Hitachi Single-Chip Microcomputer
H8S/2355

Series Overview

HITACHI

ADE-802-215

December, 1996

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.

Table Of Contents

Section 3 Peripheral Functions 57

Appendix 133

Section 1 H8S/2355 Series Features

1.1 H8S/2355 Series Functions

H8S/2355 Series microcomputers are designed for faster instruction execution, using a realtime control oriented CPU with an internal 32bit architecture, and can run programs based on the C high-level language efficiently. As well as large-capacity ROM and RAM, these microcomputers include on-chip peripheral functions needed for control systems. These functions simplify the implementation of sophisticated, high-performance systems.

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 realtime control
  - 20 MHz maximum operating frequency (20 MHz oscillation frequency)
  - High-speed arithmetic operations

    8/16/32-bit register-register add/subtract: 50 ns
    16 × 16-bit register-register multiply: 1000 ns
    32 ÷ 16-bit register-register divide: 1000 ns
- Instruction set suitable for high-speed operation
  - Sixty-five basic instructions
  - 8/16/32-bit move/arithmetic and logic instructions
  - Unsigned/signed multiply and divide instructions
  - Powerful bit-manipulation instructions
- Two CPU operating modes
  - Normal mode: H8/300 Series compatible, maximum 64-kbyte address space
  - Advanced mode: Maximum 16-Mbyte address space

On-Chip Byte PROM (Mask ROM)

- 64 kbytes or 128 kbytes

On-Chip High-Speed Static RAM

- 2 kbytes or 4 kbytes

On-Chip Bus Controller

- Address space divided into 8 areas, with bus specifications settable independently for each area
- Chip select output possible for each area
- 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 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 16-bit timer on-chip
- Pulse I/O processing capability for up to 16 pins’
- 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

On-Chip Watchdog Timer (WDT)

- Watchdog timer or interval timer selectable

Three On-Chip Serial Communication Interface (SCI) Channels

- Asynchronous mode or synchronous mode selectable
- Multiprocessor communication function
- Smart card interface function

On-Chip A/D Converter

- Resolution: 10 bits
- Input: 8 channels
- High-speed conversion : 6.7 µs minimum conversion time (at 20 MHz operation)
- Single or scan mode selectable
- Sample and hold circuit
- A/D conversion can be activated by external trigger or timer trigger

On-Chip D/A Converter

- Resolution: 8 bits
- Output: 2 channels

Thirteen I/O Ports

- 87 I/O pins, 8 input-only pins

On-Chip Interrupt Controller

- Nine external interrupt pins (NMI, $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ7}}$)
- 47 internal interrupt sources
- Selection of two interrupt control modes

Power-Down State

- Medium-speed mode
- Sleep mode
- Module stop mode
- Software standby mode
- Hardware standby mode

Seven MCU Operating Modes

				External Data Bus	
Mode	**CPU Operating Mode**	**Description**	**On-Chip ROM**	**Initial Value**	**Maximum Value**
1	Normal	On-chip ROM disabled expansion mode	Disabled	8 bits	16 bits
2		On-chip ROM enabled expansion mode	Enabled	8 bits	16 bits
3		Single-chip mode	Enabled	—	
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

- 120-pin plastic TQFP (TFP-120)
- 128-pin plastic QFP (FP-128)

Product Lineup

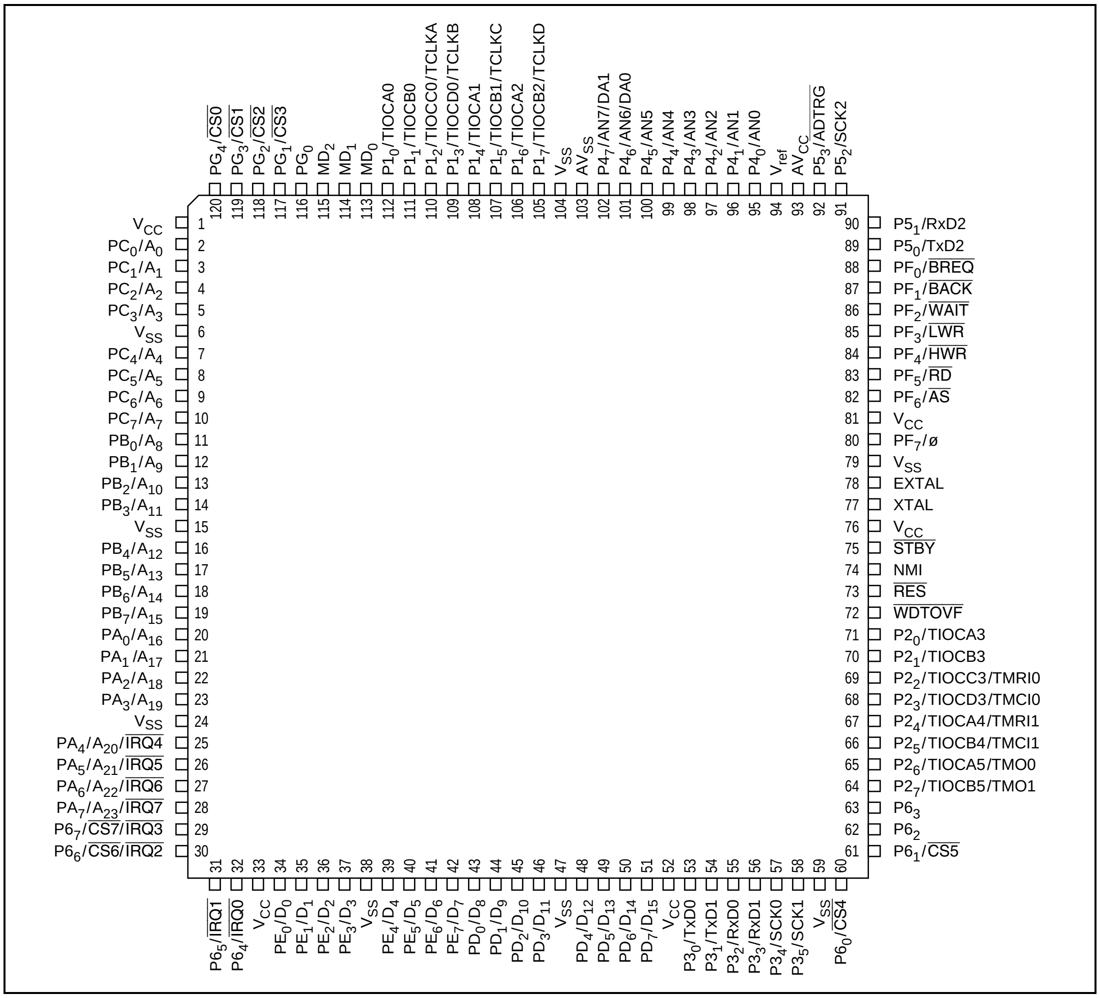
Model			
Mask ROM Version	**ZTAT™ Version**	**ROM/RAM (Bytes)**	**Packages**
HD6432355	HD6472355	128 k/4 k	TFP-120 FP-128
HD6432353*	—	64 k/2 k	TFP-120 FP-128

Note: * In the plannning stage

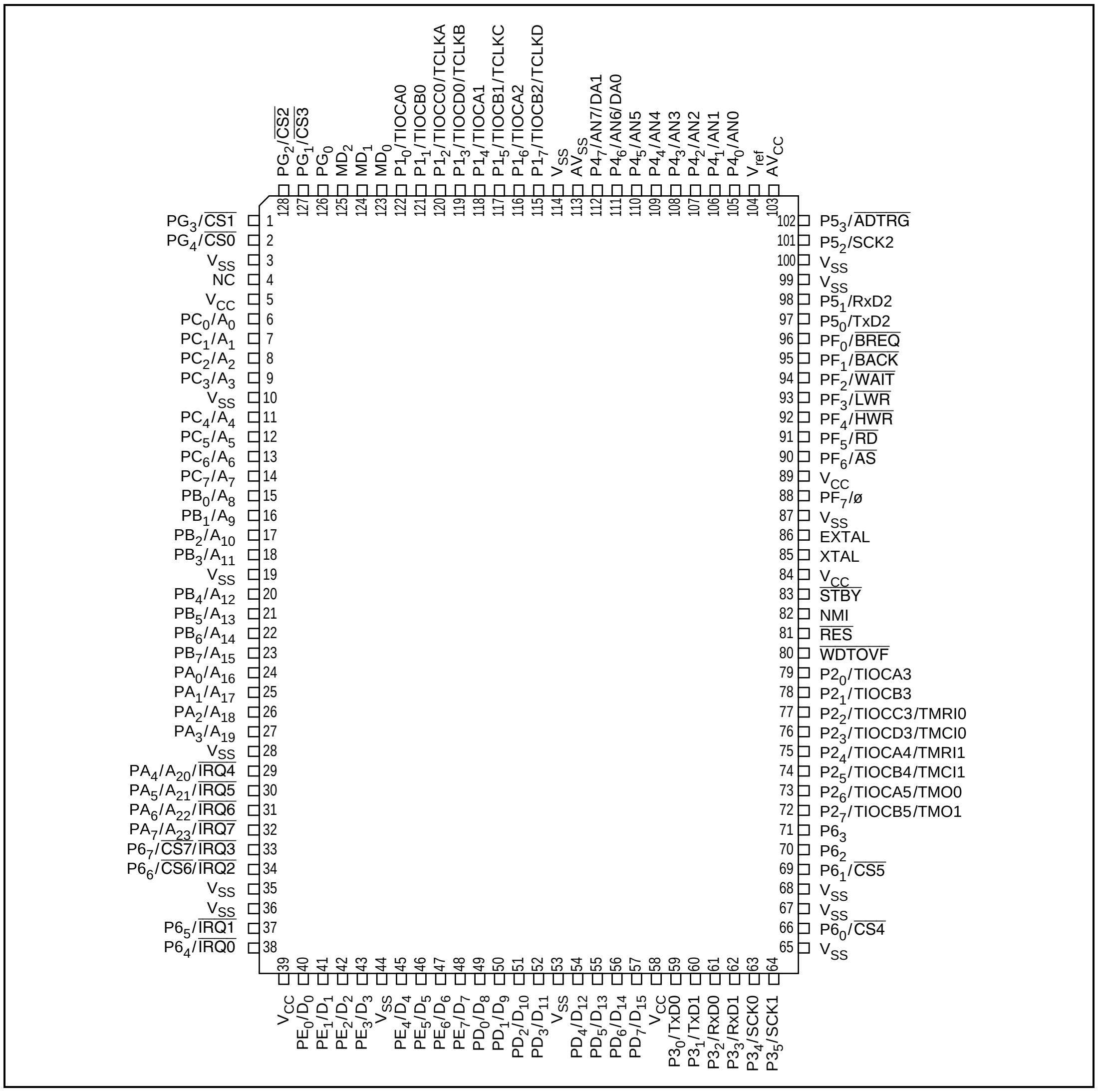
ZTAT™ is a trademark of Hitachi Ltd.

1.2 Pin Description

Pin Arrangement



120-Pin Plastic TQFP (TFP-120: Top View)



128-Pin Plastic QFP (FP-128: Top View)

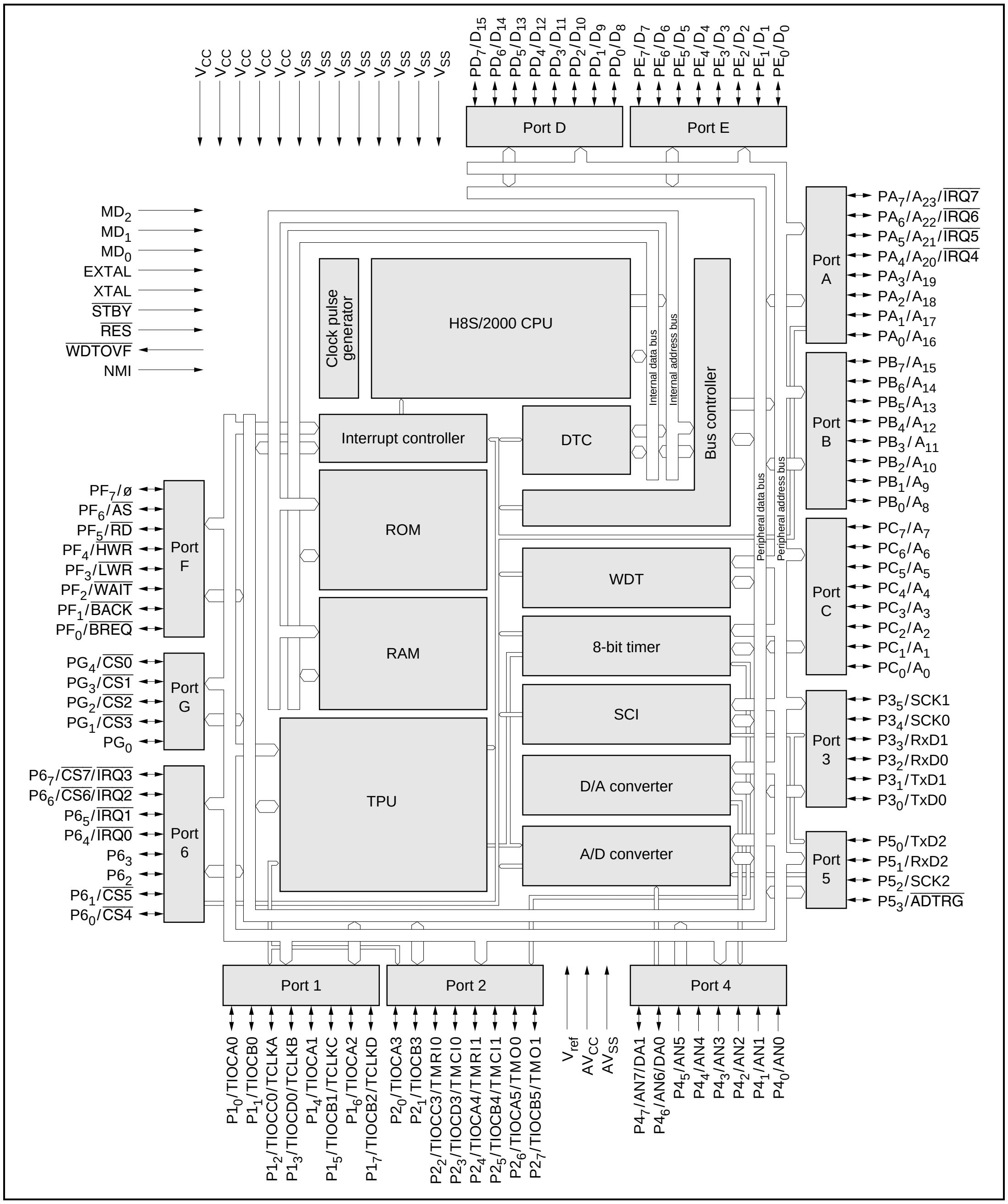
Pin Functions

Type	Symbol	I/O	Name and Function
Power	V_{CC}	Input	Power supply
	V_{SS}	Input	Ground: All V_{SS} 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.
	ϕ	Output	System clock
Operating mode control	MD_2 to MD_0	Input	Mode pins
System control	$\overline{\text{RES}}$	Input	Reset input
	$\overline{\text{STBY}}$	Input	Standby
	$\overline{\text{BREQ}}$	Input	Bus request
	$\overline{\text{BACK}}$	Output	Bus request acknowledge
Interrupts	NMI	Input	Nonmaskable interrupt
	$\overline{\text{IRQ7}}$ to $\overline{\text{IRQ0}}$	Input	Interrupt request 7 to 0
Address bus	A_{23} to A_0	Output	Address bus
Data bus	D_{15} to D_0	I/O	Data bus
Bus control	$\overline{\text{CS7}}$ to $\overline{\text{CS0}}$	Output	Chip select/low address strobe (CS5 to CS2)
	$\overline{\text{AS}}$	Output	Address strobe
	$\overline{\text{RD}}$	Output	Read
	$\overline{\text{HWR}}$	Output	High write/write enable
	$\overline{\text{LWR}}$	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
	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:** The compare match output pins.
	TMCI0, TMCI1	Input	**Counter external clock input:** Input pins for the external clock input to the counter.
	TMRI0, TMRI1	Input	**Counter external reset input:** The counter reset input pins.
Watchdog timer (WDT)	$\overline{\text{WDTOVF}}$	Output	Watchdog timer overflows
Serial communication interface (SCI) Smart Card interface	TxD2, TxD1, TxD0	Output	Transmit data (channel 2, 1, 0)
	RxD2, RxD1, RxD0	Input	Receive data (channel 2, 1, 0)
	SCK2, SCK1, SCK0	I/O	Serial clock (channel 2, 1, 0)
A/D converter	AN7 to AN0	Input	Analog input
	$\overline{\text{ADTRG}}$	Input	A/D conversion external trigger input
D/A converter	DA1, DA0	Output	Analog output
A/D converter and D/A converters	AV_{CC}	Input	This is the power supply pin for the A/D converter and D/A converter.
	AV_{SS}	Input	This is the ground pin for the A/D converter and D/A converter.
	V_{ref}	Input	This is the reference voltage input pin for the A/D converter and D/A converter.
I/O ports	$P1_7$ to $P1_0$	I/O	Port 1
	$P2_7$ to $P2_0$	I/O	Port 2
	$P3_5$ to $P3_0$	I/O	Port 3
	$P4_7$ to $P4_0$	Input	Port 4
	$P5_3$ to $P5_0$	I/O	Port 5
	$P6_7$ to $P6_0$	I/O	Port 6

Type	Symbol	I/O	Name and Function
	PA_7 to PA_0	I/O	Port A
	PB_7 to PB_0	I/O	Port B
	PC_7 to PC_0	I/O	Port C
	PD_7 to PD_0	I/O	Port D
	PE_7 to PE_0	I/O	Port E
	PF_7 to PF_0	I/O	Port F
	PG_4 to PG_0	I/O	Port G

1.3 Block Diagram



Block Diagram

Section 2 CPU

2.1 Features

The H8S/2000 CPU is a high-speed central processing unit with an internal 32-bit architecture, and 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 access space, and is ideal for realtime control.

Feature

- 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: 20 MHz
  - 8/16/32-bit register-register add/subtract: 50 ns
  - 8 × 8-bit register-register multiply: 600 ns

  - 16 ÷ 8-bit register-register divide: 600 ns
  - 16 × 16-bit register-register multiply: 1000 ns
  - 32 ÷ 16-bit register-register divide: 1000 ns
- Two CPU operating modes
  - Normal mode/advanced mode
- Low-power state
  - Transition to power-down state by SLEEP instruction
  - CPU clock speed selectable

Differences between the H8S/2600 CPU and the H8S/2000 CPU

- Register configuration
  - The MAC register is supported only by the H8S/2600 CPU.
- Basic instructions
  - The MAC, CLRMAC, LDMAC, and STMAC instructions are supported only by the H8S/2600 CPU.
- Number of states required for execution
  - The number of states required for execution of the MULXU and MULXS instructions

Differences from H8/300 CPU

In comparison with the H8/300 CPU, the H8S/2000 CPU has the following enhancements.

- More general registers and control registers
  - Eight 16-bit registers and one 8-bit control registers 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
  - For effective use of the 16-Mbyte address space
- Enhanced instructions
  - Addressing modes of bit-manipulation instructions enhanced
  - Signed multiply and divide instructions added
  - Two-bit shift instructions added
  - Instructions for saving and restoring multiple registers added
  - Test-and-set instruction added
- Higher speed
  - Basic instructions execute twice as fast

Differences from H8/300H CPU

In comparison with the H8/300H CPU, the H8S/2000 CPU has the following enhancements.

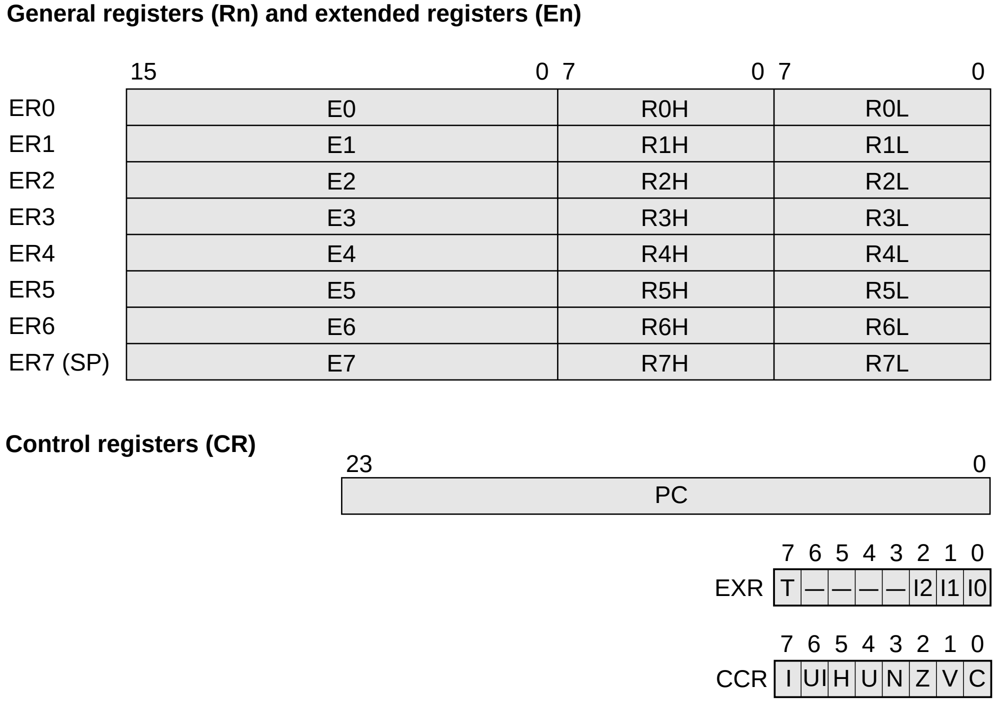
- Additional control register
  - One 8-bit control register added
- Enhanced instructions
  - Addressing modes of bit-manipulation instructions enhanced
  - Two-bit shift instructions added
  - Instructions for saving and restoring multiple registers added
  - Test-and-set instruction added
- Higher 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 extend register (EXR), and 8-bit condition code register (CCR).

CPU Internal Register Configuration



Legend

SP:	Stack pointer	H:	Half-carry flag
PC:	Program counter	U:	User bit
EXR:	Extend register	N:	Negative flag
T:	Trace bit	Z:	Zero flag
I2 to I0:	Interrupt mask bits	V:	Overflow flag
CCR:	Condition code register	C:	Carry flag
I:	Interrupt mask bit		
UI:	User bit/interrupt mask bit*		

Note: * In the H8S/2355 Series, this bit cannot be used as an interrupt mask.

General Registers

The CPU has eight 32-bit general registers. These general registers are all functionally alike and can be used as either address registers or 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.

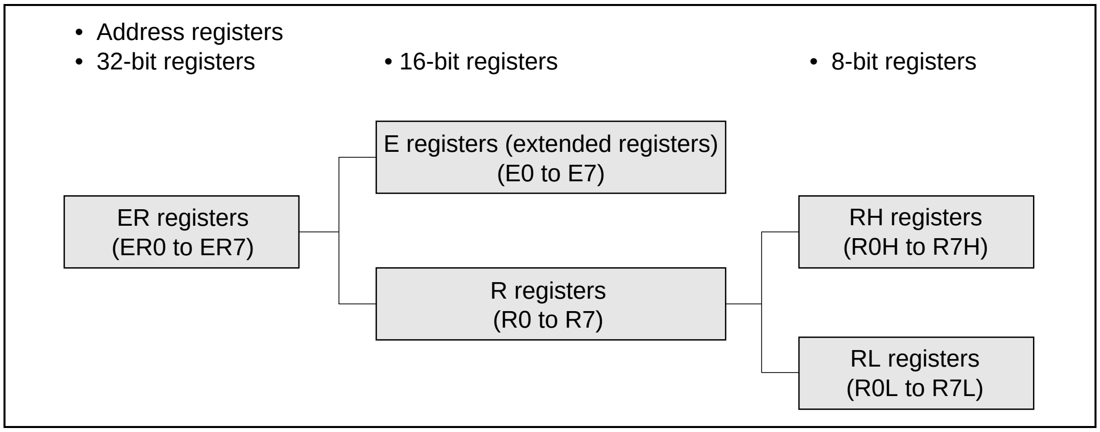
When the general registers are used as 32-bit registers or address registers, they are designated by the letters ER (ER0 to ER7).

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.

Usage of General Registers



Control Registers

The control registers are the 24-bit program counter (PC), 8-bit extend 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) or a multiple of 2 bytes, 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 the interrupt mask bit (I), and the 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 at the start of an exception-handling sequence. See section 2.9, Interrupts for details.
- Bit 6—User Bit or Interrupt Mask Bit (UI)

  Can be written or read by software using the LDC, STC, ANDC, ORC, and XORC instructions. In this IC, 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 value shifted out of the end bit

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

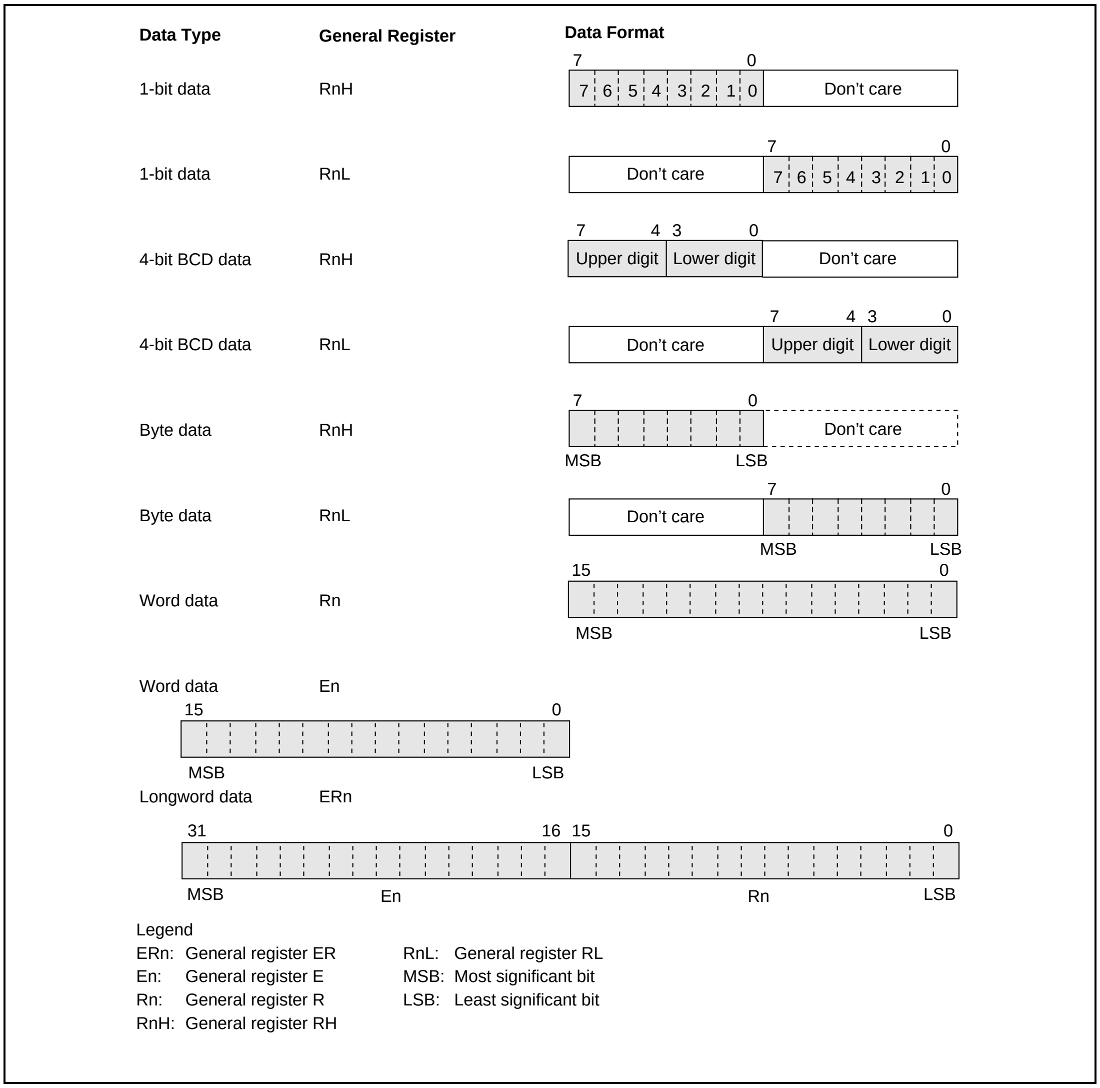
2.3 Data Formats

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

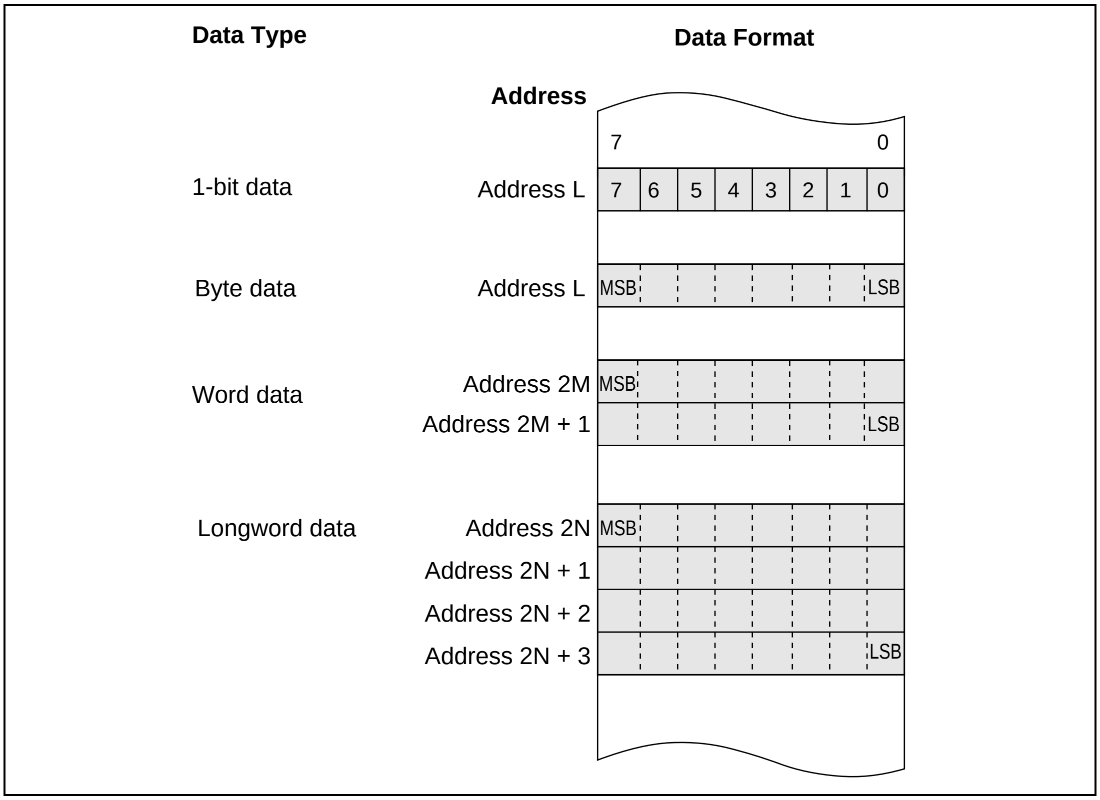
Bit-manipulation instructions operate on 1-bit data by accessing bit n (n = 0, 1, 2, ..., 7) of byte operand data.

The DAA and DAS decimal-adjust instructions treat byte data as two digits of 4-bit BCD data.

General Register Data Formats



Memory Data Formats



2.4 Addressing Modes

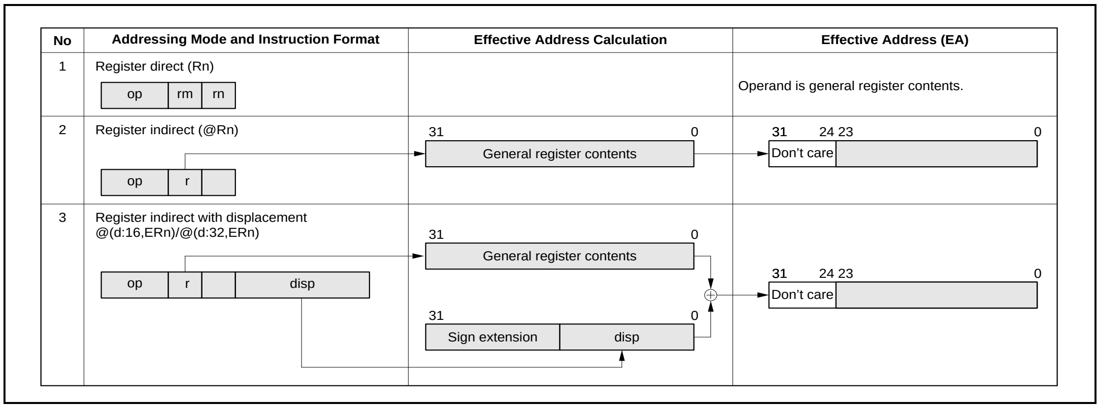
The H8S/2000 CPU supports eight addressing modes.

Addressing Modes

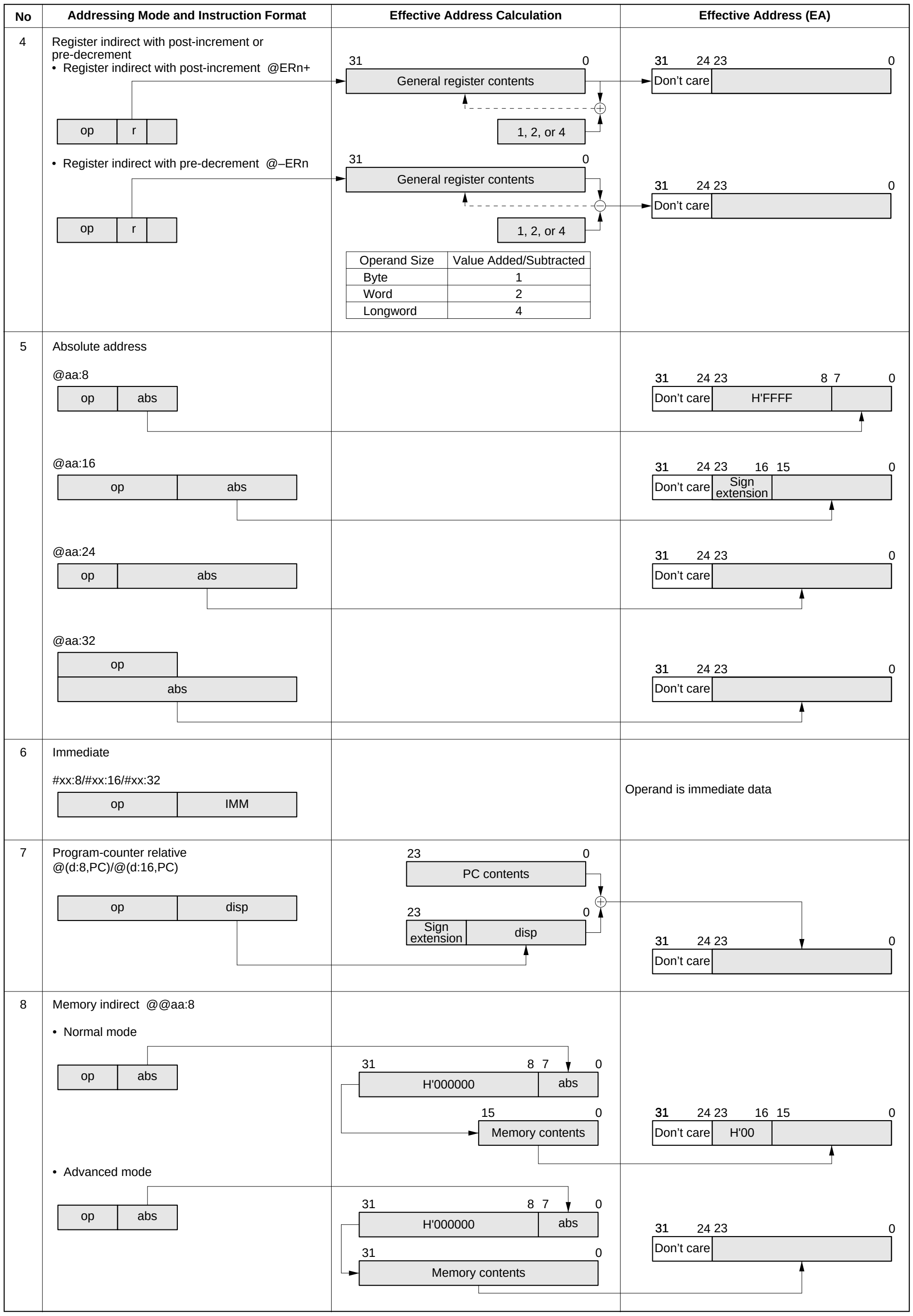
No.	Addressing Mode	Symbol
1	Register direct	Rn
2	Register indirect	@ERn
3	Register indirect with displacement	@(d:16,ERn)/@(d:32,ERn)
4	Register indirect with post-increment Register indirect with pre-decrement	@ERn+ @–ERn
5	Absolute address	@aa:8/@aa:16/@aa:24/@aa:32
6	Immediate	#xx:8/#xx:16/#xx:32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

Effective Address (EA) Calculation

In normal mode, the upper 8 bits of the effective address are ignored in order to generate a 16-bit effective address.



No	Addressing Mode and Instruction Format	Effective Address Calculation	Effective Address (EA)
1	Register direct (Rn) op | rm | rn		Operand is general register contents.
2	Register indirect (@Rn) op | r |	31 0 General register contents	31 24 23 0 Don’t care |
3	Register indirect with displacement @(d:16,ERn)/@(d:32,ERn) op | r | | disp	31 0 General register contents 31 0 Sign extension | disp	31 24 23 0 Don’t care |



Operand Size	Value Added/Subtracted
Byte	1
Word	2
Longword	4

2.5 Instruction Set

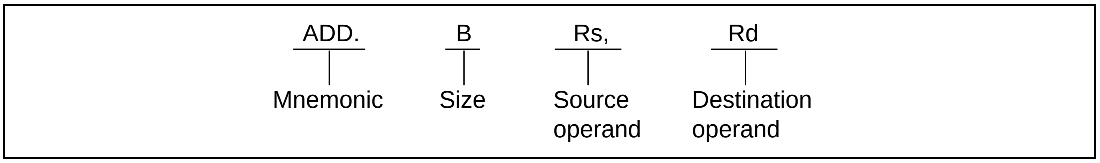
The H8S/2000 CPU has 65 types of instructions.

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.



Instruction Set Table

1. Data transfer instructions

	Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States*1	
			#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
MOV	MOV.B #xx:8,Rd	B	2									#xx:8→Rd8	—	—	Δ	Δ	0	—	1	
	MOV.B Rs,Rd	B		2								Rs8→Rd8	—	—	Δ	Δ	0	—	1	
	MOV.B @ERs,Rd	B			2							@ERs→Rd8	—	—	Δ	Δ	0	—	2	
	MOV.B @(d:16,ERs),Rd	B				4						@(d:16,ERs)→Rd8	—	—	Δ	Δ	0	—	3	
	MOV.B @(d:32,ERs),Rd	B				8						@(d:32,ERs)→Rd8	—	—	Δ	Δ	0	—	5	
	MOV.B @ERs+,Rd	B					2					@ERs→Rd8,ERs32+1→ERs32	—	—	Δ	Δ	0	—	3	
	MOV.B @aa:8,Rd	B						2				@aa:8→Rd8	—	—	Δ	Δ	0	—	2	
	MOV.B @aa:16,Rd	B						4				@aa:16→Rd8	—	—	Δ	Δ	0	—	3	
	MOV.B @aa:32,Rd	B						6				@aa:32→Rd8	—	—	Δ	Δ	0	—	4	
	MOV.B Rs,@ERd	B			2							Rs8→@ERd	—	—	Δ	Δ	0	—	2	
	MOV.B Rs,@(d:16,ERd)	B				4						Rd8→@(d:16,ERd)	—	—	Δ	Δ	0	—	3	
	MOV.B Rs,@(d:32,ERd)	B				8						Rd8→@(d:32,ERd)	—	—	Δ	Δ	0	—	5	
	MOV.B Rs,@–ERd	B					2					ERd32–1→ERd32,Rs8→@ERd	—	—	Δ	Δ	0	—	3	
	MOV.B Rs,@aa:8	B						2				Rs8→@aa:8	—	—	Δ	Δ	0	—	2	
	MOV.B Rs,@aa:16	B						4				Rs8→@aa:16	—	—	Δ	Δ	0	—	3	
	MOV.B Rs,@aa:32	B						6				Rs8→@aa:32	—	—	Δ	Δ	0	—	4	
	MOV.W #xx:16,Rd	W	4									#xx:16→Rd16	—	—	Δ	Δ	0	—	2	
	MOV.W Rs,Rd	W		2								Rs16→Rd16	—	—	Δ	Δ	0	—	1	
	MOV.W @ERs,Rd	W			2							@ERs→Rd16	—	—	Δ	Δ	0	—	2	
	MOV.W @(d:16,ERs),Rd	W				4						@(d:16,ERs)→Rd16	—	—	Δ	Δ	0	—	3	
	MOV.W @(d:32,ERs),Rd	W				8						@(d:32,ERs)→Rd16	—	—	Δ	Δ	0	—	5	
	MOV.W @ERs+,Rd	W					2					ERs→Rd16,ERs32+2→ERs32	—	—	Δ	Δ	0	—	3	
	MOV.W @aa:16,Rd	W						4				@aa:16→Rd16	—	—	Δ	Δ	0	—	3	
	MOV.W @aa:32,Rd	W						6				@aa:32→Rd16	—	—	Δ	Δ	0	—	4	
	MOV.W Rs,@ERd	W			2							Rs16→@ERd	—	—	Δ	Δ	0	—	2	
	MOV.W Rs,@(d:16,ERd)	W				4						Rs16→@(d:16,ERd)	—	—	Δ	Δ	0	—	3	
	MOV.W Rs,@(d:32,ERd)	W				8						Rs16→@(d:32,ERd)	—	—	Δ	Δ	0	—	5	
	MOV.W Rs,@–ERd	W					2					ERd32–2→ERd32,Rs16→@ERd	—	—	Δ	Δ	0	—	3	
	MOV.W Rs,@aa:16	W						4				Rs16→@aa:16	—	—	Δ	Δ	0	—	3	
	MOV.W Rs,@aa:32	W						6				Rs16→@aa:32	—	—	Δ	Δ	0	—	4	
	MOV.L #xx:32,ERd	L	6									#xx:32→ERd32	—	—	Δ	Δ	0	—	3	
	MOV.L ERs,ERd	L		2								ERs32→ERd32	—	—	Δ	Δ	0	—	1	
	MOV.L @ERs,ERd	L			4							@ERs→ERd32	—	—	Δ	Δ	0	—	4	
	MOV.L @(d:16,ERs),ERd	L				6						@(d:16,ERs)→ERd32	—	—	Δ	Δ	0	—	5	
	MOV.L @(d:32,ERs),ERd	L				10						@(d:32,ERs)→ERd32	—	—	Δ	Δ	0	—	7	
	MOV.L @ERs+,ERd	L					4					@ERs→ERd32,ERs32+4→ @ERs32	—	—	Δ	Δ	0	—	5	
	MOV.L @aa:16,ERd	L						6				@aa:16→ERd32	—	—	Δ	Δ	0	—	5	
	MOV.L @aa:32,ERd	L						8				@aa:32→ERd32	—	—	Δ	Δ	0	—	6	

	Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States*1	
			#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
MOV	MOV.L ERs,@ERd	L			4							ERs32→@ERd	—	—	Δ	Δ	0	—	4	
	MOV.L ERs,@(d:16,ERd)	L				6						ERs32→@(d:16,ERd)	—	—	Δ	Δ	0	—	5	
	MOV.L ERs,@(d:32,ERd)	L				10						ERs32→@(d:32,ERd)	—	—	Δ	Δ	0	—	7	
	MOV.L ERs,@–ERd	L					4					ERd32–4→ERd32,ERs32→@ERd	—	—	Δ	Δ	0	—	5	
	MOV.L ERs,@aa:16	L						6				ERs32→@aa:16	—	—	Δ	Δ	0	—	5	
	MOV.L ERs,@aa:32	L						8				ERs32→@aa:32	—	—	Δ	Δ	0	—	6	
POP	POP.W Rn	W									2	@SP→Rn16,SP+2→SP	—	—	Δ	Δ	0	—	3	
	POP.L ERn	L									4	@SP→ERn32,SP+4→SP	—	—	Δ	Δ	0	—	5	
PUSH	PUSH.W Rn	W									2	SP–2→SP,Rn16→@SP	—	—	Δ	Δ	0	—	3	
	PUSH.L ERn	L									4	SP–4→SP,ERn32→@SP	—	—	Δ	Δ	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]	
MOVFPE	MOVFPE @aa:16,Rd	Cannot be used in the H8S/2355 Series																	[2]	
MOVTPE	MOVTPE Rs,@aa:16																		[2]	

2. Arithmetic instructions

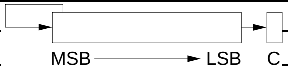
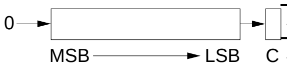
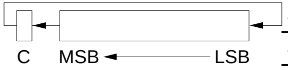
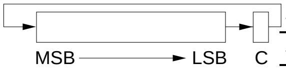
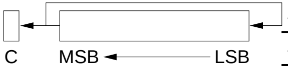
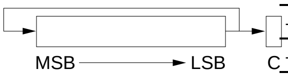
Mnemonic			Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States*1	
		Operand Size	#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
ADD	ADD.B #xx:8,Rd	B	2									Rd8+#xx:8→Rd8	—	Δ	Δ	Δ	Δ	Δ	1	
	ADD.B Rs,Rd	B		2								Rd8+Rs8→Rd8	—	Δ	Δ	Δ	Δ	Δ	1	
	ADD.W #xx:16,Rd	W	4									Rd16+#xx:16→Rd16	—	[3]	Δ	Δ	Δ	Δ	2	
	ADD.W Rs,Rd	W		2								Rd16+Rs16→Rd16	—	[3]	Δ	Δ	Δ	Δ	1	
	ADD.L #xx:32,ERd	L	6									ERd32+#xx:32→ERd32	—	[4]	Δ	Δ	Δ	Δ	3	
	ADD.L ERs,ERd	L		2								ERd32+ERs32→ERd32	—	[4]	Δ	Δ	Δ	Δ	1	
ADDX	ADDX #xx:8,Rd	B	2									Rd8+#xx:8+C→Rd8	—	Δ	Δ	[5]	Δ	Δ	1	
	ADDX Rs,Rd	B		2								Rd8+Rs8+C→Rd8	—	Δ	Δ	[5]	Δ	Δ	1	
ADDS	ADDS #1,ERd	L		2								ERd32+1→ERd32	—	—	—	—	—	—	1	
	ADDS #2,ERd	L		2								ERd32+2→ERd32	—	—	—	—	—	—	1	
	ADDS #4,ERd	L		2								ERd32+4→ERd32	—	—	—	—	—	—	1	
INC	INC.B Rd	B		2								Rd8+1→Rd8	—	—	Δ	Δ	Δ	—	1	
	INC.W #1,Rd	W		2								Rd16+1→Rd16	—	—	Δ	Δ	Δ	—	1	
	INC.W #2,Rd	W		2								Rd16+2→Rd16	—	—	Δ	Δ	Δ	—	1	
	INC.L #1,ERd	L		2								ERd32+1→ERd32	—	—	Δ	Δ	Δ	—	1	
	INC.L #2,ERd	L		2								ERd32+2→ERd32	—	—	Δ	Δ	Δ	—	1	
DAA	DAA Rd	B		2								Rd8 decimal adjust → Rd8	—	*	Δ	Δ	*	Δ	1	
SUB	SUB.B Rs,Rd	B		2								Rd8–Rs8→Rd8	—	Δ	Δ	Δ	Δ	Δ	1	
	SUB.W #xx:16,Rd	W	4									Rd16–#xx:16→Rd16	—	[3]	Δ	Δ	Δ	Δ	2	
	SUB.W Rs,Rd	W		2								Rd16–Rs16→Rd16	—	[3]	Δ	Δ	Δ	Δ	1	
	SUB.L #xx:32,ERd	L	6									ERd32–#xx:32→ERd32	—	[4]	Δ	Δ	Δ	Δ	3	
	SUB.L ERs,ERd	L		2								ERd32–ERs32→ERd32	—	[4]	Δ	Δ	Δ	Δ	1	
SUBX	SUBX #xx:8,Rd	B	2									Rd8–#xx:8–C→Rd8	—	Δ	Δ	[5]	Δ	Δ	1	
	SUBX Rs,Rd	B		2								Rd8–Rs8–C→Rd8	—	Δ	Δ	[5]	Δ	Δ	1	
SUBS	SUBS #1,ERd	L		2								ERd32–1→ERd32	—	—	—	—	—	—	1	
	SUBS #2,ERd	L		2								ERd32–2→ERd32	—	—	—	—	—	—	1	
	SUBS #4,ERd	L		2								ERd32–4→ERd32	—	—	—	—	—	—	1	
DEC	DEC.B Rd	B		2								Rd8–1→Rd8	—	—	Δ	Δ	Δ	—	1	
	DEC.W #1,Rd	W		2								Rd16–1→Rd16	—	—	Δ	Δ	Δ	—	1	
	DEC.W #2,Rd	W		2								Rd16–2→Rd16	—	—	Δ	Δ	Δ	—	1	
	DEC.L #1,ERd	L		2								ERd32–1→ERd32	—	—	Δ	Δ	Δ	—	1	
	DEC.L #2,ERd	L		2								ERd32–2→ERd32	—	—	Δ	Δ	Δ	—	1	
DAS	DAS Rd	B		2								Rd8 decimal adjust → Rd8	—	*	Δ	Δ	*	—	1	
MULXU	MULXU.B Rs,Rd	B		2								Rd8×Rs8→Rd16 (unsigned multiplication)	—	—	—	—	—	—	12	
	MULXU.W Rs,ERd	W		2								Rd16×Rs16→ERd32 (unsigned multiplication)	—	—	—	—	—	—	20	
MULXS	MULXS.B Rs,Rd	B		4								Rd8×Rs8→Rd16 (signed multiplication)	—	—	Δ	Δ	—	—	13	
	MULXS.W Rs,ERd	W		4								Rd16×Rs16→ERd32 (signed multiplication)	—	—	Δ	Δ	—	—	21	

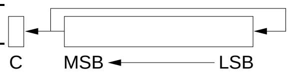
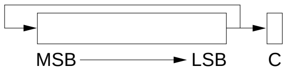
			Addressing Mode/Instruction Length (Bytes)										Condition Code						No. of States*1	
	Mnemonic	Operand Size	#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—	Operation	I	H	N	Z	V	C	Normal	Advanced
DIVXU	DIVXU.B Rs,Rd	B		2								Rd16÷Rs8→Rd16 (RdH: remainder, RdL: quotient) (unsigned division)	—	—	[6]	[7]	—	—	12	
	DIVXU.W Rs,ERd	W		2								ERd32÷Rs16→ERd32 (Ed: remainder, Rd: quotient) (unsigned division)	—	—	[6]	[7]	—	—	20	
DIVXS	DIVXS.B Rs,Rd	B		4								Rd16÷Rs8→Rd16 (RdH: remainder, RdL: quotient) (signed division)	—	—	[8]	[7]	—	—	13	
	DIVXS.W Rs,ERd	W		4								ERd32÷Rs16→ERd32 (Ed: remainder, Rd: quotient) (signed division)	—	—	[8]	[7]	—	—	21	
CMP	CMP.B #xx:8,Rd	B	2									Rd8–#xx:8	—	Δ	Δ	Δ	Δ	Δ	1	
	CMP.B Rs,Rd	B		2								Rd8–Rs8	—	Δ	Δ	Δ	Δ	Δ	1	
	CMP.W #xx:16,Rd	W	4									Rd16–#xx:16	—	[3]	Δ	Δ	Δ	Δ	2	
	CMP.W Rs,Rd	W		2								Rd16–Rs16	—	[3]	Δ	Δ	Δ	Δ	1	
	CMP.L #xx:32,ERd	L	6									ERd32–#xx:32	—	[4]	Δ	Δ	Δ	Δ	3	
	CMP.L ERs,ERd	L		2								ERd32–ERs32	—	[4]	Δ	Δ	Δ	Δ	1	
NEG	NEG.B Rd	B		2								0–Rd8→Rd8	—	Δ	Δ	Δ	Δ	Δ	1	
	NEG.W Rd	W		2								0–Rd16→Rd16	—	Δ	Δ	Δ	Δ	Δ	1	
	NEG.L ERd	L		2								0–ERd32→ERd32	—	Δ	Δ	Δ	Δ	Δ	1	
EXTU	EXTU.W Rd	W		2								0 → (<bits 15 to 8> of Rd16)	—	—	0	Δ	0	—	1	
	EXTU.L ERd	L		2								0 → (<bits 31 to 16> of ERd32)	—	—	0	Δ	0	—	1	
EXTS	EXTS.W Rd	W		2								(<bit 7> of Rd16) → (<bits 15 to 8> of Rd16)	—	—	Δ	Δ	0	—	1	
	EXTS.L ERd	L		2								(<bit 15> of ERd32) → (<bits 31 to 16> of ERd32)	—	—	Δ	Δ	0	—	1	
TAS	TAS @ERd	B			4							@ERd–0 → CRR set, (1) → (<bit 7> of @ERd)	—	—	Δ	Δ	0	—	4	

3. Logical instructions

	Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States*1	
			#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
AND	AND.B #xx:8,Rd	B	2									Rd8∧#xx:8→Rd8	—	—	∆	∆	0	—	1	
	AND.B Rs,Rd	B		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								Rd16∧Rs16→Rd16	—	—	∆	∆	0	—	1	
	AND.L #xx:32,ERd	L	6									ERd32∧#xx:32→ERd32	—	—	∆	∆	0	—	3	
	AND.L ERs,ERd	L		4								ERd32∧ERs32→ERd32	—	—	∆	∆	0	—	2	
OR	OR.B #xx:8,Rd	B	2									Rd8∨#xx:8→Rd8	—	—	∆	∆	0	—	1	
	OR.B Rs,Rd	B		2								Rd8∨Rs8→Rd8	—	—	∆	∆	0	—	1	
	OR.W #xx:16,Rd	W	4									Rd16∨#xx:16→Rd16	—	—	∆	∆	0	—	2	
	OR.W Rs,Rd	W		2								Rd16∨Rs16→Rd16	—	—	∆	∆	0	—	1	
	OR.L #xx:32,ERd	L	6									ERd32∨#xx:32→ERd32	—	—	∆	∆	0	—	3	
	OR.L ERs,ERd	L		4								ERd32∨ERs32→ERd32	—	—	∆	∆	0	—	2	
XOR	XOR.B #xx:8,Rd	B	2									Rd8⊕#xx:8→Rd8	—	—	∆	∆	0	—	1	
	XOR.B Rs,Rd	B		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								Rd16⊕Rs16→Rd16	—	—	∆	∆	0	—	1	
	XOR.L #xx:32,ERd	L	6									ERd32⊕#xx:32→ERd32	—	—	∆	∆	0	—	3	
	XOR.L ERs,ERd	L		4								ERd32⊕ERs32→ERd32	—	—	∆	∆	0	—	2	
NOT	NOT.B Rd	B		2								← Rd8→Rd8	—	—	∆	∆	0	—	1	
	NOT.W Rd	W		2								← Rd16→Rd16	—	—	∆	∆	0	—	1	
	NOT.L ERd	L		2								← Rd32→Rd32	—	—	∆	∆	0	—	1	

4. Shift instructions

			Addressing Mode/Instruction Length (Bytes)										Condition Code						No. of States[*1]	
	Mnemonic	Operand Size	#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—	Operation	I	H	N	Z	V	C	Normal	Advanced
SHAL	SHAL.B Rd	B		2									—	—	Δ	Δ	Δ	Δ	1	
	SHAL.B #2,Rd	B		2									—	—	Δ	Δ	Δ	Δ	1	
	SHAL.W Rd	W		2									—	—	Δ	Δ	Δ	Δ	1	
	SHAL.W #2,Rd	W		2									—	—	Δ	Δ	Δ	Δ	1	
	SHAL.L ERd	L		2									—	—	Δ	Δ	Δ	Δ	1	
	SHAL.L #2,ERd	L		2									—	—	Δ	Δ	Δ	Δ	1	
SHAR	SHAR.B Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	SHAR.B #2,Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	SHAR.W Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	SHAR.W #2,Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	SHAR.L ERd	L		2									—	—	Δ	Δ	0	Δ	1	
	SHAR.L #2,ERd	L		2									—	—	Δ	Δ	0	Δ	1	
SHLL	SHLL.B Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	SHLL.B #2,Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	SHLL.W Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	SHLL.W #2,Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	SHLL.L ERd	L		2									—	—	Δ	Δ	0	Δ	1	
	SHLL.L #2,ERd	L		2									—	—	Δ	Δ	0	Δ	1	
SHLR	SHLR.B Rd	B		2									—	—	0	Δ	0	Δ	1	
	SHLR.B #2,Rd	B		2									—	—	0	Δ	0	Δ	1	
	SHLR.W Rd	W		2									—	—	0	Δ	0	Δ	1	
	SHLR.W #2,Rd	W		2									—	—	0	Δ	0	Δ	1	
	SHLR.L ERd	L		2									—	—	0	Δ	0	Δ	1	
	SHLR.L #2,ERd	L		2									—	—	0	Δ	0	Δ	1	
ROTXL	ROTXL.B Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTXL.B #2,Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTXL.W Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTXL.W #2,Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTXL.L ERd	L		2									—	—	Δ	Δ	0	Δ	1	
	ROTXL.L #2,ERd	L		2									—	—	Δ	Δ	0	Δ	1	
ROTXR	ROTXR.B Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTXR.B #2,Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTXR.W Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTXR.W #2,Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTXR.L ERd	L		2									—	—	Δ	Δ	0	Δ	1	
	ROTXR.L #2,ERd	L		2									—	—	Δ	Δ	0	Δ	1	

			Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States[*1]	
	Mnemonic	Operand Size	#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
ROTL	ROTL.B Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTL.B #2,Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTL.W Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTL.W #2,Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTL.L ERd	L		2									—	—	Δ	Δ	0	Δ	1	
	ROTL.L #2,ERd	L		2									—	—	Δ	Δ	0	Δ	1	
ROTR	ROTR.B Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTR.B #2,Rd	B		2									—	—	Δ	Δ	0	Δ	1	
	ROTR.W Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTR.W #2,Rd	W		2									—	—	Δ	Δ	0	Δ	1	
	ROTR.L ERd	L		2									—	—	Δ	Δ	0	Δ	1	
	ROTR.L #2,ERd	L		2									—	—	Δ	Δ	0	Δ	1	

5. Bit manipulation instructions

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes) #xx	Rn	@ERn	@(d,ERn)	@−ERn/@ERn+	@aa	@(d,PC)	@@aa	—	Operation	Condition Code I	H	N	Z	V	C	No. of States*1 Normal	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	
	BSET Rn,@aa:32	B						8				(Rn8 of @aa:32)←1	—	—	—	—	—	—	6	
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	
	BCLR Rn,@aa:32	B						8				(Rn8 of @aa:32)←0	—	—	—	—	—	—	6	
BNOT	BNOT #xx:3,Rd	B		2								(#xx:3 of Rd8)← ¬(#xx:3 of Rd8)]	—	—	—	—	—	—	1	
	BNOT #xx:3,@ERd	B			4							(#xx:3 of @ERd)← [¬(#xx:3 of @ERd)]	—	—	—	—	—	—	4	
	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	
	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	

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States^{*1}	
			#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
BTST	BTST #xx:3,Rd	B		2								(#xx:3 of Rd8)→Z	—	—	—	Δ	—	—	1	
	BTST #xx:3,@ERd	B			4							(#xx:3 of @ERd)→Z	—	—	—	Δ	—	—	3	
	BTST #xx:3,@aa:8	B						4				(#xx:3 of @aa:8)→Z	—	—	—	Δ	—	—	3	
	BTST #xx:3,@aa:16	B						6				(#xx:3 of @aa:16)→Z	—	—	—	Δ	—	—	4	
	BTST #xx:3,@aa:32	B						8				(#xx:3 of @aa:32)→Z	—	—	—	Δ	—	—	5	
	BTST Rn,Rd	B		2								(Rn8 of Rd8)→Z	—	—	—	Δ	—	—	1	
	BTST Rn,@ERd	B			4							(Rn8 of @ERd)→Z	—	—	—	Δ	—	—	3	
	BTST Rn,@aa:8	B						4				(Rn8 of @aa:8)→Z	—	—	—	Δ	—	—	3	
	BTST Rn,@aa:16	B						6				(Rn8 of @aa:16)→Z	—	—	—	Δ	—	—	4	
	BTST Rn,@aa:32	B						8				(Rn8 of @aa:32)→Z	—	—	—	Δ	—	—	5	
BLD	BLD #xx:3,Rd	B		2								(#xx:3 of Rd8)→C	—	—	—	—	—	Δ	1	
	BLD #xx:3,@ERd	B			4							(#xx:3 of @ERd)→C	—	—	—	—	—	Δ	3	
	BLD #xx:3,@aa:8	B						4				(#xx:3 of @aa:8)→C	—	—	—	—	—	Δ	3	
	BLD #xx:3,@aa:16	B						6				(#xx:3 of @aa:16)→C	—	—	—	—	—	Δ	4	
	BLD #xx:3,@aa:32	B						8				(#xx:3 of @aa:32)→C	—	—	—	—	—	Δ	5	
BILD	BILD #xx:3,Rd	B		2								¬(#xx:3 of Rd8)→C	—	—	—	—	—	Δ	1	
	BILD #xx:3,@ERd	B			4							¬(#xx:3 of @ERd)→C	—	—	—	—	—	Δ	3	
	BILD #xx:3,@aa:8	B						4				¬(#xx:3 of @aa:8)→C	—	—	—	—	—	Δ	3	
	BILD #xx:3,@aa:16	B						6				¬(#xx:3 of @aa:16)→C	—	—	—	—	—	Δ	4	
	BILD #xx:3,@aa:32	B						8				¬(#xx:3 of @aa:32)→C	—	—	—	—	—	Δ	5	
BST	BST #xx:3,Rd	B		2								C→(#xx:3 of Rd8)	—	—	—	—	—	—	1	
	BST #xx:3,@ERd	B			4							C→(#xx:3 of @ERd24)	—	—	—	—	—	—	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 @ERd24)	—	—	—	—	—	—	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	—	—	—	—	—	Δ	1	
	BAND #xx:3,@ERd	B			4							C∧(#xx:3 of @ERd24)→C	—	—	—	—	—	Δ	3	
	BAND #xx:3,@aa:8	B						4				C∧(#xx:3 of @aa:8)→C	—	—	—	—	—	Δ	3	
	BAND #xx:3,@aa:16	B						6				C∧(#xx:3 of @aa:16)→C	—	—	—	—	—	Δ	4	
	BAND #xx:3,@aa:32	B						8				C∧(#xx:3 of @aa:32)→C	—	—	—	—	—	Δ	5	
BIAND	BIAND #xx:3,Rd	B		2								C∧ [⟵ (#xx:3 of Rd8)]→C	—	—	—	—	—	Δ	1	
	BIAND #xx:3,@ERd	B			4							C∧ [⟵ (#xx:3 of @ERd24)]→C	—	—	—	—	—	Δ	3	
	BIAND #xx:3,@aa:8	B						4				C∧ [¬(#xx:3 of @aa:8)]→C	—	—	—	—	—	Δ	3	
	BIAND #xx:3,@aa:16	B						6				C∧ [¬(#xx:3 of @aa:16)]→C	—	—	—	—	—	Δ	4	
	BIAND #xx:3,@aa:32	B						8				C∧ [¬(#xx:3 of @aa:32)]→C	—	—	—	—	—	Δ	5	

	Mnemonic	Operand Size	Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States^{*1}	
			#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
BOR	BOR #xx:3,Rd	B		2								C∨(#xx:3 of Rd8)→C	—	—	—	—	—	Δ	1	
	BOR #xx:3,@ERd	B			4							C∨(#xx:3 of @ERd24)→C	—	—	—	—	—	Δ	3	
	BOR #xx:3,@aa:8	B						4				C∨(#xx:3 of @aa:8)→C	—	—	—	—	—	Δ	3	
	BOR #xx:3,@aa:16	B						6				C∨(#xx:3 of @aa:16)→C	—	—	—	—	—	Δ	4	
	BOR #xx:3,@aa:32	B						8				C∨(#xx:3 of @aa:32)→C	—	—	—	—	—	Δ	5	
BIOR	BIOR #xx:3,Rd	B		2								C∨ [¬(#xx:3 of Rd8)]→C	—	—	—	—	—	Δ	1	
	BIOR #xx:3,@ERd	B			4							C∨ [¬(#xx:3 of @ERd24)]→C	—	—	—	—	—	Δ	3	
	BIOR #xx:3,@aa:8	B						4				C∨ [¬(#xx:3 of @aa:8)]→C	—	—	—	—	—	Δ	3	
	BIOR #xx:3,@aa:16	B						6				C∨ [¬(#xx:3 of @aa:16)]→C	—	—	—	—	—	Δ	4	
	BIOR #xx:3,@aa:32	B						8				C∨ [¬(#xx:3 of @aa:32)]→C	—	—	—	—	—	Δ	5	
BXOR	BXOR #xx:3,Rd	B		2								C⊕ (#xx:3 of Rd8)→C	—	—	—	—	—	Δ	1	
	BXOR #xx:3,@ERd	B			4							C⊕ (#xx:3 of @ERd24)→C	—	—	—	—	—	Δ	3	
	BXOR #xx:3,@aa:8	B						4				C⊕ (#xx:3 of @aa:8)→C	—	—	—	—	—	Δ	3	
	BXOR #xx:3,@aa:16	B						6				C⊕ (#xx:3 of @aa:16)→C	—	—	—	—	—	Δ	4	
	BXOR #xx:3,@aa:32	B						8				C⊕ (#xx:3 of @aa:32)→C	—	—	—	—	—	Δ	5	
BIXOR	BIXOR #xx:3,Rd	B		2								C⊕ [¬(#xx:3 of Rd8)]→C	—	—	—	—	—	Δ	1	
	BIXOR #xx:3,@ERd	B			4							C⊕ [¬(#xx:3 of @ERd24)]→C	—	—	—	—	—	Δ	3	
	BIXOR #xx:3,@aa:8	B						4				C⊕ [¬(#xx:3 of @aa:8)]→C	—	—	—	—	—	Δ	3	
	BIXOR #xx:3,@aa:16	B						6				C⊕ [¬(#xx:3 of @aa:16)]→C	—	—	—	—	—	Δ	4	
	BIXOR #xx:3,@aa:32	B						8				C⊕ [¬(#xx:3 of @aa:32)]→C	—	—	—	—	—	Δ	5	

6. Branch instructions

			Addressing Mode/Instruction Length (Bytes)									Operation		Condition Code						No. of States*1	
	Mnemonic	Operand Size	#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—	Operation	Branching Condition	I	H	N	Z	V	C	Normal	Advanced
Bcc	BRA d:8(BT d:8)	—							2			if condition is true then PC←PC+d else next;	Always	—	—	—	—	—	—	2	
	BRA d:16(BT d:16)	—							4					—	—	—	—	—	—	3	
	BRN d:8(BF d:8)	—							2				Never	—	—	—	—	—	—	2	
	BRN d:16(BF d:16)	—							4					—	—	—	—	—	—	3	
	BHI d:8	—							2				C∨Z=0	—	—	—	—	—	—	2	
	BHI d:16	—							4					—	—	—	—	—	—	3	
	BLS d:8	—							2				C∨Z=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				Z∨(N⊕V) =0	—	—	—	—	—	—	2	
	BGT d:16	—							4					—	—	—	—	—	—	3	
	BLE d:8	—							2				Z∨(N⊕V) =1	—	—	—	—	—	—	2	
	BLE d:16	—							4					—	—	—	—	—	—	3	
JMP	JMP @ERn	—			2							PC←ERn		—	—	—	—	—	—	2	
	JMP @aa:24	—						4				PC←aa:24		—	—	—	—	—	—	3	
	JMP @@aa:8	—								2		PC←@aa:8		—	—	—	—	—	—	4	5
BSR	BSR d:8	—							2			PC→@–SP,PC←PC+d:8		—	—	—	—	—	—	3	4
	BSR d:16	—							4			PC→@–SP,PC←PC+d:16		—	—	—	—	—	—	4	5
JSR	JSR @ERn	—			2							PC→@–SP,PC←ERn		—	—	—	—	—	—	3	4
	JSR @aa:24	—						4				PC→@–SP,PC←aa:24		—	—	—	—	—	—	4	5
	JSR @@aa:8	—								2		PC→@–SP,PC←@aa:8		—	—	—	—	—	—	4	6
RTS	RTS	—									2	PC←@SP+		—	—	—	—	—	—	4	5

7. System control instructions

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States[*1]	
			#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
TRAPA	TRAPA #xx:2	—									2	PC→@–SP,CCR→@–SP, EXR→@–SP,<vector>→PC	1	—	—	—	—	—	7 [9]	8 [9]
RTE	RTE	—										EXR←@SP+,CCR←@SP+, PC←@SP+	Δ	Δ	Δ	Δ	Δ	Δ	5 [9]	
SLEEP	SLEEP	—										Transition to power-down state	—	—	—	—	—	—	2	
LDC	LDC #xx:8,CCR	B	2									#xx:8→CCR	Δ	Δ	Δ	Δ	Δ	Δ	1	
	LDC #xx:8,EXR	B	4									#xx:8→EXR	—	—	—	—	—	—	2	
	LDC Rs,CCR	B		2								Rs8→CCR	Δ	Δ	Δ	Δ	Δ	Δ	1	
	LDC Rs,EXR	B		2								Rs8→EXR	—	—	—	—	—	—	1	
	LDC @ERs,CCR	W			4							@ERs→CCR	Δ	Δ	Δ	Δ	Δ	Δ	3	
	LDC @ERs,EXR	W			4							@ERs→EXR	—	—	—	—	—	—	3	
	LDC @(d:16,ERs),CCR	W				6						@(d:16,ERs)→CCR	Δ	Δ	Δ	Δ	Δ	Δ	4	
	LDC @(d:16,ERs),EXR	W				6						@(d:16,ERs)→EXR	—	—	—	—	—	—	4	
	LDC @(d:32,ERs),CCR	W				10						@(d:32,ERs)→CCR	Δ	Δ	Δ	Δ	Δ	Δ	6	
	LDC @(d:32,ERs),EXR	W				10						@(d:32,ERs)→EXR	—	—	—	—	—	—	6	
	LDC @ERs+,CCR	W					4					@ERs→CCR,ERs32+2→ERs32	Δ	Δ	Δ	Δ	Δ	Δ	4	
	LDC @ERs+,EXR	W					4					@ERs→EXR,ERs32+2→ERs32	—	—	—	—	—	—	4	
	LDC @aa:16,CCR	W						6				@aa:16→CCR	Δ	Δ	Δ	Δ	Δ	Δ	4	
	LDC @aa:16,EXR	W						6				@aa:16→EXR	—	—	—	—	—	—	4	
	LDC @aa:32,CCR	W						8				@aa:32→CCR	Δ	Δ	Δ	Δ	Δ	Δ	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	Δ	Δ	Δ	Δ	Δ	Δ	1	
	ANDC #xx:8,EXR	B	4									EXR∧#xx:8→EXR	—	—	—	—	—	—	2	
ORC	ORC #xx:8,CCR	B	2									CCR∨#xx:8→CCR	Δ	Δ	Δ	Δ	Δ	Δ	1	
	ORC #xx:8,EXR	B	4									EXR∨#xx:8→EXR	—	—	—	—	—	—	2	
XORC	XORC #xx:8,CCR	B	2									CCR⊕#xx:8→CCR	Δ	Δ	Δ	Δ	Δ	Δ	1	
	XORC #xx:8,EXR	B	4									EXR⊕#xx:8→EXR	—	—	—	—	—	—	2	
NOP	NOP	—									2	PC←PC+2	—	—	—	—	—	—	1	

8. Block transfer instructions

Mnemonic		Operand Size	Addressing Mode/Instruction Length (Bytes)									Operation	Condition Code						No. of States*1	
			#xx	Rn	@ERn	@(d,ERn)	@–ERn/@ERn+	@aa	@(d,PC)	@@aa	—		I	H	N	Z	V	C	Normal	Advanced
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] Seven states for saving or restoring two registers, nine states for three registers, or eleven states for four registers.
[2] Cannot be used in the H8S/2355 Series.
[3] Set to 1 when a carry or borrow occurs at bit 11; otherwise cleared to 0.
[4] Set to 1 when a carry or borrow occurs at bit 27; otherwise cleared to 0.
[5] Retains its previous value when the result is zero; otherwise cleared to 0.
[6] Set to 1 when the divisor is negative; otherwise cleared to 0.
[7] Set to 1 when the divisor is zero; otherwise cleared to 0.
[8] Set to 1 when the quotient is negative; otherwise cleared to 0.
[9] One additional state is required for execution when EXR is valid.

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.

	Access Conditions						
				External Data Bus			
		On-Chip Supporting Module		8-Bit Bus		16-Bit Bus	
Cycle	On-Chip Memory	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 address read							
Stack operation							
Byte data access		2		2	3+m		
Word data access		4		4	6+2m		
Internal operation	1						

Legend
m: Number of wait states inserted into external device access

Condition Code Notation

Symbol	Meaning
Δ	Changes according to the result of instruction execution
*	Undetermined (no guaranteed value)
0	Always cleared to 0
1	Always set to 1
—	Not affected by execution of the instruction

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	Extend register
CCR	Condition code register
N	N (negative) flag of CCR
Z	Z (zero) flag of CCR
V	V (overflow) flag of CCR
C	C (carry) flag of CCR
PC	Program counter
SP	Stack pointer
#IMM	Immediate data
disp	Displacement
+	Addition
–	Subtraction
×	Multiplication
÷	Division
∧	AND logical
∨	OR logical
⊕	Exclusive OR logical
→	Transfer from left-hand operand to right-hand operand, or transition from left-hand state to right-hand state
¬	NOT (logical complement)
( ) < >	Operand contents
:8/:16/:24/:32	8-, 16-, 24-, or 32-bit length

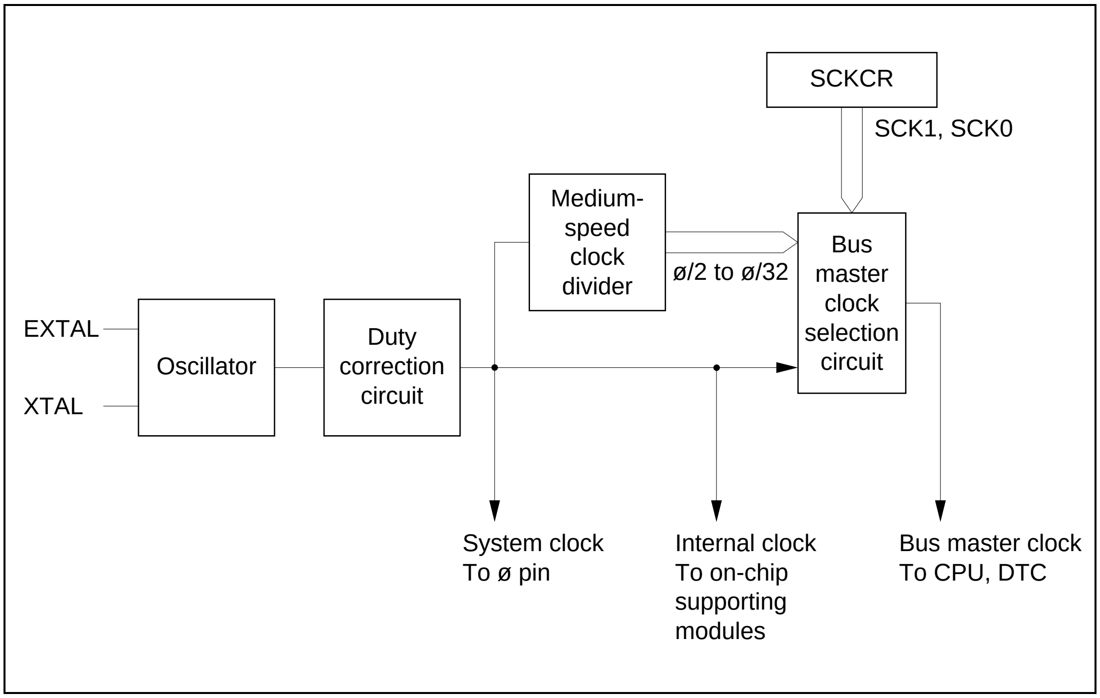
Note: * General registers include 8-bit registers (R0H to R7H, R0L to R7L), 16-bit registers (R0 to R7, E0 to E7), and 32-bit registers (ER0 to ER7).

2.6 Basic Bus Timing

The CPU operates on the basis of the system clock (ϕ). One ϕ 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 devices.

Basic Clock Timing

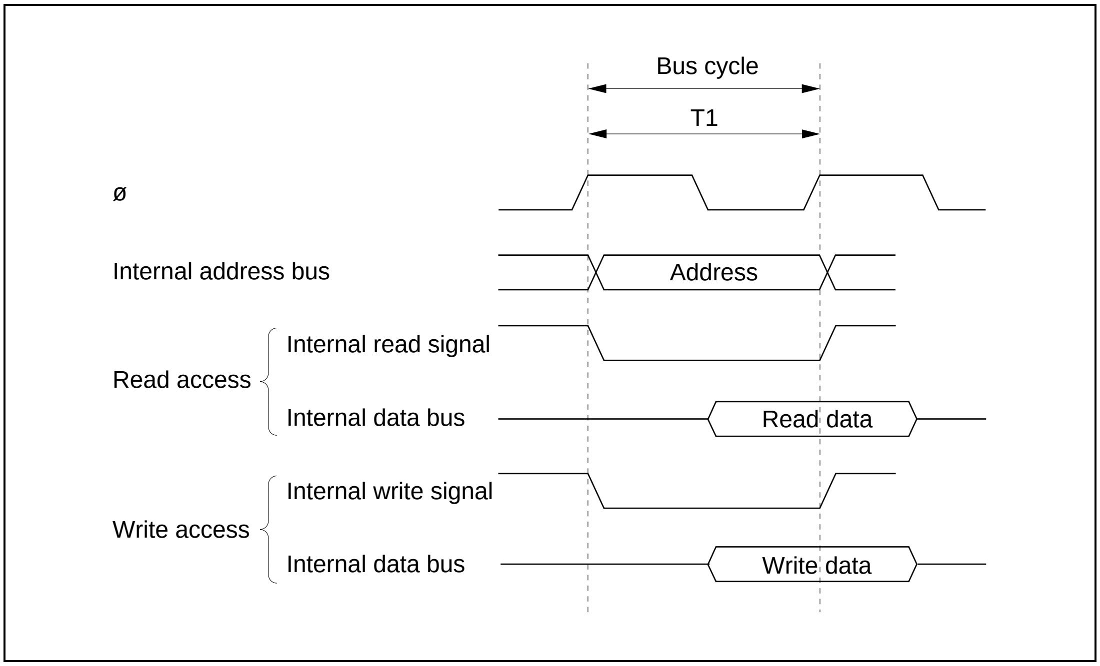
An external clock is input to the EXTAL pin, or a crystal oscillator is connected to the EXTAL pin, to generate the system clock (ϕ). An external clock or crystal oscillator of the same frequency as the ϕ clock should be used.



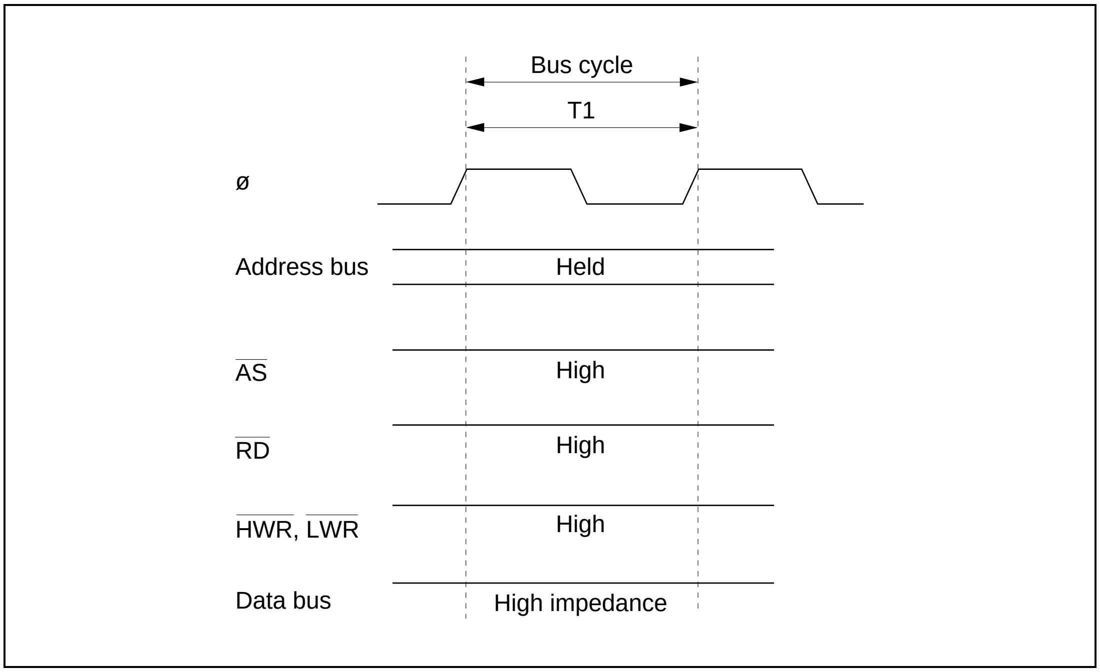
CPU Read/Write Cycles

The CPU operates on the basis of the system clock (ϕ). One ϕ 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 devices. 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.

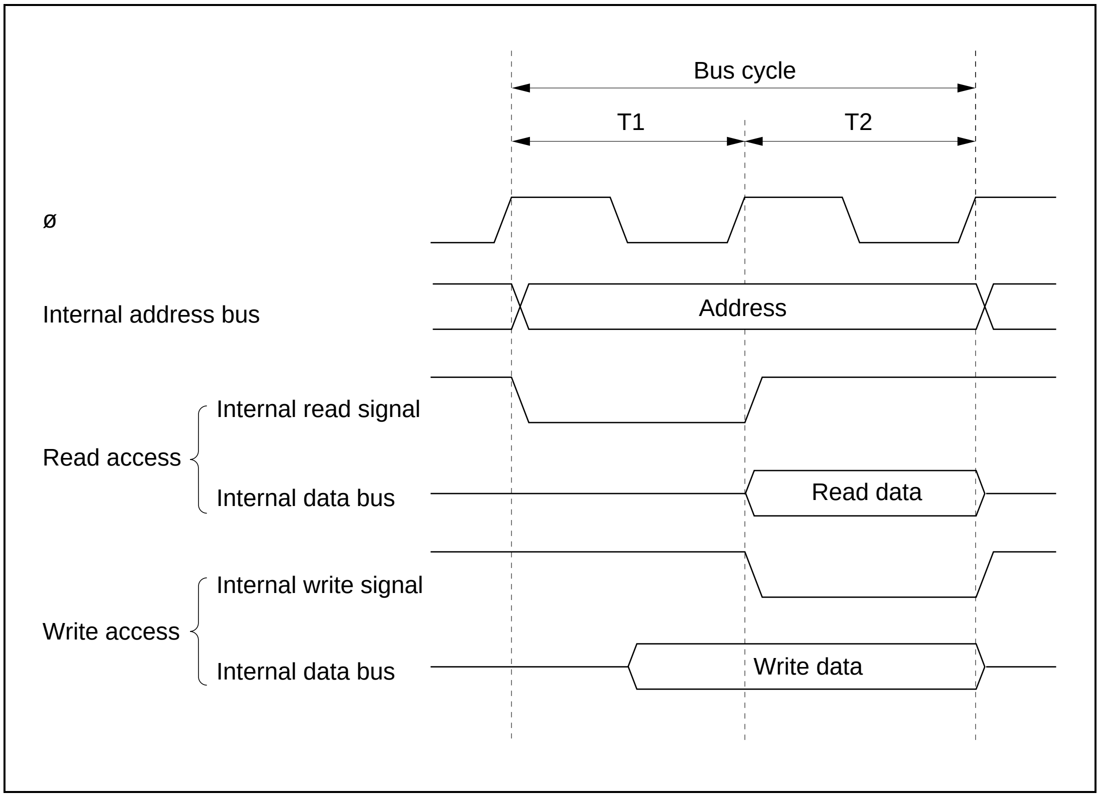


On-Chip Memory Access Cycle (One-State Access)

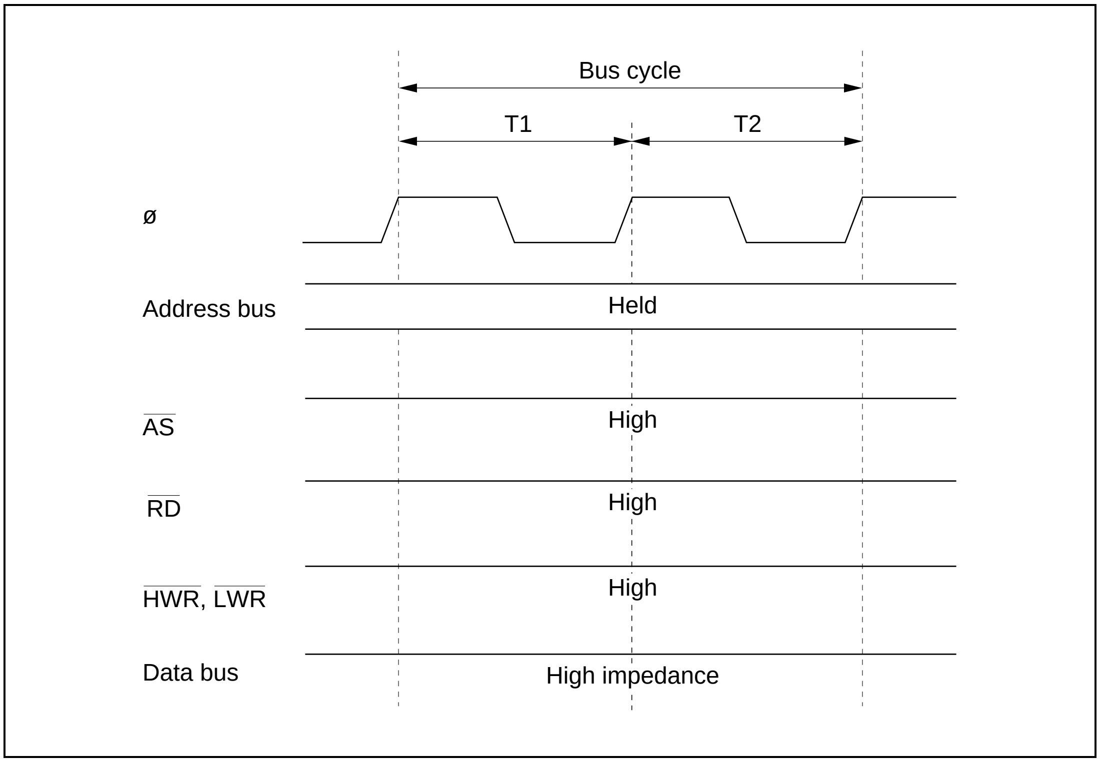


Pin States during On-Chip Memory Access

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



On-Chip Supporting Module Access Timing (Two-State Access)



Pin States during On-Chip Supporting Module Access

External Address Space: The external address space is accessed via an 8-bit or 16-bit bus, and in two or three states. Wait state insertion is possible in the case of 3-state access. See the Bus Controller section for details.

2.7 Processing States

The 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 an reset, interrupt, or trap instruction exception handling source

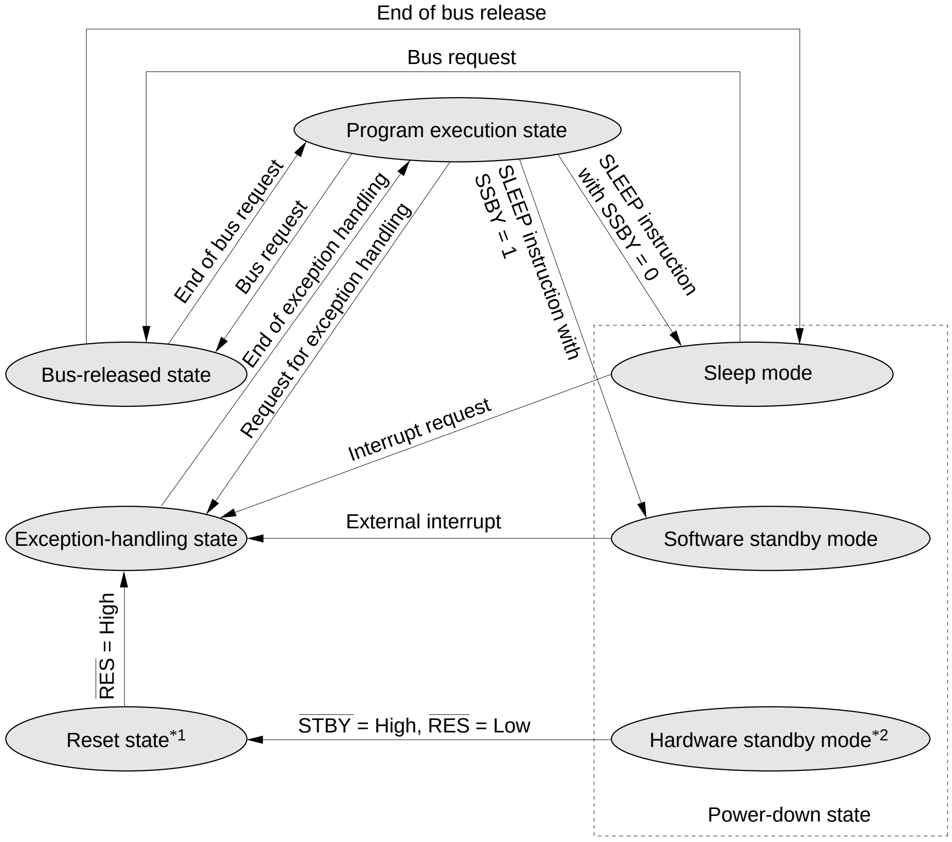
Bus-Released State

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). The power-down state also includes medium-speed mode and module stop mode.

State Transition Diagram



Notes: 1. From any state except hardware standby mode, a transition to the reset state occurs whenever $\overline{\text{RES}}$ goes low. A transition to the reset state can also be caused by watchdog timer overflow.

2. From any state, a transition to hardware standby mode occurs when $\overline{\text{STBY}}$ goes low.

2.8 Exception Handling

The 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.

Exception Handling Types and Priorities

Priority	Exception Type	Start of Exception Handling
High	Reset	After a low-to-high transition at the RES pin, or when the watchdog timer overflows
	Trace	After instruction or exception handling execution when the trace (T) bit is 1
	Interrupt	When an interrupt is generated, after instruction or exception handling execution
Low	Trap instruction (TRAPA)	When a trap (TRAPA) instruction is executed

Exception Handling Operation

Exception handling is started by any of the exception handling sources. Trap instruction exception handling is always accepted in the program execution state.

The operations in trap instruction and interrupt exception handling are as follows.

(1) The program counter (PC), condition code register (CCR), and extended register (EXR) are saved on the stack.

(2) The interrupt mask bit is updated, and the T bit is cleared to 0.

(3) The vector address corresponding to the activation source is generated, and program execution is started from the address indicates by the contents of the vector address.

In reset exception handling, only operations (2) and (3) are performed.

Exception Vector Table

			Vector Address[*1]	
Exception Source		**Vector Number**	**Normal Mode**	**Advanced Mode**
Power-on reset		0	H'0000–H'0001	H'0000–H'0003
Manual reset		1	H'0002–H'0003	H'0004–H'0007
Reserved for system use		2	H'0004–H'0006	H'0008–H'000B
		3	H'0006–H'0007	H'000C–H'000F
		4	H'0008–H'0009	H'0010–H'0013
Trace		5	H'000A–H'000B	H'0014–H0017
Reserved for system use		6	H'000C–H'000D	H'0018–H001B
External interrupt	NMI	7	H'000E–H'000F	H'001C–H'001F
Trap instruction (4 sources)		8	H'0010–H'0011	H'0020–H'0023
		9	H'0012–H'0013	H'0024–H'0027
		10	H'0014–H'0015	H'0028–H'002B
		11	H'0016–H'0017	H'002C–H'002F
Reserved for system use		12	H'0018–H'0019	H'0030–H'0033
		13	H'001A–H'001B	H'0034–H'0037
		14	H'001C–H'001D	H'0038–H'003B
		15	H'001E–H'001F	H'003C–H'003F
External interrupt	IRQ0	16	H'0020–H'0021	H'0040–H'0043
	IRQ1	17	H'0022–H'0023	H'0044–H'0047
	IRQ2	18	H'0024–H'0025	H'0048–H'004B
	IRQ3	19	H'0026–H'0027	H'004C–H'004F
	IRQ4	20	H'0028–H'0029	H'0050–H'0053
	IRQ5	21	H'002A–H'002B	H'0054–H'0057
	IRQ6	22	H'002C–H'002D	H'0058–H'005B
	IRQ7	23	H'002E–H'002F	H'005C–H'005F
Internal interrupt[*2]		24 to 91	H'0030–H'0031 to H'00B6–H'00B7	H'0060–H'0063 to H'016C–H'016F

Notes: 1. Lower 16 bits of address

2. See the Interrupt Exception Vector Table for the internal interrupt vector table.

2.9 Interrupts

This section describes the sru interrupt, one of the external interrupt sources.

Interrupts are controlled by the interrupt controller. There are a total of 56 interrupt sources, comprising nine external interrupts from the external pins (NMI, $\overline{\text{IRQ0}}$ to $\overline{\text{IRQ7}}$), and 47 internal interrupts from on-chip supporting modules. A separate vector number is assigned to each interrupt.

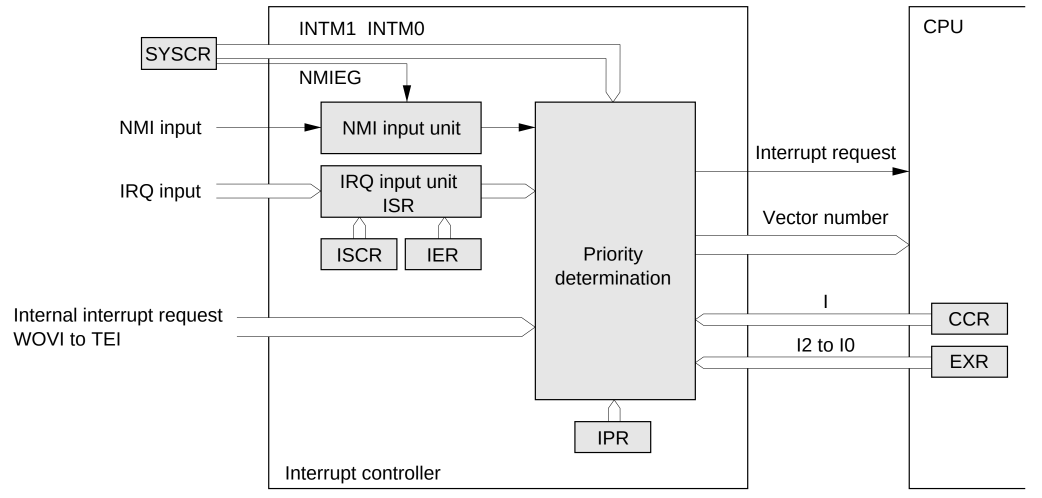
Interrupt Control

Any 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.

Block Diagram of Interrupt Controller



Legend

ISCR: IRQ sense control register

IER: IRQ enable register

ISR: IRQ status register

IPR: Interrupt priority register

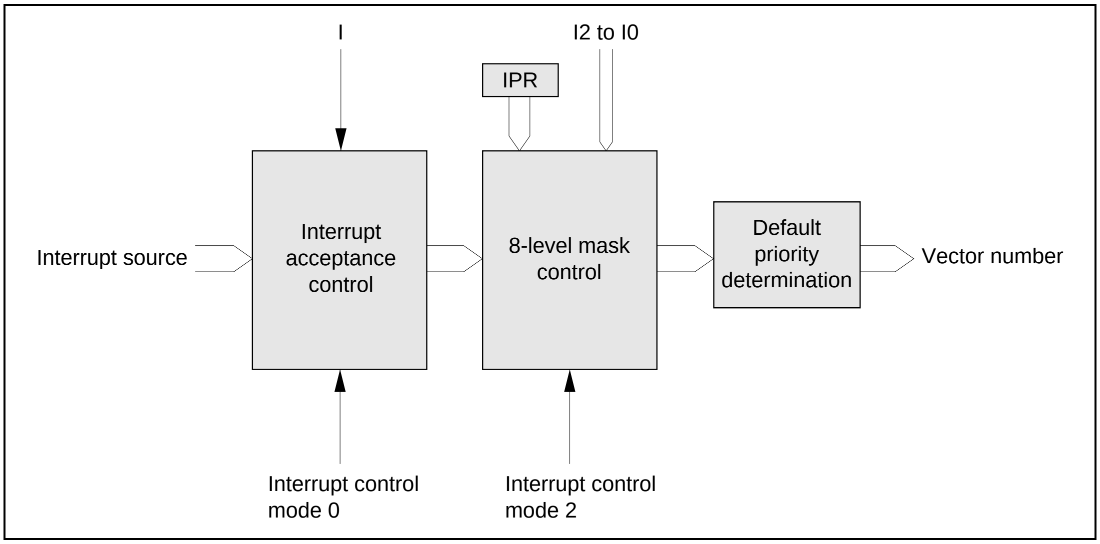
SYSCR: System control register

Interrupt Control Modes

	SYSCR				
Interrupt Control Mode	**INTM1**	**INTM0**	**Priority Setting Registers**	**Interrupt Mask Bits**	**Description**
0	0	0	—	I	Interrupt mask control is performed by the I bit.
1*		1	—	—	—
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.
3*		1	—	—	—

Note: * Interrupt control modes 1 and 3 may not be set.

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.

Interrupt Sources, Vector Addresses, and Interrupt Priorities

			Vector Address*			
Interrupt Source	**Origin of Interrupt Source**	**Vector Number**	**Normal Mode**	**Advanced Mode**	**IPR**	**Priority**
NMI	External pin	7	H'000E	H'001C		High
IRQ0		16	H'0020	H'0040	IPRA6–IPRA4	
IRQ1		17	H'0022	H'0044	IPRA2–IPRA0	
IRQ2		18	H'0024	H'0048	IPRB6–IPRB4	
IRQ3		19	H'0026	H'004C		
IRQ4		20	H'0028	H'0050	IPRB2–IPRB0	
IRQ5		21	H'002A	H'0054		
IRQ6		22	H'002C	H'0058	IPRC6–IPRC4	
IRQ7		23	H'002E	H'005C		
SWDTEND (software activation interrupt end	DTC	24	H'0030	H'0060	IPRC2–IPRC0	
WOVI (interval timer)	Watchdog timer	25	H'0032	H'0064	IPRD6–IPRD4	
ADI (A/D conversion end)	A/D	28	H'0038	H'0070	IPRE2–IPRE0	
TGI0A (TGR0A input capture/ compare-match)	TPU channel 0	32	H'0040	H'0080	IPRF6–IPRF4	
TGI0B (TGR0B input capture/ compare-match)		33	H'0042	H'0084		
TGI0C (TGR0C input capture/ compare-match)		34	H'0044	H'0088		
TGI0D (TGR0D input capture/ compare-match)		35	H'0046	H'008C		
TCI0V (overflow 0)		36	H'0048	H'0090		
TGI1A (TGR1A input capture/ compare-match)	TPU channel 1	40	H'0050	H'00A0	IPRF2–IPRF0	
TGI1B (TGR1B input capture/ compare-match)		41	H'0052	H'00A4		
TCI1V (overflow 1)		42	H'0054	H'00A8		
TCI1U (underflow 1)		43	H'0056	H'00AC		
TGI2A (TGR2A input capture/ compare-match)	TPU channel 2	44	H'0058	H'00B0	IPRG6–IPRG4	
TGI2B (TGR2B input capture/ compare-match)		45	H'005A	H'00B4		
TCI2V (overflow 2)		46	H'005C	H'00B8		
TCI2U (underflow 2)		47	H'005E	H'00BC		Low

Note: * Lower 16 bits of the start address.

			Vector Address*			
Interrupt Source	Origin of nterrupt Source	Vector Number	Normal Mode	Advanced Mode	IPR	Priority
TGI3A (TGR3A input capture/ compare-match)	TPU channel 3	48	H'0060	H'00C0	IPRG2–IPRG0	High
TGI3B (TGR3B input capture/ compare-match)		49	H'0062	H'00C4		
TGI3C (TGR3C input capture/ compare-match)		50	H'0064	H'00C8		
TGI3D (TGR3D input capture/ compare-match)		51	H'0066	H'00CC		
TCI3V (overflow 3)		52	H'0068	H'00D0		
TGI4A (TGR4A input capture/ compare-match)	TPU channel 4	56	H'0070	H'00E0	IPRH6–IPRH4	
TGI4B (TGR4B input capture/ compare-match)		57	H'0072	H'00E4		
TCI4V (overflow 4)		58	H'0074	H'00E8		
TCI4U (underflow 4)		59	H'0076	H'00EC		
TGI5A (TGR5A input capture/ compare-match)	TPU channel 5	60	H'0078	H'00F0	IPRH2–IPRH0	
TGI5B (TGR5B input capture/ compare-match)		61	H'007A	H'00F4		
TCI5V (overflow 5)		62	H'007C	H'00F8		
TCI5U (underflow 5)		63	H'007E	H'00FC		
CMIA0 (compare-match A)	8-bit timer channel 0	64	H'0080	H'0100	IPRJ6–IPRJ4	
CMIB0 (compare-match B)		65	H'0082	H'0104		
OVI0 (overflow 0)		66	H'0084	H'0108		
CMIA1 (compare-match A)	8-bit timer channel 1	68	H'0088	H'0110	IPRI2–IPRI0	
CMIB1 (compare-match B)		69	H'008A	H'0114		
OVI1 (overflow 0)		70	H'008C	H'0118		
ERI0 (receive error 0)	SCI channel 0	80	H'00A0	H'0140	IPRJ2–IPRJ0	
RXI0 (reception completed 0)		81	H'00A2	H'0144		
TXI0 (transmit data empty 0)		82	H'00A4	H'0148		
TEI0 (transmission end 0)		83	H'00A6	H'014C		
ERI1 (receive error 1)	SCI channel 1	84	H'00A8	H'0150	IPRK6–IPRK4	
RXI1 (reception completed 1)		85	H'00AA	H'0154		
TXI1 (transmit data empty 1)		86	H'00AC	H'0158		
TEI1 (transmission end 1)		87	H'00AE	H'015C		
ERI2 (receive error 2)	SCI channel 2	88	H'00B0	H'0160	IPRK2–IPRK0	
RXI2 (reception completed 2)		89	H'00B2	H'0164		
TXI2 (transmit data empty 2)		90	H'00B4	H'0168		
TEI2 (transmission end 2)		91	H'00B6	H'016C		Low

Note: * Lower 16 bits of the start address

2.10 Operating Modes

The H8S/2355 Series has seven operating modes. These modes enable selection of the CPU operating mode, enabling/disabling of on-chip ROM, and the initial bus width setting, by setting the mode pins (MD_2 to MD_0).

Normal Modes (Modes 1 to 3)

Mode 1 (Expansion Mode with On-Chip ROM Disabled): The CPU can access a 64-kbyte address space in normal mode. The on-chip ROM is disabled, and 8-bit bus mode is set immediately after a reset.

Ports B and C function as an address bus, port D functions as a data bus, and part of port F carries bus control signals.

Mode 2 (Expansion Mode with On-Chip ROM Enabled) : The CPU can access a 64-kbyte address space in normal mode. The on-chip ROM is enabled, and 8-bit bus mode is set immediately after a reset.

Ports B and C function as input ports immediately after a reset. They can each be set to output addresses by setting the corresponding bits in the data direction register (DDR) to 1. Port D functions as a data bus, and part of port F carries bus control signals.

The amount of on-chip ROM that can be used is limited to 56 kbytes.

Mode 3 (Single-Chip Mode) : The CPU can access a 64-kbyte address space in normal 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.

The amount of on-chip ROM that can be used is limited to 56 kbytes.

Advanced Modes (Modes 4 to 7)

Mode 4 (Expansion Mode with On-Chip ROM Disabled): The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is disabled.

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, if 8-bit access is designated by the bus controller for all areas, the bus mode switches to 8 bits.

Mode 5 (Expansion Mode with On-Chip ROM Disabled): The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is disabled.

Ports A, B and C function as an address bus, port D 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, 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 (Expansion Mode with On-Chip ROM Enabled) : The CPU can access a 16-Mbyte address space in advanced mode. The on-chip ROM is enabled.

Ports A, B and C function as input ports immediately after a reset. They can each be set to output addresses by setting the corresponding bits in the data direction register (DDR) to 1. Port D functions 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.

Mode 7 (Single-Chip Mode) : 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.

Kinds of Operating Mode

MCU Operating Mode	MD2	MD1	MD0	CPU Operating Mode	Description	On-Chip ROM	External Data Bus	
							Initial Width	Max. Width
0	0	0	0	—	—	—	—	—
1			1	Normal	Expanded mode with on-chip ROM disabled	Disabled	8 bits	16 bits
2		1	0		Expanded mode with on-chip ROM enabled	Enabled	8 bits	16 bits
3			1		Single-chip mode		—	—
4	1	0	0	Advanced	Expanded mode with on-chip ROM disabled	Disabled	16 bits	16 bits
5			1				8 bits	16 bits
6		1	0		Expanded mode with on-chip ROM enabled	Enabled	8 bits	16 bits
7			1		Single-chip mode		—	—

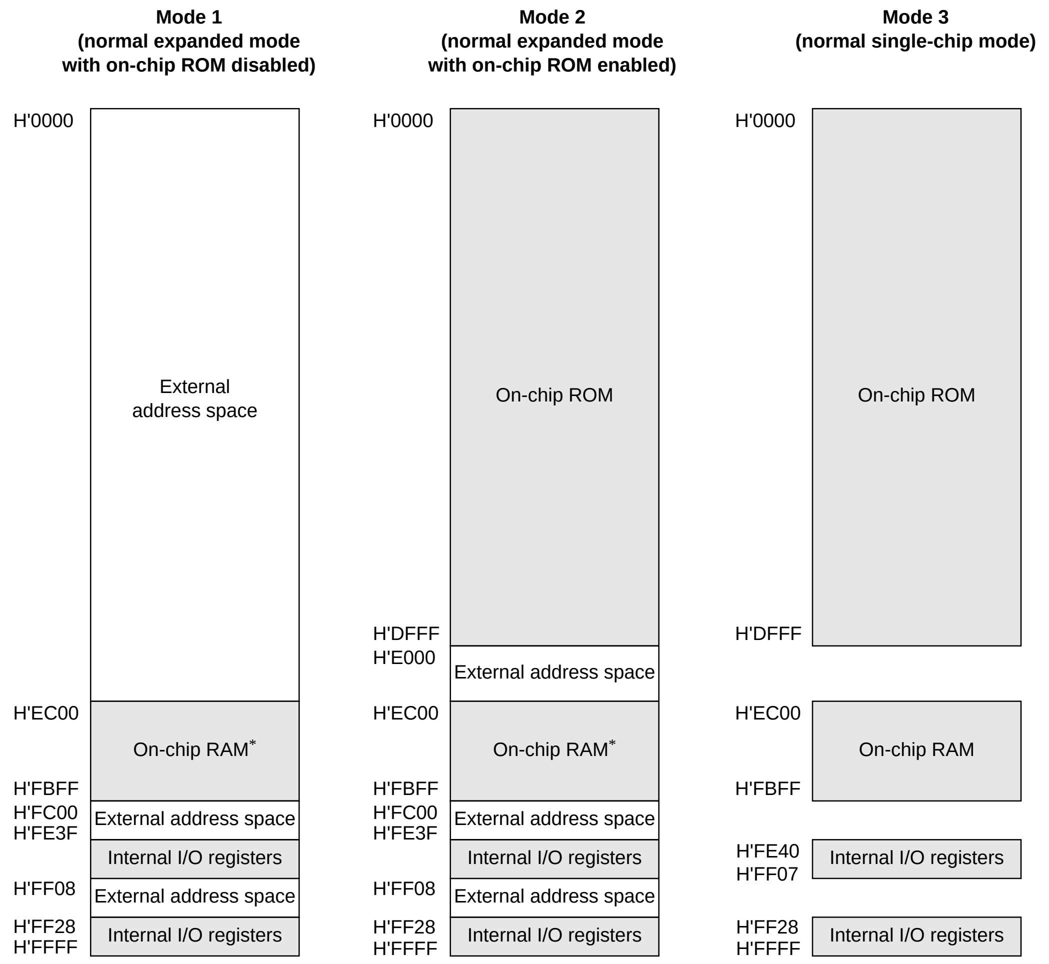
2.11 Address Map

This section shows the address map in each operating mode.

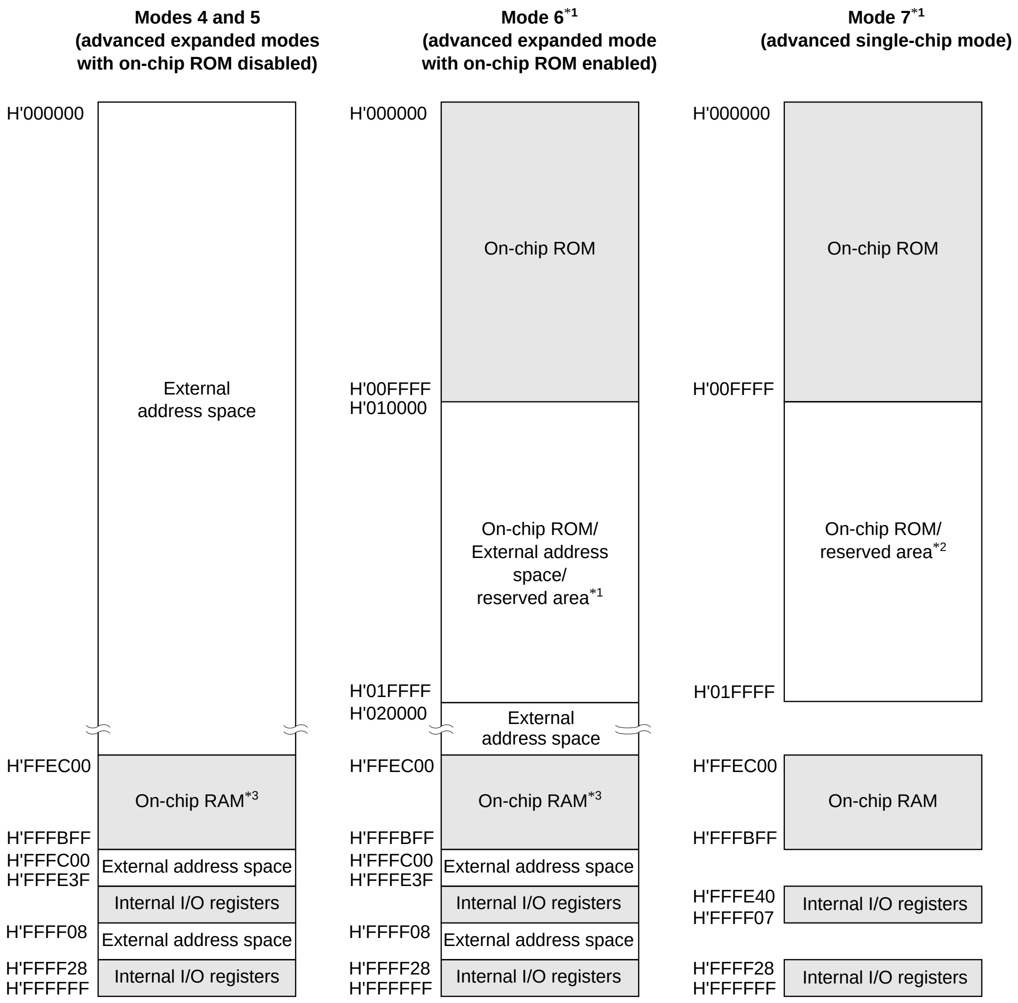
The address space is 64 kbytes in modes 1 to 3 (normal mode), and 16 Mbytes in modes 4 to 7 (advanced modes).

The on-chip ROM size is 64 kbytes, but only 56 kbytes of on-chip ROM can be used in modes 2 and 3 (normal modes).

Address Map in Each Operating Mode



Note: * External addresses can be accessed by clearing the RAME bit in SYSCR to 0.



Notes: 1. When the EAE bit in BCRL is set to 1, this area is external address space. When the EAE bit is cleared to 0, it is on-chip ROM in the H8S/2355, and a reserved area in the H8S/2353.
2. In the H8S/2355, this area is reserved when the EAE bit in BCRL is set to 1, and on-chip ROM when the EAE bit is cleared to 0. In the H8S/2353, it is a reserved area.
3. External addresses can be accessed by clearing the RAME bit in SYSCR to 0.

In modes 4 to 7 the address space is divided into 8 areas. See section 3.1.1, Area Partitioning, for details.

Section 3 Peripheral Functions

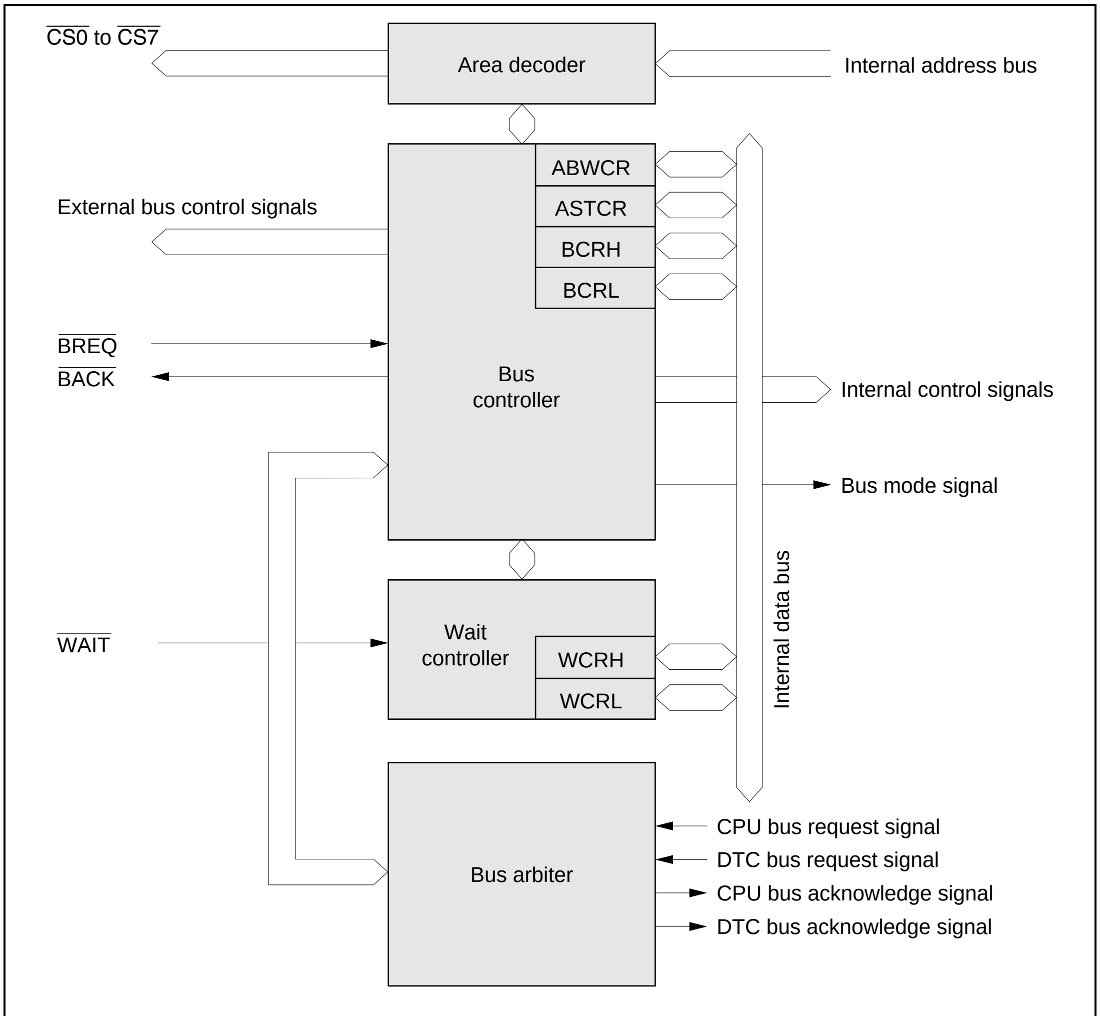
3.1 Bus Controller (BSC)

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).

Features

- Manages external address space in area units
  - In advanced mode, manages the external space as 8 areas of 2-Mbytes
  - In normal mode, manages the external space as a single area
  - Bus specifications can be set independently for each area
  - Burst ROM interfaces can be set
- Basic bus interface
  - Chip select ($\overline{\text{CS0}}$ to $\overline{\text{CS7}}$) 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
- 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

Bus Controller Block Diagram



Legend

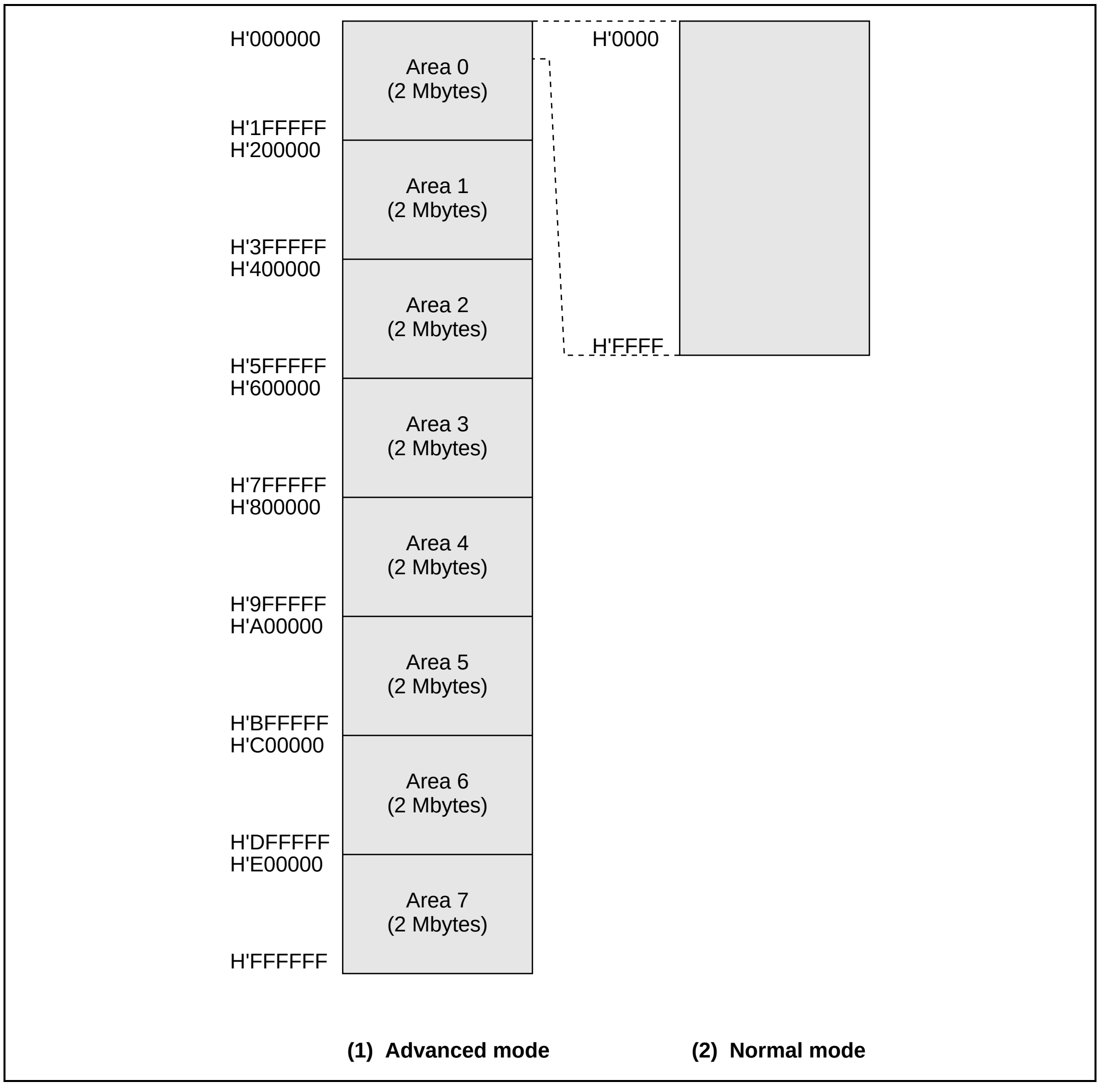
ABWCR:	Bus width control register	BCRL:	Bus control register L
ASTCR:	Access state control register	WCRH:	Wait control register H
BCRH:	Bus control register H	WCRL:	Wait control register L

3.1.1 Area Partitioning

In advanced mode, 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. In normal mode, it controls a 64-kbyte access space comprising part of area 0.

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

Overview of Area Partitioning



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.

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

Memory Interfaces

The H8S/2355 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.

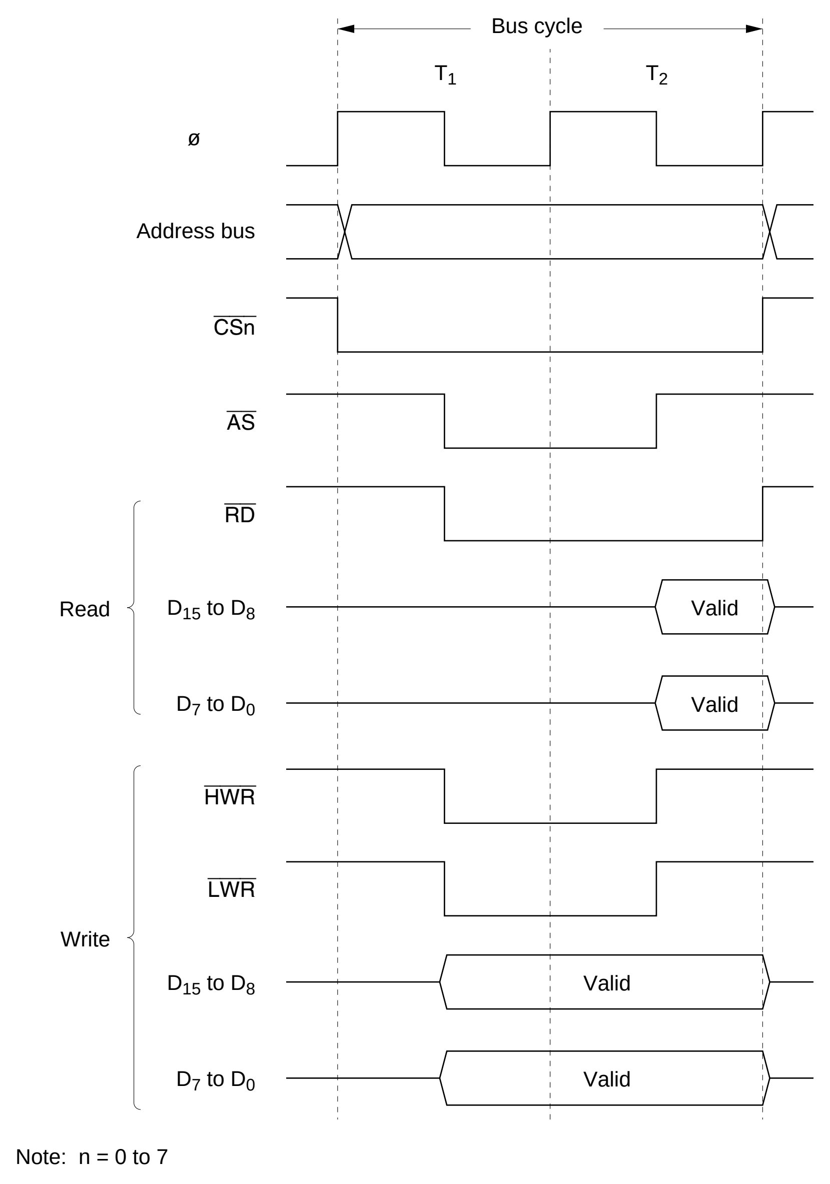
3.1.2 Basic Bus Interface

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

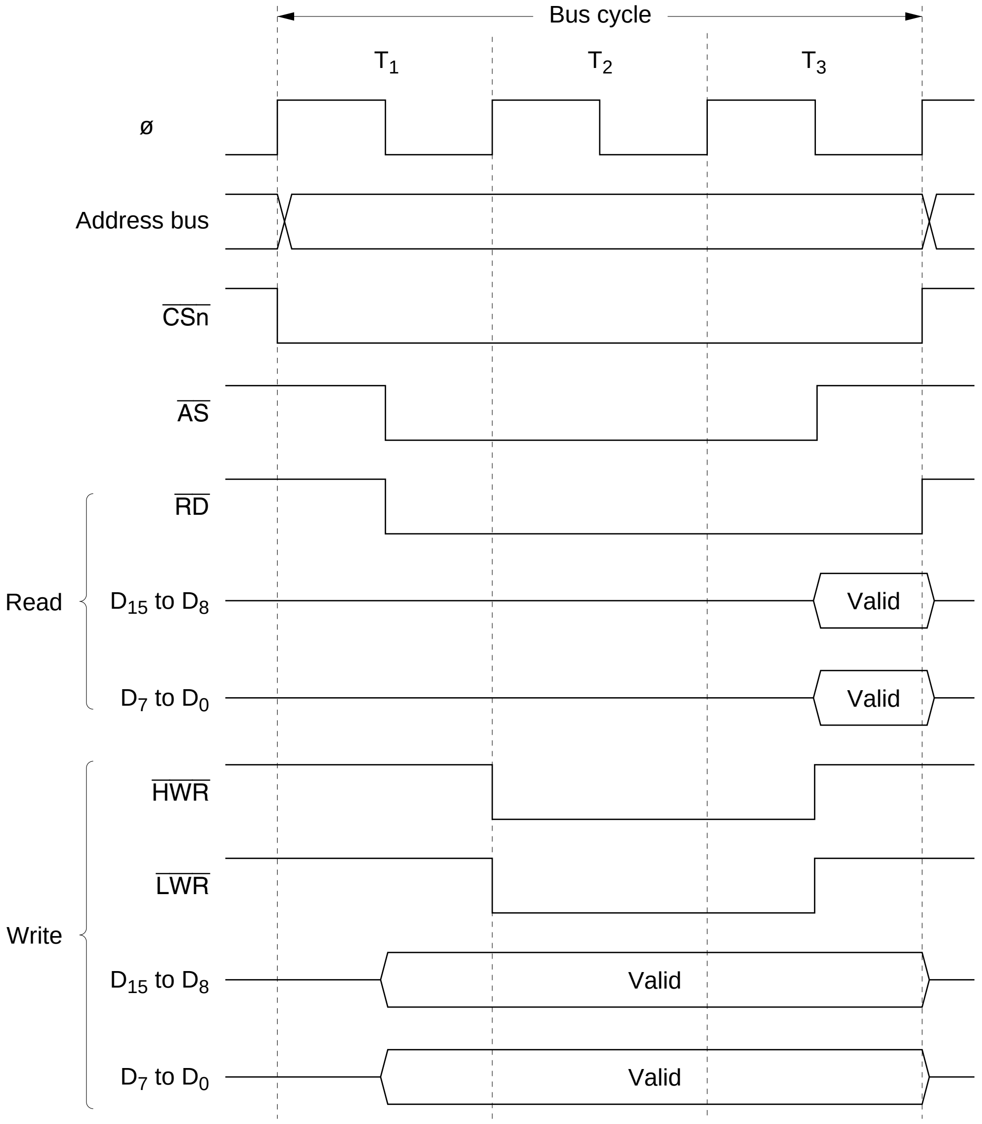
In 3-state access space, 0 to 3 program wait states or a pin wait by means of the $\overline{\text{WAIT}}$ pin can be inserted.

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



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



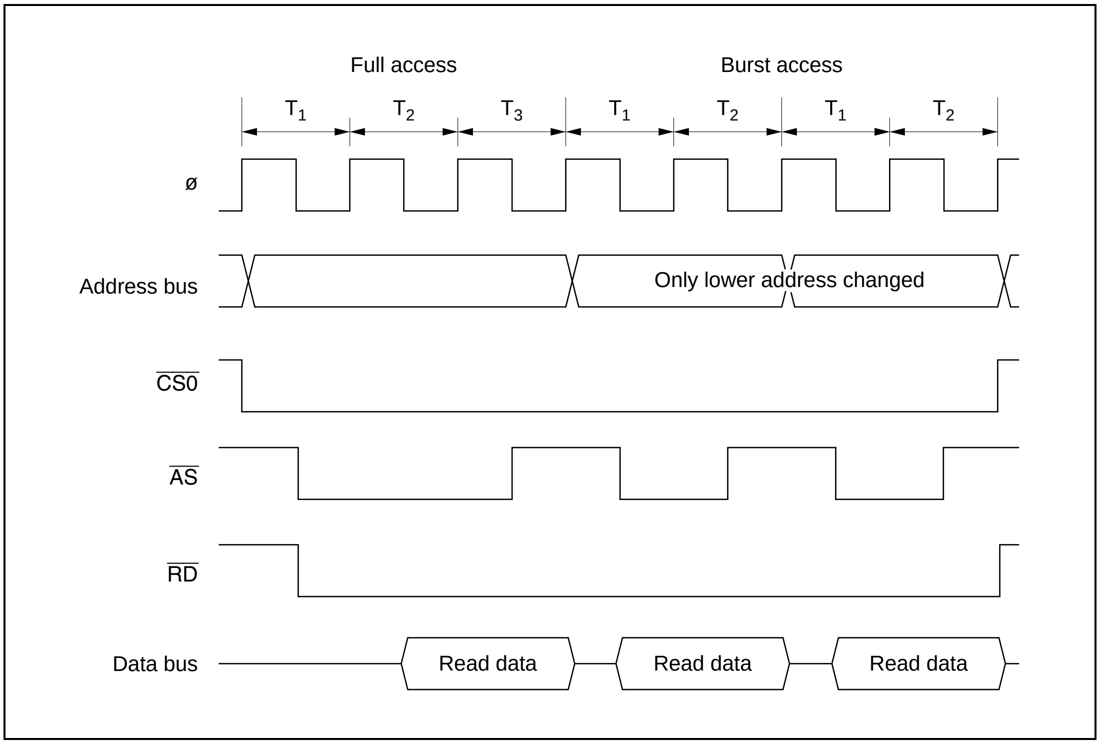
Note: n = 0 to 7

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

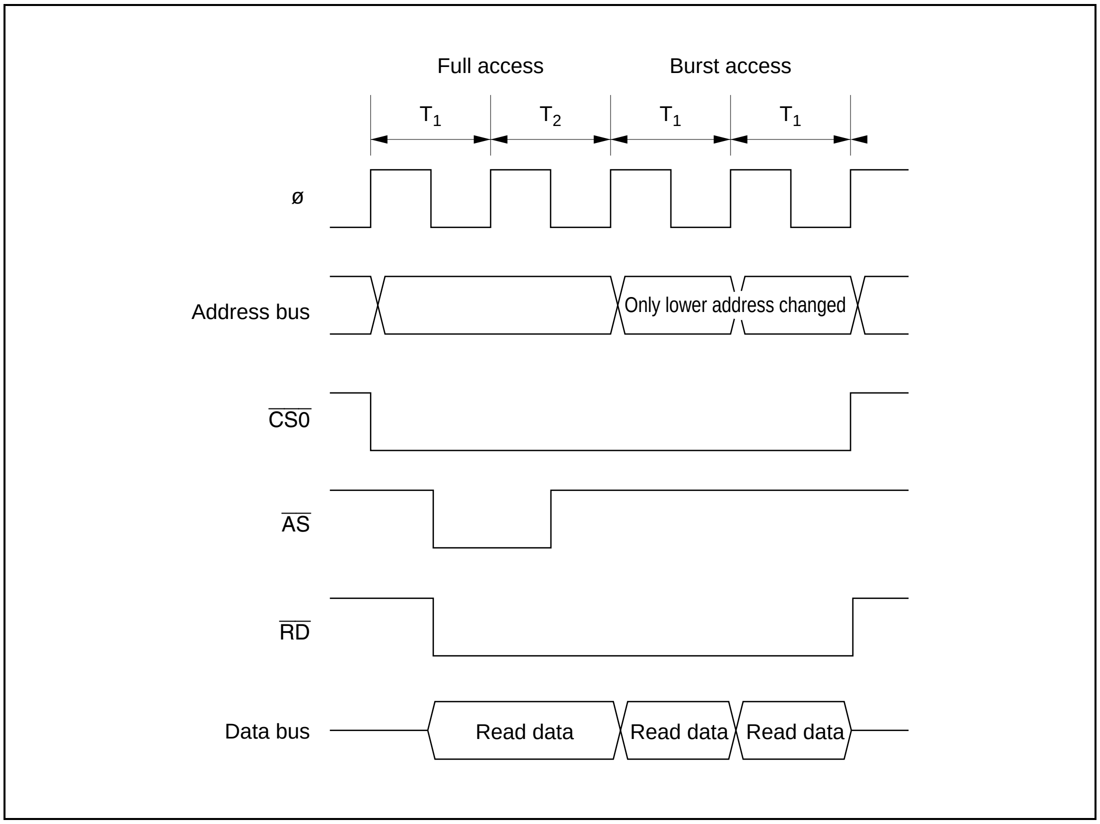
3.1.3 Burst ROM Interface

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.



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



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

3.2 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.

Features

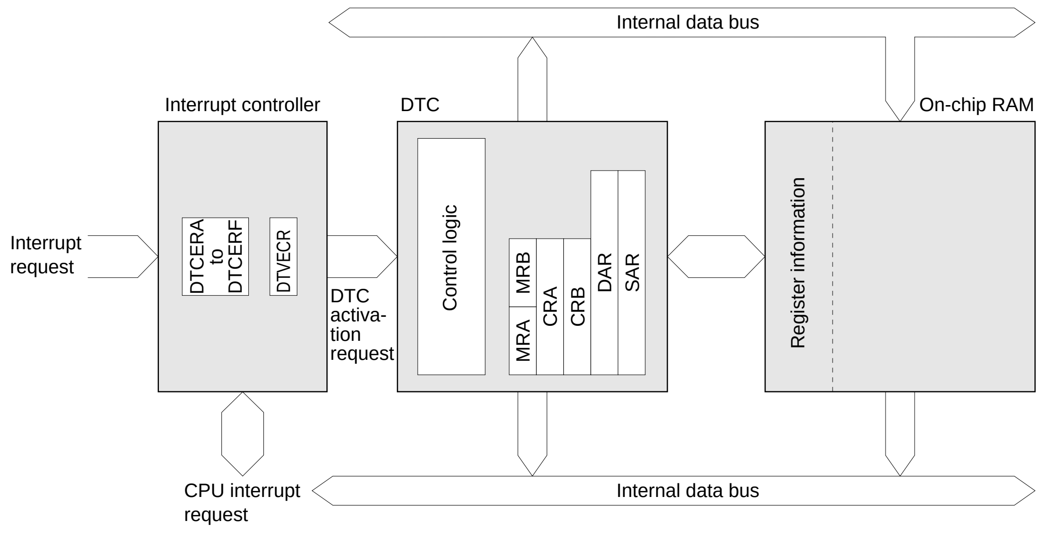
- Transfer possible over any number of channels
  - Transfer information is stored in memory
  - One activation source can trigger a number of data transfers (chain transfer)

- 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

DTC Block Diagram

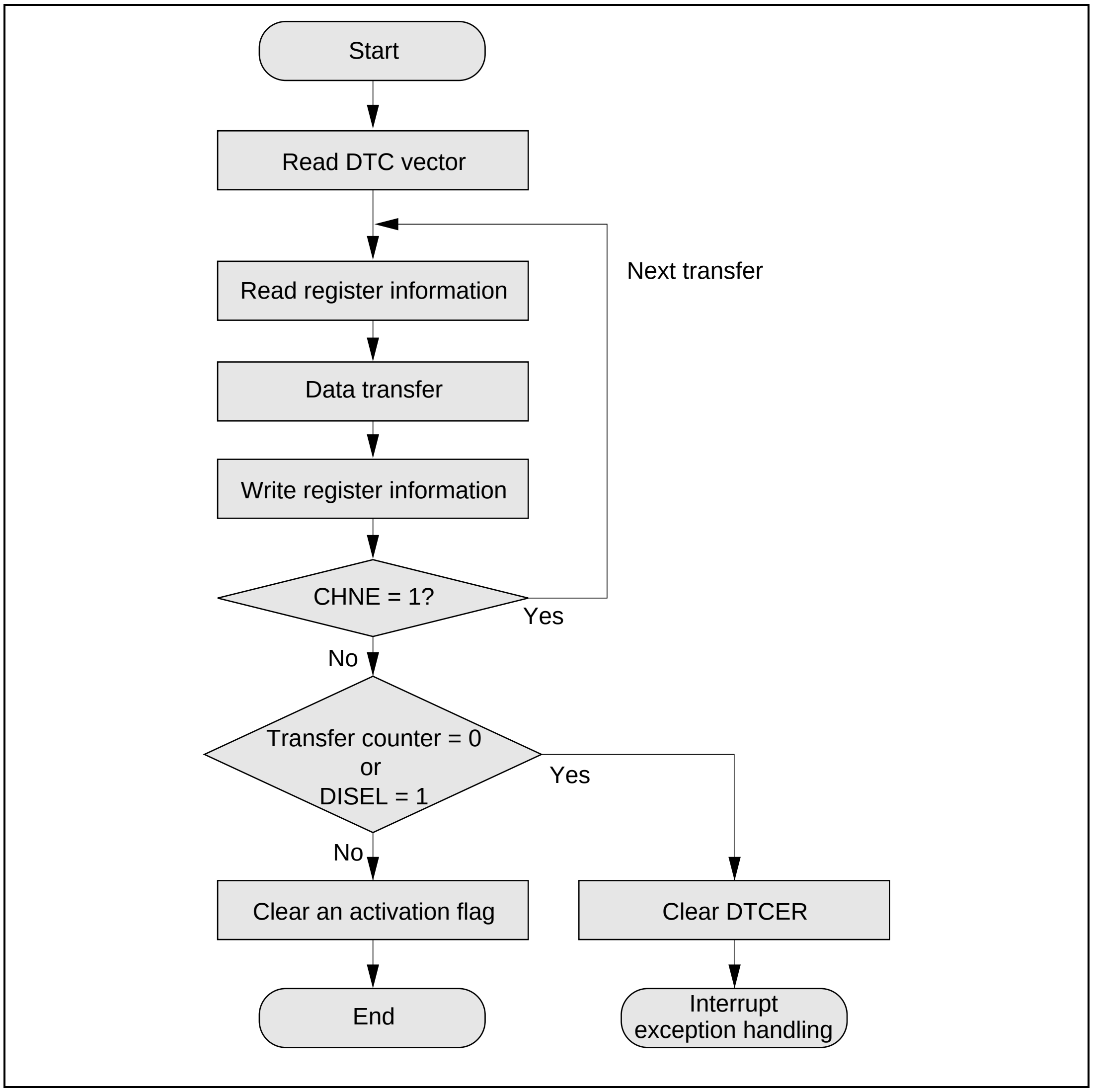


Legend
MRA, MRB: DTC mode registers A and B
CRA, CRB: DTC transfer count registers A and B
SAR: DTC source address register
DAR: DTC destination address register
DTCERA to DTCERF: DTC enable registers A to F
DTVECR: DTC vector register

3.2.1 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).



Flowchart of DTC Operation

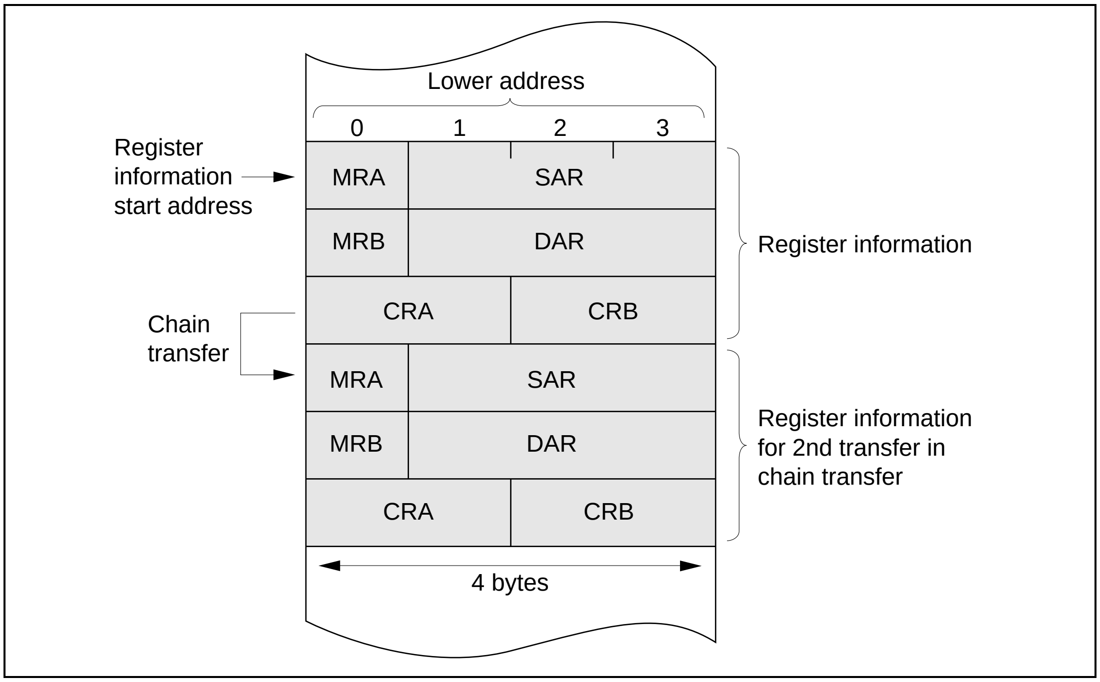
DTC Activation Sources

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'FFF800 to H'FFFBFF).



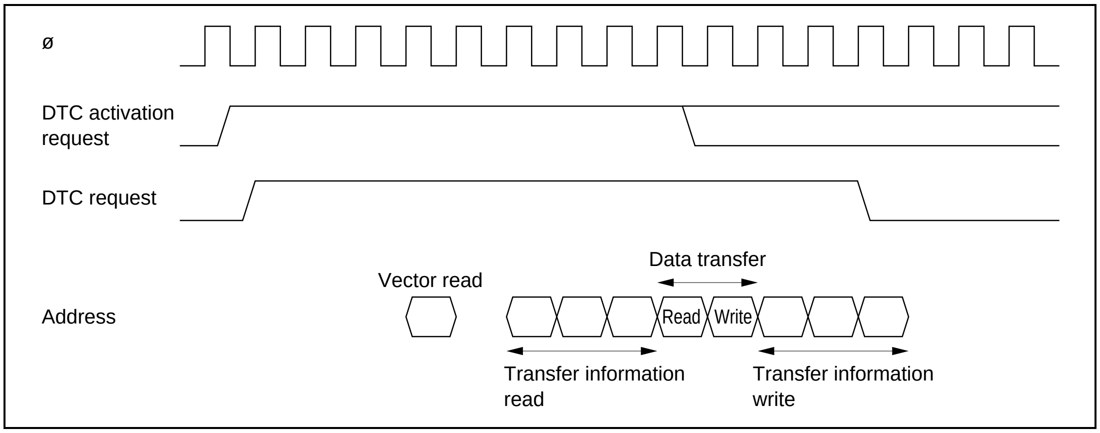
Location of DTC Register Information in Address Space

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	
TGI4A (GR4A compare-match/input capture)	TPU channel 0	56	H'0470	DTCEC1	
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	
CMI0A	8-bit timer channel 0	64	H'0480	DTCED3	
CMI0B		65	H'0482	DTCED2	
CMI1A	8-bit timer channel 1	68	H'0488	DTCED1	
CMI1B		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	Low

Note: * Lower 16 bits of the address.

DTC Operation Timing (Example for Normal and Repeat Modes)



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)

Number of States Required in Each Execution State

Access To			On-Chip RAM	On-Chip ROM	On-Chip 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.

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

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).

3.2.2 Transfer Modes

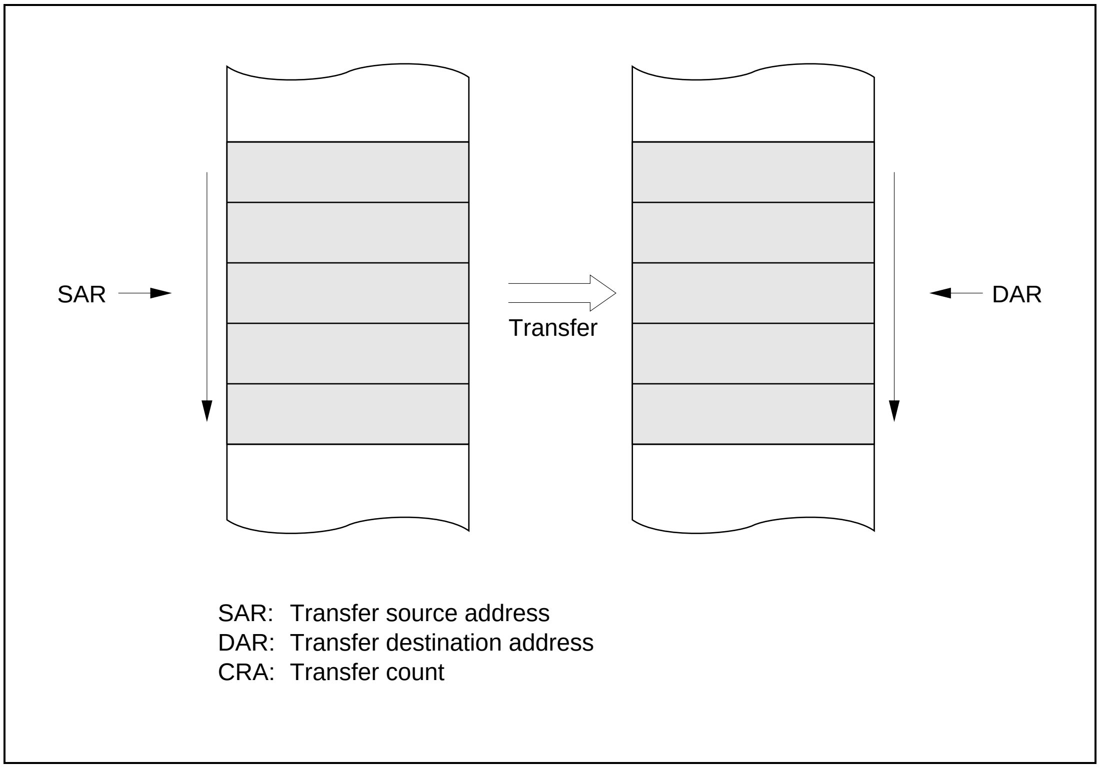
There are three DTC transfer modes—normal mode, repeat mode, and block transfer mode.

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.

		Address Registers	
Transfer Mode	**Activation Source**	**Transfer Source**	**Transfer Destination**
• Normal mode — 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 — 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 — 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	• IRQ • TPU TGI • 8-bit timer CMI • SCI TXI or RXI • A/D converter ADI • Software	24 bits	24 bits

Operation in Normal Mode

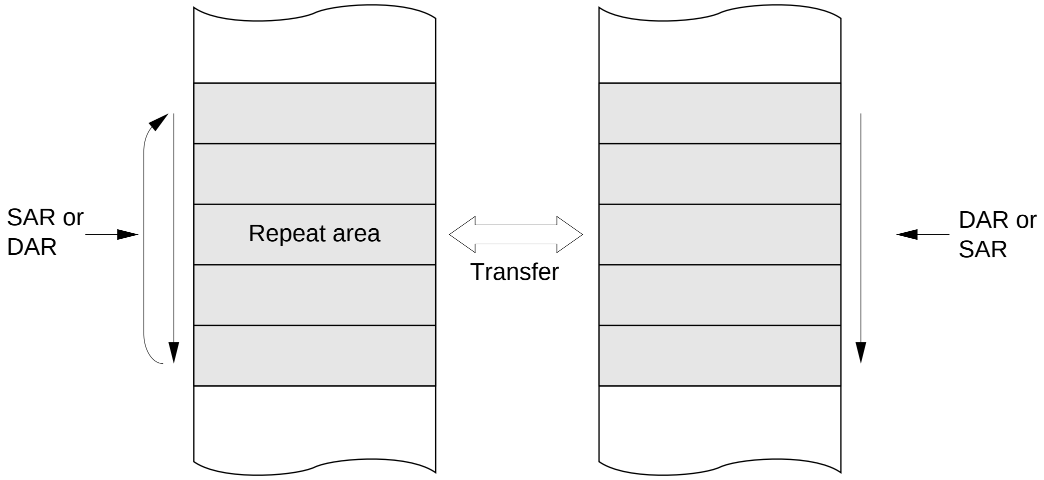
In normal mode, one operation transfers one byte or one word of data. From 1 to 65,536 transfers can be specified. When the specified number of transfers have ended, a CPU interrupt can be requested.



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.



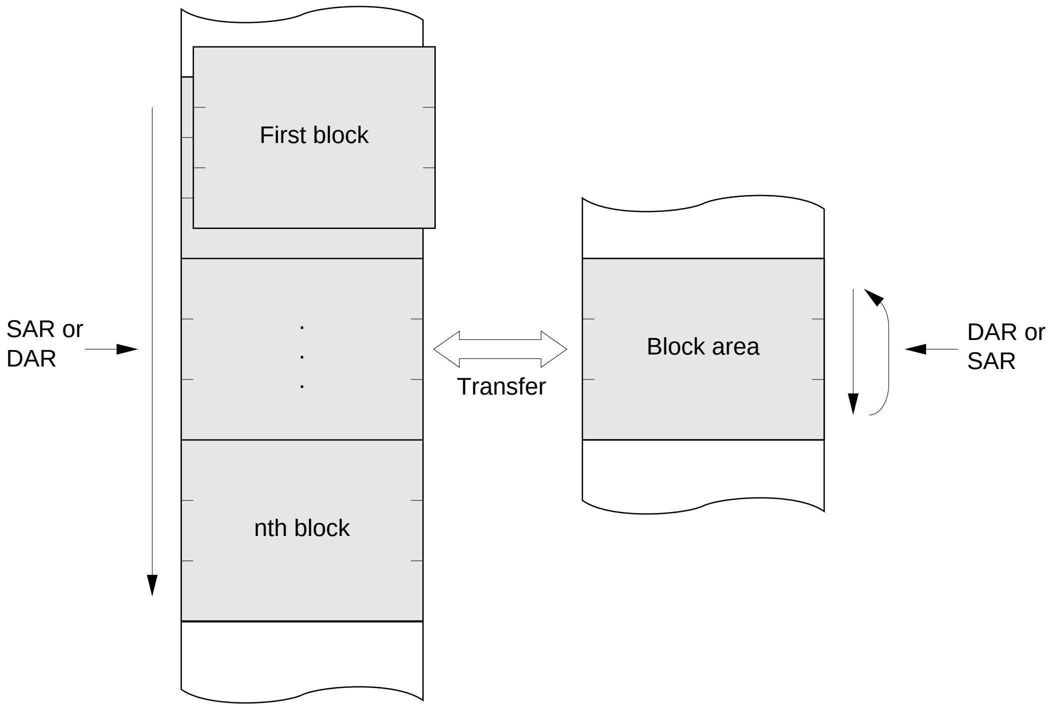
SAR: Transfer source address
DAR: Transfer destination address
CRA: Transfer count (8 bits × 2)

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.



SAR: Transfer source address
DAR: Transfer destination address
CRA: Block size (8 bits × 2)
CRB: Transfer count

Operation in Block Transfer Mode

3.3 16-Bit Timer Pulse Unit (TPU)

The 16-bit timer pulse unit (TPU) that comprises six 16-bit timer channels. The TPU can provide up to 16 kinds of pulse input/output.

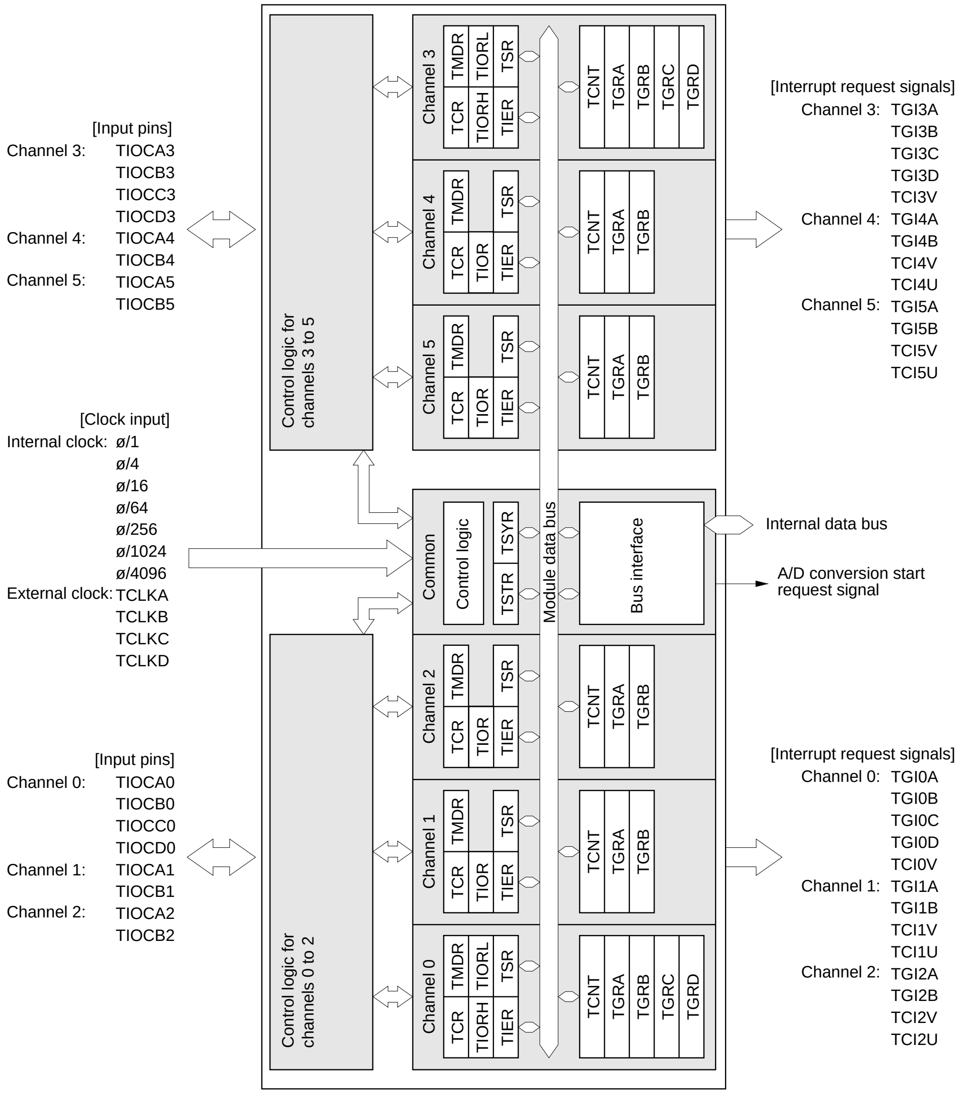
The TPU can perform PWM output, pulse width measurement, and two-phase encoder processing, and can activate the data transfer controller (DTC) . It can also generate an A/D converter start trigger.

Features

- 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: φ, φ/4, φ/16, φ/64, φ/256, φ/1024, φ/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

TPU Block Diagram



Legend
TSTR: Timer start register
TSYR: Timer synchro register
TCR: Timer control register
TMDR: Timer mode register
TIOR (H, L): Timer I/O control register (H, L)
TIER: Timer interrupt enable register
TSR: Timer status register
TGR (A, B, C, D): Timer general registers (A, B, C, D)

Interrupt Sources and Data Transfer Controller (DTC) Activation

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	TGR0Dinput capture/compare-match	Possible	
	TCI0V	TCNT0 overflow	Not possible	
1	TGI1	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	TGR3Dinput 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

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

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

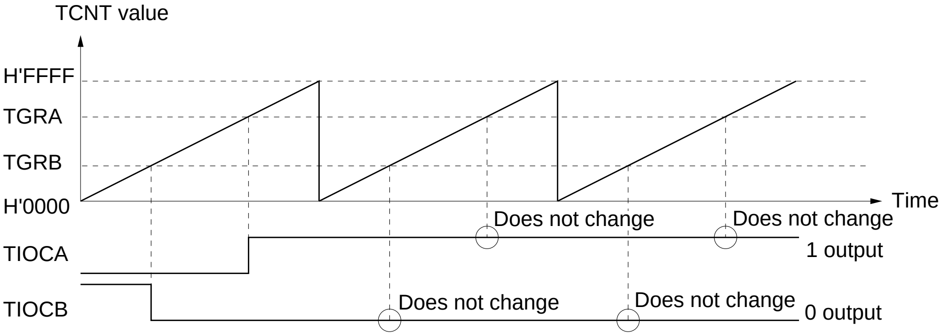
  When a compare-match occurs, the value in the buffer register for the relevant channel is transferred to TGR.
- When TGR is an input capture register

  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

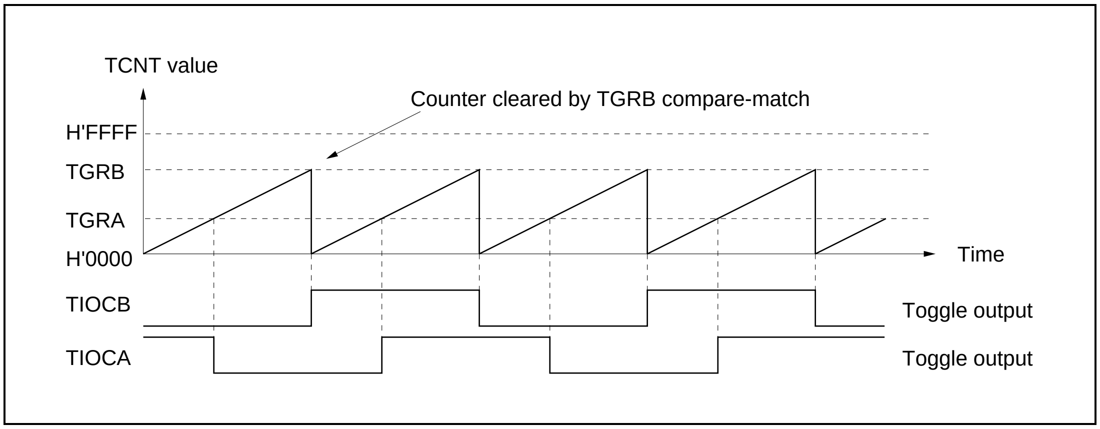
0, 1, or toggle output can be selected.

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.



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.



Example of Toggle Output Operation

PWM Modes

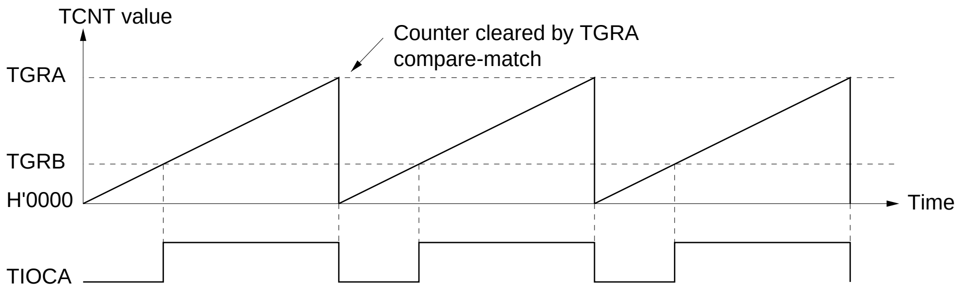
In PWM mode, PWM waveforms are output from the output pins. There are two PWM modes—PWM mode 1 with a maximum of 8-phase pulse output, and PWM mode 2 with a maximum of 15-phase pulse output.

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.

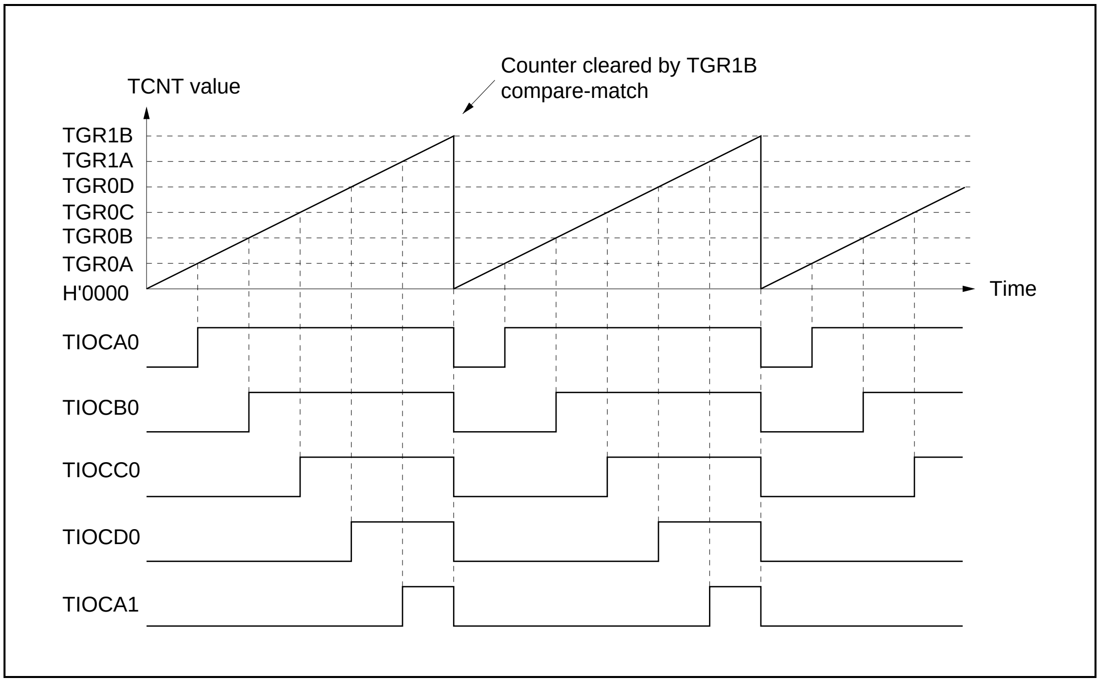


Operation in PWM Mode 1

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.



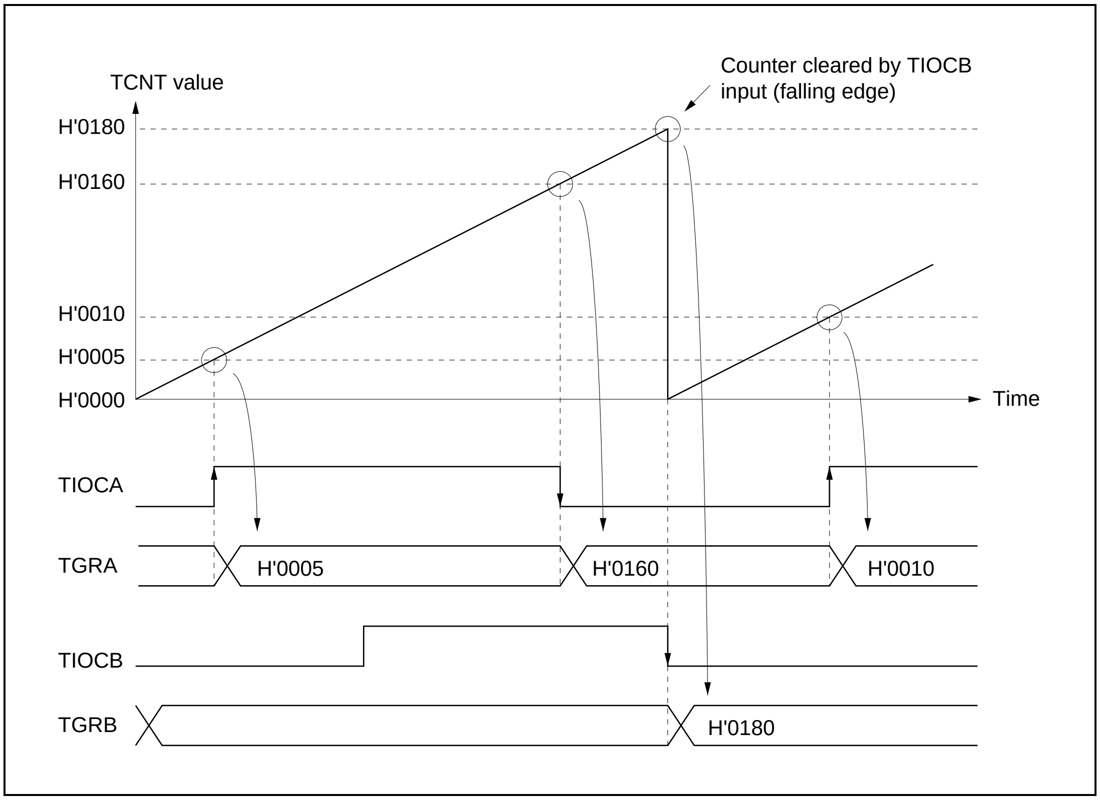
Operation in PWM Mode 2

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.

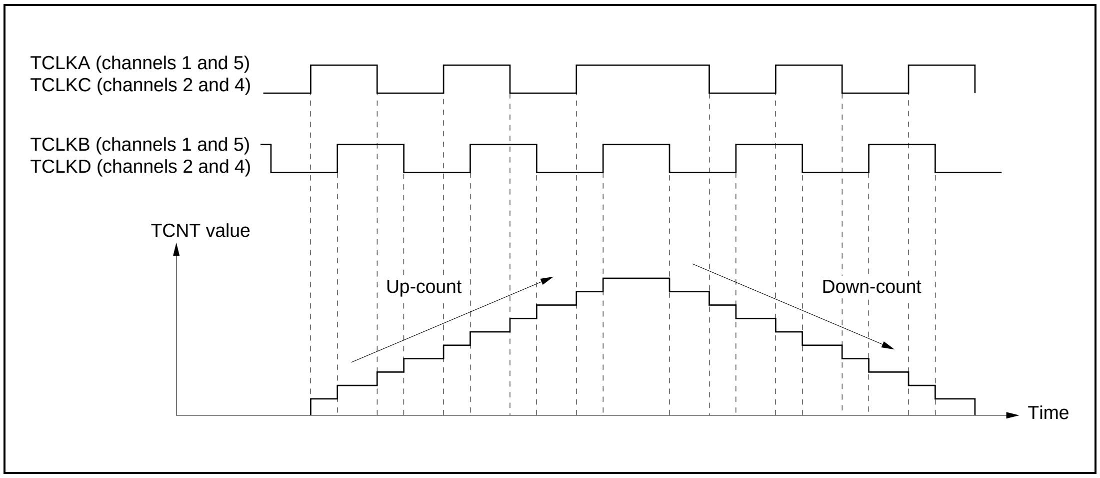


Input Capture Operation

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.

Example of Operation in Phase Counting Mode 1



- Up/Down-Count Conditions in Phase Counting Mode 1

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

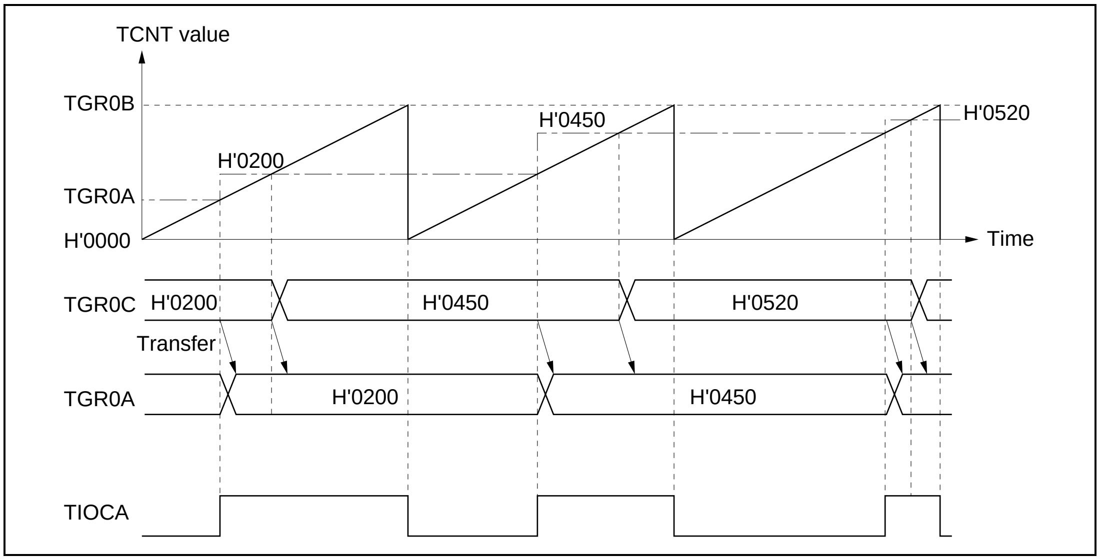
— : Don't care

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)

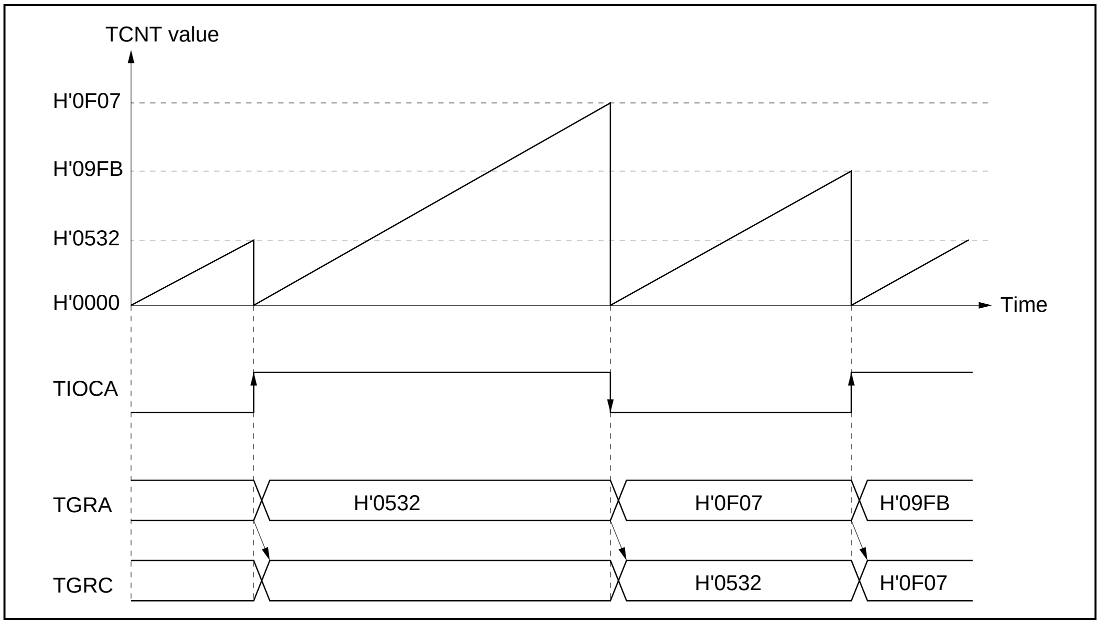
  In this example, PWM mode 1 has been designated for channel 0, and buffer operation has been designated for TGRA and TGRC. The settings used are TCNT clearing by a compare-match B, 1 output at compare-match A, and 0 output at compare-match B. When a compare-match A occurs, the output is changed and the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



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.

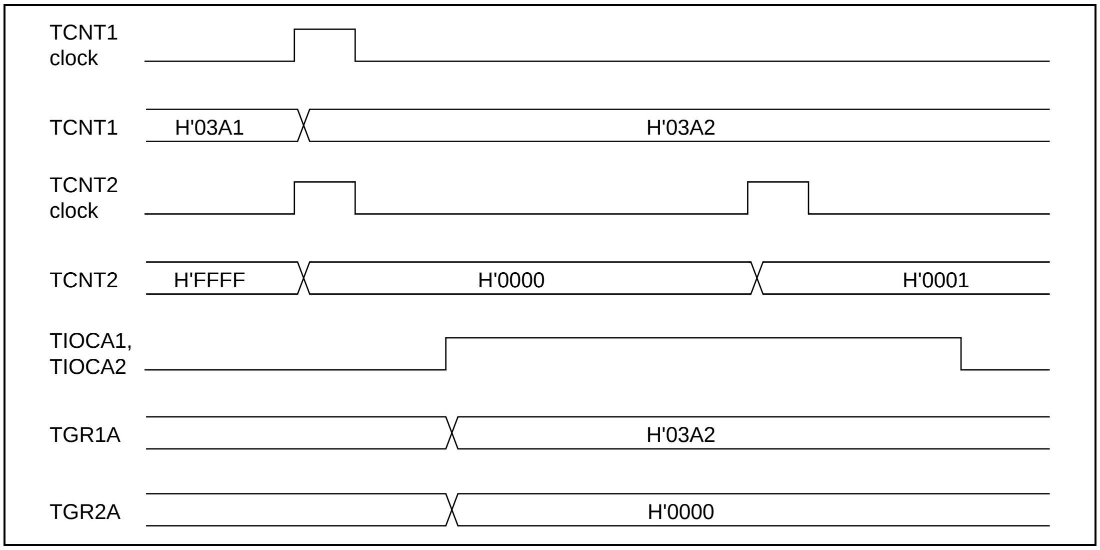


Example of Buffer Operation (2) (When TGR Is an Input Capture Register)

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.



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.

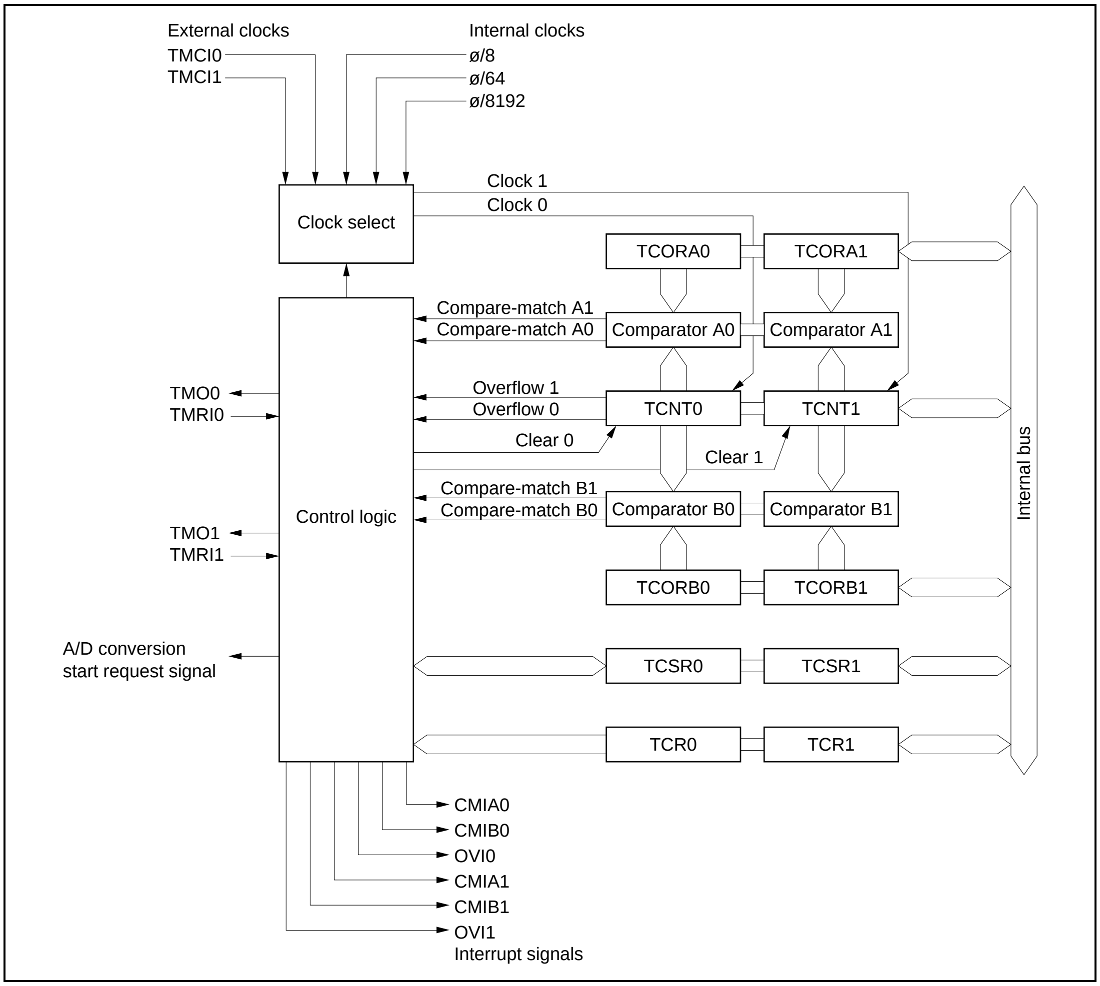
3.4 8-Bit Timer

The H8S/2355 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.

Features

- Selection of four input clock sources
  - The clock source can be selected from three internal clock signals (ϕ/8, ϕ/64, or ϕ/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.

8-Bit Timer Block Diagram



Interrupt Source and Data Transfer Controller (DTC) Activation

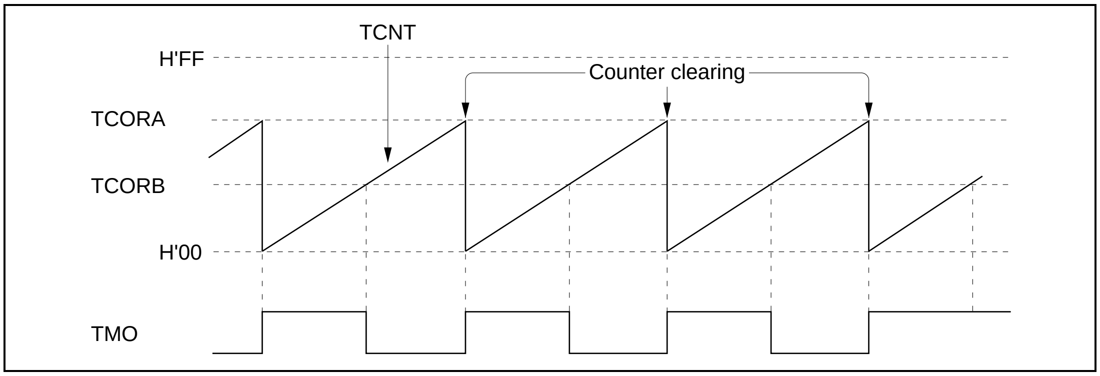
8-Bit Timer Interrupts

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	
	CMIB1	Interrupt by CMFB	Possible	
	OVI1	Interrupt by OVF	Not possible	Low

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

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 below pulses can be output continuously without software intervention.



Example of Pulse Output

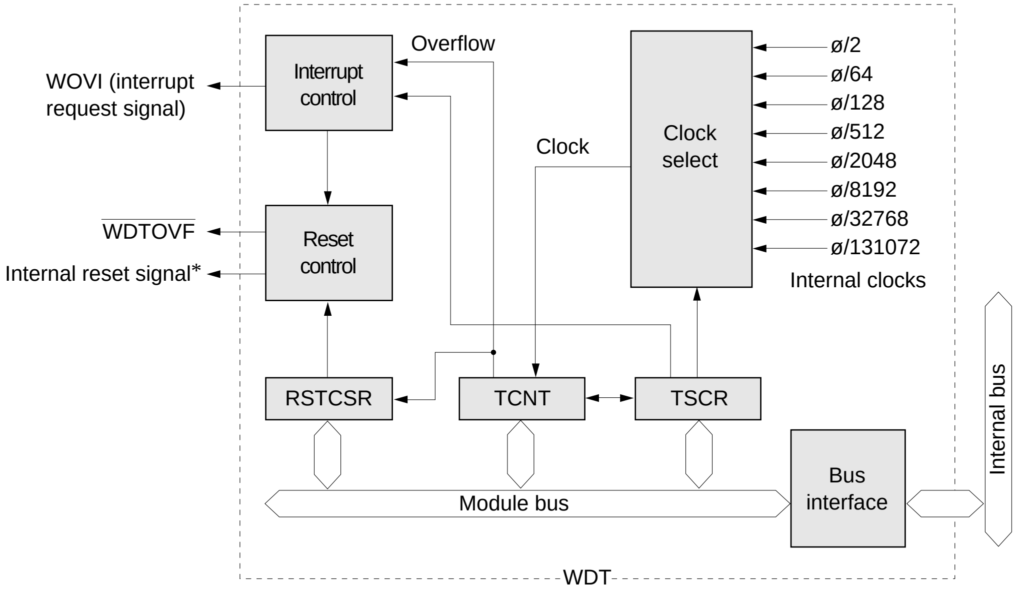
3.5 Watchdog Timer

The H8S/2355 Series can perform system monitoring using its watchdog timer (WDT). When not used as a watchdog timer, this module can be used as an interval timer.

Features

- Selection of eight counter clock sources
  - ϕ/2, ϕ/64, ϕ/128, ϕ/512, ϕ/2048, ϕ/8192, ϕ/32768, ϕ/131072
- Can be used as an interval timer
- $\overline{\text{WDTOVF}}$ signal output in watchdog timer mode
  - When the counter overflows, the WDT outputs $\overline{\text{WDTOVF}}$ signal externally. It is possible to select whether or not the entire chip is reset at the same time. Power-on reset or manual reset can be selected as the internal reset.
- Interrupt generation in interval timer mode
  - When the counter overflows, the WDT generates an interval timer interrupt.

Watchdog Timer Block Diagram



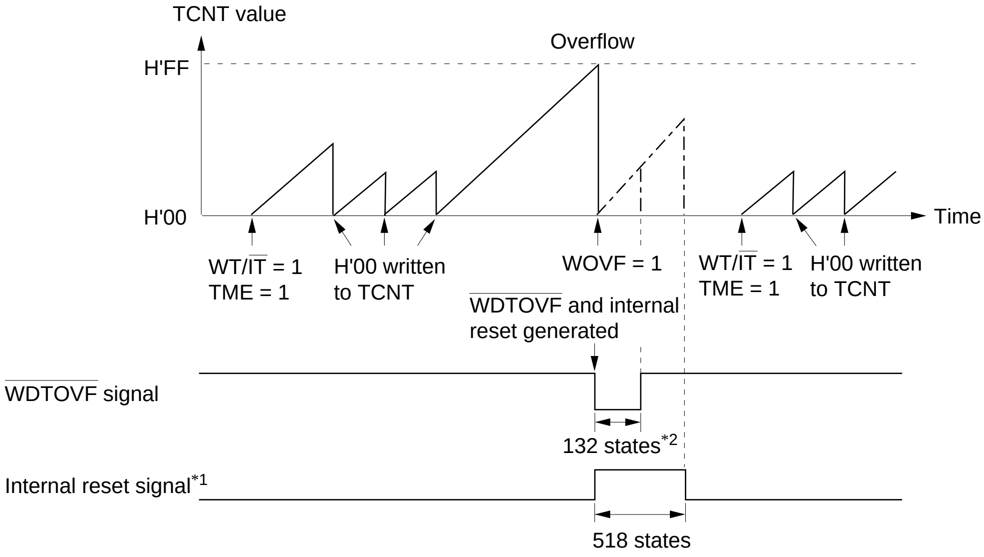
Legend

TCSR:	Timer control/status register
TCNT:	Timer counter
RSTCSR:	Reset control/status register

Note: * The internal reset signal can be generated by a register setting. Either power-on reset or manual reset can be selected.

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.

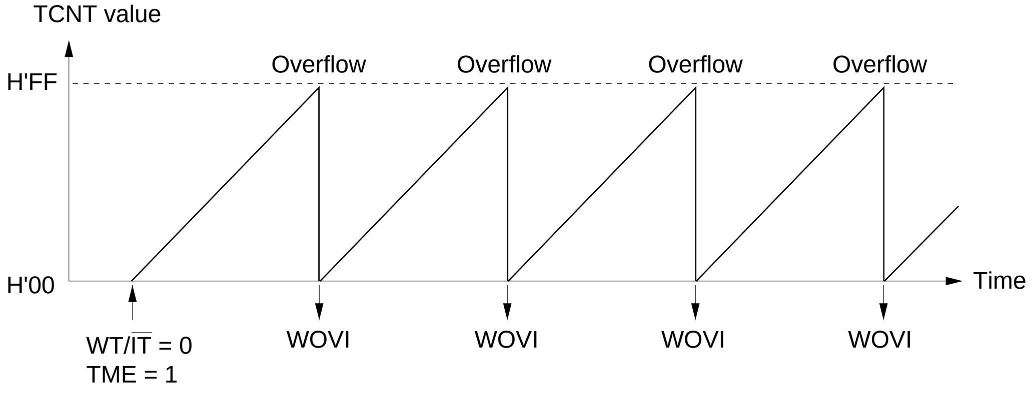


WT/$\overline{\text{IT}}$: Timer mode select bit
TME: Timer enable bit

Note: 1. The internal reset signal is generated only if the RSTE bit is set to 1.
2. 130 states when the RSTE bit is cleared to 0.

Interval Timer Operation

The example below shows this module used as an interval timer. The timer counter (TCNT) starts counting up using the specified clock, and an interval timer request (WOVI) is generated each time TCNT overflows. This function can be used to generate interrupt requests at regular intervals.



WOVI: Interval timer interrupt request generation

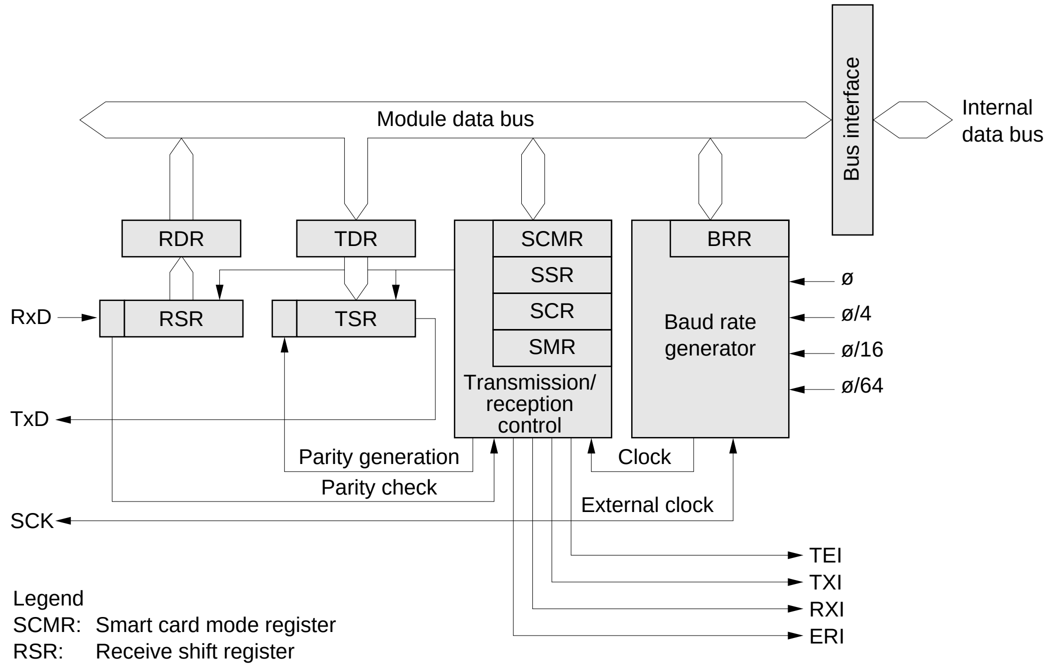
3.6 Serial Communication Interface (SCI)

The H8S/2355 Series is equipped with a three-channel serial communication interface (SCI). All three channels have the same functions, and can handle both asynchronous and synchronous serial communication. A function is also provided for serial communication between processors (multiprocessor communication function).

Features

- Selection of synchronous or asynchronous 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:DMA controller (DMAC) or data transfer controller (DTC) to execute data transfer
  - 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
- Choice of LSB-first or MSB-first transfer
  - Can be selected regardless of the communication mode (except in the case of asynchronous mode 7 bit data)

SCI Block Diagram



Legend
SCMR: Smart card mode register
RSR: Receive shift register
RDR: Receive data register
TSR: Transmit shift register
TDR: Transmit data register
SMR: Serial mode register
SCR: Serial control register
SSR: Serial status register
BRR: Bit rate register

SCI Block Diagram (One Channel)

SCI Interrupt Sources

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

Note: * This table shows the initial state immediately after a reset. Relative priorities among channels can be changed by means of interrupt controller.

3.6.1 SCI Asynchronous Mode

There are two SCI operating modes—asynchronous mode and synchronous mode. Asynchronous mode is described here.

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

Features

- 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 internal baud rate generator or external clock from SCK pin as clock source
- Transmit/receive clock can be output from SCK pin
- Break detection capability
  - Break can be detected by reading the RxD pin level directly in case of a framing error
- Multiprocessor communication capability

Transfer Format and Frame Length in Asynchronous Communication

SMR Settings				Serial Transmit/Receive 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 MPB (multiprocessor bit) added. It then sends transmit data as data with a 0 MPB added.

Receiving stations skip data until data with a 1 MPB is received. Each receiving station then compares that data with its own ID. The station whose ID matches then continues with reception, and accepts data. Stations whose ID does not match continue to skip the data until data with a 1 MPB is sent again.

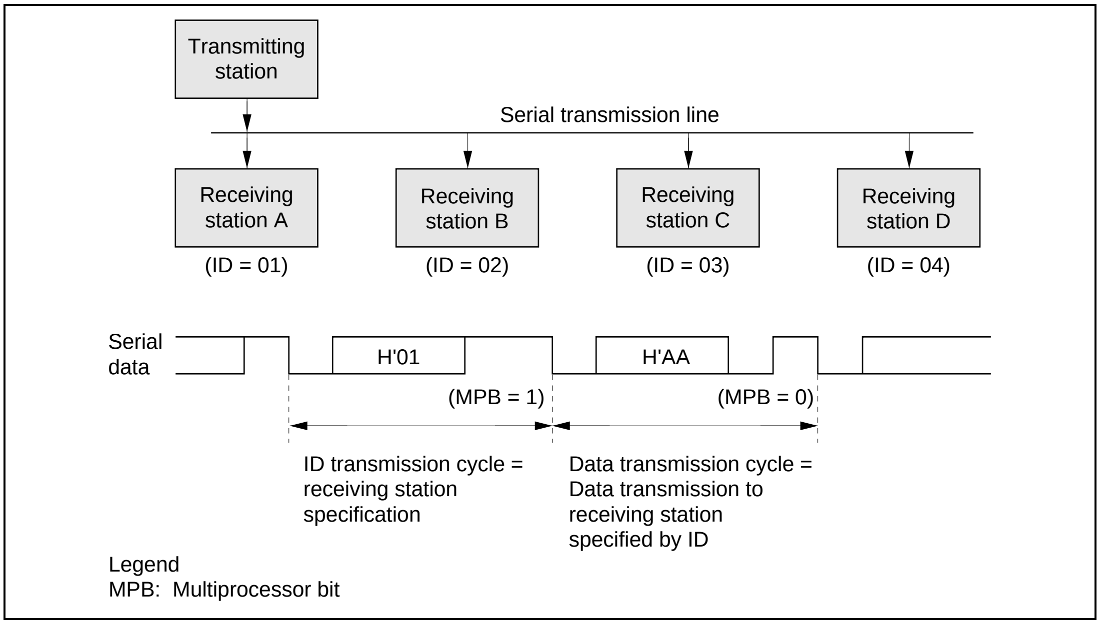
3.6.2 SCI Synchronous Communication

There are two SCI operating modes—asynchronous mode and synchronous mode. Synchronous mode is described here.

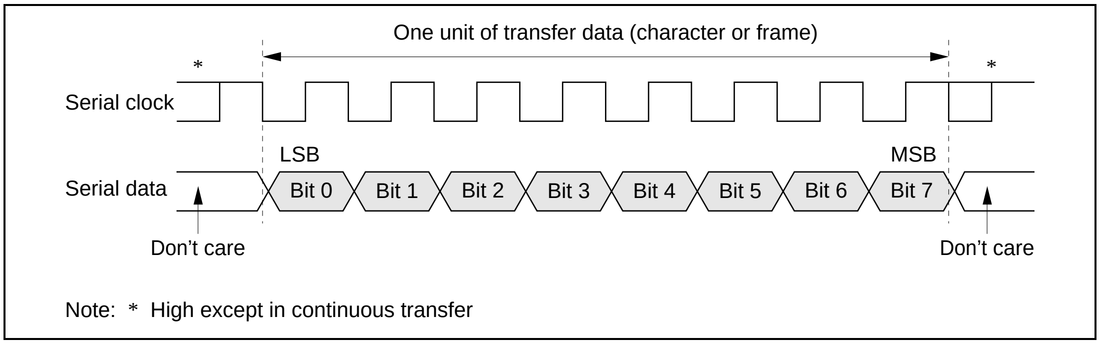
In synchronous mode, data is transmitted or received in synchronization with clock pulses, making it suitable for high-speed serial communication.

- Data length: 8 bits per character
- Overrun error detection
- Selection of internal baud rate generator or external clock from SCK pin as transmit/receive clock source
- Choice 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

When the internal baud rate generator is selected, the SCK pin is automatically set to output mode, and outputs eight synchronization clock pulses.



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



Data Format in Synchronous Communication

Sample BRR Settings for Various Bit Rates (Synchronous Mode)

	ϕ (MHz)											
	2		**4**		**8**		**10**		**16**		**20**	
Bit Rate (bit/s)	**n**	**N**	**n**	**N**	**n**	**N**	**n**	**N**	**n**	**N**	**n**	**N**
110	3	70	—	—	—	—	—	—	—	—	—	—
250	2	124	2	249	3	124	—	—	3	249	—	—
500	1	249	2	124	2	249	—	—	3	124	—	—
1 k	1	124	1	249	2	124	—	—	2	249	—	—
2.5 k	0	199	1	99	1	199	1	249	2	99	2	124
5 k	0	99	0	199	1	99	1	124	1	199	1	249
10 k	0	49	0	99	0	199	0	249	1	99	1	124
25 k	0	19	0	39	0	79	0	99	0	159	0	199
50 k	0	9	0	19	0	39	0	49	0	79	0	99
100 k	0	4	0	9	0	19	0	24	0	39	0	49
250 k	0	1	0	3	0	7	0	9	0	15	0	19
500 k	0	0*	0	1	0	3	0	4	0	7	0	9
1 M			0	0*	0	1	—	—	0	3	0	4
2.5 M					—	—	0	0*	—	—	0	1
5 M									—	—	0	0*

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

The BRR setting is found from the following formula:

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

Legend

Blank:	Cannot be set.
—:	Can be set, but there will be a degree of error.
*	Continuous transfer is not possible.
N:	Baud rate generator setting ($0 \le N \le 255$)
ϕ:	Operating frequency (MHz)
B:	Bit rate (bit/s)
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	ϕ
1	ϕ/4
2	ϕ/16
3	ϕ/64

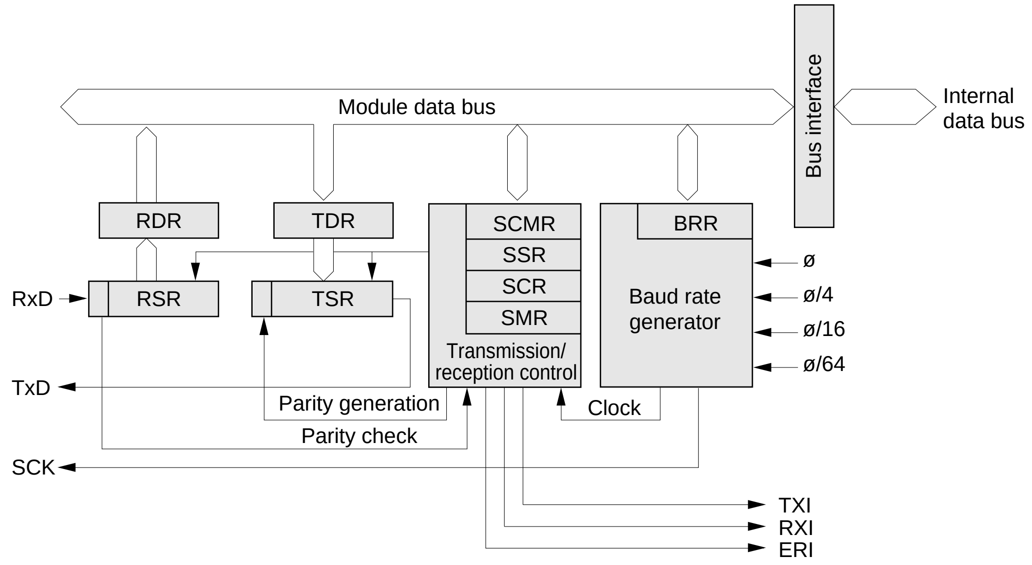
3.7 Smart Card Interface

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

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

Smart Card Interface Block Diagram



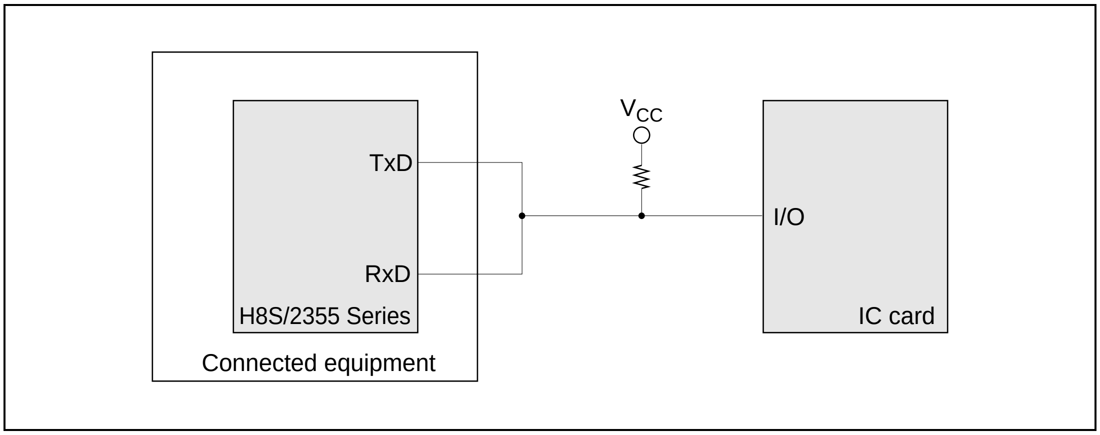
Legend

SCMR:	Smart card mode register	SMR:	Serial mode register
RSR:	Receive shift register	SCR:	Serial control register
RDR:	Receive data register	SSR:	Serial status register
TSR:	Transmit shift register	BRR:	Bit rate register
TDR:	Transmit data register		

Outline of Operation

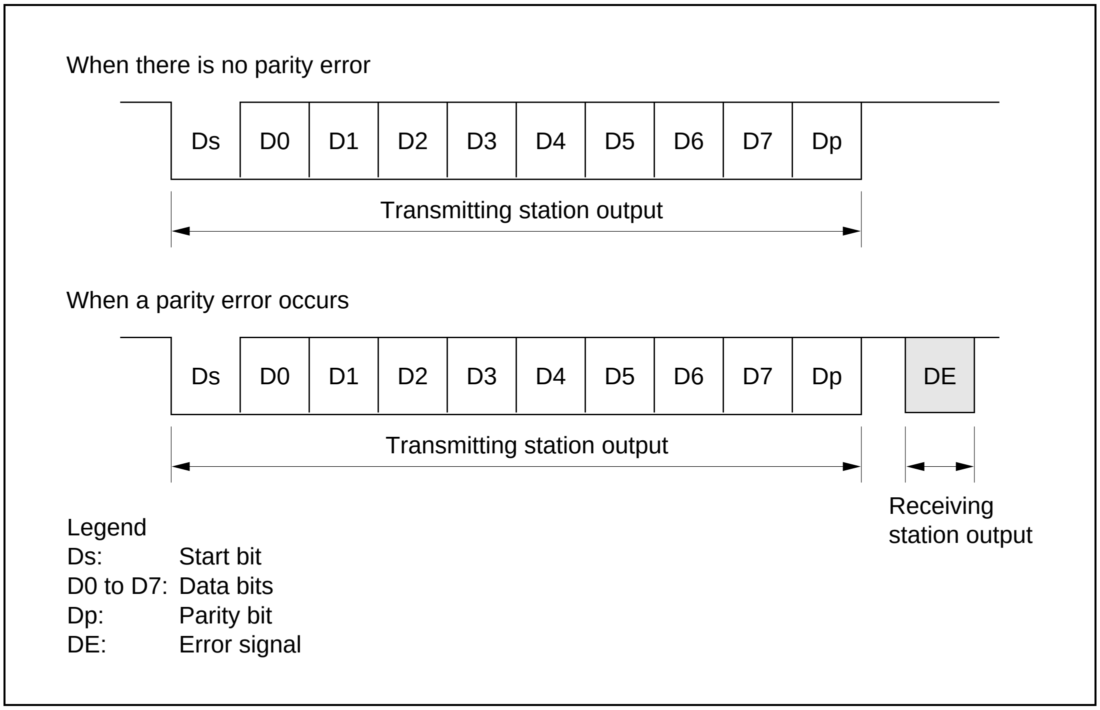
- Only asynchronous communication is supported, with one frame consisting of 8-bit data plus a parity bit.
- In transmission, a guard time of at least 2 etu (Elementary Time Unit: the time for transfer of one bit) is left between the end of the parity bit and the start of the next frame.
- If a parity 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.

Schematic Connection Diagram



Schematic Diagram of Smart Card Interface Pin Connections

Data Format



Smart Card Interface Data Format

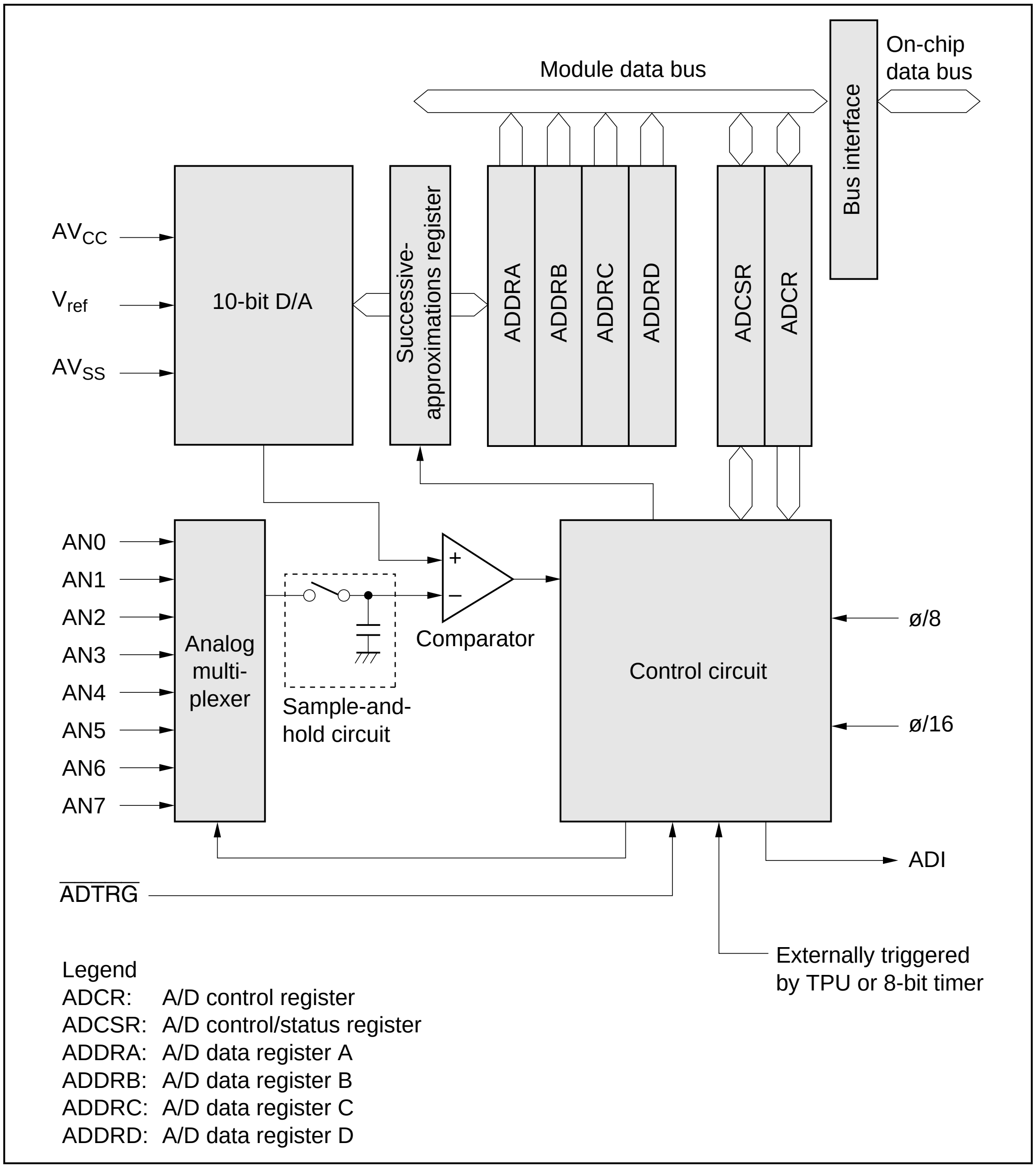
3.8 A/D Converter

The H8S/2355 Series has an on-chip A/D converter with 10-bit precision. Analog signals can be input on up to eight channels by the program.

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:
    - 6.7 µs per channel (at 20 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

A/D Converter Block Diagram



Legend

ADCR:	A/D control register
ADCSR:	A/D control/status register
ADDRA:	A/D data register A
ADDRB:	A/D data register B
ADDRC:	A/D data register C
ADDRD:	A/D data register D

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.

			Description	
Bit 2 CH2	Bit 1 CH1	Bit 0 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.

3.9 D/A Converter

The H8S/2355 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.

Features

- Eight-bit resolution
- Two output channels
- Maximum conversion time of 10 µs (with 20 pF load capacitance)
- Output voltage of 0 V to V_{ref}
- D/A output hold function in software standby mode

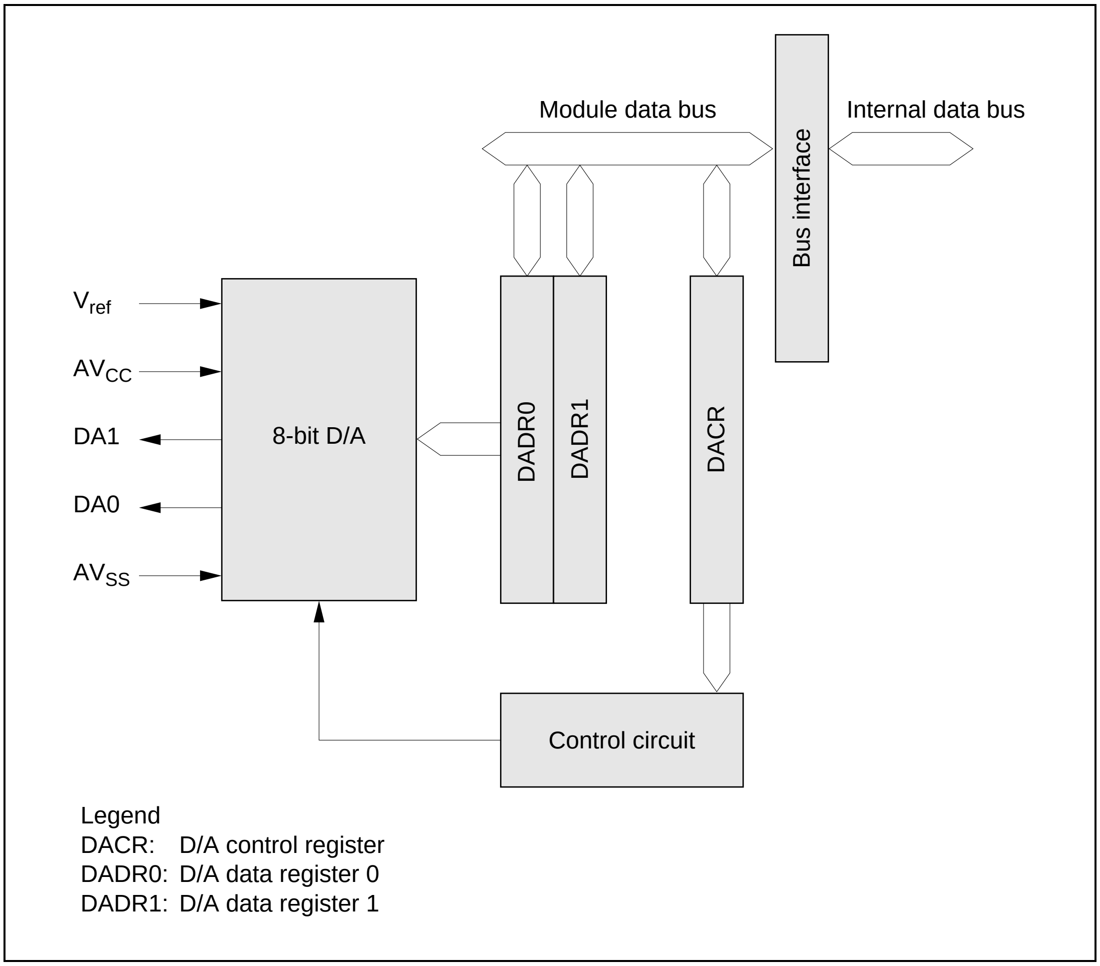
Operation

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:

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

D/A Converter Block Diagram



Block Diagram of D/A Converter

3.10 I/O Ports

The H8S/2355 Series has twelve I/O ports (ports 1, 2, 3, 5, 6, and A to G), and one input-only port (port 4). The ports also function as bus control pins and on-chip supporting module I/O pins.

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.

Port Functions in Each Operating Mode

Port Functions

Port	Description	Pins	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
Port 1	• 8-bit I/O port	$P1_7$/TIOCB2/TLCKD $P1_6$/TIOCA2 $P1_5$/TIOCB1/TLCKC $P1_4$/TIOCA1 $P1_3$/TIOCD0/TLCKB $P1_2$/TIOCC0/TLCKA $P1_1$/TIOCB0 $P1_0$/TIOCA0	8-bit I/O port multiplexed as TPU I/O pins (TCLKA, TCLKB, TCLKC, TCLKD, TIOCA0, TIOCB0, TIOCC0, TIOCD0, TIOCA1, TIOCB1, TIOCA2, TIOCB2)						
Port 2	• 8-bit I/O port • Schmitt-triggered input	$P2_7$/TIOCB5/TMO1 $P2_6$/TIOCA5/TMO0 $P2_5$/TIOCB4/TMCI1 $P2_4$/TIOCA4/TMRI1 $P2_3$/TIOCD3/TMCI0 $P2_2$/TIOCC3/TMRI0 $P2_1$/TIOCB3 $P2_0$/TIOCA3	8-bit I/O port multiplexed as TPU I/O pins (TIOCA3, TIOCB3, TIOCC3, TIOCD3, TIOCA4, TIOCB4, TIOCA5, TIOCB5), and 8-bit timer (channels 0 and 1) I/O pins (TMRI0, TMCI0, TMO0, TMRI1, TMCI1, TMO1)						
Port 3	• 6-bit I/O port • Open-drain output capability	$P3_5$/SCK1 $P3_4$/SCK0 $P3_3$/RxD1 $P3_2$/RxD0 $P3_1$/TxD1 $P3_0$/TxD0	6-bit I/O port multiplexed as SCI (channels 0 and 1) I/O pins (TxD0, RxD0, SCK0, TxD1, RxD1, SCK1)						

Port	Description	Pins	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
Port 4	• 8-bit I/O port	$P4_7$/AN7/DA1 $P4_6$/AN6/DA0 $P4_5$/AN5 $P4_4$/AN4 $P4_3$/AN3 $P4_2$/AN2 $P4_1$/AN1 $P4_0$/AN0	8-bit input port multiplexed as A/D converter analog inputs (AN7 to AN0) and D/A converter analog outputs (DA1 and DA0)						
Port 5	• 4-bit I/O port	$P5_3$/$\overline{\text{ADTRG}}$ $P5_2$/SCK2 $P5_1$/RxD2 $P5_0$/TxD2	4-bit I/O port multiplexed as A/D converter input pin ($\overline{\text{ADTRG}}$) and SCI (channel 2) I/O pins (TxD2, RxD2, SCK2)						
Port 6	• 8-bit I/O port • Schmitt-triggered input ($P6_4$ to $P6_7$)	$P6_7$/$\overline{\text{IRQ3}}$/$\overline{\text{CS7}}$ $P6_6$/$\overline{\text{IRQ2}}$/$\overline{\text{CS6}}$ $P6_5$/$\overline{\text{IRQ1}}$ $P6_4$/$\overline{\text{IRQ0}}$ $P6_3$ $P6_2$ $P6_1$/$\overline{\text{CS5}}$ $P6_0$/$\overline{\text{CS4}}$	8-bit I/O port multiplexed as interrupt input pins ($\overline{\text{IRQ0}}$ to $\overline{\text{IRQ3}}$)			8-bit I/O port multiplexed as bus control output pins ($\overline{\text{CS4}}$ to $\overline{\text{CS7}}$) and interrupt input pins ($\overline{\text{IRQ0}}$ to $\overline{\text{IRQ3}}$)			8-bit I/O port multiplexed as interrupt input pins ($\overline{\text{IRQ0}}$ to $\overline{\text{IRQ3}}$)
Port A	• 8-bit I/O port • Built-in MOS input pull-up • Open-drain output capability • Schmitt-triggered input (PA_4 to PA_7)	PA_7/A_{23}/$\overline{\text{IRQ7}}$ PA_6/A_{22}/$\overline{\text{IRQ6}}$ PA_5/A_{21}/$\overline{\text{IRQ5}}$	Multiplexed as I/O port and interrupt input pins ($\overline{\text{IRQ7}}$ to $\overline{\text{IRQ4}}$)			When DDR = 0 (after reset): multiplexed as input port and interrupt input pins ($\overline{\text{IRQ7}}$ to $\overline{\text{IRQ5}}$) When DDR = 1: address output		When DDR = 0 (after reset): multiplexed as input port and interrupt input pins ($\overline{\text{IRQ7}}$ to $\overline{\text{IRQ4}}$)	Multiplexed as I/O port and interrupt input pins ($\overline{\text{IRQ7}}$ to $\overline{\text{IRQ4}}$)
	•	PA_4/A_{20}/$\overline{\text{IRQ4}}$				Address output		When DDR = 1: address output	
	•	PA_3/A_{19} to PA_0/A_{16}	I/O port			Address output		When DDR = 0 (after reset): input port When DDR = 1: address output	I/O port
Port B	• 8-bit I/O port • • Built-in MOS input pull-up	PB_7/A_{15} to PB_0/A_8	Address output	When DDR = 0 (after reset): input port When DDR = 1: address output	I/O port	Address output		When DDR = 0 (after reset): input port When DDR = 1: address output	I/O port

Port	Description	Pins	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
Port C	• 8-bit I/O port • Built-in MOS input pull-up	PC_7/A_7 to PC_0/A_0	Address output	When DDR = 0 (after reset): input port When DDR = 1: address output	I/O port	Address output		When DDR = 0 (after reset): input port When DDR = 1: address output	I/O port
Port D	• 8-bit I/O port • Built-in MOS input pull-up	PD_7/D_{15} to PD_0/D_8	Data bus input/output		I/O port	Data bus input/output			I/O port
Port E	• 8-bit I/O port • Built-in MOS input pull-up	PE_7/D_7 to PE_0/D_0	In 8-bit bus mode: I/O port In 16-bit bus mode: data bus input/output		I/O port	In 8-bit bus mode: I/O port In 16-bit bus mode: data bus input/output			I/O port
Port F	• 8-bit I/O port	PF_7/ϕ	When DDR = 0: input port When DDR = 1 (after reset): ϕ output		When DDR = 0 (after reset): input port When DDR = 1: ϕ output	When DDR = 0: input port When DDR = 1 (after reset): ϕ output			When DDR = 0 (after reset): input port When DDR = 1: ϕ output
		$PF_6/\overline{AS}$ $PF_5/\overline{RD}$ $PF_4/\overline{HWR}$ $PF_3/\overline{LWR}$	$\overline{AS}$, $\overline{RD}$, $\overline{HWR}$, $\overline{LWR}$ output		I/O port	$\overline{AS}$, $\overline{RD}$, $\overline{HWR}$, $\overline{LWR}$ output			I/O port
		$PF_2/\overline{WAIT}$	When WAITE = 0 (after reset): I/O port When WAITE = 1: $\overline{WAIT}$ input			When WAITE = 0 (after reset): I/O port When WAITE = 1: $\overline{WAIT}$ input			
		$PF_1/\overline{BACK}$ $PF_0/\overline{BREQ}$	When BRLE = 0 (after reset): I/O port When BRLE = 1: $\overline{BREQ}$ input, $\overline{BACK}$ output			When BRLE = 0 (after reset): I/O port When BRLE = 1: $\overline{BREQ}$ input, $\overline{BACK}$ output			
Port G	• 5-bit I/O port	$PG_4/\overline{CS0}$	When DDR = 0^{*1}: input port When DDR = 1^{*2} (after reset): $\overline{CS0}$ output		I/O port	When DDR = 0^{*1}: input port When DDR = 1^{*2} (after reset): $\overline{CS0}$ output			I/O port
		$PG_3/\overline{CS1}$ $PG_2/\overline{CS2}$ $PG_1/\overline{CS3}$	I/O port			When DDR = 0 (after reset): input port When DDR = 1: $\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$ output			
		PG_0				I/O port			

Notes 1. After a reset in mode 2 or 6.

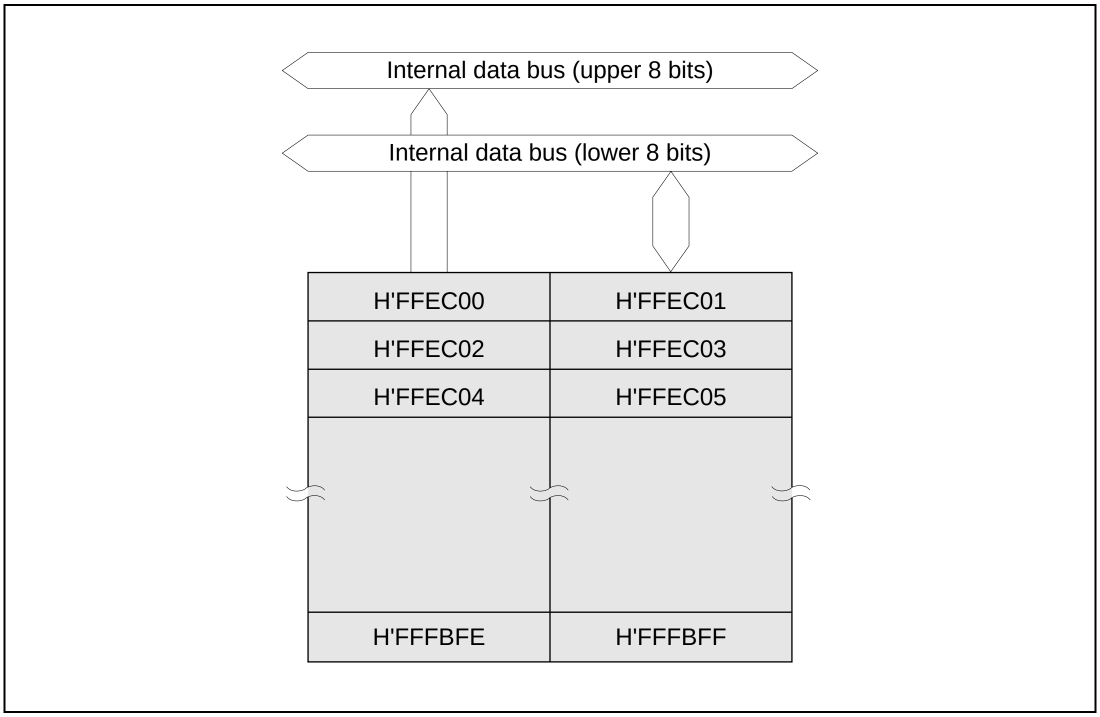
2. After a reset in mode 1, 4, or 5.

3.11 RAM

The H8S/2355 has 4 kbytes of on-chip high-speed static RAM, and H8S/2353 has 2 kbytes. 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).

Block Diagram of RAM (Example with H8S/2355)

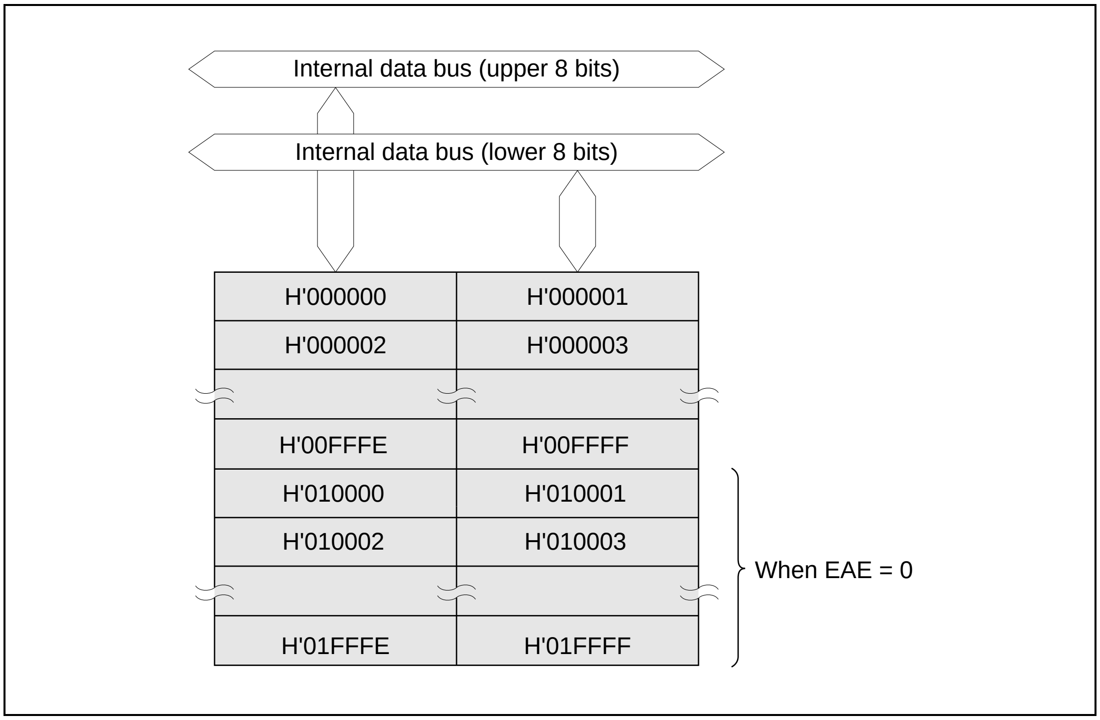


3.12 ROM (PROM)

The H8S/2355 has 128 kbytes of on-chip ROM (PROM or mask ROM), and the H8S/2353 has 64 kbytes. 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.

In normal mode, a maximum of 56 kbytes of ROM can be used.

Block Diagram of ROM



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 120- or 128-pin/32-pin socket adapter enables programming with a commercial PROM programmer. The address range is H'00000 to H'0FFFF. However, page programming is not supported.

Section 4 Power-Down State

4.1 Power-Down State

In addition to the normal program execution state, the H8S/2355 Series has a power-down state in which operation of the CPU and oscillator is halted and power consumption is reduced. The CPU, on-chip peripheral functions, etc., are controlled individually, enabling low-power operation to be achieved. The power-down state includes medium-speed mode, sleep mode, module stop mode, software standby mode, and hardware standby mode.

Medium-Speed Mode: When the SCK2 to SCK0 bits in the system clock control register (SCKCR) are set to 1, 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 (φ/2, φ/4, φ/8, φ/16, or φ/32) specified by the SCK2 to SCK0 bits. However, on-chip peripheral functions other than the bus masters operate on the high-speed clock (φ).

In medium-speed mode, a bus access is executed in the specified number of states with respect to the bus master operating clock. For example, if φ/4 is selected as the operating clock, on-chip memory is accessed in four states, and internal I/O registers in eight states.

Medium-speed mode is cleared by clearing all of bits SCK2 to SCK0 to 0. High-speed mode is restored at the end of the current bus cycle.

Sleep Mode: If a SLEEP instruction is executed when the SSBY bit in the system standby register (SBYCR) is 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 peripheral functions do not stop.

Sleep mode is cleared by a reset or any interrupt, and the CPU returns to the normal program execution state via the exception handling state.

Module Stop Mode: Module stop mode can be set for individual on-chip peripheral functions.

When the MSTP bit corresponding to a particular peripheral function 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.

When the corresponding MSTP bit is cleared to 0, module stop mode is cleared and the module starts operating at the end of the bus cycle.

In module stop mode, the internal states of modules other than the SCI and A/D are retained.

After a reset, all modules except the DTC are in module stop mode.

Software Standby Mode: If a SLEEP instruction is executed when the SSBY bit in SBYCR is set to 1, software standby mode is entered. In this mode, the CPU, on-chip peripheral functions, and oscillator all stop. However, the contents of the CPU's internal registers, RAM data, and the states of on-chip peripheral functions other than the SCI and A/D, and I/O ports are retained. Software standby mode is cleared by a reset or an external interrupt. After the elapse of the oscillation stabilization time, the program execution mode is restored via the exception handling state.

As the oscillator is stopped in this mode, power consumption is extremely low.

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 prescribed voltage is supplied, on-chip RAM data is retained. I/O ports are set to the high-impedance state.

Hardware standby mode is cleared by means of the $\overline{\text{STBY}}$ pin and the $\overline{\text{RES}}$ pin. When the $\overline{\text{STBY}}$ pin is driven high while the $\overline{\text{RES}}$ pin is low, the reset state is entered and clock oscillation is started. When the $\overline{\text{RES}}$ pin is subsequently driven high, the program execution state is restored via reset exception handling.

In this mode, as in software standby mode, power consumption is extremely low since the oscillator is stopped.

Operating States

				CPU		Modules		
Operating Mode	**Transition Condition**	**Clearing Condition**	**Oscillator**		**Registers**		**Registers**	**I/O Ports**
High-speed mode	Control register		Functions	High speed	Functions	High speed	Functions	High speed
Medium-speed mode	Control register		Functions	Medium speed	Functions	High/ medium speed*1	Functions	High speed
Sleep mode	Instruction	Interrupt	Functions	Halted	Retained	High speed	Functions	High speed
Module stop mode	Control register		Functions	High/ medium speed	Functions	Halted	Retained/ reset *2	Retained
Software standby mode	Instruction	External interrupt	Halted	Halted	Retained	Halted	Retained/ reset*2	Retained
Hardware standby mode	Pin		Halted	Halted	Undefined	Halted	Reset	High impedance

Notes: 1. The bus master operates on the medium-speed clock, and other on-chip peripheral functions operate on the high-speed clock.

2. The SCI and A/D are reset, and other on-chip peripheral functions retain their state.

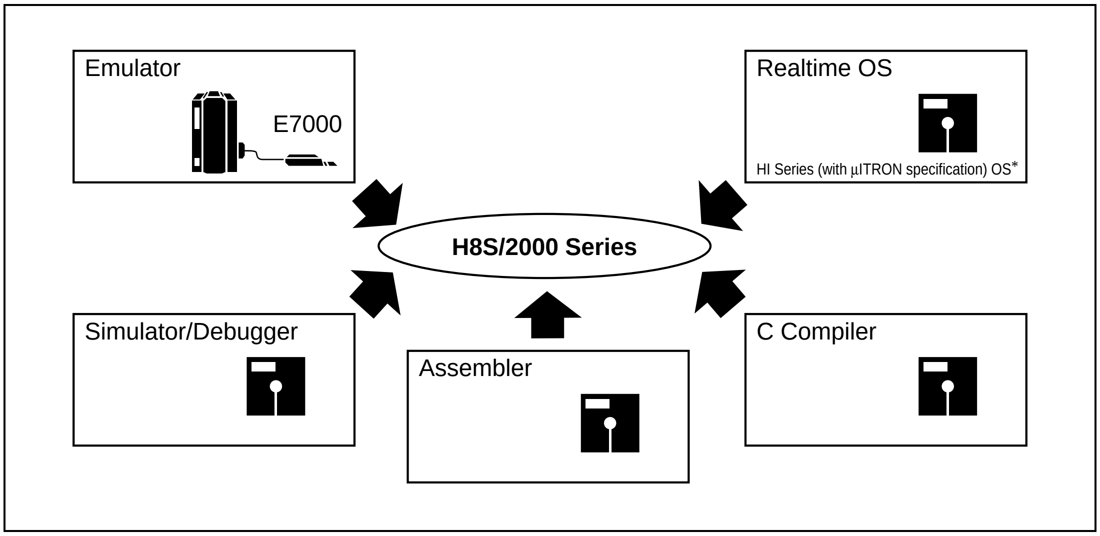
Section 5 Development Environment

5.1 Development Environment

A comprehensive development environment is provided for the H8S/2355 Series, including cross software, emulators, and HI Series OS*.

Lineup

- Cross software
  - C compiler
  - Assembler
  - Simulator/debugger
- Emulators
  - E7000/E7000S
  - E7000PC/E7000S PC
- HI Series OS*
  - HI8-2600



Note: * The HI Series OS is Hitachi realtime 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 the H8S/2355 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 H8/300H 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
  - Intrinsic function facility: H8S/2000 system control instructions can be written in function call format
  - Memory access function: Efficient memory access provided by indirect memory addressing mode and short absolute 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
- 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 all 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 function

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
  - — Stub and function call functions are supported, for easy program debugging in the initial development stage
  - — Support of C0 and C1 coverage functions, single-line assembly, disassembly, state retention, command history, command macros, etc.

Lineup

Host Machine	**PC-9800 Series**	**IBM PC FLORA (DOS/V)**	**SPARC (Sun OS)**	**SPARC (Solaris)**	**HP9000/700 HITACHI 9000**	**NEWS**
Assembler system	PS008ASM1-F3	PS008ASM1-IF3	PS008ASM1-SPC	HSS008ASCS1SM	PS008ASM1-H7D	PS008ASM1-NRF
C compiler	PS008CPC100FE3	PS008CPC100IF3	PS008CSP100	HSS008CLCS1SM	PS008CHP7100	PS008CNR100
Simulator/ debugger	PS008SIM1-F3	PS008SIM1-IF3	PS008SIM1-SPC	HSS008SDCS1SM	PS008SIM1-H7D	PS008SIM1-NRF
HI Series OS	HSS008ITPN2SFB* (for object contact)	HSS008ITIN2SFB* (for object contact (FD))	HSS008ITCN2SFB* (for object contact (FD)) HSS008ITCN2SMB* (for object contact (MT))		HSS008ITHN2STB* (for object contact (DAT))	—
	HSS008ITPN2SFS* (for source contact)	HSS008ITIN2SFS* (for source contact (FD))	HSS008ITCN2SFS* (for source contact (FD)) HSS008ITCN2SMS* (for source contact (MT))		HSS008ITHN2STS* (for source contact (DAT))	—
Multitask debugger	HSS008I7CN1SF* (E7000) HSS008I7IN1SF* (E7000PC)					_—

Note: * Under development.

5.3 Emulators

Various emulators are available for the H8S/2355 Series, including the E7000 incorporating a wide variety of functions, and the E7000PC for development on MS-WINDOWS. Additionally, the E7000S and E7000S PC functionally enhanced versions of the E7000 and E7000PC are also available.

E7000/E7000PC

These comprise an emulator station common to all models, plus an emulator pod and interface cable which can be selected for an individual microcomputer.

An efficient debugging environment is provided by a graphical user interface on a workstation (E7000) or with MS-WINDOWS on an IBM PC or other personal computer (E7000PC).

- Realtime emulation
  - Capable of realtime emulation supporting a maximum bus cycle speed of up to 50ns
- Multiwindow environment
  - Runs on standard workstation GUI (graphical user interface) X-Window (E7000)
  - Runs on standard personal computer GUI (graphical user interface) MS-WINDOWS (E7000PC)
  - Provides a multiwindow debugging environment enabling source program amendment while operating the emulator
  - Comprehensive help functions, for manual-less operation
  - Efficient mouse operations through use of pull-down menus and buttons for frequently used commands
- Fast downloading
  - 1-Mbyte/second downloading
- C source level debugging (graphical user interface)
  - Displays C source program and point at which execution halted
  - Breakpoints can be set or cleared at specified lines in the source program
  - Displays contents of specified variables in the source program
- Powerful debugging functions
  - Break functions: Six independent hardware breaks or 255 software breaks can be set
  - Trace functions: Large trace capacity of 32k cycles; trigger points can be set indiependently of breakpoints
- Graphical display of trace contents
  - Trace information displayed in graph form, facilitating analysis of program and signal operation

- Other features
  - — Performance analysis, enabling execution time or number of times executed to be recorded for up to four subroutines
  - — Parallel mode, enabling memory size to be referenced or changed without halting the program
  - — Support of memory disassembly, single-line assembly, coverage function (C0), etc.

E7000S and E7000S PC

These are functionally enhanced versions of the E7000 and E7000PC emulators.

- Enhanced performance functions
  - — Number of subroutine time measurement modules (maximum: 8 channels), maximum/minimum measurement (4 channels)
  - — Number of subroutine execution count measurement modules (maximum: 8 channels)
  - — Timeout function
  - — Number of run time measurement modules (2 channels)
  - — Sampling time (1 µs, 250 ns, 20 ns)
- Enhanced trace functions
  - — Timestamps
  - — Real time data monitor (256 bytes × 8 areas)
  - — PC trace
  - — Sequential stop function (7 levels + 1 reset point)

Emulator Product Codes

E7000	**Host System Interface Board**	**Expansion Memory**	**Bus Monitor**		**Emulator Pod**	**User Cable**	**Integrated Development Manager**
			Interface Board	**Monitor Board**			
HS7000EST01H (Workstation)	HS7000ELN01H (Lan board) HS7000ELN02H (Lan board)	HS7000EMS11H (1M-SRAM) HS7000EMS12H (4M-SRAM)	HS7000EXR10H	HS7000EBR01H	HS2655EPD70H	HS2655ECN71H (TFP-120) HS2655ECH71H (FP-128)	HS2655ID7M1ST (HP9000/700) HS2655IDCM1SM (SPARC/Solaris)

E7000 PC	**Host System Interface Board**	**Expansion Memory**	**Bus Monitor**		**Emulator Pod**	**User Cable**	**GUI**
			Interface Board	**Monitor Board**			
HS7000ESTP1H (IBM PC)	HS7000EII01H (IBM PC interface board)	HS7000EMS11H (1M-SRAM) HS7000EMS12H (4M-SRAM)	HS7000EXR10H	HS7000EBR01H	HS2655EPD70H	HS2655ECN71H (TFP-120) HS2655ECH71H (FP-128)	HS2655G7IW1SF (IBM PC)

E7000S	**Host System Interface Board**	**Expansion Memory**	**Bus Monitor**		**Emulator Pod**	**User Cable**	**Integrated Development Manager**
			Interface Board	**Monitor Board**			
HS700SEST01H* (Workstation)	HS7000ELN01H (Lan board) HS7000ELN02H (Lan board)	HS7000EMS11H (1M-SRAM) HS7000EMS12H (4M-SRAM)	HS7000EXR10H	HS7000EBR01H	HS2655EPD70H	HS2655ECN71H (TFP-120) HS2655ECH71H (FP-128)	HS2655ID7M3ST* (HP9000/700) HS2655IDCM3SM* (SPARC/Solaris)

E7000S PC	**Host System Interface Board**	**Expansion Memory**	**Bus Monitor**		**Emulator Pod**	**User Cable**	**GUI**
			Interface Board	**Monitor Board**			
HS700SESTP1H* (IBM PC)	HS7000EII01H (IBM PC interface board)	HS7000EMS11H (1M-SRAM) HS7000EMS12H (4M-SRAM)	HS7000EXR10H	HS7000EBR01H	HS2655EPD70H	HS2655ECN71H (TFP-120) HS2655ECH71H (FP-128)	HS2655G7IW2SF* (IBM PC)

Note: * Under development

5.4 Socket Adapters

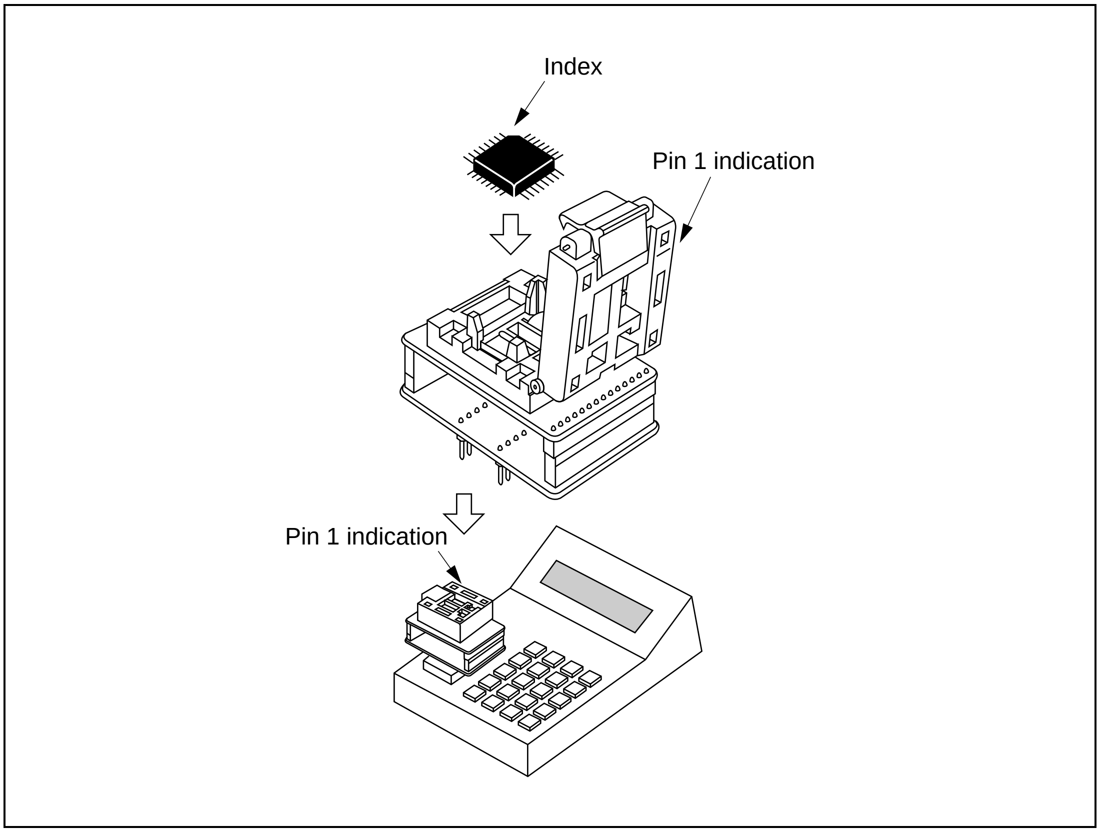
A socket adapter is a pin conversion adapter for writing or verifying a program in the on-chip PROM of a microcomputer using a general-purpose PROM writer. A different adapter is available for each product and package type, so be sure to choose the correct adapter for the product concerned.

Applicable EPROM Writer Table

Product	Package	Socket Adapter
H8S/2355	TFP-120	HS2655ESNS1H
	FP-128	HS2655ESHS1H

Method of Use (In Case of HS2655ESNS1H)

Refer to the socket adapter instruction manual for details.



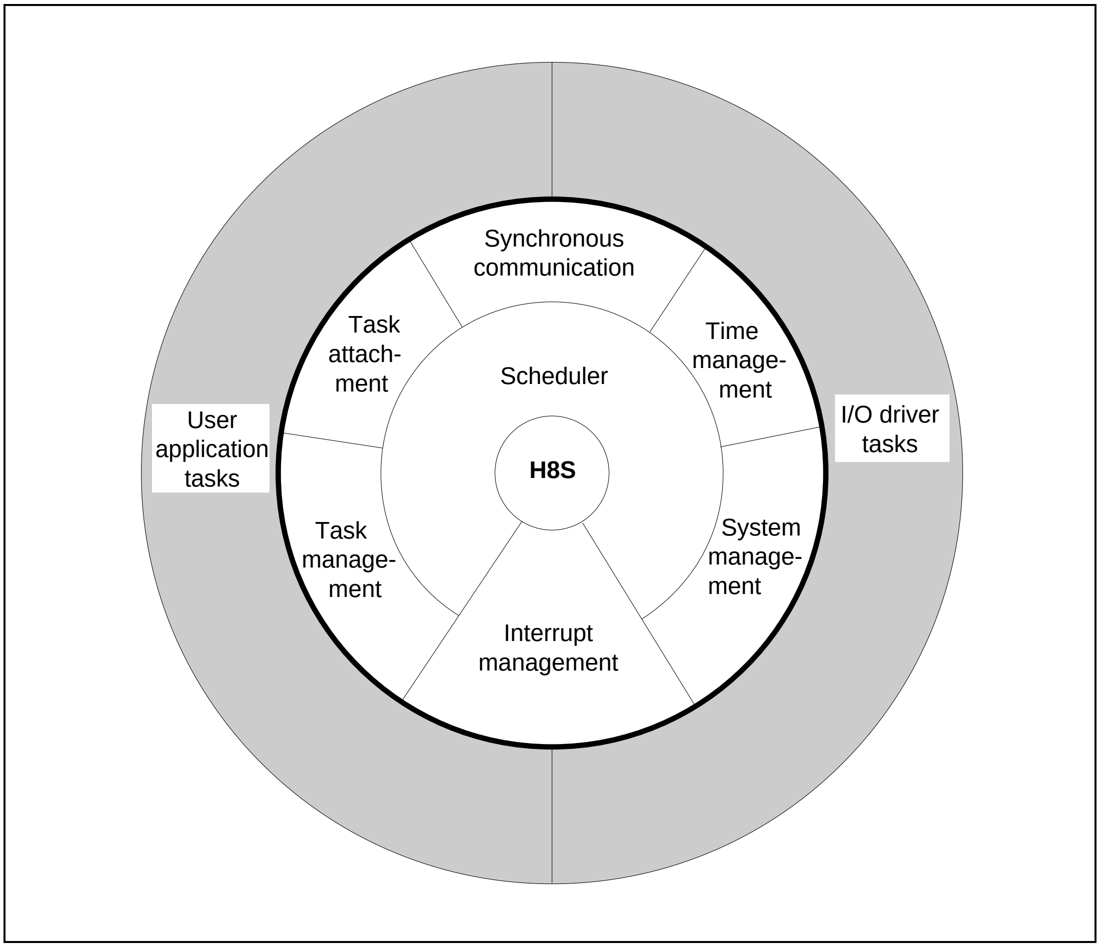
5.5 HI Series OS

The HI Series OS is available for the H8S/2355 Series, enabling the program structure to be made explicit and a custom-made operating system to be constructed.

Features

- Emphasis on realtime characteristics
  - Microsecond order response to interrupts from off-chip
- Compact OS
- Improved development efficiency thought modular software approach
- ROM implementation capability
  - Design presupposes product embedding
- Module 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
  - System call C interface library provided
  - Shorter development time, lower development costs, improved serviceability and reliability
- Multitask debugger supported on E7000 emulator
  - A multitask debugger that operates on the E7000 emulator is available, for efficient application debugging

HI8-2600 System Configuration Diagram

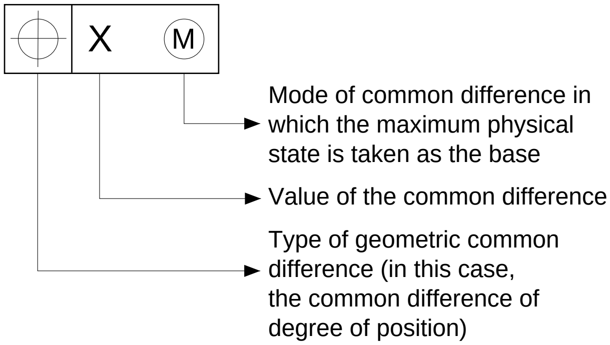


Appendix

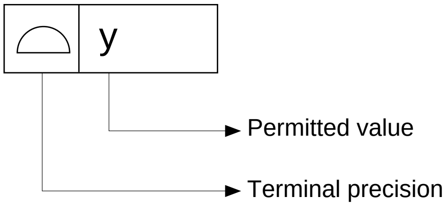
Package

Package Outline Dimensions (Unit: mm)

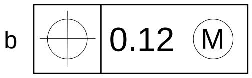
Indication of Geometric Common Difference



Indication of Terminal Precision “y”



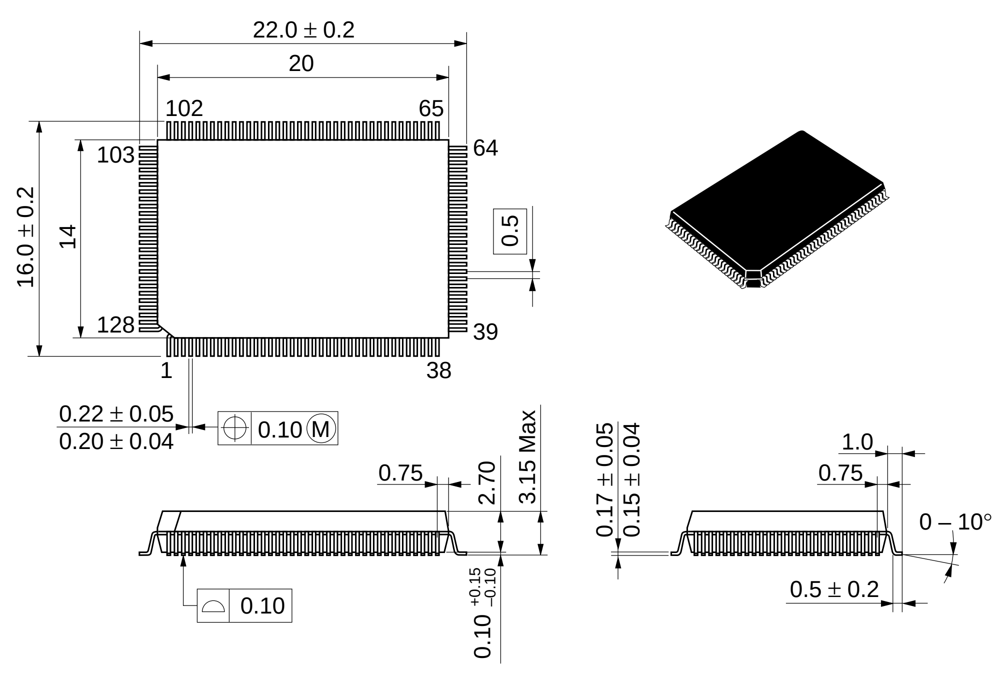
Example



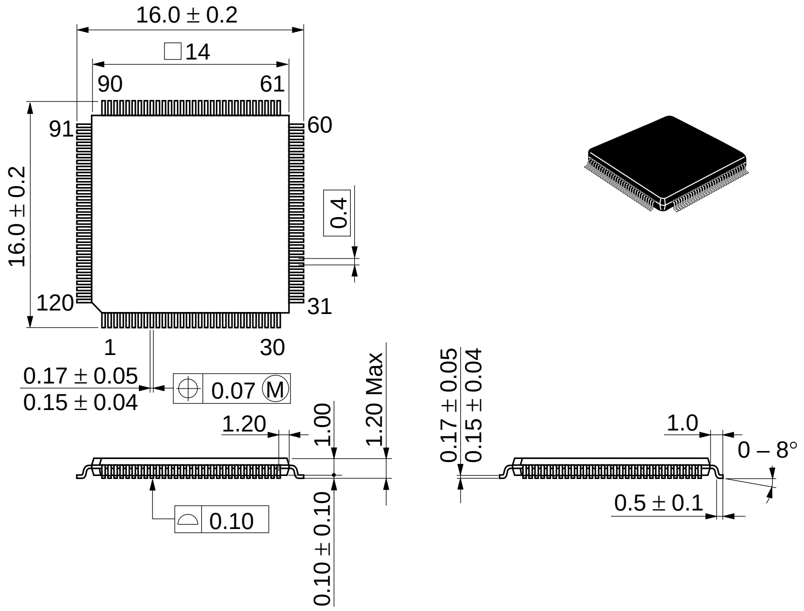
When the terminal width b is the maximum dimension, it indicates that a divergence from the true position of the center position of up to 0.12 mm is permitted.

If b is smaller than the maximum dimension, the common difference corresponding to b can be extended.

FP-128


22.0 ± 0.2
20
102
65
64
103
16.0 ± 0.2
14
0.5
128
39
1
38
0.22 ± 0.05
0.20 ± 0.04
0.10 M
0.75
2.70
3.15 Max
0.10 +0.15 −0.10
0.10
0.17 ± 0.05
0.15 ± 0.04
1.0
0.75
0 – 10°
0.5 ± 0.2

TFP-120


16.0 ± 0.2
□14
90
61
91
60
16.0 ± 0.2
0.4
120
31
1
30
0.17 ± 0.05
0.15 ± 0.04
0.07 Ⓜ
1.20
1.00
1.20 Max
0.10
0.10 ± 0.10
0.17 ± 0.05
0.15 ± 0.04
1.0
0 – 8°
0.5 ± 0.1