# Section 1 Overview

T-49-19-16

## 1.1 Features

The H8/520 is an original Hitachi CMOS microcomputer unit (MCU) comprising a high-performance CPU core plus a full range of supporting functions—an entire system integrated onto a single chip.

The CPU features a highly orthogonal instruction set that permits addressing modes and data sizes to be specified independently in each instruction. An internal 16-bit architecture and 16-bit access to on-chip memory enhance the CPU's data-processing capability and provide the speed needed for realtime control applications. The address space can be expanded to perform high-volume data processing.

The on-chip supporting functions include RAM, ROM, timers, a serial communication interface (SCI), A/D converter, and I/O ports. An on-chip data transfer controller (DTC) provides an efficient way to transfer data in either direction between memory and I/O.

For the on-chip ROM, a choice is offered between masked ROM and programmable ROM (PROM). The PROM version can be programmed by the user with a general-purpose PROM writer.

Table 1-1 lists the main features of the H8/520 chip.

**Table 1-1 Features**

| Feature | Description |
|---|---|
| CPU | General-register machine<br>• Eight 16-bit general registers<br>• Five 8-bit and two 16-bit control registers<br>High speed<br>• Maximum clock rate: 10 MHz (oscillator frequency: 20 MHz)<br>Expanded operating modes supporting external memory<br>• Minimum mode: up to 64-kbyte address space<br>• Maximum mode: up to 1-Mbyte address space<br>Highly orthogonal instruction set<br>• Addressing modes and data size can be specified independently for each instruction<br>Instructions can address registers or memory<br>• Register-register operations<br>• Register-memory operations<br>Instruction set optimized for C language<br>• Special short formats for frequently-used instructions and addressing modes |
| Memory | • 512-Byte high-speed RAM on-chip<br>• 16-kbyte programmable or masked ROM on-chip |
| 16-Bit free-running timer (2 channels) | Each channel provides:<br>• 1 free-running counter (which can count external events)<br>• 2 output-compare registers<br>• 1 input capture register |
| 8-Bit timer (1 channel) | • One 8-bit up-counter (which can count external events)<br>• 2 time constant registers |
| Watchdog timer (1 channel) | • An overflow generates a reset<br>• Can output an external reset signal<br>• Can also be used as an interval timer |

T-49-19-16

### Table 1-1 Features (cont)

| Feature | Description |
|---|---|
| Serial communication interface (SCI) (2 channels) | • Asynchronous or synchronous mode (selectable)<br>• Full duplex: can send and receive simultaneously<br>• Built-in baud rate generator |
| A/D converter | • 10-Bit resolution<br>• 4 (or 8*) channels, controllable in single mode or scan mode (selectable)<br>• Sample-and-hold function<br>• Can be externally triggered |
| I/O ports | • 46 input/output pins (five 8-bit ports, one 6-bit port)<br>• 4 (or 8*) input-only pins (one 4- or 8*-bit port) |
| Interrupt controller (INTC) | • 9 external interrupt pins (NMI, $\overline{IRQ_0}$ to $\overline{IRQ_7}$)<br>• 18 internal interrupts<br>• 8 priority levels |
| Data transfer controller (DTC) | Performs efficient, rapid, bidirectional data transfer between memory and I/O with minimal CPU programming. |
| Wait-state controller (WSC) | Can insert wait states in access to external memory or I/O |
| Operating modes | 5 MCU operating modes<br>• Expanded minimum modes, supporting up to 64 kbytes external memory with or without using on-chip ROM (Modes 1 and 2)<br>• Expanded maximum modes, supporting up to 1 Mbyte external memory with or without using on-chip ROM (Modes 3 and 4)<br>• Single-chip mode (Mode 7)<br>3 power-down modes<br>• Sleep mode<br>• Software standby mode<br>• Hardware standby mode |
| Other features | • Clock generator on-chip |

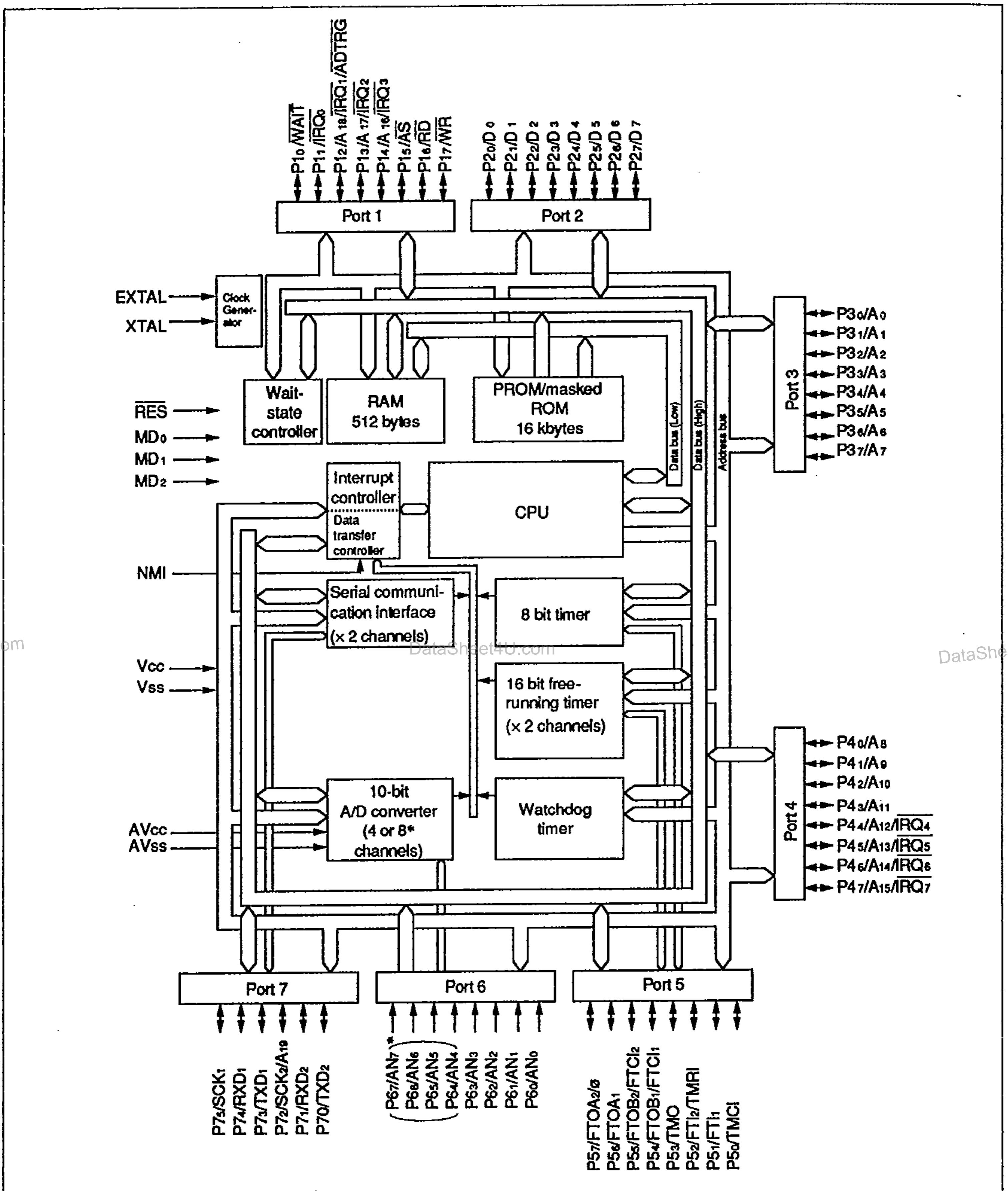
| Feature | Model Name | Package Options | ROM |
|---|---|---|---|
| Product line-up | HD6475208C | 64-Pin windowed shrink DIP (DC-64S) | PROM |
| | HD6475208P | 64-Pin shrink DIP (DP-64S) | |
| | HD6475208CP | 68-Pin PLCC (CP-68) | |
| | HD6475208F | 64-Pin QFP (FP-64A) | |
| | HD6435208P | 64-Pin shrink DIP (DP-64S) | Masked ROM |
| | HD6435208CP | 68-Pin PLCC (CP-68) | |
| | HD6435208F | 64-Pin QFP (FP-64A) | |

Note: * CP- 68 package only.

## 1.2 Block Diagram

T-49-19-16

Figure 1-1 shows a block diagram of the H8/520 chip.



Note: * CP-68 package only

Figure 1-1 Block Diagram

## 1.3 Pin Arrangements and Functions

T-49-19-16

### 1.3.1 Pin Arrangement

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

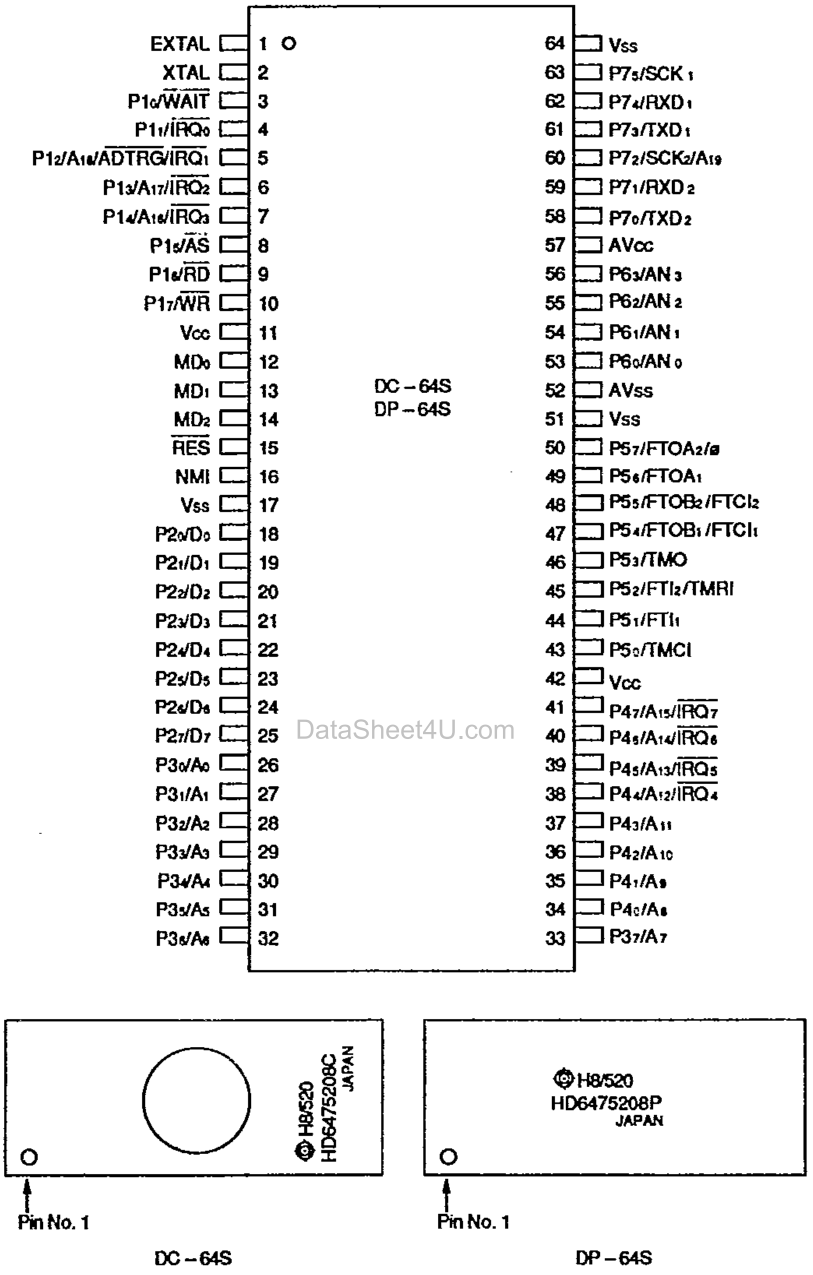


Figure 1-2 Pin Arrangement (DC-64S, DP-64S, Top View)

T-49-19-16

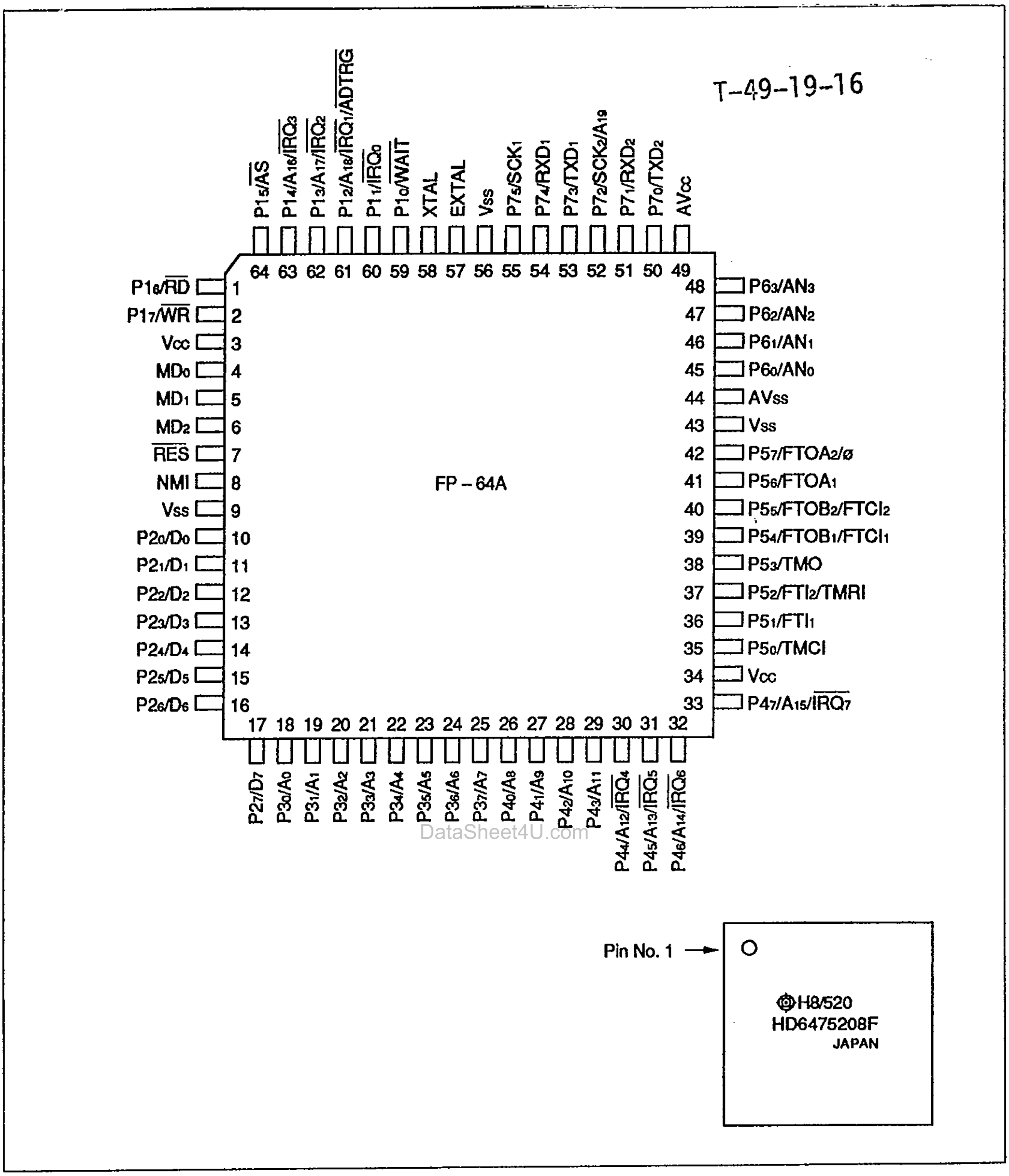


**Figure 1-3 Pin Arrangement (FP-64A, Top View)**

T-49-19-16

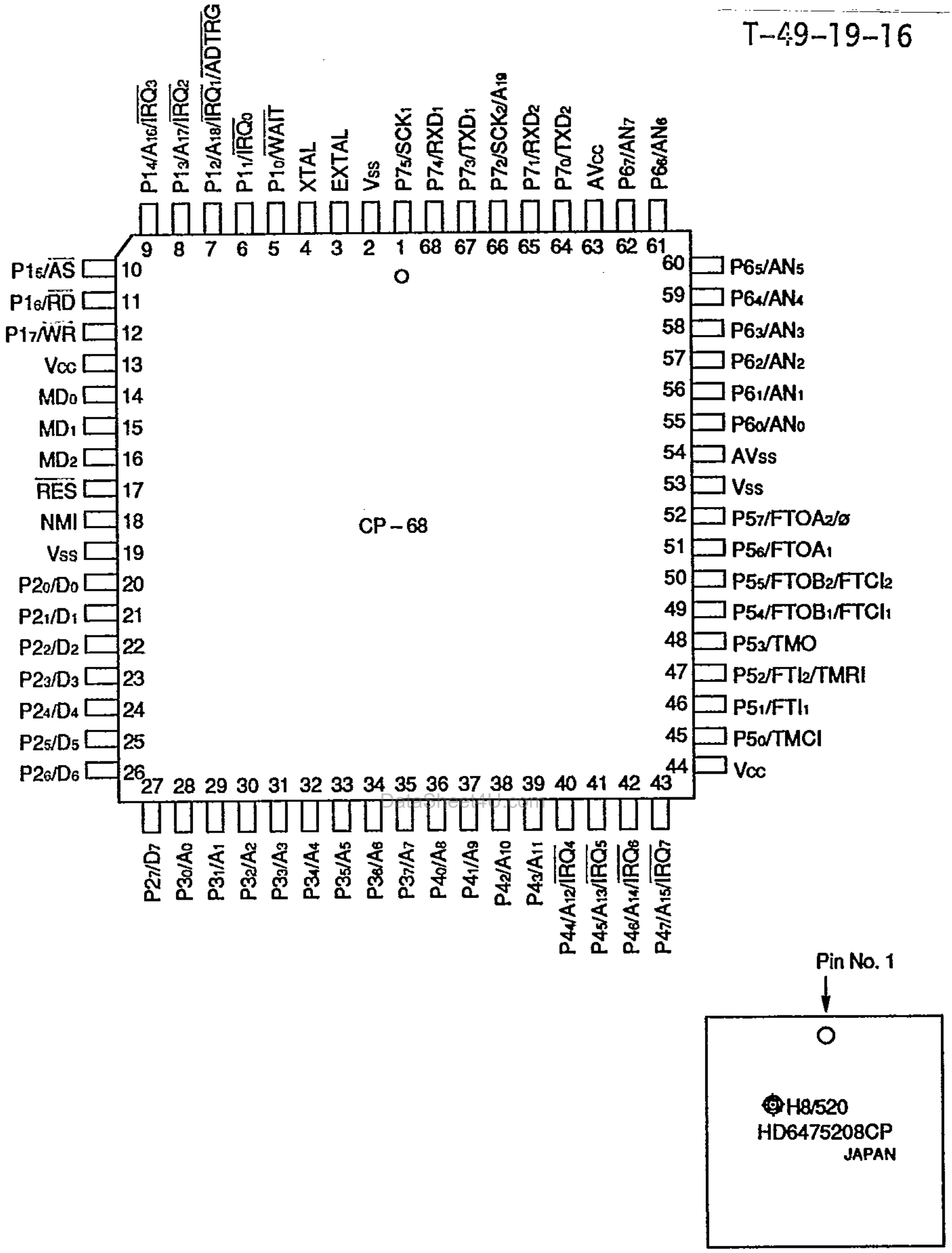


Figure 1-4 Pin Arrangement (CP-68, Top View)

### 1.3.2 Pin Functions

**Pin Arrangements in Each Operating Mode:** Table 1-2 lists the arrangements of the pins of the DC-64S and DP-64S packages in each operating mode. Table 1-3 lists the arrangements for the FP-64A package. Table 1-4 lists the arrangements for the CP-68 package.

**Table 1-2 Pin Arrangements in Each Operating Mode (DC-64S , DP-64S )**

| | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| Pin | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM |
| No. | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | Mode |
| 1 | EXTAL | EXTAL | EXTAL | EXTAL | EXTAL | NC |
| 2 | XTAL | XTAL | XTAL | XTAL | XTAL | NC |
| 3 | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ | NC |
| 4 | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | NC |
| 5 | P1$_2$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$ | P1$_2$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$ | A$_{18}$ | P1$_2$ / A$_{18}$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$ | P1$_2$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$ | NC |
| 6 | P1$_3$ / $\overline{\text{IRQ}_2}$ | P1$_3$ / $\overline{\text{IRQ}_2}$ | A$_{17}$ | P1$_3$ / A$_{17}$ / $\overline{\text{IRQ}_2}$ | P1$_3$ / $\overline{\text{IRQ}_2}$ | NC |
| 7 | P1$_4$ / $\overline{\text{IRQ}_3}$ | P1$_4$ / $\overline{\text{IRQ}_3}$ | A$_{16}$ | P1$_4$ / A$_{16}$ / $\overline{\text{IRQ}_3}$ | P1$_4$ / $\overline{\text{IRQ}_3}$ | NC |
| 8 | $\overline{\text{AS}}$ | $\overline{\text{AS}}$ | $\overline{\text{AS}}$ | $\overline{\text{AS}}$ | P1$_5$ | NC |
| 9 | $\overline{\text{RD}}$ | $\overline{\text{RD}}$ | $\overline{\text{RD}}$ | $\overline{\text{RD}}$ | P1$_6$ | NC |
| 10 | $\overline{\text{WR}}$ | $\overline{\text{WR}}$ | $\overline{\text{WR}}$ | $\overline{\text{WR}}$ | P1$_7$ | NC |
| 11 | V$_{CC}$ | V$_{CC}$ | V$_{CC}$ | V$_{CC}$ | V$_{CC}$ | V$_{CC}$ |
| 12 | MD$_0$ | MD$_0$ | MD$_0$ | MD$_0$ | MD$_0$ | V$_{CC}$ |
| 13 | MD$_1$ | MD$_1$ | MD$_1$ | MD$_1$ | MD$_1$ | V$_{SS}$ |
| 14 | MD$_2$ | MD$_2$ | MD$_2$ | MD$_2$ | MD$_2$ | V$_{CC}$ |
| 15 | $\overline{\text{RES}}$ | $\overline{\text{RES}}$ | $\overline{\text{RES}}$ | $\overline{\text{RES}}$ | $\overline{\text{RES}}$ | V$_{PP}$ |
| 16 | NMI | NMI | NMI | NMI | NMI | A$_9$ |
| 17 | V$_{SS}$ | V$_{SS}$ | V$_{SS}$ | V$_{SS}$ | V$_{SS}$ | V$_{SS}$ |
| 18 | D$_0$ | D$_0$ | D$_0$ | D$_0$ | P2$_0$ | O$_0$ |
| 19 | D$_1$ | D$_1$ | D$_1$ | D$_1$ | P2$_1$ | O$_1$ |
| 20 | D$_2$ | D$_2$ | D$_2$ | D$_2$ | P2$_2$ | O$_2$ |
| 21 | D$_3$ | D$_3$ | D$_3$ | D$_3$ | P2$_3$ | O$_3$ |
| 22 | D$_4$ | D$_4$ | D$_4$ | D$_4$ | P2$_4$ | O$_4$ |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-2 Pin Arrangements in Each Operating Mode (DC-64S, DP-64S) (cont)

T-49-19-16

| Pin No. | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM Mode |
| | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | |
| 23 | $D_5$ | $D_5$ | $D_5$ | $D_5$ | $P2_5$ | $O_5$ |
| 24 | $D_6$ | $D_6$ | $D_6$ | $D_6$ | $P2_6$ | $O_6$ |
| 25 | $D_7$ | $D_7$ | $D_7$ | $D_7$ | $P2_7$ | $O_7$ |
| 26 | $A_0$ | $A_0$ | $A_0$ | $A_0$ | $P3_0$ | $A_0$ |
| 27 | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $P3_1$ | $A_1$ |
| 28 | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $P3_2$ | $A_2$ |
| 29 | $A_3$ | $A_3$ | $A_3$ | $A_3$ | $P3_3$ | $A_3$ |
| 30 | $A_4$ | $A_4$ | $A_4$ | $A_4$ | $P3_4$ | $A_4$ |
| 31 | $A_5$ | $A_5$ | $A_5$ | $A_5$ | $P3_5$ | $A_5$ |
| 32 | $A_6$ | $A_6$ | $A_6$ | $A_6$ | $P3_6$ | $A_6$ |
| 33 | $A_7$ | $A_7$ | $A_7$ | $A_7$ | $P3_7$ | $A_7$ |
| 34 | $A_8$ | $P4_0$ / $A_8$ | $A_8$ | $P4_0$ / $A_8$ | $P4_0$ | $A_8$ |
| 35 | $A_9$ | $P4_1$ / $A_9$ | $A_9$ | $P4_1$ / $A_9$ | $P4_1$ | $\overline{OE}$ |
| 36 | $A_{10}$ | $P4_2$ / $A_{10}$ | $A_{10}$ | $P4_2$ / $A_{10}$ | $P4_2$ | $A_{10}$ |
| 37 | $A_{11}$ | $P4_3$ / $A_{11}$ | $A_{11}$ | $P4_3$ / $A_{11}$ | $P4_3$ | $A_{11}$ |
| 38 | $A_{12}$ | $P4_4$ / $A_{12}$ / $\overline{IRQ_4}$ | $A_{12}$ | $P4_4$ / $A_{12}$ / $\overline{IRQ_4}$ | $P4_4$ / $\overline{IRQ_4}$ | $A_{12}$ |
| 39 | $A_{13}$ | $P4_5$ / $A_{13}$ / $\overline{IRQ_5}$ | $A_{13}$ | $P4_5$ / $A_{13}$ / $\overline{IRQ_5}$ | $P4_5$ / $\overline{IRQ_5}$ | $A_{13}$ |
| 40 | $A_{14}$ | $P4_6$ / $A_{14}$ / $\overline{IRQ_6}$ | $A_{14}$ | $P4_6$ / $A_{14}$ / $\overline{IRQ_6}$ | $P4_6$ / $\overline{IRQ_6}$ | $A_{14}$ |
| 41 | $A_{15}$ | $P4_7$ / $A_{15}$ / $\overline{IRQ_7}$ | $A_{15}$ | $P4_7$ / $A_{15}$ / $\overline{IRQ_7}$ | $P4_7$ / $\overline{IRQ_7}$ | $\overline{CE}$ |
| 42 | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ |
| 43 | $P5_0$ / TMCI | $P5_0$ / TMCI | $P5_0$ / TMCI | $P5_0$ / TMCI | $P5_0$ / TMCI | $V_{CC}$ |
| 44 | $P5_1$ / $FTI_1$ | $P5_1$ / $FTI_1$ | $P5_1$ / $FTI_1$ | $P5_1$ / $FTI_1$ | $P5_1$ / $FTI_1$ | $V_{CC}$ |
| 45 | $P5_2$ / $FTI_2$ / TMRI | $P5_2$ / $FTI_2$ / TMRI | $P5_2$ / $FTI_2$ / TMRI | $P5_2$ / $FTI_2$ / TMRI | $P5_2$ / $FTI_2$ / TMRI | NC |
| 46 | $P5_3$ / TMO | $P5_3$ / TMO | $P5_3$ / TMO | $P5_3$ / TMO | $P5_3$ / TMO | NC |
| 47 | $P5_4$ / $FTOB_1$ / $FTCI_1$ | $P5_4$ / $FTOB_1$ / $FTCI_1$ | $P5_4$ / $FTOB_1$ / $FTCI_1$ | $P5_4$ / $FTOB_1$ / $FTCI_1$ | $P5_4$ / $FTOB_1$ / $FTCI_1$ | NC |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-2 Pin Arrangements in Each Operating Mode (DC-64S, DP-64S) (cont)

T-49-19-16

| Pin | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM |
| No. | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | Mode |
| 48 | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | NC |
| 49 | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | NC |
| 50 | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | NC |
| 51 | Vss | Vss | Vss | Vss | Vss | Vss |
| 52 | AVss | AVss | AVss | AVss | AVss | Vss |
| 53 | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | NC |
| 54 | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | NC |
| 55 | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | NC |
| 56 | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | NC |
| 57 | AVcc | AVcc | AVcc | AVcc | AVcc | Vcc |
| 58 | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | NC |
| 59 | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | NC |
| 60 | P7$_2$ / SCK$_2$ | P7$_2$ / SCK$_2$ | A$_{19}$ | P7$_2$ / SCK$_2$ / A$_{19}$ | P7$_2$ / SCK$_2$ | NC |
| 61 | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | NC |
| 62 | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | NC |
| 63 | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | NC |
| 64 | Vss | Vss | Vss | Vss | Vss | Vss |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-3 Pin Arrangements in Each Operating Mode (FP-64A)

T-49-19-16

| Pin No. | Pin Name: Expanded Minimum Modes: Mode 1 | Mode 2 | Expanded Maximum Modes: Mode 3 | Mode 4 | Single-Chip Mode: Mode 7 | PROM Mode |
|---|---|---|---|---|---|---|
| 1 | $\overline{RD}$ | $\overline{RD}$ | $\overline{RD}$ | $\overline{RD}$ | $P1_6$ | NC |
| 2 | $\overline{WR}$ | $\overline{WR}$ | $\overline{WR}$ | $\overline{WR}$ | $P1_7$ | NC |
| 3 | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ |
| 4 | $MD_0$ | $MD_0$ | $MD_0$ | $MD_0$ | $MD_0$ | $V_{CC}$ |
| 5 | $MD_1$ | $MD_1$ | $MD_1$ | $MD_1$ | $MD_1$ | $V_{SS}$ |
| 6 | $MD_2$ | $MD_2$ | $MD_2$ | $MD_2$ | $MD_2$ | $V_{CC}$ |
| 7 | $\overline{RES}$ | $\overline{RES}$ | $\overline{RES}$ | $\overline{RES}$ | $\overline{RES}$ | $V_{PP}$ |
| 8 | NMI | NMI | NMI | NMI | NMI | $A_9$ |
| 9 | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ |
| 10 | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $P2_0$ | $O_0$ |
| 11 | $D_1$ | $D_1$ | $D_1$ | $D_1$ | $P2_1$ | $O_1$ |
| 12 | $D_2$ | $D_2$ | $D_2$ | $D_2$ | $P2_2$ | $O_2$ |
| 13 | $D_3$ | $D_3$ | $D_3$ | $D_3$ | $P2_3$ | $O_3$ |
| 14 | $D_4$ | $D_4$ | $D_4$ | $D_4$ | $P2_4$ | $O_4$ |
| 15 | $D_5$ | $D_5$ | $D_5$ | $D_5$ | $P2_5$ | $O_5$ |
| 16 | $D_6$ | $D_6$ | $D_6$ | $D_6$ | $P2_6$ | $O_6$ |
| 17 | $D_7$ | $D_7$ | $D_7$ | $D_7$ | $P2_7$ | $O_7$ |
| 18 | $A_0$ | $P3_0/A_0$ | $A_0$ | $P3_0/A_0$ | $P3_0$ | $A_0$ |
| 19 | $A_1$ | $P3_1/A_1$ | $A_1$ | $P3_1/A_1$ | $P3_1$ | $A_1$ |
| 20 | $A_2$ | $P3_2/A_2$ | $A_2$ | $P3_2/A_2$ | $P3_2$ | $A_2$ |
| 21 | $A_3$ | $P3_3/A_3$ | $A_3$ | $P3_3/A_3$ | $P3_3$ | $A_3$ |
| 22 | $A_4$ | $P3_4/A_4$ | $A_4$ | $P3_4/A_4$ | $P3_4$ | $A_4$ |
| 23 | $A_5$ | $P3_5/A_5$ | $A_5$ | $P3_5/A_5$ | $P3_5$ | $A_5$ |
| 24 | $A_6$ | $P3_6/A_6$ | $A_6$ | $P3_6/A_6$ | $P3_6$ | $A_6$ |
| 25 | $A_7$ | $P3_7/A_7$ | $A_7$ | $P3_7/A_7$ | $P3_7$ | $A_7$ |
| 26 | $A_8$ | $P4_0 / A_8$ | $A_8$ | $P4_0 / A_8$ | $P4_0$ | $A_8$ |
| 27 | $A_9$ | $P4_1 / A_9$ | $A_9$ | $P4_1 / A_9$ | $P4_1$ | $\overline{OE}$ |
| 28 | $A_{10}$ | $P4_2 / A_{10}$ | $A_{10}$ | $P4_2 / A_{10}$ | $P4_2$ | $A_{10}$ |
| 29 | $A_{11}$ | $P4_3 / A_{11}$ | $A_{11}$ | $P4_3 / A_{11}$ | $P4_3$ | $A_{11}$ |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-3 Pin Arrangements in Each Operating Mode (FP-64A) (cont)

T-49-19-16

| Pin | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM |
| No. | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | Mode |
| 30 | $A_{12}$ | $P4_4$ / $A_{12}$ / $\overline{IRQ_4}$ | $A_{12}$ | $P4_4$ / $A_{12}$ / $\overline{IRQ_4}$ | $P4_4$ / $\overline{IRQ_4}$ | $A_{12}$ |
| 31 | $A_{13}$ | $P4_5$ / $A_{13}$ / $\overline{IRQ_5}$ | $A_{13}$ | $P4_5$ / $A_{13}$ / $\overline{IRQ_5}$ | $P4_5$ / $\overline{IRQ_5}$ | $A_{13}$ |
| 32 | $A_{14}$ | $P4_6$ / $A_{14}$ / $\overline{IRQ_6}$ | $A_{14}$ | $P4_6$ / $A_{14}$ / $\overline{IRQ_6}$ | $P4_6$ / $\overline{IRQ_6}$ | $A_{14}$ |
| 33 | $A_{15}$ | $P4_7$ / $A_{15}$ / $\overline{IRQ_7}$ | $A_{15}$ | $P4_7$ / $A_{15}$ / $\overline{IRQ_7}$ | $P4_7$ / $\overline{IRQ_7}$ | $\overline{CE}$ |
| 34 | Vcc | Vcc | Vcc | Vcc | Vcc | Vcc |
| 35 | $P5_0$ /TMCI | $P5_0$ /TMCI | $P5_0$ /TMCI | $P5_0$ /TMCI | $P5_0$ /TMCI | Vcc |
| 36 | $P5_1$ /$FTI_1$ | $P5_1$ /$FTI_1$ | $P5_1$ /$FTI_1$ | $P5_1$ $FTI_1$ | $P5_1$ /$FTI_1$ | Vcc |
| 37 | $P5_2$ /$FTI_2$ / TMRI | $P5_2$ /$FTI_2$ / TMRI | $P5_2$ /$FTI_2$ / TMRI | $P5_2$ /$FTI_2$ / TMRI | $P5_2$ /$FTI_2$ / TMRI | NC |
| 38 | $P5_3$/TMO | $P5_3$/TMO | $P5_3$/TMO | $P5_3$/TMO | $P5_3$/TMO | NC |
| 39 | $P5_4$ /$FTOB_1$ / $FTCI_1$ | $P5_4$ /$FTOB_1$ / $FTCI_1$ | $P5_4$ /$FTOB_1$ / $FTCI_1$ | $P5_4$ /$FTOB_1$ / $FTCI_1$ | $P5_4$ /$FTOB_1$ / $FTCI_1$ | NC |
| 40 | $P5_5$ /$FTOB_2$ / $FTCI_2$ | $P5_5$ /$FTOB_2$ / $FTCI_2$ | $P5_5$ /$FTOB_2$ / $FTCI_2$ | $P5_5$ /$FTOB_2$ / $FTCI_2$ | $P5_5$ /$FTOB_2$ / $FTCI_2$ | NC |
| 41 | $P5_6$ /$FTOA_1$ | $P5_6$ /$FTOA_1$ | $P5_6$ /$FTOA_1$ | $P5_6$ /$FTOA_1$ | $P5_6$ /$FTOA_1$ | NC |
| 42 | $P5_7$ /$FTOA_2$ / ø | $P5_7$ /$FTOA_2$ / ø | $P5_7$ /$FTOA_2$ / ø | $P5_7$ /$FTOA_2$ / ø | $P5_7$ /$FTOA_2$ / ø | NC |
| 43 | Vss | Vss | Vss | Vss | Vss | Vss |
| 44 | AVss | AVss | AVss | AVss | AVss | Vss |
| 45 | $P6_0$ / $AN_0$ | $P6_0$ / $AN_0$ | $P6_0$ / $AN_0$ | $P6_0$ / $AN_0$ | $P6_0$ / $AN_0$ | NC |
| 46 | $P6_1$ / $AN_1$ | $P6_1$ / $AN_1$ | $P6_1$ / $AN_1$ | $P6_1$ / $AN_1$ | $P6_1$ / $AN_1$ | NC |
| 47 | $P6_2$ / $AN_2$ | $P6_2$ / $AN_2$ | $P6_2$ / $AN_2$ | $P6_2$ / $AN_2$ | $P6_2$ / $AN_2$ | NC |
| 48 | $P6_3$ / $AN_3$ | $P6_3$ / $AN_3$ | $P6_3$ / $AN_3$ | $P6_3$ / $AN_3$ | $P6_3$ / $AN_3$ | NC |
| 49 | AVcc | AVcc | AVcc | AVcc | AVcc | Vcc |
| 50 | $P7_0$ / $TXD_2$ | $P7_0$ / $TXD_2$ | $P7_0$ / $TXD_2$ | $P7_0$ / $TXD_2$ | $P7_0$ / $TXD_2$ | NC |

Notes:
1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-3 Pin Arrangements in Each Operating Mode (FP-64A) (cont)

T-49-19-16

| Pin | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM |
| No. | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | Mode |
| 51 | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | NC |
| 52 | P7$_2$ / SCK$_2$ | P7$_2$ / SCK$_2$ | A$_{19}$ | P7$_2$ / SCK$_2$ / A$_{19}$ | P7$_2$ / SCK$_2$ | NC |
| 53 | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | NC |
| 54 | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | NC |
| 55 | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | P7$_5$ / SCK$_1$ | NC |
| 56 | Vss | Vss | Vss | Vss | Vss | Vss |
| 57 | EXTAL | EXTAL | EXTAL | EXTAL | EXTAL | NC |
| 58 | XTAL | XTAL | XTAL | XTAL | XTAL | NC |
| 59 | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ / $\overline{\text{WAIT}}$ | P1$_0$ | NC |
| 60 | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | P1$_1$ / $\overline{\text{IRQ}_0}$ | NC |
| 61 | P1$_2$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$ | P1$_2$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$ | A$_{18}$ | P1$_2$ / A$_{18}$ / $\overline{\text{IRQ}_1}$/ $\overline{\text{ADTRG}}$ | P1$_2$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$ | NC |
| 62 | P1$_3$ / $\overline{\text{IRQ}_2}$ | P1$_3$ / $\overline{\text{IRQ}_2}$ | A$_{17}$ | P1$_3$ / A$_{17}$ / $\overline{\text{IRQ}_2}$ | P1$_3$ / $\overline{\text{IRQ}_2}$ | NC |
| 63 | P1$_4$ / $\overline{\text{IRQ}_3}$ | P1$_4$ / $\overline{\text{IRQ}_3}$ | A$_{16}$ | P1$_4$ / A$_{16}$ / $\overline{\text{IRQ}_3}$ | P1$_4$ / $\overline{\text{IRQ}_3}$ | NC |
| 64 | $\overline{\text{AS}}$ | $\overline{\text{AS}}$ | $\overline{\text{AS}}$ | $\overline{\text{AS}}$ | P1$_5$ | NC |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-4 Pin Arrangements in Each Operating Mode (CP-68)

T-49-19-16

| | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| Pin | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM |
| No. | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | Mode |
| 1 | $P7_5$ / $SCK_1$ | $P7_5$ / $SCK_1$ | $P7_5$ / $SCK_1$ | $P7_5$ / $SCK_1$ | $P7_5$ / $SCK_1$ | NC |
| 2 | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ |
| 3 | EXTAL | EXTAL | EXTAL | EXTAL | EXTAL | NC |
| 4 | XTAL | XTAL | XTAL | XTAL | XTAL | NC |
| 5 | $P1_0$ / $\overline{WAIT}$ | $P1_0$ / $\overline{WAIT}$ | $P1_0$ / $\overline{WAIT}$ | $P1_0$ / $\overline{WAIT}$ | $P1_0$ | NC |
| 6 | $P1_1$ / $\overline{IRQ_0}$ | $P1_1$ / $\overline{IRQ_0}$ | $P1_1$ / $\overline{IRQ_0}$ | $P1_1$ / $\overline{IRQ_0}$ | $P1_1$ / $\overline{IRQ_0}$ | NC |
| 7 | $P1_2$ / $\overline{IRQ_1}$ / $\overline{ADTRG}$ | $P1_2$ / $\overline{IRQ_1}$ / $\overline{ADTRG}$ | $A_{18}$ | $P1_2$ / $A_{18}$ / $\overline{IRQ_1}$ / $\overline{ADTRG}$ | $P1_2$ / $\overline{IRQ_1}$ / $\overline{ADTRG}$ | NC |
| 8 | $P1_3$ / $\overline{IRQ_2}$ | $P1_3$ / $\overline{IRQ_2}$ | $A_{17}$ | $P1_3$ / $A_{17}$ / $\overline{IRQ_2}$ | $P1_3$ / $\overline{IRQ_2}$ | NC |
| 9 | $P1_4$ / $\overline{IRQ_3}$ | $P1_4$ / $\overline{IRQ_3}$ | $A_{16}$ | $P1_4$ / $A_{16}$ / $\overline{IRQ_3}$ | $P1_4$ / $\overline{IRQ_3}$ | NC |
| 10 | $\overline{AS}$ | $\overline{AS}$ | $\overline{AS}$ | $\overline{AS}$ | $P1_5$ | NC |
| 11 | $\overline{RD}$ | $\overline{RD}$ | $\overline{RD}$ | $\overline{RD}$ | $P1_6$ | NC |
| 12 | $\overline{WR}$ | $\overline{WR}$ | $\overline{WR}$ | $\overline{WR}$ | $P1_7$ | NC |
| 13 | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ | $V_{CC}$ |
| 14 | $MD_0$ | $MD_0$ | $MD_0$ | $MD_0$ | $MD_0$ | $V_{CC}$ |
| 15 | $MD_1$ | $MD_1$ | $MD_1$ | $MD_1$ | $MD_1$ | $V_{SS}$ |
| 16 | $MD_2$ | $MD_2$ | $MD_2$ | $MD_2$ | $MD_2$ | $V_{CC}$ |
| 17 | $\overline{RES}$ | $\overline{RES}$ | $\overline{RES}$ | $\overline{RES}$ | $\overline{RES}$ | $V_{PP}$ |
| 18 | NMI | NMI | NMI | NMI | NMI | $A_9$ |
| 19 | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ |
| 20 | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $P2_0$ | $O_0$ |
| 21 | $D_1$ | $D_1$ | $D_1$ | $D_1$ | $P2_1$ | $O_1$ |
| 22 | $D_2$ | $D_2$ | $D_2$ | $D_2$ | $P2_2$ | $O_2$ |
| 23 | $D_3$ | $D_3$ | $D_3$ | $D_3$ | $P2_3$ | $O_3$ |
| 24 | $D_4$ | $D_4$ | $D_4$ | $D_4$ | $P2_4$ | $O_4$ |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-4 Pin Arrangements in Each Operating Mode (CP-68) (cont)

| | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| Pin | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM |
| No. | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | Mode |
| 25 | D$_5$ | D$_5$ | D$_5$ | D$_5$ | P2$_5$ | O$_5$ |
| 26 | D$_6$ | D$_6$ | D$_6$ | D$_6$ | P2$_6$ | O$_6$ |
| 27 | D$_7$ | D$_7$ | D$_7$ | D$_7$ | P2$_7$ | O$_7$ |
| 28 | A$_0$ | A$_0$ | A$_0$ | A$_0$ | P3$_0$ | A$_0$ |
| 29 | A$_1$ | A$_1$ | A$_1$ | A$_1$ | P3$_1$ | A$_1$ |
| 30 | A$_2$ | A$_2$ | A$_2$ | A$_2$ | P3$_2$ | A$_2$ |
| 31 | A$_3$ | A$_3$ | A$_3$ | A$_3$ | P3$_3$ | A$_3$ |
| 32 | A$_4$ | A$_4$ | A$_4$ | A$_4$ | P3$_4$ | A$_4$ |
| 33 | A$_5$ | A$_5$ | A$_5$ | A$_5$ | P3$_5$ | A$_5$ |
| 34 | A$_6$ | A$_6$ | A$_6$ | A$_6$ | P3$_6$ | A$_6$ |
| 35 | A$_7$ | A$_7$ | A$_7$ | A$_7$ | P3$_7$ | A$_7$ |
| 36 | A$_8$ | P4$_0$ / A$_8$ | A$_8$ | P4$_0$ / A$_8$ | P4$_0$ | $\overline{A_8}$ |
| 37 | A$_9$ | P4$_1$ / A$_9$ | A$_9$ | P4$_1$ / A$_9$ | P4$_1$ | $\overline{OE}$ |
| 38 | A$_{10}$ | P4$_2$ / A$_{10}$ | A$_{10}$ | P4$_2$ / A$_{10}$ | P4$_2$ | A$_{10}$ |
| 39 | A$_{11}$ | P4$_3$ / A$_{11}$ | A$_{11}$ | P4$_3$ / A$_{11}$ | P4$_3$ | A$_{11}$ |
| 40 | A$_{12}$ | P4$_4$ / A$_{12}$ / $\overline{IRQ_4}$ | A$_{12}$ | P4$_4$ / A$_{12}$ / $\overline{IRQ_4}$ | P4$_4$ / $\overline{IRQ_4}$ | A$_{12}$ |
| 41 | A$_{13}$ | P4$_5$ / A$_{13}$ / $\overline{IRQ_5}$ | A$_{13}$ | P4$_5$ / A$_{13}$ / $\overline{IRQ_5}$ | P4$_5$ / $\overline{IRQ_5}$ | A$_{13}$ |
| 42 | A$_{14}$ | P4$_6$ / A$_{14}$ / $\overline{IRQ_6}$ | A$_{14}$ | P4$_6$ / A$_{14}$ / $\overline{IRQ_6}$ | P4$_6$ / $\overline{IRQ_6}$ | A$_{14}$ |
| 43 | A$_{15}$ | P4$_7$ / A$_{15}$ / $\overline{IRQ_7}$ | A$_{15}$ | P4$_7$ / A$_{15}$ / $\overline{IRQ_7}$ | P4$_7$ / $\overline{IRQ_7}$ | $\overline{CE}$ |
| 44 | Vcc | Vcc | Vcc | Vcc | Vcc | Vcc |
| 45 | P5$_0$ / TMCI | P5$_0$ / TMCI | P5$_0$ / TMCI | P5$_0$ / TMCI | P5$_0$ / TMCI | Vcc |
| 46 | P5$_1$ / FTI$_1$ | P5$_1$ / FTI$_1$ | P5$_1$ / FTI$_1$ | P5$_1$ / FTI$_1$ | P5$_1$ / FTI$_1$ | Vcc |
| 47 | P5$_2$ / FTI$_2$ / TMRI | P5$_2$ / FTI$_2$ / TMRI | P5$_2$ / FTI$_2$ / TMRI | P5$_2$ / FTI$_2$ / TMRI | P5$_2$ / FTI$_2$ / TMRI | NC |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

## Table 1-4 Pin Arrangements in Each Operating Mode (CP-68) (cont)

T-49-19-16

| | Pin Name | | | | | |
|---|---|---|---|---|---|---|
| Pin | Expanded Minimum Modes | | Expanded Maximum Modes | | Single-Chip Mode | PROM |
| No. | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 | Mode |
| 48 | P5$_3$ / TMO | P5$_3$ / TMO | P5$_3$ / TMO | P5$_3$ / TMO | P5$_3$ / TMO | NC |
| 49 | P5$_4$ / FTOB$_1$ / FTCI$_1$ | P5$_4$ / FTOB$_1$ / FTCI$_1$ | P5$_4$ / FTOB$_1$ / FTCI$_1$ | P5$_4$ / FTOB$_1$ / FTCI$_1$ | P5$_4$ / FTOB$_1$ / FTCI$_1$ | NC |
| 50 | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | P5$_5$ / FTOB$_2$ / FTCI$_2$ | NC |
| 51 | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | P5$_6$ / FTOA$_1$ | NC |
| 52 | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | P5$_7$ / FTOA$_2$ / ø | NC |
| 53 | Vss | Vss | Vss | Vss | Vss | Vss |
| 54 | AVss | AVss | AVss | AVss | AVss | Vss |
| 55 | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | P6$_0$ / AN$_0$ | NC |
| 56 | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | P6$_1$ / AN$_1$ | NC |
| 57 | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | P6$_2$ / AN$_2$ | NC |
| 58 | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | P6$_3$ / AN$_3$ | NC |
| 59 | P6$_4$ / AN$_4$ | P6$_4$ / AN$_4$ | P6$_4$ / AN$_4$ | P6$_4$ / AN$_4$ | P6$_4$ / AN$_4$ | NC |
| 60 | P6$_5$ / AN$_5$ | P6$_5$ / AN$_5$ | P6$_5$ / AN$_5$ | P6$_5$ / AN$_5$ | P6$_5$ / AN$_5$ | NC |
| 61 | P6$_6$ / AN$_6$ | P6$_6$ / AN$_6$ | P6$_6$ / AN$_6$ | P6$_6$ / AN$_6$ | P6$_6$ / AN$_6$ | NC |
| 62 | P6$_7$ / AN$_7$ | P6$_7$ / AN$_7$ | P6$_7$ / AN$_7$ | P6$_7$ / AN$_7$ | P6$_7$ / AN$_7$ | NC |
| 63 | AVcc | AVcc | AVcc | AVcc | AVcc | Vcc |
| 64 | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | P7$_0$ / TXD$_2$ | NC |
| 65 | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | P7$_1$ / RXD$_2$ | NC |
| 66 | P7$_2$ / SCK$_2$ | P7$_2$ / SCK$_2$ | A$_{19}$ | P7$_2$ / SCK$_2$ / A$_{19}$ | P7$_2$ / SCK$_2$ | NC |
| 67 | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | P7$_3$ / TXD$_1$ | NC |
| 68 | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | P7$_4$ / RXD$_1$ | NC |

Notes: 1. For the PROM mode, see section 16, "ROM."
2. Pins marked NC should be left unconnected.

**Pin Functions:** Table 1-5 gives a concise description of the function of each pin.

T-49-19-16

Table 1-5 Pin Functions

| Type | Symbol | Pin No. DC-64S DP-64S | Pin No. FP-64A | Pin No. CP-68 | I/O | Name and Function |
|---|---|---|---|---|---|---|
| Power | Vcc | 42, 11 | 34, 3 | 44, 13 | I | **Power:** Connected to the power supply (+5 V). Connect both Vcc pins to the system power supply (+5 V). The chip will not operate if either pin is left unconnected. |
| | Vss | 51, 17, 64 | 43, 9, 56 | 53, 19, 2 | I | **Ground:** Connected to ground (0 V). Connect all Vss pins to the system power supply (0 V). The chip will not operate if either pin is left unconnected. |
| Clock | XTAL | 2 | 58 | 4 | O | **Crystal:** Connected to a crystal oscillator. The crystal frequency should be double the desired ø clock frequency. If an external clock is input at the EXTAL pin, input an inverted clock signal at the XTAL pin. |
| | EXTAL | 1 | 57 | 3 | I | **External Crystal:** Connected to a crystal oscillator or external clock. The frequency of the external clock should be double the desired ø clock frequency. See section 8.2, "Oscillator Circuit", for examples of connections to a crystal and external clock. |
| | ø | 50 | 42 | 52 | O | **System Clock:** Supplies the ø clock to peripheral devices. |

**Table 1-5 Pin Functions (cont)**

T-49-19-16

| | | Pin No. | | | | |
|---|---|---|---|---|---|---|
| Type | Symbol | DC-64S<br>DP-64S | FP-64A | CP-68 | I/O | Name and Function |
| System control | $\overline{RES}$ | 15 | 7 | 17 | I/O | **Reset:** A low input causes the H8/520 chip to reset. If the reset output enable bit (RSTOE) is set to 1, when the watchdog timer overflows, a low signal is output for 132 system clock cycles. |
| Address bus | $A_{19}$ – $A_0$ | 60,<br>5 – 7<br>41 – 26 | 52,<br>61 – 63<br>33 – 18 | 66,<br>7 – 9<br>43 – 28 | O | **Address Bus:** Address output pins. |
| Data bus | $D_7$ – $D_0$ | 25 – 18 | 17 – 10 | 27 – 20 | I/O | **Data Bus:** 8-Bit bidirectional data bus. |
| Bus control | $\overline{WAIT}$ | 3 | 59 | 5 | I | **Wait:** Requests the CPU to insert one or more Tw states when accessing an off-chip address. |
| | $\overline{AS}$ | 8 | 64 | 10 | O | **Address Strobe:** Goes low to indicate that there is a valid address on the address bus. |
| | $\overline{RD}$ | 9 | 1 | 11 | O | **Read:** Goes low to indicate that the CPU is reading an external address. |
| | $\overline{WR}$ | 10 | 2 | 12 | O | **Write:** Goes low to indicate that the CPU is writing to an external address. |

T-49-19-16

**Table 1-5 Pin Functions (cont)**

| Type | Symbol | Pin No. DC-64S DP-64S | Pin No. FP-64A | Pin No. CP-68 | I/O | Name and Function |
|---|---|---|---|---|---|---|
| Interrupt | NMI | 16 | 8 | 18 | I | **NonMaskable Interrupt:** Highest-priority interrupt request signal. The non-maskable interrupt control register (NMICR) determines whether the interrupt is requested on the rising or falling edge of the NMI input. |
| | $\overline{IRQ_0}$ | 4 | 60 | 6 | I | **Interrupt Request 0 to 7:** Maskable interrupt request signals |
| | $\overline{IRQ_1}$ | 5 | 61 | 7 | | |
| | $\overline{IRQ_2}$ | 6 | 62 | 8 | | |
| | $\overline{IRQ_3}$ | 7 | 63 | 9 | | |
| | $\overline{IRQ_4}$ | 38 | 30 | 40 | | |
| | $\overline{IRQ_5}$ | 39 | 31 | 41 | | |
| | $\overline{IRQ_6}$ | 40 | 32 | 42 | | |
| | $\overline{IRQ_7}$ | 41 | 33 | 43 | | |
| Operating mode control | $MD_2$ | 14 | 6 | 16 | I | **Mode:** Input pins for setting the MCU operating mode according to the table below |
| | $MD_1$ | 13 | 5 | 15 | | |
| | $MD_0$ | 12 | 4 | 14 | | |

| MD2 | MD1 | MD0 | Mode | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | Mode 0 | — |
| 0 | 0 | 1 | Mode 1 | Expanded minimum mode (ROM disabled) |
| 0 | 1 | 0 | Mode 2 | Expanded minimum mode (ROM enabled) |
| 0 | 1 | 1 | Mode 3 | Expanded maximum mode (ROM disabled) |
| 1 | 0 | 0 | Mode 4 | Expanded maximum mode (ROM enabled) |
| 1 | 0 | 1 | Mode 5 | — |
| 1 | 1 | 0 | Mode 6 | Hardware standby mode |
| 1 | 1 | 1 | Mode 7 | Single-chip mode |

The inputs at these pins are latched in mode select bits 2 to 0 (MDS2 – MDS0) of the mode control register (MDCR) on the rising edge of the $\overline{RES}$ signal.

Table 1-5 Pin Functions (cont)

| Type | Symbol | Pin No. DC-64S DP-64S | Pin No. FP-64A | Pin No. CP-68 | I/O | Name and Function |
|---|---|---|---|---|---|---|
| 16-bit free-running timer (FRT) | FTOA1<br>FTOA2 | 49<br>50 | 41<br>42 | 51<br>52 | O | **FRT Output Compare A (channels 1 and 2):** Output pins for the output compare A function of free-running timer channels 1 and 2. |
| | FTOB1<br>FTOB2 | 47<br>48 | 39<br>40 | 49<br>50 | O | **FRT Output Compare B (channels 1 and 2):** Output pins for the output compare B function of free-running timer channels 1 and 2. |
| | FTCI1<br>FTCI2 | 47<br>48 | 39<br>40 | 49<br>50 | I | **FRT Counter Clock Input (channels 1 and 2):** External clock input pins for the free-running counters (FRCs) of free-running timer channels 1 and 2. |
| | FTI1<br>FTI2 | 44<br>45 | 36<br>37 | 46<br>47 | I | **FRT Input Capture (channels 1 and 2):** Input capture pins for free-running timer channels 1 and 2. |
| 8-bit timer | TMO | 46 | 38 | 48 | O | **8-bit Timer Output:** Compare-match output pin for the 8-bit timer. |
| | TMCI | 43 | 35 | 45 | I | **8-bit Timer Clock Input:** External clock input pin for the 8-bit timer counter. |
| | TMRI | 45 | 37 | 47 | I | **8-bit Timer Counter Reset Input:** A high input at this pin resets the 8-bit timer counter. |

## Table 1-5 Pin Functions (cont)

| | | Pin No. | | | | |
|---|---|---|---|---|---|---|
| Type | Symbol | DC-64S<br>DP-64S | FP-64A | CP-68 | I/O | Name and Function |
| Serial communication interface signals | TXD$_1$<br>TXD$_2$ | 61<br>58 | 53<br>50 | 67<br>64 | O | **Transmit Data (channels 1 and 2):** Data output pins for serial communication interface channels 1 and 2. |
| | RXD$_1$<br>RXD$_2$ | 62<br>59 | 54<br>51 | 68<br>65 | I | **Receive Data (channels 1 and 2):** Data input pins for serial communication interface channels 1 and 2. |
| | SCK$_1$<br>SCK$_2$ | 63<br>60 | 55<br>52 | 1<br>66 | I/O | **Serial Clock (channels 1 and 2):** Input/output pins for the serial interface clock. |
| A/D converter | AN$_3$ – AN$_0$<br>AN$_7$ – AN$_4$* | 56 – 53 | 48 – 45 | 58 – 55<br>62 – 59 | I | **Analog Input:** Analog signal input pins. |
| | AV$_{CC}$ | 57 | 49 | 63 | I | **Analog Reference Voltage:** Reference voltage pin for the A/D converter. |
| | AV$_{SS}$ | 52 | 44 | 54 | I | **Analog Ground:** Ground pin for the A/D converter. |
| | $\overline{\text{ADTRG}}$ | 5 | 61 | 7 | I | **A/D External Trigger:** External trigger input pin for the A/D converter. |
| Parallel I/O | P1$_7$ – P1$_0$ | 10 – 3 | 2 – 1,<br>64 - 59 | 12 – 5 | I/O | **Port 1:** An 8-bit input/output port. The direction of each bit is determined by the port 1 data direction register (P1DDR). |
| | P2$_7$ – P2$_0$ | 25 – 18 | 17 – 10 | 27 – 20 | I/O | **Port 2:** An 8-bit input/output port. The direction of each bit is determined by the port 2 data direction register (P2DDR). |
| | P3$_7$ – P3$_0$ | 33 – 26 | 25 – 18 | 35 – 28 | I/O | **Port 3:** An 8-bit input/output port. The direction of each bit is determined by the port 3 data direction register (P3DDR). These pins have built-in MOS input pull-ups. They can drive LED indicators. |
| | P4$_7$ – P4$_0$ | 41 – 34 | 33 – 26 | 43 – 36 | I/O | **Port 4:** An 8-bit input/output port. The direction of each bit is determined by the port 4 data direction register (P4DDR). These pins have built-in MOS input pull-ups. |

Note: * CP-68 only

### Table 1-5 Pin Functions (cont)

T-49-19-16

| Type | Symbol | Pin No.<br>DC-64S<br>DP-64S | <br>FP-64A | <br>CP-68 | I/O | Name and Function |
|---|---|---|---|---|---|---|
| Parallel I/O (cont) | $P5_7 - P5_0$ | 50 – 43 | 42 – 35 | 52 – 45 | I/O | **Port 5:** An 8-bit input/output port. The direction of each bit is determined by the port 5 data direction register (P5DDR). These pins have Schmitt inputs. |
| | $P6_3 - P6_0$<br>$P6_7 - P6_4$* | 56 – 53 | 48 – 45 | 58 – 55<br>62 – 59 | I | **Port 6:** A 4-bit (or 8-bit*) input port. |
| | $P7_5 - P7_0$ | 63 – 58 | 55 – 50 | 2 – 1,<br>68 – 64, | I/O | **Port 7:** A 6-bit input/output port. The direction of each bit is determined by the port 7 data direction register (P7DDR). |

Note: * CP-68 package only

# Section 2 MCU Operating Modes and Address Space

## 2.1 Overview

T-49-19-16

The H8/520 microcomputer unit (MCU) operates in five modes numbered 1, 2, 3, 4, and 7. The mode is selected by the inputs at the mode pins (MD2 to MD0) at the instant when the chip comes out of a reset. As indicated in table 2-1, the MCU mode determines the size of the address space, the usage of on-chip ROM, and the operating mode of the CPU. The MCU mode also affects the functions of I/O pins.

Table 2-1 Operating Modes

| MD2 | MD1 | MD0 | MCU Mode | Address Space | On-Chip RAM | On-Chip ROM | CPU Mode |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | Mode 0 | — | — | — | — |
| 0 | 0 | 1 | Mode 1 | Expanded minimum | Enabled* | Disabled | Minimum mode |
| 0 | 1 | 0 | Mode 2 | Expanded minimum | Enabled* | Enabled | Minimum mode |
| 0 | 1 | 1 | Mode 3 | Expanded maximum | Enabled* | Disabled | Maximum mode |
| 1 | 0 | 0 | Mode 4 | Expanded maximum | Enabled* | Enabled | Maximum mode |
| 1 | 0 | 1 | Mode 5 | — | — | — | — |
| 1 | 1 | 0 | Mode 6 | Hardware standby mode | — | — | — |
| 1 | 1 | 1 | Mode 7 | Single-chip only | Enabled* | Enabled | Minimum mode |

Notation: 0: Low level
1: High level
—: Cannot be used

Note: * On-chip RAM can be disabled by RAME bit to 0 in RAM control register (RAMCR).

Modes 1 to 4 are referred to as "expanded" because they permit access to off-chip memory and peripheral addresses. The expanded minimum modes (modes 1 and 2) support a maximum address space of 64 kbytes. The expanded maximum modes (modes 3 and 4) support a maximum address space of 1 Mbyte.

Interrupt service is slightly slower in the expanded maximum modes than in the other modes because the CPU has to save its code page register.

The H8/520 cannot be set to modes 0 and 5. The mode pins should never be set to these values. The hardware standby mode (mode 6) is a power-down mode, not an operating mode. See section 17.4, "Hardware Standby Mode" for details.

## 2.2 Mode Descriptions

T-49-19-16

The five MCU modes are described below. For further information on the I/O pin functions in each mode, see section 9, "I/O ports."

**Mode 1 (Expanded Minimum Mode):** Mode 1 supports a maximum 64-kbyte address space which does not include any on-chip ROM. Ports 1 to 4 are used for bus lines and bus control signals as follows:

| | |
|---|---|
| Control signals: | Port 1 (partly) |
| Data bus: | Port 2 |
| Address bus: | Ports 3 and 4 |

**Mode 2 (Expanded Minimum Mode):** Mode 2 supports a maximum 64-kbyte address space of which the first 16 kbytes are in on-chip ROM. Ports 1 to 4 are used for bus lines and bus control signals as follows:

| | |
|---|---|
| Control signals: | Port 1 (partly) |
| Data bus: | Port 2 |
| Address bus: | Ports 3 and 4 |

Note: In mode 2, port 4 is initially a general-purpose input port. Software must change the desired pins to output before using them for the address bus. See section 9.5, "Port 4" for details. The following instruction makes all pins of port 4 into output pins:

```
MOV.B  #H'FF, @H'FF85
```

**Mode 3 (Expanded Maximum Mode):** Mode 3 supports a maximum 1-Mbyte address space which does not include any on-chip ROM. Ports 1 to 4 and one pin in port 7 are used for bus lines and bus control signals as follows:

| | |
|---|---|
| Control signals: | Port 1 (partly) |
| Data bus: | Port 2 |
| Address bus: | Ports 1 (partly), 3, 4, and 7 (partly) |

**Mode 4 (Expanded Maximum Mode):** Mode 4 supports a maximum 1-Mbyte address space of which the first 16 kbytes are in on-chip ROM. Ports 1 to 4 and one pin in port 7 are used for bus lines and bus control signals as follows:

T-49-19-16

| | |
|---|---|
| Control signals: | Port 1 (partly) |
| Data bus: | Port 2 |
| Address bus: | Ports 1 (partly), 3, 4, and 7 (partly) |

Note: In mode 4, port 4, pins 2 to 4 of port 1, and pin 2 of port 7 are initially used for general-purpose input. Software must change the desired pins to output before using them for the address bus. See section 9, "I/O Ports" for details.

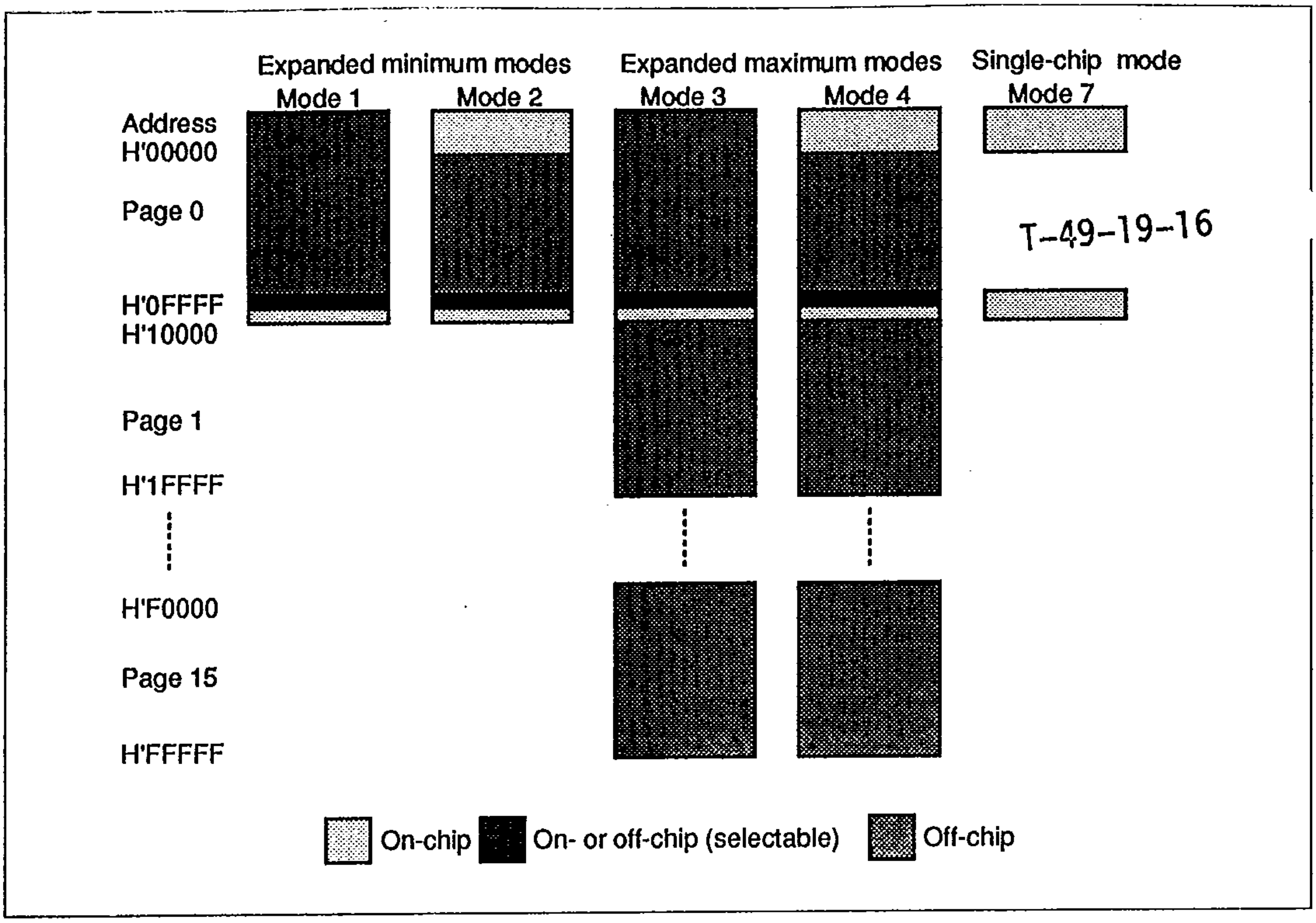
**Mode 7 (Single-Chip Mode):** In this mode all memory is on-chip, in 16 kbytes of ROM and 512 bytes of RAM. It is not possible to access off-chip addresses.

The single-chip mode provides the maximum number of ports. All the pins associated with the address and data buses in the expanded modes are available as general-purpose input/output ports in the single-chip mode.

## 2.3 Address Space Map

### 2.3.1 Page Segmentation

The H8/520's address space is segmented into 64-kbyte pages. In the single-chip mode and expanded minimum modes there is just one page: page 0. In the expanded maximum modes there can be up to 16 pages. Figure 2-1 shows the address space in each mode and indicates which parts are on- and off-chip.



T-49-19-16

Figure 2-1 Address Space in Each Mode

### 2.3.2 Page 0 Address Allocations

T-49-19-16

The high and low address areas in page 0 are reserved for registers and vector tables.

**Vector Tables:** The low address contains the exception vector table and DTC vector table. The CPU accesses the exception vector table to obtain the addresses of user-coded exception-handling routines. The DTC vector table contains pointers to tables of register information used by the on-chip chip data transfer controller. The size of these tables depends on the CPU operating mode. Details are given in section 4.1.2, "Exception Sources and Vector Table," section 5.2.3, "Interrupt Vector Table," and section 6.3.2, "DTC Vector Table."

In modes 2, 4, and 7 the vector tables are located in on-chip ROM. In modes 1 and 3 the vector tables are in external memory.

**Register Field:** The highest 128 addresses in page 0 (addresses H'FF80 to H'FFFF) belong to control, status, and data registers used by the I/O ports and on-chip supporting modules. Program code cannot be located at these addresses.

The CPU accesses addresses in this register field like other addresses in the address space. By reading and writing at these addresses the CPU controls the on-chip supporting modules and communicates via the I/O ports. A complete map of the register field is given in appendix B.

**On-Chip RAM:** One of the control registers in the register field is a RAM control register (RAMCR) containing a RAM enable bit (RAME) that enables or disables the 512-byte on-chip RAM. When this bit is set to 1 (its default value), addresses H'FD80 to H'FF7F are located on-chip. When this bit is cleared to 0, these addresses are located in external memory and the on-chip RAM is not used. See section 15, "RAM", for further information.

The RAME bit is bit 7 at address H'FFF9.

**Coding Example:**

| | |
|---|---|
| To enable on-chip RAM: | `BSET.B #7, H'FFF9` |
| To disable on-chip RAM: | `BCLR.B #7, H'FFF9` |

Note: If on-chip RAM is disabled in the single-chip mode, access to addresses H'FD80 to H'FF7F causes an address error.

Figure 2-2 is a map of page 0 of the address space.

T-49-19-16

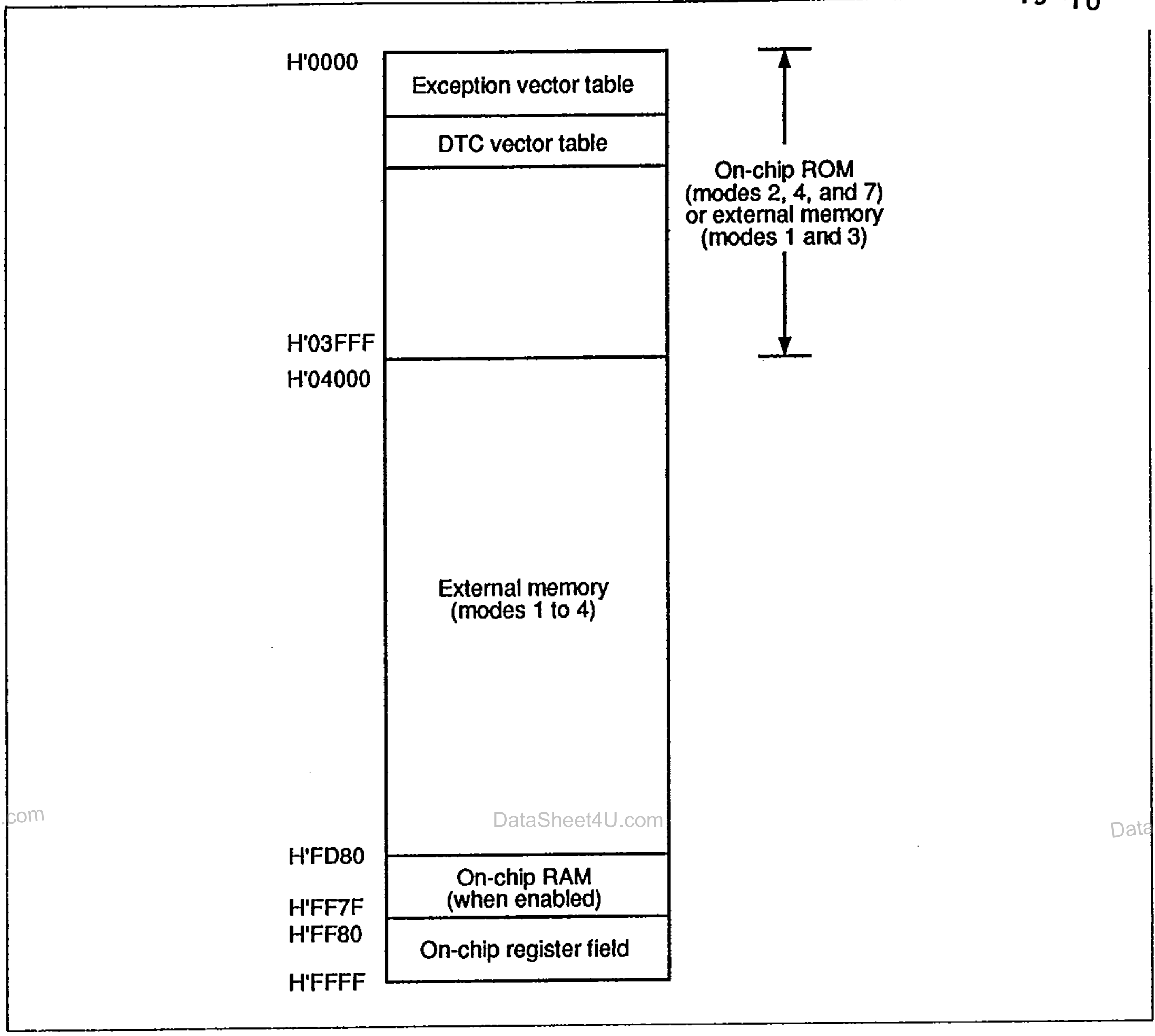


**Figure 2-2 Map of Page 0**

## 2.4 Mode Control Register (MDCR)

Another control register in the register field in page 0 is the mode control register (MDCR). The inputs at the mode pins are latched in this register on the rising edge of the signal. The mode control register can be read by the CPU, but not written. Table 2-2 lists the attributes of this register.

**Table 2-2 Mode Control Register**

| Name | Abbreviation | Read/Write | Address |
|---|---|---|---|
| Mode control register | MDCR | Read only | H'FFFA |

The bit configuration of this register is shown below.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | MDS2 | MDS1 | MDS0 |
| Initial value | 1 | 1 | 0 | 0 | 0 | —* | —* | —* |
| Read/Write | — | — | — | — | — | R | R | R |

Note: * Initialized according to MD$_2$ to MD$_0$.

**Bits 7 and 6—Reserved:** These bits cannot be modified and are always read as 1.

**Bits 5 to 3—Reserved:** These bits cannot be modified and are always read as 0.

**Bits 2 to 0—Mode Select 2 to 0 (MDS2 to MDS0):** These bits indicate the values of the mode pins (MD$_2$ to MD$_0$) latched on the rising edge of the $\overline{\text{RES}}$ signal. MDS2 corresponds to MD$_2$, MDS1 to MD$_1$, and MDS0 to MD$_0$. These bits can be read but not written.

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

```
CMP:G.B #H'C1, @H'FFFA
```

The comparison is with H'C1 instead of H'01 because bits 7 and 6 are always read as 1.

# Section 3 CPU

T-49-19-16

## 3.1 Overview

The H8/520 chip has the H8/500 Family CPU: a high-speed central processing unit designed for real-time control of a wide range of medium-scale office and industrial equipment. It features eight 16-bit general registers, internal 16-bit data paths, and an optimized instruction set.

Section 3 summarizes the CPU architecture and instruction set.

### 3.1.1 Features

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

- General-register machine
  — Eight 16-bit general registers
  — Seven control registers (two 16-bit registers, five 8-bit registers)
- High speed: maximum 10-MHz clock
  At 10 MHz a register-register add operation takes only 200 ns.
- Address space managed in 64-kbyte pages, expandable to 1 Mbyte*
  Page registers make four pages available simultaneously: a code page, stack page, data page, and extended page.
- Two CPU operating modes:
  — Minimum mode: Maximum 64-kbyte address space
  — Maximum mode: Maximum 1-Mbyte address space*
- Highly orthogonal instruction set
  Addressing modes and data sizes can be specified independently within each instruction.
- 1.5 addressing modes
  Register-register and register-memory operations are supported.
- Optimized for efficient programming in C language
  In addition to the general registers and orthogonal instruction set, the CPU has special short formats for frequently-used instructions and addressing modes.

Note: * The CPU Architecture supports up to 16 Mbytes of external memory, but the H8/520 chip has only enough address pins to address 1 Mbyte.

### 3.1.2 Address Space

T-49-19-16

The address space size depends on the operating mode.

The H8/520 MCU has five operating modes, which are selected by the input to the mode pins (MD2 to MD0) when the chip comes out of a reset. The CPU, however, has only two operating modes. The MCU operating mode determines the CPU operating mode, which in turn determines the maximum address space size as indicated in figure 3-1.

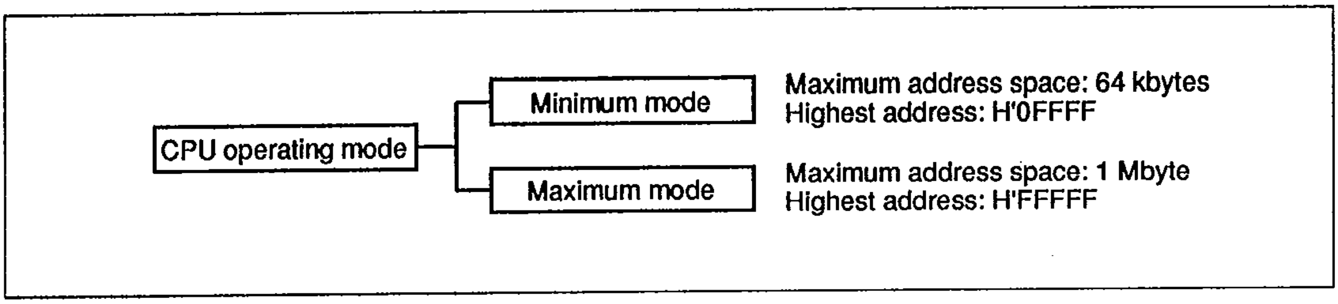


Figure 3-1 CPU Operating Modes

### 3.1.3 Register Configuration

T-49-19-16

Figure 3-2 shows the register structure of the CPU. There are two groups of registers: the general registers (Rn) and control registers (CR).

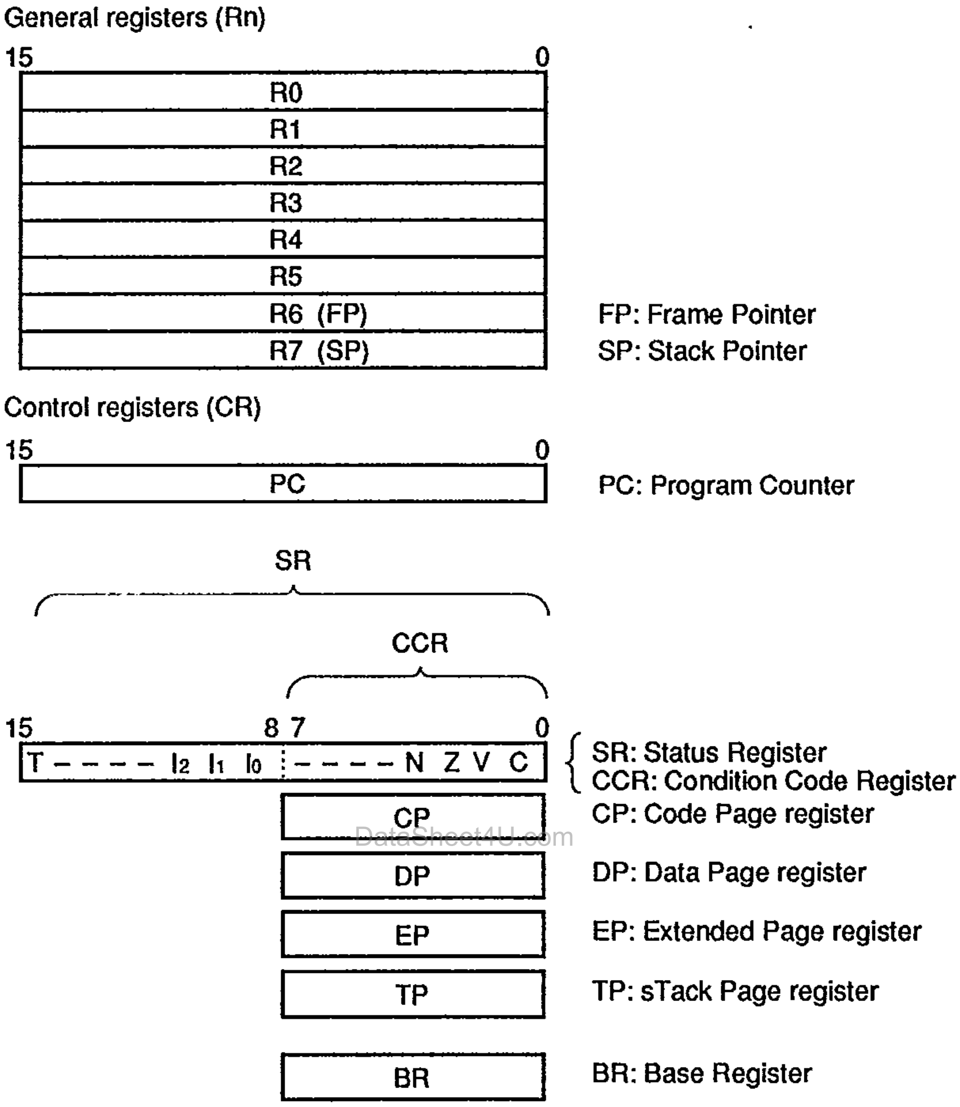


**Figure 3-2 Registers in the CPU**

## 3.2 CPU Register Descriptions

### 3.2.1 General Registers

All eight of the 16-bit general registers are functionally alike; there is no distinction between data registers and address registers. When these registers are accessed as data registers, either byte or word size can be selected.

R6 and R7, in addition to functioning as general registers, have special assignments.

R7 is the stack pointer, used implicitly in exception handling and subroutine calls. It can be designated by the name SP, which is synonymous with R7. As indicated in figure 3-3, it points to the top of the stack. It is also used implicitly by the LDM and STM instructions, which load and store multiple registers from and to the stack and pre-decrement or post-increment R7 accordingly.

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

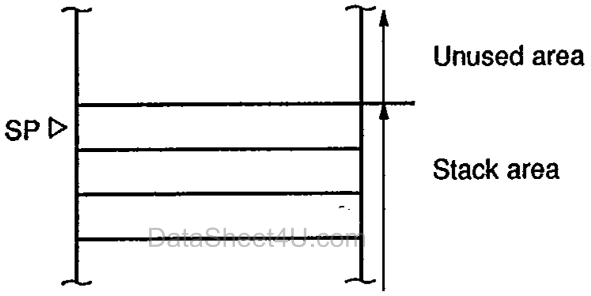


Figure 3-3 Stack Pointer

### 3.2.2 Control Registers

T-49-19-16

The CPU control registers include a 16-bit program counter (PC), a 16-bit status register (SR), four 8-bit page registers, and one 8-bit base register (BR).

**Program Counter (PC):** This 16-bit register indicates the address of the next instruction the CPU will execute.

**Status Register (SR):** This 16-bit register contains internal status information. The lower half of the status register is referred to as the condition code register (CCR): it can be accessed as a separate condition code byte.

CCR (bits 7 to 0)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | T | — | — | — | — | $I_2$ | $I_1$ | $I_0$ | — | — | — | — | N | Z | V | C |

**Bit 15—Trace (T):** When this bit is set to 1, the CPU operates in trace mode and generates a trace exception after every instruction. See section 4.4, "Trace", for a description of the trace exception-handling sequence.

When the value of this bit is 0, instructions are executed in normal continuous sequence. This bit is cleared to 0 at a reset.

**Bits 14 to 11—Reserved:** These bits cannot be modified and are always read as 0.

**Bits 10 to 8—Interrupt Mask ($I_2$, $I_1$, $I_0$):** These bits indicate the interrupt request mask level (0 to 7). As shown in table 3-1, an interrupt request is not accepted unless it has a higher level than the value of the mask. A nonmaskable interrupt (NMI), which has level 8, is accepted at any mask level. After an interrupt is accepted, $I_2$, $I_1$, and $I_0$ are changed to the level of the interrupt. Table 3-2 indicates the values of the I bits after the interrupt is accepted.

A reset sets all three bits ($I_2$, $I_1$, and $I_0$) to 1, masking all interrupts except NMI.

T-49-19-16

## Table 3-1 Interrupt Mask Levels

| | Mask | Mask Bits | | | |
|---|---|---|---|---|---|
| **Priority** | **Level** | **$I_2$** | **$I_1$** | **$I_0$** | **Interrupts Accepted** |
| High | 7 | 1 | 1 | 1 | NMI |
| ↑ | 6 | 1 | 1 | 0 | Level 7 and NMI |
| | 5 | 1 | 0 | 1 | Levels 6 to 7 and NMI |
| | 4 | 1 | 0 | 0 | Levels 5 to 7 and NMI |
| | 3 | 0 | 1 | 1 | Levels 4 to 7 and NMI |
| | 2 | 0 | 1 | 0 | Levels 3 to 7 and NMI |
| | 1 | 0 | 0 | 1 | Levels 2 to 7 and NMI |
| Low | 0 | 0 | 0 | 0 | Levels 1 to 7 and NMI |

## Table 3-2 Interrupt Mask Bits after an Interrupt is Accepted

| Level of Interrupt Accepted | $I_2$ | $I_1$ | $I_0$ |
|---|---|---|---|
| NMI (8) | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 |
| 6 | 1 | 1 | 0 |
| 5 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |

**Bits 7 to 4—Reserved:** These bits cannot be modified and are always read as 0.

**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 set to 1 when a carry or borrow occurs at the most significant bit, and is cleared to 0 (or left unchanged) at other times.

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

**Page Registers:** The code page register (CP), data page register (DP), extended page register (EP), and stack page register (TP) are 8-bit registers that are used only in the maximum mode. No use of their contents is made in the minimum mode.

In the maximum mode, the page registers combine with the program counter and general registers to generate 24-bit effective addresses as shown in figure 3-4, thereby expanding the program area, data area, and stack area.

T-49-19-16

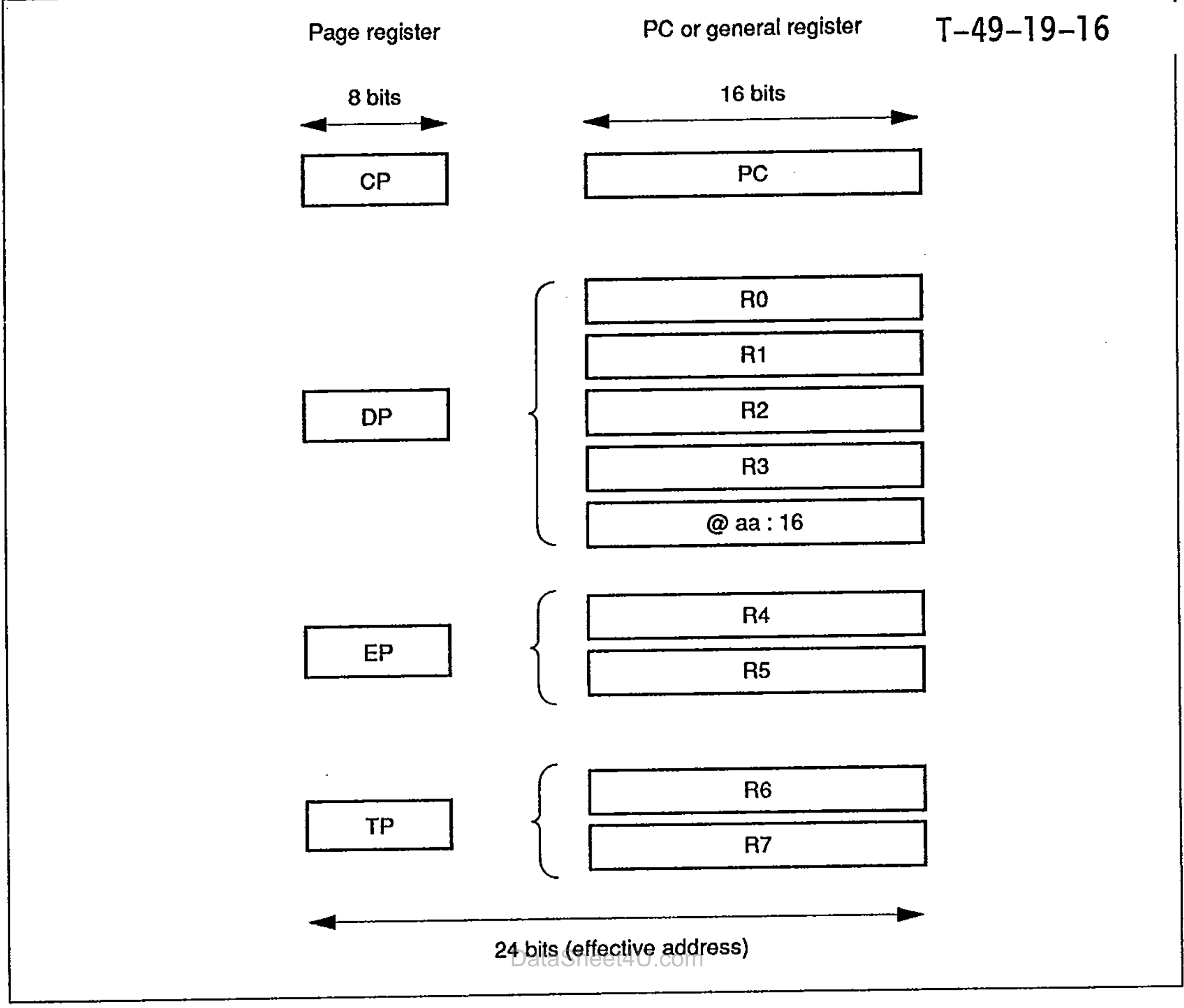


Figure 3-4 Combinations of Page Registers with Other Registers

**Code Page Register (CP):** The code page register and the program counter combine to generate at 24-bit program code address. The code page register contains the upper 8 bits of the address. In the maximum mode, the code page register is initialized at a reset to a value loaded from the vector table, and both the code page register and program counter are saved and restored in exception handling.

**Data Page Register (DP):** The data page register combines with general registers R0 to R3 to generate a 24-bit effective address. The data page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R0 to R3, and in the 16-bit absolute addressing mode (@aa:16), but not in the short absolute addressing mode (@aa:8).

The data page register is rewritten by the LDC instruction.

**Extended Page Register (EP):** The extended page register combines with general register R4 or R5 to generate a 24-bit operand address. The extended page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R4 or R5.

The extended page can be used as an additional data page.

T-49-19-16

**Stack Page Register (TP):** The stack page register combines with R6 (FP) or R7 (SP) to generate a 24-bit stack address. The stack page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R6 or R7, in exception handling, and in subroutine calls.

**Base Register (BR):** This 8-bit register stores the base address used in the short absolute addressing mode (@aa:8). In this addressing mode a 16-bit effective address in page 0 is generated by using the contents of the base register as the upper 8 bits and an address given in the instruction code as the lower 8 bits. See figure 3-5.

In the short absolute addressing mode the address is always located in page 0.

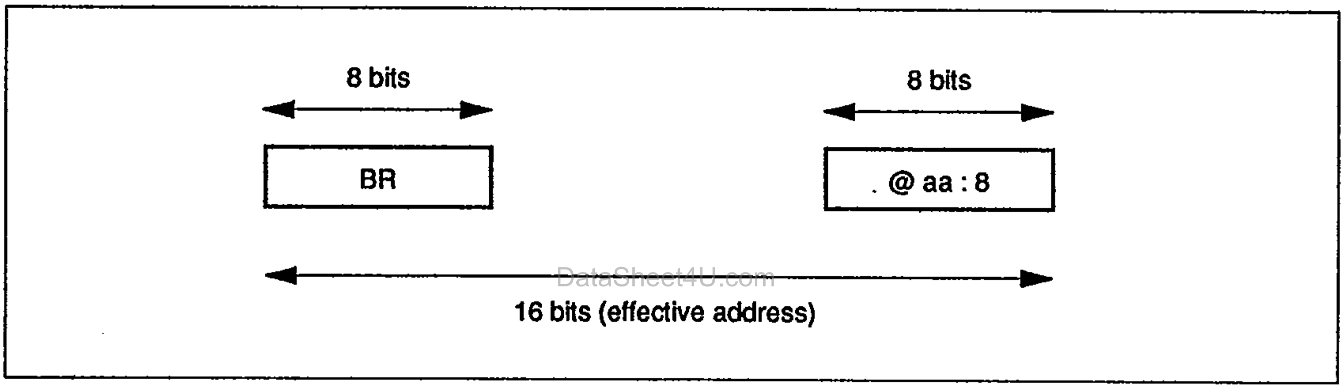


Figure 3-5 Short Absolute Addressing Mode and Base Register

### 3.2.3 Initial Register Values

When the CPU is reset, its internal registers are initialized as shown in table 3-3. Note that the stack pointer (R7) and base register (BR) are not initialized to fixed values. Also, of the page registers used in maximum mode, only the code page register (CP) is initialized; the other three page registers come out of the reset state with undetermined values.

Accordingly, in the minimum mode the first instruction executed after a reset should initialize the stack pointer. The base register must also be initialized before the short absolute addressing mode (@aa:8) is used.

In the maximum mode, the first instruction executed after a reset should initialize the stack page register (TP) and the next instruction should initialize the stack pointer. Later instructions should initialize the base register and the other page registers as necessary.

**Table 3-3 Initial Values of Registers**

| Register | Initial Value | |
|---|---|---|
| | **Minimum Mode** | **Maximum Mode** |
| **General registers** | | |
| 15 – 0<br>R7 - R0 | Undetermined | Undetermined |
| **Control registers** | | |
| 15 – 0<br>PC | Loaded from vector table | Loaded from vector table |
| SR (15 – 0), CCR (7 – 0)<br>15 – 8: T – – – – $I_2 I_1 I_0$<br>7 – 0: – – – – NZVC | H'070*<br>(*: undetermined) | H'070*<br>(*: undetermined) |
| 7 – 0<br>CP | Undetermined | Loaded from vector table |
| 7 – 0<br>DP | Undetermined | Undetermined |
| 7 – 0<br>EP | Undetermined | Undetermined |
| 7 – 0<br>TP | Undetermined | Undetermined |
| 7 – 0<br>BR | Undetermined | Undetermined |

## 3.3 Data Formats

The H8/500 can process 1-bit data, 4-bit BCD data, 8-bit (byte) data, 16-bit (word) data, and 32-bit (longword) data.

- Bit manipulation instructions operate on 1-bit data.
- Decimal arithmetic instructions operate on 4-bit BCD data.
- Almost all instructions operate on byte and word data.
- Multiply and divide instructions operate on longword data.

### 3.3.1 Data Formats in General Registers

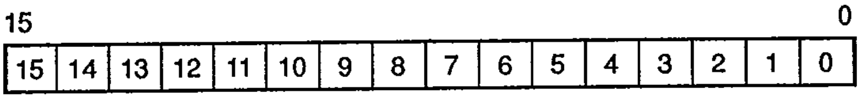
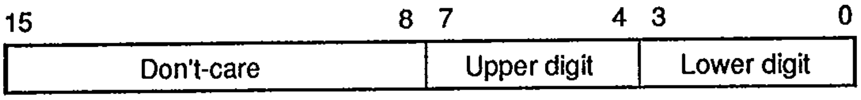
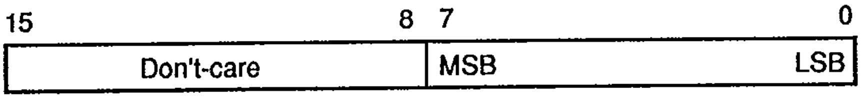
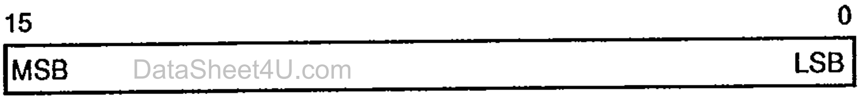
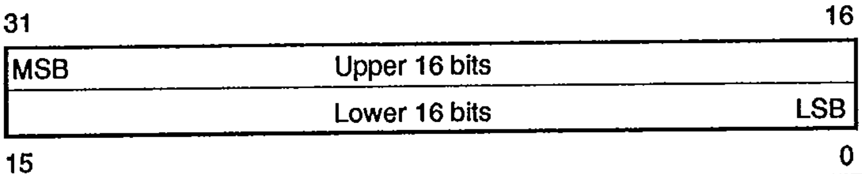
T-49-19-16

Data of all the sizes above can be stored in general registers as shown in table 3-4.

Bit data locations are specified by bit number. Bit 15 is the most significant bit. Bit 0 is the least significant bit. BCD and byte data are stored in the lower 8 bits of a general register. Word data use all 16 bits of a general register. Longword data use two general registers: the upper 16 bits are stored in Rn (n must be an even number); the lower 16 bits are stored in Rn+1.

Operations performed on BCD data or byte data do not affect the upper 8 bits of the register.

Table 3-4 General Register Data Formats

| Data Type | Register No. | Data Structure |
|---|---|---|
| 1-Bit | Rn |  |
| BCD | Rn |  |
| Byte | Rn |  |
| Word | Rn |  |
| Longword | Rn*<br>Rn + 1* |  |

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

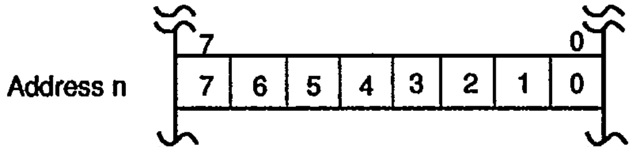
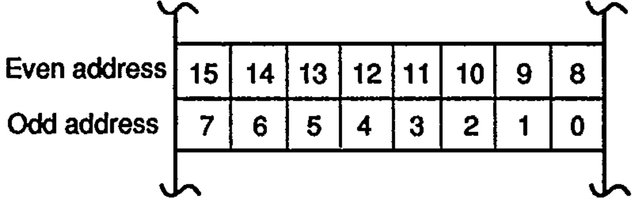
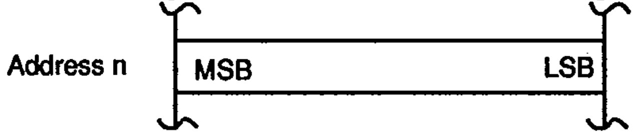
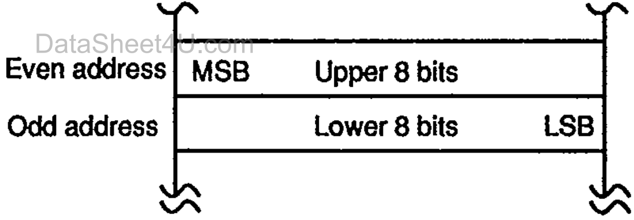
### 3.3.2 Data Formats in Memory

T-49-19-16

Table 3-5 indicates the data formats in memory.

Instructions that access bit data in memory have byte or word operands. The instruction specifies a bit number to indicate a specific bit in the operand.
Access to word data in memory must always begin at an even address. Access to word data starting at an odd address causes an address error. The upper 8 bits of word data are stored in address n (where n is an even number); the lower 8 bits are stored in address n+1.

**Table 3-5 Data Formats in Memory**

| Data Type | Data Format |
|---|---|
| 1-bit (in byte operand data) |  |
| 1-bit (in word operand data) |  |
| Byte |  |
| Word |  |

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

Table 3-6 shows the data formats on the stack.

**Table 3-6 Data Formats on the Stack**

| Data Type | Data Format | |
|---|---|---|
| Byte data on stack | Even address | Don't-care |
| | Odd address | MSB LSB |
| Word data on stack | Even address | MSB Upper 8 bits |
| | Odd address | Lower 8 bits LSB |

## 3.4 Instructions

### 3.4.1 Basic Instruction Formats

There are two basic CPU instruction formats: the general format and the special format.

**General Format:** This format consists of an effective address (EA) field, an effective address extension field, and an operation code (OP) field. The effective address is placed before the operation code because this results in faster execution of the instruction.

| | | |
|---|---|---|
| Effective address field | Effective address extension | Operation code |

- Effective address field: One byte containing information used to calculate the effective address of an operand.
- Effective address extension: Zero to two bytes containing a displacement value, immediate data, or an absolute address. The size of the effective address extension is specified in the effective address field.
- Operation code: Defines the operation to be carried out on the operand located at the address calculated from the effective address information. Some instructions (DADD, DSUB) have an extended format in which the operand code is preceded by a one-byte prefix code.

T-49-19-16

- Example of prefix code in DADD instruction

| Effective address | Prefix code | Operation code |
|---|---|---|
| 10100rrr | 00000000 | 10100rrr |

**Special Format:** In this format the operation code comes first, followed by the effective address field and effective address extension. This format is used in branching instructions, system control instructions, and other instructions that can be executed faster if the operation is specified before the operand.

| | | |
|---|---|---|
| Operation code | Effective address field | Effective address extension |

- Operation code: One or two bytes defining the operation to be performed by the instruction.

- Effective address field and effective address extension: Zero to three bytes containing information used to calculate an effective address.

### 3.4.2 Addressing Modes

The CPU supports 7 addressing modes: (1) register direct; (2) register indirect; (3) register indirect with displacement; (4) register indirect with pre-decrement or post-increment; (5) immediate; (6) absolute; and (7) PC-relative.

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

In most instructions, the addressing mode is specified in the effective address field. The effective-address extension, if present, contains a displacement, immediate data, or an absolute address.

Table 3-7 indicates how the addressing mode is specified in the effective address field.

## Table 3-7 Addressing Modes

T-49-19-16

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

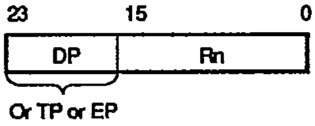
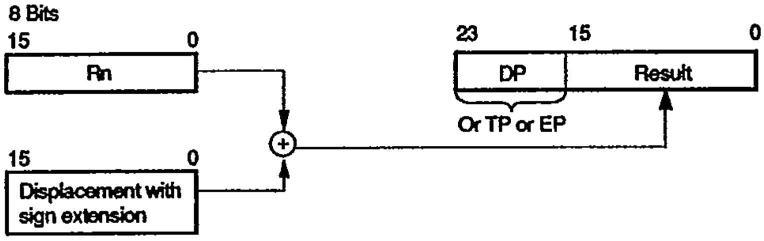
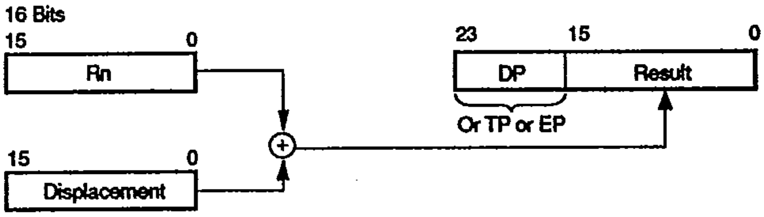
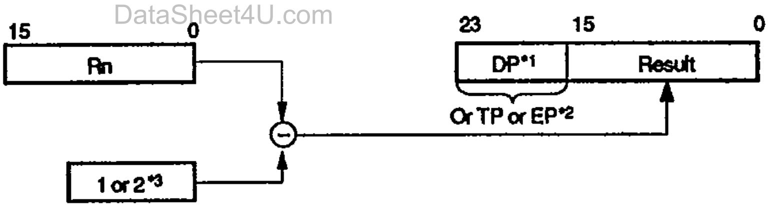
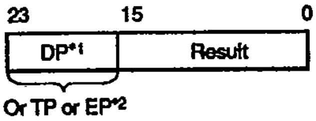
Notes: 1. Sz: Specifies the operand size.
When Sz = 0: byte operand
When Sz = 1: word operand
2. r r r: Register number field, specifying a general register number.
0 0 0—$R_0$ 0 0 1—$R_1$ 0 1 0—$R_2$ 0 1 1—$R_3$
1 0 0—$R_4$ 1 0 1—$R_5$ 1 1 0—$R_6$ 1 1 1—$R_7$
3. The @aa:8 addressing mode is also referred to as the short absolute addressing mode.

### 3.4.3 Effective Address Calculation

Table 3-8 explains how the effective address is calculated in each addressing mode.

Table 3-8 Effective Address Calculation

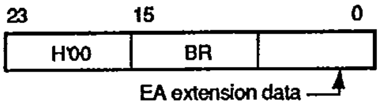
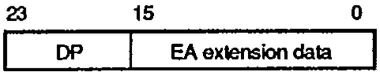
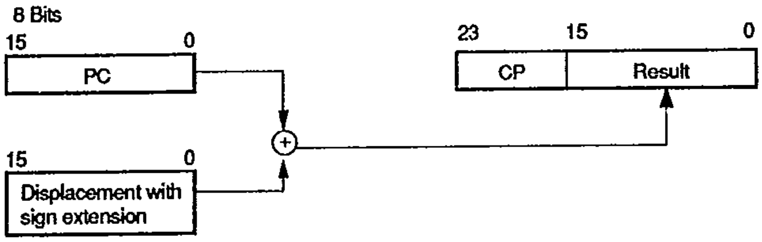
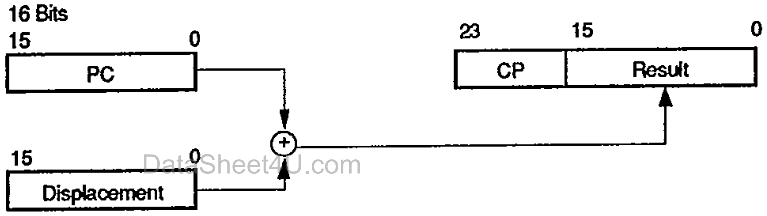
T-49-19-16

| No. | Addressing Mode | Effective Address Calculation | Effective Address |
|---|---|---|---|
| 1 | Register direct<br>Rn<br>1010Sz \| rrr | — | Operand is contents of Rn |
| 2 | Register indirect<br>@Rn<br>1101Sz \| rrr | — |  |
| 3 | Register indirect with displacement<br>@(d:8, Rn)<br>1110Sz \| rrr |  | |
| | @(d:16, Rn)<br>1111Sz \| rrr |  | |
| 4 | ① Register indirect with pre-decrement<br>@-Rn<br>1011Sz \| rrr |  <br>Rn is decremented by –1 or –2 before instruction execution. *4 | |
| | ② Register indirect with post-increment<br>@Rn +<br>1100Sz \| rrr | —<br>Rn is incremented by +1 or +2 after instruction execution. *3, 4 |  |

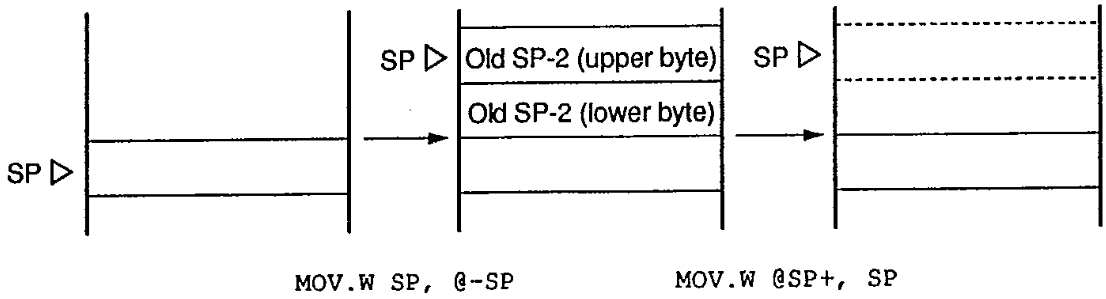
Notes:
1. The page register is ignored in minimum mode.
2. The page register used in addressing modes 2, 3, and 4 depends on the general register: DP for R0, R1, R2, or R3; EP for R4 or R5; TP for R6 or R7.
3. Decrement by –1 for a byte operand, and by –2 for a word operand.
4. The pre-decrement or post-increment is always ±2 when R7 is specified, even if the operand is byte size.

**Table 3-8 Effective Address Calculation (cont)**

T-49-19-16

| No. | Addressing Mode | Effective Address Calculation | Effective Address |
|---|---|---|---|
| 5 | Absolute address<br>@aa:8<br>0000Sz101 | — |  |
|  | @aa:16<br>0001Sz101 | — |  |
| 6 | Immediate<br>#xx:8<br>00000100 | — | Operand is 1-byte EA extension data |
|  | #xx:16<br>00001100 | — | Operand is 2-byte EA extension data |
| 7 | PC-relative<br>disp:8<br>No EA code<br>Specified in OP code |  |  |
|  | disp:16<br>No EA code<br>Specified in OP code |  |  |

Note: The drawing below shows what happens when the @–SP and @ SP+ addressing modes are used to save and restore the stack pointer.



## 3.5 Instruction Set

### 3.5.1 Overview

T-49-19-16

The main features of the CPU instruction set are:

- A general-register architecture.
- Orthogonality. Addressing modes and data sizes can be specified independently in each instruction.
- Register-register and register-memory operations are supported.
- Affinity for high-level languages, particularly C, with short formats for frequently-used instructions and addressing modes.
- Standard mnemonics, common throughout the H Series.

The CPU instruction set includes 61 (63)*1 types of instructions, listed by function in table 3-9.

**Table 3-9 Instruction Classification**

| Function | Instructions | Types |
|---|---|---|
| Data transfer | MOV, LDM, STM, XCH, SWAP, (MOVTPE, MOVFPE)*1 | 5 (7)*1 |
| Arithmetic operations | ADD, SUB, ADDS, SUBS, ADDX, SUBX, DADD, DSUB, MULXU, DIVXU, CMP, EXTS, EXTU, TST, NEG, CLR, TAS | 17 |
| Logic operations | AND, OR, XOR, NOT | 4 |
| Shift | SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR | 8 |
| Bit manipulation | BSET, BCLR, BTST, BNOT | 4 |
| Branch | Bcc*2, JMP, PJMP, BSR, JSR, PJSR, RTS, PRTD, PRTS, RTD, SCB (/F, /NE, /EQ) | 11 |
| System control | TRAPA, TRAP/VS, RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP, LINK, UNLK | 12 |
|  | Total | 61 (63)*1 |

Notes: 1. The H8/520 chip does not have an E clock output pin, so it does not support the MOVTPE and MOVFPE instructions. H8/520 software should not use these instructions.
2. Bcc is a conditional branch instruction in which cc represents a condition code.

Tables 3-10 to 3-16 give a concise summary of the instructions in each functional category. The MOV, ADD, and CMP instructions have special short formats, which are listed in table 3-17. For detailed descriptions of the instructions, refer to the *H8/500 Series Programming Manual.*

The notation used in tables 3-10 to 3-17 is defined below.

T-49-19-16

| Operation | Notation |
|---|---|
| Rd | General register (destination) |
| Rs | General register (source) |
| Rn | General register |
| (EAd) | Destination operand |
| (EAs) | Source operand |
| CCR | Condition code register |
| N | N (negative) bit of CCR |
| Z | Z (zero) bit of CCR |
| V | V (overflow) bit of CCR |
| C | C (carry) bit of CCR |
| CR | Control register |
| PC | Program counter |
| CP | Code page register |
| SP | Stack pointer |
| FP | Frame pointer |
| #IMM | Immediate data |
| disp | Displacement |
| + | Addition |
| − | Subtraction |
| × | Multiplication |
| ÷ | Division |
| ∧ | AND logical |
| ∨ | OR logical |
| ⊕ | Exclusive OR logical |
| → | Move |
| ↔ | Exchange |
| ¬ | Not |

### 3.5.2 Data Transfer Instructions

T-49-19-16

Table 3-10 describes the seven data transfer instructions.

**Table 3-10 Data Transfer Instructions**

| Instruction | | Size*2 | Function |
|---|---|---|---|
| Data transfer | MOV | | (EAs) → (EAd), #IMM → (EAd) |
| | MOV:G | B/W | Moves data between two general registers, or between a general register and memory, or moves immediate data to a general register or memory. |
| | MOV:E | B | |
| | MOV:I | W | |
| | MOV:F | B/W | |
| | MOV:L | B/W | |
| | MOV:S | B/W | |
| | LDM | W | Stack → Rn (register list)<br>Pops data from the stack to one or more registers. |
| | STM | W | Rn (register list) → stack<br>Pushes data from one or more registers onto the stack. |
| | XCH | W | Rs ↔ Rd<br>Exchanges data between two general registers. |
| | SWAP | B | Rd (upper byte) ↔ Rd (lower byte)<br>Exchanges the upper and lower bytes in a general register. |
| | (MOVTPE) *1 | — | Not supported by the H8/520 |
| | (MOVFPE) *1 | — | Not supported by the H8/520 |

Notes: 1. The H8/520 does not have an E clock output pin, so it does not support the MOVTPE and MOVFPE instructions. H8/520 software should not use these instructions.
If the MOVTPE and MOVFPE instructions are used, the H8/520 executes them in the number of cycles indicated in figures A and B.
From 7 to 14 wait states (Tw) are automatically inserted between the $T_2$ state and $T_3$ state to synchronize the bus cycle with an internal E clock obtained by dividing the system clock (ø) by eight. Accordingly, the number of cycles taken by a MOVTPE or MOVFPE instruction varies. Note that no wait states (Tw) are inserted by the wait state controller.
2. B: Byte, W: Word

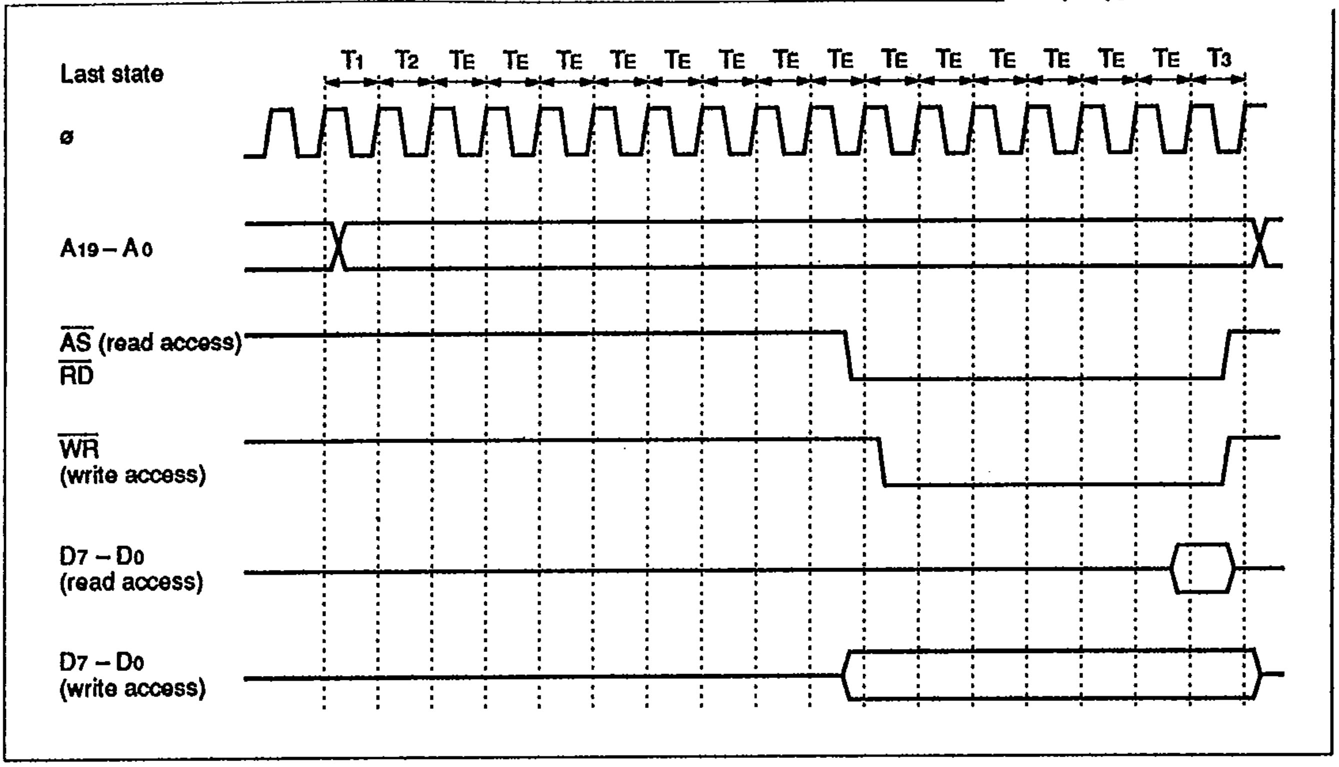


Figure A Execution Cycle Length of MOVTPE and MOVFPE Instructions in Expanded Modes (Maximum Number of Cycles)

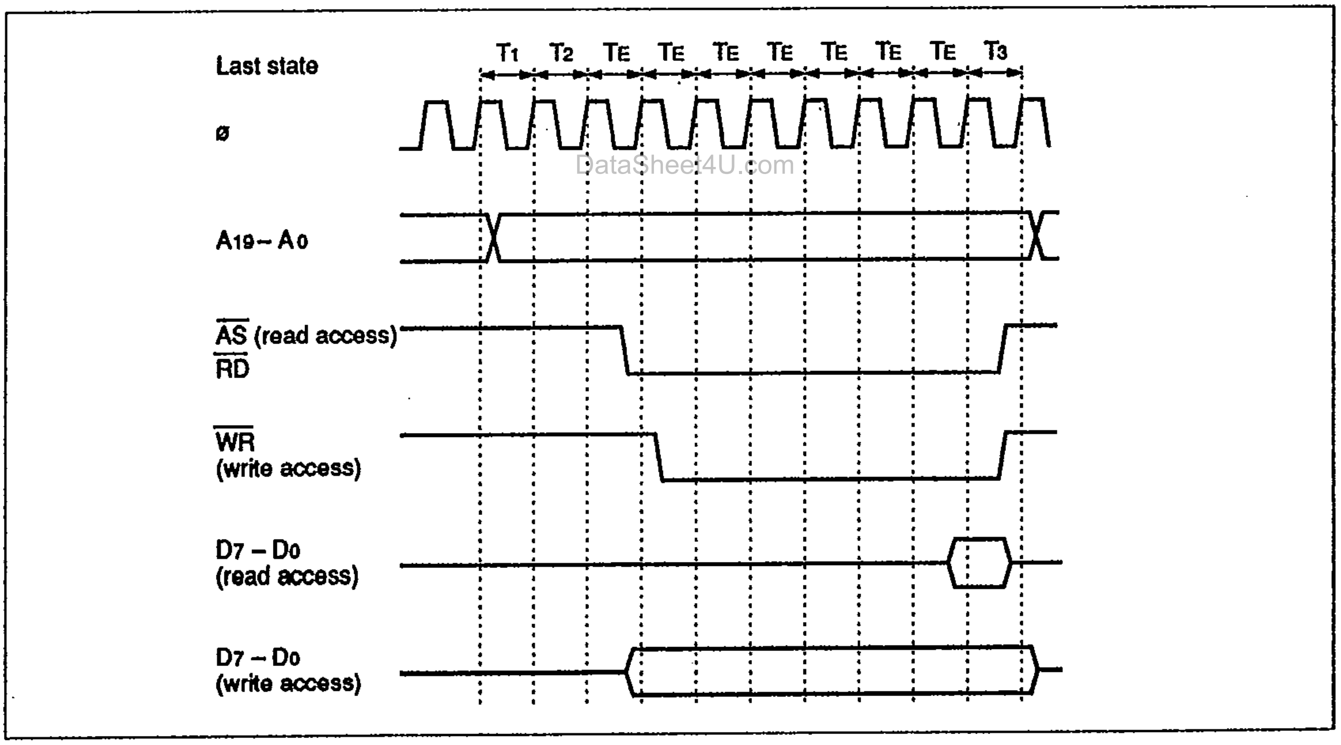


Figure B Execution Cycle Length of MOVTPE and MOVFPE Instructions in Expanded Modes (Minimum Number of Cycles)

### 3.5.3 Arithmetic Instructions

Table 3-11 describes the 17 arithmetic instructions.

**Table 3-11 Arithmetic Instructions**

T-49-19-16

| Instruction | | Size | Function |
|---|---|---|---|
| Arithmetic operations | ADD | | Rd ± (EAs) → Rd, (EAd) ± #IMM → (EAd)<br>Performs addition or subtraction on data in a general register and data in another general register or memory, or on data in a general register or memory and immediate data. |
| | ADD:G | B/W | |
| | ADD:Q | B/W | |
| | SUB | B/W | |
| | ADDS | B/W | |
| | SUBS | B/W | |
| | ADDX | B/W | Rd ± (EAs) ± C → Rd<br>Performs addition or subtraction with carry or borrow on data in a general register and data in another general register or memory, or on data in a general register and immediate data. |
| | SUBX | B/W | |
| | DADD | B | $(Rd)_{10} \pm (Rs)_{10} \pm C \rightarrow (Rd)_{10}$<br>Performs decimal addition or subtraction on data in two general registers. |
| | DSUB | B | |
| | MULXU | B/W | Rn × (EAs) → Rd<br>Performs 8-bit × 8-bit or 16-bit × 16-bit unsigned multiplication on data in a general register and data in another general register or memory, or on data in a general register and immediate data. |
| | DIVXU | B/W | Rd ÷ (EAs) → Rd<br>Performs 16-bit ÷ 8-bit or 32-bit ÷ 16-bit unsigned division on data in a general register and data in another general register or memory, or on data in a general register and immediate data. |
| | CMP | | Rn − (EAs), (EAd) − #IMM<br>Compares data in a general register with data in another general register or memory, or with immediate data, or compares data in memory with immediate data. |
| | CMP:G | B/W | |
| | CMP:E | B | |
| | CMP:I | W | |

T-49-19-16

**Table 3-11 Arithmetic Instructions (cont)**

| Instruction | | Size | Function |
|---|---|---|---|
| Arithmetic operations | EXTS | B | (<bit 7> of <Rd>) → (<bits 15 to 8>of <Rd>)<br>Converts byte data in a general register to word data by extending the sign bit. |
| | EXTU | B | 0 → (<bits 15 to 8> of <Rd>)<br>Converts byte data in a general register to word data by padding with zero bits. |
| | TST | B/W | (EAd) – 0<br>Compares general register or memory contents with 0. |
| | NEG | B/W | 0 – (EAd) → (EAd)<br>Obtains the two's complement of general register or memory contents. |
| | CLR | B/W | 0 → (EAd)<br>Clears general register or memory contents to 0. |
| | TAS | B | (EAd) – 0, $(1)_2$ → (<bit 7> of <EAd>)<br>Tests general register or memory contents, then sets the most significant bit (bit 7) to 1. |

### 3.5.4 Logic Operations

Table 3-12 lists the four instructions that perform logic operations.

**Table 3-12 Logic Operation Instructions**

| Instruction | | Size | Function |
|---|---|---|---|
| Logical operations | AND | B/W | Rd ∧ (EAs) → Rd<br>Performs a logical AND operation on a general register and another general register, memory, or immediate data. |
| | OR | B/W | Rd ∨ (EAs) → Rd<br>Performs a logical OR operation on a general register and another general register, memory, or immediate data. |
| | XOR | B/W | Rd ⊕ (EAs) → Rd<br>Performs a logical exclusive OR operation on a general register and another general register, memory, or immediate data. |
| | NOT | B/W | ¬ (EAd) → (EAd)<br>Obtains the one's complement of general register or memory contents. |

### 3.5.5 Shift Operations

Table 3-13 lists the eight shift instructions.

**Table 3-13 Shift Instructions**

T-49-19-16

| Instruction | | Size | Function |
|---|---|---|---|
| Shift operations | SHAL | B/W | (EAd) shift → (EAd) |
| | SHAR | B/W | Performs an arithmetic shift operation on general register or memory contents. |
| | SHLL | B/W | (EAd) shift → (EAd) |
| | SHLR | B/W | Performs a logical shift operation on general register or memory contents. |
| | ROTL | B/W | (EAd) rotate → (EAd) |
| | ROTR | B/W | Rotates general register or memory contents. |
| | ROTXL | B/W | (EAd) rotate through carry → (EAd) |
| | ROTXR | B/W | Rotates general register or memory contents through the C (carry) bit. |

### 3.5.6 Bit Manipulations

Table 3-14 describes the four bit-manipulation instructions

T-49-19-16

**Table 3-14 Bit-Manipulation Instructions**

| Instruction | | Size | Function |
|---|---|---|---|
| Bit manipulations | `BSET` | B/W | ¬(<bit-No.> of <EAd> → Z, 1 → (<bit-No.> of <EAd>)<br>Tests a specified bit in a general register or memory, then sets the bit to 1. The bit is specified by a bit number given in immediate data or a general register. |
| | `BCLR` | B/W | ¬(<bit-No.> of <EAd>) → Z, 0 → (<bit-No.> of <EAd>)<br>Tests a specified bit in a general register or memory, then clears the bit to 0. The bit is specified by a bit number given in immediate data or a general register. |
| | `BNOT` | B/W | ¬(<bit-No.> of <EAd>) → Z, → (<bit-No.> of <EAd>)<br>Tests a specified bit in a general register or memory, then inverts the bit. The bit is specified by a bit number given in immediate data or a general register. |
| | `BTST` | B/W | ¬(<bit-No.> of <EAd>) → Z<br>Tests a specified bit in a general register or memory. The bit is specified by a bit number given in immediate data or a general register. |

### 3.5.7 Branching Instructions

Table 3-15 describes the 11 branching instructions

T-49-19-16

**Table 3-15 Branching Instructions**

<table>
<tr><th>Instruction</th><th></th><th>Size</th><th>Function</th></tr>
<tr><td>Branch</td><td><code>Bcc</code></td><td>—</td><td>Branches if condition cc is true.
<table>
<tr><th>Mnemonic</th><th>Description</th><th>Condition</th></tr>
<tr><td>BRA (BT)</td><td>Always (true)</td><td>True</td></tr>
<tr><td>BRN (BF)</td><td>Never (false)</td><td>False</td></tr>
<tr><td>BHI</td><td>High</td><td>$C \vee Z = 0$</td></tr>
<tr><td>BLS</td><td>Low or Same</td><td>$C \vee Z = 1$</td></tr>
<tr><td>BCC (BHS)</td><td>Carry Clear<br>(High or Same)</td><td>$C = 0$</td></tr>
<tr><td>BCS (BLO)</td><td>Carry Set (Low)</td><td>$C = 1$</td></tr>
<tr><td>BNE</td><td>Not Equal</td><td>$Z = 0$</td></tr>
<tr><td>BEQ</td><td>Equal</td><td>$Z = 1$</td></tr>
<tr><td>BVC</td><td>Overflow Clear</td><td>$V = 0$</td></tr>
<tr><td>BVS</td><td>Overflow Set</td><td>$V = 1$</td></tr>
<tr><td>BPL</td><td>Plus</td><td>$N = 0$</td></tr>
<tr><td>BMI</td><td>Minus</td><td>$N = 1$</td></tr>
<tr><td>BGE</td><td>Greater or Equal</td><td>$N \oplus V = 0$</td></tr>
<tr><td>BLT</td><td>Less Than</td><td>$N \oplus V = 1$</td></tr>
<tr><td>BGT</td><td>Greater Than</td><td>$Z \vee (N \oplus V) = 0$</td></tr>
<tr><td>BLE</td><td>Less or Equal</td><td>$Z \vee (N \oplus V) = 1$</td></tr>
</table></td></tr>
<tr><td></td><td>JMP</td><td>—</td><td>Branches unconditionally to a specified address in the same page.</td></tr>
<tr><td></td><td>PJMP</td><td>—</td><td>Branches unconditionally to a specified address in a specified page.</td></tr>
<tr><td></td><td>BSR</td><td>—</td><td>Branches to a subroutine at a specified address in the same page.</td></tr>
<tr><td></td><td>JSR</td><td>—</td><td>Branches to a subroutine at a specified address in the same page.</td></tr>
<tr><td></td><td>PJSR</td><td>—</td><td>Branches to a subroutine at a specified address in a specified page.</td></tr>
<tr><td></td><td>RTS</td><td>—</td><td>Returns from a subroutine in the same page.</td></tr>
</table>

**Table 3-15 Branching Instructions (cont)**

| Instruction | | Size | Function |
|---|---|---|---|
| Branch | PRTS | — | Returns from a subroutine in a different page. |
| | RTD | — | Returns from a subroutine in the same page and adjusts the stack pointer. |
| | PRTD | — | Returns from a subroutine in a different page and adjusts the stack pointer. |
| | SCB/F<br>SCB/NE<br>SCB/EQ | —<br>— | Controls a loop using a loop counter and/or a specified termination condition. |

### 3.5.8 System Control Instructions

Table 3-16 describes the 12 system control instructions.

T-49-19-16

Table 3-16 System Control Instructions

| Instruction | | Size | Function |
|---|---|---|---|
| System control | TRAPA | — | Generates a trap exception with a specified vector number. |
| | TRAP/VS | — | Generates a trap exception if the V bit is set to 1 when the instruction is executed. |
| | RTE | — | Returns from an exception-handling routine. |
| | LINK | — | FP → @–SP; SP → FP; SP + #IMM → SP<br>Creates a stack frame. |
| | UNLK | — | FP → SP; @SP + → FP<br>Deallocates a stack frame created by the LINK instruction. |
| | SLEEP | — | Causes a transition to the power-down state. |
| | LDC | B/W* | (EAs) → CR<br>Moves immediate data or general register or memory contents to a specified control register. |
| | STC | B/W* | CR → (EAd)<br>Moves control register data to a specified general register or memory location. |
| | ANDC | B/W* | CR ∧ #IMM → CR<br>Logically ANDs a control register with immediate data. |
| | ORC | B/W* | CR ∨ #IMM → CR<br>Logically ORs a control register with immediate data. |
| | XORC | B/W* | CR ⊕ #IMM → CR<br>Logically exclusive-ORs a control register with immediate data. |
| | NOP | — | PC + 1 → PC<br>No operation. Only increments the program counter. |

Note: * The size depends on the control register.

When using the LDC and STC instructions to stack and unstack the BR, CCR, TP, DP, and EP control registers in the H8/500 family, note the following point.

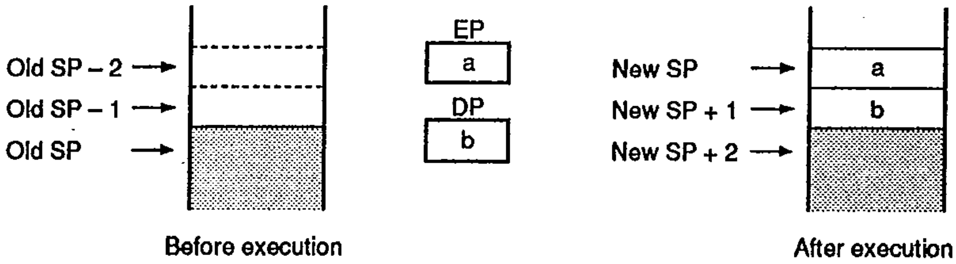
T-49-19-16

H8/500 hardware does not permit byte access to the stack. If the LDC.B or STC.B assembler mnemonic is coded with the @R7+ (@SP+) or @–R7 (@–SP) addressing mode, the stack-pointer addressing mode takes precedence and hardware automatically performs word access. Specifically, the LDC.B and STC.B instructions are executed as follows.

The following applies only to the stack-pointer addressing modes. In addressing modes that do not use the stack pointer, byte data access is performed as specified by the assembler mnemonic.

1. `STC.B EP, @-SP`

   When word data access is applied to EP, both EP and DP are accessed. This instruction stores EP at address SP (old) – 2, and DP at address SP (old) – 1.

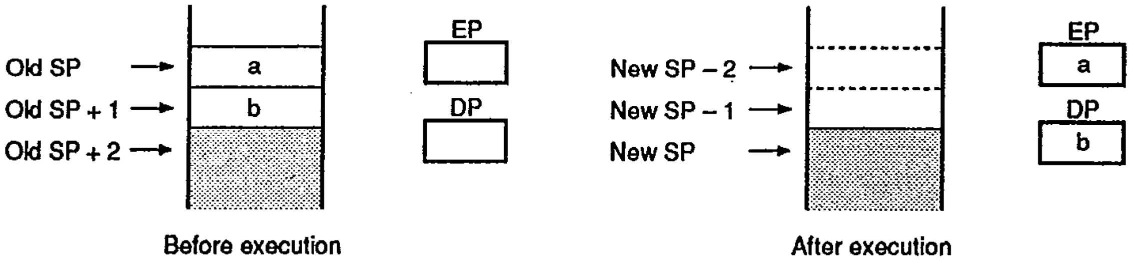


Before execution

After execution

2. `LDC.B @SP+, EP`

   When word data access is applied to EP, both EP and DP are accessed. This instruction loads EP from address SP (old), and DP from address SP (old) + 1, updating the DP value as well as the EP value.



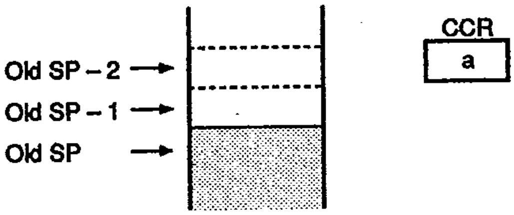
Before execution

After execution

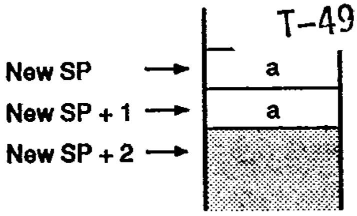
3. `STC.B CCR, @-SP`

When word data access is applied to CCR, only CCR is accessed. This instruction stores identical CCR contents at both address SP (old) – 2 and address SP (old) – 1.

T-49-19-16



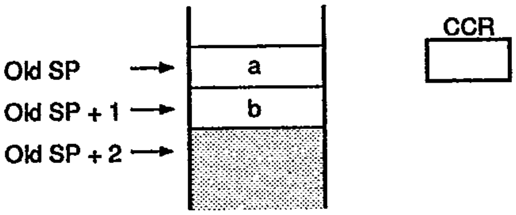
Before execution



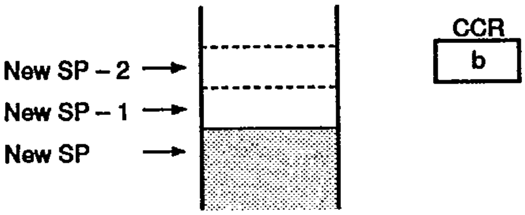
After execution

4. `LDC.B @SP+, CCR`

When word data access is applied to CCR, only CCR is accessed. This instruction loads CCR from address SP (old) + 1. Note that the value in address SP (old) is not loaded.



Before execution



After execution

BR, DP, and TP are accessed in the same way as CCR. When EP is specified, both EP and DP are accessed, but when CCR, BR, DP, or TP is specified, only the specified register is accessed.

### 3.5.9 Short-Format Instructions

T-49-19-16

The ADD, CMP, and MOV instructions have special short formats. Table 3-17 lists these short formats together with the equivalent general formats.

The short formats are a byte shorter than the corresponding general formats, and most of them execute one state faster.

**Table 3-17 Short-Format Instructions and Equivalent General Formats**

| Short-Format Instruction | Length | Execution States*2 | Equivalent General-Format Instruction | Length | Execution States*2 |
|---|---|---|---|---|---|
| `ADD:Q #xx,Rd`*1 | 2 | 2 | `ADD:G #xx:8,Rd` | 3 | 3 |
| `CMP:E #xx:8,Rd` | 2 | 2 | `CMP:G.B #xx:8,Rd` | 3 | 3 |
| `CMP:I #xx:16,Rd` | 3 | 3 | `CMP:G.W #xx:16,Rd` | 4 | 4 |
| `MOV:E #xx:8,Rd` | 2 | 2 | `MOV:G.B #xx:8,Rd` | 3 | 3 |
| `MOV:I #xx:16,Rd` | 3 | 3 | `MOV:G.W #xx:16,Rd` | 4 | 4 |
| `MOV:L @aa:8,Rd` | 2 | 5 | `MOV:G @aa:8,Rd` | 3 | 5 |
| `MOV:S Rs,@aa:8` | 2 | 5 | `MOV:G Rs,@aa:8` | 3 | 5 |
| `MOV:F @(d:8,R6),Rd` | 2 | 5 | `MOV:G @(d:8,R6),Rd` | 3 | 5 |
| `MOV:F Rs,@(d:8,R6)` | 2 | 5 | `MOV:G Rs,@(d:8,R6)` | 3 | 5 |

Notes: 1. The ADD: Q instruction accepts other destination operands in addition to a general register, but the immediate data value (#xx) is limited to ±1 or ±2.
2. Number of execution states for access to on-chip memory.

## 3.6 Operating Modes

The CPU operates in one of two modes: the minimum mode or the maximum mode. These modes are selected by the mode pins (MD2 to MD0).

### 3.6.1 Minimum Mode

The minimum mode supports a maximum address space of 64 kbytes. The page registers are ignored. Instructions that branch across page boundaries (PJMP, PJSR, PRTS, PRTD) are invalid.

### 3.6.2 Maximum Mode

In the maximum mode the page registers are valid, expanding the maximum address space to 1 Mbyte.

The address space is divided into 64-kbyte pages. The pages are separate; it is not possible to move continuously across a page boundary.

It is possible to move from one page to another with branching instructions (PJMP, PJSR, PRTS, PRTD). The TRAPA instruction and instructions that branch to interrupt-handling routines can also jump across page boundaries. It is not necessary for a program to be contained in a single 64-kbyte page.

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

For further information on the operating modes, see section 2, "MCU Operating Modes and Address Space."

## 3.7 Basic Operational Timing

### 3.7.1 Overview

The CPU operates on a system clock (ø) which is created by dividing the crystal oscillator frequency (fosc) by two. One period of the system clock is referred to as a "state." The CPU accesses memory in a cycle consisting of 2 or 3 states. The CPU uses different methods to access on-chip memory, the on-chip register field, and external devices.

**Access to On-Chip Memory (RAM, ROM):** For maximum speed, access to on-chip memory (RAM, ROM) is performed in two states, using a 16-bit-wide data bus.

Figure 3-6 shows the on-chip memory access cycle. Figure 3-7 indicates the pin states. The bus control signals output from the H8/520 chip go to the nonactive state during the access.

**Access to On-Chip Register Field (Addresses H'FF80 to H'FFFF):** The access cycle consists of three states. The data bus is 8 bits wide.

Figure 3-8 shows the on-chip supporting module access cycle. Figure 3-9 indicates the pin states.

**Access to External Devices:** The access cycle consists of three states. The data bus is 8 bits wide. Figure 3-10 (a) and (b) shows the external access cycle. Additional wait states (Tw) can be inserted by the wait-state controller (WSC).

T-49-19-16

### 3.7.2 On-Chip Memory Access Cycle

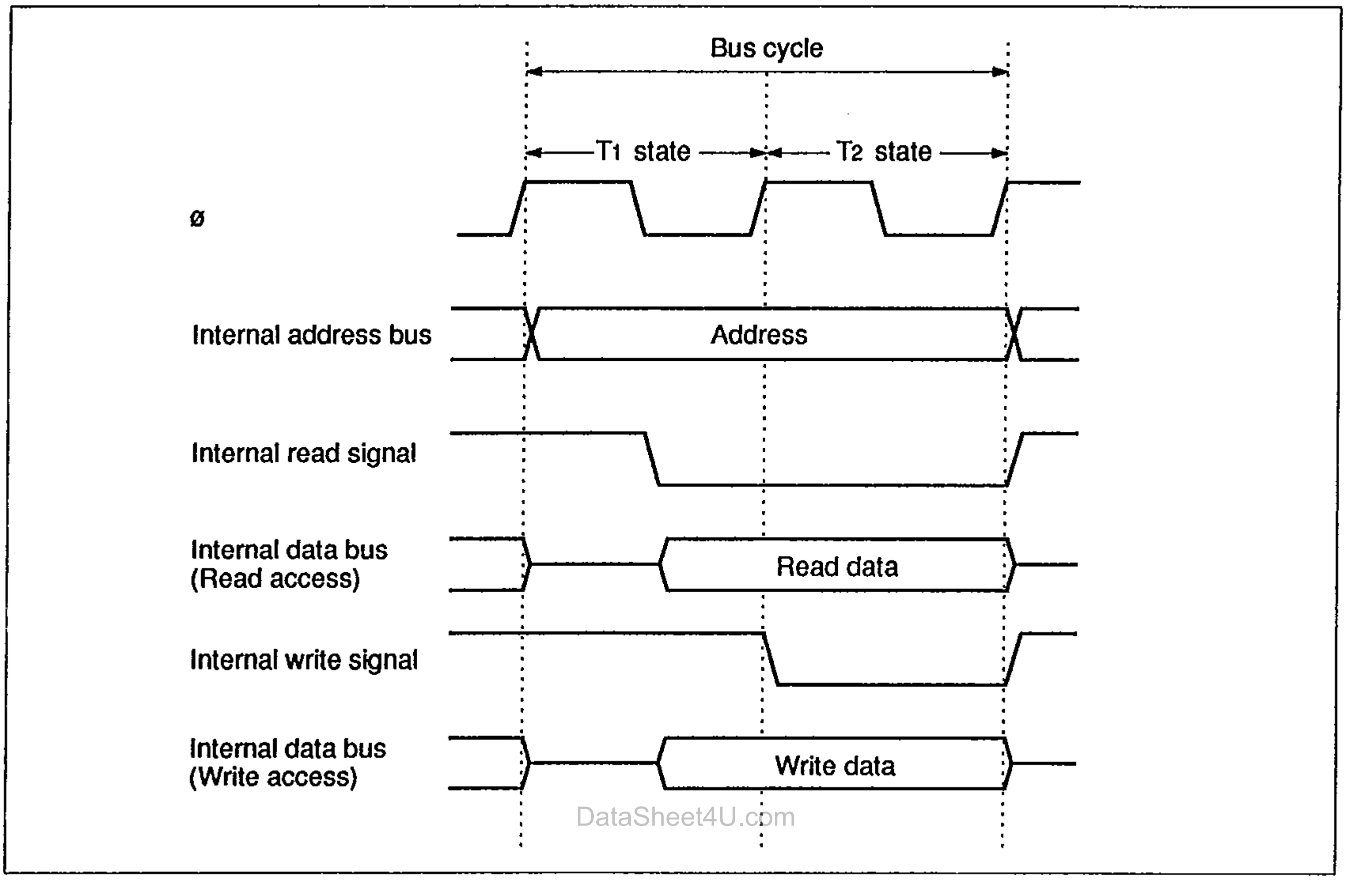


Figure 3-6 On-Chip Memory Access Timing

### 3.7.3 Pin States during On-Chip Memory Access

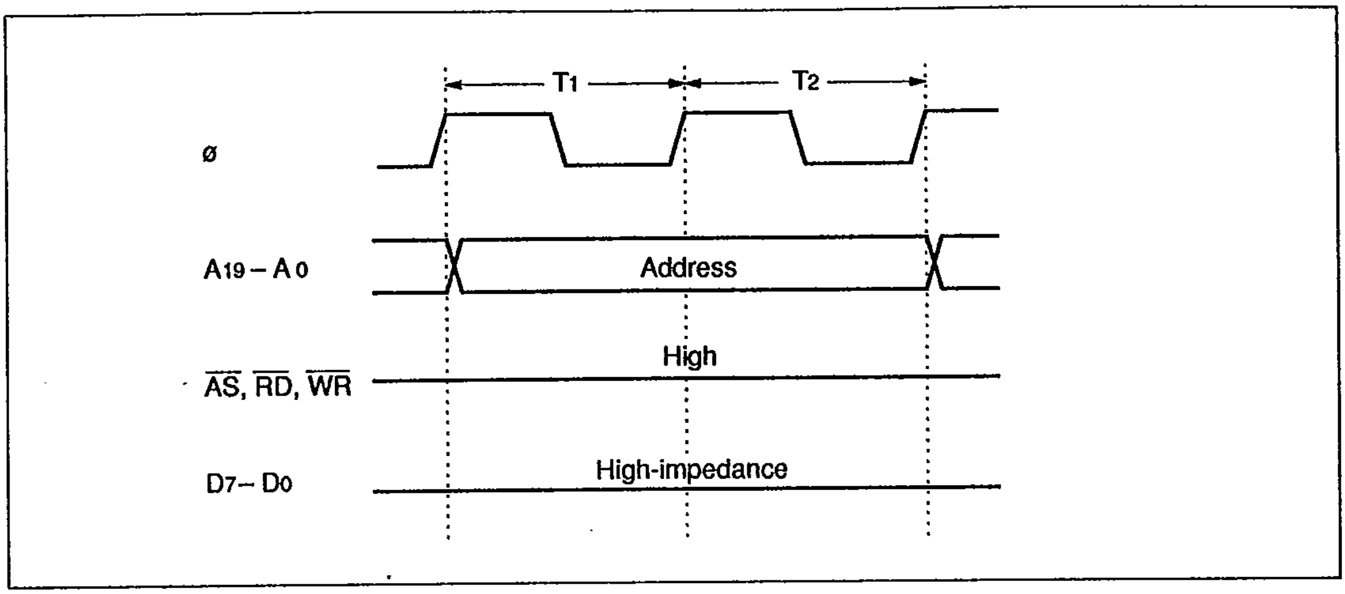


**Figure 3-7 Pin States during Access to On-Chip Memory**

### 3.7.4 Register Field Access Cycle (Addresses H'FF80 to H'FFFF)

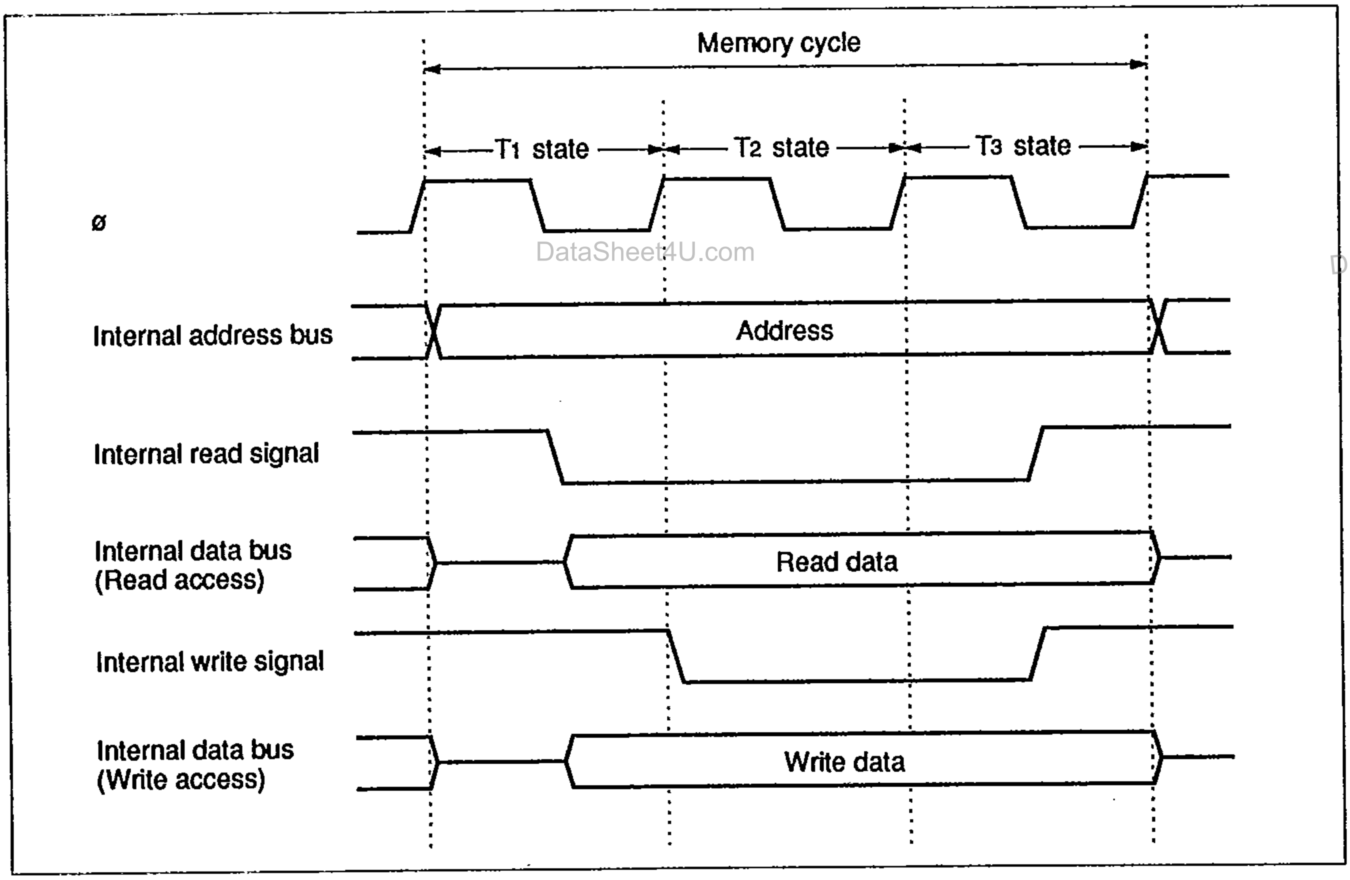


**Figure 3-8 Register Field Access Timing**

### 3.7.5 Pin States during Register Field Access (Addresses H'FF80 to H'FFFF)

T-49-19-16

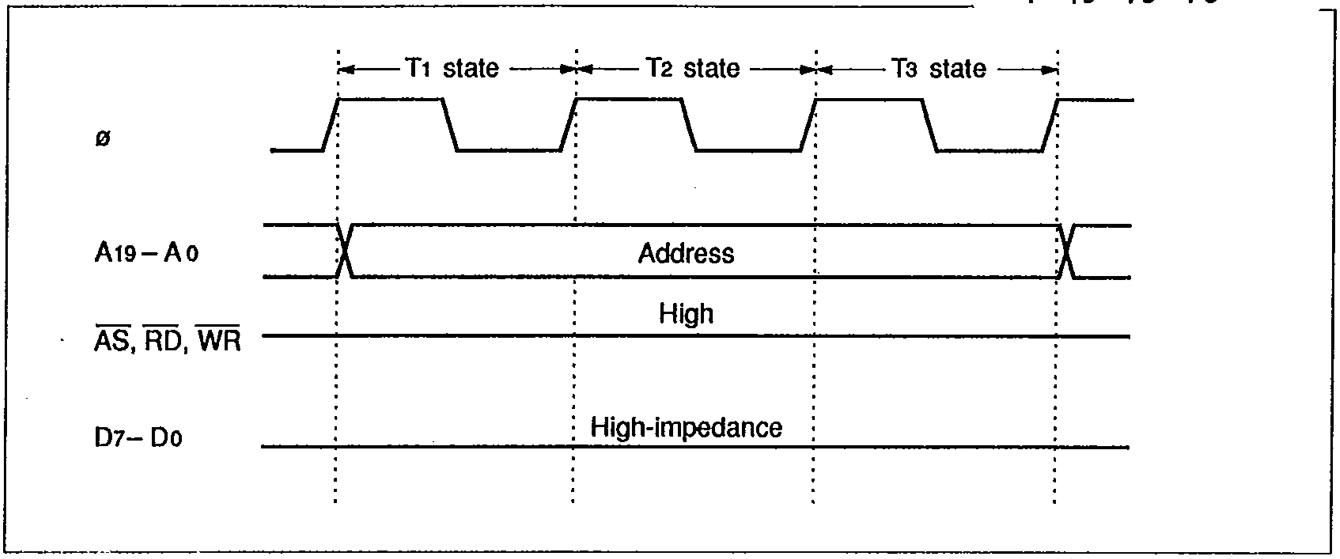


Figure 3-9 Pin States during Register Field Access

### 3.7.6 External Access Cycle

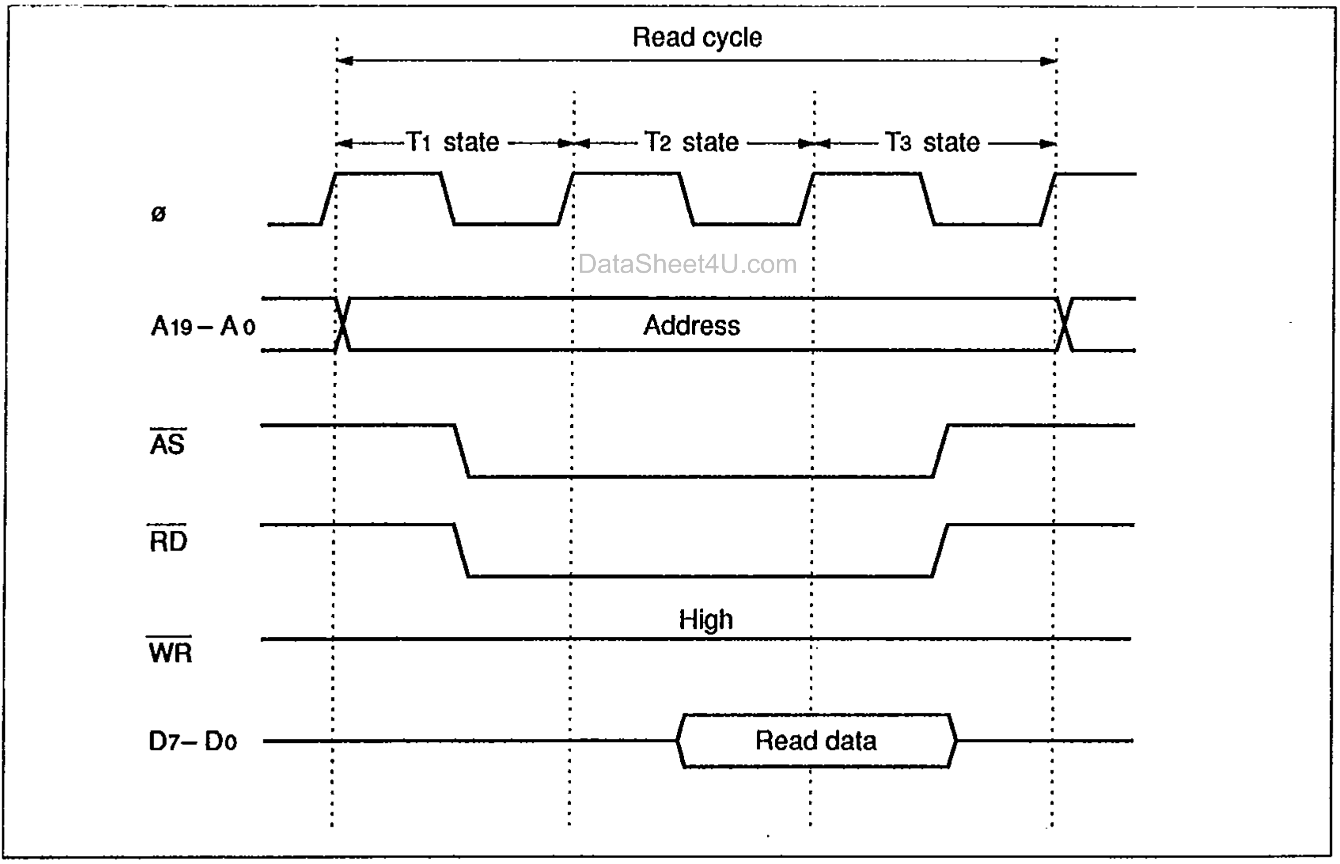


Figure 3-10 (a) External Access Cycle (Read Access)

T-49-19-16

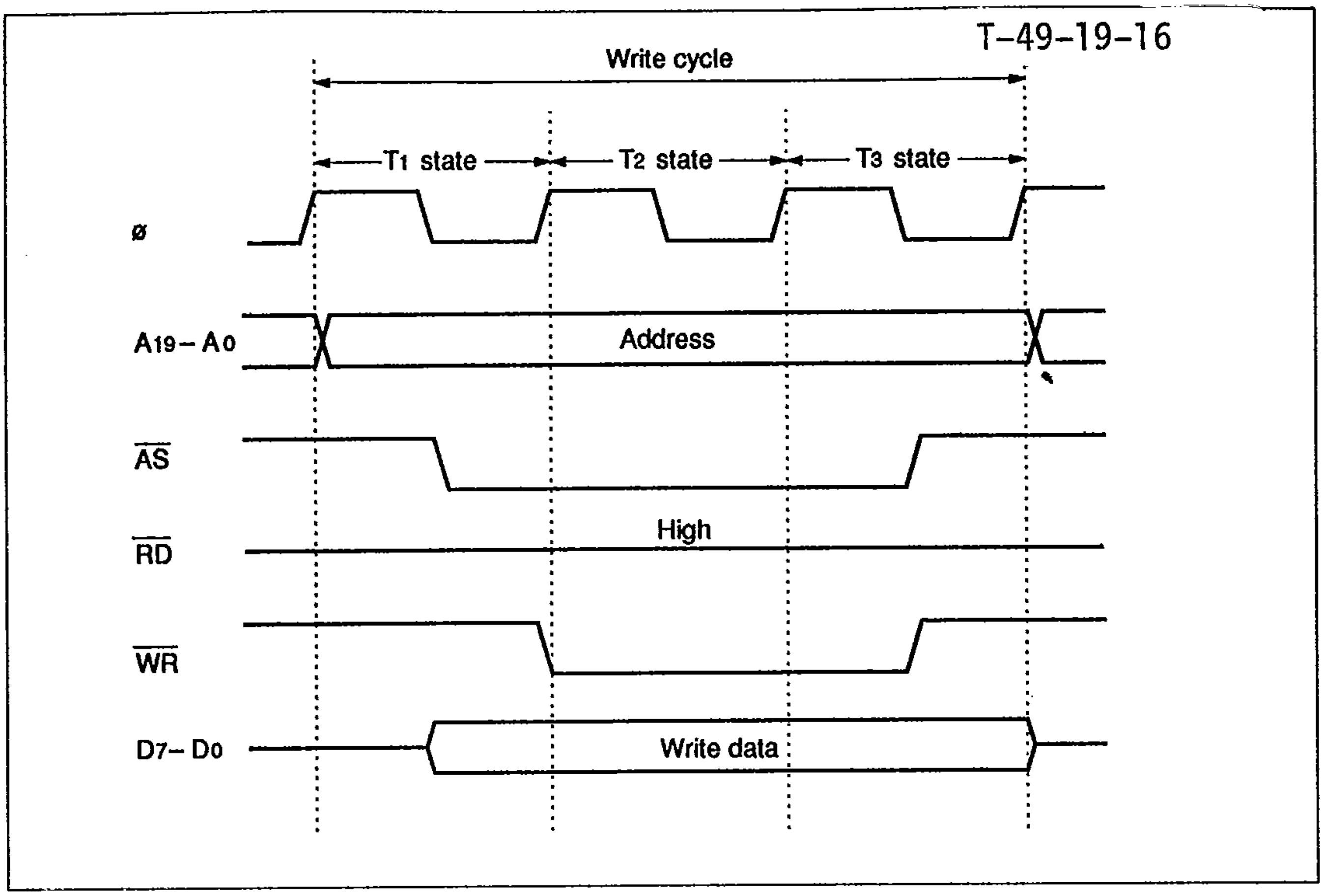


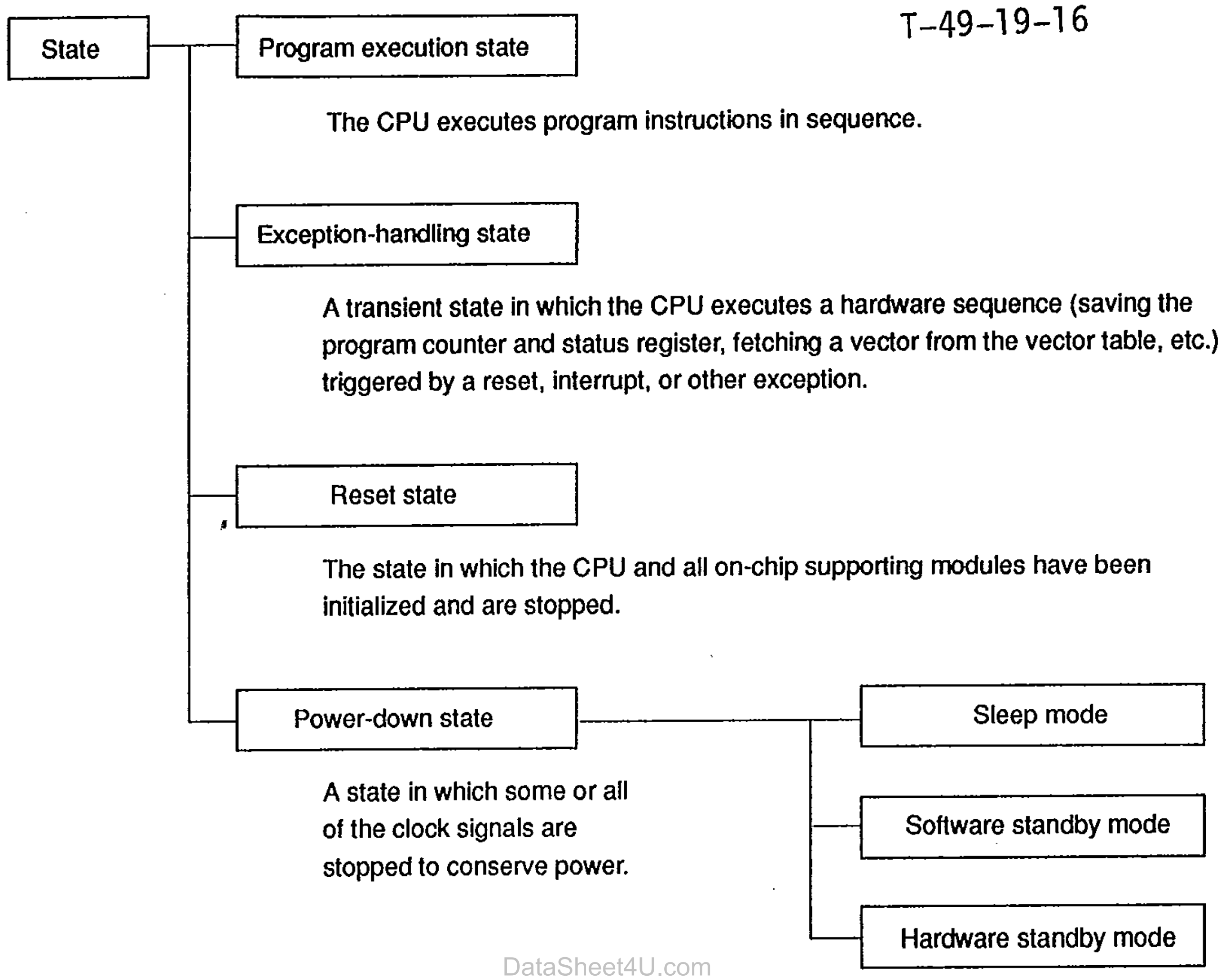
Figure 3-10 (b) External Access Cycle (Write Access)

## 3.8 CPU States

### 3.8.1 Overview

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

T-49-19-16



The CPU executes program instructions in sequence.

A transient state in which the CPU executes a hardware sequence (saving the program counter and status register, fetching a vector from the vector table, etc.) triggered by a reset, interrupt, or other exception.

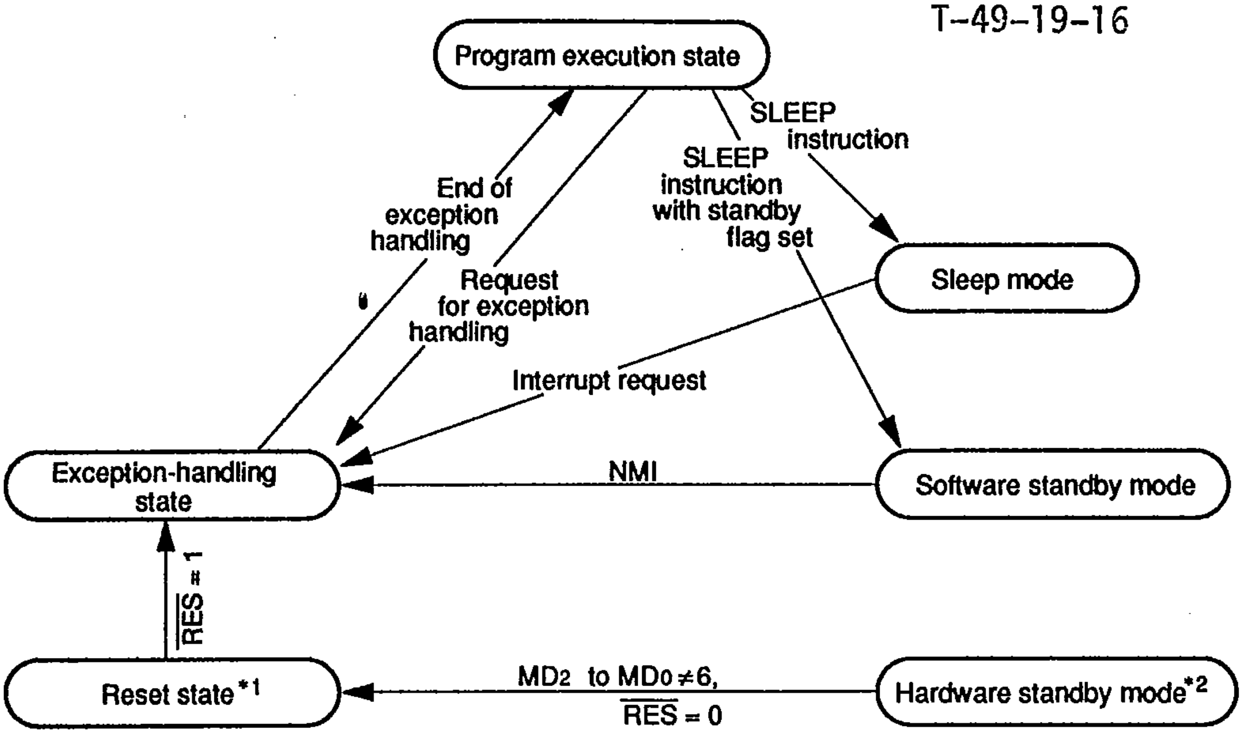
The state in which the CPU and all on-chip supporting modules have been initialized and are stopped.

A state in which some or all of the clock signals are stopped to conserve power.

Note: H8/520 does not support the bus-release function. There is no bus-released state.

Figure 3-11 Operating States

T-49-19-16



Notes: 1. From any state except the hardware standby mode, a transition to the reset state occurs whenever $\overline{\text{RES}}$ goes low or the watchdog timer requests a reset.

2. A transition to the hardware standby mode from any state occurs when the mode pins (MD2 to MD0) are set to 6.

Figure 3-12 State Transitions

### 3.8.2 Program Execution State

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

### 3.8.3 Exception-Handling State

The exception-handling state is a transient state that occurs when the CPU alters the normal program flow due to an interrupt, trap instruction, address error, or other exception. 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 status register (in minimum mode) or program counter, code page register, and status register (in maximum mode) to the stack.
2. Clears the T bit in the status register to 0.
3. Fetches the start address of the exception-handling routine from the exception 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.

### 3.8.4 Reset State

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

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

### 3.8.5 Power-Down State

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

See section 17, "Power-Down State", for further information.

# Section 4 Exception Handling

T-49-19-16

## 4.1 Overview

### 4.1.1 Types of Exception Handling and Their Priority

As indicated in table 4-1 (a) and (b), exception handling can be initiated by a reset, address error, trace, interrupt, or instruction. An instruction initiates exception handling if the instruction is an invalid instruction, a trap instruction, or a DIVXU instruction with zero divisor. Exception handling begins with a hardware exception-handling sequence which prepares for the execution of a user-coded software exception-handling routine.

There is a priority order among the different types of exceptions, as shown in table 4-1 (a). If two or more exceptions occur simultaneously, they are handled in their order of priority. An instruction exception cannot occur simultaneously with other types of exceptions.

**Table 4-1 (a) Exceptions and Their Priority**

| Priority | Exception Type | Source | Detection Timing | Start of Exception-Handling Sequence |
|---|---|---|---|---|
| High ↑ | Reset | External, internal | $\overline{\text{RES}}$ Low-to-High transition | Immediately |
| | Address error | Internal | Instruction fetch or data read/write bus cycle | End of instruction execution |
| | Trace | Internal | End of instruction execution, if T = 1 in status register | End of instruction execution |
| ↓ Low | Interrupt | External, internal | End of instruction execution or end of exception-handling sequence | End of instruction execution |

**Table 4-1 (b) Instruction Exceptions**

| Exception Type | Start of Exception-Handling Sequence |
|---|---|
| Invalid instruction | Attempted execution of instruction with undefined code |
| Trap instruction | Started by execution of trap instruction |
| Zero divide | Attempted execution of DIVXU instruction with zero divisor |

T-49-19-16

### 4.1.2 Hardware Exception-Handling Sequence

The hardware exception-handling sequence varies depending on the type of exception. When exception handling is initiated by an exception other than a reset, the CPU:

1. Saves the program counter and status register (in minimum mode) or program counter, code page register, and status register (in maximum mode) to the stack.
2. Clears the T bit in the status register to 0.
3. Fetches the start address of the exception-handling routine from the exception vector table.
4. Branches to that address.

For an interrupt, the CPU also alters the interrupt mask level in bits $I_2$ to $I_0$ of the status register.

For a reset, step 1 is omitted. See section 4.2, “Reset”, for the full reset sequence.

### 4.1.3 Exception Sources and Vector Table

The sources that initiate exception handling can be classified as shown in figure 4-1.

The starting addresses of the exception-handling routines for each source are contained in an exception vector table located in the low addresses of page 0. The vector addresses are listed in table 4-2. Note that there are different addresses for the minimum and maximum modes.

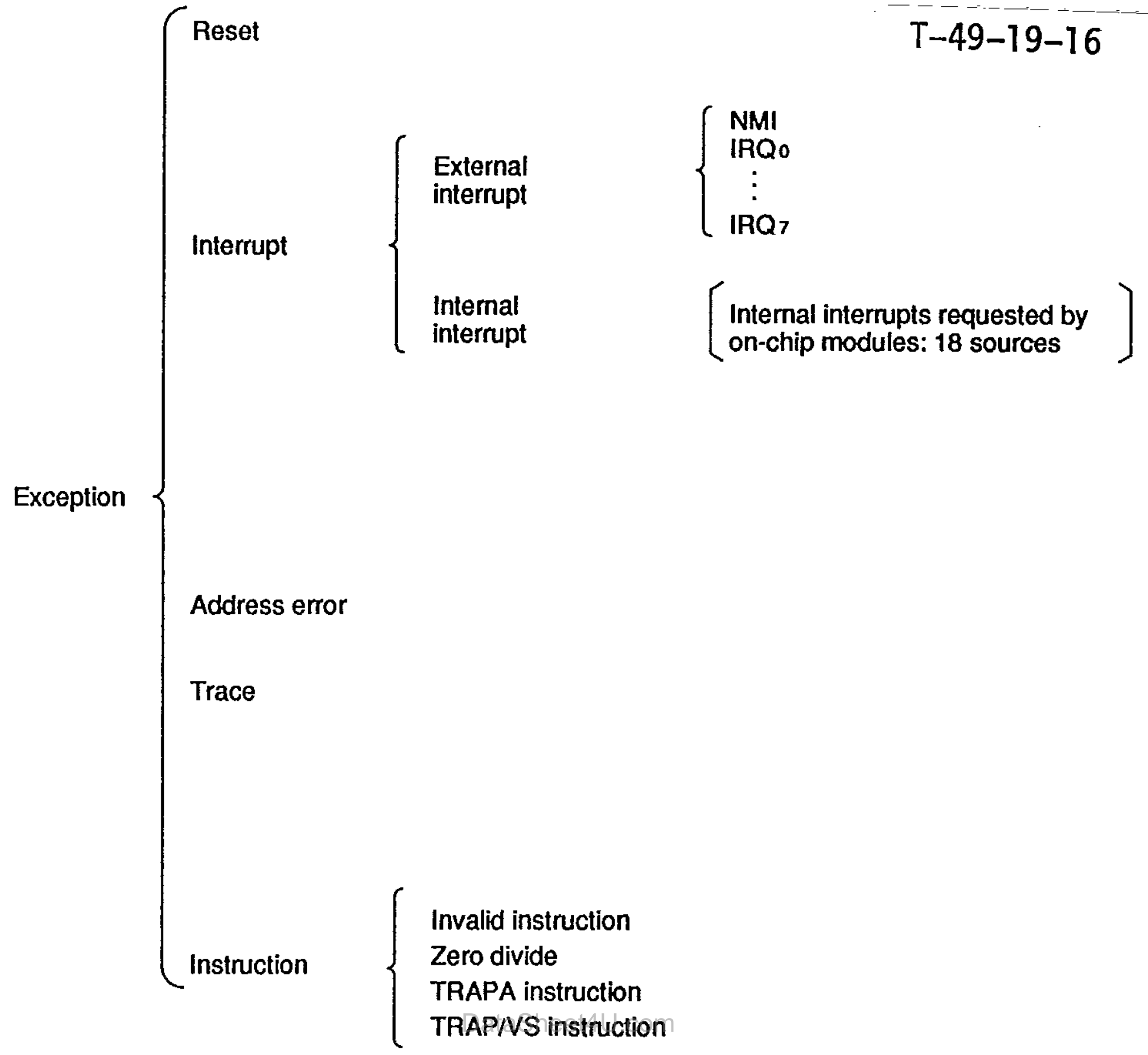


Figure 4-1 Sources Causing Exception Handling

## Table 4-2 Exception Vector Table

T-49-19-16

| Type of Exception | Vector Address<br>Minimum Mode | <br>Maximum Mode |
|---|---|---|
| Reset (initialize PC) | H'0000 to H'0001 | H'0000 to H'0003 |
| — (Reserved for system) | H'0002 to H'0003 | H'0004 to H'0007 |
| Invalid instruction | H'0004 to H'0005 | H'0008 to H'000B |
| DIVXU instruction (zero divide) | H'0006 to H'0007 | H'000C to H'000F |
| TRAP/VS instruction | H'0008 to H'0009 | H'0010 to H'0013 |
| — (Reserved for system) | H'000A to H'000B<br>to<br>H'000E to H'000F | H'0014 to H'0017<br>to<br>H'001C to H'001F |
| Address error | H'0010 to H'0011 | H'0020 to H'0023 |
| Trace | H'0012 to H'0013 | H'0024 to H'0027 |
| — (Reserved for system) | H'0014 to H'0015 | H'0028 to H'002B |
| Nonmaskable external interrupt (NMI) | H'0016 to H'0017 | H'002C to H'002F |
| — (Reserved for system) | H'0018 to H'0019<br>to<br>H'001E to H'001F | H'0030 to H'0033<br>to<br>H'003C to H'003F |
| TRAPA instruction (16 vectors) | H'0020 to H'0021<br>to<br>H'003E to H'003F | H'0040 to H'0043<br>to<br>H'007C to H'007F |
| External interrupts $IRQ_0$ to $IRQ_7$ | H'0040 to H'0041<br>to<br>H'004E to H'004F | H'0080 to H'0083<br>to<br>H'009C to H'009F |
| Internal interrupts | H'0050 to H'0051<br>to<br>H'007E to H'007F | H'00AO to H'00A3<br>to<br>H'00FC to H'00FF |

Notes: 1. The exception vector table is located at the beginning of page 0.
2. For details of the internal interrupt vectors, see table 5-2.

## 4.2 Reset

T-49-19-16

### 4.2.1 Overview

A reset has the highest exception-handling priority.

A reset can be generated by a low input at the $\overline{\text{RES}}$ pin or by a watchdog timer (WDT) overflow.

When the $\overline{\text{RES}}$ pin goes low, all current processing halts and the H8/520 chip enters the reset state. The internal status of the CPU and the contents of the registers of the on-chip supporting modules are initialized. When the $\overline{\text{RES}}$ pin returns from low to high, the hardware reset sequence described in the next section begins. To ensure that the H8/520 chip is reset correctly, the $\overline{\text{RES}}$ pin should be held low for at least 20 ms at power-up. To reset the H8/520 during operation, the $\overline{\text{RES}}$ pin should be held low for at least six system clock (ø) cycles.

When the RSTOE bit (see below) is set to 1, the $\overline{\text{RES}}$ input must be held low for at least 520 system clock (ø) cycles to reset the H8/520 chip.

When the watchdog timer operates in watchdog mode, if the watchdog timer counter (TCNT) overflows due to a program crash, for example, the watchdog timer generates an internal reset signal that resets the H8/520 chip. If in addition the reset output enable (RSTOE) bit in the reset control/status register (RSTCSR) is set to 1, a low output signal is generated at the $\overline{\text{RES}}$ pin for 132 system clock (ø) cycles. This signal can be used to reset devices controlled by the H8/520.

See section 12, "Watchdog Timer", for further information on the reset generated by the watchdog timer.

See appendix E, "Pin Status in the Reset State", for the status of pins when a reset occurs.

### 4.2.2 Reset Sequence

When the $\overline{\text{RES}}$ pin returns to the high state after being held low for the necessary time, the hardware reset exception-handling sequence begins, during which:

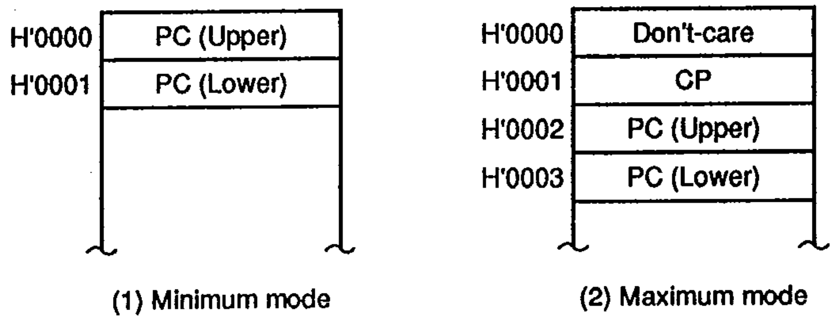
1. The value at the mode pins ($\text{MD}_2$ to $\text{MD}_0$) is latched in bits MDS2 to MDS0 of the mode control register (MDCR).
2. In the status register (SR), the T bit is cleared to disable the trace mode, and the interrupt mask level (bits $\text{I}_2$ to $\text{I}_0$) is set to 7. A reset disables all interrupts.
3. The CPU loads the reset start address from the vector table into the program counter and begins executing the program at that address.

The contents of the vector table differs between minimum mode and maximum mode as indicated in figure 4-2. This affects step 3 as described below.

T-49-19-16

**Minimum Mode:** One word is copied from addresses H'0000 and H'0001 in the vector table to the program counter. Program execution then begins from the address in the program counter (PC).

**Maximum Mode:** Two words are read from addresses H'0000 to H'0003 in the vector table. The byte in address H'0000 is ignored. The byte in address H'0001 is copied to the code page register (CP). The contents of addresses H'0002 and H'0003 are copied to the program counter. Program execution starts from the address indicated by the code page register and program counter.



(1) Minimum mode

(2) Maximum mode

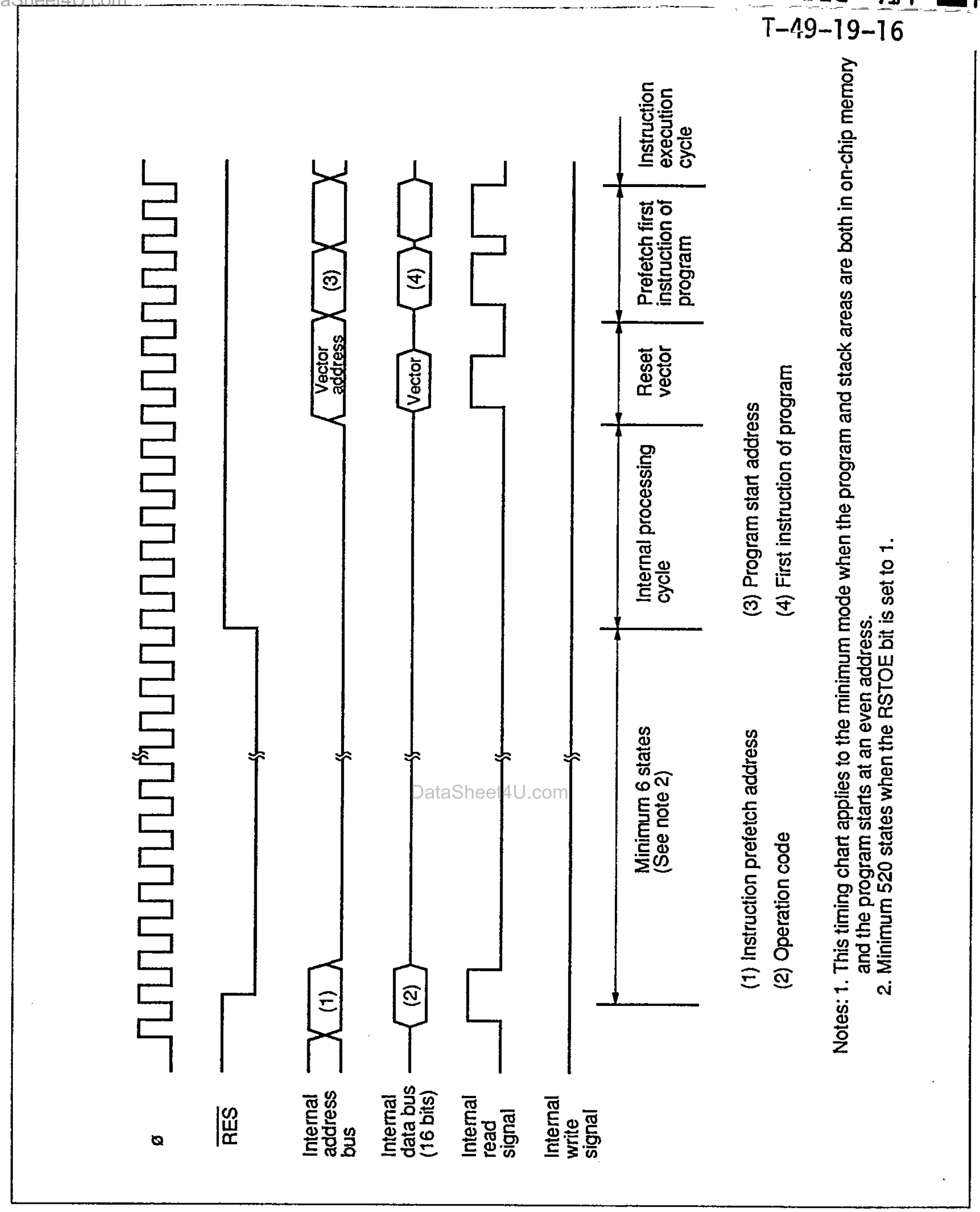
Figure 4-2 Reset Vector

Figure 4-3 shows the timing of the reset sequence in minimum mode. Figure 4-4 shows the timing of the reset sequence in maximum mode.

### 4.2.3 Stack Pointer Initialization

The hardware reset sequence does not initialize the stack pointer, so this must be done by software. If an interrupt were to be accepted after a reset and before the stack pointer (SP) is initialized, the program counter and status register would not be saved correctly, causing a program crash. This danger can be avoided by coding the reset routine as explained next.

When the chip comes out of the reset state all interrupts, including NMI, are disabled, so the instruction at the reset start address is always executed. In the minimum mode, this instruction should initialize the stack pointer (SP). In the maximum mode, this instruction should be an LDC instruction initializing the stack page register (TP), and the next instruction should initialize the stack pointer. Execution of the LDC instruction disables interrupts again, ensuring that the stack pointer initializing instruction is executed.

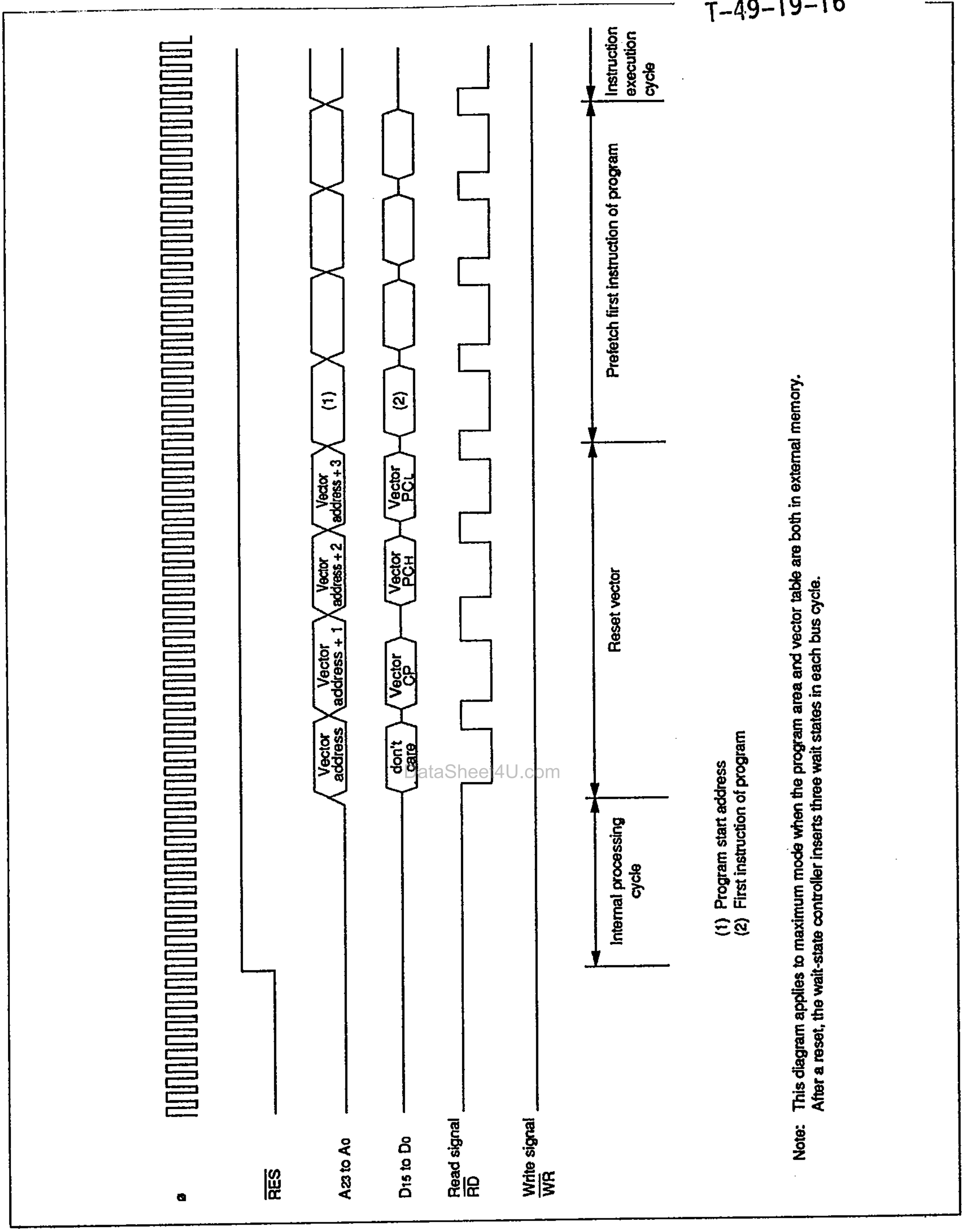


(1) Instruction prefetch address
(2) Operation code
(3) Program start address
(4) First instruction of program

Notes: 1. This timing chart applies to the minimum mode when the program and stack areas are both in on-chip memory and the program starts at an even address.
2. Minimum 520 states when the RSTOE bit is set to 1.

Figure 4-3 Reset Sequence (Minimum Mode, On-Chip Memory)

T-49-19-16



(1) Program start address
(2) First instruction of program

Note: This diagram applies to maximum mode when the program area and vector table are both in external memory. After a reset, the wait-state controller inserts three wait states in each bus cycle.

Figure 4-4 Reset Sequence (Maximum Mode, External Memory)

## 4.3 Address Error

T-49-19-16

There are three causes of address errors:

- Instruction prefetch from illegal address
- Word data access at odd address
- Off-chip access in single-chip mode

An address error initiates the address error exception-handling sequence. This sequence clears the T bit of the status register to 0 to disable the trace mode, but does not affect the interrupt mask level in bits $I_2$ to $I_0$.

### 4.3.1 Instruction Prefetch from Illegal Address

An attempt to prefetch an instruction from the register field in memory addresses H'FF80 to H'FFFF causes an address error regardless of the MCU operating mode.

Handling of this address error begins when the prefetch cycle that caused the error has been completed and execution of the current instruction has also been completed. The program counter value pushed on the stack is the address of the instruction immediately following the last instruction executed. See section 4.9, "Stack Status after Completion of Exception Handling", for a diagram of the stack.

Program code should not be located in addresses H'FF7D to H'FF7F. If the CPU executes an instruction in these addresses, it will attempt to prefetch the next instruction from the register field, causing an address error.

### 4.3.2 Word Data Access at Odd Address

If an attempt is made to access word data starting at an odd address, an address error occurs regardless of the MCU operating mode. The program counter value pushed on the stack in the handling of this error is the address of the next instruction after the instruction that attempted the illegal word access.

### 4.3.3 Off-Chip Address Access in Single-Chip Mode

In the single-chip mode there is no external memory, so in addition to the address errors described above, the following two types of address errors can occur.

**Access to Addresses H'4000 to H'FD7F:** These addresses exist neither in on-chip ROM or RAM nor in the on-chip register field, so an address error occurs if they are accessed for any purpose: for instruction prefetch, byte data access, or word data access.

T-49-19-16

Program code should not be located in the last three bytes of on-chip ROM (addresses H'3FFD to H'3FFF) in single-chip mode. If an instruction is located in these three bytes, the CPU will attempt to fetch the next instruction from addresses H'4000 to H'4002, causing an address error.

**Access to Disabled RAM Area:** The on-chip RAM area (H'FD80 to H'FF7F) can be disabled by clearing the RAME bit in the RAM control register (RAMCR). If any type of RAM access is attempted in this state in the single-chip mode, an address error occurs.

## 4.4 Trace

When the T bit of the status register is set to 1, the CPU operates in trace mode. A trace exception occurs at the completion of each instruction. The trace mode can be used to monitor program execution for debugging by a debugger.

In the trace exception sequence the T bit of the status register is cleared to 0 to disable the trace mode while the trace routine is executing. The interrupt mask level in bits $I_2$ to $I_0$ is not changed. Interrupts are accepted as usual during the trace routine.

In the status-register data saved on the stack, the T bit is set to 1. When the trace routine returns with the RTE instruction, the status register is popped from the stack and the trace mode resumes.

If an address error occurs during execution of the first instruction after the return from the trace routine, since the address error has higher priority, the address error exception-handling sequence is initiated, clearing the T bit in the status register to 0 and making it impossible to trace this instruction.

## 4.5 Interrupts

Interrupts can be requested from nine external sources (NMI and $IRQ_0$ to $IRQ_7$) and seven on-chip supporting modules: the 16-bit free-running timers (FRT1 and FRT2), the 8-bit timer, the serial communication interfaces (SCI1 and SCI2), the A/D converter, and the watchdog timer (WDT). The on-chip interrupt sources can request a total of eighteen different types of interrupts, each having its own interrupt vector. Figure 4-5 lists the interrupt sources and the number of different interrupts from each source.

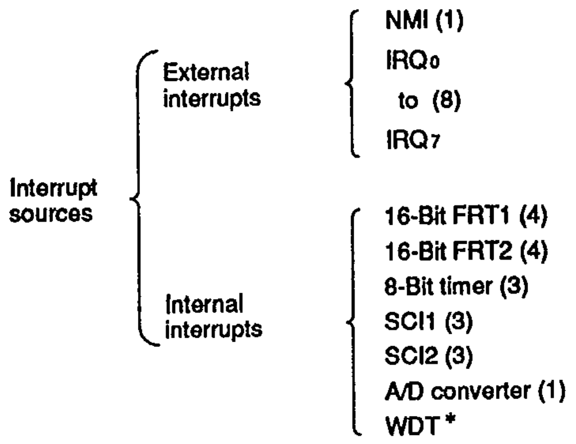
Each interrupt source has a priority. NMI interrupts have the highest priority, and are normally accepted unconditionally. The priorities of the other interrupt sources are set in interrupt priority registers A to D (IPRA to IPRD) in the register field at the high end of page 0 and can be changed by software. Priority levels range from 0 (low) to 7 (high), with NMI considered to be on level 8. $\mathrm{IRQ}_1$ to $\mathrm{IRQ}_7$ always have the same priority. The priority of $\mathrm{IRQ}_0$ can be set independently.

The on-chip interrupt controller decides whether an interrupt can be accepted by comparing its priority with the interrupt mask level, and determines the order in which to accept competing interrupt requests. Interrupts that are not accepted immediately remain pending until they can be accepted later.

When it accepts an interrupt, the interrupt controller also decides whether to have the interrupt handled by the CPU or the on-chip data transfer controller (DTC). This decision is controlled by bits set in data transfer enable registers A to D (DTEA to DTED) in the register field. The DTC is started if the corresponding DTE bit is set to 1; otherwise a CPU interrupt is generated. DTC interrupts provide an efficient way to send and receive blocks of data via the serial communication interface, or to transfer data between memory and I/O without detailed CPU programming. The CPU halts while the DTC is executing. DTC interrupts are described in section 6, "Data Transfer Controller".

The hardware exception-handling sequence for a CPU interrupt clears the T bit in the status register to 0 and sets the interrupt mask level in bits $I_2$ to $I_0$ to the level of the interrupt it has accepted. This prevents the interrupt-handling routine from being interrupted except by a higher-level interrupt. The previous interrupt mask level is restored on the return from the interrupt-handling routine.

For further information on interrupts, see section 5, "Interrupt Controller".



T-49-19-16

| | |
|---|---|
| NMI: | NonMaskable Interrupt |
| IRQ: | Interrupt Request |
| FRT: | Free-Running Timer |
| SCI: | Serial Communication Interface |
| WDT: | WatchDog Timer |

Note: * Interrupts from the watchdog timer in the interval timer mode are handled as $IRQ_0$.

Figure 4-5 Interrupt Sources (and Number of Interrupt Types)

## 4.6 Invalid Instruction

An invalid instruction exception occurs if an attempt is made to execute an instruction with an undefined operation code or illegal addressing mode specification. The program counter value pushed on the stack is the value of the program counter when the invalid instruction code was detected.

In the invalid instruction exception-handling sequence the T bit of the status register is cleared to 0, but the interrupt mask level ($I_2$ to $I_0$) is not affected. If a normal interrupt is requested while a trap or zero-divide instruction is being executed, after the trap or zero-divide exception-handling sequence, the normal interrupt exception-handling sequence is carried out.

## 4.7 Trap Instructions and Zero Divide

A trap exception occurs when the TRAPA or TRAP/VS instruction is executed. A zero divide exception occurs if an attempt is made to execute a DIVXU instruction with a zero divisor.

In the exception-handling sequences for these exceptions the T bit of the status register is cleared to 0, but the interrupt mask level ($I_2$ to $I_0$) is not affected.

**TRAPA Instruction:** The TRAPA instruction always causes a trap exception. The TRAPA instruction includes a vector number from 0 to 15, allowing the user to provide up to sixteen different trap-handling routines.

**TRAP/VS Instruction:** When the TRAP/VS instruction is executed, a trap exception occurs if the overflow (V) bit in the condition code register is set to 1. If the V bit is cleared to 0, no exception occurs and the next instruction is executed.

**DIVXU Instruction with Zero Divisor:** An exception occurs if an attempt is made to divide by zero in a DIVXU instruction.

## 4.8 Cases in Which Exception Handling is Deferred

In the case described next, the address error exception, trace exception, external interrupt (NMI and $IRQ_0$ to $IRQ_7$) requests, and internal interrupt requests (18 types) are not accepted immediately but are deferred until after the next instruction has been executed.

### 4.8.1 Instructions that Disable Interrupts

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

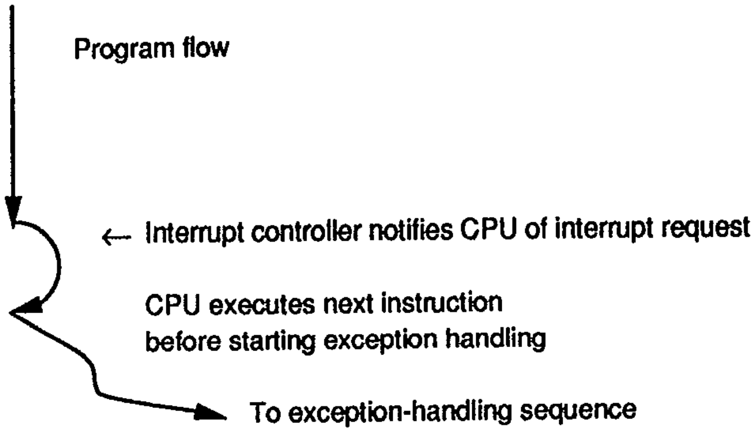
Suppose that an internal interrupt is requested and the interrupt controller, after checking the interrupt priority and interrupt mask level, notifies the CPU of the interrupt, but the CPU is currently executing one of the five instructions listed above. After executing this instruction the CPU always proceeds to the next instruction. (And if the next instruction is one of these five, the CPU also proceeds to the next instruction after that.) The exception-handling sequence starts after the next instruction that is not one of these five has been executed. The following is an example:

```
          .
          .
          .
          .
          .

LDC.B  #H'00,TP

MOV.W  #H'FF80,SP

MOV.B  #H'00,@WCR
          .
          .
          .
```



Note: When the LDC instruction alters the I bits in the status register (SR), the new I-bit values do not take effect until three states after the LDC instruction. If a program running in on-chip memory uses the LDC instruction to enable interrupts by modifying the I bits and the next instruction is a two-state instruction (NOP for example), interrupts will not be accepted after this next instruction; they will not be accepted until another instruction has been executed after that. The same applies to the ANDC, ORC, and XORC instructions.

### 4.8.2 Disabling of Exceptions Immediately after a Reset

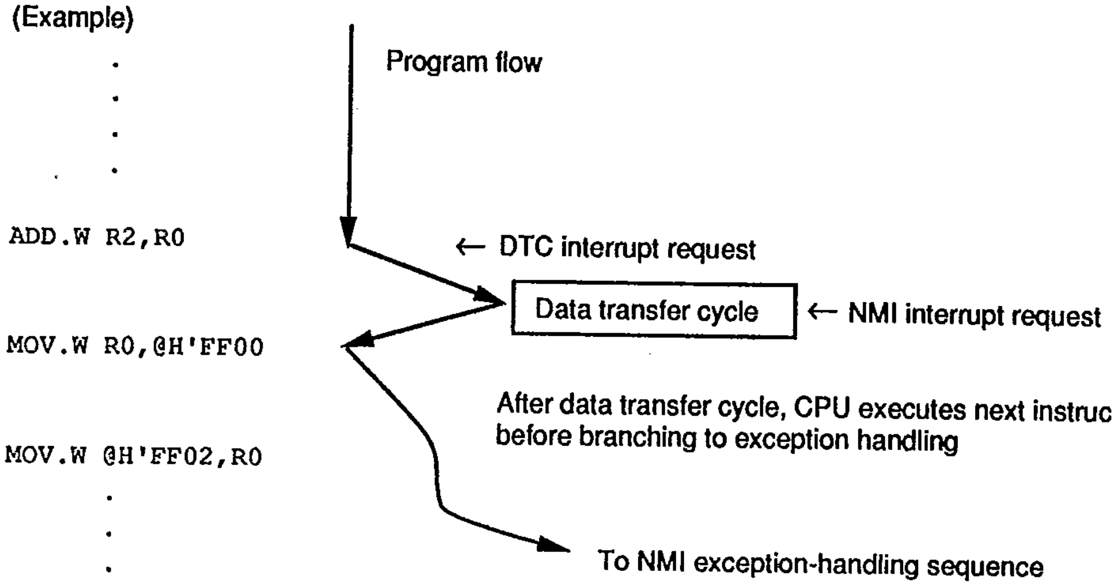
If an interrupt is accepted after a reset and before the stack pointer (SP) is initialized, the program counter and status register will not be saved correctly, leading to a program crash. To prevent this, when the chip comes out of the reset state all interrupts, including NMI, are disabled, so the first instruction of the reset routine is always executed. As noted earlier, in the minimum mode, this instruction should initialize the stack pointer (SP). In the maximum mode, the first instruction should be an LDC instruction that initializes the stack page register (TP); the next instruction should initialize the stack pointer.

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

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

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

(Example)



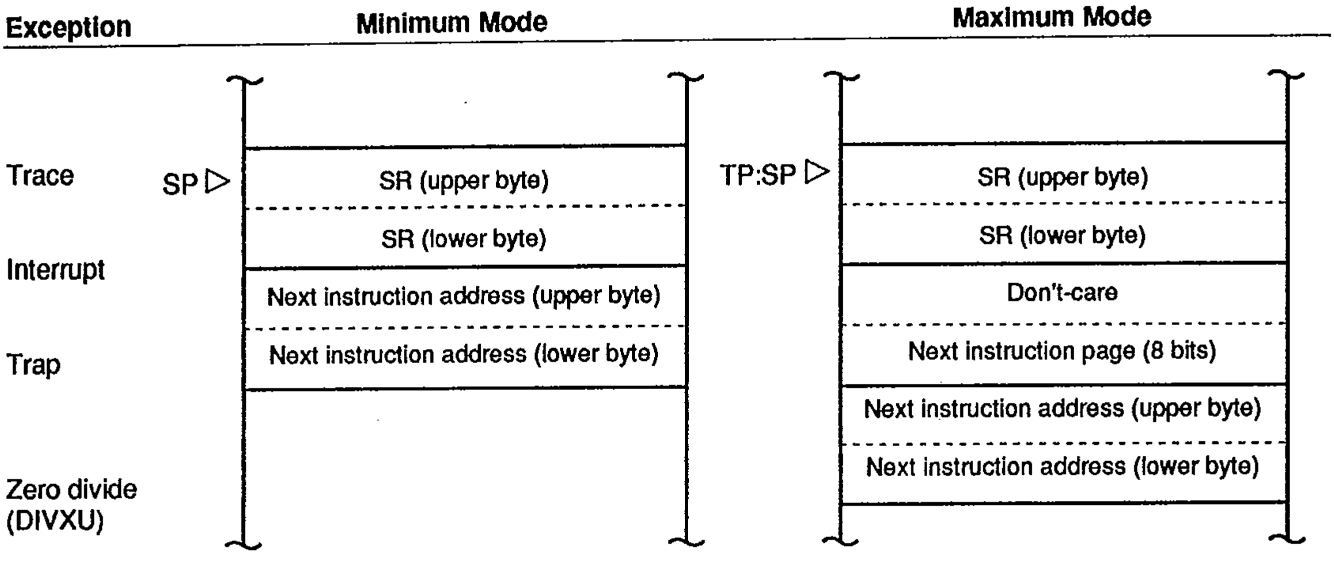
## 4.9 Stack Status after Completion of Exception Handling

T-49-19-16

The status of the stack after an exception-handling sequence is described below.

Table 4-3 shows the stack after completion of the exception-handling sequence for various types of exceptions in the minimum and maximum modes.

Table 4-3 Stack after Exception Handling Sequence

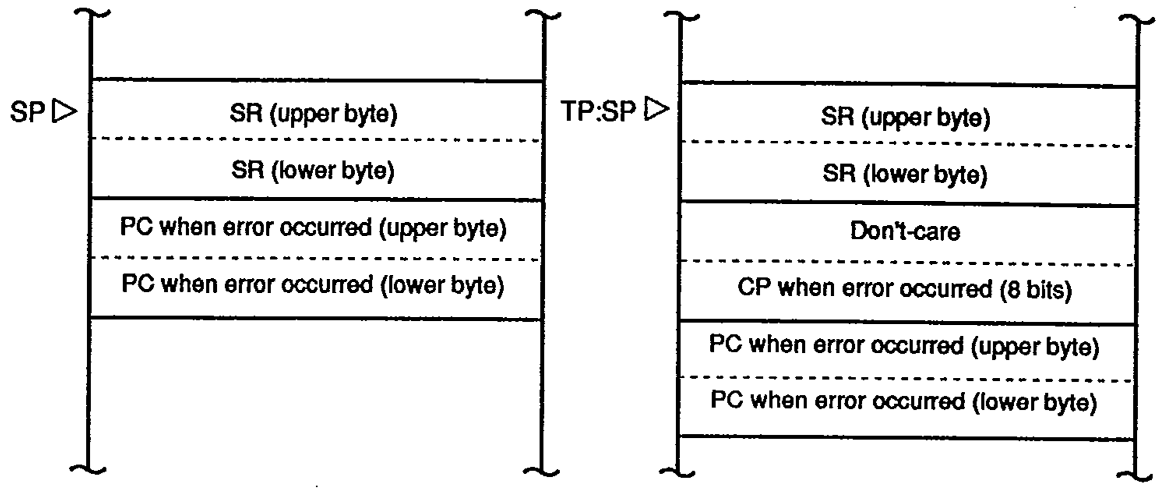


Note: The RTE instruction returns to the next instruction after the instruction being executed when the exception occurred.

## Table 4-3 Stack after Exception Handling Sequence (cont)

**Exception** **Minimum Mode** **Maximum Mode**

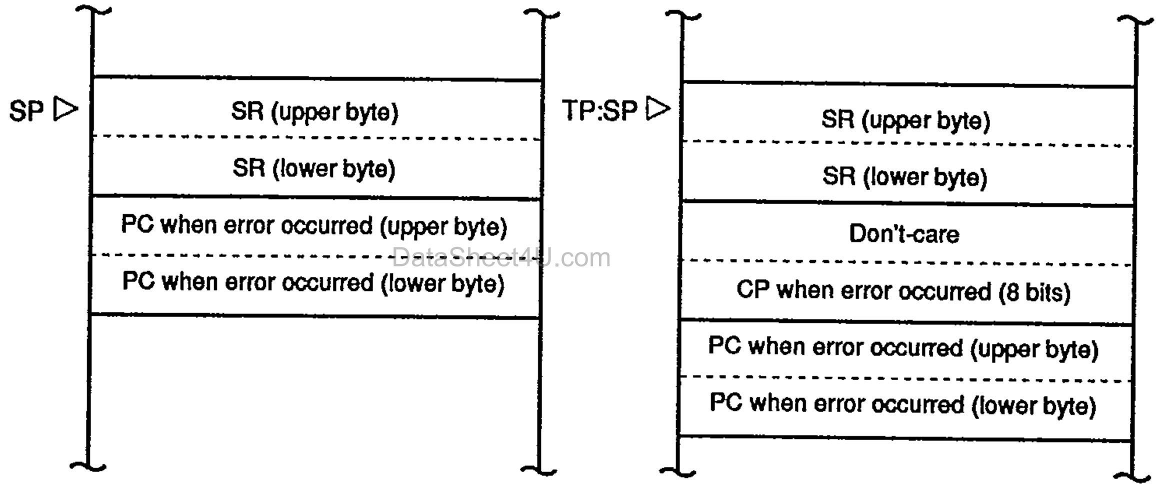
Invalid instruction



Note: The program counter value pushed on the stack is not necessarily the address of the first byte of the invalid instruction.

**Exception** **Minimum Mode** **Maximum Mode**

Address error



Note: The program counter value pushed on the stack is the address of the next instruction after the last instruction successfully executed.

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

The program counter value pushed on the stack for a trace, interrupt, trap, or zero divide exception is the address of the next instruction at the time when the interrupt or exception was accepted. The RTE instruction accordingly returns to the next instruction after the instruction executed before the exception-handling sequence.

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

The program counter value pushed on the stack for an address error or invalid instruction exception differs depending on the conditions when the exception occurred.

## 4.10 Notes on Use of the Stack

If the stack pointer is set to an odd address, an address error will occur when the stack is accessed during interrupt handling or for a subroutine call. The stack pointer should always point to an even address. To keep the stack pointer pointing to an even address, a program should use word data size when saving or restoring registers to and from the stack.

In the @–SP or @SP+ addressing mode, the CPU performs word access even if the instruction specifies byte size. (This is not true in the @–Rn and @Rn+ addressing modes when Rn is a register from R0 to R6.)

# Section 5 Interrupt Controller

T-49-19-16

## 5.1 Overview

The interrupt controller decides which interrupts to accept, and how to deal with multiple interrupts and other exceptions. It also decides whether an interrupt should be served by the CPU or by the data transfer controller (DTC). This section explains the features of the interrupt controller, describes its internal structure and control registers, and details the handling of interrupts.

For detailed information on the data transfer controller, see section 6, "Data Transfer Controller".

### 5.1.1 Features

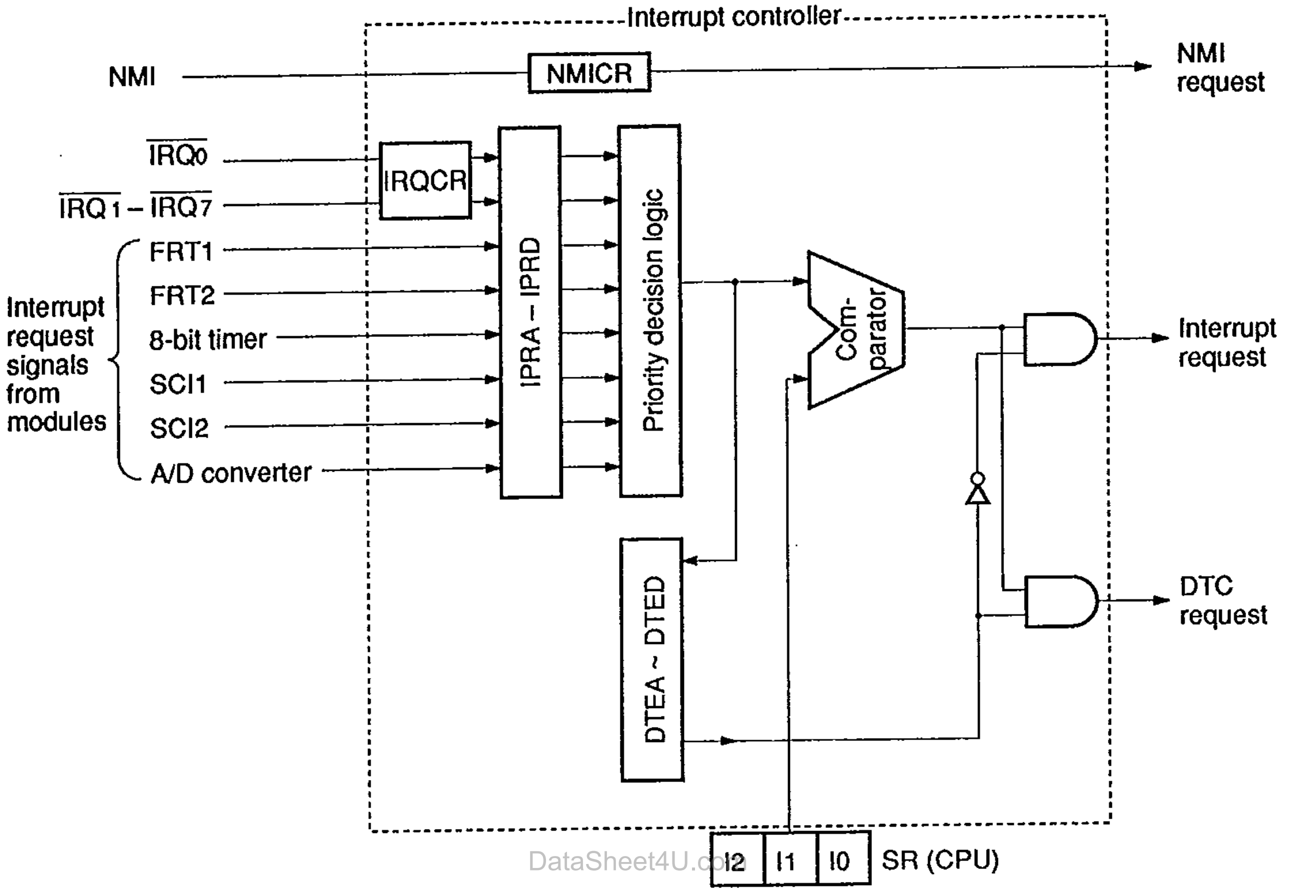
The main features of the interrupt controller are as follows:

- Interrupt priorities are user-programmable.
  User programs can set priority levels from 7 (high) to 0 (low) in four interrupt priority (IPR) registers for $IRQ_0$, $IRQ_1$ to $IRQ_7$, and each of the on-chip supporting modules—for every interrupt, that is, except the nonmaskable interrupt (NMI). NMI has the highest priority level (8) and is normally always accepted. An interrupt with priority level 0 is always masked.
- Multiple interrupts on the same level are served in a default priority order.
  Lower-priority interrupts remain pending until higher-priority interrupts have been handled.
- For most interrupts, software can select whether to have the interrupt served by the CPU or the on-chip data transfer controller (DTC).
  User programs can make this selection by setting and clearing bits in four data transfer enable (DTE) registers. The data transfer controller can be started by any interrupts except NMI, $IRQ_4$ to $IRQ_7$, the error interrupt (ERI) from the on-chip serial communication interface, and the overflow interrupts (FOVI and OVI) from the on-chip timers.
- Software can select the NMI edge and can enable or disable $\overline{IRQ_0}$ to $\overline{IRQ_7}$.
  The NMI control register (NMICR) determines whether a nonmaskable interrupt is triggered by the rising or falling edge of the NMI input signal. The IRQ control register (IRQCR) enables or disables $\overline{IRQ_0}$ to $\overline{IRQ_7}$.

### 5.1.2 Block Diagram

T-49-19-16

Figure 5-1 shows the block configuration of the interrupt controller.



(Legend)

| | |
|---|---|
| FRT: | Free-Running Timer |
| SCI: | Serial Communication Interface |
| SR: | Status Register |
| IPR: | Interrupt Priority Register |
| DTE: | Data Transfer Enable Register |
| NMICR: | Nonmaskable Interrupt Control Register |
| IRQCR: | Interrupt Request Control Register |

**Figure 5-1 Interrupt Controller Block Diagram**

### 5.1.3 Register Configuration

T-49-19-16

Table 5-1 lists the attributes of the registers used by the interrupt controller.

**Table 5-1 Interrupt Controller Registers**

| Name | | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|---|
| Interrupt priority register | A | IPRA | R/W | H'00 | H'FFF0 |
| | B | IPRB | R/W | H'00 | H'FFF1 |
| | C | IPRC | R/W | H'00 | H'FFF2 |
| | D | IPRD | R/W | H'00 | H'FFF3 |
| Data transfer enable register | A | DTEA | R/W | H'00 | F'FFF4 |
| | B | DTEB | R/W | H'00 | H'FFF5 |
| | C | DTEC | R/W | H'00 | H'FFF6 |
| | D | DTED | R/W | H'00 | H'FFF7 |
| NMI control register | | NMICR | R/W | H'FE | H'FFFC |
| IRQ control register | | IRQCR | R/W | H'00 | H'FFFD |

See section 6.2.5, "Data Transfer Enable Registers A to D", for detailed information about DTEA to DTED.

## 5.2 Interrupt Types

There are 27 distinct types of interrupts: 9 external interrupts originating off-chip and 18 internal interrupts originating in the on-chip supporting modules.

### 5.2.1 External Interrupts

The nine external interrupts are NMI and $IRQ_0$ to $IRQ_7$.

**NMI (Non Maskable Interrupt):** This interrupt has the highest priority level (8) and cannot be masked. An NMI is generated by input to the NMI pin. The input at the NMI pin is edge-sensed. A user program can select whether to have the interrupt occur on the rising edge or falling edge of the NMI input by setting or clearing the nonmaskable interrupt edge bit (NMIEG) in the NMI control register (NMICR).

In the NMI exception-handling sequence, the T (Trace) bit in the CPU status register (SR) is cleared to 0, and the interrupt mask level in $I_2$ to $I_0$ is set to 7, masking all other interrupts. The interrupt controller holds the NMI request until the NMI exception-handling sequence begins, then clears the NMI request, so if another interrupt is requested at the NMI pin during the NMI exception-handling sequence, the NMI exception-handling sequence will be carried out again.

T-49-19-16

**Coding Examples:**

To select the rising edge of the NMI input: `BSET.B #0, @H'FFFC`
To select the falling edge of the NMI input: `BCLR.B #0, @H'FFFC`

**IRQ$_0$ (Interrupt Request 0):** An IRQ$_0$ interrupt can be requested by a low input to the $\overline{\text{IRQ}_0}$ pin and/or a watchdog timer overflow. A low $\overline{\text{IRQ}_0}$ input requests an IRQ$_0$ interrupt if the interrupt request enable 0 bit (IRQ$_0$ E) in the IRQ control register (IRQCR) is set to 1. The interrupt controller samples the level of the $\overline{\text{IRQ}_0}$ pin directly, so this pin must be held low until the interrupt is accepted. Otherwise the request will be ignored.

A watchdog timer overflow requests an IRQ$_0$ interrupt if the TME bit is set to 1 and the WT/$\overline{\text{IT}}$ bit is cleared to 0 in the watchdog timer's control/status register. See section 12, "Watchdog Timer", for details of the watchdog timer.

The IRQ$_0$ interrupt can be assigned any priority level from 7 to 0 by setting the corresponding value in the upper four bits of IPRA. If bit 4 of data transfer enable register A (DTEA) is set to 1, an IRQ$_0$ interrupt starts the data transfer controller. Otherwise the interrupt is served by the CPU.

In the CPU interrupt-handling sequence for IRQ$_0$ the T bit of the status register is cleared to 0, and the interrupt mask level is set to the value in the upper four bits of IPRA.

**Coding Examples:**

To enable IRQ$_0$ to be requested by $\overline{\text{IRQ}_0}$ input: `BSET.B #0, @H'FFFD`
To assign priority level 7 to IRQ$_0$: `OR.B   #70, @H'FFF0`
To have IRQ$_0$ start the DTC: `BSET.B #4, @H'FFF4`

**IRQ$_1$ to IRQ$_7$ (Interrupt Request 1 to 7):** An IRQ$_1$ to IRQ$_7$ interrupt is requested by a high-to-low transition at the $\overline{\text{IRQ}_1}$ to $\overline{\text{IRQ}_7}$ pin. The IRQ$_1$ to IRQ$_7$ interrupt is enabled only when the interrupt request enable bit IRQ$_1$E to IRQ$_7$E in the IRQ control register is set to 1. The $\overline{\text{IRQ}_1}$ to $\overline{\text{IRQ}_7}$ input is latched in the interrupt controller and held until the interrupt request is accepted.

The IRQ1 to IRQ7 interrupts can be assigned any priority level from 7 (high) to 0 (low) by setting the corresponding value in the lower four bits of IPRA. These seven interrupts always have the same priority. They cannot be assigned priorities separately.

T-49-19-16

If bits 0 to 2 of data transfer enable register A (DTEA) are set to 1, IRQ1 to IRQ3 can start the data transfer controller. Otherwise the interrupt is served by the CPU. IRQ4 to IRQ7 cannot start the data transfer controller; they are always served by the CPU.

The interrupt controller holds IRQ1 to IRQ7 requests until the corresponding exception-handling sequence begins, then clears the request. Contention among IRQ1 to IRQ7 is resolved when the CPU accepts the interrupt by taking the interrupt with the highest priority first and holding lower-priority interrupts pending.

During the interrupt-handling routine, if the same external interrupt is requested again the request is held, but the exception-handling sequence is not carried out immediately because the interrupt is masked by bits I2 to I0 in the status register. On return from the interrupt-handling routine one more instruction is executed, then the pending exception-handling sequence is carried out.

In the CPU interrupt-handling sequence for IRQ1 to IRQ7, the T bit of the CPU status register is cleared to 0, and the interrupt mask level is set to the value in the lower four bits of IPRA.

**Coding Examples:**

| | |
|---|---|
| To enable IRQ1 to be requested by $\overline{\text{IRQ1}}$ input: | `BSET.B #1, @H'FFFD` |
| To assign priority level 7 to IRQ0 and level 5 to IRQ1 to IRQ7: | `MOV.B #75, @H'FFF0` |
| To have IRQ1 start the DTC: | `BSET.B #0, @H'FFF4` |

### 5.2.2 Internal Interrupts

Eighteen types of internal interrupts can be requested by the on-chip supporting modules. Each interrupt is separately vectored in the exception vector table, so it is not necessary for the user-coded interrupt handler routine to determine which type of interrupt has occurred.

Each of the internal interrupts can be enabled or disabled by setting or clearing an enable bit in the control register of the on-chip supporting module.

An interrupt priority level from 7 to 0 can be assigned to each on-chip supporting module by setting interrupt priority registers B to D. Within each module, different interrupts have a fixed priority order. For most of these interrupts, values set in data transfer enable registers B to D can select whether to have the interrupt served by the CPU or the data transfer controller.

T-49-19-16

In the CPU interrupt-handling sequence, the T bit of the CPU status register is cleared to 0, and the interrupt mask level in bits $I_2$ to $I_0$ is set to the value in the IPR.

### 5.2.3 Interrupt Vector Table

Table 5-2 lists the addresses of the exception vector table entries for each interrupt, and explains how their priority is determined. For the on-chip supporting modules, the priority level set in the interrupt priority register applies to the module as a whole: all interrupts from that module have the same priority level. A separate priority order is established among interrupts from the same module. If the same priority level is assigned to two or more modules and two interrupts are requested simultaneously from these modules, they are served in the priority order indicated in the rightmost column in table 5-2.

A reset clears the interrupt priority registers so that all interrupts except NMI start with priority level 0, meaning that they are unconditionally masked.

T-49-19-16

## Table 5-2 Interrupts, Vectors, and Priorities

| Interrupt | | Assignable Priority Levels (Initial Level) | IPR Bits | Priority within Module | Vector Table Entry Address: Minimum Mode | Vector Table Entry Address: Maximum Mode | Priority among Interrupts on Same Level* |
|---|---|---|---|---|---|---|---|
| NMI | | 8<br>(8) | — | — | H'0016 to H'0017 | H'002C to H'002F | High ↑ |
| IRQ | IRQ0 | 7 to 0<br>(0) | IPRA<br>Upper 4 bits | — | H'0040 to H'0041 | H'0080 to H'0083 | |
| | IRQ1 | 7 to 0 | IPRA | 6 | H'0042 to H'0043 | H'0084 to H'0087 | |
| | IRQ2 | | Lower 4 bits | 5 | H'0044 to H'0045 | H'0088 to H'008B | |
| | IRQ3 | | | 4 | H'0046 to H'0047 | H'008C to H'008F | |
| | IRQ4 | | | 3 | H'0048 to H'0049 | H'0090 to H'0093 | |
| | IRQ5 | | | 2 | H'004A to H'004B | H'0094 to H'0097 | |
| | IRQ6 | | | 1 | H'004C to H'004D | H'0098 to H'009B | |
| | IRQ7 | (0) | | 0 | H'004E to H'004F | H'009C to H'009F | |
| FRT1 | ICI | 7 to 0 | IPRB | 3 | H'0050 to H'0051 | H'00A0 to H'00A3 | |
| | OCIA | | Upper 4 bits | 2 | H'0052 to H'0053 | H'00A4 to H'00A7 | |
| | OCIB | | | 1 | H'0054 to H'0055 | H'00A8 to H'00AB | |
| | FOVI | (0) | | 0 | H'0056 to H'0057 | H'00AC to H'00AF | |
| FRT2 | ICI | 7 to 0 | IPRB | 3 | H'0058 to H'0059 | H'00B0 to H'00B3 | |
| | OCIA | | Lower 4 bits | 2 | H'005A to H'005B | H'00B4 to H'00B7 | |
| | OICB | | | 1 | H'005C to H'005D | H'00B8 to H'00BB | |
| | FOVI | (0) | | 0 | H'005E to H'005F | H'00BC to H'00BF | |
| 8-Bit timer | CMIA | 7 to 0 | IPRC | 2 | H'0060 to H'0061 | H'00C0 to H'00C3 | |
| | CMIB | | Upper 4 bits | 1 | H'0062 to H'0063 | H'00C4 to H'00C7 | |
| | OVI | (0) | | 0 | H'0064 to H'0065 | H'00C8 to H'00CB | |
| SCI1 | ERI | 7 to 0 | IPRC | 2 | H'0068 to H'0069 | H'00D0 to H'00D3 | |
| | RXI | | Upper 4 bits | 1 | H'006A to H'006B | H'00D4 to H'00D7 | |
| | TXI | (0) | | 0 | H'006C to H'006D | H'00D8 to H'00DB | |
| SCI2 | ERI | 7 to 0 | IPRD | 2 | H'0070 to H'0071 | H'00E0 to H'00E3 | |
| | RXI | | Upper 4 bits | 1 | H'0072 to H'0073 | H'00E4 to H'00E7 | |
| | TXI | (0) | | 0 | H'0074 to H'0075 | H'00E8 to H'00EB | |
| A/D converter | ADI | 7 to 0<br>(0) | IPRD<br>Lower 4 bits | — | H'0078 to H'0079 | H'00F0 to H'00F3 | Low |

Note: * If two or more interrupts are requested simultaneously, they are handled in order of priority level, as set in registers IPRA to IPRD. If they have the same priority level because they are requested from the same on-chip supporting module, they are handled in a fixed priority order within the module. If they are requested from different modules to which the same priority level is assigned, they are handled in the order indicated in the right-hand column.

## 5.3 Register Descriptions

### 5.3.1 Interrupt Priority Registers A to D (IPRA to IPRD)

T-49-19-16

$IRQ_0$, $IRQ_1$ to $IRQ_7$, and the on-chip supporting modules are each assigned three bits in one of the four interrupt priority registers (IPRA to IPRD). These bits specify a priority level from 7 (high) to 0 (low) for interrupts from the corresponding source. The drawing below shows the configuration of the interrupt priority registers. Table 5-3 lists their assignments to interrupt sources.

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

Note: Bits 7 and 3 are reserved. They cannot be modified and are always read as 0.

**Table 5-3 Assignment of Interrupt Priority Registers**

| | Interrupt Request Source | |
|---|---|---|
| **Register** | **Bits 6 to 4** | **Bits 2 to 0** |
| IPRA | $\overline{IRQ_0}$ | $\overline{IRQ_1}$ to $\overline{IRQ_7}$ |
| IPRB | FRT1 | FRT2 |
| IPRC | 8-Bit timer | SCI1 |
| IPRD | SCI2 | A/D converter |

As table 5-3 indicates, each interrupt priority register specifies priority levels for two interrupt sources. A user program can assign desired levels to these interrupt sources by writing 000 in bits 6 to 4 or bits 2 to 0 to set priority level 0, for example, or 111 to set priority level 7.

A reset clears registers IPRA to IPRD to H'00, so all interrupts except NMI are initially masked.

When the interrupt controller receives one or more interrupt requests, it selects the request with the highest priority and compares its priority level with the interrupt mask level set in bits $I_2$ to $I_0$ in the CPU status register. If the priority level is higher than the mask level, the interrupt controller passes the interrupt request to the CPU (or starts the data transfer controller). If the priority level is lower than the mask level, the interrupt controller leaves the interrupt request pending until the interrupt mask is altered to a lower level or the interrupt priority is raised. Similarly, if it receives two interrupt requests with the same priority level, the interrupt controller determines their priority as explained in table 5-2 and leaves the interrupt request with the lower priority pending.

The interrupt controller requires two system clock (ø) periods to determine the priority level of an interrupt. Accordingly, when an instruction modifies an instruction priority register, the new priority does not take effect until after the next instruction has been executed.

T-49-19-16

### 5.3.2 NMI Control Register (NMICR)—H'FFFC

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

The NMI control register (NMICR) is an 8-bit register that selects the edge of the NMI input signal which triggers a nonmaskable interrupt.

The NMICR is initialized to H'FF (falling edge) at a reset and in the hardware standby mode. It is not initialized in the software standby mode.

**Bit 7 to 0—Reserved:** These bits cannot be modified and are always read as 1.

**Bit 0—Nonmaskable Interrupt Edge (NMIEG):** This bit selects the valid edge of the NMI input signal.

| Bit 0<br>NMIEG | Description | |
|---|---|---|
| 0 | A nonmaskable interrupt is generated on the falling edge of the NMI input signal. | (Initial value) |
| 1 | A nonmaskable interrupt is generated on the rising edge of the NMI input signal. | |

### 5.3.3 IRQ Control Register (IRQCR)—H'FFFD

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IRQ7E | IRQ6E | IRQ5E | IRQ4E | IRQ3E | IRQ2E | IRQ1E | IRQ0E |
| 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 IRQ control register (IRQCR) enables or disables external interrupts on an individual basis. When an interrupt is enabled, the corresponding pin in port 1 or 4 can be used for interrupt request input. (The pin can also be read by the CPU as a port input pin.) The data direction bit in the port 1 or 4 data direction register must be cleared to 0 to designate the input mode.

The IRQCR is initialized to H'00 at a reset and in the hardware standby mode, disabling all interrupt requests. It is not initialized in the software standby mode.

**Bit 7—Interrupt Request 7 Enable (IRQ7E):** This bit determines whether a high-to-low transition at pin $P4_7$ is recognized as an $\overline{\text{IRQ}_7}$ interrupt request.

| Bit 7<br>IRQ7E | Description | |
|---|---|---|
| 0 | $P4_7$ is not used for $\overline{\text{IRQ}_7}$ input. | (Initial value) |
| 1 | $P4_7$ can be used for $\overline{\text{IRQ}_7}$ input.* | |

**Bit 6—Interrupt Request 6 Enable (IRQ6E):** This bit determines whether a high-to-low transition at pin $P4_6$ is recognized as an $\overline{\text{IRQ}_6}$ interrupt request.

| Bit 6<br>IRQ6E | Description | |
|---|---|---|
| 0 | $P4_6$ is not used for $\overline{\text{IRQ}_6}$ input. | (Initial value) |
| 1 | $P4_6$ can be used for $\overline{\text{IRQ}_6}$ input.* | |

**Bit 5—Interrupt Request 5 Enable (IRQ5E):** This bit determines whether a high-to-low transition at pin $P4_5$ is recognized as an $\overline{\text{IRQ}_5}$ interrupt request.

| Bit 5<br>IRQ5E | Description | |
|---|---|---|
| 0 | $P4_5$ is not used for $\overline{\text{IRQ}_5}$ input. | (Initial value) |
| 1 | $P4_5$ can be used for $\overline{\text{IRQ}_5}$ input.* | |

Note: * In modes 1 and 3 these pins cannot be used for $\overline{\text{IRQ}_7}$ to $\overline{\text{IRQ}_4}$ input because they are occupied by bits 15 to 12 of the address bus.

**Bit 4—Interrupt Request 4 Enable (IRQ4E):** This bit determines whether a high-to-low transition at pin $P4_4$ is recognized as an $\overline{\text{IRQ}_4}$ interrupt request.

T-49-19-16

**Bit 4**

| IRQ4E | Description | |
|---|---|---|
| 0 | $P4_4$ is not used for $\overline{\text{IRQ}_4}$ input. | (Initial value) |
| 1 | $P4_4$ can be used for $\overline{\text{IRQ}_4}$ input.* | |

Note: * In modes 1 and 3 these pins cannot be used for $\overline{\text{IRQ}_7}$ to $\overline{\text{IRQ}_4}$ input because they are occupied by bits 15 to 12 of the address bus.

**Bit 3—Interrupt Request 3 Enable (IRQ3E):** This bit determines whether a high-to-low transition at pin $P1_4$ is recognized as an $\overline{\text{IRQ}_3}$ interrupt request.

**Bit 3**

| IRQ3E | Description | |
|---|---|---|
| 0 | $P1_4$ is not used for $\overline{\text{IRQ}_3}$ input. | (Initial value) |
| 1 | $P1_4$ can be used for $\overline{\text{IRQ}_3}$ input.* | |

**Bit 2—Interrupt Request 2 Enable (IRQ2E):** This bit determines whether a high-to-low transition at pin $P1_3$ is recognized as an $\overline{\text{IRQ}_2}$ interrupt request.

**Bit 2**

| IRQ2E | Description | |
|---|---|---|
| 0 | $P1_3$ is not used for $\overline{\text{IRQ}_2}$ input. | (Initial value) |
| 1 | $P1_3$ can be used for $\overline{\text{IRQ}_2}$ input.* | |

**Bit 1—Interrupt Request 1 Enable (IRQ1E):** This bit determines whether a high-to-low transition at pin $P1_2$ is recognized as an $\overline{\text{IRQ}_1}$ interrupt request.

**Bit 1**

| IRQ1E | Description | |
|---|---|---|
| 0 | $P1_2$ is not used for $\overline{\text{IRQ}_1}$ input. | (Initial value) |
| 1 | $P1_2$ can be used for $\overline{\text{IRQ}_1}$ input.* | |

Note: * In modes 3 these pins cannot be used for $\overline{\text{IRQ}_3}$ to $\overline{\text{IRQ}_1}$ input because they are occupied by the page address bus.

**Bit 0—Interrupt Request 0 Enable (IRQ0E):** This bit determines whether a low input at pin $P1_1$ is recognized as an $\overline{IRQ_0}$ interrupt request.

T-49-19-16

| Bit 0<br>IRQ0E | Description | |
|---|---|---|
| 0 | $P1_1$ is not used for $\overline{IRQ_0}$ input. | (Initial value) |
| 1 | $P1_1$ can be used for $\overline{IRQ_0}$ input. | |

## 5.4 Interrupt-Handling Sequence

### 5.4.1 Interrupt-Handling Flow

The interrupt-handling sequence follows the flowchart in figure 5-2, which also covers address-error and trace exceptions. Note that address error, trace exception, and NMI requests bypass the interrupt controller's priority decision logic and are routed directly to the CPU.

1. Interrupt requests are generated by one or more on-chip supporting modules or external interrupt sources.

2. The interrupt controller checks the interrupt priorities set in the IPRA to IPRD and selects the interrupt with the highest priority. Interrupts with lower priorities remain pending. Among interrupts with the same priority level, the interrupt controller determines priority as explained in table 5-2.

3. The interrupt controller compares the priority level of the selected interrupt request with the mask level in the CPU status register (bits $I_2$ to $I_0$). If the priority level is equal to or less than the mask level, the interrupt request remains pending. If the priority level is higher than the mask level, the interrupt controller accepts the interrupt request and proceeds to the next step.

4. The interrupt controller checks the corresponding bit (if any) in the data transfer enable registers (DTEA to DTEB). If this bit is set to 1, the data transfer controller is started. Otherwise, the CPU interrupt exception-handling sequence is started. When the data transfer controller is started, the interrupt request is cleared (except for interrupt requests from the serial communication interface, which are cleared by writing to the TDR or reading the RDR).

If the data transfer enable bit is cleared to 0 (or is nonexistent), the sequence proceeds as follows. For the case in which the data transfer controller is started, see section 6, "Data Transfer Controller".

5. After the CPU has finished executing the current instruction, the program counter and status register (in minimum mode) or program counter, code page register, and status register (in maximum mode) are saved to the stack, leaving the stack in the condition shown in figure 5-3 (a) or (b). The program counter value saved on the stack is the address of the next instruction to be executed.

T-49-19-16

6. The T (Trace) bit of the status register is cleared to 0, and the priority level of the interrupt is copied to bits $I_2$ to $I_0$, thus masking further interrupts unless they have a higher priority level. When an NMI is accepted, the interrupt mask level in bits $I_2$ to $I_0$ is set to 7.

7. The interrupt controller generates the vector address of the interrupt, and the entry at this address in the exception vector table is read to obtain the starting address of the user-coded interrupt handling routine.

In step 7, the same difference between the minimum and maximum modes exists as in the reset handling sequence. In the minimum mode, one word is copied from the vector table to the program counter, then the interrupt-handling routine starts executing from the address indicated in the program counter. In the maximum mode, two words are read. The lower byte of the first word is copied to the code page register. The second word is copied to the program counter. The interrupt-handling routine starts executing from the address indicated in the code page register and program counter.

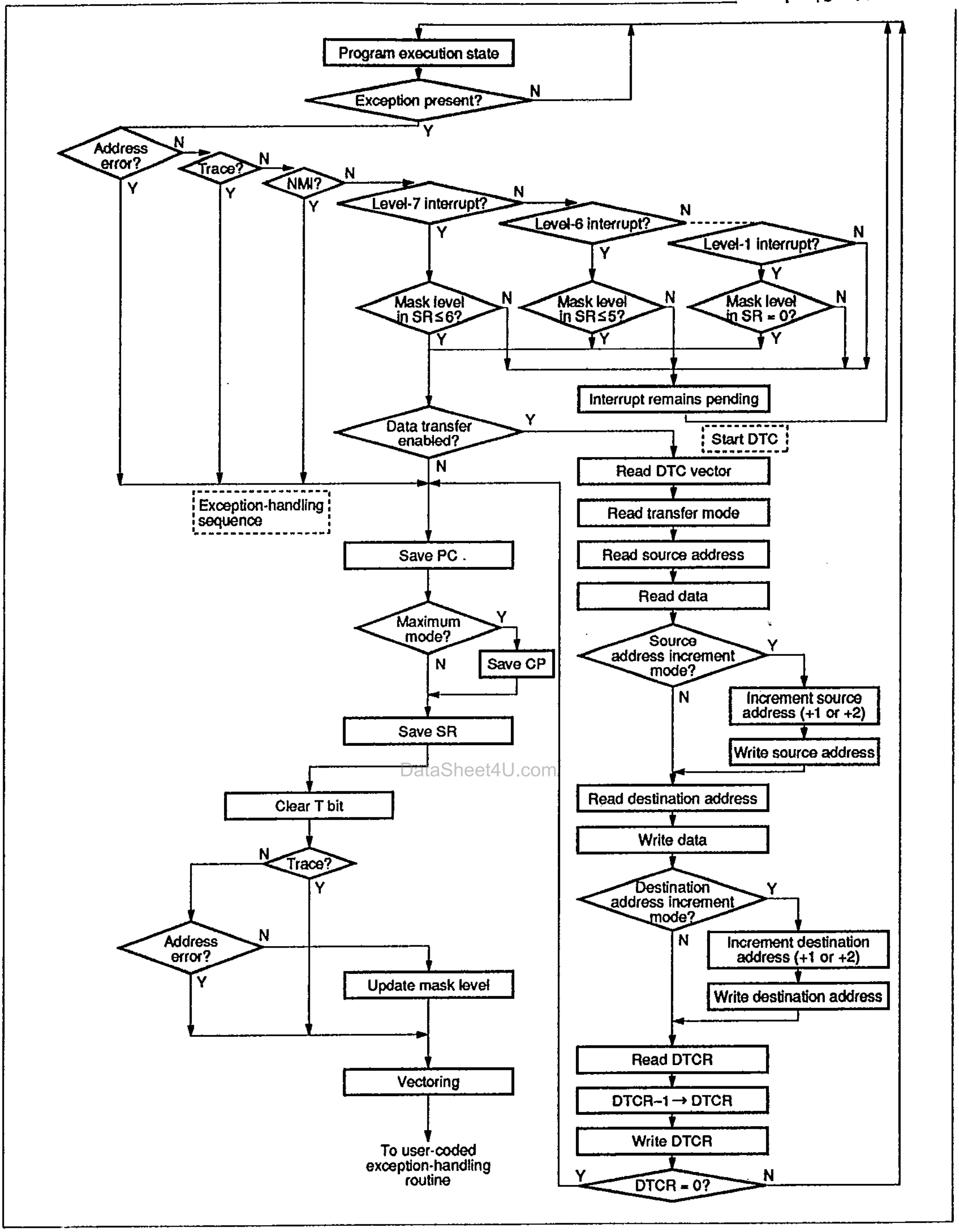
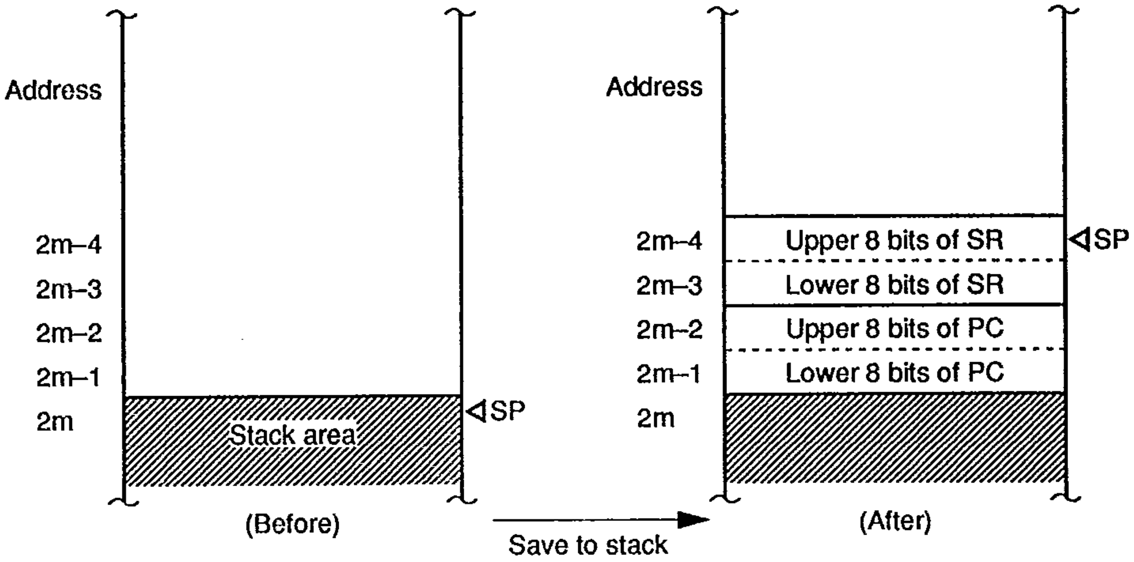


**Figure 5-2 Interrupt Handling Flowchart**

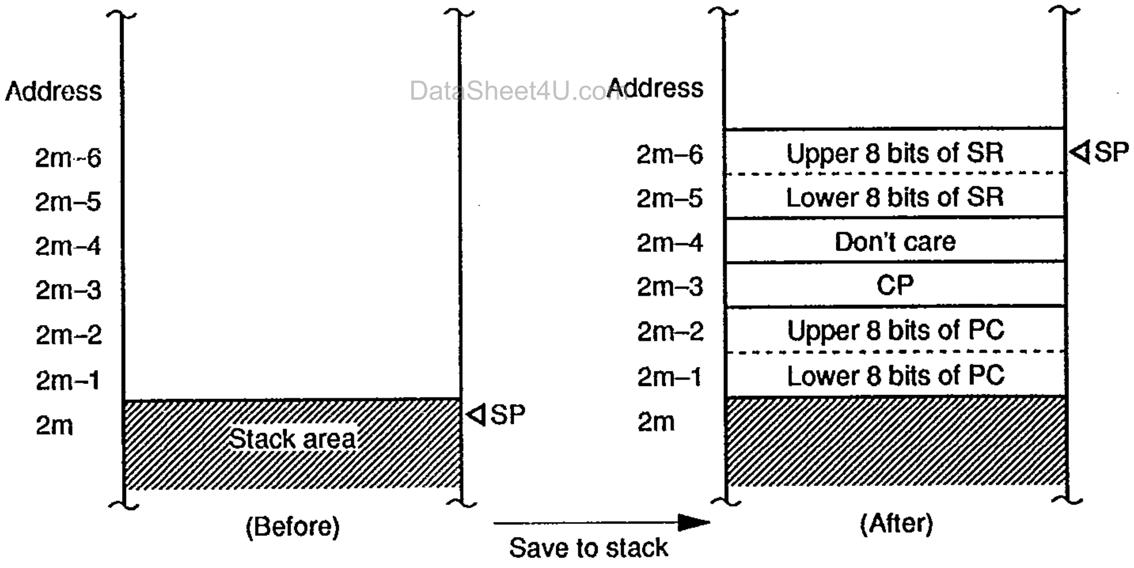
### 5.4.2 Stack Status after Interrupt Exception-Handling Sequence

Figure 5-3 (a) and (b) show the stack before and after the interrupt exception-handling sequence.



Notes: 1. PC: The address of the next instruction to be executed is saved.
2. Register saving and restoring must start at an even address.

Figure 5-3 (a) Stack before and after Interrupt Exception-Handling (Minimum Mode)



Notes: 1. PC: The address of the next instruction to be executed is saved.
2. Register saving and restoring must start at an even address.

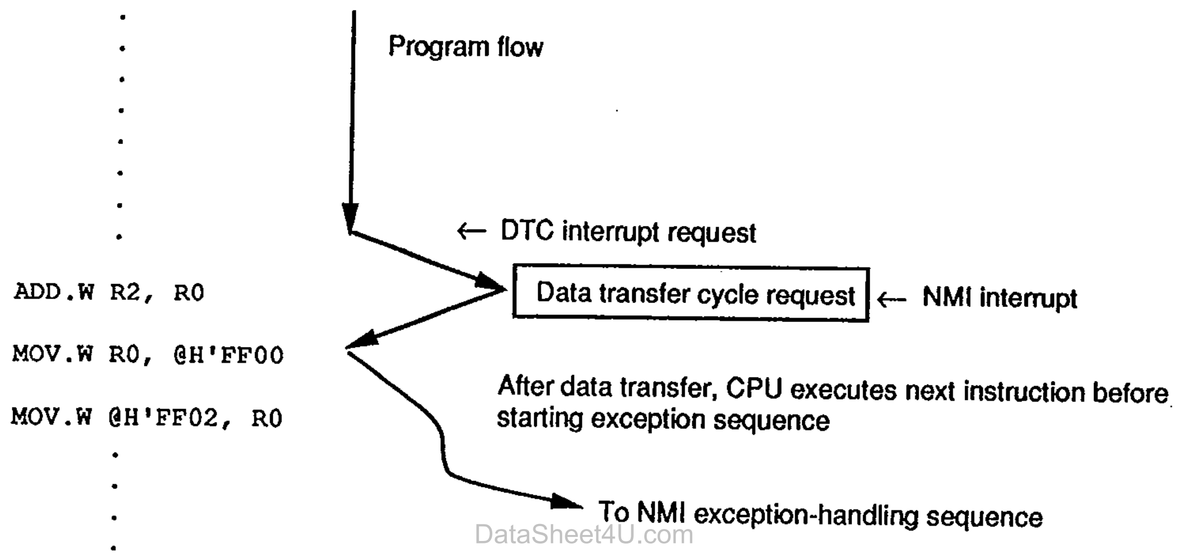
Figure 5-3 (b) Stack before and after Interrupt Exception-Handling (Maximum Mode)

### 5.4.3 Timing of Interrupt Exception-Handling Sequence

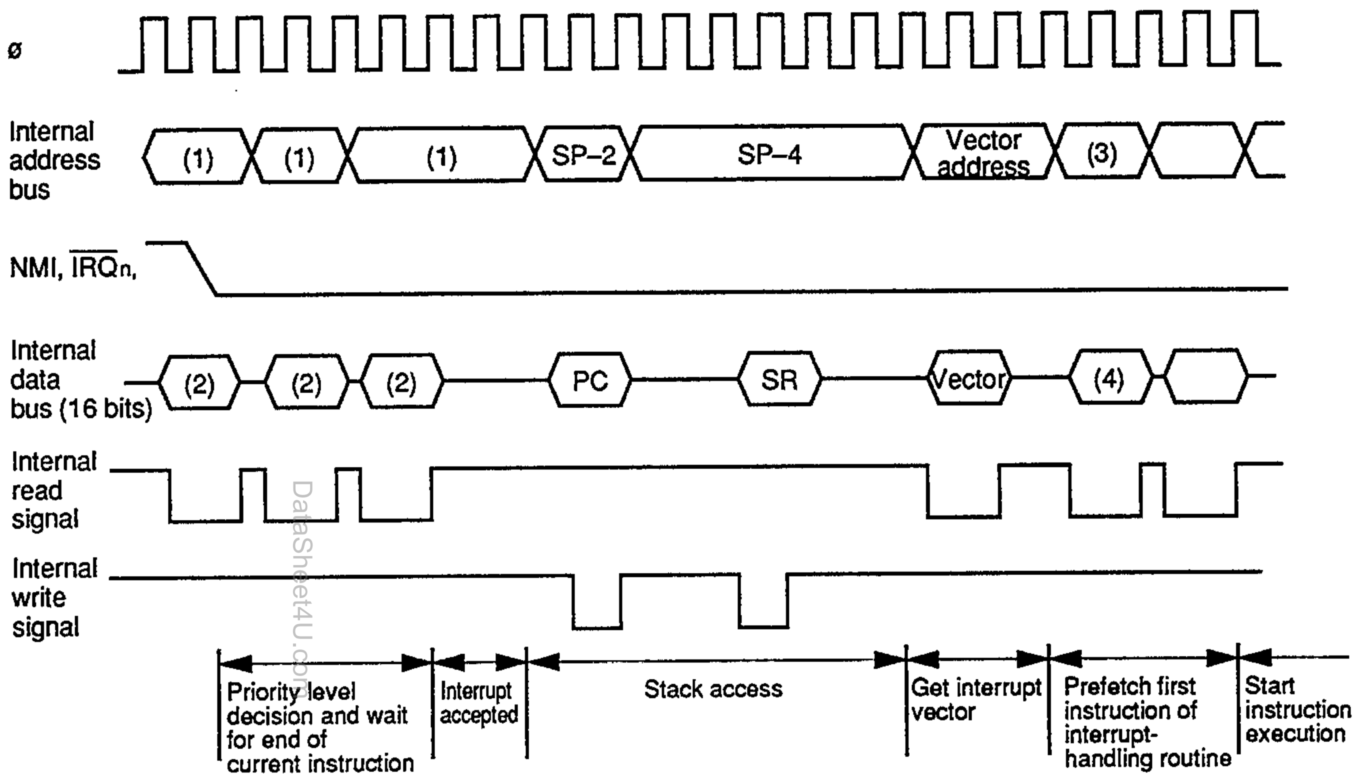
Figure 5-4 shows the timing in minimum mode when the program area and stack are both in on-chip memory and the user-coded interrupt-handling routine starts at an even address. Figure 5-5 shows the timing in maximum mode when the program area and stack are both in external memory.

## 5.5 Interrupts During Operation of the Data Transfer Controller

If an interrupt is requested during a DTC data transfer cycle, the interrupt is not accepted until the data transfer cycle has been completed and the next instruction has been executed. This is true even if the interrupt is an NMI. An example is shown below.



```
ADD.W R2, R0
MOV.W R0, @H'FF00
MOV.W @H'FF02, R0
```

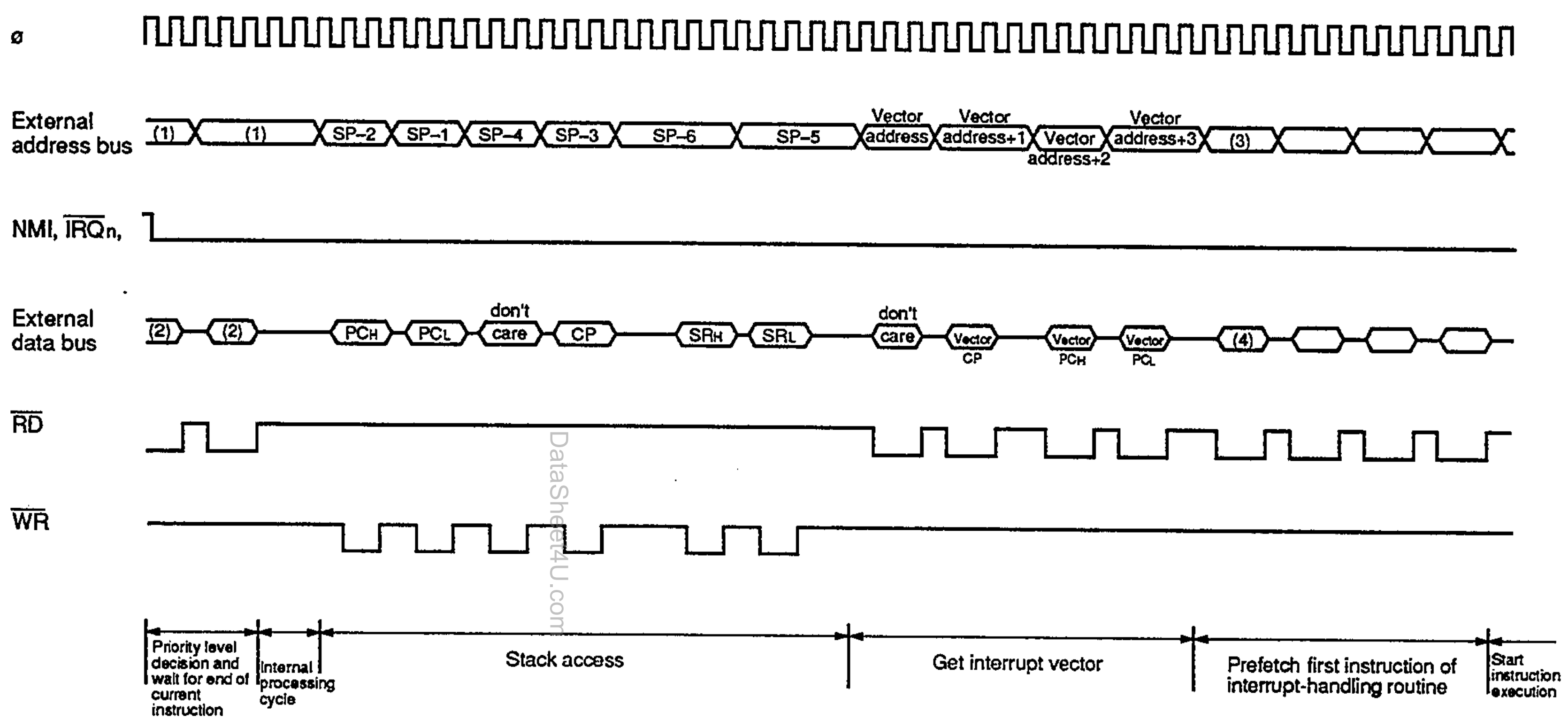


(1) Instruction prefetch address
(2) Instruction code
(3) Starting address of interrupt-handling routine
(4) First instruction of interrupt-handling routine

Note: This timing chart applies to the minimum mode when the program and stack areas are both in on-chip memory and the interrupt-handling routine starts at an even address.

Figure 5-4 Interrupt Sequence (Minimum Mode, On-Chip Memory)

T-49-19-16



(1) Instruction prefetch address

(2) Instruction code

(3) Starting address of interrupt-handling routine

(4) First instruction of interrupt-handling routine

Note: This timing chart applies to the maximum mode when the program and stack areas are both in external memory. Instruction execution is preceded by an interrupt vector fetch and 4-byte (4 bus cycles) instruction prefetch.

Figure 5-5 Interrupt Sequence (Maximum Mode, External Memory)

## 5.6 Interrupt Response Time

Table 5-4 indicates the number of states that may elapse between the generation of an interrupt request and the execution of the first instruction of the interrupt-handling routine, assuming that the interrupt is not masked and not preempted by a higher-priority interrupt. Since word access is performed to on-chip memory areas, fastest interrupt service can be obtained by placing the program in on-chip ROM and the stack in on-chip RAM.

**Table 5-4 Number of States before Interrupt Service**

| | | | Number of States | |
|---|---|---|---|---|
| **No.** | **Reason for Wait** | | **Minimum Mode** | **Maximum Mode** |
| 1 | Interrupt priority decision and comparison with mask level in CPU status register | | 2 states | |
| 2 | Maximum number of states to completion of current instruction | Instruction is in on-chip memory | x<br>(x = 38 for LDM instruction specifying all registers) | |
| | | Instruction is in external memory | y<br>(y = 74 + 16m for LDM instruction specifying all registers) | |
| 3 | Number of states from saving of PC and SR (or PC, SR, and CP) until first instruction of interrupt-handling routine is prefetched. | Stack is in on-chip RAM | 16 | 21 |
| | | Stack is in external memory | 28 + 6m | 41 + 10m |
| Total | Stack is in on-chip RAM | Instruction is in on-chip memory | 18 + x<br>(56) | 23 + x<br>(61) |
| | | Instruction is in external memory | 18 + y<br>(92 + 16m) | 23 + y<br>(97 + 16m) |
| | Stack is in external RAM | Instruction is in on-chip memory | 30 + 6m + x<br>(68 + 6m) | 43 + 10m + x<br>(81 + 10m) |
| | | Instruction is in external memory | 30 + 6m + y<br>(104 + 22m) | 43 + 10m + y<br>(117 + 26m) |

Notes: m: Number of wait states inserted in external memory access.
Values in parentheses are for the LDM instruction specifying all registers.

T-49-19-16

## 6.1 Overview

The H8/520 chip includes a data transfer controller (DTC) that can be started by designated interrupts to transfer data from a source address to a destination address located in page 0. These addresses include in particular the registers of the on-chip supporting modules and I/O ports. Typical uses of the DTC are to change the setting of a control register of an on-chip supporting module in response to an interrupt from that module, or to transfer data from memory to an I/O port or the serial communication interface. Once set up, the transfer is interrupt-driven, so it proceeds independently of program execution, although program execution temporarily stops while each byte or word is being transferred.

The data transfer functions of the DTC could also be performed by the CPU, but the DTC offers three advantages:

- It is faster.
- It requires less program coding.
- It has its own registers and does not require CPU registers to be used as pointers, etc.

### 6.1.1 Features

The main features of the DTC are listed below:

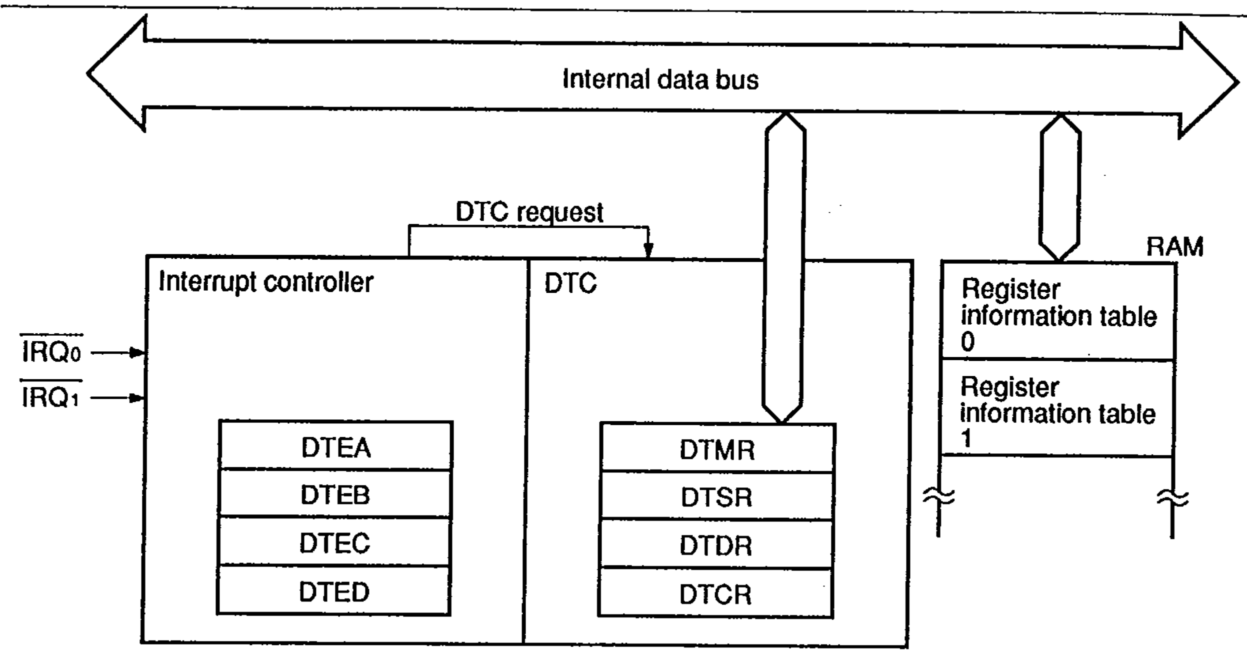
- The source address and destination address can be set anywhere in the 64-kbyte address space of page 0.
- The DTC can be programmed to transfer one byte or one word of data per interrupt.
- The DTC can be programmed to increment the source address and/or destination address after each byte or word is transferred.
- After transferring a designated number of bytes or words, the DTC generates a CPU interrupt with the vector of the interrupt source that started the DTC.
- This designated data transfer count can be set from 1 to 65,536 bytes or words.

### 6.1.2 Block Diagram

Figure 6-1 shows a block diagram of the DTC.

The four DTC control registers (DTMR, DTSR, DTDR, and DTCR) are invisible to the CPU, but corresponding information is kept in a register information table in memory. A separate table is maintained for each DTC interrupt type. When an interrupt requests DTC service, the DTC loads its control registers from the table in memory, transfers the byte or word of data, and writes any altered register information back to memory.

T-49-19-16



(Legend)

DTMR: Data Transfer Mode Register
DTSR: Data Transfer Source Address Register
DTDR: Data Transfer Destination Address Register
DTCR: Data Transfer Count Register
DTEA – DTED: Data Transfer Enable Register A – D

Figure 6-1 Block Diagram of Data Transfer Controller

### 6.1.3 Register Configuration

The four DTC control registers are listed in table 6-1. These registers are not located in the address space and cannot be written or read by the CPU. To set information in these registers, a program must write the information in a table in memory from which it will be loaded by the DTC.

Table 6-1 Internal Control Registers of the DTC

| Name | Abbreviation | Read/Write |
|---|---|---|
| Data transfer mode register | DTMR | Disabled |
| Data transfer source address register | DTSR | Disabled |
| Data transfer destination address register | DTDR | Disabled |
| Data transfer count register | DTCR | Disabled |

Starting of the DTC is controlled by the four data transfer enable registers, which are located in high addresses in page 0. Table 6-2 lists these registers.

T-49-19-16

**Table 6-2 Data Transfer Enable Registers**

| Name | | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|---|
| Data transfer | A | DTEA | R/W | H'00 | H'FFF4 |
| enable register | B | DTEB | R/W | H'00 | H'FFF5 |
| | C | DTEC | R/W | H'00 | H'FFF6 |
| | D | DTED | R/W | H'00 | H'FFF7 |

## 6.2 Register Descriptions

### 6.2.1 Data Transfer Mode Register (DTMR)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Sz | SI | DI | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Read/Write | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

The data transfer mode register is a 16-bit register, the first three bits of which designate the data size and specify whether to increment the source and destination addresses.

**Bit 15—Sz (Size):** This bit designates the size of the data transferred.

**Bit 15**

| Sz | Description |
|---|---|
| 0 | Byte transfer |
| 1 | Word transfer* (two bytes at a time) |

Note: * For word transfer, the source and designation addresses must be even addresses.

**Bit 14—SI (Source Increment):** This bit specifies whether to increment the source address.

**Bit 14**

| SI | Description |
|---|---|
| 0 | Source address is not incremented. |
| 1 | 1. If Sz = 0: Source address is incremented by +1 after each data transfer.<br>2. If Sz = 1: Source address is incremented by +2 after each data transfer. |

**Bit 13—DI (Destination Increment):** This bit specifies whether to increment the destination address.

T-49-19-16

| Bit 13<br>DI | Description |
|---|---|
| 0 | Destination address is not incremented. |
| 1 | 1. If Sz = 0: Destination address is incremented by +1 after each data transfer.<br>2. If Sz = 1: Destination address is incremented by +2 after each data transfer. |

**Bits 12 to 0—Reserved Bits:** These bits are reserved.

### 6.2.2 Data Transfer Source Address Register (DTSR)

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

The data transfer source register is a 16-bit register that designates the data transfer source address. For word transfer this must be an even address. In the maximum mode, this address is implicitly located in page 0.

### 6.2.3 Data Transfer Destination Register (DTDR)

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

The data transfer destination register is a 16-bit register that designates the data transfer destination address. For word transfer this must be an even address. In the maximum mode, this address is implicitly located in page 0.

### 6.2.4 Data Transfer Count Register (DTCR)

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

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

T-49-19-16

The data transfer count register is decremented automatically after each byte or word is transferred. When its value reaches 0, indicating that the designated number of bytes or words have been transferred, a CPU interrupt is generated with the vector of the interrupt that requested the data transfer.

### 6.2.5 Data Transfer Enable Registers A to D (DTEA to DTED)

These four registers designate whether an interrupt starts the DTC. The bits in these registers are assigned to interrupts as indicated in table 6-3. No bits are assigned to the NMI, $IRQ_4$, $IRQ_5$, $IRQ_6$, $IRQ_7$, FOVI, OVI, and ERI interrupts, which cannot request data transfers.

| 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 |

**Table 6-3 Assignment of Data Transfer Enable Registers**

| Register | Interrupt Source Module | Bits 7 to 4 | | | | Interrupt Source Module | Bits 3 to 0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | | 3 | 2 | 1 | 0 |
| DTEA | $\overline{IRQ_0}$ | — | — | — | $IRQ_0$ | $\overline{IRQ_3}$ - $\overline{IRQ_1}$ | — | $IRQ_3$ | $IRQ_2$ | $IRQ_1$ |
| DTEB | FRT1 | — | OCIB | OCIA | ICI | FRT2 | — | OCIB | OCIA | ICI |
| DTEC | 8-Bit timer | — | — | CMIB | CMIA | SCI1 | — | TXI | RXI | — |
| DTED | SCI2 | — | TXI | RXI | — | A/D converter | — | — | — | ADI |

Note: Bits marked "—" should always be cleared to 0.

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

Only the 16 interrupts indicated in table 6-3 can request DTC service. DTE bits not assigned to any interrupt (indicated by "—" in table 6-3) should be left cleared to 0.

**Note on Timing of DTE Modifications:** The interrupt controller requires two system clock (ø) periods to determine the priority level of an interrupt. Accordingly, when an instruction modifies a data transfer enable register, the new setting does not take effect until after the next instruction has been executed.

T-49-19-16

## 6.3 Data Transfer Operation

### 6.3.1 Data Transfer Cycle

When started by an interrupt, the DTC executes the following data transfer cycle:

1. From the DTC vector table, the DTC reads the address at which the register information table for that interrupt is located in memory.
2. The DTC loads the data transfer mode register and source address register from this table and reads the data (one byte or word) from the source address.
3. If so specified in the mode register, the DTC increments the source address register and writes the new source address back to the table in memory.
4. The DTC loads the data transfer destination address register and writes the byte or word of data to the destination address.
5. If so specified in the mode register, the DTC increments the destination address register and writes the new destination address back to the table in memory.
6. The DTC loads the data transfer count register from the table in memory, decrements the data count, and writes the new count back to memory.
7. If the data transfer count is now 0, the DTC generates a CPU interrupt. The interrupt vector is the vector of the interrupt type that started the DTC.

At an appropriate point during this procedure the DTC also clears the interrupt request by clearing the corresponding flag bit in the status register of the on-chip supporting module to 0. (For IRQ$_1$ to IRQ$_3$, the DTC clears an internal latch.)

But the DTC does not clear the data transfer enable bit in the data transfer enable register. This action, if necessary, must be taken by the user-coded interrupt-handling routine invoked at the end of the transfer.

The data transfer cycle is shown in a flowchart in figure 6-2.

For the steps from the occurrence of the interrupt up to the start of the data transfer cycle, see section 5.4.1, "Interrupt-Handling Flow".

T-49-19-16

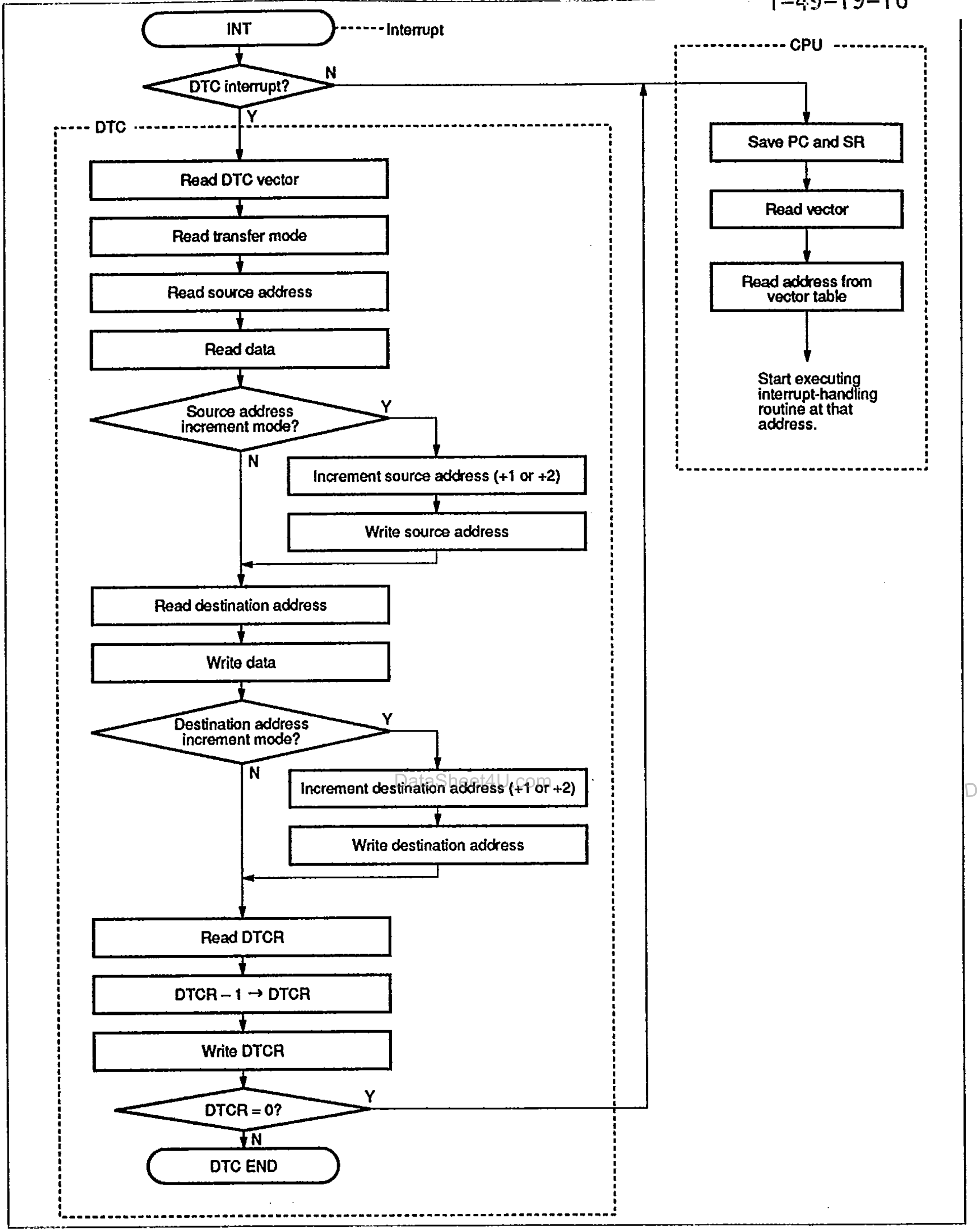
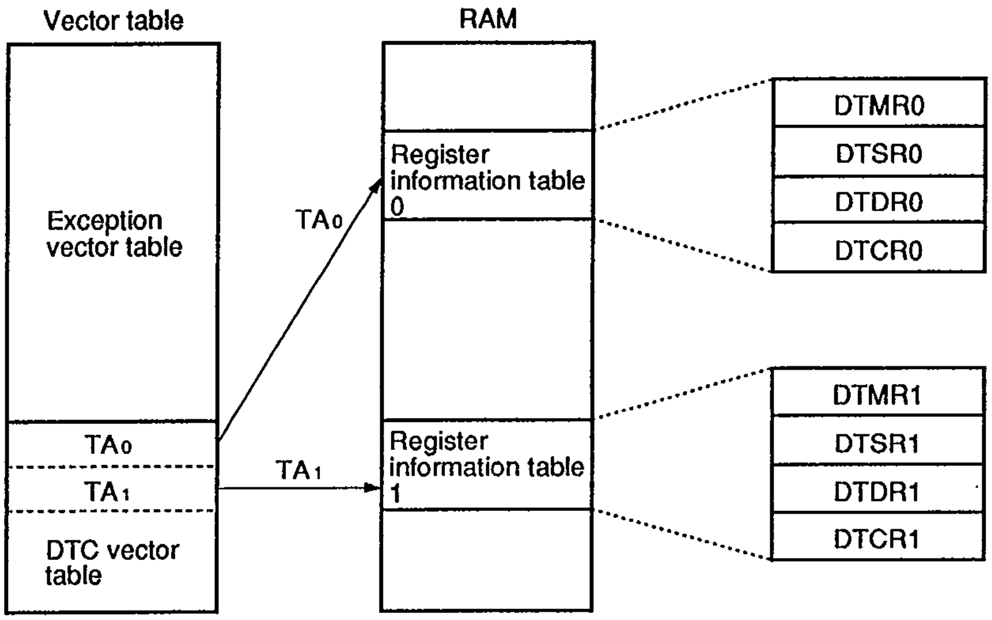


**Figure 6-2 Flowchart of Data Transfer Cycle**

### 6.3.2 DTC Vector Table

The DTC vector table is located immediately following the exception vector table at the beginning of page 0 in memory. For each interrupt that can request DTC service, the DTC vector table provides a pointer to an address in memory where the table of DTC control register information for that interrupt is stored. The register information tables can be placed in any available locations in page 0.



Notes: 1. TA0, TA1, ...: Addresses of DTC register information tables in memory.
2. The DTC register information tables should normally be located in RAM, but they may be located in ROM if it is not necessary to update the register information. Specifically, DTC register information can be located in ROM if neither the source nor the destination address is incremented and the desired number of data transfers is one (DTCR = 0) or infinite (DTCR > 0).

**Figure 6-3 DTC Vector Table**

In minimum mode, each entry in the DTC vector table consists of two bytes, pointing to an address in page 0. In maximum mode, for hardware reasons, each DTC vector table entry consists of four bytes but the first two bytes are ignored; the last two bytes point to an address which is implicitly assumed to be in page 0, regardless of the current page specifications.

Figure 6-4 shows one DTC vector table entry in minimum and maximum mode.

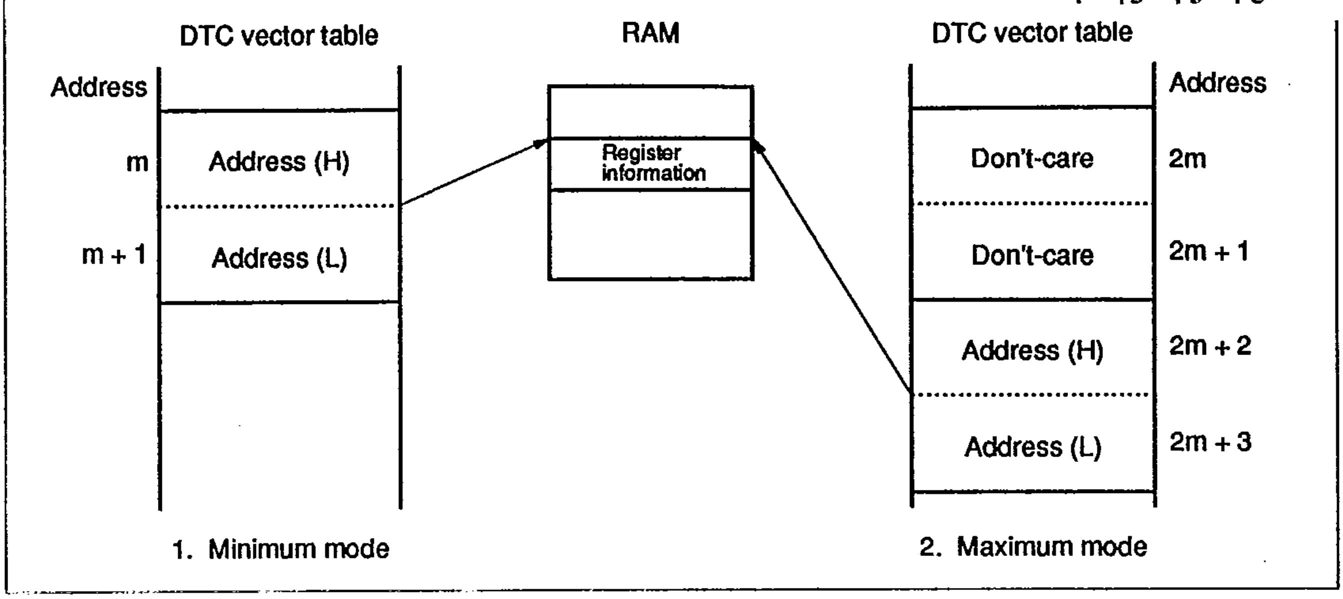


Figure 6-4 DTC Vector Table Entry

Table 6-4 lists the addresses of the entries in the DTC vector table for each interrupt.

**Table 6-4 Addresses of DTC Vectors**

| Interrupt | | Address of DTC Vector<br>Minimum Mode | <br>Maximum Mode |
|---|---|---|---|
| IRQ | $IRQ_0$ | H'0080 to H'0081 | H'0100 to H'0103 |
| | $IRQ_1$ | H'0082 to H'0083 | H'0104 to H'0107 |
| | $IRQ_2$ | H'0084 to H'0085 | H'0108 to H'010B |
| | $IRQ_3$ | H'0086 to H'0087 | H'010C to H'010F |
| FRT1 | ICI | H'0090 to H'0091 | H'0120 to H'0123 |
| | OCIA | H'0092 to H'0093 | H'0124 to H'0127 |
| | OCIB | H'0094 to H'0095 | H'0128 to H'012B |
| FRT2 | ICI | H'0098 to H'0099 | H'0130 to H'0133 |
| | OCIA | H'009A to H'009B | H'0134 to H'0137 |
| | OCIB | H'009C to H'009D | H'0138 to H'013B |

T-49-19-16

Table 6-4 Addresses of DTC Vectors (cont)

| **Interrupt** | | **Address of DTC Vector**<br>**Minimum Mode** | **Maximum Mode** |
|---|---|---|---|
| 8-Bit timer | CMIA | H'00A0 to H'00A1 | H'0140 to H'0143 |
| | CMIB | H'00A2 to H'00A3 | H'0144 to H'0147 |
| Serial communication interface 1 | RXI | H'00AA to H'00AB | H'0154 to H'0157 |
| | TXI | H'00AC to H'00AD | H'0158 to H'015B |
| Serial communication interface 2 | RXI | H'00B2 to H'00B3 | H'0164 to H'0157 |
| | TXI | H'00B4 to H'00B5 | H'0168 to H'016B |
| A/D converter | ADI | H'00B8 to H'00B9 | H'0170 to H'0173 |

### 6.3.3 Location of Register Information in Memory

For each interrupt, the DTC control register information is stored in four consecutive words in memory in the order shown in figure 6-5.

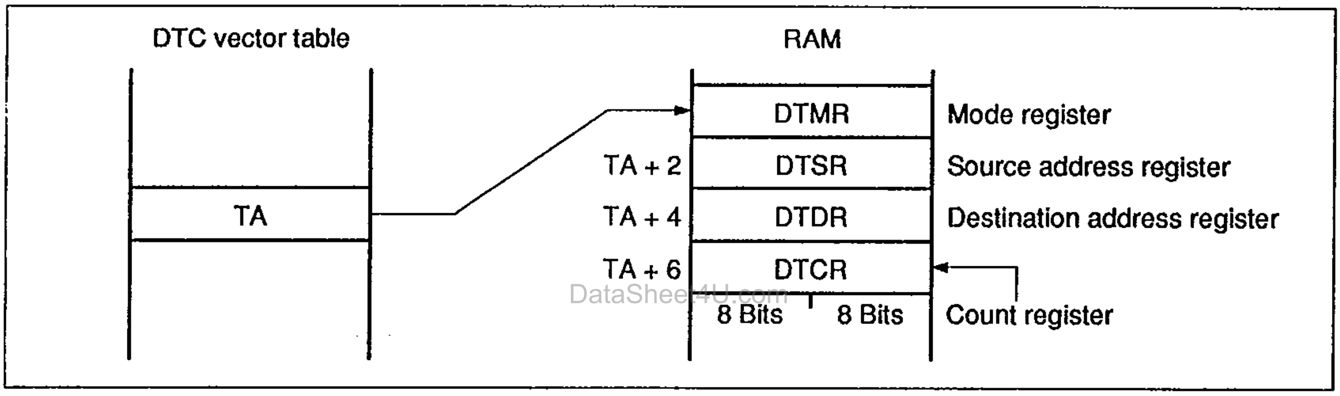


Figure 6-5 Order of Register Information

### 6.3.4 Length of Data Transfer Cycle

#### 1. Register Information in On-Chip RAM

Table 6-5 lists the number of states required per data transfer, assuming that the DTC control register information is stored in on-chip RAM. This is the number of states required for loading and saving the DTC control registers and transferring one byte or word of data. Two cases are considered: a transfer between on-chip RAM and a register belonging to an I/O port or on-chip supporting module (i.e., a register in the register field from addresses H'FF80 to H'FFFF); and a transfer between such a register and external RAM.

**Table 6-5 Number of States per Data Transfer**

| Increment Mode | | On-Chip RAM ↔ Module or I/O Register | | External RAM ↔ Module or I/O Register | |
|---|---|---|---|---|---|
| **Source (SI)** | **Destination (DI)** | **Byte Transfer** | **Word Transfer** | **Byte Transfer** | **Word Transfer** |
| 0 | 0 | 31 | 34 | 32 | 38 |
| 0 | 1 | 33 | 36 | 34 | 40 |
| 1 | 0 | 33 | 36 | 34 | 40 |
| 1 | 1 | 35 | 38 | 36 | 42 |

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

The values in table 6-5 are calculated from the formula:

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

Where Ms and MD have the following meanings:

Ms: Number of states for reading source data

MD: Number of states for writing destination data

The values of Ms and MD depend on the data location as follows:

a. Byte or word data in on-chip RAM: ➪ 2 states

b. Byte data in external RAM or register field: ➪ 3 states

c. Word data in external RAM or register field: ➪ 6 states

### 2. Register Information in External RAM

If the DTC control register information is stored in external RAM, **20 + 4 × SI + 4 × DI** must be added to the values in table 6-5.

### 3. Interrupt Controller Wait

The values given above do not include the time between the occurrence of the interrupt request and the starting of the DTC. This time includes two states for the interrupt controller to check priority and a variable wait until the end of the current CPU instruction. At maximum, this time equals the sum of the values indicated for items No. 1 and 2 in table 6-6.

If the data transfer count is 0 at the end of a data transfer cycle, the number of states from the end of the data transfer cycle until the first instruction of the user-coded interrupt-handling routine is executed is the value given for item No. 3 in table 6-6.

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

| No. | Reason for Wait | | Number of States<br>Minimum Mode | <br>Maximum Mode |
|---|---|---|---|---|
| 1 | Interrupt priority decision and comparison with mask level in CPU status register | | 2 states | |
| 2 | Maximum number of states to completion of current instruction | Instruction is in on-chip memory | x<br>(x = 38 for LDM instruction specifying all registers) | |
| | | Instruction is in external memory | y<br>(y = 74 + 16m for LDM instruction specifying all registers) | |
| 3 | Number of states from saving of PC and SR (or PC, SR, and CP) until first instruction of interrupt-handling routine is prefetched. | Stack is in on-chip RAM | 16 | 21 |
| | | Stack is in external memory | 28 + 6m | 41 + 10m |

Note: m: Number of wait states inserted in external memory access.

## 6.4 Procedure for Using the DTC

A program that uses the DTC to transfer data must do the following:

1. Set the appropriate DTMR, DTSR, DTDR, and DTCR register information in the memory location indicated in the DTC vector table.
2. Set the data transfer enable bit of the pertinent interrupt to 1, and set the priority of the interrupt source (in the interrupt priority register) and the interrupt mask level (in the CPU status register) so that the interrupt can be accepted.
3. Set the interrupt enable bit in the control register for the interrupt source. (For $IRQ_0$ to $IRQ_3$, the control register is the IRQ control register.)

Following these preparations, the DTC will be started each time the interrupt occurs. When the number of bytes or words designated by the DTCR value have been transferred, after transferring the last byte or word, the DTC generates a CPU interrupt.

The user-coded interrupt-handling routine must take action to prepare for or disable further DTC data transfer: by readjusting the data transfer count, for example, or clearing the data transfer enable bit. If no action is taken, the next interrupt of the same type will start the DTC with an initial data transfer count of 65,536.

T-49-19-16

## 6.5 Example

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

**Conditions:**

- Operating mode: Minimum mode
- Received data are to be stored in consecutive addresses starting at H'FE00.
- DTC control register information for the RXI interrupt is stored at addresses H'FD80 to H'FD87.
- Accordingly, the DTC vector table contains H'FD at address H'00AA and H'80 at address H'00AB.
- The desired interrupt mask level in the CPU status register is 4, and the desired SCI1 interrupt priority level is 5.

**Procedure**

1. The user program sets DTC control register information in addresses H'FD80 to H'FD87 as shown in table 6-7.

**Table 6-7 DTC Control Register Information Set in RAM**

| Register | Description | Value Set |
|---|---|---|
| DTMR | Byte transfer<br>Source address fixed<br>Increment destination address | H'2000 |
| DTSR | Address of SCI receive data register | H'FEDD |
| DTDR | Address H'FE00 | H'FE00 |
| DTCR | Number of bytes to be received: 128 | H'0080 |

2. The program sets the RXI (SCI Receive Interrupt) bit in the data transfer enable register (bit 1 of register DTEC) to 1.
3. The program sets the interrupt mask in the CPU status register to 4, and the SCI1 interrupt priority in bits 2 to 0 of interrupt priority register IPRC to 5.

4. The program sets SCI1 to the appropriate receive mode, and sets the receive interrupt enable (RIE) bit in the serial control register (SCR) to 1 to enable receive interrupts. T-49-19-16
5. Thereafter, each time the SCI1 receives one byte of data, it requests an RXI interrupt, which the interrupt controller directs toward the DTC. The DTC transfers the byte from SCI1's receive data register (RDR) into RAM, and clears the interrupt request before ending.
6. When 128 bytes have been transferred (DTCR = 0), the DTC generates a CPU interrupt. The interrupt source is SCI1. The interrupt type is RXI.
7. The user-coded RXI interrupt-handling routine processes the received data and disables further data transfer (by clearing the RIE bit, for example).

Figure 6-6 shows the DTC vector table and data in RAM for this example.

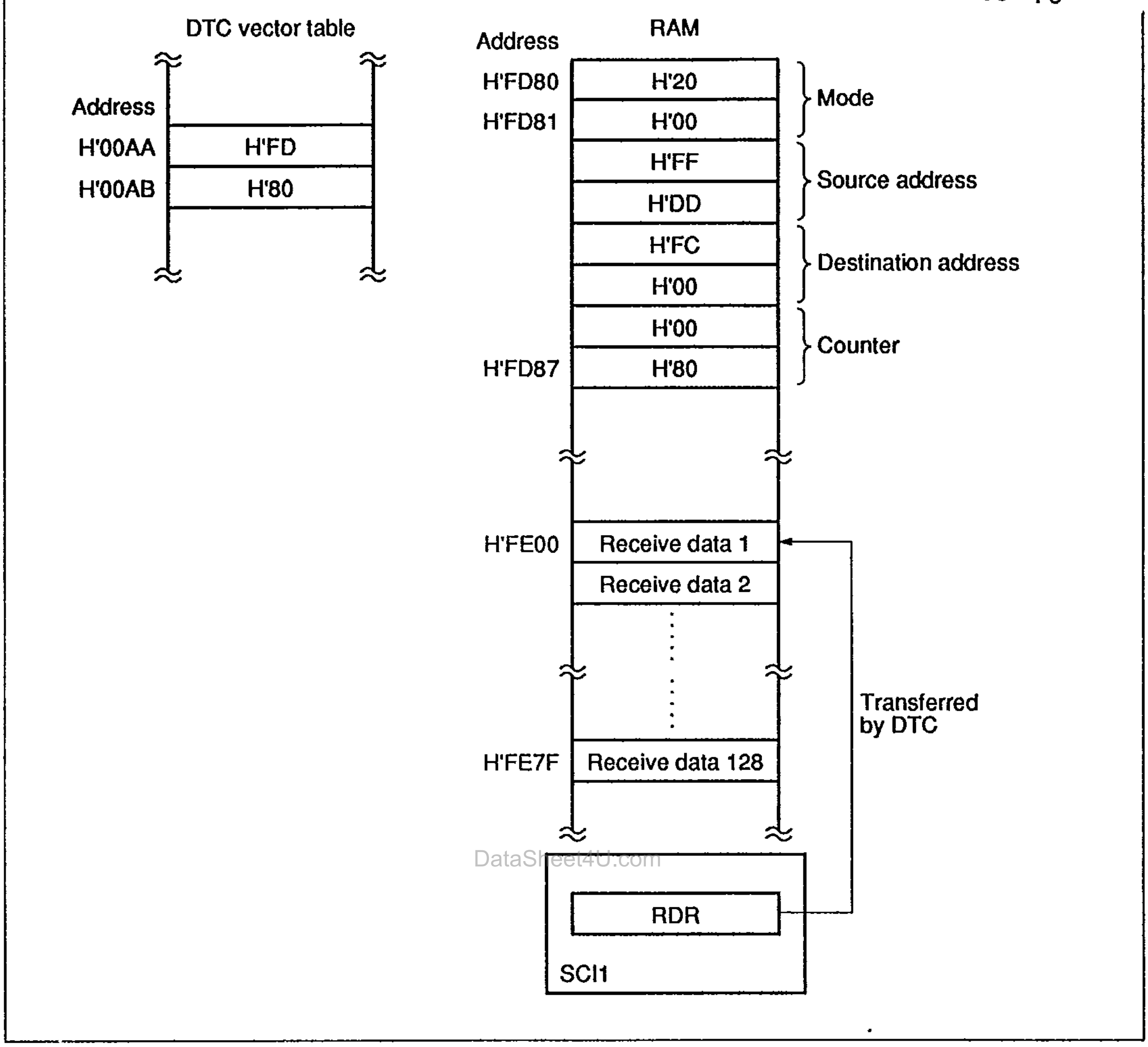


Figure 6-6 Use of DTC to Receive Data via Serial Communication Interface

# Section 7 Wait-State Controller

T-49-19-16

## 7.1 Overview

To simplify interfacing to low-speed external devices, the H8/520 has an on-chip wait-state controller (WSC) that can insert wait states (Tw) to prolong bus cycles.

The wait-state function can be used in CPU and DTC access cycles to external addresses. It is not used in access to on-chip memory or registers. The Tw states are inserted between the $T_2$ state and $T_3$ state in the bus cycle. The number of wait states can be selected by a value set in the wait-state control register (WCR), or by holding the $\overline{\text{WAIT}}$ pin low for the required interval.

### 7.1.1 Features

The main features of the wait-state controller are as follows:

- Selection of three operating modes
  Programmable wait mode, pin wait mode, or pin auto-wait mode
- 0, 1, 2, or 3 wait states can be inserted.
  And in the pin wait mode, 4 or more states can be inserted by holding the $\overline{\text{WAIT}}$ pin low.

### 7.1.2 Block Diagram

Figure 7-1 shows a block diagram of the wait-state controller.

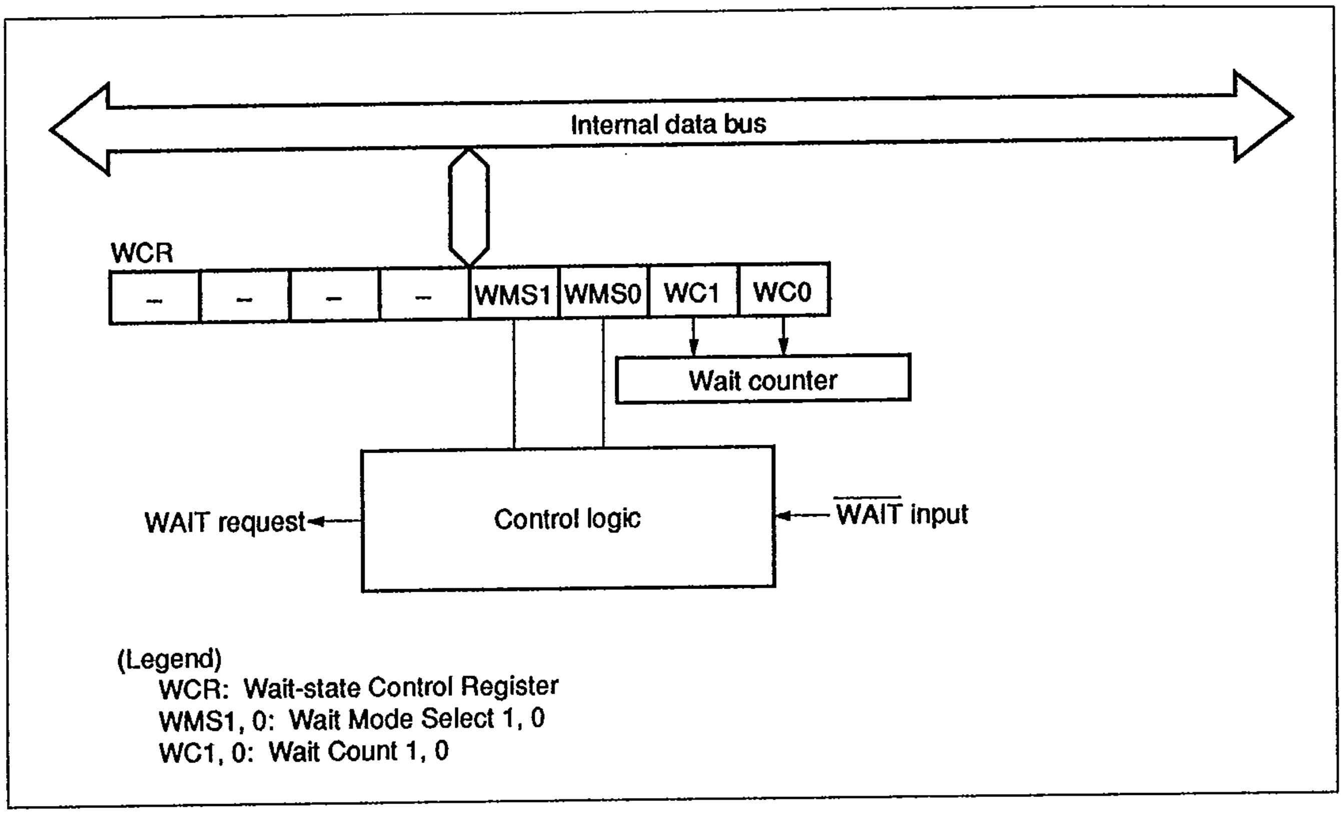


Figure 7-1 Block Diagram of Wait-State Controller

### 7.1.3 Register Configuration

The wait-state controller has one control register: the wait-state control register described in table 7-1.

Table 7-1 Register Configuration

| Name | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|
| Wait-state control register | WCR | R/W | H'F3 | H'FFF8 |

## 7.2 Wait-State Control Register

The wait-state control register (WCR) is an 8-bit register that specifies the wait mode and the number of wait states to be inserted. A reset initializes the WCR to specify the programmable wait mode with three wait states. The WCR is not initialized in the software standby mode.

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

**Bits 7 to 4—Reserved:** These bits cannot be modified and are always read as 1.

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

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

**Bits 1 and 0—Wait Count (WC1 and WC0):** These bits specify the number of wait states to be inserted.

Wait states are inserted only in bus cycles in which the CPU or DTC accesses an external address.

| Bit 1 WC1 | Bit 0 WC0 | Description | |
|---|---|---|---|
| 0 | 0 | No wait states are inserted, except in pin wait mode. | |
| 0 | 1 | 1 wait state is inserted. | |
| 1 | 0 | 2 wait states are inserted. | |
| 1 | 1 | 3 wait states are inserted. | (Initial value) |

## 7.3 Operation in Each Wait Mode

T-49-19-16

Table 7-2 summarizes the operation of the three wait modes.

**Table 7-2 Wait Modes**

| Mode | $\overline{\text{WAIT}}$ Pin Function | Insertion Conditions | Number of Wait States Inserted |
|---|---|---|---|
| Programmable wait mode<br>WMS1 = 0<br>WMS0 = 0 | Disabled | Inserted on access to an off-chip address | 0 to 3 wait states are inserted, as specified by bits WC0 and WC1. |
| Pin wait mode<br>WMS1 = 1<br>WMS0 = 0 | Enabled | Inserted on access to an off-chip address | 0 to 3 wait states are inserted, as specified by bits WC0 and WC1, plus additional wait states while the $\overline{\text{WAIT}}$ pin is held low. |
| Pin auto-wait mode<br>WMS1 = 1<br>WMS0 = 1 | Enabled | Inserted on access to an off-chip address if the $\overline{\text{WAIT}}$ pin is low | 0 to 3 wait states are inserted, as specified by bits WC0 and WC1. |

### 7.3.1 Programmable Wait Mode

The programmable wait mode is selected when WMS1 = 0 and WMS0 = 0.

Whenever the CPU or DTC accesses an off-chip address, the number of wait states set in bits WC1 and WC0 are inserted. The $\overline{\text{WAIT}}$ pin is not used for wait control; it is available as an I/O pin (P1$_0$).

Figure 7-2 shows the timing of the operation in this mode when the wait count is 1 (WC1 = 0, WC0 = 1).

T-49-19-16

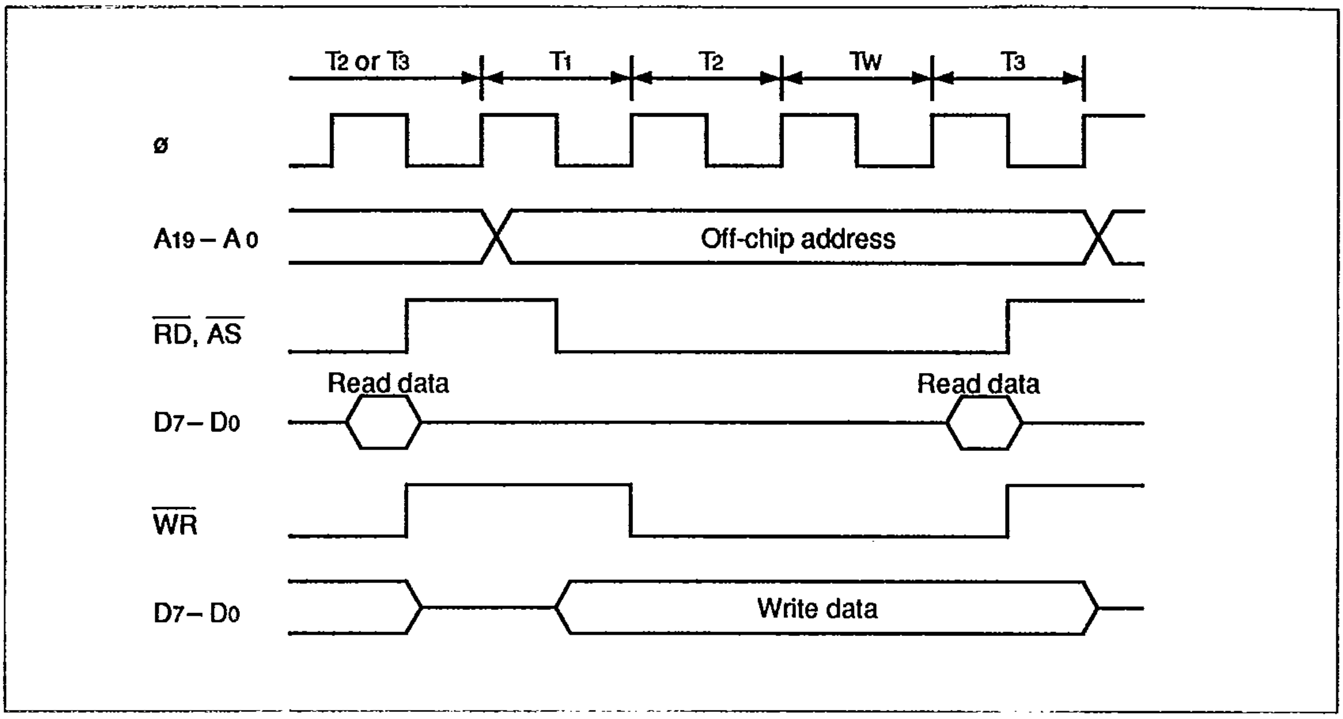


Figure 7-2 Programmable Wait Mode

### 7.3.2 Pin Wait Mode

The pin wait mode is selected when WMS1 = 1 and WMS0 = 0.

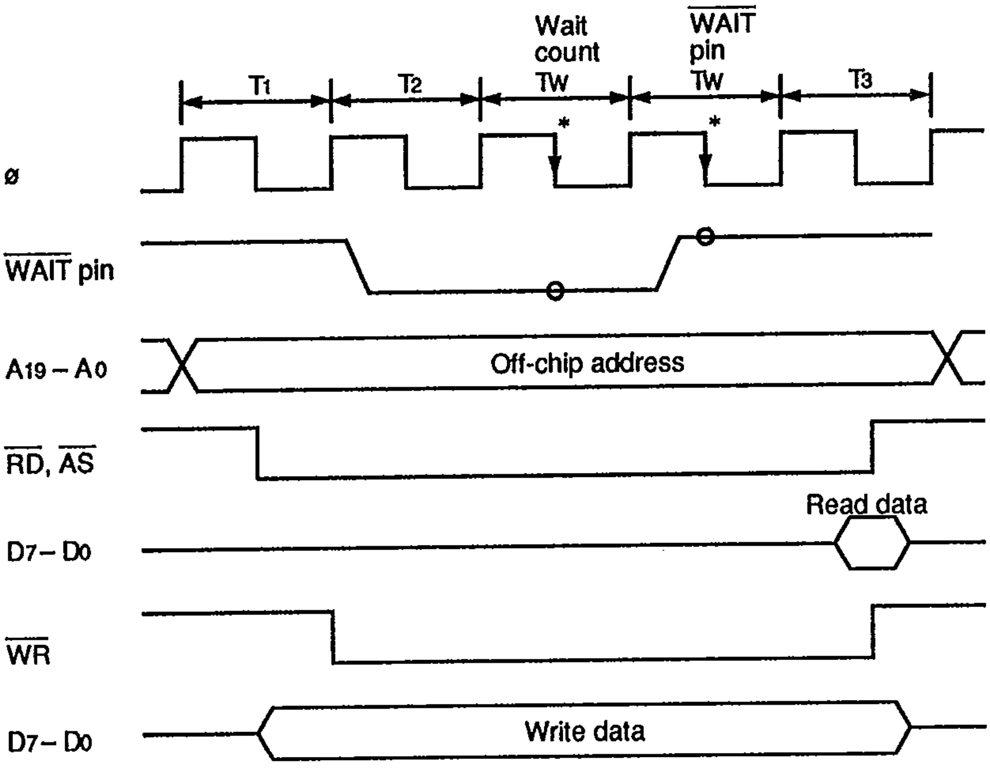
In this mode the $\overline{\text{WAIT}}$ function of the P1$_0$/$\overline{\text{WAIT}}$ pin is used automatically.

The number of wait states indicated by bits WC1 and WC0 are inserted into any bus cycle in which the CPU or DTC accesses an off-chip address. In addition, wait states continue to be inserted as long as the $\overline{\text{WAIT}}$ pin is held low. In particular, if the wait count is 0 but the $\overline{\text{WAIT}}$ pin is low at the rising edge of the ø clock in the T$_2$ state, wait states are inserted until the $\overline{\text{WAIT}}$ pin goes high.

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

Figure 7-3 shows the timing of the operation in this mode when the wait count is 1 (WC1 = 0, WC0 = 1) and the $\overline{\text{WAIT}}$ pin is held low to insert one additional wait state.

T-49-19-16



Note: * The arrowheads indicate the times at which the $\overline{\text{WAIT}}$ pin is sampled.

Figure 7-3 Pin Wait Mode

### 7.3.3 Pin Auto-Wait Mode

The pin auto-wait mode is selected when WMS1 = 1 and WMS0 = 1.

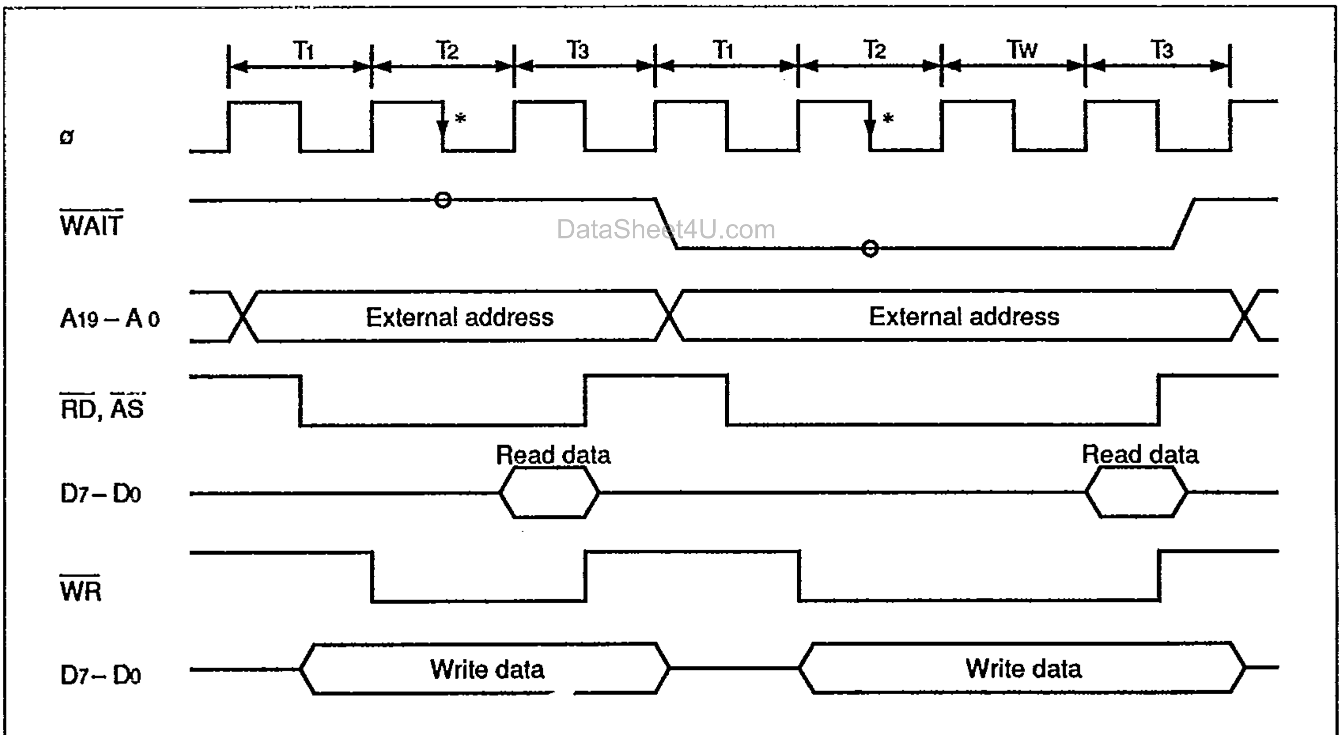
In this mode the $\overline{\mathrm{WAIT}}$ function of the P1₀/$\overline{\mathrm{WAIT}}$ pin is used automatically.

In this mode, the number of wait states indicated by bits WC1 and WC0 are inserted, but only if there is a low input at the $\overline{\mathrm{WAIT}}$ pin.

Figure 7-4 shows the timing of this operation when the wait count is 1.

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

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



Note: * The arrowheads indicate the times at which the $\overline{\mathrm{WAIT}}$ pin is sampled.

Figure 7-4 Pin Auto-Wait Mode

# Section 8 Clock Pulse Generator

## 8.1 Overview

The H8/520 chip has a built-in clock pulse generator (CPG) consisting of an oscillator circuit, a system (ø) clock divider, and a prescaler. The prescaler generates clock signals for the on-chip supporting modules.

### 8.1.1 Block Diagram

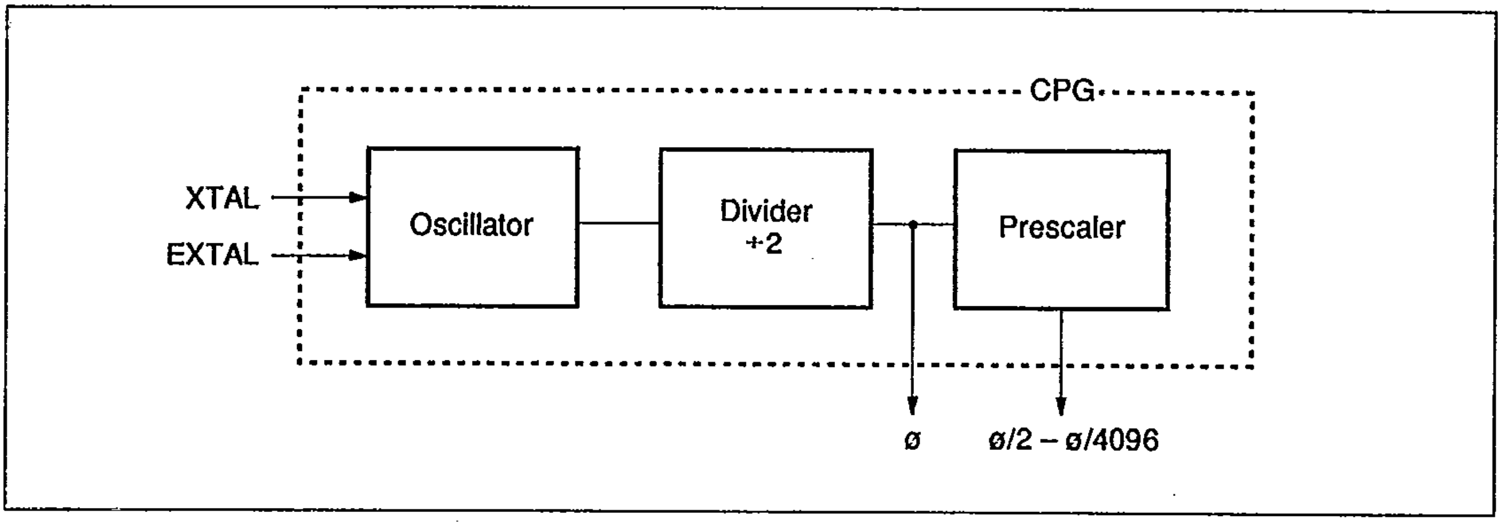


Figure 8-1 Block Diagram of Clock Pulse Generator

## 8.2 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 directly.

### 1. Connecting an External Crystal

**Circuit Configuration:** An external crystal can be connected as in the example in figure 8-2. An AT-cut parallel resonating crystal should be used.

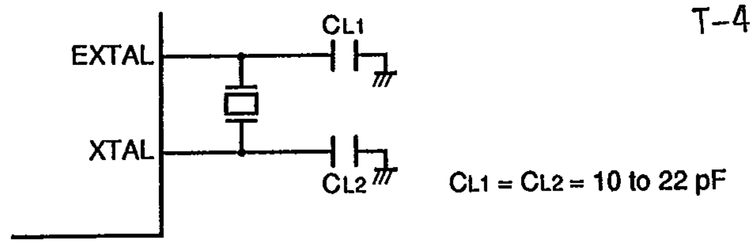


Figure 8-2 Connection of Crystal Oscillator (Example)

**Crystal Oscillator:** The external crystal should have the characteristics listed in table 8-1.

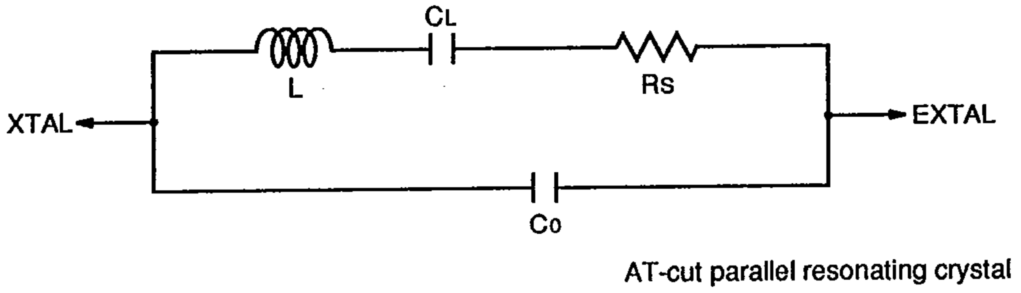


Figure 8-3 Crystal Oscillator Equivalent Circuit

**Table 8-1 External Crystal Parameters**

| Frequency (MHz) | 2 | 4 | 8 | 12 | 16 | 20 |
|---|---|---|---|---|---|---|
| Rs max (Ω) | 500 | 120 | 60 | 40 | 30 | 20 |
| Co (pF) | | | 7 pF max | | | |

**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 8-4.

When the board is designed, the crystal and its load capacitors should be placed as close as possible to the XTAL and EXTAL pins.

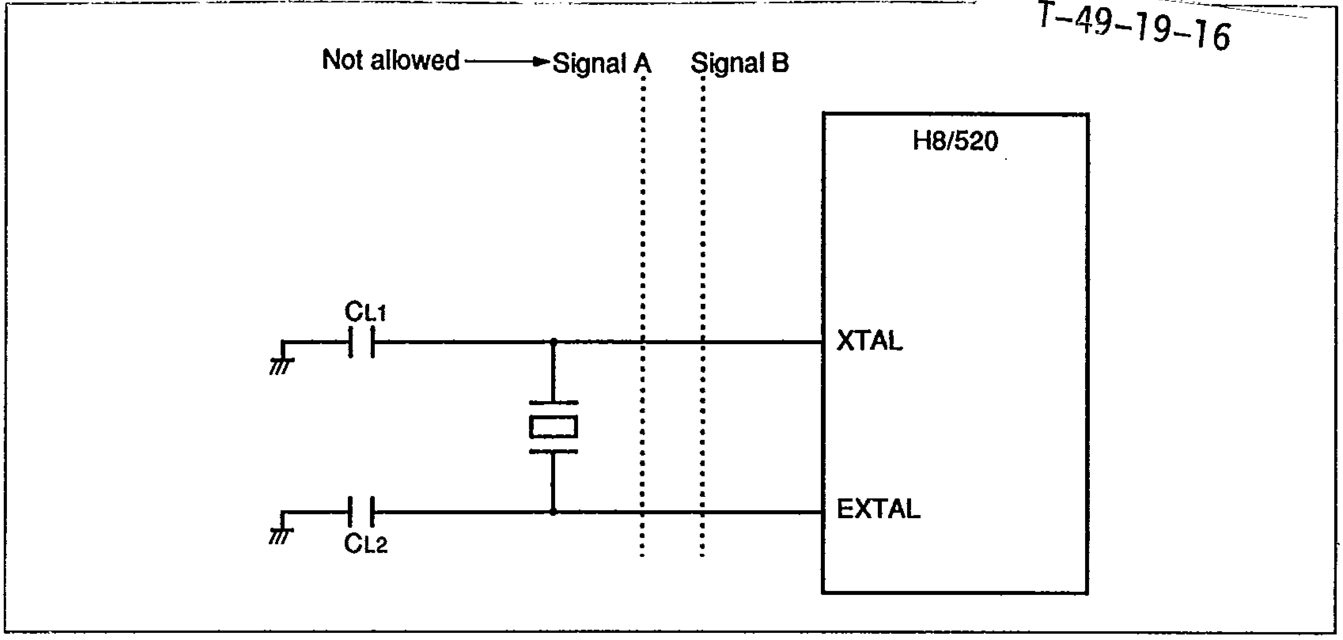


Figure 8-4 Notes on Board Design around External Crystal

## 2. Input of External Clock Signal

**Circuit Configuration:** An external clock signal can be input at the EXTAL pin as shown in the example in figure 8-5.

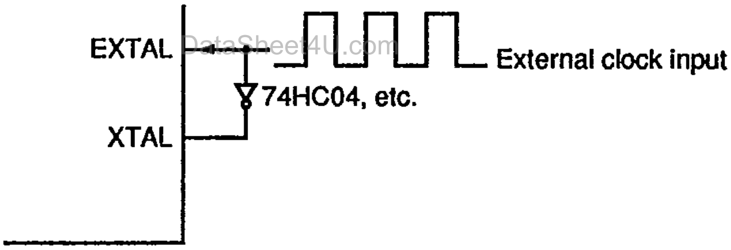


Figure 8-5 External Clock Input (Example)

Note: The masked ROM version can be driven by supplying an external clock signal to the EXTAL pin only, leaving the XTAL pin open. The PROM version can also be driven in this way, leaving the XTAL pin open, when the clock frequency is 16 MHz or less.

# Section 9 I/O Ports

## 9.1 Overview

T-49-19-16

The H8/520 has seven parallel I/O ports. Ports 1 to 5 are eight-bit input/output ports. Port 6 is a four-bit (or eight-bit*) input-only port. Port 7 is a six-bit input/output port. Table 9-1 summarizes the functions of each port.

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 page 0 of the address space. Each port (except port 6) also has a data direction register (DDR) which determines which pins are used for input and which for output.

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.

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.

As table 9-1 indicates, all of the I/O port pins have dual functions. For example, pin 0 of port 1 can be used either as a general-purpose I/O pin ($P1_0$), or for input of the $\overline{\text{WAIT}}$ signal. The function of a pin is determined by the MCU operating mode, or by a value set in a control register.

Outputs from ports 1 to 4 can drive one TTL load and a 90-pF capacitive load. Outputs from ports 5 and 7 can drive one TTL load and a 30-pF capacitive load.

Outputs from ports 1 to 5 and 7 can also drive a Darlington transistor pair. Outputs from port 3 can drive a light-emitting diode (with 10-mA current sink). Ports 3 and 4 have built-in MOS pull-ups for each input. Port 5 has Schmitt inputs.

Schematic diagrams of the I/O port circuits are shown in appendix C.

Note: * CP-68 package only

T-49-19-16

### Table 9-1 I/O Port Summary

| | | | Expanded Modes | | | | Single-Chip Mode |
|---|---|---|---|---|---|---|---|
| **Port** | **Description** | **Pins** | **Mode 1** | **Mode 2** | **Mode 3** | **Mode 4** | **Mode 7** |
| Port 1 | 8-bit input/output port | $P1_7/\overline{WR}$<br>$P1_6/\overline{RD}$<br>$P1_5/\overline{AS}$ | $\overline{AS}$, $\overline{RD}$, and $\overline{WR}$ output | | | | $P1_7$, $P1_6$, and $P1_5$ input/output |
| | | $P1_4/A_{16}/\overline{IRQ_3}$<br>$P1_3/A_{17}/\overline{IRQ_2}$ | $\overline{IRQ_3}$ and $\overline{IRQ_2}$ input and $P1_4$ and $P1_3$ input/output | | Page address output ($A_{16}$, $A_{17}$) | $\overline{IRQ_3}$ and $\overline{IRQ_2}$ input, page address ($A_{16}$, $A_{17}$) output, and $P1_4$ and $P1_3$ input | $\overline{IRQ_3}$ and $\overline{IRQ_2}$ input and $P1_4$ and $P1_3$ input/output |
| | | $P1_2/A_{18}/\overline{IRQ_1}/\overline{ADTRG}$ | $\overline{IRQ_1}$ input, $\overline{ADTRG}$ input, and $P1_2$ input/output | | Page address output ($A_{18}$) | $\overline{IRQ_1}$ input, page address ($A_{18}$) output, $\overline{ADTRG}$ input, and $P1_2$ input | $\overline{IRQ_1}$ input, $\overline{ADTRG}$ input, and $P1_2$ input/output |
| | | $P1_1/\overline{IRQ_0}$ | $\overline{IRQ_0}$ input and $P1_1$ input/output | | | | |
| | | $P1_0/\overline{WAIT}$ | $\overline{WAIT}$ input and $P1_0$ input/output | | | | $P1_0$ input/output |
| Port 2 | 8-bit input/output port | $P2_7 - P2_0/$<br>$D_7 - D_0$ | Data bus ($D_7$ to $D_0$) | | | | $P2_7$ to $P2_0$ input/output |
| Port 3 | 8-bit input/output port (Built-in MOS input pull-up. Can drive LEDs.) | $P3_7 - P3_0/$<br>$A_7 - A_0$ | Low address bus output ($A_7 - A_0$) | Low address bus output ($A_7 - A_0$) | Low address bus output ($A_7 - A_0$) | Low address bus output ($A_7 - A_0$) | $P3_7$ to $P3_0$ input/output |

T-49-19-16

**Table 9-1 I/O Port Summary (cont)**

| | | | Expanded Modes | | | | Single-Chip Mode |
|---|---|---|---|---|---|---|---|
| **Port** | **Description** | **Pins** | **Mode 1** | **Mode 2** | **Mode 3** | **Mode 4** | **Mode 7** |
| Port 4 | 8-bit input/output port (Built-in MOS input pull-up) | $P4_7/A_{15}/\overline{IRQ_7}$<br>$P4_6/A_{14}/\overline{IRQ_6}$<br>$P4_5/A_{13}/\overline{IRQ_5}$<br>$A4_4/A_{12}/\overline{IRQ_4}$ | High address bus output ($A_{15}$ to $A_8$) | $\overline{IRQ_7}$ to $\overline{IRQ_4}$ input, high address bus output ($A_{15}$ to $A_{12}$), and $P4_7$ to $P4_4$ input | High address bus output ($A_{15}$ to $A_8$) | $\overline{IRQ_7}$ to $\overline{IRQ_4}$ input, high address bus output ($A_{15}$ to $A_{12}$), and $P4_7$ to $P4_4$ input | $\overline{IRQ_7}$ to $\overline{IRQ_4}$ input and $P4_7$ to $P4_4$ input/output |
| | | $P4_3/A_{11}$<br>$P4_2/A_{10}$<br>$P4_1/A_9$<br>$P4_0/A_8$ | | High address bus output ($A_{11}$ to $A_8$), and $P4_3$ to $P4_0$ input | | High address bus output ($A_{11}$ to $A_8$), and $P4_3$ to $P4_0$ input | $P4_3$ to $P4_0$ input/output |
| Port 5 | 8-bit input/output port (Schmit trigger input) | $P5_7/FTOA_2/ø$<br>$P5_6/FTOA_1$<br>$P5_5/FTOB_2/FTCI_2$<br>$P5_4/FTOB_1/FTCI_1$<br>$P5_3/TMO$<br>$P5_2/FTI_2/TMRI$<br>$P5_1/FTI_1$<br>$P5_0/TMCI$ | General-purpose input/output pins ($P5_7$ to $P5_0$) also used for input and output by the 16-bit free-running timer module ($FTOA_2$, $FTOA_1$, $FTOB_1$, $FTCI_2$, $FTCI_1$, $FTI_2$, $FTI_1$) and 8-bit timer module (TMO, TMRI, TMCI), and for output of the system clock (ø). | | | | |

T-49-19-16

## Table 9-1 I/O Port Summary (cont)

| Port | Description | Pins | Expanded Modes |  |  |  | Single-Chip Mode |
|---|---|---|---|---|---|---|---|
|  |  |  | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 7 |
| Port 6 | 4-bit input port<br>(8-bit input port) | $P6_3 - P6_0$/<br>$AN_3 - AN_0$ | General-purpose input ($P6_3$ to $P6_0$) and analog input ($AN_0$ to $AN_3$) |  |  |  |  |
|  |  | ($P6_7 - P6_0$/<br>$AN_7 - AN_0$)* | [General-purpose input ($P6_7$ to $P6_0$) and analog input ($AN_7$ to $AN_0$)]* |  |  |  |  |
| Port 7 | 6-bit input/output port | $P7_5$/$SCK_1$<br>$P7_4$/$RXD_1$<br>$P7_3$/$TXD_1$ | General-purpose input/output pins ($P7_5$ to $P7_3$), also used for input and output by serial communication interface channel 1 ($SCK_1$, $RXD_1$, $TXD_1$) |  |  |  |  |
|  |  | $P7_2$/$A_{19}$/<br>$SCK_2$ | $SCK_2$ (serial communication interface channel 2 clock) input/output and $P7_2$ input/output |  | Page address output ($A_{19}$) | $SCK_2$ input/output, page address output ($A_{19}$), or $P7_2$ input/output | $SCK_2$ input/output or $P7_2$ input/output |
|  |  | $P7_1$/$RXD_2$<br>$P7_0$/$TXD_2$ | General-purpose input/output pins ($P7_1$, $P7_0$), also used for input and output by serial communication interface channel 2 ($RXD_2$, $TXD_2$) |  |  |  |  |

Note: * CP-68 package only.

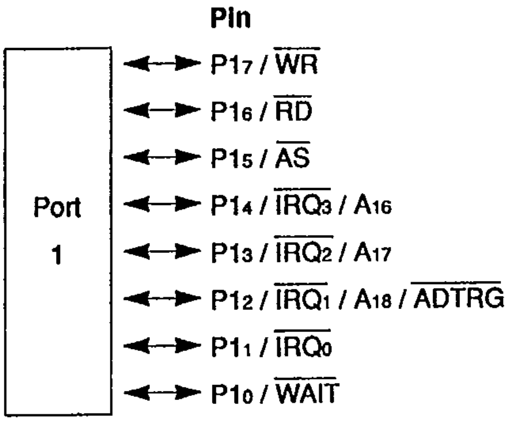
## 9.2 Port 1

### 9.2.1 Overview

Port 1 is an 8-bit input/output port with the pin configuration shown in figure 9-1. The pin functions depend on the MCU operating mode. Some pins can perform two or three functions simultaneously.

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

T-49-19-16



**Modes 1 and 2**

$\overline{WR}$ (output)
$\overline{RD}$ (output)
$\overline{AS}$ (output)
$P1_4$ (input/output) / $\overline{IRQ_3}$ (input)
$P1_3$ (input/output) / $\overline{IRQ_2}$ (input)
$P1_2$ (input/output) / $\overline{IRQ_1}$ (input) / $\overline{ADTRG}$ (input)
$P1_1$ (input/output) / $\overline{IRQ_0}$ (input)
$P1_0$ (input/output) / $\overline{WAIT}$ (input)

**Mode 3**

$\overline{WR}$ (output)
$\overline{RD}$ (output)
$\overline{AS}$ (output)
$A_{16}$ (output)
$A_{17}$ (output)
$A_{18}$ (output)
$P1_1$ (input/output) / $\overline{IRQ_0}$ (input)
$P1_0$ (input/output) / $\overline{WAIT}$ (input)

**Mode 4**

$\overline{WR}$ (output)
$\overline{RD}$ (output)
$\overline{AS}$ (output)
$P1_4$ (input) / $\overline{IRQ_3}$ (input) / $A_{16}$ (output)
$P1_3$ (input) / $\overline{IRQ_2}$ (input) / $A_{17}$ (output)
$P1_2$ (input) / $\overline{IRQ_1}$ (input) / $A_{18}$ (output) / $\overline{ADTRG}$ (input)
$P1_1$ (input/output) / $\overline{IRQ_0}$ (input)
$P1_0$ (input/output) / $\overline{WAIT}$ (input)

**Single-Chip Mode**

$P1_7$ (input/output)
$P1_6$ (input/output)
$P1_5$ (input/output)
$P1_4$ (input/output) / $\overline{IRQ_3}$ (input)
$P1_3$ (input/output) / $\overline{IRQ_2}$ (input)
$P1_2$ (input/output) / $\overline{IRQ_1}$ (input) / $\overline{ADTRG}$ (input)
$P1_1$ (input/output) / $\overline{IRQ_0}$ (input)
$P1_0$ (input/output)

**Figure 9-1 Pin Functions of Port 1**

### 9.2.2 Port 1 Registers

T-49-19-16

Table 9-2 lists the registers of port 1.

**Table 9-2 Port 1 Registers**

| Name | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|
| Port 1 data direction register | P1DDR | W | H'00* | H'FF80 |
| Port 1 data register | P1DR | R/W | H'00 | H'FF82 |

Note: * In single-chip mode.

### 1. Port 1 Data Direction Register (P1DDR)—H'FF80

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 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 |
| Initial value* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

Note: * In single-chip mode

P1DDR is an 8-bit register that selects the direction of each pin in port 1. Details are given for each MCU operating mode below.

**Modes 1 and 2 (Expanded Minimum Modes):** Bits 7 to 5 of P1DDR are fixed at 1 and cannot be written. Pins P1$_7$ to P1$_5$ are used for output of bus control signals.

When bits 4 to 0 of P1DDR are set to 1, the corresponding pin of port 1 functions as an output pin. When these bits are cleared to 0, the corresponding pin functions as an input pin.

**Mode 3 (Expanded Maximum Mode with On-Chip ROM Disabled):** Bits 7 to 2 of P1DDR are fixed at 1 and cannot be written. Pins P1$_7$ to P1$_5$ are used for output of bus control signals. Pins P1$_4$ to P1$_2$ are used for page address output.

When bits 1 and 0 of P1DDR are set to 1, the corresponding pin of port 1 functions as an output pin. When these bits are cleared to 0, the corresponding pin functions as an input pin.

**Mode 4 (Expanded Maximum Mode with On-Chip ROM Enabled):** Bits 7 to 5 of P1DDR are fixed at 1 and cannot be written. Pins P1$_7$ to P1$_5$ are used for output of bus control signals.

When bits 4 to 2 of P1DDR are set to 1, pins P14 to P12 are used for page address output. When these bits are cleared to 0, the corresponding pin becomes available for general-purpose input.

T-49-19-16

When bits 1 and 0 of P1DDR are set to 1, pins P11 and P10 function as output pins. When these bits are cleared to 0, the corresponding pin functions as an input pin.

**Mode 7 (Single-Chip Mode):** A pin functions as an output pin if the corresponding bit in P1DDR is set to 1, and as in input pin if the bit is cleared to 0.

P1DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P1DDR is initialized to H'00 by a reset and in the hardware standby mode. P1DDR is not initialized in the software standby mode, so if a P1DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 1 data register.

### 2. Port 1 Data Register (P1DR)—H'FF82

| 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 P17 to P10. When port 1 is read, output pins return the value in the P1DR latch, regardless of the actual level at the pin. Input pins return the level at the pin, not the value in the P1DR latch.

If any of the port 1 data direction bits are cleared to 0, selecting input, use only data transfer (MOV) instructions to write data in P1DR. Do not use arithmetic, logic, or bit manipulation instructions. These instructions read the input pins and may write unintended data in P1DR.

### 9.2.3 Pin Functions in Each Mode

The functions of port 1 depend on the MCU operating mode. Table 9-3 shows the pin functions in modes 1 and 2. Table 9-4 shows the pin functions in mode 3. Table 9-5 shows the pin functions in mode 4. Table 9-6 shows the pin functions in the single-chip mode.

## Table 9-3 Port 1 Pin Functions in Modes 1 and 2

T-49-19-16

| Pin | Functions |
|---|---|
| $\overline{WR}$ | Output of $\overline{WR}$ signal. |
| $\overline{RD}$ | Output of $\overline{RD}$ signal. |
| $\overline{AS}$ | Output of $\overline{AS}$ signal. |
| $P1_4$ / $\overline{IRQ_3}$ | The function depends on the IRQ3E bit in the IRQ control register (IRQCR) and the $P1_4$DDR bit as follows: |

| IRQ3E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_4$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_4$ input | $P1_4$ output | $\overline{IRQ_3}$ input<br>and $P1_4$ input | $\overline{IRQ_3}$ input<br>and $P1_4$ output |

| Pin | Functions |
|---|---|
| $P1_3$ / $\overline{IRQ_2}$ | The function depends on the IRQ2E bit and the $P1_3$DDR bit as follows: |

| IRQ2E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_3$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_3$ input | $P1_3$ output | $\overline{IRQ_2}$ input<br>and $P1_3$ input | $\overline{IRQ_2}$ input<br>and $P1_3$ output |

### Table 9-3 Port 1 Pin Functions in Modes 1 and 2 (cont)

| Pin | Functions |
|---|---|
| $P1_2$ / $\overline{IRQ_1}$ / $\overline{ADTRG}$ | The function depends on the IRQ1E bit, the $P1_2$DDR bit, and the trigger enable bit (TRGE) in the A/D control register (ADCR) as follows: |

| TRGE | 0 | | | |
|---|---|---|---|---|
| IRQ1E | 0 | | 1 | |
| $P1_2$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_2$ input | $P1_2$ output | $\overline{IRQ_1}$ input and $P1_2$ input | $\overline{IRQ_1}$ input and $P1_2$ output |

| TRGE | 1 | | | |
|---|---|---|---|---|
| IRQ1E | 0 | | 1 | |
| $P1_2$DDR | 0 | 1 | 0 | 1 |
| Pin function | $\overline{ADTRG}$ input and $P1_2$ input | $\overline{ADTRG}$ input and $P1_2$ output | $\overline{ADTRG}$ input, $\overline{IRQ_1}$ input, and $P1_2$ input | $\overline{ADTRG}$ input, $\overline{IRQ_1}$ input, and $P1_2$ output |

| Pin | Functions |
|---|---|
| $P1_1$ / $\overline{IRQ_0}$ | The function depends on the IRQ0E bit and the $P1_1$DDR bit as follows: |

| IRQ0E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_1$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_1$ input | $P1_1$ output | $\overline{IRQ_0}$ input and $P1_1$ input | $\overline{IRQ_0}$ input and $P1_1$ output |

| Pin | Functions |
|---|---|
| $P1_0$ / $\overline{WAIT}$ | The function depends on the wait mode select 1 bit (WMS1) of the wait-state control register (WCR) and the $P1_0$DDR bit as follows: |

| WMS1 | 0 | | 1 | |
|---|---|---|---|---|
| $P1_0$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_0$ input | $P1_0$ output | $\overline{WAIT}$ input | |

T-49-19-16

### Table 9-4 Port 1 Pin Functions in Mode 3

| Pin | Functions |
|---|---|
| $\overline{\text{WR}}$ | Output of $\overline{\text{WR}}$ signal. |
| $\overline{\text{RD}}$ | Output of $\overline{\text{RD}}$ signal. |
| $\overline{\text{AS}}$ | Output of $\overline{\text{AS}}$ signal. |
| $A_{16}$ | $A_{16}$ output |
| $A_{17}$ | $A_{17}$ output |
| $A_{18}$ | $A_{18}$ output |
| $P1_1$ / $\overline{\text{IRQ}_0}$ | The function depends on the IRQ0E bit and the $P1_1$DDR bit as follows: |

| IRQ0E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_1$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_1$ input | $P1_1$ output | $\overline{\text{IRQ}_0}$ input and $P1_1$ input | $\overline{\text{IRQ}_0}$ input and $P1_1$ output |

| Pin | Functions |
|---|---|
| $P1_0$ / $\overline{\text{WAIT}}$ | The function depends on the wait mode select 1 bit (WMS1) of the wait-state control register (WCR) and the $P1_1$DDR bit as follows: |

| WMS1 | 0 | | 1 | |
|---|---|---|---|---|
| $P1_0$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_0$ input | $P1_0$ output | $\overline{\text{WAIT}}$ input | |

## Table 9-5 Port 1 Pin Functions in Mode 4

<table>
<tr><th>Pin</th><th>Functions</th></tr>
<tr><td>$\overline{WR}$</td><td>Output of $\overline{WR}$ signal.</td></tr>
<tr><td>$\overline{RD}$</td><td>Output of $\overline{RD}$ signal.</td></tr>
<tr><td>$\overline{AS}$</td><td>Output of $\overline{AS}$ signal.</td></tr>
<tr><td>$P1_4$ / $\overline{IRQ_3}$ / $A_{16}$</td><td>The function depends on the IRQ3E bit in the IRQ control register (IRQCR) and the $P1_4$DDR bit as follows:
<table>
<tr><td>IRQ3E</td><td colspan="2">0</td><td colspan="2">1</td></tr>
<tr><td>$P1_4$DDR</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Pin function</td><td>$P1_4$ input</td><td>$A_{16}$ output</td><td>$\overline{IRQ_3}$ input and $P1_4$ input</td><td>$A_{16}$ output</td></tr>
</table></td></tr>
<tr><td>$P1_3$ / $\overline{IRQ_2}$ / $A_{17}$</td><td>The function depends on the IRQ2E bit and the $P1_3$DDR bit as follows:
<table>
<tr><td>IRQ2E</td><td colspan="2">0</td><td colspan="2">1</td></tr>
<tr><td>$P1_3$DDR</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Pin function</td><td>$P1_3$ input</td><td>$A_{17}$ output</td><td>$\overline{IRQ_2}$ input and $P1_3$ input</td><td>$A_{17}$ output</td></tr>
</table></td></tr>
<tr><td>$P1_2$ / $\overline{IRQ_1}$ / $A_{18}$ / $\overline{ADTRG}$</td><td>The function depends on the IRQ1E bit, the $P1_2$DDR bit, and the trigger enable bit (TRGE) in the A/D control register (ADCR) as follows:
<table>
<tr><td>TRGE</td><td colspan="4">0</td></tr>
<tr><td>IRQ1E</td><td colspan="2">0</td><td colspan="2">1</td></tr>
<tr><td>$P1_2$DDR</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Pin function</td><td>$P1_2$ input</td><td>$A_{18}$ output</td><td>$\overline{IRQ_1}$ input and $P1_2$ input</td><td>$A_{18}$ output</td></tr>
</table>
<table>
<tr><td>TRGE</td><td colspan="4">1</td></tr>
<tr><td>IRQ1E</td><td colspan="2">0</td><td colspan="2">1</td></tr>
<tr><td>$P1_2$DDR</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Pin function</td><td>$\overline{ADTRG}$ input and $P1_2$ input</td><td>$A_{18}$ output</td><td>$\overline{ADTRG}$ input,<br>$\overline{IRQ_1}$ input,<br>and $P1_2$ input</td><td>$A_{18}$ output</td></tr>
</table></td></tr>
</table>

### Table 9-5 Port 1 Pin Functions in Mode 4 (cont)

T-49-19-16

| Pin | Functions |
|---|---|
| $P1_1$ / $\overline{IRQ_0}$ | The function depends on the IRQ0E bit and the $P1_1$DDR bit as follows: |

| IRQ0E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_1$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_1$ input | $P1_1$ output | $\overline{IRQ_0}$ input<br>and $P1_1$ input | $\overline{IRQ_0}$ input<br>and $P1_1$ output |

| Pin | Functions |
|---|---|
| $P1_0$ / $\overline{\text{WAIT}}$ | The function depends on the wait mode select 1 bit (WMS1) of the wait-state control register (WCR) and the $P1_0$DDR bit as follows: |

| WMS1 | 0 | | 1 | |
|---|---|---|---|---|
| $P1_0$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_0$ input | $P1_0$ output | $\overline{\text{WAIT}}$ input | |

## Table 9-6 Port 1 Pin Functions in Single-Chip Mode

T-49-19-16

**Pin** **Functions**

$P1_7$

| $P1_7DDR$ | 0 | 1 |
|---|---|---|
| Pin function | $P1_7$ input | $P1_7$ output |

$P1_6$

| $P1_6DDR$ | 0 | 1 |
|---|---|---|
| Pin function | $P1_6$ input | $P1_6$ output |

$P1_5$

| $P1_5DDR$ | 0 | 1 |
|---|---|---|
| Pin function | $P1_5$ input | $P1_5$ output |

$P1_4$ / $\overline{IRQ_3}$ The function depends on the IRQ3E bit in the IRQ control register (IRQCR) and the $P1_4DDR$ bit as follows:

| IRQ3E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_4DDR$ | 0 | 1 | 0 | 1 |
| Pin function | $P1_4$ input | $P1_4$ output | $\overline{IRQ_3}$ input and $P1_4$ input | $\overline{IRQ_3}$ input and $P1_4$ output |

$P1_3$ / $\overline{IRQ_2}$ The function depends on the IRQ2E bit and the $P1_3DDR$ bit as follows:

| IRQ2E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_3DDR$ | 0 | 1 | 0 | 1 |
| Pin function | $P1_3$ input | $P1_3$ output | $\overline{IRQ_2}$ input and $P1_3$ input | $\overline{IRQ_2}$ input and $P1_3$ output |

## Table 9-6 Port 1 Pin Functions in Single-Chip Mode (cont)

T-49-19-16

**Pin** **Functions**

$P1_2$ / $\overline{\text{IRQ}_1}$ / $\overline{\text{ADTRG}}$

The function depends on the IRQ1E bit, the $P1_2$DDR bit, and the trigger enable bit (TRGE) in the A/D control register (ADCR) as follows:

| TRGE | 0 | | | |
|---|---|---|---|---|
| IRQ1E | 0 | | 1 | |
| $P1_2$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_2$ input | $P1_2$ output | $\overline{\text{IRQ}_1}$ input and $P1_2$ input | $\overline{\text{IRQ}_1}$ input and $P1_2$ output |

| TRGE | 1 | | | |
|---|---|---|---|---|
| IRQ1E | 0 | | 1 | |
| $P1_2$DDR | 0 | 1 | 0 | 1 |
| Pin function | $\overline{\text{ADTRG}}$ input and $P1_2$ input | $\overline{\text{ADTRG}}$ input and $P1_2$ output | $\overline{\text{ADTRG}}$ input, $\overline{\text{IRQ}_1}$ input, and $P1_2$ input | $\overline{\text{ADTRG}}$ input, $\overline{\text{IRQ}_1}$ input, and $P1_2$ output |

$P1_1$ / $\overline{\text{IRQ}_0}$

The function depends on the $\text{IRQ}_0$E bit and the $P1_1$DDR bit as follows:

| IRQ0E | 0 | | 1 | |
|---|---|---|---|---|
| $P1_1$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P1_1$ input | $P1_1$ output | $\overline{\text{IRQ}_0}$ input and $P1_1$ input | $\overline{\text{IRQ}_0}$ input and $P1_1$ output |

$P1_0$

| $P1_0$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P1_0$ input | $P1_0$ output |

## 9.3 Port 2

T-49-19-16

### 9.3.1 Overview

Port 2 is an 8-bit input/output port with the pin configuration shown in figure 9-2. In the expanded modes it operates as the external data bus ($D_7$ – $D_0$). In the single-chip mode it operates as a general-purpose input/output port.

Outputs from port 2 can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair.

| | Pin | Expanded Modes | Single-Chip Mode |
|---|---|---|---|
| Port 2 | $P2_7$ / $D_7$ | $D_7$ (input/output) | $P2_7$ (input/output) |
| | $P2_6$ / $D_6$ | $D_6$ (input/output) | $P2_6$ (input/output) |
| | $P2_5$ / $D_5$ | $D_5$ (input/output) | $P2_5$ (input/output) |
| | $P2_4$ / $D_4$ | $D_4$ (input/output) | $P2_4$ (input/output) |
| | $P2_3$ / $D_3$ | $D_3$ (input/output) | $P2_3$ (input/output) |
| | $P2_2$ / $D_2$ | $D_2$ (input/output) | $P2_2$ (input/output) |
| | $P2_1$ / $D_1$ | $D_1$ (input/output) | $P2_1$ (input/output) |
| | $P2_0$ / $D_0$ | $D_0$ (input/output) | $P2_0$ (input/output) |

Figure 9-2 Pin Functions of Port 2

### 9.3.2 Port 2 Registers

Table 9-7 lists the registers of port 2.

**Table 9-7 Port 2 Registers**

| Name | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|
| Port 2 data direction register | P2DDR | W | H'00 | H'FF81 |
| Port 2 data register | P2DR | R/W | H'00 | H'FF83 |

#### 1. Port 2 Data Direction Register (P2DDR)—H'FF81

| 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 |
| 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.

T-49-19-16

**Expanded Modes:** P2DDR is not used.

**Single-Chip Mode:** A pin functions as an output pin if the corresponding bit in P2DDR is set to 1, and as in input pin if the bit is cleared to 0.

P2DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

At a reset and in the hardware standby mode, P2DDR is initialized to H'00, making all eight pins input pins. P2DDR is not initialized in the software standby mode, so if a P2DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 2 data register.

### 2. Port 2 Data Register (P2DR)—H'FF83

| 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 $P2_0$.

At a reset and in the hardware standby mode, P2DR is initialized to H'00.

When port 2 is read, output pins return the value in the P2DR latch, regardless of the actual level at the pin. Input pins return the level at the pin, not the value in the P2DR latch.

If any of the port 2 data direction bits are cleared to 0, selecting input, use only data transfer (MOV) instructions to write data in P2DR. Do not use arithmetic, logic, or bit manipulation instructions. These instructions read the input pins and may write unintended data in P2DR.

### 9.3.3 Pin Functions in Each Mode

Port 2 has different functions in the expanded modes (modes 1, 2, 3, 4) and the single-chip mode (mode 7). Separate descriptions are given below.

**Pin Functions in Expanded Modes:** In the expanded modes (modes 1, 2, 3, and 4), port 2 is automatically used as the data bus and P2DDR is ignored. Figure 9-3 shows the pin functions for the expanded modes.

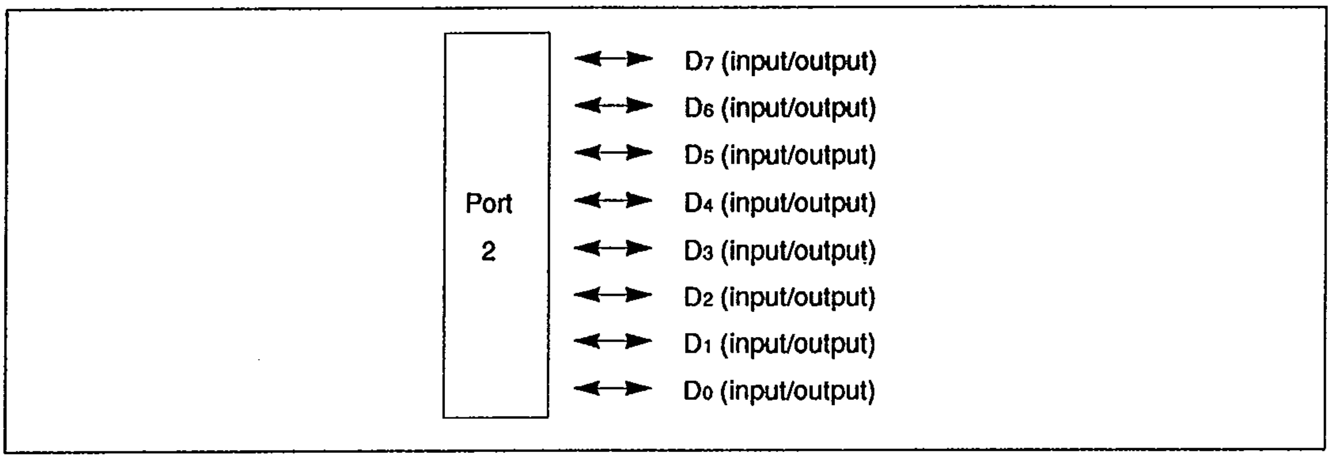


**Figure 9-3 Port 2 Pin Functions in Expanded Modes**

**Pin Functions in Single-Chip Mode:** In the single-chip mode (mode 7), each of the port 2 pins can be designated as an input pin or an output pin, as indicated in figure 9-4, by setting the corresponding bit in P2DDR to 1 for output or clearing it to 0 for input.

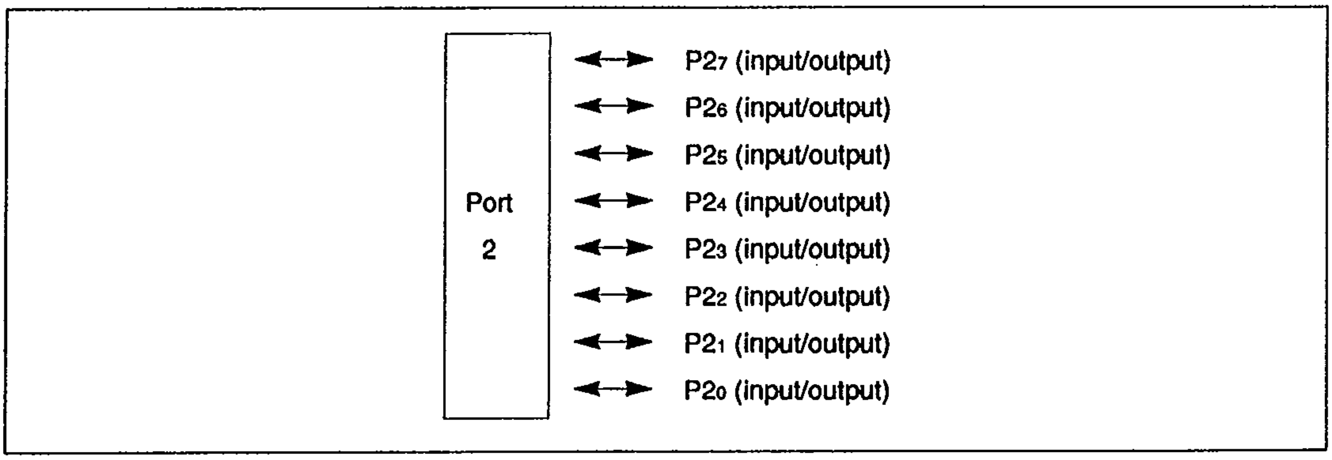


**Figure 9-4 Port 2 Pin Functions in Single-Chip Mode**

## 9.4 Port 3

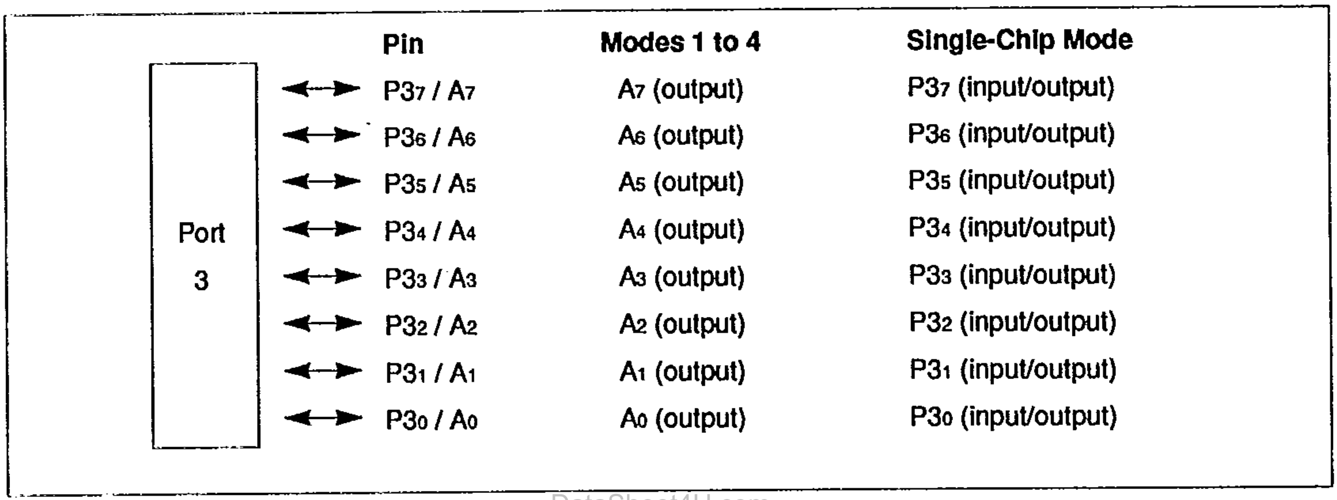
### 9.4.1 Overview

T-49-19-16

Port 3 is an 8-bit input/output port with the pin configuration shown in figure 9-5. In the expanded modes it provides the low bits ($A_7$ – $A_0$) of the address bus. In the single-chip mode it operates as a general-purpose input/output port.

Port 3 has built-in MOS pull-ups that can be turned on or off under program control.

Outputs from port 3 can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair or LED (with 10-mA current sink).



| Pin | Modes 1 to 4 | Single-Chip Mode |
|---|---|---|
| $P3_7$ / $A_7$ | $A_7$ (output) | $P3_7$ (input/output) |
| $P3_6$ / $A_6$ | $A_6$ (output) | $P3_6$ (input/output) |
| $P3_5$ / $A_5$ | $A_5$ (output) | $P3_5$ (input/output) |
| $P3_4$ / $A_4$ | $A_4$ (output) | $P3_4$ (input/output) |
| $P3_3$ / $A_3$ | $A_3$ (output) | $P3_3$ (input/output) |
| $P3_2$ / $A_2$ | $A_2$ (output) | $P3_2$ (input/output) |
| $P3_1$ / $A_1$ | $A_1$ (output) | $P3_1$ (input/output) |
| $P3_0$ / $A_0$ | $A_0$ (output) | $P3_0$ (input/output) |

Figure 9-5 Pin Functions of Port 3

### 9.4.2 Port 3 Registers

Table 9-8 lists the registers of port 3.

**Table 9-8 Port 3 Registers**

| Name | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|
| Port 3 data direction register | P3DDR | W | H'00* | H'FF84 |
| Port 3 data register | P3DR | R/W | H'00 | H'FF86 |

Note: * Initialized to H'00 in modes 2, 4, and 7. Fixed at H'FF in modes 1 and 3.

### 1. Port 3 Data Direction Register (P3DDR)—H'FF84

T-49-19-16

| 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 |

Note: * In mode 2, 4, and 7

P3DDR is an 8-bit register that selects the direction of each pin in port 3.

**Modes 1, 2, 3, and 4:** All bits of P3DDR are fixed at 1 and cannot be modified. Port 3 is used for address bus output.

**Single-Chip Mode:** 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.

P3DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

At a reset and in the hardware standby mode, P3DDR is initialized to H'00, making all eight pins input pins. P3DDR is not initialized in the software standby mode, so if a P3DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 3 data register.

### 2. Port 3 Data Register (P3DR)—H'FF86

| 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 |

T-49-19-16

P3DR is an 8-bit register containing output data for pins $P3_7$ to $P3_0$.

At a reset and in the hardware standby mode, P3DR is initialized to H'00.

When port 3 is read, output pins return the value in the P3DR latch, regardless of the actual level at the pin. Input pins return the level at the pin, not the value in the P3DR latch.

If any of the port 3 data direction bits are cleared to 0, selecting input, use only data transfer (MOV) instructions to write data in P3DR. Do not use arithmetic, logic, or bit manipulation instructions. These instructions read the input pins and may write unintended data in P3DR.

### 9.4.3 Pin Functions in Each Mode

Port 3 has different functions in the expanded modes (modes 1 to 4), and the single-chip mode (mode 7). Separate descriptions are given below.

**Pin Functions in Modes 1 to 4:** In the expanded modes, port 3 is used for output of the low bits ($A_7$ – $A_0$) of the address bus. P3DDR is automatically set for output. Figure 9-6 shows the pin functions for the expanded modes.

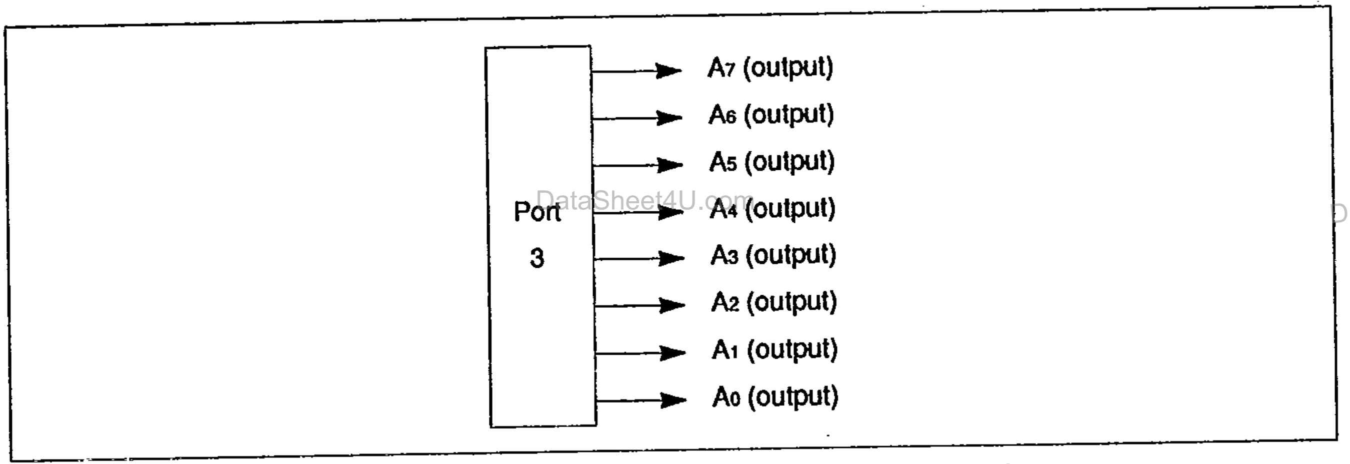


Figure 9-6 Port 3 Pin Functions in Modes 1 and 3

**Pin Functions in Single-Chip Mode:** In the single-chip mode (mode 7), each of the port 3 pins can be designated as an input pin or an output pin, as indicated in figure 9-7, by setting the corresponding bit in P3DDR to 1 for output or clearing it to 0 for input.

T-49-19-16

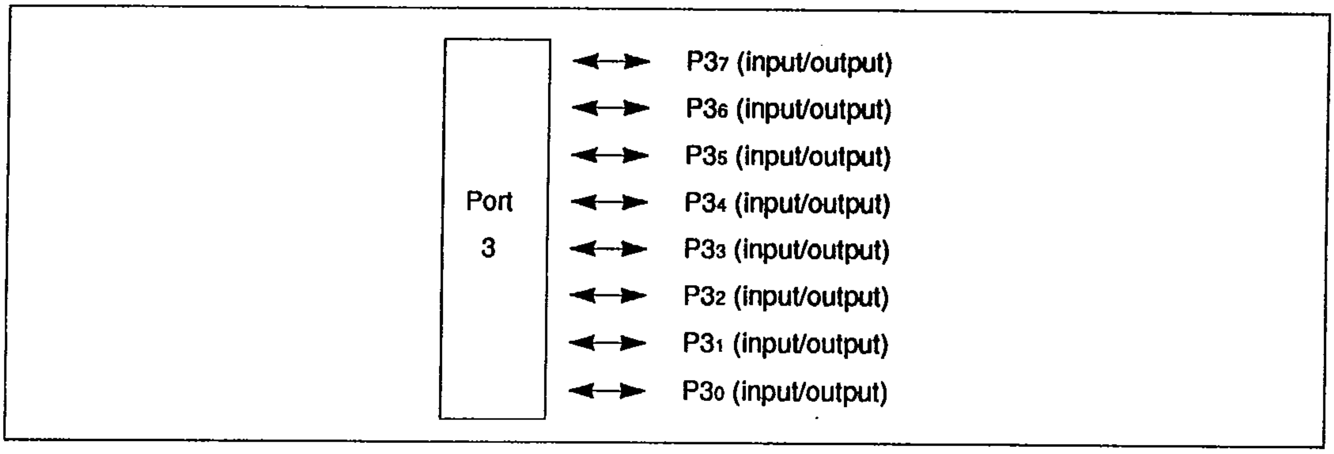


**Figure 9-7 Port 3 Pin Functions in Single-Chip Mode**

### 9.4.4 Built-in MOS Pull-Up

The MOS input pull-ups of port 3 are turned on by clearing the corresponding bit in P3DDR to 0 and writing a 1 in P3DR. These pull-ups are turned off at a reset and in the hardware standby mode. Table 9-9 indicates the status of the MOS pull-ups in various modes.

**Table 9-9 Status of MOS Pull-Ups for Port 3**

| Mode | Reset | Hardware Standby Mode | Other Operating States* |
|---|---|---|---|
| 1 | OFF | OFF | OFF |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 7 | | | ON/OFF |

Note: * Including the software standby mode.
Notation: OFF: The MOS pull-up is always off.
ON/OFF: The MOS pull-up is on when P3DDR = 0 and P3DR = 1, and off otherwise.

**Note on Usage of MOS Pull-Ups:** If a bit manipulation instruction (BSET, BCLR, or BNOT) is used to modify the port 3 data register, since the instruction rewrites the data register according to the levels of input pins, it may switch their built-in MOS pull-ups on or off unintentionally.

T-49-19-16

The same precaution applies to port 4.

**Example (BSET Instruction):** Suppose a BSET instruction is executed to set bit 0 in the port 3 data register (P3DR) under the following conditions.

| | |
|---|---|
| $P3_7$: | Input pin, low, MOS pull-up transistor on |
| $P3_6$: | Input pin, high, MOS pull-up transistor off |
| $P3_5 – P3_0$: | Output pins, low |

The intended purpose of this BSET instruction is to switch the output level at $P3_0$ from low to high.

### Before Execution of BSET Instruction

| | $P3_7$ | $P3_6$ | $P3_5$ | $P3_4$ | $P3_3$ | $P3_2$ | $P3_1$ | $P3_0$ |
|---|---|---|---|---|---|---|---|---|
| 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 | On | Off | Off | Off | Off | Off | Off | Off |

### Execution of BSET Instruction

```
BSET,B  #0,  @PORT3
```

; set bit 0 in port 3 data register

### After Execution of BSET Instruction

| | $P5_7$ | $P5_6$ | $P5_5$ | $P5_4$ | $P5_3$ | $P5_2$ | $P5_1$ | $P5_0$ |
|---|---|---|---|---|---|---|---|---|
| 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 3. Since P3$_7$ and P3$_6$ are input pins, the CPU reads the level of these pins directly, not the value in the P3DR data register. It reads P3$_7$ as low (0) and P3$_6$ as high (1).

T-49-19-16

Since P3$_5$ to P3$_0$ are output pins, for these pins the CPU reads the value in the data register (0). The CPU therefore reads the value of port 3 as H'40, although the actual value in P3DR 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 P3DR to complete the BSET instruction.

As a result, bit P3$_0$ is set to 1, switching pin P3$_0$ to high output. In addition, bits P3$_7$ and P3$_6$ are both modified, changing the on/off settings of the MOS pull-up transistors of pins P3$_7$ and P3$_6$.

**Programming Solution:** The switching of the pull-ups for P3$_7$ and P3$_6$ in this example can be avoided by reserving a one-byte work area in RAM, performing bit manipulations in the work area, then transferring the work area contents to the port 3 data register. RAM0 is a symbol for the user-selected address of the work area below.

### Before Execution of BSET Instruction

```
MOV.B   #80,   R0       ; put write data (H'80) for port 3 data register in R0
MOV.B   R0,    @RAM0    ; transfer from R0 to work area (RAM0)
MOV.B   R0,    @PORT3   ; transfer from R0 to port 3 data register
```

| | P3$_7$ | P3$_6$ | P3$_5$ | P3$_4$ | P3$_3$ | P3$_2$ | P3$_1$ | P3$_0$ |
|---|---|---|---|---|---|---|---|---|
| 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 | 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 work area (RAM0)
```

### After Execution of BSET Instruction

T-49-19-16

```
MOV.B  @RAM0,  R0       ; get value in work area (RAM0)
MOV.B  R0,  @PORT3      ; write value to port 3 data register
```

|  | $P3_7$ | $P3_6$ | $P3_5$ | $P3_4$ | $P3_3$ | $P3_2$ | $P3_1$ | $P3_0$ |
|---|---|---|---|---|---|---|---|---|
| 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 | On | Off | Off | Off | Off | Off | Off | Off |
| RAM0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 9.5 Port 4

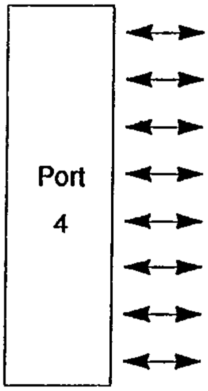
### 9.5.1 Overview

T-49-19-16

Port 4 is an eight-bit input/output port with the pin configuration shown in figure 9-8. In the expanded modes without on-chip ROM (modes 1 and 3), port 4 is used for output of bits $A_{15}$ to $A_8$ of the address bus. In the single-chip mode (mode 7) port 4 is a general-purpose input/output port which can also receive interrupt signals $\overline{IRQ_7}$ to $\overline{IRQ_4}$. In the expanded modes with on-chip ROM (modes 2 and 4), the pins of port 4 function either for output of bits $A_{15}$ – $A_8$ of the address bus, or for general-purpose input and/or output of $\overline{IRQ_7}$ to $\overline{IRQ_4}$.

Port 4 has built-in MOS pull-ups that can be turned on or off under program control.

Outputs from port 4 can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington transistor pair.



| Pin | Modes 1 and 3 | Modes 2 and 4 |
|---|---|---|
| $P4_7$ / $A_{15}$ / $\overline{IRQ_7}$ | $A_{15}$ (output) | $P4_7$ (input) / $A_{15}$ (output) / $\overline{IRQ_7}$ (input) |
| $P4_6$ / $A_{14}$ / $\overline{IRQ_6}$ | $A_{14}$ (output) | $P4_6$ (input) / $A_{14}$ (output) / $\overline{IRQ_6}$ (input) |
| $P4_5$ / $A_{13}$ / $\overline{IRQ_5}$ | $A_{13}$ (output) | $P4_5$ (input) / $A_{13}$ (output) / $\overline{IRQ_5}$ (input) |
| $P4_4$ / $A_{12}$ / $\overline{IRQ_4}$ | $A_{12}$ (output) | $P4_4$ (input) / $A_{12}$ (output) / $\overline{IRQ_4}$ (input) |
| $P4_3$ / $A_{11}$ | $A_{11}$ (output) | $P4_3$ (input) / $A_{11}$ (output) |
| $P4_2$ / $A_{10}$ | $A_{10}$ (output) | $P4_2$ (input) / $A_{10}$ (output) |
| $P4_1$ / $A_9$ | $A_9$ (output) | $P4_1$ (input) / $A_9$ (output) |
| $P4_0$ / $A_8$ | $A_8$ (output) | $P4_0$ (input) / $A_8$ (output) |

**Single-Chip Mode**

$P4_7$ (input/output) / $\overline{IRQ_7}$ (input)
$P4_6$ (input/output) / $\overline{IRQ_6}$ (input)
$P4_5$ (input/output) / $\overline{IRQ_5}$ (input)
$P4_4$ (input/output) / $\overline{IRQ_4}$ (input)
$P4_3$ (input/output)
$P4_2$ (input/output)
$P4_1$ (input/output)
$P4_0$ (input/output)

Figure 9-8 Pin Functions of Port 4

### 9.5.2 Port 4 Registers

T-49-19-16

Table 9-10 lists the registers of port 4.

Table 9-10 Port 4 Registers

| Name | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|
| Port 4 data direction register | P4DDR | W | H'00* | H'FF85 |
| Port 4 data register | P4DR | R/W | H'00 | H'FF87 |

Note: * Initialized to H'00 in modes 2, 4, and 7. Fixed at H'FF in modes 1 and 3.

#### 1. Port 4 Data Direction Register (P4DDR)—H'FF85

| 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 |
| Initial value* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | W | W | W | W | W | W |

Note: * In modes 2, 4, and 7

P4DDR is an 8-bit register that selects the direction of each pin in port 4.

**Expanded Modes Not Using On-Chip ROM (Modes 1 and 3):** All bits of P4DDR are fixed at 1 and cannot be modified. Port 4 is used for address output.

**Expanded Modes Using On-Chip ROM (Modes 2 and 4):** If a bit in P4DDR is set to 1, the corresponding pin is used for address output. If a bit in P4DDR is cleared to 0, the pin is used for general-purpose input. P4DDR is initialized to H'00 at a reset and in the hardware standby mode.

**Single-Chip Mode:** 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.

P4DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

At a reset and in the hardware standby mode, P4DDR is initialized to H'00, making all eight pins input pins. P4DDR is not initialized in the software standby mode, so if a P4DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 4 data register.

T-49-19-16

### 2. Port 4 Data Register (P4DR)—H'FF87

| 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 $P4_0$.

At a reset and in the hardware standby mode, P4DR is initialized to H'00.

When port 4 is read, output pins return the value in the P4DR latch, regardless of the actual level at the pin. Input pins return the level at the pin, not the value in the P4DR latch.

If any of the port 4 data direction bits are cleared to 0, selecting input, use only data transfer (MOV) instructions to write data in P4DR. Do not use arithmetic, logic, or bit manipulation instructions. These instructions read the input pins and may write unintended data in P4DR.

### 9.5.3 Pin Functions in Each Mode

T-49-19-16

Port 4 operates in one way in modes 1 and 3, in another way in modes 2 and 4, and in a third way in mode 7. Separate descriptions are given below.

**Pin Functions in Modes 1 and 3:** In modes 1 and 3 (expanded modes in which the on-chip ROM is not used), all bits of P4DDR are automatically set to 1 for output, and the pins of port 4 carry bits $A_{15}$ – $A_8$ of the address bus. Figure 9-9 shows the pin functions for modes 1 and 3.

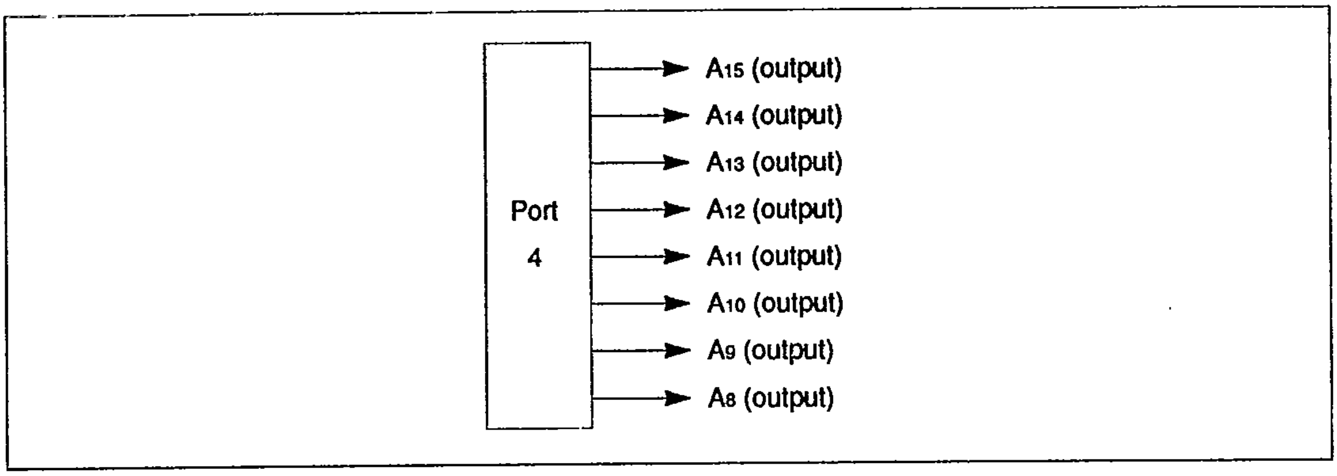


**Figure 9-9 Port 4 Pin Functions in Modes 1 and 3**

**Pin Functions in Modes 2 and 4:** Table 9-11 shows the usage of port 4 in modes 2 and 4.

**Table 9-11 Port 4 Pin Functions in Modes 2 and 4**

T-49-19-16

| Pin | Functions |
|---|---|
| $P4_7$ / $A_{15}$ / $\overline{IRQ_7}$ | The function depends on the IRQ7E bit and $P4_7$DDR bit as follows: |

| IRQ7E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_7$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_7$ input | $A_{15}$ output | $\overline{IRQ_7}$ input and $P4_7$ input | $A_{15}$ output |

| Pin | Functions |
|---|---|
| $P4_6$ / $A_{14}$ / $\overline{IRQ_6}$ | The function depends on the IRQ6E bit and $P4_6$DDR bit as follows: |

| IRQ6E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_6$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_6$ input | $A_{14}$ output | $\overline{IRQ_6}$ input and $P4_6$ input | $A_{14}$ output |

| Pin | Functions |
|---|---|
| $P4_5$ / $A_{13}$ / $\overline{IRQ_5}$ | The function depends on the IRQ5E bit and $P4_5$DDR bit as follows: |

| IRQ5E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_5$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_5$ input | $A_{13}$ output | $\overline{IRQ_5}$ input and $P4_5$ input | $A_{13}$ output |

| Pin | Functions |
|---|---|
| $P4_4$ / $A_{12}$ / $\overline{IRQ_4}$ | The function depends on the IRQ4E bit and $P4_4$DDR bit as follows: |

| IRQ4E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_4$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_4$ input | $A_{12}$ output | $\overline{IRQ_4}$ input and $P4_4$ input | $A_{12}$ output |

**Table 9-11 Port 4 Pin Functions in Modes 2 and 4 (cont)**

T-49-19-16

| Pin | Functions | | |
|---|---|---|---|
| $P4_3$ / $A_{11}$ | $P4_3$DDR | 0 | 1 |
| | Pin function | $P4_3$ input | $A_{11}$ output |
| $P4_2$ / $A_{10}$ | $P4_2$DDR | 0 | 1 |
| | Pin function | $P4_2$ input | $A_{10}$ output |
| $P4_1$ / $A_9$ | $P4_1$DDR | 0 | 1 |
| | Pin function | $P4_1$ input | $A_9$ output |
| $P4_0$ / $A_8$ | $P4_0$DDR | 0 | 1 |
| | Pin function | $P4_0$ input | $A_8$ output |

**Pin Functions in Single-Chip Mode:** Table 9-12 shows the usage of port 4 in the single-chip mode.

### Table 9-12 Port 4 Pin Functions in Single-Chip Mode

T-49-19-16

**Pin** **Functions**

$P4_7$ / $\overline{IRQ_7}$

The function depends on the IRQ7E bit and $P4_7$DDR bit as follows:

| IRQ7E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_7$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_7$ input | $P4_7$ output | $\overline{IRQ_7}$ input<br>and $P4_7$ input | $\overline{IRQ_7}$ input<br>and $P4_7$ output |

$P4_6$ / $\overline{IRQ_6}$

The function depends on the IRQ6E bit and $P4_6$DDR bit as follows:

| IRQ6E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_6$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_6$ input | $P4_6$ output | $\overline{IRQ_6}$ input<br>and $P4_6$ input | $\overline{IRQ_6}$ input<br>and $P4_6$ output |

$P4_5$ / $\overline{IRQ_5}$

The function depends on the IRQ5E bit and $P4_5$DDR bit as follows:

| IRQ5E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_5$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_5$ input | $P4_5$ output | $\overline{IRQ_5}$ input<br>and $P4_5$ input | $\overline{IRQ_5}$ input<br>and $P4_5$ output |

$P4_4$ / $\overline{IRQ_4}$

The function depends on the IRQ4E bit and $P4_4$DDR bit as follows:

| IRQ4E | 0 | | 1 | |
|---|---|---|---|---|
| $P4_4$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P4_4$ input | $P4_4$ output | $\overline{IRQ_4}$ input<br>and $P4_4$ input | $\overline{IRQ_4}$ input<br>and $P4_4$ output |

**Table 9-12 Port 4 Pin Functions in Single-Chip Mode (cont)**

T-49-19-16

**Pin** **Functions**

$P4_3$

| $P4_3$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P4_3$ input | $P4_3$ output |

$P4_2$

| $P4_2$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P4_2$ input | $P4_2$ output |

$P4_1$

| $P4_1$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P4_1$ input | $P4_1$ output |

$P4_0$

| $P4_0$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P4_0$ input | $P4_0$ output |

T-49-19-16

### 9.5.4 Built-In MOS Pull-Up

The MOS input pull-ups of port 4 are turned on by clearing the corresponding bit in P4DDR to 0 and writing a 1 in P4DR. These pull-ups are turned off at a reset and in the hardware standby mode. Table 9-13 indicates the status of the MOS pull-ups in various modes.

**Table 9-13 Status of MOS Pull-Ups for Port 4**

| Mode | Reset | Hardware Standby Mode | Other Operating States* |
|---|---|---|---|
| 1 | OFF | OFF | OFF |
| 2 | | | ON/OFF |
| 3 | | | OFF |
| 4 | | | ON/OFF |
| 7 | | | ON/OFF |

Notes: * Including the software standby mode.

Notation: OFF: The MOS pull-up is always off.

ON/OFF: The MOS pull-up is on when P4DDR = 0 and P4DR = 1, and off otherwise.

**Note on Usage of MOS Pull-Ups:** See the note in section 9.4.4, "Built-in MOS Pull-up".

## 9.6 Port 5

T-49-19-16

### 9.6.1 Overview

Port 5 is an eight-bit input/output port with the pin configuration shown in figure 9-10. Its pins also carry input and output signals for the free-running timers (FRT1 and FRT2) and 8-bit timer, and pin 7 can output the system clock (ø).

Port 5 has Schmitt inputs. Outputs from port 5 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair.

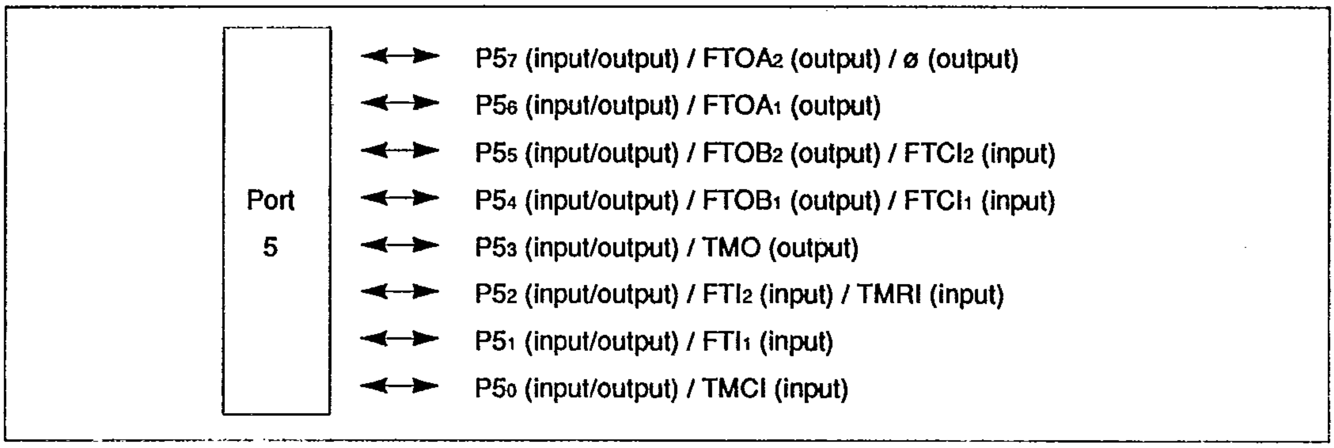


**Figure 9-10 Pin Functions of Port 5**

### 9.6.2 Port 5 Registers

Table 9-14 lists the registers of port 5.

**Table 9-14 Port 5 Registers**

| Name | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|
| Port 5 data direction register | P5DDR | W | H'00 | H'FF88 |
| Port 5 data register | P5DR | R/W | H'00 | H'FF8A |

T-49-19-16

### 1. Port 5 Data Direction Register (P5DDR)—H'FF88

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | P5$_7$DDR | P5$_6$DDR | P5$_5$DDR | P5$_4$DDR | P5$_3$DDR | P5$_2$DDR | P5$_1$DDR | P5$_0$DDR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | W | W | 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.

P5DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

At a reset and in the hardware standby mode, P5DDR is initialized to H'00, setting all pins for input. P5DDR is not initialized in the software standby mode, so if a P5DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 5 data register.

A transition to the software standby mode initializes the on-chip supporting modules, so any pins of port 5 that were being used by an on-chip timer when the transition occurs revert to general-purpose input or output, controlled by P5DDR and P5DR.

### 2. Port 5 Data Register (P5DR)—H'FF8A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | P5$_7$ | P5$_6$ | P5$_5$ | P5$_4$ | P5$_3$ | P5$_2$ | P5$_1$ | P5$_0$ |
| 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 |

P5DR is an 8-bit register containing output data for pins P5$_7$ to P5$_0$.

P5DR is initialized to H'00 by a reset and in the hardware standby mode.

When port 5 is read, output pins return the value in the P5DR latch, regardless of the actual level at the pin. Input pins return the level at the pin, not the value in the P5DR latch.

If any of the port 5 data direction bits are cleared to 0, selecting input, use only data transfer (MOV) instructions to write data in P5DR. Do not use arithmetic, logic, or bit manipulation instructions. These instructions read the input pins and may write unintended data in P5DR.

### 9.6.3 Pin Functions

T-49-19-16

The pin functions of port 5 are the same in all MCU operating modes. As figure 9-10 indicated, these pins are used for input and output of on-chip timer signals as well as for general-purpose input and output. For some pins, two or more functions can be enabled simultaneously.

Table 9-15 shows how the functions of the pins of port 5 are selected.

**Table 9-15 Port 5 Pin Functions**

| Pin | Functions |
|---|---|
| $P5_7$ /<br>$FTOA_2$ / ø | The function depends on the output enable A bit (OEA) of the FRT1 timer control register (TCR), the P57DDR bit, and the system clock output enable bit (øOE) in the port 7 data direction register, as follows: |

| øOE | 1 | | | | 0 | | | |
|---|---|---|---|---|---|---|---|---|
| OEA | 0 | | 1 | | 0 | | 1 | |
| $P5_7$DDR | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Pin function | ø output | | | | $P5_7$ input | $P5_7$ output | $FTOA_2$ output | |

Note: A reset initializes øOE to 0 in mode 7 and to 1 in modes 1, 2, 3, and 4.

## Table 9-15 Port 5 Pin Functions (cont)

T-49-19-16

| Pin | Functions |
|---|---|
| P5$_6$ / FTOA$_2$ | The function depends on the output enable A bit (OEA) of the FRT2 timer control register (TCR) and on the P5$_6$DDR bit as follows: |

| OEB | 0 | | 1 | |
|---|---|---|---|---|
| P5$_6$DDR | 0 | 1 | 0 | 1 |
| Pin function | P5$_6$ input | P5$_6$ output | FTOA$_1$ output | |

| Pin | Functions |
|---|---|
| P5$_5$ / FTOB$_2$ / FTCI$_2$ | The function depends on the output enable B bit (OEB) of the FRT2 timer control register (TCR) and on the P5$_5$DDR bit as follows: |

| OEB | 0 | | 1 | |
|---|---|---|---|---|
| P5$_5$DDR | 0 | 1 | 0 | 1 |
| Pin function | P5$_5$ input | P5$_5$ output | FTOB$_2$ output | |
| | FTCI$_2$ input | | | |

| Pin | Functions |
|---|---|
| P5$_4$ / FTOB$_1$ / FTCI$_1$ | The function depends on the output enable B bit (OEB) of the FRT1 timer control register (TCR) and on the P5$_4$DDR bit as follows: |

| OEB | 0 | | 1 | |
|---|---|---|---|---|
| P5$_4$DDR | 0 | 1 | 0 | 1 |
| Pin function | P5$_4$ input | P5$_4$ output | FTOB$_1$ output | |
| | FTCI$_1$ input | | | |

T-49-19-16

## Table 9-15 Port 5 Pin Functions (cont)

**Pin** — **Functions**

---

$P5_3$ / TMO

The function depends on output select bits 3 to 0 (OS3 to OS0) in the timer control/status register (TCSR) of the 8-bit timer, and on the $P5_3$DDR bit as follows:

| OS3 to OS0 | All three bits are 0 | | At least one bit is set to 1 | |
|---|---|---|---|---|
| $P5_3$DDR | 0 | 1 | 0 | 1 |
| Pin function | $P5_3$ input | $P5_3$ output | TMO output | |

---

$P5_2$ / $FTI_2$ / TMRI

In addition to functioning for general-purpose input or output, this pin receives the input capture signal ($FTI_2$) for free-running timer 2 and the reset input (TMRI) for the 8-bit timer. TMRI input is enabled when the counter clear bits (CCLR1 and CCLR0) in the timer control register (TCR) are both set to 1.

| $P5_2$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P5_2$ input | $P5_2$ output |
| | $FTI_2$ and TMRI input | |

---

$P5_1$ / $FTI_1$

| $P5_1$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P5_1$ input | $P5_1$ output |
| | $FTI_1$ input | |

---

$P5_0$ / TMCI

In addition to functioning for general-purpose input or output, this pin can simultaneously be used for external clock input for the 8-bit timer, depending on clock select bits 2 to 0 (CKS2, CKS1, and CKS0) in the timer control register (TCR).

| $P5_0$DDR | 0 | 1 |
|---|---|---|
| Pin function | $P5_0$ input | $P5_0$ output |
| | TMCI input | |

## 9.7 Port 6

T-49-19-16

### 9.7.1 Overview

Port 6 is a 4-bit input port that also receives inputs for the on-chip A/D converter. The pin functions are the same in all MCU operating modes, as shown in figure 9-11.

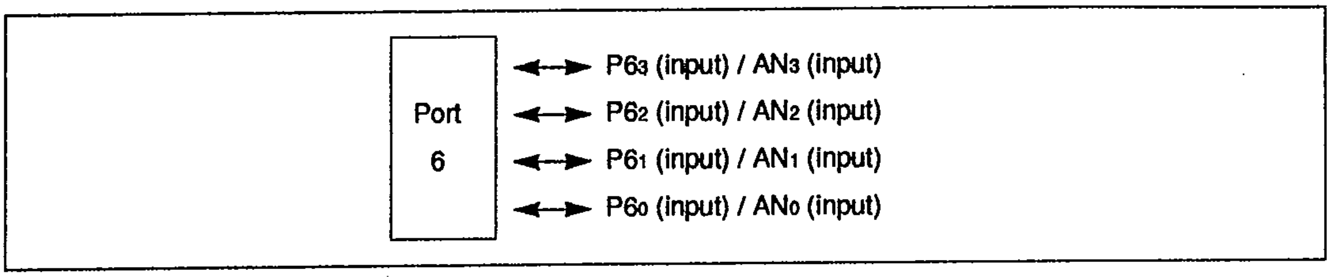


Figure 9-11 Pin Functions of Port 6

In the 68-pin CP-68 package, port 6 has eight pins for general-purpose input and analog input. Figure 9-12 shows the pin configuration of port 6 in the CP-68 package.

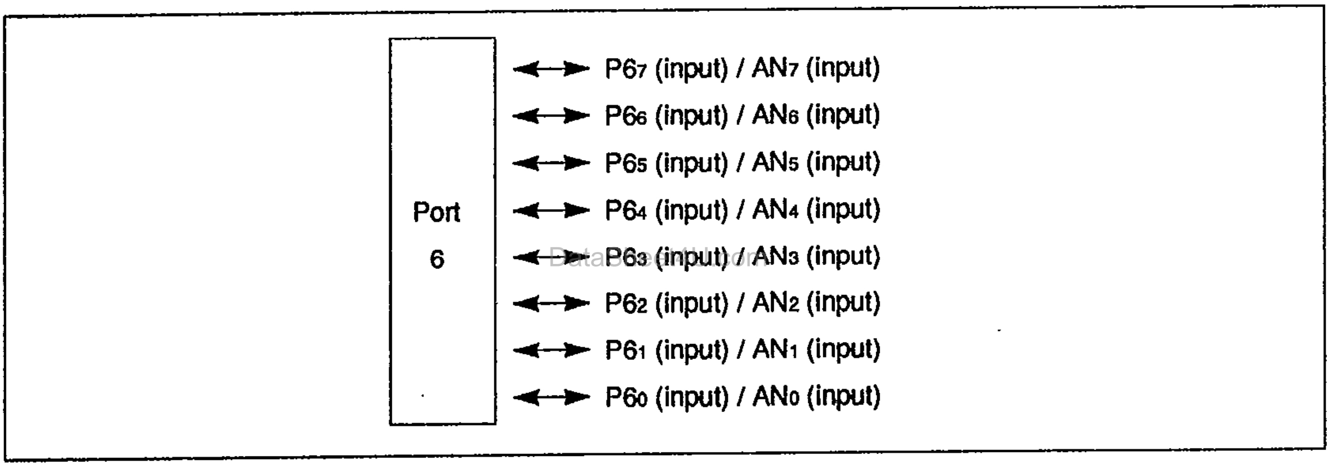


Figure 9-12 Pin Functions of Port 6 (CP-68 Package)

### 9.7.2 Port 6 Registers

Port 6 has only the data register described in table 9-16. Since it is exclusively an input port, there is no data direction register.

T-49-19-16

**Table 9-16 Port 6 Registers**

| Name | Abbreviation | Read/Write | Address |
|---|---|---|---|
| Port 6 data register | P6DR | R | H'FF8B |

### 1. Port 6 Data Register (P6DR)—H'FF8B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $P6_7$ | $P6_6$ | $P6_5$ | $P6_4$ | $P6_3$ | $P6_2$ | $P6_1$ | $P6_0$ |
| Read/Write | R | R | R | R | R | R | R | R |

Note: Bits 7 to 4 are valid in the CP-68 package only.

When the CPU reads P6DR it always reads the current status of each pin, except that during A/D conversion the pin currently being converted reads 1 regardless of the actual input voltage at that pin.

In a 64-pin package, the data read from the upper four bits are indeterminate.

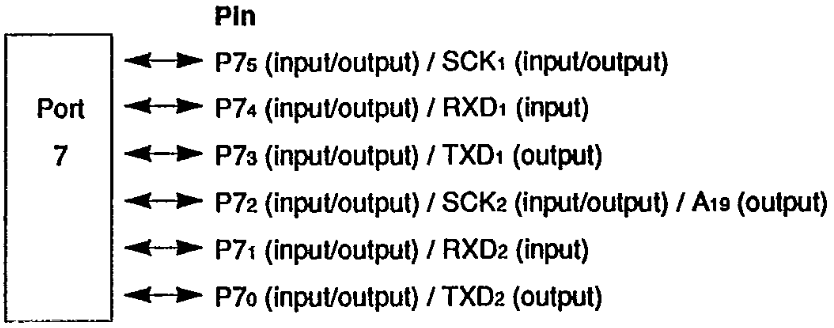
## 9.8 Port 7

### 9.8.1 Overview

Port 7 is a 6-bit input/output port with the pin configuration shown in figure 9-13. In addition to general-purpose input and output, its pins are used for input and output by the on-chip serial communication interface (SCI). In the expanded maximum modes (modes 3 and 4), it also supplies bit A19 of the page address bus.

Outputs from port 7 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington transistor pair.

T-49-19-16



**Modes 1, 2, and 7**

$P7_5$ (input/output) / $SCK_1$ (input/output)
$P7_4$ (input/output) / $RXD_1$ (input)
$P7_3$ (input/output) / $TXD_1$ (output)
$P7_2$ (input/output) / $SCK_2$ (input/output)
$P7_1$ (input/output) / $RXD_2$ (input)
$P7_0$ (input/output) / $TXD_2$ (output)

**Mode 3**

$P7_5$ (input/output) / $SCK_1$ (input/output)
$P7_4$ (input/output) / $RXD_1$ (input)
$P7_3$ (input/output) / $TXD_1$ (output)
$A_{19}$ (output)
$P7_1$ (input/output) / $RXD_2$ (input)
$P7_0$ (input/output) / $TXD_2$ (output)

**Mode 4**

$P7_5$ (input/output) / $SCK_1$ (input/output)
$P7_4$ (input/output) / $RXD_1$ (input)
$P7_3$ (input/output) / $TXD_1$ (output)
$P7_2$ (input/output) / $SCK_2$ (input/output) / $A_{19}$ (output)
$P7_1$ (input/output) / $RXD_2$ (input)
$P7_0$ (input/output) / $TXD_2$ (output)

**Figure 9-13 Pin Functions of Port 7**

## 9.8.2 Port 7 Registers

Table 9-17 lists the registers of port 7.

**Table 9-17 Port 7 Registers**

| Name | Abbreviation | Read/Write | Initial Value | Address |
|---|---|---|---|---|
| Port 7 data direction register | P7DDR | W | H'40* | H'FF8C |
| Port 7 data register | P7DR | R/W | H'00 | H'FF8E |

Note: * Initialized to H'40 in modes 1, 2, 3, and 4, and to H'00 in mode 7.

### 1. Port 7 Data Direction Register (P7DDR)—H'FF8C

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | øOE | P7$_5$DDR | P7$_4$DDR | P7$_3$DDR | P7$_2$DDR | P7$_1$DDR | P7$_0$DDR |
| Initial value | — | 1/0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | — | W | W | W | W | W | W | W |

P7DDR is an 8-bit register that selects the direction of each pin in port 7. Bit 7 is reserved. Bit 6 selects whether the system clock (ø) is output at pin P5$_7$ in port 5.

The usage of P7DDR depends on the MCU operating mode as explained below.

**Modes 1, 2, and 4:** A pin functions as an output pin if the corresponding bit in P7DDR is set to 1, and as an input pin if the bit is cleared to 0.

P7DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

At a reset and in the hardware standby mode, P7DDR is initialized to H'40, setting all pins to the input state. P7DDR is not initialized in the software standby mode, so if a P7DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 7 data register.

A transition to the software standby mode initializes the serial communication interface module, so any pins of port 7 that were being used for serial communication when the transition occurs revert to general-purpose input or output, controlled by P7DDR and P7DR.

**Mode 3:** Bit 2 is fixed at the value 1 and pin $P7_2$ is used for page address output.

Bits 5 to 3 and 1 to 0 can be set to 1 for output or cleared to 0 for input as in the other MCU modes.

**Mode 7:** In single-chip mode, P7DDR is initialized to H'00 at a reset and in the hardware standby mode.

### 2. Port 7 Data Register (P7DR)—H'FF8E

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

P7DR is an 8-bit register containing the data for pins $P7_6$ to $P7_0$. Bits 7 and 6 are reserved.

When port 7 is read, output pins return the value in the P7DR latch, regardless of the actual level at the pin. Input pins return the level at the pin, not the value in the P7DR latch.

If any of the port 7 data direction bits are cleared to 0, selecting input, use only data transfer (MOV) instructions to write data in P7DR. Do not use arithmetic, logic, or bit manipulation instructions. These instructions read the input pins and may write unintended data in P7DR.

### 9.8.3 Pin Functions

The pin functions of port 7 depend on the MCU operating mode. Table 9-18 shows how the functions are selected in modes 1, 2, and 7. Table 9-19 shows how they are selected in mode 3. Table 9-20 shows how they are selected in mode 4.

## Table 9-18 Port 7 Pin Functions in Modes 1, 2, and 7

T-49-19-16

**Pin** | **Functions**

---

**P7$_5$ / SCK$_1$**

The function depends on the communication mode bit (C/$\overline{A}$) and the clock enable 1 and 2 bits (CKE1 and CKE0) of the serial control register (SCR) of SCI1 as follows:

| C/$\overline{A}$ | 0 | | | | 1 | | | |
|---|---|---|---|---|---|---|---|---|
| CKE1 | 0 | | 1 | | 0 | | 1 | |
| CKE0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Pin function | P7$_5$ input or output* | SCI1 internal clock output | SCI1 external clock input | | SCI1 internal clock output | | SCI1 external clock input | |

Note: * Input or output is selected by the P7$_5$DDR bit.

---

**P7$_4$ / RXD$_1$**

The function depends on the receive enable bit (RE) of the serial control register (SCR) of SCI1 and on the P7$_4$DDR bit as follows:

| RE | 0 | | 1 | |
|---|---|---|---|---|
| P7$_4$DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_4$ input | P7$_4$ output | RXD$_1$ input | |

---

**P7$_3$ / TXD$_1$**

The function depends on the transmit enable bit (TE) of the serial control register (SCR) of SCI1 and on the P7$_3$DDR bit as follows:

| TE | 0 | | 1 | |
|---|---|---|---|---|
| P7$_3$DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_3$ input | P7$_3$ output | TXD$_1$ output | |

### Table 9-18 Port 7 Pin Functions in Modes 1, 2, and 7 (cont)

T-49-19-16

**Pin** **Functions**

P7$_2$ / SCK$_2$

The function depends on the communication mode bit (C/$\overline{A}$) and the clock enable 1 and 2 bits (CKE1 and CKE0) of the serial control register (SCR) of SCI2 as follows:

| C/$\overline{A}$ | 0 | | | | 1 | | | |
|---|---|---|---|---|---|---|---|---|
| CKE1 | 0 | | 1 | | 0 | | 1 | |
| CKE0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Pin function | P7$_2$ input or output* | SCI2 internal clock output | SCI2 external clock input | | SCI2 internal clock output | | SCI2 external clock input | |

Note: * Input or output is selected by the P7$_2$DDR bit.

P7$_1$ / RXD$_2$

The function depends on the receive enable bit (RE) of the serial control register (SCR) of SCI2 and on the P7$_1$DDR bit as follows:

| RE | 0 | | 1 | |
|---|---|---|---|---|
| P71DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_1$ input | P7$_1$ output | RXD$_2$ input | |

P7$_0$ / TXD$_2$

The function depends on the transmit enable bit (TE) of the serial control register (SCR) of SCI2 and on the P7$_0$DDR bit as follows:

| TE | 0 | | 1 | |
|---|---|---|---|---|
| P7$_0$DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_0$ input | P7$_0$ output | TXD$_2$ output | |

## Table 9-19 Port 7 Pin Functions in Mode 3

T-49-19-16

**Pin** **Functions**

P7$_5$ / SCK1 — The function depends on the communication mode bit (C/$\overline{A}$) and the clock enable 1 and 2 bits (CKE1 and CKE0) of the serial control register (SCR) of SCI1 as follows:

| C/$\overline{A}$ | 0 | | | | 1 | | | |
|---|---|---|---|---|---|---|---|---|
| CKE1 | 0 | | 1 | | 0 | | 1 | |
| CKE0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Pin function | P7$_5$ input or output* | SCI1 internal clock output | SCI1 external clock input | | SCI1 internal clock output | | SCI1 external clock input | |

Note: * Input or output is selected by the P7$_5$DDR bit.

P7$_4$ / RXD$_1$ — The function depends on the receive enable bit (RE) of the serial control register (SCR) of SCI1 and on the P7$_4$DDR bit as follows:

| RE | 0 | | 1 | |
|---|---|---|---|---|
| P7$_4$DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_4$ input | P7$_4$ output | RXD$_1$ input | |

T-49-19-16

### Table 9-19 Port 7 Pin Functions in Mode 3 (cont)

| Pin | Functions |
|---|---|
| P7$_3$ / TXD$_1$ | The function depends on the transmit enable bit (TE) of the serial control register (SCR) of SCI1 and on the P7$_3$DDR bit as follows: |

| TE | 0 | | 1 | |
|---|---|---|---|---|
| P7$_3$DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_3$ input | P7$_3$ output | TXD$_1$ output | |

| Pin | Functions |
|---|---|
| A$_{19}$ | A$_{19}$ page address output. |
| P7$_1$ / RXD$_2$ | The function depends on the receive enable bit (RE) of the serial control register (SCR) of SCI2 and on the P7$_1$DDR bit as follows: |

| RE | 0 | | 1 | |
|---|---|---|---|---|
| P71DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_1$ input | P7$_1$ output | RXD$_2$ input | |

| Pin | Functions |
|---|---|
| P7$_0$ / TXD$_2$ | The function depends on the transmit enable bit (TE) of the serial control register (SCR) of SCI2 and on the P7$_0$DDR bit as follows: |

| TE | 0 | | 1 | |
|---|---|---|---|---|
| P7$_0$DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_0$ input | P7$_0$ output | TXD$_2$ output | |

## Table 9-20 Port 7 Pin Functions in Mode 4

T-49-19-16

| Pin | Functions |
|---|---|
| P7₅ / SCK₁ | The function depends on the communication mode bit (C/$\overline{A}$) and the clock enable 1 and 2 bits (CKE1 and CKE0) of the serial control register (SCR) of SCI1 as follows: |

| C/$\overline{A}$ | 0 | | | | 1 | | | |
|---|---|---|---|---|---|---|---|---|
| CKE1 | 0 | | 1 | | 0 | | 1 | |
| CKE0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Pin function | P7₅ input or output* | SCI1 internal clock output | SCI1 external clock input | | SCI1 internal clock output | | SCI1 external clock input | |

Note: * Input or output is selected by the P7₅DDR bit.

| Pin | Functions |
|---|---|
| P7₄ / RXD₁ | The function depends on the receive enable bit (RE) of the serial control register (SCR) of SCI1 and on the P7₄DDR bit as follows: |

| RE | 0 | | 1 | |
|---|---|---|---|---|
| P7₄DDR | 0 | 1 | 0 | 1 |
| Pin function | P7₄ input | P7₄ output | RXD₁ input | |

## Table 9-20 Port 7 Pin Functions in Mode 4 (cont)

T-49-19-16

| Pin | Functions |
|---|---|
| P7$_3$ / TXD$_1$ | The function depends on the transmit enable bit (TE) of the serial control register (SCR) of SCI1 and on the P7$_3$DDR bit as follows: |

| TE | 0 | | 1 | |
|---|---|---|---|---|
| P7$_3$DDR | 0 | 1 | 0 | 1 |
| Pin function | P7$_3$ input | P7$_3$ output | TXD$_1$ output | |

| Pin | Functions |
|---|---|
| P7$_2$ / A$_{19}$ / SCK$_2$ | The function depends on the C/A, CKE1, and CKE0 bits of the serial control register (SCR) of SCI2 and on P7$_2$DDR as follows: |

| P7$_2$DDR | 0 | | | |
|---|---|---|---|---|
| $C/\overline{A}$ | 0 | | | |
| CKE1 | 0 | | 1 | |
| CKE0 | 0 | 1 | 0 | 1 |
| Pin function | P7$_2$ input | SCI2 internal clock output | SCI2 external clock input | |

| P7$_2$DDR | 0 | | | | 1 |
|---|---|---|---|---|---|
| $C/\overline{A}$ | 1 | | | | Don't care |
| CKE1 | 0 | | 1 | | Don't care |
| CKE0 | 0 | 1 | 0 | 1 | Don't care |
| Pin function | SCI2 internal clock output | | SCI2 external clock input | | A$_{19}$ output |

Table 9-20 Port 7 Pin Functions in Mode 4 (cont)

| Pin | Functions |
|---|---|
| $P7_1$ / $RXD_2$ | The function depends on the receive enable bit (RE) of the serial control register (SCR) of SCI2 and on the $P7_1DDR$ bit as follows: |

| RE | 0 | | 1 | |
|---|---|---|---|---|
| $P7_1DDR$ | 0 | 1 | 0 | 1 |
| Pin function | $P7_1$ input | $P7_1$ output | $RXD_2$ input | |

| Pin | Functions |
|---|---|
| $P7_0$ / $TXD_2$ | The function depends on the transmit enable bit (TE) of the serial control register (SCR) of SCI2 and on the $P7_0DDR$ bit as follows: |

| TE | 0 | | 1 | |
|---|---|---|---|---|
| $P7_0DDR$ | 0 | 1 | 0 | 1 |
| Pin function | $P7_0$ input | $P7_0$ output | $TXD_2$ output | |

# Section 10 16-Bit Free-Running Timers

## 10.1 Overview

The H8/520 has an on-chip 16-bit free-running timer (FRT) module with two independent channels (FRT1 and FRT2). Both channels are functionally identical.

Each channel has a 16-bit free-running counter that it uses as a time base. Applications of the FRT module include rectangular-wave output (up to two independent waveforms per channel), input pulse width measurement, and measurement of external clock periods.

### 10.1.1 Features

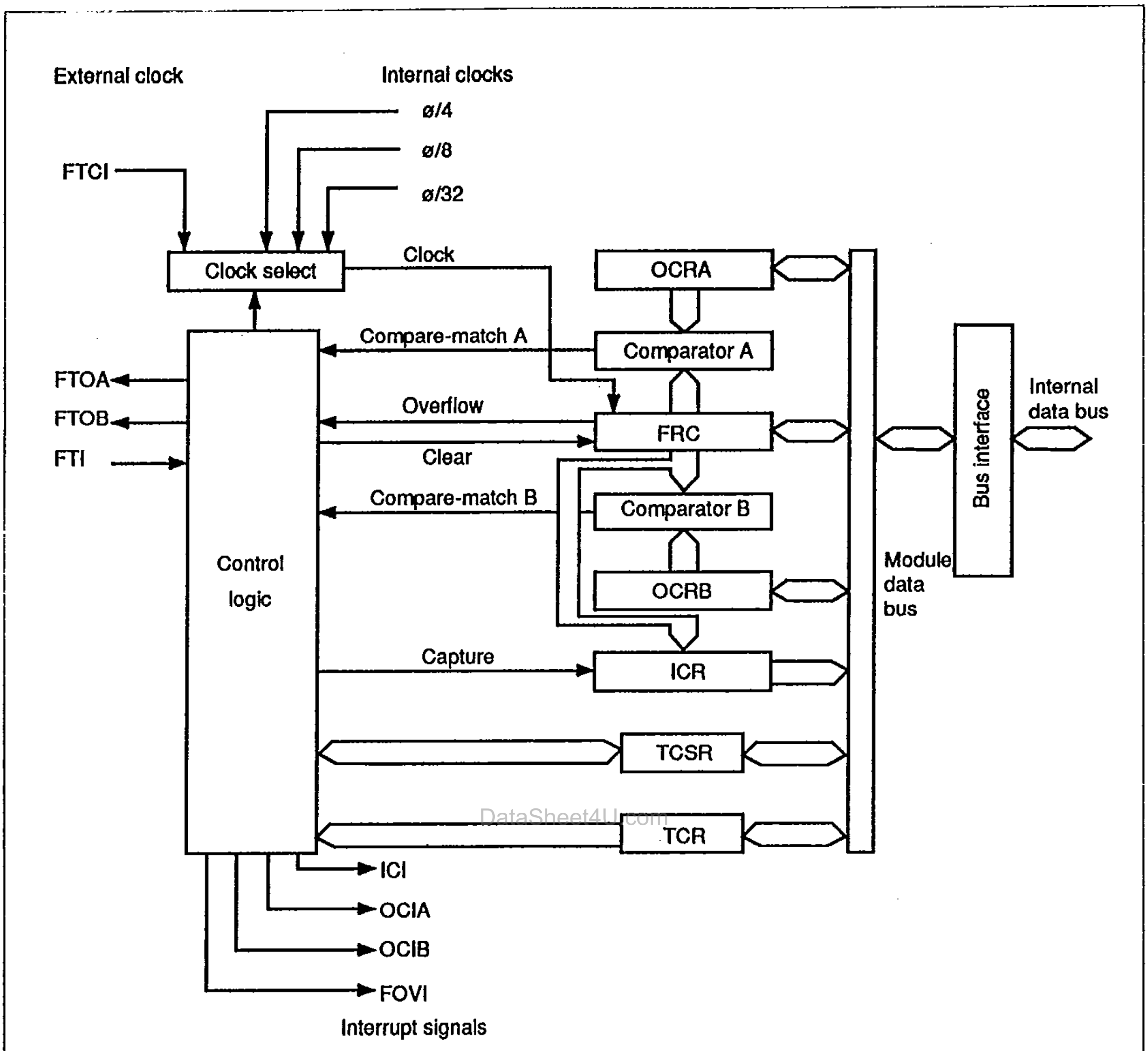
The features of the free-running timer module are listed below.

- Selection of four clock sources
  The free-running counters can be driven by an internal clock source (ø/4, ø/8, or ø/32), or an external clock input (enabling use as an external event counter).
- Two independent comparators
  Each free-running timer channel can generate two independent waveforms.
- Input capture function
  The current count can be captured on the rising or falling edge (selectable) of an input signal.
- Four types of interrupts
  Compare-match A and B, input capture, and overflow interrupts can be requested independently. The compare-match and input capture interrupts can be served by the data transfer controller (DTC), enabling interrupt-driven data transfer with minimal CPU programming.
- Counter can be cleared under program control
  The free-running counters can be cleared on compare-match A.

### 10.1.2 Block Diagram

T-49-19-16

Figure 10-1 shows a block diagram of one free-running timer channel.



OCRA: Output Compare Register A
OCRB: Output Compare Register B
FRC: Free Running Counter
ICR: Input Capture Register
TCSR: Timer Control/Status Register
TCR: Timer Control Register

Figure 10-1 Block Diagram of 16-Bit Free-Running Timer

### 10.1.3 Input and Output Pins

T-49-19-16

Table 10-1 lists the input and output pins of the free-running timer module.

**Table 10-1 Input and Output Pins of Free-Running Timer Module**

| Channel | Name | Abbreviation | I/O | Function |
|---|---|---|---|---|
| 1 | Output compare A | $FTOA_1$ | Output | Output controlled by comparator A of FRT1 |
| | Output compare B or counter clock input | $FTOB_1$ / $FTCI_1$ | Output / Input | Output controlled by comparator B of FRT1, or input of external clock source for FRT1 |
| | Input capture | $FTI_1$ | Input | Trigger for capturing current count of FRT1 |
| 2 | Output compare A* | $FTOA_2$ | Output | Output controlled by comparator A of FRT2 |
| | Output compare B or counter clock input | $FTOB_2$ / $FTCI_2$ | Output / Input | Output controlled by comparator B of FRT2, or input of external clock source for FRT2 |
| | Input capture | $FTI_2$ | Input | Trigger for capturing current count of FRT2 |

Note: * When the øOE bit in P7DDR is set to 1, this pin is used for system clock (ø) output and cannot be used for $FTOA_2$.

### 10.1.4 Register Configuration

T-49-19-16

Table 10-2 lists the registers of each free-running timer channel.

**Table 10-2 Register Configuration**

| Channel | Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|---|
| 1 | 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 |
| 2 | Timer control register | TCR | R/W | H'00 | H'FFA0 |
| | Timer control/status register | TCSR | R/(W)* | H'00 | H'FFA1 |
| | Free-running counter (High) | FRC (H) | R/W | H'00 | H'FFA2 |
| | Free-running counter (Low) | FRC (L) | R/W | H'00 | H'FFA3 |
| | Output compare register A (High) | OCRA (H) | R/W | H'FF | H'FFA4 |
| | Output compare register A (Low) | OCRA (L) | R/W | H'FF | H'FFA5 |
| | Output compare register B (High) | OCRB (H) | R/W | H'FF | H'FFA6 |
| | Output compare register B (Low) | OCRB (L) | R/W | H'FF | H'FFA7 |
| | Input capture register (High) | ICR (H) | R | H'00 | H'FFA8 |
| | Input capture register (Low) | ICR (L) | R | H'00 | H'FFA9 |

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

## 10.2 Register Descriptions

### 10.2.1 Free-Running Counter (FRC)—H'FF92, H'FFA2

| 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 |

Each 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).

The FRC can be cleared by compare-match A.

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 10.3, "CPU Interface", for details.

The FRCs are initialized to H'0000 at a reset and in the standby modes.

### 10.2.2 Output Compare Registers A and B (OCRA and OCRB)—H'FF94 and H'FF96, H'FFA4 and H'FFA6

| 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 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 timer control status register (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 when they are written. See section 10.3, "CPU Interface", for details.

T-49-19-16

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

### 10.2.3 Input Capture Register (ICR)—H'FF98, H'FFA8

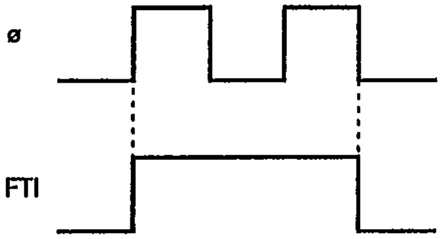
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R |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

The ICR is a 16-bit read-only register.

When the rising or falling edge of the signal at the input capture input pin is detected, the current value of the FRC is copied to the 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 ICR is a 16-bit register, a temporary register (TEMP) is used when the ICR is written or read. See section 10.3, "CPU Interface", for details.

To ensure input capture, the pulse width of the input capture signal should be at least 1.5 system clock periods (1.5 ø).



Minimum FTI Pulse Width

The ICR 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 ICR even if the input capture flag (ICF) is already set.

### 10.2.4 Timer Control Register (TCR)—H'FF90, H'FFAO

| 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 selects the FRC clock source, enables the output compare signals, and enables interrupts.

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

**Bit 7—Input Capture Interrupt Enable (ICIE):** This bit 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<br>ICIE | Description | |
|---|---|---|
| 0 | The input capture interrupt request (ICI) is disabled. | (Initial value) |
| 1 | The input capture interrupt request (ICI) is enabled. | |

**Bit 6—Output Compare Interrupt Enable B (OCIEB):** This bit 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<br>OCIEB | 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 Enable A (OCIEA):** This bit 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<br>OCIEA | 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):** This bit 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.

T-49-19-16

| Bit 4<br>OVIE | Description | |
|---|---|---|
| 0 | The free-running timer overflow interrupt request (FOVI) is disabled. | (Initial value) |
| 1 | The free-running timer overflow interrupt request (FOVI) is enabled. | |

**Bit 3—Output Enable B (OEB):** This bit selects whether to enable or disable output of the logic level selected by the OLVLB bit in the timer status/control register (TCSR) at the output compare B pin when the FRC and OCRB values match.

| Bit 3<br>OEB | Description | |
|---|---|---|
| 0 | Output compare B output is disabled. | (Initial value) |
| 1 | Output compare B output is enabled. | |

**Bit 2—Output Enable A (OEA):** This bit selects whether to enable or disable output of the logic level selected by the OLVLA bit in the timer status/control register (TCSR) at the output compare A pin when the FRC and OCRA values match.

| 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 | Internal clock source (ø/4) | (Initial value) |
| 0 | 1 | Internal clock source (ø/8) | |
| 1 | 0 | Internal clock source (ø/32) | |
| 1 | 1 | External clock source (counted on the rising edge)* | |

Note: * Output enable B (bit 3) must be cleared to 0.

### 10.2.5 Timer Control/Status Register (TCSR)—H'FF91, H'FFA1

T-49-19-16

| 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 |

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

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

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

**Bit 7—Input Capture Flag (ICF):** This status flag is set to 1 to indicate an input capture event. It signifies that the FRC value has been copied to the ICR.

| Bit 7<br>ICF | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when:<br>1. The CPU reads the ICF bit after the ICF bit has been set to 1, then writes a 0 in this bit.<br>2. The data transfer controller (DTC) serves an input capture interrupt . | (Initial value) |
| 1 | This bit is set to 1 when an input capture 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.

| Bit 6<br>OCFB | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when:<br>1. The CPU reads the OCFB bit after the OCFB bit has been set to 1, then writes a 0 in this bit.<br>2. The data transfer controller (DTC) serves output compare interrupt B. | (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.

T-49-19-16

**Bit 5**

| OCFA | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when:<br>1. The CPU reads the OCFA bit after the OCFA bit has been set to 1, then writes a 0 in this bit.<br>2. The data transfer controller (DTC) serves output compare interrupt A. | (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).

**Bit 4**

| OVF | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when the CPU reads the OVF bit after the OVF bit has been set to 1, then writes 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):** This bit selects the logic level to be output at the FTOB pin when the FRC and OCRB values match.

**Bit 3**

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

**Bit 2—Output Level A (OLVLA):** This bit selects the logic level to be output at the FTOA pin when the FRC and OCRA values match.

**Bit 2**

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

**Bit 1—Input Edge Select (IEDG):** This bit selects whether to capture the count on the rising or falling edge of the input capture signal.

**Bit 1**

| IEDG | Description | |
|---|---|---|
| 0 | The FRC value is copied to the ICR on the falling edge of the input capture signal. | (Initial value) |
| 1 | The FRC value is copied to the ICR on the rising edge of the input capture signal. | |

**Bit 0—Counter Clear A (CCLRA):** This bit 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. | |

## 10.3 CPU Interface

The FRC, OCRA, OCRB, and ICR are 16-bit registers, but they are connected to an 8-bit data bus. When the CPU accesses these four 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 upper 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.

Programs that access these four registers should normally use word access. Equivalently, they may access first the upper byte, then the lower byte. Data will not be transferred correctly if the bytes are accessed in reverse order, or if only one byte is accessed.

The same considerations apply to access by the DTC.

## Coding Examples

T-49-19-16

1. To write the contents of general register R0 to output compare register A in FRT1:

```
MOV.W   R0, @H'FF94
```

2. To read the FRT2 input capture register contents into general register R0:

```
MOV.W   @H'FFA8,  R0
```

Figure 10-2 shows the data flow when the FRC is accessed. The other registers are accessed in the same way, except that when OCRA or OCRB is read, the upper and lower bytes are both transferred directly to the CPU without using the temporary register.

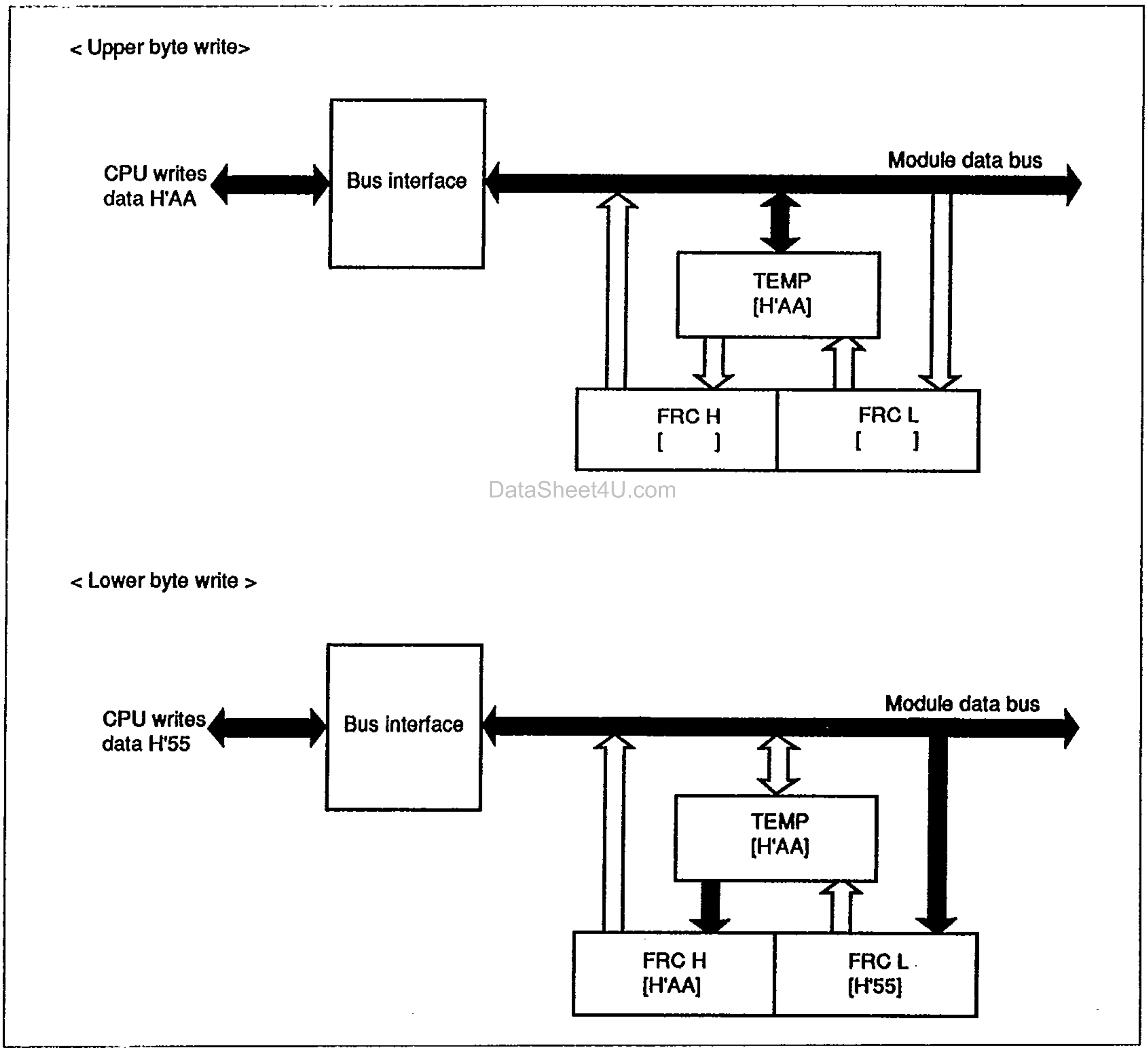


**Figure 10-2 (a) Write Access to FRC (When CPU Writes H'AA55)**

T-49-19-16

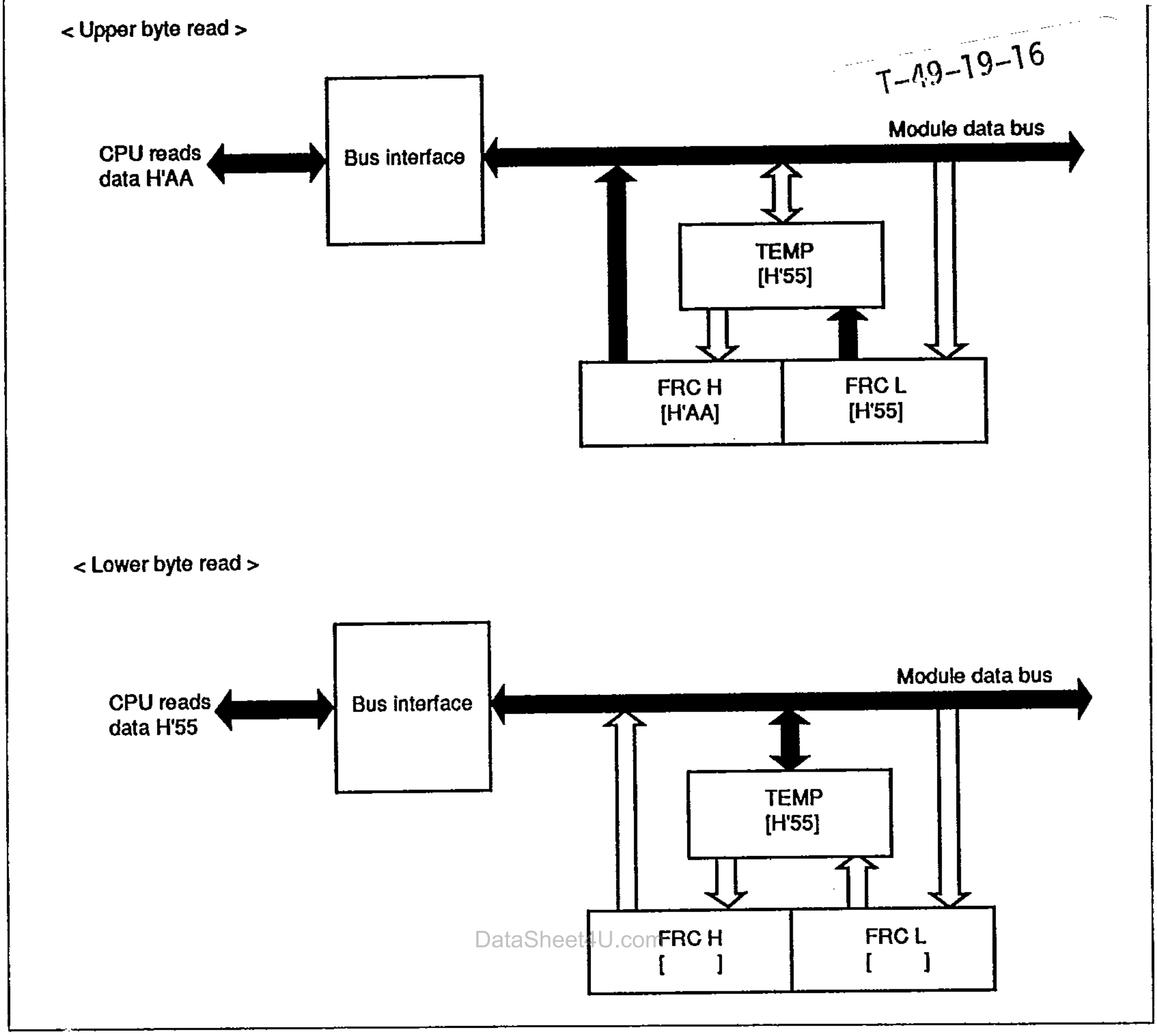


Figure 10-2 (b) Read Access to FRC (When FRC Contains H'AA55)

## 10.4 Operation

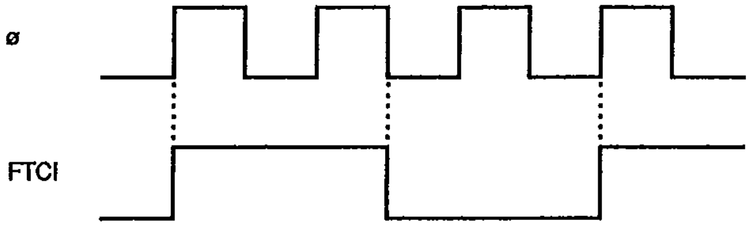
### 10.4.1 FRC Incrementation Timing

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

If external clock input is selected, the FRC increments on the rising edge of the clock signal. Figure 10-3 shows the increment timing.

T-49-19-16

The pulse width of the external clock signal must be at least 1.5 ø clock periods. The counter will not increment correctly if the pulse width is shorter than 1.5 ø clock periods.



Minimum FTCI Pulse Width

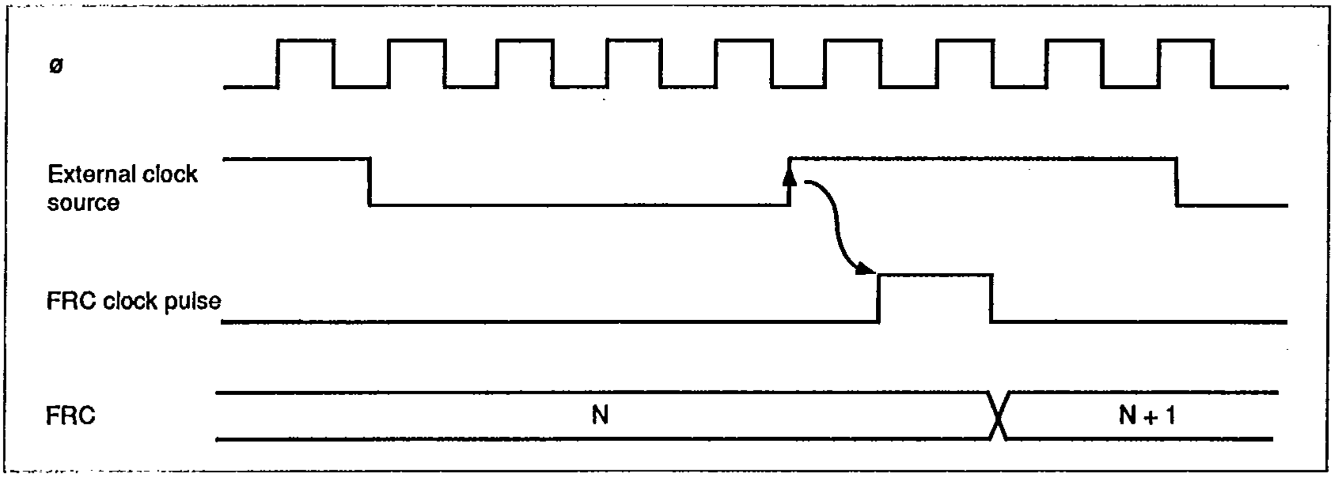


Figure 10-3 Increment Timing for External Clock Input

### 10.4.2 Output Compare Timing

**Setting of Output Compare Flags A and B (OCFA and OCFB):** The output compare flags are set to 1 by an internal compare-match signal generated when the FRC value matches the OCRA or OCRB value. This compare-match signal is generated at the last state in which the two values match, just as the FRC increments to a new value.

Accordingly, when the FRC and OCR values match, the compare-match signal is not generated until the next period of the clock source. Figure 10-4 shows the timing of the setting of the output compare flags.

T-49-19-16

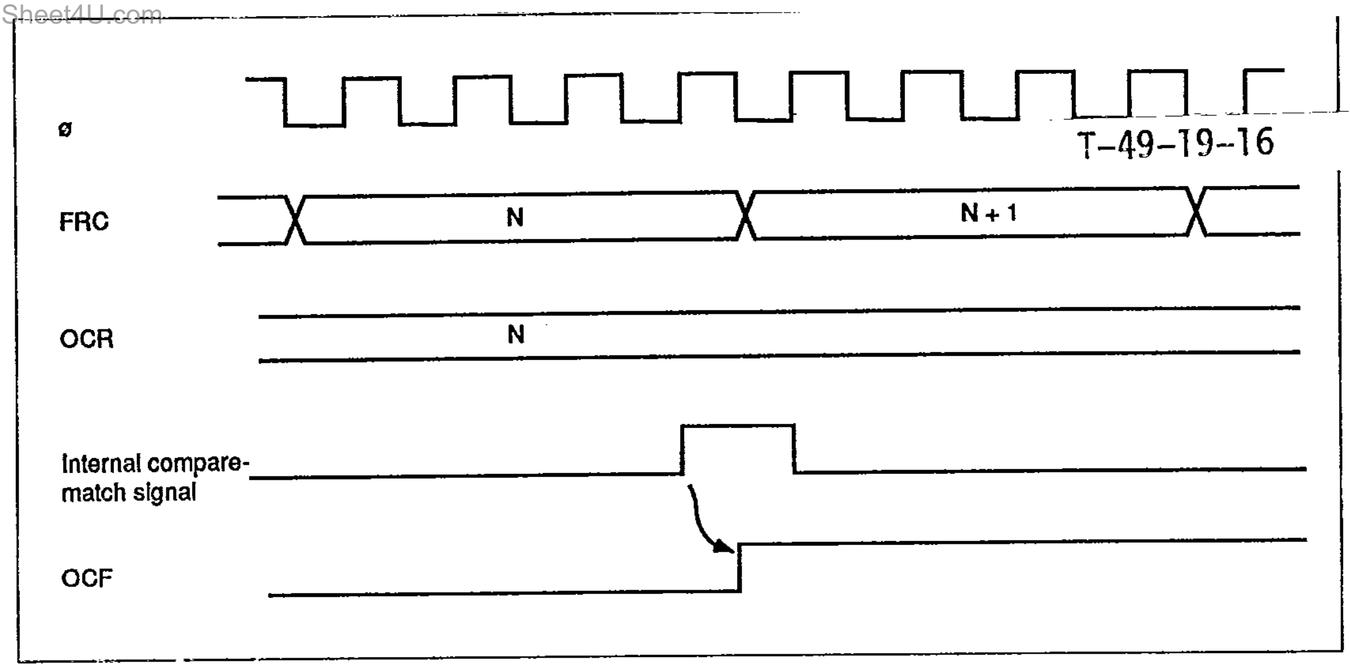


Figure 10-4 Setting of Output Compare Flags

**Output 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 10-5 shows the timing of this operation for compare-match A.

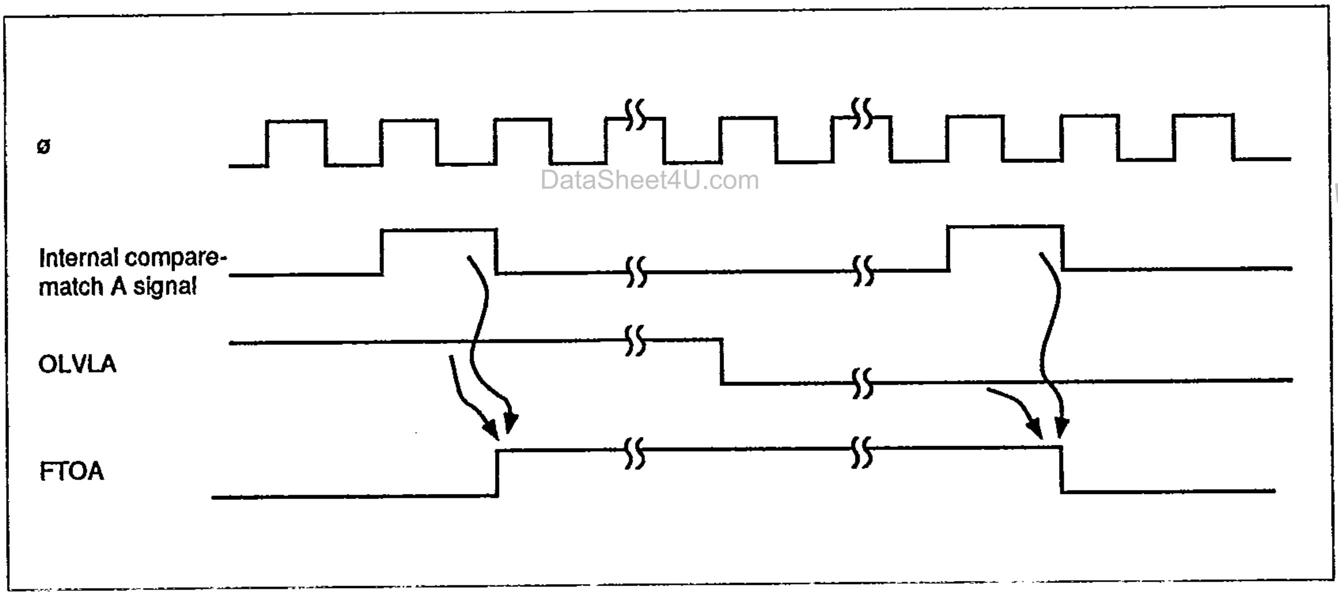


Figure 10-5 Timing of Output Compare A

**FRC Clear Timing:** If the CCLRA bit is set to 1, the FRC is cleared when compare-match A occurs. Figure 10-6 shows the timing of this operation.

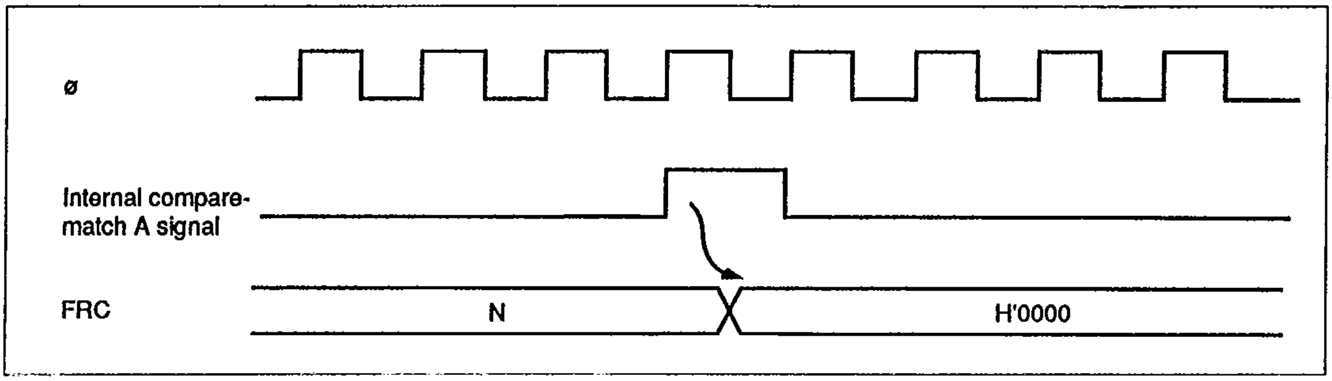


Figure 10-6 Clearing of FRC by Compare-Match A

### 10.4.3 Input Capture Timing

**Input Capture Timing:** An internal input capture signal is generated from the rising or falling edge of the input at the input capture pin (FTI), as selected by the IEDG bit in the TCSR. Figure 10-7 shows the usual input capture timing when the rising edge is selected (IEDG = 1).

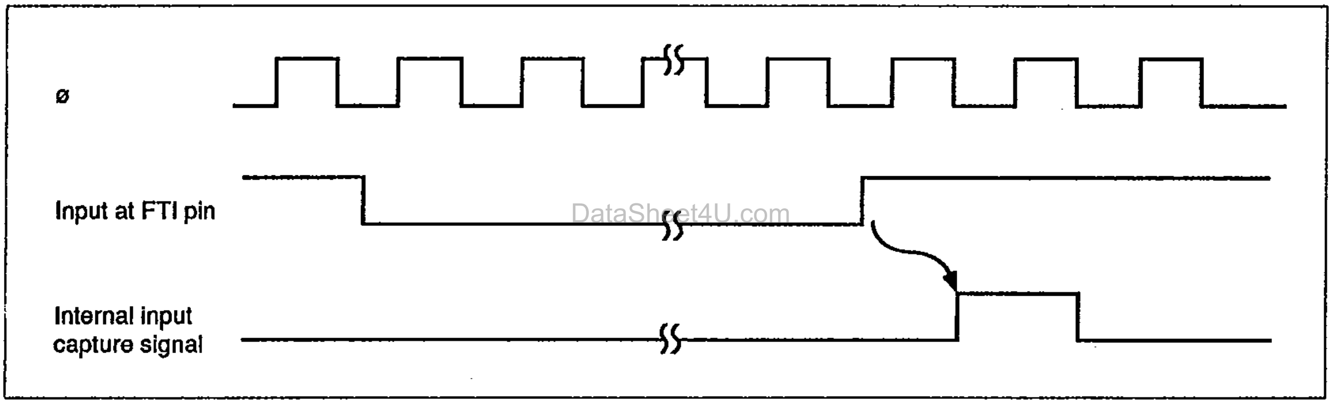
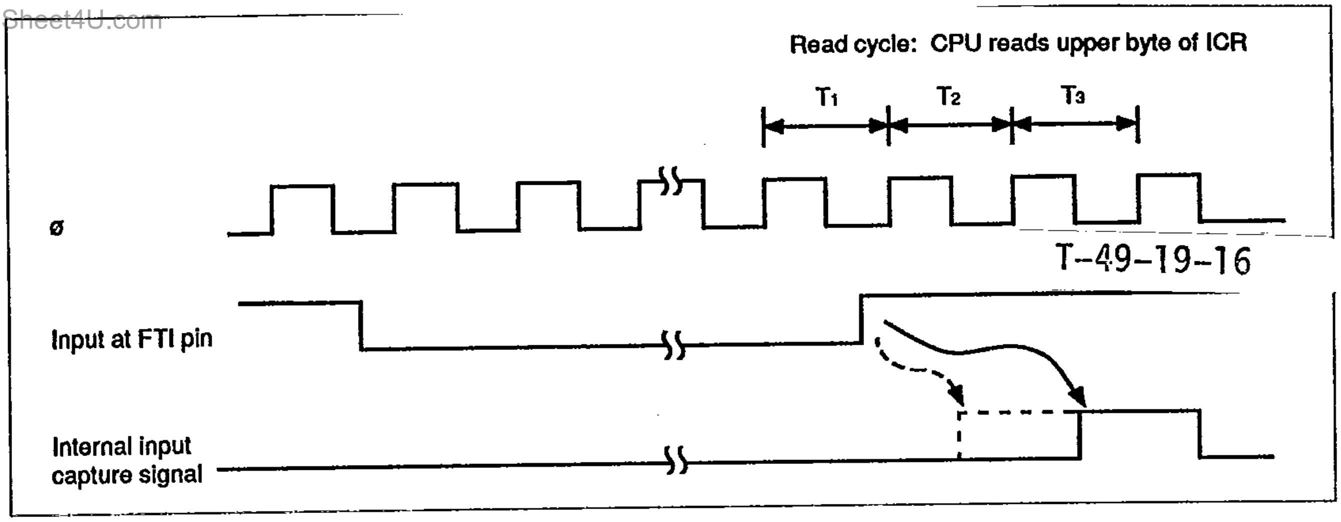


Figure 10-7 Input Capture Timing (Usual Case)

But if the upper byte of the ICR is being read when the input capture signal arrives, the internal input capture signal is delayed by one state. Figure 10-8 shows the timing for this case.



T-49-19-16

Figure 10-8 Input Capture Timing (1-State Delay)

**Timing of Input Capture Flag (ICF) Setting:** The input capture flag (ICF) is set to 1 by the internal input capture signal. Figure 10-9 shows the timing of this operation.

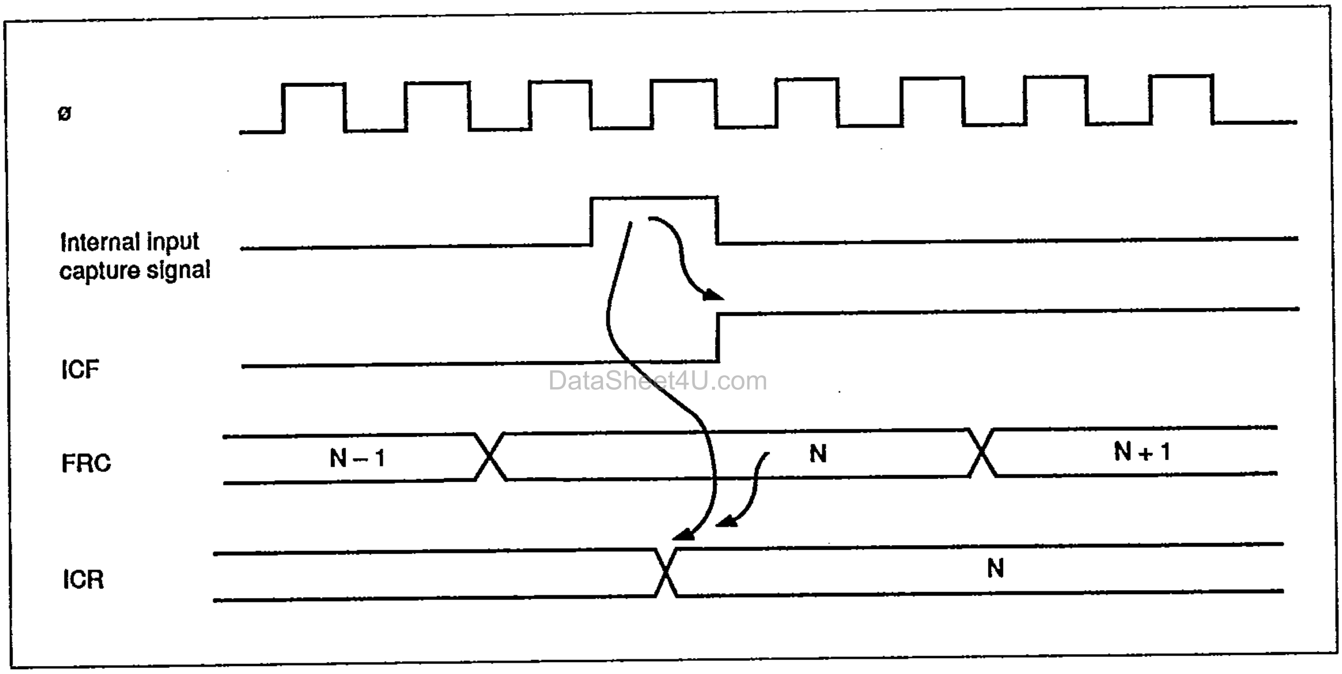


Figure 10-9 Setting of Input Capture Flag

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

The FRC overflow flag (OVF) is set to 1 when the FRC overflows (changes from H'FFFF to H'0000). Figure 10-10 shows the timing of this operation.

T-49-19-16

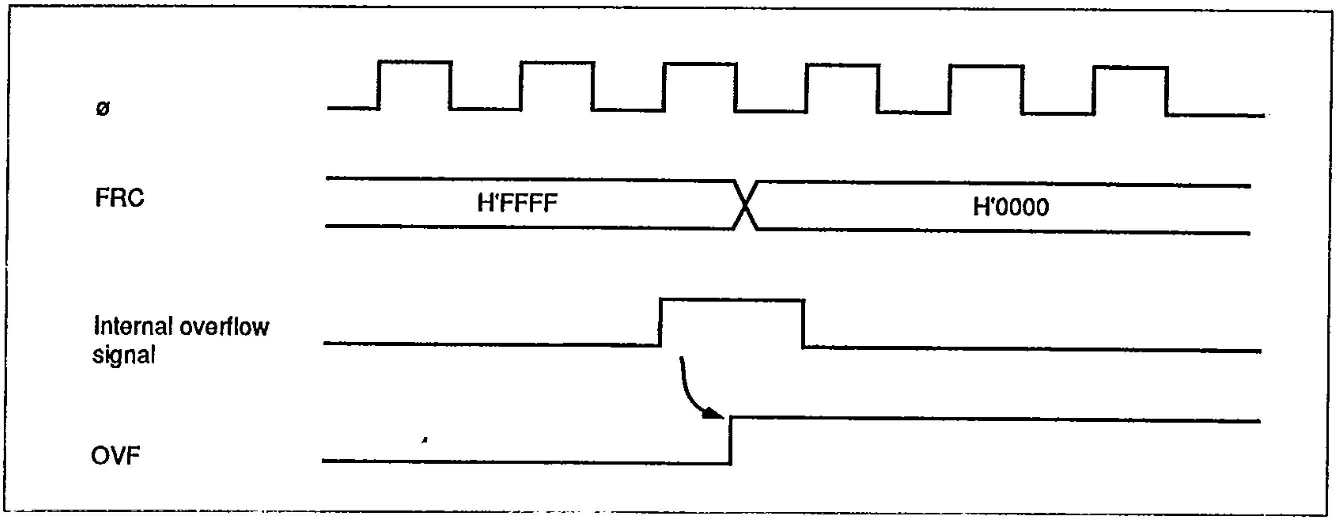


Figure 10-10 Setting of Overflow Flag (OVF)

## 10.5 CPU Interrupts and DTC Interrupts

Each 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 enable and flag bits are set. Independent signals are sent to the interrupt controller for each type of interrupt. Table 10-3 lists information about these interrupts.

Table 10-3 Free-Running Timer Interrupts

| Interrupt | Description | DTC Service Available? | Priority |
|---|---|---|---|
| ICI | Requested when ICF is set | Yes | High |
| OCIA | Requested when OCFA is set | Yes | ↑ |
| OCIB | Requested when OCFB is set | Yes | |
| FOVI | Requested when OVF is set | No | Low |

The ICI, OCIA, and OCIB interrupts can be directed to the data transfer controller (DTC) to have a data transfer performed in place of the usual interrupt-handling routine.

When the DTC serves one of these interrupts, it automatically clears the ICF, OCFA, or OCFB flag to 0. See section 6, "Data Transfer Controller", for further information on the DTC.

## 10.6 Synchronization of Free-Running Timers 1 and 2

T-49-19-16

### 10.6.1 Synchronization after a Reset

The two free-running timer channels are synchronized at a reset and remain synchronized until one of the following conditions is satisfied:

- The clock source is changed.
- FRC contents are rewritten.
- An FRC is cleared.

After a reset, each free-running counter operates on the ø/4 internal clock source.

### 10.6.2 Synchronization by Writing to FRCs

When synchronization between free-running timers 1 and 2 is lost, it can be restored by writing to the free-running counters.

**Synchronization on Internal Clock Source:** When an internal clock is selected, free-running timers 1 and 2 can be synchronized by writing data to their free-running counters as indicated in table 10-4.

Table 10-4 Synchronization by Writing to FRCs

| Clock Source | Write Interval | Write Data |
|---|---|---|
| ø/4 | 4n (states) | m (FRC1) |
| ø/8 | 8n (states) | m + n (FRC2) |
| ø/32 | 32n (states) | |

m, n: Arbitrary integers

After writing these data, synchronization can be checked by reading the two free-running counters at the same interval as the write interval. If the read data have the same relative difference as the write data, the free-running timers are synchronized.

Examples of programs for synchronizing the free-running timers are given next. Examples a, b, and c apply when the program is stored in on-chip memory. Examples d, e, and f apply when the program is stored in external memory which is accessed with zero wait states (Tw), assuming that there is no NMI input.

**Example a:** ø/4 clock source, 12-state write interval (n = 3), on-chip memory

T-49-19-16

```
LA:     LDC.B    #H'FF,BR          ; Initialize base register for short-format instruction (MOV:S)
        LDC.W    #H'0700,SR        ; Raise interrupt mask level to 7
        MOV.W    #m,R1             ; Data for free-running timer 1
        MOV.W    #m+3,R2           ; Data for free-running timer 2 (m + n = m + 3)
        BSR      SET4              ; Call write routine
              :
              :
              :
        .ALIGN   2                 ; Align write instructions (MOV:S) at even address
SET4:   MOV:S.W R1,@H'92:8         ; Write to FRC 1 (address H'FF92) 9 states ─┐
        BRN SET4:8                 ; 2-Byte dummy instruction        3 states ─┘─┐
        MOV:S.W R2,@H'A2:8         ; Write to FRC2 (address H'FFA2)        Total 12 states
        RTS
```

**Example b:** ø/8 clock source, 16-state write interval (n = 2), on-chip memory

```
LB:     LDC.B    #H'FF,BR
        LDC.W    #H'0700,SR
        MOV.W    #m,R1
        MOV.W    #m+2,R2
        BSR      SET8
              :
              :
              :
        .ALIGN   2
SET8:   MOV:S.W R1,@H'92:8         ; 9 states ─┐
        BRN SET8:8                 ; 3 states ─┼──────────── Total 16 states
        XCH R1,R1                  ; 4 states ─┘
        MOV:S.W R2,@H'A2:8
        RTS
```

**Example c:** ø/32 clock source, 32-state write interval (n = 1), on-chip memory

```
LC:      LDC.B  #H'FF,BR
         LDC.W  #H'0700,SR
         MOV.W  #m,R1
         MOV.W  #m+1,R2
         BSR   SET32
           :
           :
         .ALIGN   2                  ; Align on even address
SET32:   MOV:S.W R1,@H'92:8          ; 2 bytes,    9 states
         BSR   WAIT:8                ; 2 bytes,    9 states
         MOV:S.W   R2,@H'A2:8
         RTS


         .ALIGN   2                  ; Align on even address
WAIT:    NOP                         ; 2 states
         XCH   R1,R1                 ; 4 states
         RTS                         ; 8 states
```

Total 32 states

Note: The stack is assumed to be in on-chip RAM.

**Example d:** ø/4 clock source, 20-state write interval (n = 5), external memory

```
LD:     LDC.B    #H'FF,BR
        LDC.W    #H'0700,SR        ; Set interrupt mask level to 7
        CLR.B    @H'F8:8           ; Disable wait states
        MOV.W    #m,R1
        MOV.W    #m+5,R2
        MOV:S.W     R1,@H'92:8     ; 13 states
        BRN   LD:8                 ; 2 bytes, 7 states
        MOV:S.W     R2,@H'A2:8
```

Total 20 states

**Example e:** ø/8 clock source, 24-state write interval (n = 3), external memory

```
LE:    LDC.B     #H'FF,BR
       LDC.W     #H'0700,SR
       CLR.B     @H'F8:8
       MOV.W     #m,R1
       MOV.W     #m+3,R2
       MOV:S.W     R1,@H'92:8    ; 13 states ─────────────┐
       BRN    LE:8               ; 2 bytes,         7 states ──┼── Total 24 states
       NOP                       ; 1 byte,          4 states ──┘
       MOV:S.W R2,@H'A2;8
```

**Example f:** ø/32 clock source, 32-state write interval (n = 1), external memory

```
LF:    LDC.B     #H'FF,BR
       LDC.W     #H'0700,SR
       CLR.B     @H'F8:8
       MOV.W     #m,R1
       MOV.W     #m+1,R2
       MOV:S.W     R1,@H'92:8    ; External memory, so  13 states ─┐
       XCH    R0,R0              ;                      8 states ─┤
       BRN    LF:8               ; 2 bytes,             7 states ─┼── Total 32 states
       NOP                       ;                      4 states ─┘
       MOV:S.W R2,@H'A2:8
```

**Synchronization on External Clock Source:** When the external clock source is selected, the free-running timers can be synchronized by halting their external clock inputs, then writing identical values in their free-running counters.

## 10.7 Sample Application

In the example below, one 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.

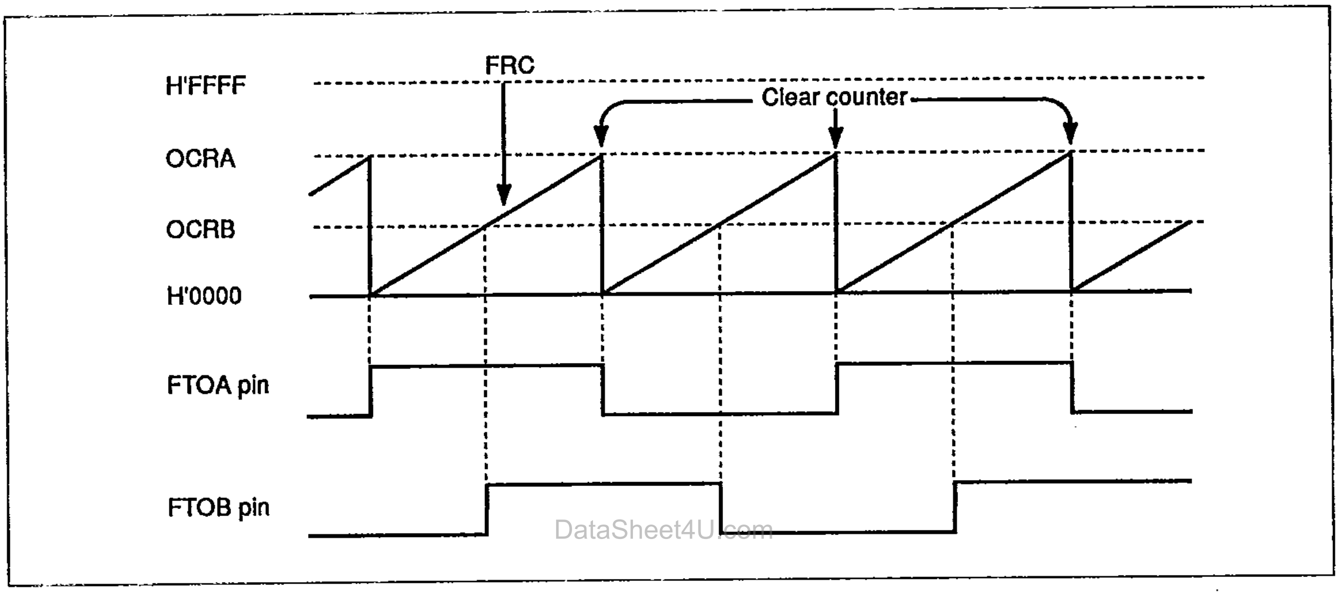


Figure 10-11 Square-Wave Output (Example)

## 10.8 Application Notes

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

**Contention between FRC Write and Clear:** If an internal counter clear signal is generated during the T3 state of a write cycle to the lower byte of a free-running counter, the clear signal takes priority and the write is not performed.

Figure 10-12 shows this type of contention.

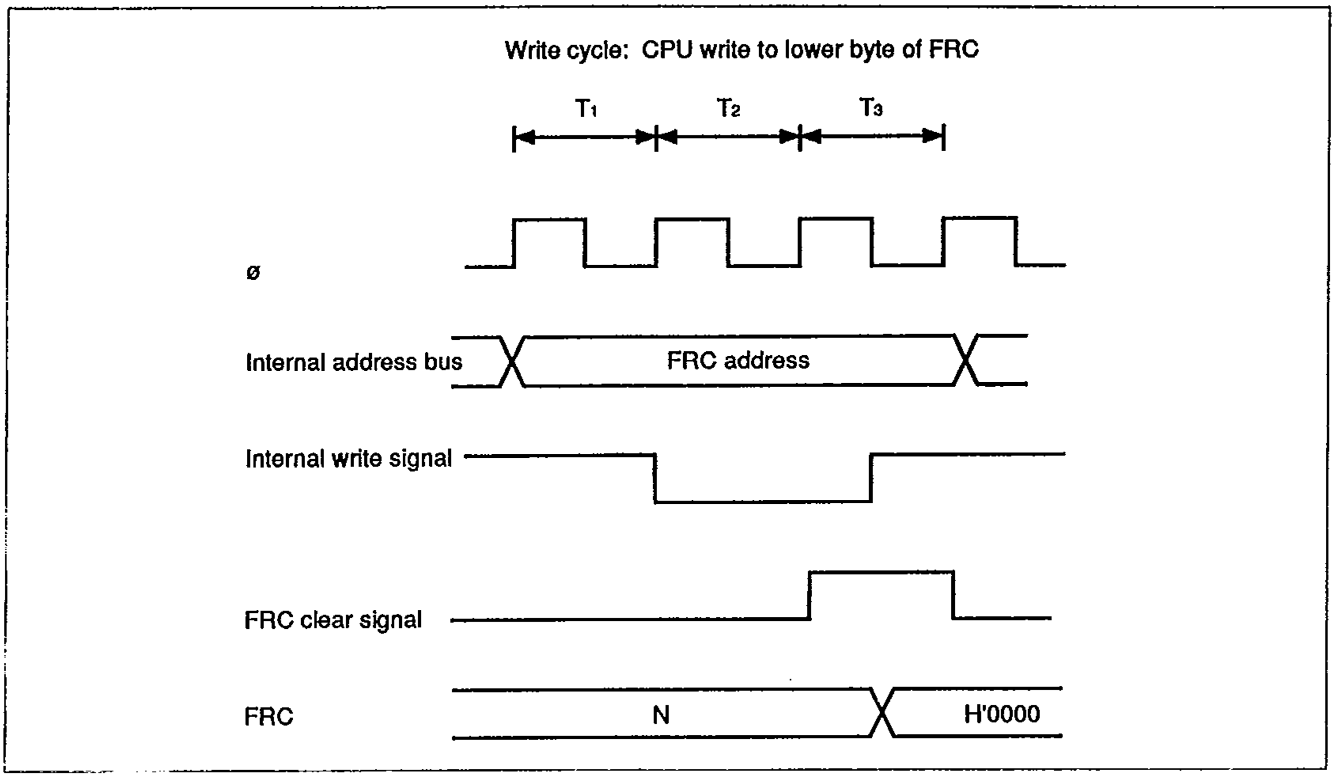


Figure 10-12 FRC Write-Clear Contention

**Contention between FRC Write and Increment:** If an FRC increment pulse is generated during the T3 state of a write cycle to the lower byte of a free-running counter, the write takes priority and the FRC is not incremented.

Figure 10-13 shows this type of contention.

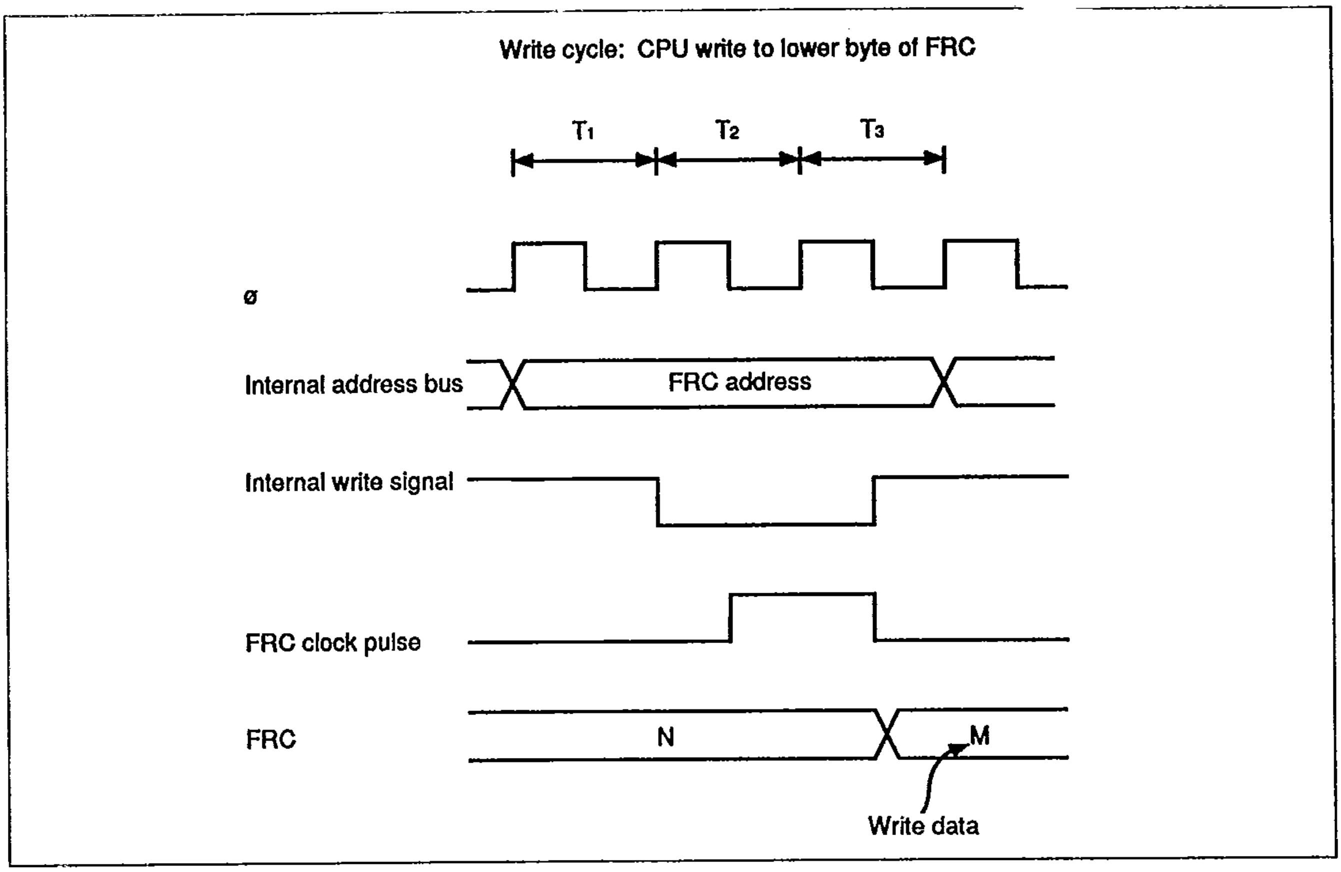


Figure 10-13 FRC Write-Increment Contention

**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.

T-49-19-16

Figure 10-14 shows this type of contention.

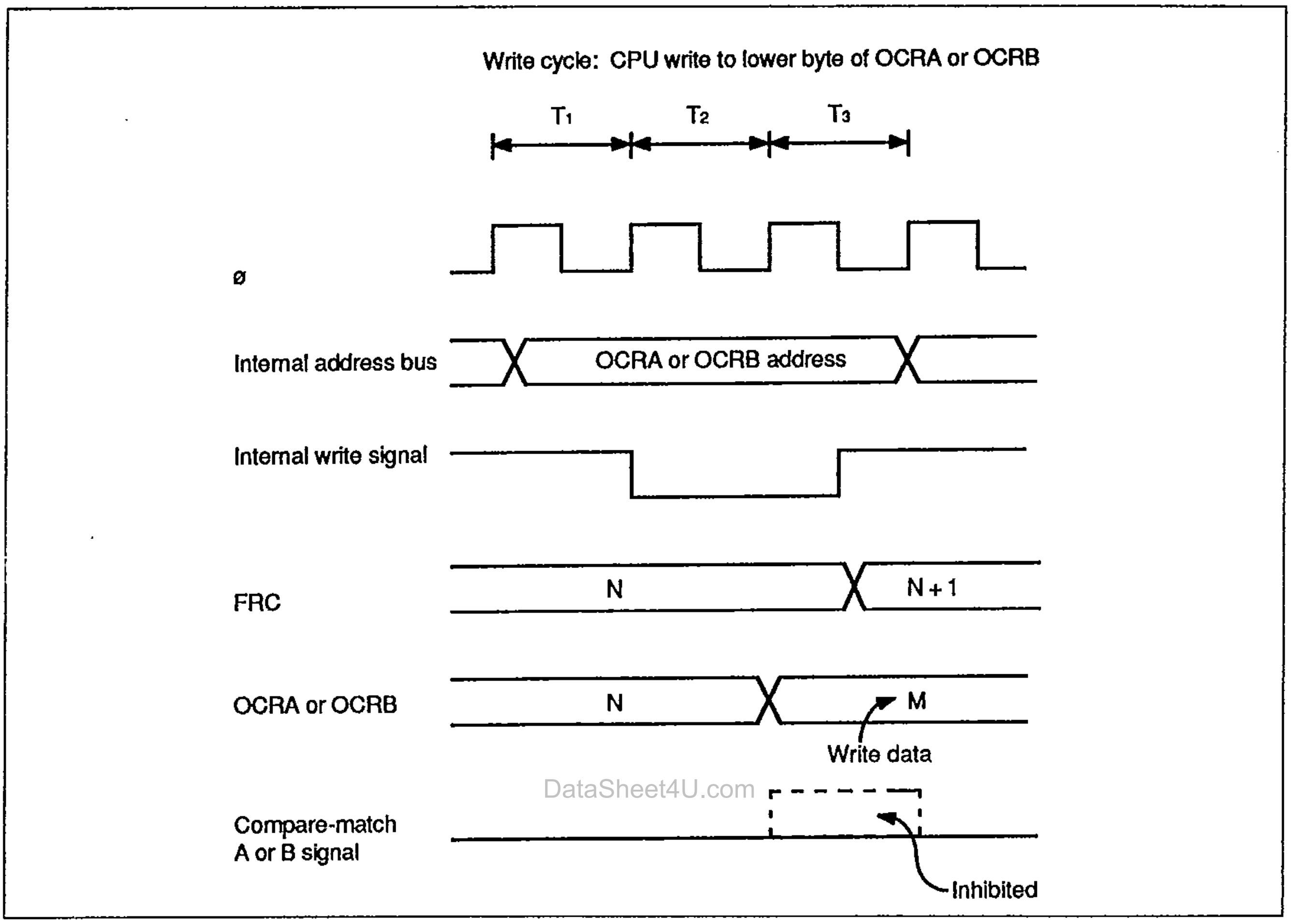


Figure 10-14 Contention between OCR Write and Compare-Match

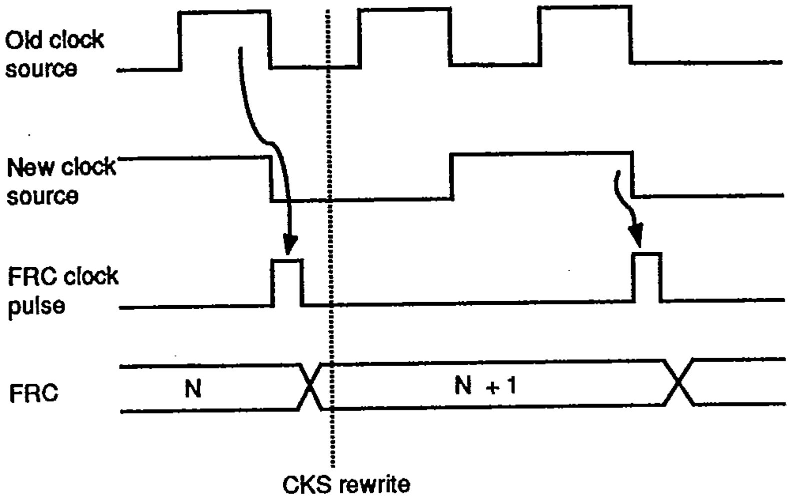
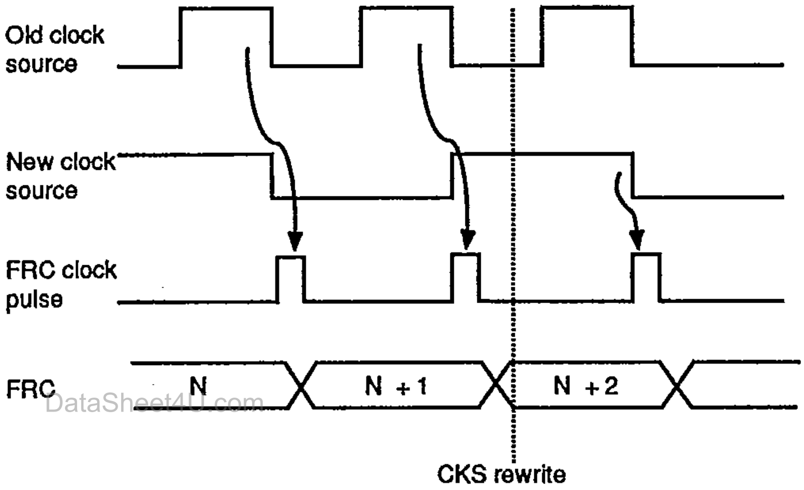
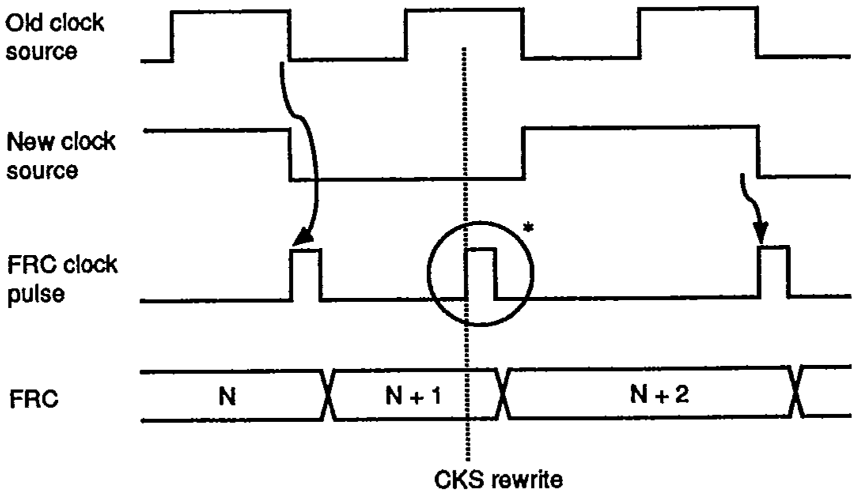
**Incrementation Caused by Changing of Internal Clock Source:** When an internal clock source is changed, the changeover may cause the FRC to increment. This depends on the time at which the clock select bits (CKS1 and CKS0) are rewritten, as shown in table 10-5.

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 10-5, the changeover generates a falling edge that triggers the FRC increment pulse.

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

## Table 10-5 Effect of Changing Internal Clock Sources

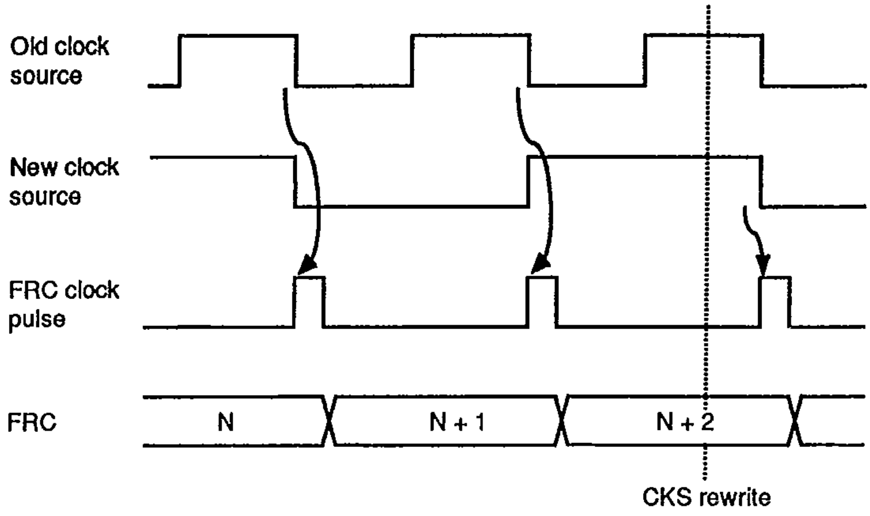
T-49-19-16

| 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. |  |

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

**Table 10-5 Effect of Changing Internal Clock Sources (cont)**

T-49-19-16

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

# Section 11 8-Bit Timer

T-49-19-16

## 11.1 Overview

The H8/520 chip includes a single 8-bit timer based on an 8-bit counter (TCNT). The timer has 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 is to generate a rectangular-wave output with an arbitrary duty factor.

### 11.1.1 Features

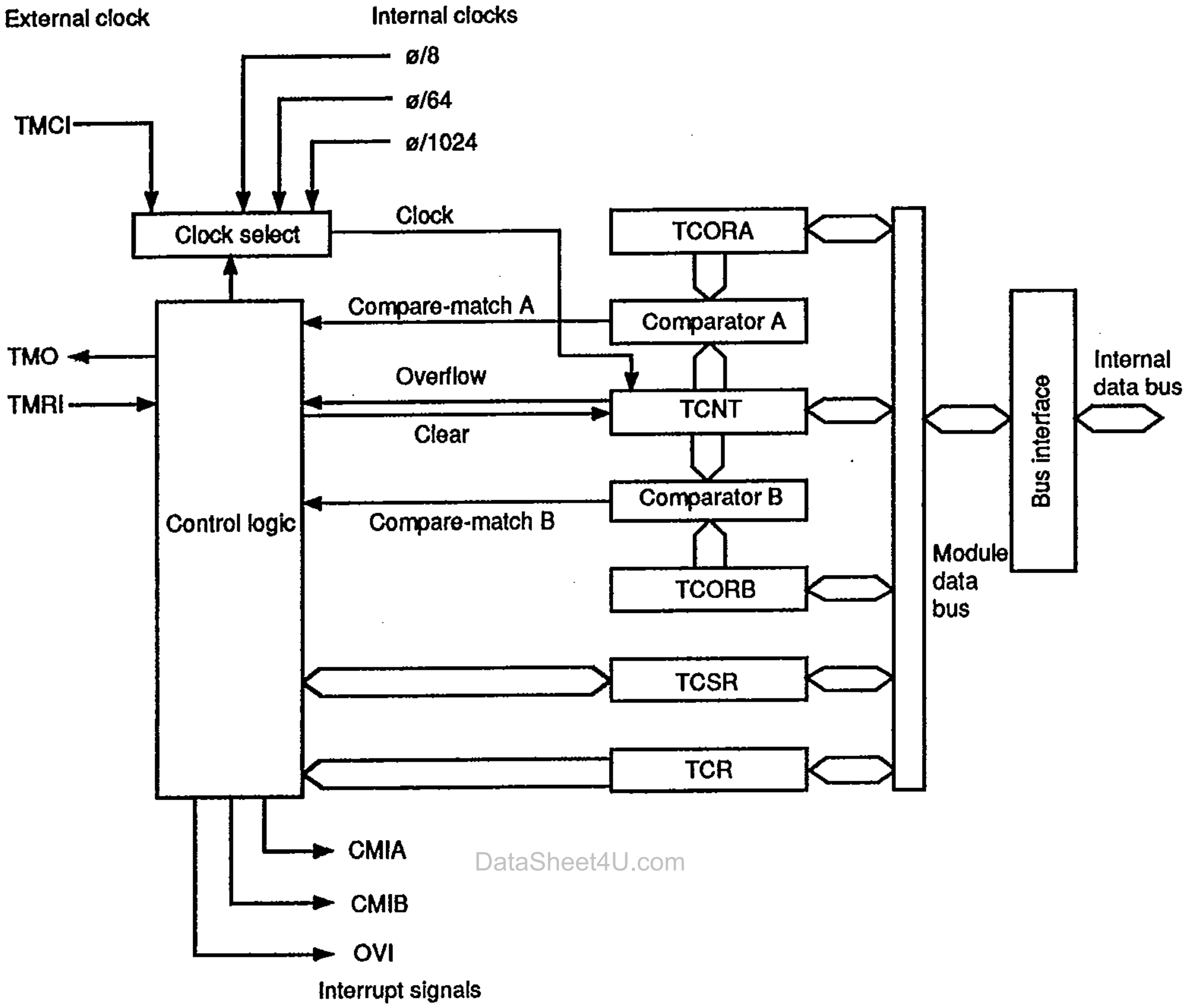
The features of the 8-bit timer are listed below.

- Selection of four clock sources
  The counter 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 counter
  The counter can be cleared on compare-match A or B, or by an external reset signal.
- Timer output controlled by two compare-match signals
  The single timer output (TMO) is controlled by two independent compare-match signals, enabling the timer to generate output waveforms with an arbitrary duty factor.
- Three types of interrupts
  Compare-match A and B and overflow interrupts can be requested independently.
  The compare match interrupts can be served by the data transfer controller (DTC), enabling interrupt-driven data transfer with minimal CPU programming.

### 11.1.2 Block Diagram

T-49-19-16

Figure 11-1 shows a block diagram of the 8-bit timer.



TCORA: Time Constant Register A
TCORB: Time Constant Register B
TCNT: Timer Counter
TCSR: Timer Control/Status Register
TCR: Timer Control Register

Figure 11-1 Block Diagram of 8-Bit Timer

T-49-19-16

### 11.1.3 Input and Output Pins

Table 11-1 lists the input and output pins of the 8-bit timer.

**Table 11-1 Input and Output Pins of 8-Bit Timer**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Timer output | TMO | Output | Output controlled by compare-match |
| Timer clock input | TMCI | Input | External clock source for the counter |
| Timer reset input | TMRI | Input | External reset signal for the counter |

### 11.1.4 Register Configuration

Table 11-2 lists the registers of the 8-bit timer.

**Table 11-2 8-Bit Timer Registers**

| Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| Timer control register | TCR | R/W | H'00 | H'FFD0 |
| Timer control/status register | TCSR | R/(W)* | H'10 | H'FFD1 |
| Timer constant register A | TCORA | R/W | H'FF | H'FFD2 |
| Timer constant register B | TCORB | R/W | H'FF | H'FFD3 |
| Timer counter | TCNT | R/W | H'00 | H'FFD4 |

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

## 11.2 Register Descriptions

### 11.2.1 Timer Counter (TCNT)—H'FFD4

| 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 |

The 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. Clock clear bits 1 and 0 (CCLR1 and CCLR0) of the timer control register select the method of clearing.

T-49-19-16

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

### 11.2.2 Time Constant Registers A and B (TCORA and TCORB)—H'FFD2 and H'FFD3

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

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

### 11.2.3 Timer Control Register (TCR)—H'FFD0

| 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 |

The 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.

The 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 status/control register (TCSR) is set to 1.

T-49-19-16

| 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 status/control 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 status/control 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.

T-49-19-16

| Bit 2<br>CKS2 | Bit 1<br>CKS1 | Bit 0<br>CKS0 | Description | |
|---|---|---|---|---|
| 0 | 0 | 0 | No clock source (timer stopped). | (Initial value) |
| 0 | 0 | 1 | Internal clock source (ø/8). | |
| 0 | 1 | 0 | Internal clock source (ø/64). | |
| 0 | 1 | 1 | Internal clock source (ø/1024). | |
| 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. | |

### 11.2.4 Timer Control/Status Register (TCSR)

| 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 |

The 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 (TMO).

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

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

**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.

T-49-19-16

| Bit 7 CMFB | Description |
|---|---|
| 0 | This bit is cleared from 1 to 0 when: (Initial value)<br>1. The CPU reads the CMFB bit after the CMFB bit has been set to 1, then writes a 0 in this bit.<br>2. Compare-match interrupt B is served by the data transfer controller (DTC). |
| 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.

| Bit 6 CMFA | Description |
|---|---|
| 0 | This bit is cleared from 1 to 0 when: (Initial value)<br>1. The CPU reads the CMFA bit after the CMFA bit has been set to 1, then writes a 0 in this bit.<br>2. Compare-match interrupt A is served by the data transfer controller (DTC). |
| 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).

| Bit 5 OVF | Description |
|---|---|
| 0 | This bit is cleared from 1 to 0 when the CPU reads the OVF bit after the OVF bit has been set to 1, then writes 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 cannot be modified and is always read as 1.

**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 (TMO). 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.

When all four output select bits are cleared to 0 the TMO signal is not output.

After a reset, the TMO output is low (0) until the first compare-match event.

| Bit 3<br>OS3 | Bit 2<br>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<br>OS1 | Bit 0<br>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. | |

## 11.3 Operation

### 11.3.1 TCNT Incrementation Timing

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

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.

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

T-49-19-16

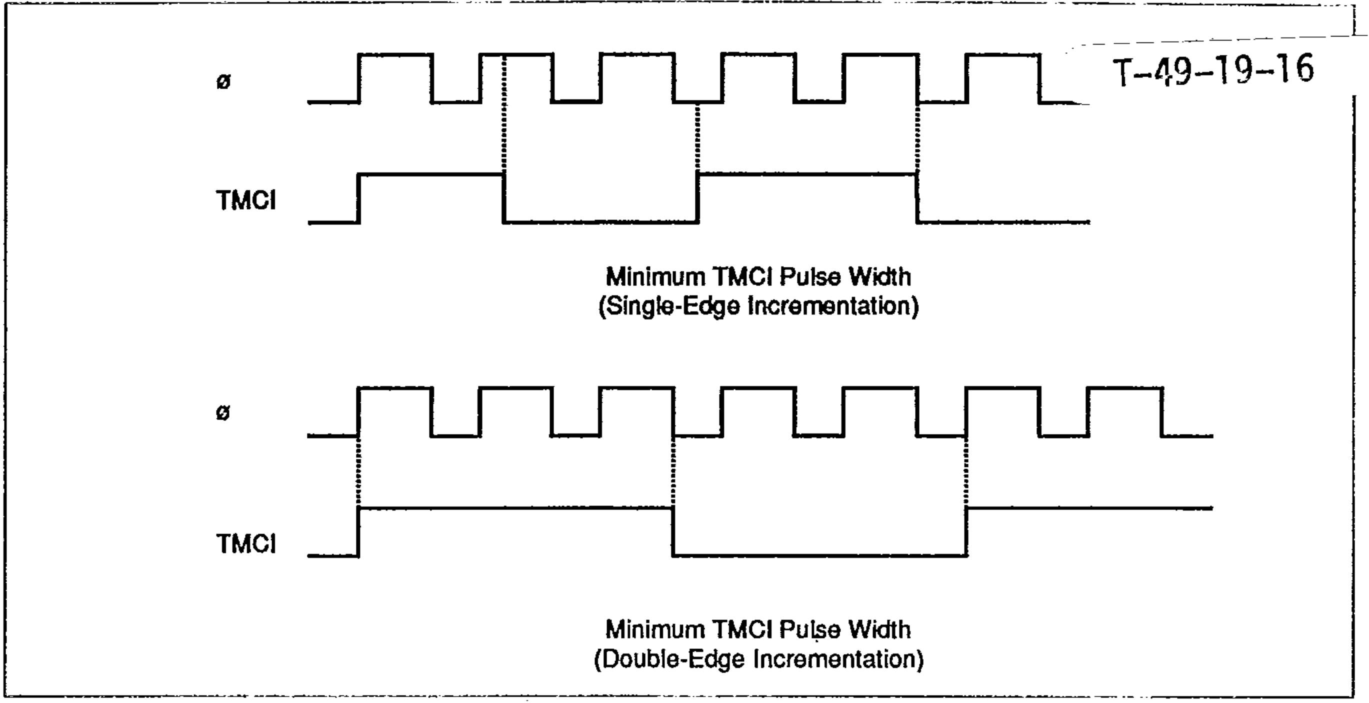


Figure 11-2 shows the timing of incrementation on both edges of an external clock signal.

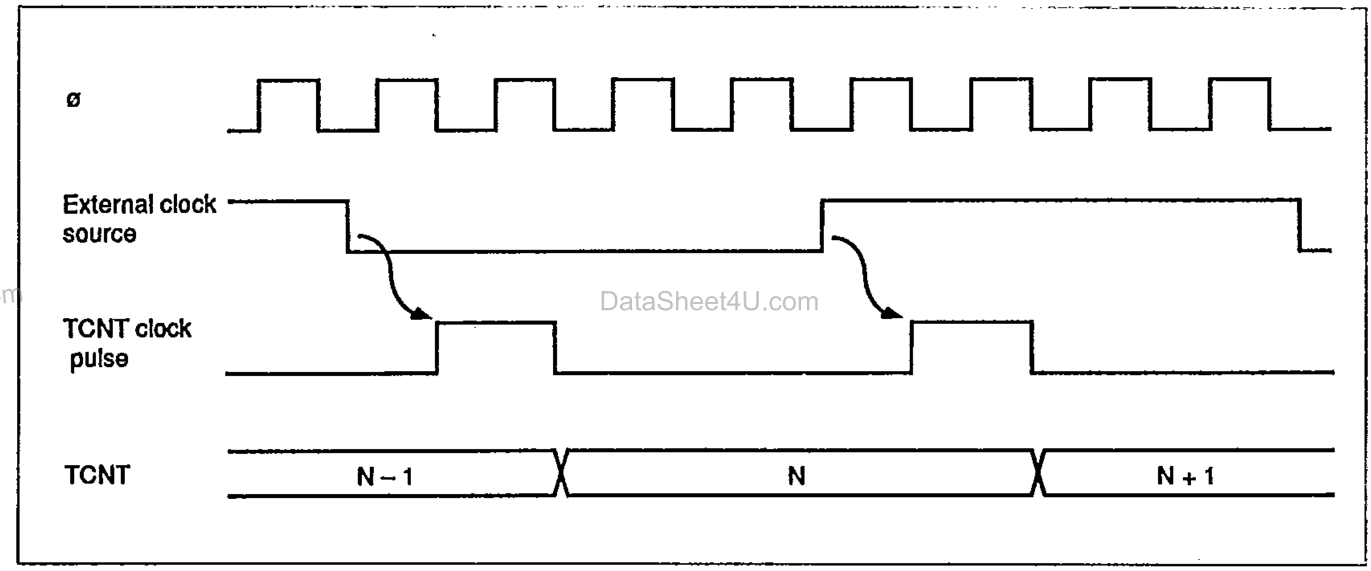


**Figure 11-2 Count Timing for External Clock Input**

### 11.3.2 Compare Match Timing

**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 TCORA or TCORB. The compare-match signal is generated at the last state in which the match is true, just as 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 11-3 shows the timing of the setting of the compare-match flags.

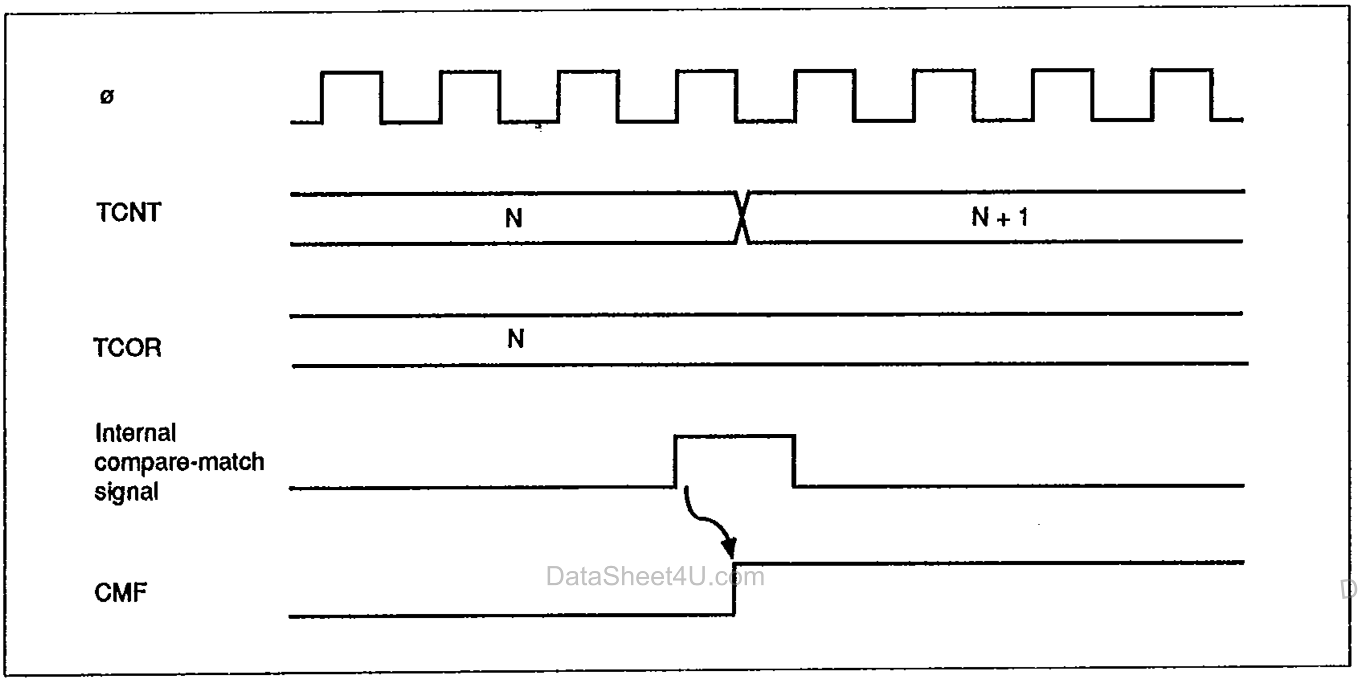


Figure 11-3 Setting of Compare-Match Flags

**Output Timing:** When a compare-match event occurs, the timer output (TMO) 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.

Figure 11-4 shows the timing when the output is set to toggle on compare-match A.

T-49-19-16

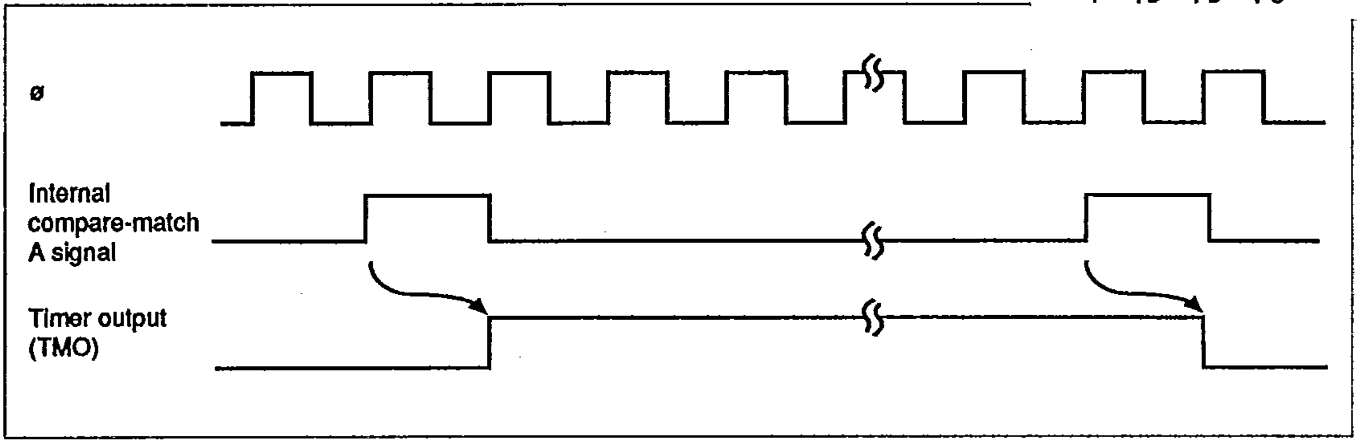


Figure 11-4 Timing of Timer Output

### 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 11-5 shows the timing of this operation.

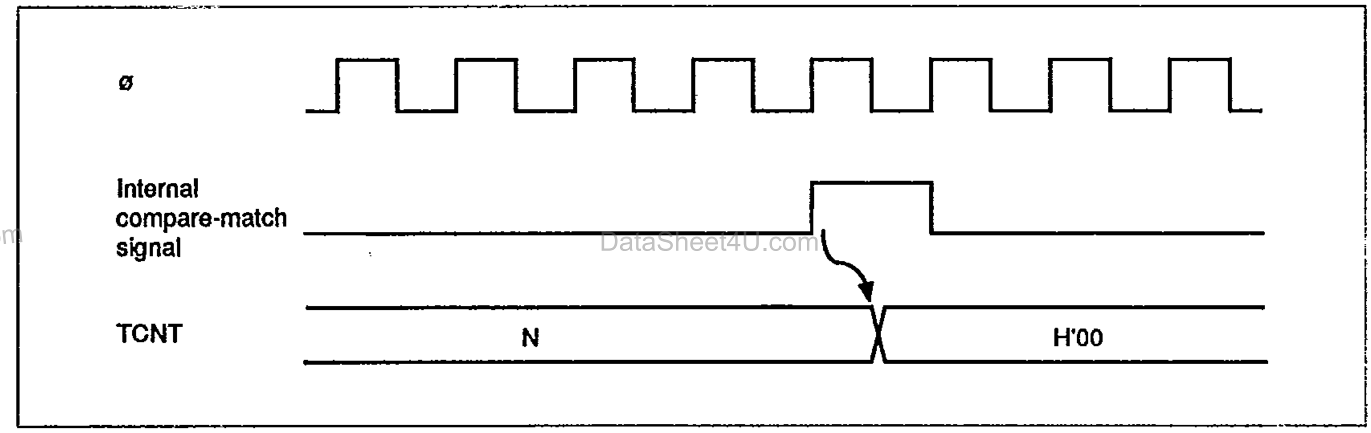


Figure 11-5 Timing of Compare-Match Clear

### 11.3.3 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 (TMRI). To ensure resetting, the TMRI pulse width must be at least 1.5 ø clock periods. Figure 11-6 shows the timing of this operation.

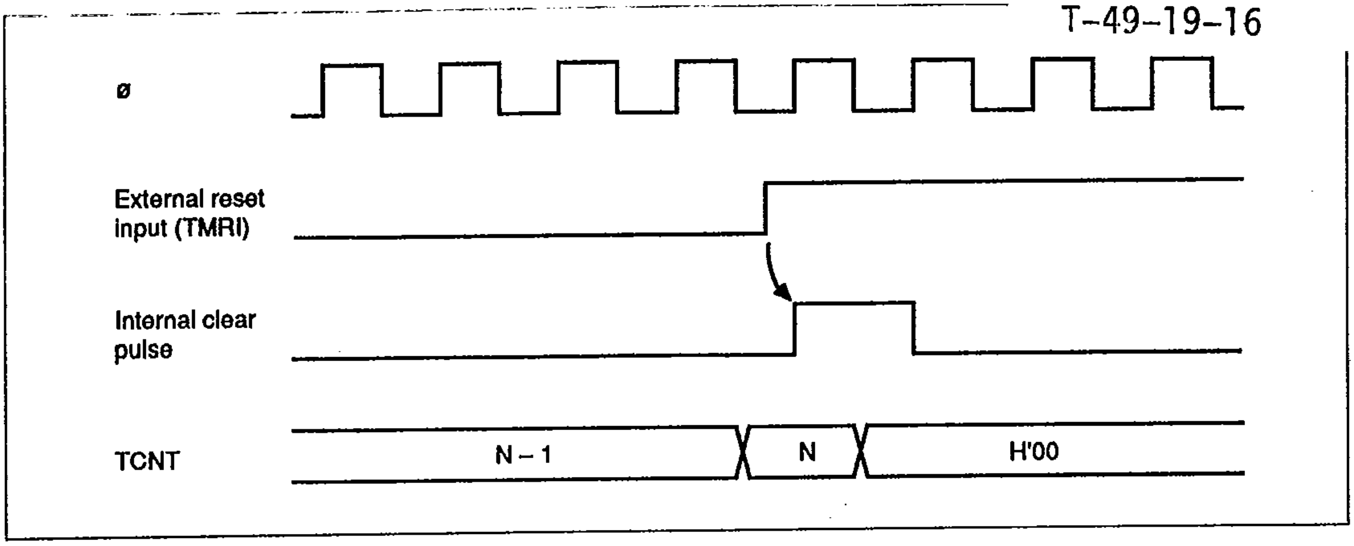


Figure 11-6 Timing of External Reset

### 11.3.4 Setting of TCNT Overflow Flag

The overflow flag (OVF) is set to 1 when the timer count overflows (changes from H'FF to H'00). Figure 11-7 shows the timing of this operation.

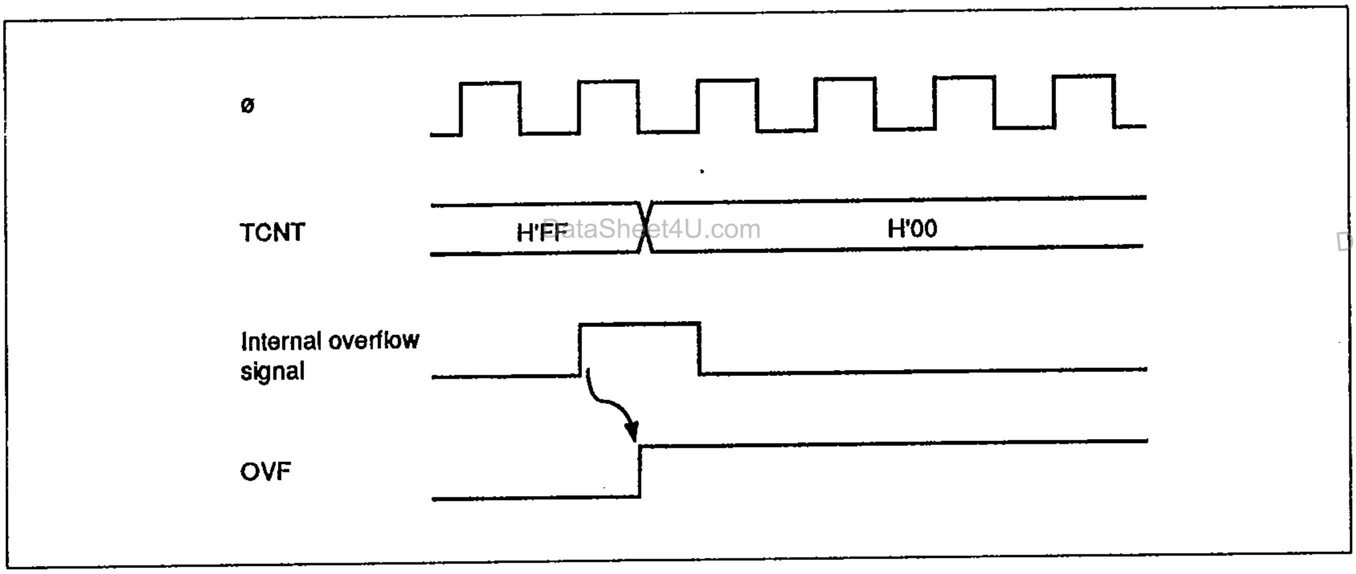


Figure 11-7 Setting of Overflow Flag (OVF)

## 11.4 CPU Interrupts and DTC Interrupts

T-49-19-16

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 and flag bits are set in the TCR and TCSR. Independent signals are sent to the interrupt controller for each type of interrupt. Table 11-3 lists information about these interrupts.

**Table 11-3 8-Bit Timer Interrupts**

| Interrupt | Description | DTC Service Available? | Priority |
|---|---|---|---|
| CMIA | Requested when CMFA is set | Yes | High |
| CMIB | Requested when CMFB is set | Yes | ↑ |
| OVI | Requested when OVF is set | No | Low |

The CMIA and CMIB interrupts can be served by the data transfer controller (DTC) to have a data transfer performed.

When the DTC serves one of these interrupts, it automatically clears the CMFA or CMFB flag to 0. See section 6, "Data Transfer Controller", for further information on the DTC.

## 11.5 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.

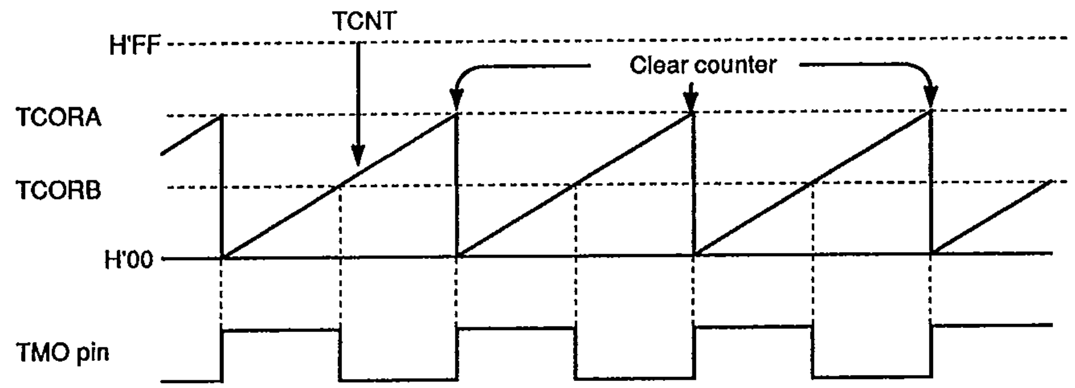


Figure 11-8 Example of Pulse Output

## 11.6 Application Notes

Application programmers should note that the following types of contention can occur in the 8-bit timer.

T-49-19-16

**Contention between TCNT Write and Clear:** If an internal counter clear signal is generated during the T3 state of a write cycle to the timer counter, the clear signal takes priority and the write is not performed.

Figure 11-9 shows this type of contention.

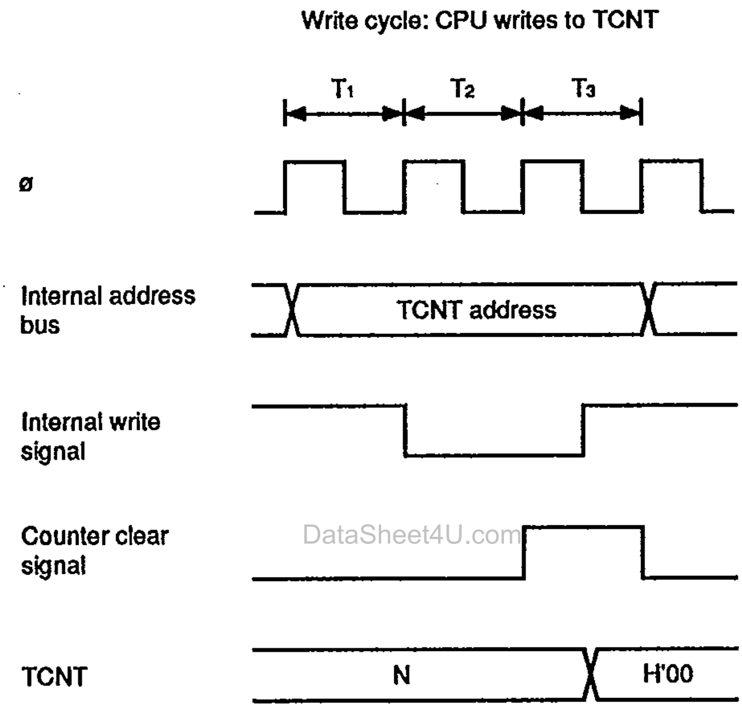


Figure 11-9 TCNT Write-Clear Contention

**Contention between TCNT Write and Increment:** If a timer counter increment pulse is generated during the T3 state of a write cycle to the timer counter, the write takes priority and the timer counter is not incremented.

Figure 11-10 shows this type of contention.

T-49-19-16

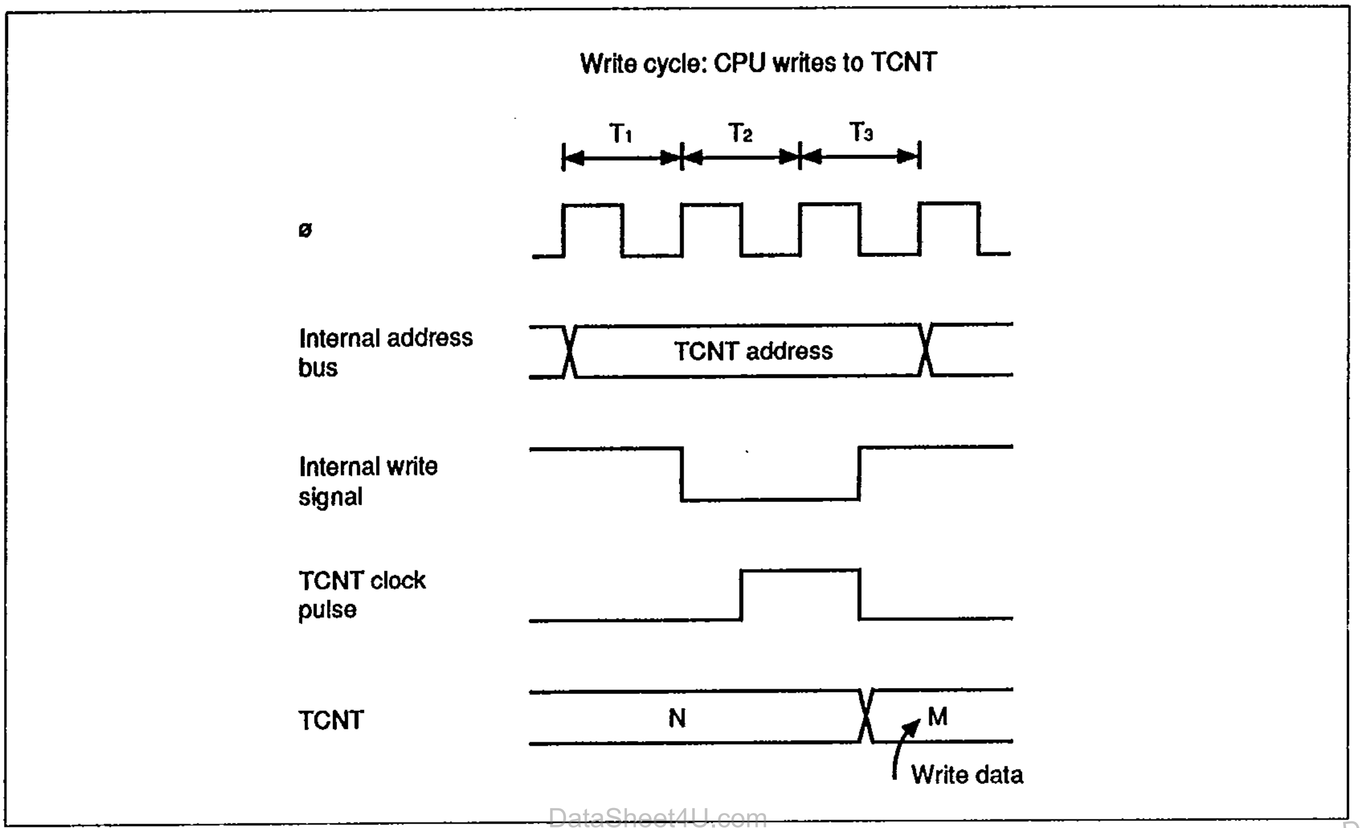


Figure 11-10 TCNT Write-Increment Contention

**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.

T-49-19-16

Figure 11-11 shows this type of contention.

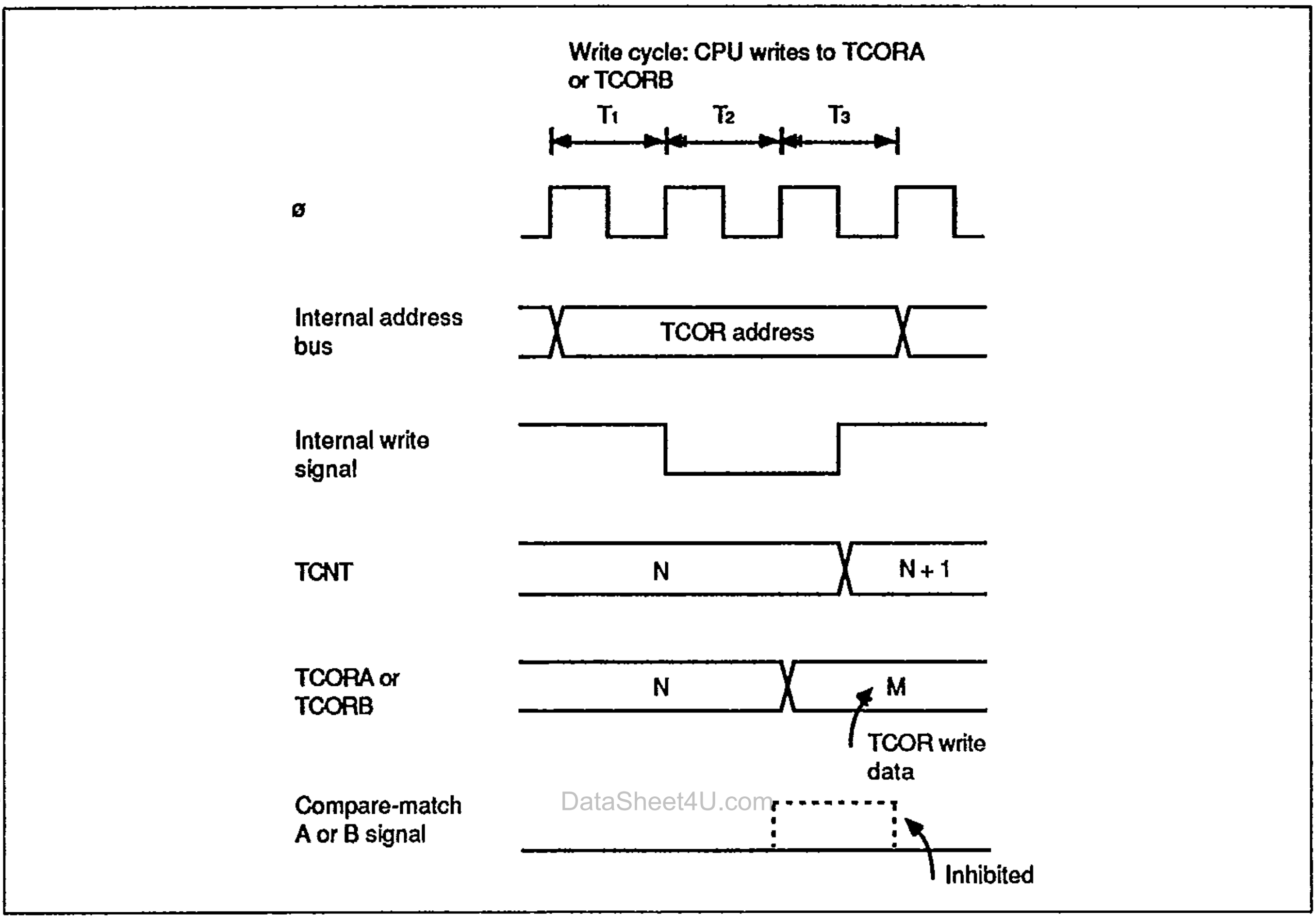


Figure 11-11 Contention between TCOR Write and Compare-Match

**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 11-4.

### Table 11-4 Priority Order of Timer Output

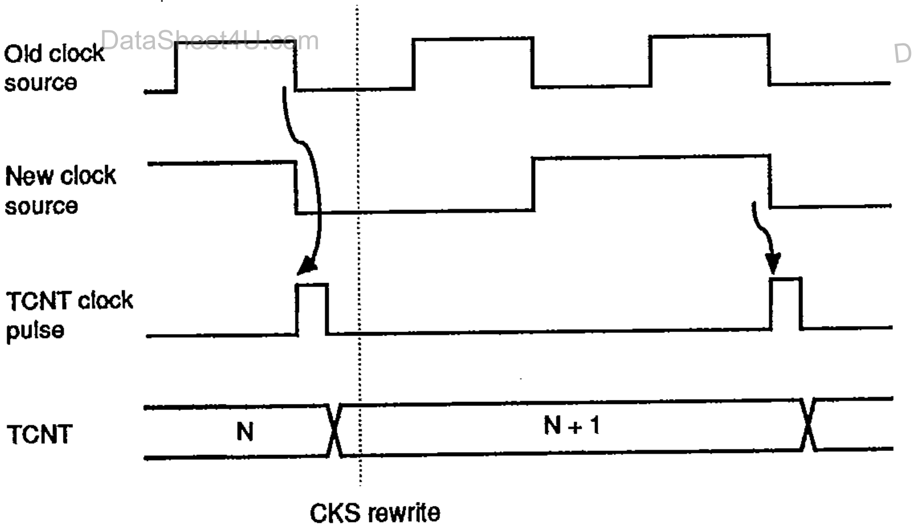
| Output Selection | Priority |
|---|---|
| Toggle | High |
| 1 Output | ↑ |
| 0 Output | |
| No change | Low |

**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 11-5.

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 11-5, 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.

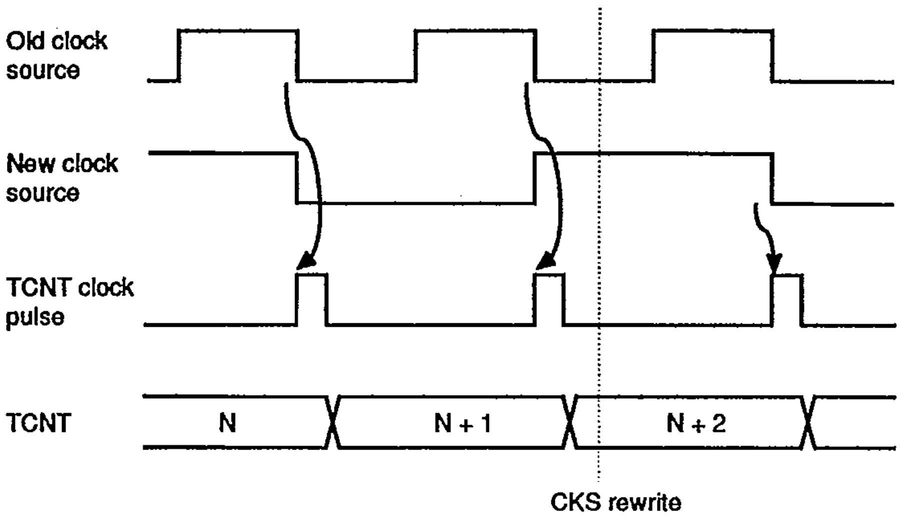
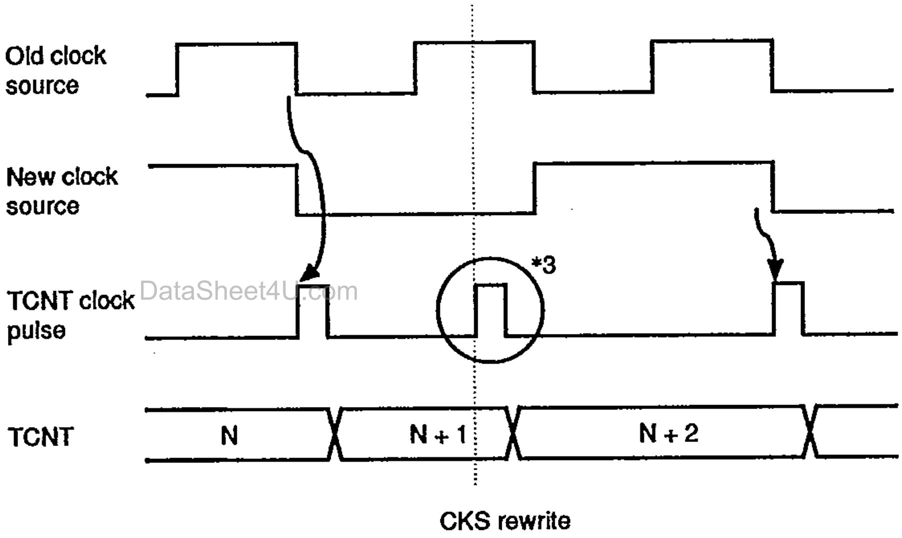
### Table 11-5 Effect of Changing Internal Clock Sources

| No. | Description | Timing Chart |
|---|---|---|
| 1 | Low → Low*:<br>CKS1 and CKS0 are rewritten while both clock sources are low. |  |

Note: * Including a transition from low to the stopped state (CKS1 = 0, CKS0 = 0), or a transition from the stopped state to low.

## Table 11-5 Effect of Changing Internal Clock Sources (cont)

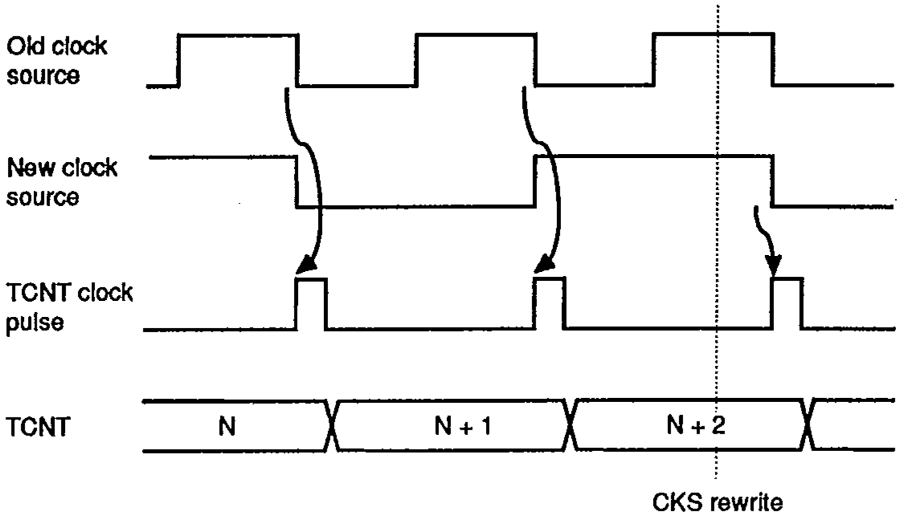
T-49-19-16

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

Notes:
1. Including a transition from the stopped state to high.
2. Including a transition from high to the stopped state.
3. The switching of clock sources is regarded as a falling edge that increments the TCNT.

## Table 11-5 Effect of Changing Internal Clock Sources (cont)

T-49-19-16

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

# Section 12 Watchdog Timer

## 12.1 Overview

The H8/520 has an on-chip watchdog timer (WDT) module. This module can monitor system operation by generating a signal that resets the H8/520 chip if a system crash allows the timer count to overflow.

When this watchdog function is not needed, the WDT module can be used as an interval timer. In the interval timer mode, an $\mathrm{IRQ_0}$ interrupt is requested at each counter overflow.

The WDT module is also used in recovering from the software standby mode.

### 12.1.1 Features

The basic features of the watchdog timer module are summarized as follows:

- Selection of eight clock sources
- Selection of two modes: watchdog timer mode and interval timer mode
- Counter overflow generates a reset signal or interrupt request
  Reset signal in the watchdog timer mode; $\mathrm{IRQ_0}$ request in the interval timer mode.
- External output of reset signal
  Depending on a reset output enable bit, the reset signal can be output externally to reset devices controlled by the H8/520, as well as the H8/520 itself.

### 12.1.2 Block Diagram

T-49-19-16

Figure 12-1 is a block diagram of the watchdog timer.

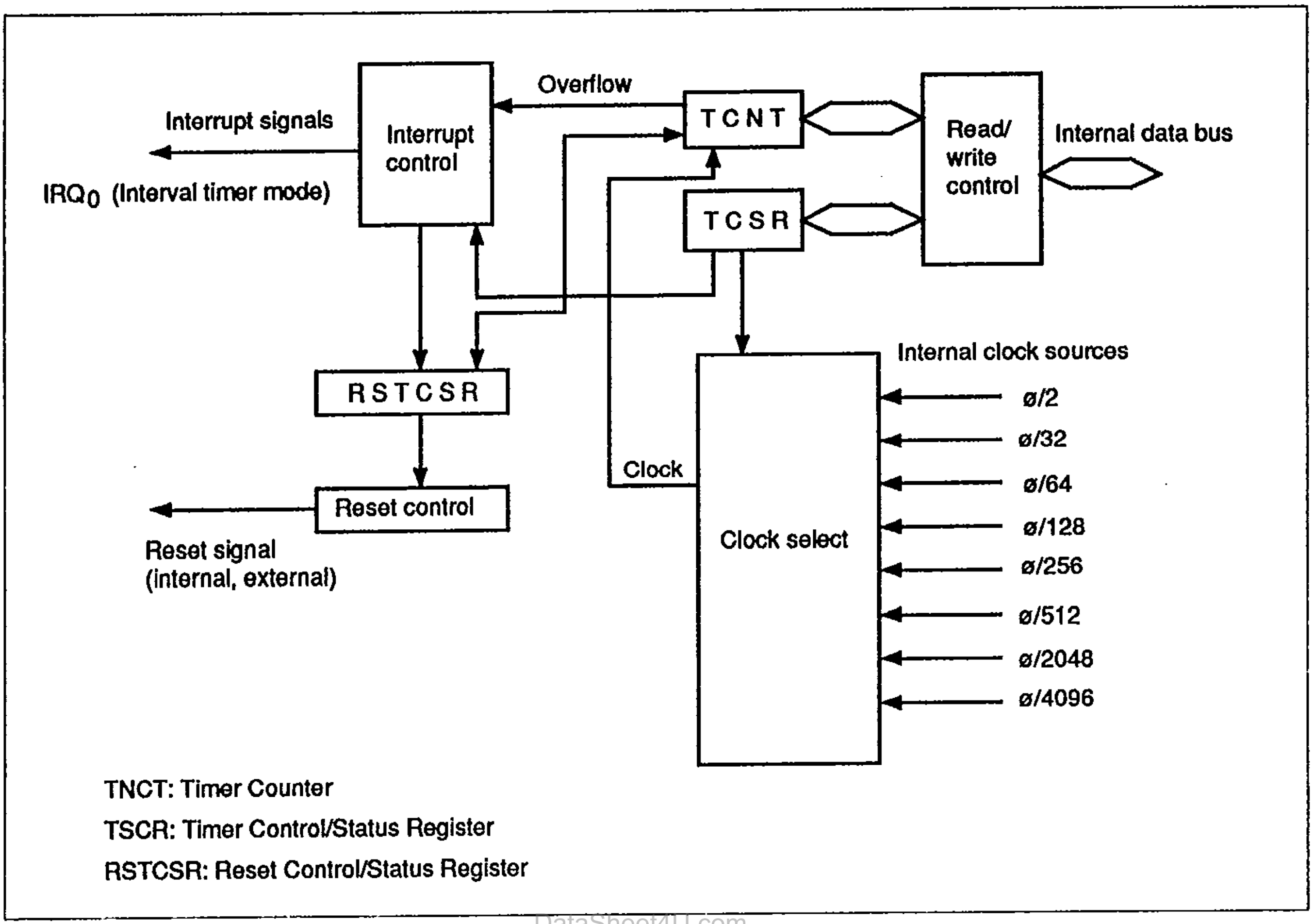


**Figure 12-1 Block Diagram of Timer Counter**

### 12.1.3 Register Configuration

T-49-19-16

Table 12-1 lists information on the watchdog timer registers.

**Table 12-1 Register Configuration**

| Name | Abbreviation | R/W | Initial value | Addresses Write | Addresses Read |
|---|---|---|---|---|---|
| Timer control/status register | TCSR | R/(W)* | H'18 | H'FFEC | H'FFEC |
| Timer counter | TCNT | R/W | H'00 | H'FFEC | H'FFED |
| Reset control/status register | RSTCSR | R/(W)* | H'3F | H'FFFE | H'FFFF |

Note: * Software can write a 0 to clear bit 7, but cannot write a 1.

## 12.2 Register Descriptions

### 12.2.1 Timer Counter (TCNT)—H'FFED

| 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 |

The watchdog timer counter (TCNT) is a readable/writable* 8-bit up-counter. When the timer enable bit (TME) in the timer control/status register (TCSR) is set to 1, the timer counter starts counting pulses of an internal clock source selected by clock select bits 2 to 0 (CKS2 to CKS0) in the TCSR. When the count overflows (changes from H'FF to H'00), a reset or interrupt signal is generated.

The watchdog timer counter is initialized to H'00 at a reset and when the TME bit is cleared to 0.

Note: * TCNT is write-protected by a password. See section 12.2.4, "Notes on Register Access", for details.

### 12.2.2 Timer Control/Status Register (TCSR)—H'FFEC (Read), H'FFED (Write)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | OVF | WT/$\overline{\text{IT}}$ | TME | — | — | CKS2 | CKS1 | CKS0 |
| Initial value | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| Read/Write | R/W*1 | R/W | R/W | — | — | R/W | R/W | R/W |

T-49-19-16

The watchdog timer control/status register (TCSR) is an 8-bit readable/writable*2 register that selects the timer mode and clock source and performs other functions.

Bits 7 to 5 are initialized to 0 at a reset and in the standby modes. Bits 2 to 0 are initialized to 0 at a reset, but retain their values in the software standby mode.

Notes: 1. Software can write a 0 in bit 7 to clear the flag, but cannot set this bit to 1.
2. The TCSR is write-protected by a password. See section 12.2.4, "Notes on Register Access", for details.

**Bit 7—Overflow Flag (OVF):** This bit indicates that the watchdog timer count has overflowed.

| Bit 7<br>OVF | Description | |
|---|---|---|
| 0 | To clear this bit, the CPU must read this bit 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.* | |

Note: * The OVF bit is not set in the watchdog timer mode.

**Bit 6—Timer Mode Select (WT/$\overline{\text{IT}}$):** This bit selects whether to operate in the watchdog timer mode or interval timer mode. If the watchdog timer mode is selected, a watchdog timer overflow resets the chip. If the interval timer mode is selected, a watchdog timer overflow generates an $IRQ_0$ interrupt request.

| Bit 6<br>WT/$\overline{\text{IT}}$ | Description | |
|---|---|---|
| 0 | Interval timer mode ($IRQ_0$ request) | (Initial value) |
| 1 | Watchdog timer mode (Reset) | |

**Bit 5—Timer Enable (TME):** This bit enables or disables the timer.

T-49-19-16

| Bit 5<br>TME | Description | |
|---|---|---|
| 0 | TCNT is initialized to H'00 and stopped. | (Initial value) |
| 1 | TCNT runs. A reset or interrupt is requested when the count overflows. | |

**Bits 4 and 3—Reserved:** These bits cannot be modified and are always read as 1.

**Bits 2, 1, and 0—Clock Select (CKS2, CKS1, and CKS0):** These bits select one of eight clock sources obtained by dividing the system clock (ø).

The overflow interval listed in the table below is the time from when the watchdog timer counter begins counting from H'00 until an overflow occurs.

In the interval timer mode, IRQ0 interrupts are requested at this interval.

| Bit 2 | Bit 1 | Bit 0 | Description | | |
|---|---|---|---|---|---|
| CKS2 | CKS1 | CKS0 | Clock Source | Overflow Interval (ø = 10 MHz) | |
| 0 | 0 | 0 | ø/2 | 51.2 µs | (Initial value) |
| 0 | 0 | 1 | ø/32 | 819.2 µs | |
| 0 | 1 | 0 | ø/64 | 1.6 ms | |
| 0 | 1 | 1 | ø/128 | 3.3 ms | |
| 1 | 0 | 0 | ø/256 | 6.6 ms | |
| 1 | 0 | 1 | ø/512 | 13.1 ms | |
| 1 | 1 | 0 | ø/2048 | 52.4 ms | |
| 1 | 1 | 1 | ø/4096 | 104.9 ms | |

### 12.2.3 Reset Control/Status Register (RSTCSR)—H'FFFF (Read), H'FFFE (Write)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WRST | RSTOE | — | — | — | — | — | — |
| Initial value | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| Read/Write | R/(W)*1 | R/W | — | — | — | — | — | — |

The reset control/status register (RSTCSR) is an 8-bit readable/writable*2 register that indicates when a reset has been caused by a watchdog timer overflow, and controls external output of the reset signal.

Bit 6 is not initialized by the reset caused by the watchdog timer overflow. It is initialized, however, by a reset caused by input at the $\overline{\text{RES}}$ pin.

Notes: 1. Software can write a 0 in bit 7 to clear the flag, but cannot set this bit to 1.
2. The RSTCSR is write-protected by a password. See section 12.2.4, "Notes on Register Access", for details.

**Bit 7—Watchdog Timer Reset (WRST):** This bit indicates that a reset signal has been generated by a watchdog timer overflow in the watchdog timer mode.

The reset signal generated by the overflow resets the entire H8/520 chip. In addition, if the reset output enable (RSTOE) bit is set to 1, a reset signal (low) is ouput at the $\overline{\text{RES}}$ pin to reset devices connected to the H8/520.

The WRST bit can be cleared by software by writing a 0. It is also cleared when a reset signal from an external device is received at the $\overline{\text{RES}}$ pin.

| Bit 7<br>WRST | Description | |
|---|---|---|
| 0 | This bit is cleared to 0 by a reset signal input from the $\overline{\text{RES}}$ pin, or when software writes a 0. | (Initial state) |
| 1 | This bit is set to 1 when the watchdog timer overflows in the watchdog timer mode and an internal reset signal is generated. | |

**Bit 6—Reset Output Enable (RSTOE):** This bit selects whether to output a reset signal from the $\overline{\text{RST}}$ pin when the timer counter overflows in the watchdog timer mode.

| Bit 6<br>RSTOE | Description | |
|---|---|---|
| 0 | The reset signal generated by a watchdog timer overflow is not output to external devices. | (Initial state) |
| 1 | The reset signal generated by a watchdog timer overflow is output to external devices. | |

**Bits 5 to 0—Reserved:** These bits cannot be modified and are always read as 1.

### 12.2.4 Notes on Register Access

The watchdog timer's TCNT, TCSR, and RSTCSR registers differ from other registers in being more difficult to write. The procedures for writing and reading these registers are given below.

**Writing to TCNT and TCSR:** These registers must be written by word access. Programs cannot write to them by byte access. The word must contain the write data and a password.

T-49-19-16

The watchdog timer's TCNT and TCSR registers both have the same write address. The write data must be contained in the lower byte of the word written at this address. The upper byte must contain H'5A (password for TCNT) or H'A5 (password for TCSR). See figure 12-2.

The result of the access depicted in figure 12-2 is to transfer the write data from the lower byte to the TCNT or TCSR.

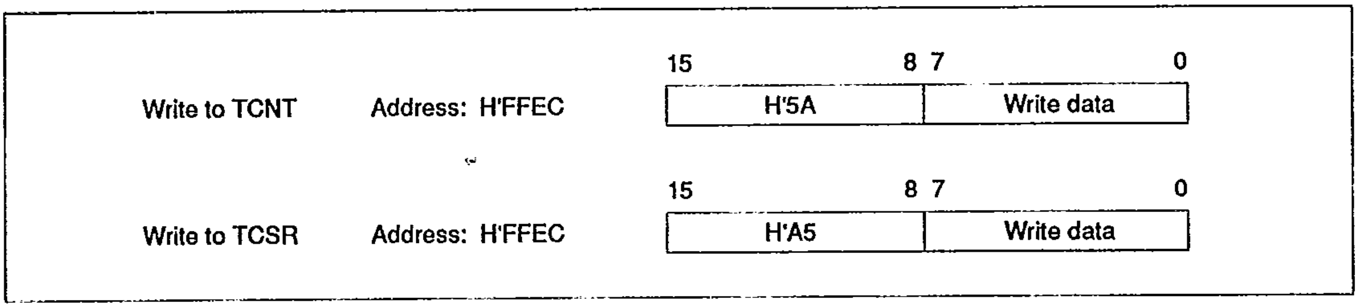


**Figure 12-2 Writing to TCNT and TCSR**

**Coding Examples:**

To clear TCNT to 00: `MOV.W #H'5A00, @H'FFEC`

To write H'4F in TCSR: `MOV.W #H'A54F, @H'FFEC`

**Writing to RSTCSR:** The RSTCSR must be written by moving word data to address H'FFFE. It cannot be written by byte access.

The upper byte of the word must contain a password. Separate passwords are used for clearing the WRST bit and for writing a 1 or 0 to the RSTOE bit.

To clear the WRST bit, the word written at address H'FFFE must contain the password H'A5 in the upper byte and the data H'00 in the lower byte. This clears the WRST bit to 0 without affecting other bits.

To set or clear the RSTOE bit, the word written at address H'FFFE must contain the password H'5A in the upper byte and the write data in the lower byte. This writes the desired data in the RSTOE bit without affecting other bits.

These write operations are illustrated in figure 12-3.

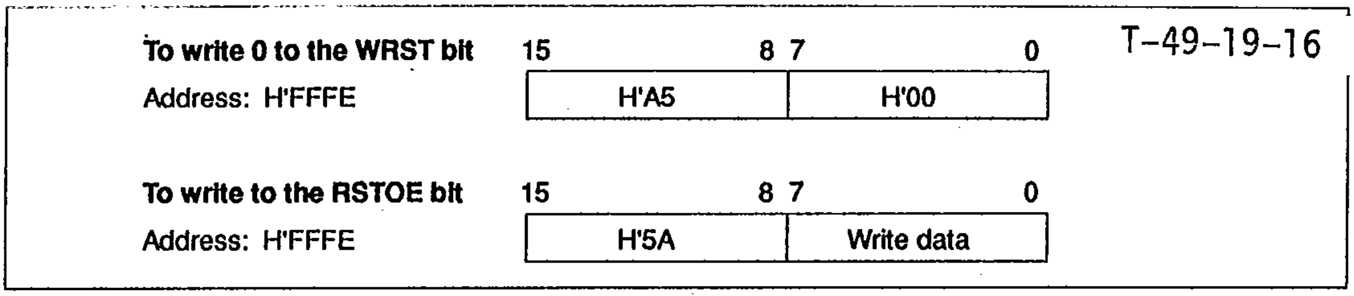


Figure 12-3 Writing to RSTCSR

**Coding Examples:**

To clear WRST to 0: `MOV.W #H'A500, @H'FFFE`

To set RSTOE to 1: `MOV.W #H'5AFF, @H'FFFE`

**Reading TCNT, TCSR, and RSTCSR:** The read addresses are H'FFEC for TCSR, H'FFED for TCNT, and H'FFFF for RSTCSR as indicated in table 12-2.

These three registers are read like other registers. Byte access instructions can be used.

**Table 12-2 Read Addresses of TCNT and TCSR**

| Read Address | Register |
|---|---|
| H'FFEC | TCSR |
| H'FFED | TCNT |
| H'FFFF | RSTCSR |

## 12.3 Operation

### 12.3.1 Watchdog Timer Mode

The watchdog timer function begins operating when software sets the WT/$\overline{\text{IT}}$ and TME bits to 1 in the TCSR. Thereafter, software should periodically rewrite the contents of the timer counter (normally by writing H'00) to prevent the count from overflowing. If a program crash allows the timer count to overflow, the watchdog timer generates a reset as shown in figure 12-4.

The reset signal from the watchdog timer can also be output from the $\overline{\text{RES}}$ pin to reset external devices. This reset output signal is a low pulse with a duration of 132 ø clock periods. The reset signal is output only if the RSTOE bit in the TCSR is set to 1.

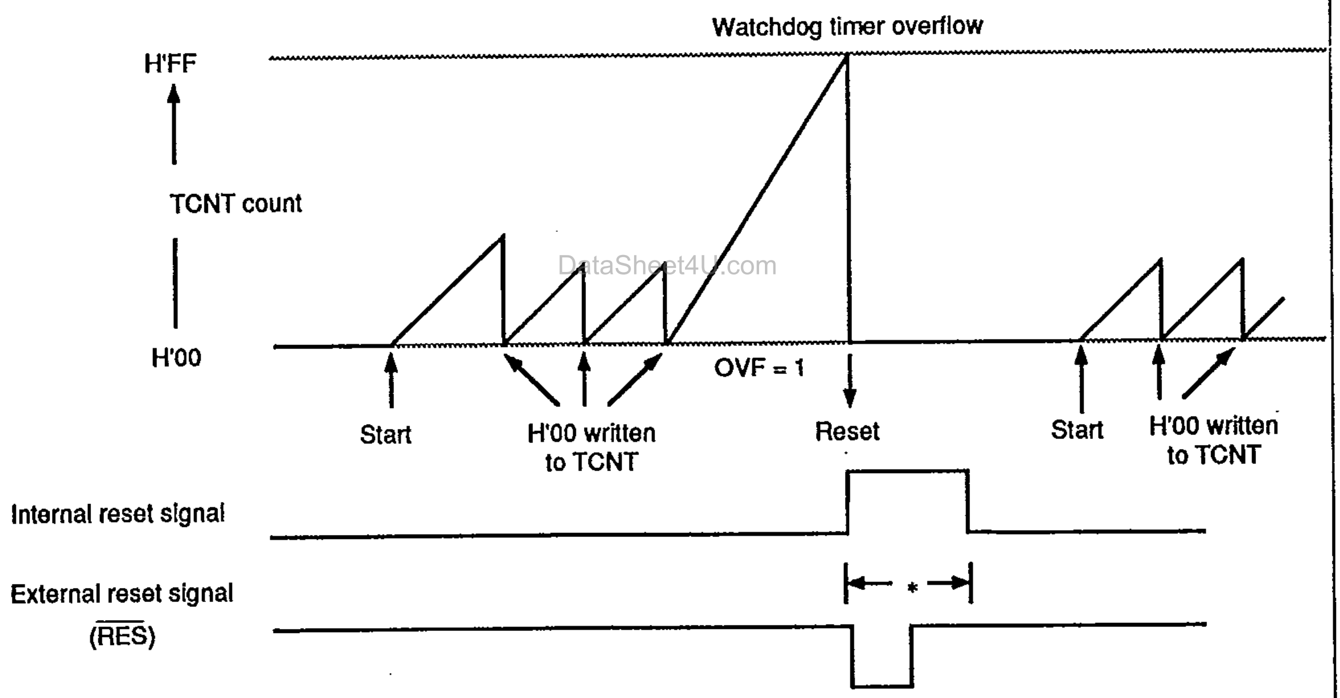
The reset signal from the watchdog timer has the same vector as a reset generated by low input at the $\overline{\text{RES}}$ pin. Software should check the WRST bit in the RSTCSR to determine the source of the reset.

If a watchdog timer overflow occurs at the same time as a low input at the $\overline{\text{RES}}$ pin, priority is given to one type of reset or the other depending on the value of the RSTOE bit in the RSTCSR.

T-49-19-16

If the RSTOE bit is set to 1 when both types of reset occur simultaneously, the watchdog timer's reset signal takes precedence. The internal state of the H8/520 chip is reset, the RSTOE bit remains set to 1, the WRST bit is also set to 1, and the $\overline{\text{RES}}$ pin is held low for 132 ø clock periods. If at the end of 520 ø clock periods there is still an external low input to the $\overline{\text{RES}}$ pin, the external reset takes effect, clearing the WRST and RSTOE bits to 0. Note that if the external reset occurs before the watchdog timer overflows, it takes effect immediately and clears the RSTOE bit.

If the RSTOE bit is cleared to 0 when both types of reset occur simultaneoualy, the reset signal input from the $\overline{\text{RES}}$ pin takes precedence and the WRST bit is cleared to 0.



Note: The external reset signal is output for 132 system clock (ø) cycles.
The internal reset signal lasts for 520 system clock (ø) cycles.

Figure 12-4 Operation in Watchdog Timer Mode

### 12.3.2 Interval Timer Mode

Interval timer operation begins when the WT/$\overline{\text{IT}}$ bit is cleared to 0 and the TME bit is set to 1.

In the interval timer mode, an IRQ$_0$ request is generated each time the timer count overflows. This function can be used to generate IRQ$_0$ requests at regular intervals. See figure 12-5.

IRQ$_0$ requests from the watchdog timer module have the same vector as IRQ$_0$ requests from the $\overline{\text{IRQ}_0}$ pin, so the IRQ$_0$ interrupt-handling routine must check the OVF bit in the TCSR to determine the source of the interrupt.

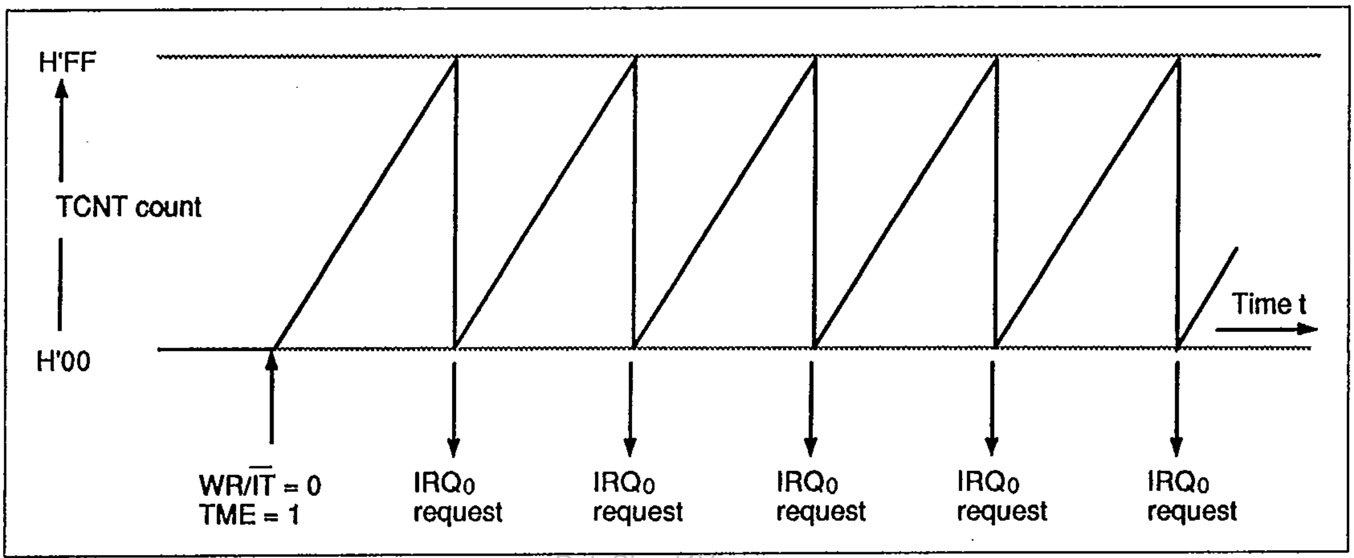


**Figure 12-5 Operation in Interval Timer Mode**

### 12.3.3 Operation in Software Standby Mode

The watchdog timer has a special function in the software standby mode. Specific watchdog timer settings are required when the software standby mode is used.

**Before Transition to the Software Standby Mode:** The TME bit must be cleared to 0 to stop the watchdog timer counter before a transition to the software standby mode. The chip cannot enter the software standby mode while the TME bit is set to 1. Before entering the software standby mode, software should also set the clock select bits (CKS2 to CKS0) to a value that makes the timer overflow interval equal to or greater than the settling time of the clock oscillator.

**Recovery from the Software Standby Mode:** Recovery from the software standby mode can be triggered by an NMI request. In this case the recovery proceeds as follows:

When an NMI request signal is received, the clock oscillator starts running and the watchdog timer starts counting at the rate selected by the clock select bits before the software standby mode was entered. When the count overflows from H'FF to H'00, the ø clock is presumed to be stable and usable, clock signals are supplied to all modules on the chip, and the NMI interrupt-handling routine starts executing.

T-49-19-16

### 12.3.4 Setting of Overflow Flag

The OVF bit is set to 1 when the timer count overflows in the interval timer mode. Simultaneously, the WDT module requests an IRQ0 interrupt. The timing is shown in figure 12-6.

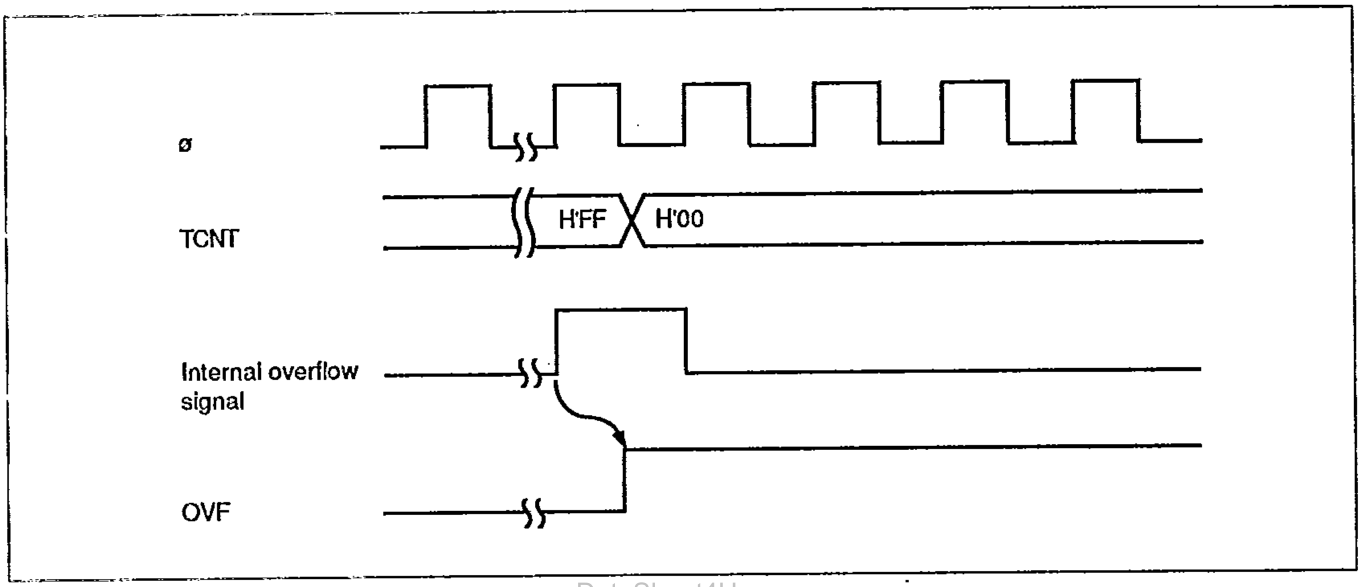


Figure 12-6 Setting of OVF Bit

### 12.3.5 Setting of Watchdog Timer Reset (WRST) Bit

The WRST bit is valid when WT/$\overline{\text{IT}}$ = 1 and TME = 1.

The WRST bit is set to 1 when the timer count overflows. An internal reset signal is simultaneously generated for the entire H8/520 chip. The timing is shown in figure 12-7.

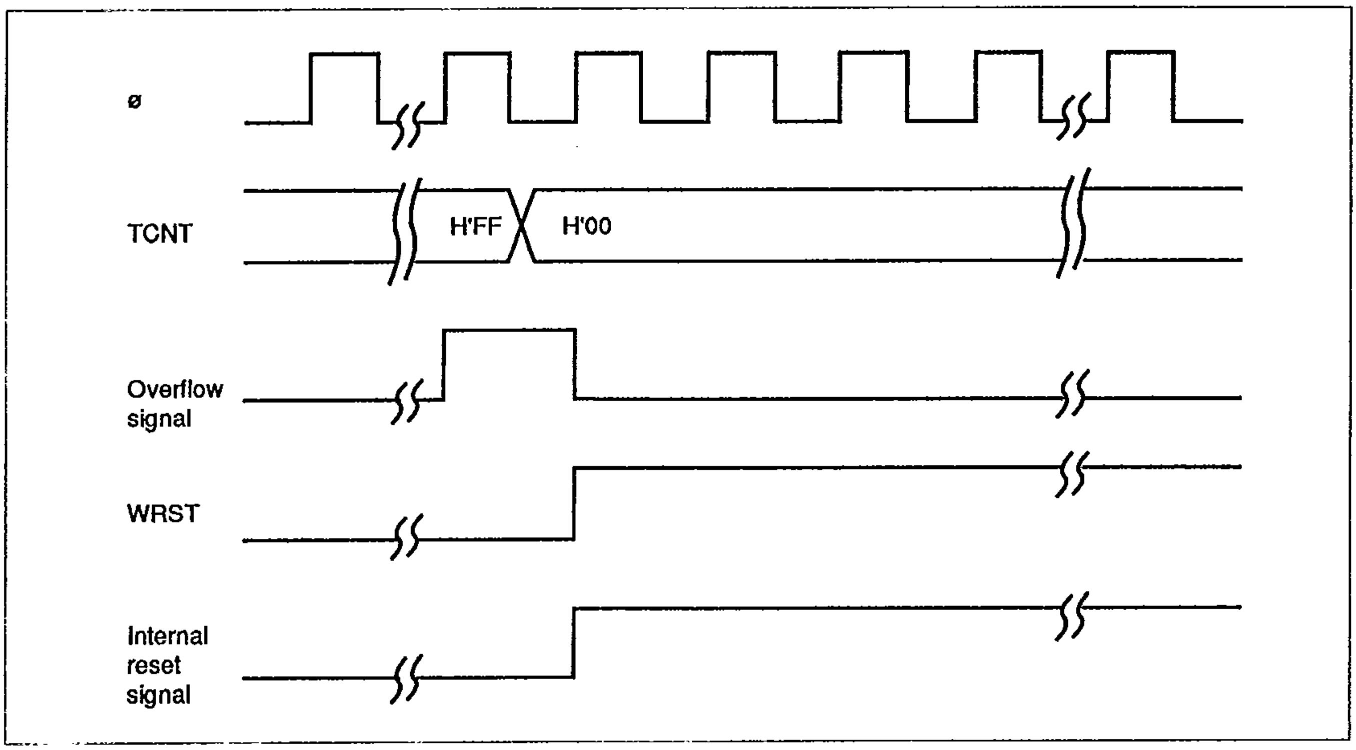


Figure 12-7 Setting of WRST Bit and Internal Reset Signal

## 12.4 Application Notes

**Contention between TCNT Write and Increment:** If a timer counter clock pulse is generated during the $T_3$ state of a write cycle to the timer counter, the write takes priority and the timer counter is not incremented. See figure 12-8.

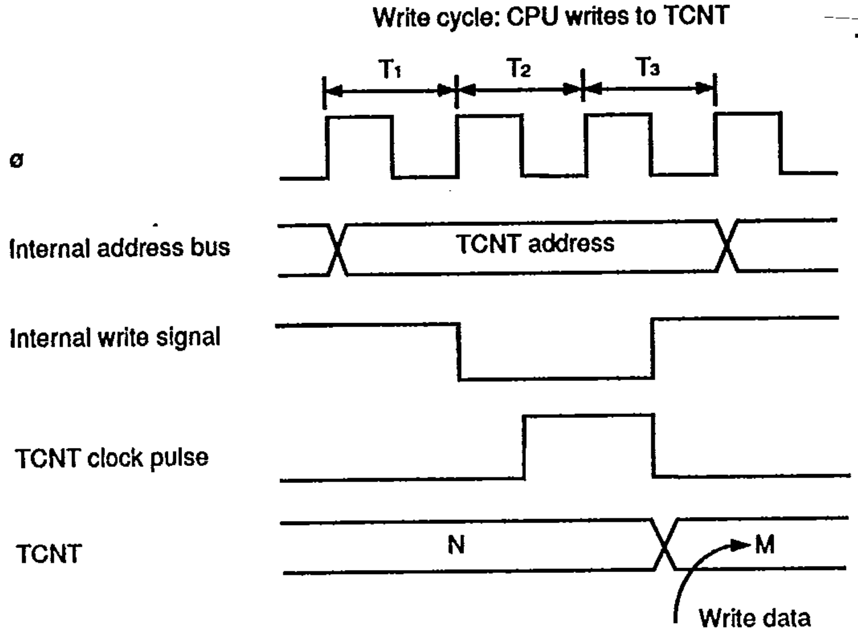


Figure 12-8 TCNT Write-Increment Contention

**Changing the Clock Select Bits (CKS2 to CKS0):** Software should stop the watchdog timer (by clearing the TME bit to 0) before changing the value of the clock select bits. If the clock select bits are modified while the watchdog timer is running, the timer count may be incremented incorrectly.

# Section 13 Serial Communication Interface

T-49-19-16

## 13.1 Overview

The H8/520 chip includes a two-channel serial communication interface (SCI) for transferring serial data to and from other chips. The two channels are independent but are functionally identical. Synchronous and asynchronous data transfer are supported on both channels.

### 13.1.1 Features

The features of the on-chip serial communication interface are as follows:

- Selection of asynchronous or synchronous mode
  - 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 synchronize data transfers with clock pulses.
    - 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 input at the SCK pin.
- Three interrupts
  Transmit-end, receive-end, and receive-error interrupts are requested independently. The transmit-end and receive-end interrupts can be served by the on-chip data transfer controller (DTC), providing a convenient way to transfer data with minimal CPU programming.

### 13.1.2 Block Diagram

Figure 13-1 shows a block diagram of one serial communication interface channel.

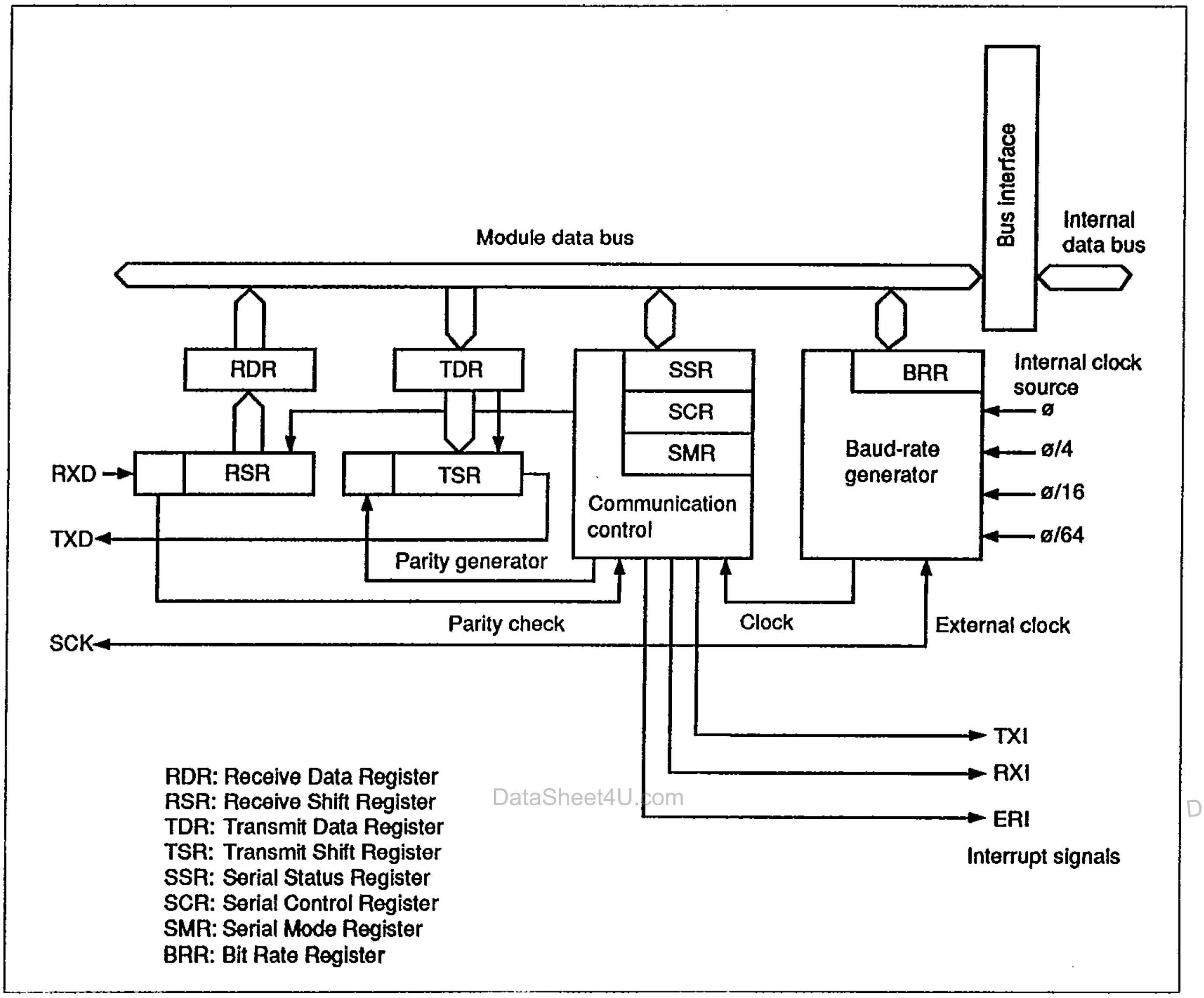


Figure 13-1 Block Diagram of Serial Communication Interface

### 13.1.3 Input and Output Pins

Table 13-1 lists the input and output pins used by the SCI module.

**Table 13-1 SCI Input/Output Pins**

| Channel | Name | Abbreviation | I/O | Function |
|---|---|---|---|---|
| 1 | Serial clock | $SCK_1$ | Input/output | Serial clock input and output for channel 1 |
| | Receive data | $RXD_1$ | Input | Receive data input for channel 1 |
| | Transmit data | $TXD_1$ | Output | Transmit data output for channel 1 |
| 2 | Serial clock | $SCK_2$ | Input/output | Serial clock input and output for channel 2 |
| | Receive data | $RXD_2$ | Input | Receive data input for channel 2 |
| | Transmit data | $TXD_2$ | Output | Transmit data output for channel 2 |

### 13.1.4 Register Configuration

Table 13-2 lists the SCI registers. These registers specify the communication mode (synchronous or asynchronous), data format, and bit rate, and control the transmit and receive sections.

**Table 13-2 SCI Registers**

| Channel | Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|---|
| 1 | 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 |
| 2 | Receive shift register | RSR | — | — | — |
| | Receive data register | RDR | R | H'00 | H'FFC5 |
| | Transmit shift register | TSR | — | — | — |
| | Transmit data register | TDR | R/W | H'FF | H'FFC3 |
| | Serial mode register | SMR | R/W | H'04 | H'FFC0 |
| | Serial control register | SCR | R/W | H'0C | H'FFC2 |
| | Serial status register | SSR | R/(W)* | H'87 | H'FFC4 |
| | Bit rate register | BRR | R/W | H'FF | H'FFC1 |

Note: * Software can write a 0 to clear the status flag bits, but cannot write a 1.

## 13.2 Register Descriptions

T-49-19-16

### 13.2.1 Receive Shift Register (RSR)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |
| Read/Write | — | — | — | — | — | — | — | — |

The RSR receives incoming data bits. When one character (one byte) has been received, it is transferred to the receive data register (RDR).

The CPU cannot read or write the RSR directly.

### 13.2.2 Receive Data Register (RDR)—H'FFDD (Channel 1), H'FFC5 (Channel 2)

| 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.

### 13.2.3 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 TDR does not contain valid data, the SCI stops transmitting.

The CPU cannot read or write the TSR directly.

### 13.2.4 Transmit Data Register (TDR)—H'FFDB (Channel 1), H'FFC3 (Channel 2)

| 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 |

T-49-19-16

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.

### 13.2.5 Serial Mode Register (SMR)—H'FFD8 (Channel 1), H'FFC0 (Channel 2)

| 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.

**Bit 7—Communication Mode (C/$\overline{A}$):** This bit selects the asynchronous or synchronous communication mode.

| Bit 7<br>C/$\overline{A}$ | Description | |
|---|---|---|
| 0 | Asynchronous communication. | (Initial value) |
| 1 | Communication is synchronized with the serial clock. | |

**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. | |

T-49-19-16

**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/$\overline{\text{E}}$):** 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 1s even. Odd parity means that the total number of 1s is made odd.

This bit is ignored when PE = 0 and in the synchronous mode.

**Bit 4**

| O/$\overline{\text{E}}$ | 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 from within the H8/520 chip.

T-49-19-16

| Bit 1<br>CKS1 | Bit 0<br>CKS0 | Description | |
|---|---|---|---|
| 0 | 0 | ø clock | (Initial value) |
| 0 | 1 | ø/4 clock | |
| 1 | 0 | ø/16 clock | |
| 1 | 1 | ø/64 clock | |

### 13.2.6 Serial Control Register (SCR)—H'FFDA (Channel 1), H'FFC2 (Channel 2)

| 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 (RIE):** This bit enables or disables the transmit-end interrupt (TXI) request when the transmit data register empty (TDRE) bit in the serial status register (SSR) is set to 1.

| Bit 7<br>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) request when the receive data register full (RDRF) bit in the serial status register (SSR) is set to 1. It also enables and disables the receive-error interrupt (ERI) request.

| Bit 6<br>RIE | Description | |
|---|---|---|
| 0 | The receive-end interrupt (RXI) and receive-error interrupt (ERI) requests are disabled. | (Initial value) |
| 1 | The receive-end interrupt (RXI) and receive-error interrupt (ERI) requests are 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<br>TE | Description | |
|---|---|---|
| 0 | The transmit function is disabled. The TXD pin can be used as a general-purpose I/O port. | (Initial value) |
| 1 | The transmit function is enabled. 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<br>RE | Description | |
|---|---|---|
| 0 | The receive function is disabled. The RXD pin can be used as a general-purpose I/O port. | (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 SCI clock source: either the internal baud rate generator or an external clock signal input at the SCK pin. When the external clock source is selected, the SCK pin is automatically used for input of the external clock signal.

| Bit 1<br>CKE1 | Description | |
|---|---|---|
| 0 | Internal clock source. | (Initial value) |
| 1 | External clock source. (The SCK pin is used for input.) | |

**Bit 0—Clock Enable 0 (CKE0):** When an internal clock source is used in synchronous 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 asynchronous mode is selected.

For further information on the communication format and clock source selection, see tables 13-5 and 13-6 in section 13.3, "Operation".

**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. | |

T-49-19-16

### 13.2.7 Serial Status Register (SSR)—H'FFDC (Channel 1), H'FFC4 (Channel 2)

| 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)* | R | R | R |

Note: * 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 | This bit is cleared from 1 to 0 when:<br>1. The CPU reads the TDRE bit after the TDRE bit has been set to 1, then writes a 0 in this bit.<br>2. The data transfer controller (DTC) writes data in the TDR. | |
| 1 | This bit is set to 1 at the following times:<br>1. The chip is reset or enters a standby mode.<br>2. When TDR contents are transferred to the TSR.<br>3. When TDRE = 0 and the TE bit is cleared to 0. | (Initial value) |

**Bit 6—Receive Data Register Full (RDRF):** This bit indicates when one character has been received and transferred to the RDR.

T-49-19-16

| Bit 6<br>RDRF | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when: | (Initial value) |
| | 1. The CPU reads the RDRF bit after the RDRF bit has been set to 1, then writes a 0 in this bit. | |
| | 2. The data transfer controller (DTC) reads the RDR. | |
| | 3. The chip is reset or enters a standby mode. | |
| 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 | This bit is cleared from 1 to 0 when: | (Initial value) |
| | 1. The CPU reads the ORER bit after the ORER bit has been set to 1, then writes a 0 in this bit. | |
| | 2. The chip is reset or enters a standby mode. | |
| 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 the asynchronous mode. It has no meaning in the synchronous mode.

| Bit 4<br>FER | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when: | (Initial value) |
| | 1. The CPU reads the FER bit after the FER bit has been set to 1, then writes a 0 in this bit. | |
| | 2. The chip is reset or enters a standby mode. | |
| 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 the asynchronous mode, when a communication format with parity bits is used.

This bit has no meaning in the synchronous mode, or when a communication format without parity bits is used.

| Bit 3<br>PER | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when:<br>1. The CPU reads the PER bit after the PER bit has been set to 1, then writes a 0 in this bit.<br>2. The chip is reset or enters a standby mode. | (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). | |

T-49-19-16

**Bits 2 to 0—Reserved:** These bits cannot be modified and are always read as 1.

### 13.2.8 Bit Rate Register (BRR)—H'FFD9 (Channel 1), H'FFC1 (Channel 2)

| 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 13-3 and 13-5 show examples of BRR (N) and CKS (n) settings for commonly used bit rates. Different values can be set for each SCI channel. Table 13-4 indicates the maximum bit rates for various crystal oscillator frequencies in the asynchronous mode.

## Table 13-3 Examples of BRR Settings in Asynchronous Mode (1)

T-49-19-16

| | 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 13-3 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 13-3 Examples of BRR Settings in Asynchronous Mode (3)

T-49-19-16

| | 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 | +0.88 |
| 150 | 1 | 255 | 0 | 2 | 64 | +0.16 | 2 | 77 | 0 | 2 | 79 | 0 |
| 300 | 1 | 127 | 0 | 1 | 129 | +0.16 | 1 | 155 | 0 | 1 | 159 | 0 |
| 600 | 0 | 255 | 0 | 1 | 64 | +0.16 | 1 | 77 | 0 | 1 | 79 | 0 |
| 1200 | 0 | 127 | 0 | 0 | 129 | +0.16 | 0 | 155 | +0.16 | 0 | 159 | 0 |
| 2400 | 0 | 63 | 0 | 0 | 64 | +0.16 | 0 | 77 | +0.16 | 0 | 79 | 0 |
| 4800 | 0 | 31 | 0 | 0 | 32 | −1.36 | 0 | 38 | +0.16 | 0 | 39 | 0 |
| 9600 | 0 | 15 | 0 | 0 | 15 | +1.73 | 0 | 19 | −2.34 | 0 | 19 | 0 |
| 19200 | 0 | 7 | 0 | 0 | 7 | +1.73 | — | — | — | 0 | 9 | 0 |
| 31250 | 0 | 4 | −1.70 | 0 | 4 | 0 | 0 | 5 | 0 | 0 | 5 | +2.40 |
| 38400 | 0 | 3 | 0 | 0 | 3 | +1.73 | — | — | — | 0 | 4 | 0 |

## Table 13-3 Examples of BRR Settings in Asynchronous Mode (4)

T-49-19-16

| | 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 |
| 307200 | — | — | — | — | — | — | 0 | 0 | 0 | — | — | — |
| 312500 | — | — | — | — | — | — | — | — | — | 0 | 0 | 0 |

Note: If possible, the error should be within 1%.

$$B = OSC \times 10^6 / [64 \times 2^{2n} \times (N + 1)]$$

B: Bit rate (bits/s)
N: BRR value ($0 \le N \le 255$)
OSC: Crystal oscillator frequency in MHz
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 13-4 Maximum Bit Rate for Various Crystal Oscillator Frequencies (In Asynchronous Mode)

T-49-19-16

| | | CKS and BRR | |
|---|---|---|---|
| XTAL (MHz) | Maximum Bit Rate (bits/s) | n | N |
| 2 | 31250 | 0 | 0 |
| 2.4576 | 38400 | 0 | 0 |
| 4 | 62500 | 0 | 0 |
| 4.194304 | 65536 | 0 | 0 |
| 4.9152 | 76800 | 0 | 0 |
| 6 | 93750 | 0 | 0 |
| 7.3728 | 115200 | 0 | 0 |
| 8 | 125000 | 0 | 0 |
| 9.8304 | 153600 | 0 | 0 |
| 10 | 156250 | 0 | 0 |
| 12 | 187500 | 0 | 0 |
| 12.288 | 192000 | 0 | 0 |
| 14.7456 | 230400 | 0 | 0 |
| 16 | 250000 | 0 | 0 |
| 19.6608 | 307200 | 0 | 0 |
| 20 | 312500 | 0 | 0 |

## Table 13-5 Examples of BRR Settings in Synchronous Mode

T-49-19-16

| Bit | XTAL Frequency (MHz) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 | — | — |
| 1 k | 0 | 249 | 1 | 124 | 1 | 249 | — | — | 2 | 124 | — | — |
| 2.5 k | 0 | 99 | 0 | 199 | 1 | 99 | 1 | 124 | 1 | 199 | 1 | 249 |
| 5 k | 0 | 49 | 0 | 99 | 0 | 199 | 0 | 249 | 1 | 99 | 1 | 124 |
| 10 k | 0 | 24 | 0 | 49 | 0 | 99 | 0 | 124 | 0 | 199 | 0 | 249 |
| 25 k | 0 | 9 | 0 | 19 | 0 | 39 | 0 | 49 | 0 | 79 | 0 | 99 |
| 50 k | 0 | 4 | 0 | 9 | 0 | 19 | 0 | 24 | 0 | 39 | 0 | 49 |
| 100 k | — | — | 0 | 4 | 0 | 9 | — | — | 0 | 19 | 0 | 24 |
| 250 k | 0 | 0* | 0 | 1 | 0 | 3 | 0 | 4 | 0 | 7 | 0 | 9 |
| 500 k | | | 0 | 0* | 0 | 1 | — | — | 0 | 3 | 0 | 4 |
| 1 M | | | | | 0 | 0* | — | — | 0 | 1 | — | — |
| 2.5 M | | | | | | | | | | | 0 | 0* |

Notes: Blank: No setting is available.
—: A setting is available, but the bit rate is inaccurate.
*: Continuous transfer is not possible.

$$B = OSC/[8 \times 2^{2n} \times (N + 1)]$$

B: Bit rate (bits/s)
N: BRR value ($0 \leq N \leq 255$)
OSC: Crystal oscillator frequency in MHz
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 |

## 13.3 Operation

T-49-19-16

### 13.3.1 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 13-6. The clock source and usage of the SCK pin depend on settings in the SMR and SCR as indicated in table 13-7.

Table 13-6 Communication Formats Used by SCI

| SMR | | | | | | | Stop Bit |
|---|---|---|---|---|---|---|---|
| **C/A̅** | **CHR** | **PE** | **STOP** | **Mode** | **Format** | **Parity** | **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 13-7 SCI Clock Source Selection

| SMR | SCR | | Clock | |
|---|---|---|---|---|
| **C/A̅** | **CKE1** | **CKE0** | **Source** | **SCK Pin** |
| 0 (Async mode) | 0 | 0 | Internal | I/O port* |
| | | 1 | | Clock output at same frequency as bit rate |
| | 1 | 0 | External | Clock input at 16 times the bit rate frequency |
| | | 1 | | |
| 1 (Sync mode) | 0 | 0 | Internal | Serial clock output |
| | | 1 | | |
| | 1 | 0 | External | Serial clock input |
| | | 1 | | |

Note: * Not used by the SCI.

Transmitting and receiving operations in the two modes are described next.

### 13.3.2 Asynchronous Mode

In asynchronous mode, each character is individually synchronized by framing it with a start bit and stop bit.

T-49-19-16

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 13-2 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).

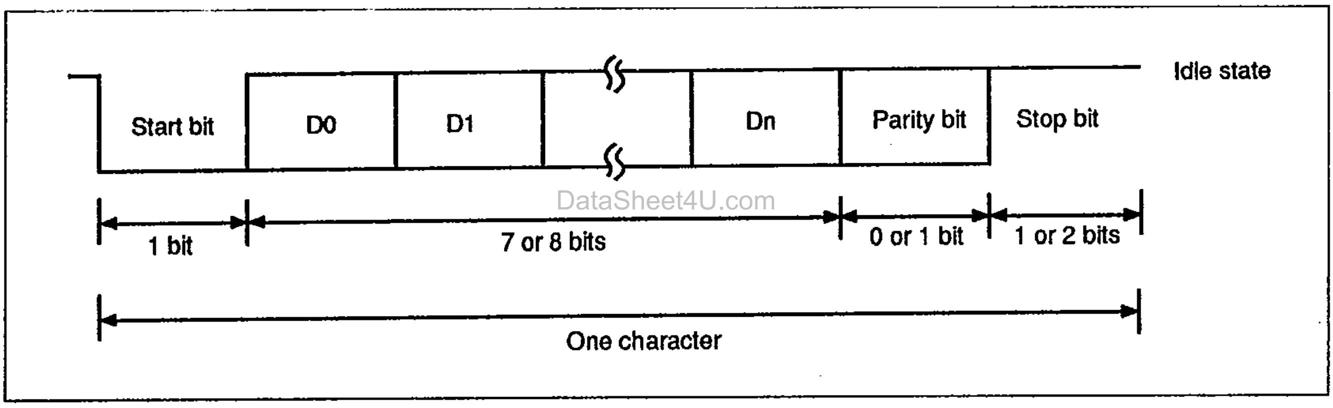


Figure 13-2 Data Format in Asynchronous Mode

**Data Format:** Table 13-8 lists the data formats that can be sent and received in asynchronous mode. Eight formats can be selected by bits in the SMR.

**Table 13-8 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

**Clock:** In the 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 13-7.

If an external clock is input at the SCK pin, its frequency should be 16 times the desired bit 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 bit rate, and the clock pulse rises at the center of the transmit data bits. Figure 13-3 shows the phase relationship between the output clock and transmit data.

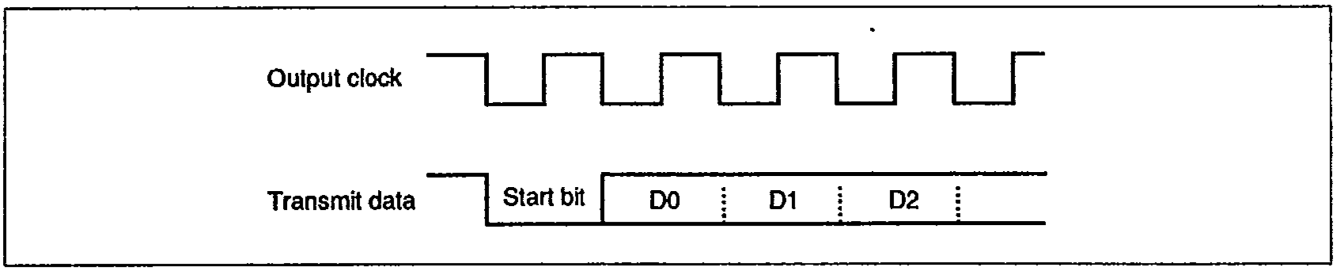


**Figure 13-3 Phase Relationship Between Clock Output and Transmit Data**

**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.

T-49-19-16

1. Set the desired communication format in the SMR.

2. Write the value corresponding to the desired bit 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 1 bit transfer time at the selected communication speed, 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 in the asynchronous mode 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 1s will be transmitted, after which the SCI is ready to transmit data.

Note: * A frame is the data for one character, including the start bit and stop bit(s).

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.
   a. Start bit (one 0 bit).
   b. Transmit data (seven or eight bits, starting from bit 0)

c. Parity bit (odd or even parity bit, or no parity bit)
d. 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.

**Data Reception:** The procedure for receiving data in the asynchronous mode 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 will 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.
   If the stop bit length is 2 bits, the SCI checks that both bits are 1.

4. When a complete frame has been received, the SCI transfers the received data 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 the CPU reads the SSR, then writes a 0 in the RDRF bit, or when the RDR is read by the data transfer controller (DTC). 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 13-9.

If a receive error occurs, the RDRF bit in the SSR is not 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. See section 13.5, "Application Notes".

T-49-19-16

To clear a receive-error flag (ORER, FER, or PER), software must read the SSR, then write a 0 in the flag bit.

Table 13-9 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 bit in the SMR.<br>The RSR contents are transferred to the RDR. |

### 13.3.3 Synchronous Mode

The synchronous mode is suited for high-speed, continuous data transfer. Each bit of data is synchronized with a serial clock pulse.

Continuous data transfer is enabled by the double buffering employed in both the transmit and receive sections of the SCI. Full duplex communication (with the same clock) is possible because the transmit and receive sections are independent.

**Data Format:** Figure 13-4 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.

T-49-19-16

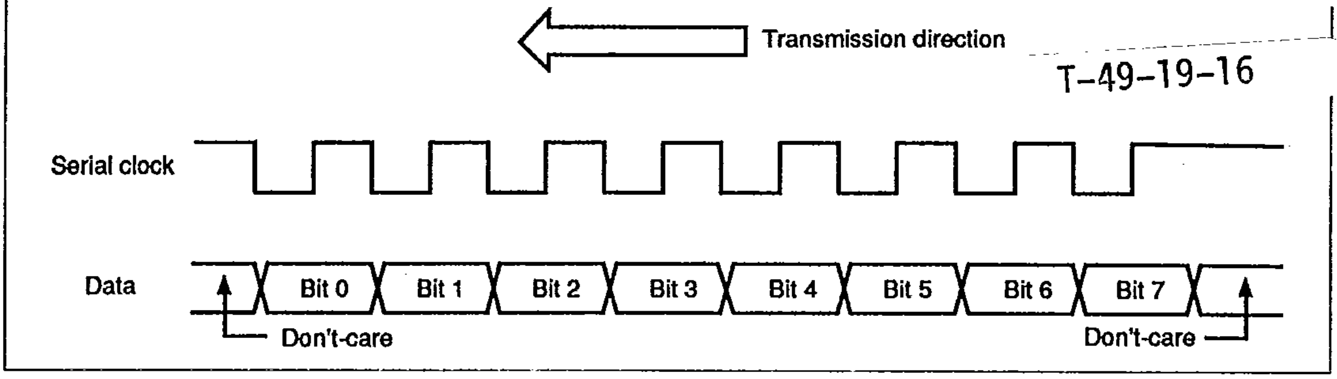


**Figure 13-4 Data Format in Synchronous Mode**

**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 13-7 for details.

**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.

3. Select the synchronous mode in the SMR.

4. Set the TE and/or RE bit in the SCR to 1.

Note: The input/output status of the SCK pin depends on the $C/\overline{A}$ bit in the SMR and the CKE0 and CKE1 bits in the SCR. (See table 13-7.) To prevent incorrect output from the SCK pin, set the SCR before the SMR.

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 in the synchronous mode 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 in the SSR 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 is held at the value of the last bit transmitted.

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 a string of bits equal to the last bit transmitted.

**Data Reception:** The procedure for receiving data in the synchronous mode 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 will automatically be switched to input and the SCI is ready to receive data.

T-49-19-16

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 to the RDR so that it can be read.
   The RDRF bit is cleared when the program reads the RDRF bit in the SSR, then writes a 0 in the RDRF bit, or when the data transfer controller (DTC) reads the RDR.

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 in the synchronous mode 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, the CPU must read the TDRE bit in the SSR, write the next transmit data in the TDR, then clear the TDRE bit to 0. Alternatively, the DTC can write the next transmit data in the TDR, in which case the TDRE bit is cleared automatically.
   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 last bit in the TSR.

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 must read the RDRF bit in the SSR, then write a 0 in the RDRF bit. Alternatively, the DTC can read the RDR, in which case the RDRF bit is cleared automatically.
   For continuous data reception, the RDRF bit must be cleared to 0 before the last bit of the next byte of data is received.

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.

An overrun error does not affect the transmit section of the SCI, which continues to transmit normally.

## 13.4 CPU Interrupts and DTC 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 13-10 lists information about these interrupts.

T-49-19-16

### Table 13-10 SCI Interrupts

| Interrupt | Description | DTC Service Available? | Priority |
|---|---|---|---|
| ERI | Receive-error interrupt, requested when ORER, FER, or PER is set. | No | High ↑ |
| RXI | Receive-end interrupt, requested when RDRF is set. | Yes | |
| TXI | Transmit-end interrupt, requested when TDRE is set. | Yes | Low |

The TXI and RXI interrupts can be served by the data transfer controller (DTC) to have a data transfer performed. When the DTC serves one of these interrupts, it clears the TDRE or RDRF bit to 0 under the following conditions, which differ between the two bits.

When invoked by a TXI request, if the DTC writes to the TDR, it automatically clears the TDRE bit to 0. When invoked by an RXI request, if the DTC reads from the RDR, it automatically clears the RDRF bit to 0.

See section 6, "Data Transfer Controller", for further information on the DTC.

## 13.5 Application Notes

Application programmers should note the following features of the SCI.

**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.

**Multiple Receive Errors:** Table 13-11 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 13-11 SSR Bit States and Data Transfer When Multiple Receive Errors Occur

T-49-19-16

| | SSR Bits | | | | |
|---|---|---|---|---|---|
| Receive Error | RDRF | ORER | FER | PER | RSR to 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 |

Notes: 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.

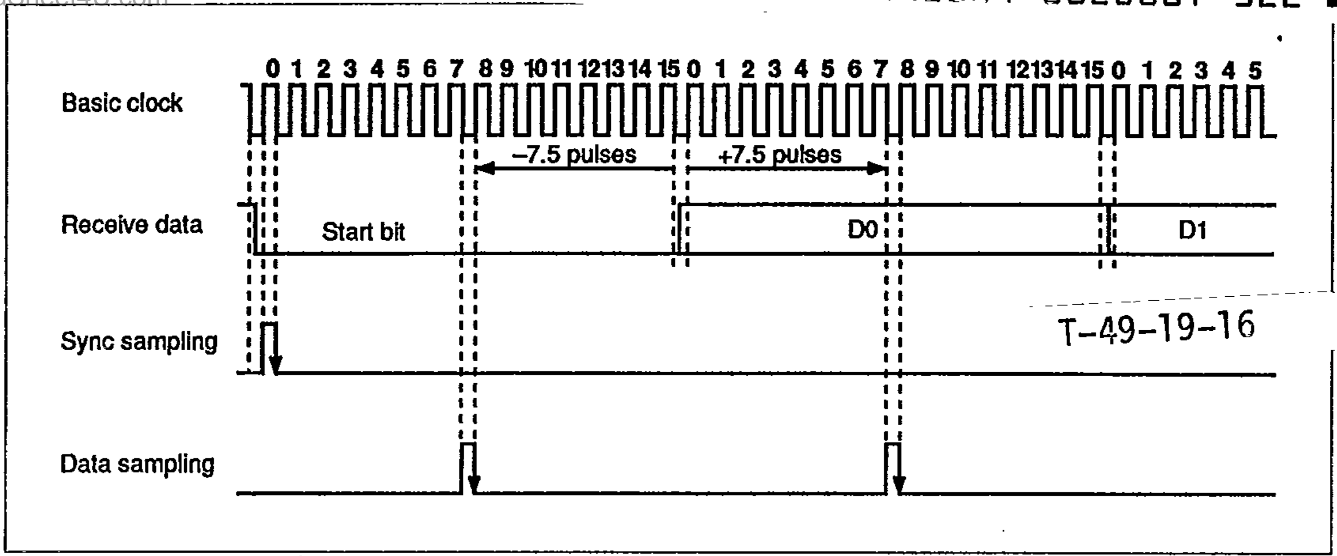
**Line Break Detection:** When the RXD pin receives a continuous stream of 0s in the asynchronous mode (line-break state), a framing error occurs because the SCI detects a 0 stop bit. The value H'00 is transferred from the RSR to the RDR. Software can detect the line-break state as a framing error accompanied 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.

**Sampling Timing and Receive Margin in Asynchronous Mode:** The serial clock used by the SCI in asynchronous mode runs at 16 times the bit 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 13-5.

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%.



T-49-19-16

Figure 13-5 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 serial clock to bit rate (N = 16)

D: Duty cycle of high or low clock pulses, whichever is longer (0.5 to 1.0)

L: Frame length (9 to 12)

F: Absolute value of clock frequency deviation

When D = 0.5 and F= 0:

$$M = (0.5 - 1/2 \times 16) \times 100\ [\%] = 46.875\% \quad (2)$$

T-49-19-16

# Section 14 A/D Converter

T-49-19-16

## 14.1 Overview

The H8/520 chip includes an analog-to-digital converter module which can be programmed for input of analog signals on up to four (or eight*) channels. A/D conversion is performed by the successive approximations method with 10-bit resolution.

### 14.1.1 Features

The features of the on-chip A/D module are as follows:

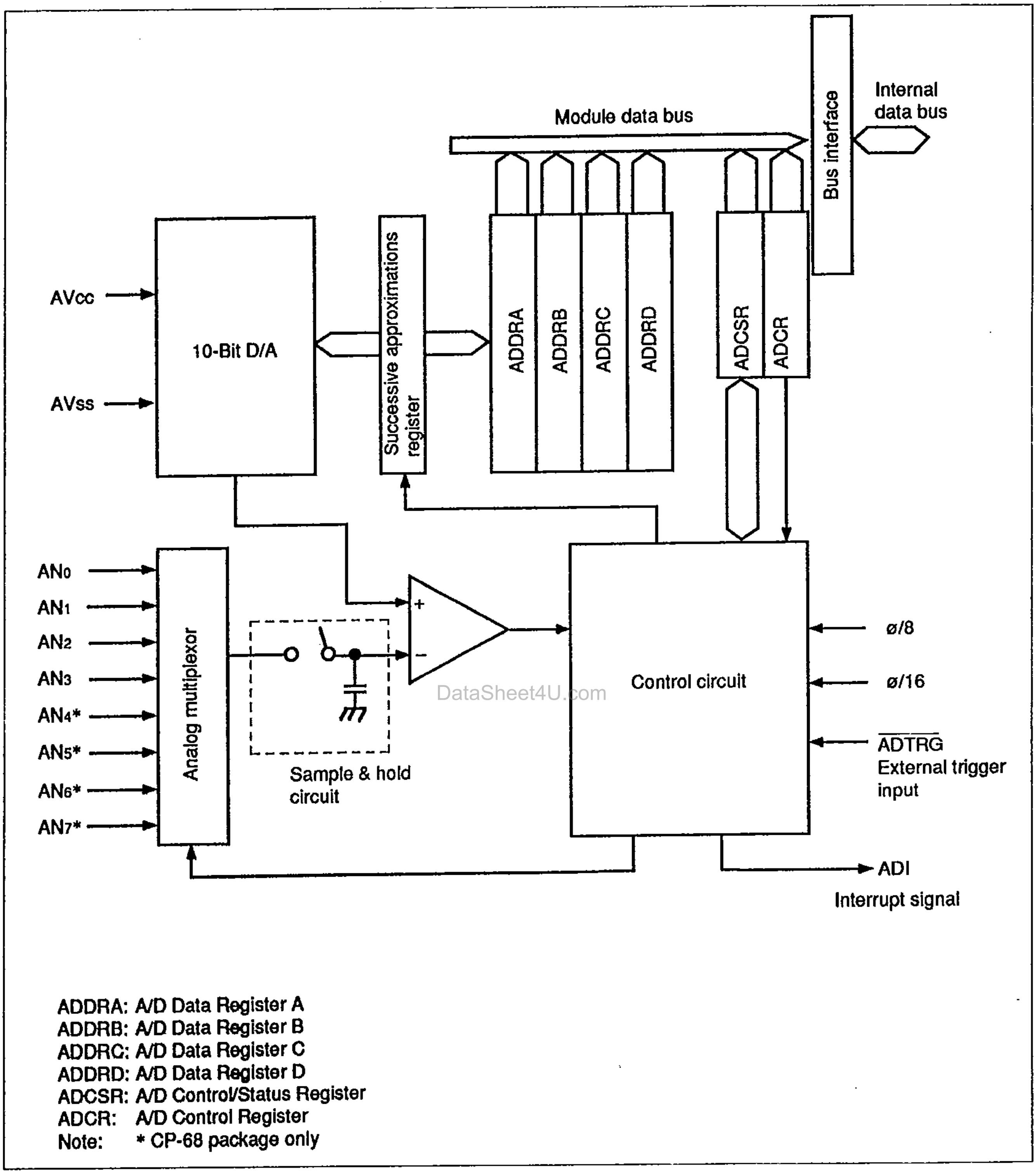
- Four (or eight*) analog input channels
- External trigger
  A/D conversion can be started by an external trigger input.
- Sample and hold circuit
- 10-Bit resolution
- Rapid conversion
  Conversion time is 13.8 µs per channel (at ø = 10 MHz)
- Single and scan modes
  –Single mode: A/D conversion is performed once.
  –Scan mode: A/D conversion is performed in a repeated cycle on one to four channels.
- Four 16-bit data registers
  These registers store A/D conversion results for up to four channels.
- A CPU interrupt (ADI) can be requested at the completion of each A/D conversion cycle.
  This interrupt can also be served by the on-chip data transfer controller (DTC), providing a convenient way to move results into memory.

Note: * CP-68 package only

### 14.1.2 Block Diagram

Figure 14-1 shows a block diagram of A/D converter.

T-49-19-16



ADDRA: A/D Data Register A
ADDRB: A/D Data Register B
ADDRC: A/D Data Register C
ADDRD: A/D Data Register D
ADCSR: A/D Control/Status Register
ADCR: A/D Control Register
Note: * CP-68 package only

**Figure 14-1 Block Diagram of A/D Converter**

### 14.1.3 Input Pins

T-49-19-16

Table 14-1 lists the input pins used by the A/D converter module.

The eight analog input pins provided in the CP-68 package are divided into two groups, consisting of analog inputs 0 to 3 (AN$_0$ to AN$_3$) and analog inputs 4 to 7 (AN$_4$ to AN$_7$), respectively.

**Table 14-1 A/D Input Pins**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Analog supply voltage | AV$_{CC}$ | Input | Power supply and reference voltage for the analog circuits. |
| Analog ground | AV$_{SS}$ | Input | Ground and reference voltage for the analog circuits. |
| Analog input 0 | AN$_0$ | Input | Analog input pins, group 0 |
| Analog input 1 | AN$_1$ | Input | |
| Analog input 2 | AN$_2$ | Input | |
| Analog input 3 | AN$_3$ | Input | |
| Analog input 4 | AN$_4$ | Input | |
| Analog input 5 | AN$_5$ | Input | Analog input pins, group 1[*1] |
| Analog input 6 | AN$_6$ | Input | |
| Analog input 7 | AN$_7$ | Input | |
| A/D external trigger input | $\overline{\text{ADTRG}}$ | Input | External trigger for starting A/D conversion[*2] |

Notes: 1. CP-68 package only.
2. Not available in MCU mode 3 because this pin is used for the page address bus (A$_{18}$).

### 14.1.4 Register Configuration

T-49-19-16

Table 14-2 lists the registers of the A/D converter module.

**Table 14-2 A/D Registers**

| Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| A/D data register A (High) | ADDRA (H) | R | H'00 | H'FFE0 |
| A/D data register A (Low) | ADDRA (L) | R | H'00 | H'FFE1 |
| A/D data register B (High) | ADDRB (H) | R | H'00 | H'FFE2 |
| A/D data register B (Low) | ADDRB (L) | R | H'00 | H'FFE3 |
| A/D data register C (High) | ADDRC (H) | R | H'00 | H'FFE4 |
| A/D data register C (Low) | ADDRC (L) | R | H'00 | H'FFE5 |
| A/D data register D (High) | ADDRD (H) | R | H'00 | H'FFE6 |
| A/D data register D (Low) | ADDRD (L) | R | H'00 | H'FFE7 |
| A/D control/status register | ADCSR | R/(W)* | H'00 | H'FFE8 |
| A/D control register | ADCR | R/W | H'7F | H'FFE9 |

Note: * Software can write a 0 to clear the status flag in bit 7 but cannot write a 1.

## 14.2 Register Descriptions

### 14.2.1 A/D Data Registers (ADDR)—H'FFE0 to H'FFE7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADDRn H | AD$_9$ | AD$_8$ | AD$_7$ | AD$_6$ | AD$_5$ | AD$_4$ | AD$_3$ | AD$_2$ |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

(n = A to D)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADDRn L | AD$_1$ | AD$_0$ | — | — | — | — | — | — |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R | R | R | R | R | R | R | R |

(n = A to D)

The four A/D data registers (ADDRA to ADDRD) are 16-bit read-only registers that store the results of A/D conversion.

Each result consist of 10 bits. The first 8 bits are stored in the upper byte of the data register corresponding to the selected channel. The last two bits are stored in the lower data register byte. The data registers are assigned to analog input channels as indicated in table 14-3.

The A/D data registers are always readable by the CPU. The upper byte can be read directly. The lower byte is read via a temporary register. See section 14-3, "CPU Interface", for details.

The unused bits (bits 5 to 0) of the lower data register byte are always read as 0.

The A/D data registers are initialized to H'0000 at a reset and in the standby modes.

**Table 14-3 Assignment of Data Registers to Analog Input Channels**

| Analog Input Channel | | |
|---|---|---|
| **Group 0** | **Group 1 *** | **A/D Data Register** |
| AN0 | AN4 | ADDRA |
| AN1 | AN5 | ADDRB |
| AN2 | AN6 | ADDRC |
| AN3 | AN7 | ADDRD |

Note: * CP-68 package only.

### 14.2.2 A/D Control/Status Register (ADCSR)—H'FFE8

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADF | ADIE | ADST | SCAN | CKS | CH2 | CH1 | CH0 |
| 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: * Software can write a 0 in bit 7 to clear the flag, but cannot write a 1 in this bit.

The A/D control/status register (ADCSR) is an 8-bit readable/writable register that controls the operation of the A/D converter module.

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

**Bit 7—A/D End Flag (ADF):** This status flag indicates the end of one cycle of A/D conversion.

| Bit 7<br>ADF | Description | |
|---|---|---|
| 0 | This bit is cleared from 1 to 0 when:<br>1. The chip is reset or placed in a standby mode.<br>2. The CPU reads the ADF bit after the ADF bit is set to 1, then writes a 0 in this bit.<br>3. An A/D interrupt is served by the data transfer controller (DTC). | (Initial value) |
| 1 | This bit is set to 1 at the following times:<br>1. Single mode: when one A/D conversion is completed.<br>2. Scan mode: when inputs on all selected channels have been converted. | |

**Bit 6—A/D Interrupt Enable (ADI):** This bit selects whether to request an A/D interrupt (ADI) when A/D conversion is completed.

| Bit 6<br>ADIE | Description | |
|---|---|---|
| 0 | The A/D interrupt request (ADI) is disabled. | (Initial value) |
| 1 | The A/D interrupt request (ADI) is enabled. | |

**Bit 5—A/D Start (ADST):** The A/D converter operates while this bit is set to 1. In the single mode, this bit is automatically cleared to 0 at the end of each A/D conversion.

| Bit 5<br>ADST | Description | |
|---|---|---|
| 0 | A/D conversion is halted. | (Initial value) |
| 1 | 1. Single mode: One A/D conversion is performed. The ADST bit is automatically cleared to 0 at the end of the conversion.<br>2. Scan mode: A/D conversion starts and continues cyclically on the selected channels until the ADST bit is cleared to 0. | |

**Bit 4—Scan Mode (SCAN):** This bit selects the scan mode or single mode of operation. See section 14.4, "Operation", for descriptions of these modes. The mode should be changed only when the ADST bit is cleared to 0.

| Bit 4<br>SCAN | Description | |
|---|---|---|
| 0 | Single mode | (Initial value) |
| 1 | Scan mode | |

**Bit 3—Clock Select (CKS):** This bit controls the A/D conversion time.

The conversion time should be changed only when the ADST bit is cleared to 0.

| Bit 3<br>CKS | Description | | |
|---|---|---|---|
| 0 | Conversion time = 274 states | (maximum) | (Initial value) |
| 1 | Conversion time = 138 states | (maximum) | |

**Bits 2 to 0—Channel Select 2 to 0 (CH2 to CH0):** These bits and the SCAN bit combine to select one or more analog input channels.

The channel selection should be changed only when the ADST bit is cleared to 0.

| Group Select | Channel Select | | Selected Channels | |
|---|---|---|---|---|
| CH2 | CH1 | CH0 | Single Mode | Scan Mode |
| 0 | 0 | 0 | $AN_0$ | $AN_0$ |
| | 0 | 1 | $AN_1$ | $AN_0$ and $AN_1$ |
| | 1 | 0 | $AN_2$ | $AN_0$ to $AN_2$ |
| | 1 | 1 | $AN_3$ | $AN_0$ to $AN_3$ |
| 1 | 0 | 0 | $AN_4$* | $AN_4$* |
| | 0 | 1 | $AN_5$* | $AN_4$ and $AN_5$* |
| | 1 | 0 | $AN_6$* | $AN_4$ to $AN_6$* |
| | 1 | 1 | $AN_7$* | $AN_4$ to $AN_7$* |

Note: * CP-68 package only

### 14.2.3 A/D Control Register (ADCR)—H'FFE9

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

The A/D control register (ADCR) is an 8-bit readable/writable register that enables or disables the A/D external trigger signal.

The ADCR is initialized to H'7F at a reset and in the standby modes.

**Bit 7—Trigger Enable (TRGE):** This bit enables the $\overline{\text{ADTRG}}$ (A/D external trigger) signal. When enabled, a high-to-low transition of $\overline{\text{ADTRG}}$ sets the ADST bit, starting A/D conversion.

T-49-19-16

| Bit 7<br>TRGE | Description | |
|---|---|---|
| 0 | A/D external trigger is disabled. $\overline{\text{ADTRG}}$ does not set the ADST bit. | (Initial value) |
| 1 | A/D external trigger is enabled. A high-to-low transition of $\overline{\text{ADTRG}}$ sets the ADST bit. | |

**Bits 6 to 0—Reserved:** These bits cannot be modified and are always read as 1.

## 14.3 CPU Interface

The A/D data registers (ADDRA to ADDRD) are 16-bit registers, but they are accessed via an 8-bit module data bus. Accordingly, the upper byte of each register can be read directly, but the lower byte is accessed through an 8-bit temporary register (TEMP).

When the CPU or DTC reads the upper byte of an A/D data register, at the same time as the upper byte is placed on the internal data bus, the lower byte is transferred to TEMP. When the lower byte is accessed, the value in TEMP is placed on the internal data bus.

A program that requires all 10 bits of an A/D result should perform word access, or should read first the upper byte, then the lower byte of the A/D data register. Either way, it is assured of obtaining consistent data. Consistent data are not assured if the program reads the lower byte first.

A program that requires only 8-bit A/D accuracy should perform byte access to the upper byte of the A/D data register. The value in TEMP can be left unread.

Figure 14-2 shows the data flow when the CPU (or DTC) reads an A/D data register.

T-49-19-16

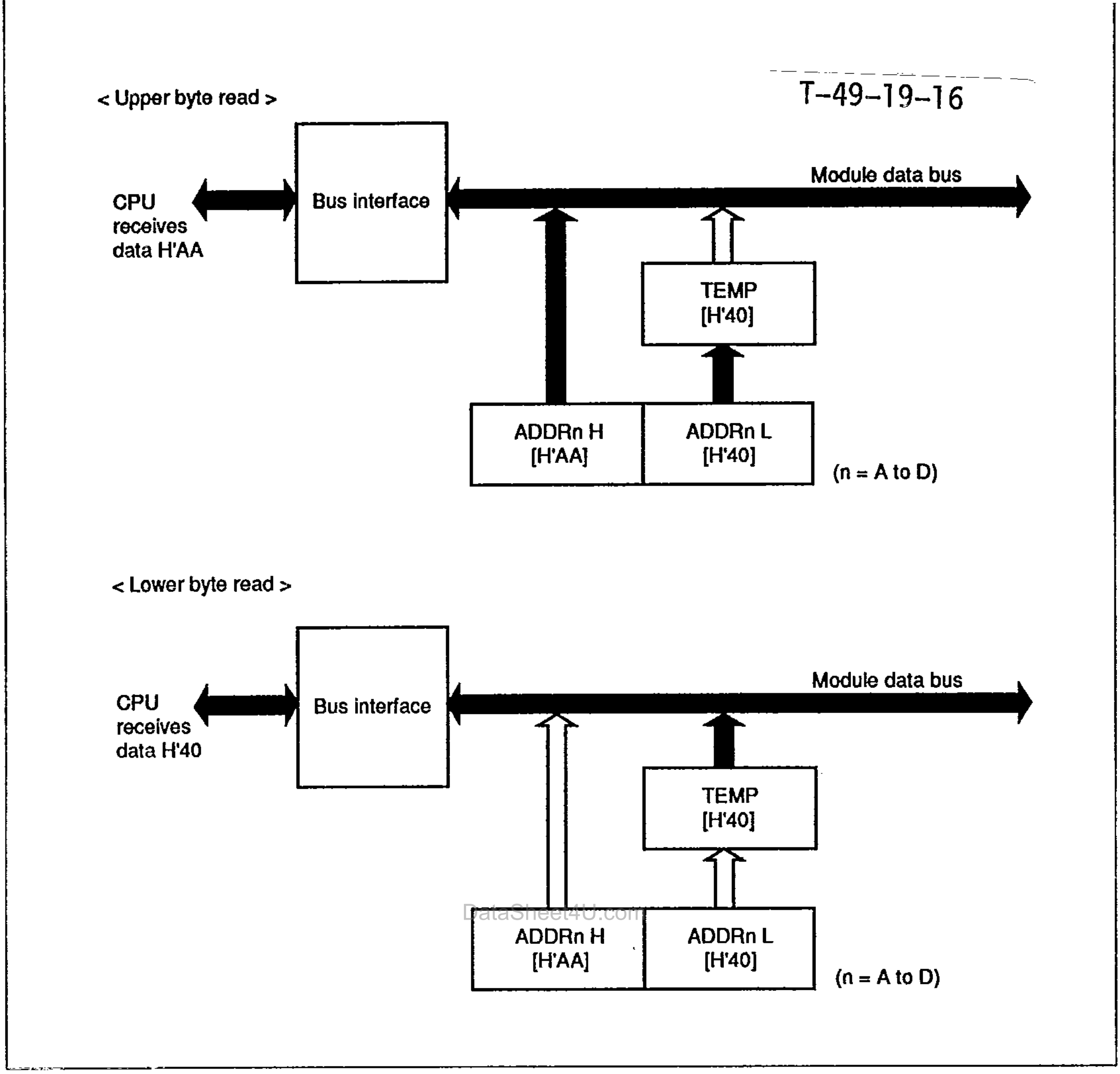


**Figure 14-2 Read Access to A/D Data Register (When Register Contains H'AA40)**

## 14.4 Operation

The A/D converter performs 10 successive approximations to obtain a result ranging from H'0000 (corresponding to AVSS) to H'FFC0 (corresponding to AVCC). Only the first 10 bits of the result are significant.

The response of the A/D converter is shown below. H'FFC0 corresponds to voltages of approximately 0.999AVcc and above.

T-49-19-16

The A/D converter module can be programmed to operate in single mode or scan mode as explained below.

### 14.4.1 Single Mode

The single mode is suitable for obtaining a single data value from a single channel. A/D conversion starts when the ADST bit is set to 1 by software or external trigger input. During the conversion process the ADST bit remains set to 1. When conversion is completed, the ADST bit is automatically cleared to 0.

When the conversion is completed, the ADF bit is set to 1. If the interrupt enable bit (ADIE) is also set to 1, an A/D conversion end interrupt (ADI) is requested, so that the converted data can be processed by an interrupt-handling routine. Alternatively, the interrupt can be served by the data transfer controller (DTC).

When an A/D interrupt is served by the DTC, the DTC automatically clears the ADF bit to 0. When an A/D interrupt is served by the CPU, however, the ADF bit remains set until the CPU reads the ADCSR, then writes a 0 in the ADF bit.

Before selecting the single mode, clock, and analog input channel, software should clear the ADST bit to 0 to make sure the A/D converter is stopped. Changing the mode, clock, or channel selection while A/D conversion is in progress can lead to conversion errors.

The following example explains the A/D conversion process in single mode when channel 1 (AN1) is selected and external triggering is not used. Figure 14-3 shows the corresponding timing chart.

1. Software clears the ADST bit to 0, then selects the single mode (SCAN = 0) and channel 1 (CH2 to CH0 = 001), enables the A/D interrupt request (ADIE = 1), and sets the ADST bit to 1 to start A/D conversion.

   **Coding Example:** (when using the slow clock, CKS = 0)

   ```
   BCLR #7, @H'FFE9
   BCLR #5, @H'FFE8
   MOV.B #H'61, @H'FFE8
   ```

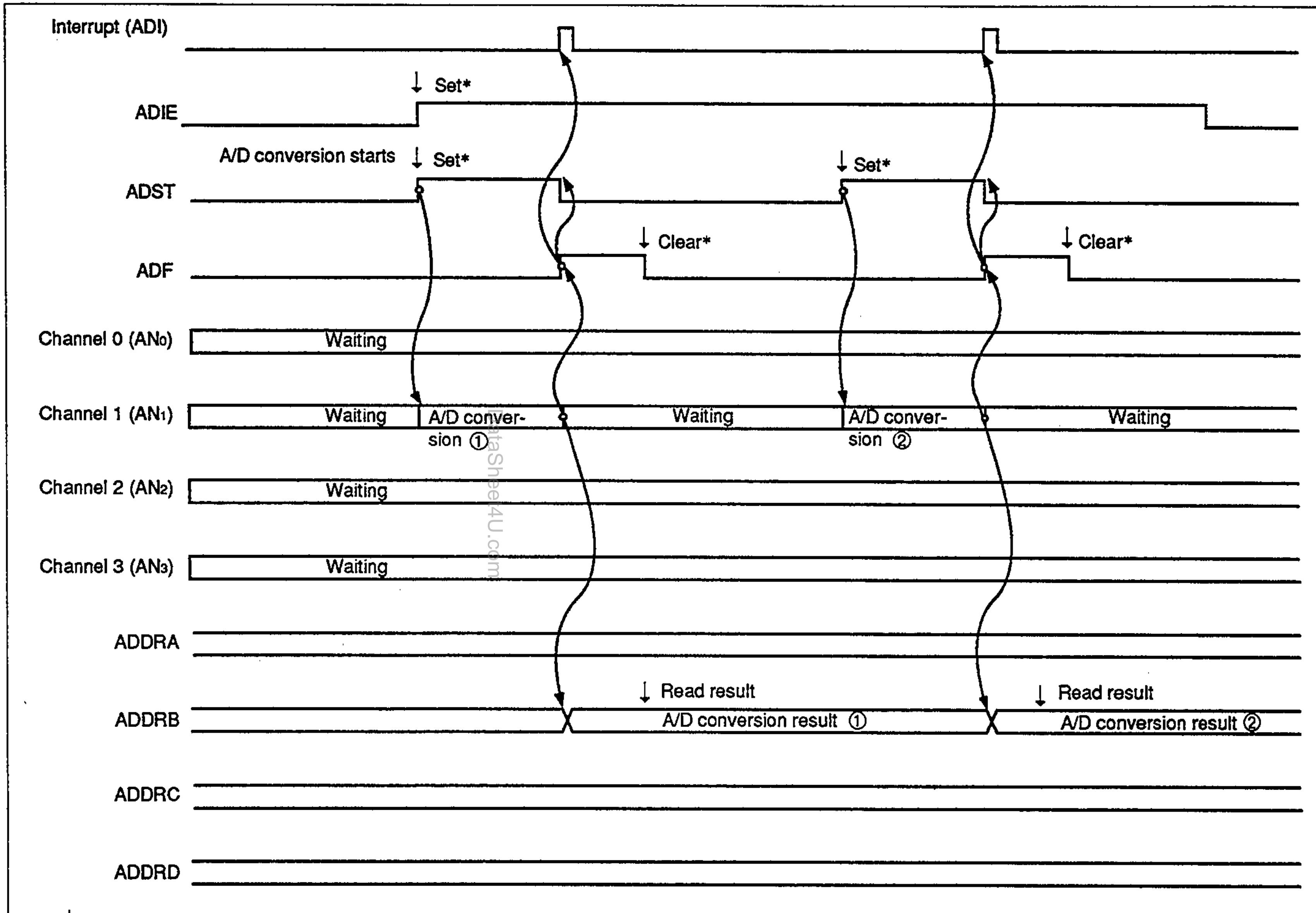
2. The A/D converter samples the $AN_1$ input and converts the voltage level to a digital value. At the end of the conversion process the A/D converter transfers the result to register ADDRB, sets the ADF bit to 1, clears the ADST bit to 0, and halts.
3. ADF = 1 and ADIE = 1, so an A/D interrupt is requested.
4. The user-coded A/D interrupt-handling routine is started.
5. The interrupt-handling routine reads the ADCSR value, then writes a 0 in the ADF bit to clear this bit to 0. The reading and writing can be done with a single BCLR #7, @H'FFE8 instruction.
6. The interrupt-handling routine reads and processes the A/D conversion result.
7. The routine ends.

T-49-19-16

Steps 2 to 7 can now be repeated by setting the ADST bit to 1 again.

If the ADI bit in data transfer enable register D (bit 0 at address H'FFF7) is set to 1, the interrupt is served by the data transfer controller (DTC). Steps 4 to 7 then change as follows.

4'. The DTC is started.
5'. The DTC automatically clears the ADF bit to 0.
6'. The DTC transfers the A/D conversion result from ADDRB to a specified destination address.
7'. The DTC ends.



* ↓ indicates execution of a software instruction

Figure 14-3 A/D Operation in Single Mode (When Channel 1 is Selected)

### 14.4.2 Scan Mode

The scan mode can be used to monitor analog inputs on one or more channels. When the ADST bit is set to 1 by software or by external trigger input, A/D conversion starts from the first channel (AN0) in the scan group.*

If the scan group includes more than one channel (i.e., if bit CH1 or CH0 is set), conversion of the next channel begins as soon as conversion of the first channel ends.

Conversion of the selected channels continues cyclically until the ADST bit is cleared to 0. The conversion results are placed in the data registers corresponding to the selected channels.

Before selecting the scan mode, clock, and analog input channels, software should clear the ADST bit to 0 to make sure the A/D converter is stopped. Changing the mode, clock, or channel selection while A/D conversion is in progress can lead to conversion errors.

The following example explains the A/D conversion process when three channels in group 0 are selected (AN0, AN1, and AN2) and external triggering is not used. Figure 14-4 shows the timing.

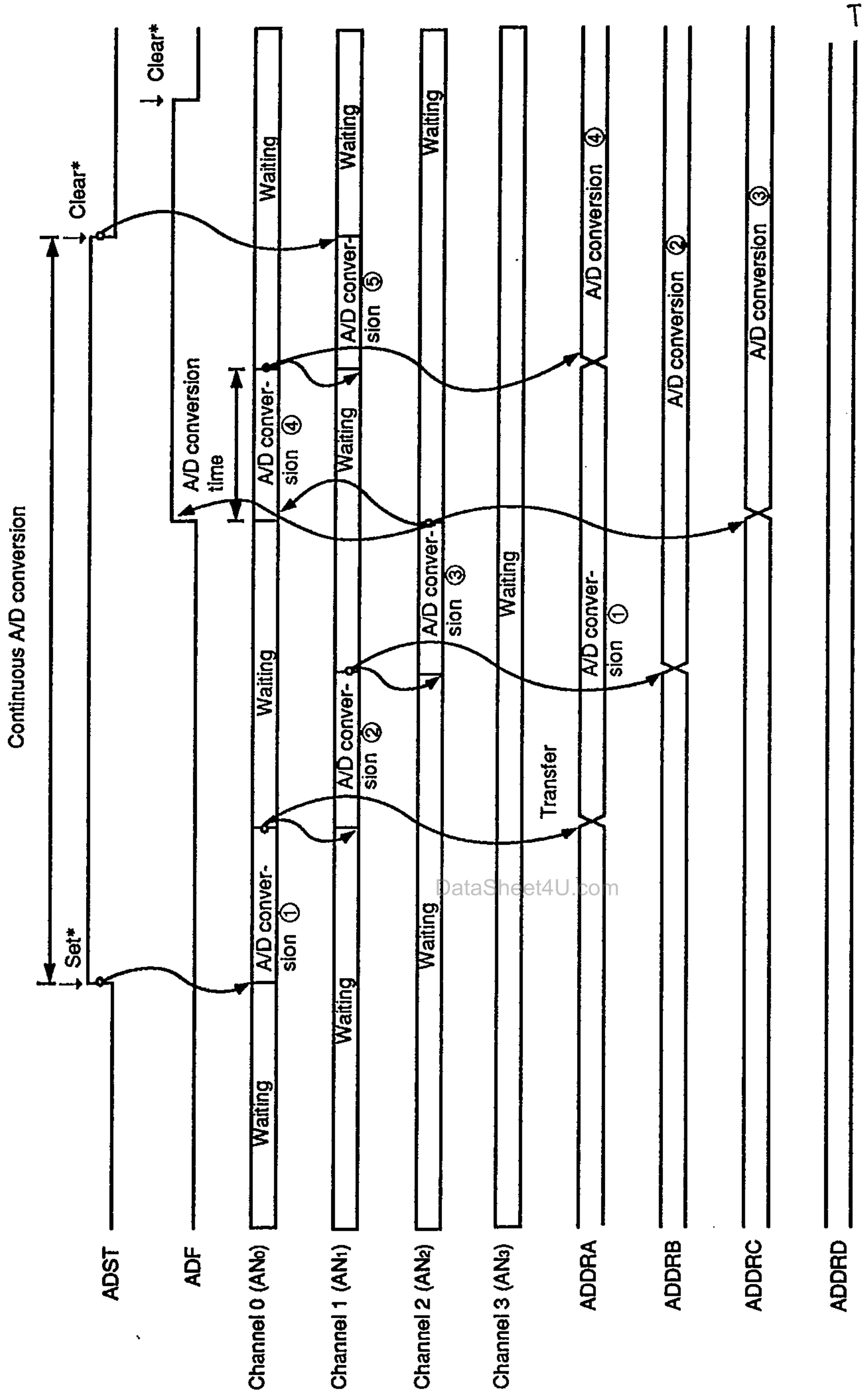
1. Software clears the ADST bit to 0, then selects the scan mode (SCAN = 1), scan group 0 (CH2 = 0), and analog input channels AN0 to AN2 (CH1 = 1, CH0 = 0) and sets the ADST bit to 1 to start A/D conversion.
   **Coding Example:** (with slow clock and ADI interrupt enabled)
   ```
   BCLR #7, @H'FFE9
   BCLR #5, @H'FFE8
   MOV.B #H'72, @FFE8
   ```
2. The A/D converter samples the input at AN0, converts the voltage level to a digital value, and transfers the result to register ADDRA.
3. Next the A/D converter samples and converts AN1 and transfers the result to ADDRB. Then it samples and converts AN2 and transfers the result to ADDRC.
4. After all selected channels (AN0 to AN2) have been converted, the AD converter sets the ADF bit to 1. If the ADIE bit is set to 1, an A/D interrupt (ADI) is requested. Then the A/D converter begins converting AN0 again.
5. Steps 2 to 4 are repeated cyclically as long as the ADST bit remains set to 1.

To stop the A/D converter, software must clear the ADST bit to 0. The data currently undergoing conversion when the ADST bit is cleared are ignored. The A/D data registers retain the last completed conversion results.

Regardless of which channel is being converted when the ADST bit is cleared to 0, when the ADST bit is set to 1 again, conversion begins from the the first selected channel (AN0 or AN4).

Note: * In the CP-68 package, the first channel is AN0 if CH2 = 0, and AN4 if CH2 = 1.

T-49-19-16



Note: *↓ Indicates execution of a software instruction

Figure 14-4 A/D Operation in Scan Mode (When Channels 0 to 2 are Selected)

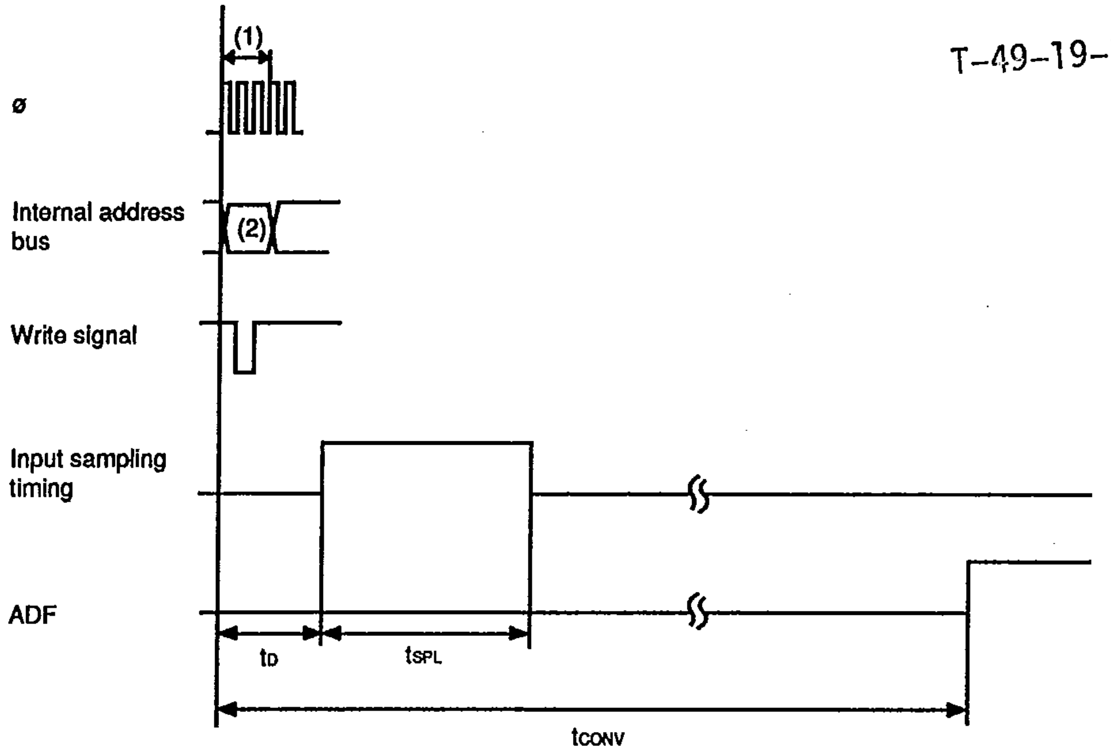
### 14.4.3 Input Sampling Time and A/D Conversion Time

The A/D converter includes a built-in sample-and-hold circuit. Sampling of the input starts at a time $t_D$ after the ADST bit is set to 1. The sampling process lasts for a time $t_{SPL}$. The actual A/D conversion begins after sampling is completed. Figure 14-5 shows the timing of these steps, and table 15-4 lists the total conversion times ($t_{CONV}$) for the single mode.

The total conversion time includes $t_D$ and $t_{SPL}$. The purpose of $t_D$ is to synchronize the ADCSR write time with the A/D conversion process, so the length of $t_D$ is variable. The total conversion time therefore varies within the minimum to maximum ranges indicated in table 14-4.

In the scan mode, the ranges given in table 14-4 apply to the first conversion. The length of the second and subsequent conversion processes is fixed at 256 states (when CKS = 0) or 128 states (when CKS = 1).

T-49-19-16



(1) : ADCSR write cycle
(2) : ADCSR address
$t_D$ : Synchronization delay
$t_{SPL}$ : Input sampling time
$t_{CONV}$ : Total A/D conversion time

Figure 14-5 A/D Conversion Timing

### Table 14-4 A/D Conversion Time (Single Mode)

| | | CKS = 0 | | | CKS = 1 | | |
|---|---|---|---|---|---|---|---|
| **Item** | **Symbol** | **min** | **typ** | **max** | **min** | **typ** | **max** |
| Synchronization delay | $t_D$ | 18 | — | 33 | 10 | — | 17 |
| Input sampling time | $t_{SPL}$ | — | 63 | — | — | 31 | — |
| Total A/D conversion time | $t_{CONV}$ | 259 | — | 274 | 131 | — | 138 |

Note: Values in the table are numbers of states.

### 14.4.4 External Triggering of A/D Conversion

T-49-19-16

The A/D conversion process can be started by an external trigger input.

External trigger input is enabled at the $\overline{\text{ADTRG}}$ pin when the TRGE bit in the ADCR is set to 1. 1.0 ø clock cycles after the $\overline{\text{ADTRG}}$ input is sampled, the ADST bit in the ADCSR is set to 1 and A/D conversion commences.

The timing of external triggering is shown in figure 14-6.

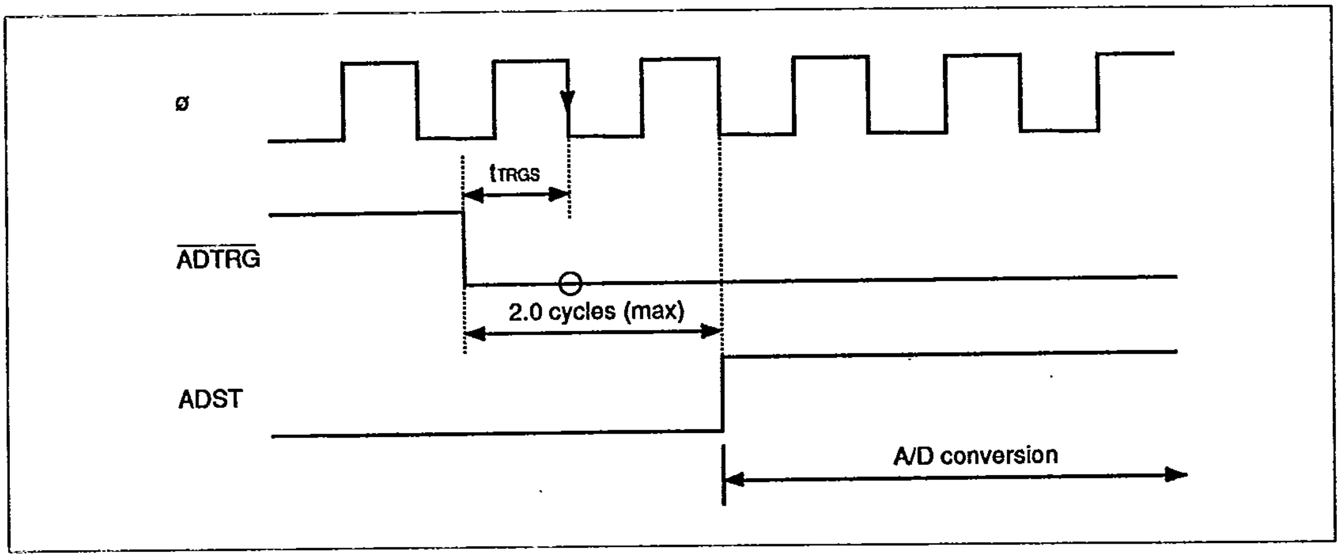


Figure 14-6 Timing of Setting of ADST Bit

## 14.5 Interrupts and the Data Transfer Controller

The ADI interrupt request is enabled or disabled by the ADIE bit in the ADCSR.

When the ADI bit in data transfer enable register DTED (bit 0 at address H'FFF7) is set to 1, the ADI interrupt is served by the data transfer controller. The DTC can be used to transfer A/D results to a buffer in memory, or to an I/O port. The DTC automatically clears the ADF bit to 0.

Note: In scan mode, the DTC can transfer data for only one channel per interrupt, even if two or more channels are selected.

T-49-19-16

# Section 15 RAM

T-49-19-16

## 15.1 Overview

The H8/520 includes 512 bytes of on-chip static RAM, 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 is assigned to addresses H'FD80 to H'FF7F in the chip's address space. A RAM control register (RAMCR) can enable or disable the on-chip RAM, permitting these addresses to be allocated to external memory instead, if so desired.

### 15.1.1 Block Diagram

Figure 15-1 shows a block diagram of the on-chip RAM.

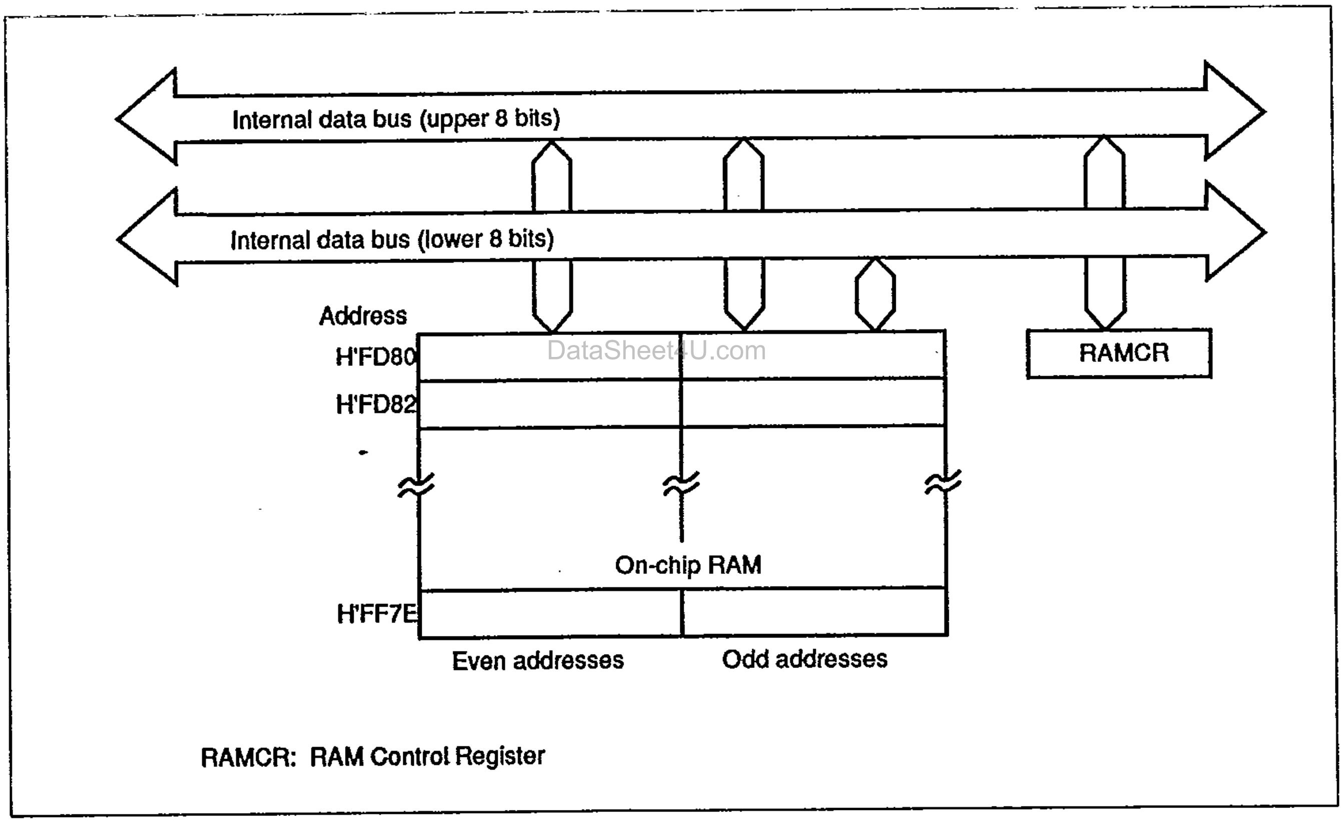


Figure 15-1 Block Diagram of On-Chip RAM

### 15.1.2 Register Configuration

T-49-19-16

The on-chip RAM is controlled by the register described in table 15-1.

**Table 15-1 RAM Control Register**

| Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| RAM control register | RAMCR | R/W | H'FF | H'FFF9 |

## 15.2 RAM Control Register (RAMCR)

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

The RAM control register (RAMCR) is an 8-bit register that enables or disables the on-chip RAM.

**Bit 7—RAM Enable (RAME):** This bit enables or disables the on-chip RAM.

The RAME bit is initialized by a reset. It is not initialized in the software standby mode.

| Bit 7<br>RAME | Description | |
|---|---|---|
| 0 | On-chip RAM is disabled. | |
| 1 | On-chip RAM is enabled. | (Initial value) |

**Bits 6 to 0—Reserved:** These bits cannot be modified and are always read as 1.

## 15.3 Operation

### 15.3.1 Expanded Modes (Modes 1, 2, 3, and 4)

If the RAME bit is set to 1, accesses to addresses H'FD80 to H'FF7F are directed to the on-chip RAM. If the RAME bit is cleared to 0, accesses to addresses H'FD80 to H'FF7F are directed to the external data bus.

### 15.3.2 Single-Chip Mode (Mode 7)

T-49-19-16

If the RAME bit is set to 1, accesses to addresses H'FD80 to H'FF7F are directed to the on-chip RAM. If the RAME bit is cleared to 0, access of any type (instruction fetch or data read or write) to addresses H'FD80 to H'FF7F causes an address error and initiates the CPU's exception-handling sequence.

T-49-19-16

# Section 16 ROM

T-49-19-16

## 16.1 Overview

The H8/520 includes 16 kbytes of high-speed on-chip ROM. The on-chip ROM is connected to the CPU via a 16-bit data bus and is accessed in two states.

Users wishing to program the chip themselves can request electrically programmable ROM (PROM). The PROM version of the H8/520 has a PROM mode in which the chip can be programmed with a standard, external PROM writer. The chip is also available with masked ROM.

The on-chip ROM is enabled or disabled depending on the MCU operating mode, which is determined by the inputs at the mode pins when the chip comes out of the reset state. See table 16-1.

**Table 16-1 ROM Usage in Each MCU Mode**

| | Mode Pins | | | |
|---|---|---|---|---|
| Mode | $MD_2$ | $MD_1$ | $MD_0$ | ROM |
| Mode 1 (expanded minimum mode) | 0 | 0 | 1 | Disabled (external addresses) |
| Mode 2 (expanded minimum mode) | 0 | 1 | 0 | Enabled |
| Mode 3 (expanded maximum mode) | 0 | 1 | 1 | Disabled (external addresses) |
| Mode 4 (expanded maximum mode) | 1 | 0 | 0 | Enabled |
| Mode 7 (single-chip mode) | 1 | 1 | 1 | Enabled |

### 16.1.1 Block Diagram

Figure 16-1 shows the block diagram of the on-chip ROM.

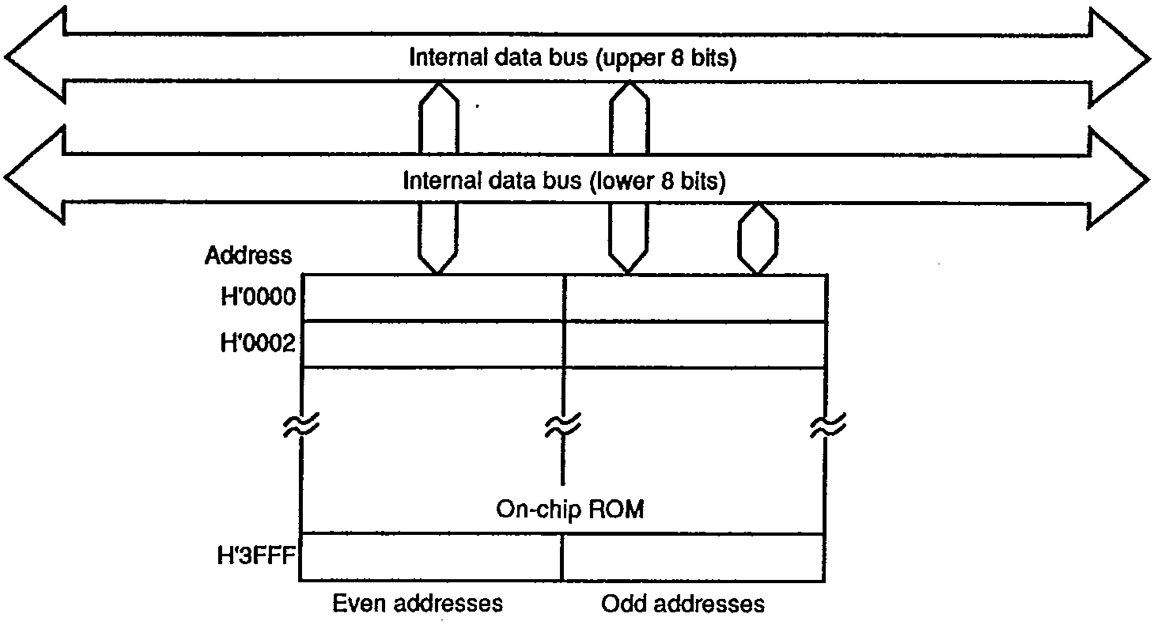


Figure 16-1 Block Diagram of On-Chip ROM

## 16.2 PROM Mode

### 16.2.1 PROM Mode Setup

The PROM version of the H8/520 has a PROM mode in which 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 16-2 to the mode pins ($MD_2$ to $MD_0$) and pins $P5_1$ and $P5_0$.

**Table 16-2 Selection of PROM Mode**

| Pin | Input |
|---|---|
| $MD_1$ | Low |
| $MD_2$ and $MD_0$ | High |
| $P5_1$ and $P5_0$ | High |

### 16.2.2 Socket Adapter Pin Arrangements and Memory Map

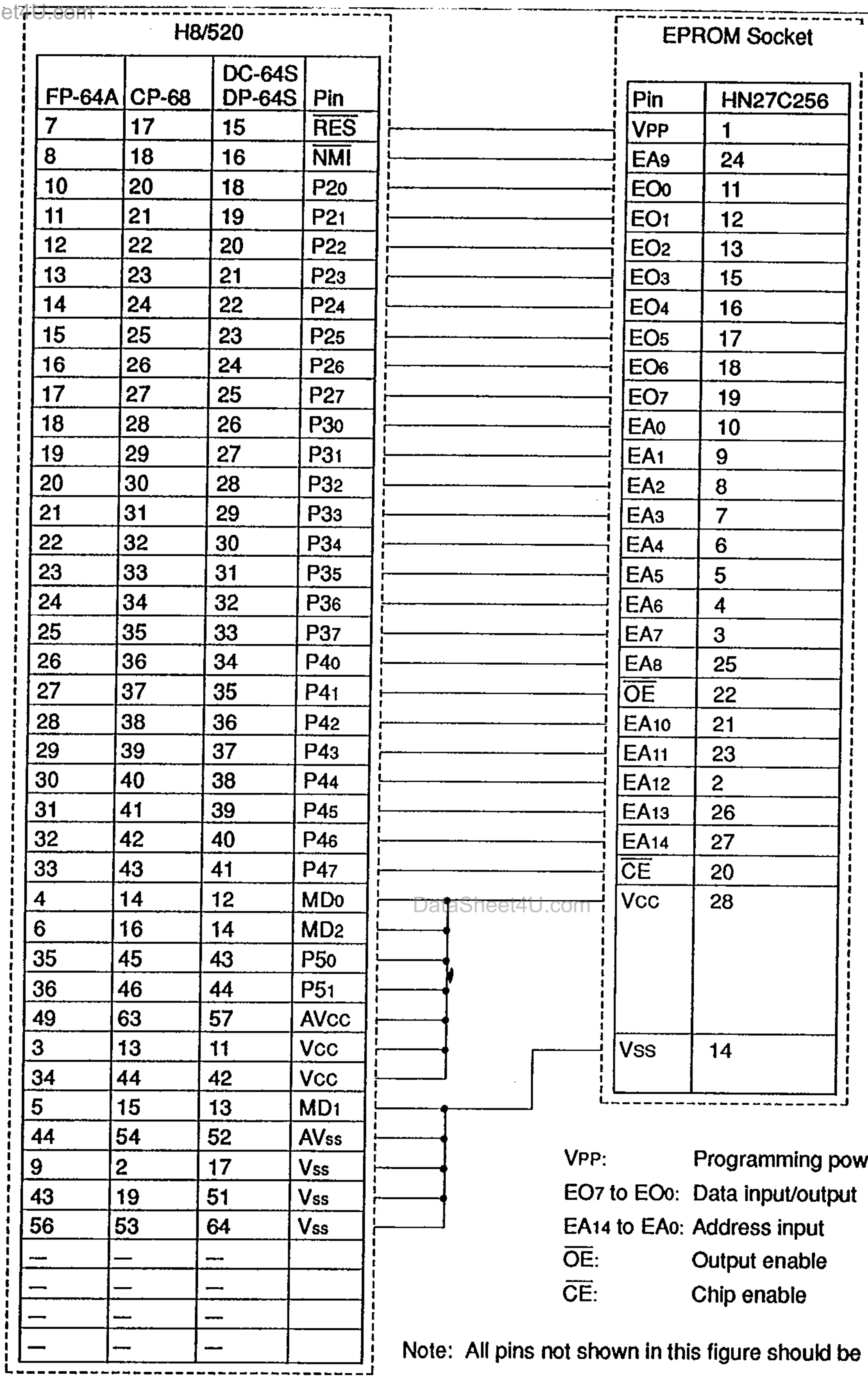
T-49-19-16

The H8/520 can be programmed with a general-purpose PROM writer by attaching a socket adapter as listed in table 16-3. The socket adapter depends on the type of package. Figure 16-2 shows the socket adapter pin arrangements by giving the correspondence between H8/520 pins and HN27C256 pin functions. Figure 16-3 is a memory map.

Table 16-3 Socket Adapter

| Package | Socket Adapter |
|---|---|
| 64-Pin windowed shrink DIP (DC-64S) | HS528ESS01H |
| 64-Pin shrink DIP (DP-64S) | |
| 64-Pin QFP (FP-64A) | HS528ESH01H |
| 68-Pin PLCC (CP-68) | HS528ESC01H |

T-49-19-16



H8/520

| FP-64A | CP-68 | DC-64S<br>DP-64S | Pin |
|---|---|---|---|
| 7 | 17 | 15 | $\overline{\text{RES}}$ |
| 8 | 18 | 16 | $\overline{\text{NMI}}$ |
| 10 | 20 | 18 | $P2_0$ |
| 11 | 21 | 19 | $P2_1$ |
| 12 | 22 | 20 | $P2_2$ |
| 13 | 23 | 21 | $P2_3$ |
| 14 | 24 | 22 | $P2_4$ |
| 15 | 25 | 23 | $P2_5$ |
| 16 | 26 | 24 | $P2_6$ |
| 17 | 27 | 25 | $P2_7$ |
| 18 | 28 | 26 | $P3_0$ |
| 19 | 29 | 27 | $P3_1$ |
| 20 | 30 | 28 | $P3_2$ |
| 21 | 31 | 29 | $P3_3$ |
| 22 | 32 | 30 | $P3_4$ |
| 23 | 33 | 31 | $P3_5$ |
| 24 | 34 | 32 | $P3_6$ |
| 25 | 35 | 33 | $P3_7$ |
| 26 | 36 | 34 | $P4_0$ |
| 27 | 37 | 35 | $P4_1$ |
| 28 | 38 | 36 | $P4_2$ |
| 29 | 39 | 37 | $P4_3$ |
| 30 | 40 | 38 | $P4_4$ |
| 31 | 41 | 39 | $P4_5$ |
| 32 | 42 | 40 | $P4_6$ |
| 33 | 43 | 41 | $P4_7$ |
| 4 | 14 | 12 | $MD_0$ |
| 6 | 16 | 14 | $MD_2$ |
| 35 | 45 | 43 | $P5_0$ |
| 36 | 46 | 44 | $P5_1$ |
| 49 | 63 | 57 | AVcc |
| 3 | 13 | 11 | Vcc |
| 34 | 44 | 42 | Vcc |
| 5 | 15 | 13 | $MD_1$ |
| 44 | 54 | 52 | AVss |
| 9 | 2 | 17 | Vss |
| 43 | 19 | 51 | Vss |
| 56 | 53 | 64 | Vss |
| — | — | — | |
| — | — | — | |
| — | — | — | |
| — | — | — | |

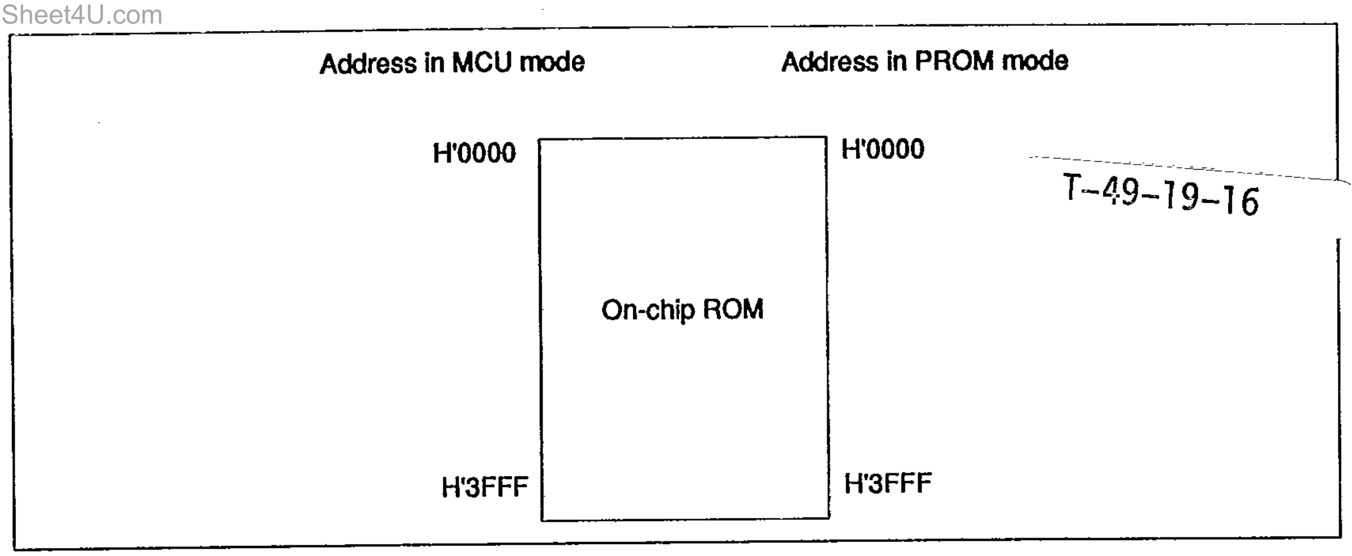
EPROM Socket

| Pin | HN27C256 |
|---|---|
| VPP | 1 |
| $EA_9$ | 24 |
| $EO_0$ | 11 |
| $EO_1$ | 12 |
| $EO_2$ | 13 |
| $EO_3$ | 15 |
| $EO_4$ | 16 |
| $EO_5$ | 17 |
| $EO_6$ | 18 |
| $EO_7$ | 19 |
| $EA_0$ | 10 |
| $EA_1$ | 9 |
| $EA_2$ | 8 |
| $EA_3$ | 7 |
| $EA_4$ | 6 |
| $EA_5$ | 5 |
| $EA_6$ | 4 |
| $EA_7$ | 3 |
| $EA_8$ | 25 |
| $\overline{\text{OE}}$ | 22 |
| $EA_{10}$ | 21 |
| $EA_{11}$ | 23 |
| $EA_{12}$ | 2 |
| $EA_{13}$ | 26 |
| $EA_{14}$ | 27 |
| $\overline{\text{CE}}$ | 20 |
| Vcc | 28 |
| Vss | 14 |

| | |
|---|---|
| VPP: | Programming power (12.5 V) |
| $EO_7$ to $EO_0$: | Data input/output |
| $EA_{14}$ to $EA_0$: | Address input |
| $\overline{\text{OE}}$: | Output enable |
| $\overline{\text{CE}}$: | Chip enable |

Note: All pins not shown in this figure should be left open.

Figure 16-2 Socket Adapter Pin Arrangements



T-49-19-16

Figure 16-3 Memory Map in PROM Mode

## 16.3 Programming

The write, verify, and inhibited sub-modes of the PROM mode are selected as shown in table 16-4.

Table 16-4 Selection of Sub-Modes in PROM Mode

| | Pins | | | | | |
|---|---|---|---|---|---|---|
| **Mode** | $\overline{CE}$ | $\overline{OE}$ | $V_{PP}$ | $V_{CC}$ | $O_7$ to $O_0$ | $A_{14}$ to $A_0$ |
| 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 |
| Read | Low | Low | $V_{PP}$ | $V_{CC}$ | Data output | Address input |

Note: The $V_{PP}$ and $V_{CC}$ pins must be held at the $V_{PP}$ and $V_{CC}$ voltage levels.

The H8/520 PROM uses the same, standard read/write specifications as the HN27C256 and HN27256.

### 16.3.1 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.

Figure 16-4 shows the basic high-speed programming flowchart.

Tables 16-5 and 16-6 list the electrical characteristics of the chip in the PROM mode. Figure 16-5 shows a write/verify timing chart.

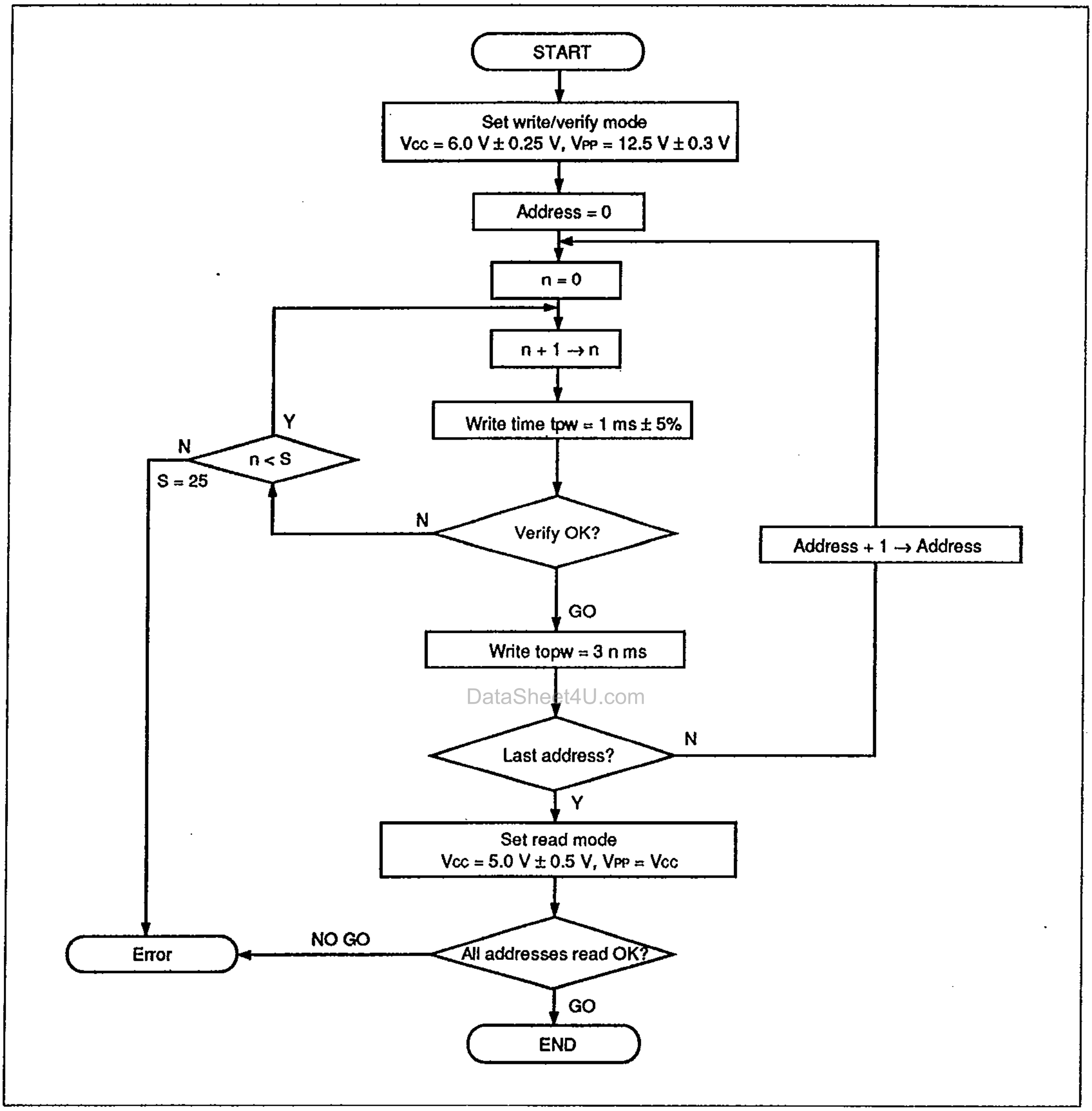


Figure 16-4 High-Speed Programming Flowchart

## Table 16-5 DC Characteristics
(When $V_{CC}$ = 6.0 V ± 0.25 V, $V_{PP}$ = 12.5 V ± 0.3 V, $V_{SS}$ = 0 V, Ta = 25°C ± 5°C)

T-49-19-16

| Item | | Symbol | min | typ | max | Unit | Measurement Conditions |
|---|---|---|---|---|---|---|---|
| Input high voltage | $O_7$ to $O_0$, $A_{14}$ to $A_0$, $\overline{OE}$, $\overline{CE}$ | $V_{IH}$ | 2.2 | — | $V_{CC}$ + 0.3 | V | |
| Input low voltage | $O_7$ to $O_0$, $A_{14}$ to $A_0$, $\overline{OE}$, $\overline{CE}$ | $V_{IL}$ | –0.3 | — | 0.8 | V | |
| Output high voltage | $O_7$ to $O_0$ | $V_{OH}$ | 2.4 | — | — | V | $I_{OH}$ = –200 µA |
| Output low voltage | $O_7$ to $O_0$ | $V_{OL}$ | — | — | 0.45 | V | $I_{OL}$ = 1.6 mA |
| Input leakage current | $O_7$ to $O_0$, $A_{14}$ to $A_0$, $\overline{OE}$, $\overline{CE}$ | $\lvert I_{LI}\rvert$ | — | — | 2 | µA | $V_{in}$ = 5.25 V/0.5 V |
| $V_{CC}$ current | | $I_{CC}$ | — | — | 50 | mA | |
| $V_{PP}$ current | | $I_{PP}$ | — | — | 40 | mA | |

## Table 16-6 AC Characteristics
(When $V_{CC}$ = 6.0 V ± 0.25 V, $V_{PP}$ = 12.5 V ± 0.3 V, $V_{SS}$ = 0 V, Ta = 25°C ± 5°C)

| Item | Symbol | min | typ | max | Unit | Measurement Conditions |
|---|---|---|---|---|---|---|
| Address setup time | $t_{AS}$ | 2 | — | — | µs | See figure 16-5* |
| $\overline{OE}$ 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 | |
| $V_{PP}$ 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 | |
| $V_{CC}$ setup time | $t_{VCS}$ | 2 | — | — | µs | |
| Data output delay time | $t_{OE}$ | 0 | — | 500 | ns | |

Note: * Input pulse level: 0.8 V to 2.2 V
Input rise/fall time ≤ 20 ns
Timing reference levels: input—1.0 V, 2.0 V; output—0.8 V, 2.0 V

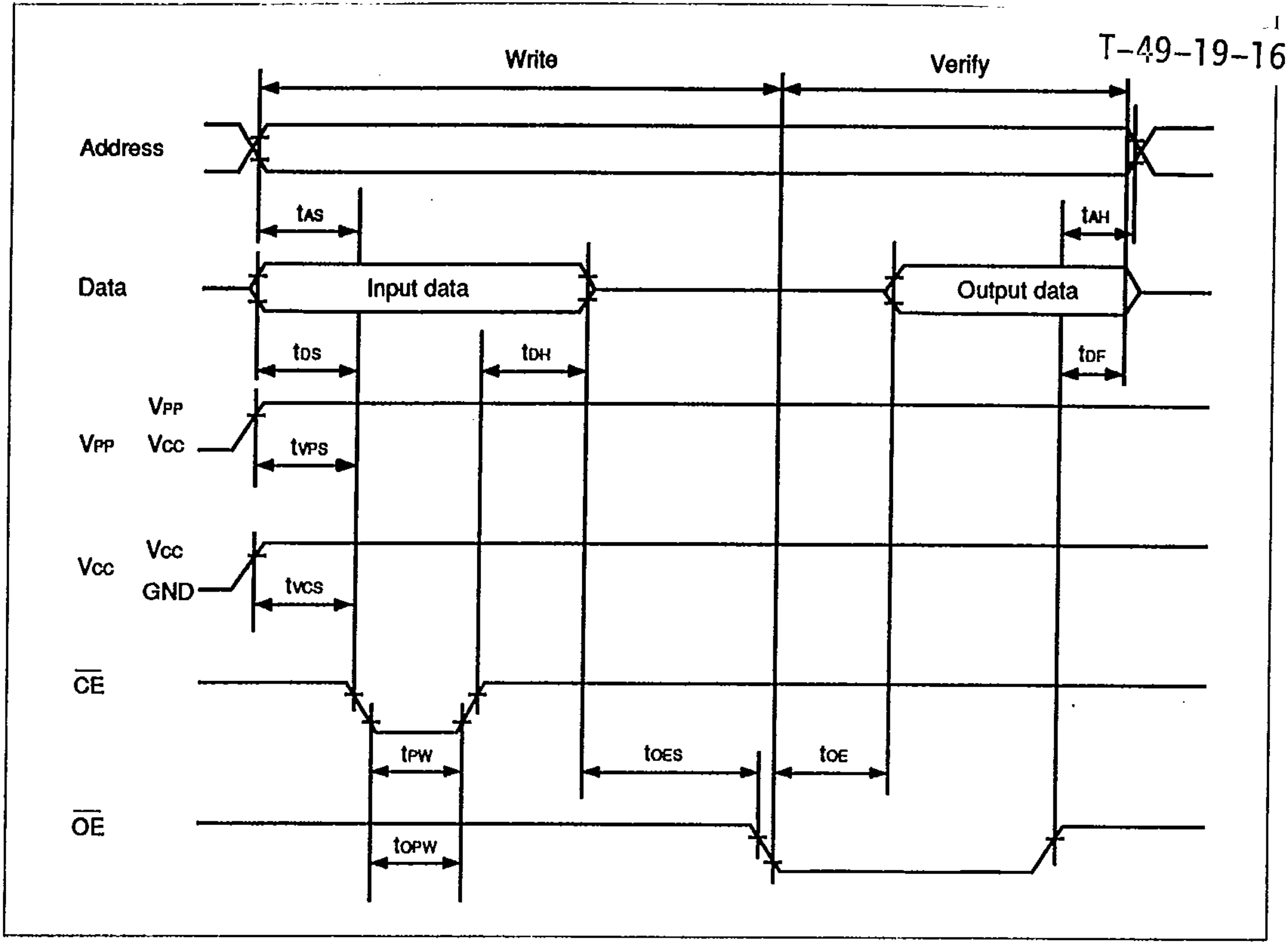


Figure 16-5 PROM Write/Verify Timing

### 16.3.2 Notes on Writing

**1. Write with the specified voltages and timing. The programming voltage (VPP) in the PROM mode 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 HN27256 or HN27C256 specifications, VPP 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.

T-49-19-16

### 16.3.3 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 16-6 shows the recommended screening procedure.

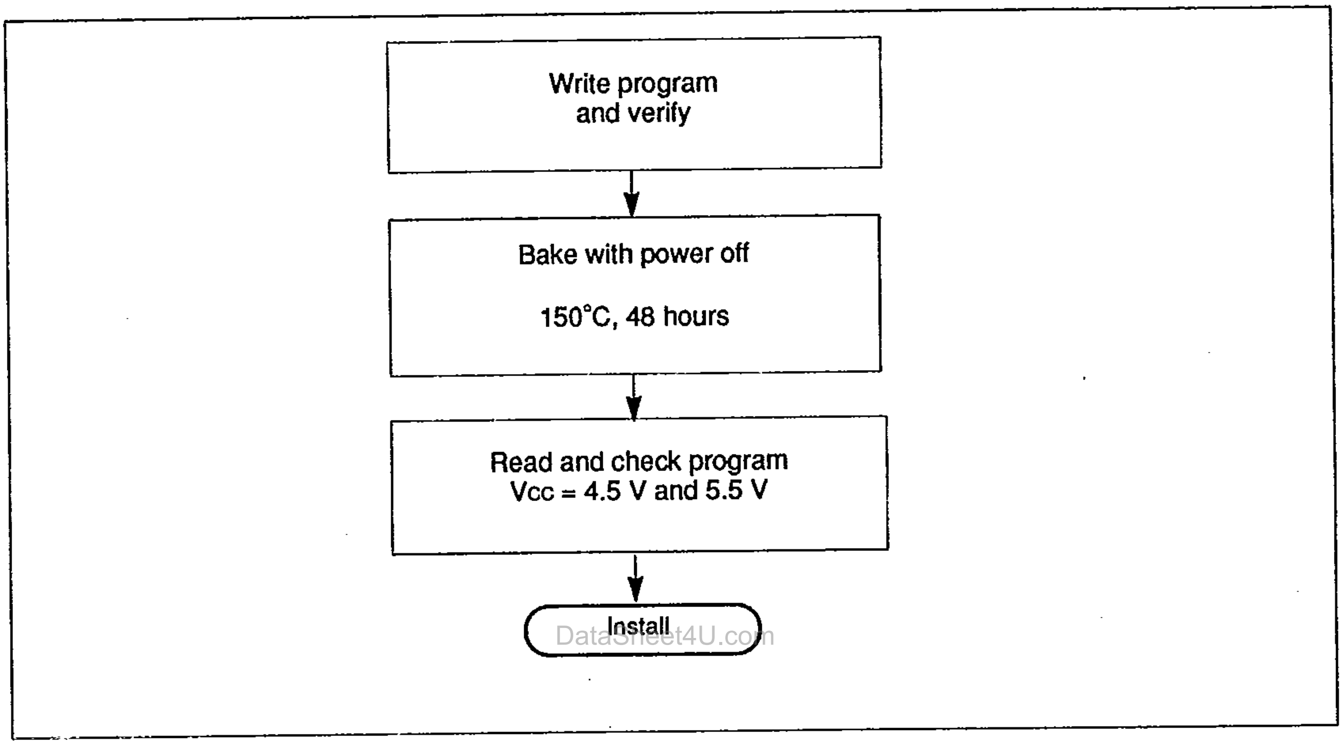


**Figure 16-6 Recommended Screening Procedure**

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 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.

### 16.3.4 Erasing of Data

The windowed package enables data to be erased by illuminating the window with ultraviolet light. Table 16-7 lists the erasing conditions.

Table 16-7 Erasing Conditions

| Item | Value |
|---|---|
| Ultraviolet wavelength | 2537 Å |
| Minimum illumination | 15 W·s/cm$^2$ |

The conditions in table 16-7 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.

## 16.4 Handling of Windowed Packages

**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.

**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, cover the erasing window with a light-proof label (such as an ultraviolet-shield label).

# Section 17 Power-Down State

T-49-19-16

## 17.1 Overview

The H8/520 has a power-down state that greatly reduces power consumption by stopping the CPU functions. The power-down state includes three modes:

1. Sleep mode: software-triggered mode in which the CPU halts but the rest of the chip remains active.
2. Software standby mode: software-triggered mode in which the entire chip is inactive.
3. Hardware standby mode: hardware-triggered mode in which the entire chip is inactive.

The sleep mode and software standby mode are entered from the program execution state by executing the SLEEP instruction under the conditions given in table 17-1. The hardware standby mode is entered from any other state by setting mode 6 at the mode pins ($MD_2$ to $MD_0$).

Table 17-1 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 17-1 Power-Down State**

| Mode | Entering Procedure | Clock | CPU | CPU Registers | Peripheral Functions | RAM | I/O Ports | Exiting Methods |
|---|---|---|---|---|---|---|---|---|
| Sleep mode | Execute SLEEP instruction | Run | Halt | Held | Run | Held | Held | • Interrupt<br>• $\overline{\text{RES}}$ low<br>• Mode 6 |
| Software standby mode | Set SSBY bit in SBYCR to 1, then execute SLEEP instruction* | Halt | Halt | Held | Halt and initialized | Held | Held | • NMI<br>• $\overline{\text{RES}}$ low<br>• Mode 6 |
| Hardware standby mode | Set mode pins to mode 6 | Halt | Halt | Not held | Halt and initialized | Held | High impedance state | • Mode 1,2,3, 4, or 7 then $\overline{\text{RES}}$ low → high |

Notes: * The watchdog timer must also be stopped.
SBYCR: Software standby control register
SSBY: Software standby bit

## 17.2 Sleep Mode

### 17.2.1 Transition to Sleep Mode

T-49-19-16

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 functions of the on-chip supporting modules do not stop in the sleep mode.

### 17.2.2 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 $\overline{\text{RES}}$ pin, or when mode 6 is set at the mode pins.

**Wake-Up by Interrupt:** An interrupt releases the sleep mode and starts either the CPU's interrupt-handling sequence or the data transfer controller (DTC).

If the interrupt is served by the DTC, after the data transfer is completed the CPU executes the instruction following the SLEEP instruction, unless the count in the data transfer count register (DTCR) is 0.

If an interrupt on a level equal to or less than the mask level in the CPU's status register (SR) is requested, the interrupt is left pending and the sleep mode continues. Also, 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.

**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.

**Wake-Up by Mode 6:** When the mode pins are set to mode 6, the chip exits from the sleep mode to the hardware standby mode.

## 17.3 Software Standby Mode

### 17.3.1 Transition to Software Standby Mode

A program enters the software standby mode by setting the standby bit (SSBY) in the software standby control register (SBYCR) to 1, then executing the SLEEP instruction. Table 17-2 lists the attributes of the software standby control register.

**Table 17-2 Software Standby Control Register**

| Name | Abbreviation | R/W | Initial Value | Address |
|---|---|---|---|---|
| Software standby control register | SBYCR | R/W | H'7F | H'FFFB |

In the software standby mode, the CPU, clock, and the on-chip supporting module functions all stop, reducing power consumption to an extremely low level. The on-chip supporting modules and their registers are reset to their initial state, but as long as a minimum necessary voltage supply is maintained (at least 2 V), the contents of the CPU registers and on-chip RAM remain unchanged. The I/O ports also remain in their current states.

### 17.3.2 Software Standby Control Register (SBYCR)

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

The software standby control register (SBYCR) is an 8-bit register that controls the action of the SLEEP instruction.

**Bit 7—Software Standby (SSBY):** This bit enables or disables the transition to the software standby mode.

| Bit 7<br>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. | |

The watchdog timer must be stopped before the chip can enter the software standby mode. To stop the watchdog timer, clear the timer enable bit (TME) in the watchdog timer's timer control/status register (TCSR) to 0. The SSBY bit cannot be set to 1 while the TME bit is set to 1.

When the chip is recovered from the software standby mode by a nonmaskable interrupt (NMI), the SSBY bit is automatically cleared to 0. It is also cleared to 0 by a reset or transition to the hardware standby mode.

**Bits 6 to 0—Reserved:** These bits cannot be modified and are always read as 1.

### 17.3.3 Exit from Software Standby Mode

The chip can be brought out of the software standby mode by an input at the NMI pin, $\overline{\text{RES}}$ pin, or mode pins.

**Recovery by NMI Pin:** When an NMI request signal is received, the clock oscillator begins operating but clock pulses are supplied only to the watchdog timer (WDT). The watchdog timer begins counting from H'00 at the rate determined by the clock select bits (CKS2 to CKS0) in its timer status/control register (TCSR). This rate should be set slow enough to allow the clock oscillator to stabilize before the count reaches H'FF. When the count overflows from H'FF to H'00, clock pulses are supplied to the whole chip, the software standby mode ends, and execution of the NMI interrupt-handling sequence begins.

The clock select bits (CKS2 to CKS0) should be set as follows.

**Crystal Oscillator:** Set CKS2 to CKS0 to a value that makes the watchdog timer interval equal to or greater than 10 ms, which is the clock stabilization time.

**External Clock Input:** CKS2 to CKS0 can be set to any value. The minimum value (CKS2 = CKS1 = CKS0 = 0) is recommended.

**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.

When the chip recovers from the software standby mode by a reset, clock pulses are supplied to the entire chip at once. Be sure to hold the $\overline{\text{RES}}$ pin low long enough for the clock to stabilize.

**Recovery by Mode 6:** When the mode pins are set to mode 6, the chip exits from the software standby mode to the hardware standby mode.

### 17.3.4 Sample Application of Software Standby Mode

In this example the chip enters the software standby mode on the falling edge of the NMI input and recovers from the software standby mode on the rising edge of NMI. Figure 17-1 shows a timing chart of the transitions.

The nonmaskable interrupt edge bit (NMIEG) in the NMI control register (NMICR) is originally cleared to 0, selecting the falling edge as the NMI trigger. After accepting an NMI interrupt in this condition, software changes the NMIEG bit to 1, sets the SSBY bit to 1, and executes the SLEEP instruction to enter the software standby mode. The chip recovers from the software standby mode on the next rising edge at the NMI pin.

T-49-19-16

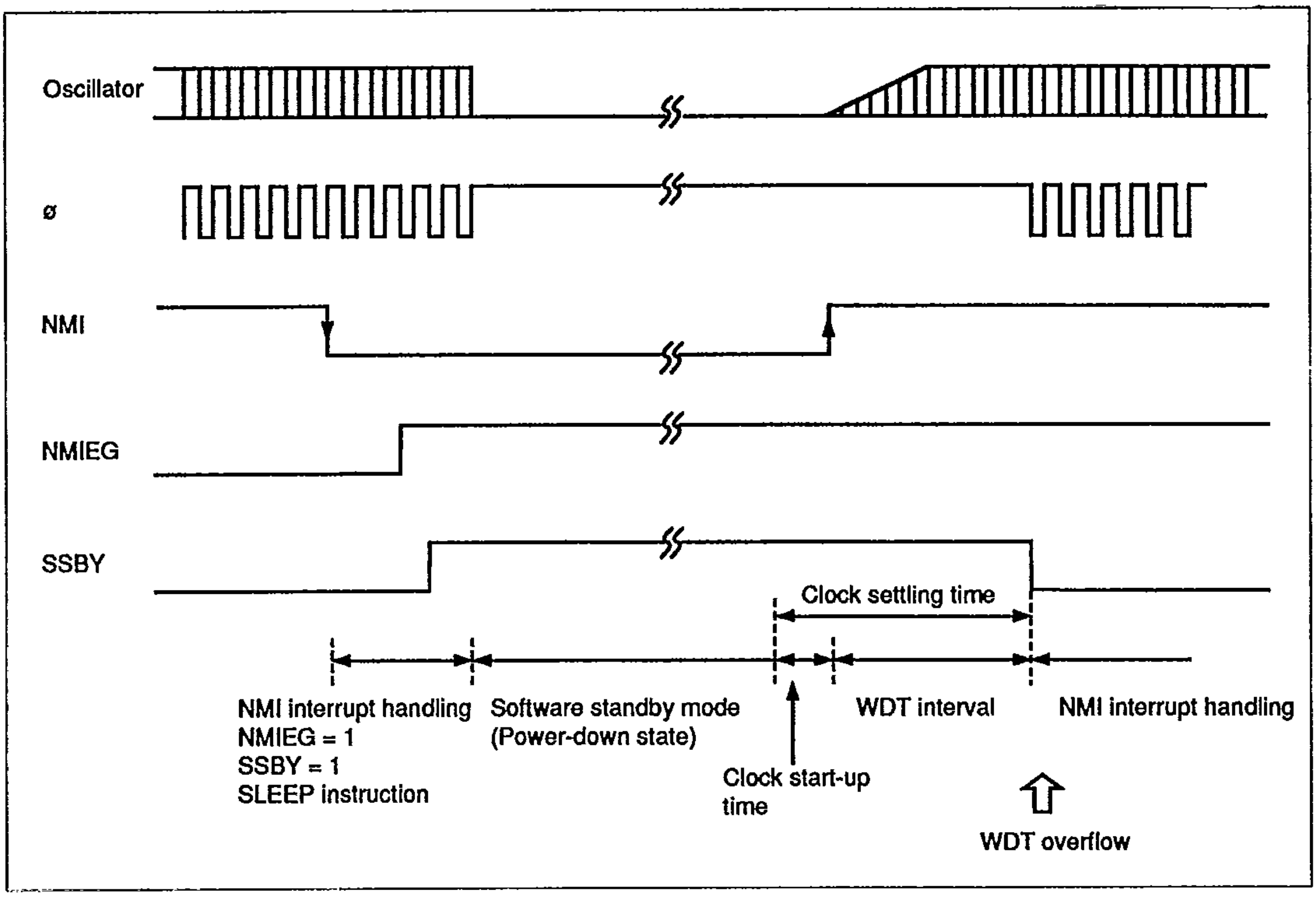


Figure 17-1 NMI Timing of Software Standby Mode (Application Example)

### 17.3.5 Application Notes

The I/O ports remain in their current states in the software standby mode. If a port is in the high output state, the output current is not reduced in the software standby mode.

## 17.4 Hardware Standby Mode

### 17.4.1 Transition to Hardware Standby Mode

Regardless of its current state, the chip enters the hardware standby mode whenever the mode pins are set to mode 6 ($MD_2$ and $MD_1$ high, $MD_0$ 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 2 V).*

Notes: 1. The RAME bit in the RAM control register should be cleared to 0 before the mode pins are set to mode 6, to disable the on-chip RAM during the hardware standby mode.

2. Do not change the inputs at the mode pins (MD2, MD1, MD0) during hardware standby mode. Be particularly careful not to let all three mode inputs go low, since that would place the chip in PROM mode, causing increased current dissipation.

T-49-19-16

### 17.4.2 Recovery from Hardware Standby Mode

Recovery from the hardware standby mode requires inputs at both the mode and RES pins.

When the mode pins are set to mode 1, 2, 3, 4, or 7, 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.

### 17.4.3 Timing Sequence of Hardware Standby Mode

Figure 17-2 shows the usual sequence for entering and leaving the hardware standby mode.

First the $\overline{\text{RES}}$ pin goes low, placing the chip in the reset state. Then the mode pins are set to mode 6, placing the chip in the hardware standby mode and stopping the clock. In the recovery sequence first the mode pins are set to mode 1, 2, 3, 4, or 7; then after the clock stabilizes, the $\overline{\text{RES}}$ pin is returned to the high level.

T-49-19-16

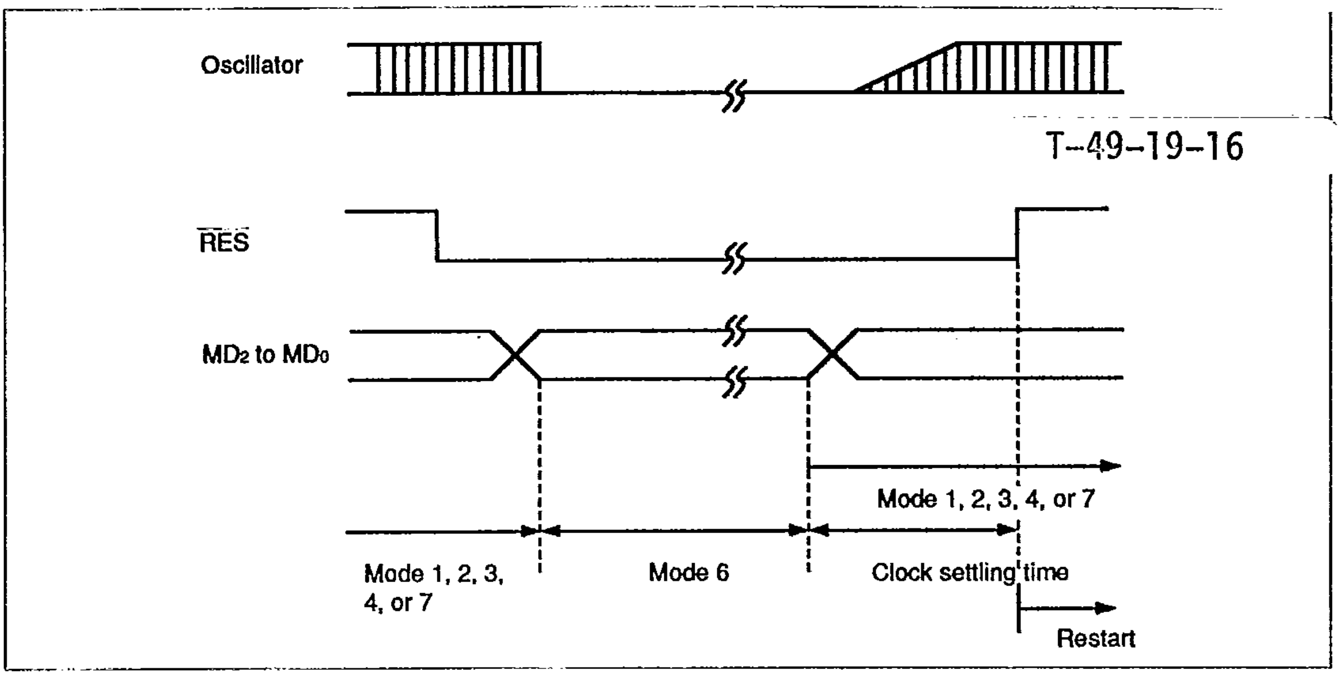


Figure 17-2 Hardware Standby Sequence

# Section 18 Electrical Specifications

T-49-19-16

## 18.1 Absolute Maximum Ratings

Table 18-1 lists the absolute maximum ratings.

**Table 18-1 Absolute Maximum Ratings**

| Item | Symbol | Rating | Unit |
|---|---|---|---|
| Supply voltage | $V_{CC}$ | –0.3 to +0.7 | V |
| Programming voltage | $V_{PP}$ | –0.3 to +13.5 | V |
| Input voltage (except port 6) | $V_{in}$ | –0.3 to $V_{CC}$ + 0.3 | V |
| Input voltage (port 6) | $V_{in}$ | –0.3 to $AV_{CC}$ + 0.3 | V |
| Analog supply voltage | $AV_{CC}$ | –0.3 to +7.0 | V |
| Analog input voltage | $V_{AN}$ | –0.3 to $AV_{CC}$ + 0.3 | V |
| Operating temperature | $T_{opr}$ | Regular specifications: –20 to +75 | °C |
| | | Wide-range specifications: –40 to +85 | °C |
| Storage temperature | $T_{stg}$ | –55 to +125 | °C |

Note: Permanent damage to the chip may result if the absolute maximum ratings shown in table 18-1 are exceeded.

## 18.2 Electrical Characteristics

### 18.2.1 DC Characteristics

Table 18-2 lists the DC characteristics.

**Table 18-2 DC Characteristics**

Conditions: $V_{CC} = AV_{CC} = 5.0$ V ± 10%*1, $V_{SS} = AV_{SS} = 0$ V,

$T_a = -20$ to 75°C (Regular specifications)

$T_a = -40$ to 85°C (Wide-range specifications)

T-49-19-16

| Item | | Symbol | min | typ | max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Input high voltage | $\overline{RES}$, MD$_2$, MD$_1$, MD$_0$ | $V_{IH}$ | $V_{CC} - 0.7$ | — | $V_{CC} + 0.3$ | V | |
| | EXTAL | | $V_{CC} \times 0.7$ | — | $V_{CC} + 0.3$ | V | |
| | Port 6 | | 2.2 | — | $AV_{CC} + 0.3$ | V | |
| | Other input pins (except port 5) | | 2.2 | — | $V_{CC} + 0.3$ | V | |
| Input low voltage | $\overline{RES}$, MD$_2$, MD$_1$, MD$_0$ | $V_{IL}$ | −0.3 | — | 0.5 | V | |
| | Other input pins (except port 5) | | −0.3 | — | 0.8 | | |
| Schmitt trigger input voltage | Port 5 | $V_{T^-}$ | 1.0 | — | 2.5 | V | |
| | | $V_{T^+}$ | 2.0 | — | 3.5 | V | |
| | | $V_{T^+} - V_{T^-}$ | 0.4 | — | — | V | |
| Input leakage current | $\overline{RES}$ | $\lvert I_{in} \rvert$ | — | — | 10.0 | µA | $V_{in} = 0.5$ to $V_{CC} - 0.5$ V |
| | NMI, MD$_2$, MD$_1$, MD$_0$, | | — | — | 1.0 | µA | |
| | Port 6 | | — | — | 1.0 | µA | $V_{in} = 0.5$ to $AV_{CC} - 0.5$ V |
| Leakage current in 3-state (off state) | Port 7, ports 5 to 1 | $\lvert I_{TSI} \rvert$ | — | — | 1.0 | µA | $V_{in} = 0.5$ to $V_{CC} - 0.5$ V |
| Input pull-up MOS current | Ports 3 and 4 | $-I_P$ | 50 | — | 200 | µA | $V_{in} = 0$ V |
| Output high voltage | All output pins | $V_{OH}$ | $V_{CC} - 0.5$ | — | — | V | $I_{OH} = -200$ µA |
| | | | 3.5 | — | — | V | $I_{OH} = -1$ mA |
| Output low voltage | All output pins (except $\overline{RES}$) | $V_{OL}$ | — | — | 0.4 | V | $I_{OL} = 1.6$ mA |
| | Port 3 | | — | — | 1.0 | V | $I_{OL} = 8$ mA |
| | | | — | — | 1.2 | V | $I_{OL} = 10$ mA |
| | $\overline{RES}$ | | — | — | 0.4 | V | $I_{OL} = 2.6$ mA |
| Input capacitance | $\overline{RES}$ | $C_{in}$ | — | — | 60 | pF | $V_{in} = 0$ V |
| | NMI | | — | — | 30 | pF | f = 1 MHz |
| | All input pins except $\overline{RES}$ | | — | — | 15 | pF | $T_a = 25$°C |

## Table 18-2 DC Characteristics (cont)

| Item | | Symbol | min | typ | max | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|
| Current dissipation* | Normal operation | $I_{CC}$ | — | 20 | 30 | mA | f = 6 MHz |
| | | | — | 25 | 40 | mA | f = 8 MHz |
| | | | — | 30 | 50 | mA | f = 10 MHz |
| | Sleep mode | | — | 12 | 20 | mA | f = 6 MHz |
| | | | — | 16 | 25 | mA | f = 8 MHz |
| | | | — | 20 | 30 | mA | f = 10 MHz |
| | Standby | | — | 0.01 | 5.0 | µA | |
| Analog supply current | During A/D conversion | $AI_{CC}$ | — | 0.6 | 2.0 | mA | |
| | While waiting | | — | 0.01 | 5.0 | µA | |
| RAM standby voltage | | $V_{RAM}$ | 2.0 | — | — | V | |

Note: AVcc must be connected to a power supply even when the A/D converter is not used.

* Current dissipation values assume that $V_{IH}$ min = $V_{CC}$ – 0.5 V, $V_{IL}$ max = 0.5 V, all output pins are in the no-load state, and all MOS input pull-ups are off.

## Table 18-3 Allowable Output Current Sink Values

Conditions: $V_{CC} = AV_{CC} = 5.0$ V ± 10%, $V_{SS} = AV_{SS} = 0$ V,
$T_a$ = –20 to 75°C (Regular specifications)
$T_a$ = –40 to 85°C (Wide-range specifications)

| Item | | Symbol | min | typ | max | Unit |
|---|---|---|---|---|---|---|
| Allowable output low current sink (per pin) | Port 3 | $I_{OL}$ | — | — | 10 | mA |
| | $\overline{\text{RES}}$ | | — | — | 2.6 | mA |
| | Other output pins | | — | — | 2.0 | mA |
| Allowable output low current sink (total) | Port 3, total of 8 pins | $\Sigma I_{OL}$ | — | — | 40 | mA |
| | Total of all other output pins | | — | — | 80 | 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 | $\Sigma -I_{OH}$ | — | — | 40 | mA |

Note: To avoid degrading the reliability of the chip, be careful not to exceed the output current sink values in table 18-3. In particular, when driving a Darlington transistor pair or LED directly, be sure to insert a current-limiting resistor in the output path. See figures 18-1 and 18-2.

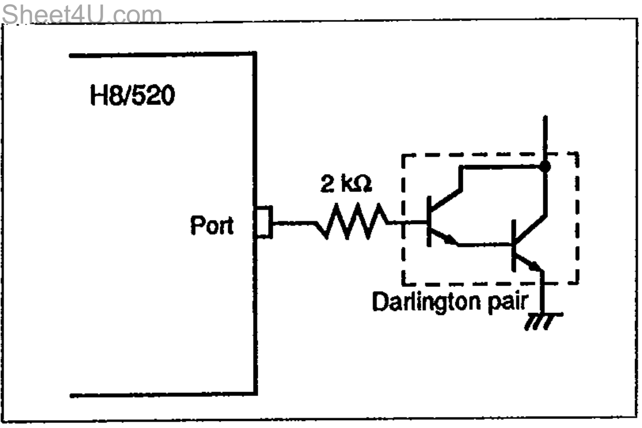


Figure 18-1 Example of Circuit for Driving a Darlington Transistor Pair

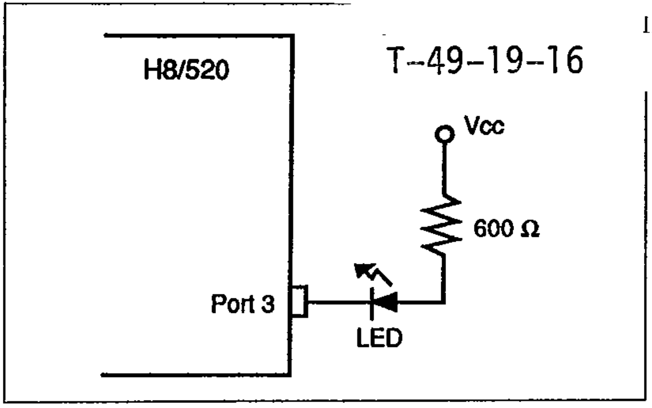


Figure 18-2 Example of Circuit for Driving an LED

### 18.2.2 AC Characteristics

The AC characteristics of the H8/520 chip are listed in three tables. Bus timing parameters are given in table 18-4, control signal timing parameters in table 18-5, and timing parameters of the on-chip supporting modules in table 18-6. See figure 18-3 for the output load circuit.

**Table 18-4 Bus Timing**

Conditions: $V_{CC} = 5.0\ V \pm 10\%$, $\phi = 0.5$ to 10 MHz, $V_{SS} = 0\ V$
$T_a = -20$ to 75°C (Regular specifications)
$T_a = -40$ to 85°C (Wide-range specifications)

| | | 6 MHz | | 8 MHz | | 10 MHz | | | Test |
|---|---|---|---|---|---|---|---|---|---|
| Item | Symbol | min | max | min | max | min | max | Unit | Conditions |
| Clock cycle time | $t_{cyc}$ | 166.7 | 2000 | 125 | 2000 | 100 | 2000 | ns | See figure 18-4 |
| Clock pulse width low | $t_{CL}$ | 65 | — | 45 | — | 35 | — | ns | |
| Clock pulse width high | $t_{CH}$ | 65 | — | 45 | — | 35 | — | ns | |
| Clock rise time | $t_{Cr}$ | — | 15 | — | 15 | — | 15 | ns | |
| Clock fall time | $t_{Cf}$ | — | 15 | — | 15 | — | 15 | ns | |
| Address delay time | $t_{AD}$ | — | 70 | — | 60 | — | 55 | ns | |
| Address hold time | $t_{AH}$ | 30 | — | 25 | — | 20 | — | ns | |
| $\overline{RD}$ delay time 1 | $t_{RDD1}$ | — | 70 | — | 60 | — | 40 | ns | |
| $\overline{RD}$ delay time 2 | $t_{RDD2}$ | — | 70 | — | 60 | — | 50 | ns | |
| $\overline{WR}$ delay time 1 | $t_{WRD1}$ | — | 70 | — | 60 | — | 50 | ns | |
| $\overline{WR}$ delay time 2 | $t_{WRD2}$ | — | 70 | — | 60 | — | 50 | ns | |
| Write data strobe pulse width | $t_{DSWW}$ | 200 | — | 150 | — | 120 | — | ns | |
| Address setup time 1 | $t_{AS1}$ | 25 | — | 20 | — | 15 | — | ns | |
| Address setup time 2 | $t_{AS2}$ | 105 | — | 80 | — | 65 | — | ns | See figure 18-4 |
| Read data setup time | $t_{RDS}$ | 60 | — | 50 | — | 40 | — | ns | |

## Table 18-4 Bus Timing (cont)

| | | 6 MHz | | 8 MHz | | 10 MHz | | | Test |
|---|---|---|---|---|---|---|---|---|---|
| Item | Symbol | min | max | min | max | min | max | Unit | Conditions |
| Read data hold time | $t_{RDH}$ | 0 | — | 0 | — | 0 | — | ns | See figure 18-4 |
| Read data access time | $t_{ACC}$ | — | 280 | — | 190 | — | 160 | ns | |
| Write data delay time | $t_{WDD}$ | — | 70 | — | 60 | — | 60 | ns | |
| Write data setup time | $t_{WDS}$ | 30 | — | 15 | — | 10 | — | ns | |
| Write data hold time | $t_{WDH}$ | 30 | — | 25 | — | 20 | — | ns | |
| Wait setup time | $t_{WTS}$ | 40 | — | 40 | — | 40 | — | ns | See figure 18-5 |
| Wait hold time | $t_{WTH}$ | 10 | — | 10 | — | 10 | — | ns | |

T-49-19-16

## Table 18-5 Control Signal Timing

Conditions: $V_{CC} = 5.0$ V ± 10%, ø = 0.5 to 10 MHz, $V_{SS} = 0$ V
$T_a = -20$ to 75°C (Regular specifications)
$T_a = -40$ to 85°C (Wide-range specifications)

| | | 6 MHz | | 8 MHz | | 10 MHz | | | Test |
|---|---|---|---|---|---|---|---|---|---|
| Item | Symbol | min | max | min | max | min | max | Unit | Conditions |
| $\overline{\text{RES}}$ setup time | $t_{RESS}$ | 200 | — | 200 | — | 200 | — | ns | See figure 18-6 |
| $\overline{\text{RES}}$ pulse width 1* | $t_{RESW1}$ | 6.0 | — | 6.0 | — | 6.0 | — | $t_{cyc}$ | |
| $\overline{\text{RES}}$ pulse width 2* | $t_{RESW2}$ | 520 | — | 520 | — | 520 | — | $t_{cyc}$ | |
| $\overline{\text{RES}}$ output delay time | $t_{RESD}$ | — | 100 | — | 100 | — | 100 | ns | See figure 18-7 |
| $\overline{\text{RES}}$ output pulse width | $t_{RESOW}$ | 132 | — | 132 | — | 132 | — | $t_{cyc}$ | |
| Mode programming setup time | $t_{MDS}$ | 4.0 | — | 4.0 | — | 4.0 | — | $t_{cyc}$ | See figure 18-6 |
| $\overline{\text{NMI}}$ setup time | $t_{NMIS}$ | 150 | — | 150 | — | 150 | — | ns | See figure 18-8 |
| $\overline{\text{NMI}}$ hold time | $t_{NMIH}$ | 10 | — | 10 | — | 10 | — | ns | |
| $\overline{\text{IRQ}_0}$ setup time | $t_{IRQ0S}$ | 50 | — | 50 | — | 50 | — | ns | |
| $\overline{\text{IRQ}_1}$ to $\overline{\text{IRQ}_7}$ setup time | $t_{IRQ1S}$ | 50 | — | 50 | — | 50 | — | ns | |
| $\overline{\text{IRQ}_1}$ to $\overline{\text{IRQ}_7}$ hold time | $t_{IRQ1H}$ | 10 | — | 10 | — | 10 | — | ns | |
| $\overline{\text{NMI}}$ pulse width (for recovery from software standby mode) | $t_{NMIW}$ | 200 | — | 200 | — | 200 | — | ns | |
| A/D trigger setup time | $t_{TRGS}$ | 50 | — | 50 | — | 50 | — | ns | See figure 18-18 |
| A/D trigger hold time | $t_{TRGH}$ | 10 | — | 10 | — | 10 | — | ns | |
| Crystal oscillator settling time (reset) | $t_{OSC1}$ | 20 | — | 20 | — | 20 | — | ms | See figure 18-9 |
| Crystal oscillator settling time (software standby) | $t_{OSC2}$ | 10 | — | 10 | — | 10 | — | ms | See figure 17-1 |

Note: * $t_{RESW2}$ applies when the RSTOE bit in the reset control/status register (RSTCR) is set to 1. $t_{RESW1}$ applies when RSTOE is cleared to 0. $t_{RESW1}$ also applies at power-up.

## Table 18-6 Timing Conditions of On-Chip Supporting Modules

Conditions: $V_{CC} = 5.0$ V ± 10%, $\phi = 0.5$ to 10 MHz, $V_{SS} = 0$ V
$T_a = -20$ to 75°C (Regular specifications)
$T_a = -40$ to 85°C (Wide-range specifications)

T-49-19-16

| Item | | | Symbol | 6 MHz | | 8 MHz | | 10 MHz | | Unit | Test Conditions |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | min | max | min | max | min | max | | |
| FRT | Timer output delay time | | $t_{FTOD}$ | — | 100 | — | 100 | — | 100 | ns | See figure 18-11 |
| | Timer input setup time | | $t_{FTIS}$ | 50 | — | 50 | — | 50 | — | ns | |
| | Timer clock input setup time | | $t_{FTCS}$ | 50 | — | 50 | — | 50 | — | ns | See figure 18-12 |
| | Timer clock pulse width | | $t_{FTCW}$ | 1.5 | — | 1.5 | — | 1.5 | — | $t_{cyc}$ | |
| TMR | Timer output delay time | | $t_{TMOD}$ | — | 100 | — | 100 | — | 100 | ns | See figure 18-13 |
| | Timer clock input setup time | | $t_{TMCS}$ | 50 | — | 50 | — | 50 | — | ns | See figure 18-14 |
| | Timer clock pulse width | | $t_{TMCW}$ | 1.5 | — | 1.5 | — | 1.5 | — | $t_{cyc}$ | |
| | Timer reset input setup time | | $t_{TMRS}$ | 50 | — | 50 | — | 50 | — | ns | See figure 18-15 |
| SCI | Input clock cycle | (Async) | $t_{Scyc}$ | 2 | — | 2 | — | 2 | — | $t_{cyc}$ | See figure 18-16 |
| | | (Sync) | | 4 | — | 4 | — | 4 | — | $t_{cyc}$ | |
| | Input clock pulse width | | $t_{SCKW}$ | 0.4 | 0.6 | 0.4 | 0.6 | 0.4 | 0.6 | $t_{Scyc}$ | |
| | Transmit data delay time | (Sync) | $t_{TXD}$ | — | 100 | — | 100 | — | 100 | ns | See figure 18-17 |
| | Receive data setup time | (Sync) | $t_{RXS}$ | 100 | — | 100 | — | 100 | — | ns | |
| | Receive data hold time | (Sync) | $t_{RXH}$ | — | 100 | — | 100 | — | 100 | ns | |
| Ports | Output data delay time | | $t_{PWD}$ | — | 100 | — | 100 | — | 100 | ns | See figure 18-10 |
| | Input data setup time | | $t_{PRS}$ | 50 | — | 50 | — | 50 | — | ns | |
| | Input data hold time | | $t_{PRH}$ | 50 | — | 50 | — | 50 | — | ns | |

- **Measurement Conditions for AC Characteristics**

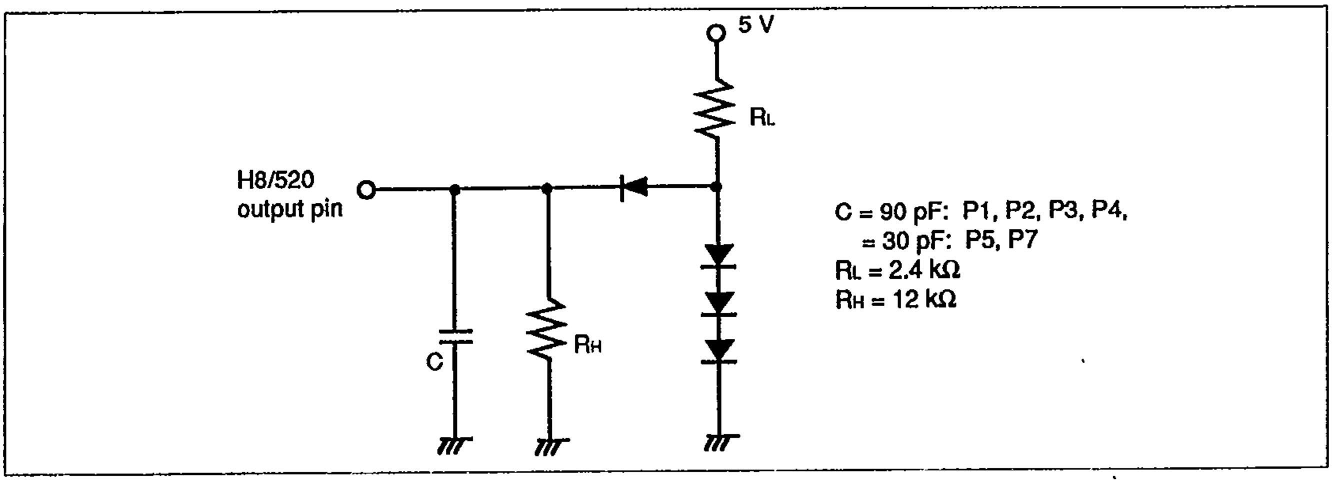


Figure 18-3 Output Load Circuit

### 18.2.3 A/D Converter Characteristics

Table 18-7 lists the characteristics of the on-chip A/D converter.

**Table 18-7 (1) A/D Converter Characteristics**

Conditions: $V_{CC} = AV_{CC} = 5.0\ V \pm 10\%$, $V_{SS} = AV_{SS} = 0\ V$,
$T_a = -40$ to 85°C (Wide-range specifications)

| | 6 MHz | | | 8 MHz | | | 10 MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Item | min | typ | max | min | typ | max | min | typ | max | Unit |
| Resolution | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | Bits |
| Conversion time | — | — | 23.0 | — | — | 17.25 | — | — | 13.8 | µs |
| Analog input capacitance | — | — | 20 | — | — | 20 | — | — | 20 | pF |
| Allowable signal-source impedance | — | — | 10 | — | — | 10 | — | — | 10 | kΩ |
| Nonlinearity error | — | — | ±2.0 | — | — | ±2.0 | — | — | ±2.0 | LSB |
| Offset error | — | — | ±2.0 | — | — | ±2.0 | — | — | ±2.0 | LSB |
| Full-scale error | — | — | ±2.0 | — | — | ±2.0 | — | — | ±2.0 | LSB |
| Quantizing error | — | — | ±0.5 | — | — | ±0.5 | — | — | ±0.5 | LSB |
| Absolute accuracy | — | — | ±2.5 | — | — | ±2.5 | — | — | ±2.5 | LSB |

**Table 18-7 (2) A/D Converter Characteristics**

Conditions: $V_{CC} = AV_{CC} = 5.0\ V \pm 10\%$, $V_{SS} = AV_{SS} = 0\ V$,
$T_a = -20$ to 75°C (Regular specifications)

| | 6 MHz | | | 8 MHz | | | 10 MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Item | min | typ | max | min | typ | max | min | typ | max | Unit |
| Resolution | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | Bits |
| Conversion time | — | — | 23.0 | — | — | 17.25 | — | — | 13.8 | µs |
| Analog input capacitance | — | — | 20 | — | — | 20 | — | — | 20 | pF |
| Allowable signal-source impedance | — | — | 10 | — | — | 10 | — | — | 10 | kΩ |
| Nonlinearity error | — | — | ±3.5 | — | — | ±3.5 | — | — | ±3.5 | LSB |
| Offset error | — | — | ±3.5 | — | — | ±3.5 | — | — | ±3.5 | LSB |
| Full-scale error | — | — | ±3.5 | — | — | ±3.5 | — | — | ±3.5 | LSB |
| Quantizing error | — | — | ±0.5 | — | — | ±0.5 | — | — | ±0.5 | LSB |
| Absolute accuracy | — | — | ±4.0 | — | — | ±4.0 | — | — | ±4.0 | LSB |

## 18.3 MCU Operational Timing

This section provides the following timing charts:

T-49-19-16

| | |
|---|---|
| 18.3.1 Bus timing | Figures 18-4 and 18-5 |
| 18.3.2 Control Signal Timing | Figures 18-6 to 18-8 |
| 18.3.3 Clock Timing | Figure 18-9 |
| 18.3.4 I/O Port Timing | Figure 18-10 |
| 18.3.5 16-Bit Free-Running Timer Timing | Figures 18-11 and 18-12 |
| 18.3.6 8-Bit Timer Timing | Figures 18-13 to 18-15 |
| 18.3.7 SCI Timing | Figures 18-16 and 18-17 |

### 18.3.1 Bus Timing

### 1. Basic Bus Cycle (without Wait States) in Expanded Modes

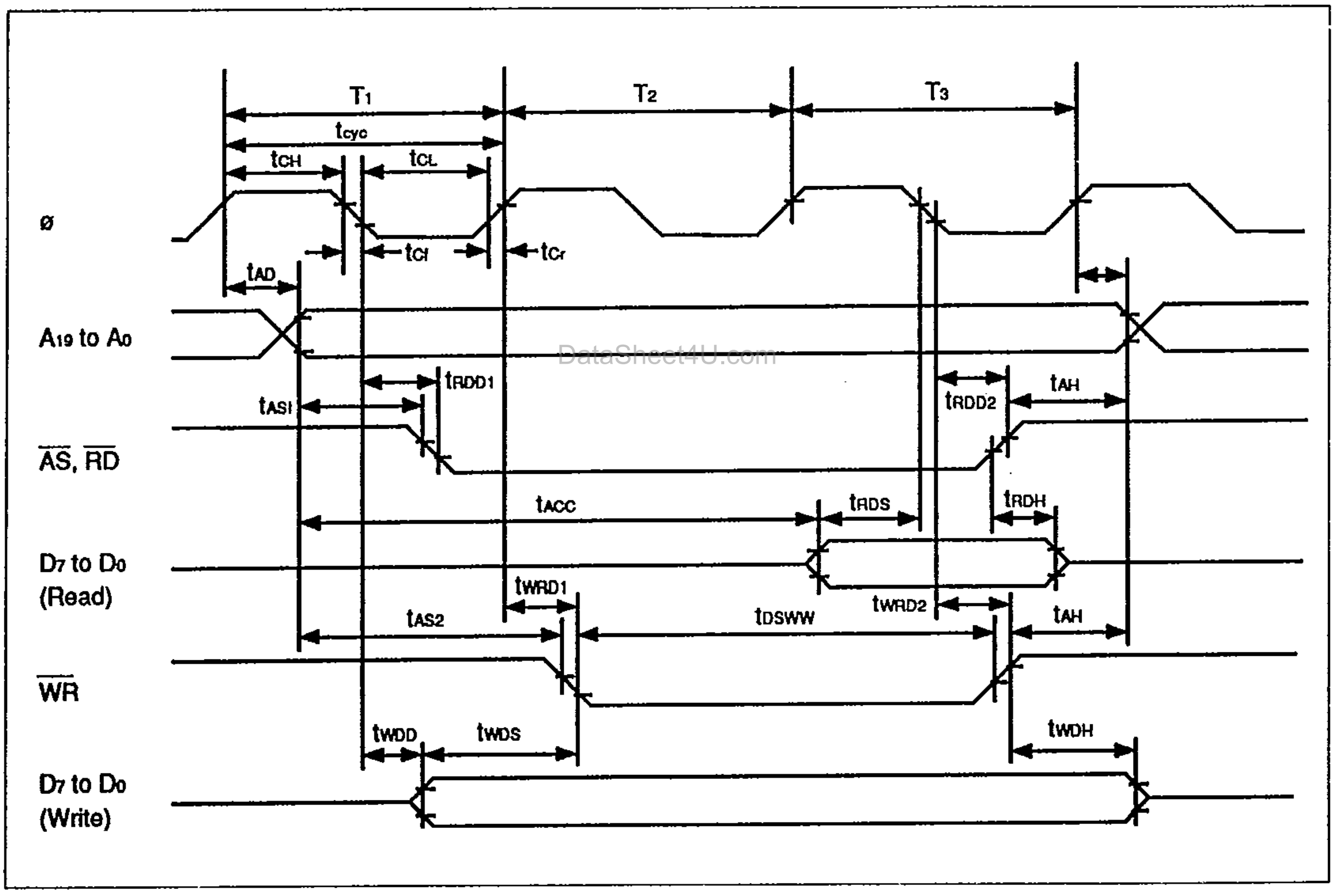


Figure 18-4 Basic Bus Cycle (without Wait States) in Expanded Modes

T-49-19-16

## 2. Basic Bus Cycle (with 1 Wait State) in Expanded Modes

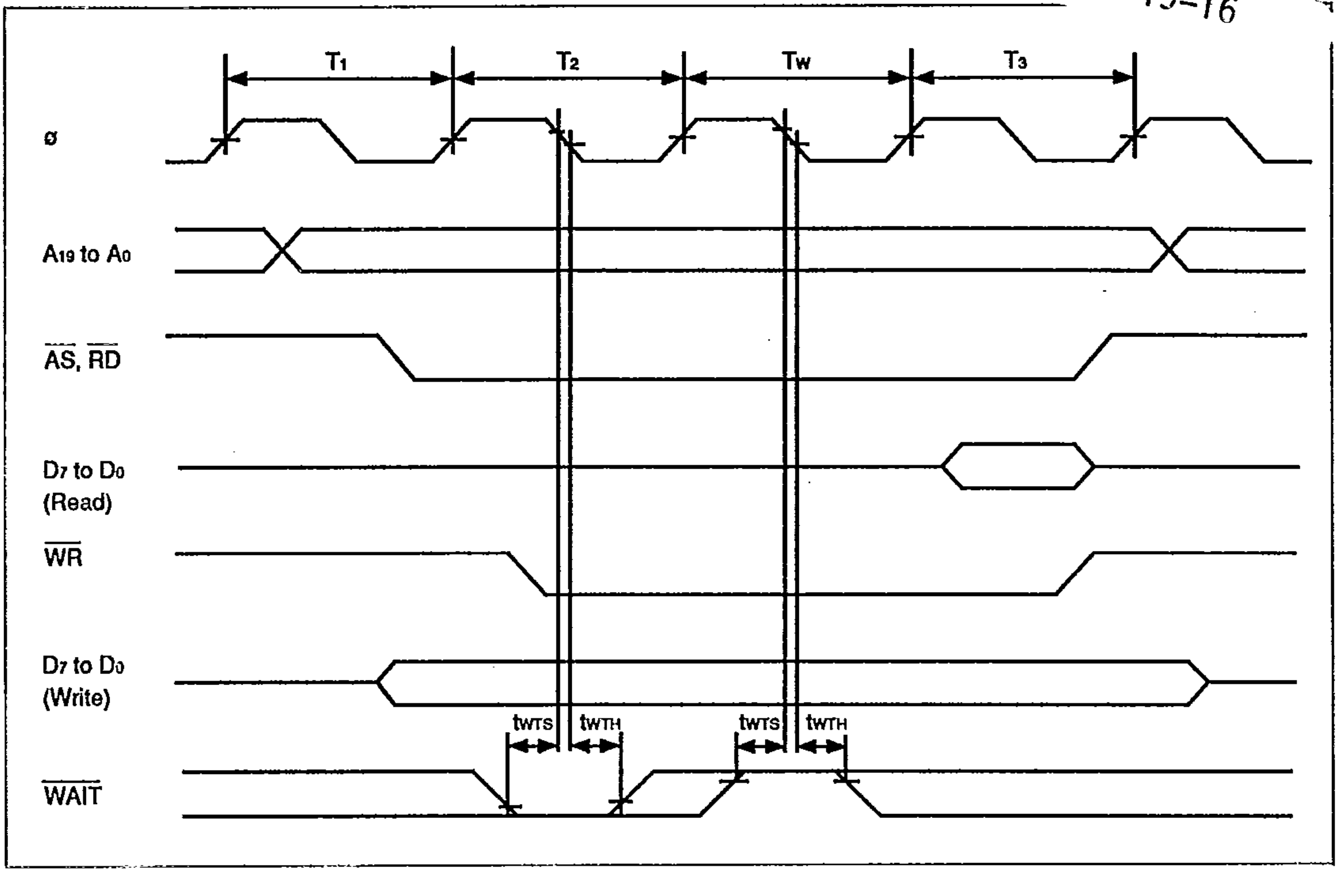


Figure 18-5 Basic Bus Cycle (with 1 Wait State) in Expanded Modes

### 18.3.2 Control Signal Timing

T-49-19-16

#### 1. Reset Input Timing

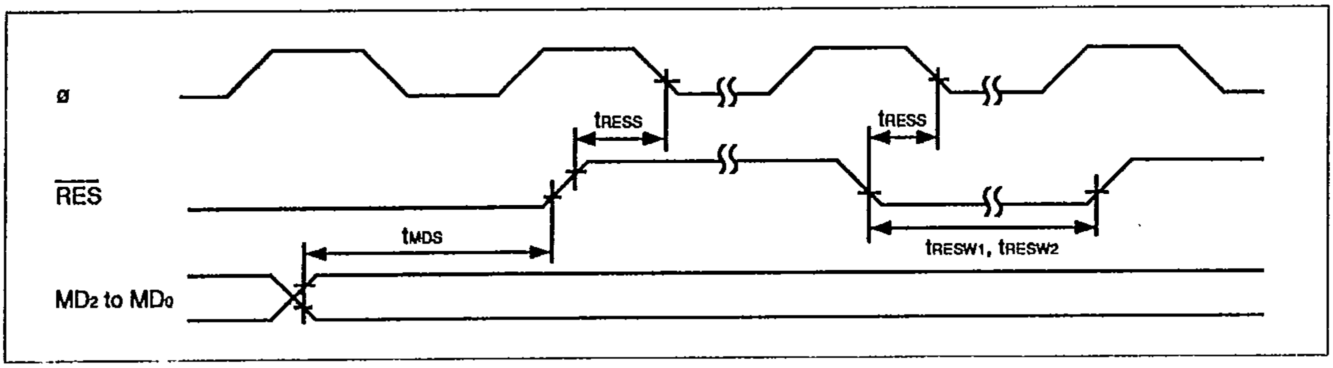


**Figure 18-6 Reset Input Timing**

#### 2. Reset Output Timing

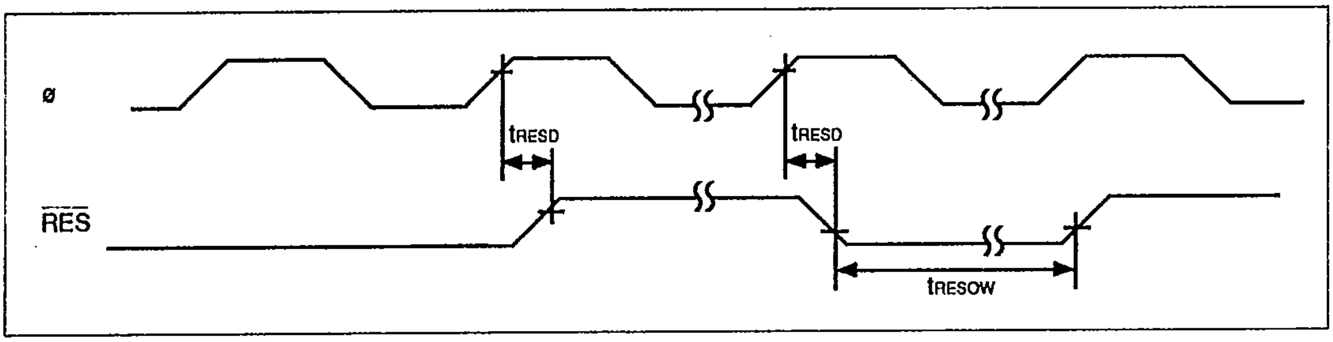


**Figure 18-7 Reset Output Timing**

#### 3. Interrupt Input Timing

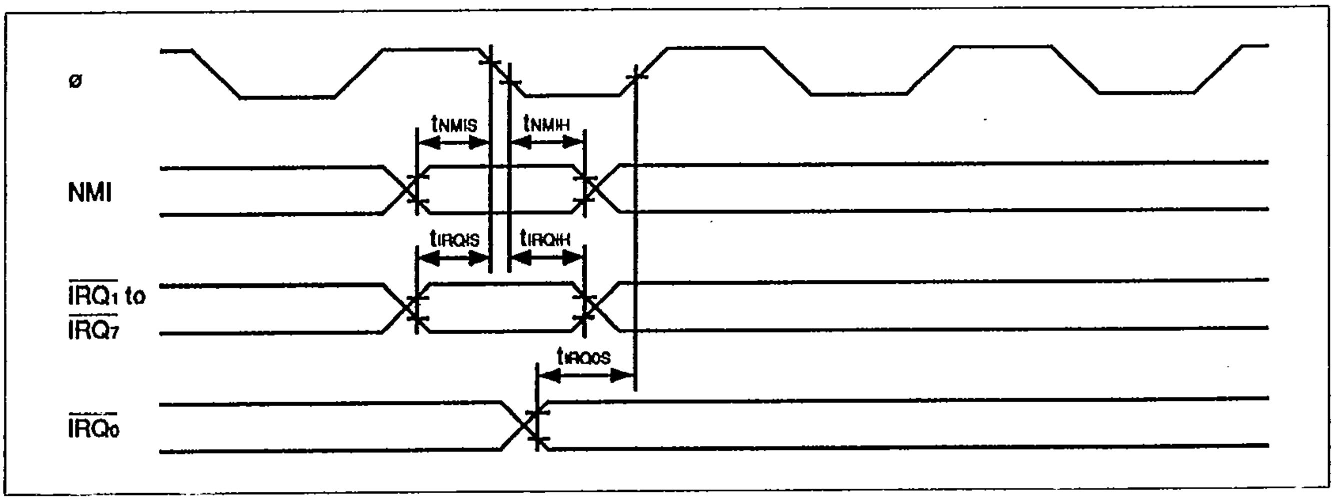
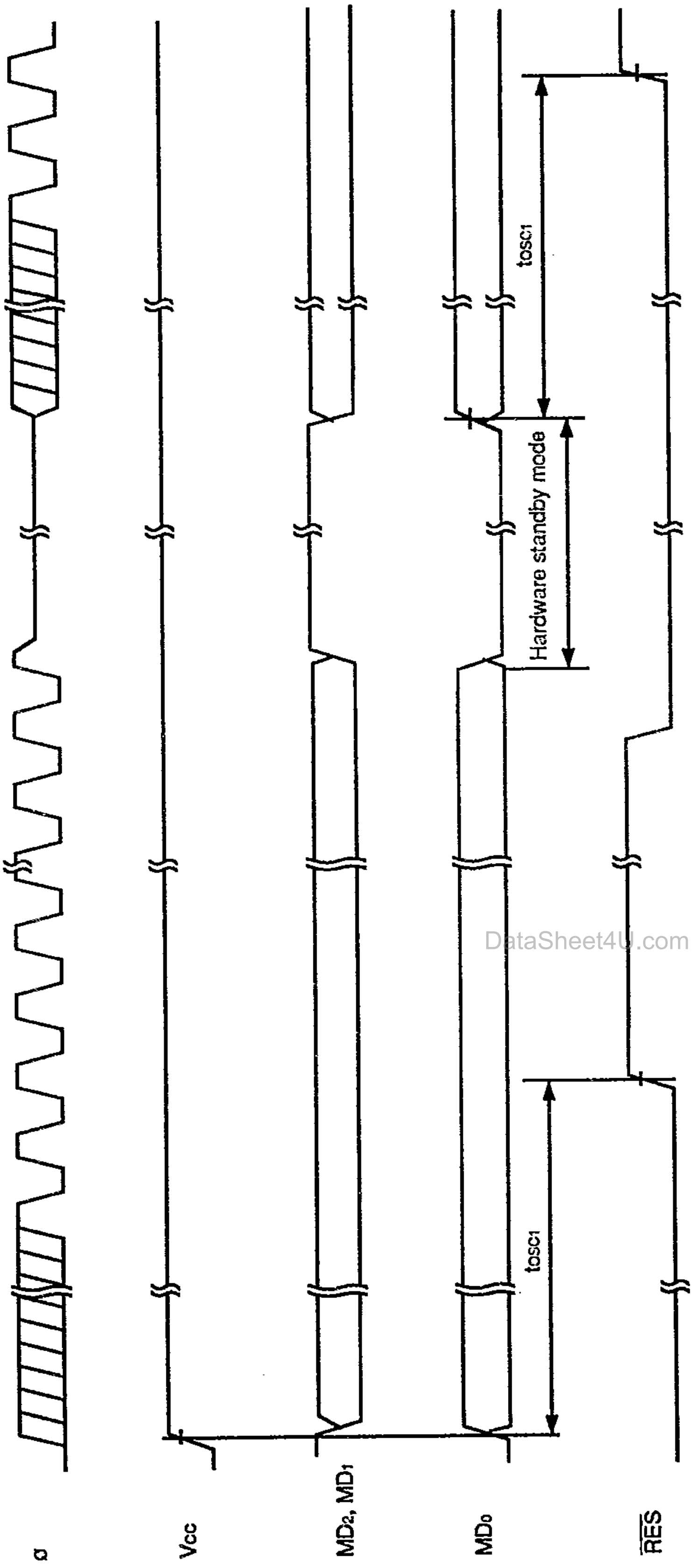


**Figure 18-8 Interrupt Input Timing**

### 18.3.3 Clock Oscillator Stabilization Timing

T-49-19-16

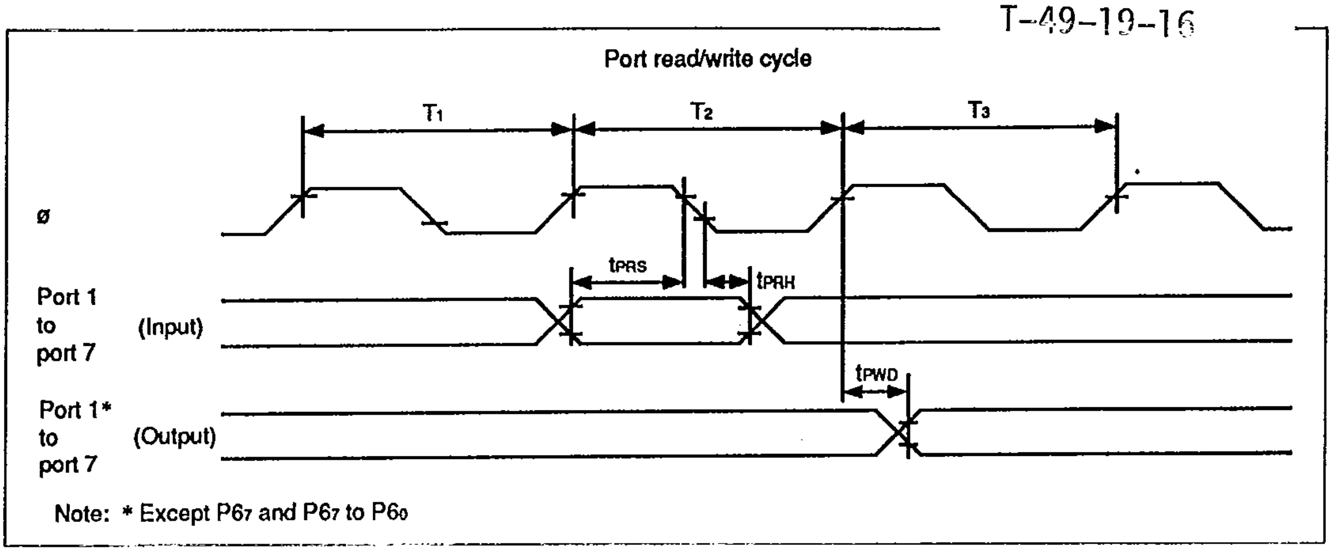


Note: The H8/520 enters hardware standby mode when $MD_2$ and $MD_1$ are driven high and $MD_0$ is driven low.

**Figure 18-9 Clock Oscillator Stabilization**

### 18.3.4 I/O Port Timing

T-49-19-16



Note: * Except $P6_7$ and $P6_7$ to $P6_0$

Figure 18-10 I/O Port Input/Output Timing

### 18.3.5 16-Bit Free-Running Timer Timing

T-49-19-16

#### 1. Free-Running Timer Input/Output Timing

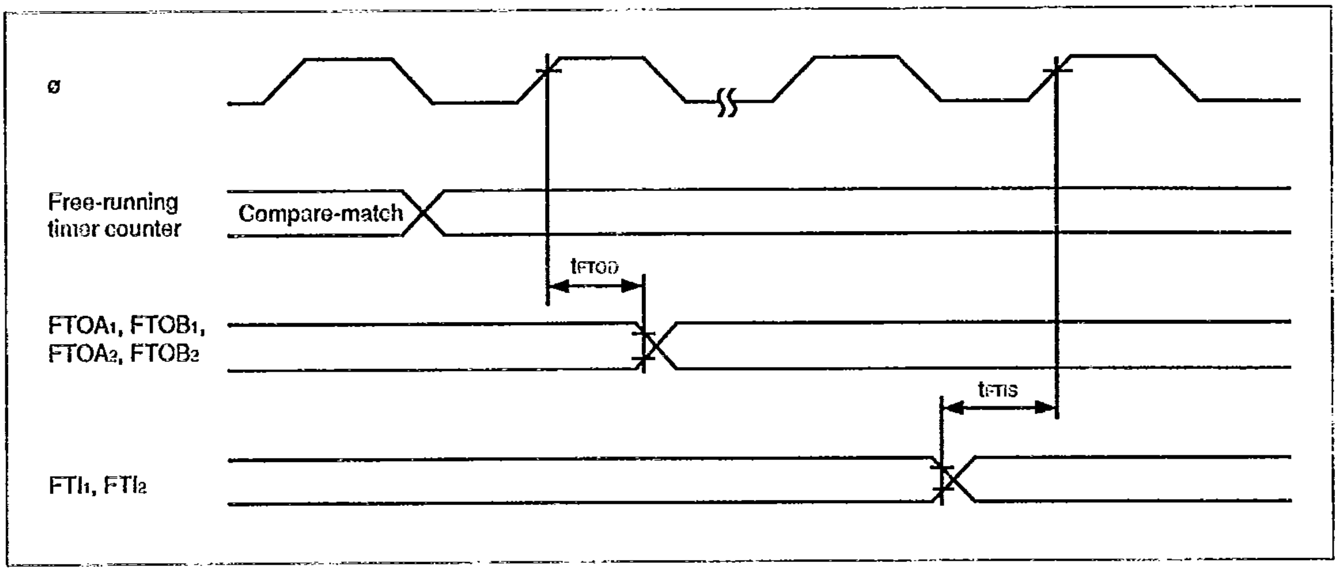


Figure 18-11 Free-Running Timer Input/Output Timing

#### 2. External Clock Input Timing for Free-Running Timers

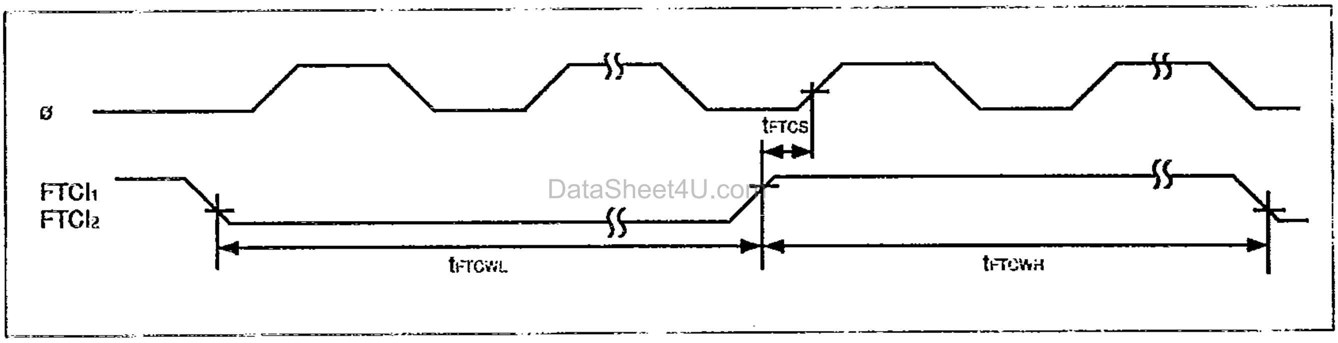


Figure 18-12 External Clock Input Timing for Free-Running Timers

### 18.3.6 8-Bit Timer Timing

T-49-19-15

#### 1. 8-Bit Timer Output Timing

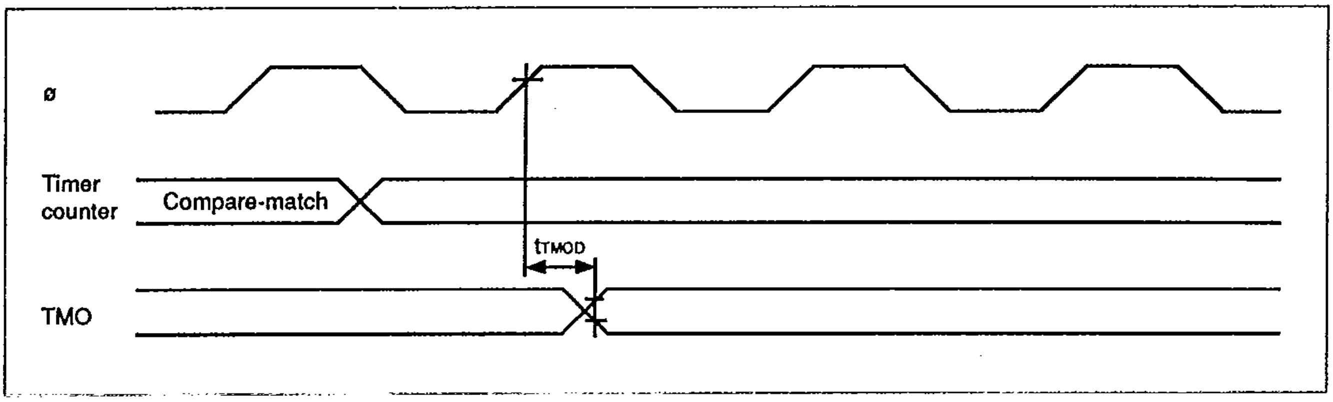


Figure 18-13 8-Bit Timer Output Timing

#### 2. 8-Bit Timer Clock Input Timing

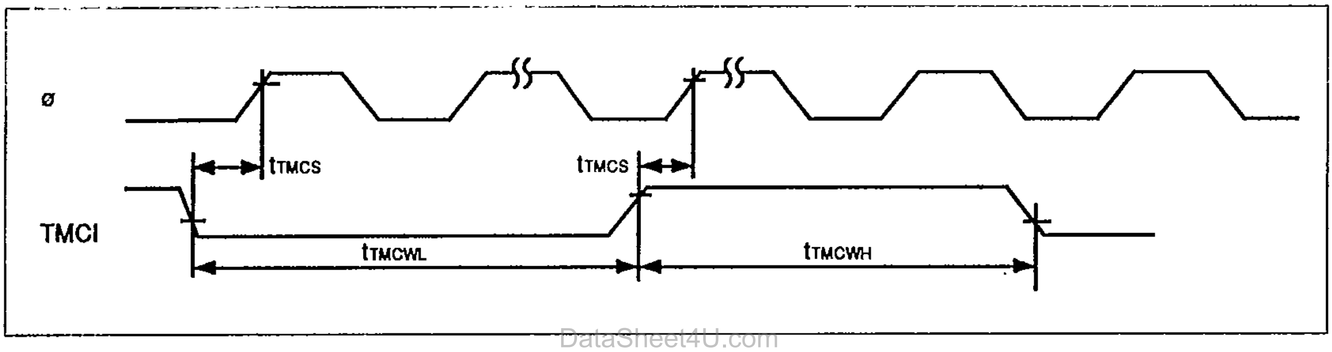


Figure 18-14 8-Bit Timer Clock Input Timing

#### 3. 8-Bit Timer Reset Input Timing

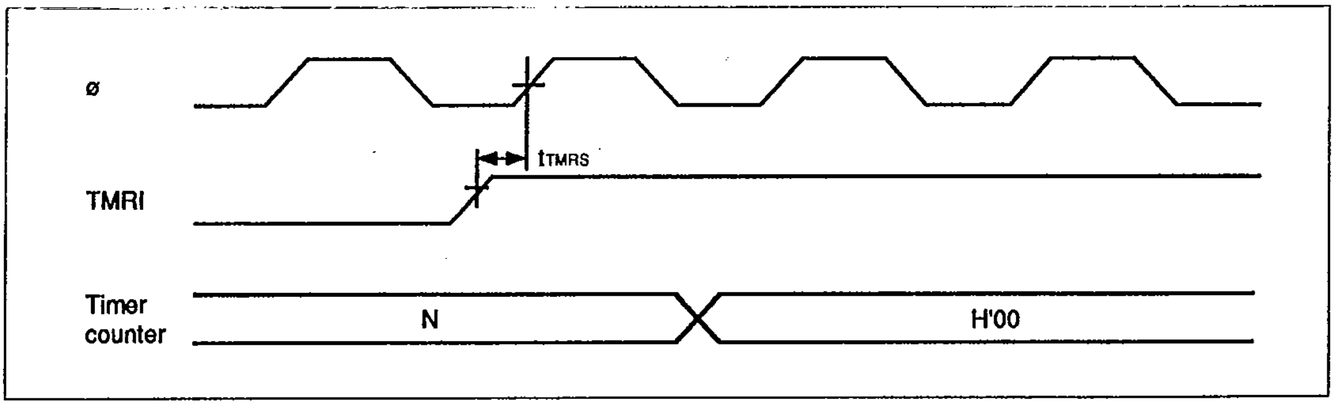


Figure 18-15 8-Bit Timer Reset Input Timing

### 18.3.7 Serial Communication Interface Timing

T-49-19-16

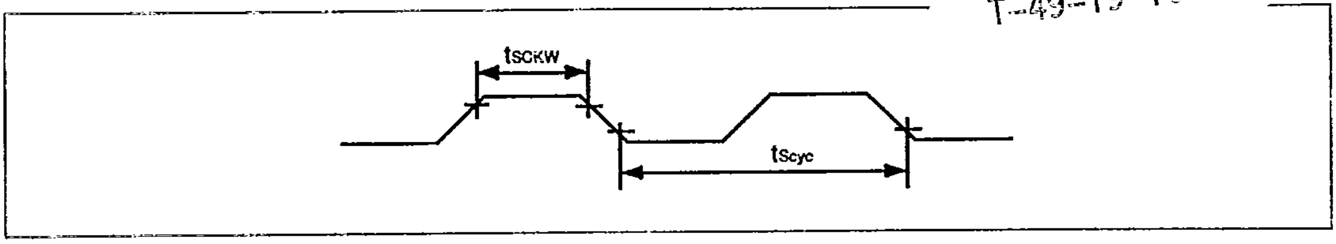


Figure 18-16 SCI Input Clock Timing

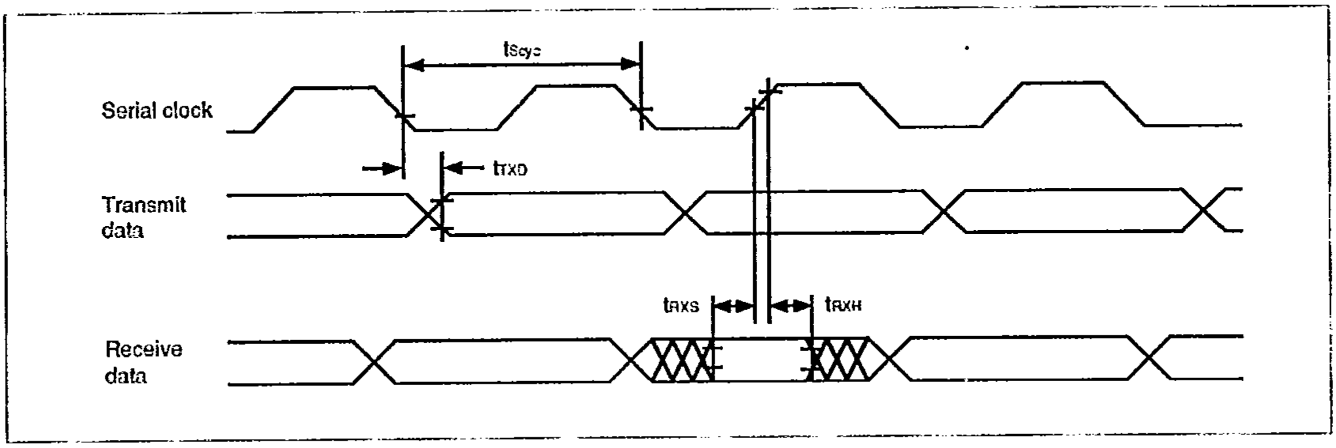


Figure 18-17 SCI Input/Output Timing (Synchronous Mode)

### 18.3.8 A/D External Trigger Input Timing

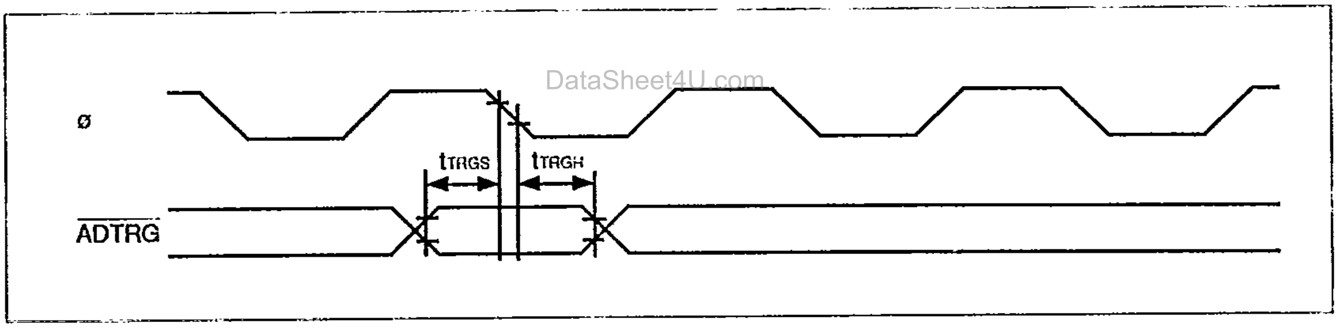


Figure 18-18 A/D External Trigger Input Timing

T-90-20

# Appendix F Package Dimensions

Figure F-1 shows the dimensions of the DC-64S package. Figure F-2 shows the dimensions of the DP-64S package. Figure F-3 shows the dimensions of the FP-64A package. Figure F-4 shows the dimensions of the CP-68 package.

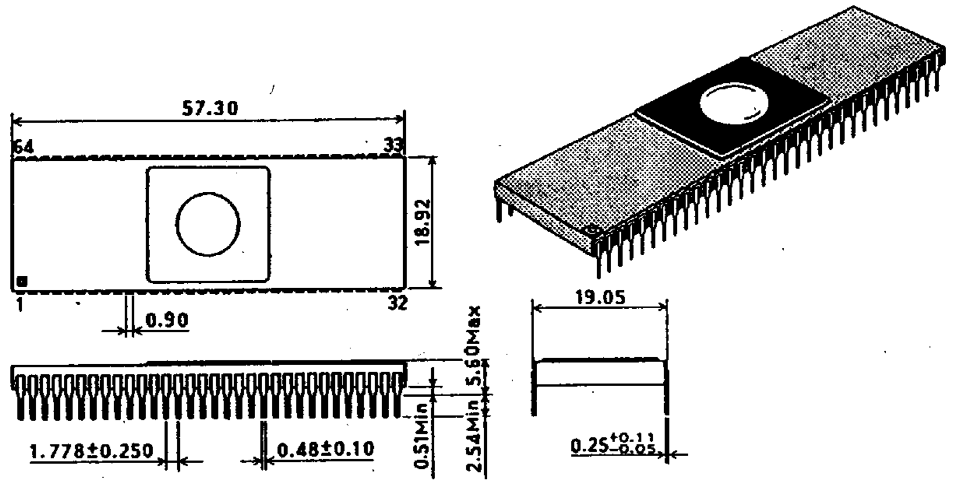


Figure F-1 Package Dimensions (DC-64S)

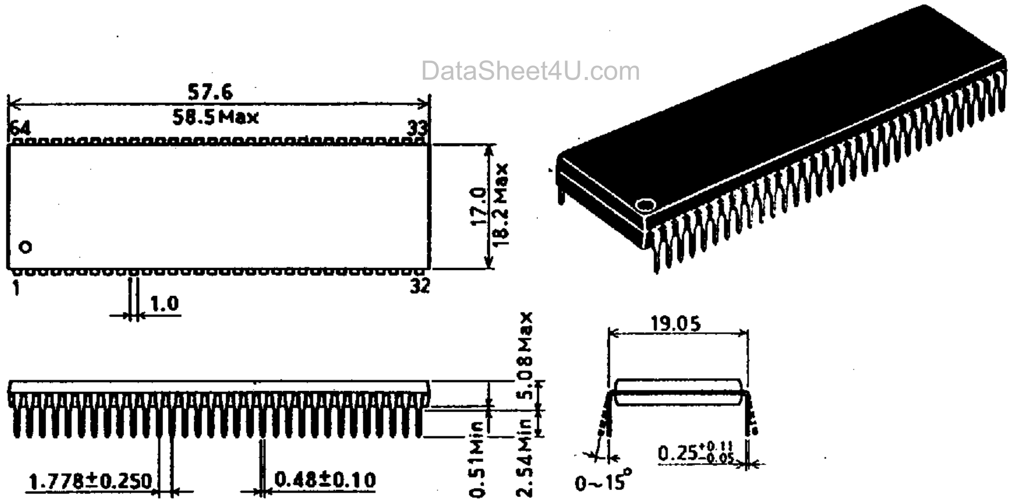


Figure F-2 Package Dimensions (DP-64S)

T-90-20

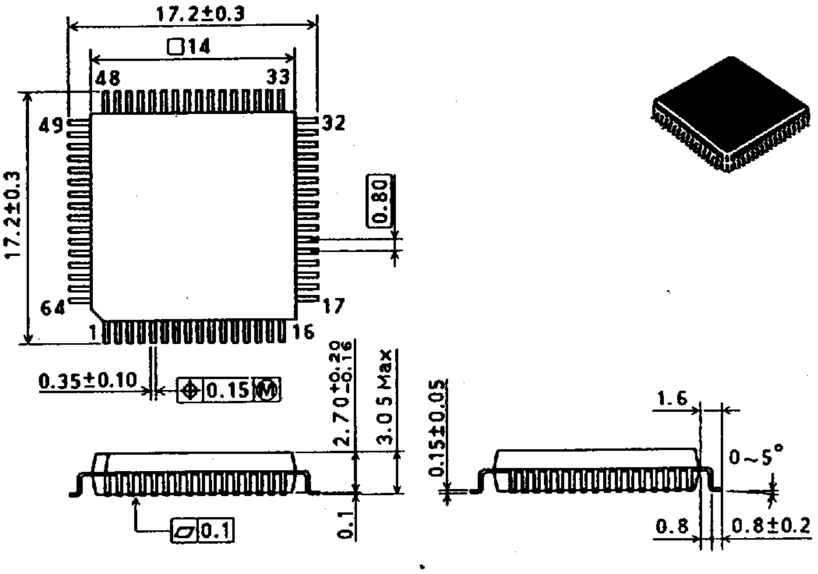


Figure F-3 Package Dimensions (FP-64A)

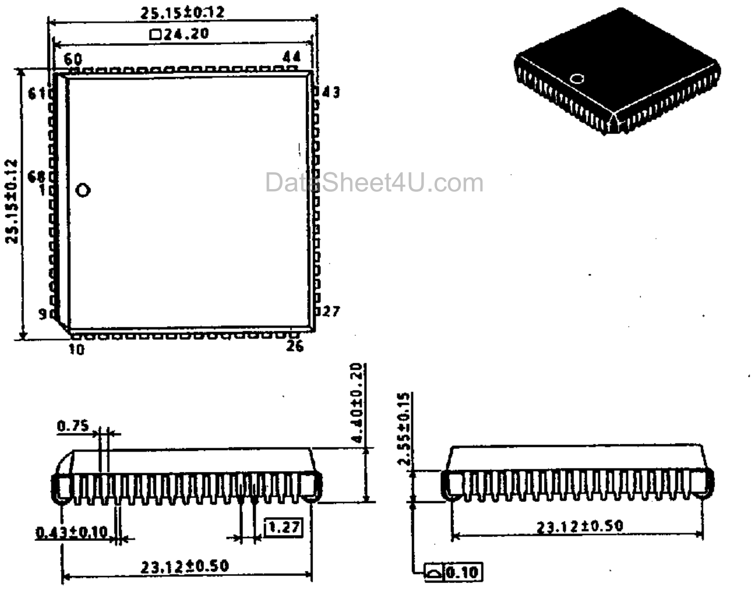


Figure F-4 Package Dimensions (CP-68)