

H8/538F

Overview

Preface

Hitachi's H8/500 single-chip microcontrollers are a high-performance family built around a fast H8/500 CPU with a 16-bit internal architecture. This document describes the H8/538F (F-ZTATTM*3 microcomputer), which has flash memory as its on-chip ROM.

The H8/538 is already available in a ZTATTM (zero turn-around time) version*1 with one-time-programmable PROM. Flash memory is electrically erasable and programmable, however, so memory contents can be modified repeatedly, even after the chip is mounted on-board. Flash memory also offers large capacity, because it has a one-transistor configuration instead of the two-transistor configuration of EEPROM.

On-chip flash memory enables an H8/538F to be reprogrammed or given new internal data while embedded in the device it is controlling. Possibilities include small, flexible, quick-turn production runs, optimization on an individual unit basis, and firmware updates and maintenance in the field.

The H8/538F supports two on-chip programming modes: boot program mode and user program mode. It also supports a write mode that enables it to be programmed with a standard EPROM programmer (set to HN28F101 specifications).

Hitachi is working to provide a full, efficient microcontroller application system development environment. The environment includes support software, a workstation-compatible E7000 realtime emulator, and an operating system conforming to the ITRON*2-specification for embedded control.

Notes: 1. ZTAT (Zero Turn-Around Time) is a trademark of Hitachi, Ltd.

2. ITRON is an abbreviation of Industrial TRON.

TRON is an abbreviation of The Realtime Operating-system Nucleus.

3. F-ZTAT (Flexible-ZTAT) is a trademark of Hitachi, Ltd.

Contents

Section 1	Features of the H8/538F	7
Section 2	Pin Arrangement and Functions	10
2.1	Pin Arrangement.....	10
2.2	Pin Arrangement in Each Mode Operating	12
2.3	Pin Functions	14
Section 3	Block Diagram	16
Section 4	CPU	17
4.1	Memory Management.....	17
4.2	Registers	18
4.2.1	General Registers	18
4.2.2	Control Registers.....	18
4.3	Data Structure	21
4.4	Addressing Modes	22
4.5	Instruction Set.....	23
4.5.1	Instruction Formats.....	23
4.5.2	H8/500 Family Instruction Set	25
4.6	Basic Operational Timing.....	32
4.6.1	Basic Clock Timing	32
4.6.2	CPU Read/Write Cycle.....	32
4.7	Operating States.....	37
4.7.1	Program Execution State	37
4.7.2	Exception-Handling State.....	37
4.7.3	Bus-Released State	37
4.7.4	Power-Down State.....	38
4.8	Exception Handling	39
4.9	Interrupts.....	40
4.10	Interrupt Controller.....	40
4.11	Bus Release.....	41
Section 5	Operating Modes	42
5.1	Expanded Minimum Modes (Modes 1, 2, and 6)	42
5.2	Expanded Maximum Modes (Modes 3, 4, and 5)	42
5.3	Single-Chip Mode (Mode 7).....	42

Section 6 On-Chip Supporting Modules 44

6.1 Data Transfer Controller..... 44

6.1.1 Functions 44

6.1.2 Features 44

6.1.3 Block Diagram 45

6.1.4 Data Transfer Operation 45

6.1.5 Number of States Required for DTC Transfer 48

6.1.6 Format of Register Information in Memory..... 48

6.1.7 Example..... 49

6.2 Wait-State Controller 51

6.2.1 Functions 51

6.2.2 Features 51

6.2.3 Block Diagram 51

6.2.4 Programmable Wait Mode..... 52

6.2.5 Pin Wait Mode..... 52

6.2.6 Pin Auto-Wait Mode 52

6.3 Sixteen-Bit Integrated-Timer Pulse Unit 53

6.3.1 Functions 53

6.3.2 Features 53

6.3.3 Block Diagrams..... 54

6.3.4 Interrupt Sources and Data Transfer Controller Activation 58

6.3.5 Examples of Pulse Output 59

6.3.6 Examples of Pulse Measurement Functions..... 61

6.4 Watchdog Timer 64

6.4.1 Functions 64

6.4.2 Block Diagram 64

6.4.3 Watchdog Timer Operation 65

6.4.4 Interval Timer Operation..... 65

6.4.5 Overflow Period 66

6.5 Serial Communication Interface 67

6.5.1 Functions 67

6.5.2 Features 67

6.5.3 Block Diagram 68

6.5.4 Asynchronous Mode..... 69

6.5.5 Multiprocessor Communication 70

6.5.6 Synchronous Mode..... 71

6.5.7 Interrupts and Data Transfer Controller Activation 71

6.5.8 BRR Settings for Typical Bit Rates (Asynchronous Mode)..... 72

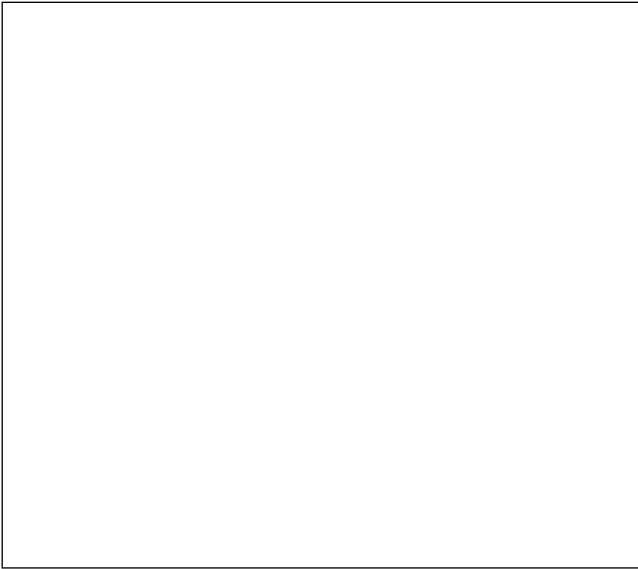
6.5.9 BRR Settings for Typical Bit Rates (Synchronous Mode)..... 73

6.6 A/D Converter 74

6.6.1 Functions 74

- 6.6.2 Features 74
 - 6.6.3 Block Diagram 75
 - 6.6.4 Operation 76
- 6.7 I/O Ports..... 77
- 6.8 RAM 79
 - 6.8.1 Functions 79
 - 6.8.2 Block Diagram 79
- 6.9 Flash Memory 80
 - 6.9.1 Features 80
 - 6.9.2 Block Diagram 81
 - 6.9.3 Register Configuration 82
 - 6.9.4 Emulation of Flash Memory by RAM 85
 - 6.9.5 Programming with an EPROM Programmer 87
- Section 7 Support Tools..... 88
 - 7.1 Software..... 88
 - 7.2 Hardware 91

Section 1 Features of the H8/538F



- High-speed H8/500 CPU
 - General-register machine
 - Eight 16-bit general registers
 - Five 8-bit and two 16-bit control registers
 - High speed for realtime control
 - Maximum clock rate: 16 MHz
 - 16-bit register add/subtract time: 125 ns (at 16 MHz)
 - Optimized, orthogonal instruction set
 - Addressing modes and data size can be specified independently for each instruction
 - Register-register and register-memory operations
 - C-language oriented (with short formats for frequently-used instructions and addressing modes)
- 60-kbyte flash memory
 - 100 program-erase cycles
 - Program/erase voltage (V_{PP}): 12 V \pm 0.6 V (external power supply)
 - Program time: 20 μ s/byte (typical)
 - Erase time: 1 s (typical)
 - Erase extent
 - Chip erase
 - Block erase: 7 large (8-kbyte) blocks; 8 small (128-byte to 1-kbyte) blocks
 - Program/erase methods
 - CPU (software) control
 - HN28F101 flash memory compatible

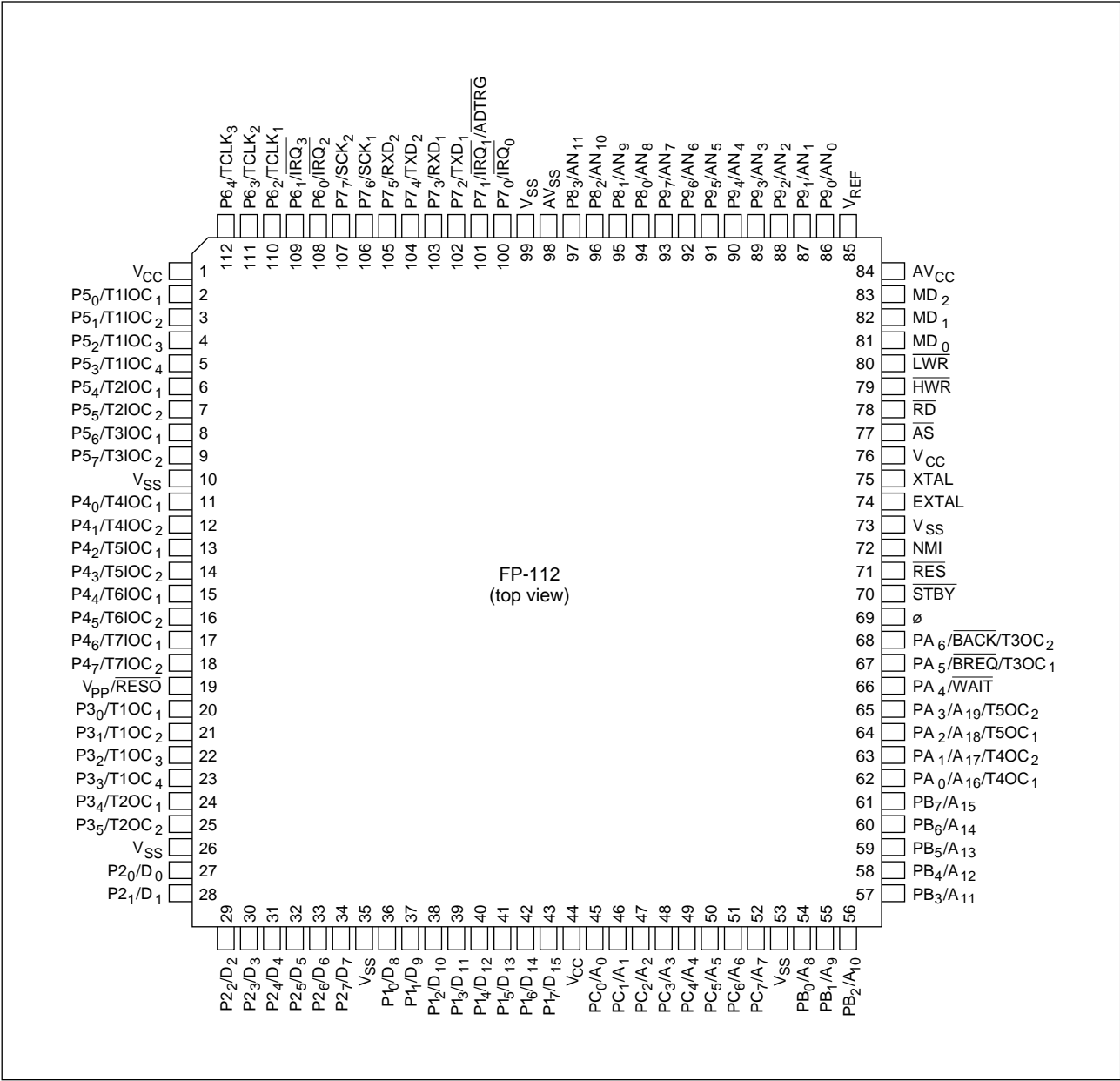
- 2-kbyte high-speed RAM
- 16-bit integrated-timer pulse unit (IPU)
 - Can generate 28 waveform outputs
 - Can measure input pulse width and period
 - Can be programmed as a pulse-width modulator with 0 to 100% duty cycle (maximum resolution: 100 ns)
- Watchdog timer (1 channel)
 - Can be used as a watchdog timer or interval timer
- Serial communication interface (2 channels)
 - Asynchronous or synchronous mode (selectable)
 - Multiprocessor communication function (asynchronous mode)
- 10-bit A/D converter (12 channels)
 - Single mode or scan mode (selectable)
 - Sample & hold function
 - Startable by external trigger signal
- Twelve I/O ports
 - 74 input/output pins
 - 12 input-only pins
- Interrupt controller
 - Five external interrupt pins ($\overline{\text{NMI}}$, $\overline{\text{IRQ}}_0$, $\overline{\text{IRQ}}_1$, $\overline{\text{IRQ}}_2$, $\overline{\text{IRQ}}_3$)
 - Thirty-nine internal interrupt sources
 - Eight priority levels
- Data transfer controller
 - Transfers data between I/O and memory independently of the CPU
- Wait-state controller
 - Can be controlled by external pins or by register values
- Bus controller
 - External bus width and bus cycle length can be selected independently
- Seven operating modes
 - Expanded minimum mode (high-speed 16-bit bus)
Maximum 64-kbyte address space (modes 1 and 2)
 - Expanded maximum mode (high-speed 16-bit bus)
Maximum 1-Mbyte address space (modes 3 and 4)

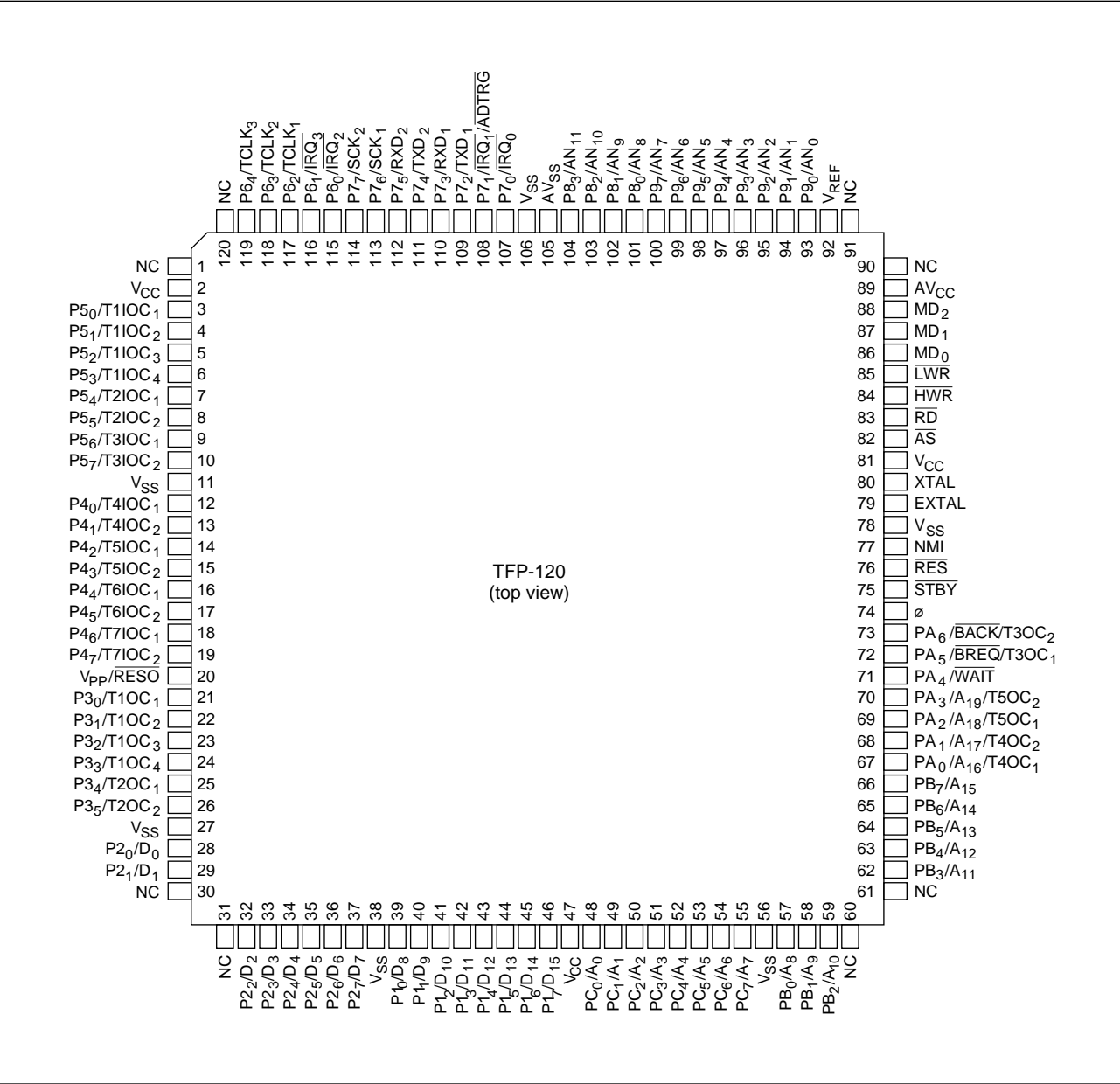
- Expanded minimum mode (low-speed 8-bit bus)
 - Maximum 64-kbyte address space (mode 6)
- Expanded maximum mode (low-speed 8-bit bus)
 - Maximum 1-Mbyte address space (mode 5)
- Single-chip mode (mode 7)
- Two on-board programming modes
 - User program mode (for modifying programs and data)
 - On-board programming by user software (mode 2, 4, or 7, $V_{PP} = 12\text{ V}$)
 - Boot mode (for initial loading of programs and data)
 - On-board programming by built-in boot program (mode 2, 4, or 7, $V_{PP} = 12\text{ V}$, $MD_2 = 12\text{ V}$)
- Clock generator on-chip
- Supply voltage
 - High-speed version: $5\text{ V} \pm 10\%$ (at 16 MHz)
 - Low-voltage version: 3.0 V to 5.5 V (at 10 MHz); 2.7 V to 5.5 V (at 8 MHz)
- Package
 - 112-Pin plastic QFP (FP-112)
 - 120-Pin TQFP (TFP-120)

Section 2 Pin Arrangement and Functions

2.1 Pin Arrangement

- 112-Pin Plastic QFP





2.2 Pin Arrangement in Each Mode Operating

Pin Name							Pin Name						
Pin No.		Expanded Min. Modes		Expanded Max. Modes		Single-Chip Mode	Pin No.		Expanded Min. Modes		Expanded Max. Modes		Single-Chip Mode
QFP- TQFP- Modes		1 & 6		Mode 2		Mode 7	QFP- TQFP- Modes		1 & 6		Mode 2		Mode 7
112	120	1	6	3 & 5	Mode 4	Mode 7	112	120	1	6	3 & 5	Mode 4	Mode 7
—	1	NC					42	45	D ₁₄				
1	2	V _{CC}					43	46	D ₁₅				
2	3	P5 ₀ /T1IOC ₁					44	47	V _{CC}				
3	4	P5 ₁ /T1IOC ₂					45	48	A ₀	PC ₀ /A ₀	A ₀	PC ₀ /A ₀	PC ₀
4	5	P5 ₂ /T1IOC ₃					46	49	A ₁	PC ₁ /A ₁	A ₁	PC ₁ /A ₁	PC ₁
5	6	P5 ₃ /T1IOC ₄					47	50	A ₂	PC ₂ /A ₂	A ₂	PC ₂ /A ₂	PC ₂
6	7	P5 ₄ /T2IOC ₁					48	51	A ₃	PC ₃ /A ₃	A ₃	PC ₃ /A ₃	PC ₃
7	8	P5 ₅ /T2IOC ₂					49	52	A ₄	PC ₄ /A ₄	A ₄	PC ₄ /A ₄	PC ₄
8	9	P5 ₆ /T3IOC ₁					50	53	A ₅	PC ₅ /A ₅	A ₅	PC ₅ /A ₅	PC ₅
9	10	P5 ₇ /T3IOC ₂					51	54	A ₆	PC ₆ /A ₆	A ₆	PC ₆ /A ₆	PC ₆
10	11	V _{SS}					52	55	A ₇	PC ₇ /A ₇	A ₇	PC ₇ /A ₇	PC ₇
11	12	P4 ₀ /T4IOC ₁					53	56	V _{SS}				
12	13	P4 ₁ /T4IOC ₂					54	57	A ₈	PB ₀ /A ₈	A ₈	PB ₀ /A ₈	PB ₀
13	14	P4 ₂ /T5IOC ₁					55	58	A ₉	PB ₁ /A ₉	A ₉	PB ₁ /A ₉	PB ₁
14	15	P4 ₃ /T5IOC ₂					56	59	A ₁₀	PB ₂ /A ₁₀	A ₁₀	PB ₂ /A ₁₀	PB ₂
15	16	P4 ₄ /T6IOC ₁					—	60	NC				
16	17	P4 ₅ /T6IOC ₂					—	61	NC				
17	18	P4 ₆ /T7IOC ₁					57	62	A ₁₁	PB ₃ /A ₁₁	A ₁₁	PB ₃ /A ₁₁	PB ₃
18	19	P4 ₇ /T7IOC ₂					58	63	A ₁₂	PB ₄ /A ₁₂	A ₁₂	PB ₄ /A ₁₂	PB ₄
19	20	V _{PP} /RESO					59	64	A ₁₃	PB ₅ /A ₁₃	A ₁₃	PB ₅ /A ₁₃	PB ₅
20	21	P3 ₀ /T1OC ₁					60	65	A ₁₄	PB ₆ /A ₁₄	A ₁₄	PB ₆ /A ₁₄	PB ₆
21	22	P3 ₁ /T1OC ₂					61	66	A ₁₅	PB ₇ /A ₁₅	A ₁₅	PB ₇ /A ₁₅	PB ₇
22	23	P3 ₂ /T1OC ₃					62	67	PA ₀ /T4OC ₁		A ₁₆	PA ₀ /A ₁₆ /T4OC ₁	PA ₀ /T4OC ₁
23	24	P3 ₃ /T1OC ₄					63	68	PA ₁ /T4OC ₂		A ₁₇	PA ₁ /A ₁₇ /T4OC ₂	PA ₁ /T4OC ₂
24	25	P3 ₄ /T2OC ₁					64	69	PA ₂ /T5OC ₁		A ₁₈	PA ₂ /A ₁₈ /T5OC ₁	PA ₂ /T5OC ₁
25	26	P3 ₅ /T2OC ₂					65	70	PA ₃ /T5OC ₂		A ₁₉	PA ₃ /A ₁₉ /T5OC ₂	PA ₃ /T5OC ₂
26	27	V _{SS}					66	71	PA ₄ /WAIT				PA ₄
27	28	D ₀	P2 ₀	D ₀		P2 ₀	67	72	PA ₅ /BREQ/T3OC ₁				PA ₅ /T3OC ₁
28	29	D ₁	P2 ₁	D ₁		P2 ₁	68	73	PA ₆ /BACK/T3OC ₂				PA ₆ /T3OC ₂
—	30	NC					69	74	ø				
—	31	NC					70	75	STBY				
29	32	D ₂	P2 ₂	D ₂		P2 ₂	71	76	RES				
30	33	D ₃	P2 ₃	D ₃		P2 ₃	72	77	NMI				
31	34	D ₄	P2 ₄	D ₄		P2 ₄	73	78	V _{SS}				
32	35	D ₅	P2 ₅	D ₅		P2 ₅	74	79	EXTAL				
33	36	D ₆	P2 ₆	D ₆		P2 ₆	75	80	XTAL				
34	37	D ₇	P2 ₇	D ₇		P2 ₇	76	81	V _{CC}				
35	38	V _{SS}					77	82	AS				
36	39	D ₈					78	83	RD				
37	40	D ₉					79	84	HWR				
38	41	D ₁₀					80	85	LWR				
39	42	D ₁₁					81	86	MD ₀				
40	43	D ₁₂					82	87	MD ₁				
41	44	D ₁₃					83	88	MD ₂				

Continued on next page

Pin Name							Pin Name						
Pin No.		Expanded Min. Modes		Expanded Max. Modes		Single-Chip Mode	Pin No.		Expanded Min. Modes		Expanded Max. Modes		Single-Chip Mode
QFP- 112	TQFP- 120	Modes 1 & 6		Modes 3 & 5		Mode 7	QFP- 112	TQFP- 120	Modes 1 & 6		Modes 3 & 5		Mode 7
84	89	AV _{CC}					98	105	AV _{SS}				
—	90	NC					99	106	V _{SS}				
—	91	NC					100	107	P7 ₀ /IRQ ₀				
85	92	V _{REF}					101	108	P7 ₁ /IRQ ₁ /ADTRG				
86	93	P9 ₀ /AN ₀					102	109	P7 ₂ /TXD ₁				
87	94	P9 ₁ /AN ₁					103	110	P7 ₃ /RXD ₁				
88	95	P9 ₂ /AN ₂					104	111	P7 ₄ /TXD ₂				
89	96	P9 ₃ /AN ₃					105	112	P7 ₅ /RXD ₂				
90	97	P9 ₄ /AN ₄					106	113	P7 ₆ /SCK ₁				
91	98	P9 ₅ /AN ₅					107	114	P7 ₇ /SCK ₂				
92	99	P9 ₆ /AN ₆					108	115	P6 ₀ /IRQ ₂				
93	100	P9 ₇ /AN ₇					109	116	P6 ₁ /IRQ ₃				
94	101	P8 ₀ /AN ₈					110	117	P6 ₂ /TCLK ₁				
95	102	P8 ₁ /AN ₉					111	118	P6 ₃ /TCLK ₂				
96	103	P8 ₂ /AN ₁₀					112	119	P6 ₄ /TCLK ₃				
97	104	P8 ₃ /AN ₁₁					—	120	NC				

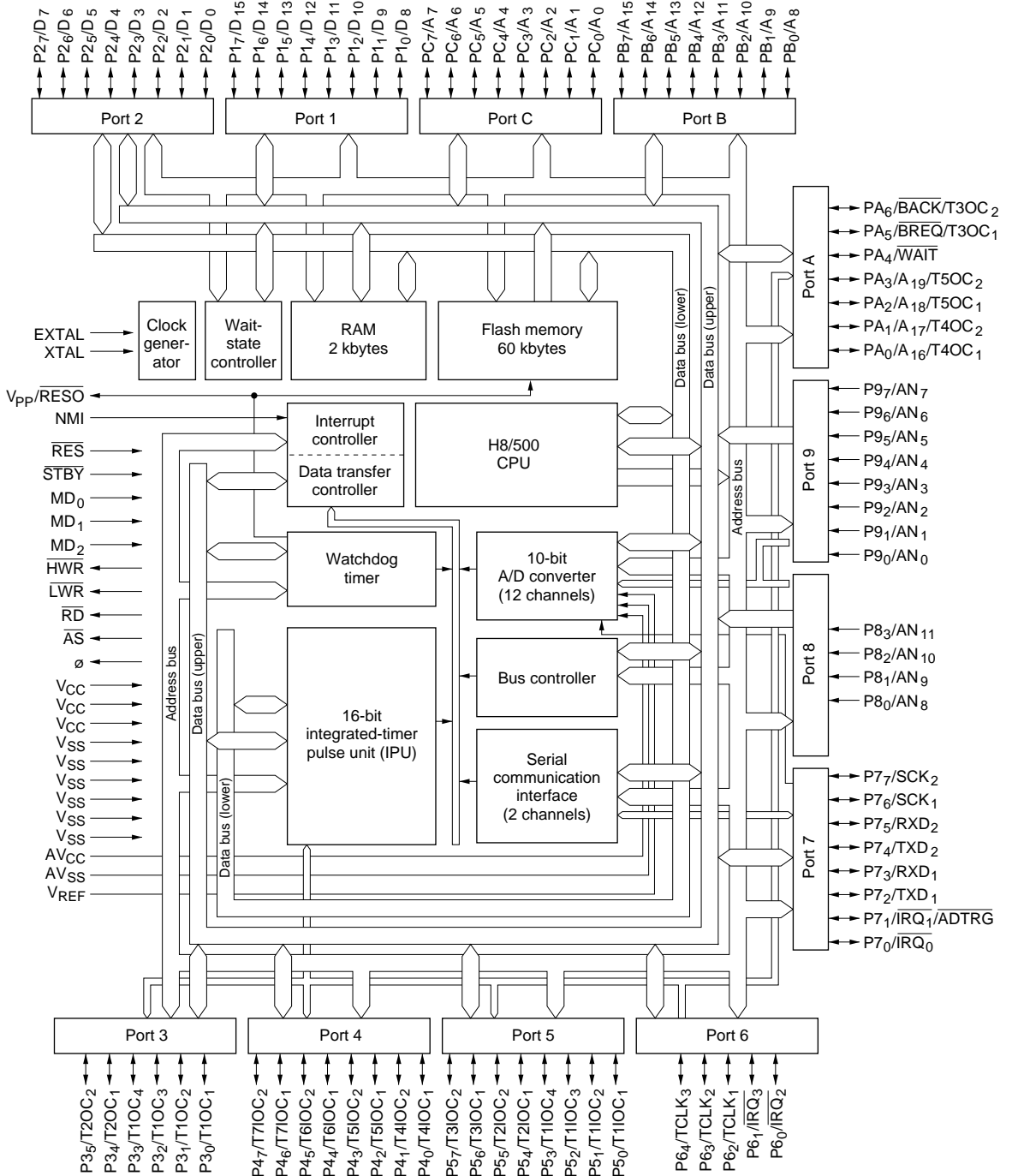
2.3 Pin Functions

Type	Signal	I/O	Function
Power	V _{CC}	I	Power supply (2.7 V to 5.5 V)
	V _{SS}	I	Ground
V _{PP} /reset output	V _{PP} / $\overline{\text{RESO}}$	I/O	Input: V _{PP} (12 V) for flash memory program/erase Output: reset signal
Clock	XTAL	I	Crystal
	EXTAL	I	Crystal or external clock
	∅	O	System clock
System control	$\overline{\text{RES}}$	I	Reset
	$\overline{\text{STBY}}$	I	Standby
	$\overline{\text{BREQ}}$	I	Bus request
	$\overline{\text{BACK}}$	O	Bus request acknowledge
Address bus	A ₀ to A ₁₉	O	Address bus
Data bus	D ₀ to D ₁₅	I/O	Data bus
Bus control	$\overline{\text{AS}}$	O	Address strobe
	$\overline{\text{LWR}}$	O	Low write
	$\overline{\text{HWR}}$	O	High write
	$\overline{\text{RD}}$	O	Read
	$\overline{\text{WAIT}}$	I	Wait
Interrupt signals	NMI	I	Nonmaskable interrupt
	$\overline{\text{IRQ}}_0$	I	Interrupt request 0 to 3
	$\overline{\text{IRQ}}_1$		
	$\overline{\text{IRQ}}_2$		
	$\overline{\text{IRQ}}_3$		
Operating mode control	MD ₀ to MD ₂	I	Mode select
16-bit integrated-timer pulse unit	T1IOC ₁ to T7IOC ₂	I/O	16-bit timer input capture/output compare
	TCLK ₁ to TCLK ₃	I	16-bit timer clock input
	T1OC ₁ to T5OC ₂	O	16-bit timer output compare
Serial communication interface	TXD ₁ , TXD ₂	O	Transmit data
	RXD ₁ , RXD ₂	I	Receive data
	SCK ₁ , SCK ₂	I/O	Serial clock input/output

Continued on next page

Type	Signal	I/O	Function
I/O ports	P1 ₀ to P1 ₇	I/O	Port 1
	P2 ₀ to P2 ₇	I/O	Port 2
	P3 ₀ to P3 ₅	I/O	Port 3
	P4 ₀ to P4 ₇	I/O	Port 4
	P5 ₀ to P5 ₇	I/O	Port 5
	P6 ₀ to P6 ₄	I/O	Port 6
	P7 ₀ to P7 ₇	I/O	Port 7
	P8 ₀ to P8 ₃	I	Port 8
	P9 ₀ to P9 ₇	I	Port 9
	PA ₀ to PA ₆	I/O	Port A
	PB ₀ to PB ₇	I/O	Port B
	PC ₀ to PC ₇	I/O	Port C
A/D converter	AN ₀ to AN ₁₁	I	Analog input
	AV _{CC}	I	Analog power supply
	AV _{SS}	I	Analog ground
	V _{REF}	I	Reference voltage
	ADTRG	I	Analog trigger

Section 3 Block Diagram



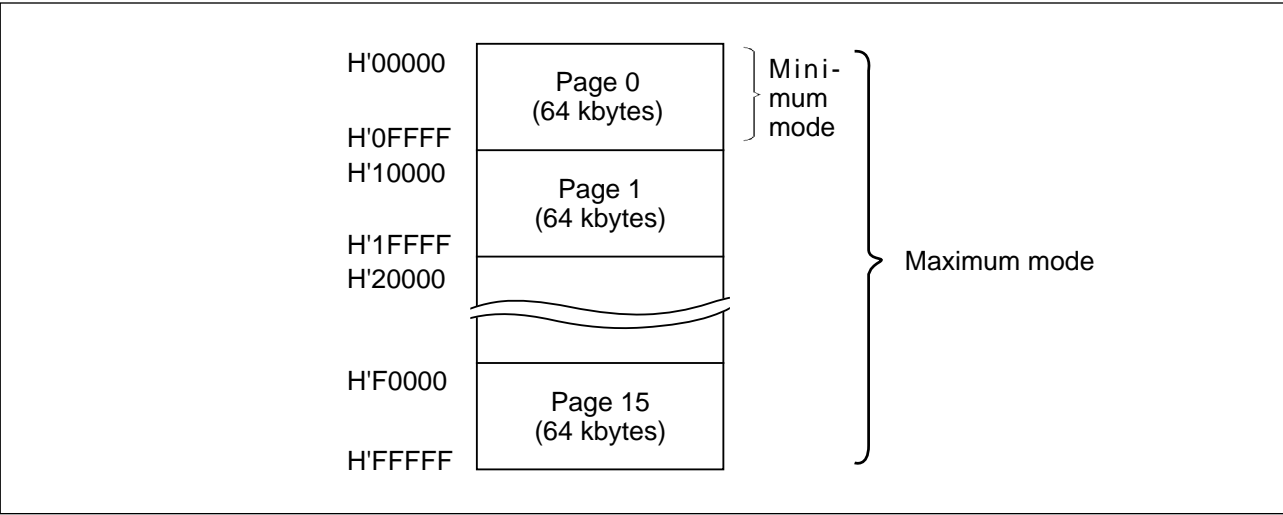
Section 4 CPU

The H8/500 Family CPU features eight general-purpose registers, 16-bit internal data width, and an optimized instruction set, providing enhanced data-processing capabilities plus the high speed demanded for realtime control.

4.1 Memory Management

The CPU views its address space as consisting of 64-kbyte pages. Maximum address space size for the H8/538F is 64 kbytes (one page) in minimum mode and 1 Mbyte* (16 pages) in maximum mode. The CPU is switched between these two operating modes by inputs at the mode pins (MD₀ to MD₂) during reset.

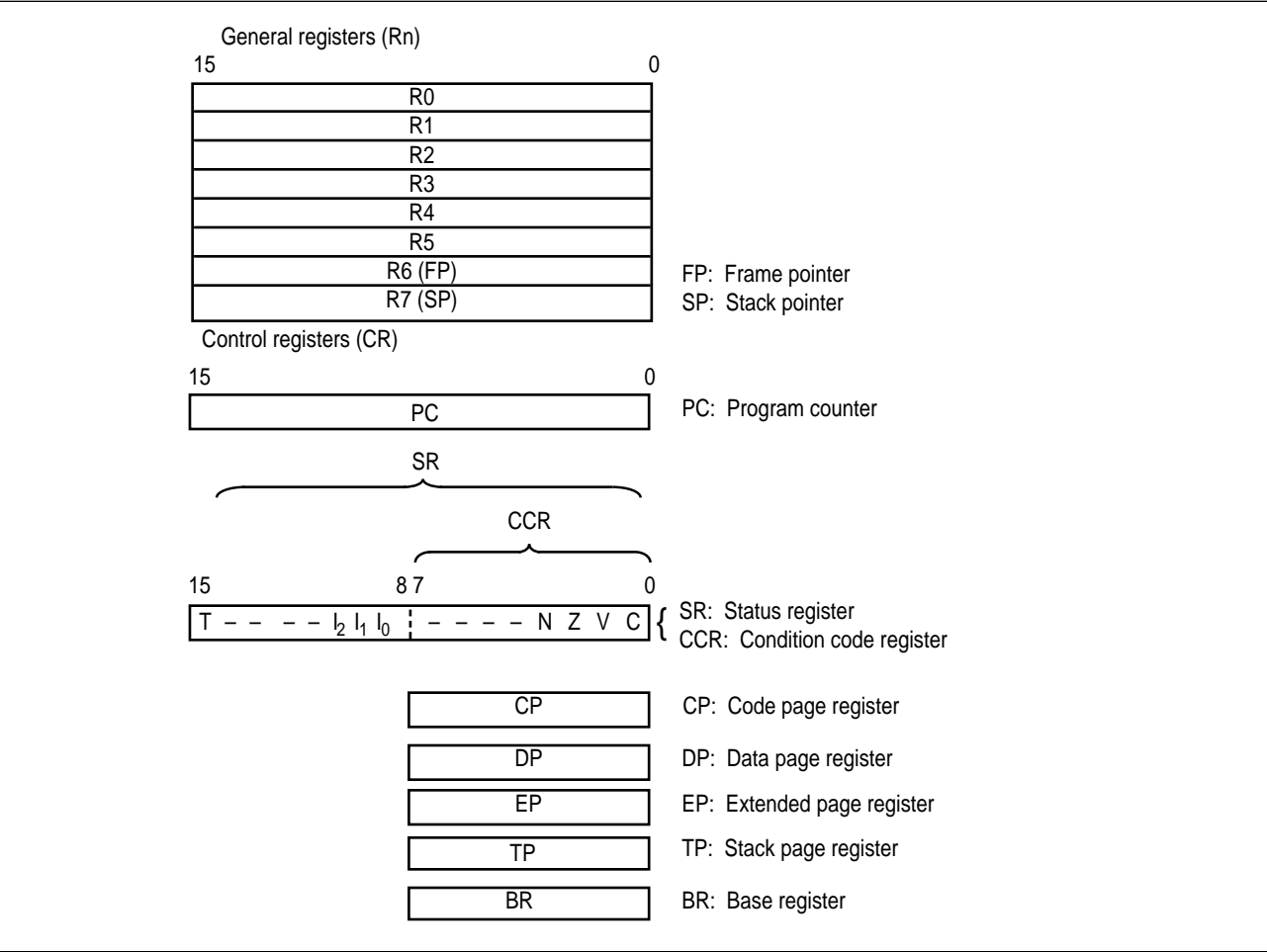
Note: * Other chips in the H8/500 Family support address spaces up to 16 Mbytes.



Memory Map

4.2 Registers

The CPU has eight 16-bit general registers (R0 to R7), two 16-bit control registers (PC and SR/CCR), and five 8-bit control registers.



CPU Registers

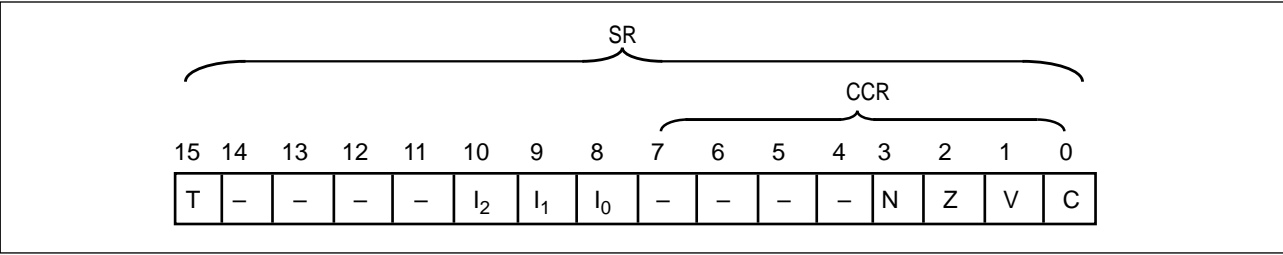
4.2.1 General Registers

All eight of the 16-bit general registers are functionally alike, offering powerful addressing mode support with no distinction between data registers and address registers. Two of these registers, in addition to functioning as general registers, have special assignments. R7 is the stack pointer, used implicitly in exception handling and subroutine calls. R6 functions as a frame pointer, used implicitly in the LINK and UNLK instructions to reserve or release a stack frame.

4.2.2 Control Registers

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

Status Register (SR): This 16-bit register indicates the internal state of the CPU. The lower half of the status register is referred to as the condition code register (CCR): its 8 bits can be accessed as a 1-byte condition code.



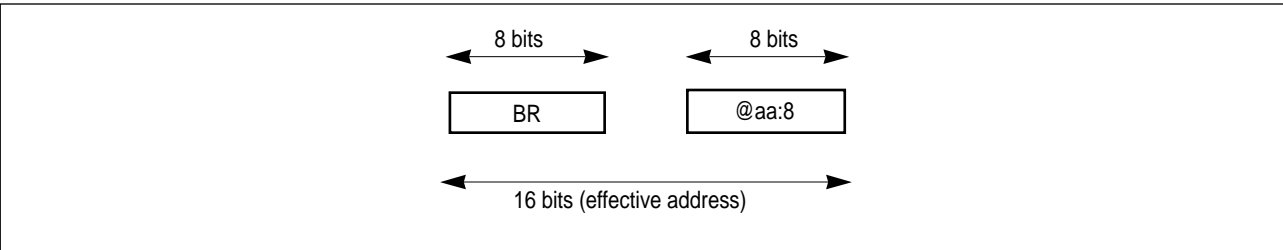
Bit 15 (T) – Trace mode: 0 – Normal mode (Instructions are executed in sequence)
1 – Trace mode

Bits 10 to 8 (I₂ to I₀)–Interrupt mask: Indicates the priority of the program currently executing, from 7 (high) to 0 (low). Interrupt requests of the indicated priority level or lower are not accepted.

Priority	Level	I ₂ I ₁ I ₀	Interrupts Accepted
↑ High ↓ Low	7	1 1 1	NMI
	6	1 1 0	Level 7 and NMI
	5	1 0 1	Levels 6 to 7 and NMI
	4	1 0 0	Levels 5 to 7 and NMI
	3	0 1 1	Levels 4 to 7 and NMI
	2	0 1 0	Levels 3 to 7 and NMI
	1	0 0 1	Levels 2 to 7 and NMI
	0	0 0 0	Levels 1 to 7 and NMI

- Bit 3 (N) – Negative
- Bit 2 (Z) – Zero
- Bit 1 (V) – Overflow
- Bit 0 (C) – Carry

Base Register (BR): This register stores the base address (the upper eight bits of the address within the zero page) used in the short absolute addressing mode. The base address can be located on any 256-byte boundary. The short absolute addressing mode specifies an offset with respect to the base address.



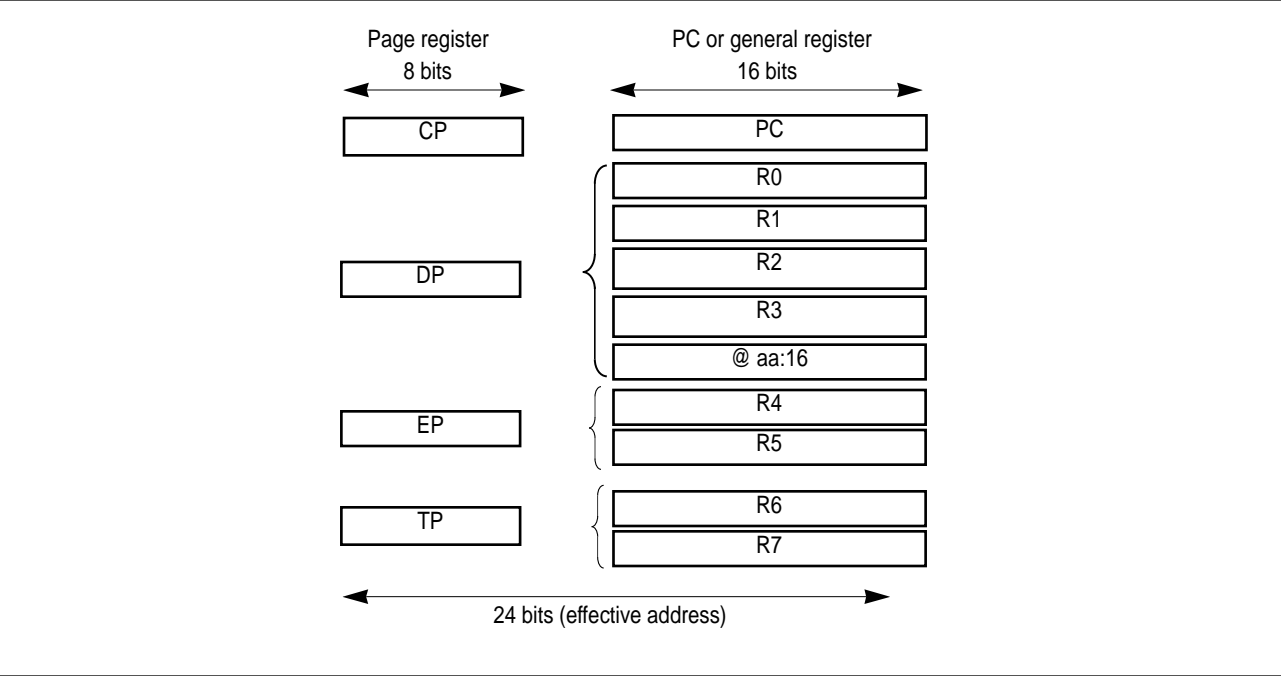
Page Registers: The four 8-bit page registers are used only in the maximum mode; their contents are ignored in the minimum mode. The page registers combine with the program counter and general registers to generate 24-bit addresses, thereby expanding the program area, data area, and stack area.

Code Page Register (CP): The code page register and the program counter combine to generate a 24-bit program code address. In the maximum mode, both the code page register and program counter are saved and restored in exception handling.

Data Page Register (DP): The data page register combines with general registers R0 to R3 to generate a 24-bit effective address. It is used to calculate effective addresses in the register indirect addressing mode using R0 to R3, and in the 16-bit absolute addressing mode. The data page register is rewritten by the LDC instruction.

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

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



Use of Page Registers

4.3 Data Structure

The CPU can process 1-bit data, 4-bit BCD data, 8-bit (byte) data, 16-bit (word) data, and 32-bit (longword) data. Bit manipulation instructions operate on 1-bit data. Decimal arithmetic instructions operate on 4-bit BCD data. Multiply and divide instructions operate on longword data. Byte and word data are used by almost all instructions.

The data formats in general registers and memory are shown below.

General Register Data Formats

Data Type	Register No.	Data Structure
1-Bit	Rn	<div>150</div> <div><div>1514131211109876543210</div></div>
BCD	Rn	<div>7430</div> <div><div>Don't careUpper digitLower digit</div></div>
Byte	Rn	<div>70</div> <div><div>Don't careMSBLSB</div></div>
Word	Rn	<div>150</div> <div><div>MSBLSB</div></div>
Longword	Rn* Rn + 1*	<div>310</div> <div><div>MSBUpper word</div><div>Lower wordLSB</div><div>150</div></div>

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

Data Formats in Memory

Data Type	Data Format
1-Bit (In byte operand)	<div><div>Address n</div><div><div>7</div><div>6</div><div>5</div><div>4</div><div>3</div><div>2</div><div>1</div><div>0</div></div></div>
1-Bit (In word operand)	<div><div>Even address</div><div>Odd address</div><div><div><div>15</div><div>14</div><div>13</div><div>12</div><div>11</div><div>10</div><div>9</div><div>8</div></div><div><div>7</div><div>6</div><div>5</div><div>4</div><div>3</div><div>2</div><div>1</div><div>0</div></div></div></div>
Byte	<div><div>Address n</div><div>MSB</div><div>LSB</div></div>
Word	<div><div>Even address</div><div>Odd address</div><div><div>MSB</div><div>Upper 8 bits</div><div>Lower 8 bits</div><div>LSB</div></div></div>
Byte in stack*2	<div><div>Even address</div><div>Odd address</div><div><div>Undefined data</div><div>MSB</div><div>LSB</div></div></div>
Word in stack*2	<div><div>Even address</div><div>Odd address</div><div><div>MSB</div><div>Upper 8 bits</div><div>Lower 8 bits</div><div>LSB</div></div></div>

Notes: 1. In memory, the upper byte of word data must always be located on an even address, and the lower byte on an odd address.
2. When the stack is accessed in exception and command processing, the memory access is always performed a word at a time, regardless of the actual data size.

4.4 Addressing Modes

The CPU supports the seven addressing modes listed in the table below. All instructions having operands can use addressing modes 1 through 6. Mode 7 is provided for branching instructions.

No.	Addressing Mode	Mnemonic	Effective Address
1	Register direct	Rn	• Register Rn
2	Register indirect	@Rn	• Address indicated by register Rn
3	Register indirect with displacement	@(d:8, Rn) @(d:16, Rn)	• Address indicated by register Rn + displacement (8 or 16 bits)
4	Register indirect with pre-decrement Register indirect with post-increment	@-Rn @Rn+	• Address resulting from decrementation of register Rn • Address indicated by register Rn (Rn is incremented after instruction execution.)
5	Immediate	#xx:8 #xx:16	• Operand data (8 or 16 bits) is included in the instruction code
6	Absolute address	@aa:8 @aa:16	• Upper 8 bits – contents of BR. • Lower 8 bits – contained in instruction code • Value specified directly in instruction code
7	PC-relative	disp	• Address indicated by PC + displacement (8 or 16 bits)

4.5 Instruction Set

The H8/500 Family CPU supports powerful and efficient machine-language instructions. Some of the features of the instruction set are:

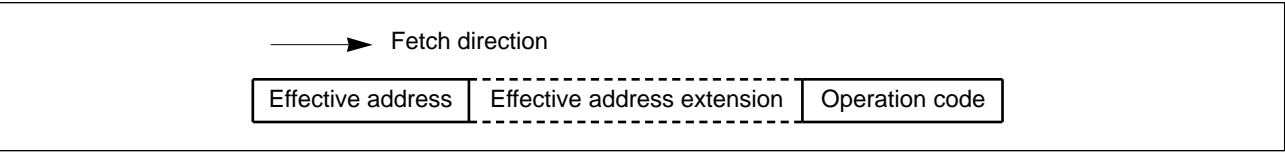
- Orthogonality: General instructions allow both register-register and register-memory operations, The data size and addressing mode can be specified independently for each operand.
- C-language support: Efficient object code can be generated from source programs coded in C.
- Short forms of frequently-used instructions

4.5.1 Instruction Formats

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

1. General format

This format is used in arithmetic instructions and other general instructions. To enable rapid execution, the effective address field is located at the beginning of the instruction, and the operation code at the end. Additional effective address information may be located in the middle of the instruction.



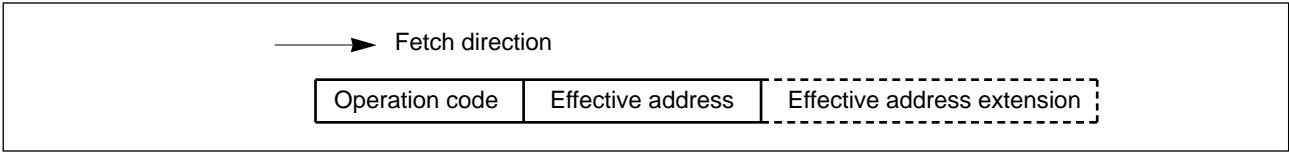
Effective Address Field: 1 byte containing information used to calculate an effective address.

Effective Address Extension: 0 to 2 bytes containing a displacement value, immediate data, or an absolute address. The number of bytes is specified in the effective address field.

Operation Code: Defines the operation to be carried out on the operand or operands located at the address calculated from the effective address information.

2. Special format

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



Operation Code: Defines the operation to be performed by the instruction.

Effective Address Field and Effective Address Extension: Zero to three bytes containing information used to calculate an effective address.

4.5.2 H8/500 Family Instruction Set

Mnemonic		Operation	Oper- and Size	Instruction Byte Length/No. of Execution States												CCR Flags			
				*1		R _n	@ R _n	@ (d:8, R _n)	@ (d:16, R _n)	@ -R _n	@ R _n +	@ aa:8	@ aa:16	#xx:8	#xx:16	N	Z	V	C
				i	j/k	1	1	2	3	1	1	2	3	2	3				
Data Transfer	MOV:G.B	(EAs) → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	0	—
	MOV:G.W	Rs → (EAd)	W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	0	—
	MOV:G.B	#IMM → (EAd)	B	1	2	3/3	3/7	4/7	5/8	3/7	3/8	4/7	5/8			↑	↑	0	—
	MOV:G.W		W	2	3	4/4	4/8	5/8	6/9	4/8	4/9	5/8	6/9			↑	↑	0	—
	MOV:E	#IMM → Rd	B	0	0									2/2		↑	↑	0	—
	MOV:I	(short format)	W	0	0										3/3	↑	↑	0	—
	MOV:L.B	(@aa:8) → Rd	B	1	0							2/5				↑	↑	0	—
	MOV:L.W	(short format)	W	2	0							2/5				↑	↑	0	—
	MOV:S.B	Rs → (@aa:8)	B	1	0							2/5				↑	↑	0	—
	MOV:S.W	(short format)	W	2	0							2/5				↑	↑	0	—
	MOV:F.B	@(d:8,FP) → Rd	B	1	0			2/5								↑	↑	0	—
	MOV:F.W	(short format)	W	2	0			2/5								↑	↑	0	—
	MOV:F.B	Rs → @(d:8,FP)	B	1	0			2/5								↑	↑	0	—
	MOV:F.W	(short format)	W	2	0			2/5								↑	↑	0	—
	LDM	@SP+ → R _n (register list)	W	2n*2	1						2/6 + 4n					—	—	—	—
	STM	R _n (register list) → @-SP	W	2n	1					2/6 + 3n						—	—	—	—
	XCH	Rs ↔ Rd	W	0	1	2/4										—	—	—	—
	SWAP	Rd (upper byte)↔Rd (lower byte)	B	0	1	2/3										↑	↑	0	—
	MOVTPE	Rs → (EAd) Synchronized with E clock	B	0	2		3/13 to 20	4/13 to 20	5/14 to 21	3/13 to 20	3/14 to 21	4/13 to 20	5/14 to 21			—	—	—	—
	MOVFPE	(EAs) → Rd Synchronized with E clock	B	0	2		3/13 to 20	4/13 to 20	5/14 to 21	3/13 to 20	3/14 to 21	4/13 to 20	5/14 to 21			—	—	—	—
Arithmetic Operations	ADD:G.B	Rd + (EAs) → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	↑	↑
	ADD:G.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	↑	↑
	ADD:Q.B	(EAd) + #IMM → (EAd)	B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	↑	↑
	ADD:Q.W	(#IMM=±1, ±2) (short format)	W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	↑	↑
	ADDS.B	Rd + (EAs) → Rd	B	1	1	2/3	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		—	—	—	—
	ADDS.W	(Rd is always word size)	W	2	1	2/3	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	—	—	—	—
	ADDX.B	Rd + (EAs) + C → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	↑	↑
	ADDX.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	↑	↑
	DADD	(Rd) ₁₀ + (Rs) ₁₀ + C → (Rd) ₁₀	B	0	2	3/4										—	↑	—	↑
	SUB.B	Rd – (EAs) → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	↑	↑
	SUB.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	↑	↑
	SUBS.B	Rd – (EAs) → Rd	B	1	1	2/3	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		—	—	—	—
	SUBS.W		W	2	1	2/3	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	—	—	—	—
	SUBX.B	Rd – (EAs) – C → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	↑	↑
	SUBX.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	↑	↑
	DSUB	(Rd) ₁₀ – (Rs) ₁₀ – C → (Rd) ₁₀	B	0	2	3/4										—	↑	—	↑
	MULXU.B	Rdx (EAs) → Rd 8 × 8	B	1	1	2/16	2/19	3/19	4/20	2/19	2/20	3/19	4/20	3/18		↑	↑	0	0
	MULXU.W	(unsigned) 16 × 16	W	2	1	2/23	2/25	3/25	4/26	2/25	2/26	3/25	4/26		4/25	↑	↑	0	0
	DIVXU.B	Rd ÷ (EAs) → Rd 16 ÷ 8	B	1	1	2/20	2/23	3/23	4/24	2/23	2/24	3/23	4/24	3/21		↑	↑	↑	0
	DIVXU.W	(unsigned) 32 ÷ 16	W	2	1	2/26	2/29	3/29	4/30	2/29	2/30	3/29	4/30		4/28	↑	↑	↑	0
	CMP:G.B	R _n – (EAs), set CCR	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	↑	↑
	CMP:G.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	↑	↑
	CMP:G.B	(EAd) – #IMM, set CCR	B	1	2	3/3	3/6	4/6	5/7	3/6	3/7	4/6	5/7			↑	↑	↑	↑
	CMP:G.W		W	2	3	4/4	4/7	5/7	6/8	4/7	4/8	5/7	6/8			↑	↑	↑	↑
	CMP:E	Rd – #IMM, set CCR	B	0	0									2/2		↑	↑	↑	↑
	CMP:I	(short format)	W	0	0										3/3	↑	↑	↑	↑

Mnemonic		Operation	Oper- and Size	Instruction Byte Length/No. of Execution States												CCR Flags			
						Rn	@ Rn	@ (d:8, Rn)	@ (d:16, Rn)	@ -Rn	@ Rn+	@ aa:8	@ aa:16	#xx:8	#xx:16	N	Z	V	C
				i	j/k														
Arithmetic Operations	EXTS	(<bit 7> of <Rd>) → (<bits 15 to 8> of <Rd>)	B	0	1	2/3										↑	↑	0	0
	EXTU	0 → (<bits 15 to 8> of <Rd>)	B	0	1	2/3										0	↑	0	0
	TST.B	(EAd) – 0, set CCR	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6			↑	↑	0	0
	TST.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6			↑	↑	0	0
	NEG.B	0 – (EAd) → (EAd)	B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	NEG.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	CLR.B	0 → (EAd)	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6			0	1	0	0
	CLR.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6			0	1	0	0
TAS	(EAd) – 0, set CCR (1) ₂ → (<bit 7> of <EAd>)	B	2	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	0	
Shift Operations	SHAL.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	↑	↑
	SHAL.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	↑	↑
	SHAR.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	SHAR.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	SHLL.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	SHLL.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	SHLR.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			0	↑	0	↑
	SHLR.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			0	↑	0	↑
	ROTL.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	ROTL.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	ROTR.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	ROTR.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	ROTXL.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	ROTXL.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	ROTXR.B		B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
	ROTXR.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	↑
Logic Operations	AND.B	Rd ∧ (EAs) → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	0	—
	AND.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	0	—
	OR.B	Rd ∨ (EAs) → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	0	—
	OR.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	0	—
	XOR.B	Rd ⊕ (EAs) → Rd	B	1	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6	3/3		↑	↑	0	—
	XOR.W		W	2	1	2/2	2/5	3/5	4/6	2/5	2/6	3/5	4/6		4/4	↑	↑	0	—
	NOT.B	¬ (EAd) → (EAd)	B	2	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	—
	NOT.W		W	4	1	2/2	2/7	3/7	4/8	2/7	2/8	3/7	4/8			↑	↑	0	—
Bit Manipulations	BSET.B	¬ (<bit-number> of <EAd>) → Z	B	2	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	↑	—	—
	BSET.W	1 → (<bit-number> of <Rn>)	W	4	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	↑	—	—
	BCLR.B	¬ (<bit-number> of <EAd>) → Z	B	2	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	↑	—	—
	BCLR.W	0 → (<bit-number> of <Rn>)	W	4	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	↑	—	—
	BTST.B	¬ (<bit-number> of <EAd>) → Z	B	1	1	2/3	2/5	3/5	4/6	2/5	2/6	3/5	4/6			—	↑	—	—
	BTST.W		W	2	1	2/3	2/5	3/5	4/6	2/5	2/6	3/5	4/6			—	↑	—	—
	BNOT.B	¬ (<bit-number> of <EAd>) → Z	B	2	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	↑	—	—
	BNOT.W	→ (<bit-number> of <Rn>)	W	4	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	↑	—	—
	LDC.B	(EAs) → CR	B	1	1	2/3	2/6	3/6	4/7	2/6	2/7	3/6	4/7	3/4		Δ	Δ	Δ	Δ
LDC.W		W	2	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8		4/6	Δ	Δ	Δ	Δ	
System Control	STC.B	CR → (EAd)	B	1	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	—	—	—
	STC.W		W	2	1	2/4	2/7	3/7	4/8	2/7	2/8	3/7	4/8			—	—	—	—
	ANDC.B	CR ∧ #IMM → CR	B	0	1									3/5		Δ	Δ	Δ	Δ
	ANDC.W		W	0	1										4/9	Δ	Δ	Δ	Δ
	ORC.B	CR ∨ #IMM → CR	B	0	1									3/5		Δ	Δ	Δ	Δ
	ORC.W		W	0	1										4/9	Δ	Δ	Δ	Δ
	XORC.B	CR ⊕ #IMM → CR	B	0	1									3/5		Δ	Δ	Δ	Δ
	XORC.W		W	0	1										4/9	Δ	Δ	Δ	Δ

Mnemonic		Operation			Addressing Modes	Instruction Byte Length/No. of Execution States	i	j	k	CCR Flags			
										N	Z	V	C
Branching Instructions	BCC	If condition is true, then PC + disp → PC else next;			@(d:8, PC), Condition false, branch not taken	2/3	0	1	1	—	—	—	—
		Mnemonic	Description	Condition	@(d:8, PC), Condition true, branch taken	2/7	0	5	0				
		BRA (BT)	Always (true)	True	@(d:16, PC), Condition false, branch not taken	3/3	0	1	2				
		BRN (BF)	Never (false)	False	@(d:16, PC), Condition true, branch taken	3/7	0	5	1				
		BHI	High	$C \vee Z = 0$									
		BLS	Low or same	$C \vee Z = 1$									
		BCC (BHS)	Carry clear (high or same)	$C = 0$									
		BCS (BLO)	Carry set (low)	$C = 1$									
		BNE	Not equal	$Z = 0$									
		BEQ	Equal	$Z = 1$									
		BVC	Overflow clear	$V = 0$									
		BVS	Overflow set	$V = 1$									
		BPL	Plus	$N = 0$									
		BMI	Minus	$N = 1$									
		BGE	Greater or equal	$N \oplus V = 0$									
		BLT	Less than	$N \oplus V = 1$									
		BGT	Greater than	$Z \vee (N \oplus V) = 0$									
		BLE	Less or equal	$Z \vee (N \oplus V) = 1$									
Branch within Page	JMP	Effective address → PC			@aa:16	3/7	0	5	0	—	—	—	—
					@Rn	2/6	0	5	0				
					@(d:8, Rn)	3/7	0	5	0				
					@(d:16, Rn)	4/8	0	5	1				
	BSR	PC → @-SP PC + disp → PC			@(d:8, PC)	2/9	2	4	0	—	—	—	—
					@(d:16, PC)	3/9	2	4	1				
	JSR	PC → @-SP Effective address → PC			@aa:16	3/9	2	5	0	—	—	—	—
					@Rn	2/9	2	5	0				
					@(d:8, Rn)	3/9	2	5	0				
					@(d:16, Rn)	4/10	2	5	1				
	RTS	@SP + → PC				1/8	2	4	0	—	—	—	—
	RTD	@SP + → PC SP + #IMM → SP			#xx:8	2/9	2	4	0	—	—	—	—
					#xx:16	3/9	2	5	0				
	SCB	If condition is true then next; else Rn - 1 → Rn; if Rn = -1 then next else PC + disp → PC;			@(d:8, PC), Condition true, branch not taken	3/3	0	3	0	—	—	—	—
					@(d:8, PC), Rn = -1, branch not taken	3/4	0	3	0				
					@(d:8, PC), Not above, branch taken	3/8	0	6	0				
Branch Across Page Boundary	PJMP	Effective address → CP, PC			@aa:24	4/9	0	6	0	—	—	—	—
					@Rn	2/8	0	5	0				
	PJSR	PC → @ - SP, CP → @ - SP Effective address → CP, PC			@aa:24	4/15	4	6	0	—	—	—	—
					@Rn	2/13	4	5	0				
	PRTS	@SP + → CP @SP + → PC				2/12	4	5	0	—	—	—	—
	PRTD	@SP + → CP @SP + → PC SP + #IMM → SP			#xx:8	3/13	4	5	0	—	—	—	—
					#xx:16	4/13	4	6	0				

Mnemonic		Operation	Addressing Modes	Instruction Byte Length/No. of Execution States	i	j	k	CCR Flags			
								N	Z	V	C
System Control	TRAPA	PC → @-SP (if max mode then CP → @-SP) SR → @-SP (if max mode then <vector> → CP) <vector> → PC	Minimum mode	2/17	6	4	0	—	—	—	—
			Maximum mode	2/22	10	4	0				
	TRAP/VS	If V flag = "1" then TRAP else next;	V = 0, Branch not taken	1/3	0	1	0	—	—	—	—
			V = 1, Branch taken, minimum mode	1/18	6	4	0				
			V = 1, Branch taken, minimum mode	1/23	10	4	0				
	RTE	@SP + → SR (If max mode then @SP + → CP) @SP + → PC	Minimum mode	1/13	4	4	0	↑	↑	↑	↑
			Maximum mode	1/15	6	4	0				
	LINK	FP (R6) → @-SP SP → FP (R6) SP + #IMM → SP	#xx:8	2/6	2	2	0	—	—	—	—
			#xx:16	3/7	2	3	0				
	UNLK	FP (R6) → SP @SP + → FP		1/5	2	1	0	—	—	—	—
	SLEEP	Normal running state → power-down state	Transition to sleep mode	1/2	0	0	0	—	—	—	—
	NOP	PC + 1 → PC		1/2	0	1	0	—	—	—	—

Notes: 1. Parameters for calculating number of execution states when external memory is used.

- i: Number of operand bytes written or read
- j + k: Number of instruction bytes fetched
- 2. n: Number of registers saved or restored

Number of Execution States: One state is one cycle of the system clock ϕ . If $\phi = 16\text{ MHz}$ then one state = 62.5 ns.

Instruction Fetch	Operand Read/Write	Formula	
16-bit-bus, 2-state-access area	16-bit-bus, 2-state-access area or general register	Value in table	
	16-bit-bus, 3-state-access area	Byte	Value in table + i
		Word	Value in table + i/2
	8-bit-bus, 3-state-access area or on-chip supporting module	Byte	Value in table + i
		Word	Value in table + 2i
16-bit-bus, 3-state-access area	16-bit-bus, 2-state-access area or general register	Value in table + (j + k)/2	
	16-bit-bus, 3-state-access area	Byte	Value in table + i + (j + k)/2
		Word	Value in table + (i + j + k)/2
	8-bit-bus, 3-state-access area or on-chip supporting module	Byte	Value in table + i + (j + k)/2
		Word	Value in table + 2i + (j + k)/2
8-bit-bus, 3-state-access area	16-bit-bus, 2-state-access area or general register	Value in table + 2(j + k)	
	16-bit-bus, 3-state-access area	Byte	Value in table + i + 2(j + k)
		Word	Value in table + i/2 + 2(j + k)
	8-bit-bus, 3-state-access area or on-chip supporting module	Byte	Value in table + i + 2(j + k)
		Word	Value in table + 2(i + j + k)

- Notes:
1. When an instruction is fetched from the 16-bit-bus access area, the number of states differs by 1 or 2 depending on whether the instruction is stored at an even or odd address. This point should be noted in software timing routines and other situations in which the precise number of states must be known.
 2. If wait states or T_p states are inserted in access to the 3-state-access area, add the necessary number of states.
 3. When an instruction is fetched from the 16-bit-bus 3-state-access area, fractions in the term (j + k)/2 should be rounded up.

■ Examples of Calculation of Number of States Required for Execution

Example 1: Instruction fetched from 16-bit-bus, 2-state-access area

Operand Read/Write	Start Address	Assembler Notation			Formula (Value in Table)	Execution States
		Address	Code	Mnemonic		
16-bit-bus, 2-state-access area or general register	Even	H'0100	D821	ADD@R0, R1	$5 + 1$	6
	Odd	H'0100	D821	ADD@R0, R1	$5 + 0$	5

Example 2: Instruction fetched from 16-bit-bus, 2-state-access area

Operand Read/Write	Start Address	Assembler Notation			Formula (Value in Table) + 2i	Execution States
		Address	Code	Mnemonic		
On-chip supporting module or 8-bit-bus, 3-state-access area (word)	Even	H'FC00	IID8	JSR@R0	$9 + 0 + 2 \times 2$	13
	Odd	H'FC01	IID8	JSR@R0	$9 + 1 + 2 \times 2$	14

Example 3: Instruction fetched from 8-bit-bus, 3-state-access area

Operand Read/Write	Assembler Notation			Formula (Value in Table) + 2(j + k)	Execution States
	Address	Code	Mnemonic		
16-bit-bus, 2-state-access area or general register	H'9002	D821	ADD@R0, R1	$5 + 2 \times (1 + 1)$	9

Example 4: Instruction fetched from 16-bit-bus, 3-state-access area

Operand Read/Write	Start Address	Assembler Notation			Formula (Value in Table)	Execution States
		Address	Code	Mnemonic		
16-bit-bus, 2-state-access area or general register	Even	H'0100	D821	ADD@R0, R1	$5 + 1 + (1 + 1)/2$	7
	Odd	H'0101	D821	ADD@R0, R1	$5 + 0 + (1 + 1)/2$	6

Operation Notation

Rd	General register (destination operand)	FP	Frame pointer
Rs	General register (source operand)	#IMM	Immediate data
Rn	General register	disp	Displacement
(EAd)	Destination operand	+	Add
(EAs)	Source operand	−	Subtract
CCR	Condition code register	×	Multiply
N	N (Negative) flag in CCR	÷	Divide
Z	Z (Zero) flag in CCR	^	AND
V	V (Overflow) flag in CCR	∨	OR
C	C (Carry) flag in CCR	⊕	Exclusive OR
CR	Control register	→	Move
PC	Program counter	↔	Swap
CP	Code page register	¬	NOT
SP	Stack pointer		

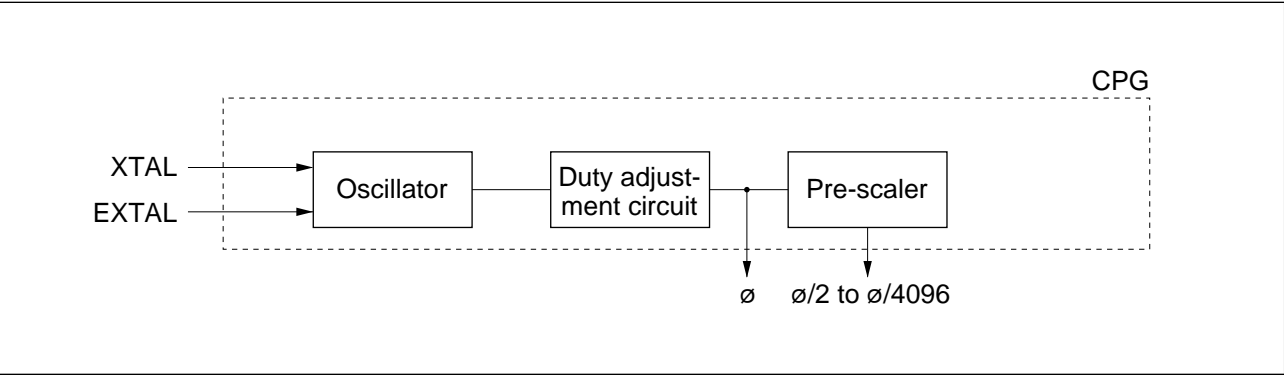
Condition Code Symbols

↑	Changed after instruction execution
0	Cleared to 0
1	Set to 1
−	Value before operation is retained
Δ	Changed depending on condition

4.6 Basic Operational Timing

4.6.1 Basic Clock Timing

The system clock (ϕ) is generated by connecting an external clock signal to the EXTAL pin, or by connecting a crystal resonator across the XTAL and EXTAL pins and supplying a frequency equal to the desired system clock frequency. The duty cycle is adjusted to 50% by a duty adjustment circuit.

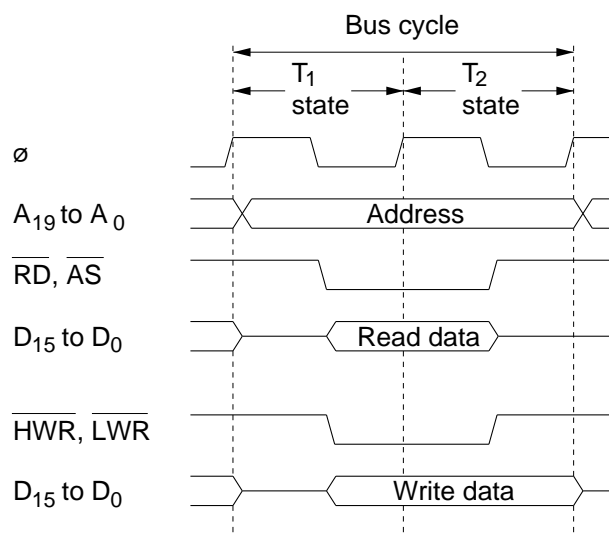


Block Diagram of Clock Oscillator

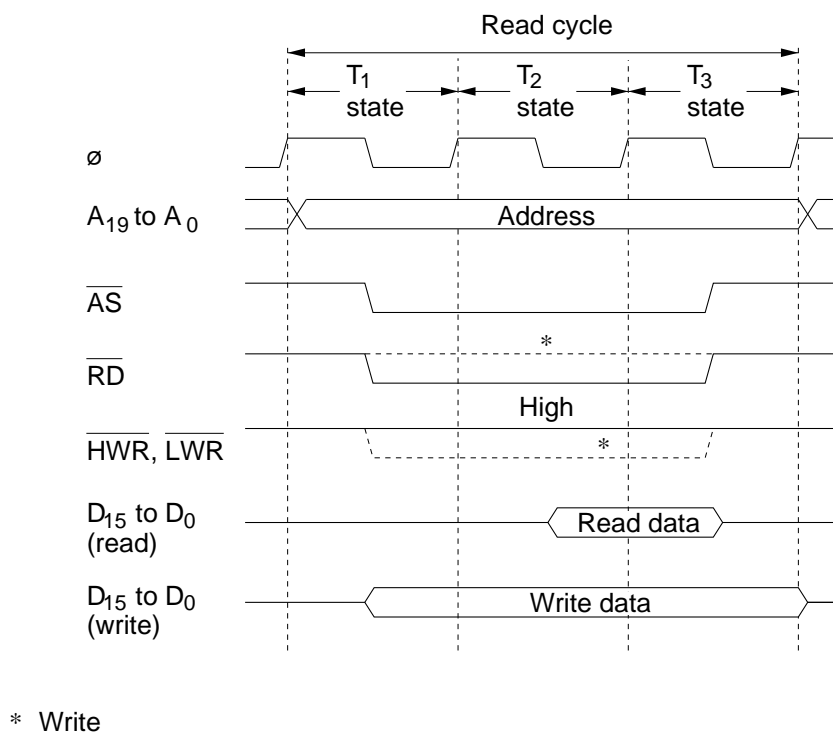
4.6.2 CPU Read/Write Cycle

The time base of a CPU read or write cycle is the ϕ clock, one period of which is called a “state.” The bus cycle consists of two or three states. The bus cycle structure differs depending on whether the access is to on-chip memory, an on-chip supporting module, or an external device.

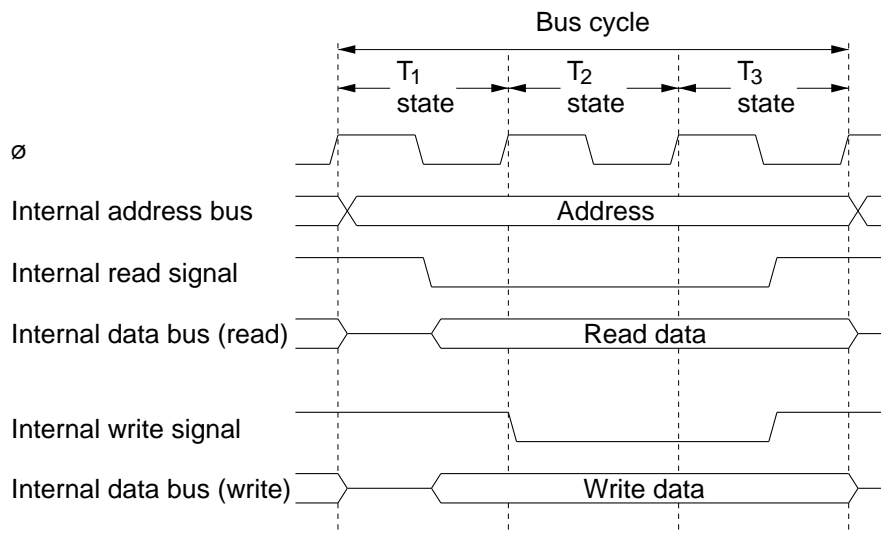
- On-chip memory (RAM or flash memory): For high speed, word access is performed in a two-state bus cycle.
- On-chip supporting module: Byte or word access is performed. The bus cycle consists of three states.



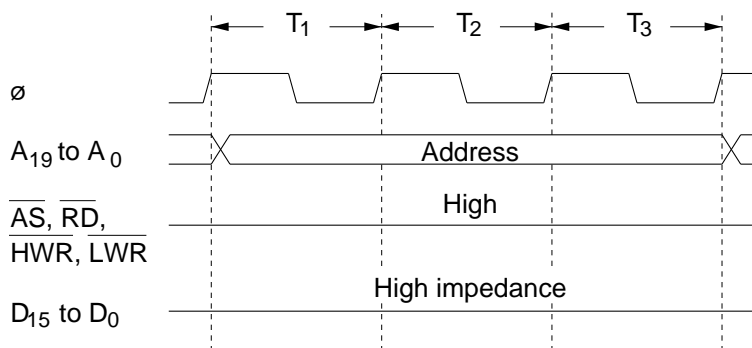
Access Cycle for 2-State-Access Area



Access Cycle for 3-State-Access Area

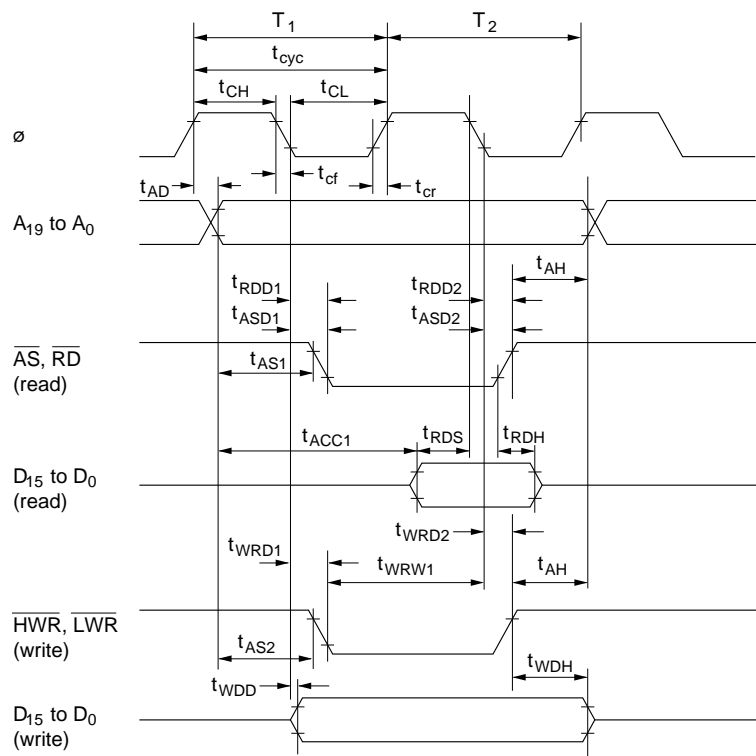


On-Chip Supporting Module Access Cycle

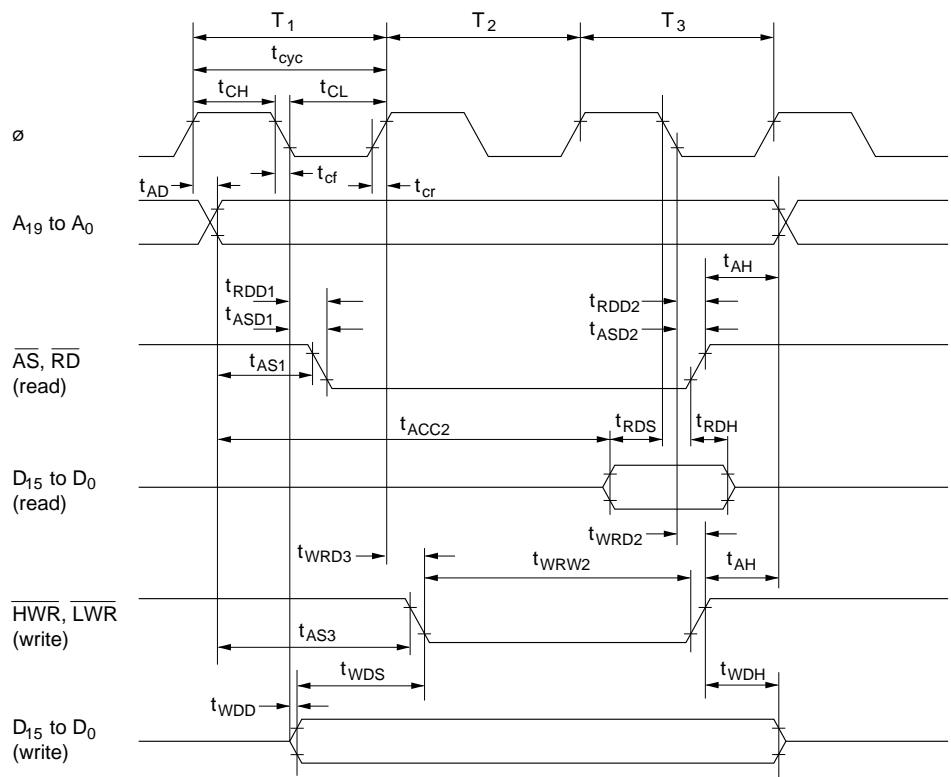


Pin States during On-Chip Supporting Module Access

- External device: Three modes can be selected for accessing external devices: 2-state word access, 3-state word access, and 3-state byte access. Wait states (T_W) cannot be inserted in 2-state access.



External Device Access Cycle (2-State-Access Area)

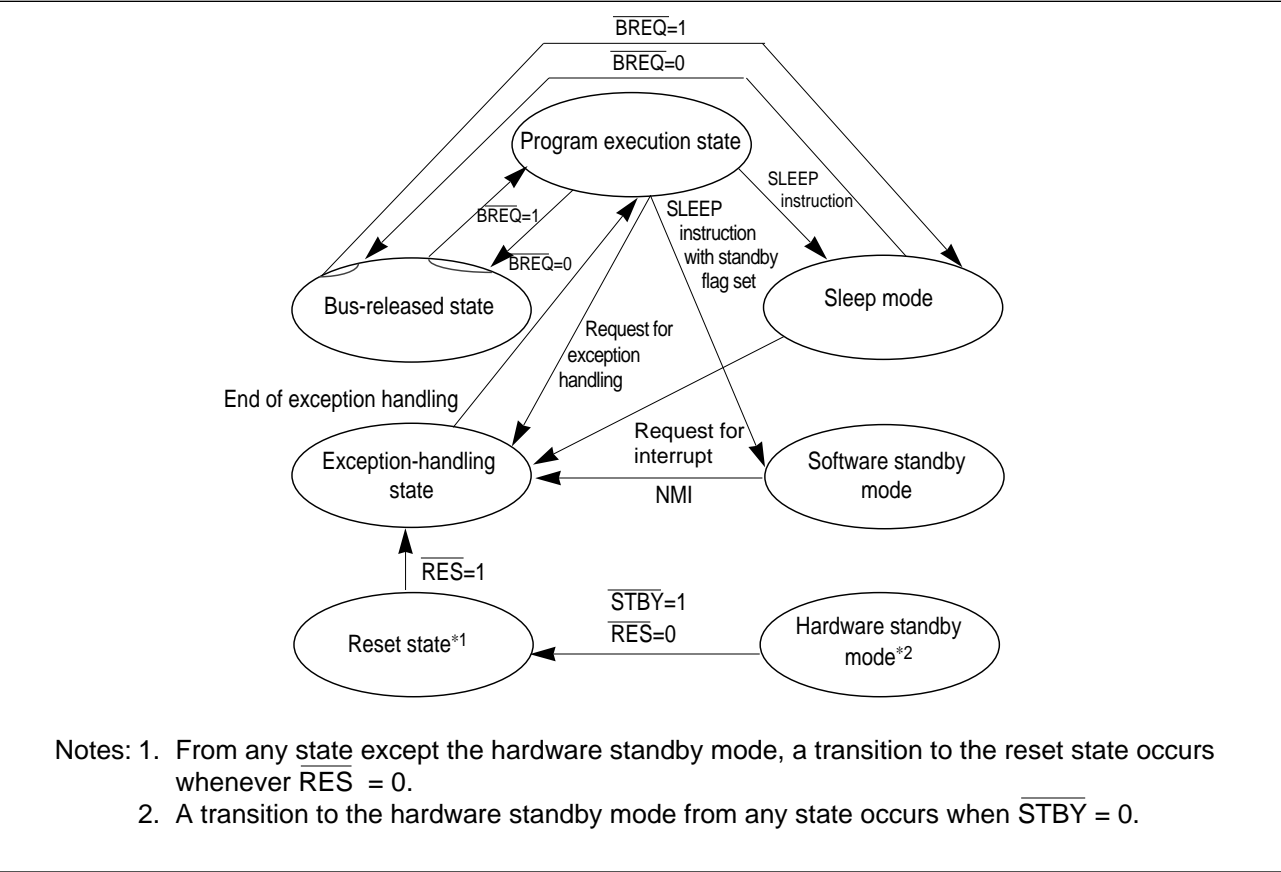


External Device Access Cycle (3-State-Access Area)

Item	Symbol	8 MHz		10 MHz		16 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Clock cycle time	t_{cyc}	125	500	100	500	62.5	500	ns
Clock low pulse width	t_{CL}	45	—	35	—	20	—	ns
Clock high pulse width	t_{CH}	45	—	35	—	20	—	ns
Clock rise time	t_{cr}	—	15	—	15	—	10	ns
Clock fall time	t_{cf}	—	15	—	15	—	10	ns
Address delay time	t_{AD}	—	35	—	25	—	20	ns
Address hold time	t_{AH}	25	—	20	—	20	—	ns
Address strobe delay time 1	t_{ASD1}	—	40	—	35	—	20	ns
Address strobe delay time 2	t_{ASD2}	—	40	—	40	—	20	ns
Read strobe delay time 1	t_{RDD1}	—	40	—	35	—	20	ns
Read strobe delay time 2	t_{RDD2}	—	40	—	40	—	20	ns
Write strobe delay time 1	t_{WRD1}	—	40	—	40	—	20	ns
Write strobe delay time 2	t_{WRD2}	—	40	—	40	—	20	ns
Write strobe delay time 3	t_{WRD3}	—	40	—	40	—	20	ns
Write data strobe pulse width 1	t_{WRW1}	110	—	90	—	60	—	ns
Write data strobe pulse width 2	t_{WRW2}	150	—	120	—	90	—	ns
Address setup time 1	t_{AS1}	20	—	20	—	20	—	ns
Address setup time 2	t_{AS2}	20	—	20	—	20	—	ns
Address setup time 3	t_{AS3}	80	—	65	—	45	—	ns
Read data setup time	t_{RDS}	30	—	20	—	5	—	ns
Read data hold time	t_{RDH}	0	—	0	—	5	—	ns
Read data access time 1	t_{ACC1}	—	125	—	100	—	60	ns
Read data access time 2	t_{ACC2}	—	230	—	200	—	120	ns
Write data delay time	t_{WDD}	—	65	—	65	—	55	ns
Write data setup time	t_{WDS}	15	—	10	—	5	—	ns
Write data hold time	t_{WDH}	25	—	20	—	20	—	ns

4.7 Operating States

The CPU operates in one of five states: the program execution state, exception-handling state, bus-released state, reset state, or power-down state. The diagram below shows the state transitions.



State Transitions

4.7.1 Program Execution State

In this state the CPU executes the current program.

4.7.2 Exception-Handling State

This is a transient state in which the CPU executes exception processing to handle an external interrupt, trap instruction, address error, or other exception event.

4.7.3 Bus-Released State

This is the state in which the CPU has relinquished the bus to an external device. When the CPU receives a bus request (\overline{BREQ}) signal from an external device, it releases the bus to that device at the end of the current bus cycle.

4.7.4 Power-Down State

In the power-down state the CPU halts to save power. There are three power-down modes:

- Sleep mode: Execution of the SLEEP instruction places the CPU in the sleep mode, in which the CPU is halted but the on-chip supporting modules remain operative.
- Software standby mode: When the software standby flag is set, execution of the SLEEP instruction places the CPU in the software standby mode. All chip functions halt, including the clock, but the contents of CPU registers and on-chip RAM are held.
- Hardware standby mode: A low input at the $\overline{\text{STBY}}$ pin places the CPU in the hardware standby mode. All chip functions halt, including the clock, but the contents of on-chip RAM are held.

Current Dissipation (Typical Values for 16-MHz Operation)

— Preliminary —

Running	100 mA
Sleep mode	60 mA
Standby mode	0.01 μA

Power-Down State

Mode	Clock	CPU	Supporting Functions	On-Chip RAM, CPU Registers	Recovery Methods
Sleep	Runs	Halts	Run	Held	Interrupt—When an interrupt is received, interrupt handling begins. $\overline{\text{RES}}$ —Transition to reset state. $\overline{\text{STBY}}$ —Transition to hardware standby mode.
Software standby	Halts	Halts	Halt and initialized	Held	NMI—NMI starts clock; NMI exception handling starts automatically after time set in watchdog timer. $\overline{\text{RES}}$ —Clock starts, followed by transition to reset state. $\overline{\text{STBY}}$ —Transition to hardware standby mode.
Hardware standby	Halts	Halts	Halt and initialized	Held*	Recovered after high input at $\overline{\text{STBY}}$ pin and low input at $\overline{\text{RES}}$ pin followed, after clock settling time, by high input at $\overline{\text{RES}}$ pin.

Note: * On-chip RAM only.

4.8 Exception Handling

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

There is a priority order among the different types of exceptions. If two or more exceptions occur simultaneously, they are handled in their order of priority. An instruction exception cannot occur simultaneously with other types of exceptions.

Exceptions and Their Priority

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

Instruction Exceptions

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

4.9 Interrupts

There are 44 interrupt sources: five external sources (NMI and $\overline{\text{IRQ}}_0$ to $\overline{\text{IRQ}}_3$) and 39 sources in on-chip supporting modules. Each type of interrupt is assigned a unique vector number. When an interrupt occurs the CPU saves the program counter and status register (in minimum mode) or program counter, status register, and code page register (in maximum mode) to the stack, then reads the starting address of the interrupt handling routine from the exception vector table and starts the interrupt handling routine.

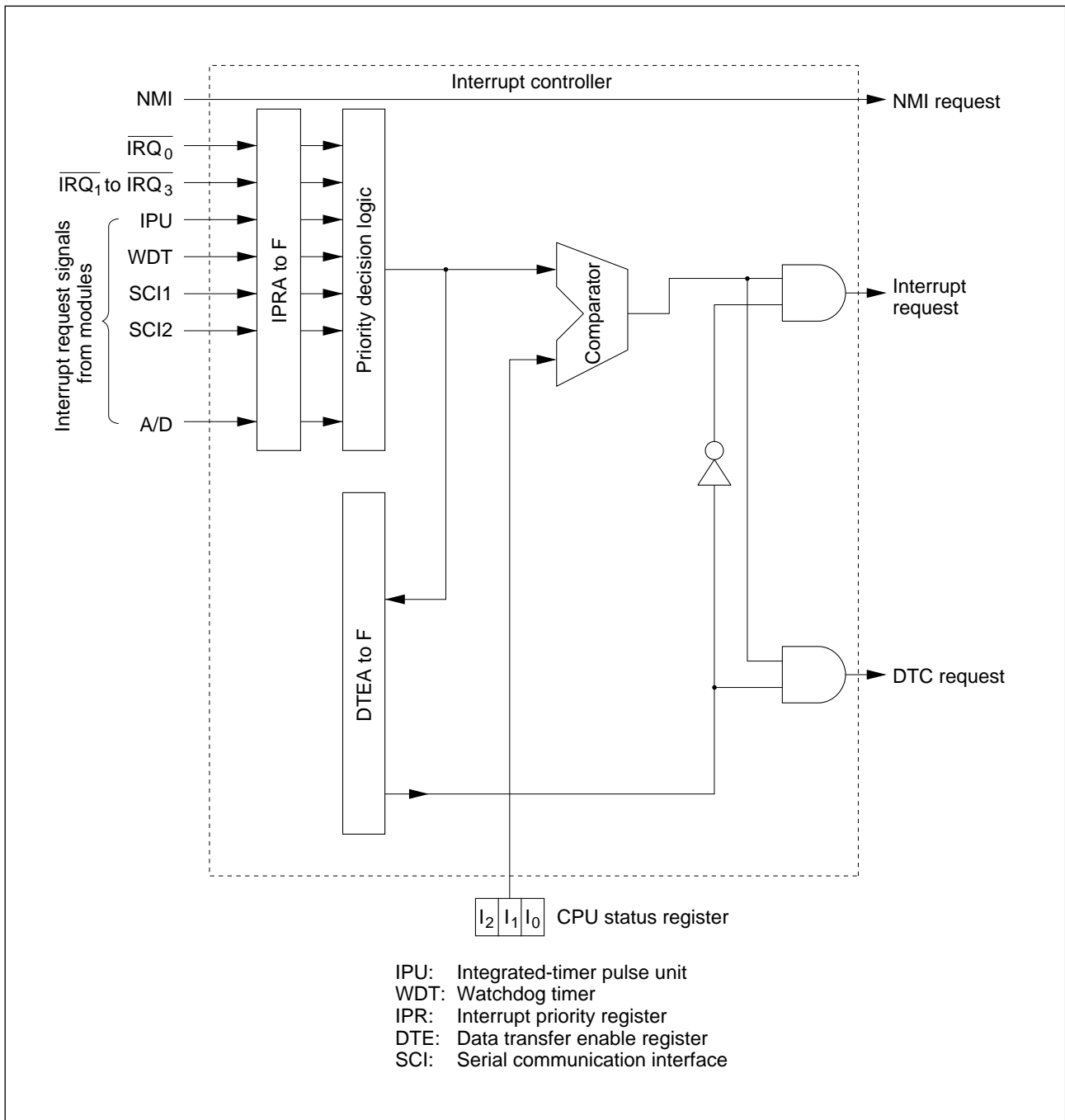
Interrupts and Their Priorities

Interrupt (number of sources)		Priority	Priority Among Interrupts at the Same Level	
External interrupts	NMI	(1)	Highest priority	
	$\overline{\text{IRQ}}_0$	(1)	Any priority from 7 (high) to 0 (low) can be assigned.	High
	$\overline{\text{IRQ}}_1$ to $\overline{\text{IRQ}}_3$	(3)		
Internal interrupts	16-bit integrated-timer pulse unit	(29)		
	Serial communication interface 1	(4)		
	Serial communication interface 2	(4)		
	A/D converter	(1)		
	Watchdog timer	(1)		Low

4.10 Interrupt Controller

The interrupt controller controls interrupts by using six interrupt priority registers (IPRA to IPRF), in which eight priority levels can be assigned to each interrupt source in each module. If two or more interrupts of the same assigned priority level occur at once, they are handled in the order indicated in the preceding table, with lower interrupts held pending while higher ones are handled. NMI has the highest priority and is always accepted.

The interrupt controller also has six data transfer enable registers (DTEA to DTEF). Values set in these registers determine whether to start the data transfer controller for each interrupt except NMI. Use of this function enables data to be transferred between I/O and memory without involving the CPU.



Block Diagram of Interrupt Controller

4.11 Bus Release

When requested by an external device, the CPU can release the bus. This function is controlled at the $\overline{\text{BREQ}}$ and $\overline{\text{BACK}}$ pins.

- $\overline{\text{BREQ}}$ pin: Input pin for bus request signals from external devices.
- $\overline{\text{BACK}}$ pin: Output pin for a signal indicating that the bus has been released to an external device.

Section 5 Operating Modes

The H8/538F has two on-board programming modes (boot program mode and user program mode), and seven operating modes: three expanded minimum modes (modes 1, 2, and 6), three expanded maximum modes (modes 3, 4, and 5), and a single-chip mode (mode 7). The H8/538F always operates in one of these modes, as selected by the mode pins (MD₀ to MD₂). On-board programming mode is selected by applying a high voltage (12 V) to the V_{PP} pin. The address space and pin functions vary depending on the mode.

5.1 Expanded Minimum Modes (Modes 1, 2, and 6)

An address space of up to 64 kbytes is accessible. The page registers are not used.

Mode 1: Ports 1 and 2 are the data bus, and ports B and C are the address bus. The address space is 64 kbytes. The on-chip flash memory cannot be used.

Mode 2: Port 1 is the data bus, and ports B and C are the address bus. The address space is 64 kbytes. The on-chip flash memory is usable.

Mode 6: Same as mode 1 except that the entire address space is an 8-bit, 3-state-access area until the bus controller's BCRES bit (bit 7) is set to 1.

5.2 Expanded Maximum Modes (Modes 3, 4, and 5)

The page registers are valid and an address space of up to 1 Mbyte is accessible.

Mode 3: Ports 1 and 2 are the data bus, and ports A, B, and C are the address bus. The address space is 1 Mbyte. The on-chip flash memory cannot be used.

Mode 4: Same as mode 3, except that the on-chip flash memory is usable.

Mode 5: Same as mode 3 except that the entire address space is an 8-bit, 3-state-access area until the bus controller's BCRES bit (bit 7) is set to 1.

5.3 Single-Chip Mode (Mode 7)

The H8/538 operates using only its on-chip memory: the on-chip flash memory and on-chip RAM. The external address space is not accessible.

Memory Map in Each Mode

Mode 1 Expanded Minimum Mode	Mode 2 Expanded Minimum Mode	Mode 3 Expanded Maximum Mode	Mode 4 Expanded Maximum Mode	Mode 5 Expanded Maximum Mode	Mode 6 Expanded Minimum Mode	Mode 7 Single-Chip Mode
<div>16-bit bus</div> <div>H'0000</div> <div>H'EDFF H'EE00</div> <div>External address space 16-bit, 2-state access</div> <div>External address space 16-bit, 3-state access</div> <div>H'F67F H'F680</div> <div>On-chip RAM 16-bit, 2-state access</div> <div>H'FE7F H'FE80</div> <div>On-chip registers 8-bit, 3-state access</div> <div>H'FFFF</div>	<div>8-bit bus</div> <div>H'0000</div> <div>H'EE7F H'EE80</div> <div>On-chip flash memory 16-bit, 2-state access</div> <div>External address space 8-bit, 3-state access</div> <div>H'F67F H'F680</div> <div>On-chip RAM 16-bit, 2-state access</div> <div>H'FE7F H'FE80</div> <div>On-chip registers 8-bit, 3-state access</div> <div>H'FFFF</div>	<div>16-bit bus</div> <div>H'00000</div> <div>H'0DFFF H'0E000</div> <div>External address space 16-bit, 2-state access</div> <div>External address space 16-bit, 3-state access</div> <div>H'0F67F H'0F680</div> <div>On-chip RAM 16-bit, 2-state access</div> <div>H'0FE7F H'0FE80</div> <div>On-chip registers 8-bit, 3-state access</div> <div>H'0FFFF H'10000</div> <div>External address space 16-bit, 3-state access</div> <div>H'FFFFFF</div>	<div>16-bit bus</div> <div>H'00000</div> <div>H'0EE7F H'0EE80</div> <div>On-chip flash memory 16-bit, 2-state access</div> <div>External address space 16-bit, 3-state access</div> <div>H'0F67F H'0F680</div> <div>On-chip RAM 16-bit, 2-state access</div> <div>H'0FE7E H'0FE80</div> <div>On-chip registers 8-bit, 3-state access</div> <div>H'0FFFF H'10000</div> <div>External address space 16-bit, 3-state access</div> <div>H'FFFFFF</div>	<div>16-bit bus</div> <div>H'00000</div> <div>H'0F67F H'0F680</div> <div>External address space 8-bit, 3-state access</div> <div>On-chip RAM 16-bit, 2-state access</div> <div>H'0FE7F H'0FE80</div> <div>On-chip registers 8-bit, 3-state access</div> <div>H'0FFFF H'10000</div> <div>External address space 8-bit, 3-state access</div> <div>H'FFFFFF</div>	<div>16-bit bus</div> <div>H'0000</div> <div>H'F67F H'F680</div> <div>External address space 8-bit, 3-state access</div> <div>On-chip RAM 16-bit, 2-state access</div> <div>H'FE7F H'FE80</div> <div>On-chip registers 8-bit, 3-state access</div> <div>H'FFFF</div>	<div>H'0000</div> <div>H'EE7F H'EE80</div> <div>On-chip flash memory 16-bit, 2-state access</div> <div>H'F67F H'F680</div> <div>On-chip RAM 16-bit, 2-state access</div> <div>H'FE7F H'FE80</div> <div>On-chip registers 8-bit, 3-state access</div> <div>H'FFFF</div>

- Notes: 1. If the on-chip RAM is disabled, external access is possible at those addresses.
2. If the on-chip RAM is disabled in single-chip mode, access to those addresses causes an address error.
3. Some timer registers in the on-chip register area support 16-bit, 3-state access.

Section 6 On-Chip Supporting Modules

6.1 Data Transfer Controller

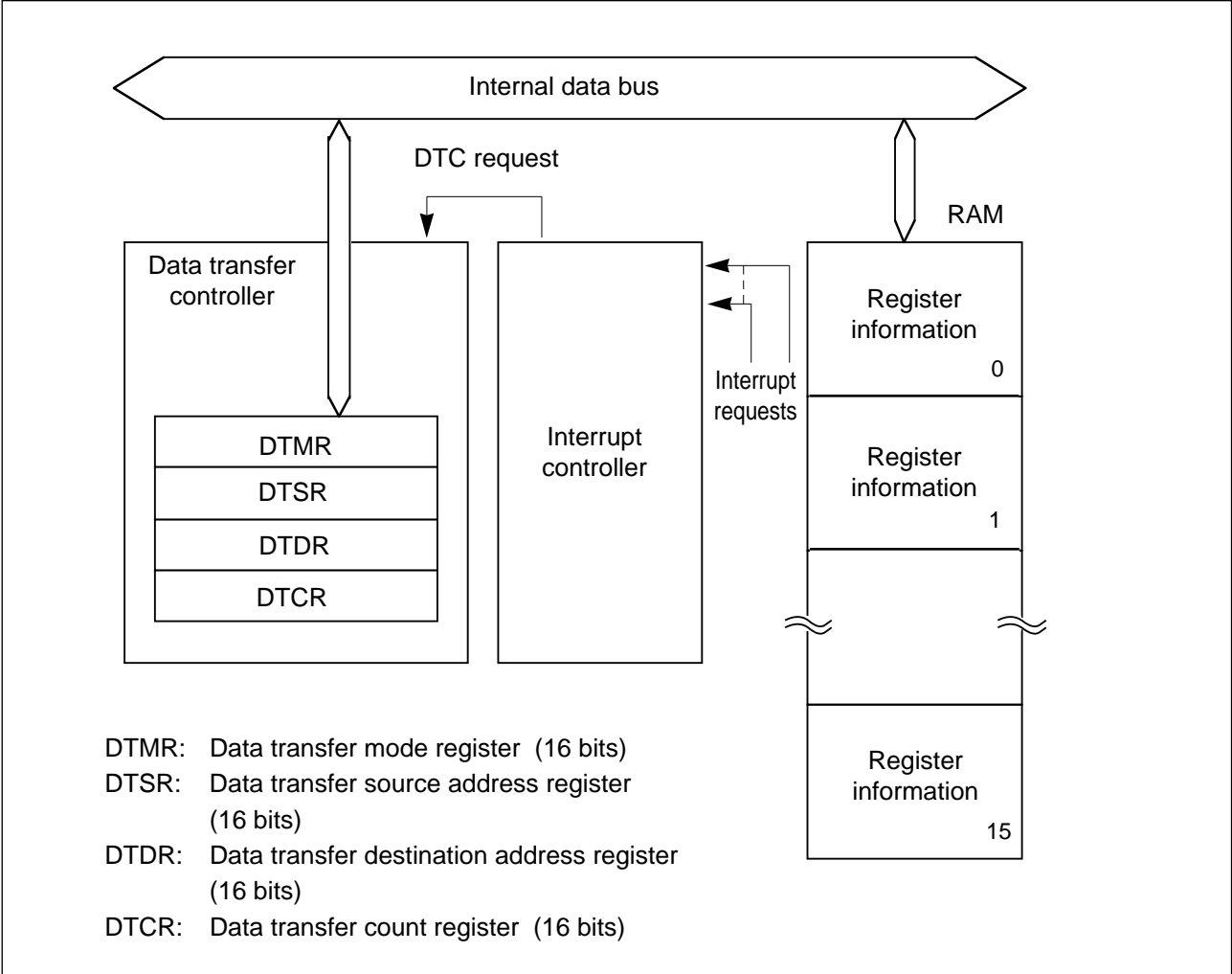
6.1.1 Functions

One of the H8/538F's supporting modules is an on-chip data transfer controller (DTC) for moving data between memory and I/O. The DTC can set data in the I/O ports and the registers of the on-chip supporting modules or move data from these modules to memory without burdening the CPU. Data transfers are performed by starting the DTC with an interrupt.

6.1.2 Features

- Addresses are specified in the 16-bit data transfer source address register (DTSR) and data transfer destination address register (DTDR). The address must be located in the 64 kbytes of page 0.
- The source and destination addresses can be held fixed or incremented.
- A transfer data count of up to 65,536 can be set.
- The transfer data size can be specified in bytes or words, as selected in the data transfer mode register (DTMR).
- An interrupt can be generated at the end of a data transfer. When the data transfer count register (DTCR) reaches 0, the CPU initiates appropriate exception handling.

6.1.3 Block Diagram



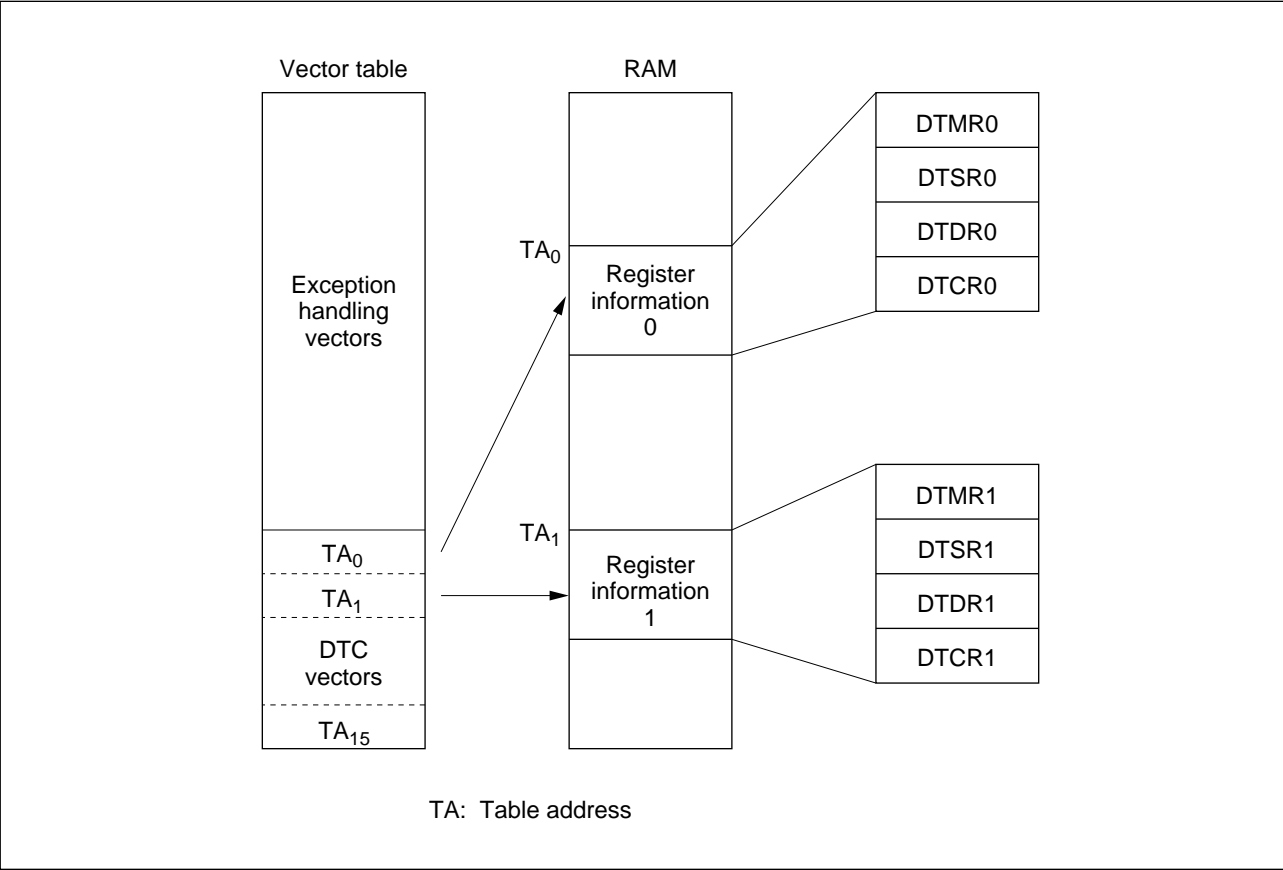
Block Diagram of Data Transfer Controller

6.1.4 Data Transfer Operation

The DTC has internal registers in which the source and destination addresses, transfer data size (byte or word), increment mode, and other information can be specified for one data transfer channel. When a DTC interrupt is requested, the register information for a particular channel is loaded from memory into these internal registers. After the transfer, updated register information is saved back to memory.

DTC interrupt requests are routed like other interrupt requests through the interrupt controller. The interrupt controller interprets an interrupt request as being for the DTC if the corresponding bit of the interrupt controller’s data transfer enable registers (DTEA to DTEF) is set to 1. If the data transfer enable bit is cleared to 0, the interrupt controller treats the request as an ordinary CPU interrupt request.

When the DTC accepts an interrupt, it reads the address at which register information for the specified channel is stored in memory from the DTC vector table, loads the register information from that address, transfers a byte or word of data, then updates and saves the register information. If the updated data transfer count register value is 0, the CPU initiates exception handling corresponding to the interrupt that started the DTC.

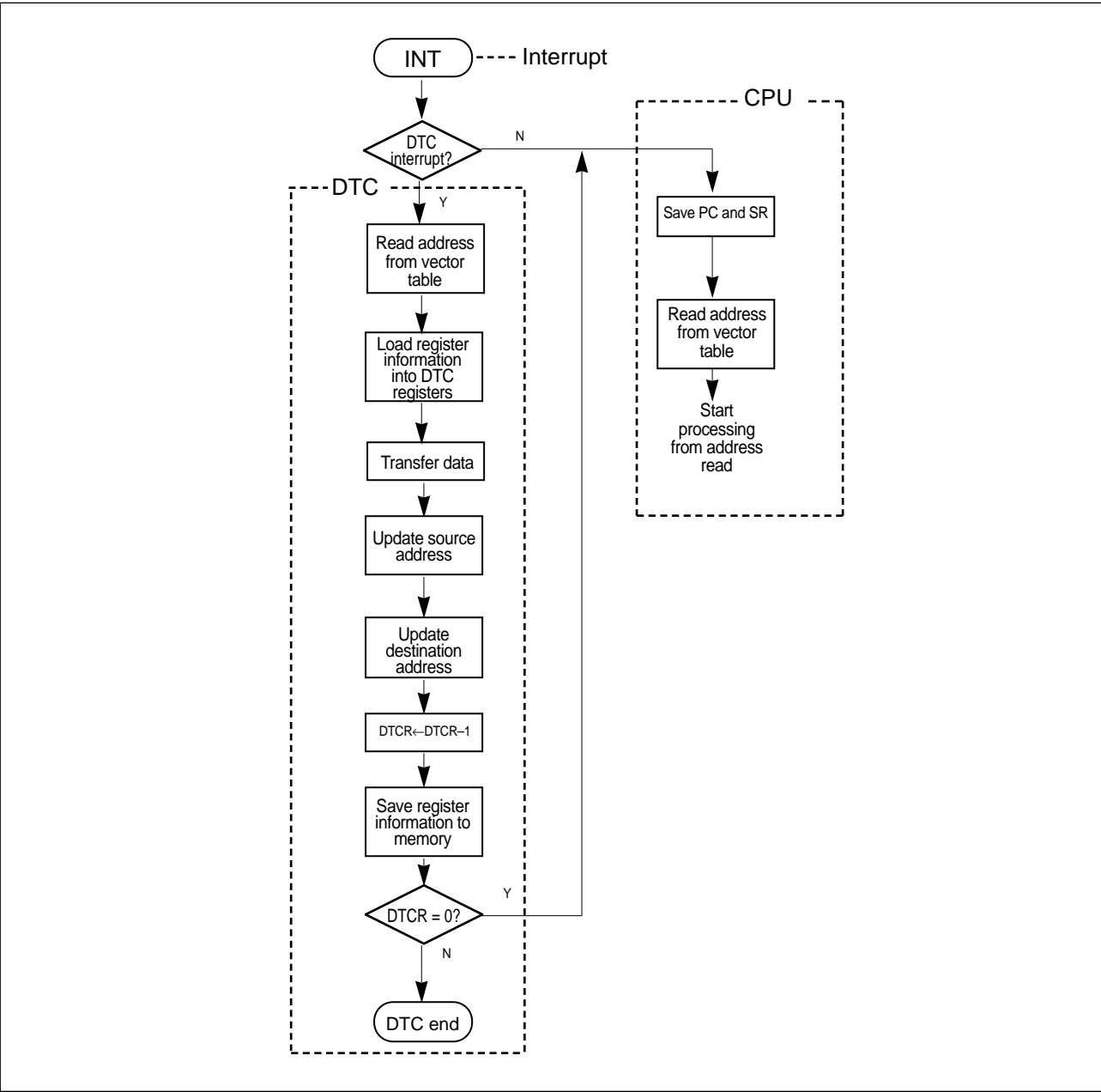


DTC Vector Table

Procedure for Using the DTC

1. Set the transfer mode, source address, destination address, and count value in the address indicated by the vector belonging to the interrupt source.
2. Set the corresponding data transfer enable bit of the interrupt controller to 1, and set the interrupt level.
3. Write appropriate register data in the on-chip supporting module, and enable the interrupt.
4. When the interrupt occurs, the DTC will be activated to perform the transfer.

Flowchart for Starting the DTC



6.1.5 Number of States Required for DTC Transfer

The DTC requires the following number of states per data transfer.

Number of DTC States

Unit: States

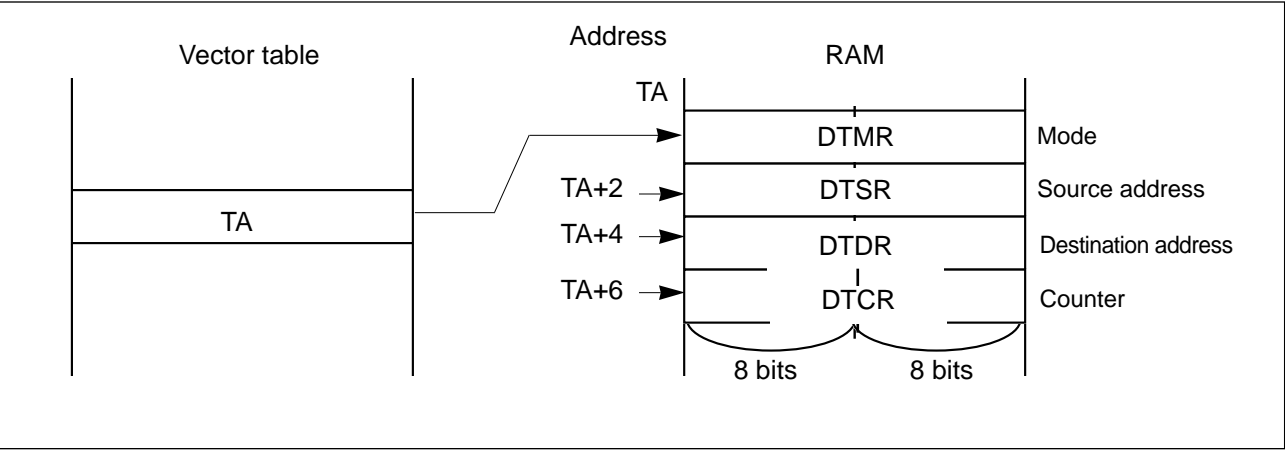
Address Incrementation		16-Bit-Bus, 2-State-Access Area ↔ I/O		8-Bit-Bus, 3-State-Access Area ↔ I/O	
Source	Destination	Byte transfer	Word transfer	Byte transfer	Word transfer
No	No	31	34	32	38
No	Yes	33	36	34	40
Yes	No	33	36	34	40
Yes	Yes	35	38	36	42

Note: These values do not include states spent waiting for instruction termination or for the interrupt controller to determine priority.

6.1.6 Format of Register Information in Memory

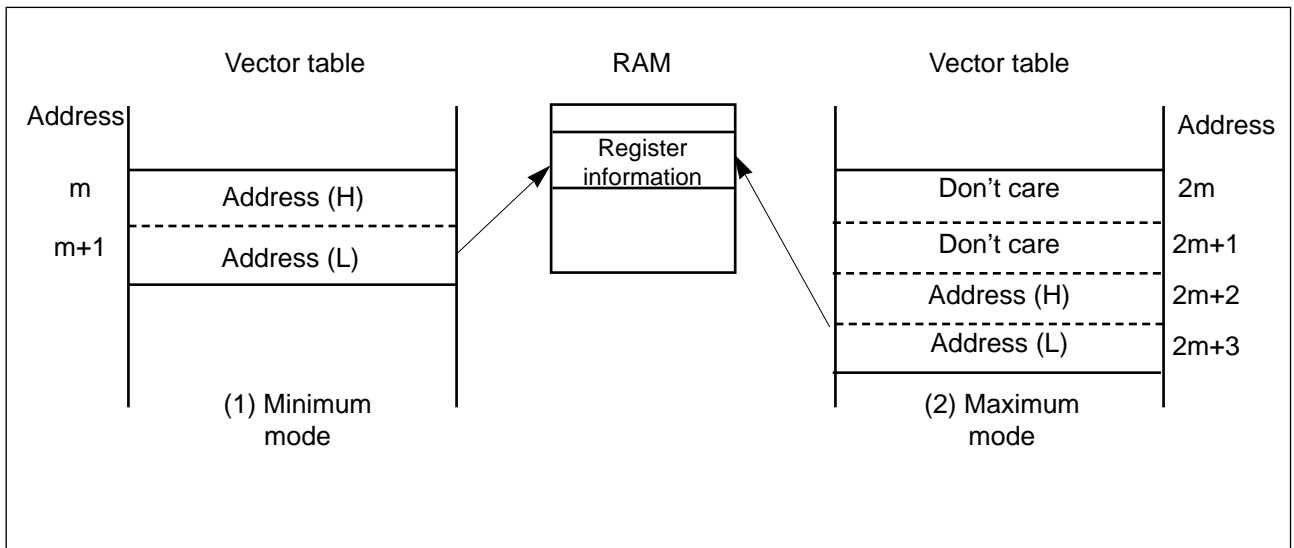
Format of Register Information

The location of the register information for each interrupt source can be designated at any position in page 0 by setting an address in the DTC vector table. For each source, the register information is stored in memory in the following format:



Format of Vector Table Contents

The DTC vector table format is indicated below for minimum mode and maximum mode. In maximum mode the page is assumed to be 0.



6.1.7 Example

An example of continuous reception of serial data using the SCI is shown below.

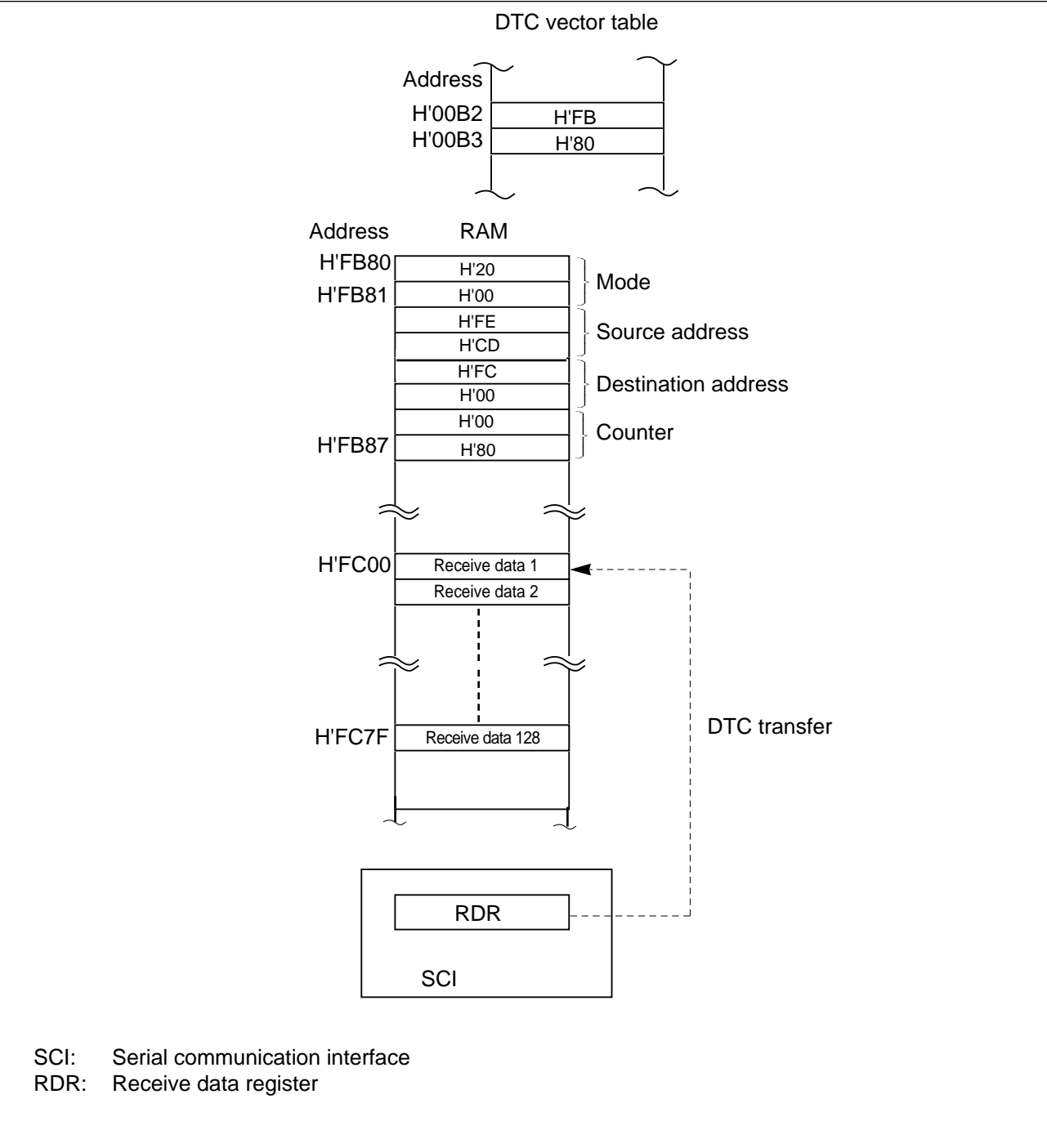
Conditions

- Continuous reception of 128-byte data
- Received data to be stored in RAM in consecutive addresses starting at H'FC00.
- DTC register information is located in RAM addresses H'FB80 to HFB87.
- The CPU is operating in minimum mode.

Procedure

1. Set DTC register information in RAM.
 - Mode: Byte transfer
Fixed source address
Increment destination address
 - Source address: SCI receive data register address
 - Destination address: H'FC00
 - Counter: 128 (H'0080)
2. Set the interrupt controller's data transfer enable bit for SCI receive-data-full to 1, and set the interrupt level.
3. Set the SCI in the required receive mode, and enable SCI interrupts.

- 4. As each byte is received, an interrupt activates the DTC to transfer the received data to RAM.
- 5. When 128 bytes have been transferred (counter value is 0), the CPU begins SCI receive-end interrupt exception handling.
- 6. The interrupt handler routine performs receive-end processing.



6.2 Wait-State Controller

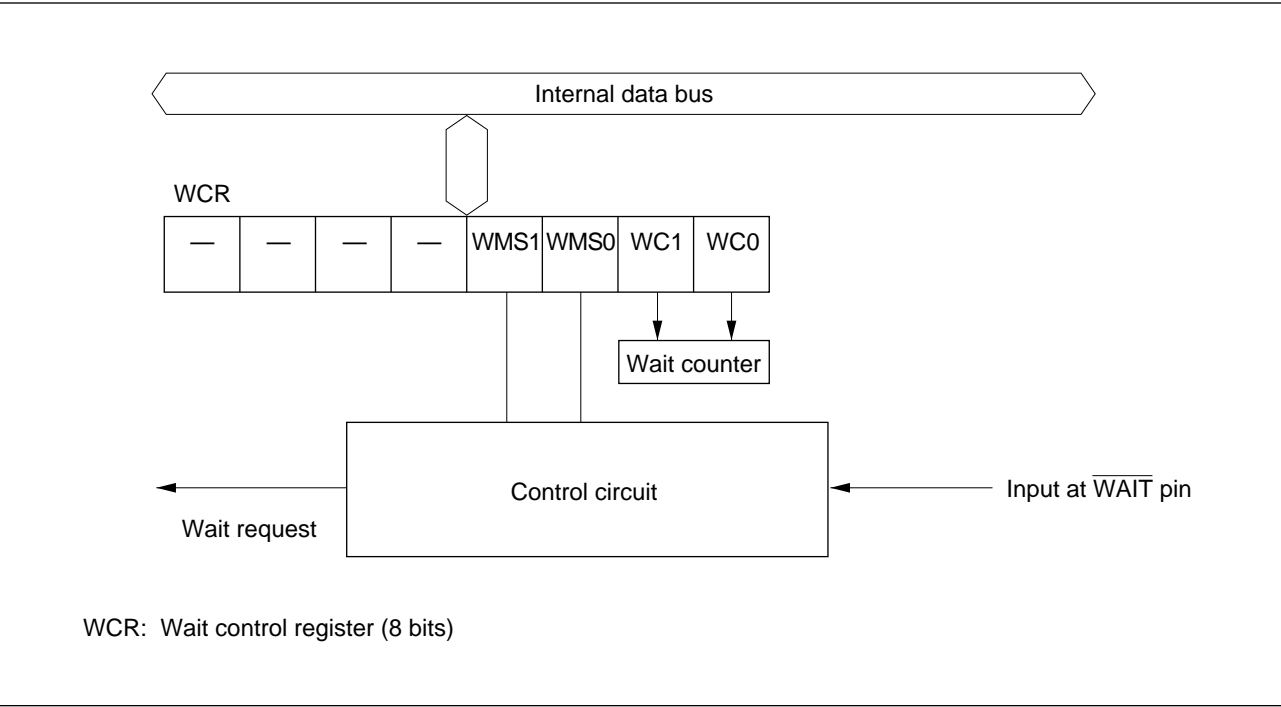
6.2.1 Functions

To simplify interfacing to low-speed external devices, the H8/538F has an on-chip wait-state controller (WSC) that can insert wait states (T_W) between the T_2 state and T_3 state to lengthen the bus cycle. The wait-state function can be used in CPU cycles and DTC cycles. The number of wait states can be selected by a value set in the wait control register (WCR) or by holding the $\overline{\text{WAIT}}$ pin low for the required interval.

6.2.2 Features

- Selection of three operating modes
 - Programmable wait mode
 - Pin wait mode
 - Pin auto-wait mode
- Zero, one, two, or three wait states can be selected. In pin wait mode, four or more wait states can be inserted by holding the $\overline{\text{WAIT}}$ pin low.

6.2.3 Block Diagram



Block Diagram of Wait-State Controller

6.2.4 Programmable Wait Mode

Whenever the CPU or DTC accesses an off-chip address, the number of wait states set in the WCR are inserted.

The external $\overline{\text{WAIT}}$ pin is not used.

6.2.5 Pin Wait Mode

The number of wait states indicated by the WCR are inserted into any bus cycle in which the CPU or DTC accesses an off-chip address.

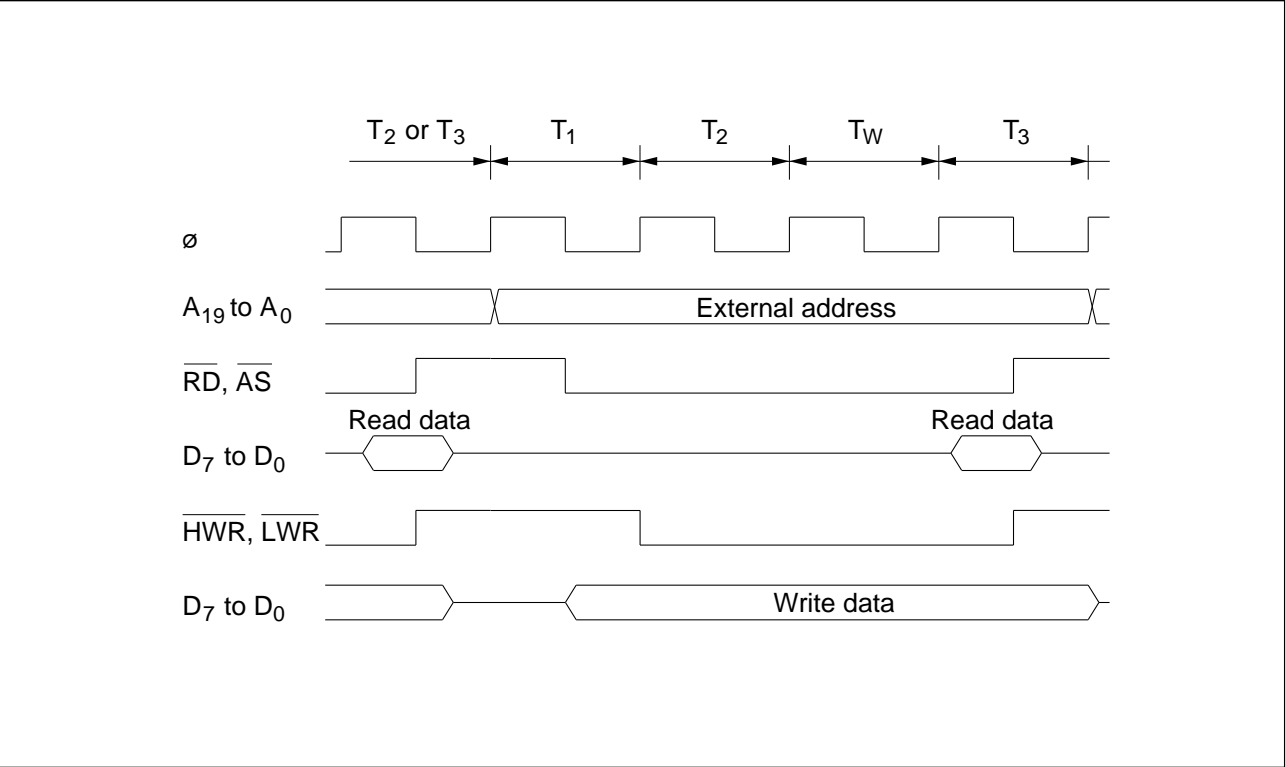
When there is a low input at the $\overline{\text{WAIT}}$ pin, however, wait states are inserted until the $\overline{\text{WAIT}}$ pin goes high. This mode is useful for inserting four or more wait states, or when different external devices require different numbers of wait states.

6.2.6 Pin Auto-Wait Mode

The number of wait states indicated by the WCR are inserted only when there is a low input at the $\overline{\text{WAIT}}$ pin.

Only the indicated number of wait states are inserted, even if the $\overline{\text{WAIT}}$ pin is held low for a longer period of time.

This mode offers a particularly simple way to interface a low-speed device: the wait states can be inserted by routing a decoded address signal to the $\overline{\text{WAIT}}$ pin.



Programmable Wait Mode (When Wait Count is 1)

6.3 Sixteen-Bit Integrated-Timer Pulse Unit

6.3.1 Functions

The H8/538F has a 16-bit integrated-timer pulse unit (IPU) with three types of timers and seven channels. The IPU can output 28 independent waveforms, or can output 14 waveforms while processing 14 pulse inputs and outputs. Other capabilities include six-phase pulse-width-modulated output, automatic pulse width and period measurement, counting of input from a two-phase encoder, and starting the A/D converter.

6.3.2 Features

- Twelve waveform outputs and sixteen pulse inputs or outputs on seven channels, based on three types of timers
- Fourteen registers with software-assignable output compare or input capture functions
- Twenty-eight independent comparators
 - Channel 1 has four output compare registers and four output compare/input capture registers.
 - Channels 2 to 5 have two output compare registers and two output compare/input capture registers.
 - Channels 6 and 7 have two output compare/input capture registers.
- Selection of sixteen counter clock sources

The selection includes \emptyset , $\emptyset/2$, $\emptyset/4$, $\emptyset/8$, $\emptyset/16$, $\emptyset/32$, $\emptyset/64$, $\emptyset/128$, $\emptyset/256$, $\emptyset/512$, $\emptyset/1024$, $\emptyset/2048$, $\emptyset/4096$, $TCLK_1$, $TCLK_2$, and $TCLK_3$. (External clock sources are shared by all channels.)

- Input capture function

The rising edge, falling edge, or both edges can be selected.

- Pulse output modes

One-shot, toggle, or PWM output is available.

- Counter synchronization function

Software can write to two or more timer counters simultaneously. Counters can be cleared simultaneously by compare match or input capture.

- Pulse-width-modulated output

One-phase, two-phase, or three-phase PWM output is available (up to nine-phase PWM output using the counter synchronization function).

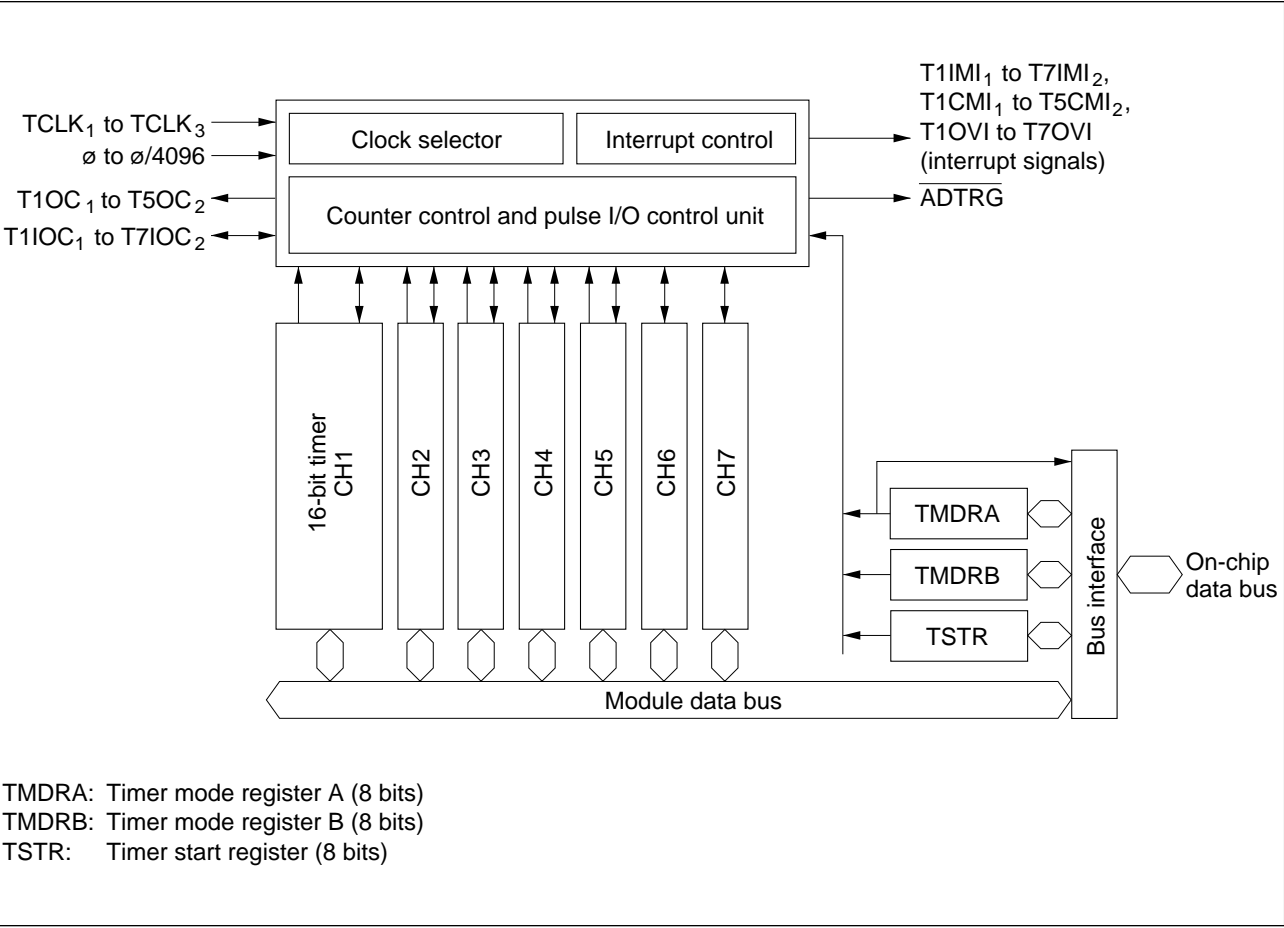
- Auto-measure function

Two timer channels can be coordinated for automatic measurement of pulse width or period and for two-phase encoder counting.

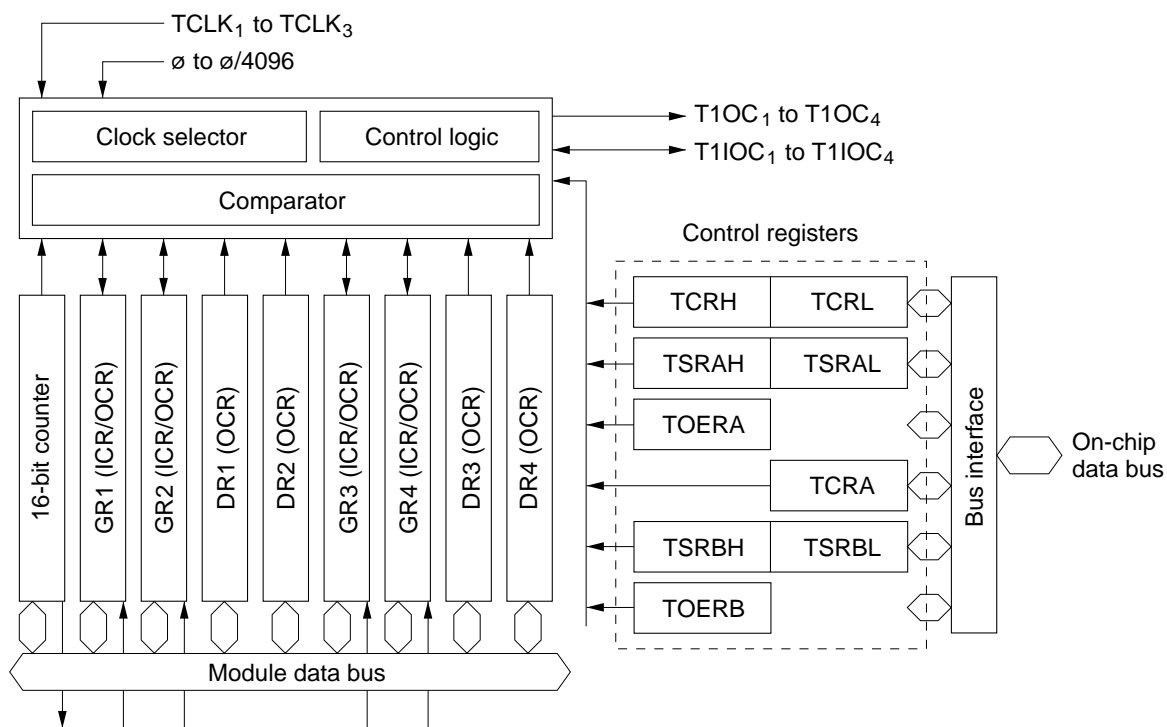
- Thirty-five interrupt sources

These include 16 compare match/input capture interrupts, 12 compare match interrupts, and 7 overflow interrupts: total 35 sources. The compare match/input capture interrupts and overflow interrupts are independently vectored. The compare match interrupts have one interrupt vector per two interrupt sources. The compare match/input capture interrupts and compare match interrupts can start the data transfer controller (DTC) to transfer data.

6.3.3 Block Diagrams

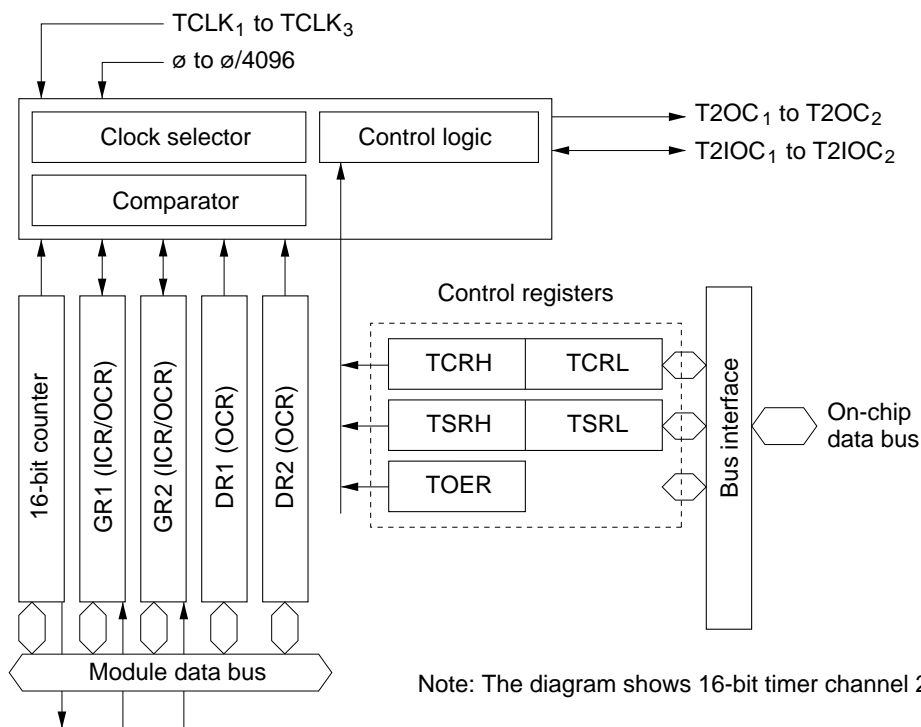


Overall IPU Block Diagram



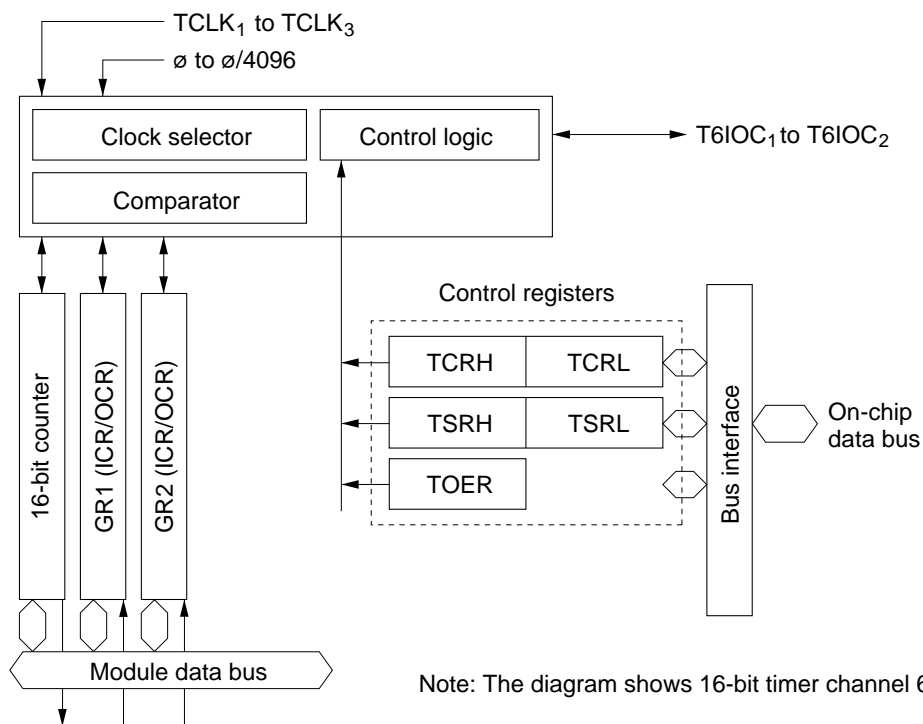
GR1 to GR4: Output compare/input capture registers (16 bits \times 4)
 DR1 to DR4: Output compare registers (16 bits \times 4)
 TCRH and TCRL: Timer control registers (8 bits \times 2)
 TSRAH and TSRAL: Timer status register A (8 bits \times 2)
 TCRA: Timer control register A (8 bits)
 TSRBH and TSRBL: Timer status register B (8 bits \times 2)
 TOERA: Timer output enable register A (8 bits)
 TOERB: Timer output enable register B (8 bits)

Block Diagram of 16-Bit Timer Channel 1



GR1 and GR2: Output compare/input capture registers (16 bits × 4)
 DR1 and DR2: Output compare registers (16 bits × 4)
 TCRH and TCRL: Timer control registers (8 bits × 2)
 TSRH and TSRL: Timer status registers (8 bits × 2)
 TOER: Timer output enable register (8 bits)

Block Diagram of 16-Bit Timer Channels 2 to 5



GR1 and GR2: Output compare/input capture registers (16 bits × 4)
 TCRH and TCRL: Timer control registers (8 bits × 2)
 TSRH and TSRL: Timer status registers (8 bits × 2)
 TOER: Timer output enable register (8 bits)

Block Diagram of 16-Bit Timer Channels 6 and 7

6.3.4 Interrupt Sources and Data Transfer Controller Activation

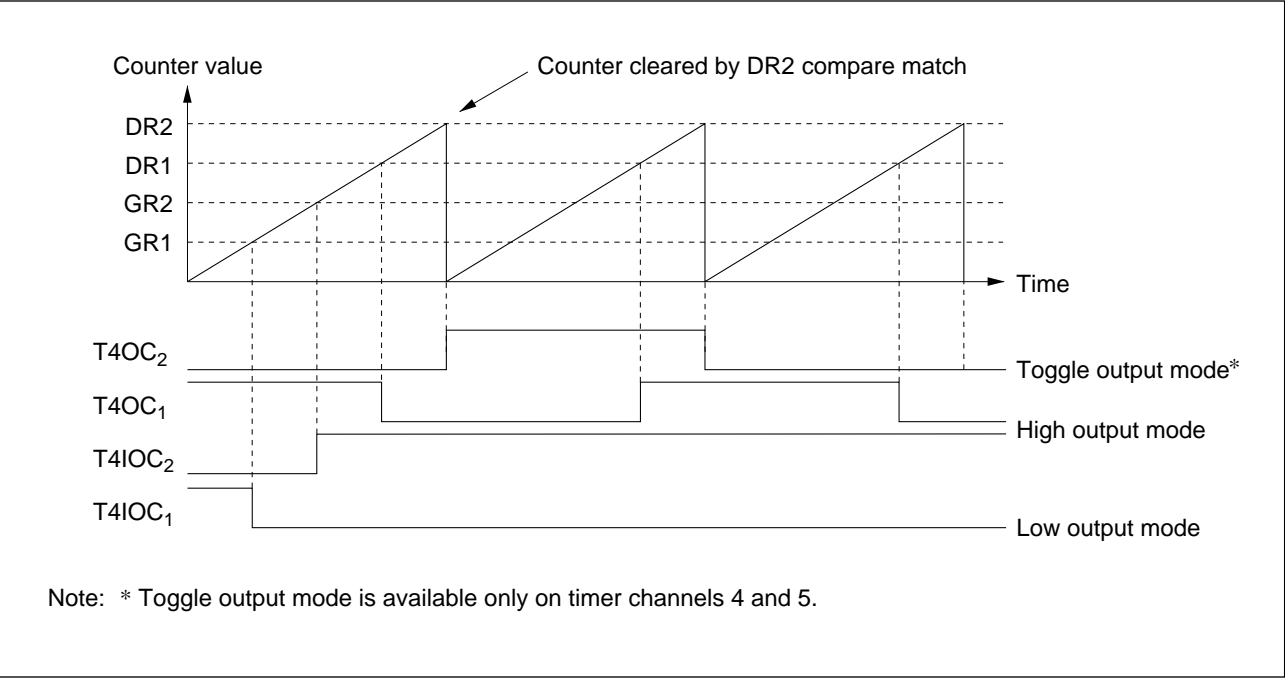
Of the IPU’s 35 interrupt sources, the compare match interrupts and compare match/input capture interrupts can start the data transfer controller (DTC) to transfer data.

Channel	Interrupt Source	Description	DTC Available?	Priority Order
CH1	T1IMI1	GR1 compare match or input capture	Yes	High ↑
	T1IMI2	GR2 compare match or input capture	Yes	
	T1CMI1/T1CMI2	DR1 or DR2 compare match	Yes	
	T1OVI	Timer counter 1 overflow	No	
	T1IMI3	GR3 compare match or input capture	Yes	
	T1IMI4	GR4 compare match or input capture	Yes	
	T1CMI3/T1CMI4	DR3 or DR4 compare match	Yes	
CH2	T2IMI1	GR1 compare match or input capture	Yes	↓
	T2IMI2	GR2 compare match or input capture	Yes	
	T2CMI1/T2CMI2	DR1 or DR2 compare match	Yes	
	T2OVI	Timer counter 2 overflow	No	
CH3	T3IMI1	GR1 compare match or input capture	Yes	
	T3IMI2	GR2 compare match or input capture	Yes	
	T3CMI1/T3CMI2	DR1 or DR2 compare match	Yes	
	T3OVI	Timer counter 3 overflow	No	
CH4	T4IMI1	GR1 compare match or input capture	Yes	
	T4IMI2	GR2 compare match or input capture	Yes	
	T4CMI1/T4CMI2	DR1 or DR2 compare match	Yes	
	T4OVI	Timer counter 4 overflow	No	
CH5	T5IMI1	GR1 compare match or input capture	Yes	
	T5IMI2	GR2 compare match or input capture	Yes	
	T5CMI1/T5CMI2	DR1 or DR2 compare match	Yes	
	T5OVI	Timer counter 5 overflow	No	
CH6	T6IMI1	GR1 compare match or input capture	Yes	
	T6IMI2	GR2 compare match or input capture	Yes	
	T6OVI	Timer counter 6 overflow	No	
CH7	T7IMI1	GR1 compare match or input capture	Yes	
	T7IMI2	GR2 compare match or input capture	Yes	
	T7OVI	Timer counter 7 overflow	No	Low

6.3.5 Examples of Pulse Output

One-Shot Output Mode: The output goes high or low at compare match.

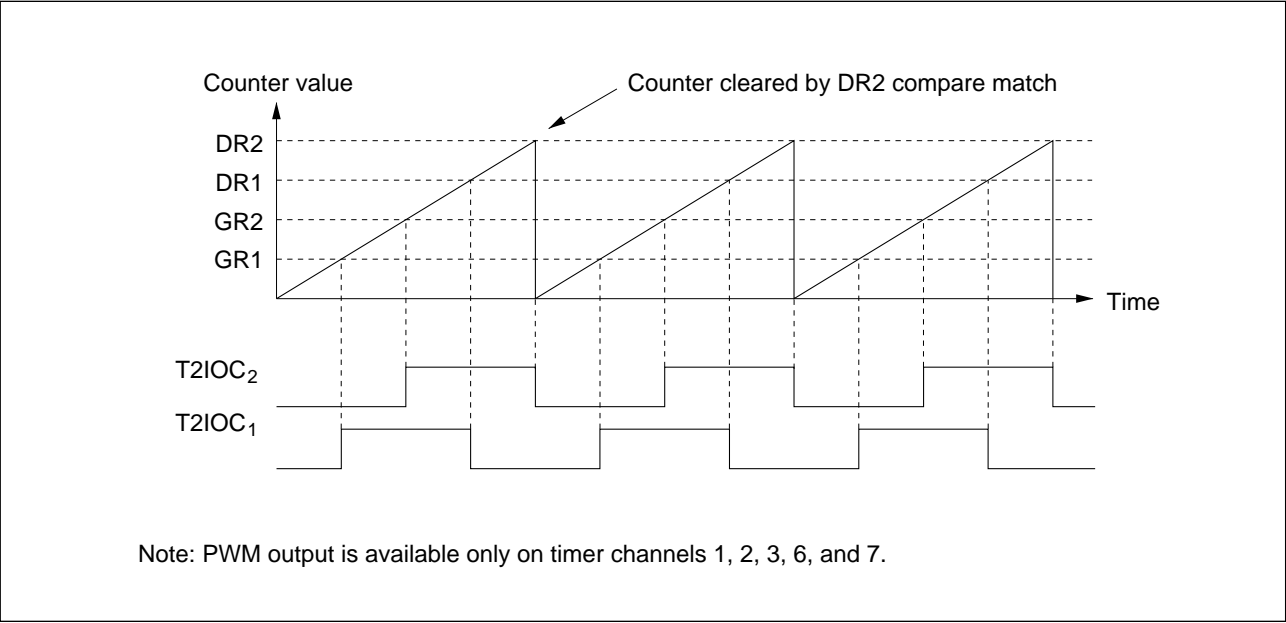
- Example of four-phase pulse output on timer channel 4



Timer Channel No.	CH1	CH2	CH3	CH4	CH5	CH6	CH7
High or low output	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Toggle output	—	—	—	Yes	Yes	—	—

PWM Output Mode: The output inverts at compare match with two registers. One- to three-phase PWM waveforms with independent periods and duty cycles can be generated. Maximum nine-phase PWM output can be obtained by using the synchronizing function.

- Example of two-phase PWM output on timer channel 2

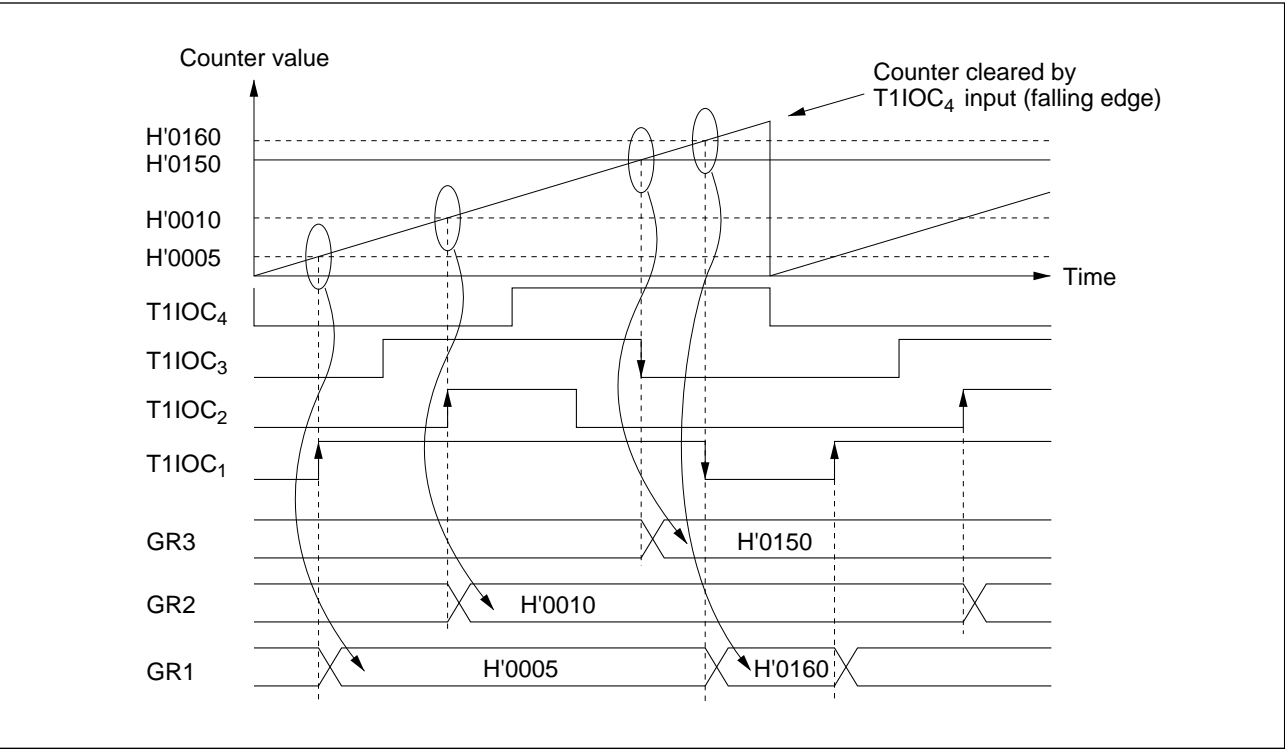


Timer Channel No.	CH1	CH2	CH3	CH4	CH5	CH6	CH7
PWM output	Yes	Yes	Yes	—	—	Yes	Yes
Number of phases output	3	2	2	—	—	1	1

6.3.6 Examples of Pulse Measurement Functions

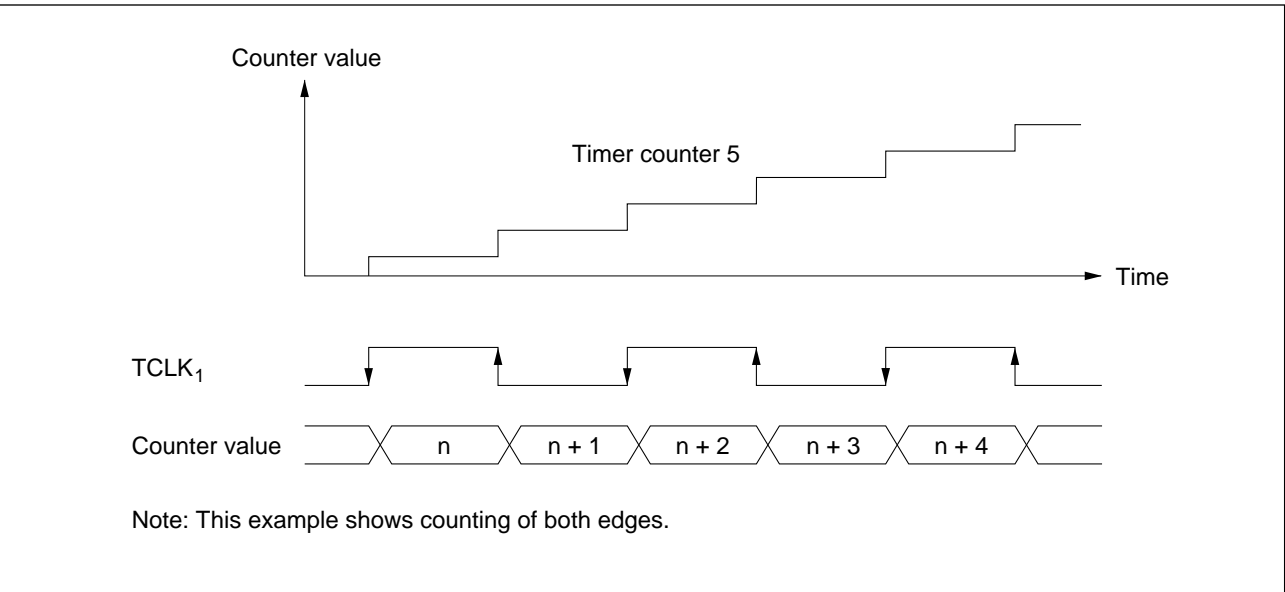
Pulse Period Measurement: Pulse period can be measured by input of an external pulse signal to an input capture pin. The rising edge, falling edge, or both edges can be captured.

- Example of pulse period measurement on timer channel 1



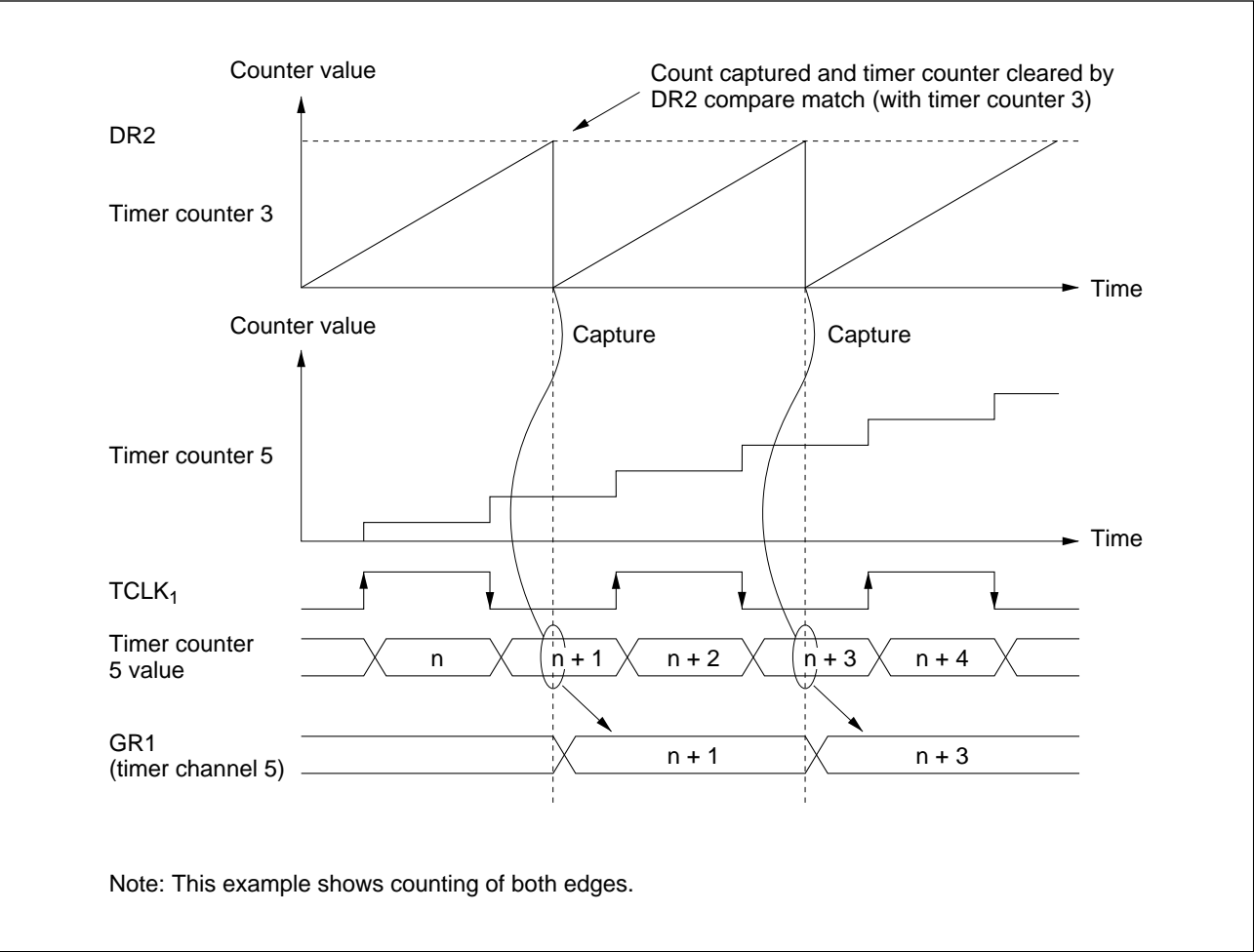
Event Counting: Events can be counted by input of external event signals at pins TCLK1 to TCLK3. Rising edges, falling edges, or both edges can be counted.

- Example of external event counting on timer channel 5



Programmable-Period Counting Mode: An event count is automatically captured when compare match occurs on another channel.

- Example of automatic external event counting: An event count on timer channel 5 is captured to GR1 in timer channel 5 when DR2 compare match occurs in timer channel 2 (i.e. at fixed intervals).

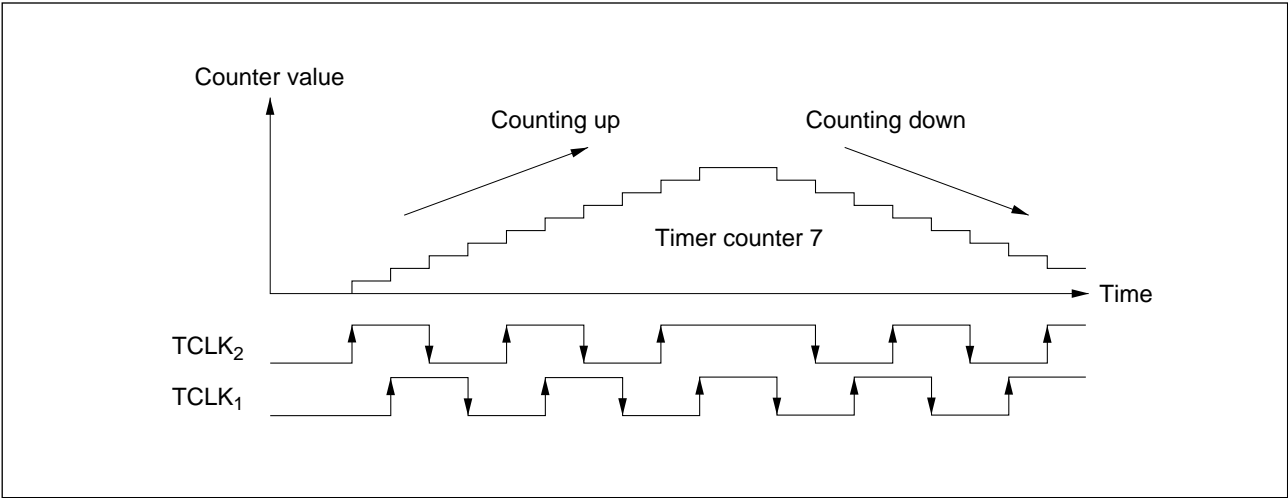


There are four programmable-period counting modes as listed below.

	Compare Match Channel		Capture Channel	
	Channel No.	Register	Channel No.	Register
MD2-6	Channel 2	DR2	Channel 6	GR1
MD3-5	Channel 3	DR2	Channel 5	GR1
MD4-7	Channel 4	DR2	Channel 7	GR1
MD6-7	Channel 6	GR2	Channel 7	GR2

Two-Phase Encoder Counting: Pulse outputs from a two-phase encoder attached to a motor can be input at pins $TCLK_1$ and $TCLK_2$. The timer counter operates as an up-down counter controlled by the phase of the input. This function is available only on channel 7.

- Example of two-phase encoder counting by timer channel 7



All edges are counted.

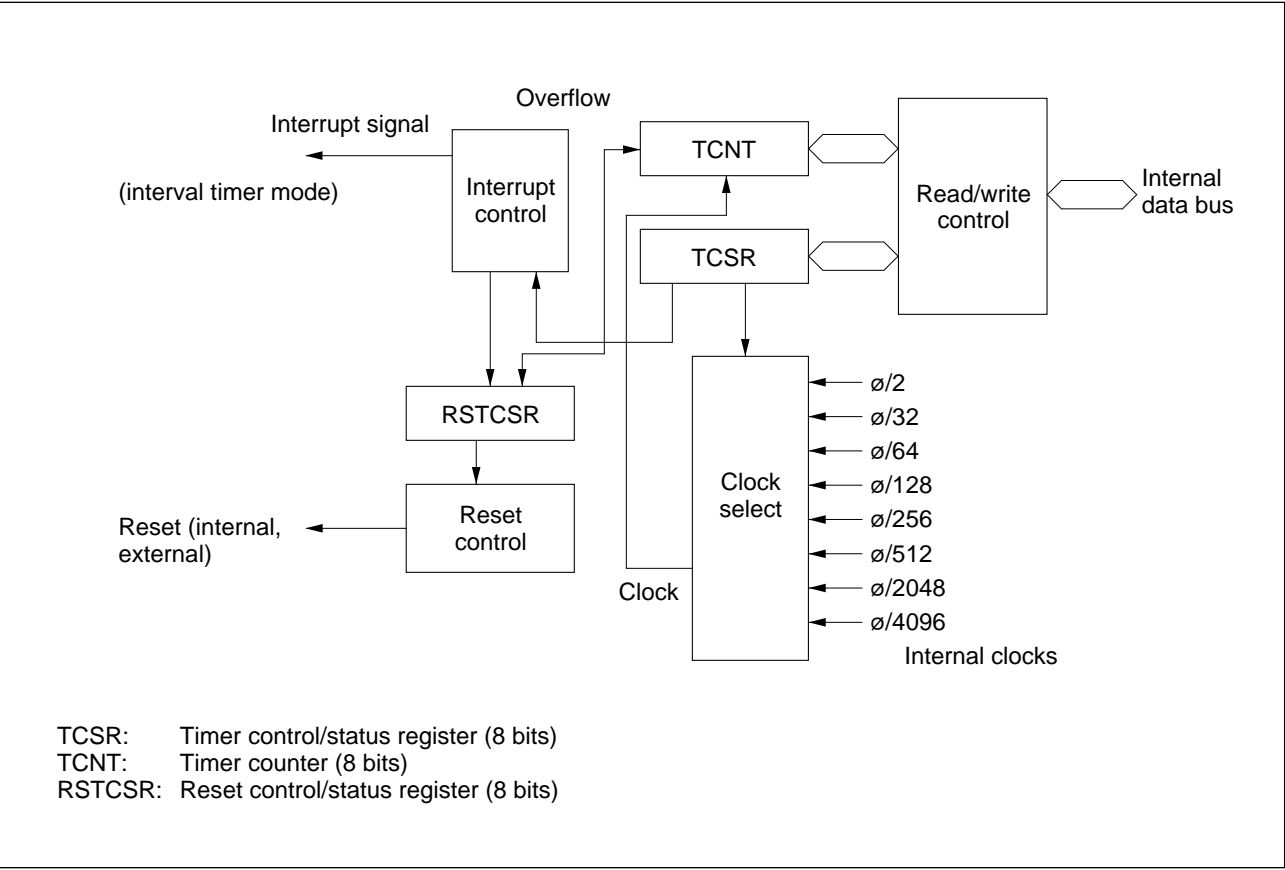
Counting Direction	Up				Down			
$TCLK_2$		High		Low		Low		High
$TCLK_1$	Low		High		High		Low	

6.4 Watchdog Timer

6.4.1 Functions

The H8/538F has an on-chip watchdog timer (WDT) for system supervision. If due to a system or other error the timer counter is allowed to overflow before its value is rewritten, a CPU reset signal is generated. A reset signal can also be output to external devices from the RESO pin. When not needed as a watchdog timer, this timer can be used as an interval timer, in which case an interval timer interrupt request is generated when the counter overflows. The WDT is also used in recovering from software standby.

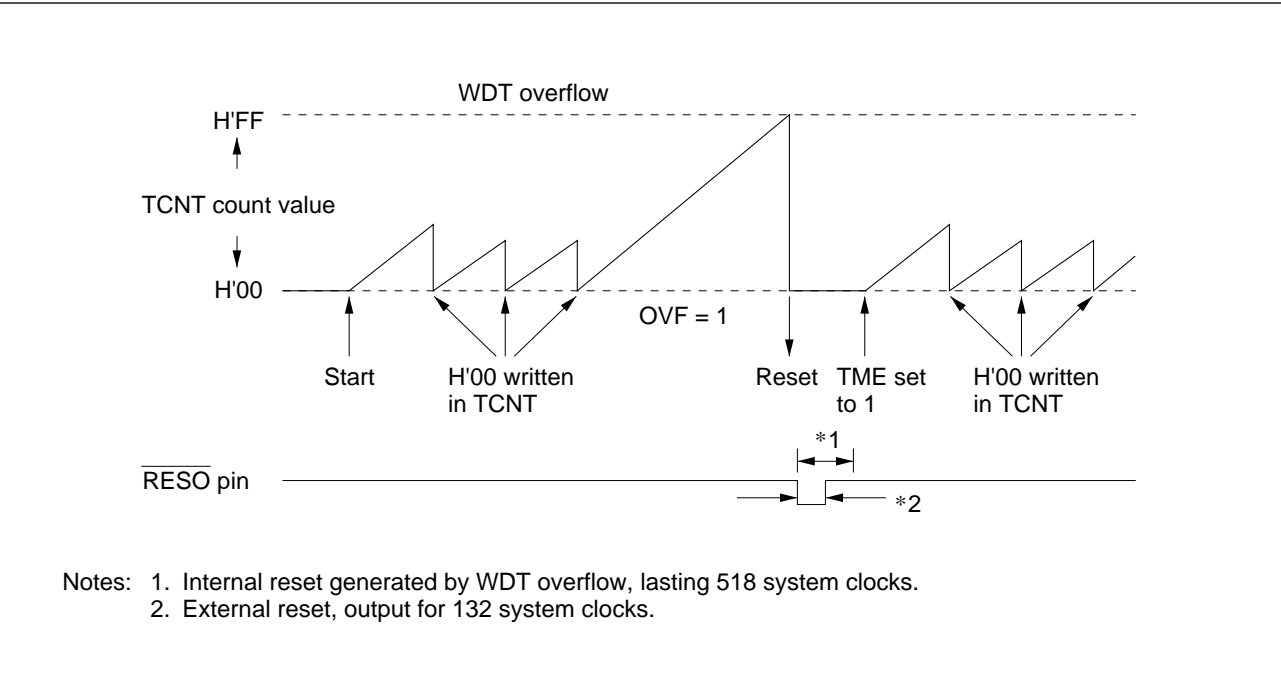
6.4.2 Block Diagram



Block Diagram of Watchdog Timer

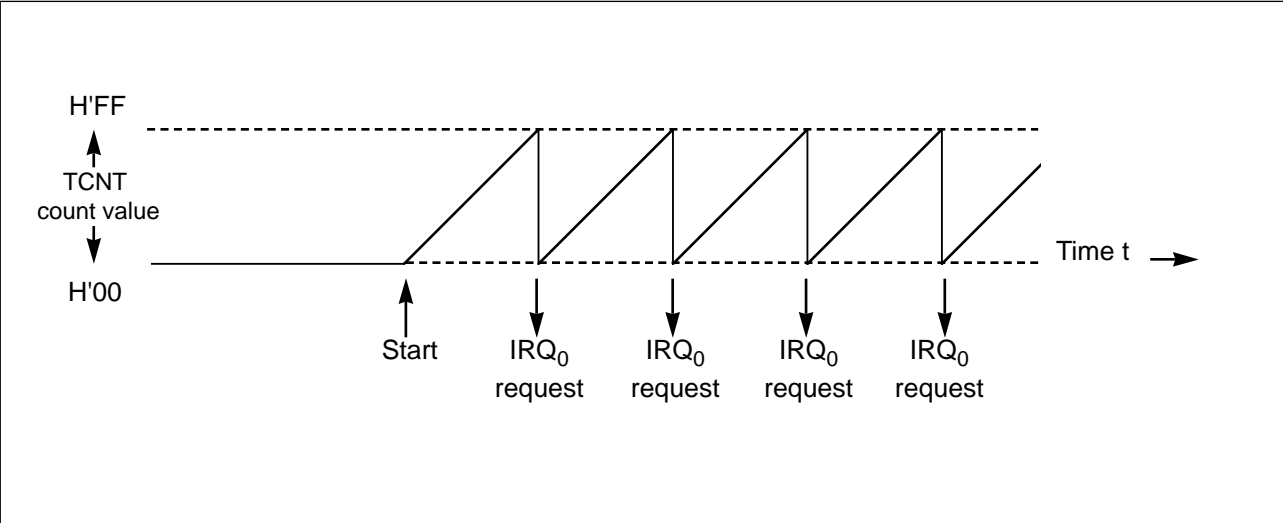
6.4.3 Watchdog Timer Operation

When set to watchdog timer mode, this timer begins incrementing TCNT at the selected clock rate. Software should rewrite the TCNT value before an overflow occurs. If not rewritten due to a system or other error, TCNT will overflow and generate a reset lasting 518 system clock cycles. A reset signal can also be output to external devices. External reset output lasts for 132 system clock cycles; note that this is different from the internal reset duration. An internal watchdog timer reset has the same vector as a reset from the $\overline{\text{RES}}$ pin.



6.4.4 Interval Timer Operation

When set to interval timer mode, this timer begins incrementing TCNT at the selected clock rate. Each time TCNT overflows, an IRQ_0 interrupt is requested. This function can be used to generate interrupts at regular intervals.



6.4.5 Overflow Period

Internal Clock	Overflow Period
$\phi/2$	51.2 μ s
$\phi/32$	819.2 μ s
$\phi/64$	1.6 ms
$\phi/128$	3.3 ms
$\phi/256$	6.6 ms
$\phi/512$	13.1 ms
$\phi/2048$	52.4 ms
$\phi/4096$	104.9 ms

Note: Values shown are for a 10-MHz system clock (ϕ).

6.5 Serial Communication Interface

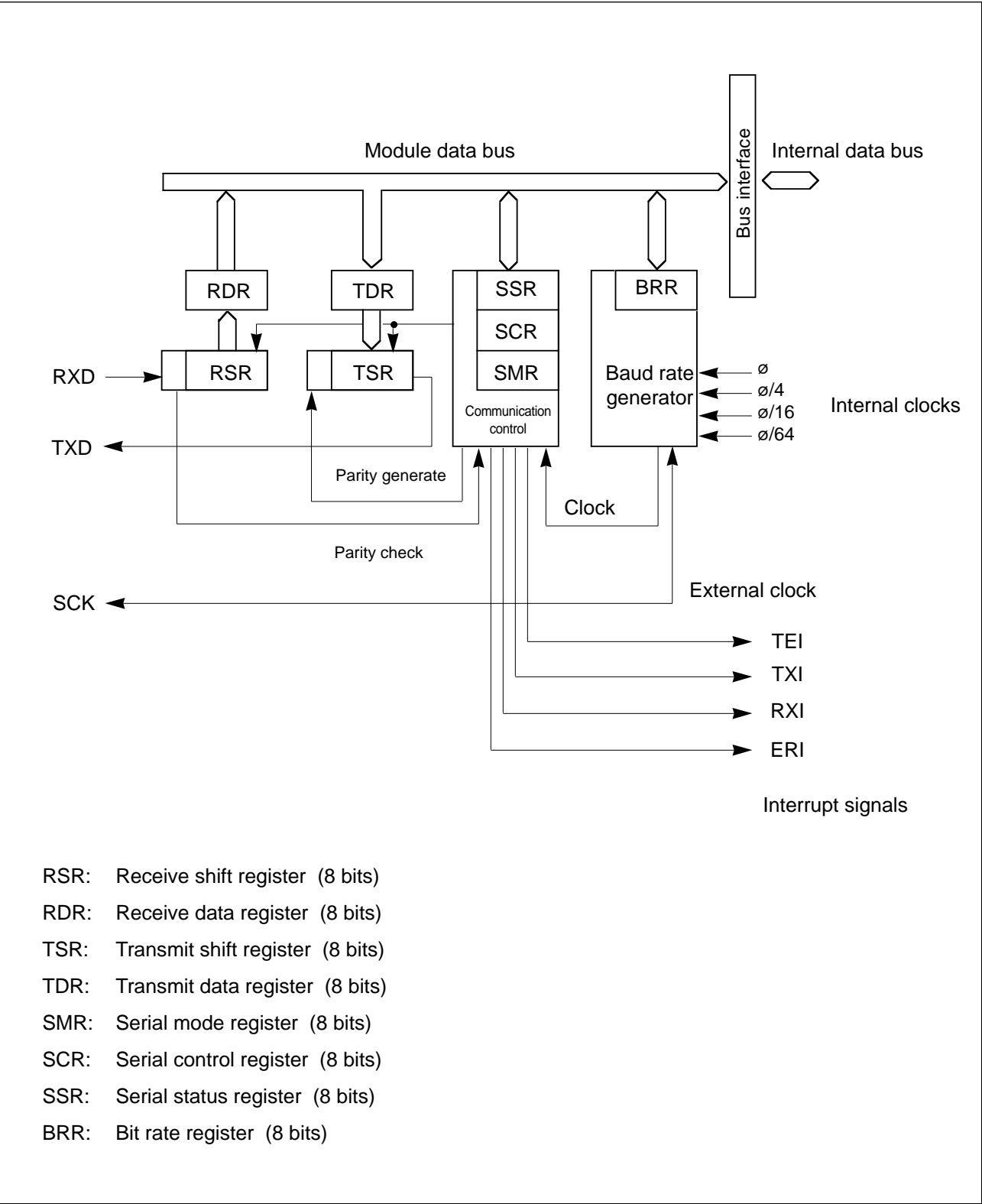
6.5.1 Functions

The H8/538F has an on-chip serial communication interface (SCI) with two independent channels. The SCI is an I/O module that communicates with external devices by synchronous or asynchronous serial data transfer.

6.5.2 Features

- Supports both synchronous and asynchronous communication.
- Supports full duplex communication.
- Can send and receive data continuously, using double-buffering data registers.
- Built-in baud rate generator can generate any bit rate.
- Selectable clock source: either the built-in baud rate generator or an external clock (SCK pin).
- A serial clock can be output from the SCK pin.
- Detects overrun errors, framing errors, and parity errors.
- Line break can be detected.
- Four interrupt sources: transmit-end, transmit-data-empty, receive-data-full, and receive-error. The transmit-data-empty and receive-data-full interrupts can start the data transfer controller.
- Both channels operate independently.
- The least significant data bit (LSB) is transmitted and received first.

6.5.3 Block Diagram



Block Diagram of Serial Communication Interface

6.5.4 Asynchronous Mode

In asynchronous communication mode individual characters are synchronized by start and stop bits.

- Twelve transfer formats
 - Data length: 7 or 8 bits
 - Stop bit length: 1 or 2 bits
 - Parity bit: even, odd, or no parity
 - Multiprocessor bit: can be added for communication among multiple processors
- Either the internal baud rate generator or an external clock can be selected as the serial clock source.
- The serial clock can be output at the SCK pin.
- Three types of receive errors can be detected:
 - Overrun error: attempt to move next RSR data into RDR before RDR has been read
 - Parity error: received parity bit does not match selected parity
 - Framing error: stop bit not detected at end of receive data
- Line break can be detected by reading the state of the RXD pin when a framing error occurs.
- Serial data can be transmitted among multiple processors by using a multiprocessor format.

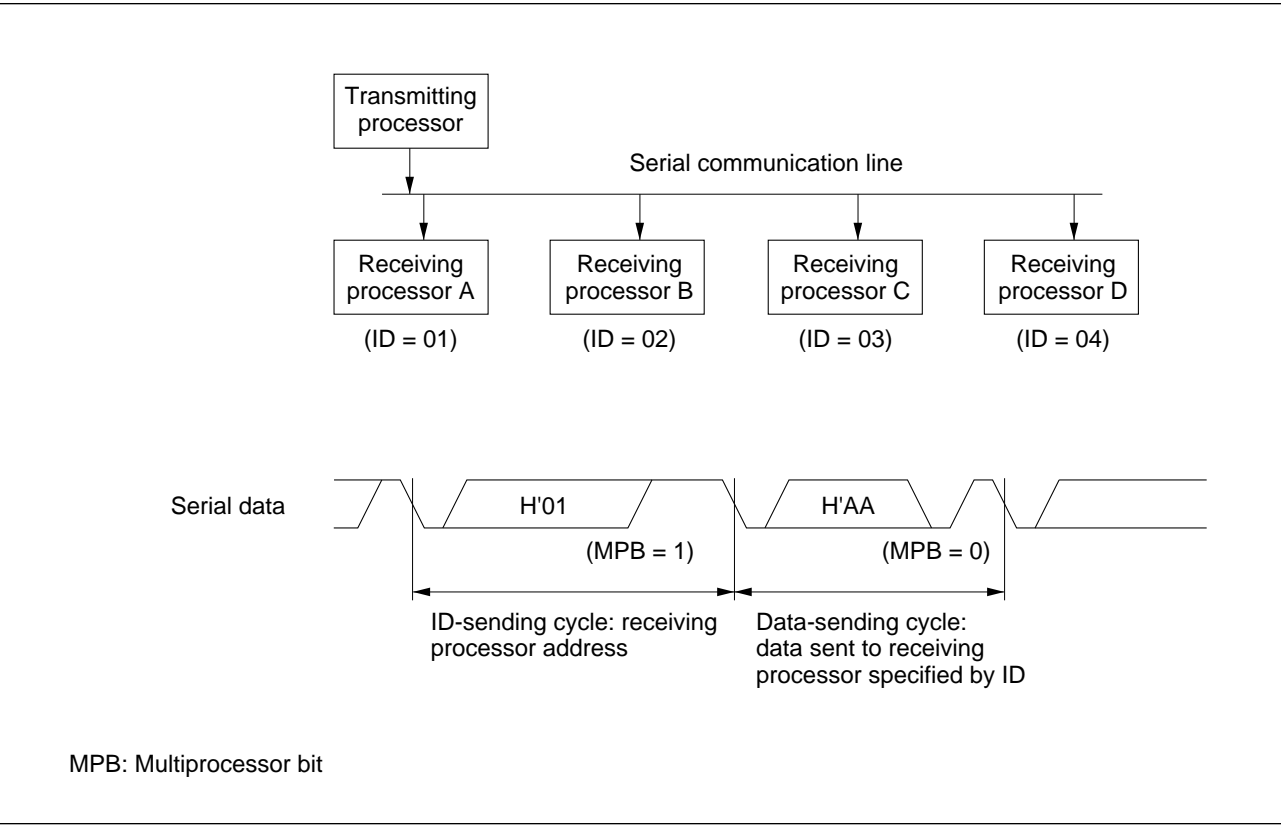
Start	7-bit data	Stop (1)		
Start	7-bit data	Stop (2)		
Start	7-bit data	Parity	Stop (1)	
Start	7-bit data	Parity	Stop (2)	
Start	7-bit data	MPB	Stop (1)	
Start	7-bit data	MPB	Stop (2)	
Start	8-bit data		Stop (1)	
Start	8-bit data		Stop (2)	
Start	8-bit data		Parity	Stop (1)
Start	8-bit data		Parity	Stop (2)
Start	8-bit data	MPB	Stop (1)	
Start	8-bit data	MPB	Stop (2)	

Twelve Data Formats

6.5.5 Multiprocessor Communication

Multiple processors can transmit and receive serial data by using a format with an additional multiprocessor bit.

- The transmitting processor starts by sending the receiving processor’s ID with the multiprocessor bit (MPB) set to 1. Next, it sends transmit data with the multiprocessor bit cleared to 0.
- Receiving processors skip incoming data until they receive data with the multiprocessor bit set to 1. When they receive data with MPB = 1, they compare the data with their IDs.
- The receiving processor with a matching ID continues to receive incoming data. Processors with non-matching IDs skip further incoming data until they again receive data with MPB = 1.

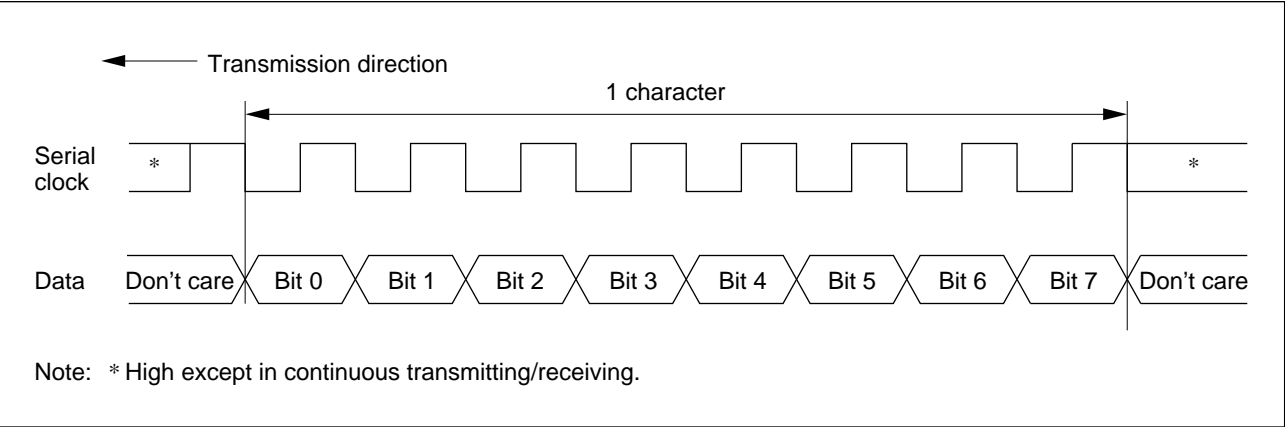


**Example of Communication among Processors using Multiprocessor Format
(Sending Data H'AA to Receiving Processor A)**

6.5.6 Synchronous Mode

In synchronous mode data transfer is synchronized with a clock pulse. This mode is suitable for continuous, high-speed serial communication.

- Data length is 8 bits/character.
- Overrun errors are detected.
- The on-chip baud rate generator or an external clock can be selected as the serial clock source. If the on-chip baud rate generator is selected, eight clock pulses are output automatically from the SCK pin.



Data Format in Synchronous Mode (Example)

6.5.7 Interrupts and Data Transfer Controller Activation

There are four types of SCI interrupts: receive-error (overrun error, framing error, or parity error), transmit-end, transmit-data-empty, and receive-data-full. The data transfer controller can be started by the transmit-data-empty and receive-data-full interrupts. Continuous transmitting or receiving can be implemented without software overhead by using the data transfer controller to write the next transmit data when a transmit-data-empty interrupt occurs, or read the RDR data when a receive-data-full interrupt occurs.

SCI Interrupts

Interrupt	Description	DTC Available?	Priority
ERI	Receive-error interrupt	No	High ↑ ↓ Low
RXI	Receive-data-full interrupt	Yes	
TXI	Transmit-data-empty interrupt	Yes	
TEI	Transmit-end interrupt	No	Low

6.5.8 BRR Settings for Typical Bit Rates (Asynchronous Mode)

Bit Rate (bit/s)	ø (MHz)								
	9.8304			10			16		
	Baud Rate Generator Input Clock	BRR Value	Error (%)	Baud Rate Generator Input Clock	BRR Value	Error (%)	Baud Rate Generator Input Clock	BRR Value	Error (%)
110	ø/16	174	−0.26	ø/64	177	−0.25	ø/16	70	+0.03
150	ø/16	127	0	ø/16	129	+0.16	ø/16	207	+0.16
300	ø/4	255	0	ø/16	64	+0.16	ø/16	103	+0.16
600	ø/4	127	0	ø/4	129	+0.16	ø/4	207	+0.16
1200	ø	255	0	ø/4	64	+0.16	ø/4	51	+0.16
2400	ø	127	0	ø	129	+0.16	ø	207	+0.16
4800	ø	63	0	ø	64	+0.16	ø	103	+0.16
9600	ø	31	0	ø	32	−1.36	ø	51	+0.16
19200	ø	15	0	ø	15	+1.73	ø	25	+0.16
31250	ø	9	−1.7	ø	9	0	ø	15	0
38400	ø	7	0	ø	7	+1.73	ø	12	+0.16

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

n	Clock
0	ø
1	ø/4
2	ø/16
3	ø/64

N: BRR value of baud rate generator; 0 ≤ N ≤ 255

ø: System clock frequency (MHz)

B: Bit rate (bit/s)

n: Baud rate generator input clock No.; n = 0, 1, 2, 3

6.5.9 BRR Settings for Typical Bit Rates (Synchronous Mode)

Bit Rate (bit/s)	ø (MHz)					
	8		10		16	
	Baud Rate Generator Input Clock	BRR Value	Baud Rate Generator Input Clock	BRR Value	Baud Rate Generator Input Clock	BRR Value
250	ø/64	124	—	—	ø/64	249
500	ø/16	249	—	—	ø/64	124
1 k	ø/16	124	—	—	ø/16	249
2.5 k	ø/4	199	ø/4	249	ø/16	99
5 k	ø/4	99	ø/4	124	ø/4	199
10 k	ø	199	ø	249	ø/4	99
25 k	ø	79	ø	99	ø	159
50 k	ø	39	ø	49	ø	79
100 k	ø	19	ø	24	ø	39
250 k	ø	7	ø	9	ø	15
500 k	ø	3	ø	4	ø	7
1 M	ø	1	—	—	ø	3
2.5 M			ø	0*	—	—

Note: Blank settings are unavailable. A dash (–) indicates that the setting is possible, but incurs error. An asterisk (*) indicates that continuous transmission/reception is not possible.

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

n	Clock
0	ø
1	ø/4
2	ø/16
3	ø/64

N: BRR value of baud rate generator; 0 ≤ N ≤ 255
ø: System clock frequency (MHz)
B: Bit rate (bits/s)
n: Baud rate generator input clock No.; n = 0, 1, 2, 3

6.6 A/D Converter

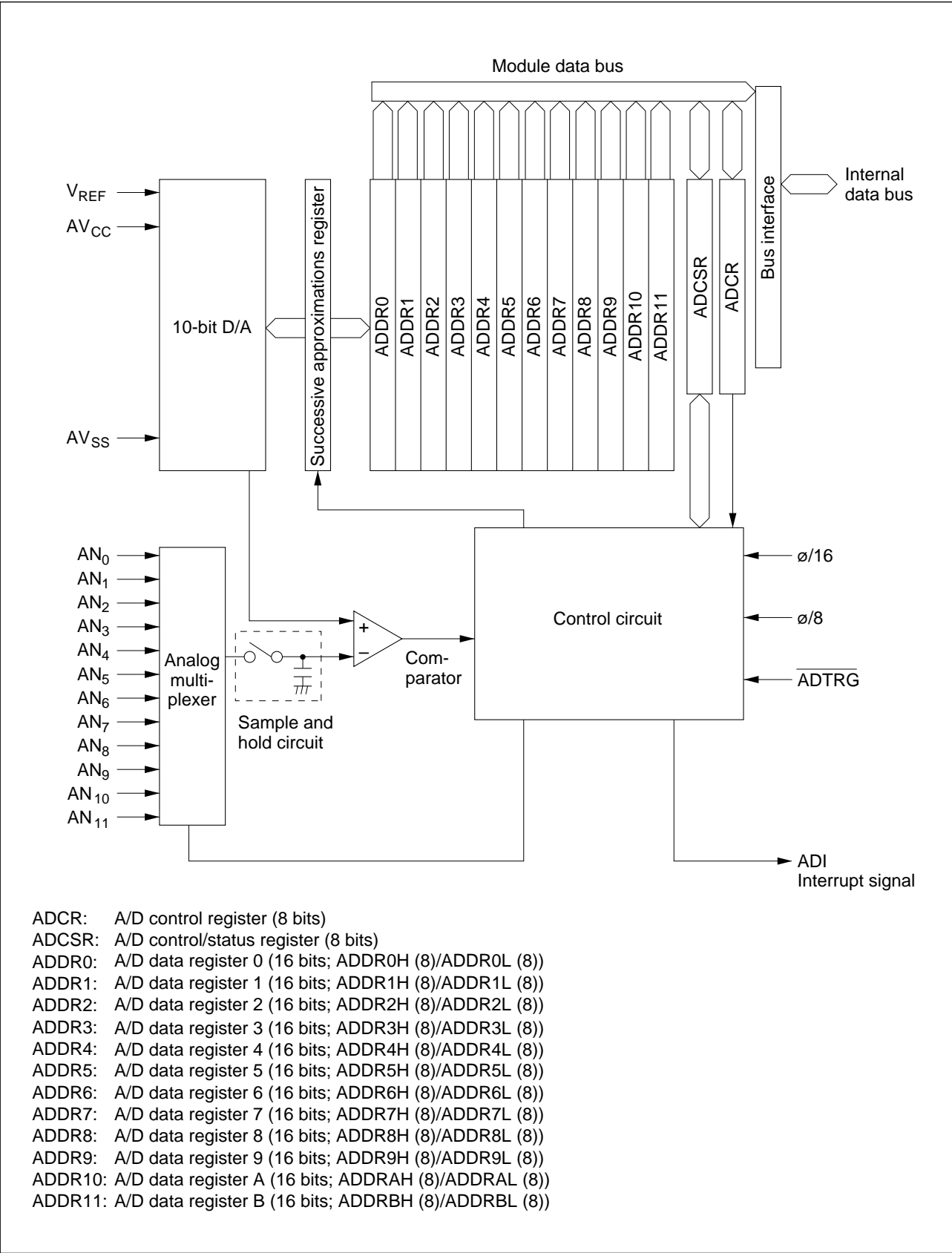
6.6.1 Functions

One of the H8/538F's on-chip supporting modules is a 10-bit analog-to-digital converter which can be programmed for input of up to 12 analog signal channels.

6.6.2 Features

- 10-bit resolution
- Twelve analog input channels
- Fast: minimum conversion time is 8.4 μ s per channel (at 16 MHz)
- Selectable modes
 - Single mode: A/D conversion of one channel
 - Scan mode: Continuous conversion of one to 12 channels
- Built-in sample-and-hold function
- External reference pin
- A/D conversion can be externally triggered
- Twelve 16-bit data registers. A/D-converted results are transferred to data registers corresponding to each channel and held.
- Generates a CPU interrupt (ADI: A/D conversion end) at the completion of A/D conversion. This interrupt can start the data transfer controller to transfer data.

6.6.3 Block Diagram



Block Diagram of A/D Converter

6.6.4 Operation

Analog-to-digital conversion is performed by successive approximations with 10-bit resolution. The channel select bits (CH3 to CH0) in ADCSR can be set to select any of the 12 analog input channels.

				Channel(s) Selected			
Bit 3 CH3	Bit 2 CH2	Bit 1 CH1	Bit 0 CH0	Single Mode	4-Channel Scan Mode	8-Channel Scan Mode	12-Channel Scan Mode
0	0	0	0	AN ₀	AN ₀	AN ₀ , AN ₄	AN ₀ , AN ₄ , AN ₈
			1	AN ₁	AN ₀ , AN ₁	AN ₀ , AN ₁ , AN ₄ , AN ₅	AN ₀ , AN ₁ , AN ₄ , AN ₅ , AN ₈ , AN ₉
0	0	1	0	AN ₂	AN ₀ to AN ₂	AN ₀ to AN ₂ , AN ₄ to AN ₆	AN ₀ to AN ₂ , AN ₄ to AN ₆ , AN ₈ to AN ₁₀
			1	AN ₃	AN ₀ to AN ₃	AN ₀ to AN ₇	AN ₀ to AN ₁₁
0	1	0	0	AN ₄	AN ₄	AN ₀ , AN ₄	AN ₀ , AN ₄ , AN ₈
			1	AN ₅	AN ₄ , AN ₅	AN ₀ , AN ₁ , AN ₄ , AN ₅	AN ₀ , AN ₁ , AN ₄ , AN ₅ , AN ₈ , AN ₉
0	1	1	0	AN ₆	AN ₄ to AN ₆	AN ₀ to AN ₂ , AN ₄ to AN ₆	AN ₀ to AN ₂ , AN ₄ to AN ₆ , AN ₈ to AN ₁₀
			1	AN ₇	AN ₄ to AN ₇	AN ₀ to AN ₇	AN ₀ to AN ₁₁
1	0*1	0	0	AN ₈	AN ₈	Reserved*2	AN ₀ , AN ₄ , AN ₈
			1	AN ₉	AN ₈ , AN ₉	Reserved*2	AN ₀ , AN ₁ , AN ₄ , AN ₅ , AN ₈ , AN ₉
1	0*1	1	0	AN ₁₀	AN ₈ to AN ₁₀	Reserved*2	AN ₀ to AN ₂ , AN ₄ to AN ₆ , AN ₈ to AN ₁₀
			1	AN ₁₁	AN ₈ to AN ₁₁	Reserved*2	AN ₀ to AN ₁₁

Notes: 1. Must be cleared to 0.
2. Reserved for future expansion. Must not be used.

Single Mode: In this mode A/D conversion is performed on one channel under CPU control. Conversion starts when the ADST bit in ADCSR is set. When conversion is completed, the A/D conversion end flag (ADF) is set. If the interrupt enable bit (ADIE) is also set, an A/D conversion end interrupt (ADI) is requested. This interrupt can start the data transfer controller.

Scan Mode: This mode can be used to monitor analog inputs on one channel, or a group of up to twelve channels. When the ADST bit is set to 1, A/D conversion starts on the first channel. As soon as conversion of the first channel ends, conversion of the second channel begins (if only one channel is selected, conversion of the first channel begins again). A/D conversion continues in this way until the ADST bit is cleared. The converted results are transferred to and held in the data registers (ADDR0 to ADDR11) corresponding to each channel. Four-channel, eight-channel, and twelve-channel scan modes can be selected by bits ADM0 and ADM1 in ADCSR.

ADM1	ADM0	Scan Mode
0	0	Single mode
0	1	Four-channel scan mode (channels 0 to 3, 4 to 7, or 8 to 11)
1	0	Eight-channel scan mode (channels 0 to 7)
1	1	Twelve-channel scan mode (channels 0 to 11)

6.7 I/O Ports

The H8/538F has twelve I/O ports. Ports 1, 2, 4, 5, 7, B, and C are 8-bit input/output ports. Port 3 is a 6-bit input/output port. Port 6 is a 5-bit input/output port. Port A is a 7-bit input/output port. Port 8 is a 4-bit input port. Port 9 is an 8-bit input port. The port functions are listed in the table below.

These ports are multiplexed with inputs and outputs of the on-chip supporting modules. The functions of ports 1, 2, A, B, and C also differ depending on the operating mode.

Each port has a data direction register (DDR) for selecting input or output, and a data register (DR) for holding output data. In addition to DR and DDR, port A has a bus release control register (BRCR), and ports B and C have pull-up transistor control registers (PBPCR and PCPCR).

Port	Description	Pins	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7
Port 1	8-bit input/ output port	P1 ₇ to P1 ₀ / D ₁₅ to D ₈		Data bus (D ₁₅ to D ₈)			General- purpose input/output
Port 2	8-bit input/ output port	P2 ₇ to P2 ₀ / D ₇ to D ₀	Data bus (D ₇ to D ₀)	General- purpose input/output	Data bus (D ₇ to D ₀)	Data bus (D ₇ to D ₀)	General- purpose input/output
Port 3	6-bit input/ output port	P3 ₅ to P3 ₀ / T2OC ₂ , T2OC ₁ , T1OC ₄ to T1OC ₁	Output (T1OC ₄ to T1OC ₁ , T2OC ₂ , T2OC ₁) from 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output				
Port 4	8-bit input/ output port	P4 ₇ /T7IOC ₂ P4 ₆ /T7IOC ₁ P4 ₅ /T6IOC ₂ P4 ₄ /T6IOC ₁ P4 ₃ /T5IOC ₂ P4 ₂ /T5IOC ₁ P4 ₁ /T4IOC ₂ P4 ₀ /T4IOC ₁	Input and output (T7IOC ₂ , T7IOC ₁ , T6IOC ₂ , T6IOC ₁ , T5IOC ₂ , T5IOC ₁ , T4IOC ₂ , T4IOC ₁) for 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output				
Port 5	8-bit input/ output port	P5 ₇ to P5 ₀ / T3IOC ₂ , T3IOC ₁ , T2IOC ₂ , T2IOC ₁ , T1IOC ₄ to T1IOC ₁	Input and output (T3IOC ₂ , T3IOC ₁ , T2IOC ₂ , T2IOC ₁ , T1IOC ₄ to T1IOC ₁) for 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output				
Port 6	5-bit input/ output port	P6 ₄ /TCLK ₃ P6 ₃ /TCLK ₂ P6 ₂ /TCLK ₁ P6 ₁ /IRQ ₃ P6 ₀ /IRQ ₂	Clock input (TCLK ₃ to TCLK ₁) for 16-bit integrated-timer pulse unit (IPU), external interrupt input (IRQ ₃ , IRQ ₂), and general-purpose input/output				
Port 7	8-bit input/ output port	P7 ₇ /SCK ₂ P7 ₆ /SCK ₁ P7 ₅ /RXD ₂ P7 ₄ /TXD ₂ P7 ₃ /RXD ₁ P7 ₂ /TXD ₁ P7 ₁ /IRQ ₁ /ADTRG P7 ₀ /IRQ ₀	Input and output (SCK ₂ , SCK ₁ , TXD ₂ , TXD ₁ , RXD ₂ , RXD ₁) for serial communication interfaces 1 and 2 (SCI1, SCI2), external interrupt input (IRQ ₁ , IRQ ₀), A/D converter trigger input (ADTRG), and general-purpose input/output				

Port	Description	Pins	Modes 1 and 6	Mode 2	Modes 3 and 5	Mode 4	Mode 7
Port 8	4-bit input port	P8 ₃ to P8 ₀ / AN ₁₁ to AN ₈	Analog input for A/D converter (AN ₁₁ to AN ₈) and general-purpose input				
Port 9	8-bit input port	P9 ₇ to P9 ₀ / AN ₇ to AN ₀	Analog input for A/D converter (AN ₇ to AN ₀) and general-purpose input				
Port A	7-bit input/ output port	PA ₆ /T3OC ₂ / <u>BACK</u> PA ₅ / <u>T3OC₁/BREQ</u> PA ₄ / <u>WAIT</u>	Output from 16-bit integrated-timer pulse unit (IPU), general-purpose input/output, and <u>BACK</u> , <u>BREQ</u> , and <u>WAIT</u> input and output if enabled by settings in bus release control register (BRCR) and wait control register (WCR)				16-bit integrated-timer pulse unit (IPU) output, and general-purpose input/output (PA4: general-purpose input/output only)
		PA ₃ /A ₁₉ /T5OC ₂ PA ₂ /A ₁₈ /T5OC ₁ PA ₁ /A ₁₇ /T4OC ₂ PA ₀ /A ₁₆ /T4OC ₁	Output (T5OC ₂ , T5OC ₁ , T4OC ₂ , T4OC ₁) from 16-bit integrated-timer pulse unit (IPU), and general-purpose input/output	Page address output (A ₁₉ to A ₁₆)	Page address output (A ₁₉ to A ₁₆) when DDR = 1, general-purpose input when DDR = 0		
Port B	8-bit input/ output port	PB ₇ to PB ₀ / A ₁₅ to A ₈	Address output (A ₁₅ to A ₀)	Address output (A ₁₅ to A ₀) when DDR = 1, general-purpose input when DDR = 0	Address output (A ₁₅ to A ₀)	Address output (A ₁₅ to A ₀) when DDR = 1, general-purpose input when DDR = 0	General-purpose input/output
Port C	8-bit input/ output port	PC ₇ to PC ₀ / A ₇ to A ₀					

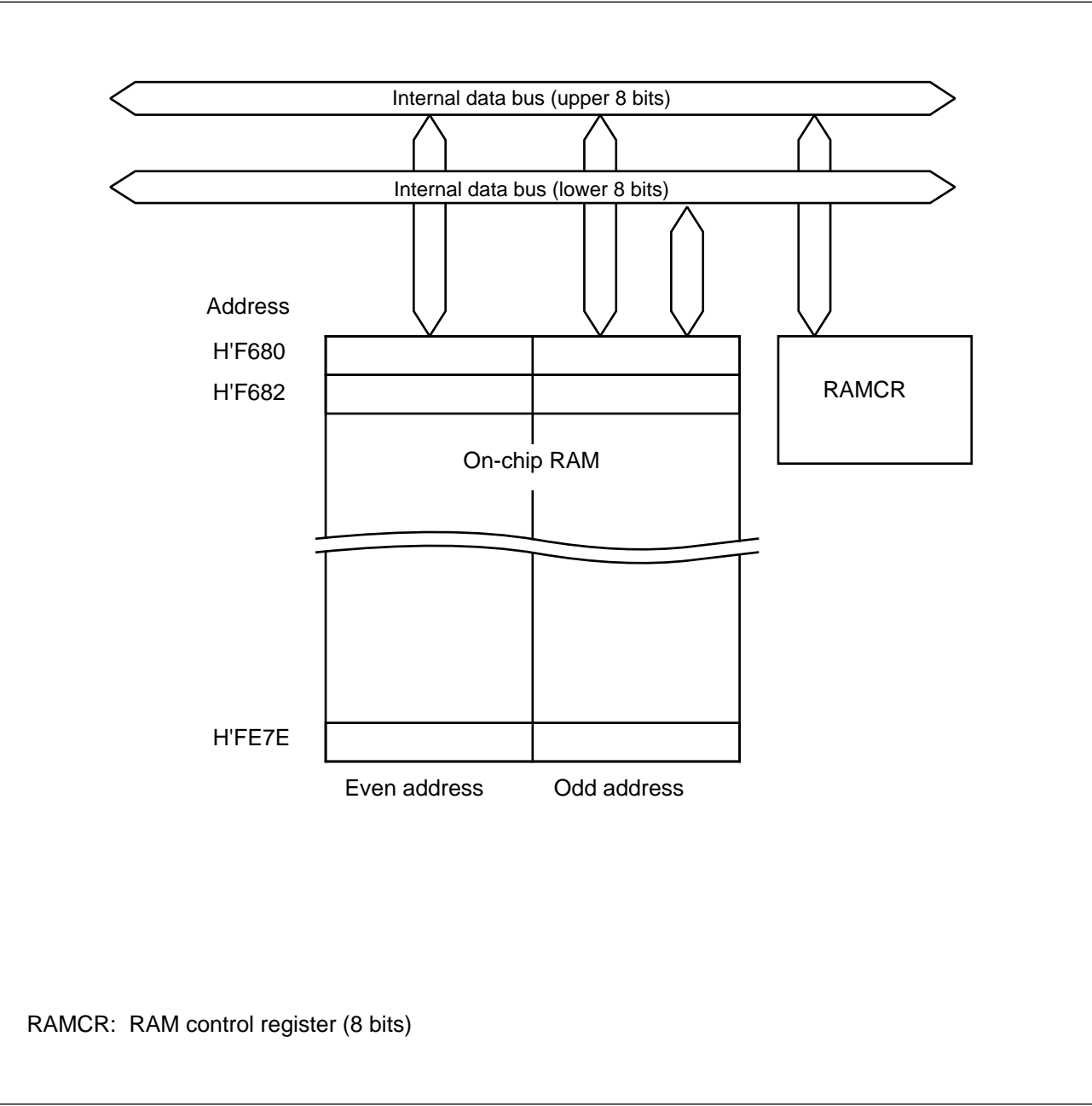
6.8 RAM

6.8.1 Functions

The H8/538F chip includes a 2-kbyte, high-speed, static RAM. This RAM is linked to the CPU by a 16-bit data bus so both byte data and word data are accessed in just two states, permitting high-speed transfer of word data and high-speed operations.

The RAM control register (RAMCR) can select either the on-chip RAM address space or an external address space.

6.8.2 Block Diagram



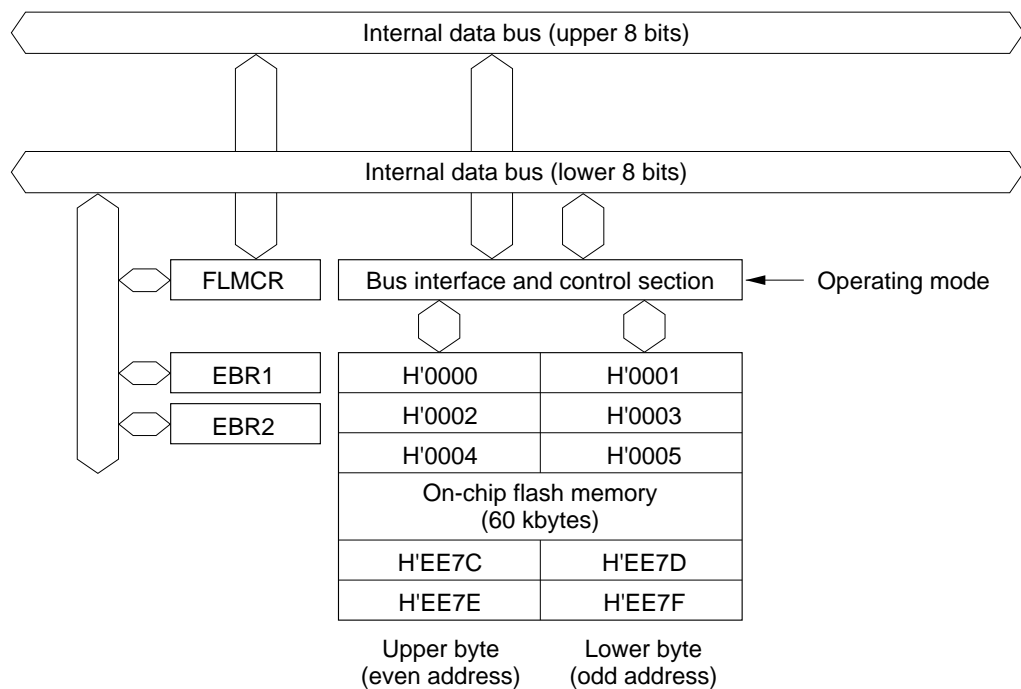
6.9 Flash Memory

The H8/538F has 60 kbytes of on-chip flash memory. The flash memory is linked to the CPU by a 16-bit data bus, permitting both byte data and word data to be accessed in two states. Write, erase, and verify operations are controlled by a flash memory control register (FLMCR) and two erase block registers (EBR1 and EBR2).

6.9.1 Features

- Flash memory capacity: 60 kbytes
- Operating power supply voltage: 5 V $\pm 10\%$; 2.7 V to 5.5 V
- Program-erase cycles: 100
- Programming voltage V_{pp} : 12 V ± 0.6 V (external power supply)
- Program time: 20 μ s/byte (typical)
- Programming methods
 - CPU (software) control
 - HN28F101 flash memory compatible (permitting use of a programmer)
- Erase voltage V_{pp} : 12 V ± 0.6 V (external power supply)
- Extent erased
 - Chip erase
 - Block erase: seven large blocks (total 56 kbytes) and eight small blocks (total 3 kbytes + 640 bytes)
- Erase time: 1 s (typical)
- Erasing methods
 - CPU (software) control
 - HN28F101 flash memory compatible (permitting use of a programmer)

6.9.2 Block Diagram



FLMCR: Flash memory control register (8 bits)
EBR1: Erase block register 1 (8 bits)
EBR2: Erase block register 2 (8 bits)

6.9.3 Register Configuration

The H8/538F’s on-chip flash memory is erased and programmed using two types of control registers: the flash memory control register (FLMCR), and the two erase block registers (EBR1 and EBR2).

Flash Memory Control Register (FLMCR)

7	6	5	4	3	2	1	0
V _{PP}	—	—	—	EV	PV	E	P

- V_{PP}: Flag indicating 12-V input at V_{PP} pin
- EV: Erase verify bit
- PV: Program verify bit
- E: Erase enable bit
- P: Program enable bit

Erase Block Register 1 (EBR1: Addresses H'0000 to H'DFFF)

7	6	5	4	3	2	1	0
—	LB6	LB5	LB4	LB3	LB2	LB1	LB0

These bits designate seven large blocks (total 56 kbytes).

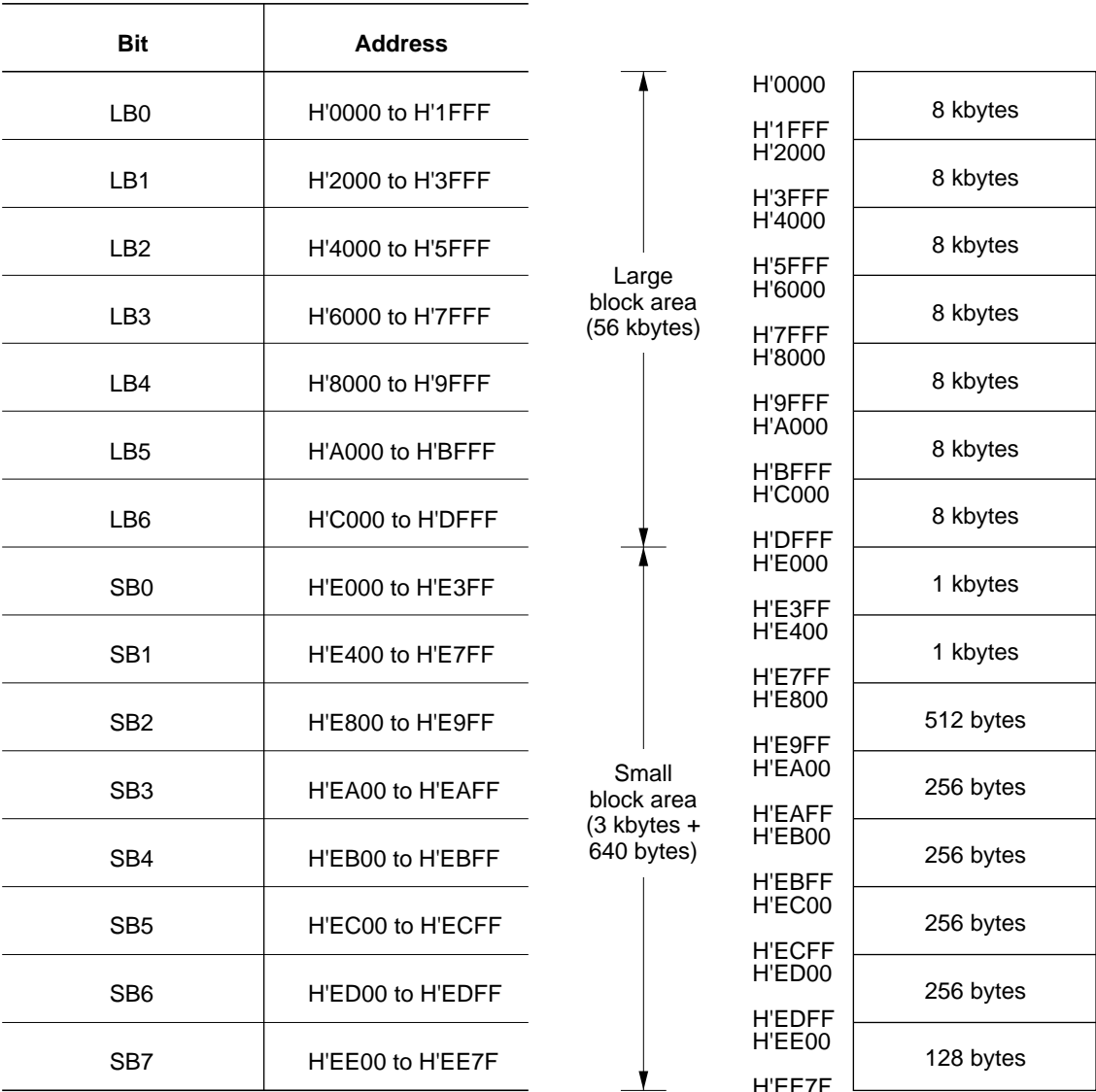
Erase Block Register 2 (EBR2: Addresses H'E000 to H'EE7F)

7	6	5	4	3	2	1	0
SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0

These bits designate eight small blocks (total 3 kbytes + 640 bytes).

A block is erased if the corresponding bit in the erase block register is set to 1.

Erase Block Addresses



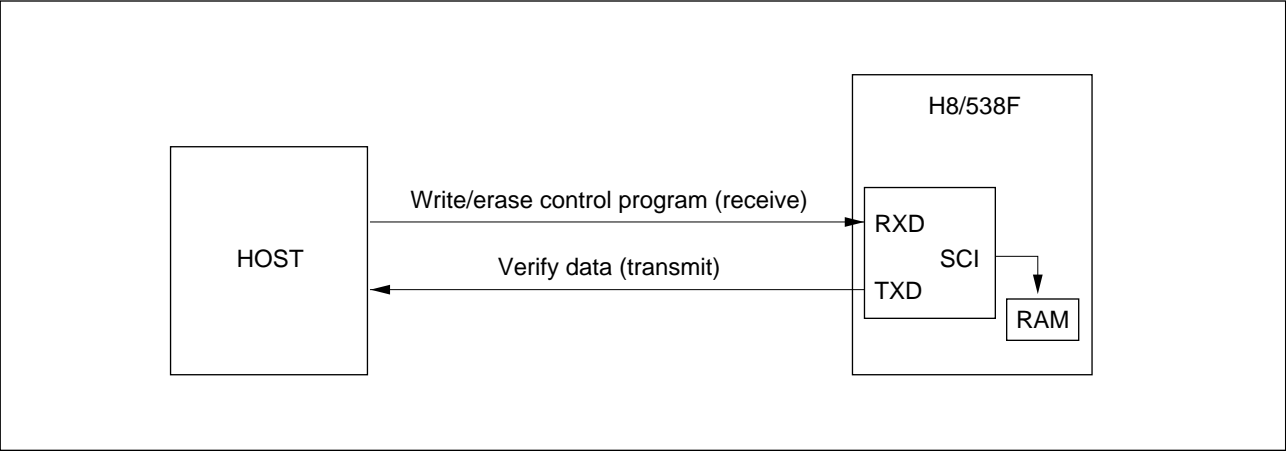
On-Board Programming Mode: This mode enables the on-chip flash memory to be programmed, erased, and verified. This mode operates in one of two further modes (boot mode and user program mode) which are selected by inputs at the mode pins (MD₂ to MD₀) and V_{PP} pin.

Selection of On-Board Programming Mode

Mode		V _{PP}	MD ₂	MD ₁	MD ₀	Remarks
Boot mode	Mode2	12 V	12 V	1	0	0: V _{IL} 1: V _{IH}
	Mode 4		12 V	0	0	
	Mode 7		12 V	1	1	
User program mode	Mode 2		0	1	0	
	Mode 4		1	0	0	
	Mode 7		1	1	1	

Boot Mode: If boot mode is selected, when the H8/538F comes out of the reset state a built-in boot program is executed. The built-in boot program uses the H8/538F's on-chip serial communication interface* (SCI) to asynchronously transfer the user's write/erase control program from the host into RAM. After the transfer is completed, the program now stored in RAM is executed.

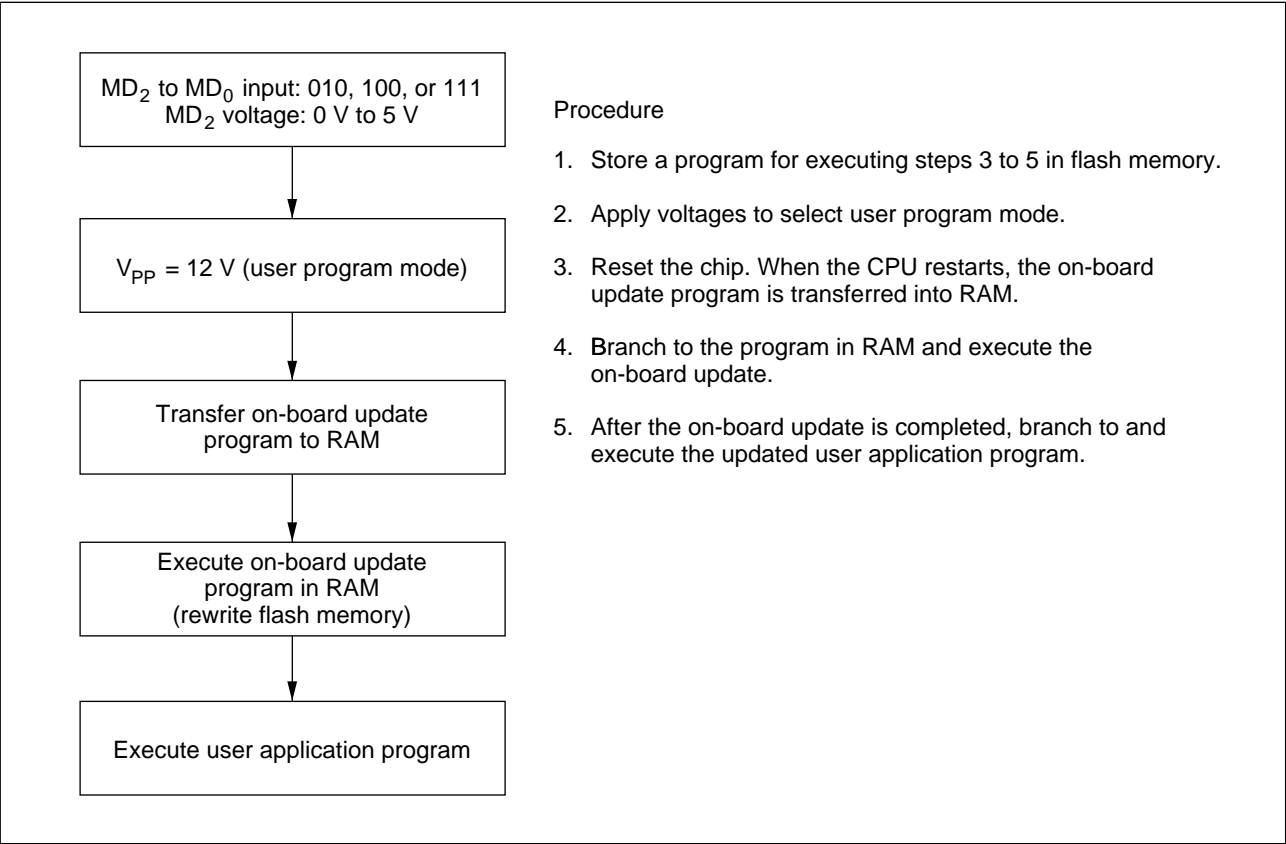
Note: * SCI channel 1.



System Configuration using Boot Mode

User Program Mode: If user program mode is selected, the H8/538F's flash memory can be erased and programmed by user software. Flash memory contents can be updated on-board by providing a V_{PP} power supply and means for supplying new data, and storing an update program in part of the program area. The flash memory cannot be read while being written or erased, so the update program should either be stored in external memory, or transferred to RAM and executed from RAM.

Procedure for User Program Mode (Using On-Chip RAM)



6.9.4 Emulation of Flash Memory by RAM

Flash memory cannot always be erased and programmed quickly enough for applications that require real-time tuning of parameter data. In such cases part of the RAM area can be used to shadow small blocks of flash memory, enabling flash memory updates to be emulated at real-time speeds. This usage of RAM is controlled by bits 3 to 0 of the RAM control register (RAMCR).

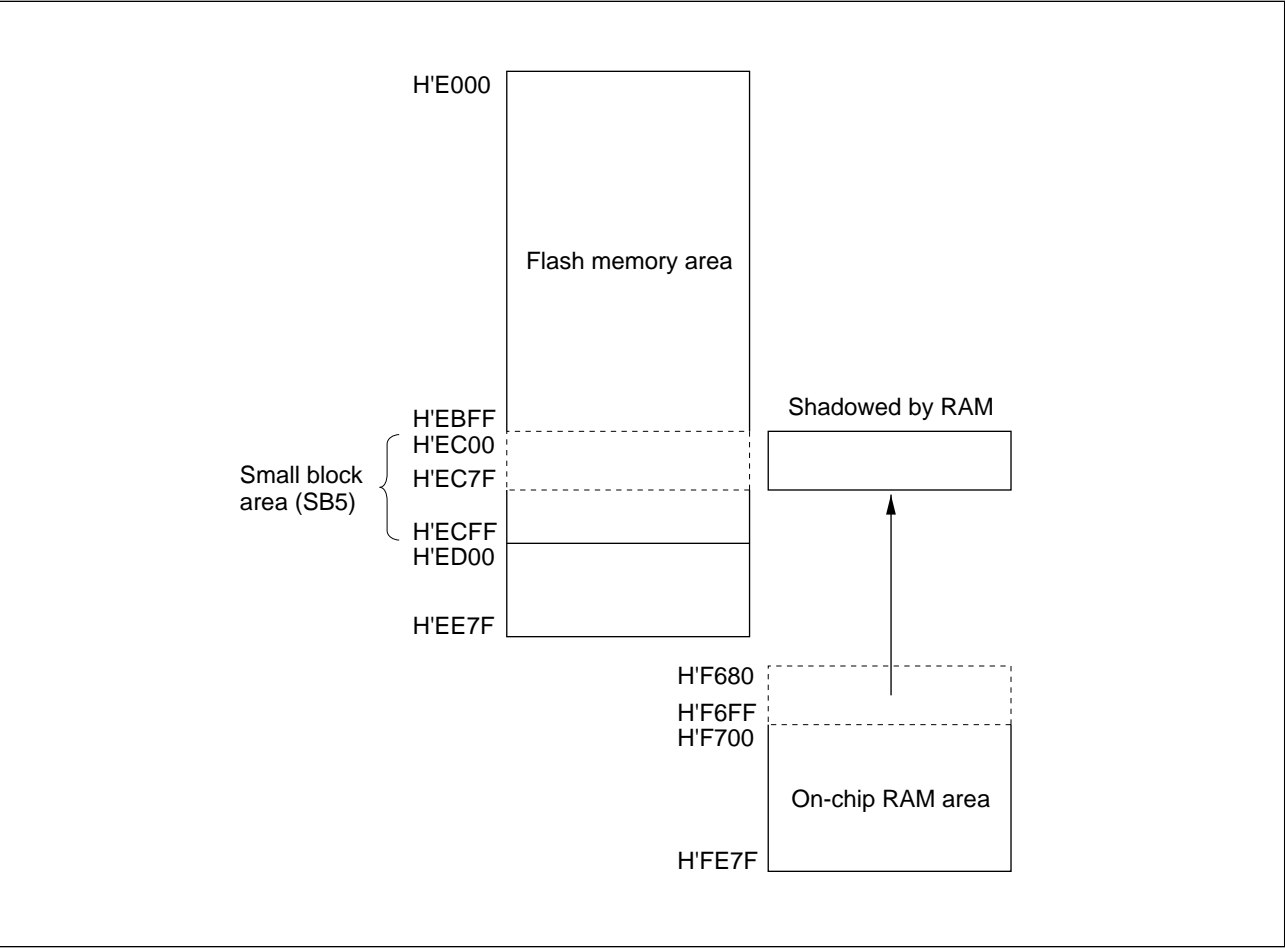
RAM Control Register (RAMCR)

7	6	5	4	3	2	1	0
RAME	—	—	—	RAMS	RAM2	RAM1	RAM0

RAM Area Selection

No.	RAM Area	Bit 3 RAMS	Bit 2 RAM2	Bit 1 RAM1	Bit 0 RAM0
1	H'F680 to H'F6FF	0	0/1	0/1	0/1
2	H'EC00 to H'EC7F	1	0	0	0
3	H'EC80 to H'ECFF	1	0	0	1
4	H'ED00 to H'ED7F	1	0	1	0
5	H'ED80 to H'EDFF	1	0	1	1
6	H'EE00 to H'EE7F	1	1	0	0

Example of Real-Time Emulation of Flash Memory Update



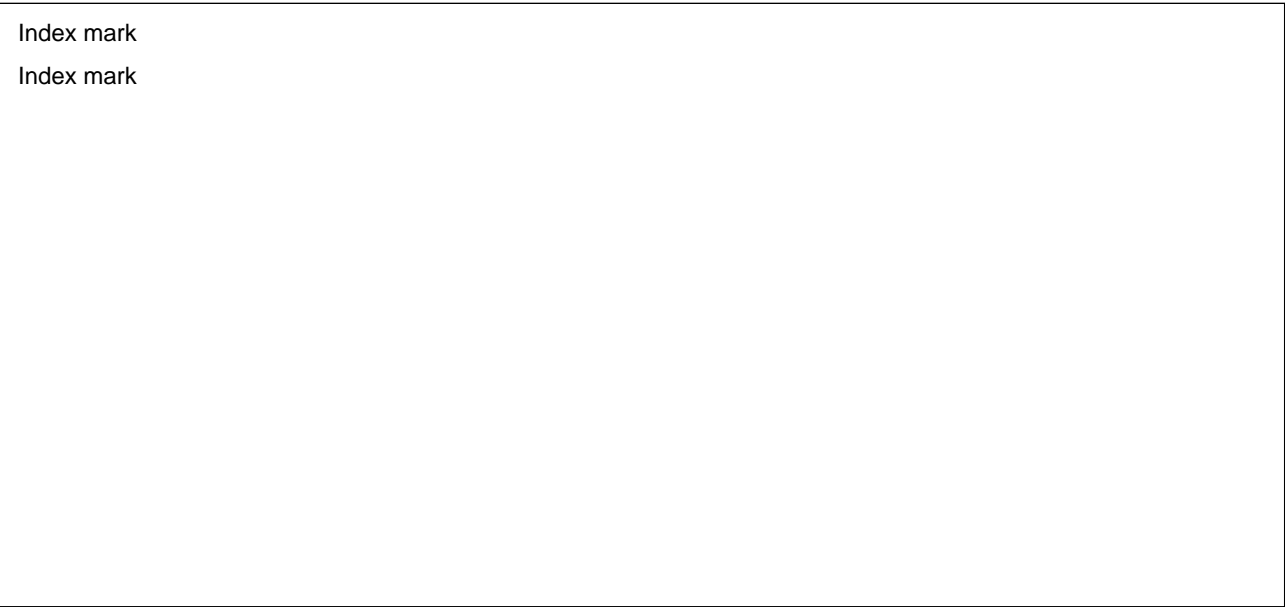
Procedure

1. Assign part of RAM (H'F680 to H'F6FF) to shadow the small block area (SB5) requiring real-time update. Set bits 3 to 0 in RAMCR to 1, 0, 0, 0.
2. Carry out the real-time update in the shadow RAM.
3. After finalizing the update values, release the shadow RAM (clear the RAMS bit to 0).
4. Program flash memory with the data written in RAM addresses H'F680 to H'F6FF.

6.9.5 Programming with an EPROM Programmer

The H8/538F’s on-chip flash memory can be written, erased, and verified with a standard EPROM programmer and socket adapter. The settings are the same as for the HN28F101 1-Mbit flash memory.

Socket Adapter: The 112-pin-to-32-pin socket adapter enables the on-chip flash memory to be programmed, erased, and verified using a standard EPROM programmer.



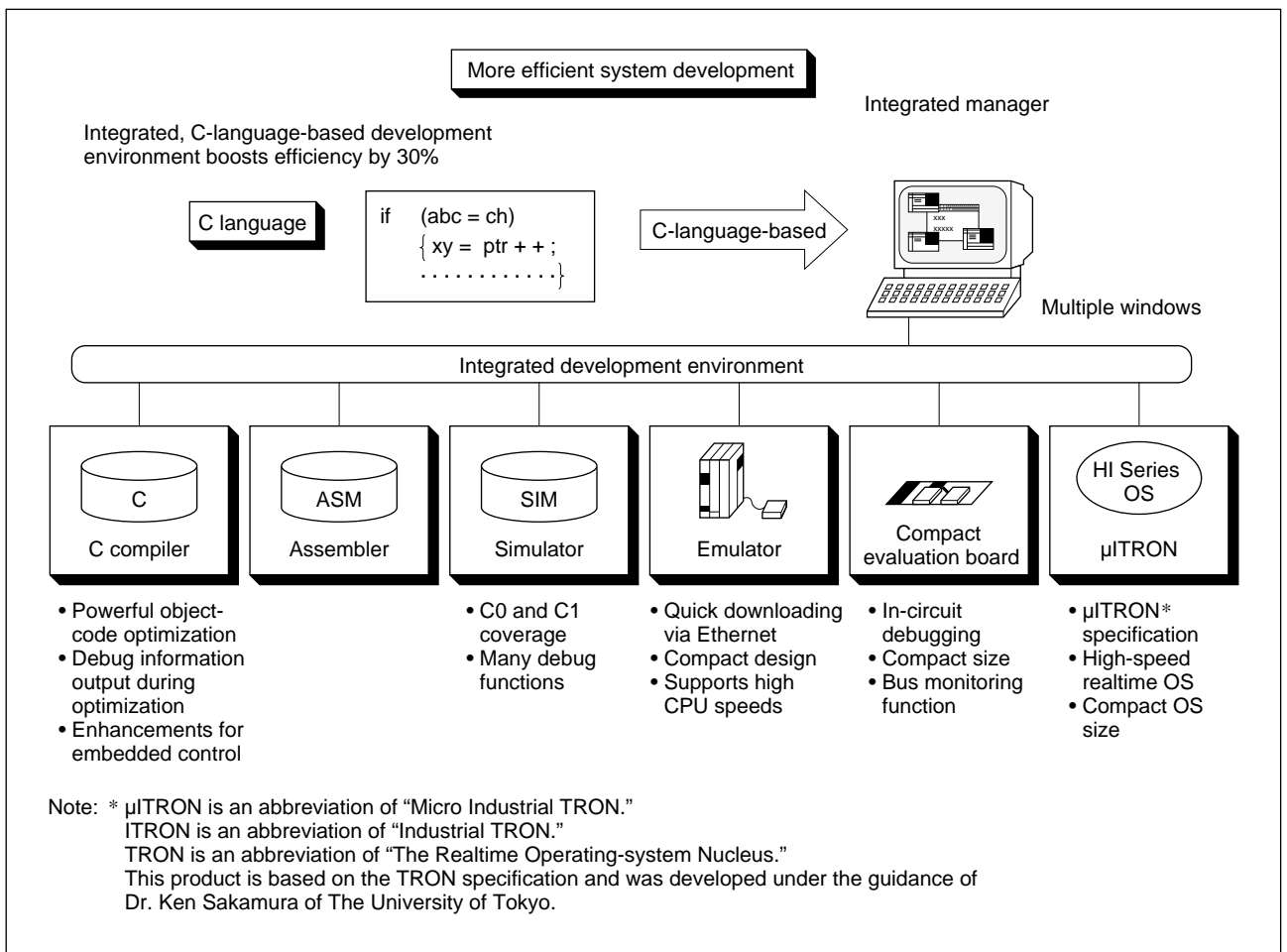
Section 7 Support Tools

The H8/538 programming environment includes both support software and hardware tools for efficient program development.

- Software
 - Integrated manager
 - H8/500-Series C compiler
 - H8/500-Series assembler
 - Linkage editor (including librarian and object module converter)
 - H8/500-Series simulator/debugger
 - H8/500-Series realtime operating systems
 - Interface software for programming flash memory
- Hardware
 - E7000 realtime emulator for H8/538 Series
 - H8/538 compact evaluation board

7.1 Software

- Integrated manager (under development)
 - Provides a menu-driven environment with the same user interface for all procedures from coding to debugging
 - Multi-window support
 - Windows can simultaneously display a variety of information, from program code to debug data.
 - Simplified user interface
 - For example: the program editor exits to an auto-make function; assemble/compile errors automatically activate the editor; completed programs can be loaded automatically to the simulator or emulator.
 - Ample help functions



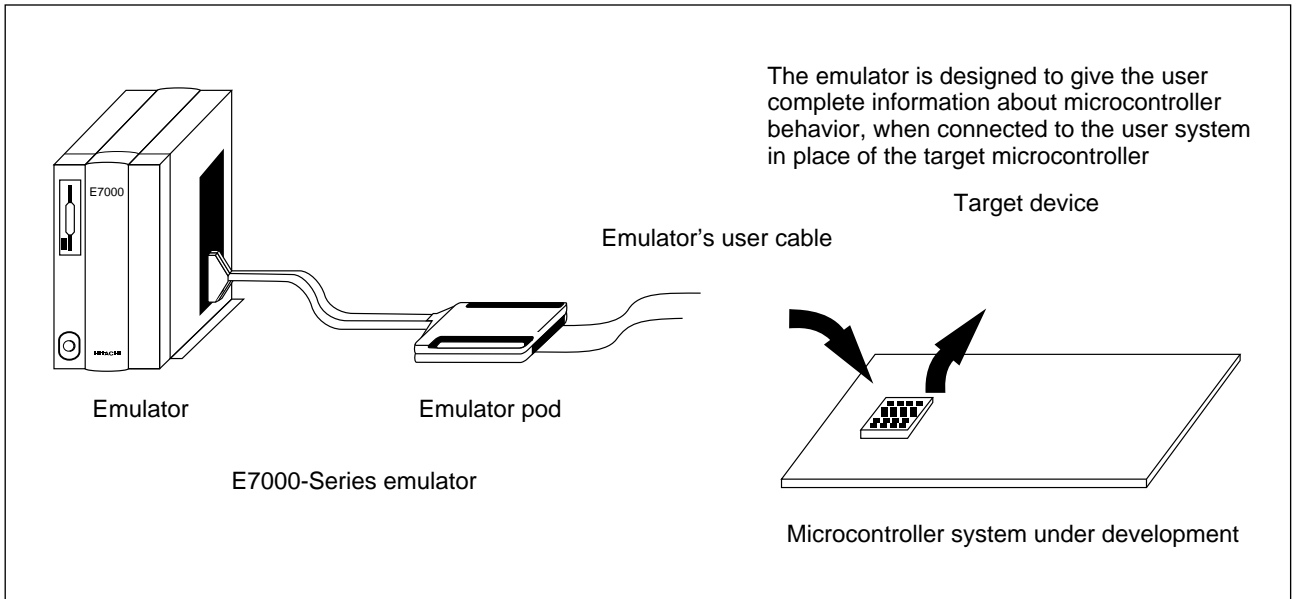
Integrated Manager

- C compiler
 - Conforms to ANSI (American National Standards Institute) C language draft specifications
 - Optimizes object code to reduce program size and execution time (1.2 times assembly language)
- Assembler
 - Generates output in SYSROF object format
- Linkage editor (including librarian and object module converter)
 - Linkage editor: links and relocates object programs output (in SYSROF format) by the assembler or C compiler, and generates load modules (in SYSROF format)
 - Librarian: stores user programs in libraries
 - Object module converter: converts load modules output by the linkage editor from SYSROF format to S-type format

- H8/500-Series simulator/debugger
 - Enables the user to debug programs on a host computer, without a target system
 - Many types of breakpoint settings
 - Permits simulation with relocatable object code
- H8/500-Series realtime operating systems (HI8-S, HI8, HI8-EX)
 - Conforms to the μ ITRON specification
 - Simplifies porting of operating systems and application programs for other H-series microcontrollers
 - High-speed operating system, designed for realtime control
 - Supports stack sharing between tasks (HI8-S only)
- Interface software for flash memory programming (under development)
 - Provides all programs needed for on-board programming via the serial port
 - Runs on MS-DOS platforms

7.2 Hardware

E7000 Realtime Emulator



- Hardware breakpointing (4 breakpoints)
 - Settable breakpoint conditions

Address (with bit-mask and range specifications), data bus (with range specification), \overline{RD} and \overline{WR} signals, external interrupts (\overline{NMI} , $\overline{IRQ0}$ to $\overline{IRQ3}$), pass count (maximum 4095), delay count (maximum 32 kcycles)
 - Sequential breakpointing

Program execution halts when two to four breakpoints are passed in a specified order
- PC breakpointing
 - Settable breakpoint conditions: program counter, pass count
- Realtime trace
 - Information traced: address bus, data bus, external probe signals, control signals
 - Trace information capture

Free trace: all information generated during execution is captured
Range specification: specified trace capture conditions
Stop trace: trace stops when a condition is satisfied
Subroutine trace: tracing of execution of a particular subroutine
 - Trace information search: information in the trace buffer can be searched according to specified conditions
 - Trigger conditions: a trigger signal can be output when trigger conditions are satisfied

- CO coverage function
Indicates the percent of the area covered in a program run
- Performance analysis
Measures program execution efficiency (four subroutines)
- Parallel mode
Commands can be entered during a program run without halting execution

H8/538 Compact Evaluation Board

- Supports realtime emulation (10 MHz)
- EPROM socket for emulating on-chip ROM
- On-chip ROM can also be emulated in emulation RAM
- Firmware mapped independently of user address space
The entire address space is available to the user
- Built-in terminal interface (RS-232C)
Supports data transfer with a host computer
- Battery backup of emulation memory (if external power supply is attached)
- Target system connectors
Three special 40-pin connectors

