## To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003.
These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.)

Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp.

Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

RENESAS

Renesas Technology Corp.

Downloaded from Arrow.com.

# Hitachi Single-Chip Microcomputer

# H8/325 Series

# H8/3257, H8/3256

# H8/325, H8/324,

# H8/323, H8/322

# Hardware Manual

# Preface

The H8/325 Series is a family of high-performance single-chip microcomputers ideally suited for embedded control of industrial equipment. The chips are built around an H8/300 CPU core: a high-speed processor. On-chip supporting modules provide ROM, RAM, two types of timers, I/O ports, and a serial communication interface for easy implementation of compact, high-speed control systems.

The H8/325 Series offers a selection of on-chip memory.
H8/3257: 60-kbyte ROM; 2-kbyte RAM
H8/3256: 48-kbyte ROM; 2-kbyte RAM
H8/325: 32-kbyte ROM; 1-kbyte RAM
H8/324: 24-kbyte ROM; 1-kbyte RAM
H8/323: 16-kbyte ROM; 512-byte RAM
H8/322: 8-kbyte ROM; 256-byte RAM

The H8/3257, H8/3256, H8/325, H8/323, and H8/322 chips are available with either electrically programmable or mask-programmable ROM. Manufacturers can use the electrically programmable ZTAT™ (Zero Turn-Around Time*) version to get production off to a fast start and make software changes quickly, then switch over to the masked version for full-scale production runs.

This manual describes the H8/325 Series hardware. Refer to the *H8/300 Series Programming Manual* for a detailed description of the instruction set.

* ZTAT is a registered trademark of Hitachi, Ltd.

# CONTENTS


Section 1. Overview ..... 1
1.1 Overview ..... 1
1.2 Block Diagram ..... 5
1.3 Pin Assignments and Functions ..... 6
1.3.1 Pin Arrangement ..... 6
1.3.2 Pin Functions ..... 8

Section 2. MCU Operating Modes and Address Space ..... 15
2.1 Overview ..... 15
2.2 Mode Descriptions ..... 16
2.3 Address Space Map ..... 16
2.3.1 Access Speed ..... 16
2.3.2 $\overline{\text{IOS}}$ ..... 17
2.4 Mode and System Control Registers (MDCR and SYSCR) ..... 24
2.4.1 Mode Control Register (MDCR) – H'FFC5 ..... 24
2.4.2 System Control Register (SYSCR) – H'FFC4 ..... 25

Section 3. CPU ..... 27
3.1 Overview ..... 27
3.1.1 Features ..... 27
3.2 Register Configuration ..... 28
3.2.1 General Registers ..... 28
3.2.2 Control Registers ..... 29
3.2.3 Initial Register Values ..... 30
3.3 Addressing Modes ..... 31
3.4 Data Formats ..... 33
3.4.1 Data Formats in General Registers ..... 34
3.4.2 Memory Data Formats ..... 35
3.5 Instruction Set ..... 36
3.5.1 Data Transfer Instructions ..... 38
3.5.2 Arithmetic Operations ..... 40
3.5.3 Logic Operations ..... 41
3.5.4 Shift Operations ..... 41
3.5.5 Bit Manipulations ..... 43
3.5.6 Branching Instructions ..... 49
3.5.7 System Control Instructions ..... 51

3.5.8 Block Data Transfer Instruction ..... 52
3.6 CPU States ..... 54
3.6.1 Program Execution State ..... 55
3.6.2 Exception-Handling State ..... 55
3.6.3 Power-Down State ..... 56
3.7 Access Timing and Bus Cycle ..... 56
3.7.1 Access to On-Chip Memory (RAM and ROM) ..... 56
3.7.2 Access to On-Chip Register Field and External Devices ..... 58

Section 4. Exception Handling ..... 61
4.1 Overview ..... 61
4.2 Reset ..... 61
4.2.1 Overview ..... 61
4.2.2 Reset Sequence ..... 61
4.2.3 Disabling of Interrupts after Reset ..... 64
4.3 Interrupts ..... 64
4.3.1 Overview ..... 64
4.3.2 Interrupt-Related Registers ..... 65
4.3.3 External Interrupts ..... 68
4.3.4 Internal Interrupts ..... 69
4.3.5 Interrupt Handling ..... 70
4.3.6 Interrupt Response Time ..... 75
4.4 Note on Stack Handling ..... 75

Section 5. I/O Ports ..... 77
5.1 Overview ..... 77
5.2 Port 1 ..... 78
5.3 Port 2 ..... 81
5.4 Port 3 ..... 84
5.5 Port 4 ..... 87
5.6 Port 5 ..... 94
5.7 Port 6 ..... 99
5.8 Port 7 ..... 104

Section 6. Parallel Handshaking Interface ..... 113
6.1 Overview ..... 113
6.1.1 Features ..... 113
6.1.2 Block Diagram ..... 114
6.1.3 Input and Output Pins ..... 115
6.1.4 Register Configuration ..... 115
6.2 Register Descriptions ..... 115
6.2.1 Port 3 Data Direction Register (P3DDR) ..... 115
6.2.2 Port 3 Data Register (P3DR) ..... 116
6.2.3 Handshake Control/Status Register (HCSR) ..... 116
6.3 Operation ..... 118
6.3.1 Output Timing of Output Strobe Signal ..... 118
6.3.2 Busy Signal Output Timing ..... 119
6.3.3 Operation in Software Standby Mode ..... 119
6.3.4 Sample Application ..... 120
6.3.5 Interrupts ..... 121

Section 7. 16-Bit Free-Running Timer ..... 123
7.1 Overview ..... 123
7.1.1 Features ..... 123
7.1.2 Block Diagram ..... 123
7.1.3 Input and Output Pins ..... 125
7.1.4 Register Configuration ..... 125
7.2 Register Descriptions ..... 126
7.2.1 Free-Running Counter (FRC) – H'FF92 ..... 126
7.2.2 Output Compare Registers A and B (OCRA and OCRB) – H'FF94 and H'FF96 ..... 126
7.2.3 Input Capture Register (ICR) – H'FF98 ..... 127
7.2.4 Timer Control Register (TCR) – H'FF90 ..... 128
7.2.5 Timer Control/Status Register (TCSR) – H'FF91 ..... 130
7.2.6 FRT Noise Canceler Control Register (FNCR) – H'FFFF ..... 133
7.3 CPU Interface ..... 133
7.4 Operation ..... 136
7.4.1 FRC Incrementation Timing ..... 136
7.4.2 Output Compare Timing ..... 138
7.4.3 FRC Clear Timing ..... 138
7.4.4 Input Capture Timing ..... 139
7.4.5 Timing of Input Capture Flag (ICF) Setting ..... 140

7.4.6 Setting of FRC Overflow Flag (OVF) ..... 141
7.5 Interrupts ..... 142
7.6 Noise Canceler ..... 142
7.7 Sample Application ..... 144
7.8 Application Notes ..... 145

Section 8. 8-Bit Timers ..... 151
8.1 Overview ..... 151
8.1.1 Features ..... 151
8.1.2 Block Diagram ..... 151
8.1.3 Input and Output Pins ..... 152
8.1.4 Register Configuration ..... 153
8.2 Register Descriptions ..... 153
8.2.1 Timer Counter (TCNT) – H'FFC8 (TMR0), H'FFD0 (TMR1) ..... 153
8.2.2 Time Constant Registers A and B (TCORA and TCORB) – H'FFCA and H'FFCB (TMR0), H'FFD2 and H'FFD3 (TMR1) ..... 154
8.2.3 Timer Control Register (TCR) – H'FFC8 (TMR0), H'FFD0 (TMR1) ..... 154
8.2.4 Timer Control/Status Register (TCSR) – H'FFC9 (TMR0), H'FFD1 (TMR1) ..... 156
8.3 Operation ..... 158
8.3.1 TCNT Incrementation Timing ..... 158
8.3.2 Compare Match Timing ..... 159
8.3.3 External Reset of TCNT ..... 161
8.3.4 Setting of TCSR Overflow Flag ..... 162
8.4 Interrupts ..... 163
8.5 Sample Application ..... 163
8.6 Application Notes ..... 164

Section 9. Serial Communication Interface ..... 169
9.1 Overview ..... 169
9.1.1 Features ..... 169
9.1.2 Block Diagram ..... 170
9.1.3 Input and Output Pins ..... 170
9.1.4 Register Configuration ..... 171
9.2 Register Descriptions ..... 171
9.2.1 Receive Shift Register (RSR) ..... 171
9.2.2 Receive Data Register (RDR) – H'FFDD ..... 172
9.2.3 Transmit Shift Register (TSR) ..... 172
9.2.4 Transmit Data Register (TDR) – H'FFDB ..... 172

9.2.5 Serial Mode Register (SMR) – H'FFD8 ..... 173
9.2.6 Serial Control Register (SCR) – H'FFDA ..... 175
9.2.7 Serial Status Register (SSR) – H'FFDC ..... 177
9.2.8 Bit Rate Register (BRR) – H'FFD9 ..... 179
9.3 Operation ..... 183
9.3.1 Overview ..... 183
9.3.2 Asynchronous Mode ..... 184
9.3.3 Synchronous Mode ..... 188
9.4 Interrupts ..... 192
9.5 Application Notes ..... 193

Section 10. RAM ..... 197
10.1 Overview ..... 197
10.2 Block Diagram ..... 197
10.3 RAM Enable Bit (RAME) ..... 198
10.4 Operation ..... 198
10.4.1 Expanded Modes (Modes 1 and 2) ..... 198
10.4.2 Single-Chip Mode (Mode 3) ..... 199

Section 11. ROM ..... 201
11.1 Overview ..... 201
11.1.1 Block Diagram ..... 202
11.2 PROM Mode ..... 202
11.2.1 PROM Mode Setup ..... 202
11.2.2 Socket Adapter Pin Assignments and Memory Map ..... 203
11.3 Programming ..... 208
11.3.1 Selection of Sub-Modes in PROM Mode ..... 208
11.3.2 Writing and Verifying ..... 209
11.3.3 Notes on Writing ..... 215
11.3.4 Reliability of Written Data ..... 215
11.3.5 Erasing of Data ..... 216
11.4 Handling of Windowed Packages ..... 216

Section 12. Power-Down State ..... 219
12.1 Overview ..... 219
12.2 System Control Register: Power-Down Control Bits ..... 220
12.3 Sleep Mode ..... 221
12.3.1 Transition to Sleep Mode ..... 222

12.3.2 Exit from Sleep Mode ..... 222
12.4 Software Standby Mode ..... 222
12.4.1 Transition to Software Standby Mode ..... 223
12.4.2 Exit from Software Standby Mode ..... 223
12.4.3 Sample Application of Software Standby Mode ..... 223
12.4.4 Notes on Current Dissipation ..... 224
12.5 Hardware Standby Mode ..... 225
12.5.1 Transition to Hardware Standby Mode ..... 225
12.5.2 Recovery from Hardware Standby Mode ..... 226
12.5.3 Timing Relationships ..... 226

Section 13. E-Clock Interface ..... 227
13.1 Overview ..... 227

Section 14. Clock Pulse Generator ..... 231
14.1 Overview ..... 231
14.1.1 Block Diagram ..... 231
14.2 Oscillator Circuit ..... 231
14.3 System Clock Divider ..... 234

Section 15. Electrical Specifications ..... 235
15.1 Absolute Maximum Ratings ..... 235
15.2 Electrical Characteristics ..... 235
15.2.1 DC Characteristics ..... 235
15.2.2 AC Characteristics ..... 242
15.3 MCU Operational Timing ..... 246
15.3.1 Bus Timing ..... 246
15.3.2 Control Signal Timing ..... 248
15.3.3 16-Bit Free-Running Timer Timing ..... 251
15.3.4 8-Bit Timer Timing ..... 252
15.3.5 Serial Communication Interface Timing ..... 253
15.3.6 I/O Port Timing ..... 254
15.3.7 Parallel Handshake Interface Timing ..... 254

Appendices

Appendix A. CPU Instruction Set ..... 257
A.1 Instruction Set List ..... 257
A.2 Operation Code Map ..... 264
A.3 Number of States Required for Execution ..... 266

Appendix B. Register Field ..... 272
B.1 Register Addresses and Bit Names ..... 272
B.2 Register Descriptions ..... 276

Appendix C. Pin States ..... 301
C.1 Pin States in Each Mode ..... 301

Appendix D. Timing of Transition to and Recovery from Hardware Standby Mode ..... 303

Appendix E. Package Dimensions ..... 304

# Section 1. Overview

## 1.1 Overview

The H8/325 Series is a series of single-chip microcomputers integrating a CPU core together with a variety of peripheral functions needed in control systems.

The H8/300 CPU is a high-speed processor featuring powerful bit-manipulation instructions, ideally suited for realtime control applications. The on-chip supporting modules include ROM, RAM, two types of timers (16-bit free-running timer and 8-bit timer), a serial communication interface, I/O ports, and a parallel handshaking interface. The on-chip memory sizes of the three chips in the H8/325 Series are:

H8/3257: 60-kbyte ROM; 2-kbyte RAM
H8/3256: 48-kbyte ROM; 2-kbyte RAM
H8/325: 32-kbyte ROM; 1-kbyte RAM
H8/324: 24-kbyte ROM; 1-kbyte RAM
H8/323: 16-kbyte ROM; 512-byte RAM
H8/322: 8-kbyte ROM; 256-byte RAM

The H8/325 Series can operate in single-chip mode or in two expanded modes, depending on the memory requirements of the application. The operating mode is referred to in this manual as the MCU mode (MCU: MicroComputer Unit).

The H8/3257, H8/3256, H8/325, H8/323, and H8/322 are available in a masked ROM version, or a ZTAT™* version with electrically programmable ROM that can be programmed at the user site.

* ZTAT is a registered trademark of Hitachi, Ltd.

Table 1-1 lists the features of the H8/325 Series.

**Table 1-1. Features**

| Feature | Description |
|---|---|
| CPU | General register architecture<br>• Eight 16-bit general registers, or<br>• Sixteen 8-bit general registers<br>High speed<br>• Maximum clock rate: 10 MHz<br>• Add/subtract: 0.2 µs<br>• Multiply/divide: 1.4 µs<br>Concise, streamlined instruction set<br>• All instructions are 2 or 4 bytes long<br>• Register-register arithmetic and logic operations<br>• Register-memory data transfer by MOV instruction<br>Instruction set features<br>• Multiply instruction (8 bits × 8 bits)<br>• Divide instruction (16 bits ÷ 8 bits)<br>• Bit-accumulator instructions<br>• Register-indirect specification of bit positions |
| Memory | H8/3257<br>• ROM: 60 kbytes<br>• RAM: 2 kbytes<br>H8/3256<br>• ROM: 48 kbytes<br>• RAM: 2 kbytes<br>H8/325<br>• ROM: 32 kbytes<br>• RAM: 1 kbyte<br>H8/324<br>• ROM: 24 kbytes<br>• RAM: 1 kbyte<br>H8/323<br>• ROM: 16 kbytes<br>• RAM: 512 bytes<br>H8/322<br>• ROM: 8 kbytes<br>• RAM: 256 bytes |
| 16-Bit free-running timer module (FRT: 1 channel) | • One 16-bit free-running counter (also usable for external event counting)<br>• Two compare outputs<br>• One capture input |
| 8-Bit timer module (2 channels) | Each channel has:<br>• One 8-bit up-counter (also usable for external event counting)<br>• Two time constant registers |

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

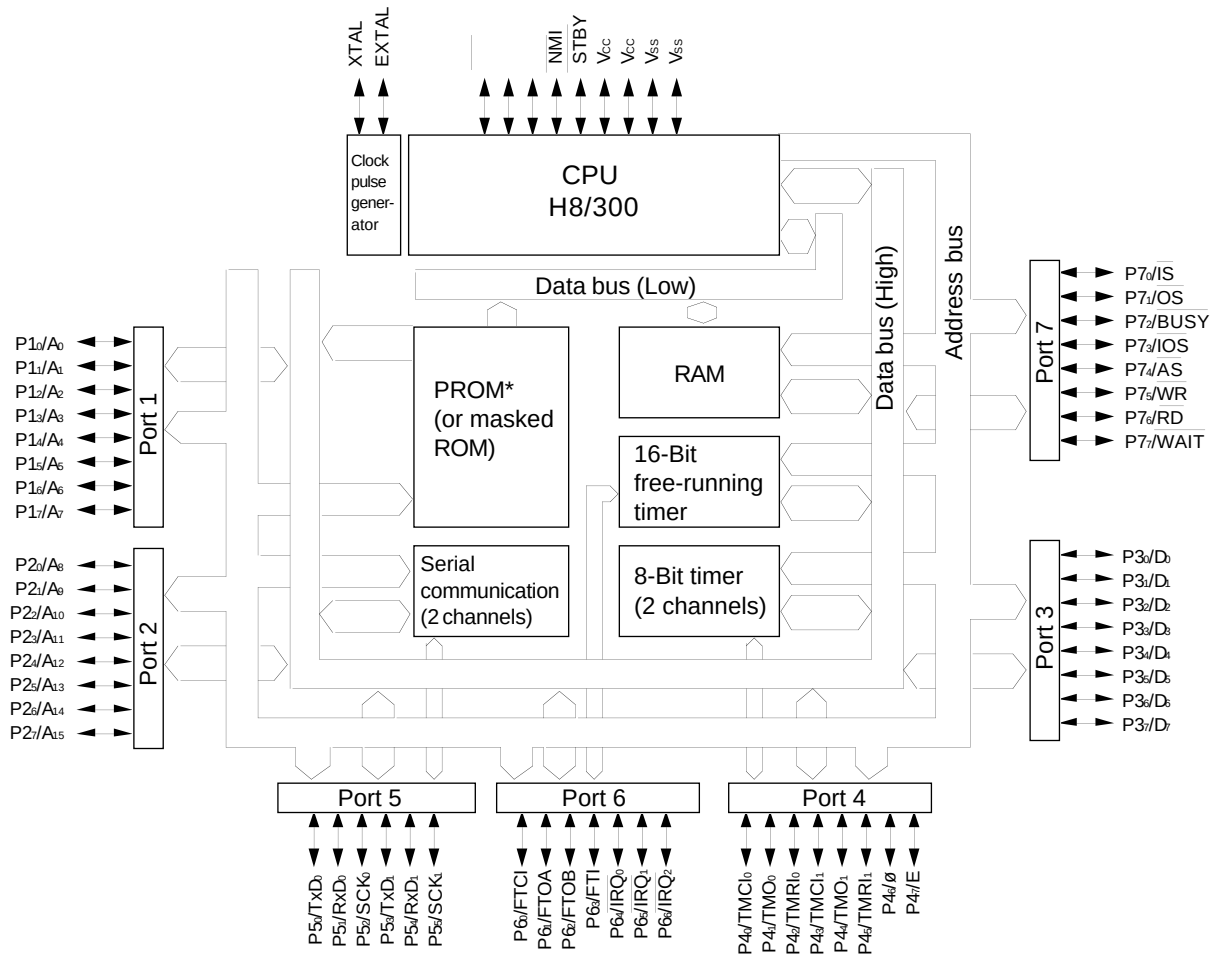
| Feature | Description | | | |
|---|---|---|---|---|
| Serial communication interface (SCI: 2 channels) | • Selection of asynchronous and synchronous modes<br>• Simultaneous transmit and receive (full duplex operation)<br>• On-chip baud rate generator | | | |
| I/O ports | • 53 input/output pins (of which 16 can drive large current loads)<br>• All input pins have programmable input pull-ups | | | |
| Parallel handshaking interface | • Built-in parallel handshaking is available at port 3 | | | |
| Interrupts | • Four external interrupt pins: $\overline{\text{NMI}}$, $\overline{\text{IRQ}_0}$ to $\overline{\text{IRQ}_2}$<br>• Seventeen on-chip interrupt sources | | | |
| Operating modes | • Mode 1: expanded mode with on-chip ROM disabled<br>• Mode 2: expanded mode with on-chip ROM enabled<br>• Mode 3: single-chip mode | | | |
| Power-down state | • Sleep mode<br>• Software standby mode<br>• Hardware standby mode | | | |
| Other features | • On-chip clock oscillator<br>• E clock output | | | |
| Product lineup | Type code (5V series) | Type code (3V series) | Package | ROM |
| | HD6473257C | HD6473257VC | 64-Pin windowed shrink DIP (DC-64S) | PROM |
| | HD6473257P | HD6473257VP | 64-Pin shrink DIP (DP-64S) | |
| | HD6473257F | HD6473257VF | 64-Pin QFP (FP-64A) | |
| | HD6473257CP | HD6473257VCP | 68-Pin PLCC (CP-68) | |
| | HD6433257P | HD6433257VP | 64-Pin shrink DIP (DP-64S) | Masked |
| | HD6433257F | HD6433257VF | 64-Pin QFP (FP-64A) | ROM |
| | HD6433257CP | HD6433257VCP | 68-Pin PLCC (CP-68) | |
| | HD6473256P | HD6473256VP | 64-Pin shrink DIP (DP-64S) | PROM |
| | HD6473256F | HD6473256VF | 64-Pin QFP (FP-64A) | |
| | HD6473256CP | HD6473256VCP | 68-Pin PLCC (CP-68) | |
| | HD6433256P | HD6433256VP | 64-Pin shrink DIP (DP-64S) | Masked |
| | HD6433256F | HD6433256VF | 64-Pin QFP (FP-64A) | ROM |
| | HD6433256CP | HD6433256VCP | 68-Pin PLCC (CP-68) | |

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

| Feature | Description | | | |
|---|---|---|---|---|
| Product lineup (cont.) | Type code (5V series) | Type code (3V series) | Package | ROM |
| | HD6473258C | | 64-Pin windowed shrink DIP (DC-64S) | PROM |
| | HD6473258P | | 64-Pin shrink DIP (DP-64S) | |
| | HD6473258F | | 64-Pin QFP (FP-64A) | |
| | HD6473258CP | | 68-Pin PLCC (CP-68) | |
| | HD6433258P | | 64-Pin shrink DIP (DP-64S) | Masked |
| | HD6433258F | | 64-Pin QFP (FP-64A) | ROM |
| | HD6433258CP | | 68-Pin PLCC (CP-68) | |
| | HD6413258P | | 64-Pin shrink DIP (DP-64S) | No |
| | HD6413258F | | 64-Pin QFP (FP-64A) | ROM |
| | HD6413258CP | | 68-Pin PLCC (CP-68) | |
| | HD6433248P | | 64-Pin shrink DIP (DP-64S) | Masked |
| | HD6433248F | | 64-Pin QFP (FP-64A) | ROM |
| | HD6433248CP | | 68-Pin PLCC (CP-68) | |
| | HD6473238P | | 64-Pin shrink DIP (DP-64S) | PROM |
| | HD6473238F | | 64-Pin QFP (FP-64A) | |
| | HD6473238CP | | 68-Pin PLCC (CP-68) | |
| | HD6433238P | | 64-Pin shrink DIP (DP-64S) | Masked |
| | HD6433238F | | 64-Pin QFP (FP-64A) | ROM |
| | HD6433238CP | | 68-Pin PLCC (CP-68) | |
| | HD6413238P | | 64-Pin shrink DIP (DP-64S) | No |
| | HD6413238F | | 64-Pin QFP (FP-64A) | ROM |
| | HD6413238CP | | 68-Pin PLCC (CP-68) | |
| | HD6473228P | | 64-Pin shrink DIP (DP-64S) | PROM |
| | HD6473228F | | 64-Pin QFP (FP-64A) | |
| | HD6473228CP | | 68-Pin PLCC (CP-68) | |
| | HD6433228P | | 64-Pin shrink DIP (DP-64S) | Masked |
| | HD6433228F | | 64-Pin QFP (FP-64A) | ROM |
| | HD6433228CP | | 68-Pin PLCC (CP-68) | |

## 1.2 Block Diagram

Figure 1-1 shows a block diagram of the H8/325 Series.




**Memory siz**

| | H8/3257 | H8/3256 | H8/325 | H8/324 | H8/323 | H8/322 |
|---|---|---|---|---|---|---|
| ROM | 60 kbytes | 48 kbytes | 32 kbytes | 24 kbytes | 16 kbytes | 8 kbytes |
| RAM | 2 kbytes | 2 kbytes | 1 kbyte | 1 kbyte | 512 bytes | 256 bytes |

* H8/3257, H8/3256, H8/325, H8/323, and H8/322 are available with PROM.

**Figure 1-1. Block Diagram**

## 1.3 Pin Assignments and Functions

### 1.3.1 Pin Arrangement

Figure 1-2 shows the pin arrangement of the H8/325 Series in the DC-64S and DP-64S packages. Figure 1-3 shows the pin arrangement in the FP-64A package. Figure 1-4 shows the pin arrangement in the CP-68 package.

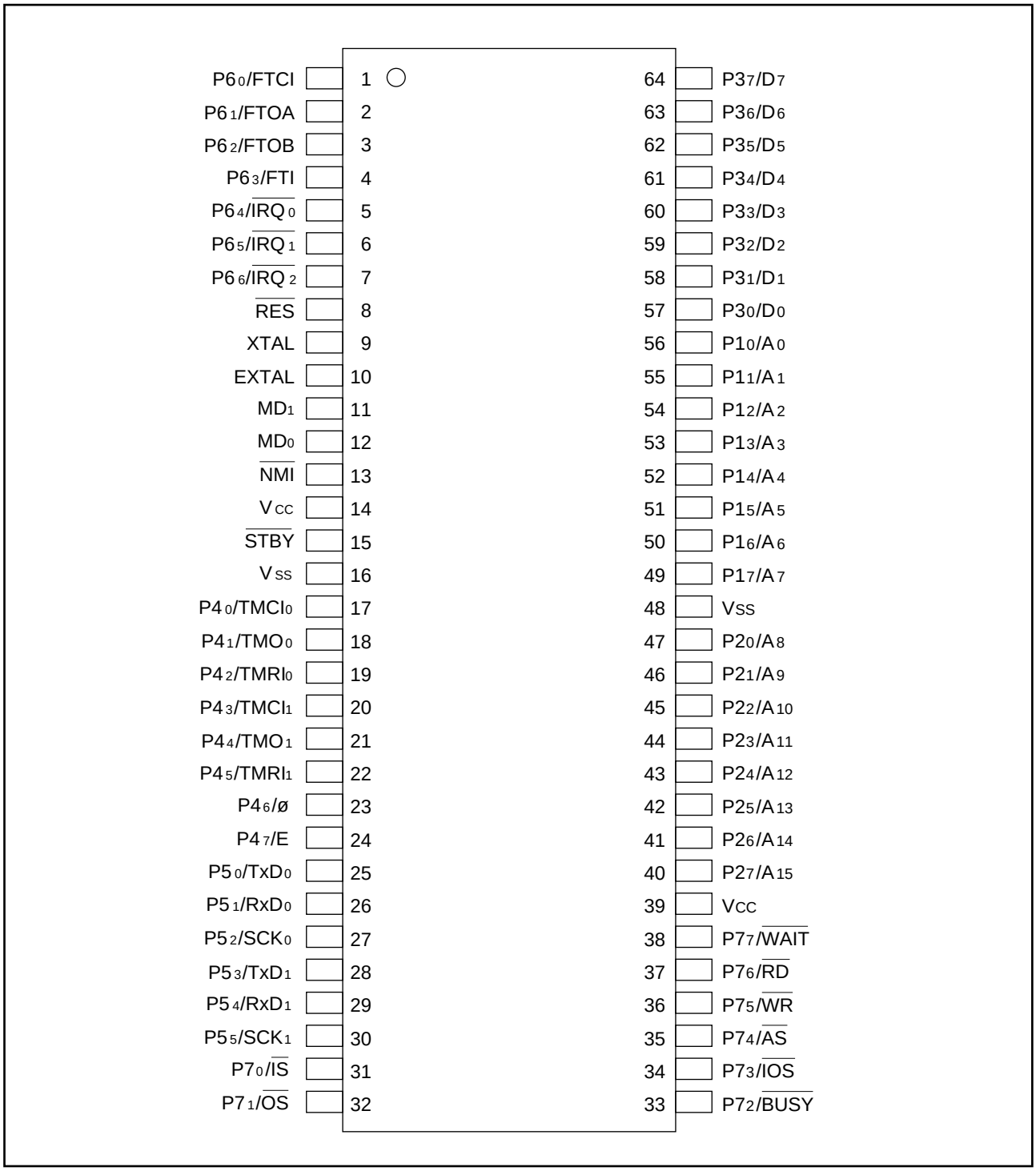



**Figure 1-2. Pin Arrangement (DC-64S, DP-64S, Top View)**

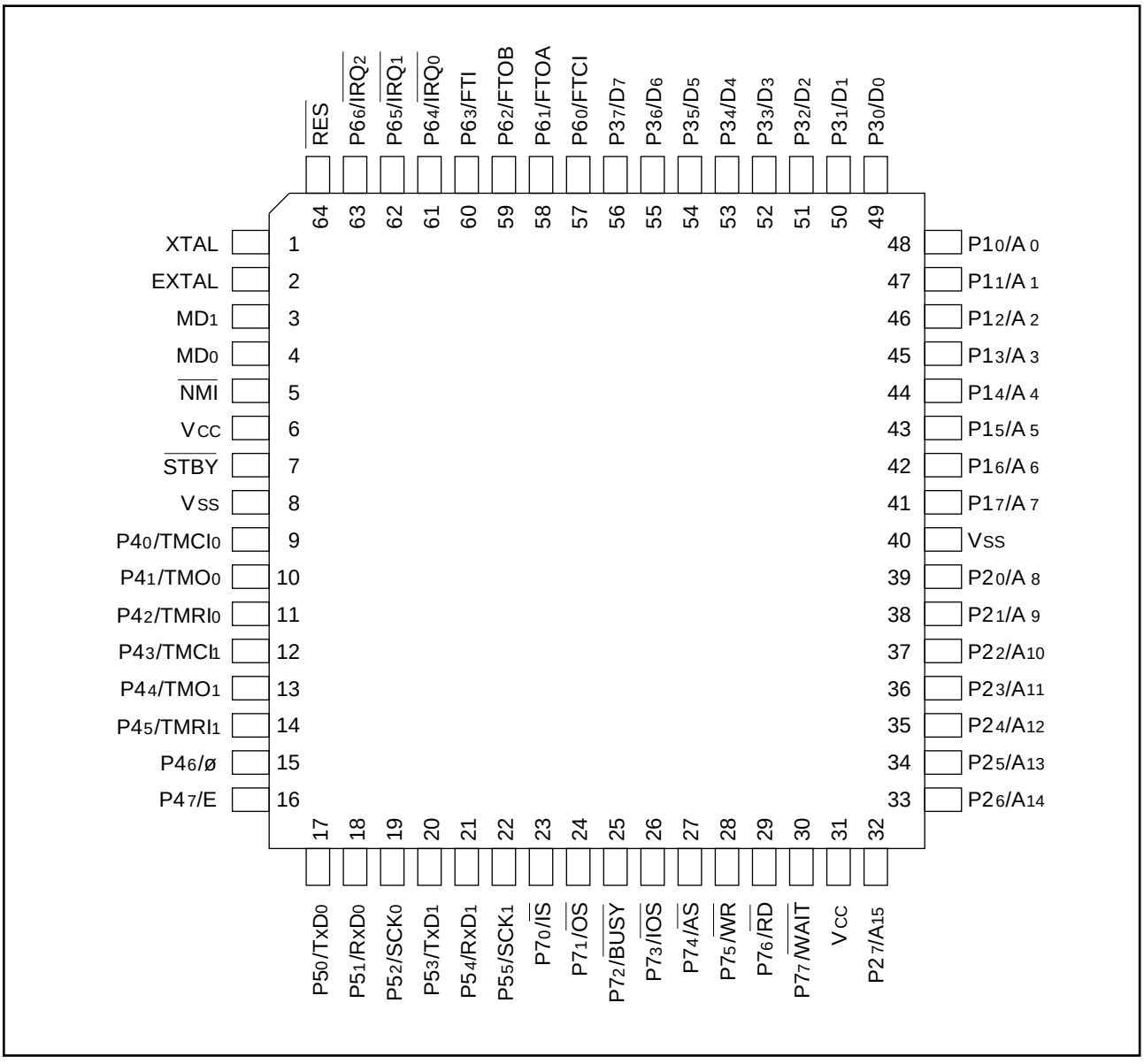



**Figure 1-3. Pin Arrangement (FP-64A, Top View)**

• **PLCC-68**

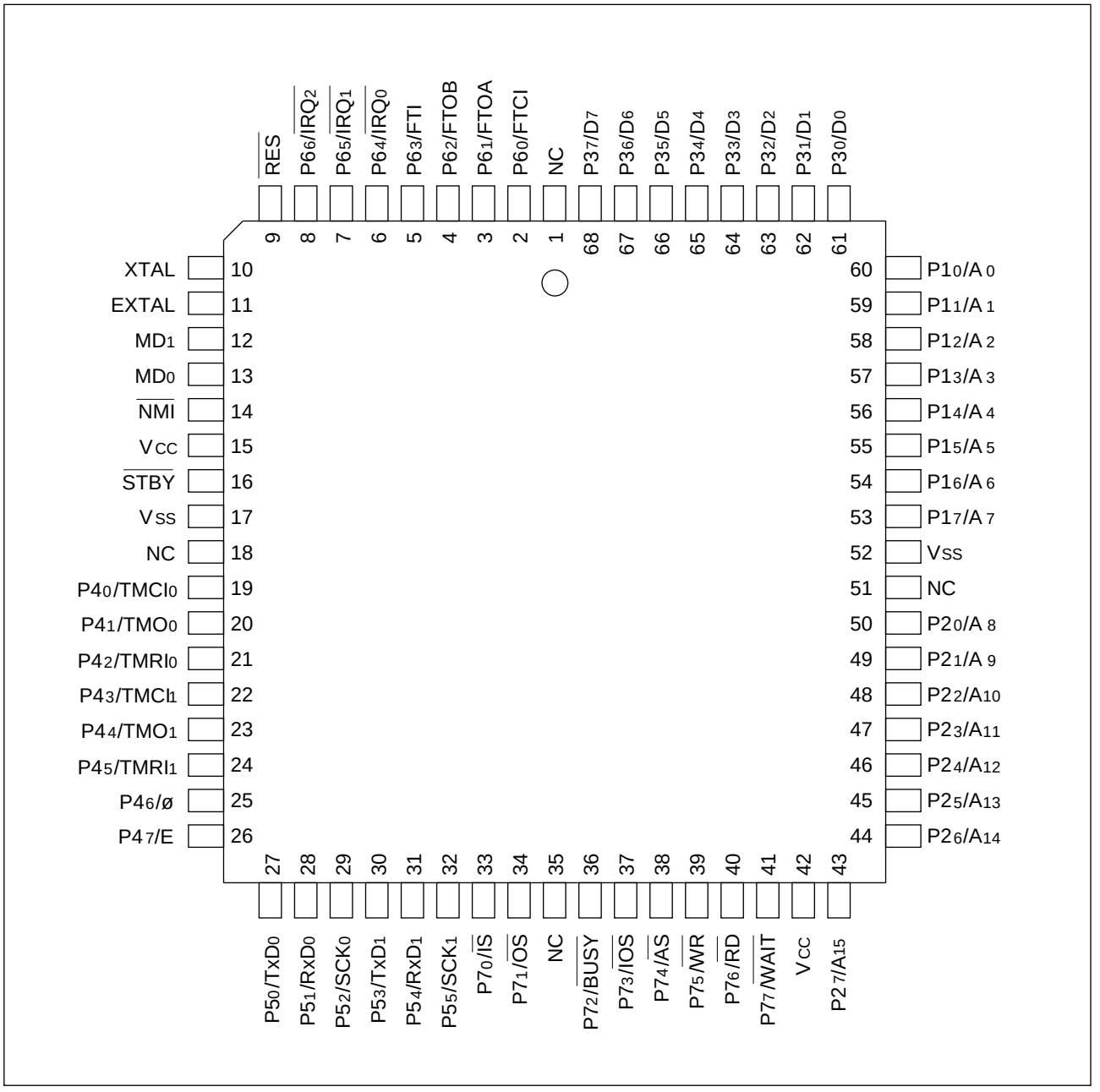



**Figure 1-4. Pin Arrangement (CP-68, Top View)**

### 1.3.2 Pin Functions

**(1) Pin Assignments in Each Operating Mode:** Table 1-2 lists the assignments of the pins of the DC-64S, DP-64S, FP-64A, and CP-68 packages in each operating mode.

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

| Pin no. | | | | | | |
|---|---|---|---|---|---|---|
| DC-64S | | | Expanded modes | | Single-chip mode | PROM |
| DP-64S | FP-64A | CP-68 | Mode 1 | Mode 2 | Mode 3 | mode |
| — | — | 1 | NC | NC | NC | NC |
| 1 | 57 | 2 | $P6_0$/FTCI | $P6_0$/FTCI | $P6_0$/FTCI | NC |
| 2 | 58 | 3 | $P6_1$/FTOA | $P6_1$/FTOA | $P6_1$/FTOA | NC |
| 3 | 59 | 4 | $P6_2$/FTOB | $P6_2$/FTOB | $P6_2$/FTOB | NC |
| 4 | 60 | 5 | $P6_3$/FTI | $P6_3$/FTI | $P6_3$/FTI | NC |
| 5 | 61 | 6 | $P6_4/\overline{IRQ_0}$ | $P6_4/\overline{IRQ_0}$ | $P6_4/\overline{IRQ_0}$ | NC |
| 6 | 62 | 7 | $P6_5/\overline{IRQ_1}$ | $P6_5/\overline{IRQ_1}$ | $P6_5/\overline{IRQ_1}$ | NC |
| 7 | 63 | 8 | $P6_6/\overline{IRQ_2}$ | $P6_6/\overline{IRQ_2}$ | $P6_6/\overline{IRQ_2}$ | NC |
| 8 | 64 | 9 | $\overline{RES}$ | $\overline{RES}$ | $\overline{RES}$ | $V_{PP}$ |
| 9 | 1 | 10 | XTAL | XTAL | XTAL | NC |
| 10 | 2 | 11 | EXTAL | EXTAL | EXTAL | NC |
| 11 | 3 | 12 | $MD_1$ | $MD_1$ | $MD_1$ | $V_{SS}$ |
| 12 | 4 | 13 | $MD_0$ | $MD_0$ | $MD_0$ | $V_{SS}$ |
| 13 | 5 | 14 | $\overline{NMI}$ | $\overline{NMI}$ | $\overline{NMI}$ | $EA_9$ |
| 14 | 6 | 15 | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ |
| 15 | 7 | 16 | $\overline{STBY}$ | $\overline{STBY}$ | $\overline{STBY}$ | $V_{SS}$ |
| 16 | 8 | 17 | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ |
| — | — | 18 | NC | NC | NC | NC |
| 17 | 9 | 19 | $P4_0/TMCI_0$ | $P4_0/TMCI_0$ | $P4_0/TMCI_0$ | $EO_0$ |
| 18 | 10 | 20 | $P4_1/TMO_0$ | $P4_1/TMO_0$ | $P4_1/TMO_0$ | $EO_1$ |
| 19 | 11 | 21 | $P4_2/TMRI_0$ | $P4_2/TMRI_0$ | $P4_2/TMRI_0$ | $EO_2$ |
| 20 | 12 | 22 | $P4_3/TMCI_1$ | $P4_3/TMCI_1$ | $P4_3/TMCI_1$ | $EO_3$ |
| 21 | 13 | 23 | $P4_4/TMO_1$ | $P4_4/TMO_1$ | $P4_4/TMO_1$ | $EO_4$ |
| 22 | 14 | 24 | $P4_5/TMRI_1$ | $P4_5/TMRI_1$ | $P4_5/TMRI_1$ | $EO_5$ |
| 23 | 15 | 25 | Ø | Ø | $P4_6$/Ø | $EO_6$ |
| 24 | 16 | 26 | $P4_7$/E | $P4_7$/E | $P4_7$ | $EO_7$ |
| 25 | 17 | 27 | $P5_0/TxD_0$ | $P5_0/TxD_0$ | $P5_0/TxD_0$ | NC |
| 26 | 18 | 28 | $P5_1/RxD_0$ | $P5_1/RxD_0$ | $P5_1/RxD_0$ | NC |
| 27 | 19 | 29 | $P5_2/SCK_0$ | $P5_2/SCK_0$ | $P5_2/SCK_0$ | NC |

**Notes:** 1. Pins marked NC should be left unconnected.

2. The PROM mode is a non-operating mode used for programming the on-chip ROM. See section 11, ROM for details.

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

| Pin no. | | | | | | |
|---|---|---|---|---|---|---|
| DC-64S | | | Expanded modes | | Single-chip mode | PROM |
| DP-64S | FP-64A | CP-68 | Mode 1 | Mode 2 | Mode 3 | mode |
| 28 | 20 | 30 | $P5_3/TxD_1$ | $P5_3/TxD_1$ | $P5_3/TxD_1$ | NC |
| 29 | 21 | 31 | $P5_4/RxD_1$ | $P5_4/RxD_1$ | $P5_4/RxD_1$ | NC |
| 30 | 22 | 32 | $P5_5/SCK_1$ | $P5_5/SCK_1$ | $P5_5/SCK_1$ | NC |
| 31 | 23 | 33 | $P7_0/\overline{IS}$ | $P7_0/\overline{IS}$ | $P7_0/\overline{IS}$ | $V_{CC}$ |
| 32 | 24 | 34 | $P7_1$ | $P7_1$ | $P7_1/\overline{OS}$ | $V_{CC}$ |
| — | — | 35 | NC | NC | NC | NC |
| 33 | 25 | 36 | $P7_2$ | $P7_2$ | $P7_2/\overline{BUSY}$ | NC |
| 34 | 26 | 37 | $P7_3/\overline{IOS}$ | $P7_3/\overline{IOS}$ | $P7_3$ | NC |
| 35 | 27 | 38 | $\overline{AS}$ | $\overline{AS}$ | $P7_4$ | NC |
| 36 | 28 | 39 | $\overline{WR}$ | $\overline{WR}$ | $P7_5$ | NC |
| 37 | 29 | 40 | $\overline{RD}$ | $\overline{RD}$ | $P7_6$ | NC |
| 38 | 30 | 41 | $\overline{WAIT}$ | $\overline{WAIT}$ | $P7_7$ | NC |
| 39 | 31 | 42 | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ |
| 40 | 32 | 43 | $A_{15}$ | $P2_7/A_{15}$ | $P2_7$ | CE |
| 41 | 33 | 44 | $A_{14}$ | $P2_6/A_{14}$ | $P2_6$ | $EA_{14}$ |
| 42 | 34 | 45 | $A_{13}$ | $P2_5/A_{13}$ | $P2_5$ | $EA_{13}$ |
| 43 | 35 | 46 | $A_{12}$ | $P2_4/A_{12}$ | $P2_4$ | $EA_{12}$ |
| 44 | 36 | 47 | $A_{11}$ | $P2_3/A_{11}$ | $P2_3$ | $EA_{11}$ |
| 45 | 37 | 48 | $A_{10}$ | $P2_2/A_{10}$ | $P2_2$ | $EA_{10}$ |
| 46 | 38 | 49 | $A_9$ | $P2_1/A_9$ | $P2_1$ | OE |
| 47 | 39 | 50 | $A_8$ | $P2_0/A_8$ | $P2_0$ | $EA_8$ |
| — | — | 51 | NC | NC | NC | NC |
| 48 | 40 | 52 | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ |
| 49 | 41 | 53 | $A_7$ | $P1_7/A_7$ | $P1_7$ | $EA_7$ |
| 50 | 42 | 54 | $A_6$ | $P1_6/A_6$ | $P1_6$ | $EA_6$ |
| 51 | 43 | 55 | $A_5$ | $P1_5/A_5$ | $P1_5$ | $EA_5$ |
| 52 | 44 | 56 | $A_4$ | $P1_4/A_4$ | $P1_4$ | $EA_4$ |
| 53 | 45 | 57 | $A_3$ | $P1_3/A_3$ | $P1_3$ | $EA_3$ |
| 54 | 46 | 58 | $A_2$ | $P1_2/A_2$ | $P1_2$ | $EA_2$ |

**Notes:**
1. Pins marked NC should be left unconnected.
2. The PROM mode is a non-operating mode used for programming the on-chip ROM. See section 11, ROM for details.

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

| Pin no. | | | | | | |
|---|---|---|---|---|---|---|
| **DC-64S** | | | **Expanded modes** | | **Single-chip mode** | **PROM** |
| **DP-64S** | **FP-64A** | **CP-68** | **Mode 1** | **Mode 2** | **Mode 3** | **mode** |
| 55 | 47 | 59 | $A_1$ | $P1_1/A_1$ | $P1_1$ | $EA_1$ |
| 56 | 48 | 60 | $A_0$ | $P1_0/A_0$ | $P1_0$ | $EA_0$ |
| 57 | 49 | 61 | $D_0$ | $D_0$ | $P3_0$ | NC |
| 58 | 50 | 62 | $D_1$ | $D_1$ | $P3_1$ | NC |
| 59 | 51 | 63 | $D_2$ | $D_2$ | $P3_2$ | NC |
| 60 | 52 | 64 | $D_3$ | $D_3$ | $P3_3$ | NC |
| 61 | 53 | 65 | $D_4$ | $D_4$ | $P3_4$ | NC |
| 62 | 54 | 66 | $D_5$ | $D_5$ | $P3_5$ | NC |
| 63 | 55 | 67 | $D_6$ | $D_6$ | $P3_6$ | NC |
| 64 | 56 | 68 | $D_7$ | $D_7$ | $P3_7$ | NC |

**Notes:**
1. Pins marked NC should be left unconnected.
2. The PROM mode is a non-operating mode used for programming the on-chip ROM. See section 11, ROM for details.

**(2) Pin Functions:** Table 1-3 gives a concise description of the function of each pin.

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

| Type | Symbol | I/O | Name and function |
|---|---|---|---|
| Power | $V_{CC}$ | I | **Power:** Connected to the power supply (+5 V or +3 V). Connect both $V_{CC}$ pins to the system power supply (+5 V or +3 V). |
| | $V_{SS}$ | I | **Ground:** Connected to ground (0 V). Connect both $V_{SS}$ pins to the system power supply (0 V). |
| Clock | XTAL | I | **Crystal:** Connected to a crystal oscillator. The crystal frequency must be double the desired system clock frequency. If an external clock is input at the EXTAL pin, a reverse-phase clock should be input at the XTAL pin. |
| | EXTAL | I | **External crystal:** Connected to a crystal oscillator or external clock. The frequency of the external clock must be double the desired system clock frequency. See section 14, Clock Pulse Generator for examples of connections to a crystal and external clock. |
| | Ø | O | **System clock:** Supplies the system clock to peripheral devices. |
| | E | O | **Enable clock:** Supplies an E clock to peripheral devices. |
| System control | $\overline{RES}$ | I | **Reset:** A low input causes the chip to reset. |
| | $\overline{STBY}$ | I | **Standby:** A transition to the hardware standby mode (a power-down state) occurs when a low input is received at the $\overline{STBY}$ pin. |
| Address bus | $A_{15}$ to $A_0$ | O | **Address bus:** Address output pins. |
| Data bus | $D_7$ to $D_0$ | I/O | **Data bus:** 8-Bit bidirectional data bus. |
| Bus control | $\overline{WAIT}$ | I | **Wait:** Requests the CPU to insert $T_W$ states into the bus cycle when an off-chip address is accessed. |
| | $\overline{RD}$ | O | **Read:** Goes low to indicate that the CPU is reading an external address. |
| | $\overline{WR}$ | O | **Write:** Goes low to indicate that the CPU is writing to an external address. |
| | $\overline{AS}$ | O | **Address Strobe:** Goes low to indicate that there is a valid address on the address bus. |

**Table 1-3. Pin Functions (2)**

| Type | Symbol | I/O | Name and function |
|---|---|---|---|
| Bus control | $\overline{IOS}$ | O | **I/O Select:** Goes low when the CPU accesses addresses H'FF00 to H'FFFF in expanded mode. Can be used as a chip select signal replacing the upper 8 bits of the address bus when external devices are mapped onto high addresses. |
| Interrupt signals | $\overline{NMI}$ | I | **NonMaskable Interrupt:** Highest-priority interrupt request. The NMIEG bit in the system control register determines whether the interrupt is requested on the rising or falling edge of the $\overline{NMI}$ input. |
| | $\overline{IRQ_0}$ to $\overline{IRQ_2}$ | I | **Interrupt Request 0 to 2:** Maskable interrupt request pins. |
| Operating mode control | $MD_1$, $MD_0$ | I | **Mode:** Input pins for setting the MCU operating mode according to the table below. |

| $MD_1$ | $MD_0$ | Mode | Description |
|---|---|---|---|
| 0 | 1 | Mode 1 | Expanded mode with on-chip ROM disabled |
| 1 | 0 | Mode 2 | Expanded mode with on-chip ROM enabled |
| 1 | 1 | Mode 3 | Single-chip mode |

The inputs at these pins are latched in mode select bits 1 to 0 (MDS1 and MDS0) of the mode control register (MDCR) on the rising edge of the $\overline{RES}$ signal.

| | | | |
|---|---|---|---|
| 16-Bit free-running timer | FTCI | I | **FRT counter Clock Input:** Input pin for an external clock signal for the free-running timer. |
| | FTOA, FTOB | O | **FRT Output compare A and B:** Output pins controlled by comparators A and B of the free-running timer. |
| | FTI | I | **FRT Input capture:** Input capture pin for the free-running timer. |
| 8-Bit timer | $TMO_0$, $TMO_1$ | O | **8-bit TiMer Output (channels 0 and 1):** Compare-match output pins for the 8-bit timers. |
| | $TMCI_0$, $TMCI_1$ | I | **8-bit TiMer Clock Input (channels 0 and 1):** External clock input pins for the 8-bit timer counters. |
| | $TMRI_0$, $TMRI_1$ | I | **8-bit TiMer Reset Input (channels 0 and 1):** High input at these pins resets the 8-bit timers. |

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

| Type | Symbol | I/O | Name and function |
|---|---|---|---|
| Serial communication interface | $TxD_0$<br>$TxD_1$ | O | **Serial Transmit Data (channels 0 and 1):** Data output pins for the serial communication interface. |
| | $RxD_0$<br>$RxD_1$ | I | **Serial Receive Data (channels 0 and 1):** Data input pins for the serial communication interface. |
| | $SCK_0$<br>$SCK_1$ | I/O | **Serial ClocK (channels 0 and 1):** Input/output pins for the serial clock signals. |
| General-purpose I/O | $P1_7$ to $P1_0$ | I/O | **Port 1:** An 8-bit input/output port with programmable MOS input pull-ups and LED driving capability. The direction of each bit can be selected in the port 1 data direction register (P1DDR). |
| | $P2_7$ to $P2_0$ | I/O | **Port 2:** An 8-bit input/output port with programmable MOS input pull-ups and LED driving capability. The direction of each bit can be selected in the port 2 data direction register (P2DDR). |
| | $P3_7$ to $P3_0$ | I/O | **Port 3:** An 8-bit input/output port with programmable MOS input pull-ups. The direction of each bit can be selected in the port 3 data direction register (P3DDR). |
| | $P4_7$ to $P4_0$ | I/O | **Port 4:** An 8-bit input/output port with programmable MOS input pull-ups. The direction of each bit (except $P4_6$) can be selected in the port 4 data direction register (P4DDR). |
| | $P5_5$ to $P5_0$ | I/O | **Port 5:** A 6-bit input/output port with programmable MOS input pull-ups. The direction of each bit can be selected in the port 5 data direction register (P5DDR). |
| | $P6_6$ to $P6_0$ | I/O | **Port 6:** A 7-bit input/output port with programmable MOS input pull-ups. The direction of each bit can be selected in the port 6 data direction register (P6DDR). |
| | $P7_7$ to $P7_0$ | I/O | **Port 7:** An 8-bit input/output port with programmable MOS input pull-ups. The direction of each bit can be selected in the port 7 data direction register (P7DDR). |
| Parallel handshaking interface | $P3_7$ to $P3_0$ | I/O | **Data Input/Output:** Data input/output pins for the parallel handshaking interface. |
| | $\overline{IS}$ | I | **Input Strobe:** Strobe input signal from an external device. |
| | $\overline{OS}$ | O | **Output Strobe:** Strobe output signal to an external device. |
| | $\overline{BUSY}$ | O | **Busy:** Notifies an external device that the H8/325 Series chip is not ready to receive data. |

# Section 2. MCU Operating Modes and Address Space

## 2.1 Overview

The H8/325 Series operates in three modes numbered 1, 2, and 3. An additional non-operating mode (mode 0) is used for programming the PROM version of the H8/325. The mode is selected by the inputs at the mode pins ($MD_1$ and $MD_0$) at the instant when the chip comes out of a reset. As indicated in table 2-1, the mode determines the size of the address space and the usage of on-chip ROM and on-chip RAM. The ROMless versions (HD6413258, HD6413238) are used only in mode 1 (expanded mode with on-chip ROM disabled).

**Table 2-1. Operating Modes**

| $MD_1$ | $MD_0$ | Mode | Address space | On-chip ROM | On-chip RAM |
|---|---|---|---|---|---|
| Low | Low | Mode 0 | — | — | — |
| Low | High | Mode 1 | Expanded | Disabled | Enabled* |
| High | Low | Mode 2 | Expanded | Enabled | Enabled* |
| High | High | Mode 3 | Single-chip | Enabled | Enabled |

* If the RAME bit in the system control register (SYSCR) is cleared to 0, off-chip memory can be accessed instead.

Modes 1 and 2 are expanded modes that permit access to off-chip memory and peripheral devices. The maximum address space supported by these externally expanded modes is 64 kbytes.

In mode 3 (single-chip mode), only on-chip ROM and RAM and the on-chip register field are used. All ports are available for general-purpose input and output.

Mode 0 is inoperative in the H8/325 Series. Avoid setting the mode pins to mode 0.

## 2.2 Mode Descriptions

**Mode 1 (Expanded Mode without On-Chip ROM):** Mode 1 supports a 64-kbyte address space most of which is off-chip. In particular, the interrupt vector table is located in off-chip memory. The on-chip ROM is not used. Software can select whether to use the on-chip RAM. Ports 1, 2, 3 and 7 are used for the address and data bus lines and control signals as follows:

Ports 1 and 2: Address bus
Port 3: Data bus
Port 7 (partly): Bus control signals

**Mode 2 (Expanded Mode with On-Chip ROM):** Mode 2 supports a 64-kbyte address space which includes the on-chip ROM. Software can select whether or not to use the on-chip RAM, and can select the usage of pins in ports 1 and 2.

Ports 1 and 2: Address bus (see note)
Port 3: Data bus
Port 7 (partly): Bus control signals

**Note:** In mode 2, ports 1 and 2 are initially general-purpose input ports. Software must change the desired pins to output before using them for the address bus. See section 5, I/O Ports for details.

**Mode 3 (Single-Chip Mode):** In this mode all memory is on-chip. Since no off-chip memory is accessed, there is no external address bus. All ports are available for general-purpose input and output.

## 2.3 Address Space Map

Figures 2-1 to 2-6 show memory maps of the H8/3257, H8/3256, H8/325, H8/324, H8/323, and H8/322 in each of the three operating modes. The on-chip register field consists of control, status, and data registers for the on-chip supporting modules and I/O ports.

Off-chip addresses can be accessed only in the expanded modes. Access to an off-chip address in the single-chip mode does not cause an address error, but all 1 data are returned.

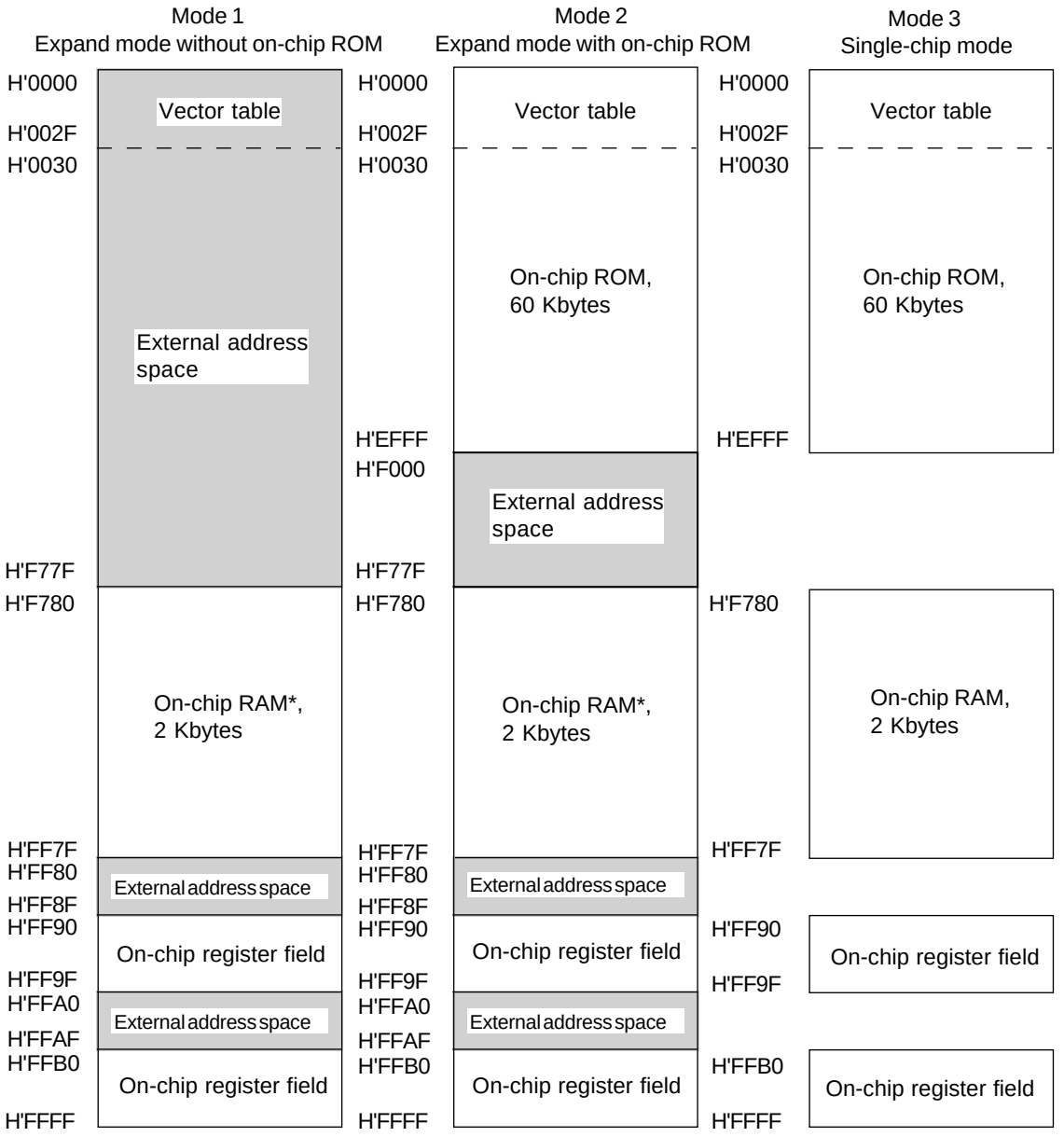
### 2.3.1 Access Speed

On-chip ROM and RAM are accessed a word (16 bits) at a time in two states. (A “state” is one system clock cycle.) The on-chip register field is accessed a byte at a time in three states.

External memory is accessed a byte at a time in three or more states. The basic bus cycle is three states, but additional wait states can be inserted on request.

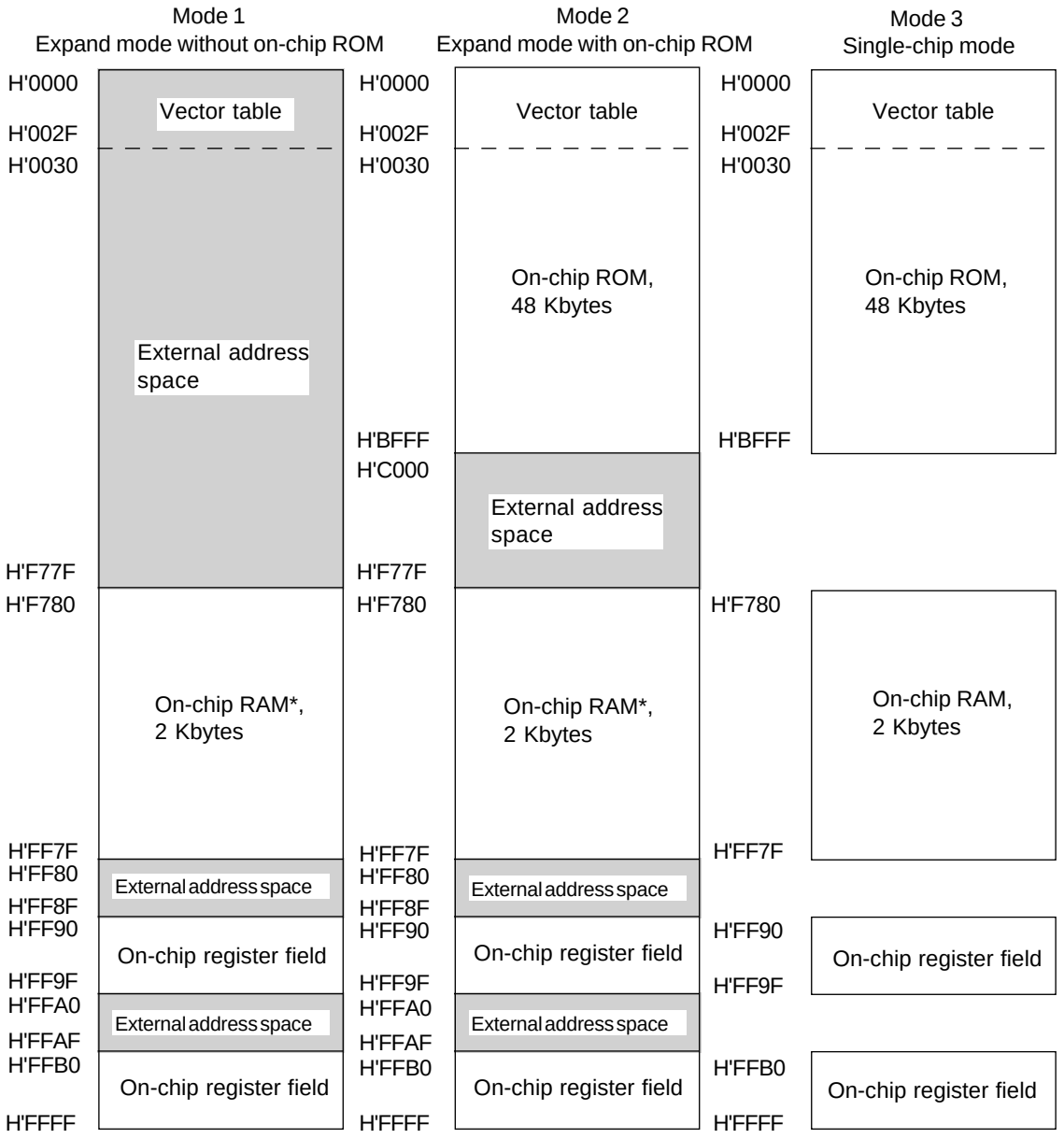
### 2.3.2 $\overline{\text{IOS}}$

There are two gaps in the on-chip address space above the on-chip RAM. Addresses H’FF80 to H’FF8F, situated between the on-chip RAM and register field, are off-chip. Addresses H’FFA0 to H’FFAF are also off-chip. These 32 addresses can be conveniently assigned to external I/O devices. To simplify the addressing of devices at these addresses, an $\overline{\text{IOS}}$ signal is provided that goes low when the CPU accesses addresses H’FF00 to H’FFFF. The $\overline{\text{IOS}}$ signal can be used in place of the upper 8 bits of the address bus.



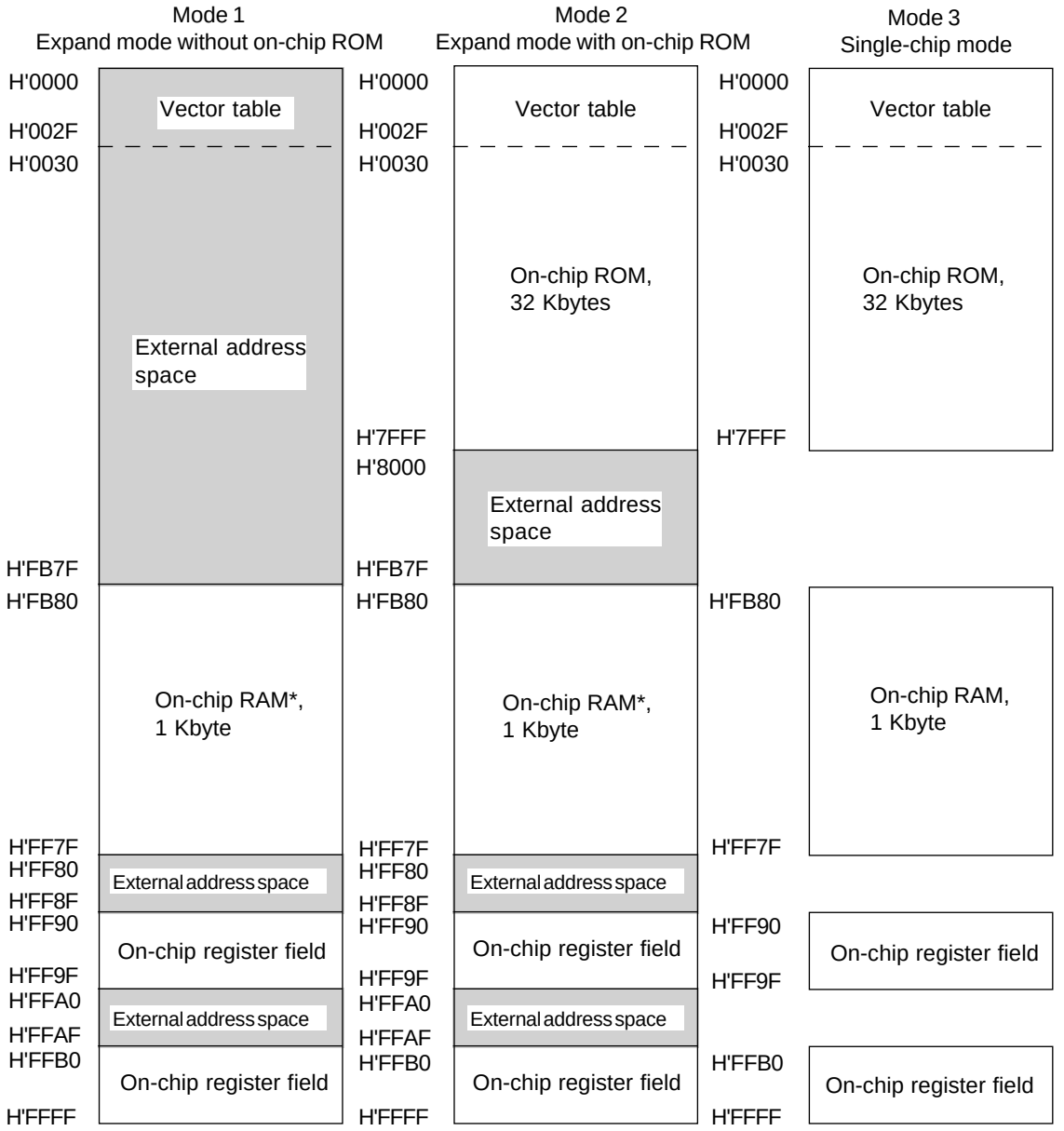

* External memory can be accessed at these addresses when the RAME bit in the system control register (SYSCR) is cleared to 0.

**Figure 2-1. H8/3257 Address Space Map**



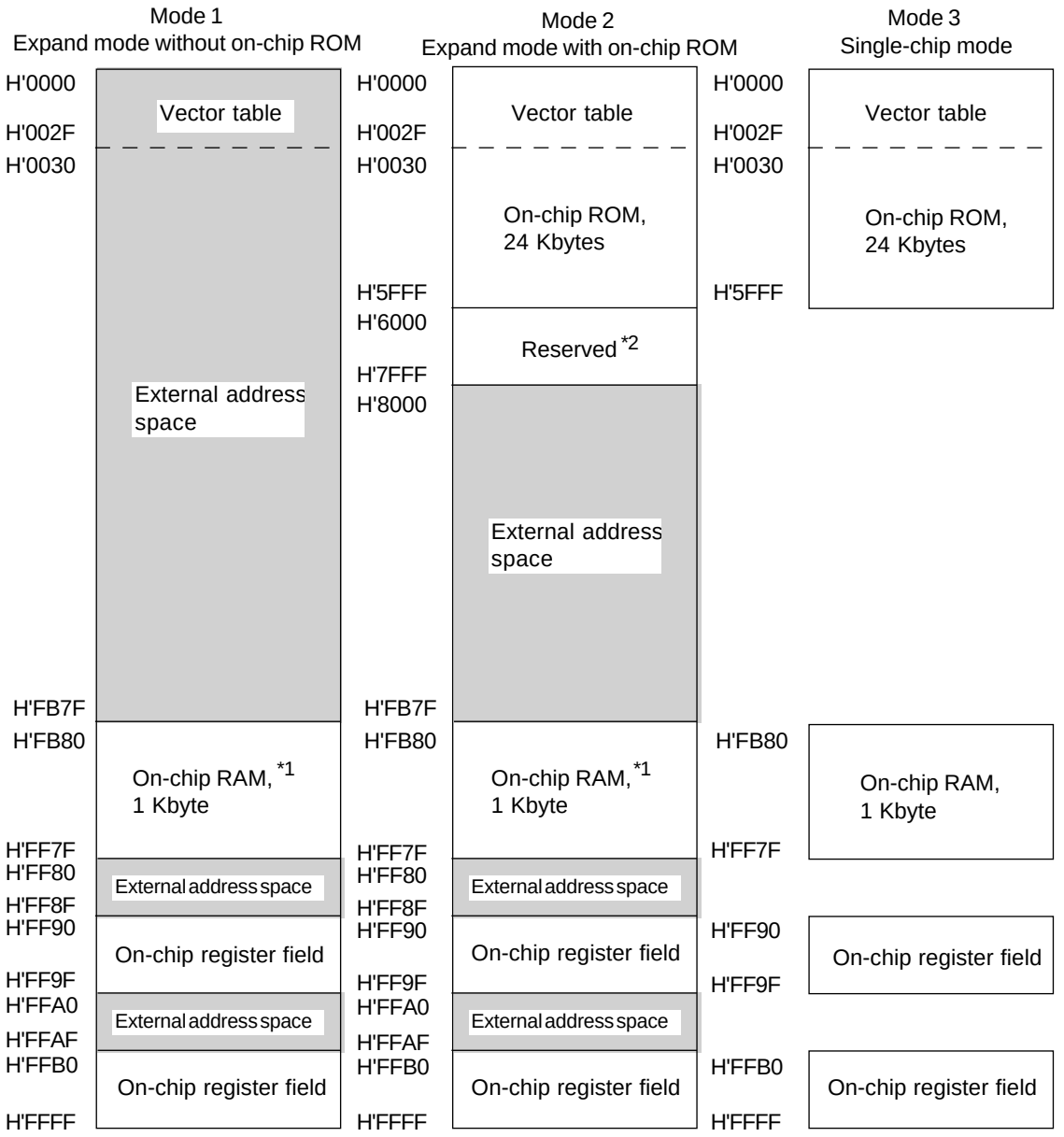

* External memory can be accessed at these addresses when the RAME bit in the system control register (SYSCR) is cleared to 0.

**Figure 2-2. H8/3256 Address Space Map**



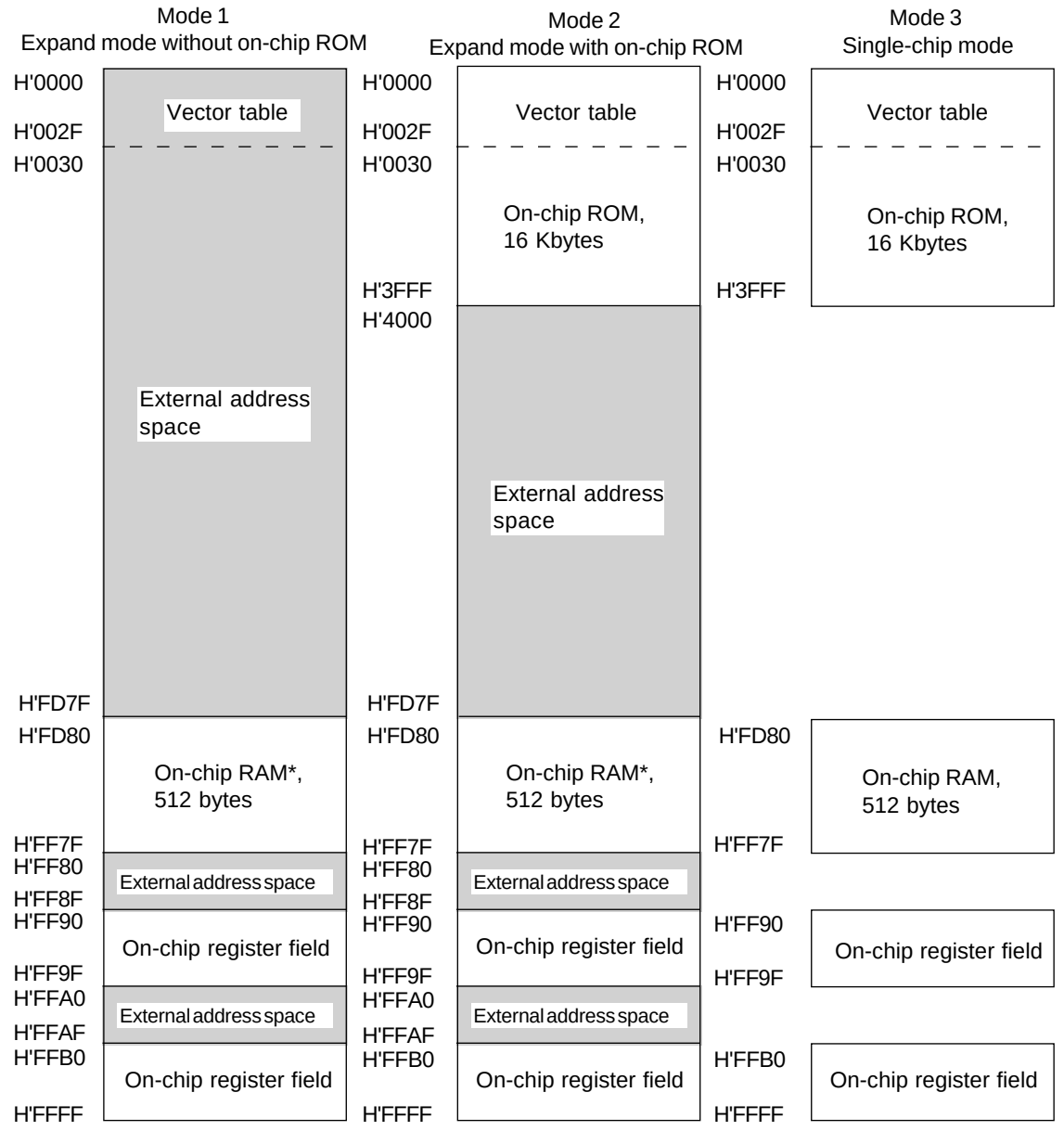

* External memory can be accessed at these addresses when the RAME bit in the system control register (SYSCR) is cleared to 0.

**Figure 2-3. H8/325 Address Space Map**



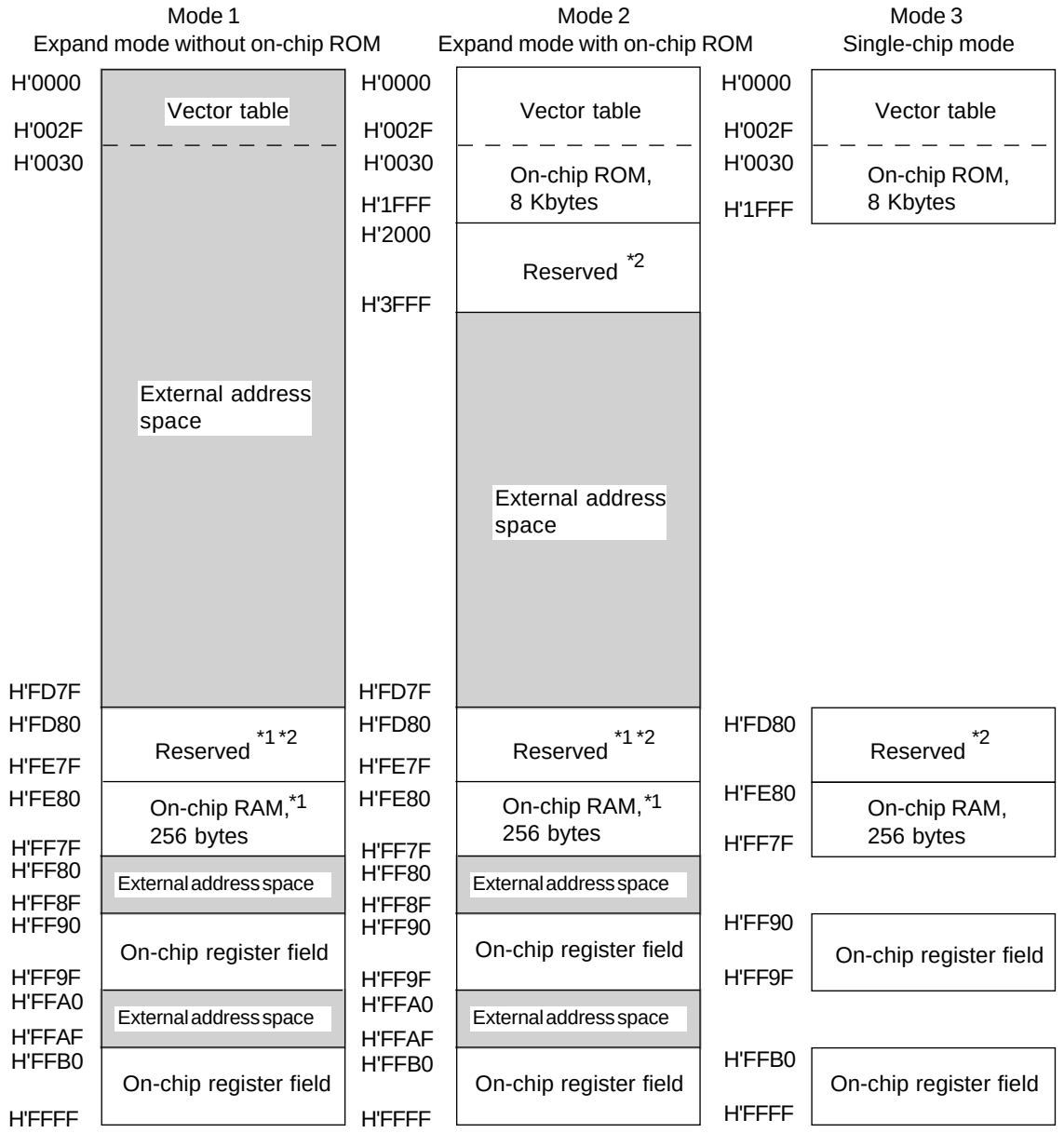

*1 This area can be used as external address space when the RAME bit of SYSCR is 0.
*2 Data read or write is not permitted in these modes.

**Figure 2-4. H8/324 Address Space Map**




* External memory can be accessed at these addresses when the RAME bit in the system control register (SYSCR) is cleared to 0.

**Figure 2-5. H8/323 Address Space Map**




*1 External memory can be accessed at these addresses when the RAME bit in the system control register (SYSCR) is cleared 0.

*2 Data read or write is not permitted in these modes.

**Figure 2-6. H8/322 Address Space Map**

## 2.4 Mode and System Control Registers (MDCR and SYSCR)

Two of the control registers in the register field are the mode control register (MDCR) and system control register (SYSCR). The mode control register controls the MCU mode: the operating mode of the H8/325 Series chip. The system control register has a bit that enables or disables the on-chip RAM. Table 2-2 lists the attributes of these registers.

**Table 2-2. Mode and System Control Registers**

| Name | Abbreviation | Read/Write | Address |
|---|---|---|---|
| Mode control register | MDCR | R | H'FFC5 |
| System control register | SYSCR | R/W | H'FFC4 |

### 2.4.1 Mode Control Register (MDCR)—H'FFC5

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MDS1 | MDS0 |
| Initial value | 1 | 1 | 1 | 0 | 0 | 1 | * | * |
| Read/Write | R | R | R | R | R | R | R | R |

* Initialized according to $MD_1$ and $MD_0$ inputs.

**Bits 7 to 5 and 2—Reserved:** These bits cannot be modified and are always read as 1.

**Bits 4 and 3—Reserved:** These bits cannot be modified and are always read as 0.

**Bits 1 and 0—Mode Select 1 and 0 (MDS1 and MDS0):** These bits indicate the values of the mode pins ($MD_1$ and $MD_0$) latched on the rising edge of the $\overline{RES}$ signal. These bits can be read but not written.

**Coding Example:** To test whether the MCU is operating in mode 1:

```
MOV.B @H'FFC5, R0L
CMP.B #H'E5, R0L
```

The comparison is with H'E5 instead of H'01 because bits 7, 6, 5, and 2 are always read as 1.

### 2.4.2 System Control Register (SYSCR)—H'FFC4

By setting or clearing bit 0 of the system control register, software can enable or disable the on-chip RAM.

The other bits in the system control register concern the software standby mode and the valid edge of the $\overline{\text{NMI}}$ signal. These bits will be described in section 4, Exception Handling and section 12, Power-Down State.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSBY | STS2 | STS1 | STS0 | — | NMIEG | — | RAME |
| Initial value | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | — | R/W | — | R/W |

**Bit 0—RAM Enable (RAME):** This bit enables or disables the on-chip RAM. When the on-chip RAM is disabled, accesses to the corresponding addresses are directed off-chip.

The RAME bit is initialized to 1 by a reset, enabling the on-chip RAM. The setting of the RAME bit is not altered in the sleep mode or software standby mode. It should be cleared to 0 before entering the hardware standby mode. See section 12, Power-Down State.

**Bit 0**

| RAME | Description | |
|---|---|---|
| 0 | The on-chip RAM is disabled. | |
| 1 | The on-chip RAM is enabled. | (Initial state) |

**Coding Example:** To disable the on-chip RAM:

```
BCLR #0, @H'FFC4
```

# Section 3. CPU

## 3.1 Overview

The H8/325 Series has the generic H8/300 CPU: an 8-bit central processing unit with a speed-oriented architecture featuring sixteen general registers. This section describes the CPU features and functions, including a concise description of the addressing modes and instruction set. For further details on the instructions, see the *H8/300 Series Programming Manual*.

### 3.1.1 Features

The main features of the H8/300 CPU are listed below.

- Two-way register configuration
  - — Sixteen 8-bit general registers, or
  - — Eight 16-bit general registers
- Instruction set with 57 basic instructions, including:
  - — Multiply and divide instructions
  - — Powerful bit-manipulation instructions
- Eight addressing modes
  - — Register direct (Rn)
  - — Register indirect (@Rn)
  - — Register indirect with displacement (@(d:16, Rn))
  - — Register indirect with post-increment or pre-decrement (@Rn+ or @–Rn)
  - — Absolute address (@aa:8 or @aa:16)
  - — Immediate (#xx:8 or #xx:16)
  - — PC-relative (@(d:8, PC))
  - — Memory indirect (@@aa:8)
- Maximum 64K-byte address space
- High-speed operation
  - — All frequently-used instructions are executed two to four states
  - — The maximum clock rate is 10MHz
    - — 8- or 16-bit register-register add or subtract: 0.2µs
    - — 8 × 8-bit multiply: 1.4µs
    - — 16 ÷ 8-bit divide: 1.4µs
- Power-down mode
  - — SLEEP instruction

## 3.2 Register Configuration

Figure 3-1 shows the register structure of the CPU. There are two groups of registers: the general registers and control registers.

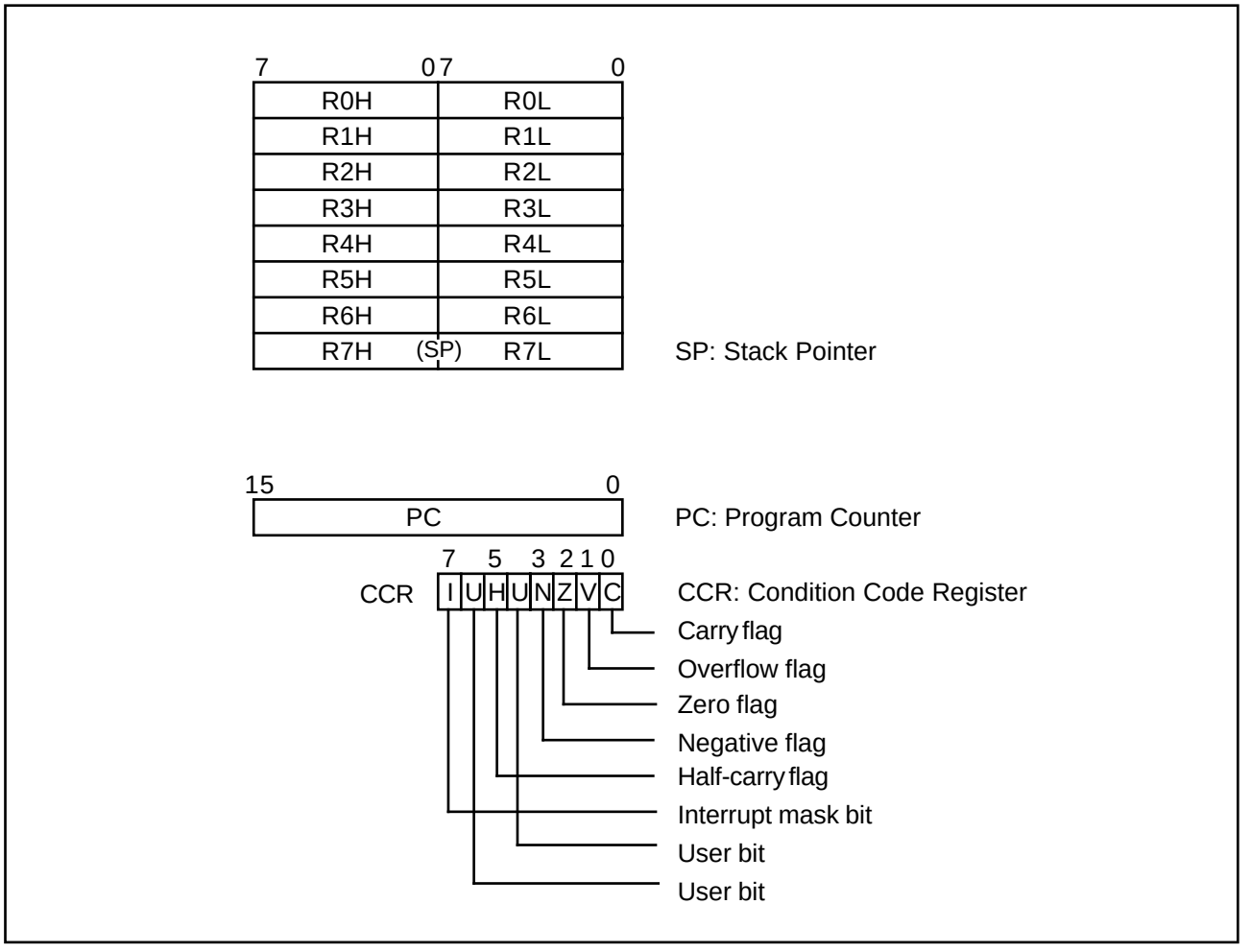



**Figure 3-1. CPU Registers**

### 3.2.1 General Registers

All the general registers can be used as both data registers and address registers. When used as address registers, the general registers are accessed as 16-bit registers (R0 to R7). When used as data registers, they can be accessed as 16-bit registers, or the high and low bytes can be accessed separately as 8-bit registers.

R7 also functions as the stack pointer, used implicitly by hardware in processing interrupts and subroutine calls. In assembly-language coding, R7 can also be denoted by the letters SP. As indicated in figure 3-2, R7 (SP) points to the top of the stack.

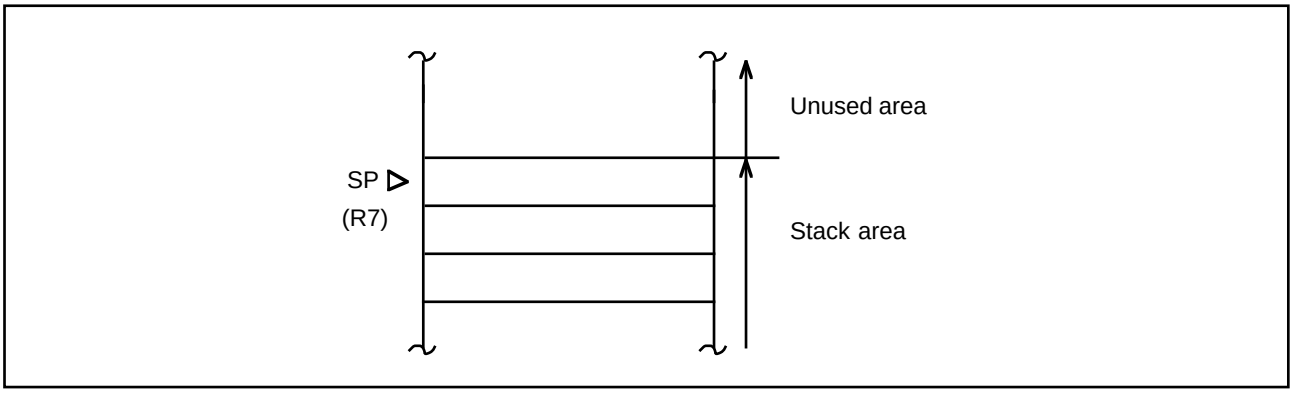



**Figure 3-2. Stack Pointer**

### 3.2.2 Control Registers

The CPU control registers include a 16-bit program counter (PC) and an 8-bit condition code register (CCR).

**(1) Program Counter (PC):** This 16-bit register indicates the address of the next instruction the CPU will execute. Each instruction is accessed in 16 bits (1 word), so the least significant bit of the PC is ignored (always regarded as 0).

**(2) Condition Code Register (CCR):** This 8-bit register contains internal status information, including carry (C), overflow (V), zero (Z), negative (N), and half-carry (H) flags and the interrupt mask bit (I).

**Bit 7—Interrupt Mask Bit (I):** When this bit is set to "1," all interrupts except NMI are masked. This bit is set to "1" automatically by a reset and at the start of interrupt handling.

**Bit 6—User Bit (U):** This bit can be written and read by software for its own purposes.

**Bit 5—Half-Carry (H):** This bit is set to "1" when the ADD.B, ADDX.B, SUB.B, SUBX.B, NEG.B, or CMP.B instruction causes a carry or borrow out of bit 3, and is cleared to "0" otherwise. Similarly, it is set to "1" when the ADD.W, SUB.W, or CMP.W instruction causes a carry or borrow out of bit 11, and cleared to "0" otherwise. It is used implicitly in the DAA and DAS instructions.

**Bit 4—User Bit (U):** This bit can be written and read by software for its own purposes.

**Bit 3—Negative (N):** This bit indicates the most significant bit (sign bit) of the result of an instruction.

**Bit 2—Zero (Z):** This bit is set to “1” to indicate a zero result and cleared to “0” to indicate a nonzero result.

**Bit 1—Overflow (V):** This bit is set to “1” when an arithmetic overflow occurs, and cleared to “0” at other times.

**Bit 0—Carry (C):** This bit is used by:

- Add and subtract instructions, to indicate a carry or borrow at the most significant bit of the result
- Shift and rotate instructions, to store the value shifted out of the most significant or least significant bit
- Bit manipulation and bit load instructions, as a bit accumulator

The LDC, STC, ANDC, ORC, and XORC instructions enable the CPU to load and store the CCR, and to set or clear selected bits by logic operations.

Some instructions leave some or all of the flag bits unchanged. The action of each instruction on the flag bits is shown in Appendix A.1, “Instruction Set List.” See the *H8/300 Series Programming Manual* for further details.

### 3.2.3 Initial Register Values

When the CPU is reset, the program counter (PC) is loaded from the vector table and the interrupt mask bit (I) in the CCR is set to “1.” The other CCR bits and the general registers are not initialized.

In particular, the stack pointer (R7) is not initialized. To prevent program crashes the stack pointer should be initialized by software, by the first instruction executed after a reset.

## 3.3 Addressing Modes

The H8/325 supports eight addressing modes. Each instruction uses a subset of these addressing modes.

**(1) Register Direct—Rn:** The register field of the instruction specifies an 8- or 16-bit general register containing the operand. In most cases the general register is accessed as an 8-bit register. Only the MOV.W, ADD.W, SUB.W, CMP.W, ADDS, SUBS, MULXU (8 bits × 8 bits), and DIVXU (16 bits ÷ 8 bits) instructions have 16-bit operands.

**(2) Register indirect—@Rn:** The register field of the instruction specifies a 16-bit general register containing the address of the operand.

**(3) Register Indirect with Displacement—@(d:16, Rn):** This mode, which is used only in MOV instructions, is similar to register indirect but the instruction has a second word (bytes 3 and 4) which is added to the contents of the specified general register to obtain the operand address. For the MOV.W instruction, the resulting address must be even.

**(4) Register Indirect with Post-Increment or Pre-Decrement—@Rn+ or @–Rn:**

- Register indirect with Post-Increment—@Rn+
  The @Rn+ mode is used with MOV instructions that load registers from memory.
  It is similar to the register indirect mode, but the 16-bit general register specified in the register field of the instruction is incremented after the operand is accessed. The size of the increment is 1 or 2 depending on the size of the operand: 1 for MOV.B; 2 for MOV.W. For MOV.W, the original contents of the 16-bit general register must be even.

- Register Indirect with Pre-Decrement—@–Rn
  The @–Rn mode is used with MOV instructions that store register contents to memory.
  It is similar to the register indirect mode, but the 16-bit general register specified in the register field of the instruction is decremented before the operand is accessed. The size of the decrement is 1 or 2 depending on the size of the operand: 1 for MOV.B; 2 for MOV.W. For MOV.W, the original contents of the 16-bit general register must be even.

**(5) Absolute Address—@aa:8 or @aa:16:** The instruction specifies the absolute address of the operand in memory. The MOV.B instruction uses an 8-bit absolute address of the form H'FFxx. The upper 8 bits are assumed to be 1, so the possible address range is H'FF00 to H'FFFF (65280 to 65535). The MOV.B, MOV.W, JMP, and JSR instructions can use 16-bit absolute addresses.

**(6) Immediate—#xx:8 or #xx:16:** The instruction contains an 8-bit operand in its second byte, or a 16-bit operand in its third and fourth bytes. Only MOV.W instructions can contain 16-bit immediate values.
The ADDS and SUBS instructions implicitly contain the value 1 or 2 as immediate data. Some bit manipulation instructions contain 3-bit immediate data (#xx:3) in the second or fourth byte of the instruction, specifying a bit number.

**(7) PC-Relative—@(d:8, PC):** This mode is used to generate branch addresses in the Bcc and BSR instructions. An 8-bit value in byte 2 of the instruction code is added as a sign-extended value to the program counter contents. The result must be an even number. The possible branching range is –126 to +128 bytes (–63 to +64 words) from the current address.

**(8) Memory Indirect—@@aa:8:** This mode can be used by the JMP and JSR instructions. The second byte of the instruction code specifies an 8-bit absolute address from H'0000 to H'00FF (0 to 255). The word located at this address contains the branch address. Note that addresses H'0000 to H'003D (0 to 61) are located in the vector table.

If an odd address is specified as a branch destination or as the operand address of a MOV.W instruction, the least significant bit is regarded as "0," causing word access to be performed at the address preceding the specified address. See section 3.4.2, "Memory Data Formats" for further information.

## 3.4 Data Formats

The H8/300 CPU can process 1-bit data, 4-bit (BCD) data, 8-bit (byte) data, and 16-bit (word) data.

- Bit manipulation instructions operate on 1-bit data specified as bit n (n = 0, 1, 2, ..., 7) in a byte operand.
- All arithmetic and logic instructions except ADDS and SUBS can operate on byte data.
- The DAA and DAS instruction perform decimal arithmetic adjustments on byte data in packed BCD form. Each nibble of the byte is treated as a decimal digit.
- The MOV.W, ADD.W, SUB.W, CMP.W, ADDS, SUBS, MULXU (8 bits × 8 bits), and DIVXU (16 bits ÷ 8 bits) instructions operate on word data.

### 3.4.1 Data Formats in General Registers

Data of all the sizes above can be stored in general registers as shown in figure 3-3.

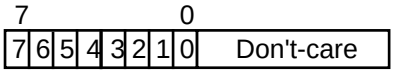
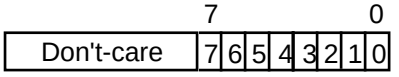
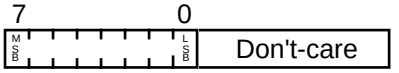
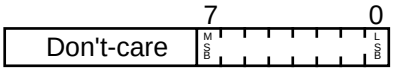
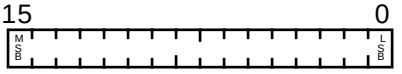
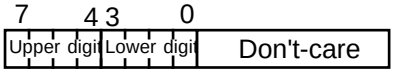
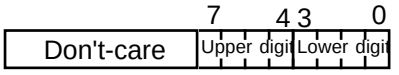
| Data type | Register No. | Data format |
|---|---|---|
| 1-Bit data | RnH |   |
| 1-Bit data | RnL |   |
| Byte data | RnH |   |
| Byte data | RnL |   |
| Word data | Rn |   |
| 4-Bit BCD data | RnH |   |
| 4-Bit BCD data | RnL |   |

**Figure 3-3. Register Data Formats**

Note:

RnH: Upper digit of general register

RnL: Lower digit of general register

MSB: Most significant Bit

LSB: Least significant Bit

### 3.4.2 Memory Data Formats

Figure 3-4 indicates the data formats in memory.

Word data stored in memory must always begin at an even address. In word access the least significant bit of the address is regarded as “0.” If an odd address is specified, no address error occurs but the access is performed at the preceding even address. This rule affects MOV.W instructions and branching instructions, and implies that only even addresses should be stored in the vector table.

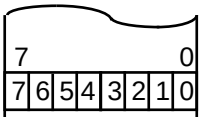
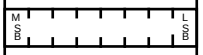
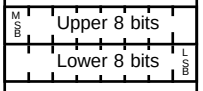
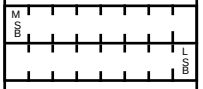
| Data type | Address | Data format |
|---|---|---|
| 1-Bit data | Address n |   |
| Byte data | Address n |   |
| Word data | Even address<br>Odd address |   |
| Byte data (CCR) on stack | Even address<br>Odd address |   |
| Word data on stack | Even address<br>Odd address |   |
| | CCR: Condition Code Register<br>*: Ignored when return | |

**Figure 3-4. Memory Data Formats**

The stack must always be accessed a word at a time. When the CCR is pushed on the stack, two identical copies of the CCR are pushed to make a complete word. When they are returned, the lower byte is ignored.

## 3.5 Instruction Set

Table 3-1 lists the H8/325 Series instruction set.

**Table 3-1. Instruction Classification**

| Function | Instructions | Types |
|---|---|---|
| Data transfer | MOV, MOVTPE, MOVFPE, PUSH*1, POP*1 | 3 |
| Arithmetic operations | ADD, SUB, ADDX, SUBX, INC, DEC, ADDS, SUBS, DAA, DAS, MULXU, DIVXU, CMP, NEG | 14 |
| Logic operations | AND, OR, XOR, NOT | 4 |
| Shift | SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR | 8 |
| Bit manipulation | BSET, BCLR, BNOT, BTST, BAND, BIAND, BOR, BIOR, BXOR, BIXOR, BLD, BILD, BST, BIST | 14 |
| Branch | Bcc*2, JMP, BSR, JSR, RTS | 5 |
| System control | RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP | 8 |
| Block data transfer | EEPMOV | 1 |
| | Total | 57 |

*1 PUSH Rn is equivalent to MOV.W Rn, @–SP.
POP Rn is equivalent to MOV.W @SP+, Rn.

*2 Bcc is a conditional branch instruction in which cc represents a condition code.

The following sections give a concise summary of the instructions in each category, and indicate the bit patterns of their object code. The notation used is defined next.

**Operation Notation**

| | |
|---|---|
| Rd | General register (destination) |
| Rs | General register (source) |
| Rn, Rm | General register |
| $r_n$, $r_m$ | General register field |
| <EAs> | Effective address: general register or memory location |
| (EAd) | Destination operand |
| (EAs) | Source operand |
| SP | Stack pointer |
| PC | Program counter |
| CCR | Condition code register |
| N | N (negative) bit of CCR |
| Z | Z (zero) bit of CCR |
| V | V (overflow) bit of CCR |
| C | C (carry) bit of CCR |
| #imm | Immediate data |
| #xx:3 | 3-Bit immediate data |
| #xx:8 | 8-Bit immediate data |

| | |
|---|---|
| #xx:16 | 16-Bit immediate data |
| op | Operation field |
| disp | Displacement |
| abs | Absolute address |
| B | Byte |
| W | Word |
| + | Addition |
| – | Subtraction |
| × | Multiplication |
| ÷ | Division |
| ∧ | AND logical |
| ∨ | OR logical |
| ⊕ | Exclusive OR logical |
| → | Move |
| ↔ | Exchange |
| ¬ | Not |
| cc | Condition field |

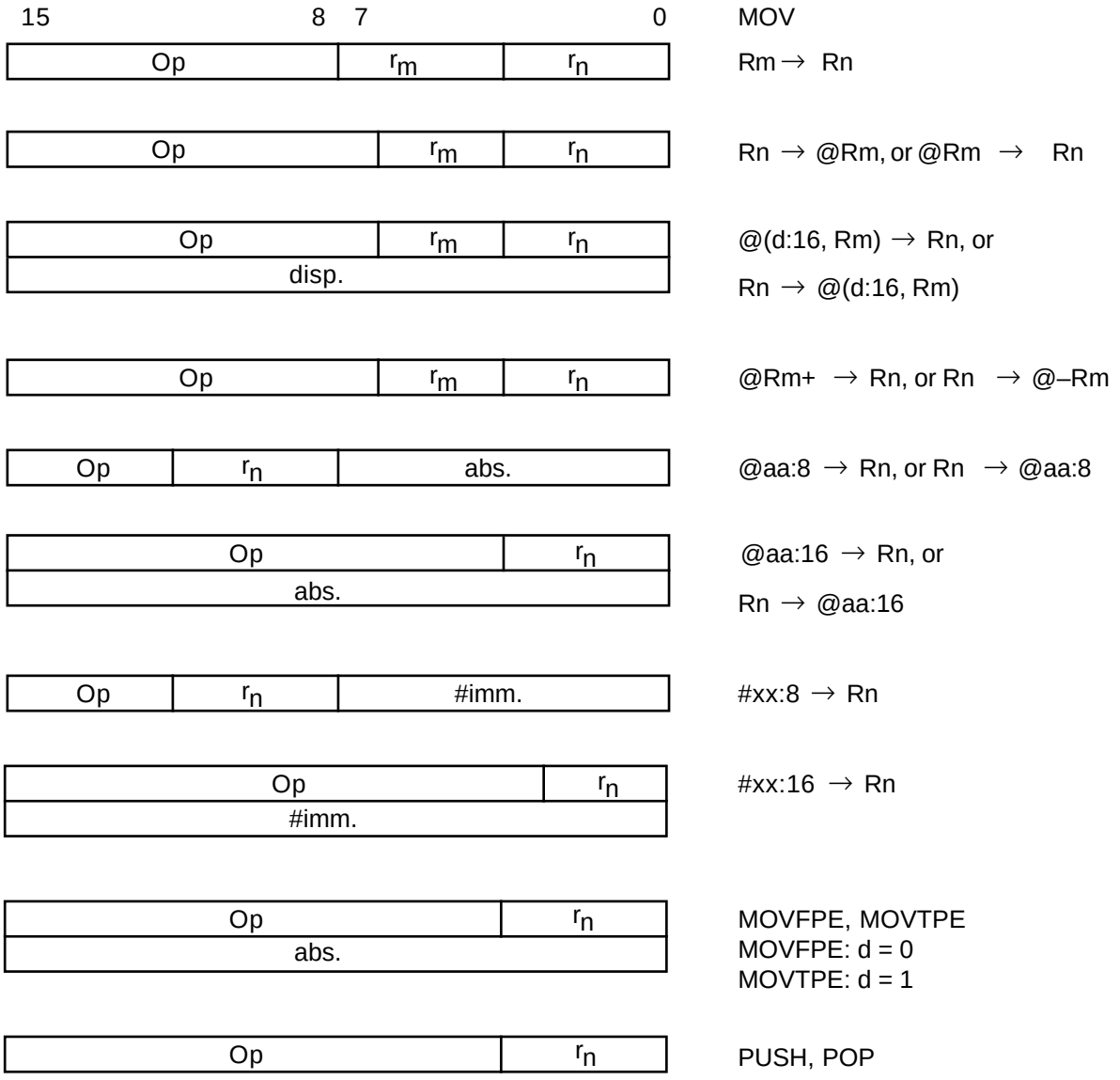
### 3.5.1 Data Transfer Instructions

Table 3-2 describes the data transfer instructions. Figure 3-5 shows their object code formats.

**Table 3-2. Data Transfer Instructions**

| Instruction | Size* | Function |
|---|---|---|
| MOV | B/W | (EAs) → Rd, Rs → (EAd)<br>Moves data between two general registers or between a general register and memory, or moves immediate data to a general register.<br>The Rn, @Rn, @(d:16, Rn), @aa:16, #xx:8 or #xx:16, @–Rn, and @Rn+ addressing modes are available for byte or word data. The @aa:8 addressing mode is available for byte data only.<br>The @–R7 and @R7+ modes require word operands. Do not specify byte size for these two modes. |
| MOVTPE | B | Rs → (EAd)<br>Transfers data from a general register to memory in synchronization with the E clock. |
| MOVFPE | B | (EAs) → Rd<br>Transfers data from memory to a general register in synchronization with the E clock. |
| PUSH | W | Rn → @–SP<br>Pushes a 16-bit general register onto the stack. Equivalent to MOV.W Rn, @–SP. |
| POP | W | @SP+ → Rn<br>Pops a 16-bit general register from the stack. Equivalent to MOV.W @SP+, Rn. |

* Size: operand size
B: Byte
W: Word




**Notation**

| | |
|---|---|
| Op: | Operation field |
| d: | Direction field (0–load from; 1–store to) |
| $r_m$, $r_n$: | Register field |
| disp.: | Displacement |
| abs.: | Absolute address |
| #imm.: | Immediate data |

**Figure 3-5. Data Transfer Instruction Codes**

### 3.5.2 Arithmetic Operations

Table 3-3 describes the arithmetic instructions. See figure 3-6 in section 3.5.4, “Shift Operations” for their object codes.

**Table 3-3. Arithmetic Instructions**

| Instruction | Size* | Function |
|---|---|---|
| ADD<br>SUB | B/W | Rd ± Rs → Rd, Rd + #imm → Rd<br>Performs addition or subtraction on data in two general registers, or addition on immediate data and data in a general register. Immediate data cannot be subtracted from data in a general register. Word data can be added or subtracted only when both words are in general registers. |
| ADDX<br>SUBX | B | Rd ± Rs ± C → Rd, Rd ± #imm ± C → Rd<br>Performs addition or subtraction with carry or borrow on byte data in two general registers, or addition or subtraction on immediate data and data in a general register. |
| INC<br>DEC | B | Rd ± #1 → Rd<br>Increments or decrements a general register. |
| ADDS<br>SUBS | W | Rd ± #imm → Rd<br>Adds or subtracts immediate data to or from data in a general register. The immediate data must be 1 or 2. |
| DAA<br>DAS | B | Rd decimal adjust → Rd<br>Decimal-adjusts (adjusts to packed BCD) an addition or subtraction result in a general register by referring to the CCR. |
| MULXU | B | Rd × Rs → Rd<br>Performs 8-bit × 8-bit unsigned multiplication on data in two general registers, providing a 16-bit result. |
| DIVXU | B | Rd ÷ Rs → Rd<br>Performs 16-bit ÷ 8-bit unsigned division on data in two general registers, providing an 8-bit quotient and 8-bit remainder. |
| CMP | B/W | Rd – Rs, Rd – #imm<br>Compares data in a general register with data in another general register or with immediate data. Word data can be compared only between two general registers. |
| NEG | B | 0 – Rd → Rd<br>Obtains the two’s complement (arithmetic complement) of data in a general register. |

* Size: operand size
B: Byte
W: Word

### 3.5.3 Logic Operations

Table 3-4 describes the four instructions that perform logic operations. See figure 3-6 in section 3.5.4, “Shift Operations” for their object codes.

**Table 3-4. Logic Operation Instructions**

| Instruction | Size* | Function |
|---|---|---|
| AND | B | Rd $\wedge$ Rs $\rightarrow$ Rd, Rd $\wedge$ #imm $\rightarrow$ Rd<br>Performs a logical AND operation on a general register and another general register or immediate data. |
| OR | B | Rd $\vee$ Rs $\rightarrow$ Rd, Rd $\vee$ #imm $\rightarrow$ Rd<br>Performs a logical OR operation on a general register and another general register or immediate data. |
| XOR | B | Rd $\oplus$ Rs $\rightarrow$ Rd, Rd $\oplus$ #imm $\rightarrow$ Rd<br>Performs a logical exclusive OR operation on a general register and another general register or immediate data. |
| NOT | B | $\neg$ (Rd) $\rightarrow$ (Rd)<br>Obtains the one’s complement (logical complement) of general register contents. |

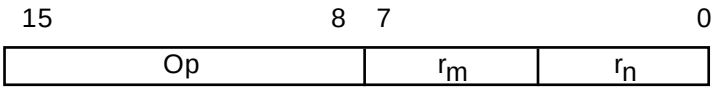
### 3.5.4 Shift Operations

Table 3-5 describes the eight shift instructions. Figure 3-6 shows the object code formats of the arithmetic, logic, and shift instructions.

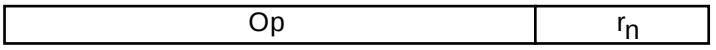
**Table 3-5. Shift Instructions**

| Instruction | Size* | Function |
|---|---|---|
| SHAL<br>SHAR | B | Rd shift $\rightarrow$ Rd<br>Performs an arithmetic shift operation on general register contents. |
| SHLL<br>SHLR | B | Rd shift $\rightarrow$ Rd<br>Performs a logical shift operation on general register contents. |
| ROTL<br>ROTR | B | Rd rotate $\rightarrow$ Rd<br>Rotates general register contents. |
| ROTXL<br>ROTXR | B | Rd rotate through carry $\rightarrow$ Rd<br>Rotates general register contents through the C (carry) bit. |

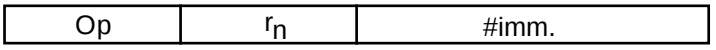
* Size: operand size
B: Byte



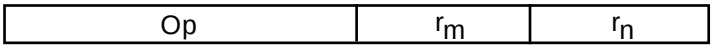

ADD, SUB, CMP
ADDX, SUBX, MULXU, DIVXU



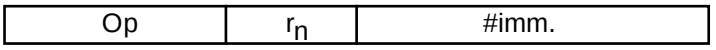

ADDS, SUBS, INC, DEC, DAA, DAS, NEG, NOT



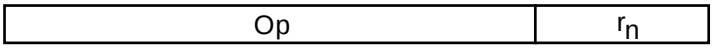

ADD, ADDX, SUBX, CMP (#xx:8)




AND, OR, XOR (Rm)




AND, OR, XOR (#xx:8)




SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR

**Notation**

| | |
|---|---|
| Op: | Operation field |
| $r_m$, $r_n$: | Register field |
| #imm.: | Immediate data |

**Figure 3-6. Arithmetic, Logic, and Shift Instruction Codes**

### 3.5.5 Bit Manipulations

Table 3-6 describes the bit-manipulation instructions. Figure 3-7 shows their object code formats.

**Table 3-6. Bit-Manipulation Instructions (1)**

| Instruction | Size* | Function |
|---|---|---|
| BSET | B | 1 → (<bit-No.> of <EAd>)<br>Sets a specified bit in a general register or memory to "1." The bit is specified by a bit number, given in 3-bit immediate data or the lower three bits of a general register. |
| BCLR | B | 0 → (<bit-No.> of <EAd>)<br>Clears a specified bit in a general register or memory to "0." The bit is specified by a bit number, given in 3-bit immediate data or the lower three bits of a general register. |
| BNOT | B | ¬ (<bit-No.> of <EAd>) → (<bit-No.> of <EAd>)<br>Inverts a specified bit in a general register or memory. The bit is specified by a bit number, given in 3-bit immediate data or the lower three bits of a general register |
| BTST | B | ¬ (<bit-No.> of <EAd>) → Z<br>Tests a specified bit in a general register or memory and sets or clears the Z flag accordingly. The bit is specified by a bit number, given in 3-bit immediate data or the lower three bits of a general register. |
| BAND | B | C ∧ (<bit-No.> of <EAd>) → C<br>ANDs the C flag with a specified bit in a general register or memory. |
| BIAND | | C ∧ [¬ (<bit-No.> of <EAd>)] → C<br>ANDs the C flag with the inverse of a specified bit in a general register or memory.<br>The bit number is specified by 3-bit immediate data. |
| BOR | B | C ∨ (<bit-No.> of <EAd>) → C<br>ORs the C flag with a specified bit in a general register or memory. |
| BIOR | | C ∨ [¬ (<bit-No.> of <EAd>)] → C<br>ORs the C flag with the inverse of a specified bit in a general register or memory.<br>The bit number is specified by 3-bit immediate data. |
| BXOR | B | C ⊕ (<bit-No.> of <EAd>) → C<br>XORs the C flag with a specified bit in a general register or memory. |

* Size: operand size
B: Byte

**Table 3-6. Bit-Manipulation Instructions (2)**

| Instruction | Size* | Function |
|---|---|---|
| BIXOR | B | C ⊕ ¬ [(<bit-No.> of <EAd>)] → C<br>XORs the C flag with the inverse of a specified bit in a general register or memory.<br>The bit number is specified by 3-bit immediate data. |
| BLD | B | (<bit-No.> of <EAd>) → C<br>Copies a specified bit in a general register or memory to the C flag. |
| BILD | | ¬ (<bit-No.> of <EAd>) → C<br>Copies the inverse of a specified bit in a general register or memory to the C flag.<br>The bit number is specified by 3-bit immediate data. |
| BST | B | C → (<bit-No.> of <EAd>)<br>Copies the C flag to a specified bit in a general register or memory. |
| BIST | | ¬ C → (<bit-No.> of <EAd>)<br>Copies the inverse of the C flag to a specified bit in a general register or memory.<br>The bit number is specified by 3-bit immediate data. |

* Size: operand size
B: Byte

**Notes on Bit Manipulation Instructions:** BSET, BCLR, BNOT, BST, and BIST are read-modify-write instructions. They read a byte of data, modify one bit in the byte, then write the byte back. Care is required when these instructions are applied to registers with write-only bits and to the I/O port registers.

| | |
|---|---|
| Read | Read one data byte at the specified address |
| Modify | Modify one bit in the data byte |
| Write | Write the modified data byte back to the specified address |

**Example 1:** BCLR is executed to clear bit 0 in the port 4 data direction register (P4DDR) under the following conditions.

$P4_7$: Input pin, Low, MOS pull-up transistor on

$P4_6$: Input pin, High, MOS pull-up transistor off

$P4_5$ – $P4_0$: Output pins, Low

The intended purpose of this BCLR instruction is to switch $P4_0$ from output to input.

**Before Execution of BCLR Instruction**

| | **$P4_7$** | **$P4_6$** | **$P4_5$** | **$P4_4$** | **$P4_3$** | **$P4_2$** | **$P4_1$** | **$P4_0$** |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low | High | Low | Low | Low | Low | Low | Low |
| DDR | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| DR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Pull-up Mos | On | Off | Off | Off | Off | Off | Off | Off |

**Execution of BCLR Instruction**

```
BCLR.B  #0, @P4DDR        ;clear bit 0 in data direction register
```

**After Execution of BCLR Instruction**

| | **$P4_7$** | **$P4_6$** | **$P4_5$** | **$P4_4$** | **$P4_3$** | **$P4_2$** | **$P4_1$** | **$P4_0$** |
|---|---|---|---|---|---|---|---|---|
| Input/output | Output | Output | Output | Output | Output | Output | Output | Input |
| Pin state | Low | High | Low | Low | Low | Low | Low | High |
| DDR | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| DR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Pull-up Mos | Off | Off | Off | Off | Off | Off | Off | Off |

**Explanation:** To execute the BCLR instruction, the CPU begins by reading P4DDR. Since P4DDR is a write-only register, it is read as H'FF, even though its true value is H'3F.

Next the CPU clears bit 0 of the read data, changing the value to H'FE.

Finally, the CPU writes this value (H'FE) back to P4DDR to complete the BCLR instruction.

As a result, $P4_0$DDR is cleared to "0," making $P4_0$ an input pin. In addition, $P4_7$DDR and $P4_6$DDR are set to "1," making $P4_7$ and $P4_6$ output pins.

**Example 2:** BSET is executed to set bit 0 in the port 4 data register (P4DR) under the following conditions.

$P4_7$: Input pin, Low, MOS pull-up transistor on

$P4_6$: Input pin, High, MOS pull-up transistor off

$P4_5$ – $P4_0$: Output pins, Low

The intended purpose of this BSET instruction is to switch the output level at $P4_0$ from Low to High.

**Before Execution of BSET Instruction**

| | **$P4_7$** | **$P4_6$** | **$P4_5$** | **$P4_4$** | **$P4_3$** | **$P4_2$** | **$P4_1$** | **$P4_0$** |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low | High | Low | Low | Low | Low | Low | Low |
| DDR | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| DR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Pull-up Mos | On | Off | Off | Off | Off | Off | Off | Off |

**Execution of BSET Instruction**

```
BSET.B  #0, @PORT4        ;set bit 0 in data register
```

**After Execution of BSET Instruction**

| | **$P4_7$** | **$P4_6$** | **$P4_5$** | **$P4_4$** | **$P4_3$** | **$P4_2$** | **$P4_1$** | **$P4_0$** |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low | High | Low | Low | Low | Low | Low | High |
| DDR | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| DR | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Pull-up | Off | On | Off | Off | Off | Off | Off | Off |

**Explanation:** To execute the BSET instruction, the CPU begins by reading port 4. Since $P4_7$ and $P4_6$ are input pins, the CPU reads the level of these pins directly, not the value in the data register. It reads $P4_7$ as Low ("0") and $P4_6$ as High ("1").

Since $P4_5$ to $P4_0$ are output pins, for these pins the CPU reads the value in the data register ("0"). The CPU therefore reads the value of port 4 as H'40, although the actual value in P4DR is H'80.

Next the CPU sets bit 0 of the read data to "1," changing the value to H'41.

Finally, the CPU writes this value (H'41) back to P4DR to complete the BSET instruction.

As a result, bit $P4_0$ is set to "1," switching pin $P4_0$ to High output. In addition, bits $P4_7$ and $P4_6$ are both modified, changing the on/off settings of the MOS pull-up transistors of pins $P4_7$ and $P4_6$.

**Programming Solution:** The switching of the pull-ups for $P4_7$ and $P4_6$ in example 2 can be avoided by reserving a byte in RAM as a temporary register for P4DR and using it as follows. RAM0 is a symbol for the user-selected address of the temporary register.

**Before Execution of BSET Instruction**

```
MOV.B  #80, R0L          ;write data (H'80) for data register
MOV.B  R0L, @RAM0        ;write to DR temporary register (RAM0)
MOV.B  R0L, @PORT4       ;write to DR
```

| | **$P4_7$** | **$P4_6$** | **$P4_5$** | **$P4_4$** | **$P4_3$** | **$P4_2$** | **$P4_1$** | **$P4_0$** |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low | High | Low | Low | Low | Low | Low | Low |
| DDR | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| DR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Pull-up Mos | On | Off | Off | Off | Off | Off | Off | Off |
| RAM0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

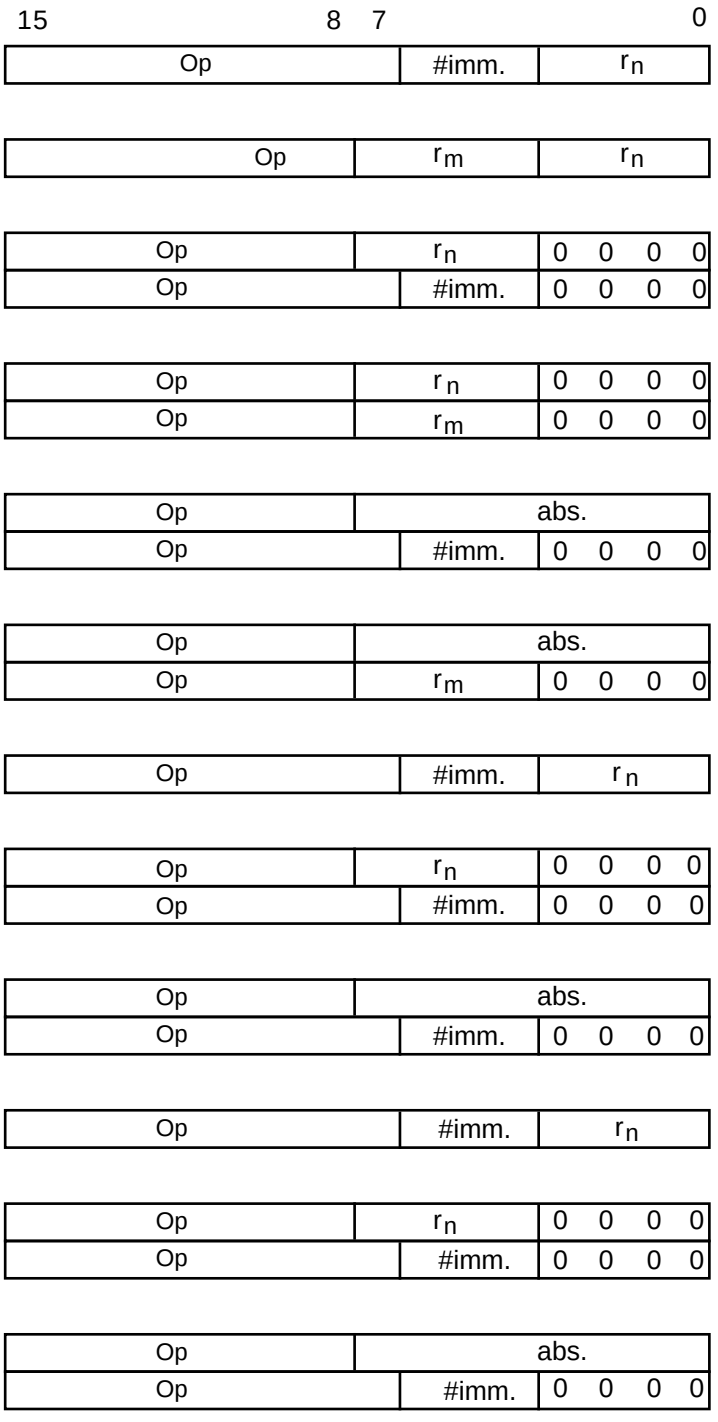
**Execution of BSET Instruction**

```
BSET.B  #0, @RAM0        ;set bit 0 in DR temporary register (RAM0)
```

**After Execution of BSET Instruction**

```
MOV.B  @RAM0, R0L        ;obtain value of temporary register RAM0
MOV.B  R0L,   @PORT4     ;write value to DR
```

| | **$P4_7$** | **$P4_6$** | **$P4_5$** | **$P4_4$** | **$P4_3$** | **$P4_2$** | **$P4_1$** | **$P4_0$** |
|---|---|---|---|---|---|---|---|---|
| Input/output | Input | Input | Output | Output | Output | Output | Output | Output |
| Pin state | Low | High | Low | Low | Low | Low | Low | High |
| DDR | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| DR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Pull-up Mos | On | Off | Off | Off | Off | Off | Off | Off |
| RAM0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |




**Notation**

| | |
|---|---|
| Op: | Operation field |
| $r_m$, $r_n$: | Register field |
| abs.: | Absolute address |
| #imm.: | Immediate data |

**Figure 3-7. Bit Manipulation Instruction Codes**

### 3.5.6 Branching Instructions

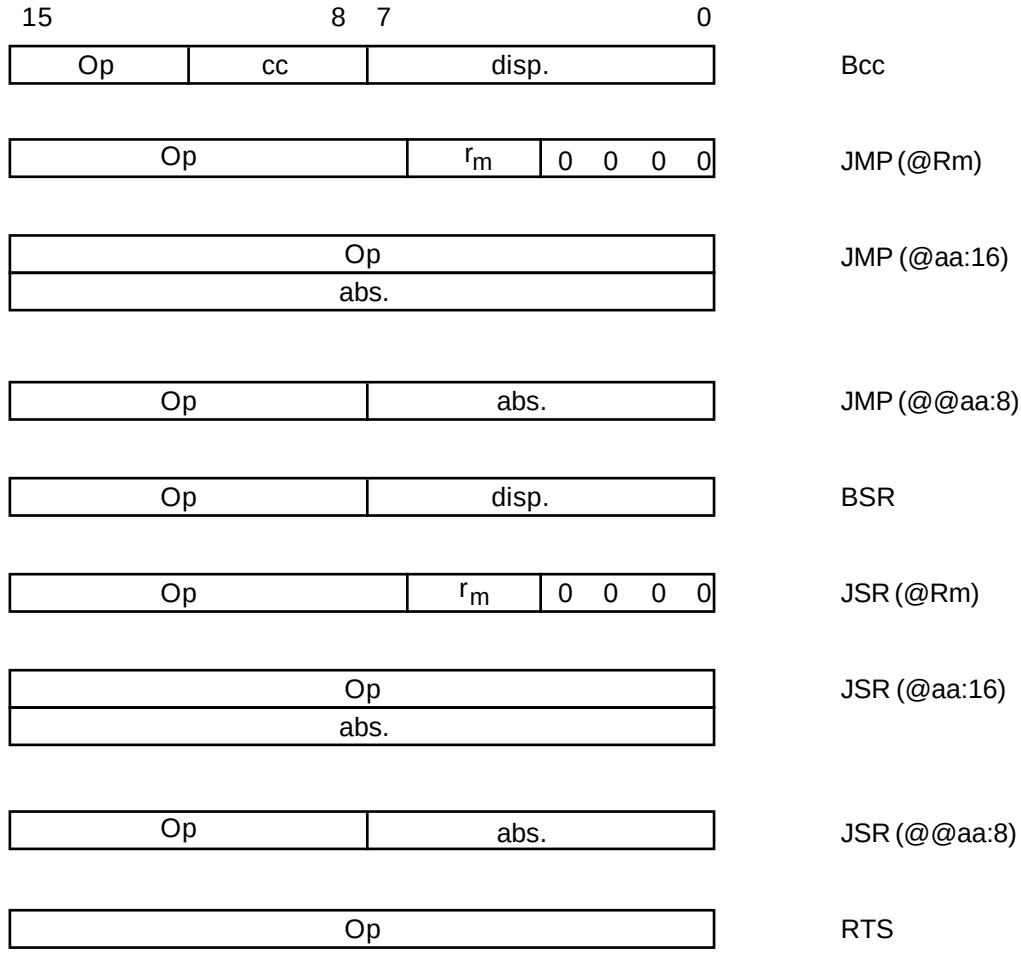
Table 3-7 describes the branching instructions. Figure 3-8 shows their object code formats.

**Table 3-7. Branching Instructions**

| Instruction | Size | Function |
|---|---|---|
| Bcc | — | Branches if condition cc is true. |

| Mnemonic | cc Field | Description | Condition |
|---|---|---|---|
| BRA (BT) | 0 0 0 0 | Always (True) | Always |
| BRN (BF) | 0 0 0 1 | Never (False) | Never |
| BHI | 0 0 1 0 | High | $C \vee Z = 0$ |
| BLS | 0 0 1 1 | Low or Same | $C \vee Z = 1$ |
| BCC (BHS) | 0 1 0 0 | Carry Clear (High or Same) | C = 0 |
| BCS (BLO) | 0 1 0 1 | Carry Set (Low) | C = 1 |
| BNE | 0 1 1 0 | Not Equal | Z = 0 |
| BEQ | 0 1 1 1 | Equal | Z = 1 |
| BVC | 1 0 0 0 | Overflow Clear | V = 0 |
| BVS | 1 0 0 1 | Overflow Set | V = 1 |
| BPL | 1 0 1 0 | Plus | N = 0 |
| BMI | 1 0 1 1 | Minus | N = 1 |
| BGE | 1 1 0 0 | Greater or Equal | $N \oplus V = 0$ |
| BLT | 1 1 0 1 | Less Than | $N \oplus V = 1$ |
| BGT | 1 1 1 0 | Greater Than | $Z \vee (N \oplus V) = 0$ |
| BLE | 1 1 1 1 | Less or Equal | $Z \vee (N \oplus V) = 1$ |

| | | |
|---|---|---|
| JMP | — | Branches unconditionally to a specified address. |
| JSR | — | Branches to a subroutine at a specified address. |
| BSR | — | Branches to a subroutine at a specified displacement from the current address. |
| RTS | — | Returns from a subroutine |




**Notation**

| | |
|---|---|
| Op: | Operation field |
| cc: | Condition field |
| $r_m$: | Register field |
| disp.: | Displacement |
| abs.: | Absolute address |

**Figure 3-8. Branching Instruction Codes**

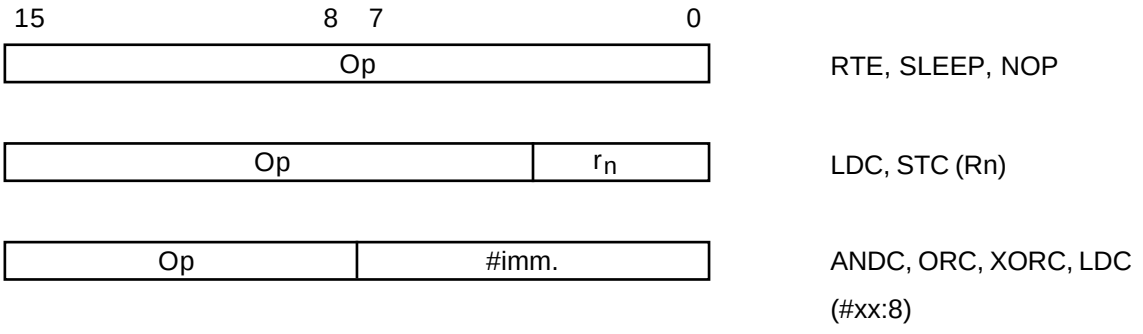
### 3.5.7 System Control Instructions

Table 3-8 describes the system control instructions. Figure 3-9 shows their object code formats.

**Table 3-8. System Control Instructions**

| Instruction | Size | Function |
|---|---|---|
| RTE | — | Returns from an exception-handling routine. |
| SLEEP | — | Causes a transition to the power-down state. |
| LDC | B | Rs → CCR, #imm → CCR<br>Moves immediate data or general register contents to the condition code register. |
| STC | B | CCR → Rd<br>Copies the condition code register to a specified general register. |
| ANDC | B | CCR ∧ #imm → CCR<br>Logically ANDs the condition code register with immediate data. |
| ORC | B | CCR ∨ #imm → CCR<br>Logically ORs the condition code register with immediate data. |
| XORC | B | CCR ⊕ #imm → CCR<br>Logically exclusive-ORs the condition code register with immediate data. |
| NOP | — | PC + 2 → PC<br>Only increments the program counter. |

* Size: operand size
B: Byte




**Notation**

Op: Operation field
$r_n$: Register field
#imm.: Immediate data

**Figure 3-9. System Control Instruction Codes**

### 3.5.8 Block Data Transfer Instruction

Table 3-9 describes the EEPMOV instruction. Figure 3-10 shows its object code format.

**Table 3-9. Block Data Transfer Instruction/EEPROM Write Operation**

| Instruction | Size | Function |
|---|---|---|
| EEPMOV | — | if R4L ≠ 0 then<br>repeat @R5+ → @R6+<br>R4L – 1 → R4L<br>until R4L = 0<br>else next;<br>Moves a data block according to parameters set in general registers R4L, R5, and R6.<br>R4L: size of block (bytes)<br>R5: starting source address<br>R6: starting destination address<br>Execution of the next instruction starts as soon as the block transfer is completed. |

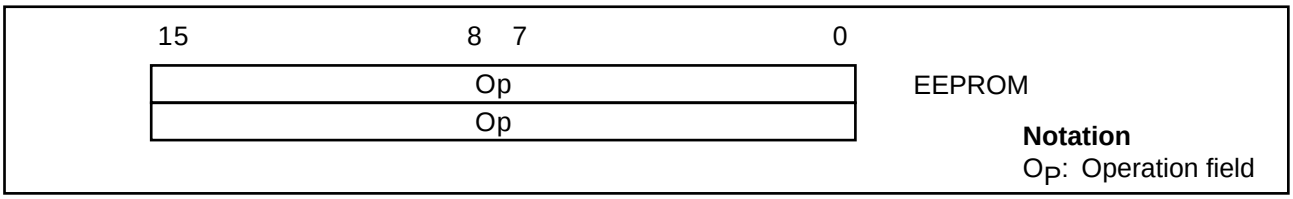
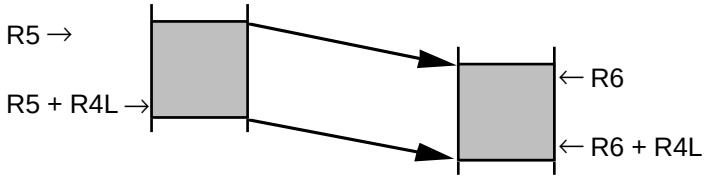



**Figure 3-10. Block Data Transfer Instruction/EEPROM Write Operation Code**

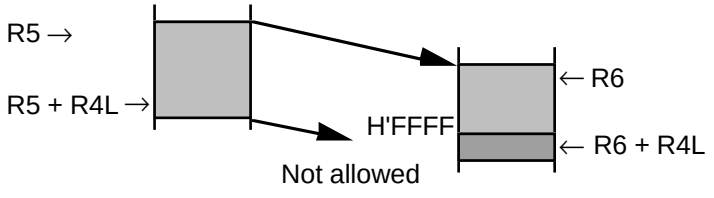
**Notes on EEPMOV Instruction**

**Note 1**

- The EEPMOV instruction is a block data transfer instruction. It moves the number of bytes specified by R4L from the address specified by R5 to the address specified by R6.




- When setting R4L and R6, make sure that the final destination address (R6 + R4L) does not exceed H'FFFF. The value in R6 must not change from H'FFFF to H'0000 during execution of the instruction.




**Note 2**

CPU will malfunction after EEPMOV instruction execution, in the following conditions. EEPMOV instruction performs block data transfer function.

- **Condition**

  When the following conditions are all true:

  — The LSI is set to expanded mode (i.e. mode 1 or mode 2).

  — The destination address of EEPMOV instruction is external area.

  — At least one wait state is inserted to the last write bus cycle to the destination address by EEPMOV instruction.

- **Phenomenon**

  — H8/300 CPU will malfunction after EEPMOV instruction execution.

- **Counter Measures by Software or Circuitry**

  Please take at least one counter measure from the followings.

  — Please use EEPMOV when the destination is in the internal area (e.g. internal RAM).

  — When the destination is the external area, please avoid wait state insertion to the bus cycle.

  — When the case that wait state(s) is required, please substitute EEPMOV by MOV and other instructions as follows:

**Example**

```
LOOP:MOV.B @R5+, R4H
     MOV.B R4H, @R6
     ADDS  #1,   R6
     INC   R4L
     BNE   LOOP
```

## 3.6 CPU States

The CPU has three states: the program execution state, exception-handling state, and power-down state. The power-down state is further divided into three modes: the sleep mode, software standby mode, and hardware standby mode. Figure 3-11 summarizes these states, and figure 3-12 shows a map of the state transitions.

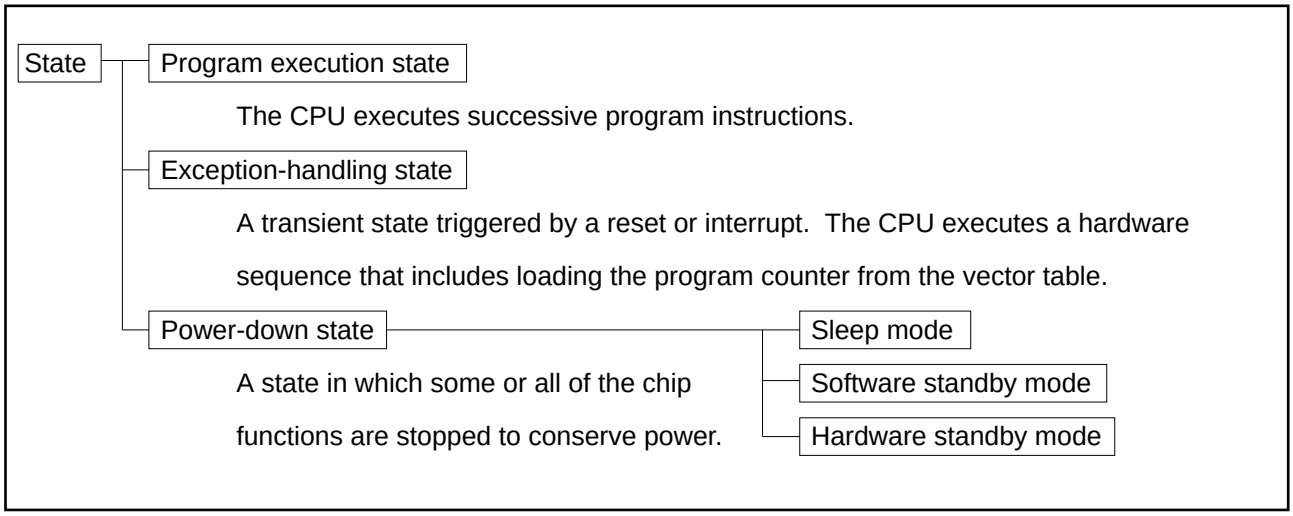
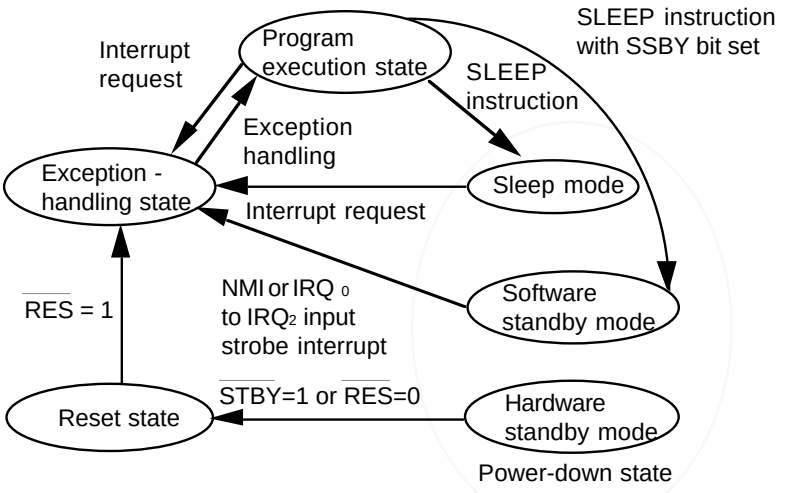



**Figure 3-11. Operating States**




Notes:

1. A transition to the reset state occurs when $\overline{RES}$ goes Low, except when the chip is in the hardware standby mode.
2. A transition from any state to the hardware standby mode occurs when $\overline{STBY}$ goes Low.

**Figure 3-12. State Transitions**

### 3.6.1 Program Execution State

In this state the CPU executes program instructions in sequence. The main program, subroutines, and interrupt-handling routines are all executed in this state.

### 3.6.2 Exception-Handling State

The exception-handling state is a transient state that occurs when the CPU is reset or accepts an interrupt. In this state the CPU carries out a hardware-controlled sequence that prepares it to execute a user-coded exception-handling routine.

In the hardware exception-handling sequence the CPU does the following:

(1) Saves the program counter and condition code register to the stack (except in the case of a reset).
(2) Sets the interrupt mask (I) bit in the condition code register to “1.”
(3) Fetches the start address of the exception-handling routine from the vector table.
(4) Branches to that address, returning to the program execution state.

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

### 3.6.3 Power-Down State

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

**(1) Sleep Mode:** The sleep mode is entered when a SLEEP instruction is executed. The CPU halts, but CPU register contents remain unchanged and the on-chip supporting modules continue to function.

When an interrupt or reset signal is received, the CPU returns through the exception-handling state to the program execution state.

**(2) Software Standby Mode:** The software standby mode is entered if the SLEEP instruction is executed while the SSBY (Software Standby) bit in the system control register (SYSCR) is set. The CPU and all on-chip supporting modules halt. The on-chip supporting modules are initialized, but the contents of the on-chip RAM and CPU registers remain unchanged. I/O port outputs also remain unchanged.

**(3) Hardware Standby Mode:** The hardware standby mode is entered when the input at the $\overline{\text{STBY}}$ pin goes Low. All chip functions halt, including I/O port output. The on-chip supporting modules are initialized, but on-chip RAM contents are held.

See section 12, “Power-Down State” for further information.

## 3.7 Access Timing and Bus Cycle

The CPU is driven by the system clock (Ø). The period from one rising edge of the system clock to the next is referred to as a “state.”

Memory access is performed in a two-or three-state bus cycle as described below. For more detailed timing diagrams of the bus cycles, see section 15, “Electrical Specifications.”

### 3.7.1 Access to On-Chip Memory (RAM and ROM)

On-chip ROM and RAM are accessed in a cycle of two states designated $T_1$ and $T_2$. Either byte or word data can be accessed, via a 16-bit data bus. Figure 3-13 shows the on-chip memory access cycle. Figure 3-14 shows the associated pin states.

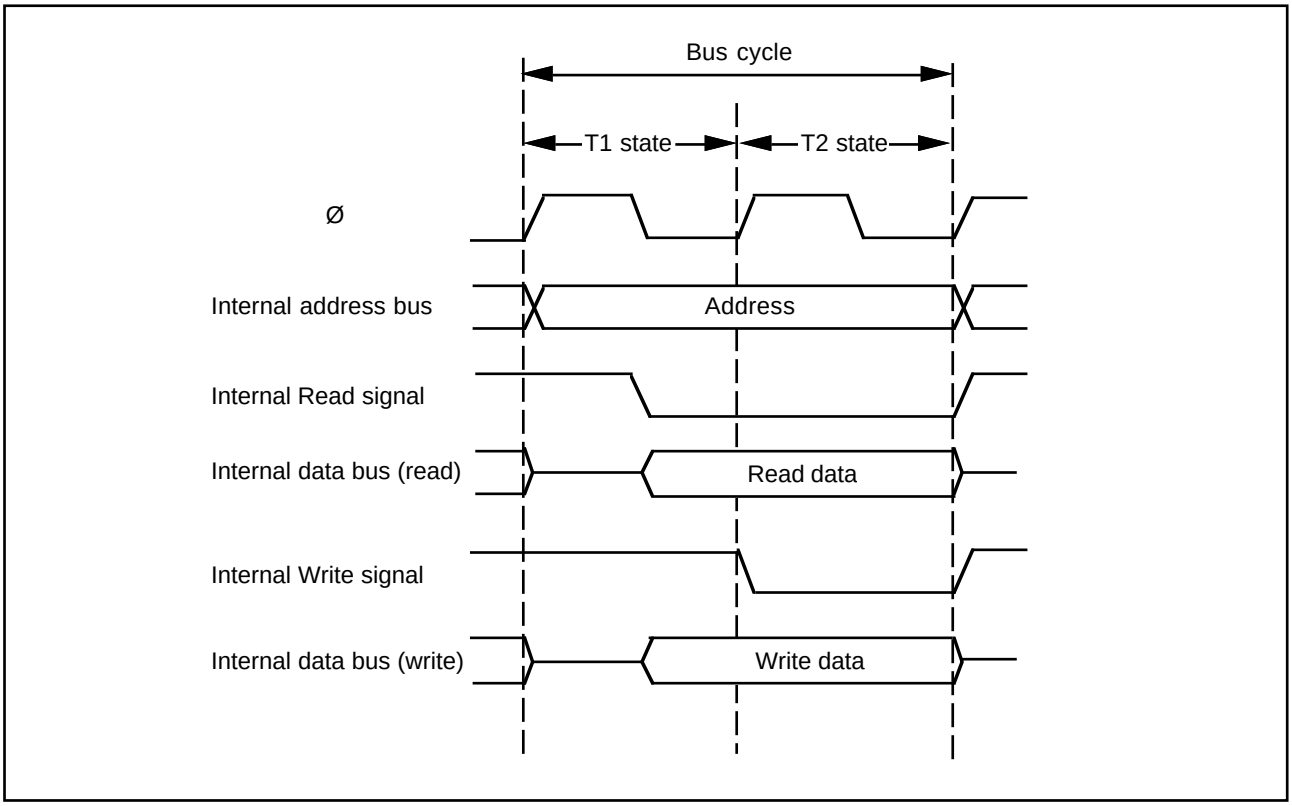



**Figure 3-13. On-Chip Memory Access Cycle**

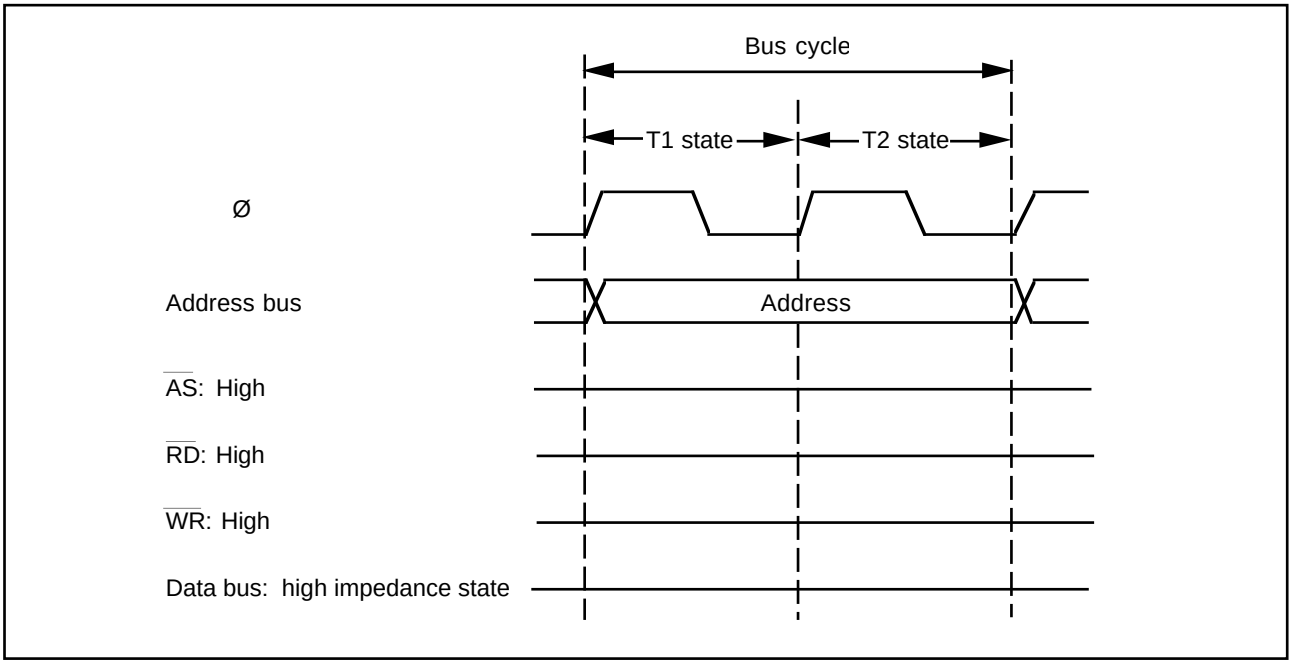



**Figure 3-14. Pin States during On-Chip Memory Access Cycle**

### 3.7.2 Access to On-Chip Register Field and External Devices

The on-chip register field (I/O ports, dual-port RAM, on-chip supporting module registers, etc.) and external devices are accessed in a cycle consisting of three states: $T_1$, $T_2$, and $T_3$. Only one byte of data can be accessed per cycle, via an 8-bit data bus. Access to word data or instruction codes requires two consecutive cycles (six states).

**Wait States:** If requested, additional wait states ($T_W$) are inserted between $T_2$ and $T_3$. The $\overline{\text{WAIT}}$ pin is sampled at the center of state $T_2$. If it is Low, a wait state is inserted after $T_2$. The $\overline{\text{WAIT}}$ pin is also sampled at the center of each wait state and if it is still Low, another wait state is inserted. An external device can have any number of wait states inserted by holding $\overline{\text{WAIT}}$ Low for the necessary duration.

The bus cycle for the MOVTPE and MOVFPE instructions will be described in section 15, "E-Clock Interface."

Figure 3-15 shows the access cycle for the on-chip register field. Figure 3-16 shows the associated pin states. Figures 3-17 (a) and (b) show the read and write access timing for external devices.

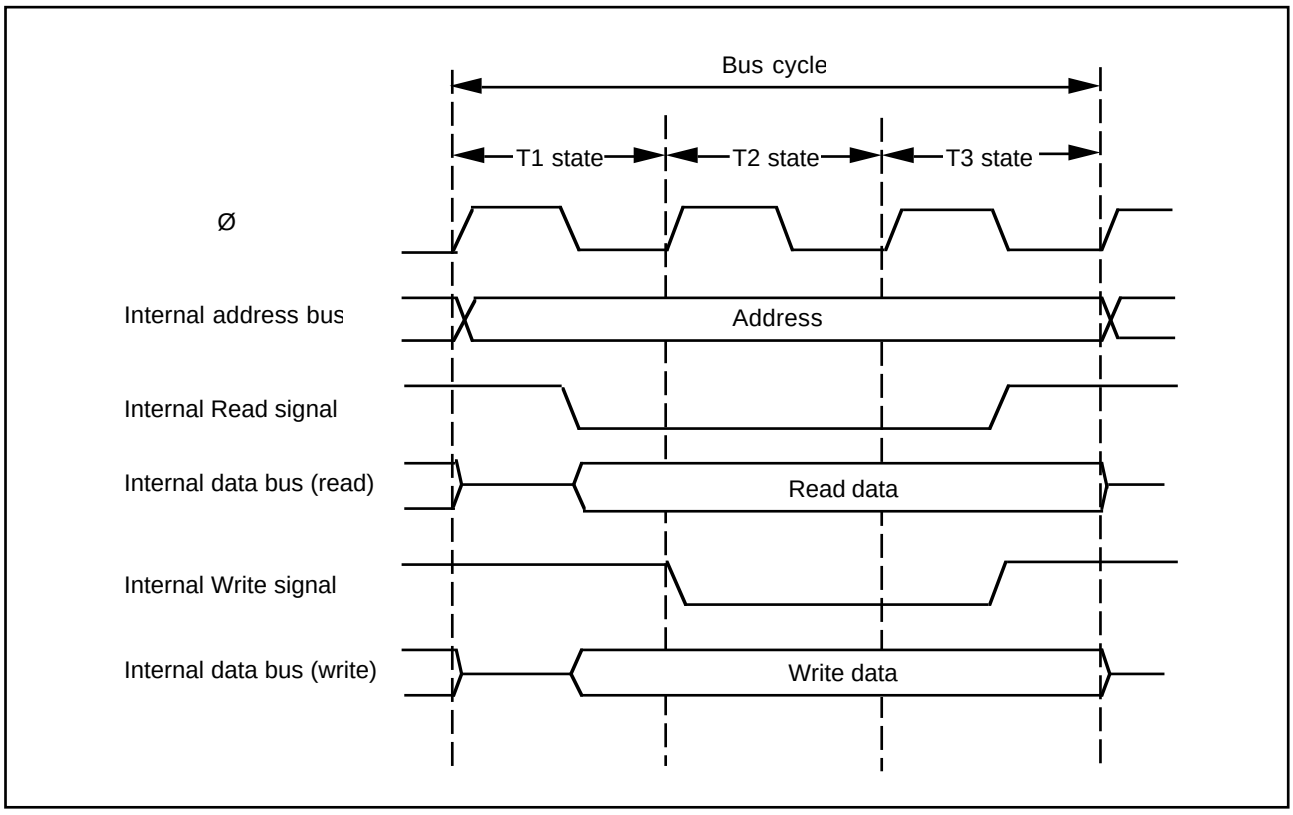



**Figure 3-15. On-Chip Register Field Access Cycle**

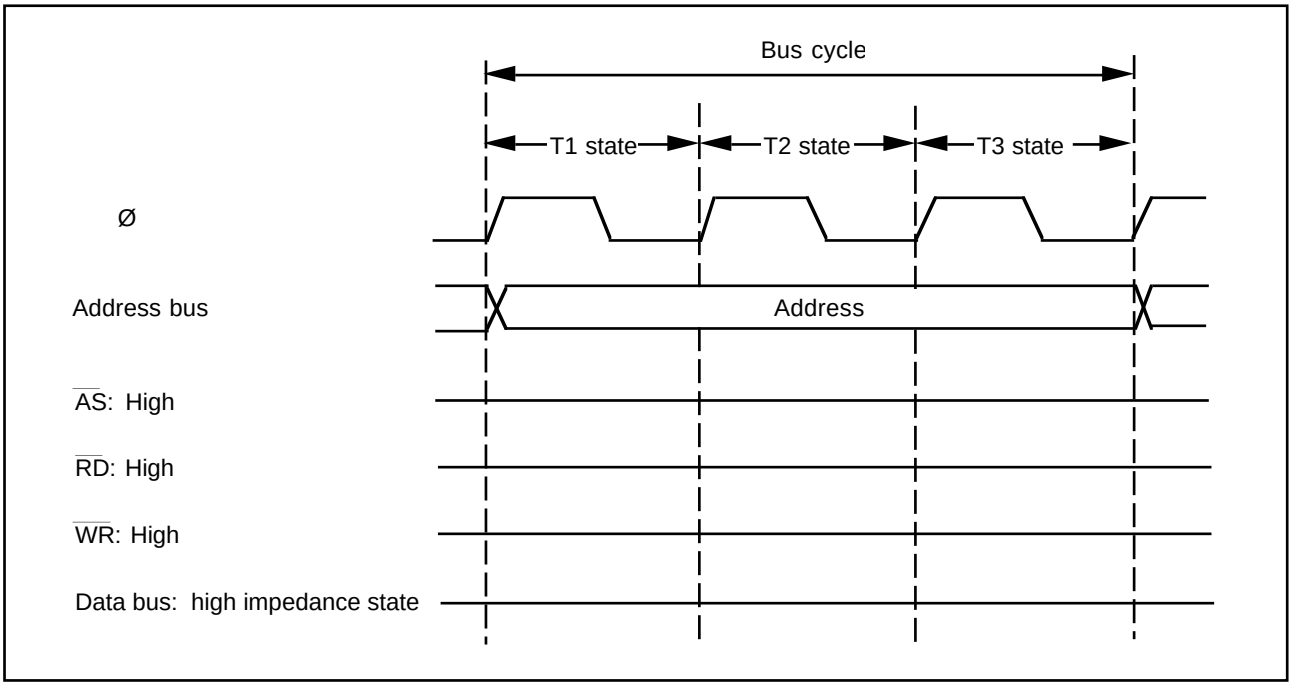



**Figure 3-16. Pin States during On-Chip Register Field Access Cycle**

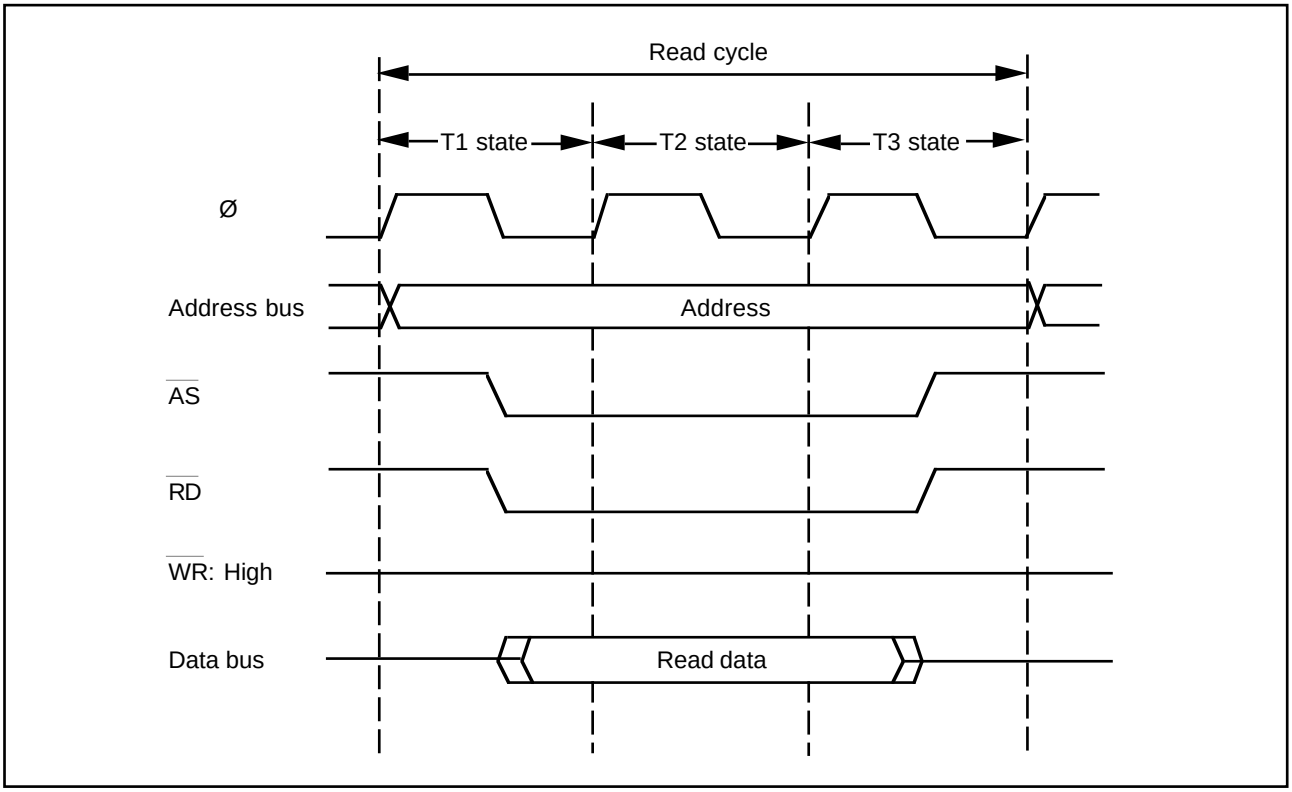



**Figure 3-17 (a). External Device Access Timing (read)**

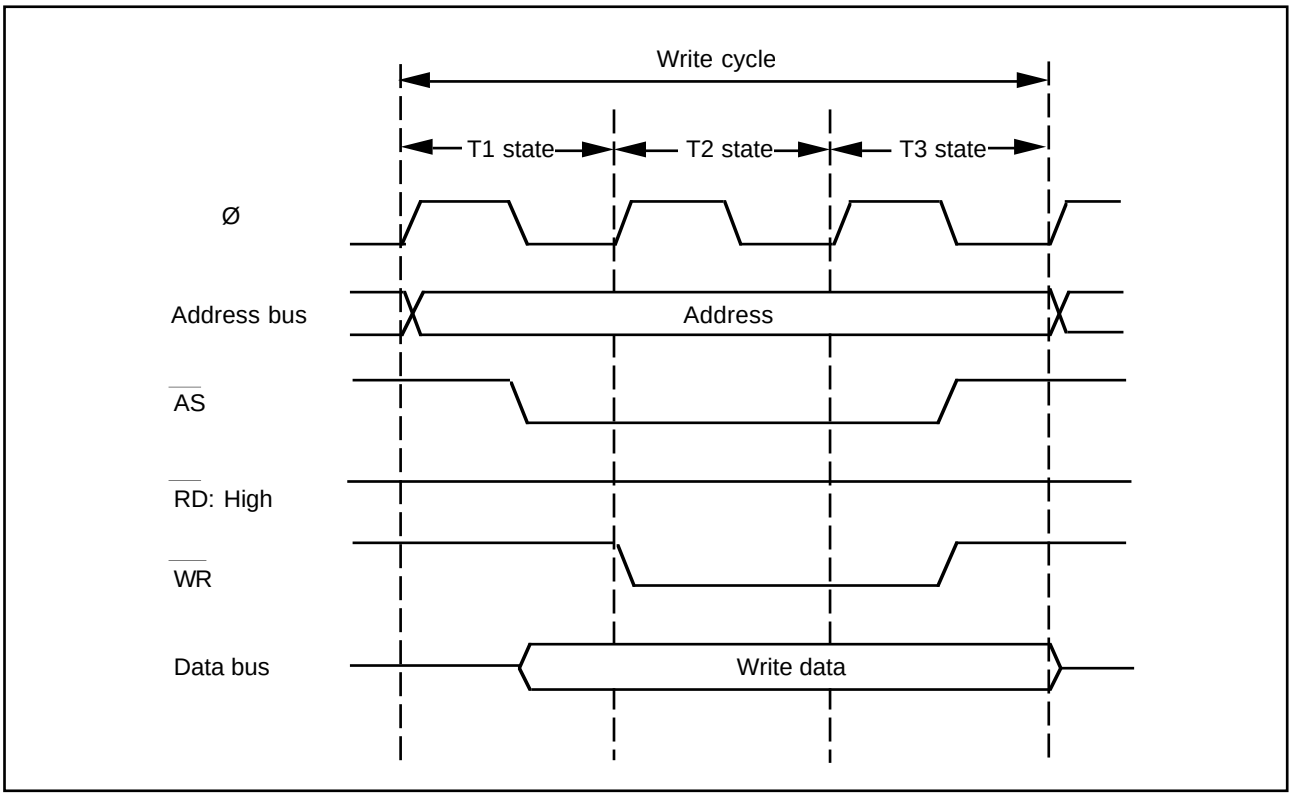



**Figure 3-17 (b). External Device Access Timing (write)**

# Section 4. Exception Handling

## 4.1 Overview

The H8/325 Series recognizes only two kinds of exceptions: interrupts and the reset. Table 4-1 indicates their priority and the timing of their hardware exception-handling sequence. The ROMless versions (HD6413258, HD6413238) are used only in mode 1 (expanded mode with on-chip ROM disabled).

**Table 4-1. Reset and Interrupt Exceptions**

| Priority | Type of exception | Timing of exception-handling sequence |
|---|---|---|
| High | Reset | When $\overline{\text{RES}}$ goes low, the chip enters the reset state immediately. The hardware exception-handling sequence (reset sequence) begins as soon as $\overline{\text{RES}}$ goes high again. |
| ↕ | Interrupt | When an interrupt is requested, the hardware exception-handling sequence (interrupt sequence) begins at the end of the current instruction, or at the end of the current hardware exception-handling sequence. |
| Low | | |

## 4.2 Reset

### 4.2.1 Overview

A reset has the highest exception-handling priority. When the $\overline{\text{RES}}$ pin goes low, all current processing stops and the chip enters the reset state. The internal state of the CPU and the registers of the on-chip supporting modules are initialized. When $\overline{\text{RES}}$ returns from low to high, the chip comes out of the reset state via the reset exception-handling sequence.

### 4.2.2 Reset Sequence

The reset state begins when $\overline{\text{RES}}$ goes low. To ensure correct resetting, at power-on the $\overline{\text{RES}}$ pin should be held low for at least 20ms. In a reset during operation, the $\overline{\text{RES}}$ pin should be held low for at least 10 system clock (Ø) cycles.

When $\overline{\text{RES}}$ returns from low to high, hardware carries out the following reset exception-handling sequence.

(1) The value at the mode pins ($MD_1$ and $MD_0$) is latched in bits MDS1 and MDS0 of the mode control register (MDCR).
(2) In the condition code register (CCR), the I bit is set to 1 to mask interrupts.
(3) The registers of the I/O ports and on-chip supporting modules are initialized.
(4) The CPU loads the program counter with the first word in the vector table (stored at addresses H'0000 and H'0001) and starts program execution.

The $\overline{RES}$ pin should be held low when power is switched off, as well as when power is switched on.

Figure 4-1 indicates the timing of the reset sequence when the vector table and reset routine are located in on-chip ROM. Figure 4-2 indicates the timing when they are in off-chip memory.

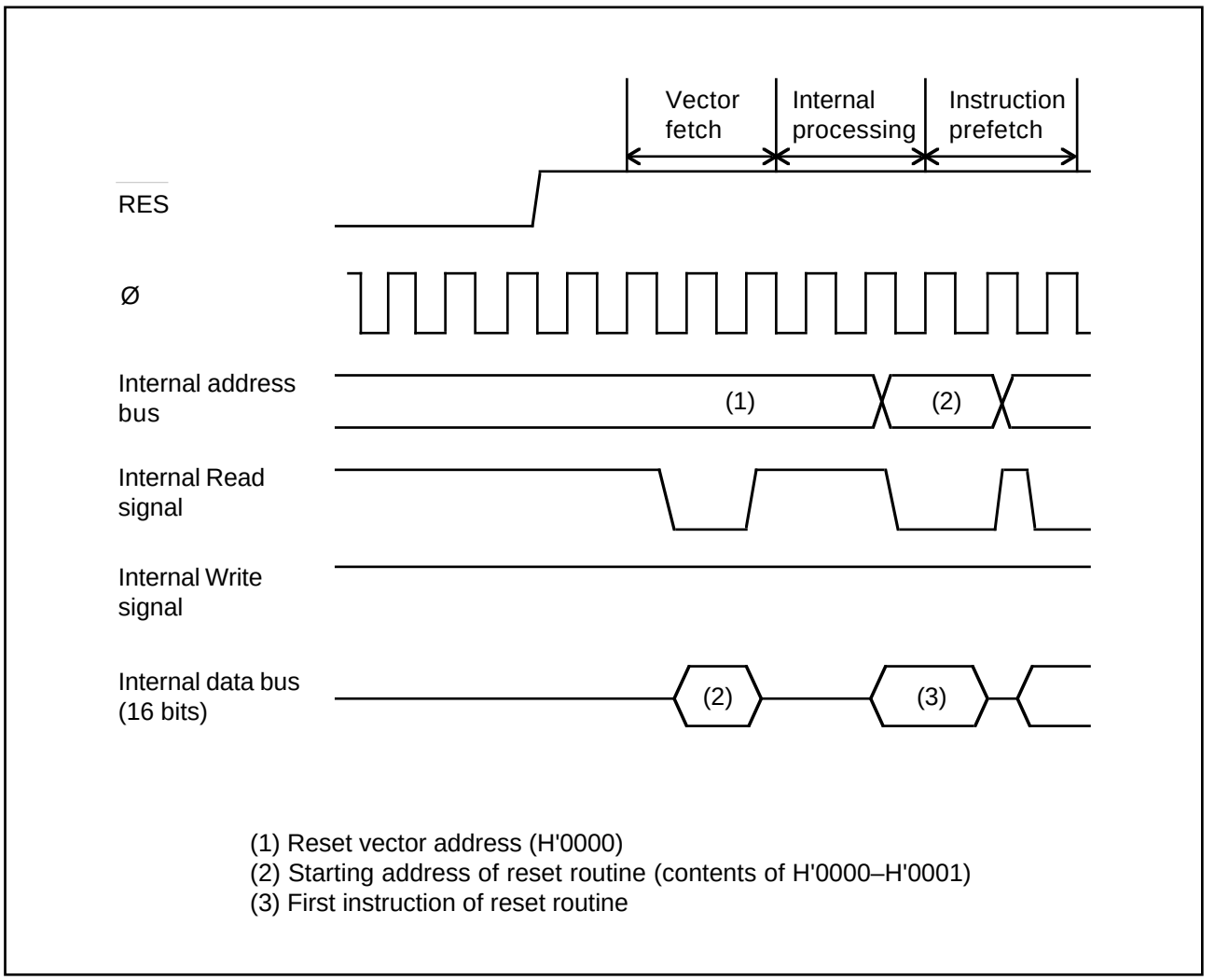



**Figure 4-1. Reset Sequence (Mode 2 or 3, Reset Routine in On-Chip ROM)**

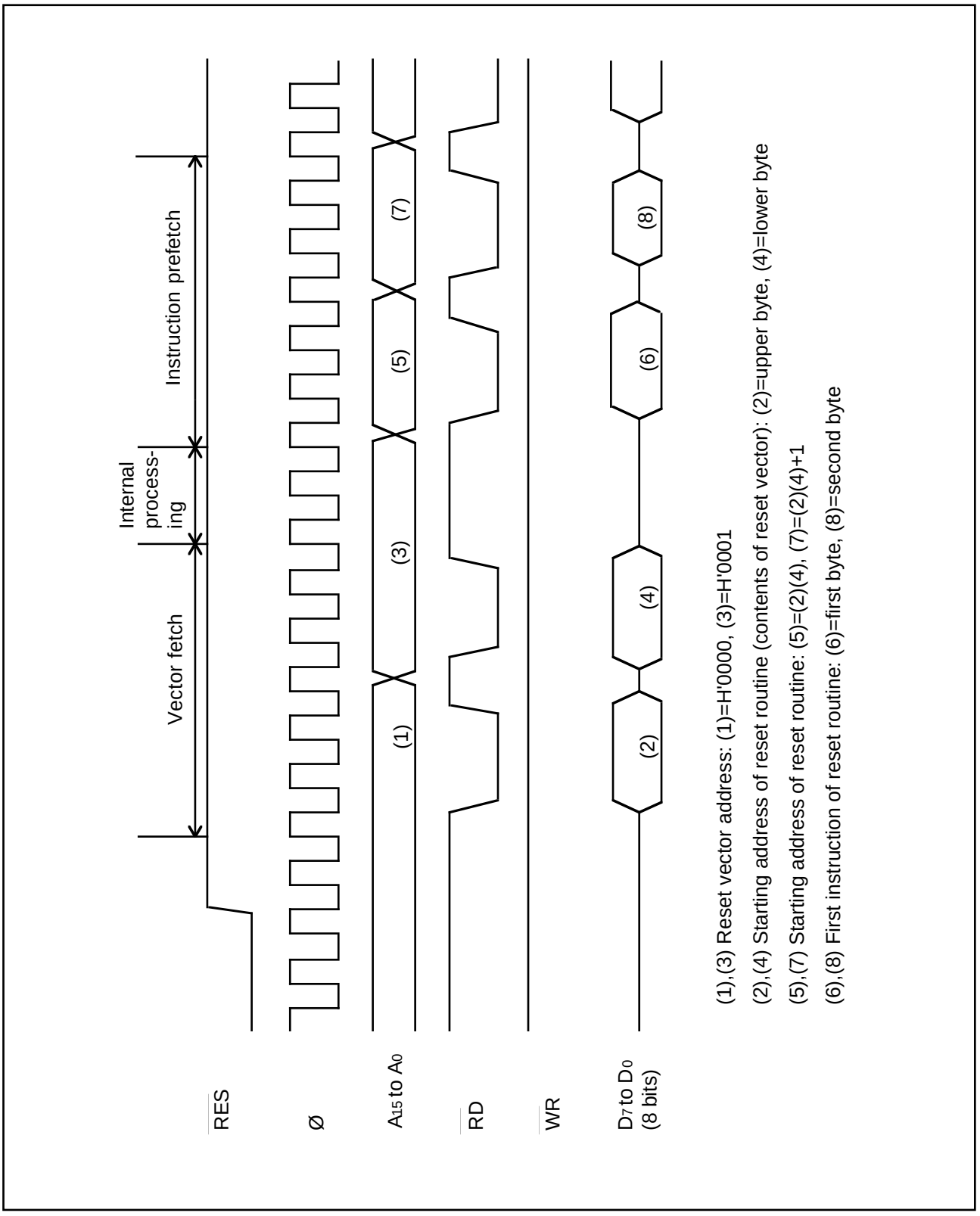



**Figure 4-2. Reset Sequence (Mode 1)**

### 4.2.3 Disabling of Interrupts after Reset

All interrupts, including NMI, are disabled immediately after a reset. The first program instruction, located at the address specified at the top of the vector table, is therefore always executed. To prevent program crashes, this instruction should initialize the stack pointer (example: MOV.W #xx:16, SP). After execution of this instruction, the NMI interrupt is enabled. Other interrupts remain disabled until their enable bits are set to 1.

## 4.3 Interrupts

### 4.3.1 Overview

There are four input pins for external interrupts (NMI, $IRQ_0$ to $IRQ_2$). There are also 17 internal interrupts originating on-chip. The features of these interrupts are:

- All internal and external interrupts except NMI can be masked by the I bit in the CCR.
- $IRQ_0$ to $IRQ_2$ can be rising-edge-sensed, falling-edge-sensed, or level-sensed. The type of sensing can be selected for each interrupt individually. NMI is edge-sensed, and either the rising or falling edge can be selected.
- Interrupts are individually vectored. The software interrupt-handling routine does not have to determine what type of interrupt has occurred.

Table 4-2 lists all the interrupts in their order of priority and gives their vector numbers and the addresses of their entries in the vector table.

**Table 4-2. Interrupts**

| Interrupt source | | No. | Address of entry in vector table | Priority |
|---|---|---|---|---|
| NMI | | 3 | H'0006 – H'0007 | High |
| $IRQ_0$ | | 4 | H'0008 – H'0009 | |
| $IRQ_1$ | | 5 | H'000A – H'000B | |
| $IRQ_2$ | | 6 | H'000C – H'000D | |
| Port | ISI (Input strobe) | 7 | H'000E – H'000F | |
| 16-Bit free- | ICI (Input capture) | 8 | H'0010 – H'0011 | |
| running timer | OCIA (Output compare A) | 9 | H'0012 – H'0013 | |
| | OCIB (Output compare B) | 10 | H'0014 – H'0015 | |
| | FOVI (Overflow) | 11 | H'0016 – H'0017 | |
| 8-Bit timer 0 | CMI0A (Compare-match A) | 12 | H'0018 – H'0019 | |
| | CMI0B (Compare-match B) | 13 | H'001A – H'001B | |
| | OVI0 (Overflow) | 14 | H'001C – H'001D | |
| 8-Bit timer 1 | CMI1A (Compare-match A) | 15 | H'001E – H'001F | |
| | CMI1B (Compare-match B) | 16 | H'0020 – H'0021 | |
| | OVI1 (Overflow) | 17 | H'0022 – H'0023 | |
| Serial | ERI0 (Receive error) | 18 | H'0024 – H'0025 | |
| communication | RXI0 (Receive end) | 19 | H'0026 – H'0027 | |
| interface 0 | TXI0 (Transmit end) | 20 | H'0028 – H'0029 | |
| Serial | ERI1 (Receive error) | 21 | H'002A – H'002B | |
| communication | RXI1 (Receive end) | 22 | H'002C – H'002D | |
| interface 1 | TXI1 (Transmit end) | 23 | H'002E – H'002F | Low |

**Notes:**

1. H'0000 and H'0001 contain the reset vector.
2. H'0002 to H'0005 are reserved in the H8/325 Series and are not available to the user.

### 4.3.2 Interrupt-Related Registers

The interrupt controller refers to three registers in addition to the CCR. The names and attributes of these registers are listed in table 4-3.

**Table 4-3. Registers Read by Interrupt Controller**

| Name | Abbreviation | Read/Write | Address |
|---|---|---|---|
| System control register | SYSCR | R/W | H'FFC4 |
| IRQ sense control register | ISCR | R/W | H'FFC6 |
| IRQ enable register | IER | R/W | H'FFC7 |

### (1) System Control Register (SYSCR)—H'FFC4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSBY | STS2 | STS1 | STS0 | — | NMIEG | — | RAME |
| Initial value | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | — | R/W | — | R/W |

Bit 2 (NMIEG) is the only bit read by the interrupt controller.

**Bit 2—Nonmaskable Interrupt Edge (NMIEG):** Determines whether a nonmaskable interrupt is generated on the falling or rising edge of the $\overline{\text{NMI}}$ input signal.

**Bit 2**

| NMIEG | Description | |
|---|---|---|
| 0 | An interrupt is generated on the falling edge of $\overline{\text{NMI}}$. | (Initial state) |
| 1 | An interrupt is generated on the rising edge of $\overline{\text{NMI}}$. | |

See section 10, RAM and section 12, Power-Down State for information on the other SYSCR bits.

### (2) IRQ Sense Control Register (ISCR)—H'FFC6

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | $IRQ_2EG$ | $IRQ_1EG$ | $IRQ_0EG$ | — | $IRQ_2SC$ | $IRQ_1SC$ | $IRQ_0SC$ |
| Initial value | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Read/Write | — | R/W | R/W | R/W | — | R/W | R/W | R/W |

**Bits 6 and 2—$IRQ_2$ Sense Control ($IRQ_2SC$ and $IRQ_2EG$):** These bits select how the input at the $\overline{IRQ_2}$ pin is sensed.

| Bit 2 IRQ2SC | Bit 6 IRQ2EG | Description |
|---|---|---|
| 0 | 0 | The low level of $\overline{IRQ_2}$ generates an interrupt request. (Initial state) |
| 0 | 1 | |
| 1 | 0 | The falling edge of $\overline{IRQ_2}$ generates an interrupt request. |
| 1 | 1 | The rising edge of $\overline{IRQ_2}$ generates an interrupt request. |

**Bits 5 and 1—IRQ1 Sense Control (IRQ1SC and IRQ1EG):** These bits select how the input at the $\overline{IRQ_1}$ pin is sensed.

| Bit 1 IRQ1SC | Bit 5 IRQ1EG | Description |
|---|---|---|
| 0 | 0 | The low level of $\overline{IRQ_1}$ generates an interrupt request. (Initial state) |
| 0 | 1 | |
| 1 | 0 | The falling edge of $\overline{IRQ_1}$ generates an interrupt request. |
| 1 | 1 | The rising edge of $\overline{IRQ_1}$ generates an interrupt request. |

**Bits 4 and 0—IRQ0 Sense Control (IRQ0SC and IRQ0EG):** These bits select how the input at the $\overline{IRQ_0}$ pin is sensed.

| Bit 0 IRQ0SC | Bit 4 IRQ0EG | Description |
|---|---|---|
| 0 | 0 | The low level of $\overline{IRQ_0}$ generates an interrupt request. (Initial state) |
| 0 | 1 | |
| 1 | 0 | The falling edge of $\overline{IRQ_0}$ generates an interrupt request. |
| 1 | 1 | The rising edge of $\overline{IRQ_0}$ generates an interrupt request. |

### (3) IRQ Enable Register (IER)—H'FFC7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | IRQ2E | IRQ1E | IRQ0E |
| Initial value | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Read/Write | — | — | — | — | — | R/W | R/W | R/W |

**Bits 0 to 2—IRQ0 to IRQ2 Enable (IRQ0E to IRQ2E):** These bits enable or disable the IRQ0, IRQ1, and IRQ2 interrupts individually.

| Bit i (i = 0 to 2) IRQiE | Description | |
|---|---|---|
| 0 | IRQi is disabled. | (Initial state) |
| 1 | IRQi is enabled. | |

Edge-sensed interrupt signals are latched (if enabled) and held until the interrupt is served. They are latched even if the interrupt mask bit (I) is set in the CCR, and even if bits $IRQ_0E$ to $IRQ_2E$ are cleared to 0. Level-sensed interrupts are not latched.

### 4.3.3 External Interrupts

The external interrupts are NMI and $IRQ_0$ to $IRQ_2$.

While the CPU is waiting for one of these interrupts, it is possible to conserve power by entering software standby mode. When the interrupt arrives, the chip will recover automatically to the program execution state, handle the interrupt, then continue executing the main program. See section 12, Power-Down State for further information on software standby mode.

**(1) NMI:** A nonmaskable interrupt is generated on the rising or falling edge of the $\overline{NMI}$ input signal regardless of whether the I (interrupt mask) bit is set in the CCR. The valid edge is selected by the NMIEG bit in the system control register.

An NMI has highest priority and is always accepted as soon as the current instruction ends, unless the current instruction is an ANDC, ORC, XORC, or LDC instruction. When an NMI interrupt is accepted the interrupt mask (I bit) is set, so the NMI handling routine cannot be interrupted except by another NMI.

The NMI vector number is 3. Its entry is located at address H'0006 in the vector table.

**(2) $IRQ_0$ to $IRQ_2$:** These interrupt signals are level-sensed or sensed on the rising or falling edge of the input, as selected by the ISCR bits. These interrupts can be masked collectively by the I bit in the CCR, and can be enabled and disabled individually by setting and clearing the bits in the IRQ enable register. When one of these interrupts is accepted, the I bit is set to 1 to mask further interrupts (except $\overline{NMI}$).

These interrupts are second in priority to NMI. Among them, $IRQ_0$ has the highest priority and $IRQ_2$ the lowest priority. Interrupts $IRQ_0$ to $IRQ_2$ do not depend on whether pins $\overline{IRQ_0}$ to $\overline{IRQ_2}$ are input or output pins. When using external interrupts $IRQ_0$ to $IRQ_2$, clear the corresponding DDR bits to 0 to set these pins to the input state.

### 4.3.4 Internal Interrupts

Seventeen internal interrupts can be requested by the on-chip supporting modules. All of them are masked when the I bit in the CCR is set. In addition, they can all be enabled or disabled by bits in the control registers of the on-chip supporting modules. When one of these interrupts is accepted, the I bit is set to 1 to mask further interrupts (except $\overline{\text{NMI}}$).

Power can be conserved by waiting for an internal interrupt in sleep mode, in which the CPU halts but the on-chip supporting modules continue to run. When the interrupt arrives, the CPU returns to the program-execution state, services the interrupt, then resumes execution of the main program. See section 12, Power-Down State for further information on the sleep mode.

The input strobe interrupt (ISI) can also be waited for in software standby mode. The chip recovers from software standby mode when an input strobe interrupt is requested.

The internal interrupt signals received by the interrupt controller are generated from flag bits in the registers of the on-chip supporting modules. The interrupt controller does not reset these flag bits when accepting the interrupt.

For the vector numbers and priority order of these interrupts, see table 4-2.

**Note:** When disabling internal interrupts, note the following points.

1. Set the interrupt mask (I) to 1 before disabling an internal interrupt or clearing its interrupt flag.
2. If an instruction that disables or clears an internal interrupt is executed while the interrupt mask (I) is cleared to 0, and the interrupt is requested during execution of the instruction, the CPU resolves this conflict as follows:
   ① If one or more other interrupts are also requested, the other interrupt with the highest priority is served.
   ② If no other interrupt is requested, the CPU branches to the reset address.

**Example:** A sample program for disabling the output compare A interrupt is shown below. The OCIAE bit in the TCR should be cleared only when I = 1, as in this example.

```
ORC     #80, CCR        ; Set I bit
BCLR    #5, @TCR        ; Disable output compare A interrupt
ANDC    #7F, CCR        ; Clear I bit
```

**Note:** Interrupt requests are not detected immediately after the ANDC, ORC, XORC, and LDC instructions.

### 4.3.5 Interrupt Handling

Figure 4-3 shows a block diagram of the interrupt controller. Figure 4-4 is a flowchart showing the operation of the interrupt controller and the sequence by which an interrupt is accepted. This sequence is outlined below.

(1) The interrupt controller receives an interrupt request signal. Interrupt request signals can be generated by $\overline{\text{NMI}}$ input, or by other interrupt sources if enabled.
(2) When notified of an interrupt, the interrupt controller scans the interrupt signals in priority order and selects the one with the highest priority. (See table 4-2 for the priority order.) Other requested interrupts remain pending.
(3) The interrupt controller accepts the interrupt if it is an NMI, or if it is another interrupt and the I bit in the CCR is cleared to 0. If the interrupt is not an NMI and the I bit is set to 1, the interrupt is held pending.
(4) When an interrupt is accepted, after completion of the current instruction, first the PC then the CCR is pushed onto the stack. See figure 4-5. The stacked PC indicates the address of first instruction executed after return from the interrupt-handling routine.
(5) The interrupt controller sets the I bit in the CCR to 1, masking all further interrupts except NMI during the interrupt-handling routine.
(6) The interrupt controller generates the vector address of the interrupt and loads the word at this address into the program counter.

The timing of this sequence is shown in figure 4-6 for the case in which the program and vector table are in on-chip ROM and the stack is in on-chip RAM.

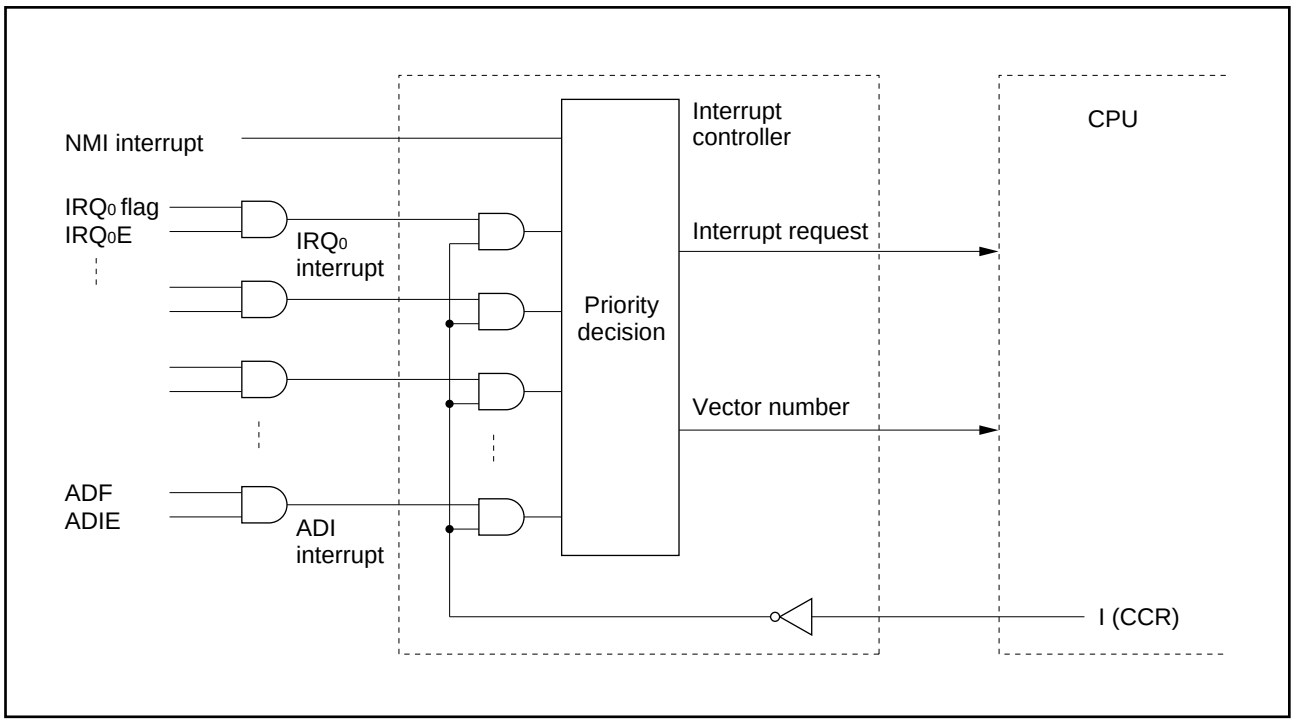



**Figure 4-3. Block Diagram of Interrupt Controller**

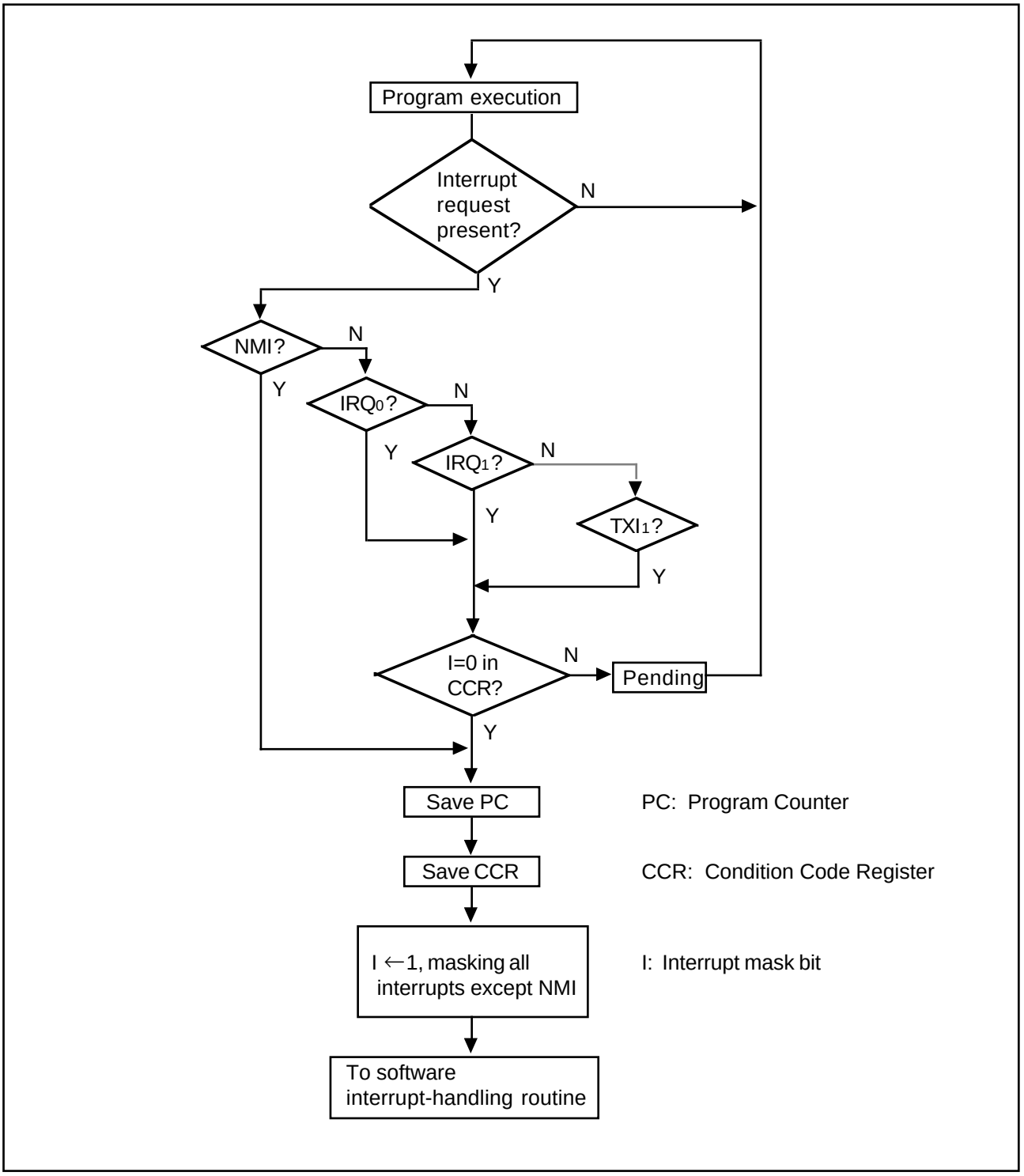
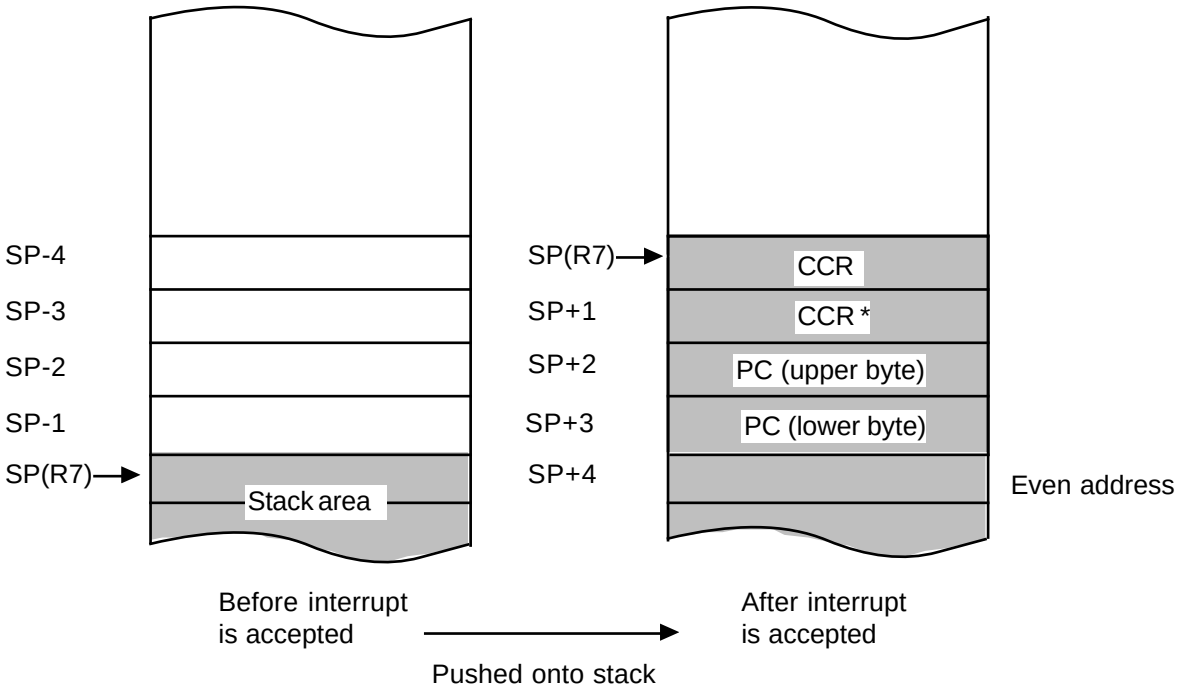



**Figure 4-4. Hardware Interrupt-Handling Sequence**



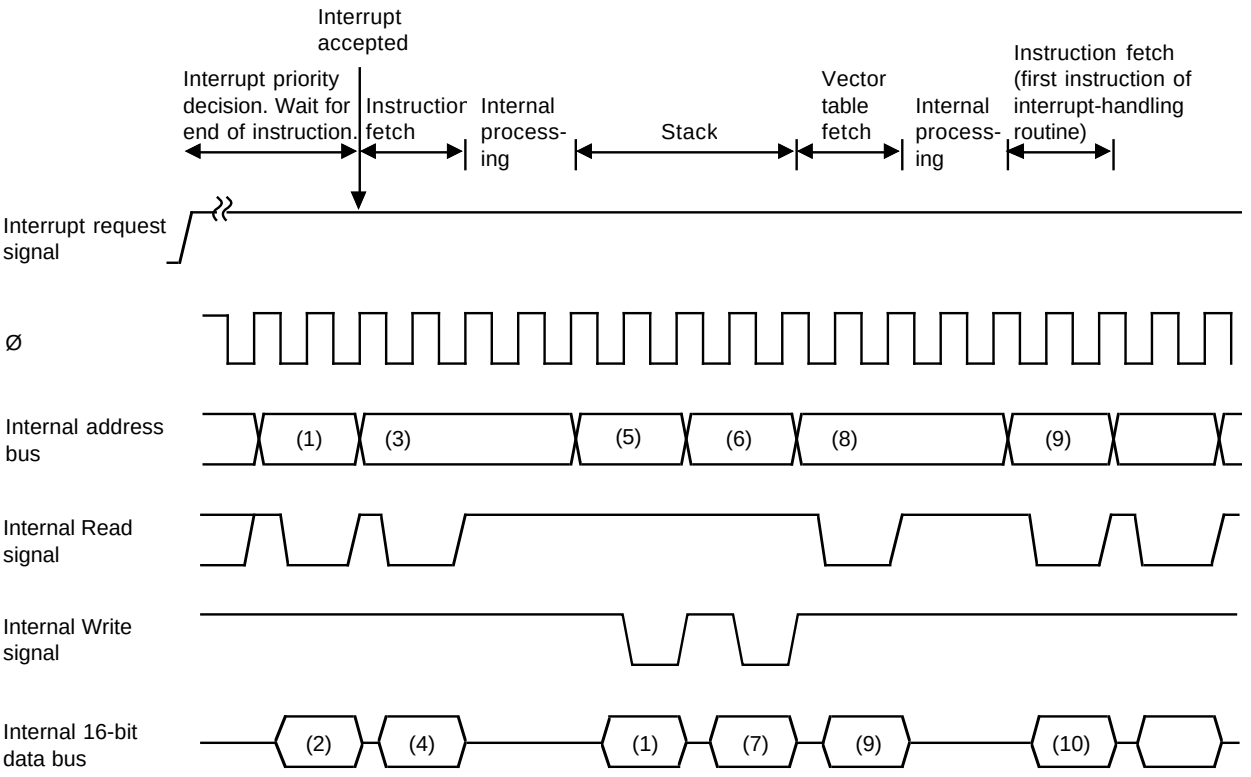

PC : Program counter
CCR : Condition code register
SP : Stack pointer

*: Ignored on return.

Notes: 1. The PC contains the address of the first instruction executed after return.

2. Registers must be saved and restored by word access at an even address.

**Figure 4-5. Usage of Stack in Interrupt Handling**




(1) Instruction prefetch address (Pushed on stack. Instruction is executed on return from interrupt-handling routine.)
(2) (4) Instruction code (Not executed)
(3) Instruction prefetch address (Not executed)
(5) SP–2
(6) SP–4
(7) CCR
(8) Address of vector table entry
(9) Vector table entry (address of first instruction interrupt-handling routine)
(10) First instruction of interrupt-handling routine

**Figure 4-6. Timing of Interrupt Sequence**

### 4.3.6 Interrupt Response Time

Table 4-4 indicates the time that elapses from an interrupt request signal until the first instruction of the software interrupt-handling routine is executed. Since the H8/325 Series accesses its on-chip memory 16 bits at a time, very fast interrupt service can be obtained by placing interrupt-handling routines in on-chip ROM and the stack in on-chip RAM.

**Table 4-4. Number of States before Interrupt Service**

| | | Number of states | |
|---|---|---|---|
| **No.** | **Reason for wait** | **On-chip memory** | **External memory** |
| 1 | Interrupt priority decision | 2*3 | 2*3 |
| 2 | Wait for completion of current instruction*1 | 1 to 13 | 5 to 17*2 |
| 3 | Save PC and CCR | 4 | 12*2 |
| 4 | Fetch vector | 2 | 6*2 |
| 5 | Fetch instruction | 4 | 12*2 |
| 6 | Internal processing | 4 | 4 |
| | Total | 17 to 29 | 41 to 53*2 |

**Notes:**
1. These values do not apply if the current instruction is an EEPMOV, MOVFPE, or MOVTPE instruction.
2. If wait states are inserted in external memory access, these values may be longer.
3. 1 for internal interrupts.

## 4.4 Note on Stack Handling

In word access, the least significant bit of the address is always assumed to be 0. The stack is always accessed by word access. Care should be taken to keep an even value in the stack pointer (general register R7). Use the PUSH and POP (or MOV.W Rn, @–SP and MOV.W @SP+, Rn) instructions to push and pop registers on the stack.

Setting the stack pointer to an odd value can cause programs to crash. Figure 4-7 shows an example of damage caused when the stack pointer contains an odd address.

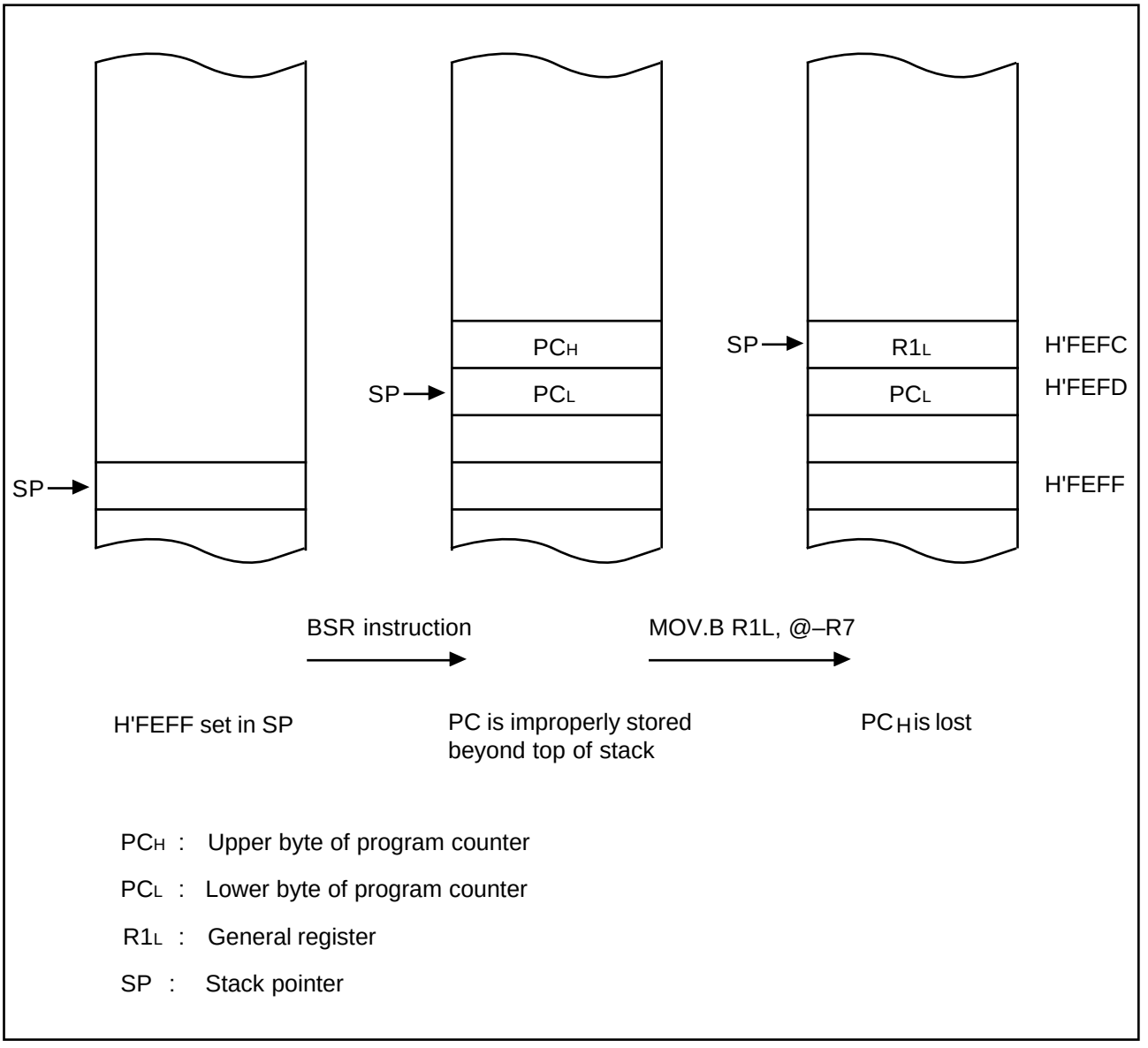



**Figure 4-7. Example of Damage Caused by Setting an Odd Address in R7**

Although the CCR consists of only one byte, it is treated as word data when pushed on the stack. In the hardware interrupt exception-handling sequence, two identical CCR bytes are pushed onto the stack to make a complete word. When popped from the stack by an RTE instruction, the CCR is loaded from the byte stored at the even address. The byte stored at the odd address is ignored.

# Section 5. I/O Ports

## 5.1 Overview

The H8/325 Series has seven parallel I/O ports, including:

- Five 8-bit input/output ports—ports 1, 2, 3, 4, and 7
- One 7-bit input/output port—port 6
- One 6-bit input/output port—port 5

All ports have programmable MOS input pull-ups. Ports 1 and 2 can drive LEDs.

Input and output are memory-mapped. The CPU views each port as a data register (DR) located in the register field at the high end of the address space. Each port also has a data direction register (DDR) which determines which pins are used for input and which for output.

**Output:** To send data to an output port, the CPU selects output in the data direction register and writes the desired data in the data register, causing the data to be held in a latch. The latch output drives the pin through a buffer amplifier. If the CPU reads the data register of an output port, it obtains the data held in the latch rather than the actual level of the pin.

**Input:** To read data from an I/O port, the CPU selects input in the data direction register and reads the data register. This causes the input logic level at the pin to be placed directly on the internal data bus. There is no intervening input latch, except for port 3 when parallel handshaking is used.

**MOS Pull-Up:** The MOS pull-ups for input pins are controlled as follows. To turn on the pull-up transistor for a pin, software must first clear its data direction bit to 0 to make the pin an input pin, then write a 1 in the data bit for that pin. The pull-up can be turned off by writing a 0 in the data bit, or a 1 in the data direction bit. The pull-ups are also turned off by a reset and by entry to the hardware standby mode.

The data direction registers are write-only registers; their contents are invisible to the CPU. If the CPU reads a data direction register all bits are read as 1, regardless of their true values. Care is required if bit manipulation instructions are used to set and clear the data direction bits. See the note on bit manipulation instructions in section 3.5.5, Bit Manipulations.

**Auxiliary Functions:** In addition to their general-purpose input/output functions, all of the I/O ports have auxiliary functions. Most of the auxiliary functions are software-selectable and must be enabled by setting bits in control registers. When selected, an auxiliary function usually replaces the general-purpose input/output function, but in some cases both functions operate simultaneously. Table 5-1 summarizes the auxiliary functions of the ports.

**Table 5-1. Auxiliary Functions of Input/Output Ports**

| I/O Port | Auxiliary functions | |
|---|---|---|
| Port 1 | Address bus (low) | (Note 1) |
| Port 2 | Address bus (high) | (Note 1) |
| Port 3 | Data bus or parallel handshaking data lines | (Note 2) |
| Port 4 | System clock and E clock output, 8-bit timer input and output | |
| Port 5 | Serial communication interface | |
| Port 6 | Free-running timer input and output, $IRQ_2$ to $IRQ_0$ | |
| Port 7 | Bus control and parallel handshaking control | |

**Notes:**

*1 Selected automatically in mode 1; software-selectable in mode 2

*2 Data bus function is selected automatically in modes 1 and 2

## 5.2 Port 1

Port 1 is an 8-bit input/output port that also provides the low bits of the address bus. The function of port 1 depends on the MCU mode as indicated in table 5-2.

**Table 5-2. Functions of Port 1**

| Mode 1 | Mode 2 | Mode 3 |
|---|---|---|
| Address bus (low) ($A_7$ to $A_0$) | Input port or Address bus (low) ($A_7$ to $A_0$)* | Input/output port |

* Depending on the bit settings in the data direction register: 0—input pin; 1—address pin

Pins of port 1 can drive a single TTL load and a 90-pF capacitive load when they are used as output pins. They can also drive light-emitting diodes or a Darlington pair.

Table 5-3 details the port 1 registers.

**Table 5-3. Port 1 Registers**

| Name | Abbreviation | Read/Write | Initial value | Address |
|---|---|---|---|---|
| Port 1 data direction register | P1DDR | W | H'FF (mode 1)<br>H'00 (modes 2 and 3) | H'FFB0 |
| Port 1 data register | P1DR | R/W | H'00 | H'FFB2 |

**Port 1 Data Direction Register (P1DDR)—H'FFB0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P1_7$DDR | $P1_6$DDR | $P1_5$DDR | $P1_4$DDR | $P1_3$DDR | $P1_2$DDR | $P1_1$DDR | $P1_0$DDR |
| Mode 1 | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | — | — | — | — |
| Modes 2 and 3 | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

P1DDR is an 8-bit register that selects the direction of each pin in port 1. A pin functions as an output pin if the corresponding bit in P1DDR is set to 1, and as an input pin if the bit is cleared to 0.

**Port 1 Data Register (P1DR)—H'FFB2**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P1_7$ | $P1_6$ | $P1_5$ | $P1_4$ | $P1_3$ | $P1_2$ | $P1_1$ | $P1_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

P1DR is an 8-bit register containing output data for pins $P1_7$ to $P1_0$, and controlling their input pull-ups.

**MOS Pull-Ups:** Are available for input pins in modes 2 and 3. Software can turn on the MOS pull-up by writing a 1 in P1DR, and turn it off by writing a 0. The pull-ups are automatically turned off for output pins in modes 2 and 3, and for all pins in mode 1.

**Mode 1:** In mode 1 (expanded mode without on-chip ROM), port 1 is automatically used for address output. The port 1 data direction register is unwritable. All bits in P1DDR are automatically set to 1 and cannot be cleared to 0.

**Mode 2:** In mode 2 (expanded mode with on-chip ROM), the usage of port 1 can be selected on a pin-by-pin basis. A pin is used for general-purpose input if its data direction bit is cleared to 0, or for address output if its data direction bit is set to 1.

**Mode 3:** In the single-chip mode port 1 is a general-purpose input/output port.

**Reset:** A reset clears P1DDR and P1DR to all 0, placing all pins in the input state with the MOS pull-ups off. In mode 1, when the chip comes out of reset P1DDR is set to all 1, making all pins address output pins.

**Hardware Standby Mode:** All pins are placed in the high-impedance state with the MOS pull-ups off.

**Software Standby Mode:** P1DDR and P1DR remain in their previous state. Address output pins are low. General-purpose output pins continue to output the data in P1DR. The MOS pull-ups of input pins are on or off depending on the values in P1DR.

Figure 5-1 shows a schematic diagram of port 1.

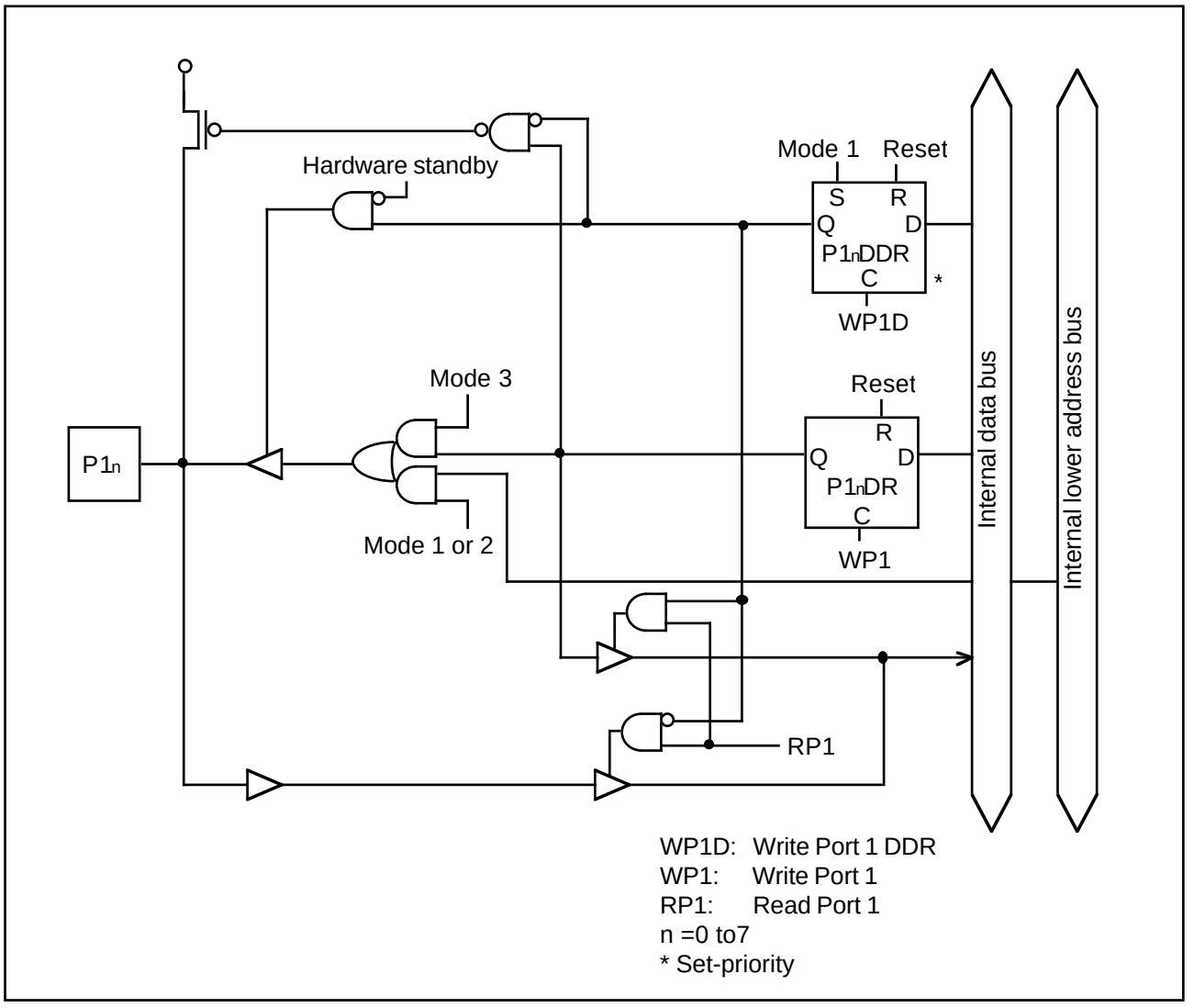



**Figure 5-1. Port 1 Schematic Diagram**

## 5.3 Port 2

Port 2 is an 8-bit input/output port that also provides the high bits of the address bus. The function of port 2 depends on the MCU mode as indicated in table 5-4.

**Table 5-4. Functions of Port 2**

| Mode 1 | Mode 2 | Mode 3 |
|---|---|---|
| Address bus (high) ($A_{15}$ to $A_8$) | Input port or Address bus (high) ($A_{15}$ to $A_8$)* | Input/output port |

* Depending on the bit settings in the data direction register: 0—input pin; 1—address pin

Pins of port 2 can drive a single TTL load and a 90-pF capacitive load when they are used as output pins. They can also drive light-emitting diodes or a Darlington pair.

Table 5-5 details the port 2 registers.

**Table 5-5. Port 2 Registers**

| Name | Abbreviation | Read/Write | Initial value | Address |
|---|---|---|---|---|
| Port 2 data direction register | P2DDR | W | H'FF (mode 1)<br>H'00 (modes 2 and 3) | H'FFB1 |
| Port 2 data register | P2DR | R/W | H'00 | H'FFB3 |

**Port 2 Data Direction Register (P2DDR)—H'FFB1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P2_7DDR$ | $P2_6DDR$ | $P2_5DDR$ | $P2_4DDR$ | $P2_3DDR$ | $P2_2DDR$ | $P2_1DDR$ | $P2_0DDR$ |
| Mode 1 | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | — | — | — | — |
| Modes 2 and 3 | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

P2DDR is an 8-bit register that selects the direction of each pin in port 2. A pin functions as an output pin if the corresponding bit in P2DDR is set to 1, and as an input pin if the bit is cleared to 0.

**Port 2 Data Register (P2DR)—H'FFB3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P2_7$ | $P2_6$ | $P2_5$ | $P2_4$ | $P2_3$ | $P2_2$ | $P2_1$ | $P2_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

P2DR is an 8-bit register containing output data for pins $P2_7$ to $P2_0$, and controlling their input pull-ups.

**MOS Pull-Ups:** Are available for input pins in modes 2 and 3. Software can turn on the MOS pull-up by writing a 1 in P2DR, and turn it off by writing a 0. The pull-ups are automatically turned off for output pins in modes 2 and 3, and for all pins in mode 1.

**Mode 1:** In mode 1 (expanded mode without on-chip ROM), port 2 is automatically used for address output. The port 2 data direction register is unwritable. All bits in P2DDR are automatically set to 1 and cannot be cleared to 0.

**Mode 2:** In mode 2 (expanded mode with on-chip ROM), the usage of port 2 can be selected on a pin-by-pin basis. A pin is used for general-purpose input if its data direction bit is cleared to 0, or for address output if its data direction bit is set to 1.

**Mode 3:** In single-chip mode port 2 is a general-purpose input/output port.

**Reset:** A reset clears P2DDR and P2DR to all 0, placing all pins in the input state with the MOS pull-ups off. In mode 1, when the chip comes out of reset P2DDR is set to all 1, making all pins address output pins.

**Hardware Standby Mode:** All pins are placed in the high-impedance state with the MOS pull-ups off.

**Software Standby Mode:** P2DDR and P2DR remain in their previous state. Address output pins are low. General-purpose output pins continue to output the data in P2DR. The MOS pull-ups of input pins are on or off depending on the values in P2DR.

Figure 5-2 shows a schematic diagram of port 2.

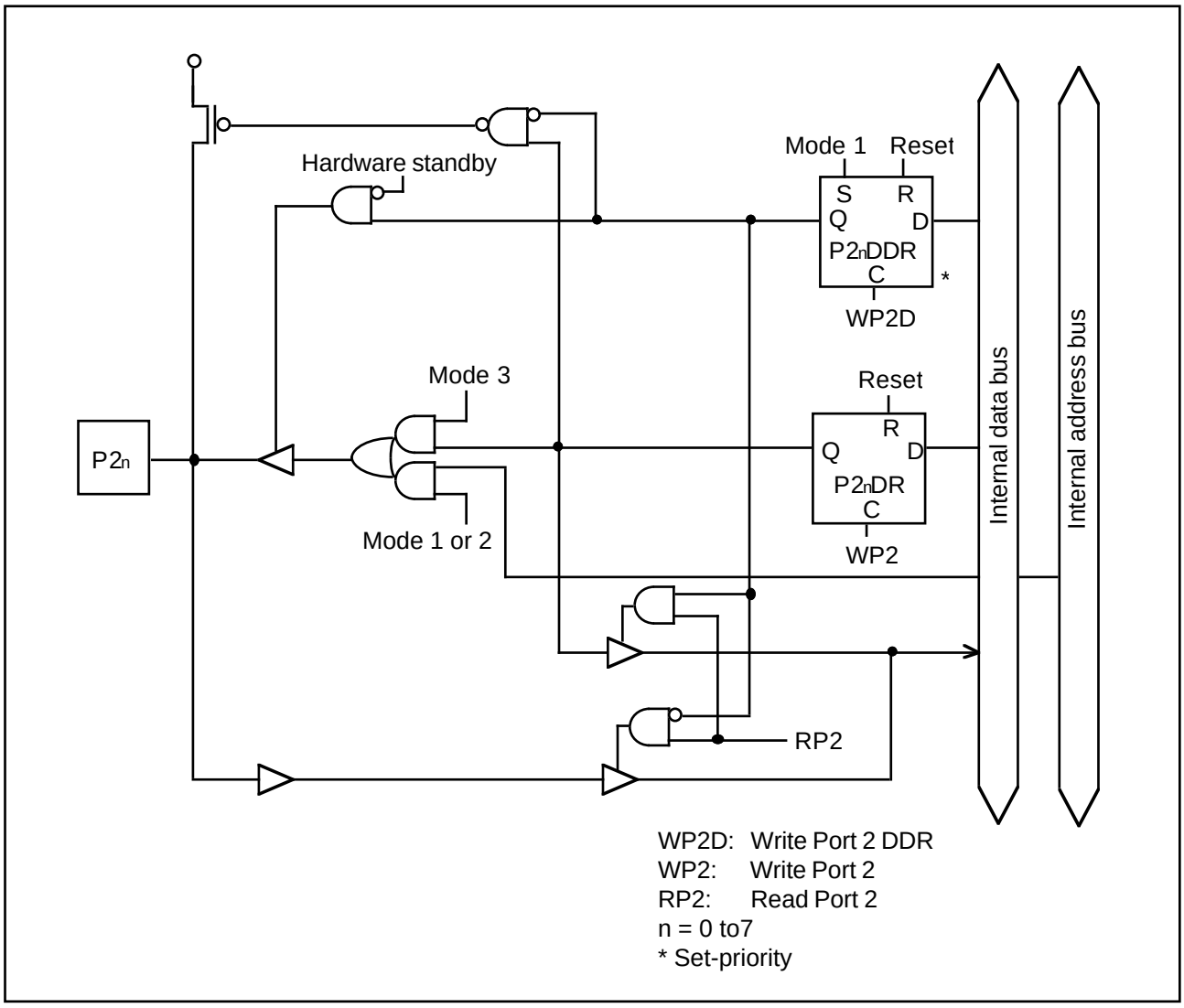



**Figure 5-2. Port 2 Schematic Diagram**

## 5.4 Port 3

Port 3 is an 8-bit input/output port that also provides the external data bus and data pins for the parallel handshaking interface. The function of port 3 depends on the MCU mode as indicated in table 5-6. For further information on parallel handshaking, see section 6, Parallel Handshaking Interface.

**Table 5-6. Functions of Port 3**

| Mode 1 | Mode 2 | Mode 3 |
|---|---|---|
| Data bus | Data bus | General-purpose input/output port or parallel handshaking port |

Pins of port 3 can drive a single TTL load and a 90-pF capacitive load when they are used as output

pins. They can also drive a Darlington pair.

Table 5-7 details the port 3 registers.

**Table 5-7. Port 3 Registers**

| Name | Abbreviation | Read/Write | Initial value | Address |
|---|---|---|---|---|
| Port 3 data direction register | P3DDR | W | H'FF | H'FFB4 |
| Port 3 data register | P3DR | R/W | H'00 | H'FFB6 |

**Port 3 Data Direction Register (P3DDR)—H'FFB4**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P3_7DDR$ | $P3_6DDR$ | $P3_5DDR$ | $P3_4DDR$ | $P3_3DDR$ | $P3_2DDR$ | $P3_1DDR$ | $P3_0DDR$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

P3DDR is an 8-bit register that selects the direction of each pin in port 3. A pin functions as an output pin if the corresponding bit in P3DDR is set to 1, and as an input pin if the bit is cleared to 0.

**Port 3 Data Register (P3DR)—H'FFB6**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P3_7$ | $P3_6$ | $P3_5$ | $P3_4$ | $P3_3$ | $P3_2$ | $P3_1$ | $P3_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

P3DR is an 8-bit register containing output data for pins $P3_7$ to $P3_0$ in mode 3, and controlling their input pull-ups.

**MOS Pull-Ups:** Are available for input pins in mode 3. Software can turn on the MOS pull-up by writing a 1 in P3DR, and turn it off by writing a 0. The pull-ups are automatically turned off for output pins in mode 3, and for all pins in modes 1 and 2.

**Modes 1 and 2:** In the expanded modes, port 3 is automatically used as the data bus. The values in P3DDR and P3DR are ignored.

**Mode 3:** In the single-chip mode, port 3 can be used as a general-purpose input/output port, or a parallel-handshaking input or output port.

**Input Latches:** All pins of port 3 have input latches which can be enabled by the LTE bit in the handshake control/status register (HCSR) in mode 3. When the LTE bit is set to 1, input data are latched on the falling edge of the input strobe ($\overline{IS}$) signal and held in the input strobe latch until read. When the LTE bit is cleared to 0, input data are passed through the input strobe latch without being held.

See section 6, Parallel Handshaking Interface for further information.

**Reset and Hardware Standby Mode:** P3DDR and P3DR are cleared to all 0, and all parallel handshaking functions are disabled. All pins are placed in the input (high-impedance) state with the MOS pull-ups off.

**Software Standby Mode:** P3DDR and P3DR remain in their previous state. In modes 1 and 2, all pins are placed in the input (high-impedance) state. In mode 3, all pins remain in their previous input or output state.

Figure 5-3 shows a schematic diagram of port 3.

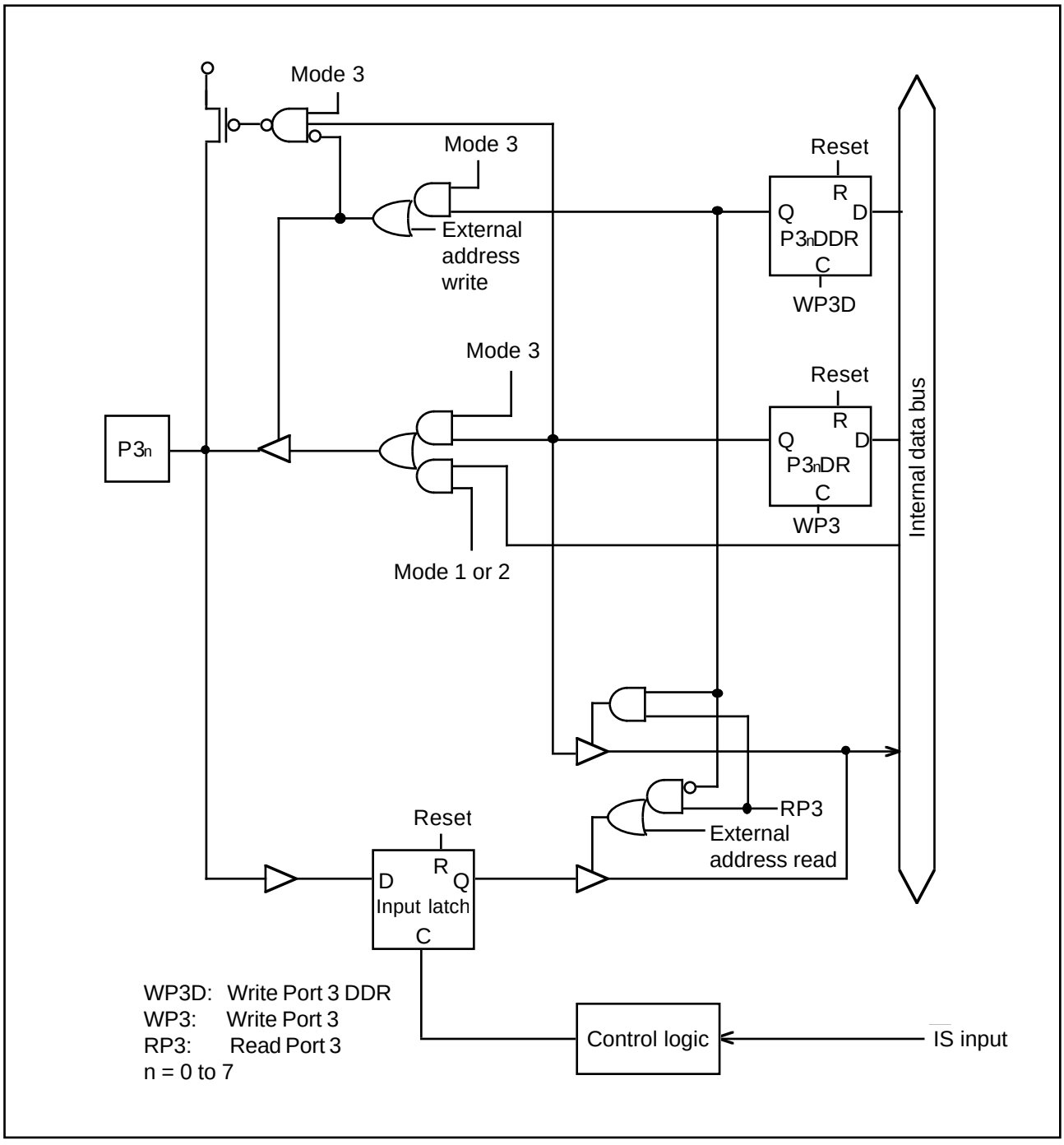



**Figure 5-3. Port 3 Schematic Diagram**

## 5.5 Port 4

Port 4 is an 8-bit input/output port that also provides input and output pins for the 8-bit timers and output pins for the system clock and E clock. The pin functions depend on the MCU mode and output select bits in the timer control/status registers. Table 5-8 lists the pin functions.

**Table 5-8. Port 4 Pin Functions**

| Usage | Pin Functions | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| I/O port | $P4_0$ | $P4_1$ | $P4_2$ | $P4_3$ | $P4_4$ | $P4_5$ | $P4_6$ | $P4_7$ |
| Timer or clock | $TMCI_0$ | $TMO_0$ | $TMRI_0$ | $TMCI_1$ | $TMO_1$ | $TMRI_1$ | Ø clock | E clock |

See section 8, 8-Bit Timer Module for details of the timer output select bits.

Pins of port 4 can drive a single TTL load and a 90-pF capacitive load when they are used as output pins. They can also drive a Darlington pair.

Table 5-9 details the port 4 registers.

**Table 5-9. Port 4 Registers**

| Name | Abbreviation | Read/Write | Initial value | Address |
|---|---|---|---|---|
| Port 4 data direction register | P4DDR | W | H’80 (modes 1 and 2)<br>H'00 (mode 3) | H’FFB5 |
| Port 4 data register | P4DR | R/W | H’00 | H’FFB7 |

**Port 4 Data Direction Register (P4DDR)—H’FFB5**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P4_7$DDR | $P4_6$DDR | $P4_5$DDR | $P4_4$DDR | $P4_3$DDR | $P4_2$DDR | $P4_1$DDR | $P4_0$DDR |
| Modes 1 and 2 | | | | | | | | |
| Initial value | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |
| Mode 3 | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

P4DDR is an 8-bit register that selects the direction of each pin in port 4. A pin functions as an output pin if the corresponding bit in P4DDR is set to 1, and as an input pin if the bit is cleared to 0.

**Port 4 Data Register (P4DR)—H'FFB7**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P4_7$ | $P4_6$ | $P4_5$ | $P4_4$ | $P4_3$ | $P4_2$ | $P4_1$ | $P4_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

P4DR is an 8-bit register containing output data for pins $P4_7$ to $P4_0$, and controlling their input pull-ups. When the CPU reads P4DR, for output pins (P4DDR = 1) it reads the value in the P4DR latch, but for input pins (P4DDR = 0), it obtains the logic level directly from the pin, bypassing the P4DR latch. This also applies to pins used for timer or clock input or output.

**MOS Pull-Ups:** Are available for input pins, including timer input pins, in all modes. Software can turn the MOS pull-up on by writing a 1 in P4DR, and turn it off by writing a 0. The pull-ups are automatically turned off for output pins.

**Pins $P4_0$, $P4_2$, $P4_3$, and $P4_5$:** As indicated in table 5-8, these pins can be used for general-purpose input or output, or input of 8-bit timer clock and reset signals. When a pin is used for timer signal input, its P4DDR bit should normally be cleared to 0; otherwise the timer will receive the value in P4DR. If input pull-up is not desired, the P4DR bit should also be cleared to 0.

**Pins $P4_1$ and $P4_4$:** As indicated in table 5-8, these pins can be used for general-purpose input or output, or for 8-bit timer output. Pins used for timer output are unaffected by the values in P4DDR and P4DR, and their MOS pull-ups are automatically turned off.

**Pin $P4_6$:** In modes 1 and 2 (expanded modes) this pin is used for system clock (Ø) output, regardless of the value in $P4_6$DDR. The MOS pull-up is automatically turned off.

In mode 3 (single-chip mode) this pin is used for general-purpose input if $P4_6$DDR is cleared to 0, or system clock output if $P4_6$DDR is set to 1. It cannot be used for general-purpose output.

**Pin $P4_7$:** In modes 1 and 2 (expanded modes) pin $P4_7$ is used for E clock output if $P4_7$DDR is set to 1, and for general-purpose input if $P4_7$DDR is cleared to 0. It cannot be used for general-purpose output.

In mode 3 (single-chip mode) pin $P4_7$ is used for general-purpose input/output.

**Reset:** P4DDR and P4DR and the 8-bit timer control registers are initialized, making pins $P4_0$ to $P4_5$ into input port pins with the MOS pull-ups off. When the chip comes out of reset into single-chip mode (mode 3), $P4_6$ and $P4_7$ also become input port pins with the MOS pull-ups off. When the chip comes out of reset into an expanded mode (mode 1 or 2), the system clock and E clock are output at $P4_6$ and $P4_7$.

**Hardware Standby Mode:** All pins are placed in the high-impedance state with the MOS pull-ups off.

**Software Standby Mode:** The 8-bit timer control registers are initialized but P4DDR and P4DR remain in their previous states. Pins $P4_0$ to $P4_5$ become input or output port pins depending on the setting of P4DDR. Pins $P4_6$ and $P4_7$ remain in their previous states, with system clock output remaining high and E clock output remaining low. The MOS pull-ups of input pins are on or off depending on the values in P4DR.

Figures 5-4 to 5-7 show schematic diagrams of port 4.

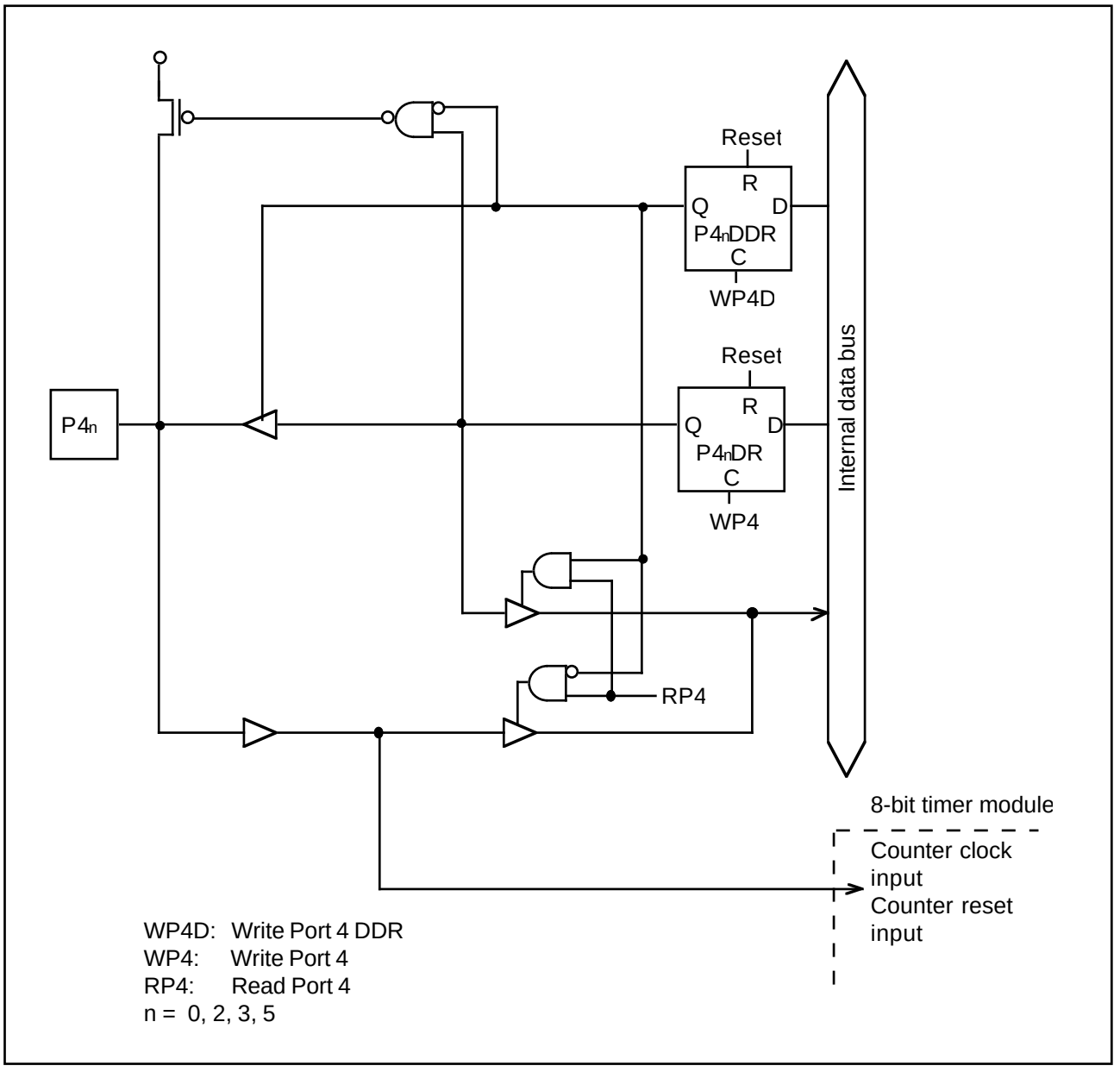



**Figure 5-4. Port 4 Schematic Diagram (Pins P4$_0$, P4$_2$, P4$_3$, and P4$_5$)**

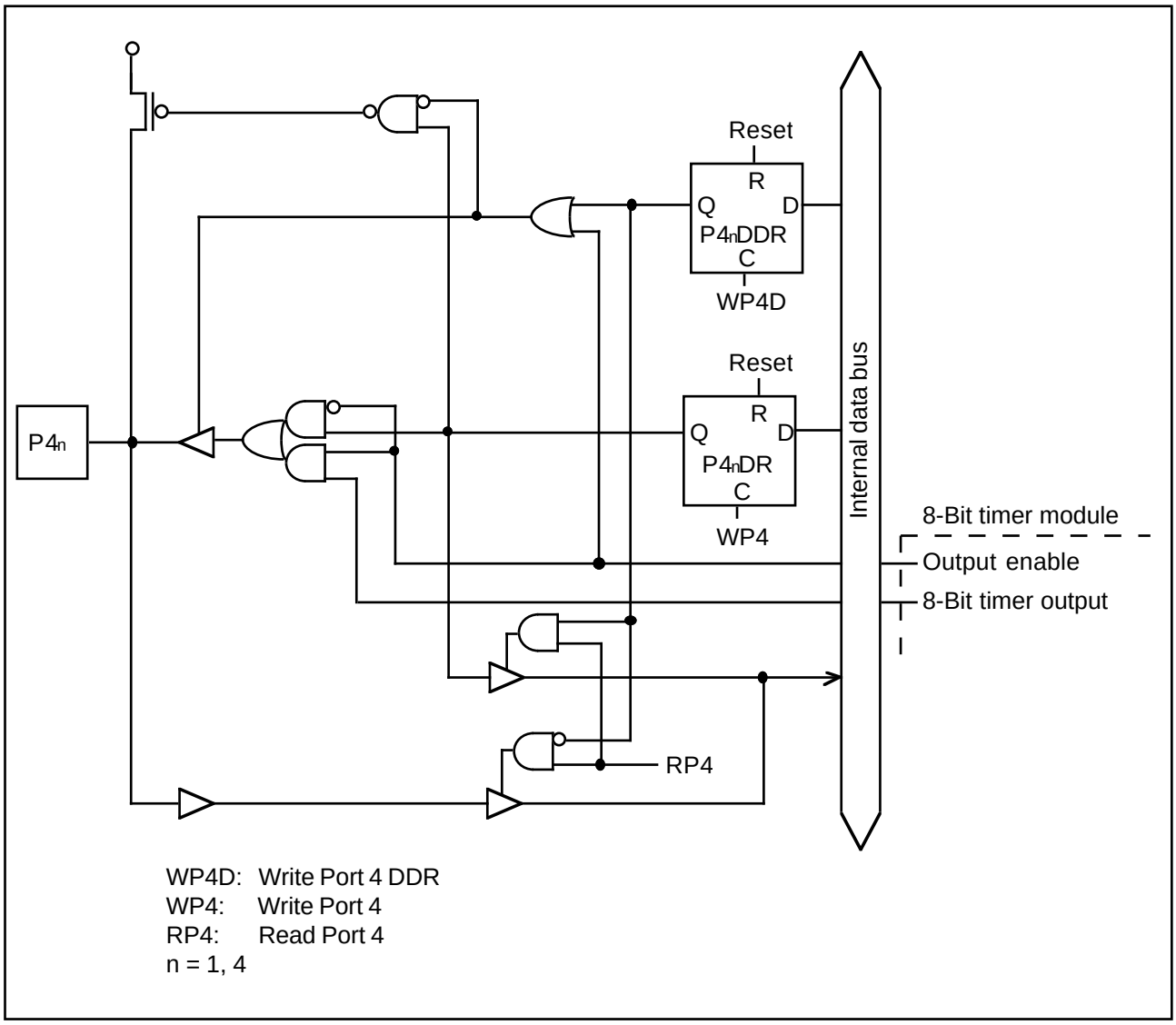



**Figure 5-5. Port 4 Schematic Diagram (Pins P4$_1$ and P4$_4$)**

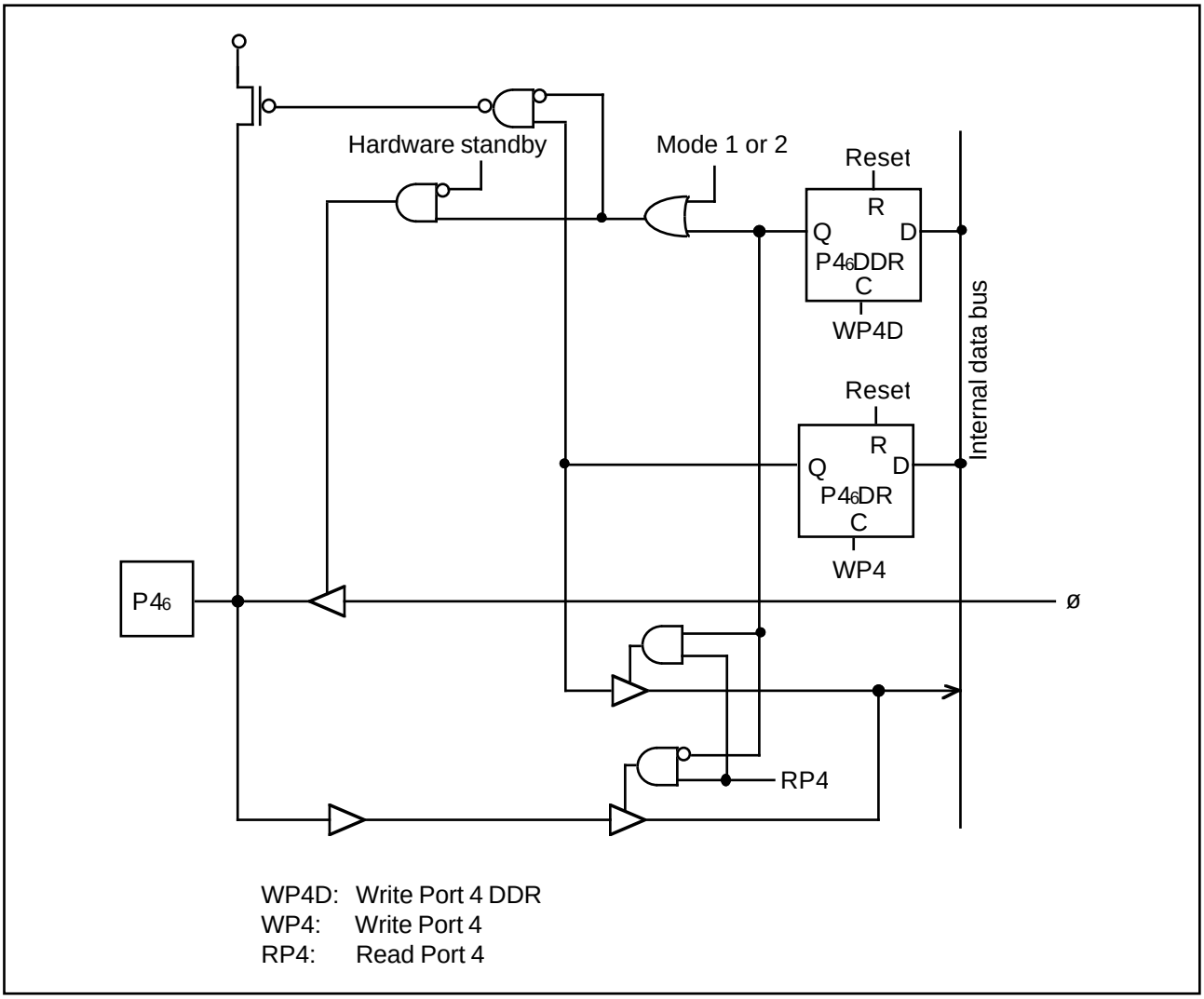



**Figure 5-6. Port 4 Schematic Diagram (Pin P46)**

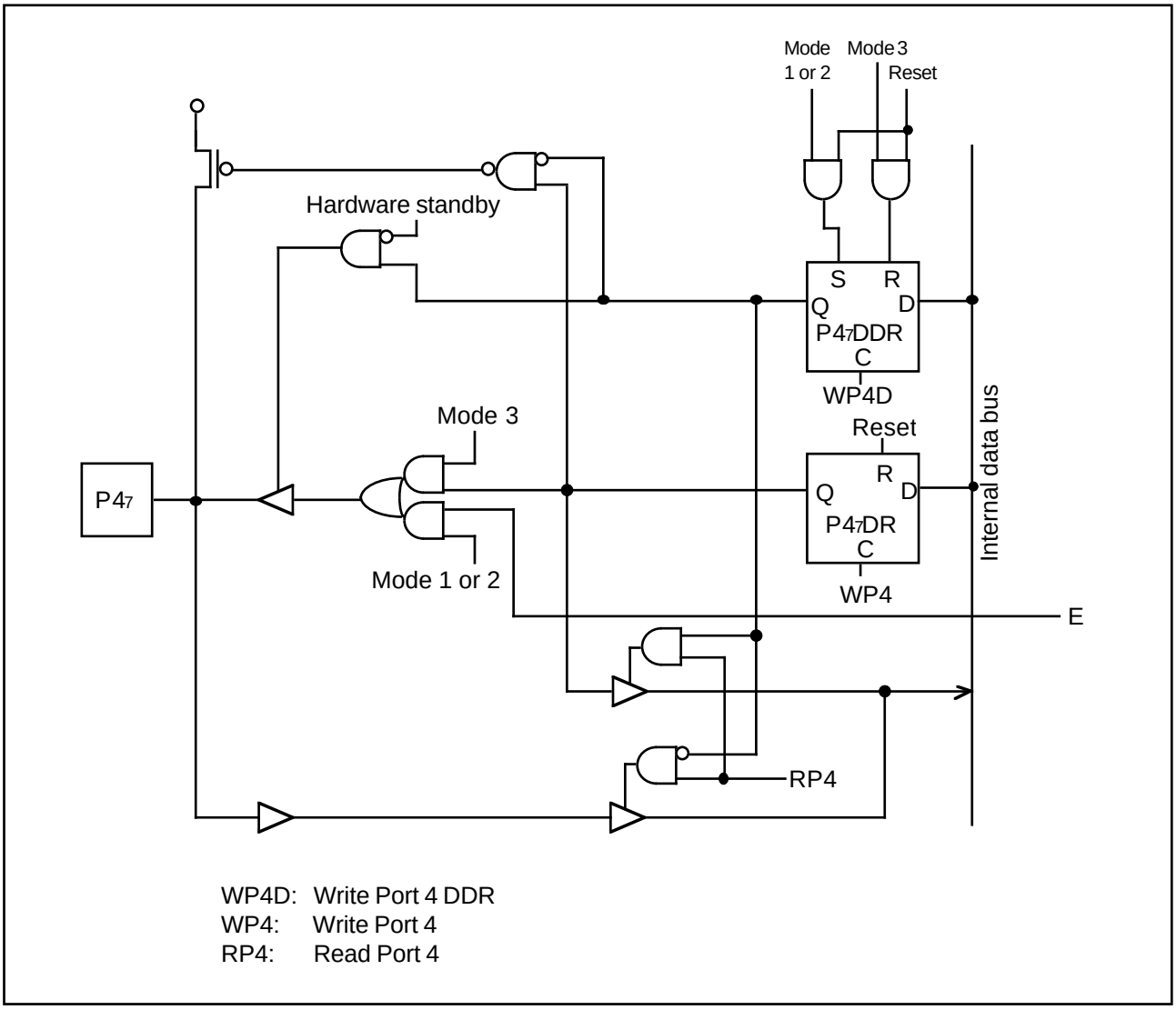



**Figure 5-7. Port 4 Schematic Diagram (Pin P47)**

## 5.6 Port 5

Port 5 is a 6-bit input/output port that also provides the input and output pins for the serial communication interface. The pin functions depend on control bits in the serial control registers. Pins not used for serial communication are available for general-purpose input/output. Table 5-10 lists the pin functions, which are the same in both the expanded and single-chip modes.

**Table 5-10. Port 5 Pin Functions (Modes 1 to 3)**

| Usage | Pin functions | | | | | |
|---|---|---|---|---|---|---|
| I/O port | P50 | P51 | P52 | P53 | P54 | P55 |
| Serial communication | TxD0 | RxD0 | SCK0 | TxD1 | RxD1 | SCK1 |

See section 9, Serial Communication Interface for details of the serial control bits. Pins used by the serial communication interface are switched between input and output without regard to the values in the data direction register.

Pins of port 5 can drive a single TTL load and a 30-pF capacitive load when they are used as output pins. They can also drive a Darlington pair.

Table 5-11 details the port 5 registers.

**Table 5-11. Port 5 Registers**

| Name | Abbreviation | Read/Write | Initial value | Address |
|---|---|---|---|---|
| Port 5 data direction register | P5DDR | W | H'C0 | H'FFB8 |
| Port 5 data register | P5DR | R/W | H'C0 | H'FFBA |

**Port 5 Data Direction Register (P5DDR)—H'FFB8**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | $P5_5DDR$ | $P5_4DDR$ | $P5_3DDR$ | $P5_2DDR$ | $P5_1DDR$ | $P5_0DDR$ |
| Initial value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | W | W | W | W | W | W |

P5DDR is an 8-bit register that selects the direction of each pin in port 5. A pin functions as an output pin if the corresponding bit in P5DDR is set to 1, and as an input pin if the bit is cleared to 0.

**Port 5 Data Register (P5DR)—H'FFBA**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | $P5_5$ | $P5_4$ | $P5_3$ | $P5_2$ | $P5_1$ | $P5_0$ |
| Initial value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | R/W | R/W | R/W | R/W | R/W | R/W |

P5DR is an 8-bit register containing output data for pins $P5_5$ to $P5_0$, and controlling their input pull-ups. When the CPU reads P5DR, for output pins (P5DDR = 1) it reads the value in the P5DR latch, but for input pins (P5DDR = 0), it obtains the logic level directly from the pin, bypassing the P5DR latch. This also applies to pins used for serial communication.

**MOS Pull-Ups:** Are available for input pins, including serial communication input pins. Software can turn the MOS pull-up on by writing a 1 in P5DR, and turn it off by writing a 0. The pull-ups are automatically turned off for output pins.

**Pins $P5_0$ and $P5_3$:** These pins can be used for general-purpose input or output, or for output of serial transmit data (TxD). When used for TxD output, these pins are unaffected by the values in P5DDR and P5DR, and their MOS pull-ups are automatically turned off.

**Pins $P5_1$ and $P5_4$:** These pins can be used for general-purpose input or output, or for input of serial receive data (RxD). When used for RxD input, these pins are unaffected by P5DDR and P5DR, except that software can turn on their MOS pull-ups by clearing their data direction bits to 0 and setting their data bits to 1.

**Pins $P5_2$ and $P5_5$:** These pins can be used for general-purpose input or output, or for serial clock input or output (SCK). When used for SCK output, these pins are unaffected by P5DDR and P5DR. When these pins are used for SCK input, software can turn on their MOS pull-ups by clearing their data direction bits to 0 and setting their data bits to 1.

**Reset and Hardware Standby Mode:** P5DDR and P5DR are cleared to all 0 and the serial control registers are initialized. All pins are placed in the input port (high-impedance) state with the MOS pull-ups off.

**Software Standby Mode:** The serial control registers are initialized but P5DDR and P5DR remain in their previous states. All pins become input or output port pins depending on the setting of P5DDR. Output pins output the values in P5DR. The MOS pull-ups of input pins are on or off depending on the values in P5DR.

Figures 5-8 to 5-10 show schematic diagrams of port 5.

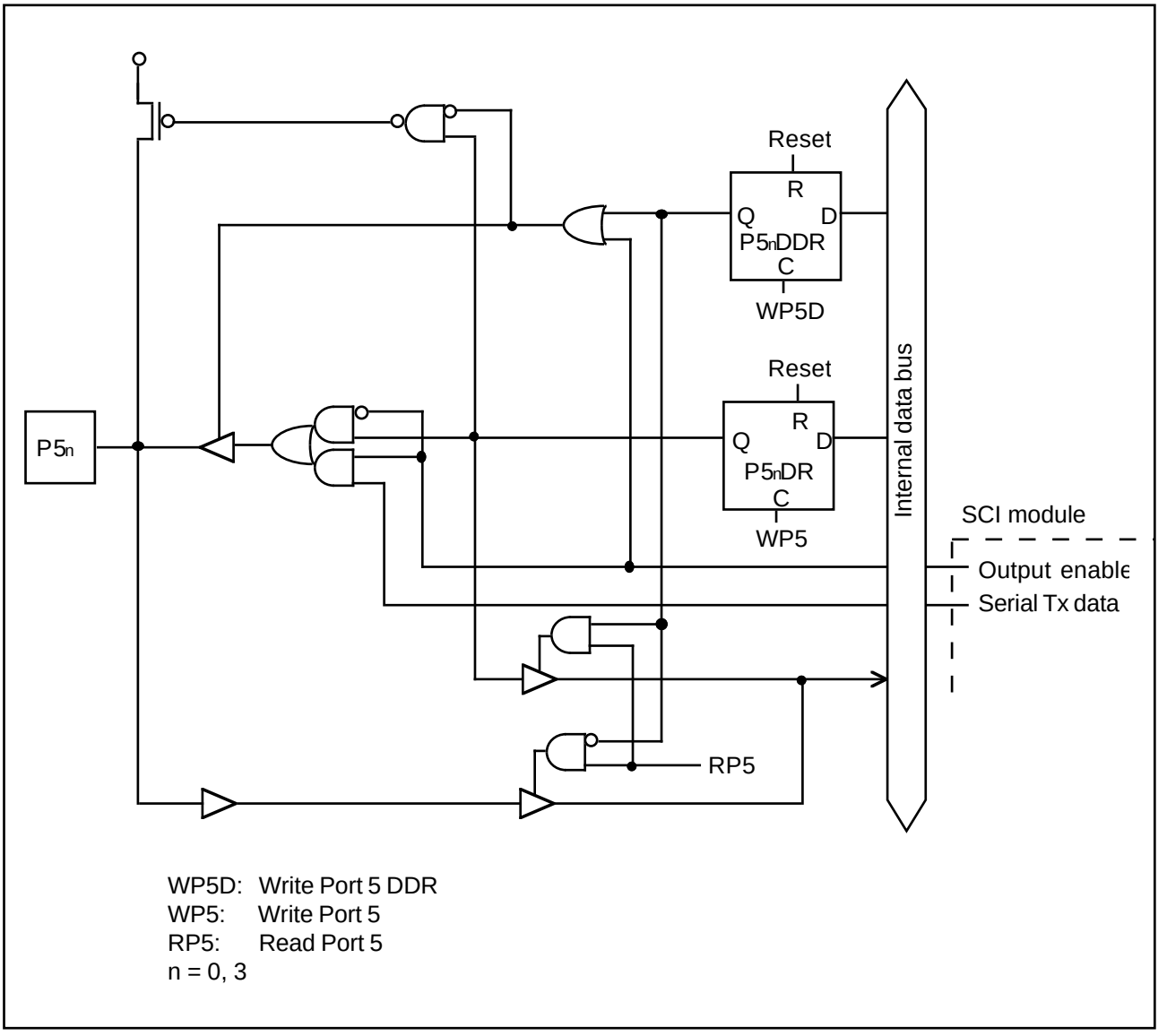



**Figure 5-8. Port 5 Schematic Diagram (Pins P5$_0$ and P5$_3$)**

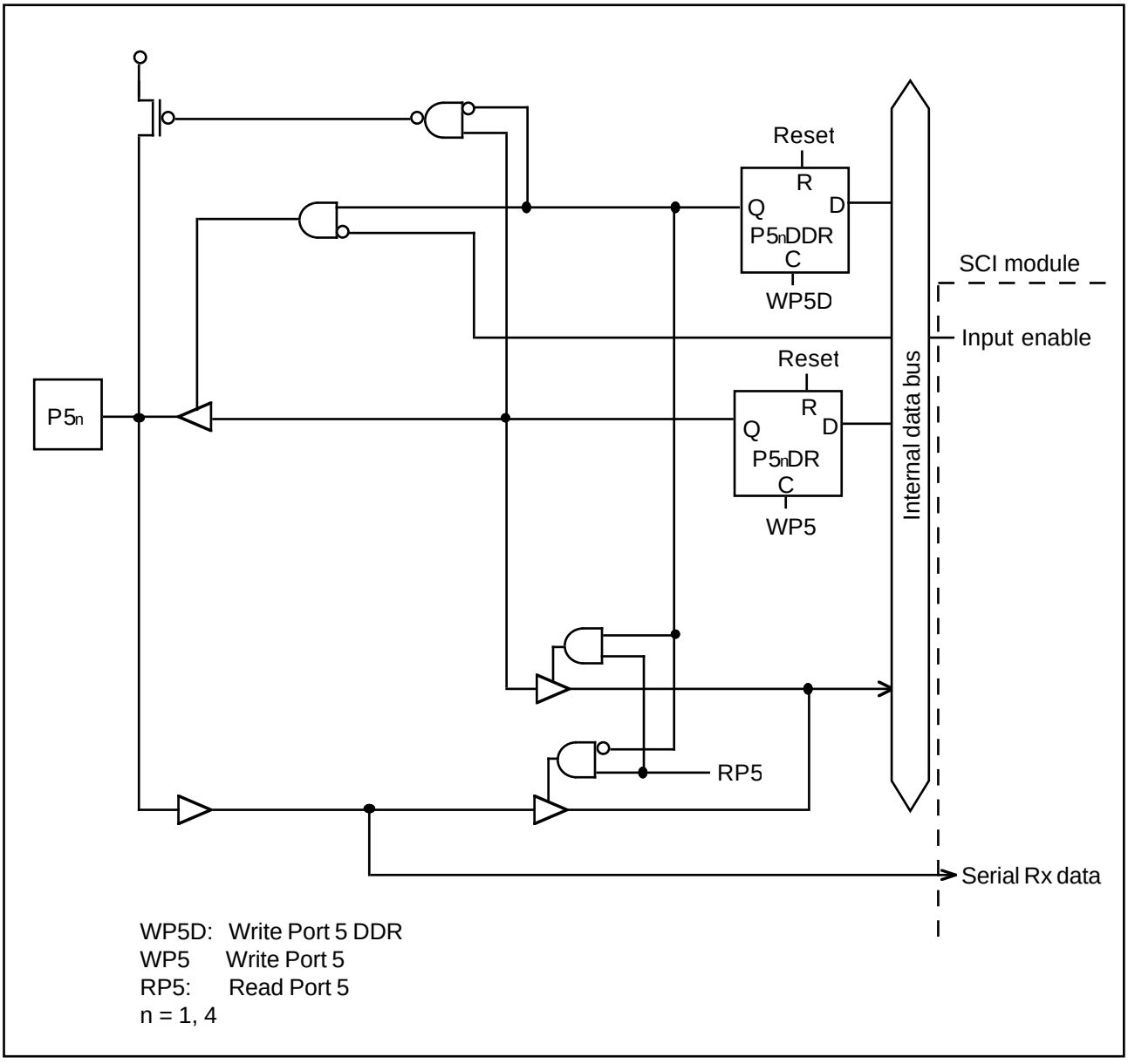



**Figure 5-9. Port 5 Schematic Diagram (Pins P5$_1$ and P5$_4$)**

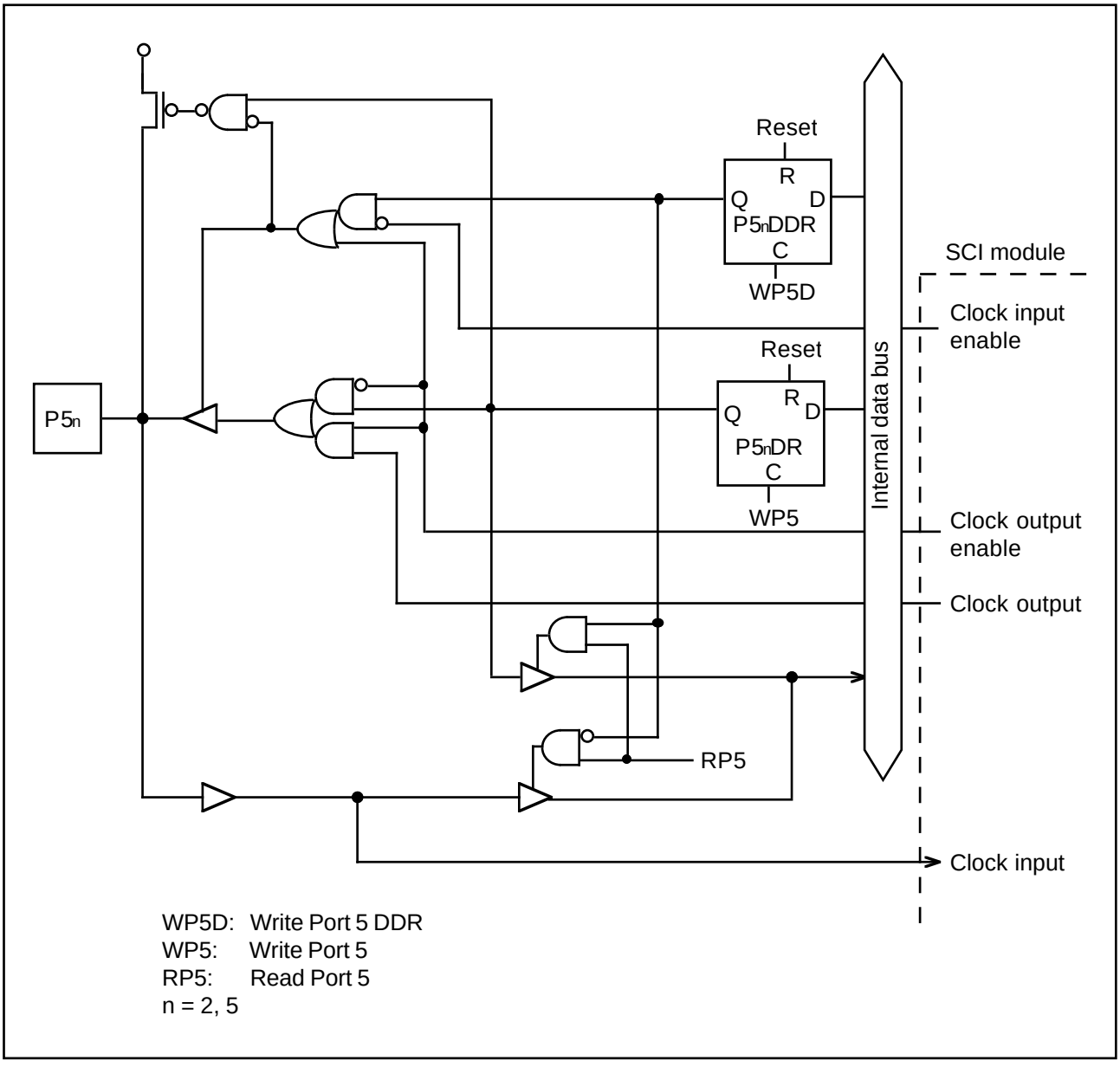



**Figure 5-10. Port 5 Schematic Diagram (Pins P5$_2$ and P5$_5$)**

## 5.7 Port 6

Port 6 is a 7-bit input/output port that also provides input and output pins for the free-running timer, and interrupt request input pins ($\overline{IRQ_0}$ to $\overline{IRQ_2}$). The pin functions depend on control bits in the free-running timer control registers and IRQ enable register. Pins not used for timer or interrupt functions are available for general-purpose input/output. Table 5-12 lists the pin functions, which are the same in both the expanded and single-chip modes.

**Table 5-12. Port 6 Pin Functions**

| Usage | Pin functions (Modes 1 to 3) | | | | | | |
|---|---|---|---|---|---|---|---|
| I/O port | $P6_0$ | $P6_1$ | $P6_2$ | $P6_3$ | $P6_4$ | $P6_5$ | $P6_6$ |
| Timer/interrupt | FTCI | FTOA | FTOB | FTI | $\overline{IRQ_0}$ | $\overline{IRQ_1}$ | $\overline{IRQ_2}$ |

See section 4, Exception Handling and section 7, Free-Running Timer Module for details of the free-running timer and interrupts.

Pins of port 6 can drive a single TTL load and a 90-pF capacitive load when they are used as output pins. They can also drive a Darlington pair.

Table 5-13 details the port 6 registers.

**Table 5-13. Port 6 Registers**

| Name | Abbreviation | Read/Write | Initial value | Address |
|---|---|---|---|---|
| Port 6 data direction register | P6DDR | W | H’80 | H’FFB9 |
| Port 6 data register | P6DR | R/W | H’80 | H’FFBB |

**Port 6 Data Direction Register (P6DDR)—H’FFB9**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | $P6_6$DDR | $P6_5$DDR | $P6_4$DDR | $P6_3$DDR | $P6_2$DDR | $P6_1$DDR | $P6_0$DDR |
| Initial value | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | W | W | W | W | W | W | W |

P6DDR is an 8-bit register that selects the direction of each pin in port 6. A pin functions as an output pin if the corresponding bit in P6DDR is set to 1, and as an input pin if the bit is cleared to 0.

**Port 6 Data Register (P6DR)—H’FFBB**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | $P6_6$ | $P6_5$ | $P6_4$ | $P6_3$ | $P6_2$ | $P6_1$ | $P6_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

P6DR is an 8-bit register containing output data for pins $P6_6$ to $P6_0$, and controlling their input pull-ups. When the CPU reads P6DR, for output pins (P6DDR = 1) it reads the value in the P6DR latch, but for input pins (P6DDR = 0), it obtains the logic level directly from the pin, bypassing the P6DR latch. This also applies to pins used for input and output of timer and interrupt signals.

**MOS Pull-Ups:** Are available for input pins, including pins used for input of timer or interrupt signals. Software can turn the MOS pull-up on by writing a 1 in P6DR, and turn it off by writing a 0. The pull-ups are automatically turned off for output pins.

**Pins $P6_0$ and $P6_3$:** As indicated in table 5-12, these pins can be used for general-purpose input or output, or for input of free-running timer clock and input capture signals. When a pin is used for free-running timer input, its P6DDR bit should be cleared to 0; otherwise the free-running timer will receive the value in P6DR. If input pull-up is not desired, the P6DR bit should also be cleared to 0.

**Pin $P6_1$ and $P6_2$:** These pins can be used for general-purpose input or output, or for the output compare signals (FTOA and FTOB) of the free-running timer. When used for FTOA or FTOB output, these pins are unaffected by the values in P6DDR and P6DR, and their MOS pull-ups are automatically turned off.

**Pins $P6_4$ to $P6_6$:** These pins can be used for general-purpose input or output, or input of interrupt request signals ($\overline{IRQ_0}$ to $\overline{IRQ_2}$). When they are used for interrupt request input, their data direction bits should normally be cleared to 0, so that the value in P6DR will not generate interrupts.

**Reset and Hardware Standby Mode:** P6DDR and P6DR are cleared to all 0. Timer output and interrupt request input are disabled. All pins are placed in the input port (high-impedance) state with the MOS pull-ups off.

**Software Standby Mode:** The free-running timer control registers are initialized but P6DDR, P6DR, and the interrupt control registers remain in their previous states. All pins become input or output port pins or interrupt request pins depending on the settings of P6DDR and the IRQ enable register. Output pins output the values in P6DR. The MOS pull-ups of input pins are on or off depending on the values in P6DR.

Figures 5-11 to 5-13 shows schematic diagrams of port 6.

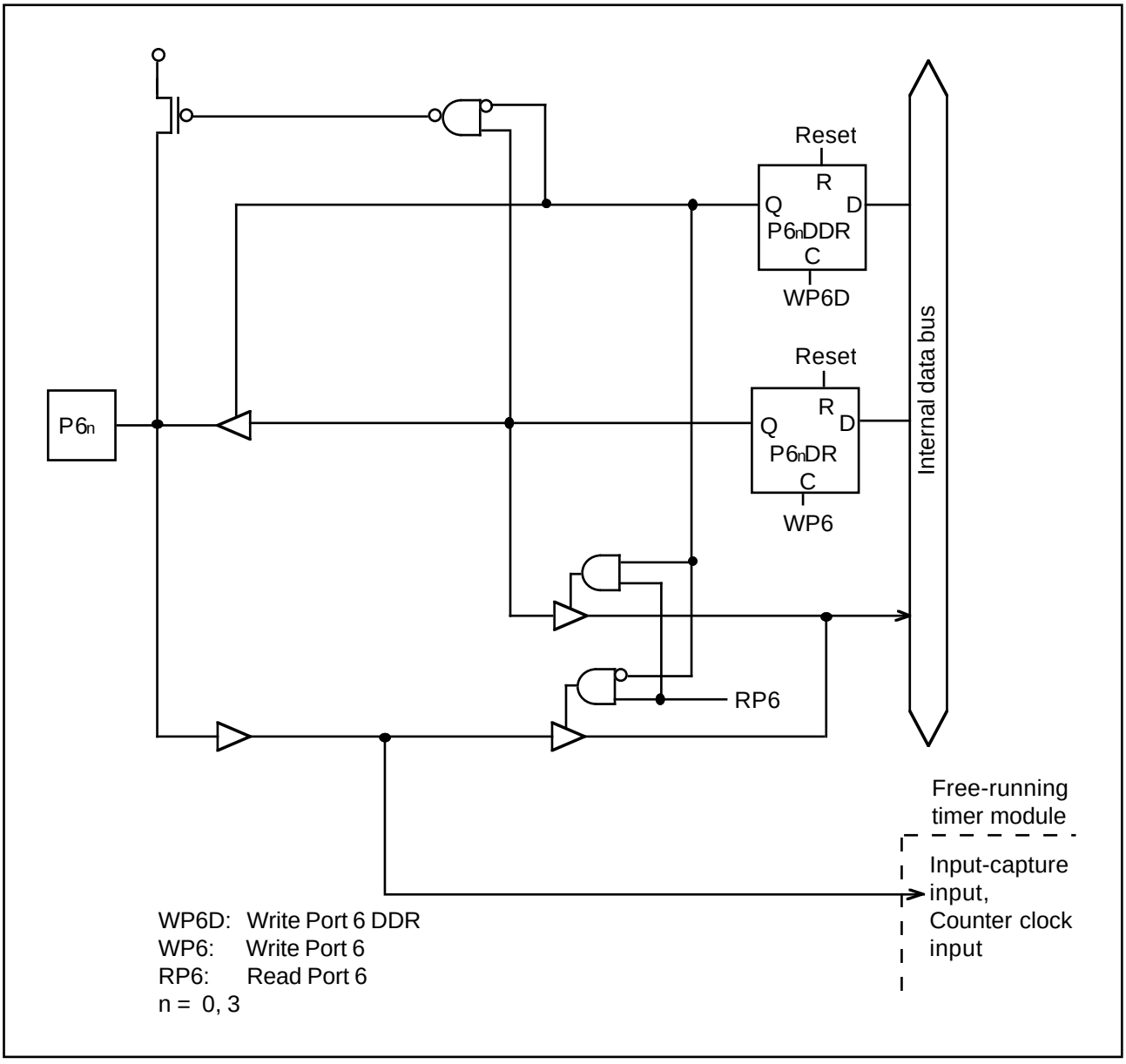



**Figure 5-11. Port 6 Schematic Diagram (Pins P6$_0$ and P6$_3$)**

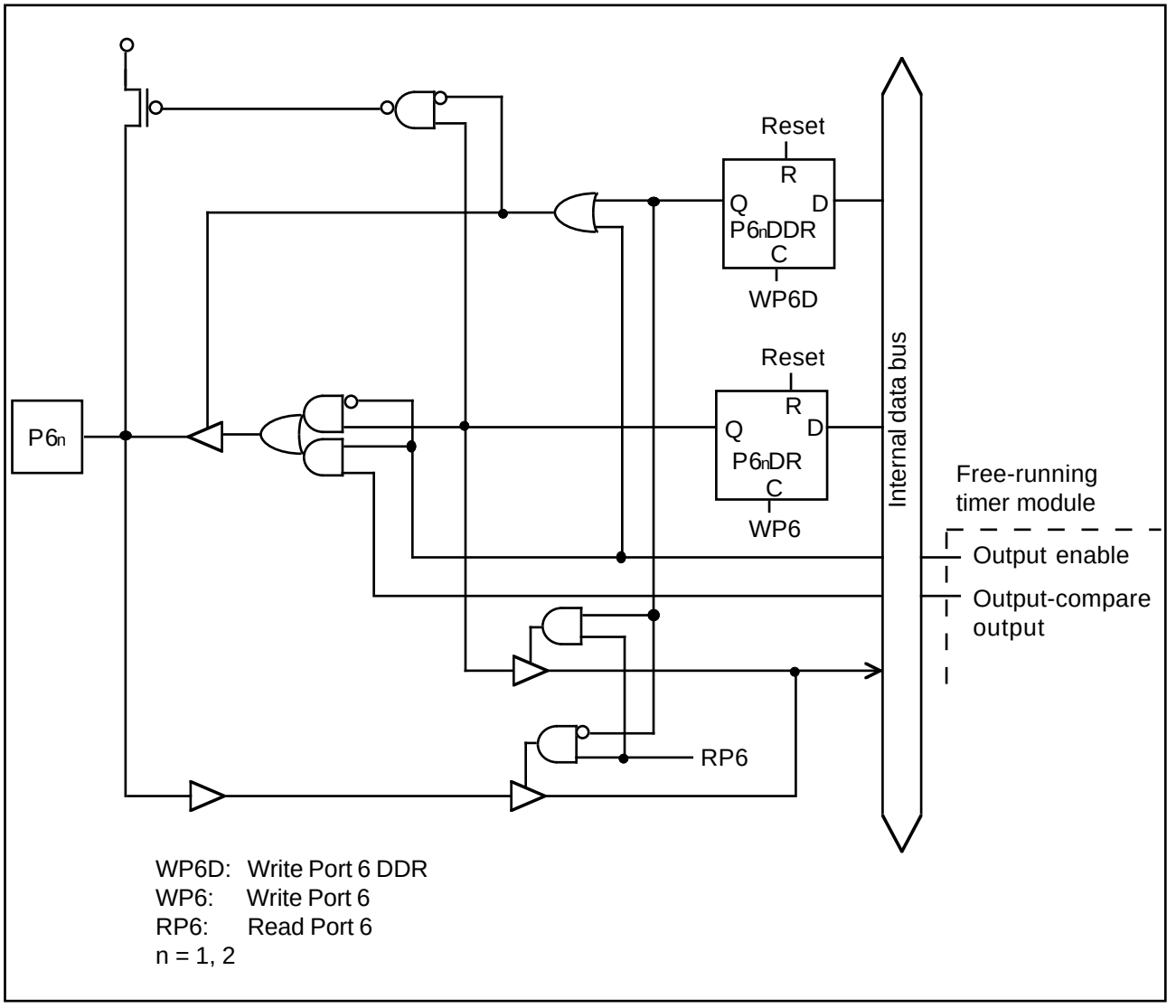



**Figure 5-12. Port 6 Schematic Diagram (Pins P6$_1$ and P6$_2$)**

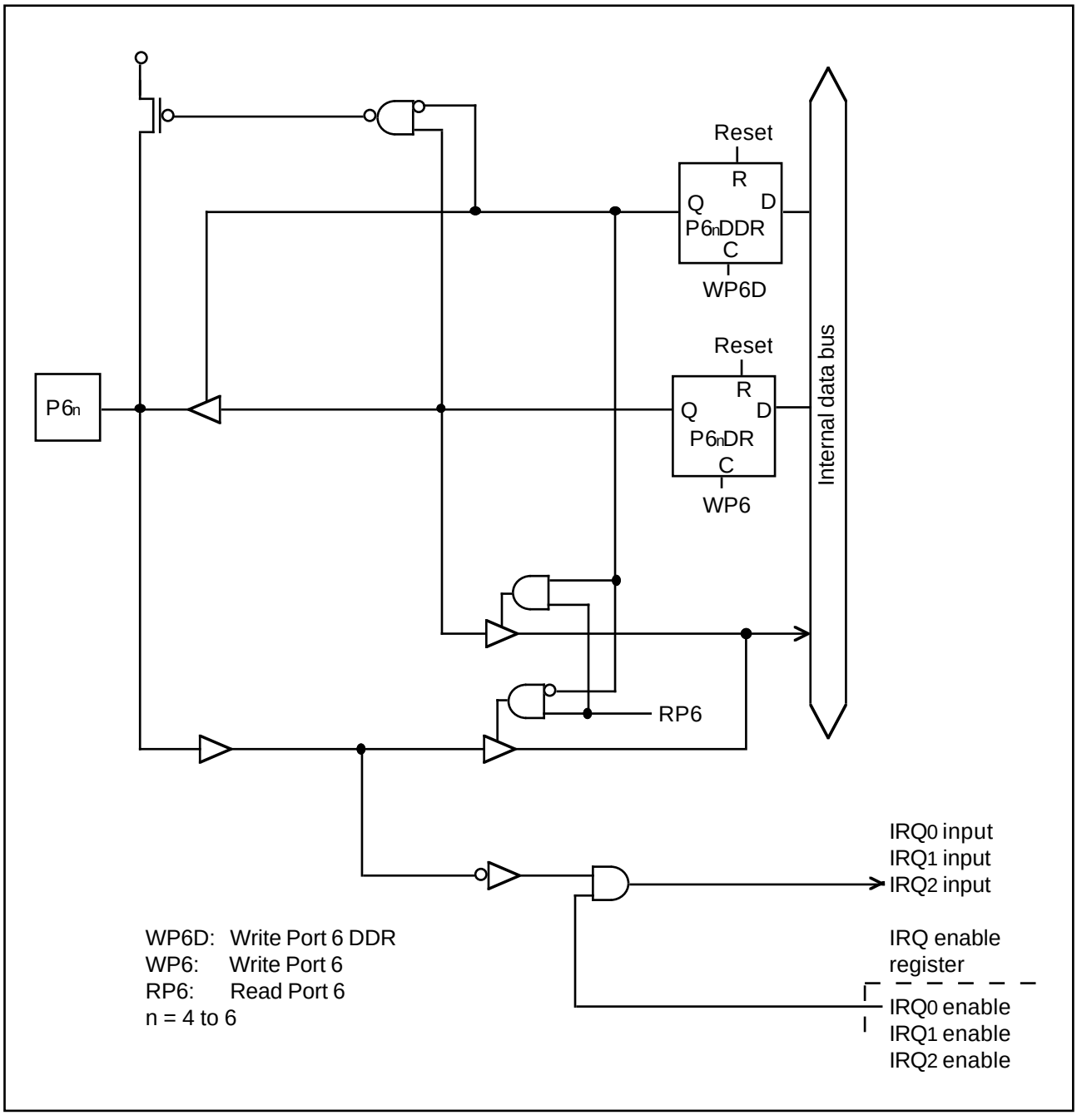



**Figure 5-13. Port 6 Schematic Diagram (Pins P6$_4$, P6$_5$, and P6$_6$)**

## 5.8 Port 7

Port 7 is an 8-bit input/output port that also provides bus control signals (in the expanded modes), and parallel handshaking control signals. Table 5-14 lists the pin functions.

**Table 5-14. Port 7 Pin Functions**

| Pin | Expanded modes | Single-chip mode |
|---|---|---|
| $P7_0$ | $P7_0$ input/output or $\overline{IS}$ input | $P7_0$ input/output or $\overline{IS}$ input |
| $P7_1$ | $P7_1$ input/output | $P7_1$ input/output or $\overline{OS}$ output |
| $P7_2$ | $P7_2$ input/output | $P7_2$ input/output or $\overline{BUSY}$ output |
| $P7_3$ | $P7_3$ input or $\overline{IOS}$ output | $P7_3$ input/output |
| $P7_4$ | $\overline{AS}$ output | $P7_4$ input/output |
| $P7_5$ | $\overline{WR}$ output | $P7_5$ input/output |
| $P7_6$ | $\overline{RD}$ output | $P7_6$ input/output |
| $P7_7$ | $\overline{WAIT}$ input | $P7_7$ input/output |

Pins of port 7 can drive a single TTL load and a 90-pF capacitive load when they are used as output pins.

Table 5-15 details the port 7 registers.

**Table 5-15. Port 7 Registers**

| Name | Abbreviation | Read/Write | Initial value | Address |
|---|---|---|---|---|
| Port 7 data direction register | P7DDR | W | H’00 | H’FFBC |
| Port 7 data register | P7DR | R/W | H’00 | H’FFBE |

**Port 7 Data Direction Register (P7DDR)—H’FFBC**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P7_7$DDR | $P7_6$DDR | $P7_5$DDR | $P7_4$DDR | $P7_3$DDR | $P7_2$DDR | $P7_1$DDR | $P7_0$DDR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

P7DDR is an 8-bit register that selects the direction of each pin in port 7. A pin functions as an output pin if the corresponding bit in P7DDR is set to 1, and as in input pin if the bit is cleared to 0.

**Port 7 Data Register (P7DR)—H’FFBE**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P7_7$ | $P7_6$ | $P7_5$ | $P7_4$ | $P7_3$ | $P7_2$ | $P7_1$ | $P7_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

P7DR is an 8-bit register containing output data for pins $P7_7$ to $P7_0$, and controlling their input pull-ups. When the CPU reads P7DR, for output pins (P7DDR = 1) it reads the value in the P7DR latch, but for input pins (P7DDR = 0), it obtains the logic level directly from the pin, bypassing the P7DR latch. This also applies to pins used for control signal input or output.

**MOS Pull-Ups:** Are available for input pins, including pins used for input of the $\overline{IS}$ and $\overline{WAIT}$ signals. Software can turn the MOS pull-up on by writing a 1 in P7DR, and turn it off by writing a 0. The pull-ups are automatically turned off for output pins.

**Pin $P7_0$:** Can be used for general-purpose input or output, or input of the input strobe ($\overline{IS}$) parallel handshake signal. When $P7_0$ is used for $\overline{IS}$ input, $P7_0$DDR should be cleared to 0, so that output from P7DR will not cause unintended strobes. If input pull-up is not desired, $P7_0$DR should also be cleared to 0.

**Pins $P7_1$ and $P7_2$:** In modes 1 and 2 (expanded modes), these pins can be used for general-purpose input or output.

In mode 3 (single-chip mode), these pins can be used for general-purpose input or output or for output of the $\overline{OS}$ and $\overline{BUSY}$ parallel handshake signals, depending on the OSE and BSE bits in the handshake control/status register. See section 6, Parallel Handshaking Interface, for further information. Pins used for parallel handshaking output are unaffected by the values in P7DDR and P7DR, and their MOS pull-ups are automatically turned off.

**Pin $P7_3$:** In modes 1 and 2 (expanded modes) $P7_3$ is used for $\overline{IOS}$ output if $P7_3$DDR is set to 1, and for general-purpose input if $P7_3$DDR is cleared 0. It cannot be used for general-purpose output.

In mode 3 (single-chip mode), pin $P7_3$ can be used for general-purpose input or output.

**Pins $P7_4$, $P7_5$, and $P7_6$:** In modes 1 and 2 (expanded modes), these pins are used for output of the $\overline{AS}$, $\overline{RD}$, and $\overline{WR}$ bus control signals. They are unaffected by the values in P7DDR and P7DR, and their MOS pull-ups are automatically turned off.

In mode 3 (single-chip mode), these pins can be used for general-purpose input or output.

**Pin $P7_7$:** In modes 1 and 2, this pin is used for input of the $\overline{WAIT}$ bus control signal. It is unaffected by the values in P7DDR and P7DR, except that software can turn on its MOS pull-up by clearing its data direction bit to 0 and setting its data bit to 1.

In mode 3 (single-chip mode), this pin can be used for general-purpose input or output.

**Reset:** In the single-chip mode (mode 3), a reset initializes all pins of port 7 to the general-purpose input state with the MOS pull-ups off. In the expanded modes (modes 1 and 2), $P7_0$ to $P7_3$ are initialized as input port pins, and $P7_4$ to $P7_7$ are initialized to their bus control functions.

**Hardware Standby Mode:** All pins are placed in the high-impedance state with the MOS pull-ups off.

**Software Standby Mode:** All pins remain in their previous state. For $\overline{RD}$, $\overline{WR}$, and $\overline{AS}$ this means the high output state.

Figures 5-14 to 5-18 show schematic diagrams of port 7.

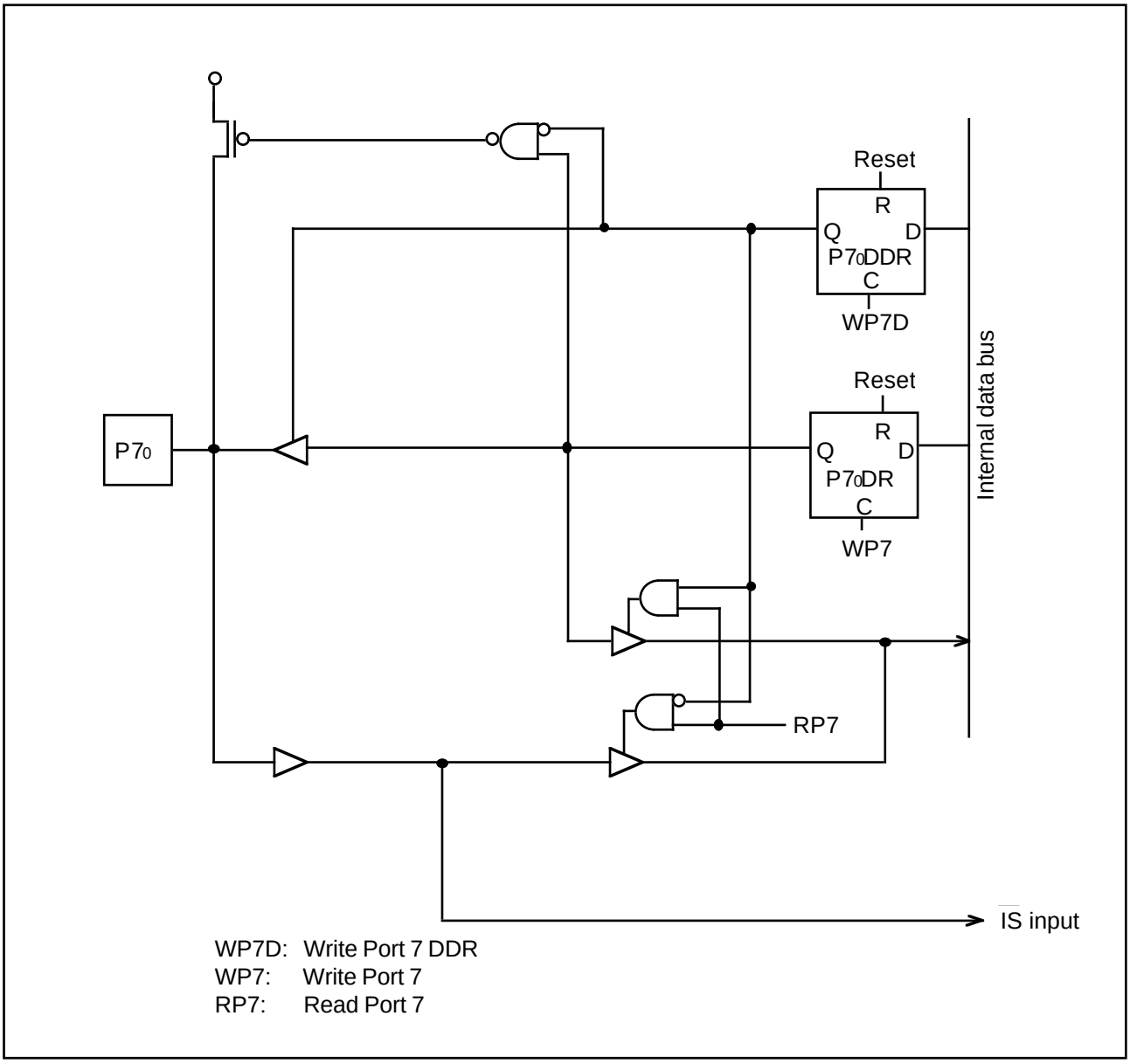



**Figure 5-14. Port 7 Schematic Diagram (Pin P7₀)**

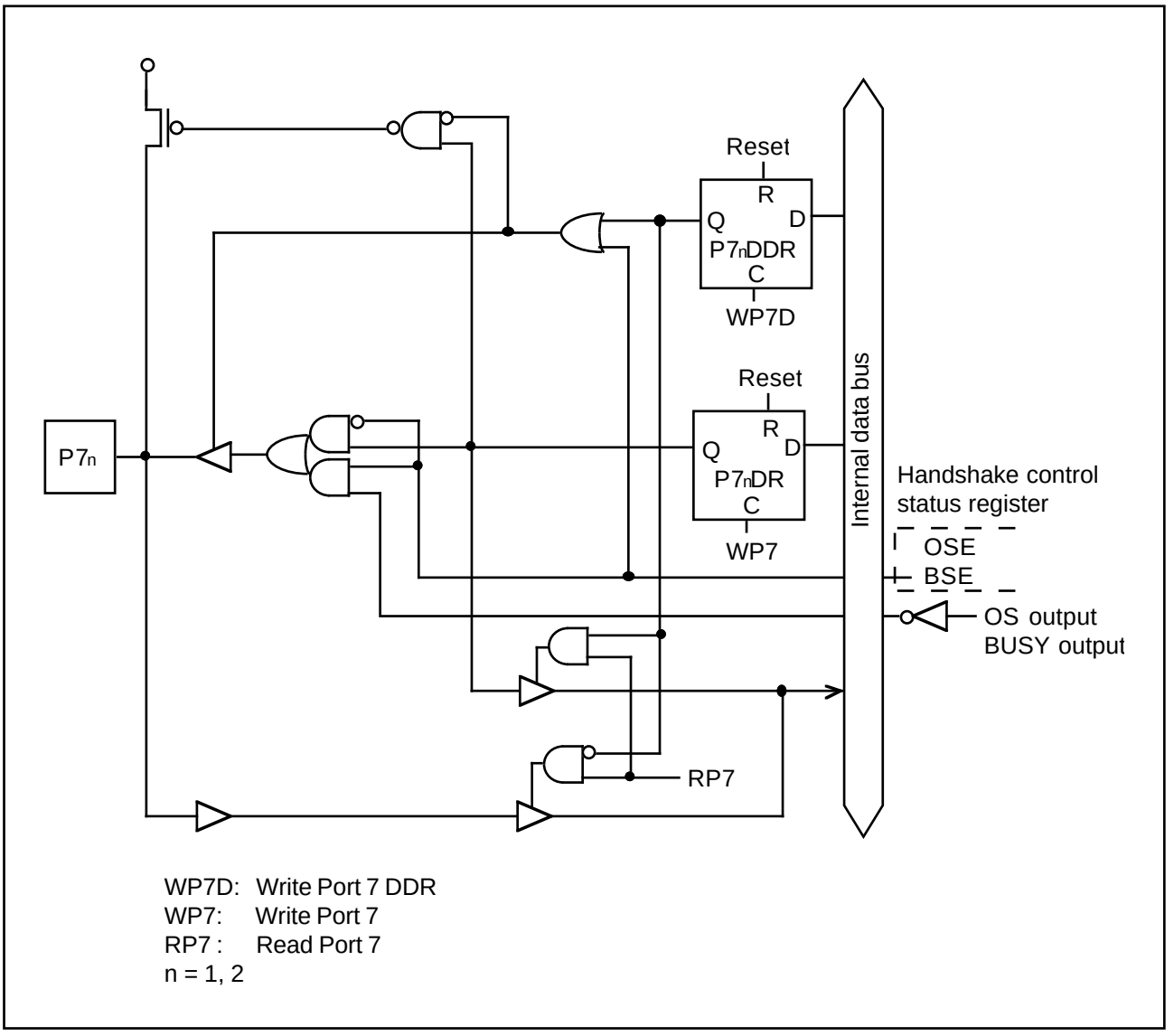



**Figure 5-15. Port 7 Schematic Diagram (Pins P7$_1$ and P7$_2$)**

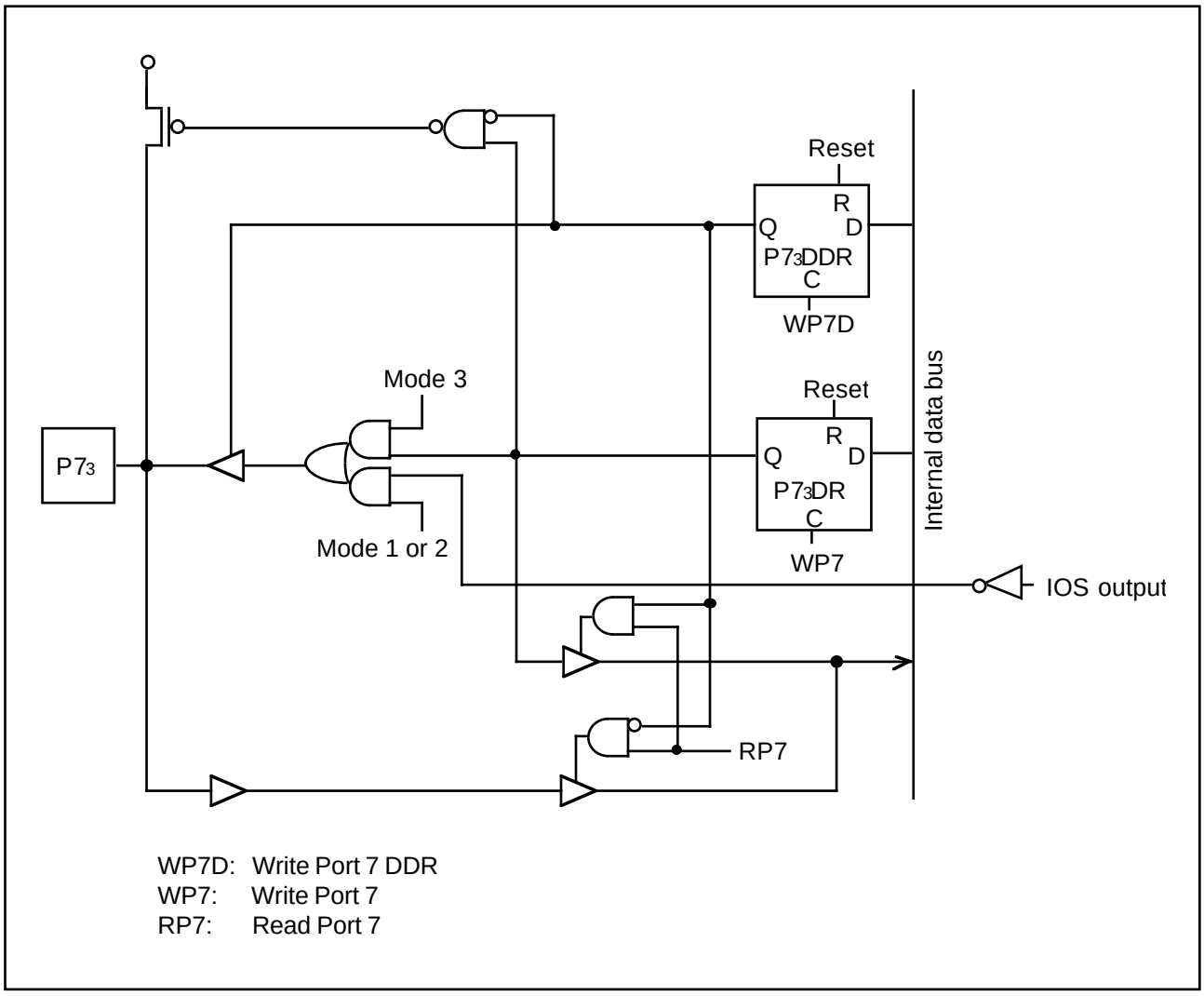



**Figure 5-16. Port 7 Schematic Diagram (Pin P7$_3$)**

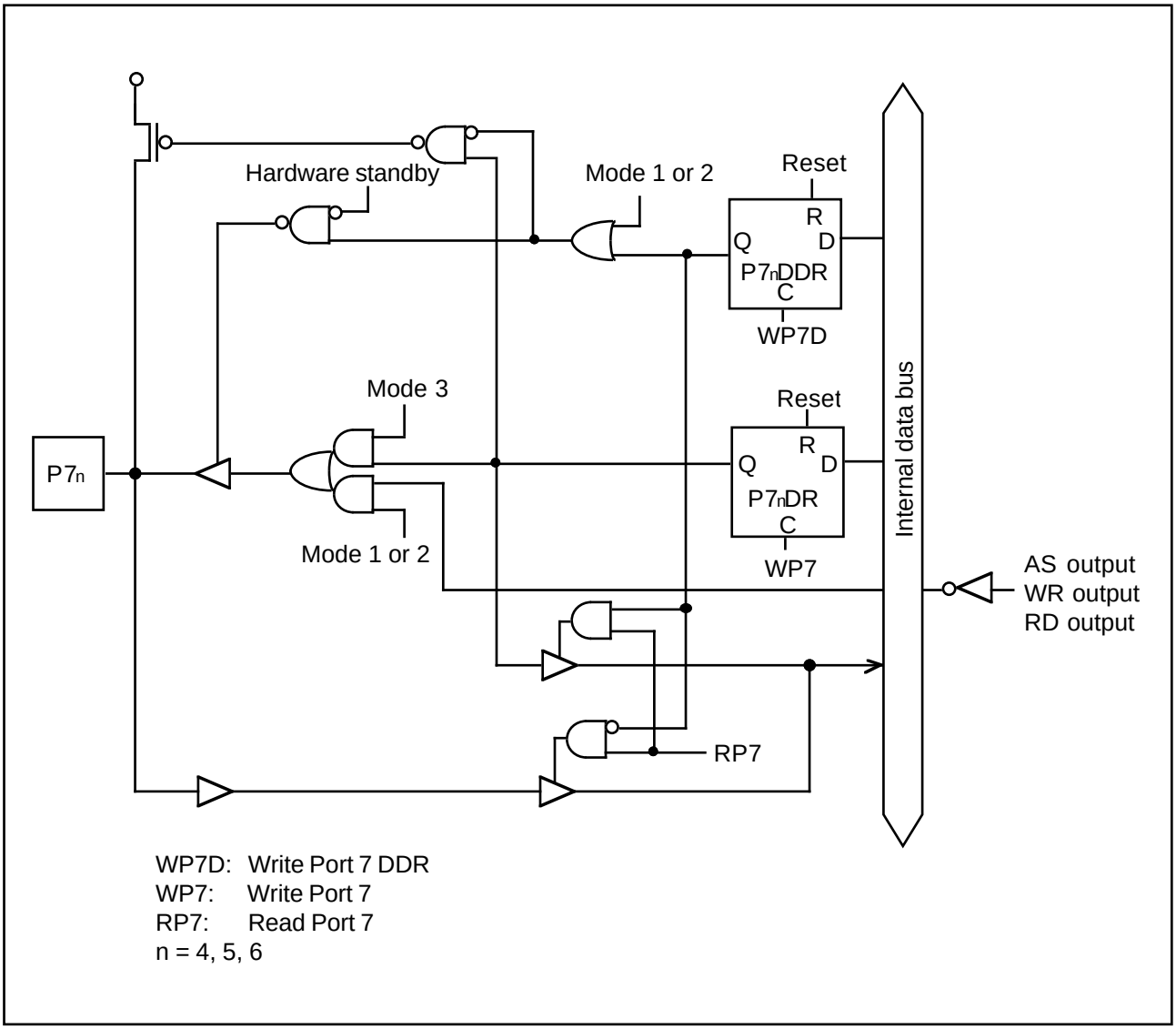



**Figure 5-17. Port 7 Schematic Diagram (Pins P7$_4$, P7$_5$, and P7$_6$)**

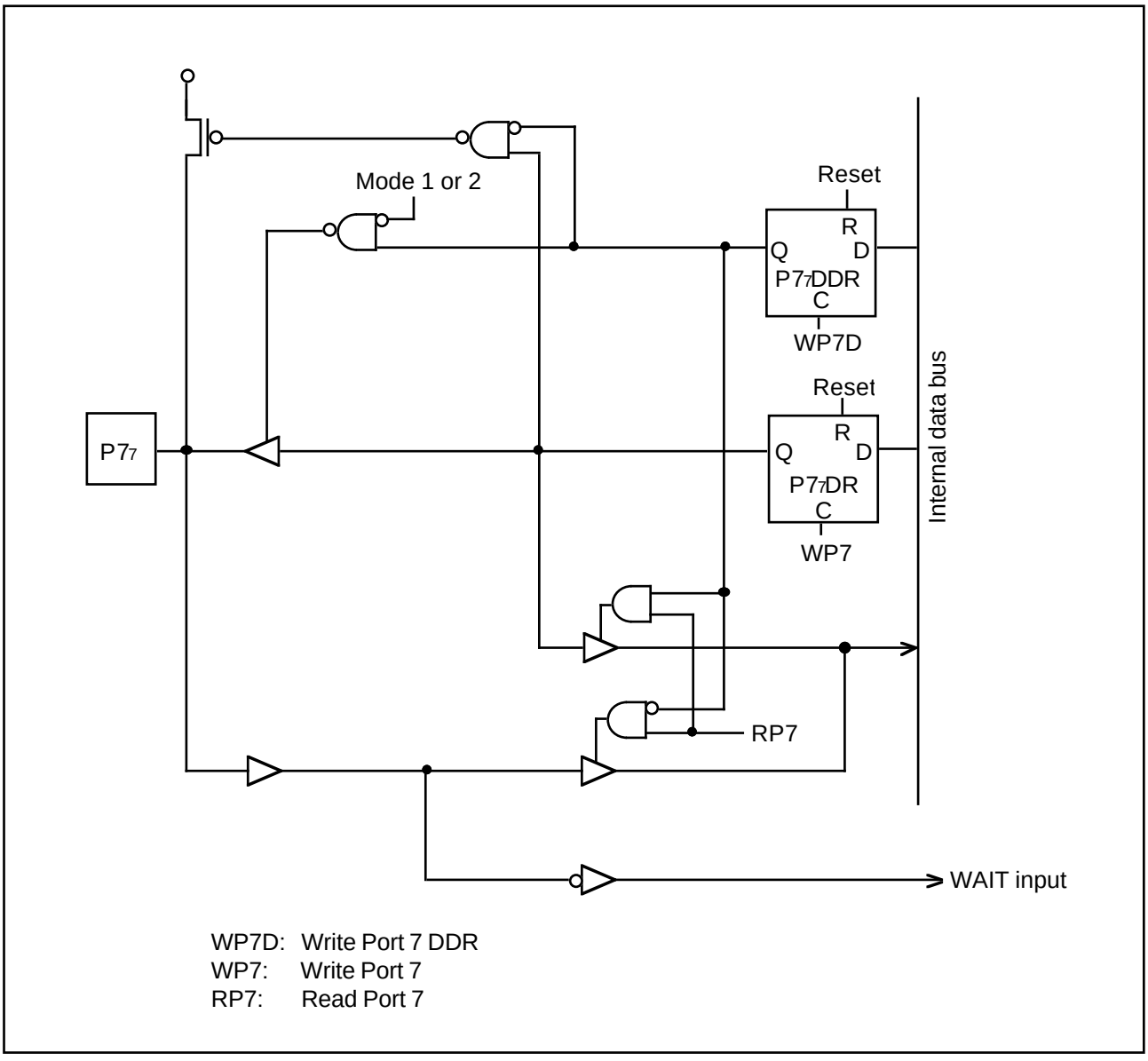



**Figure 5-18. Port 7 Schematic Diagram (Pin P77)**

# Section 6. Parallel Handshaking Interface

## 6.1 Overview

In single-chip mode (mode 3), the H8/325 Series chips can interface to another device by parallel handshaking, using port 3.

### 6.1.1 Features

- Built-in latch circuits
  Data input to port 3 can be latched on the falling edge of the $\overline{IS}$ signal.
- Strobe signal output
  A strobe signal can be output on the $\overline{OS}$ line when port 3 is written or read.
- Busy signal output
  A busy signal is output on the $\overline{BUSY}$ line from the time when data are latched on the falling edge of $\overline{IS}$ until the latched data are read, unlocking the latch.
- Input strobe interrupt
  An input strobe interrupt can be generated at the falling edge of the $\overline{IS}$ signal.
- Recovery from software standby mode
  The input strobe interrupt can be used to recover from software standby mode.

### 6.1.2 Block Diagram

Figure 6-1 is a block diagram of the parallel handshaking interface.

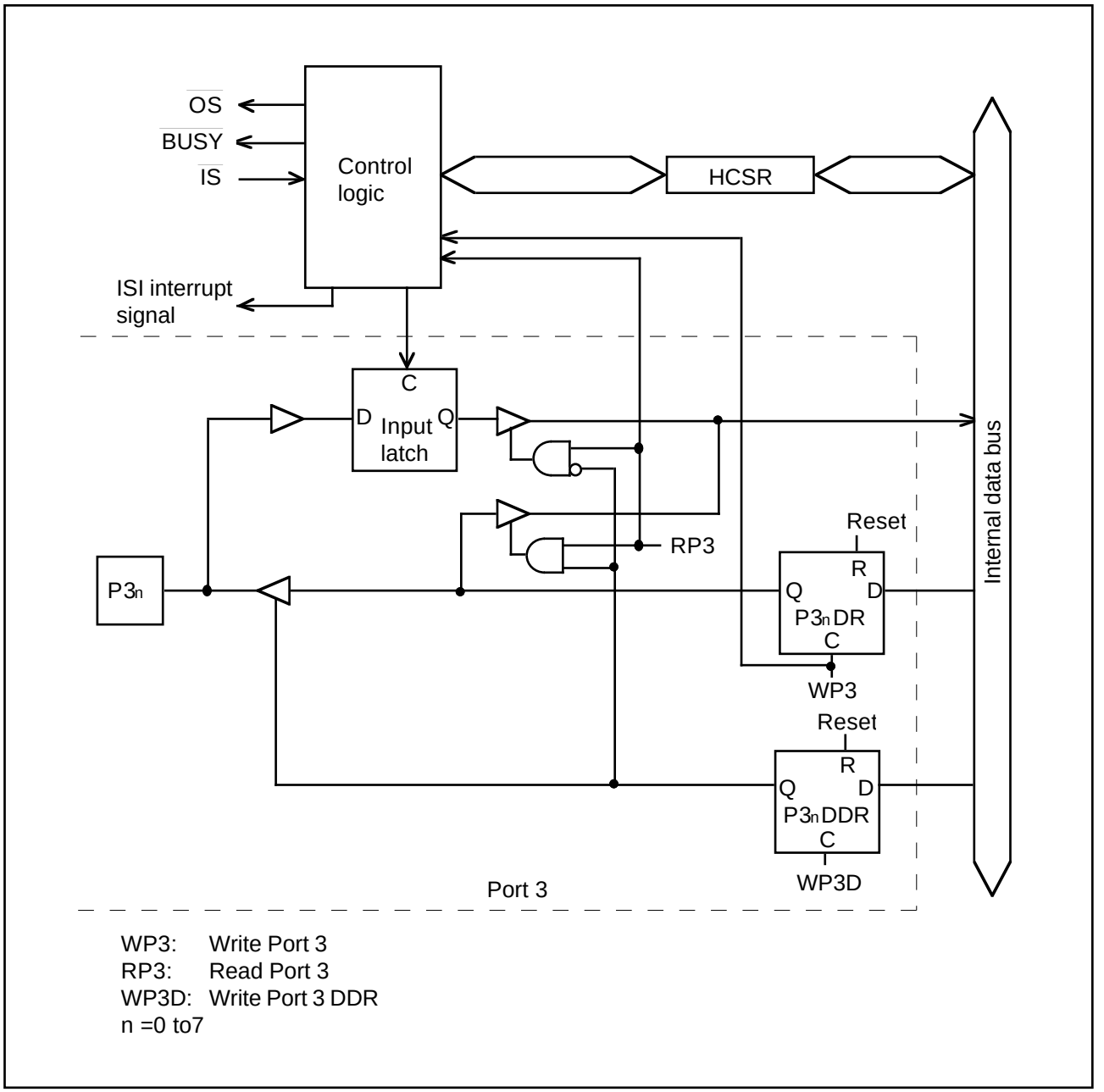



**Figure 6-1. Block Diagram of Parallel Handshaking Interface**

### 6.1.3 Input and Output Pins

Table 6-1 lists the input and output pins used by the parallel handshaking interface.

**Table 6-1. Input and Output Pins of Parallel Handshaking Interface**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Data input/output pins | $P3_7$ – $P3_0$ | I/O | Data input and output |
| Input strobe | $\overline{IS}$ | I | Strobe for input data |
| Output strobe | $\overline{OS}$ | O | Strobe for output data |
| Busy | $\overline{BUSY}$ | O | Busy signal |

### 6.1.4 Register Configuration

Table 6-2 lists information about the parallel handshaking interface registers.

**Table 6-2. Register Configuration**

| Name | Abbreviation | R/W | Initial value | Address |
|---|---|---|---|---|
| Port 3 data direction register | P3DDR | W | H'00 | H'FFB4 |
| Port 3 data register | P3DR | R/W | H'00 | H'FFB6 |
| Handshake control/status register | HCSR | R/W | H'03 | H'FFFE |

## 6.2 Register Descriptions

### 6.2.1 Port 3 Data Direction Register (P3DDR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P3_7$DDR | $P3_6$DDR | $P3_5$DDR | $P3_4$DDR | $P3_3$DDR | $P3_2$DDR | $P3_1$DDR | $P3_0$DDR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

To use the parallel handshaking interface for input, clear P3DDR to H'00. For output, set P3DDR to H'FF. Do not set the bits individually.

See Section 5.4, Port 3 for further information.

### 6.2.2 Port 3 Data Register (P3DR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P3_7$ | $P3_6$ | $P3_5$ | $P3_4$ | $P3_3$ | $P3_2$ | $P3_1$ | $P3_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

When the parallel handshaking interface is used for output (P3DDR = H'FF), P3DR stores the output data. If port 3 is read, the P3DR data are obtained.
When the parallel handshaking interface is used for input (P3DDR = H'00), P3DR has separate latches for reading and writing. The data written in P3DR control the MOS input pull-ups. When P3DR is read, data are obtained from the separate input latches if the input strobe flag (ISF) is set to 1, or directly from the input pins if ISF is cleared to 0.
See Section 5.4, Port 3 for further information.

### 6.2.3 Handshake Control/Status Register (HCSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ISF | ISIE | OSE | OSS | LTE | BSE | — | — |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Read/Write | R | R/W | R/W | R/W | R/W | R/W | — | — |

HCSR is an 8-bit register containing control and status information for parallel handshaking. In the reset and hardware standby modes, HCSR is initialized to H'03. In the software standby mode it retains its previous value.

**Bit 7—Input Strobe Flag (ISF):** Indicates that the input strobe signal ($\overline{IS}$) has gone low.
ISF is a read-only bit that is set and cleared by hardware. It is set by strobe input. It is cleared when the port 3 data register is written or read. (The handshake control/status register must be read first.)

**Bit 7**

| ISF | Description |
|---|---|
| 0 | To clear ISF, the CPU must read HCSR after ISF has been set to 1, then read or write the port 3 data register (P3DR). (Initial value) |
| 1 | ISF is set to 1 on the falling edge of $\overline{IS}$. |

**Bit 6—Input Strobe Interrupt Enable (ISIE):** Enables or disables the handshake interrupt request (ISI).

**Bit 6**

| ISIE | Description | |
|---|---|---|
| 0 | The handshake interrupt request (ISI) is disabled. | (Initial value) |
| 1 | The handshake interrupt request (ISI) is enabled. | |

**Bit 5—Output Strobe Enable (OSE):** Enables or disables output of the output strobe signal. Do not set OSE to 1 in the expanded modes (modes 1 and 2).

**Bit 5**

| OSE | Description | |
|---|---|---|
| 0 | The output strobe signal is disabled. | (Initial value) |
| 1 | The output strobe signal is enabled. | |

**Bit 4—Output Strobe Select (OSS):** Selects whether to generate an output strobe signal when the port 3 data register (P3DR) is written, or when it is read.

**Bit 4**

| OSS | Description | |
|---|---|---|
| 0 | An output strobe signal is output when P3DR is read. | (Initial value) |
| 1 | An output strobe signal is output when P3DR is written. | |

**Bit 3—Latch Enable (LTE):** Controls the input latches of port 3. Do not set LTE to 1 in the expanded modes (modes 1 and 2).
When LTE is set to 1, input data are latched on the falling edge of $\overline{IS}$. The data are retained in the input latch until the port 3 data register (P3DR) is read, after which the next data can be latched.

**Bit 3**

| LTE | Description | |
|---|---|---|
| 0 | Port 3 input data are not latched. | (Initial value) |
| 1 | Port 3 input data are latched on the falling edge of $\overline{IS}$. | |

**Bit 2—Busy Enable (BSE):** This bit enables or disables output of the busy signal. Do not set BSE to 1 in the expanded modes (modes 1 and 2).

**Bit 2**

| ISIE | Description | |
|---|---|---|
| 0 | Busy signal output is disabled. | (Initial value) |
| 1 | Busy signal output is enabled. | |

**Bits 1 and 0—Reserved:** These bits cannot be modified and are always read as 1.

## 6.3 Operation

### 6.3.1 Output Timing of Output Strobe Signal

The output strobe signal is output when the port 3 data register (P3DR) is written or read. The output strobe signal goes low at the seventh system clock cycle after P3DR is written or read, remains low for eight system clock cycles, then goes high. Figure 6-2 shows how the output strobe signal is output after P3DR is written (when OSS = 1).
Note the following point when reading or writing P3DR twice consecutively.
If P3DR is written or read once, then written or read again within 15 states, the output strobe signal is not output for the second write or read. Figure 6-3 shows an example of this when OSS = 1.

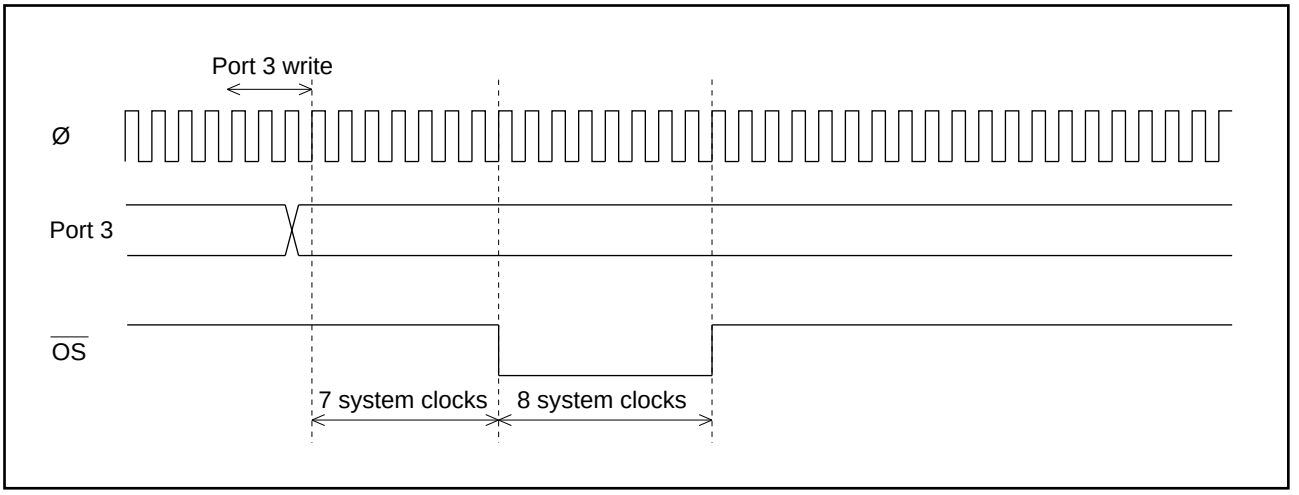



**Figure 6-2. Output Strobe Output Timing (When OSS = 1)**

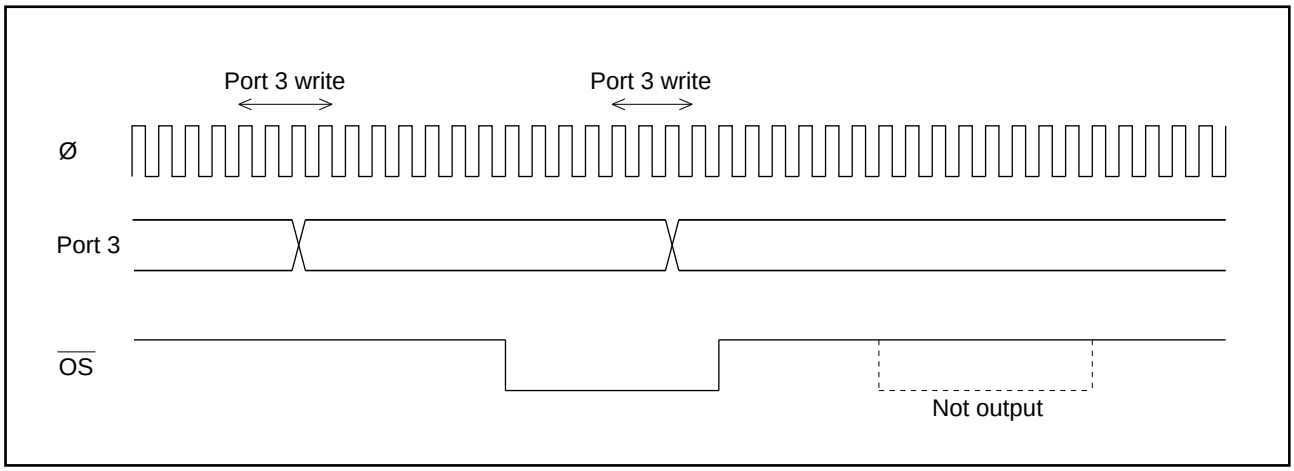



**Figure 6-3. Output Strobe Output Timing**
**(Consecutive Writing of Port 3 When OSS = 1)**

### 6.3.2 Busy Signal Output Timing

The busy signal remains low from the fall of the input strobe signal until the data latched in port 3 have been read, unlocking the latch. Figure 6-4 shows an example.
While the busy signal is low, data input to port 3 are not latched, even if the input strobe signal goes low again.

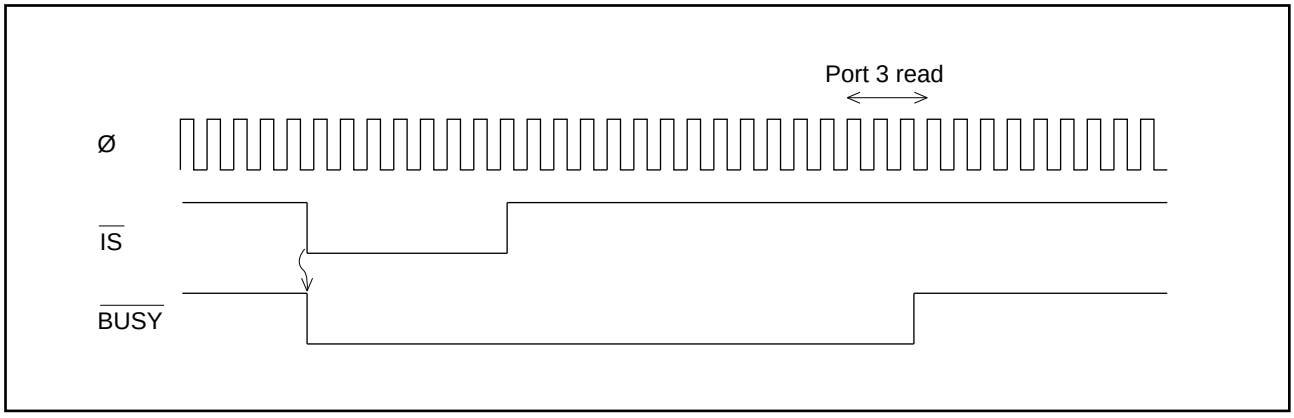



**Figure 6-4. Busy Signal Output Timing**

### 6.3.3 Operation in Software Standby Mode

In software standby mode, the $\overline{OS}$ and $\overline{BUSY}$ output pins retain their previous states.
For timing of the output strobe signal, the entire time during when the chip is in software standby mode is counted as zero system clock cycles. Figure 6-5 shows an example.

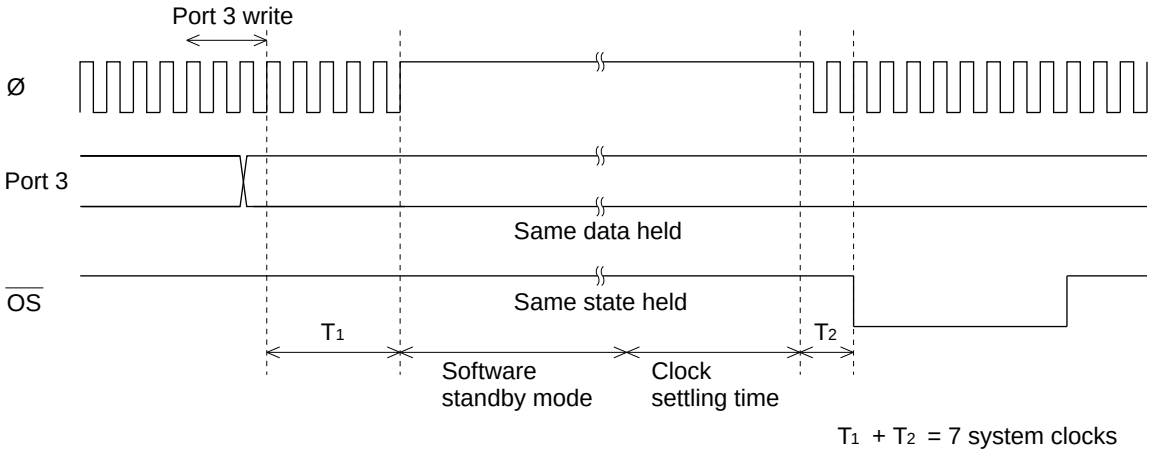



**Figure 6-5. Output Strobe Timing in Software Standby Mode**

When the ISIE and LTE bits in the handshake control/status register (HCSR) are both set to 1, if a high-to-low transition of the $\overline{IS}$ signal occurs during software standby mode, an input strobe interrupt is requested and the chip recovers from software standby mode to handle the interrupt.

If the parallel handshaking interface is set for input, the port 3 input data are also latched.

If either the ISIE or LTE bit is cleared to 0, then high-to-low transitions of the $\overline{IS}$ signal are ignored during software standby mode.

### 6.3.4 Sample Application

Figure 6-6 shows an example in which the parallel handshaking interface is used to interconnect two H8/325 chips. Figure 6-7 shows the interface timing.

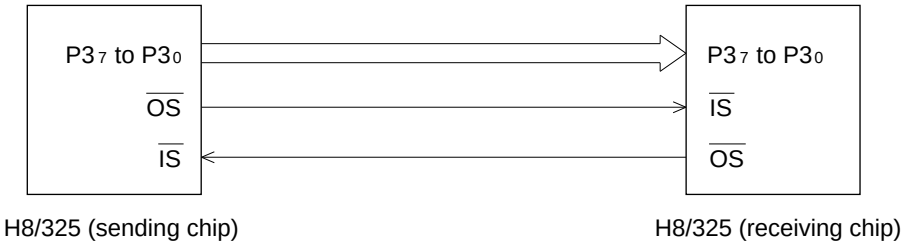



**Figure 6-6. Sample Usage of Parallel Handshaking Interface**

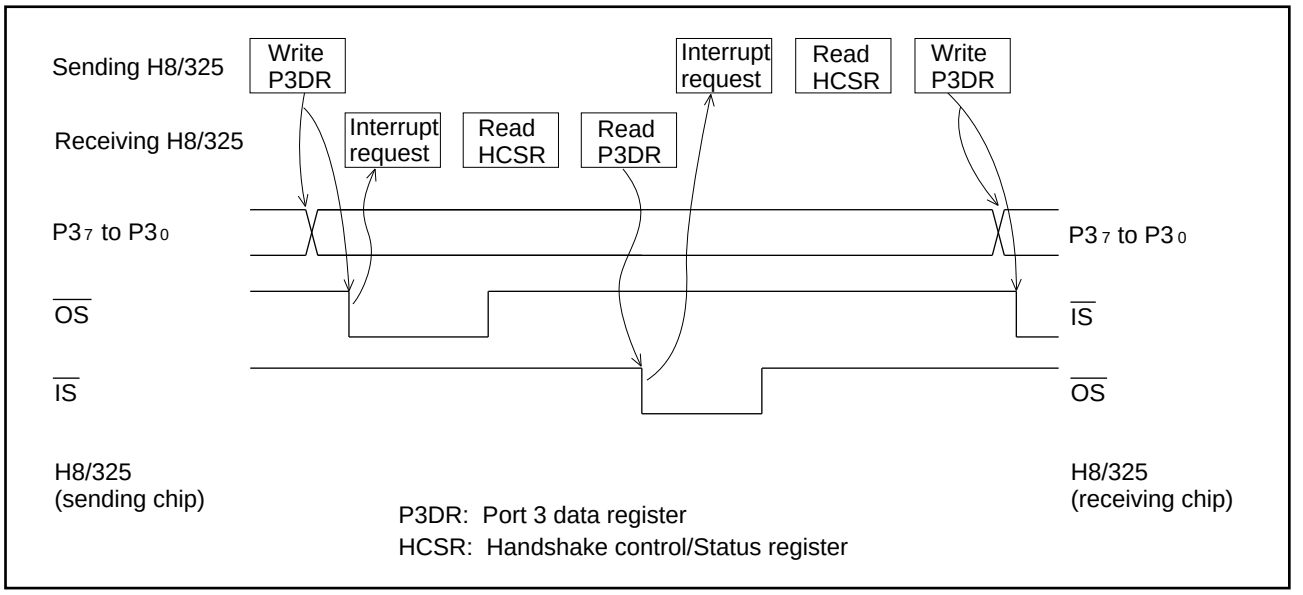



**Figure 6-7. Parallel Handshaking Interface Timing Chart (Example)**

1. The sending and receiving H8/325s set their HCSR bits as follows:
   Sending H8/325: ISIE = 1, OSE = 1, OSS = 1, LTE = 0, BSE = 0.
   Receiving H8/325: ISIE = 1, OSE = 1, OSS = 0, LTE = 1, BSE = 0.
2. The sending H8/325 writes the transmit data in the port 3 data register (P3DR). This generates an output strobe signal, notifying the receiving H8/325 of data output.
3. The receiving H8/325 receives the strobe on its input strobe line and latches the data in port 3. ISF is set to 1, generating an input strobe interrupt.
4. The receiving H8/325 reads HCSR, then reads the received data from P3DR. This clears ISF to 0 and generates an output strobe signal, notifying the sending H8/325 that the data have been received.
5. The input strobe line of the sending H8/325 goes low, setting ISF and generating an input strobe interrupt.
6. The sending H8/325 reads HCSR, then writes the next transmit data in P3DR. (If it has no next data to send, it should read P3DR.) This clears ISF to 0 and generates an output strobe signal. The process now returns to step 3.

### 6.3.5 Interrupts

Regardless of the operating mode or the value of the LTE bit, ISF is always set to 1 when the $\overline{IS}$ input changes from high to low. If ISIE is set to 1, an input strobe interrupt (ISI) is requested. In the software standby mode, LTE must also be set. See section 6.3.3, Operation in Software Standby Mode.

# Section 7. 16-Bit Free-Running Timer

## 7.1 Overview

The H8/325 Series has an on-chip 16-bit free-running timer (FRT) module that uses a 16-bit free-running counter as a time base. Applications of the FRT module include rectangular-wave output (up to two independent waveforms), input pulse width measurement, and measurement of external clock periods.

### 7.1.1 Features

The features of the free-running timer module are listed below.

- Selection of four clock sources
  The free-running counter can be driven by an internal clock source (Ø/2, Ø/8, or Ø/32), or an external clock input (enabling use as an external event counter).
- Two independent comparators
  Each comparator can generate an independent waveform.
- Input capture
  The current count can be captured on the rising or falling edge (selectable) of an input signal.
- Counter can be cleared under program control
  The free-running counter can be cleared on compare-match A.
- Four interrupt sources
  Compare-match A and B, input capture, and overflow interrupts are requested independently.
- Noise canceler
  A built-in noise canceler can remove high-frequency noise from the pulse signal input at the input capture pin.

### 7.1.2 Block Diagram

Figure 7-1 shows a block diagram of the free-running timer.

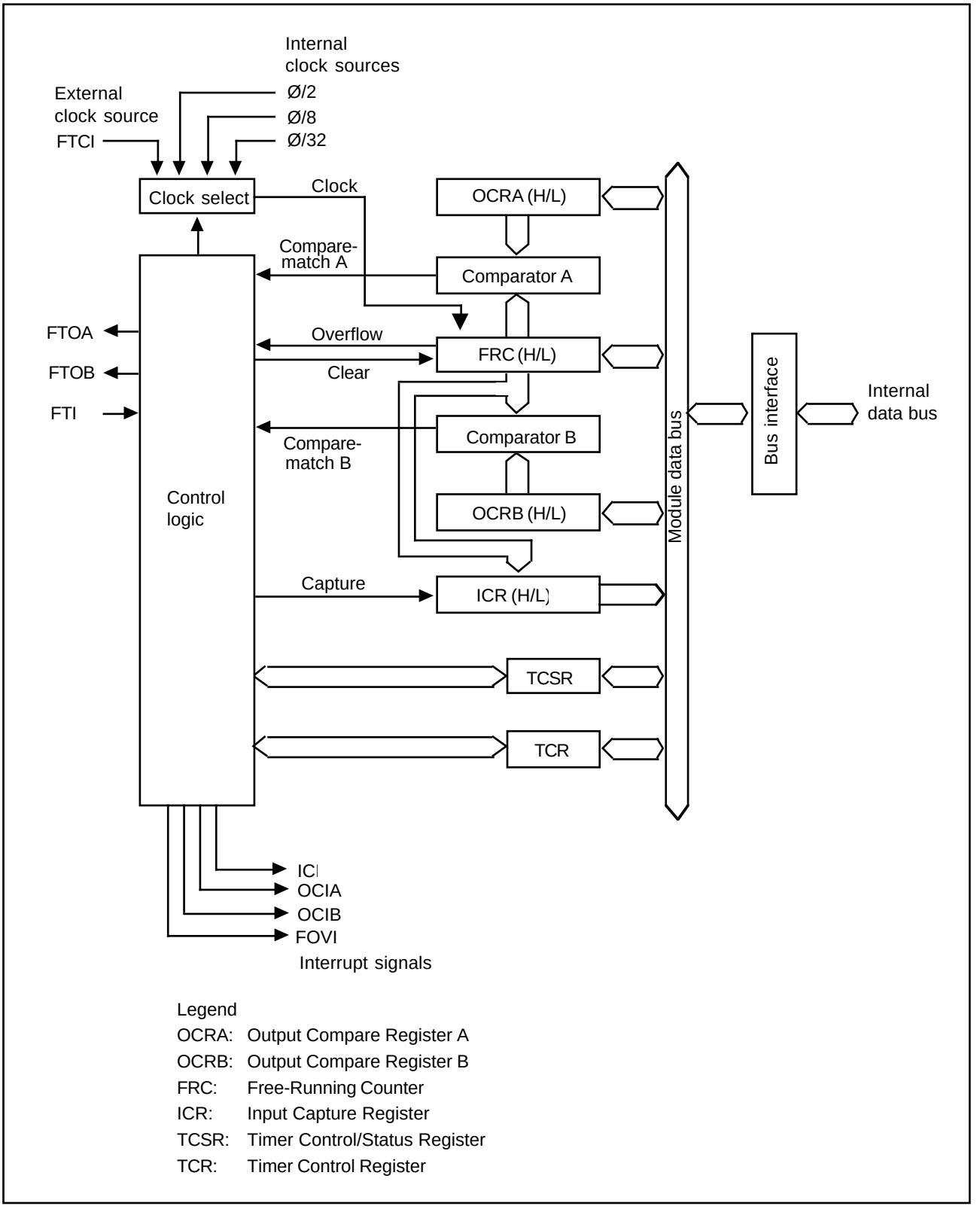



**Figure 7-1. Block Diagram of 16-Bit Free-Running Timer**

### 7.1.3 Input and Output Pins

Table 7-1 lists the input and output pins of the free-running timer module.

**Table 7-1. Input and Output Pins of Free-Running Timer Module**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Counter clock input | FTCI | Input | Input of external free-running counter clock signal |
| Output compare A | FTOA | Output | Output controlled by comparator A |
| Output compare B | FTOB | Output | Output controlled by comparator B |
| Input capture | FTI | Input | Input capture trigger |

### 7.1.4 Register Configuration

Table 7-2 lists the registers of the free-running timer module.

**Table 7-2. Register Configuration**

| Name | Abbreviation | R/W | Initial value | Address |
|---|---|---|---|---|
| Timer control register | TCR | R/W | H’00 | H’FF90 |
| Timer control/status register | TCSR | R/(W)* | H’00 | H’FF91 |
| Free-running counter (high) | FRC (H) | R/W | H’00 | H’FF92 |
| Free-running counter (low) | FRC (L) | R/W | H’00 | H’FF93 |
| Output compare register A (high) | OCRA (H) | R/W | H’FF | H’FF94 |
| Output compare register A (low) | OCRA (L) | R/W | H’FF | H’FF95 |
| Output compare register B (high) | OCRB (H) | R/W | H’FF | H’FF96 |
| Output compare register B (low) | OCRB (L) | R/W | H’FF | H’FF97 |
| Input capture register (high) | ICR (H) | R | H’00 | H’FF98 |
| Input capture register (low) | ICR (L) | R | H’00 | H’FF99 |
| FRT noise canceler control register | FNCR | R/W | H'FC | H’FFFF |

* Software can write a 0 to clear bits 7 to 4, but cannot write a 1 in these bits.

## 7.2 Register Descriptions

### 7.2.1 Free-Running Counter (FRC) – H'FF92

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/ Write | 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 |

The FRC is a 16-bit readable/writable up-counter that increments on an internal pulse generated from a clock source. The clock source is selected by the clock select 1 and 0 bits (CKS1 and CKS0) of the timer control register (TCR).

When the FRC overflows from H'FFFF to H'0000, the overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

Because the FRC is a 16-bit register, a temporary register (TEMP) is used when the FRC is written or read. See section 7.3, CPU Interface for details.

The FRC is initialized to H'0000 at a reset and in the standby modes. It can also be cleared by compare-match A.

### 7.2.2 Output Compare Registers A and B (OCRA and OCRB) – H'FF94 and H'FF96

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/ Write | 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 |

OCRA and OCRB are 16-bit readable/writable registers, the contents of which are continually compared with the value in the FRC. When a match is detected, the corresponding output compare flag (OCFA or OCFB) is set in the timer control/status register (TCSR).

In addition, if the output enable bit (OEA or OEB) in the timer output compare control register (TCR) is set to 1, when the output compare register and FRC values match, the logic level selected by the output level bit (OLVLA or OLVLB) in the TCSR is output at the output compare pin (FTOA or FTOB).

Because OCRA and OCRB are 16-bit registers, a temporary register (TEMP) is used for write access, as explained in section 7.3, CPU Interface.

OCRA and OCRB are initialized to H'FFFF at a reset and in the standby modes.

### 7.2.3 Input Capture Register (ICR) – H'FF98

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/ Write | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

The input capture register is a 16-bit read-only register.

When the rising or falling edge of the signal at the input capture pin (FTI) is detected, the current value of the FRC is copied to the input capture register (ICR). At the same time, the input capture flag (ICF) in the timer control/status register (TCSR) is set to 1. The input capture edge is selected by the input edge select bit (IEDG) in the TCSR.

Because the input capture register is a 16-bit register, a temporary register (TEMP) is used when it is read. See Section 7.3, CPU Interface for details.

To ensure input capture, when the noise canceler is not used, the width of the input capture pulse (FTI) should be at least 1.5 system clock cycles (1.5·Ø).

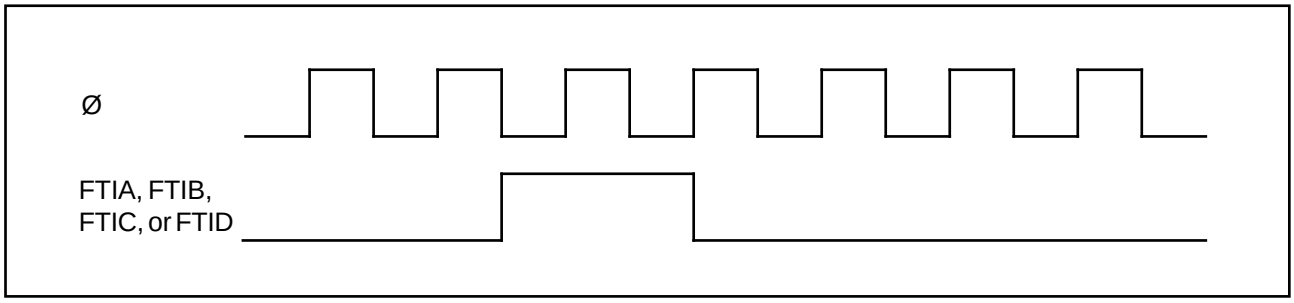



**Figure 7-2. Minimum Input Capture Pulse Width (Noise Canceler Disabled)**

The input capture register is initialized to H'0000 at a reset and in the standby modes.

**Note:** When input capture is detected, the FRC value is transferred to the input capture register even if the input capture flag is already set.

### 7.2.4 Timer Control Register (TCR) – H'FF90

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ICIE | OCIEB | OCIEA | OVIE | OEB | OEA | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The TCR is an 8-bit readable/writable register that enables and disables output signals and interrupts, and selects the timer clock source.

The TCR is initialized to H'00 at a reset and in the standby modes.

**Bit 7 – Input Capture Interrupt Enable (ICIE):** Selects whether to request an input capture interrupt (ICI) when the input capture flag (ICF) in the timer status/control register (TCSR) is set to 1.

**Bit 7**

| **ICIE** | **Description** | |
|---|---|---|
| 0 | Input capture interrupt request (ICI) is disabled. | (Initial value) |
| 1 | Input capture interrupt request (ICI) is enabled. | |

**Bit 6 – Output Compare Interrupt B Enable (OCIBE):** Selects whether to request output compare interrupt B (OCIB) when output compare flag B (OCFB) in the timer status/control register (TCSR) is set to 1.

**Bit 6**

| OCIBE | Description | |
|---|---|---|
| 0 | Output compare interrupt request B (OCIB) is disabled. | (Initial value) |
| 1 | Output compare interrupt request B (OCIB) is enabled. | |

**Bit 5 – Output Compare Interrupt A Enable (OCIAE):** Selects whether to request output compare interrupt A (OCIA) when output compare flag A (OCFA) in the timer status/control register (TCSR) is set to 1.

**Bit 5**

| OCIAE | Description | |
|---|---|---|
| 0 | Output compare interrupt request A (OCIA) is disabled. | (Initial value) |
| 1 | Output compare interrupt request A (OCIA) is enabled. | |

**Bit 4 – Timer overflow Interrupt Enable (OVIE):** Selects whether to request a free-running timer overflow interrupt (FOVI) when the timer overflow flag (OVF) in the timer status/control register (TCSR) is set to 1.

**Bit 4**

| OVIE | Description | |
|---|---|---|
| 0 | Timer  overflow interrupt request (FOVI) is disabled. | (Initial value) |
| 1 | Timer  overflow interrupt request (FOVI) is enabled. | |

**Bit 3 – Output Enable B (OEB):** Enables or disables output of the output compare B signal (FTOB). If output compare B is enabled, the FTOB pin is driven to the level selected by OLVLB in the timer status/control register (TCSR) whenever the FRC value matches the value in output compare register B (OCRB).

**Bit 3**

| OEB | Description | |
|---|---|---|
| 0 | Output compare B output is disabled. | (Initial value) |
| 1 | Output compare B output is enabled. | |

**Bit 2 – Output Enable A (OEA):** Enables or disables output of the output compare A signal (FTOA). If output compare A is enabled, the FTOA pin is driven to the level selected by OLVLA in the timer status/control register (TCSR) whenever the FRC value matches the value in output compare register A (OCRA).

| Bit 2<br>OEA | Description | |
|---|---|---|
| 0 | Output compare A output is disabled. | (Initial value) |
| 1 | Output compare A output is enabled. | |

**Bits 1 and 0 – Clock Select (CKS1 and CKS0):** These bits select external clock input or one of three internal clock sources for the FRC. External clock pulses are counted on the rising edge.

| Bit 1<br>CKS1 | Bit 0<br>CKS0 | Description | |
|---|---|---|---|
| 0 | 0 | Ø/2 Internal clock source | (Initial value) |
| 0 | 1 | Ø/8 Internal clock source | |
| 1 | 0 | Ø/32 Internal clock source | |
| 1 | 1 | External clock source (rising edge) | |

### 7.2.5 Timer Control/Status Register (TCSR) – H'FF91

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ICF | OCFB | OCFA | OVF | OLVLB | OLVLA | IEDG | CCLRA |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W) | R/(W) | R/(W) | R/W |

* Software can write a 0 in bits 7 to 4 to clear the flags, but cannot write a 1 in these bits.

The TCSR is an 8-bit readable and partially writable register that contains the four interrupt flags and selects the output compare levels, input capture edge, and whether to clear the counter on compare-match A.

The TCSR is initialized to H'00 at a reset and in the standby modes.

**Bit 7 – Input Capture Flag (ICF):** This status bit is set to 1 to flag an input capture event, indicating that the FRC value has been copied to the ICR.

ICF must be cleared by software. It is set by hardware, however, and cannot be set by software.

**Bit 7**

| ICF | Description | |
|---|---|---|
| 0 | To clear ICF, the CPU must read ICF after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when an FTI input signal causes the FRC value to be copied to the ICR. | |

**Bit 6 – Output Compare Flag B (OCFB):** This status flag is set to 1 when the FRC value matches the OCRB value.

This flag must be cleared by software. It is set by hardware, however, and cannot be set by software.

**Bit 6**

| OCFB | Description | |
|---|---|---|
| 0 | To clear OCFB, the CPU must read OCFB after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when FRC = OCRB. | |

**Bit 5 – Output Compare Flag A (OCFA):** This status flag is set to 1 when the FRC value matches the OCRA value.

This flag must be cleared by software. It is set by hardware, however, and cannot be set by software.

**Bit 5**

| OCFA | Description | |
|---|---|---|
| 0 | To clear OCFA, the CPU must read OCFA after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when FRC = OCRA. | |

**Bit 4 – Timer Overflow Flag (OVF):** This status flag is set to 1 when the FRC overflows (changes from H’FFFF to H’0000).

This flag must be cleared by software. It is set by hardware, however, and cannot be set by software.

**Bit 4**

| OVF | Description | |
|---|---|---|
| 0 | To clear OVF, the CPU must read OVF after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when FRC changes from H'FFFF to H'0000. | |

**Bit 3 – Output Level B (OLVLB):** Selects the logic level output at the FTOB pin when the FRC and OCRB values match.

**Bit 3**

| OLVLB | Description | |
|---|---|---|
| 0 | A 0 logic level is output for compare-match B. | (Initial value) |
| 1 | A 1 logic level is output for compare-match B. | |

**Bit 2 – Output Level A (OLVLA):** Selects the logic level output at the FTOA pin when the FRC and OCRA values match.

**Bit 2**

| OLVLA | Description | |
|---|---|---|
| 0 | A 0 logic level is output for compare-match A. | (Initial value) |
| 1 | A 1 logic level is output for compare-match A. | |

**Bit 1 – Input Edge Select (IEDG):** Selects the rising or falling edge of the input capture signal (FTI).

**Bit 1**

| IEDG | Description | |
|---|---|---|
| 0 | FRC contents are transferred to ICR on the falling edge of FTI. | (Initial value) |
| 1 | FRC contents are transferred to ICR on the rising edge of FTI. | |

**Bit 0 – Counter Clear A (CCLRA):** Selects whether to clear the FRC at compare-match A (when the FRC and OCRA values match).

**Bit 0**

| CCLRA | Description | |
|---|---|---|
| 0 | The FRC is not cleared. | (Initial value) |
| 1 | The FRC is cleared at compare-match A. | |

### 7.2.6 FRT Noise Canceler Control Register (FNCR) – H'FFFF

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | NCS1 | NCS0 |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Read/Write | — | — | — | — | — | — | R/W | R/W |

The FNCR is an 8-bit readable/writable register that controls the input capture noise canceler.

The FNCR is initialized to H'FC at a reset and in the standby modes.

**Bits 7 to 2 – Reserved:** These bits cannot be modified, and are always read as 1.

**Bits 1 and 0 – Noise Canceler Select 1 and 0 (NCS1 and NCS0):** Select the sampling clock provided to the noise canceler. Three internal clock rates can be selected.

The noise canceler recognizes a level change only if it is observed in four consecutive samples. When the noise canceler is enabled, the input capture pulse width must be at least four sampling clock cycles. See section 7.6, Noise Canceler for further information.

The noise canceler can be disabled by clearing both NCS1 and NCS0 to 0. The input capture pulse width must then be at least 1.5 system clock cycles (1.5·Ø) to assure capture.

| Bit 1 NCS1 | Bit 0 NCS0 | Description | |
|---|---|---|---|
| 0 | 0 | Noise canceler is disabled. | (Initial value) |
| 0 | 1 | Sampling clock frequency: Ø/32 | |
| 1 | 0 | Sampling clock frequency: Ø/64 | |
| 1 | 1 | Sampling clock frequency: Ø/128 | |

## 7.3 CPU Interface

The free-running counter (FRC), output compare registers (OCRA and OCRB), and input capture register (ICR) are 16-bit registers, but they are connected to an 8-bit data bus. When the CPU accesses these registers, to ensure that both bytes are written or read simultaneously, the access is performed using an 8-bit temporary register (TEMP).

These registers are written and read as follows:

- **Register Write**
  When the CPU writes to the upper byte, the byte of write data is placed in TEMP. Next, when the CPU writes to the lower byte, this byte of data is combined with the byte in TEMP and all 16 bits are written in the register simultaneously.

- **Register Read**
  When the CPU reads the upper byte, the upper byte of data is sent to the CPU and the lower byte is placed in TEMP. When the CPU reads the lower byte, it receives the value in TEMP.
  (As an exception, when the CPU reads OCRA or OCRB, it reads both the upper and lower bytes directly, without using TEMP.)

Programs that access these registers should normally use word access. Equivalently, they may access first the upper byte, then the lower byte by two consecutive byte accesses. Data will not be transferred correctly if the bytes are accessed in reverse order, if only one byte is accessed, or if the upper and lower bytes are accessed separately and another register is accessed in between, altering the value in TEMP.

**Coding Examples**

To write the contents of general register R0 to OCRA: `MOV.W  R0, @OCRA`

To transfer the ICR contents to general register R0: `MOV.W  @ICR, R0`

Figure 7-3 shows the data flow when the FRC is accessed. The other registers are accessed in the same way.

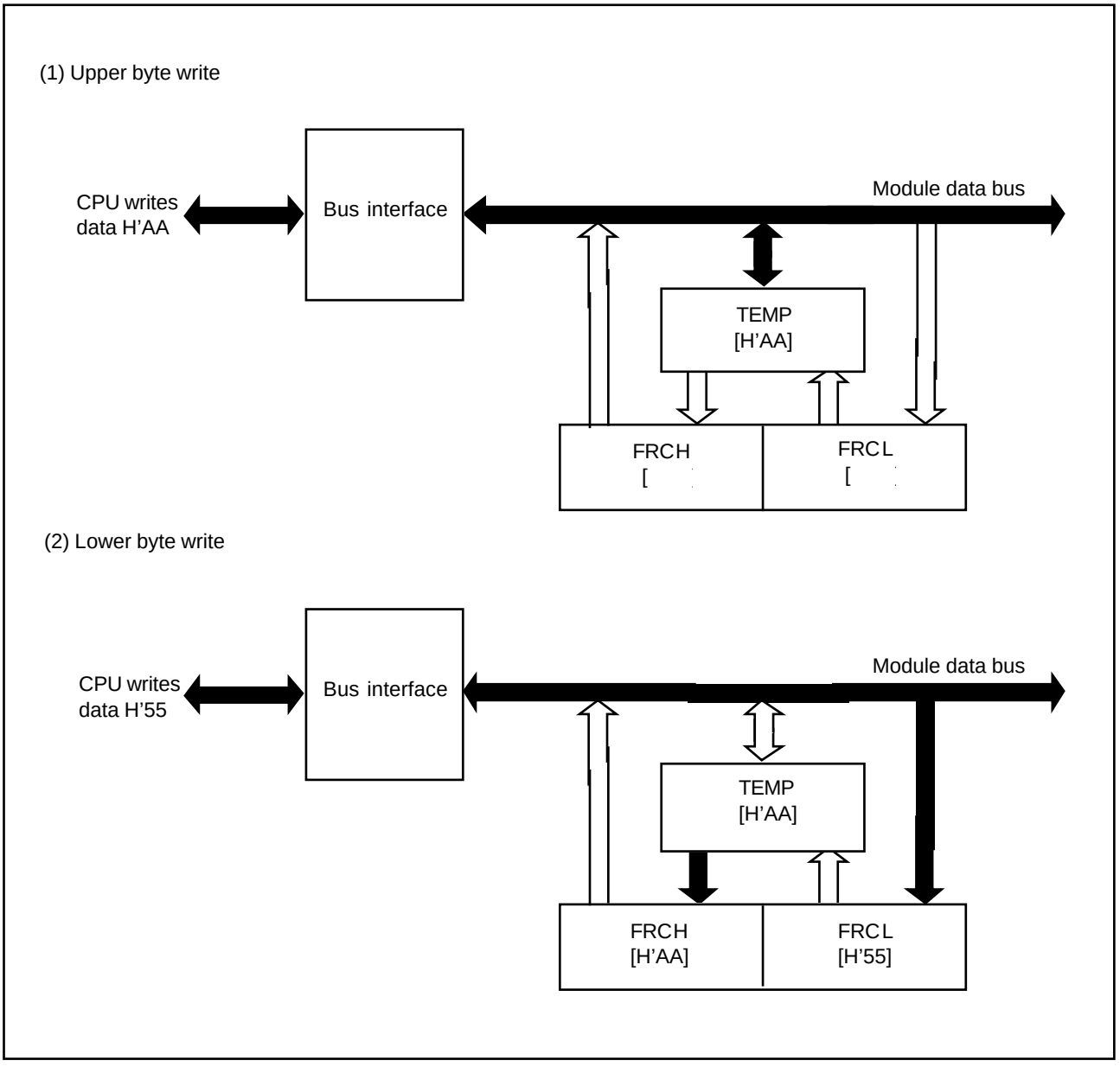



**Figure 7-3 (a). Write Access to FRC (When CPU Writes H'AA55)**

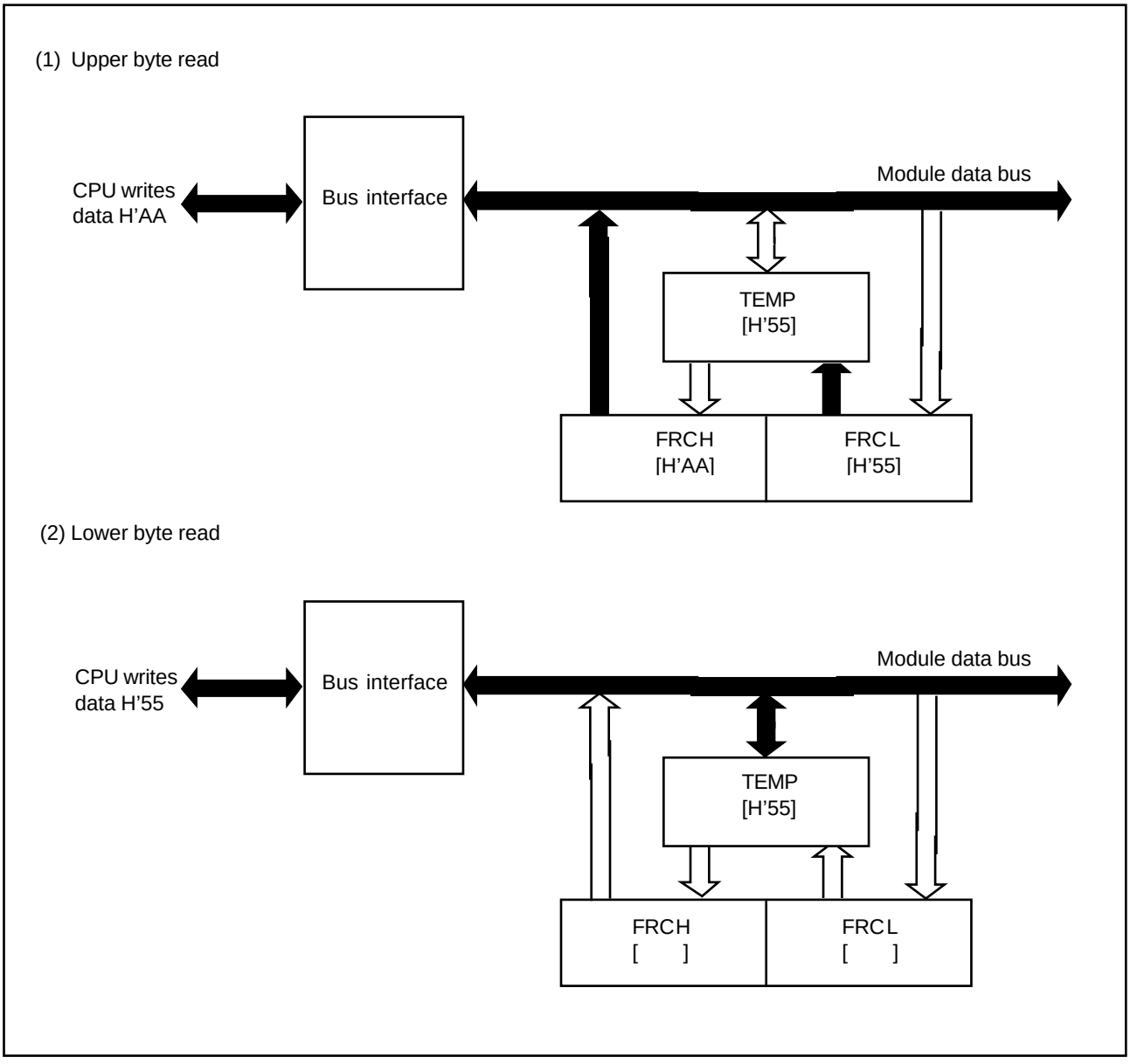



**Figure 7-3 (b). Read Access to FRC (When FRC Contains H'AA55)**

## 7.4 Operation

### 7.4.1 FRC Incrementation Timing

The FRC increments on a pulse generated once for each cycle of the selected (internal or external) clock source.

**(1) Internal Clock Sources:** Can be selected by the CKS1 and CKS0 bits in the TCR. Internal clock sources are created by dividing the system clock (Ø). Three internal clock sources are available: Ø/2, Ø/8, and Ø/32. Figure 7-4 shows the increment timing.

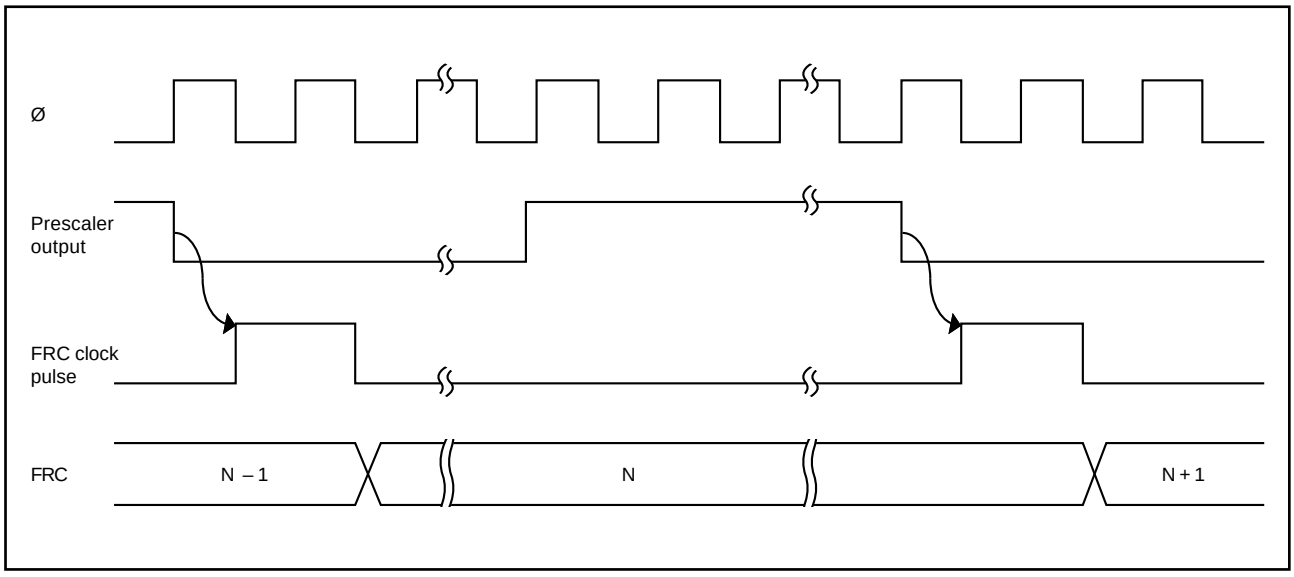



**Figure 7-4. Increment Timing for Internal Clock Source**

**(2) External Clock Input:** Can be selected by the CKS1 and CKS0 bits in the TCR. The FRC increments on the rising edge of the FTCI clock signal. The pulse width of the external clock signal must be at least 1.5 system clock (Ø) cycles. The counter will not increment correctly if the pulse width is shorter than this.

Figure 7-5 shows the increment timing. Figure 7-6 shows the minimum external clock pulse width.

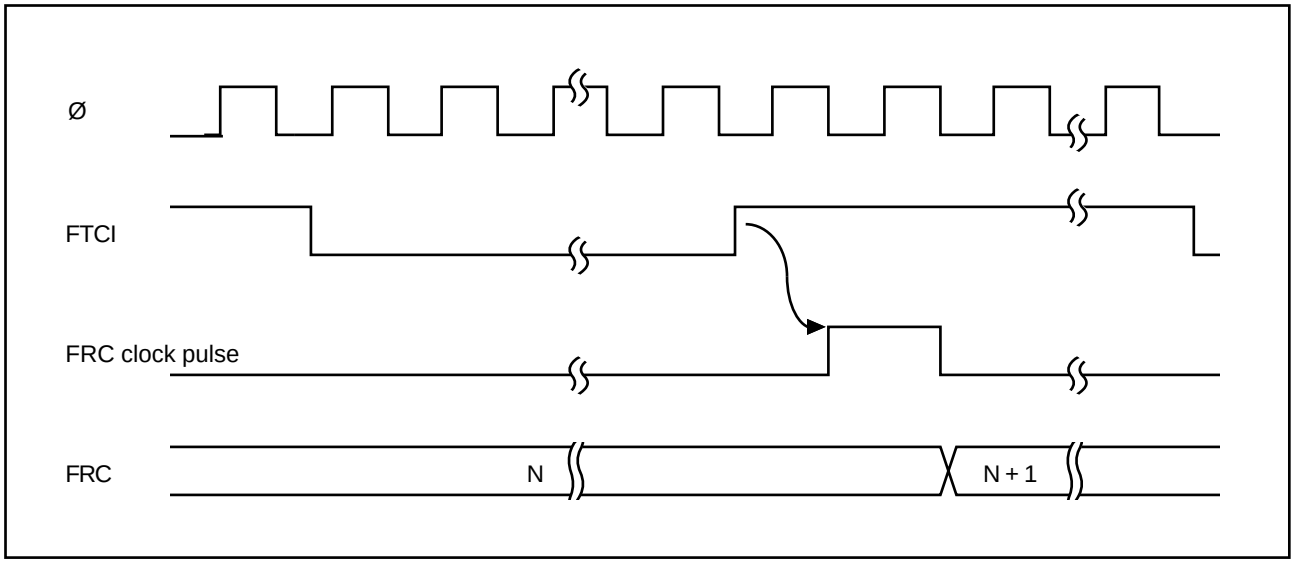



**Figure 7-5. Increment Timing for External Clock Source**

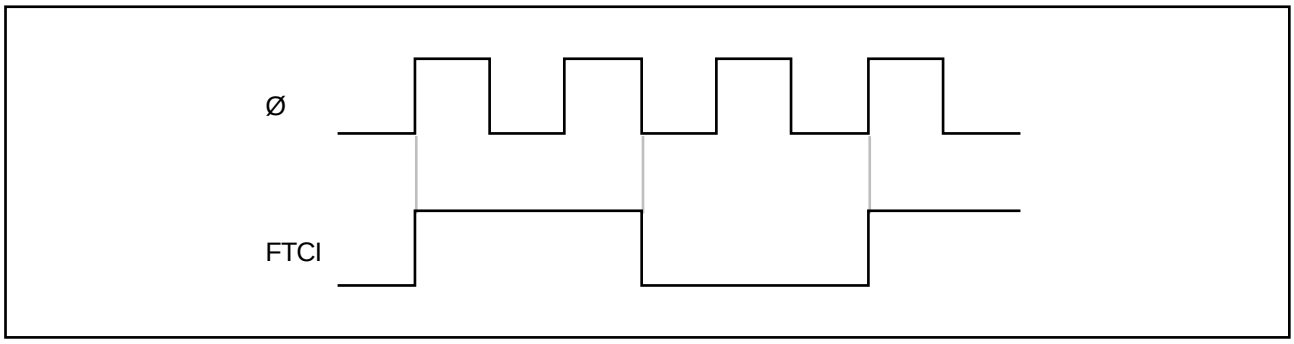



**Figure 7-6. Minimum External Clock Pulse Width**

### 7.4.2 Output Compare Timing

When a compare-match occurs, the logic level selected by the output level bit (OLVLA or OLVLB) in the TCSR is output at the output compare pin (FTOA or FTOB). Figure 7-7 shows the timing of this operation for compare-match A.

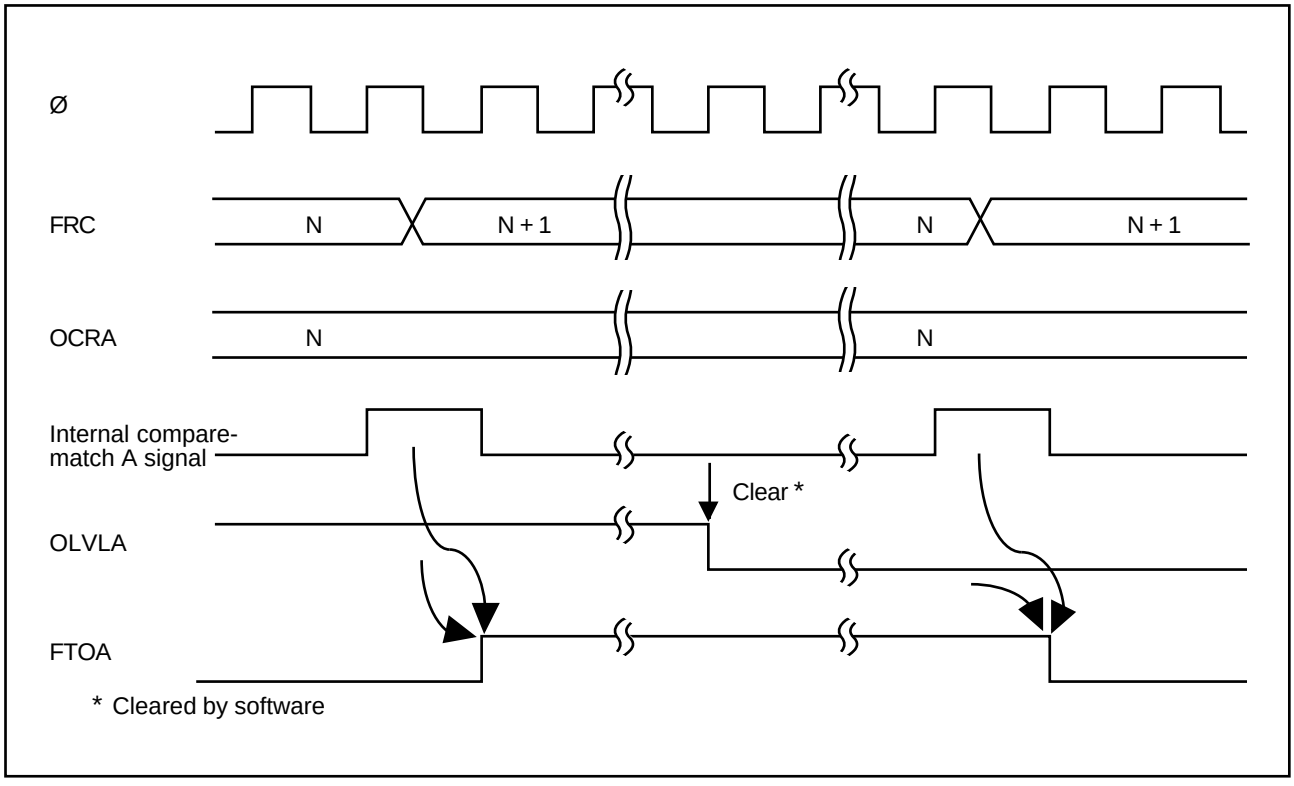



**Figure 7-7. Timing of Output Compare A**

### 7.4.3 FRC Clear Timing

If the CCLRA bit in the TCSR is set to 1, the FRC is cleared when compare-match A occurs. Figure 7-8 shows the timing of this operation.

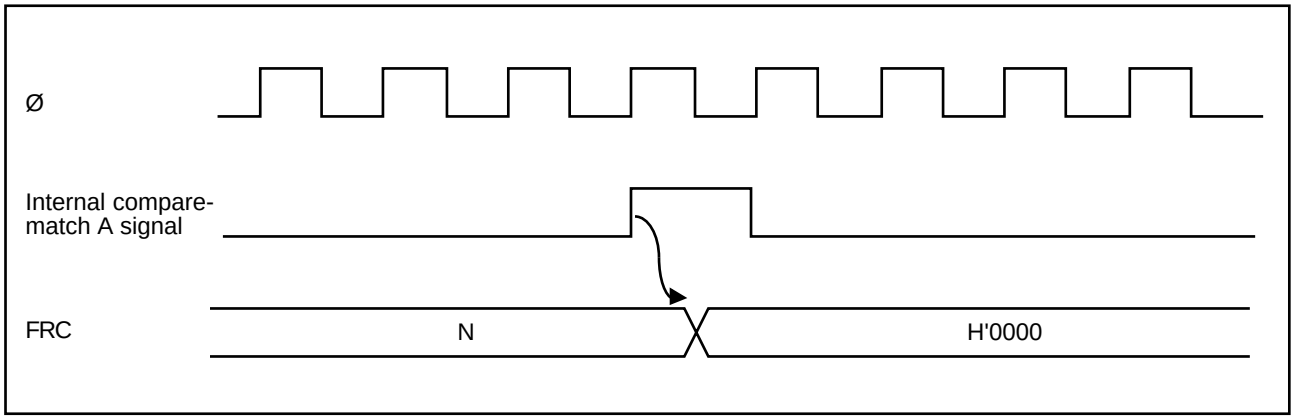



**Figure 7-8. Clearing of FRC by Compare-Match A**

### 7.4.4 Input Capture Timing

**(1) Input Capture Timing without Noise Canceler:** An internal input capture signal is generated from the rising or falling edge of the FTI input, as selected by the IEDG bit in the TCSR. Figure 7-9 shows the usual input capture timing when the rising edge is selected (IEDG = 1).

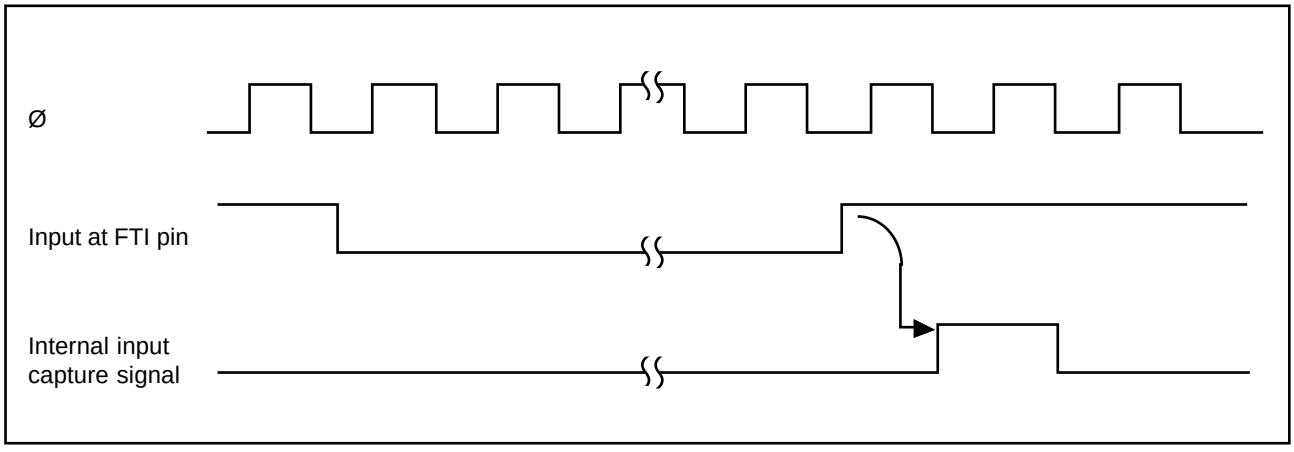



**Figure 7-9. Input Capture Timing (Usual Case)**

If the upper byte of the ICR is being read when the internal input capture signal should be generated, the internal input capture signal is delayed by one state. Figure 7-10 shows the timing for this case.

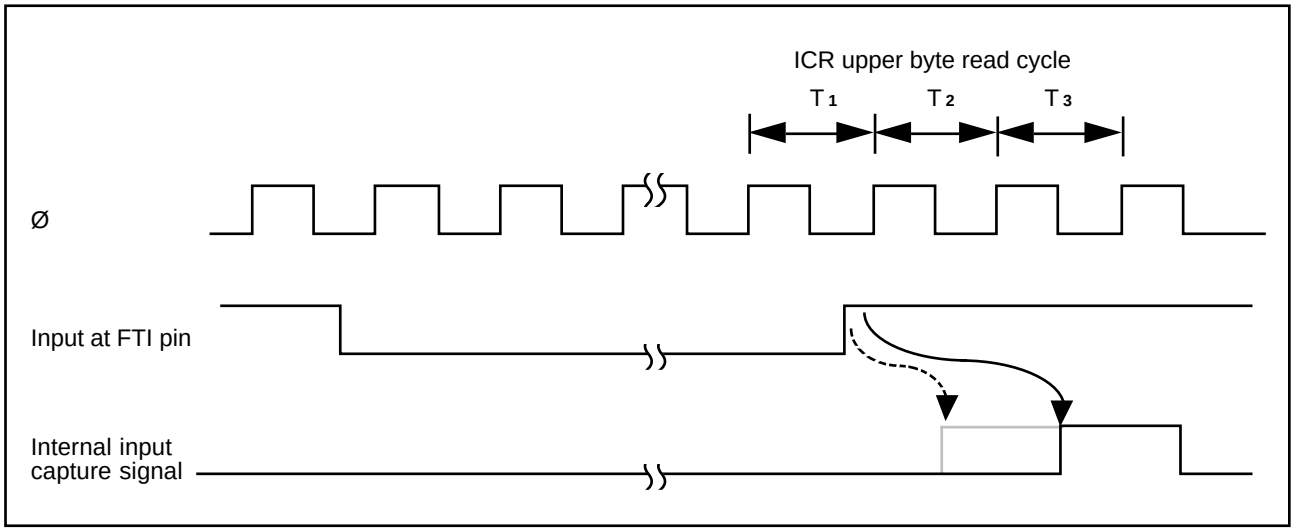



**Figure 7-10. Input Capture Timing (1-State Delay Due to ICR Read)**

**(2) Input Capture Timing with Noise Canceler:** The noise canceler samples the FTI input, and does generate an internal input capture signal until three to four sampling clock cycles after the rise or fall of FTI. Figure 7-9 shows the timing.

If the upper byte of the ICR is being read when the internal input capture signal should be generated, the internal input capture signal is additionally delayed by one system clock cycle (Ø).

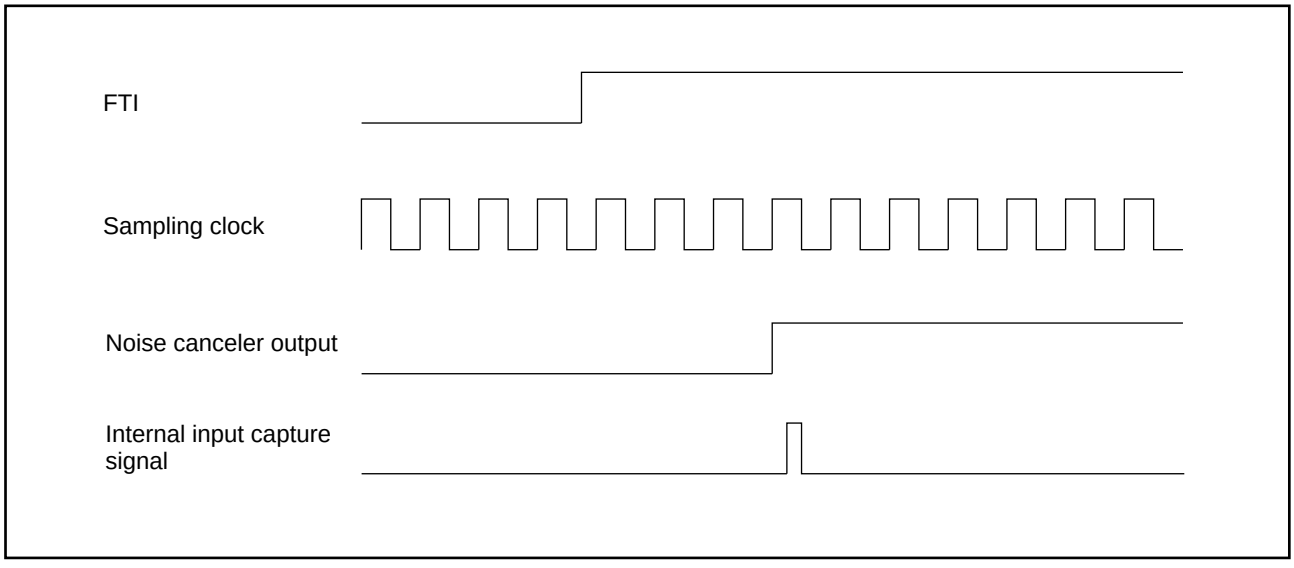



**Figure 7-11. Input Capture Timing with Noise Cancellation**

### 7.4.5 Timing of Input Capture Flag (ICF) Setting

The input capture flag ICF is set to 1 by the internal input capture signal. The FRC contents are transferred to the ICR at the same time. Figure 7-12 shows the timing of this operation.

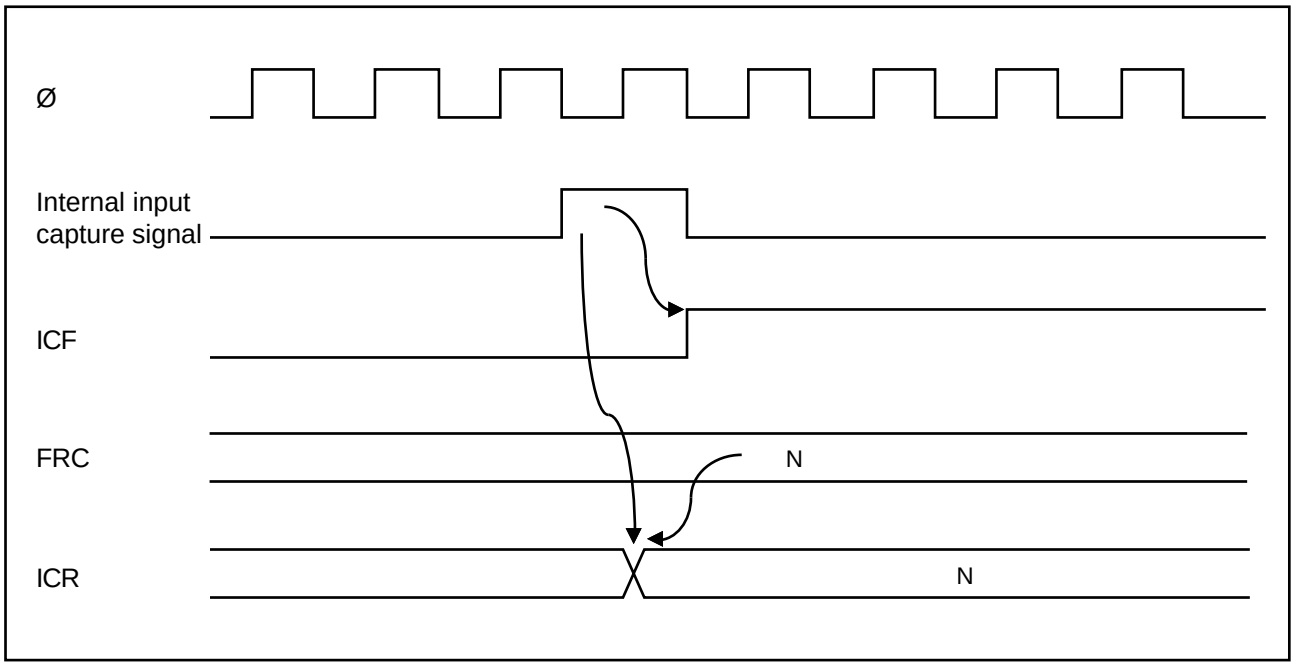



**Figure 7-12. Setting of Input Capture Flag**

### 7.4.6 Setting of FRC Overflow Flag (OVF)

The FRC overflow flag (OVF) is set to 1 when the FRC changes from H'FFFF to H'0000. Figure 7-13 shows the timing of this operation.

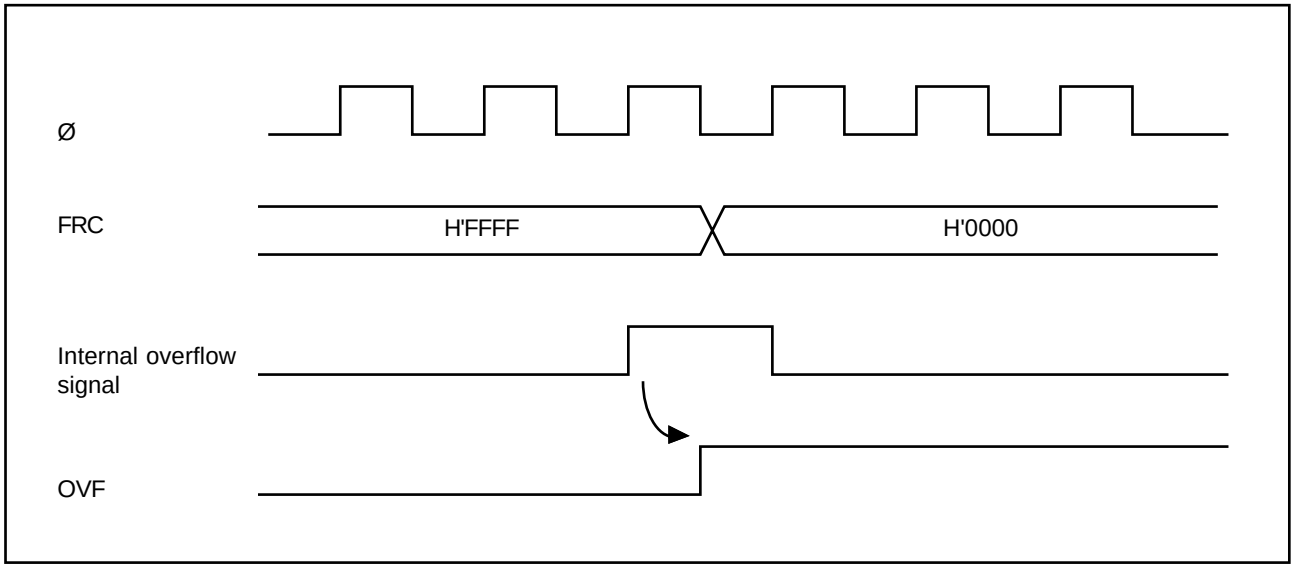



**Figure 7-13. Setting of Overflow Flag (OVF)**

## 7.5 Interrupts

The free-running timer channel can request four types of interrupts: input capture (ICI), output compare A and B (OCIA and OCIB), and overflow (FOVI). Each interrupt is requested when the corresponding flag bit is set, provided the corresponding enable bit is also set. Independent signals are sent to the interrupt controller for each type of interrupt. Table 7-3 lists information about these interrupts.

**Table 7-3. Free-Running Timer Interrupts**

| Interrupt | Description | Priority |
|---|---|---|
| ICI | Requested when ICF and ICIE are set | High |
| OCIA | Requested when OCFA and OCIAE are set | ↕ |
| OCIB | Requested when OCFB and OCIBE are set | |
| FOVI | Requested when OVF and OVIE are set | Low |

## 7.6 Noise Canceler

The noise canceler acts as a digital low-pass filter, rejecting high-frequency pulses received at the input capture (FTI) pin. Figure 7-14 shows a block diagram of the noise canceler.

The noise canceler consists of four latches connected in series, and a circuit that detects when all four latches contain the same value. The FTI input is sampled on the rising edge of the sampling clock selected by the NCS1 and NCS0 bits. When all four latches contain the same value, this value is regarded as valid and output from the noise canceler. If all four latches are not the same, the noise canceler holds its previous output. Immediately after a reset, the noise canceler output is 0.

To assure capture, the pulse input at the FTI pin must be at least four sampling clock cycles wide. The noise canceler control register (FNCR) provides a selection of three sampling clock rates and the option of disabling the noise canceler. Table 7-4 indicates the cycle times of the sampling clock for various settings.

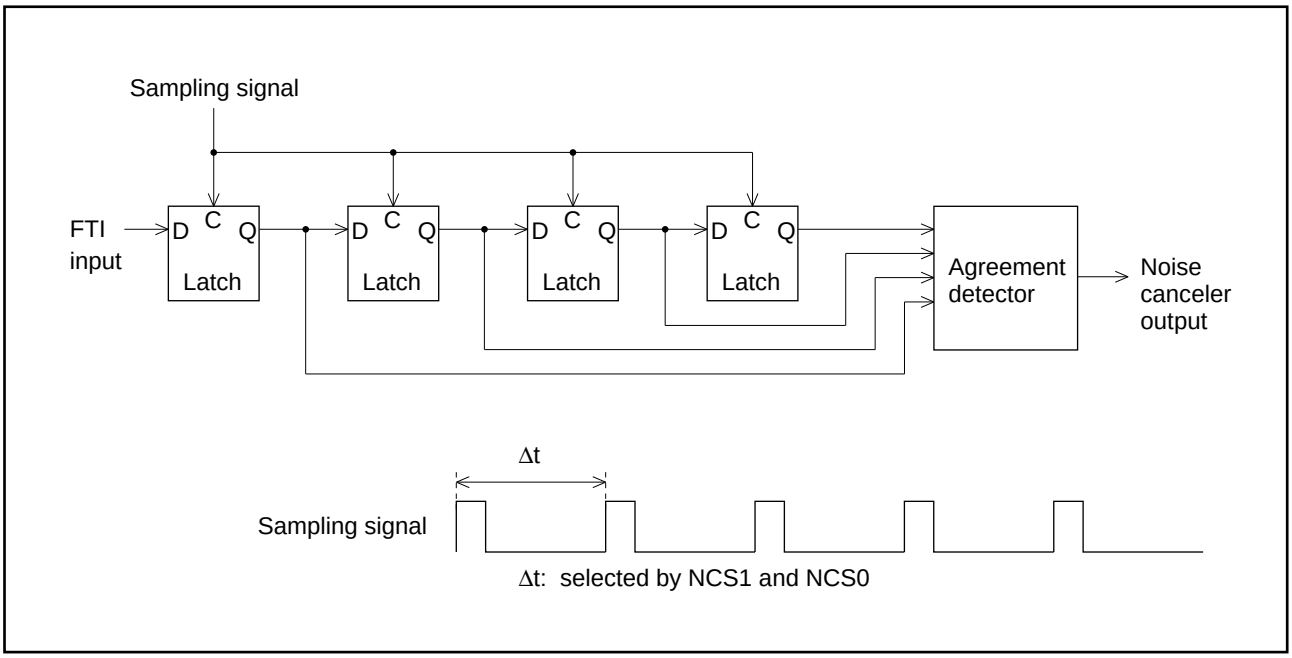



**Figure 7-14. Noise Canceler Block Diagram**

**Table 7-4. Sampling Clock Cycle for Various System Clock Frequencies**

| | | Sampling | System clock (Ø) frequency (MHz) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **NCS1** | **NCS0** | **clock** | **10** | **8** | **6** | **4** | **2** | **1** | **0.5** |
| 0 | 0 | — | — | — | — | — | — | — | — |
| 0 | 1 | Ø/32 | 3.2 | 4.0 | 5.3 | 8.0 | 16.0 | 32.0 | 64.0 |
| 1 | 0 | Ø/64 | 6.4 | 8.0 | 10.7 | 16.0 | 32.0 | 64.0 | 128.0 |
| 1 | 1 | Ø/128 | 12.8 | 16.0 | 21.3 | 32.0 | 64.0 | 128.0 | 256.0 |

(Unit: µs)

Figure 7-15 shows an example of noise cancellation. In this example, an input capture pulse narrower than four sampling clock cycles is rejected as noise.

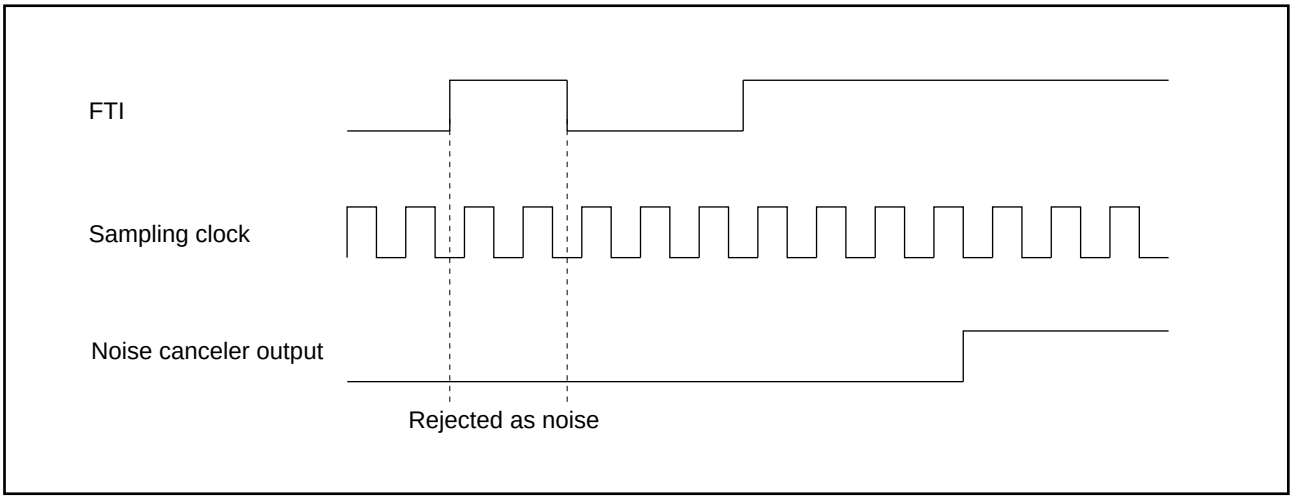



**Figure 7-15. Noise Cancellation (Example)**

## 7.7 Sample Application

In the example below, the free-running timer channel is used to generate two square-wave outputs with a 50% duty factor and arbitrary phase relationship. The programming is as follows:

(1) The CCLRA bit in the TCSR is set to 1.

(2) Each time a compare-match interrupt occurs, software inverts the corresponding output level bit in the TCSR (OLVLA or OLVLB).

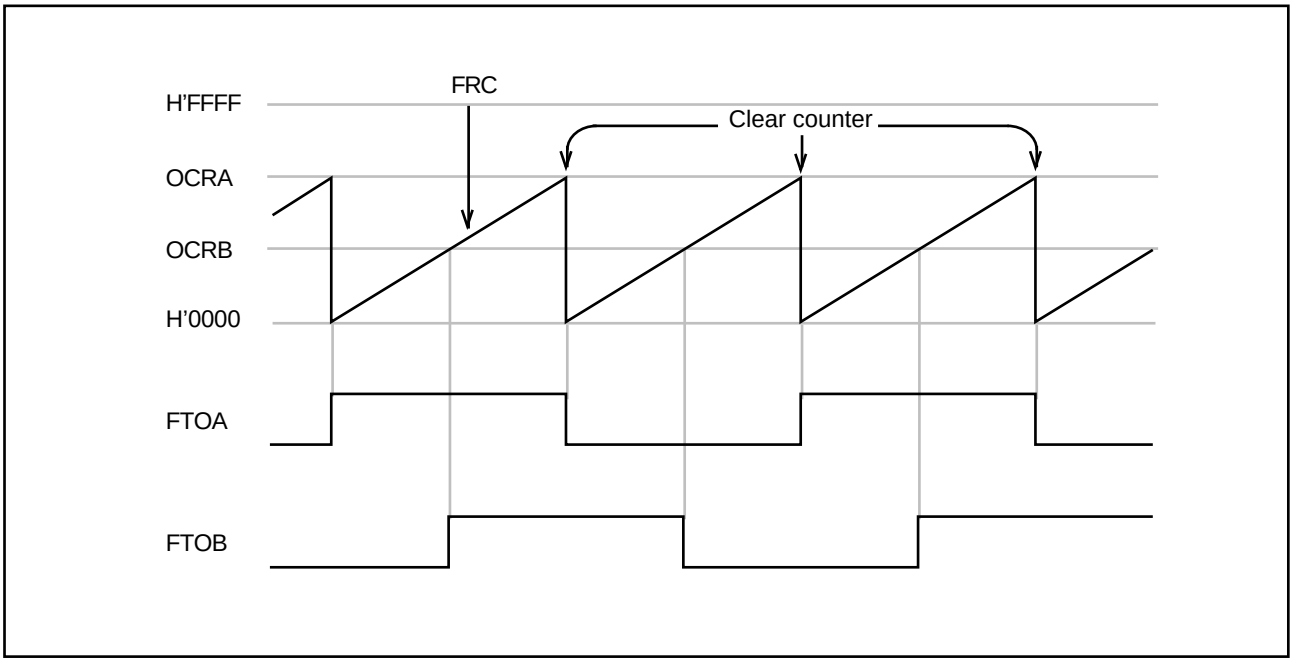



**Figure 7-16. Square-Wave Output (Example)**

## 7.8 Application Notes

Application programmers should note that the following types of contention can occur in the free-running timer.

**(1) Contention between FRC Write and Clear:** If an internal counter clear signal is generated during the $T_3$ state of a write cycle to the lower byte of the free-running counter, the clear signal takes priority and the write is not performed.

Figure 7-17 shows this type of contention.

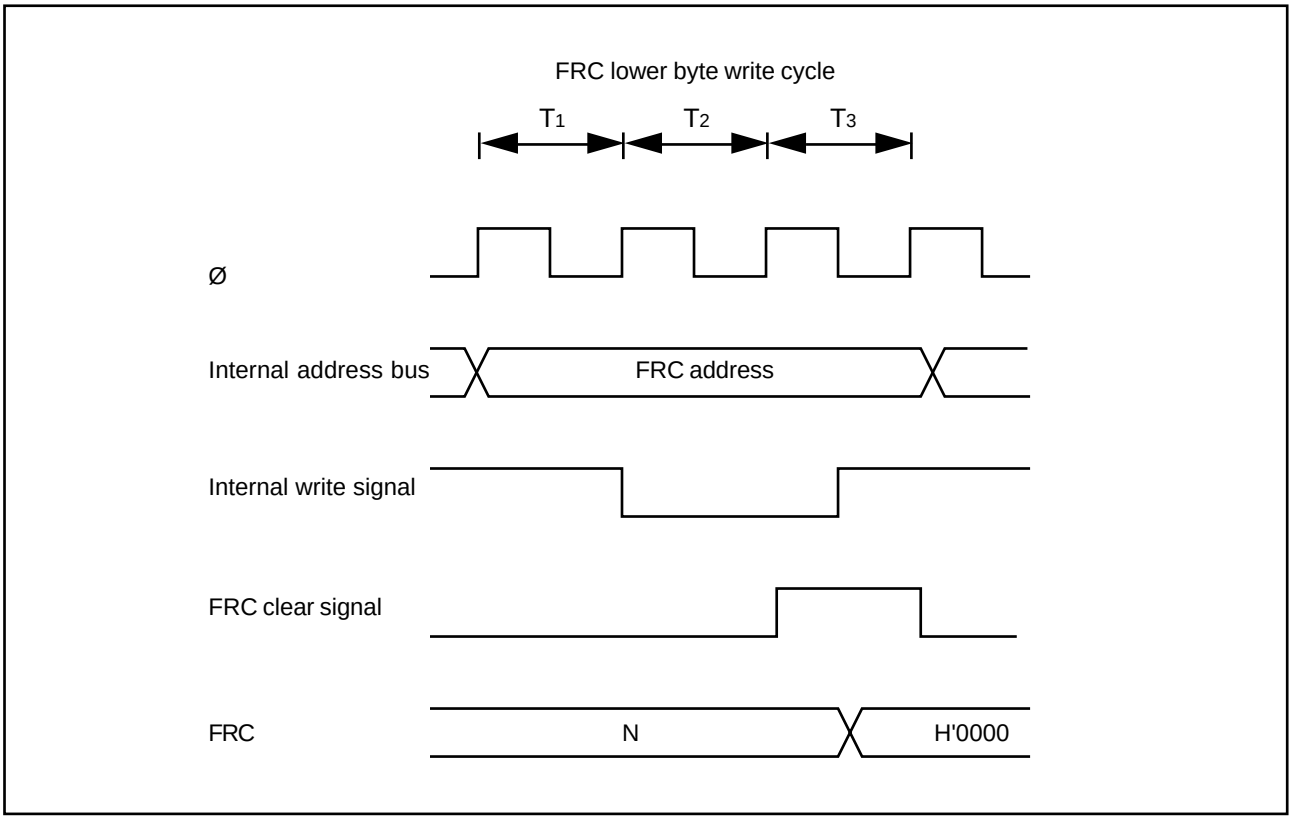



**Figure 7-17. FRC Write-Clear Contention**

**(2) Contention between FRC Write and Increment:** If an FRC increment pulse is generated during the $T_3$ state of a write cycle to the lower byte of the free-running counter, the write takes priority and the FRC is not incremented.

Figure 7-18 shows this type of contention.

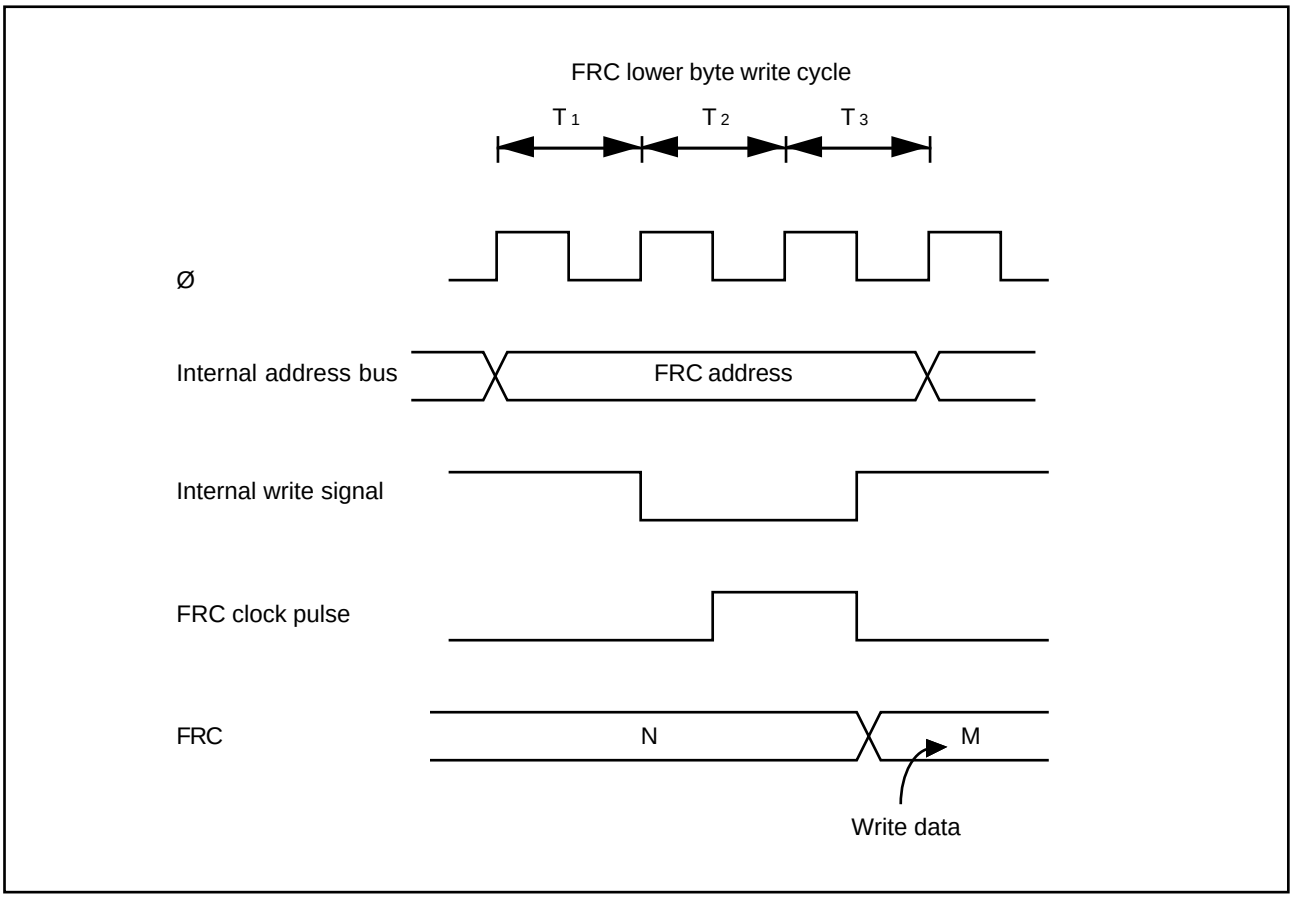



**Figure 7-18. FRC Write-Increment Contention**

**(3) Contention between OCR Write and Compare-Match:** If a compare-match occurs during the T3 state of a write cycle to the lower byte of OCRA or OCRB, the write takes precedence and the compare-match signal is inhibited.

Figure 7-19 shows this type of contention.

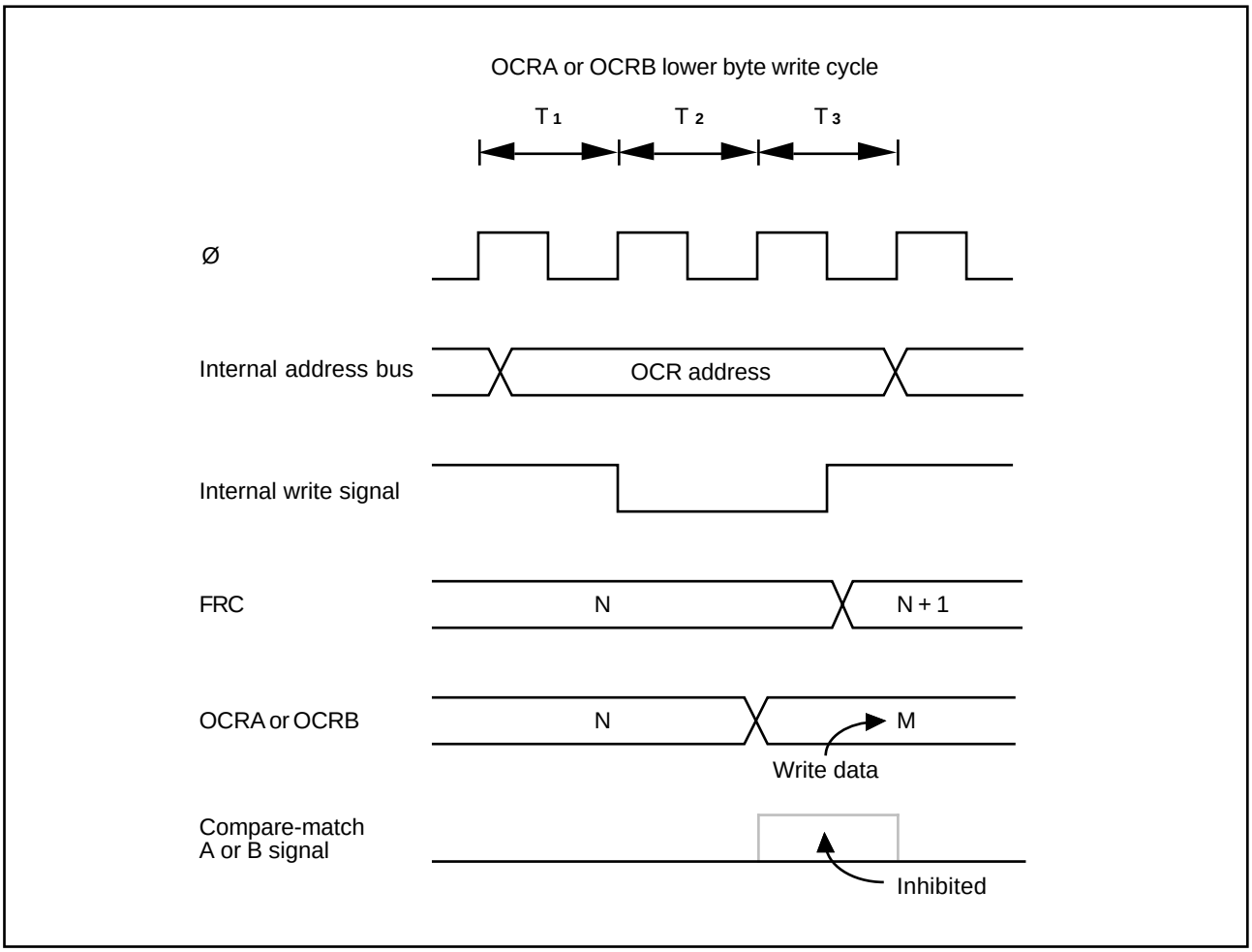


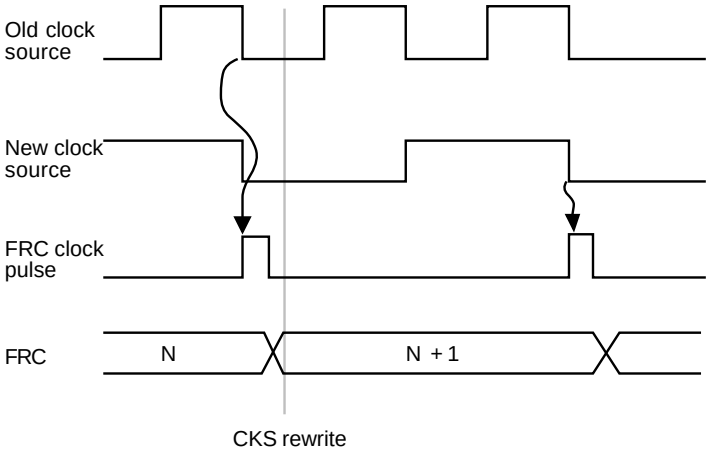
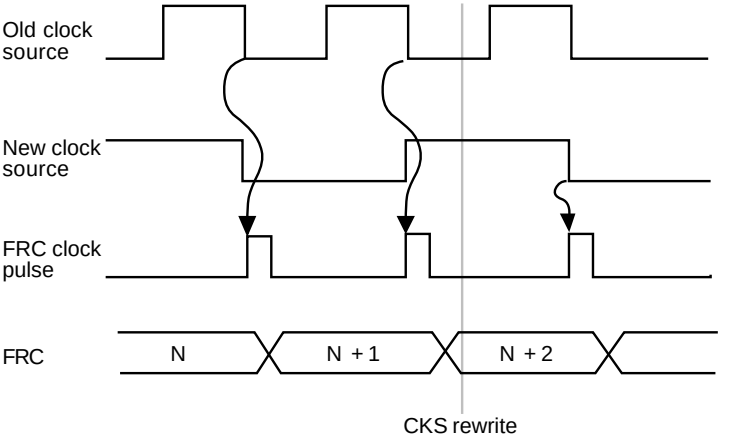
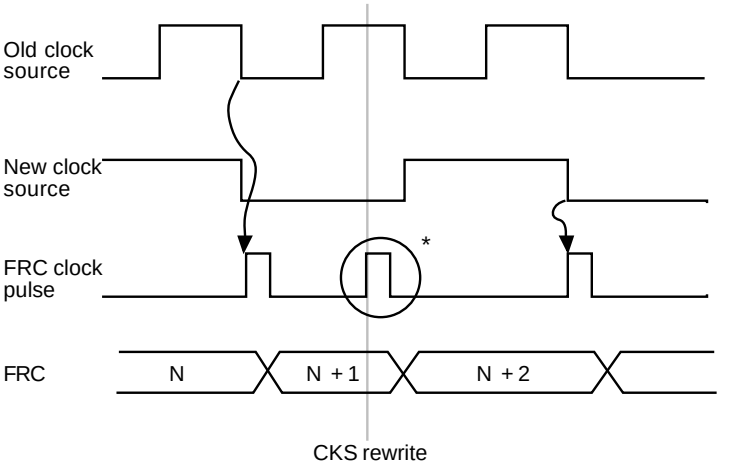

**Figure 7-19. Contention between OCR Write and Compare-Match**

**(4) Incrementation Caused by Changing of Internal Clock Source:** When an internal clock source is changed, the changeover may cause the FRC to increment. This depends on the time at which the clock select bits (CKS1 and CKS0) are rewritten, as shown in table 7-5.

The pulse that increments the FRC is generated at the falling edge of the internal clock source. If clock sources are changed when the old source is high and the new source is low, as in case No. 3 in table 7-5, the changeover generates a falling edge that triggers the FRC increment clock pulse.

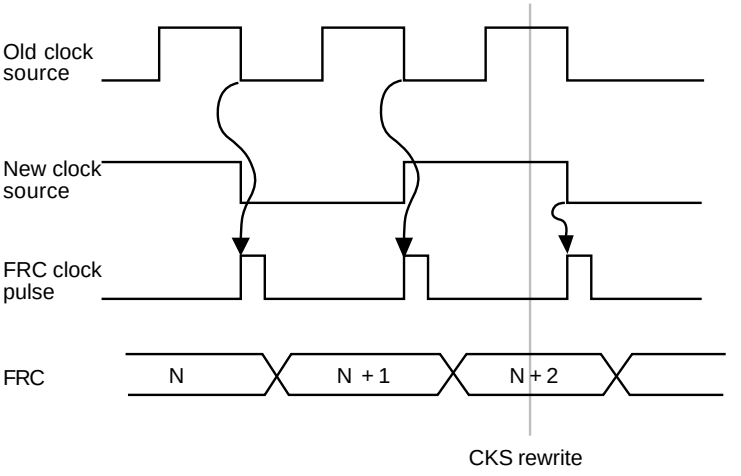
Switching between an internal and external clock source can also cause the FRC to increment.

**Table 7-5. Effect of Changing Internal Clock Sources**

| No. | Description | Timing chart |
|---|---|---|
| 1 | Low → Low:<br>CKS1 and CKS0 are rewritten while both clock sources are low. |  |
| 2 | Low → High:<br>CKS1 and CKS0 are rewritten while old clock source is low and new clock source is high. |  |
| 3 | High → Low:<br>CKS1 and CKS0 are rewritten while old clock source is high and new clock source is low. |  |

* The switching of clock sources is regarded as a falling edge that increments the FRC.

**Table 7-5. Effect of Changing Internal Clock Sources (cont.)**

| No. | Description | Timing chart |
|---|---|---|
| 4 | High → High:<br>CKS1 and CKS0 are rewritten while both clock sources are high. |  |

# Section 8. 8-Bit Timers

## 8.1 Overview

The H8/325 series chips include an 8-bit timer module with two channels. Each channel has an 8-bit counter (TCNT) and two time constant registers (TCORA and TCORB) that are constantly compared with the TCNT value to detect compare-match events. One application of the 8-bit timer module is to generate a rectangular-wave output with an arbitrary duty factor.

### 8.1.1 Features

The features of the 8-bit timer module are listed below.

- Selection of four clock sources
  The counters can be driven by an internal clock signal (Ø/8, Ø/64, or Ø/1024) or an external clock input (enabling use as an external event counter).

- Selection of three ways to clear the counters
  The counters can be cleared on compare-match A or B, or by an external reset signal.

- Timer output controlled by two time constants
  The timer output signal in each channel is controlled by two independent time constants, enabling the timer to generate output waveforms with an arbitrary duty factor.

- Three independent interrupts
  Compare-match A and B and overflow interrupts can be requested independently.

### 8.1.2 Block Diagram

Figure 8-1 shows a block diagram of one channel in the 8-bit timer module. The other channel is identical.

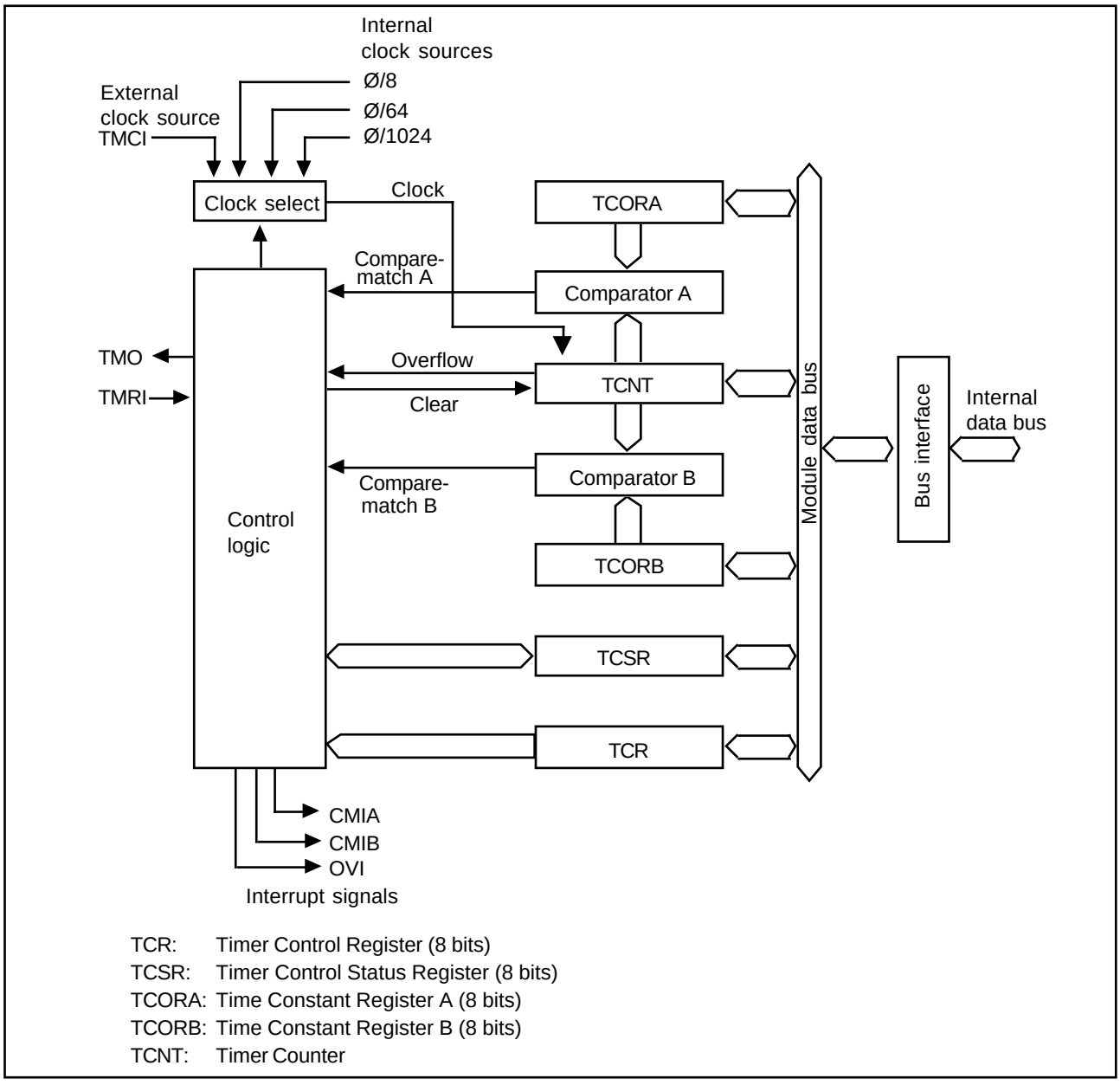



**Figure 8-1. Block Diagram of 8-Bit Timer**

### 8.1.3 Input and Output Pins

Table 8-1 lists the input and output pins of the 8-bit timer.

**Table 8-1. Input and Output Pins of 8-Bit Timer**

| | Abbreviation | | | |
|---|---|---|---|---|
| **Name** | **TMR0** | **TMR1** | **I/O** | **Function** |
| Timer output | $TMO_0$ | $TMO_1$ | Output | Output controlled by compare-match |
| Timer clock input | $TMCI_0$ | $TMCI_1$ | Input | External clock source for the counter |
| Timer reset input | $TMRI_0$ | $TMRI_1$ | Input | External reset signal for the counter |

### 8.1.4 Register Configuration

Table 8-2 lists the registers of the 8-bit timer module. Each channel has an independent set of registers.

**Table 8-2. 8-Bit Timer Registers**

| | | | | Address | |
|---|---|---|---|---|---|
| **Name** | **Abbreviation** | **R/W** | **Initial value** | **TMR0** | **TMR1** |
| Timer control register | TCR | R/W | H'00 | H'FFC8 | H'FFD0 |
| Timer control/status register | TCSR | R/(W)* | H'10 | H'FFC9 | H'FFD1 |
| Timer constant register A | TCORA | R/W | H'FF | H'FFCA | H'FFD2 |
| Timer constant register B | TCORB | R/W | H'FF | H'FFCB | H'FFD3 |
| Timer counter | TCNT | R/W | H'00 | H'FFCC | H'FFD4 |

* Software can write a 0 to clear bits 7 to 5, but cannot write a 1 in these bits.

## 8.2 Register Descriptions

### 8.2.1 Timer Counter (TCNT) – H'FFC8 (TMR0), H'FFD0 (TMR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Each timer counter (TCNT) is an 8-bit up-counter that increments on a pulse generated from one of four clock sources. The clock source is selected by clock select bits 2 to 0 (CKS2 to CKS0) of the timer control register (TCR). The CPU can always read or write the timer counter.

The timer counter can be cleared by an external reset input or by an internal compare-match signal generated at a compare-match event. Counter clear bits 1 and 0 (CCLR1 and CCLR0) of the timer control register select the method of clearing.

When a timer counter overflows from H'FF to H'00, the overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

The timer counters are initialized to H'00 at a reset and in the standby modes.

### 8.2.2 Time Constant Registers A and B (TCORA and TCORB) – H'FFCA and H'FFCB (TMR0), H'FFD2 and H'FFD3 (TMR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

TCORA and TCORB are 8-bit readable/writable registers. The timer count is continually compared with the constants written in these registers. When a match is detected, the corresponding compare-match flag (CMFA or CMFB) is set in the timer control/status register (TCSR).

The timer output signal (TMO0 or TMO1) is controlled by these compare-match signals as specified by output select bits 3 to 0 (OS3 to OS0) in the timer control/status register (TCSR).

TCORA and TCORB are initialized to H'FF at a reset and in the standby modes.

Compare-match is not detected during the $T_3$ state of a write cycle to TCORA or TCORB. See item (3) in section 8.6, Application Notes.

### 8.2.3 Timer Control Register (TCR) – H'FFC8 (TMR0), H'FFD0 (TMR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

TCR is an 8-bit readable/writable register that selects the clock source and the time at which the timer counter is cleared, and enables interrupts.

TCR is initialized to H'00 at a reset and in the standby modes.

**Bit 7 – Compare-match Interrupt Enable B (CMIEB):** This bit selects whether to request compare-match interrupt B (CMIB) when compare-match flag B (CMFB) in the timer control/status register (TCSR) is set to 1.

**Bit 7**

| CMIEB | Description | |
|---|---|---|
| 0 | Compare-match interrupt request B (CMIB) is disabled. | (Initial value) |
| 1 | Compare-match interrupt request B (CMIB) is enabled. | |

**Bit 6 – Compare-match Interrupt Enable A (CMIEA):** This bit selects whether to request compare-match interrupt A (CMIA) when compare-match flag A (CMFA) in the timer control/status register (TCSR) is set to 1.

**Bit 6**

| CMIEA | Description | |
|---|---|---|
| 0 | Compare-match interrupt request A (CMIA) is disabled. | (Initial value) |
| 1 | Compare-match interrupt request A (CMIA) is enabled. | |

**Bit 5 – Timer Overflow Interrupt Enable (OVIE):** This bit selects whether to request a timer overflow interrupt (OVI) when the overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

**Bit 5**

| OVIE | Description | |
|---|---|---|
| 0 | The timer overflow interrupt request (OVI) is disabled. | (Initial value) |
| 1 | The timer overflow interrupt request (OVI) is enabled. | |

**Bits 4 and 3 – Counter Clear 1 and 0 (CCLR1 and CCLR0):** These bits select how the timer counter is cleared: by compare-match A or B or by an external reset input.

| Bit 4 | Bit 3 | | |
|---|---|---|---|
| CCLR1 | CCLR0 | Description | |
| 0 | 0 | Not cleared. | (Initial value) |
| 0 | 1 | Cleared on compare-match A. | |
| 1 | 0 | Cleared on compare-match B. | |
| 1 | 1 | Cleared on rising edge of external reset input signal. | |

**Bits 2, 1, and 0 – Clock Select (CKS2, CKS1, and CKS0):** These bits select the internal or external clock source for the timer counter. For the external clock source they select whether to increment the count on the rising or falling edge of the clock input, or on both edges. For the internal clock sources the count is incremented on the falling edge of the clock input.

| Bit 2 CKS2 | Bit 1 CKS1 | Bit 0 CKS0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | No clock source (timer stopped) | (Initial value) |
| 0 | 0 | 1 | Ø/8 Internal clock source, counted on the falling edge | |
| 0 | 1 | 0 | Ø/64 Internal clock source, counted on the falling edge | |
| 0 | 1 | 1 | Ø/1024 Internal clock source, counted on the falling edge | |
| 1 | 0 | 0 | No clock source (timer stopped) | |
| 1 | 0 | 1 | External clock source, counted on the rising edge | |
| 1 | 1 | 0 | External clock source, counted on the falling edge | |
| 1 | 1 | 1 | External clock source, counted on both the rising and falling edges | |

### 8.2.4 Timer Control/Status Register (TCSR) – H'FFC9 (TMR0), H'FFD1 (TMR1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMFB | CMFA | OVF | — | OS3 | OS2 | OS1 | OS0 |
| Initial value | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | R/(W)* | R/(W)* | R/(W)* | — | R/W | R/W | R/W | R/W |

* Software can write a 0 in bits 7 to 5 to clear the flags, but cannot write a 1 in these bits.

TCSR is an 8-bit readable and partially writable register that indicates compare-match and overflow status and selects the effect of compare-match events on the timer output signal.

TCSR is initialized to H'10 at a reset and in the standby modes.

**Bit 7 – Compare-Match Flag B (CMFB):** This status flag is set to 1 when the timer count matches the time constant set in TCORB. CMFB must be cleared by software. It is set by hardware, however, and cannot be set by software.

| Bit 7 CMFB | Description | |
|---|---|---|
| 0 | To clear CMFB, the CPU must read CMFB after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when TCNT = TCORB. | |

**Bit 6 – Compare-Match Flag A (CMFA):** This status flag is set to 1 when the timer count matches the time constant set in TCORA. CMFA must be cleared by software. It is set by hardware, however, and cannot be set by software.

**Bit 6**

| CMFA | Description | |
|---|---|---|
| 0 | To clear CMFA, the CPU must read CMFA after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when TCNT = TCORA. | |

**Bit 5 – Timer Overflow Flag (OVF):** This status flag is set to 1 when the timer count overflows (changes from H'FF to H'00). OVF must be cleared by software. It is set by hardware, however, and cannot be set by software.

**Bit 5**

| OVF | Description | |
|---|---|---|
| 0 | To clear OVF, the CPU must read OVF after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when TCNT changes from H'FF to H'00. | |

**Bit 4 – Reserved:** This bit is always read as 1. It cannot be written.

**Bits 3 to 0 – Output Select 3 to 0 (OS3 to OS0):** These bits specify the effect of compare-match events on the timer output signal. Bits OS3 and OS2 control the effect of compare-match B on the output level. Bits OS1 and OS0 control the effect of compare-match A on the output level.

If compare-match A and B occur simultaneously, any conflict is resolved by giving highest priority to toggle, second-highest priority to 1 output, and third-highest priority to 0 output, as explained in item (4) in section 8.6, Application Notes.

After a reset, the timer output is 0 until the first compare-match event.

When all four output select bits are cleared to 0 the timer output signal is disabled.

| Bit 3 OS3 | Bit 2 OS2 | Description | |
|---|---|---|---|
| 0 | 0 | No change when compare-match B occurs. | (Initial value) |
| 0 | 1 | Output changes to 0 when compare-match B occurs. | |
| 1 | 0 | Output changes to 1 when compare-match B occurs. | |
| 1 | 1 | Output inverts (toggles) when compare-match B occurs. | |

| Bit 1 OS1 | Bit 0 OS0 | Description | |
|---|---|---|---|
| 0 | 0 | No change when compare-match A occurs. | (Initial value) |
| 0 | 1 | Output changes to 0 when compare-match A occurs. | |
| 1 | 0 | Output changes to 1 when compare-match A occurs. | |
| 1 | 1 | Output inverts (toggles) when compare-match A occurs. | |

## 8.3 Operation

### 8.3.1 TCNT Incrementation Timing

The timer counter increments on a pulse generated once for each period of the clock source selected by bits CKS2 to CKS0 of the TCR.

**Internal Clock:** Internal clock sources are created from the system clock by a prescaler. The counter increments on an internal TCNT clock pulse generated from the falling edge of the prescaler output, as shown in figure 8-2. Bits CKS2 to CKS0 of the TCR can select one of the three internal clocks (Ø/8, Ø/64, or Ø/1024).

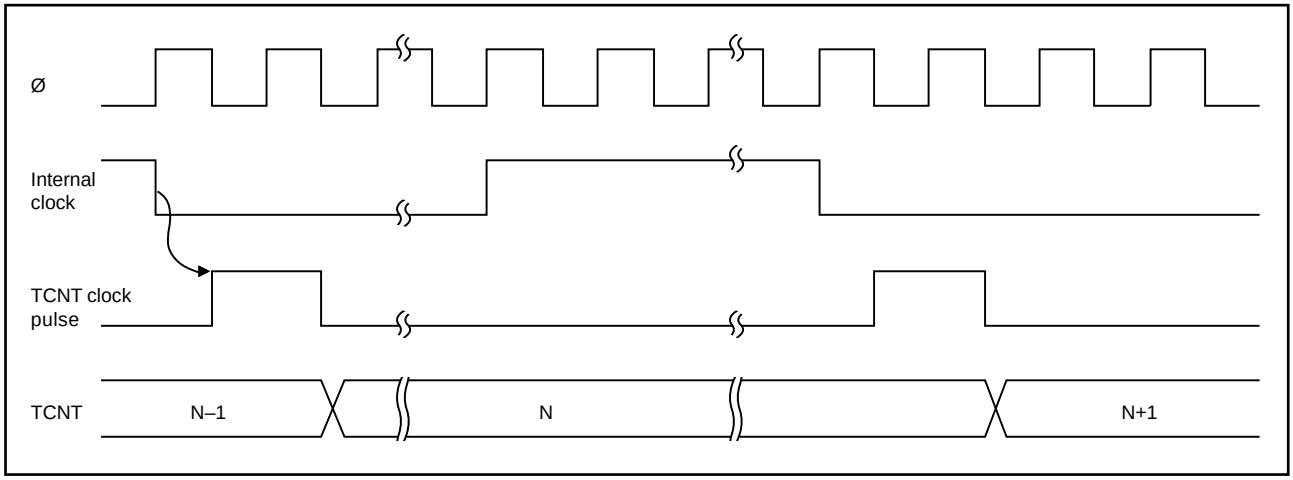



**Figure 8-2. Count Timing for Internal Clock Input**

**External Clock:** If external clock input (TMCI) is selected, the timer counter can increment on the rising edge, the falling edge, or both edges of the external clock signal. Figure 8-3 shows incrementation on both edges of the external clock signal.

The external clock pulse width must be at least 1.5 system clock periods for incrementation on a single edge, and at least 2.5 system clock periods for incrementation on both edges. See figure 8.4. The counter will not increment correctly if the pulse width is shorter than these values.

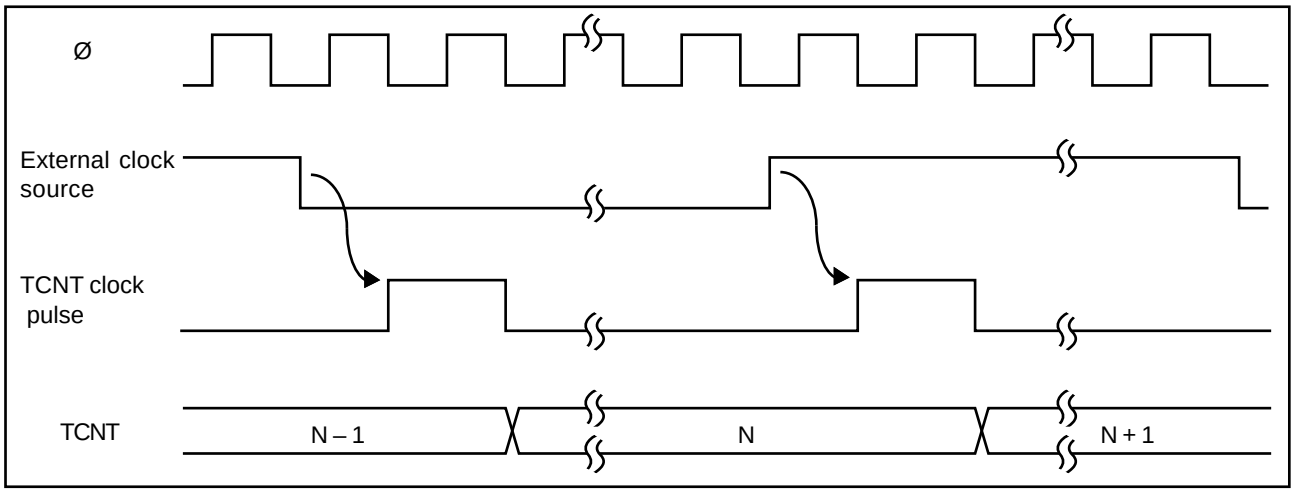



**Figure 8-3. Count Timing for External Clock Input**

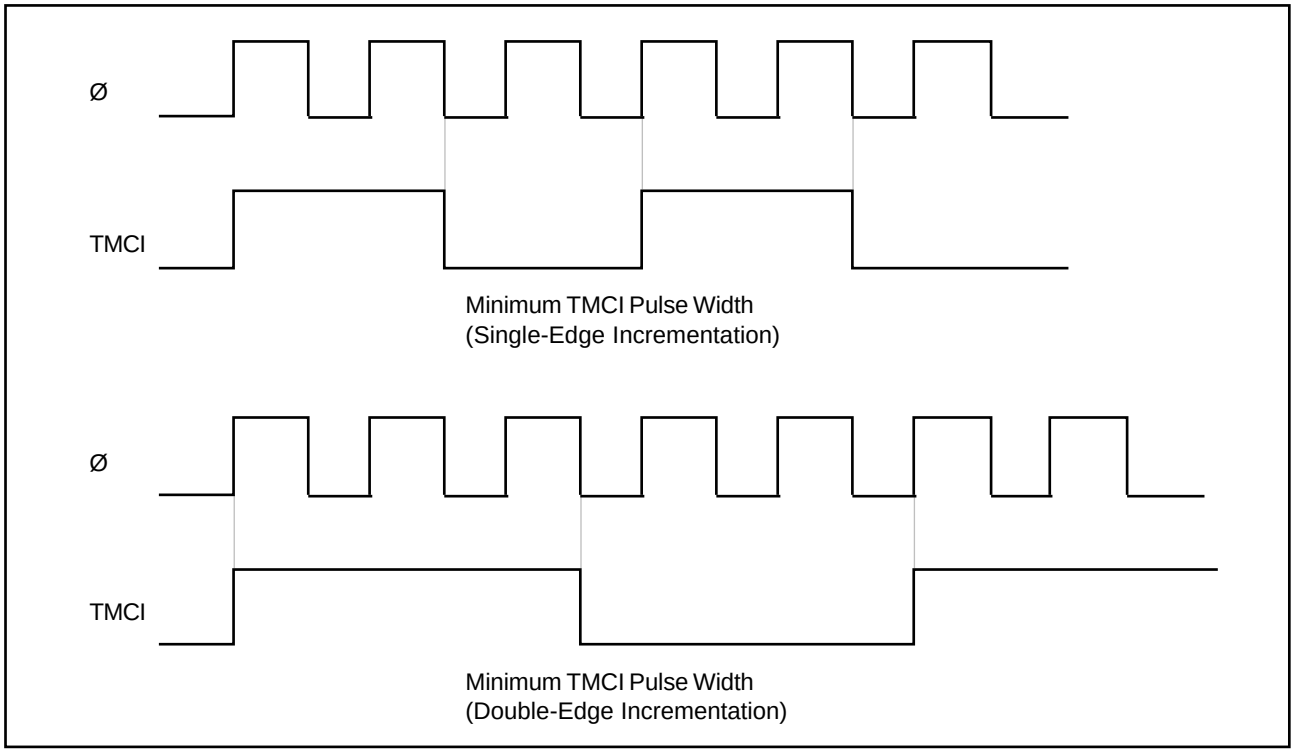



**Figure 8-4. Minimum External Clock Pulse Widths (Example)**

### 8.3.2 Compare Match Timing

**(1) Setting of Compare-Match Flags A and B (CMFA and CMFB):** The compare-match flags are set to 1 by an internal compare-match signal generated when the timer count matches the time constant in TCNT or TCOR. The compare-match signal is generated at the last state in which the match is true, just before the timer counter increments to a new value.

Accordingly, when the timer count matches one of the time constants, the compare-match signal is not generated until the next period of the clock source. Figure 8-5 shows the timing of the setting of the compare-match flags.

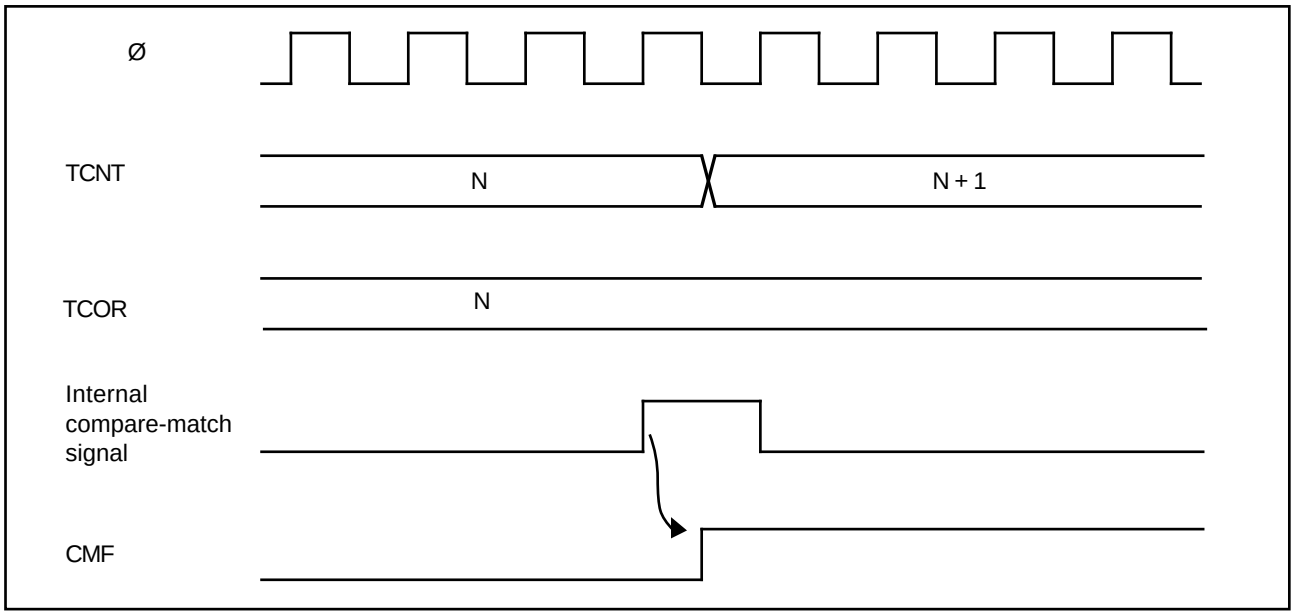



**Figure 8-5. Setting of Compare-Match Flags**

**(2) Timing of Compare-Match Flag (CMFA or CMFB) Clearing:** The compare-match flag CMFA or CMFB is cleared when the CPU writes a 0 in this bit.

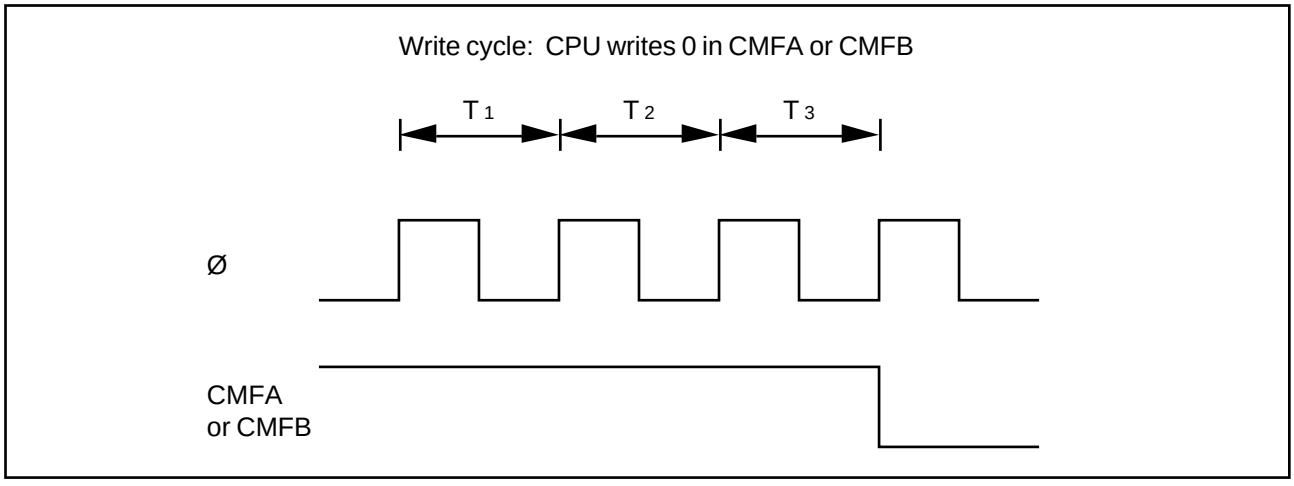



**Figure 8-6. Clearing of Compare-Match Flags**

**(3) Output Timing:** When a compare-match event occurs, the timer output (TMO0 or TMO1) changes as specified by the output select bits (OS3 to OS0) in the TCSR. Depending on these bits, the output can remain the same, change to 0, change to 1, or toggle. If compare-match A and B occur simultaneously, the higher priority compare-match determines the output level. See item (4) in section 8.6, Application Notes for details.

Figure 8-7 shows the timing when the output is set to toggle on compare-match A.

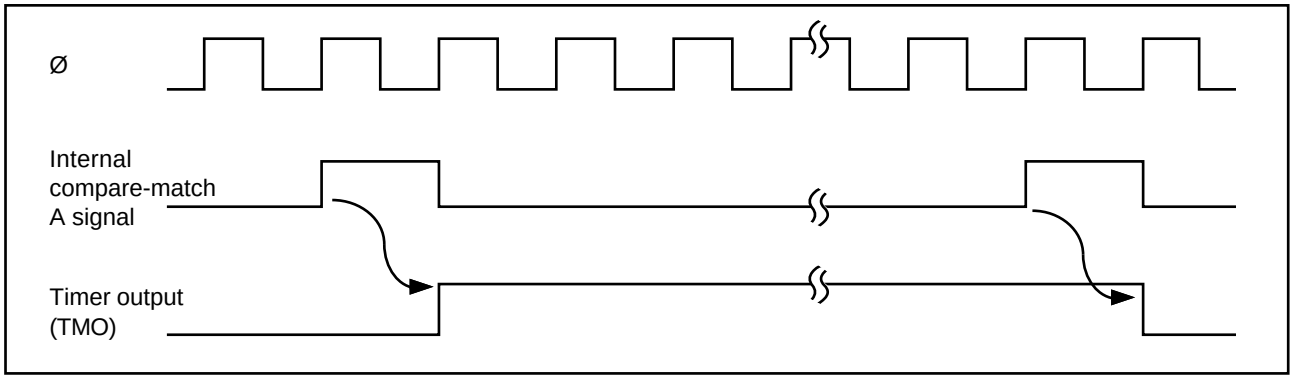



**Figure 8-7. Timing of Timer Output**

**(4) Timing of Compare-Match Clear:** Depending on the CCLR1 and CCLR0 bits in the TCR, the timer counter can be cleared when compare-match A or B occurs. Figure 8-8 shows the timing of this operation.

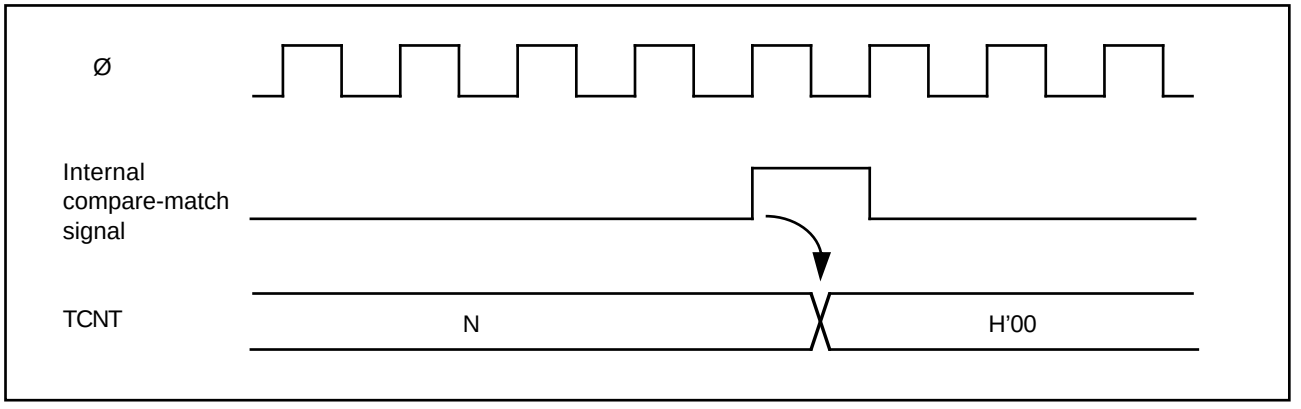



**Figure 8-8. Timing of Compare-Match Clear**

### 8.3.3 External Reset of TCNT

When the CCLR1 and CCLR0 bits in the TCR are both set to 1, the timer counter is cleared on the rising edge of an external reset input. Figure 8-9 shows the timing of this operation. The timer reset pulse width must be at least 1.5 system clock periods.

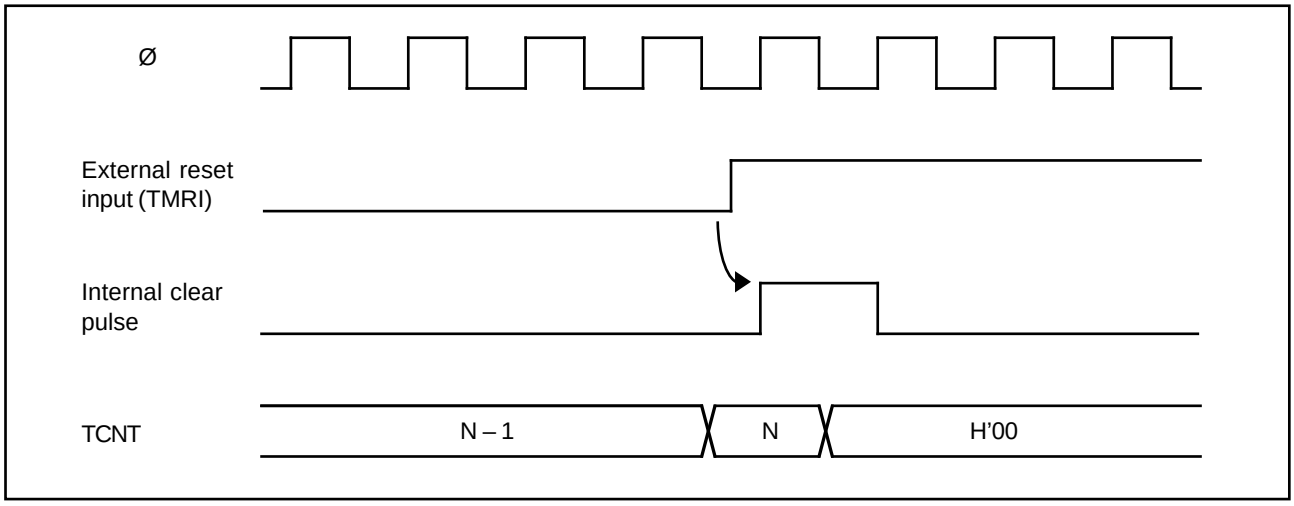



**Figure 8-9. Timing of External Reset**

### 8.3.4 Setting of TCSR Overflow Flag

**(1) Setting of TCSR Overflow Flag (OVF):** The overflow flag (OVF) is set to 1 when the timer count overflows (changes from H'FF to H'00). Figure 8-10 shows the timing of this operation.

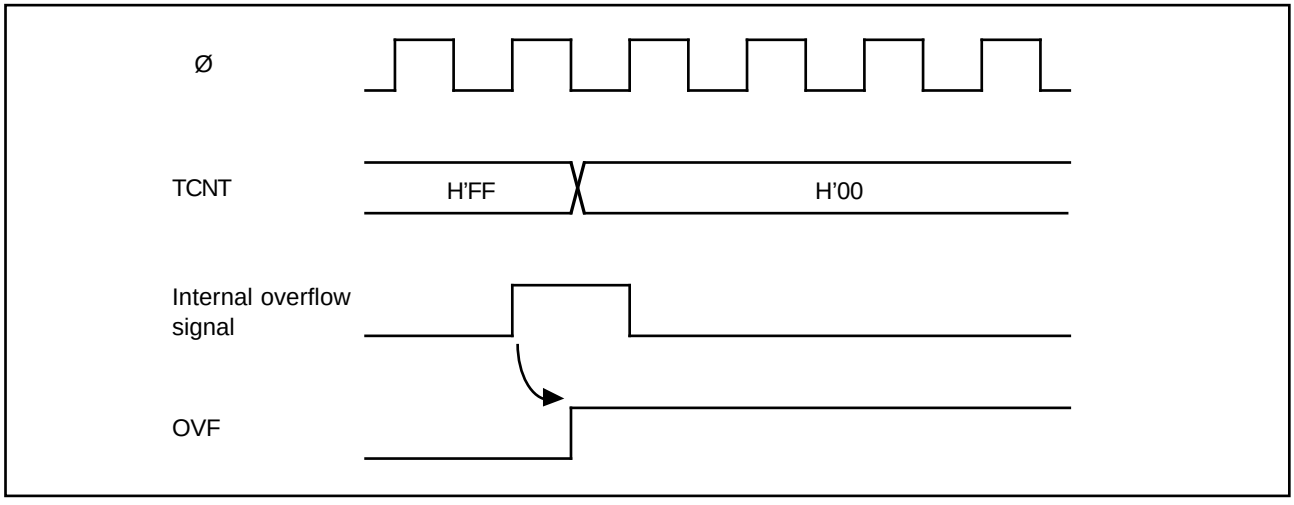



**Figure 8-10. Setting of Overflow Flag (OVF)**

**(2) Timing of TCSR Overflow Flag (OVF) Clearing:** The overflow flag (OVF) is cleared when the CPU writes a 0 in this bit.

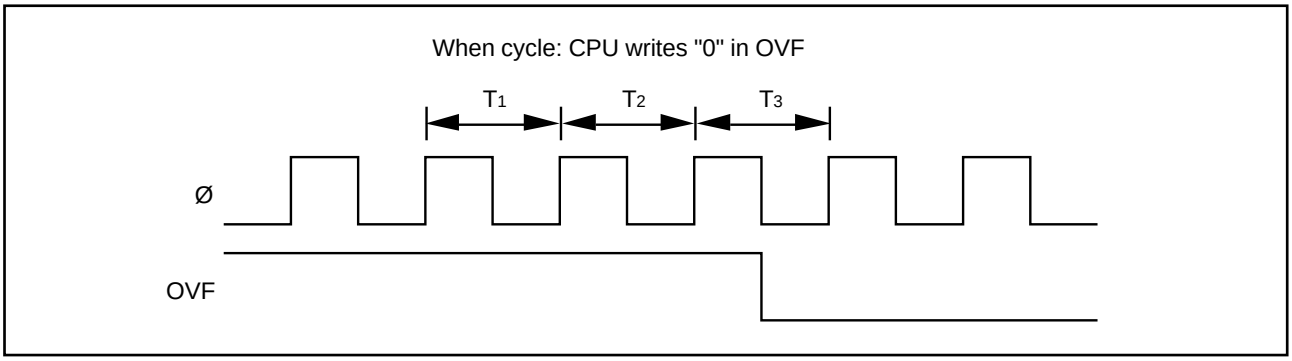



**Figure 8-11. Clearing of Overflow Flag**

## 8.4 Interrupts

Each channel in the 8-bit timer can generate three types of interrupts: compare-match A and B (CMIA and CMIB), and overflow (OVI). Each interrupt is requested when the corresponding enable bits are set in the TCR and TCSR. Independent signals are sent to the interrupt controller for each interrupt. Table 8-3 lists information about these interrupts.

**Table 8-3. 8-Bit Timer Interrupts**

| Interrupt | Description | Priority |
|---|---|---|
| CMIA | Requested when CMFA and CMIEA are set | High |
| CMIB | Requested when CMFB and CMIEB are set | ↕ |
| OVI | Requested when OVF and OVIE are set | Low |

## 8.5 Sample Application

In the example below, the 8-bit timer is used to generate a pulse output with a selected duty factor. The control bits are set as follows:

(1) In the TCR, CCLR1 is cleared to 0 and CCLR0 is set to 1 so that the timer counter is cleared when its value matches the constant in TCORA.

(2) In the TCSR, bits OS3 to OS0 are set to 0110, causing the output to change to 1 on compare-match A and to 0 on compare-match B.

With these settings, the 8-bit timer provides output of pulses at a rate determined by TCORA with a pulse width determined by TCORB. No software intervention is required.

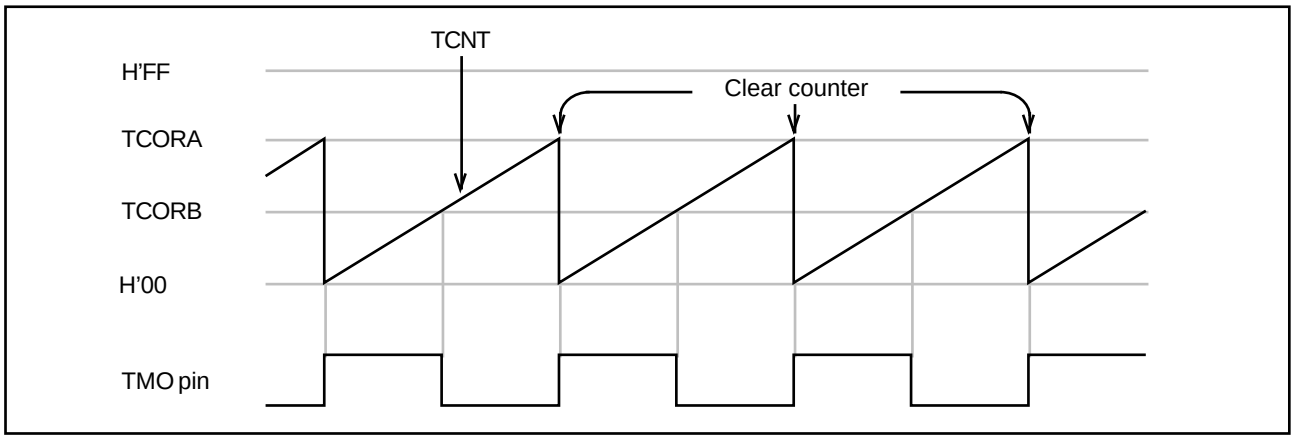



**Figure 8-12. Example of Pulse Output**

## 8.6 Application Notes

Application programmers should note that the following types of contention can occur in the 8-bit timer.

**(1) Contention between TCNT Write and Clear:** If an internal counter clear signal is generated during the $T_3$ state of a write cycle to the timer counter, the clear signal takes priority and the write is not performed.

Figure 8-13 shows this type of contention.

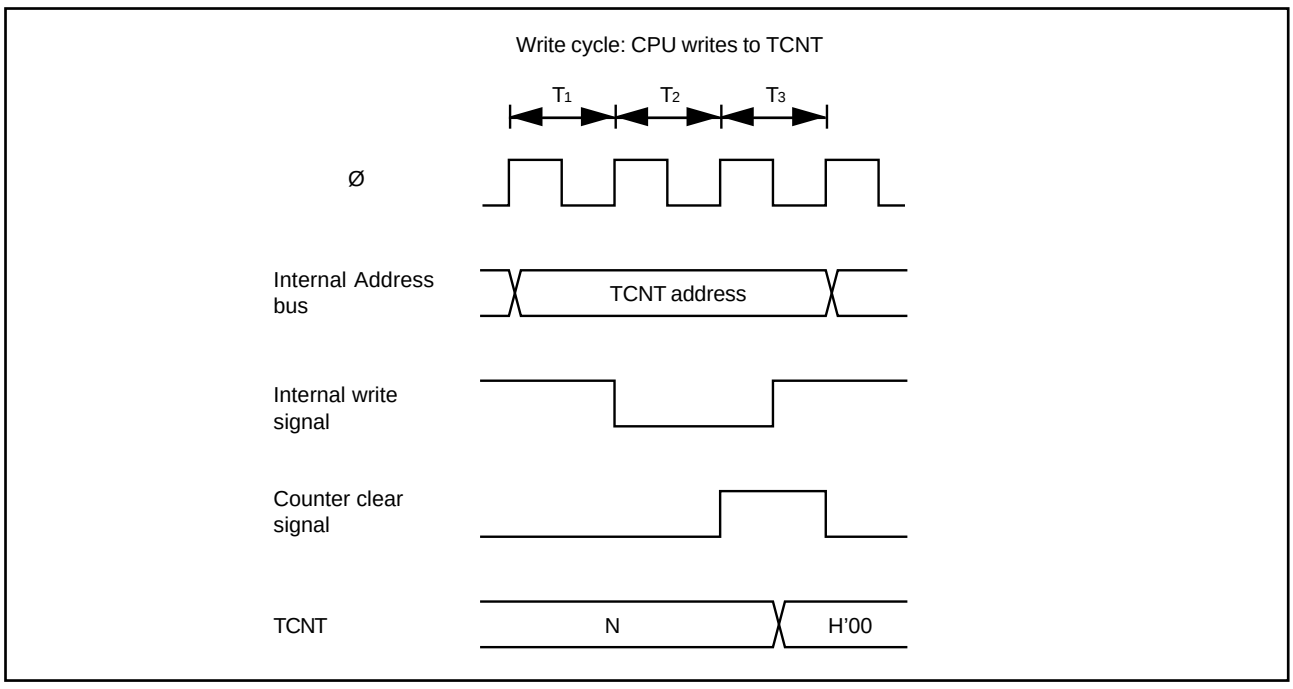



**Figure 8-13. TCNT Write-Clear Contention**

**(2) Contention between TCNT Write and Increment:** If a timer counter increment 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.

Figure 8-14 shows this type of contention.

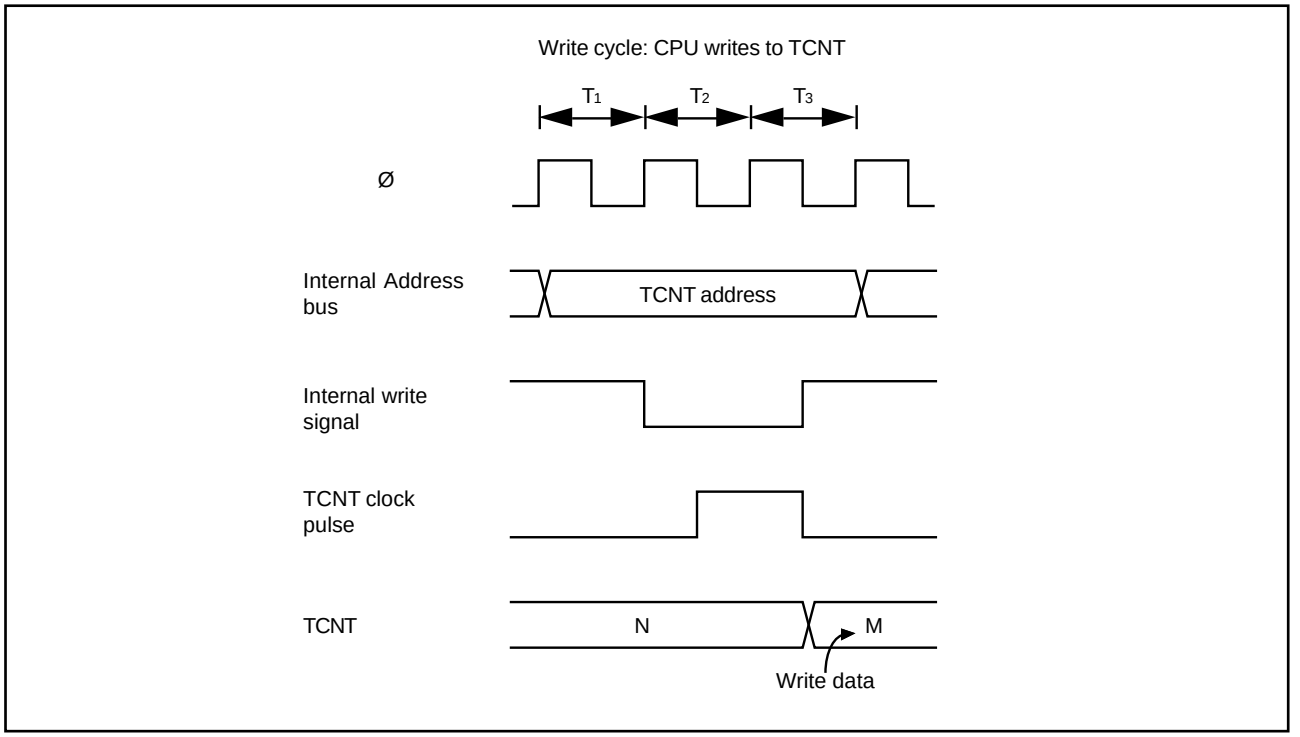



**Figure 8-14. TCNT Write-Increment Contention**

**(3) Contention between TCOR Write and Compare-Match:** If a compare-match occurs during the $T_3$ state of a write cycle to TCORA or TCORB, the write takes precedence and the compare-match signal is inhibited.

Figure 8-15 shows this type of contention.

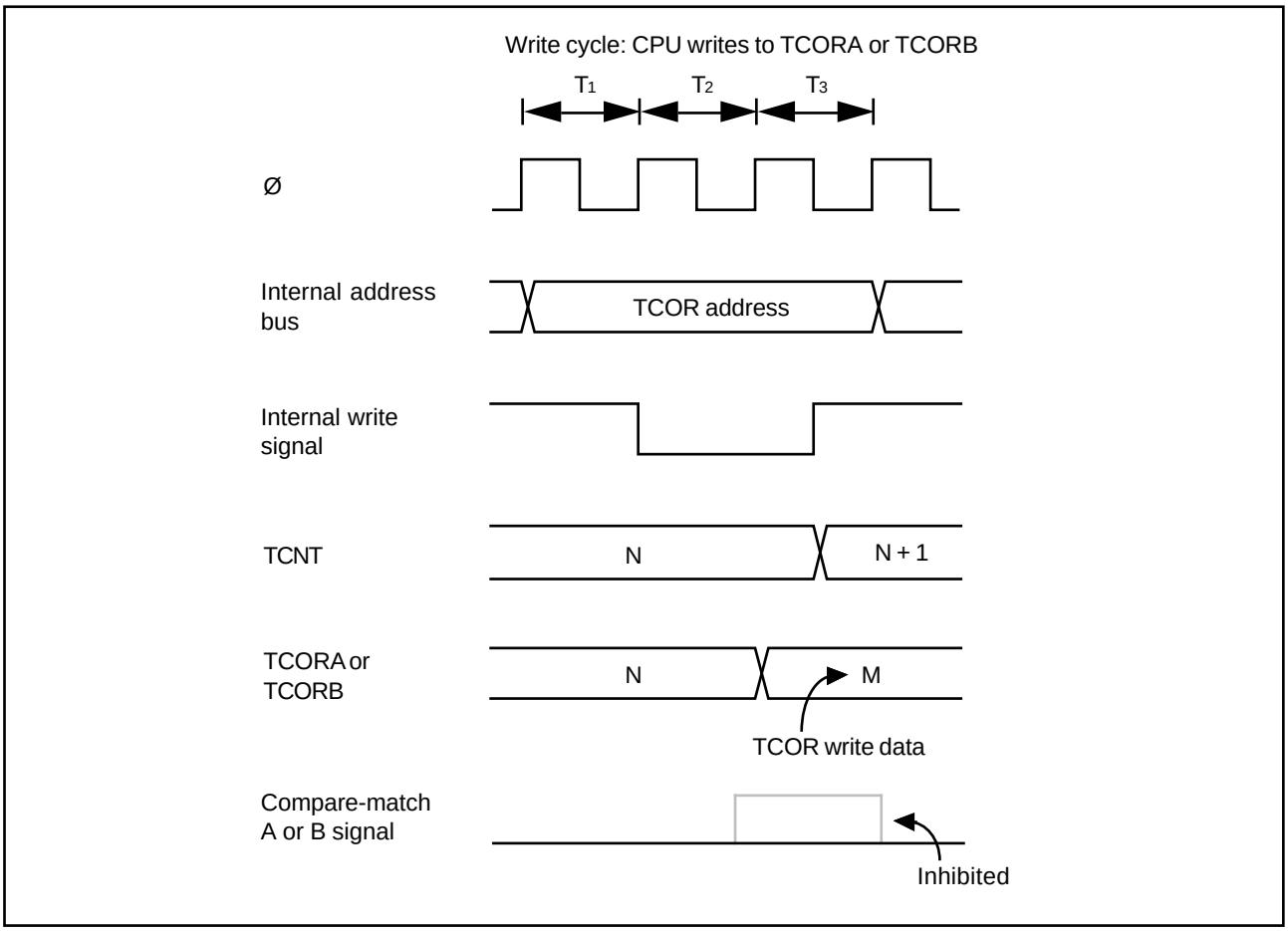



**Figure 8-15. Contention between TCOR Write and Compare-Match**

**(4) Contention between Compare-Match A and Compare-Match B:** If identical time constants are written in TCORA and TCORB, causing compare-match A and B to occur simultaneously, any conflict between the output selections for compare-match A and B is resolved by following the priority order in table 8-4.

**Table 8-4. Priority of Timer Output**

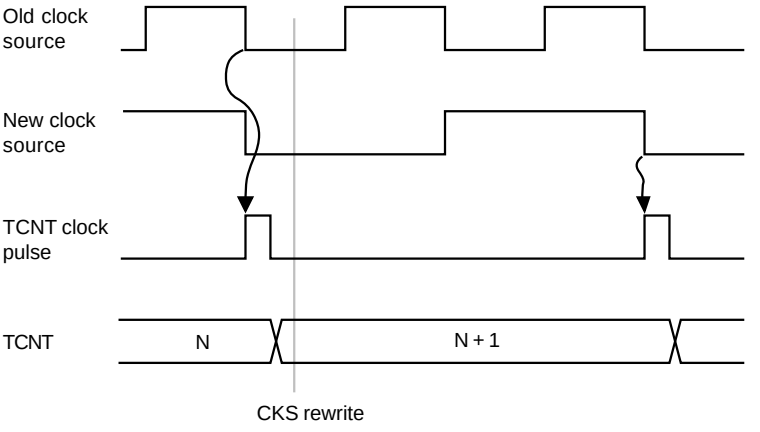
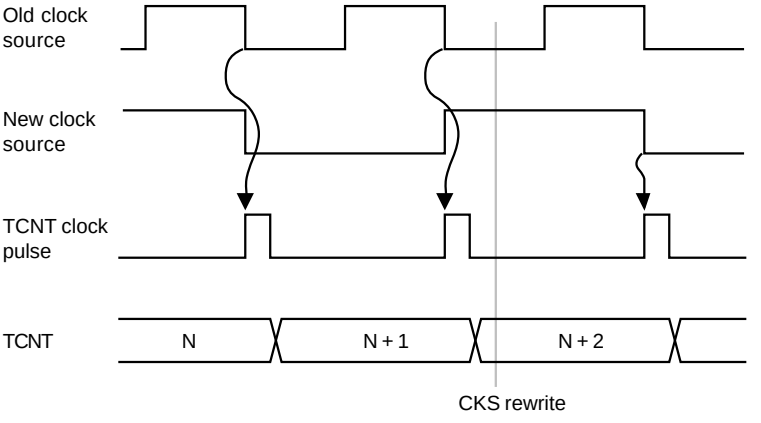
| Output selection | Priority |
|---|---|
| Toggle | High |
| 1 Output | ↑ |
| 0 Output | |
| No change | Low |

**(5) Incrementation Caused by Changing of Internal Clock Source:** When an internal clock source is changed, the changeover may cause the timer counter to increment. This depends on the time at which the clock select bits (CKS2 to CKS0) are rewritten, as shown in table 8-5.

The pulse that increments the timer counter is generated at the falling edge of the internal clock source signal. If clock sources are changed when the old source is high and the new source is low, as in case No. 3 in table 8-5, the changeover generates a falling edge that triggers the TCNT clock pulse and increments the timer counter.

Switching between an internal and external clock source can also cause the timer counter to increment. This type of switching should be avoided at external clock edges.

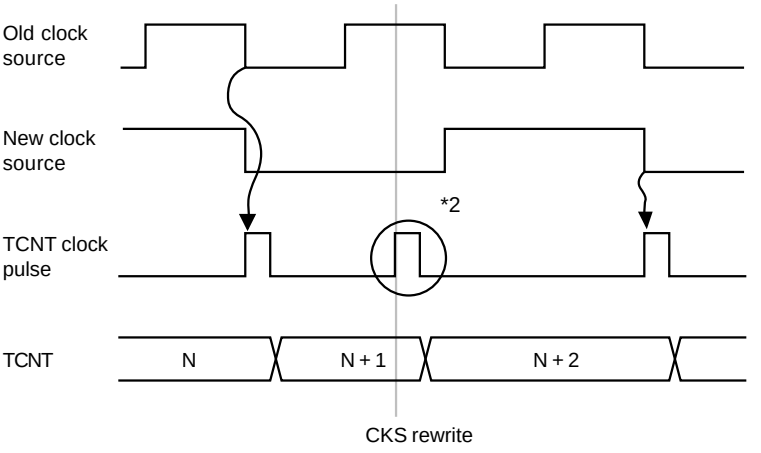
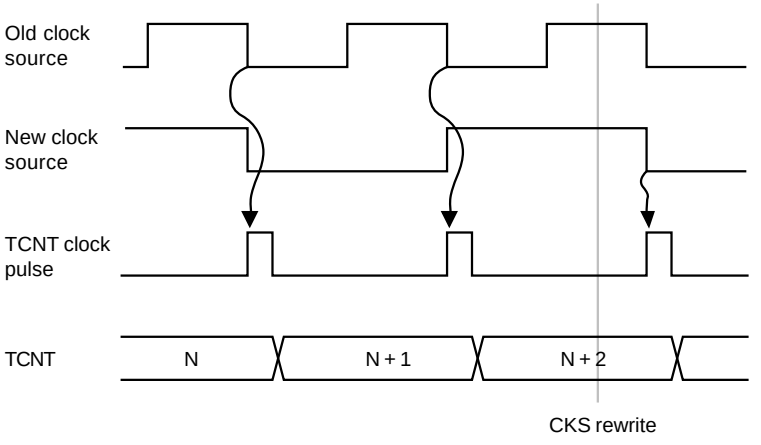
**Table 8-5. Effect of Changing Internal Clock Sources**

| No. | Description | Timing chart |
|---|---|---|
| 1 | Low → Low*1: CKS1 and CKS0 are rewritten while both clock sources are low. |   |
| 2 | Low → High*2: CKS1 and CKS0 are rewritten while old clock source is low and new clock source is high. |   |

*1 Including a transition from low to the stopped state (CKS1 = 0, CKS0 = 0), or a transition from the stopped state to low.

*2 Including a transition from the stopped state to high.

**Table 8-5. Effect of Changing Internal Clock Sources (cont.)**

| No. | Description | Timing chart |
|---|---|---|
| 3 | High → Low*1: CKS1 and CKS0 are rewritten while old clock source is high and new clock source is low. |   |
| 4 | High → High: CKS1 and CKS0 are rewritten while both clock sources are high. |   |

*1 Including a transition from high to the stopped state.

*2 The switching of clock sources is regarded as a falling edge that increments the TCNT.

# Section 9. Serial Communication Interface

## 9.1 Overview

The H8/325 series chips include a serial communication interface module (SCI) with two channels for transferring serial data to and from other chips. Either synchronous or asynchronous communication can be selected. Communication control functions are provided by internal registers.

### 9.1.1 Features

The features of the on-chip serial communication interface are:

- Asynchronous and synchronous modes
  - Asynchronous mode
    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. Eight data formats are available.
    - Data length: 7 or 8 bits
    - Stop bit length: 1 or 2 bits
    - Parity: Even, odd, or none
    - Error detection: Parity, overrun, and framing errors
  - Synchronous mode
    The SCI can communicate with chips able to perform clocked serial data transfer.
    - Data length: 8 bits
    - Error detection: Overrun errors
- Full duplex communication
  The transmitting and receiving sections are independent, so the SCI can transmit and receive simultaneously. Both the transmit and receive sections use double buffering, so continuous data transfer is possible in either direction.
- Built-in baud rate generator
  Any specified baud rate can be generated.
- Internal or external clock source
  The baud rate generator can operate on an internal clock source, or an external clock signal can be input at the SCK pin.
- Three interrupts
  Transmit-end, receive-end, and receive-error interrupts are requested independently.

### 9.1.2 Block Diagram

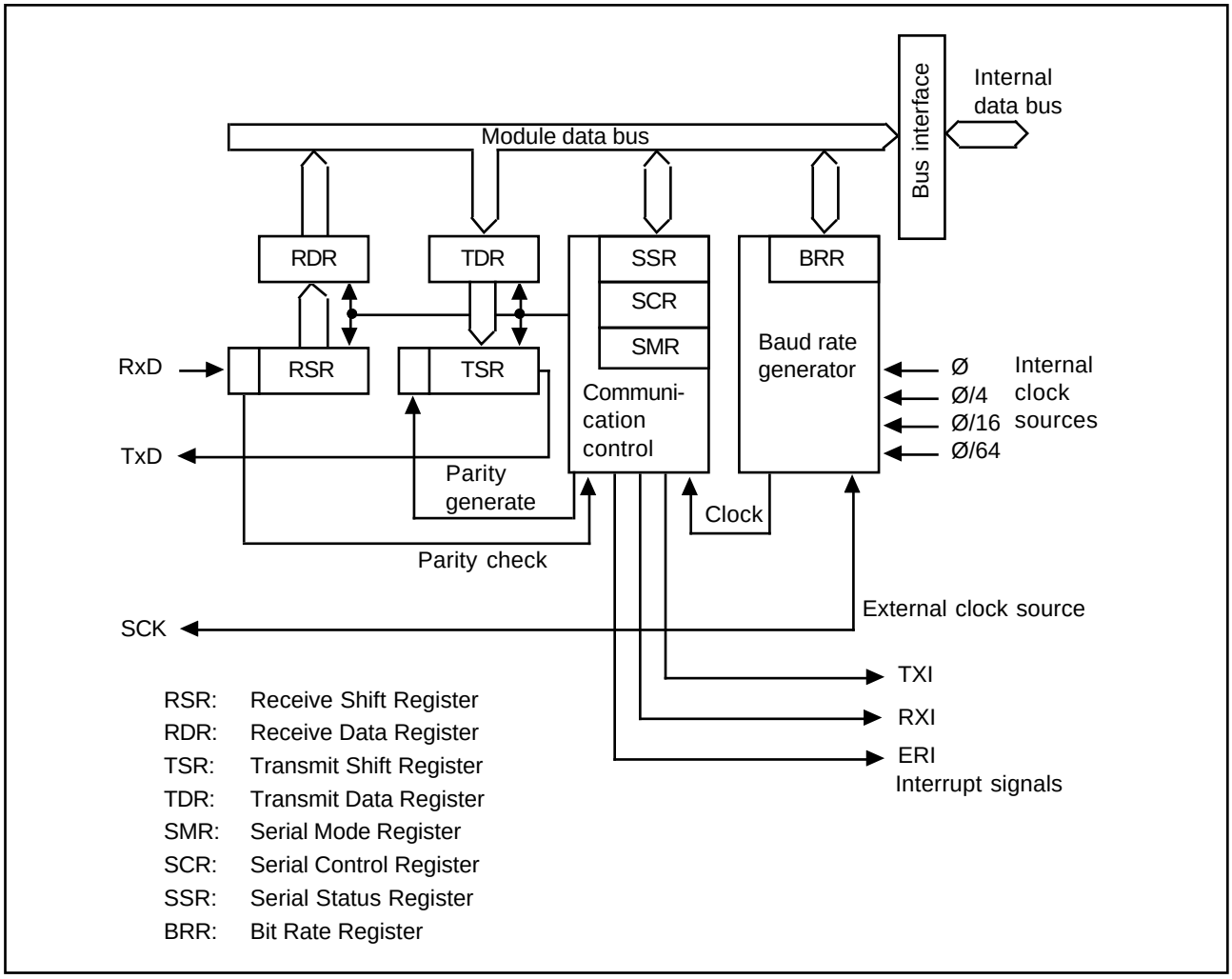



**Figure 9-1. Block Diagram of Serial Communication Interface**

### 9.1.3 Input and Output Pins

Table 9-1 lists the input and output pins used by the SCI module.

**Table 9-1. SCI Input/Output Pins**

| | Abbreviation | | | |
|---|---|---|---|---|
| **Name** | **Channel 0** | **Channel 1** | **I/O** | **Function** |
| Serial clock | $SCK_0$ | $SCK_1$ | Input/output | Serial clock input and output. |
| Serial receive data | $RxD_0$ | $RxD_1$ | Input | Receive data input. |
| Serial transmit data | $TxD_0$ | $TxD_1$ | Output | Transmit data output. |

### 9.1.4 Register Configuration

Table 9-2 lists the SCI registers.

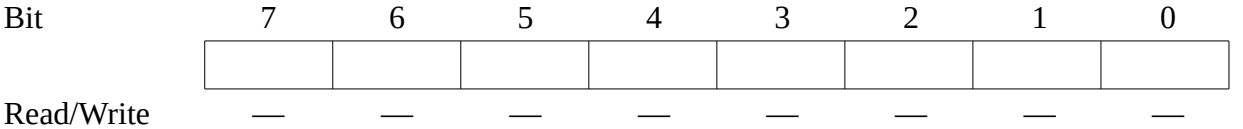
**Table 9-2. SCI Registers**

| Channel | Name | Abbreviation | R/W | Initial value | Address |
|---|---|---|---|---|---|
| 0 | Receive shift register | RSR | — | — | — |
| | Receive data register | RDR | R | H’00 | H’FFDD |
| | Transmit shift register | TSR | — | — | — |
| | Transmit data register | TDR | R/W | H’FF | H’FFDB |
| | Serial mode register | SMR | R/W | H’04 | H’FFD8 |
| | Serial control register | SCR | R/W | H’0C | H’FFDA |
| | Serial status register | SSR | R/(W)* | H’87 | H’FFDC |
| | Bit rate register | BRR | R/W | H’FF | H’FFD9 |
| 1 | Receive shift register | RSR | — | — | — |
| | Receive data register | RDR | R | H’00 | H’FFE5 |
| | Transmit shift register | TSR | — | — | — |
| | Transmit data register | TDR | R/W | H’FF | H’FFE3 |
| | Serial mode register | SMR | R/W | H’04 | H’FFE0 |
| | Serial control register | SCR | R/W | H’0C | H’FFE2 |
| | Serial status register | SSR | R/(W)* | H’87 | H’FFE4 |
| | Bit rate register | BRR | R/W | H’FF | H’FFE1 |

**Notes:**
* Software can write a 0 to clear the status flag bits, but cannot write a 1.

## 9.2 Register Descriptions

### 9.2.1 Receive Shift Register (RSR)




The RSR receives incoming data bits. When one data character (1 byte) has been received, it is transferred to the receive data register (RDR).

The CPU cannot read or write the RSR directly

### 9.2.2 Receive Data Register (RDR) – H'FFDD

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

The RDR stores received data. As each character is received, it is transferred from the RSR to the RDR, enabling the RSR to receive the next character. This double-buffering allows the SCI to receive data continuously.

The CPU can read but not write the RDR. The RDR is initialized to H'00 at a reset and in the standby modes.

### 9.2.3 Transmit Shift Register (TSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Read/Write | — | — | — | — | — | — | — | — |

The TSR holds the character currently being transmitted. When transmission of this character is completed, the next character is moved from the transmit data register (TDR) to the TSR and transmission of that character begins. If the CPU has not written the next character in the TDR, no data are transmitted.

The CPU cannot read or write the TSR directly.

### 9.2.4 Transmit Data Register (TDR) – H'FFDB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The TDR is an 8-bit readable/writable register that holds the next character to be transmitted. When the TSR becomes empty, the character written in the TDR is transferred to the TSR. Continuous data transmission is possible by writing the next byte in the TDR while the current byte is being transmitted from the TSR.

The TDR is initialized to H'FF at a reset and in the standby modes.

### 9.2.5 Serial Mode Register (SMR) – H'FFD8

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | C/$\overline{A}$ | CHR | PE | O/$\overline{E}$ | STOP | — | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | — | R/W | R/W |

The SMR is an 8-bit readable/writable register that controls the communication format and selects the clock rate for the internal clock source. It is initialized to H'04 at a reset and in the standby modes.

For further information on communication formats, see tables 9-5 and 9-7 section 9.3, Operation.

**Bit 7 – Communication Mode (C/$\overline{A}$):** This bit selects the asynchronous or synchronous communication mode.

**Bit 7**

| C/$\overline{A}$ | Description | |
|---|---|---|
| 0 | Asynchronous communication. | (Initial value) |
| 1 | Clock-synchronized communication. | |

**Bit 6 – Character Length (CHR):** This bit selects the character length in asynchronous mode. It is ignored in synchronous mode.

**Bit 6**

| CHR | Description | |
|---|---|---|
| 0 | 8 Bits per character. | (Initial value) |
| 1 | 7 Bits per character. | |

**Bit 5 – Parity Enable (PE):** This bit selects whether to add a parity bit in asynchronous mode. It is ignored in synchronous mode.

**Bit 5**

| PE | Description | |
|---|---|---|
| 0 | Transmit: No parity bit is added.<br>Receive: Parity is not checked. | (Initial value) |
| 1 | Transmit: A parity bit is added.<br>Receive: Parity is checked. | |

**Bit 4 – Parity Mode (O/$\overline{E}$ ):** In asynchronous mode, when parity is enabled (PE = 1), this bit selects even or odd parity.

Even parity means that a parity bit is added to the data bits for each character to make the total number of 1's even. Odd parity means that the total number of 1's is made odd.

This bit is ignored when PE = 0, and in the synchronous mode.

**Bit 4**

| **O/$\overline{E}$** | **Description** | |
|---|---|---|
| 0 | Even parity. | (Initial value) |
| 1 | Odd parity. | |

**Bit 3 – Stop Bit Length (STOP):** This bit selects the number of stop bits. It is ignored in the synchronous mode.

**Bit 3**

| **STOP** | **Description** | |
|---|---|---|
| 0 | 1 Stop bit. | (Initial value) |
| 1 | 2 Stop bits. | |

**Bit 2 – Reserved:** This bit cannot be modified and is always read as 1.

**Bits 1 and 0 – Clock Select 1 and 0 (CKS1 and CKS0):** These bits select the internal clock source when the baud rate generator is clocked internally.

| **Bit 1** | **Bit 0** | | |
|---|---|---|---|
| **CKS1** | **CKS0** | **Description** | |
| 0 | 0 | Ø clock | (Initial value) |
| 0 | 1 | Ø/4 clock | |
| 1 | 0 | Ø/16 clock | |
| 1 | 1 | Ø/64 clock | |

For further information about SMR settings, see tables 9-5 to 9-7 in Section 9.3, Operation.

### 9.2.6 Serial Control Register (SCR) – H'FFDA

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 |
| Initial value | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | — | — | R/W | R/W |

The SCR is an 8-bit readable/writable register that enables or disables various SCI functions. It is initialized to H'0C at a reset and in the standby modes.

**Bit 7 – Transmit Interrupt Enable (TIE):** This bit enables or disables the transmit-end interrupt (TXI) requested when the transmit data register empty (TDRE) bit in the serial status register (SSR) is set to 1.

**Bit 7**

| TIE | Description | |
|---|---|---|
| 0 | The transmit-end interrupt request (TXI) is disabled. | (Initial value) |
| 1 | The transmit-end interrupt request (TXI) is enabled. | |

**Bit 6 – Receive Interrupt Enable (RIE):** This bit enables or disables the receive-end interrupt (RXI) requested when the receive data register full (RDRF) bit in the serial status register (SSR) is set to 1, and the receive error interrupt (ERI) requested when the overrun error bit (ORER), framing error bit (FER), or parity error bit (PER) is set to 1.

**Bit 6**

| RIE | Description | |
|---|---|---|
| 0 | The receive-end interrupt (RXI) request is disabled. | (Initial value) |
| 1 | The receive-end interrupt (RXI) request is enabled. | |

**Bit 5 – Transmit Enable (TE):** This bit enables or disables the transmit function. When the transmit function is enabled, the TxD pin is automatically used for output. When the transmit function is disabled, the TxD pin can be used as a general-purpose I/O port.

**Bit 5**

| TE | Description | |
|---|---|---|
| 0 | The transmit function is disabled.<br>The TxD pin can be used for general-purpose I/O. | (Initial value) |
| 1 | The transmit function is enabled.<br>The TxD pin is used for output. | |

**Bit 4 – Receive Enable (RE):** This bit enables or disables the receive function. When the receive function is enabled, the RxD pin is automatically used for input. When the receive function is disabled, the RxD pin is available as a general-purpose I/O port.

**Bit 4**

| RE | Description | |
|---|---|---|
| 0 | The receive function is disabled. The RxD pin can be used for general-purpose I/O. | (Initial value) |
| 1 | The receive function is enabled. The RxD pin is used for input. | |

**Bits 3 and 2 – Reserved:** These bits cannot be modified and are always read as 1.

**Bit 1 – Clock Enable 1 (CKE1):** This bit selects the internal or external clock source for the baud rate generator. When the external clock source is selected, the SCK pin is automatically used for input of the external clock signal.

**Bit 1**

| CKE1 | Description | |
|---|---|---|
| 0 | Internal clock source.<br>When C/$\overline{A}$ = 1, the clock is output at SCK.<br>When C/$\overline{A}$ = 0, clock output depends on CKE0. | (Initial value) |
| 1 | External clock source, input at SCK. | |

**Bit 0 – Clock Enable 0 (CKE0):** When an internal clock source is used in asynchronous mode, this bit enables or disables serial clock output at the SCK pin.

This bit is ignored when the external clock is selected, or when the synchronous mode is selected.

**Bit 0**

| CKE0 | Description | |
|---|---|---|
| 0 | The SCK pin is not used by the SCI (and is available as a general-purpose I/O port). | (Initial value) |
| 1 | The SCK pin is used for serial clock output. | |

For further information on clock source selection, see table 9-6 in Section 9.3, Operation.

### 9.2.7 Serial Status Register (SSR) – H'FFDC

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | RDRF | ORER | FER | PER | — | — | — |
| Initial value | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Read/Write | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W)* | — | — | — |

* Software can write a 0 to clear the flags, but cannot write a 1 in these bits.

The SSR is an 8-bit register that indicates transmit and receive status. It is initialized to H'87 at a reset and in the standby modes.

**Bit 7 – Transmit Data Register Empty (TDRE):** This bit indicates when the TDR contents have been transferred to the TSR and the next character can safely be written in the TDR.

**Bit 7**

| TDRE | Description |
|---|---|
| 0 | To clear TDRE, the CPU must read TDRE after it has been set to 1, then write a 0 in this bit. |
| 1 | This bit is set to 1 at the following times: (Initial value)<br>(1) When TDR contents are transferred to the TSR.<br>(2) When the TE bit in the SCR is cleared to 0. |

**Bit 6 – Receive Data Register Full (RDRF):** This bit indicates when one character has been received and transferred to the RDR.

**Bit 6**

| RDRF | Description |
|---|---|
| 0 | To clear RDRF, the CPU must read RDRF after it has been set to 1, then write a 0 in this bit. (Initial value) |
| 1 | This bit is set to 1 when one character is received without error and transferred from the RSR to the RDR. |

**Bit 5 – Overrun Error (ORER):** This bit indicates an overrun error during reception.

| Bit 5<br>ORER | Description | |
|---|---|---|
| 0 | To clear ORER, the CPU must read ORER after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 if reception of the next character ends while the receive data register is still full (RDRF = 1). | |

**Bit 4 – Framing Error (FER):** This bit indicates a framing error during data reception in asynchronous mode. It has no meaning in synchronous mode.

| Bit 4<br>FER | Description | |
|---|---|---|
| 0 | To clear FER, the CPU must read FER after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 if a framing error occurs (stop bit = 0). | |

**Bit 3 – Parity Error (PER):** This bit indicates a parity error during data reception in asynchronous mode, when a communication format with parity bits is used.

This bit has no meaning in synchronous mode, or when a communication format without parity bits is used.

| Bit 3<br>PER | Description | |
|---|---|---|
| 0 | To clear PER, the CPU must read PER after it has been set to 1, then write a 0 in this bit. | (Initial value) |
| 1 | This bit is set to 1 when a parity error occurs (the parity of the received data does not match the parity selected by the O/$\overline{E}$ bit in the SMR). | |

**Bits 2 to 0 – Reserved:** These bits cannot be modified and are always read as 1.

### 9.2.8 Bit Rate Register (BRR) – H’FFD9

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The BRR is an 8-bit register that, together with the CKS1 and CKS0 bits in the SMR, determines the baud rate output by the baud rate generator.

The BRR is initialized to H’FF (the slowest rate) at a reset and in the standby modes.

Tables 9-3 and 9-4 show examples of BRR (N) and CKS (n) settings for commonly used bit rates.

**Table 9-3. Examples of BRR Settings in Asynchronous Mode (1)**

| | XTAL Frequency (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | | | 2.4576 | | | 4 | | | 4.194304 | | |
| Bit rate | 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 | 1 | 0 | — | — | — |
| 38400 | — | — | — | 0 | 0 | 0 | — | — | — | — | — | — |

**Table 9-3. Examples of BRR Settings in Asynchronous Mode (2)**

| | XTAL Frequency (MHz) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 4.9152 | | | 6 | | | 7.3728 | | | 8 | |
| Bit rate | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) | n | N | Error (%) |
| 110 | 1 | 174 | –0.26 | 2 | 52 | +0.50 | 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 | 11 | 0 | 0 | 12 | +0.16 |
| 19200 | 0 | 3 | 0 | — | — | — | 0 | 5 | 0 | — | — | — |
| 31250 | — | — | — | 0 | 2 | 0 | — | — | — | 0 | 3 | 0 |
| 38400 | 0 | 1 | 0 | — | — | — | 0 | 2 | 0 | — | — | — |

**Table 9-3. Examples of BRR Settings in Asynchronous Mode (3)**

| | XTAL Frequency (MHz) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 9.8304 | | | 10 | | | 12 | | | 12.288 | |
| Bit rate | 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 | 4 | 0 |
| 31250 | 0 | 4 | –1.70 | 0 | 4 | 0 | 0 | 5 | 0 | 0 | 5 | +2.40 |
| 38400 | 0 | 3 | 0 | 0 | 3 | +1.73 | — | — | — | — | — | — |

**Table 9-3. Examples of BRR Settings in Asynchronous Mode (4)**

| | XTAL Frequency (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 14.7456 | | | 16 | | | 19.6608 | | | 20 | | |
| Bit rate | 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 |

**Note:** If possible, the error should be within 1%.

$$B = OSC \times 10^6 / [64 \times 2^{2n} \times (N + 1)]$$

N: BRR value ($0 \leq N \leq 255$)
OSC: Crystal oscillator frequency in MHz
B: Bit rate (bits/second)
n: Internal clock source (0, 1, 2, or 3)

The meaning of n is given by the table below:

| n | CKS1 | CKS0 | Clock |
|---|---|---|---|
| 0 | 0 | 0 | Ø |
| 1 | 0 | 1 | Ø/4 |
| 2 | 1 | 0 | Ø/16 |
| 3 | 1 | 1 | Ø/64 |

**Table 9-4. Examples of BRR Settings in Synchronous Mode**

| | XTAL Frequency (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bit** | **2** | | **4** | | **8** | | **10** | | **16** | | **20** | |
| **rate** | **n** | **N** | **n** | **N** | **n** | **N** | **n** | **N** | **n** | **N** | **n** | **N** |
| 100 | — | — | — | — | — | — | — | — | — | — | — | — |
| 250 | 1 | 249 | 2 | 124 | 2 | 249 | — | — | 3 | 124 | — | — |
| 500 | 1 | 124 | 1 | 249 | 2 | 124 | — | — | 2 | 249 | — | — |
| 1k | 0 | 249 | 1 | 124 | 1 | 249 | — | — | 2 | 124 | — | — |
| 2.5k | 0 | 99 | 0 | 199 | 1 | 99 | 1 | 124 | 1 | 199 | 1 | 249 |
| 5k | 0 | 49 | 0 | 99 | 0 | 199 | 0 | 249 | 1 | 99 | 1 | 124 |
| 10k | 0 | 24 | 0 | 49 | 0 | 99 | 0 | 124 | 0 | 199 | 0 | 249 |
| 25k | 0 | 9 | 0 | 19 | 0 | 39 | 0 | 49 | 0 | 79 | 0 | 99 |
| 50k | 0 | 4 | 0 | 9 | 0 | 19 | 0 | 24 | 0 | 39 | 0 | 49 |
| 100k | — | — | 0 | 4 | 0 | 9 | — | — | 0 | 19 | 0 | 24 |
| 250k | 0 | 0* | 0 | 1 | 0 | 3 | 0 | 4 | 0 | 7 | 0 | 9 |
| 500k | | | 0 | 0* | 0 | 1 | — | — | 0 | 3 | 0 | 4 |
| 1M | | | | | 0 | 0* | — | — | 0 | 1 | — | — |
| 2.5M | | | | | | | | | | | 0 | 0* |

**Notes:**

Blank: No setting is available.

—: A setting is available, but the bit rate is inaccurate.

*: Continuous transfer is not possible.

$$B = OSC \times 10^6/[8 \times 2^{2n} \times (N + 1)]$$

N: BRR value ($0 \le N \le 255$)

OSC: Crystal oscillator frequency in MHz

B: Bit rate (bits per second)

n: Internal clock source (0, 1, 2, or 3)

The meaning of n is given by the table below:

| **n** | **CKS1** | **CKS0** | **Clock** |
|---|---|---|---|
| 0 | 0 | 0 | Ø |
| 1 | 0 | 1 | Ø/4 |
| 2 | 1 | 0 | Ø/16 |
| 3 | 1 | 1 | Ø/64 |

## 9.3 Operation

### 9.3.1 Overview

The SCI supports serial data transfer in both asynchronous and synchronous modes.

The communication format depends on settings in the SMR as indicated in table 9-5. The clock source and usage of the SCK pin depend on settings in the SMR and SCR as indicated in table 9-6.

**Table 9-5. Communication Formats Used by SCI**

| SMR | | | | | | | |
|---|---|---|---|---|---|---|---|
| **C/$\overline{A}$** | **CHR** | **PE** | **STOP** | **Mode** | **Format** | **Parity** | **Stop bit length** |
| 0 | 0 | 0 | 0 | Asynchronous | 8-Bit data | None | 1 |
| | | | 1 | | | | 2 |
| | | 1 | 0 | | | Yes | 1 |
| | | | 1 | | | | 2 |
| | 1 | 0 | 0 | | 7-Bit data | None | 1 |
| | | | 1 | | | | 2 |
| | | 1 | 0 | | | Yes | 1 |
| | | | 1 | | | | 2 |
| 1 | — | — | — | Synchronous | 8-Bit data | — | — |

**Table 9-6. SCI Clock Source Selection**

| SMR | SCR | | Clock | |
|---|---|---|---|---|
| **C/$\overline{A}$** | **CKE1** | **CKE0** | **source** | **SCK pin** |
| 0 (Async mode) | 0 | 0 | Internal | Input/output port* |
| | | 1 | | Serial clock output at bit rate |
| | 1 | 0 | External | Serial clock input at 16 × bit rate |
| | | 1 | | |
| 1 (Sync mode) | 0 | 0 | Internal | Serial clock output |
| | | 1 | | |
| | 1 | 0 | External | Serial clock input |
| | | 1 | | |

* Not used by the SCI.

Transmitting and receiving operations in the two modes are described next.

### 9.3.2 Asynchronous Mode

In asynchronous mode, each character is individually synchronized by framing it with a start bit and stop bit.

Full duplex data transfer is possible because the SCI has independent transmit and receive sections. Double buffering in both sections enables the SCI to be programmed for continuous data transfer.

Figure 9-2 shows the general format of one character sent or received in the asynchronous mode. The communication channel is normally held in the mark state (high). Character transmission or reception starts with a transition to the space state (low).

The first bit transmitted or received is the start bit (low). It is followed by the data bits, in which the least significant bit (LSB) comes first. The data bits are followed by the parity bit, if present, then the stop bit or bits (high) confirming the end of the frame.

In receiving, the SCI synchronizes on the falling edge of the start bit, and samples each bit at the center of the bit (at the 8th cycle of the internal serial clock, which runs at 16 times the bit rate).

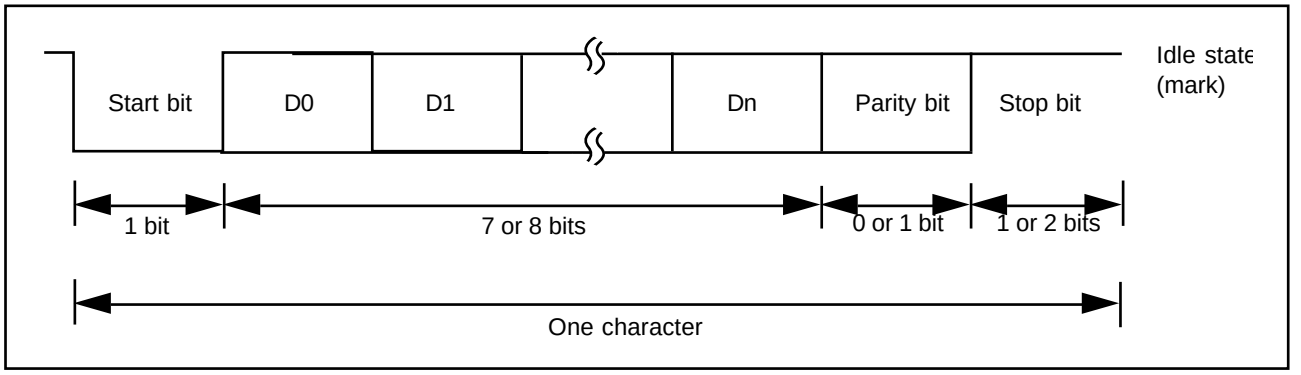



**Figure 9-2. Data Format in Asynchronous Mode**

**(1) Data Format:** Table 9-7 lists the data formats that can be sent and received in asynchronous mode. Eight formats can be selected by bits in the SMR.

**Table 9-7. Data Formats in Asynchronous Mode**

| SMR bits | | | |
|---|---|---|---|
| **CHR** | **PE** | **STOP** | **Data format** |
| 0 | 0 | 0 | START \| 8-Bit data \| STOP |
| 0 | 0 | 1 | START \| 8-Bit data \| STOP \| STOP |
| 0 | 1 | 0 | START \| 8-Bit data \| P \| STOP |
| 0 | 1 | 1 | START \| 8-Bit data \| P \| STOP \| STOP |
| 1 | 0 | 0 | START \| 7-Bit data \| STOP |
| 1 | 0 | 1 | START \| 7-Bit data \| STOP \| STOP |
| 1 | 1 | 0 | START \| 7-Bit data \| P \| STOP |
| 1 | 1 | 1 | START \| 7-Bit data \| P \| STOP \| STOP |

**Note**

START: Start bit
STOP: Stop bit
P: Parity bit

**(2) Clock:** In asynchronous mode it is possible to select either an internal clock created by the on-chip baud rate generator, or an external clock input at the SCK pin. Refer to table 9-6.

If an external clock is input at the SCK pin, its frequency should be 16 times the desired baud rate.

If the internal clock provided by the on-chip baud rate generator is selected and the SCK pin is used for clock output, the output clock frequency is equal to the baud rate, and the clock pulse rises at the center of the transmit data bits. Figure 9-3 shows the phase relationship between the output clock and transmit data.

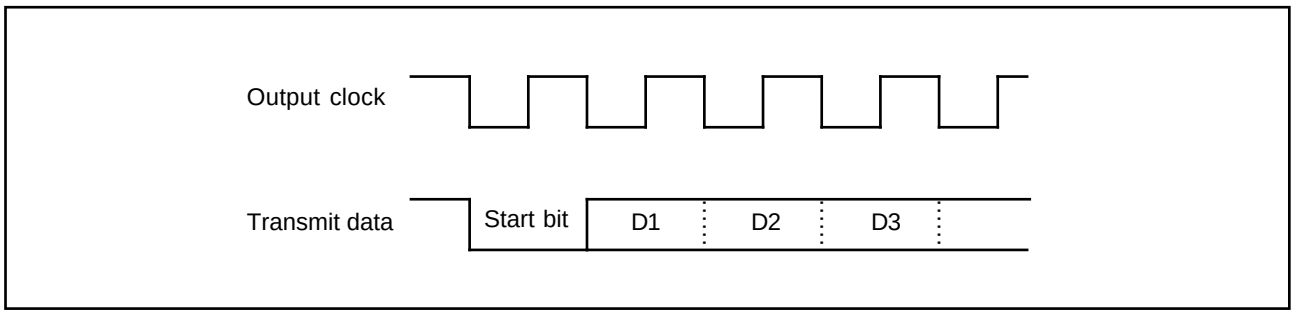



**Figure 9-3. Phase Relationship between Clock Output and Transmit Data**

### (3) Data Transmission and Reception

- **SCI Initialization:** Before data can be transmitted or received, the SCI must be initialized by software. To initialize the SCI, software must clear the TE and RE bits to 0, then execute the following procedure.

① Set the desired communication format in the SMR.

② Write the value corresponding to the desired baud rate in the BRR. (This step is not necessary if an external clock is used.)

③ Select the clock and enable desired interrupts in the SCR.

④ Set the TE and/or RE bit in the SCR to 1.

The TE and RE bits must both be cleared to 0 whenever the operating mode or data format is changed.

After changing the operating mode or data format, before setting the TE and RE bits to 1 software must wait for at least the transfer time for 1 bit at the selected baud rate, to make sure the SCI is initialized. If an external clock is used, the clock must not be stopped.

When clearing the TDRE bit during data transmission, to assure transfer of the correct data, do not clear the TDRE bit until after writing data in the TDR. Similarly, in receiving data, do not clear the RDRF bit until after reading data from the RDR.

- **Data Transmission:** The procedure for transmitting data is as follows.

① Set up the desired transmitting conditions in the SMR, SCR, and BRR.

② Set the TE bit in the SCR to 1.
The TxD pin will automatically be switched to output and one frame* of all 1's will be transmitted, after which the SCI is ready to transmit data.

③ Check that the TDRE bit is set to 1, then write the first byte of transmit data in the TDR. Next clear the TDRE bit to 0.

④ The first byte of transmit data is transferred from the TDR to the TSR and sent in the designated format as follows.
   i) Start bit (one 0 bit).
   ii) Transmit data (seven or eight bits, starting from bit 0)
   iii) Parity bit (odd or even parity bit, or no parity bit)
   iv) Stop bit (one or two consecutive 1 bits)

⑤ Transfer of the transmit data from the TDR to the TSR makes the TDR empty, so the TDRE bit is set to 1.
   If the TIE bit is set to 1, a transmit-end interrupt (TXI) is requested.
   When the transmit function is enabled but the TDR is empty (TDRE = 1), the output at the TxD pin is held at 1 until the TDRE bit is cleared to 0.

* A frame is the data for one character, including the start bit and stop bit(s).

• **Data Reception:** The procedure for receiving data is as follows.

① Set up the desired receiving conditions in the SMR, SCR, and BRR.

② Set the RE bit in the SCR to 1.
   The RxD pin is automatically be switched to input and the SCI is ready to receive data.

③ The SCI synchronizes with the incoming data by detecting the start bit, and places the received bits in the RSR. At the end of the data, the SCI checks that the stop bit is 1.

④ When a complete frame has been received, the SCI transfers the received data from the RSR to the RDR so that it can be read. If the character length is 7 bits, the most significant bit of the RDR is cleared to 0.
   At the same time, the SCI sets the RDRF bit in the SSR to 1. If the RIE bit is set to 1, a receive-end interrupt (RXI) is requested.

⑤ The RDRF bit is cleared to 0 when software reads the SSR, then writes a 0 in the RDRF bit. The RDR is then ready to receive the next character from the RSR.

When a frame is not received correctly, a receive error occurs. There are three types of receive errors, listed in table 9-8.

If a receive error occurs, the RDRF bit in the SSR is not set to 1. (For an overrun error, RDRF is already set to 1.) The corresponding error flag is set to 1 instead. If the RIE bit in the SCR is set to 1, a receive-error interrupt (ERI) is requested.

When a framing or parity error occurs, the RSR contents are transferred to the RDR. If an overrun error occurs, however, the RSR contents are not transferred to the RDR.

If multiple receive errors occur simultaneously, all the corresponding error flags are set to 1.

To clear a receive-error flag (ORER, FER, or PER), software must read the SSR and then write a 0 in the flag bit.

**Table 9-8. Receive Errors**

| Name | Abbreviation | Description |
|---|---|---|
| Overrun error | ORER | Reception of the next frame ends while the RDRF bit is still set to 1.<br>The RSR contents are not transferred to the RDR. |
| Framing error | FER | A stop bit is 0.<br>The RSR contents are transferred to the RDR. |
| Parity error | PER | The parity of a frame does not match the value selected by the O/$\overline{E}$ bit in the SMR.<br>The RSR contents are transferred to the RDR. |

### 9.3.3 Synchronous Mode

The synchronous mode is suited for high-speed, continuous data transfer. Each bit of data is synchronized with a serial clock pulse at the SCK pin.

Continuous data transfer is enabled by the double buffering employed in both the transmit and receive sections of the SCI. Full duplex communication is possible because the transmit and receive sections are independent.

**(1) Data Format:** Figure 9-4 shows the communication format used in the synchronous mode. The data length is 8 bits for both the transmit and receive directions. The least significant bit (LSB) is sent and received first. Each bit of transmit data is output from the falling edge of the serial clock pulse to the next falling edge. Received bits are latched on the rising edge of the serial clock pulse.

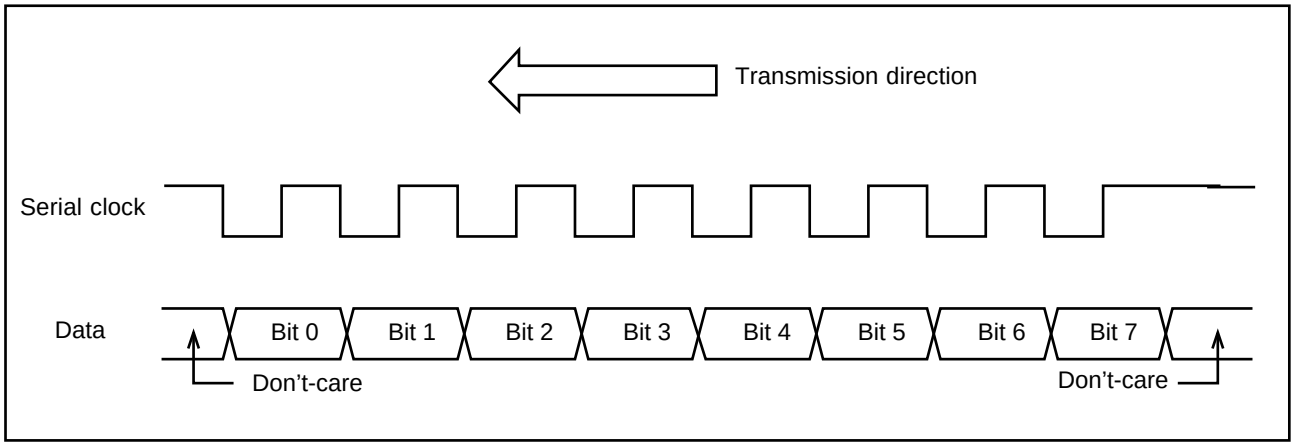



**Figure 9-4. Data Format in Synchronous Mode**

**(2) Clock:** Either the internal serial clock created by the on-chip baud rate generator or an external clock input at the SCK pin can be selected in the synchronous mode. See table 9-6 for details.

**(3) Data Transmission and Reception**

• **SCI Initialization:** Before data can be transmitted or received, the SCI must be initialized by software. To initialize the SCI, software must clear the TE and RE bits to 0 to disable both the transmit and receive functions, then execute the following procedure.

① Write the value corresponding to the desired bit rate in the BRR. (This step is not necessary if an external clock is used.)

② Select the clock and enable desired interrupts in the SCR. Leave bit 0 (CKE0) cleared to 0.

③ Select synchronous mode in the SMR.

④ Set the TE and/or RE bit in the SCR to 1.

The TE and RE bits must both be cleared to 0 whenever the operating mode or data format is changed. After changing the operating mode or data format, before setting the TE and RE bits to 1 software must wait for at least 1 bit transfer time at the selected communication speed, to make sure the SCI is initialized.

When clearing the TDRE bit during data transmission, to assure correct data transfer, do not clear the TDRE bit until after writing data in the TDR. Similarly, in receiving data, do not clear the RDRF bit until after reading data from the RDR.

- **Data Transmission:** The procedure for transmitting data is as follows.

① Set up the desired transmitting conditions in the SMR, BRR, and SCR.

② Set the TE bit in the SCR to 1.
The TxD pin will automatically be switched to output, after which the SCI is ready to transmit data.

③ Check that the TDRE bit is set to 1, then write the first byte of transmit data in the TDR. Next clear the TDRE bit to 0.

④ The first byte of transmit data is transferred from the TDR to the TSR and sent, each bit synchronized with a clock pulse. Bit 0 is sent first.
Transfer of the transmit data from the TDR to the TSR makes the TDR empty, so the TDRE bit is set to 1. If the TIE bit is set to 1, a transmit-end interrupt (TXI) is requested.

The TDR and TSR function as a double buffer. Continuous data transmission can be achieved by writing the next transmit data in the TDR and clearing the TDRE bit to 0 while the SCI is transmitting the current data from the TSR.

If an internal clock source is selected, after transferring the transmit data from the TDR to the TSR, while transmitting the data from the TSR the SCI also outputs a serial clock signal at the SCK pin. When all data bits in the TSR have been transmitted, if the TDR is empty (TDRE = 1), serial clock output is suspended until the next data byte is written in the TDR and the TDRE bit is cleared to 0. During this interval the TxD pin continues to output the value of the last bit of the previous data.

If the external clock source is selected, data transmission is synchronized with the clock signal input at the SCK pin. When all data bits in the TSR have been transmitted, if the TDR is empty (TDRE = 1) but external clock pulses continue to arrive, the TxD pin outputs the value of last bit of the previous data.

- **Data Reception:** The procedure for receiving data is as follows.

① Set up the desired receiving conditions in the SMR, BRR, and SCR.

② Set the RE bit in the SCR to 1.
The RxD pin is automatically be switched to input and the SCI is ready to receive data.

③ Incoming data bits are latched in the RSR on eight clock pulses.
When 8 bits of data have been received, the SCI sets the RDRF bit in the SSR to 1. If the RIE bit is set to 1, a receive-end interrupt (RXI) is requested.

④ The SCI transfers the received data byte from the RSR to the RDR so that it can be read.
The RDRF bit is cleared when software reads the RDRF bit in the SSR, then writes a 0 in the RDRF bit.

The RDR and RSR function as a double buffer. Data can be received continuously by reading each byte of data from the RDR and clearing the RDRF bit to 0 before the last bit of the next byte is received.

In general, an external clock source should be used for receiving data.

If an internal clock source is selected, the SCI starts receiving data as soon as the RE bit is set to 1. The serial clock is also output at the SCK pin. The SCI continues receiving until the RE bit is cleared to 0.

If the last bit of the next data byte is received while the RDRF bit is still set to 1, an overrun error occurs and the ORER bit is set to 1. If the RIE bit is set to 1, a receive-error interrupt (ERI) is requested. The data received in the RSR are not transferred to the RDR when an overrun error occurs.

After an overrun error, reception of the next data is enabled when the ORER bit is cleared to 0.

• **Simultaneous Transmit and Receive:** The procedure for transmitting and receiving simultaneously is as follows:

① Set up the desired communication conditions in the SMR, BRR, and SCR.

② Set the TE and RE bits in the SCR to 1.
The TxD and RxD pins are automatically switched to output and input, respectively, and the SCI is ready to transmit and receive data.

③ Data transmitting and receiving start when the TDRE bit in the SSR is cleared to 0.

④ Data are sent and received in synchronization with eight clock pulses.

⑤ First, the transmit data are transferred from the TDR to the TSR. This makes the TDR empty, so the TDRE bit is set to 1. If the TIE bit is set to 1, a transmit-end interrupt (TXI) is requested. If continuous data transmission is desired, software must read the TDRE bit in the SSR, write the next transmit data in the TDR, then clear the TDRE bit to 0.
If the TDRE bit is not cleared to 0 by the time the SCI finishes sending the current byte from the TSR, the TxD pin continues to output the value of last bit of the previous data.

⑥ In the receiving section, when 8 bits of data have been received they are transferred from the RSR to the RDR and the RDRF bit in the SSR is set to 1. If the RIE bit is set to 1, a receive-end interrupt (RXI) is requested.

⑦ To clear the RDRF bit software should read the RDRF bit in the SSR, read the data in the RDR, then write a 0 in the RDRF bit.
For continuous data reception, software should clear the RDRF bit to 0 before reception of the next 8 bits is completed.

If the last bit of the next byte is received while the RDRF bit is still set to 1, an overrun error occurs. The error is handled as described under “Data Reception” above. The overrun error does not affect the transmit section of the SCI, which continues to transmit normally.

## 9.4 Interrupts

The SCI can request three types of interrupts: transmit-end (TXI), receive-end (RXI), and receive-error (ERI). Interrupt requests are enabled or disabled by the TIE and RIE bits in the SCR. Independent signals are sent to the interrupt controller for each type of interrupt. The transmit-end and receive-end interrupt request signals are obtained from the TDRE and RDRF flags. The receive-error interrupt request signal is the logical OR of the three error flags: overrun error (ORER), framing error (FER), and parity error (PER). Table 9-9 lists information about these interrupts.

**Table 9-9. SCI Interrupts**

| Interrupt | Description | Priority |
|---|---|---|
| ERI | Receive-error interrupt, requested when ORER, FER, or PER is set. RIE must also be set. | High |
| RXI | Receive-end interrupt, requested when RDRF and RIE are set. | |
| TXI | Transmit-end interrupt, requested when TDRE and TIE are set. | Low |

## 9.5 Application Notes

Application programmers should note the following features of the SCI.

**(1) TDR Write:** The TDRE bit in the SSR is simply a flag that indicates that the TDR contents have been transferred to the TSR. The TDR contents can be rewritten regardless of the TDRE value. If a new byte is written in the TDR while the TDRE bit is 0, before the old TDR contents have been moved into the TSR, the old byte will be lost. Normally, software should check that the TDRE bit is set to 1 before writing to the TDR.

**(2) Multiple Receive Errors:** Table 9-10 lists the values of flag bits in the SSR when multiple receive errors occur, and indicates whether the RSR contents are transferred to the RDR.

**Table 9-10. SSR Bit States and Data Transfer When Multiple Receive Errors Occur**

| | SSR Bits | | | | |
|---|---|---|---|---|---|
| **Receive error** | **RDRF** | **ORER** | **FER** | **PER** | **RSR → RDR*₂** |
| Overrun error | 1*₁ | 1 | 0 | 0 | No |
| Framing error | 0 | 0 | 1 | 0 | Yes |
| Parity error | 0 | 0 | 0 | 1 | Yes |
| Overrun + framing errors | 1*₁ | 1 | 1 | 0 | No |
| Overrun + parity errors | 1*₁ | 1 | 0 | 1 | No |
| Framing + parity errors | 0 | 0 | 1 | 1 | Yes |
| Overrun + framing + parity errors | 1*₁ | 1 | 1 | 1 | No |

*₁ Set to 1 before the overrun error occurs.

*₂ Yes: The RSR contents are transferred to the RDR.
No: The RSR contents are not transferred to the RDR.

**(3) Line Break Detection:** When the RxD pin receives a continuous stream of 0's in asynchronous mode (line-break state), a framing error occurs because the SCI detects a 0 stop bit. The value H'00 is transferred from the RSR to the RDR. Software can detect the line-break state as a framing error accompanied by H'00 data in the RDR.

The SCI continues to receive data, so if the FER bit is cleared to 0 another framing error will occur.

**(4) Sampling Timing and Receive Margin in Asynchronous Mode:** The serial clock used by the SCI in asynchronous mode runs at 16 times the baud rate. The falling edge of the start bit is detected by sampling the RxD input on the falling edge of this clock. After the start bit is detected, each bit of receive data in the frame (including the start bit, parity bit, and stop bit or bits) is sampled on the rising edge of the serial clock pulse at the center of the bit. See figure 9-6.

It follows that the receive margin can be calculated as in equation (1).

When the absolute frequency deviation of the clock signal is 0 and the clock duty factor is 0.5, data can theoretically be received with distortion up to the margin given by equation (2). This is a theoretical limit, however. In practice, system designers should allow a margin of 20% to 30%.

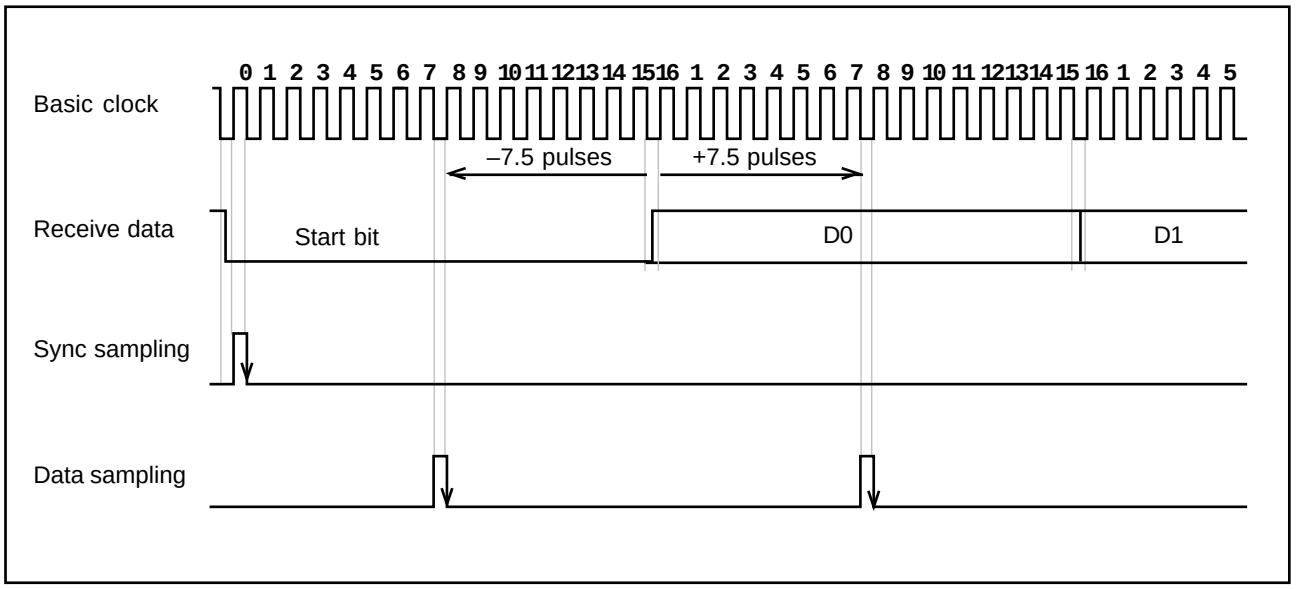



**Figure 9-5. Sampling Timing (Asynchronous Mode)**

$$M = \{(0.5 - 1/2N) - (D - 0.5)/N - (L - 0.5)F\} \times 100\ [\%] \quad (1)$$

- M: Receive margin
- N: Ratio of basic clock to baud rate (N = 16)
- D: Duty factor of clock—ratio of high pulse width to low width (0.5 to 1.0)
- L: Frame length (9 to 12)
- F: Absolute clock frequency deviation

When D = 0.5 and F= 0

$$M = (0.5 - 1/2 \times 16) \times 100\ [\%] = 46.875\% \quad (2)$$

# Section 10. RAM

## 10.1 Overview

The H8/3257 and H8/3256 have 2 Kbytes of on-chip static RAM, H8/325 and H8/324 have 1 Kbyte, the H8/323 has 512 bytes, and the H8/322 has 256 bytes. The on-chip RAM is connected to the CPU by a 16-bit data bus. Both byte and word access to the on-chip RAM are performed in two states, enabling rapid data transfer and instruction execution.

The on-chip RAM occupies the following addresses in the chip's address space.
H8/3257, H8/3256: H'F780 to H'FF7F
H8/325, H8/324: H'FB80 to H'FF7F
H8/323: H'FD80 to H'FF7F
H8/322: H'FE80 to H'FF7F
The RAME bit in the system control register (SYSCR) can enable or disable the on-chip RAM, permitting these addresses to be allocated to external memory instead, if so desired.

## 10.2 Block Diagram

Figure 10-1 is a block diagram of the on-chip RAM.

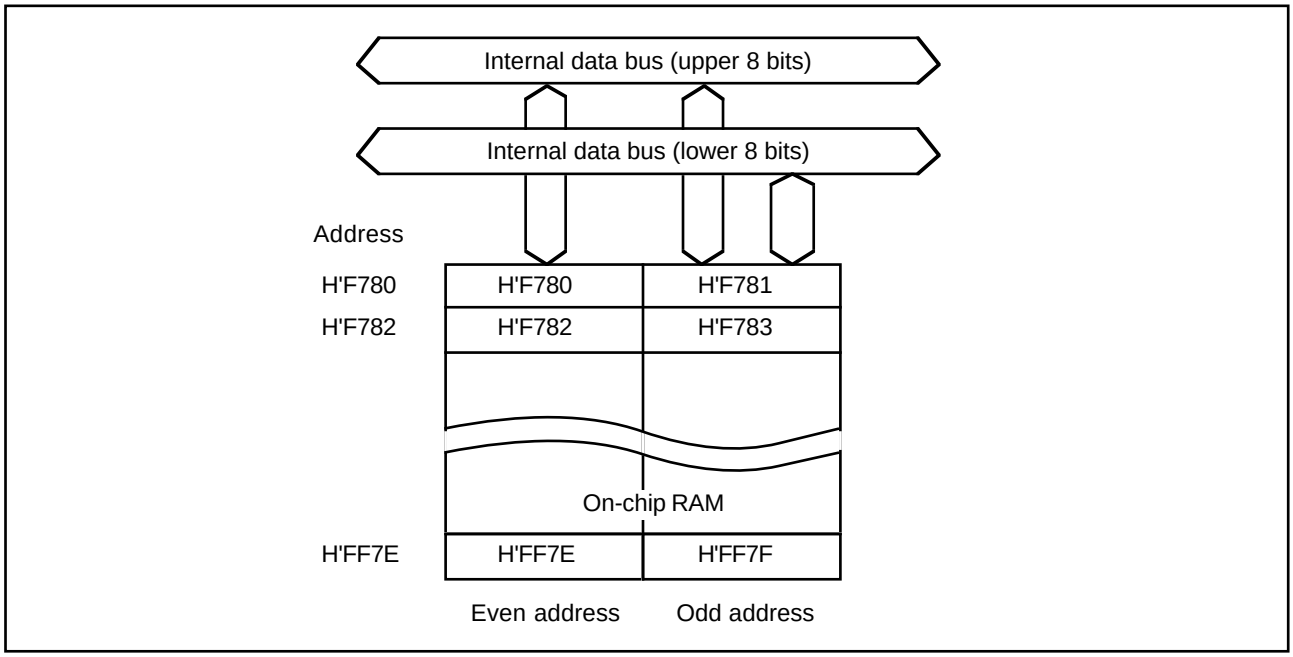



**Figure 10-1. Block Diagram of On-Chip RAM (H8/3257)**

## 10.3 RAM Enable Bit (RAME)

The on-chip RAM is enabled or disabled by the RAME (RAM Enable) bit in the system control register (SYSCR). Table 10-1 lists information about the system control register.

**Table 10-1. System Control Register**

| Name | Abbreviation | R/W | Initial value | Address |
|---|---|---|---|---|
| System control register | SYSCR | R/W | H'0B | H'FFC4 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSBY | STS2 | STS1 | STS0 | — | NMIEG | — | RAME |
| Initial value | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | — | R/W | — | R/W |

The only bit in the system control register that concerns the on-chip RAM is the RAME bit. See section 2.4.2, System Control Register for the other bits.

**Bit 0 – RAM Enable (RAME):** This bit enables or disables the on-chip RAM.

The RAME bit is initialized to 1 on the rising edge of the $\overline{\text{RES}}$ signal, so a reset enables the on-chip RAM. The RAME bit is not initialized in the software standby mode.

**Bit 7**

| RAME | Description | |
|---|---|---|
| 0 | On-chip RAM is disabled. | |
| 1 | On-chip RAM is enabled. | (Initial value) |

## 10.4 Operation

### 10.4.1 Expanded Modes (Modes 1 and 2)

If the RAME bit is set to 1, accesses to the following addresses are directed to the on-chip RAM.
H8/3257, H8/3256: H'F780 to H'FF7F
H8/325, H8/324: H'FB80 to H'FF7F
H8/323: H'FD80 to H'FF7F
H8/322: H'FE80 to H'FF7F
If the RAME bit is cleared to 0, accesses to these addresses are directed to the external data bus.

### 10.4.2 Single-Chip Mode (Mode 3)

If the RAME bit is set to 1, accesses to the following addresses are directed to the on-chip RAM.
H8/3257, H8/3256: H'F780 to H'FF7F
H8/325, H8/324: H'FB80 to H'FF7F
H8/323: H'FD80 to H'FF7F
H8/322: H'FE80 to H'FF7F
If the RAME bit is cleared to 0, the on-chip RAM data cannot be accessed. Attempted write access has no effect. Attempted read access always results in H’FF data being read.

# Section 11. ROM

## 11.1 Overview

The H8/3257 has 60 Kbytes of high-speed, on-chip ROM. The H8/3256 has 48 Kbytes. The H8/325 has 32 Kbytes. The H8/324 has 24 Kbytes. The H8/323 has 16 Kbytes. The H8/322 has 8 Kbytes. The on-chip ROM is connected to the CPU via a 16-bit data bus. Both byte data and word data are accessed in two states, enabling rapid data transfer and instruction fetching.

The H8/3257, H8/3256, H8/325, H8/323, and H8/322 are available in two versions: one with electrically programmable ROM (PROM); the other with masked ROM. The PROM version has a PROM mode in which the chip can be programmed with a standard PROM writer.

The on-chip ROM is enabled or disabled depending on the MCU operating mode, which is determined by the inputs at the mode pins ($MD_1$ and $MD_0$) when the chip comes out of the reset state. See table 11-1.

**Table 11-1. On-Chip ROM Usage in Each MCU Mode**

| | **Mode pins** | | |
|---|---|---|---|
| **Mode** | **$MD_1$** | **$MD_0$** | **On-chip ROM** |
| Mode 1 (expanded mode) | 0 | 1 | Disabled (external addresses) |
| Mode 2 (expanded mode) | 1 | 0 | Enabled |
| Mode 3 (single-chip mode) | 1 | 1 | Enabled |

### 11.1.1 Block Diagram

Figure 11-1 is a block diagram of the on-chip ROM.

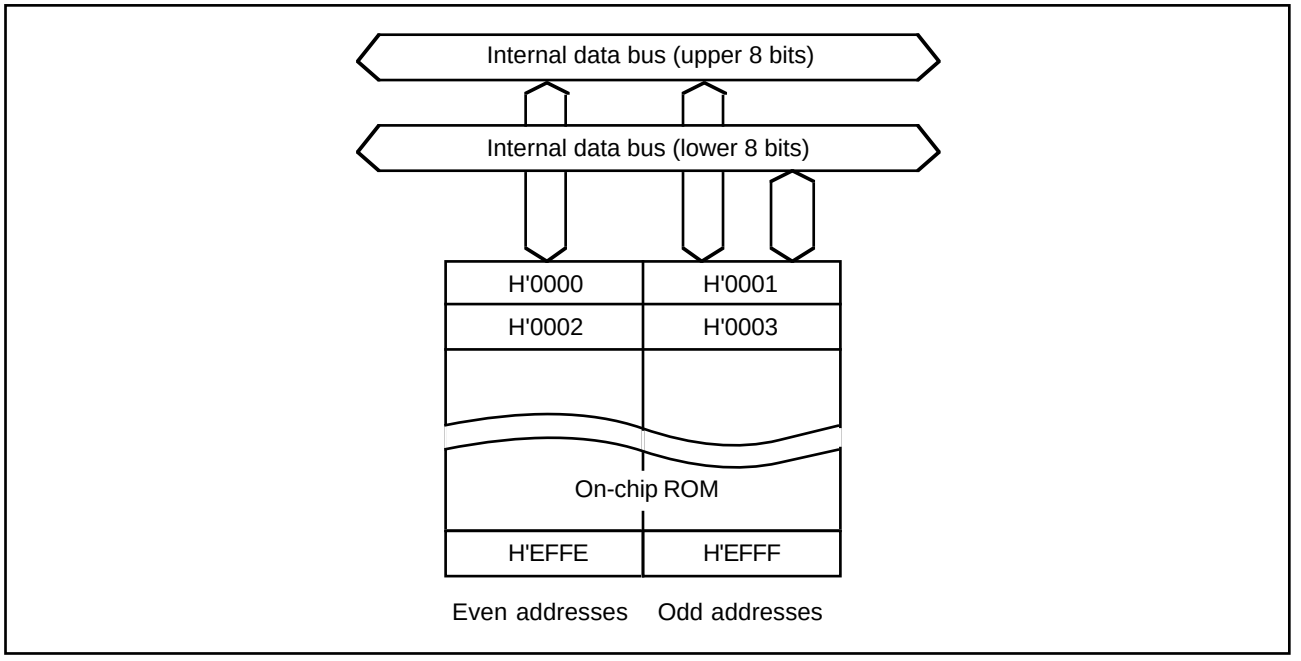



**Figure 11-1. Block Diagram of On-Chip ROM (H8/3257)**

## 11.2 PROM Mode

### 11.2.1 PROM Mode Setup

In the PROM mode of the PROM version of the H8/3257 and H8/3256, the usual microcomputer functions are halted to allow the on-chip PROM to be programmed. The programming method is the same as for the HN27C101. In the PROM mode of the PROM version of the H8/325, H8/323, and H8/322 the usual microcomputer functions are halted to allow the on-chip PROM to be programmed. The programming method is the same as for the HN27C256.

To select the PROM mode, apply the signal inputs listed in table 11-2.

**Table 11-2. Selection of PROM Mode**

| Pin | Input |
|---|---|
| Mode pin $MD_1$ | Low |
| Mode pin $MD_0$ | Low |
| $\overline{STBY}$ pin | Low |
| Pins $P7_0$ and $P7_1$ | High |

### 11.2.2 Socket Adapter Pin Assignments and Memory Map

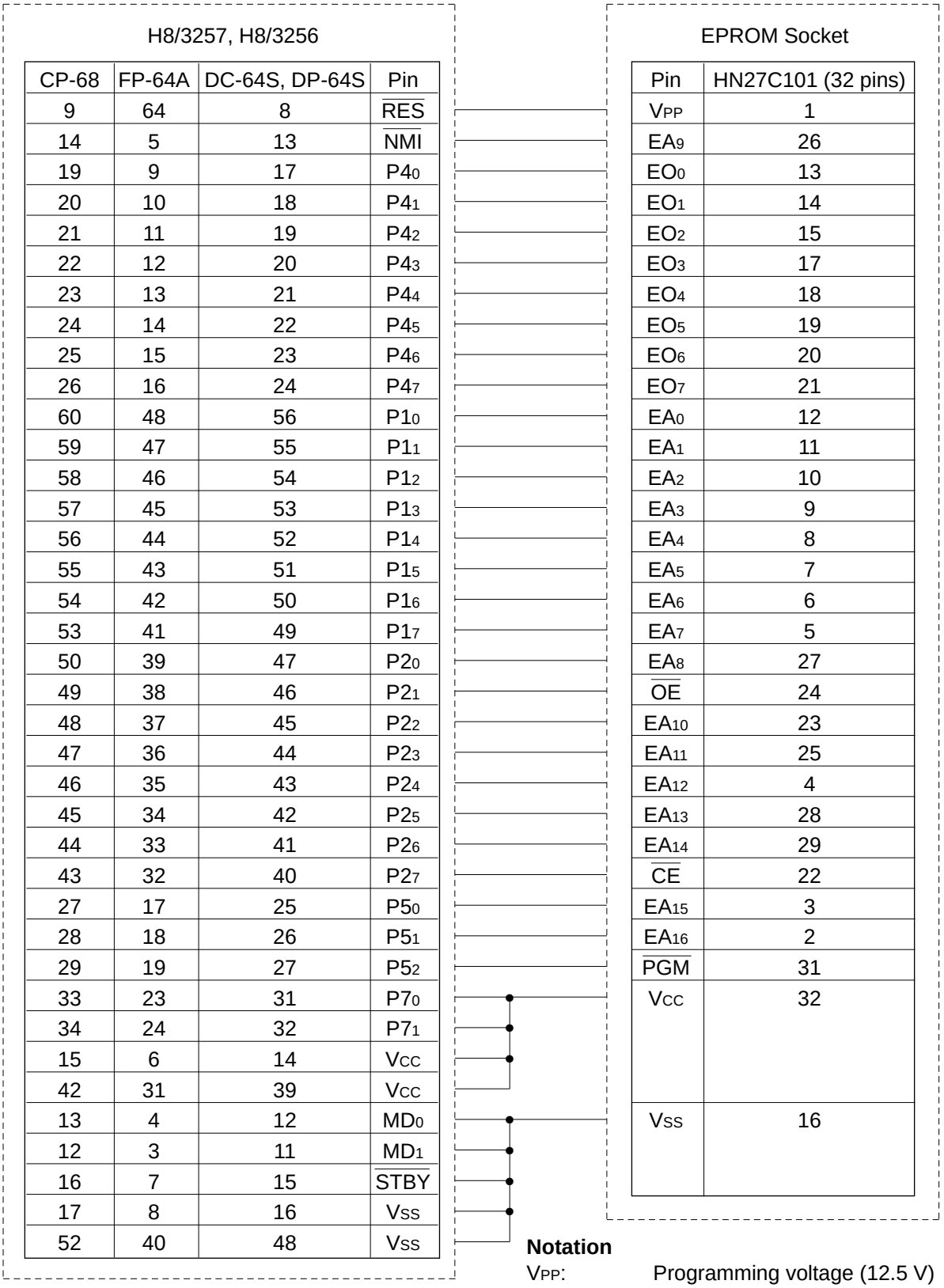
The H8/3257, H8/3256, H8/325, H8/323, and H8/322 can be programmed with a general-purpose PROM writer. Since the microcontroller package has 64 pins instead of 28 or 32 pins, a socket adapter is necessary. Table 11-3 lists recommended socket adapters. Figures 11-2 and 11-3 show the socket adapter pin assignments by giving the correspondence between microcontroller pins and HN27C101 or HN27C256 pin functions.

Figures 11-4 to 11-8 show memory maps in PROM mode. Since the H8/3257 has 60 Kbytes of on-chip PROM, the address range should be specified as H’0000 to H’EFFF. H’FF data should be specified for unused address areas.

The H8/3256 has only 48 Kbytes of PROM. The H8/325 has only 32 Kbytes. The H8/323 has only 16 Kbytes. The H8/322 has only 8 Kbytes. When programming these microcontrollers with a PROM writer, specify an address range of H’0000 to H’BFFF for the H8/3256, H’0000 to H’7FFF for the H8/325, H’0000 to H’3FFF for the H8/323, or H’0000 to H’1FFF for the H8/322. Specify H’FF data for addresses equal to or greater than H’C000 (H8/3256), H’8000 (H8/325), H’4000 (H8/323) or H’2000 (H8/322). Also specify H’FF data for unused address areas. If these areas are programmed by mistake, it may become impossible to write or verify PROM data. Be particularly careful with microcontrollers in plastic packages, in which the PROM cannot be reprogrammed.

**Table 11-3. Recommended Socket Adapters**

| Type | Package | Recommended socket adapter |
|---|---|---|
| H8/3257 | 64-Pin windowed shrink DIP (DC-64S) | HS3257ESS01H |
| H8/3256 | 64-Pin shrink DIP (DP-64S) | |
| | 64-Pin QFP (FP-64A) | HS3257ESH01H |
| | 68-Pin PLCC (CP-68) | HS3257ESC01H |
| H8/325 | 64-Pin windowed shrink DIP (DC-64S) | HS328ESS01H |
| H8/323 | 64-Pin shrink DIP (DP-64S) | |
| H8/322 | 64-Pin QFP (FP-64A) | HS328ESH01H |
| | 68-Pin PLCC (CP-68) | HS328ESC01H |




| CP-68 | FP-64A | DC-64S, DP-64S | Pin |
|---|---|---|---|
| 9 | 64 | 8 | $\overline{RES}$ |
| 14 | 5 | 13 | $\overline{NMI}$ |
| 19 | 9 | 17 | $P4_0$ |
| 20 | 10 | 18 | $P4_1$ |
| 21 | 11 | 19 | $P4_2$ |
| 22 | 12 | 20 | $P4_3$ |
| 23 | 13 | 21 | $P4_4$ |
| 24 | 14 | 22 | $P4_5$ |
| 25 | 15 | 23 | $P4_6$ |
| 26 | 16 | 24 | $P4_7$ |
| 60 | 48 | 56 | $P1_0$ |
| 59 | 47 | 55 | $P1_1$ |
| 58 | 46 | 54 | $P1_2$ |
| 57 | 45 | 53 | $P1_3$ |
| 56 | 44 | 52 | $P1_4$ |
| 55 | 43 | 51 | $P1_5$ |
| 54 | 42 | 50 | $P1_6$ |
| 53 | 41 | 49 | $P1_7$ |
| 50 | 39 | 47 | $P2_0$ |
| 49 | 38 | 46 | $P2_1$ |
| 48 | 37 | 45 | $P2_2$ |
| 47 | 36 | 44 | $P2_3$ |
| 46 | 35 | 43 | $P2_4$ |
| 45 | 34 | 42 | $P2_5$ |
| 44 | 33 | 41 | $P2_6$ |
| 43 | 32 | 40 | $P2_7$ |
| 27 | 17 | 25 | $P5_0$ |
| 28 | 18 | 26 | $P5_1$ |
| 29 | 19 | 27 | $P5_2$ |
| 33 | 23 | 31 | $P7_0$ |
| 34 | 24 | 32 | $P7_1$ |
| 15 | 6 | 14 | $V_{CC}$ |
| 42 | 31 | 39 | $V_{CC}$ |
| 13 | 4 | 12 | $MD_0$ |
| 12 | 3 | 11 | $MD_1$ |
| 16 | 7 | 15 | $\overline{STBY}$ |
| 17 | 8 | 16 | $V_{SS}$ |
| 52 | 40 | 48 | $V_{SS}$ |

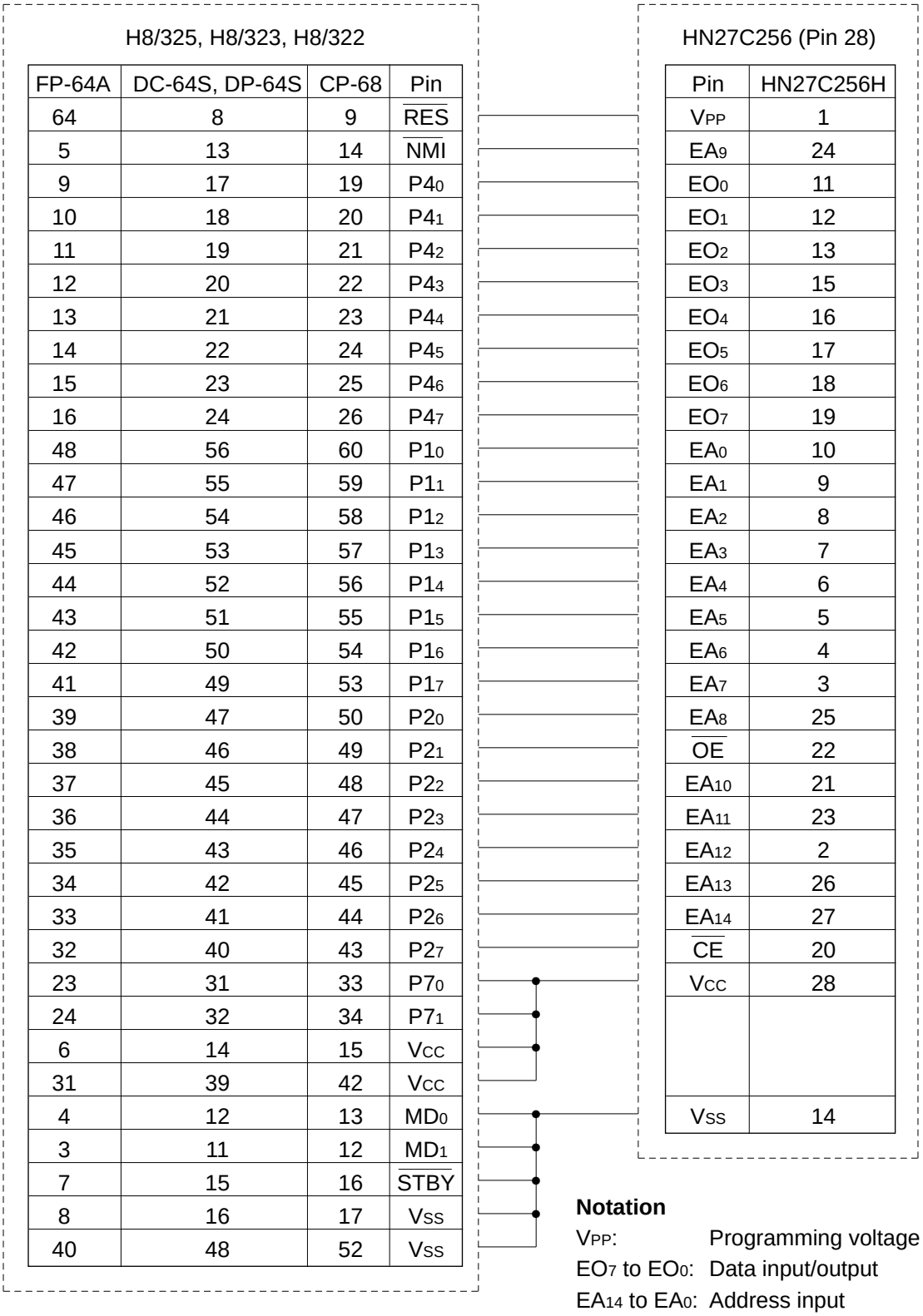
| Pin | HN27C101 (32 pins) |
|---|---|
| $V_{PP}$ | 1 |
| $EA_9$ | 26 |
| $EO_0$ | 13 |
| $EO_1$ | 14 |
| $EO_2$ | 15 |
| $EO_3$ | 17 |
| $EO_4$ | 18 |
| $EO_5$ | 19 |
| $EO_6$ | 20 |
| $EO_7$ | 21 |
| $EA_0$ | 12 |
| $EA_1$ | 11 |
| $EA_2$ | 10 |
| $EA_3$ | 9 |
| $EA_4$ | 8 |
| $EA_5$ | 7 |
| $EA_6$ | 6 |
| $EA_7$ | 5 |
| $EA_8$ | 27 |
| $\overline{OE}$ | 24 |
| $EA_{10}$ | 23 |
| $EA_{11}$ | 25 |
| $EA_{12}$ | 4 |
| $EA_{13}$ | 28 |
| $EA_{14}$ | 29 |
| $\overline{CE}$ | 22 |
| $EA_{15}$ | 3 |
| $EA_{16}$ | 2 |
| $\overline{PGM}$ | 31 |
| $V_{CC}$ | 32 |
| $V_{SS}$ | 16 |

**Notation**

| | |
|---|---|
| $V_{PP}$: | Programming voltage (12.5 V) |
| $EO_7$ to $EO_0$: | Data input/output |
| $EA_{16}$ to $EA_0$: | Address input |
| $\overline{OE}$: | Output enable |
| $\overline{CE}$: | Chip enable |
| $\overline{PGM}$: | Program enable |

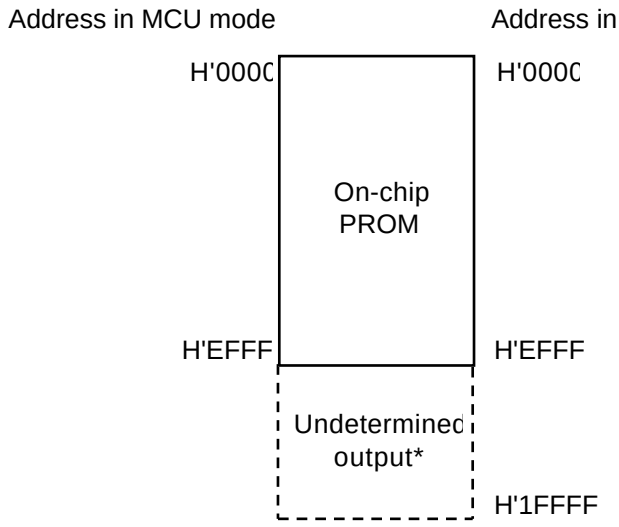
**Note:** All pins not listed in this figure should be left open.

**Figure 11-2. Socket Adapter Pin Assignments**



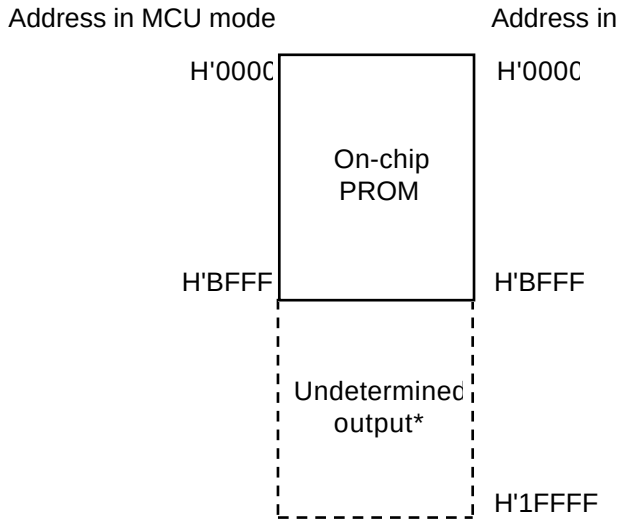

**Note:** All pins not listed in this figure should be left open.

**Figure 11-3. Socket Adapter Pin Assignments**




Note: If this address area is read in PROM mode, the output data are undetermined.

**Figure 11-4. H8/3257 Memory Map in PROM Mode**




Note: If this address area is read in PROM mode, the output data are undetermined.

**Figure 11-5. H8/3256 Memory Map in PROM Mode**

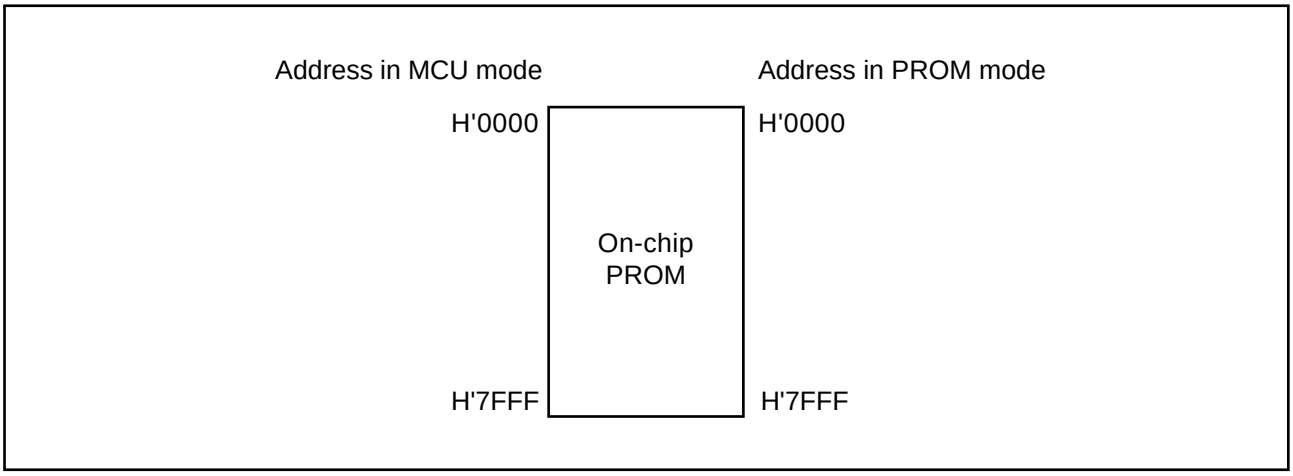



**Figure 11-6. Memory Map of the H8/325 in PROM Mode**

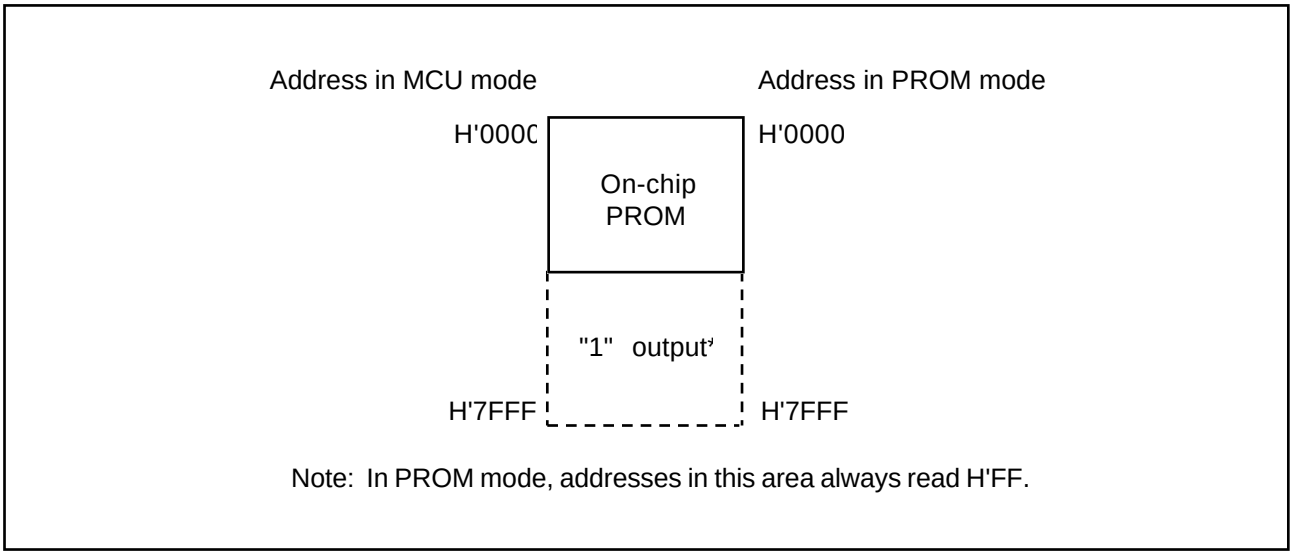



**Figure 11-7. Memory Map of the H8/323 in PROM Mode**

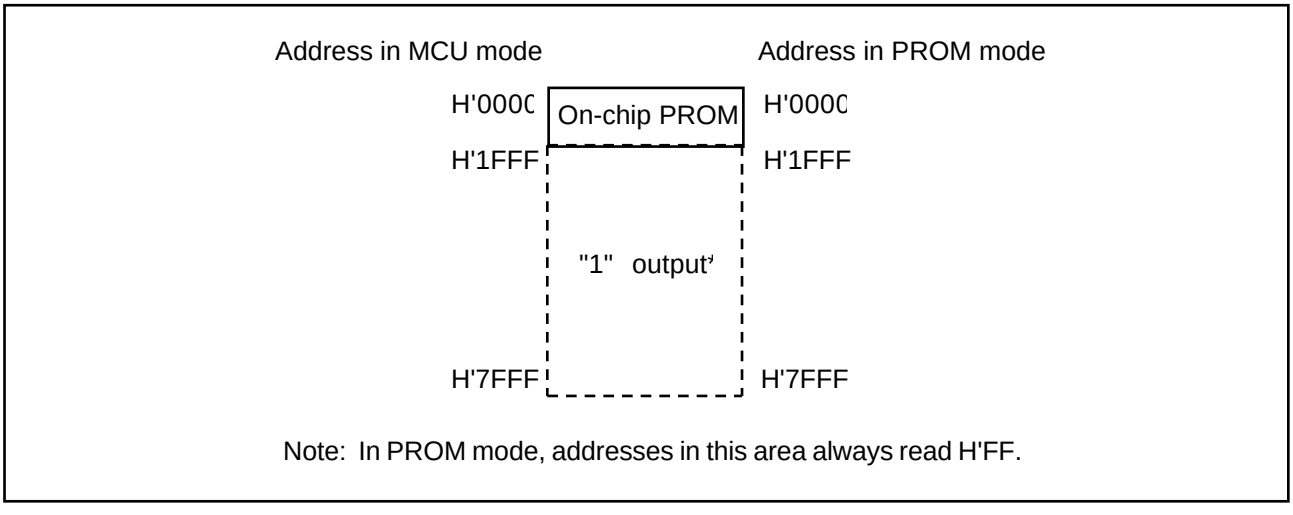



**Figure 11-8. Memory Map of the H8/322 in PROM Mode**

## 11.3 Programming

### 11.3.1 Selection of Sub-Modes in PROM Mode

#### (1) Case of H8/3257 and H8/3256

The write, verify, and other sub-modes of the PROM mode are selected as shown in table 11-4.

**Table 11-4. Selection of Sub-Modes in PROM Mode**

| | Pins | | | | | | |
|---|---|---|---|---|---|---|---|
| **Sub-mode** | **$\overline{CE}$** | **$\overline{OE}$** | **$\overline{PGM}$** | **$V_{PP}$** | **$V_{CC}$** | **$E0_7$ to $E0_0$** | **$EA_{16}$ to $EA_0$** |
| Write | Low | High | Low | $V_{PP}$ | $V_{CC}$ | Data input | Address input |
| Verify | Low | Low | High | $V_{PP}$ | $V_{CC}$ | Data output | Address input |
| Programming inhibited | Low | Low | Low | $V_{PP}$ | $V_{CC}$ | High-impedance | Address input |
| | Low | High | High | | | | |
| | High | Low | Low | | | | |
| | High | High | High | | | | |

**Note:** The $V_{PP}$ and $V_{CC}$ pins must be held at the $V_{PP}$ and $V_{CC}$ voltage levels.

The H8/3257 or H8/3256 PROM has the same standard read/write specifications as the HN27C101 EPROM. Page programming is not supported, however, so do not select page programming mode. PROM writers that provide only page programming cannot be used. When selecting a PROM writer, check that it supports the byte-at-a-time high-speed programming mode. Be sure to set the address range to H'0000 to H'EFFF for the H8/3257, and to H'0000 to H'BFFF for the H8/3256.

**(2) Case of H8/325, H8/323, and H8/322**

The write, verify, inhibited, and read sub-modes of the PROM mode are selected as shown in table 11-5.

**Table 11-5. Selection of Sub-Modes in PROM Mode**

| | Pins | | | | | |
|---|---|---|---|---|---|---|
| **Mode** | **$\overline{CE}$** | **$\overline{OE}$** | **$V_{PP}$** | **$V_{CC}$** | **$E0_7$ to $E0_0$** | **$EA_{14}$ to $EA_0$** |
| Write | Low | High | $V_{PP}$ | $V_{CC}$ | Data input | Address input |
| Verify | High | Low | $V_{PP}$ | $V_{CC}$ | Data output | Address input |
| Programming inhibited | High | High | $V_{PP}$ | $V_{CC}$ | High-impedance | Address input |

**Note:** The $V_{PP}$ and $V_{CC}$ pins must be held at the $V_{PP}$ and $V_{CC}$ voltage levels.

The H8/325 PROM uses the same, standard read/write specifications as the HN27C256 and HN27256.

### 11.3.2 Writing and Verifying

An efficient, high-speed programming procedure can be used to write and verify PROM data. This procedure writes data quickly without subjecting the chip to voltage stress and without sacrificing data reliability. It leaves the data H'FF written in unused addresses.

Figures 11-9 to 11-10 show the basic high-speed programming flowchart.

Tables 11-6 and 11-8 list the electrical characteristics of the chip in the PROM mode. Figure 11-11 shows a write/verify timing chart.

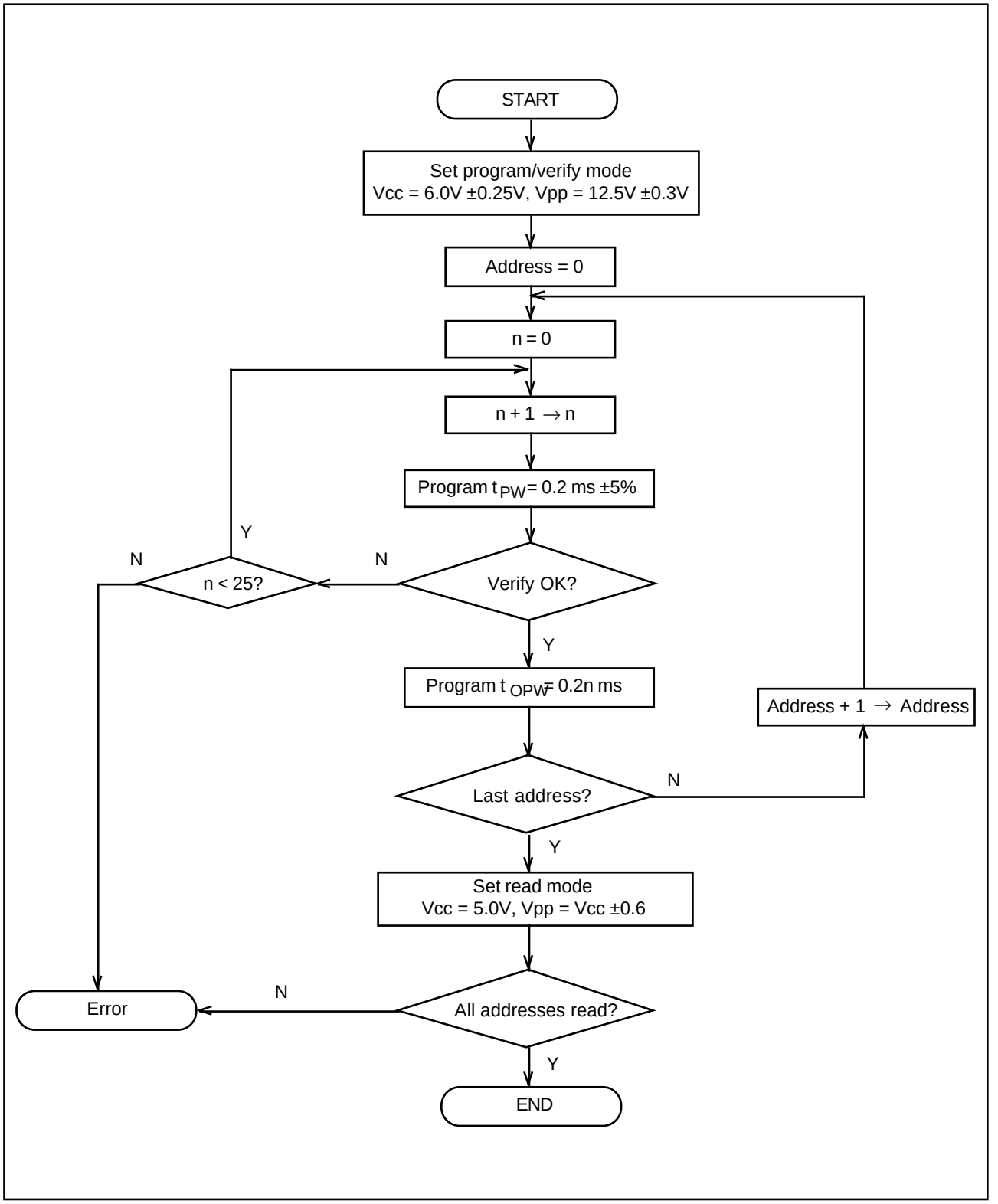



**Figure 11-9. High-Speed Programming Flowchart (H8/3257, H8/3256)**

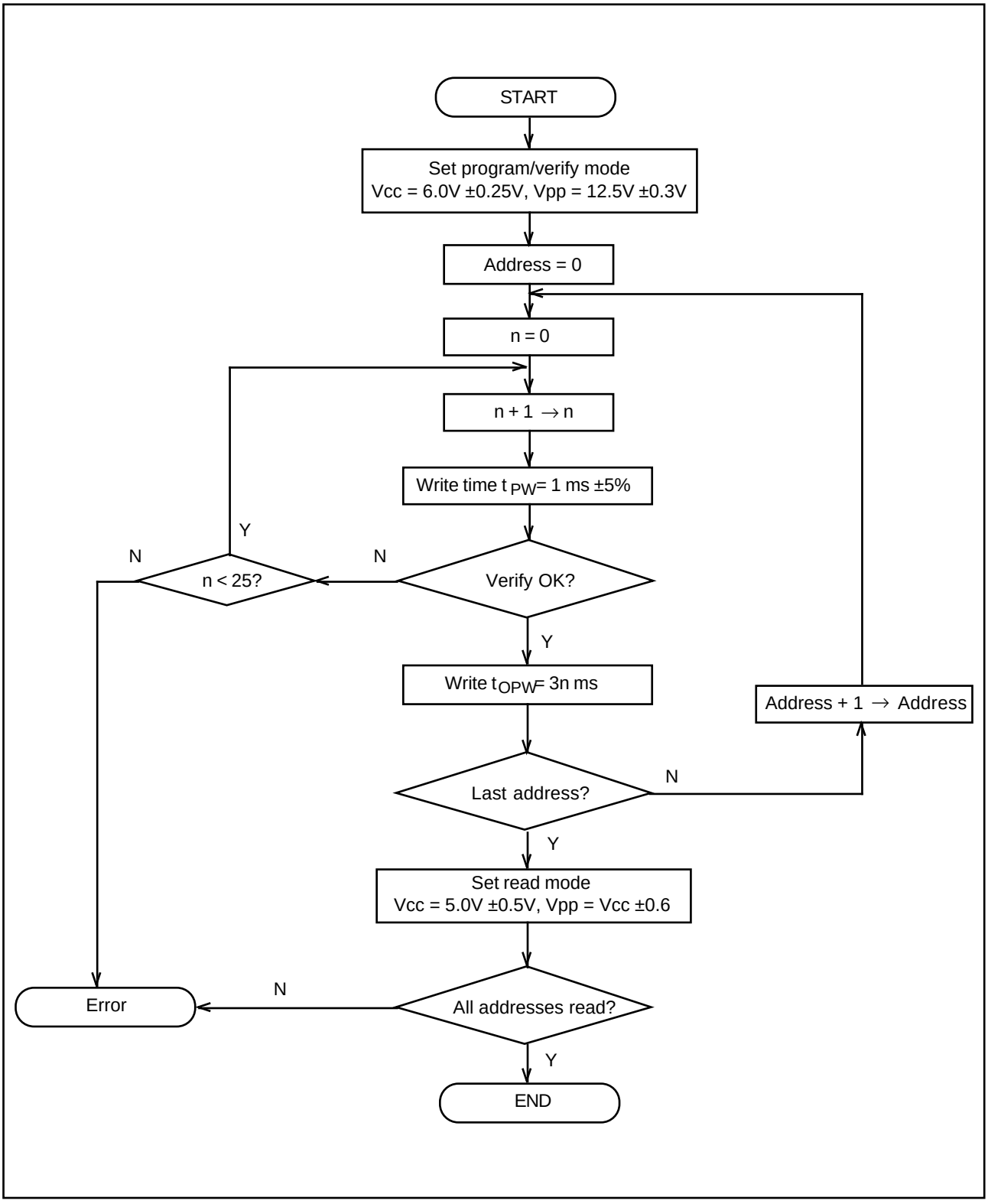



**Figure 11-10. High-Speed Programming Flowchart (H8/325, H8/323, H8/322)**

**Table 11-6. DC Characteristics**
**(When $V_{CC}$ = 6.0V ±0.25V, $V_{PP}$ = 12.5V ±0.3V, $V_{SS}$ = 0V, Ta = 25˚C ±5˚C)**

| Item | | Symbol | min | typ | max | Unit | Measurement conditions |
|---|---|---|---|---|---|---|---|
| Input high voltage | $EO_7$ – $EO_0$, $EA_{14}$ – $EA_0$, $\overline{OE}$, $\overline{CE}$ | $V_{IH}$ | 2.4 | — | $V_{CC}$ + 0.3 | V | |
| Input low voltage | $EO_7$ – $EO_0$, EA14 – $EA_0$, $\overline{OE}$, $\overline{CE}$ | $V_{IL}$ | – 0.3 | — | 0.8 | V | |
| Output high voltage | $EO_7$ – $EO_0$ | $V_{OH}$ | 2.4 | — | — | V | $I_{OH}$ = –200 µA |
| Output low voltage | $EO_7$ – $EO_0$ | $V_{OL}$ | — | — | 0.45 | V | $I_{OL}$ = 1.6 mA |
| Input leakage current | $EO_7$ – $EO_0$, $EA_{14}$ – $EA_0$, $\overline{OE}$, $\overline{CE}$ | \|$I_{LI}$\| | — | — | 2 | µA | $V_{in}$ = 5.25V/ 0.5V |
| $V_{CC}$ current | | $I_{CC}$ | — | — | 40 | mA | |
| $V_{PP}$ current | | $I_{PP}$ | — | — | 40 | mA | |

**Table 11-7. AC Characteristics (H8/3257, H8/3256)**
**(When $V_{CC}$ = 6.0V ±0.25V, $V_{PP}$ = 12.5V ±0.3V, Ta = 25˚C ±5˚C)**

| Item | Symbol | min | typ | max | Unit | Measurement conditions |
|---|---|---|---|---|---|---|
| Address setup time | $t_{AS}$ | 2 | — | — | µs | See Figure 11-11* |
| $\overline{OE}$ setup time | $t_{OES}$ | 2 | — | — | µs | |
| Data setup time | $t_{DS}$ | 2 | — | — | µs | |
| Address hold time | $t_{AH}$ | 0 | — | — | µs | |
| Data hold time | $t_{DH}$ | 2 | — | — | µs | |
| Data output disable time | $t_{DF}$ | — | — | 130 | ns | |
| Vpp setup time | $t_{VPS}$ | 2 | — | — | µs | |
| Program pulse width | $t_{PW}$ | 0.19 | 0.20 | 0.21 | ms | |
| $\overline{OE}$ pulse width for overwrite-programming | $t_{OPW}$ | 0.19 | — | 5.25 | ms | |
| $V_{CC}$ setup time | $t_{VCS}$ | 2 | — | — | µs | |
| $\overline{CE}$ setup time | $t_{CES}$ | 2 | — | — | µs | |
| Data output delay time | $t_{OE}$ | 0 | — | 150 | ns | |

* Input pulse level: 0.8V to 2.2V
Input rise/fall time ≦ 20 ns
Timing reference levels: input—1.0V, 2.0V; output—0.8V, 2.0V

**Table 11-8. AC Characteristics (H8/325, H8/323, H8/322)**
**(When $V_{CC}$ = 6.0V ±0.25V, $V_{PP}$ = 12.5V ±0.3V, Ta = 25˚C ±5˚C)**

| Item | Symbol | min | typ | max | Unit | Measurement conditions |
|---|---|---|---|---|---|---|
| Address setup time | $t_{AS}$ | 2 | — | — | µs | See Figure 11-11* |
| $\overline{OE}$ setup time | $t_{OES}$ | 2 | — | — | µs | |
| Data setup time | $t_{DS}$ | 2 | — | — | µs | |
| Address hold time | $t_{AH}$ | 0 | — | — | µs | |
| Data hold time | $t_{DH}$ | 2 | — | — | µs | |
| Data output disable time | $t_{DF}$ | — | — | 130 | ns | |
| Vpp setup time | $t_{VPS}$ | 2 | — | — | µs | |
| Program pulse width | $t_{PW}$ | 0.95 | 1.0 | 1.05 | ms | |
| $\overline{OE}$ pulse width for overwrite-programming | $t_{OPW}$ | 2.85 | — | 78.75 | ms | |
| Vcc setup time | $t_{VCS}$ | 2 | — | — | µs | |
| Data output delay time | $t_{OE}$ | 0 | — | 500 | ns | |

* Input pulse level: 0.8V to 2.2V
Input rise/fall time ≦ 20 ns
Timing reference levels: input—1.0V, 2.0V; output—0.8V, 2.0V

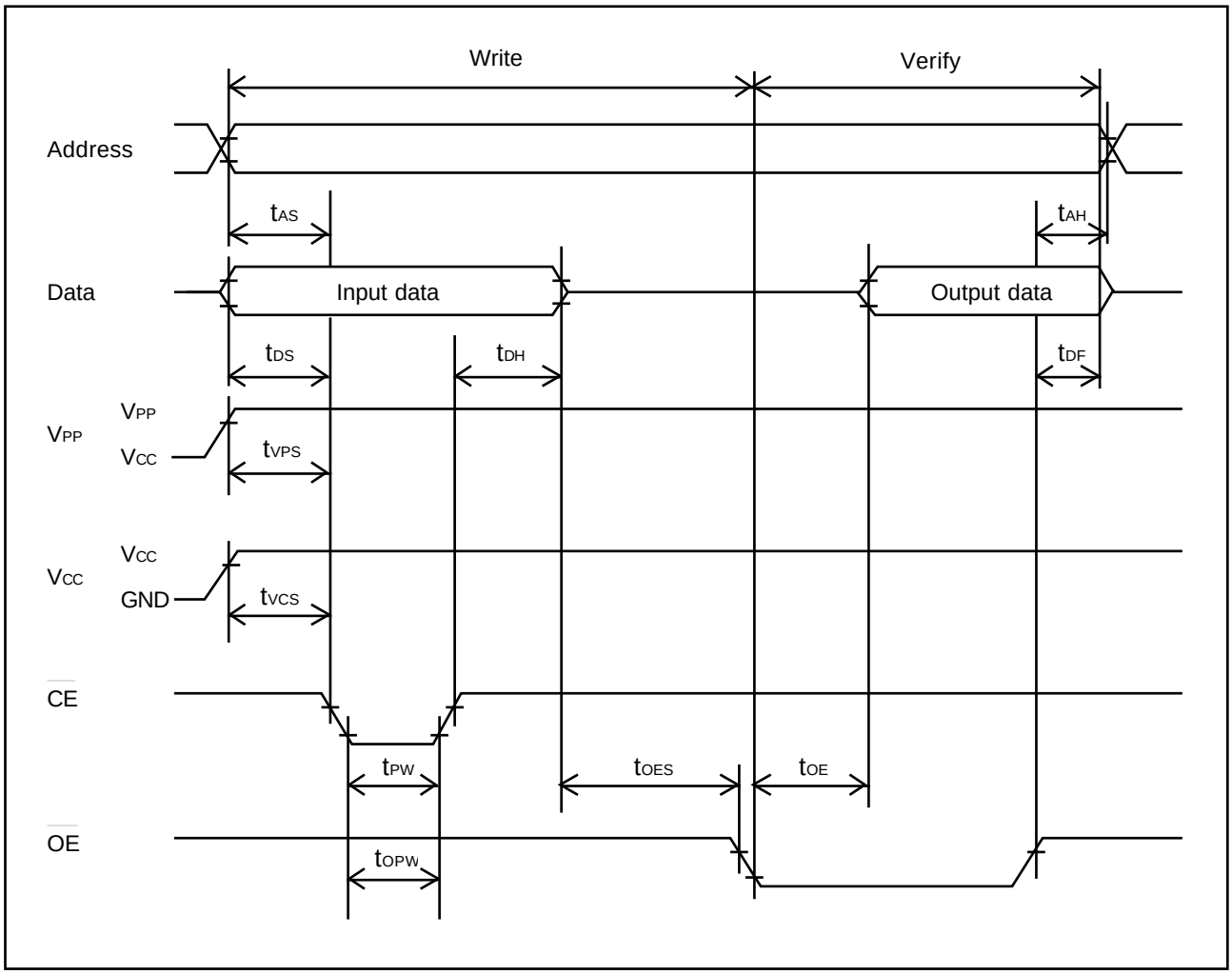



**Figure 11-11. PROM Write/Verify Timing**

### 11.3.3 Notes on Writing

**(1) Write with the specified voltages and timing. The programming voltage ($V_{PP}$) is 12.5 V.**

**Caution:** Applied voltages in excess of the specified values can permanently destroy the chip. Be particularly careful about the PROM writer's overshoot characteristics.

If the PROM writer is set to Intel specifications or Hitachi HN27C101, HN27256 or HN27C256 specifications, $V_{PP}$ will be 12.5 V.

**(2) Before writing data, check that the socket adapter and chip are correctly mounted in the PROM writer.** Overcurrent damage to the chip can result if the index marks on the PROM writer, socket adapter, and chip are not correctly aligned.

**(3) Don't touch the socket adapter or chip while writing.** Touching either of these can cause contact faults and write errors.

**(4) Page programming is not supported.** Do not select page programming mode.

### 11.3.4 Reliability of Written Data

An effective way to assure the data holding characteristics of the programmed chips is to bake them at 150˚C, then screen them for data errors. This procedure quickly eliminates chips with PROM memory cells prone to early failure.

Figure 11-12 shows the recommended screening procedure.

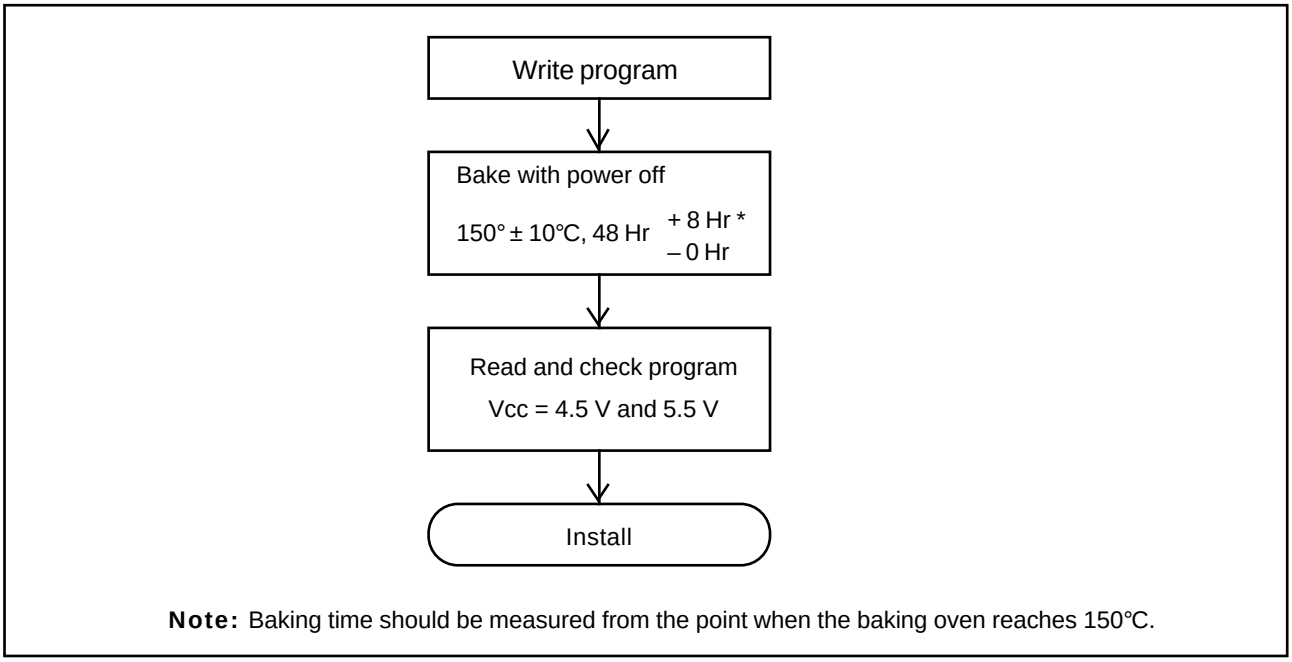



**Figure 11-12. Recommended Screening Procedure**

If a series of write errors occurs while the same PROM writer is in use, stop programming and check the PROM writer and socket adapter for defects, using a microcomputer chip with a windowed package and on-chip EPROM.

Please inform Hitachi of any abnormal conditions noted during programming or in screening of program data after high-temperature baking.

### 11.3.5 Erasing of Data

The windowed package enables data to be erased by illuminating the window with ultraviolet light. Table 11-9 lists the erasing conditions.

**Table 11-9. Erasing Conditions**

| Item | Value |
|---|---|
| Ultraviolet wavelength | 253.7 nm |
| Minimum illumination | 15W·s/cm$^2$ |

The conditions in table 11-9 can be satisfied by placing a 12000-µW/cm$^2$ ultraviolet lamp 2 or 3 centimeters directly above the chip and leaving it on for about 20 minutes.

## 11.4 Handling of Windowed Packages

**(1) Glass Erasing Window:** Rubbing the glass erasing window of a windowed package with a plastic material or touching it with an electrically charged object can create a static charge on the window surface which may cause the chip to malfunction.

If the erasing window becomes charged, the charge can be neutralized by a short exposure to ultraviolet light. This returns the chip to its normal condition, but it also reduces the charge stored in the floating gates of the PROM, so it is recommended that the chip be reprogrammed afterward.

Accumulation of static charge on the window surface can be prevented by the following precautions:

① When handling the package, ground yourself. Don’t wear gloves. Avoid other possible sources of static charge.
② Avoid friction between the glass window and plastic or other materials that tend to accumulate static charge.

③ Be careful when using cooling sprays, since they may have a slight ion content.

④ Cover the window with an ultraviolet-shield label, preferably a label including a conductive material. Besides protecting the PROM contents from ultraviolet light, the label protects the chip by distributing static charge uniformly.

**(2) Handling after Programming:** Fluorescent light and sunlight contain small amounts of ultraviolet, so prolonged exposure to these types of light can cause programmed data to invert. In addition, exposure to any type of intense light can induce photoelectric effects that may lead to chip malfunction. It is recommended that after programming the chip, you cover the erasing window with a light-proof label (such as an ultraviolet-shield label).

# Section 12. Power-Down State

## 12.1 Overview

The H8/325 series has a power-down state that greatly reduces power consumption by stopping some or all of the chip functions. The power-down state includes three modes:

(1) Sleep mode – a software-triggered mode in which the CPU halts but the rest of the chip remains active
(2) Software standby mode – a software-triggered mode in which the entire chip is inactive
(3) Hardware standby mode – a hardware-triggered mode in which the entire chip is inactive

Table 12-1 lists the conditions for entering and leaving the power-down modes. It also indicates the status of the CPU, on-chip supporting modules, etc. in each power-down mode.

**Table 12-1. Power-Down State**

| Mode | Entering procedure | Clock | CPU | CPU Reg's. | Sup. Mod.* | RAM | I/O ports | Exiting methods |
|---|---|---|---|---|---|---|---|---|
| Sleep mode | Execute SLEEP instruction | Run | Halt | Held | Run | Held | Held | • Interrupt<br>• $\overline{\text{RES}}$<br>• $\overline{\text{STBY}}$ |
| Soft-ware standby mode | Set SSBY bit in SYSCR to 1, then execute SLEEP instruction | Halt | Halt | Held | Halt and initial-ized | Held | Held | • $\overline{\text{NMI}}$<br>• $\overline{\text{IRQ}_0}$ – $\overline{\text{IRQ}_2}$<br>• $\overline{\text{STBY}}$<br>• $\overline{\text{RES}}$<br>• $\overline{\text{IS}}$ |
| Hard-ware standby mode | Set $\overline{\text{STBY}}$ pin to low level | Halt | Halt | Not held | Halt and initialized | Held | High impe-dance state | • $\overline{\text{STBY}}$ high, then $\overline{\text{RES}}$ low → high |

* On-chip supporting modules.

**Notes**

1. SYSCR: System control register
2. SSBY: Software standby bit

## 12.2 System Control Register: Power-Down Control Bits

Bits 7 to 4 of the system control register (SYSCR) concern the power-down state. Specifically, they concern the software standby mode.

Table 12-2 lists the attributes of the system control register.

**Table 12-2. System Control Register**

| Name | Abbreviation | R/W | Initial value | Address |
|---|---|---|---|---|
| System control register | SYSCR | R/W | H'0B | H'FFC4 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSBY | STS2 | STS1 | STS0 | — | NMIEG | — | RAME |
| Initial value | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | — | R/W | — | R/W |

**Bit 7 – Software Standby (SSBY):** This bit enables or disables the transition to the software standby mode.

On recovery from the software standby mode by an external interrupt or input strobe interrupt, SSBY remains set to 1. To clear this bit, software must write a 0.

| Bit 7 SSBY | Description | |
|---|---|---|
| 0 | The SLEEP instruction causes a transition to the sleep mode. | (Initial value) |
| 1 | The SLEEP instruction causes a transition to the software standby mode. | |

**Bits 6 to 4 – Standby Timer Select 2 to 0 (STS2 to STS0):** These bits select the clock settling time when the chip recovers from the software standby mode by an external interrupt. During the selected time, the clock oscillator runs but clock pulses are not supplied to the CPU or the on-chip supporting modules.

| Bit 6 STS2 | Bit 5 STS1 | Bit 4 STS0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | Settling time = 8192 states | (Initial value) |
| 0 | 0 | 1 | Settling time = 16384 states | |
| 0 | 1 | 0 | Settling time = 32768 states | |
| 0 | 1 | 1 | Settling time = 65536 states | |
| 1 | — | — | Settling time = 131072 states | |

When the on-chip clock generator is used, the STS bits should be set to allow a settling time of at least 10 ms. Table 12-3 lists the settling times selected by these bits at several clock frequencies and indicates the recommended settings.

When the chip is externally clocked, the STS bits can be set to any value. The minimum value (STS2 = STS1 = STS0 = 0) is recommended.

**Table 12-3. Times Set by Standby Timer Select Bits (Unit: ms)**

| STS2 | STS1 | STS0 | Settling time (states) | System clock frequency (MHz) 10 | 8 | 6 | 4 | 2 | 1 | 0.5 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 8192 | 0.8 | 1.0 | 1.4 | 2.0 | 4.1 | 8.2 | **16.4** |
| 0 | 0 | 1 | 16384 | 1.6 | 2.0 | 2.7 | 4.1 | 8.2 | **16.4** | 32.8 |
| 0 | 1 | 0 | 32768 | 3.3 | 4.1 | 5.5 | 8.2 | **16.4** | 32.8 | 65.5 |
| 0 | 1 | 1 | 65536 | 6.6 | 8.2 | **10.9** | **16.4** | 32.8 | 65.5 | 131.1 |
| 1 | — | — | 131072 | **13.1** | **16.4** | 21.8 | 32.8 | 65.5 | 131.1 | 262.1 |

**Notes:**
1. All times are in milliseconds.
2. Recommended values are printed in boldface.

## 12.3 Sleep Mode

The sleep mode provides an effective way to conserve power while the CPU is waiting for an external interrupt or an interrupt from an on-chip supporting module.

### 12.3.1 Transition to Sleep Mode

When the SSBY bit in the system control register is cleared to 0, execution of the SLEEP instruction causes a transition from the program execution state to the sleep mode. After executing the SLEEP instruction, the CPU halts, but the contents of its internal registers remain unchanged. The on-chip supporting modules continue to operate normally.

### 12.3.2 Exit from Sleep Mode

The chip wakes up from the sleep mode when it receives an internal or external interrupt request, or a low input at the $\overline{\text{RES}}$ or $\overline{\text{STBY}}$ pin.

**(1) Wake-Up by Interrupt:** An interrupt releases the sleep mode and starts the CPU's interrupt-handling sequence.

If an interrupt from an on-chip supporting module is disabled by the corresponding enable/disable bit in the module's control register, the interrupt cannot be requested, so it cannot wake the chip up. Similarly, the CPU cannot be awoken by an interrupt other than NMI if the I (interrupt mask) bit in the CCR (condition code register) is set when the SLEEP instruction is executed.

**(2) Wake-Up by $\overline{\text{RES}}$ pin:** When the $\overline{\text{RES}}$ pin goes low, the chip exits from the sleep mode to the reset state.

**(3) Wake-Up by $\overline{\text{STBY}}$ pin:** When the $\overline{\text{STBY}}$ pin goes low, the chip exits from the sleep mode to the hardware standby mode.

## 12.4 Software Standby Mode

In the software standby mode, the system clock stops and chip functions halt, including both CPU functions and the functions of the on-chip supporting modules. Power consumption is reduced to an extremely low level. The on-chip supporting modules and their registers are reset to their initial states, but as long as a minimum necessary voltage supply is maintained (at least 2V), the contents of the CPU registers and on-chip RAM remain unchanged. I/O ports also remain unchanged.

### 12.4.1 Transition to Software Standby Mode

To enter the software standby mode, set the standby bit (SSBY) in the system control register (SYSCR) to 1, then execute the SLEEP instruction.

### 12.4.2 Exit from Software Standby Mode

The chip can be brought out of the software standby mode by an input at one of seven pins: $\overline{\text{NMI}}$, $\overline{\text{IRQ}_0}$, $\overline{\text{IRQ}_1}$, $\overline{\text{IRQ}_2}$, $\overline{\text{IS}}$, $\overline{\text{RES}}$, or $\overline{\text{STBY}}$.

**(1) Recovery by External Interrupt:** When an $\overline{\text{NMI}}$, $\overline{\text{IRQ}_0}$, $\overline{\text{IRQ}_1}$, $\overline{\text{IRQ}_2}$, or input strobe (ISI) interrupt request signal is received, the clock oscillator begins operating. After the waiting time set in the system control register (bits STS2 to STS0), clock pulses are supplied to the CPU and on-chip supporting modules. The CPU executes the interrupt-handling sequence for the requested interrupt, then returns to the instruction after the SLEEP instruction. The SSBY bit is not cleared.

See Section 12.2, System Control Register: Power-Down Control Bits for information about the STS bits.

**(2) Recovery by $\overline{\text{RES}}$ Pin:** When the $\overline{\text{RES}}$ pin goes low, the clock oscillator starts. Next, when the $\overline{\text{RES}}$ pin goes high, the CPU begins executing the reset sequence. The SSBY bit is cleared to 0.

The $\overline{\text{RES}}$ pin must be held low long enough for the clock to stabilize.

**(3) Recovery by $\overline{\text{STBY}}$ Pin:** When the $\overline{\text{STBY}}$ pin goes low, the chip exits from the software standby mode to the hardware standby mode.

### 12.4.3 Sample Application of Software Standby Mode

In this example the chip enters the software standby mode when $\overline{\text{NMI}}$ goes low and exits when $\overline{\text{NMI}}$ goes high, as shown in figure 12-1.

The NMI edge bit (NMIEG) in the system control register is originally cleared to 0, selecting the falling edge. When $\overline{\text{NMI}}$ goes low, the $\overline{\text{NMI}}$ interrupt handling routine sets NMIEG to 1 (selecting the rising edge), sets SSBY to 1, then executes the SLEEP instruction. The chip enters the software standby mode. It recovers from the software standby mode on the next rising edge of $\overline{\text{NMI}}$.

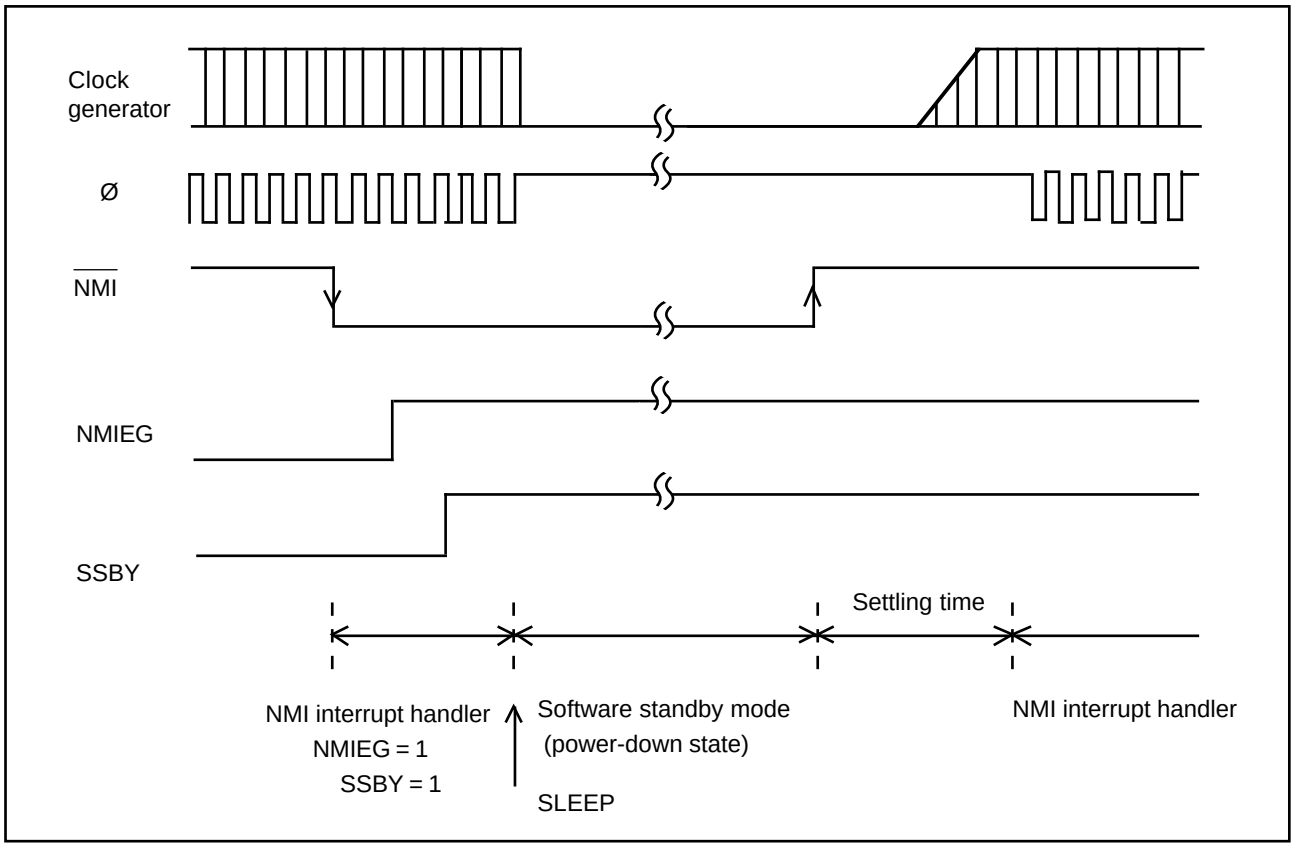



**Figure 12-1. Software Standby Mode NMI Timing (Example)**

### 12.4.4 Notes on Current Dissipation

1. The I/O ports remain in their current states in software standby mode. If a port is in the high output state, it continues to dissipate power in proportion to the output current.

2. When software standby mode is entered under condition (a) or (b) below, current dissipation is higher ($I_{CC}$ = 100 to 300 µA) than normal in standby mode.

   (a) In single-chip mode (mode 3): when software standby mode is entered by executing an instruction stored in on-chip ROM, after even one instruction not stored in on-chip ROM has been fetched (e.g. from on-chip RAM).

(b) In expanded mode with on-chip ROM enabled (mode 2): when software standby mode is entered by executing an instruction stored in on-chip ROM, after even one instruction not stored in on-chip ROM has been fetched (e.g. from external memory or on-chip RAM).

Note that the H8/300 CPU pre-fetches instructions. If an instruction stored in the last two bytes of on-chip ROM is executed, the contents of the next two bytes, not in on-chip ROM, will be fetched as the next instruction.

This problem does not occur in expanded mode when on-chip ROM is disabled (mode 1).

In hardware standby mode there is no additional current dissipation, regardless of the conditions when hardware standby mode is entered.

## 12.5 Hardware Standby Mode

### 12.5.1 Transition to Hardware Standby Mode

Regardless of its current state, the chip enters the hardware standby mode whenever the $\overline{\text{STBY}}$ pin goes low.

The hardware standby mode reduces power consumption drastically by halting the CPU, stopping all the functions of the on-chip supporting modules, and placing I/O ports in the high-impedance state. The registers of the on-chip supporting modules are reset to their initial values. Only the on-chip RAM is held unchanged, provided the minimum necessary voltage supply is maintained (at least 2V).

Notes: 1. The RAME bit in the system control register should be cleared to 0 before the $\overline{\text{STBY}}$ pin goes low, to disable the on-chip RAM during the hardware standby mode.
2. Do not change the inputs at the mode pins ($MD_1$, $MD_0$) during hardware standby mode. Be particularly careful not to let both mode pins go low in hardware standby mode, since that places the chip in PROM mode and increases current drain.

### 12.5.2 Recovery from Hardware Standby Mode

Recovery from the hardware standby mode requires inputs at both the $\overline{\text{STBY}}$ and $\overline{\text{RES}}$ pins.

When the $\overline{\text{STBY}}$ pin goes high the clock oscillator begins running. The $\overline{\text{RES}}$ pin should be low at this time and should be held low long enough for the clock to stabilize. When the $\overline{\text{RES}}$ pin changes from low to high, the reset sequence is executed and the chip returns to the program execution state.

### 12.5.3 Timing Relationships

Figure 12-2 shows the timing relationships in the hardware standby mode.

In the sequence shown, first $\overline{\text{RES}}$ goes low, then $\overline{\text{STBY}}$ goes low, at which point the chip enters the hardware standby mode. To recover, first $\overline{\text{STBY}}$ goes high, then after the clock settling time, $\overline{\text{RES}}$ goes high.

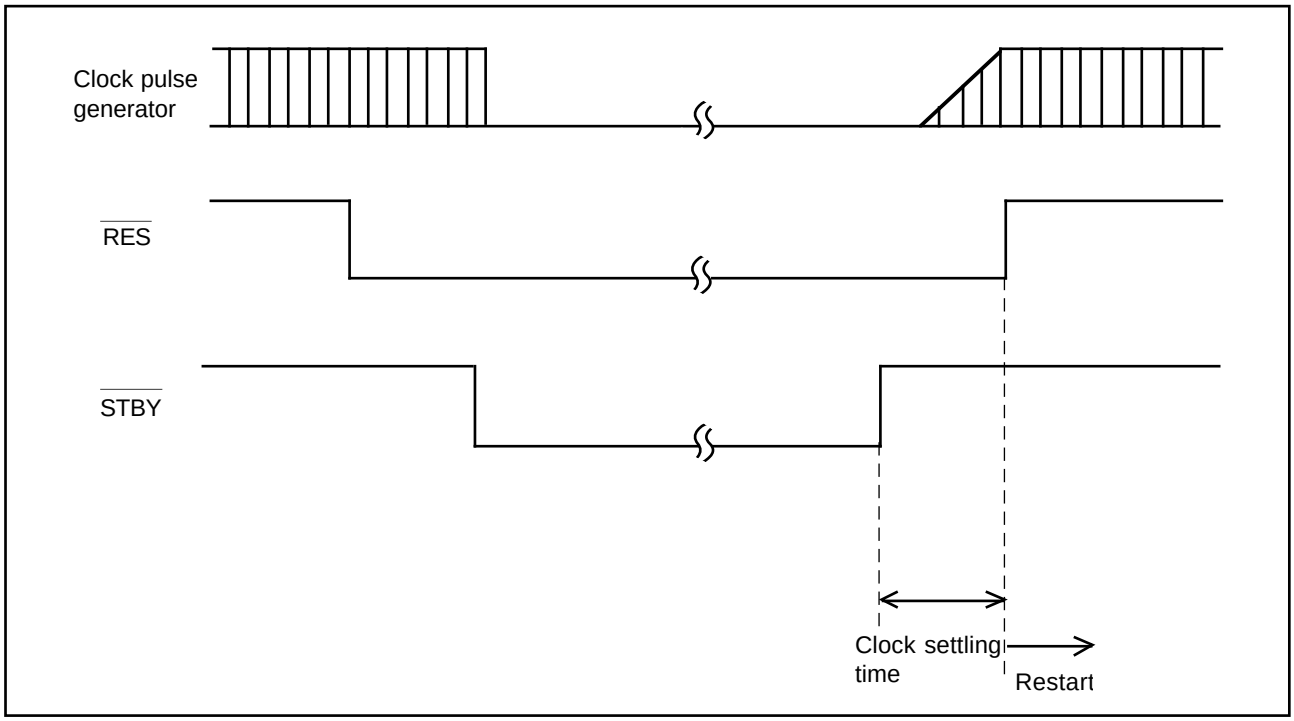



**Figure 12-2. Hardware Standby Mode Timing**

# Section 13. E-Clock Interface

## 13.1 Overview

For interfacing to peripheral devices that require it, the H8/325 series can generate an E clock output. Special instructions (MOVTPE, MOVFPE) perform data transfers synchronized with the E clock.

The E clock is created by dividing the system clock (Ø) by 8. The E clock is output at the $P4_7$ pin when the $P4_7$DDR bit in the port 4 data direction register (P4DDR) is set to 1. It is output only in the expanded modes (mode 1 and mode 2); it is not output in the single-chip mode. Output begins immediately after a reset.

When the CPU executes an instruction that synchronizes with the E clock, the address strobe ($\overline{AS}$), the address on the address bus, and the $\overline{IOS}$ signal are output as usual, but the $\overline{RD}$ and $\overline{WR}$ signal lines and the data bus do not become active until the falling edge of the E clock is detected. The length of the access cycle for an instruction synchronized with the E clock accordingly varies from 9 to 16 states. Figures 15-1 and 15-2 show the timing in the cases of maximum and minimum synchronization delay.

It is not possible to insert wait states ($T_W$) during the execution of an instruction synchronized with the E clock by input at the $\overline{WAIT}$ pin.

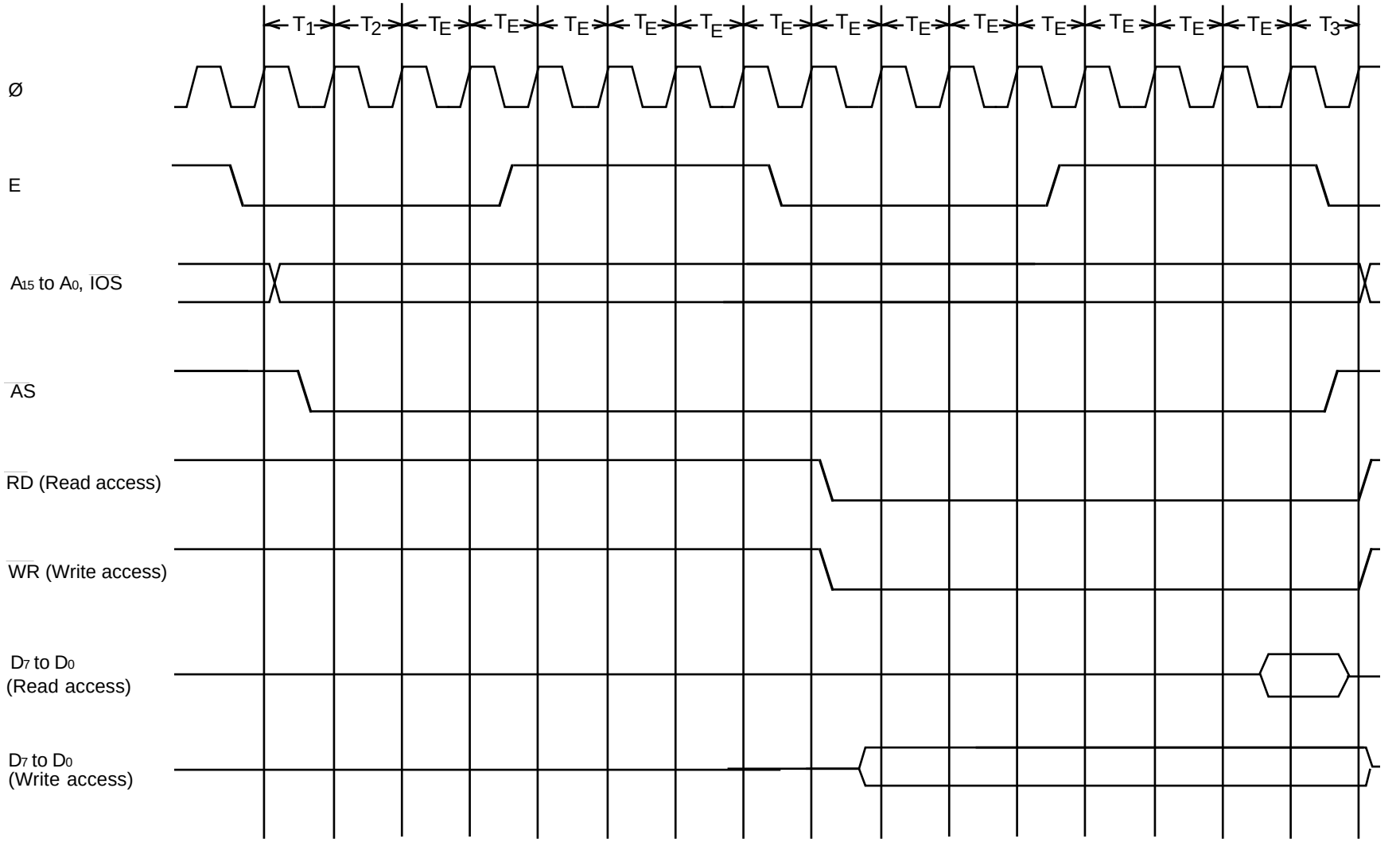



**Figure 13-1. Execution Cycle of Instruction Synchronized with E Clock in Expanded Modes (Maximum Synchronization Delay)**

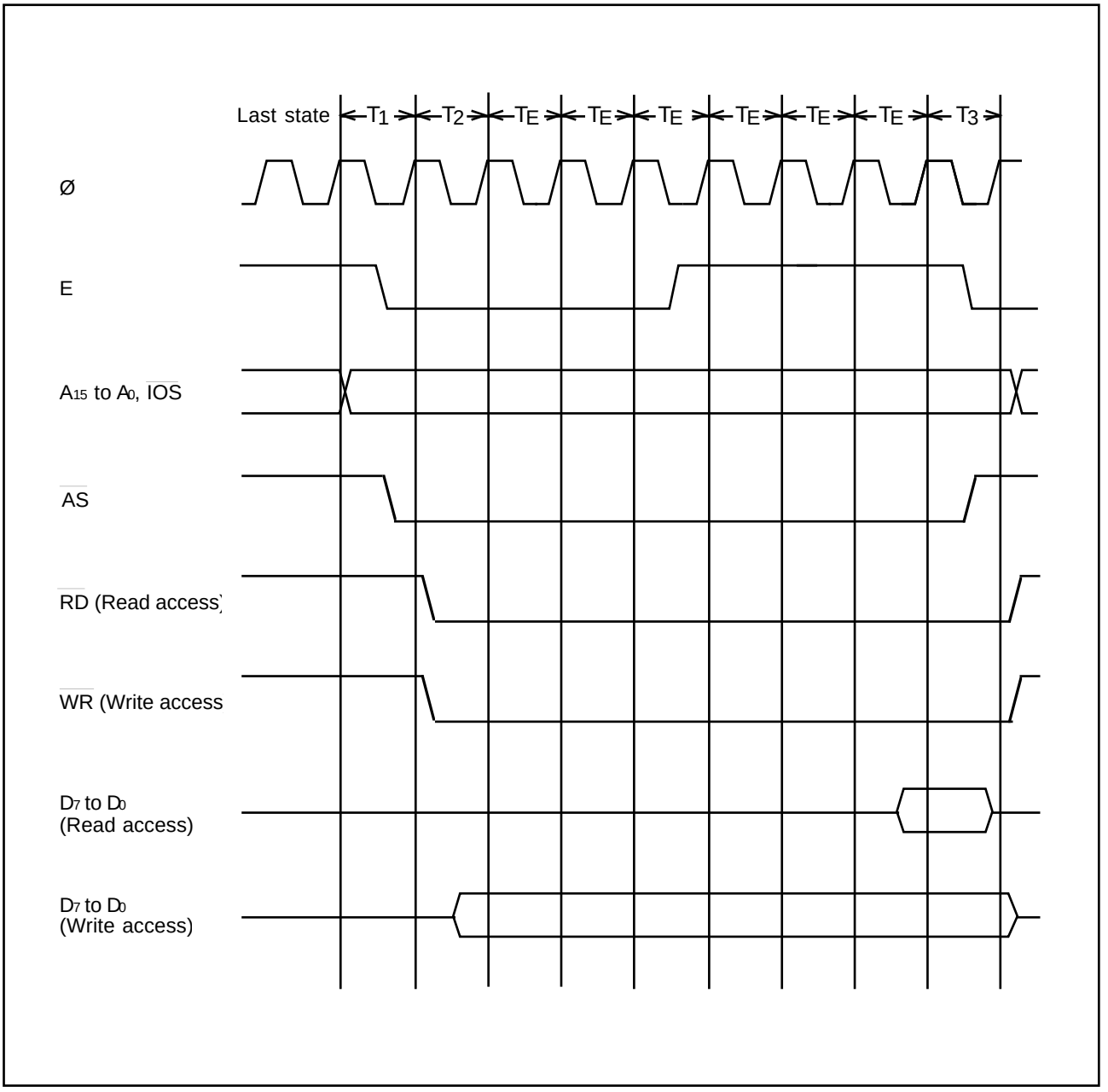



**Figure 13-2. Execution Cycle of Instruction Synchronized with E Clock in Expanded Modes (Minimum Synchronization Delay)**

# Section 14. Clock Pulse Generator

## 14.1 Overview

The H8/325 series chips have a built-in clock pulse generator (CPG) consisting of an oscillator circuit, a system clock divider, an E clock divider, and a prescaler. The prescaler generates clock signals for the on-chip supporting modules.

### 14.1.1 Block Diagram

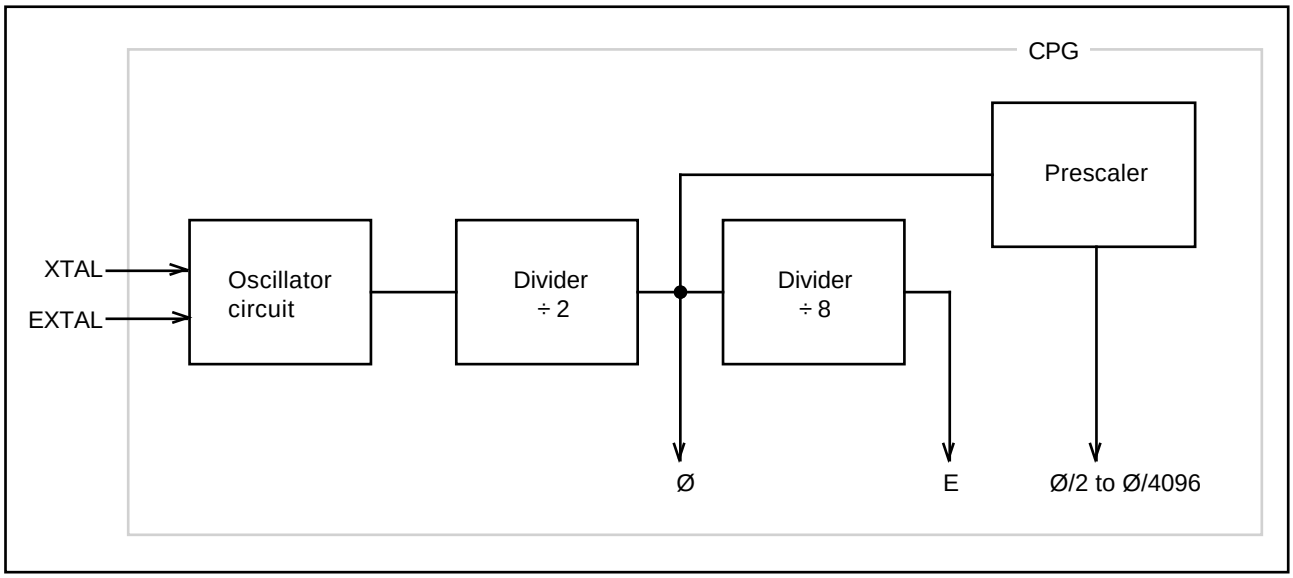



**Figure 14-1. Block Diagram of Clock Pulse Generator**

## 14.2 Oscillator Circuit

If an external crystal is connected across the EXTAL and XTAL pins, the on-chip oscillator circuit generates a clock signal for the system clock divider. Alternatively, an external clock signal can be applied to the EXTAL pin.

**(1) Connecting an External Crystal**

① **Circuit Configuration:** An external crystal can be connected as in the example in figure 14-2. An AT-cut parallel resonating crystal should be used.

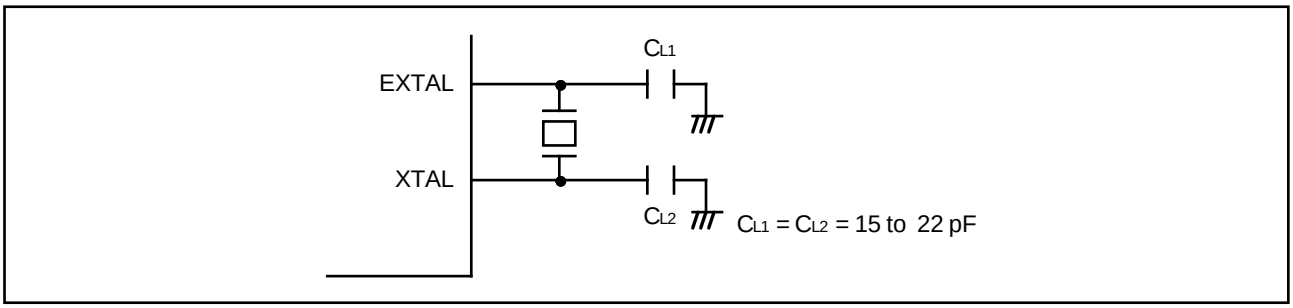



**Figure 14-2. Connection of Crystal Oscillator (Example)**

② **Crystal Oscillator:** The external crystal should have the characteristics listed in table 16-1.

**Table 14-1. External Crystal Parameters**

| Frequency (MHz) | 2 | 4 | 8 | 12 | 16 | 20 |
|---|---|---|---|---|---|---|
| Rs max (Ω) | 500 | 120 | 60 | 40 | 30 | 20 |
| $C_0$ (pF) | | | 7 pF max | | | |

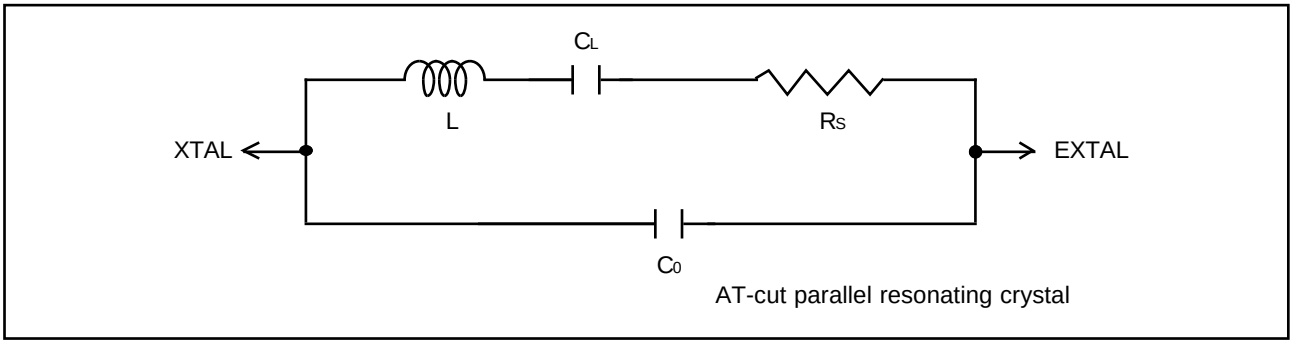



**Figure 14-3. Equivalent Circuit of External Crystal**

③ **Note on Board Design:** When an external crystal is connected, other signal lines should be kept away from the crystal circuit to prevent induction from interfering with correct oscillation. See figure 14-4. The crystal and its load capacitors should be placed as close as possible to the XTAL and EXTAL pins.

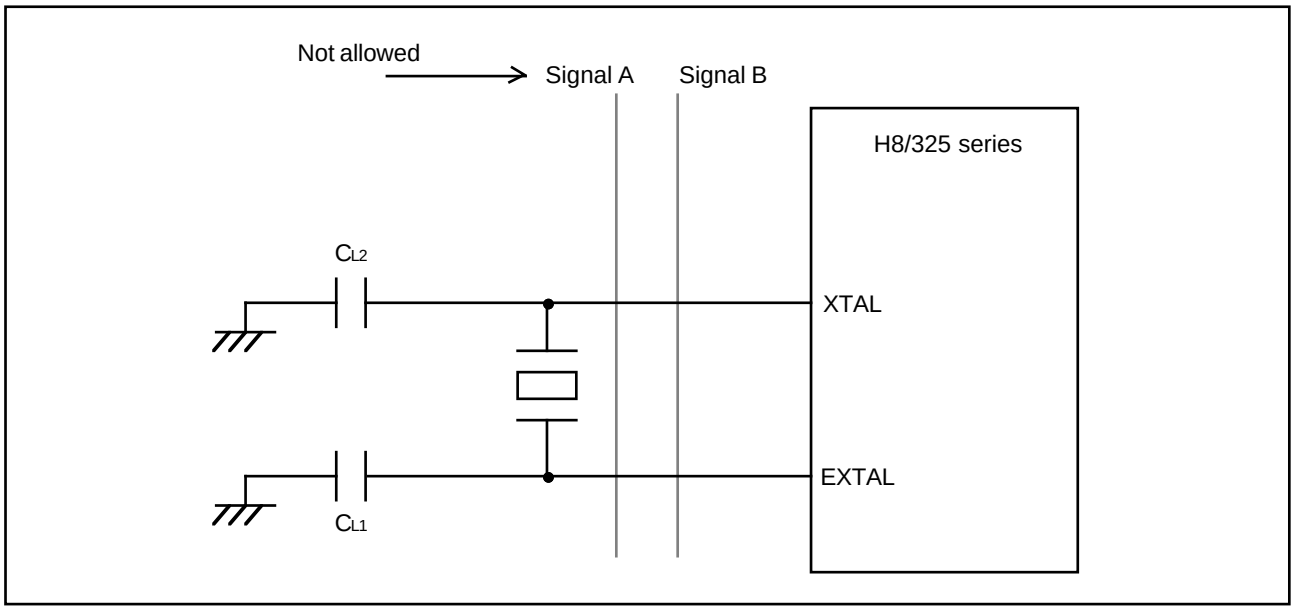



**Figure 14-4. Notes on Board Design around External Crystal**

### (2) Input of External Clock Signal

① **Circuit Configuration:** Figure 14-5 shows examples of signal connections for external clock input. In example (b), the external clock signal should be held high during the standby modes.

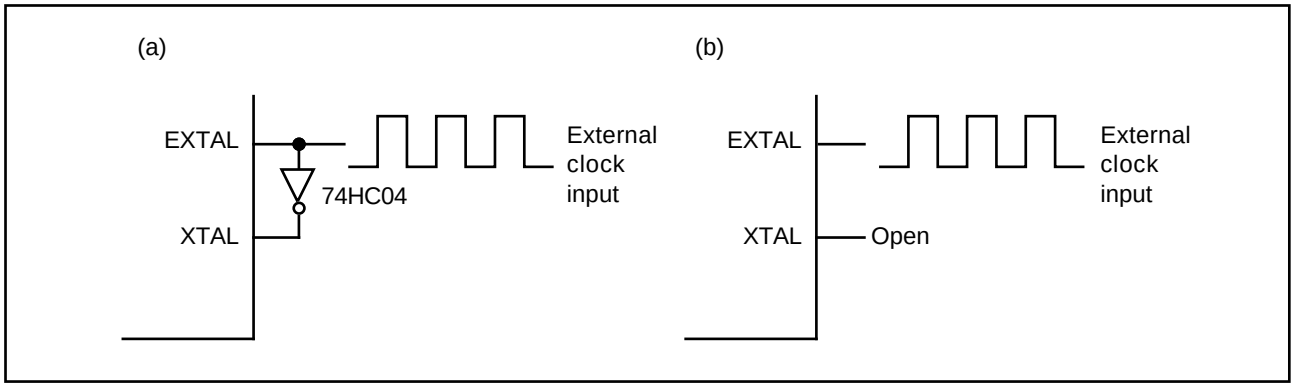



**Figure 14-5. External Clock Input (Example)**

② **External Clock Input**

| | |
|---|---|
| Frequency | Double the system clock (Ø) frequency |
| Duty factor | 45% to 55% |

## 14.3 System Clock Divider

The system clock divider divides the crystal oscillator or external clock frequency by 2 to create the system clock (Ø).

An E clock signal is created by dividing the system clock by 8.

Figure 16-6 shows the phase relationship of the E clock to the system clock.

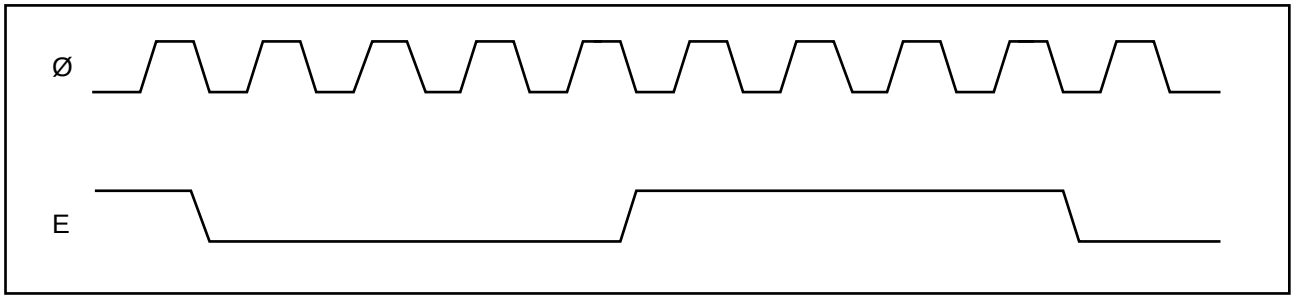



**Figure 14-6. Phase Relationship of System Clock and E Clock**

# Section 15. Electrical Specifications

## 15.1 Absolute Maximum Ratings

Table 15-1 lists the absolute maximum ratings.

**Table 15-1. Absolute Maximum Ratings**

| Item | Symbol | Rating | Unit |
|---|---|---|---|
| Supply voltage | $V_{CC}$ | –0.3 to +7.0 | V |
| Programming voltage | $V_{PP}$ | –0.3 to +13.5 | V |
| Input voltage | $V_{in}$ | –0.3 to $V_{CC}$ + 0.3 | V |
| Operating temperature | $T_{opr}$ | Regular specifications: –20 to +75 | ˚C |
| | | Wide-range specifications: –40 to +85 | ˚C |
| Storage temperature | $T_{stg}$ | –55 to +125 | ˚C |

**Note:** The input pins have protection circuits that guard against high static voltages and electric fields, but these high input-impedance circuits should never receive overvoltages exceeding the absolute maximum ratings shown in table 15-1.

## 15.2 Electrical Characteristics

### 15.2.1 DC Characteristics

Tables 15-2 and 15-3 list the DC characteristics of the H8/325 series.

**Table 15-2. DC Characteristics (5V Version)**

Conditions: $V_{CC}$ = 5.0V ±10%, $V_{SS}$ = 0V, Ta = –20 to 75˚C (regular specifications)
Ta = –40 to 85˚C (wide-range specifications)

| Item | | Symbol | min | typ | max | Unit | Measurement conditions |
|---|---|---|---|---|---|---|---|
| Schmitt trigger input voltage (1) | $P6_6$ to $P6_3$, $P6_0$, $P7_0$ | $V_T^-$ | 1.0 | – | – | V | |
| | | $V_T^+$ | – | – | $V_{CC} \times 0.7$ | V | |
| | | $V_T^+ - V_T^-$ | 0.4 | – | – | V | |
| Input high voltage (2) | $\overline{RES}$, $\overline{STBY}$ $MD_1$, $MD_0$ EXTAL, $\overline{NMI}$ | $V_{IH}$ | $V_{CC}$ – 0.7 | – | $V_{CC}$ + 0.3 | V | |
| Input high voltage | Input pins other than (1) and (2) | $V_{IH}$ | 2.0 | – | $V_{CC}$ + 0.3 | V | |
| Input low voltage (3) | $\overline{RES}$, $\overline{STBY}$ $MD_1$, $MD_0$, EXTAL | $V_{IL}$ | –0.3 | – | 0.5 | V | |
| Input low voltage | Input pins other than (1) and (3) | $V_{IL}$ | –0.3 | – | 0.8 | V | |
| Output high voltage | All output pins | $V_{OH}$ | $V_{CC}$ – 0.5 | – | – | V | $I_{OH}$ = –200 µA |
| | | | 3.5 | – | – | V | $I_{OH}$ = –1.0 mA |
| Output low voltage | All output pins | $V_{OL}$ | – | – | 0.4 | V | $I_{OL}$ = 1.6 mA |
| | $P1_7$ to $P1_0$, $P2_7$ to $P2_0$ | | – | – | 1.0 | V | $I_{OL}$ = 10.0 mA |
| Input leakage current | $\overline{RES}$ | $\|I_{in}\|$ | – | – | 10.0 | µA | $V_{in}$ = 0.5 V to $V_{CC}$ – 0.5 V |
| | $\overline{STBY}$, $\overline{NMI}$, $MD_1$, $MD_0$ | | – | – | 1.0 | µA | |
| Leakage current in 3-state (off state) | Ports 1 to 7 | $\|I_{TSI}\|$ | – | – | 1.0 | µA | $V_{in}$ = 0.5 V to $V_{CC}$ – 0.5 V |
| Input pull-up MOS current | Ports 1 to 7 | -Ip | 30 | – | 250 | µA | $V_{in}$ = 0 V |

**Table 15-2. DC Characteristics (5V Version) (cont.)**

Conditions: $V_{CC}$ = $AV_{CC}$ = 5.0V ±10%, $V_{SS}$ = 0V, Ta = –20 to 75˚C (regular specifications)
Ta = –40 to 85˚C (wide-range specifications)

| Item | | Symbol | min | typ | max | Unit | Measurement conditions |
|---|---|---|---|---|---|---|---|
| Input capacitance | $\overline{RES}$ | $C_{in}$ | – | – | 60 | pF | $V_{in}$ = 0 V |
| | $\overline{NMI}$ | | – | – | 30 | pF | f = 1 MHz |
| | All input pins except $\overline{RES}$ and $\overline{NMI}$ | | – | – | 15 | pF | Ta = 25˚C |
| Current dissipation*1 | Normal operation | $I_{CC}$ | – | 12 | 25 | mA | f = 6 MHz |
| | | | – | 16 | 30 | mA | f = 8 MHz |
| | | | – | 20 | 40 | mA | f = 10 MHz |
| | Sleep mode | | – | 8 | 15 | mA | f = 6 MHz |
| | | | – | 10 | 20 | mA | f = 8 MHz |
| | | | – | 12 | 25 | mA | f = 10 MHz |
| | Standby modes*2 | | – | 0.01 | 5.0 | µA | |
| RAM standby voltage | | $V_{RAM}$ | 2.0 | – | – | V | |

Notes: 1. Current dissipation values assume that $V_{IH\ min.}$ = $V_{CC}$ – 0.5V, $V_{IL\ max.}$ = 0.5V, all output pins are in the no-load state, and all MOS input pull-ups are off.
2. For these values it is assumed that $V_{RAM} \leq V_{CC} < 4.5$ V and $V_{IH}$ min = $V_{CC} \times 0.9$, $V_{IL}$ max = 0.3 V.

**Table 15-3. DC Characteristics (3V Version for only H8/3257 and H8/3256)**

Conditions: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C

| Item | | Symbol | min | typ | max | Unit | Measurement conditions |
|---|---|---|---|---|---|---|---|
| Schmitt trigger input voltage (1) | $P6_6$ to $P6_3$, $P6_0$, $P7_0$ | $V_T^-$ | $V_{CC} \times 0.15$ | – | – | V | |
| | | $V_T^+$ | – | – | $V_{CC} \times 0.7$ | V | |
| | | $V_T^+ - V_T^-$ | 0.2 | – | – | V | |
| Input high voltage (2) | $\overline{RES}$, $\overline{STBY}$ $MD_1$, $MD_0$ EXTAL, $\overline{NMI}$ | $V_{IH}$ | $V_{CC} \times 0.9$ | – | $V_{CC}$ + 0.3 | V | |
| Input high voltage | Input pins other than (1) and (2) | $V_{IH}$ | $V_{CC} \times 0.7$ | – | $V_{CC}$ + 0.3 | V | |
| Input low voltage (3) | $\overline{RES}$, $\overline{STBY}$ $MD_1$, $MD_0$, EXTAL | $V_{IL}$ | –0.3 | – | $V_{CC} \times 0.1$ | V | |
| Input low voltage | Input pins other than (1) and (3) | $V_{IL}$ | –0.3 | – | $V_{CC} \times 0.15$ | V | |
| Output high voltage | All output pins | $V_{OH}$ | $V_{CC}$ – 0.4 | – | – | V | $I_{OH}$ = –200 µA |
| | | | $V_{CC}$ – 0.9 | – | – | V | $I_{OH}$ = –1.0 mA |
| Output low voltage | $P1_7$ to $P1_0$, $P2_7$ to $P2_0$ | $V_{OL}$ | – | – | 0.4 | V | $I_{OL}$ = 1.6 mA |
| | All output pins | | – | – | 0.4 | V | $I_{OL}$ = 0.8 mA |
| Input leakage current | $\overline{RES}$ | $\|I_{in}\|$ | – | – | 10.0 | µA | $V_{in}$ = 0.5 V to $V_{CC}$ – 0.5 V |
| | $\overline{STBY}$, $\overline{NMI}$, $MD_1$, $MD_0$ | | – | – | 1.0 | µA | |
| Leakage current in 3-state (off state) | Ports 1 to 7 | $\|I_{TSI}\|$ | – | – | 1.0 | µA | $V_{in}$ = 0.5 V to $V_{CC}$ – 0.5 V |
| Input pull-up MOS current | Ports 1 to 7 | -Ip | 3 | – | 120 | µA | $V_{CC}$ = 3.3 V $V_{in}$ = 0 V |

**Table 15-3. DC Characteristics (3V Version for only H8/3257 and H8/3256) (cont.)**

Conditions: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C

| Item | | Symbol | min | typ | max | Unit | Measurement conditions |
|---|---|---|---|---|---|---|---|
| Input capacitance | $\overline{\text{RES}}$ | $C_{in}$ | – | – | 60 | pF | $V_{in}$ = 0 V |
| | $\overline{\text{NMI}}$ | | – | – | 30 | pF | f = 1 MHz |
| | All input pins except $\overline{\text{RES}}$ and $\overline{\text{NMI}}$ | | – | – | 15 | pF | Ta = 25˚C |
| Current dissipation* | Normal operation | $I_{CC}$ | – | 4 | – | mA | f = 3 MHz |
| | Sleep mode | | – | 3 | – | mA | |
| | Normal operation | | – | 6 | 12 | mA | f = 5 MHz |
| | Sleep mode | | – | 4 | 8 | mA | |
| | Standby modes | | – | 0.01 | 5.0 | µA | |
| RAM standby voltage | | $V_{RAM}$ | 2.0 | – | – | V | |

Note: Current dissipation values assume that $V_{IH\ min.} = V_{CC} – 0.5V$, $V_{IL\ max.} = 0.5V$, all output pins are in the no-load state, and all MOS input pull-ups are off.

**Table 15-4. Allowable Output Current Sink Values**

Conditions: $V_{CC}$ = 5.0V ±10%, $V_{SS}$ = 0V, Ta = –20 to 75˚C (regular specifications)
Ta = –40 to 85˚C (wide-range specifications)

| Item | | Symbol | min | typ | max | Unit |
|---|---|---|---|---|---|---|
| Allowable output low current sink (per pin) | Ports 1 and 2 | $I_{OL}$ | – | – | 10 | mA |
| | Other output pins | | – | – | 2.0 | mA |
| Allowable output low current sink (total) | Ports 1 and 2, total | $\Sigma I_{OL}$ | – | – | 80 | mA |
| | All output pins | | – | – | 120 | mA |
| Allowable output high current sink (per pin) | All output pins | $-I_{OH}$ | – | – | 2.0 | mA |
| Allowable output high current sink (total) | Total of all output | $\Sigma -I_{OH}$ | – | – | 40 | mA |

**Note:** To avoid degrading the reliability of the chip, be careful not to exceed the output current sink values in table 15-4. In particular, when driving a Darlington pair or LED directly, be sure to insert a current-limiting resistor in the output path. See figures 17-1 and 17-2.

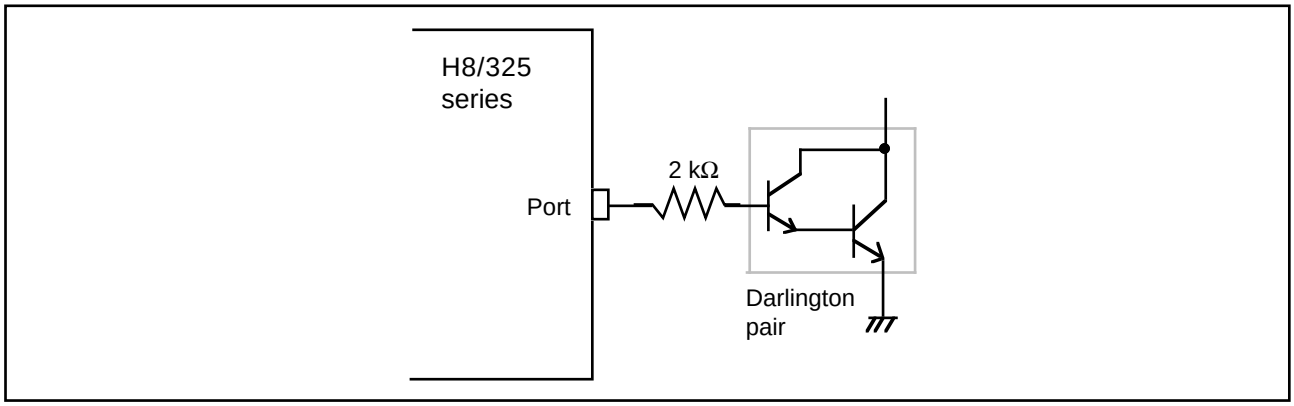



**Figure 15-1. Example of Circuit for Driving a Darlington Pair**

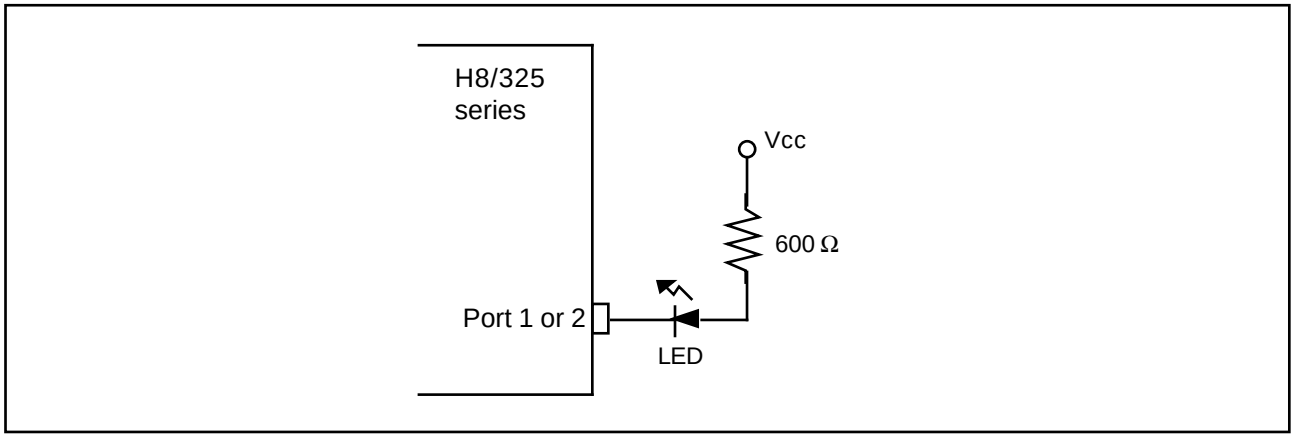



**Figure 15-2. Example of Circuit for Driving a LED**

**Table 15-5. Allowable Output Current Sink Values (3V Version for only H8/3257 and H8/3256)**

Conditions: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C

| Item | | Symbol | min | typ | max | Unit |
|---|---|---|---|---|---|---|
| Allowable output low current sink (per pin) | Ports 1 and 2 | $I_{OL}$ | – | – | 2 | mA |
| | Other output pins | | – | – | 1.0 | mA |
| Allowable output low current sink (total) | Ports 1 and 2, total of 16 pins | $\Sigma I_{OL}$ | – | – | 40 | mA |
| | Total of all other output pins | | – | – | 60 | mA |
| Allowable output high current sink (per pin) | All output pins | $-I_{OH}$ | – | – | 2.0 | mA |
| Allowable output high current sink (total) | Total of all output pins | $\Sigma -I_{OH}$ | – | – | 30 | mA |

**Note:** To avoid degrading the reliability of the chip, be careful not to exceed the output current sink values in table 15-5.

### 15.2.2 AC Characteristics

The AC characteristics of the H8/325 series are listed in three tables. Bus timing parameters are given in table 15-6, control signal timing parameters in table 15-7, and timing parameters of the on-chip supporting modules in table 15-8.

**Table 15-6. Bus Timing**

Condition A: $V_{CC}$ = 5.0V ±10%, Ø = 0.5 to 10MHz, $V_{SS}$ = 0V,
Ta = –20 to 75˚C (regular specifications), Ta = –40 to 85˚C (wide-range specifications)
Condition B: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C, for only H8/3257 and H8/3256

| | | Condition B | | Condition A | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 5MHz | | 6MHz | | 8MHz | | 10MHz | | | Measurement |
| Item | Symbol | min | max | min | max | min | max | min | max | Unit | conditions |
| Clock cycle time | $t_{cyc}$ | 200 | 2000 | 166.7 | 2000 | 125 | 2000 | 100 | 2000 | ns | Fig. 15-4 |
| Clock pulse width Low | $t_{CL}$ | 65 | – | 65 | – | 45 | – | 35 | – | ns | Fig. 15-4 |
| Clock pulse width High | $t_{CH}$ | 65 | – | 65 | – | 45 | – | 35 | – | 1ns | Fig. 15-4 |
| Clock rise time | $t_{Cr}$ | – | 25 | – | 15 | – | 15 | – | 15 | ns | Fig. 15-4 |
| Clock fall time | $t_{Cf}$ | – | 25 | – | 15 | – | 15 | – | 15 | ns | Fig. 15-4 |
| Address delay time | $t_{AD}$ | – | 90 | – | 70 | – | 60 | – | 55 | ns | Fig. 15-4 |
| Address hold time | $t_{AH}$ | 30 | – | 30 | – | 25 | – | 20 | – | ns | Fig. 15-4 |
| Address strobe delay time | $t_{ASD}$ | – | 80 | – | 70 | – | 60 | – | 40 | ns | Fig. 15-4 |
| Write strobe delay time | $t_{WSD}$ | – | 80 | – | 70 | – | 60 | – | 50 | ns | Fig. 15-4 |
| Strobe delay time | $t_{SD}$ | – | 90 | – | 70 | – | 60 | – | 50 | ns | Fig. 15-4 |
| Write strobe pulse width | $t_{WSW}$ | 200 | – | 200 | – | 150 | – | 120 | – | ns | Fig. 15-4 |
| Address setup time 1 | $t_{AS1}$ | 25 | – | 25 | – | 20 | – | 15 | – | ns | Fig. 15-4 |
| Address setup time 2 | $t_{AS2}$ | 105 | – | 105 | – | 80 | – | 65 | – | ns | Fig. 15-4 |
| Read data setup time | $t_{RDS}$ | 90 | – | 60 | – | 50 | – | 35 | – | ns | Fig. 15-4 |
| Read data hold time | $t_{RDH}$ | 0 | – | 0 | – | 0 | – | 0 | – | ns | Fig. 15-4 |
| Write data delay time | $t_{WDD}$ | – | 125 | – | 85 | – | 75 | – | 75 | ns | Fig. 15-4 |
| Read data access time | $t_{ACC}$ | – | 300 | – | 280 | – | 210 | – | 170 | ns | Fig. 15-4 |
| Write data setup time | $t_{WDS}$ | 10 | – | 30 | – | 15 | – | 10 | – | ns | Fig. 15-4 |
| Write data hold time | $t_{WDH}$ | 30 | – | 30 | – | 25 | – | 20 | – | ns | Fig. 15-4 |
| Wait setup time | $t_{WTS}$ | 60 | – | 45 | – | 45 | – | 45 | – | ns | Fig. 15-5 |
| Wait hold time | $t_{WTH}$ | 20 | – | 10 | – | 10 | – | 10 | – | ns | Fig. 15-5 |
| E clock delay time | $t_{ED}$ | – | 30 | – | 25 | – | 25 | – | 25 | ns | Fig. 15-6 |
| E clock rise time | $t_{Er}$ | – | 25 | – | 15 | – | 15 | – | 15 | ns | Fig. 15-6 |
| E clock fall time | $t_{Ef}$ | – | 25 | – | 15 | – | 15 | – | 15 | ns | Fig. 15-6 |
| Read data hold time (for E clock) | $t_{RDHE}$ | 0 | – | 0 | – | 0 | – | 0 | – | ns | Fig. 15-6 |
| Write data hold time (for E clock) | $t_{WDHE}$ | 60 | – | 50 | – | 40 | – | 30 | – | ns | Fig. 15-6 |

**Table 15-7. Control Signal Timing**

Condition A: $V_{CC}$ = 5.0V ±10%, Ø = 0.5 to 10MHz, $V_{SS}$ = 0V,
Ta = –20 to 75˚C (regular specifications), Ta = –40 to 85˚C (wide-range specifications)
Condition B: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C, for only H8/3257 and H8/3256

| | | Condition B | | Condition A | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 5MHz | | 6MHz | | 8MHz | | 10MHz | | | Measurement |
| Item | Symbol | min | max | min | max | min | max | min | max | Unit | conditions |
| $\overline{RES}$ setup time | $t_{RESS}$ | 300 | – | 200 | – | 200 | – | 200 | – | ns | Fig. 15-7 |
| $\overline{RES}$ pulse width | $t_{RESW}$ | 10 | – | 10 | – | 10 | – | 10 | – | $t_{cyc}$ | Fig. 15-7 |
| Mode programming setup time | $t_{MDS}$ | 4 | – | 4 | – | 4 | – | 4 | – | $t_{cyc}$ | Fig. 15-7 |
| $\overline{NMI}$ setup time ($\overline{NMI}$, $\overline{IRQ_0}$ to $\overline{IRQ_2}$) | $t_{NMIS}$ | 300 | – | 150 | – | 150 | – | 150 | – | ns | Fig. 15-8 |
| $\overline{NMI}$ hold time ($\overline{NMI}$, $\overline{IRQ_0}$ to $\overline{IRQ_2}$) | $t_{NMIH}$ | 10 | – | 10 | – | 10 | – | 10 | – | ns | Fig. 15-8 |
| Interrupt pulse width for recovery from software standby mode ($\overline{NMI}$, $\overline{IRQ_0}$ to $\overline{IRQ_2}$) | $t_{NMIW}$ | 300 | – | 200 | – | 200 | – | 200 | – | ns | Fig. 15-8 |
| Crystal oscillator settling time (reset) | $t_{OSC1}$ | 20 | – | 20 | – | 20 | – | 20 | – | ms | Fig. 15-9 |
| Crystal oscillator settling time (software standby) | $t_{OSC2}$ | 10 | – | 10 | – | 10 | – | 10 | – | ms | Fig. 15-10 |

**Table 15-8. Timing Conditions of On-Chip Supporting Modules**

Condition A: $V_{CC}$ = 5.0V ±10%, Ø = 0.5 to 10MHz, $V_{SS}$ = 0V,
Ta = –20 to 75˚C (regular specifications), Ta = –40 to 85˚C (wide-range specifications)
Condition B: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C, for only H8/3257 and H8/3256

| | | | Condition B | | Condition A | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 5MHz | | 6MHz | | 8MHz | | 10MHz | | | Measurement |
| Item | | Symbol | min | max | min | max | min | max | min | max | Unit | conditions |
| FRT | Timer output delay time | $t_{FTOD}$ | – | 150 | – | 100 | – | 100 | – | 100 | ns | Fig. 15-11 |
| | Timer input setup time | $t_{FTIS}$ | 80 | – | 50 | – | 50 | – | 50 | – | ns | Fig. 15-11 |
| | Timer clock input setup time | $t_{FTCS}$ | 80 | – | 50 | – | 50 | – | 50 | – | ns | Fig. 15-12 |
| | Timer clock pulse width | $t_{FTCWH}$ $t_{FTCWL}$ | 1.5 | – | 1.5 | – | 1.5 | – | 1.5 | – | $t_{cyc}$ | Fig. 15-12 |

**Table 15-8. Timing Conditions of On-Chip Supporting Modules (cont.)**

Condition A: $V_{CC}$ = 5.0V ±10%, Ø = 0.5 to 10MHz, $V_{SS}$ = 0V,
Ta = –20 to 75˚C (regular specifications), Ta = –40 to 85˚C (wide-range specifications)

Condition B: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C, for only H8/3257 and H8/3256

| | | | Condition B | | Condition A | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 5MHz | | 6MHz | | 8MHz | | 10MHz | | | Measurement |
| Item | | Symbol | min | max | min | max | min | max | min | max | Unit | conditions |
| TMR | Timer output delay time | $t_{TMOD}$ | – | 150 | – | 100 | – | 100 | – | 100 | ns | Fig. 15-13 |
| | Timer reset input setup time | $t_{TMRS}$ | 80 | – | 50 | – | 50 | – | 50 | – | ns | Fig. 15-15 |
| | Timer clock input setup time | $t_{TMCS}$ | 80 | – | 50 | – | 50 | – | 50 | – | ns | Fig. 15-14 |
| | Timer clock pulse width (single edge) | $t_{TMCWH}$ | 1.5 | – | 1.5 | – | 1.5 | – | 1.5 | – | $t_{cyc}$ | Fig. 15-14 |
| | Timer clock pulse width (both edges) | $t_{TMCWL}$ | 2.5 | – | 2.5 | – | 2.5 | – | 2.5 | – | $t_{cyc}$ | Fig. 15-14 |
| SCI | Input clock cycle (Async) | $t_{Scyc}$ | 2 | – | 2 | – | 2 | – | 2 | – | $t_{cyc}$ | Fig. 15-16 |
| | (Sync) | $t_{Scyc}$ | 4 | – | 4 | – | 4 | – | 4 | – | $t_{cyc}$ | Fig. 15-16 |
| | Transmit data delay time (Sync) | $t_{TXD}$ | – | 200 | – | 100 | – | 100 | – | 100 | ns | Fig. 15-16 |
| | Receive data setup time (Sync) | $t_{RXS}$ | 150 | – | 100 | – | 100 | – | 100 | – | ns | Fig. 15-16 |
| | Receive data hold time (Sync) | $t_{RXH}$ | 150 | – | 100 | – | 100 | – | 100 | – | ns | Fig. 15-16 |
| | Input clock pulse width | $t_{SCKW}$ | 0.4 | 0.6 | 0.4 | 0.6 | 0.4 | 0.6 | 0.4 | 0.6 | $t_{Scyc}$ | Fig. 15-17 |
| Ports | Output data delay time | $t_{PWD}$ | – | 150 | – | 100 | – | 100 | – | 100 | ns | Fig. 15-18 |
| | Input data setup time | $t_{PRS}$ | 80 | – | 50 | – | 50 | – | 50 | – | ns | Fig. 15-18 |
| | Input data hold time | $t_{PRH}$ | 80 | – | 50 | – | 50 | – | 50 | – | ns | Fig. 15-18 |

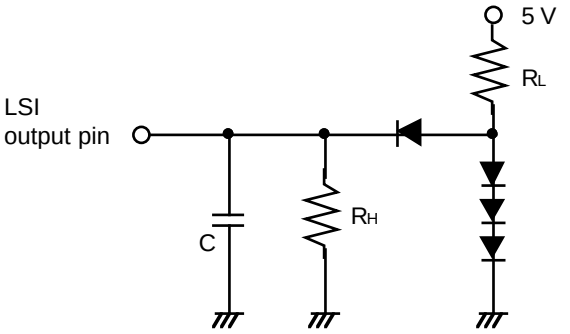
**Table 15-8. Timing Conditions of On-Chip Supporting Modules (cont.)**

Condition A: $V_{CC}$ = 5.0V ±10%, Ø = 0.5 to 10MHz, $V_{SS}$ = 0V,
Ta = –20 to 75˚C (regular specifications), Ta = –40 to 85˚C (wide-range specifications)

Condition B: $V_{CC}$ = 2.7 to 3.6V, $V_{SS}$ = 0V, Ta = –20 to 75˚C, for only H8/3257 and H8/3256

| | | | Condition B | | Condition A | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 5MHz | | 6MHz | | 8MHz | | 10MHz | | | Measurement |
| Item | | Symbol | min | max | min | max | min | max | min | max | Unit | conditions |
| Parallel handshake interface | Handshake input strobe pulse width | $t_{HISW}$ | 1.5 | – | 1.5 | – | 1.5 | – | 1.5 | – | $t_{cyc}$ | Fig. 15-19 |
| | Handshake input data setup time | $t_{HIS}$ | 10 | – | 10 | – | 10 | – | 10 | – | ns | Fig. 15-19 |
| | Handshake input data hold time | $t_{HIH}$ | 120 | – | 120 | – | 120 | – | 120 | – | ns | Fig. 15-19 |
| | Handshake output strobe delay time | $t_{HOSD1}$ | – | 100 | – | 80 | – | 80 | – | 80 | ns | Fig. 15-20 |
| | | $t_{HOSD2}$ | – | 100 | – | 80 | – | 80 | – | 80 | ns | Fig. 15-20 |
| | Busy output delay time | $t_{HBSOD1}$ | – | 150 | – | 150 | – | 150 | – | 150 | ns | Fig. 15-21 |
| | | $t_{HBSOD2}$ | – | 150 | – | 150 | – | 150 | – | 150 | ns | Fig. 15-21 |

- **Measurement Conditions for AC Characteristics**




C = 90 pF: Ports 1, 2, 3, $4_6$, 6, 7
30 pF: Ports 4 (except $4_6$), 5
$R_L$ = 2.4 kΩ
$R_H$ = 12 kΩ

Input/output timing reference levels
Low: 0.8 V
High: 2.0 V

**Figure 15-3. Output Load Circuit**

## 15.3 MCU Operational Timing

This section provides the following timing charts:


| | |
|---|---|
| 15.3.1 Bus Timing | Figures 15-4 to 15-6 |
| 15.3.2 Control Signal Timing | Figures 15-7 to 15-10 |
| 15.3.3 16-Bit Free-Running Timer Timing | Figures 15-11 to 15-12 |
| 15.3.4 8-Bit Timer Timing | Figures 15-13 to 15-15 |
| 15.3.6 SCI Timing | Figures 15-15 to 15-17 |
| 15.3.7 I/O Port Timing | Figure 15-18 |
| 15.3.8 Parallel Handshaking Interface Timing | Figures 15-19 to 15-21 |


### 15.3.1 Bus Timing

#### (1) Basic Bus Cycle (without Wait States) in Expanded Modes

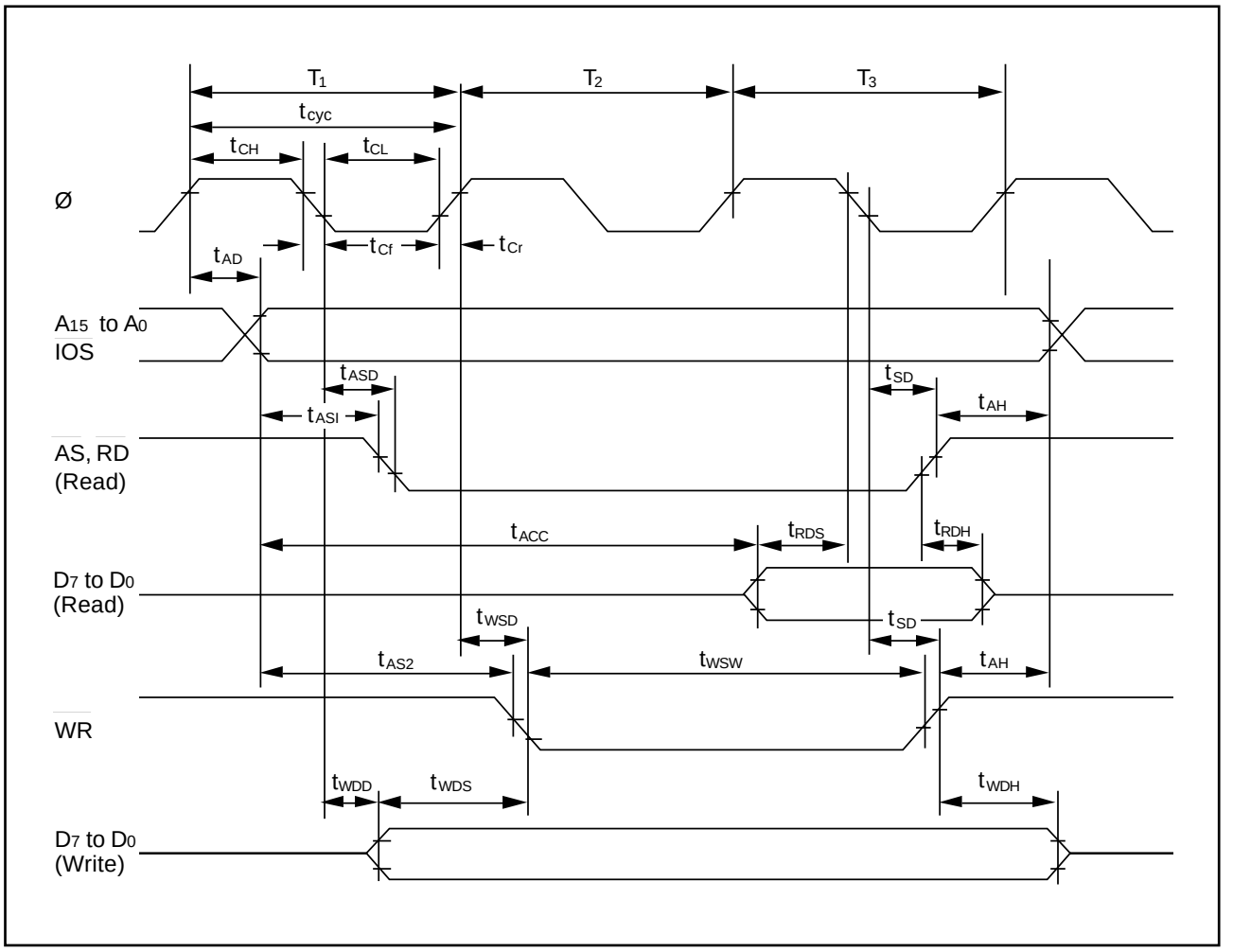



**Figure 15-4. Basic Bus Cycle (without Wait States) in Expanded Modes**

## (2) Basic Bus Cycle (with 1 Wait State) in Expanded Modes

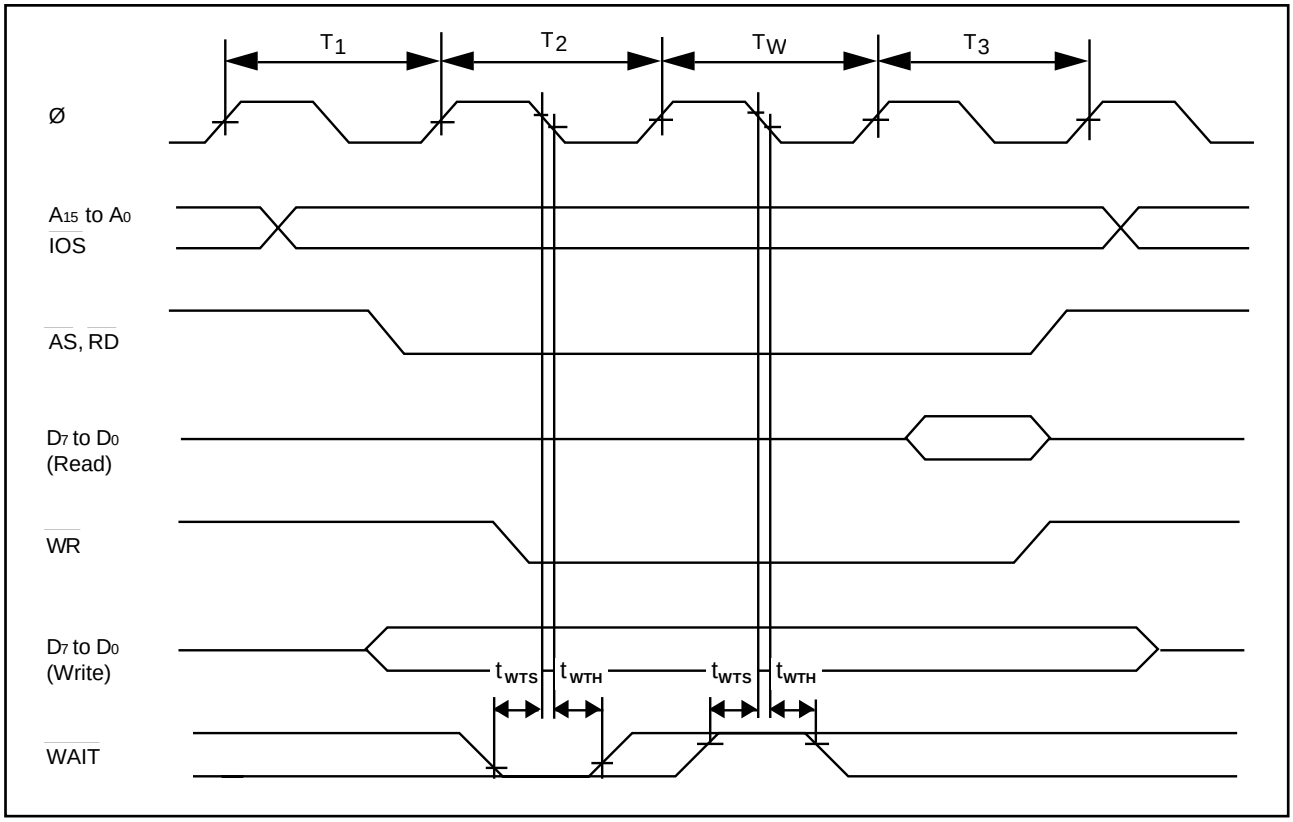



**Figure 15-5. Basic Bus Cycle (with 1 Wait State) in Expanded Modes**

### (3) E Clock Bus Cycle

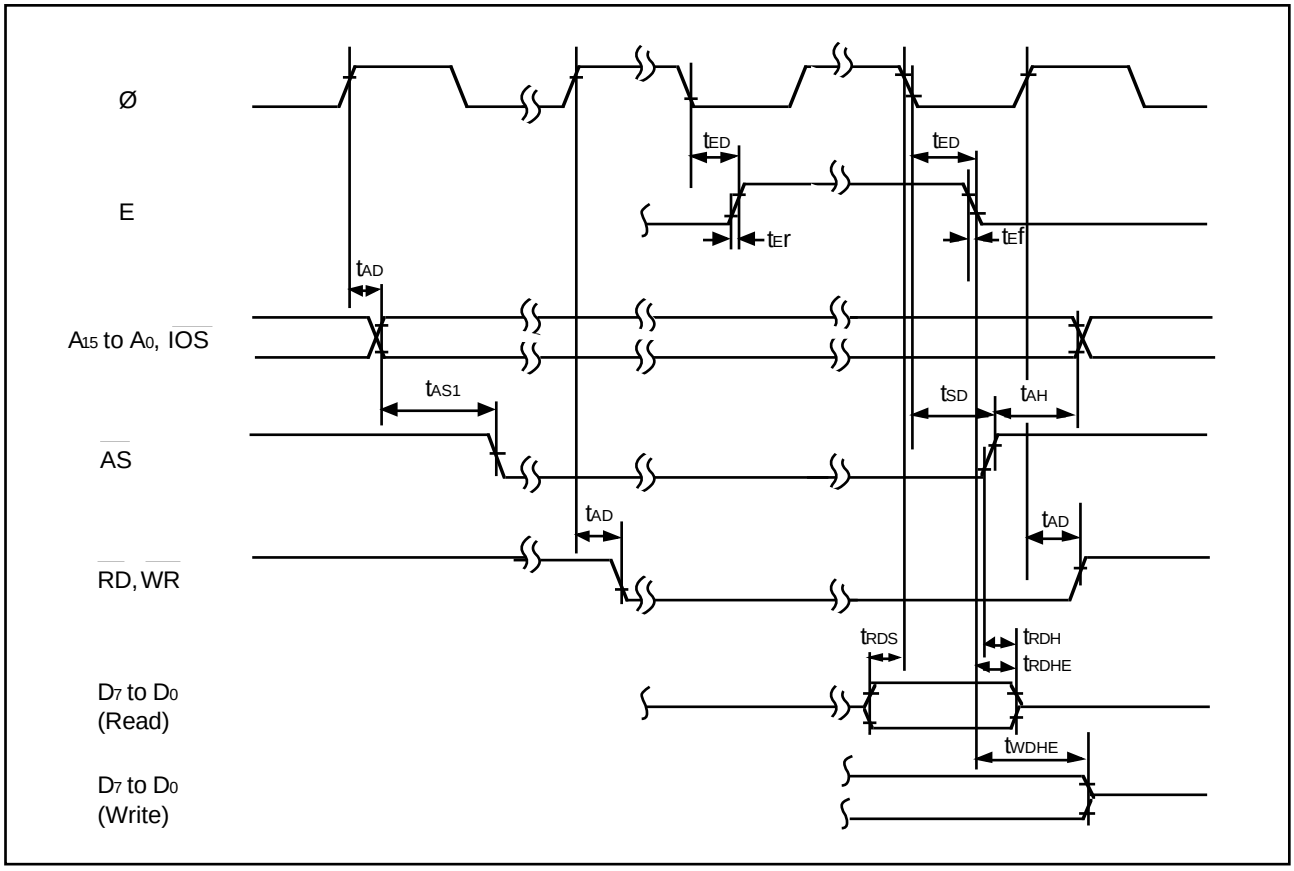



**Figure 15-6. E Clock Bus Cycle**

## 15.3.2 Control Signal Timing

### (1) Reset Input Timing

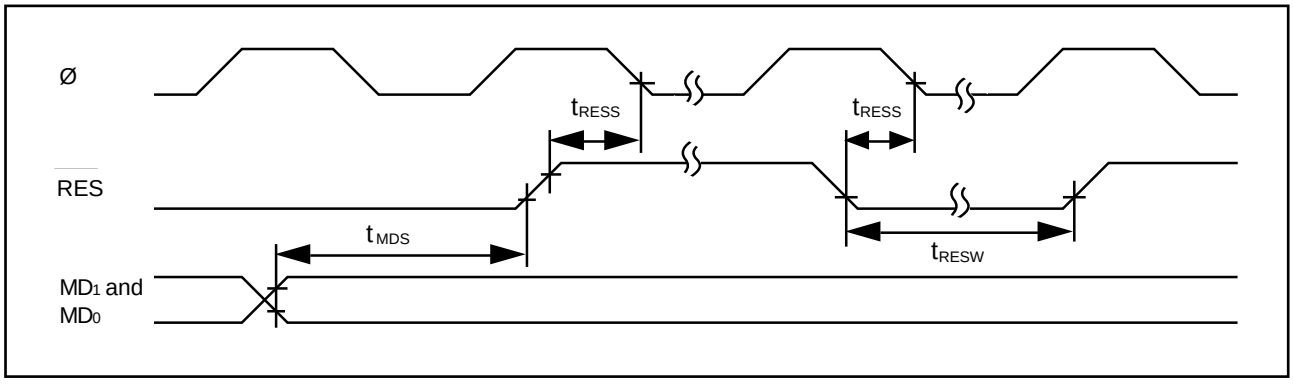



**Figure 15-7. Reset Input Timing**

## (2) Interrupt Input Timing

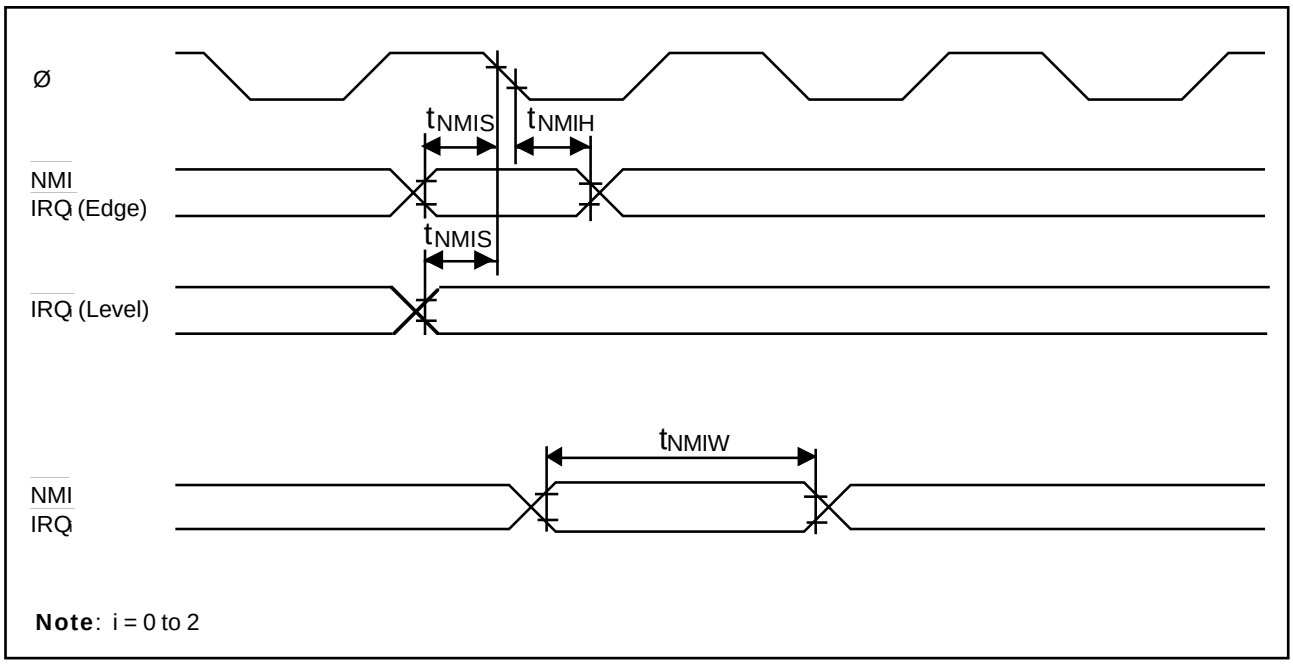



**Figure 15-8. Interrupt Input Timing**

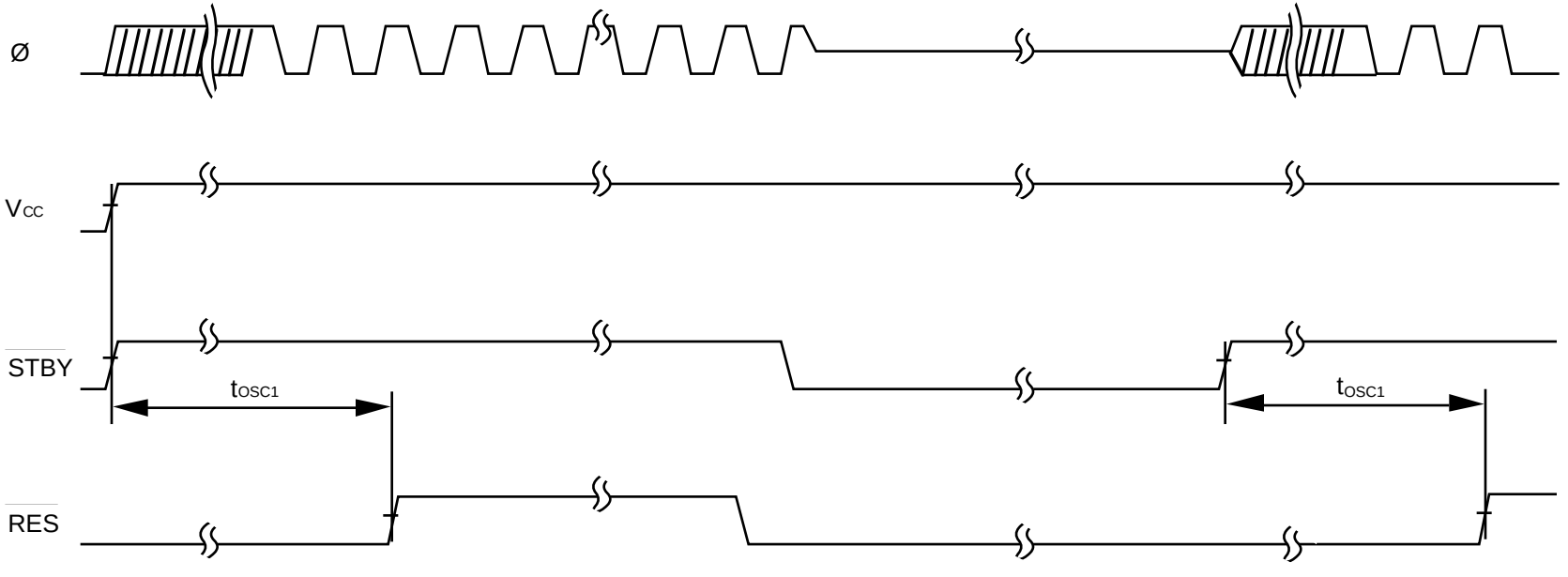



**Figure 15-9. Clock Settling Timing**

### (4) Clock Settling Timing for Recovery from Software Standby Mode

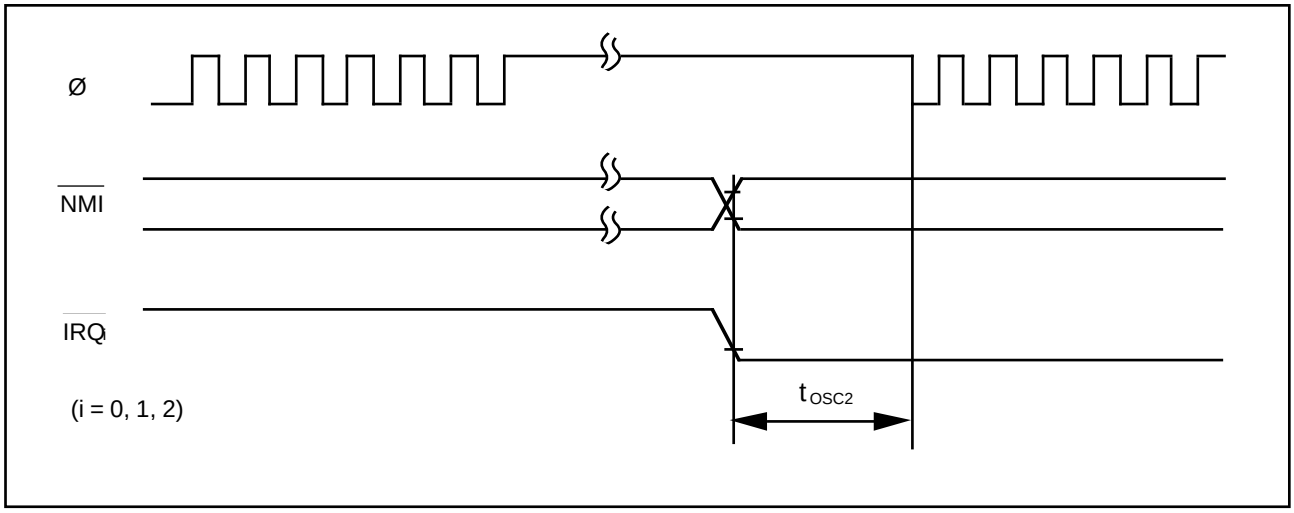



**Figure 15-10. Clock Settling Timing for Recovery from Software Standby Mode**

## 15.3.3 16-Bit Free-Running Timer Timing

### (1) Free-Running Timer Input/Output Timing

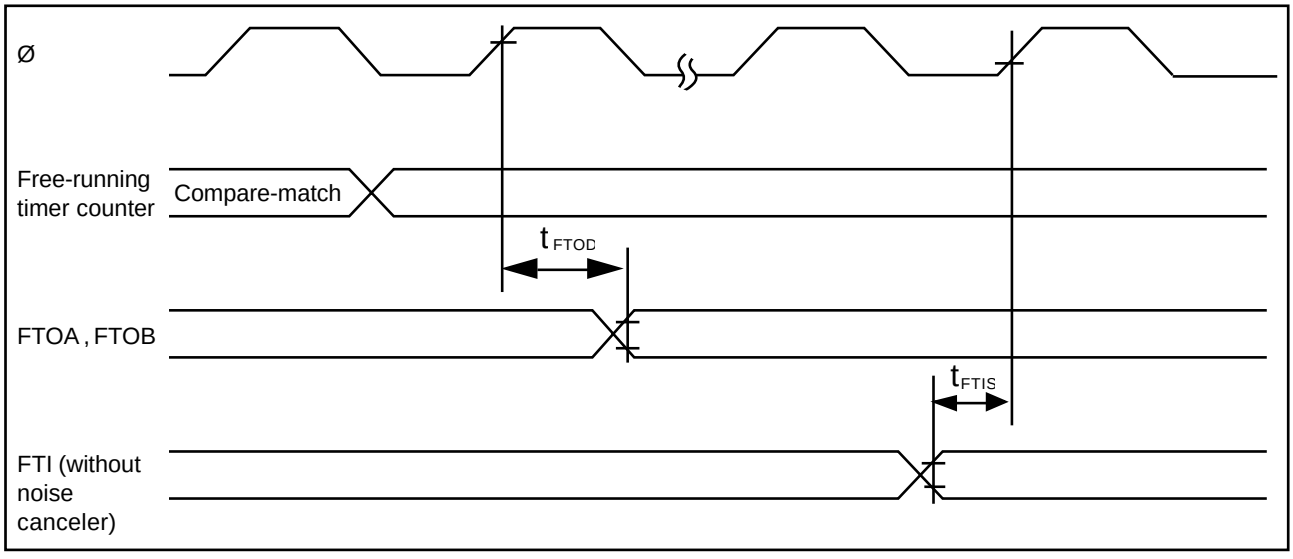



**Figure 15-11. Free-Running Timer Input/Output Timing**

### (2) External Clock Input Timing for Free-Running Timer

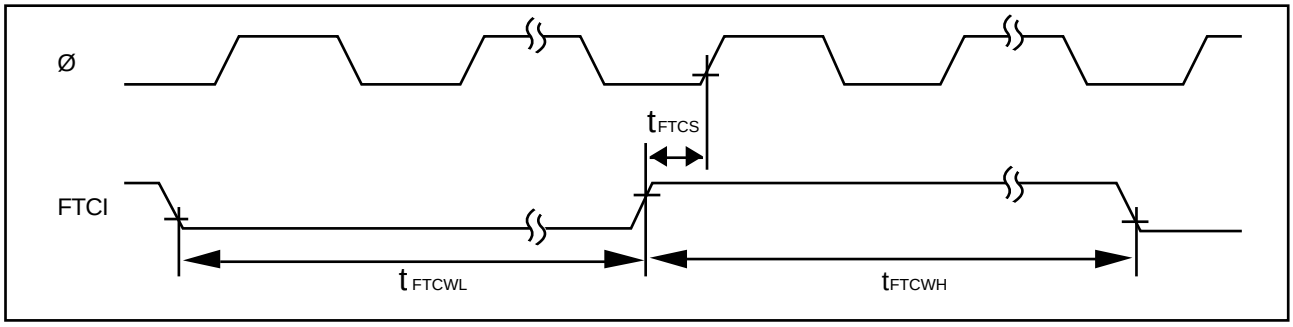



**Figure 15-12. External Clock Input Timing for Free-Running Timer**

## 15.3.4 8-Bit Timer Timing

### (1) 8-Bit Timer Output Timing

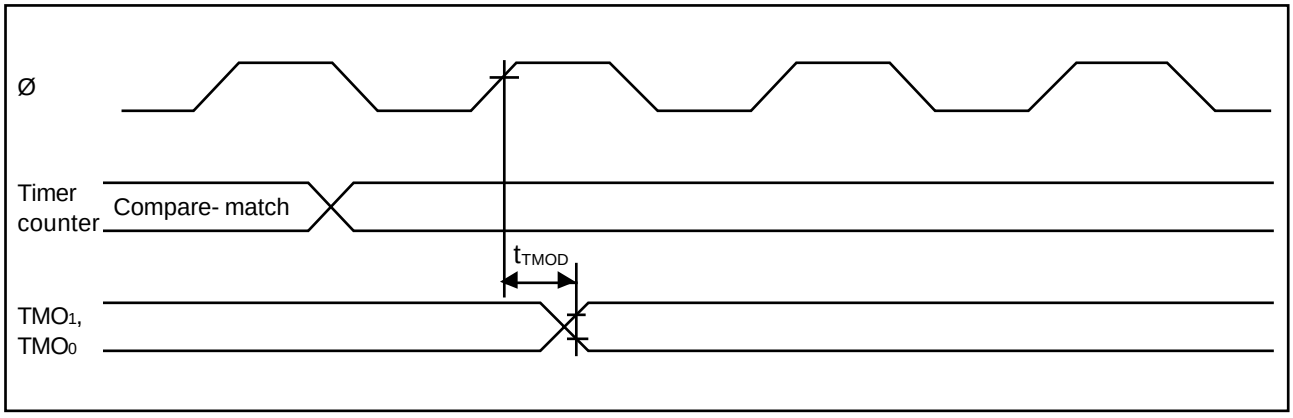



**Figure 15-13. 8-Bit Timer Output Timing**

### (2) 8-Bit Timer Clock Input Timing

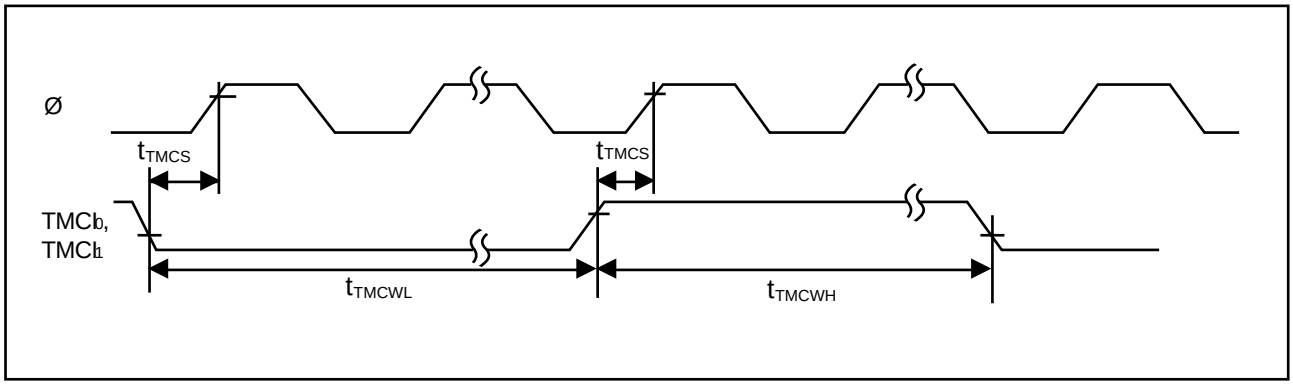



**Figure 15-14. 8-Bit Timer Clock Input Timing**

### (3) 8-Bit Timer Reset Input Timing

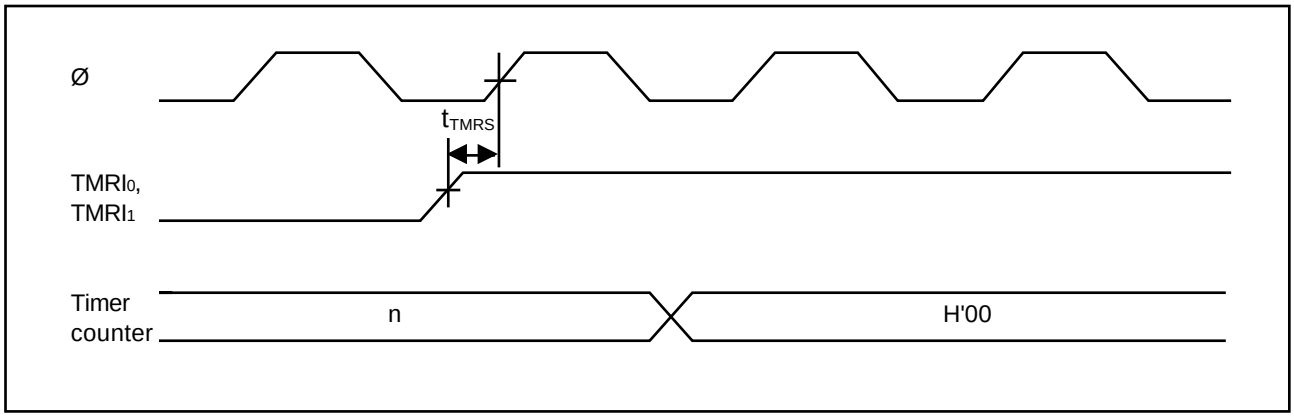



**Figure 15-15. 8-Bit Timer Reset Input Timing**

## 15.3.5 Serial Communication Interface Timing

### (1) SCI Input/Output Timing

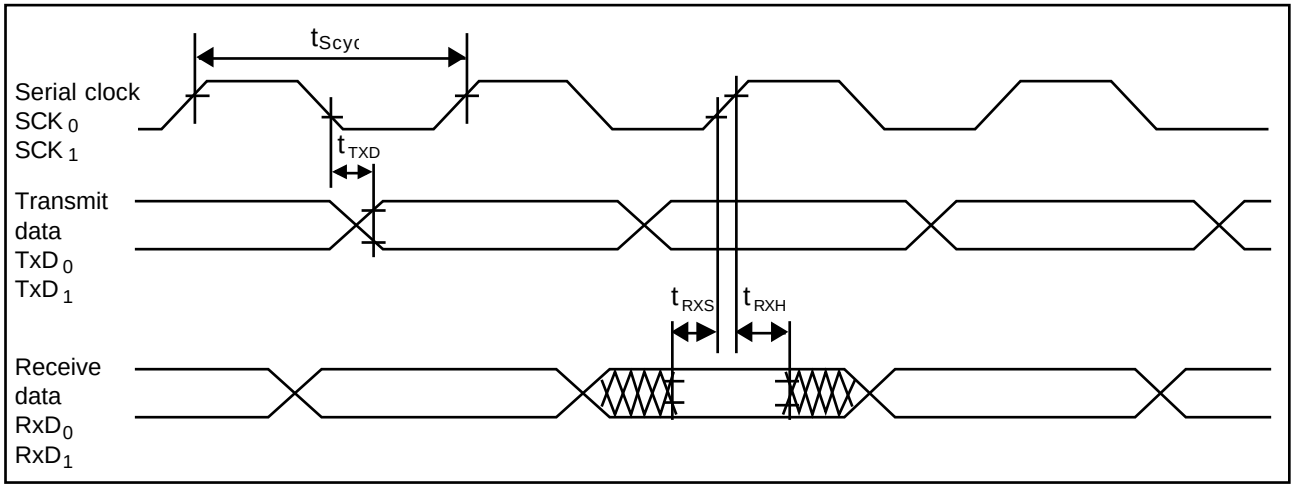



**Figure 15-16. SCI Input/Output Timing (Synchronous Mode)**

### (2) SCI Input Clock Timing

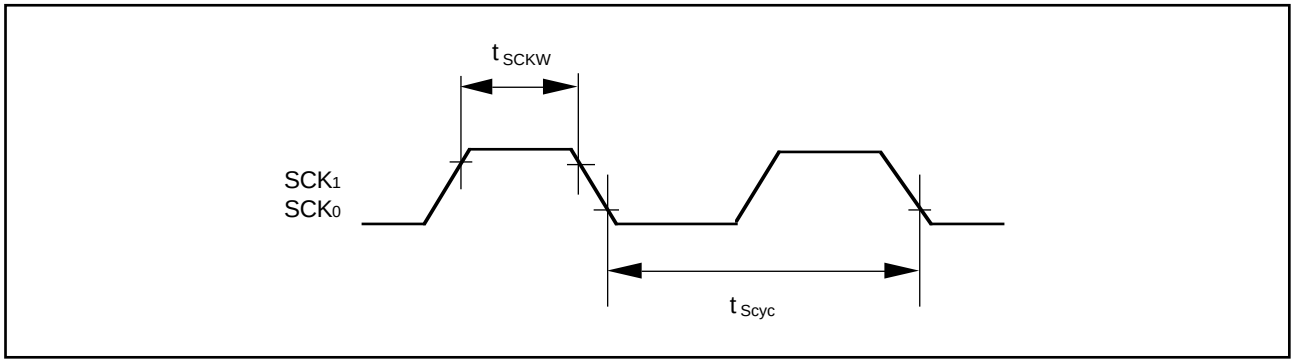



**Figure 15-17. SCI Input Clock Timing**

### 15.3.6 I/O Port Timing

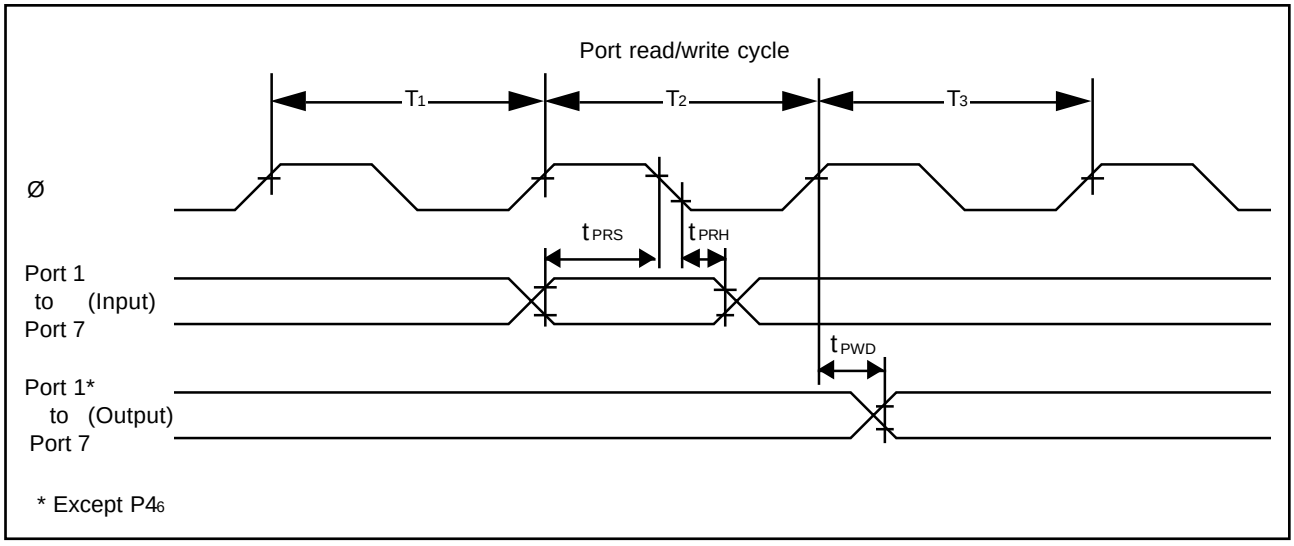



**Figure 15-18. I/O Port Input/Output Timing**

### 15.3.7 Parallel Handshake Interface Timing

### (1) Input Strobe Input Timing

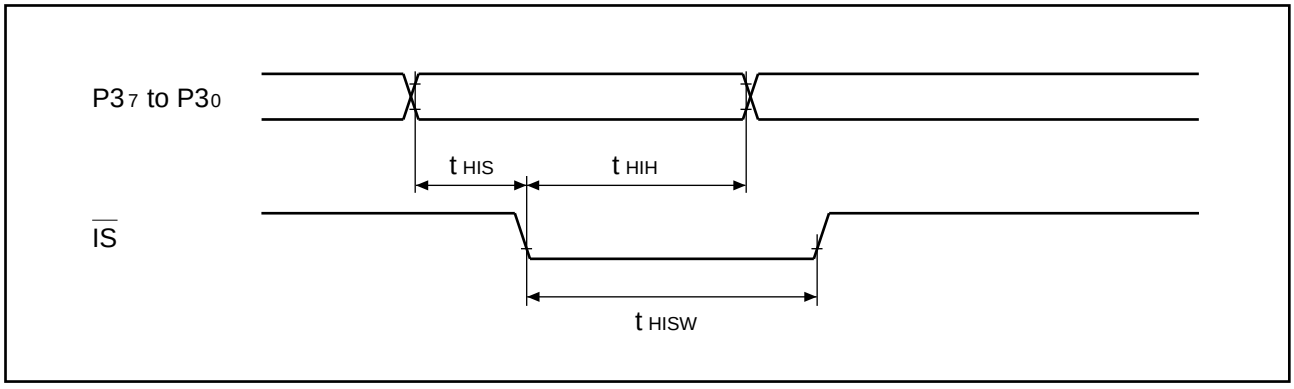



**Figure 15-19. Input Strobe Input Timing**

### (2) Output Strobe Output Timing

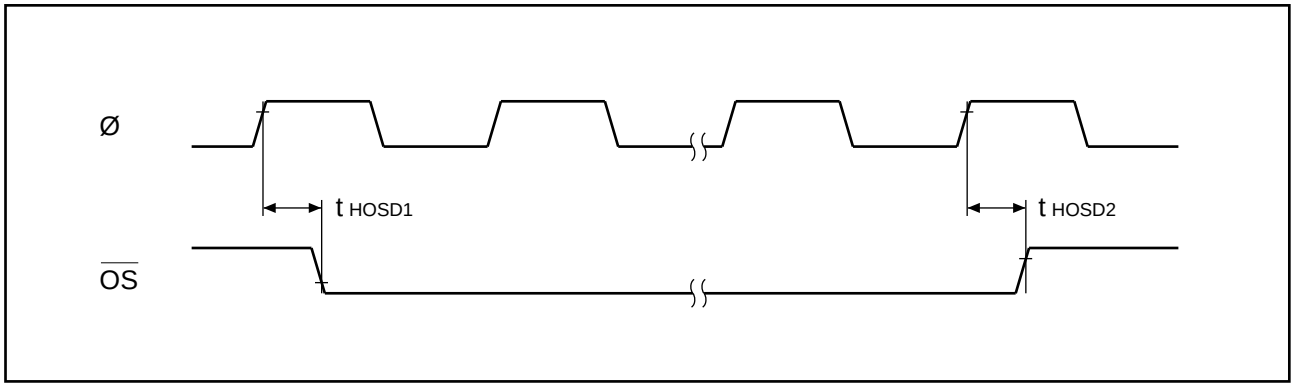



**Figure 15-20. Output Strobe Output Timing**

### (3) Busy Output Timing

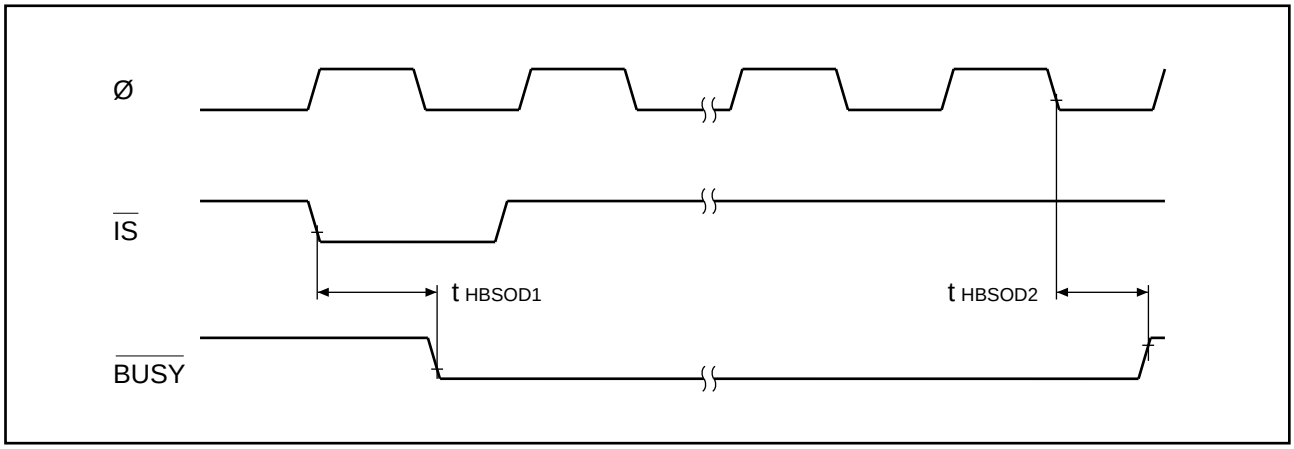



**Figure 15-21. Busy Output Timing**

# Appendix A. CPU Instruction Set

## A.1 Instruction Set List

**Operation Notation**

| | |
|---|---|
| Rd8/16 | General register (destination) (8 or 16 bits) |
| Rs8/16 | General register (source) (8 or 16 bits) |
| Rn8/16 | General register (8 or 16 bits) |
| CCR | Condition code register |
| N | N (negative) flag in CCR |
| Z | Z (zero) flag in CCR |
| V | V (overflow) flag in CCR |
| C | C (carry) flag in CCR |
| PC | Program counter |
| SP | Stack pointer |
| #xx:3/8/16 | Immediate data (3, 8, or 16 bits) |
| d:8/16 | Displacement (8 or 16 bits) |
| @aa:8/16 | Absolute address (8 or 16 bits) |
| + | Addition |
| – | Subtraction |
| × | Multiplication |
| ÷ | Division |
| ∧ | AND logical |
| ∨ | OR logical |
| ⊕ | Exclusive OR logical |
| → | Move |
| ¯ | Not |

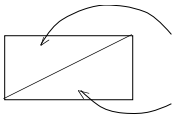
**Condition Code Notation**

| | |
|---|---|
| ↕ | Modified according to the instruction result |
| * | Undetermined (unpredictable) |
| 0 | Always cleared to "0" |
| — | Not affected by the instruction result |

## A.2 Operation Code Map

Table A-2 is a map of the operation codes contained in the first byte of the instruction code (bits 15 to 8 of the first instruction word).

Some pairs of instructions have identical first bytes. These instructions are differentiated by the first bit of the second byte (bit 7 of the first instruction word).



Instruction when first bit of byte 2 (bit 7 of first instruction word) is 0.
Instruction when first bit of byte 2 (bit 7 of first instruction word) is 1.

## Table A-2. Operation Code Map

| HI \ LO | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NOP | SLEEP | STC | LDC | ORC | XORC | ANDC | LDC | ADD | | INC | ADDS | MOV | | ADDX | DAA |
| 1 | SHLL / SHAL | SHLR / SHAR | ROTXL / ROTL | ROTXR / ROTR | OR | XOR | AND | NOT / NEG | SUB | | DEC | SUBS | CMP | | SUBX | DAS |
| 2 | MOV | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | |
| 4 | BRA*2 | BRN*2 | BHI | BLS | BCC*2 | BCS*2 | BNE | BEQ | BVC | BVS | BPL | BMI | BGE | BLT | BGT | BLE |
| 5 | MULXU | DIVXU | | | RTS | BSR | RTE | | | JMP | | | | JSR | | |
| 6 | BSET | BNOT | BCLR | BTST | | | | BST / BIST | MOV*1 | | | | | | | |
| 7 | | | | | BOR / BIOR | BXOR / BIXOR | BAND / BIAND | BLD / BILD | | MOV | | EEPMOV | Bit manipulation instruction | | | |
| 8 | ADD | | | | | | | | | | | | | | | |
| 9 | ADDX | | | | | | | | | | | | | | | |
| A | CMP | | | | | | | | | | | | | | | |
| B | SUBX | | | | | | | | | | | | | | | |
| C | OR | | | | | | | | | | | | | | | |
| D | XOR | | | | | | | | | | | | | | | |
| E | AND | | | | | | | | | | | | | | | |
| F | MOV | | | | | | | | | | | | | | | |

*1 The MOVFPE and MOVTPE instructions are identical to MOV instructions in the first byte and first bit of the second byte (bits 15 to 7 of the instruction word). The PUSH and POP instructions are identical in machine language to MOV instructions.

*2 The BT, BF, BHS, and BLO instructions are identical in machine language to BRA, BRN, BCC, and BCS, respectively.

## A.3 Number of States Required for Execution

The tables below can be used to calculate the number of states required for instruction execution. Table A-3 indicates the number of states required for each cycle (instruction fetch, branch address read, stack operation, byte data access, word data access, internal operation). Table A-4 indicates the number of cycles of each type occurring in each instruction. The total number of states required for execution of an instruction can be calculated from these two tables as follows:

$$\text{Execution states} = I \times S_I + J \times S_J + K \times S_K + L \times S_L + M \times S_M + N \times S_N$$

**Examples:** Mode 1 (on-chip ROM disabled), stack located in external memory, 1 wait state inserted in external memory access.

1. BSET #0, @FFC7
   From table A-4: I = L = 2, J = K = M = N= 0
   From table A-3: $S_I = 8$, $S_L = 3$
   Number of states required for execution: $2 \times 8 + 2 \times 3 = 22$

2. JSR @@30
   From table A-4: I = 2, J = K = 1, L = M = N = 0
   From table A-3: $S_I = S_J = S_K = 8$
   Number of states required for execution: $2 \times 8 + 1 \times 8 + 1 \times 8 = 32$

**Table A-3. Number of States Taken by Each Cycle in Instruction Execution**

| **Execution status (instruction cycle)** | | **Access location** | | |
|---|---|---|---|---|
| | | **On-chip memory** | **On-chip reg. field** | **External memory** |
| Instruction fetch | $S_I$ | 2 | 6 | 6 + 2m |
| Branch address read | $S_J$ | | | |
| Stack operation | $S_K$ | | | |
| Byte data access | $S_L$ | | 3 | 3 + m (note 2) |
| Word data access | $S_M$ | | 6 | 6 + 2m |
| Internal operation | $S_N$ | | 2 | |

Notes: 1. m: Number of wait states inserted in access to external device.
2. The byte data access cycle to an external device by the MOVFPE and MOVTPE instructions requires 9 to 16 states since it is synchronized with the E clock. See section 13, E-Clock Interface for timing details.

## Table A-4. Number of Cycles in Each Instruction

| Instruction | Mnemonic | Instruction fetch I | Branch addr. read J | Stack operation K | Byte data access L | Word data access M | Internal operation N |
|---|---|---|---|---|---|---|---|
| ADD | ADD.B #xx:8, Rd | 1 | | | | | |
| | ADD.B Rs, Rd | 1 | | | | | |
| | ADD.W Rs, Rd | 1 | | | | | |
| ADDS | ADDS.W #1/2, Rd | 1 | | | | | |
| ADDX | ADDX.B #xx:8, Rd | 1 | | | | | |
| | ADDX.B Rs, Rd | 1 | | | | | |
| AND | AND.B #xx:8, Rd | 1 | | | | | |
| | AND.B Rs, Rd | 1 | | | | | |
| ANDC | ANDC #xx:8, CCR | 1 | | | | | |
| BAND | BAND #xx:3, Rd | 1 | | | | | |
| | BAND #xx:3, @Rd | 2 | | | 1 | | |
| | BAND #xx:3, @aa:8 | 2 | | | 1 | | |
| Bcc | BRA d:8 (BT d:8) | 2 | | | | | |
| | BRN d:8 (BF d:8) | 2 | | | | | |
| | BHI d:8 | 2 | | | | | |
| | BLS d:8 | 2 | | | | | |
| | BCC d:8 (BHS d:8) | 2 | | | | | |
| | BCS d:8 (BLO d:8) | 2 | | | | | |
| | BNE d:8 | 2 | | | | | |
| | BEQ d:8 | 2 | | | | | |
| | BVC d:8 | 2 | | | | | |
| | BVS d:8 | 2 | | | | | |
| | BPL d:8 | 2 | | | | | |
| | BMI d:8 | 2 | | | | | |
| | BGE d:8 | 2 | | | | | |
| | BLT d:8 | 2 | | | | | |
| | BGT d:8 | 2 | | | | | |
| | BLE d:8 | 2 | | | | | |
| BCLR | BCLR #xx:3, Rd | 1 | | | | | |
| | BCLR #xx:3, @Rd | 2 | | | 2 | | |
| | BCLR #xx:3, @aa:8 | 2 | | | 2 | | |
| | BCLR Rn, Rd | 1 | | | | | |
| | BCLR Rn, @Rd | 2 | | | 2 | | |
| | BCLR Rn, @aa:8 | 2 | | | 2 | | |

**Note:** Blank entries are all zero.

**Table A-4. Number of Cycles in Each Instruction (cont.)**

| Instruction | Mnemonic | Instruction fetch I | Branch addr. read J | Stack operation K | Byte data access L | Word data access M | Internal operation N |
|---|---|---|---|---|---|---|---|
| BIAND | BIAND #xx:3, Rd | 1 | | | | | |
| | BIAND #xx:3, @Rd | 2 | | | 1 | | |
| | BIAND #xx:3, @aa:8 | 2 | | | 1 | | |
| BILD | BILD #xx:3, Rd | 1 | | | | | |
| | BILD #xx:3, @Rd | 2 | | | 1 | | |
| | BILD #xx:3, @aa:8 | 2 | | | 1 | | |
| BIOR | BIOR #xx:3 Rd | 1 | | | | | |
| | BIOR #xx:3 @Rd | 2 | | | 1 | | |
| | BIOR #xx:3 @aa:8 | 2 | | | 1 | | |
| BIST | BIST #xx:3, Rd | 1 | | | | | |
| | BIST #xx:3, @Rd | 2 | | | 2 | | |
| | BIST #xx:3, @aa:8 | 2 | | | 2 | | |
| BIXOR | BIXOR #xx:3, Rd | 1 | | | | | |
| | BIXOR #xx:3, @Rd | 2 | | | 1 | | |
| | BIXOR #xx:3, @aa:8 | 2 | | | 1 | | |
| BLD | BLD #xx:3, Rd | 1 | | | | | |
| | BLD #xx:3, @Rd | 2 | | | 1 | | |
| | BLD #xx:3, @aa:8 | 2 | | | 1 | | |
| BNOT | BNOT #xx:3, Rd | 1 | | | | | |
| | BNOT #xx:3, @Rd | 2 | | | 2 | | |
| | BNOT #xx:3, @aa:8 | 2 | | | 2 | | |
| | BNOT Rn, Rd | 1 | | | | | |
| | BNOT Rn, @Rd | 2 | | | 2 | | |
| | BNOT Rn, @aa:8 | 2 | | | 2 | | |
| BOR | BOR #xx:3, Rd | 1 | | | | | |
| | BOR #xx:3, @Rd | 2 | | | 1 | | |
| | BOR #xx:3, @aa:8 | 2 | | | 1 | | |
| BSET | BSET #xx:3, Rd | 1 | | | | | |
| | BSET #xx:3, @Rd | 2 | | | 2 | | |
| | BSET #xx:3, @aa:8 | 2 | | | 2 | | |
| | BSET Rn, Rd | 1 | | | | | |
| | BSET Rn, @Rd | 2 | | | 2 | | |
| | BSET Rn, @aa:8 | 2 | | | 2 | | |

**Note:** Blank entries are all zero.

**Table A-4. Number of Cycles in Each Instruction (cont.)**

| Instruction | Mnemonic | Instruction fetch I | Branch addr. read J | Stack operation K | Byte data access L | Word data access M | Internal operation N |
|---|---|---|---|---|---|---|---|
| BSR | BSR d:8 | 2 | | 1 | | | |
| BST | BST #xx:3, Rd | 1 | | | | | |
| | BST #xx:3, @Rd | 2 | | | 2 | | |
| | BST #xx:3, @aa:8 | 2 | | | 2 | | |
| BTST | BTST #xx:3, Rd | 1 | | | | | |
| | BTST #xx:3, @Rd | 2 | | | 1 | | |
| | BTST #xx:3, @aa:8 | 2 | | | 1 | | |
| | BTST Rn, Rd | 1 | | | | | |
| | BTST Rn, @Rd | 2 | | | 1 | | |
| | BTST Rn, @aa:8 | 2 | | | 1 | | |
| BXOR | BXOR #xx:3, Rd | 1 | | | | | |
| | BXOR #xx:3, @Rd | 2 | | | 1 | | |
| | BXOR #xx:3, @aa:8 | 2 | | | 1 | | |
| CMP | CMP.B #xx:8, Rd | 1 | | | | | |
| | CMP.B Rs, Rd | 1 | | | | | |
| | CMP.W Rs, Rd | 1 | | | | | |
| DAA | DAA.B Rd | 1 | | | | | |
| DAS | DAS.B Rd | 1 | | | | | |
| DEC | DEC.B Rd | 1 | | | | | |
| DIVXU | DIVXU.B Rs, Rd | 1 | | | | | 6 |
| EEPMOV | EEPMOV | 2 | | | 2n+2*1 | | |
| INC | INC.B Rd | 1 | | | | | |
| JMP | JMP @Rn | 2 | | | | | |
| | JMP @aa:16 | 2 | | | | | 1 |
| | JMP @@aa:8 | 2 | 1 | | | | 1 |
| JSR | JSR @Rn | 2 | | 1 | | | |
| | JSR @aa:16 | 2 | | 1 | | | 1 |
| | JSR @@aa:8 | 2 | 1 | 1 | | | |
| LDC | LDC #xx:8, CCR | 1 | | | | | |
| | LDC Rs, CCR | 1 | | | | | |
| MOV | MOV.B #xx:8, Rd | 1 | | | | | |
| | MOV.B Rs, Rd | 1 | | | | | |
| | MOV.B @Rs, Rd | 1 | | | 1 | | |
| | MOV.B @(d:16,Rs), Rd | 2 | | | 1 | | |

**Note:** Blank entries are all zero.

**Table A-4. Number of Cycles in Each Instruction (cont.)**

| Instruction | Mnemonic | Instruction fetch I | Branch addr. read J | Stack operation K | Byte data access L | Word data access M | Internal operation N |
|---|---|---|---|---|---|---|---|
| MOV | MOV.B @Rs+, Rd | 1 | | | 1 | | 1 |
| | MOV.B @aa:8, Rd | 1 | | | 1 | | |
| | MOV.B @aa:16, Rd | 2 | | | 1 | | |
| | MOV.B Rs, @Rd | 1 | | | 1 | | |
| | MOV.B Rs, @(d:16, Rd) | 2 | | | 1 | | |
| | MOV.B Rs, @–Rd | 1 | | | 1 | | 1 |
| | MOV.B Rs, @aa:8 | 1 | | | 1 | | |
| | MOV.B Rs, @aa:16 | 2 | | | 1 | | |
| | MOV.W #xx:16, Rd | 2 | | | | | |
| | MOV.W Rs, Rd | 1 | | | | | |
| | MOV.W @Rs, Rd | 1 | | | | 1 | |
| | MOV.W @(d:16, Rs), Rd | 2 | | | | 1 | |
| | MOV.W @Rs+, Rd | 1 | | | | 1 | 1 |
| | MOV.W @aa:16, Rd | 2 | | | | 1 | |
| | MOV.W Rs, @Rd | 1 | | | | 1 | |
| | MOV.W Rs, @(d:16, Rd) | 2 | | | | 1 | |
| | MOV.W Rs, @–Rd | 1 | | | | 1 | 1 |
| | MOV.W Rs, @aa:16 | 2 | | | | 1 | |
| MOVFPE | MOVFPE @aa:16, Rd | 2 | | | 1*2 | | |
| MOVTPE | MOVTPE.Rs, @aa:16 | 2 | | | 1*2 | | |
| MULXU | MULXU.Rs, Rd | 1 | | | | | 6 |
| NEG | NEG.B Rd | 1 | | | | | |
| NOP | NOP | 1 | | | | | |
| NOT | NOT.B Rd | 1 | | | | | |
| OR | OR.B #xx:8, Rd | 1 | | | | | |
| | OR.B Rs, Rd | 1 | | | | | |
| ORC | ORC #xx:8, CCR | 1 | | | | | |
| ROTL | ROTL.B Rd | 1 | | | | | |
| ROTR | ROTR.B Rd | 1 | | | | | |
| ROTXL | ROTXL.B Rd | 1 | | | | | |
| ROTXR | ROTXR.B Rd | 1 | | | | | |
| RTE | RTE | 2 | | 2 | | | 1 |
| RTS | RTS | 2 | | 1 | | | 1 |

**Note:** Blank entries are all zero.

**Table A-4. Number of Cycles in Each Instruction (cont.)**

| Instruction | Mnemonic | Instruction fetch I | Branch addr. read J | Stack operation K | Byte data access L | Word data access M | Internal operation N |
|---|---|---|---|---|---|---|---|
| SHAL | SHAL.B Rd | 1 | | | | | |
| SHAR | SHAR.B Rd | 1 | | | | | |
| SHLL | SHLL.B Rd | 1 | | | | | |
| SHLR | SHLR.B Rd | 1 | | | | | |
| SLEEP | SLEEP | 1 | | | | | |
| STC | STC CCR, Rd | 1 | | | | | |
| SUB | SUB.B Rs, Rd | 1 | | | | | |
| | SUB.W Rs, Rd | 1 | | | | | |
| SUBS | SUBS.W #1/2, Rd | 1 | | | | | |
| SUBX | SUBX.B #xx:8, Rd | 1 | | | | | |
| | SUBX.B Rs, Rd | 1 | | | | | |
| XOR | XOR.B #xx:8, Rd | 1 | | | | | |
| | XOR.B Rs, Rd | 1 | | | | | |
| XORC | XORC #xx:8, CCR | 1 | | | | | |

**Notes:**

*1 n: Initial value in R4L. Source and destination are accessed n + 1 times each.

*2 Data access requires 9 to 16 states.

Blank entries are all zero.

# Appendix B. Register Field

## B.1 Register Addresses and Bit Names

| Addr. (last byte) | Register name | Bit names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'80 | | | | | | | | | | External addresses (in expanded modes) |
| H'81 | | | | | | | | | | |
| H'82 | | | | | | | | | | |
| H'83 | | | | | | | | | | |
| H'84 | | | | | | | | | | |
| H'85 | | | | | | | | | | |
| H'86 | | | | | | | | | | |
| H'87 | | | | | | | | | | |
| H'88 | | | | | | | | | | |
| H'89 | | | | | | | | | | |
| H'8A | | | | | | | | | | |
| H'8B | | | | | | | | | | |
| H'8C | | | | | | | | | | |
| H'8D | | | | | | | | | | |
| H'8E | | | | | | | | | | |
| H'8F | | | | | | | | | | |
| H'90 | TCR | ICIE | OCIEB | OCIEA | OVIE | OEB | OEA | CKS1 | CKS0 | FRT |
| H'91 | TCSR | ICF | OCFB | OCFA | OVF | OLVLB | OLVLA | IEDG | CCLRA | |
| H'92 | FRC (H) | | | | | | | | | |
| H'93 | FRC (L) | | | | | | | | | |
| H'94 | OCRA (H) | | | | | | | | | |
| H'95 | OCRA (L) | | | | | | | | | |
| H'96 | OCRB (H) | | | | | | | | | |
| H'97 | OCRB (L) | | | | | | | | | |
| H'98 | ICR (H) | | | | | | | | | |
| H'99 | ICR (L) | | | | | | | | | |
| H'9A | | | | | | | | | | |
| H'9B | | | | | | | | | | |
| H'9C | | | | | | | | | | |
| H'9D | | | | | | | | | | |
| H'9E | | | | | | | | | | |
| H'9F | | | | | | | | | | |

**Notes:** FRT: 16-Bit Free-Running Timer (Continued on next page)

(Continued from previous page)

| Addr. (last byte) | Register name | Bit names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'A0 | | | | | | | | | | External addresses (in expanded modes) |
| H'A1 | | | | | | | | | | |
| H'A2 | | | | | | | | | | |
| H'A3 | | | | | | | | | | |
| H'A4 | | | | | | | | | | |
| H'A5 | | | | | | | | | | |
| H'A6 | | | | | | | | | | |
| H'A7 | | | | | | | | | | |
| H'A8 | | | | | | | | | | |
| H'A9 | | | | | | | | | | |
| H'AA | | | | | | | | | | |
| H'AB | | | | | | | | | | |
| H'AC | | | | | | | | | | |
| H'AD | | | | | | | | | | |
| H'AE | | | | | | | | | | |
| H'AF | | | | | | | | | | |
| H'B0 | P1DDR | $P1_7$DDR | $P1_6$DDR | $P1_5$DDR | $P1_4$DDR | $P1_3$DDR | $P1_2$DDR | $P1_1$DDR | $P1_0$DDR | Port 1 |
| H'B1 | P2DDR | $P2_7$DDR | $P2_6$DDR | $P2_5$DDR | $P2_4$DDR | $P2_3$DDR | $P2_2$DDR | $P2_1$DDR | $P2_0$DDR | Port 2 |
| H'B2 | P1DR | $P1_7$ | $P1_6$ | $P1_5$ | $P1_4$ | $P1_3$ | $P1_2$ | $P1_1$ | $P1_0$ | Port 1 |
| H'B3 | P2DR | $P2_7$ | $P2_6$ | $P2_5$ | $P2_4$ | $P2_3$ | $P2_2$ | $P2_1$ | $P2_0$ | Port 2 |
| H'B4 | P3DDR | $P3_7$DDR | $P3_6$DDR | $P3_5$DDR | $P3_4$DDR | $P3_3$DDR | $P3_2$DDR | $P3_1$DDR | $P3_0$DDR | Port 3 |
| H'B5 | P4DDR | $P4_7$DDR | $P4_6$DDR | $P4_5$DDR | $P4_4$DDR | $P4_3$DDR | $P4_2$DDR | $P4_1$DDR | $P4_0$DDR | Port 4 |
| H'B6 | P3DR | $P3_7$ | $P3_6$ | $P3_5$ | $P3_4$ | $P3_3$ | $P3_2$ | $P3_1$ | $P3_0$ | Port 3 |
| H'B7 | P4DR | $P4_7$ | $P4_6$ | $P4_5$ | $P4_4$ | $P4_3$ | $P4_2$ | $P4_1$ | $P4_0$ | Port 4 |
| H'B8 | P5DDR | — | — | $P5_5$DDR | $P5_4$DDR | $P5_3$DDR | $P5_2$DDR | $P5_1$DDR | $P5_0$DDR | Port 5 |
| H'B9 | P6DDR | — | $P6_6$DDR | $P6_5$DDR | $P6_4$DDR | $P6_3$DDR | $P6_2$DDR | $P6_1$DDR | $P6_0$DDR | Port 6 |
| H'BA | P5DR | — | — | $P5_5$ | $P5_4$ | $P5_3$ | $P5_2$ | $P5_1$ | $P5_0$ | Port 5 |
| H'BB | P6DR | — | $P6_6$ | $P6_5$ | $P6_4$ | $P6_3$ | $P6_2$ | $P6_1$ | $P6_0$ | Port 6 |
| H'BC | P7DDR | $P7_7$DDR | $P7_6$DDR | $P7_5$DDR | $P7_4$DDR | $P7_3$DDR | $P7_2$DDR | $P7_1$DDR | $P7_0$DDR | Port 7 |
| H'BD | — | — | — | — | — | — | — | — | — | — |
| H'BE | P7DR | $P7_7$ | $P7_6$ | $P7_5$ | $P7_4$ | $P7_3$ | $P7_2$ | $P7_1$ | $P7_0$ | Port 7 |
| H'BF | — | — | — | — | — | — | — | — | — | — |

(Continued on next page)

(Continued from preceding page)

| Addr. (last byte) | Register name | Bit names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'C0 | | | | | | | | | | |
| H'C1 | | | | | | | | | | |
| H'C2 | | | | | | | | | | |
| H'C3 | | | | | | | | | | |
| H'C4 | SYSCR | SSBY | STS2 | STS1 | STS0 | — | NMIEG | — | RAME | System |
| H'C5 | MDCR | — | — | — | — | — | — | MDS1 | MDS0 | control |
| H'C6 | ISCR | — | $IRQ_2EG$ | $IRQ_1EG$ | $IRQ_0EG$ | — | $IRQ_2SC$ | $IRQ_1SC$ | $IRQ_0SC$ | |
| H'C7 | IER | — | — | — | — | — | $IRQ_2E$ | $IRQ_1E$ | $IRQ_0E$ | |
| H'C8 | TCR | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 | TMR0 |
| H'C9 | TCSR | CMFB | CMFA | OVF | — | OS3 | OS2 | OS1 | OS0 | |
| H'CA | TCORA | | | | | | | | | |
| H'CB | TCORB | | | | | | | | | |
| H'CC | TCNT | | | | | | | | | |
| H'CD | — | — | — | — | — | — | — | — | — | |
| H'CE | — | — | — | — | — | — | — | — | — | |
| H'CF | — | — | — | — | — | — | — | — | — | |
| H'D0 | TCR | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 | TMR1 |
| H'D1 | TCSR | CMFB | CMFA | OVF | — | OS3 | OS2 | OS1 | OS0 | |
| H'D2 | TCORA | | | | | | | | | |
| H'D3 | TCORB | | | | | | | | | |
| H'D4 | TCNT | | | | | | | | | |
| H'D5 | — | — | — | — | — | — | — | — | — | |
| H'D6 | — | — | — | — | — | — | — | — | — | |
| H'D7 | — | — | — | — | — | — | — | — | — | |
| H'D8 | SMR | $C/\overline{A}$ | CHR | PE | $O/\overline{E}$ | STOP | — | CKS1 | CKS0 | SCI0 |
| H'D9 | BRR | | | | | | | | | |
| H'DA | SCR | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 | |
| H'DB | TDR | | | | | | | | | |
| H'DC | SSR | TDRE | RDRF | ORER | FER | PER | — | — | — | |
| H'DD | RDR | | | | | | | | | |
| H'DE | — | — | — | — | — | — | — | — | — | |
| H'DF | — | — | — | — | — | — | — | — | — | |

(Continued on next page)

**Notes:** TMR1: 8-Bit Timer channel 0
TMR1: 8-Bit Timer channel 1
SCI0: Serial Communication Interface channel 0

(Continued from preceding page)

| Addr. (last byte) | Register name | Bit names | | | | | | | | Module |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| H'E0 | SMR | C/$\overline{A}$ | CHR | PE | O/$\overline{E}$ | STOP | — | CKS1 | CKS0 | SCI1 |
| H'E1 | BRR | | | | | | | | | |
| H'E2 | SCR | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 | |
| H'E3 | TDR | | | | | | | | | |
| H'E4 | SSR | TDRE | RDRF | ORER | FER | PER | — | — | — | |
| H'E5 | RDR | | | | | | | | | |
| H'E6 | — | — | — | — | — | — | — | — | — | |
| H'E7 | — | — | — | — | — | — | — | — | — | |
| H'E8 | | | | | | | | | | |
| H'E9 | | | | | | | | | | |
| H'EA | | | | | | | | | | |
| H'EB | | | | | | | | | | |
| H'EC | | | | | | | | | | |
| H'ED | | | | | | | | | | |
| H'EE | | | | | | | | | | |
| H'EF | | | | | | | | | | |
| H'F0 | | | | | | | | | | |
| H'F1 | | | | | | | | | | |
| H'F2 | | | | | | | | | | |
| H'F3 | | | | | | | | | | |
| H'F4 | | | | | | | | | | |
| H'F5 | | | | | | | | | | |
| H'F6 | | | | | | | | | | |
| H'F7 | | | | | | | | | | |
| H'F8 | | | | | | | | | | |
| H'F9 | | | | | | | | | | |
| H'FA | | | | | | | | | | |
| H'FB | | | | | | | | | | |
| H'FC | | | | | | | | | | |
| H'FD | | | | | | | | | | |
| H'FE | HCSR | ISF | ISIE | OSE | OSS | LTE | BSE | — | — | Handshaking |
| H'FF | FNCR | — | — | — | — | — | — | NCS1 | NCS0 | FRT |

**Note:** SCI1: Serial Communication Interface channel 1
FRT: 16-bit Free-Running Timer

## B.2 Register Descriptions

**TCR—Timer Control Register** **H'FF90** **FRT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ICIE | OCIEB | OCIEA | OVIE | OEB | OEA | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Bits 1 and 0 – Clock Select (CKS1 and CKS0)**

| | | |
|---|---|---|
| 0 | 0 | Ø/2 Internal clock source |
| 0 | 1 | Ø/8 Internal clock source |
| 1 | 0 | Ø/32 Internal clock source |
| 1 | 1 | External clock source (rising edge) |

**Output Enable A (OEA)**

| | |
|---|---|
| 0 | Output compare A output is disabled. |
| 1 | Output compare A output is enabled. |

**Output Enable B (OEB)**

| | |
|---|---|
| 0 | Output compare B output is disabled. |
| 1 | Output compare B output is enabled. |

**Timer overflow Interrupt Enable**

| | |
|---|---|
| 0 | Timer overflow interrupt request is disabled. |
| 1 | Timer overflow interrupt request is enabled. |

**Output Compare Interrupt A Enable**

| | |
|---|---|
| 0 | Output compare interrupt request A is disabled. |
| 1 | Output compare interrupt request A is enabled. |

**Output Compare Interrupt B Enable**

| | |
|---|---|
| 0 | Output compare interrupt request B is disabled. |
| 1 | Output compare interrupt request B is enabled. |

**Input Capture Interrupt Enable**

| | |
|---|---|
| 0 | Input capture interrupt request is disabled. |
| 1 | Input capture interrupt request is enabled. |

**TCSR—Timer Control/Status Register H'FF91 FRT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ICF | OCFB | OCFA | OVF | OLVLB | OLVLA | IEDG | CCLRA |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/W | R/W | R/W | R/W |

**Counter Clear A**

| | |
|---|---|
| 0 | FRC is not cleared. |
| 1 | FRC is cleared at compare-match A. |

**Input Edge Select**

| | |
|---|---|
| 0 | Falling edge of FTI is valid. |
| 1 | Rising edge of FTI is valid. |

**Output Level A**

| | |
|---|---|
| 0 | Compare-match A causes 0 output. |
| 1 | Compare-match A causes 1 output. |

**Output Level B**

| | |
|---|---|
| 0 | Compare-match B causes 0 output. |
| 1 | Compare-match B causes 1 output. |

**Timer Overflow Flag**

| | |
|---|---|
| 0 | Cleared by reading OVF = 1, then writing 0. |
| 1 | Set when FRC changes from H'FFFF to H'0000. |

**Output Compare Flag A**

| | |
|---|---|
| 0 | Cleared by reading OCFA = 1, then writing 0. |
| 1 | Set when FRC = OCRA. |

**Output Compare Flag B**

| | |
|---|---|
| 0 | Cleared by reading OCFB = 1, then writing 0. |
| 1 | Set when FRC = OCRB. |

**Input Capture Flag**

| | |
|---|---|
| 0 | Cleared by reading ICF = 1, then writing 0. |
| 1 | Set when FTI input causes FRC to be copied to ICR. |

* Software can write a 0 in bits 7 to 4 to clear the flags, but cannot write a 1 in these bits.

**FRC (H and L)—Free-Running Counter** **H’FF92, H’FF93** **FRT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Count value

---

**OCRA (H and L)—Output Compare Register A** **H’FF94, H’FF95** **FRT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Continually compared with FRC. OCFA is set to 1 when OCRA = FRC.

---

**OCRB (H and L)—Output Compare Register B** **H’FF96, H’FF97** **FRT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Continually compared with FRC. OCFB is set to 1 when OCRB = FRC.

---

**ICR (H and L)—Input Capture Register** **H’FF98, H’FF99** **FRT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

Contains FRC count captured on FTI input.

**P1DDR—Port 1 Data Direction Register** **H'FFB0** **Port 1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P1_7$DDR | $P1_6$DDR | $P1_5$DDR | $P1_4$DDR | $P1_3$DDR | $P1_2$DDR | $P1_1$DDR | $P1_0$DDR |
| Mode 1 | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | — | — | — | — |
| Modes 2 and 3 | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

**Port 1 Input/Output Control**

| | |
|---|---|
| 0 | Input port |
| 1 | Output port |

---

**P1DR—Port 1 Data Register** **H'FFB2** **Port 1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P1_7$ | $P1_6$ | $P1_5$ | $P1_4$ | $P1_3$ | $P1_2$ | $P1_1$ | $P1_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

---

**P2DDR—Port 2 Data Direction Register** **H'FFB1** **Port 2**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P2_7$DDR | $P2_6$DDR | $P2_5$DDR | $P2_4$DDR | $P2_3$DDR | $P2_2$DDR | $P2_1$DDR | $P2_0$DDR |
| Mode 1 | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | — | — | — | — | — | — | — | — |
| Modes 2 and 3 | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

**Port 2 Input/Output Control**

| | |
|---|---|
| 0 | Input port |
| 1 | Output port |

**P2DR—Port 2 Data Register** **H'FFB3** **Port 2**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P2_7$ | $P2_6$ | $P2_5$ | $P2_4$ | $P2_3$ | $P2_2$ | $P2_1$ | $P2_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

---

**P3DDR—Port 3 Data Direction Register** **H'FFB4** **Port 3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P3_7$DDR | $P3_6$DDR | $P3_5$DDR | $P3_4$DDR | $P3_3$DDR | $P3_2$DDR | $P3_1$DDR | $P3_0$DDR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

**Port 3 Input/Output Control**

| | |
|---|---|
| 0 | Input port |
| 1 | Output port |

---

**P3DR—Port 3 Data Register** **H'FFB6** **Port 3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P3_7$ | $P3_6$ | $P3_5$ | $P3_4$ | $P3_3$ | $P3_2$ | $P3_1$ | $P3_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**P4DDR—Port 4 Data Direction Register** **H'FFB5** **Port 4**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P4_7DDR$ | $P4_6DDR$ | $P4_5DDR$ | $P4_4DDR$ | $P4_3DDR$ | $P4_2DDR$ | $P4_1DDR$ | $P4_0DDR$ |
| Modes 1 and 2 | | | | | | | | |
| Initial value | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |
| Mode 3 | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

**Port 4 Input/Output Control**

| | |
|---|---|
| 0 | Input port |
| 1 | Output port |

---

**P4DR—Port 4 Data Register** **H'FFB7** **Port 4**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P4_7$ | $P4_6$ | $P4_5$ | $P4_4$ | $P4_3$ | $P4_2$ | $P4_1$ | $P4_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

---

**P5DDR—Port 5 Data Direction Register** **H'FFB8** **Port 5**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | $P5_5DDR$ | $P5_4DDR$ | $P5_3DDR$ | $P5_2DDR$ | $P5_1DDR$ | $P5_0DDR$ |
| Initial value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | W | W | W | W | W | W |

**Port 5 Input/Output Control**

| | |
|---|---|
| 0 | Input port |
| 1 | Output port |

**P5DR—Port 5 Data Register** **H'FFBA** **Port 5**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | $P5_5$ | $P5_4$ | $P5_3$ | $P5_2$ | $P5_1$ | $P5_0$ |
| Initial value | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | — | R/W | R/W | R/W | R/W | R/W | R/W |

---

**P6DDR—Port 6 Data Direction Register** **H'FFB9** **Port 6**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | $P6_6$DDR | $P6_5$DDR | $P6_4$DDR | $P6_3$DDR | $P6_2$DDR | $P6_1$DDR | $P6_0$DDR |
| Initial value | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | W | W | W | W | W | W | W |

**Port 6 Input/Output Control**

| | |
|---|---|
| 0 | Input port |
| 1 | Output port |

---

**P6DR—Port 6 Data Register** **H'FFBB** **Port 6**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | $P6_6$ | $P6_5$ | $P6_4$ | $P6_3$ | $P6_2$ | $P6_1$ | $P6_0$ |
| Initial value | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

---

**P7DDR—Port 7 Data Direction Register** **H'FFBC** **Port 7**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P7_7$DDR | $P7_6$DDR | $P7_5$DDR | $P7_4$DDR | $P7_3$DDR | $P7_2$DDR | $P7_1$DDR | $P7_0$DDR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

**Port 7 Input/Output Control**

| | |
|---|---|
| 0 | Input port |
| 1 | Output port |

**P7DR—Port 7 Data Register** **H'FFBE** **Port 7**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P7_7$ | $P7_6$ | $P7_5$ | $P7_4$ | $P7_3$ | $P7_2$ | $P7_1$ | $P7_0$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

---

**SYSCR—System Control Register** **H'FFC4** **System Control**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSBY | STS2 | STS1 | STS0 | — | NMIEG | — | RAME |
| Initial value | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | — | R/W | — | R/W |

**RAM Enable**

| | |
|---|---|
| 0 | On-chip RAM is disabled. |
| 1 | On-chip RAM is enabled. |

**NMI Edge**

| | |
|---|---|
| 0 | Falling edge of $\overline{\text{NMI}}$ is detected. |
| 1 | Rising edge of $\overline{\text{NMI}}$ is detected. |

**Standby Timer Select**

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | Clock settling time = 8192 states |
| 0 | 0 | 1 | Clock settling time = 16384 states |
| 0 | 1 | 0 | Clock settling time = 32768 states |
| 0 | 1 | 1 | Clock settling time = 65536 states |
| 1 | – | – | Clock settling time = 131072 states |

**Software Standby**

| | |
|---|---|
| 0 | SLEEP instruction causes transition to sleep mode. |
| 1 | SLEEP instruction causes transition to software standby mode. |

**MDCR—Mode Control Register** **H'FFC5** **System Control**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MDS1 | MDS0 |
| Initial value | 1 | 1 | 1 | 0 | 0 | 1 | * | * |
| Read/Write | — | — | — | — | — | — | R | R |

**Mode Select**

Value at mode pins.

* Determined by inputs at pins $MD_1$ and $MD_0$.

**ISCR—IRQ Sense Control Register** **H'FFC6** **System Control**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | IRQ2EG | IRQ1EG | IRQ0EG | — | IRQ2SC | IRQ1SC | IRQ0SC |
| Initial value | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**IRQ0 Sense Control, IRQ0 Edge**

| IRQ0SC | IRQ0EG | Description |
|---|---|---|
| 0 | 0 | Low level of $\overline{IRQ_0}$ generates an interrupt request. |
| 0 | 1 | |
| 1 | 0 | Falling edge of $\overline{IRQ_0}$ generates an interrupt request. |
| 1 | 1 | Rising edge of $\overline{IRQ_0}$ generates an interrupt request. |

**IRQ1 Sense Control, IRQ1 Edge**

| IRQ1SC | IRQ1EG | Description |
|---|---|---|
| 0 | 0 | Low level of $\overline{IRQ_1}$ generates an interrupt request. |
| 0 | 1 | |
| 1 | 0 | Falling edge of $\overline{IRQ_1}$ generates an interrupt request. |
| 1 | 1 | Rising edge of $\overline{IRQ_1}$ generates an interrupt request. |

**IRQ2 Sense Control, IRQ2 Edge**

| IRQ2SC | IRQ2EG | Description |
|---|---|---|
| 0 | 0 | Low level of $\overline{IRQ_2}$ generates an interrupt request. |
| 0 | 1 | |
| 1 | 0 | Falling edge of $\overline{IRQ_2}$ generates an interrupt request. |
| 1 | 1 | Rising edge of $\overline{IRQ_2}$ generates an interrupt request. |

**IER—IRQ Enable Register** **H'FFC7** **System Control**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | $IRQ_2E$ | $IRQ_1E$ | $IRQ_0E$ |
| Initial value | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Read/Write | — | — | — | — | — | R/W | R/W | R/W |

**$IRQ_i$ Enable (i = 0 to 2)**

| | |
|---|---|
| 0 | $IRQ_i$ is disabled. |
| 1 | $IRQ_i$ is enabled. |

**TCR—Timer Control Register** **H'FFC8** **TMR0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Clock Select**

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | No clock source; timer stops. |
| 0 | 0 | 1 | Internal clock source: Ø/8, counted on falling edge. |
| 0 | 1 | 0 | Internal clock source: Ø/64, counted on falling edge. |
| 0 | 1 | 1 | Internal clock source: Ø/1024, counted on falling edge. |
| 1 | 0 | 0 | No clock source; timer stops. |
| 1 | 0 | 1 | External clock source, counted on rising edge. |
| 1 | 1 | 0 | External clock source, counted on falling edge. |
| 1 | 1 | 1 | External clock source, counted on both rising and falling edges. |

**Counter Clear**

| | | |
|---|---|---|
| 0 | 0 | Counter is not cleared. |
| 0 | 1 | Cleared by compare-match A. |
| 1 | 0 | Cleared by compare-match B. |
| 1 | 1 | Cleared on rising edge of external reset input. |

**Timer Overflow Interrupt Enable**

| | |
|---|---|
| 0 | Overflow interrupt request is disabled. |
| 1 | Overflow interrupt request is enabled. |

**Compare-Match Interrupt Enable A**

| | |
|---|---|
| 0 | Compare-match A interrupt request is disabled. |
| 1 | Compare-match A interrupt request is enabled. |

**Compare-Match Interrupt Enable B**

| | |
|---|---|
| 0 | Compare-match B interrupt request is disabled. |
| 1 | Compare-match B interrupt request is enabled. |

**TCSR—Timer Control/Status Register H'FFC9 TMR0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMFB | CMFA | OVF | — | OS3*2 | OS2*2 | OS1*2 | OS0*2 |
| Initial value | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | R/(W)*1 | R/(W)*1 | R/(W)*1 | — | R/W | R/W | R/W | R/W |

**Output Select**

| | | |
|---|---|---|
| 0 | 0 | No change on compare-match A. |
| 0 | 1 | Output 0 on compare-match A. |
| 1 | 0 | Output 1 on compare-match A. |
| 1 | 1 | Invert (toggle) output on compare-match A. |

**Output Select**

| | | |
|---|---|---|
| 0 | 0 | No change on compare-match B. |
| 0 | 1 | Output 0 on compare-match B. |
| 1 | 0 | Output 1 on compare-match B. |
| 1 | 1 | Invert (toggle) output on compare-match B. |

**Timer Overflow Flag**

| | |
|---|---|
| 0 | Cleared by reading OVF = 1, then writing 0. |
| 1 | Set when TCNT changes from H'FF to H'00. |

**Compare-Match Flag A**

| | |
|---|---|
| 0 | Cleared by reading CMFA = 1, then writing 0. |
| 1 | Set when TCNT = TCORA. |

**Compare-Match Flag B**

| | |
|---|---|
| 0 | Cleared by reading CMFB = 1, then writing 0. |
| 1 | Set when TCNT = TCORB. |

*1 Software can write a 0 in bits 7 to 5 to clear the flags, but cannot write a 1 in these bits.

*2 When all four bits (OS3 to OS0) are cleared to 0, output is disabled.

**TCORA—Time Constant Register A** **H'FFCA** **TMR0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The CMFA bit is set to 1 when TCORA = TCNT.

---

**TCORB—Time Constant Register B** **H'FFCB** **TMR0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The CMFB bit is set to 1 when TCORB = TCNT.

---

**TCNT—Timer Counter** **H'FFCC** **TMR0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Count value

---

**TCR—Timer Control Register** **H'FFD0** **TMR1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMIEB | CMIEA | OVIE | CCLR1 | CCLR0 | CKS2 | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Note:** Bit functions are the same as for TMR0.

**TCSR—Timer Control/Status Register** **H'FFD1** **TMR1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CMFB | CMFA | OVF | — | OS3*2 | OS2*2 | OS1*2 | OS0*2 |
| Initial value | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Read/Write | R/(W)*1 | R/(W)*1 | R/(W)*1 | — | R/W | R/W | R/W | R/W |

**Note:** Bit functions are the same as for TMR0.

*1 Software can write a 0 in bits 7 to 5 to clear the flags, but cannot write a 1 in these bits.
*2 When all four bits (OS3 to OS0) are cleared to 0, output is disabled.

---

**TCORA—Time Constant Register A** **H'FFD2** **TMR1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Note:** Bit functions are the same as for TMR0.

---

**TCORB—Time Constant Register B** **H'FFD3** **TMR1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Note:** Bit functions are the same as for TMR0.

**TCNT—Timer Counter** **H’FFD4** **TMR1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Note:** Bit functions are the same as for TMR0.

**SMR—Serial Mode Register** **H'FFD8** **SCI0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | C/$\overline{A}$ | CHR | PE | O/$\overline{E}$ | STOP | — | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | — | R/W | R/W |

**Clock Select**

| | | |
|---|---|---|
| 0 | 0 | Ø clock |
| 0 | 1 | Ø/4 clock |
| 1 | 0 | Ø/16 clock |
| 1 | 1 | Ø/64 clock |

**Stop Bit Length**

| | |
|---|---|
| 0 | One stop bit |
| 1 | Two stop bits |

**Parity Mode**

| | |
|---|---|
| 0 | Even parity |
| 1 | Odd parity |

**Parity Enable**

| | |
|---|---|
| 0 | Transmit: No parity bit added.<br>Receive: Parity bit not checked. |
| 1 | Transmit: Parity bit added.<br>Receive: Parity bit checked. |

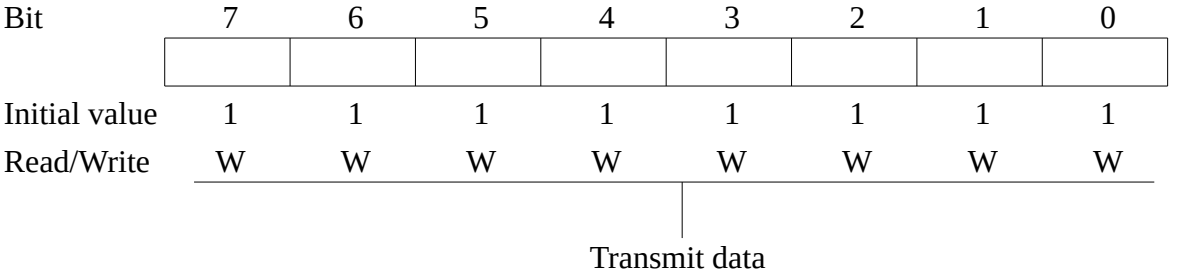
**Character Length**

| | |
|---|---|
| 0 | 8-Bit data length |
| 1 | 7-Bit data length |

**Communication Mode**

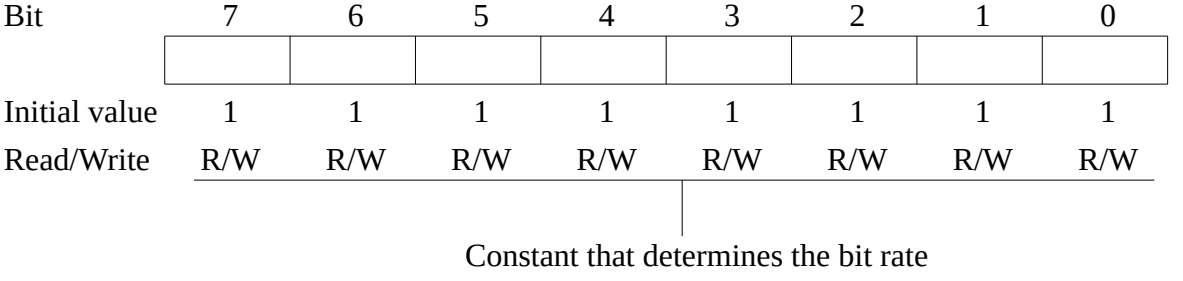
| | |
|---|---|
| 0 | Asynchronous |
| 1 | Synchronous |

**TDR—Transmit Data Register** **H’FFDB** **SCI0**




---

**BRR—Bit Rate Register** **H’FFD9** **SCI0**



**SCR—Serial Control Register** **H'FFDA** **SCI0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 |
| Initial value | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | — | — | R/W | R/W |

**Clock Enable 0**

| | |
|---|---|
| 0 | Asynchronous serial clock not output at SCK pin |
| 1 | Asynchronous serial clock output at SCK pin |

**Clock Enable 1**

| | |
|---|---|
| 0 | Internal clock |
| 1 | External clock |

**Receive Enable**

| | |
|---|---|
| 0 | Receive disabled |
| 1 | Receive enabled |

**Transmit Enable**

| | |
|---|---|
| 0 | Transmit disabled |
| 1 | Transmit enabled |

**Receive Interrupt Enable**

| | |
|---|---|
| 0 | Receive interrupt request is disabled. |
| 1 | Receive interrupt request is enabled. |

**Transmit Interrupt Enable**

| | |
|---|---|
| 0 | Transmit interrupt request is disabled. |
| 1 | Transmit interrupt request is enabled. |

**SSR—Serial Status Register** **H'FFDC** **SCI0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | RDRF | ORER | FER | PER | — | — | — |
| Initial value | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Read/Write | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W)* | — | — | — |

**Parity Error**

| | |
|---|---|
| 0 | Cleared by reading PER = 1, then writing 0. |
| 1 | Set when a parity error occurs (parity of receive data does not match parity selected by O/$\overline{E}$ bit). |

**Framing Error**

| | |
|---|---|
| 0 | Cleared by reading FER = 1, then writing 0. |
| 1 | Set when a framing error occurs (stop bit is 0). |

**Overrun Error**

| | |
|---|---|
| 0 | Cleared by reading ORER = 1, then writing 0. |
| 1 | Set when an overrun error occurs (reception of next data is completed while RDRF bit is set to 1). |

**Receive Data Register Full**

| | |
|---|---|
| 0 | Cleared by reading RDRF = 1, then writing 0. |
| 1 | Set when one character is received normally and transferred from RSR to RDR. |

**Transmit Data Register Empty**

| | |
|---|---|
| 0 | Cleared by reading TDRE = 1, then writing 0. |
| 1 | Set when:<br>1. Data is transferred from TDR to TSR.<br>2. TE is cleared while TDRE = 0. |

* Software can write a 0 in bits 7 to 3 to clear the flags, but cannot write a 1 in these bits.

**RDR—Receive Data Register** **H'FFDD** **SCI0**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

Receive data

---

**SMR—Serial Mode Register** **H'FFE0** **SCI1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | C/$\overline{A}$ | CHR | PE | O/$\overline{E}$ | STOP | — | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | — | R/W | R/W |

**Note:** Bit functions are the same as for SCI0.

---

**BRR—Bit Rate Register** **H'FFE1** **SCI1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Note:** Bit functions are the same as for SCI0.

---

**SCR—Serial Control Register** **H'FFE2** **SCI1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | RIE | TE | RE | — | — | CKE1 | CKE0 |
| Initial value | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | — | — | R/W | R/W |

**Note:** Bit functions are the same as for SCI0.

**TDR—Transmit Data Register** **H'FFE3** **SCI1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | W | W | W | W | W | W | W | W |

**Note:** Bit functions are the same as for SCI0.

---

**SSR—Serial Status Register** **H'FFE4** **SCI1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | RDRF | ORER | FER | PER | — | — | — |
| Initial value | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Read/Write | R/(W)* | R/(W)* | R/(W)* | R/(W)* | R/(W)* | — | — | — |

**Note:** Bit functions are the same as for SCI0.

* Software can write a 0 in bits 7 to 3 to clear the flags, but cannot write a 1 in these bits.

---

**RDR—Receive Data Register** **H'FFE5** **SCI1**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

**Note:** Bit functions are the same as for SCI0.

**HCSR—Handshake Control/Status Register H'FFFE Handshaking**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ISF | ISIE | OSE | OSS | LTE | BSE | — | — |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Read/Write | R | R/W | R/W | R/W | R/W | R/W | — | — |

**Busy Enable**

| | |
|---|---|
| 0 | $\overline{\text{BUSY}}$ output is disabled. |
| 1 | $\overline{\text{BUSY}}$ output is enabled. |

**Latch Enable**

| | |
|---|---|
| 0 | Input latches are disabled. |
| 1 | Input data are latched on falling edge of $\overline{\text{IS}}$. |

**Output Strobe Select**

| | |
|---|---|
| 0 | $\overline{\text{OS}}$ is output when port 3 is read. |
| 1 | $\overline{\text{OS}}$ is output when port 3 is written. |

**Output Strobe Enable**

| | |
|---|---|
| 0 | $\overline{\text{OS}}$ output is disabled. |
| 1 | $\overline{\text{OS}}$ output is enabled. |

**Input Strobe Interrupt Enable**

| | |
|---|---|
| 0 | Input strobe interrupt is disabled. |
| 1 | Input strobe interrupt is enabled. |

**Interrupt Strobe Flag**

| | |
|---|---|
| 0 | Cleared by reading HCSR when ISF = 1, then reading or writing port 3. |
| 1 | Set when $\overline{\text{IS}}$ goes low. |

**FNCR—FRT Noise Canceler Control Register H'FFFF FRT**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | NCS1 | NCS0 |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Read/Write | — | — | — | — | — | — | R/W | R/W |

**Noise Canceler Select**

| NCS1 | NCS0 | Description |
|---|---|---|
| 0 | 0 | Noise canceler is disabled. |
| 0 | 1 | Ø/32 sampling clock |
| 1 | 0 | Ø/64 sampling clock |
| 1 | 1 | Ø/128 sampling clock |

# Appendix C. Pin States

## C.1 Pin States in Each Mode

**Table C-1. Pin States**

| Pin Name | MCU Mode | Reset | Hardware Standby | Software Standby | Sleep Mode | Normal Operation |
|---|---|---|---|---|---|---|
| $P1_7$ to $P1_0$ | 1 | Low | 3-State | Low | Prev. state | Addr. output |
| $A_7$ to $A_0$ | 2 | 3-State | | Low if DDR = 1, Prev. state if DDR = 0 | (Addr. output pins: last address accessed) | Addr. output or input port |
| | 3 | | | Prev. state | | I/O port |
| $P2_7$ to $P2_0$ | 1 | Low | 3-State | Low | Prev. state | Addr. output |
| $A_{15}$ to $A_8$ | 2 | 3-State | | Low if DDR = 1, Prev. state if DDR = 0 | (Addr. output pins: last address accessed) | Addr. output or input port |
| | 3 | | | Prev. state | | I/O port |
| $P3_7$ to $P3_0$ $D_7$ to $D_0$ | 1 2 | 3-State | 3-State | 3-state | 3-State | $D_7$ to $D_0$ |
| | 3 | | | Prev. state | Prev. state | I/O port |
| $P4_7$/E | 1 2 | E clock output | 3-State | Low if DDR = 1, 3-state if DDR = 0 | E clock if DDR = 1, 3-state if DDR = 0 | E clock if DDR = 1, Input port if DDR = 0 |
| | 3 | 3-State | | Prev. state | Prev. state | I/O port |
| $P4_6$/Ø | 1 2 | Clock output | 3-state | High | Clock output | Clock output |
| | 3 | 3-State | | High if DDR = 1, 3-state if DDR = 0 | Clock output if DDR = 1, 3-state if DDR = 0 | Clock output if DDR = 1, input port if DDR = 0 |
| $P4_5$ to $P4_0$, | 1 2 3 | 3-State | 3-State | Prev. state (note 3) | Prev. state | I/O port |

**Table C-1. Pin States (cont.)**

| Pin Name | MCU Mode | Reset | Hardware Standby | Software Standby | Sleep Mode | Normal Operation |
|---|---|---|---|---|---|---|
| $P5_5$ to $P5_0$, | 1 | 3-State | 3-State | Prev. state | Prev. state | I/O port |
| | 2 | | | (note 3) | | |
| | 3 | | | | | |
| $P6_6$ to $P6_0$, | 1 | 3-State | 3-State | Prev. state | Prev. state | I/O port |
| | 2 | | | (note 3) | | |
| | 3 | | | | | |
| $P7_7$/$\overline{WAIT}$ | 1 | 3-State | 3-State | 3-state | 3-state | $\overline{WAIT}$ |
| | 2 | | | | | |
| | 3 | | | Prev. state | Prev. state | I/O port |
| $P7_6$ to $P7_4$, | 1 | High | 3-State | High | High | $\overline{AS}$, $\overline{WR}$, |
| $\overline{AS}$, $\overline{WR}$, $\overline{RD}$, | 2 | | | | | $\overline{RD}$ |
| | 3 | 3-State | | Prev. state | Prev. state | I/O port |
| $P7_3$ to $P7_0$, | 1 | 3-State | 3-State | Prev. state | Prev. state | I/O port |
| | 2 | | | | | |
| | 3 | | | | | |

**Notes:**

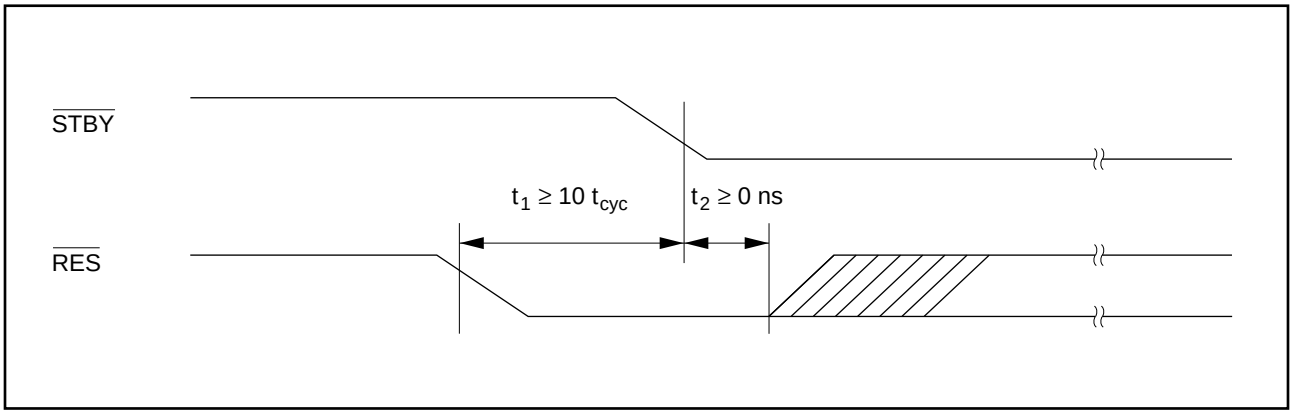
1. 3-state: High-impedance state
2. Prev. state: Previous state. Input ports are in the high-impedance state (with the MOS pull-up on if DDR = 0 and DR = 1). Output ports hold their previous output level.
3. On-chip supporting modules are initialized, so these pins revert to I/O ports according to the DDR and DR bits.
4. I/O port: Direction depends on the data direction (DDR) bit. Note that these pins may also be used by the on-chip supporting modules.

See section 5, I/O Ports for further information.

# Appendix D. Timing of Transition to and Recovery from Hardware Standby Mode

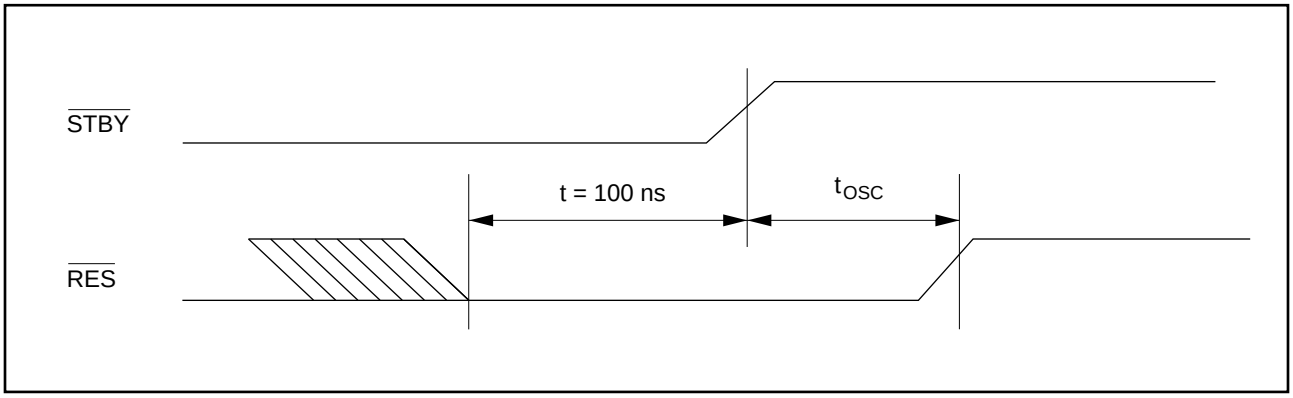
**Timing of Transition to Hardware Standby Mode**

(1) To retain RAM contents, drive the $\overline{\text{RES}}$ signal low 10 system clock cycles before the $\overline{\text{STBY}}$ signal goes low, as shown below. $\overline{\text{RES}}$ must remain low until $\overline{\text{STBY}}$ goes low (minimum delay from $\overline{\text{STBY}}$ low to $\overline{\text{RES}}$ high: 0 ns).




(2) When it is not necessary to retain RAM contents, $\overline{\text{RES}}$ does not have to be driven low as in (1).

Timing of Recovery From Hardware Standby Mode: Drive the $\overline{\text{RES}}$ signal low approximately 100 ns before $\overline{\text{STBY}}$ goes high.



# Appendix E. Package Dimensions

Figure E-1 shows the dimensions of the DC-64S package. Figure E-2 shows the dimensions of the DP-64S package. Figure E-3 shows the dimensions of the FP-64A package. Figure E-4 shows the dimensions of the CP-68 package.

Unit: mm

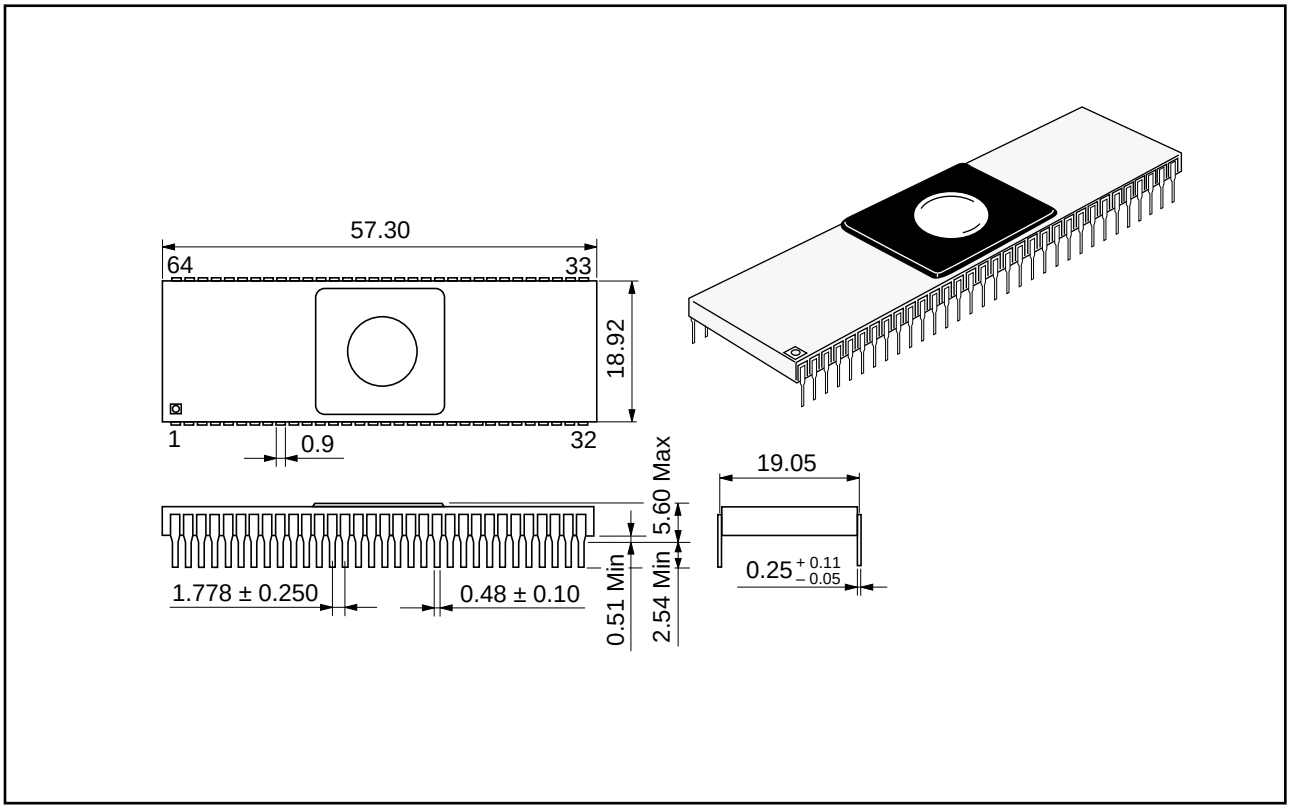



**Figure E-1. Package Dimensions (DC-64S)**

Unit: mm

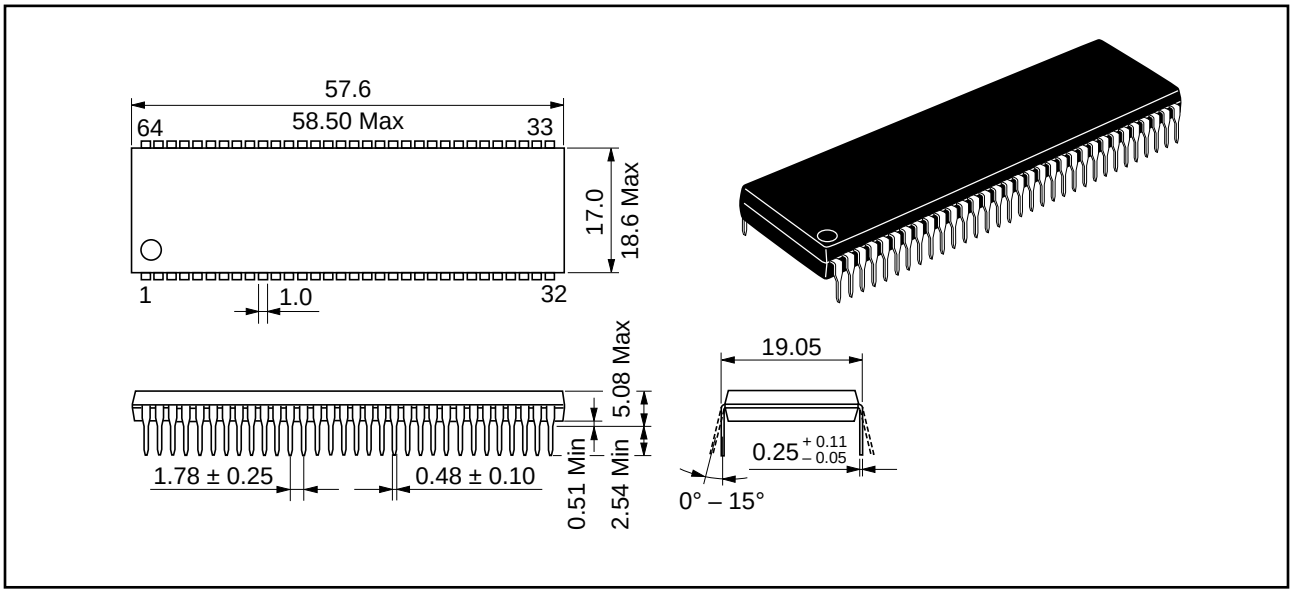



**Figure E-2. Package Dimensions (DP-64S)**

Unit: mm

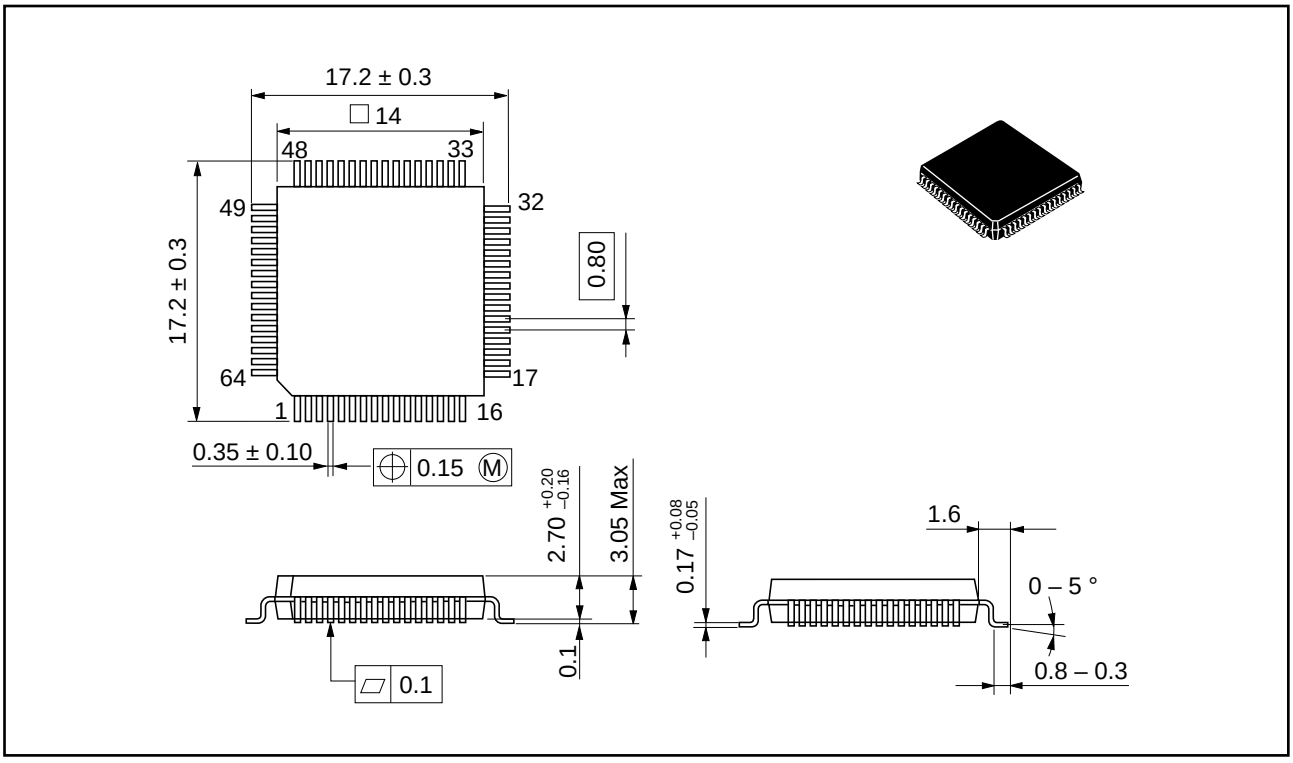



**Figure E-3. Package Dimensions (FP-64A)**

Unit: mm

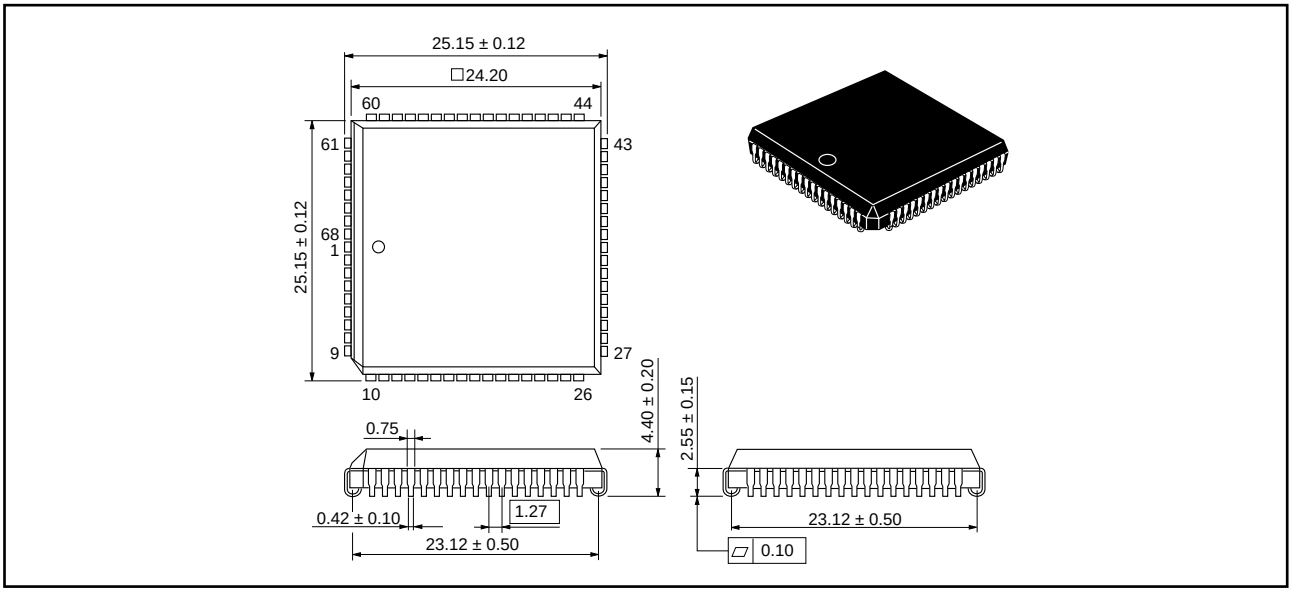



**Figure E-4. Package Dimensions (CP-68)**