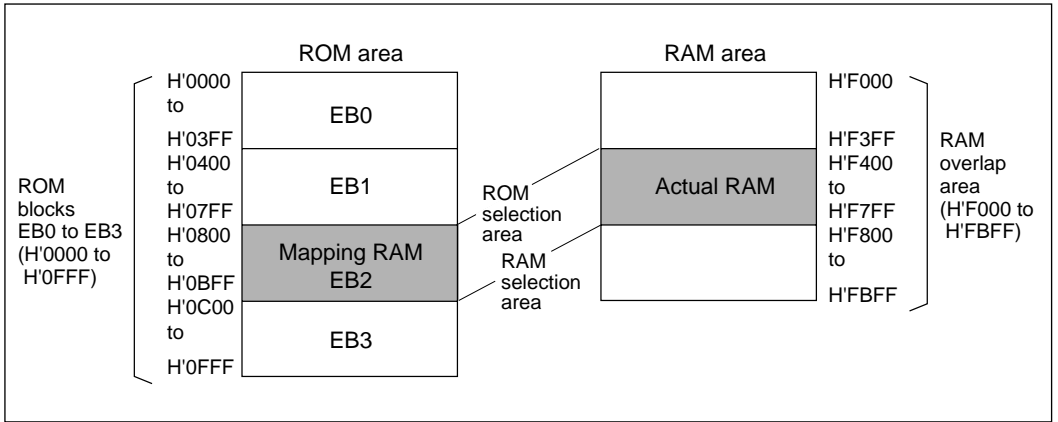
**Table 20-5 RAM Area\*2 Setting**

RAM Area*2 (Mapping RAM Area)	FLMER Register		RAMCR Register	
	Bit 3*1	Bit 2*1	Bit 7*1	Bit 5*1
	A11E	A10E	RAME1	RAME2
H'F000-H'F3FF (1024 bytes)	0	0	1	1
H'F400-H'F7FF (1024 bytes)	0	1	1	1
H'F800-H'FBFF (1024 bytes)	1	0	1	1
Use prohibited *3	1	1	1	1
Use prohibited *4	*	*	0	0
Use prohibited *4	*	*	0	1
Use prohibited *4	*	*	1	0

- Notes: 1. Bits 7, 3, and 2 of the flash memory emulation register (FLMER) and bits 2 and 1 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7.  
(In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{PP}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{PP}$  voltage. The H8/539F S-mask model uses a single 5 V power source, and does not have a  $V_{PP}$  pin. As with the regular H8/539F, the H8/539F S-mask model can perform RAM emulation in modes 2, 4, and 7 (on-chip flash memory enabled). Flash memory programming must not performed in mode 2.)
- RAM area overlapped onto flash memory.
  - Use prohibited when A11E and A10E are both set to 1. (A10E cannot be set to 1 when A11E is set to 1).
  - Use prohibited when RAME1 = 0 or RAME2 = 0. (Can only be used when RAME1 = RAME2 = 1.)



# Example of ROM Area/RAM Area Overlap



## 20.3 On-Board Programming Mode

When on-board programming mode is set, on-chip flash memory programming, erasing, and verifying can be carried out. There are two operating modes in this mode—boot mode and user program mode—set by the mode pins (MD2 to MD0) and the FWE pin. The pin settings for entering each mode are shown in table 20-6. Boot mode and user program mode cannot be used in H8/539F mode 2 (on-chip ROM enabled). For notes on FWE pin application and disconnection, see section 20.7, Flash Memory Programming and Erasing Precautions.

**Table 20-6 On-Board Programming Mode Settings**

Mode Selection		FWE*2	MD <sub>2</sub> *2	MD <sub>1</sub>	MD <sub>0</sub>	Notes
Boot mode	Mode 4	1 *1	1	0	1 *1	0:V <sub>IL</sub> 1:V <sub>IH</sub>
	Mode 7		1	1	0 *1	
User program mode	Mode 4		1	0	0	
	Mode 7		1	1	1	

Notes: 1. (1) For the high level application timing, see items (6) and (7) in Notes on Use of Boot Mode.

(2) In boot mode, the input is the inverse of the MD<sub>0</sub> setting.

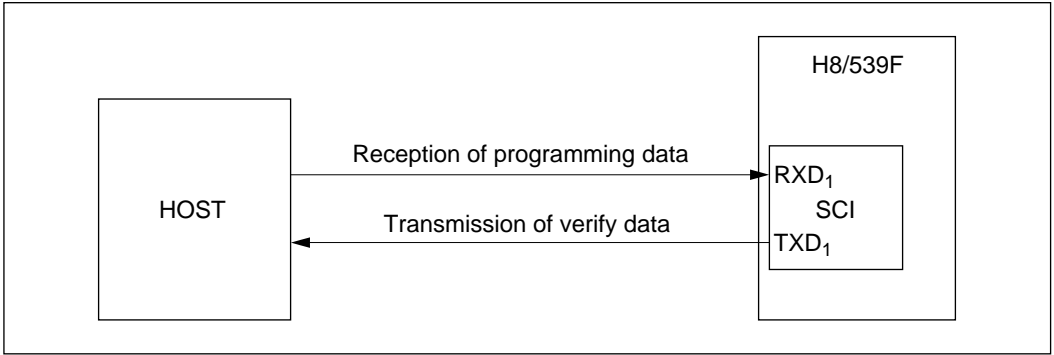
(3) In boot mode, the mode control register (MDCR) can be used to monitor the status of modes 4 and 7 in the same way as for normal mode.

2. 12 V must on no account be applied to the S-mask model (single power source), as this will permanently damage the chip.

### 20.3.1 Boot Mode

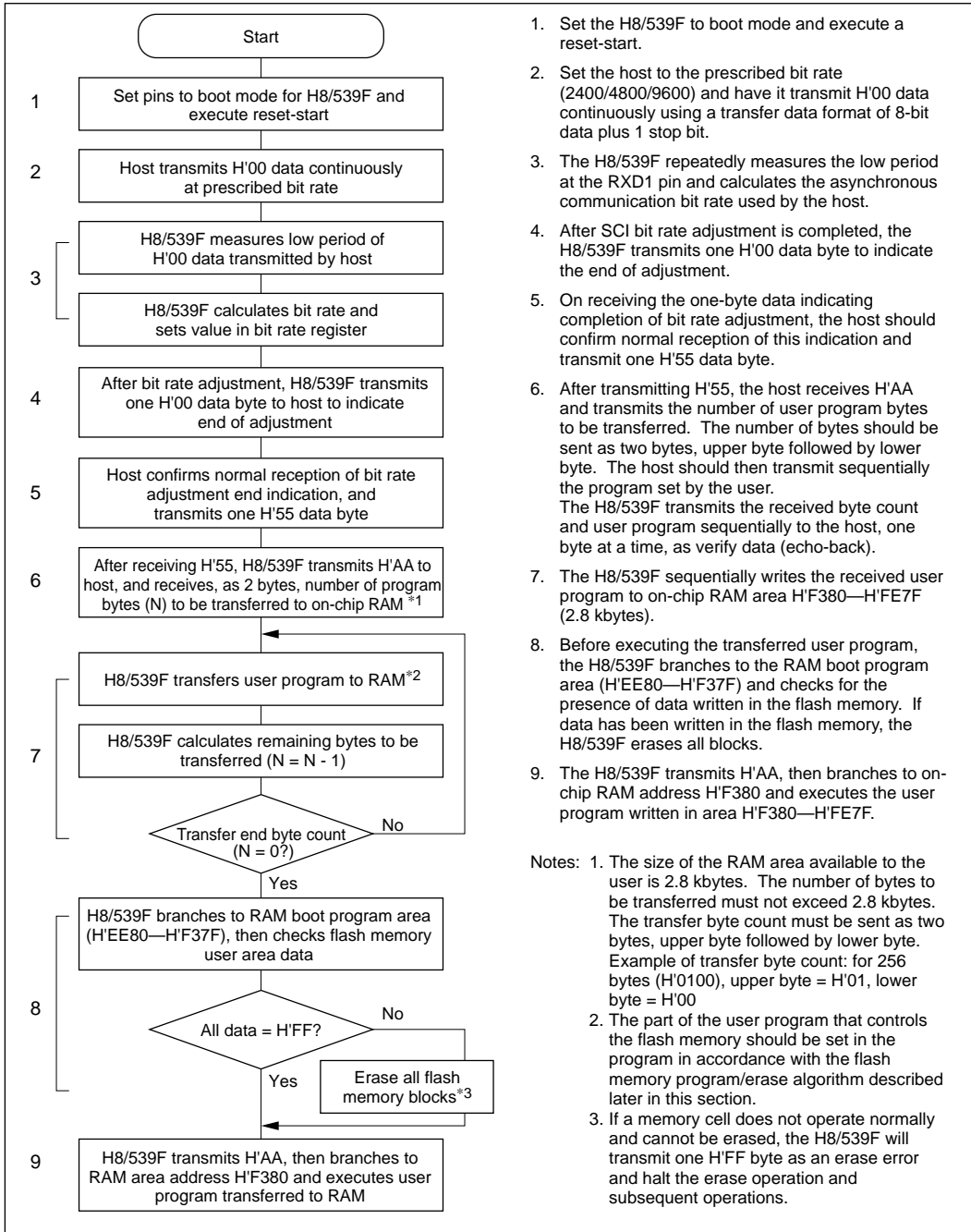
When boot mode is used, a user program for flash memory programming and erasing must be prepared beforehand in the host personal computer, etc., and channel 1 of the SCI to be used must be set to asynchronous mode. When the H8/539F is set to boot mode, after reset release the boot program already incorporated in the H8/539F is activated, the low period of the data sent from the host is first measured, and the bit rate register (BRR) value determined. It is then possible to receive a user program from off-chip using the H8/539F's on-chip serial communication interface (SCI), and the received user program is written into RAM.

Control then branches to the start address (H'F380) of the on-chip RAM, the program written in RAM is executed, and flash memory programming/erasing can be carried out. Figure 20-3 shows a system configuration diagram when using boot mode, and figure 20-4 shows the boot program mode execution procedure.



**Figure 20-3 System Configuration When Using Boot Mode**

The boot mode execution procedure is shown below.

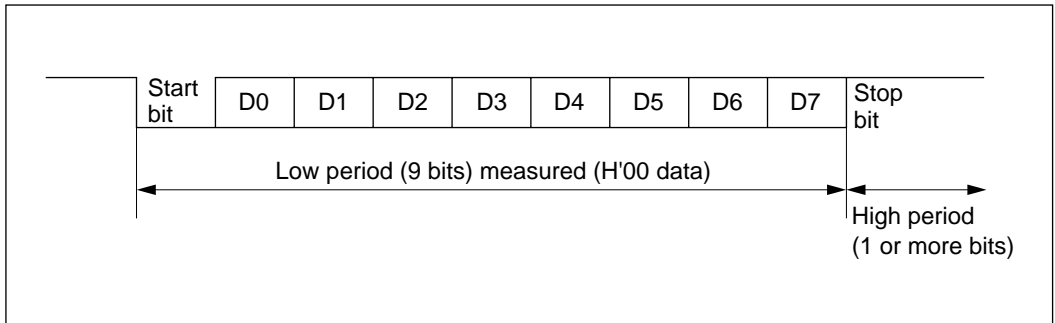


1. Set the H8/539F to boot mode and execute a reset-start.
2. Set the host to the prescribed bit rate (2400/4800/9600) and have it transmit H'00 data continuously using a transfer data format of 8-bit data plus 1 stop bit.
3. The H8/539F repeatedly measures the low period at the RXD1 pin and calculates the asynchronous communication bit rate used by the host.
4. After SCI bit rate adjustment is completed, the H8/539F transmits one H'00 data byte to indicate the end of adjustment.
5. On receiving the one-byte data indicating completion of bit rate adjustment, the host should confirm normal reception of this indication and transmit one H'55 data byte.
6. After transmitting H'55, the host receives H'AA and transmits the number of user program bytes to be transferred. The number of bytes should be sent as two bytes, upper byte followed by lower byte. The host should then transmit sequentially the program set by the user. The H8/539F transmits the received byte count and user program sequentially to the host, one byte at a time, as verify data (echo-back).
7. The H8/539F sequentially writes the received user program to on-chip RAM area H'F380—H'FE7F (2.8 kbytes).
8. Before executing the transferred user program, the H8/539F branches to the RAM boot program area (H'EE80—H'F37F) and checks for the presence of data written in the flash memory. If data has been written in the flash memory, the H8/539F erases all blocks.
9. The H8/539F transmits H'AA, then branches to on-chip RAM address H'F380 and executes the user program written in area H'F380—H'FE7F.

- Notes:
1. The size of the RAM area available to the user is 2.8 kbytes. The number of bytes to be transferred must not exceed 2.8 kbytes. The transfer byte count must be sent as two bytes, upper byte followed by lower byte. Example of transfer byte count: for 256 bytes (H'0100), upper byte = H'01, lower byte = H'00
  2. The part of the user program that controls the flash memory should be set in the program in accordance with the flash memory program/erase algorithm described later in this section.
  3. If a memory cell does not operate normally and cannot be erased, the H8/539F will transmit one H'FF byte as an erase error and halt the erase operation and subsequent operations.

Figure 20-4 Boot Mode Flowchart

## Automatic SCI Bit Rate Adjustment



**Figure 20-5 Measurement of Low Period of Host's Transmit Data**

When boot mode is initiated, the H8/539F measures the low period of the asynchronous SCI communication data transmitted continuously from the host (figure 20-5). The data format should be set as 8-bit data, 1 stop bit, no parity. The H8/539F calculates the bit rate of the transmission from the host from the measured low period (9 bits), and transmits one H'00 byte to the host to indicate the end of bit rate adjustment. The host should confirm that this adjustment end indication has been received normally, and transmit one H'55 byte to the H8/539F. If reception cannot be performed normally, initiate boot mode again (reset), and repeat the above operations. Depending on the host's transmission bit rate and the H8/539F's system clock frequency, there will be a discrepancy between the bit rates of the host and the H8/539F. To ensure correct SCI operation, the host's transfer bit rate should be set to 2400, 4800, or 9600 bps<sup>\*1</sup>. Table 20-7 shows typical host transfer bit rates and system clock frequencies for which automatic adjustment of the H8/539F bit rate is possible. The boot program should be executed within this system clock range<sup>\*2</sup>.

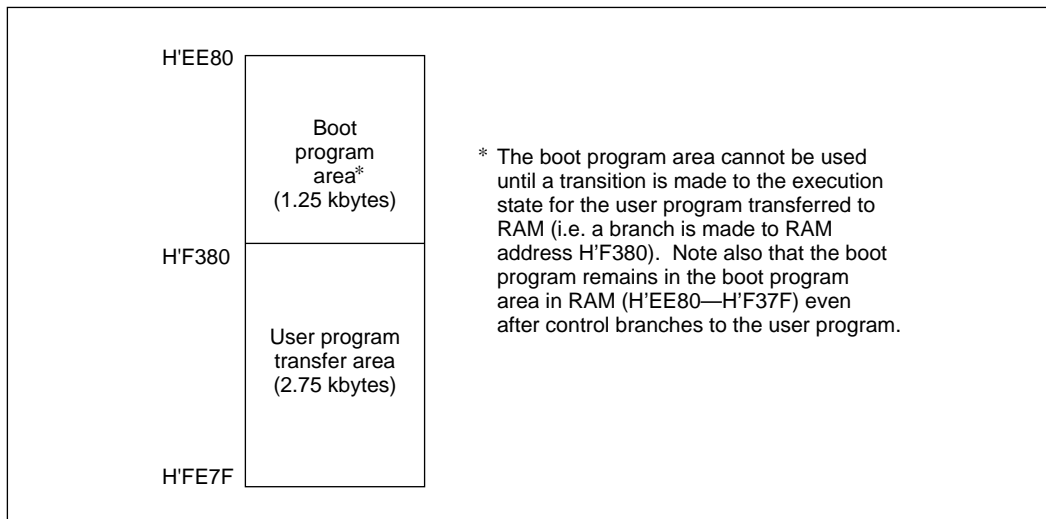
**Table 20-7 System Clock Frequencies for which Automatic Adjustment of H8/539F Bit Rate is Possible**

Host Bit Rate	System Clock Frequency for which Automatic Adjustment of H8/539F Bit Rate is Possible
9600 bps	8 MHz to 16 MHz
4800 bps	4 MHz to 16 MHz
2400 bps	2 MHz to 16 MHz

- Notes: 1. Use a host bit rate setting of 2400, 4800, or 9600 bps only. No other setting should be used.
2. Although the H8/539F may also perform automatic bit rate adjustment with bit rate and system clock combinations other than those shown in table 20-7, a degree of error will arise between the bit rates of the host and the H8/539F, and subsequent transfer will not be performed normally. Therefore, only a combination of bit rate and system clock frequency within one of the ranges shown in table 20-7 can be used for boot mode execution.

## RAM Area Divisions in Boot Mode

In boot mode, the 1280-byte area from H'EE80 to H'F37F is reserved for boot program use, as shown in figure 20-6. The area to which the user program is transferred is H'F380 to H'FE7F (2.75 kbytes). The boot program area becomes available when a transition is made to the execution state for the user program transferred to RAM.



**Figure 20-6 RAM Areas in Boot Mode**

### Notes on Use of Boot Mode:

1. When the H8/539F comes out of reset in boot mode, it measures the low period of the input at the SCI's RXD<sub>1</sub> pin. The reset should end with RXD<sub>1</sub> high. After the reset ends, it takes about 100 states for the chip to get ready to measure the low period of the RXD<sub>1</sub> input.
2. In boot mode, if any data has been programmed into the flash memory (if all data is not H'FF), all flash memory blocks are erased. Boot mode is for use when user program mode is unavailable, such as the first time on-board programming is performed, or if the program activated in user program mode is accidentally erased.
3. Interrupts cannot be used while the flash memory is being programmed or erased.
4. The RXD<sub>1</sub> and TXD<sub>1</sub> lines should be pulled up on the board.
5. Before branching to the user program (RAM address H'F380), the H8/539F terminates transmit and receive operations by the on-chip SCI (channel 1) (by clearing the RE and TE bits to 0 in the serial control register (SCR)), but the adjusted bit rate value remains set in the bit rate register (BRR). The transmit data output pin, TXD<sub>1</sub>, goes to the high-level output state (P7<sub>2</sub>DDR = 1 in the port 7 data direction register, P7<sub>2</sub>DR = 1 in the port 7 data register).

The contents of the CPU's internal general registers are undefined at this time, so these registers must be initialized immediately after branching to the user program. In particular, since the stack pointer (SP) is used implicitly in subroutine calls, etc., a stack area must be specified for use by the user program.

The initial values of other on-chip registers are not changed.

6. Boot mode can be entered by setting pins MD<sub>0</sub> to MD<sub>2</sub> and FWE in accordance with the mode setting conditions shown in table 20.5, and then executing a reset-start.

On reset release (a low-to-high transition)<sup>\*1</sup>, the H8/539F latches the current mode pin states internally and maintains the boot mode state. Boot mode can be cleared by driving the FWE pin low, then executing reset release<sup>\*1</sup>, but the following points must be noted.

- (a) When switching from boot mode to normal mode, the boot mode state within the chip must first be cleared by reset input via the  $\overline{\text{RES}}$  pin. The  $\overline{\text{RES}}$  pin must be held low for at least 20 system clock cycles.<sup>\*3</sup>
  - (b) If input level at the MD<sub>0</sub> and FWE pins is changed in boot mode, the boot mode state will be maintained in the MCU, and boot mode continued, unless  $\overline{\text{RES}}$  pin reset input is performed. Also, if a watchdog timer reset occurs in the boot mode state, the MCU's internal state will not be cleared, and the on-chip boot program will be restarted regardless of the state of the mode pins.
  - (c) The FWE pin must not be driven low while the boot program is running or flash memory is being programmed or erased<sup>\*2</sup>.
7. If the MD<sub>0</sub> and FWE pin input levels are changed from 0 V to 5 V or from 5 V to 0 V during a reset (while a low level is being input at the  $\overline{\text{RES}}$  pin), the MCU's operating mode will change. As a result, the state of ports with multiplexed address functions and bus control output pins ( $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{HWR}}$ ,  $\overline{\text{LWR}}$ ) will also change. Therefore, care must be taken to make pin settings to prevent these pins from becoming output signal pins during a reset, or to prevent collision with signals outside the MCU.

Notes: 1. Mode pin and FWE pin input must satisfy the mode programming setup time (tMDS) with respect to the reset release timing.

2. For further information on FWE application and disconnection, see section 20.7, Flash Memory Programming and Erasing Precautions.

3. See section 4.2.2, Reset Sequence, and section 20.7, Flash Memory Programming and Erasing Precautions. The reset period during operation is a minimum of 6 system clock cycles for the H8/538F and H8/539F, but a minimum of 20 system clock cycles for the H8/539F S-mask model.

### 20.3.2 User Program Mode

When set to user program mode, the H8/539F can program and erase its flash memory by executing a user program. Therefore, on-board reprogramming of the on-chip flash memory can be carried out by providing on-board means of FWE control and supply of programming data, and storing a programming control program in part of the program area as necessary.

To select user program mode, select a mode that enables the on-chip flash memory (mode 4 or 7), and apply a high level to the FWE pin. In this mode, on-chip supporting modules other than flash memory operate as they normally would in modes 4 and 7.

The flash memory itself cannot be read while being programmed or erased, so the control program that performs programming should be placed in external memory or transferred to RAM and executed there. Flash memory programming and erasing should not be carried out in mode 2. When mode 2 is set, the FWE pin must be driven low.

#### User Program Mode Execution Procedure\*1

The execution procedure when user program mode is entered during program execution in RAM is shown below. It is also possible to start from user program mode in a reset-start.

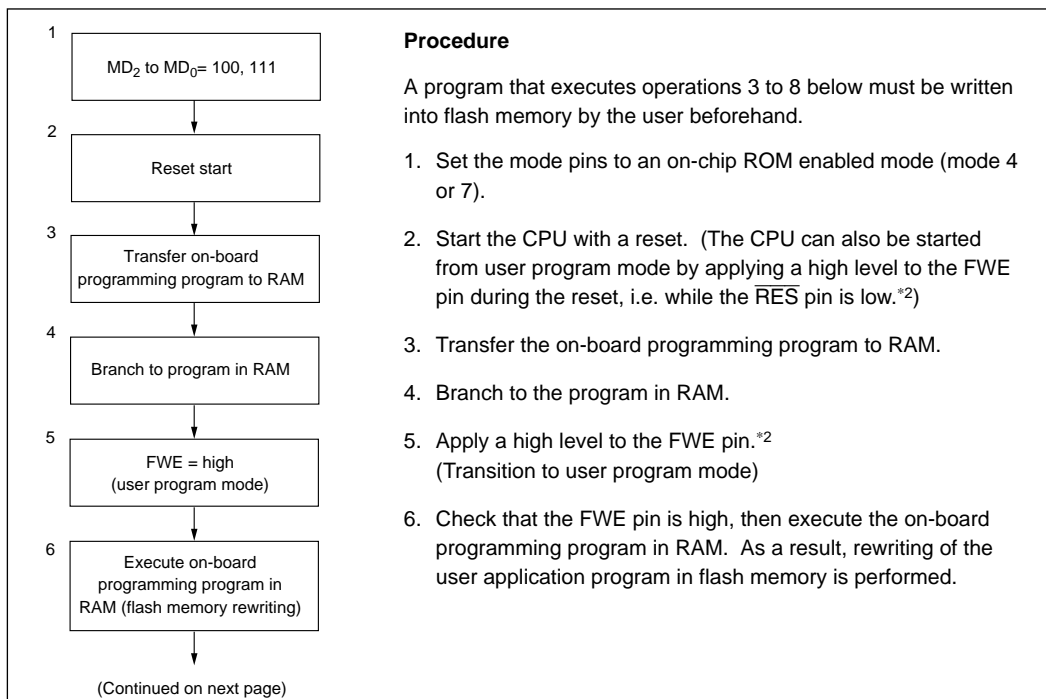
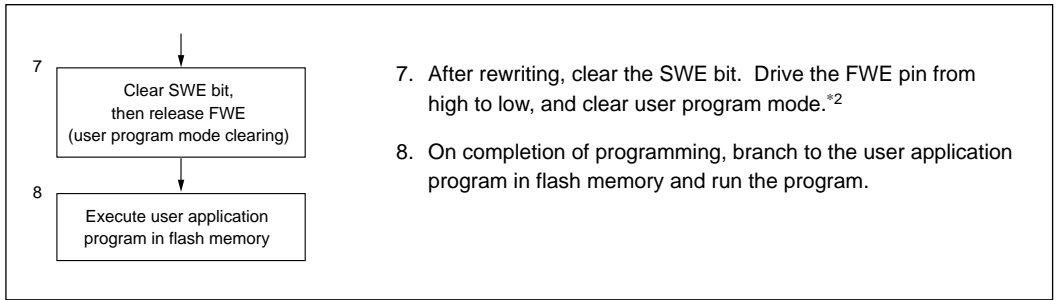


Figure 20-7 Example of User Program Mode Operation





- Notes: 1. Do not apply a constant high level to the FWE pin. To prevent inadvertent programming or erasing due to program runaway, etc., apply a high level to the FWE pin only when the flash memory is programmed or erased (including execution of flash memory emulation using RAM). Memory cells may not operate normally if overprogrammed or overerased due to program runaway, etc. Also, while a high level is applied to the FWE pin, the watchdog timer should be activated to prevent overprogramming or overerasing due to program runaway, etc. Flash memory rewriting should not be carried out in mode 2. When mode 2 is set, the FWE pin must be driven low.
2. For further information on FWE application and disconnection, see section 20.7, Flash Memory Programming and Erasing Precautions.

**Figure 20-7 Example of User Program Mode Operation (cont)**

## 20.4 Flash Memory Programming/Erasing

A software method, using the CPU, is employed to program and erase flash memory in the on-board programming modes. There are four flash memory operating modes: program mode, erase mode, program-verify mode, and erase-verify mode. Transitions to these modes are made by setting the PSU, ESU, P, E, PV, and EV bits in FLMCR. The state transitions permitted by the various FLMCR bit settings are shown in figure 20-9.

The flash memory cannot be read while being programmed or erased. Therefore, the program (user program) that controls flash memory programming/erasing should be located and executed in on-chip RAM or external memory.

See section 20.7, Flash Memory Programming and Erasing Precautions, for points to note concerning programming and erasing, and section 23.2.4, Flash Memory Characteristics, for the wait times after setting or clearing FLMCR bits.

- Notes:
1. Operation is not guaranteed if setting/resetting of the SWE, ESU, PSU, EV, PV, E, and P bits in FLMCR is executed by a program in flash memory.
  2. When programming or erasing, set FWE to 1 (programming/erasing will not be executed if FWE = 0).

### 20.4.1 Program Mode

When writing data or programs to flash memory, the program/program-verify flowchart shown in figure 20-10 should be followed. Performing programming operations according to this flowchart will enable data or programs to be written to flash memory without subjecting the device to voltage stress or sacrificing program data reliability. Programming should be carried out 32 bytes at a time.

The wait times ( $x$ ,  $y$ ,  $z$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\epsilon$ ,  $\eta$ ) after bits are set or cleared in the flash memory control register (FLMCR) and the maximum number of programming operations ( $N$ ) are shown in table 23-9 in section 23.2.4, Flash Memory Characteristics.

Following the elapse of ( $x$ )  $\mu$ s or more after the SWE bit is set to 1 in FLMCR, 32-byte data is written consecutively to the write addresses. The value of the lower 8 bits of the first address written to must be divisible by 32 (H'00, H'20, H'40, H'60, H'80, H'A0, H'C0, or H'E0). Thirty-two consecutive byte data transfers are performed. The program address and program data are latched in the flash memory. A 32-byte data transfer must be performed even if writing fewer than 32 bytes; in this case, H'FF data must be written to the extra addresses.

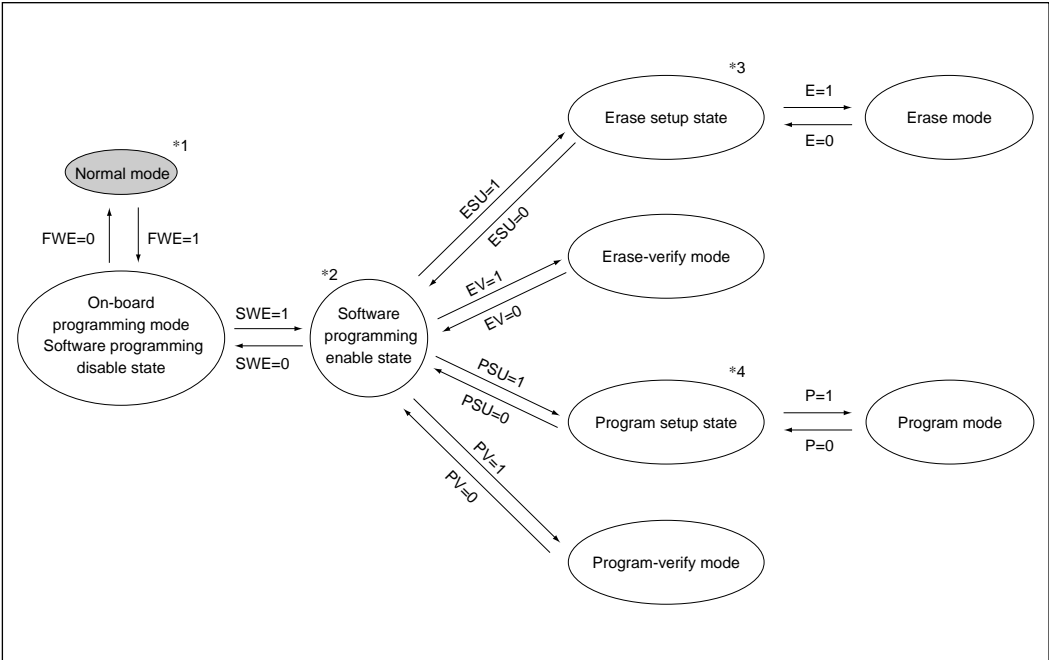
Next, preparation for program mode (program setup) is carried out by setting the PSU bit in FLMCR, and after the elapse of ( $y$ )  $\mu$ s or more, the watchdog timer is set to prevent overprogramming in the event of program runaway, etc.

Set a value greater than  $(z + \alpha) \mu\text{s}$  as the WDT overflow period. The operating mode is then switched to program mode by setting the P bit in FLMCR. The time during which the P bit is set is the flash memory programming time. Make a program setting so that the time for one programming operation is within the range of  $(\alpha) \mu\text{s}$ .

#### **20.4.2 Program-Verify Mode**

In program-verify mode, the data written in program mode is read to check whether it has been correctly written in the flash memory.

After the elapse of a given programming time, the programming mode is exited (the P bit in FLMCR is cleared, then the watchdog timer and PSU bit are cleared at least  $(\alpha) \mu\text{s}$  later). The operating mode is then switched to program-verify mode by setting the PV bit in FLMCR. Before reading in program-verify mode, a dummy write of H'FF data should be made to the addresses to be read. The dummy write should be executed after the elapse of  $(\gamma) \mu\text{s}$  or more. When the flash memory is read in this state (verify data is read in 16-bit units), the data at the latched address is read. Wait at least  $(\epsilon) \mu\text{s}$  after the dummy write before performing this read operation. Next, the originally written data is compared with the verify data, and reprogram data is computed (see figure 20-10) and transferred to RAM. After 32 bytes of data have been verified, exit program-verify mode, wait for at least  $(\eta) \mu\text{s}$ , then clear the SWE bit in FLMCR. If reprogramming is necessary, set program mode again, and repeat the program/program-verify sequence as before. However, ensure that the program/program-verify sequence is not repeated more than (N) times on the same bits.



- Notes: 1. : Normal mode : On-board programming mode
- Do not make a state transition by setting or clearing multiple bits simultaneously.
  - After a transition from erase mode to the erase setup state, do not enter erase mode without passing through the software programming enable state.
  - After a transition from program mode to the program setup state, do not enter program mode without passing through the software programming enable state.

**Figure 20-8 Flash Memory Programming State Transitions Permitted by FLMCR Bit Settings**

## 20.4.3 Sample 32-Byte Programming Flowchart

### 32-Byte Programming Flowchart

- Preliminary -

- Programming should be performed in the erased state. Do not perform additional programming. (Perform 32-byte programming on memory after all 32 bytes have been erased.)
- Data transfer is performed by byte transfer (word transfer is not possible). The first address written to must be at a 32-byte boundary with a lower 8-bit value of H'00, H'20, H'40, H'60, H'80, H'A0, H'C0, or H'E0. A 32-byte data transfer must be performed even if writing fewer than 32 bytes; H'FF data must be written to the extra addresses.
- Verify data is read in 16-bit (word) units. (Byte-unit reading is also possible.)
- Reprogram data is determined by the computation shown in the table below.
- An area for storing write data (32 bytes) and an area for storing reprogram data (32 bytes) must be provided in RAM. The contents of the latter are rewritten in accordance with the reprogram data computation.
- The values of x, y, z,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\epsilon$ ,  $\eta$  and N are shown in section 23.2.4, Flash Memory Characteristics.

Write Data	Verify Data	Reprogram Data	Comments
0	0	1	Programming completed
0	1	0	Programming incomplete; reprogram
1	0	1	
1	1	1	Still in erased state; no action

Note: \* The memory erased state is "1". Programming is performed on "0" reprogram data.

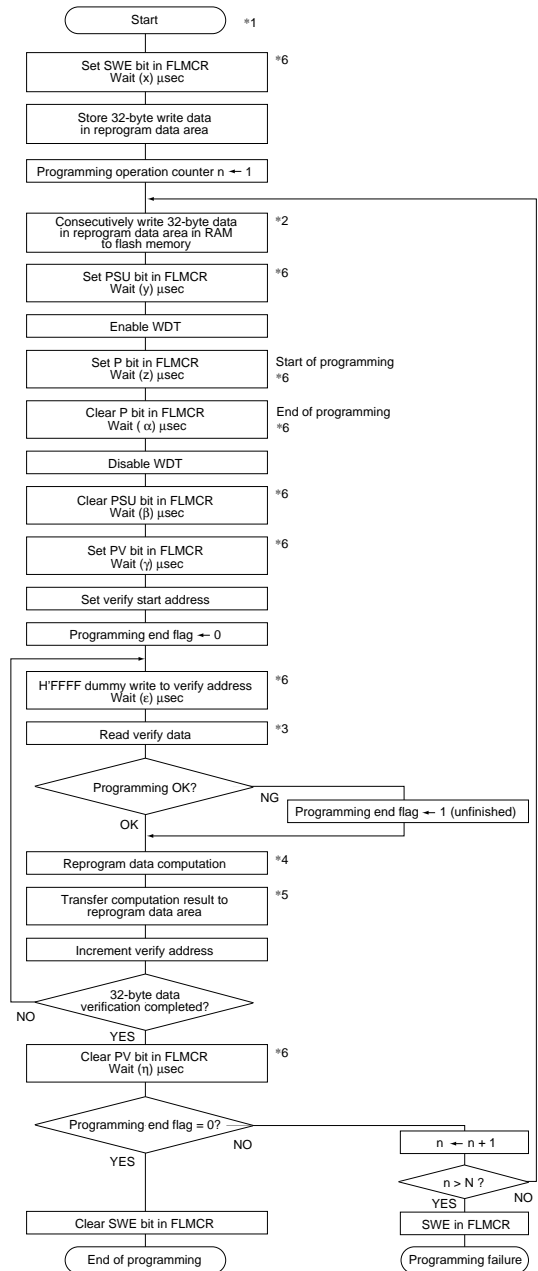


Figure 20-9 Programming Flowchart

## 1. Registers and method used

After the values of the following registers (R1, R2, R3, R4) have been specified, arbitrary 32-byte data can be written to an arbitrary 32-byte boundary address by calling the fwrites subroutine.

R1: Page of address storing data to be written  
 R2: Address storing data to be written  
 R3: Page of programming address in flash memory  
 R4: Programming address in flash memory

## 2. Wait times after setting/clearing FLMCR bits

The number of software loops required to provide a specific wait time depends on the operating frequency. The relevant operating frequency can be specified by setting the MHZ symbol value.

## 3. In this program the wait time (number of loops) is calculated on the assumption that the scb/f instruction jump destination is located at an even address in two-state access space (on-chip RAM).

## 4. Programming should be performed only on flash memory in the erased state.

## List 1: Sample Program for Programming 32 Bytes

- Preliminary -

```

0001:;*****
0002:;* fwrites, src          (Ver. 0.07)    - Preliminary -          *
0003:;* Sample program for H8/539FS flash memory 32-byte programming *
0004:;*                                                                *
0005:;*****
0006:;
0007:;
0008:MHZ          .equ    d'16      ; Depends on operating frequency (16 MHz)
0009:RAMSTR       .equ    H'EE80    ; Program transfer destination RAM address
0010:MAXWT       .equ    d'1000    ; Maximum number of writes
0011:;Register addresses
0012:FLMCR       .equ    H'FEE0    ; Flash memory control register
0013:TCSR        .equ    H'FF10    ; Timer control/status register
0014:WCR         .equ    H'FF14    ; Wait control register
0015:;
0016:;
0017:    .align H'2
0018:main:       .equ    $
0019:    ldc.b   #H'00:8,tp      ; Stack page register setting
0020:    mov.w   #H'FE80,sp     ; Stack pointer setting
0021:    ldc.b   #H'00:8,dp     ; Data page register setting
0022:    ldc.b   #H'00:8,ep     ; Extension page register setting
0023:    ;
0024:    ;
0025:    ;

```

```

0026:  mov.w  #prog_start,R0 ; Transfer start address
0027:  mov.w  #prog_start,R1 ; Transfer end address
0028:  bsr    tensou:16      ; Program transfer to RAM
0029:;
0030:  ; Argument setting and subroutine call
0031:  jsr    @RAMSTR       ; JMP SUB to RAM area program (prog_start)
0032:  ; (All-mat definition programming example)
0033:;
0034:main_end:  ; End of programming
0035:  bra   main_end
0036:;
0037:;
0038:;*****
0039:;* tensou SUB *
0040:;* Copy RAM execution program to RAM *
0041:;*****
0042:  .align  H'2
0043:tensou: .equ  $
0044:  ; Arguments      R0 transfer start address
0045:  ; R1 transfer end address
0046:  stm    (R0-R3),@-sp ; Save used registers
0047:  ; R2 transfer destination RAM address
0048:  ; R3 transfer data work
0049:  mov.w  #RAMSTR,R2   ; Transfer destination address setting
0050:tensou 01:
0051:  mov.w  @R0+,R3      ; ROM PROG DATA → R3
0052:  mov.w  R3,@R2+     ; R3 → RAM WRITE
0053:  cmp.w  R1,R0       ; R1:END R0:INCREASED ADDR.
0054:  bit   tensou01    ; R0 = R1 → NEXT INSTRUCTION.
0055:  ldm    @sp+,(R0-R3) ; Restore used registers
0056:  rts    ; Subroutine return
0057:;
0058:;
0059:;*****
0060:;** Start of program for transfer to RAM **
0061:;*****
0062:  .align  H'2
0063:prog_start: .equ  $ ; Start of program for transfer to RAM
0064:  ;
0065:  ;
0066:  ;
0067:;
0068:;
0069:;*****
0070:; all0_write SUB *
0071:; Flash memory all-mat H'00 programming *
0072:;*****
0073:  .align  H'2
0074:all0_write: .equ  $
0075:  ; Arguments      R0 Return code 0: Normal
0076:  ;                1: Programming error
0077:  ;                2: Initial value, etc., error
0078:  stm    (R1-R5),@-sp ; Save used registers
0079:  ; R1 Programming data page specification
0080:  ; R2 Programming data address specification
0081:  ; R3 Programming page specification
0082:  ; R4 Programming address specification
0083:  ; R5 Wait loop counts
0084:  link  FP,#-d'34    ; Reserve stack area
0085:;
0086:  mov.w  FP,R2
0087:  sub.w  #d'32,R2
0088:  mov.w  R2.@(-34,FP) ; Save programming data start address

```

```

0089:  mov.w   #d'15,R5
0090:all0_w01:
0091:  clr.w   @R2+           ; Zero-clear 32-byte programming data
0092:  scb/f   R5,all0_w01
0093:;
0094:  clr     @WCR           ; No wait state inserted
0095:  ldc.w   #H'0700,SR     ; Disable interrupts during programming/erasing
0096:;
0097:  bset    #d'6,@FLMCR    ; Set SWE bit
0098:  mov.w   #(d'10*MHZ/d'8),R5
0099:;                               ; Set SWE wait loop counter
0100:all0_w02:
0101:  scb/f   R5,all0_w02    ; SWE wait (10 µs or more)
0102:;
0103:;                               ; Argument setting and subroutine call
0104:  clr.w   R1             ; Set programming data page
0105:  mov.w   @(-34FP),R2    ; Set programming data address
0106:  mov.w   #H'01,R3      ; Set programming page
0107:all0_w03:
0108:  clr.w   R4             ; Set start of page as programming address
0109:all0_w04:
0110:;
0111:  bsr     fwrites        ; 32-byte programming
0112:;
0113:  tst.w   R0             ; Return code check
0114:  bne     all0_w06       ; If error, end
0115:;
0116:  cmp.w   #H'FFE0,R4     ; Last address of page?
0117:  bhs     all0_w05       ; If last address, next page
0118:  adds.w  #H'20,R4      ; Add 32 bytes to programming address
0119:  bra     all0_w04       ;
0120:all0_w05:
0121:  cmp.b   #H'02,R3      ; Last page?
0122:  bhs     all0_w06       ; If last page,end
0123:  adds.b  #H'01,R3      ; Page address + 1
0124:  bra     all0_w03       ;
0125:all0_w06:
0126:  bclr.b  #d'6,@FLMCR   ; Clear SWE bit
0127:  mov.w   #(d'0*MHZ/d'8),R5
0128:;                               ; Set SWE clear wait counter
0129:all0_w07:
0130:  scb/f   R5,all0_w07    ; SWE clear wait (0 µs or more)
0131:;
0132:  unlk   FP             ; Release stack area
0133:  ldm     @sp+,(R1-R5)   ; Restore used registers
0134:  rts     ; Subroutine return
0135:all0_write_end:  .equ  $
0136:;
0137:;
0138:;*****
0139:;* Flash memory 32-byte programming (SUB)                               *
0140:;* Arguments R0 -/ Return code 0: Normal                               *
0141:;*****
0142:  .align H'2
0143:fwrites:.equ      $
0144:;                               ; Arguments      R0 transfer start address
0145:;                               ;                   1: Programming error
0146:;                               ;                   2: Initial value, etc., error
0147:;                               ; R1 Programming data page
0148:;                               ; R2 Programming data address
0149:;                               ; R3 Programming page
0150:;                               ; R4 Programming address
0151:  stm     (R1-R5),@-sp    ; Save used registers

```



```

0152:                ; R0 Wait loop count (for PV)
0153:                ; R1 Programming counter
0154:                ; R2 Number of repeats,programming data address
0155:                ; R3 Reprogramming data address
0156:                ; R4 Programming address
0157:                ; R5 Wait loop counts
0158: stc.b   ep,@-sp      ; Save used page register
0159: link   FP,#-d'72     ; Reserve stack area
0160:;
0161: mov.w   FP,R3
0162: subs.w  #d'32,R3
0163: mov.w   R3,@(-34,FP)  ; Save reprogramming data address
0164:;
0165: mov.w   FP,R3
0166: subs.w  #d'66,R3
0167: mov.w   R3, @(-68,FP) ; Save programming data address
0168: ldc.b   @(5,FP),ep    ; Programming source data page
0169: mov.w   @(6,FP),R4    ; Programming source data address
0170: mov.w   #d'15,R5
0171:fwrites 01:
0172: mov.w   @R4+,R2
0173: mov.w   R2,@(34,R3)   ; Copy programming source data to reprogramming data
0174: mov.w   R2,@R3+      ; Copy programming source data to programming data
0175: scb/f   R5,fwrites01
0176:;*****
0177:;
0178: clr.w   R1            ; Clear programming counter
0179: clr.w   @(-70,FP)    ; Clear programming NG flag
0180:fwrites 02:
0181:;
0182: best.b  #d'2,@FLMCR   ; Set PV bit
0183: mov.w   #(d'4*MHZ/d'8),R5
0184:                ; Set PV wait counter
0185:fwrites 03:
0186: scb/f   R5,fwrites03  ; PV wait (4 µs or more)
0187:;
0188: mov.w   @(-68,FP),R2  ; Set programming data address
0189: mov.w   @(-34,FP),R3  ; Set reprogramming data address
0190: ldc.b   @(9,FP)ep     ; Programming address page
0191: mov.w   @(10,FP),R4   ; Programming address
0192: mov.w   #d'15,R0      ; Verify loop counter
0193:fwrites 04:
0194: mov.w   #H'FFFF,@R4   ; Dummy write (latch)
0195: mov.w   #(d'2*MHZ/d'8),R5
0196:                ; Set loop counter for wait after latch
0197:fwrites 05:
0198: scb/f   R5,fwrites05  ; Wait after latch (2 µs or more)
0199: mov.w   @R4+,R5       ; Data read,address + 2
0200: not.w   R5            ; Data inversion
0201: mov.w   R5,@(-72,FP)  ; Set inverted data in calculation work area
0202: or.w    @R3,R5        ; Inverted data or reprogramming data
0203: mov.w   R5,@R3+      ; Set reprogramming data, address + 2
0204: mov.w   @(-72,FP),R5  ; Calculation work area
0205: and.w   R2+,R5       ; Inverted data and programming data
0206: tst.w   R5            ; In case of read data 0 and programming data 1
0207: beq    fwrites06
0208: mov.w   #H'1,@(-70,FP) ; Programming NG flag ← NG
0209:fwrites 06:
0210: scb/f   R0,fwrites04  ; Verify loop counter (R0) = 15-0
0211:;
0212: bclr.b  #d'2,@FLMCR   ; Clear PV bit
0213: mov.w   #(d'4*MHZ/d'8),R5
0214:                ; Set counter for wait after clearing PV

```

```

0215:fwrites 07:
0216: scb/f R5,fwrites07 ; Wait after clearing PV (4 µs or more)
0217:;*****
0218:;
0219: mov.w @(-34,FP),R3 ; Reprogramming data address
0220: mov.w #d'15,R5 ; Repeat count = 16
0221:fwrites 08:
0222: cmp.w #H'FFFF,@R3+ ; 32-byte reprogramming data all FF? R3 + 2
0223: bne fwrites09 ; Decision: if not FF, go to reprogramming
0224: scb/f R5,fwrites08 ;
0225: bra fwrites16 ; If all FF, go to end processing
0226:fwrites 09:
0227: add.w #H'01,R1 ; Programming counter (R1) + 1
0228: cmp.w #MAXWT,R1 ; Count decision (max.programming times)
0229: bhi fwrites16 ; If count > max.programming times,go to end processing
0230: tst.w @(-70,FP) ; When programming NG flag ≠ 0
0231: bne fwrites16 ; Go to end processing
0232:;*****
0233:;
0234: ; 32-byte dummy write (latch)
0235: mov.w @(-34,FP),R3 ; Set reprogramming data address
0236: ldc.b @(9,FP),ep ; Programming address page
0237: mov.w @(10,FP),R4 ; Programming address
0238: mov.w #d'31,R5 ; Repeat count = 32
0239:fwrites 10:
0240: mov.b @R3+,R2 ; Reprogramming data (byte unit), R3 + 1
0241: mov.b R2,@R4+ ; Dummy write (byte unit), R4 + 1
0242: scb/f R5,fwrites10 ; Repeat for 32 bytes
0243:;
0244: bset.b #d'4,@FLMCR ; Set PSU bit
0245: MOV.W #(d'50*MHZ/d'8),R5
0246: ; Set PSU wait counter
0247:fwrites 11:
0248: scb/f R5,fwrites11 ; PSU wait (50 µs or more)
0249:;
0250: ; P setting/P clearing (programming)
0251: mov.w #H'A57B,@TCSR ; Set watchdog timer
0252: mov.w #((d'200*MHZ-d'4)/d'8),R5
0253: ; Set P time loop counter
0254: bra fwrites12 ; Adjust so that scb/f jump destination is even address
0255: .align H'2 ; Adjust so that scb/f jump destination is even address
0256:fwrites 12:
0257: nop ; Adjust so that scb/f jump destination is even address
0258: ; If odd,insert NOP
0259: bset.b #d'0,@FLMCR ; Set P bit
0260: ; Adjust so that P wait scb/f jump destination is even address
0261:fwrites 13:
0262: SCB/F R5,fwrites13 ; P wait (200 Ms) programming time wait
0263: bclr.b #d'0,@FLMCR ; Clear P bit
0264:;
0265: mov.w #(d'10*MHZ/d'8),R5
0266: ; Set counter for wait after clearing P
0267:fwrites 14:
0268: scb/f R5,fwrites14 ; Wait after clearing P (10 µs)
0269: mov.w #H'A500,@TCSR ; Stop watchdog timer
0270:;
0271: bclr.b #d'4,@FLMCR ; Clear PSU bit
0272: mov.w #(d'10*MHZ/d'8),R5
0273: ; Set counter for wait after clearing PSU
0274:fwrites 15:
0275: scb/f R5,fwrites15 ; PSU clear wait (10 µs or more)
0276: bra fwrites02
0277:;*****

```

```

0278;;
0279:fwrites 16:
0280:  clr.w  R0                ; Set return value (R0) = OK
0281:  cmp.w  #MAXWT,R1        ; Programming times decision
0282:  bls   fwrites17         ; Count not exceeded
0283:  mov.w  #H'01,R0         ; Return value (R0) = count exceeded (programming NG)
0284:fwrites 17:
0285:  tst.w  @(-70,FP)         ; When programming NG flag ≠ 0
0286:  beq   fwrites18         ;
0287:  mov.w  #H'02,R0         ; Return value (R0) = initial value error
0288:fwrites 18:
0289;;
0290:  unlk  FP                ; Release stack area
0291:  ldc.b  @sp+,ep           ; Restore used page register
0292:  ldm   @sp+,(R1-R5)       ; Restore used registers
0293:  rts                    ; Subroutine return
0294;;
0295:;*****
0296;;
0297:  ;
0298:  ;
0299:  ;
0300:prog_stop:.equ  $        ; End of program for transfer to RAM
0301;;
0302;;
0303:  .end
0304:

```

- Notes: 1. The sample programs in this manual are provided to illustrate programming/erasing of flash memory incorporated in an MCU. They do not make provisions for various kinds of applications, and cannot be used as they are. They are intended solely for reference in program development.
2. Program operation must be confirmed in actual use.
  3. These programs are preliminary samples, and are subject to change without notice for the purpose of improvement, etc.

#### 20.4.4 Erase Mode

When erasing flash memory, the single-block erase flowchart shown in figure 20-10 (single-block erase) should be followed for each block.

The wait times ( $x$ ,  $y$ ,  $z$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\epsilon$ ,  $\eta$ ) after bits are set or cleared in the flash memory control register (FLMCR) and the maximum number of erase operations ( $N$ ) are shown in table 23-9 in section 23.2.4, Flash Memory Characteristics.

To erase flash memory contents, make a 1-bit setting for the flash memory area to be erased in erase block register 1 (EBR1) at least ( $x$ )  $\mu$ s after setting the SWE bit to 1 in FLMCR. Next, preparation for erase mode (erase setup) is carried out by setting the ESU bit in FLMCR, and after the elapse of ( $y$ )  $\mu$ s or more, the watchdog timer is set to prevent overerasing in the event of program runaway, etc. Set a value greater than ( $z + \alpha$ ) ms as the WDT overflow period. The operating mode is then switched to erase mode by setting the E bit in FLMCR. The time during which the E bit is set is the flash memory erase time. Ensure that the erase time does not exceed ( $z$ ) ms.

Note: With flash memory erasing in the H8/539F S-mask model, preprogramming (setting all memory data in the memory to be erased to all “0”) is not necessary before starting the erase procedure.

#### 20.4.5 Erase-Verify Mode

In erase-verify mode, data is read after memory has been erased to check whether it has been correctly erased.

After the elapse of a the erase time, erase mode is exited (the E bit in FLMCR is cleared, then the watchdog timer and ESU bit are cleared at least ( $z$ )  $\mu$ s later), and the operating mode is switched to erase-verify mode by setting the EV bit in FLMCR. Before reading in erase-verify mode, a dummy write of H'FF data should be made to the addresses to be read. The dummy write should be executed after the elapse of ( $y$ )  $\mu$ s or more. When the flash memory is read in this state (verify data is read in 16-bit units), the data at the latched address is read. Wait at least ( $\epsilon$ )  $\mu$ s after the dummy write before performing this read operation. If the read data has been erased (all “1”), a dummy write is performed to the next address, and erase-verify is performed. If the read data is unerased, perform an erase-verify on the next block. The erase-verify operation is carried out on all the erase blocks; the erase block register bit for an erased block should be cleared to prevent excessive application of the erase voltage. When verification is completed, exit erase-verify mode, and wait for at least ( $\eta$ )  $\mu$ s. If erasure has been completed on all the erase blocks after completing erase-verify operations on all these blocks, clear the SWE bit in FLMCR. If there are any unerased blocks, set erase mode again, and repeat the erase/erase-verify sequence as before. However, ensure that the erase/erase-verify sequence is not repeated more than ( $N$ ) times.

## 20.4.6 Sample Flowchart for Erasing One Block

### Flowchart for Erasing One Block

- Preliminary -

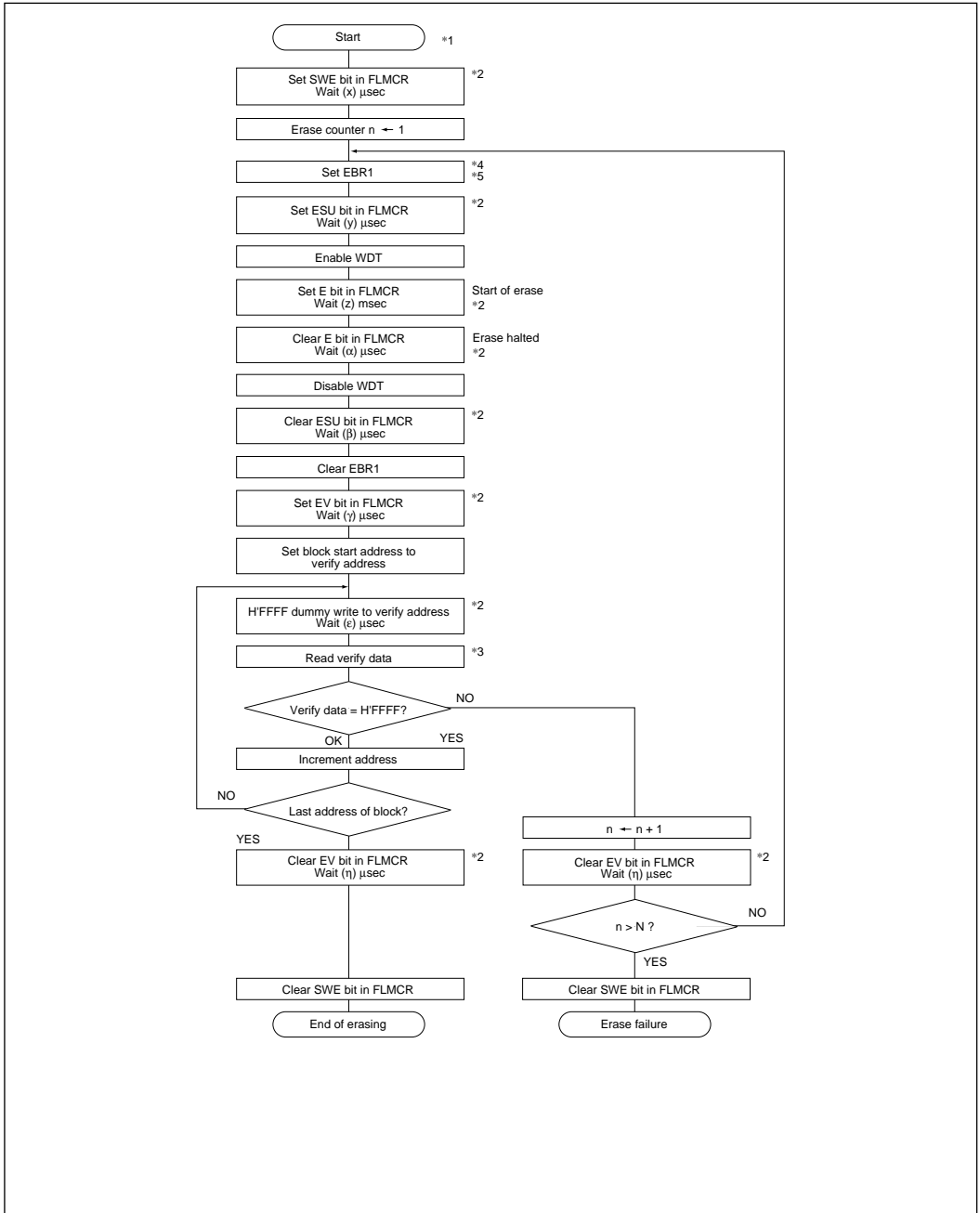


Figure 20-10 Flowchart for Erasing One Block

## Setting Loop Counter Values and WDT Overflow Intervals in the Program

- Preliminary -

A wait time is necessary after a bit is set or cleared in the flash memory control register (FLMCR). In the program examples, wait times are provided by means of software loops. The software loop counter value depends on the operating frequency, and whether the scb/f instruction is located at an even address and in two-state access space, and whether wait state insertion is disabled. In these program examples, the calculation assumes an even address, two-state access space, and no wait state insertion.

The value set in TCSR to provide the watchdog timer overflow interval setting when erase mode is selected depends on the operating frequency. TCSR set values for different operating frequencies are shown in table 20-8.

As software loops are used, there is intrinsic error in the wait times, and the calculated value and actual time may not be the same. Therefore, initial values should be set so that the total programming time does not exceed  $tP$  s/byte (max.), and the total erase time does not exceed  $tE$  s (max.). The total programming time,  $tP$   $\mu$ s/byte (max.), and total erase time,  $tE$  s (max.), are shown in table 23-9, Flash Memory Characteristics.

The set value for the watchdog timer (WDT) is calculated on the basis of the number of instructions including the programming time and erase time from the time the WDT is started until it stops. Therefore, no other instructions should be added between starting and stopping of the WDT in these program examples.

The loop counter value for each operating frequency is calculated as shown below.

### Formulas:

Formulas for calculating loop counter value in program

**(1) Program time (P bit setting) calculation formula** When the scb/f instruction is at an even address in two-state access space, the processing time is 4 states when the register value = 0 (no branch) and 8 states when the register value  $\geq 1$  (branch). Thus the calculation formula, with truncation of the decimal part, is as follows.

$$\text{Loop counter value} = \frac{(\text{Wait time } (\mu\text{s}) \times \text{operating frequency (MHz)}) \text{ states} - 4 \text{ states}}{8 \text{ states}}$$

**(2) Erase time (E bit setting) calculation formula** For the same access space as in (1) above, the calculation formula is as follows.

$$\text{Loop counter value} = \frac{(\text{Wait time } (\mu\text{s}) \times \text{operating frequency (MHz)}) \text{ states} - 14 \text{ states}}{18 \text{ states}}$$

**(3) Wait time (time after FLMCR bit setting/clearing other than P bit setting or E bit setting)** With the same number of states as in (1) above, the calculation formula, with rounding of the decimal part, is as follows.

$$\text{Loop counter value} = \frac{(\text{Wait time } (\mu\text{s}) \times \text{operating frequency (MHz)}) \text{ states} - 4 + 4 \text{ states}}{8 \text{ states}}$$

**Table 20-8 WDT Overflow Interval Setting when Erase Mode is Set**

Operating Frequency [MHz]	Variable
	Value Set in TCSR
10 MHz to 16 MHz	H'A57F
2 MHz to less than 10 MHz	H'A57E

**(1) Registers and method used** After the values of the following register (R0) has been specified, an block can be erased by calling the ferases subroutine.

R0: Specification of block to be erased by EBR1 bit position in lower byte

**(2) Wait times after setting/clearing FLMCR bits** The number of software loops required to provide a specific wait time depends on the operating frequency. The relevant operating frequency can be specified by setting the MHZ symbol value.

**(3) In this program the wait time (number of loops) is calculated on the assumption that the scb/f instruction jump destination is located at an even address in two-state access space (on-chip RAM).**

## List 2: Sample Block Erase Program

- Preliminary -

```

0001:;*****
0002:;* ferase, src          (B version Ver. 0.05)- Preliminary - *
0003:;* Sample program for H8/539FS flash memory block erasing *
0004:;*                                                              *
0005:;*****
0006:;
0007:;
0008:MHZ .equ d'16          ; Depends on operating frequency (16 MHz)
0009:RAMSTR .equ H'EE80     ; Program transfer destination RAM address
0010:MAXET .equ d'60       ; Maximum number of erases
0011:;Register addresses
0012:FLMCR .equ H'FEE0     ; Flash memory control register
0013:EBR1 .equ H'FEE2     ; Target block specification register
0014:TCSR .equ H'FE10     ; Timer control/status register
0015:WCR .equ H'FE14      ; Wait control register
0016:MDCR .equ H'FE19     ; Mode control register
0017:;
0018:;
0019:          .align H'2
0020:main:.equ $
0021:          ldc.b #H'00:B,tp          ; Stack page register setting
0022:          mov.w #H'FE80,sp         ; Stack pointer setting
0023:          ldc.b #H'00:8,dp         ; Data page register setting
0024:          ldc.b #H'00:8,ep         ; Extension page register setting
0025:          ;
0026:          ;
0027:          ;
0028:          mov.w #prog_start,R0     ; Transfer start address
0029:          mov.w #prog_stop,R1     ; Transfer end address
0030:          bsr tensou:16           ; Program transfer to RAM
0031:;
0032:          ; Argument setting and subroutine call
0033:          jsr @RAMSTR             ; JMP SUB to RAM area program (prog_start)
0034:          ; (All-mat erase example)
0035:;
0036:main_end:          ; End of erase
0037:          bra main_end

```



```

0038;;
0039;;
0040;*****
0041;* tensou SUB *
0042;* Copy RAM execution program to RAM *
0043;*****
0044: .align H'2
0045:tensou: .equ $
0046: ; Arguments R0 transfer start address
0047: ; R1 transfer end address
0048: stm(R0-R3),@-sp ; Save used registers
0049: ; R2 transfer destination RAM address
0050: ; R3 transfer data work
0051: mov:w #RAMSTR,R2 ; Transfer destination address setting
0052:tensou 01:
0053: mov.w @R0+,R3 ; RAM PROG DATA → R3
0054: mov.w R3,@R2+ ; R3 → RAM WRITE
0055: cmp.w R1,R0 ; R1:END R0:INCREASED ADDR.
0056: blt tensou 01 ; R0=R1 → NEXT INSTRUCTION.
0057: ldm @sp+,(R0-R3) ; Restore used registers
0058: rts ; Subroutine return
0059;;
0060;;
0061;*****
0062;** Start of program for transfer to RAM **
0063;*****
0064: .align H'2
0065:prog_start:.equ $ ; Start of program for transfer to RAM
0066: ;
0067: ;
0068: ;
0069;;
0070;;
0071;*****
0072;** all_erase SUB *
0073;; Flash memory all-mat erase *
0074;*****
0075: .align H'2
0076:all_erase:.equ $
0077: ; Arguments
0078: ; R0-/Return code 0: Normal
0079: ; 1: Erase error
0080: ; R1-/Erase NG block
0081:stm(R2-R3),@-sp ; Save used registers
0082: ; R0 R0 Erase-verify target block
0083: ; R2 Wait loop counts
0084;;
0085: clr.b @WCR ; No wait state inserted
0086: ldc.w #H'0700,sr ; Disable interrupts during programming/erasing
0087;;
0088: bset.b #d'6,@FLMCR ; Set SWE bit
0089: mov.w #(d'10,*MHZ/d'8),R2
0090: ; Set SWE wait loop counter
0091:all_e01:
0092: scb/f R2,all_e01 ; SWE wait (10 μs or more)
0093:
0094: mov.w #H'00FF,R0 ; Erase target block specification
0095: bsr erasevf:16 ; Erase-verify
0096: tst.b R0 ; Unerased block check
0097: beq all_e02 ; If there is unerased block,
0098: bsr ferases ; erase block
0099: ; Return R0 return code(0: normal, 1: erase NG)
0100: ; R1 Erase NG block

```

```

0101:all_e02
0102:      bclr #d'6,@FLMCR      ; Clear SWE bit
0103:      mov.w #(d'0*MHZ/d'8),R2
0104:                                     ; Set SWE clear wait counter
0105:all_e03:
0106:      scb/f R2,all_e03      ; SWE clear wait (0 µs or more)
0107:;
0108:      ldm @sp+,(R2-R3)      ; Restore used registers
0109:      rts                    ; Subroutine return
0110:all_erase_end: .equ $
0111:;
0112:;
0113:;*****
0114:;* ferase SUB
0115:;*   Flash memory block erase (SUB)
0116:;*****
0117:      .align H'2
0118:ferases: .equ $
0119:                                     ; Arguments      R0 Erase target block/return code
0120:                                     ; Return code  0: Normal
0121:                                     ;                    1: Erase error
0122:                                     ; R1 - / erase NG block
0123:      stm(R2-R5),@-sp        ; Save used registers
0124:                                     ; R1 Erase counter
0125:;
0126:      and.w #H'00FF,R0        ; Mask bits except EBR1
0127:      cmp.b #H'C2,@MDCR      ; Mode check
0128:      bne ferases 01         ; If mode 2,
0129:      and.w #H'000F,R0        ; mask target blocks except page 0
0130:ferases 01:
0131:;
0132:      mov.w #H'0001,R1
0133:ferases 02:
0134:      bsr erase:16            ; Multiple-block erase subroutine
0135:                                     ; R0 Erase target block
0136:                                     ; R1 Erase count
0137:      bsr erasevf:16          ; Erase-verify subroutine
0138:                                     ; R0 Erase target block/unerased block
0139:      tst.b R0
0140:      beq ferase 03          ; If all erasing completed, go to end
0141:      add.w #H'01,R1
0142:      cmp.w #MAXET,R1
0143:      bls ferases 02         ; If erase count ≤ max erase count, go to re-erase
0144:ferases 03:
0145:;
0146:      mov.w R0,R1            ; Set NG block in R1
0147:      tst.w R0                ; Are there unerased blocks?
0148:      beq ferases 04         ; If so,
0149:      mov.w #H'0001,R0        ; return code = erase error
0150:ferases 04:
0151:;
0152:      ldm @sp+,(R2-R5)      ; Restore used registers
0153:      rts                    ; Subroutine return
0154:;
0155:;
0156:;*****
0157:;* erases SUB
0158:;*   Flash memory multiple-block erase (SUB)
0159:;*****
0160:      .align2
0161:erases: .equ $
0162:                                     ; Arguments      R0 Erase target block
0163:                                     ; R1 Erase count (not used at present)

```

```

0164:          stm (R2-R4),@-sp      ; Save used registers
0165:          ; R2 Block bit number
0166:          ; R3 Wait loop count 1
0167:          ; R4 Wait loop count 2
0168:;;
0169:          clr.w R2              ; Block bit number
0170:erases 01:
0171:          btst.w R2,R0           ; R0 bit R2 test
0172:          beq erases 07         ; If not erase target block, go to next block
0173:          bset.b R2,@EBR       ; Set erase target block bit in EBR1
0174:;;
0175:          bset.b #d'5,@FLMCR    ; Set ESU bit
0176:          mov.w #(d'200*MHZ/d'8),R3
0177:          ; Set ESU wait counter
0178:erases 02:
0179:          stb/f R3,erases 02   ; ESU wait (200 µs or more)
0180:;;
0181:          ; E setting/E clearing (erase)
0182:          mov.w #H'A57F,@TCSR   ; Set watchdog timer
0183:          mov.w #((d'5000*MHZ-d'14),d'18/H'10000),R3
0184:          mov.w #((d'5000*MHZ-d'14),d'18 & H'FFFF),R4
0185:          ; Set E time loop count (5.0 ms)
0186:;;
0187:          bra erases 03        ; Adjust so that scb/f jump destination is an
                                ; even address
0188:          .align H'2           ; Adjust so that scb/f jump destination is an
                                ; even address
0189:erases 03:
0190:          nop                  ; Adjust so that scb/f jump destination is an
                                ; even address
0191:          ; If an odd address, insert NOP
0192:          bset.b #d'1,@FLMCR   ; Set E bit
0193:          ; Adjust so that E wait scb/f jump destination
                                ; is an even address
0194:erases 04:
0195:          nop                  ; nop to increase wait time
0196:          nop                  ;
0197:          nop                  ;
0198:          nop                  ;
0199:          nop                  ;
0200:          scb/f R4,erases 04
0201:          scb/f R3,erases 04   ; Erase time wait (5.0 ms)
0202:          bclr.b #d'1,@FLMCR  ; Clear E bit
0203:;;
0204:          mov.w #(d'10*MHZ/d'8),R3
0205:          ; Set counter for wait after clearing E
0206:erases 05:
0207:          scb/f R3,erases 05   ; Wait after clearing E (10 µs or more)
0208:          mov.w #H'A500,@TCSR  ; Stop watchdog timer
0209:;;
0210:          bclr.b #d'5,@FLMCR   ; Clear ESU bit
0211:          mov.w #(d'10*MHZ/d'8),R3
0212:          ; Set counter for wait after clearing ESU
0213:erases 06:
0214:          scb/f R3,erases 06   ; Wait after clearing ESU (10 µs or more)
0215:          clr.b @EBR1          ; Clear erase target block
0216:;;
0217:erases07:
0218:          add.w #H'01,02        ; Block bit number + 1
0219:          cmp.w #d'7,R2        ;
0220:          bls erases 01        ; If end of all blocks (0-7) end
0221:;;
0222:          ldm @sp+,(R2-R4)     ; Restore used registers

```

```

0223:          rts          ; Subroutine return
0224:;
0225:;
0226:;*****
0227:; *   erasevf   SUB          *
0228:; *           Erase-verify (SUB) *
0229:;*****
0230:          .align H'2
0231:erasevf:.equ $
0232:;
0233:          stm (R2-R5),@-sp ; Arguments R0 Erase target block/unerased block
0234:;          ; Save used registers
0235:;          ; R2 Block table ADR
0236:;          ; R3 Target block bit number
0237:;          ; R4 Verify address
0238:;          ; R5 Wait loop counts
0239:;          stc.b ep,@-sp ; Save used page register
0240:;          bset.b #d'3,@FLMCR ; Set EV bit
0241:;          mov.w #(d'20*MHZ/d'8),R5
0242:;          ; Set EV wait loop counter
0243:erasevf 01:
0244:;          scb/f R5,erasevf 01 ; EV wait (20 µs or more)
0245:;
0246:;          mov.w #RAMSTR,R2 ; Start of program in RAM
0247:;          add.w #blockadr,R2 ; + block table relative address
0248:;          sub.w #prog_start,R2 ; - transfer block start address
0249:;          ; → block table start address
0250:;          clr.w R3 ; Target block bit number = 0
0251:erasevf 02:
0252:;          bst.w R3,R0
0253:;          beq erasevf 07 ; If R3 bit of R0= B'1, execute the following
0254:;          mov.w @R2, R4
0255:;          ldc.b R4,ep ; Set block start address page in ep
0256:;          mov.w @(H'02,R2),R4 ; Set block start address in R4
0257:erasevf 03:
0258:;          mov.w #H'FFFF,@R4 ; Dummy write (address latch)
0259:;          mov.w #(d'2*MHZ/d'8),R5
0260:;          ; Set loop counter for wait after latch
0261:erasevf 04:
0262:;          scb/f R5,erasevf 04 ; Wait after latch (2 µs or more)
0263:;
0264:;          cmp.w #H'FFFF,@R4 ; Verify
0265:;          bne erasevf 06 ; If target address is unerased, end
0266:;          cmp.w @(H'04,R2),R4
0267:;          bhs erasevf 05 ; If R4 ≥ block end ADR, end
0268:;          add.w #H'02,R4 ; R4 = R4 + 2
0269:;          bra erasevf 03 ;
0270:erasevf 05:
0271:;          bclr.w R3,R0 ; Clear R0 (EBR1) erased block bit
0272:erasevf 06:
0273:;
0274:erasevf 07:
0275:;          add.w #H'0006,R2 ; Block table next line address
0276:;
0277:;          add.w #H'01,R3
0278:;          cmp.b #H'07,R3
0279:;          bls erasevf 02 ; If target block bit number > 7, end
0280:;
0281:;          bclr.b #d'3,@FLMCR ; Clear EV bit
0282:;          mov.w #(d'5*MHZ/d'8),R5
0283:;          ; Set loop counter for wait after clearing EV
0284:erasevf 08:
0285:;          scb/f R5,erasevf 08 ; Wait after clearing EV (5 µs or more)

```

```

0286;;
0287:      lbc.b @sp+,ep      ; Restore used page register
0288:      lbm @sp+,(R2-R5)   ; Restore used registers
0289:      rts                ; Subroutine return
0290;;
0291;;
0292;*****
0293; * blockadr  DATA
0294; *          Flash memory block addresses
0295;*****
0296:      .align H'2
0297:blockadr: .equ $
0298:      .data.w H'0001,H'0000,H'03FE      ;EB0
0299:      .data.w H'0001,H'0400,H'07FE      ;EB1
0300:      .data.w H'0001,H'0800,H'0BFE      ;EB2
0301:      .data.w H'0001,H'0C00,H'0FFE      ;EB3
0302:      .data.w H'0001,H'1000,H'7FFE      ;EB4
0303:      .data.w H'0001,H'8000,H'FFFE      ;EB5
0304:      .data.w H'0002,H'0000,H'7FFE      ;EB6
0305:      .data.w H'0002,H'8000,H'FFFE      ;EB7
0306;;
0307;*****
0308;;
0309:      ;
0310:      ;
0311:      ;
0312:prog_stop: .equ $          ; End of program for transfer to RAM
0313;;
0314;;
0315:      .end
0316:

```

- Notes: 1. The sample programs in this manual are provided to illustrate programming/erasing of flash memory incorporated in an MCU. They do not make provisions for various kinds of applications, and cannot be used as they are. They are intended solely for reference in program development.
2. Program operation must be confirmed in actual use.
3. These programs are preliminary samples, and are subject to change without notice for the purpose of improvement, etc.

## 20.4.7 Protection Modes

There are two modes for flash memory program/erase protection: hardware protection and software protection. These protection modes are described below.

**(1) Software Protection** With software protection, setting the P or E bit in the flash memory control register (FLMCR) does not cause a transition to program mode or erase mode.

Details of software protection are given below.

Item	Description	Function		
		Program	Erase	Verify* <sup>1</sup>
Block specification protection	Erase protection can be set for individual blocks by settings in the erase block register (EBR1). However, protection against programming is disabled. Setting EBR1 to H'00 places all blocks in the erase-protected state.	possible	Not possible	possible
Emulation protection* <sup>2</sup>	Setting the OVLPE bit in the flash memory emulation register (FLMER) enables program/erase protection for all blocks.	Not possible	Not possible* <sup>3</sup>	possible

**(2) Hardware Protection** Hardware protection refers to a state in which programming/erasing of flash memory is forcibly aborted or disabled. At this time, the flash memory control register (FLMCR) and erase block register (EBR1) settings are cleared. In the case of error protection, the P bit and E bit can be set, but a transition is not made to program mode or erase mode.

Details of the hardware protection state are given below.

Item	Description	Function		
		Program	Erase	Verify*1
FWE pin protection	When a high level is not being applied to the FWE pin, FLMCR and EBR1 are initialized, and the program/erase-protected state is entered.*4	Not possible	Not possible*3	Not possible
Reset/standby protection	In a reset (including a watchdog timer reset) and in standby mode, FLMCR and EBR1 are initialized, and the program/erase-protected state is entered. In a reset via the $\overline{\text{RES}}$ pin, the reset state is not reliably entered unless the $\overline{\text{RES}}$ pin is held low for at least 20 ms (oscillation settling time) after powering on. In the case of a reset during operation, the $\overline{\text{RES}}$ pin must be held low for a minimum of 20 system clock cycles (20 $\phi$ ).*5	Not possible	Not possible*3	Not possible
Error protection	If abnormal MCU operation is detected during flash memory programming or erasing (error occurrence: FLER = 1), error protection is enabled. FLMCR and EBR1 settings are retained but programming/erasing is aborted at the point at which the error occurred. Error protection is released only by a $\overline{\text{RES}}$ pin reset*6 and in hardware standby mode.	Not possible	Not possible*3	Possible

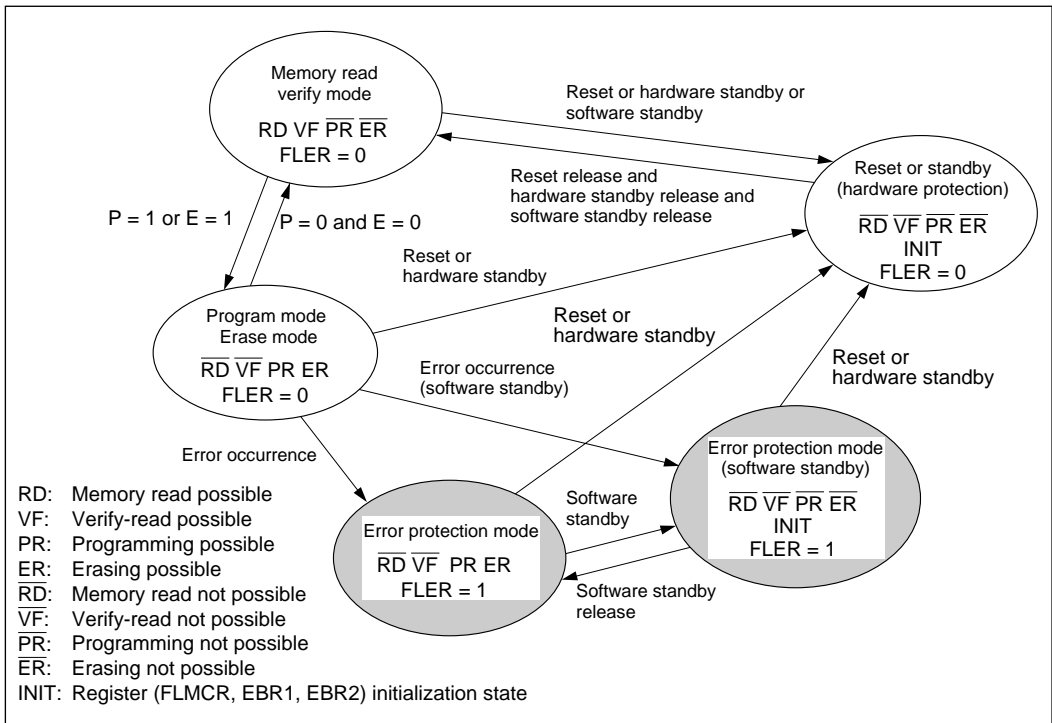
- Notes: 1. Two modes: program-verify and erase-verify.  
 2. Excluding a RAM area overlapping flash memory.  
 3. All blocks are unerasable and block-by-block specification is not possible.  
 4. For details, see section 20.7, Flash Memory Programming and Erasing Precautions.  
 5. See section 4.2.2, Reset Sequence, and section 20.7, Flash Memory Programming and Erasing Precautions. The reset period during operation is a minimum of 6 system clock cycles for the H8/538F and H8/539F, but a minimum of 20 system clock cycles for the H8/539F S-mask model.  
 6. In the H8/538F, a watchdog timer reset is included in the FLER bit clearing conditions, but only  $\overline{\text{RES}}$  pin reset input is applicable in the case of the H8/539F.

**Error Protection** A flash memory error is detected (the FLER bit is set in FLMSR\*<sup>1</sup>) when abnormal MCU operation (operation not in accordance with the program/erase algorithm) occurs during flash memory programming or erasing (when the P or E bit is set in FLMCR\*<sup>3</sup>). Flash memory then enters the error protection state (this indicates the flash memory operating status and does not affect the operation of the MCU).

In this state, FLMCR and EBR1 settings\*<sup>2</sup> are retained, but program mode or erase mode is aborted at the point at which the error occurred. Program mode or erase mode cannot be entered by re-setting the P or E bit in FLMCR. However, PV and EV bit setting is enabled, and a transition can be made to verify mode.

The error protection state is released only by a  $\overline{\text{RES}}$  pin reset and in hardware standby mode.

- Notes: 1. For details of FLER bit setting conditions, see section 20.2.4, RAM Control Register (RAMCR).  
 2. FLMCR and EBR1 can be written to. However, registers will be initialized if a transition is made to software standby mode in the error protection state.  
 3. Note that NMI input is disabled when the P or E bit is set. For details see section 20.4.9, NMI Input Disabling Conditions.



**Figure 20-11 Flash Memory State Transitions**  
 (Modes 4 and 7 (on-chip ROM enabled), high level applied to FWE pin)



The error protection mode is provided to prevent damage to the flash memory due to overprogramming or overerasing by detecting MCU runaway or abnormal operation not in accordance with the program/erase algorithm during flash memory programming or erasing,

However, the error protection function is invalid for abnormal operations other than the FLER bit setting conditions. Also, if a certain time has elapsed before this protection state is entered, damage may already have been caused to the flash memory. Consequently, this function cannot provide complete protection against damage to flash memory.

To prevent such abnormal operations, therefore, it is necessary to ensure correct operation in accordance with the program/erase algorithm, with the flash write enable (FWE) voltage applied, and to conduct constant monitoring for MCU errors, internally and externally, using the watchdog timer or other means. There may also be cases where the flash memory is in an erroneous programming or erroneous erasing state at the point of transition to this protection mode, or where programming or erasing is not properly carried out because of an abort. In cases such as these, a forced recovery (program rewrite) must be executed using boot mode. However, it may also happen that boot mode cannot be normally initiated because of overprogramming or overerasing.

#### **20.4.8 NMI Input Disabling Conditions**

NMI input is disabled when flash memory is being programmed or erased (while the P or E bit is set in FLMCR) and while the boot program is executing in boot mode (until a branch is made to the on-chip RAM area<sup>\*1</sup>), to give priority to the program or erase operation. There are three reasons for this:

1. NMI input during programming or erasing might cause a violation of the programming or erasing algorithm, with the result that normal operation could not be assured.
2. In the NMI exception handling sequence during programming or erasing, the vector would not be read correctly<sup>\*2</sup>, possibly resulting in MCU runaway.
3. If NMI input occurred during boot program execution, it would not be possible to execute the normal boot mode sequence.

For these reasons, in the H8/539F's on-board programming modes alone there are conditions for disabling NMI input, as an exception to the general rule. However, this provision does not guarantee normal erasing and programming or MCU operation. All requests, including NMI, must therefore be restricted inside and outside the MCU during FWE application. NMI input is also disabled in the error protection state and while the P or E bit remains set in FLMCR during flash memory emulation in RAM.

- Notes: 1. This is the interval until a branch is made to the boot program area (H'EE80—H'F37F) in the on-chip RAM. (This branch takes place immediately after transfer of the user program is completed.) Consequently, after the branch to the RAM area, NMI input is enabled except during programming and erasing. Interrupt requests must therefore be disabled inside and outside the MCU until the user program has completed initial programming (including the vector table and the NMI interrupt handling routine).
2. The vector may not be read correctly in this case for the following two reasons:
- a. If flash memory is read while being programmed or erased (while the P or E bit is set in FLMCR), correct read data will not be obtained (undetermined values will be returned).
  - b. If the NMI entry in the vector table has not been programmed yet, NMI exception handling will not be executed correctly.

## 20.5 Flash Memory Emulation in RAM

As flash memory programming and erasing takes time, it may be difficult to carry out tuning by writing parameters and other data in real time. In this case, real-time programming of flash memory can be emulated by overlapping an actual RAM area onto a small block area in flash memory (H'0000—H'0FFF).

This RAM area overlapping is executed by means of bits 7, 3, and 2 in the flash memory emulation register (FLMER) and bits 2 and 1 in the RAM control register (RAMCR). After RAM overlapping, access is possible both from the area overlapped onto flash memory (mapping RAM area) and from the original RAM area (actual RAM area).

Bits 7, 3, and 2 in FLMER and bits 2 and 1 in RAMCR are valid in modes 2, 4, and 7. Flash memory programming must not be performed in mode 2. In other modes, a read will always return 0 and RAM area overlapping cannot be carried out. When the flash memory emulation function is used, RAMCR bits 7 and 5 should both be set (RAME1 = 1 and RAME2 = 1). The mapping RAM area setting method is shown in tables 20-9 and 20-10.

### Flash Memory Emulation Register (FLMER)

Bit	7	6	5	4	3	2	1	0
	OVLPE	—	—	—	A11E	A10E	—	—
Initial value	0	1	1	1	0	0	1	1
R/W	R/W	—	—	—	R/W	R/W	—	—

**Table 20-9 ROM Area Setting**

ROM Area (Mapping RAM Area)	FLMER Register Bit 7*1	RAMCR Register		Overlap Function	Program /Erase Protection
	OVLPE	Bit 2*1	Bit 1*1		
		RAM2	RAM1		
—	0	0/1	0/1	Disabled	Disabled
H'0000-H'03FF	1	0	0	Enabled	Enabled
H'0400-H'07FF	1	0	1	Enabled	Enabled
H'0800-H'0BFF	1	1	0	Enabled	Enabled
H'0C00-H'0FFF	1	1	1	Enabled	Enabled

**Table 20-10 RAM Area\*2 Setting**

RAM Area*2 (Mapping RAM Area)	FLMER Register		RAMCR Register	
	Bit 3*1	Bit 2*1	Bit 7*1	Bit 5*1
	A11E	A10E	RAME1	RAME2
H'F000-H'F3FF (1024 byts)	0	0	1	1
H'F400-H'F7FF (1024 byts)	0	1	1	1
H'F800-H'FBFF (1024 byts)	1	0	1	1
Use prohibited *3	1	1	1	1
Use prohibited *4	*	*	0	0
Use prohibited *4	*	*	0	1
Use prohibited *4	*	*	1	0

Notes: 1. Bits 7, 3, and 2 of the flash memory emulation register (FLMER) and bits 2 and 1 of the RAM control register (RAMCR) can be written to in modes 2, 4, and 7.

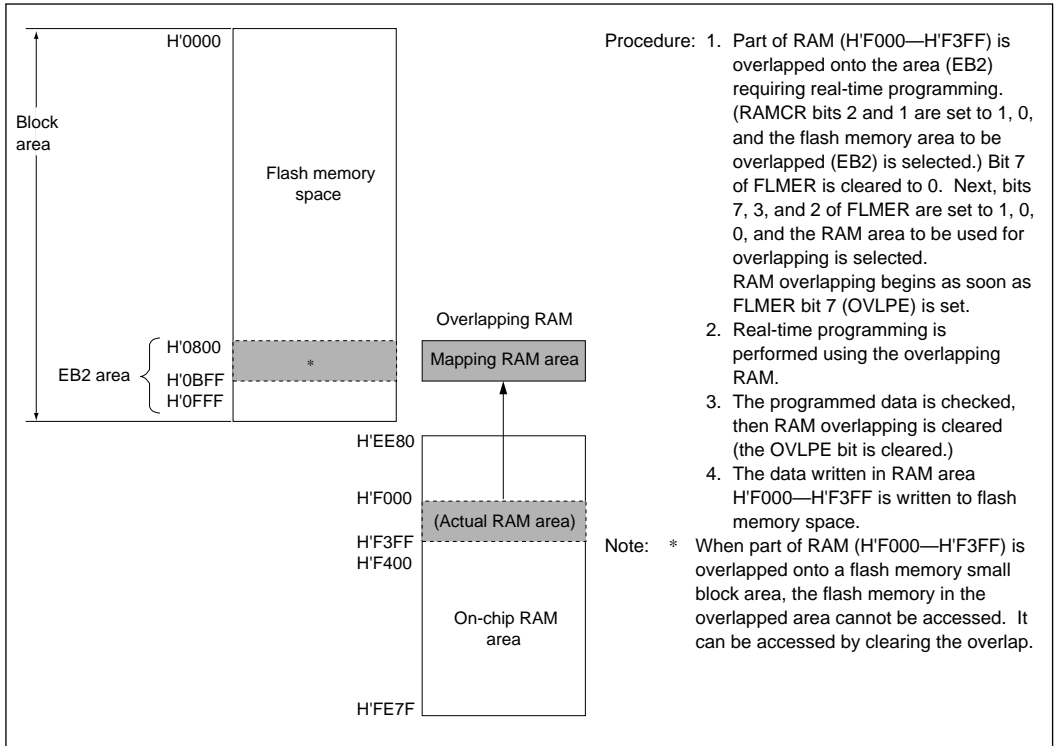
(In the H8/538F it was necessary to apply 12 V as the program voltage  $V_{pp}$  when performing RAM emulation, but in the H8/539F RAM emulation can be performed regardless of the  $V_{pp}$  voltage. The H8/539F S-mask model uses a single 5 V power source, and does not have a  $V_{pp}$  pin. As with the regular H8/539F, the H8/539F S-mask model can perform RAM emulation in modes 2, 4, and 7 (on-chip flash memory enabled).)

Flash memory programming must not performed in mode 2.

2. RAM area overlapped onto flash memory
3. Use prohibited when A11E and A10E are both set to 1. (When A11E and A10E are both set to 1, the state is the same as when A11E is 1 and A10E is 0.)
4. Use prohibited when RAME1 = 0 or RAME2 = 0. (Can only be used when RAME1 = RAME2 = 1.)

## Example of Emulation of Real-Time Flash Memory Programming

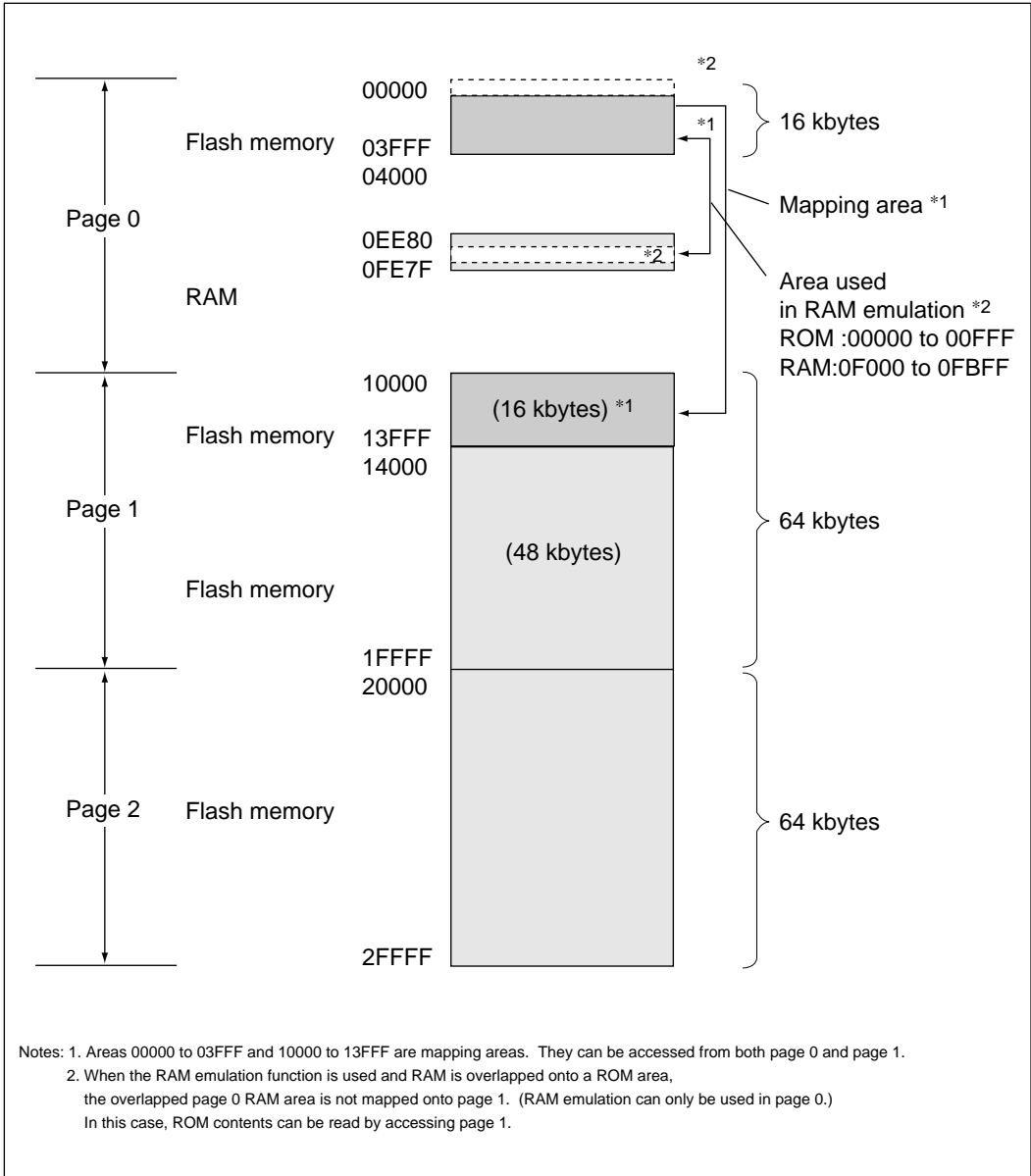
In the following example, RAM area H'F000 to H'F3FF is overlapped onto flash memory area EB2 (H'0800 to H'0BFF).



**Figure 20-12 Example of RAM Overlap Operation**

## Notes on Use of Emulation in RAM

### (1) Notes on emulation RAM access



**Figure 20-13 Notes on Emulation RAM Access**

## **(2) Flash write enable (FWE) application and releasing**

As in on-board program mode, care is required when applying and releasing FWE to prevent erroneous programming or erasing. To prevent erroneous programming and erasing due to program runaway during FWE application, in particular, the watchdog timer should be set when the P or E bit is set in the flash memory control register (FLMCR), even while the emulation function is being used. For details, see section 20.7, Flash Memory Programming and Erasing Precautions.

## **(3) NMI input disabling conditions**

When the emulation function is used, NMI input is disabled when the P bit or E bit is set to 1 in FLMCR, in the same way as with normal programming and erasing.

The P and E bits are cleared by a reset (including a watchdog timer reset), in standby mode, when a high level is not being input to the FWE pin, or when the SWE bit in FLMCR is 0 while a high level is being applied to the FWE pin.

## 20.6 PROM Mode

### 20.6.1 PROM Mode Setting

The H8/539F S-mask model, in which the on-chip ROM is flash memory, has a PROM mode as well as the on-board programming modes for programming and erasing flash memory. In PROM mode, the on-chip ROM can be freely programmed using a PROM programmer that supports Hitachi microcomputer device types with 128-kbyte on-chip flash memory.

PROM mode requires the use of the socket adapter shown in table 20-11. For notes on the use of PROM mode, see section 20.6.9, Notes on Memory Programming, and section 20.7, Flash Memory Programming and Erasing Precautions.

### 20.6.2 Socket Adapter and Memory Map

For program writing and verification, a special-purpose 112-to-32-pin adapter is mounted on the PROM programmer. The socket adapter product code is given in table 20-11.

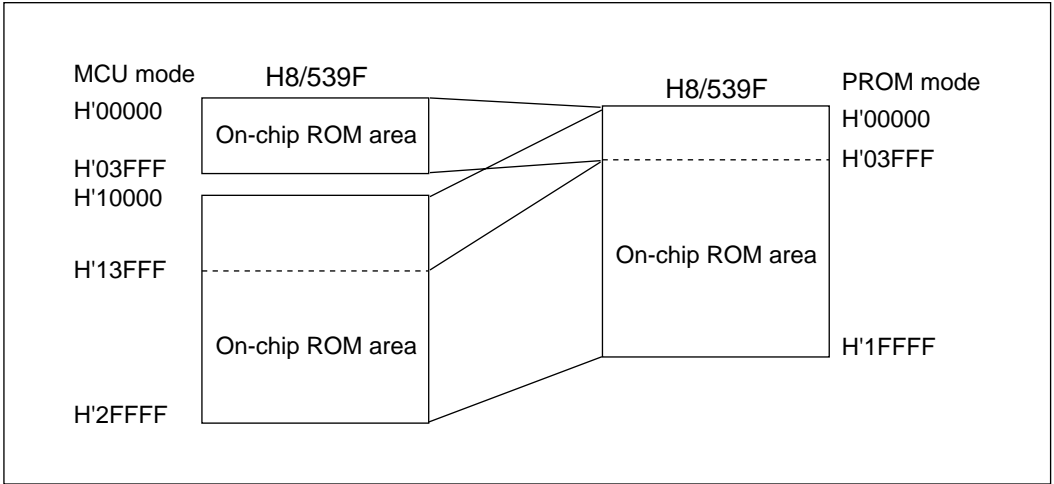
Figure 20-15 shows the memory map in PROM mode.

**Table 20-11 Socket Adapter Product Code**

<b>Product Code</b>	<b>Package</b>	<b>Socket Adapter Product Code</b>	<b>Device Type</b>
HD64F5389FS	112-pin plastic QFP (FP-112)	HS538FESH01H*	Hitachi microcomputer with 128-kbyte on-chip flash memory

Note: \* The same socket adapter is used for the H8/539F dual power source model, but the Hitachi 128-kbyte on-chip flash memory microcomputer device type must be selected. Selecting HN28F101 may cause permanent damage to the chip.





**Figure 20-14 Memory Map in PROM Mode**

Note: \* An appropriate tool should be used to insert the adapter into, and remove it from, the IC socket. A sample tool is shown in table 20-12.

**Table 20-12 Sample Tool**

<b>Manufacturer</b>	<b>Model</b>
ENPLAS Corporation	HP-100 (vacuum pen)

### 20.6.3 PROM Mode Operation

The PROM mode program/erase/verify specifications are the Hitachi 128-kbyte on-chip flash memory microcomputer device type specifications.

Table 20-13 shows how the different operating modes are set when using PROM mode, and table 20-14 lists the commands used in PROM mode. Details of each mode are given below.

#### Memory Read Mode

Memory read mode supports byte reads.

#### Auto-Program Mode

Auto-program mode supports programming of 128 bytes at a time. Status polling is used to confirm the end of auto-programming.

#### Auto-Erase Mode

Auto-erase mode supports automatic erasing of the entire flash memory. Status polling is used to confirm the end of auto-erasing.

## Status Read Mode

Status polling is used for auto-programming and auto-erasing, and normal termination can be confirmed by reading the I/O6 signal. In status read mode, error information is output if an error occurs.

**Table 20-13 Settings for Operating Modes In PROM Mode**

Mode	Pin Names					
	FWE	$\overline{\text{CE}}$	$\overline{\text{OE}}$	$\overline{\text{WE}}$	D <sub>0</sub> to D <sub>7</sub>	A <sub>0</sub> to A <sub>17</sub>
Read	V <sub>CC</sub> or 0	L	L	H	Data output	Ain
Output disable	V <sub>CC</sub> or 0	L	H	H	Hiz	X
Command write	V <sub>CC</sub> or 0	L	H	L	Data input	*Ain
Chip disable	V <sub>CC</sub> or 0	H	X	X	Hiz	X

### Legend

L: Low level  
H: High level  
X: Undefined  
Hi-z: High impedance

- Notes:
1. Chip disable is not a standby state; internally, it is an operation state.
  2. \*Ain indicates that there is also address input in auto-program mode.
  3. For the command write when making a transition to auto-program or auto-erase mode, set the FWE pin to V<sub>CC</sub> (V).

**Table 20-14 PROM Mode Commands**

Command	Cycles	1st Cycle			2nd Cycle		
		Mode	Address	Data	Mode	Address	Data
Memory read mode	1	write	X	H'00	read	RA	Dout
Auto-program mode	129	write	X	H'40	write	WA	Din
Auto-erase mode	2	write	X	H'20	write	X	H'20
Status read mode	2	write	X	H'71	write	X	H'71

RA: Read address  
Din: Program data  
WA: Program address  
Dout: Read data

- Notes:
1. In auto-program mode, 129 cycles are required for command writing by a simultaneous 128-byte write.

**Table 20-15 DC Characteristics in Memory Read Mode****- Preliminary -**(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item		Symbol	Min	Typ	Max	Unit	Test Conditions
Input high voltage	$O_7$ to $O_0$ , $A_{16}$ to $A_0$	$V_{IH}$	2.2	—	$V_{CC} + 0.3$	V	
Input low voltage	$O_7$ to $O_0$ , $A_{16}$ to $A_0$	$V_{IL}$	-0.3	—	0.8	V	
Schmitt trigger input voltage	$\overline{OE}$ , $\overline{CE}$ , $\overline{WE}$	$V_{T-}$	1.0	—	2.5	V	
		$V_{T+}$	2.0	—	3.5	V	
		$V_{T-}-V_{T+}$	0.4	—	—	V	
Output high voltage	$O_7$ to $O_0$	$V_{OH}$	2.4	—	—	V	$I_{OH} = -200\ \mu\text{A}$
Output low voltage	$O_7$ to $O_0$	$V_{OL}$	—	—	0.45	V	$I_{OL} = 1.6\ \text{mA}$
Input leakage current	$O_7$ to $O_0$ , $A_{16}$ to $A_0$	$ I_{LI} $	—	—	2	$\mu\text{A}$	$V_{in} = 0$ to $V_{CC}$ V
$V_{CC}$ current	Reading	$I_{CC}$	—	40	65	mA	
	Programming	$I_{CC}$	—	50	85	mA	
	Erasing	$I_{CC}$	—	50	85	mA	

Note: For the absolute maximum ratings, see section 23.1, Absolute Maximum Ratings.  
Exceeding the absolute maximum ratings may cause permanent damage to the chip.

## 20.6.4 Memory Read Mode

- Preliminary -

Table 20-16 shows the AC characteristics in a transition to memory read mode.

Figure 20-16 shows the timing waveforms in a transition to memory read mode.

Table 20-17 shows the AC characteristics in a memory contents read.

Figures 20-17 and 20-18 show the timing waveforms in a memory contents read.

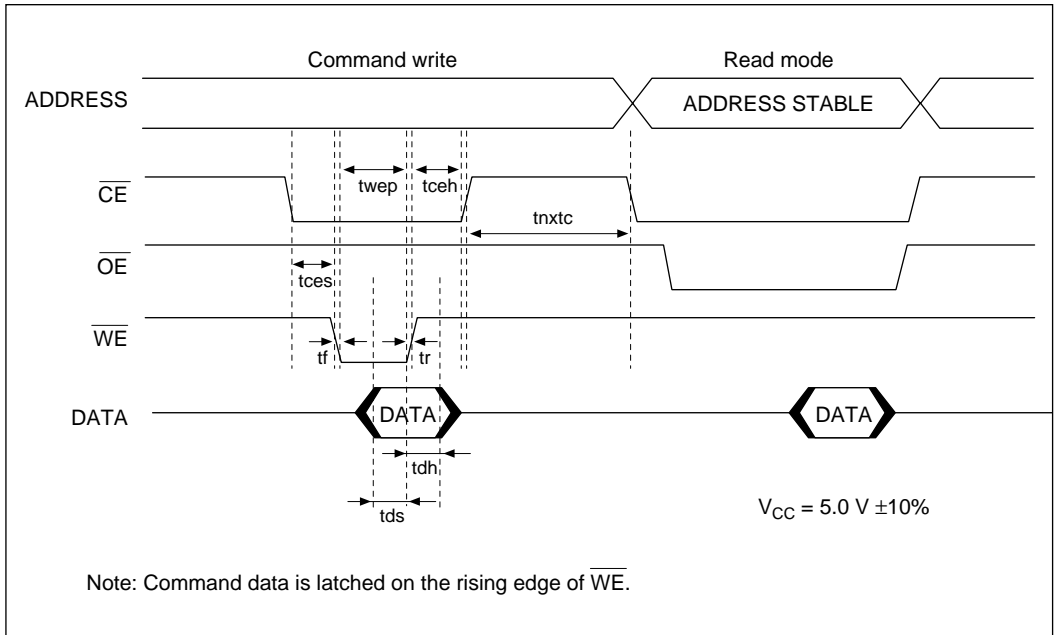
Table 20-18 shows the AC characteristics in a transition from memory read mode to another mode.

Figure 20-19 shows the timing waveforms in a transition from memory read mode to another mode.

**Table 20-16 AC Characteristics in Memory Read Mode Transition**

(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item	Symbol	Min	Max	Unit	Notes
Command write cycle	tnxtc	20		$\mu\text{s}$	
$\overline{\text{CE}}$ hold time	tceh	0		ns	
$\overline{\text{CE}}$ setup time	tces	0		ns	
Data hold time	tdh	50		ns	
Data setup time	tds	50		ns	
Write pulse width	twep	70		ns	
$\overline{\text{WE}}$ rise time	tr		30	ns	
$\overline{\text{WE}}$ fall time	tf		30	ns	



**Figure 20-15 Timing Waveforms for Memory Read Mode Transition**

**Table 20-17 AC Characteristics in Memory Contents Read**

**- Preliminary -**

(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item	Symbol	Min	Max	Unit	Notes
Access time	$t_{acc}$		20	$\mu\text{s}$	
$\overline{CE}$ output delay time	$t_{ce}$		150	ns	
$\overline{OE}$ output delay time	$t_{oe}$		150	ns	
Output disable delay time	$t_{df}$		100	ns	
Data output hold time	$t_{oh}$	5		ns	

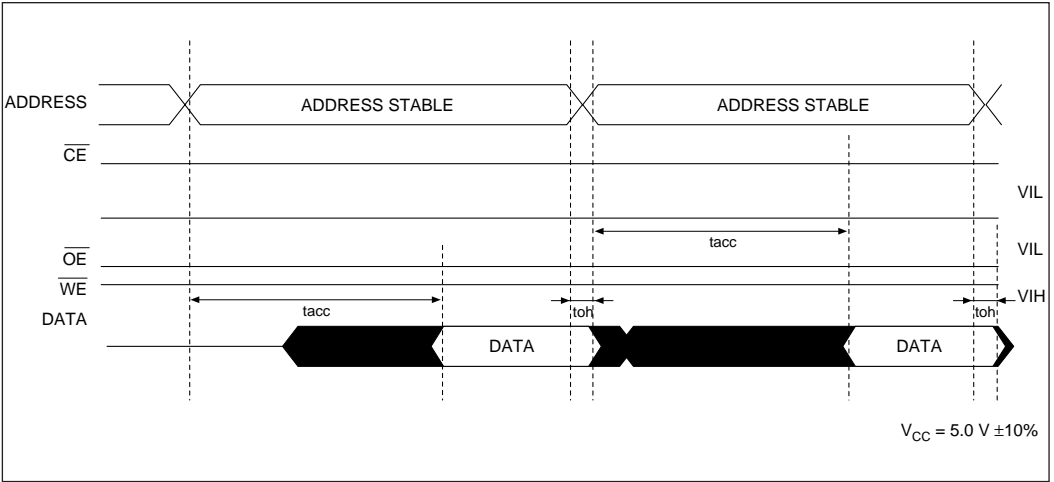


Figure 20-16  $\overline{CE}$ ,  $\overline{OE}$  Enable State Read

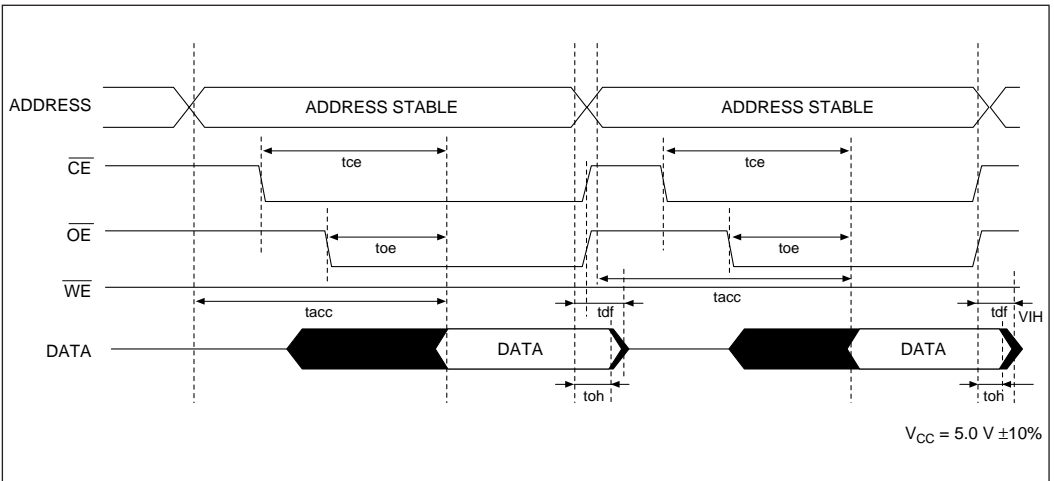
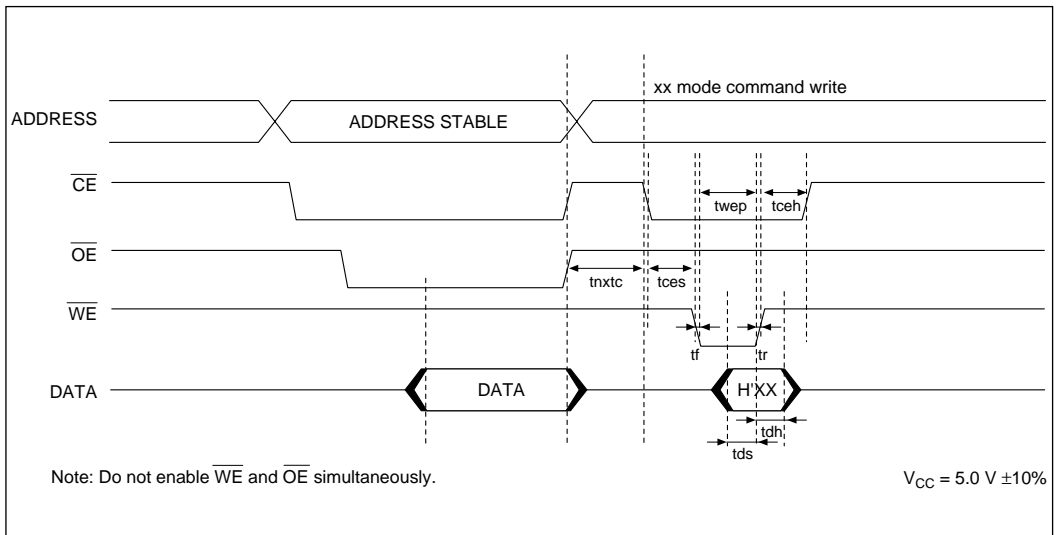


Figure 20-17  $\overline{CE}$ ,  $\overline{OE}$  Clocked Read

**Table 20-18 AC Characteristics in Transition from Memory Read Mode to Another Mode**  
**- Preliminary -**

(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item	Symbol	Min	Max	Unit	Notes
Command write cycle	tnxtc	20		$\mu\text{s}$	
$\overline{\text{CE}}$ hold time	tceh	0		ns	
$\overline{\text{CE}}$ setup time	tces	0		ns	
Data hold time	tdh	50		ns	
Data setup time	tds	50		ns	
Write pulse width	twep	70		ns	
$\overline{\text{WE}}$ rise time	tr		30	ns	
$\overline{\text{WE}}$ fall time	tf		30	ns	



**Figure 20-18 Transition from Memory Read Mode to Another Mode**

## 20.6.5 Auto-Program Mode

- Preliminary -

Table 20-19 shows the AC characteristics in auto-program mode.

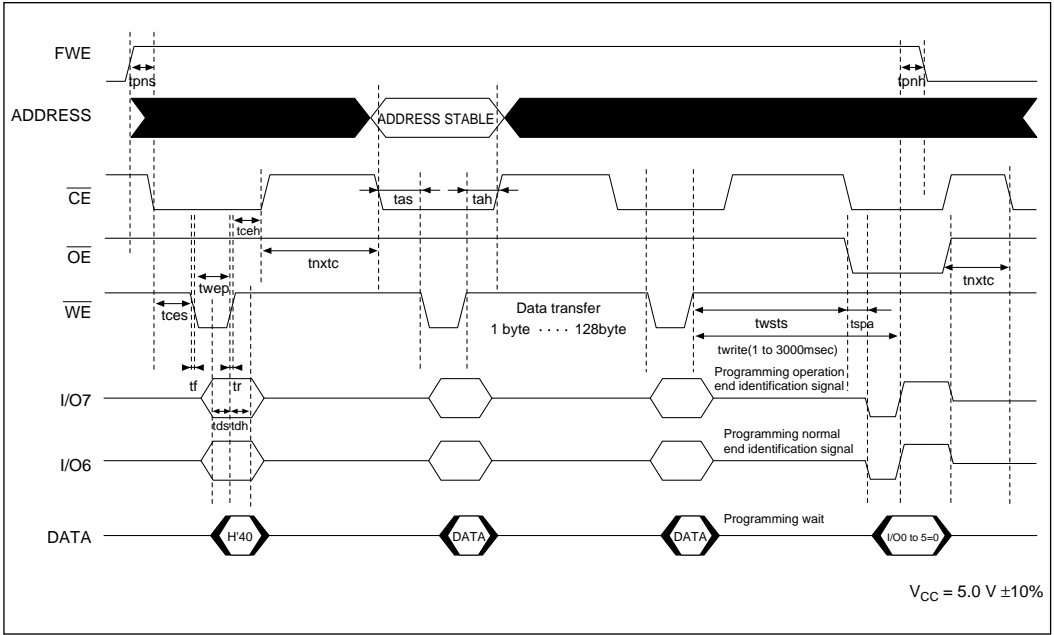
Figure 20-20 shows the timing waveforms in auto-program mode

**Table 20-19 AC Characteristics in Auto-Program Mode**

(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item	Symbol	Min	Max	Unit	Notes
Command write cycle	tnxtc	20		$\mu\text{s}$	
$\overline{\text{CE}}$ hold time	tceh	0		ns	
$\overline{\text{CE}}$ setup time	tces	0		ns	
Data hold time	tdh	50		ns	
Data setup time	tds	50		ns	
Write pulse width	twep	70		ns	
Status polling start time	twsts	1		ms	
Status polling access time	tspa		150	ns	
Address setup time	tas	0		ns	
Address hold time	tah	60		ns	
Memory write time	twrite	1	3000	ms	
$\overline{\text{WE}}$ rise time	tr		30	ns	
$\overline{\text{WE}}$ fall time	tf		30	ns	
Write setup time	tpns	100		ns	
Write end setup time	tpnh	100		ns	





**Figure 20-19 Auto-Program Mode Timing Waveforms**

### Notes on Use of Auto-Program Mode

- (1) In auto-program mode, 128 bytes are programmed simultaneously. This should be carried out by executing 128 consecutive byte transfers.
- (2) A 128-byte data transfer is necessary even when programming fewer than 128 bytes. In this case, H'FF data must be written to the extra addresses.
- (3) If a value other than an effective address is input, processing will switch to a memory write operation but a write error will be flagged.
- (4) Memory address transfer is performed in the second cycle (figure 20-20). Do not perform transfer after the second cycle.
- (5) Do not perform a command write during a programming operation.
- (6) Perform one auto-programming operation for a 128-byte block for each address. Characteristics are not guaranteed for two or more programming operations.
- (7) Confirm normal end of auto-programming by checking I/O6. Alternatively, status read mode can also be used for this purpose.

## 20.6.6 Auto-Erase Mode

- Preliminary -

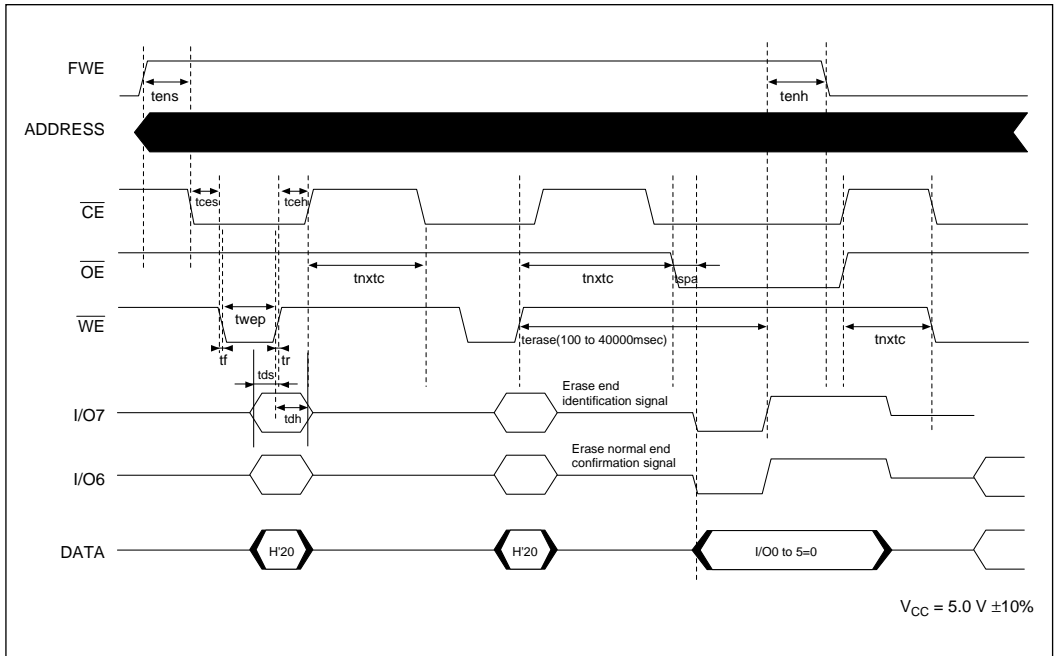
Table 20-20 shows the AC characteristics in auto-erase mode.

Figure 20-21 shows the timing waveforms in auto-erase mode.

**Table 20-20 AC Characteristics in Auto-Erase Mode**

(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item	Symbol	Min	Max	Unit	Notes
Command write cycle	tnxtc	20		$\mu\text{s}$	
$\overline{\text{CE}}$ hold time	tceh	0		ns	
$\overline{\text{CE}}$ setup time	tces	0		ns	
Data hold time	tdh	50		ns	
Data setup time	tds	50		ns	
Write pulse width	twep	70		ns	
Status polling start time	tests	1		ms	
Status polling access time	tspa		150	ns	
Memory write time	terase	100	40000	ms	
$\overline{\text{WE}}$ rise time	tr		30	ns	
$\overline{\text{WE}}$ fall time	tf		30	ns	
Write setup time	tens	100		ns	
Write end setup time	tenh	100		ns	



**Figure 20-20 Auto-Erase Mode Timing Waveforms**

### Notes on Use of Auto-Erase Mode

- (1) Auto-erase mode supports only total memory erasing.
- (2) Do not perform a command write during auto-erasing.
- (3) Confirm normal end of auto-erasing by checking I/O6. Alternatively, status read mode can also be used for this purpose.

### 20.6.7 Status Read Mode

**- Preliminary -**

Table 20-21 shows the AC characteristics in status read mode.

Figure 20-22 shows the timing waveforms in status read mode.

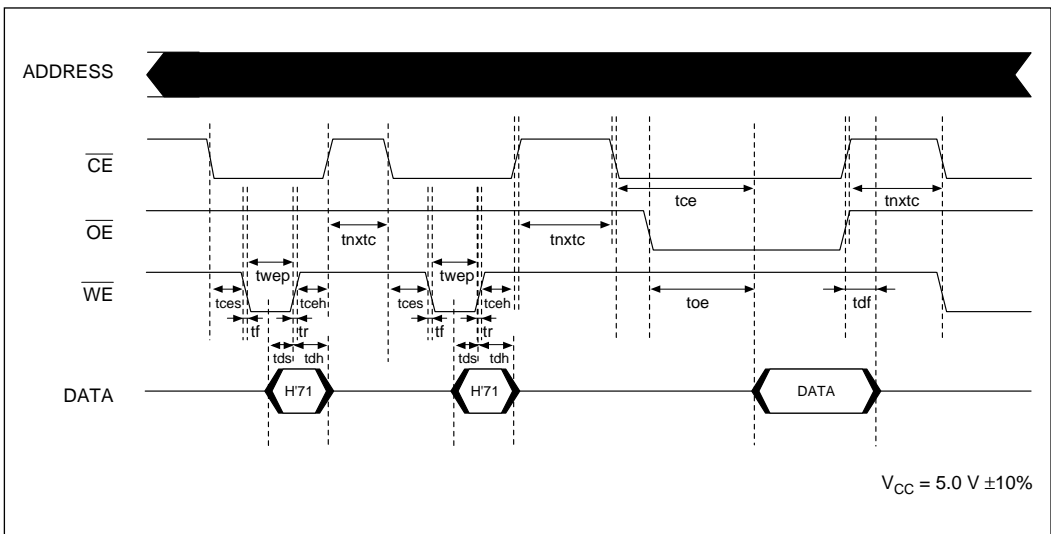
Table 20-22 shows status read mode return codes.

**Table 20-21 AC Characteristics in Status Read Mode**

- Preliminary -

(Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$ ,  $T_a = 25^\circ\text{C} \pm 5^\circ\text{C}$ )

Item	Symbol	Min	Max	Unit	Notes
Command write cycle	tnxtc	20		$\mu\text{s}$	
$\overline{\text{CE}}$ hold time	tceh	0		ns	
$\overline{\text{CE}}$ setup time	tces	0		ns	
Data hold time	tdh	50		ns	
Data setup time	tds	50		ns	
Write pulse width	twep	70		ns	
$\text{OE}$ output delay time	toe		150	ns	
Disable delay time	tdf		100	ns	
$\overline{\text{CE}}$ output delay time	tce		150	ns	
$\overline{\text{WE}}$ rise time	tr		30	ns	
$\overline{\text{WE}}$ fall time	tf		30	ns	



**Figure 20-21 Status Read Mode Timing Waveforms**

**Table 20-22 Status Read Mode Return Codes**

Pin Name	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0
Attribute	Normal end identification	Command error	Programming error	Erase error	—	—	Programming or erase count exceeded	Effective address error
Initial value	0	0	0	0	0	0	0	0
Indications	Normal end: 0 Abnormal end: 1	Command error: 1 Otherwise: 0	Programming error: 1 Otherwise: 0	Erase error: 1 Otherwise: 0	—	—	Count exceeded: 1 Otherwise: 0	Effective address error: 1 Otherwise: 0

**Note on Use of Status Read Mode**

After exiting auto-program mode or auto-erase mode, status read mode must be executed without dropping the power supply.

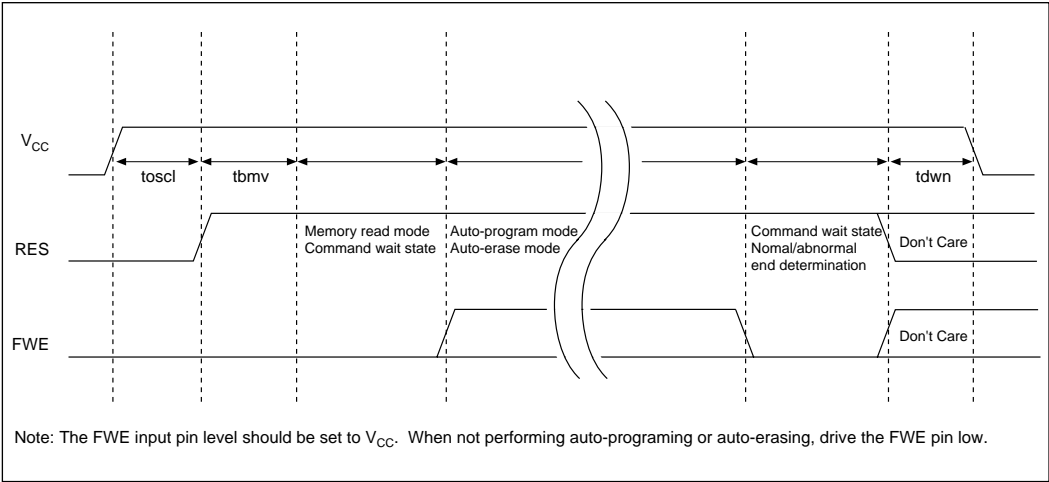
The return code is undefined immediately after powering on, or if the power supply is dropped.

**20.6.8 PROM Mode Transition Time**

Commands cannot be accepted during the oscillation settling period or the PROM mode setup period. A transition is made to memory read mode after the PROM mode setup time.

**Table 20-23 Stipulated Transition Times to Command Wait State**

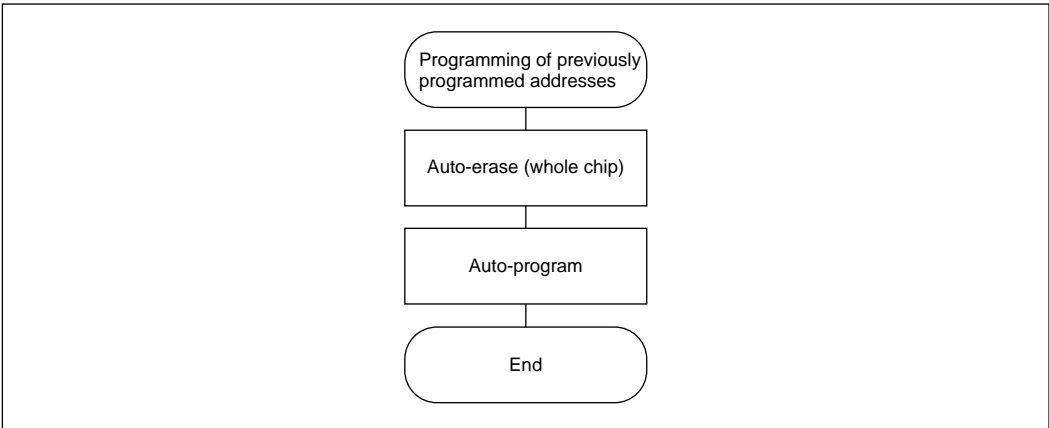
Item	Symbol	Min	Max	Unit	Notes
Standby release (oscillation setting time)	toscl	20		msec	
PROM mode setup time	tbmV	10		msec	
$V_{CC}$ hold time	tdwn	0		msec	



**Figure 20-22 Oscillation Settling Time and Boot Program Transfer Time**

### 20.6.9 Notes on Memory Programming

- (1) When programming addresses which have previously been programmed, carry out auto-erasing before auto-programming.



**Figure 20-23 Programming of Previously Programmed Addresses**

(2) When performing programming using a PROM programmer on a chip that has been programmed/erased in an on-board programming mode, auto-erasing is recommended before carrying out auto-programming.

- Notes:
1. The memory is initially in the erased state when the device is shipped by Hitachi. For other chips for which the erasure history is unknown, it is recommended that auto-erasing be executed to check and supplement the initialization (erase) level.
  2. In PROM mode, auto-programming should be performed once only on a 128-byte programming unit block. Additional programming cannot be performed on a previously programmed 128-byte programming unit block. Programming should be carried out using auto-programming after an auto-erase.

## 20.7 Flash Memory Programming and Erasing Precautions

Precautions concerning the use of on-board programming mode, the RAM emulation function, and PROM mode are summarized below.

**(1) Use the specified voltages and timing for programming and erasing.** Applied voltages in excess of the rating can permanently damage the device. Note, in particular, that the maximum ratings for the FWE pin and the  $V_{PP}$  and  $MD_2$  pins are different for the H8/539F S-mask model (single power source) and the H8/539F (dual power source).

Use a PROM programmer that supports the Hitachi microcomputer device type with 128-kbyte on-chip flash memory.

Do not select the HN28F101 setting for the PROM programmer. An incorrect setting will result in application of 12.0 V to the FWE pin, damaging the device.

**(2) Powering on and off (see figures 20-25 to 20-27)** Do not apply a high level to the FWE pin until  $V_{CC}$  has stabilized. Also, drive the FWE pin low before turning off  $V_{CC}$ .

When applying or disconnecting  $V_{CC}$  power, fix the FWE pin low and place the flash memory in the hardware protection state.

The power-on and power-off timing requirements should also be satisfied in the event of a power failure and subsequent recovery. Failure to do so may result in overprogramming or overerasing due to MCU runaway, and loss of normal memory cell operation.

**(3) FWE application/disconnection (see figures 20-25 to 20-27)** FWE application should be carried out when MCU operation is in a stable condition. If MCU operation is not stable, fix the FWE pin low and set the protection state.

The following points must be observed concerning FWE application and disconnection to prevent unintentional programming or erasing of flash memory:

- Apply FWE when the  $V_{CC}$  voltage has stabilized within its rated voltage range. If FWE is applied when the MCU's  $V_{CC}$  power supply is not within its rated voltage range ( $V_{CC} = 4.5\text{ V to }5.5\text{ V}$ ), MCU operation will be unstable and flash memory may be erroneously programmed or erased.
- Apply FWE when oscillation has stabilized (after the elapse of the oscillation settling time). When  $V_{CC}$  power is turned on, hold the  $\overline{RES}$  pin low for the duration of the oscillation settling time ( $t_{OSC1} = 20\text{ ms}$ ) before applying FWE. Do not apply FWE when oscillation has stopped or is unstable.



- In boot mode, apply and disconnect FWE during a reset.  
In a transition to boot mode, FWE = 1 input and MD<sub>2</sub> to MD<sub>0</sub> setting should be performed while the  $\overline{\text{RES}}$  input is low. FWE and MD<sub>2</sub> to MD<sub>0</sub> pin input must satisfy the mode programming setup time (tMDS) with respect to the reset release timing. When making a transition from boot mode to another mode, also, a mode programming setup time is necessary with respect to the  $\overline{\text{RES}}$  release timing.  
In a reset during operation, the  $\overline{\text{RES}}$  pin must be held low for a minimum of 20 system clock cycles (20  $\phi$ ).
- In user program mode, FWE can be switched between high and low level regardless of  $\overline{\text{RES}}$  input.  
FWE input can also be switched during execution of a program in flash memory.
- Do not apply FWE if program runaway has occurred.  
During FWE application, the program execution state must be monitored using the watchdog timer or some other means.
- Disconnect FWE only when the SWE, ESU, PSU, EV, PV, E, and P bits in FLMCR are cleared.  
Make sure that the SWE, ESU, PSU, EV, PV, E, and P bits are not set by mistake when applying or disconnecting FWE.

**(4) Do not apply a constant high level to the FWE pin.** To prevent erroneous programming or erasing due to program runaway, etc., apply a high level to the FWE pin only when programming or erasing flash memory (including execution of flash memory emulation using RAM). A system configuration in which a high level is constantly applied to the FWE pin should be avoided. Also, while a high level is applied to the FWE pin, the watchdog timer should be activated to prevent overprogramming or overerasing due to program runaway, etc.

**(5) Use the recommended algorithm when programming and erasing flash memory.** The recommended algorithm enables programming and erasing to be carried out without subjecting the device to voltage stress or sacrificing program data reliability. When setting the program (P) or erase (E) bit in FLMCR, the watchdog timer should be set beforehand as a precaution against program runaway, etc.

**(6) Do not set or clear the SWE bit during execution of a program in flash memory.** Clear the SWE bit before executing a program or reading data in flash memory. When the SWE bit is set, data in flash memory can be rewritten, but flash memory should only be accessed for verify operations (verification during programming/erasing).

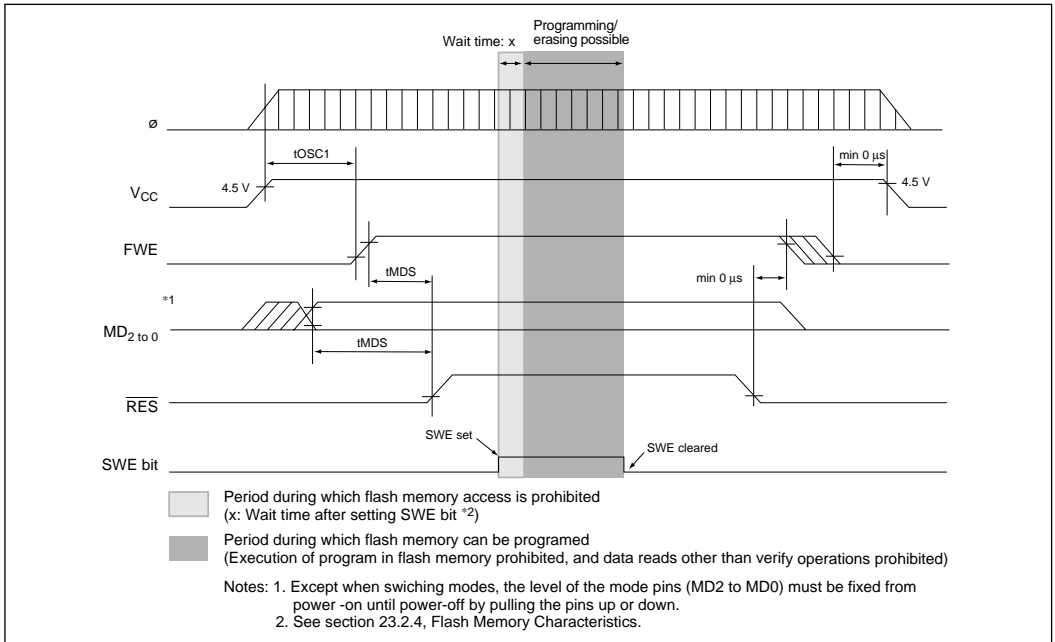
Similarly, when using the RAM emulation function while a high level is being input to the FWE pin, the SWE bit must be cleared before executing a program or reading data in flash memory. However, the RAM area overlapping flash memory space can be read and written to regardless of whether the SWE bit is set or cleared.

**(7) Do not use interrupts while flash memory is being programmed or erased.** All interrupt requests, including NMI, should be disabled during FWE application to give priority to program/erase operations (including emulation in RAM).

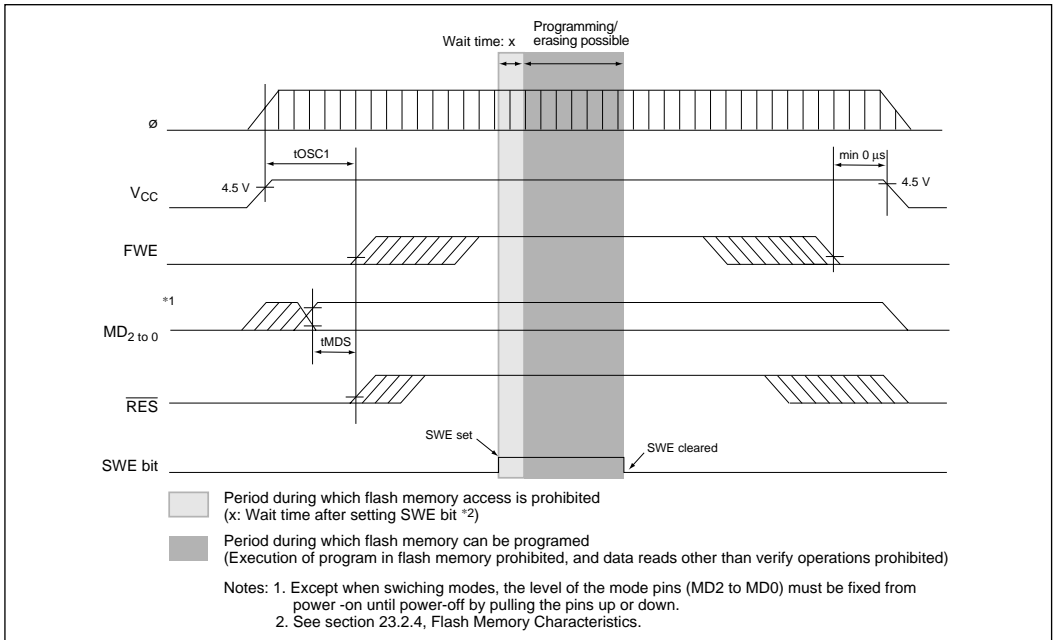
**(8) Do not perform additional programming. Erase the memory before reprogramming.** In on-board programming, perform only one programming operation on a 32-byte programming unit block. In PROM mode, too, perform only one programming operation on a 128-byte programming unit block. Programming should be carried out with the entire programming unit block erased.

**(9) Before programming, check that the chip is correctly mounted in the PROM programmer.** Overcurrent damage to the device can result if the index marks on the PROM programmer socket, socket adapter, and chip are not correctly aligned.

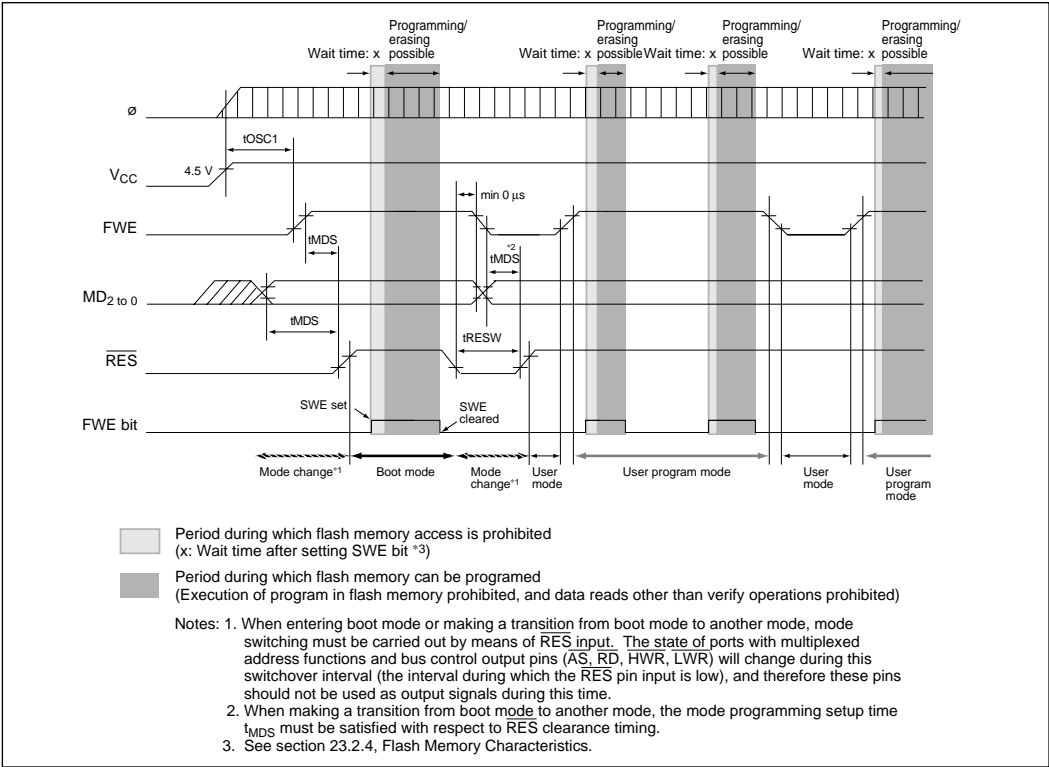
**(10) Do not touch the socket adapter or chip during programming.** Touching either of these can cause contact faults and write errors.



**Figure 20-24 Power-On/Off Timing (Boot Mode)**



**Figure 20-25 Power-On/Off Timing (User Program Mode)**



**Figure 20-26 Mode Transition Timing**  
**(Example: Boot Mode => User Mode <=> User Program Mode)**

# Section 21 Power-Down State

## 21.1 Overview

The H8/539F has a power-down state that greatly reduces power consumption by halting CPU functions. The power-down state includes three modes: sleep mode, software standby mode, and hardware standby mode. Table 21-1 indicates the methods of entering and exiting the power-down modes.

**Table 19-1 Power-Down Mode Transition Conditions**

Mode	Entering Procedure	State						Exiting Methods
		Clock	CPU	CPU Registers	Peripheral Functions	RAM	I/O Ports	
Sleep mode	Execute SLEEP instruction	Active	Halted	Held	Active	Held	Held	<ul style="list-style-type: none"> <li>Interrupt</li> <li>RES</li> <li>STBY</li> </ul>
Software standby mode	Set SSBY bit in SBYCR to 1, then execute SLEEP instruction	Halted	Halted	Held	Halted and initialized	Held	Held	<ul style="list-style-type: none"> <li>NMI</li> <li>RES</li> <li>STBY</li> </ul>
Hardware standby mode	Low input at STBY pin	Halted	Halted	Not held	Halted	Held	High impedance	<ul style="list-style-type: none"> <li>STBY &amp; RES</li> </ul>

### Legend

SBYCR: Software standby control register  
 SSBY: Software standby bit

## 21.2 Sleep Mode

This section describes sleep mode.

### 21.2.1 Transition to Sleep Mode

Execution of the SLEEP instruction causes a transition from the program execution state to sleep mode. Immediately after executing the SLEEP instruction the H8/500 CPU halts, but the contents of its internal registers remain unchanged. The on-chip peripheral modules do not halt in sleep mode.

### 21.2.2 Exit from Sleep Mode

The chip exits sleep mode when it receives an interrupt request, or a low input at the  $\overline{\text{RES}}$  or  $\overline{\text{STBY}}$  pin.

- (1) **Exit by Interrupt:** An interrupt terminates sleep mode and starts the interrupt-handling routine or data transfer controller (DTC). The chip does not exit sleep mode if the interrupt priority level is equal to or less than the level set in the H8/500 CPU's status register (SR), or if the interrupt is disabled in an on-chip peripheral module.
- (2) **Exit by  $\overline{\text{RES}}$  Input:** When the  $\overline{\text{RES}}$  signal goes low, the chip exits from sleep mode to the reset state.
- (3) **Exit by  $\overline{\text{STBY}}$  Input:** When the  $\overline{\text{STBY}}$  signal goes low, the chip exits from sleep mode to hardware standby mode.

## 21.3 Software Standby Mode

This section describes software standby mode.

### 21.3.1 Transition to Software Standby Mode

If software sets the standby bit (SSBY) to 1 in the software standby control register (SBYCR), then executes the SLEEP instruction, the chip enters software standby mode. Table 21-2 gives register information about SBYCR.

In software standby mode current dissipation is reduced to an extremely low level because the CPU and on-chip peripheral modules all halt. The on-chip peripheral modules are reset. As long as the specified voltage is supplied, however, CPU register contents, on-chip RAM data, and I/O port states are held.

**Table 21-2 Standby Control Register**

Address	Name	Abbreviation	R/W	Initial Value
H'FF1A	Software standby control register	SBYCR	R/W	H'7F

### 21.3.2 Software Standby Control Register

The software standby control register (SBYCR) is an eight-bit register that must be set in order to enter software standby mode. The bit structure is described next.

Bit	7	6	5	4	3	2	1	0
	SSBY	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
R/W	R/W	—	—	—	—	—	—	—

Reserved bits  
Software standby bit  
Enables transition to software standby mode

(1) **Bit 7—Software Standby (SSBY):** Enables transition to software standby mode.

## Bit 7

SSBY	Description
0	SLEEP instruction causes transition to sleep mode. (Initial value)
1	SLEEP instruction causes transition to software standby mode

The SSBY bit cannot be set to 1 while the timer enable bit (TME) is set to 1 in the timer control/status register (TCSR) of the watchdog timer (WDT). Before entering software standby mode, software must clear the TME bit to 0.

The SSBY bit is automatically cleared to 0 when the chip recovers from software standby mode by NMI or reset, or enters hardware standby mode.

(2) **Bits 6 to 0—Reserved:** Read-only bits, always read as 1.

### 21.3.3 Exit from Software Standby Mode

The chip can be brought out of software standby mode by input at the NMI,  $\overline{\text{RES}}$ , or  $\overline{\text{STBY}}$  pin.

(1) **Recovery by NMI:** To recover from software standby mode by NMI input, software must set clock select bits 2 to 0 (CKS2 to CKS0) in the watchdog timer's timer control/status register (TCSR) beforehand to select the oscillator setting time\*, and must also select the desired NMI input edge.

When an NMI interrupt request signal is input, the clock oscillator begins operating. At first clock pulses are supplied only to the watchdog timer. The watchdog timer receives the supplied clock and starts counting. After the oscillator settling time selected by bits CKS2 to CKS0 in the control/status register (TCSR), the watchdog timer overflows. After the watchdog timer overflows, the clock is supplied to the entire chip, software standby mode ends, and the NMI exception-handling sequence begins.

(2) **Recovery by  $\overline{\text{RES}}$  Input:** When software standby mode is exited by  $\overline{\text{RES}}$  input, clock pulses are supplied to the entire chip as soon as the clock oscillator starts. The clock oscillator starts when the  $\overline{\text{RES}}$  signal goes low. After the oscillator settling time, when the  $\overline{\text{RES}}$  signal goes high, the CPU begins executing the reset sequence. The  $\overline{\text{RES}}$  signal must be held low long enough for the clock to stabilize.

(3) **Recovery by  $\overline{\text{STBY}}$  Input:** When the  $\overline{\text{STBY}}$  signal goes low, the chip exits from software standby mode to hardware standby mode.

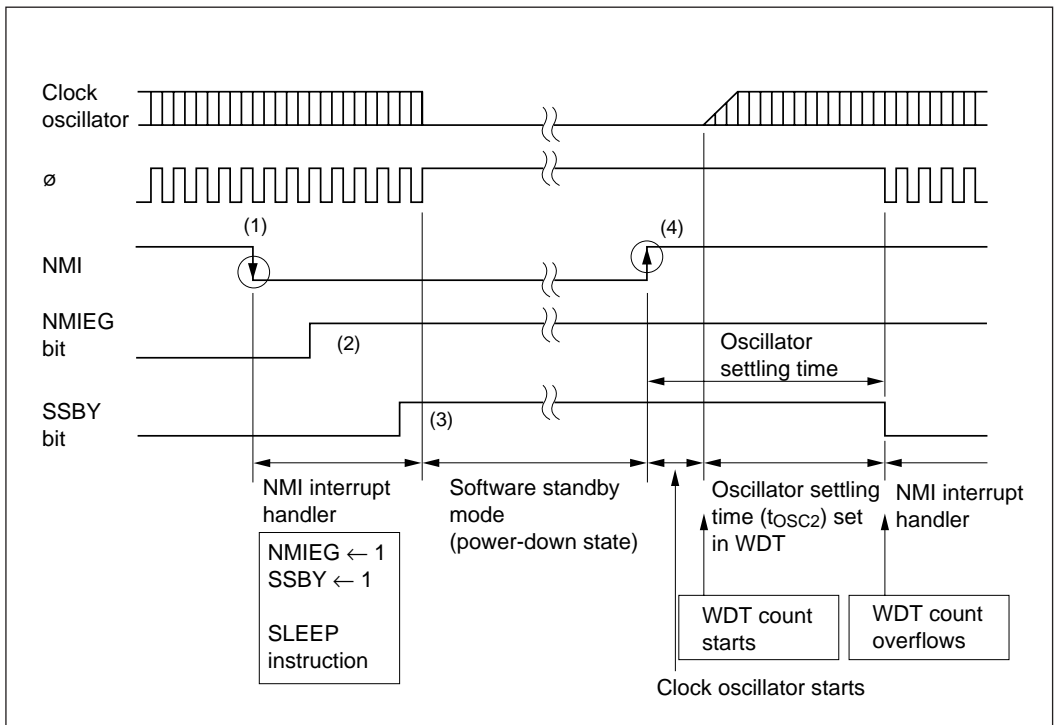
Note: \* When using an external clock, the watchdog timer's timer control/status register (TCSR) should be set so as to secure the external clock output settling delay time ( $t_{\text{DEXT}}$ ).



### 21.3.4 Sample Application of Software Standby Mode

Figure 21-1 illustrates NMI timing for software standby mode.

- (1) With the nonmaskable interrupt edge bit (NMIEG) in the NMI control register (NMICR) cleared to 0 (falling edge), NMI goes low.
- (2) The NMIEG bit is set to 1.
- (3) Software sets the SSBY bit to 1, then executes the SLEEP instruction. The chip enters software standby mode.
- (4) When the NMI signal goes high, the chip exits software standby mode.



**Figure 21-1 NMI Timing for Software Standby Mode (Example)**

### 21.3.5 Note

The I/O ports are not initialized in software standby mode. If a port is in the high output state, it remains in that state and power reduction is lessened by the amount of current output.

## 21.4 Hardware Standby Mode

This section describes hardware standby mode.

### 21.4.1 Transition to Hardware Standby Mode

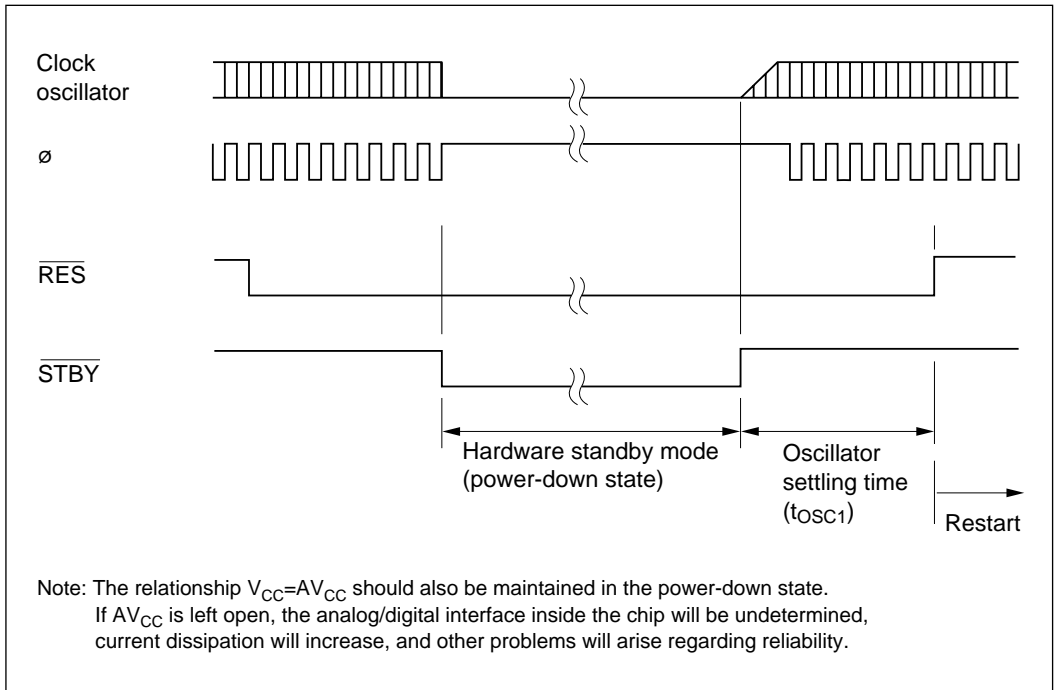
Regardless of its current state, the chip enters hardware standby mode whenever the  $\overline{\text{STBY}}$  pin goes low. Hardware standby mode reduces power consumption drastically by halting the CPU and stopping all functions of the on-chip peripheral modules. The on-chip peripheral modules are reset, but as long as the specified voltage is supplied, on-chip RAM contents are held. To hold RAM contents, the RAME bit in the RAM control register (RAMCR) should be cleared to 0. I/O ports are placed in the high-impedance state.

### 21.4.2 Recovery from Hardware Standby Mode

Recovery from the hardware standby mode requires inputs on both the  $\overline{\text{STBY}}$  and  $\overline{\text{RES}}$  lines. When  $\overline{\text{STBY}}$  goes high, the clock oscillator begins running.  $\overline{\text{RES}}$  should be low at this time. After the oscillator settling time, when the  $\overline{\text{RES}}$  signal goes high, the H8/500 CPU begins executing the reset sequence. The H8/500 CPU then returns to the program execution state, ending hardware standby mode.

### 21.4.3 Timing for Hardware Standby Mode

Figure 21-2 shows the timing relationships in hardware standby mode.



**Figure 21-2 Hardware Standby Mode Timing**

## Section 22 Electrical Characteristics (H8/539F)

### 22.1 Absolute Maximum Ratings

Table 22-1 lists the absolute maximum ratings.

**Table 22-1 Absolute Maximum Ratings**

Item	Symbol	Value	Unit
Power supply voltage	$V_{CC}$	−0.3 to +7.0	V
Programming voltage	$V_{PP}$	−0.3 to +13.0	V
Input voltage (MD2)	$V_{in}$	−0.3 to + 13.0	V
Input voltage (except ports 8, 9 and MD2)	$V_{in}$	−0.3 to $V_{CC} + 0.3$	V
Input voltage (ports 8 and 9)	$V_{in}$	−0.3 to $AV_{CC} + 0.3$	V
Reference voltage	$V_{REF}$	−0.3 to $AV_{CC} + 0.3$	V
Analog power supply voltage	$AV_{CC}$	−0.3 to +7.0	V
Analog input voltage	$V_{AN}$	−0.3 to $AV_{CC} + 0.3$	V
Operating temperature	$T_{opr}$	Regular specifications: −20 to +75	°C
		Wide-range specifications: −40 to +85	
Storage temperature	$T_{stg}$	−55 to +125	°C

Note: Permanent damage to the chip may result if absolute maximum ratings are exceeded. Ensure that the voltage applied to the Vpp and MD2 pins, including the peak overshoot, does not exceed 13 V. (Also be sure to connect a decoupling capacitor to the Vpp and MD2 pins.)

## 22.2 Electrical Characteristics

### 22.2.1 DC Characteristics

Tables 22-2 lists the DC characteristics. Table 22-3 lists the permissible output currents.

**Table 22-2 DC Characteristics**

Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$  ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Min	Typ	Max	Unit	Test Conditions	
Input high voltage	$\overline{RES}$ , $\overline{STBY}$ , $MD_2$ – $MD_0$	$V_{CC} - 0.7$	—	$V_{CC} + 0.3$	V		
	EXTAL	$V_{CC} \times 0.7$	—	$V_{CC} + 0.3$	V		
	Ports 8 and 9	2.2	—	$AV_{CC} + 0.3$	V		
	Other input pins (except ports 4 and 5)	2.2	—	$V_{CC} + 0.3$	V		
Input low voltage	$\overline{RES}$ , $\overline{STBY}$ , $MD_2$ – $MD_0$	–0.3	—	0.4	V		
	Other input pins (except ports 4 and 5)	–0.3	—	0.8	V		
Schmitt trigger input voltages	Ports 4 and 5	$VT^-$	1.0	—	2.5	V	
		$VT^+$	2.0	—	3.5	V	
		$VT^+ - VT^-$	0.4	—	—	V	
Input leakage current	$\overline{RES}/V_{PP}$	$I_{in}$	—	—	20	mA	$V_{CC} + 0.5\text{V} < V_{in} \leq 12.6\text{V}$
			—	—	10.0	$\mu\text{A}$	$0.5\text{V} \leq V_{in} \leq V_{CC} + 0.5\text{V}$
	$MD_2$		—	—	50.0	$\mu\text{A}$	$V_{CC} + 0.5\text{V} < V_{in} \leq 12.6\text{V}$
			—	—	10.0	$\mu\text{A}$	$0.5\text{V} \leq V_{in} \leq V_{CC} + 0.5\text{V}$
	$\overline{RES}$ , $\overline{STBY}$ , $NMI$ , $MD_0$ – $MD_2$		—	—	1.0	$\mu\text{A}$	$V_{in} = 0.5\text{ to }V_{CC} - 0.5\text{ V}$
	Ports 8 and 9		—	—	1.0	$\mu\text{A}$	$V_{in} = 0.5\text{ to }AV_{CC} - 0.5\text{ V}$
Leakage current in 3-state (off-state)	Ports 1 to 7 and A to C	$I_{STI}$	—	—	1.0	$\mu\text{A}$	$V_{in} = 0.5\text{ to }V_{CC} - 0.5\text{ V}$
Input pull-up transistor current	Ports B and C	$-I_P$	50	—	300	$\mu\text{A}$	$V_{in} = 0\text{ V}$

**Table 22-2 DC Characteristics (cont)**

Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$ , ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item		Symbol	Min	Typ	Max	Unit	Conditions
Output high voltage	All output pins	$V_{OH}$	$V_{CC} - 0.5$	—	—	V	$I_{OH} = -200\ \mu\text{A}$
			3.5	—	—	V	$I_{OH} = -1\ \text{mA}$
Output low voltage	All output pins (except $\overline{\text{RESO}}$ )	$V_{OL}$	—	—	0.4	V	$I_{OL} = 1.6\ \text{mA}$
			Ports 3 and 5	—	—	1.0	V
	$\overline{\text{RESO}}$	$V_{OL}$	—	—	1.2	V	$I_{OL} = 10\ \text{mA}$
			—	—	0.4	V	$I_{OL} = 2.6\ \text{mA}$
High voltage (12V)*1 applied criterion	$\overline{\text{RESO}}/V_{PP}$ , MD <sub>2</sub>	$V_H$	$V_{CC} + 2.0$	—	11.4	V	$V_{CC} = 4.5\text{ to }5.5\ \text{V}$
Input capacitance	$\overline{\text{RESO}}/V_{PP}$	$C_{in}$	—	—	100	pF	$V_{in} = 0\ \text{V}$
	NMI, MD <sub>2</sub>		—	—	50	pF	$f = 1\ \text{MHz}$
	All input pins except $\overline{\text{RESO}}$ , $\overline{V_{PP}}$ , NMI, and MD <sub>2</sub>		—	—	20	pF	$T_a = 25^\circ\text{C}$
Current dissipation	Normal operation	$I_{CC}$	—	65	100	mA	$f = 16\ \text{MHz}$
	Sleep mode		—	40	60	mA	$f = 16\ \text{MHz}$
	Standby mode		—	0.01	5.0	$\mu\text{A}$	$T_a \leq 50^\circ\text{C}$
			—	—	20.0	$\mu\text{A}$	$50^\circ\text{C} < T_a$
Analog power supply current	During A/D conversion	$AI_{CC}$	—	1.2	2.0	mA	
	Idle		—	0.01	5.0	$\mu\text{A}$	
Reference current	During A/D conversion	$AI_{CC}$	—	0.2	0.5	mA	
	Idle		—	0.01	5.0	$\mu\text{A}$	

\*1: The high voltage applied criterion is as shown in the table, but in boot mode and for flash memory programming and erasing, a value of  $12.0 \pm 0.6\ \text{V}$  should be set.

**Table 22-2 DC Characteristics (cont)**

Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$ , ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item		Symbol	Min	Typ	Max	Unit	Conditions
$V_{PP}$ current	Read	$I_{PP}$	—	—	10	$\mu\text{A}$	$V_{PP} = 5.0\text{ V}$
			—	10	20	$\text{mA}$	$V_{PP} = 12.6\text{ V}$
	Program	—	20	40	$\text{mA}$		
	Erase	—	20	40	$\text{mA}$		
RAM standby voltage		$V_{RAM}$	2.0	—	—	$\text{V}$	

- Notes: 1. Never leave the  $AV_{CC}$ ,  $AV_{SS}$ , and  $V_{REF}$  pins open. If the A/D converter is not used, connect  $AV_{CC}$  and  $V_{REF}$  to  $V_{CC}$  and connect  $AV_{SS}$  to  $V_{SS}$ .
2. Current dissipation values are for  $V_{IHmin} = V_{CC} - 0.5\text{ V}$  and  $V_{ILmax} = 0.5\text{ V}$  with all output pins unloaded and the on-chip pull-up transistors in the off state.

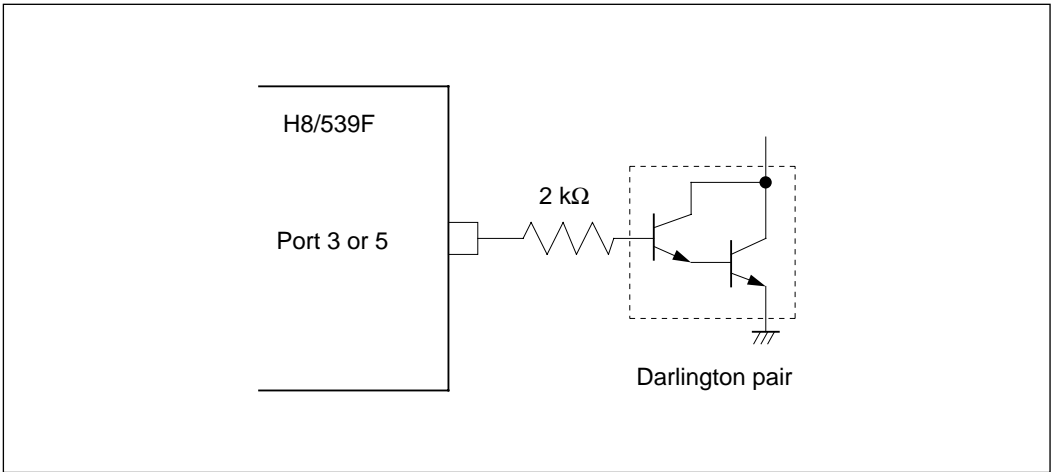
**Table 22-3 Permissible Output Currents**

Condition:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$  ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

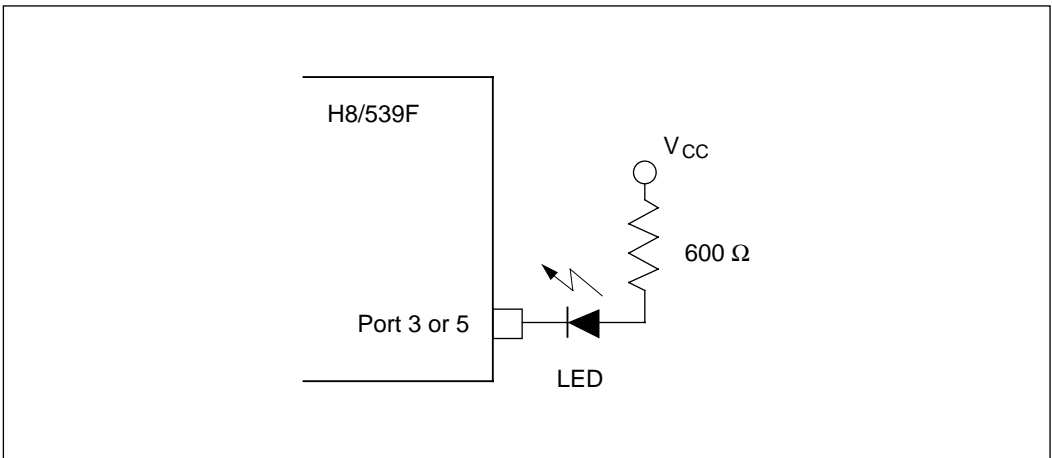
Item	Symbol	Condition			Unit	
		Min	Typ	Max		
Permissible output low current (per pin)	Ports 3 and 5	$I_{OL}$	—	—	10	mA
	<u>RESO</u>		—	—	3.0	mA
	Other output pins		—	—	2.0	mA
Permissible output low current (total)	Total of 14 pins in ports 3 and 5	$\Sigma I_{OL}$	—	—	40	mA
	Total of all output pins, including the above		—	—	80	mA
Permissible output high current	Per pin	$-I_{OH}$	—	—	2.0	mA
Permissible output high current	Total of all output pins	$\Sigma -I_{OH}$	—	—	25	mA

- Notes: 1. To protect chip reliability, do not exceed the output current values in table 22-3.  
 2. When driving a Darlington pair or LED, always insert a current-limiting resistor in the output line, as shown in figures 22-1 and 22-2.





**Figure 22-1 Darlington Pair Drive Circuit (Example)**



**Figure 22-2 LED Drive Circuit (Example)**

### 22.2.2 AC Characteristics

The AC characteristics of the H8/539F are described below. Bus timing parameters are listed in table 22-4. Control signal timing parameters are listed in table 22-5. Timing parameters of the on-chip peripheral modules are listed in table 22-6.

**Table 22-4 Bus Timing**

Condition:  $V_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $V_{REF} = 5.0 \text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0 \text{ V}$ ,  $T_a = -20 \text{ to } +75^\circ\text{C}$  (regular specifications),  $T_a = -40 \text{ to } +85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Condition		Unit	Test Conditions
		Min	Max		
		<b>16 MHz</b>			
Clock cycle time	$t_{CYC}$	62.5	500	ns	Fig. 22-4, Fig. 22-5
Clock low pulse width	$t_{CL}$	20	—	ns	
Clock high pulse width	$t_{CH}$	20	—	ns	
Clock rise time	$t_{Cr}$	—	15	ns	
Clock fall time	$t_{Cf}$	—	15	ns	
Address delay time	$t_{AD}$	—	25	ns	
Address hold time	$t_{AH}$	10	—	ns	
Address strobe delay time 1	$t_{ASD1}$	—	25	ns	
Address strobe delay time 2	$t_{ASD2}$	—	25	ns	
Read strobe delay time 1	$t_{RDD1}$	—	25	ns	
Read strobe delay time 2	$t_{RDD2}$	—	25	ns	
Write strobe delay time 1	$t_{WRD1}$	—	25	ns	
Write strobe delay time 2	$t_{WRD2}$	—	25	ns	
Write strobe delay time 3	$t_{WRD3}$	—	25	ns	
Write data strobe pulse width 1	$t_{WRW1}$	50	—	ns	
Write data strobe pulse width 2	$t_{WRW2}$	170	—	ns	
Address setup time 1	$t_{AS1}$	10	—	ns	
Address setup time 2	$t_{AS2}$	10	—	ns	
Address setup time 3	$t_{AS3}$	30	—	ns	
Read data setup time	$t_{RDS}$	20	—	ns	
Read data hold time	$t_{RDH}$	0	—	ns	
Read data access time 1	$t_{ACC1}$	—	60	ns	
Read data access time 2	$t_{ACC2}$	—	120	ns	

**Table 22-4 Bus Timing (cont)**

Condition:  $V_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $V_{REF} = 5.0 \text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0 \text{ V}$ ,  $T_a = -20 \text{ to } +75^\circ\text{C}$  (regular specifications),  $T_a = -40 \text{ to } +85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Condition		Unit	Test Conditions
		Min	Max		
Write data delay time	$t_{WDD}$	—	55	ns	Fig. 22-4, Fig. 22-5
Write data setup time	$t_{WDS}$	2	—	ns	
Write data hold time	$t_{WDH}$	10	—	ns	
Wait setup time	$t_{WTS}$	25	—	ns	Fig. 22-6
Wait hold time	$t_{WTH}$	10	—	ns	
Bus request setup time	$t_{BRQS}$	30	—	ns	Fig. 22-10
Bus acknowledge delay time 1	$t_{BACD1}$	—	30	ns	
Bus acknowledge delay time 2	$t_{BACD2}$	—	30	ns	
Bus-floating delay time	$t_{BZD}$	—	$t_{BACD1}$	ns	

**Table 22-5 Control Signal Timing**

Condition:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

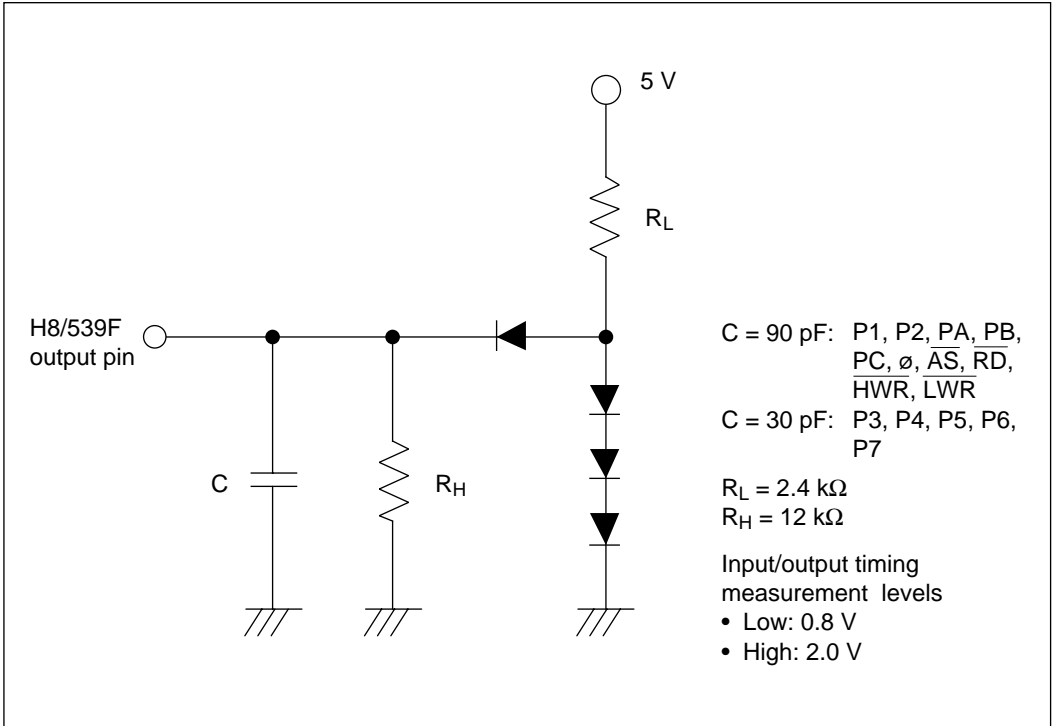
Item	Symbol	Condition		Unit	Test Conditions
		Min	Max		
$\overline{\text{RES}}$ setup time	$t_{\text{RESS}}$	200	—	ns	Fig. 22-7
$\overline{\text{RES}}$ pulse width	$t_{\text{RESW}}$	6.0	—	$t_{\text{cyc}}$	
Mode programming setup time*	$t_{\text{MDS}}$	4.0	—	$t_{\text{cyc}}$	
$\overline{\text{RESO}}$ output delay time	$t_{\text{RESD}}$	—	200	ns	Fig. 22-8
$\overline{\text{RESO}}$ output pulse width	$t_{\text{RESOW}}$	132	—	$t_{\text{cyc}}$	
NMI setup time	$t_{\text{NMIS}}$	150	—	ns	Fig. 22-9
NMI hold time	$t_{\text{NMIH}}$	10	—	ns	
$\overline{\text{IRQ}}_0$ setup time	$t_{\text{IRQ0S}}$	50	—	ns	
$\overline{\text{IRQ}}_{1-3}$ setup time	$t_{\text{IRQ1S}}$	50	—	ns	
$\overline{\text{IRQ}}_{1-3}$ hold time	$t_{\text{IRQ1H}}$	10	—	ns	
NMI pulse width (for recovery from software standby mode)	$t_{\text{NMIW}}$	200	—	ns	
Clock oscillator settling time at reset (crystal)	$t_{\text{OSC1}}$	20	—	ms	Fig. 22-11
Clock oscillator settling time in software standby (crystal)	$t_{\text{OSC2}}$	10	—	ms	Fig. 21-1
External clock output settling delay time (When inputting external clock from the EXTAL pin)	$t_{\text{DEXT}}$	500	—	$\mu\text{s}$	Fig. 22-12

Note:\* In boot mode, the input high voltage to MD2 should satisfy the high voltage (12V) applied criterion ( $V_H$  max).

**Table 22-6 Timing of On-Chip Supporting Modules**

Condition:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item	Item	Symbol	Condition		Unit	Test conditions
			Min	Max		
IPU	Timer output delay time	$t_{TOD}$	—	100	ns	Fig. 22-15
	Timer input setup time	$t_{TICS}$	50	—	ns	
	Timer clock input setup time	$t_{TCKS}$	50	—	ns	Fig. 22-16
	Timer clock pulse width	$t_{TCKW}$	1.5	—	$t_{CYC}$	
SCI	Input clock cycle	Asynchronous	4	—	$t_{CYC}$	Fig. 22-17
		Clocked synchronous	6	—	$t_{CYC}$	
	Input clock pulse width	$t_{SCKW}$	0.4	0.6	$t_{SCYC}$	
	Transmit data delay time	$t_{TXD}$	—	100	ns	Fig. 22-18
	Receive data setup time (clocked synchronous)	$t_{RXS}$	100	—	ns	
	Receive data hold time (clocked synchronous)	$t_{RXH}$	100	—	ns	
Ports	Output data delay time	$t_{PWD}$	—	50	ns	Fig. 22-13
	Receive data setup time (clocked synchronous)	$t_{PRS}$	50	—	ns	
	Receive data hold time (clocked synchronous)	$t_{PRH}$	50	—	ns	
PWM	output delay time	$t_{PWDD}$	—	100	ns	Fig. 22-14



**Figure 22-3 Output Load Circuit**

### 22.2.3 A/D Conversion Characteristics

Table 22-7 lists the A/D conversion characteristics of the H8/539F. Table 22-8 lists the permissible signal-source impedance for the A/D converter.

**Table 22-7 A/D Converter Characteristics**

Condition:  $V_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $V_{REF} = 5.0 \text{ V}$  ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0 \text{ V}$ ,  $T_a = -20$  to  $+75^\circ\text{C}$  (regular specifications),  $T_a = -40$  to  $+85^\circ\text{C}$  (wide-range specifications)

Item	Condition			Unit
	16 MHz			
	Min	Typ	Max	
Resolution	10	10	10	Bits
Conversion time	—	—	8.38	$\mu\text{s}$
Analog input capacitance	—	—	20	pF
Nonlinearity error	—	—	$\pm 2.0$	LSB
Offset error	—	—	$\pm 2.0$	LSB
Full-scale error	—	—	$\pm 2.0$	LSB
Quantization error	—	—	$\pm 1/2$	LSB
Absolute accuracy	—	—	$\pm 2.5$	LSB

**Table 22-8 A/D Converter Characteristics: Allowable Signal-Source Impedance**

Item	Conditions	Min	Typ	Max	Unit
Allowable signal-source impedance	$8.38 \mu\text{s} \leq \text{conversion time} < 13.4 \mu\text{s}$	—	—	5	$\text{k}\Omega$
	$4.5 \text{ V} \leq AV_{CC} < 5.5 \text{ V}$	—	—	10	

## 22.2.4 Flash Memory Characteristics

Table 22-9 lists the flash memory characteristics of the H8/539F.

**Table 22-9 Flash Memory Characteristics**

Conditions:  $V_{CC} = 4.5$  to  $5.5$  V,  $AV_{CC} = 4.5$  to  $5.5$  V,  $V_{REF} = 4.5$  to  $5.5$  V,  $V_{SS} = AV_{SS} = 0$  V,  $V_{PP} = 12.0 \pm 0.6$  V,  $T_a = -20$  to  $+75^\circ\text{C}$  (regular specifications),  $T_a = -40$  to  $85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Min	Typ	Max	Unit	Test Conditions
Programming time*1, *2	$t_P$	—	50	1000	$\mu\text{s}/\text{byte}$	
Erase time*1, *3	$t_E$	—	1	30	s	
Number of writing / erasing count	$N_{WEC}$	—	—	100	times	
Verify-setup time 1*1	$t_{vs1}$	4	—	—	$\mu\text{s}$	
Verify- setup time 2*1	$t_{vs2}$	2	—	—	$\mu\text{s}$	
$V_{PP}$ enable setup time	$t_{VPS}$	5	—	—	$\mu\text{s}$	
Flash-memory-read setup time*4	$t_{FRS}$	50	—	—		$V_{CC} \geq 4.5\text{V}$ Fig.22-19 Fig.22-20

- Notes: 1. Set the times following the programming/erasing algorithm shown in section 19, "Flash Memory."
- The programming time is the time during which a byte is programmed or the P bit in the flash memory control register (FLMCR) is set. It does not include the program-verify time.
  - The erase time is the time during which all blocks (128 kbytes) are erased or the E bit in the flash memory control register (FLMCR) is set. It does not include the prewriting time before erasure or erase-verify time.
  - After power-on when using an external clock source, after return from standby mode, or after clearing the  $V_{PP}$  enable bit, make sure that this read setup time has elapsed before reading flash memory.



## 22.3 Operational Timing

This section shows timing diagrams of H8/539F operations.

### 22.3.1 Bus Timing

This section gives the following bus timing diagrams:

1. Basic bus cycle: two-state access

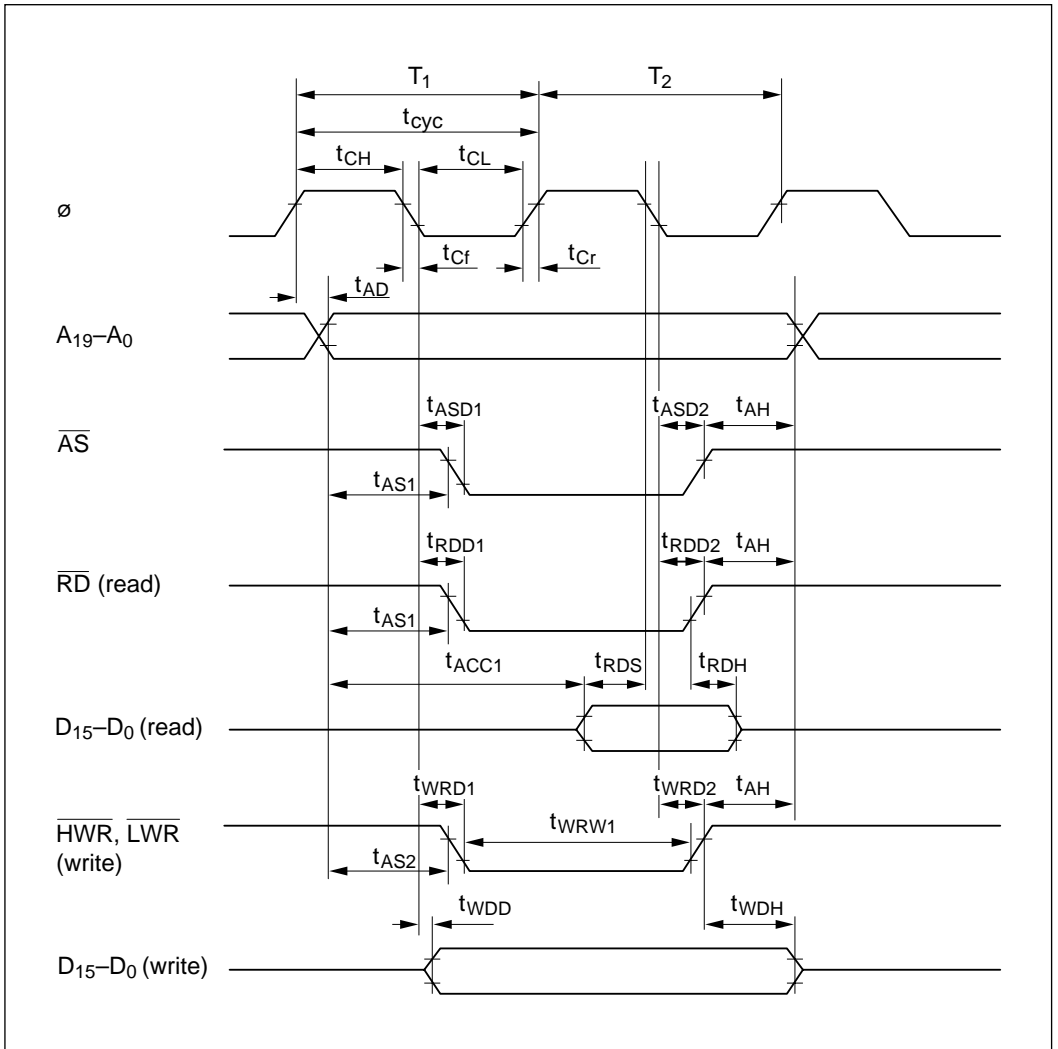
Figure 22-4 shows the timing of the external two-state access cycle.

2. Basic bus cycle: three-state access

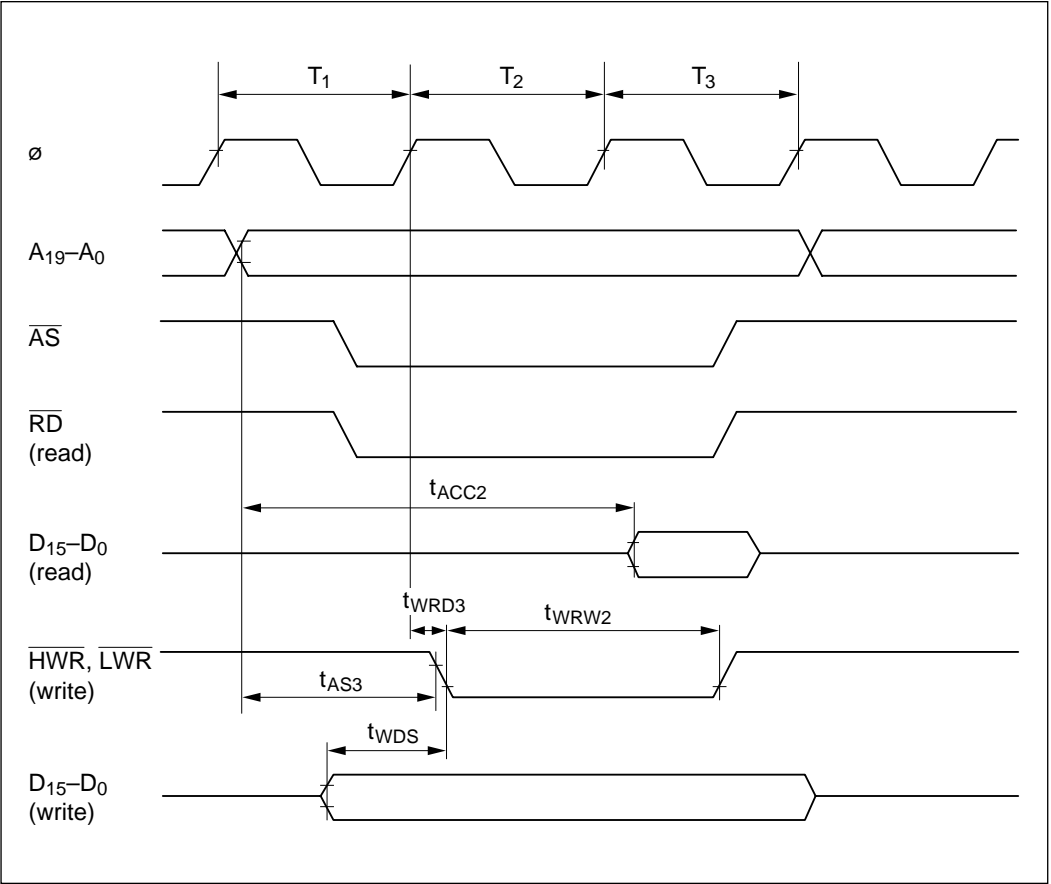
Figure 22-5 shows the timing of the external three-state access cycle.

3. Basic bus cycle: three-state access with one wait state

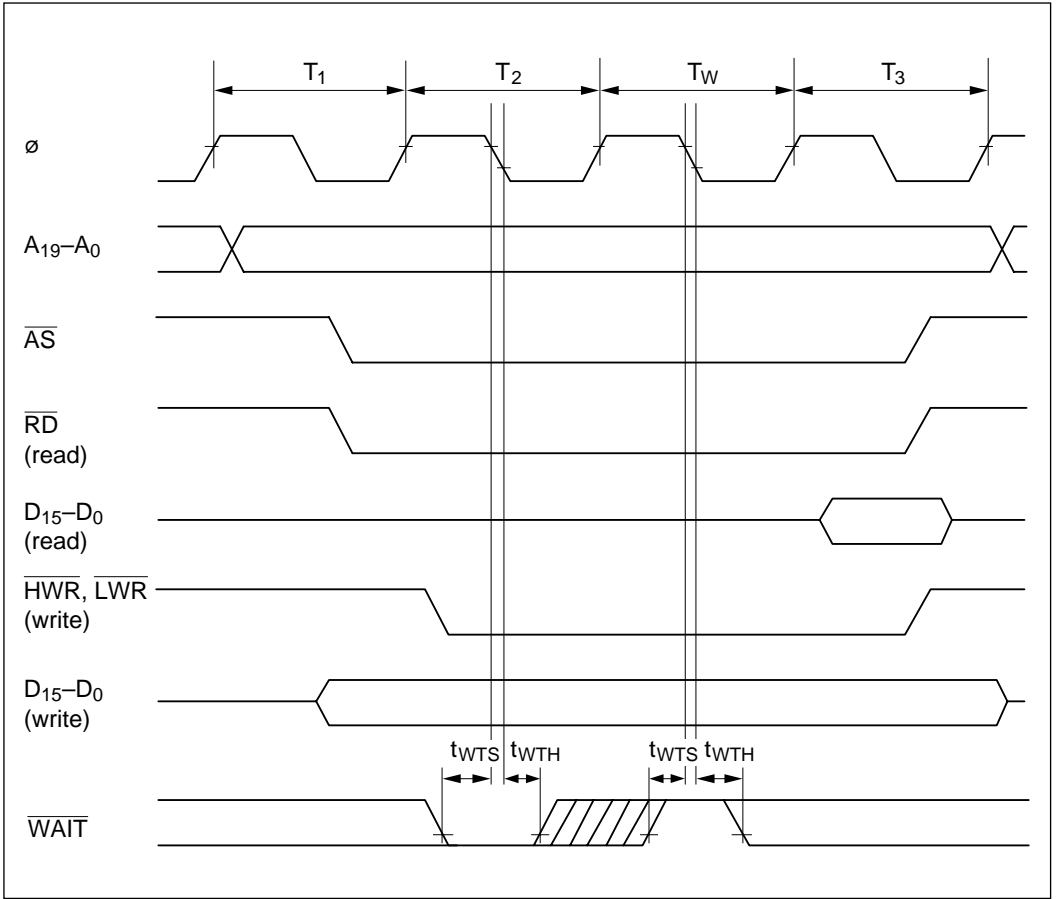
Figure 22-6 shows the timing of the external three-state access cycle with one wait state inserted.



**Figure 22-4 Basic Bus Cycle: Two-State Access**



**Figure 22-5 Basic Bus Cycle: Three-State Access**



**Figure 22-6 Basic Bus Cycle: Three-State Access with One Wait State**

### 22.3.2 Control Signal Timing

This section gives the following control signal timing diagrams:

1. Reset input timing

Figure 22-7 shows the reset input timing.

2. Reset output timing

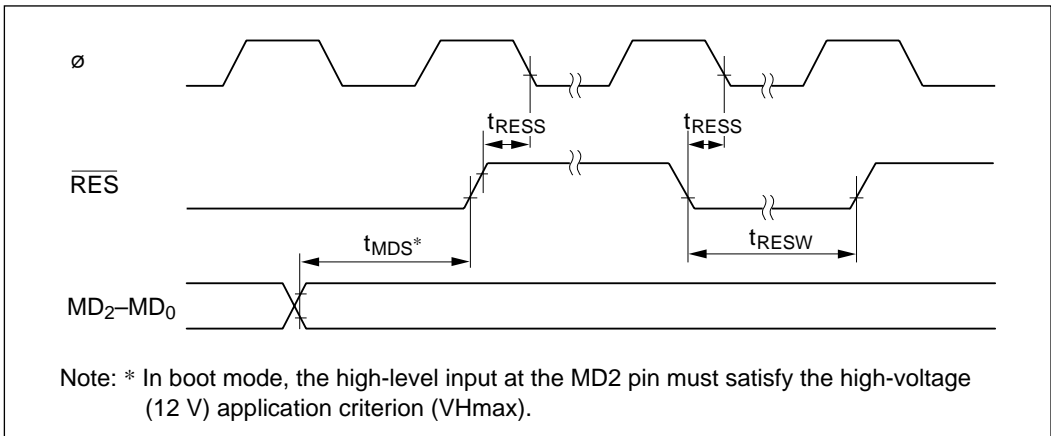
Figure 22-8 shows the reset output timing.

3. Interrupt input timing

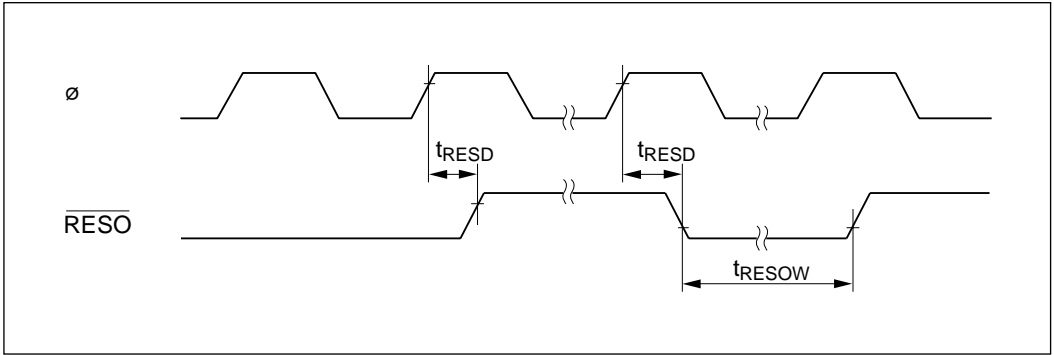
Figure 22-9 shows the input timing for NMI, IRQ<sub>0</sub>, and IRQ<sub>1</sub> to IRQ<sub>3</sub>.

4. Bus-release mode timing

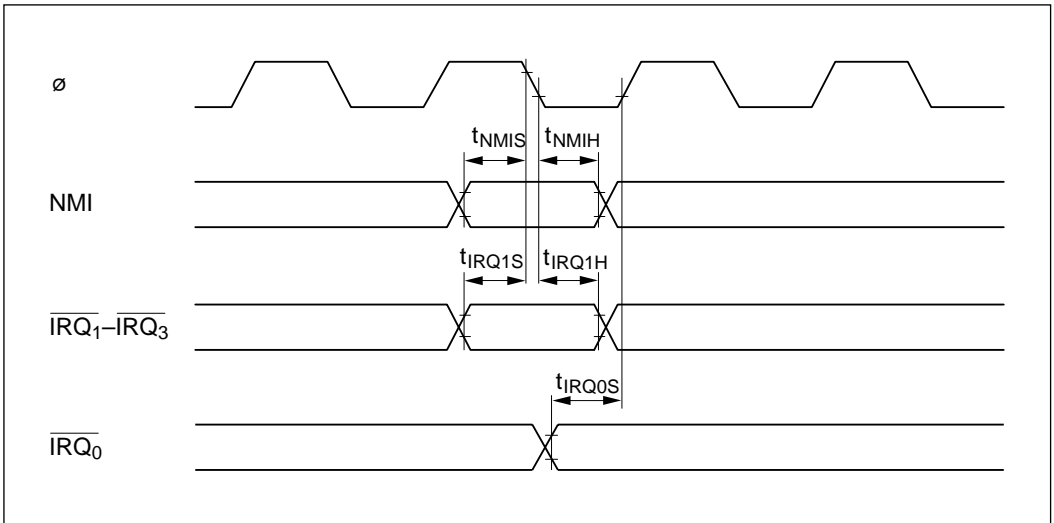
Figure 22-10 shows the bus-release mode timing.



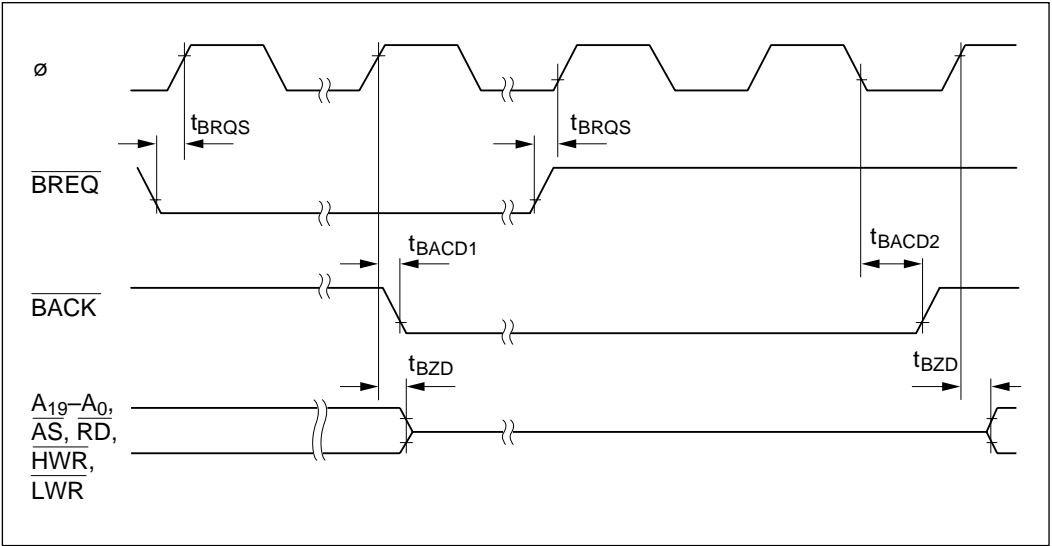
**Figure 22-7 Reset Input Timing**



**Figure 22-8 Reset Output Timing**



**Figure 22-9 Interrupt Input Timing**



**Figure 22-10 Bus-Release Mode Timing**

### 22.3.3 Clock Timing

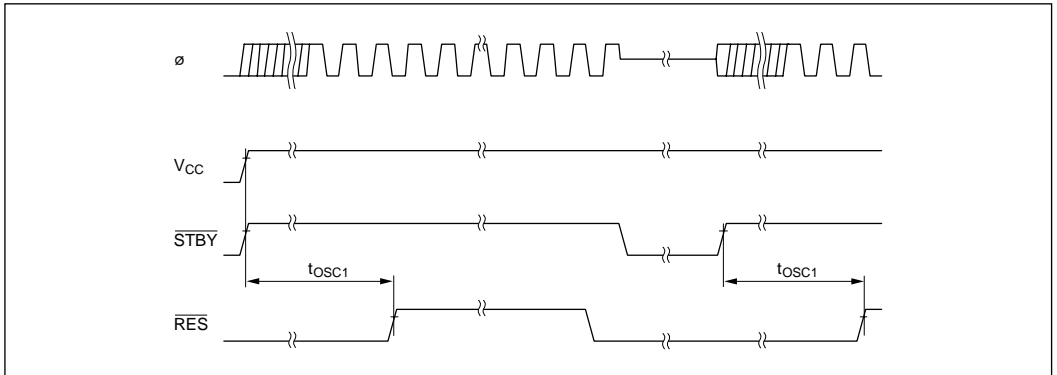
This section gives the following H8/539F clock timing diagram:

1. Oscillator settling timing

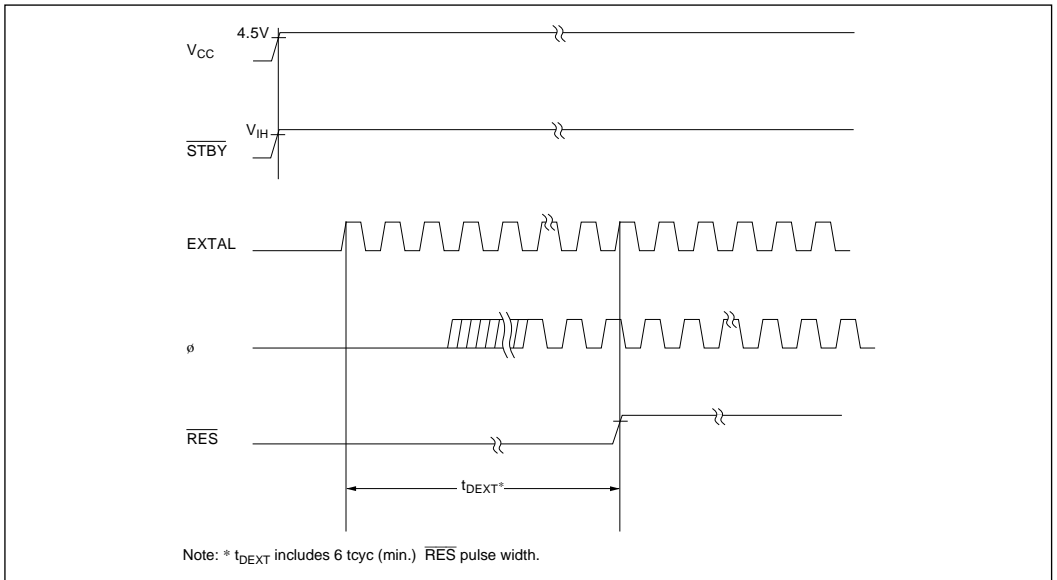
Figure 22-11 shows the oscillator settling timing.

2. External clock output settling delay timing

Figure 22-12 shows the external clock output settling delay timing.



**Figure 22-11 Oscillator Settling Timing**



**Figure 22-12 External Clock Output Settling Delay Timing**



### 22.3.4 I/O Port Timing

This section gives the following H8/539F I/O port input/output timing diagram:

#### 1. I/O port input/output timing

Figure 22-13 shows the I/O port input/output timing.

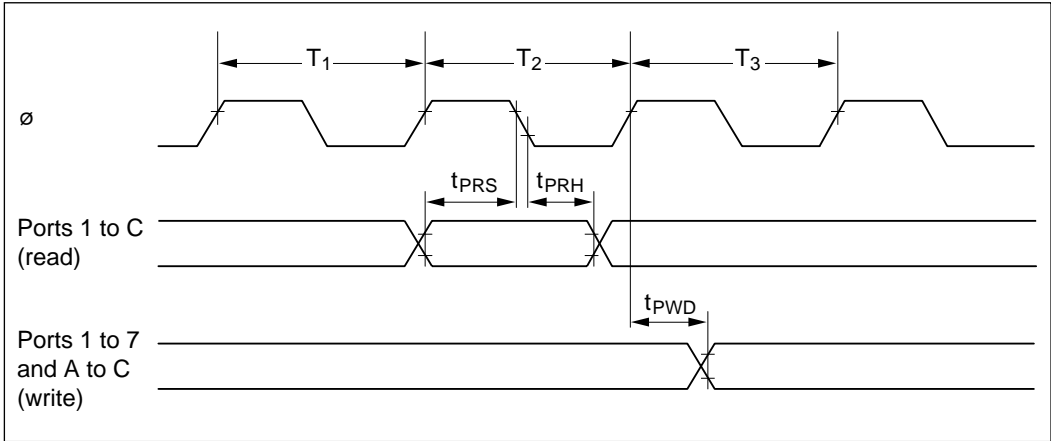


Figure 22-13 I/O Port Input/Output Timing

### 22.3.5 PWM Timer Output Timing

This section gives the following H8/539F PWM timer output timing diagram.

#### 1. PWM timer output timing

Figure 22-14 shows the PWM timer output timing.

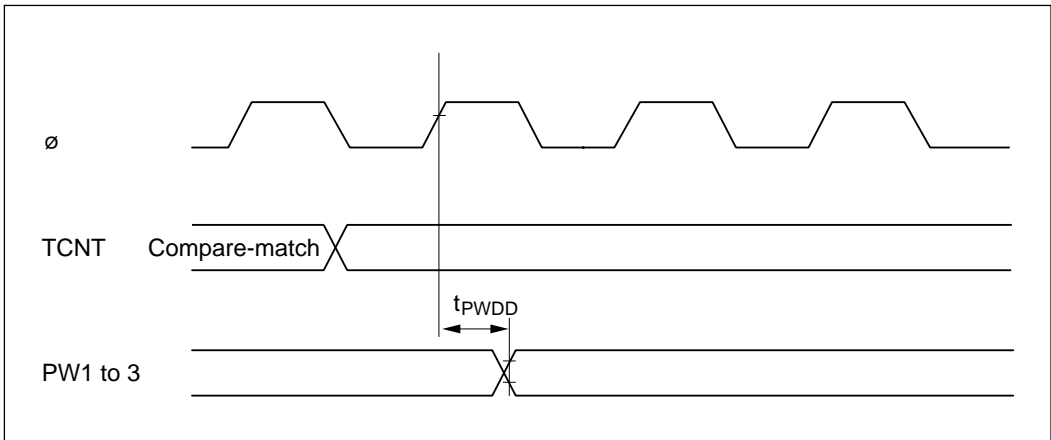


Figure 22-14 PWM Timer Output Timing

### 22.3.6 IPU Timing

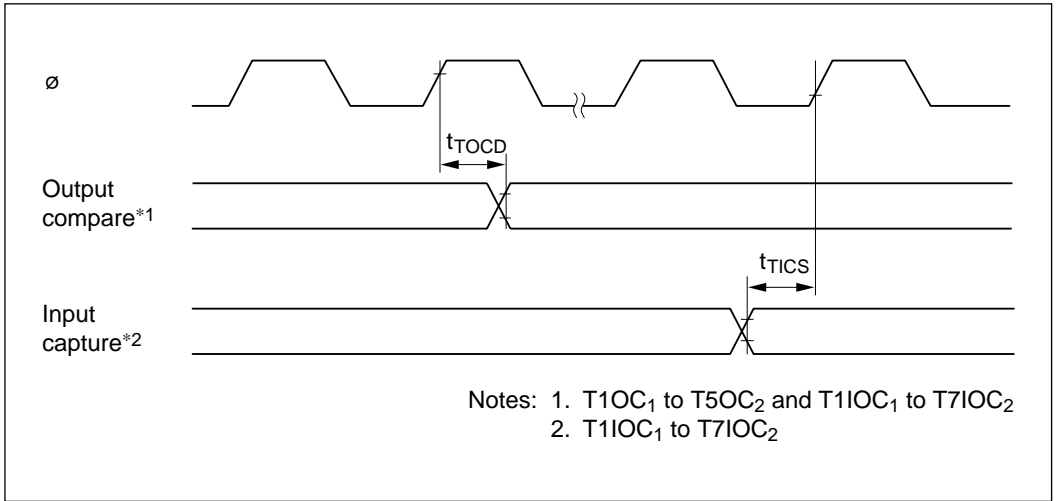
This section gives the following H8/539F IPU timing diagrams:

1. IPU input/output timing

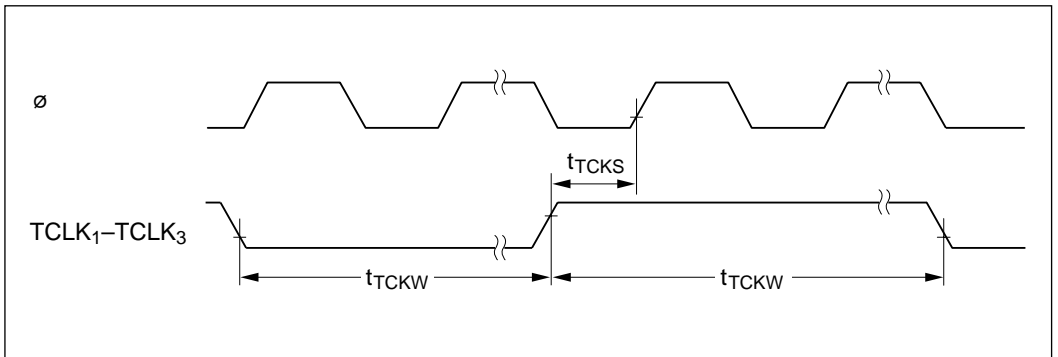
Figure 22-15 shows the IPU input/output timing.

2. IPU external clock input timing

Figure 22-16 shows the IPU external clock input timing.



**Figure 22-15 IPU Input/Output Timing**



**Figure 22-16 IPU Clock Input Timing**

### 22.3.7 SCI Input/Output Timing

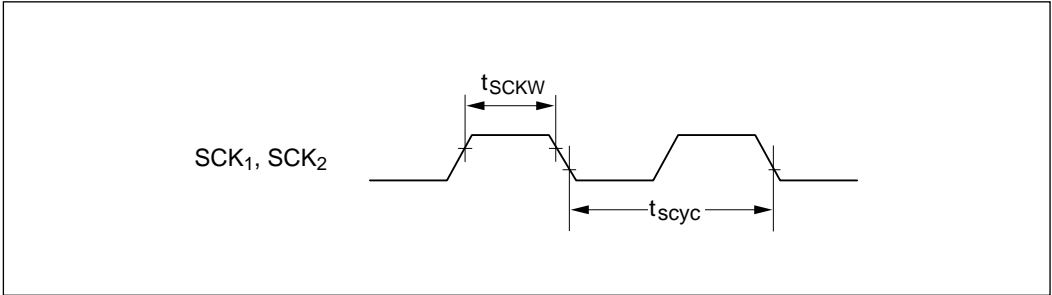
This section gives the following H8/539F SCI timing diagrams:

1. SCI input clock timing

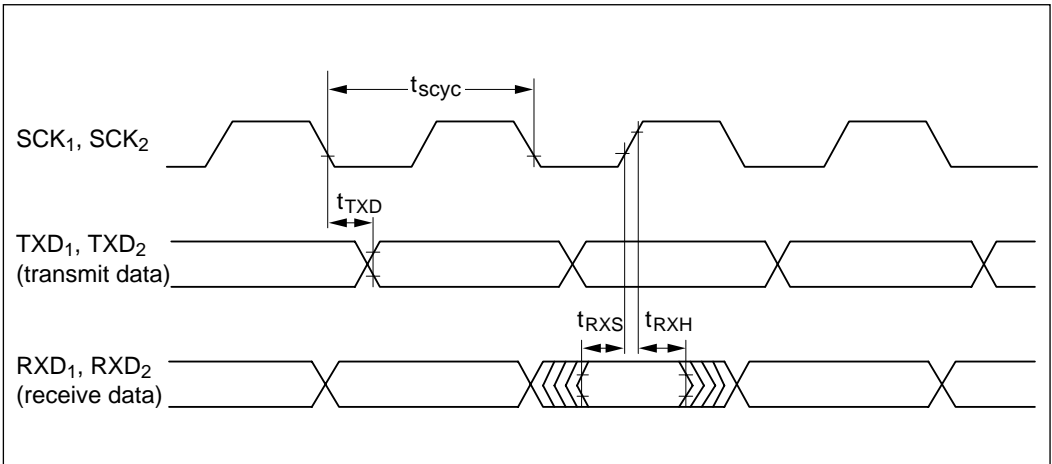
Figure 22-17 shows the SCI input clock timing.

2. SCI input/output timing (clocked synchronous mode)

Figure 22-18 shows the SCI input/output timing in clocked synchronous mode.



**Figure 22-17 SCK Input Clock Timing**

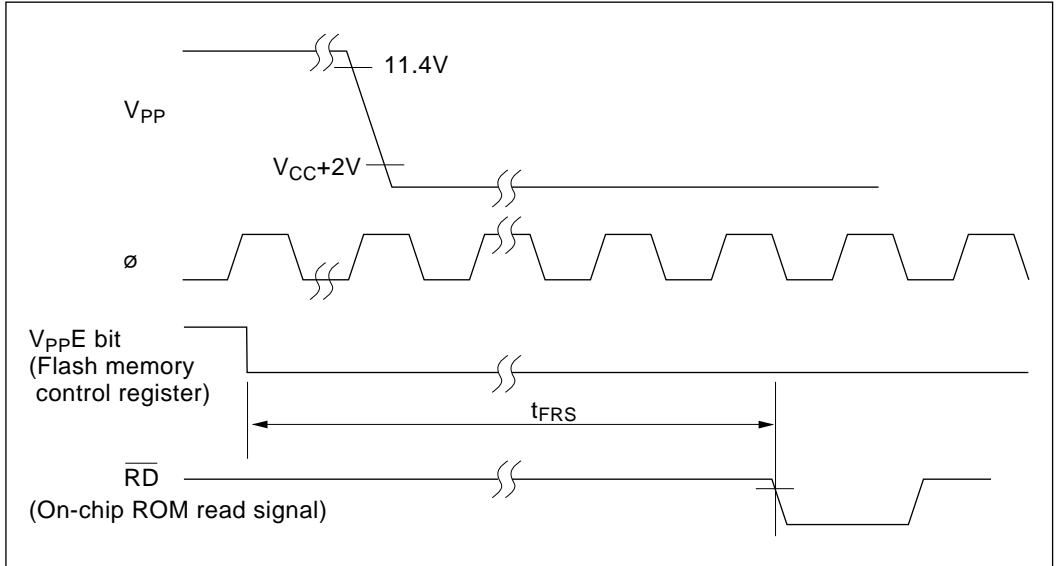


**Figure 22-18 SCI Input/Output Timing  
(Clocked Synchronous Mode)**

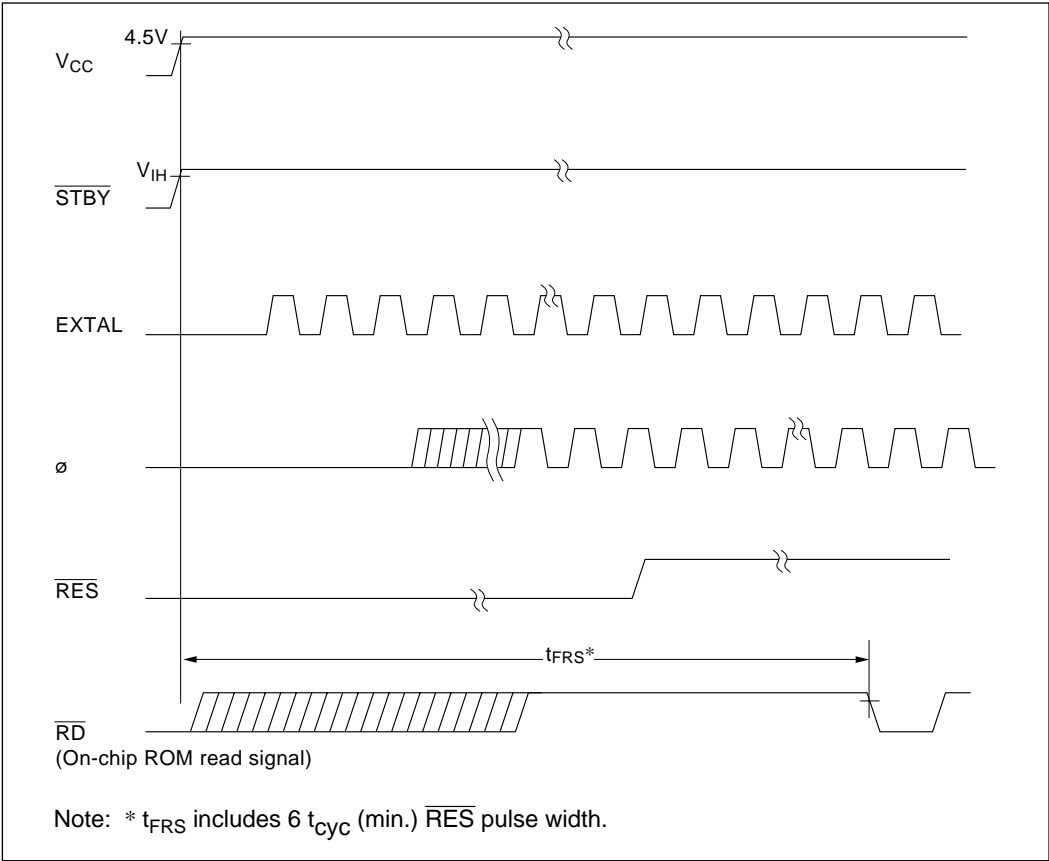
### 22.3.8 Flash Memory Read Timing

This section gives the following H8/539F on-chip flash memory read timing diagrams:

1. Flash memory read timing (after clearing  $V_{PP}E$  bit)  
Figure 22-19 shows the flash memory read timing after  $V_{PP}$  power-off.
2. Flash memory read timing (when using external clock)  
Figure 22-20 shows the flash memory read timing after  $V_{CC}$  power-on and exit from standby mode when an external clock is input from the EXTAL pin.



**Figure 22-19 Flash Memory Read Timing (After Clearing  $V_{PP}E$  Bit)**



**Figure 22-20 Flash Memory Read Timing (When Using External Clock)**

## Section 23 Electrical Characteristics (H8/539F S Mask model)

### 23.1 Absolute Maximum Ratings (H8/539F S Mask model)

Table 23-1 lists the absolute maximum ratings.

**Table 23-1 Absolute Maximum Ratings**

Item	Symbol	Value	Unit
Power supply voltage	$V_{CC}$	-0.3 to +7.0	V
Programming voltage (FWE)* <sup>1</sup>	$V_{in}$	-0.3 to $V_{CC} + 0.3$	V
Input voltage (MD2)* <sup>1</sup>	$V_{in}$	-0.3 to $V_{CC} + 0.3$	V
Input voltage (except ports 8 and 9)	$V_{in}$	-0.3 to $V_{CC} + 0.3$	V
Input voltage (ports 8 and 9)	$V_{in}$	-0.3 to $AV_{CC} + 0.3$	V
Reference voltage	$V_{REF}$	-0.3 to $AV_{CC} + 0.3$	V
Analog power supply voltage	$AV_{CC}$	-0.3 to +7.0	V
Analog input voltage	$V_{AN}$	-0.3 to $AV_{CC} + 0.3$	V
Operating temperature	$T_{opr}$	Regular specifications: -20 to +75* <sup>2</sup>	°C
		Wide-range specifications: -40 to +85* <sup>2</sup>	
Storage temperature	$T_{stg}$	-55 to +125	°C

Note: Permanent damage to the chip may result if absolute maximum ratings are exceeded.

\*1: 12 V must not be applied to the S-mask model (single power source), as this may cause permanent damage to the chip.

\*2: The operating temperature range during flash memory programming and erasing is:  $T_a = 0$  to +70°C.

## 23.2 Electrical Characteristics (H8/539F S Mask Model)

### 23.2.1 DC Characteristics

Tables 23-2 and lists the DC characteristics. Table 23-3 lists the permissible output currents.

**Table 23-2 DC Characteristics**

Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$  ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Min	Typ	Max	Unit	Test Conditions	
Input high voltage	RES, STBY, MD <sub>2</sub> –MD <sub>0</sub> , FWE	$V_{CC} - 0.7$	—	$V_{CC} + 0.3$	V		
	EXTAL	$V_{CC} \times 0.7$	—	$V_{CC} + 0.3$	V		
	Ports 8 and 9	2.2	—	$AV_{CC} + 0.3$	V		
	Other input pins (except ports 4 and 5)	2.2	—	$V_{CC} + 0.3$	V		
Input low voltage	RES, STBY, MD <sub>2</sub> –MD <sub>0</sub> , FWE	–0.3	—	0.4	V		
	Other input pins (except ports 4 and 5)	–0.3	—	0.8	V		
Schmitt trigger input voltages	Ports 4 and 5	VT <sup>–</sup>	1.0	—	2.5	V	
		VT <sup>+</sup>	2.0	—	3.5	V	
		VT <sup>+</sup> – VT <sup>–</sup>	0.4	—	—	V	
Input leakage current	FWE	—	—	10	μA	V <sub>in</sub> = 0.5 to V <sub>CC</sub> – 0.5 V	
	MD <sub>2</sub>	—	—	1.0	μA	V <sub>in</sub> = 0.5 to V <sub>CC</sub> – 0.5 V	
	RES, STBY, NMI, MD <sub>0</sub> –MD <sub>2</sub>	—	—	1.0	μA	V <sub>in</sub> = 0.5 to V <sub>CC</sub> – 0.5 V	
	Ports 8 and 9	—	—	1.0	μA	V <sub>in</sub> = 0.5 to AV <sub>CC</sub> – 0.5 V	
Leakage current in 3-state (off-state)	Ports 1 to 7 and A to C	I <sub>STI</sub>	—	—	1.0	μA	V <sub>in</sub> = 0.5 to V <sub>CC</sub> – 0.5 V
Input pull-up transistor current	Ports B and C	–I <sub>P</sub>	50	—	300	μA	V <sub>in</sub> = 0 V

**Table 23-2 DC Characteristics (cont)**

Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$ , ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to } +75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to } +85^\circ\text{C}$  (wide-range specifications)

Item		Symbol	Min	Typ	Max	Unit	Conditions
Output high voltage	All output pins	$V_{OH}$	$V_{CC} - 0.5$	—	—	V	$I_{OH} = -200\ \mu\text{A}$
			3.5	—	—	V	$I_{OH} = -1\ \text{mA}$
Output low voltage	All output pins	$V_{OL}$	—	—	0.4	V	$I_{OL} = 1.6\ \text{mA}$
	Ports 3 and 5		—	—	1.0	V	$I_{OL} = 8\ \text{mA}$
			—	—	1.2	V	$I_{OL} = 10\ \text{mA}$
Input capacitance	FWE	$C_{in}$	—	—	60	pF	$V_{in} = 0\ \text{V}$
	NMI, MD <sub>2</sub>		—	—	50	pF	$f = 1\ \text{MHz}$
	All input pins except FWE, NMI, and MD <sub>2</sub>		—	—	20	pF	$T_a = 25^\circ\text{C}$
Current dissipation	Normal operation	$I_{CC}$	—	65	100	mA	$f = 16\ \text{MHz}$
	Sleep mode		—	40	60	mA	$f = 16\ \text{MHz}$
	Standby mode		—	0.01	5.0	$\mu\text{A}$	$T_a \leq 50^\circ\text{C}$
			—	—	20.0	$\mu\text{A}$	$50^\circ\text{C} < T_a$
Analog power supply current	During A/D conversion	$AI_{CC}$	—	1.2	2.0	mA	
	Idle		—	0.01	5.0	$\mu\text{A}$	
Reference current	During A/D conversion	$AI_{CC}$	—	0.6	0.8	mA	
	Idle		—	0.01	5.0	$\mu\text{A}$	



**Table 23-2 DC Characteristics (cont)**

Conditions:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$ , ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Min	Typ	Max	Unit	Conditions
RAM standby voltage	$V_{RAM}$	2.0	—	—	V	

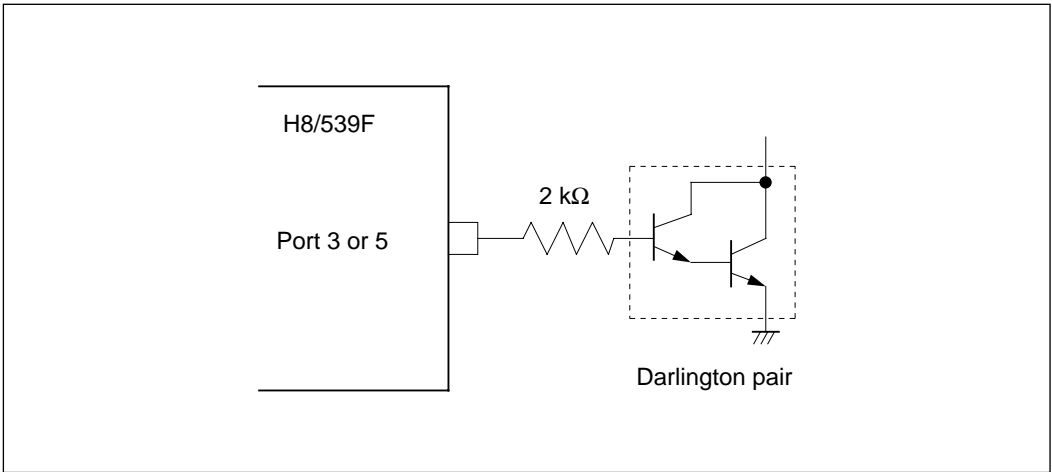
- Notes: 1. Never leave the  $AV_{CC}$ ,  $AV_{SS}$ , and  $V_{REF}$  pins open. If the A/D converter is not used, connect  $AV_{CC}$  and  $V_{REF}$  to  $V_{CC}$  and connect  $AV_{SS}$  to  $V_{SS}$ .
2. Current dissipation values are for  $V_{IHmin} = V_{CC} - 0.5\text{ V}$  and  $V_{ILmax} = 0.5\text{ V}$  with all output pins unloaded and the on-chip pull-up transistors in the off state. (When read is operating in flash memory).
3. The current dissipation value in flash memory programming and erasing ( $T_a = 0\text{ to }+70^\circ\text{C}$ ) is 20 mA (max.) (preliminary) higher than in normal operation.

**Table 23-3 Permissible Output Currents (H8/539F S Mask Model)**

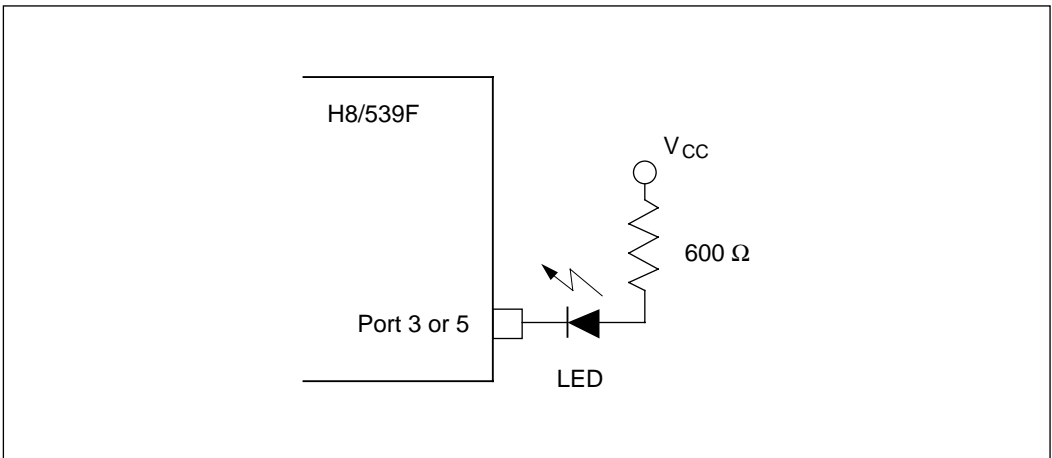
Condition:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$  ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Condition			Unit	
		Min	Typ	Max		
Permissible output low current (per pin)	Ports 3 and 5	$I_{OL}$	—	—	10	mA
	Other output pins		—	—	2.0	mA
Permissible output low current (total)	Total of 14 pins in ports 3 and 5	$\Sigma I_{OL}$	—	—	40	mA
	Total of all output pins, including the above		—	—	80	mA
Permissible output high current (per pin)	All output pins	$-I_{OH}$	—	—	2.0	mA
Permissible output high current	Total of all output pins	$\Sigma -I_{OH}$	—	—	25	mA

- Notes: 1. To protect chip reliability, do not exceed the output current values in table 23-3.  
 2. When driving a Darlington pair or LED, always insert a current-limiting resistor in the output line, as shown in figures 23-1 and 23-2.



**Figure 23-1 Darlington Pair Drive Circuit (Example)**



**Figure 23-2 LED Drive Circuit (Example)**

### 23.2.2 AC Characteristics (H8/539F S Mask Model)

The AC characteristics of the H8/539F are described below. Bus timing parameters are listed in table 23-4. Control signal timing parameters are listed in table 23-5. Timing parameters of the on-chip peripheral modules are listed in table 23-6.

**Table 23-4 Bus Timing**

Condition:  $V_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $V_{REF} = 5.0 \text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0 \text{ V}$ ,  $T_a = -20 \text{ to } +75^\circ\text{C}$  (regular specifications),  $T_a = -40 \text{ to } +85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Condition		Unit	Test Conditions
		Min	Max		
		<b>16 MHz</b>			
Clock cycle time	$t_{CYC}$	62.5	500	ns	Fig. 23-4, Fig. 23-5
Clock low pulse width	$t_{CL}$	20	—	ns	
Clock high pulse width	$t_{CH}$	20	—	ns	
Clock rise time	$t_{Cr}$	—	15	ns	
Clock fall time	$t_{Cf}$	—	15	ns	
Address delay time	$t_{AD}$	—	25	ns	
Address hold time	$t_{AH}$	10	—	ns	
Address strobe delay time 1	$t_{ASD1}$	—	25	ns	
Address strobe delay time 2	$t_{ASD2}$	—	25	ns	
Read strobe delay time 1	$t_{RDD1}$	—	25	ns	
Read strobe delay time 2	$t_{RDD2}$	—	25	ns	
Write strobe delay time 1	$t_{WRD1}$	—	25	ns	
Write strobe delay time 2	$t_{WRD2}$	—	25	ns	
Write strobe delay time 3	$t_{WRD3}$	—	25	ns	
Write data strobe pulse width 1	$t_{WRW1}$	50	—	ns	
Write data strobe pulse width 2	$t_{WRW2}$	70	—	ns	
Address setup time 1	$t_{AS1}$	10	—	ns	
Address setup time 2	$t_{AS2}$	10	—	ns	
Address setup time 3	$t_{AS3}$	30	—	ns	
Read data setup time	$t_{RDS}$	20	—	ns	
Read data hold time	$t_{RDH}$	0	—	ns	
Read data access time 1	$t_{ACC1}$	—	60	ns	
Read data access time 2	$t_{ACC2}$	—	120	ns	

**Table 23-4 Bus Timing (cont)**

Condition:  $V_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $V_{REF} = 5.0 \text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0 \text{ V}$ ,  $T_a = -20$  to  $+75^\circ\text{C}$  (regular specifications),  $T_a = -40$  to  $+85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Condition		Unit	Test Conditions
		Min	Max		
Write data delay time	$t_{WDD}$	—	55	ns	Fig. 23-4, Fig. 23-5
Write data setup time	$t_{WDS}$	2	—	ns	
Write data hold time	$t_{WDH}$	10	—	ns	
Wait setup time	$t_{WTS}$	25	—	ns	Fig. 23-6
Wait hold time	$t_{WTH}$	10	—	ns	
Bus request setup time	$t_{BRQS}$	30	—	ns	Fig. 23-9
Bus acknowledge delay time 1	$t_{BACD1}$	—	30	ns	
Bus acknowledge delay time 2	$t_{BACD2}$	—	30	ns	
Bus-floating delay time	$t_{BZD}$	—	$t_{BACD1}$	ns	

**Table 23-5 Control Signal Timing (H8/539F S Mask Model)**

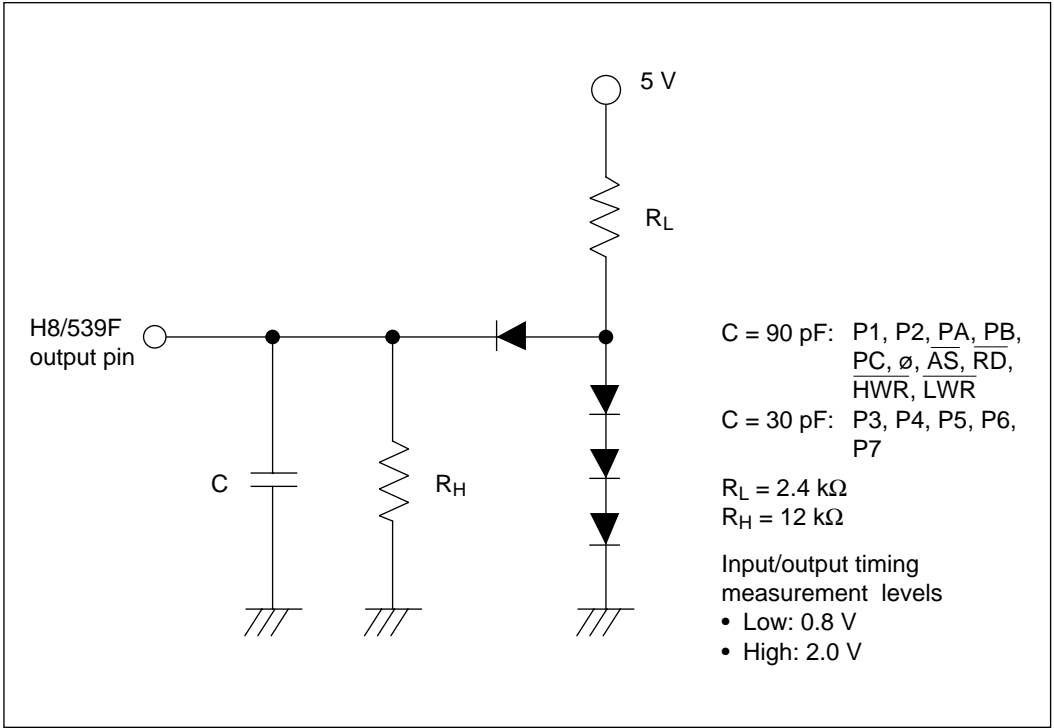
Condition:  $V_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $V_{REF} = 5.0 \text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0 \text{ V}$ ,  $T_a = -20 \text{ to } +75^\circ\text{C}$  (regular specifications),  $T_a = -40 \text{ to } +85^\circ\text{C}$  (wide-range specifications)

Item	Symbol	Condition		Unit	Test Conditions
		Min	Max		
RES setup time	$t_{RESS}$	200	—	ns	Fig. 23-7
RES pulse width	$t_{RESW}$	20	—	$t_{cyc}$	
Mode programming setup time	$t_{MDS}$	4.0	—	$t_{cyc}$	
NMI setup time	$t_{NMIS}$	150	—	ns	Fig. 23-8
NMI hold time	$t_{NMIH}$	10	—	ns	
$\overline{IRQ}_0$ setup time	$t_{IRQ0S}$	50	—	ns	
$\overline{IRQ}_{1-3}$ setup time	$t_{IRQ1S}$	50	—	ns	
$\overline{IRQ}_{1-3}$ hold time	$t_{IRQ1H}$	10	—	ns	
NMI pulse width (for recovery from software standby mode)	$t_{NMIW}$	200	—	ns	
Clock oscillator settling time at reset (crystal)	$t_{OSC1}$	20	—	ms	Fig. 23-10
Clock oscillator settling time in software standby (crystal)	$t_{OSC2}$	10	—	ms	Fig. 21-1
External clock output settling delay time (When inputting external clock from the EXTAL pin)	$t_{DEXT}$	500	—	$\mu\text{s}$	Fig. 23-11

**Table 23-6 Timing of On-Chip Supporting Modules (H8/539F S Mask Model)**

Condition:  $V_{CC} = 5.0\text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0\text{ V} \pm 10\%$ ,  $V_{REF} = 5.0\text{ V} \pm 10\%$ ,  $V_{SS} = AV_{SS} = 0\text{ V}$ ,  $T_a = -20\text{ to }+75^\circ\text{C}$  (regular specifications),  $T_a = -40\text{ to }+85^\circ\text{C}$  (wide-range specifications)

Item	Item	Symbol	16 MHz		Unit		
			Min	Max			
IPU	Timer output delay time	$t_{TODD}$	—	100	ns	Fig. 23-14	
	Timer input setup time	$t_{TICS}$	50	—	ns		
	Timer clock input setup time	$t_{TCKS}$	50	—	ns	Fig. 23-15	
	Timer clock pulse width	$t_{TCKW}$	1.5	—	$t_{CYC}$		
PWM	output delay time	$t_{PWDD}$	—	100	ns	Fig.23-13	
SCI	Input clock cycle	Asynchronous	$t_{SCYC}$	4	—	$t_{CYC}$	Fig. 23-16
		Clocked synchronous		6	—	$t_{CYC}$	
	Input clock pulse width	$t_{SCKW}$	0.4	0.6	$t_{SCYC}$		
	Transmit data delay time	$t_{TXD}$	—	100	ns	Fig. 23-17	
	Receive data setup time (clocked synchronous)	$t_{RXS}$	100	—	ns		
	Receive data hold time (clocked synchronous)	$t_{RXH}$	100	—	ns		
Ports	Output data delay time	$t_{PWD}$	—	50	ns	Fig. 23-12	
	Receive data setup time (clocked synchronous)	$t_{PRS}$	50	—	ns		
	Receive data hold time (clocked synchronous)	$t_{PRH}$	50	—	ns		



**Figure 23-3 Output Load Circuit**



### 23.2.3 A/D Conversion Characteristics (H8/539F S Mask Model)

Table 23-7 lists the A/D conversion characteristics of the H8/539F. Table 23-8 lists the permissible signal-source impedance for the A/D converter.

**Table 23-7 A/D Converter Characteristics**

Condition:  $V_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $AV_{CC} = 5.0 \text{ V} \pm 10\%$ ,  $V_{REF} = 5.0 \text{ V}$  ( $V_{REF} \leq AV_{CC}$ ),  
 $V_{SS} = AV_{SS} = 0 \text{ V}$ ,  $T_a = -20$  to  $+75^\circ\text{C}$  (regular specifications),  $T_a = -40$  to  $+85^\circ\text{C}$  (wide-range specifications)

Item	Condition			Unit
	Min	Typ	Max	
Resolution	10	10	10	Bits
Conversion time	—	—	8.38	$\mu\text{s}$
Analog input capacitance	—	—	20	pF
Nonlinearity error	—	—	$\pm 2.0$	LSB
Offset error	—	—	$\pm 2.0$	LSB
Full-scale error	—	—	$\pm 2.0$	LSB
Quantization error	—	—	$\pm 1/2$	LSB
Absolute accuracy	—	—	$\pm 2.5$	LSB

**Table 23-8 A/D Converter Characteristics: Allowable Signal-Source Impedance**

Item	Conditions	Min	Typ	Max	Unit
Allowable signal-source impedance	$8.38 \mu\text{s} \leq \text{conversion time} < 13.4 \mu\text{s}$	—	—	5	$\text{k}\Omega$
	$4.5 \text{ V} \leq AV_{CC} < 5.5 \text{ V}$	—	—	10	

### 23.2.4 Flash Memory Characteristics (H8/539F S Mask Model)

Table 23-9 lists the flash memory characteristics of the H8/539F.

**Table 23-9 Flash Memory Characteristics**

Conditions:  $V_{CC} = 4.5$  to  $5.5$  V,  $AV_{CC} = 4.5$  to  $5.5$  V,  $V_{REF} = 4.5$  to  $5.5$  V,  $V_{SS} = AV_{SS} = 0$  V,  
 $T_a = 0$  to  $+70^{\circ}\text{C}$  (regular specifications),  $T_a = 0$  to  $70^{\circ}\text{C}$  (wide-range specifications)

Item	Symbol	Min	Typ	Max	Unit	Test Conditions
Programming time*1, *2, *4	$t_P$	—	10	200	ms/32 byte	
Erase time*1, *3, *5	$t_E$	—	100	300	ms/block	
Number of writing / erasing count	$N_{WEC}$	—	—	100	times	
Programming time	Wait time after SWE-bit set*1	x	10	—	—	$\mu\text{s}$
	Wait time after PSU-bit set*1	y	50	—	—	$\mu\text{s}$
	Wait time after P-bit set*1, *4	z	—	—	200	$\mu\text{s}$
	Wait time after P-bit clear*1	$\alpha$	10	—	—	$\mu\text{s}$
	Wait time after PSU-bit clear*1	$\beta$	10	—	—	$\mu\text{s}$
	Wait time after PV-bit set*1	$\gamma$	4	—	—	$\mu\text{s}$
	Wait time after H'FF dummy write*1	$\varepsilon$	2	—	—	$\mu\text{s}$
	Wait time after PV-bit clear*1	$\eta$	4	—	—	$\mu\text{s}$
	Maximum number of programs*1, *4	N	—	—	1000	times
Erase time	Wait time after SWE-bit set*1	x	10	—	—	$\mu\text{s}$
	Wait time after ESU-bit set*1	y	200	—	—	$\mu\text{s}$
	Wait time after E-bit set*1, *5	z	—	—	5	ms
	Wait time after E-bit clear*1	$\alpha$	10	—	—	$\mu\text{s}$
	Wait time after ESU-bit clear*1	$\beta$	10	—	—	$\mu\text{s}$
	Wait time after EV-bit set*1	$\gamma$	20	—	—	$\mu\text{s}$
	Wait time after H'FF dummy write*1	$\varepsilon$	2	—	—	$\mu\text{s}$
	Wait time after EV-bit clear*1	$\eta$	5	—	—	$\mu\text{s}$
Maximum number of erasures*1, *5	N	—	—	60	times	

Notes: 1. Make the time settings in accordance with the program/erase algorithm.

2. The programming time for 32 bytes. (Indicates the total time for which the P bit in the flash memory control register (FLMCR) is set. The program/verify time is not included.)
3. The time required to erase one block. (Indicates the time for which the E bit in the flash memory control register (FLMCR) is set. The erase/verify time is not included.)

4. Programming time maximum value ( $t_P(\max)$ ) = wait time after P bit setting (z) x maximum number of writes (N)
5. Erase time maximum value ( $t_E(\max)$ ) = wait time after E bit setting (z) x maximum number of erases (N)

## 23.3 Operational Timing (H8/539F S Mask Model)

This section shows timing diagrams of H8/539F operations.

### 23.3.1 Bus Timing

This section gives the following bus timing diagrams:

1. Basic bus cycle: two-state access

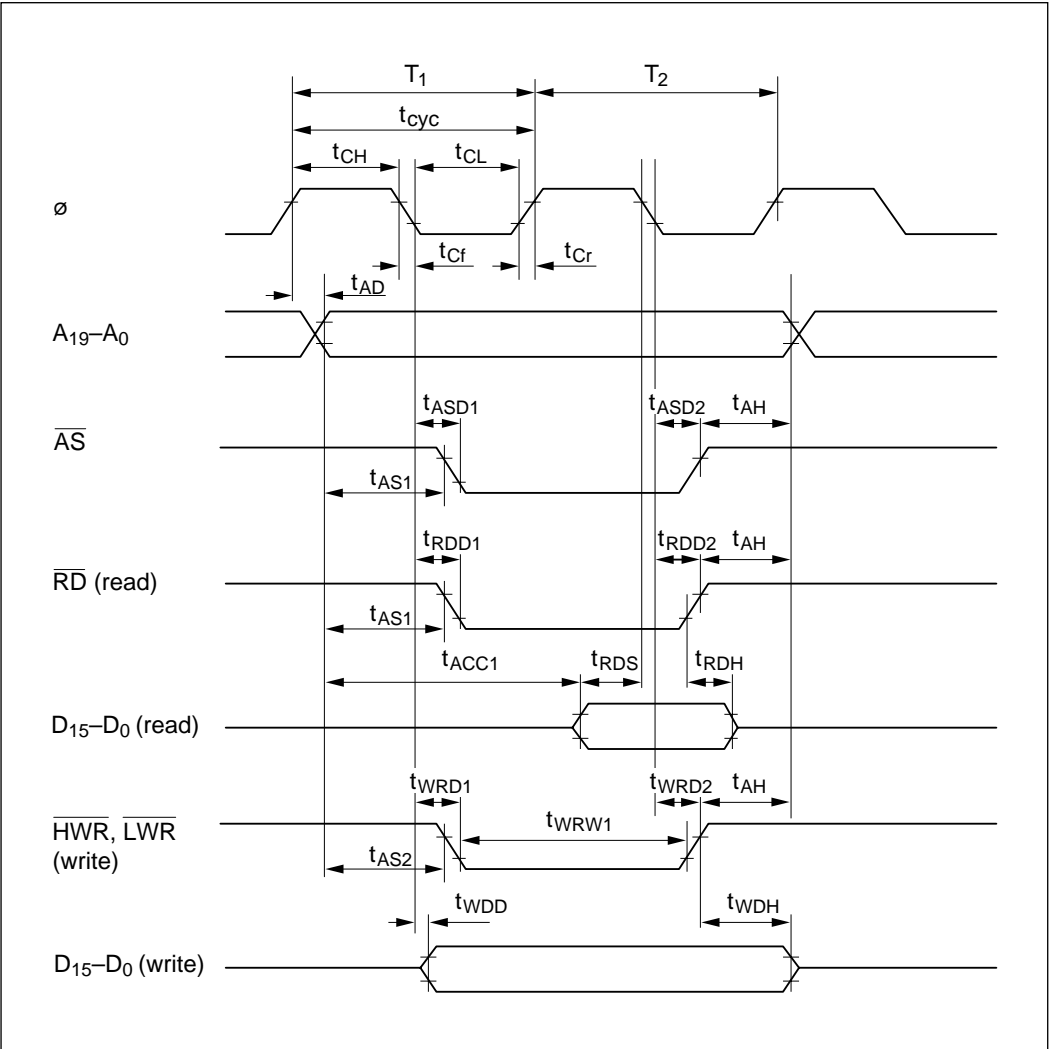
Figure 23-4 shows the timing of the external two-state access cycle.

2. Basic bus cycle: three-state access

Figure 23-5 shows the timing of the external three-state access cycle.

3. Basic bus cycle: three-state access with one wait state

Figure 23-6 shows the timing of the external three-state access cycle with one wait state inserted.



**Figure 23-4 Basic Bus Cycle: Two-State Access**

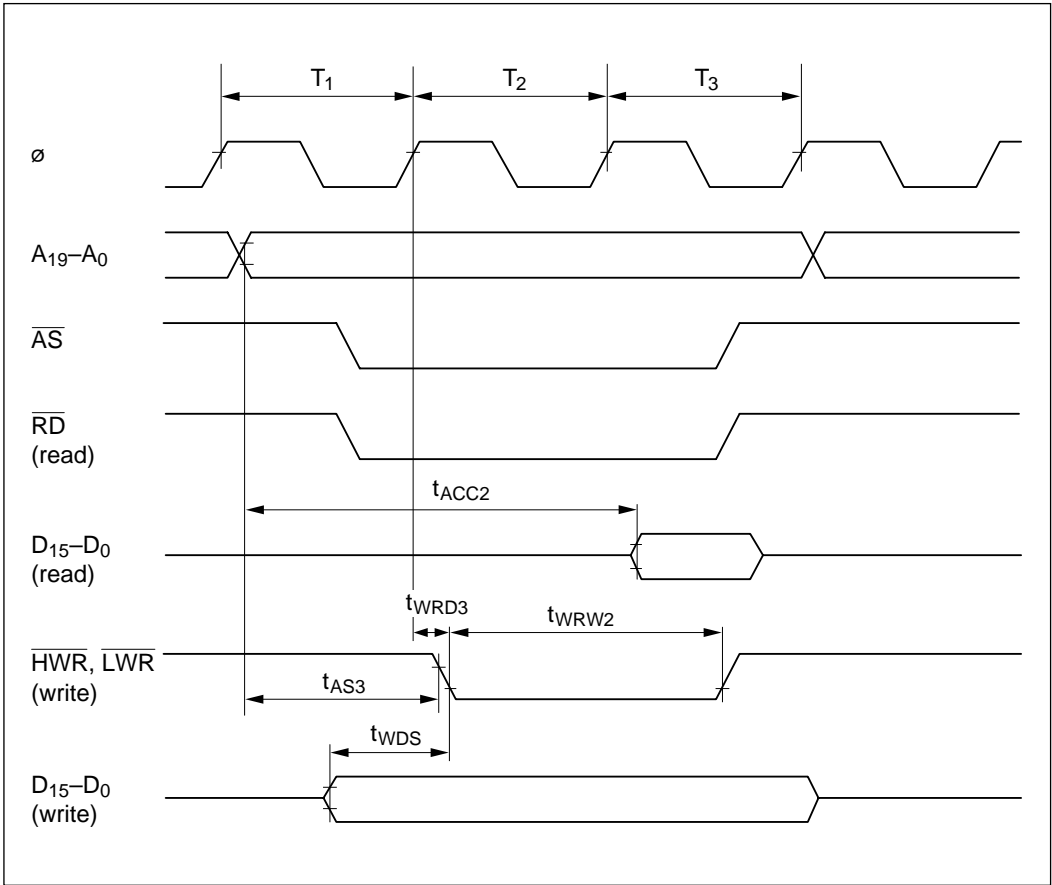
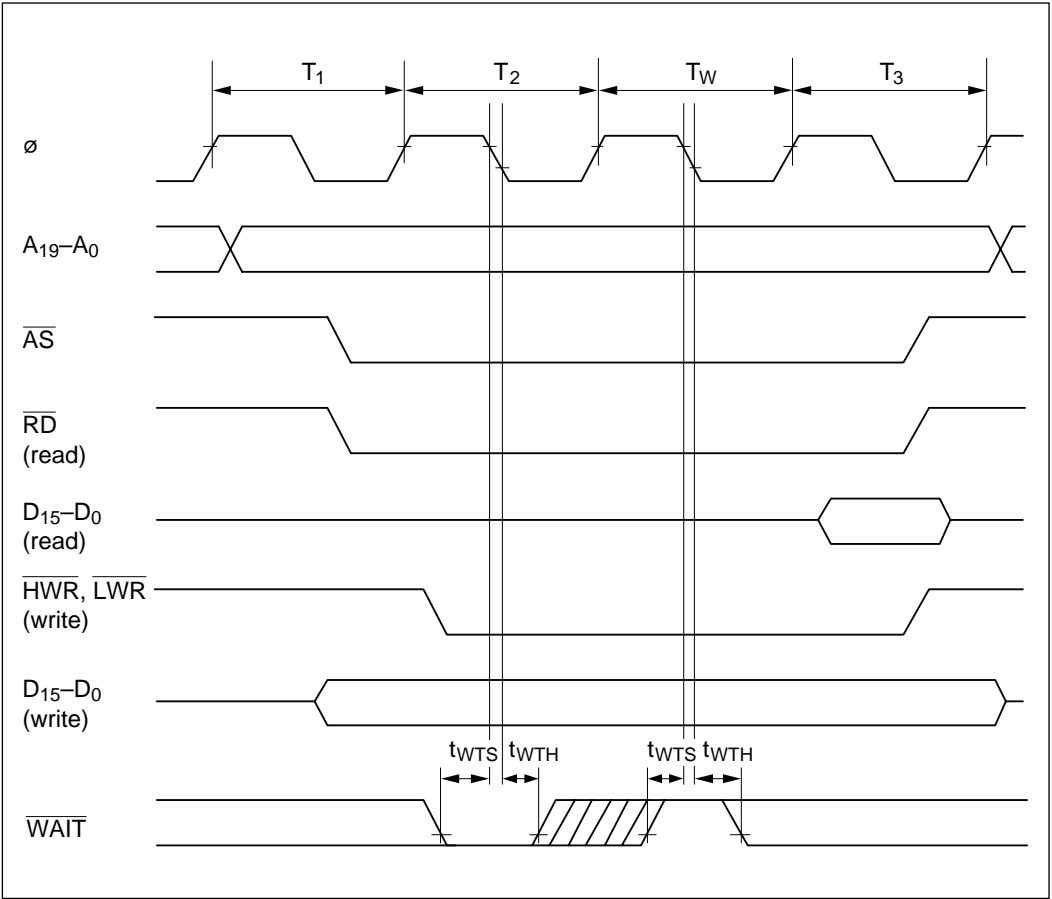


Figure 23-5 Basic Bus Cycle: Three-State Access



**Figure 23-6 Basic Bus Cycle: Three-State Access with One Wait State**

### 23.3.2 Control Signal Timing (H8/539F S Mask Model)

This section gives the following control signal timing diagrams:

1. Reset input timing

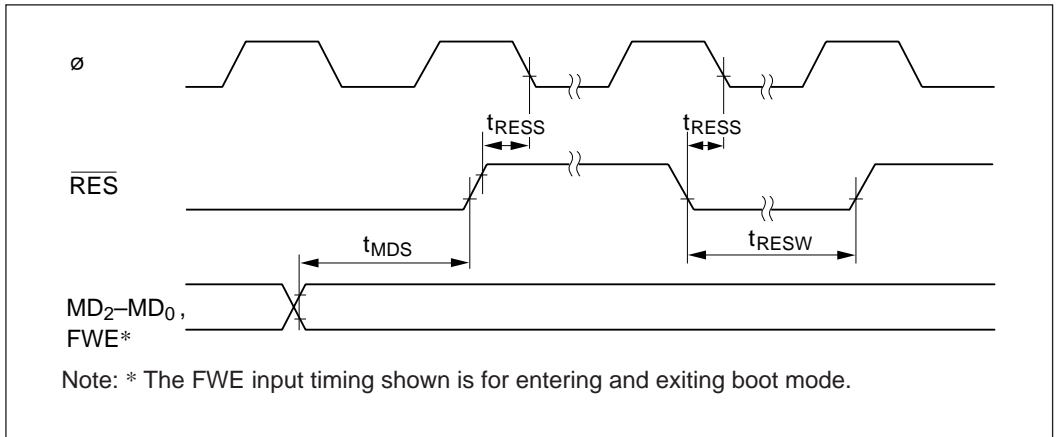
Figure 23-7 shows the reset input timing.

2. Interrupt input timing

Figure 23-8 shows the input timing for NMI,  $\overline{\text{IRQ}}_0$ , and  $\overline{\text{IRQ}}_1$  to  $\overline{\text{IRQ}}_3$ .

3. Bus-release mode timing

Figure 23-9 shows the bus-release mode timing.



**Figure 23-7 Reset Input Timing**



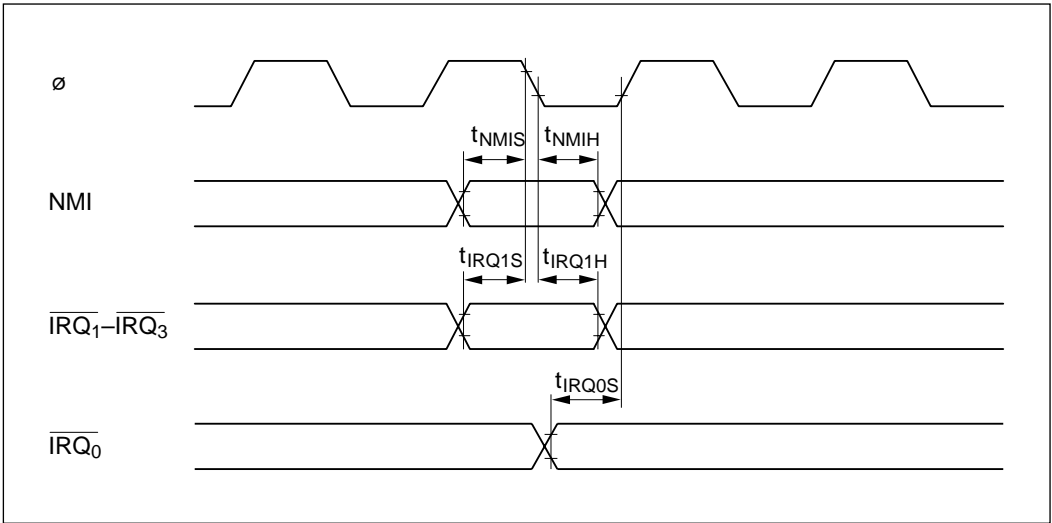


Figure 23-8 Interrupt Input Timing

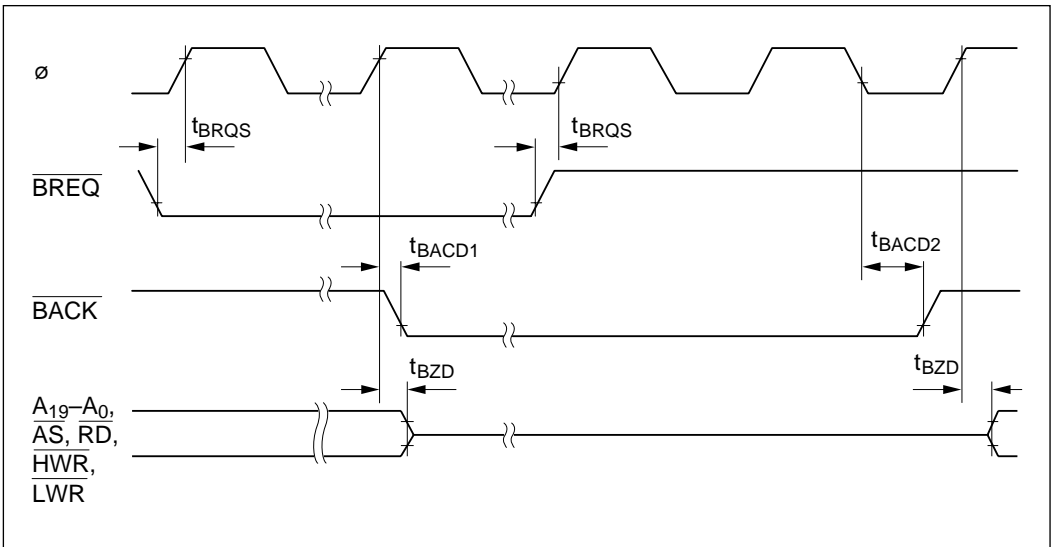


Figure 23-9 Bus-Release Mode Timing

### 23.3.3 Clock Timing (H8/539F S Mask Model)

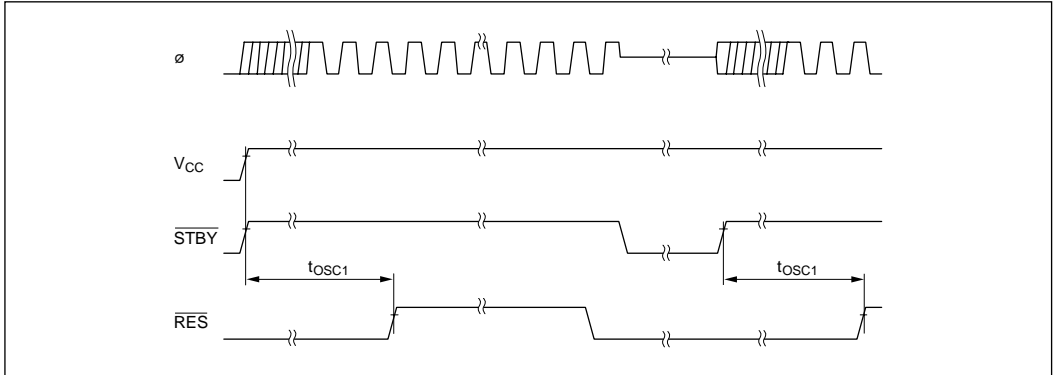
This section gives the following H8/539F clock timing diagram:

1. Oscillator settling timing

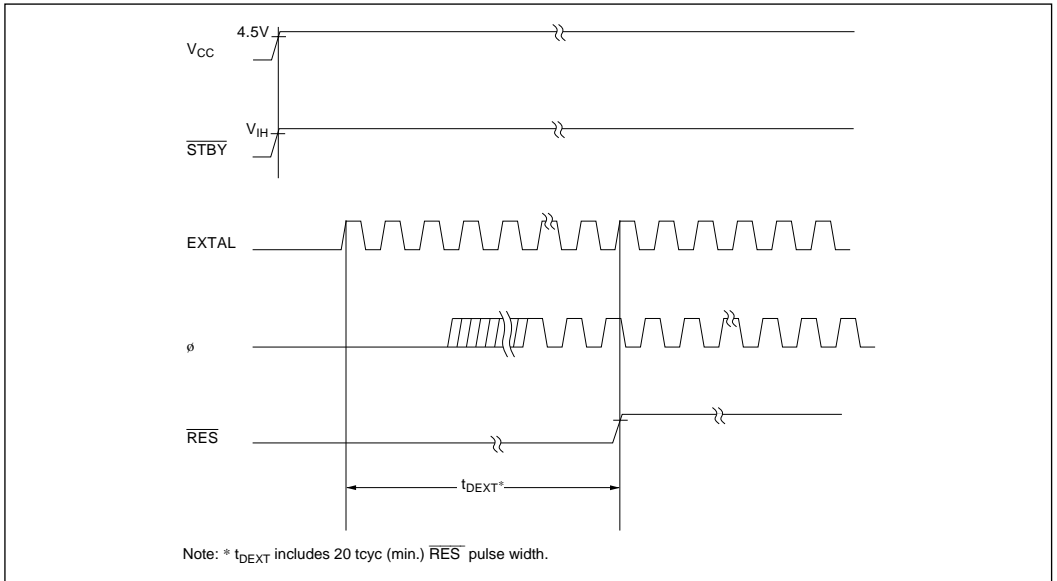
Figure 23-10 shows the oscillator settling timing.

2. External clock output settling delay timing

Figure 23-11 shows the external clock output settling delay timing.



**Figure 23-10 Oscillator Settling Timing**



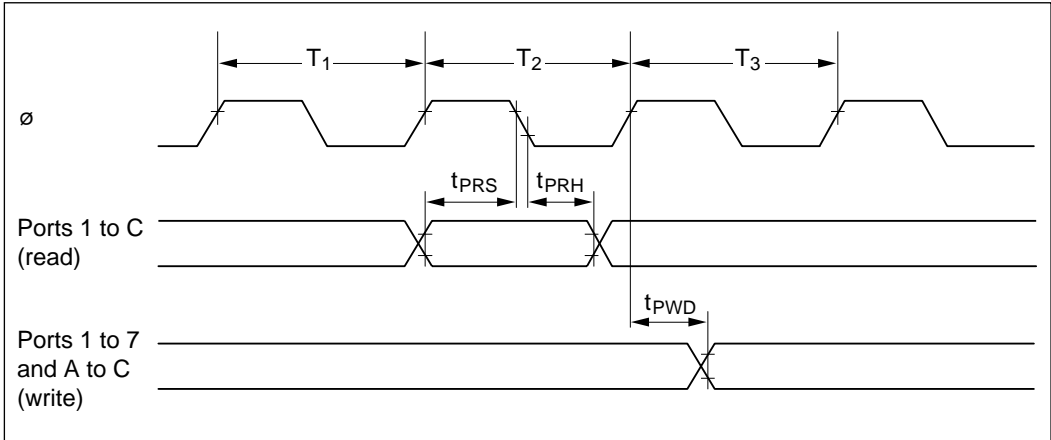
**Figure 23-11 External Clock Output Settling Delay Timing**

### 23.3.4 I/O Port Timing (H8/539F S Mask Model)

This section gives the following H8/539F I/O port input/output timing diagram:

#### 1. I/O port input/output timing

Figure 23-12 shows the I/O port input/output timing.



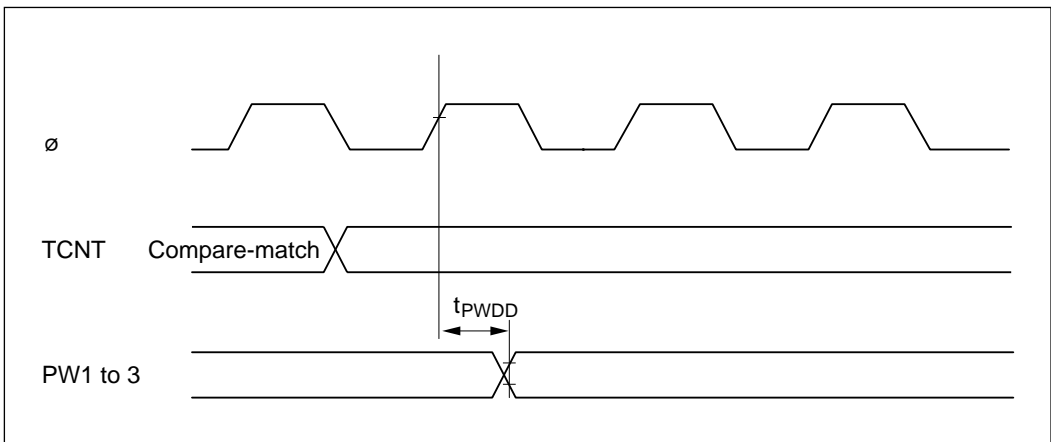
**Figure 23-12 I/O Port Input/Output Timing**

### 23.3.5 PWM Timer Output Timing

This section gives the following H8/539F PWM timer output timing diagram.

#### 1. PWM timer output timing

Figure 23-13 shows the PWM timer output timing.



**Figure 23-13 PWM Timer Output Timing**

### 23.3.6 IPU Timing (H8/569F S Mask Model)

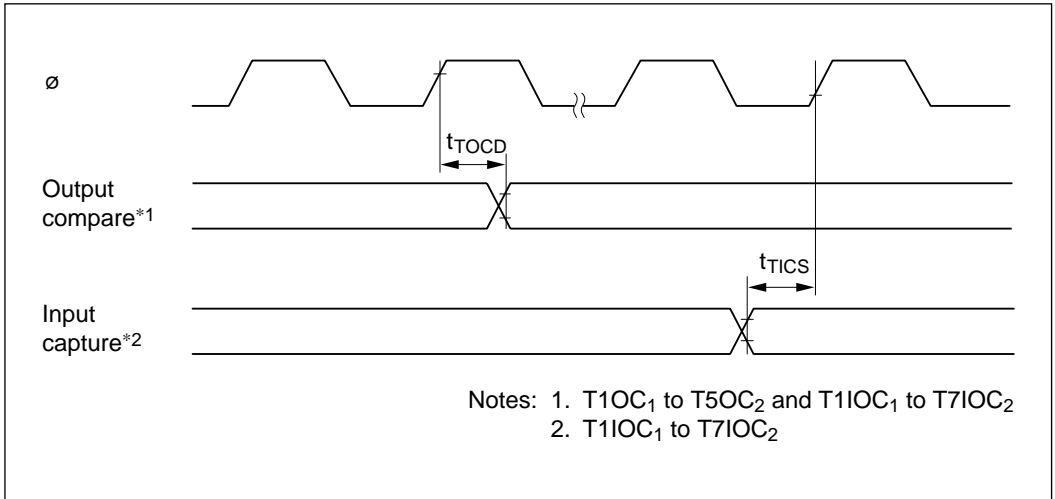
This section gives the following H8/539F IPU timing diagrams:

1. IPU input/output timing

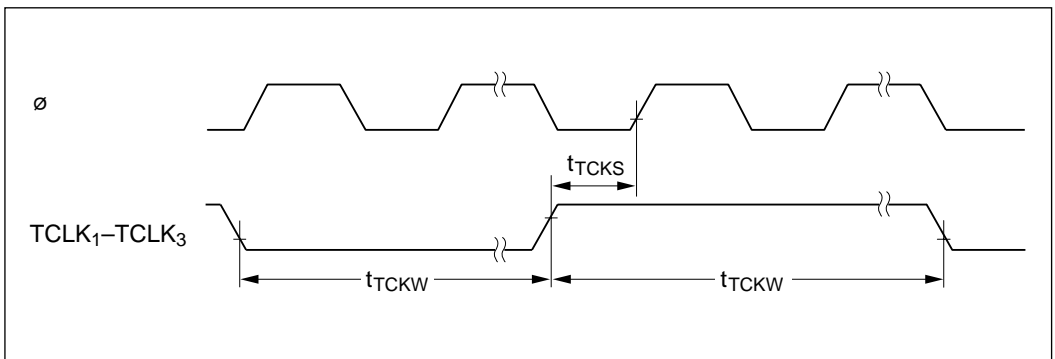
Figure 23-14 shows the IPU input/output timing.

2. IPU external clock input timing

Figure 23-15 shows the IPU external clock input timing.



**Figure 23-14 IPU Input/Output Timing**



**Figure 23-15 IPU Clock Input Timing**

### 23.3.7 SCI Input/Output Timing (H8/569F S Mask Model)

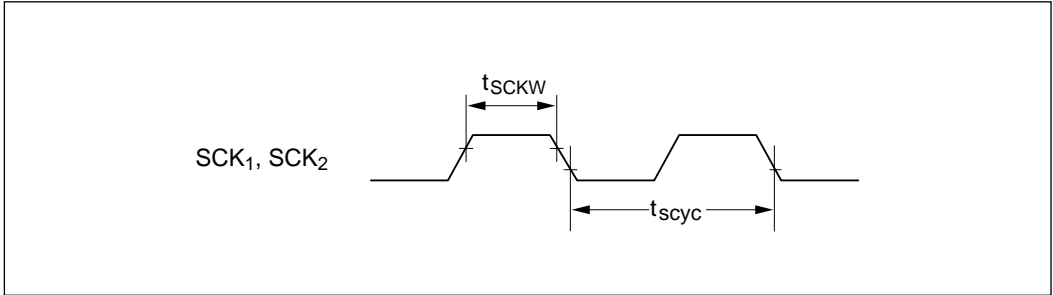
This section gives the following H8/539F SCI timing diagrams:

1. SCI input clock timing

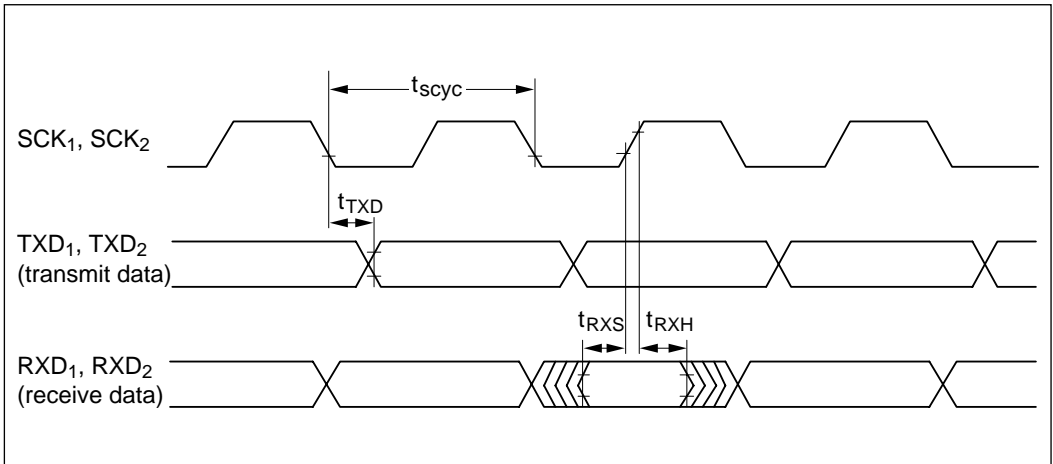
Figure 23-16 shows the SCI input clock timing.

2. SCI input/output timing (clocked synchronous mode)

Figure 23-17 shows the SCI input/output timing in clocked synchronous mode.



**Figure 23-16 SCK Input Clock Timing**



**Figure 23-17 SCI Input/Output Timing  
(Clocked Synchronous Mode)**

# Appendix A Instruction Set

## A.1 Instruction List

### Operand Notation

Rd	General register (destination)
Rs	General register (source)
Rn	General register
(EAd)	Destination operand
(EAs)	Source operand
CCR	Condition code register
N	N (negative) bit in CCR
Z	Z (zero) bit in CCR
V	V (overflow) bit in CCR
C	C (carry) bit in CCR
CR	Control register
PC	Program counter
CP	Code page register
SP	Stack pointer
FP	Frame pointer
#IMM	Immediate data
disp	Displacement
+	Add
-	Subtract
×	Multiply
÷	Divide
^	Logical AND
∨	Logical OR
⊕	Exclusive logical OR
→	Move
↔	Exchange
¬	Logical NOT

## Condition Code Notation

---

↕	Changed according to execution result
0	Cleared to 0
—	Previous value remains unchanged
△	Varies depending on conditions

---

Mnemonic		Operation	Size	CCR Bits				
			B/W	N	Z	V	C	
Data transfer instructions	MOV:G	(EAs) → Rd Rs → (EAd) #IMM → (EAd)	B/W	↑	↑	0	—	
	MOV:E	#IMM → Rd (short format)	B	↑	↑	0	—	
	MOV:F	@(d:8,FP) → Rd Rs → @(d:8,FP) (short format)	B/W	↑	↑	0	—	
	MOV:I	#IMM → Rd (short format)	W	↑	↑	0	—	
	MOV:L	(@aa:8) → Rd (short format)	B/W	↑	↑	0	—	
	MOV:S	Rs → (@aa:8) (short format)	B/W	↑	↑	0	—	
	LDM	@SP+ → Rn (register list)	W	—	—	—	—	
	STM	Rn (register list) → @-SP	W	—	—	—	—	
	XCH	Rs ↔ Rd	W	—	—	—	—	
	SWAP	Rd (upper byte) ↔ Rd (lower byte)	B	↑	↑	0	—	
	(MOVTPE)	Not available in H8/539F						
	(MOVFPPE)	Not available in H8/539F						
Arithmetic instructions	ADD:G	Rd+ (EAs) → Rd	B/W	↑	↑	↑	↑	
	ADD:Q	(EAd) + #IMM → (EAd) (#IMM = ±1, ±2) (short format)	B/W	↑	↑	↑	↑	
	ADDS	Rd+ (EAs) → Rd (Rd is always word size)	B/W	—	—	—	—	
	ADDX	Rd+ (EAs) + C → Rd	B/W	↑	↑	↑	↑	
	DADD	(Rd) 10+ (Rs) 10+ C → (Rd) 10	B	—	↑	—	↑	
	SUB	Rd- (EAs) → Rd	B/W	↑	↑	↑	↑	
	SUBS	Rd- (EAs) → Rd	B/W	—	—	—	—	
	SUBX	Rd- (EAs) - C → Rd	B/W	↑	↑	↑	↑	
	DSUB	(Rd) 10- (Rs) 10- C → (Rd) 10	B	—	↑	—	↑	
	MULXU	Rd × (EAs) → Rd (unsigned)	8 × 8 16 × 16	B/W	↑	↑	0	0
	DIVXU	Rd ÷ (EAs) → Rd (unsigned)	16 ÷ 8 32 ÷ 16	B/W	↑	↑	↑	0
CMP:G	Rd - (EAs), set CCR flags (EAd) - #IMM, set CCR flags	B/W	↑	↑	↑	↑		



Mnemonic		Operation	Size	CCR Bits			
			B/W	N	Z	V	C
Arithmetic instructions	CMP:E	Rd – #IMM, set CCR flags (short format)	B	↕	↕	↕	↕
	CMP:I	Rd – #IMM, set CCR flags (short format)	W	↕	↕	↕	↕
	EXTS	(<Bit 7> of <Rd>) → (<bits 15 to 8> of <Rd>)	B	↕	↕	0	0
	EXTU	0 → (<bits 15 to 8> of <Rd>)	B	0	↕	0	0
	TST	(EAd) – 0, set CCR flags	B/W	↕	↕	0	0
	NEG	0 – (EAd) → (EAd)	B/W	↕	↕	0	↕
	CLR	0 → (EAd)	B/W	0	1	0	0
	TAS	(EAd) – 0, set CCR flags (1) 2 → (<bit 7> of <EAd>)	B	↕	↕	0	0
Shift instructions	SHAL		B/W	↕	↕	↕	↕
	SHAR		B/W	↕	↕	0	↕
	SHLL		B/W	↕	↕	0	↕
	SHLR		B/W	0	↕	0	↕
	ROTL		B/W	↕	↕	0	↕
	ROTR		B/W	↕	↕	0	↕
	ROTXL		B/W	↕	↕	0	↕
	ROTXR		B/W	↕	↕	0	↕

Mnemonic		Operation	Size	CCR Bits			
			B/W	N	Z	V	C
Logic instructions	AND	$Rd \wedge (EAs) \rightarrow Rd$	B/W	$\updownarrow$	$\updownarrow$	0	—
	OR	$Rd \vee (EAs) \rightarrow Rd$	B/W	$\updownarrow$	$\updownarrow$	0	—
	XOR	$Rd \oplus (EAs) \rightarrow Rd$	B/W	$\updownarrow$	$\updownarrow$	0	—
	NOT	$\neg(EAd) \rightarrow (EAd)$	B/W	$\updownarrow$	$\updownarrow$	0	—
Bit manipulation instructions	BSET	$\neg(\langle \text{Bit No.} \rangle \text{ of } \langle EAd \rangle) \rightarrow Z$ $1 \rightarrow (\langle \text{Bit No.} \rangle \text{ of } \langle EAd \rangle)$	B/W	—	$\updownarrow$	—	—
	BCLR	$\neg(\langle \text{Bit No.} \rangle \text{ of } \langle EAd \rangle) \rightarrow Z$ $0 \rightarrow (\langle \text{Bit No.} \rangle \text{ of } \langle EAd \rangle)$	B/W	—	$\updownarrow$	—	—
	BTST	$\neg(\langle \text{Bit No.} \rangle \text{ of } \langle EAd \rangle) \rightarrow Z$	B/W	—	$\updownarrow$	—	—
	BNOT	$\neg(\langle \text{Bit No.} \rangle \text{ of } \langle EAd \rangle) \rightarrow Z$ $\rightarrow (\langle \text{Bit No.} \rangle \text{ of } \langle EAd \rangle)$	B/W	—	$\updownarrow$	—	—
Branch instructions	Bcc	If condition is true then PC + disp $\rightarrow$ PC else next;	—	—	—	—	—
		<b>Mnemonic</b> <b>Description</b> <b>Condition</b>					
		BRA (BT)      Always (true)      True					
		BRN (BF)      Never (false)      False					
		BHI      High $C \vee Z = 0$					
		BLS      Low or same $C \vee Z = 1$					
		Bcc (BHS)      Carry clear (high or same) $C = 0$					
		BCS (BLO)      Carry set (low) $C = 1$					
		BNE      Not equal $Z = 0$					
		BEQ      Equal $Z = 1$					
		BVC      Overflow clear $V = 0$					
		BVS      Overflow set $V = 1$					
		BPL      Plus $N = 0$					
		BMI      Minus $N = 1$					
		BGE      Greater or equal $N \oplus V = 0$					
		BLT      Less than $N \oplus V = 1$					
		BGT      Greater than $Z \vee (N \oplus V) = 0$					
	BLE      Less or equal $Z \vee (N \oplus V) = 1$						

Mnemonic		Operation	Size	CCR Bits															
			B/W	N	Z	V	C												
Branch instructions	JMP	Effective address → PC	—	—	—	—	—												
	PJMP	Effective address → CP, PC	—	—	—	—	—												
	BSR	PC → @ – SP PC + disp → PC	—	—	—	—	—												
	JSR	PC → @ – SP Effective address → PC	—	—	—	—	—												
	PJSR	PC → @ – SP CP → @ – SP Effective address → CP, PC	—	—	—	—	—												
	RTS	@SP + → PC	—	—	—	—	—												
	PRTS	@SP + → CP @SP + → PC	—	—	—	—	—												
	RTD	@SP + → PC SP + #IMM → SP	—	—	—	—	—												
	PRTD	@SP + → CP @SP + → PC SP + #IMM → SP	—	—	—	—	—												
	SCB SCB/F SCB/NE SCB/EQ	If condition is true then next; else Rn – 1 → Rn; If Rn = –1 then next else PC + disp → PC;	—	—	—	—	—												
		<table border="1"> <thead> <tr> <th>Mnemonic</th> <th>Description</th> <th>Condition</th> </tr> </thead> <tbody> <tr> <td>SCB/F</td> <td></td> <td>False</td> </tr> <tr> <td>SCB/NE</td> <td>Not equal</td> <td>Z = 0</td> </tr> <tr> <td>SCB/EQ</td> <td>Equal</td> <td>Z = 1</td> </tr> </tbody> </table>	Mnemonic	Description	Condition	SCB/F		False	SCB/NE	Not equal	Z = 0	SCB/EQ	Equal	Z = 1					
Mnemonic	Description	Condition																	
SCB/F		False																	
SCB/NE	Not equal	Z = 0																	
SCB/EQ	Equal	Z = 1																	

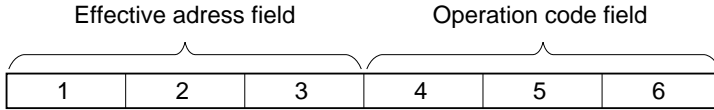
Mnemonic		Operation	Size	CCR Bits				
			B/W	N	Z	V	C	
System control instructions	TRAPA	PC → @-SP (If Max. mode then CP → @-SP) SR → @-SP (If Max. mode then <vector> → CP) <Vector> → PC	—	—	—	—	—	—
	TRAP/VS	If V bit = 1 then TRAP else next;	—	—	—	—	—	
	RTE	@SP + → SR (If Max. mode then @SP + → CP) @SP + → PC	—	↕	↕	↕	↕	
	LINK	FP (R6) → @ - SP SP → FP (R6) SP + #IMM → SP	—	—	—	—	—	
	UNLK	FP (R6) → SP @SP + → FP	—	—	—	—	—	
	SLEEP	Normal operating mode → power-down state	—	—	—	—	—	
	LDC	(EAs) → CR	B/W*	△	△	△	△	
	STC	CR → (EAd)	B/W*	—	—	—	—	
	ANDC	CR ∧ #IMM → CR	B/W*	△	△	△	△	
	ORC	CR ∨ #IMM → CR	B/W*	△	△	△	△	
	XORC	CR ⊕ #IMM → CR	B/W*	△	△	△	△	
	NOP	PC + 1 → PC	—	—	—	—	—	

Note: \* Depends on the control register.

## A.2 Machine-Language Instruction Codes

Tables A-1 (a) to (d) indicate the machine-language code for each instruction.

**How to Read Tables A-1 (a) to (d):** The general format consists of an effective address (EA) field followed by an operation code (OP) field.



Bytes 2, 3, 5, and 6 are not present in all instructions.

Instruction		Effective Address (EA) Field			Operation Code (OP) Field						
		Addressing Mode	1	2	3	4	5	6			
Data transfer	MOV:G.B <EAs>,Rd	Rn	1010Szrrr								
	MOV:G.W <EAs>,Rd	@Rn	1101Szrrr								
	MOV:G.B Rs,<EAd>	@(d:8,Rn)	1110Szrrr	disp							
	MOV:G.W Rs,<EAd>	@(d:16,Rn)	1111Szrrr	dispH	dispL						
	MOV:G.B #xx:8,<EAd>	@-Rn	1011Szrrr								
	MOV:G.W #xx:16,<EAd>	@Rn+	1100Szrrr								
	LDM.W @SP+, <register list>	@aa:8	0000Sz101	Address							
	@aa:16	0001Sz101	Address (high)	Address (low)							
	#xx:8	00000100	Data								
	#xx:16	00001100	Data (high)	Data (low)							

Byte length of instruction Shading indicates addressing modes that cannot be specified in the instruction.

In special-format instructions the operation code field precedes the effective address field.

The following notation is used in the tables:

- Sz: operand size designation (byte or word)

Sz = 0: byte size

Sz = 1: word size

- rrr: general register number field

rrr	Sz = 0 (byte)			Sz = 1 (word)	
	15	8	7	15	0
000	Not used		R0	R0	
001	Not used		R1	R1	
010	Not used		R2	R2	
011	Not used		R3	R3	
100	Not used		R4	R4	
101	Not used		R5	R5	
110	Not used		R6	R6	
111	Not used		R7	R7	

- ccc: control register number field

ccc	Sz = 0 (byte)			Sz = 1 (word)	
	15	8	7	15	0
000	(disallowed*)			SR	
001	Not used		CCR	(disallowed*)	
010	(disallowed*)			(disallowed*)	
011	Not used		BR	(disallowed*)	
100	Not used		EP	(disallowed*)	
101	Not used		DP	(disallowed*)	
110	(disallowed*)			(disallowed*)	
111	Not used		TP	(disallowed*)	

Note: \* Do not use combinations marked as disallowed, since they may cause incorrect operation.

- d: direction of transfer

d = 0: load

d = 1: store

- Register list: a byte in which bits indicate general registers as follows.

Bit	7	6	5	4	3	2	1	0
	R7	R6	R5	R4	R3	R2	R1	R0

- #VEC: four bits specifying a vector number from 0 to 15. These vector numbers designate vector addresses as follows:

#VEC	Vector Address	
	Minimum Mode	Maximum Mode
0	H'0020–H'0021	H'0040–H'0043
1	H'0022–H'0023	H'0044–H'0047
2	H'0024–H'0025	H'0048–H'004B
3	H'0026–H'0027	H'004C–H'004F
4	H'0028–H'0029	H'0050–H'0053
5	H'002A–H'002B	H'0054–H'0057
6	H'002C–H'002D	H'0058–H'005B
7	H'002E–H'002F	H'005C–H'005F
8	H'0030–H'0031	H'0060–H'0063
9	H'0032–H'0033	H'0064–H'0067
A	H'0034–H'0035	H'0068–H'006B
B	H'0036–H'0037	H'006C–H'006F
C	H'0038–H'0039	H'0070–H'0073
D	H'003A–H'003B	H'0074–H'0077
E	H'003C–H'003D	H'0078–H'007B
F	H'003E–H'003F	H'007C–H'007F

## Examples of Machine-Language Instruction Codes

### Example 1: ADD:G.B @R0, R1

	<b>EA Field</b>	<b>OP Field</b>	<b>Remarks</b>
Table A-1	1101Szrrr	00100 <sub>r<sub>a</sub>r<sub>a</sub>r<sub>a</sub>r<sub>d</sub></sub>	ADD:G.B @Rs, Rd instruction code
Instruction code	11010000 H'D021	00100001	Sz = 0 (byte) Rs = R0, Rd = R1

### Example 2: ADD:G.W @H'11:8, R1

	<b>EA Field</b>	<b>OP Field</b>	<b>Remarks</b>
Table A-1	0000Sz101	00010001	00100 <sub>r<sub>a</sub>r<sub>a</sub>r<sub>a</sub>r<sub>d</sub></sub> ADD:G.W @aa:8, Rd instruction code
Instruction code	00001101 H'0D1121	00010001	00100001 Sz = 1 (word) aa = H'11, Rd = R1



**Table A-1 (a) Machine-Language Instruction Codes [General Format] (1)**

Instruction		Effective Address (EA) Field										Operation Code (OP) Field			
		Address- sing Mode	1	2	3								4	5	6
		Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16				
Data transfer	MOV:G.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		10000	RdRdRd		
	MOV:G.W <EAs>,Rd	2	2	3	4	2	2	3	4	4		10000	RdRdRd		
	MOV:G.B Rs,<EAd>		2	3	4	2	2	3	4			10010	RsfRd		
	MOV:G.W Rs,<EAd>		2	3	4	2	2	3	4			10010	RsfRd		
	MOV:G.B #xx:8,<EAd>		3	4	5	3	3	4	5			00000110	Data		
	MOV:G.W #xx:8,<EAd>		3	4	5	3	3	4	5			00000110	Data		
	MOV:G.W #xx:16,<EAd>		4	5	6	4	4	5	6			00000111	Data (high)	Data (low)	
	LDM.W @SP+, <register list>						2					00000010	Register list		
	STM.W <register list> @-SP					2						00010010	Register list		
	XCH.W Rs,Rd	2										10010	RdRdRd		
	SWAP.B Rd	2										00010000			
	(MOVTP.E.B Rs,<EAd>)*1		3	4	5	3	3	4	5			00000000	10010	RsfRd	
(MOVFP.E.B <EAs>,Rd)*1		3	4	5	3	3	4	5			00000000	10000	RdRdRd		
Arithmetic operations	ADD:G.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		00100	RdRdRd		
	ADD:G.W <EAs>,Rd	2	2	3	4	2	2	3	4	4		00100	RdRdRd		
	ADD:Q.B #1,<EAd>*2	2	2	3	4	2	2	3	4			00001000			
	ADD:Q.W #1,<EAd>*2	2	2	3	4	2	2	3	4			00001000			
	ADD:Q.B #2,<EAd>*2	2	2	3	4	2	2	3	4			00001001			
	ADD:Q.W #2,<EAd>*2	2	2	3	4	2	2	3	4			00001001			
	ADD:Q.B #-1,<EAd>*2	2	2	3	4	2	2	3	4			00001100			
	ADD:Q.W #-1,<EAd>*2	2	2	3	4	2	2	3	4			00001100			
	ADD:Q.B #-2,<EAd>*2	2	2	3	4	2	2	3	4			00001101			
	ADD:Q.W #-2,<EAd>*2	2	2	3	4	2	2	3	4			00001101			
	ADDS.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		00101	RdRdRd		
	ADDS.W <EAs>,Rd	2	2	3	4	2	2	3	4	4		00101	RdRdRd		
ADDX.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		10100	RdRdRd			
ADDX.W <EAs>,Rd	2	2	3	4	2	2	3	4	4		10100	RdRdRd			

Notes: 1. Not available in the H8/539F.  
2. Short format.

**Table A-1 (a) Machine-Language Instruction Codes [General Format] (cont) (2)**

Instruction		Effective Address (EA) Field											Operation Code (OP) Field				
		Address- sing Mode											4	5	6		
		1	2	3													
Arithmetic operations	DADD.B Rs,Rd	3													00000000	10100rafaRa	
	SUB.B <EAs>,Rd	2	2	3	4	2	2	3	4	3					00110rafaRa		
	SUB.W <EAs>,Rd	2	2	3	4	2	2	3	4	4					00110rafaRa		
	SUBS.B <EAs>,Rd	2	2	3	4	2	2	3	4	3					00111rafaRa		
	SUBS.W <EAs>,Rd	2	2	3	4	2	2	3	4	4					00111rafaRa		
	SUBX.B <EAs>,Rd	2	2	3	4	2	2	3	4	3					10110rafaRa		
	SUBX.W <EAs>,Rd	2	2	3	4	2	2	3	4	4					10110rafaRa		
	DSUB.B Rs,Rd	3													00000000	10110rdrdRd	
	MULXU.B <EAs>,Rd	2	2	3	4	2	2	3	4	3					10101rafaRa		
	MULXU.W <EAs>,Rd	2	2	3	4	2	2	3	4	4					10101rafaRa		
	DIVXU.B <EAs>,Rd	2	2	3	4	2	2	3	4	3					10111rafaRa		
	DIVXU.W <EAs>,Rd	2	2	3	4	2	2	3	4	4					10111rafaRa		
	CMP:G.B <EAs>,Rd	2	2	3	4	2	2	3	4	3					01110rafaRa		
	CMP:G.W <EAs>,Rd	2	2	3	4	2	2	3	4	4					01110rafaRa		
	CMP:G.B #xx,<EAd>			3	4	5	3	3	4	5					00000100	Data	
	CMP:G.W #xx,<EAd>			4	5	6	4	4	5	6					00000101	Data (high)	Data (low)
	EXTS.B Rd	2													00010001		
	EXTU.B Rd	2													00010010		
	TST.B <EAd>	2	2	3	4	2	2	3	4						00010110		
	TST.W <EAd>	2	2	3	4	2	2	3	4						00010110		
	NEG.B <EAd>	2	2	3	4	2	2	3	4						00010100		
	NEG.W <EAd>	2	2	3	4	2	2	3	4						00010100		
	CLR.B <EAd>	2	2	3	4	2	2	3	4						00010011		
CLR.W <EAd>	2	2	3	4	2	2	3	4						00010011			
TAS.B <EAd>	2	2	3	4	2	2	3	4						00010111			

**Table A-1 (a) Machine-Language Instruction Codes [General Format] (3)**

Instruction		Effective Address (EA) Field									Operation Code (OP) Field			
		Addressing Mode	1			2			3			4	5	6
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8			
Shift	SHAL.B <EAd>	2	2	3	4	2	2	3	4			00011000		
	SHAL.W <EAd>	2	2	3	4	2	2	3	4			00011000		
	SHAR.B <EAd>	2	2	3	4	2	2	3	4			00011001		
	SHAR.W <EAd>	2	2	3	4	2	2	3	4			00011001		
	SHLL.B <EAd>	2	2	3	4	2	2	3	4			00011010		
	SHLL.W <EAd>	2	2	3	4	2	2	3	4			00011010		
	SHLR.B <EAd>	2	2	3	4	2	2	3	4			00011011		
	SHLR.W <EAd>	2	2	3	4	2	2	3	4			00011011		
	ROTL.B <EAd>	2	2	3	4	2	2	3	4			00011100		
	ROTL.W <EAd>	2	2	3	4	2	2	3	4			00011100		
	ROTR.B <EAd>	2	2	3	4	2	2	3	4			00011101		
	ROTR.W <EAd>	2	2	3	4	2	2	3	4			00011101		
	ROTXL.B <EAd>	2	2	3	4	2	2	3	4			00011110		
	ROTXL.W <EAd>	2	2	3	4	2	2	3	4			00011110		
	ROTXR.B <EAd>	2	2	3	4	2	2	3	4			00011111		
	ROTXR.W <EAd>	2	2	3	4	2	2	3	4			00011111		
Logic operations	AND.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		01010r <sub>a</sub> r <sub>a</sub> r <sub>d</sub>		
	AND.W <EAs>,Rd	2	2	3	4	2	2	3	4	4		01010r <sub>a</sub> r <sub>a</sub> r <sub>d</sub>		
	OR.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		01000r <sub>a</sub> r <sub>a</sub> r <sub>d</sub>		
	OR.W <EAs>,Rd	2	2	3	4	2	2	3	4	4		01000r <sub>a</sub> r <sub>a</sub> r <sub>d</sub>		
	XOR.B <EAs>,Rd	2	2	3	4	2	2	3	4	3		01100r <sub>a</sub> r <sub>a</sub> r <sub>d</sub>		
	XOR.W <EAs>,Rd	2	2	3	4	2	2	3	4	4		01100r <sub>a</sub> r <sub>a</sub> r <sub>d</sub>		
	NOT.B <EAd>	2	2	3	4	2	2	3	4			00010101		
	NOT.W <EAd>	2	2	3	4	2	2	3	4			00010101		

**Table A-1 (a) Machine-Language Instruction Codes [General Format] (4)**

Instruction		Effective Address (EA) Field										Operation Code (OP) Field			
		Address- sing Mode										4	5	6	
		1	2	3											
		Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16				
Bit operations	BSET.B #xx,<EAd>	2	2	3	4	2	2	3	4			1100	data		
	BSET.W #xx,<EAd>	2	2	3	4	2	2	3	4			1100	data		
	BSET.B Rs,<EAd>	2	2	3	4	2	2	3	4			01001	rsfsfs		
	BSET.W Rs,<EAd>	2	2	3	4	2	2	3	4			01001	rsfsfs		
	BCLR.B #xx,<EAd>	2	2	3	4	2	2	3	4			1101	data		
	BCLR.W #xx,<EAd>	2	2	3	4	2	2	3	4			1101	data		
	BCLR.B Rs,<EAd>	2	2	3	4	2	2	3	4			01011	rsfsfs		
	BCLR.W Rs,<EAd>	2	2	3	4	2	2	3	4			01011	rsfsfs		
	BTST.B #xx,<EAd>	2	2	3	4	2	2	3	4			1111	data		
	BTST.W #xx,<EAd>	2	2	3	4	2	2	3	4			1111	data		
	BTST.B Rs,<EAd>	2	2	3	4	2	2	3	4			01111	rsfsfs		
	BTST.W Rs,<EAd>	2	2	3	4	2	2	3	4			01111	rsfsfs		
	BNOT.B #xx,<EAd>	2	2	3	4	2	2	3	4			1110	data		
	BNOT.W #xx,<EAd>	2	2	3	4	2	2	3	4			1110	data		
BNOT.B Rs,<EAd>	2	2	3	4	2	2	3	4			01101	rsfsfs			
BNOT.W Rs,<EAd>	2	2	3	4	2	2	3	4			01101	rsfsfs			
System control	LDC.B <EAs>,CR	2	2	3	4	2	2	3	4	3		10001	ccc		
	LDC.W <EAs>,CR	2	2	3	4	2	2	3	4		4	10001	ccc		
	STC.B CR,<EAd>	2	2	3	4	2	2	3	4			10011	ccc		
	STC.W CR,<EAd>	2	2	3	4	2	2	3	4			10011	ccc		
	ANDC.B #xx:8, CR										3	01011	ccc		
	ANDC.W #xx:16, CR										4	01011	ccc		
	ORC.B #xx:8, CR										3	01001	ccc		
	ORC.W #xx:16, CR										4	01001	ccc		
	XORC.B #xx:8, CR										3	01101	ccc		
XORC.W #xx:16, CR										4	01101	ccc			

**Table A-1 (b) Machine-Language Instruction Codes [Special Format: Short Format]**

Instruction	Byte Length	Machine-Language Code			
		1	2	3	4
MOV:E.B #xx:8, Rd	2	01010rdrdrd	Data		
MOV:I.W #xx:16, Rd	3	01011rdrdrd	Data (high)	Data (low)	
MOV:L.B @aa:8, Rd	2	01100rdrdrd	Address (low)		
MOV:L.W @aa:8, Rd	2	01101rdrdrd	Address (low)		
MOV:S.B Rs, @aa:8	2	01110rsrsfs	Address (low)		
MOV:S.W Rs, @aa:8	2	01111rsrsfs	Address (low)		
MOV:F.B @(d:8,R6), Rd	2	10000rdrdrd	disp		
MOV:F.W @(d:8,R6), Rd	2	10001rdrdrd	disp		
MOV:F.B Rs, @(d:8, R6)	2	10010rsrsfs	disp		
MOV:F.W Rs, @(d:8, R6)	2	10011rsrsfs	disp		
CMP:E #xx8, Rd	2	01000rdrdrd	Data		
CMP:I #xx16, Rd	3	01001rdrdrd	Data (high)	Data (low)	

**Table A-1 (c) Machine-Language Instruction Codes**  
**[Special Format: Branch Instructions] (1)**

Instruction		Byte Length	Machine-Language Code			
			1	2	3	4
Bcc d:8	BRA (BT)	2	00100000	disp		
	BRN (BF)		00100001	disp		
	BHI		00100010	disp		
	BLS		00100011	disp		
	BCC (BHS)		00100100	disp		
	BCS (BLO)		00100101	disp		
	BNE		00100110	disp		
	BEQ		00100111	disp		
	BVC		00101000	disp		
	BVS		00101001	disp		
	BPL		00101010	disp		
	BMI		00101011	disp		
	BGE		00101100	disp		
	BLT		00101101	disp		
	BGT		00101110	disp		
BLE	00101111	disp				
Bcc d:16	BRA (BT)	3	00110000	disp H	disp L	
	BRN (BF)		00110001	disp H	disp L	
	BHI		00110010	disp H	disp L	
	BLS		00110011	disp H	disp L	
	BCC (BHS)		00110100	disp H	disp L	
	BCS (BLO)		00110101	disp H	disp L	
	BNE		00110110	disp H	disp L	
	BEQ		00110111	disp H	disp L	
	BVC		00111000	disp H	disp L	
	BVS		00111001	disp H	disp L	
	BPL		00111010	disp H	disp L	
	BMI		00111011	disp H	disp L	
BGE	00111100	disp H	disp L			

**Table A-1 (c) Machine-Language Instruction Codes  
[Special Format: Branch Instructions] (2)**

Instruction	Byte Length	Machine-Language Code			
		1	2	3	4
Bcc d:16	3	00111101	disp H	disp L	
BGT		00111110	disp H	disp L	
BLE		00111111	disp H	disp L	
JMP @Rn	2	00010001	11010rrr		
JMP @aa:16	3	00010000	Address (high)	Address (low)	
JMP @(d:8, Rn)	3	00010001	11100rrr	disp	
JMP @(d:16, Rn)	4	00010001	11110rrr	disp H	disp L
BSR d:8	2	00001110	disp		
BSR d:16	3	00011110	disp H	disp L	
JSR @Rn	2	00010001	11011rrr		
JSR @aa:16	3	00011000	Address (high)	Address (low)	
JSR @(d:8, Rn)	3	00010001	11101rrr	disp	
JSR @(d:16, Rn)	4	00010001	11111rrr	disp H	disp L
RTS	1	00011001			
RTD #xx:8	2	00010100	Data		
RTD #xx:16	3	00011100	Data (high)	Data (low)	
SCB/cc Rn,disp	3	00000001	10111rrr	disp	
SCB/F		00000110	10111rrr	disp	
SCB/NE		00000111	10111rrr	disp	
SCB/EQ					
PJMP @aa:24	4	00010011	Page	Address (high)	Address (low)
PJMP @Rn	2	00010001	11000rrr		
PJSR @aa:24	4	00000011	Page	Address (high)	Address (low)
PJSR @Rn	2	00010001	11001rrr		
PRTS	2	00010001	00011001		
PRTD #xx:8	3	00010001	00010100	Data	
PRTD #xx:16	4	00010001	00011100	Data (high)	Data (low)

**Table A-1 (d) Machine-Language Instruction Codes**  
**[Special Format: System Control Instructions]**

Instruction	Byte Length	Machine-Language Code			
		1	2	3	4
TRAPA #xx	2	00001000	0001 #VEC		
TRAP/VS	1	00001001			
PTE	1	00001010			
LINK FP,#xx:8	2	00010111	Data		
LINK FP,#xx:16	3	00011111	Data (high)	Data (low)	
UNLK FP	1	00001111			
SLEEP	1	00011010			
NOP	1	00000000			



## A.3 Operation Code Map

Tables A-2 to A-6 show a map of the machine-language instruction codes. The map includes the effective address (EA) and operation code (OP) fields but not the effective address extension.

**Table A-2 First Byte of Instruction Code**

HI	LO															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	SCB/F Table A-6	LDM	PJSR @aa:24	#xx:8 Table A-5	@aa:8.B Table A-4	SCB/NE Table A-6	SCB/EQ Table A-6	TRAPA	TRAP/VS	RTE		#xx:16 Table A-5	@aa:8.W Table A-4	BSR d:8	UNLK
1	JMP	Table A-6*	STM	PJMP @aa:24	RTD #xx:8	@aa:16.B Table A-4		LINK #xx:8	JSR	RTS	SLEEP		RTD #xx:16	@aa:16.W Table A-4	BSR d:16	LINK #xx:16
2	BRA d:8	BRN	BHI	BLS	Bcc	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
3	BRA d:16	BRN	BHI	BLS	Bcc	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
4	CMP:E #xx:8, Rn R0 R1 R2 R3 R4 R5 R6 R7								CMP:I #xx:16, Rn R0 R1 R2 R3 R4 R5 R6 R7							
5	MOV:E #xx:8,Rn								MOV:I #xx:16,Rn							
6	MOV:L.B @aa:8,Rn								MOV:L.W @aa:8,Rn							
7	MOV:S.B Rn,@aa:8								MOV:S.W Rn,@aa:8							
8	MOV:F.B @(d:8,R6),Rn								MOV:F.W @(d:8,R6),Rn							
9	MOV:F.B Rn,@(d:8,R6)								MOV:F.W Rn,@(d:8,R6)							
A	Rn (byte) Table A-3								Rn (word) Table A-3							
B	@-Rn (byte) Table A-4								@-Rn (word) Table A-4							
C	@Rn+ (byte) Table A-4								@Rn+ (word) Table A-4							
D	@Rn (byte) Table A-4								@Rn (word) Table A-4							
E	@ (d:8,Rn) (byte) Table A-4								@ (d:8,Rn) (word) Table A-4							
F	@ (d:16,Rn) (byte) Table A-4								@ (d:16,Rn) (word) Table A-4							

Note: \* References to tables A-3 to A-6 indicate the table giving the second or a subsequent byte of the machine-language code.

H'11 is the first byte of the machine-language code of the following instructions:  
 JMP, JSR, PJMP, and PJSR in register indirect addressing mode;  
 JMP and JSR in register indirect addressing mode with displacement;  
 PRTS and PRTD.

**Table A-3 Second Byte of Axxx Instruction Codes**

	LO																		
HI		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
0	Table A-6*								ADD:Q #1	ADD:Q #2				ADD:Q #-1	ADD:Q #-2				
1	SWAP	EXTS	EXTU	CLR	NEG	NOT	TST	TAS	SHAL	SHAR	SHLL	SHLR	ROTL	ROTR	ROTXL	ROTXR			
2	ADD R0 R1 R2 R3 R4 R5 R6 R7							ADDS R0 R1 R2 R3 R4 R5 R6 R7											
3	SUB							SUBS											
4	OR							BSET (register indirect specification of bit number)											
5	AND							BCLR (register indirect specification of bit number)											
6	XOR							BNOT (register indirect specification of bit number)											
7	CMP							BTST (register indirect specification of bit number)											
8	MOV							LDC											
9	XCH							STC											
A	ADDX							MULXU											
B	SUBX							DIVXU											
C	b0	b1	b2	b3	b4	b5	BSET (direct specification of bit number) b6 b7 b8 b9 b10				b11	b12	b13	b14	b15				
D	BCLR (direct specification of bit number)																		
E	BNOT (direct specification of bit number)																		
F	BTST (direct specification of bit number)																		

Note: \* Prefix code of the DADD and DSUB instructions. Table A-6 gives the third byte of the instruction code.

**Table A-4 Second Byte of 05xx, 15xx, 0Dxx, 1Dxx, Bxxx, Cxxx, Dxxx, Exxx, and Fxxx Instruction Codes**

	LO															
HI	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Table A-6*				CMP #xx:8	CMP #xx:16	MOV #xx:8	MOV #xx:16	ADD:Q #1	ADD:Q #2		ADD:Q #-1	ADD:Q #-2			
1				CLR	NEG	NOT	TST	TAS	SHAL	SHAR	SHLL	SHLR	ROTL	ROTR	ROTXL	ROTXR
2	ADD R0 R1 R2 R3 R4 R5 R6 R7								ADDS R0 R1 R2 R3 R4 R5 R6 R7							
3	SUB								SUBS							
4	OR								BSET (register indirect specification of bit number)							
5	AND								BCLR (register indirect specification of bit number)							
6	XOR								BNOT (register indirect specification of bit number)							
7	CMP								BTST (register indirect specification of bit number)							
8	MOV (load)								LDC							
9	MOV (store)								STC							
A	ADDX								MULXU							
B	SUBX								DIVXU							
C	b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15
D	BSET (direct specification of bit number)															
E	BCLR (direct specification of bit number)															
F	BNOT (direct specification of bit number)															
	BTST (direct specification of bit number)															

Note: \* Prefix code of the DADD and DSUB instructions. Table A-6 gives the third byte of the instruction code.

**Table A-5 Second Byte of 04xx and 0Cxx Instruction Codes**

	LO															
HI	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2	ADD								ADDS							
	R0	R1	R2	R3	R4	R5	R6	R7	R0	R1	R2	R3	R4	R5	R6	R7
3	SUB								SUBS							
4	OR								ORC							
5	AND								ANDC							
6	XOR								XORC							
7	CMP															
8	MOV								LDC							
9																
A	ADDX								MULXU							
B	SUBX								DIVXU							
C																
D																
E																
F																

**Table A-6 Second or Third Byte of 11xx, 01xx, 06xx, 07xx, and xx00xx Instruction Codes**

LO HI		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
		0																	
1					PRTD #xx:8					PRTS					PRTD #xx:16				
2																			
3																			
4																			
5																			
6																			
7																			
8					(MOVFPE)*														
	R0	R1	R2	R3	R4	R5	R6	R7											
9	(MOVTPE)*																		
A	DADD																		
B	DSUB																		
C	PJOB @Rn																		
D	JMP @Rn																		
E	JMP @ (d:8,Rn)																		
F	JMP @ (d:16,Rn)																		
													SCB						
	R0	R1	R2	R3	R4	R5	R6	R7											
	PJSR @Rn																		
	JSR @Rn																		
	JSR @ (d:8,Rn)																		
	JSR @ (d:16,Rn)																		

Note: \* Not available in the H8/539F.

## A.4 Number of States Required for Execution

Tables A-7 (1) to (6) indicate the number of states required to execute each instruction in each addressing mode. These tables are read as explained below. The values of I, J, and K are used to calculate the number of execution states when the instruction is fetched from an external address or an operand is written or read at an external address. Formulas for calculating the number of states are given on the next page.

### How to Read Table A-7

J + K is the number of instruction fetches

I is the total number of bytes written or read when the operand is in memory

Instruction	I	J	K	Addressing Mode									
				Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
				1	1	2	3	1	1	2	3	2	3
ADD.B	1	1	2	5	5	6	5	6	5	6	3		
ADD.W	2	1	2	5	5	6	5	6	5	6			4
ADD:Q.B	2	1	2	7	7	8	7	8	7	8			
ADD:Q.W	4	1	2	7	7	8	7	8	7	8			
DADD		2	4										

Shading in the I column indicates that the instruction cannot have a memory operand.

Shading in these columns indicates addressing modes that cannot be specified for the instruction.

**Calculation of Number of States Required for Execution:** One state is one cycle of the system clock ( $\phi$ ). When  $\phi = 10$  MHz, one state is 100 ns.

Instruction Fetch	Operand Read/Write	Formula		
16-bit-bus, 2-state-access area	16-bit-bus, 2-state-access area or general register	(value in table A-7) + (value in table A-8)		
	16-bit-bus, 3-state-access area	Byte	(value in table A-7) + (value in table A-8) + I	
		Word	(value in table A-7) + (value in table A-8) + I/2	
8-bit-bus, 2-state-access area or on-chip supporting module	8-bit-bus, 2-state-access area or on-chip supporting module	Byte	(value in table A-7) + (value in table A-8) + I	
		Word	(value in table A-7) + (value in table A-8) + 2I	
16-bit-bus, 3-state-access area	16-bit-bus, 2-state-access area or general register	(value in table A-7) + (value in table A-8) + (J + K)/2		
	16-bit-bus, 3-state-access area	Byte	(value in table A-7) + (value in table A-8) + I + (J + K)/2	
		Word	(value in table A-7) + (value in table A-8) + (I + J + K)/2	
	8-bit-bus, 2-state-access area or on-chip supporting module	8-bit-bus, 2-state-access area or on-chip supporting module	Byte	(value in table A-7) + (value in table A-8) + I + (J + K)/2
			Word	(value in table A-7) + (value in table A-8) + 2I + (J + K)/2
	8-bit-bus, 3-state-access area	16-bit-bus, 2-state-access area or general register	(value in table A-7) + 2 + (J + K)	
16-bit-bus, 3-state-access area		Byte	(value in table A-7) + I + 2 (J + K)	
		Word	(value in table A-7) + I/2 + 2 (J + K)	
8-bit-bus, 2-state-access area or on-chip supporting module		8-bit-bus, 2-state-access area or on-chip supporting module	Byte	(value in table A-7) + I + 2 (J + K)
			Word	(value in table A-7) + 2 (I + J + K)

- Notes:
1. When an instruction is fetched from the 16-bit-bus access area, the number of states differs by 1 or 2 depending on whether the instruction is stored at an even or odd address. This point should be noted in software timing routines and other situations in which the precise number of states must be known.
  2. If wait states or  $T_p$  states are inserted in access to the 3-state-access area, add the necessary number of states.
  3. When an instruction is fetched from the 16-bit-bus 3-state-access area, fractions in the term  $(J + K)/2$  should be rounded up.

## Examples of Calculation of Number of States Required for Execution

**Example 1:** Instruction fetched from 16-bit-bus, 2-state-access area

Operand Read/Write	Start Address	Assembler Notation			Formula (Value in Table A-7) + (Value in Table A-8 (b))	Execution States
		Address	Code	Mnemonic		
16-bit-bus, 2-state- access area or general register	Even	H'0100	D821	ADD @R0,R1	$5 + 1$	6
	Odd	H'0101	D821	ADD @R0,R1	$5 + 0$	5

**Example 2:** Instruction fetched from 16-bit-bus, 2-state-access area

Operand Read/Write	Branch destination Address	Assembler Notation			Formula (Value in Table A-7) + (Value in Table A-8 (a)) + 2I	Execution States
		Address	Code	Mnemonic		
On-chip supporting module or 8-bit-bus, 3-state- access area (word)	Even	H'FC00	11D8	JSR @R0	$9 + 0 + 2 \times 2$	13
	Odd	H'FC01	11D8	JSR @R0	$9 + 1 + 2 \times 2$	14

**Example 3:** Instruction fetched from 8-bit-bus, 3-state-access area

Operand Read/Write	Assembler Notation			Formula (Value in Table A-7) + 2 (J + K)	Execution States
	Address	Code	Mnemonic		
16-bit-bus, 2-state- access area or general register	H'9002	D821	ADD @R0,R1	$5 + 2 \times (1 + 1)$	9



**Example 4:** Instruction fetched from 16-bit-bus, 3-state-access area

Operand Read/Write	Start Address	Assembler Notation			Formula (Value in Table A-7) + (Value in Table A-8 (b)) + (J + K)/2	Execution States
		Address	Code	Mnemonic		
16-bit-bus, 2-state- access area or general register	Even	H'0100	D821	ADD @R0,R1	$5 + 1 + (1 + 1)/2$	7
	Odd	H'0101	D821	ADD @R0,R1	$5 + 0 + (1 + 1)/2$	6

**Table A-7 Number of States Required for Instruction Execution (1)**

Instruction	I J K			Addressing Mode									
				Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
				1	1	2	3	1	1	2	3	2	3
ADD:G.B <EAs>,Rd	1	1	1	2	5	5	6	5	6	5	6	3	
ADD:G.W <EAs>,Rd	2	1	1	2	5	5	6	5	6	5	6		4
ADD:Q.B #xx, <EAd>	2	1	1	2	7	7	8	7	8	7	8		
ADD:Q.W #xx, <EAd>	4	1	1	2	7	7	8	7	8	7	8		
ADDS.B <EAs>, Rd	1	1	1	3	5	5	6	5	6	5	6	3	
ADDS.W <EAs>, Rd	2	1	1	3	5	5	6	5	6	5	6		4
ADDX.B <EAs>, Rd	1	1	1	2	5	5	6	5	6	5	6	3	
ADDX.W <EAs>, Rd	2	1	1	2	5	5	6	5	6	5	6		4
AND.B <EAs>, Rd	1	1	1	2	5	5	6	5	6	5	6	3	
AND.W <EAs>, Rd	2	1	1	2	5	5	6	5	6	5	6		4
ANDC #xx,CR			1									5	9
BCLR.B #xx, <EAd>	*	2	1	4	7	7	8	7	8	7	8		
BCLR.W #xx, <EAd>	*	4	1	4	7	7	8	7	8	7	8		
BNOT.B #xx, <EAd>	*	2	1	4	7	7	8	7	8	7	8		
BNOT.W #xx, <EAd>	*	4	1	4	7	7	8	7	8	7	8		
BSET.B #xx, <EAd>	*	2	1	4	7	7	8	7	8	7	8		
BSET.W #xx, <EAd>	*	4	1	4	7	7	8	7	8	7	8		
BTST.B #xx, <EAd>	*	1	1	3	5	5	6	5	6	5	6		
BTST.W #xx, <EAd>	*	2	1	3	5	5	6	5	6	5	6		
CLR.B <EAd>		1	1	2	5	5	6	5	6	5	6		
CLR.W <EAd>		2	1	2	5	5	6	5	6	5	6		
CMP:G.B <EAs>,Rd		1	1	2	5	5	6	5	6	5	6	3	
CMP:G.W <EAs>,Rd		2	1	2	5	5	6	5	6	5	6		4
CMP:G.B #xx:8, <EA>		1	2		6	6	7	6	7	6	7		
CMP:G.B #xx:16, <EA>		2	3		7	7	8	7	8	7	8		

Note: \* Rs can also be specified for the source operand.

**Table A-7 Number of States Required for Instruction Execution (2)**

Instruction	I	J \ K	Addressing Mode										
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16	
			1	1	2	3	1	1	2	3	2	3	
CMP:E #xx:8,Rd		0										2	
CMP:I #xx:16,Rd		0											3
DADD Rs,Rd		2	4										
DIVXU.B <EAs>,Rd	1	1	20	23	23	24	23	24	23	24	21		
DIVXU.W <EAs>,Rd	2	1	26	29	29	30	29	30	29	30			28
DSUB Rs,Rd		2	4										
EXTS Rd		1	3										
EXTU Rd		1	3										
LDC.B <EAs>,CR	1	1	3	6	6	7	6	7	6	7	4		
LDC.W <EAs>,CR	2	1	4	7	7	8	7	8	7	8			6
MOV:G.B	1	1	2	5	5	6	5	6	5	6	3		
MOV:G.W	2	1	2	5	5	6	5	6	5	6			4
MOV:G.B #xx:8,<EAd>	1	2		7	7	8	7	8	7	8			
MOV:G.W #xx:16,<EAd>	2	3		8	8	9	8	9	8	9			
MOV:E #xx:8,Rd		0										2	
MOV:I #xx:16,Rd		0											3
MOV:L.B @aa:8,Rd	1	0							5				
MOV:L.W @aa:8,Rd	2	0							5				
MOV:S.B Rs,@aa:8	1	0							5				
MOV:S.W Rs,@aa:8	2	0							5				
MOV:F.B @(d:8,R6),Rd	1	0			5								
MOV:F.W @(d:8,R6),Rd	2	0			5								
MOV:F.B Rs,@(d:8,R6)	1	0			5								
MOV:FW Rs,@(d:8,R6)	2	0			5								

**Table A-7 Number of States Required for Instruction Execution (3)**

Instruction	I	J	K	Addressing Mode										
				Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16	
				1	1	2	3	1	1	2	3	2	3	
(MOVFPPE <EAs>,Rd)*	0	2			13 20	13 20	14 21	13 20	14 21	13 20	14 21			
(MOVTPE Rs,<EA>)*	0	2			13 20	13 20	14 21	13 20	14 21	13 20	14 21			
MULXU.B <EAs>,Rd	1	1	16	19	19	20	19	20	19	20	18			
MULXU.W <EAs>,Rd	2	1	23	25	25	26	25	26	25	26	25		25	
NEG.B <EAd>	2	1	2	7	7	8	7	8	7	8				
NEG.W <EAd>	4	1	2	7	7	8	7	8	7	8				
NOT.B <EAd>	2	1	2	7	7	8	7	8	7	8				
NOT.W <EAd>	4	1	2	7	7	8	7	8	7	8				
OR.B <EAs>,Rd	1	1	2	5	5	6	5	6	5	6	3			
OR.W <EAs>,Rd	2	1	2	5	5	6	5	6	5	6			4	
ORC #xx,CR		1										5	9	
ROTL.B <EAd>	2	1	2	7	7	8	7	8	7	8				
ROTL.W <EAd>	4	1	2	7	7	8	7	8	7	8				
ROTR.B <EAd>	2	1	2	7	7	8	7	8	7	8				
ROTR.W <EAd>	4	1	2	7	7	8	7	8	7	8				
ROTXL.B <EAd>	2	1	2	7	7	8	7	8	7	8				
ROTXL.W <EAd>	4	1	2	7	7	8	7	8	7	8				
ROTXR.B <EAd>	2	1	2	7	7	8	7	8	7	8				
ROTXR.W <EAd>	4	1	2	7	7	8	7	8	7	8				
SHAL.B <EAd>	2	1	2	7	7	8	7	8	7	8				
SHAL.W <EAd>	4	1	2	7	7	8	7	8	7	8				
SHAR.B <EAd>	2	1	2	7	7	8	7	8	7	8				
SHAR.W <EAd>	4	1	2	7	7	8	7	8	7	8				
SHILL.B <EAd>	2	1	2	7	7	8	7	8	7	8				
SHLL.W <EAd>	4	1	2	7	7	8	7	8	7	8				

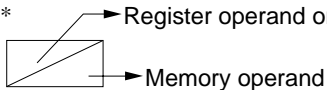
Note: \* Not available in the H8/539F.

**Table A-7 Number of States Required for Instruction Execution (4)**

Instruction	I	J	Addressing Mode									
			Rn	@Rn	@(d:8,Rn)	@(d:16,Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
SHLR.B <EAd>	2	1	2	7	7	8	7	8	7	8		
SHLR.W <EAd>	4	1	2	7	7	8	7	8	7	8		
STC.B CR,<EAd>	1	1	4	7	7	8	7	8	7	8		
STC.W CR,<EAd>	2	1	4	7	7	8	7	8	7	8		
SUB.B <EAs>,Rd	1	1	2	5	5	6	5	6	5	6	3	
SUB.W <EAs>,Rd	2	1	2	5	5	6	5	6	5	6		4
SUBS.B <EAs>,Rd	1	1	3	5	5	6	5	6	5	6	3	
SUBS.W <EAs>,Rd	2	1	3	5	5	6	5	6	5	6		4
SUBX.B <EAs>,Rd	1	1	2	5	5	6	5	6	5	6	3	
SUBX.W <EAs>,Rd	2	1	2	5	5	6	5	6	5	6		4
SWAP Rd		1	3									
TAS <EAd>	2	1	4	7	7	8	7	8	7	8		
TST.B <EAd>	1	1	2	5	5	6	5	6	5	6		
TST.W <EAd>	2	1	2	5	5	6	5	6	5	6		
XCH Rs,Rd		1	4									
XOR.B <EAs>,Rd	1	1	2	5	6	5	5	6	5	6	3	
XOR.W <EAs>,Rd	2	1	2	5	6	5	5	6	5	6		4
XORC #xx,CR		1									5	9

DIVXU.B zero divide, minimum mode	6 7	1	20	23	23	24	23	24	23	24	21	
DIVXU.B zero divide, maximum mode	10 11	1	25	28	28	29	28	29	28	29	21	
DIVXU.W zero divide, minimum mode	6 8	1	20	23	23	24	23	24	23	24		27
DIVXU.W zero divide, maximum mode	10 12	1	25	28	28	29	28	29	28	29		27
DIVXU.B overflow	1	1	8	11	11	12	11	12	11	12	9	
DIVXU.W overflow	2	1	8	11	11	12	11	12	11	12		10

Note: \* → Register operand or immediate data



**Table A-7 Number of States Required for Instruction Execution (5)**

<b>Instruction</b>	<b>(Condition)</b>	<b>Execution States</b>	<b>I</b>	<b>J + K</b>
Bcc d:8	Condition false, branch not taken	3		2
	Condition true, branch taken	7		5
Bcc d:16	Condition false, branch not taken	3		3
	Condition true, branch taken	7		6
BSR	d:8	9	2	4
	d:16	9	2	5
JMP	@aa:16	7		5
	@Rn	6		5
	@(d:8,Rn)	7		5
	@(d:16,Rn)	8		6
JSR	@aa:16	9	2	5
	@Rn	9	2	5
	@(d:8,Rn)	9	2	5
	@(d:16,Rn)	10	2	6
LDM		$6 + 4n^*$	$2n$	2
LINK	#xx:8	6	2	2
	#xx:16	7	2	3
NOP		2		1
RTD	#xx:8	9	2	4
	#xx:16	9	2	5
RTE	Minimum mode	13	4	4
	Maximum mode	15	6	4
RTS		8	2	4
SCB	Condition true, branch not taken	3		3
	Count = -1, branch not taken	4		3
	Other conditions, branch taken	8		6
SLEEP	Until transition to sleep mode	2		0
STM		$6 + 3n^*$	$2n$	2
TRAPA	Minimum mode	17	6	4
	Maximum mode	22	10	4

Note: \* n: number of registers in register list

**Table A-7 Number of States Required for Instruction Execution (6)**

Instruction	Condition	Execution States	I	J + K
TRAP/VS	V = 0, branch not taken	3		1
	V = 1, branch taken, minimum mode	18	6	4
	V = 1, branch taken, maximum mode	23	10	4
UNLK		5	2	1
PJMP	@aa:24	9		6
	@Rn	8		5
PJSR	@aa:24	15	4	6
	@Rn	13	4	5
PRTS		12	4	5
PRTD	#xx:8	13	4	5
	#xx:16	13	4	6

**Table A-8 (a) Correction Values (Branch Instructions)**

Instruction	Branch Address	Correction
BSR,JMP,JSR,RTS,RTD,RTE, TRAPA,PJMP,PJSR,PRTS,PRTD	Even	0
	Odd	1
Bcc,SCB,TRAP/VS (if branch is taken)	Even	0
	Odd	1

**Table A-8 (b) Correction Values (General Instructions, for Each Addressing Mode)**

Instruction	Start Address	Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
MOV.B #xx:8 <EA> MOVTPC,MOVFPD	Even		1	1	1	1	1	1	1		
	Odd		1	1	1	1	1	1	1		
MOV.W #xx:16 <EA>	Even		2	0	2	2	2	0	2		
	Odd		0	2	0	0	0	2	0		
All other instructions	Even	0	1	0	1	1	1	0	1	0	0
	Odd	0	0	1	0	0	0	1	0	0	0

## A.5 Instruction Set

### A.5.1 Features

Features of the H8/500 CPU instruction set are as follows:

- General-register architecture
- Highly orthogonal instruction set
- 1.5-address instructions
- C-oriented instruction set

### A.5.2 Instruction Types

The H8/500 CPU instruction set consists of 63 instructions. Table A-9 classifies the instruction set.

**Table A-9 Instruction Types**

Type	Instructions	Number of Instructions
Data transfer	MOV LDM STM XCH SWAP MOVTPE MOVFPE	7
Arithmetic operations	ADD SUB ADDS SUBS ADDX SUBX DADD DSUB MULXU DIVXU CMP EXTS EXTU TST NEG CLR TAS	17
Logic operations	AND OR XOR NOT	4
Shift	SHAL SHAR SHLL SHLR ROTL ROTR ROTXL ROTXR	8
Bit manipulation	BSET BCLR BTST BNOT	4
Branch	Bcc* JMP PJMP BSR JSR PJSR RTS PRTS RTD PRTD SCB(/F/NE/EQ)	11
System control	TRAPA TRAP/VS RTE SLEEP LDC STC ANDC ORC XORC NOP LINK UNLK	12

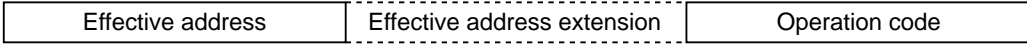
Note: \* Bcc is the generic designation for a conditional branch instruction.



### A.5.3 Basic Instruction Formats

There are two kinds of basic instruction format, general and special.

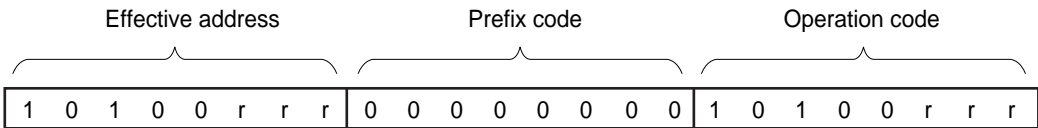
**(1) General Format:** This format consists of an effective address (EA) field, an effective address extension field, and an operation code (OP) field. The effective address is placed before the operation code because this results in faster execution of the instruction. Table A-10 describes the three fields of the general instruction format.



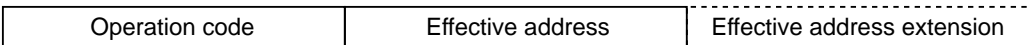
**Table A-10 Fields in General Instruction Format**

Name	Byte Length	Description
EA field	1	Information used to calculate the effective address
EA extension	0–2	Byte length is defined in EA field Displacement value, immediate data, or absolute address
OP field	1–3	Defines the operation carried out on the operand Some instructions (DADD, DSUB, MOVFPE, MOVTPPE) have an extended format in which the operand code is preceded by a one-byte prefix code (example 1)

**Example 1:** Instruction with prefix code: DADD instruction



**(2) Special Format:** In this format the operation code comes first, followed by the effective address field and effective address extension. This format is used in branching instructions, system control instructions, and some short-format instructions that can be executed faster if the operation is specified before the operand. Table A-11 describes the three fields of the special instruction format.



**Table A-11 Fields in Special Instruction Format**

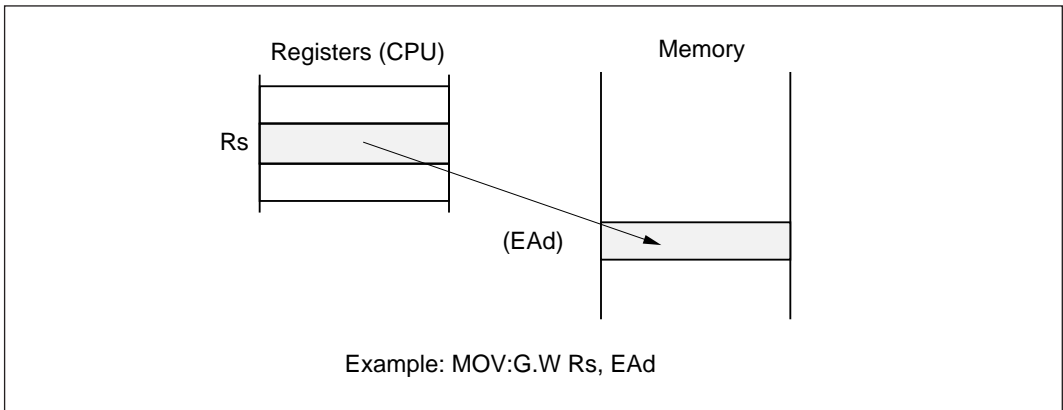
Name	Byte Length	Description
OP field	1–2	Defines the operation performed by the instruction
EA field and EA extension	0–3	Information used to calculate an effective address

**A.5.4 Data Transfer Instructions**

There are seven data transfer instructions. The function of each instruction is described next.

**(1) MOV Instruction:** Transfers data between two general registers, or between a general register and memory. Can also transfer immediate data to a general register or memory.

**Operation:** (EAs) → (EAd),  
 #IMM → (EAd)



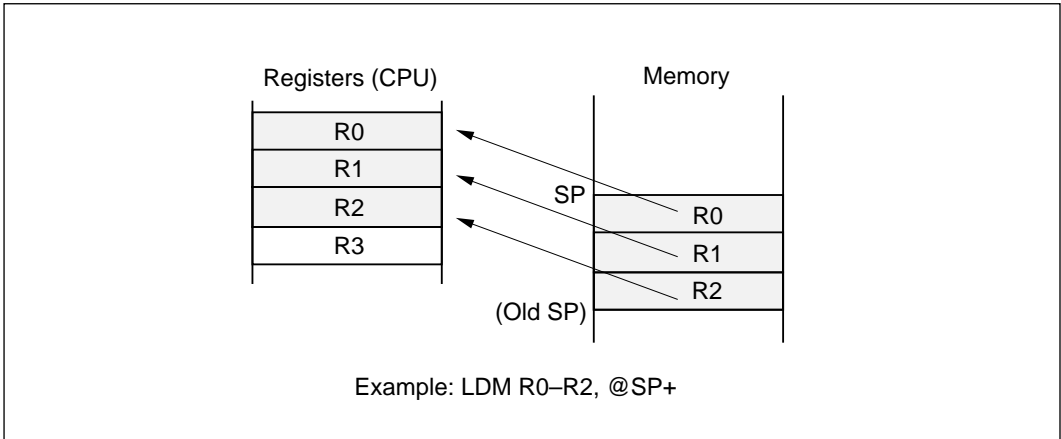
**Instructions and Operand Sizes:** The following table lists the possible combinations.

Instruction	Size		
	B/W	B	W
MOV:G	0		
MOV:E		0	
MOV:F	0		
MOV:I			0
MOV:L	0		
MOV:S	0		

B: Byte  
 W: Word

(2) **LDM Instruction (W):** Loads data saved on the stack into one or more registers. Multiple registers can be loaded simultaneously.

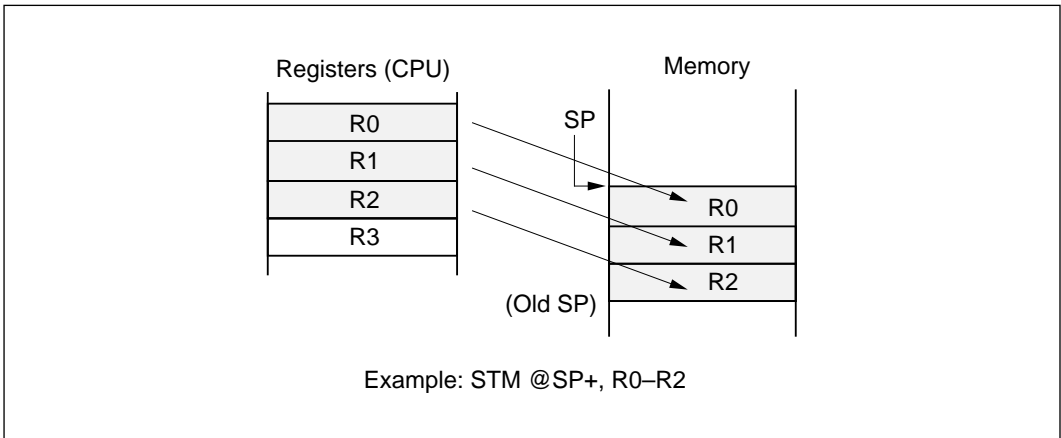
**Operation:** @SP+ (stack) → Rn (register list)



**Instructions and Operand Sizes:** The operand size is always word size.

(3) **STM Instruction (W):** Saves data onto the stack. Multiple registers can be saved simultaneously.

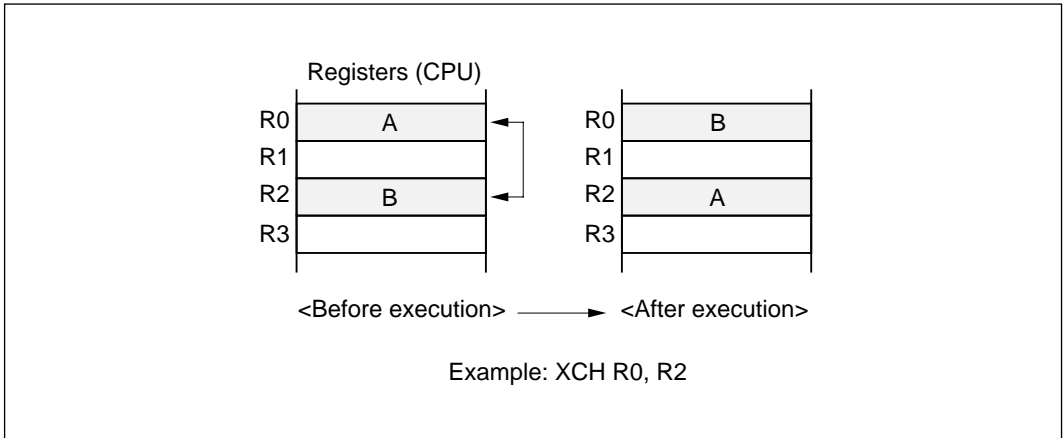
**Operation:** Rn (register list) → @SP+ (stack)



**Instructions and Operand Sizes:** The operand size is always word size.

(4) **XCH Instruction (W)**: Exchanges data between two general registers.

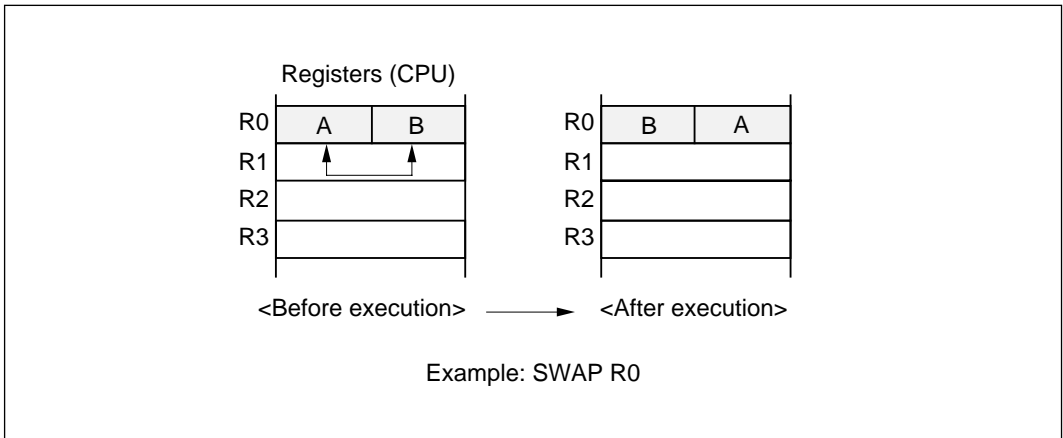
**Operation:**  $R_s \rightarrow R_d, R_d \rightarrow R_s$



**Instructions and Operand Sizes:** The operand size is always word size.

(5) **SWAP Instruction (B)**: Exchanges data between the upper and lower bytes of a general register.

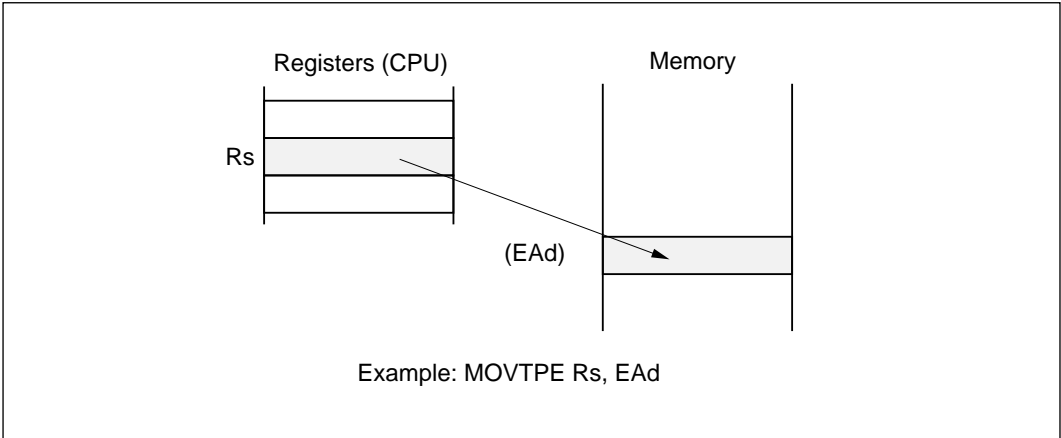
**Operation:**  $R_d \text{ (upper byte)} \leftrightarrow R_d \text{ (lower byte)}$



**Instructions and Operand Sizes:** The operand size is always byte size.

**(6) MOVTPE Instruction (B):** Transfers general register contents to memory in synchronization with the E clock. (Note: The H8/539F does not output an E clock).

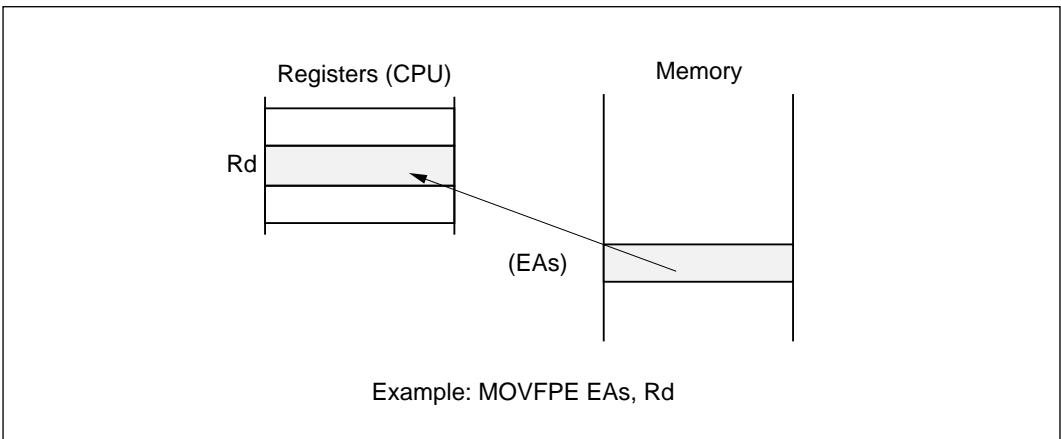
**Operation:**  $R_n \rightarrow (EAd)$



**Instructions and Operand Sizes:** The operand size is always byte size.

**(7) MOVFPE Instruction (B):** Transfers memory contents to a general register in synchronization with the E clock. (Note: The H8/539F does not output an E clock).

**Operation:**  $(EAs) \rightarrow R_d$



**Instructions and Operand Sizes:** The operand size is always byte size.

### A.5.5 Arithmetic Instructions

There are 17 arithmetic instructions. The function of each instruction is described next.

(1) **ADD Instruction (B/W)**

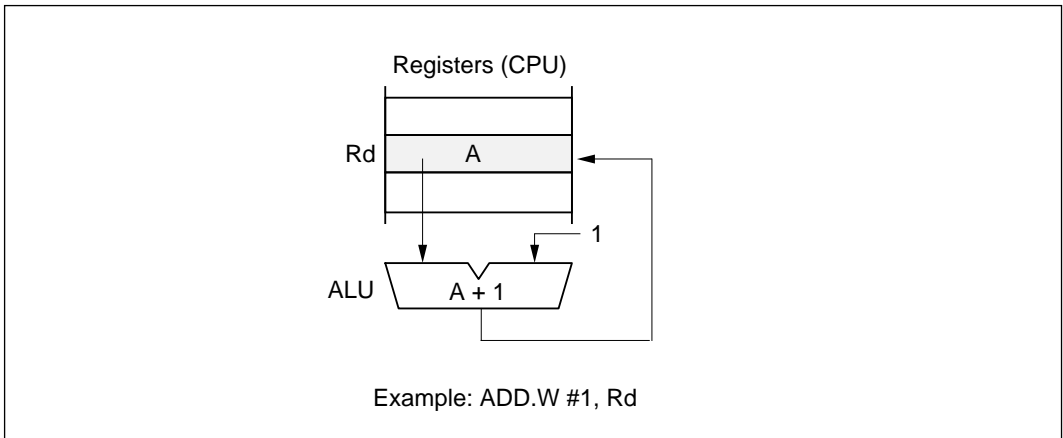
(2) **SUB Instruction (B/W)**

(3) **ADDS Instruction (B/W)**

(4) **SUBS Instruction (B/W)**

These instructions perform addition and subtraction on data in two general registers, data in a general register and memory, data in a general register and immediate data, or data in memory and immediate data.

**Operation:**  $Rd \pm (EAs) \rightarrow Rd$ ,  $(EAd) \pm \#IMM \rightarrow (EAd)$



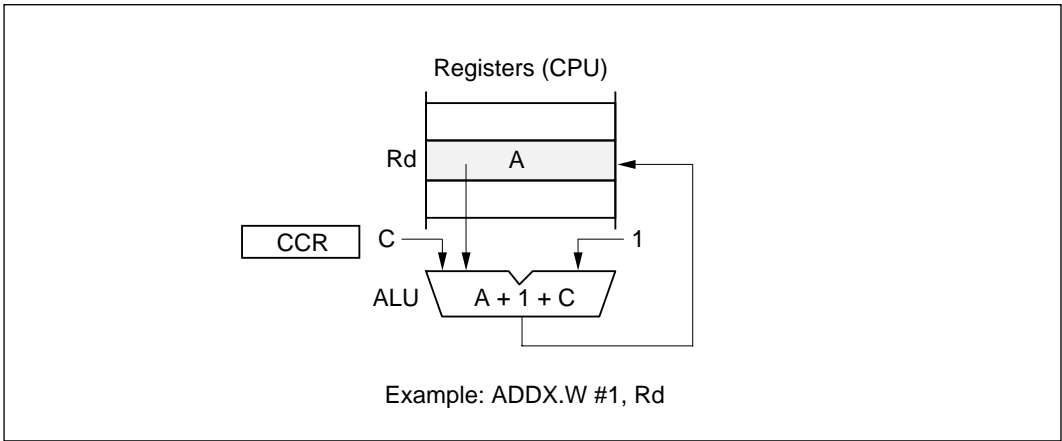
**Instructions and Operand Sizes:** Byte or word operand size can be selected.

(5) **ADDX Instruction (B/W)**

(6) **SUBX Instruction (B/W)**

These instructions perform addition and subtraction with carry on data in two general registers, data in a general register and memory, or data in a general register and immediate data.

**Operation:**  $Rd \pm (EAs) \pm C \rightarrow Rd$



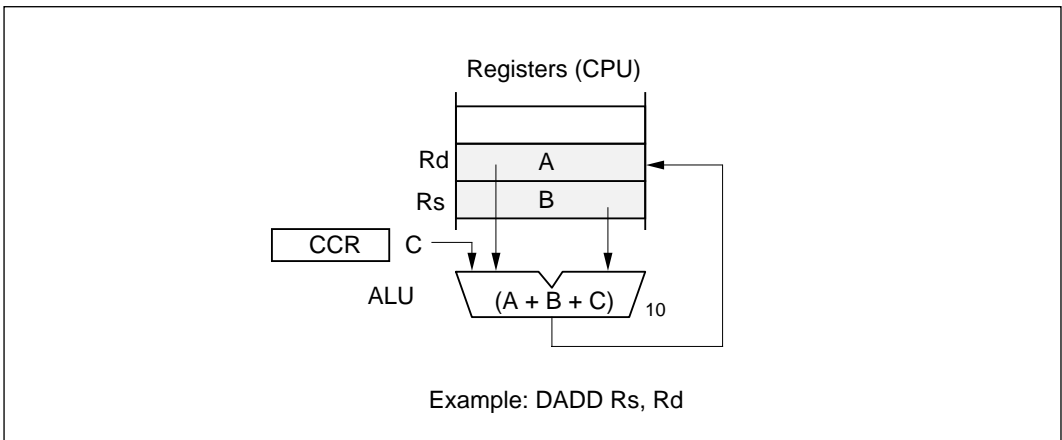
**Instructions and Operand Sizes:** Byte or word operand size can be selected.

(7) **DADD Instruction (B)**

(8) **DSUB Instruction (B)**

These instructions perform decimal addition and subtraction on data in two general registers.

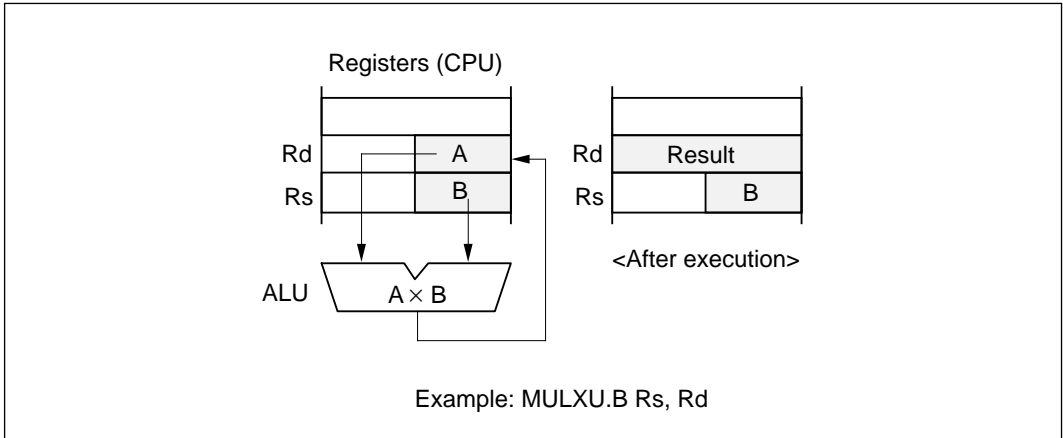
**Operation:**  $(Rd)_{10} \pm (Rs)_{10} \pm C \rightarrow (Rd)_{10}$



**Instructions and Operand Sizes:** The operand size is always byte size.

**(9) MULXU Instruction (B/W):** Performs 8-bit  $\times$  8-bit or 16-bit  $\times$  16-bit unsigned multiplication on data in a general register and data in another general register or memory, or on data in a general register and immediate data.

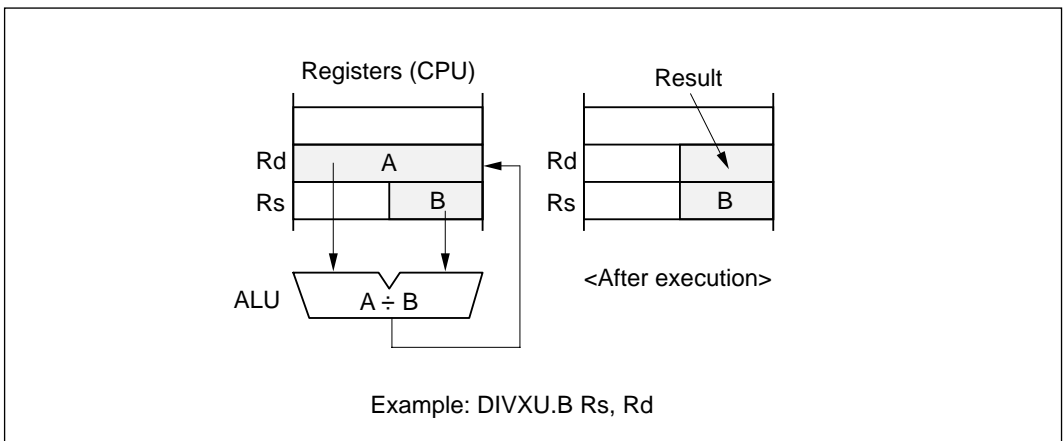
**Operation:**  $Rd \times (EAs) \rightarrow Rd$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(10) DIVXU Instruction (B/W):** Performs 16-bit  $\div$  8-bit or 32-bit  $\div$  16-bit unsigned division on data in a general register and data in another general register or memory, or on data in a general register and immediate data.

**Operation:**  $Rd \div (EAs) \rightarrow Rd$

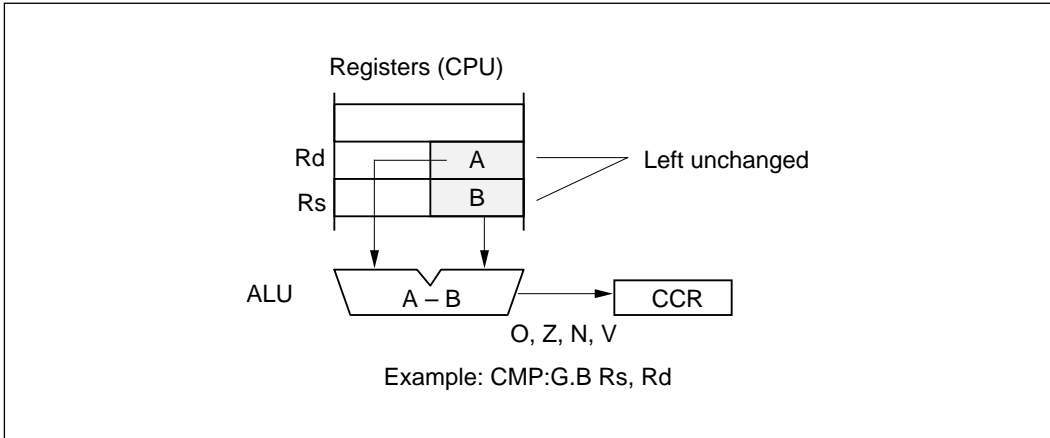


**Instructions and Operand Sizes:** Byte or word operand size can be selected.



**(11) CMP Instruction:** Compares data in a general register with data in another general register or memory, or with immediate data, or compares immediate data with data in memory.

**Operation:**  $Rd - (EAs), (EAd) - \#IMM$



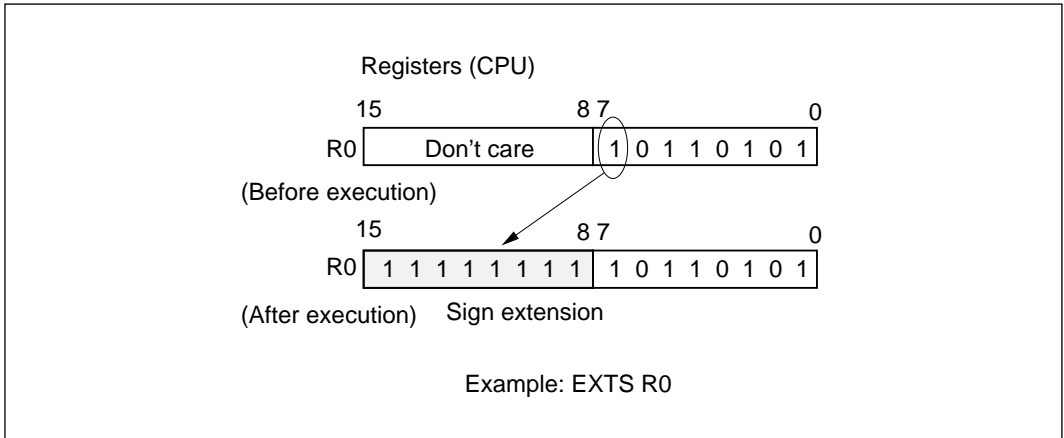
**Instructions and Operand Sizes:** The following table lists the possible combinations.

Instruction	Size		
	B/W	B	W
CMP:G	O		
CMP:E		O	
CMP:I			O

B: Byte  
W: Word

**(12) EXTS Instruction (B):** Converts byte data in a general register to word data by extending the sign bit.

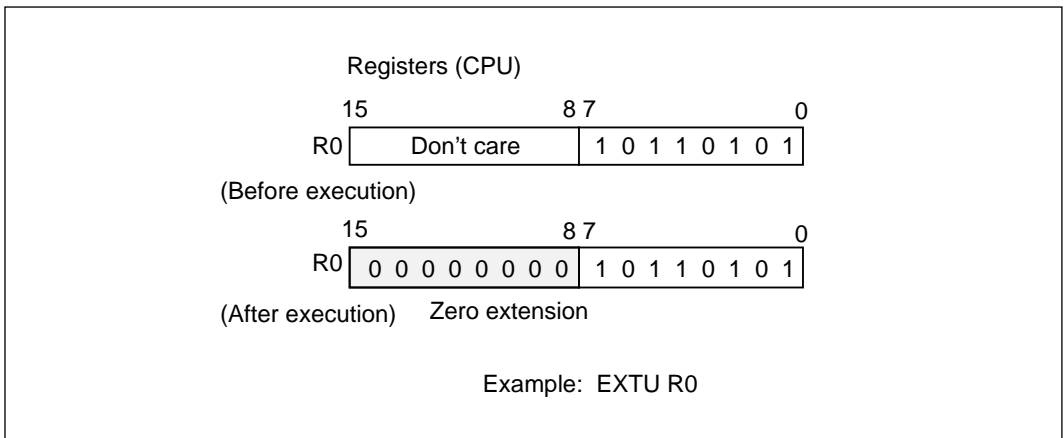
**Operation:** (<bit 7> of <Rd>) → (<bits 15 to 8> of <Rd>)



**Instructions and Operand Sizes:** The operand size is always byte size.

**(13) EXTU Instruction (B):** Converts byte data in a general register to word data by padding with zero bits.

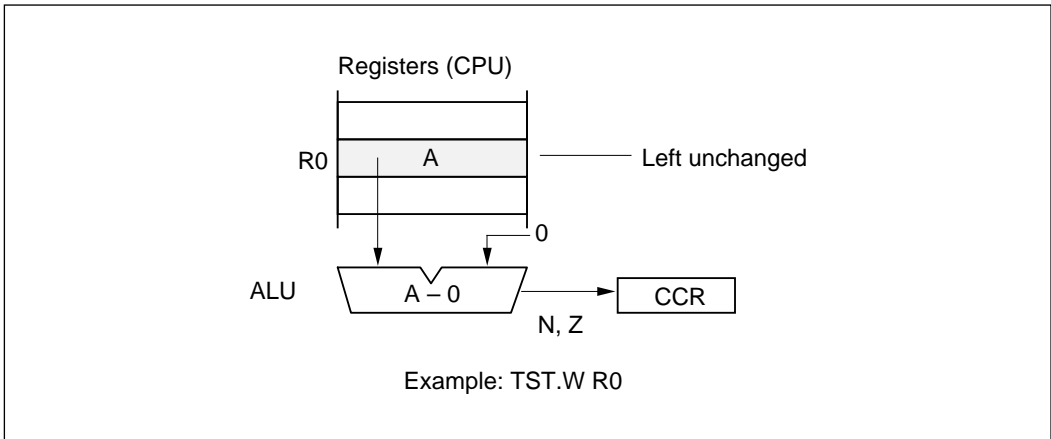
**Operation:** 0 → (<bits 15 to 8> of <Rd>)



**Instructions and Operand Sizes:** The operand size is always byte size.

**(14) TST Instruction (B/W):** Compares general register or memory contents with zero.

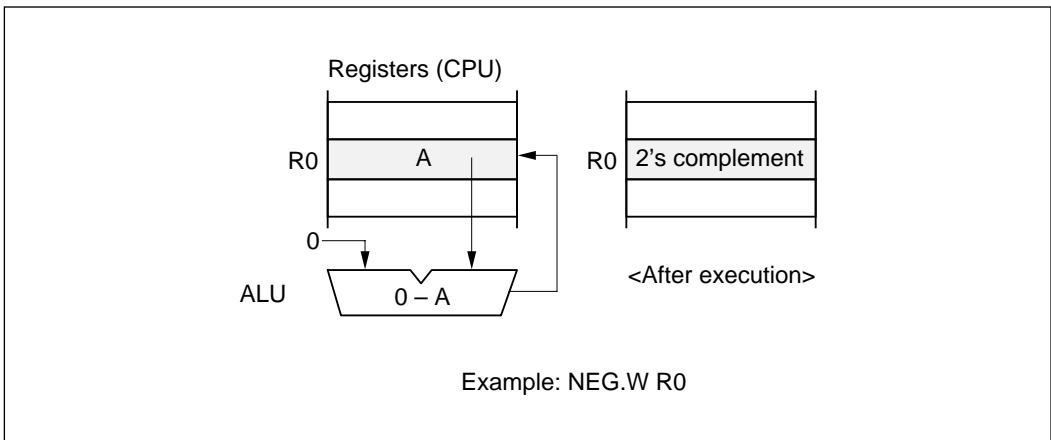
**Operation:** (EAd) - 0



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(15) NEG Instruction (B/W):** Obtains the two's complement of general register or memory contents.

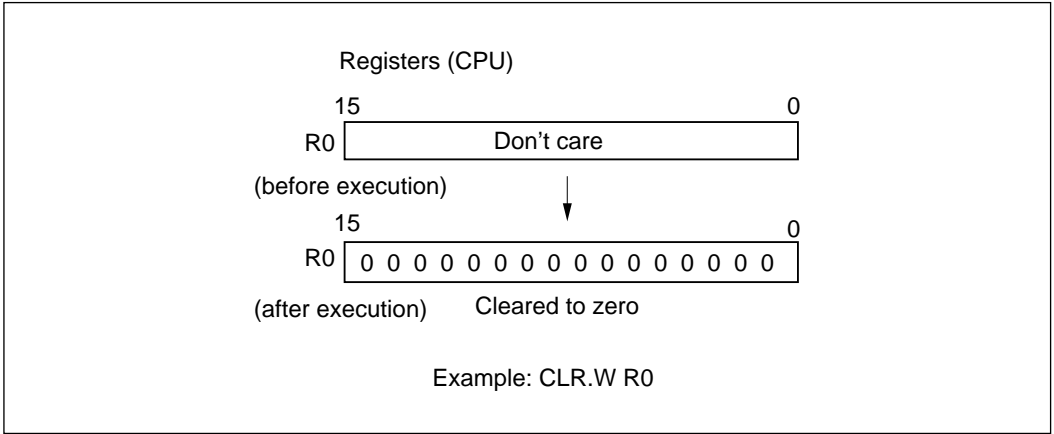
**Operation:** 0 - (EAd) → (EAd)



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(16) CLR Instruction (B/W):** Clears general register or memory contents to zero.

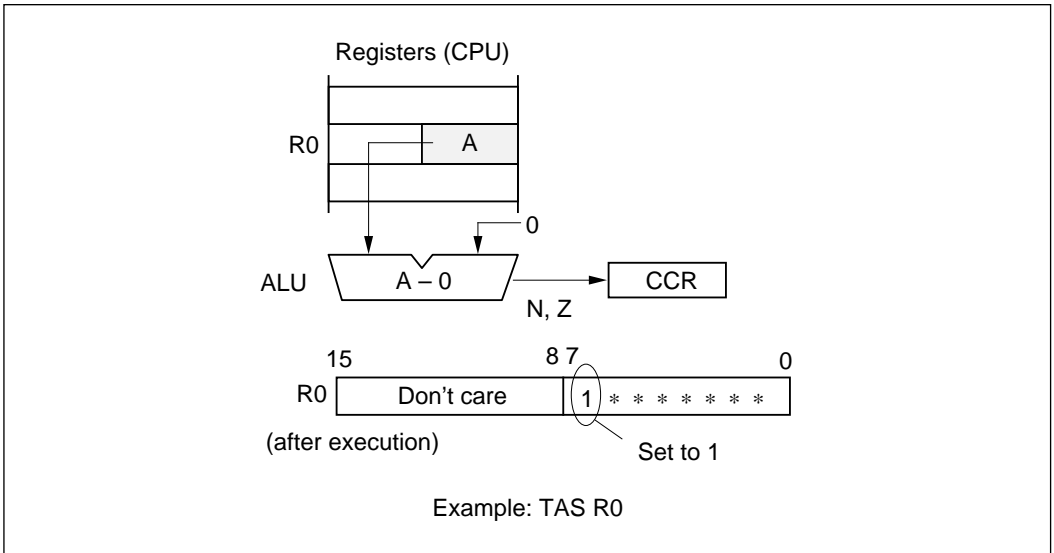
**Operation:**  $0 \rightarrow (EAd)$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(17) TAS Instruction (B):** Tests general register or memory contents, then sets the most significant bit (bit 7) to 1.

**Operation:**  $(EAd) - 0, (1)_2 \rightarrow (<bit 7> \text{ of } <EAd>)$



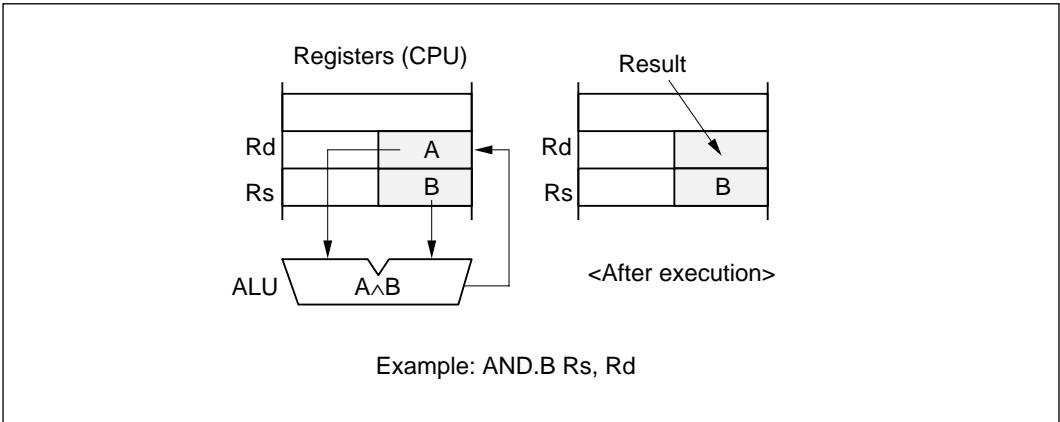
**Instructions and Operand Sizes:** The operand size is always byte size.

## A.5.6 Logic Instructions

There are four logic instructions. The function of each instruction is described next.

**(1) AND Instruction (B/W):** Performs a logical AND operation on a general register and another general register, memory, or immediate data.

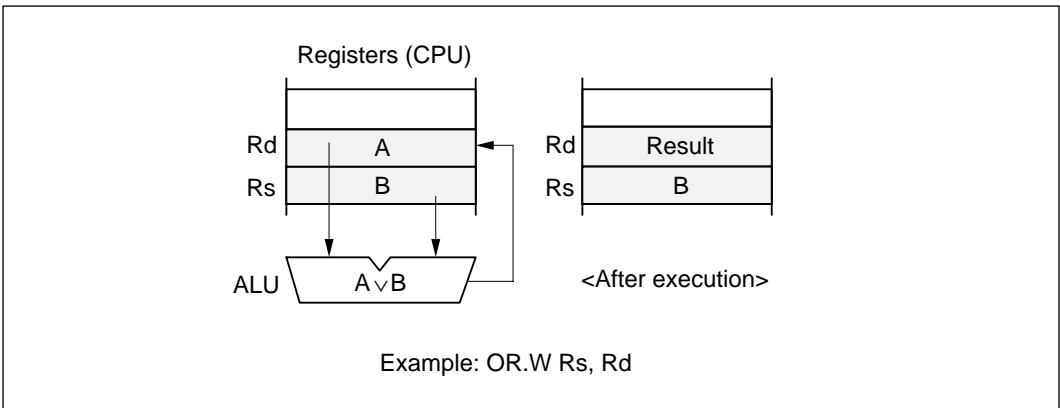
**Operation:**  $Rd \wedge (EAs) \rightarrow Rd$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(2) OR Instruction (B/W):** Performs a logical OR operation on a general register and another general register, memory, or immediate data.

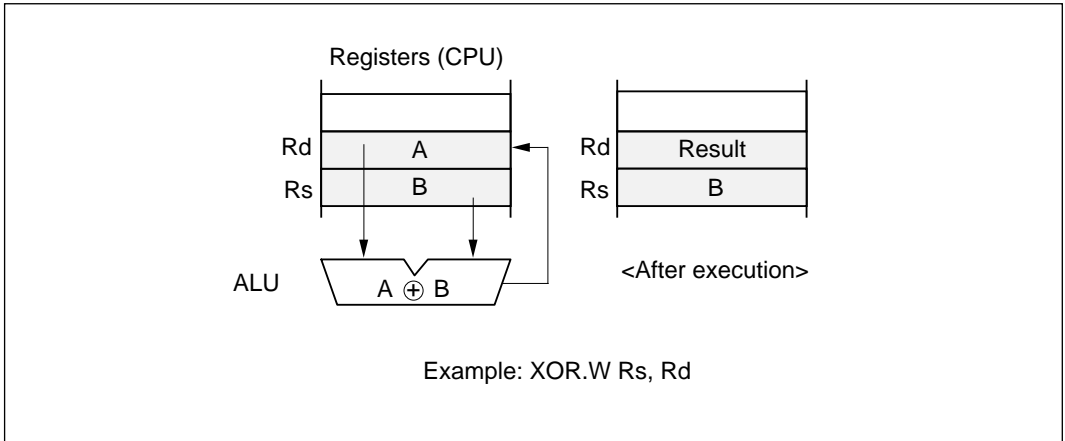
**Operation:**  $Rd \vee (EAs) \rightarrow Rd$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

(3) **XOR Instruction (B/W):** Performs a logical exclusive OR operation on a general register and another general register, memory, or immediate data.

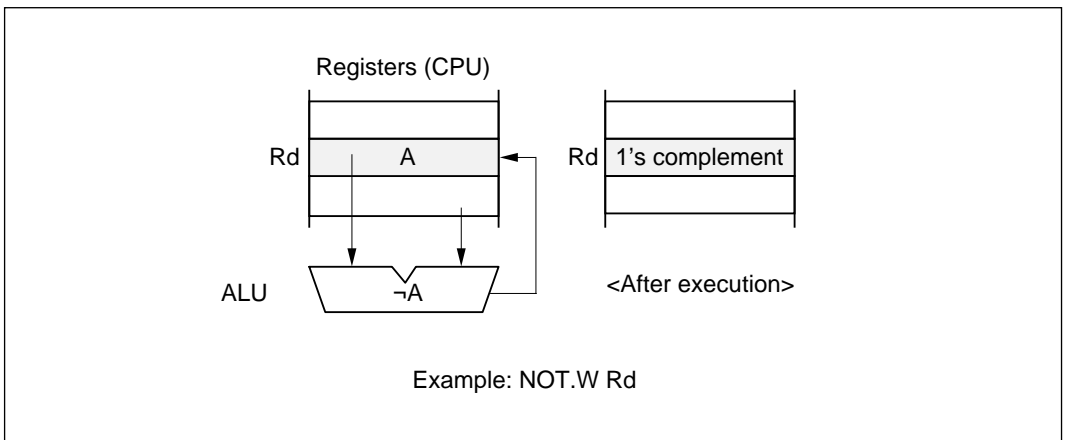
**Operation:**  $Rd \oplus (EAs) \rightarrow Rd$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

(4) **NOT Instruction (B/W):** Takes the one's complement of general register or memory contents.

**Operation:**  $\neg (EAd) \rightarrow (EAd)$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

### A.5.7 Shift Instructions

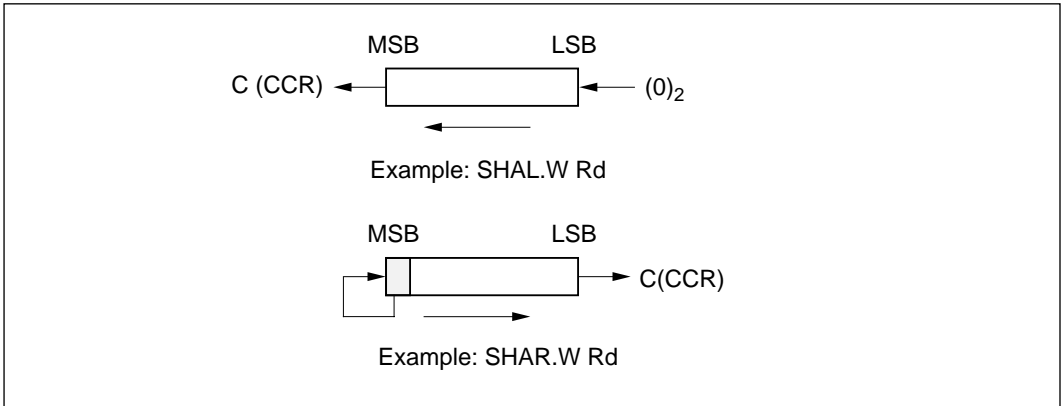
There are eight shift instructions. The function of each instruction is described next.

#### (1) SHAL Instruction (B/W)

#### (2) SHAR Instruction (B/W)

These instructions perform an arithmetic shift operation on general register or memory contents.

**Operation:** (EAd) arithmetic shift  $\rightarrow$  (EAd)



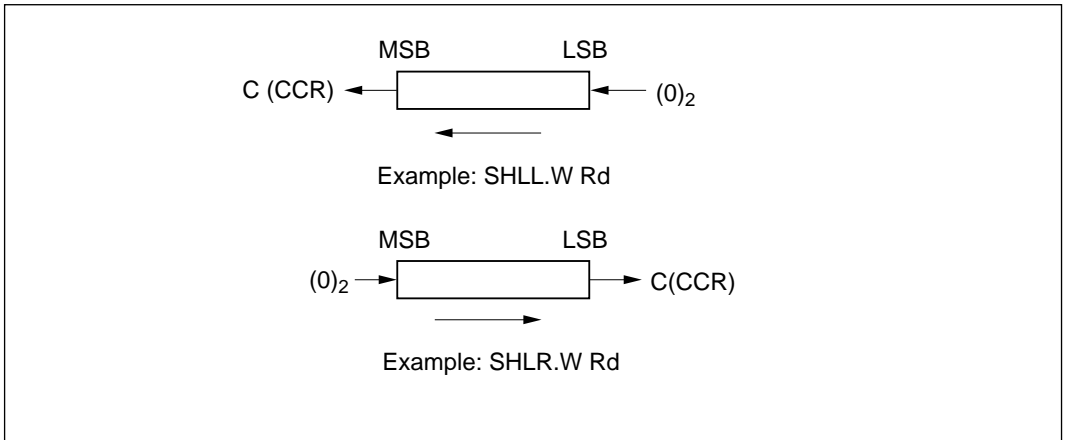
**Instructions and Operand Sizes:** Byte or word operand size can be selected.

#### (3) SHLL Instruction (B/W)

#### (4) SHLR Instruction (B/W)

These instructions perform a logic shift operation on general register or memory contents.

**Operation:** (EAd) logic shift → (EAd)



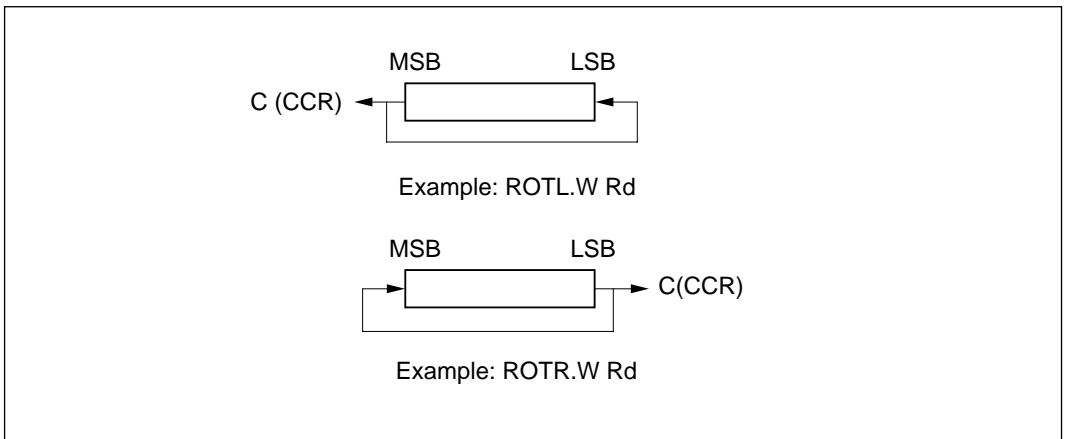
**Instructions and Operand Sizes:** Byte or word operand size can be selected.

(5) **ROTL Instruction (B/W)**

(6) **ROTR Instruction (B/W)**

These instructions rotate general register or memory contents.

**Operation:** (EAd) rotate → (EAd)



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

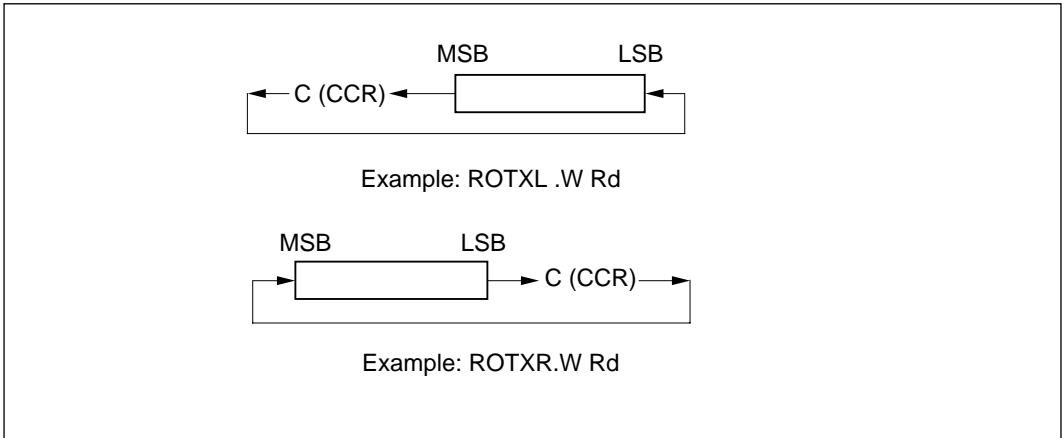


**(7) ROTXL Instruction (B/W)**

**(8) ROTXR Instruction (B/W)**

These instructions rotate general register or memory contents through the carry bit.

**Operation:** (EAd) rotate through carry → (EAd)



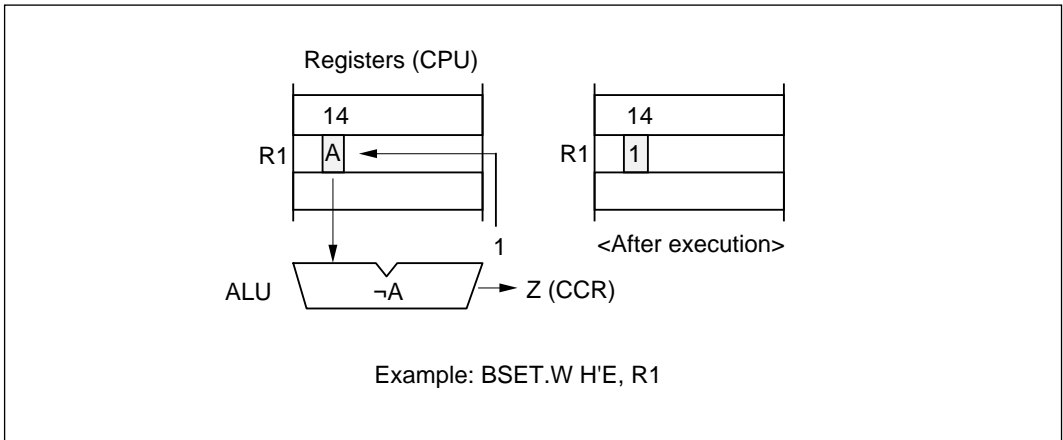
**Instructions and Operand Sizes:** Byte or word operand size can be selected.

### A.5.8 Bit Manipulation Instructions

There are four bit manipulation instructions. The function of each instruction is described next.

**(1) BSET Instruction (B/W):** Tests a specified bit in a general register or memory, then sets the bit to 1. The bit is specified by immediate data or a bit number in a general register.

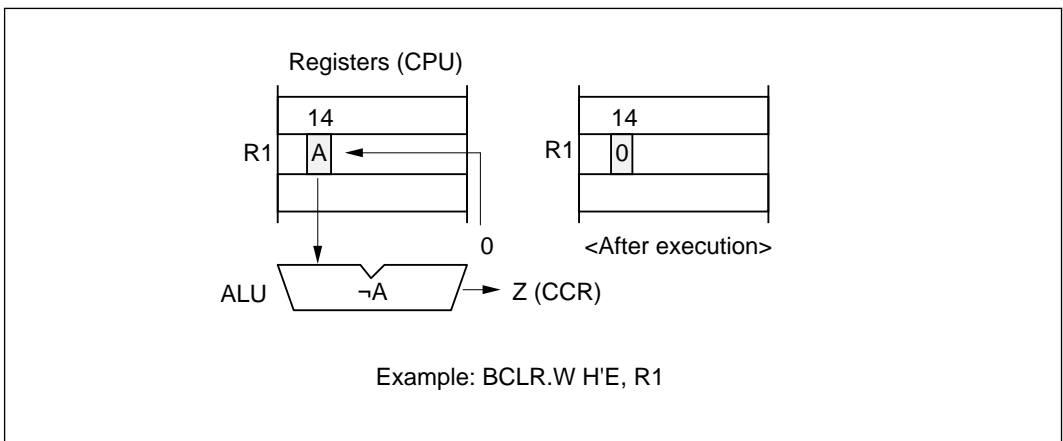
**Operation:**  $\neg(\text{<bit-No.> of <EAd>}) \rightarrow Z$   
 $1 \rightarrow (\text{<bit-No.> of <EAd>})$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(2) BCLR Instruction (B/W):** Tests a specified bit in a general register or memory, then clears the bit to 0. The bit is specified by immediate data or a bit number in a general register.

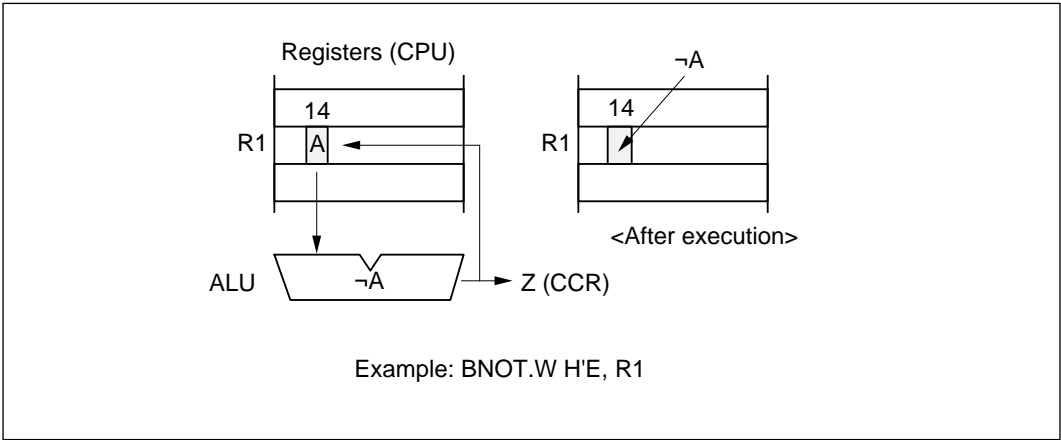
**Operation:**  $\neg(\text{<bit-No.> of <EAd>}) \rightarrow Z$   
 $0 \rightarrow (\text{<bit-No.> of <EAd>})$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(3) BNOT Instruction (B/W):** Tests a specified bit in a general register or memory, then inverts the bit. The bit is specified by immediate data or a bit number in a general register.

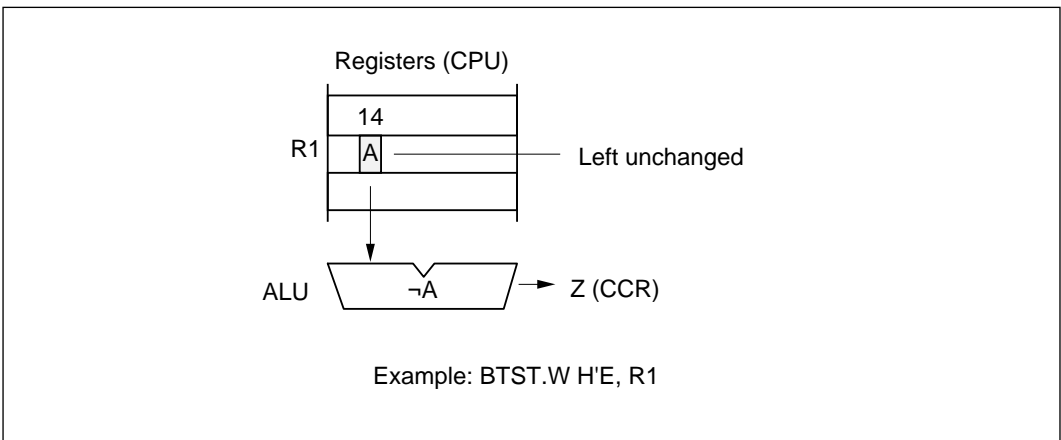
**Operation:**  $\neg(\langle\text{bit-No.}\rangle \text{ of } \langle\text{EAd}\rangle) \rightarrow Z \rightarrow (\langle\text{bit-No.}\rangle \text{ of } \langle\text{EAd}\rangle)$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

**(4) BTST Instruction (B/W):** Tests a specified bit in a general register or memory. The bit is specified by immediate data or a bit number in a general register.

**Operation:**  $\neg(\langle\text{bit-No.}\rangle \text{ of } \langle\text{EAd}\rangle) \rightarrow Z$



**Instructions and Operand Sizes:** Byte or word operand size can be selected.

### A.5.9 Branch Instructions

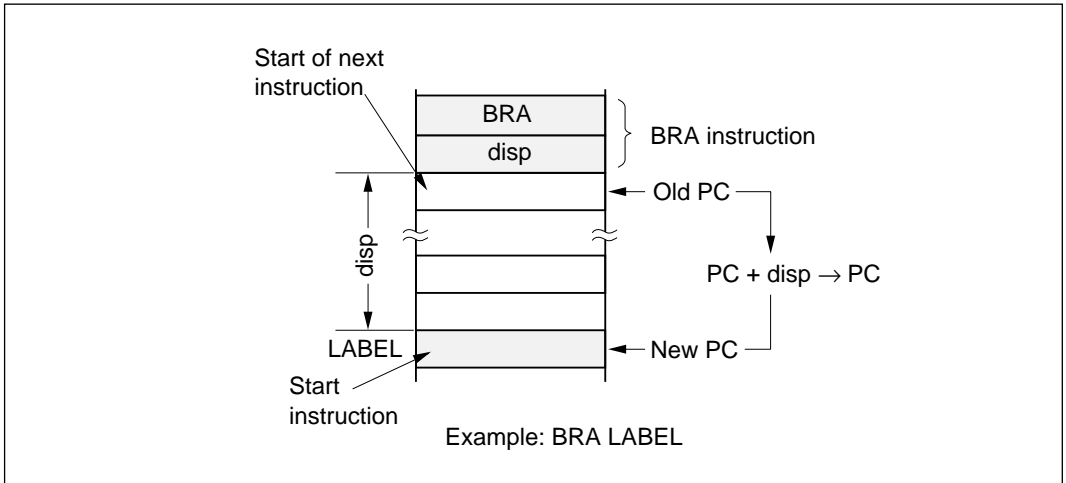
There are 11 branch instructions. The function of each instruction is described next.

(1) **Bcc Instruction (—):** Branches if the condition specified in the instruction is true.

**Operation:** If condition is true then

$$PC + \text{disp} \rightarrow PC$$

else next;



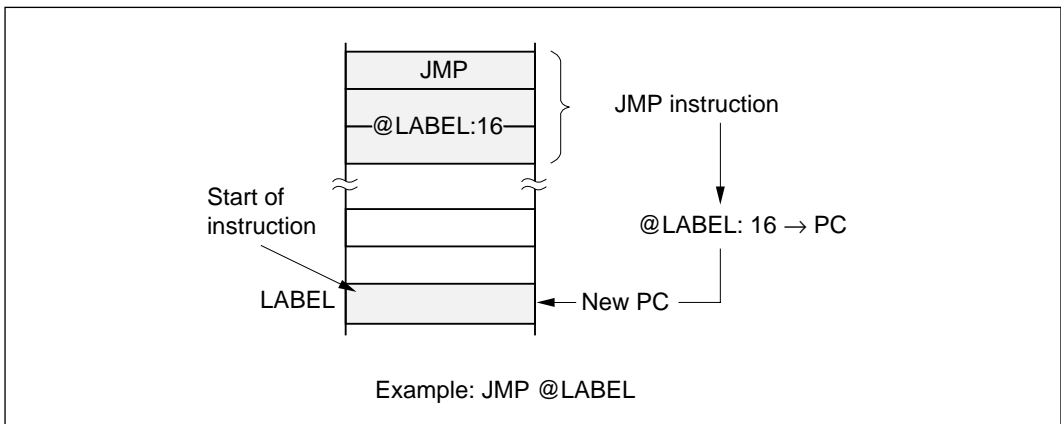
Note: This instruction cannot branch across a page boundary.

**Addressing of Branch Destination:** Specified by an eight-bit or 16-bit displacement.

Mnemonic	Description	Condition
BRA (BT)	Always (true)	True
BRN (BF)	Never (false)	False
BHI	High	$C \vee Z = 0$
BLS	Low or same	$C \vee Z = 1$
BCC (BHS)	Carry clear (high or same)	$C = 0$
BCS (BLO)	Carry set (low)	$C = 1$
BNE	Not equal	$Z = 0$
BEQ	Equal	$Z = 1$
BVC	Overflow clear	$V = 0$
BVS	Overflow set	$V = 1$
BPL	Plus	$N = 0$
BMI	Minus	$N = 1$
BGE	Greater or equal	$N \oplus V = 0$
BLT	Less than	$N \oplus V = 1$
BGT	Greater than	$Z \vee (N \oplus V) = 0$
BLE	Less or equal	$Z \vee (N \oplus V) = 1$

(2) **JMP Instruction** (—): Branches unconditionally to a specified address in the same page.

**Operation:**  $\langle EA \rangle \rightarrow PC$

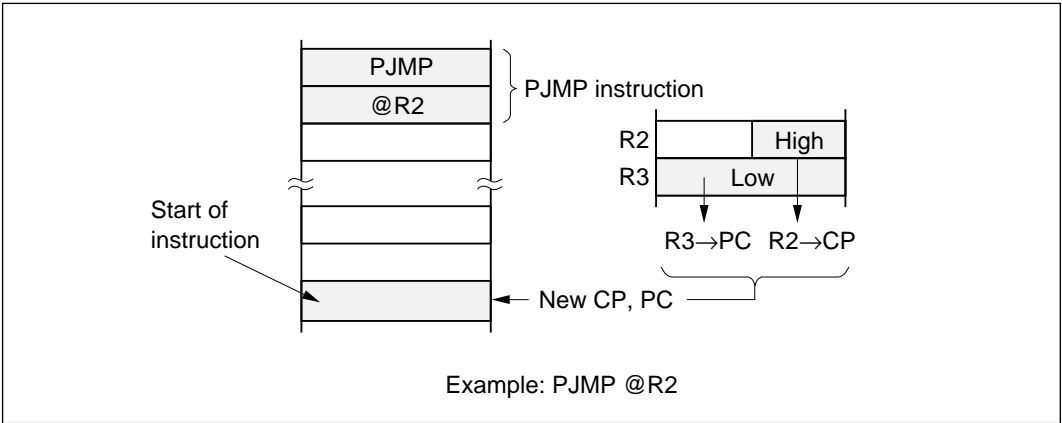


**Addressing of Branch Destination:** Register indirect, register indirect with eight-bit or 16-bit displacement, or 16-bit direct addressing.

Note: This instruction cannot branch across a page boundary.

(3) **PJMP Instruction** (—): Branches unconditionally to a specified address in a specified page.

**Operation:**  $\langle EA \rangle \rightarrow CP, PC$

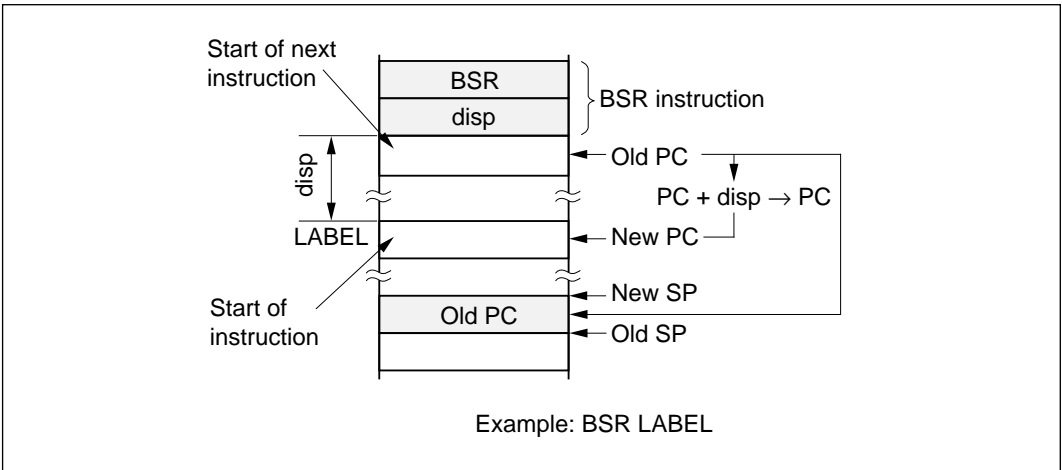


**Addressing of Branch Destination:** Register indirect or 24-bit direct addressing.

Note: This instruction is invalid in minimum mode.

(4) **BSR Instruction** (—): Branches to a subroutine at a specified address in the same page.

**Operation:**  $PC \rightarrow @-SP, PC + disp \rightarrow PC$

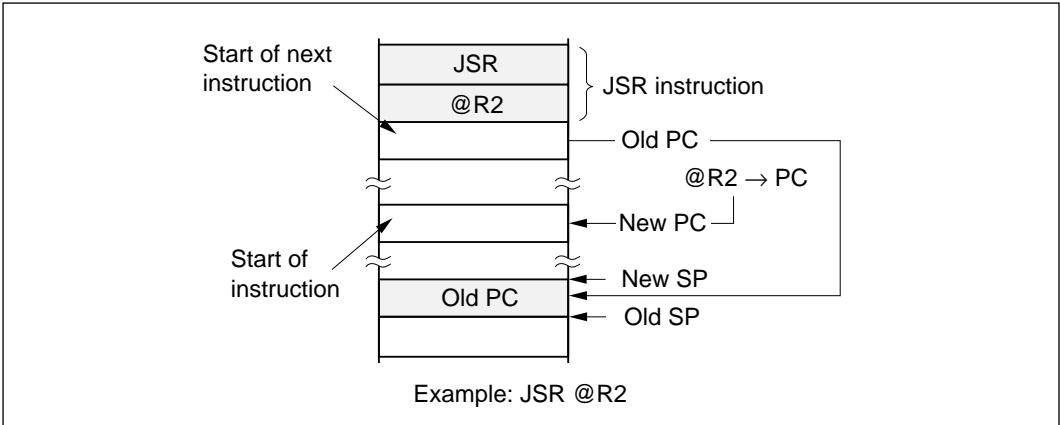


**Addressing of Branch Destination:** Specified by an eight-bit or 16-bit displacement.

Note: This instruction cannot branch across a page boundary.

(5) **JSR Instruction** (—): Branches to a subroutine at a specified address in the same page.

**Operation:**  $PC \rightarrow @-SP, \langle EA \rangle \rightarrow PC$

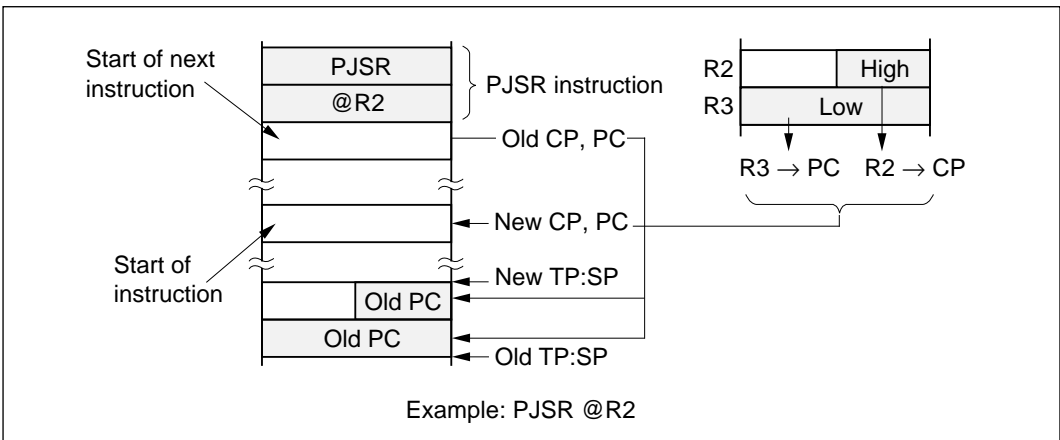


**Addressing of Branch Destination:** Register indirect, register indirect with eight-bit or 16-bit displacement, or 16-bit direct addressing.

Note: This instruction cannot branch across a page boundary.

(6) **PJSR Instruction** (—): Branches to a subroutine at a specified address in a specified page.

**Operation:**  $PC \rightarrow @-SP, CP \rightarrow @-SP, \langle EA \rangle \rightarrow PC$

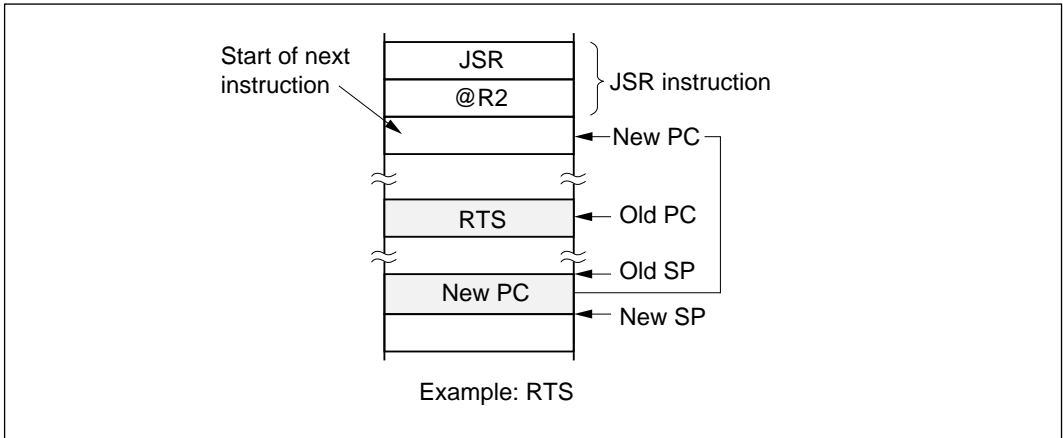


**Addressing of Branch Destination:** Register indirect or 24-bit direct addressing.

Note: This instruction is invalid in minimum mode.

(7) **RTS Instruction** (—): Returns from a subroutine in the same page.

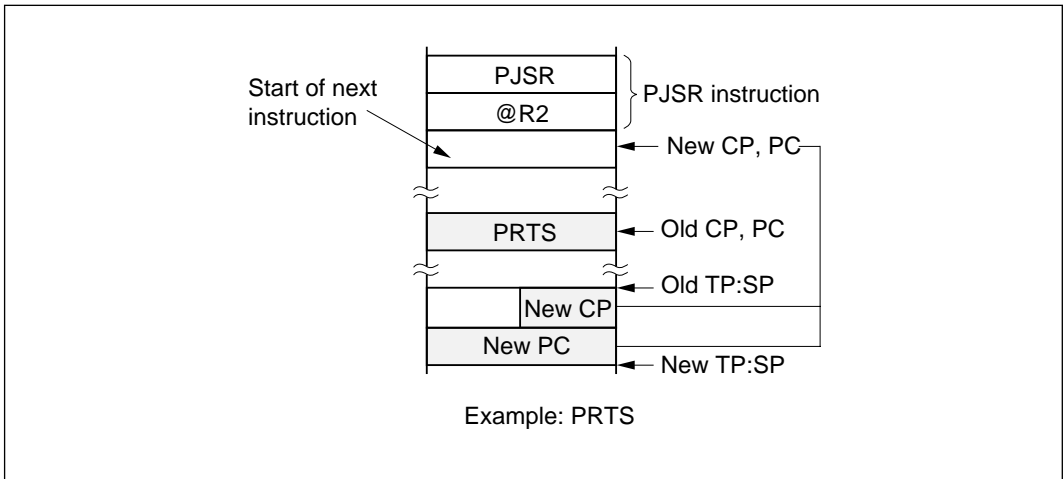
**Operation:** @SP+ → PC



RTS can return from a subroutine called by a BSR or JSR instruction.

(8) **PRTS Instruction** (—): Returns from a subroutine in another page.

**Operation:** @SP+ → PC, @SP+ → CP

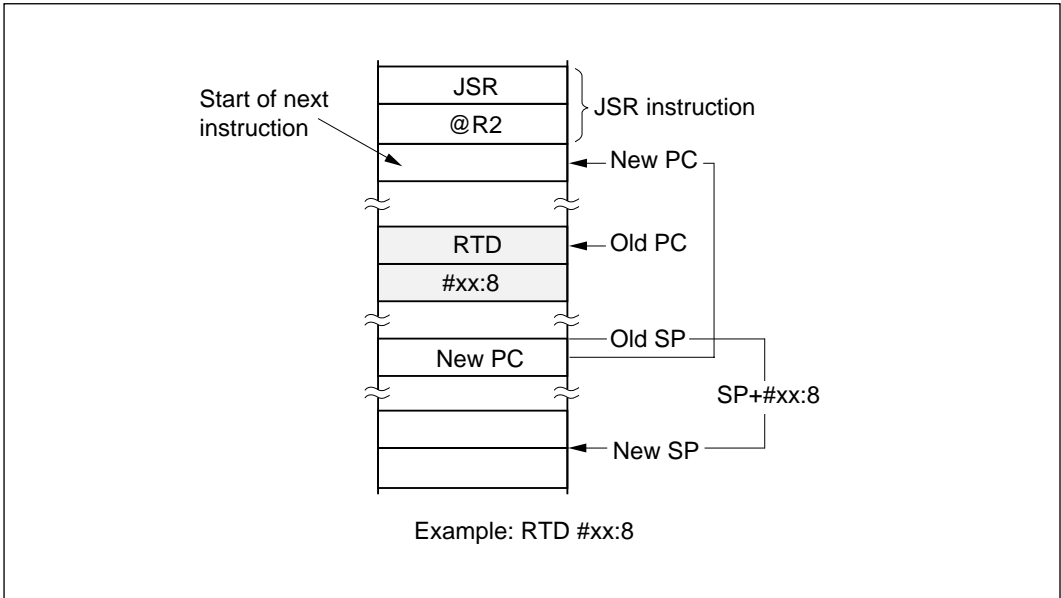


PRTS can return from a subroutine called by a PJSR instruction.



**(9) RTD Instruction (—):** Returns from a subroutine in the same page and adjusts the stack pointer.

**Operation:**  $@SP+ \rightarrow PC$ ,  $SP + \#IMM \rightarrow SP$

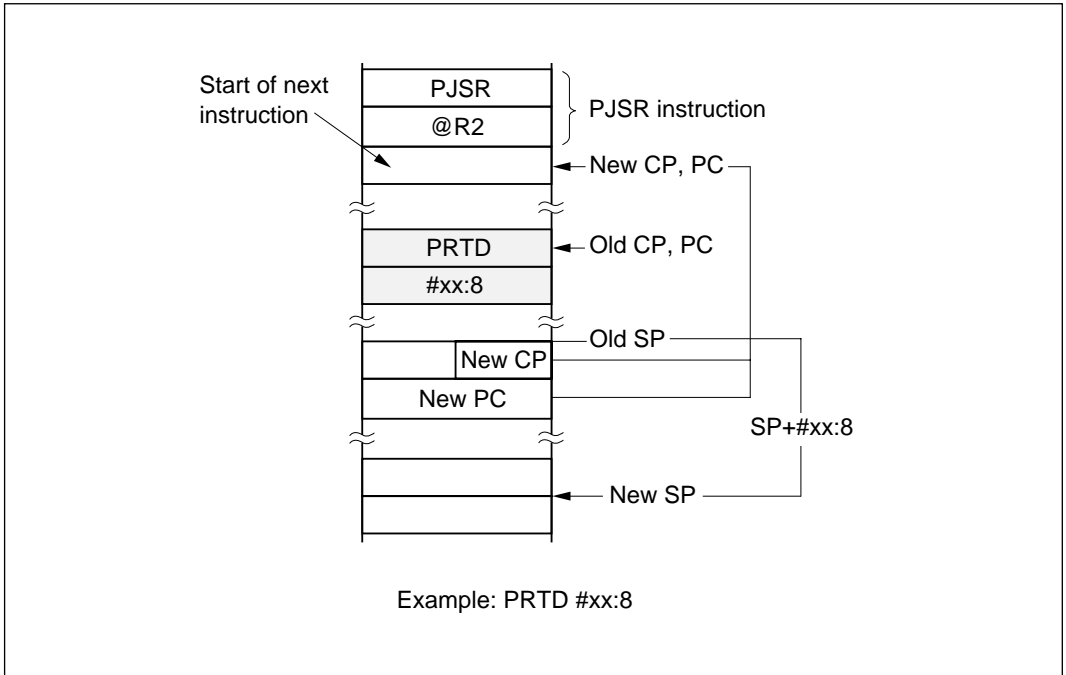


RTD can return from a subroutine called by a BSR or JSR instruction. The stack-pointer adjustment is specified by eight-bit or 16-bit immediate data.

**Note:** The immediate data must have an even value. If the stack pointer is set to an odd address, an address error will occur when the stack is accessed.

**(10) PRTD Instruction (—):** Returns from a subroutine in another page and adjusts the stack pointer.

**Operation:** @SP+ → PC, @SP+ → CP, SP + #IMM → SP

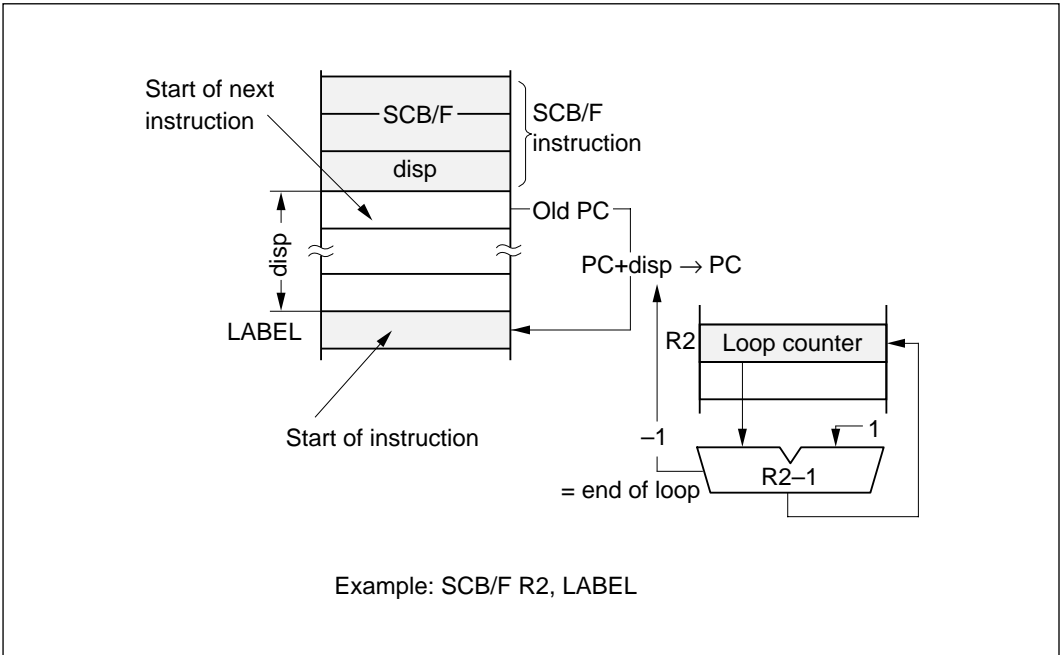


PRTD can return from a subroutine called by a BSR or JSR instruction. The stack-pointer adjustment is specified by eight-bit or 16-bit immediate data.

**Note:** The immediate data must have an even value. If the stack pointer is set to an odd address, an address error will occur when the stack is accessed.

**(11) SCB Instruction:** This is a loop instruction. The termination condition is specified by CCR and a loop counter.

**Operation:** If condition is true then next;  
 else  $R_n - 1 \rightarrow R_n$   
 If  $R_n = -1$  then next;  
 else  $PC + disp \rightarrow PC$



**Addressing of Branch Destination:** Specified by an eight-bit displacement.

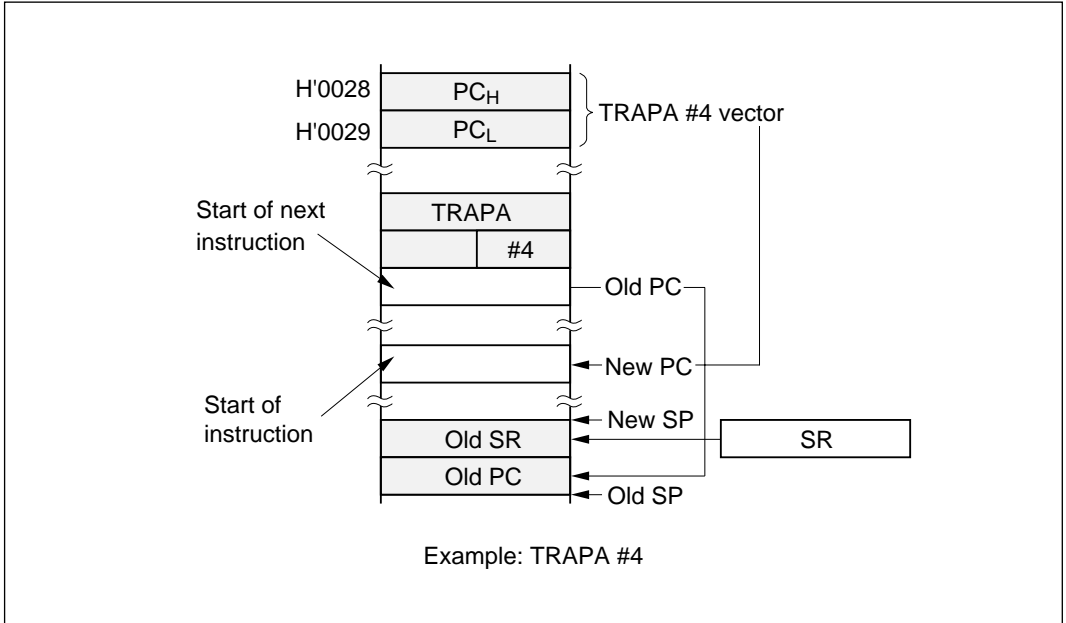
Instruction	Description	
	Function	Condition
SCB/F	False	—
SCB/NE	Not Equal	$Z = 0$
SCB/EQ	Equal	$Z = 1$

### A.5.10 System Control Instructions

There are 12 system control instructions. The function of each instruction is described next.

(1) **TRAPA Instruction** (—): Generates a trap exception with a specified vector number.

**Operation:** PC → @-SP, (maximum mode: CP → @-SP), SR → @-SP,  
 <vector> → PC, (maximum mode: <vector> → CP)



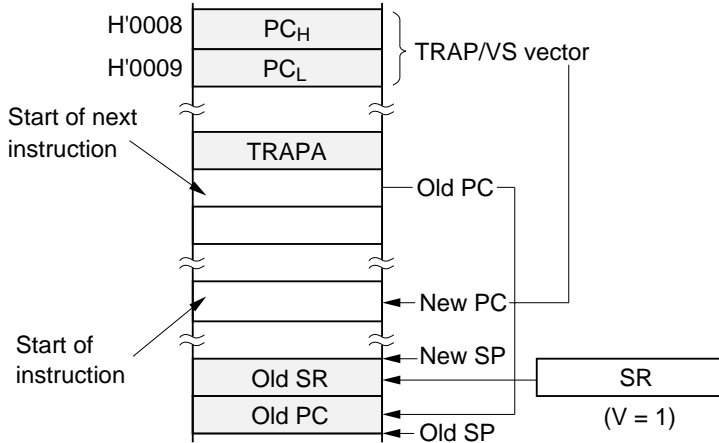
(2) **TRAP/VS Instruction** (—): Generates a trap exception if the V bit is set to 1.

**Operation:** If V bit of CCR = 1 then;

PC → @-SP, (maximum mode: CP → @-SP), SR → @-SP,

<vector> → PC, (maximum mode: <vector> → CP)

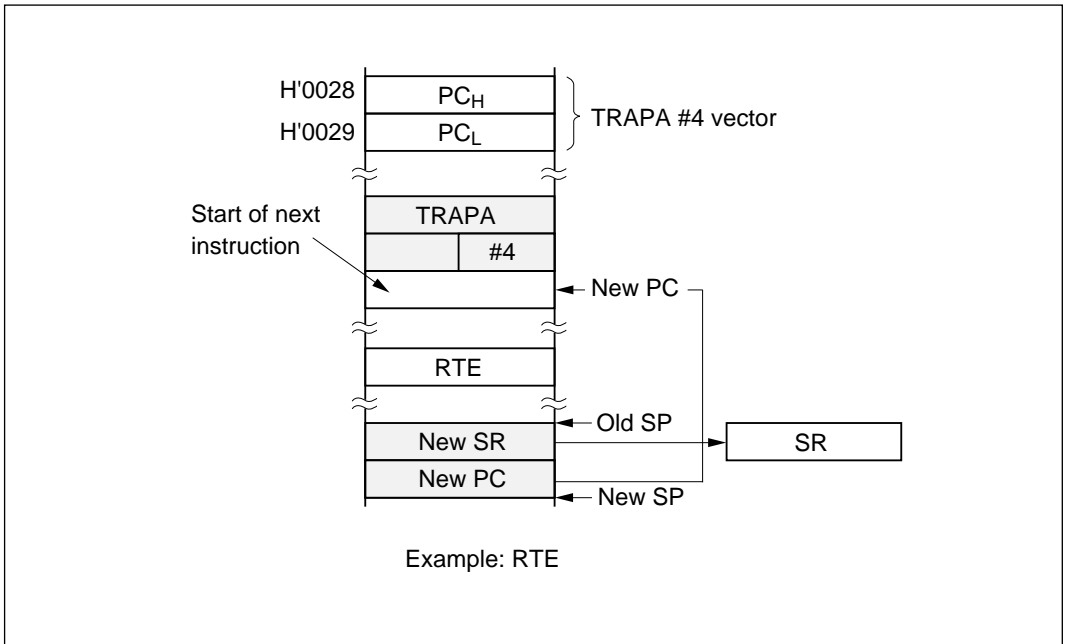
else next;



Example: TRAP/VS

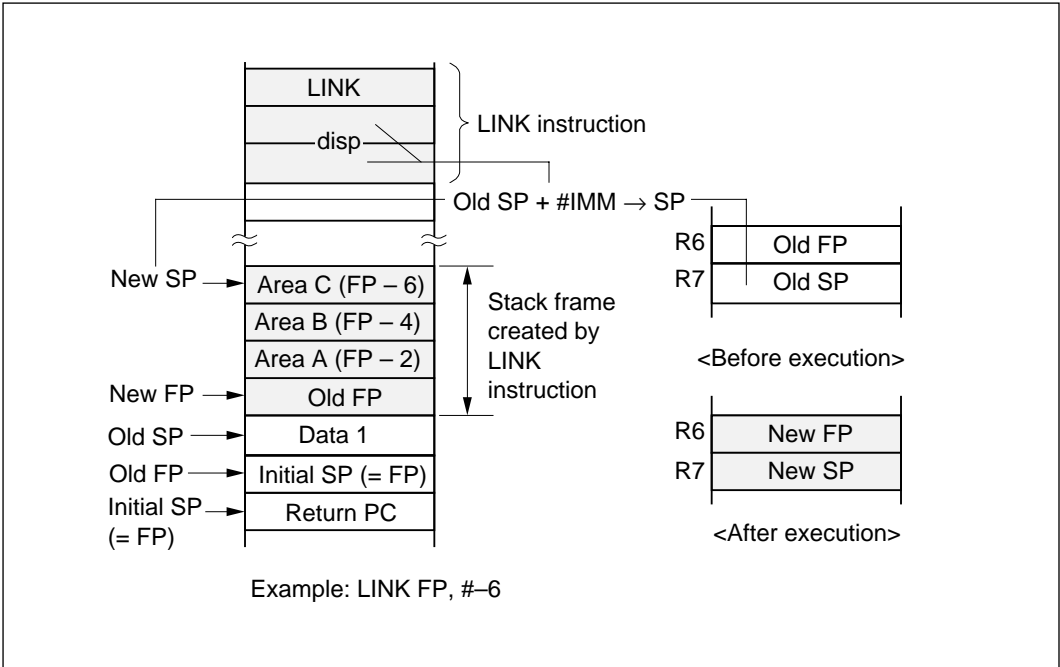
(3) **RTE Instruction (—)**: Returns from an exception-handling routine.

**Operation:** @SP+ → PC,  
(maximum mode: @SP+ → CP),  
@SP+ → SR



**(4) LINK Instruction (—):** Creates a stack frame.

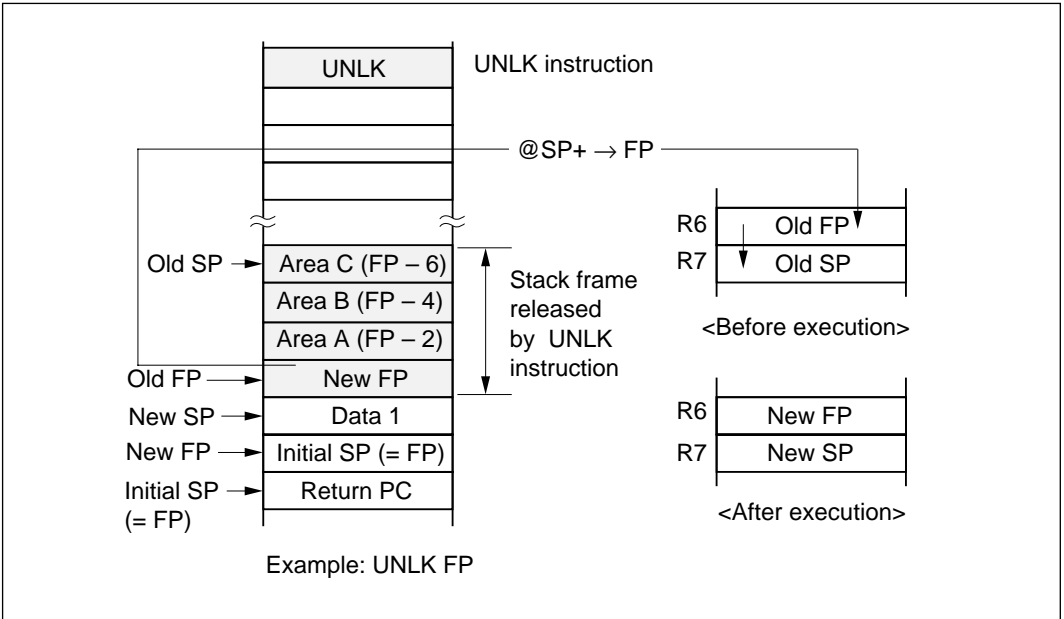
**Operation:** FP (R6) → @-SP,  
 SP → FP (R6),  
 SP + #IMM → SP



**Stack Frame Area:** Specified by eight-bit or 16-bit immediate data.

(5) **UNLK Instruction** (—): Releases a stack frame created by the LINK instruction.

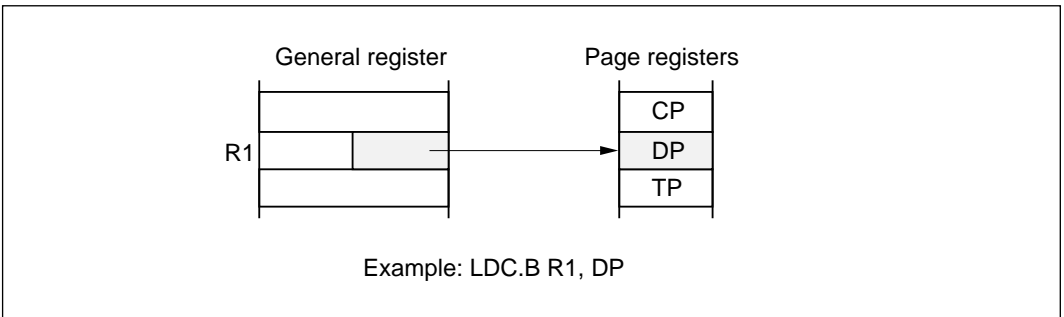
**Operation:** FP (R6) → SP,  
 @SP+ → FP (R6)



(6) **SLEEP Instruction** (—): Causes a transition to a power-down state.

(7) **LDC Instruction (B/W)**: Moves immediate data or general register or memory contents into a specified control register.

**Operation:** (EAs) → CR

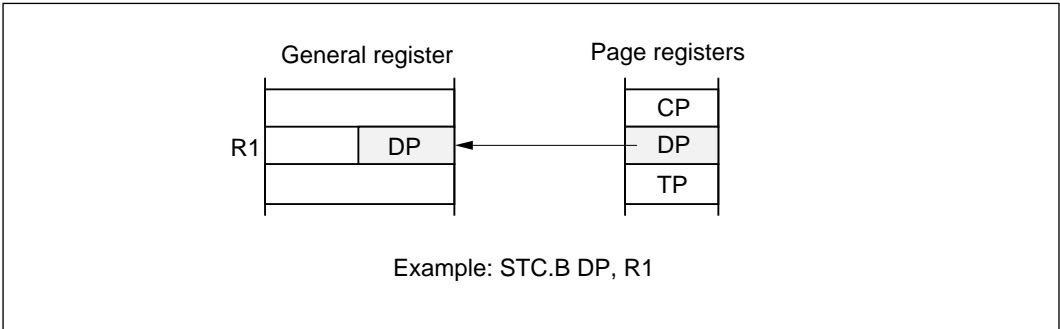


**Instructions and Operand Sizes:** The operand size depends on the control register.



**(8) STC Instruction (B/W):** Moves specified control register data to a general register or memory.

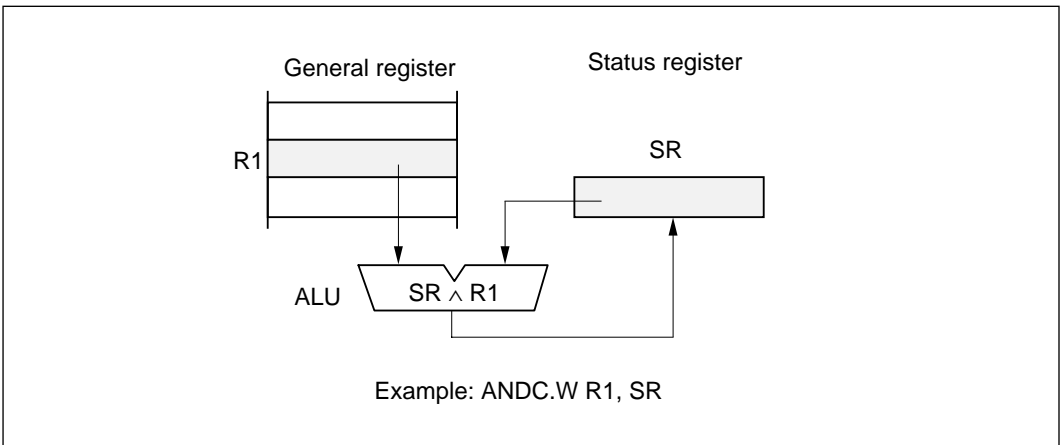
**Operation:** CR  $\rightarrow$  (EAd)



**Instructions and Operand Sizes:** The operand size depends on the control register.

**(9) ANDC Instruction (B/W):** Logically ANDs a control register with immediate data.

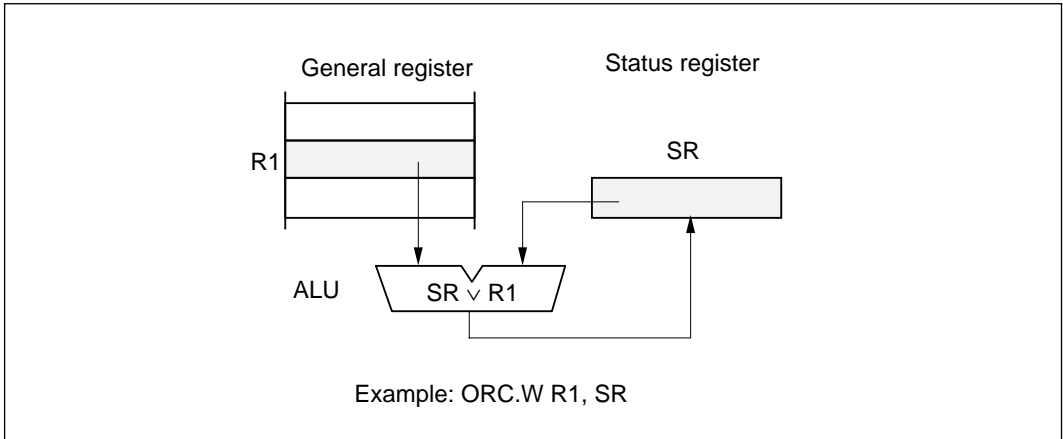
**Operation:** CR  $\wedge$  #IMM  $\rightarrow$  CR



**Instructions and Operand Sizes:** The operand size depends on the control register.

**(10) ORC Instruction (B/W):** Logically ORs a control register with immediate data.

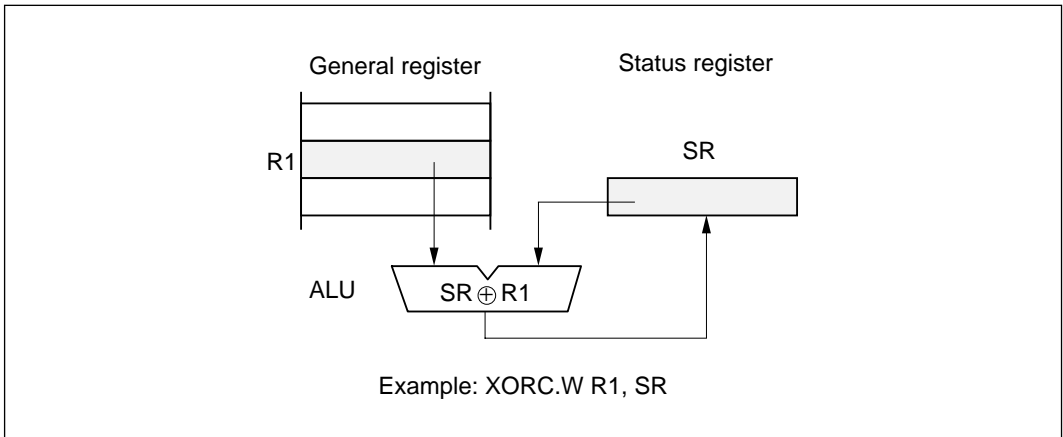
**Operation:**  $CR \vee \#IMM \rightarrow CR$



**Instructions and Operand Sizes:** The operand size depends on the control register.

**(11) XORC Instruction (B/W):** Logically exclusive-ORs a control register with immediate data.

**Operation:**  $CR \oplus \#IMM \rightarrow CR$



**Instructions and Operand Sizes:** The operand size depends on the control register.

**(12) NOP Instruction (—):** Only increments the program counter.

**Operation:**  $PC + 1 \rightarrow PC$

### A.5.11 Short-Format Instructions

The ADD, CMP, and MOV instructions have special short formats. The short formats are a byte shorter than the corresponding general formats, and most of them execute one state faster. Table A-12 lists these short formats together with the equivalent general formats.

**Table A-12 Short-Format Instructions and Equivalent General Formats**

Short-Format Instruction	Length	Execution States* <sup>2</sup>	Equivalent General-Format Instruction	Length	Execution States* <sup>2</sup>
ADD: Q #xx, Rd* <sup>1</sup>	2	2	ADD: G #xx: 8, Rd	3	3
CMP: E #xx: 8, Rd	2	2	CMP: G.B #xx: 8, Rd	3	3
CMP: I #xx: 16, Rd	3	3	CMP: G.W #xx: 16, Rd	4	4
MOV: E #xx: 8, Rd	2	2	MOV: G.B #xx: 8, Rd	3	3
MOV: I #xx: 16, Rd	3	3	MOV: G.W #xx: 16, Rd	4	4
MOV: L @aa: 8, Rd	2	5	MOV: G @aa: 8, Rd	3	5
MOV: S Rs, @aa: 8	2	5	MOV: G Rs, @aa: 8	3	5
MOV: F @ (d; 8, R6), Rd	2	5	MOV: G @ (d: 8, R6), Rd	3	5
MOV: F Rs, @ (d: 8, R6)	2	5	MOV: G Rs, @ (d: 8, R6)	3	5

Notes: 1. The ADD:Q instruction accepts other destination operands in addition to a general register.

2. Number of execution states for access to on-chip memory.

# Appendix B Initial Values of CPU Registers

**Table B-1 Register Values after Reset Exception Handling**

Register	Initial Value												
	Minimum Mode	Maximum Mode											
<div style="display: flex; justify-content: space-between; margin-bottom: 5px;"> <span>15</span> <span>0</span> </div> <table border="1" style="margin: auto; border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 10px;">R0</td></tr> <tr><td style="padding: 2px 10px;">R1</td></tr> <tr><td style="padding: 2px 10px;">R2</td></tr> <tr><td style="padding: 2px 10px;">R3</td></tr> <tr><td style="padding: 2px 10px;">R4</td></tr> <tr><td style="padding: 2px 10px;">R5</td></tr> <tr><td style="padding: 2px 10px;">R6 (FP)</td></tr> <tr><td style="padding: 2px 10px;">R7 (SP)</td></tr> </table>	R0	R1	R2	R3	R4	R5	R6 (FP)	R7 (SP)	Undetermined	Undetermined			
R0													
R1													
R2													
R3													
R4													
R5													
R6 (FP)													
R7 (SP)													
<div style="display: flex; justify-content: space-between; margin-bottom: 5px;"> <span>15</span> <span>0</span> </div> <table border="1" style="margin: auto; border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 10px;">PC</td></tr> </table>	PC	Loaded from vector table	Loaded from vector table										
PC													
<div style="display: flex; justify-content: space-between; margin-bottom: 5px;"> <span>15</span> <span>8</span> <span>7</span> <span>0</span> </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-bottom: 5px;"> <div style="text-align: center;">             SR  <span style="font-size: 0.8em;">┌───────────┴───────────┐</span> </div> <div style="text-align: center;">             CCR  <span style="font-size: 0.8em;">┌───┴───┐</span> </div> </div> <table border="1" style="margin: auto; border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px 5px;">T</td> <td style="padding: 2px 5px;">-----</td> <td style="padding: 2px 5px;">I<sub>2</sub></td> <td style="padding: 2px 5px;">I<sub>1</sub></td> <td style="padding: 2px 5px;">I<sub>0</sub></td> <td style="padding: 2px 5px;"> </td> <td style="padding: 2px 5px;">-----</td> <td style="padding: 2px 5px;">N</td> <td style="padding: 2px 5px;">V</td> <td style="padding: 2px 5px;">Z</td> <td style="padding: 2px 5px;">C</td> </tr> </table>	T	-----	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>		-----	N	V	Z	C	H'070*  * The last four bits (N, V, Z, and C) are undetermined.	H'070*  * The last four bits (N, V, Z, and C) are undetermined.
T	-----	I <sub>2</sub>	I <sub>1</sub>	I <sub>0</sub>		-----	N	V	Z	C			
<div style="display: flex; justify-content: space-between; margin-bottom: 5px;"> <span>7</span> <span>0</span> </div> <table border="1" style="margin: auto; border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 10px;">CP</td></tr> <tr><td style="padding: 2px 10px;">DP</td></tr> <tr><td style="padding: 2px 10px;">EP</td></tr> <tr><td style="padding: 2px 10px;">TP</td></tr> </table>	CP	DP	EP	TP	Undetermined	CP: loaded from vector table DP, EP, and TP: undetermined							
CP													
DP													
EP													
TP													
<div style="display: flex; justify-content: space-between; margin-bottom: 5px;"> <span>7</span> <span>0</span> </div> <table border="1" style="margin: auto; border-collapse: collapse; text-align: center;"> <tr><td style="padding: 2px 10px;">BR</td></tr> </table>	BR	Undetermined	Undetermined										
BR													

# Appendix C On-Chip Registers

## C.1 H8/539F Dual power source model

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FE80	Port 1	P1DDR	P1 <sub>7</sub> DDR	P1 <sub>6</sub> DDR	P1 <sub>5</sub> DDR	P1 <sub>4</sub> DDR	P1 <sub>3</sub> DDR	P1 <sub>2</sub> DDR	P1 <sub>1</sub> DDR	P1 <sub>0</sub> DDR	H'00
H'FE81	Port 2	P2DDR	P2 <sub>7</sub> DDR	P2 <sub>6</sub> DDR	P2 <sub>5</sub> DDR	P2 <sub>4</sub> DDR	P2 <sub>3</sub> DDR	P2 <sub>2</sub> DDR	P2 <sub>1</sub> DDR	P2 <sub>0</sub> DDR	H'00
H'FE82	Port 1	P1DR	P1 <sub>7</sub>	P1 <sub>6</sub>	P1 <sub>5</sub>	P1 <sub>4</sub>	P1 <sub>3</sub>	P1 <sub>2</sub>	P1 <sub>1</sub>	P1 <sub>0</sub>	H'00
H'FE83	Port 2	P2DR	P2 <sub>7</sub>	P2 <sub>6</sub>	P2 <sub>5</sub>	P2 <sub>4</sub>	P2 <sub>3</sub>	P2 <sub>2</sub>	P2 <sub>1</sub>	P2 <sub>0</sub>	H'00
H'FE84	Port 3	P3DDR	—	—	P3 <sub>5</sub> DDR	P3 <sub>4</sub> DDR	P3 <sub>3</sub> DDR	P3 <sub>2</sub> DDR	P3 <sub>1</sub> DDR	P3 <sub>0</sub> DDR	H'C0
H'FE85	Port 4	P4DDR	P4 <sub>7</sub> DDR	P4 <sub>6</sub> DDR	P4 <sub>5</sub> DDR	P4 <sub>4</sub> DDR	P4 <sub>3</sub> DDR	P4 <sub>2</sub> DDR	P4 <sub>1</sub> DDR	P4 <sub>0</sub> DDR	H'00
H'FE86	Port 3	P3DR	—	—	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>	H'C0
H'FE87	Port 4	P4DR	P4 <sub>7</sub>	P4 <sub>6</sub>	P4 <sub>5</sub>	P4 <sub>4</sub>	P4 <sub>3</sub>	P4 <sub>2</sub>	P4 <sub>1</sub>	P4 <sub>0</sub>	H'00
H'FE88	Port 5	P5DDR	P5 <sub>7</sub> DDR	P5 <sub>6</sub> DDR	P5 <sub>5</sub> DDR	P5 <sub>4</sub> DDR	P5 <sub>3</sub> DDR	P5 <sub>2</sub> DDR	P5 <sub>1</sub> DDR	P5 <sub>0</sub> DDR	H'00
H'FE89	Port 6	P6DDR	—	—	—	P6 <sub>4</sub> DDR	P6 <sub>3</sub> DDR	P6 <sub>2</sub> DDR	P6 <sub>1</sub> DDR	P6 <sub>0</sub> DDR	H'E0
H'FE8A	Port 5	P5DR	P5 <sub>7</sub>	P5 <sub>6</sub>	P5 <sub>5</sub>	P5 <sub>4</sub>	P5 <sub>3</sub>	P5 <sub>2</sub>	P5 <sub>1</sub>	P5 <sub>0</sub>	H'00
H'FE8B	Port 6	P6DR	—	—	—	P6 <sub>4</sub>	P6 <sub>3</sub>	P6 <sub>2</sub>	P6 <sub>1</sub>	P6 <sub>0</sub>	H'E0
H'FE8C	Port 7	P7DDR	P7 <sub>7</sub> DDR	P7 <sub>6</sub> DDR	P7 <sub>5</sub> DDR	P7 <sub>4</sub> DDR	P7 <sub>3</sub> DDR	P7 <sub>2</sub> DDR	P7 <sub>1</sub> DDR	P7 <sub>0</sub> DDR	H'00
H'FE8D			—	—	—	—	—	—	—	—	H'FF
H'FE8E	Port 7	P7DR	P7 <sub>7</sub>	P7 <sub>6</sub>	P7 <sub>5</sub>	P7 <sub>4</sub>	P7 <sub>3</sub>	P7 <sub>2</sub>	P7 <sub>1</sub>	P7 <sub>0</sub>	H'00
H'FE8F	Port 8	P8DR	—	—	—	—	P8 <sub>3</sub>	P8 <sub>2</sub>	P8 <sub>1</sub>	P8 <sub>0</sub>	Undetermined

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FE90			—	—	—	—	—	—	—	—	H'FF
H'FE91	Port A	PADDR	—	PA <sub>6</sub> DDR	PA <sub>5</sub> DDR	PA <sub>4</sub> DDR	PA <sub>3</sub> DDR	PA <sub>2</sub> DDR	PA <sub>1</sub> DDR	PA <sub>0</sub> DDR	H'80
H'FE92	Port 9	P9DR	P9 <sub>7</sub>	P9 <sub>6</sub>	P9 <sub>5</sub>	P9 <sub>4</sub>	P9 <sub>3</sub>	P9 <sub>2</sub>	P9 <sub>1</sub>	P9 <sub>0</sub>	Undetermined
H'FE93	Port A	PADR	—	PA <sub>6</sub>	PA <sub>5</sub>	PA <sub>4</sub>	PA <sub>3</sub>	PA <sub>2</sub>	PA <sub>1</sub>	PA <sub>0</sub>	H'80
H'FE94	Port B	PBDDR	PB <sub>7</sub> DDR	PB <sub>6</sub> DDR	PB <sub>5</sub> DDR	PB <sub>4</sub> DDR	PB <sub>3</sub> DDR	PB <sub>2</sub> DDR	PB <sub>1</sub> DDR	PB <sub>0</sub> DDR	H'00
H'FE95	Port C	PCDDR	PC <sub>7</sub> DDR	PC <sub>6</sub> DDR	PC <sub>5</sub> DDR	PC <sub>4</sub> DDR	PC <sub>3</sub> DDR	PC <sub>2</sub> DDR	PC <sub>1</sub> DDR	PC <sub>0</sub> DDR	H'00
H'FE96	Port B	PBDR	PB <sub>7</sub>	PB <sub>6</sub>	PB <sub>5</sub>	PB <sub>4</sub>	PB <sub>3</sub>	PB <sub>2</sub>	PB <sub>1</sub>	PB <sub>0</sub>	H'00
H'FE97	Port C	PCDR	PC <sub>7</sub>	PC <sub>6</sub>	PC <sub>5</sub>	PC <sub>4</sub>	PC <sub>3</sub>	PC <sub>2</sub>	PC <sub>1</sub>	PC <sub>0</sub>	H'00
H'FE98	Port B	PBPCR	PB <sub>7</sub> PON	PB <sub>6</sub> PON	PB <sub>5</sub> PON	PB <sub>4</sub> PON	PB <sub>3</sub> PON	PB <sub>2</sub> PON	PB <sub>1</sub> PON	PB <sub>0</sub> PON	H'00
H'FE99	Port C	PCPCR	PC <sub>7</sub> PON	PC <sub>6</sub> PON	PC <sub>5</sub> PON	PC <sub>4</sub> PON	PC <sub>3</sub> PON	PC <sub>2</sub> PON	PC <sub>1</sub> PON	PC <sub>0</sub> PON	H'00
H'FE9A	ø <sub>CR</sub>	ø <sub>CR</sub>	ø <sub>OE</sub>	—	—	—	—	—	—	—	Undefined*
H'FE9B			—	—	—	—	—	—	—	—	H'FF
H'FE9C			—	—	—	—	—	—	—	—	H'FF
H'FE9D			—	—	—	—	—	—	—	—	H'FF
H'FE9E			—	—	—	—	—	—	—	—	H'FF
H'FE9F			—	—	—	—	—	—	—	—	H'FF

(continued on next page)

Note: \* Initialized to H'FF in standby mode.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEA0	A/D	ADDR0H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA1		ADDR0L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA2		ADDR1H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA3		ADDR1L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA4		ADDR2H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA5		ADDR2L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA6		ADDR3H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA7		ADDR3L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA8		ADDR4H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA9		ADDR4L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEAA		ADDR5H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEAB		ADDR5L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEAC		ADDR6H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEAD		ADDR6L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEAE		ADDR7H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEAF		ADDR7L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00

**Legend**

A/D: A/D converter

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEB0	A/D	ADDR8H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB1		ADDR8L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB2		ADDR9H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB3		ADDR9L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB4		ADDRAH	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB5		ADDRAL	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB6		ADDRBH	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB7		ADDRBL	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB8		ADCSR	ADF	ADIE	ADM1	ADM0	CH3	CH2	CH1	CH0	H'00
H'FEB9		ADCR	TRGE	CKS	ADST	—	—	—	—	—	H'1F
H'FEBA			—	—	—	—	—	—	—	—	H'FF
H'FEBB			—	—	—	—	—	—	—	—	H'FF
H'FEBC			—	—	—	—	—	—	—	—	H'FF
H'FEBD			—	—	—	—	—	—	—	—	H'FF
H'FEBE			—	—	—	—	—	—	—	—	H'FF
H'FEBF			—	—	—	—	—	—	—	—	H'FF

**Legend**

A/D: A/D converter

(continued on next page)



(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEC0	SCI3	SMR	C/ $\bar{A}$	CHR	PE	O/ $\bar{E}$	STOP	MP	CKS1	CKS0	H'00
H'FEC1		BRR									H'FF
H'FEC2		SCR	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	H'00
H'FEC3		TDR									H'FF
H'FEC4		SSR	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	H'84F
H'FEC5		RDR									H'00
H'FEC6				—	—	—	—	—	—	—	—
H'FEC7			—	—	—	—	—	—	—	—	Undetermined
H'FEC8	SCI1	SMR	C/ $\bar{A}$	CHR	PE	O/ $\bar{E}$	STOP	MP	CKS1	CKS0	H'00
H'FEC9		BRR									H'FF
H'FECA		SCR	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	H'00
H'FECEB		TDR									H'FF
H'FECC		SSR	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	H'84
H'FECD		RDR									H'00
H'FECE				—	—	—	—	—	—	—	—
H'FECE			—	—	—	—	—	—	—	—	Undetermined

**Legend**

SCI1: Serial communication interface 1

SCI3: Serial communication interface 3

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FED0	SCI2	SMR	C/A	CHR	PE	O/E	STOP	MP	CKS1	CKS0	H'00
H'FED1		BRR									H'FF
H'FED2		SCR	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	H'00
H'FED3		TDR									H'FF
H'FED4		SSR	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	H'84
H'FED5		RDR									H'00
H'FED6			—	—	—	—	—	—	—	—	H'FF
H'FED7			—	—	—	—	—	—	—	—	Undetermined
H'FED8			—	—	—	—	—	—	—	—	H'FF
H'FED9			—	—	—	—	—	—	—	—	H'FF
H'FEDA	Port A	PACR	—	TXD3E	RXD3E	—	SCK3E	PW3E	PW2E	PW1E	H'90
H'FEDB	Port 6/7	P67CR	PW2E	PW1E	—	—	—	—	—	PW3E	H'3E
H'FEDC	A/D	ADTRGR	EXTRG	—	—	—	—	—	—	—	H'FF
H'FEDD			—	—	—	—	—	—	—	—	H'FF
H'FEDE	INTC	IRQFR	—	—	—	—	IRQ3F	IRQ2F	IRQ1F	—	H'F1
H'FEDF	BSC	BCR	BCRE	0P3T	—	P9AE	EXIOP	PCRE	PBCE	P12E	H'3F*

**Legend**

SCI2: Serial communication interface 2

INTC: Interrupt controller

BSC: Bus controller

A/D: A/D converter

(continued on next page)

Note: \* Initial value in modes 5 and 6. In modes 1 to 4 and mode 7 the initial value is H'BF.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value	
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		
H'FEE0	Flash memory	FLMCR	V <sub>PP</sub>	V <sub>PP</sub> E	—	—	EV	PV	E	P	H'00*	
H'FEE1		—	—	—	—	—	—	—	—	—	H'FF	
H'FEE2		EBR1	LB7	LB6	LB5	LB4	LB3	LB2	LB1	LB0	H'00	
H'FEE3		EBR2	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0	H'00	
H'FEE4		—	—	—	—	—	—	—	—	—	H'FF	
H'FEE5		—	—	—	—	—	—	—	—	—	H'FF	
H'FEE6		—	—	—	—	—	—	—	—	—	H'FF	
H'FEE7		—	—	—	—	—	—	—	—	—	H'FF	
H'FEE8		—	—	—	—	—	—	—	—	—	H'FF	
H'FEE9		—	—	—	—	—	—	—	—	—	H'FF	
H'FEEA		—	—	—	—	—	—	—	—	—	H'FF	
H'FEEB		—	—	—	—	—	—	—	—	—	H'FF	
H'FEEC		Flash memory	FLMER	OVLP	—	—	—	A11E	A10E	A9E	—	H'71
H'FEED			FLMSR	FLER	—	—	—	—	—	—	—	H'7F
H'FEEE			—	—	—	—	—	—	—	—	—	H'FF
H'FE EF			—	—	—	—	—	—	—	—	—	H'FF

(continued on next page)

Note: \* When 12 V is being applied to the V<sub>PP</sub> pin, the initial value is H'80.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEF0	PWM1	TCR	OE	OS	—	—	—	CKS2	CKS1	CKS0	H'38
H'FEF1		DTR									H'FF
H'FEF2		TCNT									H'00
H'FEF3			—	—	—	—	—	—	—	—	H'FF
H'FEF4	PWM2	TCR	OE	OS	—	—	—	CKS2	CKS1	CKS0	H'38
H'FEF5		DTR									H'FF
H'FEF6		TCNT									H'00
H'FEF7			—	—	—	—	—	—	—	—	H'FF
H'FEF8	PWM3	TCR	OE	OS	—	—	—	CKS2	CKS1	CKS0	H'38
H'FEF9		DTR									H'FF
H'FEFA		TCNT									H'00
H'FEFB			—	—	—	—	—	—	—	—	H'FF
H'FEFC			—	—	—	—	—	—	—	—	H'FF
H'FEFD			—	—	—	—	—	—	—	—	H'FF
H'FEFE			—	—	—	—	—	—	—	—	H'FF
H'FEFF			—	—	—	—	—	—	—	—	H'FF

**Legend**

(continued on next page)

PWM1: Pulse width modulation timer 1

PWM2: Pulse width modulation timer 2

PWM3: Pulse width modulation timer 3

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF00	INTC	IPRA	0				0				H'00
H'FF01		IPRB	0				0				H'00
H'FF02		IPRC	0				0				H'00
H'FF03		IPRD	0				0				H'00
H'FF04		IPRE	0				0				H'00
H'FF05		IPRF	0				0				H'00
H'FF06	DTC		—	—	—	—	—	—	—	—	Undetermined
H'FF07			—	—	—	—	—	—	—	—	Undetermined
H'FF08		DTEA	0	ADI	(IRQ0)	IRQ0	0	IRQ3	IRQ2	IRQ1	H'00
H'FF09		DTEB	0	T1CMI1,2	T1IMI2	T1IMI1	0	T1CMI3,4	T1IMI4	T1IMI3	H'00
H'FF0A		DTEC	0	T2CMI1,2	T2IMI2	T2IMI1	0	T3CMI1,2	T3IMI2	T3IMI1	H'00
H'FF0B		DTED	0	T4CMI1,2	T4IMI2	T4IMI1	0	T5CMI1,2	T5IMI2	T5IMI1	H'00
H'FF0C		DTEE	0	0	T6IMI2	T6IMI1	0	0	T7IMI2	T7IMI1	H'00
H'FF0D		DTEF	0	TI1	RI1	0	0	TI2	RI2	0	H'00
H'FF0E				—	—	—	—	—	—	—	Undetermined
H'FF0F				—	—	—	—	—	—	—	Undetermined

**Legend**

INTC: Interrupt controller

DTC: Data transfer controller

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF10	WDT	(TCSR)*1	OVF	WT/IT	TME	—	—	CKS2	CKS1	CKS0	H'18
H'FF11		TCNT*1									H'00
H'FF12			—	—	—	—	—	—	—	—	H'FF
H'FF13			—	—	—	—	—	—	—	—	H'FF
H'FF14	WSC	WCR	—	—	—	—	WMS1	WMS0	WC1	WC0	H'F3
H'FF15	RAM	RAMCR	RAME1	—	RAME2	—	—	RAM2	RAM1	RAM0	Undetermined
H'FF16	BSC	ARBT									H'FF
H'FF17		AR3T									H'0E*2
H'FF18			—	—	—	—	—	—	—	—	H'FF
H'FF19	SYSC	MDCR	—	—	—	—	—	MDS2	MDS1	MDS0	Undetermined
H'FF1A		SBYCR	SSBY	—	—	—	—	—	—	—	H'7F
H'FF1B		BRCR	—	—	—	—	—	—	—	BRLE	H'FE
H'FF1C		NMICR	—	—	—	—	—	—	—	NMIEG	H'FE
H'FF1D		IRQCR	—	—	—	—	IRQ3E	IRQ2E	IRQ1E	IRQ0E	H'F0
H'FF1E		(Write CR)									
H'FF1F		RSTCSR	WRST	RSTOE	—	—	—	—	—	—	H'3F

**Legend**

WDT: Watchdog timer  
WSC: Wait-state controller  
RAMCR: RAM controller  
BSC: Bus controller

(continued on next page)

- Notes: 1. These registers are write-protected by a password. See section 13.2.4, "Notes on Register Access" for details.  
2. Initial value in modes 5 and 6. In modes 1 to 4 and mode 7 the initial value is H'EE.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF20	IPU Channel 1	T1CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF21		T1CRL	—	CCLR2	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'80
H'FF22		T1SRAH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF23		T1SRAL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF24		T1OERA	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF25		TMDRA	MD6-7	MD4-7	MD3-5	MD2-6	SYNC3	SYNC2	SYNC1	SYNC0	H'00
H'FF26		T1CNTH*									H'00
H'FF27		T1CNTL*									H'00
H'FF28		T1GR1H*									H'FF
H'FF29		T1GR1L*									H'FF
H'FF2A		T1GR2H*									H'FF
H'FF2B		T1GR2L*									H'FF
H'FF2C		T1DR1H*									H'FF
H'FF2D		T1DR1L*									H'FF
H'FF2E		T1DR2H*									H'FF
H'FF2F		T1DR2L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF30	IPU Channel 1	TSTR	—	STR7	STR6	STR5	STR4	STR3	STR2	STR1	H'80
H'FF31		T1CRA	—	—	—	—	IEG41	IEG40	IEG31	IEG30	H'F0
H'FF32		T1SRBH	—	—	—	—	CMIE4	CMIE3	IMIE4	IMIE3	H'F0
H'FF33		T1SRBL	—	—	—	—	CMF4	CMF3	IMF4	IMF3	H'F0
H'FF34		T1OERB	DOE41	DOE40	DOE31	DOE30	GOE41	GOE40	GOE31	GOE30	H'00
H'FF35		TMDRB	—	—	MDF	PWM4	PWM3	PWM2	PWM1	PWM0	H'C0
H'FF36			—	—	—	—	—	—	—	—	H'FF
H'FF37			—	—	—	—	—	—	—	—	H'FF
H'FF38		T1GR3H*									H'FF
H'FF39		T1GR3L*									H'FF
H'FF3A		T1GR4H*									H'FF
H'FF3B		T1GR4L*									H'FF
H'FF3C		T1DR3H*									H'FF
H'FF3D		T1DR3L*									H'FF
H'FF3E		T1DR4H*									H'FF
H'FF3F		T1DR4L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.



(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF40	IPU Channel 2	T2CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF41		T2CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF42		T2SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF43		T2SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF44		T2OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF45			—	—	—	—	—	—	—	—	H'FF
H'FF46		T2CNTH*									H'00
H'FF47		T2CNTL*									H'00
H'FF48		T2GR1H*									H'FF
H'FF49		T2GR1L*									H'FF
H'FF4A		T2GR2H*									H'FF
H'FF4B		T2GR2L*									H'FF
H'FF4C		T2DR1H*									H'FF
H'FF4D		T2DR1L*									H'FF
H'FF4E		T2DR2H*									H'FF
H'FF4F		T2DR2L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF50	IPU Channel 3	T3CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF51		T3CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF52		T3SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF53		T3SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF54		T3OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF55				—	—	—	—	—	—	—	H'FF
H'FF56		T3CNTH*									H'00
H'FF57		T3CNTL*									H'00
H'FF58		T3GR1H*									H'FF
H'FF59		T3GR1L*									H'FF
H'FF5A		T3GR2H*									H'FF
H'FF5B		T3GR2L*									H'FF
H'FF5C		T3DR1H*									H'FF
H'FF5D		T3DR1L*									H'FF
H'FF5E		T3DR2H*									H'FF
H'FF5F		T3DR2L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF60	IPU Channel 4	T4CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF61		T4CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF62		T4SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF63		T4SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF64		T4OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF65			—	—	—	—	—	—	—	—	H'FF
H'FF66			T4CNTH*								H'00
H'FF67			T4CNTL*								H'00
H'FF68			T4GR1H*								H'FF
H'FF69			T4GR1L*								H'FF
H'FF6A			T4GR2H*								H'FF
H'FF6B			T4GR2L*								H'FF
H'FF6C			T4DR1H*								H'FF
H'FF6D			T4DR1L*								H'FF
H'FF6E			T4DR2H*								H'FF
H'FF6F			T4DR2L*								H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF70	IPU Channel 5	T5CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF71		T5CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF72		T5SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF73		T5SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF74		T5OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF75				—	—	—	—	—	—	—	H'FF
H'FF76		T5CNTH*									H'00
H'FF77		T5CNTL*									H'00
H'FF78		T5GR1H*									H'FF
H'FF79		T5GR1L*									H'FF
H'FF7A		T5GR2H*									H'FF
H'FF7B		T5GR2L*									H'FF
H'FF7C		T5DR1H*									H'FF
H'FF7D		T5DR1L*									H'FF
H'FF7E		T5DR2H*									H'FF
H'FF7F		T5DR2L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF80	IPU Channel 6	T6CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF81		T6CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF82		T6SRH	—	—	—	—	—	OVIE	IMIE2	IMIE1	H'F8
H'FF83		T6SRL	—	—	—	—	—	OVF	IMF2	IMF1	H'F8
H'FF84		T6OER	—	—	—	—	GOE21	GOE20	GOE11	GOE10	H'F0
H'FF85				—	—	—	—	—	—	—	H'FF
H'FF86		T6CNTH*									H'00
H'FF87		T6CNTL*									H'00
H'FF88		T6GR1H*									H'FF
H'FF89		T6GR1L*									H'FF
H'FF8A		T6GR2H*									H'FF
H'FF8B		T6GR2L*									H'FF
H'FF8C				—	—	—	—	—	—	—	H'FF
H'FF8D				—	—	—	—	—	—	—	H'FF
H'FF8E			—	—	—	—	—	—	—	H'FF	
H'FF8F			—	—	—	—	—	—	—	H'FF	

**Legend** (continued on next page)

IPU: 16-bit integrated timer pulse unit

Note: \* These registers support 16-bit access.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF90	IPU Channel 7	T7CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF91		T7CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF92		T7SRH	—	—	—	—	—	OVIE	IMIE2	IMIE1	H'F8
H'FF93		T7SRL	—	—	—	—	—	OVF	IMF2	IMF1	H'F8
H'FF94		T7OER	—	—	—	—	GOE21	GOE20	GOE11	GOE10	H'F0
H'FF95			—	—	—	—	—	—	—	—	H'FF
H'FF96		T7CNTH*									H'00
H'FF97		T7CNTL*									H'00
H'FF98		T7GR1H*									H'FF
H'FF99		T7GR1L*									H'FF
H'FF9A		T7GR2H*									H'FF
H'FF9B		T7GR2L*									H'FF
H'FF9C			—	—	—	—	—	—	—	—	H'FF
H'FF9D			—	—	—	—	—	—	—	—	H'FF
H'FF9E			—	—	—	—	—	—	—	—	H'FF
H'FF9F			—	—	—	—	—	—	—	—	H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

Note: \* These registers support 16-bit access.

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FFA0	MULT	MLTCR	CLR	S_ON	—	—	—	SIGN	MUL	MAC	H'38
H'FFA1		MLTBR	—	—	—	—	—	—	—	—	H'00
H'FFA2		MLTMAR	—	—	—	—	—	—	—	—	H'00
H'FFA3		MLTAR	—	—	—	—	—	—	—	—	H'00
H'FFA4		—	—	—	—	—	—	—	—	—	H'FF
H'FFA5		—	—	—	—	—	—	—	—	—	H'FF
H'FFA6		—	—	—	—	—	—	—	—	—	H'FF
H'FFA7		—	—	—	—	—	—	—	—	—	H'FF
H'FFA8		—	—	—	—	—	—	—	—	—	H'FF
H'FFA9		—	—	—	—	—	—	—	—	—	H'FF
H'FFAA	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAB	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAC	—	—	—	—	—	—	—	—	—	H'FF	
H'FEED	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAE	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAF	—	—	—	—	—	—	—	—	—	H'FF	

**Legend**

MULT: Multiplier

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FFB0	MULT	CA									H'00
H'FFB1		(CA)									H'00
H'FFB2		CB									H'00
H'FFB3		(CB)									H'00
H'FFB4		CC									H'00
H'FFB5		(CC)									H'00
H'FFB6		XH									Undetermined
H'FFB7		(XH)									Undetermined
H'FFB8		H									Undetermined
H'FFB9		(H)									Undetermined
H'FFBA		L									Undetermined
H'FFBB		(L)									Undetermined
H'FFBC		MR									H'00
H'FFBD		(MR)									H'00
H'FFBE		MMR									H'00
H'FFBF		(MMR)									H'00

**Legend**

MULT: Multiplier



## C.2 H8/539F S Mask model

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FE80	Port 1	P1DDR	P1 <sub>7</sub> DDR	P1 <sub>6</sub> DDR	P1 <sub>5</sub> DDR	P1 <sub>4</sub> DDR	P1 <sub>3</sub> DDR	P1 <sub>2</sub> DDR	P1 <sub>1</sub> DDR	P1 <sub>0</sub> DDR	H'00
H'FE81	Port 2	P2DDR	P2 <sub>7</sub> DDR	P2 <sub>6</sub> DDR	P2 <sub>5</sub> DDR	P2 <sub>4</sub> DDR	P2 <sub>3</sub> DDR	P2 <sub>2</sub> DDR	P2 <sub>1</sub> DDR	P2 <sub>0</sub> DDR	H'00
H'FE82	Port 1	P1DR	P1 <sub>7</sub>	P1 <sub>6</sub>	P1 <sub>5</sub>	P1 <sub>4</sub>	P1 <sub>3</sub>	P1 <sub>2</sub>	P1 <sub>1</sub>	P1 <sub>0</sub>	H'00
H'FE83	Port 2	P2DR	P2 <sub>7</sub>	P2 <sub>6</sub>	P2 <sub>5</sub>	P2 <sub>4</sub>	P2 <sub>3</sub>	P2 <sub>2</sub>	P2 <sub>1</sub>	P2 <sub>0</sub>	H'00
H'FE84	Port 3	P3DDR	—	—	P3 <sub>5</sub> DDR	P3 <sub>4</sub> DDR	P3 <sub>3</sub> DDR	P3 <sub>2</sub> DDR	P3 <sub>1</sub> DDR	P3 <sub>0</sub> DDR	H'C0
H'FE85	Port 4	P4DDR	P4 <sub>7</sub> DDR	P4 <sub>6</sub> DDR	P4 <sub>5</sub> DDR	P4 <sub>4</sub> DDR	P4 <sub>3</sub> DDR	P4 <sub>2</sub> DDR	P4 <sub>1</sub> DDR	P4 <sub>0</sub> DDR	H'00
H'FE86	Port 3	P3DR	—	—	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>	H'C0
H'FE87	Port 4	P4DR	P4 <sub>7</sub>	P4 <sub>6</sub>	P4 <sub>5</sub>	P4 <sub>4</sub>	P4 <sub>3</sub>	P4 <sub>2</sub>	P4 <sub>1</sub>	P4 <sub>0</sub>	H'00
H'FE88	Port 5	P5DDR	P5 <sub>7</sub> DDR	P5 <sub>6</sub> DDR	P5 <sub>5</sub> DDR	P5 <sub>4</sub> DDR	P5 <sub>3</sub> DDR	P5 <sub>2</sub> DDR	P5 <sub>1</sub> DDR	P5 <sub>0</sub> DDR	H'00
H'FE89	Port 6	P6DDR	—	—	—	P6 <sub>4</sub> DDR	P6 <sub>3</sub> DDR	P6 <sub>2</sub> DDR	P6 <sub>1</sub> DDR	P6 <sub>0</sub> DDR	H'E0
H'FE8A	Port 5	P5DR	P5 <sub>7</sub>	P5 <sub>6</sub>	P5 <sub>5</sub>	P5 <sub>4</sub>	P5 <sub>3</sub>	P5 <sub>2</sub>	P5 <sub>1</sub>	P5 <sub>0</sub>	H'00
H'FE8B	Port 6	P6DR	—	—	—	P6 <sub>4</sub>	P6 <sub>3</sub>	P6 <sub>2</sub>	P6 <sub>1</sub>	P6 <sub>0</sub>	H'E0
H'FE8C	Port 7	P7DDR	P7 <sub>7</sub> DDR	P7 <sub>6</sub> DDR	P7 <sub>5</sub> DDR	P7 <sub>4</sub> DDR	P7 <sub>3</sub> DDR	P7 <sub>2</sub> DDR	P7 <sub>1</sub> DDR	P7 <sub>0</sub> DDR	H'00
H'FE8D			—	—	—	—	—	—	—	—	H'FF
H'FE8E	Port 7	P7DR	P7 <sub>7</sub>	P7 <sub>6</sub>	P7 <sub>5</sub>	P7 <sub>4</sub>	P7 <sub>3</sub>	P7 <sub>2</sub>	P7 <sub>1</sub>	P7 <sub>0</sub>	H'00
H'FE8F	Port 8	P8DR	—	—	—	—	P8 <sub>3</sub>	P8 <sub>2</sub>	P8 <sub>1</sub>	P8 <sub>0</sub>	Undetermined

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FE90			—	—	—	—	—	—	—	—	H'FF
H'FE91	Port A	PADDR	—	PA <sub>6</sub> DDR	PA <sub>5</sub> DDR	PA <sub>4</sub> DDR	PA <sub>3</sub> DDR	PA <sub>2</sub> DDR	PA <sub>1</sub> DDR	PA <sub>0</sub> DDR	H'80
H'FE92	Port 9	P9DR	P9 <sub>7</sub>	P9 <sub>6</sub>	P9 <sub>5</sub>	P9 <sub>4</sub>	P9 <sub>3</sub>	P9 <sub>2</sub>	P9 <sub>1</sub>	P9 <sub>0</sub>	Undetermined
H'FE93	Port A	PADR	—	PA <sub>6</sub>	PA <sub>5</sub>	PA <sub>4</sub>	PA <sub>3</sub>	PA <sub>2</sub>	PA <sub>1</sub>	PA <sub>0</sub>	H'80
H'FE94	Port B	PBDDR	PB <sub>7</sub> DDR	PB <sub>6</sub> DDR	PB <sub>5</sub> DDR	PB <sub>4</sub> DDR	PB <sub>3</sub> DDR	PB <sub>2</sub> DDR	PB <sub>1</sub> DDR	PB <sub>0</sub> DDR	H'00
H'FE95	Port C	PCDDR	PC <sub>7</sub> DDR	PC <sub>6</sub> DDR	PC <sub>5</sub> DDR	PC <sub>4</sub> DDR	PC <sub>3</sub> DDR	PC <sub>2</sub> DDR	PC <sub>1</sub> DDR	PC <sub>0</sub> DDR	H'00
H'FE96	Port B	PBDR	PB <sub>7</sub>	PB <sub>6</sub>	PB <sub>5</sub>	PB <sub>4</sub>	PB <sub>3</sub>	PB <sub>2</sub>	PB <sub>1</sub>	PB <sub>0</sub>	H'00
H'FE97	Port C	PCDR	PC <sub>7</sub>	PC <sub>6</sub>	PC <sub>5</sub>	PC <sub>4</sub>	PC <sub>3</sub>	PC <sub>2</sub>	PC <sub>1</sub>	PC <sub>0</sub>	H'00
H'FE98	Port B	PBPCR	PB <sub>7</sub> PON	PB <sub>6</sub> PON	PB <sub>5</sub> PON	PB <sub>4</sub> PON	PB <sub>3</sub> PON	PB <sub>2</sub> PON	PB <sub>1</sub> PON	PB <sub>0</sub> PON	H'00
H'FE99	Port C	PCPCR	PC <sub>7</sub> PON	PC <sub>6</sub> PON	PC <sub>5</sub> PON	PC <sub>4</sub> PON	PC <sub>3</sub> PON	PC <sub>2</sub> PON	PC <sub>1</sub> PON	PC <sub>0</sub> PON	H'00
H'FE9A	ø <sub>CR</sub>	ø <sub>CR</sub>	ø <sub>OE</sub>	—	—	—	—	—	—	—	Undefined*
H'FE9B			—	—	—	—	—	—	—	—	H'FF
H'FE9C			—	—	—	—	—	—	—	—	H'FF
H'FE9D			—	—	—	—	—	—	—	—	H'FF
H'FE9E			—	—	—	—	—	—	—	—	H'FF
H'FE9F			—	—	—	—	—	—	—	—	H'FF

(continued on next page)

Note: \* Initialized to H'FF in standby mode.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEA0	A/D	ADDR0H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA1		ADDR0L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA2		ADDR1H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA3		ADDR1L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA4		ADDR2H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA5		ADDR2L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA6		ADDR3H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA7		ADDR3L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEA8		ADDR4H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEA9		ADDR4L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEAA		ADDR5H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEAB		ADDR5L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEAC		ADDR6H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEAD		ADDR6L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEAE		ADDR7H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEAF		ADDR7L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00

**Legend**

A/D: A/D converter

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEB0	A/D	ADDR8H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB1		ADDR8L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB2		ADDR9H	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB3		ADDR9L	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB4		ADDRAH	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB5		ADDRAL	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB6		ADDRBH	AD <sub>9</sub>	AD <sub>8</sub>	AD <sub>7</sub>	AD <sub>6</sub>	AD <sub>5</sub>	AD <sub>4</sub>	AD <sub>3</sub>	AD <sub>2</sub>	H'00
H'FEB7		ADDRBL	AD <sub>1</sub>	AD <sub>0</sub>	—	—	—	—	—	—	H'00
H'FEB8		ADCSR	ADF	ADIE	ADM1	ADM0	CH3	CH2	CH1	CH0	H'00
H'FEB9		ADCR	TRGE	CKS	ADST	—	—	—	—	—	H'1F
H'FEBA			—	—	—	—	—	—	—	—	H'FF
H'FEBB			—	—	—	—	—	—	—	—	H'FF
H'FEBC			—	—	—	—	—	—	—	—	H'FF
H'FEBD			—	—	—	—	—	—	—	—	H'FF
H'FEBE			—	—	—	—	—	—	—	—	H'FF
H'FEBF			—	—	—	—	—	—	—	—	H'FF

**Legend**

A/D: A/D converter

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEC0	SCI3	SMR	C/Ā	CHR	PE	O/Ē	STOP	MP	CKS1	CKS0	H'00
H'FEC1		BRR									H'FF
H'FEC2		SCR	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	H'00
H'FEC3		TDR									H'FF
H'FEC4		SSR	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	H'84F
H'FEC5		RDR									H'00
H'FEC6				—	—	—	—	—	—	—	—
H'FEC7			—	—	—	—	—	—	—	—	Undetermined
H'FEC8	SCI1	SMR	C/Ā	CHR	PE	O/Ē	STOP	MP	CKS1	CKS0	H'00
H'FEC9		BRR									H'FF
H'FECA		SCR	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	H'00
H'FECEB		TDR									H'FF
H'FECC		SSR	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	H'84
H'FECD		RDR									H'00
H'FECE				—	—	—	—	—	—	—	—
H'FECE			—	—	—	—	—	—	—	—	Undetermined

**Legend**

SCI1: Serial communication interface 1

SCI3: Serial communication interface 3

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FED0	SCI2	SMR	C/A	CHR	PE	O/E	STOP	MP	CKS1	CKS0	H'00
H'FED1		BRR									H'FF
H'FED2		SCR	TIE	RIE	TE	RE	MPIE	TEIE	CKE1	CKE0	H'00
H'FED3		TDR									H'FF
H'FED4		SSR	TDRE	RDRF	ORER	FER	PER	TEND	MPB	MPBT	H'84
H'FED5		RDR									H'00
H'FED6			—	—	—	—	—	—	—	—	H'FF
H'FED7			—	—	—	—	—	—	—	—	Undetermined
H'FED8			—	—	—	—	—	—	—	—	H'FF
H'FED9			—	—	—	—	—	—	—	—	H'FF
H'FEDA	Port A	PACR	—	TXD3E	RXD3E	—	SCK3E	PW3E	PW2E	PW1E	H'90
H'FEDB	Port 6/7	P67CR	PW2E	PW1E	—	—	—	—	—	PW3E	H'3E
H'FEDC	A/D	ADTRGR	EXTRG	—	—	—	—	—	—	—	H'FF
H'FEDD			—	—	—	—	—	—	—	—	H'FF
H'FEDE	INTC	IRQFR	—	—	—	—	IRQ3F	IRQ2F	IRQ1F	—	H'F1
H'FEDF	BSC	BCR	BCRE	0P3T	—	P9AE	EXIOP	PCRE	PBCE	P12E	H'3F*

**Legend**

SCI2: Serial communication interface 2

INTC: Interrupt controller

BSC: Bus controller

A/D: A/D converter

(continued on next page)

Note: \* Initial value in modes 5 and 6. In modes 1 to 4 and mode 7 the initial value is H'BF.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEE0	Flash memory	FLMCR	FWE	SWE	ESU	PSU	EV	PV	E	P	H'00*
H'FEE1		—	—	—	—	—	—	—	—	—	H'FF
H'FEE2		EBR1	EB7	EB6	EB5	EB4	EB3	EB2	EB1	EB0	H'00
H'FEE3		—	—	—	—	—	—	—	—	—	H'FF
H'FEE4		—	—	—	—	—	—	—	—	—	H'FF
H'FEE5		—	—	—	—	—	—	—	—	—	H'FF
H'FEE6		—	—	—	—	—	—	—	—	—	H'FF
H'FEE7		—	—	—	—	—	—	—	—	—	H'FF
H'FEE8		—	—	—	—	—	—	—	—	—	H'FF
H'FEE9		—	—	—	—	—	—	—	—	—	H'FF
H'FEEA		—	—	—	—	—	—	—	—	—	H'FF
H'FEEB		—	—	—	—	—	—	—	—	—	H'FF
H'FEEC		Flash memory	FLMER	OVLPE	—	—	—	A11E	A10E	—	—
H'FEED	FLMSR		FLER	—	—	—	—	—	—	—	H'7F
H'FEEE	—		—	—	—	—	—	—	—	—	H'FF
H'FE EF	—		—	—	—	—	—	—	—	—	H'FF

(continued on next page)

Note: \* When a high level is being applied to the FWE to the FWE pin, the initial value is H'80.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEF0	PWM1	TCR	OE	OS	—	—	—	CKS2	CKS1	CKS0	H'38
H'FEF1		DTR									H'FF
H'FEF2		TCNT									H'00
H'FEF3			—	—	—	—	—	—	—	—	H'FF
H'FEF4	PWM2	TCR	OE	OS	—	—	—	CKS2	CKS1	CKS0	H'38
H'FEF5		DTR									H'FF
H'FEF6		TCNT									H'00
H'FEF7			—	—	—	—	—	—	—	—	H'FF
H'FEF8	PWM3	TCR	OE	OS	—	—	—	CKS2	CKS1	CKS0	H'38
H'FEF9		DTR									H'FF
H'FEFA		TCNT									H'00
H'FEFB			—	—	—	—	—	—	—	—	H'FF
H'FEFC			—	—	—	—	—	—	—	—	H'FF
H'FEFD			—	—	—	—	—	—	—	—	H'FF
H'FEFE			—	—	—	—	—	—	—	—	H'FF
H'FEFF			—	—	—	—	—	—	—	—	H'FF

**Legend**

(continued on next page)

PWM1: Pulse width modulation timer 1

PWM2: Pulse width modulation timer 2

PWM3: Pulse width modulation timer 3



(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF00	INTC	IPRA	0				0				H'00
H'FF01		IPRB	0				0				H'00
H'FF02		IPRC	0				0				H'00
H'FF03		IPRD	0				0				H'00
H'FF04		IPRE	0				0				H'00
H'FF05		IPRF	0				0				H'00
H'FF06	DTC		—	—	—	—	—	—	—	—	Undetermined
H'FF07			—	—	—	—	—	—	—	—	Undetermined
H'FF08		DTEA	0	ADI	(IRQ0)	IRQ0	0	IRQ3	IRQ2	IRQ1	H'00
H'FF09		DTEB	0	T1CMI1,2	T1IMI2	T1IMI1	0	T1CMI3,4	T1IMI4	T1IMI3	H'00
H'FF0A		DTEC	0	T2CMI1,2	T2IMI2	T2IMI1	0	T3CMI1,2	T3IMI2	T3IMI1	H'00
H'FF0B		DTED	0	T4CMI1,2	T4IMI2	T4IMI1	0	T5CMI1,2	T5IMI2	T5IMI1	H'00
H'FF0C		DTEE	0	0	T6IMI2	T6IMI1	0	0	T7IMI2	T7IMI1	H'00
H'FF0D		DTEF	0	T11	R11	0	0	T12	R12	0	H'00
H'FF0E			—	—	—	—	—	—	—	Undetermined	
H'FF0F			—	—	—	—	—	—	—	Undetermined	

**Legend**

INTC: Interrupt controller

DTC: Data transfer controller

(continued on next page)

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF10	WDT	(TCSR)*1	OVF	WT/IT	TME	—	—	CKS2	CKS1	CKS0	H'18
H'FF11		TCNT*1	—	—	—	—	—	—	—	—	H'00
H'FF12		—	—	—	—	—	—	—	—	—	H'FF
H'FF13	—	—	—	—	—	—	—	—	—	—	H'FF
H'FF14	WSC	WCR	—	—	—	—	WMS1	WMS0	WC1	WC0	H'F3
H'FF15	RAM	RAMCR	RAME1	—	RAME2	—	—	RAM2	RAM1	—	Undetermined
H'FF16	BSC	ARBT	—	—	—	—	—	—	—	—	H'FF
H'FF17		AR3T	—	—	—	—	—	—	—	—	H'0E*2
H'FF18		—	—	—	—	—	—	—	—	—	H'FF
H'FF19	SYSC	MDCR	—	—	—	—	—	MDS2	MDS1	MDS0	Undetermined
H'FF1A	—	SBYCR	SSBY	—	—	—	—	—	—	—	H'7F
H'FF1B	—	BRCR	—	—	—	—	—	—	—	BRLE	H'FE
H'FF1C	—	NMICR	—	—	—	—	—	—	—	NMIEG	H'FE
H'FF1D	—	IRQCR	—	—	—	—	IRQ3E	IRQ2E	IRQ1E	IRQ0E	H'F0
H'FF1E	—	(Write CR)	—	—	—	—	—	—	—	—	—
H'FF1F	—	RSTCSR	WRST	RSTOE	—	—	—	—	—	—	H'3F

**Legend**

(continued on next page)

WDT: Watchdog timer  
WSC: Wait-state controller  
RAMCR: RAM controller  
BSC: Bus controller

- Notes: 1. These registers are write-protected by a password. See section 13.2.4, "Notes on Register Access" for details.  
2. Initial value in modes 5 and 6. In modes 1 to 4 and mode 7 the initial value is H'EE.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF20	IPU Channel 1	T1CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF21		T1CRL	—	CCLR2	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'80
H'FF22		T1SRAH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF23		T1SRAL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF24		T1OERA	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF25		TMDRA	MD6-7	MD4-7	MD3-5	MD2-6	SYNC3	SYNC2	SYNC1	SYNC0	H'00
H'FF26		T1CNTH*									H'00
H'FF27		T1CNTL*									H'00
H'FF28		T1GR1H*									H'FF
H'FF29		T1GR1L*									H'FF
H'FF2A		T1GR2H*									H'FF
H'FF2B		T1GR2L*									H'FF
H'FF2C		T1DR1H*									H'FF
H'FF2D		T1DR1L*									H'FF
H'FF2E		T1DR2H*									H'FF
H'FF2F		T1DR2L*									H'FF

**Legend** (continued on next page)

IPU: 16-bit integrated timer pulse unit

Note: \* These registers support 16-bit access.

(continued from previous page)

Address	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF30	IPU Channel 1	TSTR	—	STR7	STR6	STR5	STR4	STR3	STR2	STR1	H'80
H'FF31		T1CRA	—	—	—	—	IEG41	IEG40	IEG31	IEG30	H'F0
H'FF32		T1SRBH	—	—	—	—	CMIE4	CMIE3	IMIE4	IMIE3	H'F0
H'FF33		T1SRBL	—	—	—	—	CMF4	CMF3	IMF4	IMF3	H'F0
H'FF34		T1OERB	DOE41	DOE40	DOE31	DOE30	GOE41	GOE40	GOE31	GOE30	H'00
H'FF35		TMDRB	—	—	MDF	PWM4	PWM3	PWM2	PWM1	PWM0	H'C0
H'FF36			—	—	—	—	—	—	—	—	H'FF
H'FF37			—	—	—	—	—	—	—	—	H'FF
H'FF38		T1GR3H*									H'FF
H'FF39		T1GR3L*									H'FF
H'FF3A		T1GR4H*									H'FF
H'FF3B		T1GR4L*									H'FF
H'FF3C		T1DR3H*									H'FF
H'FF3D		T1DR3L*									H'FF
H'FF3E		T1DR4H*									H'FF
H'FF3F		T1DR4L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

Note: \* These registers support 16-bit access.

(continued on next page)

(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF40	IPU Channel 2	T2CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF41		T2CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF42		T2SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF43		T2SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF44		T2OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF45				—	—	—	—	—	—	—	H'FF
H'FF46			T2CNTH*								H'00
H'FF47			T2CNTL*								H'00
H'FF48			T2GR1H*								H'FF
H'FF49			T2GR1L*								H'FF
H'FF4A			T2GR2H*								H'FF
H'FF4B			T2GR2L*								H'FF
H'FF4C			T2DR1H*								H'FF
H'FF4D			T2DR1L*								H'FF
H'FF4E			T2DR2H*								H'FF
H'FF4F			T2DR2L*								H'FF

**Legend** (continued on next page)

IPU: 16-bit integrated timer pulse unit

Note: \* These registers support 16-bit access.

(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF50	IPU Channel 3	T3CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF51		T3CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF52		T3SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF53		T3SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF54		T3OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF55				—	—	—	—	—	—	—	H'FF
H'FF56		T3CNTH*									H'00
H'FF57		T3CNTL*									H'00
H'FF58		T3GR1H*									H'FF
H'FF59		T3GR1L*									H'FF
H'FF5A		T3GR2H*									H'FF
H'FF5B		T3GR2L*									H'FF
H'FF5C		T3DR1H*									H'FF
H'FF5D		T3DR1L*									H'FF
H'FF5E		T3DR2H*									H'FF
H'FF5F		T3DR2L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.

(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF60	IPU Channel 4	T4CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF61		T4CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF62		T4SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF63		T4SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF64		T4OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF65				—	—	—	—	—	—	—	H'FF
H'FF66			T4CNTH*								H'00
H'FF67			T4CNTL*								H'00
H'FF68			T4GR1H*								H'FF
H'FF69			T4GR1L*								H'FF
H'FF6A			T4GR2H*								H'FF
H'FF6B			T4GR2L*								H'FF
H'FF6C			T4DR1H*								H'FF
H'FF6D			T4DR1L*								H'FF
H'FF6E			T4DR2H*								H'FF
H'FF6F			T4DR2L*								H'FF

**Legend** (continued on next page)

IPU: 16-bit integrated timer pulse unit

Note: \* These registers support 16-bit access.

(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF70	IPU Channel 5	T5CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF71		T5CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF72		T5SRH	—	—	—	OVIE	CMIE2	CMIE1	IMIE2	IMIE1	H'E0
H'FF73		T5SRL	—	—	—	OVF	CMF2	CMF1	IMF2	IMF1	H'E0
H'FF74		T5OER	DOE21	DOE20	DOE11	DOE10	GOE21	GOE20	GOE11	GOE10	H'00
H'FF75				—	—	—	—	—	—	—	H'FF
H'FF76		T5CNTH*									H'00
H'FF77		T5CNTL*									H'00
H'FF78		T5GR1H*									H'FF
H'FF79		T5GR1L*									H'FF
H'FF7A		T5GR2H*									H'FF
H'FF7B		T5GR2L*									H'FF
H'FF7C		T5DR1H*									H'FF
H'FF7D		T5DR1L*									H'FF
H'FF7E		T5DR2H*									H'FF
H'FF7F		T5DR2L*									H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.



(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF80	IPU Channel 6	T6CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF81		T6CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF82		T6SRH	—	—	—	—	—	OVIE	IMIE2	IMIE1	H'F8
H'FF83		T6SRL	—	—	—	—	—	OVF	IMF2	IMF1	H'F8
H'FF84		T6OER	—	—	—	—	GOE21	GOE20	GOE11	GOE10	H'F0
H'FF85				—	—	—	—	—	—	—	H'FF
H'FF86		T6CNTH*									H'00
H'FF87		T6CNTL*									H'00
H'FF88		T6GR1H*									H'FF
H'FF89		T6GR1L*									H'FF
H'FF8A		T6GR2H*									H'FF
H'FF8B		T6GR2L*									H'FF
H'FF8C				—	—	—	—	—	—	—	H'FF
H'FF8D				—	—	—	—	—	—	—	H'FF
H'FF8E			—	—	—	—	—	—	—	H'FF	
H'FF8F			—	—	—	—	—	—	—	H'FF	

**Legend** (continued on next page)

IPU: 16-bit integrated timer pulse unit

Note: \* These registers support 16-bit access.

(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF90	IPU Channel 7	T7CRH	—	—	CKEG1	CKEG0	TPSC3	TPSC2	TPSC1	TPSC0	H'00
H'FF91		T7CRL	—	—	CCLR1	CCLR0	IEG21	IEG20	IEG11	IEG10	H'00
H'FF92		T7SRH	—	—	—	—	—	OVIE	IMIE2	IMIE1	H'F8
H'FF93		T7SRL	—	—	—	—	—	OVF	IMF2	IMF1	H'F8
H'FF94		T7OER	—	—	—	—	GOE21	GOE20	GOE11	GOE10	H'F0
H'FF95			—	—	—	—	—	—	—	—	H'FF
H'FF96		T7CNTH*									H'00
H'FF97		T7CNTL*									H'00
H'FF98		T7GR1H*									H'FF
H'FF99		T7GR1L*									H'FF
H'FF9A		T7GR2H*									H'FF
H'FF9B		T7GR2L*									H'FF
H'FF9C			—	—	—	—	—	—	—	—	H'FF
H'FF9D			—	—	—	—	—	—	—	—	H'FF
H'FF9E			—	—	—	—	—	—	—	—	H'FF
H'FF9F			—	—	—	—	—	—	—	—	H'FF

**Legend**

IPU: 16-bit integrated timer pulse unit

(continued on next page)

Note: \* These registers support 16-bit access.

(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FFA0	MULT	MLTCR	CLR	S_ON	—	—	—	SIGN	MUL	MAC	H'38
H'FFA1		MLTBR	—	—	—	—	—	—	—	—	H'00
H'FFA2		MLTMAR	—	—	—	—	—	—	—	—	H'00
H'FFA3		MLTAR	—	—	—	—	—	—	—	—	H'00
H'FFA4		—	—	—	—	—	—	—	—	—	H'FF
H'FFA5		—	—	—	—	—	—	—	—	—	H'FF
H'FFA6		—	—	—	—	—	—	—	—	—	H'FF
H'FFA7		—	—	—	—	—	—	—	—	—	H'FF
H'FFA8		—	—	—	—	—	—	—	—	—	H'FF
H'FFA9		—	—	—	—	—	—	—	—	—	H'FF
H'FFAA	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAB	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAC	—	—	—	—	—	—	—	—	—	H'FF	
H'FEED	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAE	—	—	—	—	—	—	—	—	—	H'FF	
H'FFAF	—	—	—	—	—	—	—	—	—	H'FF	

**Legend**

MULT: Multiplier

(continued on next page)

(continued from previous page)

Address (low)	Module Name	Register Name	Bit Names								Initial Value
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FFB0	MULT	CA									H'00
H'FFB1		(CA)									H'00
H'FFB2		CB									H'00
H'FFB3		(CB)									H'00
H'FFB4		CC									H'00
H'FFB5		(CC)									H'00
H'FFB6		XH									Undeter- mined
H'FFB7		(XH)									Undeter- mined
H'FFB8		H									Undeter- mined
H'FFB9		(H)									Undeter- mined
H'FFBA		L									Undeter- mined
H'FFBB		(L)									Undeter- mined
H'FFBC		MR									H'00
H'FFBD		(MR)									H'00
H'FFBE		MMR									H'00
H'FFBF		(MMR)									H'00

**Legend**

MULT: Multiplier

# Appendix D Pin Function Selection

## D.1 Port 3 Function Selection

**Table D-1 IPU and P3DDR Settings and Selected Functions of P3<sub>0</sub>/T1OC<sub>1</sub>**

DOE11, 10 (T1OERA)	00		01, 10, 11
P3 <sub>0</sub> DDR	0	1	Don't care
Selected function	P3 <sub>0</sub> input port	P3 <sub>0</sub> output port	T1OC <sub>1</sub> output

**Table D-2 IPU and P3DDR Settings and Selected Functions of P3<sub>1</sub>/T1OC<sub>2</sub>**

DOE21, 20 (T1OERA)	00		01, 10, 11
P3 <sub>1</sub> DDR	0	1	Don't care
Selected function	P3 <sub>1</sub> input port	P3 <sub>1</sub> output port	T1OC <sub>2</sub> output

**Table D-3 IPU and P3DDR Settings and Selected Functions of P3<sub>2</sub>/T1OC<sub>3</sub>**

DOE31, 30 (T1OERB)	00		01, 10, 11
P3 <sub>2</sub> DDR	0	1	Don't care
Selected function	P3 <sub>2</sub> input port	P3 <sub>2</sub> output port	T1OC <sub>3</sub> output

**Table D-4 IPU and P3DDR Settings and Selected Functions of P3<sub>3</sub>/T1OC<sub>4</sub>**

DOE41, 40 (T1OERB)	00		01, 10, 11
P3 <sub>3</sub> DDR	0	1	Don't care
Selected function	P3 <sub>3</sub> input port	P3 <sub>3</sub> output port	T1OC <sub>4</sub> output

**Table D-5 IPU and P3DDR Settings and Selected Functions of P3<sub>4</sub>/T2OC<sub>1</sub>**

DOE11, 10 (T2OER)	00		01, 10, 11
P3 <sub>4</sub> DDR	0	1	Don't care
Selected function	P3 <sub>4</sub> input port	P3 <sub>4</sub> output port	T2OC <sub>1</sub> output

**Table D-6 IPU and P3DDR Settings and Selected Functions of P3<sub>5</sub>/T2OC<sub>2</sub>**

DOE21, 20 (T2OER)	00		01, 10, 11
P3 <sub>5</sub> DDR	0	1	Don't care
Selected function	P3 <sub>5</sub> input port	P3 <sub>5</sub> output port	T2OC <sub>2</sub> output

## D.2 Port 4 Function Selection

**Table D-7 IPU and P4DDR Settings and Selected Functions of P4<sub>0</sub>/T4IOC<sub>1</sub>**

GOE11, 10 (T4OER)	00		Don't care		01, 10, 11	
IEG11, 10 (T4CRL)	00		01, 10, 11		00	
P4 <sub>0</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>0</sub> input port	P4 <sub>0</sub> output port	P4 <sub>0</sub> input port	P4 <sub>0</sub> output port	T4IOC <sub>1</sub> output	
			T4IOC <sub>1</sub> input			

**Table D-8 IPU and P4DDR Settings and Selected Functions of P4<sub>1</sub>/T4IOC<sub>2</sub>**

GOE21, 20 (T4OER)	00		Don't care		01, 10, 11	
IEG21, 20 (T4CRL)	00		01, 10, 11		00	
P4 <sub>1</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>1</sub> input port	P4 <sub>1</sub> output port	P4 <sub>1</sub> input port	P4 <sub>1</sub> output port	T4IOC <sub>2</sub> output	
			T4IOC <sub>2</sub> input			

**Table D-9 IPU and P4DDR Settings and Selected Functions of P4<sub>2</sub>/T5IOC<sub>1</sub>**

GOE11, 10 (T5OER)	00		Don't care		01, 10, 11	
IEG11, 10 (T5CRL)	00		01, 10, 11		00	
P4 <sub>2</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>2</sub> input port	P4 <sub>2</sub> output port	P4 <sub>2</sub> input port	P4 <sub>2</sub> output port	T5IOC <sub>1</sub> output	
			T5IOC <sub>1</sub> input			

**Table D-10 IPU and P4DDR Settings and Selected Functions of P4<sub>3</sub>/T5IOC<sub>2</sub>**

GOE21, 20 (T5OER)	00		Don't care		01, 10, 11	
IEG21, 20 (T5CRL)	00		01, 10, 11		00	
P4 <sub>3</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>3</sub> input port	P4 <sub>3</sub> output port	P4 <sub>3</sub> input port	P4 <sub>3</sub> output port	T5IOC <sub>2</sub> output	
			T5IOC <sub>2</sub> input			

**Table D-11 IPU and P4DDR Settings and Selected Functions of P4<sub>4</sub>/T6IOC<sub>1</sub>**

GOE11, 10 (T6OER)	00		Don't care		01, 10, 11	
IEG11, 10 (T6CRL)	00		01, 10, 11		00	
P4 <sub>4</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>4</sub> input port	P4 <sub>4</sub> output port	P4 <sub>4</sub> input port	P4 <sub>4</sub> output port	T6IOC <sub>1</sub> output	
			T6IOC <sub>1</sub> input			

**Table D-12 IPU and P4DDR Settings and Selected Functions of P4<sub>5</sub>/T6IOC<sub>2</sub>**

GOE21, 20 (T6OER)	00		Don't care		01, 10, 11	
IEG21, 20 (T6CRL)	00		01, 10, 11		00	
P4 <sub>5</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>5</sub> input port	P4 <sub>5</sub> output port	P4 <sub>5</sub> input port	P4 <sub>5</sub> output port	T6IOC <sub>2</sub> output	
			T6IOC <sub>2</sub> input			

**Table D-13 IPU and P4DDR Settings and Selected Functions of P4<sub>6</sub>/T7IOC<sub>1</sub>**

GOE11, 10 (T7OER)	00		Don't care		01, 10, 11	
IEG11, 10 (T7CRL)	00		01, 10, 11		00	
P4 <sub>6</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>6</sub> input port	P4 <sub>6</sub> output port	P4 <sub>6</sub> input port	P4 <sub>6</sub> output port	T7IOC <sub>1</sub> output	
			T7IOC <sub>1</sub> input			

**Table D-14 IPU and P4DDR Settings and Selected Functions of P4<sub>7</sub>/T7IOC<sub>2</sub>**

GOE21, 20 (T7OER)	00		Don't care		01, 10, 11	
IEG21, 20 (T7CRL)	00		01, 10, 11		00	
P4 <sub>7</sub> DDR	0	1	0	1	0	1
Selected function	P4 <sub>7</sub> input port	P4 <sub>7</sub> output port	P4 <sub>7</sub> input port	P4 <sub>7</sub> output port	T7IOC <sub>2</sub> output	
			T7IOC <sub>2</sub> input			

### D.3 Port 5 Function Selection

**Table D-15 IPU and P5DDR Settings and Selected Functions of P5<sub>0</sub>/T1IOC<sub>1</sub>**

GOE11, 10 (T1OERA)	00		Don't care		01, 10, 11	
IEG11, 10 (T1CRAL)	00		01, 10, 11		00	
P5 <sub>0</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>0</sub> input port	P5 <sub>0</sub> output port	P5 <sub>0</sub> input port	P5 <sub>0</sub> output port	T1IOC <sub>1</sub> output	
			T1IOC <sub>1</sub> input			

**Table D-16 IPU and P5DDR Settings and Selected Functions of P5<sub>1</sub>/T1IOC<sub>2</sub>**

GOE21, 20 (T1OERA)	00		Don't care		01, 10, 11	
IEG21, 20 (T1CRAL)	00		01, 10, 11		00	
P5 <sub>1</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>1</sub> input port	P5 <sub>1</sub> output port	P5 <sub>1</sub> input port	P5 <sub>1</sub> output port	T1IOC <sub>2</sub> output	
			T1IOC <sub>2</sub> input			

**Table D-17 IPU and P5DDR Settings and Selected Functions of P5<sub>2</sub>/T1IOC<sub>3</sub>**

GOE31, 30 (T1OERB)	00		Don't care		01, 10, 11	
IEG31, 30 (T1CRB)	00		01, 10, 11		00	
P5 <sub>2</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>2</sub> input port	P5 <sub>2</sub> output port	P5 <sub>2</sub> input port	P5 <sub>2</sub> output port	T1IOC <sub>3</sub> output	
			T1IOC <sub>3</sub> input			

**Table D-18 IPU and P5DDR Settings and Selected Functions of P5<sub>3</sub>/T1IOC<sub>4</sub>**

GOE41, 40 (T1OERA)	00		Don't care		01, 10, 11	
IEG41, 40 (T1CRB)	00		01, 10, 11		00	
P5 <sub>3</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>3</sub> input port	P5 <sub>3</sub> output port	P5 <sub>3</sub> input port	P5 <sub>3</sub> output port	T1IOC <sub>4</sub> output	
			T1IOC <sub>4</sub> input			



**Table D-19 IPU and P5DDR Settings and Selected Functions of P5<sub>4</sub>/T2IOC<sub>1</sub>**

GOE11, 10 (T2OER)	00		Don't care		01, 10, 11	
IEG11, 10 (T2CRL)	00		01, 10, 11		00	
P5 <sub>4</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>4</sub> input port	P5 <sub>4</sub> output port	P5 <sub>4</sub> input port	P5 <sub>4</sub> output port	T2IOC <sub>1</sub> output	
			T2IOC <sub>1</sub> input			

**Table D-20 IPU and P5DDR Settings and Selected Functions of P5<sub>5</sub>/T2IOC<sub>2</sub>**

GOE21, 20 (T2OER)	00		Don't care		01, 10, 11	
IEG21, 20 (T2CRL)	00		01, 10, 11		00	
P5 <sub>5</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>5</sub> input port	P5 <sub>5</sub> output port	P5 <sub>5</sub> input port	P5 <sub>5</sub> output port	T2IOC <sub>2</sub> output	
			T2IOC <sub>2</sub> input			

**Table D-21 IPU and P5DDR Settings and Selected Functions of P5<sub>6</sub>/T3IOC<sub>1</sub>**

GOE11, 10 (T3OER)	00		Don't care		01, 10, 11	
IEG11, 10 (T3CRL)	00		01, 10, 11		00	
P5 <sub>6</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>6</sub> input port	P5 <sub>6</sub> output port	P5 <sub>6</sub> input port	P5 <sub>6</sub> output port	T3IOC <sub>1</sub> output	
			T3IOC <sub>1</sub> input			

**Table D-22 IPU and P5DDR Settings and Selected Functions of P5<sub>7</sub>/T3IOC<sub>2</sub>**

GOE21, 20 (T3OER)	00		Don't care		01, 10, 11	
IEG21, 20 (T3CRL)	00		01, 10, 11		00	
P5 <sub>7</sub> DDR	0	1	0	1	0	1
Selected function	P5 <sub>7</sub> input port	P5 <sub>7</sub> output port	P5 <sub>7</sub> input port	P5 <sub>7</sub> output port	T3IOC <sub>2</sub> output	
			T3IOC <sub>2</sub> input			

## D.4 Port 6 Function Selection

**Table D-23 P67CR, PWM3, IRQCR, and P6DDR Settings and Selected Functions of P60/ $\overline{\text{IRQ}}_2$ /PW<sub>3</sub>**

PW <sub>3</sub> E (P67CR)	0				1							
OE (TCR: PWM3)	*				0				1			
IRQ <sub>2</sub> E (IRQCR)	0		1		0		1		0		1	
P6 <sub>1</sub> DDR	0	1	0	1	0	1	0	1	0	1	0	1
Selected function	P6 <sub>0</sub> input port	P6 <sub>0</sub> output port	$\overline{\text{IRQ}}_2$ input P6 <sub>0</sub> input port P6 <sub>0</sub> output port		P6 <sub>0</sub> input port	P6 <sub>0</sub> output port	$\overline{\text{IRQ}}_2$ input P6 <sub>0</sub> input port P6 <sub>0</sub> output port		PW <sub>3</sub> output		PW <sub>3</sub> output and $\overline{\text{IRQ}}_2$ input	

**Table D-24 IRQCR and P6DDR Settings and Selected Functions of P6<sub>1</sub>/ $\overline{\text{IRQ}}_3$**

IRQ <sub>3</sub> E (IRQCR)	0			1	
P6 <sub>1</sub> DDR	0		1		1
Selected function	P6 <sub>1</sub> input port		P6 <sub>1</sub> output port		P6 <sub>1</sub> output port
					$\overline{\text{IRQ}}_3$ input

**Table D-25 IPU and P6DDR Settings and Selected Functions of P6<sub>2</sub>/TCLK<sub>1</sub>**

TPSC3-0 (TCRH)	0000-1100, 1110, 1111			1101	
P6 <sub>2</sub> DDR	0		1		1
Selected function	P6 <sub>2</sub> input port		P6 <sub>2</sub> output port		P6 <sub>2</sub> output port
					TCLK <sub>1</sub> input

**Table D-26 IPU and P6DDR Settings and Selected Functions of P6<sub>3</sub>/TCLK<sub>2</sub>**

TPSC3-0 (TCRH)	0000-1101, 1111			1110	
P6 <sub>3</sub> DDR	0		1		1
Selected function	P6 <sub>3</sub> input port		P6 <sub>3</sub> output port		P6 <sub>3</sub> output port
					TCLK <sub>2</sub> input

**Table D-27 IPU and P6DDR Settings and Selected Functions of P6<sub>4</sub>/TCLK<sub>3</sub>**

TPSC3-0 (TCRH)	0000-1110		1111	
P6 <sub>4</sub> DDR	0	1	0	1
Selected function	P6 <sub>4</sub> input port	P6 <sub>4</sub> output port	P6 <sub>4</sub> input port	P6 <sub>4</sub> output port
			TCLK <sub>3</sub> input	

## D.5 Port 7 Function Selection

**Table D-28 IRQCR and P7DDR Settings and Selected Functions of P7<sub>0</sub>/ $\overline{\text{IRQ}}_0$** 

IRQ0E (IRQCR)	0		1	
P7 <sub>0</sub> DDR	0	1	0	1
Selected function	P7 <sub>0</sub> input port	P7 <sub>0</sub> output port	P7 <sub>0</sub> input port	P7 <sub>0</sub> output port
			$\overline{\text{IRQ}}_0$ input	

**Table D-29 IRQCR, A/D Converter, and P7DDR Settings and Selected Functions of P7<sub>1</sub>/ $\overline{\text{IRQ}}_1$ /ADTRG**

TRGE (ADC: A/D)	0		0		1		1	
IRQ1E (IRQCR)	0		1		0		1	
P7 <sub>1</sub> DDR	0	1	0	1	0	1	0	1
Selected function	*1	*2	*1	*2	*1	*2	*1	*2
			$\overline{\text{IRQ}}_1$ input		ADTRG input		$\overline{\text{IRQ}}_1$ and ADTRG input	

Notes: 1. P7<sub>1</sub> input port  
 2. P7<sub>1</sub> output port

**Table D-30 SCI1 and P7DDR Settings and Selected Functions of P7<sub>2</sub>/TXD<sub>1</sub>**

TE (SCR: SCI1)	0		1	
P7 <sub>2</sub> DDR	0	1	0	1
Selected function	P7 <sub>2</sub> input port	P7 <sub>2</sub> output port	TXD <sub>1</sub> output	

**Table D-31 SCI1 and P7DDR Settings and Selected Functions of P7<sub>3</sub>/RXD<sub>1</sub>**

RE (SCR: SCI1)	0				1			
P7 <sub>3</sub> DDR	0		1		0		1	
Selected function	P7 <sub>3</sub> input port		P7 <sub>3</sub> output port		RXD <sub>1</sub> input			

**Table D-32 SCI2 and P7DDR Settings and Selected Functions of P7<sub>4</sub>/TXD<sub>2</sub>**

TE (SCR: SCI2)	0				1			
P7 <sub>4</sub> DDR	0		1		0		1	
Selected function	P7 <sub>4</sub> input port		P7 <sub>4</sub> output port		TXD <sub>2</sub> output			

**Table D-33 SCI2 and P7DDR Settings and Selected Functions of P7<sub>5</sub>/RXD<sub>2</sub>**

RE (SCR: SCI2)	0				1			
P7 <sub>5</sub> DDR	0		1		0		1	
Selected function	P7 <sub>5</sub> input port		P7 <sub>5</sub> output port		RXD <sub>2</sub> input			

**Table D-34 P67CR, PWM1, SCI1, and P7DDR Settings and Selected Functions of P7<sub>6</sub>/SCK<sub>1</sub>/PW<sub>1</sub>**

PW1E (P67CR)	0						1					
OE (TCR: PWM1)	*						0				1	
C/A (SMR: SCI1)	0			1			*				*	
CKE1 (SMR: SCI1)	0		1		0		1		0		1	
CKE0 (SMR: SCI1)	0		1		*		*		*		*	
P7 <sub>6</sub> DDR	0	1	*	*	*	*	0	1	0	1	*	*
Selected function	P7 <sub>6</sub> input port	P7 <sub>6</sub> output port	SCK <sub>1</sub> output	SCK <sub>1</sub> input	SCK <sub>1</sub> output	SCK <sub>1</sub> input	P7 <sub>6</sub> input port	P7 <sub>6</sub> output port	P7 <sub>6</sub> input port and SCK <sub>1</sub> input	P7 <sub>6</sub> output port and SCK <sub>1</sub> input	PW <sub>1</sub> output	PW <sub>1</sub> output and SCK <sub>1</sub> input

**Table D-35 P67CR, PWM2, SCI2, and P7DDR Settings and Selected Functions of P7<sub>7</sub>/SCK<sub>2</sub>/PW<sub>2</sub>**

PW2E (P67CR)	0						1					
OE (TCR: PWM2)	*						0				1	
C/A (SMR: SCI2)	0			1			*				*	
CKE1 (SMR: SCI2)	0		1	0	1	0	1	0	1	0	1	
CKE0 (SMR: SCI2)	0	1	*	*	*	*	*	*	*	*	*	
P7 <sub>7</sub> DDR	0	1	*	*	*	*	0	1	0	1	*	*
Selected function	P7 <sub>7</sub> input port	P7 <sub>7</sub> output port	SCK <sub>2</sub> output	SCK <sub>2</sub> input	SCK <sub>2</sub> output	SCK <sub>2</sub> input	P7 <sub>7</sub> input port	P7 <sub>7</sub> output port	P7 <sub>7</sub> input port and SCK <sub>2</sub> input	P7 <sub>7</sub> output port and SCK <sub>2</sub> input	PW <sub>2</sub> output	PW <sub>2</sub> output and SCK <sub>2</sub> input

## D.6 Port A Function Selection

**Table D-36 Operating Mode, PACR, IPU, PWM1, and PADDR Settings, and Selected Functions of PA<sub>0</sub>/A<sub>16</sub>/T4OC<sub>1</sub>/PW<sub>1</sub>**

Operating mode	Modes 1, 2, 6, 7						Mode 3 or 5	Mode 4				
PW1E (PACR)	0			1			*	0		1		
OE (TCR: PWM1)	*			0		1	*	*	0		1	
DOE11, 10 (T4OER)	00		01,10,11	*	*	*	*	*	*		*	
PA <sub>0</sub> DDR	0	1	*	0	1	*	*	0	1	0	1	*
Selected function	PA <sub>0</sub> input port	PA <sub>0</sub> output port	T4OC <sub>1</sub> output	PA <sub>0</sub> input port	PA <sub>0</sub> output port	PW <sub>1</sub> output	A <sub>16</sub> address bus	PA <sub>0</sub> input port	A <sub>16</sub> address bus	PA <sub>0</sub> input port	PA <sub>0</sub> output port	PW <sub>1</sub> output

**Table D-37 Operating Mode, PACR, IPU, PWM2, and PADDR Settings, and Selected Functions of PA<sub>1</sub>/A<sub>17</sub>/T4OC<sub>2</sub>/PW<sub>2</sub>**

Operating mode	Modes 1, 2, 6, 7						Mode 3 or 5	Mode 4				
	0		1					0	1			
PW2E (PACR)	0		1				*	0	1			
OE (TCR: PWM2)	*		0	1	*	*	*	0	1			
DOE21, 20 (T4OER)	00		01,10,11	*	*	*	*	*	*			
PA <sub>1</sub> DDR	0	1	*	0	1	*	*	0	1	0	1	*
Selected function	PA <sub>1</sub> input port	PA <sub>1</sub> output port	T4OC <sub>2</sub> output	PA <sub>1</sub> input port	PA <sub>1</sub> output port	PW <sub>2</sub> output	A <sub>17</sub> address bus	PA <sub>1</sub> input port	A <sub>17</sub> address bus	PA <sub>1</sub> input port	PA <sub>1</sub> output port	PW <sub>2</sub> output

**Table D-38 Operating Mode, PACR, IPU, PWM3, and PADDR Settings, and Selected Functions of PA<sub>2</sub>/A<sub>18</sub>/T5OC<sub>1</sub>/PW<sub>3</sub>**

Operating mode	Modes 1, 2, 6, 7						Mode 3 or 5	Mode 4				
	0		1					0	1			
PW3E (PACR)	0		1				*	0	1			
OE (TCR: PWM3)	*		0	1	*	*	*	0	1			
DOE11, 10 (T5OER)	00		01,10,11	*	*	*	*	*	*			
PA <sub>2</sub> DDR	0	1	*	0	1	*	*	0	1	0	1	*
Selected function	PA <sub>2</sub> input port	PA <sub>2</sub> output port	T5OC <sub>1</sub> output	PA <sub>2</sub> input port	PA <sub>2</sub> output port	PW <sub>3</sub> output	A <sub>18</sub> address bus	PA <sub>2</sub> input port	A <sub>18</sub> address bus	PA <sub>2</sub> input port	PA <sub>2</sub> output port	PW <sub>3</sub> output

**Table D-39 (1) Operating Mode, PACR, IPU, SCI3, and PADDR Settings, and Selected Functions of PA<sub>3</sub>/A<sub>19</sub>/T5OC<sub>2</sub>/SCK<sub>3</sub>**

Operating mode	Modes 1, 2, 6, 7								Mode 3 or 5
SCK3E (PACR)	0			1					*
C/A (SMR: SCI3)	*			0					*
CKE1 (SMR: SCI3)	*			0			1		*
CKE0 (SMR: SCI3)	*			0	1	0	1	*	
DOE11, 10 (T5OER)	00		01, 10, 11	*	*	*	*	*	
PA <sub>3</sub> DDR	0	1	*	0	1	*	*	*	*
Selected function	PA <sub>3</sub> input port	PA <sub>3</sub> output port	T5OC <sub>2</sub> output	PA <sub>3</sub> input port	PA <sub>3</sub> output port	SCK <sub>3</sub> output	SCK <sub>3</sub> input	SCK <sub>3</sub> input	A <sub>19</sub> address bus

**Table D-39 (2) Operating Mode, PACR, IPU, SCI3, and PADDR Settings, and Selected Functions of PA<sub>3</sub>/A<sub>19</sub>/T5OC<sub>2</sub>/SCK<sub>3</sub>**

Operating mode	Mode 4						
SCK3E (PACR)	0			1			
C/A (SMR: SCI3)	*			0			
CKE1 (SMR: SCI3)	*			0			1
CKE0 (SMR: SCI3)	*			0	1	0	1
DOE11, 10 (T5OER)	*			*	*	*	*
PA <sub>3</sub> DDR	0	1	0	1	*	*	*
Selected function	PA <sub>3</sub> input port	A <sub>19</sub> address bus	PA <sub>3</sub> input port	PA <sub>3</sub> output port	SCK <sub>3</sub> output	SCK <sub>3</sub> input	SCK <sub>3</sub> input

**Table D-40 Operating Mode, WCR and PADDR Settings, and Selected Functions of PA<sub>4</sub>/WAIT**

Operating Mode	Modes 1 to 6				Mode 7	
WMS1 (WCR)	0			1	Don't care	
PA <sub>4</sub> DDR	0	1	0	1	0	1
Selected function	PA <sub>4</sub> input port	PA <sub>4</sub> output port	WAIT input		PA <sub>4</sub> input port	PA <sub>4</sub> output port

**Table D-41 (1) Operating Mode, PACR, BRCCR, IPU, SCI3, and PADDR Settings, and Selected Functions of PA<sub>5</sub>/T3OC<sub>1</sub>/ $\overline{\text{BREQ}}$ /RXD<sub>3</sub>**

Operating mode	Modes 1 to 6									
RXD3E (PACR)	0				1				1	
RE (SCR: SCI3)	*				0				1	
BRLE (BRCCR)	0		1		0		1		0	1
DOE11, 10 (T3OER)	00		01, 10, 11		*		00		01, 10, 11	
PA <sub>5</sub> DDR	0	1	*	*	0	1	*	*	*	*
Selected function	PA <sub>5</sub> input port	PA <sub>5</sub> output port	T3OC <sub>1</sub> output	$\overline{\text{BREQ}}$ input	PA <sub>5</sub> input port	PA <sub>5</sub> output port	T3OC <sub>1</sub> output	$\overline{\text{BREQ}}$ input	RXD <sub>3</sub> input	$\overline{\text{BREQ}}$ input and RXD <sub>3</sub> input

**Table D-41 (2) Operating Mode, PACR, BRCCR, IPU, SCI3, and PADDR Settings, and Selected Functions of PA<sub>5</sub>/T3OC<sub>1</sub>/ $\overline{\text{BREQ}}$ /RXD<sub>3</sub>**

Operating mode	Mode 7							
RXD3E (PACR)	0				1			
RE (SCR: SCI3)	*				0			
BRLE (BRCCR)	*				*			
DOE11, 10 (T3OER)	00		01, 10, 11		00		01, 10, 11	
PA <sub>5</sub> DDR	0	1	*	0	1	*	*	
Selected function	PA <sub>5</sub> input port	PA <sub>5</sub> output port	T3OC <sub>1</sub> output	PA <sub>5</sub> input port	PA <sub>5</sub> output port	T3OC <sub>1</sub> output	RXD <sub>3</sub> input	



**Table D-42 (1) Operating Mode, PACR, BRCR, IPU, SCI3, and PADDR Settings, and Selected Functions of PA<sub>6</sub>/T3OC<sub>2</sub>/ $\overline{\text{BACK}}$ /TXD<sub>3</sub>**

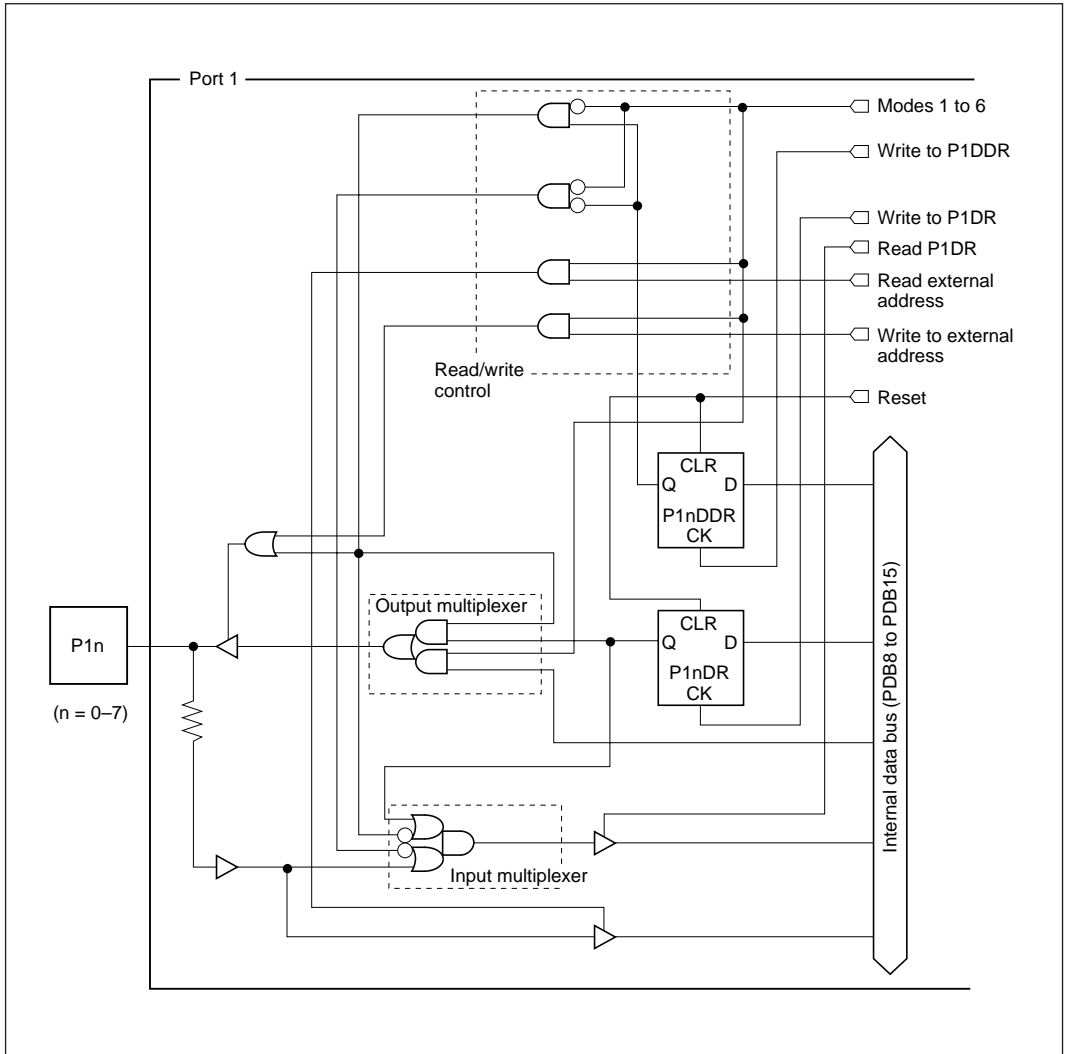
Operating mode	Modes 1 to 6								
TXD3E (PACR)	0			1			1		
TE (SCR: SCI3)	*			0			1		
BRLE (BRCR)	0		1	0		1	0	1	
DOE21, 20 (T3OER)	00		01, 10, 11	*		*	*	*	
PA <sub>6</sub> DDR	0	1	*	*	0	1	*	*	*
Selected function	PA <sub>6</sub> input port	PA <sub>6</sub> output port	T3OC <sub>2</sub> output	$\overline{\text{BACK}}$ output	PA <sub>6</sub> input port	PA <sub>6</sub> output port	$\overline{\text{BACK}}$ output	TXD <sub>3</sub> output	$\overline{\text{BACK}}$ output

Note: For the H8/538, refer to the case where TXD3E = 0.

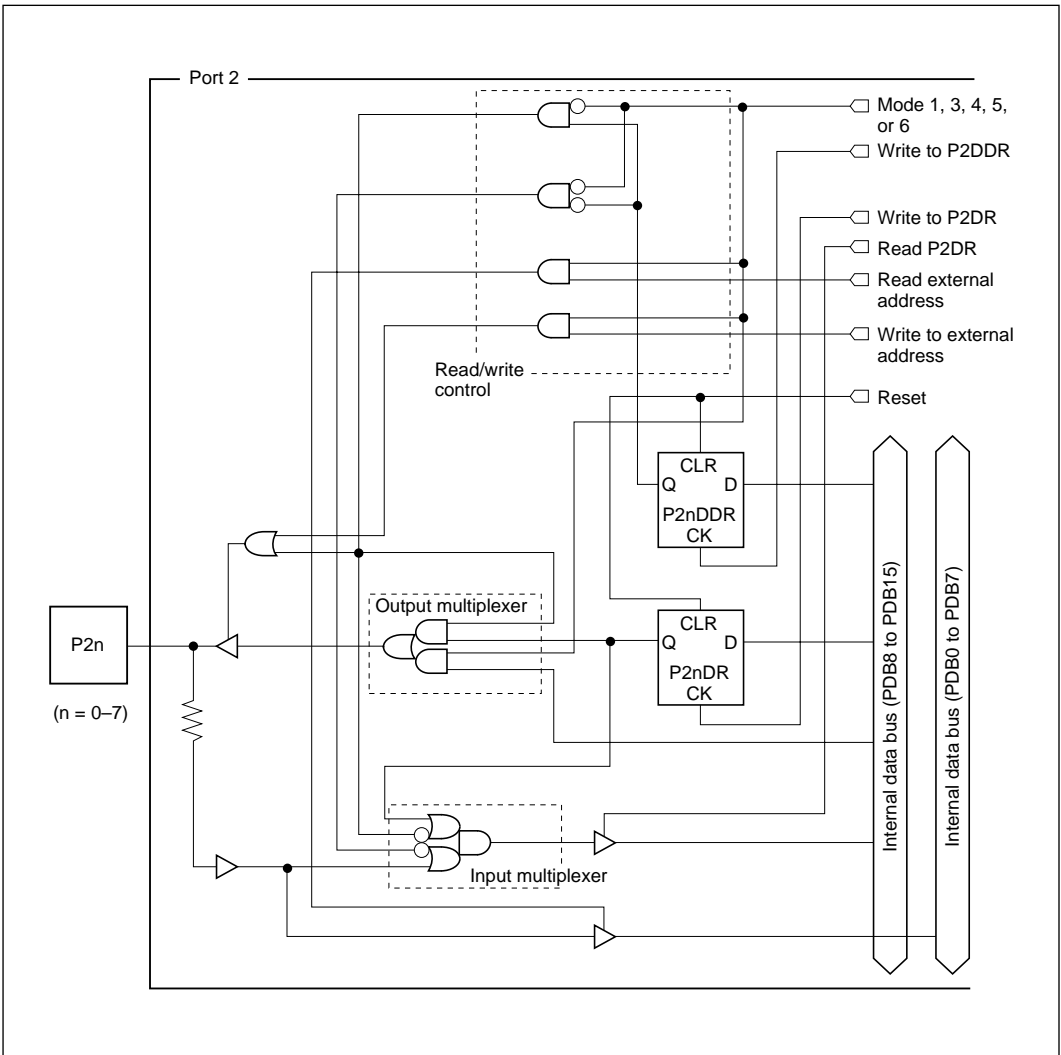
**Table D-42 (2) Operating Mode, PACR, BRCR, IPU, SCI3, and PADDR Settings, and Selected Functions of PA<sub>6</sub>/T3OC<sub>2</sub>/ $\overline{\text{BACK}}$ /TXD<sub>3</sub>**

Operating mode	Mode 7					
TXD3E (PACR)	0			1		1
TE (SCR: SCI3)	*			0		1
BRLE (BRCR)	*			*		*
DOE21, 20 (T3OER)	00		01, 10, 11	*		*
PA <sub>6</sub> DDR	0	1	*	0	1	*
Selected function	PA <sub>6</sub> input port	PA <sub>6</sub> output port	T3OC <sub>2</sub> output	PA <sub>6</sub> input port	PA <sub>6</sub> output port	TXD <sub>3</sub> output

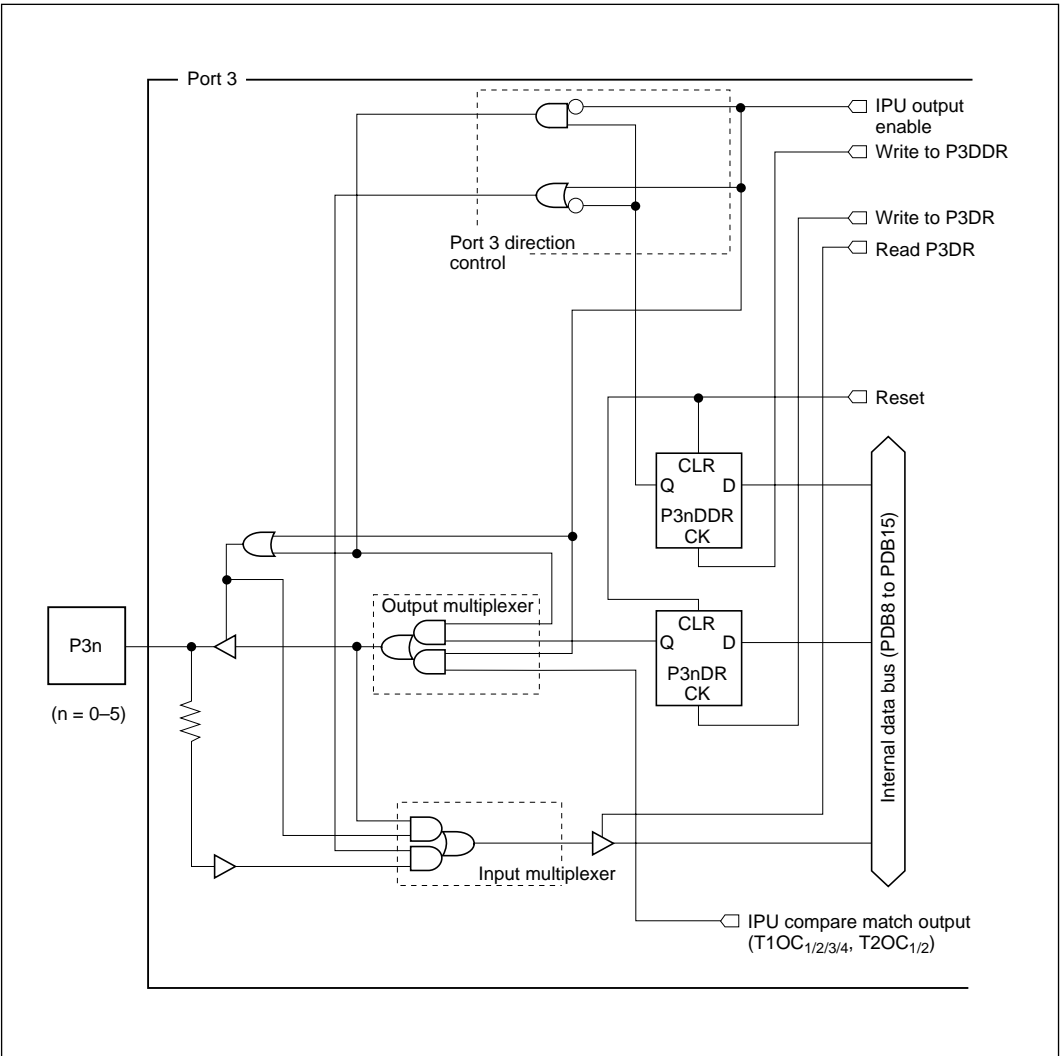
# Appendix E I/O Port Block Diagrams



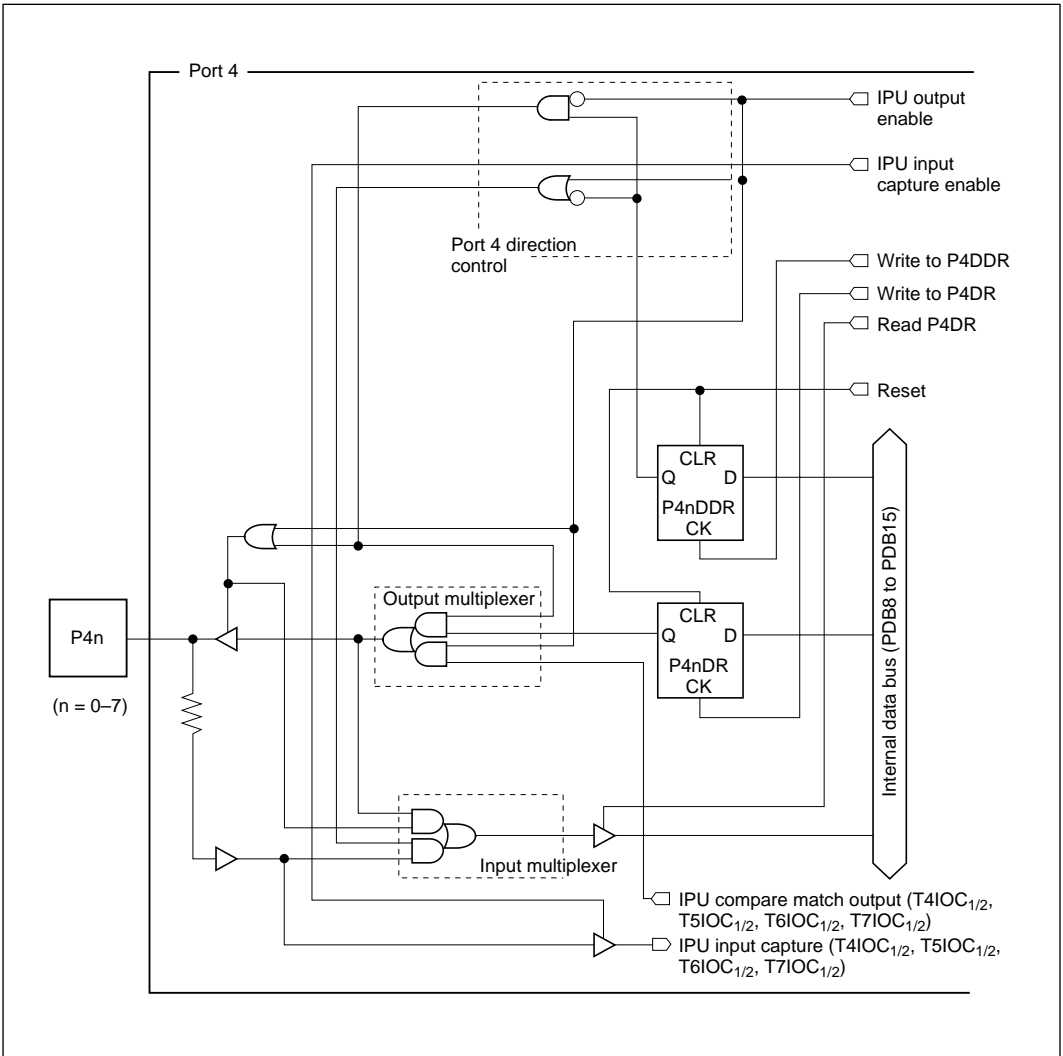
**Figure E-1 Port 1 Block Diagram**



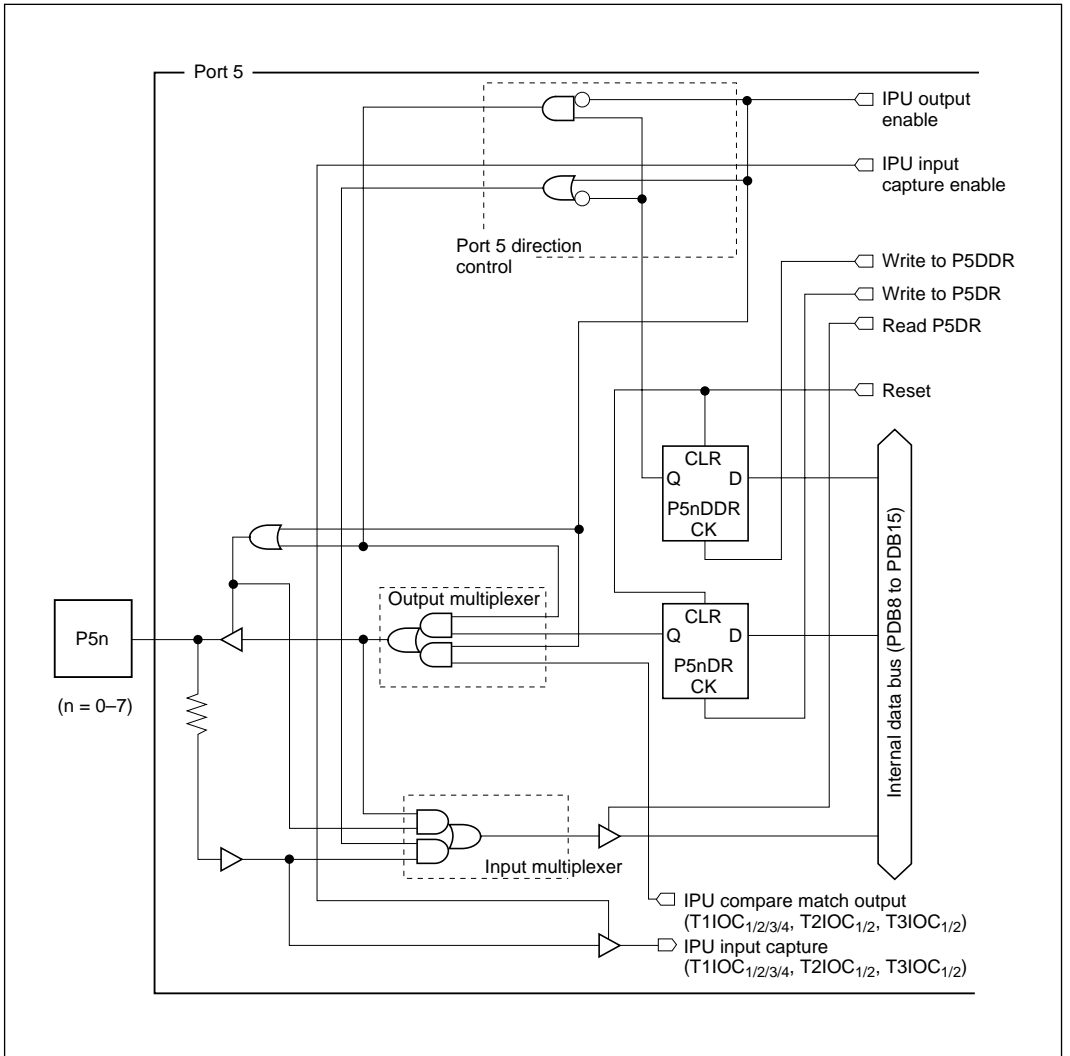
**Figure E-2 Port 2 Block Diagram**



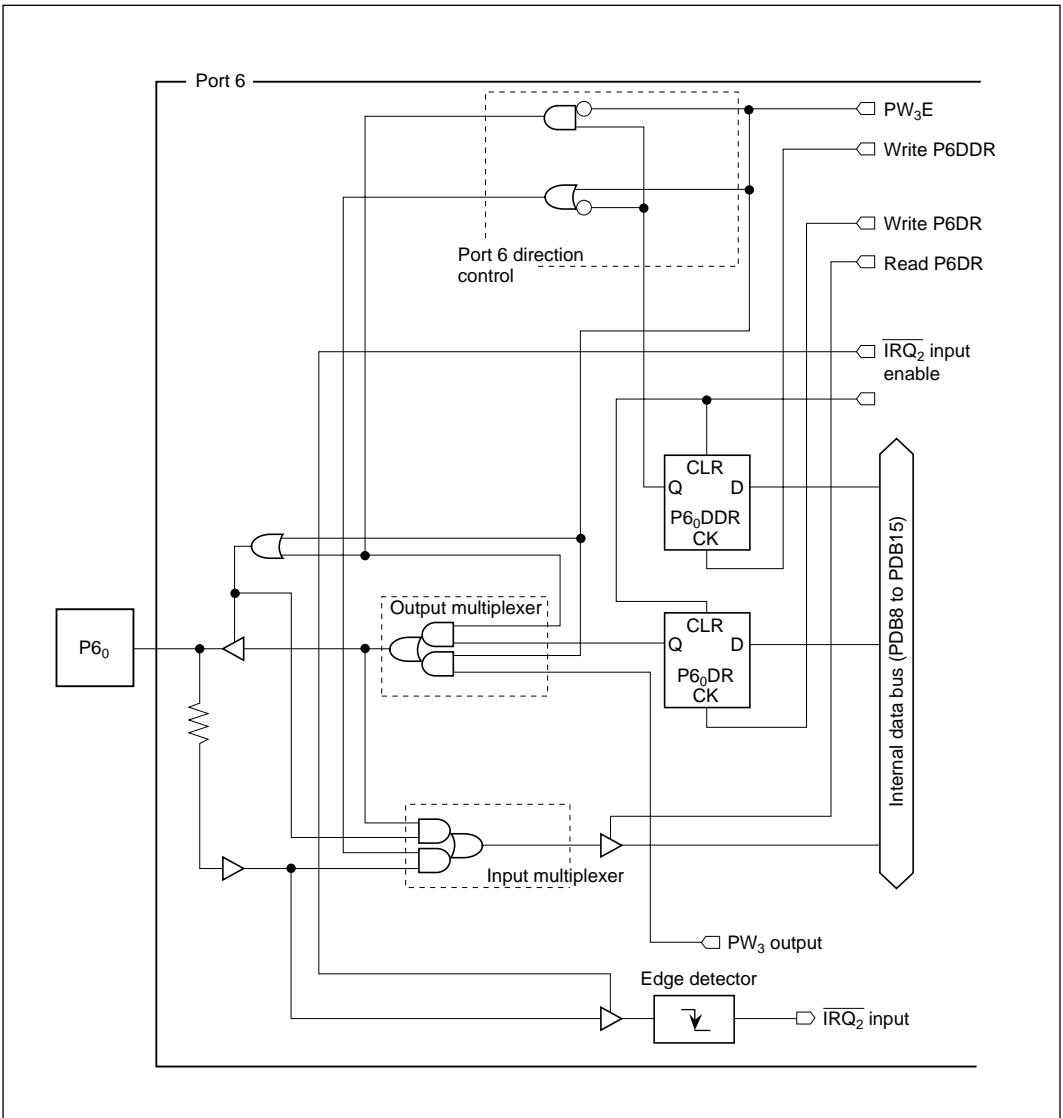
**Figure E-3 Port 3 Block Diagram**



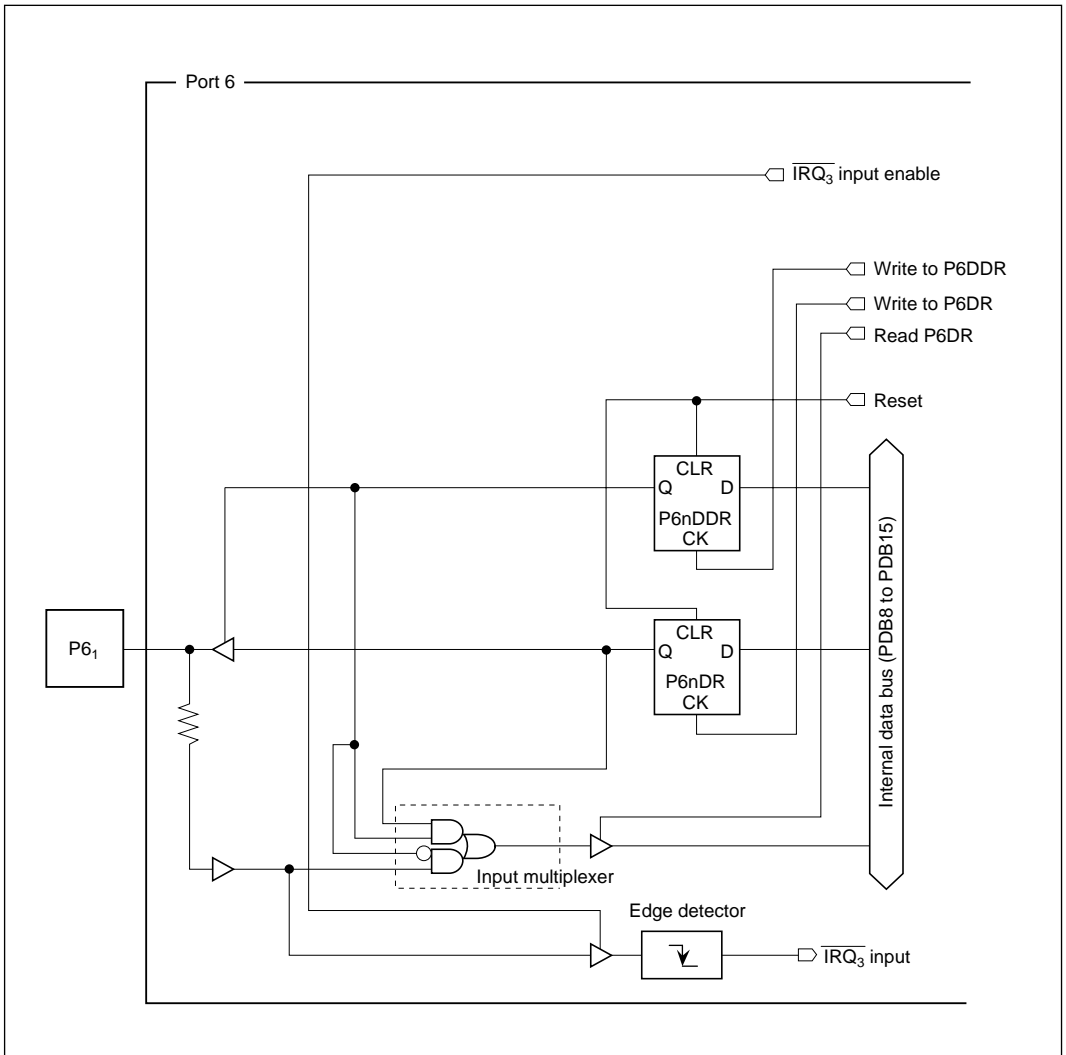
**Figure E-4 Port 4 Block Diagram**



**Figure E-5 Port 5 Block Diagram**

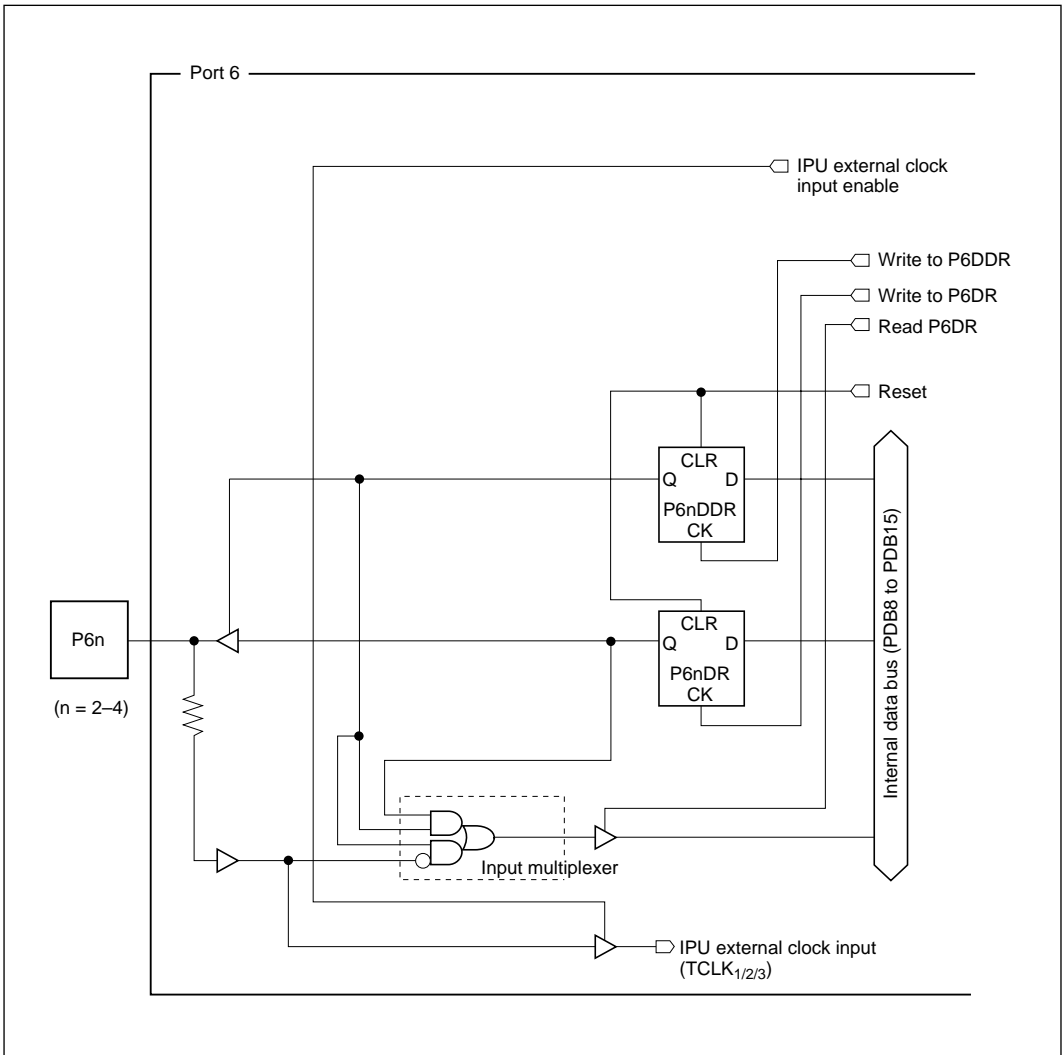


**Figure E-6 Port 6 Block Diagram (1)**

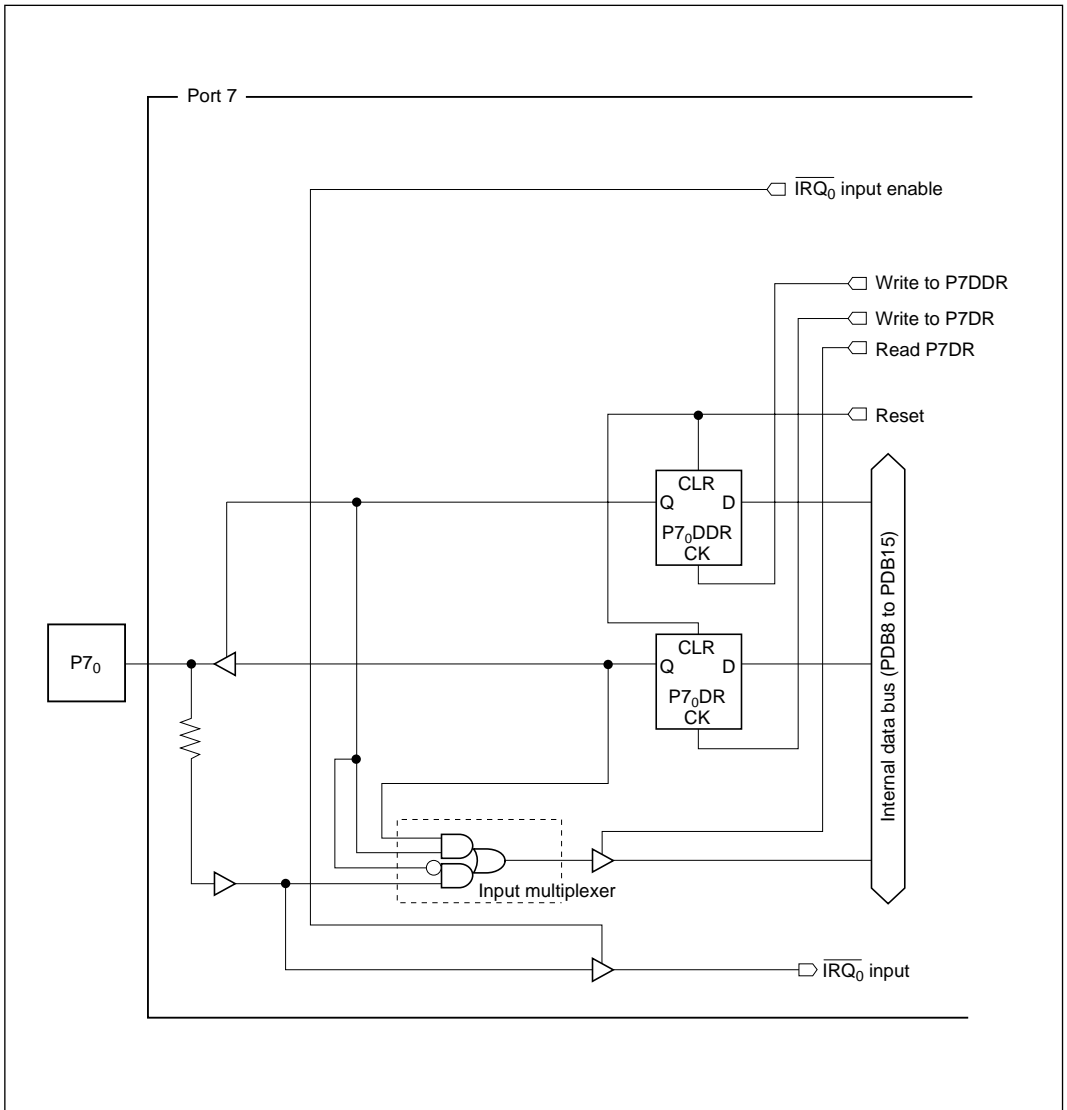


**Figure E-7 Port 6 Block Diagram (2)**

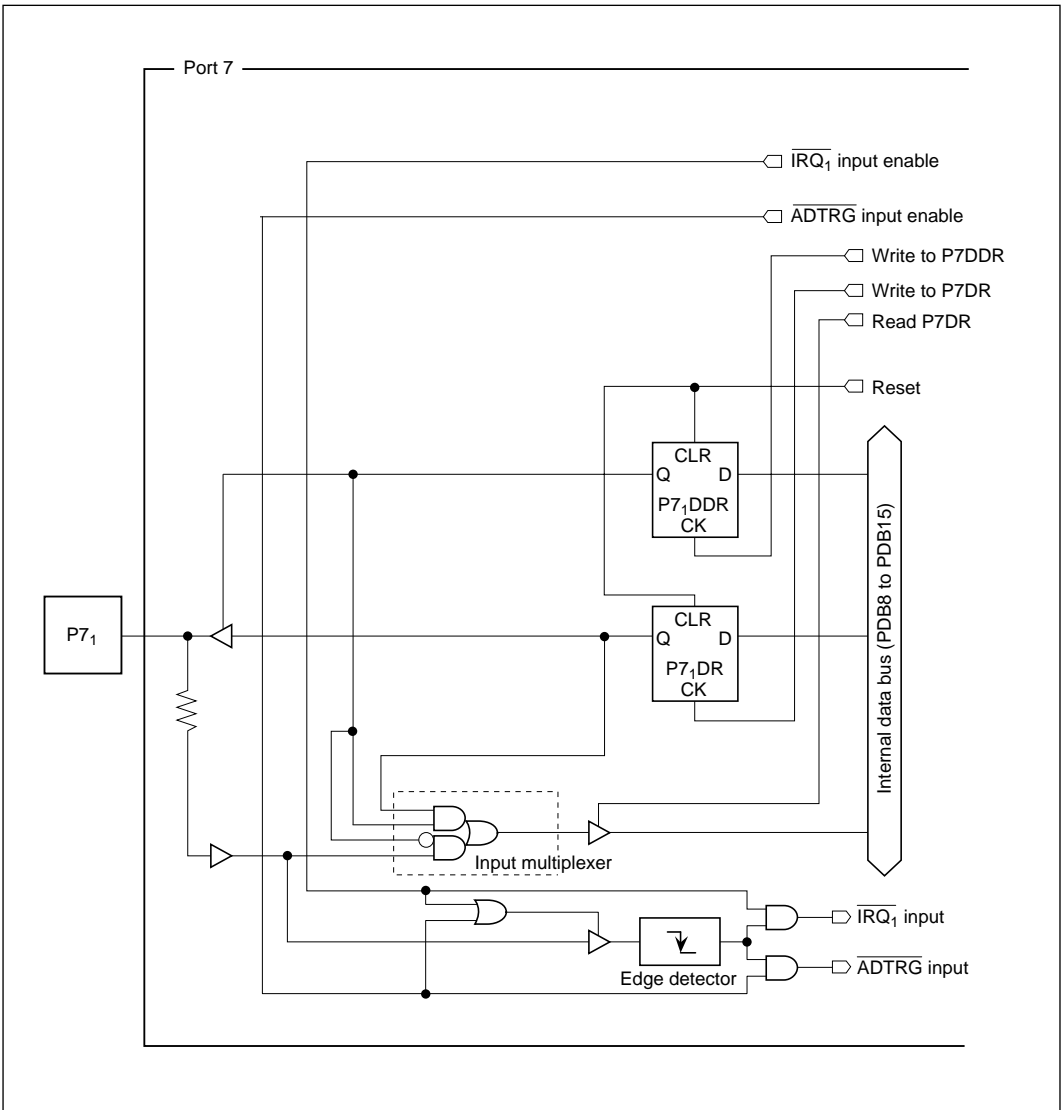




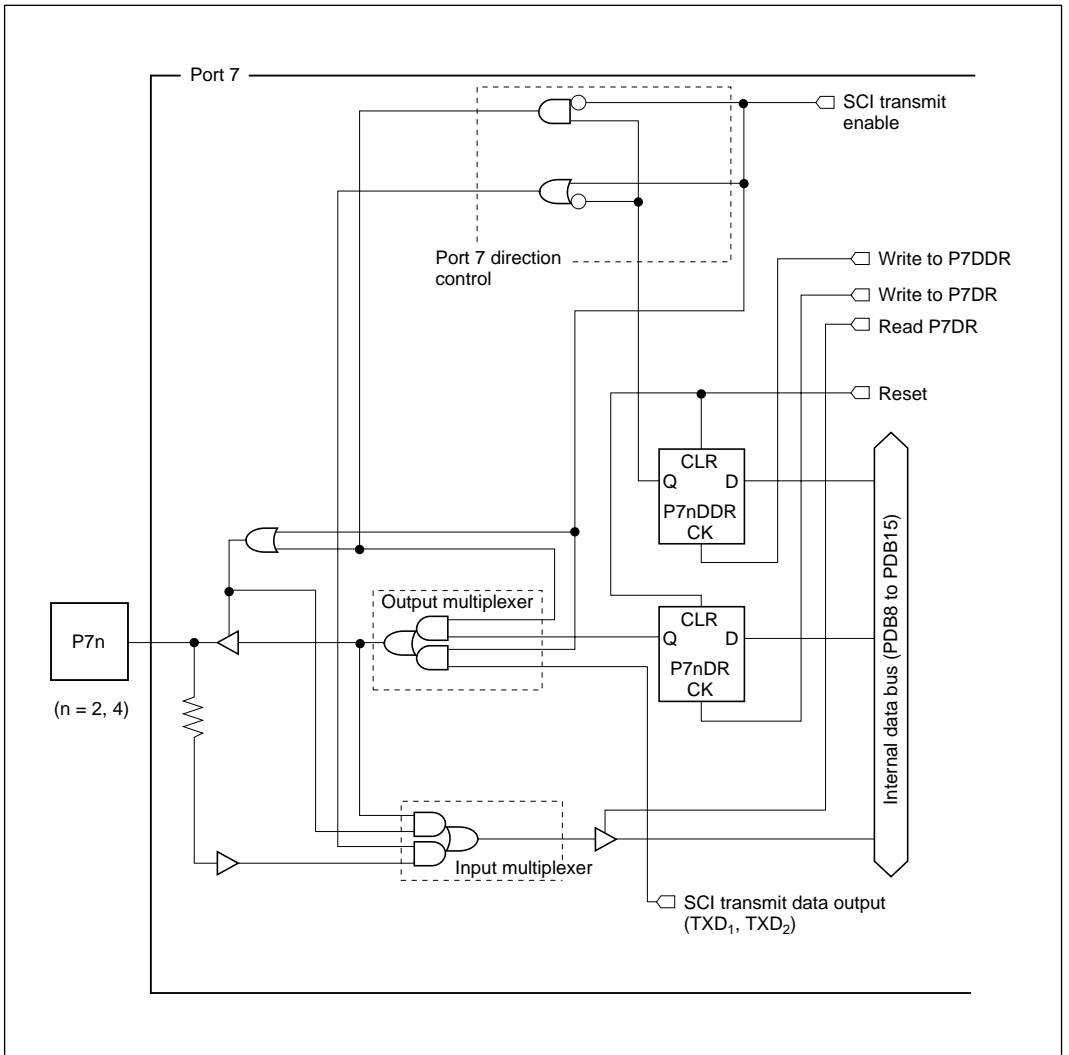
**Figure E-8 Port 6 Block Diagram (3)**



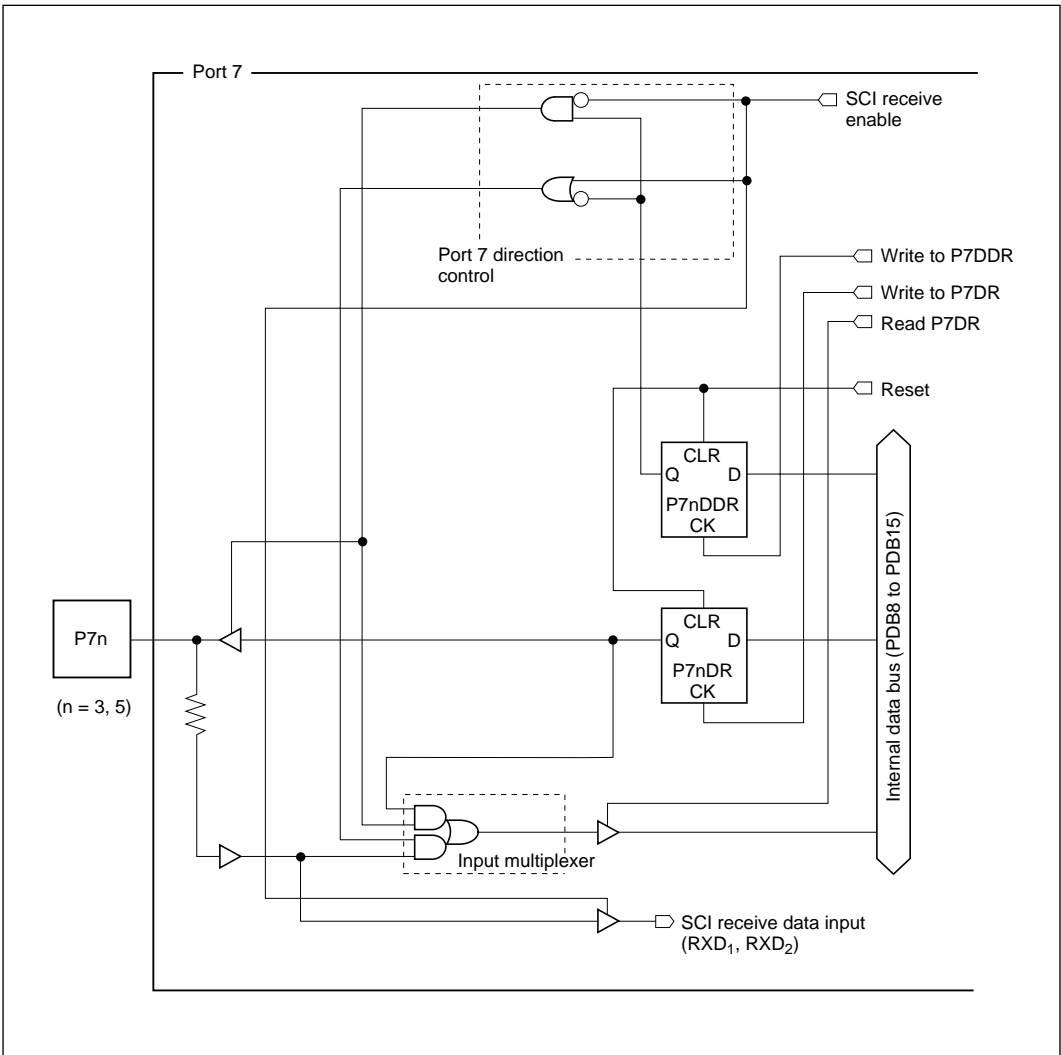
**Figure E-9 Port 7 Block Diagram (1)**



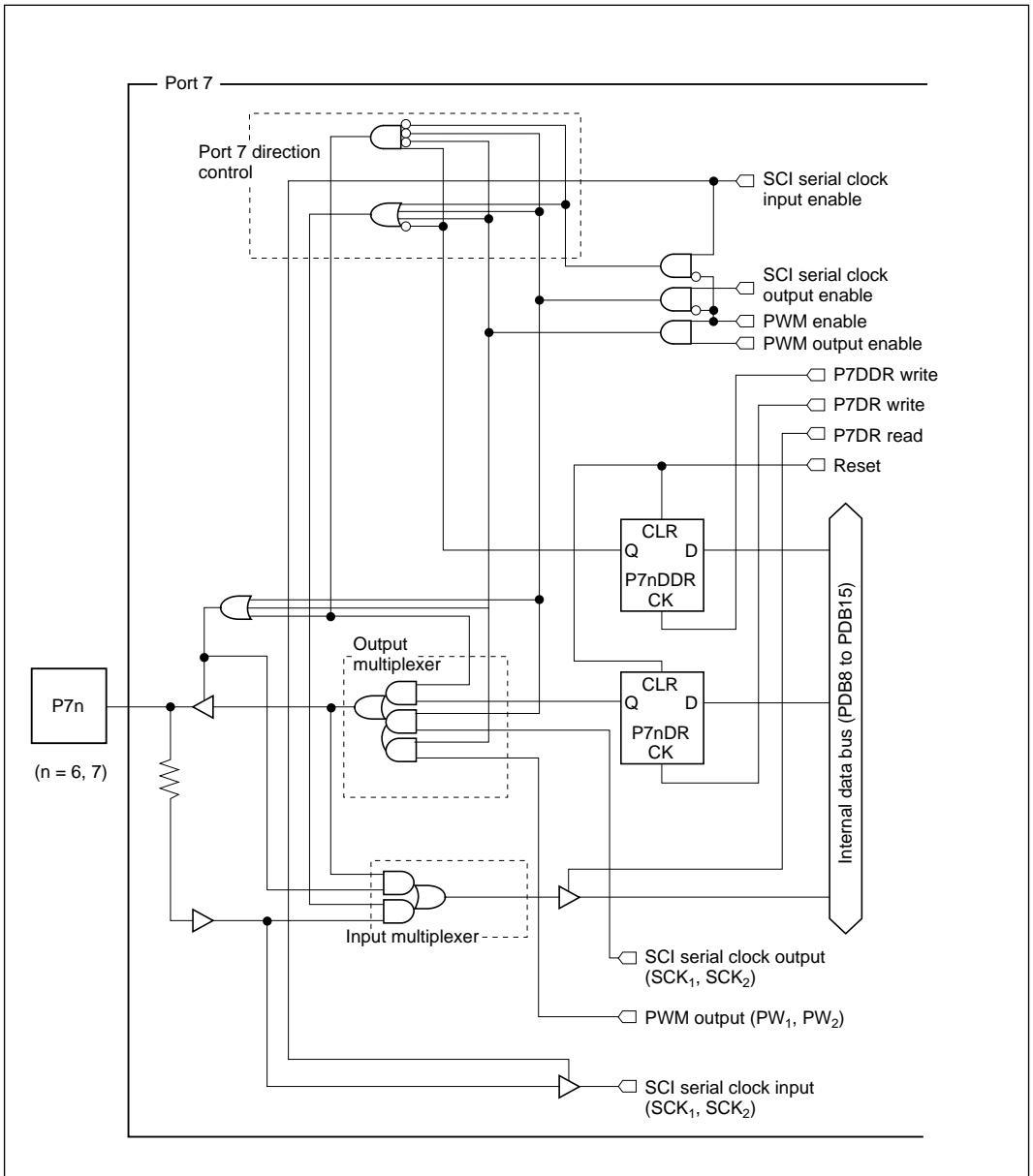
**Figure E-10 Port 7 Block Diagram (2)**



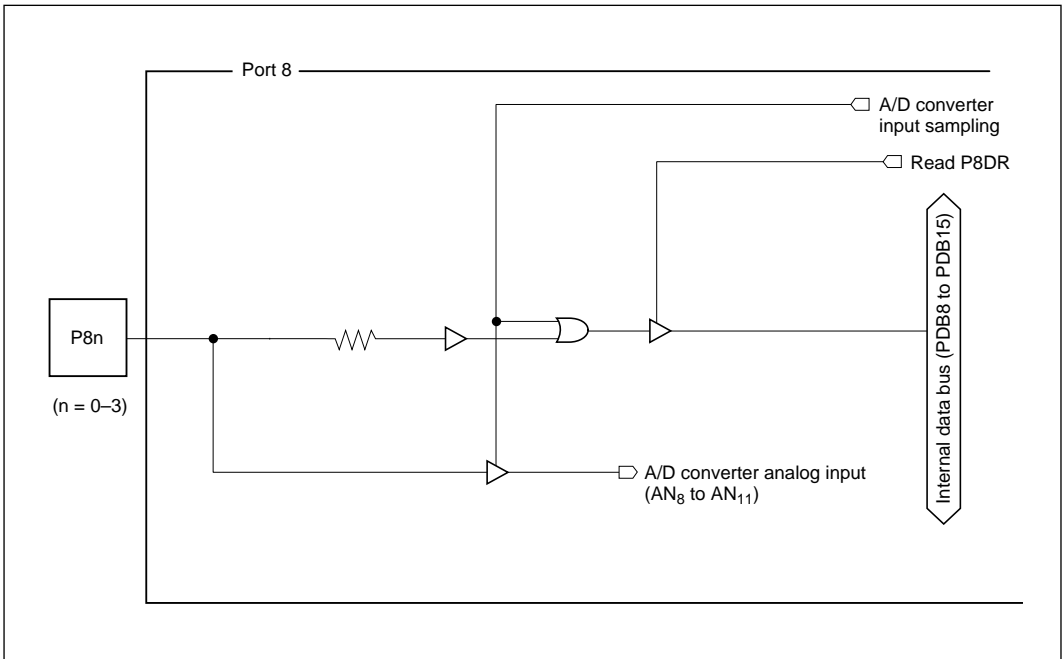
**Figure E-11 Port 7 Block Diagram (3)**



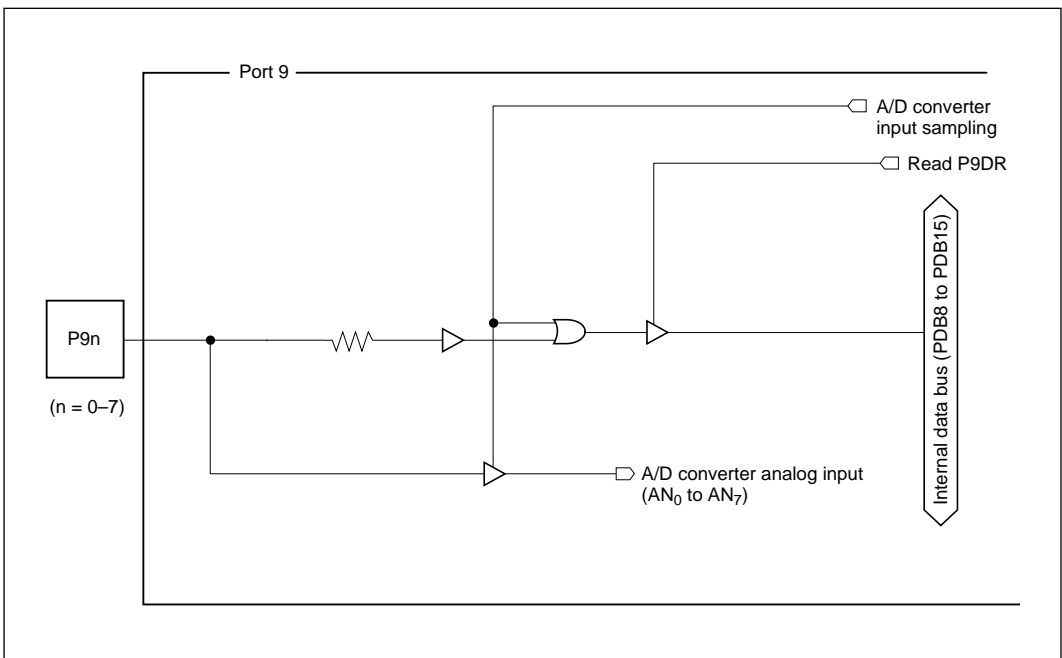
**Figure E-12 Port 7 Block Diagram (4)**



**Figure E-13 Port 7 Block Diagram (5)**



**Figure E-14 Port 8 Block Diagram**



**Figure E-15 Port 9 Block Diagram**

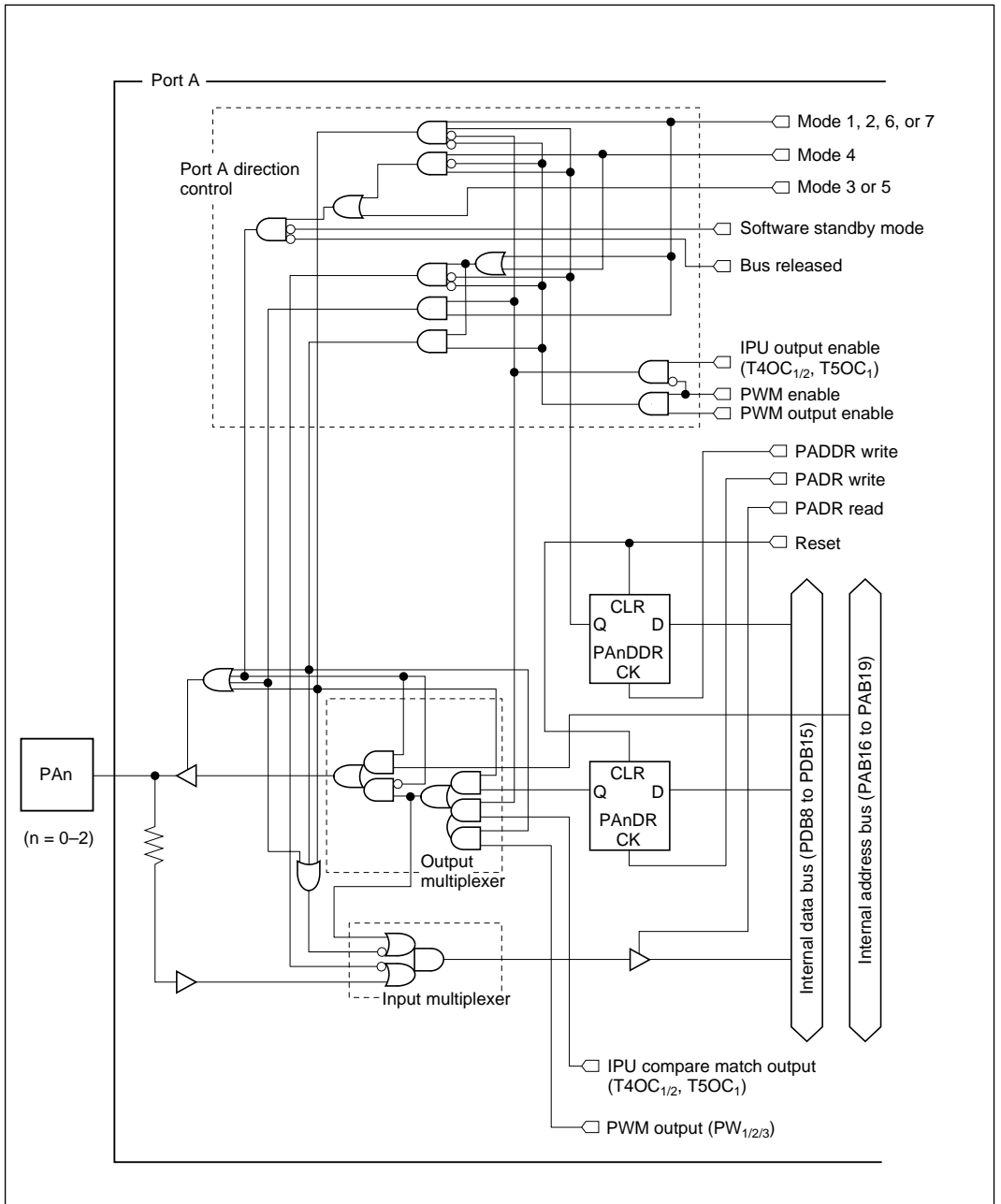


Figure E-16 Port A Block Diagram (1)



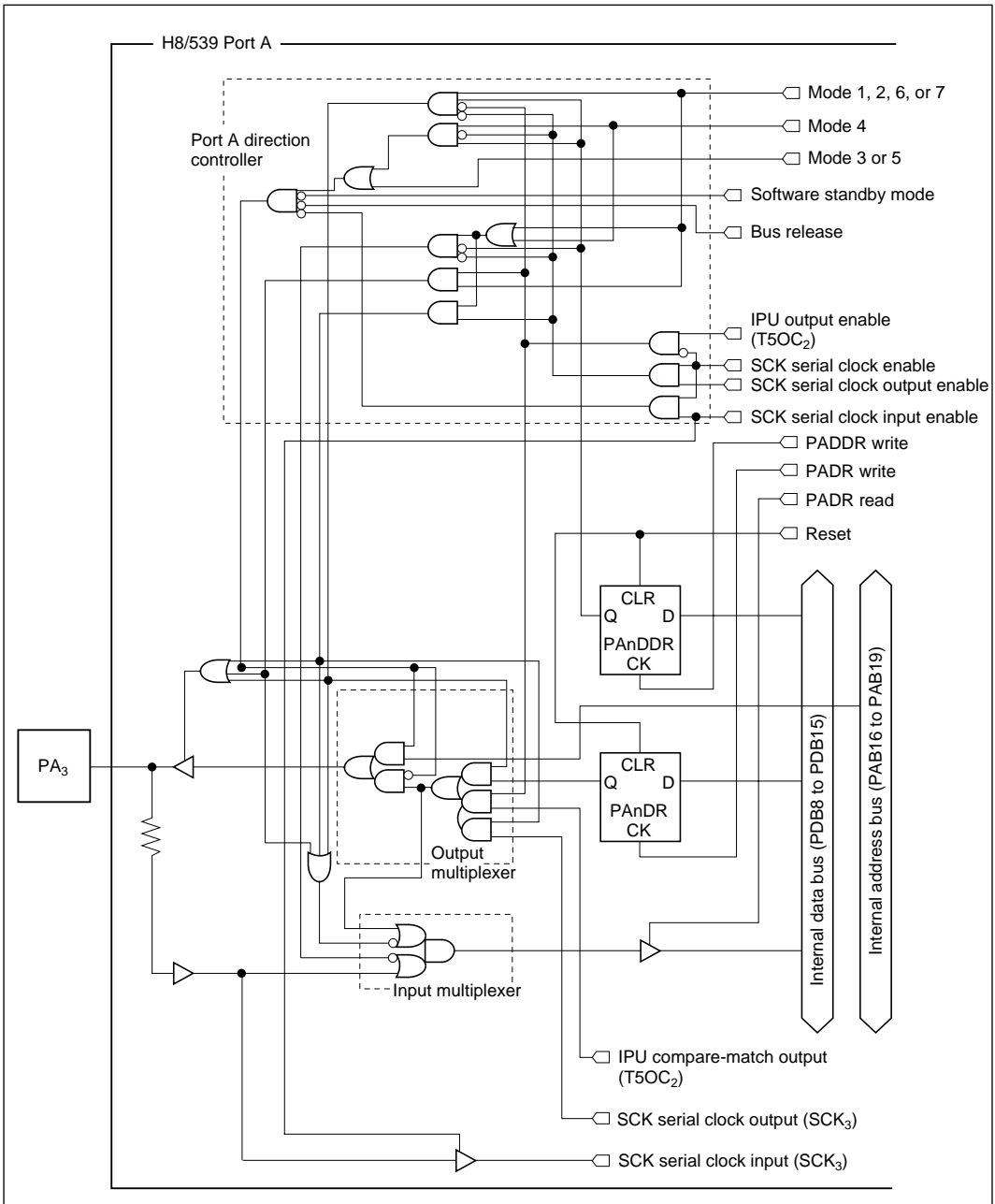
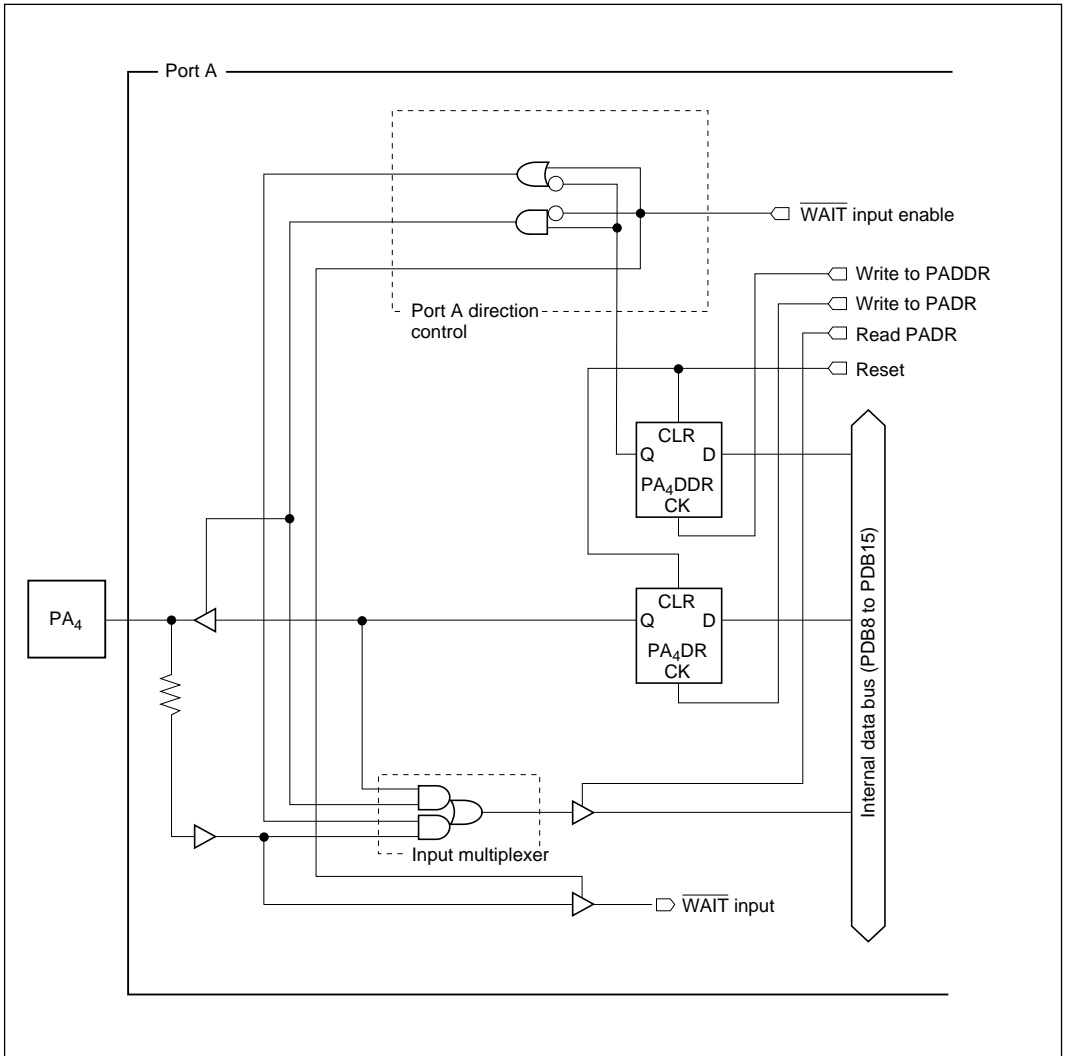
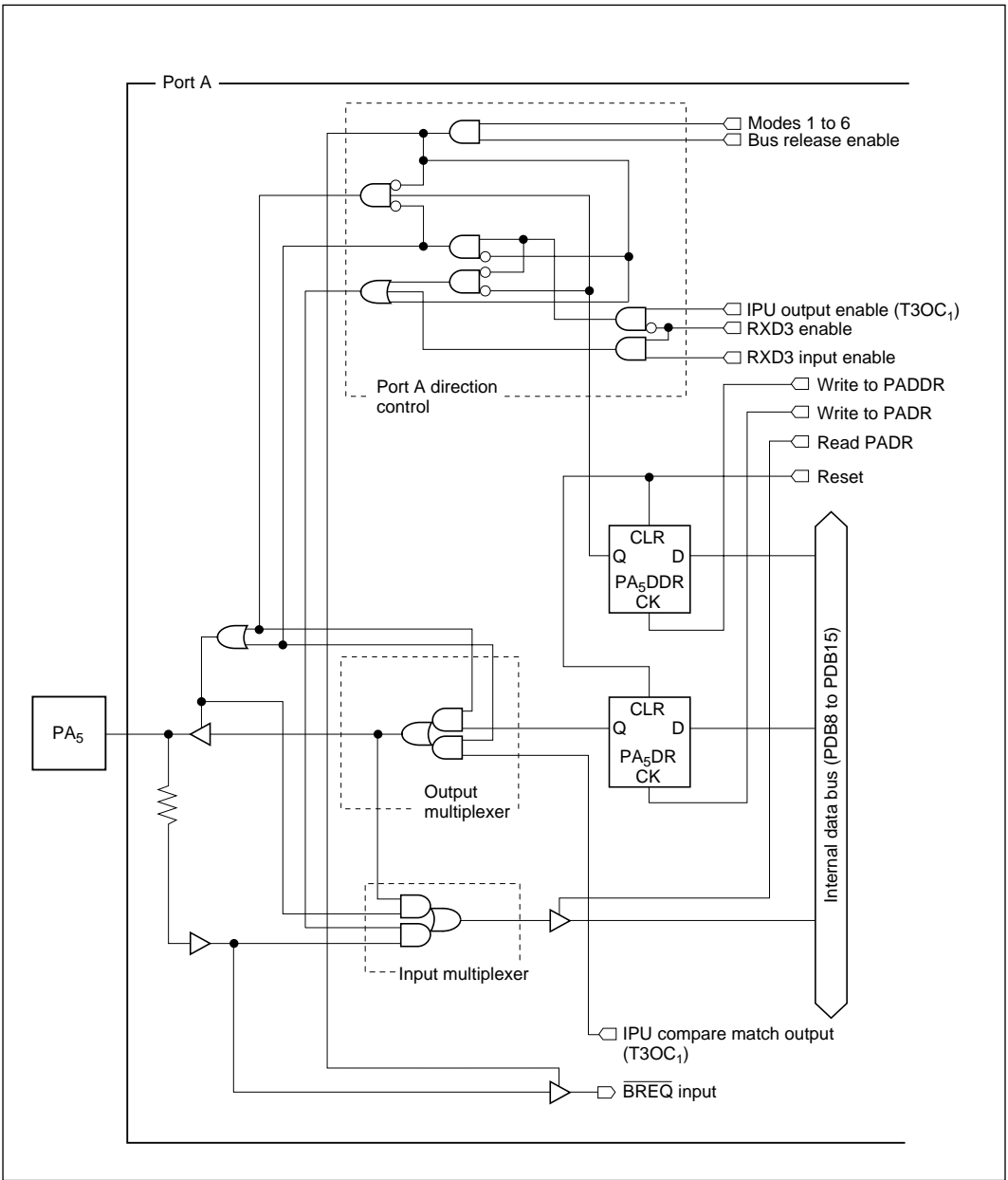


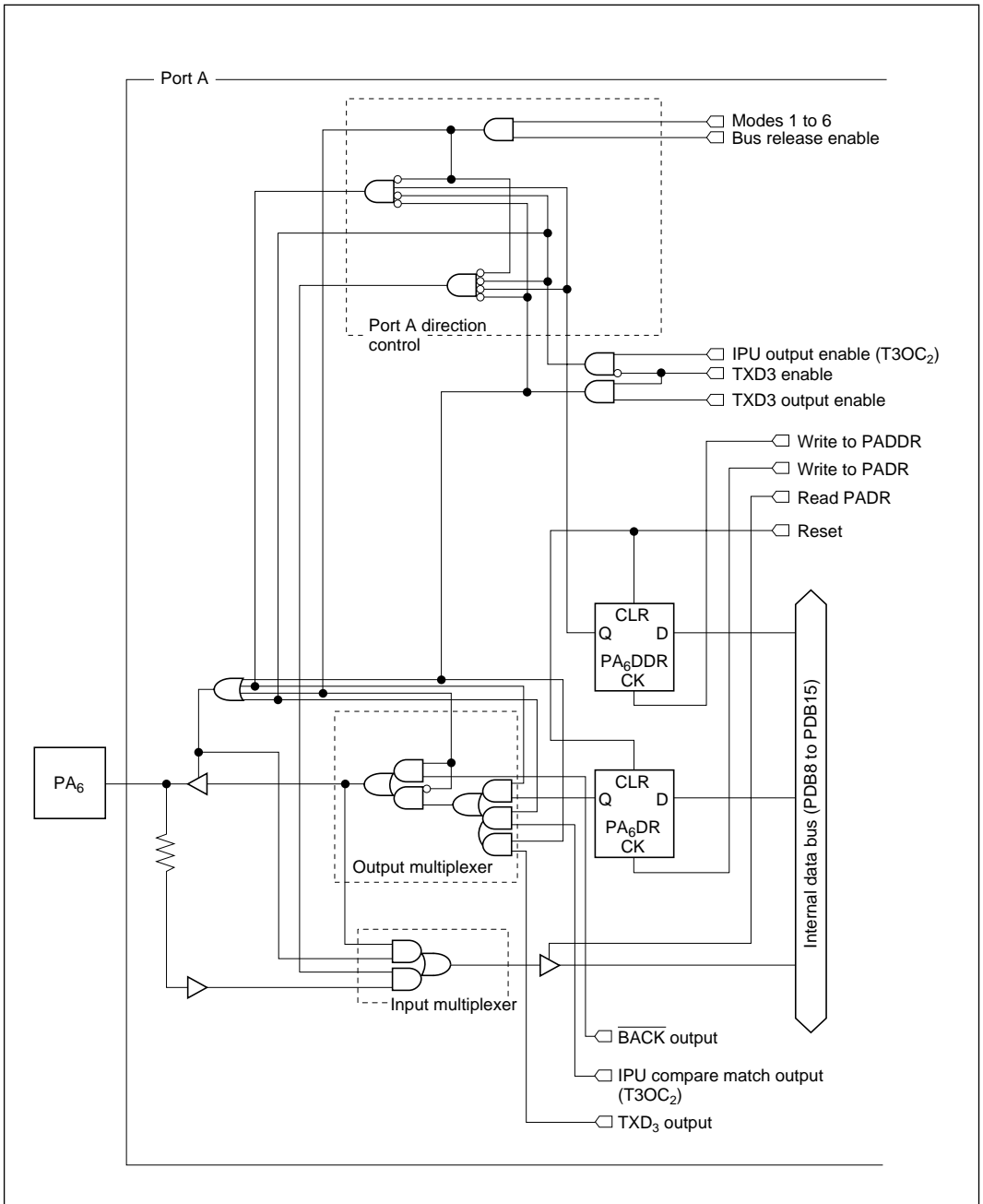
Figure E-17 Port A Block Diagram (2)



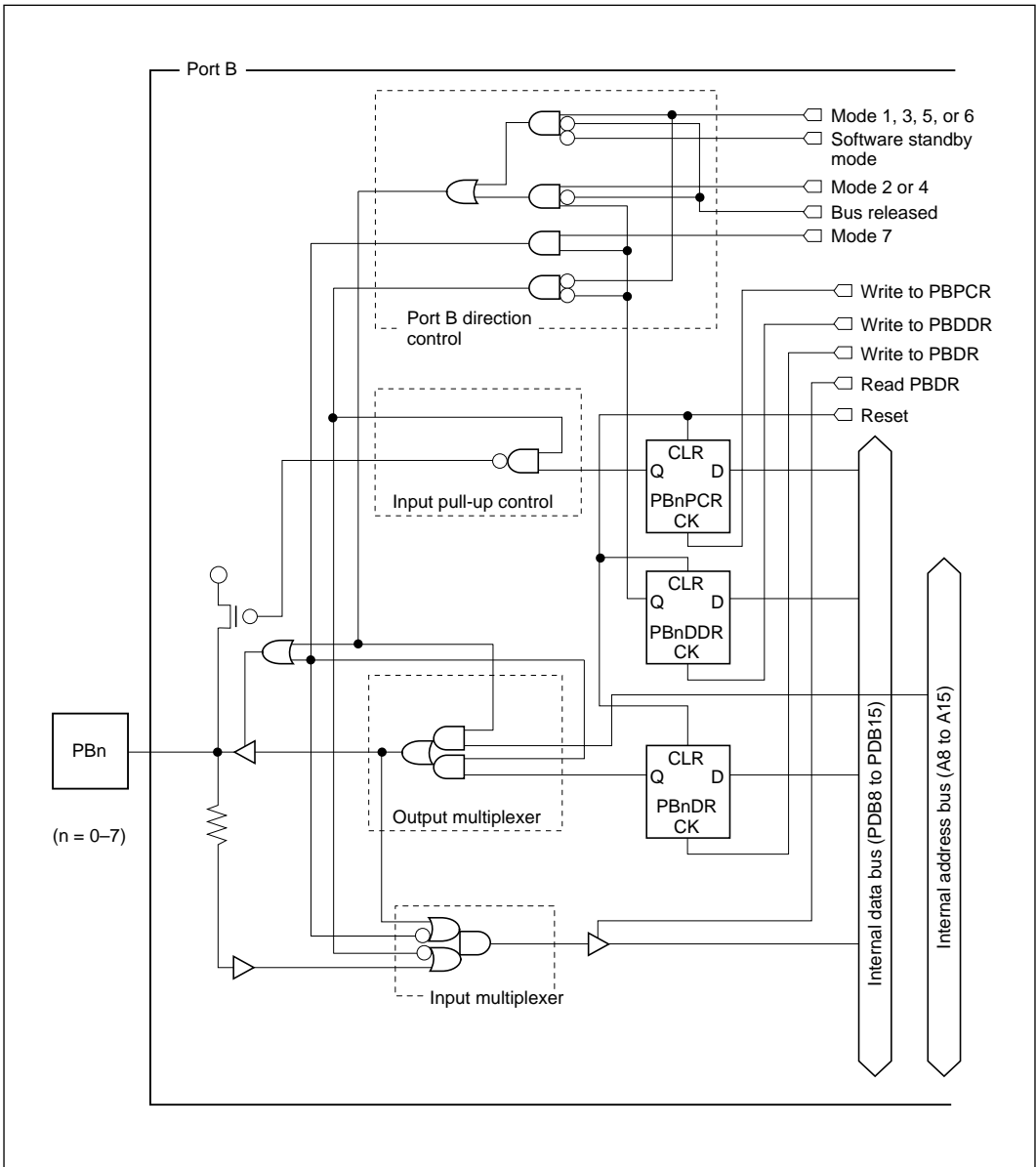
**Figure E-18 Port A Block Diagram (3)**



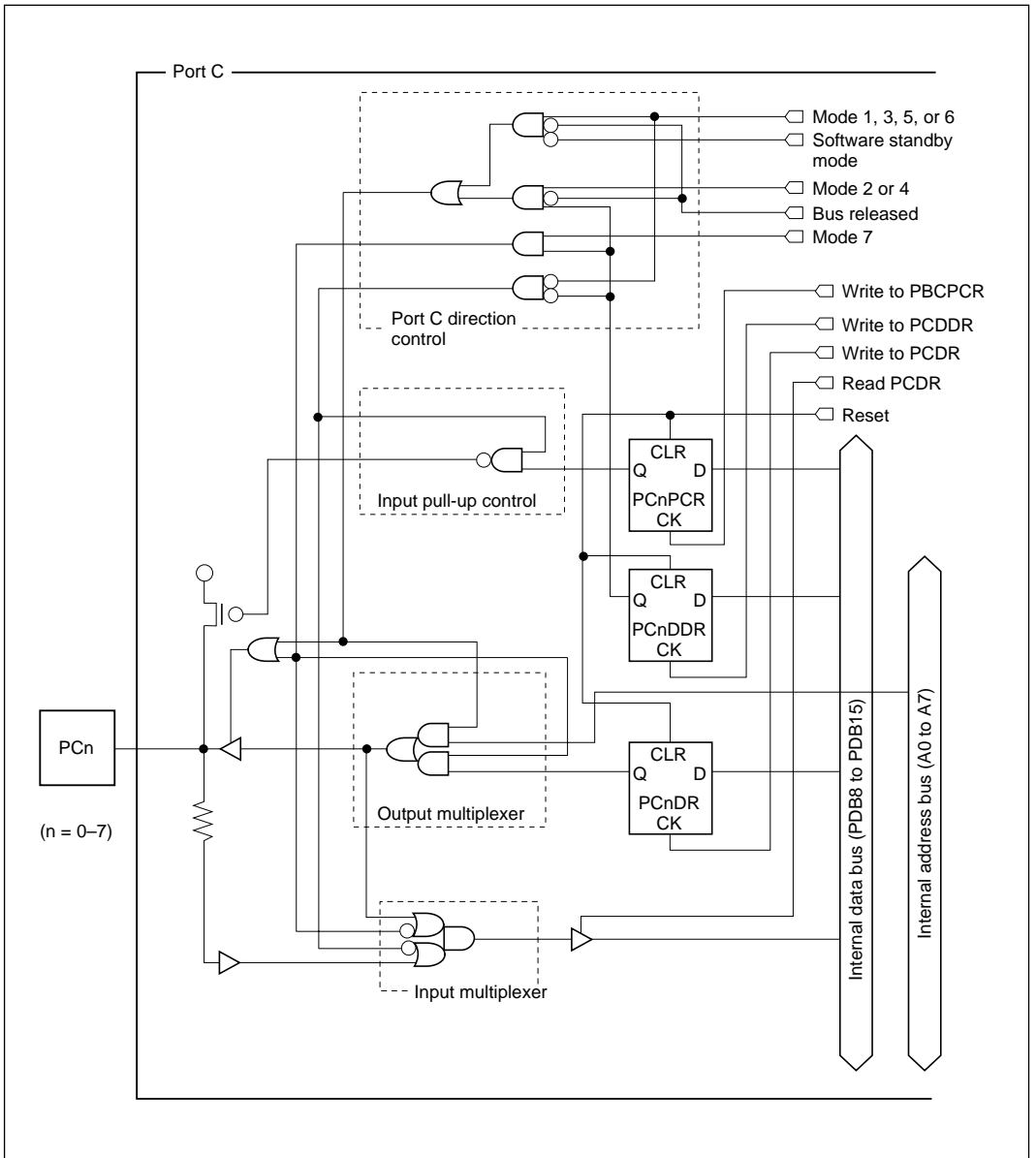
**Figure E-19 Port A Block Diagram (4)**



**Figure E-20 Port A Block Diagram (5)**

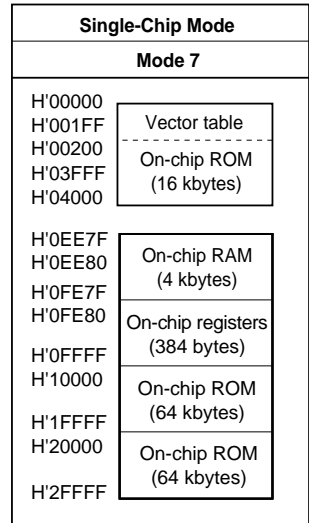
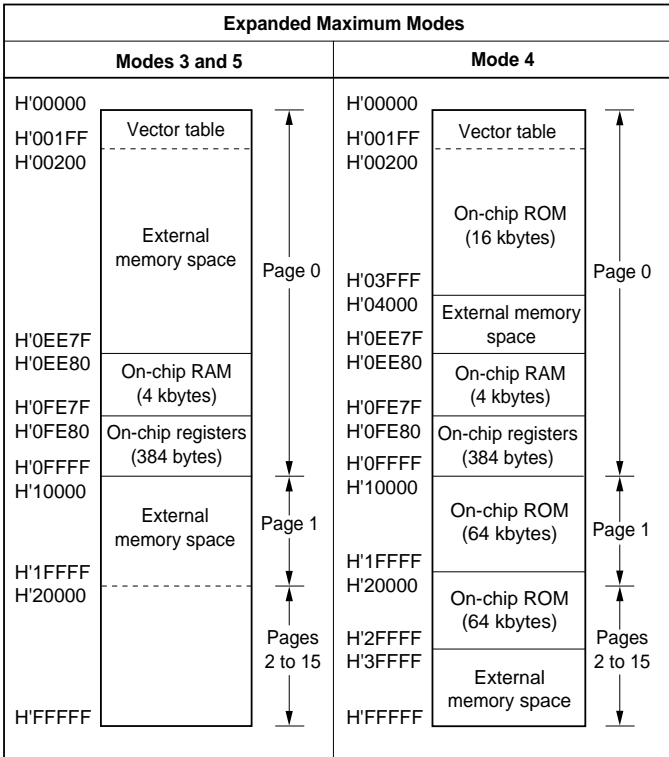
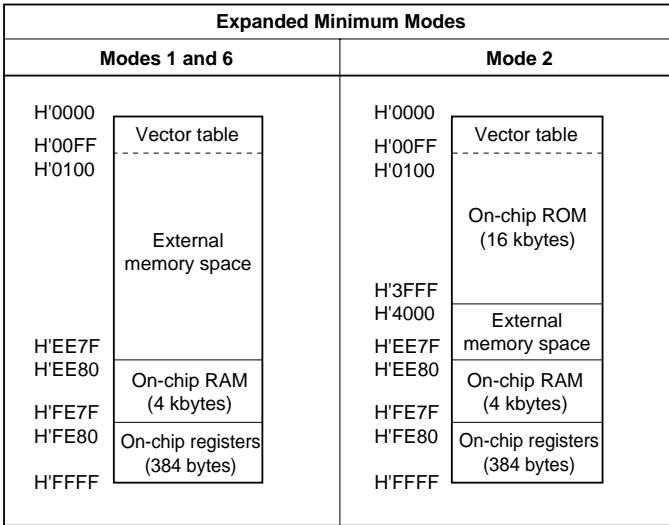


**Figure E-21 Port B Block Diagram**



**Figure E-22 Port C Block Diagram**

# Appendix F Memory Maps



# Appendix G Pin States

## G.1 States of I/O Ports

Table G-1 States of I/O Ports

Pin Name	Mode	Reset	Hardware Standby Mode	Software Standby Mode	Sleep Mode	Bus Release Mode	Program Execution Mode (normal operation)
∅	—	Clock output	T	H	Clock output	Clock output	Clock output
$\overline{RD}$ , $\overline{AS}$ , HWR, LWR	1–6	H	T	T	H	T	$\overline{RD}$ , $\overline{AS}$ , HWR, LWR
	7	H	T	T	H	H	—
P1 <sub>7</sub> –P1 <sub>0</sub>	1–6	T	T	T	T	T	D <sub>15</sub> –D <sub>8</sub>
	7			keep	keep	keep	I/O port
P2 <sub>7</sub> –P2 <sub>0</sub>	1, 3–5, 6	T	T	T	T	T	D <sub>7</sub> –D <sub>0</sub>
	2, 7			keep	keep	keep	I/O port
P3 <sub>5</sub> –P3 <sub>0</sub> P4 <sub>7</sub> –P4 <sub>0</sub> P5 <sub>7</sub> –P5 <sub>0</sub> P6 <sub>4</sub> –P6 <sub>0</sub> P7 <sub>7</sub> –P7 <sub>0</sub>	1–7	T	T	keep* <sup>1</sup>	keep	keep	I/O port
P8 <sub>4</sub> –P8 <sub>0</sub> P9 <sub>7</sub> –P9 <sub>0</sub>	1–7	T	T	T	T	T	Input port
PA <sub>6</sub> –PA <sub>4</sub>	1–7	T	T	keep* <sup>2</sup>	keep* <sup>3</sup>	keep* <sup>4</sup>	I/O port or control input/output
PA <sub>3</sub> –PA <sub>0</sub>	3, 5	L	T	T	L	T	A <sub>19</sub> –A <sub>16</sub>
	1, 2, 4, 6, 7	T		keep* <sup>1</sup>	keep	keep	I/O port
PB <sub>7</sub> –PB <sub>0</sub> PC <sub>7</sub> –PC <sub>0</sub>	1, 3, 5, 6	L	T	T	L	T	A <sub>15</sub> –A <sub>0</sub>
	2, 4, 7	T		keep	keep	keep	I/O port

### Legend

H: High, L: Low, T: High-impedance state

keep: Input pins are in the high-impedance state; output pins maintain their previous state.

- Notes:
1. The on-chip supporting modules are reset, so these pins become input or output pins according to their DDR and DR bits.
  2. If PA<sub>5</sub> is set for  $\overline{BACK}$  output, it goes to the high-impedance state.
  3. BREQ can be received, and BACK is high.



4.  $\overline{\text{BACK}}$  is low.
5. In modes 5 and 6, the external bus space has a 16-bit bus width, but an 8-bit bus width is set after a reset. In this case, the upper half of the data bus (D15 to D8) is enabled, and the lower half (D7 to D0) is disabled. After the BCRE bit in the bus control register (BCR) has been set to 1 by software, the bus width can be changed to 16 bits (D15 to D0) by a byte area top register (ARBT) setting. In modes 1, 3, and 4, the external bus space has a 16-bit bus width (D15 to D0) after a reset, but this can be changed to 8 bits by an ARBT setting. In this case, the upper half of the data bus (D15 to D8) is enabled, and the lower half (D7 to D0) is disabled. For details of the settings, see section 16, Bus Controller.

## G.2 Pin States at Reset

(1) **Modes 1 and 6:** Figure G-1 is a timing diagram for the case in which  $\overline{\text{RES}}$  goes low during three-state access in mode 1 or 6. As soon as  $\overline{\text{RES}}$  goes low, all ports are initialized to the input state.  $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{LWR}}$ , and  $\overline{\text{HWR}}$  go high, and  $\text{D}_{15}$  to  $\text{D}_0$  go to the high-impedance state.  $\text{A}_{15}$  to  $\text{A}_0$  are initialized to the low state 1.5 system clock cycles ( $1.5\phi$ ) after the low level of  $\overline{\text{RES}}$  is sampled.

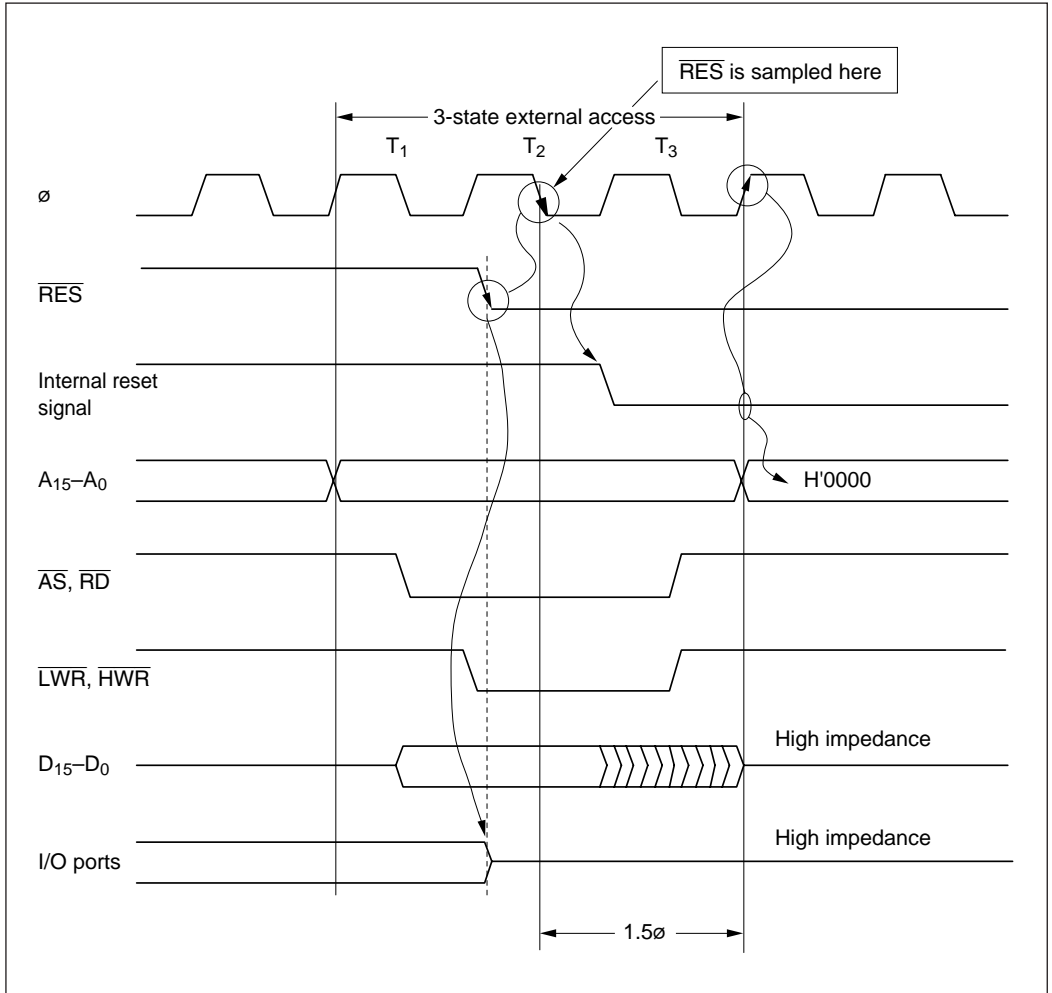
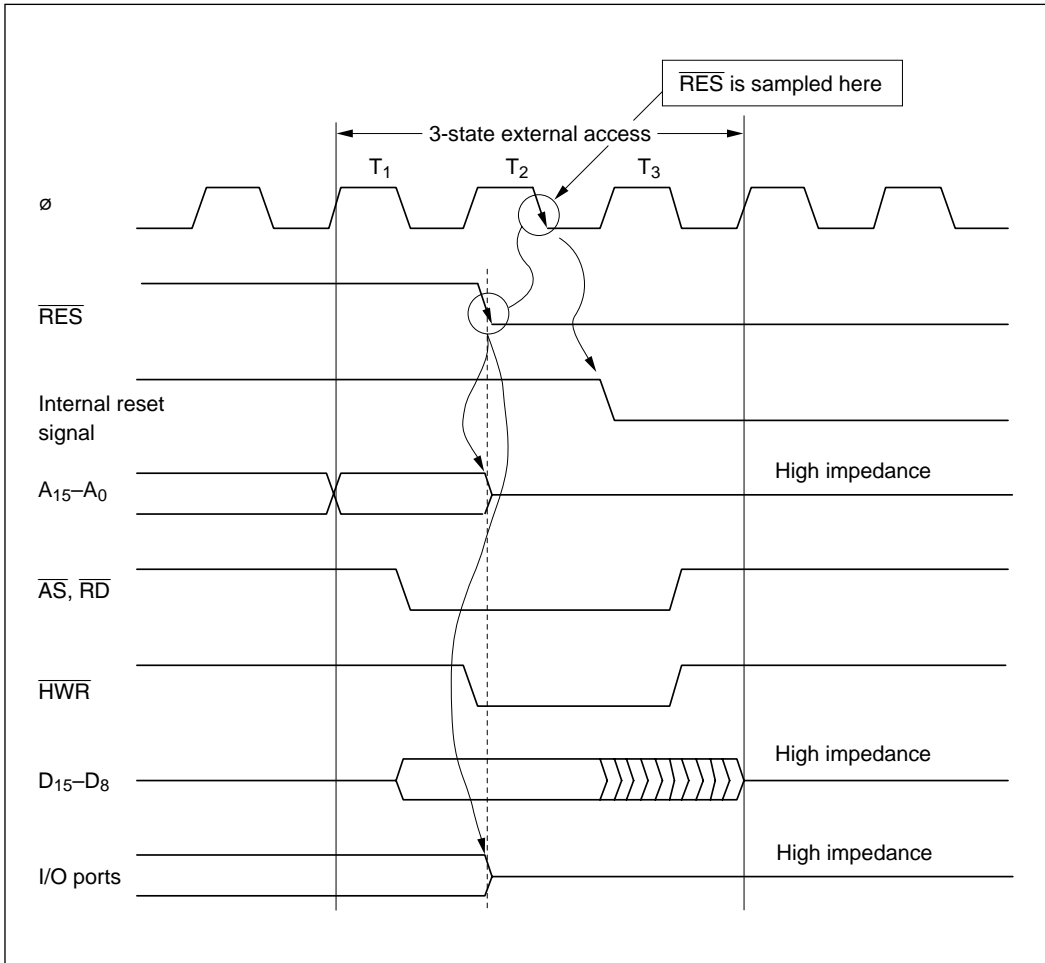


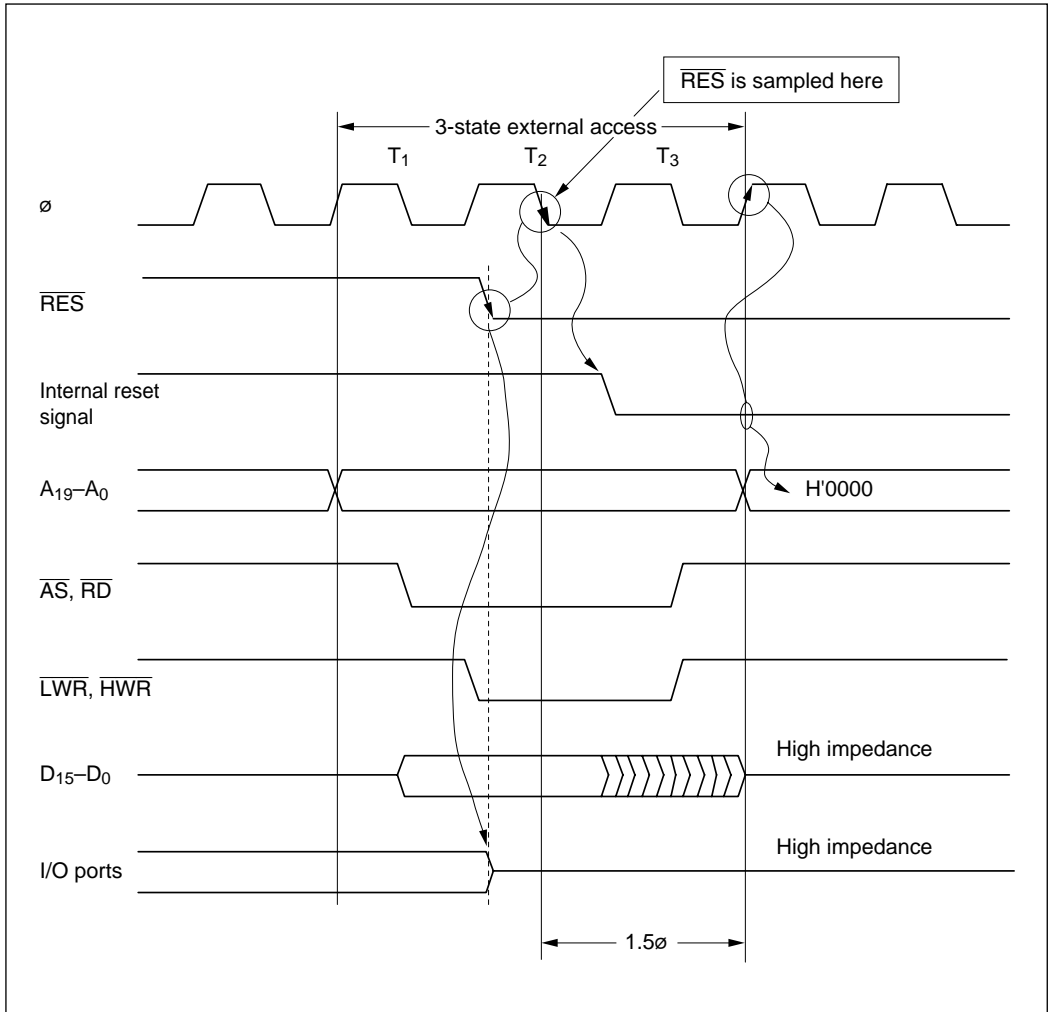
Figure G-1 Reset during Three-State Access (Modes 1 and 6)

(2) **Mode 2:** Figure G-2 is a timing diagram for the case in which  $\overline{\text{RES}}$  goes low during three-state access in mode 2. As soon as  $\overline{\text{RES}}$  goes low, all ports are initialized to the input state.  $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{LWR}}$ , and  $\overline{\text{HWR}}$  go high, and  $\text{D}_{15}$  to  $\text{D}_8$  go to the high-impedance state.  $\text{A}_{15}$  to  $\text{A}_0$  are initialized as soon as  $\overline{\text{RES}}$  goes low, and become input ports.



**Figure G-2 Reset during Three-State Access (Mode 2)**

**(3) Modes 3 and 5:** Figure G-3 is a timing diagram for the case in which  $\overline{\text{RES}}$  goes low during three-state access in mode 3 or 5. As soon as  $\overline{\text{RES}}$  goes low, all ports are initialized to the input state.  $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{LWR}}$ , and  $\overline{\text{HWR}}$  go high, and  $\text{D}_{15}$  to  $\text{D}_0$  go to the high-impedance state.  $\text{A}_{19}$  to  $\text{A}_0$  are initialized to the low state 1.5 system clock cycles ( $1.5\phi$ ) after the low level of  $\overline{\text{RES}}$  is sampled.



**Figure G-3 Reset during Three-State Access (Modes 3 and 5)**

(4) **Mode 4:** Figure G-4 is a timing diagram for the case in which  $\overline{\text{RES}}$  goes low during three-state access in mode 4. As soon as  $\overline{\text{RES}}$  goes low, all ports are initialized to the input state.  $\overline{\text{AS}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{LWR}}$ , and  $\overline{\text{HWR}}$  go high, and  $\text{D}_{15}$  to  $\text{D}_0$  go to the high-impedance state.  $\text{A}_{19}$  to  $\text{A}_0$  are initialized as soon as  $\overline{\text{RES}}$  goes low, and become input ports.

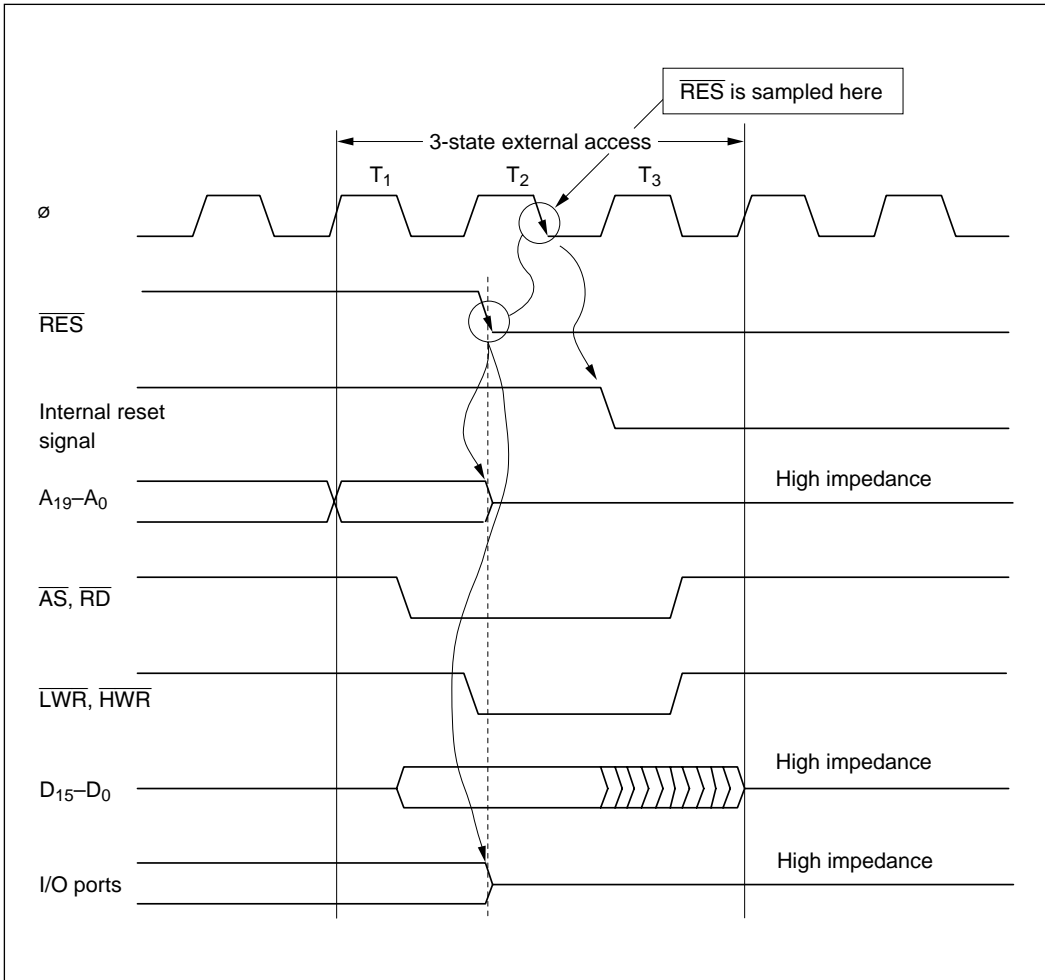
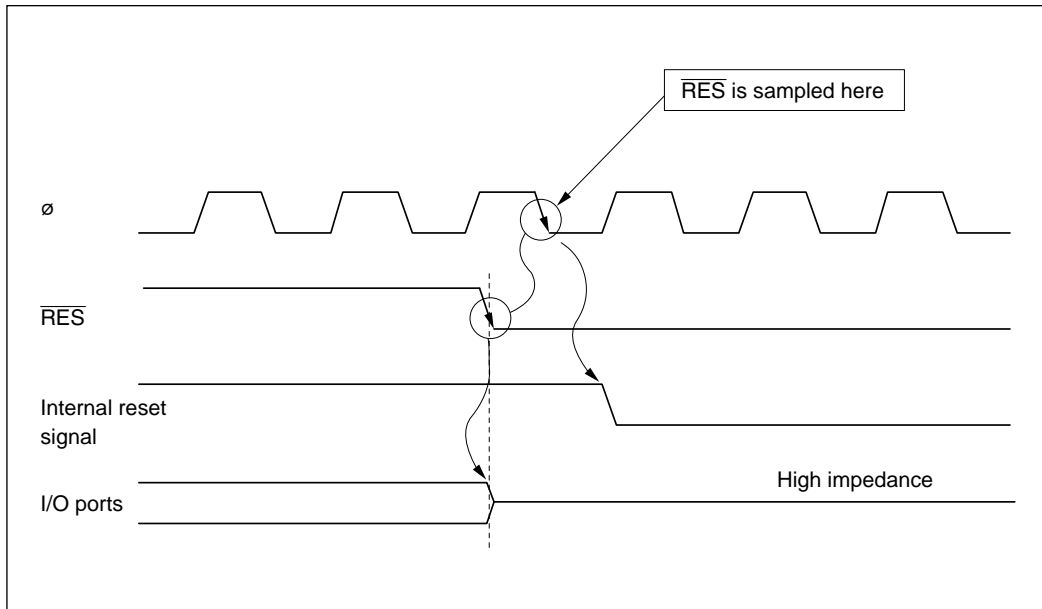


Figure G-4 Reset during Three-State Access (Mode 4)

(5) **Mode 7:** Figure G-5 is a timing diagram for the case in which  $\overline{\text{RES}}$  goes low in mode 7. As soon as  $\overline{\text{RES}}$  goes low, all ports are initialized to the input state.

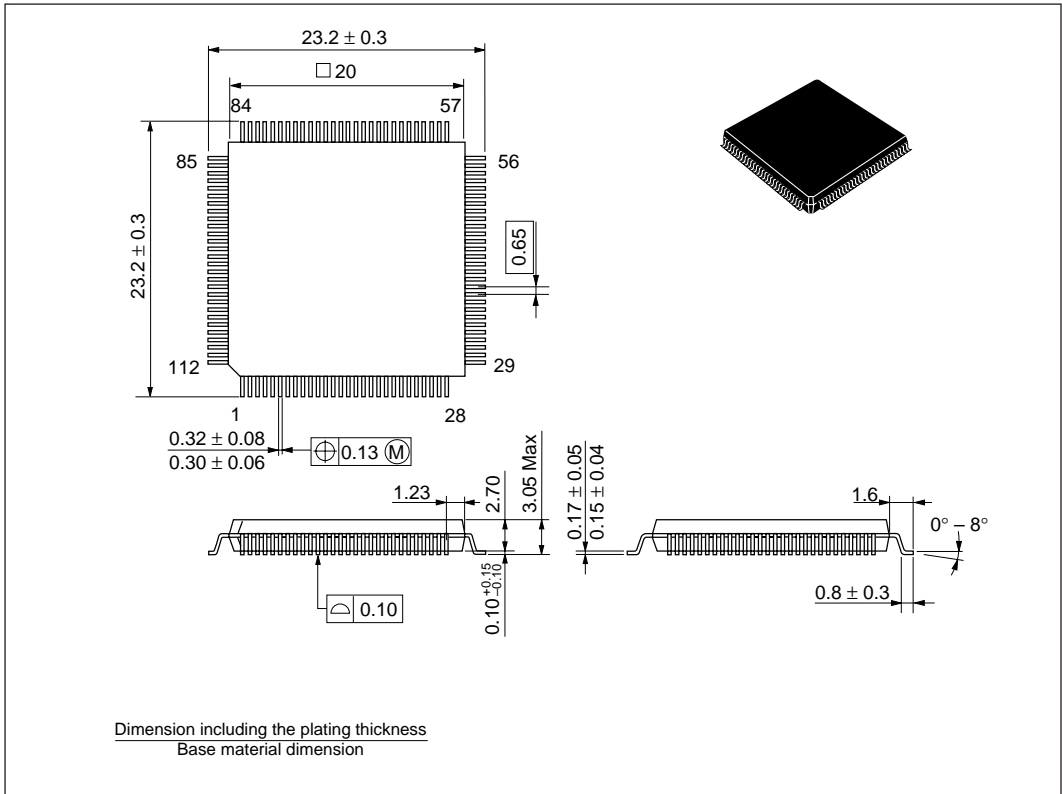


**Figure G-5 Resetting of I/O Ports (Mode 7)**

# Appendix H Package Dimensions

Figure H-1 shows the FP-112 package dimensions of the H8/539F.

Unit: mm



**Figure H-1 Package Dimensions (FP-112)**

---

## H8/539F Hardware Manual

Publication Date: 1st Edition, January 1997  
2nd Edition, March 1998

Published by: Electronic Devices Business Group  
Hitachi, Ltd.

Edited by: Technical Documentation Center  
Hitachi, Microcomputer System Ltd.

Copyright © Hitachi, Ltd., 1997. All rights reserved. Printed in Japan.