H8 HITACHI SINGLE-CHIP MICROCOMPUTER

# H8/3437 Series
# H8/3434F-ZTAT™

Microcomputer-controlled devices have made remarkable progress, but even more powerful microcontrollers will be needed to take us into the future. Applying its long experience with microcontroller technology, Hitachi has developed the H8 Series, an extensive product lineup designed to fulfil a wide variety of needs.

Hitachi's H8 Series of single-chip microcontrollers includes the H8/500, H8/300H, H8/300, and H8/300L families. These four families have the following features.

**H8/500:** Highly orthogonal instruction set; optimized for high-level languages; general-register architecture; address space expandable to 16 Mbytes

**H8/300H:** 32-bit CPU upward-compatible with the H8/300 Series; 16-Mbyte linear address space; concise instruction set; powerful 8-, 16-, and 32-bit arithmetic instructions

**H8/300:** 16-bit CPU with general-register architecture; 64-kbyte address space; concise instruction set; powerful bit-manipulation instructions

**H8/300L:** CPU compatible with the H8/300 Series; 64-kbyte address space; low-voltage, low-power operation; on-chip supporting modules oriented toward consumer applications

This document gives a general description of the H8/3437 Series, a new series of 100-pin devices in the H8/300 family. The H8/3437 Series has an H8/300 CPU core and provides many of the peripheral functions needed in application systems on-chip: ROM, RAM, four types of timers, two channels of serial communication interface, host interface, A/D converter, D/A converter, and I/O ports. Compact, high-performance systems can be implemented easily.

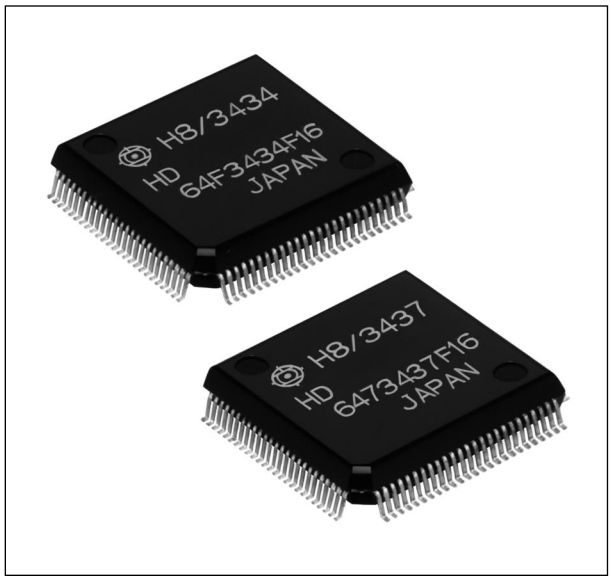
The H8/3437 Series includes ZTAT™*1 versions with on-chip PROM, and F-ZTAT™*2 versions with flash memory that enables application systems to be reprogrammed on-board. Together, the ZTAT™*1, F-ZTAT™*2, and mask-ROM versions offer a selection of devices to suit differing production volumes, firmness of specifications, and user conditions.

Hitachi endeavors to provide a full, efficient development environment for microcontroller application systems. Particular emphasis is being placed on development of the E3000 emulator, which connects easily to a personal computer, and a full suite of support software.

Notes: 1. ZTAT™ (Zero Turn-Around Time) is a registered trademark of Hitachi, Ltd.
2. F-ZTAT™ (Flexible-ZTAT) is a trademark of Hitachi, Ltd.

# Contents

# Section 1. Features of H8/3437 Series and H8/3434F-ZTAT™




- High-speed H8/300 CPU
  - — General-register machine
    Sixteen 8-bit general registers
    Also configurable as eight 16-bit general registers
  - — High speed
    Maximum clock rate: 16 MHz at 5 V, 10 MHz at 3 V
    High-speed instructions

| | |
|---|---|
| 8- or 16-bit register-register add | 125 ns (at 16 MHz), 200 ns (at 10 MHz) |
| 8 × 8-bit multiply | 875 ns (at 16 MHz), 1400 ns (at 10 MHz) |
| 16 ÷ 8-bit divide | 875 ns (at 16 MHz), 1400 ns (at 10 MHz) |

  - — Speed-oriented instruction set
    Fifty-seven types of instructions
    Instruction length: 2 or 4 bytes
    Fast multiply and divide instructions; powerful bit-manipulation instructions
  - — Maximum 64-kbyte address space

- 32 kbytes of flash memory, PROM, or masked ROM on-chip (H8/3434)
  60 kbytes of PROM or masked ROM on-chip (H8/3437)

- On-chip RAM: 1 kbyte (H8/3434); 2 kbytes (H8/3437)

- 16-Bit free-running timer on-chip (1 channel)
  - — One 16-bit free-running counter (can also count external events)
  - — Two output-compare registers
  - — Four input-capture registers (configurable for buffering)

- 8-Bit multifunction timer on-chip (2 channels)
  - — One 8-bit up-counter (can also count external events)
  - — Two time-constant registers
  - — One timer output

- 8-Bit PWM timer (2 channels)
  - — Duty cycle: 0 to 100%
  - — Resolution: 1/250

- Watchdog timer (1 channel)
  - — Usable as watchdog timer or interval timer

- Serial communication interface (2 channels)
  - — Asynchronous or clocked synchronous mode (selectable)
  - — Simultaneous transmit/receive (full duplex operation)
  - — On-chip baud rate generator

- Host interface
  - — Two 8-bit register sets (input data register, output data register, status register)
  - — Three host interrupt requests ($HIRQ_1$, $HIRQ_{11}$, $HIRQ_{12}$)
  - — Normal and high-speed gate $A_{20}$ output

- A/D converter
  - — Resolution: 10 bits
  - — Eight channels (selection of single or scan mode)
  - — A/D conversion can be externally triggered
  - — Sample-and-hold function

- D/A converter
  - — Resolution: 8 bits
  - — Two channels

- I/O ports
  - — Seventy-four input/output pins
    Sixteen pins can drive LEDs
    Forty-eight pins have MOS input pull-up transistors
  - — Eight input-only pins

- Interrupts
  - — Nine external interrupt pins (NMI, $\overline{IRQ_0}$ to $\overline{IRQ_7}$), not including key-sense interrupt pins
  - — Twenty-six internal interrupt sources
  - — Sixteen key-sense interrupt pins (same vector as IRQ6, can be used for wake-up from software standby mode)

- Operating modes
  - Mode 1 (expanded mode, on-chip ROM disabled)
  - Mode 2 (expanded mode, on-chip ROM enabled)
  - Mode 3 (single-chip mode)

- Power-down state
  - Sleep mode (wake-up by any interrupt)
  - Software standby mode (wake-up by IRQ0, IRQ1, IRQ2, IRQ6, or key-sense interrupt)
  - Hardware standby mode

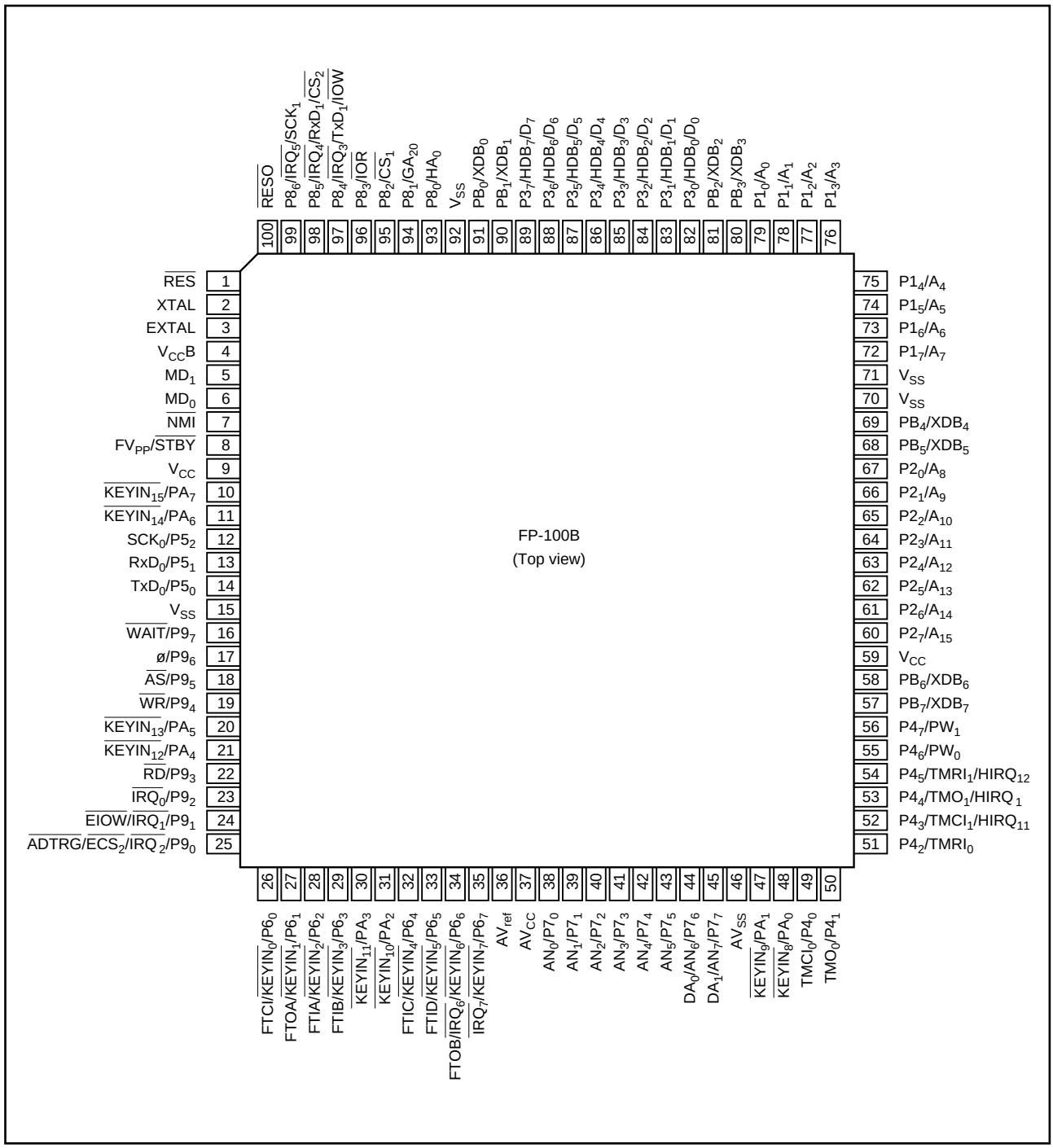
- Clock oscillator on-chip
  - System clock frequency equals oscillator frequency

- Package
  - 100-pin QFP (FP-100B)

# Section 2. Pin Assignments and Functions

## 2.1 Pin Assignments

- 100-pin QFP (FP-100B)



## 2.2 Pin Assignments in Each Operating Mode

| Pin No. | Pin Name | | |
|---|---|---|---|
| | Single-Chip Mode | Expanded Modes | |
| FP-100B | Mode 3 | Mode 2 | Mode 1 |
| 1 | RES | ← | ← |
| 2 | XTAL | ← | ← |
| 3 | EXTAL | ← | ← |
| 4 | $V_{CC}$B | ← | ← |
| 5 | $MD_1$ | ← | ← |
| 6 | $MD_0$ | ← | ← |
| 7 | NMI | ← | ← |
| 8 | STBY/$FV_{PP}$ | ← | STBY |
| 9 | $V_{CC}$ | ← | ← |
| 10 | $PA_7/\overline{KEYIN_{15}}$ | ← | ← |
| 11 | $PA_6/\overline{KEYIN_{14}}$ | ← | ← |
| 12 | $P5_2/SCK_0$ | ← | ← |
| 13 | $P5_1/RxD_0$ | ← | ← |
| 14 | $P5_0/TxD_0$ | ← | ← |
| 15 | $V_{SS}$ | ← | ← |
| 16 | $P9_7$ | $P9_7$/WAIT | ← |
| 17 | $P9_6$/ø | ø | ← |
| 18 | $P9_5$ | AS | ← |
| 19 | $P9_4$ | WR | ← |
| 20 | $PA_5/\overline{KEYIN_{13}}$ | ← | ← |
| 21 | $PA_4/\overline{KEYIN_{12}}$ | ← | ← |
| 22 | $P9_3$ | RD | ← |
| 23 | $P9_2/\overline{IRQ_0}$ | ← | ← |
| 24*1 | $P9_1/\overline{IRQ_1}$ | ← | ← |
| | EIOW/$\overline{IRQ_1}$ | | |
| 25*1 | $P9_0/\overline{IRQ_2}$/ADTRG | ← | ← |
| | $\overline{ECS_2}/\overline{IRQ_2}$ | | |
| 26 | $P6_0/\overline{KEYIN_0}$/FTCI | ← | ← |
| 27 | $P6_1/\overline{KEYIN_1}$/FTOA | ← | ← |
| 28 | $P6_2/\overline{KEYIN_2}$/FTIA | ← | ← |

Continued on next page

Note: 1. Upper entry applies when host interface is disabled or STAC = 0. Lower entry applies when host interface is enabled and STAC = 1.

| Pin No. | Pin Name | | |
|---|---|---|---|
| | Single-Chip Mode | Expanded Modes | |
| FP-100B | Mode 3 | Mode 2 | Mode 1 |
| 29 | $P6_3/\overline{KEYIN_3}$/FTIB | ← | ← |
| 30 | $PA_3/\overline{KEYIN_{11}}$ | ← | ← |
| 31 | $PA_2/\overline{KEYIN_{10}}$ | ← | ← |
| 32 | $P6_4/\overline{KEYIN_4}$/FTIC | ← | ← |
| 33 | $P6_5/\overline{KEYIN_5}$/FTID | ← | ← |
| 34 | $P6_6/\overline{KEYIN_6}/\overline{IRQ_6}$/FTOB | ← | ← |
| 35 | $P6_7/\overline{KEYIN_7}/\overline{IRQ_7}$ | ← | ← |
| 36 | $AV_{ref}$ | ← | ← |
| 37 | $AV_{CC}$ | ← | ← |
| 38 | $P7_0/AN_0$ | ← | ← |
| 39 | $P7_1/AN_1$ | ← | ← |
| 40 | $P7_2/AN_2$ | ← | ← |
| 41 | $P7_3/AN_3$ | ← | ← |
| 42 | $P7_4/AN_4$ | ← | ← |
| 43 | $P7_5/AN_5$ | ← | ← |
| 44 | $P7_6/AN_6/DA_0$ | ← | ← |
| 45 | $P7_7/AN_7/DA_1$ | ← | ← |
| 46 | $AV_{SS}$ | ← | ← |
| 47 | $PA_1/\overline{KEYIN_9}$ | ← | ← |
| 48 | $PA_0/\overline{KEYIN_8}$ | ← | ← |
| 49 | $P4_0/TMCI_0$ | ← | ← |
| 50 | $P4_1/TMO_0$ | ← | ← |
| 51 | $P4_2/TMRI_0$ | ← | ← |
| 52 | $P4_3/TMCI_1$ | ← | ← |
| | $TMCI_1/HIRQ_{11}$ | | |
| 53 | $P4_4/TMO_1$ | ← | ← |
| | $TMO_1/HIRQ_1$ | | |
| 54 | $P4_5/TMRI_1$ | ← | ← |
| | $TMRI_1/HIRQ_{12}$ | | |
| 55 | $P4_6/PW_0$ | ← | ← |
| 56 | $P4_7/PW_1$ | ← | ← |
| 57 | $PB_7$ | $PB_7$ | ← |
| | | $XDB_7$ | |

Continued on next page

Note: Upper entry applies when host interface is disabled. Lower entry applies when host interface is enabled.

| Pin No. | Pin Name | | |
|---|---|---|---|
| | Single-Chip Mode | Expanded Modes | |
| FP-100B | Mode 3 | Mode 2 | Mode 1 |
| 58 | $PB_6$ | $PB_6$ | ← |
| | | $XDB_6$ | |
| 59 | $V_{CC}$ | ← | ← |
| 60 | $P2_7$ | $P2_7/A_{15}$ | $A_{15}$ |
| 61 | $P2_6$ | $P2_6/A_{14}$ | $A_{14}$ |
| 62 | $P2_5$ | $P2_5/A_{13}$ | $A_{13}$ |
| 63 | $P2_4$ | $P2_4/A_{12}$ | $A_{12}$ |
| 64 | $P2_3$ | $P2_3/A_{11}$ | $A_{11}$ |
| 65 | $P2_2$ | $P2_2/A_{10}$ | $A_{10}$ |
| 66 | $P2_1$ | $P2_1/A_9$ | $A_9$ |
| 67 | $P2_0$ | $P2_0/A_8$ | $A_8$ |
| 68 | $PB_5$ | $PB_5$ | ← |
| | | $XDB_5$ | |
| 69 | $PB_4$ | $PB_4$ | ← |
| | | $XDB_4$ | |
| 70 | $V_{SS}$ | ← | ← |
| 71 | $V_{SS}$ | ← | ← |
| 72 | $P1_7$ | $P1_7/A_7$ | $A_7$ |
| 73 | $P1_6$ | $P1_6/A_6$ | $A_6$ |
| 74 | $P1_5$ | $P1_5/A_5$ | $A_5$ |
| 75 | $P1_4$ | $P1_4/A_4$ | $A_4$ |
| 76 | $P1_3$ | $P1_3/A_3$ | $A_3$ |
| 77 | $P1_2$ | $P1_2/A_2$ | $A_2$ |
| 78 | $P1_1$ | $P1_1/A_1$ | $A_1$ |
| 79 | $P1_0$ | $P1_0/A_0$ | $A_0$ |
| 80 | $PB_3$ | $PB_3$ | ← |
| | | $XDB_3$ | |
| 81 | $PB_2$ | $PB_2$ | ← |
| | | $XDB_2$ | |
| 82 | $P3_0$ | $D_0$ | ← |
| | $HDB_0$ | | |
| 83 | $P3_1$ | $D_1$ | ← |
| | $HDB_1$ | | |

Continued on next page

Note: Upper entry applies when host interface is disabled. Lower entry applies when host interface is enabled.

| Pin No. | Pin Name | | |
|---|---|---|---|
| | Single-Chip Mode | Expanded Modes | |
| FP-100B | Mode 3 | Mode 2 | Mode 1 |
| 84 | $P3_2$ | $D_2$ | ← |
| | $HDB_2$ | | |
| 85 | $P3_3$ | $D_3$ | ← |
| | $HDB_3$ | | |
| 86 | $P3_4$ | $D_4$ | ← |
| | $HDB_4$ | | |
| 87 | $P3_5$ | $D_5$ | ← |
| | $HDB_5$ | | |
| 88 | $P3_6$ | $D_6$ | ← |
| | $HDB_6$ | | |
| 89 | $P3_7$ | $D_7$ | ← |
| | $HDB_7$ | | |
| 90 | $PB_1$ | $PB_1$ | ← |
| | | $XDB_1$ | |
| 91 | $PB_0$ | $PB_0$ | ← |
| | | $XDB_0$ | |
| 92 | $V_{SS}$ | ← | ← |
| 93 | $P8_0$ | ← | ← |
| | $HA_0$ | | |
| 94 | $P8_1$ | ← | ← |
| | $P8_1/GA_{20}$ | | |
| 95 | $P8_2$ | ← | ← |
| | $\overline{CS_1}$ | | |
| 96 | $P8_3$ | ← | ← |
| | $\overline{IOR}$ | | |
| 97*2 | $P8_4/\overline{IRQ_3}/TxD_1$ | ← | ← |
| | $\overline{IOW}/\overline{IRQ_3}$ | | |
| 98*2 | $P8_5/\overline{IRQ_4}/RxD_1$ | ← | ← |
| | $\overline{CS_2}/\overline{IRQ_4}$ | | |
| 99 | $P8_6/\overline{IRQ_5}/SCK_1$ | ← | ← |
| 100 | $\overline{RESO}$ | ← | ← |

Notes: Upper entry applies when host interface is disabled. Lower entry applies when host interface is enabled.

2. Upper entry applies when host interface is disabled or STAC = 1. Lower entry applies when host interface is enabled and STAC = 0.

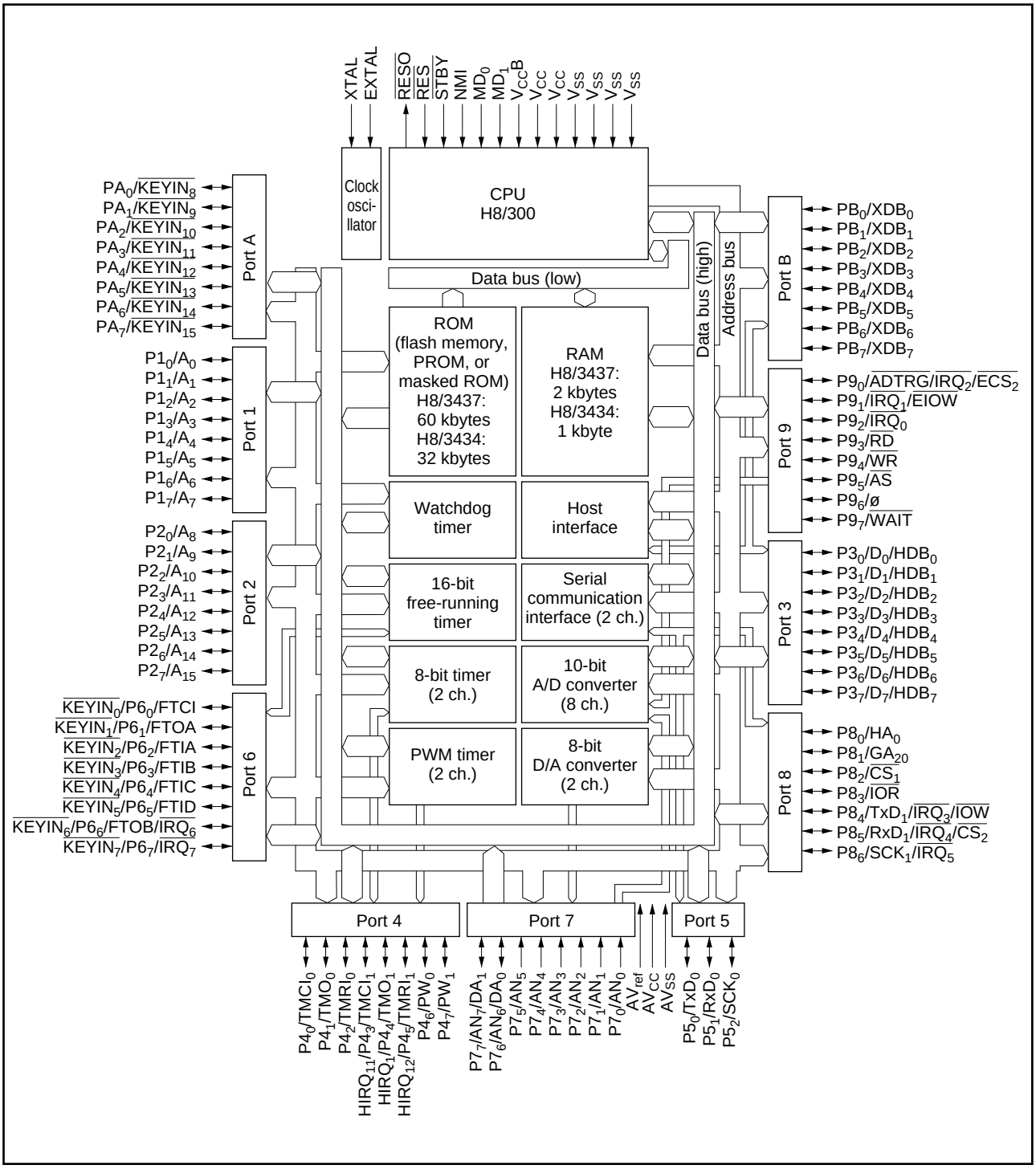
## 2.3 Pin Functions

| Type | Symbol | I/O | Function |
|---|---|---|---|
| Power | $V_{CC}$, $V_{CC}$B | I | Power supply, power supply for $P8_6$, $P9_7$, $PA_4$ to $PA_7$ |
| | $V_{SS}$ | I | Ground |
| Clock | XTAL | I | Crystal |
| | EXTAL | I | External clock |
| | ø | O | System clock |
| System control | STBY | I | Standby |
| | RES | I | Reset |
| | RESO | O | Reset output |
| Operating mode control | $MD_0$, $MD_1$ | I | Mode select |
| Address bus | $A_0$–$A_{15}$ | O | Address bus |
| Data bus | $D_0$–$D_7$ | I/O | Data bus |
| Bus control | AS | O | Address strobe |
| | RD | O | Read |
| | WR | O | Write |
| | WAIT | I | Wait |
| Interrupts | NMI | I | Nonmaskable interrupt |
| | $\overline{IRQ_0}$–$\overline{IRQ_7}$ | I | Interrupt request |
| | $\overline{KEYIN_0}$–$\overline{KEYIN_{15}}$ | I | Key-sense interrupt request |
| 16-bit free-running timer | FTCI | I | Counter clock input |
| | FTIA, FTIB, FTIC, FTID | I | Input-capture input |
| | FTOA, FTOB | O | Output-compare output |
| 8-bit timer | $TMCI_0$, $TMCI_1$ | I | Counter clock input |
| | $TMRI_0$, $TMRI_1$ | I | Counter reset input |
| | $TMO_0$, $TMO_1$ | O | Timer output |
| PWM timer | $PW_0$, $PW_1$ | O | PWM timer output |
| Serial communication interface (SCI) | $TxD_0$, $TxD_1$ | O | Transmit data output |
| | $RxD_0$, $RxD_1$ | I | Receive data input |
| | $SCK_0$, $SCK_1$ | I/O | Serial clock input/output |

Continued on next page

| Type | Symbol | I/O | Function |
|---|---|---|---|
| Host interface (HIF) | $HDB_0$–$HDB_7$, $XDB_0$–$XDB_7$ | I/O | Host data bus (single-chip mode, expanded modes) |
| | $HA_0$ | I | Host address bus 0 |
| | $\overline{CS_1}$, $\overline{CS_2}$, $\overline{ECS_2}$ | I | Host chip select |
| | IOR | I | Host read |
| | IOW, EIOW | I | Host write |
| | $GA_{20}$ | O | High-speed gate $A_{20}$ output |
| | $HIRQ_1$, $HIRQ_{11}$, $HIRQ_{12}$ | O | Host interrupt request output |
| A/D converter | $AN_0$–$AN_7$ | I | Analog input |
| | ADTRG | I | External trigger input for A/D converter |
| | $AV_{CC}$ | I | Analog power supply |
| | $AV_{SS}$ | I | Analog ground |
| | $AV_{ref}$ | I | Analog reference voltage |
| D/A converter | $DA_0$, $DA_1$ | O | D/A output |
| | $AV_{CC}$ | I | Analog power supply |
| | $AV_{SS}$ | I | Analog ground |
| | $AV_{ref}$ | I | Analog reference voltage |
| Flash memory | $FV_{PP}$ | I | Power supply for on-board programming |

# Section 3. Block Diagram




**Block Diagram of H8/3437 Series and H8/3434F-ZTAT™**

# Section 4. CPU

The H8/3437 series and H8/3434 have the generic H8/300 CPU: a central processing unit offering high cost-performance with sixteen 8-bit general registers and a concise, optimized instruction set. The general registers can also be paired as eight 16-bit registers. Arithmetic and logic operations and data transfers can be carried out at high speed with a maximum system clock rate of 16 MHz (at 5 V).

## 4.1 Address Space

The CPU supports a maximum address space of 64 kbytes. The address space configuration depends on the operating mode of the chip, which is selected by the inputs at the mode pins ($MD_0$ and $MD_1$). For details, see section 5, Operating Modes.

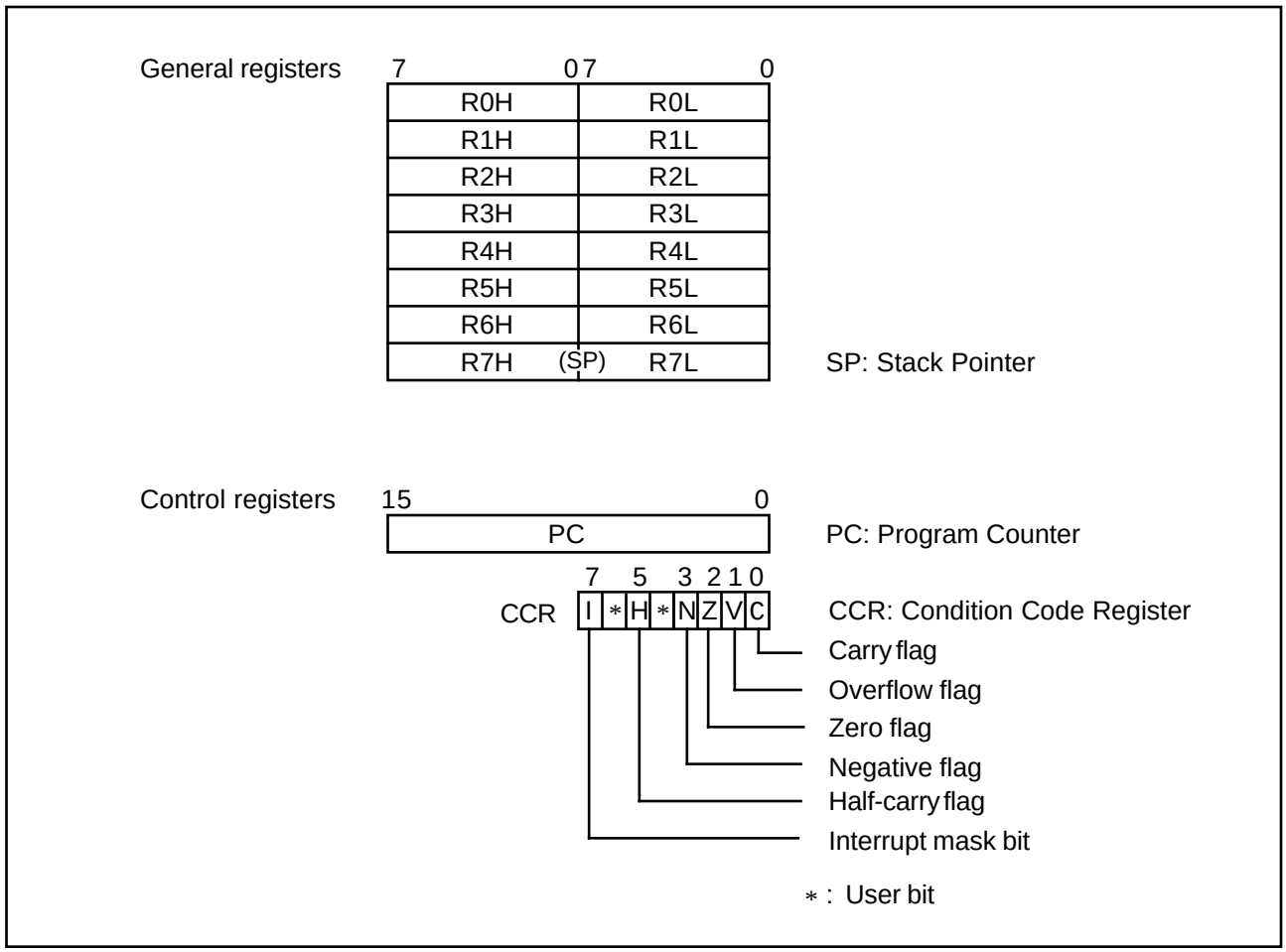
| Mode | $MD_1$ | $MD_0$ | Description | ROM | RAM | Interrupt Vector Table | Register Field |
|---|---|---|---|---|---|---|---|
| Mode 1 | 0 | 1 | Expanded mode | External | On-chip* | External | On-chip |
| Mode 2 | 1 | 0 | Expanded mode | On-chip | On-chip* | On-chip | On-chip |
| Mode 3 | 1 | 1 | Single-chip mode | On-chip | On-chip | On-chip | On-chip |

0: Low 1: High

* : External address space when the RAM enable (RAME) bit in the system control register is cleared to 0.

## 4.2 Register Configuration

The register structure of the CPU is shown below. Besides the 8-bit general registers (R0H/R0L to R7H/R7L) there is a 16-bit program counter (PC) and an 8-bit condition code register (CCR).




**CPU Registers**

• **General Registers:** All the general registers are functionally alike; there is no distinction between data registers and address registers. When used as address registers, the general registers are accessed as 16-bit registers (R0 to R7). When used as data registers, they can be accessed as 16-bit registers, or the high and low bytes can be accessed separately as 8-bit registers. The register length is determined by the instruction.

R7 also functions as the stack pointer (SP), used implicitly in exception handling and subroutine calls.

- **Control Registers**

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

**Condition Code Register (CCR):** This 8-bit register contains internal status information, including carry (C), overflow (V), zero (Z), negative (N), and half-carry (H) flags and the interrupt mask bit (I). Operations on the CCR can be performed by CCR logic/transfer instructions.

Bit 7—Interrupt Mask Bit (I): This bit masks interrupts (other than NMI) when set to 1. It is set to 1 at the beginning of interrupt handling.

Bit 6—User Bit: This bit can be written and read by software for its own purposes, using the CCR logic/transfer mentioned above.

Bit 5—Half-Carry (H): This bit is set to 1 when the ADD.B, ADDX.B, SUB.B, SUBX.B, or CMP.B instruction causes a carry or borrow out of bit 3, and is cleared to 0 when one of these instructions is executed without causing a carry out of bit 3. Similarly, it is set to 1 when the ADD.W, SUB.W, or CMP.W instruction causes a carry or borrow out of bit 11, and cleared to 0 when one of these instructions is executed without causing a carry or borrow out of bit 11. It is used implicitly by the DAA and DAS instructions.

Bit 4—User Bit: This bit can be written and read by software for its own purposes, using the CCR logic/transfer mentioned above.

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

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

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

Bit 0—Carry (C): This bit is used by:

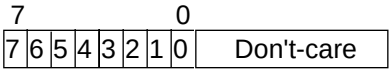
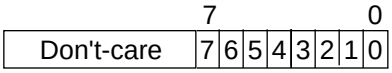
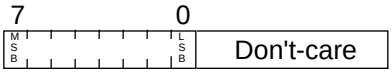
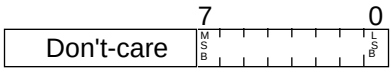
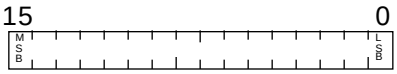
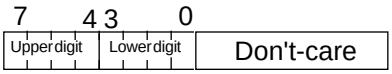
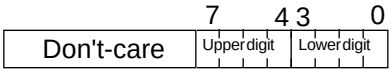
- Add and subtract instructions, to indicate a carry or borrow at the most significant bit
- Shift and rotate instructions, to store the value shifted out of the most significant or least significant bit
- Bit manipulation and bit load instructions, as a bit accumulator

## 4.3 Data Formats

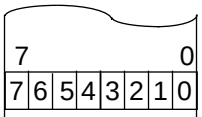
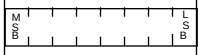
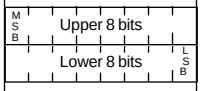
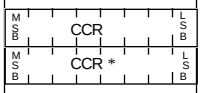
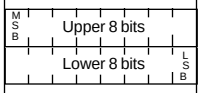
The CPU can process 1-bit data, 4-bit (BCD) data, 8-bit (byte) data, and 16-bit (word) data. Essentially all instructions can process byte data. The bit manipulation instructions process 1-bit data. Data transfer instructions and some arithmetic instructions handle word data. The decimal adjust instructions permit BCD data to be used.

The formats in which data are stored in general registers and memory are shown next.

- **Register Data Formats**

| Data Type | Register No. | Data Format |
|---|---|---|
| 1-Bit data | RnH |   |
| 1-Bit data | RnL |   |
| Byte data | RnH |   |
| Byte data | RnL |   |
| Word data | Rn |   |
| Byte data (packed BCD) | RnH |   |
| Byte data (packed BCD) | RnL |   |

• **Memory Data Formats**

| Data Type | Address | Data Format |
|---|---|---|
| 1-Bit data | Address n |   |
| Byte data | Address n |   |
| Word data | Even address<br>Odd address |   |
| Byte data (CCR) on stack | Even address<br>Odd address |   |
| Word data on stack | Even address<br>Odd address |   |

Note: Word data must begin at an even address.

## 4.4 Addressing Modes

The CPU supports the following eight addressing modes.

• **Addressing Modes**

| No. | Addressing Mode | Symbol |
|---|---|---|
| 1 | Register direct | Rn |
| 2 | Register indirect | @Rn |
| 3 | Register indirect with displacement | @(d:16, Rn) |
| 4 | Register indirect with pre-decrement | @–Rn |
| | or post-increment | @Rn+ |
| 5 | Immediate | #xx:8, #xx:16 |
| 6 | Absolute address | @aa:8, @aa:16 |
| 7 | PC-relative | @(d:8, PC) |
| 8 | Memory indirect | @@aa:8 |

Note: Data transfer instructions can use modes 1 through 6.

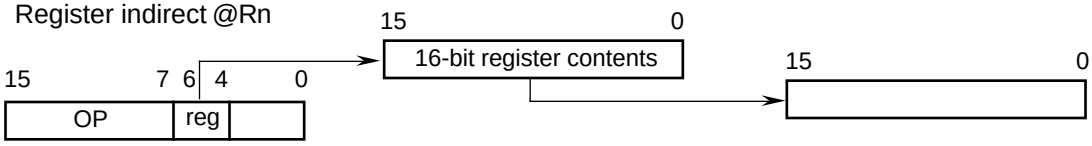
• **Effective Address Calculation**

| No. | Addressing Mode, Instruction Format | Effective Address Calculation | Effective Address |
|---|---|---|---|

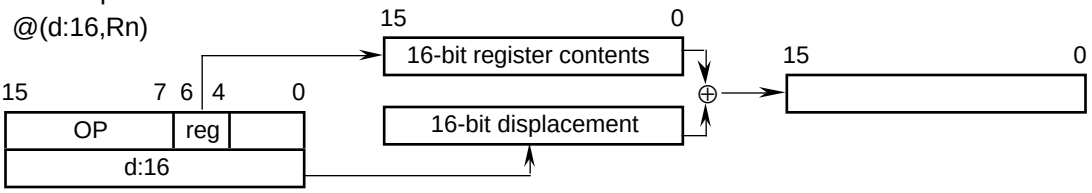
1 Register direct Rn.


Operands are contained in registers 1 and 2

2 Register indirect @Rn

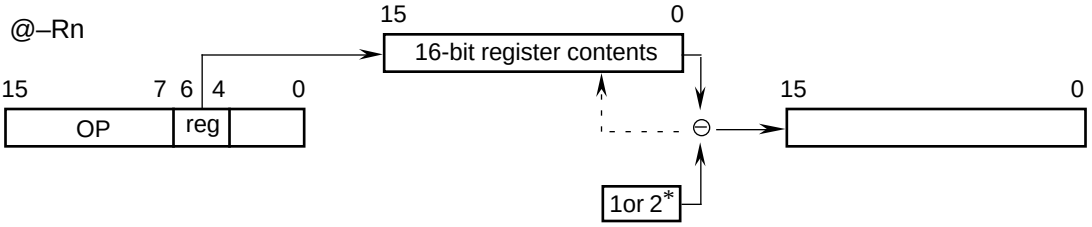



3 Register indirect with displacement @(d:16,Rn)



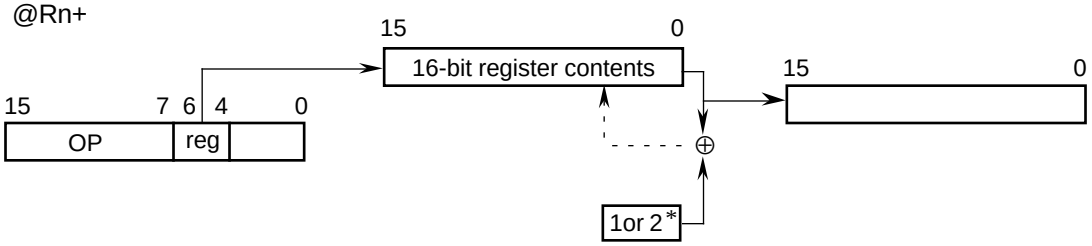

4 Register indirect with pre-decrement

@–Rn

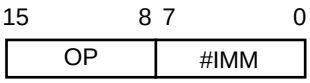
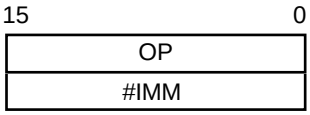
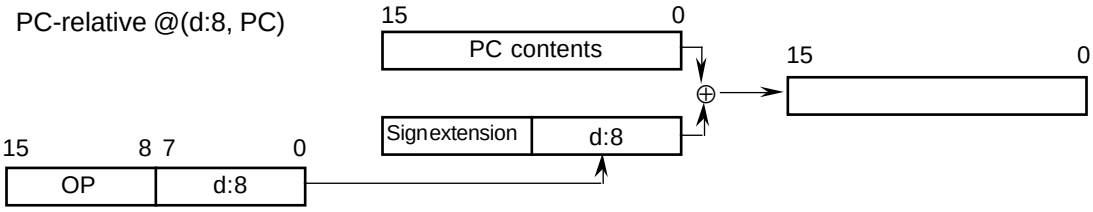
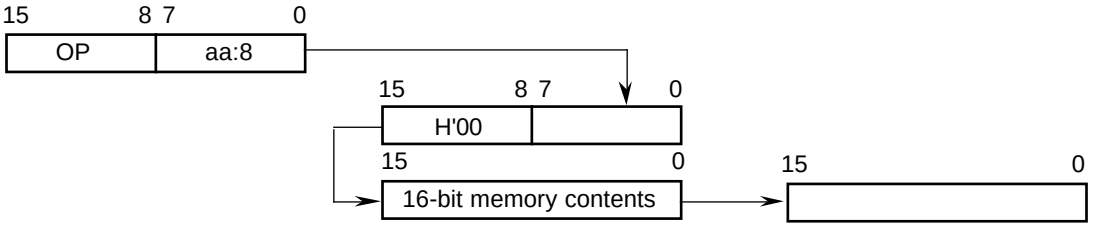



Register indirect with post-increment

@Rn+




∗ : 1 for a byte operand,
2 for a word operand

| No. | Addressing Mode, Instruction Format | Effective Address Calculation | Effective Address |
|---|---|---|---|
| 5 | Immediate #xx:8<br>  | | Operand is 1-byte immediate data |
| | Immediate #xx:16<br>  | | Operand is 2-byte immediate data |
| 6 | Absolute address @aa:8<br>  | | |
| | Absolute address @aa:16<br>  | | |
| 7 | PC-relative @(d:8, PC)<br>  | | |
| 8 | Memory indirect @@aa:8<br>  | | |

reg: General register
#IMM: Immediate data
d: Displacement
aa: Absolute address

## 4.5 Instruction Set

The CPU executes 57 types of instructions, among which the MOVFPE and MOVTPE instructions are not supported. The instruction set has the following features.

- **Features**

  — Concise instruction set suited for high-speed performance; all instructions are 2 or 4 bytes long

  — High speed
  All frequently-used instructions execute in 2 to 4 states
  A 16-bit register-register add takes 125 ns (with 16 MHz system clock)

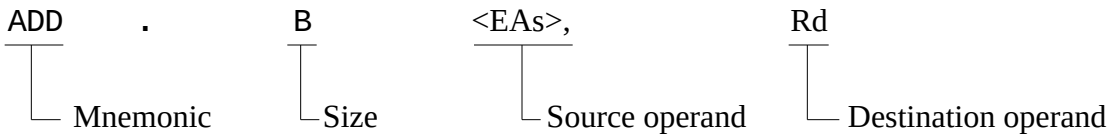
  — General-register architecture

  — Powerful bit-manipulation instructions

  — Standard H-Series mnemonics

- **Assembly-Language Format**

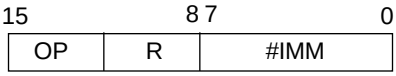
The ADD instruction below gives an example of the assembly-language format. The letter B (byte) or W (word) designates the operand size. For some instructions only one of these sizes is available.



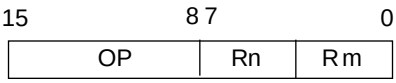
• **Main Instruction Formats**

The main instruction formats of the H8/300 CPU are shown below.

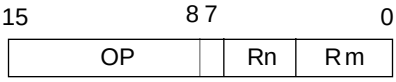
• Arithmetic or logic operation on register contents and immediate data



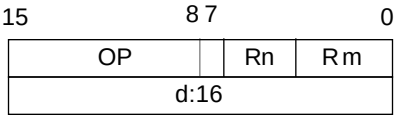

• Register-register arithmetic or logic operation



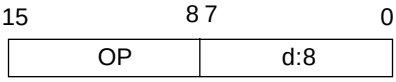

• Data transfer [@Rn↔Rm]



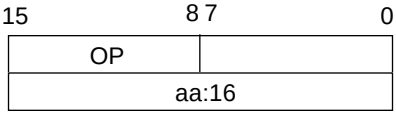

• Data transfer [@(d:16, Rn)↔Rm]



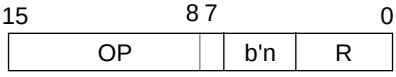

• Branching instruction [@(d:8, PC)]




• Branching instruction [@aa:16]




• Bit manipulation instruction (with direct specification of bit position)

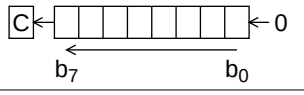
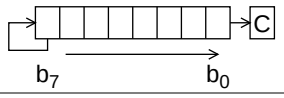
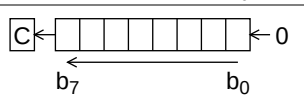
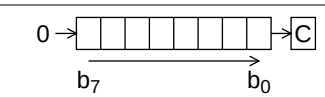
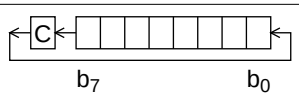
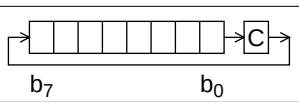
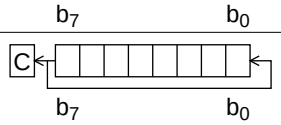
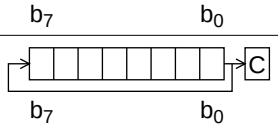



| | |
|---|---|
| OP: | Operation code |
| R, Rn, Rm: | General registers |
| #IMM: | Immediate data |
| d: | Displacement |
| aa: | Absolute address |
| b'n: | Bit number |

• **Instruction Set List**

| | Mnemonic | Operand Size | Operation | Addressing Mode/ Instruction Length | | | | | | | | | Condition Code | | | | | | No. of States |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #xx: | Rn | @Rn | @(d:16,Rn) | @-Rn/@Rn+ | @aa: | @(d:8,PC) | @@aa | Implied | I | H | N | Z | V | C | |
| Data Transfer Instructions | MOV.B Rs,Rd | B | Rs8 → Rd8 | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | MOV.B #xx:8,Rd | B | #xx:8 → Rd8 | 2 | | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | MOV.B @Rs,Rd | B | @Rs16 → Rd8 | | | 2 | | | | | | | – | – | ↕ | ↕ | 0 | – | 4 |
| | MOV.B @(d:16,Rs),Rd | B | @(d:16,Rs16)→ Rd8 | | | | 4 | | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.B @Rs+,Rd | B | @Rs16 → Rd8<br>Rs16+1 → Rs16 | | | | | 2 | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.B @aa:8,Rd | B | @aa:8 → Rd8 | | | | | | 2 | | | | – | – | ↕ | ↕ | 0 | – | 4 |
| | MOV.B @aa:16,Rd | B | @aa:16 → Rd8 | | | | | | 4 | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.B Rs,@Rd | B | Rs8 → @Rd16 | | | 2 | | | | | | | – | – | ↕ | ↕ | 0 | – | 4 |
| | MOV.B Rs,@(d:16,Rd) | B | Rs8 → @(d:16,Rd16) | | | | 4 | | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.B Rs,@–Rd | B | Rd16–1 → Rd16<br>Rs8 → @Rd16 | | | | | 2 | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.B Rs,@aa:8 | B | Rs8 → @aa:8 | | | | | | 2 | | | | – | – | ↕ | ↕ | 0 | – | 4 |
| | MOV.B Rs,@aa:16 | B | Rs8 → @aa:16 | | | | | | 4 | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.W Rs,Rd | W | Rs16 → Rd16 | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | MOV.W @Rs,Rd | W | @Rs16 → Rd16 | | | 2 | | | | | | | – | – | ↕ | ↕ | 0 | – | 4 |
| | MOV.W @(d:16,Rs),Rd | W | @(d:16,Rs16) → Rd16 | | | | 4 | | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.W @Rs+,Rd | W | @Rs16 → Rd16<br>Rs16+2 → Rs16 | | | | | 2 | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.W @aa:16,Rd | W | @aa:16 → Rd16 | | | | | | 4 | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.W Rs,@Rd | W | Rs16 → @Rd16 | | | 2 | | | | | | | – | – | ↕ | ↕ | 0 | – | 4 |
| | MOV.W Rs,@(d:16,Rd) | W | Rs16 → @(d:16,Rd16) | | | | 4 | | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.W Rs,@–Rd | W | Rd16–2 → Rd16<br>Rs16 → @Rd16 | | | | | 2 | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.W Rs, @aa:16 | W | Rs16 → @aa:16 | | | | | | 4 | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOV.W #xx:16,Rd | W | #xx:16 → Rd | 4 | | | | | | | | | – | – | ↕ | ↕ | 0 | – | 4 |
| | POP Rd | W | @SP+ → Rd | | | | | 2 | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | PUSH Rs | W | Rs → @–SP | | | | | 2 | | | | | – | – | ↕ | ↕ | 0 | – | 6 |
| | MOVFPE @aa:16,Rd | B | Not supported | | | | | | 4 | | | | | | | | | | |
| | MOVTPE Rs,@aa:16 | B | | | | | | | 4 | | | | | | | | | | |
| | EEPMOV | B | if R4L≠0 then<br>Repeat @R5 → @R6<br>R5+1 → R5<br>R6+1 → R6<br>R4L–1 → R4L<br>Until R4L=0<br>else next | | | | | | | | | 4 | – | – | – | – | – | – | ④ |

| | Mnemonic | Operand Size | Operation | Addressing Mode/ Instruction Length | | | | | | | | | Condition Code | | | | | | No. of States |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #xx: | Rn | @Rn | @(d:16,Rn) | @-Rn/@Rn+ | @aa: | @(d:8,PC) | @@aa | Implied | I | H | N | Z | V | C | |
| Arithmetic Instructions | ADD.B #xx:8,Rd | B | Rd8+#xx:8 → Rd8 | 2 | | | | | | | | | – | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | ADD.B Rs,Rd | B | Rs8+Rd8 → Rd8 | | 2 | | | | | | | | – | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | ADD.W Rs,Rd | W | Rs16+Rd16 → Rd16 | | 2 | | | | | | | | – | ① | ↕ | ↕ | ↕ | ↕ | 2 |
| | ADDX.B #xx:8,Rd | B | Rd8+#xx:8 +C → Rd8 | 2 | | | | | | | | | – | ↕ | ↕ | ② | ↕ | ↕ | 2 |
| | ADDX.B Rs,Rd | B | Rd8+Rs8 +C → Rd8 | | 2 | | | | | | | | – | ↕ | ↕ | ② | ↕ | ↕ | 2 |
| | ADDS.W #1,Rd | W | Rd16+1 → Rd16 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | ADDS.W #2,Rd | W | Rd16+2 → Rd16 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | INC.B Rd | B | Rd8+1 → Rd8 | | 2 | | | | | | | | – | – | ↕ | ↕ | ↕ | – | 2 |
| | DAA.B Rd | B | Rd8 decimal adjust → Rd8 | | 2 | | | | | | | | – | * | ↕ | ↕ | * | ③ | 2 |
| | NEG.B Rd | B | 0–Rd → Rd | | 2 | | | | | | | | – | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | SUB.B Rs,Rd | B | Rd8–Rs8 → Rd8 | | 2 | | | | | | | | – | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | SUB.W Rs,Rd | W | Rd16–Rs16 → Rd16 | | 2 | | | | | | | | – | ① | ↕ | ↕ | ↕ | ↕ | 2 |
| | SUBX.B #xx:8,Rd | B | Rd8–#xx:8 –C → Rd8 | 2 | | | | | | | | | – | ↕ | ↕ | ② | ↕ | ↕ | 2 |
| | SUBX.B Rs,Rd | B | Rd8–Rs8 → Rd8 | | 2 | | | | | | | | – | ↕ | ↕ | ② | ↕ | ↕ | 2 |
| | SUBS.W #1,Rd | W | Rd16–1 → Rd16 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | SUBS.W #2,Rd | W | Rd16–2 → Rd16 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | DEC.B Rd | B | Rd8–1 → Rd8 | | 2 | | | | | | | | – | – | ↕ | ↕ | ↕ | – | 2 |
| | DAS.B Rd | B | Rd8 decimal adjust → Rd8 | | 2 | | | | | | | | – | * | ↕ | ↕ | * | – | 2 |
| | CMP.B #xx:8,Rd | B | Rd8–#xx:8 | 2 | | | | | | | | | – | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | CMP.B Rs,Rd | B | Rd8–Rs8 | | 2 | | | | | | | | – | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | CMP.W Rs,Rd | W | Rd16–Rs16 | | 2 | | | | | | | | – | ① | ↕ | ↕ | ↕ | ↕ | 2 |
| | MULXU.B Rs,Rd | B | Rd8×Rs8 → Rd16 | | 2 | | | | | | | | – | – | – | – | – | – | 14 |
| | DIVXU.B Rs,Rd | B | Rd16÷Rs8 → Rd16<br>(RdH:remainder,RdL:quotient) | | 2 | | | | | | | | – | – | ↕ | ↕ | – | – | 14 |
| Logic Instructions | AND.B #xx:8,Rd | B | Rd8∧#xx:8 → Rd8 | 2 | | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | AND.B Rs,Rd | B | Rd8∧Rs8 → Rd8 | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | OR.B #xx:8,Rd | B | Rd8∨#xx:8 → Rd8 | 2 | | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | OR.B Rs,Rd | B | Rd8∨Rs8 → Rd8 | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | XOR.B #xx:8,Rd | B | Rd8⊕#xx:8 → Rd8 | 2 | | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | XOR.B Rs,Rd | B | Rd8⊕Rs8 → Rd8 | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |
| | NOT.B Rd | B | $\overline{\text{Rd}}$ → Rd | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | – | 2 |

| | Mnemonic | Operand Size | Operation | Addressing Mode/ Instruction Length | | | | | | | | | Condition Code | | | | | | No. of States |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #xx: | Rn | @Rn | @(d:16,Rn) | @-Rn/@Rn+ | @aa: | @(d:8,PC) | @@aa | Implied | I | H | N | Z | V | C | |
| Shift Instructions | SHAL.B Rd | B |  | | 2 | | | | | | | | – | – | ↕ | ↕ | ↕ | ↕ | 2 |
| | SHAR.B Rd | B |  | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | ↕ | 2 |
| | SHLL.B Rd | B |  | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | ↕ | 2 |
| | SHLR.B Rd | B |  | | 2 | | | | | | | | – | – | 0 | ↕ | 0 | ↕ | 2 |
| | ROTXL.B Rd | B |  | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | ↕ | 2 |
| | ROTXR.B Rd | B |  | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | ↕ | 2 |
| | ROTL.B Rd | B |  | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | ↕ | 2 |
| | ROTR.B Rd | B |  | | 2 | | | | | | | | – | – | ↕ | ↕ | 0 | ↕ | 2 |
| Bit Manipulation Instructions | BSET #xx:3,Rd | B | (#xx:3 of Rd8) ← 1 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BSET #xx:3,@Rd | B | (#xx:3 of @Rd16) ← 1 | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BSET #xx:3,@aa:8 | B | (#xx:3 of @aa:8) ← 1 | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | BSET Rn,Rd | B | (Rn8 of Rd8) ← 1 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BSET Rn,@Rd | B | (Rn8 of @Rd16) ← 1 | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BSET Rn,@aa:8 | B | (Rn8 of @aa:8) ← 1 | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | BCLR #xx:3,Rd | B | (#xx:3 of Rd8) ← 0 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BCLR #xx:3,@Rd | B | (#xx:3 of @Rd16) ← 0 | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BCLR #xx:3,@aa:8 | B | (#xx:3 of @aa:8) ← 0 | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | BCLR Rn,Rd | B | (Rn8 of Rd8) ← 0 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BCLR Rn,@Rd | B | (Rn8 of @Rd16) ← 0 | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BCLR Rn,@aa:8 | B | (Rn8 of @aa:8) ← 0 | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | BNOT #xx:3,Rd | B | (#xx:3 of Rd8)←$\overline{\text{(#xx:3 of Rd8)}}$ | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BNOT #xx:3,@Rd | B | (#xx:3 of @Rd16)←$\overline{\text{(#xx:3 of @Rd16)}}$ | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BNOT #xx:3,@aa:8 | B | (#xx:3 of @aa:8)←$\overline{\text{(#xx:3 of @aa:8)}}$ | | | | | | 4 | | | | – | – | – | – | – | – | 8 |

| | Mnemonic | Operand Size | Operation | Addressing Mode/ Instruction Length | | | | | | | | | Condition Code | | | | | | No. of States |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #xx: | Rn | @Rn | @(d:16,Rn) | @-Rn/@Rn+ | @aa: | @(d:8,PC) | @@aa | Implied | I | H | N | Z | V | C | |
| Bit Manipulation Instructions | BNOT Rn,Rd | B | (Rn8 of Rd8) ← $\overline{\text{(Rn8 of Rd8)}}$ | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BNOT Rn,@Rd | B | (Rn8 of @Rd16) ← $\overline{\text{(Rn8 of @Rd16)}}$ | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BNOT Rn,@aa:8 | B | (Rn8 of @aa:8) ← $\overline{\text{(Rn8 of @aa:8)}}$ | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | BTST #xx:3,Rd | B | $\overline{\text{(\#xx:3 of Rd8)}}$ → Z | | 2 | | | | | | | | – | – | – | ↕ | – | – | 2 |
| | BTST #xx:3,@Rd | B | $\overline{\text{(\#xx:3 of @Rd16)}}$ → Z | | | 4 | | | | | | | – | – | – | ↕ | – | – | 6 |
| | BTST #xx:3,@aa:8 | B | $\overline{\text{(\#xx:3 of @aa:8)}}$ → Z | | | | | | 4 | | | | – | – | – | ↕ | – | – | 6 |
| | BTST Rn,Rd | B | $\overline{\text{(Rn8 of Rd8)}}$ → Z | | 2 | | | | | | | | – | – | – | ↕ | – | – | 2 |
| | BTST Rn,@Rd | B | $\overline{\text{(Rn8 of @Rd16)}}$ → Z | | | 4 | | | | | | | – | – | – | ↕ | – | – | 6 |
| | BTST Rn,@aa:8 | B | $\overline{\text{(Rn8 of @aa:8)}}$ → Z | | | | | | 4 | | | | – | – | – | ↕ | – | – | 6 |
| | BLD #xx:3,Rd | B | (#xx:3 of Rd8) → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |
| | BLD #xx:3,@Rd | B | (#xx:3 of @Rd16) → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BLD #xx:3,@aa:8 | B | (#xx:3 of @aa:8) → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| | BILD #xx:3,Rd | B | $\overline{\text{(\#xx:3 of Rd8)}}$ → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |
| | BILD #xx:3,@Rd | B | $\overline{\text{(\#xx:3 of @Rd16)}}$ → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BILD #xx:3,@aa:8 | B | $\overline{\text{(\#xx:3 of @aa:8)}}$ → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| | BST #xx:3,Rd | B | C → (#xx:3 of Rd8) | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BST #xx:3,@Rd | B | C → (#xx:3 of @Rd16) | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BST #xx:3,@aa:8 | B | C → (#xx:3 of @aa:8) | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | BIST #xx:3,Rd | B | $\overline{\text{C}}$ → (#xx:3 of Rd8) | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | BIST #xx:3,@Rd | B | $\overline{\text{C}}$ → (#xx:3 of @Rd16) | | | 4 | | | | | | | – | – | – | – | – | – | 8 |
| | BIST #xx:3,@aa:8 | B | $\overline{\text{C}}$ → (#xx:3 of @aa:8) | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | BAND #xx:3,Rd | B | C∧(#xx:3 of Rd8) → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |
| | BAND #xx:3,@Rd | B | C∧(#xx:3 of @Rd16) → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BAND #xx:3,@aa:8 | B | C∧(#xx:3 of @aa:8) → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| | BIAND #xx:3,Rd | B | C∧$\overline{\text{(\#xx:3 of Rd8)}}$ → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |
| | BIAND #xx:3,@Rd | B | C∧$\overline{\text{(\#xx:3 of @Rd16)}}$ → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BIAND #xx:3, @aa:8 | B | C∧$\overline{\text{(\#xx:3 of @aa:8)}}$ → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| | BOR #xx:3,Rd | B | C∨(#xx:3 of Rd8) → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |
| | BOR #xx:3,@Rd | B | C∨(#xx:3 of @Rd16) → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BOR #xx:3, @aa:8 | B | C∨(#xx:3 of @aa:8) → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| | BIOR #xx:3,Rd | B | C∨$\overline{\text{(\#xx:3 of Rd8)}}$ → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |

| | Mnemonic | Operand Size | Operation | Addressing Mode/ Instruction Length | | | | | | | | | Condition Code | | | | | | No. of States |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #xx: | Rn | @Rn | @(d:16,Rn) | @-Rn/@Rn+ | @aa: | @(d:8,PC) | @@aa | Implied | I | H | N | Z | V | C | |
| Bit Manipulation Instructions | BIOR #xx:3,@Rd | B | C∨(#xx:3 of @Rd16) → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BIOR #xx:3, @aa:8 | B | C∨(#xx:3 of @aa:8) → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| | BXOR #xx:3,Rd | B | C⊕(#xx:3 of Rd8) → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |
| | BXOR #xx:3,@Rd | B | C⊕(#xx:3 of @Rd16) → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BXOR #xx:3, @aa:8 | B | C⊕(#xx:3 of @aa:8) → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| | BIXOR #xx:3,Rd | B | C⊕(#xx:3 of Rd8) → C | | 2 | | | | | | | | – | – | – | – | – | ↕ | 2 |
| | BIXOR #xx:3,@Rd | B | C⊕(#xx:3 of @Rd16) → C | | | 4 | | | | | | | – | – | – | – | – | ↕ | 6 |
| | BIXOR #xx:3, @aa:8 | B | C⊕(#xx:3 of @aa:8) → C | | | | | | 4 | | | | – | – | – | – | – | ↕ | 6 |
| Branching Instructions | BRA(BT) | – | PC ← PC+d:8 | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BRN(BF) | – | No operation | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BHI | – | if true then PC ← PC+d:8 else next | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BLS | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BCC(BHS) | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BCS(BLO) | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BNE | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BEQ | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BVC | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BVS | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BPL | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BMI | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BGE | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BLT | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BGT | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | BLE | – | | | | | | | | 2 | | | – | – | – | – | – | – | 4 |
| | JMP @Rn. | – | PC ← Rn16 | | | 2 | | | | | | | – | – | – | – | – | – | 4 |
| | JMP @aa:16 | – | PC ← aa:16 | | | | | | 4 | | | | – | – | – | – | – | – | 6 |
| | JMP @@aa:8 | – | PC ← @aa:8 | | | | | | | | 2 | | – | – | – | – | – | – | 8 |
| | BSR | – | SP–2 → SP<br>PC → @SP<br>PC ← PC+d:8 | | | | | | | 2 | | | – | – | – | – | – | – | 6 |

| | Mnemonic | Operand Size | Operation | Addressing Mode/ Instruction Length | | | | | | | | | Condition Code | | | | | | No. of States |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | #xx: | Rn | @Rn | @(d:16,Rn) | @-Rn/@Rn+ | @aa: | @(d:8,PC) | @@aa | Implied | I | H | N | Z | V | C | |
| Branching Instructions | JSR @Rn | – | SP–2 → SP<br>PC → @SP<br>PC ← Rn16 | | | 2 | | | | | | | – | – | – | – | – | – | 6 |
| | JSR @aa:16 | – | SP–2 → SP<br>PC → @SP<br>PC ← aa:16 | | | | | | 4 | | | | – | – | – | – | – | – | 8 |
| | JSR @@aa:8 | | SP–2 → SP<br>PC → @SP<br>PC ← @aa:8 | | | | | | | | 2 | | – | – | – | – | – | – | 8 |
| | RTS | – | PC ← @SP<br>SP+2 → SP | | | | | | | | | 2 | – | – | – | – | – | – | 8 |
| System Control Instructions | RTE | – | CCR ← @SP<br>SP+2 → SP<br>PC ← @SP<br>SP+2 → SP | | | | | | | | | 2 | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 10 |
| | SLEEP | – | Transit to sleep mode. | | | | | | | | | 2 | – | – | – | – | – | – | 2 |
| | LDC #xx:8,CCR | B | #xx:8 → CCR | 2 | | | | | | | | | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | LDC Rs,CCR | B | Rs8 → CCR | | 2 | | | | | | | | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | STC CCR,Rd | B | CCR → Rd8 | | 2 | | | | | | | | – | – | – | – | – | – | 2 |
| | ANDC #xx:8,CCR | B | CCR∧#xx:8 → CCR | 2 | | | | | | | | | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | ORC #xx:8,CCR | B | CCR∨#xx:8 → CCR | 2 | | | | | | | | | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | XORC #xx:8,CCR | B | CCR⊕#xx:8 → CCR | 2 | | | | | | | | | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | 2 |
| | NOP | – | No operation | | | | | | | | | 2 | – | – | – | – | – | – | 2 |

Notes: The number of states is the number of states required for execution when the instruction and its operands are located in on-chip memory.

① Set to 1 when there is a carry or borrow from bit 11; otherwise cleared to 0.

② If the result is zero, the previous value of the flag is retained; otherwise the flag is cleared to 0.

③ Set to 1 if decimal adjustment produces a carry; otherwise cleared to 0.

④ The number of states required for execution is 4n+8, where n is the value of R4L.

**Number of States Required for Instruction Execution:** The number of states indicated in the instruction set list applies when the instruction and its operands are located in on-chip memory. If the instruction or an operand must be accessed at an off-chip address or in the on-chip register field, the number of states increases as indicated in the following table.

| Access Type | | Location | Size | Additional States |
|---|---|---|---|---|
| Operand data | | On-chip supporting module | Byte data | 1 |
| | | | Word data | 4 |
| | | External address | Byte data | 1 + m |
| | | | Word data | 4 + 2m |
| Instruction | Non-branching instruction | External address | 2-byte instruction | 4 + 2m |
| | | | 4-byte instruction | 8 + 4m |
| | Branching instruction | | 2-byte instruction | 8 + 4m |
| | | | 4-byte instruction | 8 + 4m |

Note: m—number of wait states inserted in access to external device.

## Operation Notation

| Symbol | Meaning |
|---|---|
| PC | Program counter |
| SP | Stack pointer |
| CCR | Condition code register |
| Z | Zero flag in CCR |
| C | Carry flag in CCR |
| Rs, Rd, Rn | General registers (8-bit—R0H/R0L to R7H/L7U; 16-bit—R0 to R7) |
| d:8, d:16 | 8-Bit or 16-bit displacement |
| #xx:3, #xx:8, #xx:16 | 3-Bit, 8-bit or 16-bit immediate data |
| → | The result of the operation on the left is assigned to the operand on the right. |
| + | Addition |
| – | Subtraction |
| × | Multiplication |
| ÷ | Division |
| ∧ | AND logical |
| ∨ | OR logical |
| ⊕ | Exclusive OR logical |
| ¬ | Not |
| ( ) and < > | Contents of effective address |

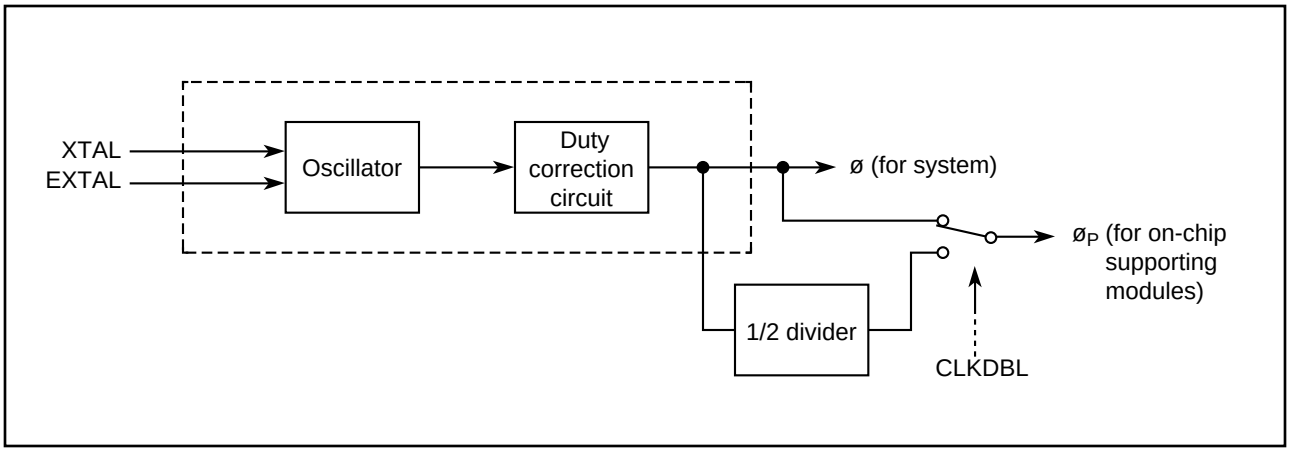
**Condition Code Notation**

| Symbol | Meaning |
|---|---|
| ↕ | The flag is altered according to the result of the instruction. |
| * | Undetermined; the flag is left in an unpredictable state. |
| 0 | The flag is cleared to 0. |
| — | The flag is not changed. |

## 4.6 Bus Timing

- **System Clock Timing**

The system clock (ø) is generated from an external clock signal input at the EXTAL pin, or by connecting a crystal oscillator across the XTAL and EXTAL pins. The system clock frequency is the same as the oscillator frequency. For timers and other on-chip supporting modules there is a choice between using the system clock frequency or dividing this frequency by two. This choice is controlled by software with the CLKDBL bit.

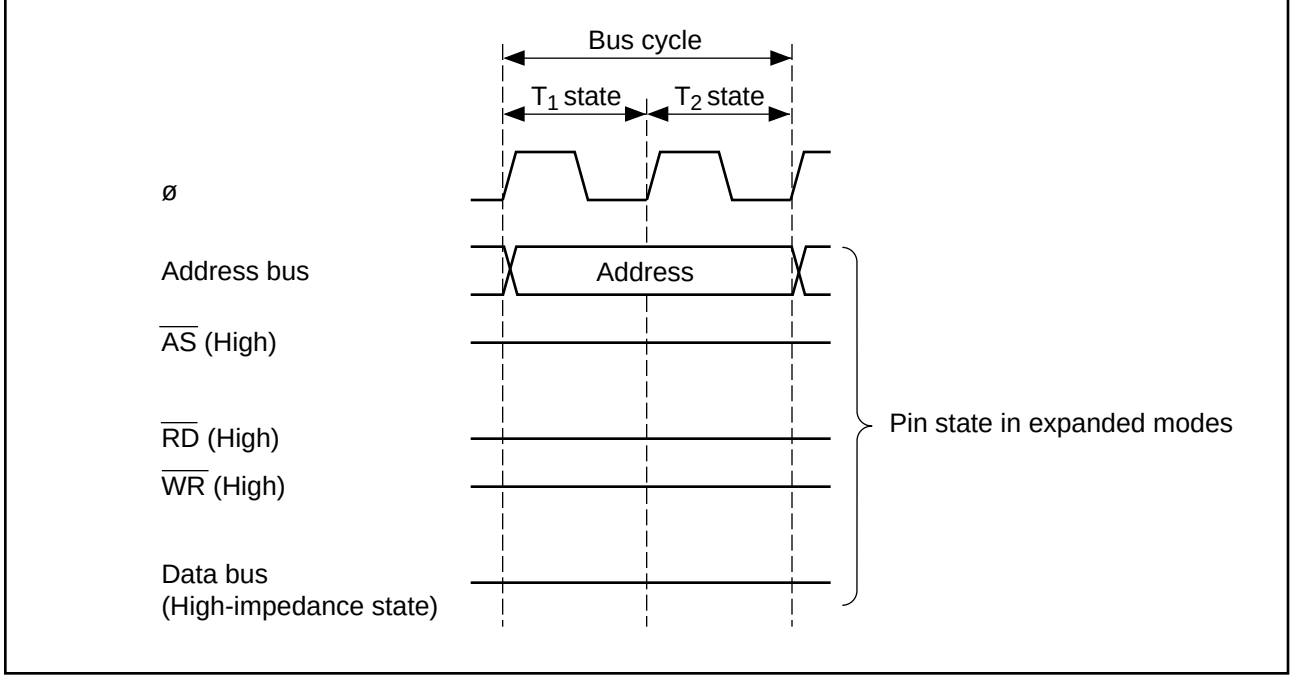



**Block Diagram of Clock Oscillator**

- **CPU Read/Write Cycle**

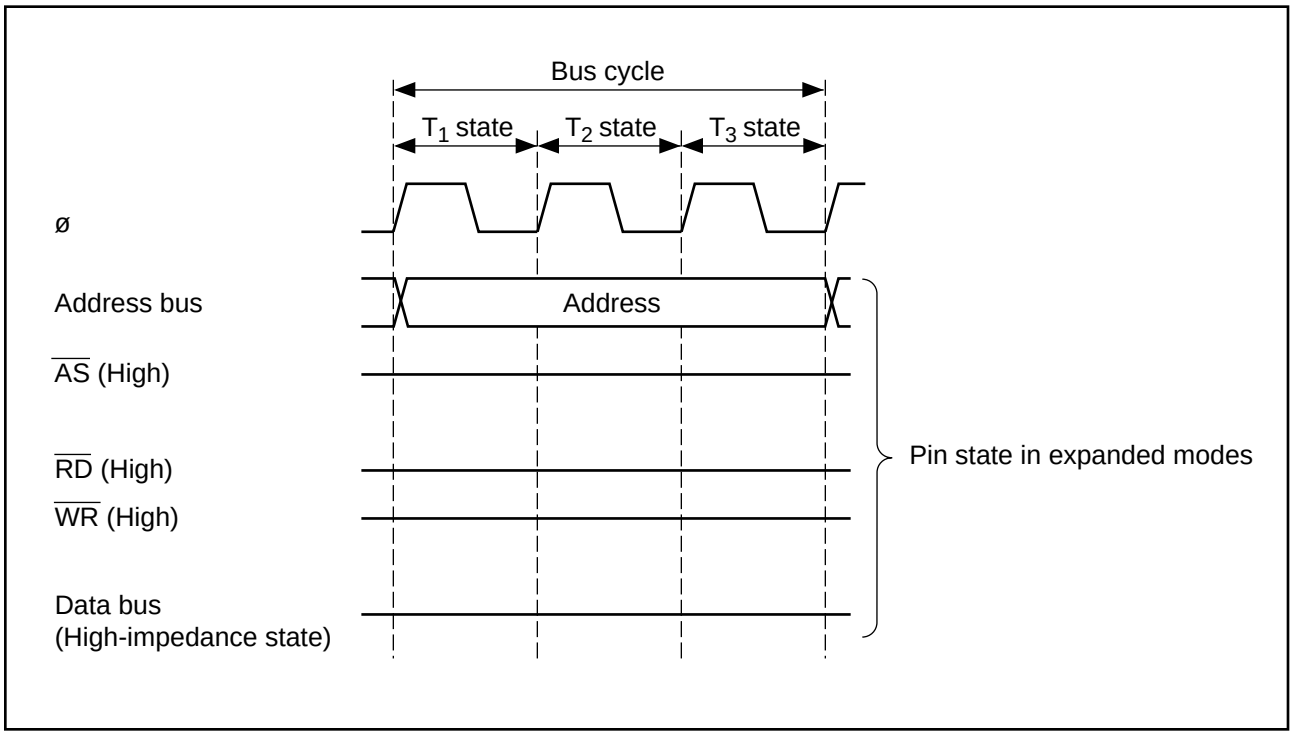
The CPU operates on the system clock. One period of the system clock is called a "state." The structure of the bus cycle depends on whether the access is to on-chip memory, the on-chip register field, or an external address. The basic bus cycle length is two or three states.

- **On-Chip Memory Access Timing (RAM, ROM):** On-chip memory is accessed in two states. The data bus is 16 bits wide, permitting either byte access or word access.

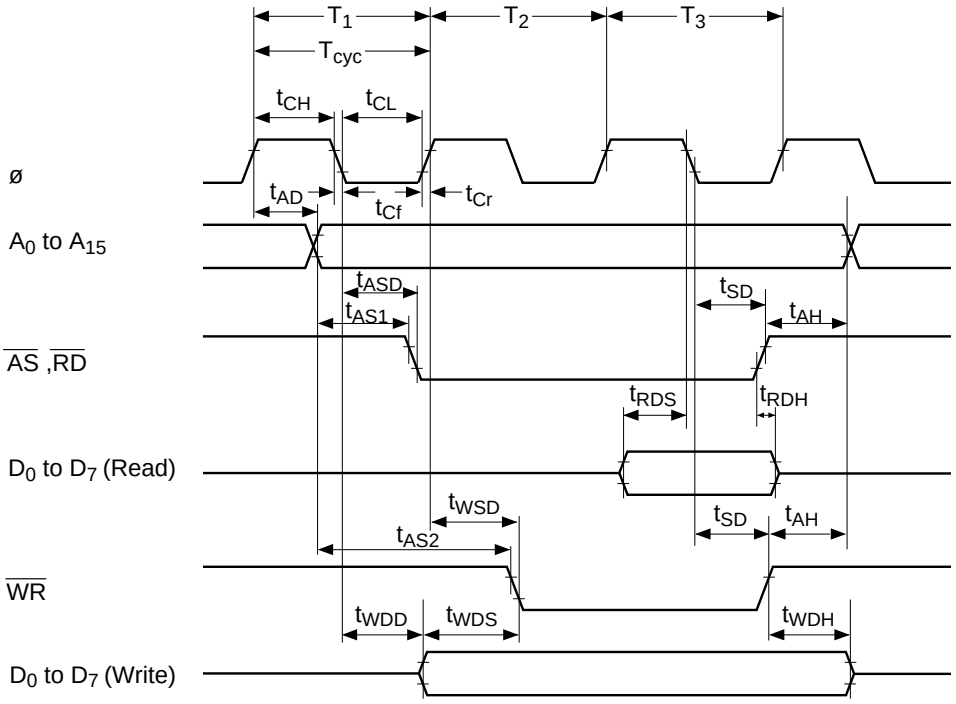



**Access Timing for On-Chip Memory**

• **Access Timing for On-Chip Register Field and External Addresses:** In both cases the access is performed in three states and the data bus is 8 bits wide. Word data and instruction codes are accessed by two consecutive byte accesses.




**Access Timing for On-Chip Register Field**




## External Address Access Cycle

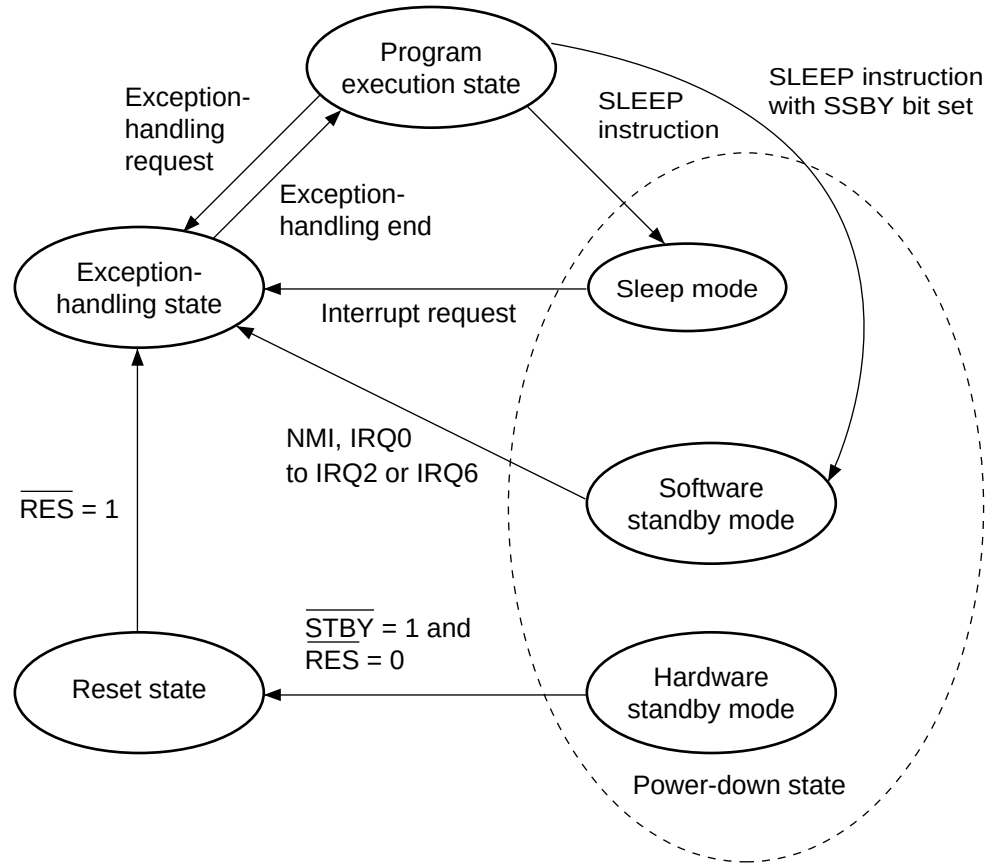
—Preliminary specifications—

| | | 5 MHz | | 10 MHz | | 16 MHz | | |
|---|---|---|---|---|---|---|---|---|
| **Item** | **Symbol** | **min.** | **max.** | **min.** | **max.** | **min.** | **max.** | **Unit** |
| Clock cycle time | $t_{CYC}$ | 200 | 2000 | 100 | 2000 | 62.5 | 2000 | ns |
| Clock pulse width low | $t_{CL}$ | 70 | — | TBD | — | TBD | — | ns |
| Clock pulse width high | $t_{CH}$ | 70 | — | TBD | — | TBD | — | ns |
| Clock rise time | $t_{Cr}$ | — | 25 | — | TBD | — | TBD | ns |
| Clock fall time | $t_{Cf}$ | — | 25 | — | TBD | — | TBD | ns |
| Address delay time | $t_{AD}$ | — | 90 | — | TBD | — | TBD | ns |
| Address hold time | $t_{AH}$ | 30 | — | TBD | — | TBD | — | ns |
| Address strobe delay time | $t_{ASD}$ | — | 80 | — | TBD | — | TBD | ns |
| Write strobe delay time | $t_{WSD}$ | — | 80 | — | TBD | — | TBD | ns |
| Strobe delay time | $t_{SD}$ | — | 90 | — | TBD | — | TBD | ns |
| Address setup time 1 | $t_{AS1}$ | 25 | — | TBD | — | TBD | — | ns |
| Address setup time 2 | $t_{AS2}$ | 105 | — | TBD | — | TBD | — | ns |
| Read data setup time | $t_{RDS}$ | 90 | — | TBD | — | TBD | — | ns |
| Read data hold time | $t_{RDH}$ | 0 | — | TBD | — | TBD | — | ns |
| Write data delay time | $t_{WDD}$ | — | 125 | — | TBD | — | TBD | ns |
| Write data setup time | $t_{WDS}$ | 10 | — | TBD | — | TBD | — | ns |
| Write data hold time | $t_{WDH}$ | 30 | — | TBD | — | TBD | — | ns |

Values other than $t_{CYC}$ are to be determined.

## 4.7 Operating States

The CPU operates in four states: the program execution state, exception-handling state, reset state, and power-down state. The following diagram shows the state transitions.




Notes:

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

**State Transitions**

- **Reset State:** In this state the CPU is reset.

- **Program Execution State:** In this state the CPU executes instructions in normal sequence.

- **Exception-Handling State:** This is a transient state in which the CPU executes a hardware sequence preparatory to handling a reset or interrupt. For an interrupt, the program counter and condition code register are saved to the location indicated by the stack pointer.

- **Power-Down State:** This state comprises three modes: the sleep mode, software standby mode, and hardware standby mode. The CPU halts to conserve power. In the standby modes the clock also stops. See section 7, Power-Down Modes, for details.

## 4.8 Exception Handling

There are two types of exceptions: interrupts and the reset. When an exception occurs, the CPU reads the stack pointer and pushes the program counter and condition code register onto the stack (unless the exception is a reset), then sets the interrupt mask bit in the CCR to 1 and fetches a restarting address from the exception vector table.

The reset exception has the highest priority. The interrupts have fixed priorities (see the interrupt vector table) which determines the order in which they are handled when two or more interrupts are requested simultaneously.

| Priority | Exception Type | When Detected | How Requested |
|---|---|---|---|
| 1 | Reset | Each clock period | When RES changes from low to high, or a watchdog timer reset ends |
| 2 | Interrupt | At end of instruction execution* | Requested by 9 external and 26 on-chip interrupt sources |

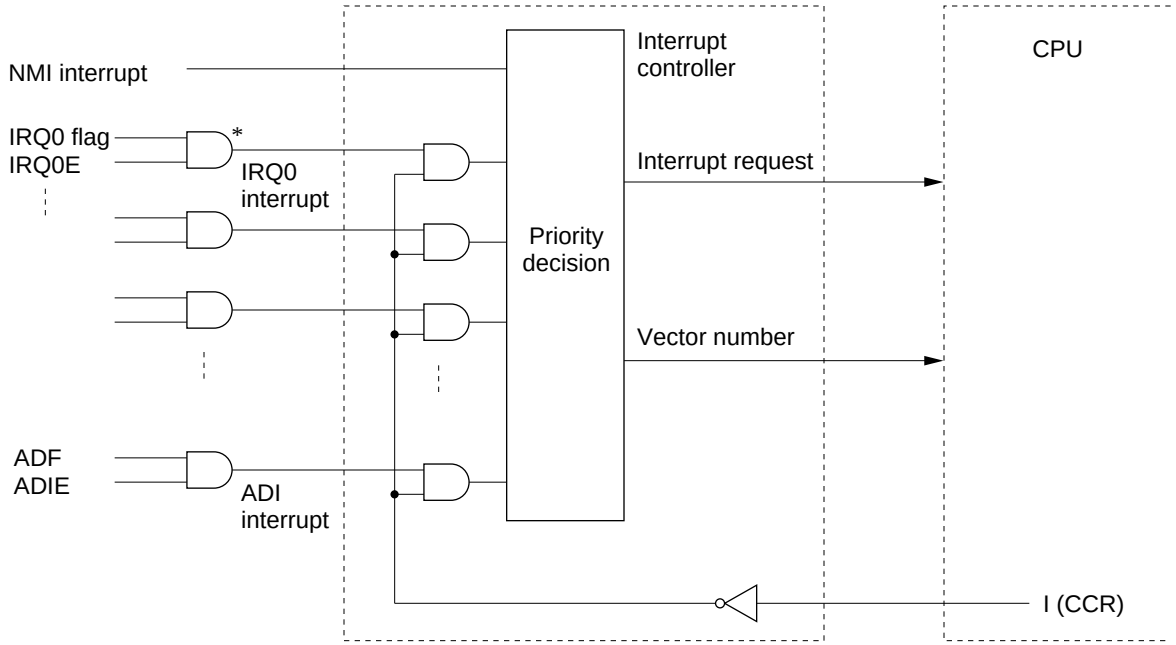
* Not detected after the ANDC, ORC, XORC, and LDC instructions.

## 4.9 Interrupts

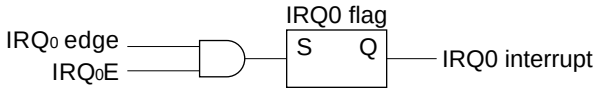
There are 35 sources of interrupts: 9 requested by inputs at the external interrupt pins ($\overline{\text{NMI}}$, $\overline{\text{IRQ}_0}$ to $\overline{\text{IRQ}_7}$, $\overline{\text{KEYIN}_0}$ to $\overline{\text{KEYIN}_{15}}$), and 26 requested from the on-chip supporting modules. NMI has the highest priority and is always accepted. The other external interrupts and the internal interrupts can be masked by the I bit in the CCR, which masks all interrupts except NMI. All interrupts are individually vectored.

- **Interrupt Controller**

Interrupts are controlled by an on-chip interrupt controller. When two or more interrupts are requested at the same time, the interrupt controller selects the one with the highest priority and leaves the others pending. When interrupted, the CPU stores the program counter and CCR contents at the location indicated by the stack pointer, then fetches the address of the interrupt-handling routine from the vector table and begins executing the interrupt-handling routine.




Note: * For edge-sensed interrupts, these AND gates change to the circuit shown below.




**Interrupt Controller Block Diagram**

• **Interrupt Vector Table**

| Interrupt | | Source | Vector No. | Address of Vector | Priority |
|---|---|---|---|---|---|
| NMI | | External signals | 3 | H'0006–07 | High |
| IRQ0 | | | 4 | H'0008–09 | |
| IRQ1 | | | 5 | H'000A–0B | |
| IRQ2 | | | 6 | H'000C–0D | |
| IRQ3 | | | 7 | H'000E–0F | |
| IRQ4 | | | 8 | H'0010–11 | |
| IRQ5 | | | 9 | H'0012–13 | |
| IRQ6 | (include key-sense interrupt) | | 10 | H'0014–15 | |
| IRQ7 | | | 11 | H'0016–17 | |
| ICIA | (input capture A) | 16-bit | 12 | H'0018–19 | |
| ICIB | (input capture B) | free-running | 13 | H'001A–1B | |
| ICIC | (input capture C) | timer | 14 | H'001C–1D | |
| ICID | (input capture D) | | 15 | H'001E–1F | |
| OCIA | (output compare A) | | 16 | H'0020–21 | |
| OCIB | (output compare B) | | 17 | H'0022–23 | |
| FOVI | (overflow) | | 18 | H'0024–25 | |
| CMI0A | (compare-match A) | 8-bit timer 0 | 19 | H'0026–27 | |
| CMI0B | (compare-match B) | | 20 | H'0028–29 | |
| OVI0 | (overflow) | | 21 | H'002A–2B | |
| CMI1A | (compare-match A) | 8-bit timer 1 | 22 | H'002C–2D | |
| CMI1B | (compare-match B) | | 23 | H'002E–2F | |
| OVI1 | (overflow) | | 24 | H'0030–31 | |
| IBF1 | (IDR1 receive-end) | Host interface | 25 | H'0032–H'0033 | |
| IBF2 | (IDR2 receive-end) | | 26 | H'0034–H'0035 | |
| ERI0 | (receive error) | Serial | 27 | H'0036–37 | |
| RXI0 | (receive-end) | communication | 28 | H'0038–39 | |
| TXI0 | (TDR empty) | interface 0 | 29 | H'003A–3B | |
| TEI0 | (TSR empty) | | 30 | H'003C–3D | |
| ERI1 | (receive error) | Serial | 31 | H'003E–3F | |
| RXI1 | (receive-end) | communication | 32 | H'0040–41 | |
| TXI1 | (TDR empty) | interface 1 | 33 | H'0042–43 | |
| TEI1 | (TSR empty) | | 34 | H'0044–45 | |
| ADI | (end of A/D conversion) | A/D converter | 35 | H'0046–47 | |
| WOVF | (watchdog timer overflow) | Watchdog timer | 36 | H'0048–H'0049 | |
| Reserved | | | 37 | H'004A–H'004B | Low |

Notes: 1. H'0000 – H'0001 is the reset vector.
2. H'0002 to H'0005 are reserved for system use and are not available to the user.

# Section 5. Operating Modes

There are three operating modes. The mode is selected by the input at the mode pins ($MD_1$ and $MD_0$).

## 5.1 Mode 1 (Expanded Mode with On-Chip ROM Disabled)

This mode permits access to external memory and peripheral devices. Ports 1 and 2 are used as the address bus and port 3 as the data bus. Port 9 is partly used for control signals.

The on-chip ROM is disabled. The corresponding addresses are mapped onto external memory.

The total address space, including on-chip and external addresses, is 64 kbytes.

## 5.2 Mode 2 (Expanded Mode with On-Chip ROM Enabled)

Like mode 1, this mode permits access to external memory and peripheral devices, but when the chip comes out of reset, ports 1 and 2 are assigned as input ports. Software can select which pins of these ports to use for address output by setting bits in their data direction registers. Port 3 is used as the data bus, and port 9 is partly used for control signals. The on-chip ROM is enabled.

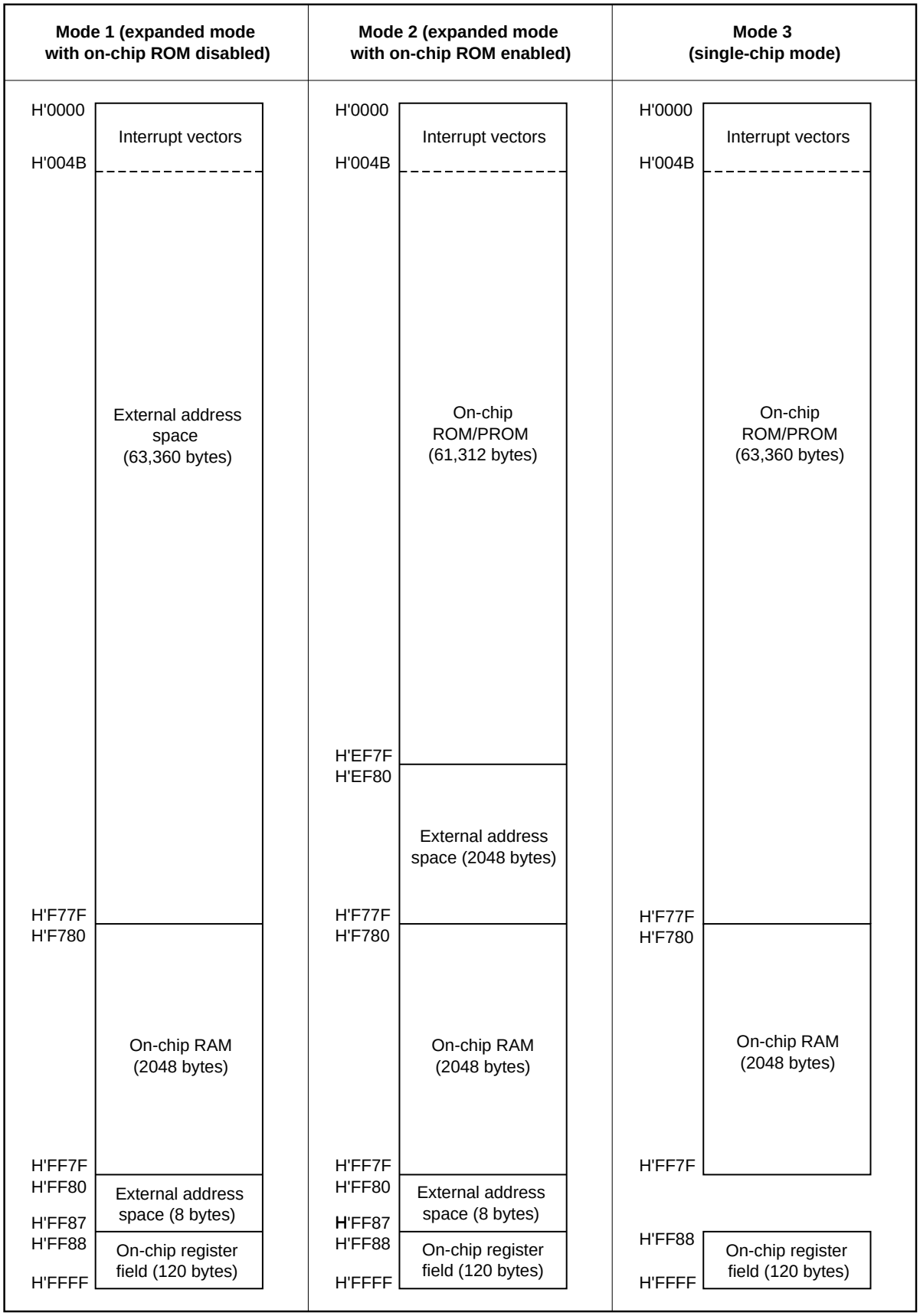
The total address space, including on-chip and external addresses, is 64 kbytes.

## 5.3 Mode 3 (Single-Chip Mode)

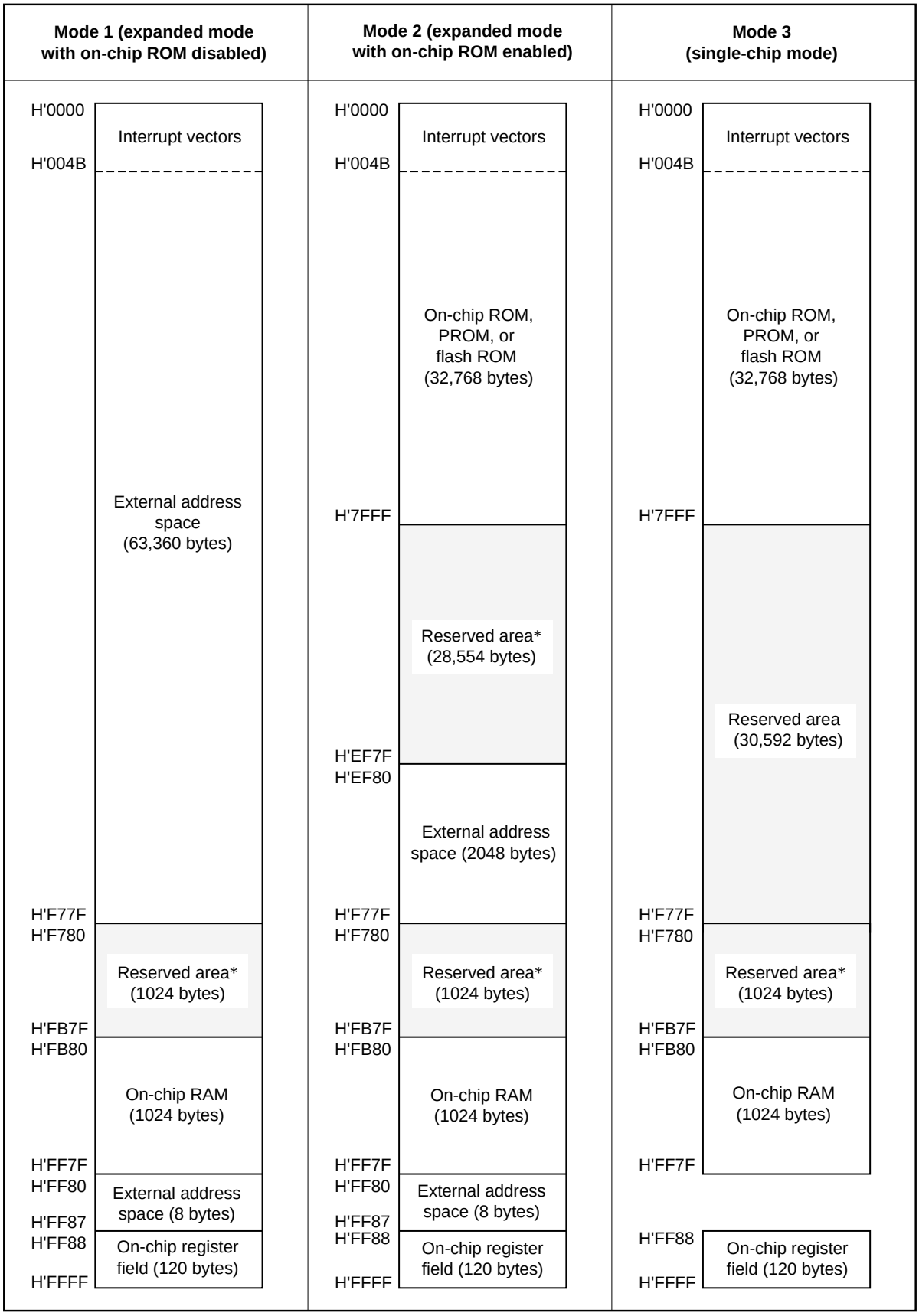
In this mode the external address space cannot be used. The chip operates with only its on-chip ROM and RAM and on-chip register field. All I/O ports are available for general-purpose input and output.

• **Memory Maps in Each Mode**

(1) H8/3437



(2) H8/3434




Note: * Don't access reserved areas.

# Section 6. On-Chip Supporting Functions

## 6.1 16-Bit Free-Running Timer

The chip includes a single-channel free-running timer module (FRT). Using a 16-bit free-running counter (FRC) as a time base, the FRT can provide two independent waveform outputs, and can measure the width or period of input pulse signals.

- **Features**

- Selection of four counter clock sources
  - — Three internal clock sources ($ø_P/2$, $ø_P/8$, $ø_P/32$, where $ø_P$ is the clock for the on-chip supporting modules)
  - — External clock source

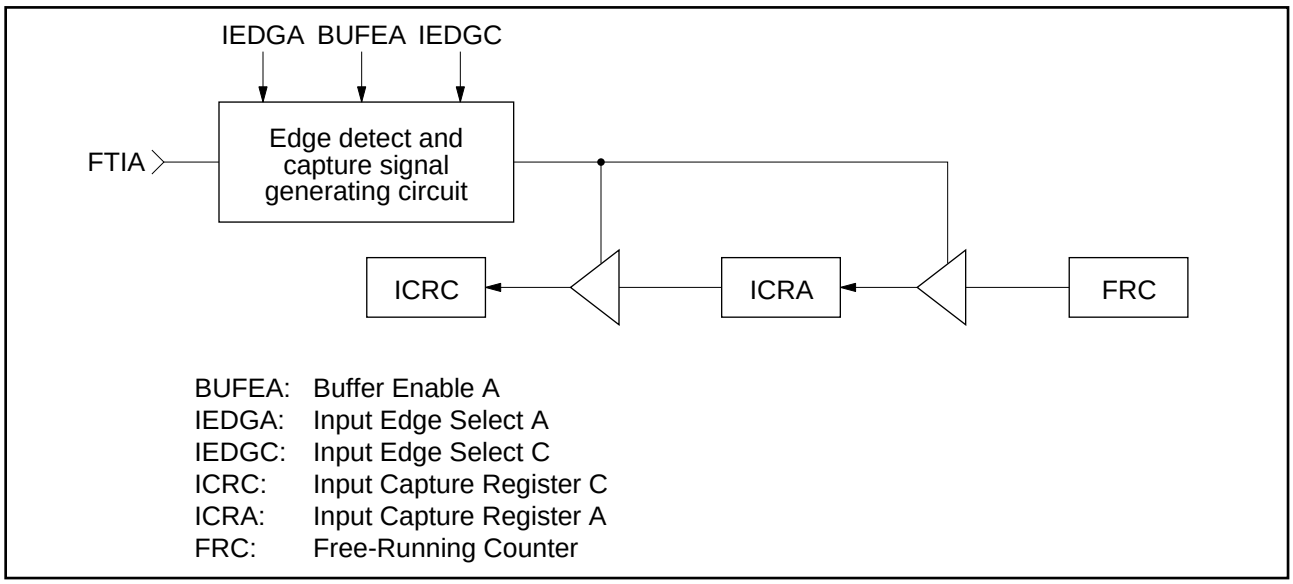
- Two independent comparators, for output of two independent waveforms

- Input capture function
  - — Selection of rising or falling edge
  - — Four independent input-capture registers
  - — Input capture can be buffered*

- Free-running counter (FRC) can be cleared by compare-match A

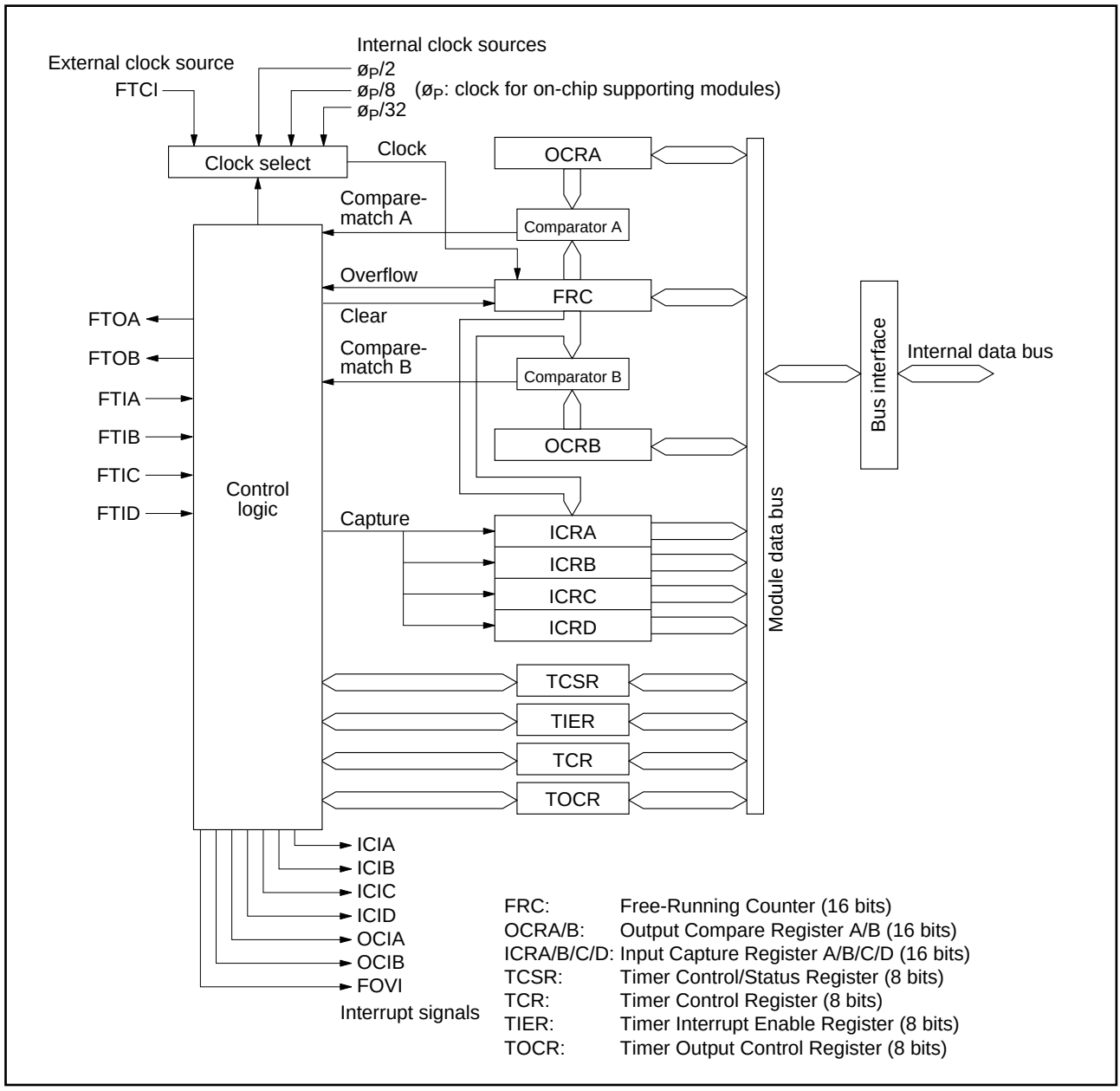
- Seven independent interrupts
  - — Two compare-match interrupts
  - — Four input capture interrupts
  - — One overflow interrupt

*: In buffer mode, the four 16-bit input capture registers are used in two pairs. In each pair, one of the registers is used as a buffer for the other.

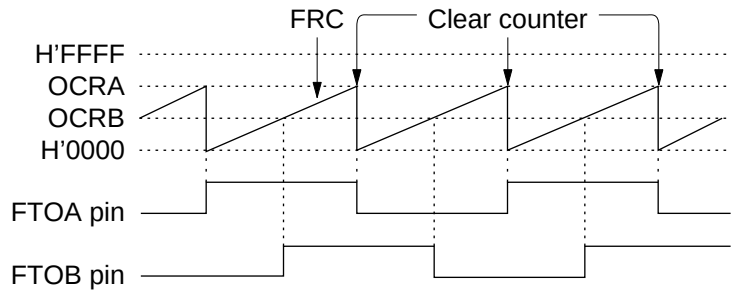



**Buffer Operation**

• **Block Diagram**




• **Example of 2-Phase Pulse Output:** An example of the output of two pulse signals with a 50% duty ratio and arbitrary phase difference is shown below.




Note: The counter clear A bit in the TCSR is set to 1. The output level is inverted by software at each compare-match event.

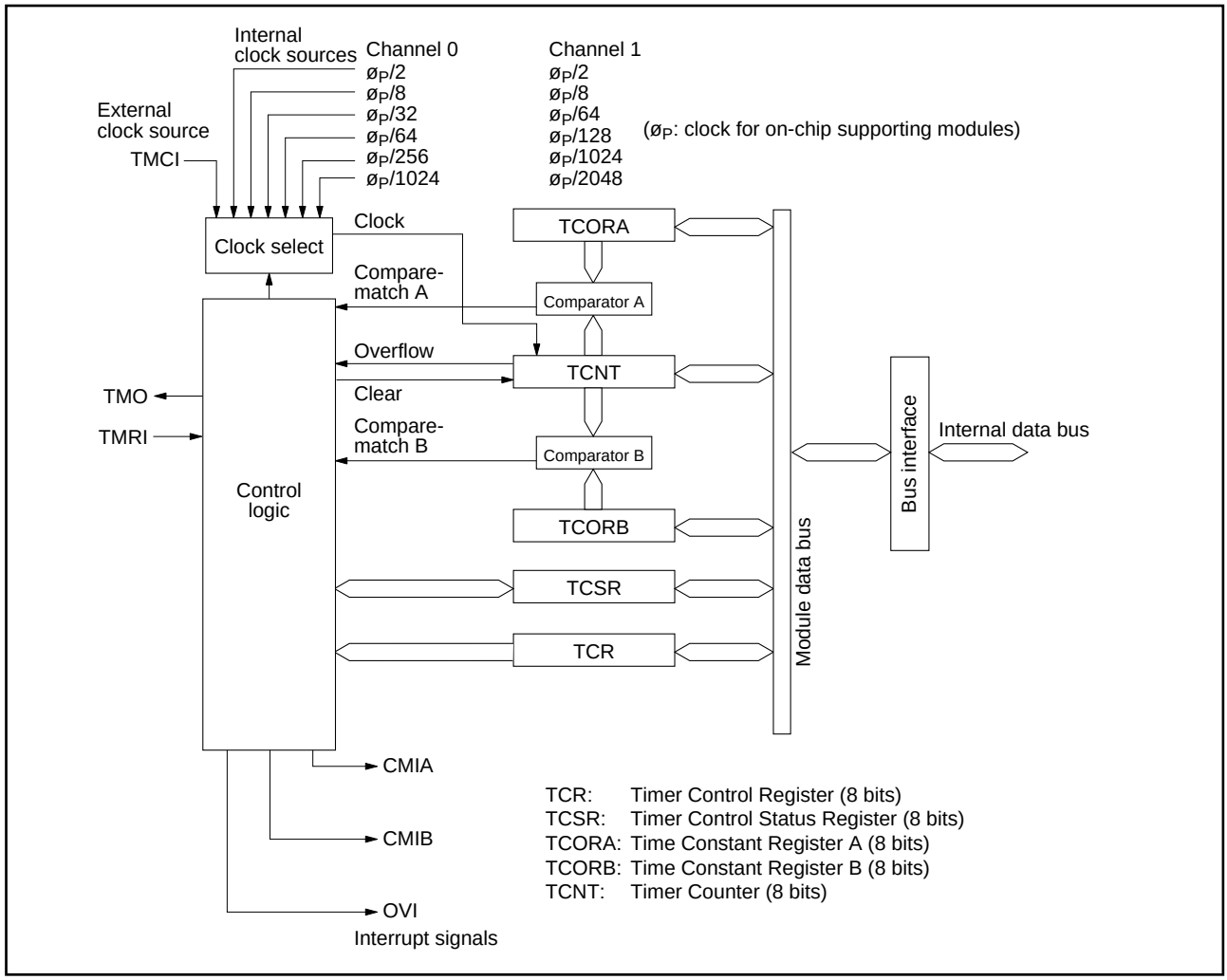
## 6.2 8-Bit Timer

The chip includes a multifunction 8-bit timer module with two independent channels. Each uses an 8-bit counter as a time base. These 8-bit timers can be employed to generate arbitrary pulse outputs.

- **Features**

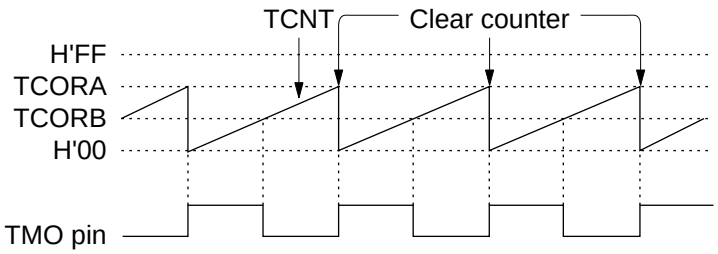
- Selection of seven counter clock sources
  - — Six internal clock sources
  - — External clock source
- Two independent comparators
  - — Two independent compare-match signals can be combined to obtain pulse output with an arbitrary duty cycle.
- The counter can be cleared by compare-match A or B, or by an external reset signal.
- Three independent interrupts
  - — Two compare-match interrupts
  - — One overflow interrupt

- **Block Diagram**




**Block Diagram of 8-Bit Timer**

• **Example of Pulse Output:** An example of the output of a pulse signal with an arbitrary duty cycle is shown below.




Note: A bit in TCR is set to have the counter cleared when it matches TCORA. TCORA determines the period and TCORB the duty cycle. The pulse signal is output without software intervention.

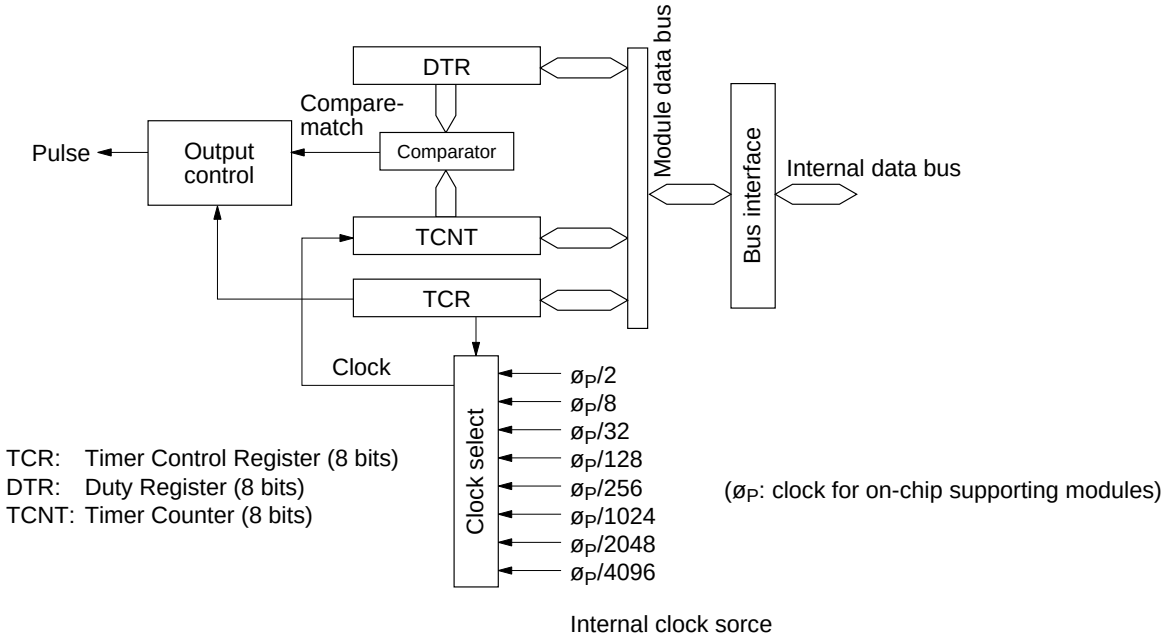
## 6.3 PWM Timer

The chip includes a PWM (Pulse Width Modulation) timer module with two independent channels. Each includes an 8-bit duty register which can be set to give pulses with any duty ratio from 0 to 100%.

• **Features**

- Selection of eight frequencies
- Resolution: 1/250
- Selection of positive or negative output logic

• **Block Diagram**




**Block Diagram of PWM Timer**

• **PWM Period and Resolution:** The PWM period is specified by setting bits 2 to 0 (CKS2 to CKS0) of the timer control register (TCR). These bits select one of eight internal clock frequencies obtained by dividing the on-chip supporting module clock frequency ($ø_P$).

| Bit 2 | Bit 1 | Bit 0 | | Description | |
|---|---|---|---|---|---|
| **CKS2** | **CKS1** | **CKS0** | **Internal Clock** | **Resolution** | **PWM Period (Frequency)** |
| 0 | 0 | 0 | ø/2 | 200 ns | 50 µs (20 kHz) |
| 0 | 0 | 1 | ø/8 | 800 ns | 200 µs (5 kHz) |
| 0 | 1 | 0 | ø/32 | 3.2 µs | 800 µs (1.25 kHz) |
| 0 | 1 | 1 | ø/128 | 12.8 µs | 3.2 ms (312.5 Hz) |
| 1 | 0 | 0 | ø/256 | 25.6 µs | 6.4 ms (156.3 Hz) |
| 1 | 0 | 1 | ø/1024 | 102.4 µs | 25.6 ms (39.1 Hz) |
| 1 | 1 | 0 | ø/2048 | 204.8 µs | 51.2 ms (19.5 Hz) |
| 1 | 1 | 1 | ø/4096 | 409.6 µs | 102.4 ms (9.8 Hz) |

Note: Values shown are for a 10-MHz on-chip supporting module clock ($ø_P$), and when $ø_P$ = ø.

## 6.4 Watchdog Timer

The watchdog timer can be used to supervise system operation. Either this watchdog function or an interval timer function can be selected.

- **Features**

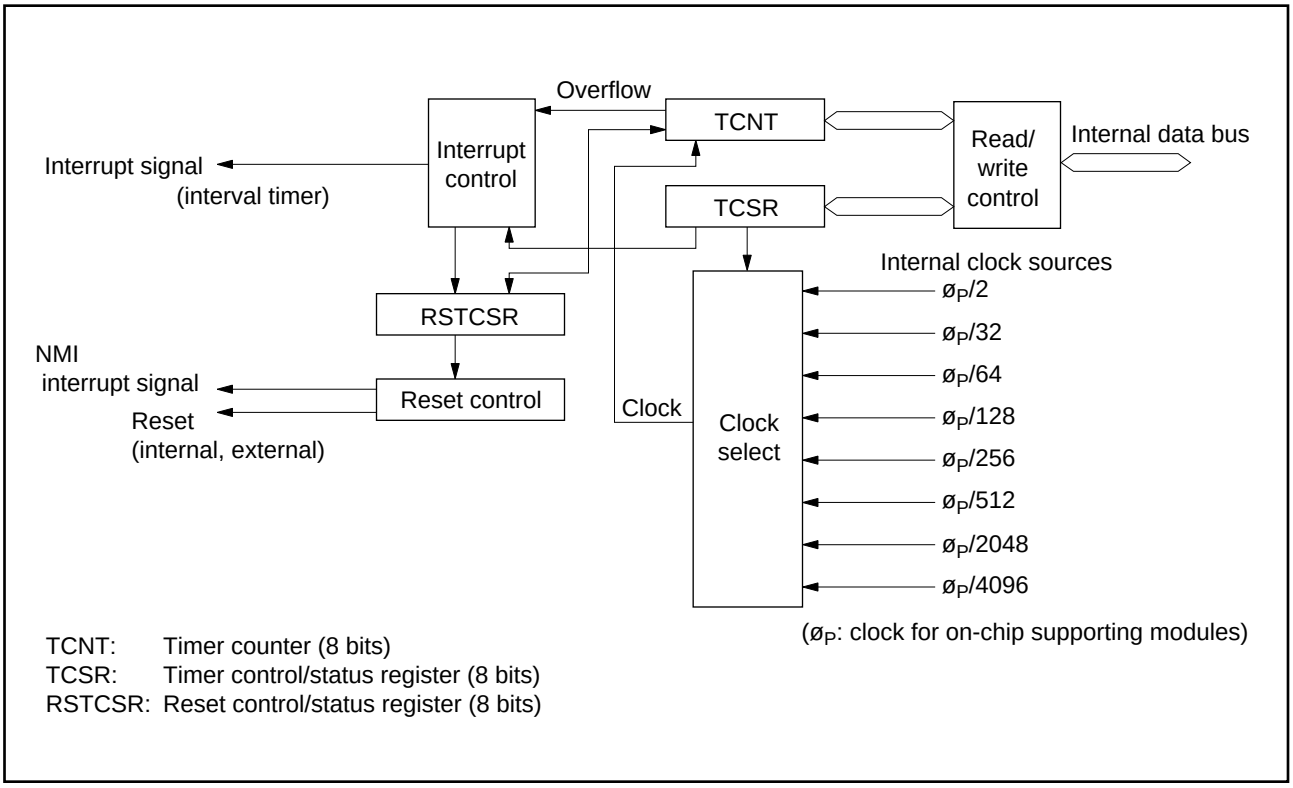
- Selection of eight counter clock sources
  — $ø_P/2$, $ø_P/32$, $ø_P/64$, $ø_P/128$, $ø_P/256$, $ø_P/512$, $ø_P/2048$, $ø_P/4096$

- Can be used as an interval timer

- Timer counter overflow can generate a reset signal or interrupt
  — A reset signal or a NMI interrupt is generated by the watchdog timer function; an interval timer interrupt is generated by the interval timer function.

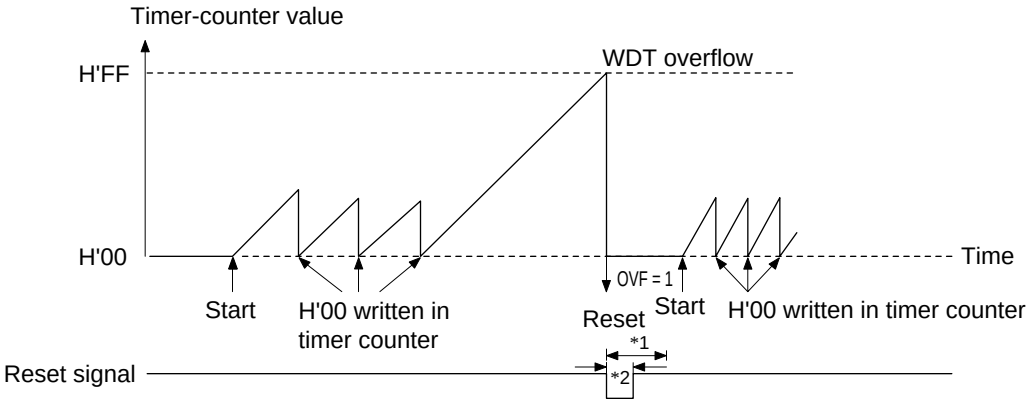
- Total internal reset by watchdog timer reset signal; external reset signal output also available
  — When a reset signal is generated by overflow of the watchdog timer counter, the entire chip is reset internally. A reset signal can simultaneously be output from the RESO pin for a total system reset.

• **Block Diagram**



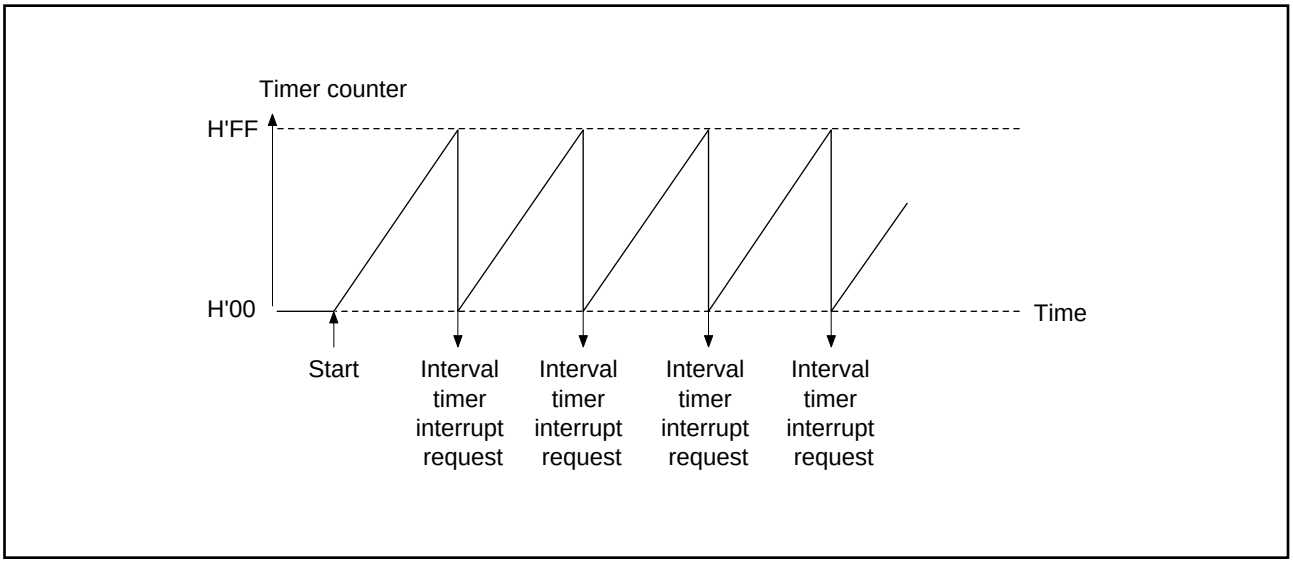

**Block Diagram of Watchdog Timer**

• **Watchdog Timer Operation:** Use of this timer as a watchdog timer is illustrated below. The timer counter (TCNT) functions as an up-counter driven by the specified clock.




Notes: 1. The internal reset generated by WDT overflow lasts 518 states.
2. The external reset signal is output for 132 states.

• **Interval Timer Operation:** Interval timer mode is illustrated below. The timer counter (TCNT) functions as an up-counter driven by the specified clock. An interval timer interrupt is generated at each TCNT overflow. This function can be used to provide interrupts at regular intervals.



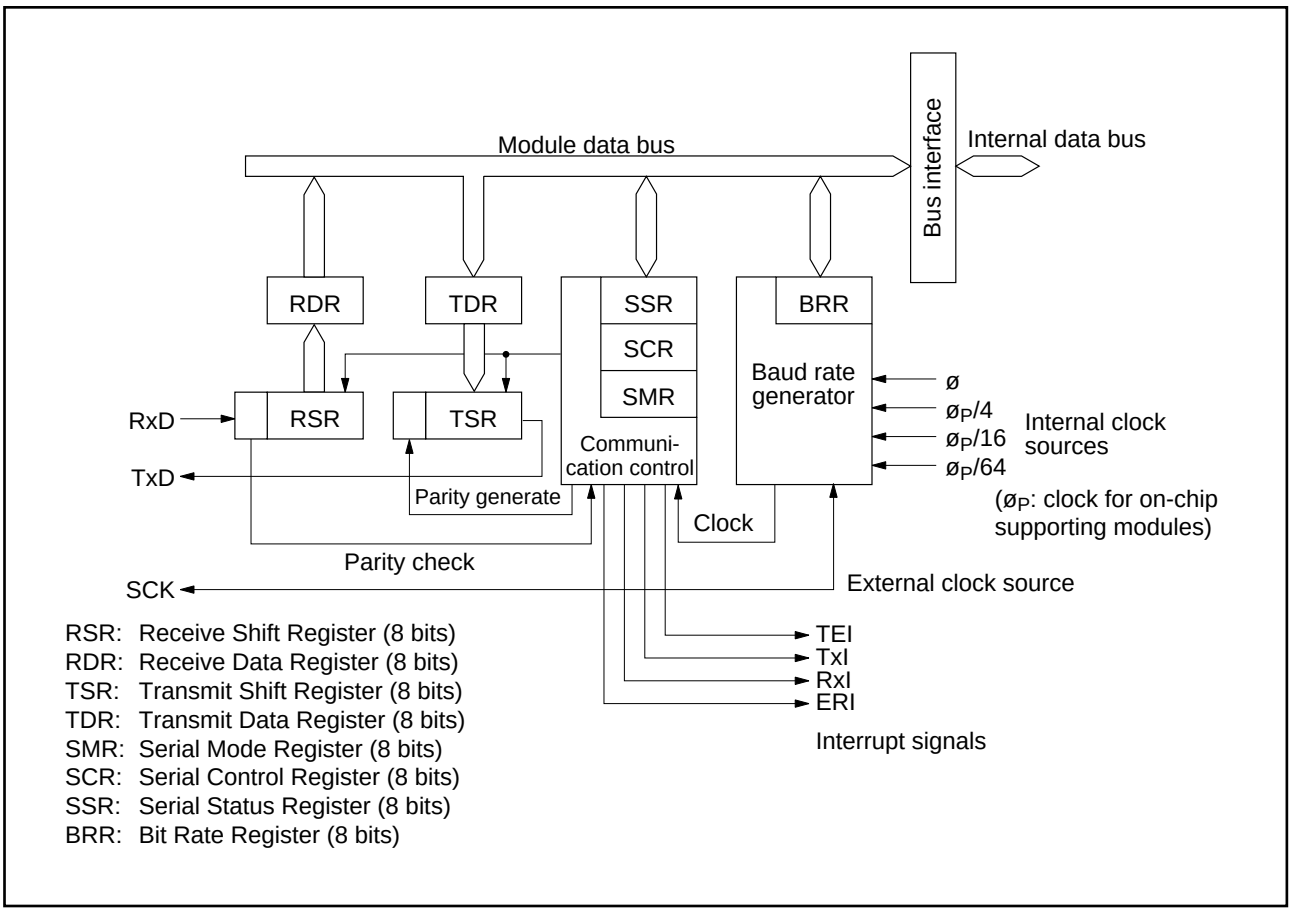

## 6.5 Serial Communication Interface

The chip includes a dual-channel serial communication interface (SCI) that can communicate with external devices by either asynchronous or clocked synchronous serial data transfer (selectable). A multiprocessor communication function supports communication among two or more processors.

• **Features**

- Supports both synchronous and asynchronous communication.
- Supports full duplex communication.
- Can send and receive data continuously, using double-buffering data registers.
- Built-in baud rate generator can generate any clock rate.
- Selectable clock source: either the built-in baud rate generator or an external clock signal (SCK pin).
- Detects overrun errors, framing errors, and parity errors.
- Four independent interrupts: TDR empty, TSR empty, receive-end, and receive error.

• **Block Diagram**



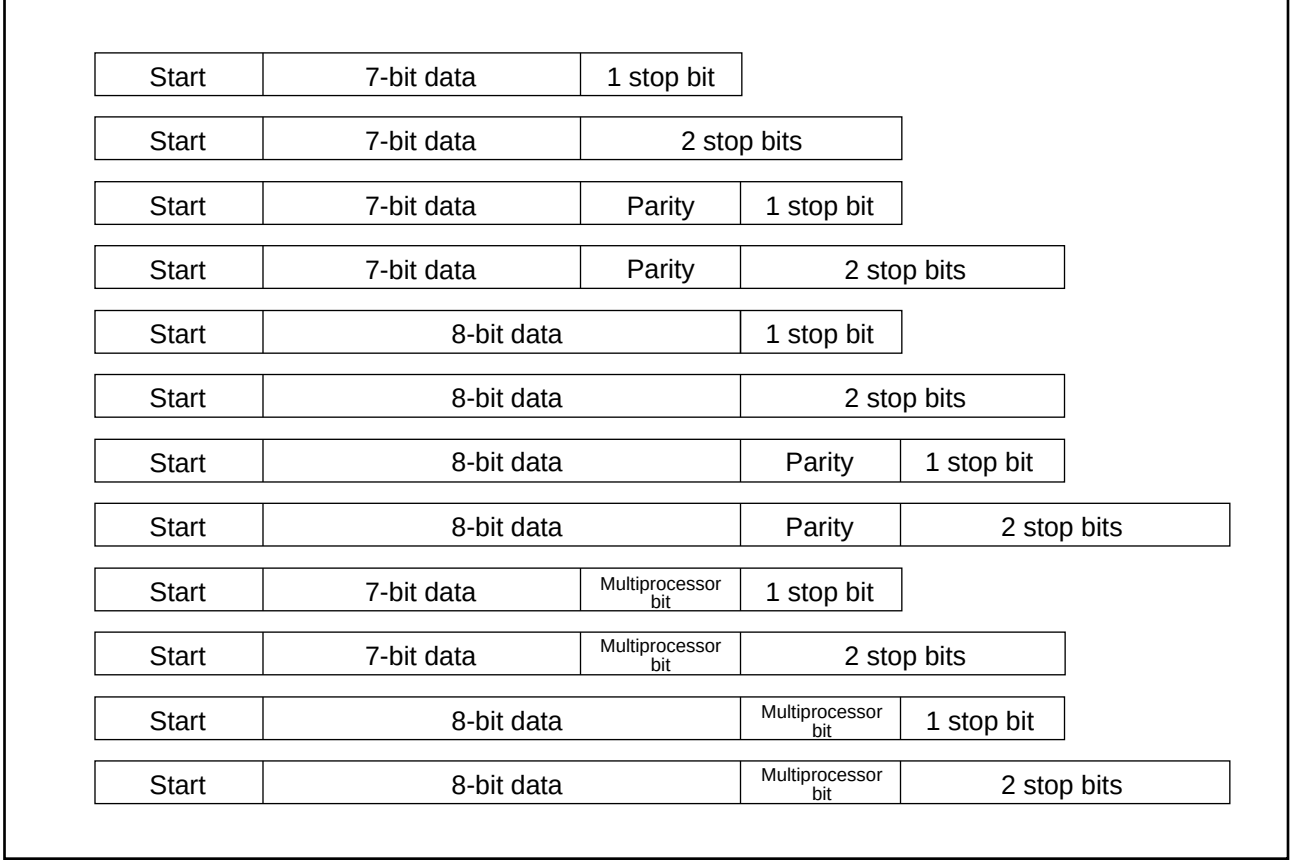

**Block Diagram of Serial Communication Interface**

• **Asynchronous Mode:** In asynchronous communication mode individual characters are synchronized by start bits and stop bits.

• Twelve transfer formats

— Data length: 7 or 8 bits
— Stop bit length: 1 or 2 bits
— Parity or multiprocessor bit: even parity, odd parity, multiprocessor bit, or no parity

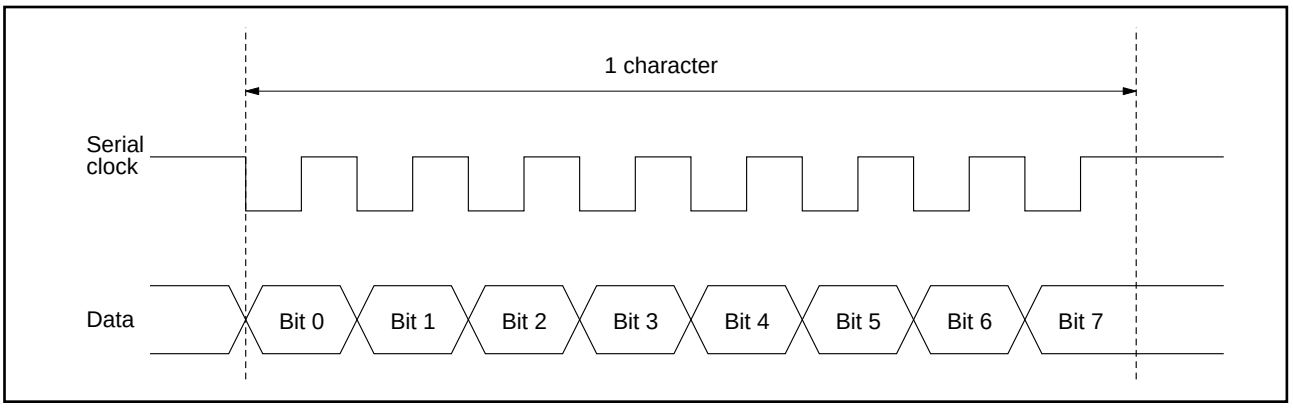
• Either the internal baud rate generator or an external clock input at the SCK pin can be selected as the clock source.




**Twelve Data Formats**

• **Clocked Synchronous Mode:** Data transfer is synchronized with a clock pulse. This mode is suitable for continuous, high-speed serial communication.

- Data length: 8 bits/character
- Detects overrun errors
- The transmit/receive clock can be provided by the on-chip baud rate generator or by an external clock input at the SCK pin.
- LSB-first: the least significant data bit is sent and received first.
- Can communicate with the HD64180 and other chips supporting a synchronous communication mode.




**Data Format in Synchronous Mode (Example)**

- **BRR Settings for Typical Bit Rates (Asynchronous Mode)**

| XTAL (MHz) | 9.8304 | | | 10 | | |
|---|---|---|---|---|---|---|
| ø (MHz) | 9.8304 | | | 10 | | |
| Bit Rate (Bits/s) | Baud Rate Generator Input Clock | BRR Value | Error (%) | Baud Rate Generator Input Clock | BRR Value | Error (%) |
| 110 | ø/16 | 174 | –0.26 | ø/64 | 43 | +0.88 |
| 150 | ø/16 | 127 | 0 | ø/16 | 129 | +0.16 |
| 300 | ø/4 | 255 | 0 | ø/16 | 64 | +0.16 |
| 600 | ø/4 | 127 | 0 | ø/4 | 129 | +0.16 |
| 1200 | ø | 255 | 0 | ø/4 | 64 | +0.16 |
| 2400 | ø | 127 | 0 | ø | 129 | +0.16 |
| 4800 | ø | 63 | 0 | ø | 64 | +0.16 |
| 9600 | ø | 31 | 0 | ø | 32 | –1.36 |
| 19200 | ø | 15 | 0 | ø | 15 | +1.73 |
| 31250 | ø | 9 | –1.7 | ø | 9 | 0 |
| 38400 | ø | 7 | 0 | ø | 7 | +1.73 |

Note: Values shown are for a 10 MHz on-chip supporting module clock ($ø_P$), and when $ø_P$ = ø. The error should preferably be 1% or less.

$$N = [OSC \div (64 \times 2^{2n} \times B)] \times 10^6 - 1$$

N: BRR value of baud rate generator; $0 \leq N \leq 255$

OSC: Frequency of crystal (MHz)

B: Bit rate (bits/s)

n: Baud rate generator input clock number; n = 0, 1, 2, 3

| CKS1 | CKS0 | n | Clock |
|---|---|---|---|
| 0 | 0 | 0 | ø |
| 0 | 1 | 1 | $ø_P$/4 |
| 1 | 0 | 2 | $ø_P$/16 |
| 1 | 1 | 3 | $ø_P$/64 |

## 6.6 Host Interface

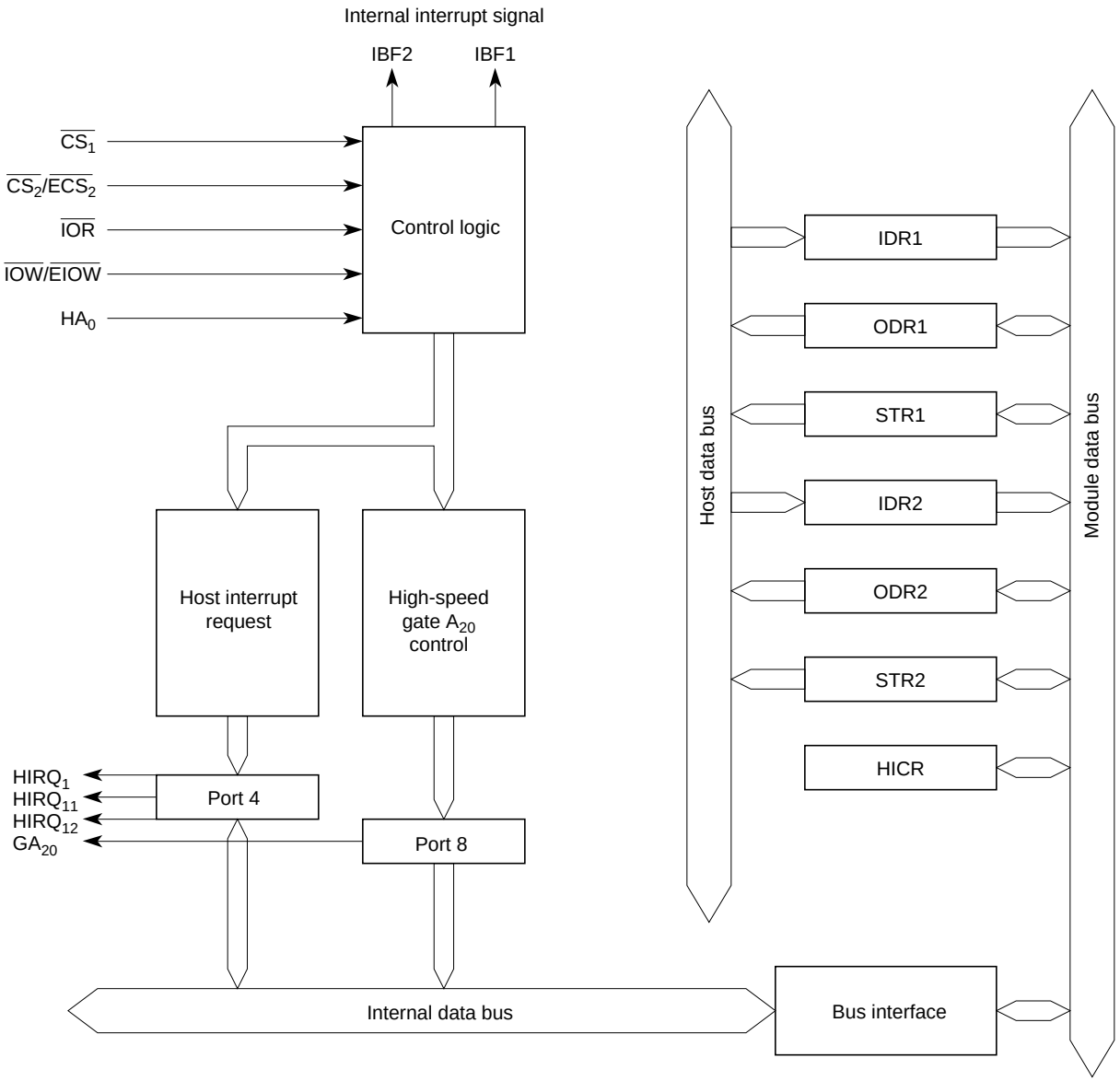
Two host interface channels are provided on-chip. The host interface is a parallel interface between the internal CPU and a host device. Other host interface facilities are provided by a high-speed gate $A_{20}$ logic circuit and a host interrupt request circuit.

Both the on-chip CPU and the host device can access the input data registers, output data registers, and status registers in the host interface as registers located in the memory address space. Interrupt requests can be sent to the on-chip CPU when the host device writes to an input register, and to the host device when the on-chip CPU writes to an output register. The host device can control the output of the high-speed gate $A_{20}$ signal by writing special data sequences to the input register.

- **Features**

- Parallel interface with 8-bit data width
- Independent operation of two host interface channels, using two chip select input signals
- Interrupt request to on-chip CPU can be generated when host device writes to input data register
- Interrupt request to host device can be generated when on-chip CPU writes to output data register
- Host device can control high-speed gate $A_{20}$ signal output by writing special data sequences to input register (turn-on/turn-off sequences)

- **Block Diagram**




<Legend>

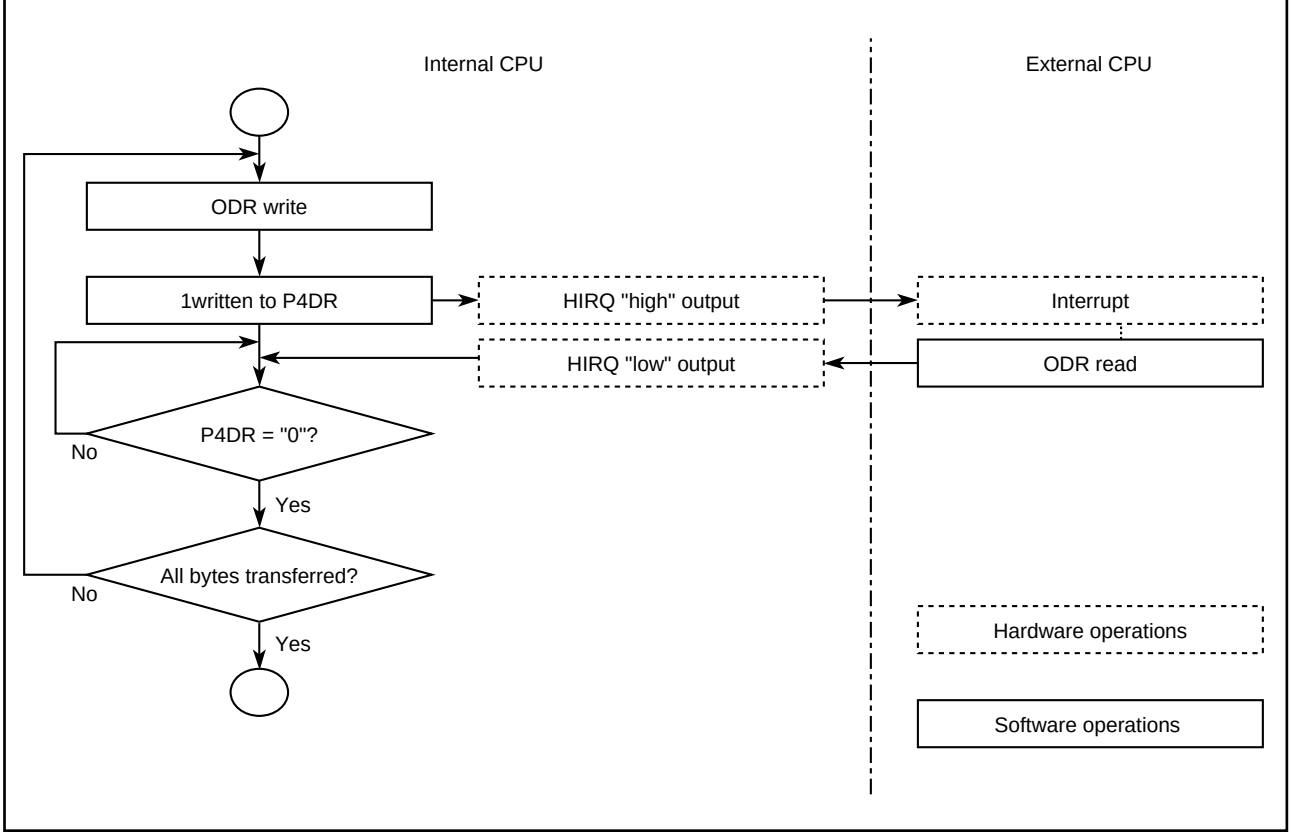
IDR1: Input data register 1 (8 bits)
IDR2: Input data register 2 (8 bits)
ODR1: Output data register 1 (8 bits)
ODR2: Output data register 2 (8 bits)
STR1: Status register 1 (8 bits)
STR2: Status register 2 (8 bits)
HICR: Host interface control register (8 bits)

- **Host Read/Host Write**

| $\overline{CS_2}$/$\overline{ECS_2}$ | $\overline{CS_1}$ | IOR | IOW/EIOW | HA0 | Operation |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | Read data from output data register 1 (ODR1) |
| | | | | 1 | Read status from status register 1 (STR1) |
| | | 1 | 0 | 0 | Write data to input data register 1 (IDR1) |
| | | | | 1 | Write command to input data register 1 (IDR1) |
| 0 | 1 | 0 | 1 | 0 | Read data from output data register 2 (ODR2) |
| | | | | 1 | Read status from status register 2 (STR2) |
| | | 1 | 0 | 0 | Write data to input data register 2 (IDR2) |
| | | | | 1 | Write command to input data register 2 (IDR2) |
| 1<br>1<br>0 | 1<br>0<br>1 | 1 | 1 | * | Idle state |
| 1 | 1 | 1<br>0<br>1 | 1<br>1<br>0 | * | |
| 0 | 0 | * | * | * | Illegal settings |
| * | * | 0 | 0 | * | |

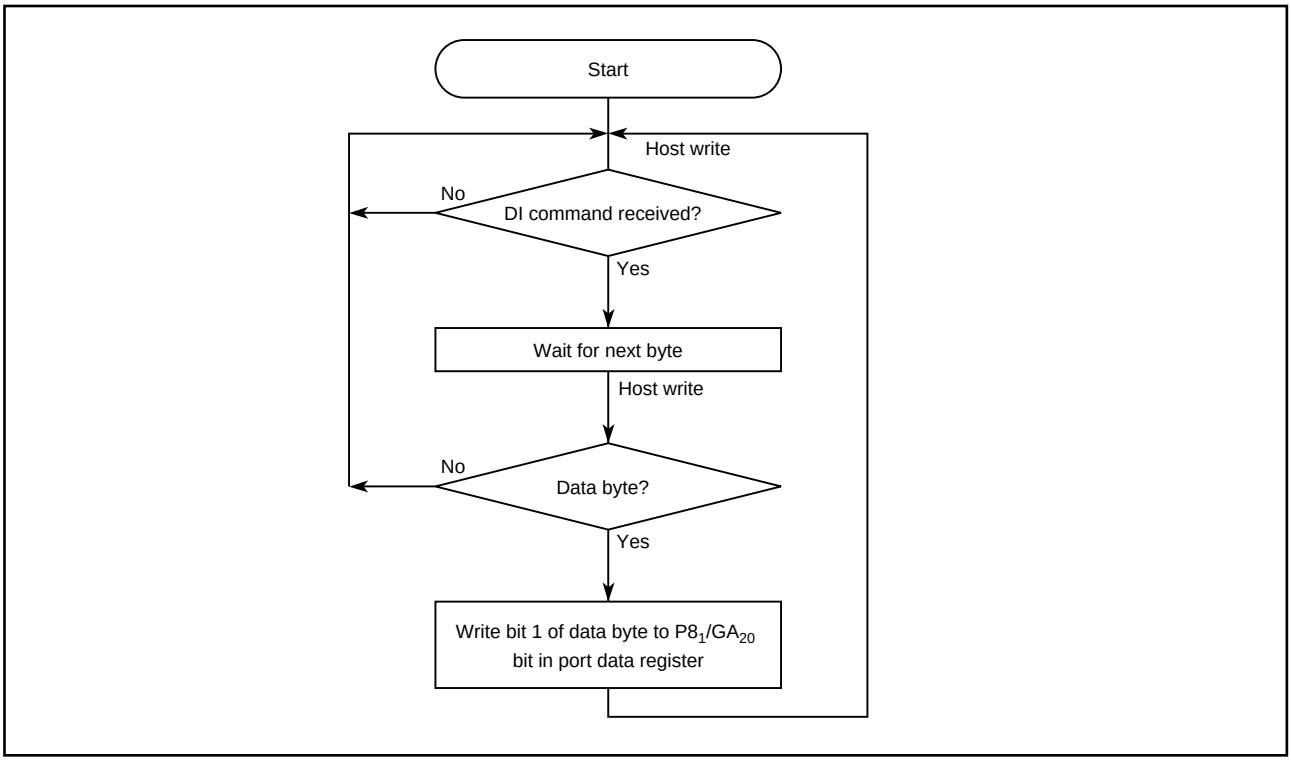
- **Host Interrupt Request Output**

| Host Interrupt Signal | Setting Conditions | Clearing Conditions |
|---|---|---|
| HIRQ11 ($P4_3$) | On-chip CPU reads bit 3 in port 4 DR as 0, then writes 1 | On-chip CPU writes 0 to bit 3 in port 4 DR, or host reads output data register 2 |
| HIRQ1 ($P4_4$) | On-chip CPU reads bit 4 in port 4 DR as 0, then writes 1 | On-chip CPU writes 0 to bit 4 in port 4 DR, or host reads output data register 1 |
| HIRQ12 ($P4_5$) | On-chip CPU reads bit 5 in port 4 DR as 0, then writes 1 | On-chip CPU writes 0 to bit 5 in port 4 DR, or host reads output data register 1 |


Internal CPU
External CPU
ODR write
1written to P4DR
HIRQ "high" output
Interrupt
HIRQ "low" output
ODR read
P4DR = "0"?
No
Yes
All bytes transferred?
No
Yes
Hardware operations
Software operations

- **High-Speed Gate $A_{20}$ Output**

| High-Speed Gate $A_{20}$ Signal | Setting Conditions | Clearing Conditions |
|---|---|---|
| $GA_{20}$ ($P8_1$) | Host writes D1 command followed by data with bit 1 set to 1 | Host writes D1 command followed by data with bit 1 cleared to 0 |




| HA0 | Data/Command | On-Chip CPU Interrupt Flag | $GA_{20}$ ($P8_1$) | Remarks |
|---|---|---|---|---|
| 1 | D1 command | 0 | $Q_n$ | Turn-on sequence |
| 0 | 1 data*1 | 0 | 1 | |
| 1 | FF command | 0 | $Q_{n+1}$ (1) | |
| 1 | D1 command | 0 | $Q_n$ | Turn-off sequence |
| 0 | 0 data*2 | 0 | 0 | |
| 1 | FF command | 0 | $Q_{n+1}$ (0) | |
| 1 | D1 command | 0 | $Q_n$ | Turn-on sequence |
| 0 | 1 data*1 | 0 | 1 | (short form) |
| 1/0 | Command other than FF or D1 | 1 | $Q_{n+1}$ (1) | |
| 1 | D1 command | 0 | $Q_n$ | Turn-off sequence |
| 0 | 0 data*2 | 0 | 0 | (short form) |
| 1/0 | Command other than FF or D1 | 1 | $Q_{n+1}$ (0) | |
| 1 | D1 command | 0 | $Q_n$ | Cancels sequence |
| 1 | Command other than D1 | 1 | $Q_n$ | |
| 1 | D1 command | 0 | $Q_n$ | Retriggers sequence |
| 1 | D1 command | 0 | $Q_n$ | |
| 1 | D1 command | 0 | $Q_n$ | Executes sequence |
| 0 | Any data | 0 | 1/0 | continuously |
| 1 | D1 command | 0 | $Q_{n+1}$ (1/0) | |

Notes: 1. Any data with bit 1 set to 1.
2. Any data with bit 1 cleared to 0.

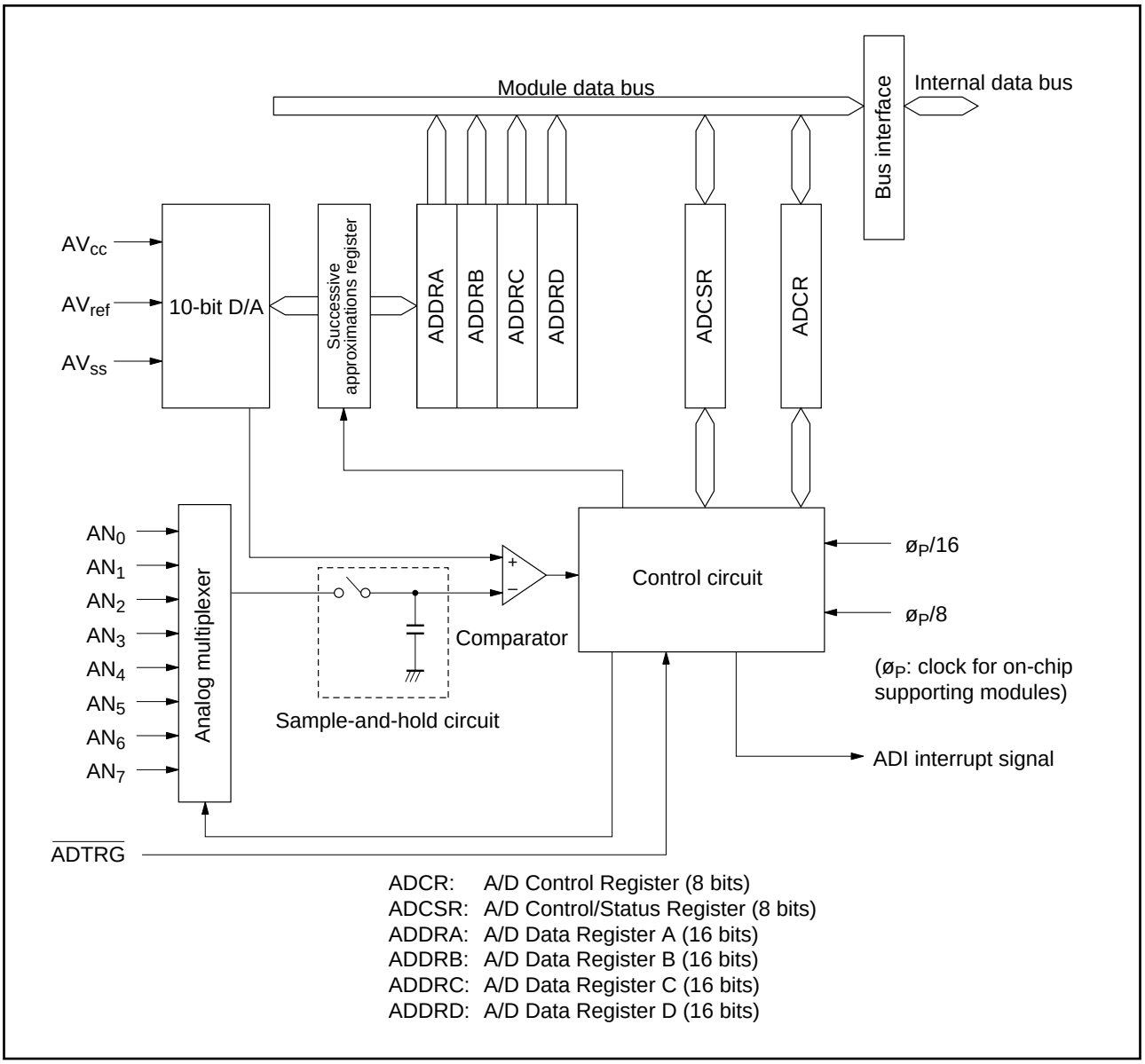
## 6.7 A/D Converter

One of the on-chip supporting modules is a 10-bit analog-to-digital converter which can be programmed for input of up to eight analog signal channels.

- **Features**

- Ten-bit resolution (input range: $AV_{SS}$ to $AV_{ref}$)
- Eight analog input pins
- High speed: minimum conversion time is 8.375 µs (at 16 MHz) or 13.4 µs per channel (at 10 MHz)
- Selectable mode
  - — Single mode: A/D conversion of one channel
  - — Scan mode: Scanning of a group of channels
- Four data registers. A/D-converted results are transferred to data registers corresponding to each channel and stored.
- A/D conversion can be started by an external trigger signal
- Generates a CPU interrupt (ADI: A/D conversion end) at the completion of A/D conversion.

- **Block Diagram**




**Block Diagram of A/D Converter**

• **A/D Operation:** Analog-to-digital conversion is performed by successive approximations, with 10-bit resolution. The input channels are selected by the channel select bits (CH2 to CH0) in ADCSR.

| Bit 2 | Bit 1 | Bit 0 | Channel(s) Selected | |
|---|---|---|---|---|
| **CH2** | **CH1** | **CH0** | **Single Mode** | **Scan Mode** |
| 0 | 0 | 0 | AN0 | AN0 |
| | | 1 | AN1 | AN0 and AN1 |
| | 1 | 0 | AN2 | AN0 to AN2 |
| | | 1 | AN3 | AN0 to AN3 |
| 1 | 0 | 0 | AN4 | AN4 |
| | | 1 | AN5 | AN4 and AN5 |
| | 1 | 0 | AN6 | AN4 to AN6 |
| | | 1 | AN7 | AN4 to AN7 |

**Single Mode:** A/D conversion starts when the ADST bit in ADCSR is set to 1. When conversion is completed, the completion flag (ADF) is set. If the interrupt enable bit (ADIE) is also set, an A/D conversion end interrupt (ADI) is requested so that an interrupt-handling routine can process the converted result.

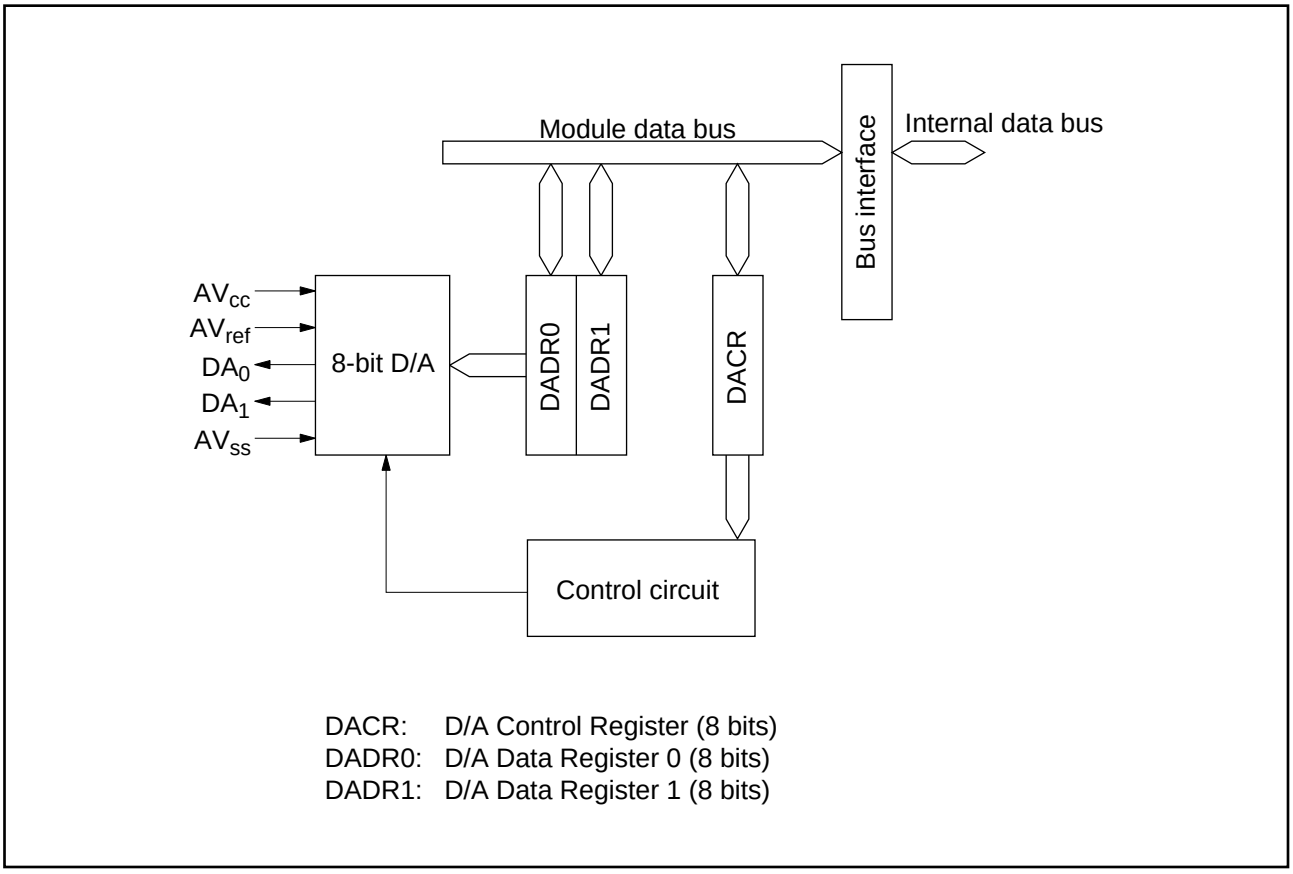
**Scan Mode:** This mode can be used to monitor analog inputs on one to four channels. When the ADST bit is set to 1, A/D conversion starts on the first selected channel. As soon as conversion of the first channel is completed, conversion of the next channel begins. When all of the specified channels have been converted, the converter starts over again from the first channel. A/D conversion continues in this way until the ADST bit is cleared. The converted results for each channel are transferred to and stored in the data registers ADDRA to ADDRD.

The ADST bit can be set to 1 by software, or by the A/D external trigger signal ($\overline{\text{ADTRG}}$).

## 6.8 D/A Converter

The chip also includes an 8-bit D/A converter. Analog signal output can be programmed on two channels.

- **Features**
- Resolution: 8 bits
- Two analog output pins
- Maximum conversion time: 10 µs
- Output voltage: 0 V to $AV_{ref}$
- **Block Diagram**




• **D/A Operation:** The D/A converter is enabled when the D/A enable bit is set to “1”. When enabled, the DADR contents are constantly converted and output. The output value is (DADR contents/256) × $AV_{ref}$.

## 6.9 I/O Ports

The I/O ports include 74 input/output pins and 8 input-only pins. The direction of the input/output pins is specified individually by setting or clearing a bit in a data direction register (DDR) for each port.

- **Port Functions**

| | | | Expanded Modes | | Single-Chip Mode |
|---|---|---|---|---|---|
| **Port** | **Description** | **Pins** | **Mode 1** | **Mode 2** | **Mode 3** |
| Port 1 | • 8-bit input-output port<br>• Can drive LEDs<br>• Input pull-ups | $P1_0$–$P1_7$/$A_0$–$A_7$ | Low address ($A_0$–$A_7$) | General input or low address ($A_0$–$A_7$) | General input/output |
| Port 2 | • 8-bit input-output port<br>• Can drive LEDs<br>• Input pull-ups | $P2_0$–$P2_7$/$A_8$–$A_{15}$ | High address ($A_8$–$A_{15}$) | General input or high address ($A_8$–$A_{15}$) | |
| Port 3 | • 8-bit input-output port<br>• Input pull-ups | $P3_0$–$P3_7$/<br>$HDB_0$–$HDB_7$/<br>$D_0$–$D_7$ | Data bus ($D_0$–$D_7$) | | HIF data bus ($HDB_0$–$HDB_7$) or general input/output |
| Port 4 | • 8-bit input-output port | $P4_0$/$TMCI_0$<br>$P4_1$/$TMO_0$<br>$P4_2$/$TMRI_0$ | General input/output or 8-bit timer 0 input/output ($TMCI_0$, $TMO_0$, $TMRI_0$) | | |
| | | $P4_3$/$TMCI_1$/$HIRQ_{11}$<br>$P4_4$/$TMO_1$/$HIRQ_1$<br>$P4_5$/$TMRI_1$/$HIRQ_{12}$ | General input/output, 8-bit timer 1 input/output ($TMCI_1$, $TMO_1$, $TMRI_1$), or HIF host CPU interrupt request output ($HIRQ_{11}$, $HIRQ_1$, $HIRQ_{12}$) | | |
| | | $P4_6$/$PW_0$<br>$P4_7$/$PW_1$ | General input/output or PWM timer 0/1 output ($PW_0$, $PW_1$) | | |
| Port 5 | • 3-bit input-output port | $P5_0$/$TxD_0$<br>$P5_1$/$RxD_0$<br>$P5_2$/$SCK_0$ | General input/output or serial communication interface 0 input/output ($TxD_0$, $RxD_0$, $SCK_0$) | | |
| Port 6 | • 8-bit input-output port<br>• Input pull-ups<br>• Key-sense interrupts | $P6_0$/$\overline{KEYIN_0}$/FTCI<br>$P6_1$/$\overline{KEYIN_1}$/FTOA<br>$P6_2$/$\overline{KEYIN_2}$/FTIA<br>$P6_3$/$\overline{KEYIN_3}$/FTIB<br>$P6_4$/$\overline{KEYIN_4}$/FTIC<br>$P6_5$/$\overline{KEYIN_5}$/FTID<br>$P6_6$/$\overline{KEYIN_6}$/<br>FTOB/$\overline{IRQ_6}$<br>$P6_7$/$\overline{KEYIN_7}$/$\overline{IRQ_7}$ | General input/output, 16-bit free-running timer input/output (FTCI, FTOA, FTIA, FTIB, FTIC, FTID, FTOB), key-sense interrupts ($\overline{KEYIN_0}$–$\overline{KEYIN_7}$, or external interrupt input ($\overline{IRQ_6}$, $\overline{IRQ_7}$) | | |
| Port 7 | • 8-bit input port | $P7_0$–$P7_5$/<br>$AN_0$–$AN_5$ | General input or analog input ($AN_0$–$AN_5$) to A/D converter | | |
| | | $P7_6$/$AN_6$/$DA_0$<br>$P7_7$/$AN_7$/$DA_1$ | General input, analog input ($AN_6$, $AN_7$) to A/D converter, or analog output ($DA_0$, $DA_1$) from D/A converter | | |
| Port 8 | • 7-bit input-output port<br>• Can drive bus buffer ($P8_6$) | $P8_0$/$HA_0$<br>$P8_1$/$GA_{20}$<br>$P8_2$/$\overline{CS_1}$<br>$P8_3$/IOR | General input/output or HIF control input/output ($HA_0$, $GA_{20}$, $\overline{CS_1}$, IOR) | | |
| | | $P8_4$/$\overline{IRQ_3}$/$TxD_1$/<br>IOW<br>$P8_5$/$\overline{IRQ_4}$/$RxD_1$/<br>$\overline{CS_2}$<br>$P8_6$/$\overline{IRQ_5}$/$SCK_1$ | General input/output, serial communication interface 1 input/output ($TxD_1$, $RxD_1$, $SCK_1$), HIF control input/output ($\overline{CS_2}$, IOW), or external interrupt input ($\overline{IRQ_3}$ to $\overline{IRQ_5}$) | | |

Continued on next page

| | | | Expanded Modes | | Single-Chip Mode |
|---|---|---|---|---|---|
| **Port** | **Description** | **Pins** | **Mode 1** | **Mode 2** | **Mode 3** |
| Port 9 | • 8-bit input-output port<br>• Can drive bus buffers ($P9_7$) | $P9_0$/$\overline{IRQ_2}$/$\overline{ECS_2}$/ADTRG<br>$P9_1$/$\overline{IRQ_1}$/EIOW<br>$P9_2$/$\overline{IRQ_0}$ | General input/output, HIF control input/output ($\overline{ECS_2}$, EIOW), A/D converter trigger input (ADTRG), or external interrupt input ($\overline{IRQ_2}$–$\overline{IRQ_0}$) | | |
| | | $P9_3$/RD<br>$P9_4$/W R<br>$P9_5$/AS | Expanded data bus control output (RD, W R, AS) | | General input/output |
| | | $P9_6$/ø | System clock output (ø) | | General input or system clock output |
| | | $P9_7$/WAIT | Expanded data bus control input (WAIT) | | General input/output |
| Port A | • 8-bit input-output port<br>• Input pull-ups<br>• Key-sense interrupts<br>• Can drive bus buffers ($PA_4$, $PA_5$, $PA_6$, $PA_7$) | $PA_0$–$PA_7$/<br>$\overline{KEYIN_8}$–$\overline{KEYIN_{15}}$ | General input/output or key-sense interrupts ($\overline{KEYIN_8}$–$\overline{KEYIN_{15}}$) | | |
| Port B | • 8-bit input-output port<br>• Input pull-ups | $PB_0$–$PB_7$/<br>$XDB_0$–$XDB_7$ | General input/output or HIF data bus ($XDB_0$–$XDB_7$) | | General input/output |

• **MOS Input Pull-Ups:** MOS input pull-up transistors are provided on-chip for 48 pins in six ports. The input pull-up transistors in ports 1, 2, 3, and 6 can be switched on or off by setting corresponding bits in a pull-up control register (PCR, KMPCR). The pull-up transistors for output pins are automatically switched off.

The input pull-up transistors in ports A and B are switched on or off by the combination of settings in the corresponding output data register (ODR) and data direction register (DDR). A pull-up is switched on when the corresponding DDR bit is cleared to 0 (input mode) and the ODR bit is set to 1 (1 output). ODR has a different address from the input data register (PIN), so PIN values will not affect the ODR settings when bit manipulation instructions are used.

## • Input/Output Characteristics (5-V version) — Preliminary —

| Parameter | | Symbol | Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| Output high voltage (all outputs) | | $V_{OH}$ | $I_{OH}$ = –200 µA | $V_{CC}$ – 0.5 | — | V |
| | | | $I_{OH}$ = –1.0 mA | 3.5 | — | |
| Output low voltage | (All outputs) | $V_{OL}$ | $I_{OL}$ = 1.6 mA | — | 0.4 | V |
| | ($P8_6$, $P9_7$, $PA_7$ to $PA_4$) | | $I_{OL}$ = 16 mA | — | TBD | |
| | (Ports 1 and 2) | | $I_{OL}$ = 10 mA | — | 1.0 | |
| Input high voltage | | $V_{IH}$ | | 2.0 | $V_{CC}$ + 0.3 | V |
| Input low voltage | | $V_{IL}$ | | –0.3 | 0.8 | V |
| Schmitt trigger input voltages | | $V_T^+$ | | — | $V_{CC} \times 0.7$ | V |
| | | $V_T^-$ | | 1.0 | — | |
| | | $V_T^+ - V_T^-$ | | 0.4 | — | |
| Input pull-up current | (Ports 1, 2, 3) | $-I_P$ | $V_{IN}$ = 0 V | 30 | 250 | µA |
| | (Ports 6, A, B) | | | 60 | 500 | |

## • Input/Output Characteristics (3-V version) — Preliminary —

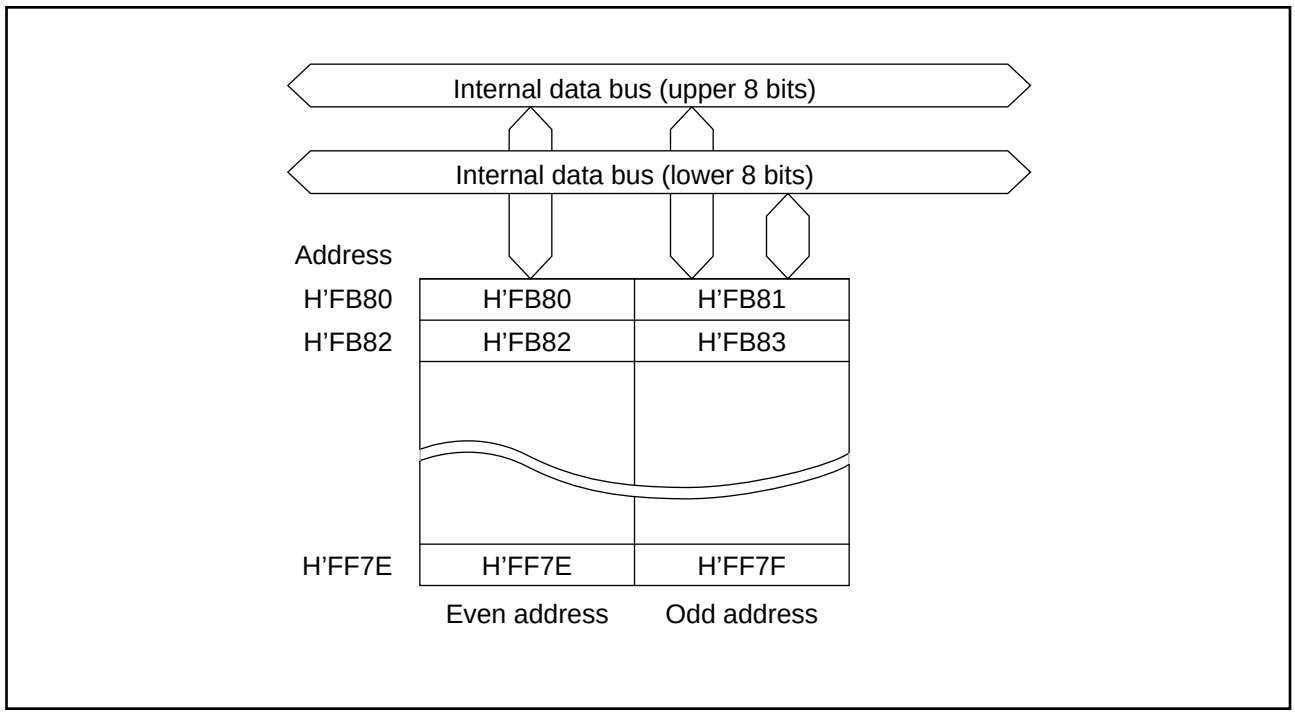
| Parameter | | Symbol | Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| Output high voltage (all outputs) | | $V_{OH}$ | $I_{OH}$ = –200 µA | $V_{CC}$ – 0.4 | — | V |
| | | | $I_{OH}$ = –1.0 mA | $V_{CC}$ – 0.9 | — | |
| Output low voltage | (All outputs) | $V_{OL}$ | $I_{OL}$ = 0.8 mA | — | 0.4 | V |
| | ($P8_6$, $P9_7$, $PA_7$ to $PA_4$) | | $I_{OL}$ = 16 mA | — | TBD | |
| | (Ports 1 and 2) | | $I_{OL}$ = 1.6 mA | — | 0.4 | |
| Input high voltage | | $V_{IH}$ | | $V_{CC} \times 0.7$ | $V_{CC}$ + 0.3 | V |
| Input low voltage | | $V_{IL}$ | | –0.3 | $V_{CC} \times 0.15$ | V |
| Schmitt trigger input voltages | | $V_T^+$ | | — | $V_{CC} \times 0.7$ | V |
| | | $V_T^-$ | | $V_{CC} \times 0.15$ | — | |
| | | $V_T^+ - V_T^-$ | | 0.2 | — | |
| Input pull-up current | (Ports 1, 2, 3) | $-I_P$ | $V_{IN}$ = 0 V | 3 | 120 | µA |
| | (Ports 6, A, B) | | | 30 | 250 | |

## 6.10 RAM

The H8/3437 has 2 kbytes of high-speed static RAM on-chip. The H8/3434 has 1 kbyte. The RAM is linked to the CPU via a 16-bit data bus, so both byte and word data are accessed in two states. This allows particularly rapid transfer of word data.

The on-chip RAM can be enabled and disabled by a bit in the system control register (SYSCR).

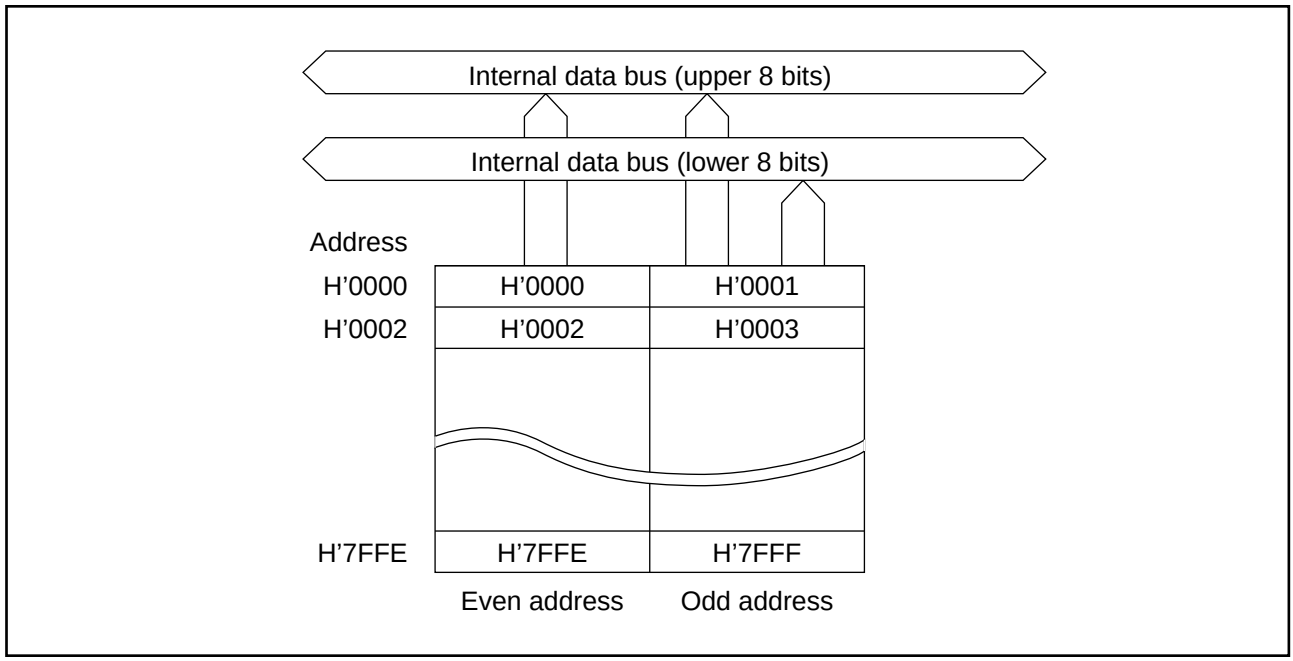
- **Block Diagram (H8/3434)**



## 6.11 Flash Memory, PROM, or Masked ROM

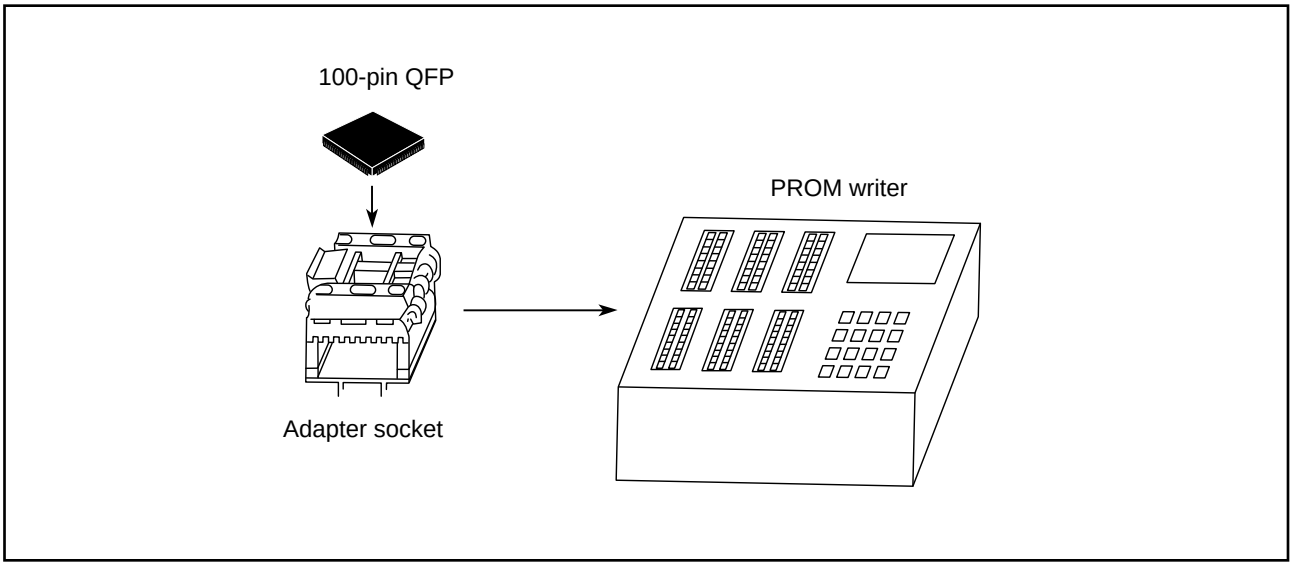
The H8/3437 has 60 kbytes of on-chip PROM or masked ROM. The H8/3434 has 32 kbytes of on-chip flash memory, PROM, or masked ROM. Like the on-chip RAM, the on-chip ROM is linked to the CPU via a 16-bit data bus. Both byte and word data are accessed in two states.

- **Block Diagram (H8/3434)**

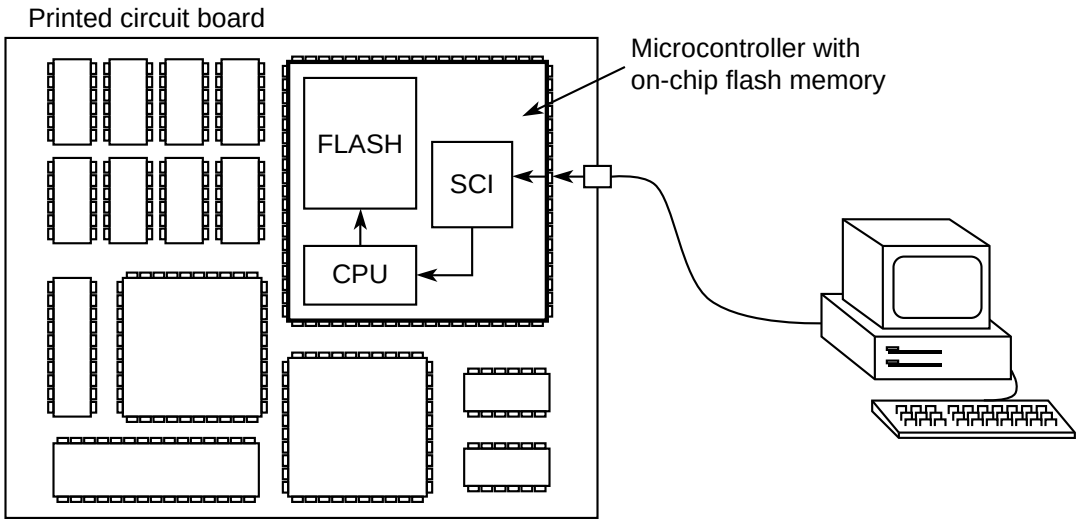


• **Flash Memory and PROM Programming:** When a chip version with on-chip PROM is set to the PROM mode, the chip suspends its microcomputer functions and enables data to be written directly to the on-chip PROM. The on-chip PROM can be written and read according to the same specifications as the 27C101 EPROM ($V_{PP}$ = 12.5 V). If an adapter socket is used to convert from 100 to 32 pins, the PROM can be programmed easily using a commercially available PROM programmer. When selecting a PROM programmer, note that the chip does not support page programming.

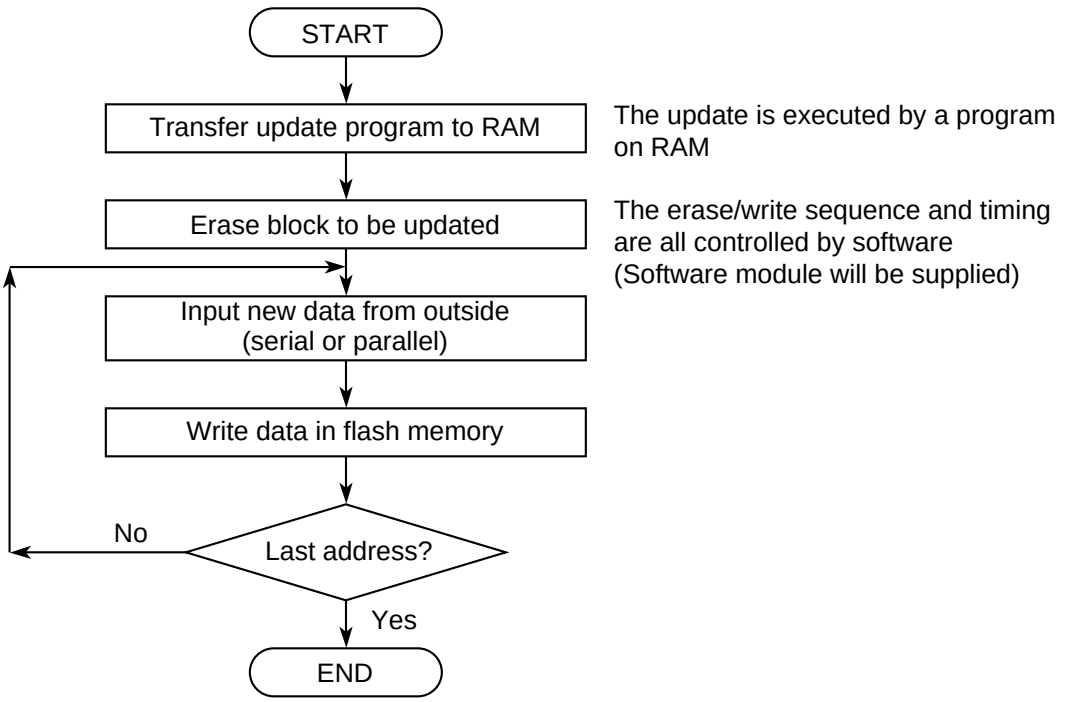
When the H8/3434F-ZTAT™ is set to flash memory mode, data can be written directly to its on-chip flash memory. The read/write specifications for this flash memory are the same as for the 27F101 ($V_{PP}$ = 12.0 V). If an adapter socket is used to convert from 100 to 32 pins, the flash memory can be programmed easily using a commercially available PROM programmer. The adapter socket is a special type for F-ZTAT™ devices.




• **On-Board Programming of Flash Memory:** The flash memory of the H8/3434F-ZTAT™ can be updated while the chip is mounted in the system. The system must supply 12.0 V to the $FV_{PP}$/STBY pin. Updating of the flash memory is controlled by a software routine in the F-ZTAT™. This software routine must be included in the user's program. Sample software for updating the flash memory is provided separately.




The flash memory in the microcontroller (containing programs and parameter data) can be updated on-board




The update is executed by a program on RAM

The erase/write sequence and timing are all controlled by software
(Software module will be supplied)

# Section 7. Power-Down Modes

In addition to the normal program execution state, the chip has a power-down state in which the CPU halts to conserve power. There are three power-down modes: sleep mode, software standby mode, and hardware standby mode. In the two standby modes the system clock also stops.

## 7.1 Sleep Mode

Execution of the SLEEP instruction places the chip in sleep mode, in which the CPU halts but CPU register contents are held and the on-chip supporting modules continue to operate.

Recovery from sleep mode is possible by a reset or any interrupt. The CPU returns via the exception-handling state to the program execution state.

## 7.2 Software Standby Mode

When the software standby (SSBY) flag is set to “1”, execution of the SLEEP instruction places the chip in software standby mode. All chip functions halt, including the clock, but the contents of CPU registers and on-chip RAM are held. I/O ports also hold their existing states.

Recovery from software standby mode is by an external interrupt. After the clock stabilizes, the CPU returns via the exception-handling state to the program execution state.

By stopping the clock, the software standby mode reduces power consumption to an extremely low level.

## 7.3 Hardware Standby Mode

A low input at the $\overline{\text{STBY}}$ pin places the chip in hardware standby mode. As in software standby mode, all chip functions halt but the CPU register contents and on-chip RAM contents are held.

To recover from the hardware standby mode, first drive $\overline{\text{RES}}$ low, then drive $\overline{\text{STBY}}$ high, then drive $\overline{\text{RES}}$ high to restart from the reset state.

By stopping the clock, the hardware standby mode, like the software standby mode, reduces power consumption to an extremely low level.

| Mode | Clock | CPU | Supporting Functions | On-Chip RAM, CPU Registers | Recovery Methods |
|---|---|---|---|---|---|
| Sleep | Runs | Halts | Run | Held | Interrupt—Interrupt is accepted and interrupt handling begins.<br>$\overline{\text{RES}}$—Transition to reset state<br>$\overline{\text{STBY}}$—Transition to hardware standby mode |
| Software standby | Halts | Halts | Halt | Held | External interrupt—<br>The clock starts, and after a clock stabilization time set in an on-chip timer, execution of the external interrupt handling routine starts automatically.<br>$\overline{\text{RES}}$—Clock starts, followed by transition to reset state.<br>$\overline{\text{STBY}}$—Transition to hardware standby mode |
| Hardware standby | Halts | Halts | Halt | Held | Recovered by making the $\overline{\text{RES}}$ pin low, then the $\overline{\text{STBY}}$ pin high; waiting for the clock to stabilize; then making $\overline{\text{RES}}$ high. |

# Section 8. Support Tools

The H8/3437 Series has many software and hardware tools for efficient program development.

**Software**

- H8/300 C compiler
- H8/300 assembler
- Linkage editor (including librarian and load module converter)
- H8/300 simulator/debugger

**Hardware**

- E3000 emulator for H8/3437 Series

## 8.1 Software

- **C Compiler**

- Conforms to ANSI (American National Standards Institute) C-language standard.
- Optimizing function minimizes object program size.
- Can generate assembly-language output in addition to object code.

- **Assembler**

- Supports structured assembly.
- IEEE standard executable instructions and directives.
- Generates object code in SYSROF format.

- **Linkage Editor (including librarian and load module converter)**

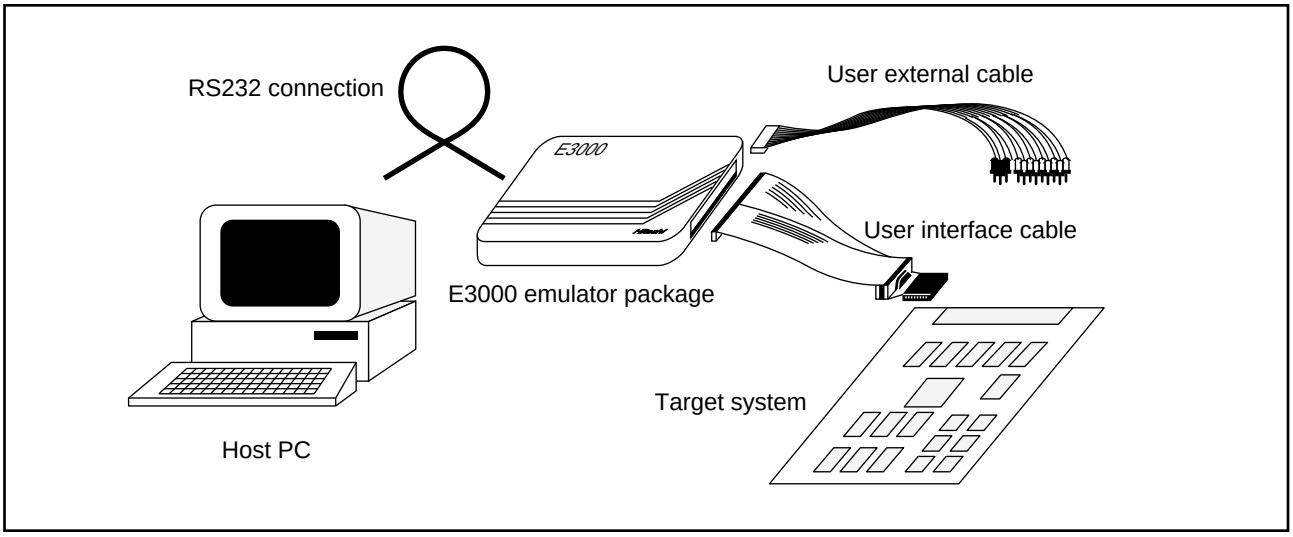
- The linkage editor links and relocates object programs output by the assembler or C compiler (in SYSROF format) to create load modules (in SYSROF format).
- The librarian stores user programs in libraries.
- The load module converter converts SYSROF-format load-module files output by the linkage editor to S-type format.

- **H8/300 Simulator/Debugger**

- Enables the user to debug programs on a host computer, without a target system.
- Can trace registers with breakpoint control.
- Allows simulation with relocatable object code.
- Supports multiuser debugging for team program development.

## 8.2 Hardware

- **Realtime Emulator (E3000) — On Planning/Preliminary Specification —**




- Development host: PC/AT or better
- Host interface: RS232
- Real-time emulation, up to maximum speed of MCU
- Supports MCU modes 1, 2, 3
- Memory
  - — Provides 64-kbyte overlay memory and 54 bits by 1.7 k deep trace memory
  - — 2-byte memory mapping resolution
  - — Write protection
  - — Access protection

- Breakpoints
  - — 4 complex hardware breakpoints
    - – Breaks on specified combinations of MCU’s address, data, range, control, and external signals
    - – Up to 4 levels of hardware sequential break
    - – Event pass count up to 64 k times
    - – Delayed break up to 64 k cycles
  - — 31 instruction breakpoints
  - — 3 breakpoints from user external cable
  - — ROM area protection break with 2-byte resolution
  - — No-memory area protection break with 2-byte resolution
  - — Address range hardware breaks with 2-byte resolution

- Real-time trace
  - Trace conditions are independent of breakpoints
  - 4 complex hardware trace conditions
    - Trace on specified combinations of MCU’s address, data, range, control, and external signals
    - Up to 4 levels of hardware sequential trace
    - Event pass count up to 64 k times
    - Delayed trace up to 64 k cycles
  - Address range exclusion with 2-byte resolution
  - Starts, centers, or ends trace on triggers
  - Trace timer resolution up to 50 ns
  - External scope trigger

- Single-step
  - Displays mnemonics of execution commands
  - Displays mnemonics of the current and next command, register and memory contents
  - Step over functions

- Clock selection
  - Target system clock
  - Emulator built-in clocks
  - User supplied crystal

- Performance analysis
  - Execution time measurement
  - Command and error logging

- Software
  - Command line monitor
  - Online help facilities
  - Symbolic debugging