

***H8*** HITACHI MICROCOMPUTER  
**H8/3048 Series**  
**(H8/3048, H8/3047, H8/3044)**

**HITACHI**

# Preface

Within the H8/300H Series, the H8/3048 Series is a series of single-chip microcontrollers with on-chip ROM and RAM. This document gives a general description of the H8/3048 Series.

The H8/3048 Series includes the H8/3048, H8/3047, and H8/3044, which differ in their ROM and RAM size. All three chips provide many of the peripheral functions required in system control, including a DMA controller, an interrupt controller, a bus controller, a refresh controller, a powerful set of timers, a serial communication interface, an A/D converter, a D/A converter, realtime port output, and I/O ports. These on-chip resources enable high-performance systems with advanced functionality to be implemented easily.

The H8/3048 has a ZTAT™\* (zero turn-around time) version with user-programmable PROM on-chip.

Hitachi is endeavoring to provide a full, efficient development environment for microcontroller application systems. In addition to a C compiler and other support software, the environment includes a realtime emulator (E7000 Series) that can be connected to workstations in a local area network. Also to be provided is a realtime operating system (OS) conforming to the ITRON\*\* specification.

Related manuals:

For hardware information about the H8/3048 Series:

*H8/3048 Series Hardware Manual* (No. ADE-602-073)

For information about the H8/300H Series instruction set:

*H8/300H Series Programming Manual* (No. ADE-602-053)

Notes: \* ZTAT is a registered trademark of Hitachi, Ltd.

\*\* ITRON is an abbreviation of "Industrial TRON." TRON is an abbreviation of "The Realtime Operating-System Nucleus."

# Contents

Section 1	H8/3048 Series Features .....	6
Section 2	Pin Arrangement and Functions.....	10
2.1	Pin Arrangement .....	10
2.2	Pin Arrangement in Each Mode.....	11
2.3	Pin Functions .....	14
Section 3	Block Diagram.....	16
Section 4	CPU .....	17
4.1	Registers.....	17
4.2	Data Formats .....	20
4.3	Addressing Modes .....	22
4.4	Instruction Set .....	24
4.5	Basic Timing .....	32
4.6	Processing States.....	39
4.7	Exception Handling .....	41
4.8	Interrupts .....	42
Section 5	Operating Modes.....	45
5.1	Expanded Modes with On-Chip ROM Disabled (Modes 1 to 4).....	45
5.2	Expanded Modes with On-Chip ROM Enabled (Modes 5 and 6) .....	46
5.3	Single-Chip Mode (Mode 7).....	46
5.4	Memory Maps.....	47
Section 6	Supporting Modules .....	53
6.1	Bus Controller.....	53
6.2	Refresh Controller.....	58
6.3	DMA Controller.....	61
6.4	16-Bit Integrated Timer Unit (ITU) .....	68
6.5	Programmable Timing Pattern Controller (TPC).....	81
6.6	Watchdog Timer (WDT).....	84
6.7	Serial Communication Interface (SCI) .....	87
6.8	Smart Card Interface .....	94
6.9	A/D Converter.....	96

6.10	D/A Converter.....	99
6.11	I/O Ports.....	100
6.12	RAM.....	103
6.13	ROM (PROM or Masked ROM).....	104
<b>Section 7 Power-Down State.....</b>		<b>106</b>
7.1	Sleep Mode.....	106
7.2	Software Standby Mode.....	106
7.3	Hardware Standby Mode.....	106
7.4	Module Standby Function.....	107
7.5	Gear Function.....	108
<b>Section 8 System Development Environment.....</b>		<b>109</b>
8.1	Software.....	110
8.2	Hardware.....	112

## Section 1 H8/3048 Series Features



- **Advanced H8/300H CPU**
- General-register machine
  - Sixteen 16-bit general registers (also useable as sixteen 8-bit plus eight 16-bit general registers, or eight 32-bit general registers)
- High-speed operation for realtime control
  - Maximum clock rate: 18 MHz (oscillator frequency: 18 MHz)
  - High-speed arithmetic operations (times below are at 18 MHz)
    - 8/16/32-bit register-register add/subtract: 111 ns
    - 8 × 8-bit register-register signed multiply: 778 ns
    - 16 ÷ 8-bit register-register signed divide: 778 ns
    - 16 × 16-bit register-register signed multiply: 1.221 μs
    - 32 ÷ 16-bit register-register signed divide: 1.221 μs
- Speed-oriented instruction set
  - 62 basic instruction types
  - 8/16/32-bit data transfer, arithmetic, and logic instructions
  - Signed and unsigned multiply and divide instructions
  - Powerful bit manipulation instructions
  - Instruction length: 2 to 10 bytes
- Maximum 16-Mbyte linear address space

- **On-chip ROM and RAM**

	ROM	RAM
H8/3048	128 kbytes	4 kbytes
H8/3047	96 kbytes	4 kbytes
H8/3044	32 kbytes	2 kbytes

ROM: masked ROM or PROM

- **Bus controller on-chip**

- Address space can be partitioned into eight areas, with independent bus specifications in each area
- Chip select output available for each area
- 8-bit access or 16-bit access selectable for each area
- Two-state or three-state access selectable for each area

- **Wait-state controller on-chip**

- Selection of four wait modes: programmable wait mode, pin auto-wait mode, and pin wait modes 0 and 1

- **Refresh controller on-chip**

- Selection of DRAM refresh, PSRAM refresh, or interval timer function
- Directly connectable to 16-bit-wide DRAM or PSRAM
- Selection of  $2\overline{\text{CAS}}$  or  $2\overline{\text{WE}}$  control
- Programmable battery-back-up mode and refresh interval
- $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$  (CBR) refresh

- **DMA controller (DMAC), 4 channels on-chip**

- Selection of short address mode or full address mode
- Can be activated by internal interrupts
- Four memory-I/O data transfer channels or two memory-memory data transfer channels, independent of the CPU
- Selection of burst mode or cycle-steal mode for auto-requested memory-memory transfers
- Repeat mode selectable for each channel in memory-I/O transfer
- Block transfer mode selectable in memory-memory transfer

- **16-bit integrated timer unit (ITU) on-chip**
  - Five 16-bit timer channels on-chip
  - Up to 12 pulse outputs (with maximum 55.6-ns resolution at 18 MHz)
  - Up to 10 pulse inputs can be processed
  - Automatic counting of two-phase encoder output (phase-counting mode)
  - Can output three complementary PWM waveforms (PWM mode, complementary PWM mode, reset-synchronized PWM mode)
  
- **Programmable timing pattern controller (TPC) on-chip**
  - Maximum 16-bit pulse output, using ITU as time base
  - Output triggering selectable in 4-bit groups
  - Non-overlap intervals can be set between different pulse outputs
  - Selectable timing source (timer)
  
- **Watchdog timer (WDT), 1 channel on-chip**
  - Selection of watchdog timer or interval timer function
  
- **Serial communication interface (SCI), 2 channels on-chip**
  - Selection of asynchronous or synchronous mode
  - Multiprocessor communication function built-in
  - Smart card interface built-in (1 channel)
  
- **10-bit A/D converter, 8 channels on-chip**
  - Selection of single or scan mode
  - Sample-and-hold function
  - Can be externally triggered
  - Conversion range programmable at reference voltage pin ( $V_{REF}$ )
  
- **8-bit D/A converter, 2 channels on-chip**
  
- **Eleven I/O ports**
  - 70 I/O pins, 8 input-only pins
  
- **Interrupt controller**
  - 7 external interrupt pins (NMI, IRQ<sub>0</sub> to IRQ<sub>5</sub>)
  - 30 internal interrupt sources
  - Three-level prioritization available

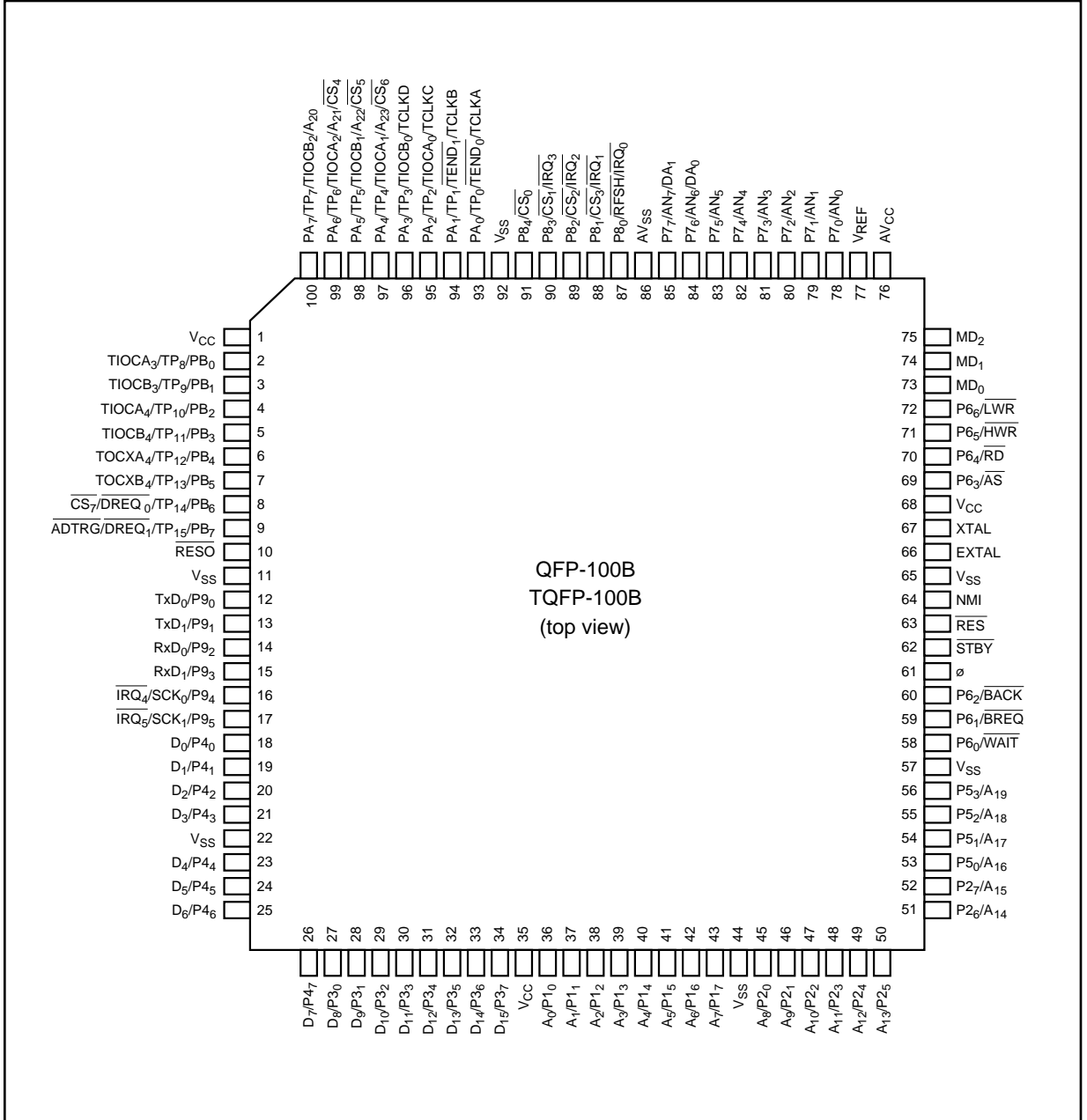
- **Seven operating modes**
  - Expanded 1-Mbyte mode with on-chip ROM disabled, 8-bit bus
  - Expanded 1-Mbyte mode with on-chip ROM disabled, 16-bit bus
  - Expanded 16-Mbyte mode with on-chip ROM disabled, 8-bit bus
  - Expanded 16-Mbyte mode with on-chip ROM disabled, 16-bit bus
  - Expanded 1-Mbyte mode with on-chip ROM enabled, 8-bit bus
  - Expanded 16-Mbyte mode with on-chip ROM enabled, 8-bit bus
  - Single-chip advanced mode (1-Mbyte mode)
  
- **Power-down modes**
  - Sleep mode, software standby mode, hardware standby mode
  - On-chip module standby function
  - On-chip gear (clock-dividing) function
  
- **Clock oscillator on-chip (1:1 oscillator)**
  
- **Package**
  - 100-pin plastic QFP (QFP-100B, TQFP-100B)



# Section 2 Pin Arrangement and Functions

## 2.1 Pin Arrangement

### 100-Pin Plastic QFP (QFP-100B, TQFP-100B)



## 2.2 Pin Arrangement in Each Mode

Pin No.	Pin Name							
	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7	PROM Mode
1	V <sub>CC</sub>				V <sub>CC</sub>			V <sub>CC</sub>
2	PB <sub>0</sub> /TP <sub>8</sub> /TIOCA <sub>3</sub>				PB <sub>0</sub> /TP <sub>8</sub> /TIOCA <sub>3</sub>			NC
3	PB <sub>1</sub> /TP <sub>9</sub> /TIOCB <sub>3</sub>				PB <sub>1</sub> /TP <sub>9</sub> /TIOCB <sub>3</sub>			NC
4	PB <sub>2</sub> /TP <sub>10</sub> /TIOCA <sub>4</sub>				PB <sub>2</sub> /TP <sub>10</sub> /TIOCA <sub>4</sub>			NC
5	PB <sub>3</sub> /TP <sub>11</sub> /TIOCB <sub>4</sub>				PB <sub>3</sub> /TP <sub>11</sub> /TIOCB <sub>4</sub>			NC
6	PB <sub>4</sub> /TP <sub>12</sub> /TOCXA <sub>4</sub>				PB <sub>4</sub> /TP <sub>12</sub> /TOCXA <sub>4</sub>			NC
7	PB <sub>5</sub> /TP <sub>13</sub> /TOCXB <sub>4</sub>				PB <sub>5</sub> /TP <sub>13</sub> /TOCXB <sub>4</sub>			NC
8	PB <sub>6</sub> /TP <sub>14</sub> /DREQ <sub>0</sub> /CS <sub>7</sub>				PB <sub>6</sub> /TP <sub>14</sub> / DREQ <sub>0</sub> /CS <sub>7</sub>	PB <sub>6</sub> /TP <sub>14</sub> / DREQ <sub>0</sub> /CS <sub>7</sub>	PB <sub>6</sub> /TP <sub>14</sub> / DREQ <sub>0</sub>	NC
9	PB <sub>7</sub> /TP <sub>15</sub> /DREQ <sub>1</sub> /ADTRG				PB <sub>7</sub> /TP <sub>15</sub> /DREQ <sub>1</sub> /ADTRG			NC
10	RESO				RESO			V <sub>PP</sub>
11	V <sub>SS</sub>				V <sub>SS</sub>			V <sub>SS</sub>
12	P9 <sub>0</sub> /TXD <sub>0</sub>				P9 <sub>0</sub> /TXD <sub>0</sub>			NC
13	P9 <sub>1</sub> /TXD <sub>1</sub>				P9 <sub>1</sub> /TXD <sub>1</sub>			NC
14	P9 <sub>2</sub> /RXD <sub>0</sub>				P9 <sub>2</sub> /RXD <sub>0</sub>			NC
15	P9 <sub>3</sub> /RXD <sub>1</sub>				P9 <sub>3</sub> /RXD <sub>1</sub>			NC
16	P9 <sub>4</sub> /SCK <sub>0</sub> /IRQ <sub>4</sub>				P9 <sub>4</sub> /SCK <sub>0</sub> /IRQ <sub>4</sub>			NC
17	P9 <sub>5</sub> /SCK <sub>1</sub> /IRQ <sub>5</sub>				P9 <sub>5</sub> /SCK <sub>1</sub> /IRQ <sub>5</sub>			NC
18	P4 <sub>0</sub> /D <sub>0</sub> *	P4 <sub>0</sub> /D <sub>0</sub> *	P4 <sub>0</sub> /D <sub>0</sub> *	P4 <sub>0</sub> /D <sub>0</sub> *	P4 <sub>0</sub> /D <sub>0</sub> *		P4 <sub>0</sub>	NC
19	P4 <sub>1</sub> /D <sub>1</sub> *	P4 <sub>1</sub> /D <sub>1</sub> *	P4 <sub>1</sub> /D <sub>1</sub> *	P4 <sub>1</sub> /D <sub>1</sub> *	P4 <sub>1</sub> /D <sub>1</sub> *		P4 <sub>1</sub>	NC
20	P4 <sub>2</sub> /D <sub>2</sub> *	P4 <sub>2</sub> /D <sub>2</sub> *	P4 <sub>2</sub> /D <sub>2</sub> *	P4 <sub>2</sub> /D <sub>2</sub> *	P4 <sub>2</sub> /D <sub>2</sub> *		P4 <sub>2</sub>	NC
21	P4 <sub>3</sub> /D <sub>3</sub> *	P4 <sub>3</sub> /D <sub>3</sub> *	P4 <sub>3</sub> /D <sub>3</sub> *	P4 <sub>3</sub> /D <sub>3</sub> *	P4 <sub>3</sub> /D <sub>3</sub> *		P4 <sub>3</sub>	NC
22	V <sub>SS</sub>				V <sub>SS</sub>			V <sub>SS</sub>
23	P4 <sub>4</sub> /D <sub>4</sub> *	P4 <sub>4</sub> /D <sub>4</sub> *	P4 <sub>4</sub> /D <sub>4</sub> *	P4 <sub>4</sub> /D <sub>4</sub> *	P4 <sub>4</sub> /D <sub>4</sub> *		P4 <sub>4</sub>	NC
24	P4 <sub>5</sub> /D <sub>5</sub> *	P4 <sub>5</sub> /D <sub>5</sub> *	P4 <sub>5</sub> /D <sub>5</sub> *	P4 <sub>5</sub> /D <sub>5</sub> *	P4 <sub>5</sub> /D <sub>5</sub> *		P4 <sub>5</sub>	NC
25	P4 <sub>6</sub> /D <sub>6</sub> *	P4 <sub>6</sub> /D <sub>6</sub> *	P4 <sub>6</sub> /D <sub>6</sub> *	P4 <sub>6</sub> /D <sub>6</sub> *	P4 <sub>6</sub> /D <sub>6</sub> *		P4 <sub>6</sub>	NC
26	P4 <sub>7</sub> /D <sub>7</sub> *	P4 <sub>7</sub> /D <sub>7</sub> *	P4 <sub>7</sub> /D <sub>7</sub> *	P4 <sub>7</sub> /D <sub>7</sub> *	P4 <sub>7</sub> /D <sub>7</sub> *		P4 <sub>7</sub>	NC
27	D <sub>8</sub>				D <sub>8</sub>		P3 <sub>0</sub>	EO <sub>0</sub>
28	D <sub>9</sub>				D <sub>9</sub>		P3 <sub>1</sub>	EO <sub>1</sub>
29	D <sub>10</sub>				D <sub>10</sub>		P3 <sub>2</sub>	EO <sub>2</sub>
30	D <sub>11</sub>				D <sub>11</sub>		P3 <sub>3</sub>	EO <sub>3</sub>
31	D <sub>12</sub>				D <sub>12</sub>		P3 <sub>4</sub>	EO <sub>4</sub>
32	D <sub>13</sub>				D <sub>13</sub>		P3 <sub>5</sub>	EO <sub>5</sub>
33	D <sub>14</sub>				D <sub>14</sub>		P3 <sub>6</sub>	EO <sub>6</sub>
34	D <sub>15</sub>				D <sub>15</sub>		P3 <sub>7</sub>	EO <sub>7</sub>
35	V <sub>CC</sub>				V <sub>CC</sub>			V <sub>CC</sub>

Note: \* Following a reset, these pins function as P4<sub>7</sub> to P4<sub>0</sub> in modes 1, 3, 5, and 6, and as D<sub>7</sub> to D<sub>0</sub> in modes 2 and 4. These functions can be changed by software.

Pin No.	Pin Name							PROM Mode
	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7	
36		A <sub>0</sub>			P1 <sub>0</sub> /A <sub>0</sub>		P1 <sub>0</sub>	EA <sub>0</sub>
37		A <sub>1</sub>			P1 <sub>1</sub> /A <sub>1</sub>		P1 <sub>1</sub>	EA <sub>1</sub>
38		A <sub>2</sub>			P1 <sub>2</sub> /A <sub>2</sub>		P1 <sub>2</sub>	EA <sub>2</sub>
39		A <sub>3</sub>			P1 <sub>3</sub> /A <sub>3</sub>		P1 <sub>3</sub>	EA <sub>3</sub>
40		A <sub>4</sub>			P1 <sub>4</sub> /A <sub>4</sub>		P1 <sub>4</sub>	EA <sub>4</sub>
41		A <sub>5</sub>			P1 <sub>5</sub> /A <sub>5</sub>		P1 <sub>5</sub>	EA <sub>5</sub>
42		A <sub>6</sub>			P1 <sub>6</sub> /A <sub>6</sub>		P1 <sub>6</sub>	EA <sub>6</sub>
43		A <sub>7</sub>			P1 <sub>7</sub> /A <sub>7</sub>		P1 <sub>7</sub>	EA <sub>7</sub>
44		V <sub>SS</sub>			V <sub>SS</sub>			V <sub>SS</sub>
45		A <sub>8</sub>			P2 <sub>0</sub> /A <sub>8</sub>		P2 <sub>0</sub>	EA <sub>8</sub>
46		A <sub>9</sub>			P2 <sub>1</sub> /A <sub>9</sub>		P2 <sub>1</sub>	OE
47		A <sub>10</sub>			P2 <sub>2</sub> /A <sub>10</sub>		P2 <sub>2</sub>	EA <sub>10</sub>
48		A <sub>11</sub>			P2 <sub>3</sub> /A <sub>11</sub>		P2 <sub>3</sub>	EA <sub>11</sub>
49		A <sub>12</sub>			P2 <sub>4</sub> /A <sub>12</sub>		P2 <sub>4</sub>	EA <sub>12</sub>
50		A <sub>13</sub>			P2 <sub>5</sub> /A <sub>13</sub>		P2 <sub>5</sub>	EA <sub>13</sub>
51		A <sub>14</sub>			P2 <sub>6</sub> /A <sub>14</sub>		P2 <sub>6</sub>	EA <sub>14</sub>
52		A <sub>15</sub>			P2 <sub>7</sub> /A <sub>15</sub>		P2 <sub>7</sub>	CE
53		A <sub>16</sub>			P5 <sub>0</sub> /A <sub>16</sub>		P5 <sub>0</sub>	V <sub>CC</sub>
54		A <sub>17</sub>			P5 <sub>1</sub> /A <sub>17</sub>		P5 <sub>1</sub>	V <sub>CC</sub>
55		A <sub>18</sub>			P5 <sub>2</sub> /A <sub>18</sub>		P5 <sub>2</sub>	NC
56		A <sub>19</sub>			P5 <sub>3</sub> /A <sub>19</sub>		P5 <sub>3</sub>	NC
57		V <sub>SS</sub>			V <sub>SS</sub>			V <sub>SS</sub>
58		P6 <sub>0</sub> /WAIT			P6 <sub>0</sub> /WAIT		P6 <sub>0</sub>	EA <sub>15</sub>
59		P6 <sub>1</sub> /BREQ			P6 <sub>1</sub> /BREQ		P6 <sub>1</sub>	NC
60		P6 <sub>2</sub> /BACK			P6 <sub>2</sub> /BACK		P6 <sub>2</sub>	NC
61		∅			∅			NC
62		STBY			STBY			V <sub>SS</sub>
63		RES			RES			NC
64		NMI			NMI			EA <sub>9</sub>
65		V <sub>SS</sub>			V <sub>SS</sub>			V <sub>SS</sub>
66		EXTAL			EXTAL			NC
67		XTAL			XTAL			NC
68		V <sub>CC</sub>			V <sub>CC</sub>			V <sub>CC</sub>
69		AS			AS		P6 <sub>3</sub>	NC
70		RD			RD		P6 <sub>4</sub>	NC
71		HWR			HWR		P6 <sub>5</sub>	NC
72		LWR			LWR		P6 <sub>6</sub>	NC
73		MD <sub>0</sub>			MD <sub>0</sub>			V <sub>SS</sub>
74		MD <sub>1</sub>			MD <sub>1</sub>			V <sub>SS</sub>
75		MD <sub>2</sub>			MD <sub>2</sub>			V <sub>SS</sub>

Pin No.	Pin Name							PROM Mode
	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7	
76	AV <sub>CC</sub>				AV <sub>CC</sub>			V <sub>CC</sub>
77	V <sub>REF</sub>				V <sub>REF</sub>			V <sub>CC</sub>
78	P7 <sub>0</sub> /AN <sub>0</sub>				P7 <sub>0</sub> /AN <sub>0</sub>			NC
79	P7 <sub>1</sub> /AN <sub>1</sub>				P7 <sub>1</sub> /AN <sub>1</sub>			NC
80	P7 <sub>2</sub> /AN <sub>2</sub>				P7 <sub>2</sub> /AN <sub>2</sub>			NC
81	P7 <sub>3</sub> /AN <sub>3</sub>				P7 <sub>3</sub> /AN <sub>3</sub>			NC
82	P7 <sub>4</sub> /AN <sub>4</sub>				P7 <sub>4</sub> /AN <sub>4</sub>			NC
83	P7 <sub>5</sub> /AN <sub>5</sub>				P7 <sub>5</sub> /AN <sub>5</sub>			NC
84	P7 <sub>6</sub> /AN <sub>6</sub> /DA <sub>0</sub>				P7 <sub>6</sub> /AN <sub>6</sub> /DA <sub>0</sub>			NC
85	P7 <sub>7</sub> /AN <sub>7</sub> /DA <sub>1</sub>				P7 <sub>7</sub> /AN <sub>7</sub> /DA <sub>1</sub>			NC
86	AV <sub>SS</sub>				AV <sub>SS</sub>			V <sub>SS</sub>
87	P8 <sub>0</sub> /RFSH/IRQ <sub>0</sub>				P8 <sub>0</sub> /RFSH/IRQ <sub>0</sub>	P8 <sub>0</sub> /IRQ <sub>0</sub>		EA <sub>16</sub>
88	P8 <sub>1</sub> /CS <sub>3</sub> /IRQ <sub>1</sub>				P8 <sub>1</sub> /CS <sub>3</sub> /IRQ <sub>1</sub>	P8 <sub>1</sub> /IRQ <sub>1</sub>		PGM
89	P8 <sub>2</sub> /CS <sub>2</sub> /IRQ <sub>2</sub>				P8 <sub>2</sub> /CS <sub>2</sub> /IRQ <sub>2</sub>	P8 <sub>2</sub> /IRQ <sub>2</sub>		NC
90	P8 <sub>3</sub> /CS <sub>1</sub> /IRQ <sub>3</sub>				P8 <sub>3</sub> /CS <sub>1</sub> /IRQ <sub>3</sub>	P8 <sub>3</sub> /IRQ <sub>3</sub>		NC
91	P8 <sub>4</sub> /CS <sub>0</sub>				P8 <sub>4</sub> /CS <sub>0</sub>	P8 <sub>4</sub>		NC
92	V <sub>SS</sub>				V <sub>SS</sub>			V <sub>SS</sub>
93	PA <sub>0</sub> /TP <sub>0</sub> /TEND <sub>0</sub> /TCLKA				PA <sub>0</sub> /TP <sub>0</sub> /TEND <sub>0</sub> /TCLKA			NC
94	PA <sub>1</sub> /TP <sub>1</sub> /TEND <sub>1</sub> /TCLKB				PA <sub>1</sub> /TP <sub>1</sub> /TEND <sub>1</sub> /TCLKB			NC
95	PA <sub>2</sub> /TP <sub>2</sub> /TIOCA <sub>0</sub> /TCLKC				PA <sub>2</sub> /TP <sub>2</sub> /TIOCA <sub>0</sub> /TCLKC			NC
96	PA <sub>3</sub> /TP <sub>3</sub> /TIOCB <sub>0</sub> /TCLKD				PA <sub>3</sub> /TP <sub>3</sub> /TIOCB <sub>0</sub> /TCLKD			NC
97	PA <sub>4</sub> /TP <sub>4</sub> /TIOCA <sub>1</sub> / CS <sub>6</sub>	PA <sub>4</sub> /TP <sub>4</sub> /TIOCA <sub>1</sub> / A <sub>23</sub> /CS <sub>6</sub>		PA <sub>4</sub> /TP <sub>4</sub> / TIOCA <sub>1</sub> / CS <sub>6</sub>	PA <sub>4</sub> /TP <sub>4</sub> / TIOCA <sub>1</sub> / A <sub>23</sub> /CS <sub>6</sub>	PA <sub>4</sub> /TP <sub>4</sub> / TIOCA <sub>1</sub>		NC
98	PA <sub>5</sub> /TP <sub>5</sub> /TIOCB <sub>1</sub> / CS <sub>5</sub>	PA <sub>5</sub> /TP <sub>5</sub> /TIOCB <sub>1</sub> / A <sub>22</sub> /CS <sub>5</sub>		PA <sub>5</sub> /TP <sub>5</sub> / TIOCB <sub>1</sub> / CS <sub>5</sub>	PA <sub>5</sub> /TP <sub>5</sub> / TIOCB <sub>1</sub> / A <sub>22</sub> /CS <sub>5</sub>	PA <sub>5</sub> /TP <sub>5</sub> / TIOCB <sub>1</sub>		NC
99	PA <sub>6</sub> /TP <sub>6</sub> /TIOCA <sub>2</sub> / CS <sub>4</sub>	PA <sub>6</sub> /TP <sub>6</sub> /TIOCA <sub>2</sub> / A <sub>21</sub> /CS <sub>4</sub>		PA <sub>6</sub> /TP <sub>6</sub> / TIOCA <sub>2</sub> / CS <sub>4</sub>	PA <sub>6</sub> /TP <sub>6</sub> / TIOCA <sub>2</sub> / A <sub>21</sub> /CS <sub>4</sub>	PA <sub>6</sub> /TP <sub>6</sub> / TIOCA <sub>2</sub>		NC
100	PA <sub>7</sub> /TP <sub>7</sub> /TIOCB <sub>2</sub>	A <sub>20</sub>		PA <sub>7</sub> /TP <sub>7</sub> / TIOCB <sub>2</sub>	A <sub>20</sub>		PA <sub>7</sub> /TP <sub>7</sub> / TIOCB <sub>2</sub>	NC

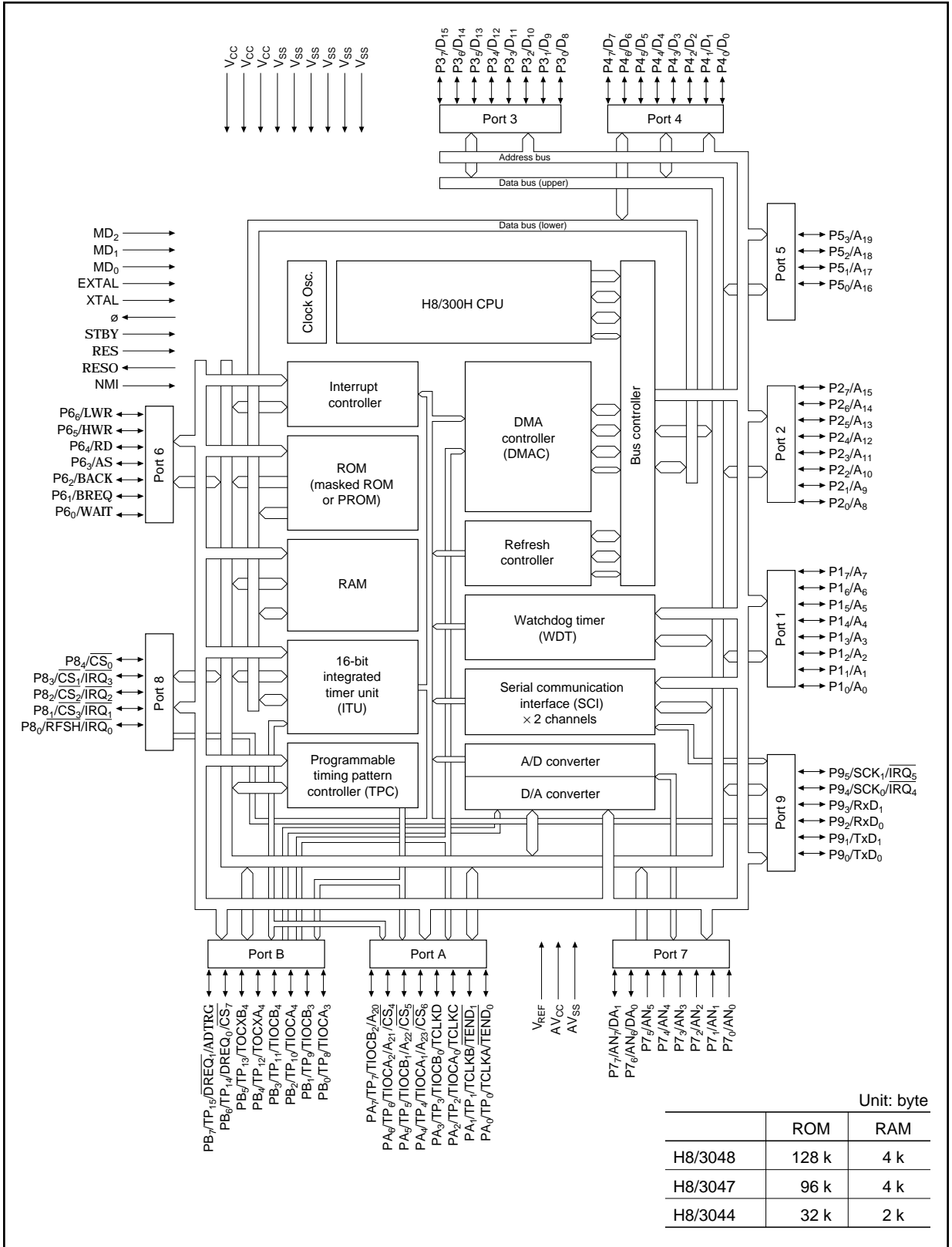
## 2.3 Pin Functions

Type	Symbol	Input/Output	Function
Power	$V_{CC}$	Input	Power supply, connect to +5-V supply
	$V_{SS}$	Input	Ground, connect to 0-V supply
Clock	XTAL	Input	External crystal resonator
	EXTAL	Input	External crystal resonator or external clock input
	$\emptyset$	Output	System clock
System control	RES	Input	Reset input
	RESO	Output	Reset output
	STBY	Input	Standby
	BREQ	Input	Bus request
	BACK	Output	Bus request acknowledge
Address bus	$A_{23}$ to $A_0$	Output	Address bus
Data bus	$D_{15}$ to $D_0$	Input/output	Data bus
Bus control	AS	Output	Address strobe
	$\overline{CS}_7$ to $\overline{CS}_0$	Output	Chip select
	LWR	Output	Low write
	HWR	Output	High write
	RD	Output	Read
	WAIT	Output	Wait
Refresh controller	RFSH	Output	Refresh cycle
	$\overline{CS}_3$	Output	Row address strobe (RAS)
	RD	Output	Column address strobe (CAS)
			Write enable (WE)
	HWR	Output	Upper write (UW) when DRAM is connected to area 3
			Upper column address strobe (UCAS)
LWR	Output	Lower write (LW) when DRAM is connected to area 3	
		Lower column address strobe (LCAS)	
Interrupt signals	NMI	Input	Nonmaskable interrupt request
	$\overline{IRQ}_0$ to $\overline{IRQ}_5$	Input	Interrupt request 0 to 5
DMA controller	$\overline{DREQ}_1, \overline{DREQ}_0$	Input	DMA request 1 and 0
	$\overline{TEND}_1, \overline{TEND}_0$	Output	DMA end 1 and 0
Operating mode control	$MD_2$ to $MD_0$	Input	Mode select
16-bit integrated timer unit	$TIOCA_4$ to $TIOCB_0$	Input/output	Input capture/output compare
	TCLKD to TCLKA	Input	Clock input D to A
	$TOCXA_4, TOCXB_4$	Output	Output compare (or PWM output)
Programmable timing pattern controller	$TP_{15}$ to $TP_0$	Output	Timing pattern output (pulse output)

Type	Symbol	Input/Output	Function
Serial communication interface	TxD <sub>1</sub> , TxD <sub>0</sub>	Output	Transmit data (SCI channels 1 and 0)
	RxD <sub>1</sub> , RxD <sub>0</sub>	Input	Receive data (SCI channels 1 and 0)
	SCK <sub>1</sub> , SCK <sub>0</sub>	Input/output	Serial clock (SCI channels 1 and 0)
A/D converter	AN <sub>7</sub> to AN <sub>0</sub>	Input	Analog input
	ADTRG	Input	A/D converter external trigger input
	AV <sub>CC</sub>	Input	A/D converter power
	AV <sub>SS</sub>	Input	A/D converter ground
	V <sub>REF</sub>	Input	A/D converter reference voltage input
D/A converter	DA1, DA0	Output	D/A output
	AV <sub>CC</sub>	Input	D/A converter power supply
	AV <sub>SS</sub>	Input	D/A converter ground
	V <sub>REF</sub>	Input	D/A converter reference power supply
I/O ports	P1 <sub>7</sub> to P1 <sub>0</sub>	Input/output	Port 1
	P2 <sub>7</sub> to P2 <sub>0</sub>	Input/output	Port 2
	P3 <sub>7</sub> to P3 <sub>0</sub>	Input/output	Port 3
	P4 <sub>7</sub> to P4 <sub>0</sub>	Input/output	Port 4
	P5 <sub>3</sub> to P5 <sub>0</sub>	Input/output	Port 5
	P6 <sub>6</sub> to P6 <sub>0</sub>	Input/output	Port 6
	P7 <sub>7</sub> to P7 <sub>0</sub>	Input	Port 7
	P8 <sub>4</sub> to P8 <sub>0</sub>	Input/output	Port 8
	P9 <sub>5</sub> to P9 <sub>0</sub>	Input/output	Port 9
	PA <sub>7</sub> to PA <sub>0</sub>	Input/output	Port A
	PB <sub>7</sub> to PB <sub>0</sub>	Input/output	Port B

# Section 3 Block Diagram

## Internal Block Diagram



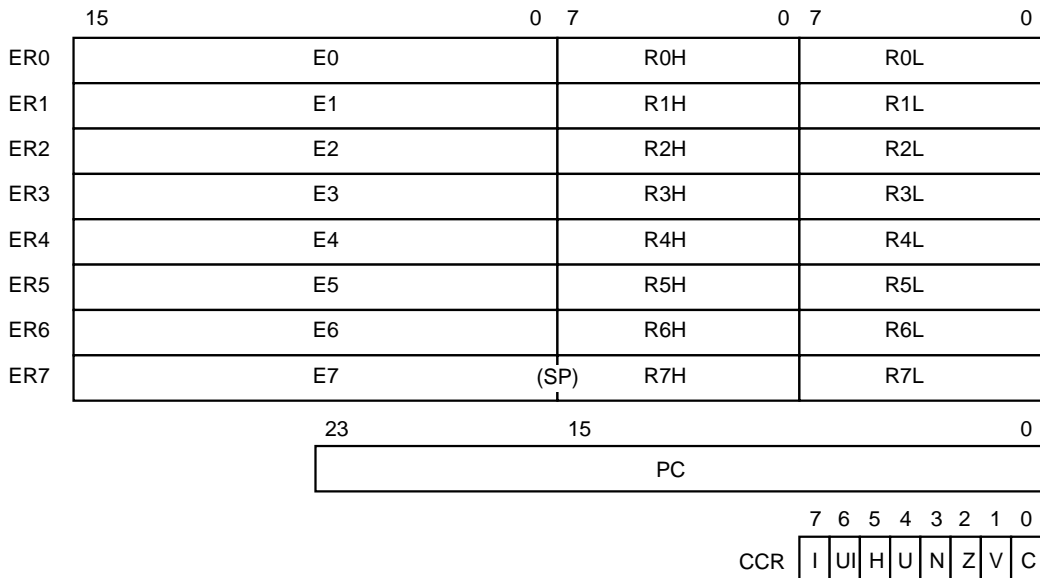
## Section 4 CPU

The central processing unit of the H8/3048 Series is the H8/300H CPU, a high-speed CPU with 32-bit internal data paths and an architecture that is upward-compatible with the H8/300 CPU. The H8/300H CPU has sixteen 16-bit general registers and can address a 16-Mbyte linear address space.

### 4.1 Registers

As shown in the following diagram, the H8/300H CPU has general registers and control registers. The general registers are configured as eight functionally identical 32-bit registers that can be used both as address registers and data registers. The control registers include a 24-bit program counter (PC) and an 8-bit condition code register (CCR).

#### CPU Registers



**Legend**

- SP: Stack pointer
- PC: Program counter
- CCR: Condition code register
- I: Interrupt mask bit
- UI: User bit or interrupt mask bit
- H: Half-carry flag
- U: User bit
- N: Negative flag
- Z: Zero flag
- V: Overflow flag
- C: Carry flag



## General Registers

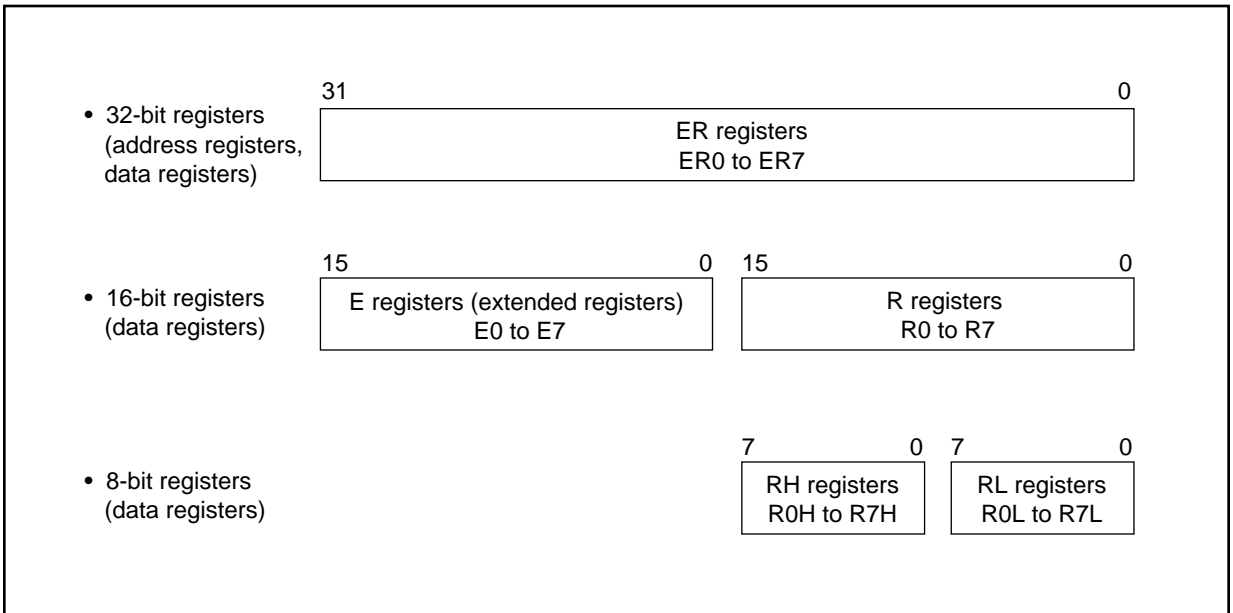
The eight 32-bit general registers are all functionally alike and can be used as both address registers and data registers. When a general register is used as a data register, it can be accessed as a 32-bit, 16-bit, or 8-bit register. When the general registers are used as 32-bit registers or as address registers, they are designated by the letters ER (ER0 to ER7).

The ER registers divide into 16-bit general registers designated by the letters E (E0 to E7) and R (R0 to R7). These registers are functionally equivalent, providing a maximum sixteen 16-bit registers. The E registers (E0 to E7) are also referred to as extended registers.

The R registers divide into 8-bit general registers designated by the letters RH (R0H to R7H) and RL (R0L to R7L). These registers are functionally equivalent, providing a maximum sixteen 8-bit registers.

The following diagram illustrates the usage of the general registers. The usage of each register can be selected independently.

### Usage of General Registers



General register ER7 has the function of stack pointer (SP) in addition to its general-register function, and is used implicitly in exception handling and subroutine calls.

## Control Registers

**Program Counter (PC):** This 24-bit counter indicates the address of the next instruction the CPU will execute.

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

Bit	Symbol	Bit Name	Function
Bit 7	I	Interrupt mask bit	Masks interrupts other than NMI when set to 1. NMI is recognized regardless of the I bit setting. For information on interrupts see 4.8, Interrupts.
Bit 6	UI	User bit or interrupt mask bit	Can be written and read by software. This bit can also be used as an interrupt mask bit.
Bit 5	H	Half-carry flag	When an add, subtract, or compare instruction is executed, this flag is set to 1 if there is a carry or borrow at an intermediate bit, and cleared to 0 otherwise. The intermediate bit is:  ADD.B, ADDX.B, SUB.B, SUBX.B, CMP.B, NEG.B: bit 3 ADD.W, SUB.W, CMP.W, NEG.W: bit 11 ADD.L, SUB.L, CMP.L, NEG.L: bit 27
Bit 4	U	User bit	Can be written and read by software.
Bit 3	N	Negative flag	Indicates the most significant bit (sign bit) of data.
Bit 2	Z	Zero flag	Set to 1 to indicate zero data, and cleared to 0 to indicate non-zero data.
Bit 1	V	Overflow flag	Set to 1 when an arithmetic overflow occurs, and cleared to 0 at other times.
Bit 0	C	Carry flag	Set to 1 when a carry occurs, and cleared to 0 otherwise. Used by add instructions to indicate a carry, subtract instructions to indicate a borrow, and shift and rotate instructions to store the value shifted out of the end bit. The carry flag is also used as a bit accumulator by bit manipulation instructions.

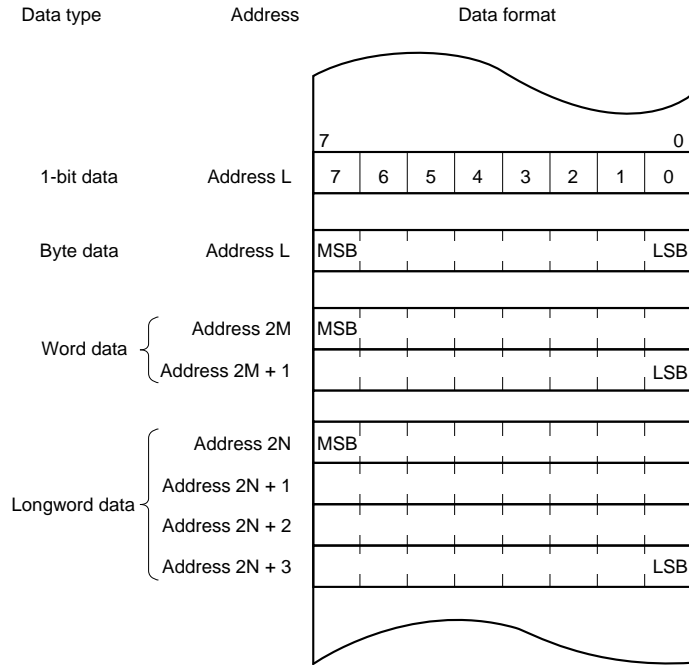
## 4.2 Data Formats

The H8/300H CPU can process 1-bit, 4-bit (BCD), 8-bit (byte), 16-bit (word), and 32-bit (longword) data. Bit-manipulation instructions operate on 1-bit data by accessing bit  $n$  ( $n = 0, 1, 2, \dots, 7$ ) of byte operand data. The DAA and DAS decimal-adjust instructions treat byte data as two digits of 4-bit BCD data.

### General Register Data Formats

Data type	General register	Data format
1-bit data	RnH	
1-bit data	RnL	
4-bit BCD data	RnH	
4-bit BCD data	RnL	
Byte data	RnH	
Byte data	RnL	
Word data	Rn	
Word data	En	
Longword data	ERn	

# Memory Data Formats



Note: Word data and longword data must start at an even address.

## 4.3 Addressing Modes

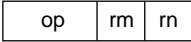

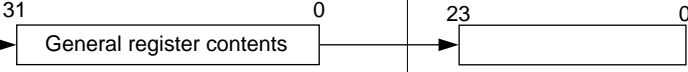
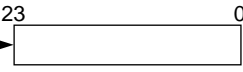
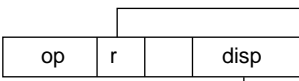
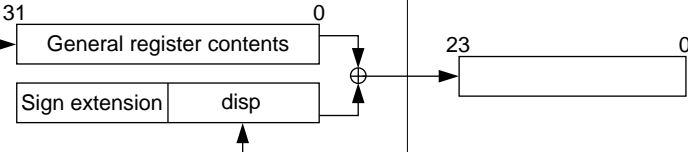
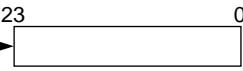

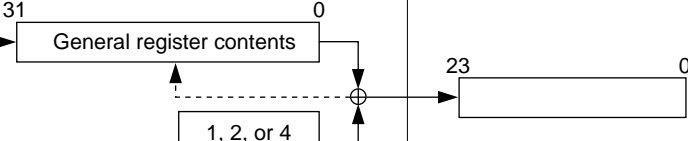
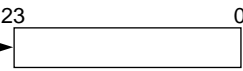
The H8/300H CPU supports the eight addressing modes listed in the following table.

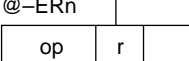
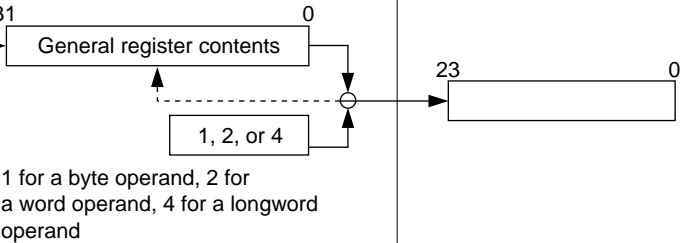

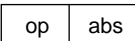


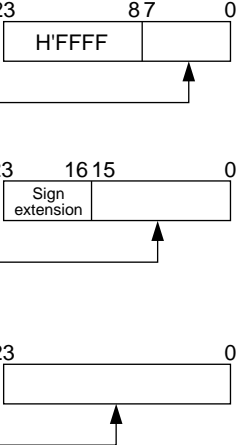


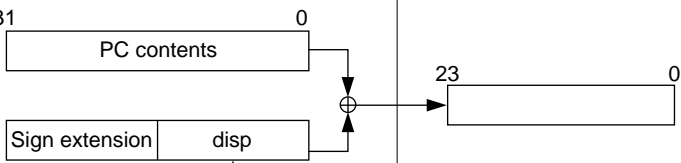
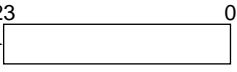
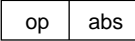
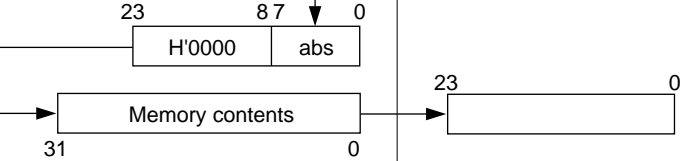
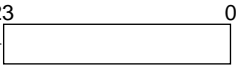
### Addressing Modes

No.	Addressing Mode	Symbol
1	Register direct	Rn
2	Register indirect	@ERn
3	Register indirect with displacement	@(d:16,ERn)/@(d:24,ERn)
4	Register indirect with post-increment Register indirect with pre-decrement	@ERn+ @-ERn
5	Absolute address	@aa:8/@aa:16/@aa:24
6	Immediate	#aa:8/#aa:16/#aa:32
7	Program-counter relative	@(d:8,PC)/@(d:16,PC)
8	Memory indirect	@@aa:8

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

### Effective Address Calculation

No.	Addressing Mode and Instruction Format	Effective Address Calculation	Effective Address (EA)
1	Register direct (Rn) 		Operand is general register contents
2	Register indirect (@ERn) 		
3	Register indirect with displacement @(d:16, ERn) or @(d:24, ERn) 		
4	Register indirect with post-increment or pre-decrement • Register indirect with post-increment @ERn+ 		

No.	Addressing Mode and Instruction Format	Effective Address Calculation	Effective Address (EA)
4	<ul style="list-style-type: none"> <li>Register indirect with pre-decrement</li> </ul> @-ERn 	 <p>1 for a byte operand, 2 for a word operand, 4 for a longword operand</p>	
5	Absolute address @aa:8  @aa:16  @aa:24 		
6	Immediate #xx:8, #xx:16, or #xx:32 		Operand is immediate data
7	Program-counter relative @(d:8, PC) or @(d:16, PC) 		
8	Memory indirect @aa:8 		

### Legend

r, rm, rn: Register field  
 op: Operation field  
 disp: Displacement  
 IMM: Immediate data  
 abs: Absolute address

## 4.4 Instruction Set

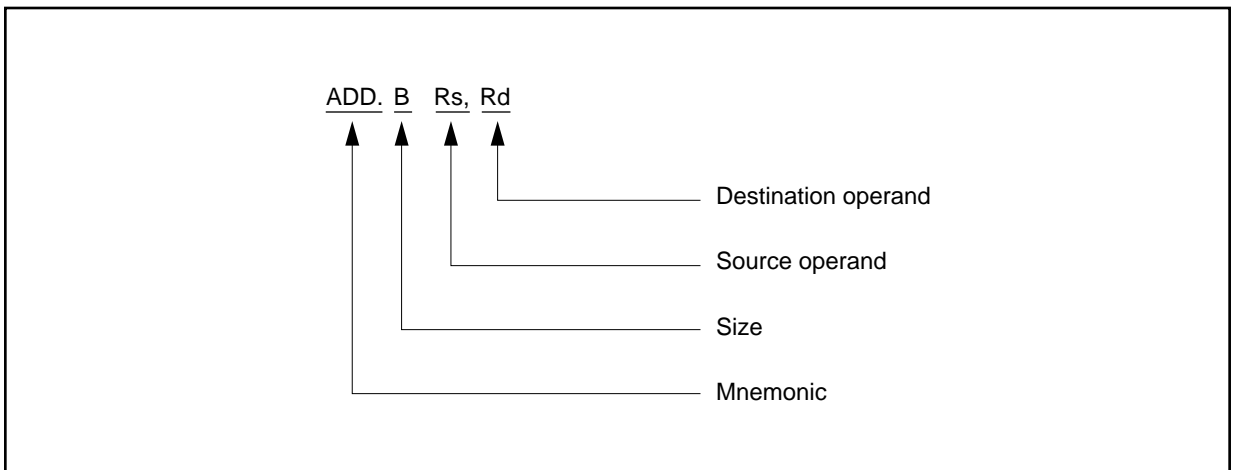
The H8/300H CPU has a set of 62 types of instructions with the following features.

### Features

- General-register architecture
- 8/16/32-bit data transfer, arithmetic, and logic instructions  
Data transfer instructions and basic arithmetic and logic instructions can operate on byte (B), word (W), and longword (L) data
- Signed and unsigned multiply and divide instructions
- Powerful bit manipulation instructions
- High-speed operation

### Assembler Format

The following example illustrates the ADD instruction.



# Instruction Set

	Mnemonic	Operand Size	Operation	Addressing Mode and Instruction Length								Condition Code						No. of States		
				#xx:	Rn	@ERn	@ (d:ERn)	@-ERn/@ERn	@aa:	@ (d:PC)	@@aa	Implied	I	UI	H	N	Z		V	C
Data transfer instructions	MOV.B #xx:8,Rd	B	#xx:8 → Rd8	2										↑	↓	0	—	2		
	MOV.B Rs,Rd	B	Rs8 → Rd8		2									↑	↓	0	—	2		
	MOV.B @ERs,Rd	B	@ERs → Rd8			2								↑	↓	0	—	4		
	MOV.B @(d:16,ERs),Rd	B	@(d:16,ERs) → Rd8				4							↑	↓	0	—	6		
	MOV.B @(d:24,ERs),Rd	B	@(d:24,ERs) → Rd8					8						↑	↓	0	—	10		
	MOV.B @ERs+,Rd	B	@ERs → Rd8, ERs+1 → ERs						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 @aa:24,Rd	B	@aa:24 → Rd8									6		↑	↓	0	—	8		
	MOV.B Rs,@ERd	B	Rs8 → @ERd			2								↑	↓	0	—	4		
	MOV.B Rs,@(d:16,ERd)	B	Rs8 → @(d:16,ERd)				4							↑	↓	0	—	6		
	MOV.B Rs,@(d:24,ERd)	B	Rs8 → @(d:24,ERd)					8						↑	↓	0	—	10		
	MOV.B Rs,@-ERd	B	ERd-1 → ERd, Rs8 → @ERd						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.B Rs,@aa:24	B	Rs8 → @aa:24									6		↑	↓	0	—	8		
	MOV.W #xx:16,Rd	W	#xx:16 → Rd16		4									↑	↓	0	—	4		
	MOV.W Rs,Rd	W	Rs16 → Rd16			2								↑	↓	0	—	2		
	MOV.W @ERs,Rd	W	@ERs → Rd16				2							↑	↓	0	—	4		
	MOV.W @(d:16,ERs),Rd	W	@(d:16,ERs) → Rd16					4						↑	↓	0	—	6		
	MOV.W @(d:24,ERs),Rd	W	@(d:24,ERs) → Rd16						8					↑	↓	0	—	10		
	MOV.W @ERs+,Rd	W	@ERs → Rd16, ERs+2 → ERs							2				↑	↓	0	—	6		
	MOV.W @aa:16,Rd	W	@aa:16 → Rd16								4			↑	↓	0	—	6		
	MOV.W @aa:24,Rd	W	@aa:24 → Rd16									6		↑	↓	0	—	8		
	MOV.W Rs,@ERd	W	Rs16 → @ERd				2							↑	↓	0	—	4		
	MOV.W Rs,@(d:16,ERd)	W	Rs16 → @(d:16,ERd)					4						↑	↓	0	—	6		
	MOV.W Rs,@(d:24,ERd)	W	Rs16 → @(d:24,ERd)						8					↑	↓	0	—	10		
	MOV.W Rs,@-ERd	W	ERd-2 → ERd, Rs16 → @ERd							2				↑	↓	0	—	6		
	MOV.W Rs,@aa:16	W	Rs16 → @aa:16								4			↑	↓	0	—	6		
	MOV.W Rs,@aa:24	W	Rs16 → @aa:24									6		↑	↓	0	—	8		
	MOV.L #xx:32,ERd	L	#xx:32 → ERd32		6									↑	↓	0	—	8		
	MOV.L ERs,ERd	L	ERs32 → ERd32			2								↑	↓	0	—	2		
	MOV.L @ERs,ERd	L	@ERs → ERd32				4							↑	↓	0	—	8		
	MOV.L @(d:16,ERs),ERd	L	@(d:16,ERs) → ERd32					6						↑	↓	0	—	10		
	MOV.L @(d:24,ERs),ERd	L	@(d:24,ERs) → ERd32						10					↑	↓	0	—	14		
	MOV.L @ERs+,ERd	L	@ERs → ERd32, ERs+4 → ERs							4				↑	↓	0	—	10		
	MOV.L @aa:16,ERd	L	@aa:16 → ERd32								6			↑	↓	0	—	10		
	MOV.L @aa:24,ERd	L	@aa:24 → ERd32									8		↑	↓	0	—	12		
	MOV.L ERs,@ERd	L	ERs32 → @ERd				4							↑	↓	0	—	8		
	MOV.L ERs,@(d:16,ERd)	L	ERs32 → @(d:16,ERd)					6						↑	↓	0	—	10		
	MOV.L ERs,@(d:24,ERd)	L	ERs32 → @(d:24,ERd)						10					↑	↓	0	—	14		
	MOV.L ERs,@-ERd	L	ERd-4 → ERd, ERs32 → @ERd							4				↑	↓	0	—	10		
MOV.L ERs,@aa:16	L	ERs32 → @aa:16								6			↑	↓	0	—	10			
MOV.L ERs,@aa:24	L	ERs32 → @aa:24									8		↑	↓	0	—	12			
POP.W Rd	W	@SP+ → Rd16				2							↑	↓	0	—	6			
PUSH.W Rs	W	Rs16 → @-SP					2						↑	↓	0	—	6			
POP.L ERd	L	@SP+ → ERd32					4						↑	↓	0	—	8			
PUSH.L ERs	L	ERs32 → @-SP						4					↑	↓	0	—	8			
MOVFPE@aa:16,Rd	B	Cannot be used in the H8/3048 Series																		
MOVTPE Rs, @aa:16	B	Cannot be used in the H8/3048 Series																		



Mnemonic	Operand Size	Operation	Addressing Mode and Instruction Length							Condition Code						No. of States			
			#xx:	Rn	@ERn	@ (d: .ERn)	@ -ERn/@ERn	@aa:	@ (d: .PC)	@@aa	Implied	I	U	H	N		Z	V	C
Block transfer instructions	EEPMOV.B	—								4	—	—	—	—	—	—	—	⑤	
	if R4L≠0 then repeat @ER5 → @ER6, ER5+1 → ER5, ER6+1 → ER6, R4L-1 → R4L until R4L=0 else next										4	—	—	—	—	—	—	⑤	
	EEPMOV.W	—								4	—	—	—	—	—	—	—	⑤	
	if R4≠0 then repeat @ER5 → @ER6, ER5+1 → ER5, ER6+1 → ER6, R4-1 → R4 until R4=0 else next										4	—	—	—	—	—	—	⑤	
Arithmetic instructions	ADD.B #xx:8,Rd	B	Rd8+#xx:8 → Rd8	2							—	—	↑	↑	↑	↑	↑	2	
	ADD.B Rs,Rd	B	Rd8+Rs8 → Rd8		2						—	—	↑	↑	↑	↑	↑	2	
	ADD.W #xx:16,Rd	W	Rd16+#xx:16 → Rd16	4							—	—	①	↑	↑	↑	↑	4	
	ADD.W Rs,Rd	W	Rd16+Rs16 → Rd16		2						—	—	①	↑	↑	↑	↑	2	
	ADD.L #xx:32,ERd	L	ERd32+#xx:32 → ERd32	6							—	—	②	↑	↑	↑	↑	6	
	ADD.L ERs,ERd	L	ERd32+ERs32 → ERd32		2						—	—	②	↑	↑	↑	↑	2	
	ADDX.B #xx:8,Rd	B	Rd8+#xx:8+C → Rd8	2							—	—	—	↑	↑	③	↑	2	
	ADDX.B Rs,Rd	B	Rd8+Rs8+C → Rd8		2						—	—	—	↑	↑	③	↑	2	
	ADDS #1,ERd	L	ERd32+1 → ERd32	2							—	—	—	—	—	—	—	2	
	ADDS #2,ERd	L	ERd32+2 → ERd32	2							—	—	—	—	—	—	—	2	
	ADDS #4,ERd	L	ERd32+4 → ERd32	2							—	—	—	—	—	—	—	2	
	INC.B Rd	B	Rd8+1 → Rd8	2							—	—	—	↑	↑	↑	↑	2	
	INC.W #1,Rd	W	Rd16+1 → Rd16	2							—	—	—	↑	↑	↑	↑	2	
	INC.W #2,Rd	W	Rd16+2 → Rd16	2							—	—	—	↑	↑	↑	↑	2	
	INC.L #1,ERd	L	ERd32+2 → ERd32	2							—	—	—	↑	↑	↑	↑	2	
	INC.L #2,ERd	L	ERd32+2 → ERd32	2							—	—	—	↑	↑	↑	↑	2	
	DAA Rd	B	Rd8 decimal adjust → Rd8	2							—	—	*	↑	↑	↑	*	④	2
	NEG.B Rd	B	0-Rd8 → Rd8	2							—	—	—	↑	↑	↑	↑	2	
	NEG.W Rd	W	0-Rd16 → Rd16	2							—	—	①	↑	↑	↑	↑	2	
	NEG.L ERd	L	0-ERd32 → ERd32	2							—	—	②	↑	↑	↑	↑	2	
	SUB.B Rs,Rd	B	Rd8-Rs8 → Rd8	2							—	—	—	↑	↑	↑	↑	2	
	SUB.W #xx:16,Rd	W	Rd16-#xx:16 → Rd16	4							—	—	①	↑	↑	↑	↑	4	
	SUB.W Rs,Rd	W	Rd16-Rs16 → Rd16		2						—	—	①	↑	↑	↑	↑	2	
	SUB.L #xx:32,ERd	L	ERd32-#xx:32 → ERd32	6							—	—	②	↑	↑	↑	↑	6	
	SUB.L ERs,ERd	L	ERd32-ERs32 → ERd32		2						—	—	②	↑	↑	↑	↑	2	
	SUBX.B #xx:8,Rd	B	Rd8-#xx:8-C → Rd8	2							—	—	—	↑	↑	③	↑	2	
	SUBX.B Rs,Rd	B	Rd8-Rs8-C → Rd8		2						—	—	—	↑	↑	③	↑	2	
	SUBS #1,ERd	L	ERd32-1 → ERd32	2							—	—	—	—	—	—	—	2	
	SUBS #2,ERd	L	ERd32-2 → ERd32	2							—	—	—	—	—	—	—	2	
	SUBS #4,ERd	L	ERd32-4 → ERd32	2							—	—	—	—	—	—	—	2	
	DEC.B Rd	B	Rd8-1 → Rd8	2							—	—	—	↑	↑	↑	↑	2	
	DEC.W #1,Rd	W	Rd16-1 → Rd16	2							—	—	—	↑	↑	↑	↑	2	
	DEC.W #2,Rd	W	Rd16-2 → Rd16	2							—	—	—	↑	↑	↑	↑	2	
	DEC.L #1,ERd	L	ERd32-1 → ERd32	2							—	—	—	↑	↑	↑	↑	2	
	DEC.L #2,ERd	L	ERd32-2 → ERd32	2							—	—	—	↑	↑	↑	↑	2	
	DAS 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 #xx:16,Rd	W	Rd16-#xx:16	4							—	—	①	↑	↑	↑	↑	4	
	CMP.W Rs,Rd	W	Rd16-Rs16		2						—	—	①	↑	↑	↑	↑	2	
	CMP.L #xx:32,ERd	L	ERd32-#xx:32	6							—	—	②	↑	↑	↑	↑	6	
	CMP.L ERs,ERd	L	ERd32-ERs32		2						—	—	②	↑	↑	↑	↑	2	
MULXU.B Rs,Rd	B	Rd8×Rs8 → Rd16	2							—	—	—	—	—	—	—	14		
MULXU.W Rs,ERd	W	Rd16×Rs16 → ERd32	2							—	—	—	—	—	—	—	22		
DIVXU.B Rs,Rd	B	Rd16÷Rs8 → Rd16 (H: remainder, L: quotient)	2							—	—	—	—	—	—	—	14		
DIVXU.W Rs,ERd	W	ERd32÷Rs16 → ERd32 (E: remainder, R: quotient)	2							—	—	—	—	—	—	—	22		

	Mnemonic	Operand Size	Operation	Addressing Mode and Instruction Length								Condition Code						No. of States		
				#xx:	Rn	@.ERn	@(.ERn)	@-ERn/@ERn	@aa:	@(.PC)	@@aa	Implied	I	U	H	N	Z		V	C
Arithmetic instructions	MULXS.B Rs,ERd	B	Rd8×Rs8 → Rd16	4										↑	↓	—	—	16		
	MULXS.W Rs,ERd	W	Rd16×Rs16 → ERd32	4										↑	↓	—	—	24		
	DIVXS.B Rs,Rd	B	Rd16÷Rs8 → Rd16 (H: remainder, L: quotient)	4										↑	↓	—	—	16		
	DIVXS.W Rs,ERd	W	ERd32÷Rs16 → ERd32 (E: remainder, R: quotient)	4										↑	↓	—	—	24		
	EXTU.W Rd	W	RdL8 zero extension → Rd16	2										↑	↓	0	—	2		
	EXTU.L ERd	L	Rd16 zero extension → ERd32	2										↑	↓	0	—	2		
	EXTS.W Rd	W	RdL8 sign extension → Rd16	2										↑	↓	0	—	2		
EXTS.L ERd	L	Rd16 sign extension → ERd32	2										↑	↓	0	—	2			
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		
	AND.W #xx:16,Rd	W	Rd16∧#xx:16 → Rd16	4										↑	↓	0	—	4		
	AND.W Rs,Rd	W	Rd16∧Rs16 → Rd16	2										↑	↓	0	—	2		
	AND.L #xx:32,ERd	L	ERd32∧#xx:32 → ERd32	6										↑	↓	0	—	6		
	AND.L ERs,ERd	L	ERd32∧ERs32 → ERd32	4										↑	↓	0	—	4		
	OR.B #xx:8,Rd	B	Rd8∨#xx:8 → Rd8	2										↑	↓	0	—	2		
	OR.B Rs,Rd	B	Rd8∨Rs8 → Rd8	2										↑	↓	0	—	2		
	OR.W #xx:16,Rd	W	Rd16∨#xx:16 → Rd16	4										↑	↓	0	—	4		
	OR.W Rs,Rd	W	Rd16∨Rs16 → Rd16	2										↑	↓	0	—	2		
	OR.L #xx:32, ERd	L	ERd32∨#xx:32 → ERd32	6										↑	↓	0	—	6		
	OR.L ERs,ERd	L	ERd32∨ERs32 → ERd32	4										↑	↓	0	—	4		
	XOR.B #xx:8,Rd	B	Rd8⊕#xx:8 → Rd8	2										↑	↓	0	—	2		
	XOR.B Rs,Rd	B	Rd8⊕Rs8 → Rd8	2										↑	↓	0	—	2		
	XOR.W #xx:16,Rd	W	Rd16⊕#xx:16 → Rd16	4										↑	↓	0	—	4		
	XOR.W Rs,Rd	W	Rd16⊕Rs16 → Rd16	2										↑	↓	0	—	2		
	XOR.L #xx:32,ERd	L	ERd32⊕#xx:32 → ERd32	6										↑	↓	0	—	6		
	XOR.L ERs,ERd	L	ERd32⊕ERs32 → ERd32	4										↑	↓	0	—	4		
	NOT.B Rd	B	Rd8 → Rd8	2										↑	↓	0	—	2		
	NOT.W Rd	W	Rd16 → Rd16	2										↑	↓	0	—	2		
NOT.L ERd	L	ERd32 → ERd32	2										↑	↓	0	—	2			
Shift instructions	SHAL.B Rd	B	Rd8 shift arithmetic left → Rd8	2										↑	↓	↑	↓	2		
	SHAL.W Rd	W	Rd16 shift arithmetic left → Rd16	2										↑	↓	↑	↓	2		
	SHAL.L ERd	L	ERd32 shift arithmetic left → ERd32	2										↑	↓	↑	↓	2		
	SHAR.B Rd	B	Rd8 shift arithmetic right → Rd8	2										↑	↓	0	↓	2		
	SHAR.W Rd	W	Rd16 shift arithmetic right → Rd16	2										↑	↓	0	↓	2		
	SHAR.L ERd	L	ERd32 shift arithmetic right → ERd32	2										↑	↓	0	↓	2		
	SHLL.B Rd	B	Rd8 shift logic left → Rd8	2										↑	↓	0	↓	2		
	SHLL.W Rd	W	Rd16 shift logic left → Rd16	2										↑	↓	0	↓	2		
	SHLL.L ERd	L	ERd32 shift logic left → ERd32	2										↑	↓	0	↓	2		
	SHLR.B Rd	B	Rd8 shift logic right → Rd8	2										↑	↓	0	↓	2		
	SHLR.W Rd	W	Rd16 shift logic right → Rd16	2										↑	↓	0	↓	2		
	SHLR.L ERd	L	ERd32 shift logic right → ERd32	2										↑	↓	0	↓	2		
	ROTXL.B Rd	B	Rd8C rotate left → Rd8C	2										↑	↓	0	↓	2		
	ROTXL.W Rd	W	Rd16C rotate left → Rd16C	2										↑	↓	0	↓	2		
	ROTXL.L ERd	L	ERd32C rotate left → ERd32C	2										↑	↓	0	↓	2		
	ROTXR.B Rd	B	Rd8C rotate right → Rd8C	2										↑	↓	0	↓	2		
	ROTXR.W Rd	W	Rd16C rotate right → Rd16C	2										↑	↓	0	↓	2		
	ROTXR.L ERd	L	ERd32C rotate right → ERd32C	2										↑	↓	0	↓	2		
	ROTL.B Rd	B	Rd8 rotate left → Rd8	2										↑	↓	0	↓	2		
	ROTL.W Rd	W	Rd16 rotate left → Rd16	2										↑	↓	0	↓	2		
	ROTL.L ERd	L	ERd32 rotate left → ERd32	2										↑	↓	0	↓	2		
	ROTR.B Rd	B	Rd8 rotate right → Rd8	2										↑	↓	0	↓	2		
	ROTR.W Rd	W	Rd16 rotate right → Rd16	2										↑	↓	0	↓	2		
	ROTR.L ERd	L	ERd32 rotate right → ERd32	2										↑	↓	0	↓	2		

Mnemonic	Operand Size	Operation	Addressing Mode and Instruction Length							Condition Code						No. of States			
			#xx:	Rn	@ERn	@ (d;, ERn)	@ -ERn/@ERn	@aa:	@ (d;, PC)	@ @aa	Implied	I	UI	H	N		Z	V	C
Bit manipulation instructions	BSET #xx:3,Rd	B	(#xx:3 of Rd8) ← 1		2													2	
	BSET #xx:3,@ERd	B	(#xx:3 of @ERd) ← 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,@ERd	B	(Rn8 of @ERd) ← 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,@ERd	B	(#xx:3 of @ERd) ← 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,@ERd	B	(Rn8 of @ERd) ← 0			4												8	
	BCLR Rn,@aa:8	B	(Rn8 of @aa:8) ← 0					4										8	
	BNOT #xx:3,Rd	B	(#xx:3 of Rd8) ← (#xx:3 of Rd8)		2													2	
	BNOT #xx:3,@ERd	B	(#xx:3 of @ERd) ← (#xx:3 of @ERd)			4												8	
	BNOT #xx:3,@aa:8	B	(#xx:3 of @aa:8) ← (#xx:3 of @aa:8)					4										8	
	BNOT Rn,Rd	B	(Rn8 of Rd8) ← (Rn8 of Rd8)		2													2	
	BNOT Rn,@ERd	B	(Rn8 of @ERd) ← (Rn8 of @ERd)			4												8	
	BNOT Rn,@aa:8	B	(Rn8 of @aa:8) ← (Rn8 of @aa:8)					4										8	
	BTST #xx:3,Rd	B	(#xx:3 of Rd8) → Z		2											↑		2	
	BTST #xx:3,@ERd	B	(#xx:3 of @ERd) → Z			4										↑		6	
	BTST #xx:3,@aa:8	B	(#xx:3 of @aa:8) → Z					4								↑		6	
	BTST Rn,Rd	B	(Rn8 of Rd8) → Z		2											↑		2	
	BTST Rn,@ERd	B	(Rn8 of @ERd) → Z			4										↑		6	
	BTST Rn,@aa:8	B	(Rn8 of @aa:8) → Z					4								↑		6	
	BLD #xx:3,Rd	B	(#xx:3 of Rd8) → C		2												↑	2	
	BLD #xx:3,@ERd	B	(#xx:3 of @ERd) → C			4											↑	6	
	BLD #xx:3,@aa:8	B	(#xx:3 of @aa:8) → C					4									↑	6	
	BILD #xx:3,Rd	B	(#xx:3 of Rd8) → C		2												↑	2	
	BILD #xx:3,@ERd	B	(#xx:3 of @ERd) → C			4											↑	6	
	BILD #xx:3,@aa:8	B	(#xx:3 of @aa:8) → C					4									↑	6	
	BST #xx:3,Rd	B	C → (#xx:3 of Rd8)		2													2	
	BST #xx:3,@ERd	B	C → (#xx:3 of @ERd)			4												8	
	BST #xx:3,@aa:8	B	C → (#xx:3 of @aa:8)					4										8	
	BIST #xx:3,Rd	B	C → (#xx:3 of Rd8)		2													2	
	BIST #xx:3,@ERd	B	C → (#xx:3 of @ERd)			4												8	
	BIST #xx:3,@aa:8	B	C → (#xx:3 of @aa:8)					4										8	
	BAND #xx:3,Rd	B	C ∧ (#xx:3 of Rd8) → C		2												↑	2	
	BAND #xx:3,@ERd	B	C ∧ (#xx:3 of @ERd) → C			4											↑	6	
	BAND #xx:3,@aa:8	B	C ∧ (#xx:3 of @aa:8) → C					4									↑	6	
	BIAND #xx:3,Rd	B	C ∧ (#xx:3 of Rd8) → C		2												↑	2	
BIAND #xx:3,@ERd	B	C ∧ (#xx:3 of @ERd) → C			4											↑	6		
BIAND #xx:3,@aa:8	B	C ∧ (#xx:3 of @aa:8) → C					4									↑	6		
BOR #xx:3,Rd	B	C ∨ (#xx:3 of Rd8) → C		2												↑	2		
BOR #xx:3,@ERd	B	C ∨ (#xx:3 of @ERd) → C			4											↑	6		
BOR #xx:3,@aa:8	B	C ∨ (#xx:3 of @aa:8) → C					4									↑	6		
BIOR #xx:3,Rd	B	C ∨ (#xx:3 of Rd8) → C		2												↑	2		
BIOR #xx:3,@ERd	B	C ∨ (#xx:3 of @ERd) → 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,@ERd	B	C ⊕ (#xx:3 of @ERd) → 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,@ERd	B	C ⊕ (#xx:3 of @ERd) → C			4											↑	6		
BIXOR #xx:3,@aa:8	B	C ⊕ (#xx:3 of @aa:8) → C					4									↑	6		

Mnemonic	Operand Size	Operation	Addressing Mode and Instruction Length								Condition Code						No. of States		
			#xx:	Rn	@ERn	@ (d:ERn)	@-ERn/@ERn	@aa:	@ (d:PC)	@@aa	Implied	I	U	H	N	Z		V	C
Branching instructions	Bcc d:8	— if condition is true, then PC ← PC+d:8 else next							2									4	
	Bcc d:16	— if condition is true, then PC ← PC+d:16 else next							4									6	
	JMP @ERn	— PC ← ERn		2														4	
	JMP @aa:24	— PC ← aa:24						4										6	
	JMP @@aa:8 (normal)	— PC ← (aaa:8)16							2									8	
	JMP @@aa:8 (advanced)	— PC ← (aaa:8)24							2									10	
	BSR d:8 (normal)	— SP-2 → SP, PC16 → @SP PC ← PC+d:8							2									6	
	BSR d:8 (advanced)	— SP-4 → SP, PC24 → @SP PC ← PC+d:8							2									8	
	BSR d:16 (normal)	— SP-2 → SP, PC16 → @SP PC ← PC+d:16							4									6	
	BSR d:16 (advanced)	— SP-4 → SP, PC24 → @SP PC ← PC+d:16							4									8	
	JSR @ERn (normal)	— SP-2 → SP, PC16 → @SP PC ← ERn		2														6	
	JSR @ERn (advanced)	— SP-4 → SP, PC24 → @SP PC ← ERn		2														8	
	JSR @aa:24 (normal)	— SP-2 → SP, PC16 → @SP PC ← aa:24						4										8	
	JSR @aa:24 (advanced)	— SP-4 → SP, PC24 → @SP PC ← aa:24						4										10	
	JSR @@aa:8 (normal)	— SP-2 → SP, PC16 → @SP PC ← (@aa:8)16							2									8	
	JSR @@aa:8 (advanced)	— SP-4 → SP, PC24 → @SP PC ← (@aa:8)24							2									12	
	RTS (normal)	— PC ← (@SP)16 SP+2 → SP							2									8	
	RTS (advanced)	— PC24 ← (@SP)24 SP+4 → SP							2									10	
System control instructions	RTE	— CCR ← (@SP)8, PC24 ← (@SP)24 SP+4 → SP							2	↑	↑	↑	↑	↑	↑	↑	10		
	TRAPA #xx:2	— SP-4 → SP, CCR ← (@SP)8, PC24 ← (@SP)24, vector → PC							2	1							14		
	SLEEP	— Transition to sleep mode							2								2		
	NOP	— No operation							2								2		
	LDC #xx:8,CCR	B #xx:8 → CCR	2								↑	↑	↑	↑	↑	↑	2		
	LDC Rs,CCR	B Rs8 → CCR		2							↑	↑	↑	↑	↑	↑	2		
	LDC @ERs,CCR	W @ERs (even) → CCR			4						↑	↑	↑	↑	↑	↑	6		
	LDC @(d:16,ERs),CCR	W @(d:16,ERs) (even) → CCR				6					↑	↑	↑	↑	↑	↑	8		
	LDC @(d:24,ERs),CCR	W @(d:24,ERs) (even) → CCR					10				↑	↑	↑	↑	↑	↑	12		
	LDC @ERs+,CCR	W @ERs (even) → CCR, ERs+2 → ERs					4				↑	↑	↑	↑	↑	↑	8		
	LDC @aa:16,CCR	W @aa:16 (even) → CCR						6			↑	↑	↑	↑	↑	↑	8		
	LDC @aa:24,CCR	W @aa:24 (even) → CCR							8		↑	↑	↑	↑	↑	↑	10		
	STC CCR,Rd	B CCR → Rd8		2													2		
	STC CCR,@ERd	W CCR → @ERd (even)			4												6		
	STC CCR,@(d:16,ERd)	W CCR → @(d:16,ERd) (even)				6											8		
	STC CCR,@(d:24,ERd)	W CCR → @(d:24,ERd) (even)					10										12		
	STC CCR,@-ERd	W ERd-2 → ERd, CCR → @ERd (even)						4									8		
	STC CCR, @aa:16	W CCR → @aa:16 (even)							6								8		
	STC CCR,@aa:24	W CCR → @aa:24 (even)								8							10		
	ANDC #xx:8,CCR	B #xx:8∧CCR → CCR	2								↑	↑	↑	↑	↑	↑	2		
ORC #xx:8,CCR	B #xx:8∨CCR → CCR	2								↑	↑	↑	↑	↑	↑	2			
XORC #xx:8,CCR	B #xx:8⊕CCR → CCR	2								↑	↑	↑	↑	↑	↑	2			

Notes: The number of states is the number of states required for execution when the instruction and its operands are located in the 16-bit, two-state-access address space or in on-chip RAM.

- ① Set to 1 when a carry or borrow occurs at bit 11; otherwise cleared to 0.
- ② Set to 1 when a carry or borrow occurs at bit 27; otherwise cleared to 0.
- ③ Retains its previous value when the result is zero; otherwise cleared to 0.
- ④ Set to 1 when the adjustment produces a carry; otherwise retains its previous value.
- ⑤ The number of states required for execution is  $4n + 8$ , where  $n$  is the value set in R4L for EEPMOV.B or R4 for EEPMOV.W.

## Operand Notation

Symbol	Description
PC	Program counter
SP	Stack pointer
CCR	Condition code register
Z	Zero flag in condition code register
C	Carry flag in condition code register
Rs, Rd, Rn	General data register: 8-bit register (R0H/R0L to R7H/R7L) or 16-bit register (R0 to R7, E0 to E7)
ERs, ERd	General register: 24-bit address register (ER0 to ER7) or 32-bit data register (ER0 to ER7)
d:8, d:16, d:24	8/16/24-bit displacement
#xx:2/3/8/16/32	2/3/8/16/32-bit immediate data
→	Transfer from the operand on the left to the operand on the right
+	Addition of the operands on both sides
−	Subtraction of the operand on the right from the operand on the left
x	Multiplication of the operands on both sides
÷	Division of the operand on the left by the operand on the right
^	Logical AND of the operands on both sides
∨	Logical OR of the operands on both sides
⊕	Exclusive logical OR of the operands on both sides
—	NOT (logical complement, 1's complement)
( ) < >	Contents of operand

## Condition Code Notation

Symbol	Description
↕	Changed according to execution result
*	Undetermined (no guaranteed value)
0	Cleared to 0
1	Set to 1
—	Not affected by execution of the instruction

**Number of States Required for Execution:** The number of states given in the instruction set table applies when the instruction and its operand data are located in a two-state word access area, such as on-chip RAM. Instruction code is located in external memory, but the attributes of the memory area (byte or word access, two- or three-state access, wait states inserted or not, number of wait states) can be selected by the bus controller and wait-state controller. The attributes of the on-chip supporting modules are fixed at either three-state word access or three-state byte access. The number of additional states required for instruction execution depends on the combination of these factors as indicated in the following tables.

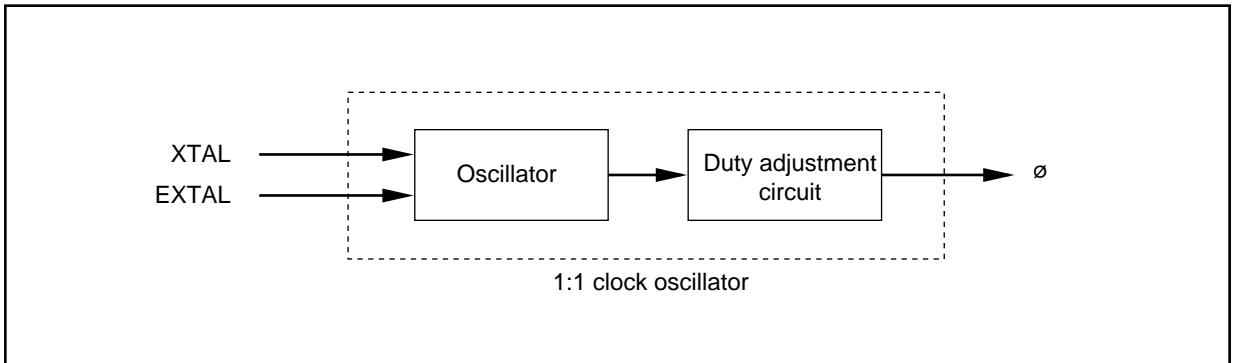
Access Conditions		Number of Additional States Required for Execution	
		Byte Data	Word Data
Operand data	External address (2-state byte access)	0	2
	External address or on-chip RAM (2-state word access)	0	0
	On-chip supporting module (3-state byte access)	1	4
	On-chip supporting module (3-state word access)	1	1
	External address (3-state byte access with m wait states)	1 + m	4 + 2 m
	External address (3-state word access with m wait states)	1 + m	1 + m

Access Conditions			Number of Additional States Required for Execution						
			Branching Instruction		Non-Branching Instruction				
			Instruction Length (Bytes)	2	4	2	4	6	8
Instruction code	External address (2-state byte access)	4	6	2	4	6	8	10	
	External address or on-chip RAM (2-state word access)	0	0	0	0	0	0	0	
	External address (3-state byte access with m wait states)	8 + 4 m	12 + 6 m	4 + 2 m	8 + 4 m	12 + 6 m	16 + 8 m	20 + 10 m	
	External address (3-state word access with m wait states)	2 + 2 m	3 + 3 m	1 + m	2 + 2 m	3 + 3 m	4 + 4 m	5 + 5 m	

## 4.5 Basic Timing

### Basic Clock Timing

The system clock ( $\phi$ ) is generated by input of an external clock signal to the EXTAL pin, or by connecting a crystal resonator to the XTAL and EXTAL pins. The system clock frequency is the same as the frequency of the crystal resonator or external clock signal.

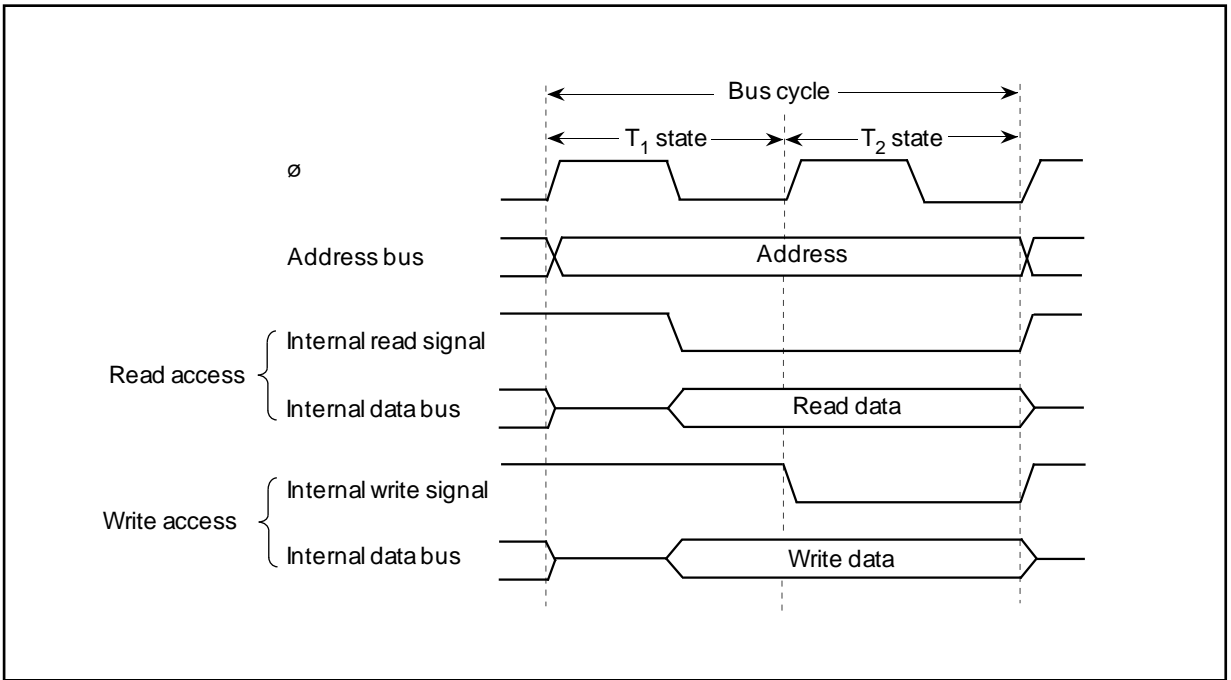


**Block Diagram of Clock Pulse Generator**

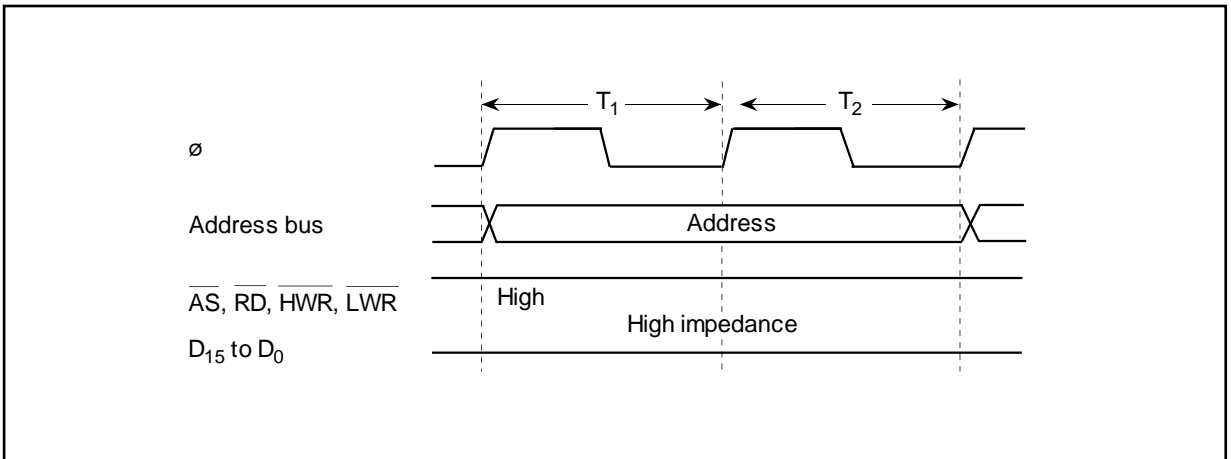
### CPU Read and Write Cycles

The CPU operates using the system clock ( $\phi$ ) as a time base. One cycle of the system clock is called a state. Each bus cycle consists of two or three states. The CPU uses different methods to access on-chip memory, the on-chip supporting modules, and external devices. Access to the external address space can be controlled by the bus controller.

**On-Chip Memory:** For high-speed processing, on-chip memory is accessed in word size in two states.



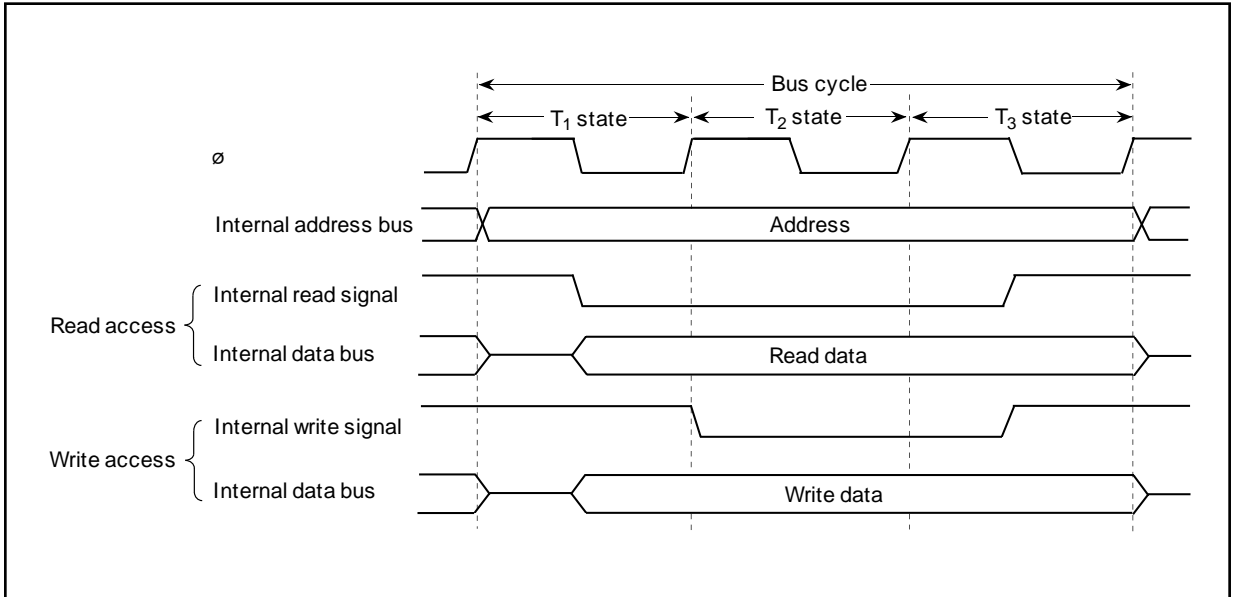
**On-Chip Memory Access Cycle**



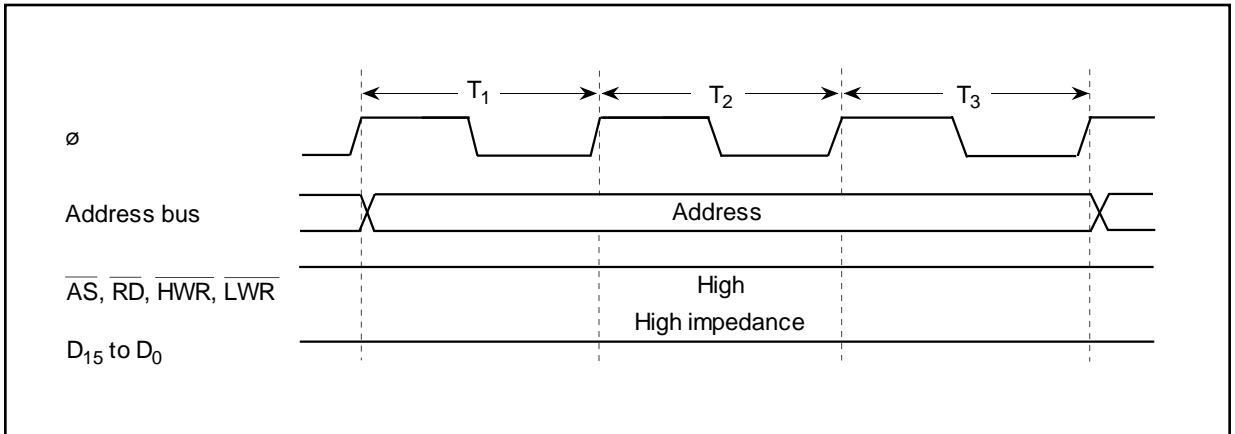
**Pin States During On-Chip Memory Access**



**On-Chip Supporting Modules:** The on-chip supporting modules are accessed in three states, in byte size or word size.

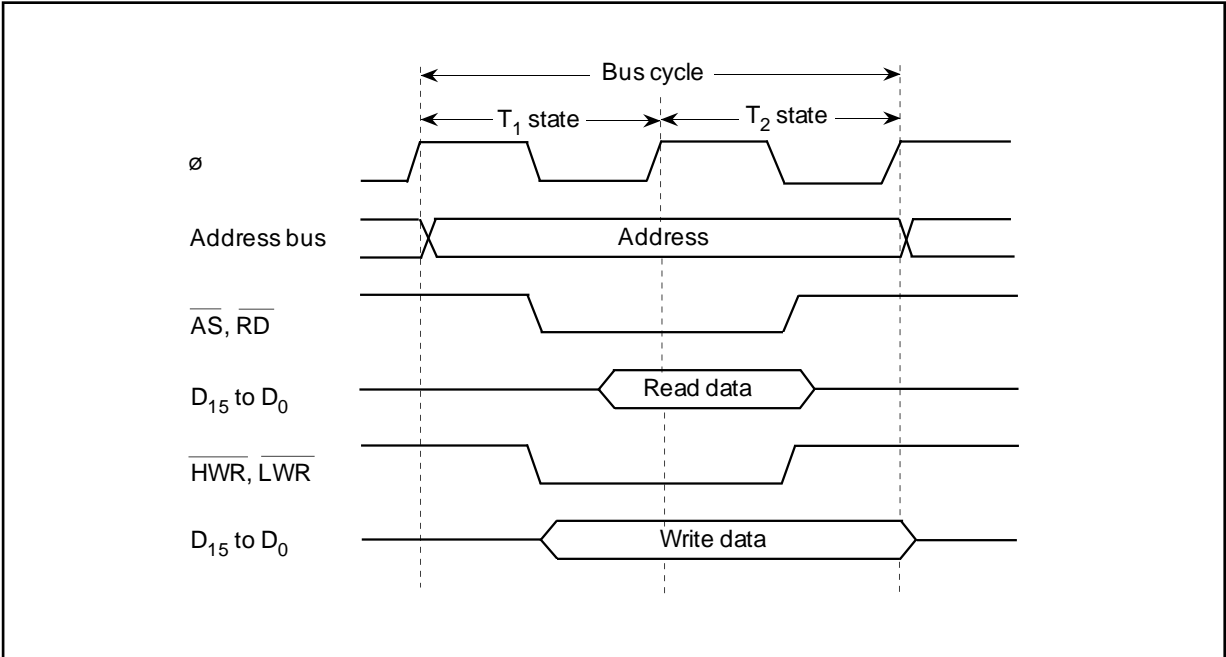


**On-Chip Supporting Module Access Cycle**

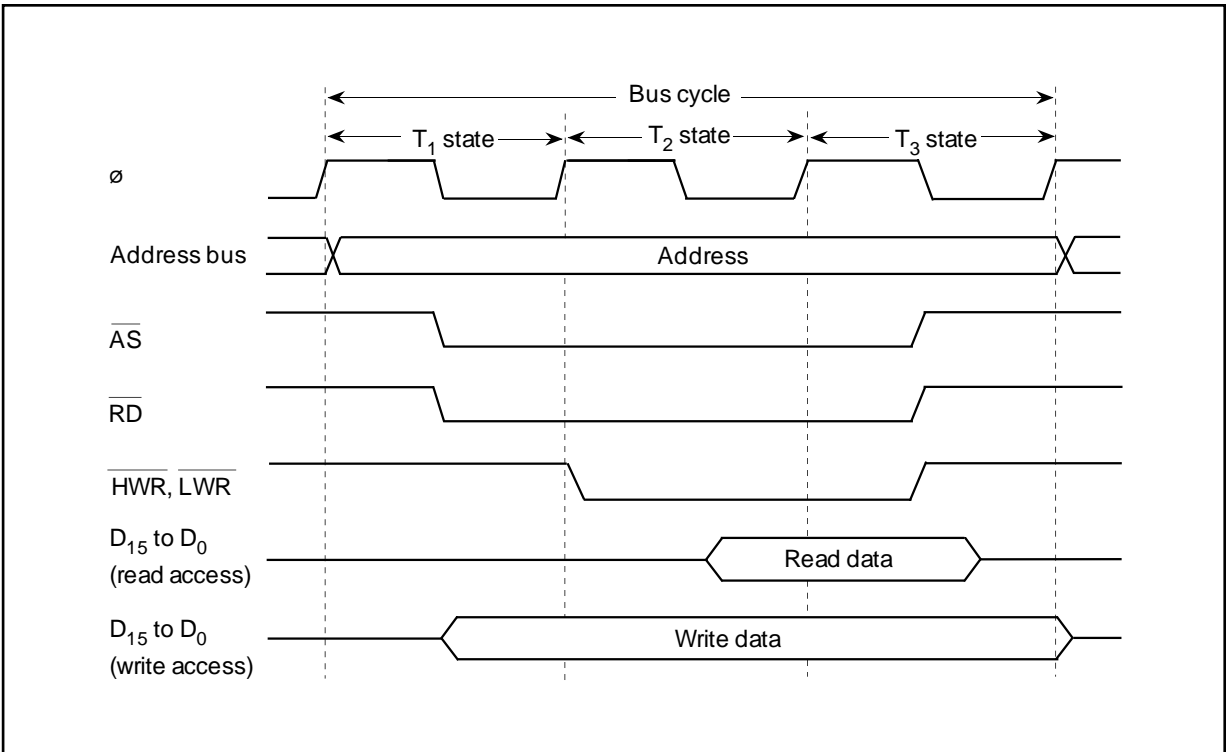


**Pin States During On-Chip Supporting Module Access**

**External Address Space:** The external address space is divided into eight areas. The data bus width (8 or 16 bits) and access cycle length (two or three states) can be selected for each area independently.



**Access Cycle for Two-State-Access Areas**

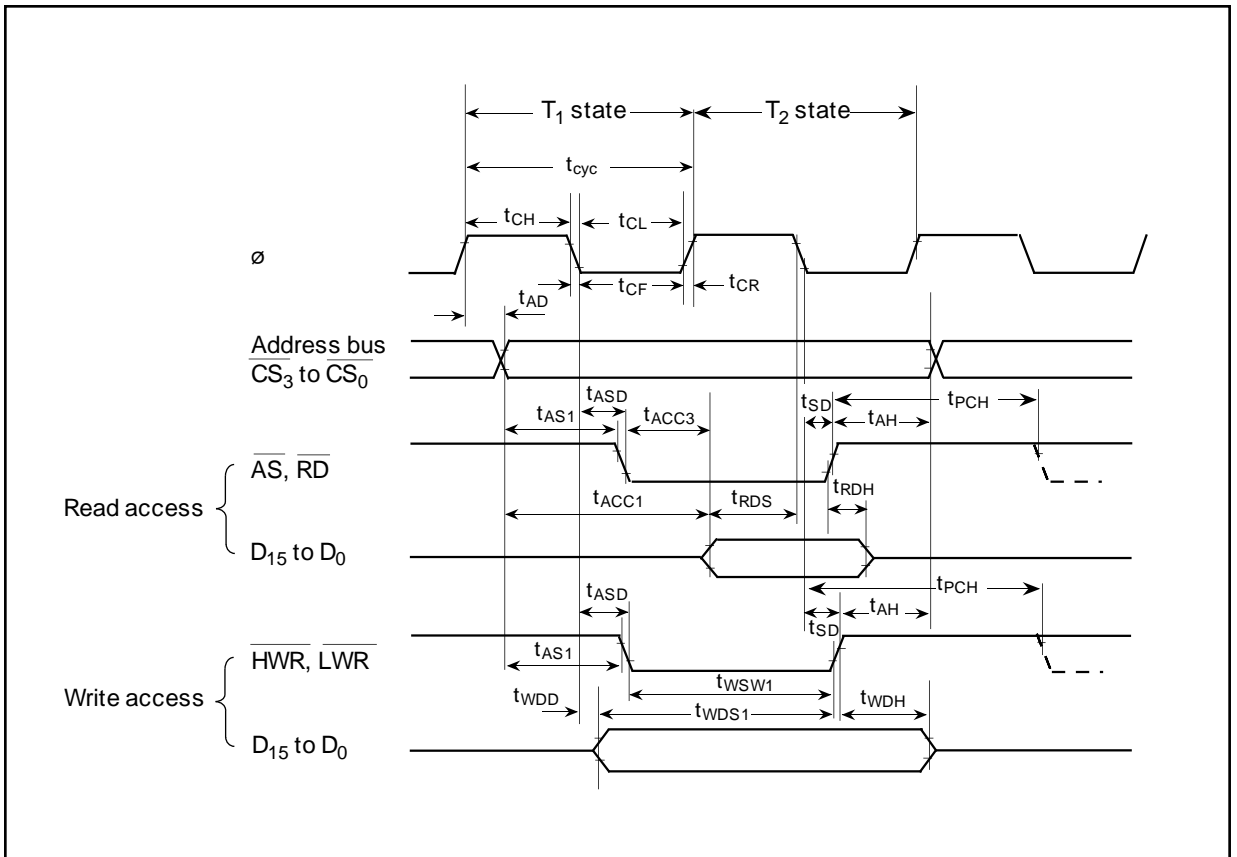


**Access Cycle for Three-State-Access Areas**

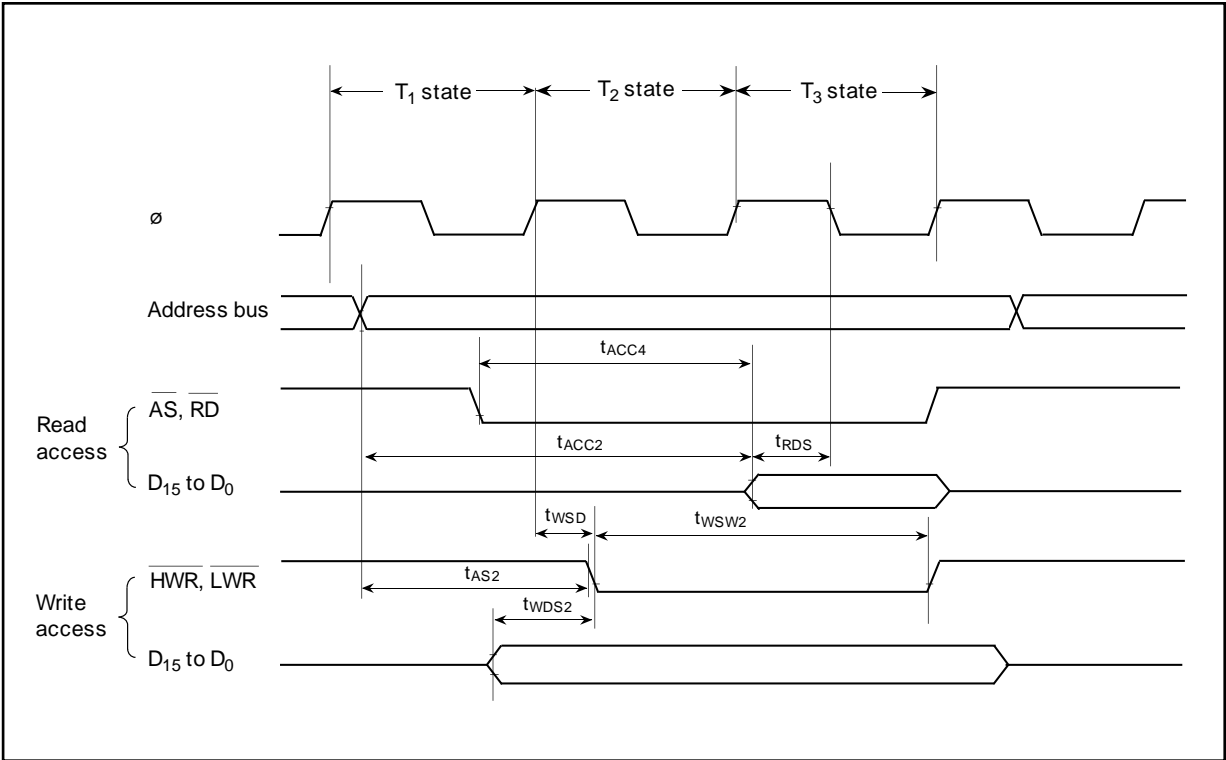
Each of the eight areas can be designated as an 8-bit access area or a 16-bit access area. The upper part of the data bus ( $D_{15}$  to  $D_8$ ) is used to access an 8-bit access area. Both the upper part ( $D_{15}$  to  $D_8$ ) and lower part ( $D_7$  to  $D_0$ ) are used to access a 16-bit access area. The following table gives further details.

### Usage of Data Bus to Access Different Areas

Area	Access Size	Read/Write	Address	Valid Strobe	Upper Data Bus ( $D_{15}$ to $D_8$ )	Lower Data Bus ( $D_7$ to $D_0$ )
8-bit-access area	Byte	Read	Even/odd	RD	Valid	Invalid
		Write	Even/odd	HWR		Undetermined
16-bit-access area	Byte	Read	Even	RD	Valid	Invalid
			Odd		Invalid	Valid
		Write	Even	HWR	Valid	Undetermined
			Odd	LWR	Undetermined	Valid
	Word	Read	Even	RD	Valid	Valid
		Write	Even	HWR, LWR	Valid	Valid



**External Device Access Cycle (Two-State-Access Area)**



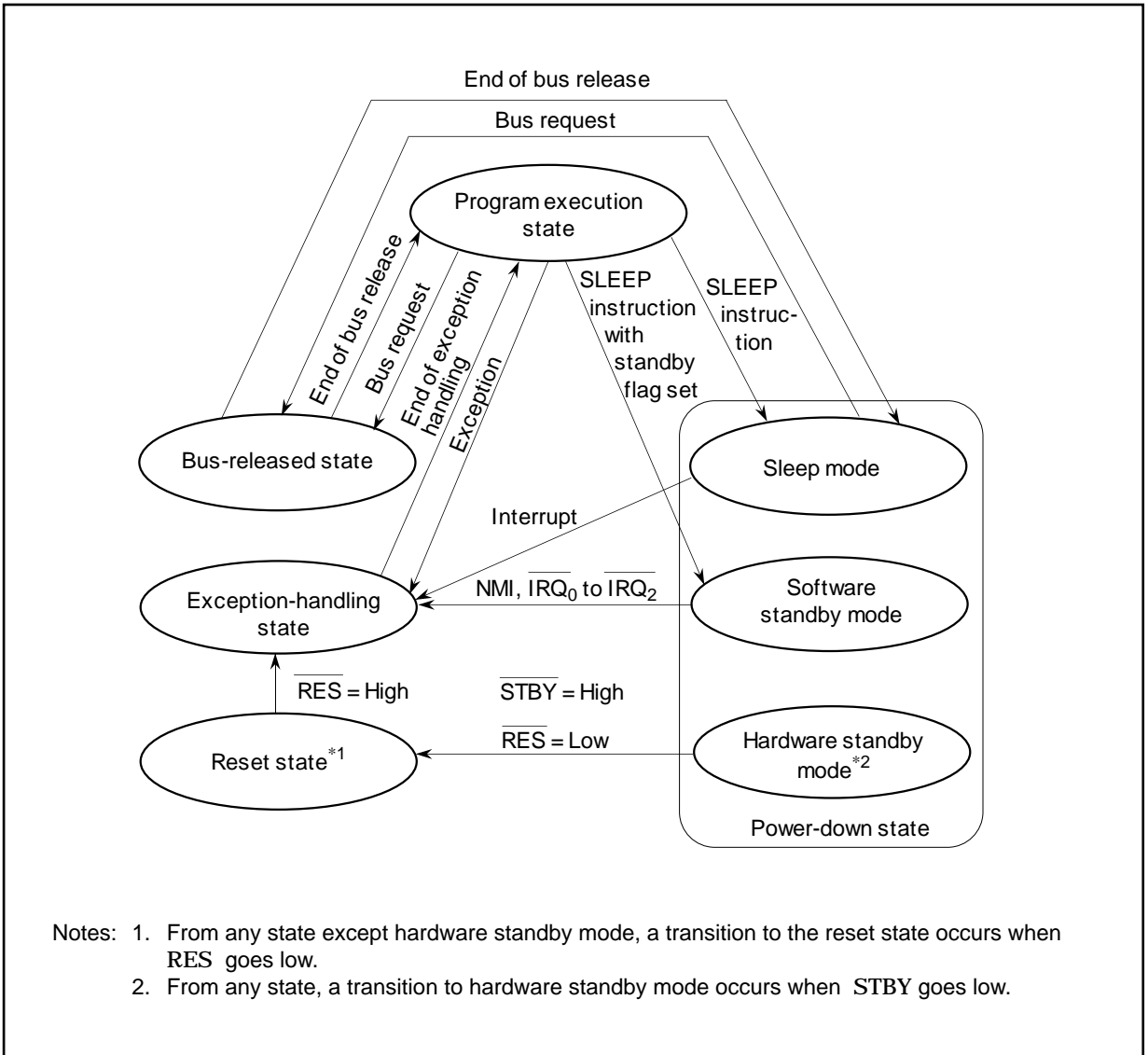
**External Device Access Cycle (Three-State-Access Area)**

Item	Symbol	$V_{CC} = 2.7\text{ V to }5.5\text{ V}$		$V_{CC} = 5.0\text{ V } \pm 10\%$	
		8 MHz		18 MHz	
		Min	Max	Min	Max
Clock cycle time	$t_{cyc}$	125	1000	55.5	1000
Clock low pulse width	$t_{CL}$	40	—	17	—
Clock high pulse width	$t_{CH}$	40	—	17	—
Clock rise time	$t_{CR}$	—	20	—	10
Clock fall time	$t_{CF}$	—	20	—	10
Address delay time	$t_{AD}$	—	60	—	25
Address hold time	$t_{AH}$	25	—	10	—
Address strobe delay time	$t_{ASD}$	—	60	—	25
Write strobe delay time	$t_{WSD}$	—	60	—	25
Strobe delay time	$t_{SD}$	—	60	—	25
Write data strobe pulse width 1	$t_{WSW1}$	85	—	32	—
Write data strobe pulse width 2	$t_{WSW2}$	150	—	62	—
Address setup time 1	$t_{AS1}$	20	—	10	—
Address setup time 2	$t_{AS2}$	80	—	38	—
Read data setup time	$t_{RDS}$	50	—	15	—
Read data hold time	$t_{RDH}$	0	—	0	—
Write data delay time	$t_{WDD}$	—	75	—	50
Write data setup time 1	$t_{WDS1}$	90	—	10	—
Write data setup time 2	$t_{WDS2}$	15	—	10	—
Write data hold time	$t_{WDH}$	25	—	20	—
Read data access time 1	$t_{ACC1}$	—	120	—	50
Read data access time 2	$t_{ACC2}$	—	240	—	105
Read data access time 3	$t_{ACC3}$	—	70	—	20
Read data access time 4	$t_{ACC4}$	—	180	—	80
Precharge time	$t_{PCH}$	85	—	40	—
Wait setup time	$t_{WTS}$	40	—	25	—
Wait hold time	$t_{WTH}$	10	—	5	—
Bus request setup time	$t_{BRQS}$	40	—	40	—
Bus acknowledge delay time 1	$t_{BACD1}$	—	60	—	30
Bus acknowledge delay time 2	$t_{BACD2}$	—	60	—	30
Bus-floating delay time	$t_{BZD}$	—	70	—	40

Unit: ns

## 4.6 Processing States

The CPU has five processing states: the reset state, program execution state, exception-handling state, bus-released state, and power-down state. The state transitions are as shown next.



**State Transition Diagram**

## **Program Execution State**

In this state the CPU executes program instructions in sequence.

## **Exception-Handling State**

This is a transient state in which the CPU executes an exception-handling sequence in response to a reset, interrupt, or other exception.

## **Bus-Released State**

In this state the external bus has been released in response to a bus request signal from a bus master other than the CPU.

## **Power-Down State**

The CPU is halted to conserve power. This state includes three modes: a sleep mode, software standby mode, and hardware standby mode.

## 4.7 Exception Handling

The sources of exception handling are classified in the following table. When exception handling is requested by one of these sources, the CPU saves PC and CCR onto the stack, sets the interrupt mask bit to 1 in CCR, fetches a starting address from the exception vector table, and starts program execution from that address. If two or more exceptions occur simultaneously, they are handled in priority order. Trap exceptions are accepted at all times in the program execution state.

### Exception Handling Types and Priority

Priority	Type of Exception	Start of Exception Handling
High	Reset	Exception handling starts immediately when RES changes from low to high
↑	Interrupt	When an interrupt is requested, exception handling starts at the end of the current instruction or current exception-handling sequence
↓	Trap instruction	Exception handling starts when a trap (TRAPA) instruction is executed
Low		

### Exception Vector Table

Priority	Exception	Vector No.	Vector Address	
High	Reset	0	H'0000 to H'0003	
↑	(Reserved for system use)	1 to 6	H'0004 to H'001B	
	NMI interrupt	7	H'001C to H'001F	
	Trap instruction (four sources)	8	H'0020 to H'0023	
		9	H'0024 to H'0027	
		10	H'0028 to H'002B	
		11	H'002C to H'002F	
	External interrupts	IRQ <sub>0</sub>	12	H'0030 to H'0033
		IRQ <sub>1</sub>	13	H'0034 to H'0037
		IRQ <sub>2</sub>	14	H'0038 to H'003B
		IRQ <sub>3</sub>	15	H'003C to H'003F
		IRQ <sub>4</sub>	16	H'0040 to H'0043
		IRQ <sub>5</sub>	17	H'0044 to H'0047
	(Reserved for system use)	18	H'0048 to H'004B	
		19	H'004C to H'004F	
Low	Internal interrupts	20 to 60	H'0050 to H'00F3	



## 4.8 Interrupts

Interrupts are controlled by the interrupt controller. Interrupt exception handling can be requested by a total of 37 sources, including seven external sources (NMI,  $\overline{\text{IRQ}}_0$  to  $\overline{\text{IRQ}}_5$ ), and 30 internal sources in the on-chip supporting modules. Each interrupt source has a separate vector address.

NMI is the highest-priority interrupt and is always accepted.

The interrupt-handling sequence differs depending on the state of the UE bit in the system control register (SYSCR). Interrupts are controlled by the I bit when UE = 1, and by the I and UI bits when UE = 0. The following table indicates how interrupts operate under all combinations of the UE, I, and UI bits.

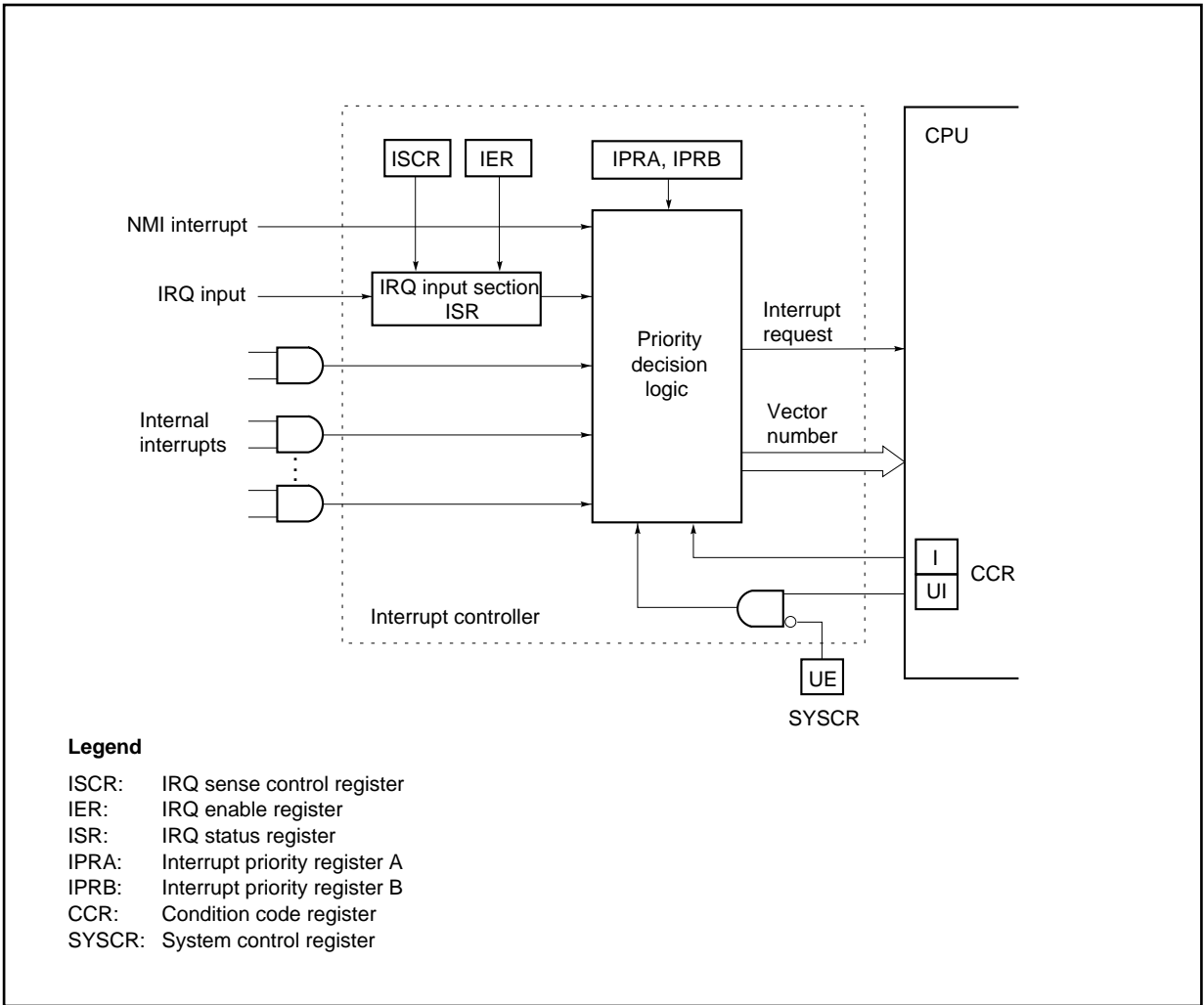
### Interrupt Control by UE, I, and UI Bits

SYSCR	CCR		Description
	I	UI	
1	0	—	All interrupts are accepted Priority is given to interrupt sources with priority level 1
	1	—	Only NMI is accepted
0	0	—	All interrupts are accepted Priority is given to interrupt sources with priority level 1
	1	0	Only NMI and interrupt sources with priority level 1 are accepted
		1	Only NMI is accepted

When two or more interrupts occur simultaneously, if all are assigned to the same priority level, the highest-priority interrupt is selected according to an intrinsic interrupt priority order, and the lower-priority interrupts are held pending. When an interrupt occurs the CPU references the stack pointer, saves PC and CCR onto the stack, then fetches a starting address from the exception vector table and starts executing an interrupt-handling routine from that address.

Priority levels are assigned in interrupt priority registers A and B (IPRA and IPRB). Word access is permitted to IPRA and IPRB, which simplifies priority settings. A specific set of interrupts can be temporarily enabled or masked by making appropriate settings in IPRA and IPRB, then clearing the UE bit to 0, setting the I bit to 1, and clearing the UI bit to 0.

## Block Diagram



**Interrupt Controller Block Diagram**

# Interrupt Vector Table

Priority	Interrupt Source	Origin	Vector Number	Vector Address*	IPR	
High	NMI	External pins	7	H'001C to H'001F	—	
	IRQ <sub>0</sub>		12	H'0030 to H'0033	IPRA7	
	IRQ <sub>1</sub>		13	H'0034 to H'0037	IPRA6	
	IRQ <sub>2</sub>		14	H'0038 to H'003B	IPRA5	
	IRQ <sub>3</sub>		15	H'003C to H'003F		
	IRQ <sub>4</sub>		16	H'0040 to H'0043	IPRA4	
	Reserved	—	17	H'0044 to H'0047		
			18	H'0048 to H'004B		
	19	H'004C to H'004F				
	WOVI (interval timer)	WDT	20	H'0050 to H'0053	IPRA3	
	CMI (compare match)	Refresh controller	21	H'0054 to H'0057		
	Reserved		—	22	H'0058 to H'005B	
				23	H'005C to H'005F	
	IMIA <sub>0</sub> (compare match/input capture A0)	ITU channel 0	24	H'0060 to H'0063	IPRA2	
	IMIB <sub>0</sub> (compare match/input capture B0)		25	H'0064 to H'0067		
	OVI <sub>0</sub> (overflow 0)		26	H'0068 to H'006B		
Reserved	—	27	H'006C to H'006F			
IMIA <sub>1</sub> (compare match/input capture A1)	ITU channel 1	28	H'0070 to H'0073	IPRA1		
IMIB <sub>1</sub> (compare match/input capture B1)		29	H'0074 to H'0077			
OVI <sub>1</sub> (overflow 1)		30	H'0078 to H'007B			
Reserved	—	31	H'007C to H'007F			
IMIA <sub>2</sub> (compare match/input capture A2)	ITU channel 2	32	H'0080 to H'0083	IPRA0		
IMIB <sub>2</sub> (compare match/input capture B2)		33	H'0084 to H'0087			
OVI <sub>2</sub> (overflow 2)		34	H'0088 to H'008B			
Reserved	—	35	H'008C to H'008F			
IMIA <sub>3</sub> (compare match/input capture A3)	ITU channel 3	36	H'0090 to H'0093	IPRB7		
IMIB <sub>3</sub> (compare match/input capture B3)		37	H'0094 to H'0097			
OVI <sub>3</sub> (overflow 3)		38	H'0098 to H'009B			
Reserved	—	39	H'009C to H'009F			
IMIA <sub>4</sub> (compare match/input capture A4)	ITU channel 4	40	H'00A0 to H'00A3	IPRB6		
IMIB <sub>4</sub> (compare match/input capture B4)		41	H'00A4 to H'00A7			
OVI <sub>4</sub> (overflow 4)		42	H'00A8 to H'00AB			
Reserved	—	43	H'00AC to H'00AF			
DEND <sub>0A</sub>	DMAC	44	H'00B0 to H'00B3	IPRB5		
DEND <sub>0B</sub>		45	H'00B4 to H'00B7			
DEND <sub>1A</sub>		46	H'00B8 to H'00BB			
DEND <sub>1B</sub>		47	H'00BC to H'00BF			
Reserved	—	48	H'00C0 to H'00C3	—		
		49	H'00C4 to H'00C7			
		50	H'00C8 to H'00CB			
		51	H'00CC to H'00CF			
		52	H'00D0 to H'00D3			
ERI <sub>0</sub> (receive error 0)	SCI channel 0			IPRB3		
RXI <sub>0</sub> (receive data full 0)		53	H'00D4 to H'00D7			
TXI <sub>0</sub> (transmit data empty 0)		54	H'00D8 to H'00DB			
TEI <sub>0</sub> (transmit end 0)		55	H'00DC to H'00DF			
ERI <sub>1</sub> (receive error 1)	SCI channel 1			IPRB2		
RXI <sub>1</sub> (receive data full 1)		56	H'00E0 to H'00E3			
TXI <sub>1</sub> (transmit data empty 1)		57	H'00E4 to H'00E7			
TEI <sub>1</sub> (transmit end 1)		58	H'00E8 to H'00EB			
			59	H'00EC to H'00EF		
Low	ADI (A/D end)	A/D	60	H'00F0 to H'00F3	IPRB1	

Note: \* Lower 16 bits of the address.

## Section 5 Operating Modes

The H8/3048 Series has seven operating modes that are selected by the mode pins ( $MD_2$  to  $MD_0$ ). The input at these pins determines the size of the address space and the initial bus mode.

After program execution begins, the bus controller can select an 8-bit or 16-bit data bus width for each area individually. Both the address space and the pin functions differ depending on the operating mode.

### 5.1 Expanded Modes with On-Chip ROM Disabled (Modes 1 to 4)

**Mode 1 (Expanded 1-Mbyte Mode with 8-Bit Bus and On-Chip ROM Disabled):** Address pins  $A_{19}$  to  $A_0$  are enabled, permitting access to a maximum 1-Mbyte address space. The initial bus mode after a reset is 8 bits, with 8-bit access to all areas. If at least one area is designated for 16-bit access by the bus controller, the bus mode switches to 16 bits.

**Mode 2 (Expanded 1-Mbyte Mode with 16-Bit Bus and On-Chip ROM Disabled):** Address pins  $A_{19}$  to  $A_0$  are enabled, permitting access to a maximum 1-Mbyte address space. The initial bus mode after a reset is 16 bits, with 16-bit access to all areas. If all areas are designated for 8-bit access by the bus controller, the bus mode switches to 8 bits.

**Mode 3 (Expanded 16-Mbyte Mode with 8-Bit Bus and On-Chip ROM Disabled):** Address pins  $A_{23}$  to  $A_0$  are enabled, permitting access to a maximum 16-Mbyte address space. The initial bus mode after a reset is 8 bits, with 8-bit access to all areas. If at least one area is designated for 16-bit access by the bus controller, the bus mode switches to 16 bits.

**Mode 4 (Expanded 16-Mbyte Mode with 16-Bit Bus and On-Chip ROM Disabled):** Address pins  $A_{23}$  to  $A_0$  are enabled, permitting access to a maximum 16-Mbyte address space. The initial bus mode after a reset is 16 bits, with 16-bit access to all areas. If all areas are designated for 8-bit access by the bus controller, the bus mode switches to 8 bits.

## 5.2 Expanded Modes with On-Chip ROM Enabled (Modes 5 and 6)

### Mode 5 (Expanded 1-Mbyte Mode with 8-Bit Bus and On-Chip ROM Enabled):

The on-chip ROM is enabled. Address pins A<sub>19</sub> to A<sub>0</sub> are enabled, permitting access to a maximum 1-Mbyte address space. The initial bus mode after a reset is 8 bits, with 8-bit access to all areas. If at least one area is designated for 16-bit access by the bus controller, the bus mode switches to 16 bits.

### Mode 6 (Expanded 16-Mbyte Mode with 8-Bit Bus and On-Chip ROM Enabled):

The on-chip ROM is enabled. Address pins A<sub>23</sub> to A<sub>0</sub> are enabled, permitting access to a maximum 16-Mbyte address space. The initial bus mode after a reset is 8 bits, with 8-bit access to all areas. If at least one area is designated for 16-bit access by the bus controller, the bus mode switches to 16 bits.

## 5.3 Single-Chip Mode (Mode 7)

**Mode 7 (Single-Chip 1-Mbyte Mode):** This mode runs all operations from on-chip ROM (PROM or masked ROM) and RAM. The address-space size is 1 Mbyte, but off-chip addresses cannot be accessed.

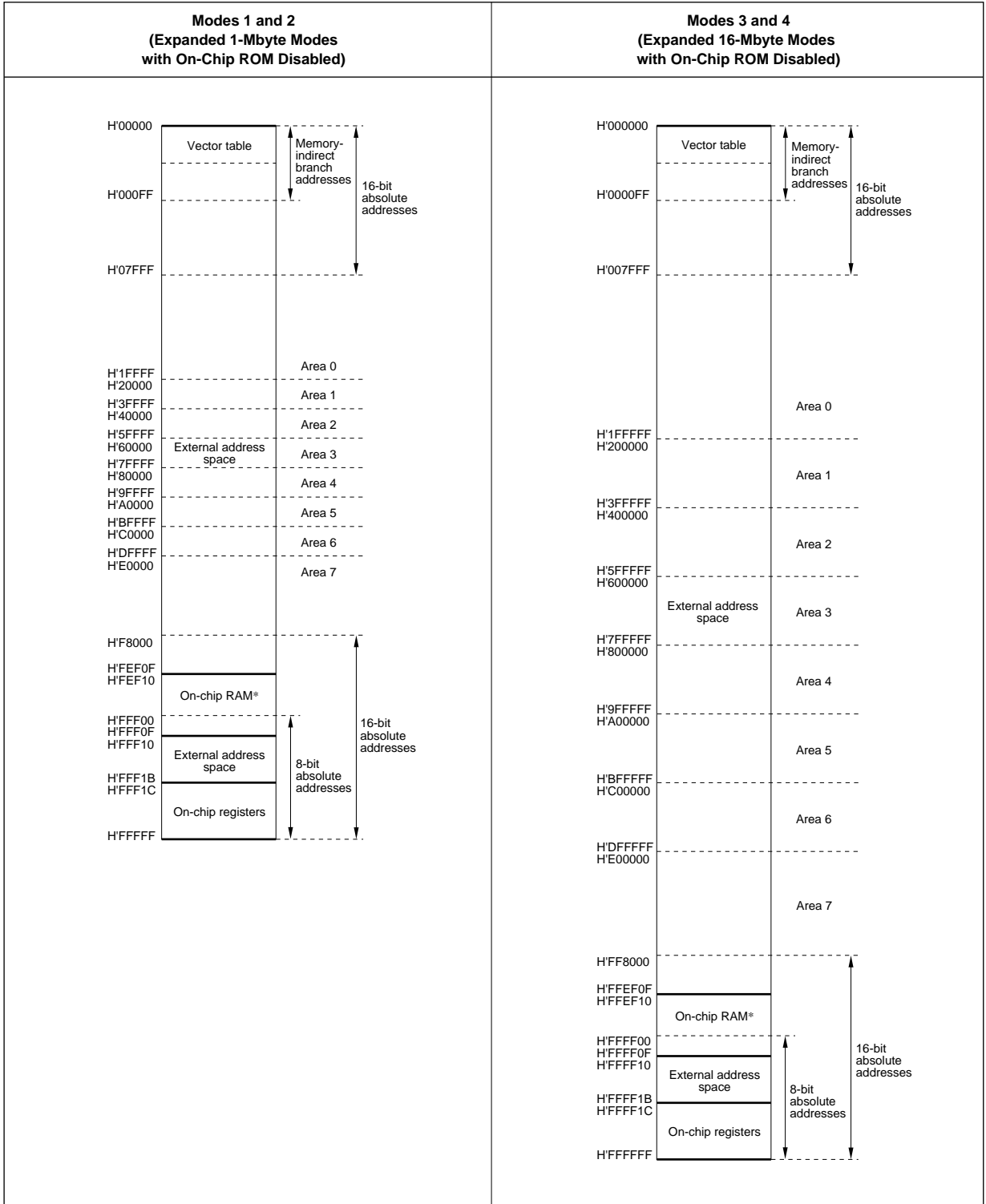
### Operating Mode Summary

Mode	Mode Pins			Address Space	Address Bus	Initial Bus Mode	Data Bus	On-Chip ROM	On-Chip RAM
	MD <sub>2</sub>	MD <sub>1</sub>	MD <sub>0</sub>						
Mode 1	0	0	1	1 Mbyte	A <sub>19</sub> to A <sub>0</sub>	8 bits	D <sub>15</sub> to D <sub>8</sub>	Disabled	Enabled*
Mode 2	0	1	0	1 Mbyte	A <sub>19</sub> to A <sub>0</sub>	16 bits	D <sub>15</sub> to D <sub>0</sub>	Disabled	Enabled*
Mode 3	0	1	1	16 Mbytes	A <sub>23</sub> to A <sub>0</sub>	8 bits	D <sub>15</sub> to D <sub>8</sub>	Disabled	Enabled*
Mode 4	1	0	0	16 Mbytes	A <sub>23</sub> to A <sub>0</sub>	16 bits	D <sub>15</sub> to D <sub>0</sub>	Disabled	Enabled*
Mode 5	1	0	1	1 Mbyte	A <sub>19</sub> to A <sub>0</sub>	8 bits	D <sub>15</sub> to D <sub>8</sub>	Enabled	Enabled*
Mode 6	1	1	0	16 Mbytes	A <sub>23</sub> to A <sub>0</sub>	8 bits	D <sub>15</sub> to D <sub>8</sub>	Enabled	Enabled*
Mode 7	1	1	1	1 Mbyte	—	—	—	Enabled	Enabled

Note: \* Can be switched to external address space.

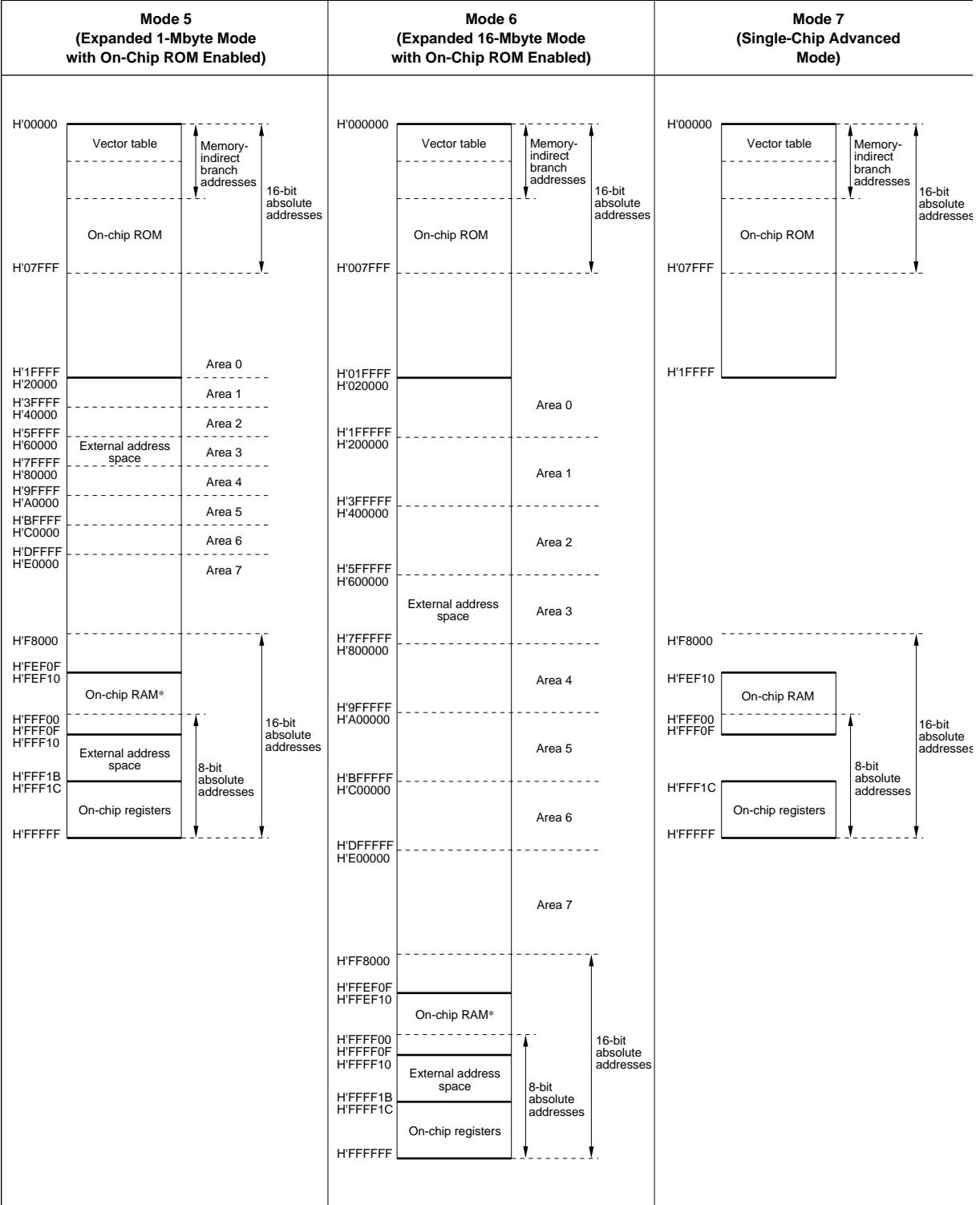
# 5.4 Memory Maps

## H8/3048 Memory Maps in Each Operating Mode (1)



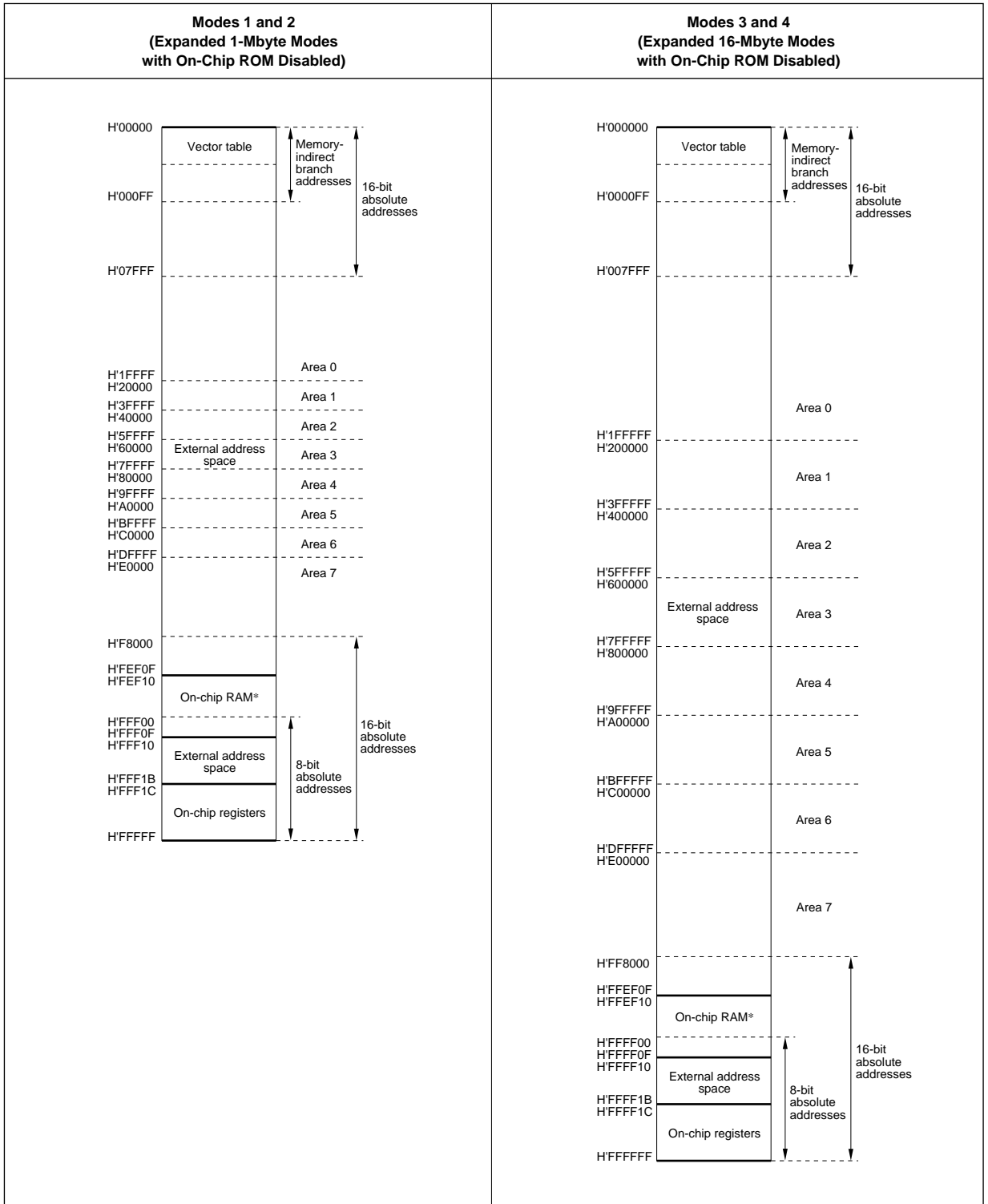
Note: \* External address space if on-chip RAM is disabled.

# H8/3048 Memory Maps in Each Operating Mode (2)



Note: \* External address space if on-chip RAM is disabled.

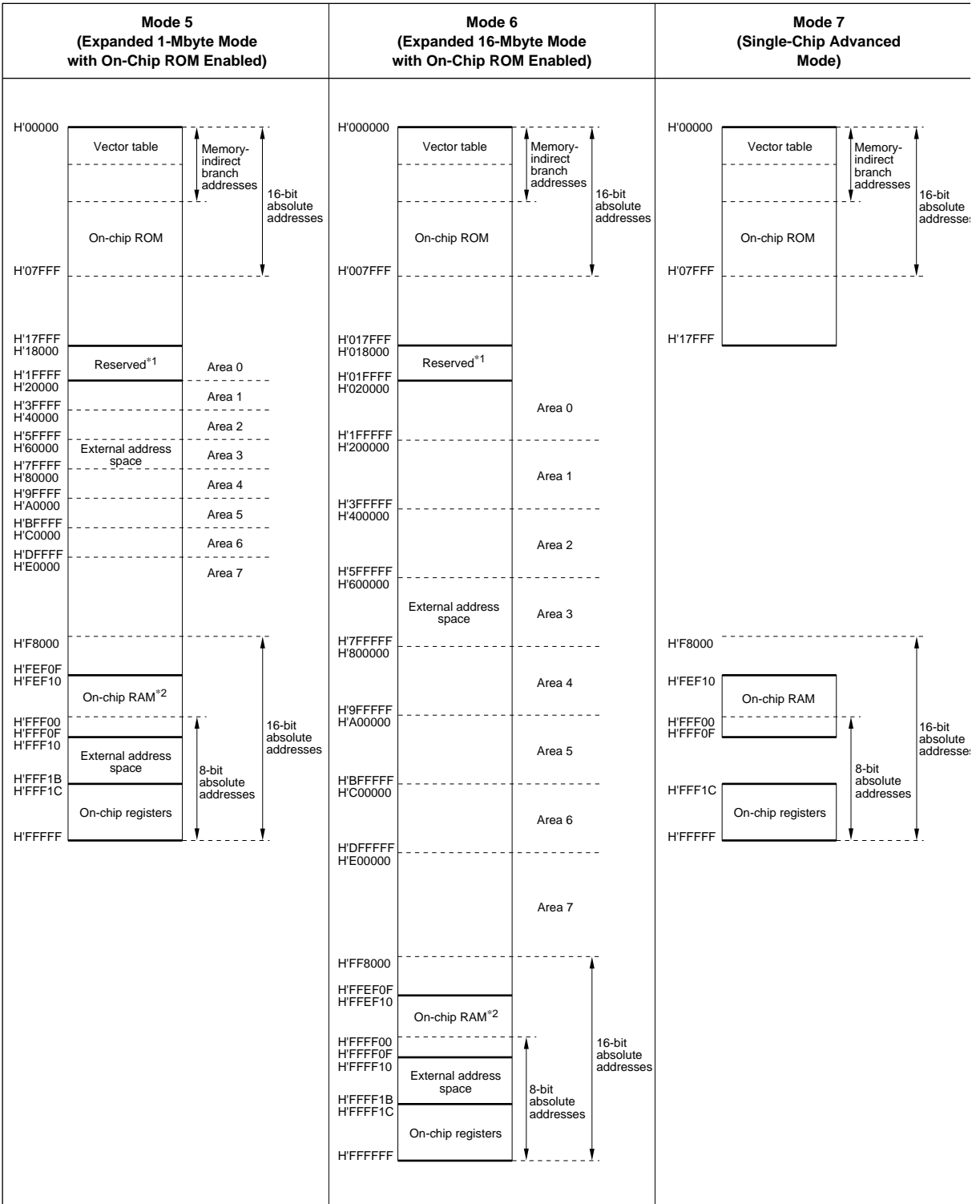
# H8/3047 Memory Maps in Each Operating Mode (1)



Note: \* External address space if on-chip RAM is disabled.

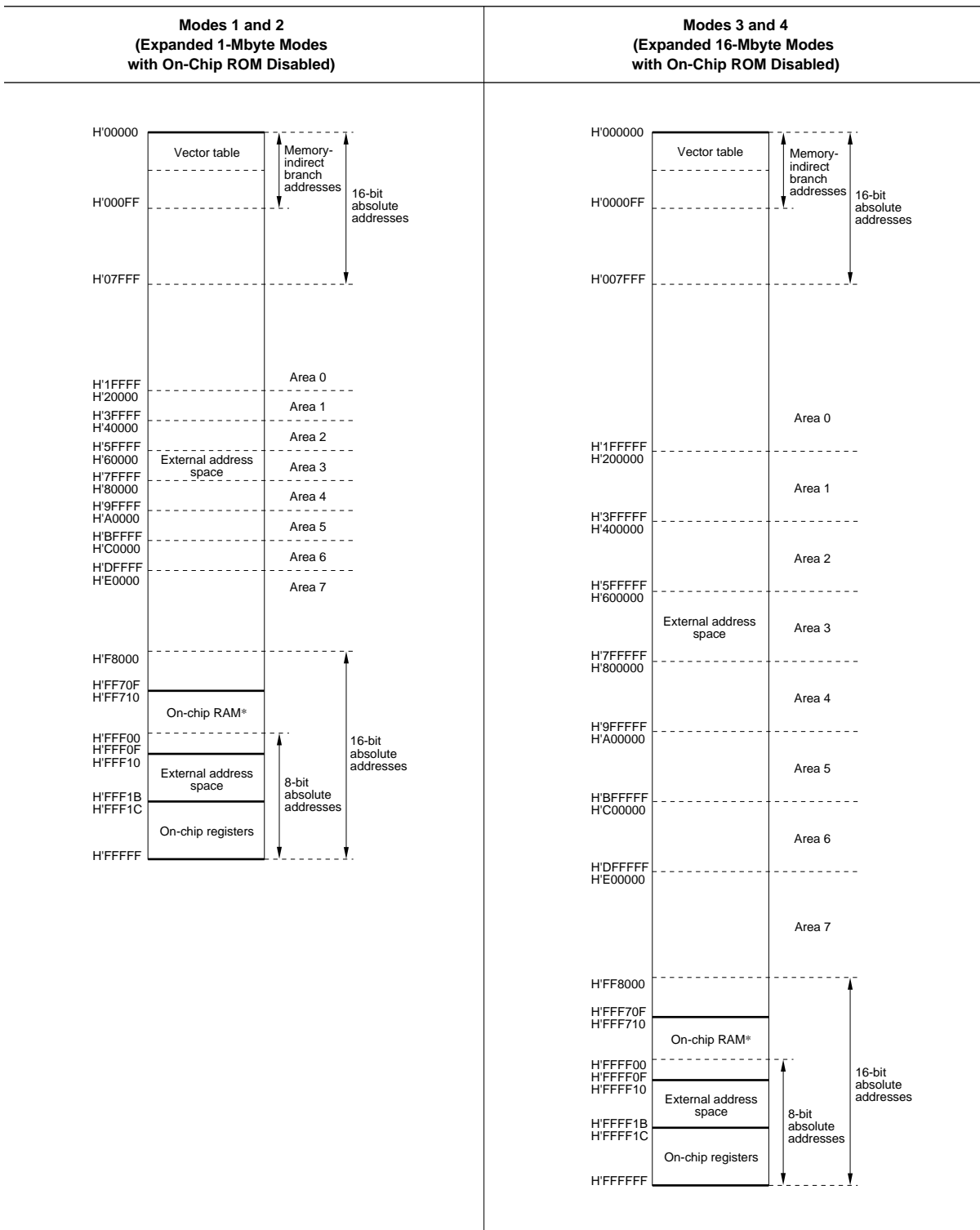


## H8/3047 Memory Maps in Each Operating Mode (2)



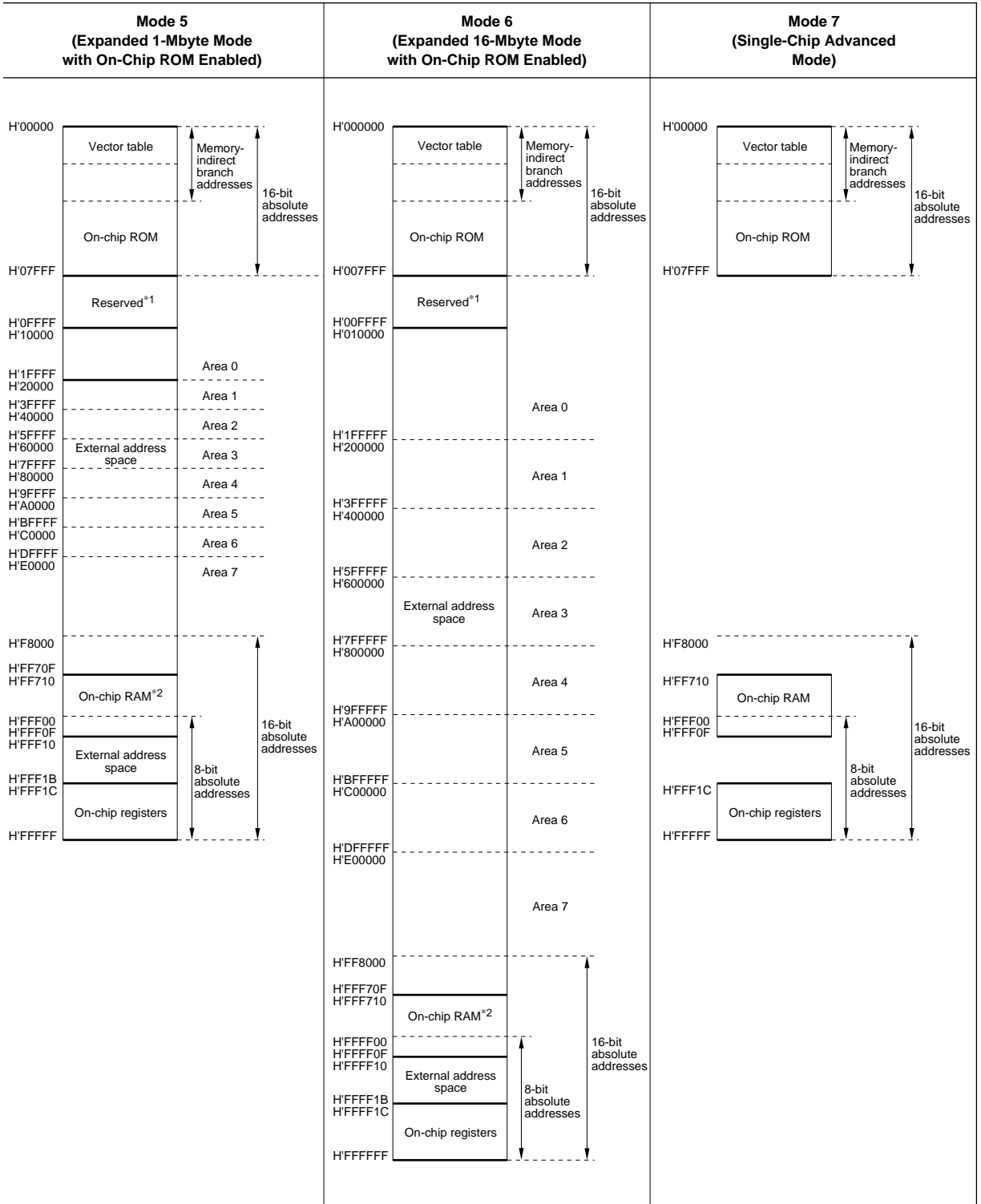
- Notes: 1. Do not access the reserved area.  
 2. External address space if on-chip RAM is disabled.

# H8/3044 Memory Maps in Each Operating Mode (1)



ote: \* External address space if on-chip RAM is disabled.

## H8/3044 Memory Maps in Each Operating Mode (2)



- Notes: 1. Do not access the reserved area.  
 2. External address space if on-chip RAM is disabled.

# Section 6 Supporting Modules

## 6.1 Bus Controller

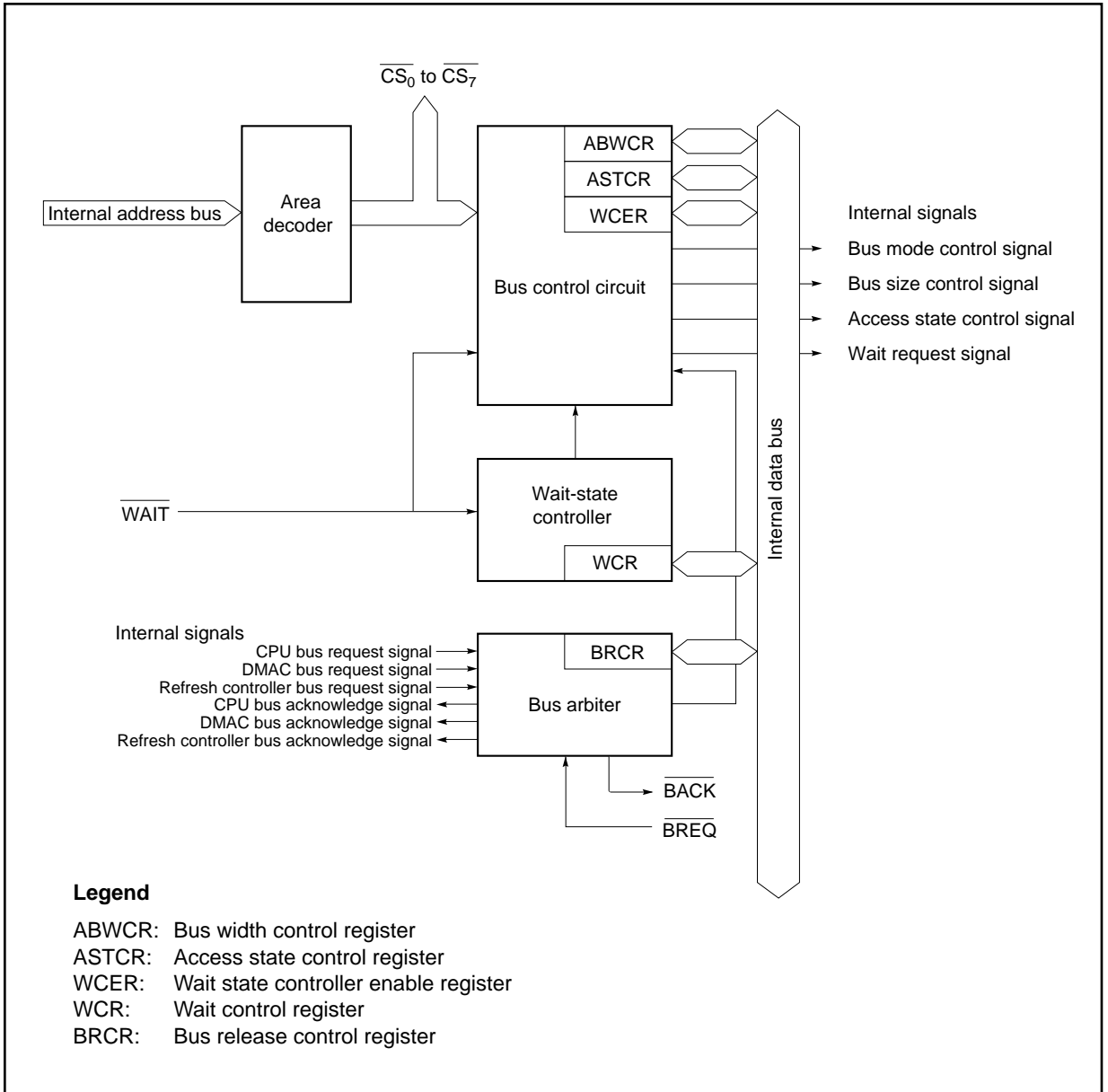
### Functions

The on-chip bus controller divides the address space into eight areas and can assign different bus control specifications to each. This enables different types of memory to be connected easily. The bus controller also has a bus arbitration function that controls the operation of the DMA controller (DMAC) and refresh controller, and can release the bus to an external device.

### Features

- Independent settings for eight address areas
  - 128-kbyte areas in 1-Mbyte modes
  - 2-Mbyte areas in 16-Mbyte modes
  - Output of chip select signals ( $\overline{CS}_0$  to  $\overline{CS}_7$ )
  - Selection of 8-bit or 16-bit access
  - Selection of two-state or three-state access
- Four wait modes
  - Selection of programmable wait mode, pin auto-wait mode, and pin wait modes 0 and 1
  - Automatic insertion of zero to three wait states
- Bus arbitration function
  - Bus right granted to the CPU, DMAC, refresh controller, or an external bus master by the built-in bus arbiter

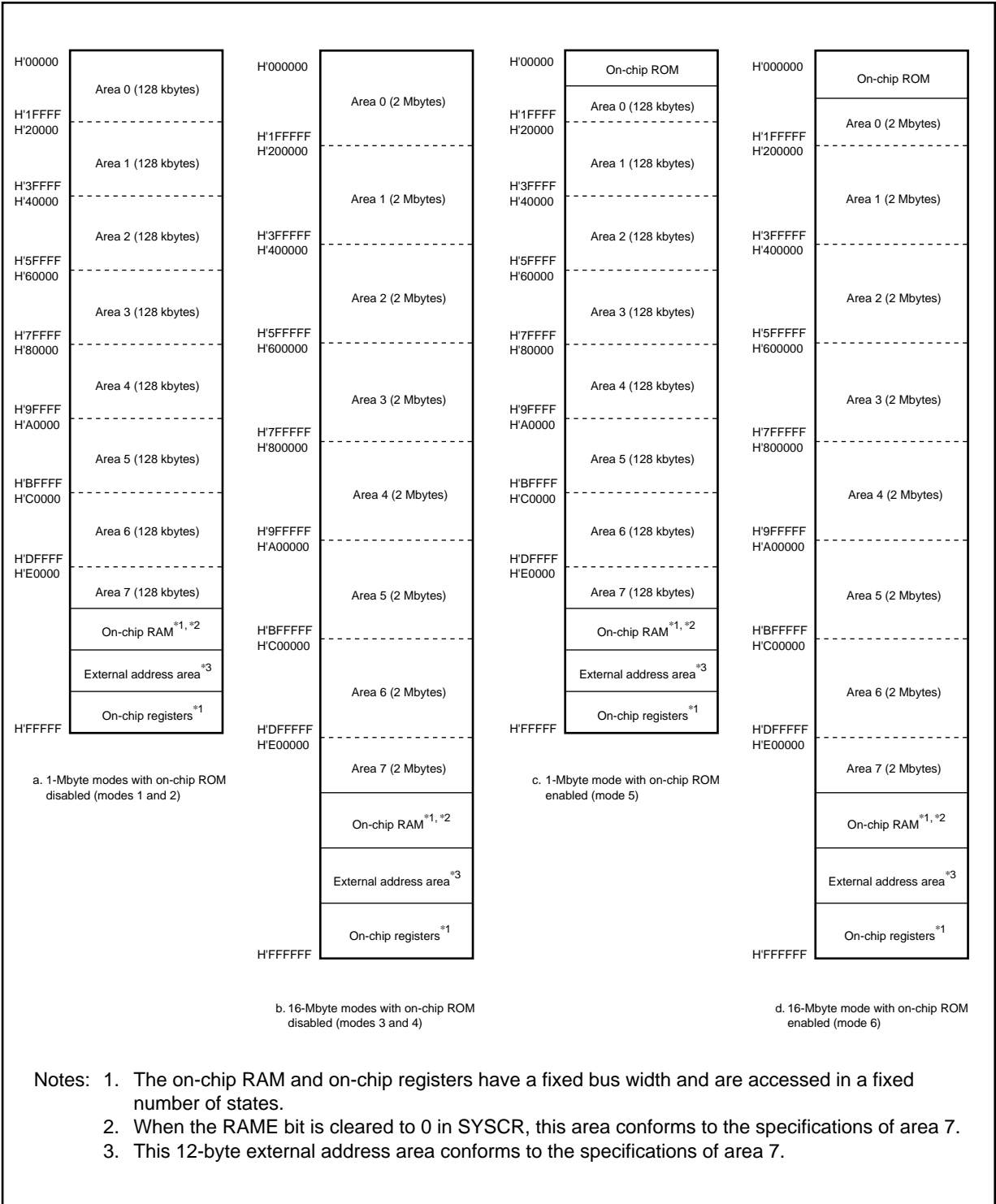
# Block Diagram



**Block Diagram of Bus Controller**

# Area Subdivision

The external address space is divided into areas 0 to 7. Each area has a size of 128 kbytes in the 1-Mbyte modes, or 2 Mbytes in the 16-Mbyte modes. The memory map is shown next.



- Notes:
1. The on-chip RAM and on-chip registers have a fixed bus width and are accessed in a fixed number of states.
  2. When the RAME bit is cleared to 0 in SYSCR, this area conforms to the specifications of area 7.
  3. This 12-byte external address area conforms to the specifications of area 7.

## Access Area Map for Modes 1 to 6

The bus specifications for each area can be selected in ABWCR, ASTCR, WCER, and WCR as shown in the following table.

## Bus Specifications

ABWCR	ASTCR	WCER	WCR		Bus Specifications			
ABWn	ASTn	WCEn	WMS1	WMS0	Bus Width	Access States	Wait Mode	
0	0	—	—	—	16	2	Wait disabled	
	1	0	—	—	16	3	Pin wait mode 0	
		1	0	0	0	16	3	Programmable wait mode
				1	1	16	3	Wait disabled
		1	0	0	0	16	3	Pin wait mode 1
				1	1	16	3	Pin auto-wait mode
1	0	—	—	—	8	2	Wait disabled	
	1	0	—	—	8	3	Pin wait mode 0	
		1	0	0	0	8	3	Programmable wait mode
				1	1	8	3	Wait disabled
		1	0	0	0	8	3	Pin wait mode 1
				1	1	8	3	Pin auto-wait mode

Note: n = 0 to 7

## Chip Select Signals

A chip select signal ( $\overline{CS}_0$  to  $\overline{CS}_7$ ) can be output for each of areas 0 to 7.

## Data Bus

Each of areas 0 to 7 can be designated as an 8-bit-access area or a 16-bit-access area.

## Number of Access States

Each of areas 0 to 7 can be designated as a two-state-access area or a three-state-access area.

## Wait Modes

Four wait modes can be selected for each area: pin wait mode 0 or 1, pin auto-wait mode, or programmable wait mode.

**Pin Wait Mode 0:** The wait state controller is disabled. Wait states ( $T_W$ ) can only be inserted by  $\overline{\text{WAIT}}$  pin control.

**Pin Wait Mode 1:** In all accesses to external three-state-access areas, the designated number of wait states ( $T_W$ ) are inserted. Additional wait states can be inserted by  $\overline{\text{WAIT}}$  pin control.

**Pin Auto-Wait Mode:** If the  $\overline{\text{WAIT}}$  pin is low, the designated number of wait states ( $T_W$ ) are inserted.

**Programmable Wait Mode:** The designated number of wait states ( $T_W$ ) are inserted in all accesses to external three-state-access areas.

## Bus Arbiter

The bus controller has a built-in bus arbiter that arbitrates between different bus masters. There are four bus masters: the CPU, DMAC, refresh controller, and an external bus master. These bus masters use the bus to carry out read, write, and refresh operations.

When bus request signals are received from one or more bus masters, at fixed times the bus arbiter determines priority and grants the use of the bus to one bus master.

The bus master priority order is:

(High) External bus master > refresh controller > DMAC > CPU (Low)



## 6.2 Refresh Controller

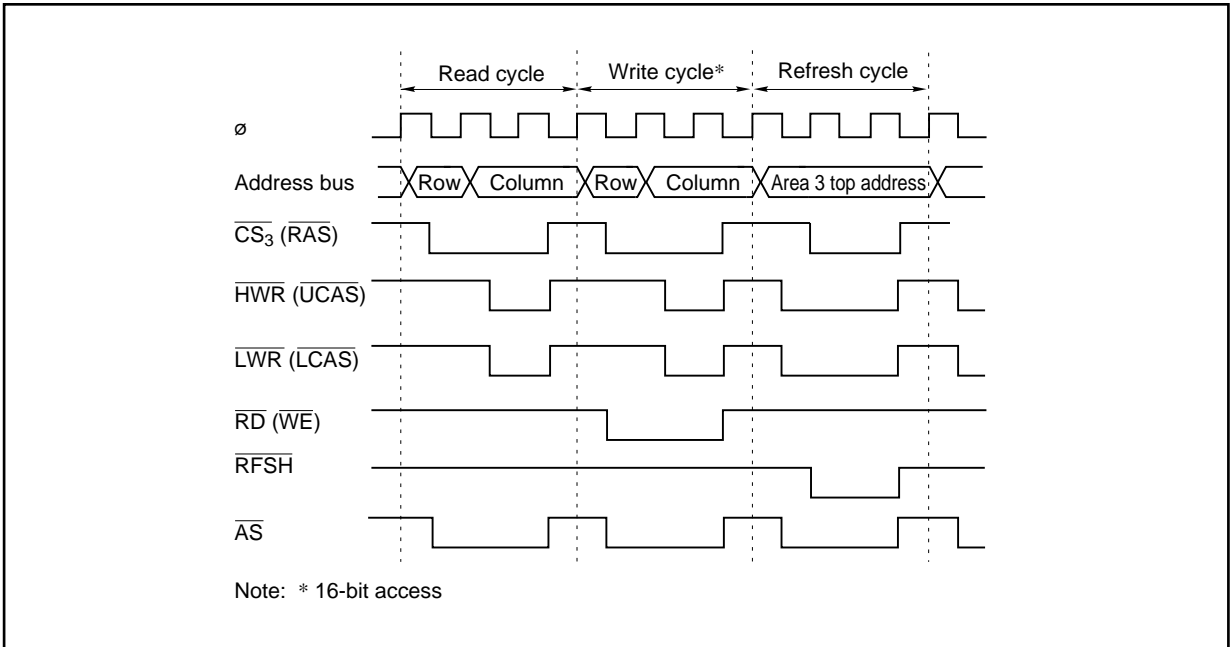
### Functions

The refresh controller can be used for one of three functions: DRAM refresh control, pseudo-static RAM refresh control, or as an interval timer.

### Features

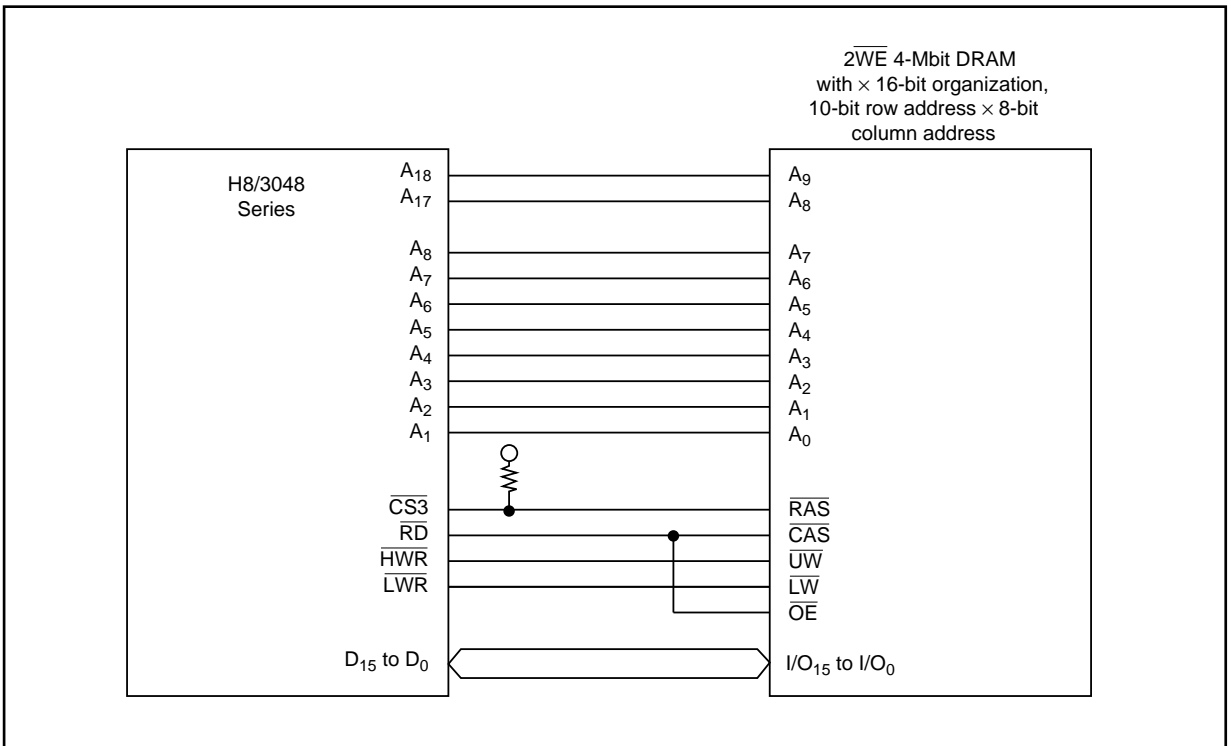
- Functions as a DRAM refresh controller
  - Enables direct connection of 16-bit-wide DRAM
  - Selection of  $2\overline{\text{CAS}}$  or  $2\overline{\text{WE}}$  mode
  - Selection of 8-bit or 9-bit column address multiplexing for DRAM address input
  - $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$  refresh control
  - Software-selectable refresh interval
  - Software-selectable self-refresh mode
  - Wait states can be inserted
- Functions as a pseudo-static RAM refresh controller
  - $\overline{\text{RFSH}}$  signal output for refresh control
  - Software-selectable refresh interval
  - Software-selectable self-refresh mode
  - Wait states can be inserted
- Functions as an interval timer
  - Refresh timer counter (RTCNT) can be used as an 8-bit up-counter
  - Selection of seven counter clock sources:  $\phi/2$ ,  $\phi/8$ ,  $\phi/32$ ,  $\phi/128$ ,  $\phi/512$ ,  $\phi/2048$ ,  $\phi/4096$
  - Interrupts can be generated by compare match between RTCNT and the refresh time constant register (RTCOR)

**DRAM Refresh Control:** The following diagrams show the interface timing for  $2\overline{\text{CAS}}$  DRAM and an interface circuit for  $2\overline{\text{WE}}$  DRAM.

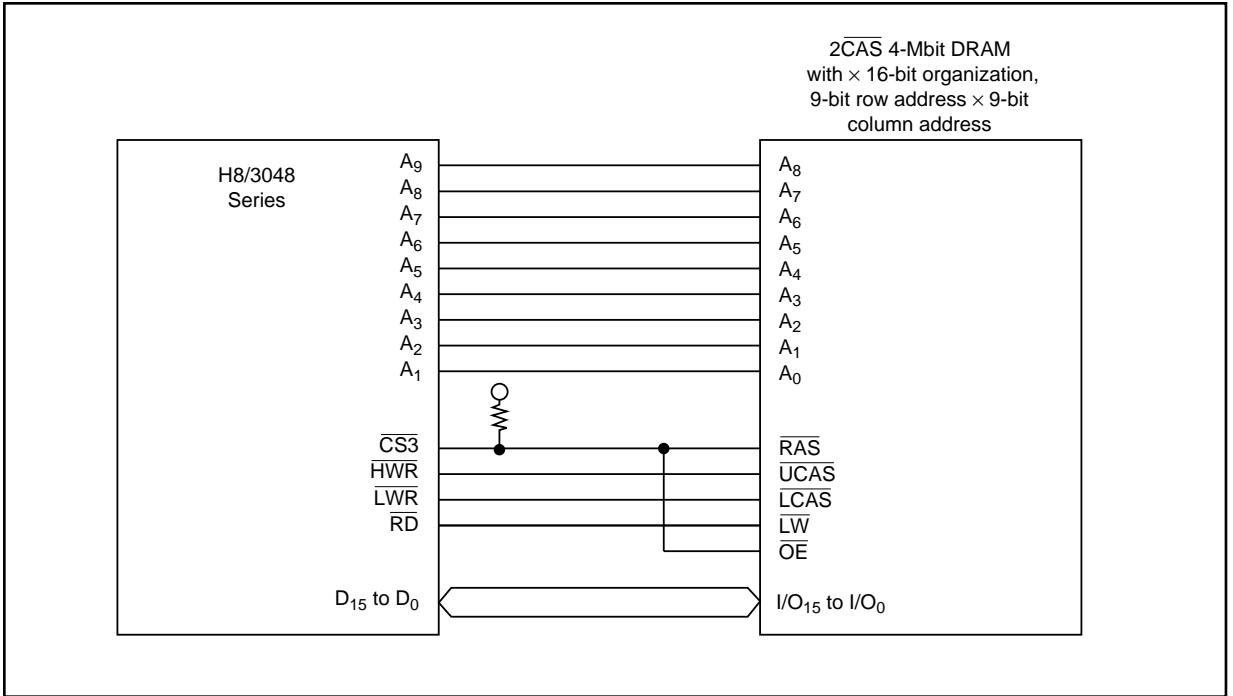


**DRAM Control Signal Output Timing ( $2\overline{\text{CAS}}$  Mode)**

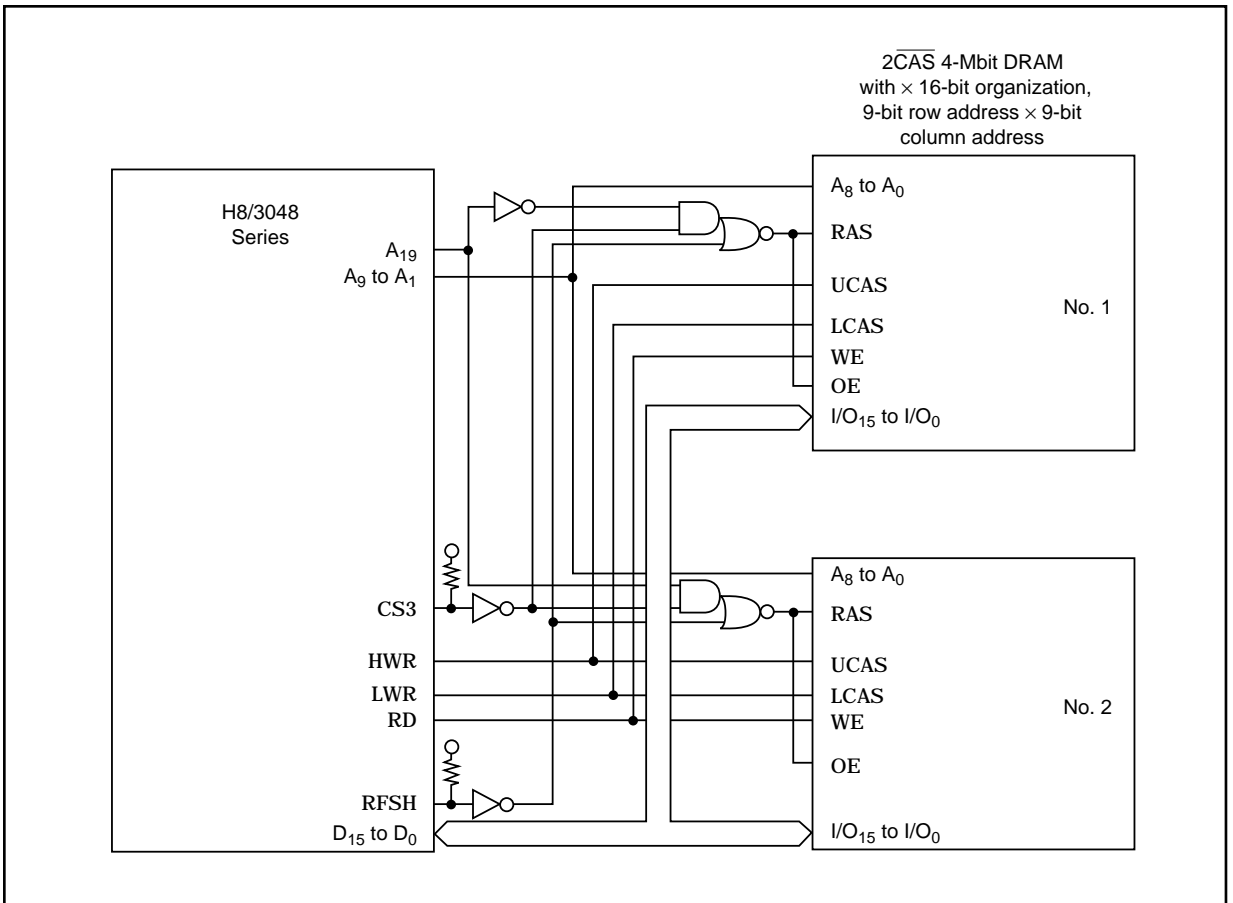
An example of an interconnection to  $2\overline{\text{WE}}$  4-Mbit DRAM is shown next.



**Interconnections for  $2\overline{\text{WE}}$  4-Mbit DRAM (Example in Expanded 16-Mbyte Mode)**



**Interconnections for 2CAS 4-Mbit DRAM (Example in Expanded 16-Mbyte Mode)**



**Interconnections for Two 4-Mbit DRAM Chips (Example in 16-Mbyte Mode)**

## 6.3 DMA Controller

### Functions

The DMA controller (DMAC) can transfer data on up to four channels (channels 0A, 0B, 1A, and 1B). Each channel can execute memory-I/O transfers independently in short address mode. Pairs of channels can combine to execute memory-memory transfers in full address mode.

### Features

- Selection of short address mode or full address mode

#### Short address mode

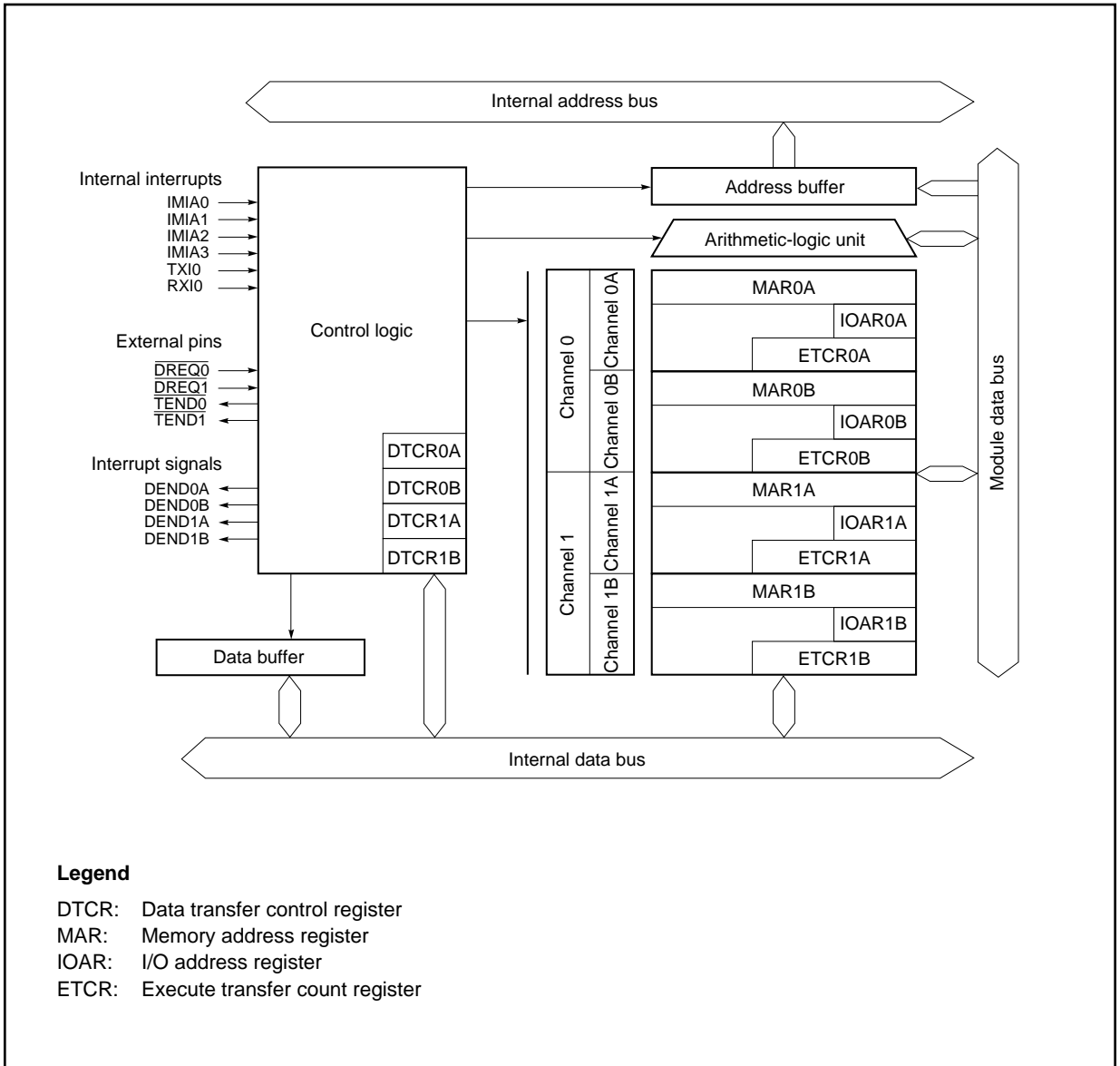
- 8-bit source address and 24-bit destination address, or vice versa
- Maximum four channels available
- Selection of I/O mode, idle mode, or repeat mode

#### Full address mode

- 24-bit source and destination addresses
- Maximum two channels available
- Selection of normal mode or block transfer mode

- Directly addressable 16-Mbyte address space
- Selection of byte or word transfer
- Activation by internal interrupts, external requests, or auto-request (depending on transfer mode)
  - 16-bit integrated timer unit (ITU) compare match/input capture interrupts (four)
  - Serial communication interface (SCI) transmit-data-empty/receive-data-full interrupts
  - External requests
  - Auto-request

# Block Diagram



## Legend

- DTCR: Data transfer control register
- MAR: Memory address register
- IOAR: I/O address register
- ETCR: Execute transfer count register

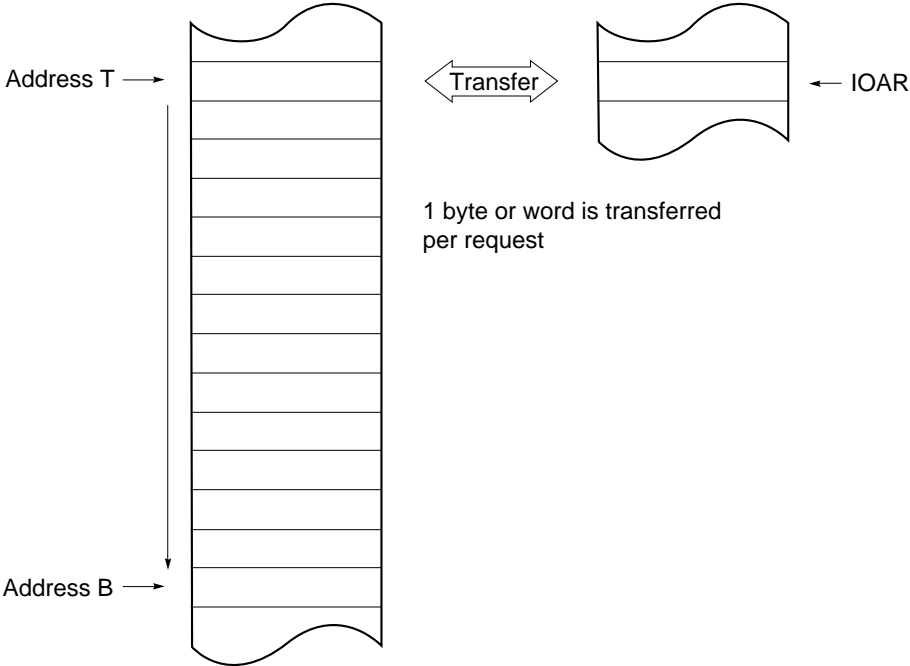
**Block Diagram of DMA Controller**

## Transfer Modes

The DMAC provides a choice of short and full address modes. The short address mode includes I/O mode, idle mode, and repeat mode, and permits independent transfers on channels A and B. The full address mode includes normal mode and block transfer mode, in which channels A and B are combined. The functions of the transfer modes are summarized as follows.

Transfer Mode	Activation	Address Reg. Length	
		Source	Destination
Short address mode	1. I/O mode <ul style="list-style-type: none"> <li>• Transfers one byte or one word per request</li> <li>• Increments or decrements the memory address by 1 or 2</li> <li>• Executes 1 to 65,536 transfers</li> </ul>		
	• Compare match/input capture A interrupts from ITU channels 0 to 3	24	8
	• Transmit-data-empty interrupt from SCI		
Short address mode	2. Idle mode <ul style="list-style-type: none"> <li>• Transfers one byte or one word per request</li> <li>• Holds the memory address fixed</li> <li>• Executes 1 to 65,536 transfers</li> </ul>		
	• Receive-data-full interrupt from SCI	8	24
Short address mode	3. Repeat mode <ul style="list-style-type: none"> <li>• Transfers one byte or one word per request</li> <li>• Increments or decrements the memory address by 1 or 2</li> <li>• Executes a specified number (1 to 256) of transfers, then returns to the initial state and continues</li> </ul>		
	• External request	24	8
Full address mode	1. Normal mode <ul style="list-style-type: none"> <li>• Auto-request</li> <li>• External request</li> <li>— Retains the transfer request internally</li> <li>— Executes a specified number (1 to 65,536) of transfers continuously</li> <li>— Selection of burst mode or cycle-steal mode</li> <li>• External request</li> <li>— Transfers one byte or one word per request</li> <li>— Executes 1 to 65,536 transfers</li> </ul>		
		24	24
Full address mode	2. Block transfer mode <ul style="list-style-type: none"> <li>• Transfers one block of a specified size per request</li> <li>• Executes 1 to 65,536 transfers</li> <li>• Allows either the source or destination to be a fixed block area</li> <li>• Block size can be 1 to 256 bytes or words</li> </ul>		
	• Compare match/input capture A interrupts from ITU channels 0 to 3	24	24
	• External request		

# Operation in I/O Mode

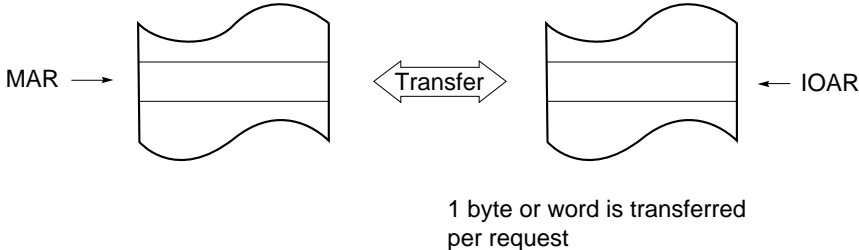


**Legend**

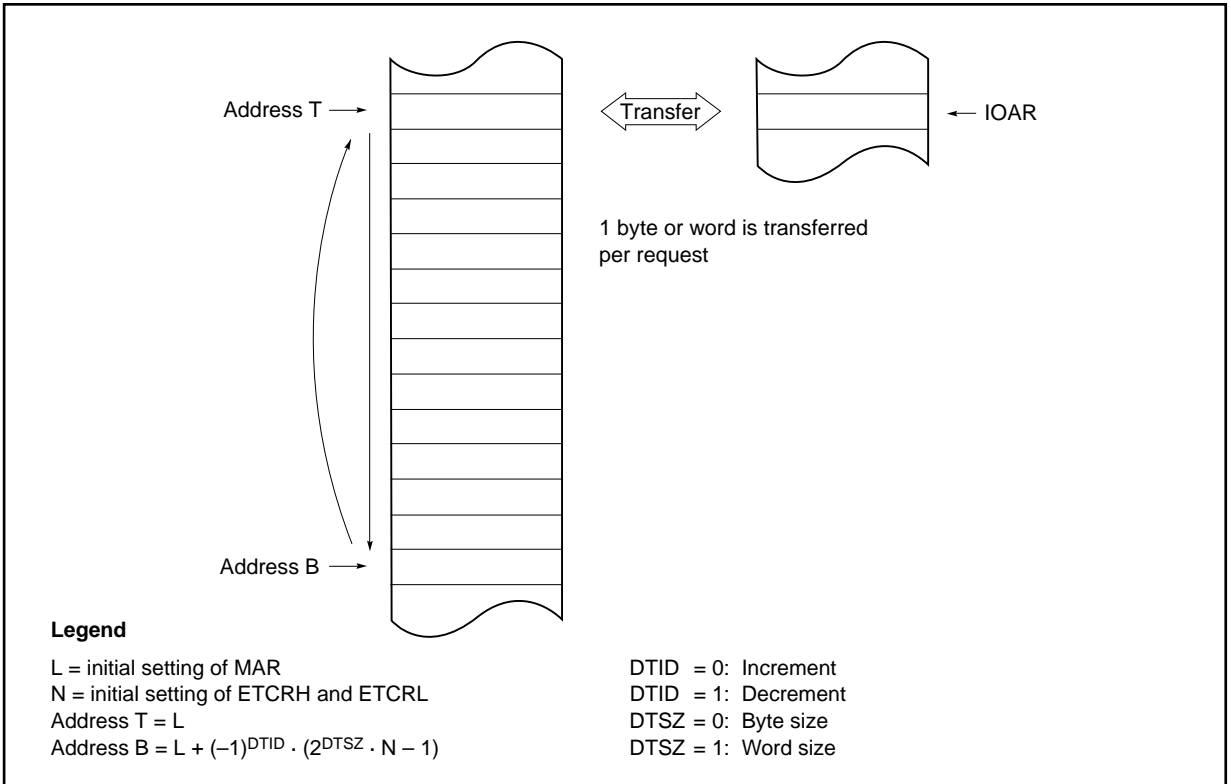
L = initial setting of memory address register (MAR)  
 N = initial setting of execute transfer count register (ETCR)  
 Address T = L  
 Address B =  $L + (-1)^{DTID} \cdot (2^{DTSZ} \cdot N - 1)$

DTID = 0: Increment  
 DTID = 1: Decrement  
 DTSZ = 0: Byte size  
 DTSZ = 1: Word size

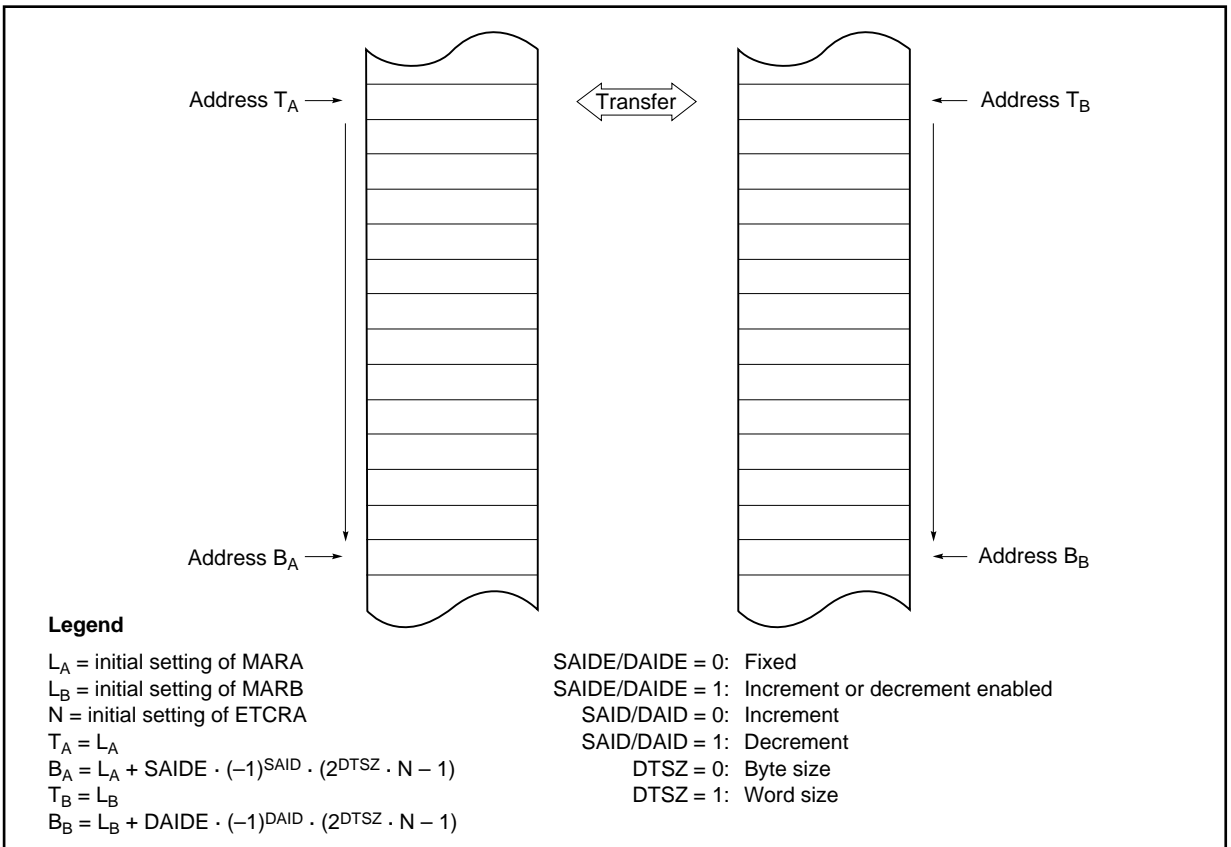
# Operation in Idle Mode



## Operation in Repeat Mode

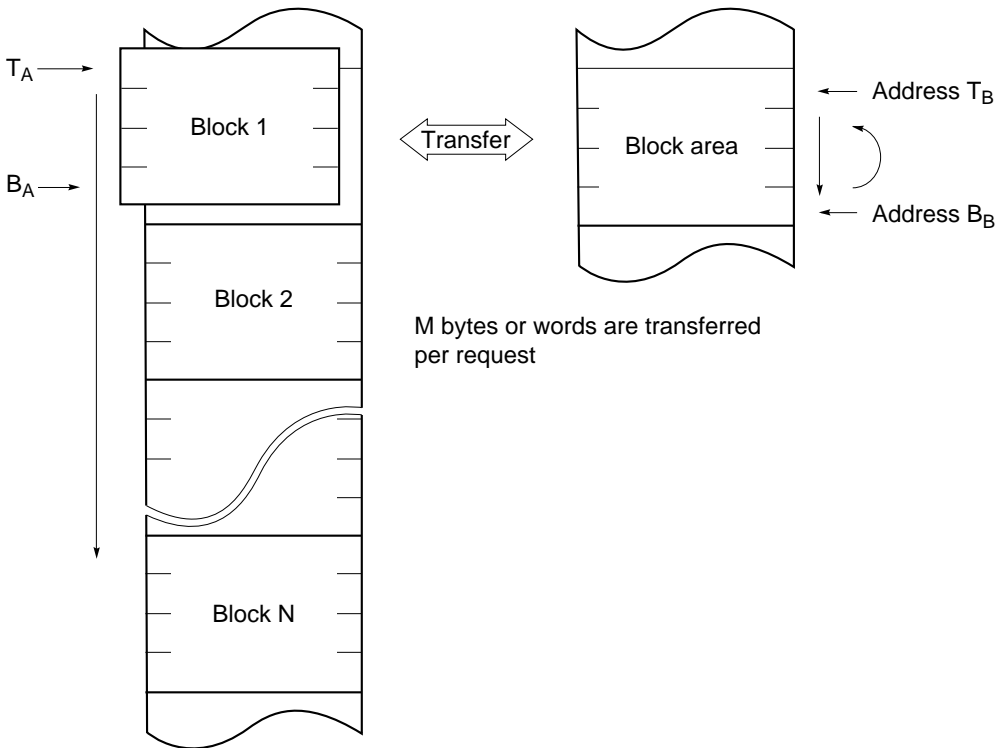


## Operation in Normal Mode





## Operation in Block Transfer Mode



### Legend

$L_A$  = initial setting of MARA

$L_B$  = initial setting of MARB

$M$  = initial setting of ETCRAH and ETCRAL

$N$  = initial setting of ETCRB

$T_A = L_A$

$B_A = L_A + SAIDE \cdot (-1)^{SAID} \cdot (2^{DTSZ} \cdot M - 1)$

$T_B = L_B$

$B_B = L_B + DAIDE \cdot (-1)^{DAID} \cdot (2^{DTSZ} \cdot M - 1)$

SAIDE/DAIDE = 0: Fixed

SAIDE/DAIDE = 1: Increment or decrement enabled

SAID/DAID = 0: Increment

SAID/DAID = 1: Decrement

DTSZ = 0: Byte size

DTSZ = 1: Word size

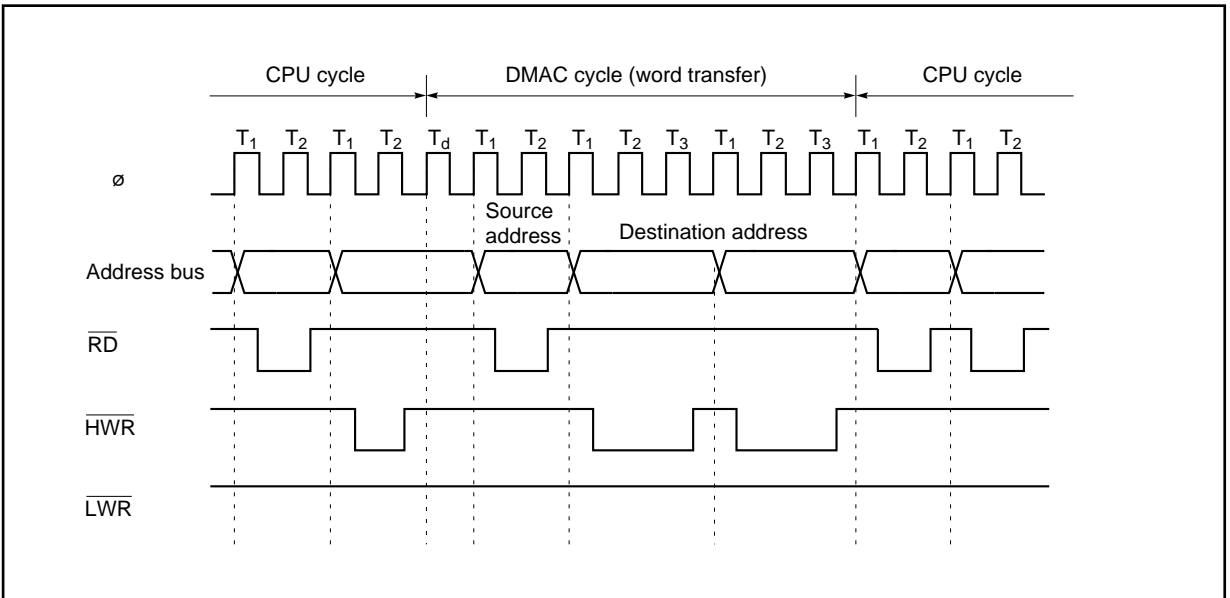
## DMAC Activation

The DMAC can be activated by an internal interrupt, external request, or auto-request. The available activation sources differ depending on the transfer mode and channel as follows.

Activation Source		Short Address Mode		Full Address Mode	
		Channels 0A and 1A	Channels 0B and 1B	Normal	Block
Internal interrupts	IMIA0		○	×	○
	IMIA1		○	×	○
	IMIA2		○	×	○
	IMIA3		○	×	○
	TXI0		○	×	×
	RXI0		○	×	×
External requests	Falling edge of DREQ	×	○	○	○
	Low input at DREQ	×	○	○	×
Auto-request			×	○	×

## DMAC Bus Cycle

The following diagram shows an example of the timing of the basic DMAC bus cycle for a word-size transfer from a 16-bit two-state-access area to an 8-bit three-state-access area.



**DMA Transfer Bus Timing (Example)**

## 6.4 16-Bit Integrated Timer Unit (ITU)

### Functions

The H8/3048 Series has a built-in 16-bit integrated timer unit (ITU) with five 16-bit timer channels. The ITU can process up to 12 pulse outputs, or up to 10 pulse inputs. It can also provide complementary PWM output, measure pulse width, process input from a two-phase encoder, and activate the programmable timing pattern controller (TPC) and DMA controller (DMAC).

### Features

- Capability to process up to 12 pulse outputs or 10 pulse inputs
- Ten general registers (GRs, two per channel) with independently-assignable output compare or input capture functions
- Selection of eight counter clock sources for each channel:
  - Internal clocks:  $\phi$ ,  $\phi/2$ ,  $\phi/4$ ,  $\phi/8$
  - External clocks: TCLKA, TCLKB, TCLKC, TCLKD
- Five operating modes selectable in all channels:
  - Waveform output by compare match
    - Selection of 0 output, 1 output, or toggle output (only 0 or 1 output in channel 2)
  - Input capture function
    - Rising edge, falling edge, or both edges (selectable)
  - Counter clearing function
    - Counters can be cleared by compare match or input capture
  - Synchronization
    - Two or more timer counters (TCNTs) can be preset simultaneously, or cleared simultaneously by compare match or input capture. Counter synchronization enables synchronous register input and output.

— PWM mode

PWM output can be provided with an arbitrary duty cycle. With synchronization, up to five-phase PWM output is possible

- Phase counting mode selectable in channel 2  
Two-phase encoder output can be counted automatically.

- Three additional modes selectable in channels 3 and 4

— Reset-synchronized PWM mode

If channels 3 and 4 are combined, three-phase PWM output is possible with three pairs of complementary waveforms.

— Complementary PWM mode

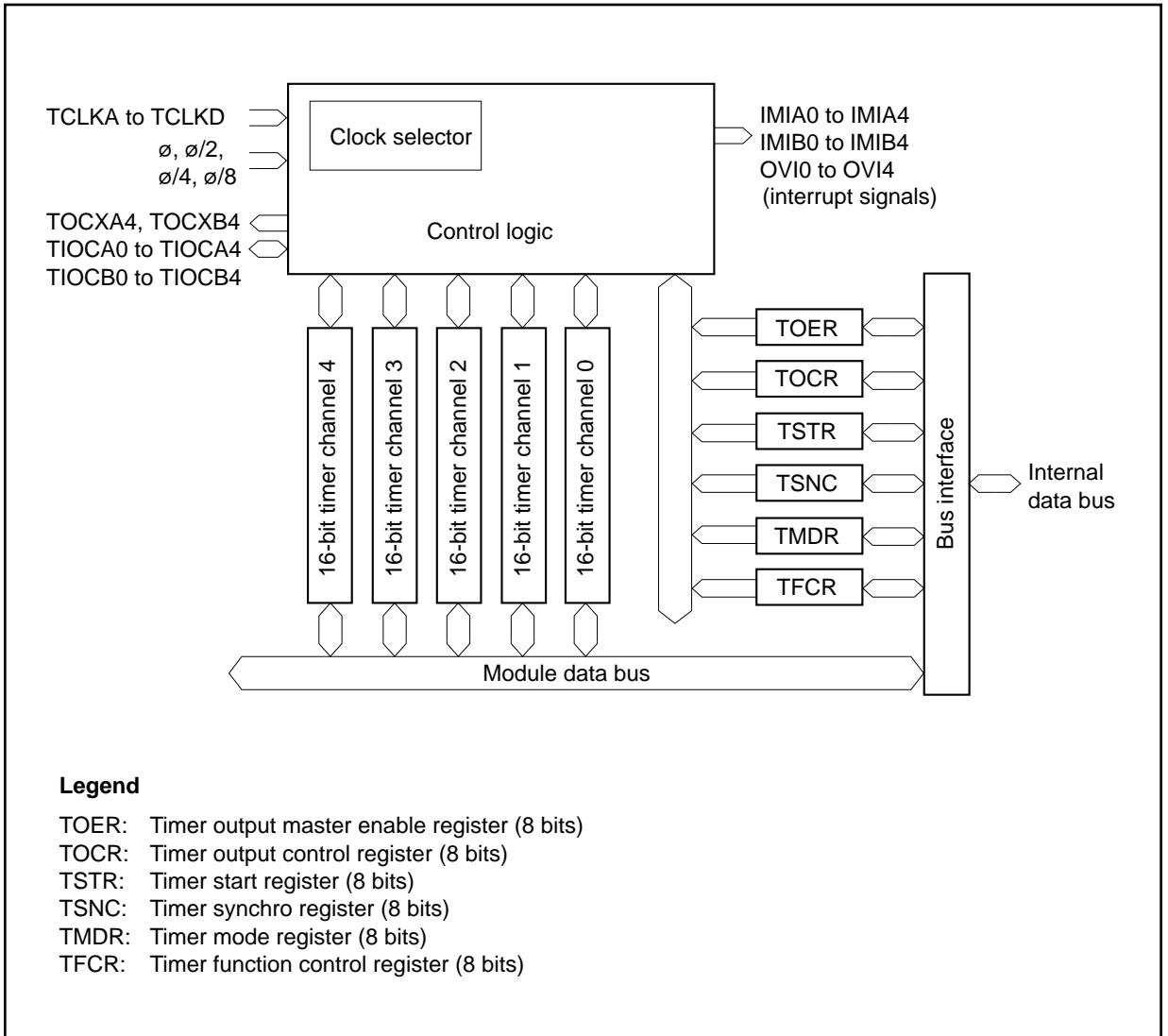
If channels 3 and 4 are combined, three-phase PWM output is possible with three pairs of non-overlapping complementary waveforms.

— Buffering

Input capture registers can be double-buffered. Output compare registers can be updated automatically.

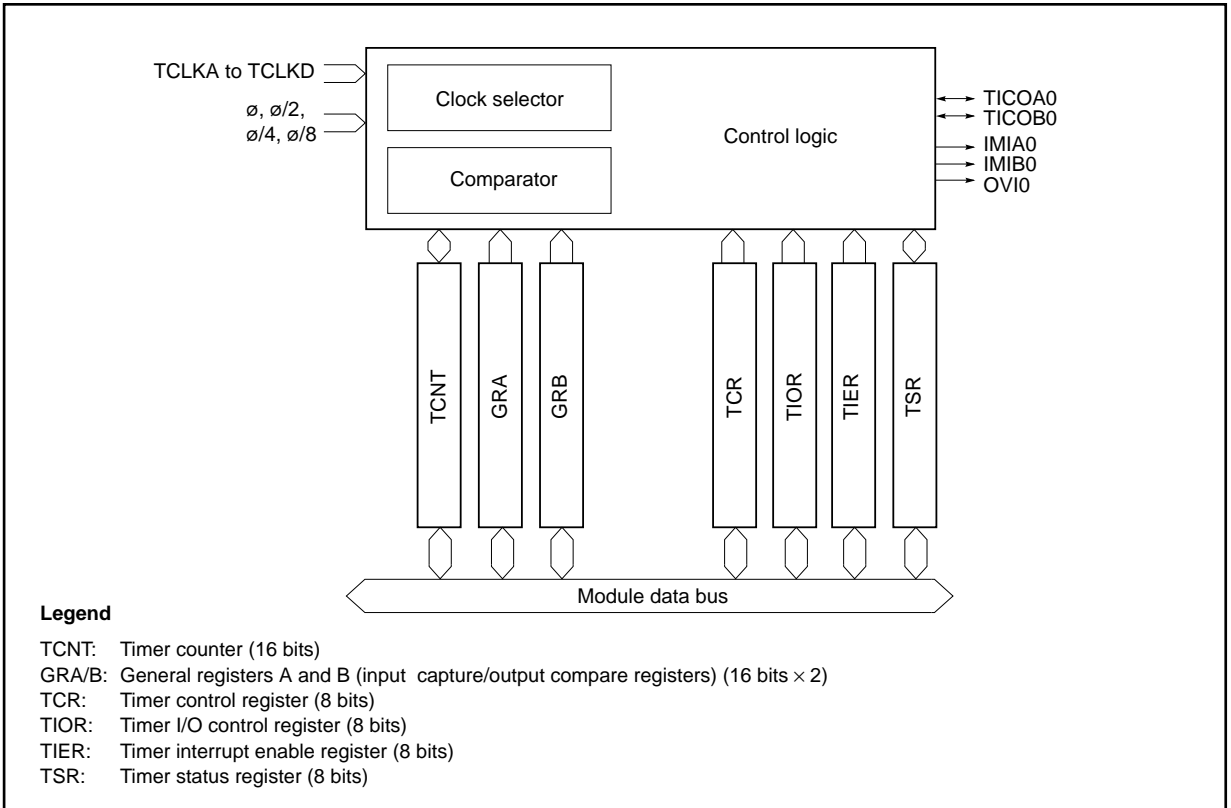
- High-speed access via internal 16-bit bus  
The 16-bit timer counters, general registers, and buffer registers can be accessed at high speed via a 16-bit bus.
- Fifteen interrupt sources  
Each channel has two compare match/input capture interrupts and an overflow interrupt. All interrupts can be requested independently.
- Activation of DMA controller (DMAC)  
Four of the compare match/input capture interrupts from channels 0 to 3 can start the DMAC.
- Output triggering of programmable pattern controller (TPC)  
Compare match/input capture signals from channels 0 to 3 can be used as TPC output triggers.

## Block Diagram

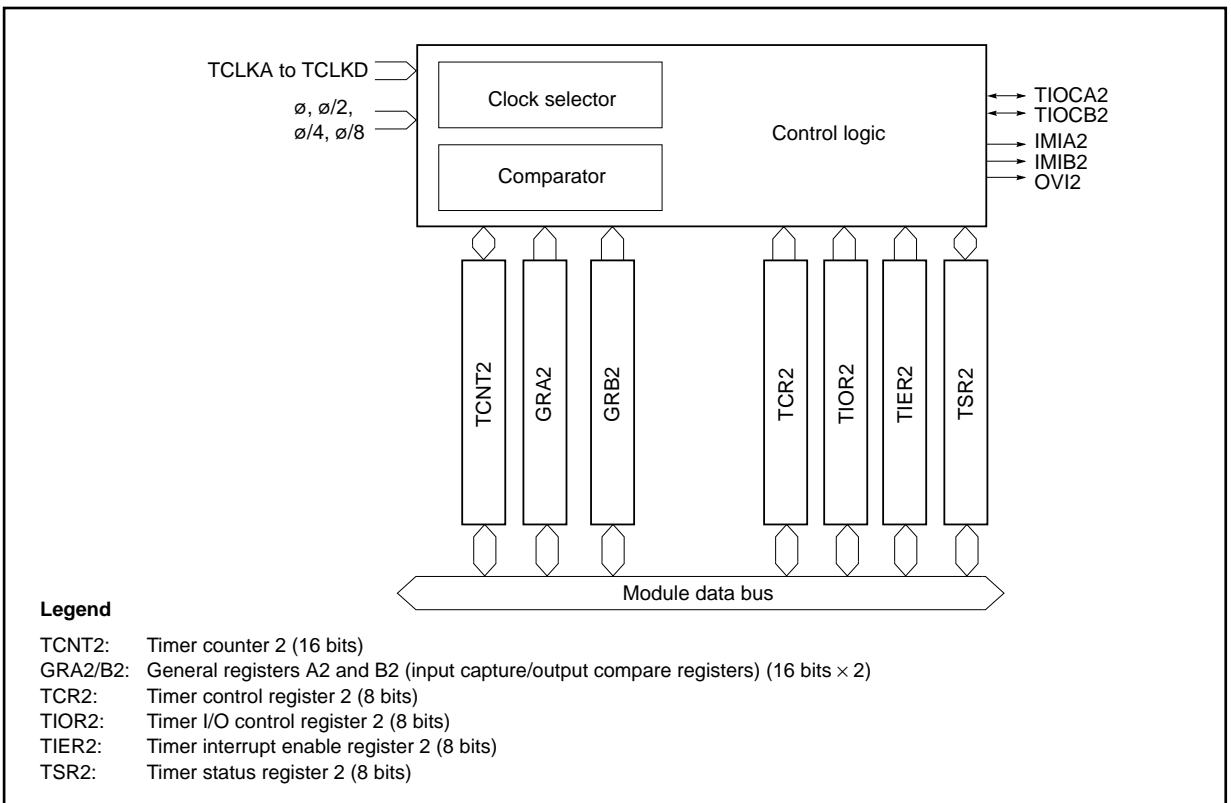


**ITU Block Diagram (Overall)**

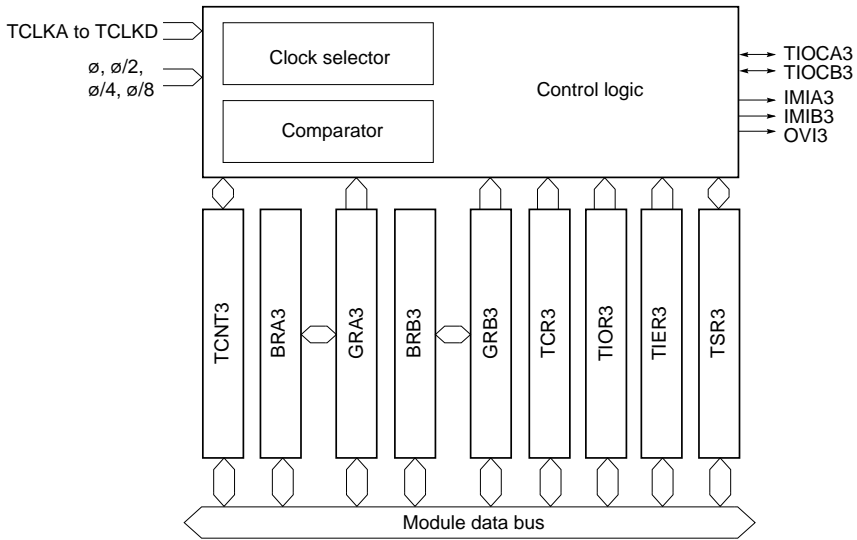
## Block Diagram of Channels 0 and 1 (One Channel)



## Block Diagram of Channel 2



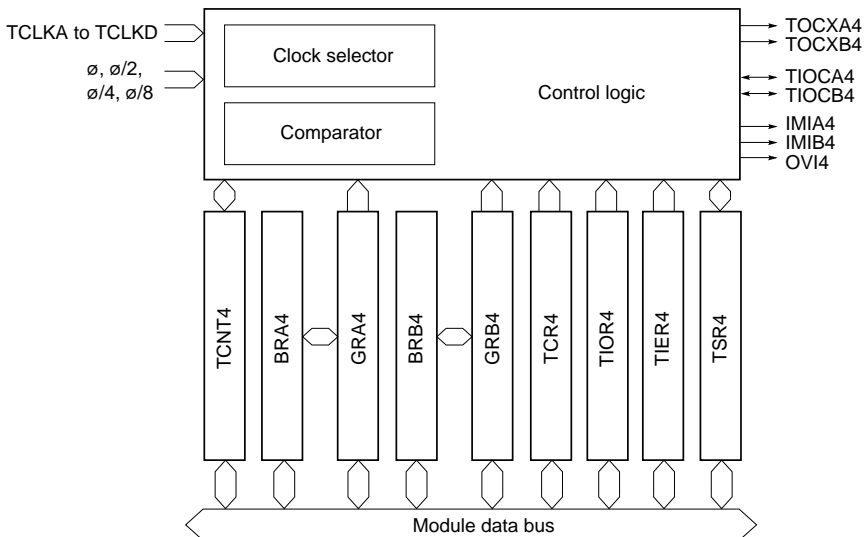
### Block Diagram of Channel 3



#### Legend

- TCNT3: Timer counter 3 (16 bits)
- GRA3/B3: General registers A3 and B3 (input capture/output compare registers) (16 bits × 2)
- BRA3/B3: Buffer registers A3 and B3 (input capture/output compare buffer registers) (16 bits × 2)
- TCR3: Timer control register 3 (8 bits)
- TIOR3: Timer I/O control register 3 (8 bits)
- TIER3: Timer interrupt enable register 3 (8 bits)
- TSR3: Timer status register 3 (8 bits)

### Block Diagram of Channel 4

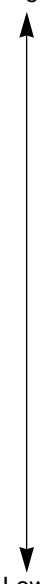


#### Legend

- TCNT4: Timer counter 4 (16 bits)
- GRA4/B4: General registers A4 and B4 (input capture/output compare registers) (16 bits × 2)
- BRA4/B4: Buffer registers A4 and B4 (input capture/output compare buffer registers) (16 bits × 2)
- TCR4: Timer control register 4 (8 bits)
- TIOR4: Timer I/O control register 4 (8 bits)
- TIER4: Timer interrupt enable register 4 (8 bits)
- TSR4: Timer status register 4 (8 bits)

## Interrupt Sources and DMA Controller (DMAC) Activation

The ITU has 15 interrupt sources, among which four of the compare match/input capture interrupts can activate the DMAC to perform data transfers.

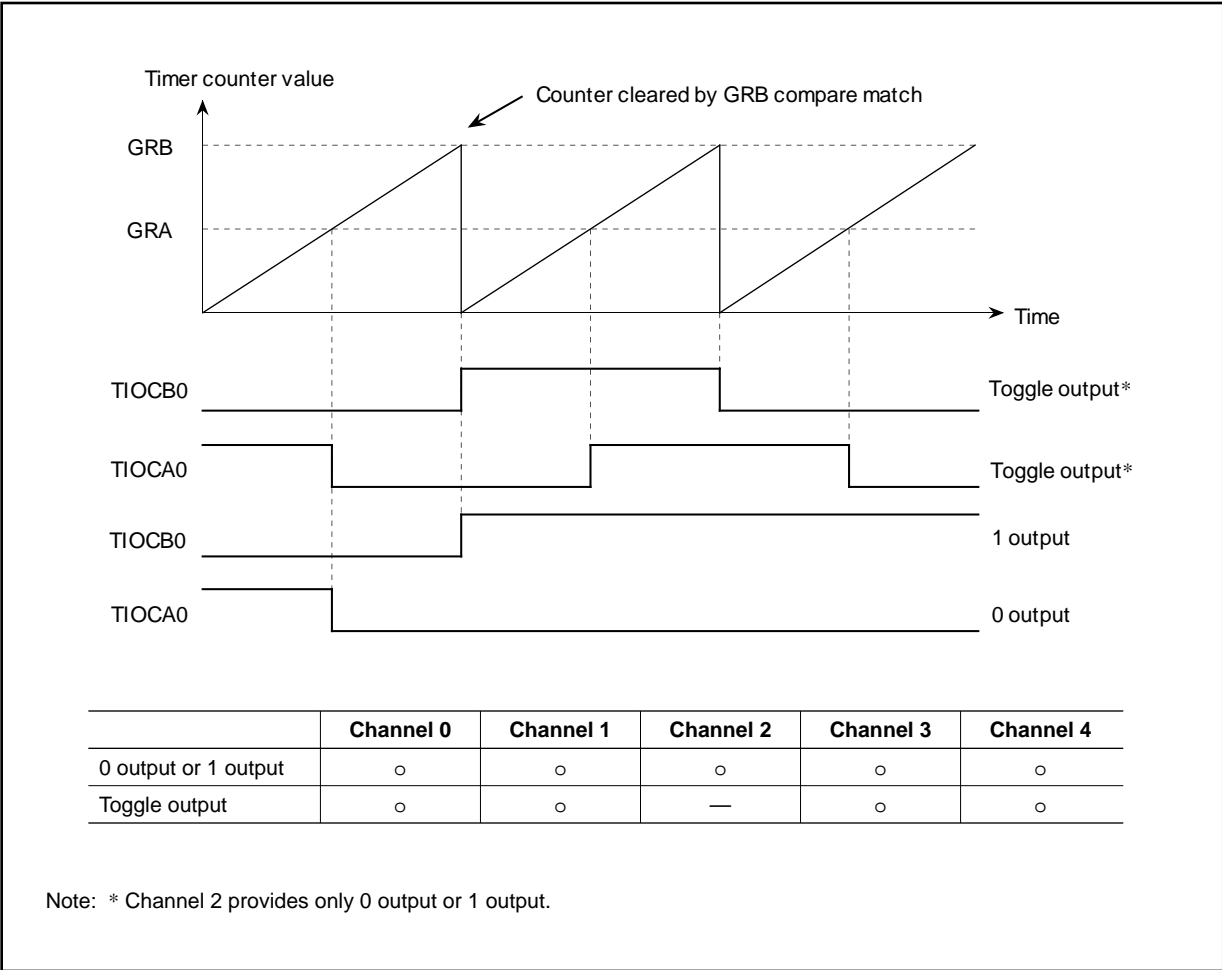
Channel	Interrupt Source	Description	DMAC Activatable	Priority
0	IMIA <sub>0</sub>	GRA compare match or input capture	Yes	
	IMIB <sub>0</sub>	GRB compare match or input capture	No	
	OVI <sub>0</sub>	Timer counter 0 overflow	No	
1	IMIA <sub>1</sub>	GRA compare match or input capture	Yes	
	IMIB <sub>1</sub>	GRB compare match or input capture	No	
	OVI <sub>1</sub>	Timer counter 1 overflow	No	
2	IMIA <sub>2</sub>	GRA compare match or input capture	Yes	
	IMIB <sub>2</sub>	GRB compare match or input capture	No	
	OVI <sub>2</sub>	Timer counter 2 overflow	No	
3	IMIA <sub>3</sub>	GRA compare match or input capture	Yes	
	IMIB <sub>3</sub>	GRB compare match or input capture	No	
	OVI <sub>3</sub>	Timer counter 3 overflow	No	
4	IMIA <sub>4</sub>	GRA compare match or input capture	No	
	IMIB <sub>4</sub>	GRB compare match or input capture	No	
	OVI <sub>4</sub>	Timer counter 4 overflow	No	



# Pulse Output

0 output, 1 output, or toggle output can be selected.

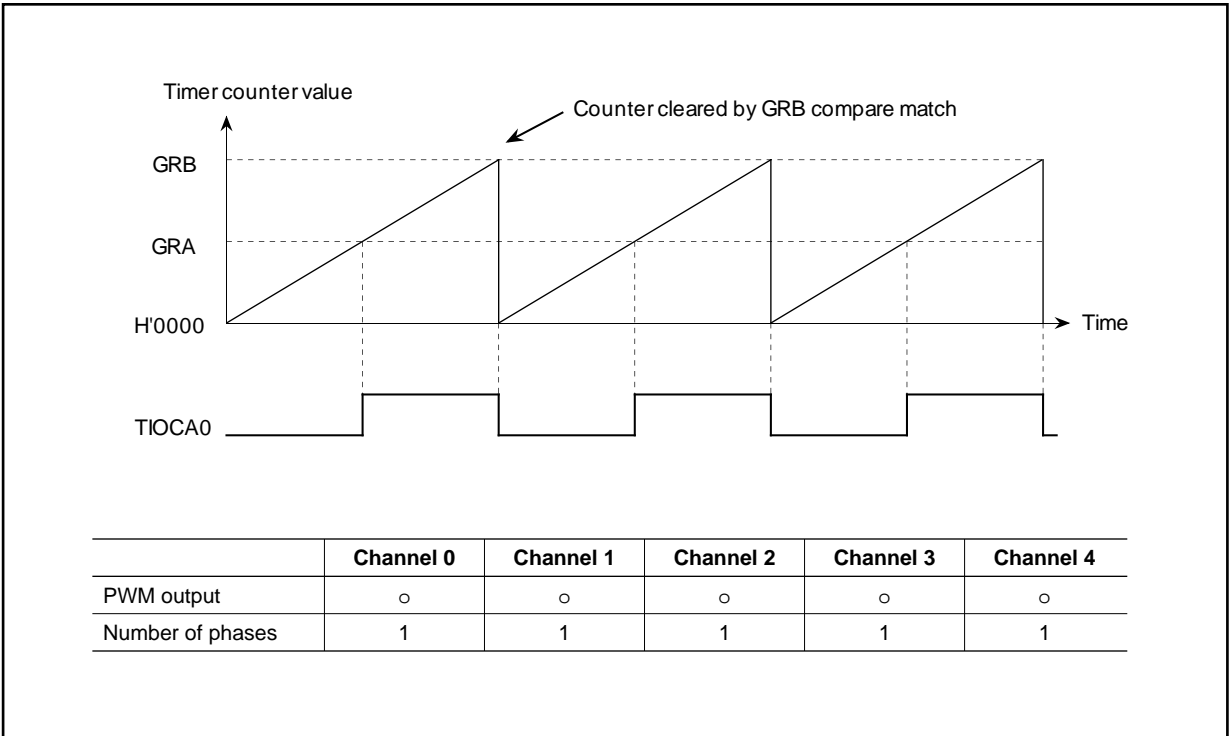
**Example of Two-Phase Pulse Output:** The following example shows two-phase pulse output on channel 0.



## PWM Mode

GRA and GRB can be paired to generate a PWM waveform with independently selectable period and duty cycle. With synchronization, up to five-phase PWM output is possible.

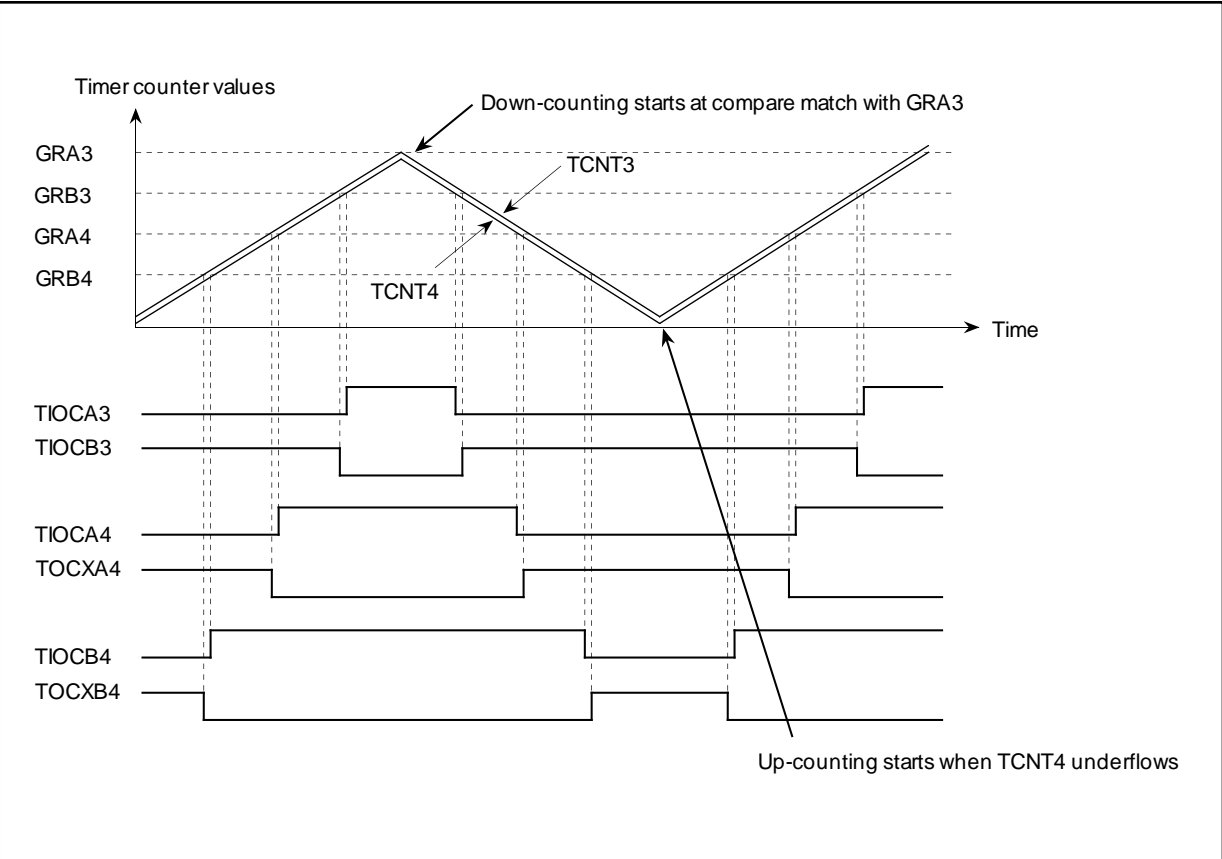
**Example of Single-Phase PWM Output:** The following example shows single-phase PWM output on channel 0.



# Complementary PWM Output Mode

Channels 3 and 4 can be combined to output complementary, non-overlapping PWM waveforms.

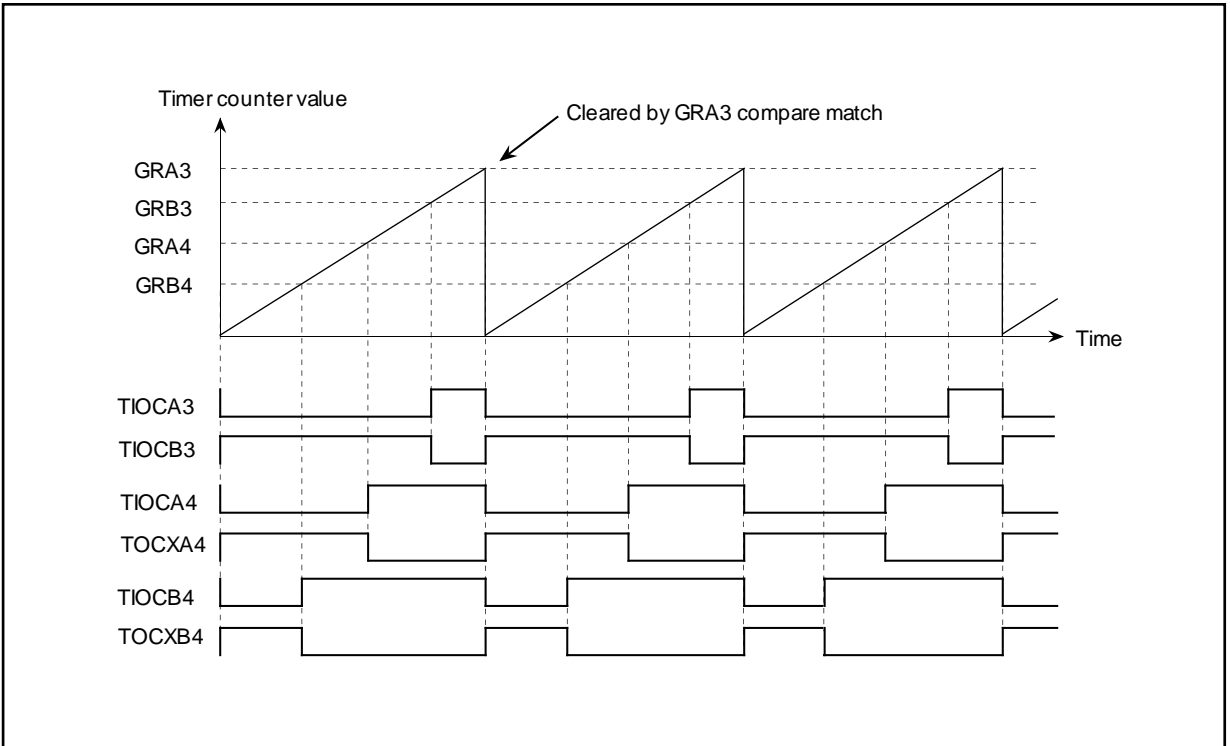
**Example of Operation in Complementary PWM Mode (Up/Down Counting):** The following example shows output of complementary PWM waveforms on channels 3 and 4.



## Reset-Synchronized PWM Mode

Channels 3 and 4 are combined to produce three pairs of complementary PWM waveforms, all having one waveform transition point in common.

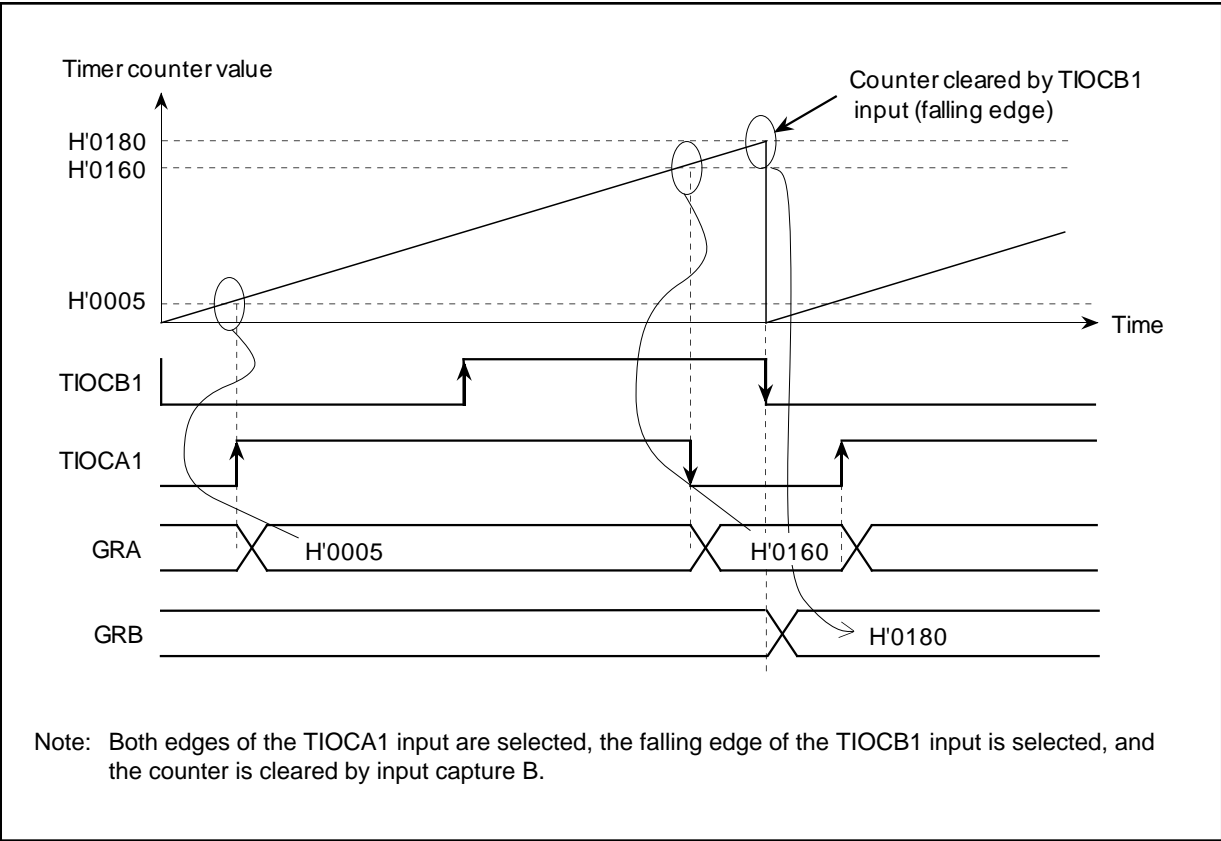
**Example of Operation in Reset-Synchronized PWM Mode (Up-Counting):** The following example shows reset-synchronized PWM output on channels 3 and 4.



# Pulse Measurement Functions

The pulse period of an external pulse signal can be measured by connecting the signal to an input capture pin. Input capture on the rising edge, falling edge, or both edges can be selected. (Similarly, when an external clock is selected, the rising edge, falling edge, or both edges of the clock signal can be counted.)

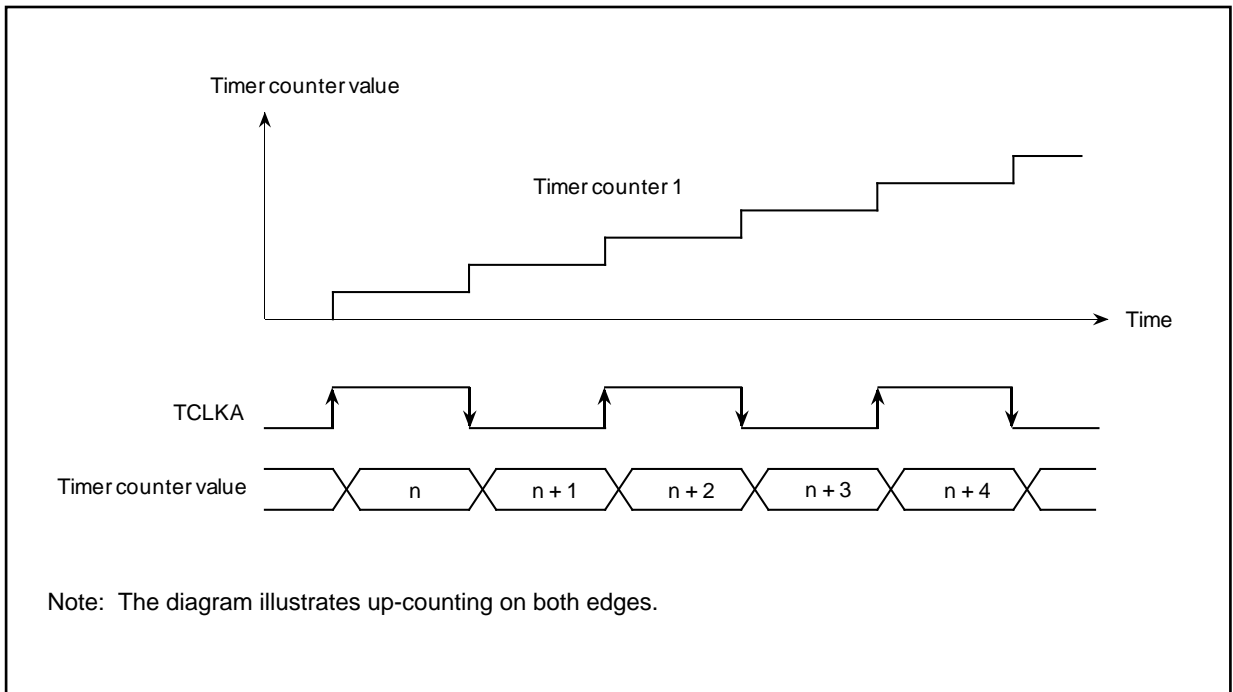
**Example of Operation in Input Capture Mode:** The following example shows pulse cycle measurement on channel 1.



## Event Counting

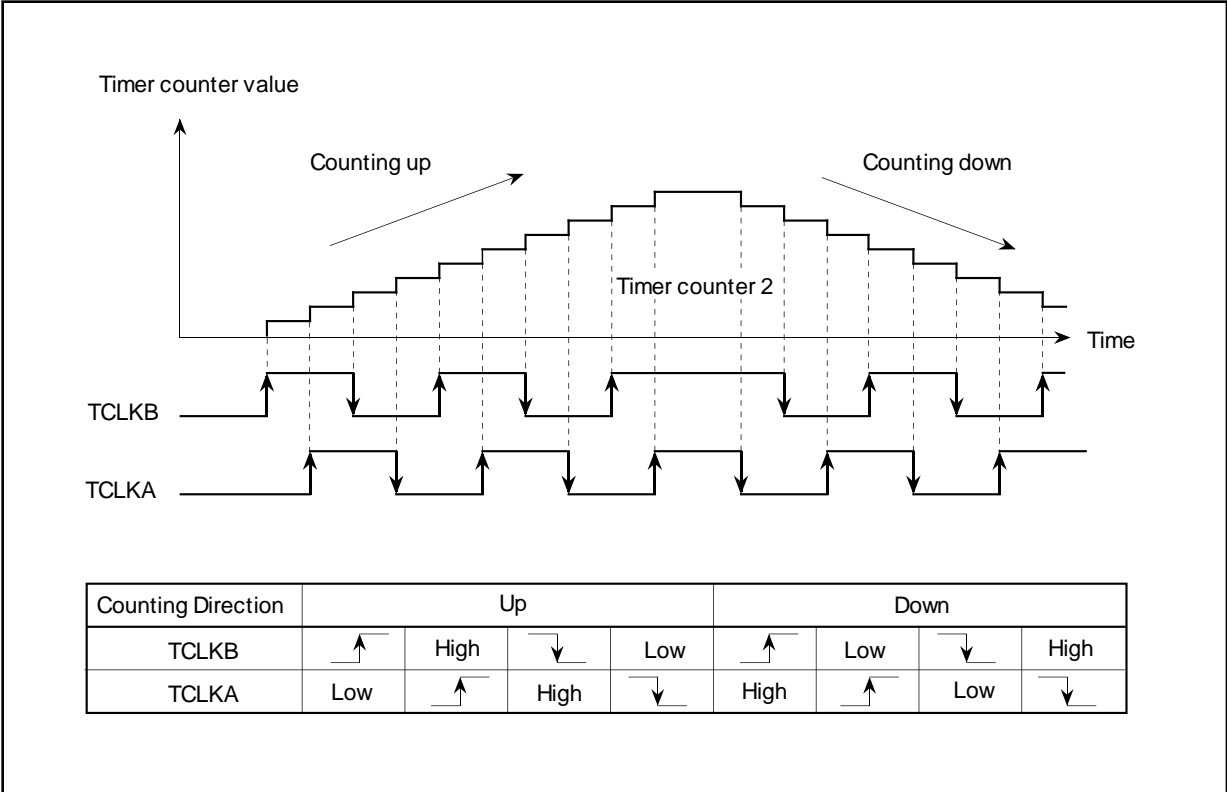
External events can be counted by inputting event signals to pins TCLKA to TCLKD. Counting of the rising edge, falling edge, or both edges can be selected.

**Example of External Event Counting:** The following example shows counting of external events on channel 1.



# Two-Phase Encoder Processing

This function is available only on channel 2. Output pulses from a two-phase encoder in a motor are input to pins TCLKA and TCLKB. The timer counter counts up or down according to the phase difference. The following example illustrates two-phase encoder processing by channel 2.



## 6.5 Programmable Timing Pattern Controller (TPC)

### Functions

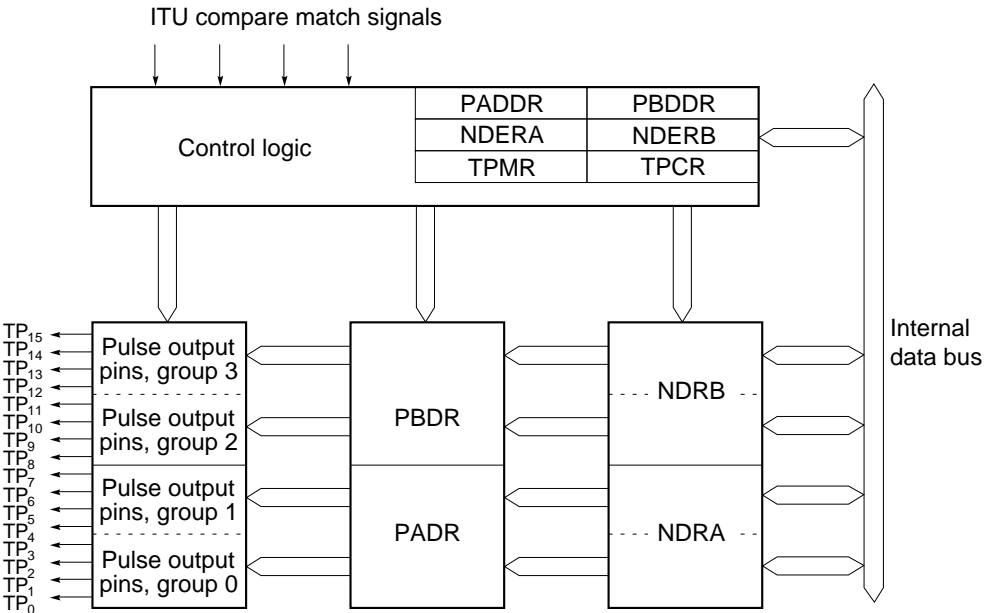
The programmable timing pattern controller (TPC) receives input signals from the 16-bit integrated timer unit (ITU), and can control up to 16 output signals simultaneously.

### Features

- 16-bit output data  
Maximum 16-bit data can be output. TPC output can be enabled on a bit-by-bit basis.
- Four output groups  
Output trigger signals can be selected in 4-bit groups to provide up to four independent 4-bit outputs.
- Selectable output trigger signals  
Output trigger signals can be selected for each group from the compare-match signals of four ITU channels.
- Non-overlap mode  
A non-overlap margin can be provided between pulse outputs.
- Can be teamed with the DMA controller (DMAC)  
The compare-match signals selected as output trigger signals can activate the DMAC for output of data sequences without CPU intervention.



# Block Diagram



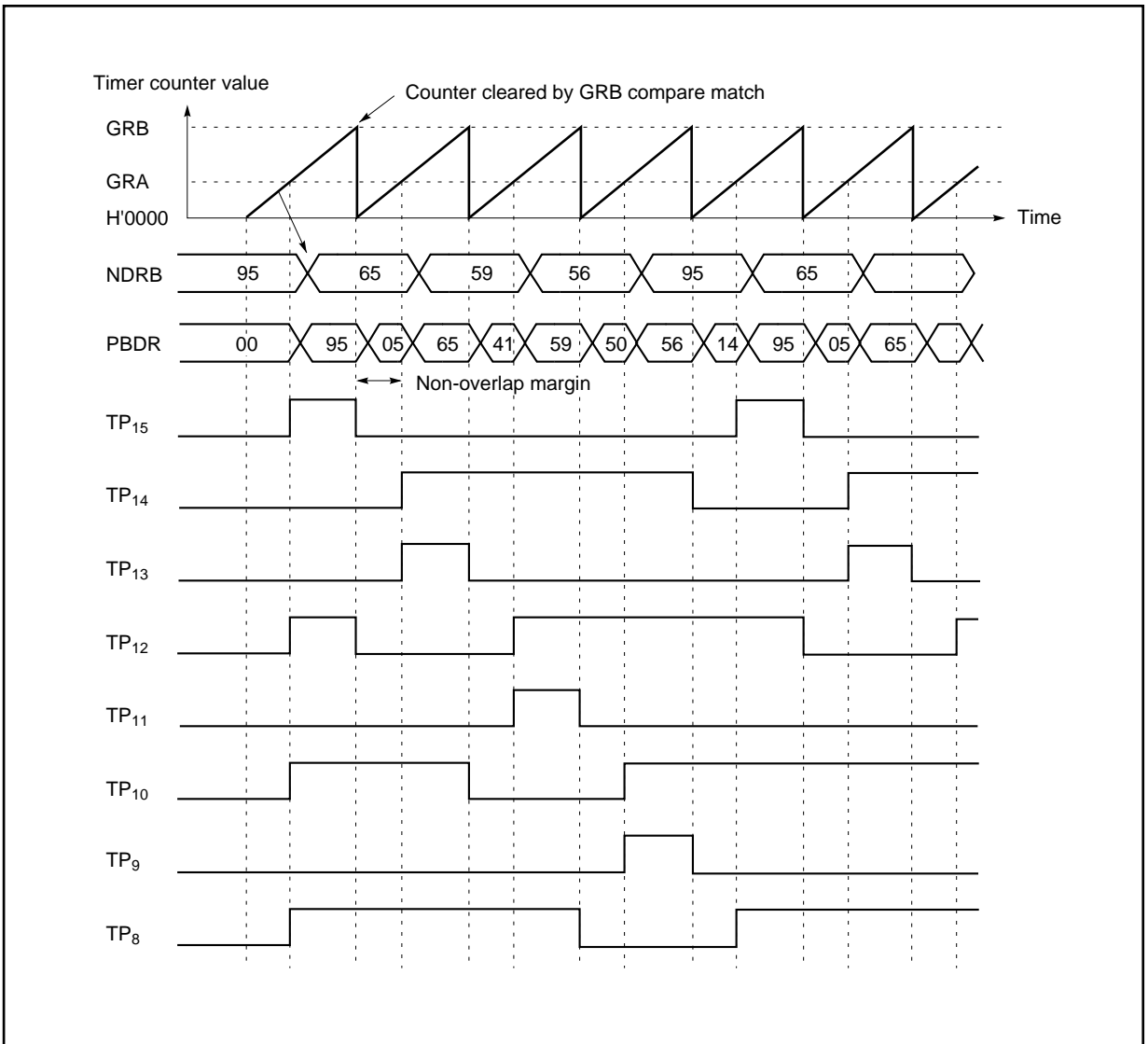
**Legend**

- TPMP: TPC output mode register
- TPCR: TPC output control register
- NDERB: Next data enable register B
- NDERA: Next data enable register A
- PBDDR: Port B data direction register
- PADDR: Port A data direction register
- NDRB: Next data register B
- NDRA: Next data register A
- PBDR: Port B data register
- PADR: Port A data register

**TPC Block Diagram**

## Example of Four-Phase Complementary Non-Overlapping Output

The following example shows four-phase complementary non-overlapping pulse output. Output goes to 0 at compare match B and to 1 at compare match A. The non-overlap margin is set in register GRA of the ITU channel selected as the output trigger. The period from one pulse to the next is set in GRB. The pulse outputs are obtained by writing the illustrated output data at IMFA interrupts.



## 6.6 Watchdog Timer (WDT)

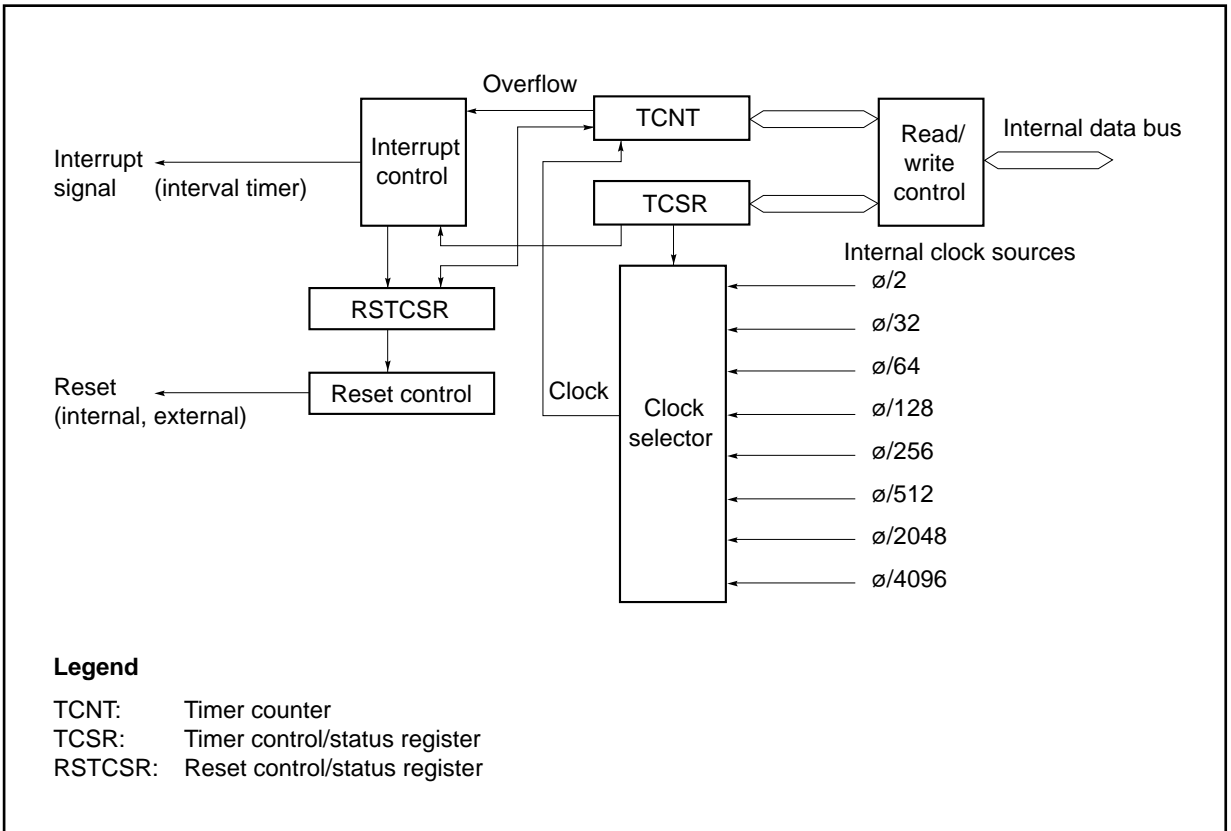
### Functions

The watchdog timer (WDT) can be used for system supervision. The WDT has two selectable functions: it can operate as a watchdog timer to supervise system operation, or it can operate as an interval timer.

### Features

- Selection of eight counter clock sources  
 $\phi/2$ ,  $\phi/32$ ,  $\phi/64$ ,  $\phi/128$ ,  $\phi/256$ ,  $\phi/512$ ,  $\phi/2048$ , or  $\phi/4096$
- Interval timer option
- Timer counter overflow generates a reset signal or interrupt  
The reset signal is generated in watchdog timer operation. An interval timer interrupt is generated in interval timer operation.
- Watchdog timer reset signal resets the entire chip internally, and can also be output externally  
The reset signal generated by timer counter overflow during watchdog timer operation resets the entire chip internally. An external reset signal can be output from the  $\overline{\text{RESO}}$  pin to reset other system devices simultaneously.

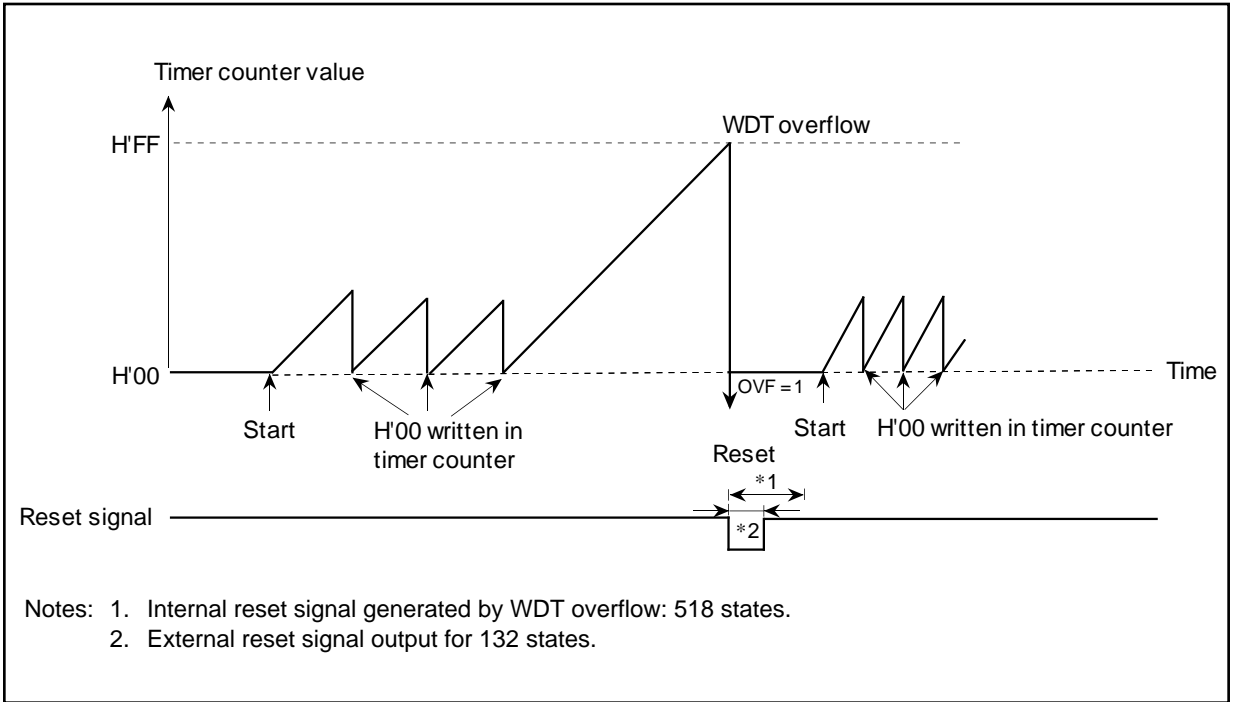
## Block Diagram



**Watchdog Timer Block Diagram**

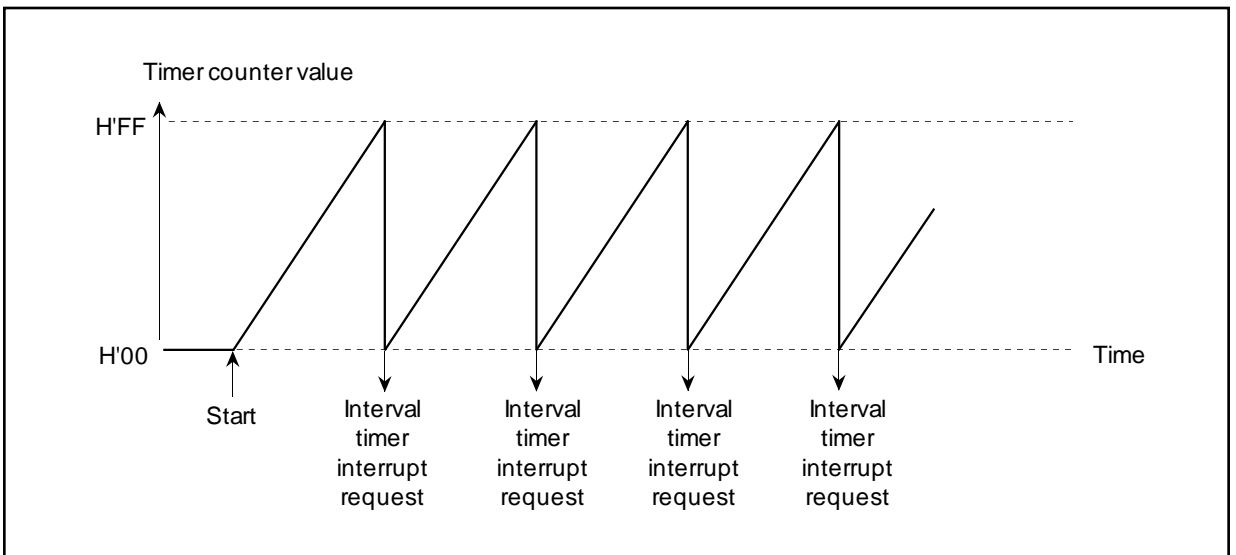
## Watchdog Timer Operation

The following example shows watchdog timer usage. The timer counter (TCNT) starts counting up, using the specified clock source.



## Interval Timer Operation

The following example shows interval timer usage. The timer counter (TCNT) starts counting up, using the specified clock source, and generates an interval timer interrupt request at each TCNT overflow. This function can be used to generate interrupts at regular intervals.



## 6.7 Serial Communication Interface (SCI)

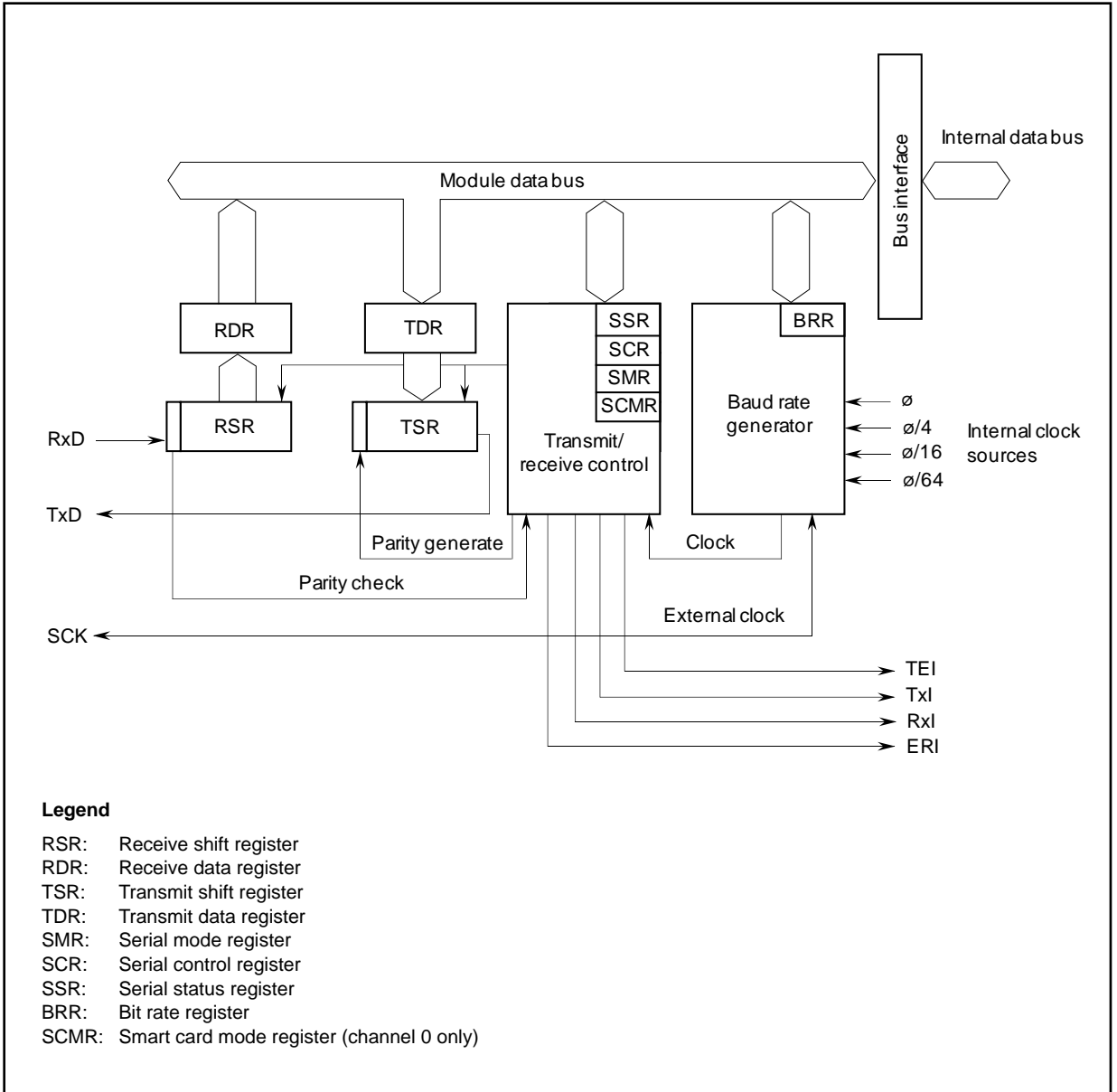
### Functions

The H8/3048 Series has a serial communication interface (SCI) with two channels. The SCI is an on-chip supporting module that can perform serial communication with external devices in asynchronous or synchronous mode. It also has a multiprocessor communication function for communication among two or more processors, and one of the two SCI channels has a smart card interface function for communication with an IC card.

### Features

- Selection of asynchronous mode or synchronous mode
- Full duplex communication
- Double-buffered data registers for continuous transmit/receive
- Built-in baud rate generator with selection of arbitrary bit rate
- Selection of built-in baud rate generator or external clock input (SCK pin) as clock source
- Three types of receive errors detected: overrun errors, framing errors, and parity errors
- Detection of line break state
- Four independently requestable interrupt sources: transmit data empty, transmit end, receive data full, and receive error, of which transmit data empty and receive data full can activate the DMA controller
- Built-in multiprocessor communication function
- Built-in smart card interface function for communication with IC cards

## Block Diagram



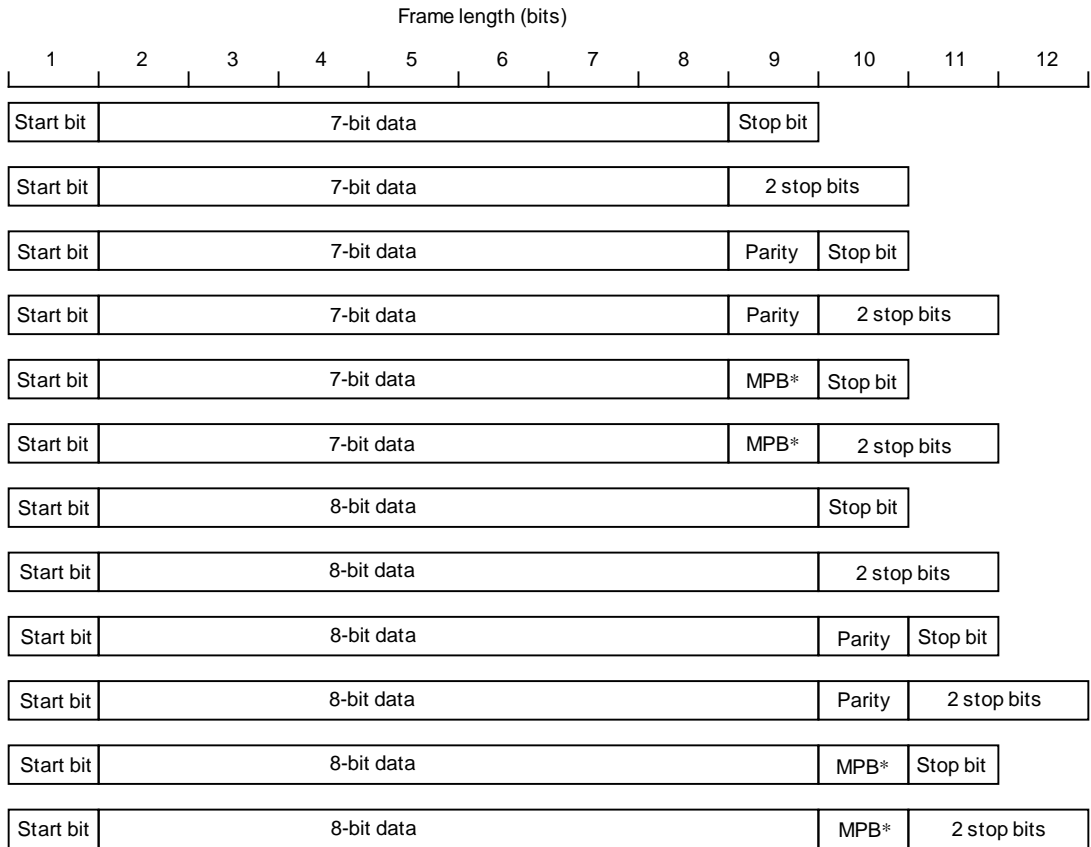
**Serial Communication Interface Block Diagram (One Channel)**

## **Asynchronous Mode**

Asynchronous mode is a serial communication mode in which each character is framed by a start bit and a stop bit.

- Twelve selectable communication formats
  - Data length: 7 or 8 bits
  - Stop bit length: 1 or 2 bits
  - Parity bit: even, odd, or none
  - Multiprocessor bit: 1 or 0
- Selection of built-in baud rate generator or external clock input (SCK pin) as clock source
- Output of serial clock at SCK pin
- Detection of break state by reading state of RxD pin when a framing error occurs
- Multiprocessor communication





Note: \* MPB: Multiprocessor bit.

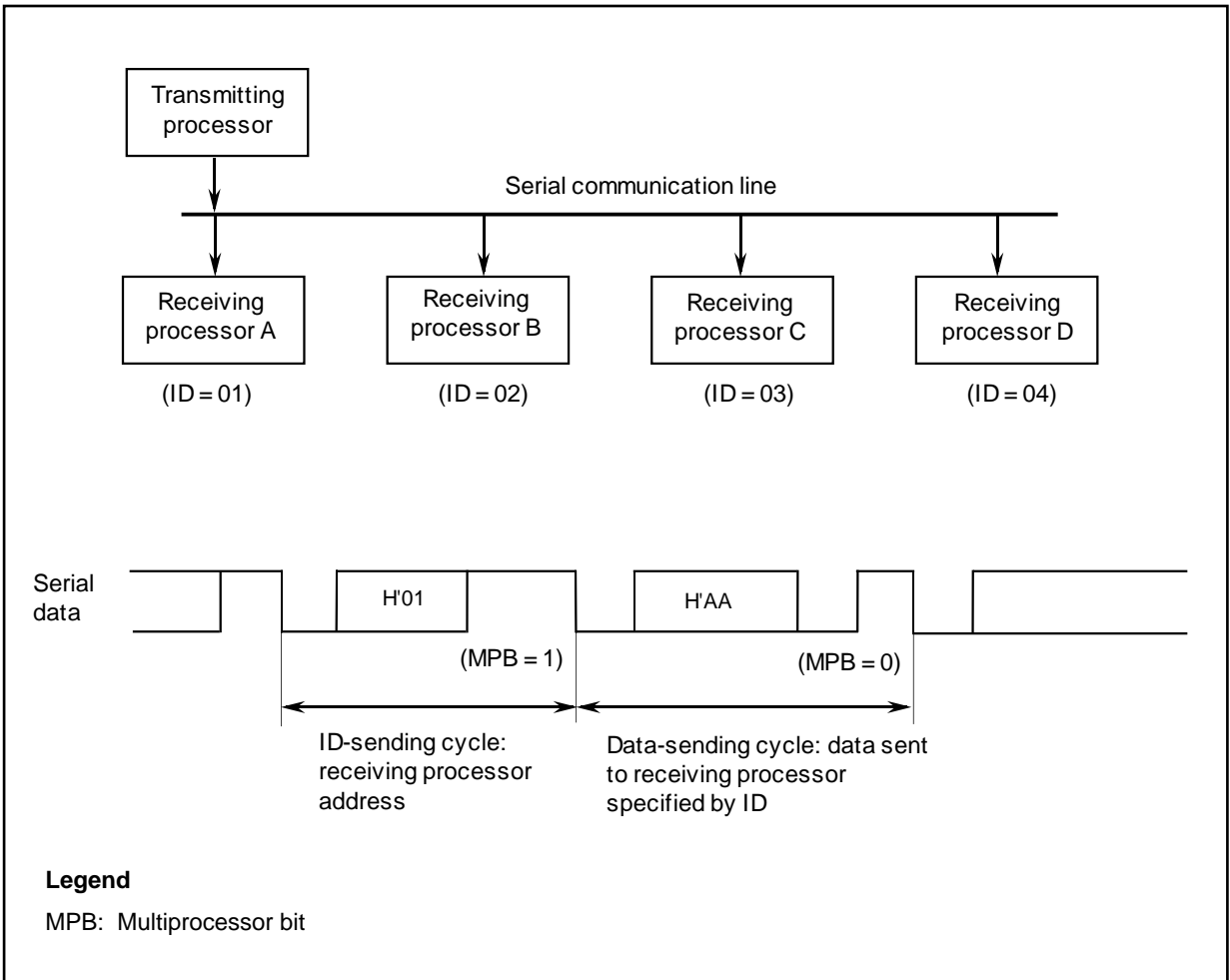
## Communication Formats and Frame Length in Asynchronous Mode

## Multiprocessor Communication Function

Two or more processors can share a single serial communication line by using a format with an additional multiprocessor bit.

The transmitting processor starts by sending the ID of the receiving processor with which it wants to communicate as data with the multiprocessor bit (MPB) set to 1. Next the transmitting processor sends transmit data with the MPB cleared to 0.

A receiving processor skips incoming data until it receives data with the MPB set to 1, at which point it compares the data with its own ID. If the ID matches, the receiving processor continues to receive further incoming data. If the ID does not match, the receiving processor skips further incoming data until it again receives data with the MPB set to 1.



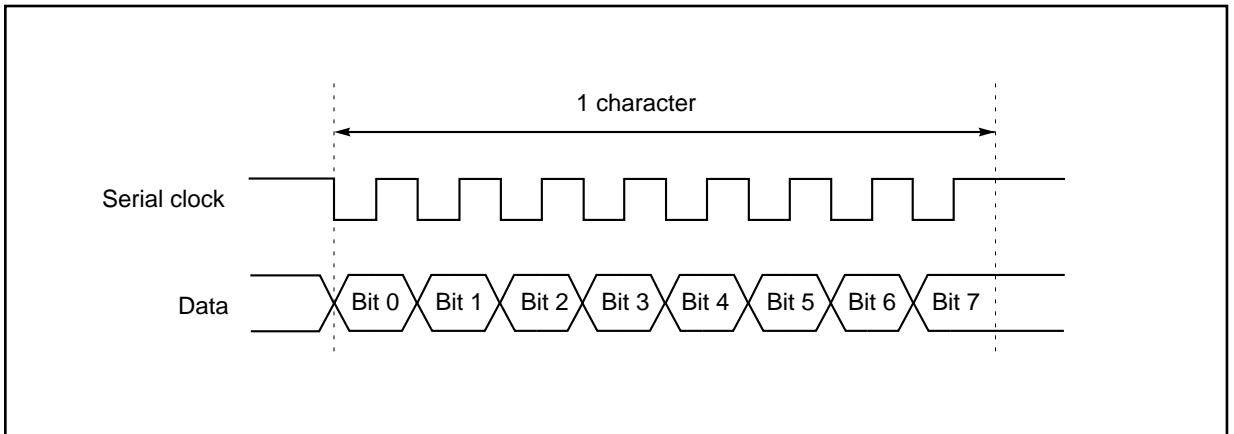
**Example of Communication among Processors using Multiprocessor Format  
(Sending Data H'AA to Receiving Processor A)**

## Synchronous Mode

In synchronous mode, the SCI transmits and receives data in synchronization with clock pulses. This mode is suitable for continuous, high-speed serial communication.

- Data length: 8 bits/character
- Overrun errors detectable
- Selection of built-in baud rate generator or external clock input (SCK pin) as clock source
- LSB-first format: least significant bit is transmitted or received first
- Can communicate with H8/500 Series, H8/300 Series, HD64180, and other chips having a synchronous mode

If the internal baud rate generator is selected, the SCK pin automatically switches to output mode and eight serial clock pulses are output. Communication is possible with other chips having a built-in synchronous serial interface, including the HD64180, HD6301, and H8 Series.



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

## Interrupt Sources and DMA Controller Activation

The SCI has four interrupt sources: receive error (overflow error, framing error, or parity error), transmit end (MSB sent), transmit data empty, and receive data full. The transmit data empty and receive data full interrupts in channel 0 can activate the DMA controller. Data can be transmitted or received continuously without software overhead by using the DMA controller to write the next transmit data when the transmit data empty interrupt occurs, or read receive data when the receive data full interrupt occurs.

### SCI Interrupt Sources

Interrupt Source	Description	DMAC Activatable	Priority
ERI	Receive error interrupt	No	<div style="text-align: center;">                     High                      ↑                      ↓                      Low                 </div>
RxI	Receive data register full interrupt	Yes	
TxI	Transmit data register empty interrupt	Yes	
TEI	Transmit end interrupt	No	

### Examples of Bit Rates and BRR Settings (Synchronous Mode)

Bit Rate (bits/s) \ $\phi$ (MHz)	2		4		8		10		16	
	n	N	n	N	n	N	n	N	n	N
110	3	70	—	—	—	—	—	—	—	—
250	2	124	2	249	3	124	—	—	3	249
500	1	249	2	124	2	249	—	—	3	124
1 k	1	124	1	249	2	124	—	—	2	249
2.5 k	0	199	1	99	1	199	1	249	2	99
5 k	0	99	0	199	1	99	1	124	1	199
10 k	0	49	0	99	0	199	0	249	1	99
25 k	0	19	0	39	0	79	0	99	0	159
50 k	0	9	0	19	0	39	0	49	0	79
100 k	0	4	0	9	0	19	0	24	0	39
250 k	0	1	0	3	0	7	0	9	0	15
500 k	0	0	0	1	0	3	0	4	0	7
1 M			0	0	0	1	—	—	0	3
2 M					0	0	—	—	0	1
2.5 M					—	—	0	0	—	—
4 M									0	0

The BRR setting is calculated as follows:

$$N = \frac{\varnothing}{8 \times 2^{2n-1} \times B} \times 10^6 - 1$$

n	Clock
0	$\varnothing$
1	$\varnothing/4$
2	$\varnothing/16$
3	$\varnothing/64$

### Legend

N: BRR setting for baud rate generator ( $0 \leq N \leq 255$ )

$\varnothing$ : System clock frequency (MHz)

B: Bit rate (bits/s)

n: Baud rate generator clock source ( $n = 0, 1, 2, 3$ )

(For the clock sources and values of n, see the accompanying table.)

## 6.8 Smart Card Interface

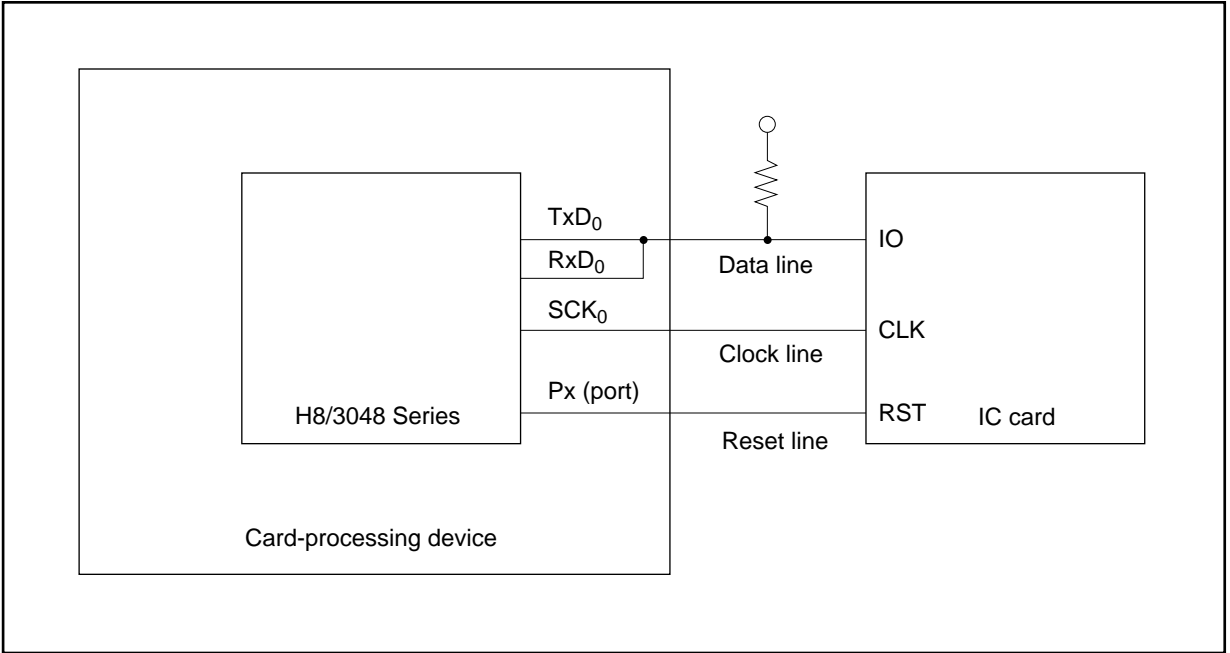
Of the two SCI channels, SCI0 supports a smart card interface by providing functions for serial communication with an IC card conforming to the ISO/IEC7816-3 (Identification Card) standard.

### Features

The main features of the smart-card interface are as follows.

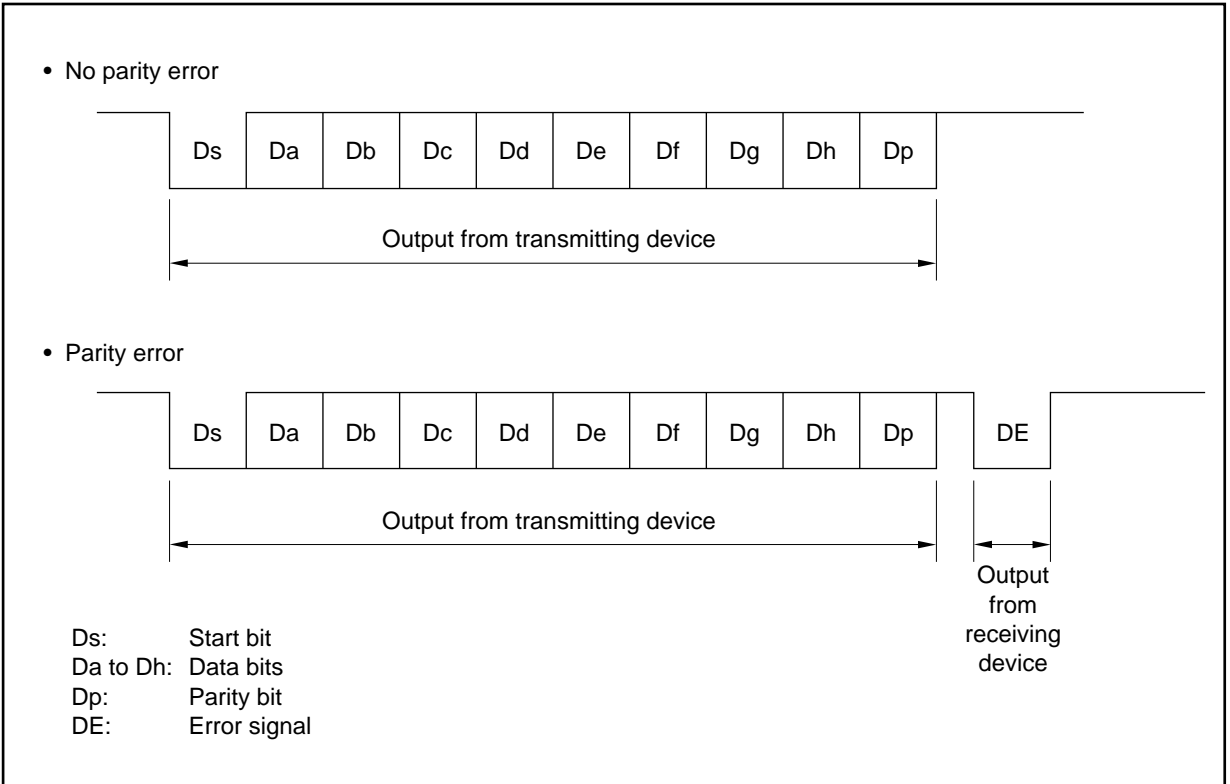
- One frame consists of eight data bits and a parity bit.
- In transmitting, a guard time of at least two elementary time units (2 etu) is provided between the end of the parity bit and the start of the next frame.
- In receiving, if a parity error is detected, a low error signal is output for 1 etu, beginning 10.5 etu after the start bit.
- In transmitting, if an error signal is received, after at least 2 etu, the same data is automatically transmitted again.
- Only asynchronous communication is supported. There is no clock synchronous operation.
- Selection of direct or inverse convention.

## Connection Diagram



**Smart Card Interface Connection Diagram**

## Data Format



**Smart Card Interface Data Format**

## 6.9 A/D Converter

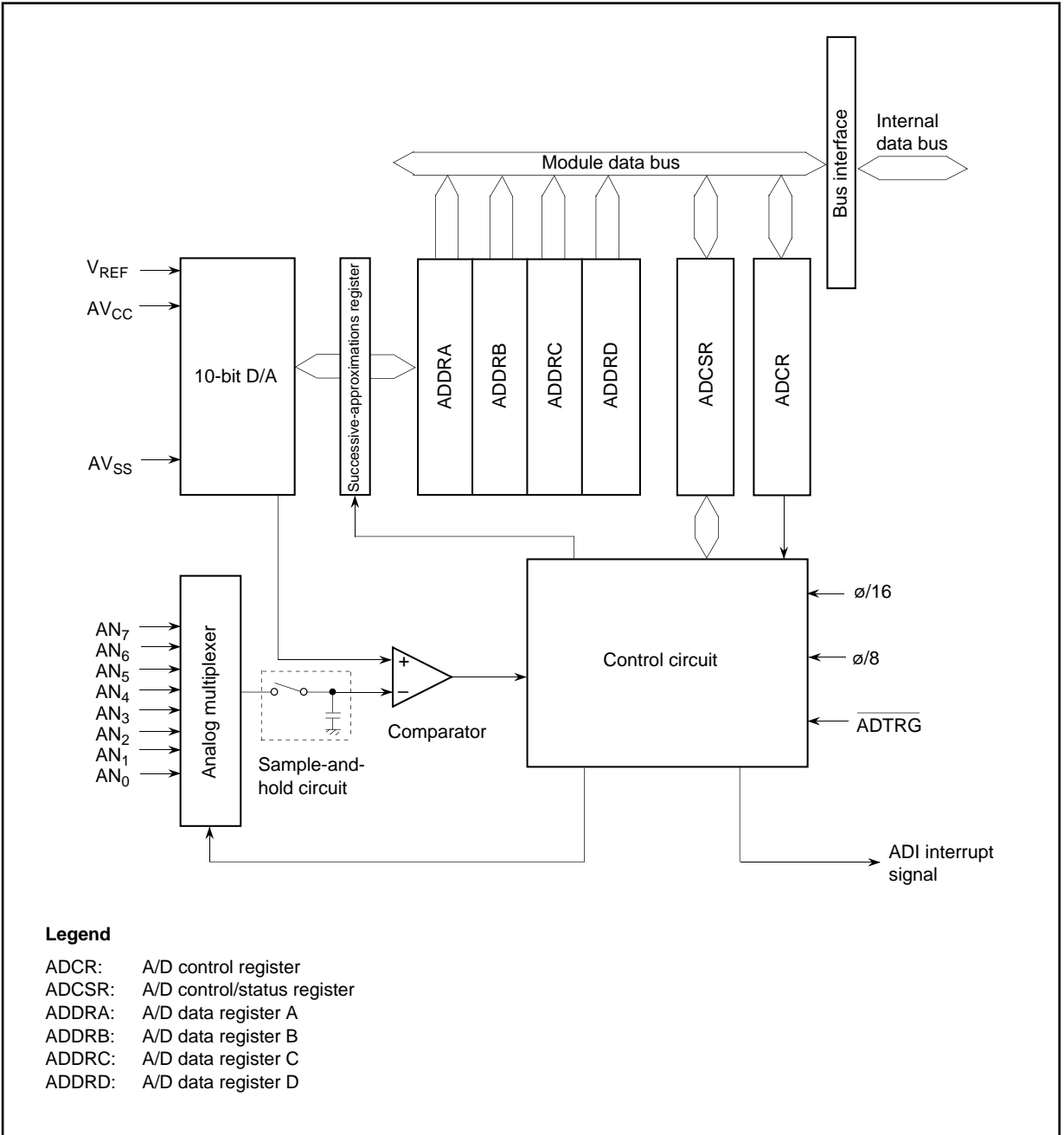
### Functions

The H8/3048 Series includes a 10-bit A/D converter with a selection of up to eight software-selectable analog input channels.

### Features

- 10-bit resolution
- Eight analog input channels
- High-speed conversion  
Conversion time: minimum 7.5  $\mu$ s per channel (with 18-MHz system clock)
- Selection of two conversion modes: scan and single mode  
Single mode: A/D conversion of one channel  
Scan mode: continuous A/D conversion of one to four channels
- Sample-and-hold function
- External reference voltage pin
- A/D conversion can be externally triggered
- Four 16-bit data registers  
A/D conversion results are transferred for storage into data registers corresponding to the channels.
- A/D interrupt requested at end of conversion  
At the end of A/D conversion, an A/D end interrupt request (ADI) can be sent to the CPU.

# Block Diagram



**A/D Converter Block Diagram**



## Operation

The A/D converter operates by successive approximations with 10-bit resolution. The eight analog input channels are selected by bits CH2 to CH0 in ADCSR.

Bit 2	Bit 1	Bit 0	Channel Selection	
CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	AN <sub>0</sub>	AN <sub>0</sub>
		1	AN <sub>1</sub>	AN <sub>0</sub> and AN <sub>1</sub>
	1	0	AN <sub>2</sub>	AN <sub>0</sub> to AN <sub>2</sub>
		1	AN <sub>3</sub>	AN <sub>0</sub> to AN <sub>3</sub>
1	0	0	AN <sub>4</sub>	AN <sub>4</sub>
		1	AN <sub>5</sub>	AN <sub>4</sub> and AN <sub>5</sub>
	1	0	AN <sub>6</sub>	AN <sub>4</sub> to AN <sub>6</sub>
		1	AN <sub>7</sub>	AN <sub>4</sub> to AN <sub>7</sub>

**Single Mode:** In this mode A/D conversion is performed on one channel under CPU control. A/D conversion starts when the ADST bit is set to 1 in ADCSR. When conversion ends, the end flag (ADF) is set. If the interrupt enable bit (ADIE) is also set, an A/D conversion end interrupt (ADI) is requested at this time, and the interrupt-handling routine can process the A/D conversion result.

**Scan Mode:** This mode can be used to monitor analog inputs on one to four channels. A/D conversion starts on the first channel when the ADST bit is set to 1. As soon as conversion of the first channel ends, conversion of the second channel begins. (If only one channel is selected, conversion of the same channel begins again.) A/D conversion continues until the ADST bit is cleared. The conversion results are transferred to and stored in data registers ADDRA to ADDRD, using a separate data register for each channel.

The ADST bit can be set by software, or by an external trigger signal ( $\overline{\text{ADTRG}}$ ).

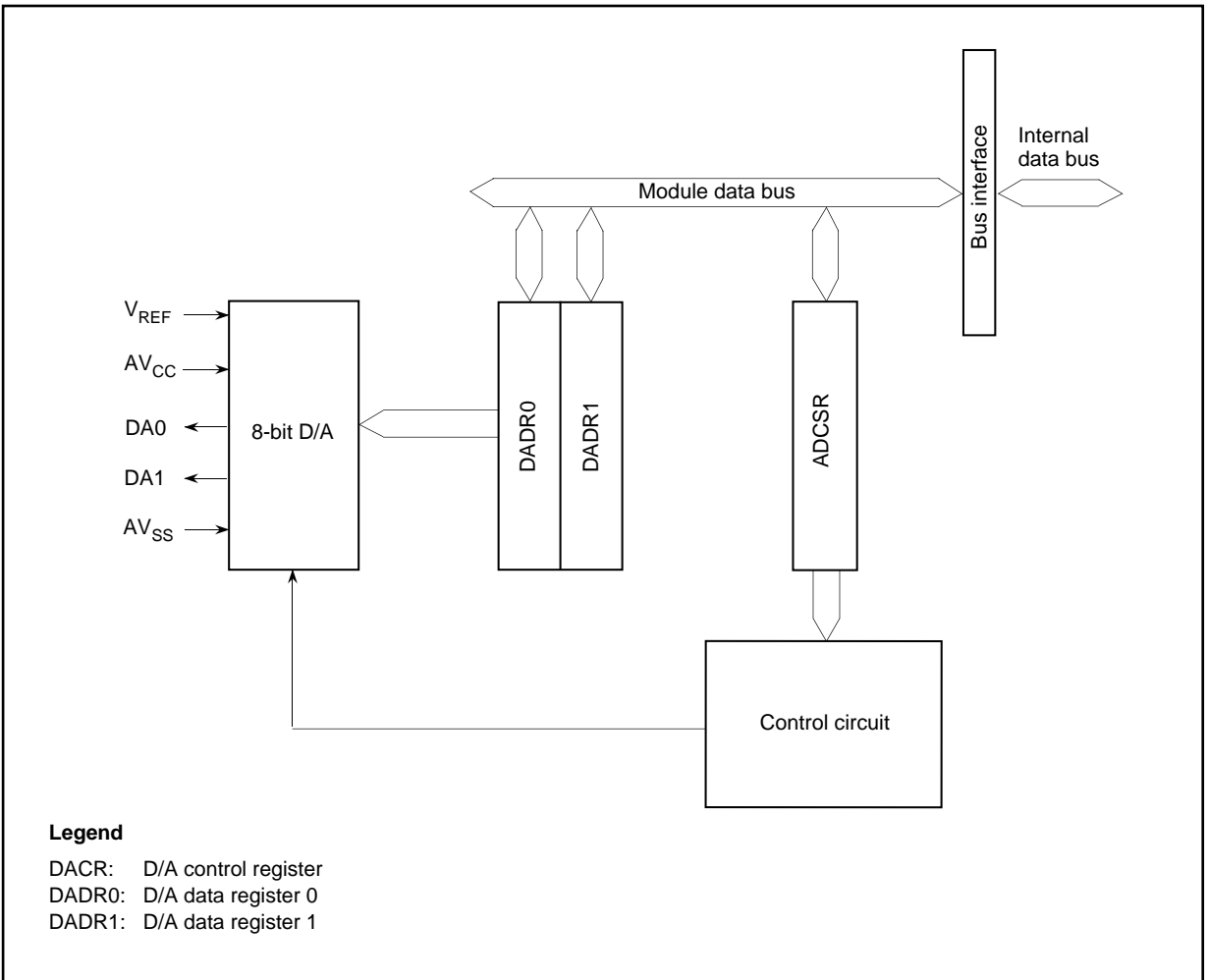
## 6.10 D/A Converter

The H8/3048 Series has a built-in 8-bit D/A converter that can be programmed to output analog signals on a maximum two channels.

### Features

- Eight-bit resolution
- Two analog output pins
- Conversion time: maximum 10  $\mu$ s
- Output voltage: 0 V to  $V_{REF}$
- D/A outputs maintained during software standby

### Block Diagram



## Operation

Setting a D/A output enable bit to 1 enables the D/A converter. The data-register (DADR) contents are converted to a signal that is output constantly from the corresponding pin. The output voltage is:

$$\frac{\text{DADR contents}}{256} \times V_{\text{REF}}$$

## 6.11 I/O Ports

The H8/3048 Series has eleven input/output ports (ports 1, 2, 3, 4, 5, 6, 7, 8, 9, A, and B), the functions of which are indicated in the following table. The pins in each port are multiplexed so that they can also be used as input/output pins of the on-chip supporting modules.

Each port has a data direction register (DDR) for selecting input or output, and a data register (DR) for storing output data. In addition to their DDR and DR, ports 4 and 5 have an input pull-up control register (PCR) for switching input pull-up transistors on and off.

Port	Description	Pins	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
Port 1	<ul style="list-style-type: none"> <li>8-bit I/O port</li> <li>Can drive LEDs</li> </ul>	P1 <sub>7</sub> to P1 <sub>0</sub> / A <sub>7</sub> to A <sub>0</sub>	Address output pins (A <sub>7</sub> to A <sub>0</sub> )				Address output (A <sub>7</sub> to A <sub>0</sub> ) and generic input DDR = 0: generic input DDR = 1: address output		Generic input/output
Port 2	<ul style="list-style-type: none"> <li>8-bit I/O port</li> <li>Input pull-up on-chip</li> <li>Can drive LEDs</li> </ul>	P2 <sub>7</sub> to P2 <sub>0</sub> / A <sub>15</sub> to A <sub>8</sub>	Address output pins (A <sub>15</sub> to A <sub>8</sub> )				Address output (A <sub>15</sub> to A <sub>8</sub> ) and generic input DDR = 0: generic input DDR = 1: address output		Generic input/output
Port 3	• 8-bit I/O port	P3 <sub>7</sub> to P3 <sub>0</sub> / D <sub>15</sub> to D <sub>8</sub>	Data input/output (D <sub>15</sub> to D <sub>8</sub> )						Generic input/output
Port 4	<ul style="list-style-type: none"> <li>8-bit I/O port</li> <li>Input pull-up on-chip</li> </ul>	P4 <sub>7</sub> to P4 <sub>0</sub> / D <sub>7</sub> to D <sub>0</sub>	Data input/output (D <sub>7</sub> to D <sub>0</sub> ) and 8-bit generic input/output 8-bit bus mode: generic input/output 16-bit bus mode: data input/output						Generic input/output
Port 5	<ul style="list-style-type: none"> <li>4-bit I/O port</li> <li>Input pull-up on-chip</li> <li>Can drive LEDs</li> </ul>	P5 <sub>3</sub> to P5 <sub>0</sub> / A <sub>19</sub> to A <sub>16</sub>	Address output (A <sub>19</sub> to A <sub>16</sub> )				Address output (A <sub>19</sub> to A <sub>16</sub> ) and generic input DDR = 0: generic input DDR = 1: address output		Generic input/output
Port 6	• 7-bit I/O port	P6 <sub>6</sub> /LWR P6 <sub>5</sub> /HWR P6 <sub>4</sub> /RD P6 <sub>3</sub> /AS	Bus control signal output (LWR, HWR, RD, AS)						Generic input/output
		P6 <sub>2</sub> /BACK P6 <sub>1</sub> /BREQ P6 <sub>0</sub> /WAIT	Bus control signal input/output (BACK, BREQ, WAIT) and 3-bit generic input/output						
Port 7	• 8-bit I/O port	P7 <sub>7</sub> /AN <sub>7</sub> /DA <sub>1</sub> P7 <sub>6</sub> /AN <sub>6</sub> /DA <sub>0</sub>	Analog input (AN <sub>7</sub> , AN <sub>6</sub> ) to A/D converter, analog output (DA <sub>1</sub> , DA <sub>0</sub> ) from D/A converter, and generic input						
		P7 <sub>5</sub> to P7 <sub>0</sub> / AN <sub>5</sub> to AN <sub>0</sub>	Analog input (AN <sub>5</sub> to AN <sub>0</sub> ) to A/D converter, and generic input						
Port 8	<ul style="list-style-type: none"> <li>5-bit I/O port</li> <li>P8<sub>2</sub> to P8<sub>0</sub> have Schmitt inputs</li> </ul>	P8 <sub>4</sub> / $\overline{CS_0}$	DDR = 0: generic input DDR = 1 (reset value): $\overline{CS_0}$ output						Generic input/output
		P8 <sub>3</sub> / $\overline{CS_1}$ / $\overline{IRQ_3}$ P8 <sub>2</sub> / $\overline{CS_2}$ / $\overline{IRQ_2}$ P8 <sub>1</sub> / $\overline{CS_3}$ / $\overline{IRQ_1}$	$\overline{IRQ_3}$ to $\overline{IRQ_1}$ input, $\overline{CS_1}$ to $\overline{CS_3}$ output, and generic input DDR = 0 (reset value): generic input DDR = 1: $\overline{CS_1}$ to $\overline{CS_3}$ output						
		P8 <sub>0</sub> /RFSH/ $\overline{IRQ_0}$	$\overline{IRQ_0}$ input, RFSH output, and generic input/output						
Port 9	• 6-bit I/O port	P9 <sub>5</sub> /SCK <sub>1</sub> / $\overline{IRQ_5}$ P9 <sub>4</sub> /SCK <sub>0</sub> / $\overline{IRQ_4}$ P9 <sub>3</sub> /Rx <sub>D1</sub> P9 <sub>2</sub> /Rx <sub>D0</sub> P9 <sub>1</sub> /Tx <sub>D1</sub> P9 <sub>0</sub> /Tx <sub>D0</sub>	Input and output (SCK <sub>1</sub> , SCK <sub>0</sub> , Rx <sub>D1</sub> , Rx <sub>D0</sub> , Tx <sub>D1</sub> , Tx <sub>D0</sub> ) for serial communication interfaces 0 and 1 (SCI0/1), $\overline{IRQ_5}$ and $\overline{IRQ_4}$ input, and 6-bit generic input/output						

Port	Description	Pins	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7	
Port A	<ul style="list-style-type: none"> <li>8-bit I/O port</li> <li>Schmitt inputs</li> </ul>	PA <sub>7</sub> /TP <sub>7</sub> /TIOCB <sub>2</sub> /A <sub>20</sub>	Output (TP <sub>7</sub> ) from programmable timing pattern controller (TPC), input or output (TIOCB <sub>2</sub> ) for 16-bit integrated timer unit (ITU), and generic input/output		Address output (A <sub>20</sub> )		TPC output (TP <sub>7</sub> ), ITU input or output (TIOCB <sub>2</sub> ), and generic input/output	Address output (A <sub>20</sub> )	TPC output (TP <sub>7</sub> ), ITU input or output (TIOCB <sub>2</sub> ), and generic input/output	
		PA <sub>6</sub> /TP <sub>6</sub> /TIOCA <sub>2</sub> /A <sub>21</sub> /CS <sub>4</sub> PA <sub>5</sub> /TP <sub>5</sub> /TIOCB <sub>1</sub> /A <sub>22</sub> /CS <sub>5</sub> PA <sub>4</sub> /TP <sub>4</sub> /TIOCA <sub>1</sub> /A <sub>23</sub> /CS <sub>6</sub>	TPC output (TP <sub>6</sub> to TP <sub>4</sub> ), ITU input and output (TIOCA <sub>2</sub> , TIOCB <sub>1</sub> , TIOCA <sub>1</sub> ), CS <sub>4</sub> to CS <sub>6</sub> output, and generic input/output		TPC output (TP <sub>6</sub> to TP <sub>4</sub> ), ITU input and output (TIOCA <sub>2</sub> , TIOCB <sub>1</sub> , TIOCA <sub>1</sub> ), address output (A <sub>23</sub> to A <sub>21</sub> ), CS <sub>4</sub> to CS <sub>6</sub> output, and generic input/output		TPC output (TP <sub>6</sub> to TP <sub>4</sub> ), ITU input and output (TIOCA <sub>2</sub> , TIOCB <sub>1</sub> , TIOCA <sub>1</sub> ), CS <sub>4</sub> to CS <sub>6</sub> output, and generic input/output		TPC output (TP <sub>6</sub> to TP <sub>4</sub> ), ITU input and output (TIOCA <sub>2</sub> , TIOCB <sub>1</sub> , TIOCA <sub>1</sub> ), address output (A <sub>23</sub> to A <sub>21</sub> ), CS <sub>4</sub> to CS <sub>6</sub> output, and generic input/output	TPC output (TP <sub>6</sub> to TP <sub>4</sub> ), ITU input and output (TIOCA <sub>2</sub> , TIOCB <sub>1</sub> , TIOCA <sub>1</sub> ), and generic input/output
		PA <sub>3</sub> /TP <sub>3</sub> /TIOCB <sub>0</sub> /TCLKD PA <sub>2</sub> /TP <sub>2</sub> /TIOCA <sub>0</sub> /TCLKC PA <sub>1</sub> /TP <sub>1</sub> /TEND <sub>1</sub> /TCLKB PA <sub>0</sub> /TP <sub>0</sub> /TEND <sub>0</sub> /TCLKA	TPC output (TP <sub>3</sub> to TP <sub>0</sub> ), output (TEND <sub>1</sub> , TEND <sub>0</sub> ) from DMA controller (DMAC), ITU input and output (TCLKD, TCLKC, TCLKB, TCLKA, TIOCB <sub>0</sub> , TIOCA <sub>0</sub> ), and generic input/output							
Port B	<ul style="list-style-type: none"> <li>8-bit I/O port</li> <li>Can drive LEDs</li> <li>PB<sub>3</sub> to PB<sub>0</sub> have Schmitt inputs</li> </ul>	PB <sub>7</sub> /TP <sub>15</sub> /DREQ <sub>1</sub> /ADTRG	TPC output (TP <sub>15</sub> ), DMAC input (DREQ <sub>1</sub> ), external trigger input (ADTRG) to A/D converter, and generic input/output							
		PB <sub>6</sub> /TP <sub>14</sub> /DREQ <sub>0</sub> /CS <sub>7</sub>	TPC output (TP <sub>14</sub> ), DMAC input (DREQ <sub>0</sub> ), CS <sub>7</sub> output, and generic input/output						TPC output (TP <sub>14</sub> ), DMAC input (DREQ <sub>0</sub> ), and generic input/output	
		PB <sub>5</sub> /TP <sub>13</sub> /TOCXB <sub>4</sub> PB <sub>4</sub> /TP <sub>12</sub> /TOCXA <sub>4</sub> PB <sub>3</sub> /TP <sub>11</sub> /TIOCB <sub>4</sub> PB <sub>2</sub> /TP <sub>10</sub> /TIOCA <sub>4</sub> PB <sub>1</sub> /TP <sub>9</sub> /TIOCB <sub>3</sub> PB <sub>0</sub> /TP <sub>8</sub> /TIOCA <sub>3</sub>	TPC output (TP <sub>13</sub> to TP <sub>8</sub> ), ITU input and output (TOCXB <sub>4</sub> , TOCXA <sub>4</sub> , TIOCB <sub>4</sub> , TIOCA <sub>4</sub> , TIOCB <sub>3</sub> , TIOCA <sub>3</sub> ), and generic input/output							

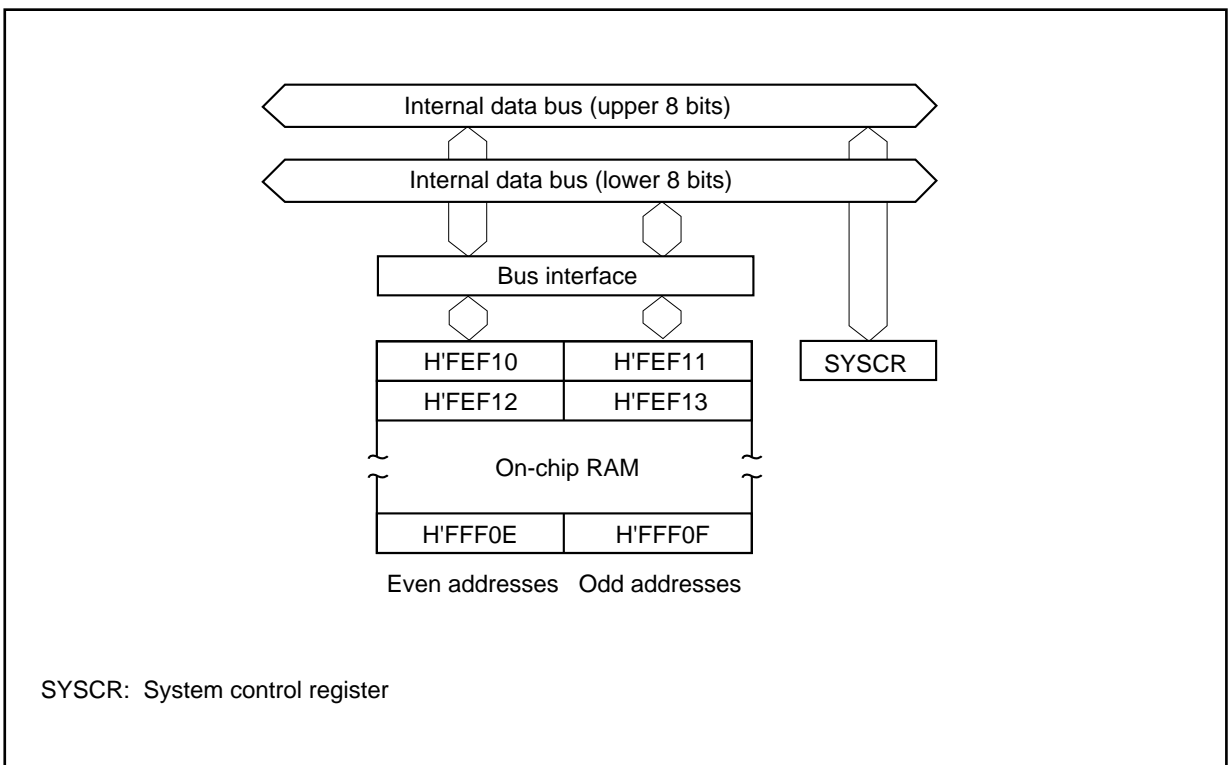
## 6.12 RAM

### Functions

The H8/3048 and H8/3047 have 4 kbytes of high-speed static RAM on-chip. The H8/3044 has 2 kbytes. The RAM is connected to the CPU by a 16-bit data bus. The CPU accesses both byte data and word data in two states, making the RAM useful for rapid transfer of word data and high-speed computation.

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

### Block Diagram



**RAM Block Diagram (H8/3048, Mode 7)**

	<b>RAM Size</b>	<b>Addresses</b>
H8/3048	4 kbytes	H'FEF10 to H'FFF0F
H8/3047	4 kbytes	H'FEF10 to H'FFF0F
H8/3044	2 kbytes	H'FF710 to H'FFF0F

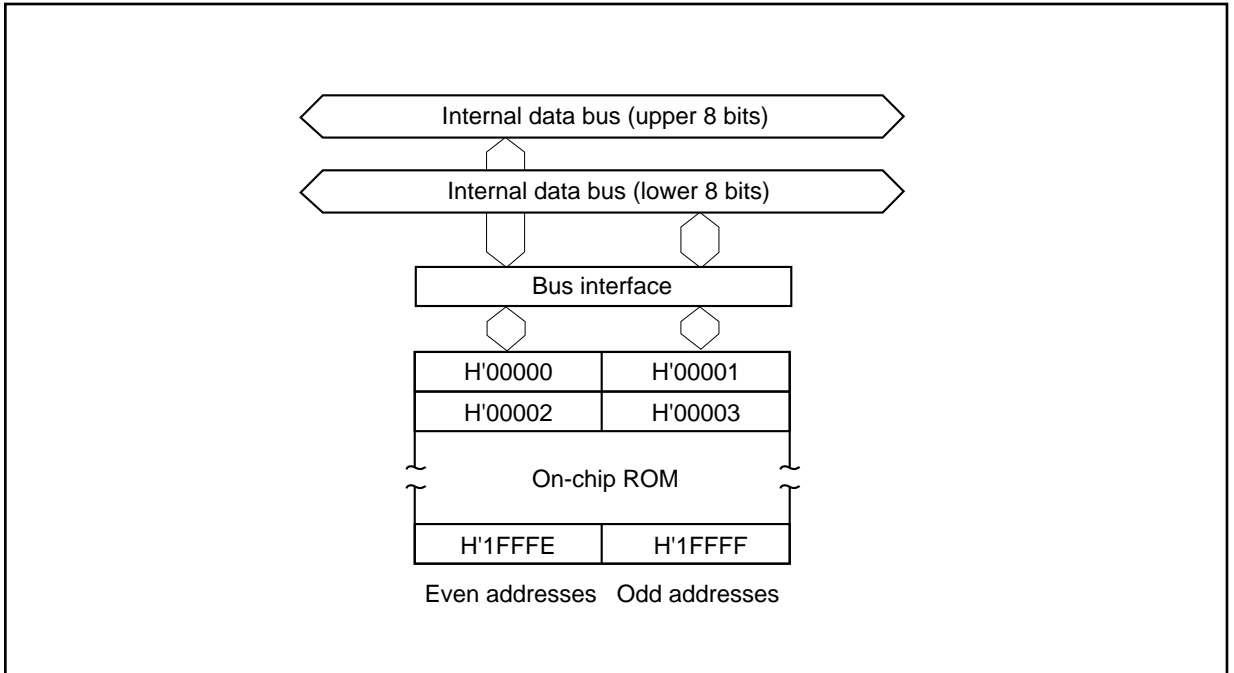
Note: The listed addresses apply in mode 7.

## 6.13 ROM (PROM or Masked ROM)

### Functions

The H8/3048 has 128 kbytes of on-chip PROM or masked ROM. The H8/3047 has 96 bytes of masked ROM. The H8/3044 has 32 kbytes of masked ROM. Like the on-chip RAM, the ROM is connected to the CPU by a 16-bit bus and accessed at high speed in two states.

### Block Diagram

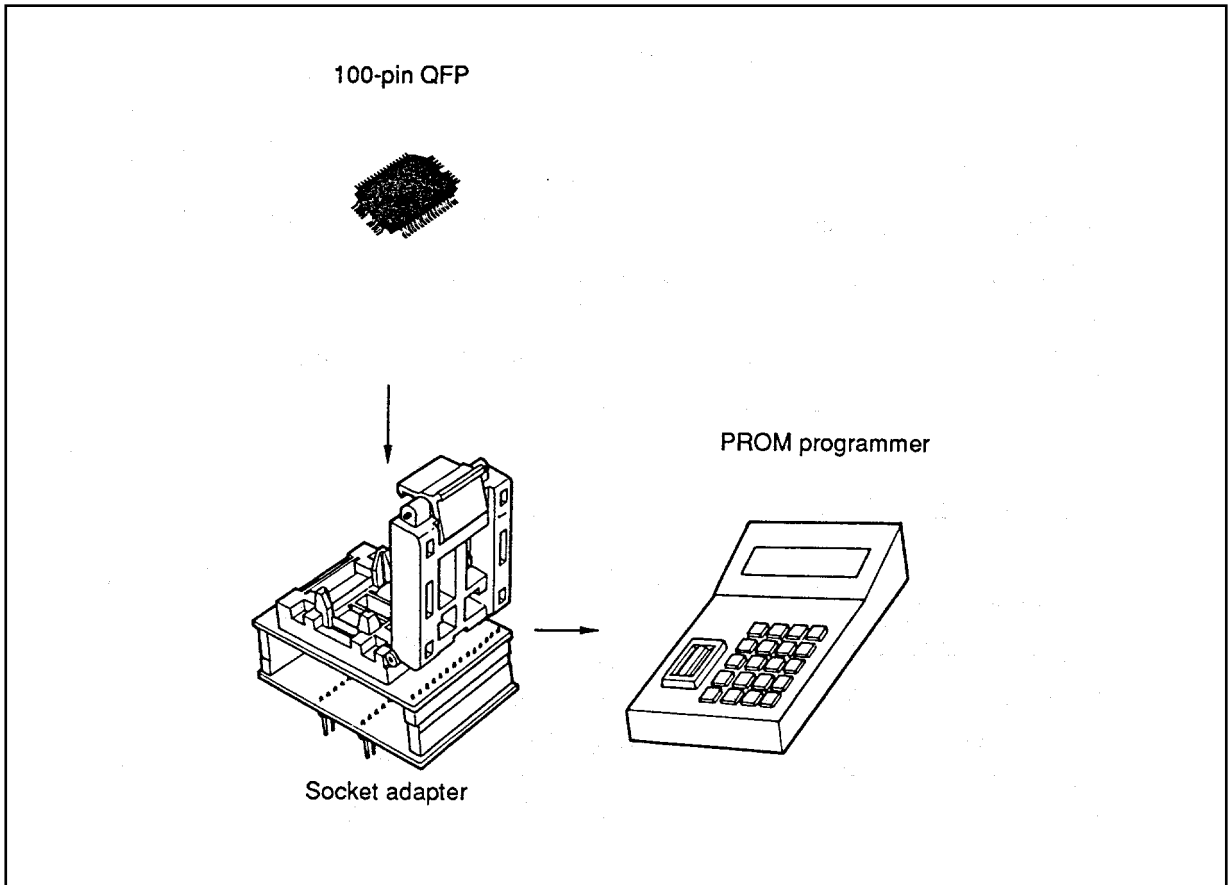


**ROM Block Diagram (H8/3048, Mode 7)**

	ROM Size	Addresses	Reserved Addresses	PROM Version
H8/3048	128 kbytes	H'00000 to H'1FFFF	—	Yes
H8/3047	96 kbytes	H'00000 to H'17FFF	H'18000 to H'1FFFF	—
H8/3044	32 kbytes	H'00000 to H'07FFF	H'08000 to H'0FFFF	—

## PROM Programming

When the PROM version of the H8/3048 is placed in PROM mode, it suspends its MCU functions and allows data to be written directly into the on-chip PROM. The PROM can be programmed and read in the same way as the HN27C101 EPROM ( $V_{PP} = 12.5\text{ V}$ ), except that page programming is not supported. A 100-pin/32-pin socket adapter enables the programming to be done with a commercially available PROM programmer.





## Section 7 Power-Down State

Besides its normal program execution state, the H8/3048 Series has a power-down state that reduces power consumption by halting the CPU and clock oscillator. The power-down state includes three modes: sleep mode, software standby mode, and hardware standby mode. Additional functions include module standby, which reduces power consumption in sleep mode, and a gear function that reduces power consumption by dividing the system clock frequency.

### 7.1 Sleep Mode

The CPU enters sleep mode when the SLEEP instruction is executed. In sleep mode CPU operations halt, but the contents of CPU registers are retained. Supporting modules other than the CPU do not halt.

Sleep mode can be exited by a reset or by any interrupt. The chip returns via the exception-handling state to the normal program execution state.

### 7.2 Software Standby Mode

Software standby mode is entered by setting the software standby bit (SSBY) to 1, then executing the SLEEP instruction. In this mode all functions of the CPU, on-chip supporting modules, and clock oscillator halt, but CPU register contents and on-chip RAM data are retained. I/O ports also remain in their existing states.

Software standby mode can be exited by an external interrupt. After waiting for clock oscillation to stabilize, the chip returns via the exception-handling state to the normal program execution state.

This mode reduces power consumption very sharply because the clock oscillator halts.

### 7.3 Hardware Standby Mode

Hardware standby mode is entered when the  $\overline{\text{STBY}}$  pin is driven low. As in software standby mode, all operations halt but CPU register contents and on-chip RAM data are retained.

Hardware standby mode is exited by driving the  $\overline{\text{RES}}$  pin low, then driving  $\overline{\text{STBY}}$  high. The chip starts operating from the reset state.

Like software standby mode, this mode reduces power consumption very sharply because the clock oscillator halts.

## 7.4 Module Standby Function

The module standby function can halt output of the system clock. It can also place the 16-bit integrated timer unit (ITU), serial communication interface (SCI0, SCI1), DMA controller, refresh controller, and A/D converter in an independent standby state. This function enables power consumption to be reduced in sleep mode. The modules to be placed in standby are specified by corresponding register bits.

Mode	Clock	CPU	Refresh Controller	Supporting Functions	RAM	I/O Ports	Exiting Method
Sleep mode	Active	Halted (held)	Active	Active	Held	Held	<p><b>Interrupt:</b> Interrupt is recognized and interrupt exception handling starts.</p> <p><b>RES:</b> Enters reset state.</p> <p><b>STBY:</b> Transition to hardware standby mode.</p>
Software standby mode	Halted	Halted (held)	Halted (held)	Halted (reset)	Held	Held	<p><b>Interrupt:</b> Interrupt starts clock oscillator. After waiting for time set in on-chip timer to let clock stabilize, interrupt exception handling starts automatically.</p> <p><b>RES:</b> Clock oscillator starts and chip enters reset state.</p> <p><b>STBY:</b> Transition to hardware standby mode.</p>
Hardware standby mode	Halted	Halted (not held)	Halted (reset)	Halted (reset)	Held	High impedance	To exit, drive RES low, then drive STBY high, wait for clock to stabilize, and drive RES high.
Module standby function	Not output	—	Standby if corresponding bit is set to 1 in MSTCR		—	—	Clear corresponding bit to 0 in MSTCR

Note: Status of registers is indicated in parentheses. The module standby function is available only for the refresh controller, DMAC, ITU, SCI0, SCI1, and A/D converter.

## 7.5 Gear Function

After duty adjustment, the clock frequency can be divided in the range from 1/1 to 1/8. Power consumption is reduced in approximately direct proportion to the division ratio, which is specified in the division control register (DIVCR).

## **Section 8 System Development Environment**

The following tools are provided for efficient program development.

### **Software**

- Integrated manager
- H8/300 Series C compiler
- H8/300 Series cross-assembler
- Linkage editor (including librarian and object module converter)
- H8/300 Series simulator/debugger
- H8/300H Series realtime operating systems

### **Hardware**

- H8/300H Series realtime emulator (E7000)
- H8/300H Series compact evaluation board

## 8.1 Software

### • Integrated Manager

Efficient working environment: the tools cooperate, from editor to debugger

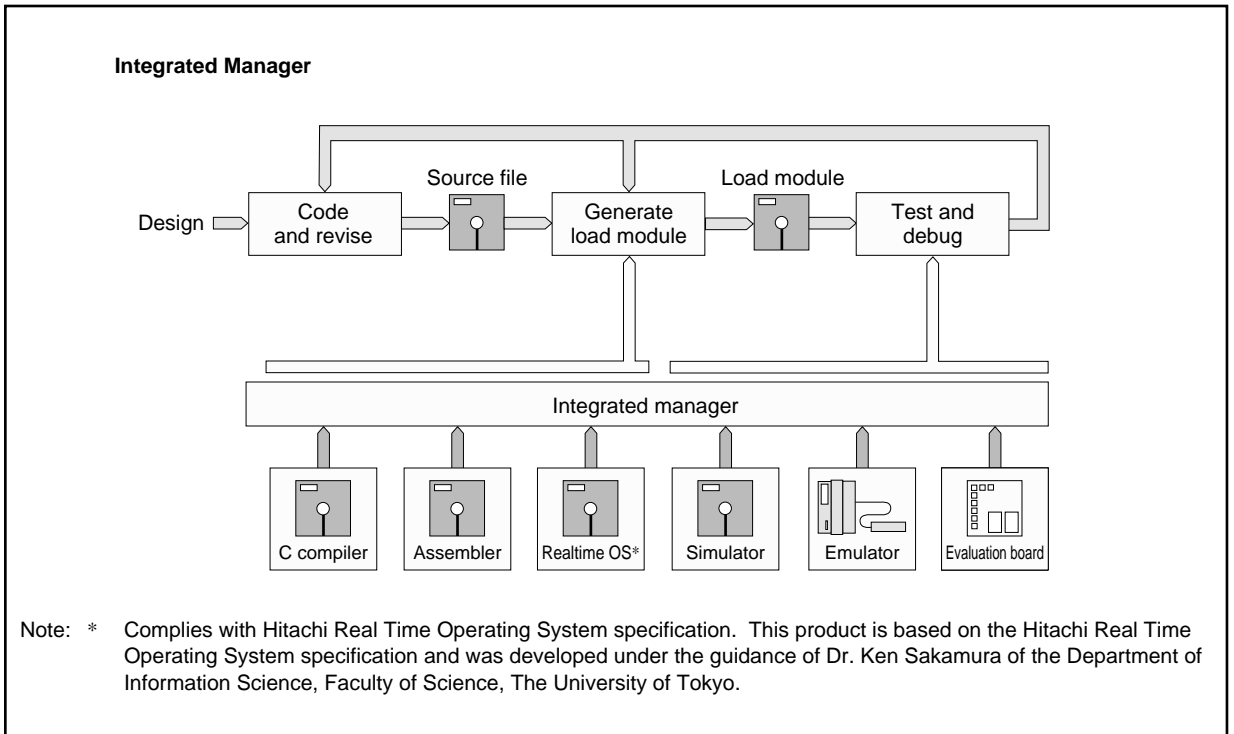
- Assemble/compile errors automatically activate the editor (with the cursor positioned on the line of the error in the source program).
- Automation of assemble/compile, object-module link, and loading to debugger

Simplified, standardized user interface for debugger

- Multi-window support for C-language source-level debug
- Easy menu- and button-driven interface
- Same interface for simulator, emulator, and compact evaluation board
- Multitasking debugger for Hitachi Real Time Operating System

On-line help functions

- Window display of detailed descriptions of the functions of each tool
- Window display of error messages

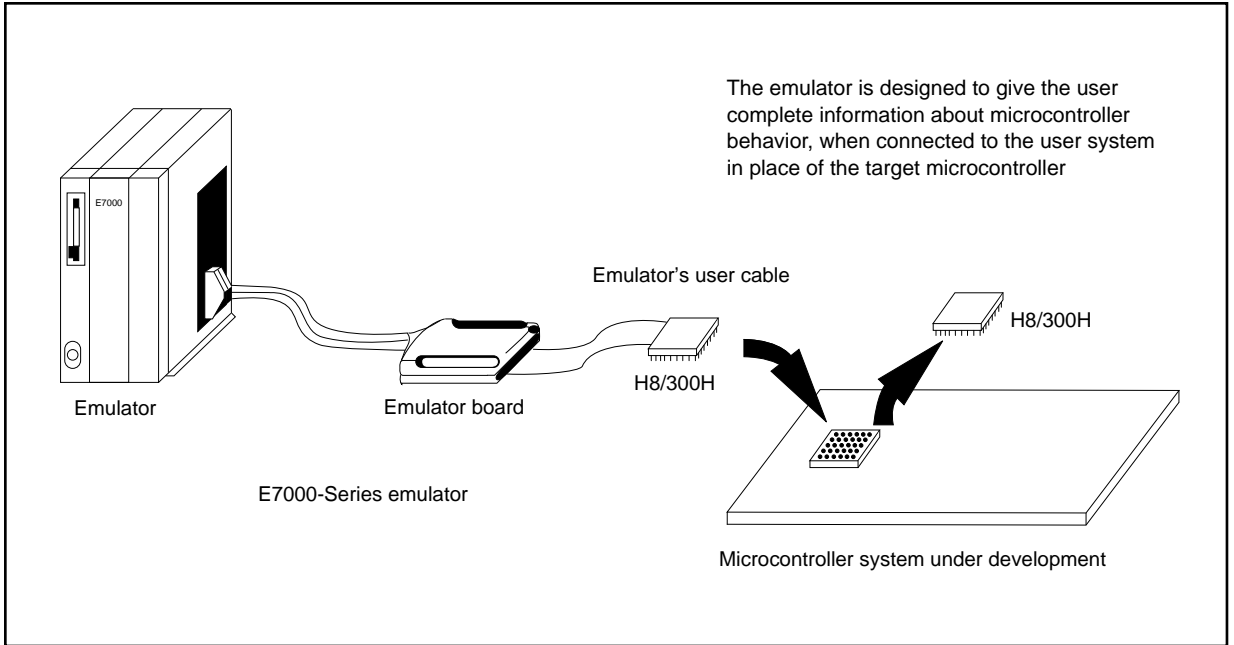


### Integrated Development Environment

- **C compiler**
  - Supports all H8/300, H8/300L, and H8/300H Series microcontrollers
  - Conforms to ANSI (American National Standards Institute) C language draft specifications
  - Optimizing features reduce object program size and execution time
  
- **Assembler**
  - Common H-Series assembly-language specifications
  - Generates output in SYSROF object format
  
- **Linkage editor (including librarian and object module converter)**
  - Linkage editor: links and relocates object programs output (in SYSROF format) by the assembler or C compiler, and generates load modules (in SYSROF format)
  - Librarian: stores user programs in libraries
  - Object module converter: converts load module files output by the linkage editor from SYSROF format to S-type format
  
- **H8/300-Series simulator/debugger**
  - Enables the user to debug programs on a host computer, without a target system
  - Supports C-language source-level debug functions
  - Register-based break/trace
  - Stub and function-call features, enabling programs to be debugged from the initial development stages
  - Permits simulation with relocatable object code
  
- **H8/300H-Series realtime operating system (HI8-3H)**
  - Conforms to the Hitachi Real Time Operating System specification
  - Simplifies porting of operating systems and application programs for other H-series microcontrollers
  - High-speed operating system: task-starting system call executes in 35  $\mu$ s
  - Supports stack sharing between tasks

## 8.2 Hardware

### E7000 Realtime Emulator



- Hardware breakpointing (4 breakpoints)
  - Settable breakpoint conditions  
Address (with bit-mask, range, and net specifications), data bus (with bit mask and net specification),  $\overline{RD}$  and  $\overline{WR}$  signals, external interrupts ( $\overline{NMI}$ ,  $\overline{IRQ_0}$  to  $\overline{IRQ_5}$ ), pass count (maximum 4095), delay count (maximum 32,000 cycles)
  - Sequential breakpointing  
Program execution halts when two to four breakpoints are passed in a specified order
- PC breakpointing (specifiable separately for each Mbyte in 2-Mbyte address space)
  - Settable breakpoint conditions: program counter, pass count

- Realtime trace
  - Information traced: address bus, data bus, external probe signals, control signals
  - Trace information capture
    - Free trace: all information generated during execution is captured
    - Range specification: specified trace capture conditions
    - Stop trace: trace stops when a condition is satisfied
    - Subroutine trace: tracing of execution of a particular subroutine
  - Trace information search: information in the trace buffer can be searched according to specified conditions
  - Trigger conditions: a trigger signal can be output when trigger conditions are satisfied
- C0 coverage function (specifiable separately for each Mbyte in 2-Mbyte address space)
  - Indicates the percent of the area covered in a program run
- Performance analysis
  - Measures program execution efficiency
- Parallel mode
  - Commands can be entered during a program run without halting execution



- E7000 graphical interface software

- Source-level debug

Displays position (PC) where execution stopped in source program

Displays contents of selected variables in source program

Breakpoint setting on selected line in source program

Supports source-program line-mode (step-mode) execution

- Realtime display of trace information

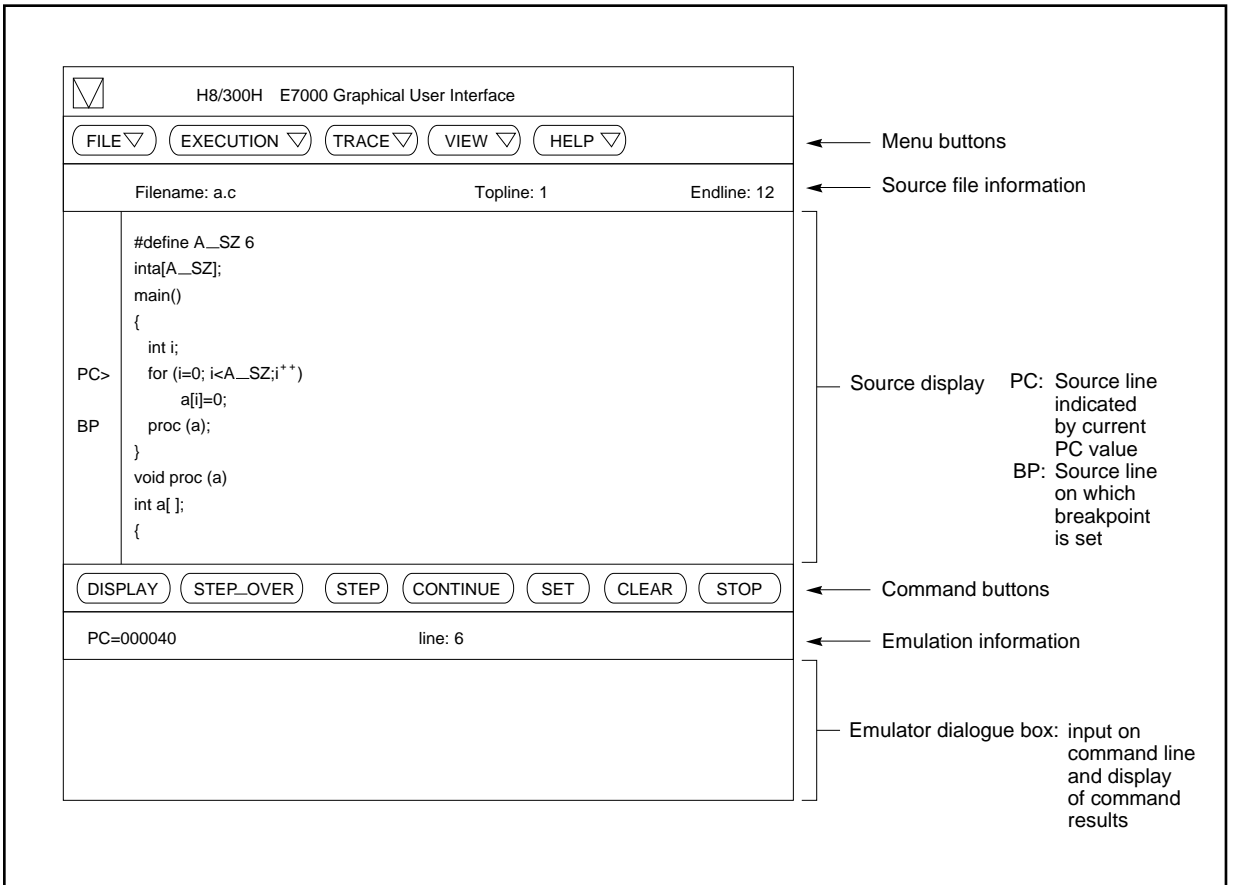
Displays trace information using source program

Waveform display of address bus, data bus, interrupt signals, external probe signals, etc.

- Multiple windows

Connected display of base frame and subframes (example: if register subframe is displayed and program is executed in step mode, register contents display is updated)

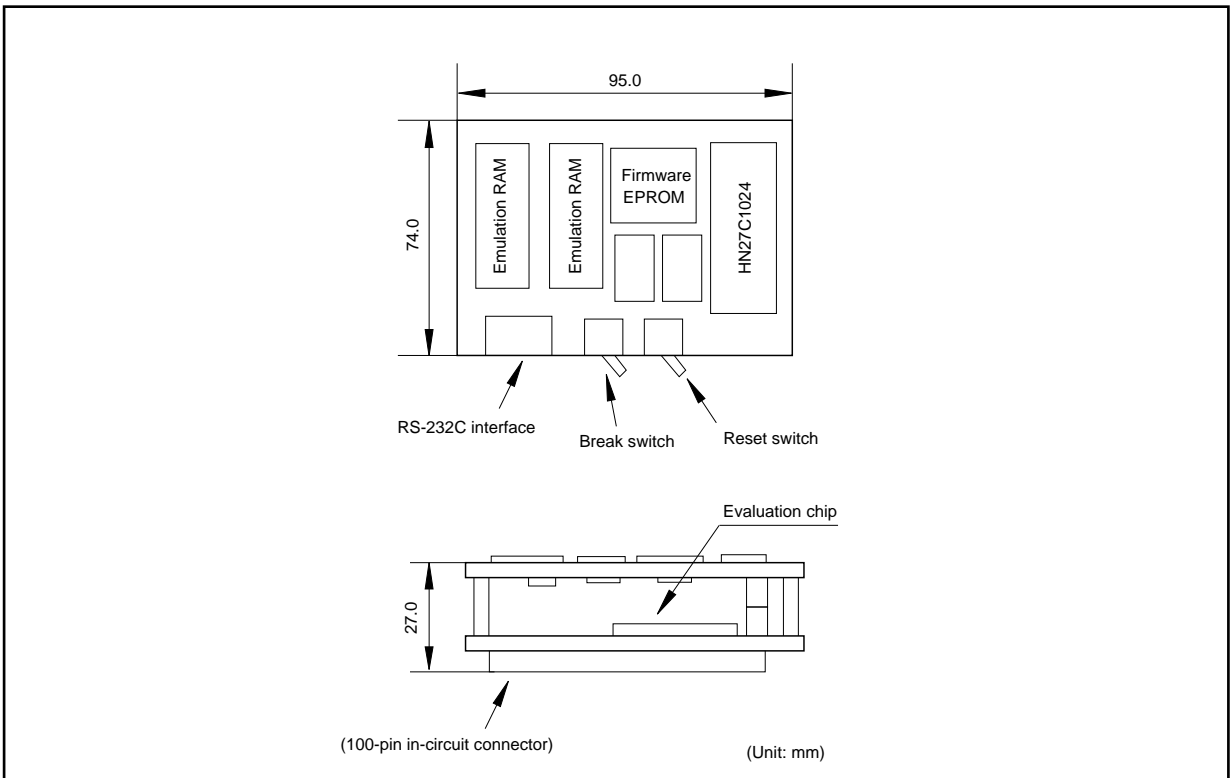
Easy operation by command subframe



**Base Frame of Graphical Interface (Start-Up Window)**

## H8/300H Series Compact Evaluation Board

- Makes all H8/300H Series functions available to the user
- Supports realtime emulation (Max. 18 MHz)
- EPROM socket for emulating on-chip ROM
- On-chip ROM can also be emulated in emulation RAM
- Firmware mapped independently of user address space
  - The entire address space is available to the user
- Built-in terminal interface (RS-232C interface)
  - Supports data transfer with host computer
- Allows battery backup of emulation memory (by external battery)
- Target system connectors
  - Special 100-pin connector
  - Can connect via special adapter board to E7000 emulator's user cable



**H8/300H Series Compact Evaluation Board**