

H8S/2237 Series H8S/2227 Series

Overview

HITACHI

ADE-
Rev. 0.2
Nov. 13, 1997
Hitachi, Ltd.
MC-Setsu



Notice

When using this document, keep the following in mind:

1. This document may, wholly or partially, be subject to change without notice.
2. All rights are reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without Hitachi's permission.
3. Hitachi will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's unit according to this document.
4. Circuitry and other examples described herein are meant merely to indicate the characteristics and performance of Hitachi's semiconductor products. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.
5. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.
6. **MEDICAL APPLICATIONS:** Hitachi's products are not authorized for use in **MEDICAL APPLICATIONS** without the written consent of the appropriate officer of Hitachi's sales company. Such use includes, but is not limited to, use in life support systems. Buyers of Hitachi's products are requested to notify the relevant Hitachi sales offices when planning to use the products in **MEDICAL APPLICATIONS**.

Preface

Hitachi's H8S family of single-chip microcomputers comprises a number of new series offering the high performance and low power consumption of the existing H8 Series, which is widely used for machine control, etc., together with significantly greater ease of use.

The H8S/2000 Series features CPU object-level compatibility with the H8/300H Series, H8/300 Series, and H8/300L Series within the H8 Series.

Series	Features
H8S/2000	Upward-compatible with the H8/300H Series and H8/300 Series; twice the performance at the same frequency
H8/300H	16-Mbyte linear address space; upward-compatible with the H8/300 Series; concise instruction set; powerful word-size and longword-size arithmetic instructions
H8/300	64-kbyte address space; general register system; concise instruction set; powerful bit manipulation instructions
H8/300L	Same CPU as the H8/300 Series; consumer application oriented supporting modules; low voltage, low power consumption

This document gives an overview of the new H8S/2000 Series products, which is suitable for single chip application in the H8S Series.

Intended Readership: This Overview is intended for readers who have a basic understanding of microcomputers, and are looking for information on the features and functions of the H8S/2237, and H8S/2227 Series. Readers undertaking system design using these products, or requiring more detailed information on their use, should refer to the relevant Hardware Manuals and the H8S/2600 and H8S/2000 Series Programming Manual.

Related Documents

Contents	Title	Document No.
H8S/2237 Series and H8S/2227 Series hardware	H8S/2237 Series and H8S/2227 Series Hardware Manual	TBD (Scheduled publication: 5/98)
H8S/2000 Series execution instructions	H8S/2600 Series and H8S/2000 Series Programming Manual	ADE-602-083A

The product specifications in this Overview are subject to change without notice. The relevant Hardware Manual must be used when undertaking product design.

HITACHI

On-Chip Supporting Modules

Series	H8S/2237 Series	H8S/2227 Series
Product names	H8S/2237, 2235, 2233	H8S/2227, 2225, 2223
Bus controller (BSC)	Available (16-bit)	Available (16-bit)
Data transfer controller (DTC)	Available	Available
16-bit timer pulse unit (TPU)	×6	×3
8-bit timer (TMR)	×2	×2
Watchdog timer (WDT)	×2	×2
Serial communication interface (SCI)	×4	×3
A/D converter	×8	×8
D/A converter	×2	—
PC break controller	×2	×2

HITACHI

Contents

Section 1	Features of H8S/2237 Series and H8S/2227 Series	1
1.1	Features of H8S/2237 Series and H8S/2227 Series	1
1.1.1	High-performance H8S/2000 CPU	1
1.2	Pin Arrangement and Functions	5
1.2.1	H8S/2237 Series Pin Arrangement	5
1.2.2	H8S/2227 Series Pin Arrangement	7
1.3	Internal Block Diagram	11
Section 2	CPU	13
2.1	Overview.....	13
2.1.1	Features	13
2.2	Register Configuration	16
2.3	Data Formats.....	19
2.4	Addressing Modes	22
2.5	Instruction Set.....	26
2.6	Basic Timing.....	38
2.7	Processing States	42
2.8	Exception Handling	44
2.9	Interrupts.....	46
2.10	MCU Operating Modes	51
2.11	Address Maps	53
Section 3	Supporting Modules.....	56
3.1	PC Break Controller (PBC)	56
3.1.1	Features	56
3.2	Bus Controller (BSC)	58
3.2.1	Features	58
3.2.2	Area Partitioning	60
3.2.3	Bus Specifications	61
3.2.4	Memory Interfaces.....	61
3.3	Data Transfer Controller (DTC)	66
3.3.1	Features	66
3.3.2	Data Transfer Operation	68
3.4	16-Bit Timer Pulse Unit (TPU)	78
3.4.1	Features	78
3.4.2	Interrupt Sources and Data Transfer Controller (DTC) Activation.....	82
3.4.3	Operation	83
3.4.4	PWM Modes	84
3.4.5	Input Capture Operation	85

3.5	8-Bit Timer (TMR).....	91
3.5.1	Features	91
3.5.2	Interrupt Source and Data Transfer Controller (DTC) Activation	93
3.5.3	Example of Pulse Output.....	93
3.6	Watchdog Timer (WDT)	94
3.6.1	Features	94
3.6.2	Block Diagram.....	95
3.7	Serial Communication Interface (SCI).....	99
3.7.1	Features	99
3.7.2	Block Diagram.....	100
3.8	Smart Card Interface.....	107
3.8.1	Features	107
3.8.2	Operation	108
3.8.3	Schematic Connection Diagram	109
3.8.4	Data Format	109
3.9	A/D Converter	110
3.9.1	Features	110
3.10	D/A Converter	113
3.10.1	Features	113
3.10.2	Operation	113
3.10.3	D/A Converter Block Diagram	114
3.11	I/O Ports.....	114
3.12	RAM.....	121
3.12.1	Features	122
3.13	ROM.....	123
Section 4 Power-Down Modes		124
Section 5 Development Environment.....		129
5.1	Development Environment.....	129
5.1.1	Lineup.....	129
5.2	Cross Software.....	130
5.3	Emulators	132
5.4	Socket Adapters	134
5.5	HI Series OS	135
Appendix		137
A.1	Packages	137

Figures

Figure 1.1	100-pin plastic TQFP (TFP100B, TFP-100G), 100-pin plastic QFP (FP-100B) ..	5
Figure 1.2	100-pin plastic QFP (FP-100A).....	6
Figure 1.3	100-pin plastic TQFP (TFP-100B, TFP-100G), 100-pin plastic QFP (FP-100B)..	7

Figure 1.4	100-pin plastic QFP (FP-100A).....	8
Figure 1.5	Internal Block Diagram of H8S/2237 Series	11
Figure 1.6	Internal Block Diagram of H8S/2227 Series	12
Figure 2.1	CPU Registers	16
Figure 2.2	Usage of General Registers	17
Figure 2.3	General Register Data Formats	19
Figure 2.4	Memory Data Formats	21
Figure 2.5	Basic Clock Timing	38
Figure 2.6	On-Chip Memory Access Cycle (One-state Access).....	39
Figure 2.7	Pin States During On-Chip Memory Access	39
Figure 2.8	On-Chip Supporting Module Access Cycle (Two-state Access).....	40
Figure 2.9	Pin States during On-Chip Supporting Module Access.....	41
Figure 2.10	State Transition Diagram	43
Figure 2.11	Block Diagram of Interrupt Controller	46
Figure 2.12	Block Diagram of Interrupt Control Operation	47
Figure 2.13	H8S/2237 and H8S/2227 Address Map in Each Operating Mode	53
Figure 2.14	H8S/2235 and H8S/2225 Address Map in Each Operating Mode	54
Figure 2.15	H8S/2233 and H8S/2223 Address Map in Each Operating Mode	55
Figure 3.1	PBC Block Diagram	57
Figure 3.2	Bus Controller Block Diagram	59
Figure 3.3	Overview of Area Partitioning.....	60
Figure 3.4	Basic Bus Timing (Word Access to 16-Bit 2-State Access Space).....	62
Figure 3.5	Basic Bus Timing (Word Access to 16-Bit 3-State Access Space).....	63
Figure 3.6	Example of Burst ROM Access Timing (When AST0 = BRSTS1 = 1)	64
Figure 3.7	Example of Burst ROM Access Timing (When AST0 = BRSTS1 = 0)	65
Figure 3.8	DTC Block Diagram	67
Figure 3.9	Flowchart of DTC Operation.....	68
Figure 3.10	Location of DTC Register Information in Address Space.....	69
Figure 3.11	DTC Operation Timing (Example for Normal and Repeat Modes)	72
Figure 3.12	Operation in Normal Mode.....	75
Figure 3.13	Operation in Repeat Mode.....	76
Figure 3.14	Operation in Block Transfer Mode	77
Figure 3.15	H8S/2237 Series TPU Block Diagram	80
Figure 3.16	H8S/2227 Series TPU Block Diagram	81
Figure 3.17	Example of 0 Output/1 Output Operation	83
Figure 3.18	Example of Toggle Output Operation	84
Figure 3.19	Operation in PWM Mode 1	84
Figure 3.20	Operation in PWM Mode 2	85
Figure 3.21	Input Capture Operation	86
Figure 3.22	Example of Operation in Phase Counting Mode 1	86
Figure 3.23	Example of Buffer Operation (1) (When TGR Is an Output Compare Register)...	88
Figure 3.24	Example of Buffer Operation (2) (When TGR Is an Input Capture Register).....	89
Figure 3.25	Example of Cascaded Operation (32-Bit Input Capture Operation)	90

Figure 3.26	8-Bit Timer Block Diagram.....	92
Figure 3.27	Example of Pulse Output.....	93
Figure 3.28	Block Diagram of WDT0.....	95
Figure 3.29	Block Diagram of WDT1.....	96
Figure 3.30	Operation in Watchdog Timer Mode.....	97
Figure 3.31	Interval Timer Operation.....	98
Figure 3.32	Block Diagram of SCI.....	100
Figure 3.33	Example of Inter-Processor Communication Using Multiprocessor Format (Transmission of Data H'AA to Receiving Station A)	105
Figure 3.34	Data Format in Synchronous Communication.....	105
Figure 3.35	Smart Card Interface Block Diagram	108
Figure 3.36	Schematic Diagram of Smart Card Interface Pin Connections	109
Figure 3.37	Smart Card Interface Data Format.....	109
Figure 3.38	A/D Converter Block Diagram	111
Figure 3.39	Block Diagram of D/A Converter.....	114
Figure 3.40	Block Diagram of RAM (Example of H8S/2237, H8S/2227).....	121
Figure 3.41	ROM Block Diagram (Example of H8S/2237 in Modes 6 and 7).....	123
Figure 4.1	Mode Transitions	126
Figure 5.1	Method of Use	134
Figure 5.2	H18-2600 System Configuration Diagram.....	135

Tables

Table 1.1	Four MCU Operating Modes.....	4
Table 1.2	Product Lineup (preliminary)	4
Table 1.3	Pin Functions	9
Table 2.1	Addressing Modes	22
Table 2.2	Number of States per Cycle.....	36
Table 2.3	Condition Code Notation.....	36
Table 2.4	Operation Notation.....	37
Table 2.5	Exception Types and Priority.....	44
Table 2.6	Exception Vector Table	45
Table 2.7	Interrupt Control Modes	47
Table 2.8	Interrupt Sources, Vector Addresses, and Interrupt Priorities	48
Table 2.9	MCU Operating Mode Selection	52
Table 3.1	Bus Specifications for Each Area (Basic Bus Interface).....	61
Table 3.2	Interrupt Sources, DTC Vector Addresses, and Corresponding DTCEs.....	70
Table 3.3	Number of DTC Execution States	72
Table 3.4	Number of States Required in Each Execution State.....	73
Table 3.5	8-Bit Timer Interrupts	93
Table 3.6	SCI Interrupt Sources and Data Transfer Controller (DTC) Activation.....	101
Table 3.7	Serial Transfer Formats (Asynchronous Mode)	103
Table 3.8	BRR Settings for Various Bit Rates (Clocked Synchronous Mode)	106
Table 3.9	H8S/2237 Series Port Functions in Each Operating Mode	115

Table 3.10	H8S/2227 Series Port Functions in Each Operating Mode	118
Table 4.1	Internal States in Each Mode.....	127
Table 4.2	Power-Down Mode Transition Conditions.....	128
Table 5.1	Cross Software Product Lineup	131
Table 5.2	Emulator Product Lineup.....	133
Table 5.3	Applicable EPROM Programmers.....	134
Table 5.4	HI Series OS Product Lineup.....	136

Section 1 Features of H8S/2237 Series and H8S/2227 Series

1.1 Features of H8S/2237 Series and H8S/2227 Series

H8S/2237 Series and H8S/2227 Series microcomputers are designed for faster instruction execution, using a real-time control oriented CPU with an internal 32-bit architecture, and can run programs based on the C high-level language efficiently. As well as large-capacity ROM and RAM, these microcomputers include comprehensive on-chip supporting modules needed for control systems, simplifying the implementation of sophisticated, high-performance systems.

1.1.1 High-performance H8S/2000 CPU

- General-register architecture
 - Sixteen 16-bit general registers (also usable as sixteen 8-bit registers or eight 32-bit registers)
- High-speed operation suitable for real-time control
 - 10 MHz maximum operating frequency (10 MHz oscillation frequency): ZTAT
 - 13 MHz maximum operating frequency (13 MHz oscillation frequency): Mask ROM
 - High-speed arithmetic operations (at 10 MHz operation)
 - 8/16/32-bit register-register add/subtract: 100 ns
 - 16 × 16-bit register-register multiply: 2000 ns
 - 32 ÷ 16-bit register-register divide: 2000 ns
- Instruction set suitable for high-speed operation
 - Sixty-five types of basic instructions
 - 8/16/32-bit transfer instructions
 - Unsigned/signed multiply and divide instructions
 - Powerful bit manipulation instructions
- CPU operating mode
 - Advanced mode: Maximum 16-Mbyte address space

On-chip byte PROM (Mask ROM)

- 64 kbytes or 128 kbytes

On-chip high-speed static RAM

- 4 kbytes or 16 kbytes

Bus controller

- Address space divided into 8 areas, with bus specifications settable independently for each area
- Chip select output possible for areas 0 to 7
- Selection of 8-bit or 16-bit access space for each area
- 2-state or 3-state access space can be designated for each area
- Number of program wait states can be set for each area
- Burst ROM directly connectable
- External bus release function

Data transfer controller (DTC)

- Activated by internal interrupt or software
- Multiple transfers or multiple types of transfer possible for one activation source
- Transfer possible in repeat mode, block transfer mode, etc.
- Request can be sent to CPU for interrupt that activated DTC

16-bit timer-pulse unit (TPU)

- Six-channel (H8S/2237 Series) or three-channel (H8S/2227 Series) 16-bit timer on-chip
- Pulse I/O processing capability for up to 16 pins (H8S/2237 Series) or 8 pins (H8S/2227 Series)
- Automatic 2-phase encoder count capability

Two on-chip 8-bit timer channels

- 8-bit up-counter (external event count capability)
- Two time constant registers
- Two-channel connection possible

Watchdog timer (WDT: 2 channels)

- Watchdog timer or interval timer function selectable
- Subclock operation capability (channel 1 only)

On-chip serial communication interface (SCI) channels

- Channels
 - H8S/2237 Series: 4 channels (SCI0, SCI1, SCI2, SCI3)
 - H8S/2227 Series: 3 channels (SCI0, SCI1, SCI3)
- Asynchronous mode or synchronous mode selectable
- Multiprocessor communication function
- Smart card interface function

A/D converter

- Resolution: 10 bits
- Input: 8 channels
- High-speed conversion: 13.4 μ s minimum conversion time (10 MHz operation)
- Single or scan mode selectable
- Sample-and-hold function
- A/D conversion can be activated by external trigger or timer trigger

On-chip D/A converter

- Resolution: 8 bits
- Output: 2 channels

PC break controller (PBC: 2 channels)

- Supports debugging functions by means of PC break interrupts
- Two break channels

Eleven I/O ports

- 72 I/O pins, 10 input-only pins

Interrupt controller

- Nine external interrupt pins (NMI, $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ7}}$)
- 53 internal interrupt sources
- Eight priority levels settable

Power-down state

- Medium-speed mode
- Subactive mode
- Subsleep mode
- Watch mode
- Sleep mode
- Module stop mode
- Software standby mode
- Hardware standby mode

Table 1.1 Four MCU Operating Modes

Mode	CPU Operating Mode	Description	On-Chip ROM	External Data Bus	
				Initial Value	Maximum Value
4	Advanced	On-chip ROM disabled expansion mode	Disabled	16 bits	16 bits
5		On-chip ROM disabled expansion mode	Disabled	8 bits	16 bits
6		On-chip ROM enabled expansion mode	Enabled	8 bits	16 bits
7		Single-chip mode	Enabled	—	

On-chip clock pulse generator (1:1 oscillation)

- Built-in duty correction circuit

Packages

- 100-pin plastic TQFP (TFP-100B, TFP-100G)
- 100-pin plastic QFP (FP-100A, FP-100B)

Table 1.2 Product Lineup (preliminary)

Series	Product Name			Package
	Mask ROM Version	ZTAT™ Version	ROM/RAM (Bytes)	
H8S/2237 Series	HD6432237	HD6472237	128 k/16 k	TFP-100B, TFP-100G, FP-100A, FP-100B
	HD6432235	—	128 K/4 k	
	HD6432233	—	64 k/4 k	
H8S/2227 Series	HD6432227	—	128 k/16 k	
	HD6432225	—	128 k/4 k	
	HD6432223	—	64 k/4 k	

Note: ZTAT™ is a trademark of Hitachi Ltd.

1.2 Pin Arrangement and Functions

1.2.1 H8S/2237 Series Pin Arrangement

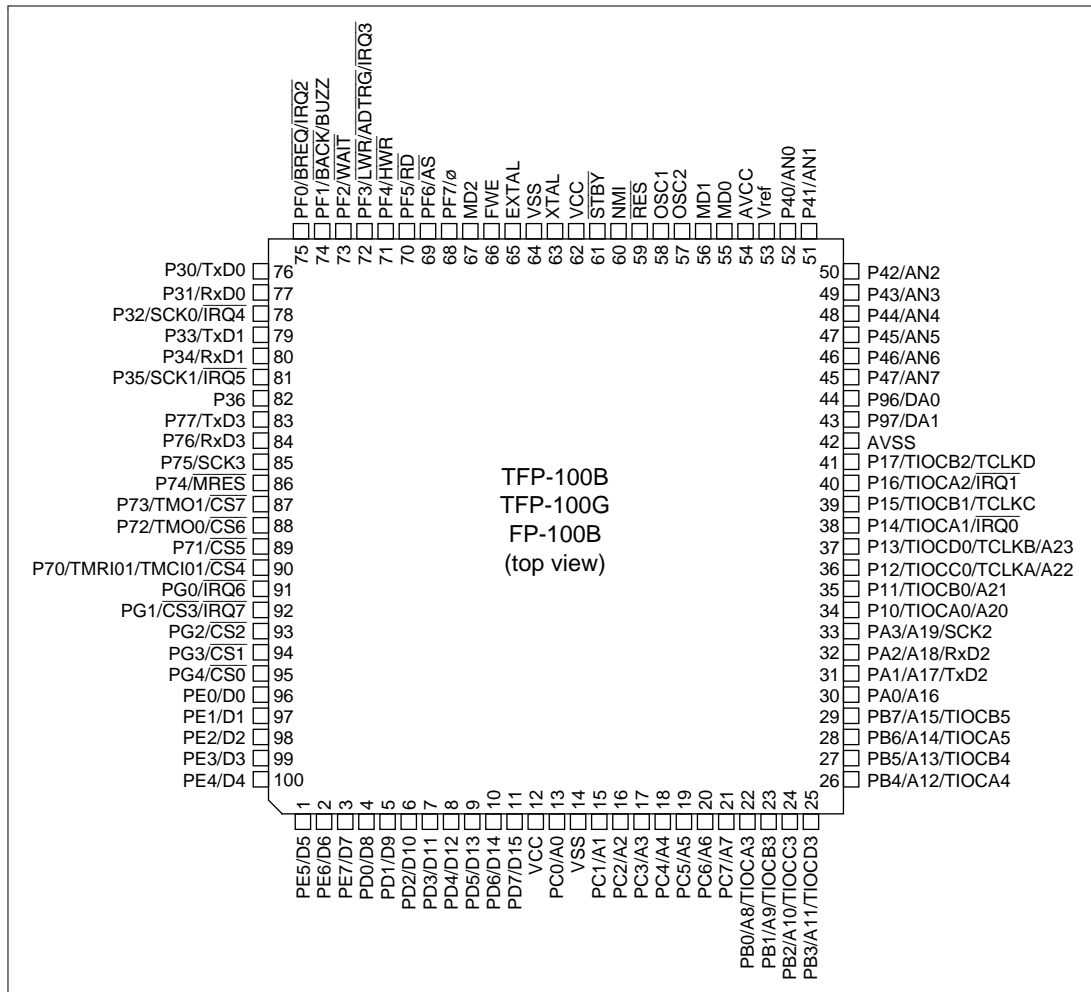


Figure 1.1 100-pin plastic TQFP (TFP100B, TFP-100G), 100-pin plastic QFP (FP-100B)

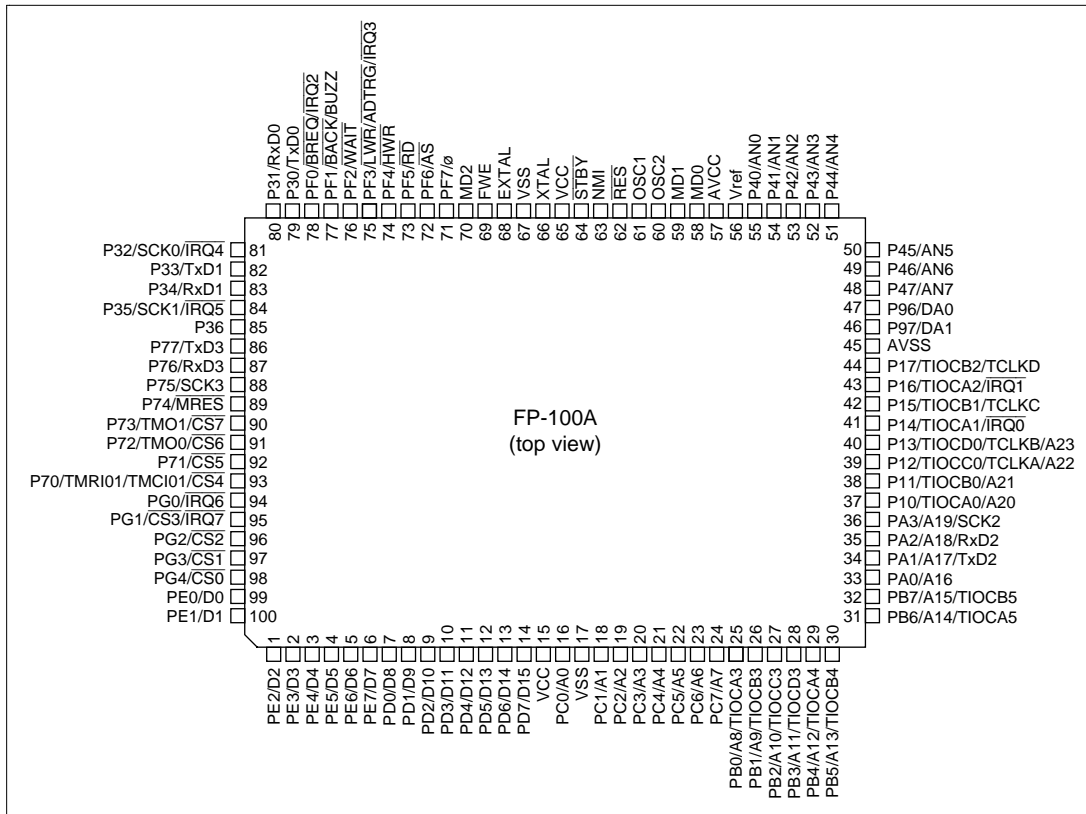


Figure 1.2 100-pin plastic QFP (FP-100A)

1.2.2 H8S/2227 Series Pin Arrangement

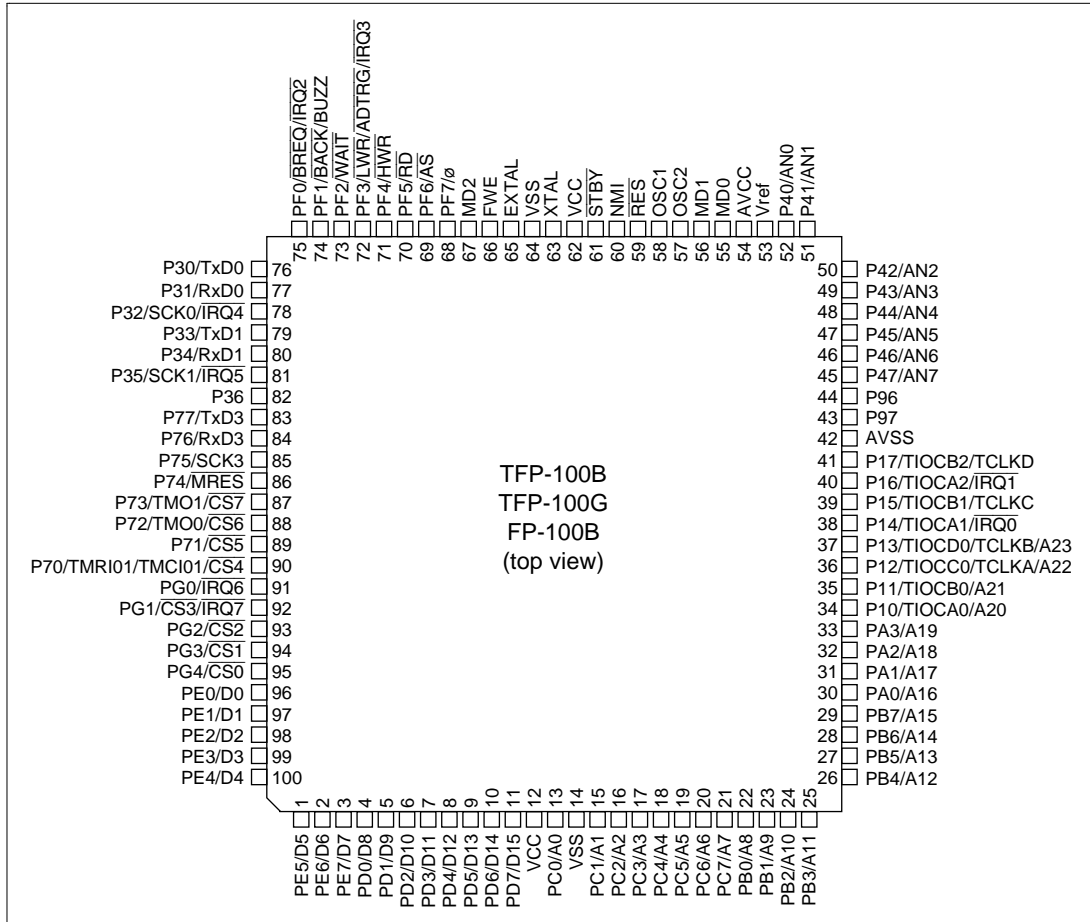


Figure 1.3 100-pin plastic TQFP (TFP-100B, TFP-100G), 100-pin plastic QFP (FP-100B)

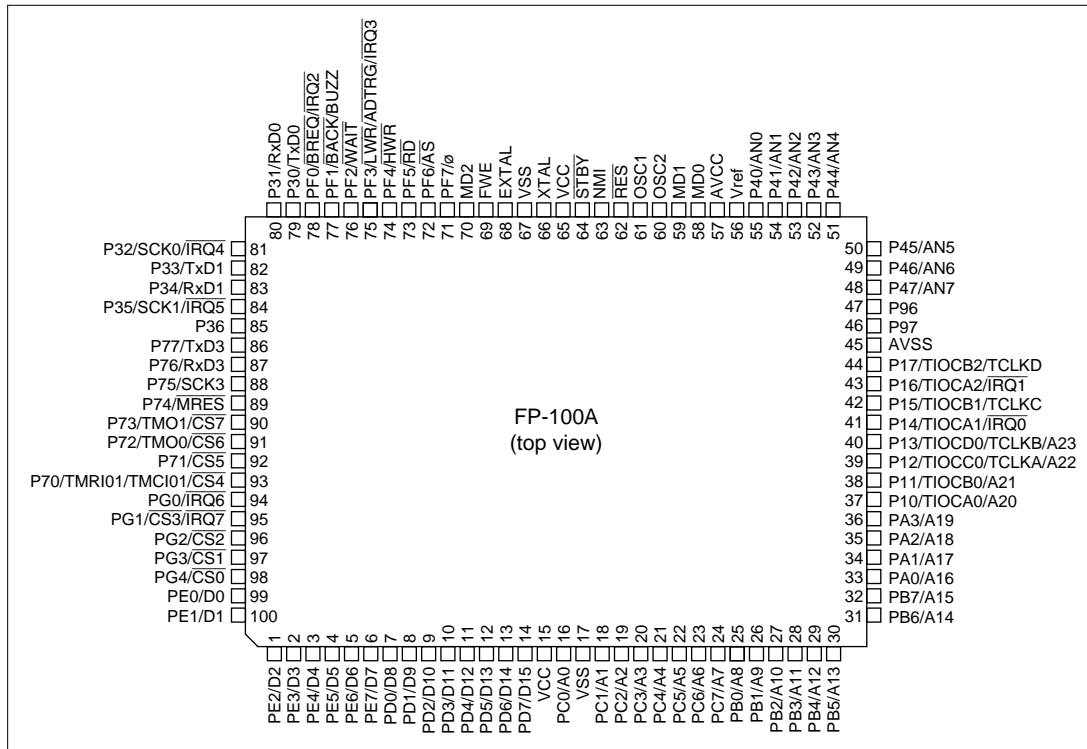


Figure 1.4 100-pin plastic QFP (FP-100A)

Table 1.3 Pin Functions

Type	Symbol	I/O	Name and Function
Power	VCC	Input	Power supply
	VSS	Input	Ground: All VSS pins should be connected to the system power supply (0 V).
Clock	XTAL	Input	Connects to a crystal oscillator.
	EXTAL	Input	Connects to a crystal oscillator, or external clock input.
	OSC1	Input	Connect to a 32.768 kHz crystal oscillator
	OSC2	Input	
	Ø	Output	System clock
Operating mode control	MD2 to MD0	Input	Mode pins
System control	$\overline{\text{RES}}$	Input	Reset input
	$\overline{\text{MRES}}$	Input	Manual reset input
	$\overline{\text{STBY}}$	Input	Standby
	$\overline{\text{BREQ}}$	Input	Bus request
	$\overline{\text{BACK}}$	Output	Bus request acknowledge
	FWE	Input	Flash write enable
Interrupts	NMI	Input	Nonmaskable interrupt
	$\overline{\text{IRQ7}}$ to $\overline{\text{IRQ0}}$	Input	Interrupt request 7 to 0
Address bus	A23 to A0	Output	Address bus
Data bus	D15 to D0	I/O	Data bus
Bus control	$\overline{\text{CS7}}$ to $\overline{\text{CS0}}$	Output	Chip select
	$\overline{\text{AS}}$	Output	Address strobe
	$\overline{\text{RD}}$	Output	Read
	$\overline{\text{HWR}}$	Output	High write
	$\overline{\text{LWR}}$	Output	Low write
	$\overline{\text{WAIT}}$	Input	Wait
16-bit timer-pulse unit (TPU)	TCLKA to TCLKD	Input	Clock input A to D
	TIOCA0, TIOCB0, TIOCC0, TIOCD0	I/O	Input capture/output compare match A0 to D0
	TIOCA1, TIOCB1	I/O	Input capture/output compare match A1 and B1
	TIOCA2, TIOCB2	I/O	Input capture/output compare match A2 and B2

Type	Symbol	I/O	Name and Function
16-bit timer-pulse unit (TPU)	TIOCA3, TIOCB3, TIOCC3, TIOCD3	I/O	Input capture/output compare match A3 to D3
	TIOCA4, TIOCB4	I/O	Input capture/output compare match A4 and B4
	TIOCA5, TIOCB5	I/O	Input capture/output compare match A5 and B5
8-bit timer	TMO0, TMO1	Output	Compare match output
	TMCI0, TMCI1	Input	Counter external clock input
	TMRI0, TMRI1	Input	Counter external reset input
Watchdog timer (WDT)	BUZZ	Output	Buzzer output
Serial communication interface (SCI) Smart Card interface	TxD3, TxD2, TxD1, TxD0	Output	Transmit data (channel 3, 2, 1, 0)
	RxD3, RxD2, RxD1, RxD0	Input	Receive data (channel 3, 2, 1, 0)
	SCK3, SCK2, SCK1, SCK0	I/O	Serial clock (channel 3, 2, 1, 0)
A/D converter	AN7 to AN0	Input	Analog input
	ADTRG	Input	A/D conversion external trigger input
D/A converter	DA1, DA0	Output	Analog output
A/D converter and D/A converters	AVCC	Input	This is the power supply pin for the A/D converter and D/A converter.
	AVSS	Input	This is the ground pin for the A/D converter and D/A converter.
	Vref	Input	This is the reference voltage input pin for the A/D converter and D/A converter.
I/O ports	P17 to P10	I/O	Port 1
	P36 to P30	I/O	Port 3
	P47 to P40	Input	Port 4
	P77 to P70	I/O	Port 7
	P97, P96	Input	Port 9
	PA3 to PA0	I/O	Port A
	PB7 to PB0	I/O	Port B
	PC7 to PC0	I/O	Port C
	PD7 to PD0	I/O	Port D
	PE7 to PE0	I/O	Port E
	PF7 to PF0	I/O	Port F
	PG4 to PG0	I/O	Port G

1.3 Internal Block Diagram

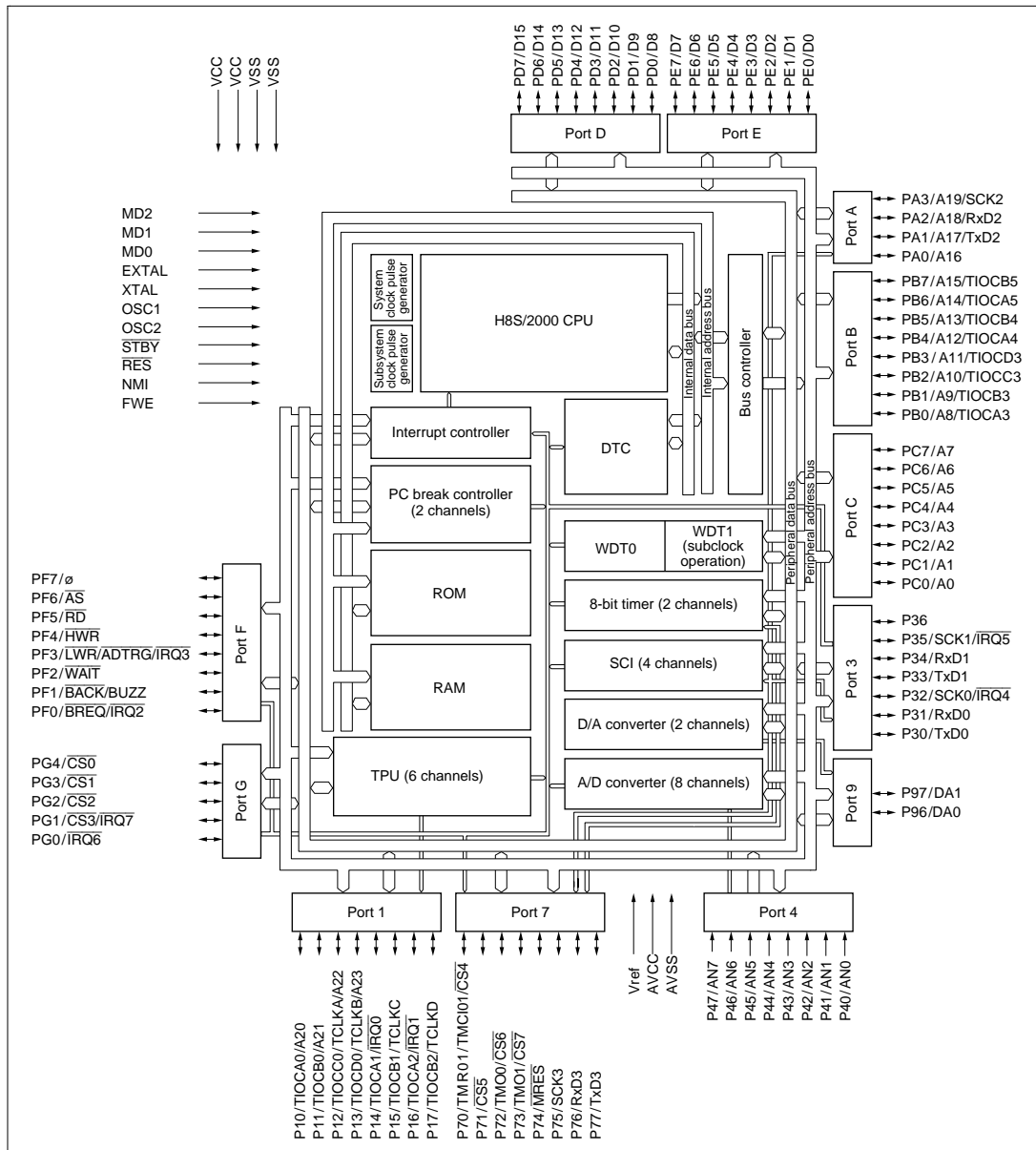


Figure 1.5 Internal Block Diagram of H8S/2237 Series

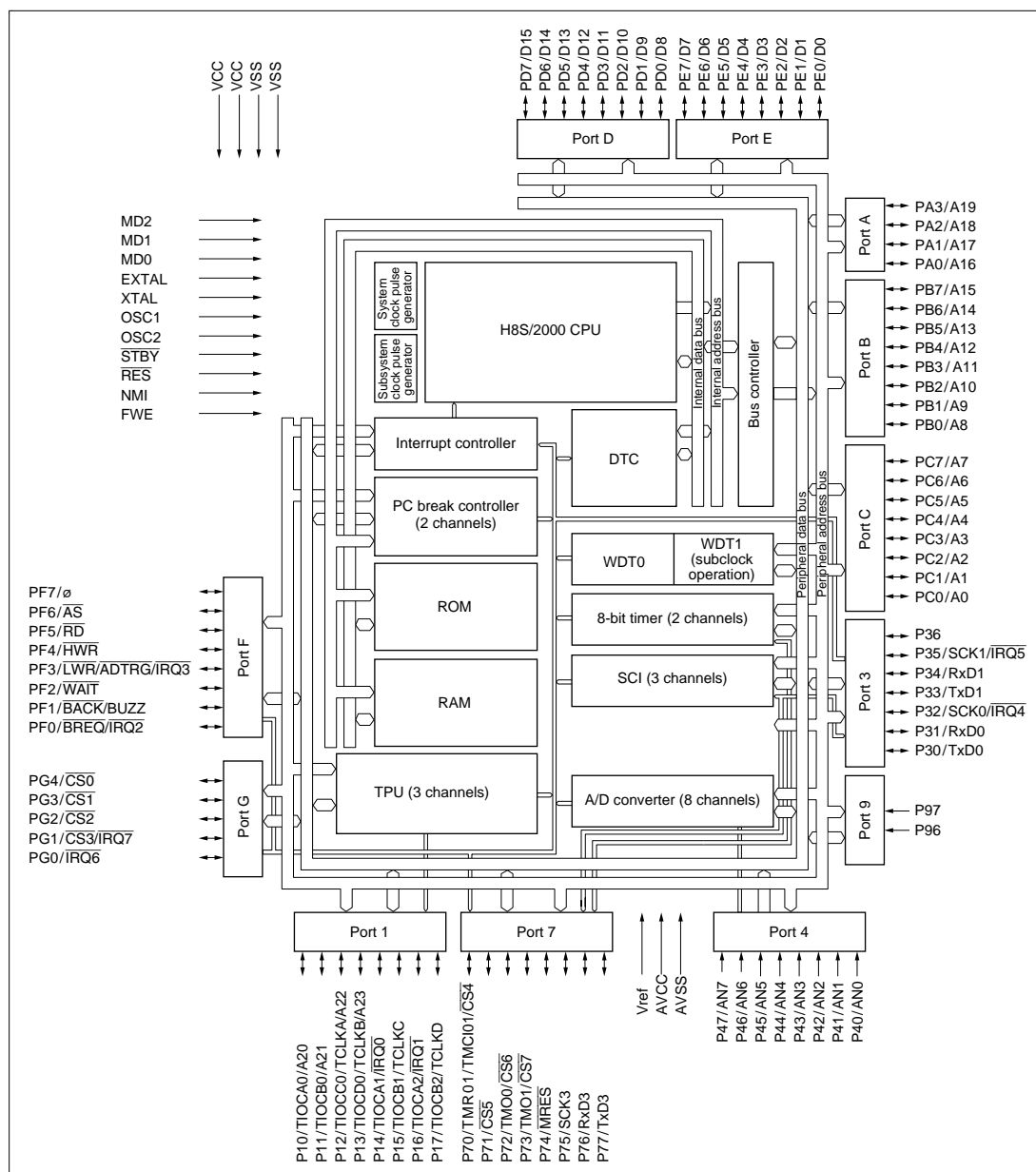


Figure 1.6 Internal Block Diagram of H8S/2227 Series

Section 2 CPU

2.1 Overview

The H8S/2000 CPU is a high-speed central processing unit with an internal 32-bit architecture that is upward-compatible with the H8/300 and H8/300H CPUs. The H8S/2000 CPU has sixteen 16-bit general registers, can address a 16-Mbyte (architecturally 4-Gbyte) linear address space, and is ideal for real-time control.

2.1.1 Features

- Upward-compatible with H8/300 and H8/300H CPUs
 - Can execute H8/300 and H8/300H object programs
- General-register architecture
 - Sixteen 16-bit general registers (also usable as sixteen 8-bit registers or eight 32-bit registers)
- Sixty-five basic instructions
 - 8/16/32-bit arithmetic and logic instructions
 - Multiply and divide instructions
 - Powerful bit-manipulation instructions
- Eight addressing modes
 - Register direct [Rn]
 - Register indirect [@ERn]
 - Register indirect with displacement [@(d:16,ERn) or @(d:32,ERn)]
 - Register indirect with post-increment or pre-decrement [@ERn+ or @-ERn]
 - Absolute address [@aa:8, @aa:16, @aa:24, or @aa:32]
 - Immediate [#xx:8, #xx:16, or #xx:32]
 - Program-counter relative [@(d:8,PC) or @(d:16,PC)]
 - Memory indirect [@@aa:8]
- 16-Mbyte address space
 - Program: 16 Mbytes
 - Data: 16 Mbytes (4 Gbytes architecturally)

- High-speed operation
 - All frequently-used instructions execute in one or two states
 - Maximum clock frequency: 10 MHz (ZTAT), 13 MHz (Mask ROM)
 - 8/16/32-bit register-register add/subtract: 100 ns (at 10 MHz operation)
 - 8 × 8-bit register-register multiply: 1200 ns (at 10 MHz operation)
 - 16 ÷ 8-bit register-register divide: 1200 ns (at 10 MHz operation)
 - 16 × 16-bit register-register multiply: 2000 ns (at 10 MHz operation)
 - 32 ÷ 16-bit register-register divide: 2000 ns (at 10 MHz operation)
- CPU operating mode
 - Advanced mode
- Power-down state
 - Transition to power-down state by SLEEP instruction
 - CPU clock speed selection

Differences between H8S/2600 CPU and H8S/2000 CPU: The differences between the H8S/2600 CPU and the H8S/2000 CPU are as shown below.

- Register configuration

The MAC register is supported only by the H8S/2600 CPU.
- Basic instructions

The four instructions MAC, CLRMAC, LDMAC, and STMAC are supported only by the H8S/2600 CPU.
- Number of execution states

The number of execution states of the MULXU and MULXS instructions.

Differences from H8/300 CPU: In comparison to the H8/300 CPU, the H8S/2000 CPU has the following enhancements.

- More general registers and control registers
 - Eight 16-bit expanded registers, and one 8-bit control register, have been added.
- Expanded address space
 - Normal mode supports the same 64-kbyte address space as the H8/300 CPU.
 - Advanced mode supports a maximum 16-Mbyte address space.
- Enhanced addressing
 - The addressing modes have been enhanced to make effective use of the 16-Mbyte address space.

- Enhanced instructions
 - Addressing modes of bit-manipulation instructions have been enhanced.
 - Signed multiply and divide instructions have been added.
 - Two-bit shift instructions have been added.
 - Instructions for saving and restoring multiple registers have been added.
 - A test and set instruction has been added.
- Higher speed
 - Basic instructions execute twice as fast.

Differences from H8/300H CPU: In comparison to the H8/300H CPU, the H8S/2000 CPU has the following enhancements.

- Additional control register
 - One 8-bit control register has been added.
- Enhanced instructions
 - Addressing modes of bit-manipulation instructions have been enhanced.
 - Two-bit shift instructions have been added.
 - Instructions for saving and restoring multiple registers have been added.
 - A test and set instruction has been added.
- Higher speed
 - Basic instructions execute twice as fast.

2.2 Register Configuration

The H8S/2000 CPU has general registers and control registers. The eight 32-bit general registers all have identical functions and can be used as either address registers or data registers. The control registers are the 24-bit program counter (PC), 8-bit extended register (EXR), and 8-bit condition code register (CCR).

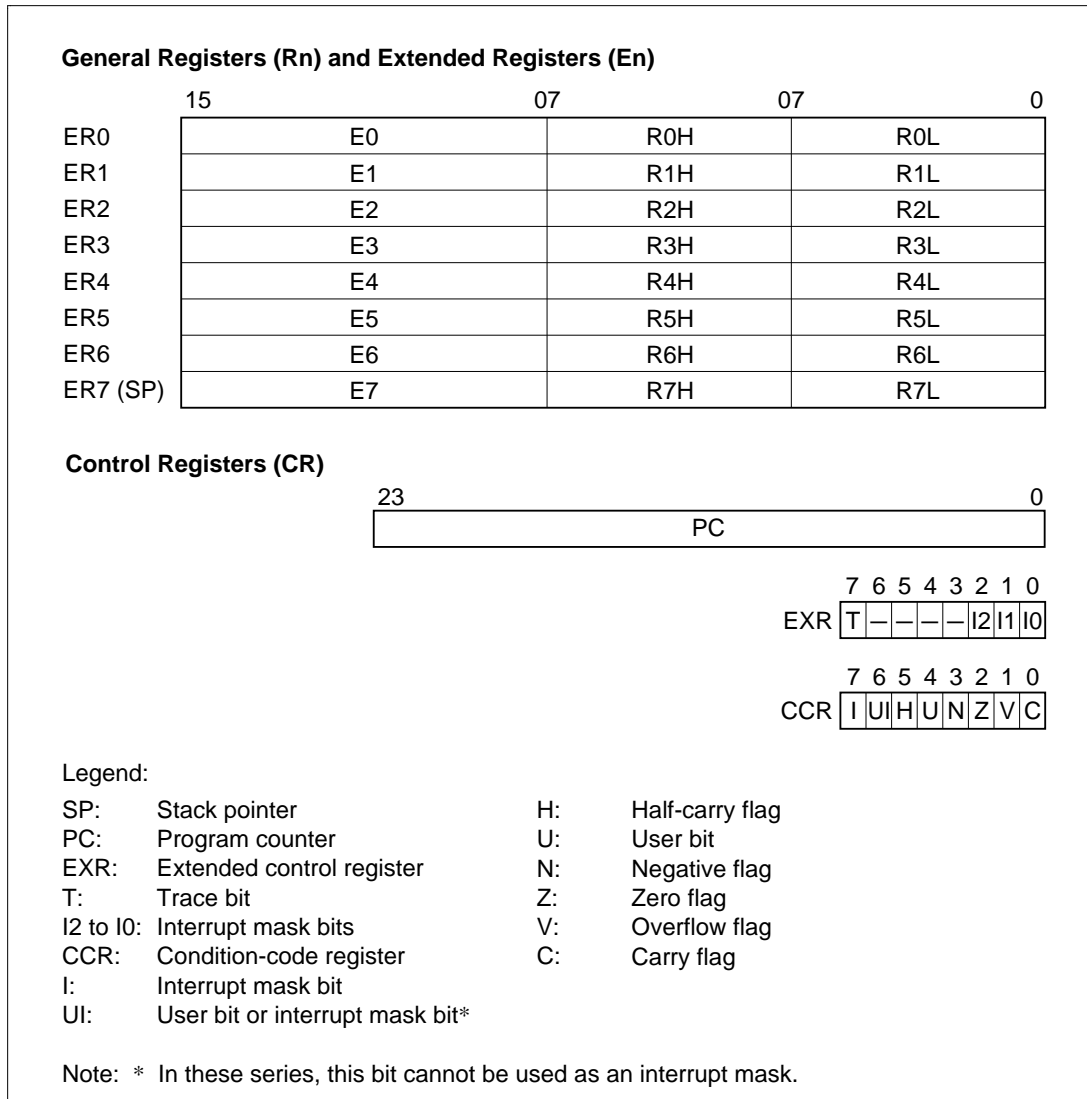


Figure 2.1 CPU Registers

General Registers: The CPU has eight 32-bit general registers. These 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 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 figure below illustrates the usage of the general registers. The usage of each register can be selected independently.

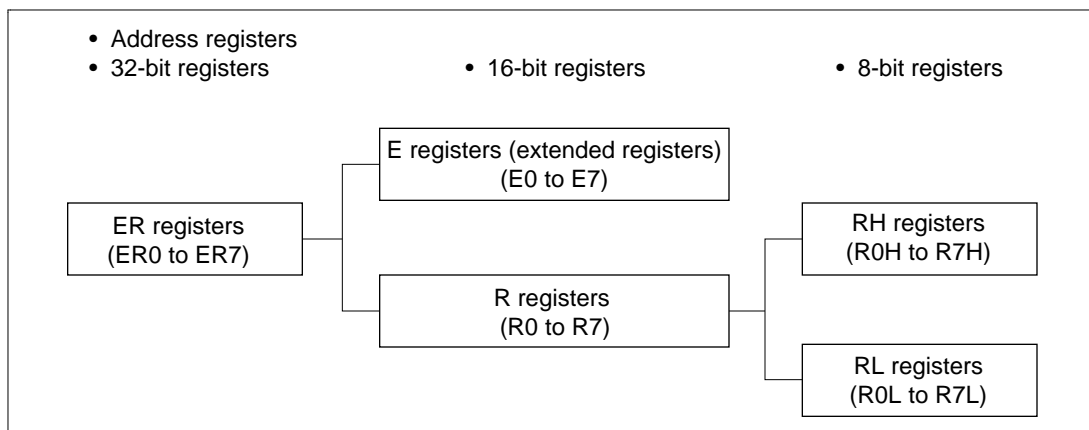


Figure 2.2 Usage of General Registers

Control Registers: The control registers are the 24-bit program counter (PC), 8-bit extended control register (EXR), and 8-bit condition-code register (CCR).

- **Program Counter (PC)**

This 24-bit counter indicates the address of the next instruction the CPU will execute. The length of all CPU instructions is 2 bytes (one word), so the least significant PC bit is ignored. (When an instruction is fetched, the least significant PC bit is regarded as 0.)

- **Extend Register (EXR)**

This 8-bit register comprises a trace bit (T) and interrupt mask bits (I2 to I0).

— Bit 7—Trace Bit (T)

Specifies whether or not trace mode is set. When this bit is cleared to 0, instructions are executed sequentially. When set to 1, trace exception handling is started each time an instruction is executed.

— Bits 6 to 3—Reserved

— Bits 2 to 0—Interrupt Mask Bits (I2 to I0)

These bits specify the interrupt request mask level (0 to 7). See section 2.9, Interrupts, for details.

EXR can be manipulated by the LDC, STC, ANDC, ORC, and XORC instructions. Except in the case of STC, interrupts (including NMI) are not accepted for 3 states after the instruction is executed.

- **Condition-Code Register (CCR)**

This 8-bit register contains internal CPU status information, including an interrupt mask bit (I) and half-carry (H), negative (N), zero (Z), overflow (V), and carry (C) flags.

- Bit 7—Interrupt Mask Bit (I): Masks interrupts other than NMI when set to 1. (NMI is accepted regardless of the I bit setting.) The I bit is set to 1 by hardware at the start of an exception-handling sequence. For details, refer to section 2.9, Interrupts.
- Bit 6—User Bit or Interrupt Mask Bit (UI): Can be written and read by software using the LDC, STC, ANDC, ORC, and XORC instructions. In these series, this bit cannot be used as an interrupt mask.
- Bit 5—Half-Carry Flag (H): When the ADD.B, ADDX.B, SUB.B, SUBX.B, CMP.B, or NEG.B instruction is executed, this flag is set to 1 if there is a carry or borrow at bit 3, and cleared to 0 otherwise. When the ADD.W, SUB.W, CMP.W, or NEG.W instruction is executed, the H flag is set to 1 if there is a carry or borrow at bit 11, and cleared to 0 otherwise. When the ADD.L, SUB.L, CMP.L, or NEG.L instruction is executed, the H flag is set to 1 if there is a carry or borrow at bit 27, and cleared to 0 otherwise.
- Bit 4—User Bit (U): Can be written and read by software using the LDC, STC, ANDC, ORC, and XORC instructions.
- Bit 3—Negative Flag (N): Stores the value of the most significant bit (sign bit) of data.
- Bit 2—Zero Flag (Z): Set to 1 to indicate zero data, and cleared to 0 to indicate non-zero data.
- Bit 1—Overflow Flag (V): Set to 1 when an arithmetic overflow occurs, and cleared to 0 at other times.
- Bit 0—Carry Flag (C): 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
 - Shift and rotate instructions, to store the carry

The carry flag is also used as a bit accumulator by bit manipulation instructions.

2.3 Data Formats

The 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.

Data Type	Register Number	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	

Figure 2.3 General Register Data Formats

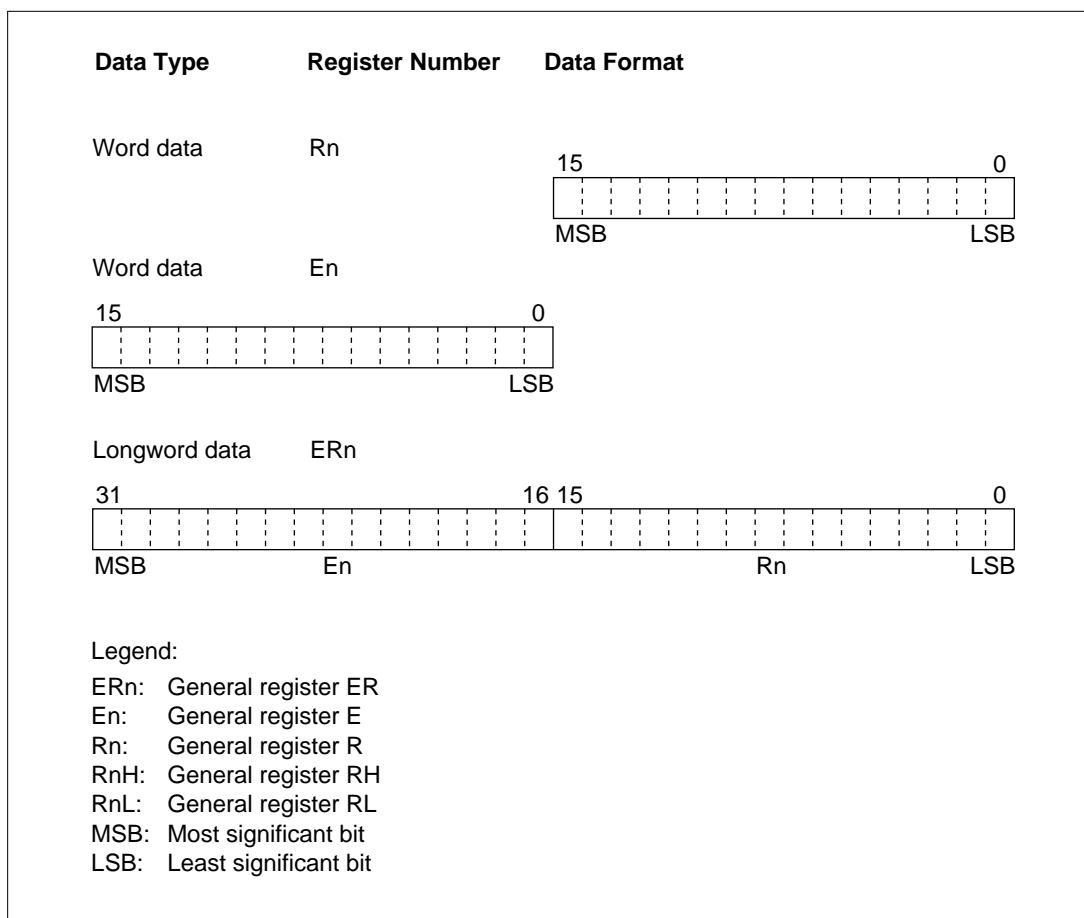


Figure 2.3 General Register Data Formats (cont)

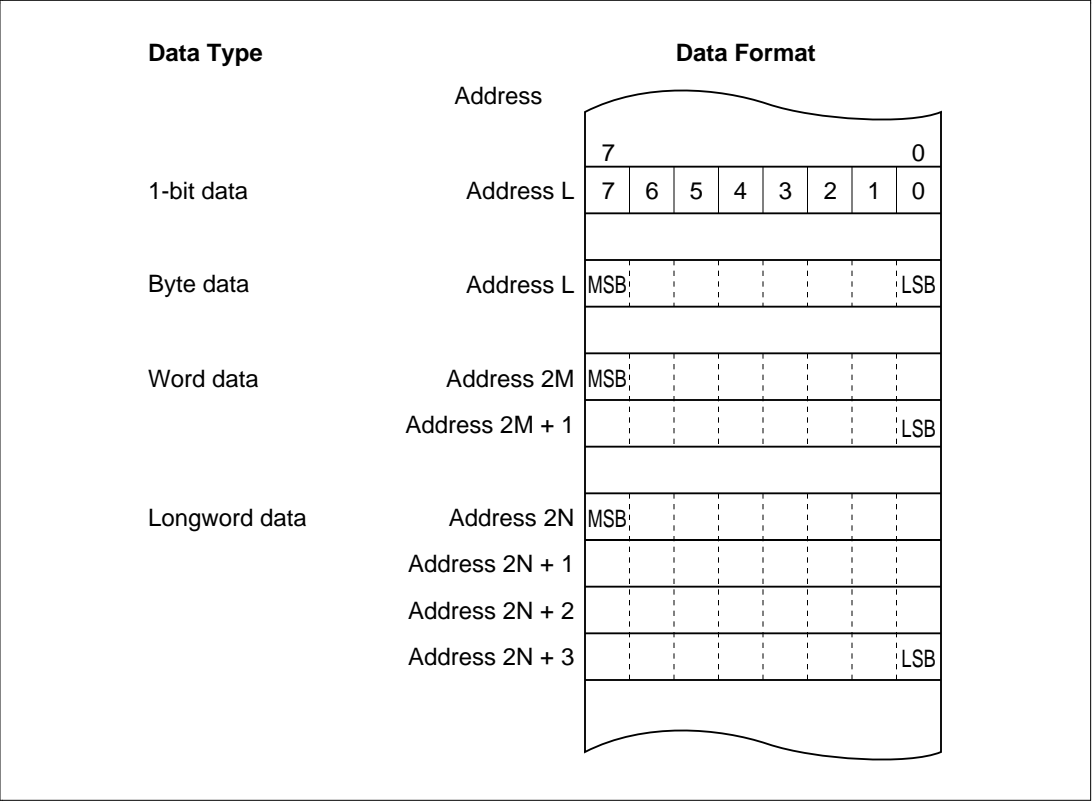


Figure 2.4 Memory Data Formats

2.4 Addressing Modes

The H8S/2000 CPU supports eight addressing modes.

Table 2.1 Addressing Modes

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

Effective Address Calculation: The upper 8 bits of the effective address are ignored, giving a 16-bit address.

No.	Addressing Mode and Instruction Format	Effective Address Calculation	Effective Address (EA)								
1	Register direct (Rn) <div><div>op</div><div>rm</div><div>rn</div></div>		Operand is general register contents.								
2	Register indirect (@ERn) <div><div>op</div><div>r</div></div>	<div><div>31</div><div>General register contents</div><div>0</div></div>	<div><div>31</div><div>Don't care</div><div>24 23</div><div>0</div></div>								
3	Register indirect with displacement @d:16, ERn) or @d:32, ERn) <div><div>op</div><div>r</div><div>disp</div></div>	<div><div>31</div><div>General register contents</div><div>0</div></div> <div><div>31</div><div>Sign extension</div><div>0</div><div>disp</div></div>	<div><div>31</div><div>Don't care</div><div>24 23</div><div>0</div></div>								
4	Register indirect with post-increment or pre-decrement • Register indirect with post-increment @ERn+ <div><div>op</div><div>r</div></div> • Register indirect with pre-decrement @-ERn <div><div>op</div><div>r</div></div>	<div><div>31</div><div>General register contents</div><div>0</div></div> <div><div>1, 2, or 4</div></div> <div><div>31</div><div>General register contents</div><div>0</div></div> <div><div>1, 2, or 4</div></div> <div><table><tr><th>Operand Size</th><th>Value added</th></tr><tr><td>Byte</td><td>1</td></tr><tr><td>Word</td><td>2</td></tr><tr><td>Longword</td><td>4</td></tr></table></div>	Operand Size	Value added	Byte	1	Word	2	Longword	4	<div><div>31</div><div>Don't care</div><div>24 23</div><div>0</div></div>
Operand Size	Value added										
Byte	1										
Word	2										
Longword	4										

No.	Addressing Mode and Instruction Format	Effective Address Calculation	Effective Address (EA)
5	Absolute address		
	@aa:8 <div> <div>op</div> <div>abs</div> </div>		<div> <div>31 24 23 8 7 0</div> <div>Don't care HFFFF</div> </div>
	@aa:16 <div> <div>op</div> <div>abs</div> </div>		<div> <div>31 24 23 16 15 0</div> <div>Don't care Sign extension</div> </div>
	@aa:24 <div> <div>op</div> <div>abs</div> </div>		<div> <div>31 24 23 0</div> <div>Don't care</div> </div>
6	@aa:32 <div> <div>op</div> <div>abs</div> </div>		<div> <div>31 24 23 0</div> <div>Don't care</div> </div>
	Immediate #xx:8/#xx:16/#xx:32 <div> <div>op</div> <div>IMM</div> </div>		Operand is immediate data.

No.	Addressing Mode and Instruction Format	Effective Address Calculation	Effective Address (EA)
7	<p>Program-counter relative @ (d:8, PC)/@ (d:16, PC)</p>		
8	<p>Memory indirect @aa:8</p> <ul style="list-style-type: none"> Advanced mode 		

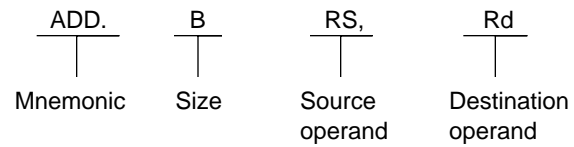
2.5 Instruction Set

The H8S/2000 CPU has 65 types of instructions.

2.5.1 Features

- Upward-compatible at object level with H8/300H and H8/300 CPUs
- General register architecture
- 8/16/32-bit transfer instructions and arithmetic and logic instructions
 - Byte (B), word (W), and longword (L) formats for transfer instructions and basic arithmetic and logic instructions
- Unsigned and signed multiply and divide instructions
- Powerful bit manipulation instructions
- Instructions for saving and restoring multiple registers

Assembler Format: The ADD instruction format is shown below as an example.



- Data transfer Instructions

Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code						Number of States*1
		#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@aa		I	H	N	Z	V	C	
																	Advanced
MOV	MOV.B #xx:8,Rd	B	2							#xx:8→Rd8	—	—	↑	↑	0	—	1
	MOV.B Rs,Rd	B		2						Rs8→Rd8	—	—	↑	↑	0	—	1
	MOV.B @ERs,Rd	B			2					@ERs→Rd8	—	—	↑	↑	0	—	2
	MOV.B @(d:16,ERs),Rd	B				4				@(d:16,ERs)→Rd8	—	—	↑	↑	0	—	3
	MOV.B @(d:32,ERs),Rd	B					8			@(d:32,ERs)→Rd8	—	—	↑	↑	0	—	5
	MOV.B @ERs+,Rd	B						2		@ERs→Rd8,ERs32+1→ERs32	—	—	↑	↑	0	—	3
	MOV.B @aa:8,Rd	B						2		@aa:8→Rd8	—	—	↑	↑	0	—	2
	MOV.B @aa:16,Rd	B						4		@aa:16→Rd8	—	—	↑	↑	0	—	3
	MOV.B @aa:32,Rd	B						6		@aa:32→Rd8	—	—	↑	↑	0	—	4
	MOV.B Rs,@ERd	B		2						Rs8→@ERd	—	—	↑	↑	0	—	2
	MOV.B Rs,@(d:16,ERd)	B			4					Rs8→@(d:16,ERd)	—	—	↑	↑	0	—	3
	MOV.B Rs,@(d:32,ERd)	B				8				Rs8→@(d:32,ERd)	—	—	↑	↑	0	—	5
	MOV.B Rs,@-ERd	B					2			ERd32-1→ERd32,Rs8→@ERd	—	—	↑	↑	0	—	3
	MOV.B Rs,@aa:8	B						2		Rs8→@aa:8	—	—	↑	↑	0	—	2
	MOV.B Rs,@aa:16	B						4		Rs8→@aa:16	—	—	↑	↑	0	—	3
	MOV.B Rs,@aa:32	B						6		Rs8→@aa:32	—	—	↑	↑	0	—	4
	MOV.W #xx:16,Rd	W	4							#xx:16→Rd16	—	—	↑	↑	0	—	2
	MOV.W Rs,Rd	W		2						Rs16→Rd16	—	—	↑	↑	0	—	1
	MOV.W @ERs,Rd	W			2					@ERs→Rd16	—	—	↑	↑	0	—	2
	MOV.W @(d:16,ERs),Rd	W				4				@(d:16,ERs)→Rd16	—	—	↑	↑	0	—	3
	MOV.W @(d:32,ERs),Rd	W					8			@(d:32,ERs)→Rd16	—	—	↑	↑	0	—	5
	MOV.W @ERs+,Rd	W						2		@ERs→Rd16,ERs32+2→ERs32	—	—	↑	↑	0	—	3
	MOV.W @aa:16,Rd	W						4		@aa:16→Rd16	—	—	↑	↑	0	—	3
	MOV.W @aa:32,Rd	W						6		@aa:32→Rd16	—	—	↑	↑	0	—	4
	MOV.W Rs,@ERd	W		2						Rs16→@ERd	—	—	↑	↑	0	—	2
	MOV.W Rs,@(d:16,ERd)	W			4					Rs16→@(d:16,ERd)	—	—	↑	↑	0	—	3
	MOV.W Rs,@(d:32,ERd)	W				8				Rs16→@(d:32,ERd)	—	—	↑	↑	0	—	5
	MOV.W Rs,@-ERd	W					2			ERd32-2→ERd32,Rs16→@ERd	—	—	↑	↑	0	—	3
	MOV.W Rs,@aa:16	W						4		Rs16→@aa:16	—	—	↑	↑	0	—	3
	MOV.W Rs,@aa:32	W						6		Rs16→@aa:32	—	—	↑	↑	0	—	4
	MOV.L #xx:32,ERd	L	6							#xx:32→ERd32	—	—	↑	↑	0	—	3
	MOV.L ERs,ERd	L		2						ERs32→ERd32	—	—	↑	↑	0	—	1
	MOV.L @ERs,ERd	L			4					@ERs→ERd32	—	—	↑	↑	0	—	4
	MOV.L @(d:16,ERs),ERd	L				6				@(d:16,ERs)→ERd32	—	—	↑	↑	0	—	5
	MOV.L @(d:32,ERs),ERd	L					10			@(d:32,ERs)→ERd32	—	—	↑	↑	0	—	7
	MOV.L @ERs+,ERd	L						4		@ERs→ERd32,ERs32+4→ERs32	—	—	↑	↑	0	—	5
	MOV.L @aa:16,ERd	L						6		@aa:16→ERd32	—	—	↑	↑	0	—	5
	MOV.L @aa:32,ERd	L						8		@aa:32→ERd32	—	—	↑	↑	0	—	6
	MOV.L ERs,@ERd	L			4					ERs32→@ERd	—	—	↑	↑	0	—	4
	MOV.L ERs,@(d:16,ERd)	L				6				ERs32→@(d:16,ERd)	—	—	↑	↑	0	—	5
	MOV.L ERs,@(d:32,ERd)	L					10			ERs32→@(d:32,ERd)	—	—	↑	↑	0	—	7
	MOV.L ERs,@-ERd	L						4		ERd32-4→ERd32,ERs32→@ERd	—	—	↑	↑	0	—	5
	MOV.L ERs,@aa:16	L						6		ERs32→@aa:16	—	—	↑	↑	0	—	5
	MOV.L ERs,@aa:32	L						8		ERs32→@aa:32	—	—	↑	↑	0	—	6
POP	POP.W Rn	W							2	@SP→Rn16,SP+2→SP	—	—	↑	↑	0	—	3
	POP.L ERn	L							4	@SP→ERn32,SP+4→SP	—	—	↑	↑	0	—	5
PUSH	PUSH.W Rn	W							2	SP-2→SP,Rn16→@SP	—	—	↑	↑	0	—	3
	PUSH.L ERn	L							4	SP-4→SP,ERn32→@SP	—	—	↑	↑	0	—	5
LDM	LDM @SP+,(ERm-ERn)	L							4	(@SP→ERn32,SP+4→SP) Repeated for each register restored	—	—	—	—	—	—	7/9/11 [1]
STM	STM (ERm-ERn),@-SP	L							4	(SP-4→SP,ERn32→@SP) Repeated for each register saved	—	—	—	—	—	—	7/9/11 [1]
MOVFP	MOVFP @aa:16,Rd	Cannot be used with these series.															[2]
MOVTP	MOVTP Rs,@aa:16																[2]

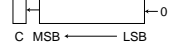

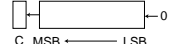
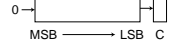
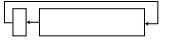

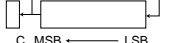
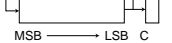
- Arithmetic instructions

Mnemonic	Addressing Mode/Instruction Length (Bytes)	Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code					Number of States ^{*1}
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@aa		I	H	N	Z	V	
																	Advanced
ADD	ADD.B #xx:8,Rd	B	2								Rd8+#xx:8→Rd8	N	↑	↑	↑	↑	1
	ADD.B Rs,Rd	B	2								Rd8+Rs8→Rd8	N	↑	↑	↑	↑	1
	ADD.W #xx:16,Rd	W	4								Rd16+#xx:16→Rd16	N	[3]	↑	↑	↑	2
	ADD.W Rs,Rd	W	2								Rd16+Rs16→Rd16	N	[3]	↑	↑	↑	1
	ADD.L #xx:32,ERd	L	6								ERd32+#xx:32→ERd32	N	[4]	↑	↑	↑	3
ADDDX	ADD.L ERs,ERd	L	2								ERd32+ERs32→ERd32	N	[4]	↑	↑	↑	1
	ADDDX #xx:8,Rd	B	2								Rd8+#xx:8+C→Rd8	N	↑	↑	[5]	↑	1
ADDS	ADDDX Rs,Rd	B	2								Rd8+Rs8+C→Rd8	N	↑	↑	[5]	↑	1
	ADDS #1,ERd	L	2								ERd32+1→ERd32	N	N	N	N	N	1
INC	ADDS #2,ERd	L	2								ERd32+2→ERd32	N	N	N	N	N	1
	ADDS #4,ERd	L	2								ERd32+4→ERd32	N	N	N	N	N	1
	INC.B Rd	B	2								Rd8+1→Rd8	N	N	↑	↑	N	1
	INC.W #1,Rd	W	2								Rd16+1→Rd16	N	N	↑	↑	N	1
	INC.W #2,Rd	W	2								Rd16+2→Rd16	N	N	↑	↑	N	1
DAA	INC.L #1,ERd	L	2								ERd32+1→ERd32	N	N	↑	↑	N	1
	INC.L #2,ERd	L	2								ERd32+2→ERd32	N	N	↑	↑	N	1
SUB	DAA Rd	B	2								Rd8 decimal adjust → Rd8	N	*	↑	↑	*	1
	SUB.B Rs,Rd	B	2								Rd8-Rs8→Rd8	N	↑	↑	↑	↑	1
	SUB.W #xx:16,Rd	W	4								Rd16-Rs16→Rd16	N	[3]	↑	↑	↑	2
	SUB.W Rs,Rd	W	2								Rd16-Rs16→Rd16	N	[3]	↑	↑	↑	1
	SUB.L #xx:32,ERd	L	6								ERd32-#xx:32→ERd32	N	[4]	↑	↑	↑	3
SUBX	SUB.L ERs,ERd	L	2								ERd32-ERs32→ERd32	N	[4]	↑	↑	↑	1
	SUBX #xx:8,Rd	B	2								Rd8-#xx:8-C→Rd8	N	↑	↑	[5]	↑	1
SUBS	SUBX Rs,Rd	B	2								Rd8-Rs8-C→Rd8	N	↑	↑	[5]	↑	1
	SUBS #1,ERd	L	2								ERd32-1→ERd32	N	N	N	N	N	1
	SUBS #2,ERd	L	2								ERd32-2→ERd32	N	N	N	N	N	1
	SUBS #4,ERd	L	2								ERd32-4→ERd32	N	N	N	N	N	1
DEC	DEC.B Rd	B	2								Rd8-1→Rd8	N	N	↑	↑	N	1
	DEC.W #1,Rd	W	2								Rd16-1→Rd16	N	N	↑	↑	N	1
	DEC.W #2,Rd	W	2								Rd16-2→Rd16	N	N	↑	↑	N	1
	DEC.L #1,ERd	L	2								ERd32-1→ERd32	N	N	↑	↑	N	1
	DEC.L #2,ERd	L	2								ERd32-2→ERd32	N	N	↑	↑	N	1
DAS	DAS Rd	B	2								Rd8 decimal adjust → Rd8	N	*	↑	↑	*	1
MULXU	MULXU.B Rs,Rd	B	2								Rd8×Rs8→Rd16 (unsigned multiplication)	N	N	N	N	N	12
	MULXU.W Rs,ERd	W	2								Rd16×Rs16→ERd32 (unsigned multiplication)	N	N	N	N	N	20
MULXS	MULXS.B Rs,Rd	B	4								Rd8×Rs8→Rd16 (signed multiplication)	N	N	↑	↑	N	13
	MULXS.W Rs,ERd	W	4								Rd16×Rs16→ERd32 (signed multiplication)	N	N	↑	↑	N	21
DIVXU	DIVXU.B Rs,Rd	B	2								Rd16@Rs8→Rd16 (RdH: remainder, RdL: quotient) (unsigned division)	N	N	[6]	[7]	N	12
	DIVXU.W Rs,ERd	W	2								ERd32@Rs16→ERd32 (Ed: remainder, Rd: quotient) (unsigned division)	N	N	[6]	[7]	N	20
DIVXS	DIVXS.B Rs,Rd	B	4								Rd16@Rs8→Rd16 (RdH: remainder, RdL: quotient) (signed division)	N	N	[8]	[7]	N	13
	DIVXS.W Rs,ERd	W	4								ERd32@Rs16→ERd32 (Ed: remainder, Rd: quotient) (signed division)	N	N	[8]	[7]	N	21
CMP	CMP.B #xx:8,Rd	B	2								Rd8-#xx:8	N	↑	↑	↑	↑	1
	CMP.B Rs,Rd	B	2								Rd8-Rs8	N	↑	↑	↑	↑	1
	CMP.W #xx:16,Rd	W	4								Rd16-#xx:16	N	[3]	↑	↑	↑	2
	CMP.W Rs,Rd	W	2								Rd16-Rs16	N	[3]	↑	↑	↑	1
	CMP.L #xx:32,ERd	L	6								ERd32-#xx:32	N	[4]	↑	↑	↑	3
	CMP.L ERs,ERd	L	2								ERd32-ERs32	N	[4]	↑	↑	↑	1
NEG	NEG.B Rd	B	2								0-Rd8→Rd8	N	↑	↑	↑	↑	1
	NEG.W Rd	W	2								0-Rd16→Rd16	N	↑	↑	↑	↑	1
	NEG.L ERd	L	2								0-ERd32→ERd32	N	↑	↑	↑	↑	1
EXTU	EXTU.W Rd	W	2								0 → (<bits 15 to 8> of Rd16)	N	N	0	↑	0	1
	EXTU.L ERd	L	2								0 → (<bits 31 to 16> of ERd32)	N	N	0	↑	0	1
EXTS	EXTS.W Rd	W	2								(<bit 7> of Rd16) → (<bits 15 to 8> of Rd16)	N	N	↑	↑	0	1
	EXTS.L ERd	L	2								(<bit 15> of ERd32) → (<bits 31 to 16> of ERd32)	N	N	↑	↑	0	1
TAS	TAS @ERd	B			4						@ERd-0 → CRR set, (1) → (<bit 7> of @ERd)	N	N	↑	↑	0	4

- Logical instructions

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code						Number of States ^{*1}	
			#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn++	@aa	@(d,PC)	@@aa			I	H	N	Z	V		C
			Advanced																
AND	AND.B #xx:8,Rd	B	2									Rd8^#xx:8→Rd8	—	—	↑	↑	0	—	1
	AND.B Rs,Rd	B	2	2								Rd8^Rs8→Rd8	—	—	↑	↑	0	—	1
	AND.W #xx:16,Rd	W	4									Rd16^#xx:16→Rd16	—	—	↑	↑	0	—	2
	AND.W Rs,Rd	W	2	2								Rd16^Rs16→Rd16	—	—	↑	↑	0	—	1
	AND.L #xx:32,ERd	L	6									ERd32^#xx:32→ERd32	—	—	↑	↑	0	—	3
	AND.L ERs,ERd	L	4		4							ERd32^ERs32→ERd32	—	—	↑	↑	0	—	2
OR	OR.B #xx:8,Rd	B	2									Rd8v#xx:8→Rd8	—	—	↑	↑	0	—	1
	OR.B Rs,Rd	B	2	2								Rd8vRs8→Rd8	—	—	↑	↑	0	—	1
	OR.W #xx:16,Rd	W	4									Rd16v#xx:16→Rd16	—	—	↑	↑	0	—	2
	OR.W Rs,Rd	W	2	2								Rd16vRs16→Rd16	—	—	↑	↑	0	—	1
	OR.L #xx:32,ERd	L	6									ERd32v#xx:32→ERd32	—	—	↑	↑	0	—	3
	OR.L ERs,ERd	L	4		4							ERd32vERs32→ERd32	—	—	↑	↑	0	—	2
XOR	XOR.B #xx:8,Rd	B	2									Rd8@#xx:8→Rd8	—	—	↑	↑	0	—	1
	XOR.B Rs,Rd	B	2	2								Rd8@Rs8→Rd8	—	—	↑	↑	0	—	1
	XOR.W #xx:16,Rd	W	4									Rd16@#xx:16→Rd16	—	—	↑	↑	0	—	2
	XOR.W Rs,Rd	W	2	2								Rd16@Rs16→Rd16	—	—	↑	↑	0	—	1
	XOR.L #xx:32,ERd	L	6									ERd32@#xx:32→ERd32	—	—	↑	↑	0	—	3
	XOR.L ERs,ERd	L	4		4							ERd32@ERs32→ERd32	—	—	↑	↑	0	—	2
NOT	NOT.B Rd	B	2		2							~Rd8→Rd8	—	—	↑	↑	0	—	1
	NOT.W Rd	W	2		2							~Rd16→Rd16	—	—	↑	↑	0	—	1
	NOT.L ERd	L	2		2							~ERd32→ERd32	—	—	↑	↑	0	—	1

- Shift instructions

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code					Number of States ^{*1}	
			#xx	Rn	@ ERn	@ (d,ERn)	@ -ERn/@ERn+	@ aa	@ (d,PC)	@ @aa								
													I	H	N	Z	V	C
SHAL	SHAL.B Rd	B	2									—	—	↑	↑	↑	↑	1
	SHAL.B #2,Rd	B	2									—	—	↑	↑	↑	↑	1
	SHAL.W Rd	W	2									—	—	↑	↑	↑	↑	1
	SHAL.W #2,Rd	W	2									—	—	↑	↑	↑	↑	1
	SHALL.L ERd	L	2									—	—	↑	↑	↑	↑	1
	SHALL.L #2,ERd	L	2									—	—	↑	↑	↑	↑	1
SHAR	SHAR.B Rd	B	2									—	—	↑	↑	0	↑	1
	SHAR.B #2,Rd	B	2									—	—	↑	↑	0	↑	1
	SHAR.W Rd	W	2									—	—	↑	↑	0	↑	1
	SHAR.W #2,Rd	W	2									—	—	↑	↑	0	↑	1
	SHAR.L ERd	L	2									—	—	↑	↑	0	↑	1
	SHAR.L #2,ERd	L	2									—	—	↑	↑	0	↑	1
SHLL	SHLL.B Rd	B	2									—	—	↑	↑	0	↑	1
	SHLL.B #2,Rd	B	2									—	—	↑	↑	0	↑	1
	SHLL.W Rd	W	2									—	—	↑	↑	0	↑	1
	SHLL.W #2,Rd	W	2									—	—	↑	↑	0	↑	1
	SHLLL.L ERd	L	2									—	—	↑	↑	0	↑	1
	SHLLL.L #2,ERd	L	2									—	—	↑	↑	0	↑	1
SHLR	SHLR.B Rd	B	2									—	0	↑	↑	0	↑	1
	SHLR.B #2,Rd	B	2									—	0	↑	↑	0	↑	1
	SHLR.W Rd	W	2									—	0	↑	↑	0	↑	1
	SHLR.W #2,Rd	W	2									—	0	↑	↑	0	↑	1
	SHLR.L ERd	L	2									—	0	↑	↑	0	↑	1
	SHLR.L #2,ERd	L	2									—	0	↑	↑	0	↑	1
ROTXL	ROTXL.B Rd	B	2									—	—	↑	↑	0	↑	1
	ROTXL.B #2,Rd	B	2									—	—	↑	↑	0	↑	1
	ROTXL.W Rd	W	2									—	—	↑	↑	0	↑	1
	ROTXL.W #2,Rd	W	2									—	—	↑	↑	0	↑	1
	ROTXL.L ERd	L	2									—	—	↑	↑	0	↑	1
	ROTXL.L #2,ERd	L	2									—	—	↑	↑	0	↑	1
ROTXR	ROTXR.B Rd	B	2									—	—	↑	↑	0	↑	1
	ROTXR.B #2,Rd	B	2									—	—	↑	↑	0	↑	1
	ROTXR.W Rd	W	2									—	—	↑	↑	0	↑	1
	ROTXR.W #2,Rd	W	2									—	—	↑	↑	0	↑	1
	ROTXR.L ERd	L	2									—	—	↑	↑	0	↑	1
	ROTXR.L #2,ERd	L	2									—	—	↑	↑	0	↑	1
ROTL	ROTL.B Rd	B	2									—	—	↑	↑	0	↑	1
	ROTL.B #2,Rd	B	2									—	—	↑	↑	0	↑	1
	ROTL.W Rd	W	2									—	—	↑	↑	0	↑	1
	ROTL.W #2,Rd	W	2									—	—	↑	↑	0	↑	1
	ROTL.L ERd	L	2									—	—	↑	↑	0	↑	1
	ROTL.L #2,ERd	L	2									—	—	↑	↑	0	↑	1
ROTR	ROTR.B Rd	B	2									—	—	↑	↑	0	↑	1
	ROTR.B #2,Rd	B	2									—	—	↑	↑	0	↑	1
	ROTR.W Rd	W	2									—	—	↑	↑	0	↑	1
	ROTR.W #2,Rd	W	2									—	—	↑	↑	0	↑	1
	ROTR.L ERd	L	2									—	—	↑	↑	0	↑	1
	ROTR.L #2,ERd	L	2									—	—	↑	↑	0	↑	1

- Bit manipulation instructions

Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code						Number of States ^{*1}
		#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@aa		I	H	N	Z	V	C	
																	Advanced
BSET	BSET #xx:3,Rd	B	2							(#xx:3 of Rd8)←1	—	—	—	—	—	1	
	BSET #xx:3,@ERd	B		4						(#xx:3 of @ERd)←1	—	—	—	—	—	4	
	BSET #xx:3,@aa:8	B					4			(#xx:3 of @aa:8)←1	—	—	—	—	—	4	
	BSET #xx:3,@aa:16	B					6			(#xx:3 of @aa:16)←1	—	—	—	—	—	5	
	BSET #xx:3,@aa:32	B					8			(#xx:3 of @aa:32)←1	—	—	—	—	—	6	
	BSET Rn,Rd	B	2							(Rn8 of Rd8)←1	—	—	—	—	—	1	
	BSET Rn,@ERd	B		4						(Rn8 of @ERd)←1	—	—	—	—	—	4	
	BSET Rn,@aa:8	B					4			(Rn8 of @aa:8)←1	—	—	—	—	—	4	
	BSET Rn,@aa:16	B					6			(Rn8 of @aa:16)←1	—	—	—	—	—	5	
BCLR	BCLR #xx:3,Rd	B	2							(#xx:3 of Rd8)←0	—	—	—	—	—	1	
	BCLR #xx:3,@ERd	B		4						(#xx:3 of @ERd)←0	—	—	—	—	—	4	
	BCLR #xx:3,@aa:8	B					4			(#xx:3 of @aa:8)←0	—	—	—	—	—	4	
	BCLR #xx:3,@aa:16	B					6			(#xx:3 of @aa:16)←0	—	—	—	—	—	5	
	BCLR #xx:3,@aa:32	B					8			(#xx:3 of @aa:32)←0	—	—	—	—	—	6	
	BCLR Rn,Rd	B	2							(Rn8 of Rd8)←0	—	—	—	—	—	1	
	BCLR Rn,@ERd	B		4						(Rn8 of @ERd)←0	—	—	—	—	—	4	
	BCLR Rn,@aa:8	B					4			(Rn8 of @aa:8)←0	—	—	—	—	—	4	
	BCLR Rn,@aa:16	B					6			(Rn8 of @aa:16)←0	—	—	—	—	—	5	
BNOT	BCLR Rn,@aa:32	B					8			(Rn8 of @aa:32)←0	—	—	—	—	—	6	
	BCLR Rn,Rd	B	2							(Rn8 of Rd8)←¬(Rn8 of Rd8)	—	—	—	—	—	1	
	BNOT #xx:3,Rd	B	2							(#xx:3 of Rd8)←¬(#xx:3 of Rd8)	—	—	—	—	—	4	
	BNOT #xx:3,@ERd	B		4						(#xx:3 of @ERd)←¬(#xx:3 of @ERd)	—	—	—	—	—	1	
	BNOT #xx:3,@aa:8	B					4			(#xx:3 of @aa:8)←¬(#xx:3 of @aa:8)	—	—	—	—	—	4	
	BNOT #xx:3,@aa:16	B					6			(#xx:3 of @aa:16)←¬(#xx:3 of @aa:16)	—	—	—	—	—	5	
	BNOT #xx:3,@aa:32	B					8			(#xx:3 of @aa:32)←¬(#xx:3 of @aa:32)	—	—	—	—	—	6	
	BNOT Rn,Rd	B	2							(Rn8 of Rd8)←¬(Rn8 of Rd8)	—	—	—	—	—	1	
	BNOT Rn,@ERd	B		4						(Rn8 of @ERd)←¬(Rn8 of @ERd)	—	—	—	—	—	4	
BTST	BNOT Rn,@aa:8	B					4			(Rn8 of @aa:8)←¬(Rn8 of @aa:8)	—	—	—	—	—	4	
	BNOT Rn,@aa:16	B					6			(Rn8 of @aa:16)←¬(Rn8 of @aa:16)	—	—	—	—	—	5	
	BNOT Rn,@aa:32	B					8			(Rn8 of @aa:32)←¬(Rn8 of @aa:32)	—	—	—	—	—	6	
	BTST #xx:3,Rd	B	2							¬(#xx:3 of Rd8)→Z	—	—	—	↑	—	1	
	BTST #xx:3,@ERd	B		4						¬(#xx:3 of @ERd)→Z	—	—	—	↑	—	3	
	BTST #xx:3,@aa:8	B					4			¬(#xx:3 of @aa:8)→Z	—	—	—	↑	—	3	
	BTST #xx:3,@aa:16	B					6			¬(#xx:3 of @aa:16)→Z	—	—	—	↑	—	4	
	BTST #xx:3,@aa:32	B					8			¬(#xx:3 of @aa:32)→Z	—	—	—	↑	—	5	
	BTST Rn,Rd	B	2							¬(Rn8 of Rd8)→Z	—	—	—	↑	—	1	
BLD	BTST Rn,@ERd	B		4						¬(Rn8 of @ERd)→Z	—	—	—	↑	—	3	
	BTST Rn,@aa:8	B					4			¬(Rn8 of @aa:8)→Z	—	—	—	↑	—	3	
	BTST Rn,@aa:16	B					6			¬(Rn8 of @aa:16)→Z	—	—	—	↑	—	4	
	BTST Rn,@aa:32	B					8			¬(Rn8 of @aa:32)→Z	—	—	—	↑	—	5	
	BLD #xx:3,Rd	B	2							(#xx:3 of Rd8)→C	—	—	—	—	↑	1	
	BLD #xx:3,@ERd	B		4						(#xx:3 of @ERd)→C	—	—	—	—	↑	3	
	BLD #xx:3,@aa:8	B					4			(#xx:3 of @aa:8)→C	—	—	—	—	↑	3	
	BLD #xx:3,@aa:16	B					6			(#xx:3 of @aa:16)→C	—	—	—	—	↑	4	
	BLD #xx:3,@aa:32	B					8			(#xx:3 of @aa:32)→C	—	—	—	—	↑	5	
BILD	BILD #xx:3,Rd	B	2							¬(#xx:3 of Rd8)→C	—	—	—	—	↑	1	
	BILD #xx:3,@ERd	B		4						¬(#xx:3 of @ERd)→C	—	—	—	—	↑	3	
	BILD #xx:3,@aa:8	B					4			¬(#xx:3 of @aa:8)→C	—	—	—	—	↑	3	
	BILD #xx:3,@aa:16	B					6			¬(#xx:3 of @aa:16)→C	—	—	—	—	↑	4	
	BILD #xx:3,@aa:32	B					8			¬(#xx:3 of @aa:32)→C	—	—	—	—	↑	5	
BST	BST #xx:3,Rd	B	2							C→(#xx:3 of Rd8)	—	—	—	—	—	1	
	BST #xx:3,@ERd	B		4						C→(#xx:3 of @ERd)	—	—	—	—	—	4	
	BST #xx:3,@aa:8	B					4			C→(#xx:3 of @aa:8)	—	—	—	—	—	4	
	BST #xx:3,@aa:16	B					6			C→(#xx:3 of @aa:16)	—	—	—	—	—	5	
	BST #xx:3,@aa:32	B					8			C→(#xx:3 of @aa:32)	—	—	—	—	—	6	
BIST	BIST #xx:3,Rd	B	2							C→(#xx:3 of Rd8)	—	—	—	—	—	1	
	BIST #xx:3,@ERd	B		4						C→(#xx:3 of @ERd)	—	—	—	—	—	4	
	BIST #xx:3,@aa:8	B					4			C→(#xx:3 of @aa:8)	—	—	—	—	—	4	
	BIST #xx:3,@aa:16	B					6			C→(#xx:3 of @aa:16)	—	—	—	—	—	5	
	BIST #xx:3,@aa:32	B					8			C→(#xx:3 of @aa:32)	—	—	—	—	—	6	
BAND	BAND #xx:3,Rd	B	2							C∧(#xx:3 of Rd8)→C	—	—	—	—	↑	1	
	BAND #xx:3,@ERd	B		4						C∧(#xx:3 of @ERd)→C	—	—	—	—	↑	3	
	BAND #xx:3,@aa:8	B					4			C∧(#xx:3 of @aa:8)→C	—	—	—	—	↑	3	
	BAND #xx:3,@aa:16	B					6			C∧(#xx:3 of @aa:16)→C	—	—	—	—	↑	4	
	BAND #xx:3,@aa:32	B					8			C∧(#xx:3 of @aa:32)→C	—	—	—	—	↑	5	

- Bit manipulation instructions (cont)

Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code					Number of States*1
		#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@aa		I	H	N	Z	V	
																Advanced
BIAND	BIAND #xx:3,Rd	B	2							$C \wedge \neg(\#xx:3 \text{ of } Rd8) \rightarrow C$	—	—	—	—	↑	1
	BIAND #xx:3,@ERd	B		4						$C \wedge \neg(\#xx:3 \text{ of } @ERd) \rightarrow C$	—	—	—	—	↑	3
	BIAND #xx:3,@aa:8	B					4			$C \wedge \neg(\#xx:3 \text{ of } @aa:8) \rightarrow C$	—	—	—	—	↑	3
	BIAND #xx:3,@aa:16	B					6			$C \wedge \neg(\#xx:3 \text{ of } @aa:16) \rightarrow C$	—	—	—	—	↑	4
	BIAND #xx:3,@aa:32	B					8			$C \wedge \neg(\#xx:3 \text{ of } @aa:32) \rightarrow C$	—	—	—	—	↑	5
BOR	BOR #xx:3,Rd	B	2							$C \vee (\#xx:3 \text{ of } Rd8) \rightarrow C$	—	—	—	—	↑	1
	BOR #xx:3,@ERd	B		4						$C \vee (\#xx:3 \text{ of } @ERd) \rightarrow C$	—	—	—	—	↑	3
	BOR #xx:3,@aa:8	B					4			$C \vee (\#xx:3 \text{ of } @aa:8) \rightarrow C$	—	—	—	—	↑	3
	BOR #xx:3,@aa:16	B					6			$C \vee (\#xx:3 \text{ of } @aa:16) \rightarrow C$	—	—	—	—	↑	4
	BOR #xx:3,@aa:32	B					8			$C \vee (\#xx:3 \text{ of } @aa:32) \rightarrow C$	—	—	—	—	↑	5
BIOR	BIOR #xx:3,Rd	B	2							$C \vee \neg(\#xx:3 \text{ of } Rd8) \rightarrow C$	—	—	—	—	↑	1
	BIOR #xx:3,@ERd	B		4						$C \vee \neg(\#xx:3 \text{ of } @ERd) \rightarrow C$	—	—	—	—	↑	3
	BIOR #xx:3,@aa:8	B					4			$C \vee \neg(\#xx:3 \text{ of } @aa:8) \rightarrow C$	—	—	—	—	↑	3
	BIOR #xx:3,@aa:16	B					6			$C \vee \neg(\#xx:3 \text{ of } @aa:16) \rightarrow C$	—	—	—	—	↑	4
	BIOR #xx:3,@aa:32	B					8			$C \vee \neg(\#xx:3 \text{ of } @aa:32) \rightarrow C$	—	—	—	—	↑	5
BXOR	BXOR #xx:3,Rd	B	2							$C \oplus (\#xx:3 \text{ of } Rd8) \rightarrow C$	—	—	—	—	↑	1
	BXOR #xx:3,@ERd	B		4						$C \oplus (\#xx:3 \text{ of } @ERd) \rightarrow C$	—	—	—	—	↑	3
	BXOR #xx:3,@aa:8	B					4			$C \oplus (\#xx:3 \text{ of } @aa:8) \rightarrow C$	—	—	—	—	↑	3
	BXOR #xx:3,@aa:16	B					6			$C \oplus (\#xx:3 \text{ of } @aa:16) \rightarrow C$	—	—	—	—	↑	4
	BXOR #xx:3,@aa:32	B					8			$C \oplus (\#xx:3 \text{ of } @aa:32) \rightarrow C$	—	—	—	—	↑	5
BIXOR	BIXOR #xx:3,Rd	B	2							$C \oplus \neg(\#xx:3 \text{ of } Rd8) \rightarrow C$	—	—	—	—	↑	1
	BIXOR #xx:3,@ERd	B		4						$C \oplus \neg(\#xx:3 \text{ of } @ERd) \rightarrow C$	—	—	—	—	↑	3
	BIXOR #xx:3,@aa:8	B					4			$C \oplus \neg(\#xx:3 \text{ of } @aa:8) \rightarrow C$	—	—	—	—	↑	3
	BIXOR #xx:3,@aa:16	B					6			$C \oplus \neg(\#xx:3 \text{ of } @aa:16) \rightarrow C$	—	—	—	—	↑	4
	BIXOR #xx:3,@aa:32	B					8			$C \oplus \neg(\#xx:3 \text{ of } @aa:32) \rightarrow C$	—	—	—	—	↑	5

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code	Number of States ¹						
			#xx	Rn	@ERn	@ (d,ERn)	@-ERn/@ERn+	@aa	@ (d,PC)	@@aa				I					
Bcc	BRA d:8(BT d:8)	—						2			if condition is true then PC←PC+d else next;	Branch Condition	I	H	N	Z	V	C	Advanced
	BRA d:16(BT d:16)	—						4				Always	—	—	—	—	—	—	2
	BRN d:8(BF d:8)	—						2				Never	—	—	—	—	—	—	2
	BRN d:16(BF d:16)	—						4					—	—	—	—	—	—	3
	BHI d:8	—						2				CvZ=0	—	—	—	—	—	—	2
	BHI d:16	—						4					—	—	—	—	—	—	3
	BLS d:8	—						2				CvZ=1	—	—	—	—	—	—	2
	BLS d:16	—						4					—	—	—	—	—	—	3
	BCC d:8(BHS d:8)	—						2				C=0	—	—	—	—	—	—	2
	BCC d:16(BHS d:16)	—						4					—	—	—	—	—	—	3
	BCS d:8(BLO d:8)	—						2				C=1	—	—	—	—	—	—	2
	BCS d:16(BLO d:16)	—						4					—	—	—	—	—	—	3
	BNE d:8	—						2				Z=0	—	—	—	—	—	—	2
	BNE d:16	—						4					—	—	—	—	—	—	3
	BEQ d:8	—						2				Z=1	—	—	—	—	—	—	2
	BEQ d:16	—						4					—	—	—	—	—	—	3
	BVC d:8	—						2				V=0	—	—	—	—	—	—	2
	BVC d:16	—						4					—	—	—	—	—	—	3
	BVS d:8	—						2				V=1	—	—	—	—	—	—	2
	BVS d:16	—						4					—	—	—	—	—	—	3
	BPL d:8	—						2				N=0	—	—	—	—	—	—	2
	BPL d:16	—						4					—	—	—	—	—	—	3
	BMI d:8	—						2				N=1	—	—	—	—	—	—	2
	BMI d:16	—						4					—	—	—	—	—	—	3
	BGE d:8	—						2				N@V=0	—	—	—	—	—	—	2
	BGE d:16	—						4					—	—	—	—	—	—	3
	BLT d:8	—						2				N@V=1	—	—	—	—	—	—	2
	BLT d:16	—						4					—	—	—	—	—	—	3
	BGT d:8	—						2				Zv(N@V)=0	—	—	—	—	—	—	2
	BGT d:16	—						4					—	—	—	—	—	—	3
	BLE d:8	—						2				Zv(N@V)=1	—	—	—	—	—	—	2

- System control instructions

Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code					Number of States*1
		#xx	Rn	@ERn	@(d,ERn)	@-ERn/@ERn+	@aa	@(d,PC)	@aa		I	H	N	Z	V	
																Advanced
TRAPA	TRAPA #xx:2	—								PC→@-SP,CCR→@-SP, EXR→@-SP,<vector>→PC	1	—	—	—	—	8 [9]
RTE	RTE	—								EXR←@SP+,CCR←@SP+, PC←@SP+	↑	↑	↑	↑	↑	5 [9]
SLEEP	SLEEP	—								Transition to the power-down state	—	—	—	—	—	2
LDC	LDC #xx:8,CCR	B	2							#xx:8→CCR	↑	↑	↑	↑	↑	1
	LDC #xx:8,EXR	B	4							#xx:8→EXR	—	—	—	—	—	2
	LDC Rs,CCR	B		2						Rs8→CCR	↑	↑	↑	↑	↑	1
	LDC Rs,EXR	B		2						Rs8→EXR	—	—	—	—	—	1
	LDC @ERs,CCR	W			4					@ERs→CCR	↑	↑	↑	↑	↑	3
	LDC @ERs,EXR	W			4					@ERs→EXR	—	—	—	—	—	3
	LDC @(d:16,ERs),CCR	W				6				@(d:16,ERs)→CCR	↑	↑	↑	↑	↑	4
	LDC @(d:16,ERs),EXR	W				6				@(d:16,ERs)→EXR	—	—	—	—	—	4
	LDC @(d:32,ERs),CCR	W				10				@(d:32,ERs)→CCR	↑	↑	↑	↑	↑	6
	LDC @(d:32,ERs),EXR	W				10				@(d:32,ERs)→EXR	—	—	—	—	—	6
	LDC @ERs+,CCR	W					4			@ERs→CCR,ERs32+2→ERs32	↑	↑	↑	↑	↑	4
	LDC @ERs+,EXR	W					4			@ERs→EXR,ERs32+2→ERs32	—	—	—	—	—	4
	LDC @aa:16,CCR	W						6		@aa:16→CCR	↑	↑	↑	↑	↑	4
	LDC @aa:16,EXR	W						6		@aa:16→EXR	—	—	—	—	—	4
	LDC @aa:32,CCR	W							8	@aa:32→CCR	↑	↑	↑	↑	↑	5
	LDC @aa:32,EXR	W							8	@aa:32→EXR	—	—	—	—	—	5
STC	STC CCR,Rd	B		2						CCR→Rd8	—	—	—	—	—	1
	STC EXR,Rd	B		2						EXR→Rd8	—	—	—	—	—	1
	STC CCR,@ERd	W			4					CCR→@ERd	—	—	—	—	—	3
	STC EXR,@ERd	W			4					EXR→@ERd	—	—	—	—	—	3
	STC CCR,@(d:16,ERd)	W				6				CCR→@(d:16,ERd)	—	—	—	—	—	4
	STC EXR,@(d:16,ERd)	W				6				EXR→@(d:16,ERd)	—	—	—	—	—	4
	STC CCR,@(d:32,ERd)	W				10				CCR→@(d:32,ERd)	—	—	—	—	—	6
	STC EXR,@(d:32,ERd)	W				10				EXR→@(d:32,ERd)	—	—	—	—	—	6
	STC CCR,@-ERd	W					4			ERd32-2→ERd32,CCR→@ERd	—	—	—	—	—	4
	STC EXR,@-ERd	W					4			ERd32-2→ERd32,EXR→@ERd	—	—	—	—	—	4
	STC CCR,@aa:16	W						6		CCR→@aa:16	—	—	—	—	—	4
	STC EXR,@aa:16	W						6		EXR→@aa:16	—	—	—	—	—	4
	STC CCR,@aa:32	W							8	CCR→@aa:32	—	—	—	—	—	5
	STC EXR,@aa:32	W							8	EXR→@aa:32	—	—	—	—	—	5
ANDC	ANDC #xx:8,CCR	B	2							CCR^#xx:8→CCR	↑	↑	↑	↑	↑	1
	ANDC #xx:8,EXR	B	4							EXR^#xx:8→EXR	—	—	—	—	—	2
ORC	ORC #xx:8,CCR	B	2							CCR v #xx:8→CCR	↑	↑	↑	↑	↑	1
	ORC #xx:8,EXR	B	4							EXR v #xx:8→EXR	—	—	—	—	—	2
XORC	XORC #xx:8,CCR	B	2							CCR@#xx:8→CCR	↑	↑	↑	↑	↑	1
	XORC #xx:8,EXR	B	4							EXR@#xx:8→EXR	—	—	—	—	—	2
NOP	NOP	—							2	PC←PC+2	—	—	—	—	—	1

- Block transfer instructions

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes)								Operation	Condition Code						Number of States ¹
			#xx	Rn	@ERn	@(d,ERn)	@-ERw/@ERn+	@aa	@(d,PC)	@@aa		l	I	H	N	Z	V	
EEPMOV	EEPMOV.B	—									4	if R4L≠0 Repeat @ER5→@ER6 ER5+1→ER5 ER6+1→ER6 R4L-1→R4L Until R4L=0 else next;	—	—	—	—	—	4+2n ²
	EEPMOV.W	—									4	if R4≠0 Repeat @ER5→@ER6 ER5+1→ER5 ER6+1→ER6 R4-1→R4 Until R4=0 else next;	—	—	—	—	—	4+2n ²

Notes: ^{*1}: The number of states is the number of states required for execution when the instruction and its operands are located in on-chip memory.

^{*2}: n is the initial value of R4L or R4.

[1] 7 states when the number of restored/saved registers is 2, 9 states when 3, and 11 states when 4.

[2] Cannot be used with these series.

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

[4] Set to 1 when there is a carry from or borrow to bit 27; otherwise cleared to 0.

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

[6] Set to 1 if the divisor is negative; otherwise cleared to 0.

[7] Set to 1 if the divisor is zero; otherwise cleared to 0.

[8] Set to 1 if the quotient is negative; otherwise cleared to 0.

[9] When EXR is valid, the number of states is increased by 1.

Number of States Required for Execution: The number of states shown in the instruction set table is the number of states required for execution when the op code and operand data are located in a one-cycle area on which word access is possible, such as on-chip memory. When the op code or operand data is accessed from an on-chip supporting module or an external address, the number of states increases as shown in the table below.

Table 2.2 Number of States per Cycle

Cycle	Access Conditions						
	On-Chip Memory	On-Chip Supporting Module		External Device			
		8-Bit Bus	16-Bit Bus	8-Bit Bus		16-Bit Bus*	
				2-State Access	3-State Access	2-State Access	3-State Access
Instruction fetch	1	4	2	4	6 + 2m	2	3 + m
Branch instruction read							
Stack operation							
Byte data access		2		2	3 + m		
Word data access		4		4	6 + 2m		
Internal operation	1	1	1	1	1	1	1

Legend

m: Number of wait states inserted in external device access

Note: * Cannot be used with these series.

Table 2.3 Condition Code Notation

Symbol	Meaning
↑ ↓	Changes according operation result.
*	Indeterminate (value not guaranteed).
0	Always cleared to 0.
1	Always set to 1.
—	Not affected by operation result.

Table 2.4 Operation Notation

Rd	General register (destination)*
Rs	General register (source)*
Rn	General register*
ERn	General register (32-bit register)
(EAd)	Destination operand
(EAs)	Source operand
EXR	Extended control register
CCR	Condition-code register
N	N (negative) flag in CCR
Z	Z (zero) flag in CCR
V	V (overflow) flag in CCR
C	C (carry) flag in CCR
PC	Program counter
SP	Stack pointer
#IMM	Immediate data
disp	Displacement
+	Addition
–	Subtraction
×	Multiplication
÷	Division
^	Logical AND
∨	Logical OR
⊕	Logical exclusive OR
→	Move
¬	NOT (logical complement)
() < >	Operand contents
:8/:16/:24/:32	8-, 16-, 24-, or 32-bit length

Note: * General registers include 8-bit registers (R0H to R7H, R0L to R7L), 16-bit registers (R0 to R7, E0 to E7), and 32-bit registers (ER0 to ER7).

2.6 Basic Timing

The CPU is driven by a system clock, denoted by the symbol \emptyset . The period from one rising edge of \emptyset to the next is referred to as a “state.” The memory cycle or bus cycle consists of one, two, or three states. Different methods are used to access on-chip memory, on-chip supporting modules, and the external address space.

Basic Clock Timing: An external clock is input to the EXTAL pin, or a crystal oscillator is connected to the EXTAL pin, to generate the system clock (\emptyset). An external clock or crystal oscillator of the same frequency as the \emptyset clock should be used.

The following methods can be used to generate the system clock (\emptyset):

1. Input an external clock of the same frequency as the system clock to the EXTAL pin
2. Connect a crystal oscillator of the same frequency as the system clock to the EXTAL and XTAL pins
3. Connect a 32.768 kHz crystal oscillator to the OSC1 and OSC2 pins

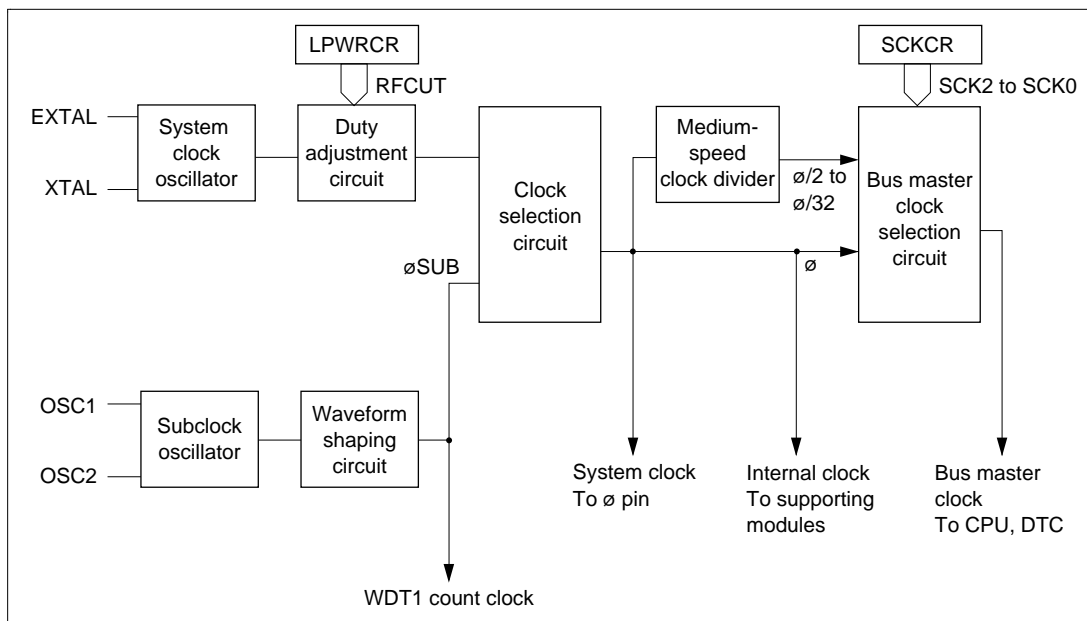


Figure 2.5 Basic Clock Timing

CPU Read/Write Cycles: The CPU operates on the basis of the \emptyset clock. One \emptyset clock cycle is called a state, and a bus cycle consists of one, two, or three states. Different access methods are used for on-chip memory, on-chip supporting modules, and external address space. Access to the external address space can be controlled by the bus controller.

On-Chip Memory: On-chip memory is accessed in one state. The data bus is 16 bits wide, permitting both byte and word access.

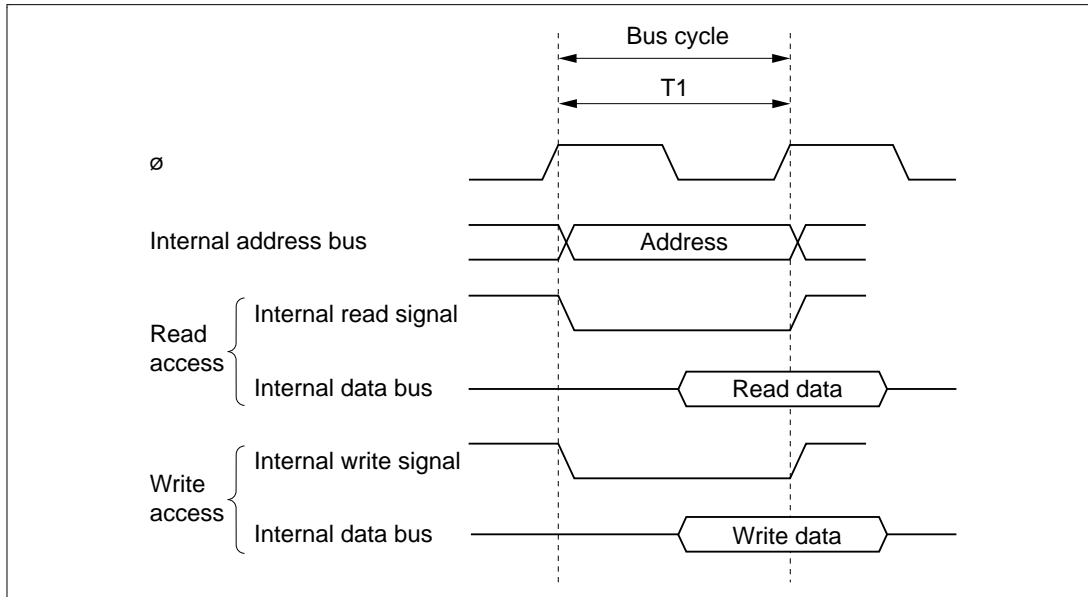


Figure 2.6 On-Chip Memory Access Cycle (One-state Access)

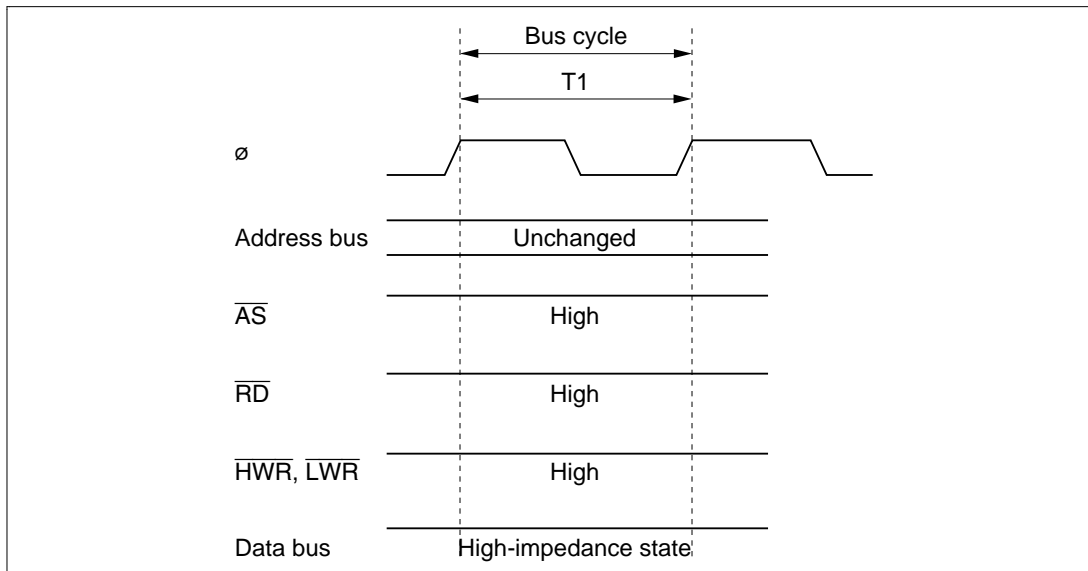


Figure 2.7 Pin States During On-Chip Memory Access

On-Chip Supporting Module Access Timing: The on-chip supporting modules are accessed in two states. The data bus is either 8 bits or 16 bits wide, depending on the particular internal I/O register being accessed.

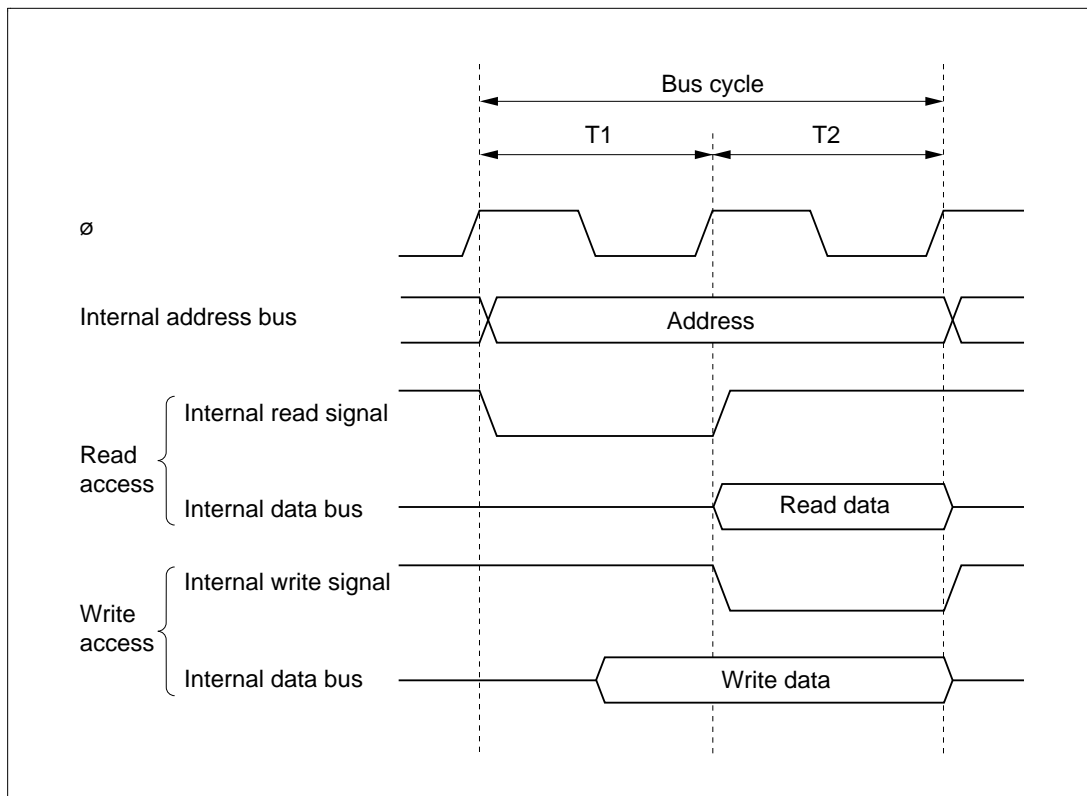


Figure 2.8 On-Chip Supporting Module Access Cycle (Two-state Access)

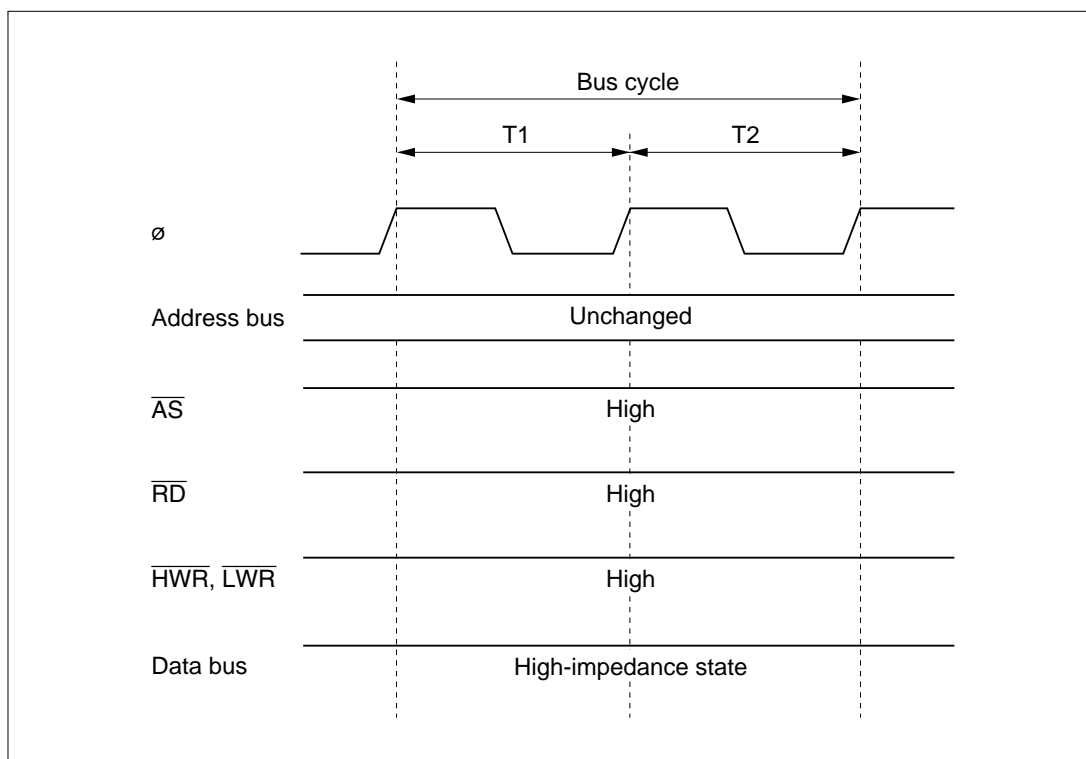


Figure 2.9 Pin States during On-Chip Supporting Module Access

External Address Space Access Timing: The external address space is accessed with an 8-bit or 16-bit data bus width in a two-state or three-state bus cycle. In three-state access, wait states can be inserted. For further details, refer to section 3.2, Bus Controller (BSC).

2.7 Processing States

The H8S/2000 CPU has five processing states: the reset state, program execution state, exception-handling state, bus-released state, and power-down state.

Reset State: State in which the CPU and all on-chip supporting modules are initialized and halted

Program Execution State: State in which the CPU executes the program sequentially

Exception-Handling State: Transient state in which exception handling is executed as the result of a reset, interrupt, or trap instruction exception handling source

Bus-Released State: State in which the external bus is released in response to a bus request signal from a bus master other than the CPU

Power-Down State: State in which CPU operation is stopped, and power consumption is kept low (sleep mode, software standby mode, hardware standby mode, subsleep mode, watch mode). The power-down state also includes medium-speed mode, module stop mode, and subactive mode.

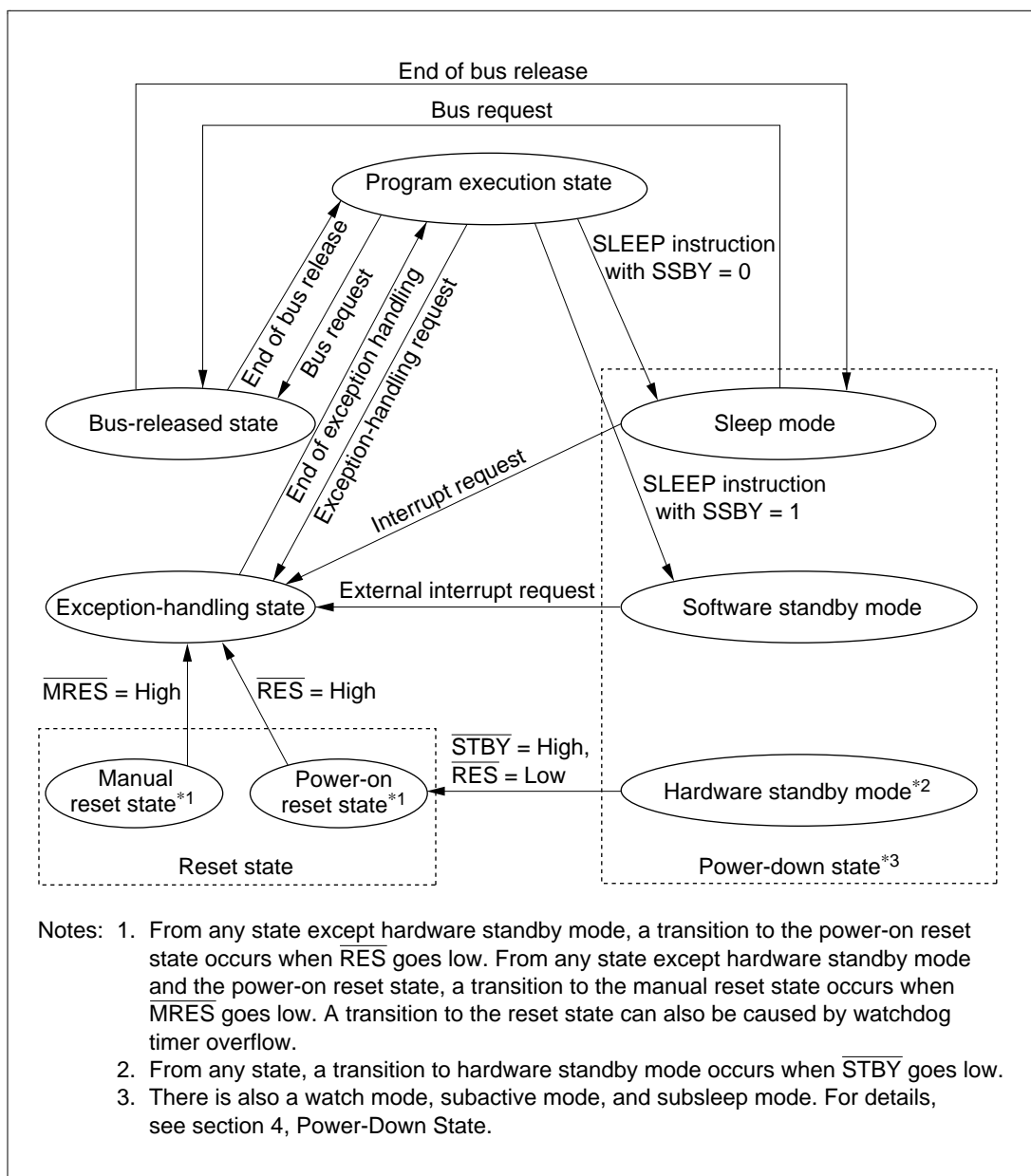


Figure 2.10 State Transition Diagram

2.8 Exception Handling

H8S/2000 CPU exception handling is initiated by a reset, a trap instruction, or an interrupt. A priority system is provided for exception handling, and simultaneously generated exceptions are handled in order of priority.

Table 2.5 Exception Types and Priority

Priority	Exception Type	Start of Exception Handling
High	Reset	Starts immediately after a low-to-high transition at the $\overline{\text{RES}}$ pin, or when the watchdog timer overflows.
	Trace	Starts when execution of the current instruction or exception handling ends, if the trace (T) bit is set to 1
	Direct transition	Started by a direct transition resulting from execution of a SLEEP instruction
	Interrupt	Starts when execution of the current instruction or exception handling ends, if an interrupt request has been issued
Low	Trap instruction (TRAPA)	Started by execution of a trap instruction (TRAPA)

Exception Handling Operation: Exceptions originate from various sources. Trap instruction exception handling is always accepted in the program execution state. Trap instructions and interrupts are handled as follows:

1. The program counter (PC), condition code register (CCR), and extend register (EXR) are pushed onto the stack.
2. The interrupt mask bits are updated. The T bit is cleared to 0.
3. A vector address corresponding to the exception source is generated, and program execution starts from that address.

For a reset exception, steps 2 and 3 above are carried out.

Table 2.6 Exception Vector Table

Exception Source		Vector Number	Vector Address* ¹
			Advanced Mode
Power-on reset		0	H'0000 to H'0003
Manual reset		1	H'0004 to H'0007
Reserved for system use		2	H'0008 to H'000B
		3	H'000C to H'000F
		4	H'0010 to H'0013
Trace		5	H'0014 to H'0017
Direct transition		6	H'0018 to H'001B
External interrupt	NMI	7	H'001C to H'001F
Trap instruction (4 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
Reserved for system use		12	H'0030 to H'0033
		13	H'0034 to H'0037
		14	H'0038 to H'003B
		15	H'003C to H'003F
External interrupt	IRQ0	16	H'0040 to H'0043
	IRQ1	17	H'0044 to H'0047
	IRQ2	18	H'0048 to H'004B
	IRQ3	19	H'004C to H'004F
	IRQ4	20	H'0050 to H'0053
	IRQ5	21	H'0054 to H'0057
	IRQ6	22	H'0058 to H'005B
	IRQ7	23	H'005C to H'005F
Internal interrupt* ²		24	H'0060 to H'0063
		to 123	to H'01EC to H'01EF

Notes: 1. Lower 16 bits of the address.

2. For details of internal interrupt vectors, see section 2.9, Interrupts.

2.9 Interrupts

Interrupts are controlled by the interrupt controller. There are a total of 62 interrupt sources, comprising nine external interrupts from the external pins (NMI, $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ7}}$), and 53 internal interrupts from on-chip supporting modules. A separate vector number is assigned to each interrupt.

Interrupt Control: Either of two interrupt control modes can be set by means of the INTM1 and INTM0 bits in the system control register (SYSCR).

The interrupt controller controls interrupts on the basis of the control mode set by the INTM1 and INTM0 bits, the interrupt priorities set by interrupt priority register (IPR), and the masking conditions set by the I bit in CCR and bits I2 to I0 in EXR.

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

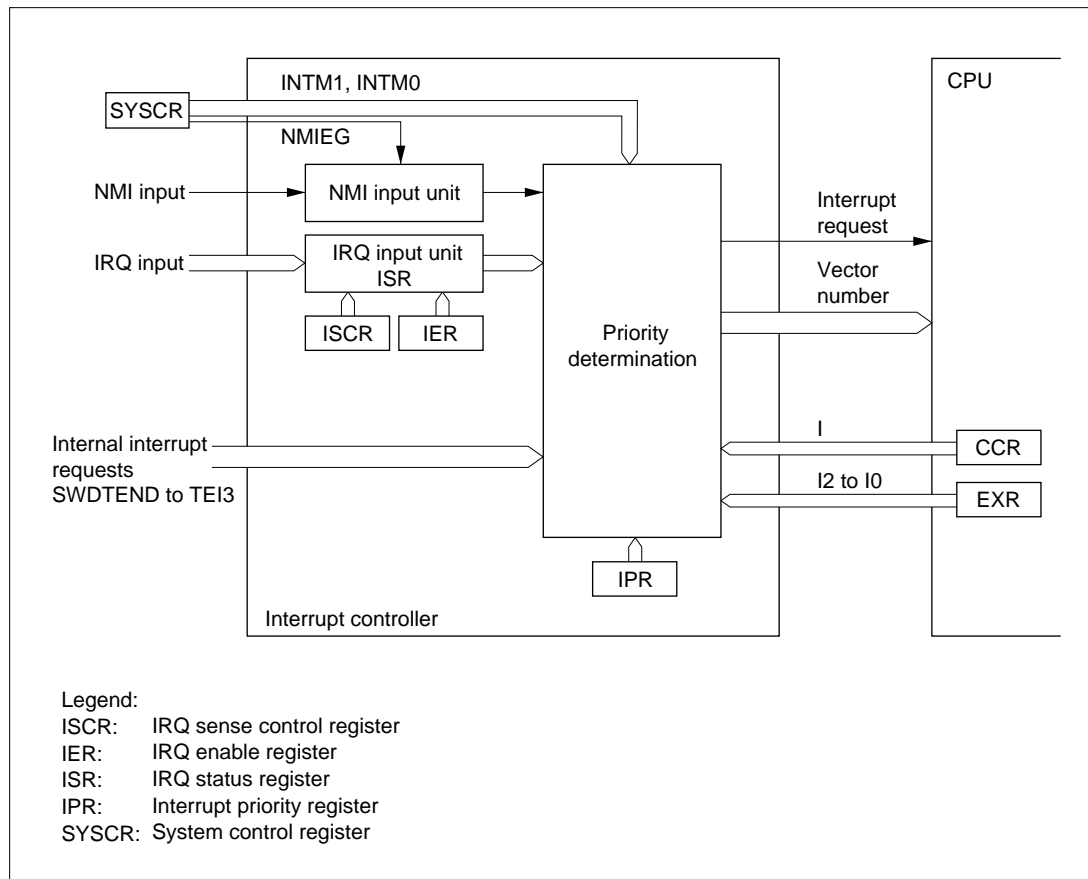


Figure 2.11 Block Diagram of Interrupt Controller

Table 2.7 Interrupt Control Modes

Interrupt Control Mode	SYSCR		Priority Setting Registers	Interrupt Mask Bits	Description
	INTM1	INTM0			
0	0	0	—	I	Interrupt mask control is performed by the I bit.
—	—	1	—	—	Setting prohibited
2	1	0	IPR	I2 to I0	8-level interrupt mask control is performed by bits I2 to I0. 8 priority levels can be set with IPR.
—	—	1	—	—	Setting prohibited

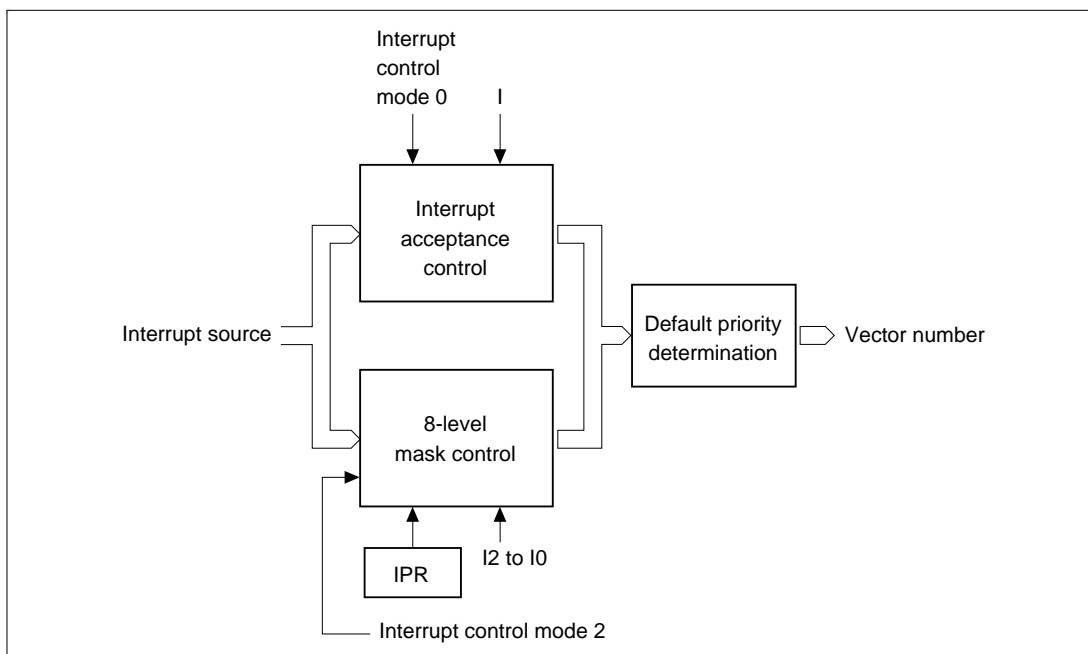


Figure 2.12 Block Diagram of Interrupt Control Operation

Interrupt Control Mode 0: Enabling and disabling of IRQ interrupts and on-chip supporting module interrupts can be set by means of the I bit in CCR. Interrupts are enabled when the I bit is cleared to 0, and disabled when set to 1.

Interrupt Control Mode 2: Eight-level masking can be implemented for IRQ interrupts and on-chip supporting module interrupts by comparing the interrupt mask level bits (I2 to I0) in EXR and the IPR priority level.

Table 2.8 Interrupt Sources, Vector Addresses, and Interrupt Priorities

Interrupt Source	Origin of Interrupt Source	Vector Number	Vector Address*		Priority
			Advanced Mode	IPR	
NMI	External pin	7	H'001C		High
IRQ0		16	H'0040	IPRA6–IPRA4	
IRQ1		17	H'0044	IPRA2–IPRA0	
IRQ2		18	H'0048	IPRB6–IPRB4	
IRQ3		19	H'004C		
IRQ4		20	H'0050	IPRB2–IPRB0	
IRQ5		21	H'0054		
IRQ6		22	H'0058	IPRC6–IPRC4	
IRQ7		23	H'005C		
SWDTEND (software activation interrupt end)	DTC	24	H'0060	IPRC2–IPRC0	
WOVI0 (interval timer 0)	Watchdog timer 0	25	H'0064	IPRD6–IPRD4	
PC break	PC break	27	H'006C	IPRE6–IPRE4	
ADI (A/D conversion end)	A/D	28	H'0070	IPRE2–IPRE0	
WOVI1 (interval timer 1)	Watchdog timer 1	29	H'0074		
TGI0A (TGR0A input capture/compare-match)	TPU channel 0	32	H'0080	IPRF6–IPRF4	
TGI0B (TGR0B input capture/compare-match)		33	H'0084		
TGI0C (TGR0C input capture/compare-match)		34	H'0088		
TGI0D (TGR0D input capture/compare-match)		35	H'008C		
TCI0V (overflow 0)		36	H'0090		
TGI1A (TGR1A input capture/compare-match)	TPU channel 1	40	H'00A0	IPRF2–IPRF0	
TGI1B (TGR1B input capture/compare-match)		41	H'00A4		
TCI1V (overflow 1)		42	H'00A8		
TCI1U (underflow 1)		43	H'00AC		Low

Note: * Lower 16 bits of the start address

Interrupt Source	Origin of Interrupt Source	Vector Number	Vector Address*	IPR	Priority
			Advanced Mode		
TGI2A (TGR2A input capture/ compare-match)	TPU channel 2	44	H'00B0	IPRG6–IPRG4	High
TGI2B (TGR2B input capture/ compare-match)		45	H'00B4		
TCI2V (overflow 2)		46	H'00B8		
TCI2U (underflow 2)		47	H'00BC		
TGI3A (TGR3A input capture/ compare-match)	TPU channel 3	48	H'00C0	IPRG2–IPRG0	
TGI3B (TGR3B input capture/ compare-match)		49	H'00C4		
TGI3C (TGR3C input capture/ compare-match)		50	H'00C8		
TGI3D (TGR3D input capture/ compare-match)		51	H'00CC		
TCI3V (overflow 3)		52	H'00D0		
TGI4A (TGR4A input capture/ compare-match)	TPU channel 4	56	H'00E0	IPRH6–IPRH4	
TGI4B (TGR4B input capture/ compare-match)		57	H'00E4		
TCI4V (overflow 4)		58	H'00E8		
TCI4U (underflow 4)		59	H'00EC		
TGI5A (TGR5A input capture/ compare-match)	TPU channel 5	60	H'00F0	IPRH2–IPRH0	
TGI5B (TGR5B input capture/ compare-match)		61	H'00F4		
TCI5V (overflow 5)		62	H'00F8		
TCI5U (underflow 5)		63	H'00FC		
CMIA0 (compare-match A)	8-bit timer channel 0	64	H'0100	IPRJ6–IPRJ4	
CMIB0 (compare-match B)		65	H'0104		
OVI0 (overflow 0)		66	H'0108		
CMIA1 (compare-match A)	8-bit timer channel 1	68	H'0110	IPRI2–IPRI0	
CMIB1 (compare-match B)		69	H'0114		
OVI1 (overflow 0)		70	H'0118		Low

Note: * Lower 16 bits of the start address

Interrupt Source	Origin of Interrupt Source	Vector Number	Vector Address*	IPR	Priority
			Advanced Mode		
ERI0 (receive error 0)	SCI channel 0	80	H'0140	IPRJ2–IPRJ0	High
RXI0 (reception completed 0)		81	H'0144		
TXI0 (transmit data empty 0)		82	H'0148		
TEI0 (transmission end 0)		83	H'014C		
ERI1 (receive error 1)	SCI channel 1	84	H'0150	IPRK6–IPRK4	
RXI1 (reception completed 1)		85	H'0154		
TXI1 (transmit data empty 1)		86	H'0158		
TEI1 (transmission end 1)		87	H'015C		
ERI2 (receive error 2)	SCI channel 2	88	H'0160	IPRK2–IPRK0	
RXI2 (reception completed 2)		89	H'0164		
TXI2 (transmit data empty 2)		90	H'0168		
TEI2 (transmission end 2)		91	H'016C		
ERI3 (receive error 3)	SCI channel 3	120	H'01E0	IPRO6–IPRO4	
RXI3 (reception completed 3)		121	H'01E4		
TXI3 (transmit data empty 3)		122	H'01E8		
TEI3 (transmission end 3)		123	H'01EC		Low

Note: * Lower 16 bits of the start address.

2.10 MCU Operating Modes

These series have four operating modes, determined by the setting of the mode pins (MD1 and MD0).

Mode 4: The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is disabled.

Pins 13 to 10 and ports A, B, and C function as an address bus, ports D and E function as a data bus, and part of port F carries bus control signals.

The initial bus mode after a reset is 16 bits, with 16-bit access to all areas. However, note that if 8-bit access is designated by the bus controller for all areas, the bus mode switches to 8 bits.

Mode 5: The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is disabled.

Pins 13 to 10 and ports A, B, and C function as an address bus, ports D and E function as a data bus, and part of port F carries bus control signals.

The initial bus mode after a reset is 8 bits, with 8-bit access to all areas. However, note that if at least one area is designated for 16-bit access by the bus controller, the bus mode switches to 16 bits and port E becomes a data bus.

Mode 6: The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is enabled.

Pins 13 to 10 and ports A, B, and C function as input ports immediately after a reset. They can each be set to output addresses: by settings in the pin state control register in the case of pins 13 to 10 and ports A and B, and by setting the corresponding bits in the data direction register (DDR) to 1 in the case of port C. Ports D and E function as a data bus, and part of port F carries bus control signals.

The initial bus mode after a reset is 8 bits, with 8-bit access to all areas. However, note that if at least one area is designated for 16-bit access by the bus controller, the bus mode switches to 16 bits and port E becomes a data bus.

Mode 7: The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is enabled, but external addresses cannot be accessed.

All I/O ports are available for use as input-output ports.

Table 2.9 MCU Operating Mode Selection

MCU Operating Mode				CPU Operating Mode	Description	On-Chip ROM	External Data Bus	
	MD2	MD1	MD0				Initial Width	Max. Width
4	1	0	0	Advanced	On-chip ROM disabled,	Disabled	16 bits	16 bits
5			1		expanded mode		8 bits	16 bits
6		1	0		On-chip ROM enabled,	Enabled	8 bits	16 bits
7			1		Single-chip mode		—	—

2.11 Address Maps

This section shows the address maps in each operating mode.

The address space is 16 Mbytes in modes 4 to 7 (advanced mode).

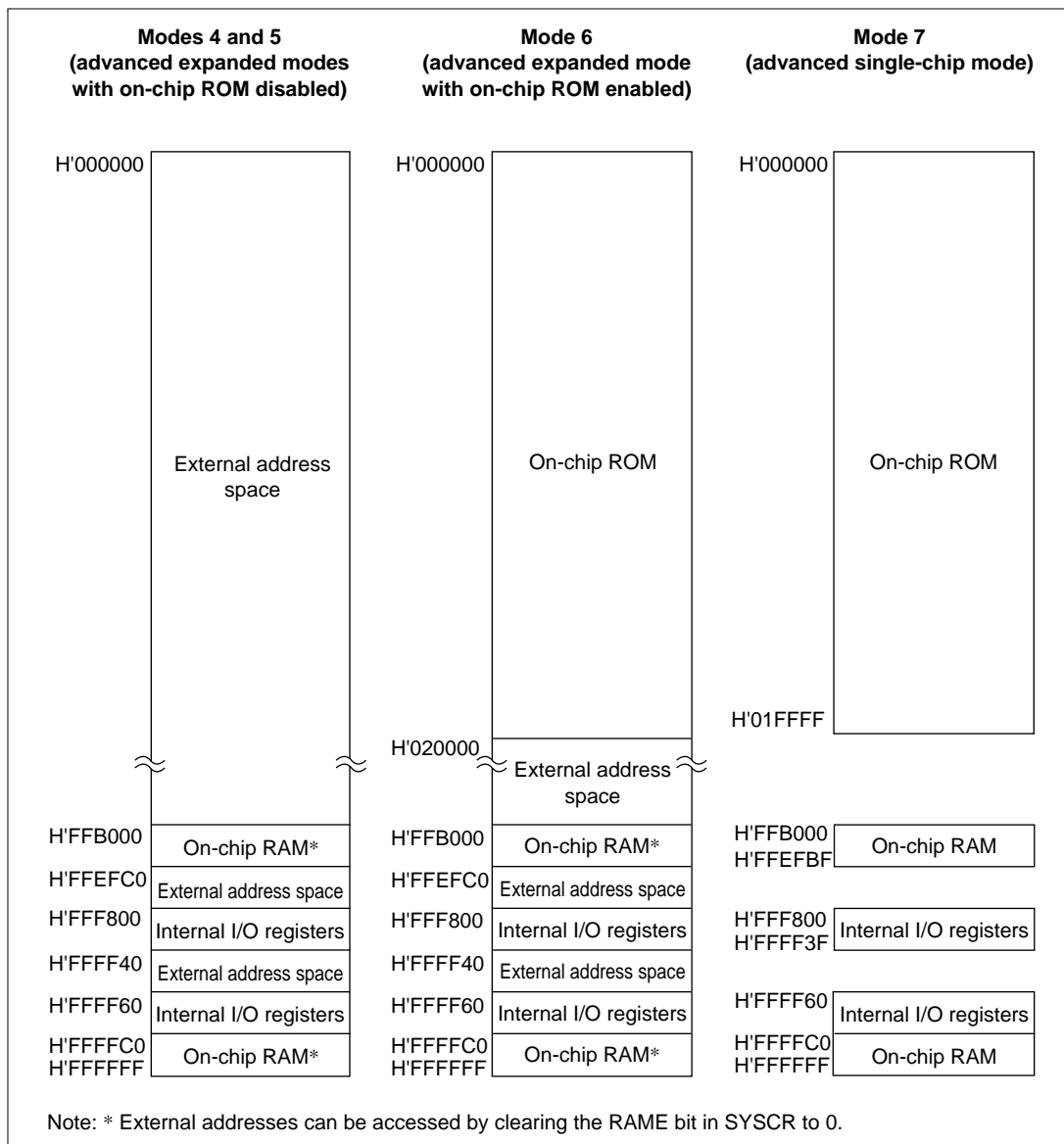


Figure 2.13 H8S/2237 and H8S/2227 Address Map in Each Operating Mode

Section 3 Supporting Modules

3.1 PC Break Controller (PBC)

These series have a two-channel on-chip PC break controller (PBC) that simplifies user program debugging. This function makes it easy to create a high-performance self-monitoring debugger, enabling programs to be debugged with the chip alone, without using a large-scale in-circuit emulator.

3.1.1 Features

- Two channels (A and B) can be set independently
- The following can be set as break conditions:
 - 24 address bits
 - Bit masking possible
 - Bus cycle
 - Instruction fetch
 - Data access: data read, data write, data read/write
 - Bus master
 - Either CPU or CPU/DTC can be selected
- The timing of PC break exception handling after the occurrence of a break condition is as follows:
 - Immediately before execution of the instruction fetched at the set address (instruction fetch)
 - Immediately after execution of the instruction that accesses data at the set address (data access)
- Module stop mode can be set
 - As the initial setting, PBC operation halted. Register access is enabled by exiting module stop mode.

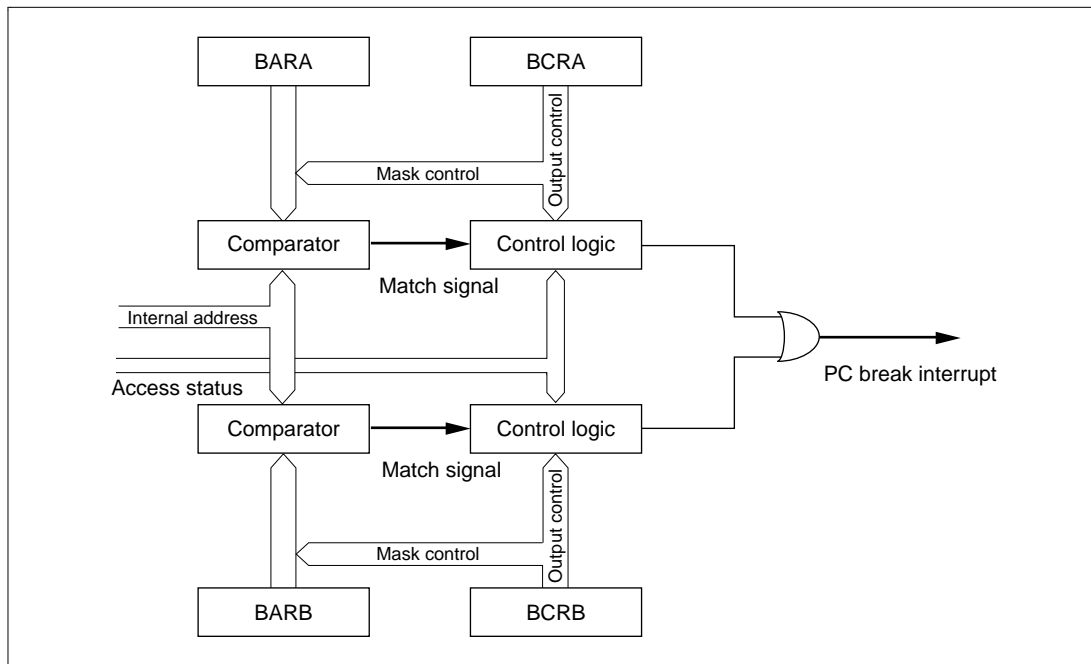


Figure 3.1 PBC Block Diagram

3.2 Bus Controller (BSC)

The bus controller (BSC) manages the external address space divided into eight areas. The bus specifications, such as bus width and number of access states, can be set independently for each area, enabling multiple memories to be connected easily. The bus controller also has a bus arbitration function, and controls the operation of the internal bus masters: the CPU and data transfer controller (DTC).

3.2.1 Features

- Manages external address space in area units
 - In advanced mode, manages the external space as 8 areas of 2-Mbytes
 - Bus specifications can be set independently for each area
 - Burst ROM interfaces can be set
- Basic bus interface
 - Chip select ($\overline{CS0}$ to $\overline{CS7}$) can be output for areas 0 to 7
 - 8-bit access or 16-bit access can be selected for each area
 - 2-state access or 3-state access can be selected for each area
 - Program wait states can be inserted for each area
- Burst ROM interface
 - Burst ROM interface can be set for area 0
 - Choice of 1- or 2-state burst access
- Idle cycle insertion
- Bus arbitration function
 - Includes a bus arbiter that arbitrates bus mastership among the CPU and DTC
- Other features
 - External bus release function

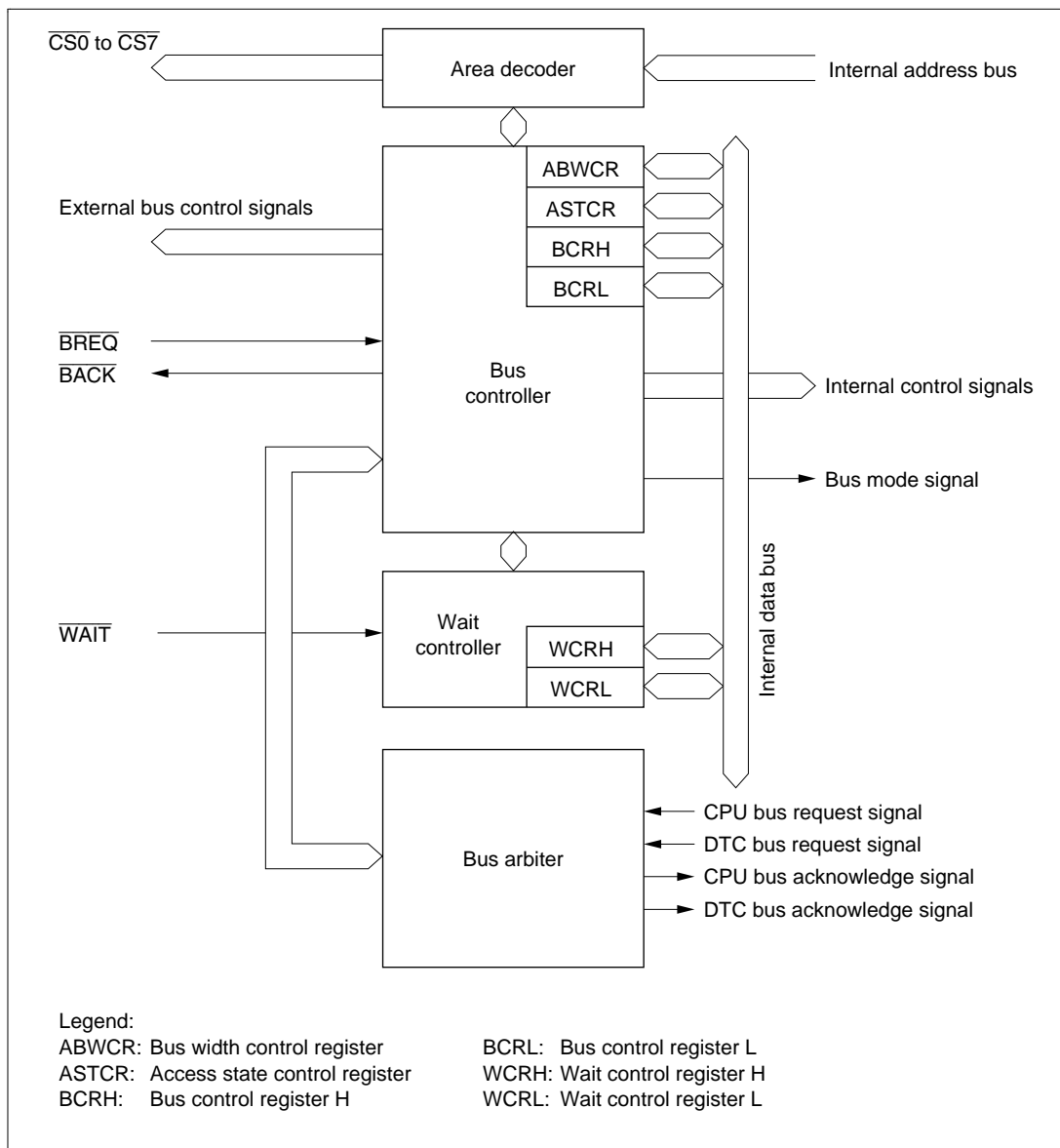


Figure 3.2 Bus Controller Block Diagram

3.2.2 Area Partitioning

The bus controller partitions the 16-Mbyte address space into eight areas, 0 to 7, in 2-Mbyte units, and performs bus control for external space in area units.

Area partitioning is only effective in expanded mode, and has no significance in single-chip mode.

H'000000	Area 0 (2 Mbytes)
H'1FFFFF H'200000	Area 1 (2 Mbytes)
H'3FFFFF H'400000	Area 2 (2 Mbytes)
H'5FFFFF H'600000	Area 3 (2 Mbytes)
H'7FFFFF H'800000	Area 4 (2 Mbytes)
H'9FFFFF H'A00000	Area 5 (2 Mbytes)
H'BFFFFF H'C00000	Area 6 (2 Mbytes)
H'DFFFFF H'E00000	Area 7 (2 Mbytes)
H'FFFFFF	

Figure 3.3 Overview of Area Partitioning

3.2.3 Bus Specifications

The external address space bus specifications consist of three elements: bus width, number of access states, and number of program wait states. The bus width and number of access states for on-chip memory and internal I/O registers are fixed, and are not affected by the bus controller.

Bus specifications can be set as shown below by means of the bus controller control registers.

Table 3.1 Bus Specifications for Each Area (Basic Bus Interface)

ABWCR	ASTCR	WCRH, WCRL		Bus Specifications (Basic Bus Interface)		
ABWn	ASTn	Wn1	Wn0	Bus Width	Access States	Program Wait States
0	0	—	—	16	2	0
	1	0	0		3	0
			1			1
		1	0			2
			1			3
1	0	—	—	8	2	0
	1	0	0		3	0
			1			1
		1	0			2
			1			3

3.2.4 Memory Interfaces

These series' memory interfaces comprise (1) a **basic bus interface** that allows direct connection of ROM, SRAM, and so on; and (2) a **burst ROM interface** that allows direct connection of burst ROM. The interface can be designated independently for each area.

Basic Bus Interface

This interface can be designated for areas 0 to 7. When external space is accessed, the chip select signal ($\overline{CS0}$ to $\overline{CS7}$) for the relevant area (0 to 7) can be output.

In 3-state access space, 0 to 3 program wait states or a pin wait by means of the \overline{WAIT} pin can be inserted.

After a reset, all areas are designated as basic bus interface, 3-state access space (the bus width is determined by the MCU operating mode).

Basic Bus Timing

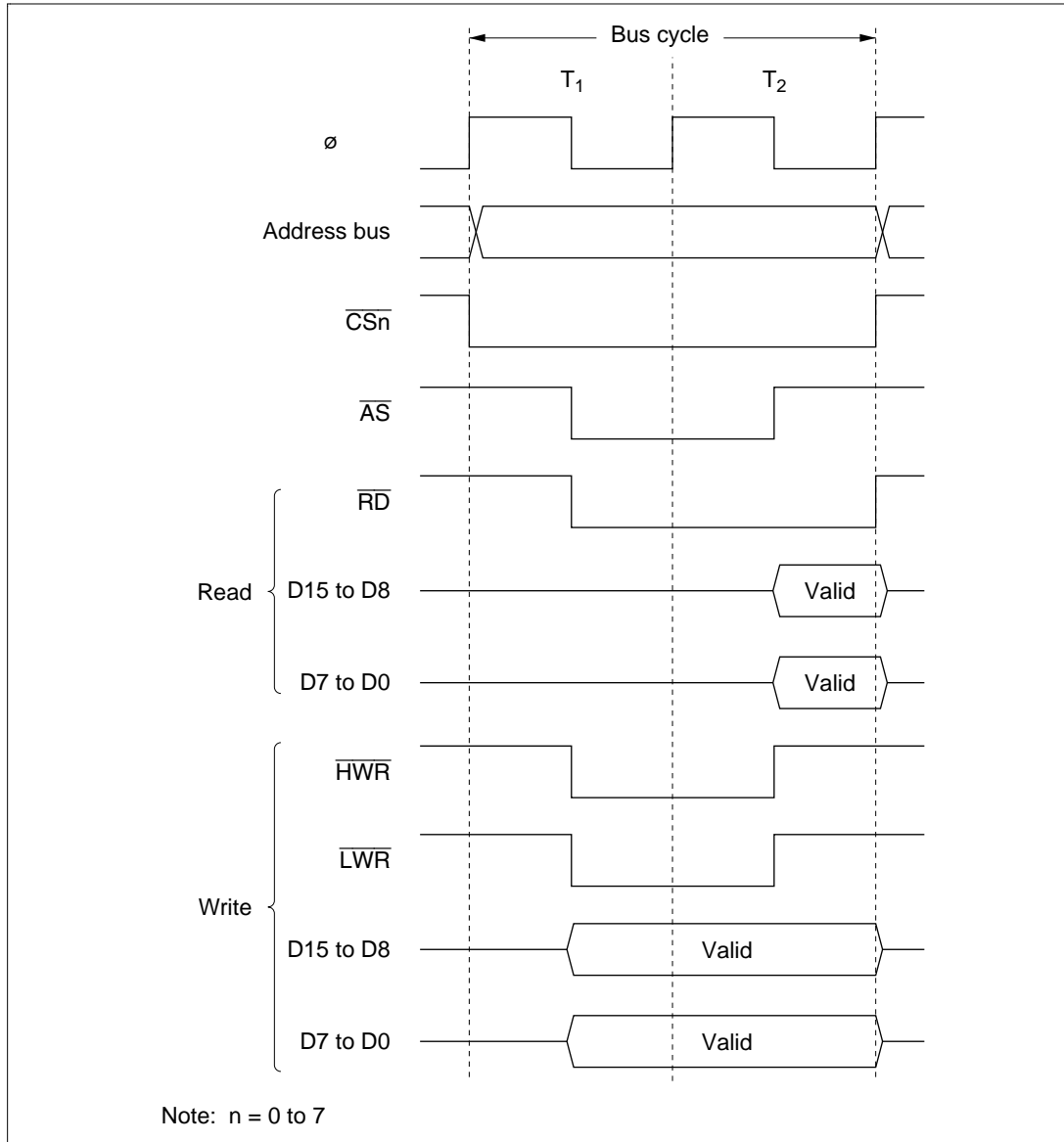


Figure 3.4 Basic Bus Timing (Word Access to 16-Bit 2-State Access Space)

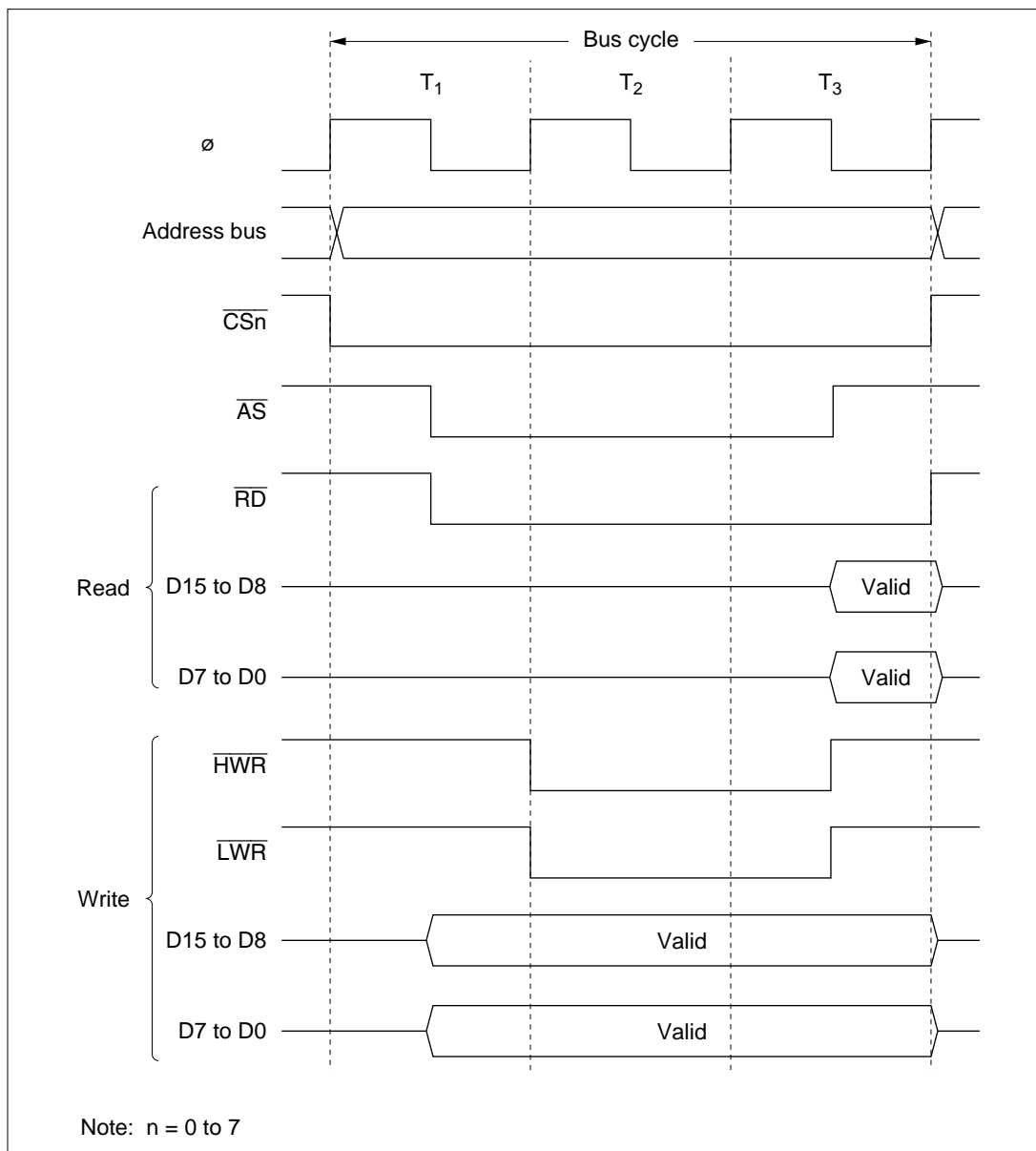


Figure 3.5 Basic Bus Timing (Word Access to 16-Bit 3-State Access Space)

Burst ROM Interface

External space area 0 can be designated as burst ROM space, and burst ROM space interfacing can be performed. The burst ROM space interface enables 16-bit configuration ROM with burst access capability to be accessed at high speed.

Consecutive burst accesses of a maximum 4 words or 8 words can be performed for CPU instruction fetches only. One or two states can be selected for burst access.

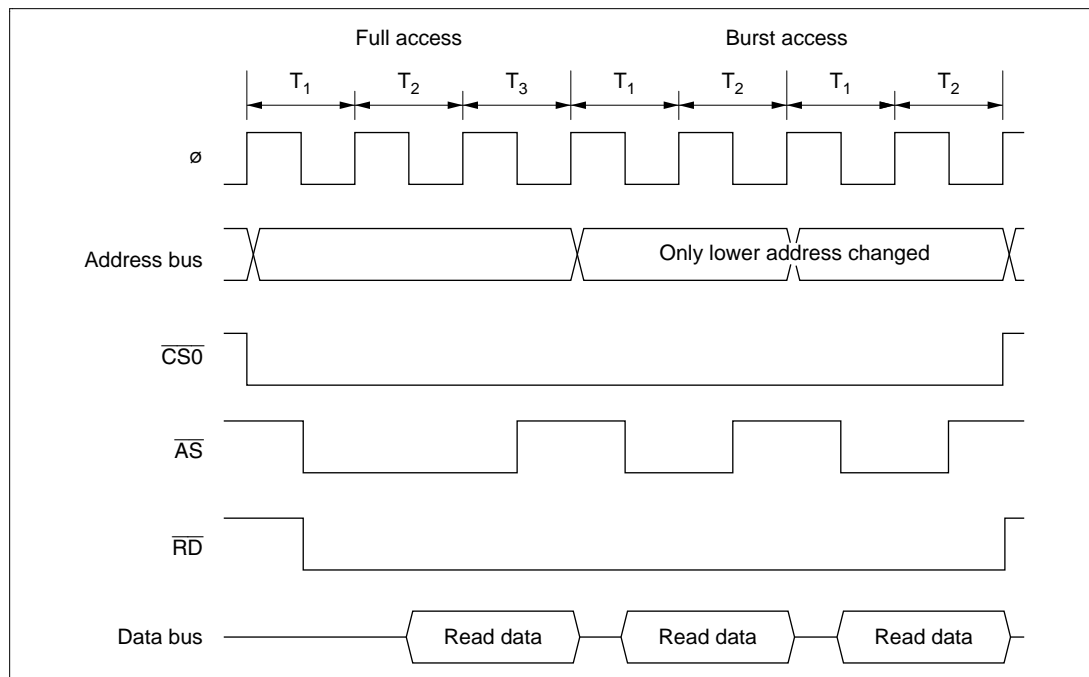


Figure 3.6 Example of Burst ROM Access Timing (When $AST0 = BRSTS1 = 1$)

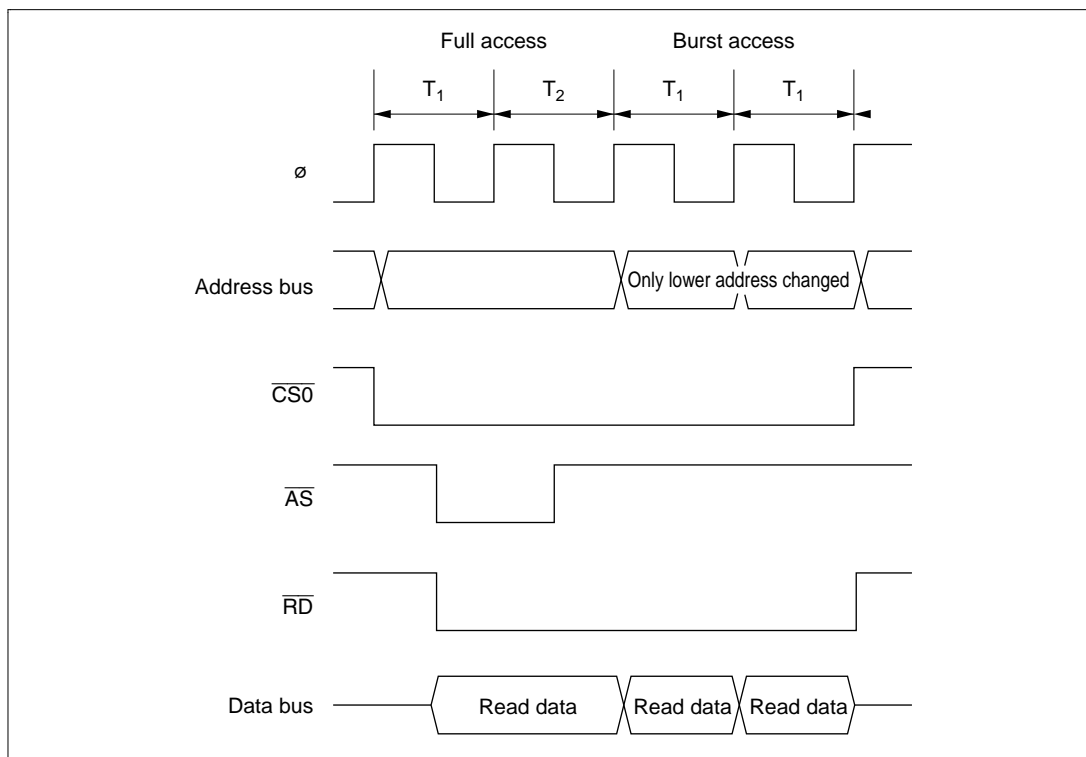


Figure 3.7 Example of Burst ROM Access Timing (When $AST0 = BRSTS1 = 0$)

3.3 Data Transfer Controller (DTC)

The data transfer controller (DTC) is activated by an interrupt or software, and can transfer data without imposing any load on the CPU.

3.3.1 Features

- Transfer possible over any number of channels
 - Transfer information is stored in memory
 - One activation source can trigger a number of data transfers (chain transfer)
- Variety of transfer modes
 - Normal, repeat, and block transfer modes available
 - Incrementing, decrementing, and fixing of source and destination addresses can be selected
- Direct specification of 16-Mbyte address space possible
- Transfer can be set in byte or word units
- A CPU interrupt can be requested for the interrupt that activated the DTC
 - An interrupt request can be issued to the CPU after one data transfer ends
 - An interrupt request can be issued to the CPU after all specified data transfers have ended
- Can be activated by software
- Module stop mode can be set
 - The initial setting enables DTC registers to be accessed. DTC operation is halted by setting module stop mode.

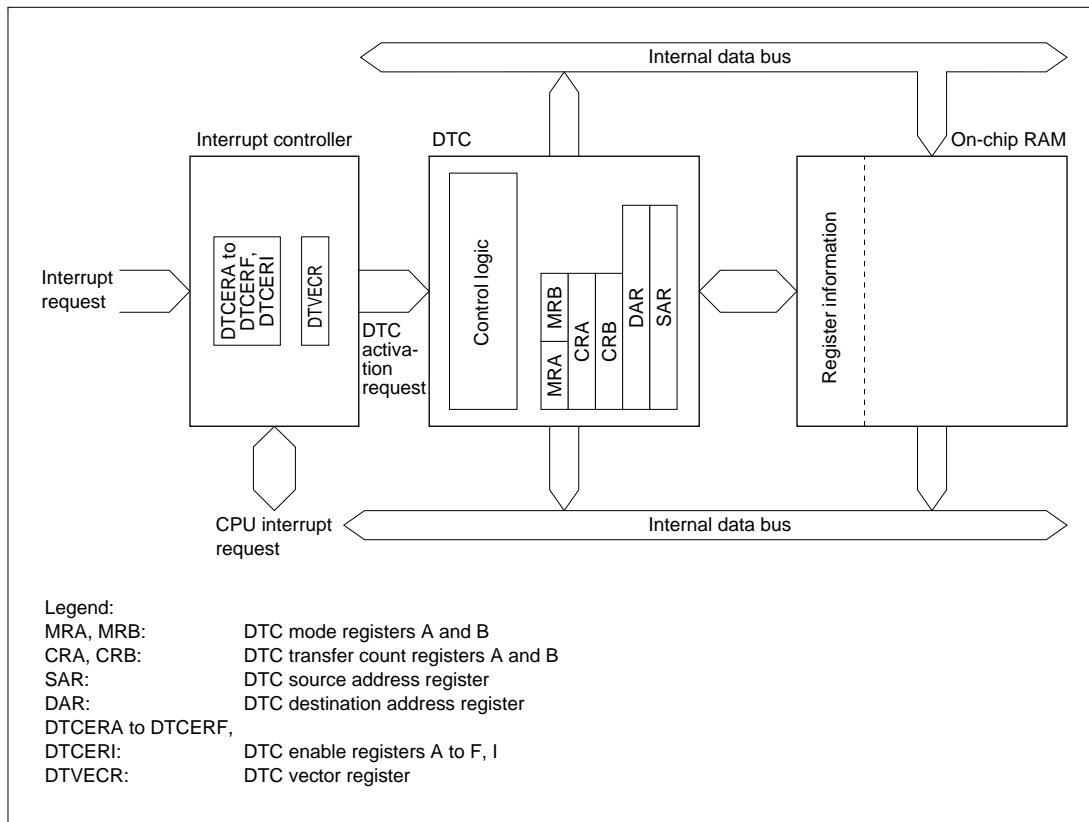


Figure 3.8 DTC Block Diagram

3.3.2 Data Transfer Operation

The DTC reads register information previously stored in memory, and transfers data on the basis of that register information. After the data transfer, it writes updated register information back to memory.

Pre-storage of register information in memory makes it possible to transfer data over any required number of channels. The DTC can also execute a number of transfers with a single activation (chain transfer).

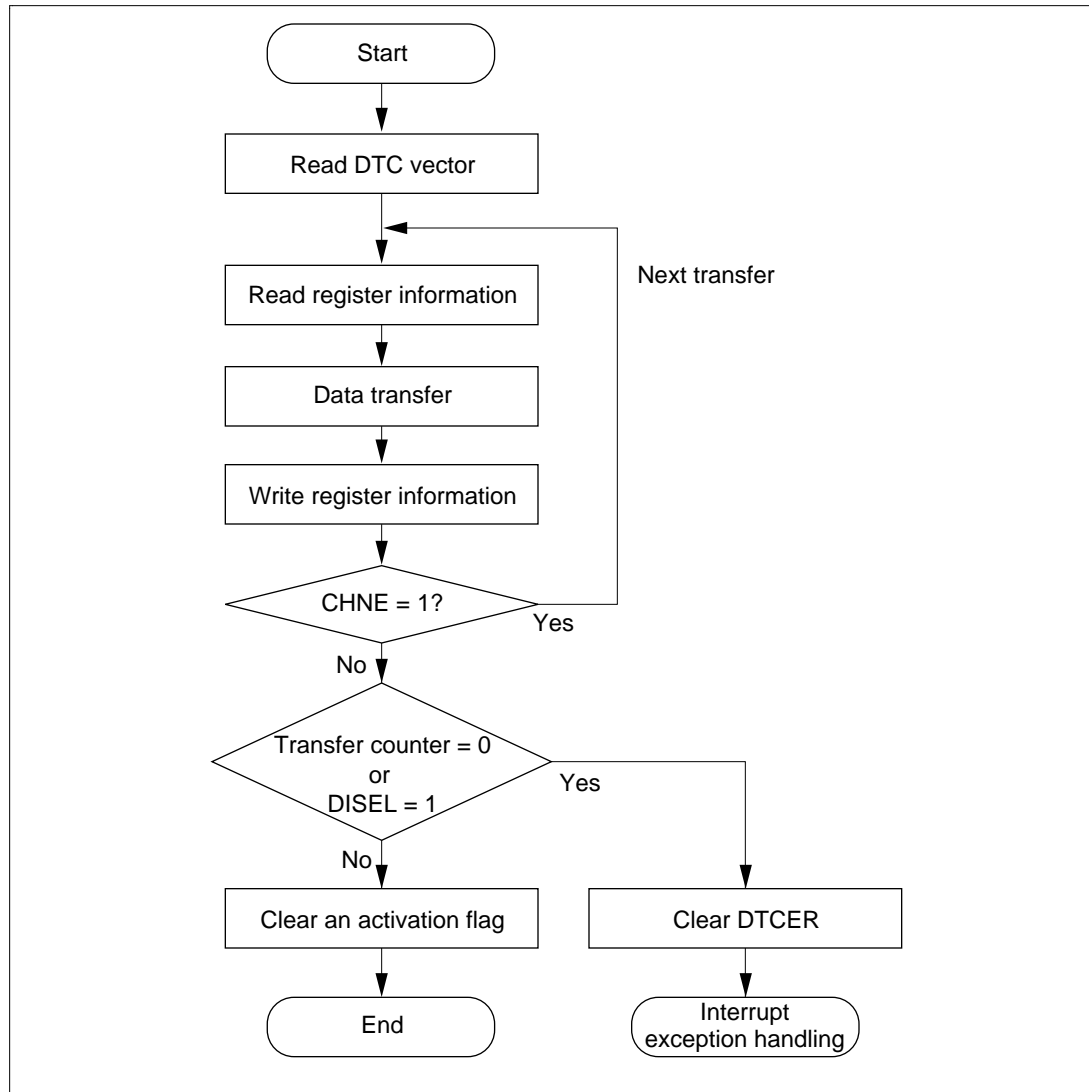


Figure 3.9 Flowchart of DTC Operation

DTC Activation Sources: The DTC operates when activated by an interrupt or by a write to the DTC vector register (DTVECR) by software. An interrupt request can be designated as a CPU interrupt source or a DTC activation source.

When an interrupt has been designated a DTC activation source, existing CPU mask level and interrupt controller priorities have no effect. If there is more than one activation source at the same time, the DTC operates in accordance with the default priorities.

Interrupt Sources and DTC Vector Address: The DTC vector address indicates the start address of the register information in memory. The MRA, SAR, MRB, DAR, CRA, and CRB registers are located in that order from the start address of the register information. Locate the register information in the on-chip RAM (addresses H'FFEB0 to H'FFEFBF).

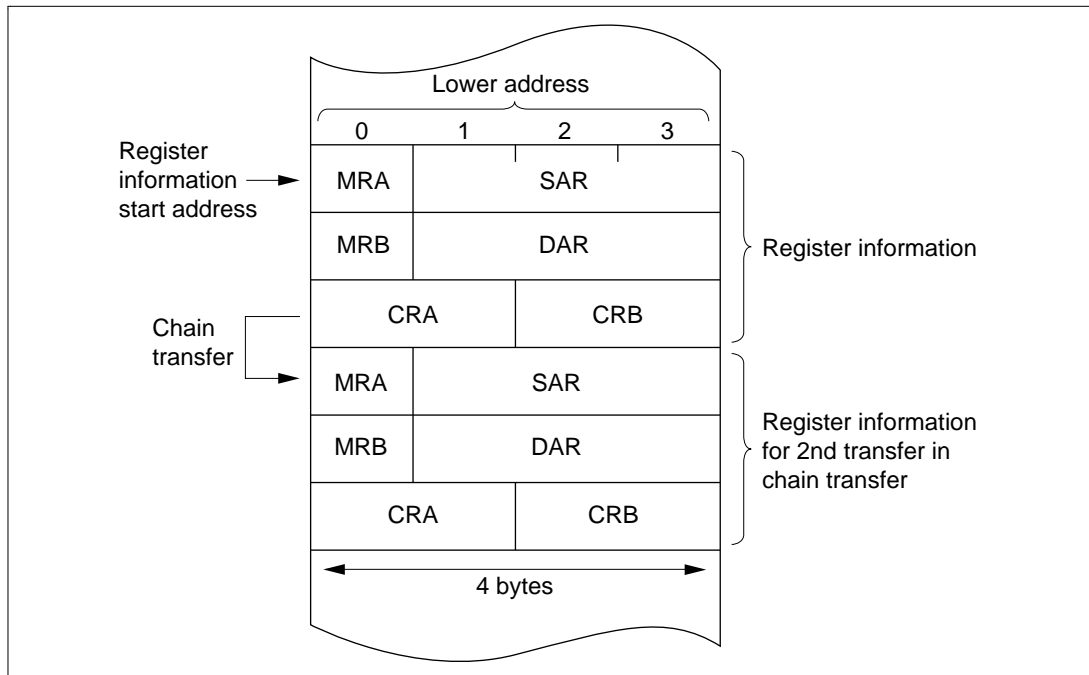


Figure 3.10 Location of DTC Register Information in Address Space

Table 3.2 Interrupt Sources, DTC Vector Addresses, and Corresponding DTCEs

Interrupt Source	Origin of Interrupt Source	Vector Number	Vector Address*	DTCE	Priority
Write to DTVECR	Software	DTVECR	H'0400+ (DTVECR [6:0]<<1)	—	High
IRQ0	External pin	16	H'0420	DTCEA7	
IRQ1		17	H'0422	DTCEA6	
IRQ2		18	H'0424	DTCEA5	
IRQ3		19	H'0426	DTCEA4	
IRQ4		20	H'0428	DTCEA3	
IRQ5		21	H'042A	DTCEA2	
IRQ6		22	H'042C	DTCEA1	
IRQ7		23	H'042E	DTCEA0	
ADI (A/D conversion end)	A/D	28	H'0438	DTCEB6	
TGI0A (GR0A compare-match/input capture)	TPU channel 0	32	H'0440	DTCEB5	
TGI0B (GR0B compare-match/input capture)		33	H'0442	DTCEB4	
TGI0C (GR0C compare-match/input capture)		34	H'0444	DTCEB3	
TGI0D (GR0D compare-match/input capture)		35	H'0446	DTCEB2	
TGI1A (GR1A compare-match/input capture)	TPU channel 1	40	H'0450	DTCEB1	
TGI1B (GR1B compare-match/input capture)		41	H'0452	DTCEB0	
TGI2A (GR2A compare-match/input capture)	TPU channel 2	44	H'0458	DTCEC7	
TGI2B (GR2B compare-match/input capture)		45	H'045A	DTCEC6	
TGI3A (GR3A compare-match/input capture)	TPU channel 3	48	H'0460	DTCEC5	
TGI3B (GR3B compare-match/input capture)		49	H'0462	DTCEC4	
TGI3C (GR3C compare-match/input capture)		50	H'0464	DTCEC3	

TGI3D (GR3D compare-match/input capture)		51	H'0466	DTCEC2	Low
Interrupt Source	Origin of Interrupt Source	Vector Number	Vector Address*	DTCE	Priority
TGI4A (GR4A compare-match/input capture)	TPU channel 4	56	H'0470	DTCEC1	High
TGI4B (GR4B compare-match/input capture)		57	H'0472	DTCEC0	
TGI5A (GR5A compare-match/input capture)	TPU channel 5	60	H'0478	DTCED5	
TGI5B (GR5B compare-match/input capture)		61	H'047A	DTCED4	
CMIA0	8-bit timer channel 0	64	H'0480	DTCED3	
CMIB0		65	H'0482	DTCED2	
CMIA1	8-bit timer channel 1	68	H'0488	DTCED1	
CMIB1		69	H'048A	DTCED0	
RXI0 (reception complete 0)	SCI channel 0	81	H'04A2	DTCEE3	
TXI0 (transmit data empty 0)		82	H'04A4	DTCEE2	
RXI1 (reception complete 1)	SCI channel 1	85	H'04AA	DTCEE1	
TXI1 (transmit data empty 1)		86	H'04AC	DTCEE0	
RXI2 (reception complete 2)	SCI channel 2	89	H'04B2	DTCEF7	
TXI2 (transmit data empty 2)		90	H'04B4	DTCEF6	
RXI3 (reception complete 3)	SCI channel 3	121	H'04F2	DTCEI7	
TXI3 (transmit data empty 3)		122	H'04F4	DTCEI6	Low

Note: * Lower 16 bits of the address.

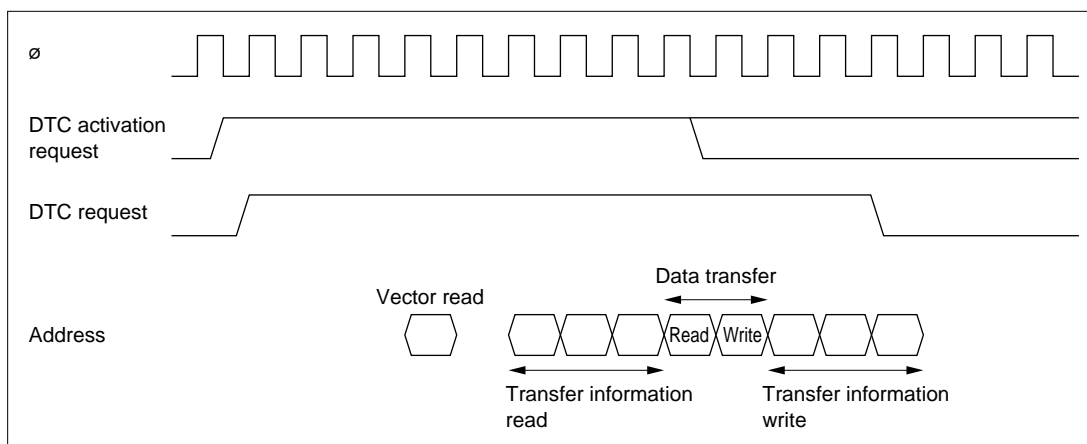


Figure 3.11 DTC Operation Timing (Example for Normal and Repeat Modes)

Table 3.3 Number of DTC Execution States

Mode	Vector Read I	Register Information Read/Write J	Data Read K	Data Write L	Internal Operations M
Normal	1	6	1	1	3
Repeat	1	6	1	1	3
Block transfer	1	6	N	N	3

N: Block size (initial setting of CRAH and CRAL)

Table 3.4 Number of States Required in Each Execution State

Access To			On- Chip RAM	On- Chip ROM	Internal I/O Registers		External Devices			
Bus width			32	16	8	16	8	8	16	16
Access states			1	1	2	2	2	3	2	3
Execution state	Vector read	S_I	—	1	—	—	4	6+2m	2	3+m
	Register information read/write	S_J	1	—	—	—	—	—	—	—
	Byte data read	S_K	1	1	2	2	2	3+m	2	3+m
	Word data read	S_K	1	1	4	2	4	6+2m	2	3+m
	Byte data write	S_L	1	1	2	2	2	3+m	2	3+m
	Word data write	S_L	1	1	4	2	4	6+2m	2	3+m
	Internal operation	S_M	1	1	1	1	1	1	1	1

The number of execution states is calculated from the formula below.

$$\text{Number of execution states} = I \cdot S_I + \Sigma (J \cdot S_J + K \cdot S_K + L \cdot S_L) + M \cdot S_M$$

Σ indicates the sum of all transfers activated by one activation event (the number in which the CHNE bit is set to 1, plus 1).

3.3.3 Transfer Modes

There are three DTC transfer modes—normal mode, repeat mode, and block transfer mode.

The 24-bit DTC source address register (SAR) designates the DTC transfer source address and the 24-bit destination address register (DAR) designates the transfer destination address. After each transfer, SAR and DAR are independently incremented, decremented, or left fixed.

Transfer Mode	Activation Source	Address Registers	
		Transfer Source	Transfer Destination
<ul style="list-style-type: none"> • Normal mode <ul style="list-style-type: none"> — One transfer request transfers one byte or one word — Memory addresses are incremented or decremented by 1 or 2 — Up to 65,536 transfers possible • Repeat mode <ul style="list-style-type: none"> — One transfer request transfers one byte or one word — Memory addresses are incremented or decremented by 1 or 2 — After the specified number of transfers (1 to 256), the initial state resumes and operation continues • Block transfer mode <ul style="list-style-type: none"> — One transfer request transfers a block of the specified size — Block size is from 1 to 256 bytes or words — Up to 65,536 transfers possible — A block area can be designated at either the source or destination 	<ul style="list-style-type: none"> • IRQ • TPU TGI • 8-bit timer CMI • SCI TXI or RXI • A/D converter ADI • Software 	24 bits	24 bits

Operation in Normal Mode: In normal mode, one operation transfers one byte or one word of data. From 1 to 65,536 transfers can be specified. When the specified number of transfers have ended, a CPU interrupt can be requested.

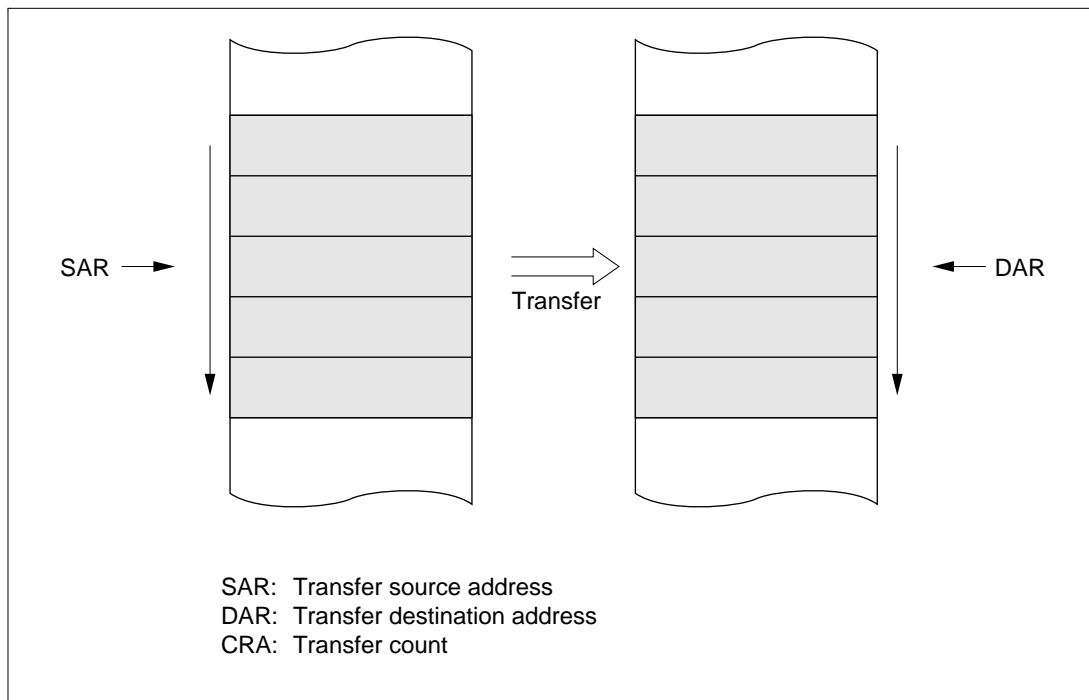


Figure 3.12 Operation in Normal Mode

Operation in Repeat Mode: In repeat mode, one operation transfers one byte or one word of data. From 1 to 256 transfers can be specified. When the specified number of transfers have ended, the initial settings are restored and transfer is repeated. A CPU interrupt is not requested.

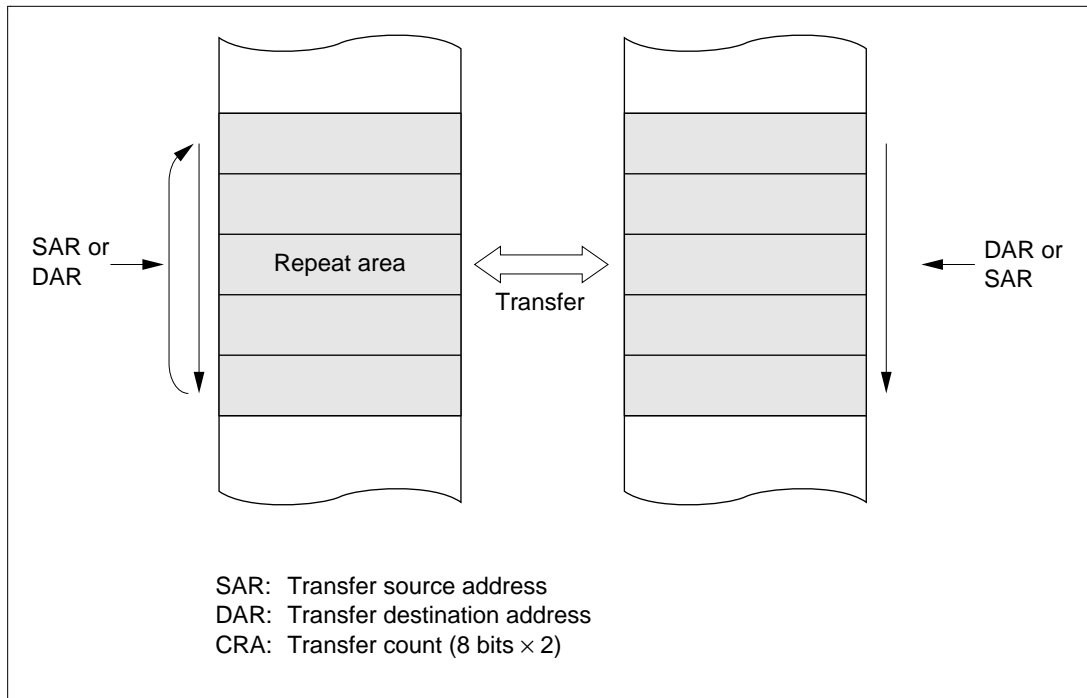


Figure 3.13 Operation in Repeat Mode

Operation in Block Transfer Mode: In block transfer mode, one operation transfers one block of data. Either the transfer source or the transfer destination is specified as a block area. The block size is 1 to 256. When the transfer of one block ends, the initial setting of the address register specified in the block area is restored. The other address register is incremented, decremented, or left fixed.

From 1 to 65,536 transfers can be specified. When the specified number of transfers have ended, a CPU interrupt can be requested.

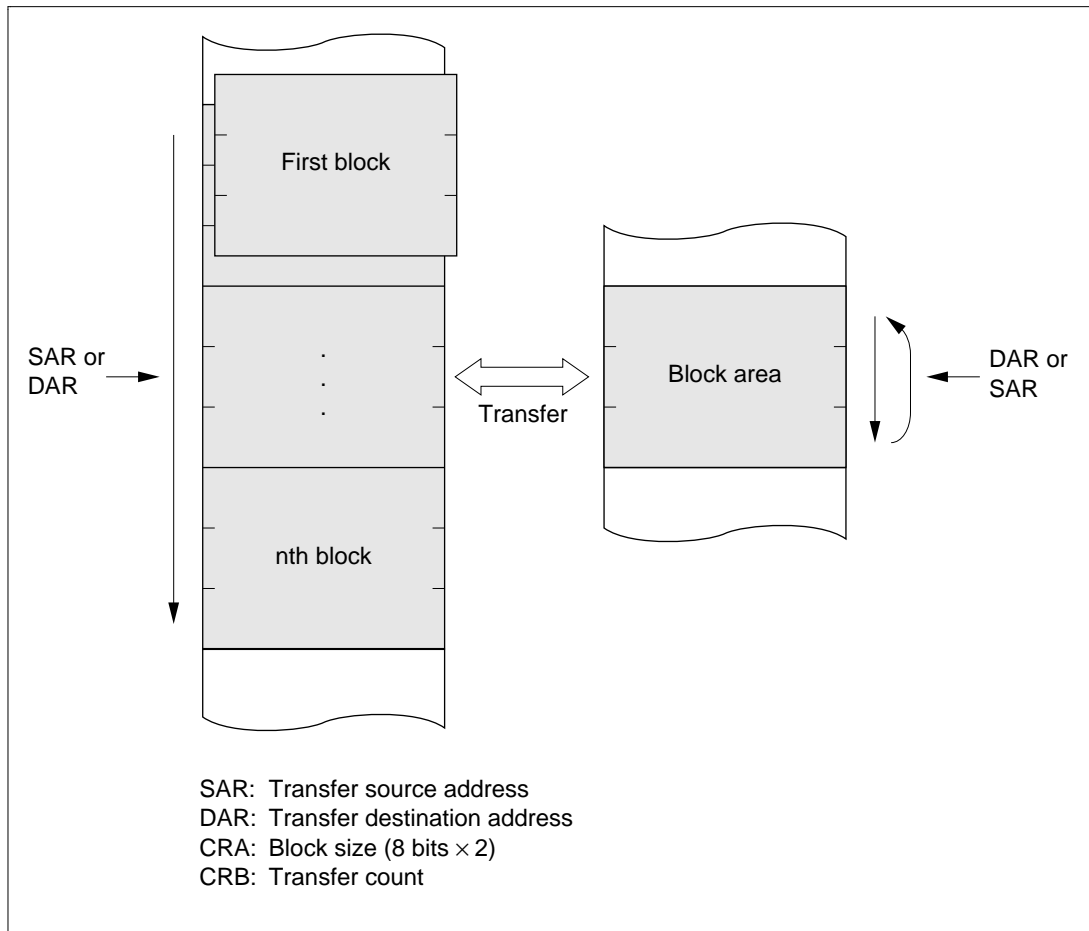


Figure 3.14 Operation in Block Transfer Mode

3.4 16-Bit Timer Pulse Unit (TPU)

These series have an on-chip 16-bit timer pulse unit (six channels in the H8S/2227 Series, three channels in the H8S/2227 Series). The TPU can provide up to 16 kinds of pulse input/output.

The TPU can perform PWM output, pulse width measurement, and two-phase encoder processing, and can activate the data transfer controller (DTC) . It can also generate an A/D converter start trigger.

3.4.1 Features

- On-chip channels
 - H8S/2237 Series: 0, 1, 2, 3, 4, 5
 - H8S/2227 Series: 0, 1, 2
- Maximum 16 pulse input/outputs
 - A total of 16 timer general registers (TGRs) are provided (four each for channels 0 and 3, and two each for channels 1, 2, 4, and 5), each of which can be set independently as an output compare/input capture register
- Selection of eight counter input clocks for each channel
 - Internal clocks: \emptyset , $\emptyset/4$, $\emptyset/16$, $\emptyset/64$, $\emptyset/256$, $\emptyset/1024$, $\emptyset/4096$
 - External clocks: TCLKA, TCLKB, TCLKC, TCLKD
- The following operations can be set for each channel:
 - Waveform output at compare-match: Selection of 0, 1, or toggle output
 - Input capture function: Selection of rising edge, falling edge, or both edge detection
 - Counter clear operation: Counter clearing possible by compare-match or input capture
 - Synchronous operation:
 - Multiple timer counters (TCNT) can be written to simultaneously
 - Simultaneous clearing by compare-match and input capture possible
 - Simultaneous input/output possible for each register by counter synchronous operation
 - PWM mode:
 - Any PWM output duty can be set
 - Maximum 15-phase PWM output possible by combination with synchronous operation
- Buffer operation settable for channels 0 and 3
 - Input capture register double-buffering possible
 - Automatic rewriting of output compare register possible
- Phase counting mode settable independently for each of channels 1, 2, 4, and 5
 - Two-phase encoder pulse up/down-count possible

- Cascaded operation
 - Channel 2 (channel 5) input clock operates as 32-bit counter by setting channel 1 (channel 4) overflow/underflow
- Fast access via internal 16-bit bus
 - Fast access is possible via a 16-bit bus interface
- 26 interrupt sources
 - For channels 0 and 3, four compare-match/input capture dual-function interrupts and one overflow interrupt can be requested independently
 - For channels 1, 2, 4, and 5, two compare-match/input capture dual-function interrupts, one overflow interrupt, and one underflow interrupt can be requested independently
- Automatic transfer of register data
 - Block transfer, one-word transfer, and one-byte transfer possible by data transfer controller (DTC) activation
- A/D converter conversion start trigger can be generated
 - Channel 0 to 5 compare-match A/input capture A signals can be used as an A/D converter conversion start trigger
- Module stop mode can be set
 - As the initial setting, TPU operation is halted. Register access is enabled by exiting module stop mode.

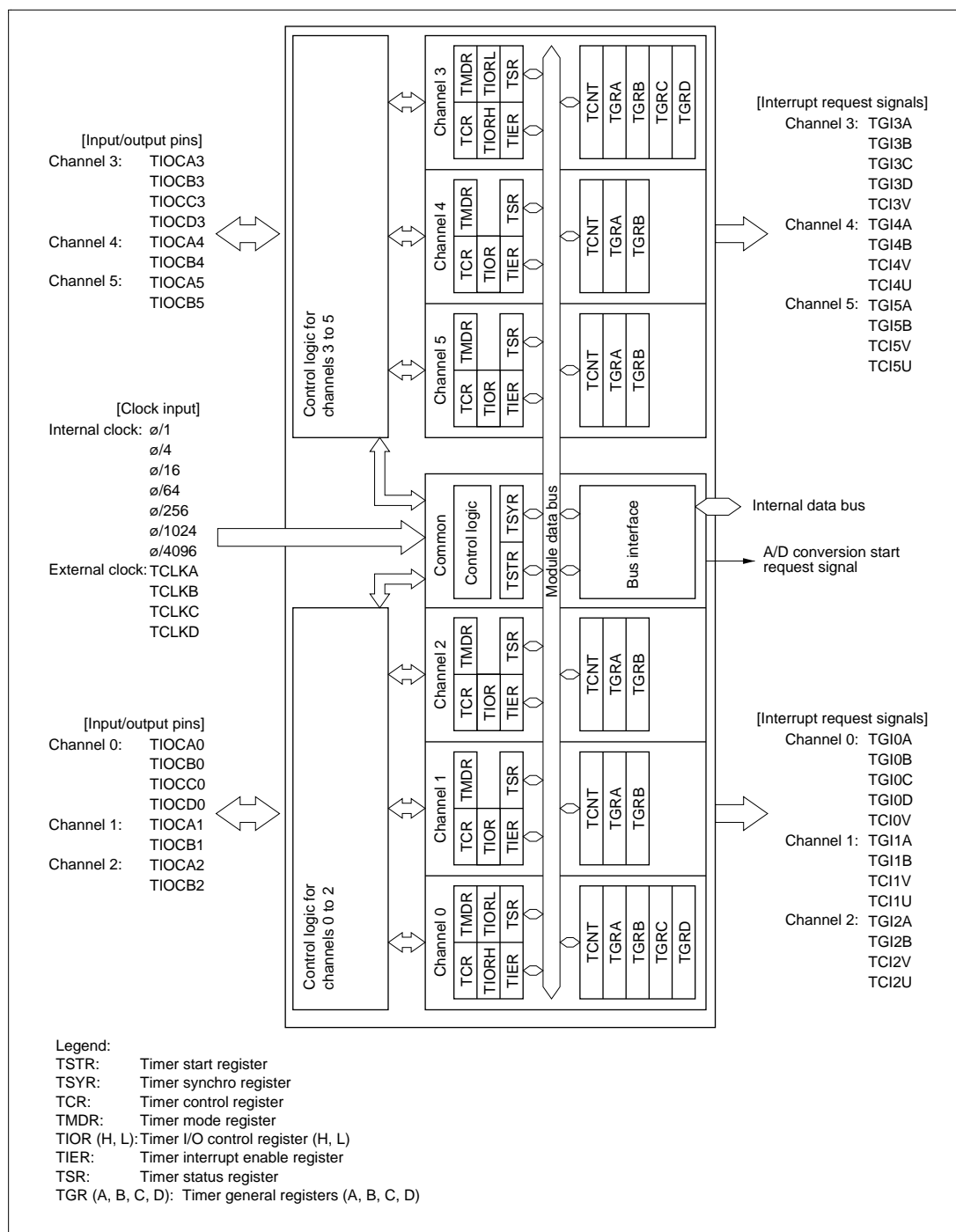


Figure 3.15 H8S/2237 Series TPU Block Diagram

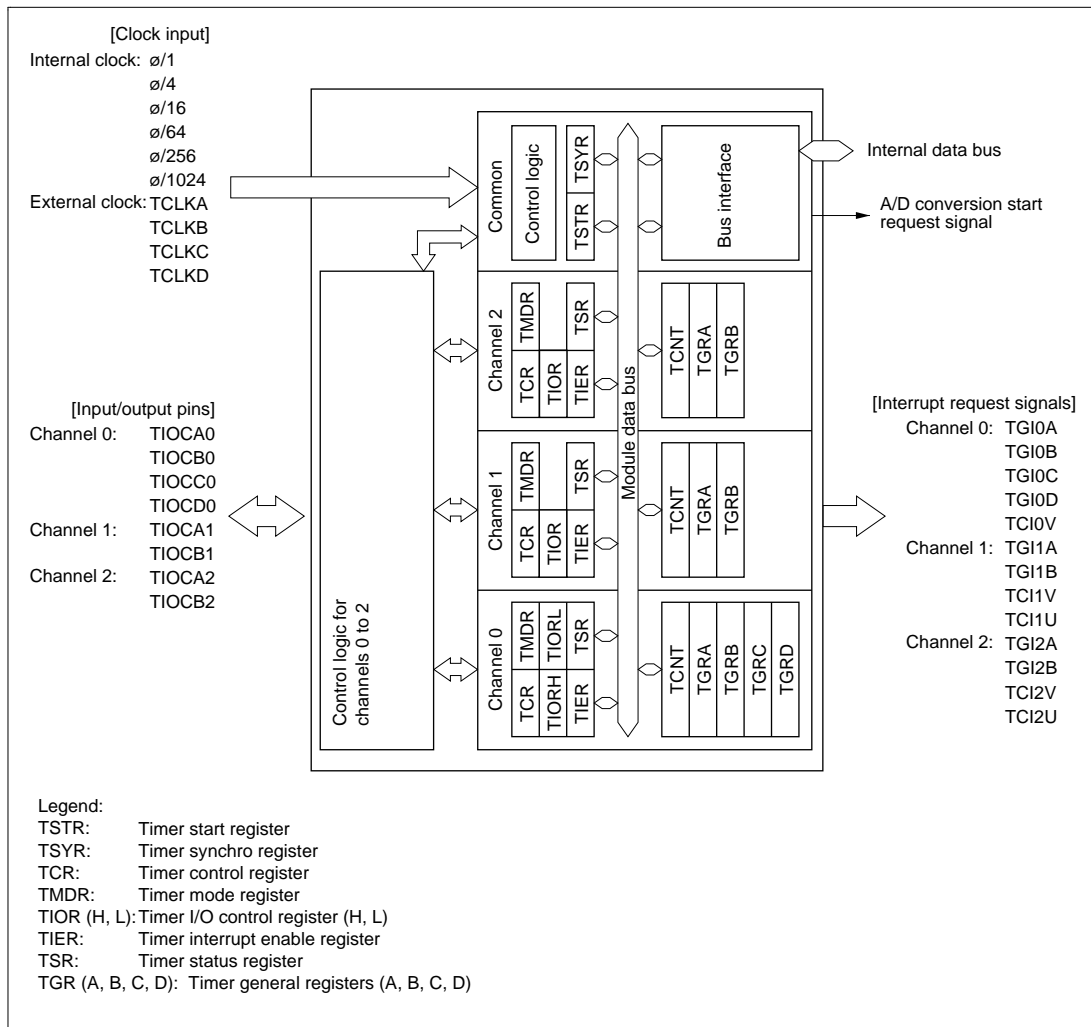


Figure 3.16 H8S/2227 Series TPU Block Diagram

3.4.2 Interrupt Sources and Data Transfer Controller (DTC) Activation

Table 3.3 TPU Interrupts

Channel	Interrupt Source	Description	DTC Activation	Priority
0	TGI0A	TGR0A input capture/compare-match	Possible	High
	TGI0B	TGR0B input capture/compare-match	Possible	
	TGI0C	TGR0C input capture/compare-match	Possible	
	TGI0D	TGR0D input capture/compare-match	Possible	
	TCI0V	TCNT0 overflow	Not possible	
1	TGI1A	TGR1A input capture/compare-match	Possible	
	TGI1B	TGR1B input capture/compare-match	Possible	
	TCI1V	TCNT1 overflow	Not possible	
	TCI1U	TCNT1 underflow	Not possible	
2	TGI2A	TGR2A input capture/compare-match	Possible	
	TGI2B	TGR2B input capture/compare-match	Possible	
	TCI2V	TCNT2 overflow	Not possible	
	TCI2U	TCNT2 underflow	Not possible	
3*	TGI3A	TGR3A input capture/compare-match	Possible	
	TGI3B	TGR3B input capture/compare-match	Possible	
	TGI3C	TGR3C input capture/compare-match	Possible	
	TGI3D	TGR3D input capture/compare-match	Possible	
	TCI3V	TCNT3 overflow	Not possible	
4*	TGI4A	TGR4A input capture/compare-match	Possible	
	TGI4B	TGR4B input capture/compare-match	Possible	
	TCI4V	TCNT4 overflow	Not possible	
	TCI4U	TCNT4 underflow	Not possible	
5*	TGI5A	TGR5A input capture/compare-match	Possible	
	TGI5B	TGR5B input capture/compare-match	Possible	
	TCI5V	TCNT5 overflow	Not possible	
	TCI5U	TCNT5 underflow	Not possible	Low

Notes: This table shows the initial state immediately after a reset. The relative channel priorities can be changed by the interrupt controller.

* Only applies to the H8S/2237 Series.

3.4.3 Operation

Normal Operation: Each channel has a TCNT and TGR register. TCNT performs up-counting, and is also capable of free-running operation, synchronous counting, and external event counting. Each TGR can be used as an input capture register or output compare register.

Buffer Operation

- When TGR is an output compare register
When a compare-match occurs, the value in the buffer register for the relevant channel is transferred to TGR.
- When TGR is an input capture register
When input capture occurs, the value in TCNT is transfer to TGR and the value previously held in TGR is transferred to the buffer register.

Waveform Output by Compare-Match

0, 1, or toggle output can be selected.

Example of 0 Output/1 Output Operation: In this example, TCNT has been designated as a free-running counter, and settings have been made so that 0 is output by compare-match A, and 1 is output by compare-match B.

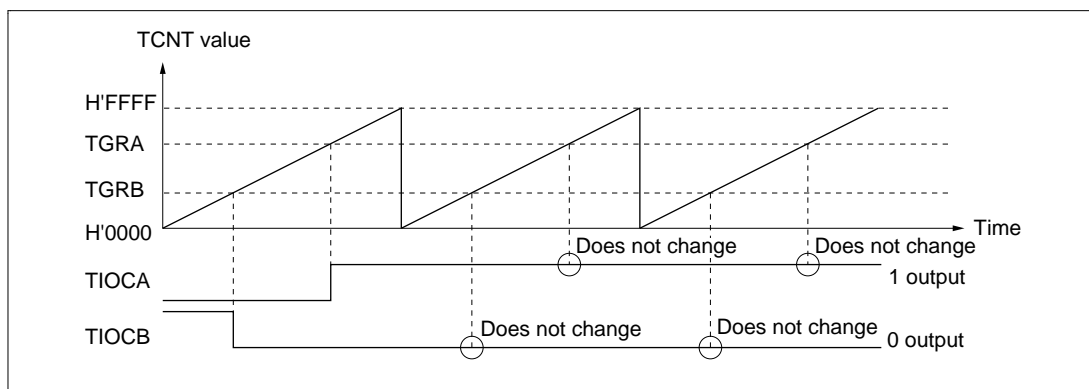


Figure 3.17 Example of 0 Output/1 Output Operation

Example of Toggle Output: In this example, settings have been made so that TCNT counter clearing is performed by compare-match B, and output is toggled by both by compare-match A and compare-match B.

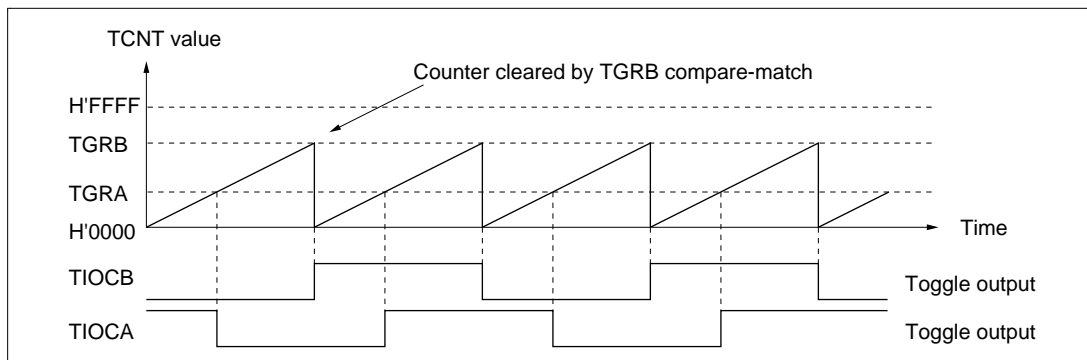


Figure 3.18 Example of Toggle Output Operation

3.4.4 PWM Modes

In PWM mode, PWM waveforms are output from the output pins. There are two PWM modes—PWM mode 1 with a maximum of 8-phase pulse output, and PWM mode 2 with a maximum of 15-phase pulse output.

PWM Mode 1: PWM output is generated by pairing TGRA with TGRB and TGRC with TGRD.

In PWM mode 1, a maximum 8-phase PWM output is possible.

- Example of operation in PWM mode 1

In this example, TGRA compare-match is set as the TCNT clearing source, 0 is set for the TGRA initial output value and output value, and 1 is set as the TGRB output value. In this case, the value set in TGRA is the cycle, and the value set in TGRB is the duty.

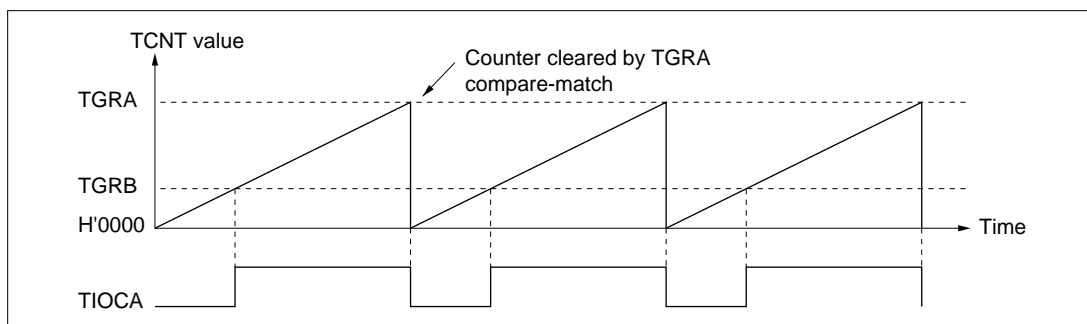


Figure 3.19 Operation in PWM Mode 1

PWM Mode 2: PWM output is generated using one TGR register as the cycle register and the others as duty registers. In PWM mode 2, a maximum 15-phase PWM output is possible by combined use with synchronous operation.

- Example of operation in PWM mode 2

In this example, synchronous operation is designated for channels 0 and 1, TGR1B compare-match is set as the TCNT clearing source, and 0 is set for the initial output value and 1 for the output value of the other TGR registers, to output a 5-phase PWM waveform. In this case, the value set in TGR1B is the cycle, and the value set in the other TGR registers is the duty.

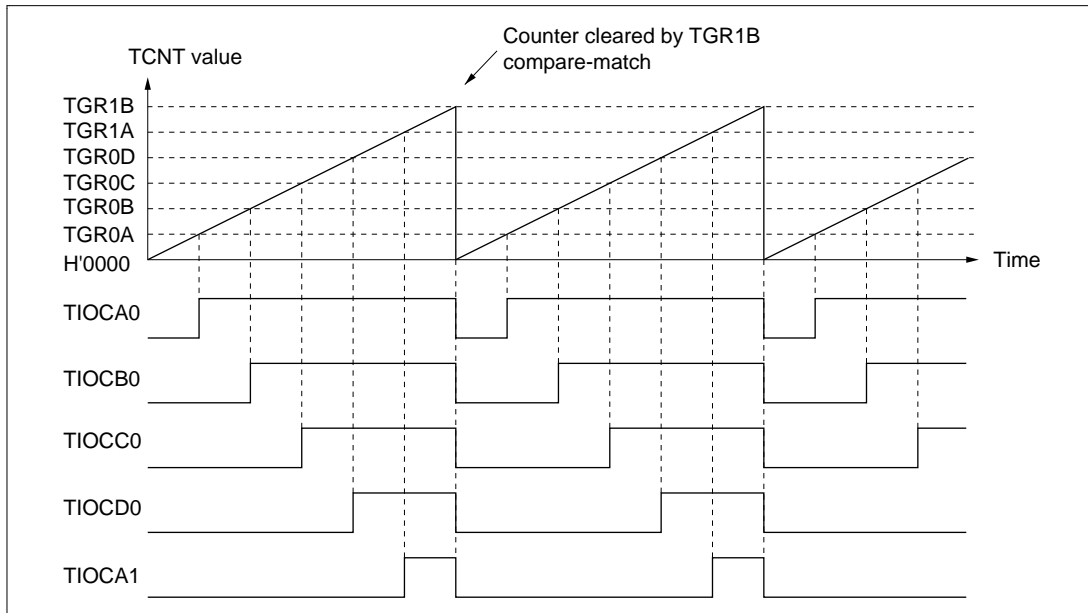


Figure 3.20 Operation in PWM Mode 2

3.4.5 Input Capture Operation

The TCNT value can be transferred to TGR on detection of the TIOC pin input edge.

Rising edge, falling edge, or both edges can be selected as the input edge.

- Example of input capture operation

In this example both rising and falling edges have been selected as the TIOCA pin input edge, falling edge has been selected as the TIOCB pin input edge, and counter clearing by TGRB input capture has been designated for TCNT.

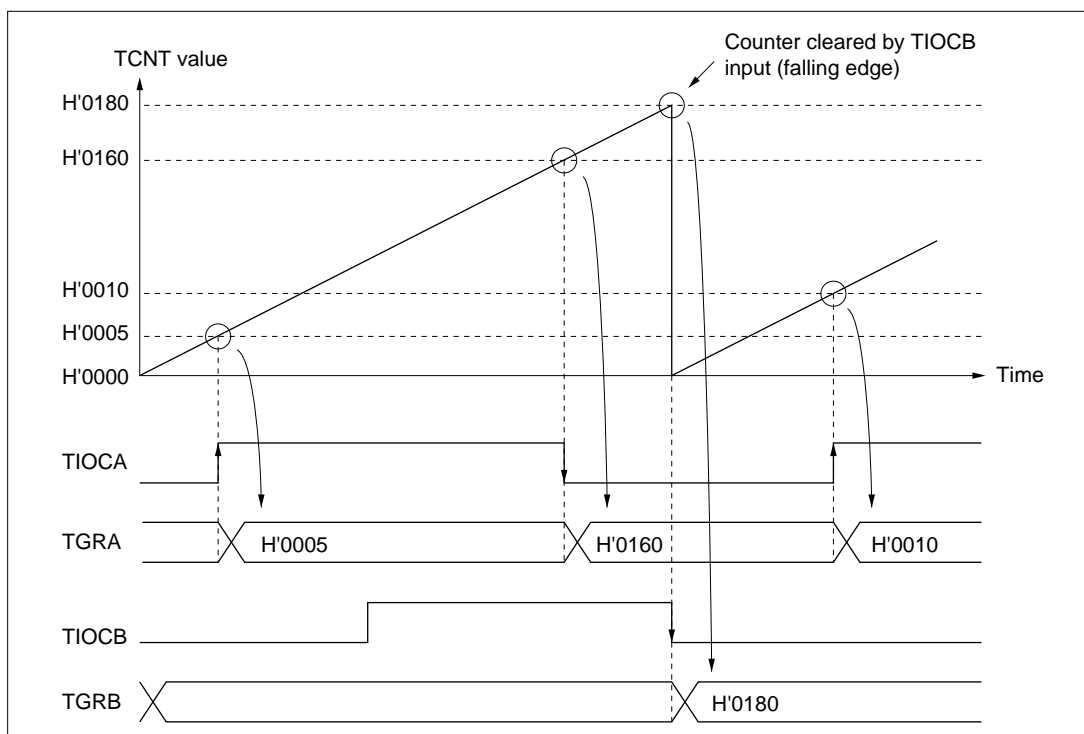


Figure 3.21 Input Capture Operation

3.4.6 Phase Counting Mode

In phase counting mode, the phase difference between two external clock inputs is detected and TCNT operates as an up/down-counter. There are four modes (phase counting modes 1 to 4) with different setting conditions. These modes can be set for channels 1, 2, 4, and 5.

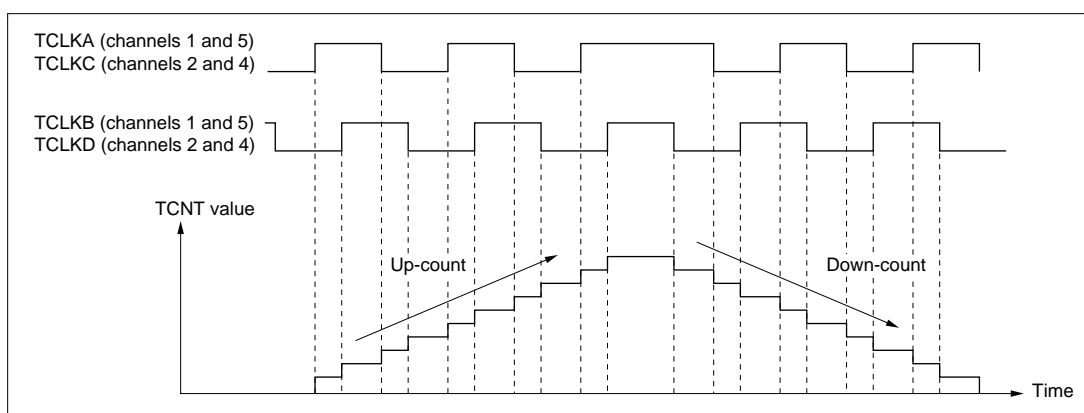










Figure 3.22 Example of Operation in Phase Counting Mode 1

- Up/Down-Count Conditions in Phase Counting Mode

TCLKA (Channels 1 and 5)	TCLKB (Channels 1 and 5)	Phase Counting Mode			
TCLKC (Channels 2 and 4)	TCLKD (Channels 2 and 4)	1	2	3	4
High level		Up-count	—	—	Up-count
Low level					
	Low level				—
	High level		Up-count	Up-count	
High level		Down-count	—	Down-count	Down-count
Low level				—	
	High level				—
	Low level		Down-count		

Legend

 : Rising edge

 : Falling edge

—: Don't care

3.4.7 Buffer Operation

Buffer operation, provided for channels 0 and 3, enables TGRC and TGRD to be used as buffer registers.

- Example of buffer operation (1) (When TGR is an output compare register)
In this example, PWM mode 1 has been designated for channel 0, and buffer operation has been designated for TGRA and TGRC. The settings used are TCNT clearing by a compare-match B, 1 output at compare-match A, and 0 output at compare-match B. When a compare-match A occurs, the output is changed and the value in buffer register TGRC is simultaneously transferred to timer general register TGRA.

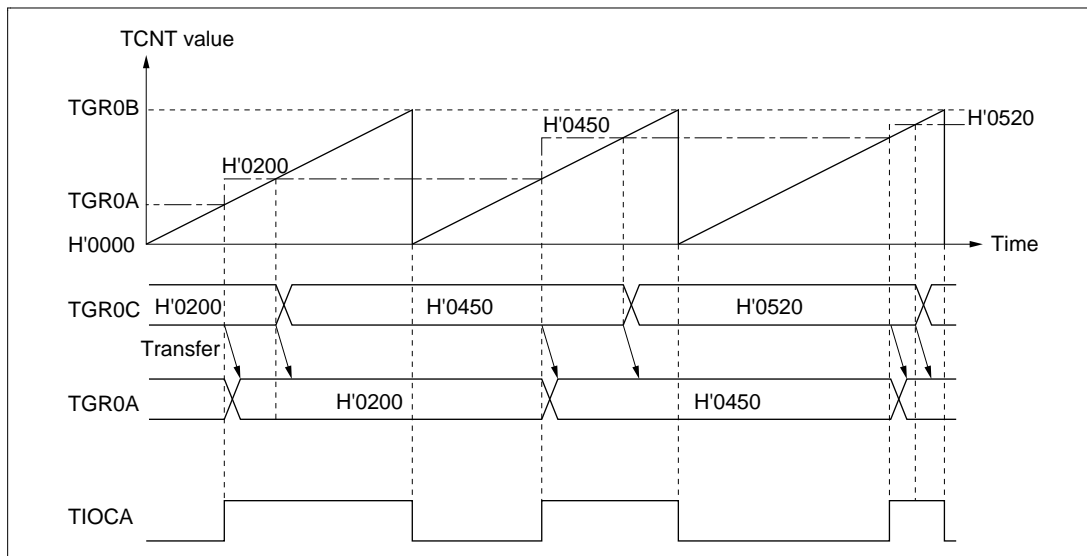
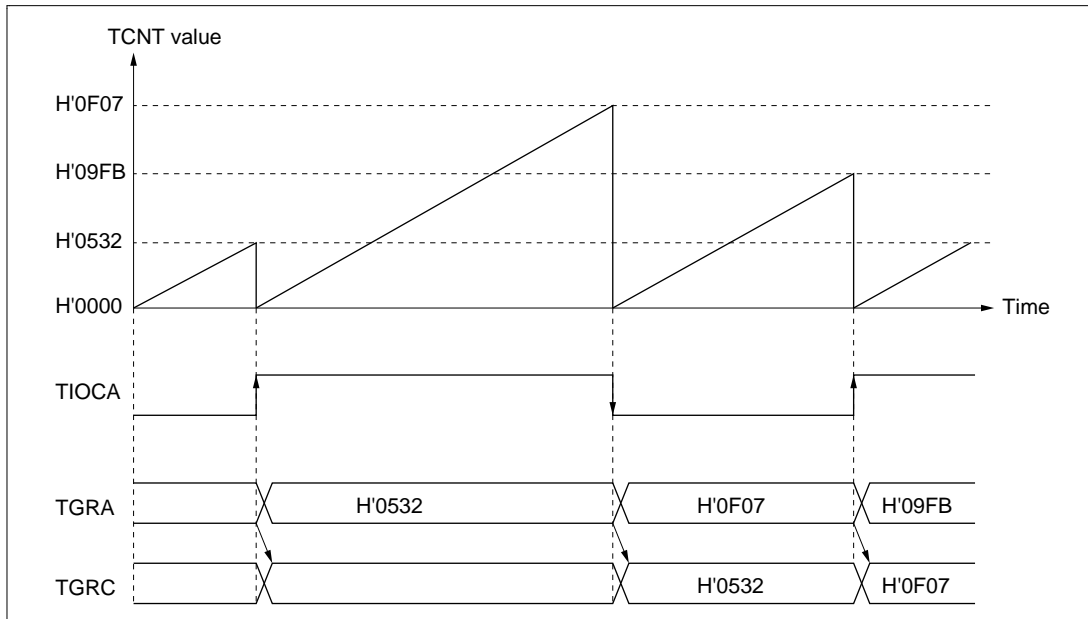


Figure 3.23 Example of Buffer Operation (1)
(When TGR Is an Output Compare Register)

- Example of buffer operation (2) (When TGR is an input capture register)

In this example, TGRA has been designated as an input capture register, and buffer operation has been designated for TGRA and TGRC. Counter clearing by TGRA input capture has been set for TCNT, and detection of both rising and falling edges has been selected for the TIOCA pin. When the TCNT value is stored in TGRA upon occurrence of input capture A, the value previously stored in TGRA is simultaneously transferred to TGRC.



**Figure 3.24 Example of Buffer Operation (2)
(When TGR Is an Input Capture Register)**

Cascading

In cascaded operation, two 16-bit counters for different channels are used together as a 32-bit counter. Channels 1 and 2, and channels 4 and 5, can be cascaded.

- Example of cascaded operation

In this example, counting upon TCNT2 overflow/underflow has been set for TCNT1, TGR1A and TGR2A have been designated as input capture registers, and TIOC pin rising edge detection has been selected. When a rising edge is input to the TIOCA1 and TIOCA2 pins simultaneously, the upper 16 bits of the 32-bit data are transferred to TGR1A, and the lower 16 bits to TGR2A.

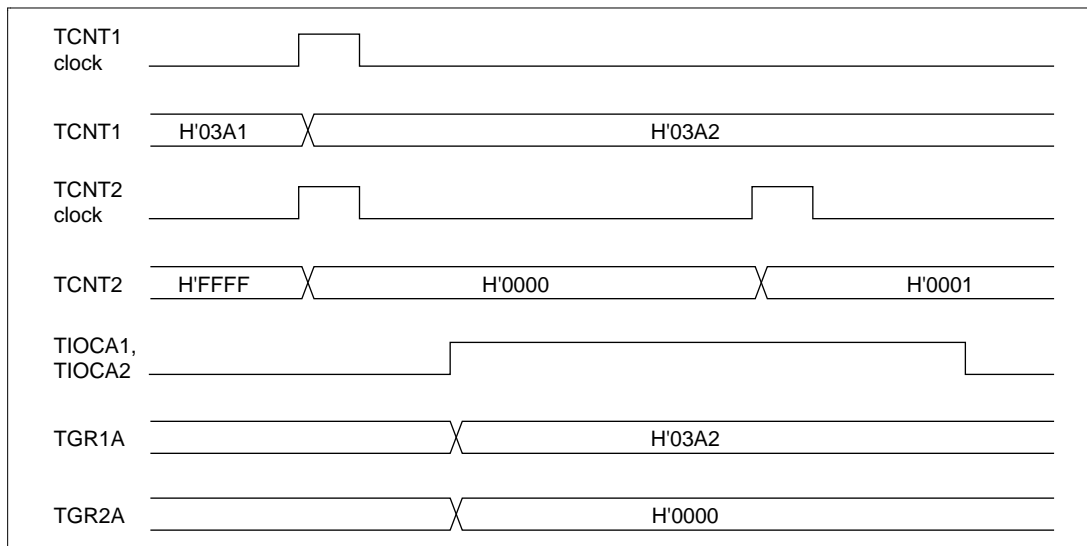


Figure 3.25 Example of Cascaded Operation (32-Bit Input Capture Operation)

Synchronous Operation

When synchronous operation is designated for a channel, TCNT for that channel performs synchronous presetting and clearing. That is, when TCNT for a channel designated for synchronous operation is rewritten, the TCNT counters for the other channels are also rewritten at the same time. When any clearing condition occurs, the TCNT counters for the other channels are also cleared simultaneously.

3.5 8-Bit Timer (TMR)

These series includes an 8-bit timer with two channels based on an 8-bit counter. The 8-bit timer can be used for a variety of applications as a multifunctional timer, including pulse output with an arbitrary duty cycle.

3.5.1 Features

- Selection of four input clock sources
 - The clock source can be selected from three internal clock signals ($\phi/8$, $\phi/64$, or $\phi/8192$) or an external clock (external event counting is possible).
- Counter clearing specification
 - The counters can be cleared on compare-match A or B, or by an external reset signal.
- Timer output controlled by combination of two compare-match signals
 - The timer output signal in each channel is controlled by two independent compare-match signals, enabling the timer to generate pulse output or PWM output with an arbitrary duty cycle.
- Provision for cascading of two channels
 - Operation as a 16-bit timer is possible, using channel 0 for the upper 8 bits and channel 1 for the lower 8 bits (16-bit count mode).
 - Channel 1 can be used to count channel 0 compare matches (compare match count mode).
- Three interrupt sources for each channel
 - There are two compare-match sources and one overflow source, capable of independent requests.
- A/D converter conversion start trigger can be generated
 - Channel 0 compare-match A signal can be used as an A/D converter conversion start trigger.
- Module stop mode can be set
 - As the initial setting, 8-bit timer operation is halted. Register access is enabled by exiting module stop mode.

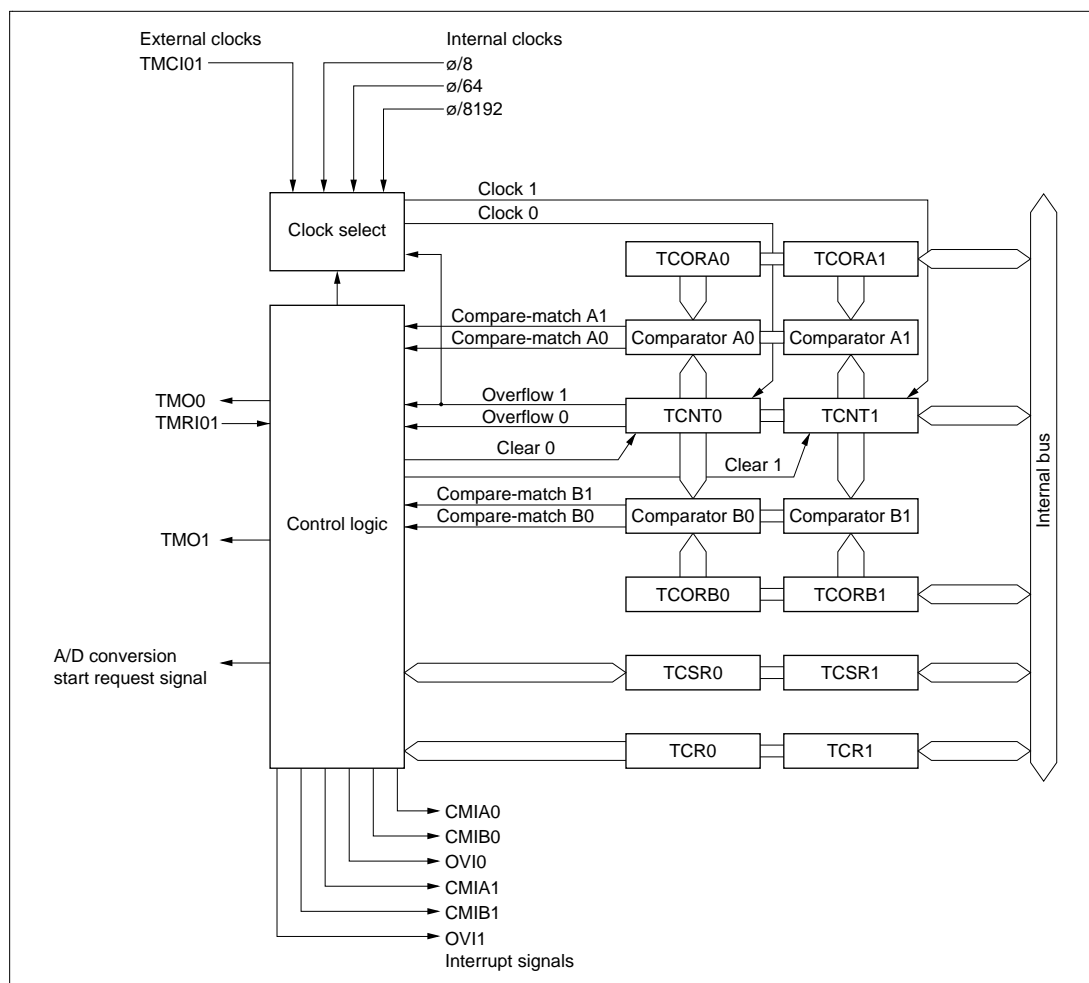


Figure 3.26 8-Bit Timer Block Diagram

3.5.2 Interrupt Source and Data Transfer Controller (DTC) Activation

Table 3.5 8-Bit Timer Interrupts

Channel	Interrupt Source	Description	DTC Activation	Priority
0	CMIA0	Interrupt by CMFA	Possible	High
	CMIB0	Interrupt by CMFB	Possible	
	OVI0	Interrupt by OVF	Not possible	
1	CMIA1	Interrupt by CMFA	Possible	Low
	CMIB1	Interrupt by CMFB	Possible	
	OVI1	Interrupt by OVF	Not possible	

Note: This table shows the initial state immediately after a reset. The relative channel priorities can be changed by the interrupt controller.

3.5.3 Example of Pulse Output

TCR is used to set counter clearing by a TCORA compare-match. The cycle is set in TCORA, and the duty in TCORB. The pulses shown below can be output continuously without software intervention.

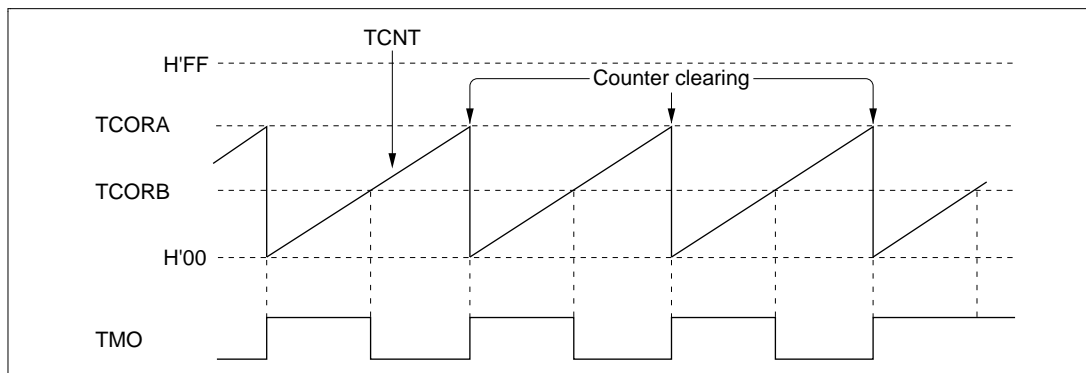


Figure 3.27 Example of Pulse Output

3.6 Watchdog Timer (WDT)

These series have an on-chip watchdog timer with two channels (WDT0, WDT1) for monitoring system operation.

When this watchdog function is not needed, the WDT can be used as an interval timer.

When the subclock is selected as the input clock, the WDT can be used as a real-time clock timer.

3.6.1 Features

- Choice of 8 (WDT0) or 16 (WDT1) counter input clocks
 - Maximum WDT interval: $\text{system clock period} \times 131072 \times 256$
 - Subclock can be selected for the WDT1 input counter
 - Maximum interval when the subclock is selected: $\text{subclock period} \times 256 \times 256$
- Can be used as an interval timer
- Generation of internal reset or internal interrupt in watch timer mode
 - WDT0: Choice of whether or not the chip is internally reset when the counter overflows
 - WDT1: Choice of internal reset or NMI interrupt generation when the counter overflows
- Interrupt generation in interval timer mode
 - When the counter overflows, the WDT generates an interval timer interrupt.

3.6.2 Block Diagram

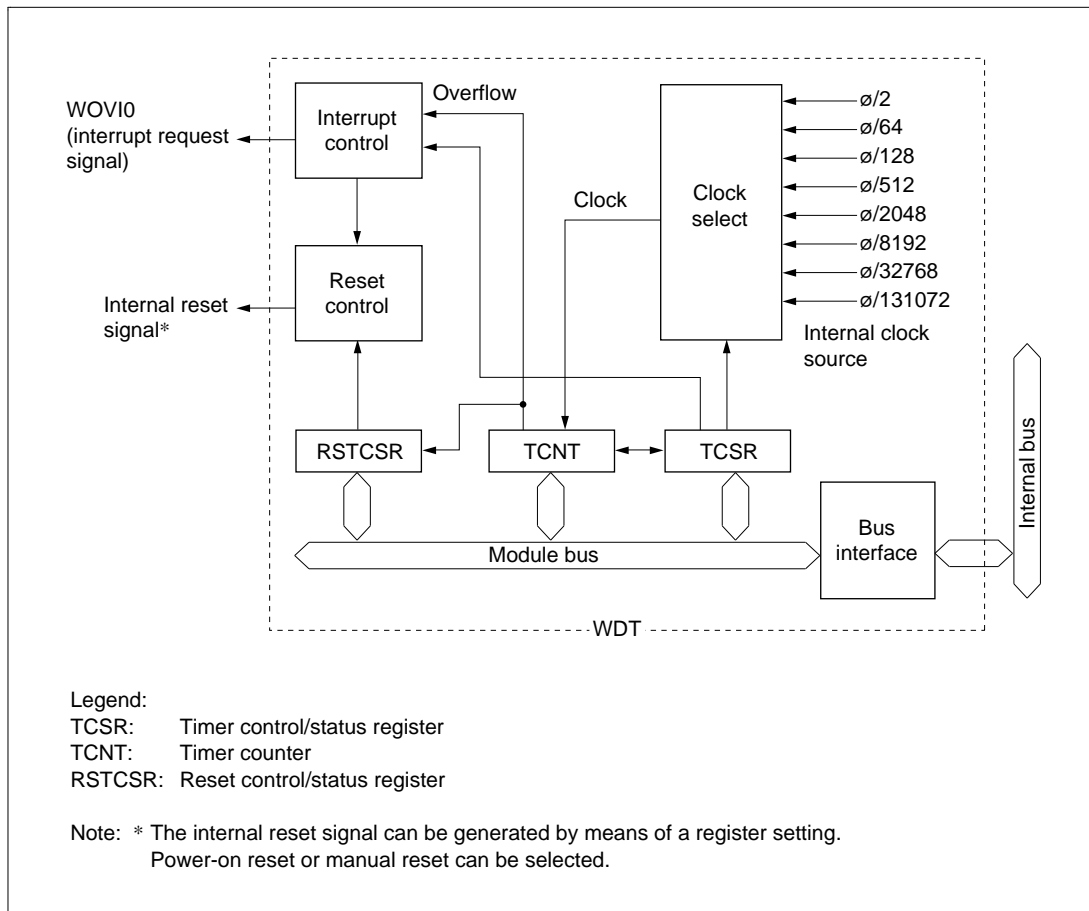


Figure 3.28 Block Diagram of WDT0

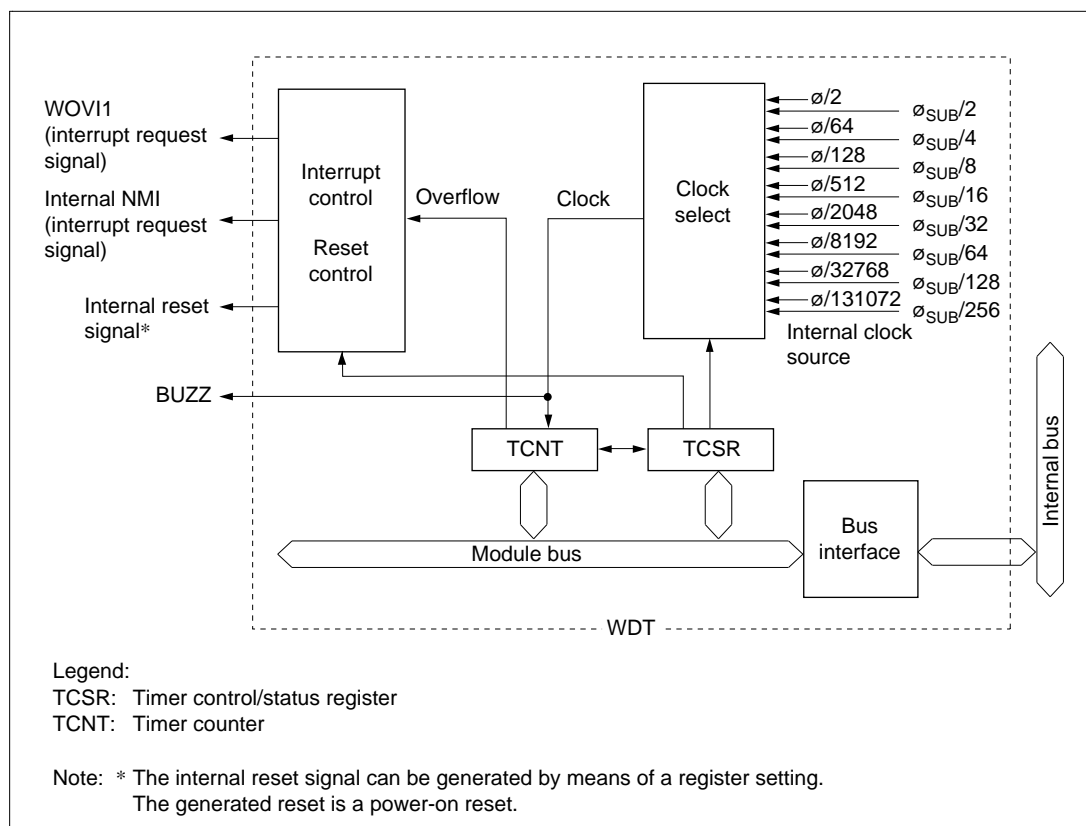


Figure 3.29 Block Diagram of WDT1

Watchdog Timer Operation: The example below shows this module used as a watchdog timer. The timer counter (TCNT) starts counting up using the specified clock.

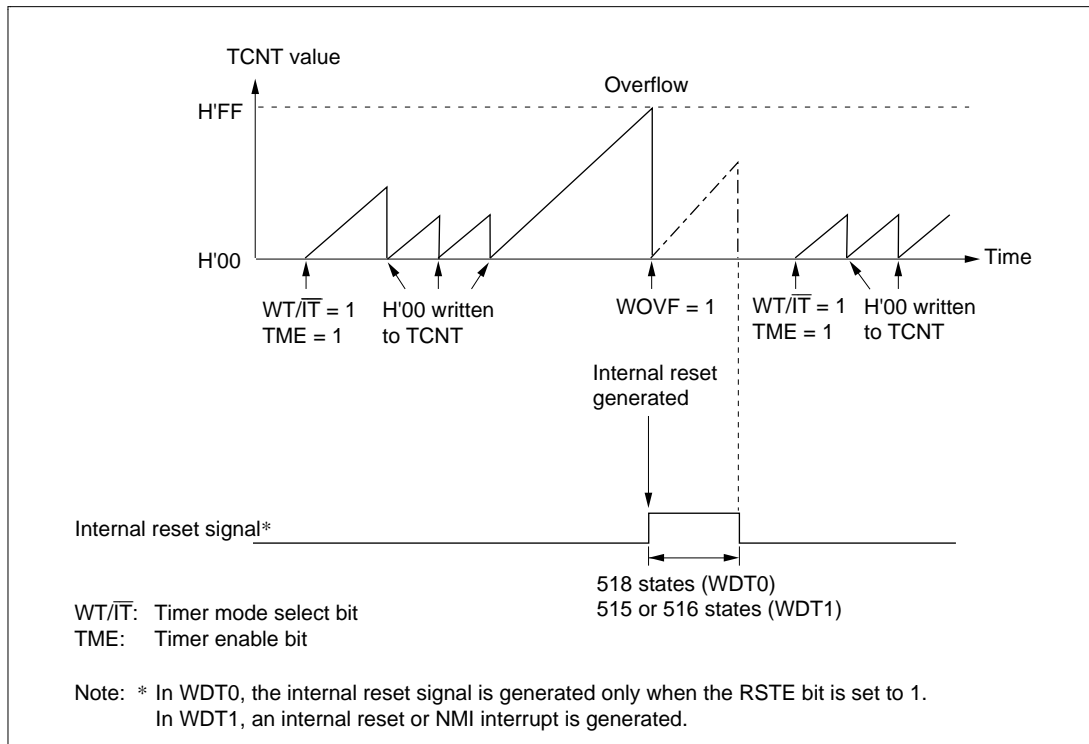


Figure 3.30 Operation in Watchdog Timer Mode

Interval Timer Operation: An example of the use of the WDT as an interval timer is shown here. The timer counter (TCNT) starts counting up on the specified clock, and an interval timer interrupt (WOVI) occurs each time TCNT overflows. This function can be used to generate interrupt requests at regular intervals.

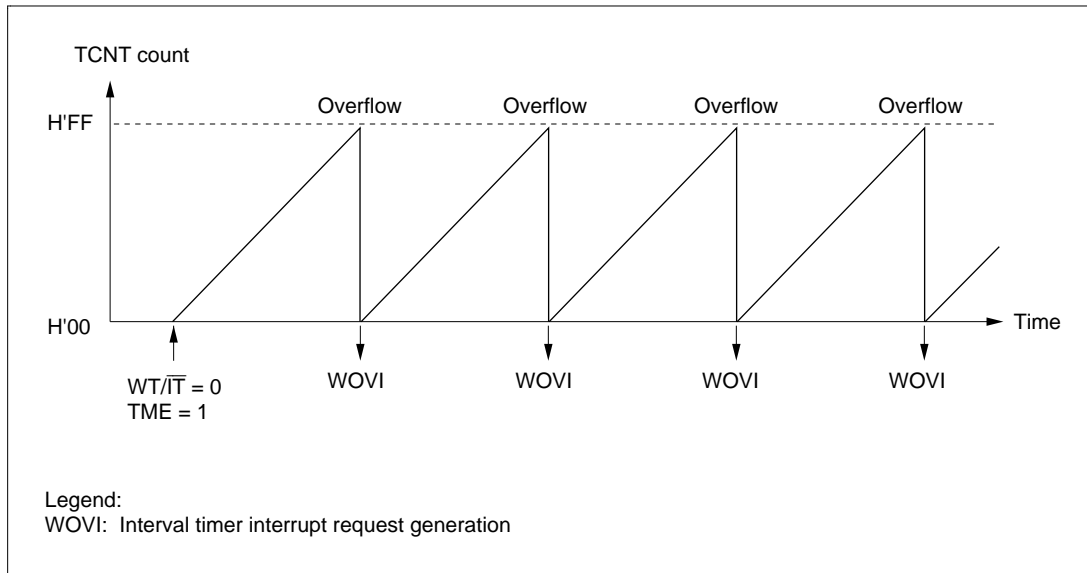


Figure 3.31 Interval Timer Operation

3.7 Serial Communication Interface (SCI)

These series are equipped with a 4- or 3-channel serial communication interface (SCI). The SCI can handle both asynchronous and clocked synchronous serial communication. A function is also provided for serial communication between processors (multiprocessor communication function).

3.7.1 Features

- On-chip channels
 - H8S/2237 Series: 0, 1, 2, 3
 - H8S/2227 Series: 0, 1, 3
- Choice of asynchronous or clocked synchronous serial communication mode
- Full-duplex communication capability
- Data register double-buffering enables continuous transmission/reception
- On-chip dedicated baud rate generator allows any bit rate to be selected
- Selection of internal clock from baud rate generator or external clock input (SCK pin) as serial clock source
- Detection of three receive errors
 - Overrun errors, framing errors, and parity errors can be detected
- Break detection
- Four interrupt sources
 - Four interrupt sources—transmit data empty, transmission end, receive data full, and receive error—that can issue requests independently
 - The transmit data empty interrupt and receive data full interrupt can activate the data transfer controller (DTC) to execute data transfer
- Built-in multiprocessor communication function
- Selection of LSB-first or MSB-first transfer
 - This choice can be made regardless of the communication mode (with the exception of 7-bit data transfer in asynchronous mode)
- Module stop mode can be set
 - As the initial setting, SCI operation is halted. Register access is enabled by exiting module stop mode.

3.7.2 Block Diagram

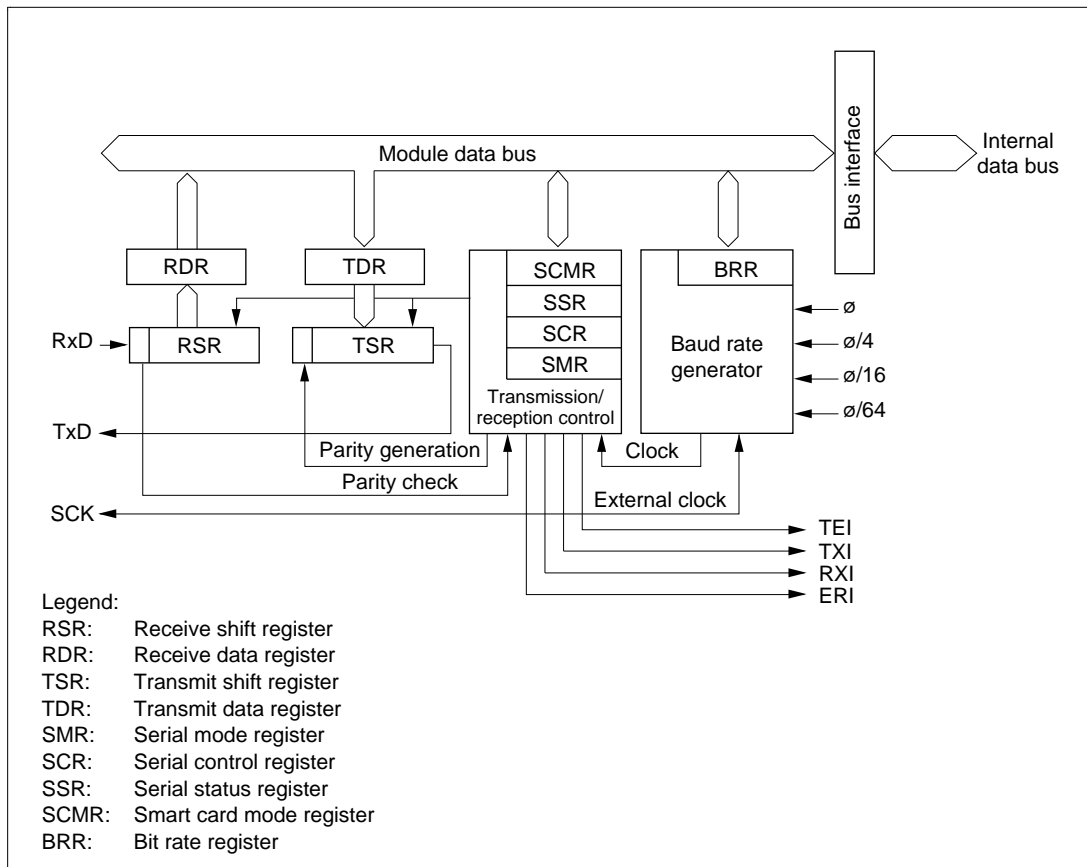


Figure 3.32 Block Diagram of SCI

Table 3.6 SCI Interrupt Sources and Data Transfer Controller (DTC) Activation

Channel	Interrupt Source	Description	DTC Activation	Priority*1
0	ERI0	Interrupt due to receive error (ORER, FER, or PER)	Not possible	High
	RXI0	Interrupt due to receive data full state (RDRF)	Possible	
	TXI0	Interrupt due to transmit data empty state (TDRE)	Possible	
	TEI0	Interrupt due to transmission end (TEND)	Not possible	
1	ERI1	Interrupt due to receive error (ORER, FER, or PER)	Not possible	
	RXI1	Interrupt due to receive data full state (RDRF)	Possible	
	TXI1	Interrupt due to transmit data empty state (TDRE)	Possible	
	TEI1	Interrupt due to transmission end (TEND)	Not possible	
2*2	ERI2	Interrupt due to receive error (ORER, FER, or PER)	Not possible	
	RXI2	Interrupt due to receive data full state (RDRF)	Possible	
	TXI2	Interrupt due to transmit data empty state (TDRE)	Possible	
	TEI2	Interrupt due to transmission end (TEND)	Not possible	
3	ERI3	Interrupt due to receive error (ORER, FER, or PER)	Not possible	
	RXI3	Interrupt due to receive data full state (RDRF)	Possible	
	TXI3	Interrupt due to transmit data empty state (TDRE)	Possible	
	TEI3	Interrupt due to transmission end (TEND)	Not possible	Low

Notes: 1. The table shows the initial state immediately after a reset. Relative channel priorities can be changed by means of the interrupt controller.
2. Only applies to the H8S/2237 Series.

SCI Asynchronous Communication: Asynchronous mode is a serial communication mode in which synchronization is achieved on a character by character basis, using a start bit and one or two stop bits.

- Twelve serial data transfer formats
 - Data length: 7 or 8 bits
 - Stop bit length: 1 or 2 bits
 - Parity: Even/odd/none
 - Multiprocessor bit: 1 or 0
- Selection of on-chip baud rate generator or external clock from SCK pin as clock source
- Transmit/receive clock can be output from SCK pin
- Break detection
 - A break can be detected by reading the RxD pin level directly in case of a framing error
- Multiprocessor communication capability

Table 3.7 Serial Transfer Formats (Asynchronous Mode)

SMR Settings				Serial Transfer Format and Frame Length											
CHR	PE	MP	STOP	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	S	8-bit data								STOP		
0	0	0	1	S	8-bit data								STOP	STOP	
0	1	0	0	S	8-bit data								P	STOP	
0	1	0	1	S	8-bit data								P	STOP	STOP
1	0	0	0	S	7-bit data							STOP			
1	0	0	1	S	7-bit data							STOP	STOP		
1	1	0	0	S	7-bit data							P	STOP		
1	1	0	1	S	7-bit data							P	STOP	STOP	
0	—	1	0	S	8-bit data								MPB	STOP	
0	—	1	1	S	8-bit data								MPB	STOP	STOP
1	—	1	0	S	7-bit data							MPB	STOP		
1	—	1	1	S	7-bit data							MPB	STOP	STOP	

Legend:

S: Start bit

STOP: Stop bit

P: Parity bit

MPB: Multiprocessor bit

Multiprocessor Communication Function: A multiprocessor format, in which a multiprocessor bit is added to the transfer data, can be used for serial communication, enabling data transfer to be performed among a number of processors.

The transmitting station first sends the ID of the receiving station with which it wants to perform serial communication as data with a 1 multiprocessor bit added. It then sends transmit data as data with a 0 multiprocessor bit added.

The receiving station skips the data until data with a 1 multiprocessor bit is sent.

When data with a 1 multiprocessor bit is received, the receiving station compares that data with its own ID. The station whose ID matches then receives the data sent next. Stations whose ID does not match continue to skip the data until data with a 1 multiprocessor bit is again received.

SCI Synchronous Communication: In synchronous mode, data is transmitted or received in synchronization with clock pulses, making it suitable for high-speed serial communication.

- Data length: 8 bits per character
- Overrun error detection
- Selection of on-chip baud rate generator or external clock from SCK pin as transmit/receive clock source
- Selection of LSB-first or MSB-first transfer
- Communication is possible with chips provided with a synchronous mode, such as the H8 Series, HD64180, and HD6301

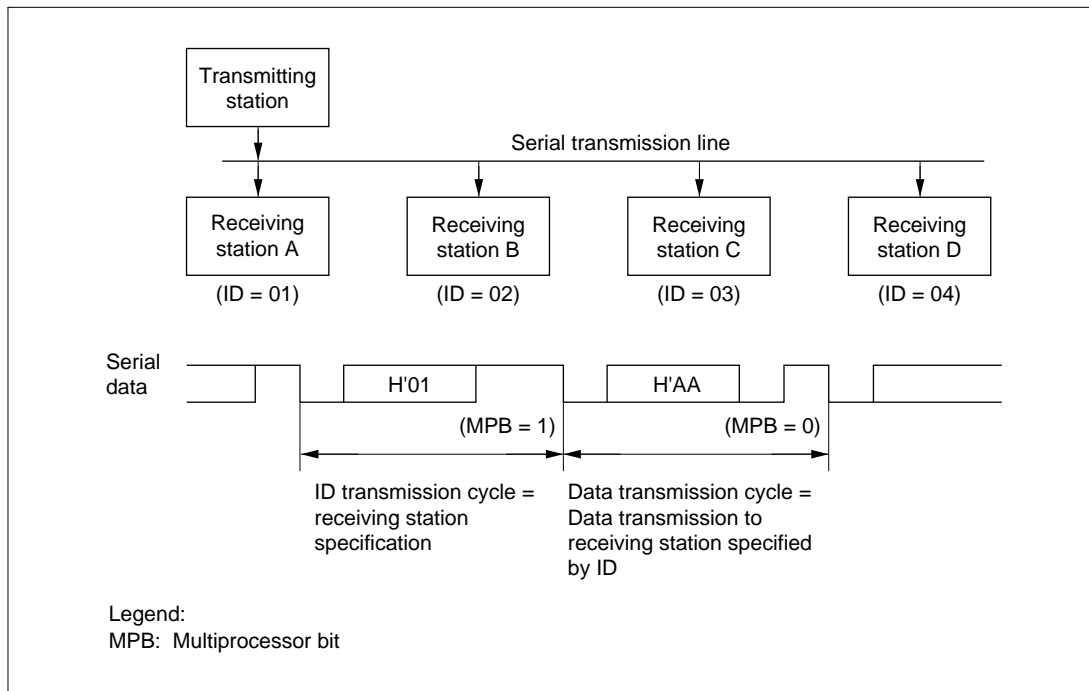


Figure 3.33 Example of Inter-Processor Communication Using Multiprocessor Format (Transmission of Data H'AA to Receiving Station A)

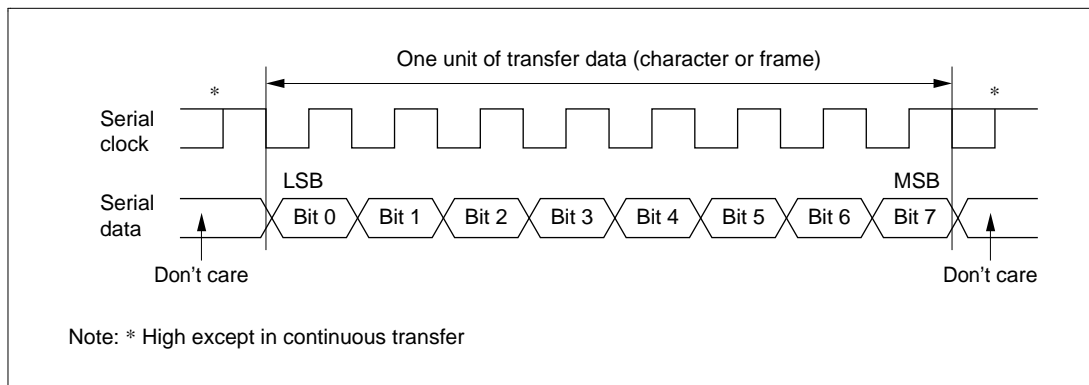


Figure 3.34 Data Format in Synchronous Communication

Table 3.8 BRR Settings for Various Bit Rates (Clocked Synchronous Mode)

Bit Rate (bit/s)	Operating Frequency ϕ (MHz)									
	$\phi = 2$ MHz		$\phi = 4$ MHz		$\phi = 6$ MHz		$\phi = 8$ MHz		$\phi = 10$ MHz	
	n	N	n	N	n	N	n	N	n	N
110	3	70	—	—						
250	2	124	2	249			3	124	—	—
500	1	249	2	124			2	249	—	—
1 k	1	124	1	249			2	124	—	—
2.5 k	0	199	1	99	1	149	1	199	1	249
5 k	0	99	0	199	1	74	1	99	1	124
10 k	0	49	0	99	0	149	0	199	0	249
25 k	0	19	0	39	0	59	0	79	0	99
50 k	0	9	0	19	0	29	0	39	0	49
100 k	0	4	0	9	0	14	0	19	0	24
250 k	0	1	0	3	0	5	0	7	0	9
500 k	0	0*	0	1	0	2	0	3	0	4
1 M			0	0*			0	1		
2.5 M									0	0*
5 M										

Note: As far as possible, the setting should be made so that the error is no more than 1%.

Legend:

Blank: Cannot be set.

—: Can be set, but there will be a degree of error.

*: Continuous transfer is not possible.

The BRR setting is found from the following formulas.

Asynchronous mode:

$$N = \frac{\phi}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

Clocked synchronous mode:

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

Where B: Bit rate (bit/s)

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

Ø: Operating frequency (MHz)

n: Baud rate generator input clock (n = 0 to 3)

(See the table below for the relation between n and the clock.)

n	Clock
0	Ø
1	Ø/4
2	Ø/16
3	Ø/64

3.8 Smart Card Interface

The SCI supports a smart card interface as an IC card interface serial communication function conforming to ISO/IEC7816-3 (Identification Card).

3.8.1 Features

- Asynchronous mode
 - Data length: 8 bits
 - Parity bit generation and checking
 - Transmission of error signal (parity error) in receive mode
 - Error signal detection and automatic data retransmission in transmit mode
 - Direct convention and inverse convention both supported
- Internal baud rate generator allows any bit rate to be selected
- Three interrupt sources
 - Three interrupt sources—transmit data empty, receive data full, and transmit/receive error—that can issue requests independently
 - The transmit data empty interrupt and receive data full interrupt can activate the data transfer controller (DTC) to execute data transfer

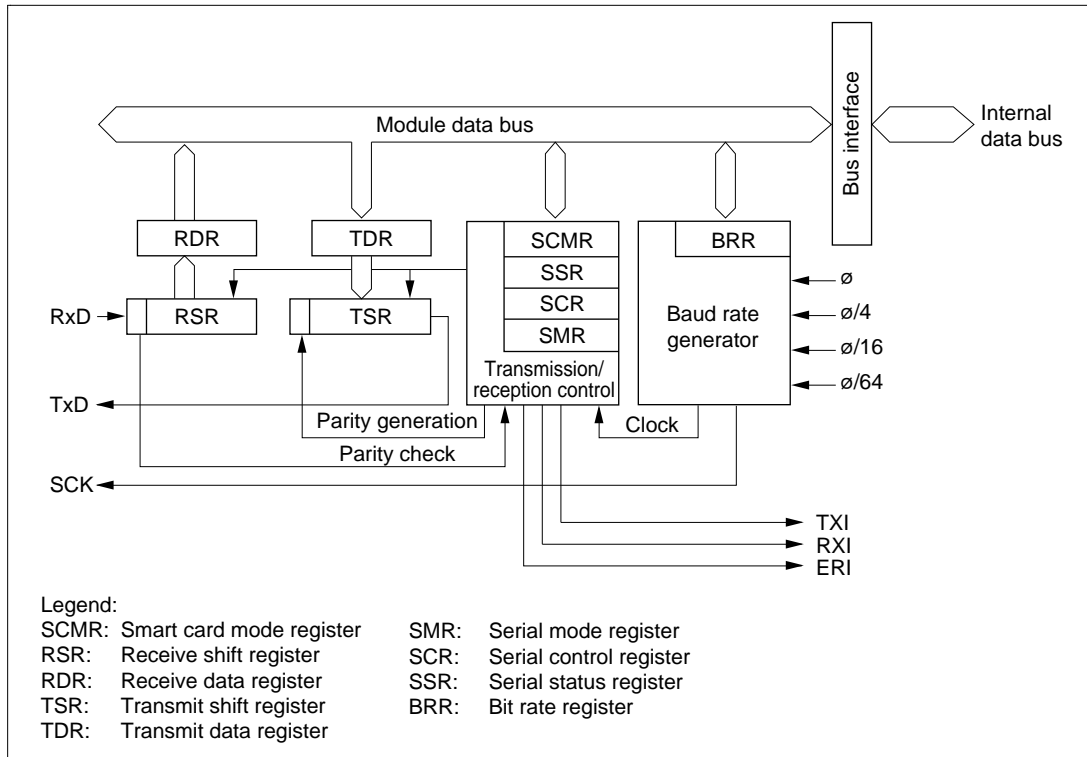


Figure 3.35 Smart Card Interface Block Diagram

3.8.2 Operation

- Only asynchronous communication is supported, with one frame consisting of 8-bit data plus and a parity bit.
- In transmission, a guard time of at least 2 etu (Elementary Time Unit: the time for transfer of one bit) is left between the end of the parity bit and the start of the next frame.
- If a parity error is detected during reception, a low error signal level is output for a 1 etu period 10.5 etu after the start bit.
- If the error signal is sampled during transmission, the same data is transmitted automatically after the elapse of 2 etu or longer.

3.8.3 Schematic Connection Diagram

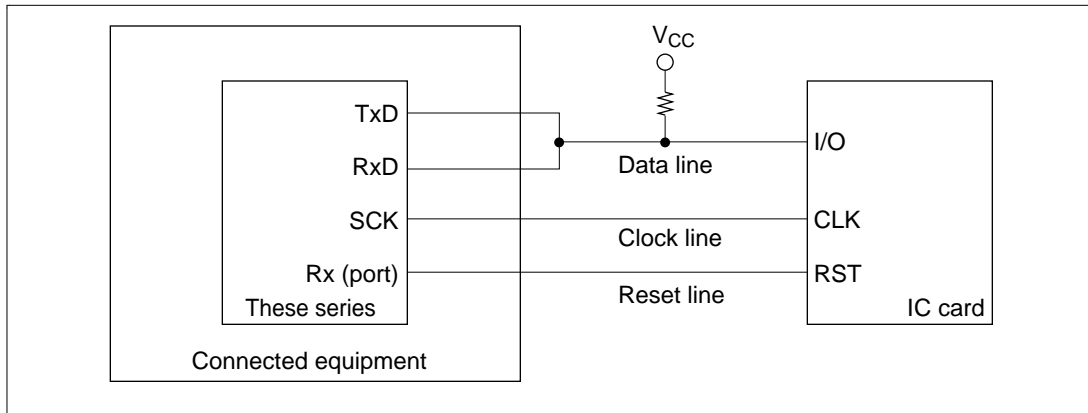


Figure 3.36 Schematic Diagram of Smart Card Interface Pin Connections

3.8.4 Data Format

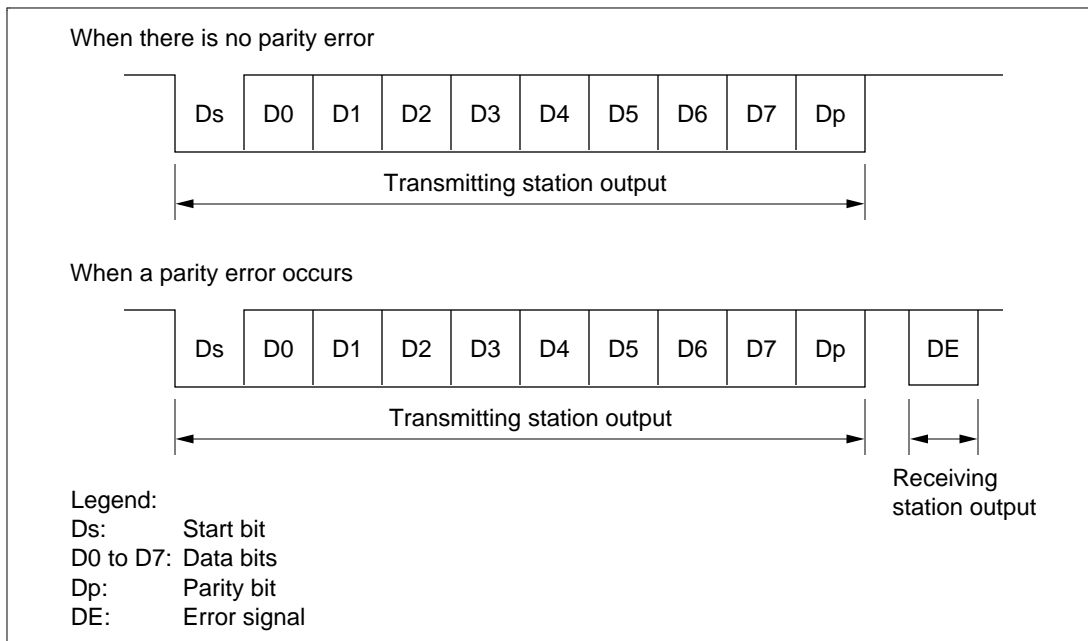


Figure 3.37 Smart Card Interface Data Format

3.9 A/D Converter

These series incorporate an on-chip A/D converter with 10-bit precision. Analog signals can be input on up to eight channels by the program.

3.9.1 Features

- 10-bit resolution
- Eight input channels
- Settable analog conversion voltage range
 - Conversion of analog voltages from 0 V to V_{ref} , with the reference voltage pin (V_{ref}) as the analog reference voltage
- High-speed conversion
 - Minimum conversion time:
 - 13.4 μs per channel (at 10 MHz operation)
- Selection of single mode or scan mode
 - Single mode: A/D conversion of one channel
 - Scan mode: continuous conversion on one to four channels
- Three kinds of conversion start
 - Selection of software or timer conversion start trigger (TPU or 8-bit timer), or $\overline{\text{ADTRG}}$ pin
- Four data registers
 - Conversion results held in a data register for each channel
- Sample and hold function
- A/D conversion end interrupt generation
 - A/D conversion end interrupt (ADI) request can be generated at the end of A/D conversion
- Module stop mode can be set
 - As the initial setting, A/D converter operation is halted. Register access is enabled by exiting module stop mode.

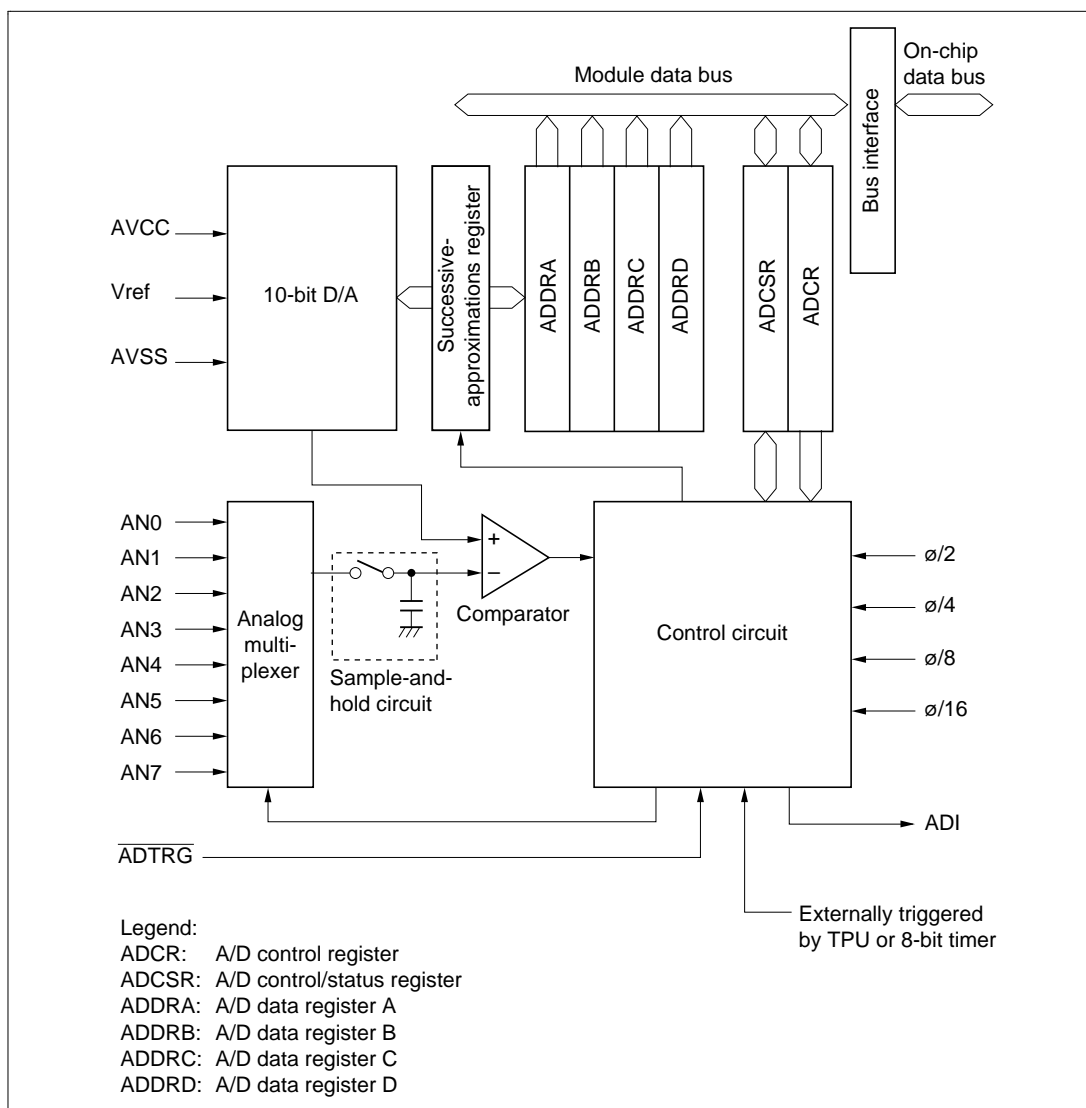


Figure 3.38 A/D Converter Block Diagram

Input Channel Setting

Eight-channel analog input is performed by means of the scan mode bit (SCAN) and channel select bits (CH2 to CH0) in ADCSR.

Bit 2	Bit 1	Bit 0	Description	
CH2	CH1	CH0	Single mode (SCAN = 0)	Scan mode (SCAN = 1)
0	0	0	AN0 (initial value)	AN0
		1	AN1	AN0 to AN1
	1	0	AN2	AN0 to AN2
		1	AN3	AN0 to AN3
1	0	0	AN4	AN4
		1	AN5	AN4 to AN5
	1	0	AN6	AN4 to AN6
		1	AN7	AN4 to AN7

Operation

The successive comparison method is used for A/D conversion, with a 10-bit resolution. There are two operating modes—single or scan.

Single Mode: Single mode is selected when A/D conversion is to be performed on a single channel only.

A/D conversion is started when the ADST bit is set to 1, according to the specified conversion start condition.

On completion of conversion, the ADF flag is set to 1. If the ADIE bit is set to 1 at this time, an ADI interrupt request is generated.

Scan Mode: Scan mode is selected when A/D conversion is to be performed repeatedly on a number of channels.

Once the ADST bit is set to 1 according to the specified conversion start condition, A/D conversion is performed repeatedly on the selected channel until the ADST bit is cleared to 0 by software.

An ADI interrupt request can be generated on completion of the first conversion operation for all the selected input channels.

3.10 D/A Converter

The H8S/2237 Series has an on-chip D/A converter with 8-bit precision. Analog signals can be output on up to two channels by the program.

3.10.1 Features

- Eight-bit resolution
- Two output channels
- Maximum conversion time of 10 μ s (with 20 pF load capacitance)
- Output voltage of 0 V to V_{ref}
- D/A output hold function in software standby mode
- Module stop mode can be set
 - As the initial setting, D/A converter operation is halted. Register access is enabled by exiting module stop mode.

3.10.2 Operation

D/A converter operation is enabled by setting the D/A output enable bit to 1. While this bit is set to 1, DADR contents are constantly converted and output to the corresponding pin.

The output value is:
$$\frac{\text{DADR contents}}{256} \times V_{\text{ref}}$$

3.10.3 D/A Converter Block Diagram

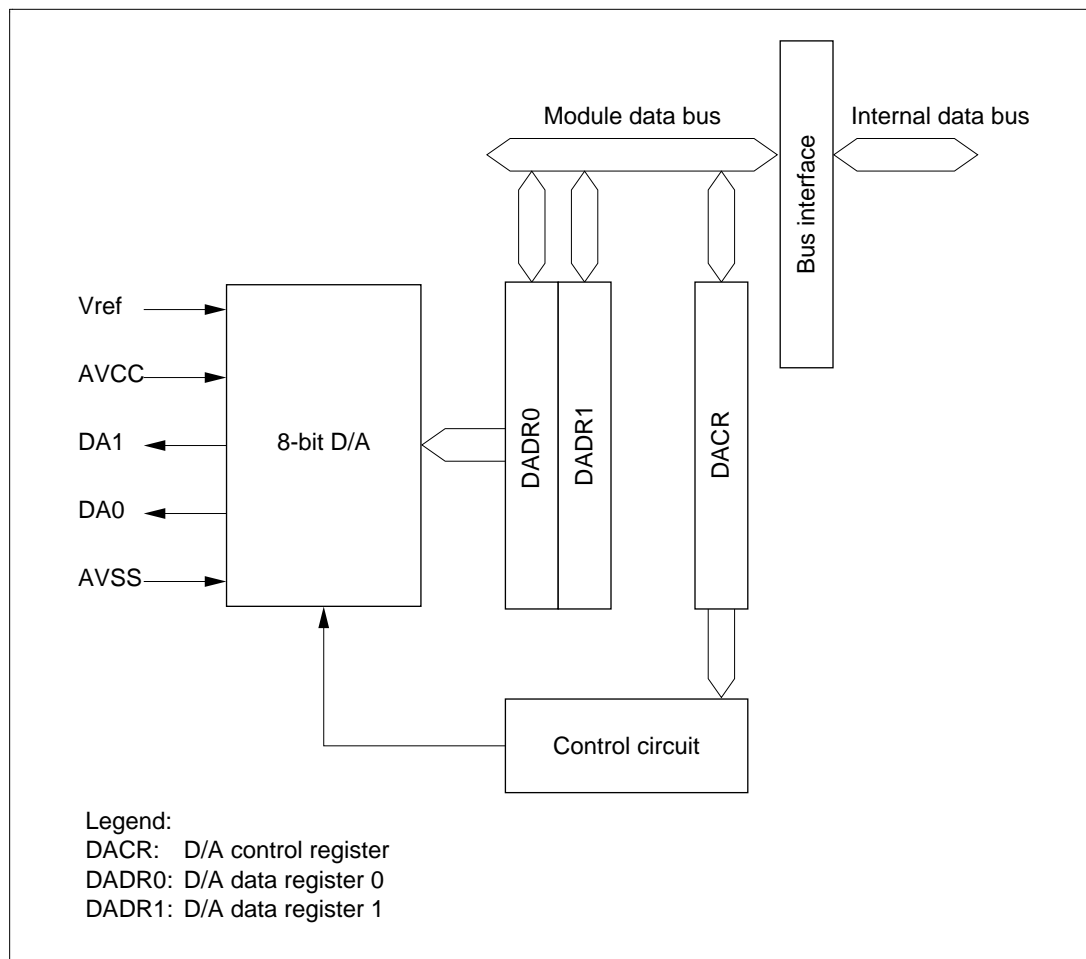


Figure 3.39 Block Diagram of D/A Converter

3.11 I/O Ports

The H8S/2237 Series and H8S/2227 Series have ten I/O ports (ports 1, 3, 7, and A to G), and two input-only ports (ports 4 and 9).

Each port includes a data direction register (DDR) that controls input/output, a data register (DR) that stores output data, and a port register (PORT) used to read the pin states.

In addition to DDR and DR, ports A to E also have a MOS input pull-up control register (PCR) to control the on/off state of MOS pull-up.

Table 3.9 H8S/2237 Series Port Functions in Each Operating Mode

Port	Description	Pins	Modes 4 and 5	Mode 6	Mode 7
Port 1	8-bit I/O port Schmitt-triggered input ($\overline{\text{IRQ1}}$, $\overline{\text{IRQ0}}$)	P17/TIOCB2/TCLK D	8-bit I/O port also functioning as TPU I/O pins (TCLKA, TCLKB, TCLKC, TCLKD, TIOCA0, TIOCB0, TIOCC0, TIOCD0, TIOCA1, TIOCB1, TIOCA2, TIOCB2), and external interrupt input ($\overline{\text{IRQ0}}$, $\overline{\text{IRQ1}}$)		
		P16/TIOCA2/ $\overline{\text{IRQ1}}$			
		P15/TIOCB1/TCLK C			
		P14/TIOCA1/ $\overline{\text{IRQ0}}$	8-bit I/O port also functioning as TPU I/O pins (TCLKA, TCLKB, TIOCA0, TIOCB0, TIOCC0, TIOCD0), external interrupt input ($\overline{\text{IRQ0}}$, $\overline{\text{IRQ1}}$), and address output (A20 to A23)		
		P13/TIOCD0/TCLK B/A23			
		P12/TIOCC0/TCLK A/A22			
	P11/TIOCB0/A21				
	P10/TIOCA0/A20				
Port 3	7-bit I/O port Open-drain output capability Schmitt-triggered input ($\overline{\text{IRQ5}}$, $\overline{\text{IRQ4}}$)	P36	7-bit I/O port also functioning as SCI (channel 0 and 1) I/O pins (TxD0, RxD0, SCK0, TxD1, RxD1, SCK1) and interrupt input ($\overline{\text{IRQ4}}$, $\overline{\text{IRQ5}}$)		
		P35/SCK1/ $\overline{\text{IRQ5}}$			
		P34/RxD1			
		P33/TxD1			
		P32/SCK0/ $\overline{\text{IRQ4}}$			
		P31/RxD0			
	P30/TxD0				
Port 4	8-bit input port	P47/AN7–P40/AN0	8-bit input port also functioning as A/D converter analog input (AN7 to AN0)		
Port 7	8-bit I/O port	P77/TxD3	8-bit I/O port also functioning as SCI (channel 3) I/O pins (TxD3, RxD3, SCK3), manual reset pin ($\overline{\text{MRES}}$), and 8-bit timer (channel 0 and 1) I/O pins (TMRI01, TMCIO1, TMO0, TMO1)		
		P76/RxD3			
		P75/SCK3			
		P74/ $\overline{\text{MRES}}$			
		P73/TMO1/ $\overline{\text{CS7}}$	When DDR = 0: Input port also functioning as 8-bit timer (channel 0 and 1) I/O pins		
		P72/TMO0/ $\overline{\text{CS6}}$			
		P71/ $\overline{\text{CS5}}$	When DDR = 1: 8-bit timer (channel 0 and 1) I/O pins also functioning as $\overline{\text{CS7}}$ to $\overline{\text{CS4}}$ output		
		P70/TMRI01/TMCIO1/ $\overline{\text{CS4}}$			

Port	Description	Pins	Modes 4 and 5	Mode 6	Mode 7
Port 9	• 2-bit input port	P97/DA1–P96/DA0	2-bit input port also functioning as D/A converter analog output (DA1, DA0)		
Port A	• 4-bit I/O port • Built-in MOS input pull-up • Open-drain output capability	PA3/A19/SCK2 PA2/A18/RxD2 PA1/A17/TxD2 PA0/A16	I/O port also functioning as SCI (channel 2) I/O pins (TxD2, RxD2, SCK2) and address output (A16 to A19)		I/O port also functioning as SCI (channel 2) I/O pins (TxD2, RxD2, SCK2)
Port B	• 8-bit I/O port • Built-in MOS input pull-up	PB7/A15/TIOCB5 PB6/A14/TIOCA5 PB5/A13/TIOCB4 PB4/A12/TIOCA4 PB3/A11/TIOCD3 PB2/A10/TIOCC3 PB1/A9/TIOCB3 PB0/A8/TIOCA3	I/O port also functioning as TPU I/O pins (TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, TIOCB5) and address output (A8 to A15)		I/O port also functioning as TPU I/O pins (TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, TIOCB5)
Port C	• 8-bit I/O port • Built-in MOS input pull-up	PC7/A7–PC0/A0	Address output (A0 to A7)	When DDR = 0: Input port When DDR = 1: Address output	I/O port
Port D	• 8-bit I/O port • Built-in MOS input pull-up	PD7/D15–PD0/D8	Data bus input/output		I/O port
Port E	• 8-bit I/O port • Built-in MOS input pull-up	PE7/D7–PE0/D0	8-bit bus mode: I/O port 16-bit bus mode: Data bus input/output		I/O port
Port F	• 8-bit I/O port • Schmitt-triggered input ($\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$)	PF7/ \emptyset	When DDR = 0: Input port When DDR = 1 (after reset): \emptyset output		When DDR = 0 (after reset): Input port When DDR = 1: \emptyset output
		PF6/ $\overline{\text{AS}}$ PF5/ $\overline{\text{RD}}$ PF4/ $\overline{\text{HWR}}$	$\overline{\text{AS}}$, $\overline{\text{RD}}$, $\overline{\text{HWR}}$ output		I/O port
		PF3/ $\overline{\text{LWR}}$ / $\overline{\text{ADTRG}}$ / $\overline{\text{IRQ3}}$	16-bit bus mode: $\overline{\text{LWR}}$ output 8-bit bus mode: I/O port also functioning as interrupt input pin ($\overline{\text{IRQ3}}$) and A/D converter input ($\overline{\text{ADTRG}}$)		I/O port also functioning as interrupt input pin ($\overline{\text{IRQ3}}$) and A/D converter input ($\overline{\text{ADTRG}}$)
		PF2/ $\overline{\text{WAIT}}$	When WAITE = 0 (after reset): I/O port When WAITE = 1: $\overline{\text{WAIT}}$ input		I/O port

Port	Description	Pins	Modes 4 and 5	Mode 6	Mode 7
Port F	<ul style="list-style-type: none"> 8-bit I/O port Schmitt-triggered input ($\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$) 	PF1/ $\overline{\text{BACK}}$ /BUZZ	When BRLE = 0 (after reset): I/O port also functioning as WDT output pin (BUZZ) When BRLE = 1: $\overline{\text{BACK}}$ output		I/O port also functioning as WDT output pin (BUZZ)
		PF0/ $\overline{\text{BREQ}}$ / $\overline{\text{IRQ2}}$	When BRLE = 0 (after reset): I/O port also functioning as interrupt input pin ($\overline{\text{IRQ2}}$) When BRLE = 1: $\overline{\text{BREQ}}$ input also functioning as interrupt input pin ($\overline{\text{IRQ2}}$)		I/O port also functioning as interrupt input pin ($\overline{\text{IRQ2}}$)
Port G	<ul style="list-style-type: none"> 5-bit I/O port Schmitt-triggered input ($\overline{\text{IRQ7}}$, $\overline{\text{IRQ6}}$) 	PG4/ $\overline{\text{CS0}}$	When DDR = 0 ^{*1} : Input port When DDR = 1 ^{*2} : $\overline{\text{CS0}}$ output		I/O port also functioning as interrupt input pins ($\overline{\text{IRQ6}}$, $\overline{\text{IRQ7}}$)
		PG3/ $\overline{\text{CS1}}$	When DDR = 0 (after reset): Input port also functioning as interrupt input pin ($\overline{\text{IRQ7}}$)		
		PG2/ $\overline{\text{CS2}}$			
		PG1/ $\overline{\text{CS3}}$ / $\overline{\text{IRQ7}}$	When DDR = 1: Interrupt input pin ($\overline{\text{IRQ7}}$) also functions as $\overline{\text{CS1}}$, $\overline{\text{CS2}}$, $\overline{\text{CS3}}$ output		
		PG0/ $\overline{\text{IRQ6}}$			

Notes: 1. After a mode 6 reset
2. After a mode 4 or 5 reset

Table 3.10 H8S/2227 Series Port Functions in Each Operating Mode

Port	Description	Pins	Modes 4 and 5	Mode 6	Mode 7
Port 1	<ul style="list-style-type: none"> • 8-bit I/O port • Schmitt-triggered input ($\overline{\text{IRQ1}}$, $\overline{\text{IRQ0}}$) 	P17/TIOCB2/TCLKD P16/TIOCA2/ $\overline{\text{IRQ1}}$ P15/TIOCB1/TCLKC P14/TIOCA1/ $\overline{\text{IRQ0}}$	I/O port also functioning as TPU I/O pins (TCLKA, TCLKB, TCLKC, TCLKD, TIOCA0, TIOCB0, TIOCC0, TIOCD0, TIOCA1, TIOCB1, TIOCA2, TIOCB2) and interrupt input ($\overline{\text{IRQ0}}$, $\overline{\text{IRQ1}}$)		
		P13/TIOCD0/TCLKB/A23 P12/TIOCC0/TCLKA/A22 P11/TIOCB0/A21 P10/TIOCA0/A20	I/O port also functioning as TPU I/O pins (TCLKA, TCLKB, TIOCA0, TIOCB0, TIOCC0, TIOCD0) and address output (A20 to A23)		
Port 3	<ul style="list-style-type: none"> • 7-bit I/O port • Open-drain output capability • Schmitt-triggered input ($\overline{\text{IRQ5}}$, $\overline{\text{IRQ4}}$) 	P36 P35/SCK/ $\overline{\text{IRQ5}}$ P34/RxD1 P33/TxD1 P32/SCK0/ $\overline{\text{IRQ4}}$ P31/RxD0 P30/TxD0	7-bit I/O port also functioning as SCI (channel 0 and 1) I/O pins (TxD0, RxD0, SCK0, TxD1, RxD1, SCK1) and interrupt input ($\overline{\text{IRQ4}}$, $\overline{\text{IRQ5}}$)		
Port 4	<ul style="list-style-type: none"> • 8-bit input port 	P47/AN7–P40/AN0	8-bit input port also functioning as A/D converter analog input (AN7 to AN0)		
Port 7	<ul style="list-style-type: none"> • 8-bit I/O port 	P77/TxD3 P76/RxD3 P75/SCK3 P74/MRES	8-bit I/O port also functioning as SCI (channel 3) I/O pins (TxD3, RxD3, SCK3), manual reset pin ($\overline{\text{MRES}}$), and 8-bit timer (channel 0 and 1) I/O pins (TMRI01, TMCi01, TMO0, TMO1)		
		P73/TMO1/ $\overline{\text{CS7}}$ P72/TMO0/ $\overline{\text{CS6}}$ P71/ $\overline{\text{CS5}}$	When DDR = 0: Input port also functioning as 8-bit timer (channel 0 and 1) I/O pins		
		P70/TMRI01/TMCi01/ $\overline{\text{CS4}}$	When DDR = 1: 8-bit timer (channel 0 and 1) I/O pins also functioning as $\overline{\text{CS7}}$ to $\overline{\text{CS4}}$ output		
Port 9	<ul style="list-style-type: none"> • 2-bit input port 	P97, P96	Input port		
Port A	<ul style="list-style-type: none"> • 4-bit I/O port • Built-in MOS input pull-up • Open-drain output capability 	PA3/A19–PA0/A16	I/O port also functioning as address output (A16 to A19)		I/O port
Port B	<ul style="list-style-type: none"> • 8-bit I/O port • Built-in MOS input pull-up 	PB7/A15–PB0/A8	I/O port also functioning as address output (A8 to A15)		I/O port

Port	Description	Pins	Modes 4 and 5	Mode 6	Mode 7
Port C	<ul style="list-style-type: none"> 8-bit I/O port Built-in MOS input pull-up 	PC7/A7–PC0/A0	Address output (A0 to A7)	I/O port also functioning as address output (A0 to A7)	I/O port
Port D	<ul style="list-style-type: none"> 8-bit I/O port Built-in MOS input pull-up 	PD7/D15–PD0/D8	Data bus input/output		I/O port
Port E	<ul style="list-style-type: none"> 8-bit I/O port Built-in MOS input pull-up 	PE7/D7–PE0/D0	8-bit bus mode: I/O port 16-bit bus mode: Data bus input/output		I/O port
Port F	<ul style="list-style-type: none"> 8-bit I/O port Schmitt-triggered input ($\overline{\text{IRQ3}}$, $\overline{\text{IRQ2}}$) 	PF7/ \emptyset	When DDR = 0: Input port When DDR = 1 (after reset): \emptyset output		When DDR = 0 (after reset): Input port When DDR = 1: \emptyset output
		PF6/ $\overline{\text{AS}}$ PF5/ $\overline{\text{RD}}$ PF4/ $\overline{\text{HWR}}$	$\overline{\text{AS}}$, $\overline{\text{RD}}$, $\overline{\text{HWR}}$ output		I/O port
		PF3/ $\overline{\text{LWR}}$ / $\overline{\text{ADTRG}}$ / $\overline{\text{IRQ3}}$	16-bit bus mode: $\overline{\text{LWR}}$ output 8-bit bus mode: I/O port also functioning as interrupt input pin ($\overline{\text{IRQ3}}$) and A/D converter input ($\overline{\text{ADTRG}}$)		I/O port also functioning as interrupt input pin ($\overline{\text{IRQ3}}$) and A/D converter input ($\overline{\text{ADTRG}}$)
		PF2/ $\overline{\text{WAIT}}$	When WAITE = 0 (after reset): I/O port When WAITE = 1: $\overline{\text{WAIT}}$ input		I/O port
		PF1/ $\overline{\text{BACK}}$ / $\overline{\text{BUZZ}}$	When BRLE = 0 (after reset): I/O port also functioning as WDT output pin ($\overline{\text{BUZZ}}$) When BRLE = 1: $\overline{\text{BACK}}$ output		I/O port also functioning as WDT output pin ($\overline{\text{BUZZ}}$)
		PF0/ $\overline{\text{BREQ}}$ / $\overline{\text{IRQ2}}$	When BRLE = 0 (after reset): I/O port also functioning as interrupt input pin ($\overline{\text{IRQ2}}$) When BRLE = 1: $\overline{\text{BREQ}}$ input also functioning as interrupt input pin ($\overline{\text{IRQ2}}$)		I/O port also functioning as interrupt input pin ($\overline{\text{IRQ2}}$)

Port	Description	Pins	Modes 4 and 5	Mode 6	Mode 7
Port G	<ul style="list-style-type: none"> 5-bit I/O port Schmitt-triggered input ($\overline{\text{IRQ7}}$, $\overline{\text{IRQ6}}$) 	$\text{PG4}/\overline{\text{CS0}}$	When $\text{DDR} = 0^{*1}$: Input port When $\text{DDR} = 1^{*2}$: $\overline{\text{CS0}}$ output		I/O port also functioning as interrupt input pins ($\overline{\text{IRQ6}}$, $\overline{\text{IRQ7}}$)
		$\text{PG3}/\overline{\text{CS1}}$	When $\text{DDR} = 0$ (after reset): Input port also functioning as interrupt input pin ($\overline{\text{IRQ7}}$)		
		$\text{PG2}/\overline{\text{CS2}}$			
		$\text{PG1}/\overline{\text{CS3}}/\overline{\text{IRQ7}}$	When $\text{DDR} = 1$: Interrupt input pin ($\overline{\text{IRQ7}}$) also functions as $\overline{\text{CS1}}$, $\overline{\text{CS2}}$, $\overline{\text{CS3}}$ output		
		$\text{PG0}/\overline{\text{IRQ6}}$			

Notes: 1. After a mode 6 reset
2. After a mode 4 or 5 reset

3.12 RAM

The on-chip RAM is connected to the CPU by a 16-bit data bus, enabling both byte data and word data to be accessed in one state. This makes it possible to perform fast word data transfer.

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

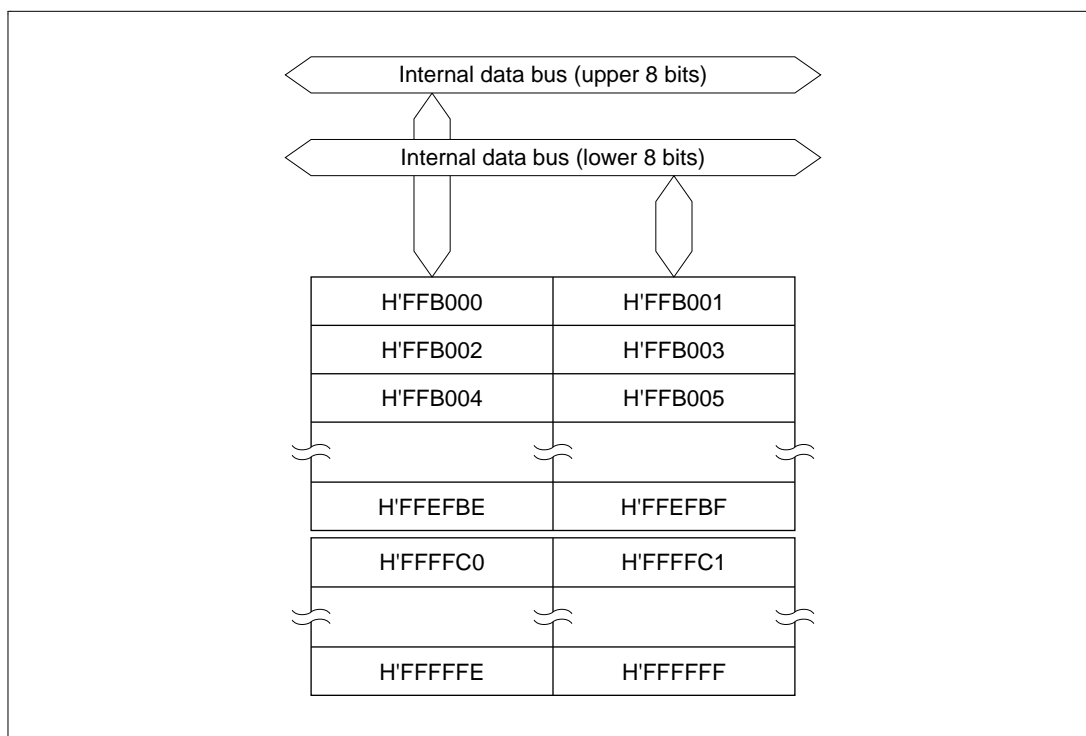


Figure 3.40 Block Diagram of RAM (Example of H8S/2237, H8S/2227)

- RAM Size

RAM (Bytes)	H8S/2237 Series	H8S/2227 Series
16 k	H8S/2237	H8S/2227
4 k	H8S/2235, H8S/2233	H8S/2225, H8S/2223

3.12.1 Features

The on-chip RAM in the H8S/2237 and H8S/2227 is allocated to addresses H'FFB000—H'FFEFBF and H'FFFC0—H'FFFFFF (16 kbytes).

The on-chip ROM in the H8S/2235, H8S/2233, H8S/2225, and H8S/2223 is allocated to addresses H'FFE000—H'FFEFBF and H'FFFC0—H'FFFFFF (4 kbytes).

3.13 ROM

The ROM is connected to the CPU by a 16-bit data bus, enabling both byte data and word data to be accessed in one state. This makes possible rapid instruction fetches and high-speed processing.

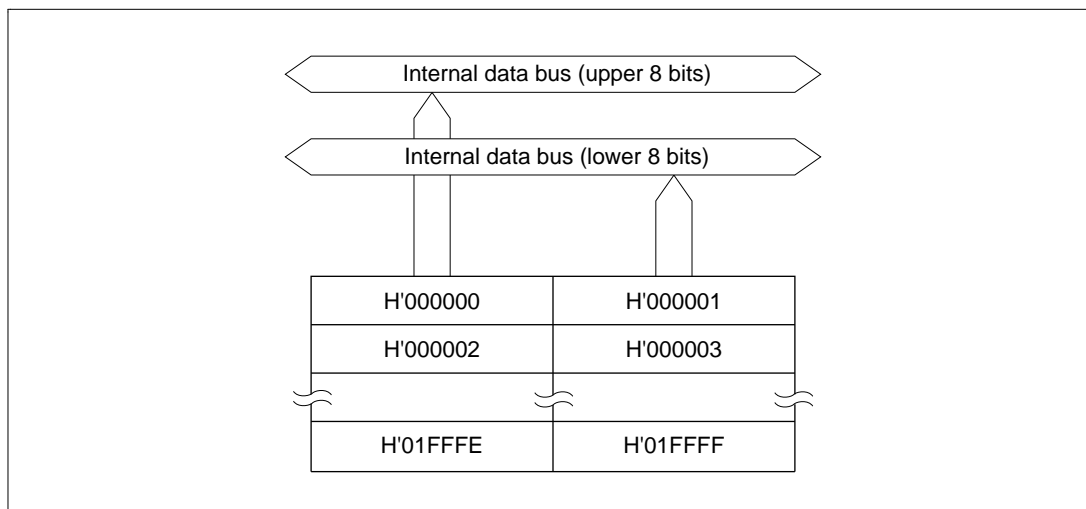


Figure 3.41 ROM Block Diagram (Example of H8S/2237 in Modes 6 and 7)

- PROM or Mask ROM Size

ROM (Bytes)	H8S/2237 Series	H8S/2227 Series
128 k	H8S/2237, H8/2235	H8S/2227, H8S/2225
64 k	H8S/2233	H8S/2223

- PROM Programming (ZTAT™)

This programming can be done with a PROM programmer set up in the same way as for the HN27C101 EPROM ($V_{PP} = 12.5$ V). Use of a 100-pin/32-pin socket adapter enables programming with a commercial PROM programmer. The address range is H'00000 to H'1FFFF. However, page programming is not supported.

Section 4 Power-Down Modes

In addition to the normal program execution state, the these series have power-down modes in which operation of the CPU and oscillator is halted and power dissipation is reduced. Low-power operation can be achieved by individually controlling the CPU, on-chip supporting modules, and so on.

These series operating modes are as follows:

1. High-speed mode
2. Medium-speed mode
3. Subactive mode
4. Sleep mode
5. Subsleep mode
6. Watch mode
7. Module stop mode
8. Software standby mode
9. Hardware standby mode

Of these, 2 to 9 are power-down modes. Sleep mode and subsleep mode are CPU modes, medium-speed mode is a CPU and bus master mode, subactive mode is a CPU, bus master, and on-chip supporting module mode, and module stop mode is an on-chip supporting module mode (including bus masters other than the CPU). A combination of certain of these modes can be set.

- **Medium-Speed Mode**

When bits SCK2 to SCK0 in the standby control register (SBYCR) are set to 1 in high-speed mode, medium-speed mode is entered as soon as the current bus cycle ends. In medium-speed mode, the bus masters—the CPU and DTC—operate on the operating clock ($f/2$, $f/4$, $f/8$, $f/16$, or $f/32$) specified by bits SCK2 to SCK0. However, on-chip supporting modules other than the bus masters operate on the high-speed clock (f).

- **Subactive Mode**

If a SLEEP instruction is executed in high-speed mode when the SSBY bit in SBYCR is set to 1, the DTON bit and LSON bit in the low-power control register (LPWRCR) are both set to 1, and the PSS bit in the timer control/status register (TCSR (WDT1)) is set to 1, the CPU enters subactive mode. A transition to subactive mode is also caused by an interrupt generated in watch mode when the LSON bit in LPWRCR is set to 1, and by an interrupt generated in subsleep mode.

In subactive mode, the CPU performs low-speed sequential program execution using the subclock, and supporting modules other than TMR, WDT0, and WDT1 are halted.

- **Sleep Mode**
If a SLEEP instruction is executed when the SSBY bit in SBYCR and the LSON bit in LPWRCR are both cleared to 0, the CPU enters sleep mode. In sleep mode, CPU operation stops but the contents of the CPU's internal registers are retained. Other supporting modules do not stop.
- **Subsleep Mode**
If a SLEEP instruction is executed in subactive mode when the SSBY bit in SBYCR is cleared to 0, the LSON bit in LPWRCR is set to 1, and the PSS bit in TCSR (WDT1) is set to 1, the CPU enters subsleep mode.
In subsleep mode, CPU operation stops, together with the operation of supporting modules other than TMR, WDT0, and WDT1. As long as the specified voltage is supplied, the contents of CPU registers, some on-chip peripheral registers, and on-chip RAM are retained, and I/O ports retain their states prior to the transition.
- **Watch Mode**
If a SLEEP instruction is executed in high-speed mode or subactive mode when the SSBY bit in SBYCR is set to 1, the DTON bit in LPWRCR is cleared to 0, and the PSS bit in TCSR (WDT1) is set to 1, the CPU enters watch mode.
In watch mode, CPU operation stops, together with the operation of supporting modules other than WDT1. As long as the specified voltage is supplied, the contents of CPU registers, some on-chip peripheral registers, and on-chip RAM are retained, and I/O ports retain their states prior to the transition.
- **Module Stop Mode**
Module stop mode can be set for individual on-chip supporting modules.
When the MSTP bit corresponding to a particular supporting module in the module stop control register (MSTPCR) is set to 1, operation of the specified module stops at the end of the bus cycle and a transition is made to module stop mode. The CPU continues operating independently.
- **Software Standby Mode**
If a SLEEP instruction is executed when the SSBY bit in SBYCR is set to 1, the LSON bit in LPWRCR is cleared to 0, and the PSS bit in TCSR (WDT1) is cleared to 0, software standby mode is entered. In this mode, the CPU, on-chip supporting modules, and oscillator all stop. However, the contents of the CPU's internal registers, on-chip RAM data, the states of on-chip supporting modules other than the A/D and D/A, and the states of I/O ports, are retained.
- **Hardware Standby Mode**
When the STBY pin is driven low, a transition is made to hardware standby mode from any state.
In hardware standby mode, all functions enter the reset state and stop operation, resulting in extremely low power consumption. As long as the specified voltage is supplied, on-chip RAM data is retained. I/O ports go to the high-impedance state.

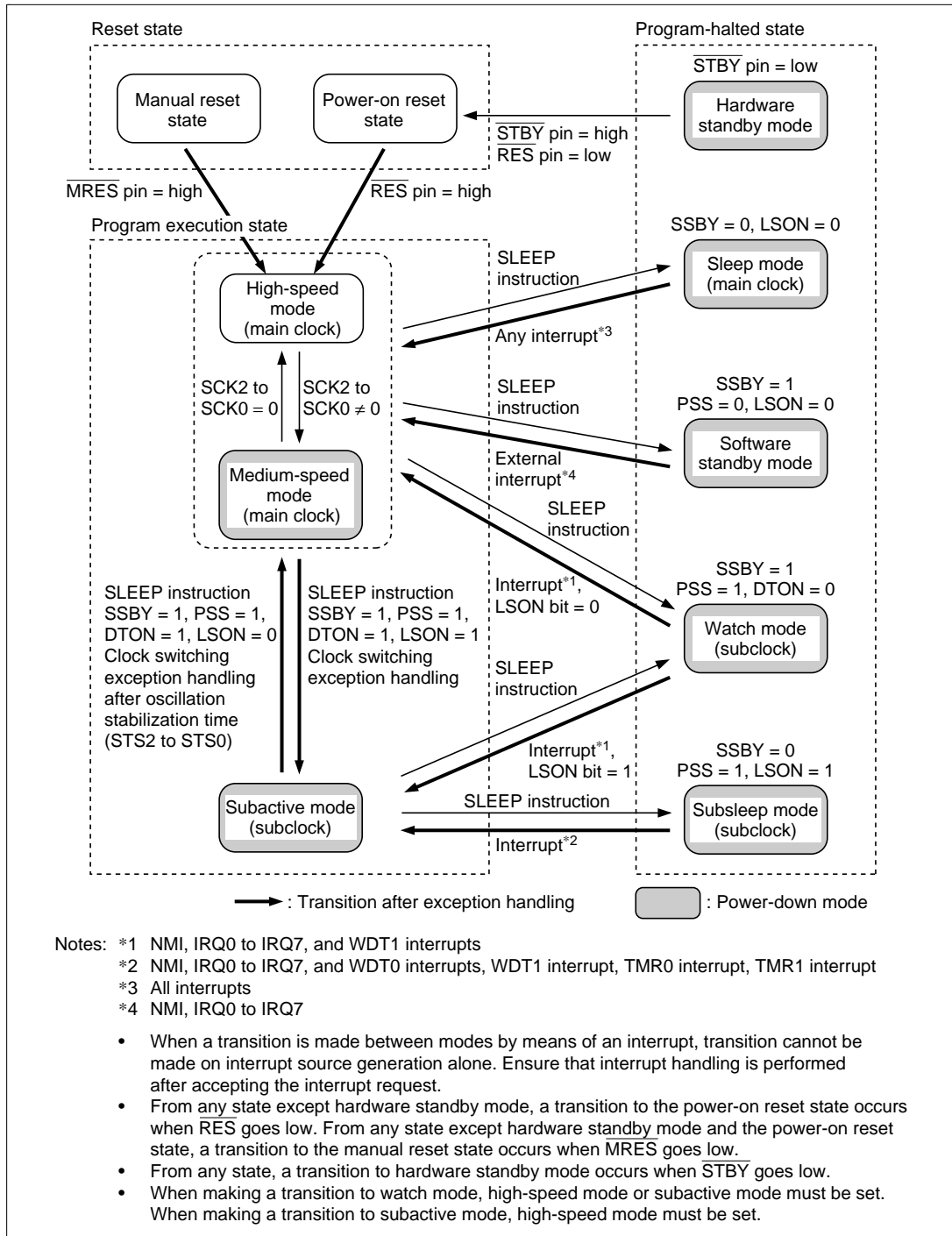


Figure 4.1 Mode Transitions

Table 4.1 Internal States in Each Mode

Function		High-Speed	Medium-Speed	Sleep	Module Stop	Watch	Subactive	Subsleep	Software Standby	Hardware Standby
System clock oscillator		Functioning	Functioning	Functioning	Functioning	Halted	Halted	Halted	Halted	Halted
Subclock input		Functioning	Functioning	Functioning	Functioning	Functioning	Functioning	Functioning	Functioning/halted	Halted
CPU operation	Instructions	Functioning	Medium-speed operation	Halted	Functioning	Halted	Subclock operation	Halted	Halted	Halted
	Registers			Retained		Retained		Retained	Retained	Undefined
External interrupts		Functioning	Functioning	Functioning	Functioning	Functioning	Functioning	Functioning	Functioning	Halted
Peripheral function operation	PBC	Functioning	Medium-speed operation	Functioning	Functioning/halted (retained)	Halted (retained)	Subclock operation	Halted (retained)	Halted (retained)	Halted (reset)
	DTC						Halted (retained)			
	WDT1	Functioning	Functioning	Functioning	Functioning	Subclock operation	Subclock operation	Subclock operation		
	WDT0					Halted (retained)				
	TMR				Functioning/halted (retained)					
	TPU						Halted (retained)	Halted (retained)		
	SCI									
	A/D				Functioning/halted (reset)	Halted (reset)	Halted (reset)	Halted (reset)	Halted (reset)	
	D/A									
	RAM	Functioning	Functioning	Functioning (DTC)	Functioning	Retained	Functioning	Retained	Retained	Retained
	I/O	Functioning	Functioning	Functioning	Functioning	Retained	Functioning	Functioning	Retained	High impedance

Note: "Halted (retained)" means that internal register values are retained. The internal state is "operation interrupted."

"Halted (reset)" means that internal register values and internal states are initialized.

In module stop mode, only modules for which a stop setting has been made are halted (reset or retained).

Table 4.2 Power-Down Mode Transition Conditions

State before Transition	Control Bit States at Time of Transition				State after Transition by SLEEP Instruction	State after Return by Interrupt
	SSBY	PSS	LSON	DTON		
High-speed/ medium-speed	0	*	0	*	Sleep	High-speed/ medium-speed
	0	*	1	*	—	—
	1	0	0	*	Software standby	High-speed/ medium-speed
	1	0	1	*	—	—
	1	1	0	0	Watch	High-speed
	1	1	1	0	Watch	Subactive
	1	1	0	1	—	—
	1	1	1	1	Subactive	—
Subactive	0	0	*	*	—	—
	0	1	0	*	—	—
	0	1	1	*	Subsleep	Subactive
	1	0	*	*	—	—
	1	1	0	0	Watch	High-speed
	1	1	1	0	Watch	Subactive
	1	1	0	1	High-speed	—
	1	1	1	1	—	—

*: Don't care

—: Do not set.

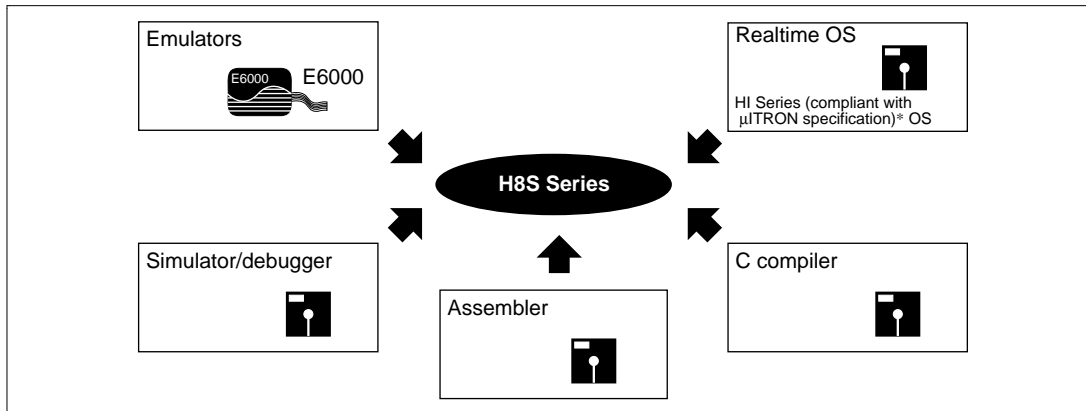
Section 5 Development Environment

5.1 Development Environment

A comprehensive development environment is provided for these series, including cross software, emulators, an HI Series OS*, and products offered by third-party suppliers.

5.1.1 Lineup

- Cross software
 - C compiler
 - Assembler
 - Simulator/debugger
- Emulators
 - E6000
- HI Series OS*
 - HI8-2600
- Third-party products
 - Cross software
 - Emulators



Note: * The HI Series OS is Hitachi real-time OS compliant with μITRON specifications. The product described in this publication is based on the ITRON specifications and was developed under the guidance of Dr. Ken Sakamura of The University of Tokyo. ITRON is an acronym of "Industrial TRON." TRON is an acronym of "The Real Time Operating System Nucleus." μITRON is an acronym of "Micro Industrial TRON."

5.2 Cross Software

Various kinds of cross software—including a C compiler, assembler, and simulator/debugger—are available for these series, to improve development efficiency.

C Compiler: The C programming language allows system operation and control structures to be written in a concise form. The C compiler converts a program written in C into machine language.

- Comprehensive support of H8S/2600, H8S/2000, H8/300, H8/300L, and H8/300 Series
- Complies with ANSI (American National Standard Institute) C standard proposals
- Object program size reduced by an average 30% or more through optimization processing
- Extended functions for product embedding
 - Built-in function facility: H8/2000 system control instructions can be written in function call format
 - Memory access function: Efficient memory access provided by short absolute addressing mode and indirect memory addressing mode
 - Assembler embedding: Assembly language code can be mixed in with C code
 - Interrupt function creation: Interrupt vector setting and interrupt handling routines can be written in C
- Generates object designed with ROM programming in mind
- Implements optimization between modules dependent on memory location and function call relationships, across source files
- Supports debugging information output in optimization

Assembler: Assembly language is suited to hardware-dependent processing that requires fast execution. The assembler converts a program written in assembly language into machine language.

- Common assembly language for entire H Series
- Structured assembler for easy maintenance
 - Supports IF, FOR, WHILE, REPEAT statements, etc.
- Execution instructions and control instructions compliant with IEEE standards
- Efficient macro functions

Simulator/Debugger: CPU operation can be simulated by software, and operation of a completed program can be tested without using the actual device.

- Operates on host computer without a target system
- Various powerful debugging functions
 - PC, data, register, sequential, and other break conditions can be set
 - A 1023-instruction trace buffer is supported, and traces can be executed in instruction or subroutine units
 - Support of single-line assembly, disassembly, state retention, command history, etc.

Table 5.1 Cross Software Product Lineup

Host Machine	C Compiler	Assembler System	Simulator/Debugger
IBM PC	PPS008PC200IA* ¹	PPS008PC200IA* ¹	—
FLORA (DOS/V)	PS008CAS2-IF3* ²	PS008CAS2-IF3* ²	PS008CAS2-IF3* ²
PC-9800 Series	PS008CPC200IE3	PS008ASM2-IF3	PS008SIM1-IF3
Windows 95	PPS008PC200WA* ¹	PPS008PC200WA* ¹	—
(PC-9800 Series,	PS008CAS2-MW* ²	PS008CAS2-MW* ²	PS008CAS2-MW* ²
IBM PC, FLORA)	PS008CPC200IW	PS008ASM2-MW	PS008SIM1-MW
SPARC	PPS008SL200A* ¹	PPS008SL200A* ¹	—
(Sun OS)	PS008CAS2-SLC* ²	PS008CAS2-SLC* ²	PS008CAS2-SLC* ²
	PS008CSL200	PS008ASM2-SLC	PS008SIM1-SPC
SPARC	PPS008SL200A* ¹	PPS008SL200A* ¹	—
(Solaris)	PS008CAS2-SLC* ²	PS008CAS2-SLC* ²	PS008CAS2-SLC* ²
	HSS008CLCS2SM	HSS008ASCS2SM	HSS008SDCS1SM
HP9000/700	PPS008HP200A* ¹	PPS008HP200A* ¹	—
HITACHI 9000	PS008CAS2-H7D* ²	PS008CAS2-H7D* ²	PS008CAS2-H7D* ²
HITACHI 9000V	PS008CHP7200	PS008ASM2-H7D	PS008SIM1-H7D

Notes: 1. C compiler package (C compiler + assembler)

2. C compiler/simulator package (C compiler + assembler + simulator)

5.3 Emulators

The E6000 emulator, incorporating a wide variety of functions, is available for use with these series.

E6000: The E6000 emulator provides an efficient debugging environment within a Windows environment. It is equipped with a DOS/V personal computer interface (covering desktop to notebook models), and offers simple operation via a graphical user interface on Windows 3.1 or Windows 95. The E6000 provides enhanced emulation functions in a compact A5-size body.

- Trace functions
 - 80-bit \times 32k-cycle trace capability, with provision for up to 12 trigger points and sequential settings up to 8 levels
 - Subroutine trace, specific-condition trace, and point-to-point trace settings
 - Provision for a time trace every trace cycle
- Break functions
 - Up to 12 hardware breaks can be set, with provision for sequential settings of up to 8 levels (shared with trace triggers)
 - A maximum of 255 software breaks can be set
- Fast downloading
 - 4-Mbyte/minute downloading
- C source level debugging
 - Halting of program execution in the C source program
 - Displays contents of specified variables in the source program
- Support for Windows environment
 - Runs on Windows 3.1 or Windows 95
 - Comprehensive help functions, for manual-less operation
 - More efficient mouse operations through use of pull-down menus and buttons for frequently used commands
- Performance functions
 - Number of subroutine time measurement modules (max. 8 channels), maximum/minimum measurement (4 channels)
 - Number of subroutine execution count measurement modules (max. 8 channels)
 - Timeout function
 - Number of run time measurement modules (2 channels)
 - Sampling time (1 μ s, 250 ns, 20 ns)

Table 5.2 Emulator Product Lineup

E6000 Emulator	Host System Interface Board	Expansion Memory	User Cable
HS2633EPI60H* (for IBM PC)	HS6000EII01H (IBM PC interface board)	HS6000EMS11H (1M-SRAM) HS6000EMS12H (4M-SRAM)	HS2237ECH61H* (FP-100B, TFP-100B) HS2237ECF61H* (FP-100A)

Note: * Under development

5.4 Socket Adapters

A socket adapter is a pin conversion adapter for writing or verifying a program in the on-chip PROM of a microcomputer using a PROM programmer. A different adapter is available for each product and package type, so be sure to choose the correct adapter for the product concerned.

Table 5.3 Applicable EPROM Programmers

Product	Package	Socket Adapter
H8S/2237	TFP-100B	Under development
	TFP-100G	
	FP-100A	
	FP-100B	

Method of Use: Refer to the socket adapter instruction manual for details.

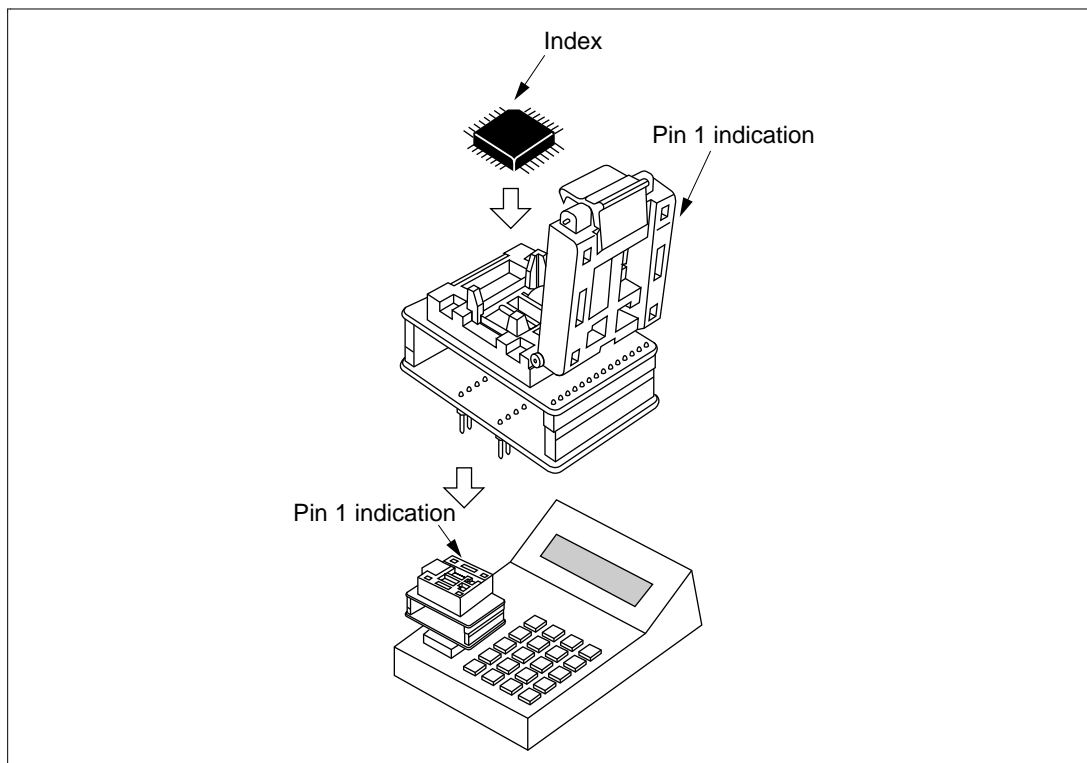


Figure 5.1 Method of Use

5.5 HI Series OS

An HI Series OS is available for these series, which enables the program structure to be made explicit and a custom-made operating system to be constructed.

5.5.2 Features of HI8-2600 Series

- Emphasis on real-time performance
 - Microsecond-order response to interrupts from off-chip
- Support for synchronous communication timeout specification
- Variable-length memory pool function
- ROM implementation capability
 - Designed with product embedding
- Modular structure using building-block system
 - System can be constructed by selecting modules according to the scale of the application system
- Improved development efficiency and reliability
 - Shorter development time, lower development costs, improved serviceability and reliability
- Support for E6000 emulator multitask debugger
 - A multitask debugger that runs in an E6000 emulator Windows environment is available, offering more efficient application debugging via a graphical user interface.

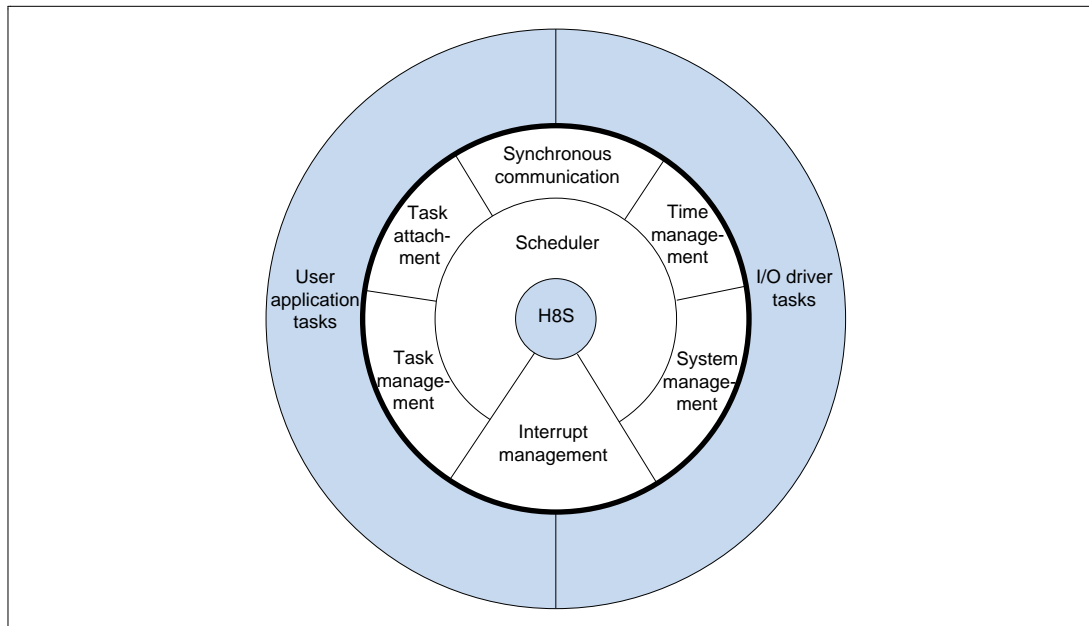


Figure 5.2 HI8-2600 System Configuration Diagram

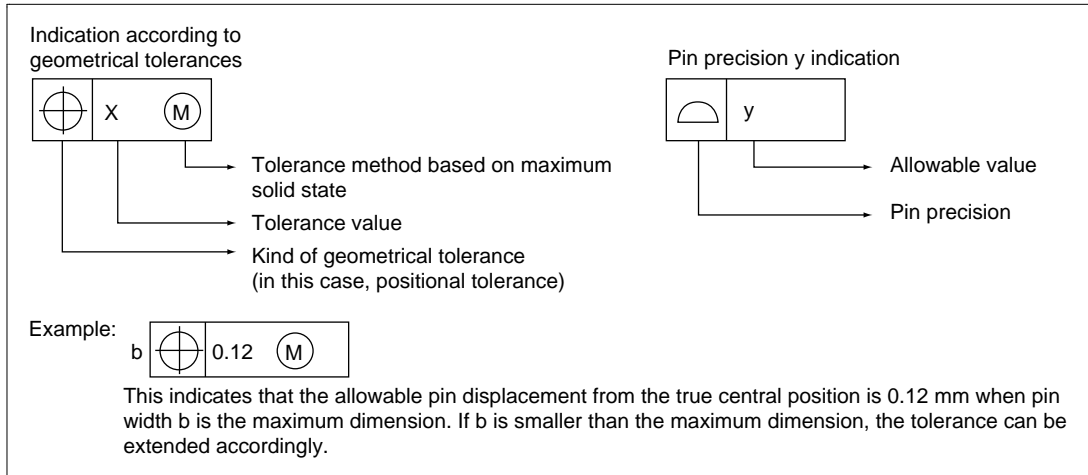
Table 5.4 HI Series OS Product Lineup

Host Machine	IBM PC FLORA (DOS/V) PC-9800 Series	SPARC (Sun OS)	SPARC (Solaris)	HP9000/700 HITACHI 9000
HI Series OS	HSS008ITIN2SFB (for object contract)	HSS008ITCN2SMB (for object contract)	HSS008ITCN2SMB (for object contract)	HSS008ITHN2STB (for object contract)
	HSS008ITIN2SFS (for source contract)	HSS008ITCN2SMS (for source contract)	HSS008ITCN2SMS (for source contract)	HSS008ITHN2STS (for source contract)
Multitask debugger	HSS008IHIN1SF*	HSS008IHIN1SF*	HSS008IHIN1SF*	HSS008IHIN1SF*
Note: * Under development				

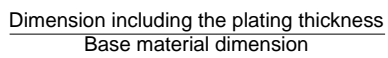
Appendix

A.1 Packages

Package Dimension Diagrams (Unit: mm)

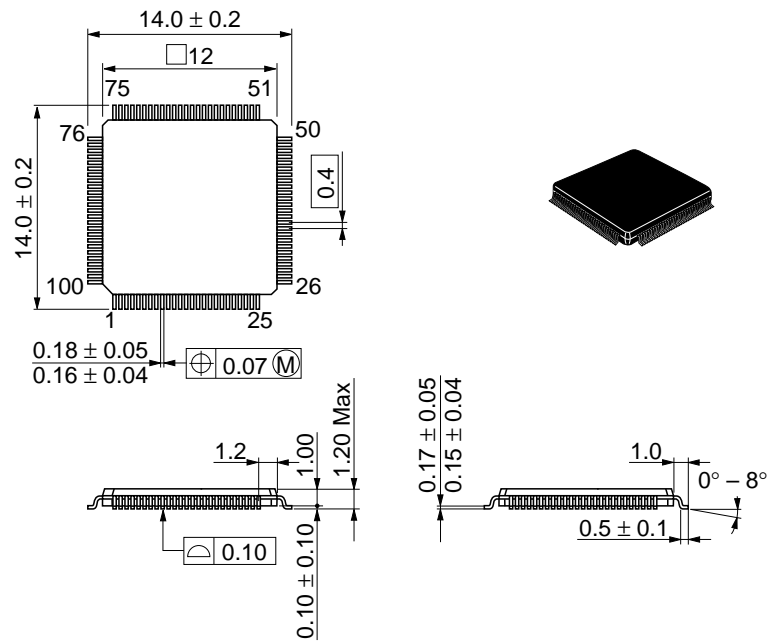


Unit: mm



TFP-100G

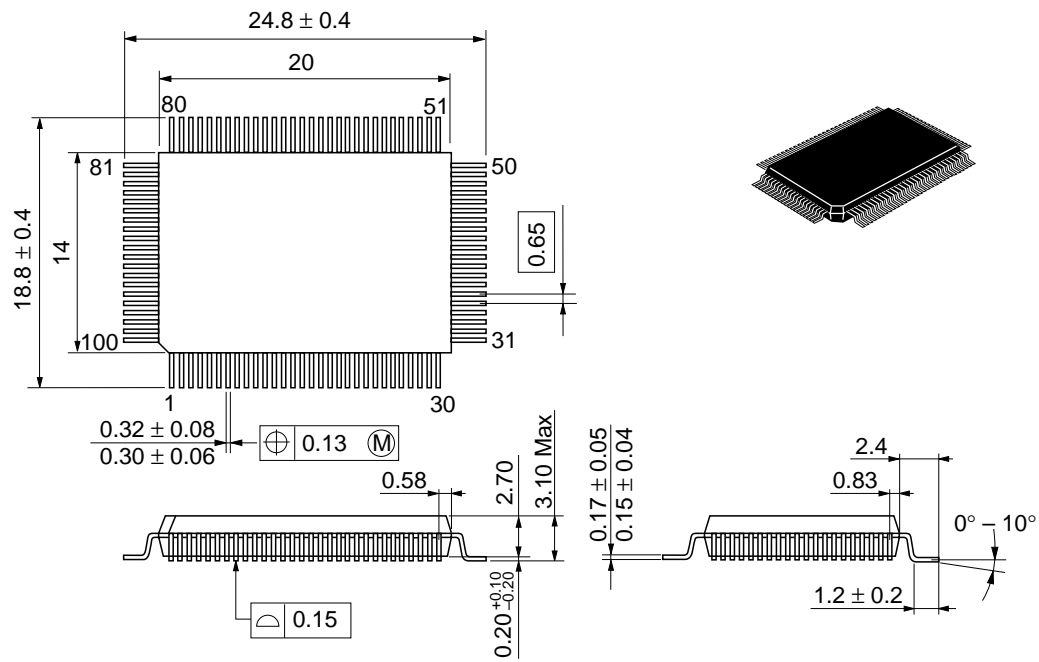
Unit: mm



Dimension including the plating thickness
Base material dimension

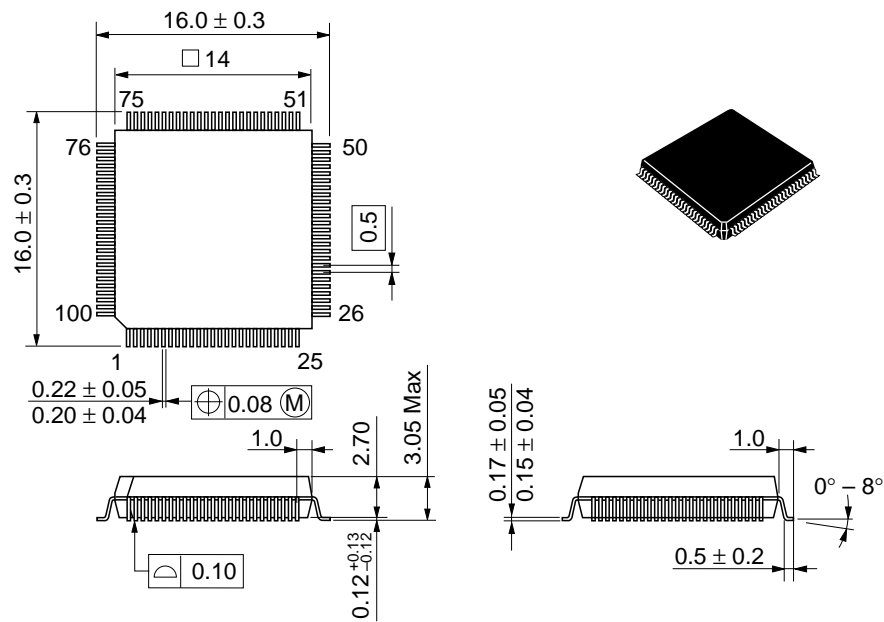
FP-100A

Unit: mm



FP-100B

Unit: mm



Dimension including the plating thickness
Base material dimension

H8S/2237 Series, H8S/2227 Series Overview

Publication Date: 1st Edition, March 1998

Published by: Electronic Devices Business Group
Hitachi, Ltd.

Edited by: Technical Documentation Center
Hitachi Microcomputer System Ltd.

Copyright © Hitachi, Ltd., 1998. All rights reserved. Printed in Japan.