

[查询H8/510供应商](#)

[捷多邦，专业PCB打样工厂，24小时加急出货](#)

H8/510 Hardware Manual



Preface

The H8/510 is a high-performance microcomputer, featuring a high-speed CPU with 16-bit internal data paths and a full complement of on-chip supporting modules. The H8/510 is an ideal microcontroller for a wide variety of medium-scale devices, including both office and industrial equipment and consumer products.

The CPU has a highly orthogonal, optimized instruction set designed for fast execution of programs coded in the high-level C language.

On-chip facilities include a DRAM refresh controller, numerous timers, serial I/O, an A/D converter, I/O ports, and other functions for compact implementation of high-performance application systems.

This manual gives a hardware description of the H8/510. For details of the instruction set, refer to the *H8/500 Series Programming Manual*, which applies to all chips in the H8/500 Series.

Contents

Section 1	Overview	1
1.1	Features	1
1.2	Block Diagram	4
1.3	Pin Arrangements and Functions	5
1.3.1	Pin Arrangement	5
1.3.2	Pin Functions	6
Section 2	MCU Operating Modes and Address Space	15
2.1	Overview	15
2.1.1	Selection of MCU Mode	15
2.1.2	Register Control of MCU Mode	15
2.2	Mode Control Register (MDCR)	16
2.3	Mode Descriptions	16
2.4	Pin Functions in Each MCU Mode	17
2.5	Memory Map in Each MCU Mode	19
Section 3	CPU	21
3.1	Overview	21
3.1.1	Features	21
3.1.2	Address Space	22
3.1.3	Register Configuration	23
3.2	CPU Register Descriptions	24
3.2.1	General Registers	24
3.2.2	Control Registers	25
3.2.3	Initial Register Values	30
3.3	Data Formats	31
3.3.1	Data Formats in General Registers	31
3.3.2	Data Formats in Memory	32
3.4	Instructions	34
3.4.1	Basic Instruction Formats	34
3.4.2	Addressing Modes	35
3.4.3	Effective Address Calculation	37
3.5	Instruction Set	40
3.5.1	Overview	40
3.5.2	Data Transfer Instructions	42
3.5.3	Arithmetic Instructions	43
3.5.4	Logic Operations	44
3.5.5	Shift Operations	45
3.5.6	Bit Manipulations	46

3.5.7	Branching Instructions	47
3.5.8	System Control Instructions	49
3.5.9	Short-Format Instructions	52
3.6	Operating Modes	52
3.6.1	Minimum Mode	52
3.6.2	Maximum Mode	53
3.7	Basic Operational Timing	53
3.7.1	Overview	53
3.7.2	Two-State Access Cycle	54
3.7.3	Three-State Access Cycle	55
3.7.4	On-Chip Register Field Access Cycle	56
3.7.5	Pin States during Register Field Access	57
3.8	CPU States	57
3.8.1	Overview	57
3.8.2	Program Execution State	59
3.8.3	Exception-Handling State	59
3.8.4	Bus-Released State	60
3.8.5	Reset State	64
3.8.6	Power-Down State	64
Section 4 Exception Handling		65
4.1	Overview	65
4.1.1	Types of Exception Handling and Their Priority	65
4.1.2	Hardware Exception-Handling Sequence	66
4.1.3	Exception Factors and Vector Table	66
4.2	Reset	69
4.2.1	Overview	69
4.2.2	Reset Sequence	69
4.2.3	Stack Pointer Initialization	70
4.3	Address Error	73
4.3.1	Illegal Instruction Prefetch	73
4.3.2	Word Data Access at Odd Address	73
4.4	Trace	74
4.5	Interrupts	74
4.6	Invalid Instruction	76
4.7	Trap Instructions and Zero Divide	76
4.8	Cases in Which Exception Handling is Deferred	76
4.8.1	Instructions that Disable Interrupts	76
4.8.2	Disabling of Exceptions Immediately after a Reset	77
4.8.3	Disabling of Interrupts after a Data Transfer Cycle	78

4.9	Stack Status after Completion of Exception Handling	78
4.9.1	PC Value Pushed on Stack for Trace, Interrupts, Trap Instructions, and Zero Divide Exceptions	80
4.9.2	PC Value Pushed on Stack for Address Error and Invalid Instruction Exceptions	80
4.10	Notes on Use of the Stack	80
Section 5 Interrupt Controller		81
5.1	Overview	81
5.1.1	Features	81
5.1.2	Block Diagram	82
5.1.3	Register Configuration	83
5.2	Interrupt Types	83
5.2.1	External Interrupts	83
5.2.2	Internal Interrupts	87
5.2.3	Interrupt Vector Table	87
5.3	Register Descriptions	89
5.3.1	Interrupt Priority Registers A to D (IPRA to IPRD)	89
5.3.2	Timing of Priority Setting	90
5.4	Interrupt Handling Sequence	90
5.4.1	Interrupt Handling Flow	90
5.4.2	Stack Status after Interrupt Handling Sequence	93
5.4.3	Timing of Interrupt Exception-Handling Sequence	94
5.5	Interrupts During Operation of the Data Transfer Controller	94
5.6	Interrupt Response Time	97
Section 6 Data Transfer Controller		99
6.1	Overview	99
6.1.1	Features	99
6.1.2	Block Diagram	99
6.1.3	Register Configuration	100
6.2	Register Descriptions	101
6.2.1	Data Transfer Mode Register (DTMR)	101
6.2.2	Data Transfer Source Address Register (DTSR)	102
6.2.3	Data Transfer Destination Register (DTDR)	102
6.2.4	Data Transfer Count Register (DTCR)	102
6.2.5	Data Transfer Enable Registers A to D (DTEA to DTED)	103
6.3	Data Transfer Operation	104
6.3.1	Data Transfer Cycle	104
6.3.2	DTC Vector Table	106
6.3.3	Location of Register Information in Memory	108

6.3.4	Length of Data Transfer Cycle	108
6.4	Procedure for Using the DTC	110
6.5	Example	111
Section 7 Wait-State Controller.....		113
7.1	Overview	113
7.1.1	Features	113
7.1.2	Block Diagram	114
7.1.3	Register Configuration	114
7.2	Wait-State Control Register	115
7.3	Operation in Each Wait Mode	116
7.3.1	Programmable Wait Mode	116
7.3.2	Pin Wait Mode	117
7.3.3	Pin Auto-Wait Mode	119
Section 8 Clock Pulse Generator.....		121
8.1	Overview	121
8.1.1	Block Diagram	121
8.2	Oscillator Circuit	121
8.3	System Clock Divider	124
Section 9 I/O Ports.....		125
9.1	Overview	125
9.2	Port 1	127
9.2.1	Overview	127
9.2.2	Port 1 Registers	127
9.2.3	Pin Functions in Each Mode	128
9.3	Port 2	130
9.3.1	Overview	130
9.3.2	Port 2 Registers	130
9.3.3	Pin Functions in Each Mode	131
9.4	Port 3	132
9.4.1	Overview	132
9.4.2	Port 3 Registers	133
9.4.3	Pin Functions	135
9.5	Port 4	135
9.5.1	Overview	135
9.5.2	Port 4 Registers	136
9.5.3	Pin Functions	137
9.6	Port 5	139
9.6.1	Overview	139

9.6.2	Port 5 Registers	140
9.6.3	Pin Functions	141
9.7	Port 6	142
9.7.1	Overview	142
9.7.2	Port 6 Registers	143
9.8	Port 7	144
9.8.1	Overview	144
9.8.2	Port 7 Registers	144
9.9	Port 8	145
9.9.1	Overview	145
9.9.2	Port 8 Registers	145
9.9.3	Pin Functions	147

Section 10 16-Bit Free-Running Timers151

10.1	Overview	151
10.1.1	Features	151
10.1.2	Block Diagram	152
10.1.3	Input and Output Pins	153
10.1.4	Register Configuration	154
10.2	Register Descriptions	155
10.2.1	Free-Running Counter (FRC)—H'FEA2, H'FEB2	155
10.2.2	Output Compare Registers A and B (OCRA and OCRB)—H'FEA4 and H'FEA6, H'FEB4 and H'FEB6	156
10.2.3	Input Capture Register (ICR)—H'FEA8, H'FEB8	156
10.2.4	Timer Control Register (TCR)—H'FEA0, H'FEB0	157
10.2.5	Timer Control/Status Register (TCSR)—H'FEA1, H'FEB1	159
10.3	CPU Interface	162
10.4	Operation	164
10.4.1	FRC Incrementation Timing	164
10.4.2	Output Compare Timing	165
10.4.3	Input Capture Timing	167
10.4.4	Setting of FRC Overflow Flag (OVF)	169
10.5	CPU Interrupts and DTC Interrupts	169
10.6	Synchronization of Free-Running Timers 1 and 2	170
10.6.1	Synchronization after a Reset	170
10.6.2	Synchronization by Writing to FRCs	170
10.7	Sample Application	174
10.8	Application Notes	174

Section 11 8-Bit Timer181

11.1	Overview	181
------	----------------	-----

11.1.1	Features	181
11.1.2	Block Diagram	182
11.1.3	Input and Output Pins	183
11.1.4	Register Configuration	183
11.2	Register Descriptions	183
11.2.1	Timer Counter (TCNT)—H'FEC4	183
11.2.2	Time Constant Registers A and B (TCORA and TCORB)—H'FEC2 and H'FEC3	184
11.2.3	Timer Control Register (TCR)—H'FEC0	184
11.2.4	Timer Control/Status Register (TCSR)—H'FEC1	186
11.3	Operation	188
11.3.1	TCNT Incrementation Timing	188
11.3.2	Compare Match Timing	189
11.3.3	External Reset of TCNT	190
11.3.4	Setting of TCNT Overflow Flag	191
11.4	CPU Interrupts and DTC Interrupts	192
11.5	Sample Application	193
11.6	Application Notes	194
Section 12 Refresh Controller		201
12.1	Overview	201
12.1.1	Features	201
12.1.2	Block Diagram	201
12.1.3	Register Configuration	202
12.2	Refresh Control Register (RFSHCR)—H'FED8	203
12.3	Operation	205
12.3.1	Wait State Insertion	206
12.3.2	TP Insertion	207
12.4	Operation in Power-Down State	209
12.5	Operation in Reset State	209
12.6	Application Notes	209
Section 13 Serial Communication Interface		211
13.1	Overview	211
13.1.1	Features	211
13.1.2	Block Diagram	212
13.1.3	Input and Output Pins	213
13.1.4	Register Configuration	213
13.2	Register Descriptions	214
13.2.1	Receive Shift Register (RSR)	214
13.2.2	Receive Data Register (RDR)—H'FECD and H'FED5	214

13.2.3	Transmit Shift Register (TSR)	214
13.2.4	Transmit Data Register (TDR)—H'FECB and H'FED3	215
13.2.5	Serial Mode Register (SMR)—H'FEC8 and H'FED0	215
13.2.6	Serial Control Register (SCR)—H'FECA and H'FED2	217
13.2.7	Serial Status Register (SSR)—H'FECC and H'FED4	219
13.2.8	Bit Rate Register (BRR)—H'FEC9 and H'FED1	221
13.3	Operation	226
13.3.1	Overview	226
13.3.2	Asynchronous Mode	227
13.3.3	Synchronous Mode	231
13.4	CPU Interrupts and DTC Interrupts	234
13.5	Application Notes	235
Section 14 A/D Converter		239
14.1	Overview	239
14.1.1	Features	239
14.1.2	Block Diagram	240
14.1.3	Input Pins	241
14.1.4	Register Configuration	241
14.2	Register Descriptions	242
14.2.1	A/D Data Register (ADDR)—H'FE90 to H'FE97	242
14.2.2	A/D Control/Status Register (ADCSR)—H'FE98	243
14.2.3	A/D Control Register (ADCR)—H'FE99	245
14.2.4	External Triggering of A/D Conversion	245
14.3	CPU Interface	246
14.4	Operation	247
14.4.1	Single Mode	248
14.4.2	Scan Mode	251
14.5	Input Sampling Time and A/D Conversion Time	253
14.6	Interrupts and the Data Transfer Controller	255
Section 15 Bus Controller		257
15.1	Overview	257
15.1.1	Features	257
15.1.2	Block Diagram	258
15.1.3	Register Configuration	259
15.2	Register Descriptions	259
15.2.1	Byte Area Top Register (ARBT)—H'FF16	259
15.2.2	Three-State Area Top Register (AR3T)—H'FF17	260
15.3	Operation	261
15.4	Notes and Precautions	266

Section 16 Watchdog Timer	271
16.1 Overview	271
16.1.1 Features	271
16.1.2 Block Diagram	272
16.1.3 Register Configuration	272
16.2 Register Descriptions	273
16.2.1 Timer Counter TCNT—H'FF10 (Write), H'FF11 (Read)	273
16.2.2 Timer Control/Status Register (TCSR)—H'FF10	273
16.2.3 Reset Control/Status Register (RSTCSR)—H'FF1F (Read), H'FF1E (Write)	275
16.2.4 Notes on Register Access	276
16.3 Operation	278
16.3.1 Watchdog Timer Mode	278
16.3.2 Interval Timer Mode	279
16.3.3 Operation in Software Standby Mode	280
16.3.4 Setting of Overflow Flag	280
16.3.5 Setting of Watchdog Timer Reset (WRST) Bit	281
16.4 Application Notes	282
Section 17 Power-Down State	285
17.1 Overview	285
17.2 Sleep Mode	286
17.2.1 Transition to Sleep Mode	286
17.2.2 Exit from Sleep Mode	286
17.3 Software Standby Mode	286
17.3.1 Transition to Software Standby Mode	286
17.3.2 Software Standby Control Register (SBYCR)	287
17.3.3 Exit from Software Standby Mode	288
17.3.4 Sample Application of Software Standby Mode	288
17.3.5 Application Notes	289
17.4 Hardware Standby Mode	289
17.4.1 Transition to Hardware Standby Mode	289
17.4.2 Recovery from Hardware Standby Mode	290
17.4.3 Timing Sequence of Hardware Standby Mode	290
Section 18 E Clock Interface	291
18.1 Overview	291
Section 19 Electrical Specifications	295
19.1 Absolute Maximum Ratings	295
19.2 Electrical Characteristics	295
19.2.1 DC Characteristics	295

19.2.2 AC Characteristics	299
19.2.3 A/D Converter Characteristics	306
19.3 MCU Operational Timing	307
19.3.1 Bus Timing	307
19.3.2 Control Signal Timing	311
19.3.3 Clock Timing	312
19.3.4 I/O Port Timing	314
19.3.5 16-Bit Free-Running Timer Timing	314
19.3.6 8-Bit Timer Timing	315
19.3.7 Serial Communication Interface Timing	316
19.3.8 Refresh Timing	317
Appendix A Instructions	319
A.1 Instruction Set	319
A.2 Instruction Codes	324
A.3 Operation Code Map	335
A.4 Instruction Execution Cycles	340
Appendix B Register Field	351
B.1 Register Addresses and Bit Names	351
B.2 Register Descriptions	355
Appendix C I/O Port Schematic Diagrams	390
C.1 Schematic Diagram of Port 1	390
C.2 Schematic Diagram of Port 2	391
C.3 Schematic Diagram of Port 3	392
C.4 Schematic Diagram of Port 4	396
C.5 Schematic Diagram of Port 5	402
C.6 Schematic Diagram of Port 6	404
C.7 Schematic Diagram of Port 7	405
C.8 Schematic Diagram of Port 8	406
Appendix D Memory Map	410
Appendix E Pin States	411
E.1 States of I/O Ports	411
E.2 Pin Status in the Reset State	412
Appendix F Package Dimensions	418

Section 1 Overview

1.1 Features

The H8/510 is an original Hitachi CMOS microcomputer unit (MCU) comprising a high-performance CPU core with an internal 16-bit architecture plus a full range of supporting functions.

The CPU features a highly orthogonal instruction set that permits addressing modes and data sizes to be specified independently in each instruction. An internal 16-bit architecture and the capability for 16-bit, two-state access to external memory enhance the CPU's data-processing capability and provide the speed needed for realtime control applications.

The on-chip supporting functions include timers, a serial communication interface (SCI), refresh controller, bus controller, A/D converter, and I/O ports. An on-chip data transfer controller (DTC) provides an efficient way to transfer data in either direction between memory and I/O.

Table 1-1 lists the main features of the H8/510 chip.

Table 1-1 Features

Feature	Description
CPU	General-register machine <ul style="list-style-type: none">• Eight 16-bit general registers• Five 8-bit and two 16-bit control registers High speed <ul style="list-style-type: none">• Maximum clock rate: 10 MHz (oscillator frequency: 20 MHz) Two operating modes <ul style="list-style-type: none">• Minimum mode: up to 64-kbyte address space• Maximum mode: up to 16-Mbyte address space Highly orthogonal instruction set <ul style="list-style-type: none">• Addressing modes and data size can be specified independently for each instruction Register and memory addressing modes <ul style="list-style-type: none">• Register-register operations• Register-memory operations Instruction set optimized for C language <ul style="list-style-type: none">• Special short formats for frequently-used instructions and addressing modes
16-Bit free-running timer (FRT) (2 channels)	Each channel provides: <ul style="list-style-type: none">• 1 free-running counter (which can count external events)• 2 output-compare registers• 1 input capture register
8-Bit timer (1 channel)	<ul style="list-style-type: none">• One 8-bit up-counter (which can count external events)• 2 time constant registers
Serial communication interface (SCI) (2 channels)	Each channel has the following features: <ul style="list-style-type: none">• Asynchronous or synchronous mode (selectable)• Full duplex: can send and receive simultaneously• Built-in baud rate generator
Refresh controller	<ul style="list-style-type: none">• Selectable refresh interval and refresh cycle length• Can output 12-bit refresh addresses• A T_P state can be inserted before the T_1 state to satisfy $\overline{\text{RAS}}$ precharge time requirements of DRAM chips
A/D converter	<ul style="list-style-type: none">• 10-Bit resolution• 4 channels, controllable in single mode or scan mode (selectable)• Sample-and-hold function• Conversion can be externally triggered

Table 1-1 Features (cont)

Feature	Description				
I/O ports	<ul style="list-style-type: none"> • 56 input/output pins (seven 8-bit ports) • 4 input-only pins (one 4-bit port) 				
Interrupt controller (INTC)	<ul style="list-style-type: none"> • 5 external interrupt pins: NMI, $\overline{\text{IRQ}}_0$ (level), $\overline{\text{IRQ}}_1$ to $\overline{\text{IRQ}}_3$ (edge) • 18 on-chip interrupt sources • 8 priority levels 				
Data transfer controller (DTC)	<ul style="list-style-type: none"> • Performs efficient, rapid, bidirectional data transfer between memory and I/O with minimal CPU programming 				
Wait-state controller (WSC)	<ul style="list-style-type: none"> • Can insert wait states in access to external memory or I/O 				
Operating modes	4 MCU operating modes <ul style="list-style-type: none"> • Expanded minimum modes, supporting a 64-kbyte address space (modes 1 and 2) • Expanded maximum modes, supporting a 16-Mbyte address space (modes 3 and 4) 3 power-down modes <ul style="list-style-type: none"> • Sleep mode • Software standby mode • Hardware standby mode 				
Watchdog timer (1 channel)	<ul style="list-style-type: none"> • Can output a reset signal when the timer overflows • Can also be used as an interval timer 				
Bus controller	<ul style="list-style-type: none"> • Can select types of bus cycles 				
Other features	<ul style="list-style-type: none"> • E clock output • Clock generator on-chip 				
Product code and package	<table> <tr> <td>Product code</td><td>Package</td></tr> <tr> <td>HD6415108F</td><td>112-Pin QFP (FP-112)</td></tr> </table>	Product code	Package	HD6415108F	112-Pin QFP (FP-112)
Product code	Package				
HD6415108F	112-Pin QFP (FP-112)				

1.2 Block Diagram

Figure 1-1 shows a block diagram of the H8/510 chip.

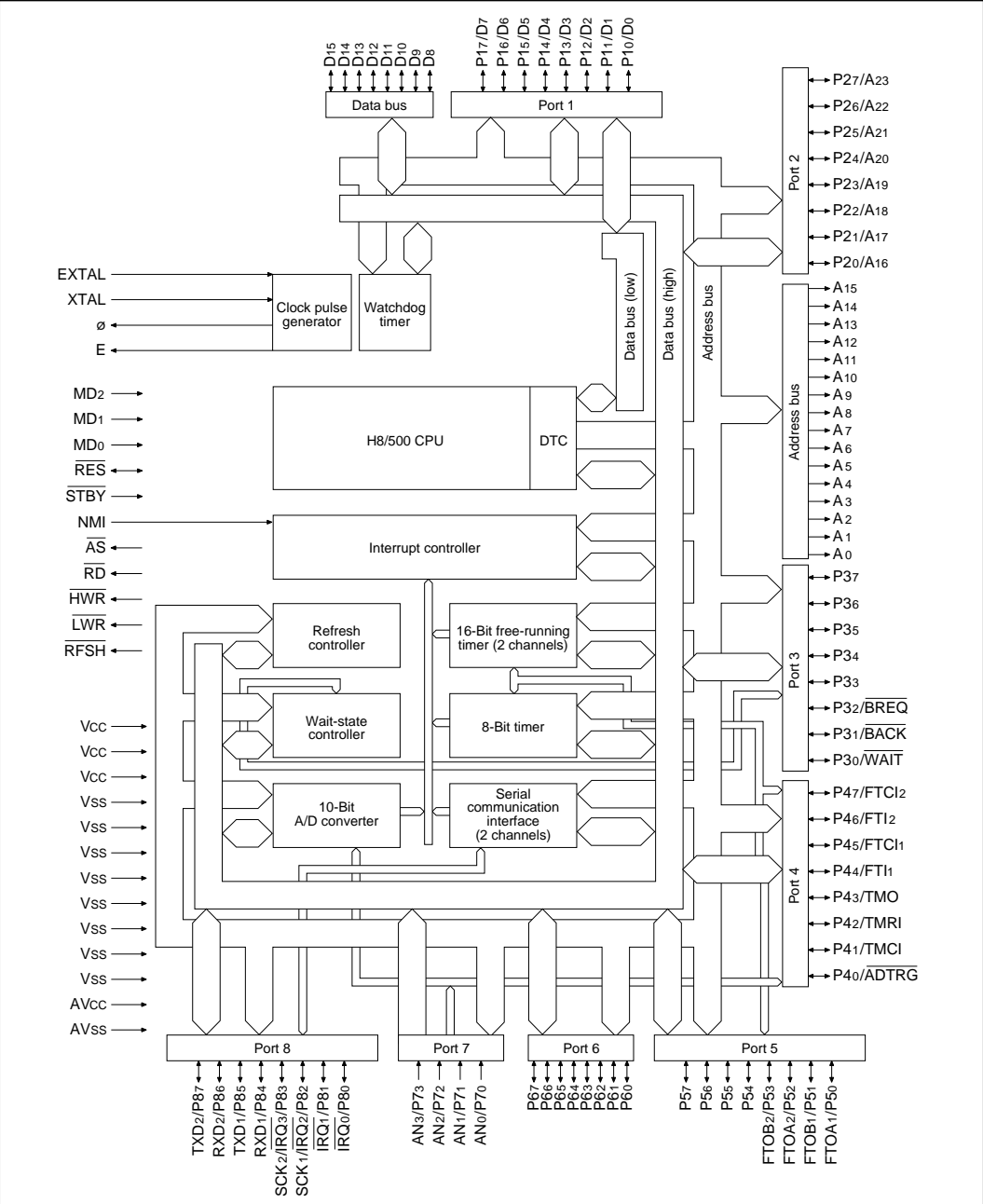


Figure 1-1 Block Diagram

1.3 Pin Arrangements and Functions

1.3.1 Pin Arrangement

Figure 1-2 shows the pin arrangement of the H8/510.

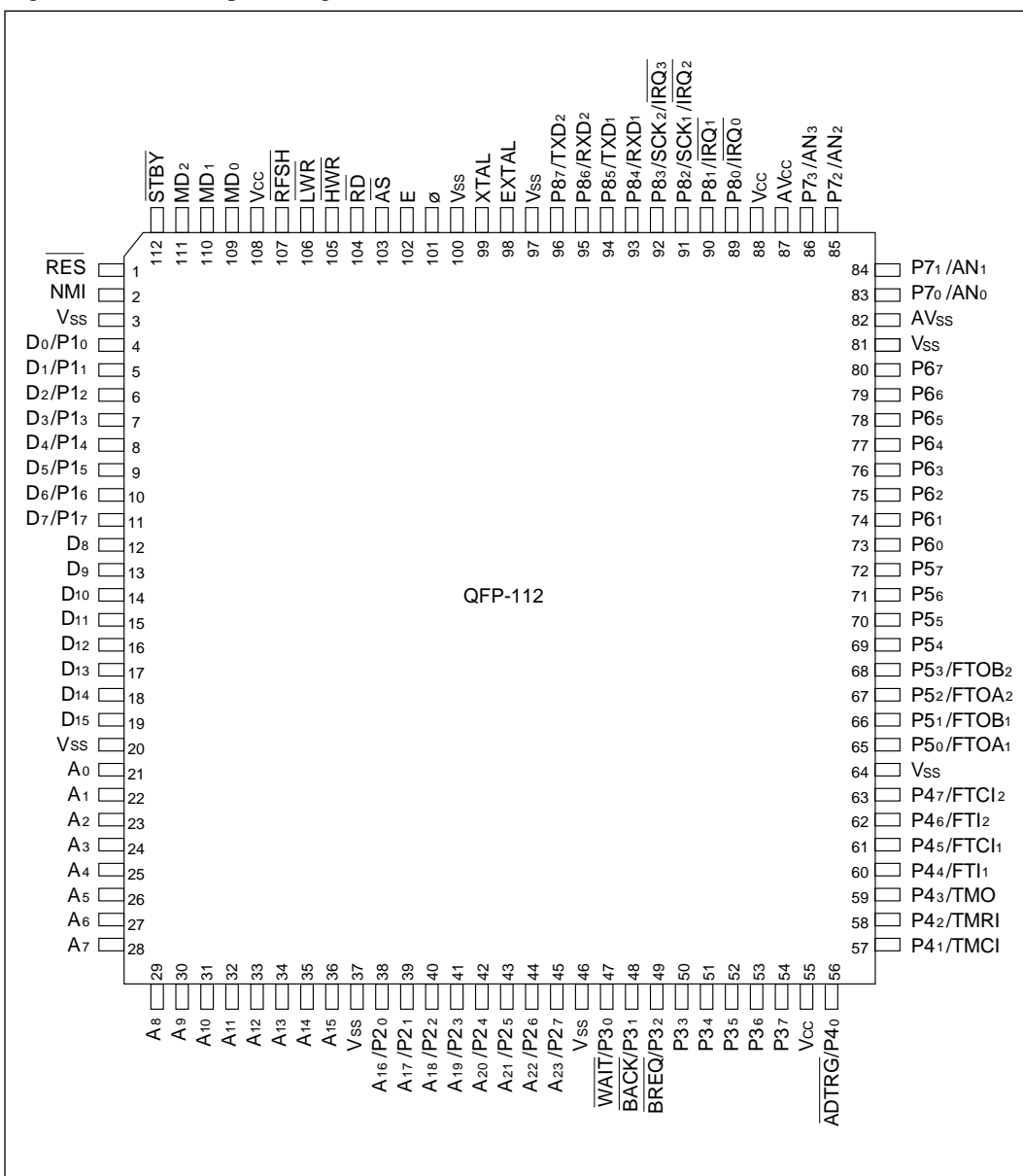


Figure 1-2 Pin Arrangement (Top View)

1.3.2 Pin Functions

Pin Arrangements in Each Operating Mode: Table 1-2 lists the pin arrangements in each operating mode.

Table 1-2 Pin Arrangements in Each Operating Mode

Pin No.	Pin Name			
	Expanded Minimum Modes		Expanded Maximum Modes	
	Mode 1	Mode 2	Mode 3	Mode 4
1	$\overline{\text{RES}}$	$\overline{\text{RES}}$	$\overline{\text{RES}}$	$\overline{\text{RES}}$
2	NMI	NMI	NMI	NMI
3	Vss	Vss	Vss	Vss
4	P10	D0	P10	D0
5	P11	D1	P11	D1
6	P12	D2	P12	D2
7	P13	D3	P13	D3
8	P14	D4	P14	D4
9	P15	D5	P15	D5
10	P16	D6	P16	D6
11	P17	D7	P17	D7
12	D8	D8	D8	D8
13	D9	D9	D9	D9
14	D10	D10	D10	D10
15	D11	D11	D11	D11
16	D12	D12	D12	D12
17	D13	D13	D13	D13
18	D14	D14	D14	D14
19	D15	D15	D15	D15
20	Vss	Vss	Vss	Vss
21	A0	A0	A0	A0
22	A1	A1	A1	A1
23	A2	A2	A2	A2
24	A3	A3	A3	A3
25	A4	A4	A4	A4
26	A5	A5	A5	A5

Table 1-2 Pin Arrangements in Each Operating Mode (cont)

Pin No.	Pin Name			
	Expanded Minimum Modes		Expanded Maximum Modes	
	Mode 1	Mode 2	Mode 3	Mode 4
27	A6	A6	A6	A6
28	A7	A7	A7	A7
29	A8	A8	A8	A8
30	A9	A9	A9	A9
31	A10	A10	A10	A10
32	A11	A11	A11	A11
33	A12	A12	A12	A12
34	A13	A13	A13	A13
35	A14	A14	A14	A14
36	A15	A15	A15	A15
37	VSS	VSS	VSS	VSS
38	P20	P20	A16	A16
39	P21	P21	A17	A17
40	P22	P22	A18	A18
41	P23	P23	A19	A19
42	P24	P24	A20	A20
43	P25	P25	A21	A21
44	P26	P26	A22	A22
45	P27	P27	A23	A23
46	VSS	VSS	VSS	VSS
47	P30/ $\overline{\text{WAIT}}$	P30/ $\overline{\text{WAIT}}$	P30/ $\overline{\text{WAIT}}$	P30/ $\overline{\text{WAIT}}$
48	P31/ $\overline{\text{BACK}}$	P31/ $\overline{\text{BACK}}$	P31/ $\overline{\text{BACK}}$	P31/ $\overline{\text{BACK}}$
49	P32/ $\overline{\text{BREQ}}$	P32/ $\overline{\text{BREQ}}$	P32/ $\overline{\text{BREQ}}$	P32/ $\overline{\text{BREQ}}$
50	P33	P33	P33	P33
51	P34	P34	P34	P34
52	P35	P35	P35	P35
53	P36	P36	P36	P36
54	P37	P37	P37	P37
55	VCC	VCC	VCC	VCC

Table 1-2 Pin Arrangements in Each Operating Mode (cont)

Pin No.	Pin Name			
	Expanded Minimum Modes		Expanded Maximum Modes	
	Mode 1	Mode 2	Mode 3	Mode 4
56	P40/ $\overline{\text{ADTRG}}$	P40/ $\overline{\text{ADTRG}}$	P40/ $\overline{\text{ADTRG}}$	P40/ $\overline{\text{ADTRG}}$
57	P41/TMCI	P41/TMCI	P41/TMCI	P41/TMCI
58	P42/TMRI	P42/TMRI	P42/TMRI	P42/TMRI
59	P43/TMO	P43/TMO	P43/TMO	P43/TMO
60	P44/FTI1	P44/FTI1	P44/FTI1	P44/FTI1
61	P45/FTCI1	P45/FTCI1	P45/FTCI1	P45/FTCI1
62	P46/FTI2	P46/FTI2	P46/FTI2	P46/FTI2
63	P47/FTCI2	P47/FTCI2	P47/FTCI2	P47/FTCI2
64	Vss	Vss	Vss	Vss
65	P50/FTOA1	P50/FTOA1	P50/FTOA1	P50/FTOA1
66	P51/FTOB1	P51/FTOB1	P51/FTOB1	P51/FTOB1
67	P52/FTOA2	P52/FTOA2	P52/FTOA2	P52/FTOA2
68	P53/FTOB2	P53/FTOB2	P53/FTOB2	P53/FTOB2
69	P54	P54	P54	P54
70	P55	P55	P55	P55
71	P56	P56	P56	P56
72	P57	P57	P57	P57
73	P60	P60	P60	P60
74	P61	P61	P61	P61
75	P62	P62	P62	P62
76	P63	P63	P63	P63
77	P64	P64	P64	P64
78	P65	P65	P65	P65
79	P66	P66	P66	P66
80	P67	P67	P67	P67
81	Vss	Vss	Vss	Vss
82	AVss	AVss	AVss	AVss
83	P70/AN0	P70/AN0	P70/AN0	P70/AN0
84	P71/AN1	P71/AN1	P71/AN1	P71/AN1
85	P72/AN2	P72/AN2	P72/AN2	P72/AN2
86	P73/AN3	P73/AN3	P73/AN3	P73/AN3
87	AVcc	AVcc	AVcc	AVcc

Table 1-2 Pin Arrangements in Each Operating Mode (cont)

Pin No.	Pin Name			
	Expanded Minimum Modes		Expanded Maximum Modes	
	Mode 1	Mode 2	Mode 3	Mode 4
88	Vcc	Vcc	Vcc	Vcc
89	P80/ $\overline{\text{IRQ}}_0$	P80/ $\overline{\text{IRQ}}_0$	P80/ $\overline{\text{IRQ}}_0$	P80/ $\overline{\text{IRQ}}_0$
90	P81/ $\overline{\text{IRQ}}_1$	P81/ $\overline{\text{IRQ}}_1$	P81/ $\overline{\text{IRQ}}_1$	P81/ $\overline{\text{IRQ}}_1$
91	P82/ $\overline{\text{IRQ}}_2/\text{SCK}_1$	P82/ $\overline{\text{IRQ}}_2/\text{SCK}_1$	P82/ $\overline{\text{IRQ}}_2/\text{SCK}_1$	P82/ $\overline{\text{IRQ}}_2/\text{SCK}_1$
92	P83/ $\overline{\text{IRQ}}_3/\text{SCK}_2$	P83/ $\overline{\text{IRQ}}_3/\text{SCK}_2$	P83/ $\overline{\text{IRQ}}_3/\text{SCK}_2$	P83/ $\overline{\text{IRQ}}_3/\text{SCK}_2$
93	P84/RXD ₁	P84/RXD ₁	P84/RXD ₁	P84/RXD ₁
94	P85/TXD ₁	P85/TXD ₁	P85/TXD ₁	P85/TXD ₁
95	P86/RXD ₂	P86/RXD ₂	P86/RXD ₂	P86/RXD ₂
96	P87/TXD ₂	P87/TXD ₂	P87/TXD ₂	P87/TXD ₂
97	Vss	Vss	Vss	Vss
98	EXTAL	EXTAL	EXTAL	EXTAL
99	XTAL	XTAL	XTAL	XTAL
100	Vss	Vss	Vss	Vss
101	∅	∅	∅	∅
102	E	E	E	E
103	$\overline{\text{AS}}$	$\overline{\text{AS}}$	$\overline{\text{AS}}$	$\overline{\text{AS}}$
104	$\overline{\text{RD}}$	$\overline{\text{RD}}$	$\overline{\text{RD}}$	$\overline{\text{RD}}$
105	$\overline{\text{HWR}}$	$\overline{\text{HWR}}$	$\overline{\text{HWR}}$	$\overline{\text{HWR}}$
106	$\overline{\text{LWR}}$	$\overline{\text{LWR}}$	$\overline{\text{LWR}}$	$\overline{\text{LWR}}$
107	$\overline{\text{RFSH}}$	$\overline{\text{RFSH}}$	$\overline{\text{RFSH}}$	$\overline{\text{RFSH}}$
108	Vcc	Vcc	Vcc	Vcc
109	MD ₀	MD ₀	MD ₀	MD ₀
110	MD ₁	MD ₁	MD ₁	MD ₁
111	MD ₂	MD ₂	MD ₂	MD ₂
112	$\overline{\text{STBY}}$	$\overline{\text{STBY}}$	$\overline{\text{STBY}}$	$\overline{\text{STBY}}$

Pin Functions: Table 1-3 gives a concise description of the function of each pin.

Table 1-3 Pin Functions

Type	Symbol	Pin No.	I/O	Name and Function
Power	Vcc	55, 88, 108	I	Power: Connected to the power supply (+ 5 V). Connect all Vcc pins to the system power supply (+ 5 V). The chip will not operate if any Vcc pin is left unconnected.
	Vss	3, 20, 37, 46, 64, 81, 97, 100	I	Ground: Connected to ground (0 V). Connect all Vss pins to the system power supply (0 V). The chip will not operate if any Vss pin is left unconnected.
Clock	XTAL	99	I	Crystal: Connected to a crystal oscillator. The crystal frequency should be double the desired system clock frequency. If an external clock is input at the EXTAL pin, the XTAL pin should be left open.
	EXTAL	98	I	External Crystal: Connected to a crystal oscillator or external clock. The frequency of the external clock should be double the desired system clock frequency. See section 8.2, "Oscillator Circuit," for examples of connections to a crystal and external clock.
	∅	101	O	System Clock: Supplies the system clock to peripheral devices.
	E	102	O	Enable Clock: Supplies an E clock to peripheral devices.
System control	BACK	48	O	Bus Request Acknowledge: Indicates that the bus right has been granted to an external device. Notifies an external device that issued a BREQ signal that it now has control of the bus.

Table 1-3 Pin Functions (cont)

Type	Symbol	Pin No.	I/O	Name and Function
System control	$\overline{\text{BREQ}}$	49	I	Bus Request: Sent by an external device to the H8/510 chip to request the bus right.
	$\overline{\text{STBY}}$	112	I	Standby: A transition to the hardware standby mode (a power-down state) occurs when a Low input is received at the $\overline{\text{STBY}}$ pin.
	$\overline{\text{RES}}$	1	I/O	Reset: A Low input resets the H8/510 chip.
Address bus	A23 – A16	45 – 38	O	Address Bus: Address output pins.
	A15 – A0	36 – 21		
Data bus	D15 – D0	19 – 4	I/O	Data Bus: 16-bit bidirectional data bus.
Bus control	$\overline{\text{WAIT}}$	47	I	Wait: Requests the CPU to insert one or more Tw states when accessing an off-chip address.
	$\overline{\text{AS}}$	103	O	Address Strobe: Goes Low to indicate that there is a valid address on the address bus.
	$\overline{\text{RFSH}}$	107	O	Refresh Cycle: Goes Low to indicate that the address output on the address bus is a refresh address.
	$\overline{\text{RD}}$	104	O	Read: Goes Low to indicate that the CPU is reading an external address.
	$\overline{\text{LWR}}$	106	O	Low Write: Goes Low to indicate that the CPU is writing to an external address using the low data bus.
	$\overline{\text{HWR}}$	105	O	High Write: Goes Low to indicate that the CPU is writing to an external address using the high data bus.

Table 1-3 Pin Functions (cont)

Type	Symbol	Pin No.	I/O	Name and Function
Interrupt signals	NMI	2	I	NonMaskable Interrupt: Highest priority interrupt request. The NMI control register (NMICR) determines whether the interrupt is requested on the rising or falling edge of the NMI input.
	$\overline{\text{IRQ}}_0$	89	I	Interrupt Request 0, 1, 2, and 3: Maskable interrupt request pins.
	$\overline{\text{IRQ}}_1$	90		
	$\overline{\text{IRQ}}_2$	91		
	$\overline{\text{IRQ}}_3$	92		
Operating mode control	MD2	111	I	Mode: Input pins for setting the MCU operating mode according to the table below.
	MD1	110		
	MD0	109		
MD2	MD1	MD0	Mode	Description
0	0	0	Mode 0	—
0	0	1	Mode 1	Expanded minimum mode (8-bit bus)
0	1	0	Mode 2	Expanded minimum mode (16-bit bus)
0	1	1	Mode 3	Expanded maximum mode (8-bit bus)
1	0	0	Mode 4	Expanded maximum mode (16-bit bus)
1	0	1	Mode 5	—
1	1	0	Mode 6	—
1	1	1	Mode 7	—
The inputs at these pins are indicated in mode select bits 2 to 0 (MDS2 – MDS0) of the mode control register (MDCR).				

Table 1-3 Pin Functions (cont)

Type	Symbol	Pin No.	I/O	Name and Function
16-Bit free-running timer (FRT)	FTOA1	65	O	FRT Output Compare A (channels 1 and 2):
	FTOA2	67		Output pins for the output compare A function of the free-running timer channels 1 and 2.
	FTOB1	66	O	FRT Output Compare B (channels 1 and 2):
	FTOB2	68		Output pins for the output compare B function of the free-running timer channels 1 and 2.
	FTCl1	61	I	FRT Counter Clock Input (channels 1 and 2):
	FTCl2	63		External clock input pins for the free-running counters (FRCs) of free-running timer channels 1 and 2.
	FTI1	60	I	FRT Input Capture (channels 1 and 2):
	FTI2	62		Input capture pins for free-running timer channels 1 and 2.
8-Bit timer	TMO	59	O	8-bit Timer Output: Compare-match output pin for the 8-bit timer.
	TMCI	57	I	8-bit Timer Clock Input: External clock input pin for the 8-bit timer counter.
	TMRI	62	I	8-bit Timer Counter Reset Input: High input at this pin resets the 8-bit timer counter.
Serial communication interface	TXD1	94	O	Transmit Data: Data output pins for the serial communication interface.
	TXD2	96		
	RXD1	93	I	Receive Data: Data input pins for the serial communication interface.
	RXD2	95		
	SCK1	91	I/O	Serial Clock: Input/output pins for the serial interface clock.
A/D converter	SCK2	92		
	AN3 – AN0	86 – 83	I	Analog Input: Analog signal input pins.
	AVcc	87	I	Analog Reference Voltage: Reference voltage pin for the A/D converter. If not used, connect to Vcc.
	AVss	82	I	Analog Ground: Ground pin for the A/D converter. If not used, connect to Vss.
	ADTRG	56	I	A/D Trigger: External trigger input pin for the A/D converter.

Table 1-3 Pin Functions (cont)

Type	Symbol	Pin No.	I/O	Name and Function
I/O ports	P17 – P10	11 – 4	I/O	Port 1: An 8-bit input/output port. The direction of each bit is determined by the port 1 data direction register (P1DDR).
	P27 – P20	45 – 38	I/O	Port 2: A 8-bit input/output port. The direction of each bit is determined by the port 2 data direction register (P2DDR).
	P37 – P30	54 – 47	I/O	Port 3: An 8-bit input/output port. The direction of each bit is determined by the port 3 data direction register (P3DDR).
	P47 – P40	63 – 56	I/O	Port 4: An 8-bit input/output port with Schmitt inputs. The direction of each bit is determined by the port 4 data direction register (P4DDR).
	P57 – P50	72 – 65	I/O	Port 5: An 8-bit input/output port. The direction of each bit is determined by the port 5 data direction register (P5DDR).
	P67 – P60	80 – 73	I/O	Port 6: An 8-bit input/output port. The direction of each bit is determined by the port 6 data direction register (P6DDR).
	P73 – P70	86 – 83	I	Port 7: A 4-bit input port.
	P87 – P80	96 – 89	I/O	Port 8: An 8-bit input/output port. The direction of each bit is determined by the port 8 data direction register (P8DDR).

Section 2 MCU Operating Modes and Address Space

2.1 Overview

2.1.1 Selection of MCU Mode

The H8/510 microcomputer unit (MCU) operates in four modes numbered 1, 2, 3, and 4. The mode is selected by the inputs at the mode pins (MD2 to MD0).

Table 2-1 Operating Modes

MCU Mode	MD2	MD1	MD0	Description	CPU Mode	Data Bus Width
Mode 0	Low	Low	Low	—	—	—
Mode 1	Low	Low	High	Expanded minimum mode	Minimum mode	8 Bits
Mode 2	Low	High	Low	Expanded minimum mode	Minimum mode	16 Bits
Mode 3	Low	High	High	Expanded maximum mode	Maximum mode	8 Bits
Mode 4	High	Low	Low	Expanded maximum mode	Maximum mode	16 Bits
Mode 5	High	Low	High	—	—	—
Mode 6	High	High	Low	—	—	—
Mode 7	High	High	High	—	—	—

Note: Modes marked with dashes (—) cannot be used.

The expanded minimum modes (modes 1 and 2) support a maximum address space of 64 kbytes. The expanded maximum modes (modes 3 and 4) support a maximum address space of 16 Mbytes. The H8/510 does not support modes 0, 5, 6, and 7. The mode pins should never be set to these values.

The MCU mode determines the size of the address space and the usage of I/O pins.

2.1.2 Register Control of MCU Mode

The MCU operating mode is monitored by the mode control register (MDCR) described in table 2-2.

Table 2-2 Mode Control Register

Name	Abbreviation	Read/Write	Address
Mode control register	MDCR	R	H'FF19

2.2 Mode Control Register (MDCR)

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	MDS2	MDS1	MDS0
Initial value	1	1	0	0	0	*	*	*
Read/Write	—	—	—	—	—	R	R	R

* Determined by MD2 to MD0 inputs.

The MDCR bits are set by the inputs at the mode pins (MD2 to MD0).

MDCR is an 8-bit register that is used to monitor the current operating mode of the H8/510. The MDCR bits can be read but not written.

Bits 7 and 6—Reserved: These bits cannot be modified and are always read as 1.

Bits 5 to 3—Reserved: These bits cannot be modified and are always read as 0.

Bits 2 to 0—Mode Select 2 to 0 (MDS2 to MDS0): These bits indicate the values of the mode pins (MD2 to MD0) thus indicating the current MCU mode. MDS2 corresponds to MD2, MDS1 to MD1, and MDS0 to MD0.

2.3 Mode Descriptions

Mode 1 (Expanded Minimum Mode): Mode 1 supports a maximum 64-kbyte address space which is accessed via an 8-bit data bus. The byte area register is ignored. (See section 15, “Bus Controller,” for details of the byte area register).

Mode 2 (Expanded Minimum Mode): Mode 2 supports a maximum 64-kbyte address space that is accessed via a 16-bit data bus. Part of the address space, designated by the byte area register, is accessed via an 8-bit data bus.

Port 1 is used as part of the data bus.

Mode 3 (Expanded Maximum Mode): Mode 3 supports a maximum 16-Mbyte address space that is accessed via an 8-bit bus. The byte area register is ignored.

Port 2 is used as part of the address bus.

Mode 4 (Expanded Maximum Mode): Mode 4 supports a maximum 16-Mbyte address space

that is accessed via a 16-bit data bus. Part of the address space, designated by the byte area register, is accessed via an 8-bit data bus.

Port 1 is used as part of the data bus. Port 2 is used as part of the address bus.

2.4 Pin Functions in Each MCU Mode

The functions of the I/O ports depend on the MCU mode. Table 2-3 lists the pin functions in modes 1 to 4.

For a more detailed description of the control of pin functions, see section 9, “I/O Ports.”

Table 2-3 Pin Functions in Each MCU Mode

		MCU Mode			
		Expanded Minimum Modes		Expanded Maximum Modes	
Port		Mode 1	Mode 2	Mode 3	Mode 4
Port 1		I/O port	Data bus (D7 – D0)	I/O port	Data bus (D7 – D0)
Port 2		I/O port	I/O port	Address bus (A23 – A16)	Address bus (A23 – A16)
Port 3	P37	I/O port	I/O port	I/O port	I/O port
	P36	I/O port	I/O port	I/O port	I/O port
	P35	I/O port	I/O port	I/O port	I/O port
	P34	I/O port	I/O port	I/O port	I/O port
	P33	I/O port	I/O port	I/O port	I/O port
	P32	I/O port/BREQ	I/O port/BREQ	I/O port/BREQ	I/O port/BREQ
	P31	I/O port/BACK	I/O port/BACK	I/O port/BACK	I/O port/BACK
	P30	I/O port/WAIT	I/O port/WAIT	I/O port/WAIT	I/O port/WAIT
Port 4		I/O port*	I/O port*	I/O port*	I/O port*
Port 5	P57	I/O port	I/O port	I/O port	I/O port
	P56	I/O port	I/O port	I/O port	I/O port
	P55	I/O port	I/O port	I/O port	I/O port
	P54	I/O port	I/O port	I/O port	I/O port
	P53	I/O port*	I/O port*	I/O port*	I/O port*
	P52	I/O port*	I/O port*	I/O port*	I/O port*
	P51	I/O port*	I/O port*	I/O port*	I/O port*
	P50	I/O port*	I/O port*	I/O port*	I/O port*
Port 6		I/O port	I/O port	I/O port	I/O port
Port 7		Input port*	Input port*	Input port*	Input port*
Port 8		I/O port*	I/O port*	I/O port*	I/O port*

* Also used as input/output pins for on-chip supporting modules.

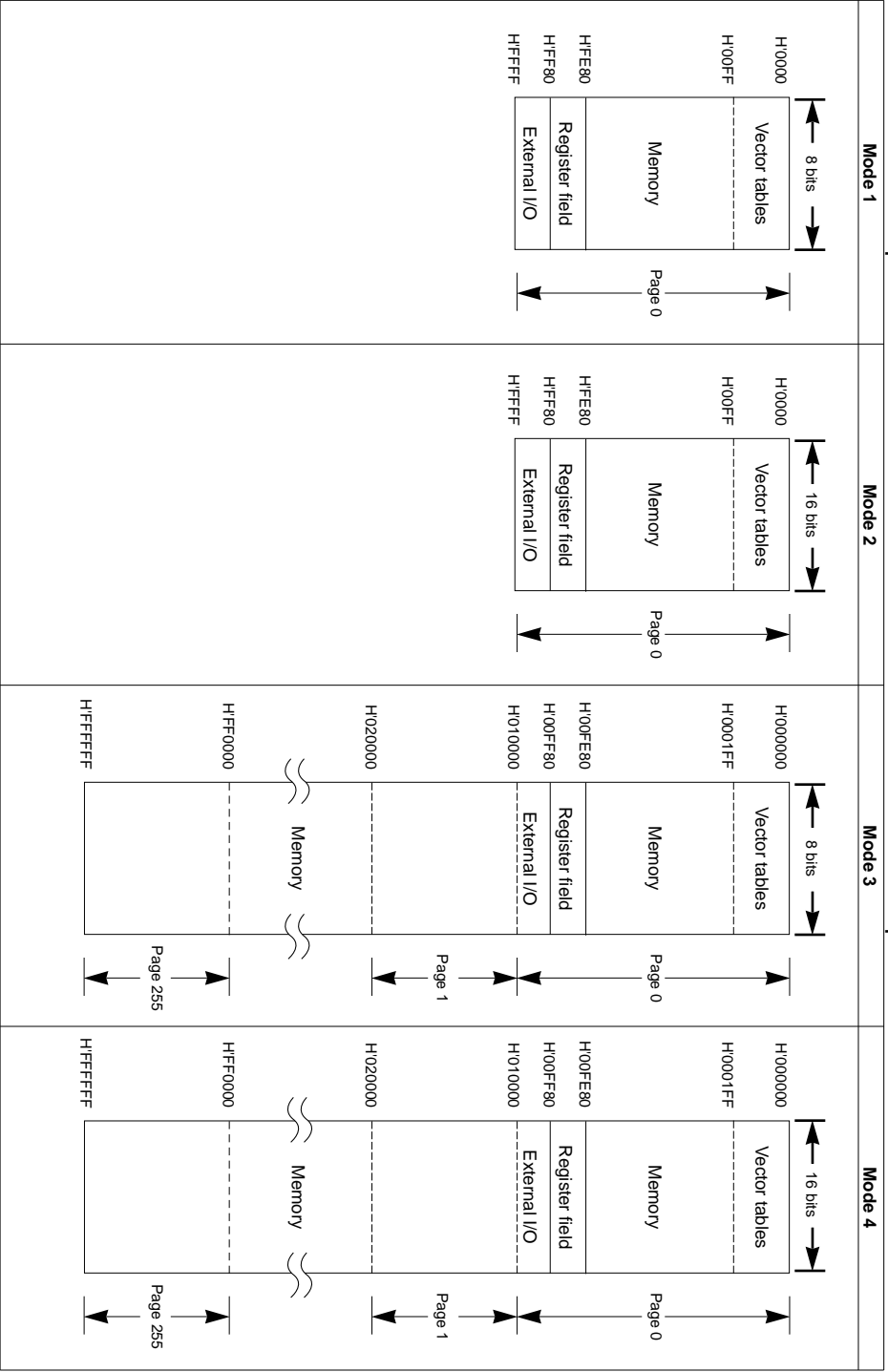


Figure 2-1 Memory Map in Each MCU Mode

Section 3 CPU

3.1 Overview

The H8/510 chip has the H8/500 Family CPU: a high-speed central processing unit designed for realtime control of a wide range of medium-scale office and industrial equipment. It features eight 16-bit general registers, internal 16-bit data paths, and an optimized instruction set.

Section 3 summarizes the CPU architecture and instruction set.

3.1.1 Features

The main features of the H8/500 CPU are listed below.

- General-register machine
 - Eight 16-bit general registers
 - Seven control registers (two 16-bit registers, five 8-bit registers)
- High speed: maximum 10 MHz
 - At 10 MHz a register-register add operation takes only 200 ns.
- Address space managed in 64-kbyte pages, expandable to 16 Mbytes
 - Page registers make four pages available simultaneously: a code page, stack page, data page, and extended page.
- Two CPU operating modes:
 - Minimum mode: Maximum 64-kbyte address space
 - Maximum mode: Maximum 16-Mbyte address space
- Highly orthogonal instruction set
 - Addressing modes and data sizes can be specified independently within each instruction.
- Addressing modes
 - Register-register and register-memory operations are supported.
- Optimized for efficient programming in C language
 - In addition to the general registers and orthogonal instruction set, the CPU has special short formats for frequently-used instructions and addressing modes.

3.1.2 Address Space

The CPU has two operating modes as shown in figure 3-1. The CPU operating mode is selected by the input to the mode pins (MD2 to MD0).

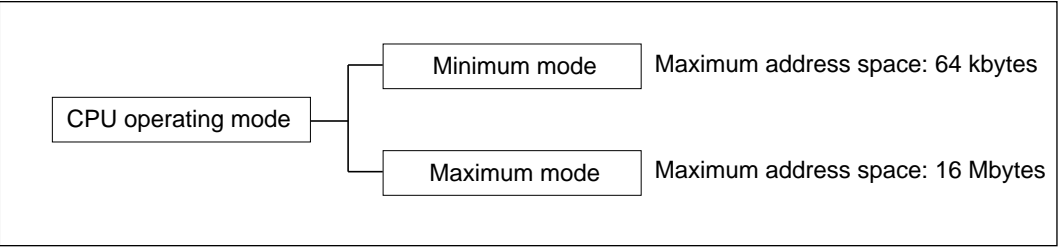


Figure 3-1 CPU Operating Modes

Figure 3-2 compares the memory maps of these two modes.

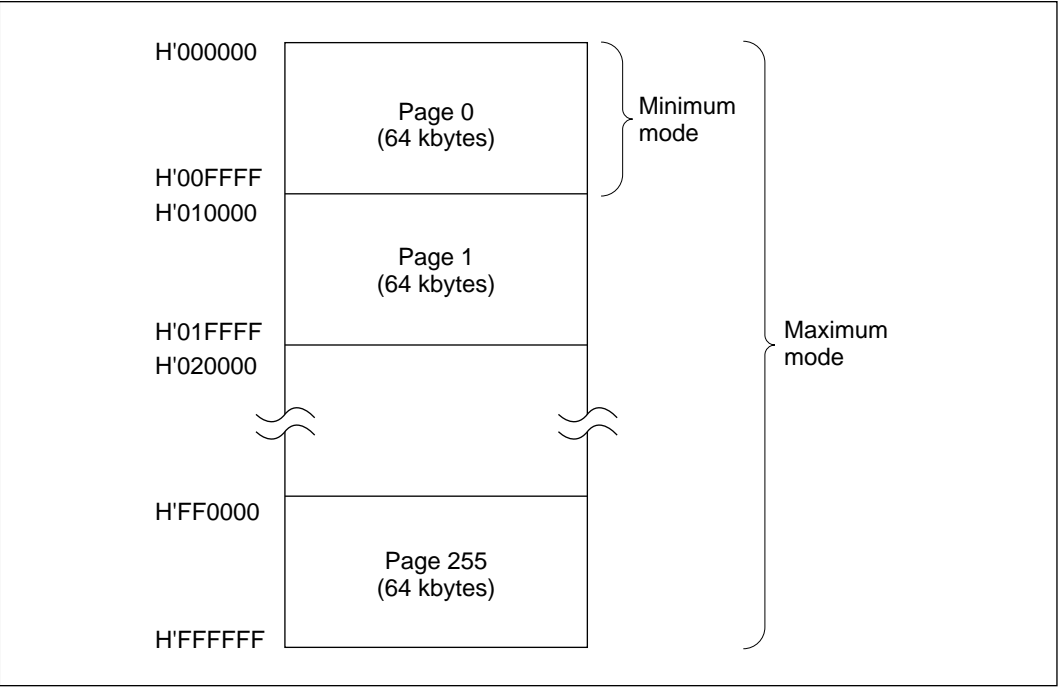


Figure 3-2 Memory Map

3.1.3 Register Configuration

Figure 3-3 shows the register structure of the CPU. There are two groups of registers: the general registers (Rn) and control registers (CR).

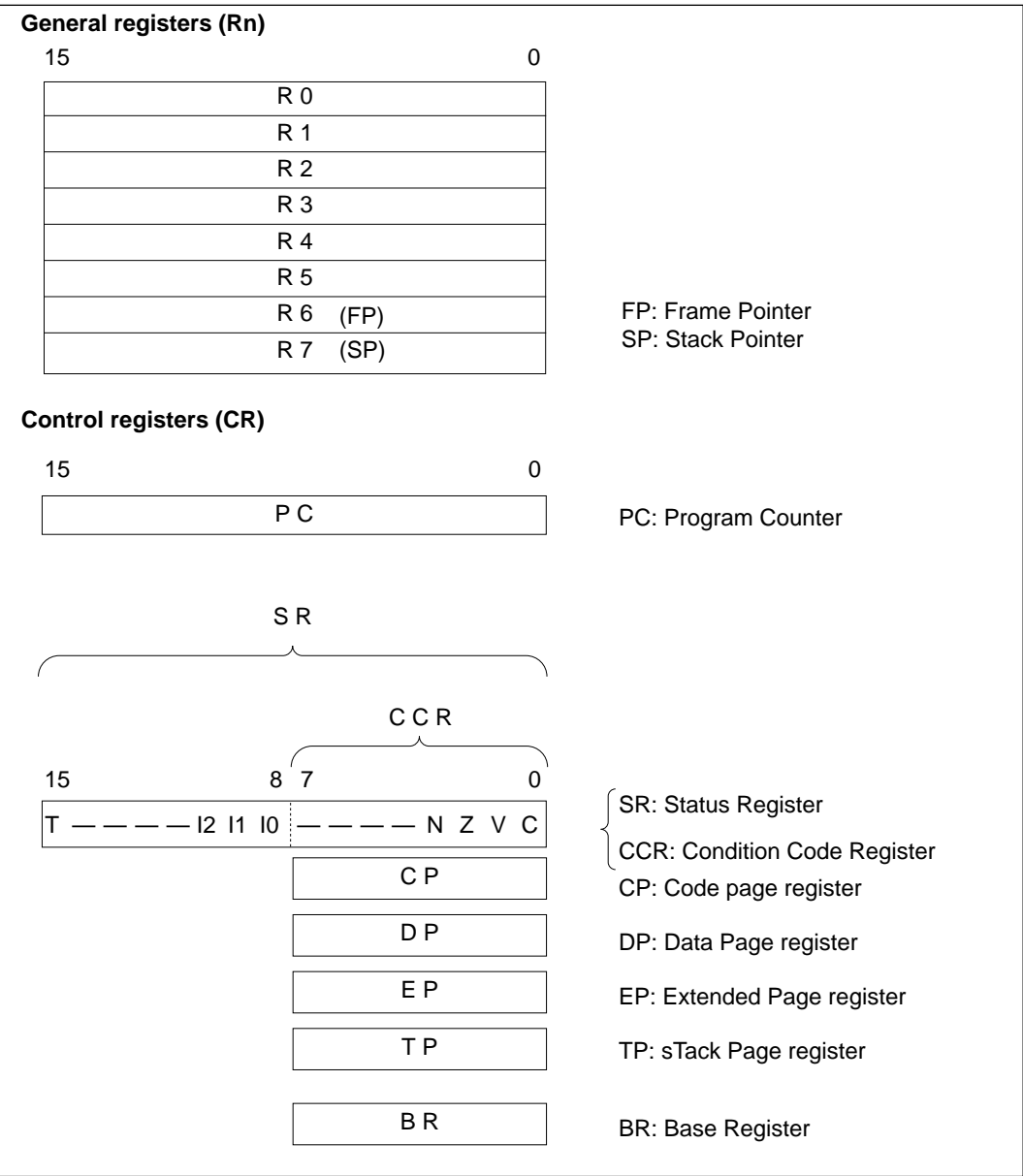


Figure 3-3 Registers in the CPU

3.2 CPU Register Descriptions

3.2.1 General Registers

All eight of the 16-bit general registers are functionally alike; there is no distinction between data registers and address registers. When these registers are accessed as data registers, either byte or word size can be selected.

R6 and R7, in addition to functioning as general registers, have special assignments.

R7 is the stack pointer, used implicitly in exception handling and subroutine calls. It can be designated by the name SP, which is synonymous with R7. As indicated in figure 3-4, it points to the top of the stack. It is also used implicitly by the LDM and STM instructions, which load and store multiple registers from and to the stack and pre-decrement or post-increment R7 accordingly.

R6 functions as a frame pointer (FP). The LINK and UNLK instructions use R6 implicitly to reserve or release a stack frame.

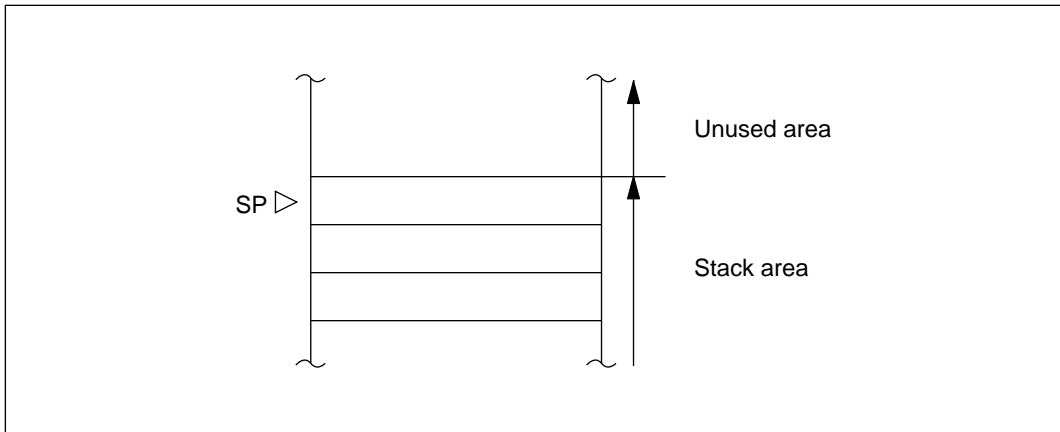


Figure 3-4 Stack Pointer

3.2.2 Control Registers

The CPU control registers (CR) include a 16-bit program counter (PC), a 16-bit status register (SR), four 8-bit page registers, and one 8-bit base register (BR).

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 contains internal status information. The lower half of the status register is referred to as the condition code register (CCR): it can be accessed as a separate condition code byte.

									CCR							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	T	—	—	—	—	I ₂	I ₁	I ₀	—	—	—	—	N	Z	V	C

Bit 15—Trace (T): When this bit is set to 1, the CPU operates in trace mode and generates a trace exception after every instruction. See section 4.4, “Trace” for a description of the trace exception-handling sequence.

When the value of this bit is 0, instructions are executed in normal continuous sequence. This bit is cleared to 0 at a reset.

Bits 14 to 11—Reserved: These bits cannot be modified and are always read as 0.

Bits 10 to 8—Interrupt Mask (I2, I1, I0): These bits indicate the interrupt request mask level (7 to 0). As shown in table 3-1, an interrupt request is not accepted unless it has a higher level than the value of the mask. A nonmaskable interrupt (NMI) is accepted at any mask level. After an interrupt is accepted, I2, I1, and I0 are changed to the level of the interrupt. Table 3-2 indicates the values of the I bits after an interrupt is accepted.

A reset sets all three of these bits (I2, I1, and I0) to 1, masking all interrupts except NMI.

Table 3-1 Interrupt Mask Levels

Priority	Mask Level	Mask Bits			Interrupts Accepted
		I2	I1	I0	
↑ High ↓ Low	7	1	1	1	NMI
	6	1	1	0	Level 7 and NMI
	5	1	0	1	Levels 7 to 6 and NMI
	4	1	0	0	Levels 7 to 5 and NMI
	3	0	1	1	Levels 7 to 4 and NMI
	2	0	1	0	Levels 7 to 3 and NMI
	1	0	0	1	Levels 7 to 2 and NMI
	0	0	0	0	Levels 7 to 1 and NMI

Table 3-2 Interrupt Mask Bits after an Interrupt is Accepted

Level of Interrupt Accepted	I2	I1	I0
NMI	1	1	1
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1

Bits 7 to 4—Reserved: These bits cannot be modified and are always read as 0.

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

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

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

Bit 0—Carry (C): This bit is set to 1 when a carry or borrow occurs at the most significant bit, and is cleared to 0 (or left unchanged) at other times.

The specific changes that occur in the condition code bits when each instruction is executed are listed in appendix A.1, “Instruction Tables.” See the *H8/500 Series Programming Manual* for further details.

Page Registers: The code page register (CP), data page register (DP), extended page register (EP), and stack page register (TP) are 8-bit registers that are used only in the maximum mode. No use of their contents is made in the minimum mode.

In the maximum mode, the page registers combine with the program counter or general registers to generate 24-bit effective addresses as shown in figure 3-5, thereby expanding the program area, data area, and stack area.

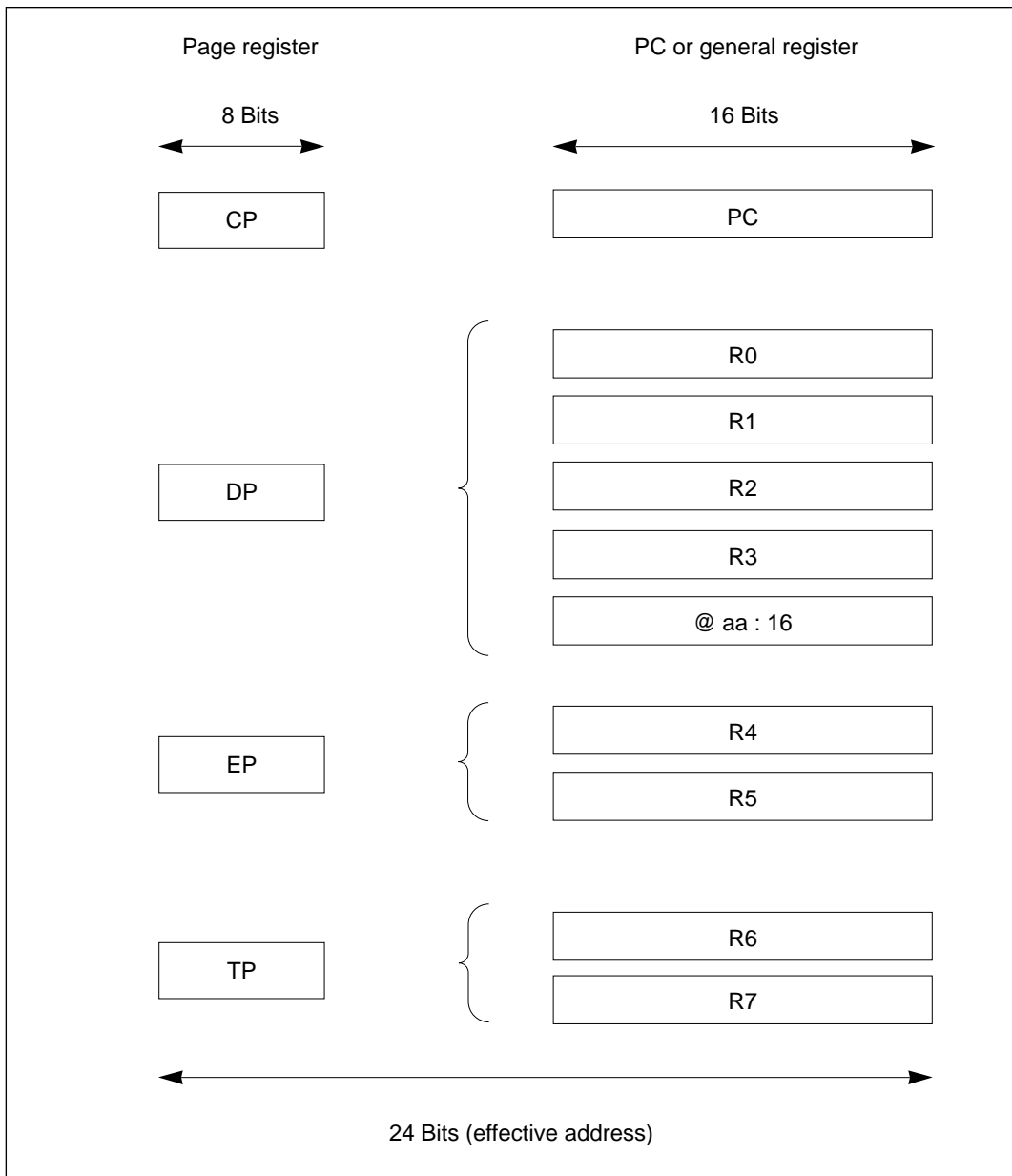


Figure 3-5 Combinations of Page Registers with Other Registers

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, the code page register is initialized at a reset to a value loaded from the vector table, and 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 R3 to R0 to generate a 24-bit effective address. The data page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R3 to R0, and in the 16-bit absolute addressing mode (@aa:16).

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 operand address. The extended page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R4 or R5.

The extended page can be used as an additional data page.

Stack Page Register (TP): The stack page register combines with R6 (FP) or R7 (SP) to generate a 24-bit stack address. The stack page register contains the upper 8 bits of the address. It is used to calculate effective addresses in the register indirect addressing mode using R6 or R7, in exception handling, and subroutine calls.

Base Register (BR): This 8-bit register stores the base address used in the short absolute addressing mode (@aa:8). In this addressing mode a 16-bit effective address in page 0 is generated by using the contents of the base register as the upper 8 bits and an address given in the instruction code as the lower 8 bits. See figure 3-6.

In the short absolute addressing mode the address is always located in page 0.

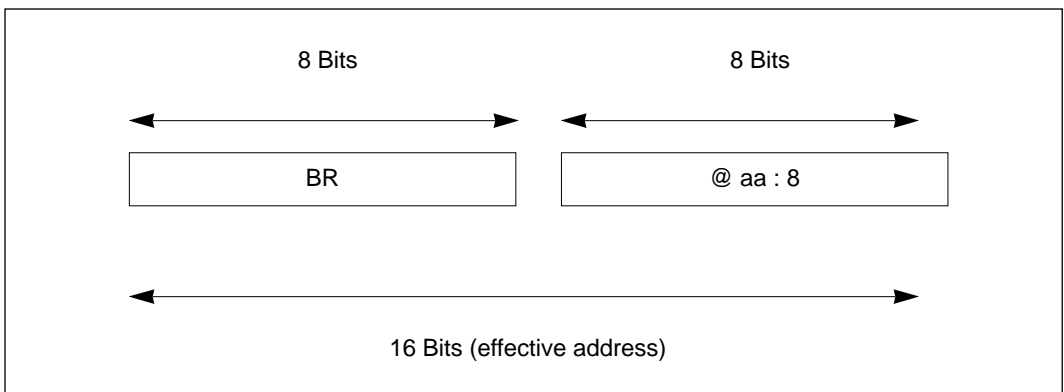


Figure 3-6 Short Absolute Addressing Mode and Base Register

3.2.3 Initial Register Values

When the CPU is reset, its internal registers are initialized as shown in table 3-3. Note that the stack pointer (R7) and base register (BR) are not initialized to fixed values. Also, of the page registers used in maximum mode, only the code page register (CP) is initialized; the other three page registers come out of the reset state with undetermined values.

Accordingly, in the minimum mode the first instruction executed after a reset should initialize the stack pointer. The base register must also be initialized before the short absolute addressing mode (@aa:8) is used.

In the maximum mode, the first instruction executed after a reset should initialize the stack page register (TP) and the next instruction should initialize the stack pointer. Later instructions should initialize the base register and the other page registers as necessary.

Table 3-3 Initial Values of Registers

Register	Initial Value	
	Minimum Mode	Maximum Mode
General registers		
15 0 R7 – R0	Undetermined	Undetermined
Control registers		
15 0 PC	Loaded from vector table	Loaded from vector table
SR		
CCR		
15 8 7 0 T---- I2I1I0 ---- NZVC	H'070x (x: undetermined)	H'070x (x: undetermined)
7 0 CP	Undetermined	Loaded from vector table
7 0 DP	Undetermined	Undetermined
7 0 EP	Undetermined	Undetermined
7 0 TP	Undetermined	Undetermined
7 0 BR	Undetermined	Undetermined

3.3 Data Formats

The H8/500 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.
- Almost all instructions operate on byte and word data.
- Multiply and divide instructions operate on longword data.

3.3.1 Data Formats in General Registers

Data of all the sizes above can be stored in general registers as shown in table 3-4.

Bit data locations are specified by bit number. Bit 15 is the most significant bit. Bit 0 is the least significant bit. BCD and byte data are stored in the lower 8 bits of a general register. Word data use all 16 bits of a general register. Longword data use two general registers: the upper 16 bits are stored in Rn (n must be an even number); the lower 16 bits are stored in Rn+1.

Operations performed on BCD data or byte data do not affect the upper 8 bits of the register.

Table 3-4 General Register Data Formats

Data Type	Register No.	Data Structure
1-Bit	Rn	<div> <div>15</div> <div>0</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>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>
BCD	Rn	<div> <div>15</div> <div>8</div> <div>7</div> <div>4</div> <div>3</div> <div>0</div> <div> <div>Don't care</div> <div>Upper digit</div> <div>Lower digit</div> </div> </div>
Byte	Rn	<div> <div>15</div> <div>8</div> <div>7</div> <div>0</div> <div> <div>Don't care</div> <div>MSB</div> <div>LSB</div> </div> </div>
Word	Rn	<div> <div>15</div> <div>0</div> <div> <div>MSB</div> <div>LSB</div> </div> </div>
Longword	Rn* Rn+1*	<div> <div>31</div> <div>16</div> <div> <div>MSB</div> <div>Upper 16 bits</div> <div>Lower 16 bits</div> <div>LSB</div> </div> </div> <div> <div>15</div> <div>0</div> </div>

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

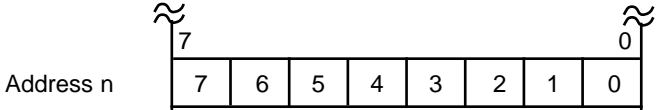
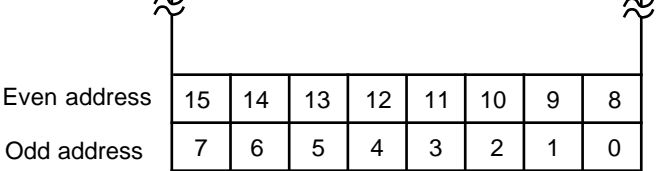
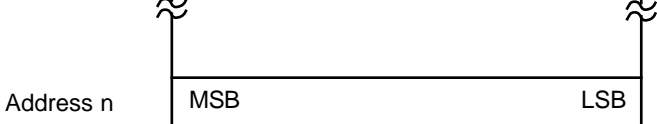
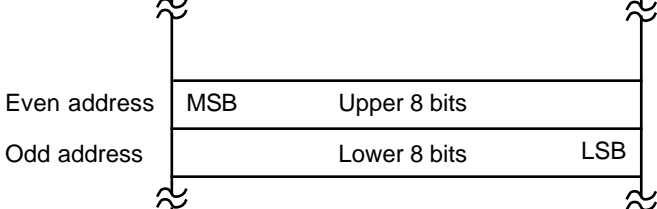
3.3.2 Data Formats in Memory

Table 3-5 indicates the data formats in memory.

Instructions that access bit data in memory have byte or word operands. The instruction specifies a bit number to indicate a specific bit in the operand.

Access to word data in memory must always begin at an even address. Access to word data starting at an odd address causes an address error. The upper 8 bits of word data are stored in address n (where n is an even number); the lower 8 bits are stored in address n+1.

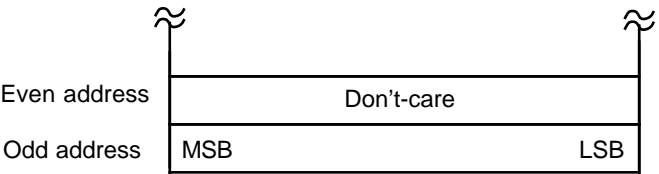
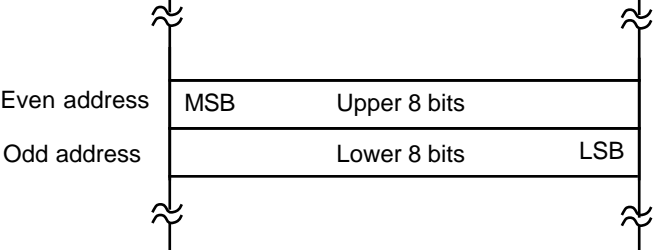
Table 3-5 Data Formats in Memory

Data Type	Data Format
1-Bit (in byte operand data)	
1-Bit (in word operand data)	
Byte	
Word	

When the stack is accessed in exception processing (to save or restore the program counter, code page register, or status register), word access is always performed, regardless of the actual data size. Similarly, when the stack is accessed by an instruction using the pre-decrement or post-increment register indirect addressing mode specifying R7 (@-R7 or @R7+), which is the stack pointer, word access is performed regardless of the operand size specified in the instruction. An address error will therefore occur if the stack pointer indicates an odd address. Programs should be coded so that the stack pointer always indicates an even address.

Table 3-6 shows the data formats on the stack.

Table 3-6 Data Formats on the Stack

Data Type	Data Format
Byte data on stack	 <p>Even address: Don't-care</p> <p>Odd address: MSB, LSB</p>
Word data on stack	 <p>Even address: MSB, Upper 8 bits</p> <p>Odd address: Lower 8 bits, LSB</p>

3.4 Instructions

3.4.1 Basic Instruction Formats

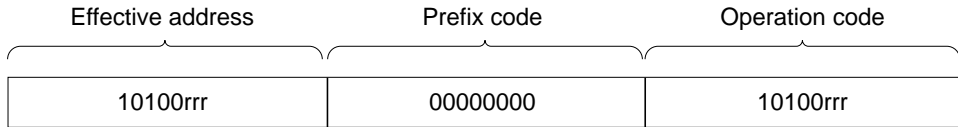
There are two basic CPU instruction formats: the general format and the special format.

General Format: This format consists of an effective address (EA) field, an effective address extension field, and an operation code (OP) field. The effective address is placed before the operation code because this results in faster execution of the instruction.

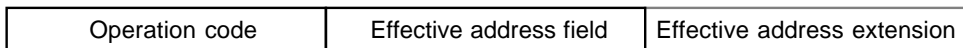
Effective address field	Effective address extension	Operation code
-------------------------	-----------------------------	----------------

- **Effective address field:** One byte containing information used to calculate the effective address of an operand.
- **Effective address extension:** Zero to two bytes containing a displacement value, immediate data, or an absolute address. The size of the effective address extension is specified in the effective address field.
- **Operation code:** Defines the operation to be carried out on the operand located at the address calculated from the effective address information. Some instructions (DADD, DSUB, MOVFPE, MOVTPE) have an extended format in which the operand code is preceded by a one-byte prefix code.

- (Example of prefix code in DADD instruction)



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 is specified before the operand.



- Operation code: One or two bytes defining 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.

3.4.2 Addressing Modes

The CPU supports 7 addressing modes: (1) register direct; (2) register indirect; (3) register indirect with displacement; (4) register indirect with pre-decrement or post-increment; (5) immediate; (6) absolute; and (7) PC-relative.

Due to the highly orthogonal nature of the instruction set, most instructions having operands can use any applicable addressing mode from (1) through (6). The PC-relative mode (7) is used by branching instructions.

In most instructions, the addressing mode is specified in the effective address field. The effective-address extension, if present, contains a displacement, immediate data, or an absolute address.

Table 3-7 indicates how the addressing mode is specified in the effective address field.

Table 3-7 Addressing Modes

No.	Addressing Mode	Mnemonic	EA Field	EA Extension
1	Register direct	Rn	<div>1 0 1 0 Sz r r r</div> <div>*1 *2</div>	None
2	Register indirect	@Rn	<div>1 1 0 1 Sz r r r</div>	None
3	Register indirect with displacement	@(d:8, Rn)	<div>1 1 1 0 Sz r r r</div>	Displacement (1 byte)
		@(d:16, Rn)	<div>1 1 1 1 Sz r r r</div>	Displacement (2 bytes)
4	Register indirect with pre-decrement	@-Rn	<div>1 0 1 1 Sz r r r</div>	None
	Register indirect with post-increment	@Rn+	<div>1 1 0 0 Sz r r r</div>	
5	Immediate	#xx: 8	<div>0 0 0 0 0 1 0 0</div>	Immediate data (1 byte)
		#xx: 16	<div>0 0 0 0 1 1 0 0</div>	Immediate data (2 bytes)
6	Absolute *3	@aa: 8	<div>0 0 0 0 Sz 1 0 1</div>	1-byte absolute address (offset from BR)
		@aa: 16	<div>0 0 0 1 Sz 1 0 1</div>	2-byte absolute address
7	PC-relative	disp	No EA field. Addressing mode is specified in the operation code.	1- or 2-byte displacement

Notes: * 1 Sz: Specifies the operand size.

When Sz = 0: byte operand

When Sz = 1: word operand

* 2 rrr: Register number field, specifying a general register number.

0 0 0 — R0 0 0 1 — R1 0 1 0 — R2 0 1 1 — R3

1 0 0 — R4 1 0 1 — R5 1 1 0 — R6 1 1 1 — R7

* 3 The @aa: 8 addressing mode is also referred to as the short absolute addressing mode.

3.4.3 Effective Address Calculation

Table 3-8 explains how the effective address is calculated in each addressing mode.

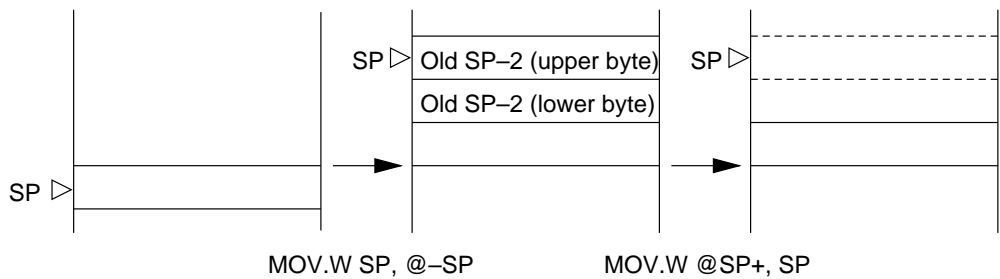
Table 3-8 Effective Address Calculation

No.	Addressing Mode	Effective Address Calculation	Effective Address
1	Register direct Rn <div>1010Sz rrr</div>	—	Operand is contents of Rn
2	Register indirect @Rn <div>1101Sz rrr</div>	—	<div> <div>23 15 0</div> <div>DP *1 Rn</div> </div> Or TP or EP *2
3	Register indirect with displacement @(d:8,Rn) <div>1110Sz rrr</div>	8 Bits <div> <div>15 0</div> <div>Rn</div> </div> <div> <div>15 0</div> <div>Displacement with sign extension</div> </div> <div>+</div>	<div> <div>23 15 0</div> <div>DP *1 Result</div> </div> Or TP or EP *2
	@(d:16,Rn) <div>1111Sz rrr</div>	16 Bits <div> <div>15 0</div> <div>Rn</div> </div> <div> <div>15 0</div> <div>Displacement</div> </div> <div>+</div>	<div> <div>23 15 0</div> <div>DP *1 Result</div> </div> Or TP or EP *2
4	Register indirect with pre-decrement @-Rn <div>1011Sz rrr</div>	<div> <div>15 0</div> <div>Rn</div> </div> <div> <div>1 or 2</div> <div>⊖</div> </div> Rn is decremented by -1 or -2 before instruction execution. *3*4*5	<div> <div>23 15 0</div> <div>DP *1 Result</div> </div> Or TP or EP *2
	Register indirect with post-increment @Rn+ <div>1100Sz rrr</div>	— Rn is incremented by +1 or +2 after instruction execution. *3*4*5	<div> <div>23 15 0</div> <div>DP *1 Rn</div> </div> Or TP or EP *2

Table 3-8 Effective Address Calculation (cont)

No.	Addressing Mode	Effective Address Calculation	Effective Address
5	Absolute address @aa:8	—	23 15 0 <div> <div>H'00</div> <div>BR</div> <div>EA extension data</div> </div>
	@aa:16	—	23 15 0 <div> <div>DP</div> <div>EA extension data</div> </div>
6	Immediate #xx:8	—	Operand is 1-byte EA extension data.
	#xx:16	—	Operand is 2-byte EA extension data.
7	PC-relative disp:8 No EA code Specified in OP code	8 bits <div> <div>15 0</div> <div>PC</div> <div>15 0</div> <div>Displacement with sign extension</div> <div>⊕</div> </div>	23 15 0 <div> <div>CP *1</div> <div>Result</div> </div>
	disp:16 No EA code Specified in OP code	16 bits <div> <div>15 0</div> <div>PC</div> <div>15 0</div> <div>Displacement</div> <div>⊕</div> </div>	23 15 0 <div> <div>CP *1</div> <div>Result</div> </div>

- Notes:**
- * 1 The page register is ignored in minimum mode.
 - * 2 The page register used in addressing modes 2, 3, and 4 depends on the general register:
DP for R0, R1, R2, or R3; EP for R4 or R5; TP for R6 or R7.
 - * 3 Decrement by -1 for a byte operand, and by -2 for a word operand.
 - * 4 The pre-decrement or post-increment is always ± 2 when R7 is specified, even if the operand is byte size.
 - * 5 The drawing below shows what happens when the @-SP and @ SP+ addressing modes are used to save and restore the stack pointer.



3.5 Instruction Set

3.5.1 Overview

The main features of the CPU instruction set are:

- A general-register architecture.
- Orthogonality. Addressing modes and data sizes can be specified independently in each instruction.
- Addressing modes supporting register-register and register-memory operations.
- Affinity for high-level languages, particularly C, with short formats for frequently-used instructions and addressing modes.

The CPU instruction set includes 63 types of instructions, listed by function in table 3-9.

Table 3-9 Instruction Classification

Function	Instructions	Types
Data transfer	MOV, LDM, STM, XCH, SWAP, MOVTPE, MOVFPE	7
Arithmetic operations	ADD, SUB, ADDS, SUBS, ADDX, SUBX, DADD, DSUB, MULXU, DIVXU, CMP, EXTS, EXTU, TST, NEG, CLR, TAS	17
Logic operations	AND, OR, XOR, NOT	4
Shift	SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR	8
Bit manipulation	BSET, BCLR, BTST, BNOT	4
Branch	Bcc*, JMP, PJMP, BSR, JSR, PJSR, RTS, PRTD, PRTS, RTD, SCB (/F, /NE, /EQ)	11
System control	TRAPA, TRAP/VS, RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP, LINK, UNLK	12
Total		63

* Bcc is a conditional branch instruction in which cc represents a condition code.

Tables 3-10 to 3-16 give a concise summary of the instructions in each functional category. The MOV, ADD, and CMP instructions have special short formats, which are listed in table 3-17. For detailed descriptions of the instructions, refer to the *H8/500 Series Programming Manual*.

The notation used in tables 3-10 to 3-17 is defined below.

Operation Notation

Rd	General register (destination)
Rs	General register (source)
Rn	General register
(EAd)	Destination operand
(EAs)	Source operand
CCR	Condition code register
N	N (negative) bit of CCR
Z	Z (zero) bit of CCR
V	V (overflow) bit of CCR
C	C (carry) bit of CCR
CR	Control register
PC	Program counter
CP	Code page register
SP	Stack pointer
FP	Frame pointer
#IMM	Immediate data
disp	Displacement
+	Addition
−	Subtraction
×	Multiplication
÷	Division
^	AND logical
∨	OR logical
⊕	Exclusive OR logical
→	Move
↔	Exchange
¬	Not

3.5.2 Data Transfer Instructions

Table 3-10 describes the seven data transfer instructions.

Table 3-10 Data Transfer Instructions

Instruction		Size*	Function
Data transfer	MOV		(EAs) → (EAd), #IMM → (EAd)
	MOV:G	B/W	Moves data between two general registers, or between a general register and memory, or moves immediate data to a general register or memory.
	MOV:E	B	
	MOV:I	W	
	MOV:F	B/W	
	MOV:L	B/W	
	MOV:S	B/W	
	LDM	W	Stack → Rn (register list) Pops data from the stack to one or more registers.
	STM	W	Rn (register list) → stack Pushes data from one or more registers onto the stack.
	XCH	W	Rs ↔ Rd Exchanges data between two general registers.
	SWAP	B	Rd (upper byte) ↔ Rd (lower byte) Exchanges the upper and lower bytes in a general register.
	MOVTPE	B	Rn → (EAd) Transfers data from a general register to memory in synchronization with the E clock.
	MOVFPPE	B	(EAs) → Rd Transfers data from memory to a general register in synchronization with the E clock.

Note: B—byte; W—word

3.5.3 Arithmetic Instructions

Table 3-11 describes the 17 arithmetic instructions.

Table 3-11 Arithmetic Instructions

Instruction	Size	Function
Arithmetic operations		
ADD		$Rd \pm (EAs) \rightarrow Rd, (EAd) \pm \#IMM \rightarrow (EAd)$
└ ADD:G	B/W	Performs addition or subtraction on data in a general register and data in another general register or memory, or on immediate data and data in a general register or memory.
└ ADD:Q	B/W	
SUB	B/W	
ADDS	B/W	
SUBS	B/W	
ADDX	B/W	$Rd \pm (EAs) \pm C \rightarrow Rd$
SUBX	B/W	Performs addition or subtraction with carry or borrow on data in a general register and data in another general register or memory, or on immediate data and data in a general register or memory.
DADD	B	$(Rd)_{10} \pm (Rs)_{10} \pm C \rightarrow (Rd)_{10}$
DSUB	B	Performs decimal addition or subtraction on data in two general registers.
MULXU	B/W	$Rd \times (EAs) \rightarrow Rd$ Performs 8-bit \times 8-bit or 16-bit \times 16-bit unsigned multiplication on data in a general register and data in another general register or memory, or on data in a general register and immediate data.
DIVXU	B/W	$Rd \div (EAs) \rightarrow Rd$ Performs 16-bit \div 8-bit or 32-bit \div 16-bit unsigned division on data in a general register and data in another general register or memory, or on data in a general register and immediate data.
CMP		$Rn - (EAs), (EAd) - \#IMM$
└ CMP:G	B/W	Compares data in a general register with data in another general register or memory, or with immediate data, or compares immediate data with data in memory.
└ CMP:E	B	
└ CMP:I	W	

Note: B—byte; W—word

Table 3-11 Arithmetic Instructions (cont)

Instruction		Size	Function
Arithmetic operations	EXTS	B	(<bit 7> of <Rd>) \rightarrow (<bits 15 to 8> of <Rd>) Converts byte data in a general register to word data by extending the sign bit.
	EXTU	B	$0 \rightarrow$ (<bits 15 to 8> of <Rd>) Converts byte data in a general register to word data by padding with zero bits.
	TST	B/W	(EAd) $- 0$ Compares general register or memory contents with 0.
	NEG	B/W	$0 - (\text{EAd}) \rightarrow (\text{EAd})$ Obtains the two's complement of general register or memory contents.
	CLR	B/W	$0 \rightarrow (\text{EAd})$ Clears general register or memory contents to 0.
	TAS	B	(EAd) $\leftarrow 0, (1)_2 \rightarrow$ (<bit 7> of <EAd>) Tests general register or memory contents, then sets the most significant bit (bit 7) to 1.

Note: B—byte; W—word

3.5.4 Logic Operations

Table 3-12 lists the four instructions that perform logic operations.

Table 3-12 Logic Operation Instructions

Instruction		Size	Function
Logical operations	AND	B/W	$\text{Rd} \wedge (\text{EAs}) \rightarrow \text{Rd}$ Performs a logical AND operation on a general register and another general register, memory, or immediate data.
	OR	B/W	$\text{Rd} \vee (\text{EAs}) \rightarrow \text{Rd}$ Performs a logical OR operation on a general register and another general register, memory, or immediate data.
	XOR	B/W	$\text{Rd} \oplus (\text{EAs}) \rightarrow \text{Rd}$ Performs a logical exclusive OR operation on a general register and another general register, memory, or immediate data.
	NOT	B/W	$\neg (\text{EAd}) \rightarrow (\text{EAd})$ Obtains the one's complement of general register or memory contents.

Note: B—byte; W—word

3.5.5 Shift Operations

Table 3-13 lists the eight shift instructions.

Table 3-13 Shift Instructions

Instruction		Size	Function
Shift operations	SHAL	B/W	(EAd) shift → (EAd)
	SHAR	B/W	Performs an arithmetic shift operation on general register or memory contents.
	SHLL	B/W	(EAd) shift → (EAd)
	SHLR	B/W	Performs a logical shift operation on general register or memory contents.
	ROTL	B/W	(EAd) rotate → (EAd)
	ROTR	B/W	Rotates general register or memory contents.
	ROTXL	B/W	(EAd) rotate through carry → (EAd)
	ROTXR	B/W	Rotates general register or memory contents through the C (carry) bit.

Note: B—byte; W—word

3.5.6 Bit Manipulations

Table 3-14 describes the four bit-manipulation instructions.

Table 3-14 Bit-Manipulation Instructions

Instruction		Size	Function
Bit manipulations	BSET	B/W	\neg (<bit-No.> of <EAd>) \rightarrow Z, 1 \rightarrow (<bit-No.> of <EAd>) Tests a specified bit in a general register or memory, then sets the bit to 1. The bit is specified by a bit number given in immediate data or a general register.
	BCLR	B/W	\neg (<bit-No.> of <EAd>) \rightarrow Z, 0 \rightarrow (<bit-No.> of <EAd>) Tests a specified bit in a general register or memory, then clears the bit to 0. The bit is specified by a bit number given in immediate data or a general register.
	BNOT	B/W	\neg (<bit-No.> of <EAd>) \rightarrow Z, \rightarrow (<bit-No.> of <EAd>) Tests a specified bit in a general register or memory, then inverts the bit. The bit is specified by a bit number given in immediate data or a general register.
	BTST	B/W	\neg (<bit-No.> of <EAd>) \rightarrow Z Tests a specified bit in a general register or memory. The bit is specified by a bit number given in immediate data or a general register.

Note: B—byte; W—word

3.5.7 Branching Instructions

Table 3-15 describes the 11 branching instructions.

Table 3-15 Branching Instructions

Instruction	Size	Function
Branch	Bcc	—
Branches if condition cc is true.		
Mnemonic	Description	Condition
BRA (BT)	Always (true)	True
BRN (BF)	Never (false)	False
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	$Z \vee (N \oplus V) = 1$
JMP	—	Branches unconditionally to a specified address in the same page.
PJMP	—	Branches unconditionally to a specified address in a specified page.
BSR	—	Branches to a subroutine at a specified address in the same page.
JSR	—	Branches to a subroutine at a specified address in the same page.
PJSR	—	Branches to a subroutine at a specified address in a specified page.
RTS	—	Returns from a subroutine in the same page.

Table 3-15 Branching Instructions (cont)

Instruction		Size	Function
Branch	PRTS	—	Returns from a subroutine in a different page.
	RTD	—	Returns from a subroutine in the same page and adjusts the stack pointer.
	PRTD	—	Returns from a subroutine in a different page and adjusts the stack pointer.
	SCB/F	—	Controls a loop using a loop counter and/or a specified termination condition.
	SCB/NE	—	
	SCB/EQ	—	

3.5.8 System Control Instructions

Table 3-16 describes the 12 system control instructions.

Table 3-16 System Control Instructions

Instruction		Size	Function
System control	TRAPA	—	Generates a trap exception with a specified vector number.
	TRAP/VS	—	Generates a trap exception if the V bit is set to 1 when the instruction is executed.
	RTE	—	Returns from an exception-handling routine.
	LINK	—	FP → @-SP; SP → FP; SP + #IMM → SP Creates a stack frame.
	UNLK	—	FP → SP; @SP+ → FP Deallocates a stack frame created by the LINK instruction.
	SLEEP	—	Causes a transition to the power-down state.
	LDC	B/W*	(EAs) → CR Moves immediate data or general register or memory contents to a specified control register.
	STC	B/W*	CR → (EAd) Moves control register data to a specified general register or memory location.
	ANDC	B/W*	CR ∧ #IMM → CR Logically ANDs a control register with immediate data.
	ORC	B/W*	CR ∨ #IMM → CR Logically ORs a control register with immediate data.
	XORC	B/W*	CR ⊕ #IMM → CR Logically exclusive-ORs a control register with immediate data.
	NOP	—	PC + 1 → PC No operation. Only increments the program counter.

* The size depends on the control register.

When using the LDC and STC instructions to stack and unstack the BR, CCR, TP, DP, and EP control registers in the H8/500 family, note the following point.

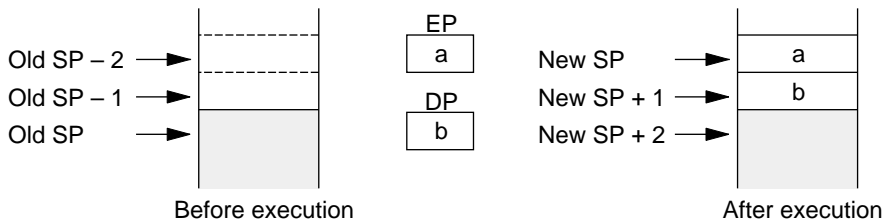
H8/500 hardware does not permit byte access to the stack. If the LDC.B or STC.B assembler mnemonic is coded with the @R7 + (@SP+) or @-R7 (@-SP) addressing mode, the stack-pointer addressing mode takes precedence and hardware automatically performs word access.

Specifically, the LDC.B and STC.B instructions are executed as follows.

The following applies only to the stack-pointer addressing modes. In addressing modes that do not use the stack pointer, byte data access is performed as specified by the assembler mnemonic.

1 STC.B EP, @-SP

When word data access is applied to EP, both EP and DP are accessed. This instruction stores EP at address SP (old) -2, and DP at address SP (old) -1.



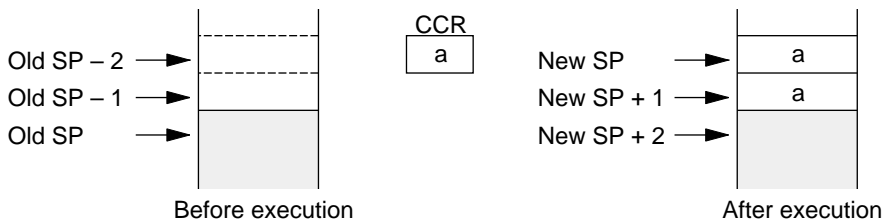
2 LDC.B @SP+, EP

When word data access is applied to EP, both EP and DP are accessed. This instruction loads EP from address SP (old), and DP from address SP (old) +1, updating the DP value as well as the EP value.



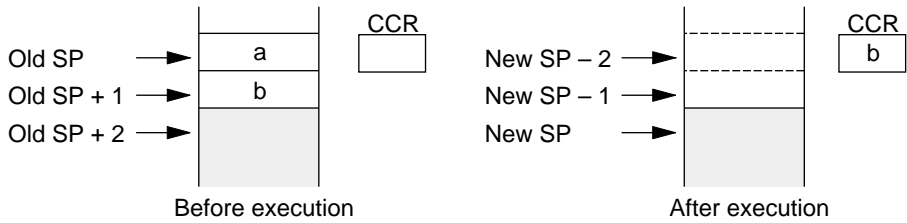
3 STC.B CCR, @-SP

When word data access is applied to CCR, only CCR is accessed. This instruction stores identical CCR contents at both address SP (old) -2 and address SP (old) -1.



4 LDC.B @SP+, CCR

When word data access is applied to CCR, only CCR is accessed. This instruction loads CCR from address SP (old) +1. Note that the value in address SP (old) is not loaded.



BR, DP, and TP are accessed in the same way as CCR. When DP is specified, both EP and DP are accessed, but when CCR, BR, DP, or TP is specified, only the specified register is accessed.

3.5.9 Short-Format Instructions

The ADD, CMP, and MOV instructions have special short formats. Table 3-17 lists these short formats together with the equivalent general formats.

The short formats are a byte shorter than the corresponding general formats, and most of them execute one state faster.

Table 3-17 Short-Format Instructions and Equivalent General Formats

Short-Format Instruction	Length	Execution States *2	Equivalent General-Format Instruction	Length	Execution States *2
ADD:Q #xx,Rd *1	2	2	ADD:G #xx:8,Rd	3	3
CMP:E #xx:8,Rd	2	2	CMP:G.B #xx:8,Rd	3	3
CMP:I #xx:16,Rd	3	3	CMP:G.W #xx:16,Rd	4	4
MOV:E #xx:8,Rd	2	2	MOV:G.B #xx:8,Rd	3	3
MOV:I #xx:16,Rd	3	3	MOV:G.W #xx:16,Rd	4	4
MOV:L @aa:8,Rd	2	5	MOV:G @aa:8,Rd	3	5
MOV:S Rs,@aa:8	2	5	MOV:G Rs,@aa:8	3	5
MOV:F @(d:8,R6),Rd	2	5	MOV:G @(d:8,R6),Rd	3	5
MOV:F Rs,@(d:8,R6)	2	5	MOV:G Rs,@(d:8,R6)	3	5

Notes: * 1 The ADD:Q instruction accepts other destination operands in addition to a general register, but the immediate data value (#xx) is limited to ± 1 or ± 2 .

* 2 Number of execution states for access to a general register.

3.6 Operating Modes

The CPU operates in one of two modes: minimum mode or maximum mode. These modes are selected by the mode pins MD2 to MD0.

3.6.1 Minimum Mode

The minimum mode supports a maximum address space of 64-kbytes. The page registers are ignored. Instructions that branch across page boundaries (PJMP, PJSR, PRTS, PRTD) are invalid.

3.6.2 Maximum Mode

In maximum mode the page registers are valid, expanding the maximum address space to 16 Mbytes.

The address space is divided into 64-kbyte pages. The pages are separate; it is not possible to move continuously across a page boundary.

It is possible to move from one page to another with branching instructions (PJMP, PJSR, PRTS, PRD, TRAPA), or by branching to an interrupt-handling routine. It is not necessary for a program to be contained in a single 64-kbyte page.

When data access crosses a page boundary, the program must rewrite the page register before it can access the data in the next page.

For further information on the operating modes, see section 2, “MCU Operating Modes and Address Space.”

3.7 Basic Operational Timing

3.7.1 Overview

The CPU operates on a system clock (ϕ) which is created by dividing an oscillator frequency (f_{osc}) by two. One system clock cycle is referred to as a “state.” The CPU accesses memory in a bus cycle consisting of two or three states.

1. **Two-State Access:** Is provided for high-speed processing. No wait states (T_w) can be inserted.

Figure 3-7 shows the two-state access cycle.

2. **Three-State Access:** Is provided for interfacing low-speed devices. Figure 3-8 shows the three-state access cycle. Wait states (T_w) can be inserted by the wait-state controller (WSC).
3. **Access to On-Chip Register Field:** The access cycle consists of three states. The data bus is 8 bits wide.

Figure 3-9 shows the on-chip register field access cycle. Figure 3-10 includes the pin states.

3.7.2 Two-State Access Cycle

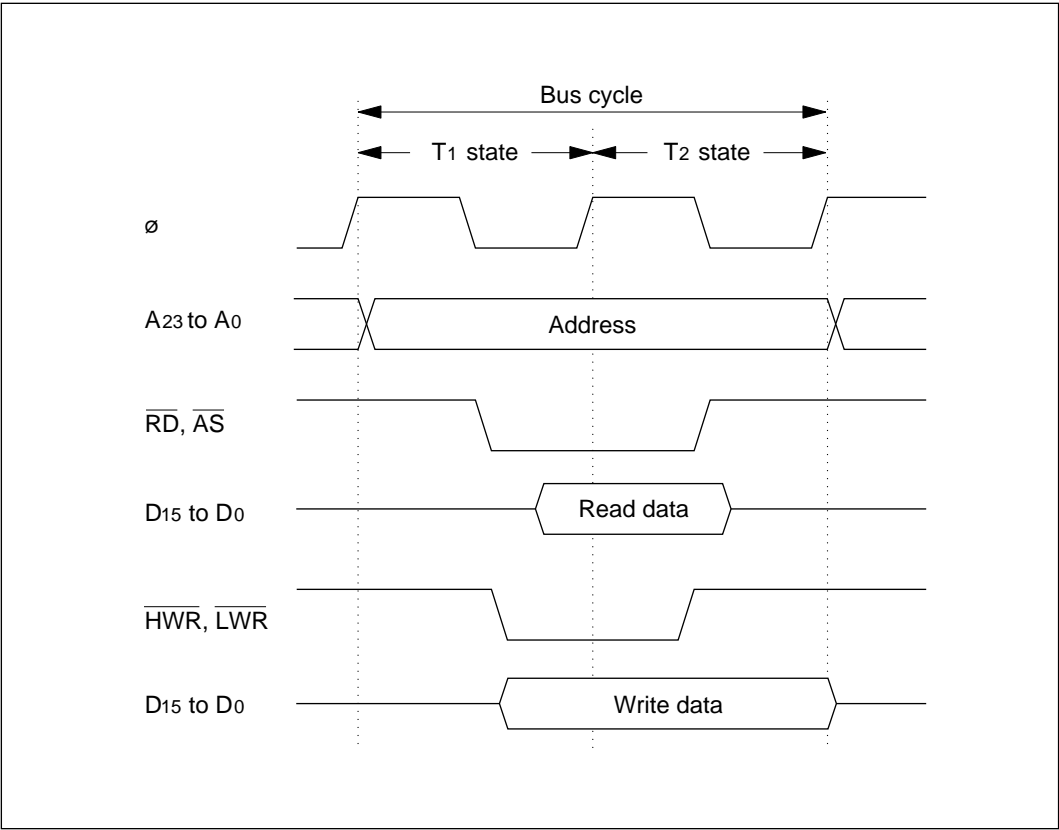


Figure 3-7 Two-State Access Cycle

3.7.3 Three-State Access Cycle

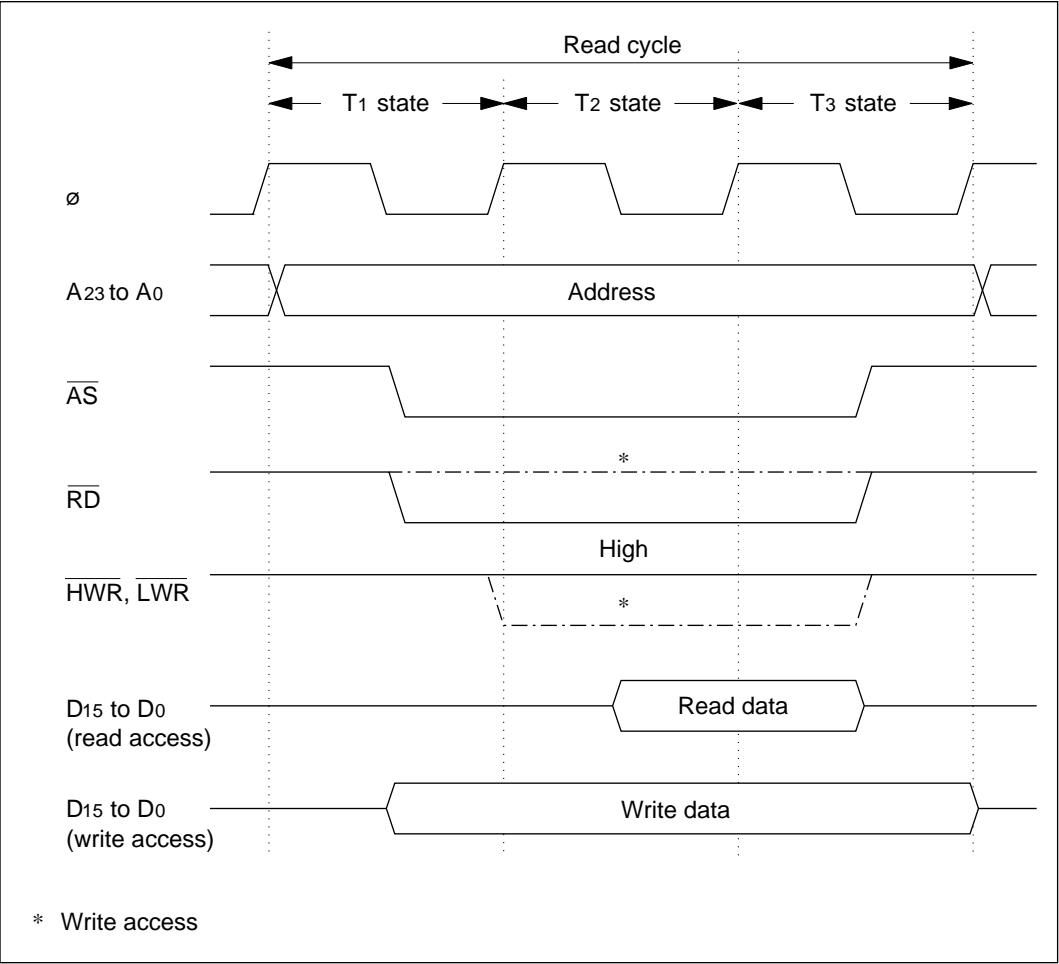


Figure 3-8 Three-State Access Cycle

3.7.4 On-Chip Register Field Access Cycle

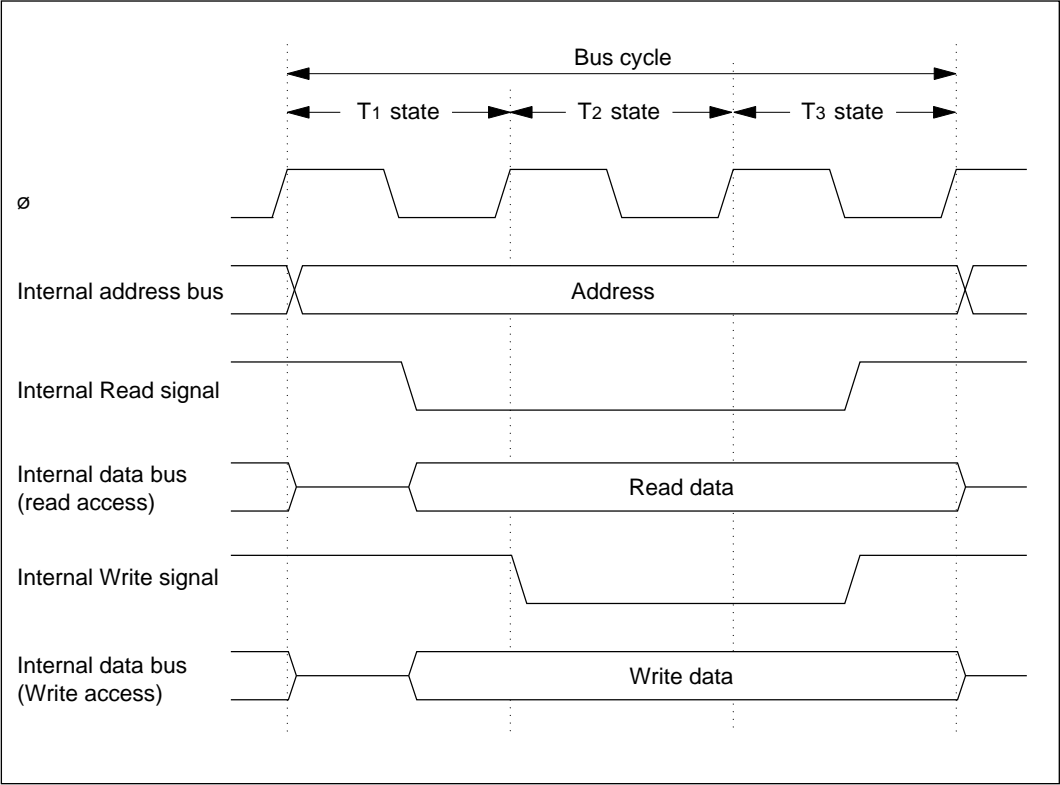


Figure 3-9 Register Field Access Cycle

3.7.5 Pin States during Register Field Access

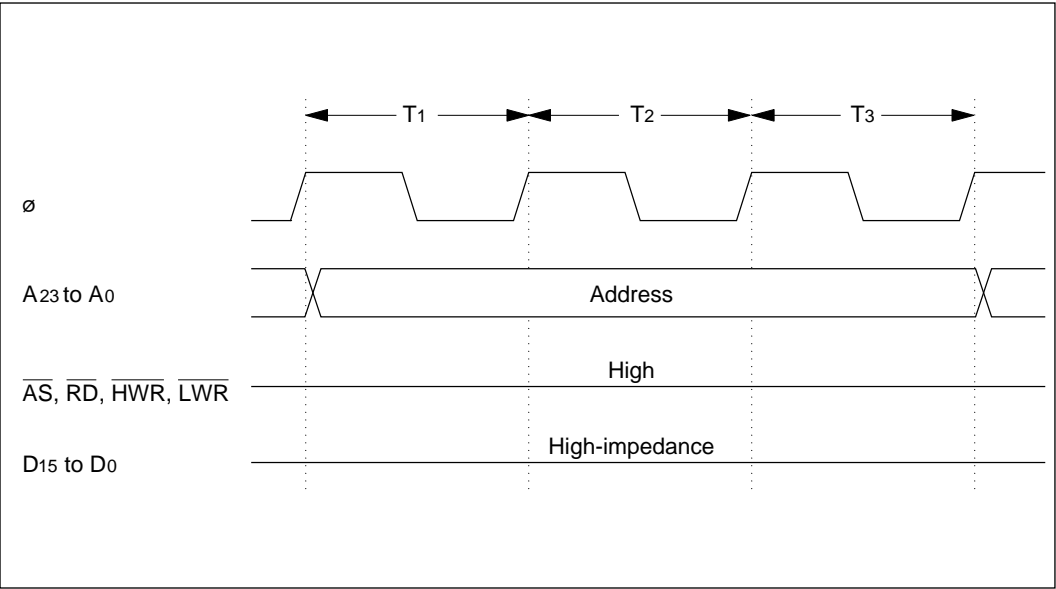


Figure 3-10 Pin States during Register Field Access

3.8 CPU States

3.8.1 Overview

The CPU has five states: the program execution state, exception-handling state, bus-released state, reset state, and power-down state. The power-down state is further divided into the sleep mode, software standby mode, and hardware standby mode. Figure 3-11 summarizes these states, and figure 3-12 shows a map of the state transitions.

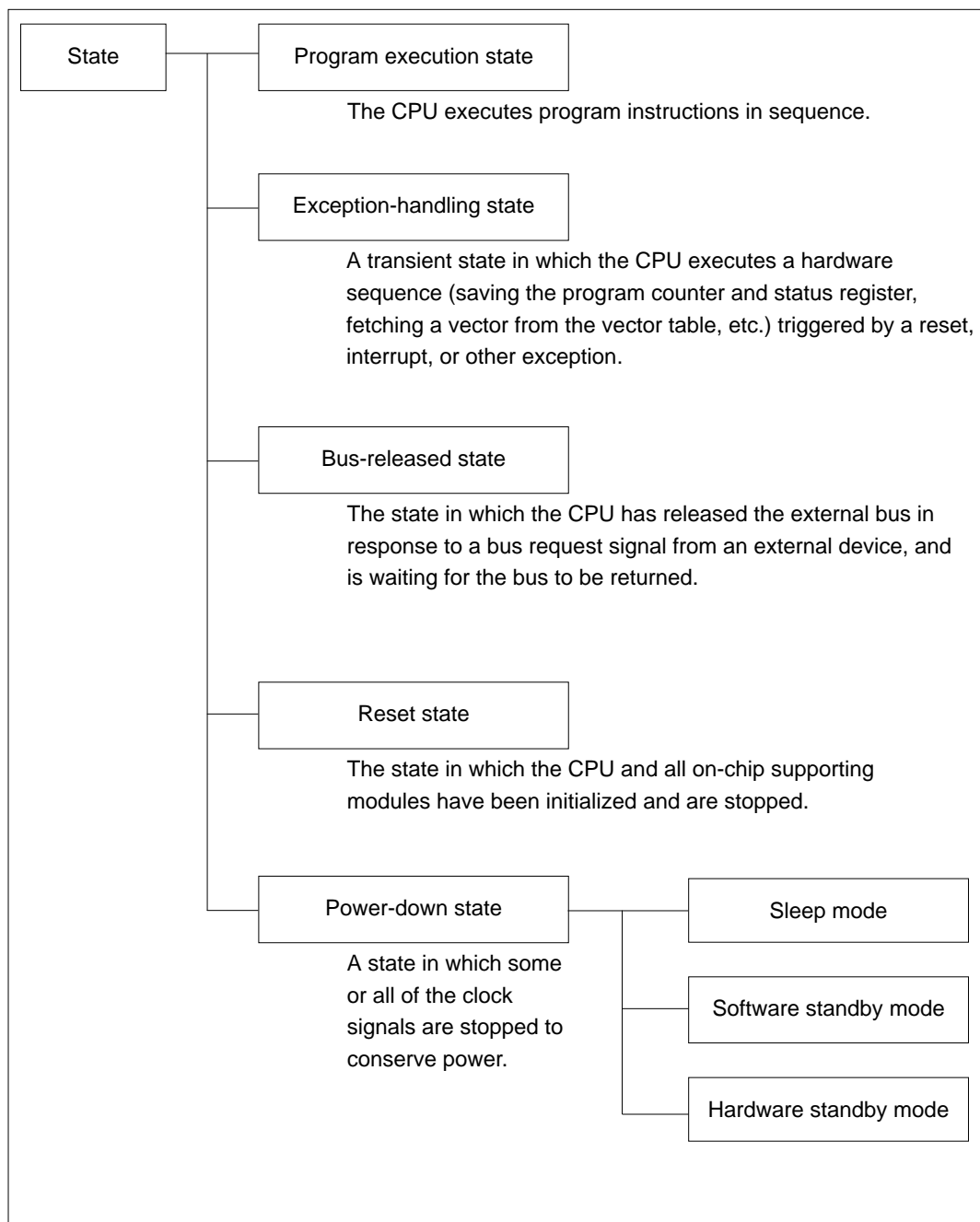


Figure 3-11 Operating States

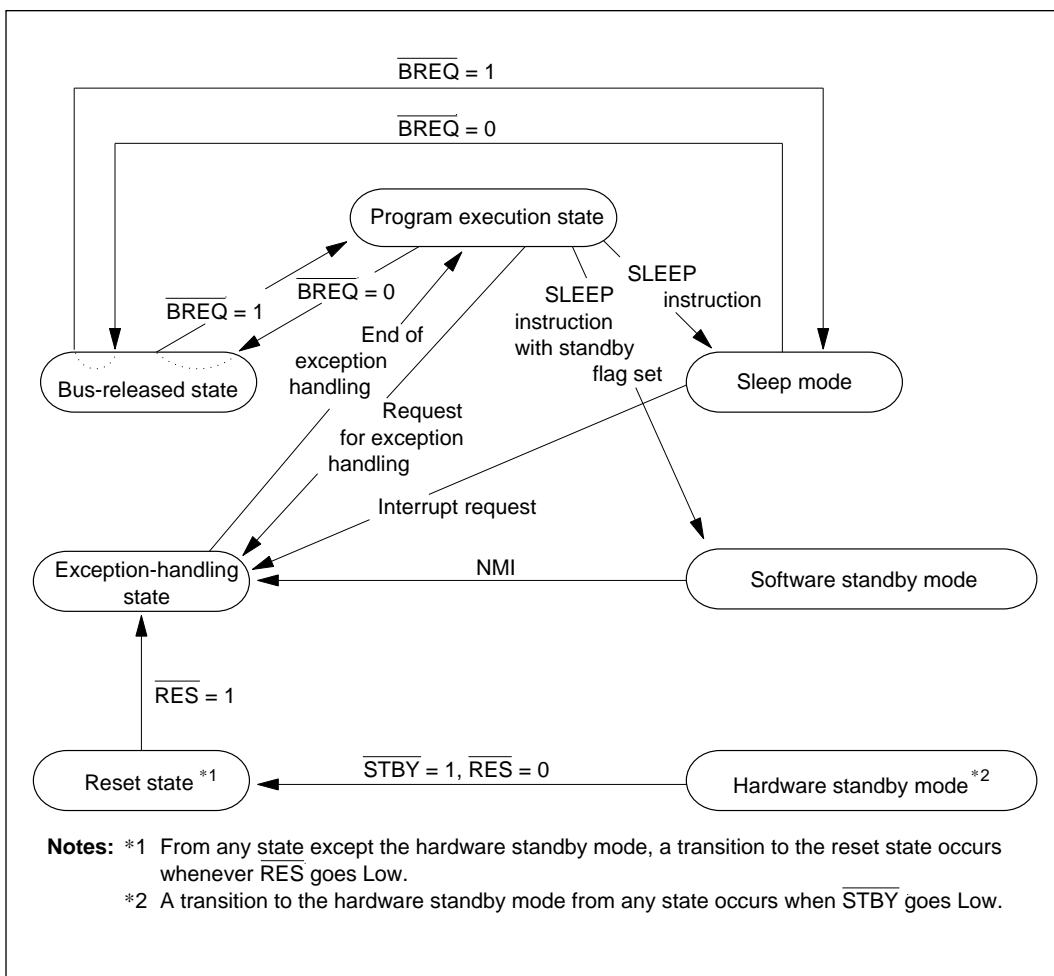


Figure 3-12 State Transitions

3.8.2 Program Execution State

In this state the CPU executes program instructions in normal sequence.

3.8.3 Exception-Handling State

The exception-handling state is a transient state that occurs when the CPU alters the normal program flow due to an interrupt, trap instruction, address error, or other exception. In this state the CPU carries out a hardware-controlled sequence that prepares it to execute a user-coded exception-handling routine.

In the hardware exception-handling sequence the CPU does the following:

1. Saves the program counter and status register (in minimum mode) or program counter, code page register, and status register (in maximum mode) to the stack.
2. Clears the T bit in the status register to 0.
3. Fetches the start address of the exception-handling routine from the exception vector table.
4. Branches to that address, returning to the program execution state.

See section 4, “Exception Handling,” for further information on the exception-handling state.

3.8.4 Bus-Released State

When so requested, the CPU can grant control of the external bus to an external device. While an external device has the bus right, the CPU is said to be in the bus-released state. The bus right is controlled by two pins:

- $\overline{\text{BREQ}}$: Input pin for the Bus Request signal from an external device
- $\overline{\text{BACK}}$: Output pin for the Bus Request Acknowledge signal from the CPU, indicating that the CPU has released the bus

The procedure by which the CPU enters and leaves the bus-released state is:

1. The CPU receives a Low $\overline{\text{BREQ}}$ signal from an external device.
2. The CPU places the address bus pins (A23 – A0), data bus pins (D15 – D0) and bus control pins ($\overline{\text{RD}}$, $\overline{\text{LWR}}$, $\overline{\text{HWR}}$, and $\overline{\text{AS}}$) in the high-impedance state, sets the $\overline{\text{BACK}}$ pin to the Low level to indicate that it has released the bus, then halts.
3. The external device that requested the bus (with the $\overline{\text{BREQ}}$ signal) becomes the bus master. It can use the data bus and address bus. The external device is responsible for manipulating the bus control signals ($\overline{\text{RD}}$, $\overline{\text{LWR}}$, $\overline{\text{HWR}}$, and $\overline{\text{AS}}$).
4. When the external device finishes using the bus, it clears the $\overline{\text{BREQ}}$ signal to the High level. The CPU then reassumes control of the bus and returns to the program execution state.

Bus Release Timing: The CPU can release the bus at the following times:

1. The $\overline{\text{BREQ}}$ signal is sampled during every memory access cycle (instruction prefetch or data read/write). If $\overline{\text{BREQ}}$ is Low, the CPU releases the bus right at the end of the cycle. (In word data access to the on-chip register field, or to external memory via an 8-bit data bus, the CPU does not release the bus until it has accessed both the upper and lower data bytes.)
2. During execution of the MULXU and DIVXU instructions, since considerable time may pass without an instruction prefetch or data read/write, $\overline{\text{BREQ}}$ is also sampled at internal machine cycles, and the bus is released if $\overline{\text{BREQ}}$ is Low.
3. The bus can also be released in the sleep mode.

The CPU does not recognize interrupts while the bus is released.

Timing Charts: Timing charts of the operation by which the bus is released are shown in figure 3-13 for the case of bus release during a two-state read cycle, in figure 3-14 for bus release during a three-state read cycle, and in figure 3-15 for bus release while the CPU is performing an internal operation.

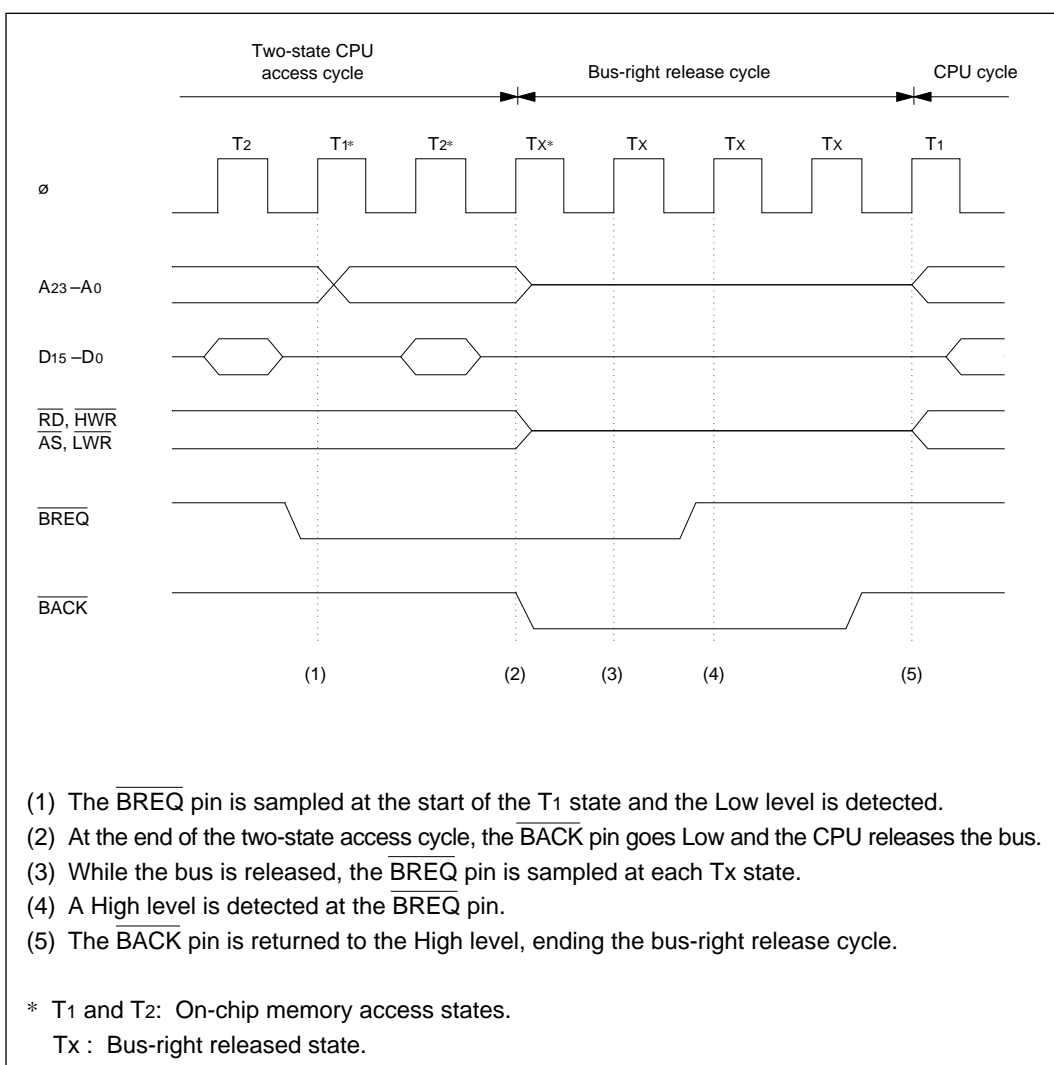
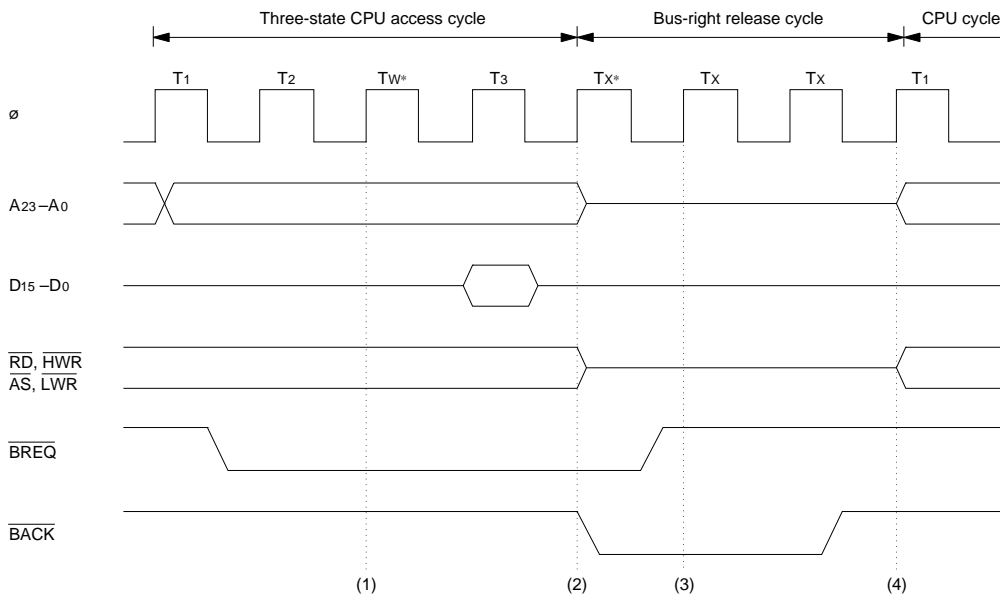


Figure 3-13 Bus-Right Release Cycle (During Two-State Access Cycle)



- (1) The \overline{BREQ} pin is sampled at the start of the Tw state and the Low level is detected.
- (2) At the end of the three-state access cycle, the \overline{BACK} pin goes Low and the CPU releases the bus.
- (3) The \overline{BREQ} pin is sampled at the Tx state and a High level is detected.
- (4) The \overline{BACK} pin is returned to the High level, ending the bus-right release cycle.

* Tw : Wait state.

Tx : Bus-right released state.

Figure 3-14 Bus-Right Release Cycle (During Three-State Access Cycle)

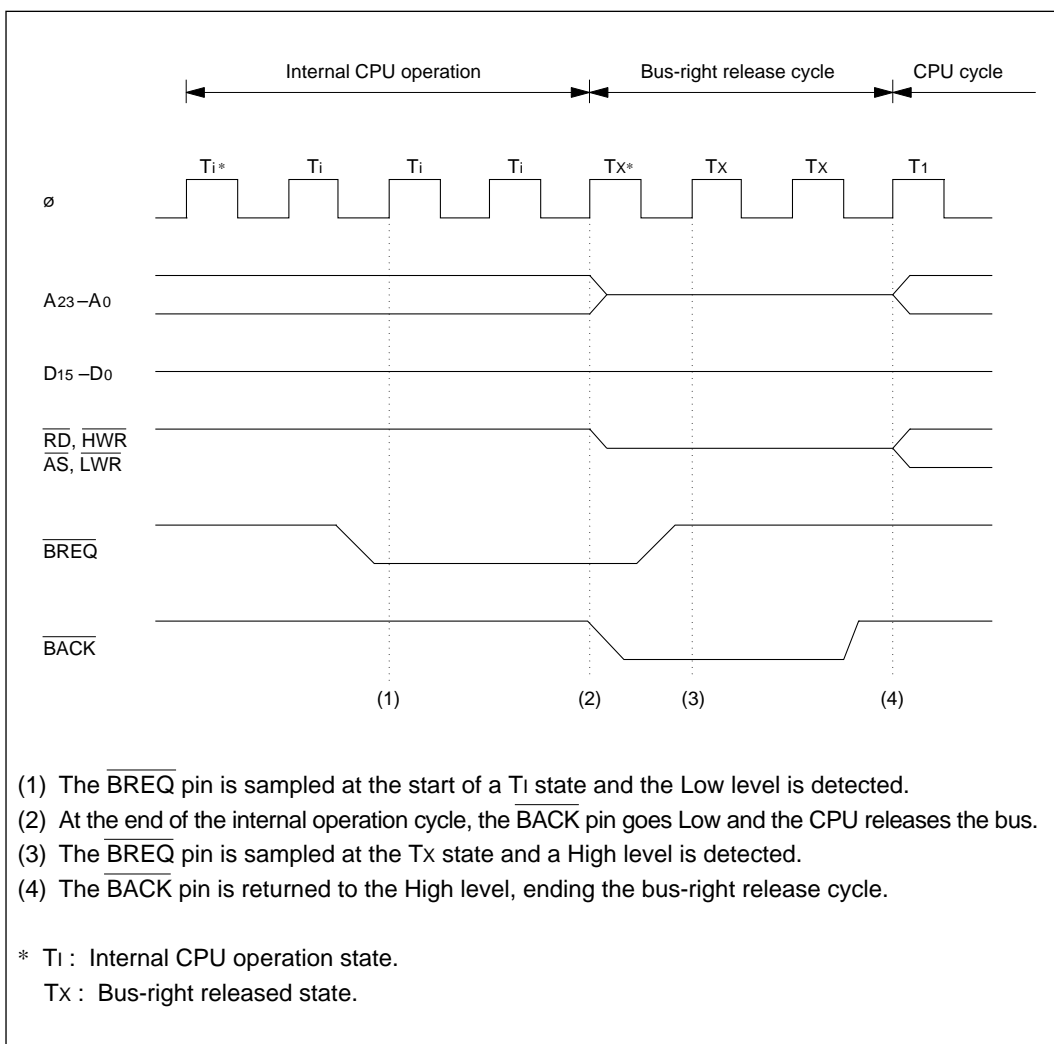


Figure 3-15 Bus-Right Release Cycle (During Internal CPU Operation)

Notes: The \overline{BREQ} signal must be held Low until \overline{BACK} goes Low. If \overline{BREQ} returns to the High level before \overline{BACK} goes Low, the bus release operation may be executed incorrectly.

To leave the bus-released state, the High level at the \overline{BREQ} pin must be sampled two times. If the \overline{BREQ} returns to Low before it is sampled two times, the bus released cycle will not end.

The bus release operation is enabled only when the BRLE bit in the Bus Release Control (BRCR) is set to 1. When this bit is cleared to 0 (its initial value), the $\overline{\text{BREQ}}$ and $\overline{\text{BACK}}$ pins are used for general-purpose input and output, as P32 and P31.

3.8.5 Reset State

In the reset state, the CPU and all on-chip supporting modules are initialized and placed in the stopped state. The CPU enters the reset state whenever the $\overline{\text{RES}}$ pin goes Low, unless the CPU is currently in the hardware standby mode. It remains in the reset state until the $\overline{\text{RES}}$ pin goes High.

See section 4.2, “Reset,” for further information on the reset state.

3.8.6 Power-Down State

The power-down state comprises three modes: the sleep mode, the software standby mode, and the hardware standby mode.

See section 17, “Power-Down State,” for further information.

Section 4 Exception Handling

4.1 Overview

4.1.1 Types of Exception Handling and Their Priority

As indicated in table 4-1 (a) and (b), 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, as shown in table 4-1 (a). 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.

Table 4-1 (a) Exceptions and Their Priority


Priority	Exception Type	Source	Detection Timing	Start of Exception-Handling Sequence
High	Reset	External	$\overline{\text{RES}}$ 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
Low				

Table 4-1 (b) 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

4.1.2 Hardware Exception-Handling Sequence

The hardware exception-handling sequence varies depending on the type of exception. When exception handling is initiated by a factor other than a reset, the CPU:

1. Saves the program counter and status register (in minimum mode) or program counter, code page register, and status register (in maximum mode) to the stack.
2. Clears the T bit in the status register to 0.
3. Fetches the start address of the exception-handling routine from the exception vector table.
4. Branches to that address.

For an interrupt, the CPU also alters the interrupt mask level in bits I2 to I0 of the status register.

For a reset, step 1 is omitted. See section 4.2, “Reset,” for the full reset sequence.

4.1.3 Exception Factors and Vector Table

The factors that initiate exception handling can be classified as shown in figure 4-1.

The starting addresses of the exception-handling routines for each factor are contained in an exception vector table located in the low addresses of page 0. The vector addresses are listed in table 4-2. Note that there are different addresses for the minimum and maximum modes.

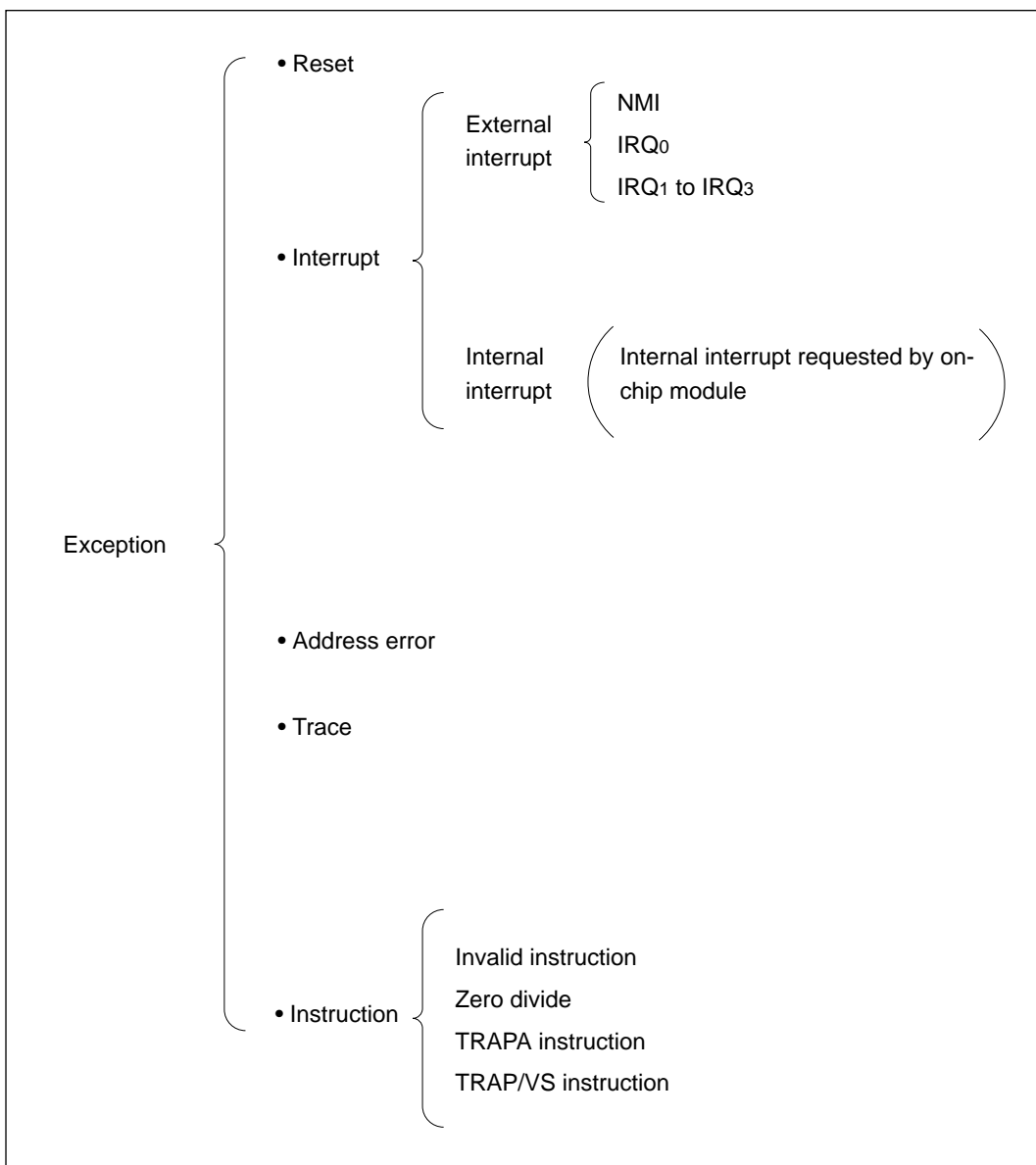


Figure 4-1 Types of Factors Causing Exception Handling

Table 4-2 Exception Vector Table

Type of Exception		Vector Address	
		Minimum Mode	Maximum Mode *1
Reset (initialize PC)		H'0000 to H'0001	H'0000 to H'0003
— (Reserved for system)		H'0002 to H'0003	H'0004 to H'0007
Invalid instruction		H'0004 to H'0005	H'0008 to H'000B
DIVXU instruction (zero divide)		H'0006 to H'0007	H'000C to H'000F
TRAP/VS instruction		H'0008 to H'0009	H'0010 to H'0013
— (Reserved for system)		H'000A to H'000B	H'0014 to H'0017
		to	to
		H'000E to H'000F	H'001C to H'001F
Address error		H'0010 to H'0011	H'0020 to H'0023
Trace		H'0012 to H'0013	H'0024 to H'0027
— (Reserved for system)		H'0014 to H'0015	H'0028 to H'002B
Nonmaskable external interrupt (NMI)		H'0016 to H'0017	H'002C to H'002F
— (Reserved for system)		H'0018 to H'0019	H'0030 to H'0033
		to	to
		H'001E to H'001F	H'003C to H'003F
TRAPA instruction (16 vectors)		H'0020 to H'0021	H'0040 to H'0043
		to	to
		H'003E to H'003F	H'007C to H'007F
External interrupt	IRQ0	H'0040 to H'0041	H'0080 to H'0083
Watchdog timer interval interrupt		H'0042 to H'0043	H'0084 to H'0087
External interrupts	IRQ1	H'0048 to H'0049	H'0090 to H'0093
	IRQ2	H'004A to H'004B	H'0094 to H'0097
	IRQ3	H'004C to H'004D	H'0098 to H'009B
Internal interrupts *2		H'0050 to H'0051	H'00A0 to H'00A3
		to	to
		H'0078 to H'0079	H'00F0 to H'00F3

Notes: * 1 The exception vector table is located at the beginning of page 0 in maximum mode.

* 2 For details of the internal interrupt vectors, see table 5-2.

4.2 Reset

4.2.1 Overview

A reset has the highest exception-handling priority.

When the $\overline{\text{RES}}$ pin goes Low, all current processing is halted and the H8/510 chip enters the reset state.

A reset initializes the internal status of the CPU and the registers of the on-chip supporting modules and I/O ports.

When the $\overline{\text{RES}}$ pin returns from Low to High, the H8/510 chip comes out of the reset state and begins executing the hardware reset sequence.

4.2.2 Reset Sequence

The Reset signal is detected when the $\overline{\text{RES}}$ pin goes Low.

To ensure that the H8/510 is reset, the $\overline{\text{RES}}$ pin should be held Low for at least 20 ms at power-up. To reset the H8/510 during operation, the $\overline{\text{RES}}$ pin should be held Low for at least 6 system clock cycles. See table E.1, “Status of I/O Ports” in appendix E for the status of other pins in the reset state.

When the $\overline{\text{RES}}$ pin returns to the High state after being held Low for the necessary time, the hardware reset exception-handling sequence begins, during which:

1. In the status register (SR), the T bit is cleared to disable the trace mode, and the interrupt mask level (bits I2 to I0) is set to 7. A reset disables all interrupts, including NMI.
2. The CPU loads the reset start address from the vector table into the program counter and begins executing the program at that address.

The contents of the vector table differs between minimum mode and maximum mode as indicated in figure 4-2. This affects step 3 as follows:

Minimum Mode: One word is copied from addresses H'0000 and H'0001 in the vector table to the program counter. Program execution then begins from the address in the program counter (PC).

Maximum Mode: Two words are read from addresses H'0000 to H'0003 in the vector table. The byte in address H'0000 is ignored. The byte in address H'0001 is copied to the code page register (CP). The contents of addresses H'0002 and H'0003 are copied to the program counter. Program execution starts from the address indicated by the code page register and program counter.

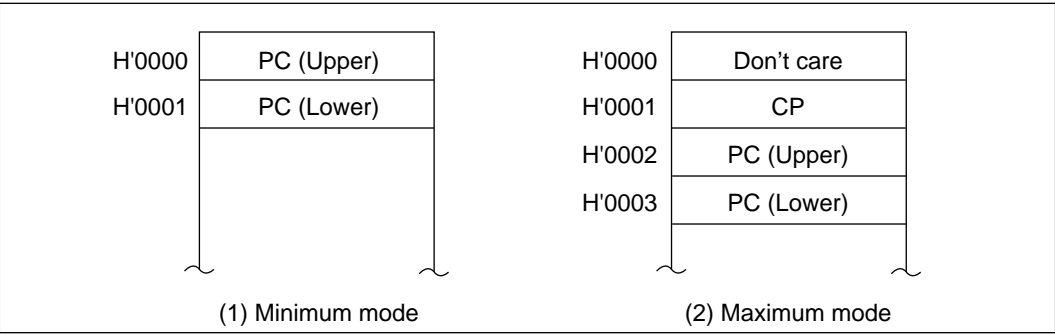


Figure 4-2 Reset Vector

Figure 4-3 shows the timing of the reset sequence in minimum mode. Figure 4-4 shows the timing of the reset sequence in maximum mode.

4.2.3 Stack Pointer Initialization

The hardware reset sequence does not initialize the stack pointer, so this must be done by software. If an interrupt were to be accepted after a reset and before the stack pointer (SP) is initialized, the program counter and status register would not be saved correctly, causing a program crash. This danger can be avoided by coding the reset routine as explained next.

When the chip comes out of the reset state all interrupts, including NMI, are disabled, so the instruction at the reset start address is always executed. In the minimum mode, this instruction should initialize the stack pointer (SP). In the maximum mode, this instruction should be an LDC instruction initializing the stack page register (TP), and the next instruction should initialize the stack pointer. Execution of the LDC instruction disables interrupts again, ensuring that the stack pointer initializing instruction is executed.

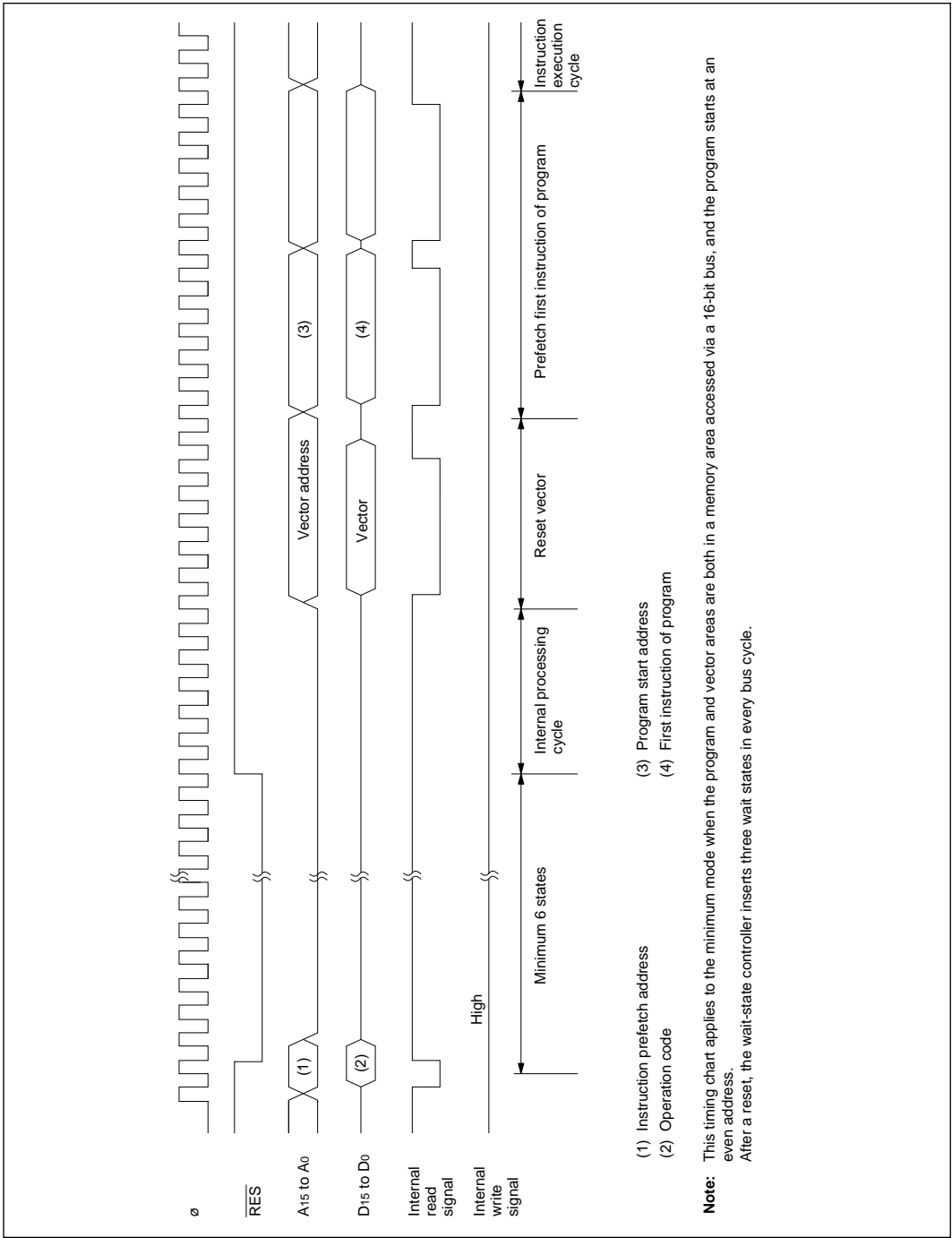
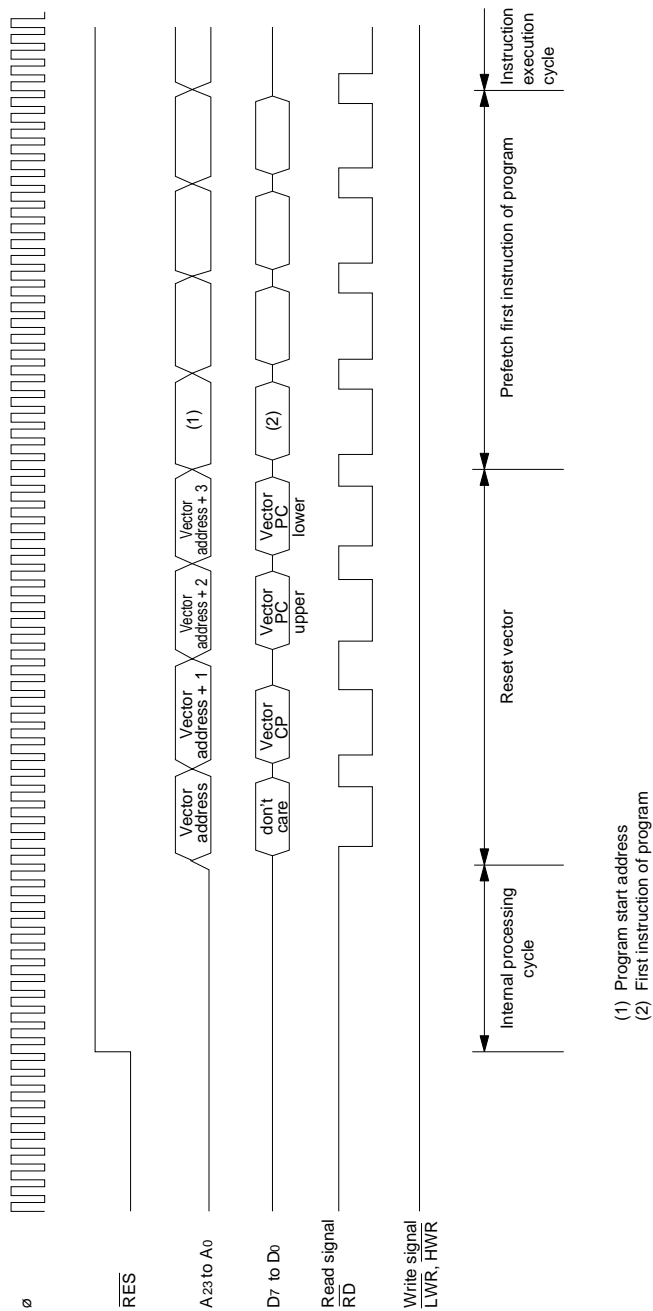


Figure 4-3 Reset Sequence (Minimum Mode)



Note: This timing chart applies to maximum mode when the program and vector areas are both accessed via an 8-bit bus. After a reset, the wait-state controller inserts three wait states in each bus cycle.

Figure 4-4 Reset Sequence (Maximum Mode)

4.3 Address Error

There are two causes of address errors:

- Illegal instruction prefetch
- Word data access at odd address

An address error initiates the address error exception-handling sequence. This sequence clears the T bit of the status register to 0 to disable the trace mode, but does not affect the interrupt mask level in bits I2 to I0.

4.3.1 Illegal Instruction Prefetch

An attempt to prefetch an instruction from the register field and external I/O area in memory addresses H'FE80 to H'FFFF causes an address error regardless of the MCU operating mode.

Handling of this address error begins when the prefetch cycle that caused the error has been completed and execution of the current instruction has also been completed. The program counter value pushed on the stack is the address of the instruction immediately following the last instruction executed.

Program code should not be located in addresses H'FE7D to H'FE7F. If the CPU executes an instruction in these addresses, it will attempt to prefetch the next instruction from the register field, causing an address error.

4.3.2 Word Data Access at Odd Address

If an attempt is made to access word data starting at an odd address, an address error occurs regardless of the MCU operating mode. The program counter value pushed on the stack in the handling of this error is the address of the next instruction (or next but one) after the instruction that attempted the illegal word access.

4.4 Trace

When the T bit of the status register is set to 1, the CPU operates in trace mode. A trace exception occurs at the completion of each instruction. The trace mode can be used to execute a program for debugging by a debugger.

In the trace exception sequence the T bit of the status register is cleared to 0 to disable the trace mode while the trace routine is executing. The interrupt mask level in bits I2 to I0 is not changed. Interrupts are accepted as usual during the trace routine.

In the status-register data saved on the stack, the T bit is set to 1. When the trace routine returns with the RTE instruction, the status register is popped from the stack and the trace mode resumes.

If an address error occurs during execution of the first instruction after the return from the trace routine, since the address error has higher priority, the address error exception-handling sequence is initiated, clearing the T bit in the status register to 0 and making it impossible to trace this instruction.

4.5 Interrupts

Interrupts can be requested from five external sources (NMI, IRQ0, IRQ1, IRQ2, IRQ3) and seven on-chip supporting modules: the 16-bit free-running timers (FRT1 and FRT2), the 8-bit timer, the serial communication interfaces (SCI1 and SCI2), the A/D converter, and the watchdog timer (WDT). The on-chip interrupt sources can request a total of eighteen different types of interrupts, each having its own interrupt vector. Figure 4-5 lists the interrupt sources and the number of different interrupts from each source.

Each interrupt source has a priority. NMI interrupts have the highest priority, and are normally accepted unconditionally. The priorities of the other interrupt sources are set in control registers (IPRA to IPRD) in the register field at the high end of page 0 and can be changed by software. Priority levels range from 7 (high) to 0 (low), with NMI considered to be on level 8. Priorities can be assigned to IRQ0 individually and to IRQ1 to IRQ3 as a group. For the other interrupt sources, priorities are assigned to the on-chip supporting module in which the interrupt originates.

The on-chip interrupt controller decides whether an interrupt can be accepted by comparing its priority with the interrupt mask level, and determines the order in which to accept competing interrupt requests. Interrupts that are not accepted immediately remain pending until they can be accepted later.

When it accepts an interrupt, the interrupt controller also decides whether to interrupt the CPU or start the on-chip data transfer controller (DTC). This decision is controlled by bits set in four data transfer enable registers (DTEA to DTED) in the register field. The DTC is started if the corresponding DTE bit is set to 1; otherwise a CPU interrupt is generated. DTC interrupts provide

an efficient way to send and receive blocks of data via the serial communication interface, or to transfer data between memory and I/O without detailed CPU programming. The CPU halts during DTC operation. DTC interrupts are described in section 6, “Data Transfer Controller.”

The hardware exception-handling sequence for a CPU interrupt clears the T bit in the status register to 0 and sets the interrupt mask level in bits I2 to I0 to the level of the interrupt it has accepted. This prevents the interrupt-handling routine from being interrupted except by a higher-level interrupt. The previous interrupt mask level is restored on the return from the interrupt-handling routine.

For further information on interrupts, see section 5, “Interrupt Controller.”

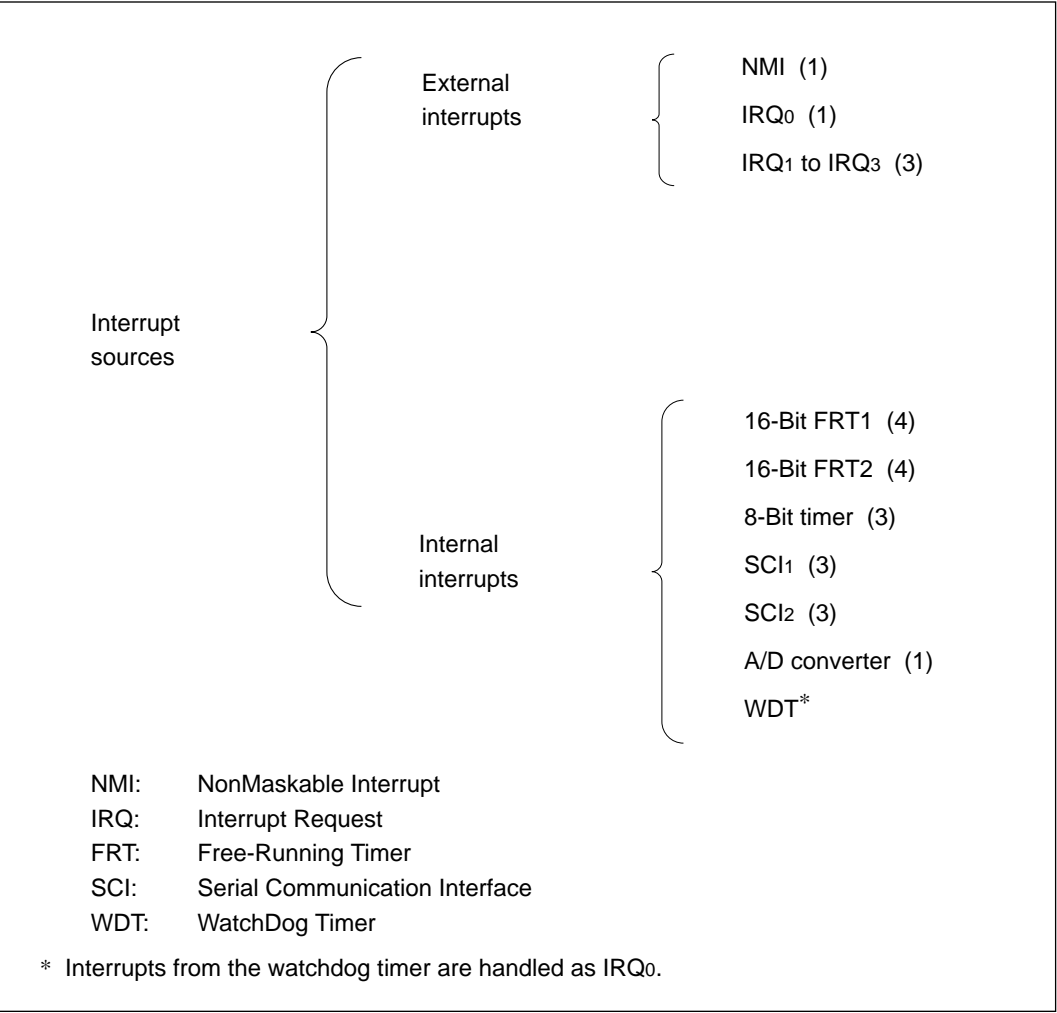


Figure 4-5 Interrupt Sources (and Number of Interrupt Types)

4.6 Invalid Instruction

An invalid instruction exception occurs if an attempt is made to execute an instruction with an undefined operation code or illegal addressing mode specification. The program counter value pushed on the stack is the value of the program counter when the invalid instruction code was detected.

In the invalid instruction exception-handling sequence the T bit of the status register is cleared to 0, but the interrupt mask level (I2 to I0) is not affected.

4.7 Trap Instructions and Zero Divide

A trap exception occurs when the TRAPA or TRAP/VS instruction is executed. A zero divide exception occurs if an attempt is made to execute a DIVXU instruction with a zero divisor.

In the exception-handling sequences for these exceptions the T bit of the status register is cleared to 0, but the interrupt mask level (I2 to I0) is not affected. If a normal interrupt is requested while a trap or zero-divide instruction is being executed, after the trap or zero-divide exception-handling sequence, the normal interrupt exception-handling sequence is carried out.

TRAPA Instruction: The TRAPA instruction always causes a trap exception. The TRAPA instruction includes a vector number from 0 to 15, allowing the user to provide up to sixteen different trap-handling routines.

TRAP/VS Instruction: When the TRAP/VS instruction is executed, a trap exception occurs if the overflow (V) bit in the condition code register is set to 1. If the V bit is cleared to 0, no exception occurs and the next instruction is executed.

DIVXU Instruction with Zero Divisor: An exception occurs if an attempt is made to divide by zero in a DIVXU instruction.

4.8 Cases in Which Exception Handling is Deferred

In the cases described next, the address error exception, trace exception, external interrupt (NMI, IRQ0 to IRQ3) requests, and internal interrupt requests (18 types) are not accepted immediately but are deferred until after the next instruction has been executed.

4.8.1 Instructions that Disable Interrupts

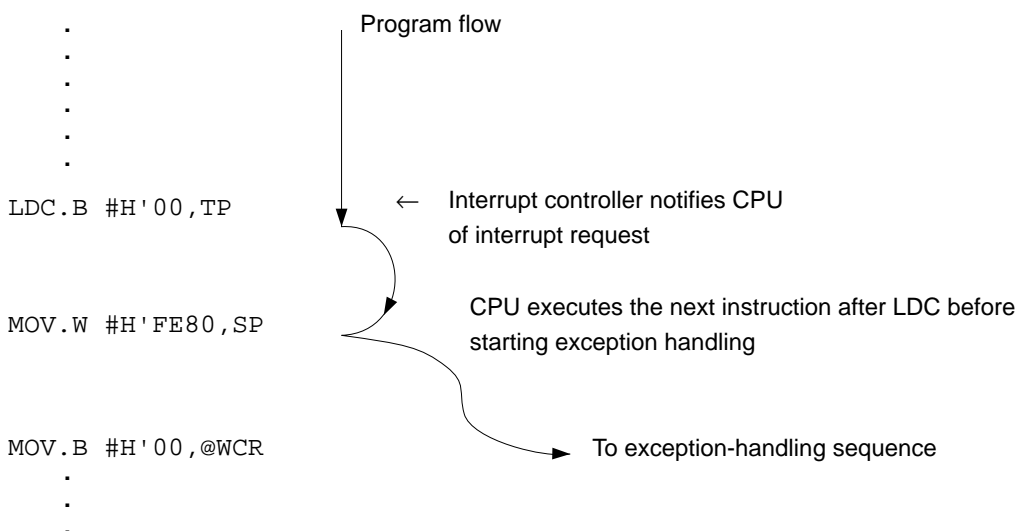
Interrupts are disabled immediately after the execution of five instructions: XORC, ORC, ANDC, LDC, and RTE.

Suppose that an internal interrupt is requested and the interrupt controller, after checking the interrupt priority and interrupt mask level, notifies the CPU of the interrupt, but the CPU is

currently executing one of the five instructions listed above. After executing this instruction the CPU always proceeds to the next instruction. (And if the next instruction is one of these five, the CPU also proceeds to the next instruction after that.) The exception-handling sequence starts after the next instruction that is not one of these five has been executed. The following is an example:

Note: If the LDC instruction modifies the interrupt mask bits in the status register, the new interrupt mask level does not take effect until the third state after the LDC instruction has been executed. If an LDC instruction in a program stored in the memory area accessed in two states via a 16-bit bus modifies the interrupt mask level in order to enable an interrupt, but the next instruction is a two-state instruction (such as NOP), the interrupt will not be accepted after this two-state instruction. It will not be accepted until another instruction has been executed. The same applies to ANDC, ORC, and XORC.

(Example)



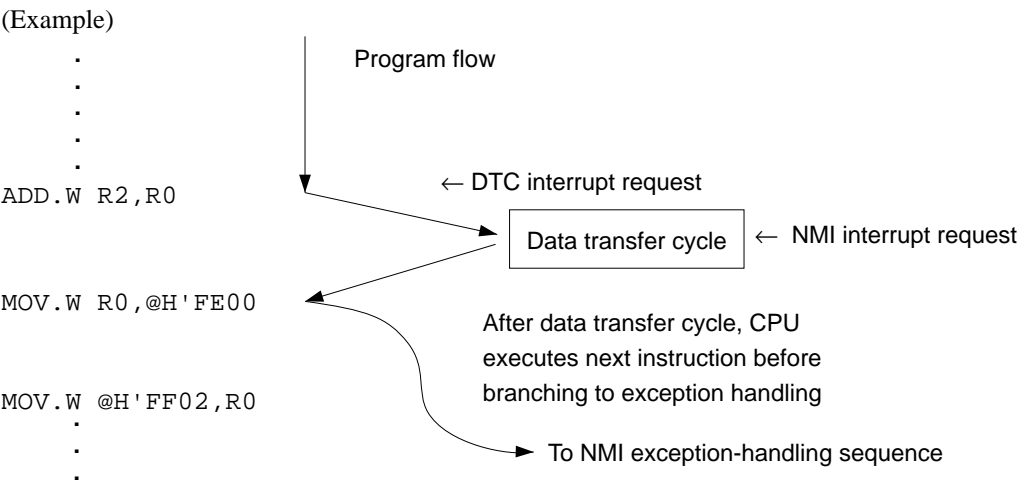
4.8.2 Disabling of Exceptions Immediately after a Reset

If an interrupt is accepted after a reset and before the stack pointer (SP) is initialized, the program counter and status register will not be saved correctly, leading to a program crash. To prevent this, when the chip comes out of the reset state all interrupts, including the NMI, are disabled, so the first instruction of the reset routine is always executed. As noted earlier, in the minimum mode, this instruction should initialize the stack pointer (SP). In the maximum mode, the first instruction should be an LDC instruction that initializes the stack page register (TP); the next instruction should initialize the stack pointer.

4.8.3 Disabling of Interrupts after a Data Transfer Cycle

If an interrupt starts the data transfer controller and another interrupt is requested during the data transfer cycle, when the data transfer cycle ends, the CPU always executes the next instruction before handling the second interrupt.

Even if a nonmaskable interrupt (NMI) occurs during a data transfer cycle, it is not accepted until the next instruction has been executed. An example of this is shown below.



4.9 Stack Status after Completion of Exception Handling

The status of the stack after an exception-handling sequence is described below.

Table 4-3 shows the stack after completion of the exception-handling sequence for various types of exceptions in the minimum and maximum modes.

Table 4-3 Stack after Exception Handling Sequence

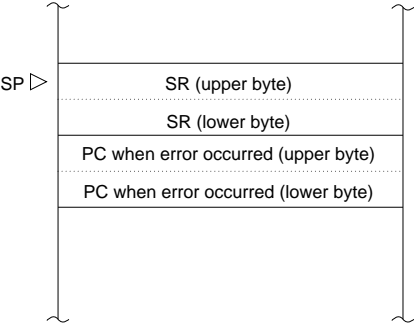
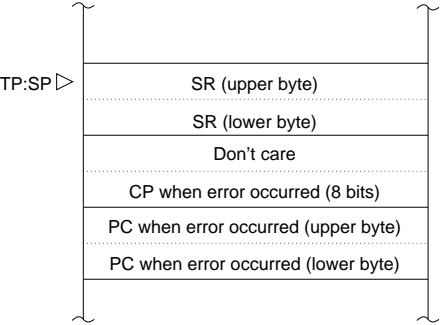
Exception Cause	Minimum Mode	Maximum Mode
Trace	<div>SP ▷</div> <div>SR (upper byte)</div> <div>SR (lower byte)</div> <div>Next instruction address (upper byte)</div> <div>Next instruction address (lower byte)</div>	<div>TP:SP ▷</div> <div>SR (upper byte)</div> <div>SR (lower byte)</div> <div>Don't care</div> <div>Next instruction page (8 bits)</div> <div>Next instruction address (upper byte)</div> <div>Next instruction address (lower byte)</div>
Interrupt		
Trap		
Zero divide (DIVXU)		

Note: The RTE instruction returns to the next instruction after the instruction being executed when the exception occurred.

Exception Cause	Minimum Mode	Maximum Mode
Invalid instruction	SP ▷	TP:SP ▷
	SR (upper byte)	SR (upper byte)
	SR (lower byte)	SR (lower byte)
	PC when error occurred (upper byte)	Don't care
	PC when error occurred (lower byte)	CP when error occurred (8 bits)
		PC when error occurred (upper byte)
		PC when error occurred (lower byte)

Note: The program counter value pushed on the stack is not necessarily the address of the first byte of the invalid instruction.

Table 4-3 Stack after Exception Handling Sequence (cont)

Exception Cause	Minimum Mode	Maximum Mode
Address error		

Note: The program counter value pushed on the stack is the address of the next instruction after the last instruction successfully executed.

4.9.1 PC Value Pushed on Stack for Trace, Interrupts, Trap Instructions, and Zero Divide Exceptions

The program counter value pushed on the stack for a trace, interrupt, trap, or zero divide exception is the address of the next instruction at the time when the interrupt was accepted. The RTE instruction accordingly returns to the next instruction after the instruction executed before the exception-handling sequence.

4.9.2 PC Value Pushed on Stack for Address Error and Invalid Instruction Exceptions

The program counter value pushed on the stack for an address error or invalid instruction exception differs depending on the conditions when the exception occurred.

4.10 Notes on Use of the Stack

If the stack pointer is set to an odd address, an address error will occur when the stack is accessed during interrupt handling or for a subroutine call. The stack pointer should always point to an even address. To keep the stack pointer pointing to an even address, a program should use word data size when saving or restoring registers to and from the stack.

In the @-SP or @SP+ addressing mode, the CPU performs word access even if the instruction specifies byte size. (This is not true in the @-Rn and @Rn+ addressing modes when Rn is a register from R6 to R0.)

Section 5 Interrupt Controller

5.1 Overview

The interrupt controller decides which interrupts to accept, and how to deal with multiple interrupts. It also decides whether an interrupt should be served by the CPU or by the data transfer controller (DTC). This section explains the features of the interrupt controller, describes its internal structure and control registers, and details the handling of interrupts.

For detailed information on the data transfer controller, see section 6, “Data Transfer Controller.”

5.1.1 Features

Three main features of the interrupt controller are:

- Interrupt priorities are user-programmable.
User programs can set priority levels from 7 (high) to 0 (low) in four interrupt priority registers (IPRs) for IRQ0, IRQ1 to IRQ3, and each of the on-chip supporting modules—for every interrupt, that is, except the nonmaskable interrupt (NMI). NMI has the highest priority level (8) and is normally always accepted. An interrupt with priority level 0 is always masked.
- Multiple interrupts on the same level are served in a default priority order.
Lower-priority interrupts remain pending until higher-priority interrupts have been handled.
- For most interrupts, software can select whether to have the interrupt served by the CPU or the on-chip data transfer controller (DTC).

User programs can make this selection by setting and clearing bits in four data transfer enable (DTE) registers. The data transfer controller can be started by any interrupts except NMI, the error interrupt (ERI) from the on-chip serial communication interface, and the overflow interrupts (FOVI and OVI) from the on-chip timers.

5.1.2 Block Diagram

Figure 5-1 shows the block configuration of the interrupt controller.

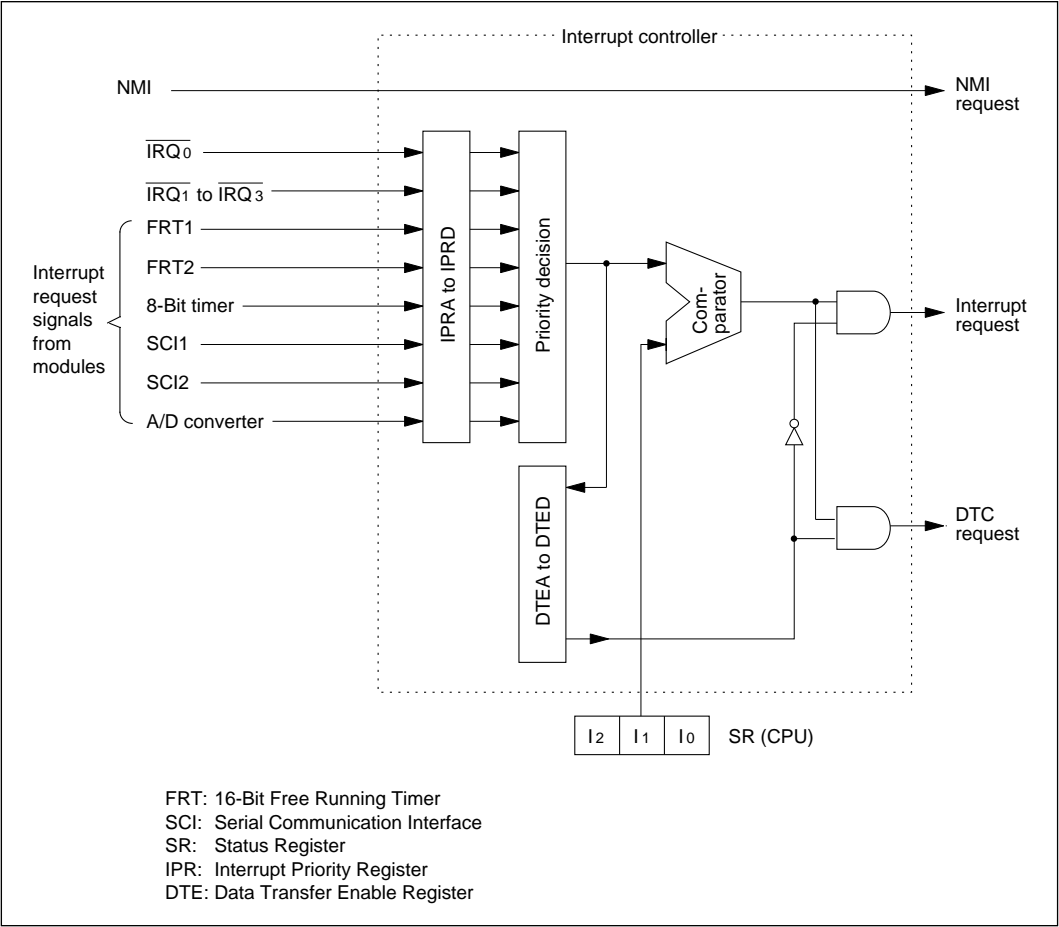


Figure 5-1 Interrupt Controller Block Diagram

5.1.3 Register Configuration

The four interrupt priority registers (IPRA to IPRD) and four data transfer enable registers (DTEA to DTED) are 8-bit registers located in the register field in page 0 of the address space. Table 5-1 lists their attributes.

Table 5-1 Interrupt Controller Registers

Name		Abbreviation	Read/Write	Address	Initial Value
Interrupt priority register	A	IPRA	R/W	H'FF00	H'00
	B	IPRB	R/W	H'FF01	H'00
	C	IPRC	R/W	H'FF02	H'00
	D	IPRD	R/W	H'FF03	H'00
Data transfer enable register	A	DTEA	R/W	H'FF08	H'00
	B	DTEB	R/W	H'FF09	H'00
	C	DTEC	R/W	H'FF0A	H'00
	D	DTED	R/W	H'FF0B	H'00

5.2 Interrupt Types

There are 23 distinct types of interrupts: 5 external interrupts originating off-chip and 18 internal interrupts originating in the on-chip supporting modules.

5.2.1 External Interrupts

The five external interrupts are NMI and IRQ0 to IRQ3.

NMI (NonMaskable Interrupt): This interrupt has the highest priority level (8) and cannot be masked. The input at the NMI pin is edge-sensed. A user program can select whether to have the interrupt occur on the rising edge or falling edge of the NMI input by setting or clearing the nonmaskable interrupt edge bit (NMIEG) in the NMI control register (NMICR).

In the NMI exception-handling sequence, the T (Trace) bit in the CPU status register (SR) is cleared to 0, and the interrupt mask level in I2 to I0 is set to 7, masking all other interrupts. The interrupt controller holds the NMI request until the NMI exception-handling sequence begins, then clears the NMI request, so if another interrupt is requested at the NMI pin during the NMI exception-handling sequence, the NMI exception-handling sequence will be carried out again.

NMI Control Register (NMICR)—H'FF1C

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	NMIEG
Initial value	1	1	1	1	1	1	1	0
Read/Write	R	R	R	R	R	R	R	R/W

The NMI control register (NMICR) is an 8-bit register that selects the edge of the NMI input signal which triggers a nonmaskable interrupt.

The NMICR is initialized to H'FE (falling edge) at a reset and in the hardware standby mode. It is not initialized in the software standby mode.

Bit 7 to 1—Reserved: These bits cannot be modified and are always read as 1.

Bit 0—Nonmaskable Interrupt Edge (NMIEG): This bit selects the valid edge of the NMI input signal.

Bit 0

NMIEG Description

0	A nonmaskable interrupt is generated on the falling edge of the NMI input signal.	(Initial state)
1	A nonmaskable interrupt is generated on the rising edge of the NMI input signal.	

IRQ Control Register (IRQCR)—H'FFFD

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	IRQ3E	IRQ2E	IRQ1E	IRQ0E
Initial value	1	1	1	1	0	0	0	0
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W

The IRQ control register (IRQCR) enables or disables external interrupts on an individual basis. When an interrupt is enabled, the corresponding pin in port 8 is used for interrupt request input.

The IRQ control register is initialized to H'F0 at a reset and in the hardware standby mode, disabling all four IRQ interrupt requests. It is not initialized in the software standby mode.

Bit 7 to 4—Reserved: These bits cannot be modified and are always read as 1.

Bit 3—Interrupt Request 3 Enable (IRQ3E): This bit determines the function of pin P83.

Bit 3

IRQ3E	Description
0	P83 is used as an input/output pin. (Initial state)
1	P83 is used for $\overline{\text{IRQ3}}$ input, regardless of the setting of P83DDR. (The CPU can also read the logic level of the P83 pin.)

Bit 2—Interrupt Request 2 Enable (IRQ2E): This bit determines the function of pin P82.

Bit 2

IRQ2E	Description
0	P82 is used as an input/output pin. (Initial state)
1	P82 is used for $\overline{\text{IRQ2}}$ input, regardless of the setting of P82DDR. (The CPU can also read the logic level of the P82 pin.)

Bit 1—Interrupt Request 1 Enable (IRQ1E): This bit determines the function of pin P81.

Bit 1

IRQ1E	Description
0	P81 is used as an input/output pin. (Initial state)
1	P81 is used for $\overline{\text{IRQ1}}$ input, regardless of the setting of P81DDR. (The CPU can also read the logic level of the P81 pin.)

Bit 0—Interrupt Request 0 Enable (IRQ0E): This bit determines the function of pin P80.

Bit 0

IRQ0E	Description
0	P80 is used as an input/output pin. (Initial state)
1	P80 is used for $\overline{\text{IRQ0}}$ input, regardless of the setting of P80DDR. (The CPU can also read the logic level of the P80 pin.)

IRQ0 (Interrupt Request 0): An IRQ0 interrupt can be requested by a low input to the $\overline{\text{IRQ0}}$ pin and/or a watchdog timer overflow. A low $\overline{\text{IRQ0}}$ input requests an IRQ0 interrupt if the interrupt request enable 0 bit (IRQ0E) in the IRQ control register is set to 1. $\overline{\text{IRQ0}}$ must be held low until the CPU accepts the interrupt. Otherwise the request will be ignored. A watchdog timer overflow requests an IRQ0 interrupt if the TME bit is set to 1 and the WT/IT bit is cleared to 0 in the watchdog timer's control/status register. Different interrupt vectors are provided for low IRQ0 input and watchdog timer overflow.

The IRQ0 interrupt can be assigned any priority level from 7 to 0 by setting the corresponding value in the upper four bits of IPRA. If bit 4 of data transfer enable register A (DTEA) is set to 1, an IRQ0 interrupt starts the data transfer controller. Otherwise the interrupt is served by the CPU.

In the CPU interrupt-handling sequence for IRQ0, the T bit of the status register is cleared to 0, and the interrupt mask level is set to the value in the upper four bits of IPRA.

IRQn (Interrupt Request n: n=1 to 3): An IRQn interrupt is requested by a high-to-low transition at the $\overline{\text{IRQn}}$ pin. The IRQn interrupt is enabled only when the interrupt request enable n bit (IRQnE) in the IRQ control register is set to 1.

The interrupt controller holds IRQ1 to IRQ3 requests until the corresponding exception-handling sequence begins, then clears the request. Contention among IRQ1 to IRQ3 is resolved when the CPU accepts the interrupt by taking the interrupt with the highest priority first and holding lower-priority interrupts pending.

The IRQn interrupts can be collectively assigned any priority level from 7 (high) to 0 (low) by setting the corresponding value in the lower four bits of IPRA. Whether they are served by the data transfer controller or CPU can be selected individually by bits 2 to 0 of data transfer enable register A (DTEA).

In the CPU interrupt-handling sequence for IRQn, the T bit of the CPU status register is cleared to 0, and the interrupt mask level is set to the value in the lower four bits of IPRA.

5.2.2 Internal Interrupts

Eighteen types of internal interrupts can be requested by the on-chip supporting modules. Each interrupt is separately vectored in the exception vector table, so it is not necessary for the user-coded interrupt handler routine to determine which type of interrupt has occurred.

Each of the internal interrupts can be enabled or disabled by setting or clearing an enable bit in the control register of the on-chip supporting module.

An interrupt priority level from 7 to 0 can be assigned to each on-chip supporting module by setting interrupt priority registers B to D. Within each module, different interrupts have a fixed priority order. For most of these interrupts, values set in data transfer enable registers B to D can select whether to have the interrupt served by the CPU or the data transfer controller.


In the CPU interrupt-handling sequence, the T bit of the CPU status register is cleared to 0, and the interrupt mask level in bits I2 to I0 is set to the value in the interrupt priority register.

5.2.3 Interrupt Vector Table

Table 5-2 lists the addresses of the exception vector table entries for each interrupt, and explains how their priority is determined. For the on-chip supporting modules, the priority level set in the interrupt priority register applies to the module as a whole: all interrupts from that module have the same priority level. A separate priority order is established among interrupts from the same module. If the same priority level is assigned to two or more modules and two interrupts are requested simultaneously from these modules, they are served in the priority order indicated in the rightmost column in table 5-2.

A reset clears the interrupt priority registers so that all interrupts except NMI start with priority level 0, meaning that they are unconditionally masked.

Table 5-2 Interrupts, Vectors, and Priorities

Interrupt		Assignable Priority Levels (Initial Level)	IPR Bits	Priority within Module	Vector Table Entry Address		Priority among Interrupts on Same Level*
					Minimum Mode	Maximum Mode	
NMI		8 (8)	—	—	H'16 - H'17	H'2C - H'2F	
IRQ ₀		7 to 0	IPRA	—	H'40 - H'41	H'80 - H'83	
WDT interval timer		(0)	bits 6 to 4		H'42 - H'43	H'84 - H'87	
IRQ ₁		7 to 0	IPRA	2	H'48 - H'49	H'90 - H'93	
IRQ ₂		(0)	bits 2 to 0	1	H'4A - H'4B	H'94 - H'97	
IRQ ₃				0	H'4C - H'4D	H'98 - H'9B	
16-bit ICI		7 to 0	IPRB	3	H'50 - H'51	H'A0 - H'A3	
FRT1	OCIA	(0)	bits 6 to 4	2	H'52 - H'53	H'A4 - H'A7	
	OCIB			1	H'54 - H'55	H'A8 - H'AB	
	FOVI			0	H'56 - H'57	H'AC - H'AF	
16-bit ICI		7 to 0	IPRB	3	H'58 - H'59	H'B0 - H'B3	
FRT2	OCIA	(0)	bits 2 to 0	2	H'5A - H'5B	H'B4 - H'B7	
	OCIB			1	H'5C - H'5D	H'B8 - H'BB	
	FOVI			0	H'5E - H'5F	H'BC - H'BF	
8-bit timer	CMIA	7 to 0	IPRC	2	H'60 - H'61	H'C0 - H'C3	
	CMIB	(0)	bits 6 to 4	1	H'62 - H'63	H'C4 - H'C7	
	OVI			0	H'64 - H'65	H'C8 - H'CB	
SCI1	ERI	7 to 0	IPRC	2	H'68 - H'69	H'D0 - H'D3	
	RXI	(0)	bits 2 to 0	1	H'6A - H'6B	H'D4 - H'D7	
	TXI			0	H'6C - H'6D	H'D8 - H'DB	
SCI2	ERI	7 to 0	IPRD	2	H'70 - H'71	H'E0 - H'E3	
	RXI	(0)	bits 6 to 4	1	H'72 - H'73	H'E4 - H'E7	
	TXI			0	H'74 - H'75	H'E8 - H'EB	
A/D converter	ADI	7 to 0 (0)	IPRD bits 2 to 0	—	H'78 - H'79	H'F0 - H'F3	Low

* If two or more interrupts are requested simultaneously, they are handled in order of priority level, as set in registers IPRA to IPRD. If they have the same priority level because they are requested from the same on-chip supporting module, they are handled in a fixed priority order within the module. If they are requested from different modules to which the same priority level is assigned, they are handled in the order indicated in the right-hand column.

5.3 Register Descriptions

5.3.1 Interrupt Priority Registers A to D (IPRA to IPRD)

IRQ0, IRQ1 to IRQ3, and the on-chip supporting modules are each assigned three bits in one of the four interrupt priority registers (IPRA to IPRD). These bits specify a priority level from 7 (high) to 0 (low) for interrupts from the corresponding source. The drawing below shows the configuration of the interrupt priority registers. Table 5-3 lists their assignments to interrupt sources.

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W

Note: Bits 7 and 3 are reserved. They cannot be modified and are always read as 0.

Table 5-3 Assignment of Interrupt Priority Registers

Register	Interrupt Request Source		Address
	Bits 6 to 4	Bits 2 to 0	
IPRA	IRQ0	IRQ1 – IRQ3	H'FF00
IPRB	16-bit FRT1	16-bit FRT2	H'FF01
IPRC	8-bit timer	SCI1	H'FF02
IPRD	SCI2	A/D converter	H'FF03

As table 5-3 indicates, each interrupt priority register specifies priority levels for two interrupt sources. A user program can assign desired levels to these interrupt sources by writing 000 in bits 6 to 4 or bits 2 to 0 to set priority level 0, for example, or 111 to set priority level 7.

A reset clears registers IPRA to IPRD to H'00, so all interrupts except NMI are initially masked.

When the interrupt controller receives one or more interrupt requests, it selects the request with the highest priority and compares its priority level with the interrupt mask level set in bits I2 to I0 in the CPU status register. If the priority level is higher than the mask level, the interrupt controller passes the interrupt request to the CPU (or starts the data transfer controller). If the priority level is lower than the mask level, the interrupt controller leaves the interrupt request pending until the interrupt mask is altered to a lower level or the interrupt priority is raised. Similarly, if it receives two interrupt requests with the same priority level, the interrupt controller determines their priority as explained in table 5-2 and leaves the interrupt request with the lower priority pending.

5.3.2 Timing of Priority Setting

The interrupt controller requires two system clock (ϕ) periods to determine the priority level of an interrupt. Accordingly, when an instruction modifies an instruction priority register, the new priority does not take effect until after the third state after the instruction has been executed.

5.4 Interrupt Handling Sequence

5.4.1 Interrupt Handling Flow

The interrupt-handling sequence follows the flowchart in figure 5-2. Note that address error, trace exception, and NMI requests bypass the interrupt controller's priority decision logic and are routed directly to the CPU.

1. Interrupt requests are generated by one or more on-chip supporting modules or external interrupt sources.
2. The interrupt controller checks the interrupt priorities set in the IPRA to IPRD and selects the interrupt with the highest priority. Interrupts with lower priorities remain pending. Among interrupts with the same priority level, the interrupt controller determines priority as explained in table 5-2.
3. The interrupt controller compares the priority level of the selected interrupt request with the mask level in the CPU status register (bits I2 to I0). If the priority level is equal to or less than the mask level, the interrupt request remains pending. If the priority level is higher than the mask level, the interrupt controller accepts the interrupt request and proceeds to the next step.
4. The interrupt controller checks the corresponding bit (if any) in the data transfer enable registers (DTEA to DTED). If this bit is set to 1, the data transfer controller is started. Otherwise, the CPU interrupt exception-handling sequence is started. When the data transfer controller is started, the interrupt request is cleared (except for interrupt requests from the serial communication interface, which are cleared by writing to the TDR or reading the RDR).

If the data transfer enable bit is cleared to 0 (or is nonexistent), the sequence proceeds as follows. For the case in which the data transfer controller is started, see section 6, “Data Transfer Controller.”

5. After the CPU has finished executing the current instruction, the program counter and status register (in minimum mode) or program counter, code page register, and status register (in maximum mode) are saved to the stack, leaving the stack in the condition shown in figure 5-3 (a) or (b). The program counter value saved on the stack is the address of the next instruction to be executed.
6. The T (Trace) bit of the status register is cleared to 0, and the priority level of the interrupt is copied to bits I2 to I0, thus masking further interrupts unless they have a higher priority level. When an NMI is accepted, the interrupt mask level in bits I2 to I0 is set to 7.
7. The interrupt controller generates the vector address of the interrupt, and the entry at this address in the exception vector table is read to obtain the starting address of the user-coded interrupt handling routine.

In step 7, the same difference between the minimum and maximum modes exists as in the reset handling sequence. In the minimum mode, one word is copied from the vector table to the program counter, then the interrupt-handling routine starts executing from the address indicated in the program counter. In the maximum mode, two words are read. The lower byte of the first word is copied to the code page register. The second word is copied to the program counter. The interrupt-handling routine starts executing from the address indicated in the code page register and program counter.

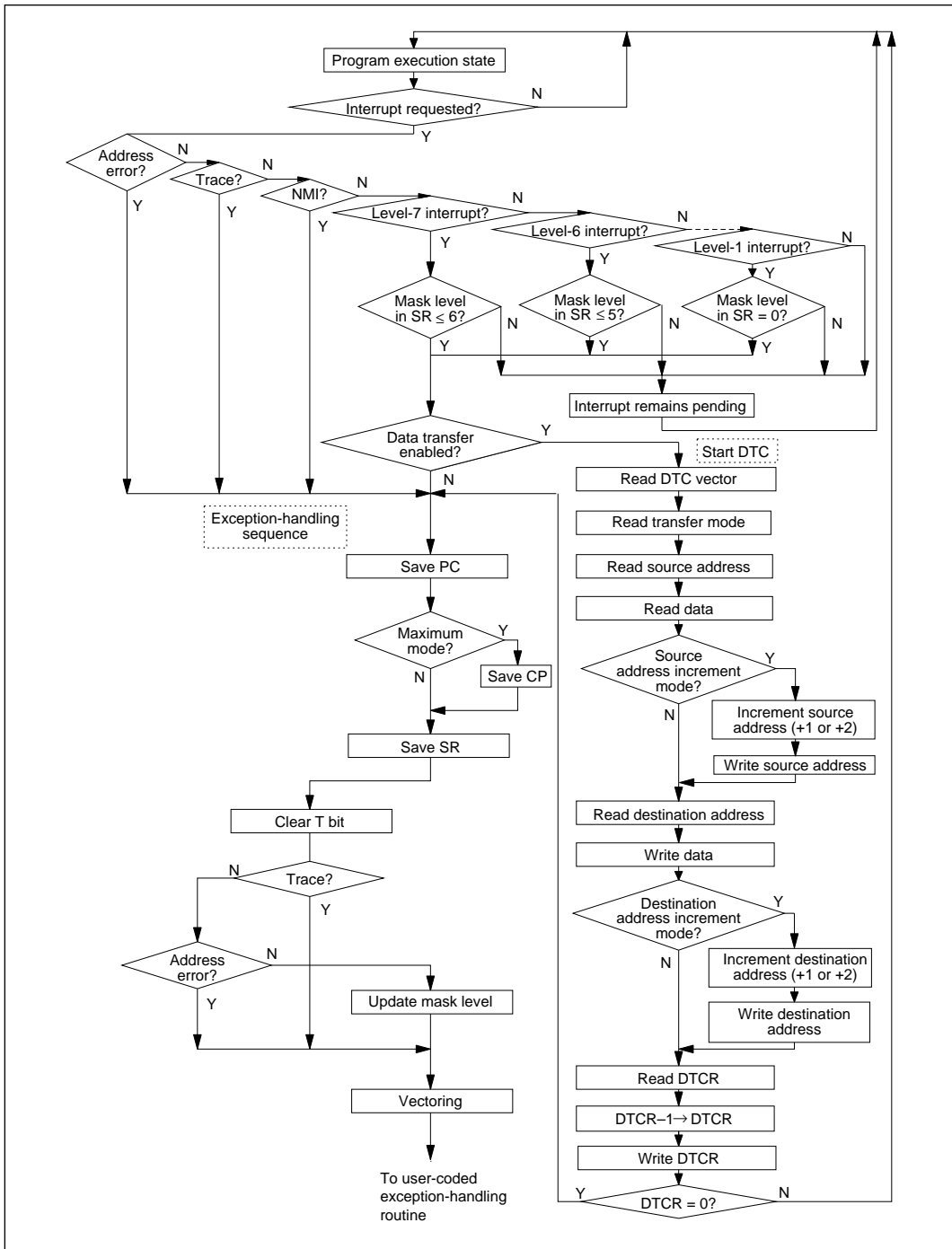


Figure 5-2 Interrupt Handling Flowchart

5.4.2 Stack Status after Interrupt Handling Sequence

Figure 5-3 (a) and (b) show the stack before and after the interrupt exception-handling sequence.

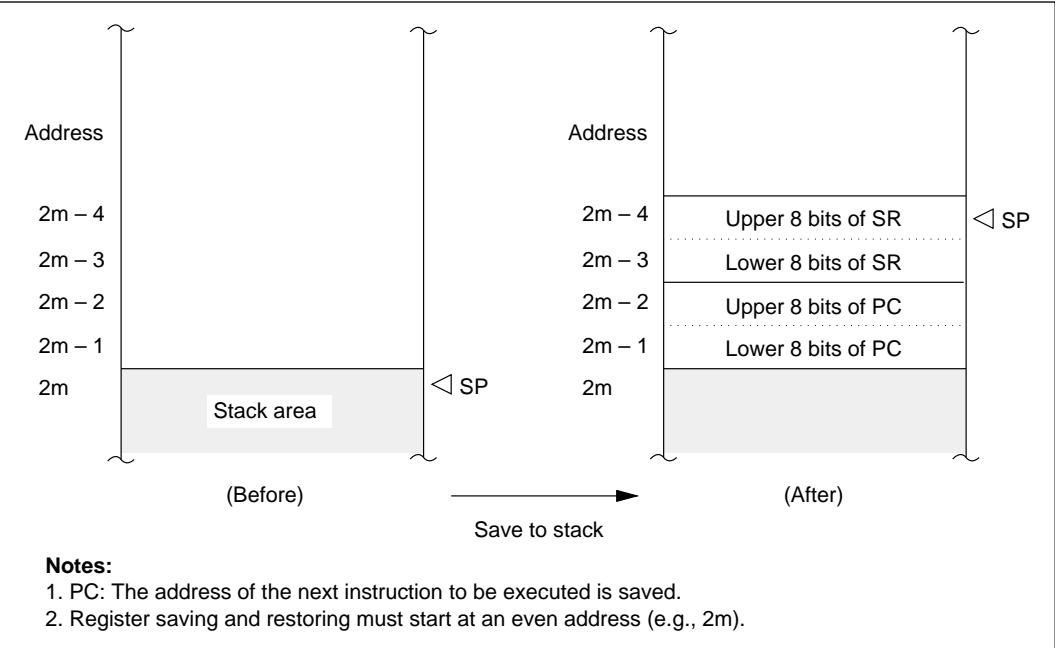


Figure 5-3 (a) Stack before and after Interrupt Exception-Handling (Minimum Mode)

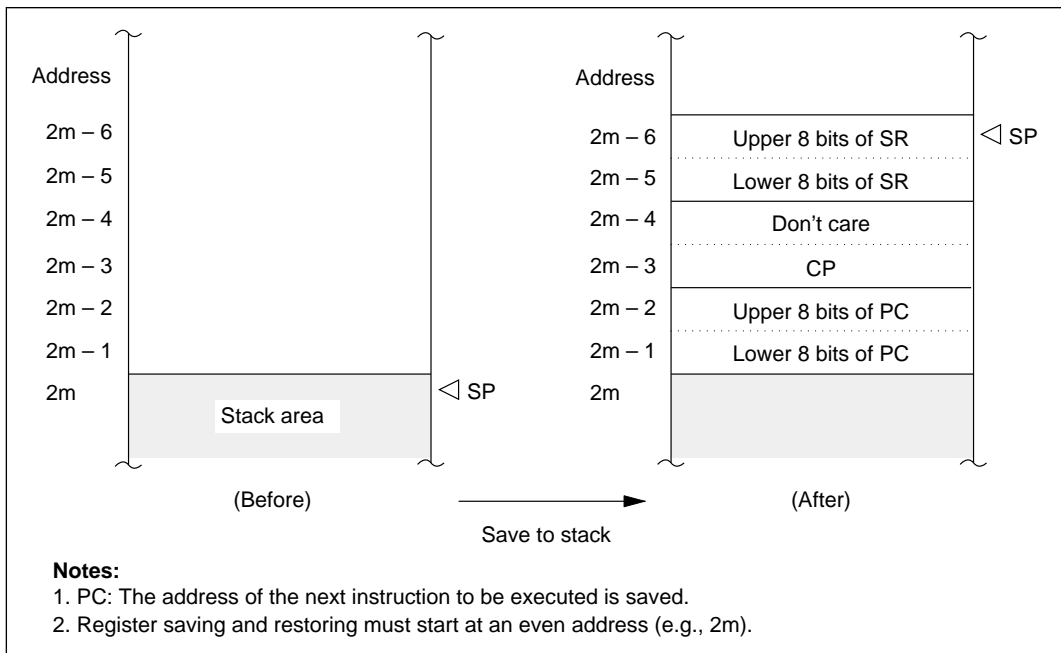


Figure 5-3 (b) Stack before and after Interrupt Exception-Handling (Maximum Mode)

5.4.3 Timing of Interrupt Exception-Handling Sequence

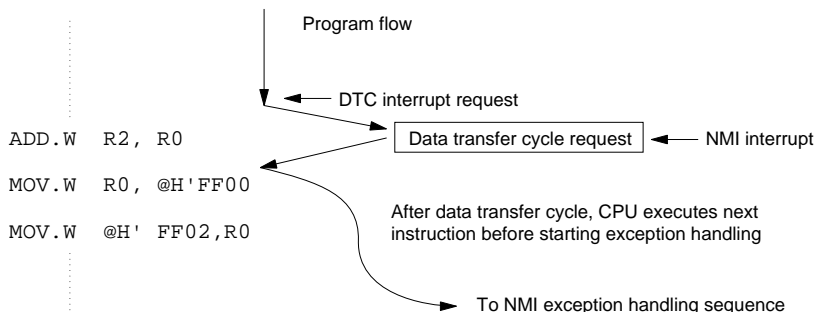
Figure 5-4 shows the timing of the exception-handling sequence for an interrupt in minimum mode when the user-coded interrupt handling routine starts at an even address.

Figure 5-5 shows the timing of the exception-handling sequence for an interrupt in maximum mode when the user-coded interrupt handling routine starts at an odd address.

5.5 Interrupts During Operation of the Data Transfer Controller

If an interrupt is requested during a DTC data transfer cycle, the interrupt is not accepted until the data transfer cycle has been completed and the next instruction has been executed. This is true even if the interrupt is an NMI. An example is shown below.

(Example)



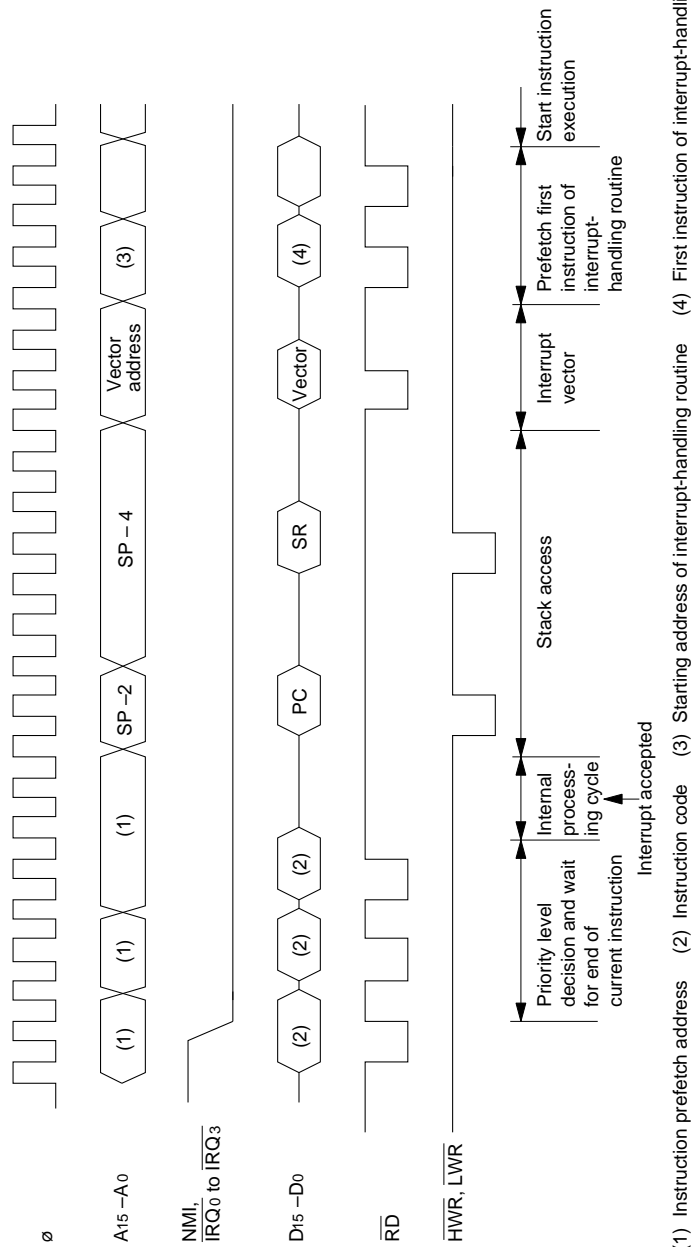
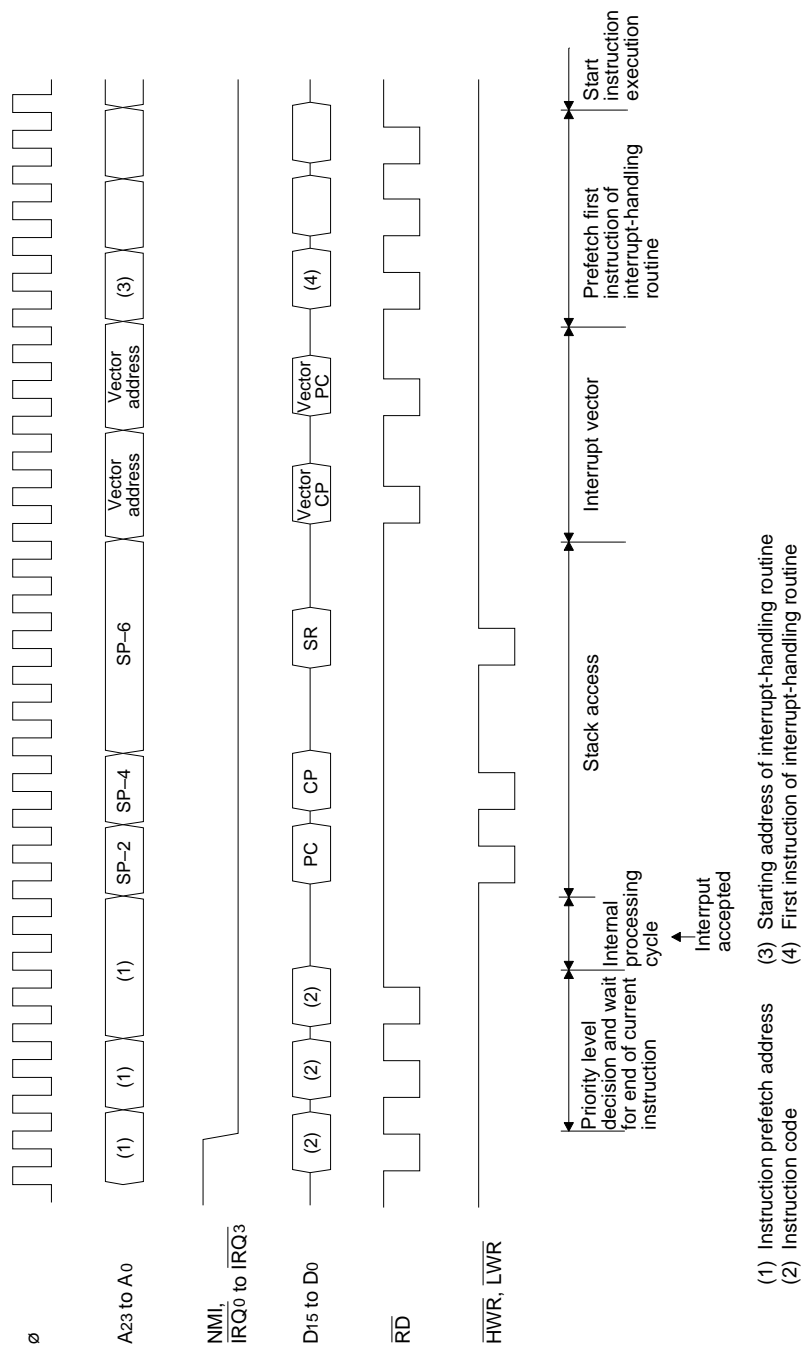


Figure 5-4 Interrupt Sequence (Minimum Mode)



Note: This timing chart applies to the maximum mode when the interrupt-handling routine starts at an odd address and the program, vector, and stack areas are all in a memory area accessed in two states via a 16-bit bus.

Figure 5-5 Interrupt Sequence (Maximum Mode)

5.6 Interrupt Response Time

Table 5-4 indicates the number of states that may elapse between the generation of an interrupt request and the execution of the first instruction of the interrupt-handling routine, assuming that the interrupt is not masked and not preempted by a higher-priority interrupt. Fastest interrupt service can be obtained by placing the program and stack in a memory area that can be accessed in two states via a 16-bit bus.

Table 5-4 Number of States before Interrupt Service

No.	Reason for Wait		Number of States		
			Minimum Mode	Maximum Mode	
1	Interrupt priority decision and comparison with mask level in status register		2 states	2 states	
2	Maximum number of states to completion of current instruction	Instruction is in 16-bit bus, 2-state access memory area	x (x = 38 for LDM instruction specifying all registers)		
		Instruction is in 8-bit bus, 3-state access memory area	y (y = 74 + 16m for LDM instruction specifying all registers)		
3	Saving of PC and SR or PC, CP, and SR and instruction prefetch	Stack is in 16-bit bus, 2-state access memory area	16	21	
		Stack is in 8-bit bus, 3-state access memory area	28 + 6m	41 + 10m	
Total		Stack is in 16-bit bus, 2-state access memory area	Instruction is in 16-bit bus, 2-state access memory area	18 + x (56)	23 + x (61)
		Stack is in 8-bit bus, 3-state access memory area	Instruction is in 8-bit bus, 3-state access memory area	18 + y (92 + 16m)	23 + y (97 + 16m)
			Stack is in 16-bit bus, 2-state access memory area	30 + 6m + x (68 + 6m)	43 + 10m + x (81 + 10m)
			Instruction is in 8-bit bus, 3-state access memory area	30 + 6m + y (104 + 22m)	43 + 10m + y (117 + 26m)

Note: m = Number of wait states inserted in memory access.

Figure in parentheses are for the LDM instruction specifying all registers.

Section 6 Data Transfer Controller

6.1 Overview

The H8/510 chip includes a data transfer controller (DTC) that can be started by designated interrupts to transfer data from a source address to a destination address located in page 0. These addresses include in particular the registers of the on-chip supporting modules and I/O ports. Typical uses of the DTC are to change the setting of a control register of an on-chip supporting module in response to an interrupt from that module, or to transfer data from memory to an I/O port or serial communication interface. Once set up, the transfer is interrupt-driven, so it proceeds independently of program execution, although program execution temporarily stops while each byte or word is being transferred.

6.1.1 Features

The main features of the DTC are listed below.

- The source address and destination address can be set anywhere in the 64-kbyte address space of page 0.
- The DTC can be programmed to transfer one byte or one word of data per interrupt.
- The DTC can be programmed to increment the source address and/or destination address after each byte or word is transferred.
- After transferring a designated number of bytes or words, the DTC generates a CPU interrupt with the vector of the interrupt source that started the DTC.
- This designated data transfer count can be set from 1 to 65,536 bytes or words.

6.1.2 Block Diagram

Figure 6-1 shows a block diagram of the DTC.

The four DTC control registers (DTMR, DTSR, DTDR, and DTCR) are invisible to the CPU, but corresponding information is kept in a register information table in memory. A separate table is maintained for each DTC interrupt type. When an interrupt requests DTC service, the DTC loads its control registers from the table in memory, transfers the byte or word of data, and writes any altered register information back to memory.

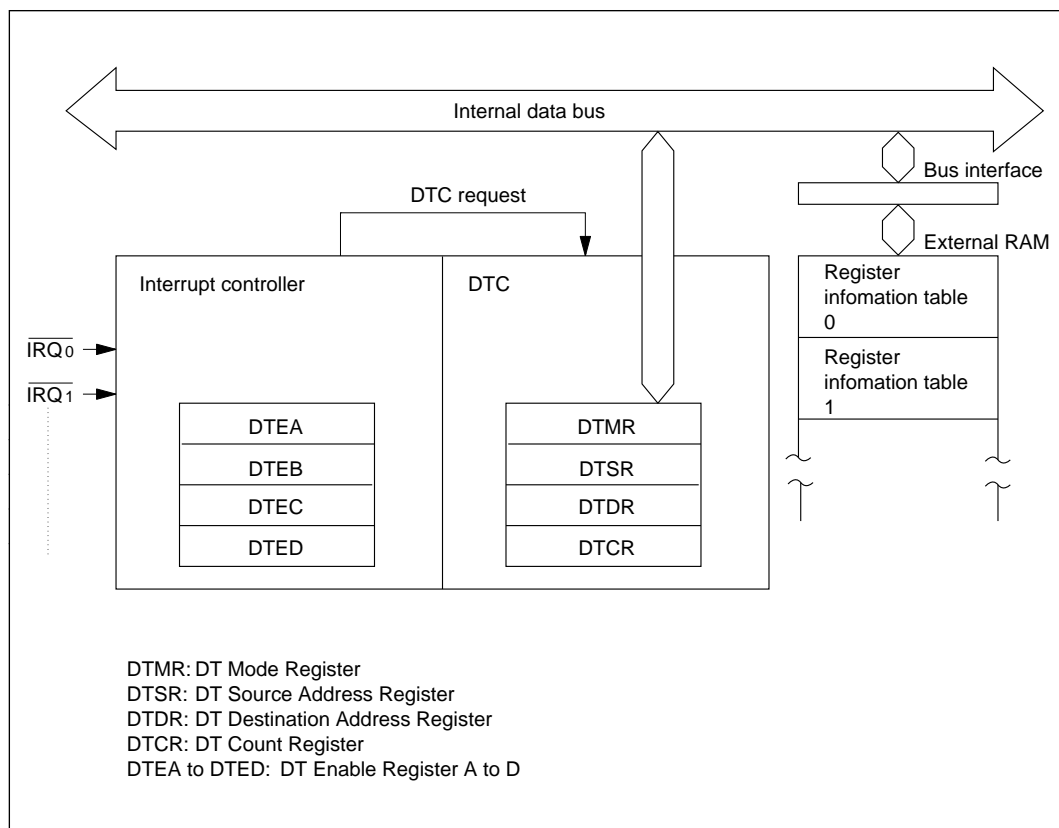


Figure 6-1 Block Diagram of Data Transfer Controller

6.1.3 Register Configuration

The four DTC control registers are listed in table 6-1. These registers are not located in the address space and cannot be written or read by the CPU. To set information in these registers, a program must write the information in a table in memory from which it will be loaded by the DTC.

Table 6-1 Internal Control Registers of the DTC

Name	Abbreviation	Read/Write
Data transfer mode register	DTMR	Disabled
Data transfer source address register	DTSR	Disabled
Data transfer destination address register	DTDR	Disabled
Data transfer count register	DTCR	Disabled

Starting of the DTC is controlled by the four data transfer enable registers, which are located in high addresses in page 0. Table 6-2 lists these registers.

Table 6-2 Data Transfer Enable Registers

Name		Abbreviation	Read/Write	Address	Initial Value
Data transfer enable register	A	DTEA	R/W	H'FF08	H'00
	B	DTEB	R/W	H'FF09	H'00
	C	DTEC	R/W	H'FF0A	H'00
	D	DTED	R/W	H'FF0B	H'00

6.2 Register Descriptions

6.2.1 Data Transfer Mode Register (DTMR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Sz	SI	DI	—	—	—	—	—	—	—	—	—	—	—	—	—
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer mode register is a 16-bit register, the first three bits of which designate the data size and specify whether to increment the source and destination addresses.

Bit 15—Sz (Size): This bit designates the size of the data transferred.

Bit 15

Sz	Description
0	Byte transfer
1	Word transfer* (two bytes at a time)

* For word transfer, the source and destination addresses must be even addresses.

Bit 14—SI (Source Increment): This bit specifies whether to increment to source address.

Bit 14

SI	Description
0	Source address is not incremented.
1	1) If Sz = 0: Source address is incremented by +1 after each data transfer. 2) If Sz = 1: Source address is incremented by +2 after each data transfer.

Bit 13—DI (Destination Increment): This bit specifies whether to increment to destination address.

Bit 13

DI	Description
0	Destination address is not incremented.
1	1) If Sz = 0: Destination address is incremented by +1 after each data transfer. 2) If Sz = 1: Destination address is incremented by +2 after each data transfer.

Bits 12 to 0—Reserved Bits: These bits are reserved.

6.2.2 Data Transfer Source Address Register (DTSR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer source register is a 16-bit register that designates the data transfer source address. For word transfer this must be an even address. In the maximum mode, this address is implicitly located in page 0.

6.2.3 Data Transfer Destination Register (DTDR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer destination register is a 16-bit register that designates the data transfer destination address. For word transfer this must be an even address. In the maximum mode, this address is implicitly located in page 0.

6.2.4 Data Transfer Count Register (DTCR)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

The data transfer count register is a 16-bit register that counts the number of bytes or words of data remaining to be transferred. The initial count can be set from 1 to 65,536. A register value of 0 designates an initial count of 65,536.

The data transfer count register is decremented automatically after each byte or word is transferred. When its value reaches 0, indicating that the designated number of bytes or words have been transferred, a CPU interrupt is generated with the vector of the interrupt that requested the data transfer.

6.2.5 Data Transfer Enable Registers A to D (DTEA to DTED)

These four registers designate whether an interrupt starts the DTC. The bits in these registers are assigned to interrupts as indicated in table 6-3. No bits are assigned to the NMI, FOVI, OVI, and ERI interrupts, which cannot request data transfers.

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 6-3 Assignment of Data Transfer Enable Registers

Register	Interrupt Source				Interrupt Source					
	Module	Bits 7 to 4				Module	Bits 3 to 0			
		7	6	5	4		3	2	1	0
DTEA	$\overline{\text{IRQ}}_0$	—	—	—	IRQ_0	$\overline{\text{IRQ}}_1$ to $\overline{\text{IRQ}}_3$	—	IRQ_3	IRQ_2	IRQ_1
DTEB	16-Bit FRT1	—	OCIB	OCIA	ICI	16-Bit FRT2	—	OCIB	OCIA	ICI
DTEC	8-Bit timer	—	—	CMIB	CMIA	SCI1	—	TXI	RXI	—
DTED	SCI2	—	TXI	RXI	—	A/D converter	—	—	—	ADI

Note: Bits marked “—” should always be cleared to 0.

If the bit for a certain interrupt is set to 1, that interrupt is regarded as a request for DTC service. If the bit is cleared to 0, the interrupt is regarded as a CPU interrupt request.

Only the 16 interrupts indicated in table 6-3 can request DTC service. DTE bits not assigned to any interrupt (indicated by “—” in table 6-3) should be left cleared to 0.

Note on Timing of DTE Modifications: The interrupt controller requires two system clock (ϕ) periods to determine the priority level of an interrupt. Accordingly, when an instruction modifies a data transfer enable register, the new setting does not take effect until after the next instruction has been executed.

6.3 Data Transfer Operation

6.3.1 Data Transfer Cycle

When started by an interrupt, the DTC executes the following data transfer cycle:

1. From the DTC vector table, the DTC reads the address at which the register information table for that interrupt is located in memory.
2. The DTC loads the data transfer mode register and source address register from this table and reads the data (one byte or word) from the source address.
3. If so specified in the mode register, the DTC increments the source address register and writes the new source address back to the table in memory.
4. The DTC loads the data transfer destination address register and writes the byte or word of data to the destination address.
5. If so specified in the mode register, the DTC increments the destination address register and writes the new destination address back to the table in memory.
6. The DTC loads the data transfer count register from the table in memory, decrements the data count, and writes the new count back to memory.
7. If the data transfer count is now 0, the DTC generates a CPU interrupt. The interrupt vector is the vector of the interrupt type that started the DTC.

At an appropriate point during this procedure the DTC also clears the interrupt request by clearing the corresponding flag bit in the status register of the on-chip supporting module to 0.

But the DTC does not clear the data transfer enable bit in the data transfer enable register. This action, if necessary, must be taken by the user-coded interrupt-handling routine invoked at the end of the transfer.

The data transfer cycle is shown in a flowchart in figure 6-2.

For the steps from the occurrence of the interrupt up to the start of the data transfer cycle, see section 5.4.1, “Interrupt Handling Flow.”

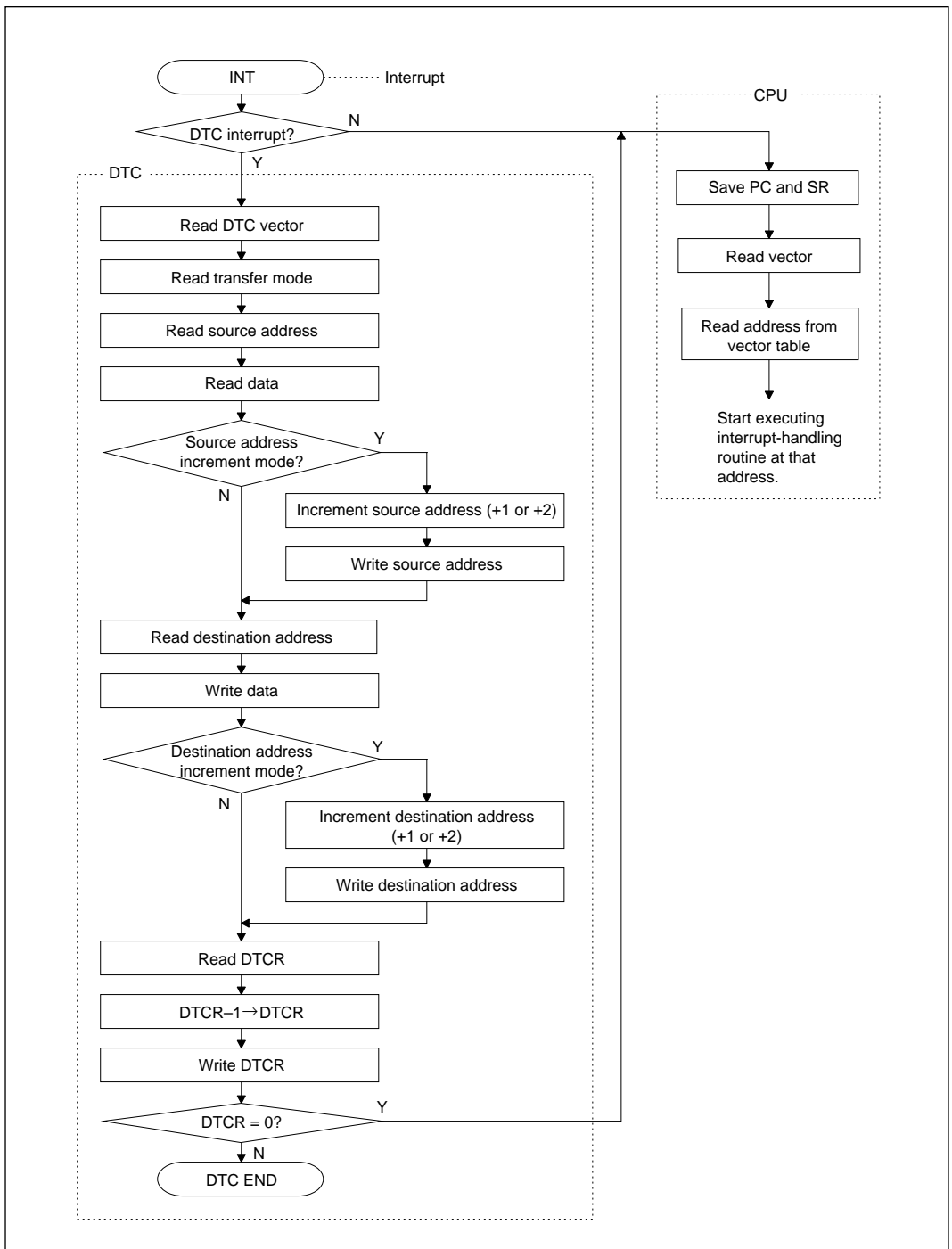


Figure 6-2 Flowchart of Data Transfer Cycle

6.3.2 DTC Vector Table

The DTC vector table is located immediately following the exception vector table at the beginning of page 0 in memory. For each interrupt that can request DTC service, the DTC vector table provides a pointer to an address in memory where the table of DTC control register information for that interrupt is stored. The register information tables can be placed in any available locations in page 0.

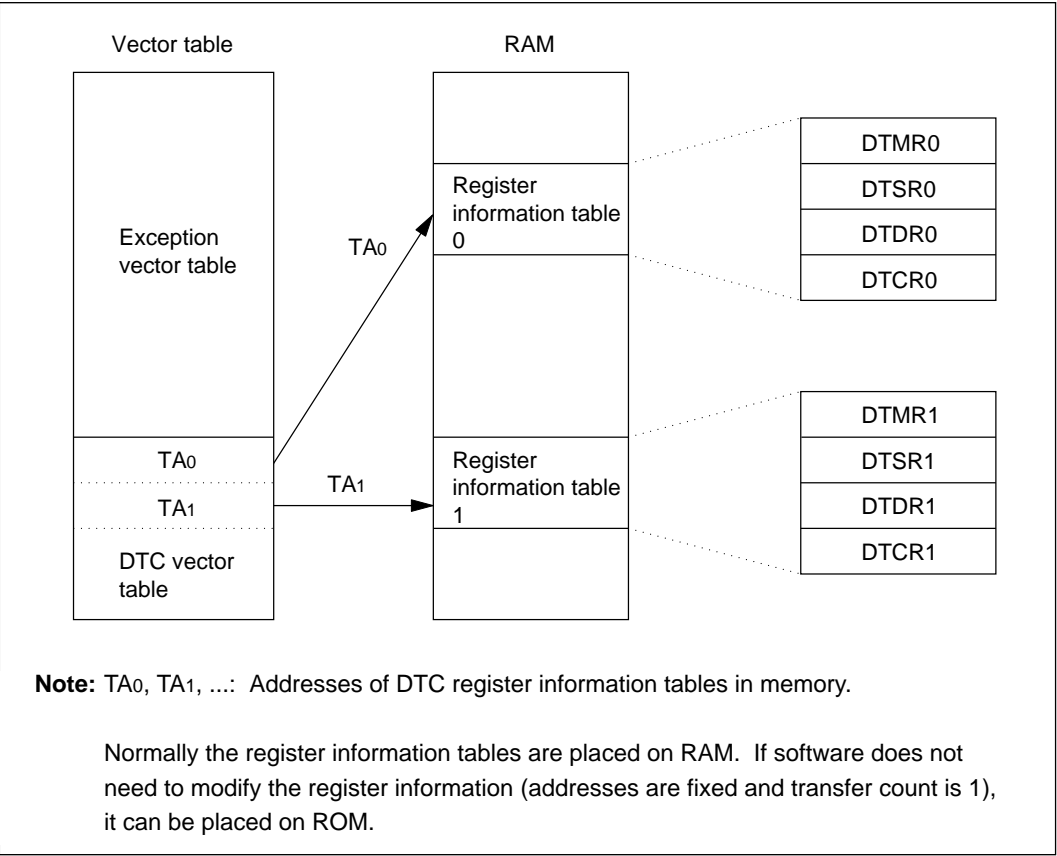


Figure 6-3 DTC Vector Table

In minimum mode, each entry in the DTC vector table consists of two bytes, pointing to an address in page 0. In maximum mode, for compatibility reasons, each DTC vector table entry consists of four bytes but the first two bytes are ignored; the last two bytes point to an address which is implicitly assumed to be in page 0, regardless of the current page specifications.

Figure 6-4 shows one DTC vector table entry in minimum and maximum mode.

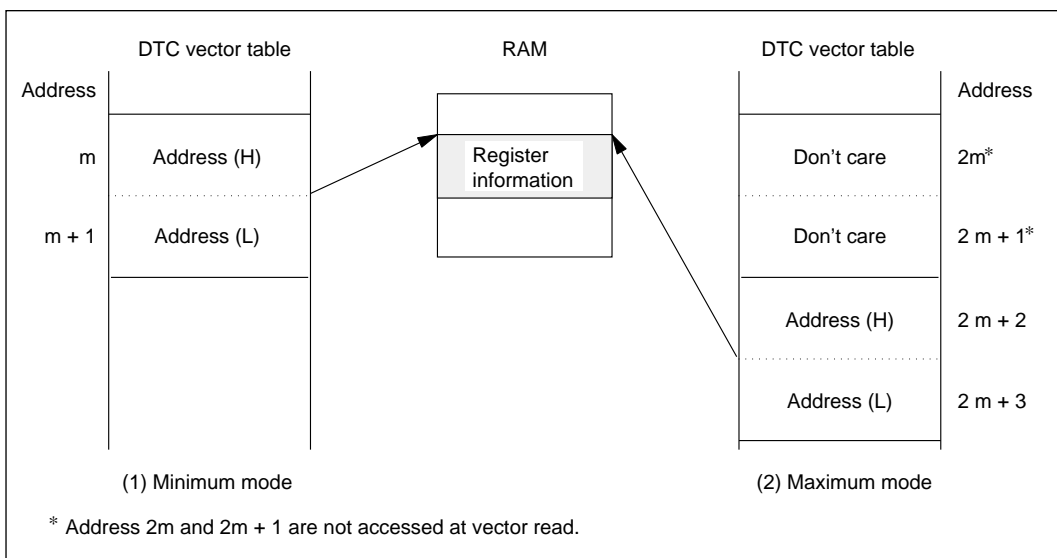


Figure 6-4 DTC Vector Table Entry

Table 6-4 lists the addresses of the entries in the DTC vector table for each interrupt.

Table 6-4 Addresses of DTC Vectors

		Address of DTC Vector	
Interrupt		Minimum Mode	Maximum Mode
IRQ ₀		H'00C0 - H'00C1	H'0180 - H'0183
IRQ ₁		H'00C8 - H'00C9	H'0190 - H'0193
IRQ ₂		H'00CA - H'00CB	H'0194 - H'0197
IRQ ₃		H'00CC - H'00CD	H'0198 - H'019B
16-Bit	ICI	H'00D0 - H'00D1	H'01A0 - H'01A3
FRT1	OCIA	H'00D2 - H'00D3	H'01A4 - H'01A7
	OCIB	H'00D4 - H'00D5	H'01A8 - H'01AB
16-Bit	ICI	H'00D8 - H'00D9	H'01B0 - H'01B3
FRT2	OCIA	H'00DA - H'00DB	H'01B4 - H'01B7
	OCIB	H'00DC - H'00DD	H'01B8 - H'01BB
8-Bit timer	CMIA	H'00E0 - H'00E1	H'01C0 - H'01C3
	CMIB	H'00E2 - H'00E3	H'01C4 - H'01C7
SCI1	RXI	H'00EA - H'00EB	H'01D4 - H'01D7
	TXI	H'00EC - H'00ED	H'01D8 - H'01DB
SCI2	RXI	H'00F2 - H'00F3	H'01E4 - H'01E7
	TXI	H'00F4 - H'00F5	H'01E8 - H'01EB
A/D converter	ADI	H'00F8 - H'00F9	H'01F0 - H'01F3

6.3.3 Location of Register Information in Memory

For each interrupt, the DTC control register information is stored in four consecutive words in memory in the order shown in figure 6-5.

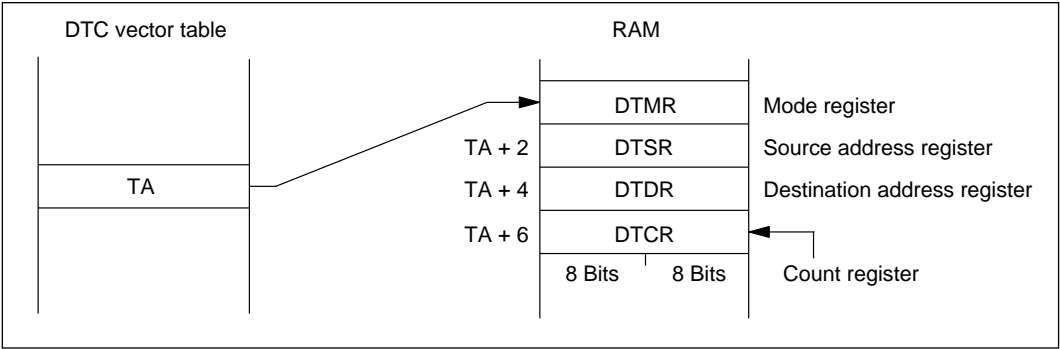


Figure 6-5 Order of Register Information

6.3.4 Length of Data Transfer Cycle

Table 6-5 lists the number of states required per data transfer, assuming that the DTC control register information is stored in a memory area accessed in two states via a 16-bit bus.

Table 6-5 Number of States per Data Transfer

Increment Mode		16-Bit Bus, 2-State Access ↔ Module or I/O Memory Area Register		8-Bit Bus, 3-State Access ↔ Module or I/O Memory Area Register	
Source (SI)	Destination (DI)	Byte Transfer	Word Transfer	Byte Transfer	Word Transfer
0	0	31	34	32	38
0	1	33	36	34	40
1	0	33	36	34	40
1	1	35	38	36	42

Note: Numbers in the table are the number of states.

The values in table 6-5 are calculated from the formula:

$$N = 26 + 2 \times SI + 2 \times DI + MS + MD$$

Where MS and MD have the following meanings:

MS: Number of states for reading source data

MD: Number of states for writing destination data

The values of MS and MD depend on the data location as follows:

1. Byte or word data in 16-bit bus, 2-state-access memory area: ⇨ 2 states
2. Byte data in 8-bit bus, 3-state-access memory area or register field: ⇨ 3 states
3. Word data in 8-bit bus, 3-state-access memory area or register field: ⇨ 6 states

If the DTC control register information is stored in the RAM, $20 + 4 \times SI + 4 \times DI$ must be added to the values in table 6-5.

The values given above do not include the time between the occurrence of the interrupt request and the starting of the DTC. This time includes two states for the interrupt controller to check priority and a variable wait until the end of the current CPU instruction. At maximum, this time equals the sum of the values indicated for items No. 1 and 2 in table 6-6.

If the data transfer count is 0 at the end of a data transfer cycle, the number of states from the end of the data transfer cycle until the first instruction of the user-coded interrupt-handling routine is executed is the value given for item No. 3 in table 6-6.

Table 6-6 Number of States before Interrupt Service

No.	Reason for Wait	Number of States	
		Minimum Mode	Maximum Mode
1	Interrupt priority decision and comparison with mask level in status register	2 states	2 states
2	Maximum number of states to completion of current instruction	Instruction is in 16-bit bus,	38
		2-state access memory area (LDM instruction specifying all registers)	
		Instruction is in 8-bit bus,	74 + 16m
3	Saving of PC and SR or PC, CP, and SR, and instruction prefetch	3-state access memory area (LDM instruction specifying all registers)	
		Stack is in 16-bit bus,	16
		2-state access memory area	21
		Stack is in 8-bit bus,	28 + 6m
		3-state access memory area	41 + 10m

m: Number of wait states inserted in memory access

6.4 Procedure for Using the DTC

A program that uses the DTC to transfer data must do the following:

1. Set the appropriate DTMR, DTSR, DTDR, and DTCR register information in the memory location indicated in the DTC vector table.
2. Set the data transfer enable bit of the pertinent interrupt to 1, and set the priority of the interrupt source (in the interrupt priority register) and the interrupt mask level (in the CPU status register) so that the interrupt can be accepted.
3. Set the interrupt enable bit in the control register for the interrupt source. (For IRQ0 and IRQ1, the control register is the port 1 control register, P1CR).

Following these preparations, the DTC will be started each time the interrupt occurs. When the number of bytes or words designated by the DTCR value have been transferred, after transferring the last byte or word, the DTC generates a CPU interrupt.

The user-coded interrupt-handling routine must take action to prepare for or disable further DTC data transfer: by readjusting the data transfer count, for example, or clearing the interrupt enable bit. If no action is taken, the next interrupt of the same type will start the DTC with an initial data transfer count of 65,536.

6.5 Example

1. Purpose

To receive 128 bytes of serial data via serial communication interface1.

2. Conditions

- Operating mode: Minimum mode
- Received data are to be stored in consecutive addresses starting at H'FC00.
- DTC control register information for the RXI interrupt is stored at addresses H'FB80 to H'FB87.
- Accordingly, the DTC vector table contains H'FB at address H'00EA and H'80 at address H'00EB.
- The desired interrupt mask level in the CPU status register is 4, and the desired SCI1 interrupt priority level is 5.

3. Procedure

1. The user program sets DTC control register information as shown in table 6-7.

Table 6-7 DTC Control Register Information Set in RAM

Register	Description	Value Set
DTMR	Byte transfer	
	Source address fixed	H'2000
	Increment destination address	
DTSR	Address of SCI1 receive data register	H'FECD
DTDR	Address H'FC00	H'FC00
DTCR	Number of bytes to be received: 128	H'0080

2. The program sets the RXI (SCI Receive Interrupt) bit in the data transfer enable register (DTEC) to 1.
3. The program sets the interrupt mask in the CPU status register to 4, and the SCI interrupt priority in bits 2 to 0 of interrupt priority register IPRC to 5.
4. The program sets SCI1 to the appropriate receive mode, and sets the receive interrupt enable (RIE) bit in the serial control register (SCR) to 1 to enable receive interrupts.
5. Thereafter, each time SCI1 receives one byte of data, it requests an RXI interrupt, which the interrupt controller directs toward the DTC. The DTC transfers the byte from the SCI1's receive data register (RDR) into RAM, and clears the interrupt request before ending.

6. When 128 bytes have been transferred (DTCR = 0), the DTC generates a CPU interrupt. The interrupt type is RXI from SCI1.
7. The user-coded RXI interrupt-handling routine processes the received data and disables further data transfer (by clearing the RIE bit, for example).

Figure 6-6 shows the DTC vector table and data in RAM for this example.

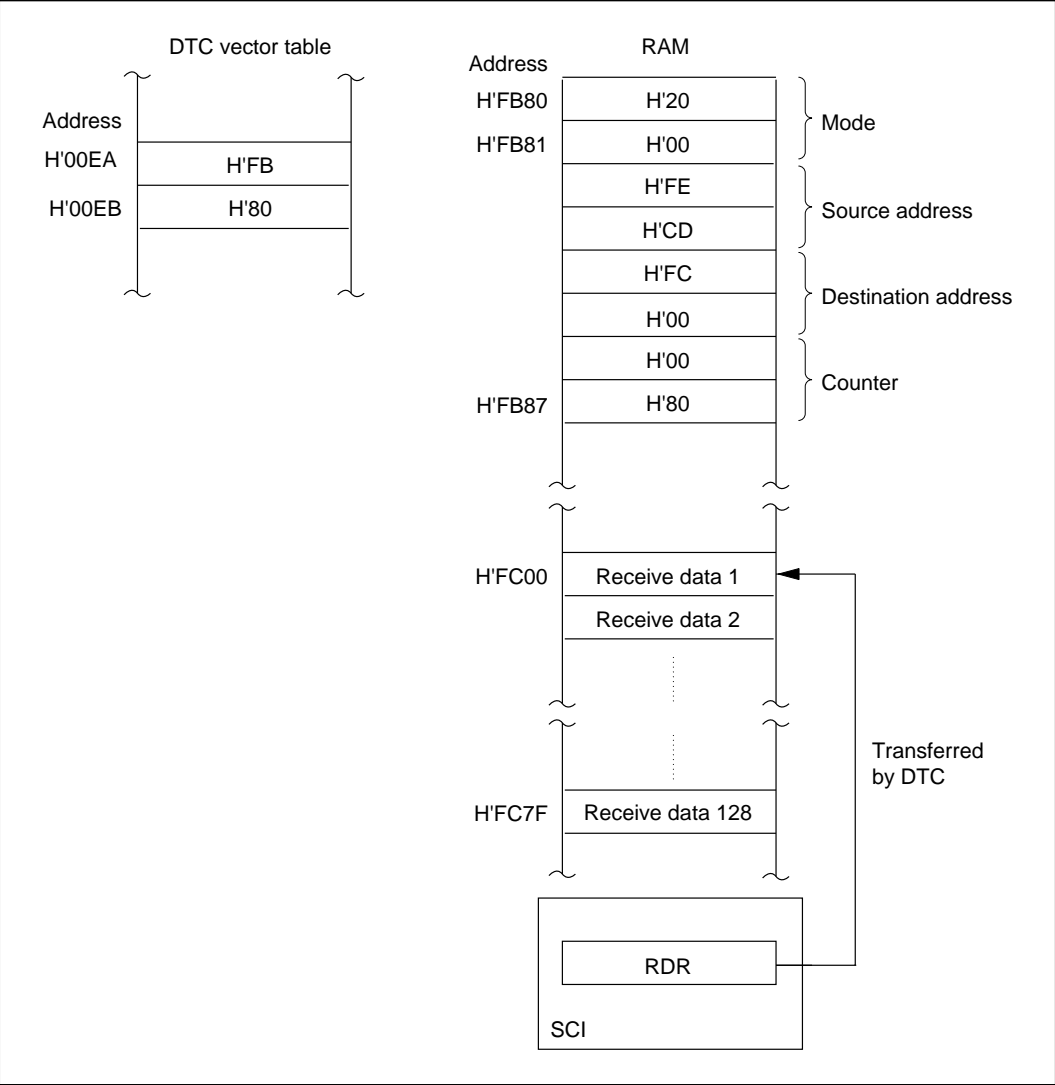


Figure 6-6 Use of DTC to Receive Data via Serial Communication Interface

Section 7 Wait-State Controller

7.1 Overview

To simplify interfacing to low-speed external devices, the H8/510 has an on-chip wait-state controller (WSC) that can insert wait states (Tw) to prolong bus cycles.

The wait-state function can be used in CPU and DTC access cycles to the three-state-access memory area. It is not used in access to the two-state-access memory area or on-chip supporting modules. The Tw states are inserted between the T2 state and T3 state in the bus cycle. The number of wait states can be selected by a value set in the wait-state control register (WCR), or by holding the $\overline{\text{WAIT}}$ pin Low for the required interval.

7.1.1 Features

The main features of the wait-state controller are:

- Selection of three operating modes
Programmable wait mode, pin wait mode, or pin auto-wait mode
- 0, 1, 2, or 3 wait states can be inserted.
And in the pin wait mode, 4 or more states can be inserted by holding the $\overline{\text{WAIT}}$ pin Low.

7.1.2 Block Diagram

Figure 7-1 shows a block diagram of the wait-state controller.

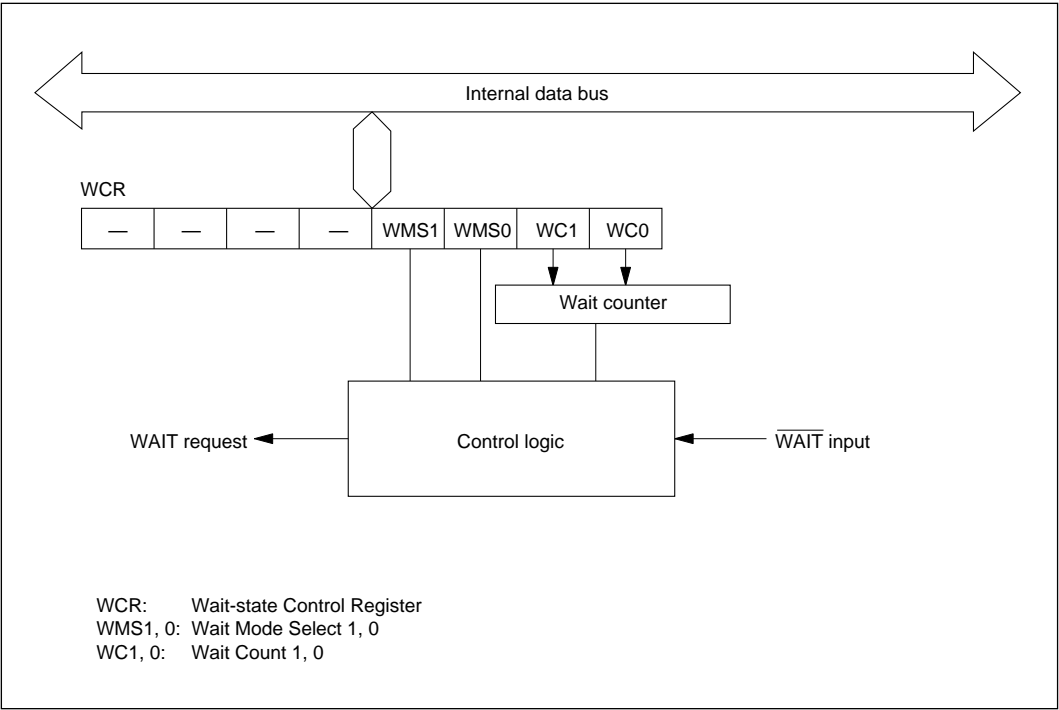


Figure 7-1 Block Diagram of Wait-State Controller

7.1.3 Register Configuration

The wait-state controller has one control register: the wait-state control register described in table 7-1.

Table 7-1 Register Configuration

Name	Abbreviation	Read/Write	Initial Value	Address
Wait-state control register	WCR	R/W	H'F3	H'FF14

7.2 Wait-State Control Register

The wait-state control register (WCR) is an 8-bit register that specifies the wait mode and the number of wait states to be inserted. A reset initializes the WCR to specify the programmable wait mode with three wait states. The WCR is not initialized in the software standby mode.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	WMS1	WMS0	WC1	WC0
Initial value	1	1	1	1	0	0	1	1
Read/Write	—	—	—	—	R/W	R/W	R/W	R/W

Bits 7 to 4—Reserved: These bits cannot be modified and are always read as 1.

Bits 3 and 2—Wait Mode Select 1 and 0 (WMS1 and WMS0): These bits select the wait mode as shown below.

Bit 3 WMS1	Bit 2 WMS0	Description
0	0	Programmable wait mode (Initial value)
0	1	No wait states are inserted, regardless of the wait count.
1	0	Pin wait mode
1	1	Pin auto-wait mode

Bits 1 and 0—Wait Count (WC1 and WC0): These bits specify the number of wait states to be inserted.

Wait states are inserted only in bus cycles in which the CPU or DTC accesses the three-state-access memroy area.

Bit 1 WC1	Bit 0 WC0	Description
0	0	No wait states are inserted, except in pin wait mode.
0	1	1 Wait state in inserted.
1	0	2 Wait states are inserted.
1	1	3 Wait states are inserted. (Initial value)

7.3 Operation in Each Wait Mode

Table 7-2 summarizes the operation of the three wait modes.

Table 7-2 Wait Modes

Mode	$\overline{\text{WAIT}}$ Pin Function	Insertion Conditions	Number of Wait States Inserted
Programmable wait mode WMS1 = 0 WMS0 = 0	Disabled	Inserted on access to three-state-access memory area	1 to 3 wait states are inserted, as specified by bits WC0 and WC1.
Pin wait mode WMS1 = 1 WMS0 = 0	Enabled	Inserted on access to three-state-access memory area	0 to 3 wait states are inserted, as specified by bits WC0 and WC1, plus additional wait states while the $\overline{\text{WAIT}}$ pin is held Low.
Pin auto-wait mode WMS1 = 1 WMS0 = 1	Enabled	Inserted on access to three-state-access memory area if the $\overline{\text{WAIT}}$ pin is Low	1 to 3 wait states are inserted, as specified by bits WC0 and WC1.

7.3.1 Programmable Wait Mode

The programmable wait mode is selected when WMS1 = 0 and WMS0 = 0.

Whenever the CPU or DTC accesses an address in the three-state-access memory area, the number of wait states set in bits WC1 and WC0 are inserted. The $\overline{\text{WAIT}}$ pin is not used for wait control; it is available as an I/O pin (P30).

Figure 7-2 shows the timing of the operation in this mode when the wait count is 1 ($WC1 = 0$, $WC0 = 1$).

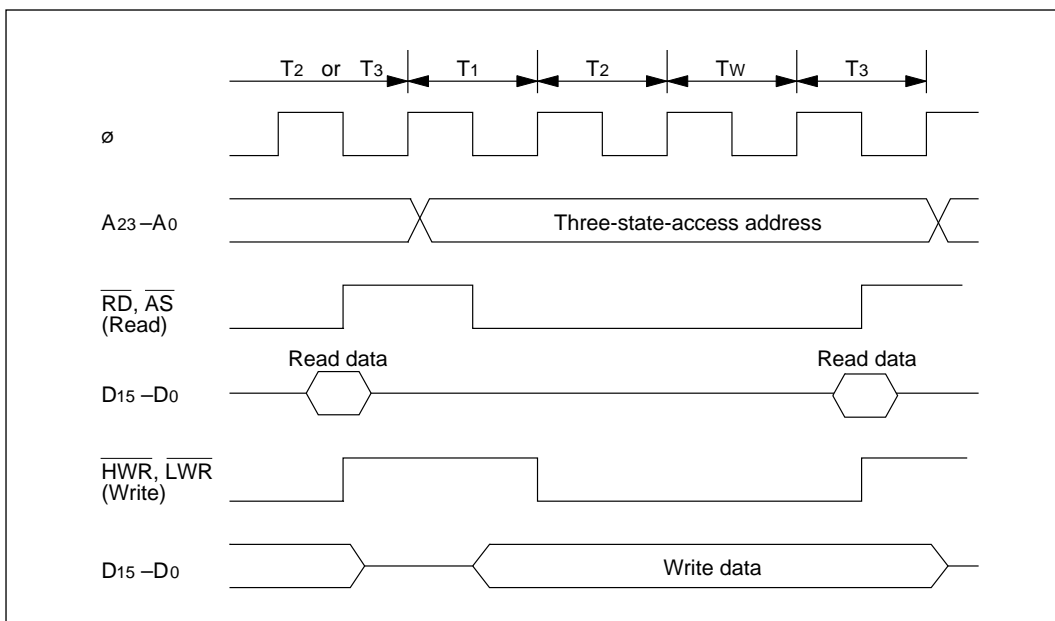


Figure 7-2 Programmable Wait Mode

7.3.2 Pin Wait Mode

The pin wait mode is selected when $WMS1 = 1$ and $WMS0 = 0$.

In this mode the \overline{WAIT} function of the $P30 / \overline{WAIT}$ pin is used automatically.

The number of wait states indicated by bits $WC1$ and $WC0$ are inserted into any bus cycle in which the CPU or DTC accesses an address in the three-state-access memory area. In addition, wait states continue to be inserted as long as the \overline{WAIT} pin is held low. In particular, if the wait count is 0 but the \overline{WAIT} pin is low at the rising edge of the system clock in the T_2 state, wait states are inserted until the \overline{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.

Figure 7-3 shows the timing of the operation in this mode when the wait count is 1 ($WC1 = 0$, $WC0 = 1$) and the $\overline{\text{WAIT}}$ pin is held low to insert one additional wait state.

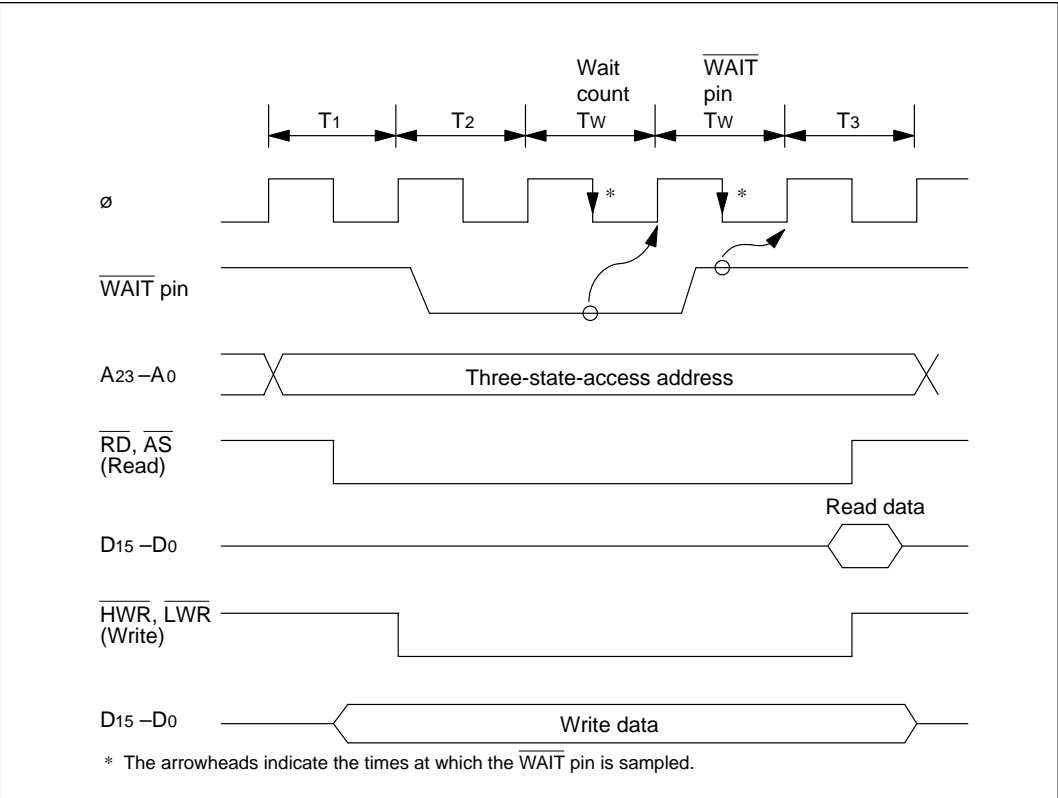


Figure 7-3 Pin Wait Mode

7.3.3 Pin Auto-Wait Mode

The pin auto-wait mode is selected when $WMS1 = 1$ and $WMS0 = 1$.

In this mode the \overline{WAIT} function of the P30 / \overline{WAIT} pin is used automatically.

In this mode, the number of wait states indicated by bits WC1 and WC0 are inserted, but only if there is a Low input at the \overline{WAIT} pin.

Figure 7-4 shows the timing of this operation when the wait count is 1.

In the pin auto-wait mode, the \overline{WAIT} pin is sampled only once, on the falling edge of the ϕ clock in the T2 state. If the \overline{WAIT} pin is Low at this time, the wait-state controller inserts the number of wait states indicated by bits WC1 and WC0. The \overline{WAIT} pin is not sampled during the T_w and T_3 states, so no additional wait states are inserted even if the \overline{WAIT} pin continues to be held Low.

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

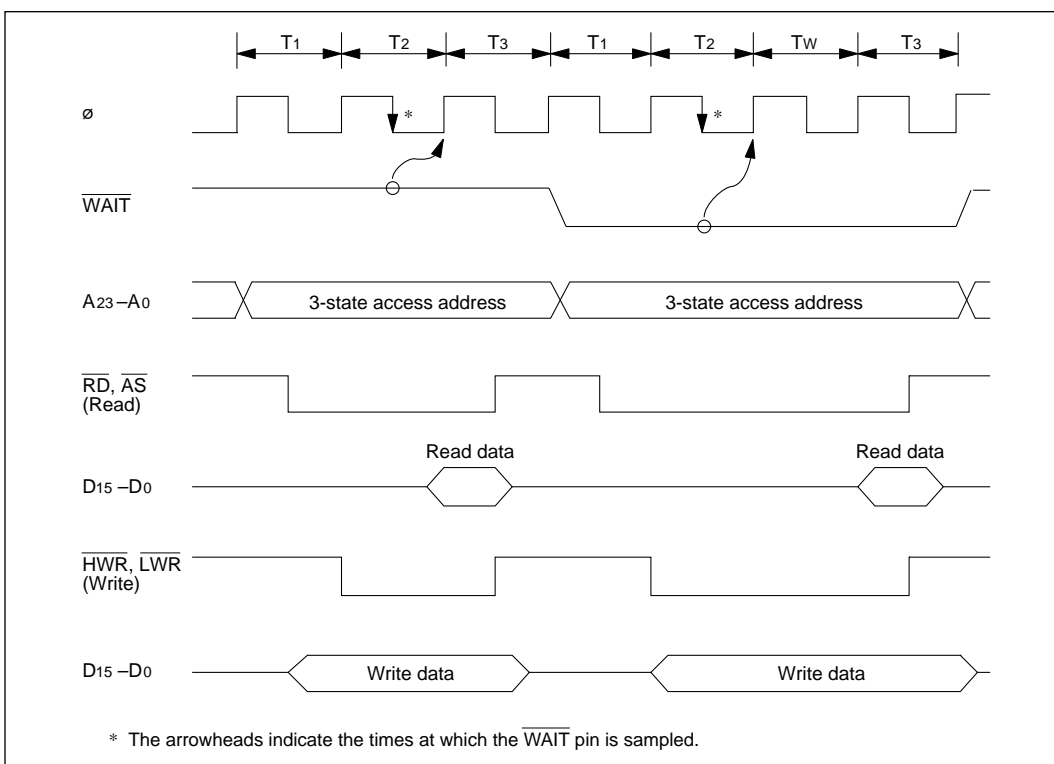


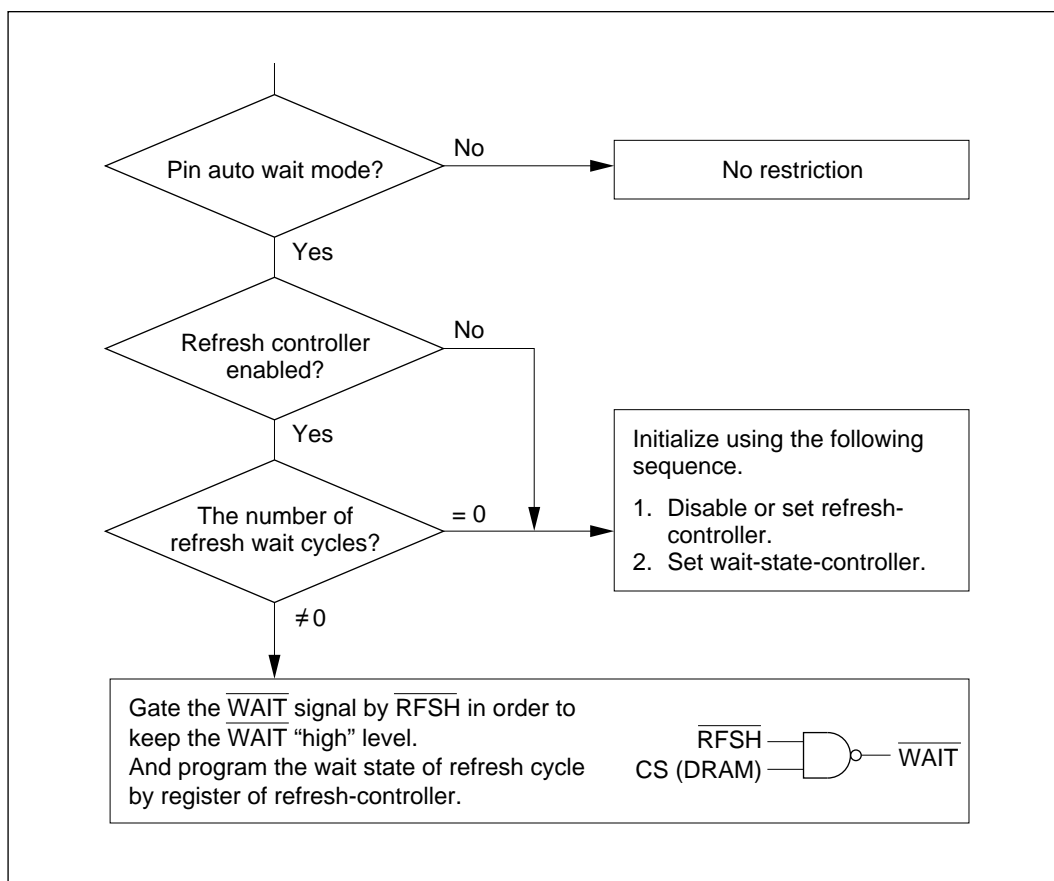
Figure 7-4 Pin Auto-Wait Mode

The H8/510 wait-state-controller supports programmable wait mode, pin wait mode, and pin auto wait mode. These functions will be effective to bus cycles when the CPU accesses external address space.

On the other hand, the refresh-controller supports a wait state insertion programmable by the refresh control register independently of the wait-state controller. The refresh-controller, however, supports only programmable wait mode and pin wait mode, not pin auto wait mode.

Therefore, if pin auto wait mode is selected by the wait-state-controller, and if CS of DRAM is connected to WAIT, the wait state will not be released in refresh cycles.

Please refer and follow the flowchart below.



Section 8 Clock Pulse Generator

8.1 Overview

The H8/510 chip has a built-in clock pulse generator (CPG) consisting of an oscillator circuit, a system (ϕ) clock divider, an E clock divider, and a group of prescalers. The prescalers generate clock signals for the on-chip supporting modules.

8.1.1 Block Diagram

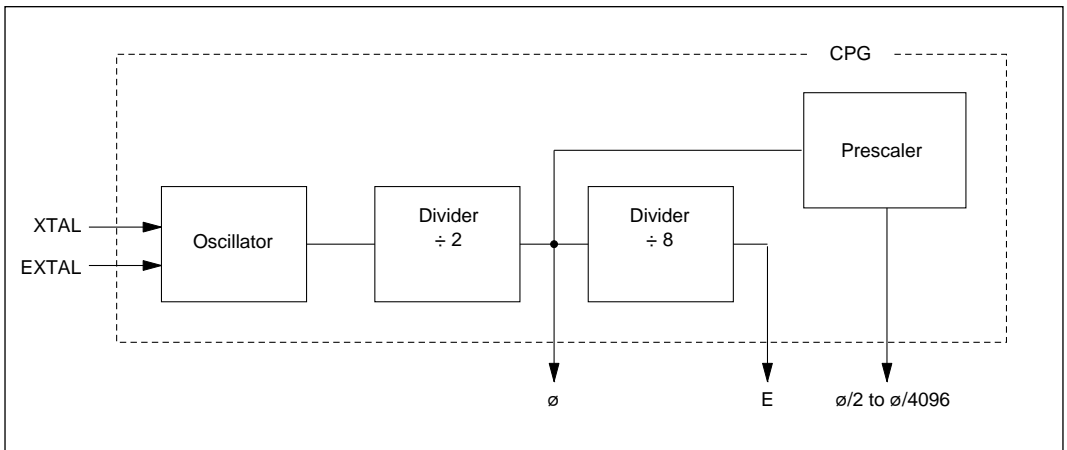


Figure 8-1 Block Diagram of Clock Pulse Generator

8.2 Oscillator Circuit

If an external crystal is connected across the EXTAL and XTAL pins, the on-chip oscillator circuit generates a clock signal for the system clock divider. Alternatively, an external clock signal can be applied directly.

1. Connecting an External Crystal

Circuit Configuration: An external crystal can be connected as in the example in figure 8-2. An AT-cut parallel resonating crystal should be used.

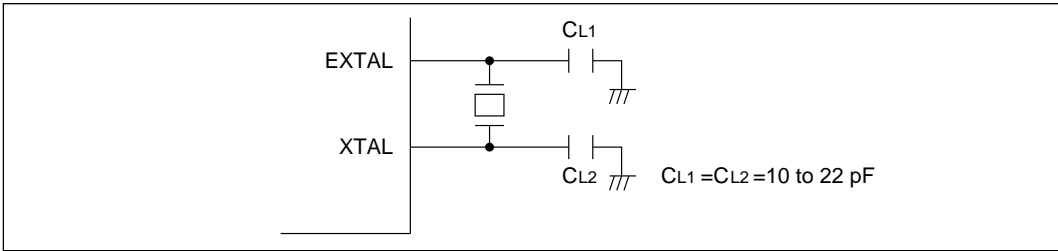


Figure 8-2 Connection of Crystal Oscillator (Example)

Crystal Oscillator: The external crystal should have the characteristics listed in table 8-1.

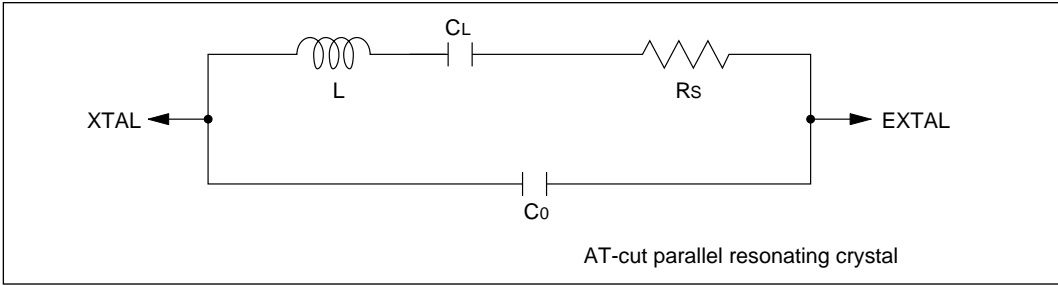


Figure 8-3 Crystal Oscillator Equivalent Circuit

Table 8-1 External Crystal Parameters

Frequency (MHz)	2	4	8	12	16	20
Rs max (Ω)	500	120	60	40	30	20
C0 (pF)	7 pF max					

Note on Board Design: When an external crystal is connected, other signal lines should be kept away from the crystal circuit to prevent induction from interfering with correct oscillation. See figure 8-4.

When the board is designed, the crystal and its load capacitors should be placed as close as possible to the XTAL and EXTERNAL pins.

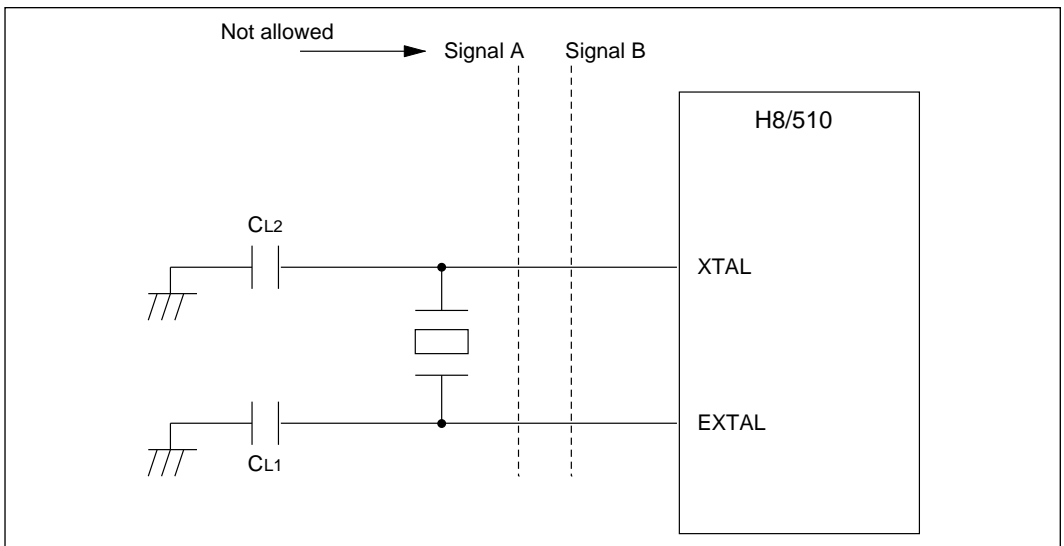


Figure 8-4 Notes on Board Design around External Crystal

2. Input of External Clock Signal

Circuit Configuration: An external clock signal can be input at the EXTAL and XTAL pins as shown in the example in figure 8-5.

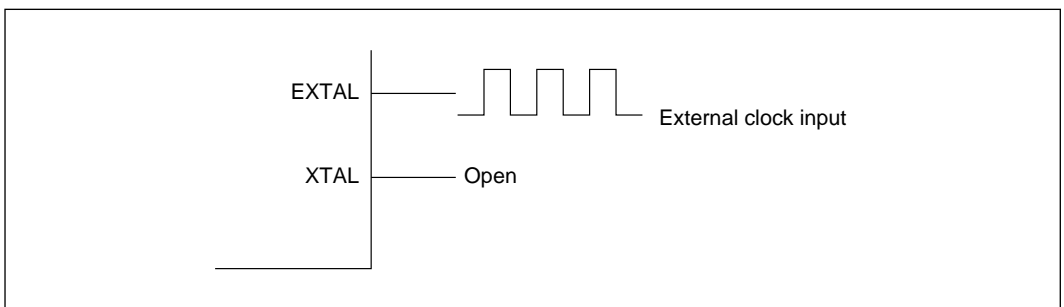


Figure 8-5 External Clock Input (Example)

External Clock Input

Frequency	Double the system clock (ϕ) frequency
Duty factor	45% to 55%

Note on Connection: Leave the XTAL pin open.

8.3 System Clock Divider

The system clock divider divides the crystal oscillator or external clock frequency (f_{osc}) by 2 to create the ϕ clock.

An E clock signal is created by dividing the ϕ clock by 8. The E clock is used for interfacing to E clock based devices.

Figure 8-6 shows the phase relationship of the E clock to the ϕ clock.

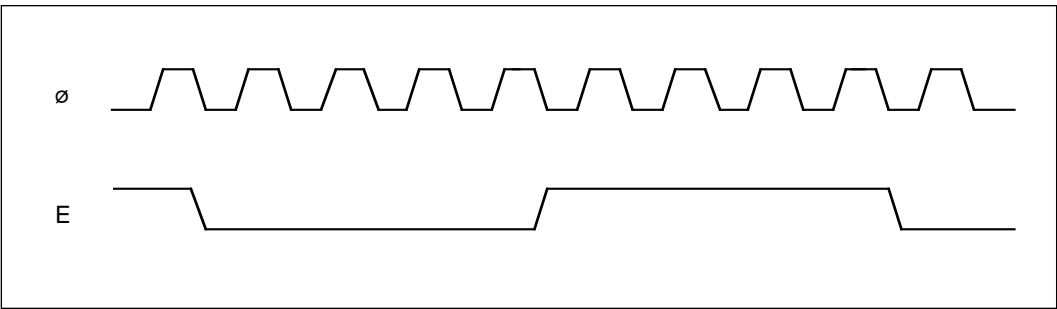


Figure 8-6 Phase Relationship of ϕ Clock and E Clock

Section 9 I/O Ports

9.1 Overview

The H8/510 has eight ports. Ports 1, 2, 3, 4, 5, 6, and 8 are eight-bit input/output ports. Port 7 is a four-bit input port. Table 9-1 summarizes the functions of each port.

Input and output are memory-mapped. The CPU views each port as a data register (DR) located in the on-chip register field at the high end of page 0 of the address space. Each port (except port 7) also has a data direction register (DDR) which determines which pins are used for input and which for output.

In addition to the data and data direction registers, the bus release control register (BRCR) affects the operation of port 3.

To read data from an I/O port, the CPU selects input in the data direction register and reads the data register. This causes the input logic level at the pin to be placed directly on the internal data bus. There is no intervening input latch.

To send data to an output port, the CPU selects output in the data direction register and writes the desired data in the data register, causing the data to be held in a latch. The latch output drives the pin through a buffer amplifier. If the CPU reads the data register of an output port, it obtains the data held in the latch rather than the actual level of the pin.

Outputs from ports 1 to 2 can drive one TTL load and a 90-pF capacitive load. Outputs from ports 3 to 6 and port 8 can drive one TTL load and a 30-pF capacitive load.

Port 4 has Schmitt-triggered inputs.

Outputs from ports 1 to 6 and port 8 can also drive a Darlington transistor pair.

Schematic diagrams of the I/O port circuits are shown in appendix C.

Table 9-1 Input/Output Port Summary

Port	Description	Pins	Mode 1	Mode 2	Mode 3	Mode 4
Port 1	8-Bit I/O port	P17 – P10 D7 – D0	I/O port	Data bus (D7 – D0)	I/O port	Data bus (D7 – D0)
Port 2	8-Bit I/O port	P27 – P20 A23 – A16	I/O port		Address bus	(A23 – A16)
Port 3	8-Bit I/O port	P37 P36 P35 P34 P33 P32 / $\overline{\text{BREQ}}$ P31 / $\overline{\text{BACK}}$ P30 / $\overline{\text{WAIT}}$	8-Bit I/O port, also used for $\overline{\text{BACK}}$, $\overline{\text{BREQ}}$, and $\overline{\text{WAIT}}$ • $\overline{\text{BACK}}$, $\overline{\text{BREQ}}$, $\overline{\text{WAIT}}$: These pin functions are used when the corresponding control register bit is set to 1. If the control bit is cleared to 0, the pin is used as an input/output port. • The three pins used for these signals are P30 to P32.			
Port 4	8-Bit I/O port	P47 / FTCl2 P46 / FTl2 P45 / FTCl1 P44 / FTl1 P43 / TMO P42 / TMRI P41 / TMCI P40 / $\overline{\text{ADTRG}}$	Input and output pins (FTl1, FTl2, FTCl1, FTCl2) for the 16-bit free-running timers (FRT1 and FRT2), input (TMCI, TMRI) and output (TMO) pins for the 8-bit timer, input pin for ADTRG, and I/O port. • $\overline{\text{ADTRG}}$: This pin function is used when the corresponding control register bit is set to 1. If the control bit is cleared to 0, the pin is used as an input/output port.			
Port 5	8-Bit I/O port	P57 P56 P55 P54 P53 / FTOB2 P52 / FTOA2 P51 / FTOB1 P50 / FTOA1	8-Bit I/O port, also providing output pins (FTOB2, FTOA2, FTOB1, FTOA1) for FRT1 and FRT2. • The four pins with dual functions are P50 to P53.			
Port 6	8-Bit I/O port	P67 - P60	I/O port			
Port 7	4-Bit input port	P73 – P70 / AN3 – AN0	Input port, also providing analog input pins (AN3 to AN0) for the A/D converter			
Port 8	8-Bit I/O port	P87 / TXD2 P86 / RXD2 P85 / TXD1 P84 / RXD1 P83 / $\overline{\text{IRQ3}}$ /SCK2 P82 / $\overline{\text{IRQ2}}$ /SCK1 P81 / $\overline{\text{IRQ1}}$ P80 / $\overline{\text{IRQ0}}$	I/O port, also providing input and output pins (RXD1, TXD1, RXD2, TXD2, SCK1, SCK2) for the serial communication interfaces (SCI1 and SCI2) and interrupt request input pins ($\overline{\text{IRQ0}}$ to $\overline{\text{IRQ3}}$).			

9.2 Port 1

9.2.1 Overview

Port 1 is an 8-bit input/output port with the pin configuration shown in figure 9-1. The pins are used for the data bus in modes 2 and 4, and as general-purpose input or output pins in modes 1 and 3.

Outputs from port 1 can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington pair.

	Pin	Modes 2 and 4	Modes 1 and 3
Port 1	↔ P17 / D7	D7 (input/output)	P17 (input/output)
	↔ P16 / D6	D6 (input/output)	P16 (input/output)
	↔ P15 / D5	D5 (input/output)	P15 (input/output)
	↔ P14 / D4	D4 (input/output)	P14 (input/output)
	↔ P13 / D3	D3 (input/output)	P13 (input/output)
	↔ P12 / D2	D2 (input/output)	P12 (input/output)
	↔ P11 / D1	D1 (input/output)	P11 (input/output)
	↔ P10 / D0	D0 (input/output)	P10 (input/output)

Figure 9-1 Pin Functions of Port 1

9.2.2 Port 1 Registers

Table 9-2 lists the registers of port 1.

Table 9-2 Port 1 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 1 data direction register	P1DDR	W	H'00	H'FE80
Port 1 data register	P1DR	R/W	H'00	H'FE82

1. Port 1 Data Direction Register (P1DDR)—H'FE80

Bit	7	6	5	4	3	2	1	0
	P17DDR	P16DDR	P15DDR	P14DDR	P13DDR	P12DDR	P11DDR	P10DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P1DDR is an 8-bit register that selects the direction of each pin in port 1.

Modes 1 and 3: A pin functions as an output pin if the corresponding bit in P1DDR is set to 1, and as an input pin if the bit is cleared to 0.

P1DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P1DDR is initialized to H'00 by a reset and in the hardware standby mode. P1DDR is not initialized in the software standby mode, so if a P1DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 1 data register.

2. Port 1 Data Register (P1DR)—H'FE82

Bit	7	6	5	4	3	2	1	0
	P17	P16	P15	P14	P13	P12	P11	P10
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P1DR is an 8-bit register containing output data for pins P17 to P10.

P1DR is initialized to H'00 by a reset and in the hardware standby mode.

When the CPU reads P1DR, for output pins it reads the value in the P1DR latch. For input pins, it reads the logic level directly from the pin.

9.2.3 Pin Functions in Each Mode

The function of port 1 depends on whether the chip is operating in mode 1 or 3, or in mode 2 or 4. This information is summarized below.

Pin Functions in Modes 2 and 4: Port 1 is automatically used for the data bus. The direction bits in P1DDR are ignored. Figure 9-2 shows the pin functions in modes 2 and 4.

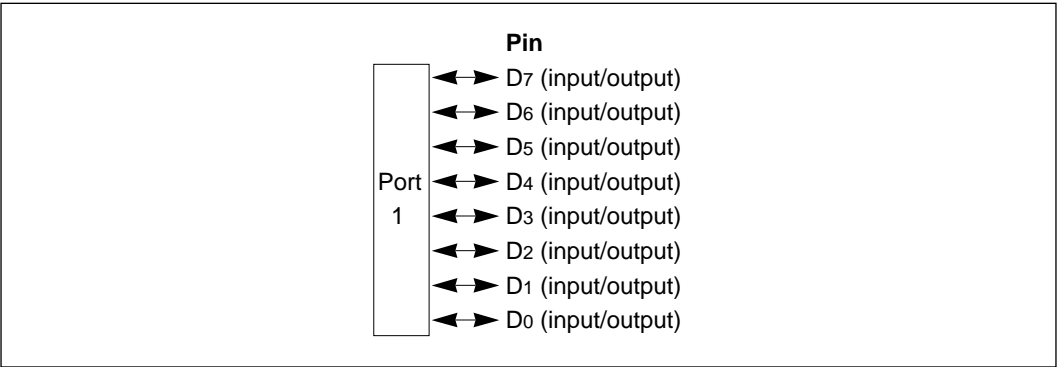


Figure 9-2 Pin Functions of Port 1 in Modes 2 and 4

Pin Functions in Modes 1 and 3: Port 1 is a general-purpose input/output port in which each pin can be set individually for input or output. See figure 9-3.

A pin becomes an output pin when the corresponding P1DDR bit is set to 1, and an input pin when this bit is cleared to 0.

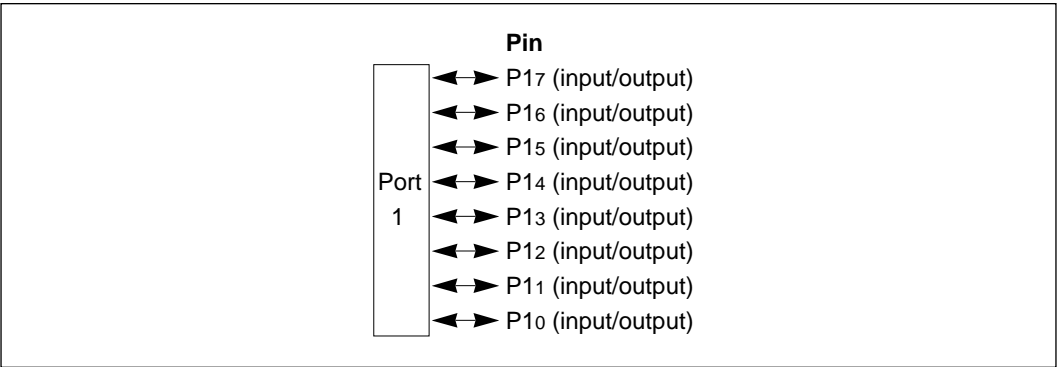


Figure 9-3 Pin Functions of Port 1 in Modes 1 and 3

9.3 Port 2

9.3.1 Overview

Port 2 is a 8-bit input/output port with the pin configuration shown in figure 9-4. The pins are used for page address output (A23 to A16) in the maximum modes, and as general-purpose input or output pins in the minimum modes.

Outputs from port 1 can drive one TTL load and a 90-pF capacitive load. They can also drive a Darlington pair.

	Pin	Maximum Modes	Minimum Modes
Port 2	↔ P27 / A23	A23 (output)	P27 (input/output)
	↔ P26 / A22	A22 (output)	P26 (input/output)
	↔ P25 / A21	A21 (output)	P25 (input/output)
	↔ P24 / A20	A20 (output)	P24 (input/output)
	↔ P23 / A19	A19 (output)	P23 (input/output)
	↔ P22 / A18	A18 (output)	P22 (input/output)
	↔ P21 / A17	A17 (output)	P21 (input/output)
	↔ P20 / A16	A16 (output)	P20 (input/output)

Figure 9-4 Pin Functions of Port 2

9.3.2 Port 2 Registers

Table 9-3 lists the registers of port 2.

Table 9-3 Port 2 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 2 data direction register	P2DDR	W	H'00	H'FE81
Port 2 data register	P2DR	R/W	H'00	H'FE83

1. Port 2 Data Direction Register (P2DDR)—H'FE81

Bit	7	6	5	4	3	2	1	0
	P27DDR	P26DDR	P25DDR	P24DDR	P23DDR	P22DDR	P21DDR	P20DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P2DDR is an 8-bit register that selects the direction of each pin in port 2.

Minimum Modes: A pin functions as an output pin if the corresponding bit in P2DDR is set to 1, and as an input pin if the bit is cleared to 0.

P2DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P2DDR is initialized to H'00 by a reset and in the hardware standby mode. P2DDR is not initialized in the software standby mode, so if a P2DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 2 data register.

Modes 2 and 4: All bits in P2DDR are automatically set to 1 and cannot be modified.

2. Port 2 Data Register (P2DR)—H'FE83

Bit	7	6	5	4	3	2	1	0
	P27	P26	P25	P24	P23	P22	P21	P20
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P2DR is an 8-bit register containing output data for pins P27 to P20.

P2DR is initialized to H'00 by a reset and in the hardware standby mode.

When the CPU reads P2DR, for output pins it reads the value in the P2DR latch. For input pins, it reads the logic level directly from the pin.

9.3.3 Pin Functions in Each Mode

The function of port 2 depends on whether the chip is operating in maximum mode (mode 3 or 4) or minimum mode (mode 1 or 2). This information is summarized below.

Pin Functions in Maximum Modes: P2DDR is automatically set for output and port 2 is used for output of the page address (A23 to A16). Figure 9-5 shows the pin functions in the maximum modes.

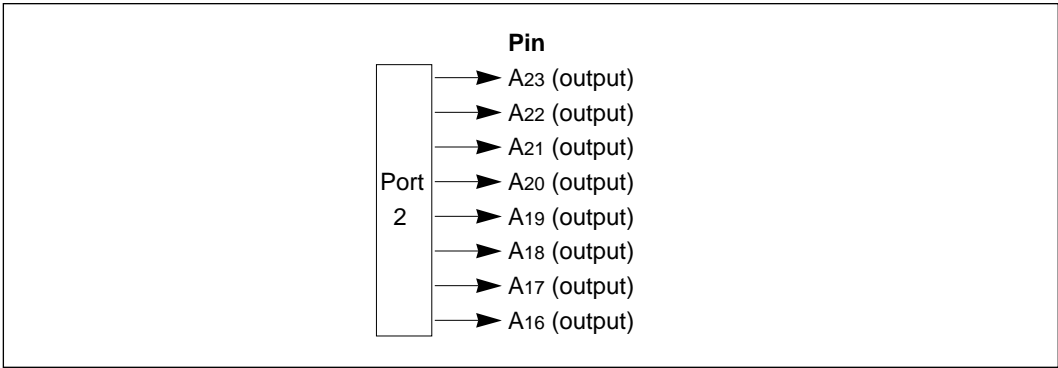


Figure 9-5 Pin Functions of Port 2 in Maximum Modes

Pin Functions in Minimum Modes: Port 2 is a general-purpose input/output port in which each pin can be set individually for input or output. See figure 9-6.

A pin becomes an output pin when the corresponding P2DDR bit is set to 1, and an input pin when this bit is cleared to 0.

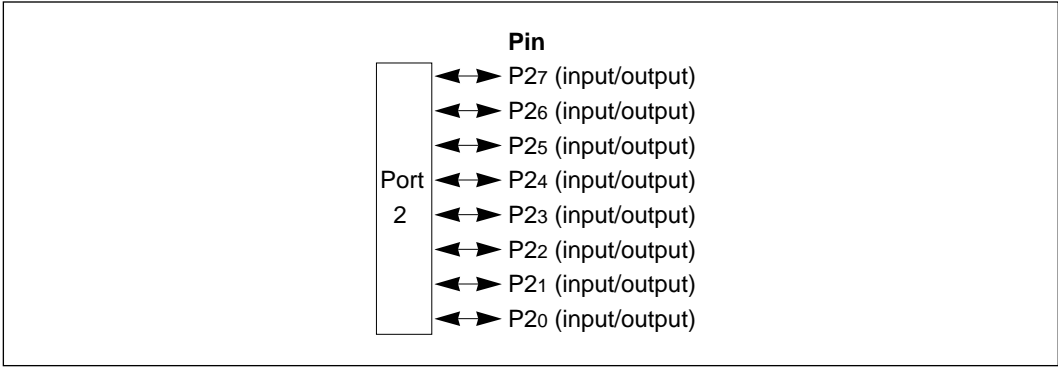


Figure 9-6 Pin Functions of Port 2 in Minimum Modes

9.4 Port 3

9.4.1 Overview

Port 3 is an 8-bit input/output port with the pin configuration shown in figure 9-7. The pin functions are the same in all MCU modes. Three of the pins are used for input and output of the $\overline{\text{BACK}}$, $\overline{\text{BREQ}}$, and $\overline{\text{WAIT}}$ signals.

Outputs from port 3 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington pair.

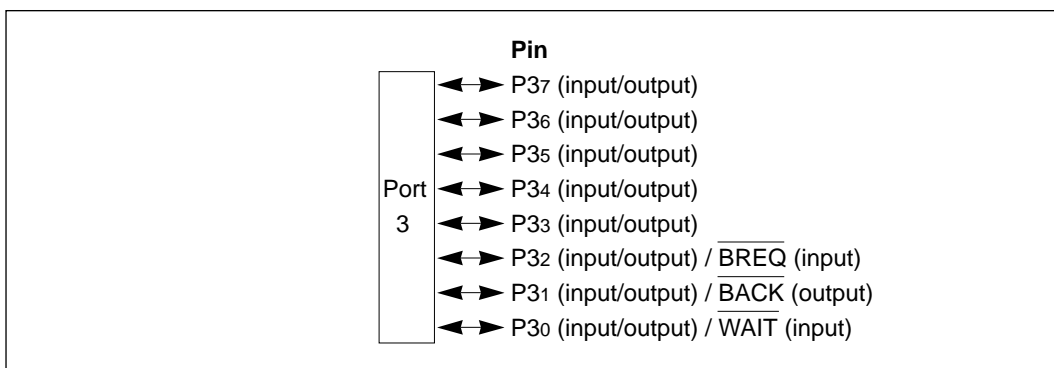


Figure 9-7 Pin Functions of Port 3

9.4.2 Port 3 Registers

Table 9-4 lists the registers of port 3.

Table 9-4 Port 3 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 3 data direction register	P3DDR	W	H'00	H'FE84
Port 3 data register	P3DR	R/W	H'00	H'FE86
Bus release control register	BRCR	R/W	H'FE	H'FF1B

1. Port 3 Data Direction Register (P3DDR)—H'FE84

Bit	7	6	5	4	3	2	1	0
	P37DDR	P36DDR	P35DDR	P34DDR	P33DDR	P32DDR	P31DDR	P30DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P3DDR is an 8-bit register that selects the direction of each pin in port 3.

A pin functions as an output pin if the corresponding bit in P3DDR is set to 1, and as an input pin if the bit is cleared to 0.

P3DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P3DDR is initialized to H'00 by a reset and in the hardware standby mode. P3DDR is not initialized in the software standby mode, so if a P3DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 3 data register.

2. Port 3 Data Register (P3DR)—H'FE86

Bit	7	6	5	4	3	2	1	0
	P37	P36	P35	P34	P33	P32	P31	P30
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P3DR is an 8-bit register containing the data for pins P37 to P30.

P3DR is initialized to H'00 by a reset and in the hardware standby mode.

When the CPU reads P3DR, for output pins it reads the value in the P3DR latch. For input pins, it reads the logic level directly from the pin.

3. Bus Release Control Register (BRCR)—H'FF1B

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	BRLE
Initial value	1	1	1	1	1	1	1	0
Read/Write	—	—	—	—	—	—	—	R/W

BRCR controls the selection of pin functions for port 3.

BRCR is initialized to H'FE by a reset and in the hardware standby mode. It is not initialized in the software standby mode.

Bits 7 to 1—Reserved: These bits cannot be written and are always read as 1.

Bits 0—Bus Release Enable (BRLE): Controls the functions of P32 and P31.

Bit 0

BRLE	Description
0	P32 and P31 are general-purpose input/output pins. (Initial value)
1	P32 is used for $\overline{\text{BREQ}}$ input, and P31 for $\overline{\text{BACK}}$ output.

9.4.3 Pin Functions

Port 3 has same pin functions in all modes. Pins P33 to P30 are also used for input of $\overline{\text{BREQ}}$ and $\overline{\text{WAIT}}$ and output of $\overline{\text{BACK}}$ as shown in figure 9-7.

Table 9-5 details the pin functions of port 3.

Table 9-5 Port 3 Pin Functions

Pin	Selection of Pin Functions			
P32 / $\overline{\text{BREQ}}$	Depends on the BRLE and P32DDR bits as follows:			
	BRLE	0		1
	P32DDR	0	1	0 1
	Pin function	P32 input	P32 output	$\overline{\text{BREQ}}$ input

P31 / $\overline{\text{BACK}}$	Depends on the BRLE and P31DDR bits as follows:			
	BRLE	0		1
	P31DDR	0	1	0 1
	Pin function	P31 input	P31 output	$\overline{\text{BACK}}$ output

P30 / $\overline{\text{WAIT}}$	Depends on the wait mode select 1 bit (WMS1) in the wait state control register (WCR) and P30DDR as follows:			
	WMS1	0		1
	P30DDR	0	1	0 1
	Pin function	P30 input	P30 output	$\overline{\text{WAIT}}$ input

9.5 Port 4

9.5.1 Overview

Port 4 is an 8-bit input/output port with the pin configuration shown in figure 9-8. It also provides input pins for the 16-bit free-running timers ($\overline{\text{FRT1}}$ and $\overline{\text{FRT2}}$), input (TMCI, TMRI) and output (TMO) pins for the 8-bit timer, and the $\overline{\text{ADTRG}}$ input pin. The pin functions are the same in all MCU modes.

Port 4 has Schmitt inputs. Outputs from port 4 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington pair.

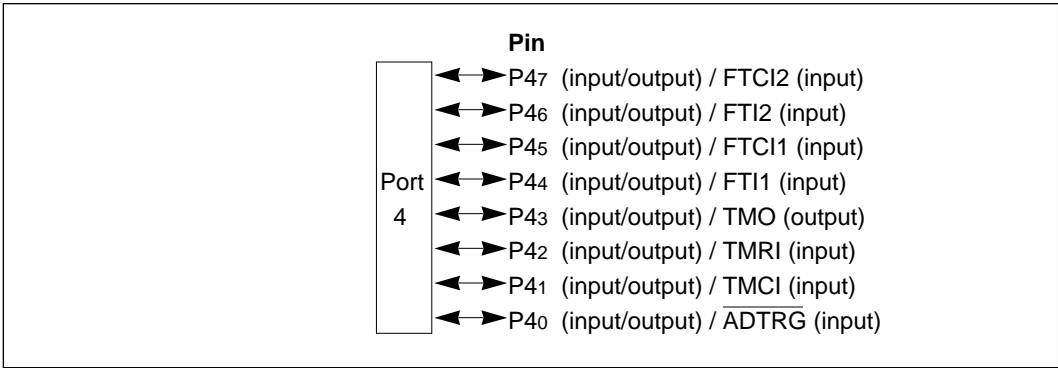


Figure 9-8 Pin Functions of Port 4

9.5.2 Port 4 Registers

Table 9-6 lists the registers of port 4.

Table 9-6 Port 4 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 4 data direction register	P4DDR	W	H'00	H'FE85
Port 4 data register	P4DR	R/W	H'00	H'FE87

1. Port 4 Data Direction Register (P4DDR)—H'FF85

Bit	7	6	5	4	3	2	1	0
	P47DDR	P46DDR	P45DDR	P44DDR	P43DDR	P42DDR	P41DDR	P40DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P4DDR is an 8-bit register that selects the direction of each pin in port 4.

A pin functions as an output pin if the corresponding bit in P4DDR is set to 1, and as an input pin if the bit is cleared to 0.

P4DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P4DDR is initialized to H'00 by a reset and in the hardware standby mode. P4DDR is not initialized in the software standby mode, so if a P4DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 4 data register.

When a pin of port 4 is used by an on-chip supporting module (as an 8-bit timer output pin, for example), if a transition to the software standby mode occurs the on-chip supporting module is initialized, so the pin becomes a general-purpose input/output pin according to P4DDR and P4DR.

2. Port 4 Data Register (P4DR)—H'FE87

Bit	7	6	5	4	3	2	1	0
	P47	P46	P45	P44	P43	P42	P41	P40
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P4DR is an 8-bit register containing the data for pins P47 to P40.

P4DR is initialized to H'00 by a reset and in the hardware standby mode.

When the CPU reads P4DR, for output pins it reads the value in the P4DR latch. For input pins, it reads the logic level directly from the pin.

9.5.3 Pin Functions

Port 4 has the same pin functions in all modes. As shown in figure 9-8, it also provides input pins for FRT1 and FRT2, input and output pins for the 8-bit timer, and the $\overline{\text{ADTRG}}$ input pin.

Table 9-7 lists the registers of port 4.

Table 9-7 Port 4 Pin Functions

Pin	Selection of Pin Functions	
P47 / FTCI2	Used for input of the FRT2 external clock when the CKSO and CKS1 bits in the FRT2 timer control register (TCR) select the external clock source.	
	P47DDR	0 1
	Pin function	P47 input P47 output
		FTCI2 input
P46 / FTI2		
	P46DDR	0 1
	Pin function	P46 input P46 output
		FTI2 input

Table 9-7 Port 4 Pin Functions (cont)**Pin Selection of Pin Functions**

P45 / FTCL1 Used for input of the FRT1 external clock when the CKS0 and CKS1 bits in the FRT1 timer control register (TCR) select the external clock source.

P45DDR	0	1
Pin function	P45 input	P45 output
	FTCL1 input	

P44 / FTI1

P44DDR	0	1
Pin function	P44 input	P44 output
	FTI1 input	

P43 / TMO Use depends on the P43DDR bit and output select bits 3 to 0 (OS3 to OS0) in the timer control/status register (TCSR) of the 8-bit timer.

OS3 to OS0	All 0		Not all 0	
P43DDR	0	1	0	1
Pin function	P43 input	P43 output	TMO output	

P42 / TMRI Used for reset input for the 8-bit timer when counter clear bits 1 and 0 (CCLR1 and CCLR0) in the timer control register (TCR) of the 8-bit timer are both set to 1.

P42DDR	0	1
Pin function	P42 input	P42 output
	TMRI input	

P41 / TMCI Used for external clock input for the 8-bit timer when clock select bits 2 to 0 (CKS2 to CKS0) in the timer control register (TCR) of the 8-bit timer select the external clock source.

P41DDR	0	1
Pin function	P41 input	P41 output
	TMCI input	

Table 9-7 Port 4 Pin Functions (cont)

Pin	Selection of Pin Functions			
P40 / $\overline{\text{ADTRG}}$	Depends on the P40DDR bit and the trigger enable bit (TRGE) in the A/D control register (ADCR) as follows:			
	0		1	
P40DDR	0	1	0	1
Pin function	P40 input	P40 output	$\overline{\text{ADTRG}}$ input	

9.6 Port 5

9.6.1 Overview

Port 5 is an 8-bit input/output port with the pin configuration shown in figure 9-9. The pin functions are the same in all MCU modes. Four of the pins are used for output of signals from the 16-bit free-running timers (FRT1 and FRT2).

Outputs from port 5 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington pair.

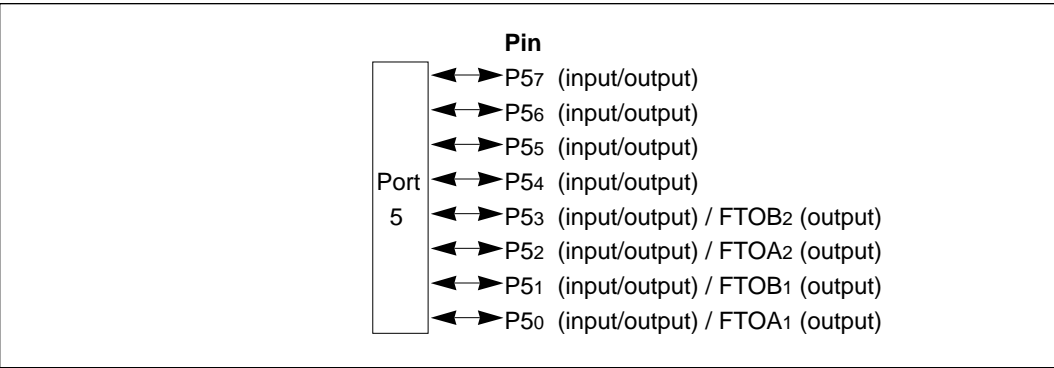


Figure 9-9 Pin Functions of Port 5

9.6.2 Port 5 Registers

Table 9-8 lists the registers of port 5.

Table 9-8 Port 5 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 5 data direction register	P5DDR	W	H'00	H'FE88
Port 5 data register	P5DR	R/W	H'00	H'FE8A

1. Port 5 Data Direction Register (P5DDR)—H'FE88

Bit	7	6	5	4	3	2	1	0
	P57DDR	P56DDR	P55DDR	P54DDR	P53DDR	P52DDR	P51DDR	P50DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P5DDR is an 8-bit register that selects the direction of each pin in port 5.

A pin functions as an output pin if the corresponding bit in P5DDR is set to 1, and as an input pin if the bit is cleared to 0.

P5DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P5DDR is initialized to H'00 by a reset and in the hardware standby mode. P5DDR is not initialized in the software standby mode, so if a P5DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 5 data register.

When a pin of port 5 is used by an on-chip supporting module (as an FRT output pin), if a transition to the software standby mode occurs the on-chip supporting module is initialized, so the pin becomes a general-purpose input/output pin according to P5DDR and P5DR.

2. Port 5 Data Register (P5DR)—H'FE8A

Bit	7	6	5	4	3	2	1	0
	P57	P56	P55	P54	P53	P52	P51	P50
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P5DR is an 8-bit register containing output data for pins P57 to P50.

P5DR is initialized to H'00 by a reset and in the hardware standby mode.

When the CPU reads P5DR, for output pins it reads the value in the P5DR latch. For input pins, it reads the logic level directly from the pin.

9.6.3 Pin Functions

Port 5 has the same pin functions in all modes. Some pins are also used for FRT output as shown in figure 9-9.

Table 9-9 details the pin functions of port 5.

Table 9-9 Port 5 Pin Functions

Pin	Selection of Pin Functions			
P53 / FTOB2	Usage depends on the P53DDR bit and the output enable B bit (OEB) in the FRT2 timer control register (TCR) as follows:			
	0		1	
P53DDR	0	1	0	1
Pin function	P53 input	P53 output	FTOB2 output	

P52 / FTOA2	Usage depends on the P52DDR bit and the output enable A bit (OEA) in the FRT2 timer control register (TCR) as follows:			
	0		1	
P52DDR	0	1	0	1
Pin function	P52 input	P52 output	FTOA2 output	

Table 9-9 Port 5 Pin Functions (cont)

Pin

Selection of Pin Functions

P51 / FTOB1

Usage depends on the P51DDR bit and the output enable B bit (OEB) in the FRT1 timer control register (TCR) as follows:

OEB	0		1	
P51DDR	0	1	0	1
Pin function	P51 input	P51 output	FTOB1 output	

P50 / FTOA1

Usage depends on the P50DDR bit and the output enable A bit (OEA) in the FRT1 timer control register (TCR) as follows:

OEA	0		1	
P50DDR	0	1	0	1
Pin function	P50 input	P50 output	FTOA1 output	

9.7 Port 6

9.7.1 Overview

Port 6 is an 8-bit input/output port with the pin configuration shown in figure 9-10.

Outputs from port 6 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington pair.

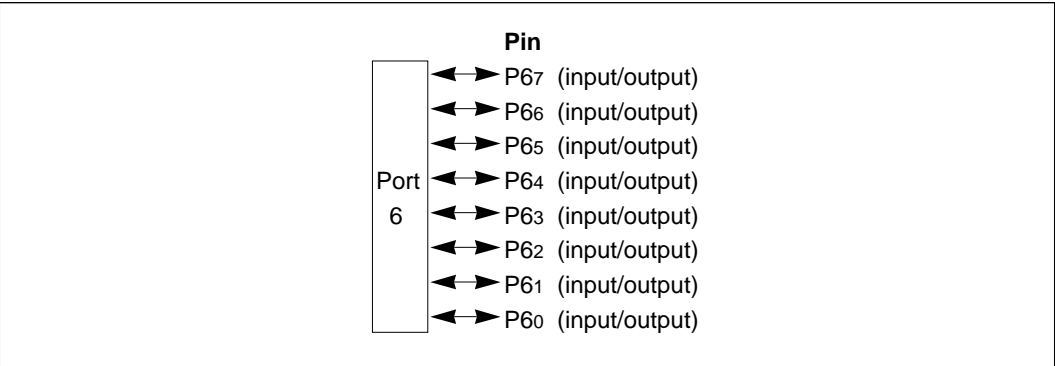


Figure 9-10 Pin Functions of Port 6

9.7.2 Port 6 Registers

Table 9-10 lists the registers of port 6.

Table 9-10 Port 6 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 6 data direction register	P6DDR	W	H'00	H'FE89
Port 6 data register	P6DR	R/W	H'00	H'FE8B

1. Port 6 Data Direction Register (P6DDR)—H'FE89

Bit	7	6	5	4	3	2	1	0
	P67DDR	P66DDR	P65DDR	P64DDR	P63DDR	P62DDR	P61DDR	P60DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P6DDR is an 8-bit register that selects the direction of each pin in port 6.

A pin functions as an output pin if the corresponding bit in P6DDR is set to 1, and as an input pin if the bit is cleared to 0.

P6DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P6DDR is initialized to H'00 by a reset and in the hardware standby mode. P6DDR is not initialized in the software standby mode, so if a P6DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 6 data register.

2. Port 6 Data Register (P6DR)—H'FE8B

Bit	7	6	5	4	3	2	1	0
	P67	P66	P65	P64	P63	P62	P61	P60
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P6DR is an 8-bit register containing output data for pins P67 to P60.

P6DR is initialized to H'00 by a reset and in the hardware standby mode.

When the CPU reads P6DR, for output pins it reads the value in the P6DR latch. For input pins, it reads the logic level directly from the pin.

9.8 Port 7

9.8.1 Overview

Port 7 is a 4-bit input port that also receives inputs for the on-chip A/D converter. The pin functions are the same in all MCU operating modes, as shown in figure 9-11.

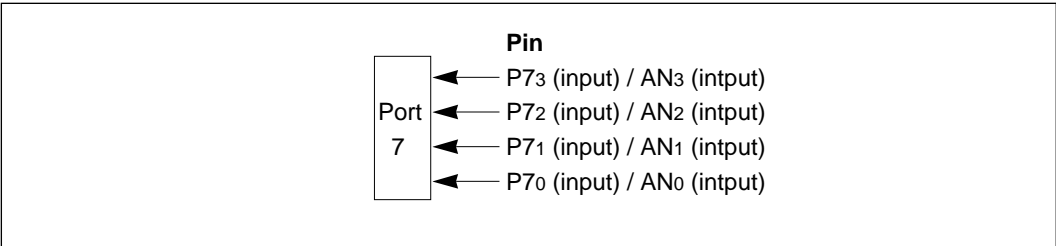


Figure 9-11 Pin Functions of Port 7

9.8.2 Port 7 Registers

Port 7 has only the data register described in table 9-11. Since it is exclusively an input port, there is no data direction register.

Table 9-11 Port 7 Registers

Name	Abbreviation	Read/Write	Address
Port 7 data register	P7DR	R	H'FE8E

Port 7 Data Register (P7DR)—H'FE8E

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	P73	P72	P71	P70
Read/Write	—	—	—	—	R	R	R	R

When the CPU reads P7DR it always reads the current logic level of each pin.

9.9 Port 8

9.9.1 Overview

Port 8 is an 8-bit input/output port with the pin configuration shown in figure 9-12. It also provides input pins for $\overline{\text{IRQ}}_0$ to $\overline{\text{IRQ}}_3$ and input and output pins for the serial communication interfaces (SCI1 and SCI2). The pin functions are the same in all MCU modes.

Outputs from port 8 can drive one TTL load and a 30-pF capacitive load. They can also drive a Darlington pair.

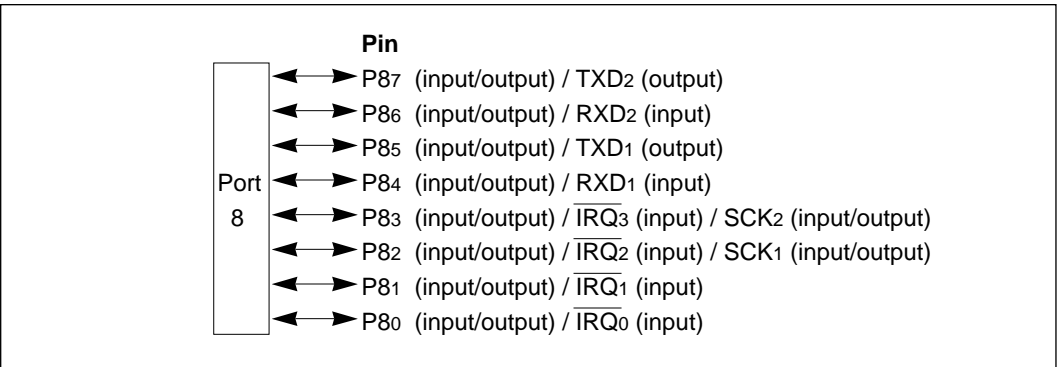


Figure 9-12 Pin Functions of Port 8

9.9.2 Port 8 Registers

Table 9-12 lists the registers of port 8.

Table 9-12 Port 8 Registers

Name	Abbreviation	Read/Write	Initial Value	Address
Port 8 data direction register	P8DDR	W	H'00	H'FE8D
Port 8 data register	P8DR	R/W	H'00	H'FE8F

1. Port 8 Data Direction Register (P8DDR)—H'FE8D

Bit	7	6	5	4	3	2	1	0
	P87DDR	P86DDR	P85DDR	P84DDR	P83DDR	P82DDR	P81DDR	P80DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

P8DDR is an 8-bit register that selects the direction of each pin in port 8.

A pin functions as an output pin if the corresponding bit in P8DDR is set to 1, and as an input pin if the bit is cleared to 0.

P8DDR can be written but not read. An attempt to read this register does not cause an error, but all bits are read as 1, regardless of their true values.

P8DDR is initialized to H'00 by a reset and in the hardware standby mode. P8DDR is not initialized in the software standby mode, so if a P8DDR bit is set to 1 when the chip enters the software standby mode, the corresponding pin continues to output the value in the port 8 data register.

When a pin of port 8 is used by an on-chip supporting module (as an SCI output pin, for example), if a transition to the software standby mode occurs the on-chip supporting module is initialized, so the pin becomes a general-purpose input/output pin according to P8DDR and P8DR.

2. Port 8 Data Register (P8DR)—H'FE8F

Bit	7	6	5	4	3	2	1	0
	P87	P86	P85	P84	P83	P82	P81	P80
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P8DR is an 8-bit register containing output data for pins P87 to P80.

P8DR is initialized to H'00 by a reset and in the hardware standby mode.

When the CPU reads P8DR, for output pins it reads the value in the P8DR latch. For input pins, it reads the logic level directly from the pin.

9.9.3 Pin Functions

Port 8 has the same pin functions in all modes. As shown in figure 9-12, it also provides input pins for $\overline{\text{IRQ}}_0$ to $\overline{\text{IRQ}}_3$ and input and output pins for the serial communication interface.

Table 9-13 shows the pin functions of port 8.

Table 9-13 Port 8 Pin Functions

Pin

Selection of Pin Functions

P87 / TXD2

Usage depends on the P87DDR bit and the transmit enable (TE) bit in the SCI2 serial control register (SCR) as follows:

TE	0		1	
P87DDR	0	1	0	1
Pin function	P87 input	P87 output	TXD2 output	

P86 / RXD2

Usage depends on the P86DDR bit and the receive enable (RE) bit in the SCI2 serial control register (SCR) as follows:

RE	0		1	
P86DDR	0	1	0	1
Pin function	P86 input	P86 output	RXD2 input	

P85 / TXD1

Usage depends on the P85DDR bit and the transmit enable (TE) bit in the SCI1 serial control register (SCR) as follows:

TE	0		1	
P85DDR	0	1	0	1
Pin function	P85 input	P85 output	TXD1 output	

P84 / TXD1

Usage depends on the P84DDR bit and the receive enable (RE) bit in the SCI1 serial control register (SCR) as follows:

RE	0		1	
P84DDR	0	1	0	1
Pin function	P84 input	P84 output	RXD1 input	

Table 9-13 Port 8 Pin Functions (cont)

Pin Selection of Pin Functions

P83 / SCK2 / $\overline{\text{IRQ}}_3$ Usage depends on the communication mode bit ($\text{C}/\overline{\text{A}}$) and clock enable bits 1 and 0 (CKE1 and CKE0) in the SCI2 serial control register (SCR) as follows:

$\text{C}/\overline{\text{A}}$	0				1			
CKE1	0		1		0		1	
CKE0	0	1	0	1	0	1	0	1
Pin function	See below	SCI2 output	SCI2 input		SCI2 output		SCI2 input	

When $\text{C}/\overline{\text{A}}$, CKE1, and CKE0 are all cleared to 0, usage depends on the $\overline{\text{IRQ}}_3\text{E}$ and P83DDR bits as follows:

$\overline{\text{IRQ}}_3\text{E}$	0		1	
P83DDR	0	1	0	1
Pin function	P83 input	P83 output	$\overline{\text{IRQ}}_3$ input	

P82 / SCK1 / $\overline{\text{IRQ}}_2$ Usage depends on the communication mode bit ($\text{C}/\overline{\text{A}}$) and clock enable bits 1 and 0 (CKE1 and CKE0) in the SCI1 serial control register (SCR) as follows:

$\text{C}/\overline{\text{A}}$	0				1			
CKE1	0		1		0		1	
CKE0	0	1	0	1	0	1	0	1
Pin function	See below	SCI1 output	SCI1 input		SCI1 output		SCI1 input	

When $\text{C}/\overline{\text{A}}$, CKE1, and CKE0 are all cleared to 0, usage depends on the $\overline{\text{IRQ}}_2\text{E}$ and P82DDR bits as follows:

$\overline{\text{IRQ}}_2\text{E}$	0		1	
P82DDR	0	1	0	1
Pin function	P82 input	P82 output	$\overline{\text{IRQ}}_2$ input	

Table 9-13 Port 8 Pin Functions (cont)

Pin Selection of Pin Functions

P8₁ / $\overline{\text{IRQ}}_1$ Usage depends on the IRQ₁E and P8₁DDR bits as follows:

IRQ ₁ E	0		1	
P8 ₁ DDR	0	1	0	1
Pin function	P8 ₁ input	P8 ₁ output	$\overline{\text{IRQ}}_1$ input	

P8₀ / $\overline{\text{IRQ}}_0$ Usage depends on the IRQ₀E and P8₀DDR bits as follows:

IRQ ₀ E	0		1	
P8 ₀ DDR	0	1	0	1
Pin function	P8 ₀ input	P8 ₀ output	$\overline{\text{IRQ}}_0$ input	

Section 10 16-Bit Free-Running Timers

10.1 Overview

The H8/510 has an on-chip 16-bit free-running timer (FRT) module with two independent channels (FRT1 and FRT2). Both channels are functionally identical.

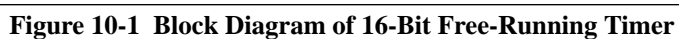
Each channel has a 16-bit free-running counter that it uses as a time base. Applications of the FRT module include rectangular-wave output (up to two independent waveforms per channel), input pulse width measurement, and measurement of external clock periods.

10.1.1 Features

The features of the free-running timer module are listed below.

- Selection of four clock sources
The free-running counters can be driven by an internal clock source ($\phi/4$, $\phi/8$, or $\phi/32$), or an external clock input (enabling use as an external event counter).
- Two independent comparators
Each free-running timer channel can generate two independent waveforms.
- Input capture function
The current count can be captured on the rising or falling edge (selectable) of an input signal.
- Four types of interrupts
Compare-match A and B, input capture, and overflow interrupts can be requested independently.
The compare-match and input capture interrupts can be served by the data transfer controller (DTC), enabling interrupt-driven data transfer with minimal CPU programming.
- Counter can be cleared under program control
The free-running counters can be cleared on compare-match A.

Figure 10-1 shows a block diagram of one free-running timer channel.



10.1.3 Input and Output Pins

Table 10-1 lists the input and output pins of the free-running timer module.

Table 10-1 Input and Output Pins of Free-Running Timer Module

Channel	Name	Abbreviation	I/O	Function
1	Output compare A	FTOA1	Output	Output controlled by comparator A of FRT1
	Output compare B or	FTOB1 /	Output /	Output controlled by comparator B of FRT1
	counter clock input	FTCl1	Input	External clock source for FRT1
	Input capture	FTI1	Input	Trigger for capturing current count of FRT1
2	Output compare A	FTOA2	Output	Output controlled by comparator A of FRT2
	Output compare B or	FTOB2 /	Output /	Output controlled by comparator B of FRT2
	counter clock input	FTCl2	Input	External clock source for FRT2
	Input capture	FTI2	Input	Trigger for capturing current count of FRT2

10.1.4 Register Configuration

Table 10-2 lists the registers of each free-running timer channel.

Table 10-2 Register Configuration

Channel	Name	Abbreviation	R/W	Initial Value	Address
1	Timer control register	TCR	R/W	H'00	H'FEA0
	Timer control/status register	TCSR	R/(W)*	H'00	H'FEA1
	Free-running counter (High)	FRC (H)	R/W	H'00	H'FEA2
	Free-running counter (Low)	FRC (L)	R/W	H'00	H'FEA3
	Output compare register A (High)	OCRA (H)	R/W	H'FF	H'FEA4
	Output compare register A (Low)	OCRA (L)	R/W	H'FF	H'FEA5
	Output compare register B (High)	OCRB (H)	R/W	H'FF	H'FEA6
	Output compare register B (Low)	OCRB (L)	R/W	H'FF	H'FEA7
	Input capture register (High)	ICR (H)	R	H'00	H'FEA8
	Input capture register (Low)	ICR (L)	R	H'00	H'FEA9
2	Timer control register	TCR	R/W	H'00	H'FEB0
	Timer control/status register	TCSR	R/(W)*	H'00	H'FEB1
	Free-running counter (High)	FRC (H)	R/W	H'00	H'FEB2
	Free-running counter (Low)	FRC (L)	R/W	H'00	H'FEB3
	Output compare register A (High)	OCRA (H)	R/W	H'FF	H'FEB4
	Output compare register A (Low)	OCRA (L)	R/W	H'FF	H'FEB5
	Output compare register B (High)	OCRB (H)	R/W	H'FF	H'FEB6
	Output compare register B (Low)	OCRB (L)	R/W	H'FF	H'FEB7
	Input capture register (High)	ICR (H)	R	H'00	H'FEB8
	Input capture register (Low)	ICR (L)	R	H'00	H'FEB9

* Software can write a 0 to clear bits 7 to 4, but cannot write a 1 in these bits.

10.2 Register Descriptions

10.2.1 Free-Running Counter (FRC)—H'FEA2, H'FEB2

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Each FRC is a 16-bit readable/writable up-counter that increments on an internal pulse generated from a clock source. The clock source is selected by the clock select 1 and 0 bits (CKS1 and CKS0) of the timer control register (TCR).

The FRC can be cleared by compare-match A.

When the FRC overflows from H'FFFF to H'0000, the overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

Because the FRC is a 16-bit register, a temporary register (TEMP) is used when the FRC is written or read. See section 10.3, “CPU Interface,” for details.

The FRCs are initialized to H'0000 at a reset and in the standby modes.

10.2.2 Output Compare Registers A and B (OCRA and OCRB)—H'FEA4 and H'FEA6, H'FEB4 and H'FEB6

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCRA and OCRB are 16-bit readable/writable registers, the contents of which are continually compared with the value in the FRC. When a match is detected, the corresponding output compare flag (OCFA or OCFB) is set in the timer control/status register (TCSR).

In addition, if the output enable bit (OEA or OEB) in the timer control register (TCR) is set to 1, when the output compare register and FRC values match, the logic level selected by the output level bit (OLVLA or OLVLB) in the timer control status register (TCSR) is output at the output compare pin (FTOA or FTOB).

After a reset, the FTOA and FTOB outputs are 0 until the first compare-match.

Because OCRA and OCRB are 16-bit registers, a temporary register (TEMP) is used when they are written. See section 10.3, “CPU Interface” for details.

OCRA and OCRB are initialized to H'FFFF at a reset and in the standby modes.

10.2.3 Input Capture Register (ICR)—H'FEA8, H'FEB8

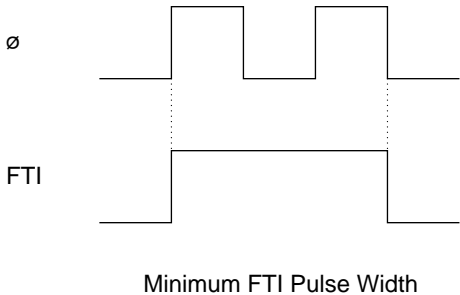
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

The ICR is a 16-bit read-only register.

When the rising or falling edge of the signal at the input capture input pin is detected, the current value of the FRC is copied to the ICR. At the same time, the input capture flag (ICF) in the timer control/status register (TCSR) is set to 1. The input capture edge is selected by the input edge select bit (IEDG) in the TCSR.

Because the ICR is a 16-bit register, a temporary register (TEMP) is used when the ICR is written or read. See section 10.3, “CPU Interface” for details.

To ensure input capture, the pulse width of the input capture signal should be at least 1.5 system clock periods ($1.5 \cdot \phi$).



The ICR is initialized to H'0000 at a reset and in the standby modes.

Note: When input capture is detected, the FRC value is transferred to the ICR even if the input capture flag (ICF) is already set.

10.2.4 Timer Control Register (TCR)—H'FEA0, H'FEB0

Bit	7	6	5	4	3	2	1	0
	ICIE	OCIEB	OCIEA	OVIE	OEB	OEA	CKS1	CKS0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The TCR is an 8-bit readable/writable register that selects the FRC clock source, enables the output compare signals, and enables interrupts.

The TCR is initialized to H'00 at a reset and in the standby modes.

Bit 7—Input Capture Interrupt Enable (ICIE): This bit selects whether to request an input capture interrupt (ICI) when the input capture flag (ICF) in the timer control/status register (TCSR) is set to 1.

Bit 7

ICIE	Description
0	The input capture interrupt request (ICI) is disabled. (Initial value)
1	The input capture interrupt request (ICI) is enabled.

Bit 6—Output Compare Interrupt Enable B (OCIEB): This bit selects whether to request output compare interrupt B (OCIB) when output compare flag B (OCFB) in the timer control/status register (TCSR) is set to 1.

Bit 6**OCIEB Description**

0	Output compare interrupt request B (OCIB) is disabled. (Initial value)
1	Output compare interrupt request B (OCIB) is enabled.

Bit 5—Output Compare Interrupt Enable A (OCIEA): This bit selects whether to request output compare interrupt A (OCIA) when output compare flag A (OCFA) in the timer control/status register (TCSR) is set to 1.

Bit 5**OCIEA Description**

0	Output compare interrupt request A (OCIA) is disabled. (Initial value)
1	Output compare interrupt request A (OCIA) is enabled.

Bit 4—Timer Overflow Interrupt Enable (OVIE): This bit selects whether to request a free-running timer overflow interrupt (FOVI) when the timer overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

Bit 4**OVIE Description**

0	The free-running timer overflow interrupt request (FOVI) is disabled. (Initial value)
1	The free-running timer overflow interrupt request (FOVI) is enabled.

Bit 3—Output Enable B (OEB): This bit selects whether to enable or disable output of the logic level selected by the OLVLB bit in the timer control/status register (TCSR) at the output compare B pin when the FRC and OCRB values match.

Bit 3**OEB Description**

0	Output compare B output is disabled. (Initial value)
1	Output compare B output is enabled.

Bit 2—Output Enable A (OEA): This bit selects whether to enable or disable output of the logic level selected by the OLVLA bit in the timer control/status register (TCSR) at the output compare A pin when the FRC and OCRA values match.

Bit 2

OEA	Description
0	Output compare A output is disabled. (Initial value)
1	Output compare A output is enabled.

Bits 1 and 0—Clock Select (CKS1 and CKS0): These bits select external clock input or one of three internal clock sources for the FRC. External clock pulses are counted on the rising edge.

Bit 1 Bit 0

CKS1	CKS0	Description
0	0	Internal clock source ($\phi/4$) (Initial value)
0	1	Internal clock source ($\phi/8$)
1	0	Internal clock source ($\phi/32$)
1	1	External clock source (counted on the rising edge)*

* Output enable bit (bit 3) must be cleared to 0.

10.2.5 Timer Control/Status Register (TCSR)—H'FEA1, H'FEB1

Bit	7	6	5	4	3	2	1	0
	ICF	OCFB	OCFA	OVF	OLVLB	OLVLA	IEDG	CCLRA
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/W	R/W	R/W	R/W

The TCSR is an 8-bit readable and partially writable* register that selects the input capture edge and output compare levels, and specifies whether to clear the counter on compare-match A. It also contains four status flags.

The TCSR is initialized to H'00 at a reset and in the standby modes.

* Software can write a 0 in bits 7 to 4 to clear the flags, but cannot write a 1 in these bits.

Bit 7—Input Capture Flag (ICF): This status flag is set to 1 to indicate an input capture event. It signifies that the FRC value has been copied to the ICR.

Bit 7

ICF	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the ICF bit after it has been set to 1, then writes a 0 in this bit. 2. The data transfer controller (DTC) serves an input capture interrupt .
1	This bit is set to 1 when an input capture signal causes the FRC value to be copied to the ICR.

Bit 6—Output Compare Flag B (OCFB): This status flag is set to 1 when the FRC value matches the OCRB value.

Bit 6

OCFB	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the OCFB bit after it has been set to 1, then writes a 0 in this bit. 2. The data transfer controller (DTC) serves output compare interrupt B.
1	This bit is set to 1 when FRC = OCRB.

Bit 5—Output Compare Flag A (OCFA): This status flag is set to 1 when the FRC value matches the OCRA value.

Bit 5

OCFA	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the OCFA bit after it has been set to 1, then writes a 0 in this bit. 2. The data transfer controller (DTC) serves output compare interrupt A.
1	This bit is set to 1 when FRC = OCRA.

Bit 4—Timer Overflow Flag (OVF): This status flag is set to 1 when the FRC overflows (changes from H'FFFF to H'0000).

Bit 4

OVF	Description
0	This bit is cleared from 1 to 0 when the CPU reads (Initial value) the OVF bit after it has been set to 1, then writes a 0 in this bit.
1	This bit is set to 1 when FRC changes from H'FFFF to H'0000.

Bit 3—Output Level B (OLVLB): This bit selects the logic level to be output at the FTOB pin when the FRC and OCRB values match.

Bit 3**OLVLB Description**

0	A 0 logic level (Low) is output for compare-match B. (Initial value)
1	A 1 logic level (High) is output for compare-match B.

Bit 2—Output Level A (OLVLA): This bit selects the logic level to be output at the FTOA pin when the FRC and OCRA values match.

Bit 2**OLVLA Description**

0	A 0 logic level (Low) is output for compare-match A. (Initial value)
1	A 1 logic level (High) is output for compare-match A.

Bit 1—Input Edge Select (IEDG): This bit selects whether to capture the count on the rising or falling edge of the input capture signal.

Bit 1**IEDG Description**

0	The FRC value is copied to the ICR on the falling edge of the input capture signal. (Initial value)
1	The FRC value is copied to the ICR on the rising edge of the input capture signal.

Bit 0—Counter Clear A (CCLRA): This bit selects whether to clear the FRC at compare-match A (when the FRC and OCRA values match).

Bit 0**CCLRA Description**

0	The FRC is not cleared. (Initial value)
1	The FRC is cleared at compare-match A.

10.3 CPU Interface

The FRC, OCRA, OCRB, and ICR are 16-bit registers, but they are connected to an 8-bit data bus. When the CPU accesses these four registers, to ensure that both bytes are written or read simultaneously, the access is performed using an 8-bit temporary register (TEMP).

These registers are written and read as follows.

- **Register Write**
When the CPU writes to the upper byte, the upper byte of write data is placed in TEMP. Next, when the CPU writes to the lower byte, this byte of data is combined with the byte in TEMP and all 16 bits are written in the register simultaneously.
- **Register Read**
When the CPU reads the upper byte, the upper byte of data is sent to the CPU and the lower byte is placed in TEMP. When the CPU reads the lower byte, it receives the value in TEMP.

Programs that access these four registers should normally use word access. Equivalently, they may access first the upper byte, then the lower byte. Data will not be transferred correctly if the bytes are accessed in reverse order, or if only one byte is accessed.

Coding Examples : Write the contents of R0 into OCRA in FRT1

MOV.W R0, @H'FEAA

: Read ICR of FRT2

MOV.W, @H'FEB8, R0

The same considerations apply to access by the DTC.

Figure 10-2 shows the data flow when the FRC is accessed. The other registers are accessed in the same way, except that when OCRA or OCRB is read, the upper and lower bytes are both transferred directly to the CPU without using the temporary register.

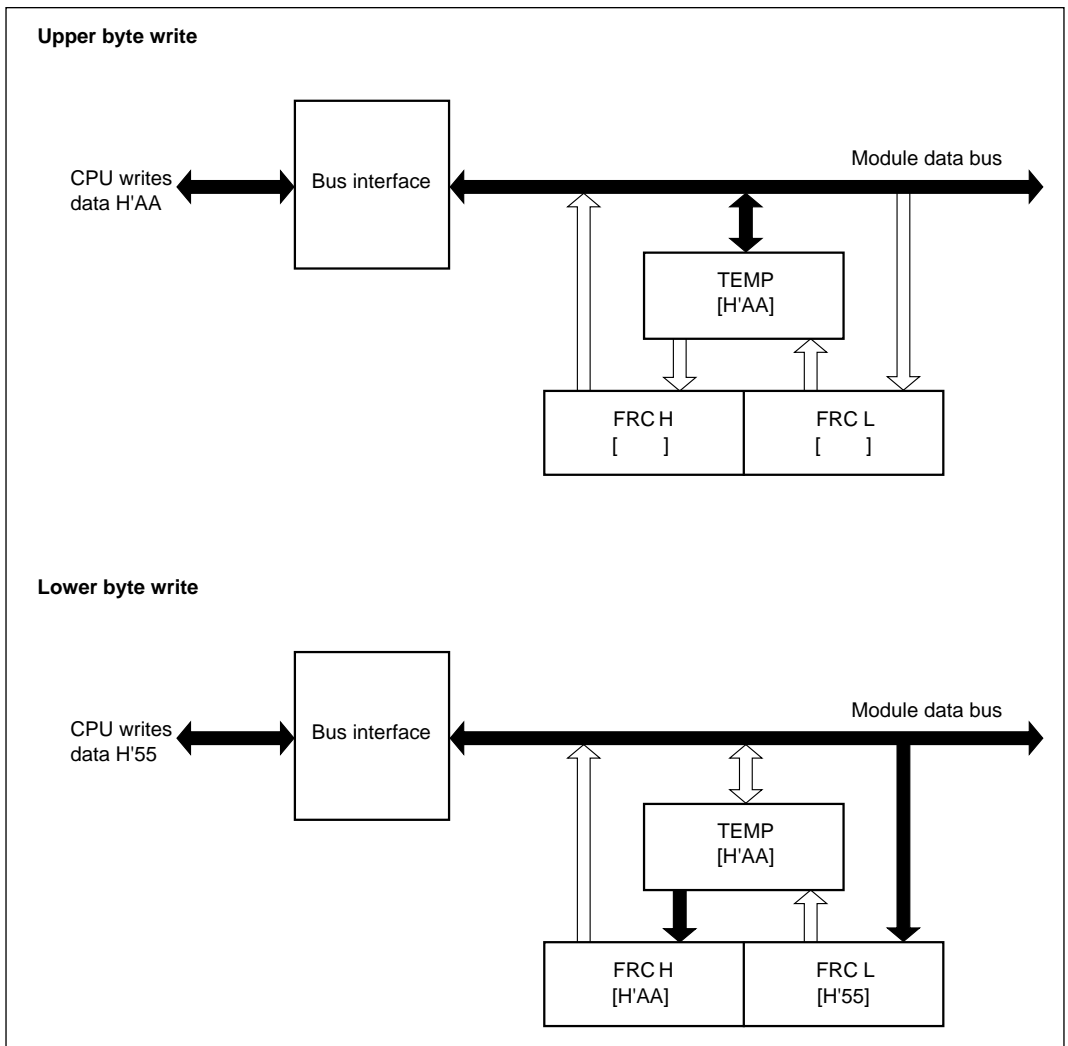


Figure 10-2 (a) Write Access to FRC (When CPU Writes H'AA55)

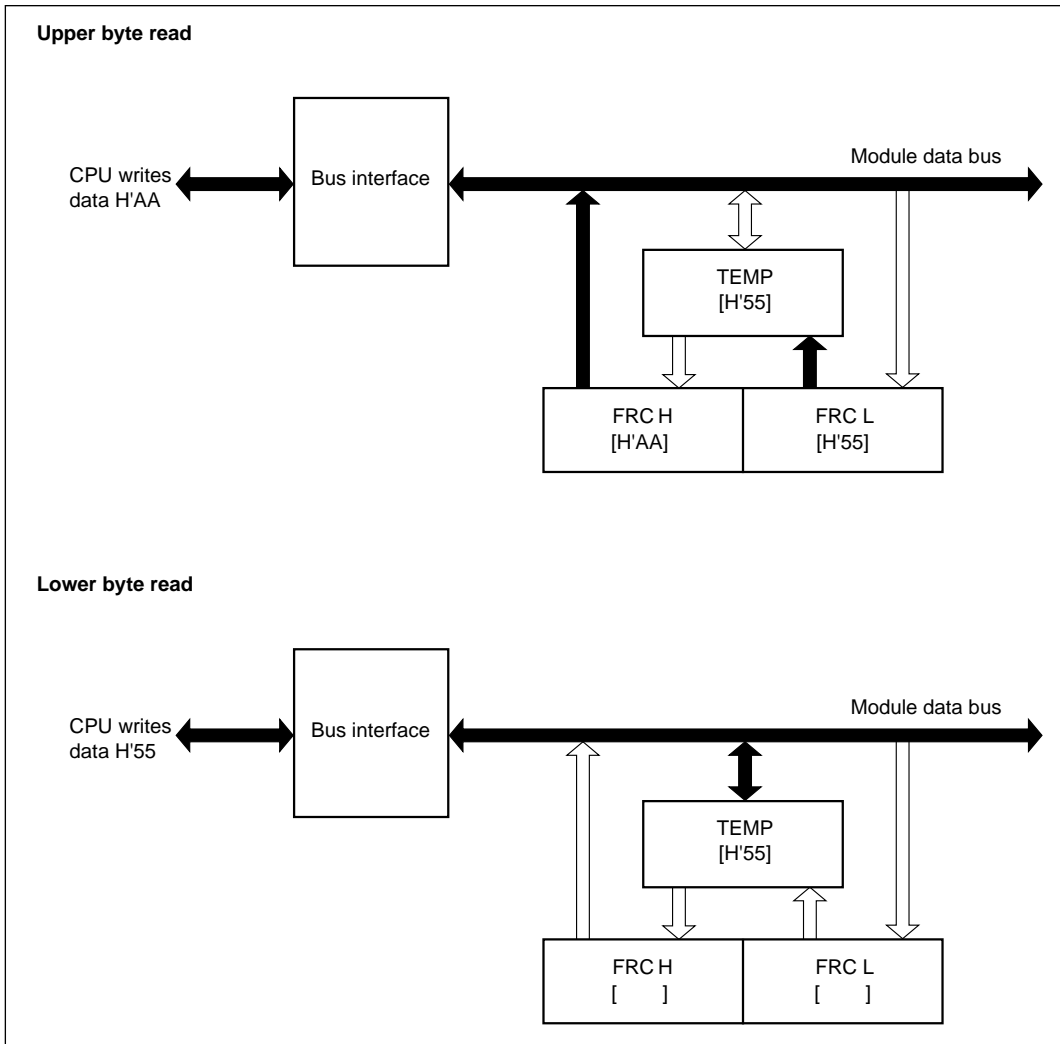


Figure 10-2 (b) Read Access to FRC (When FRC Contains H'AA55)

10.4 Operation

10.4.1 FRC Incrementation Timing

The FRC increments on a pulse generated once for each period of the selected (internal or external) clock source.

If external clock input is selected, the FRC increments on the rising edge of the clock signal. Figure 10-3 shows the increment timing.

The pulse width of the external clock signal must be at least 1.5ϕ clock periods. The counter will not increment correctly if the pulse width is shorter than 1.5ϕ clock periods.

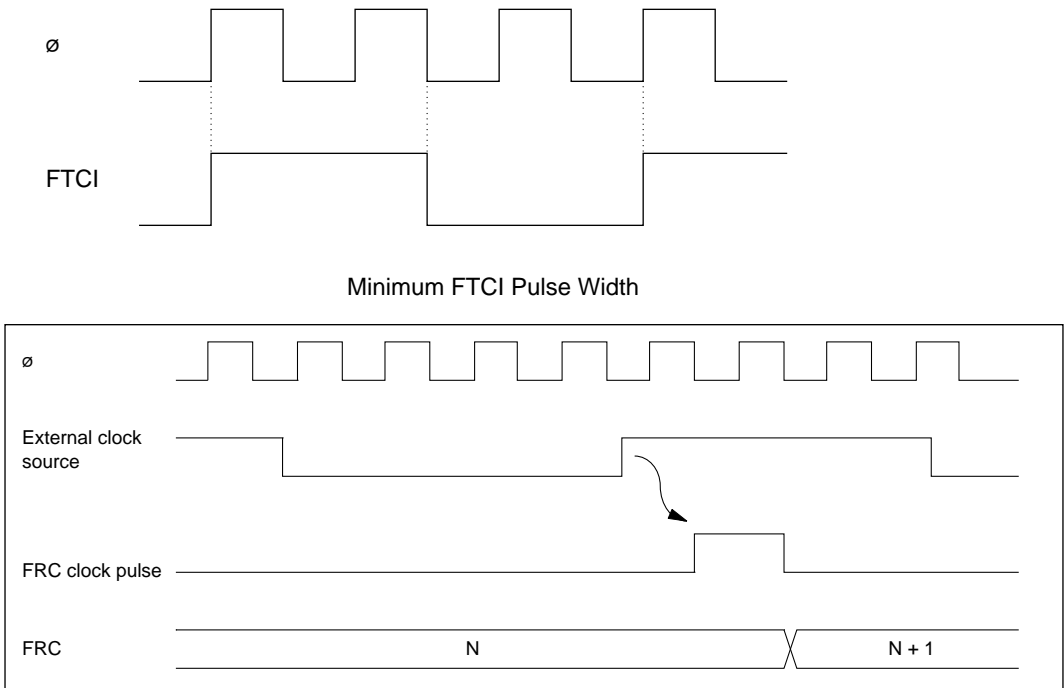


Figure 10-3 Increment Timing for External Clock Input

10.4.2 Output Compare Timing

Setting of Output Compare Flags A and B (OCFA and OCFB): The output compare flags are set to 1 by an internal compare-match signal generated when the FRC value matches the OCRA or OCRB value. This compare-match signal is generated at the last state in which the two values match, just before the FRC increments to a new value.

Accordingly, when the FRC and OCR values match, the compare-match signal is not generated until the next period of the clock source. Figure 10-4 shows the timing of the setting of the output compare flags.

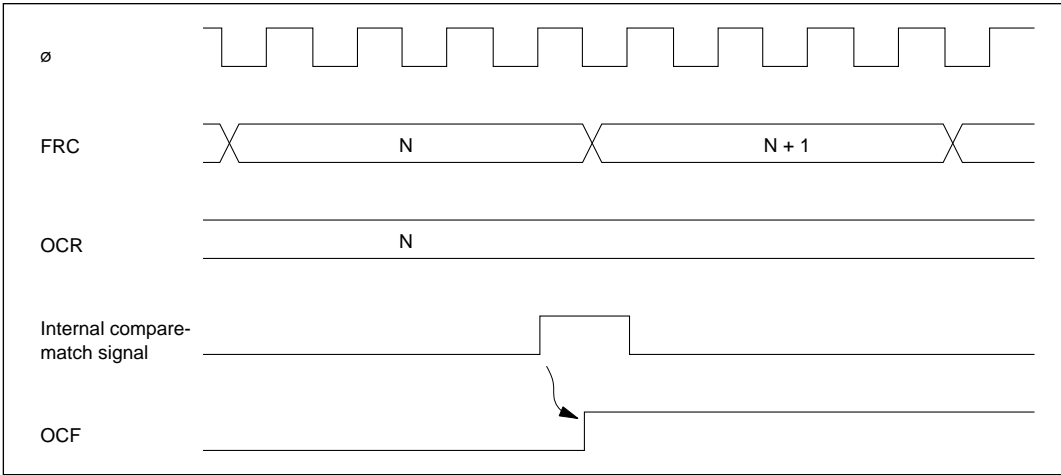


Figure 10-4 Setting of Output Compare Flags

Output Timing: When a compare-match occurs, the logic level selected by the output level bit (OLVLA or OLVLB) in the TCSR is output at the output compare pin (FTOA or FTOB). Figure 10-5 shows the timing of this operation for compare-match A.

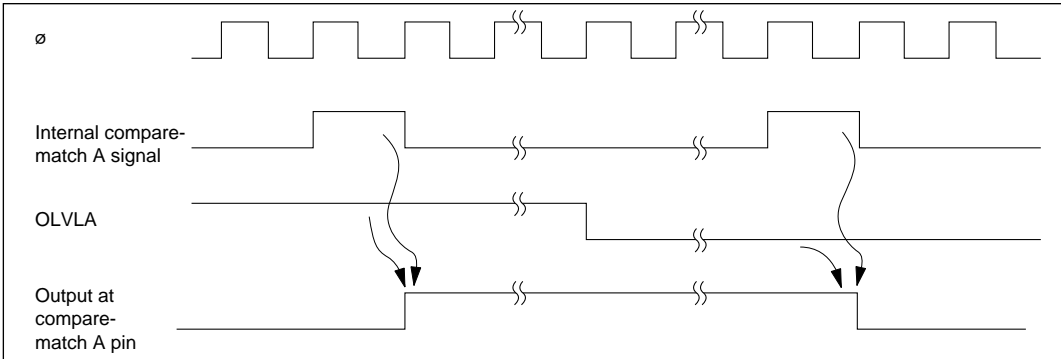


Figure 10-5 Timing of Output Compare A

FRC Clear Timing: If the CCLRA bit is set to 1, the FRC is cleared when compare-match A occurs. Figure 10-6 shows the timing of this operation.

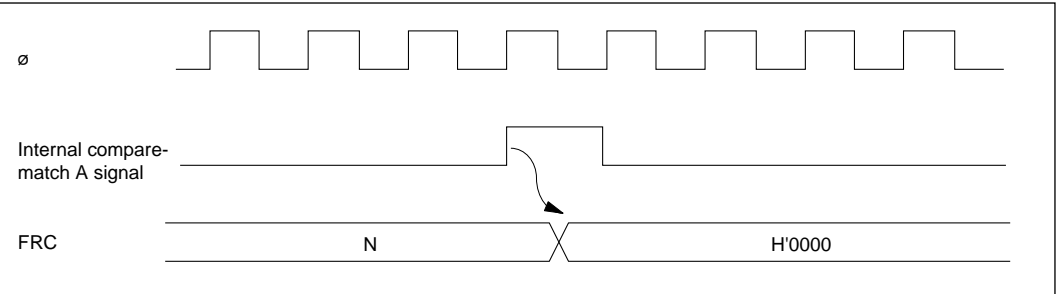


Figure 10-6 Clearing of FRC by Compare-Match A

10.4.3 Input Capture Timing

Input Capture Timing: An internal input capture signal is generated from the rising or falling edge of the input at the input capture pin (FTI), as selected by the IEDG bit in the TCSR. Figure 10-7 shows the usual input capture timing when the rising edge is selected (IEDG = 1).

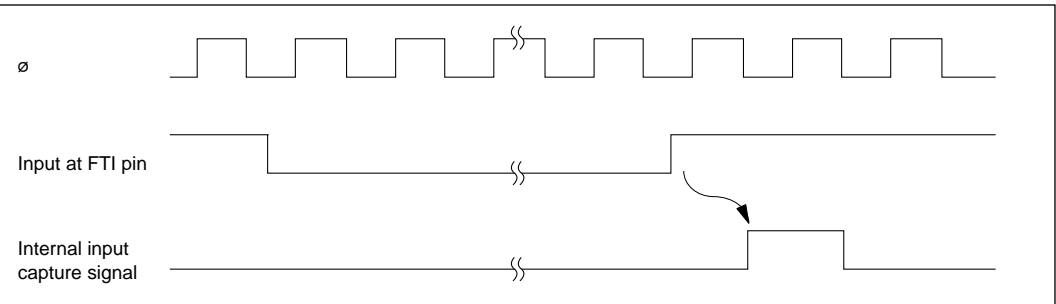


Figure 10-7 Input Capture Timing (Usual Case)

But if the upper byte of the ICR is being read when the input capture signal arrives, the internal input capture signal is delayed by one state. Figure 10-8 shows the timing for this case.

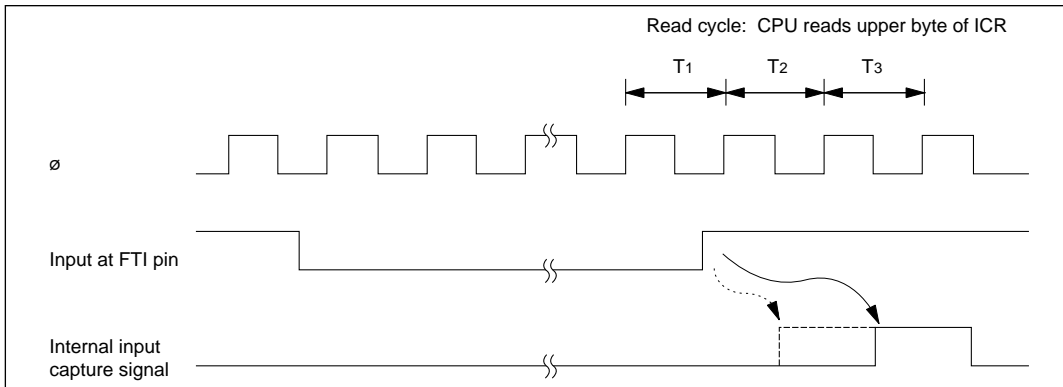


Figure 10-8 Input Capture Timing (1-State Delay)

Timing of Input Capture Flag (ICF) Setting: The input capture flag (ICF) is set to 1 by the internal input capture signal. Figure 10-9 shows the timing of this operation.

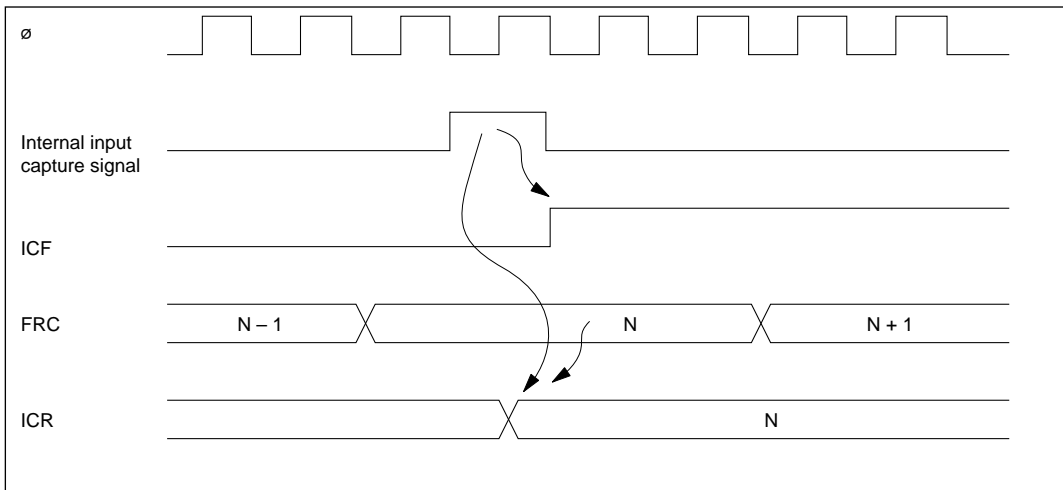


Figure 10-9 Setting of Input Capture Flag

10.4.4 Setting of FRC Overflow Flag (OVF)

The FRC overflow flag (OVF) is set to 1 when the FRC overflows (changes from H'FFFF to H'0000). Figure 10-10 shows the timing of this operation.

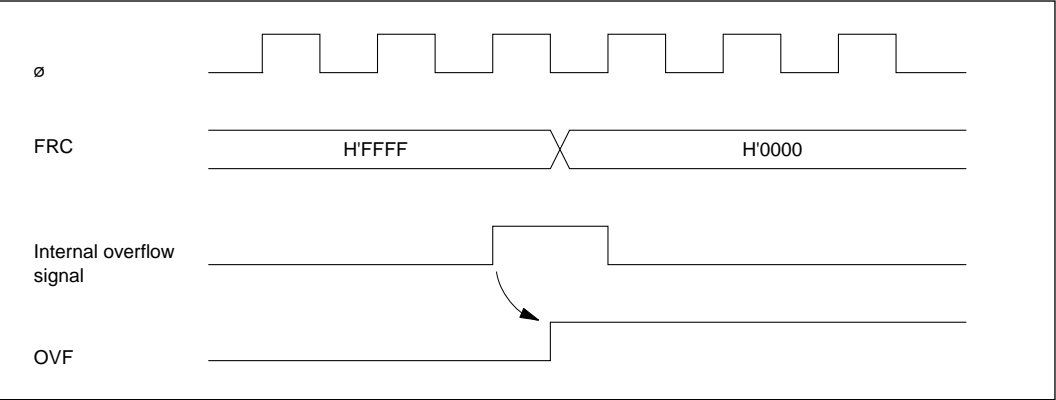


Figure 10-10 Setting of Overflow Flag (OVF)

10.5 CPU Interrupts and DTC Interrupts

Each free-running timer channel can request four types of interrupts: input capture (ICI), output compare A and B (OCIA and OCIB), and overflow (FOVI). Each interrupt is requested when the corresponding enable and flag bits are set. Independent signals are sent to the interrupt controller for each type of interrupt. Table 10-3 lists information about these interrupts.

Table 10-3 Free-Running Timer Interrupts

Interrupt	Description	DTC Service Available?	Priority
ICI	Requested when ICF is set	Yes	High ↑ Low
OCIA	Requested when OCFA is set	Yes	
OCIB	Requested when OCFB is set	Yes	
FOVI	Requested when OVF is set	No	Low

The ICI, OCIA, and OCIB interrupts can be directed to the data transfer controller (DTC) to have a data transfer performed in place of the usual interrupt-handling routine.

When the DTC serves one of these interrupts, it automatically clears the ICF, OCFA, or OCFB flag to 0. See section 6, Data Transfer Controller, for further information on the DTC.

10.6 Synchronization of Free-Running Timers 1 and 2

10.6.1 Synchronization after a Reset

The three free-running timer channels are synchronized at a reset and remained synchronized until:

- the clock source is changed;
- FRC contents are rewritten; or
- an FRC is cleared.

After a reset, each free-running counter operates on the $\phi/4$ internal clock source.

10.6.2 Synchronization by Writing to FRCs

When synchronization of free-running timers 1 and 2 is lost, it can be restored by writing to the free-running counters.

Synchronization on Internal Clock Source: When an internal clock is selected, free-running timers 1 and 2 can be synchronized by writing data to their free-running counters as indicated in table 10-4.

Table 10-4 Synchronization by Writing to FRCs

Clock Source	Write Interval	Write Data	
$\phi/4$	$4n$ (states)	m	(FRC1)
$\phi/8$	$8n$ (states)	$m + n$	(FRC2)
$\phi/32$	$32n$ (states)		

m, n : Arbitrary integers

Note: When the FRC1 count is $m + n$, the same value must be written at the timing indicated in table 10-4.

After writing these data, synchronization can be checked by reading the free-running counters at the same interval as the write interval. If the read data have the same relative differences as the write data, the free-running timers are synchronized.

Examples of synchronizing programs are shown next.

Examples a, b, and c can be executed from a memory area accessed in two states via a 16-bit bus. Examples d, e, and f can be executed from a memory area accessed in three states via an 8-bit bus. These examples assume that no wait states (T_w) are inserted and no NMI input occurs.

Example a: $\phi/4$ clock source, 12-state write interval ($n = 3$), 16-bit bus, two-state-access memory

```

LA:   LDC.B  #H'FE, BR           ; Initialize base register for short-format instruction (MOV:S)
      LDC.W  #H'0700, SR        ; Raise interrupt mask level to 7
      MOV.W  #m, R1             ; Data for free-running timer 1
      MOV.W  #m+3, R2           ; Data for free-running timer 2 ( $m + n = m + 3$ )
      BSR    SET4               ; Call write routine
      .
      .ALIGN 2                  ; Align write instructions (MOV:S) at even address
SET4: MOV:S.W R1, @H'A2:8        ; Write to FRC 1 (address H'FEA2)  9 states
      BRN    SET4:8             ; 2-Byte dummy instruction        3 states
      MOV:S.W R2, @H'B2:8        ; Write to FRC 2 (address H'FEB2)
      RTS

```

Total 12 states

Example b: $\phi/8$ clock source, 16-state write interval ($n = 2$), 16-bit bus, two-state-access memory

```

LB:   LDC.B  #H'FE, BR
      LDC.W  #H'0700, SR
      MOV.W  #m, R1
      MOV.W  #m+2, R2
      BSR    SET8
      .
      .ALIGN 2
SET8: MOV:S.W R1, @H'A2:8        ; 9 States
      BRN    SET8:8             ; 3 States
      XCH    R1, R1              ; 4 States
      MOV:S.W R2, @H'B2:8
      RTS

```

Total 16 states

Example c: $\phi/32$ clock source, 32-state write interval ($n = 1$), 16-bit bus, two-state-access memory

LC:	LDC.B #H'FE, BR		
	LDC.W #H'0700, SR		
	MOV.W #m, R1		
	MOV.W #m+1, R2		
	BSR SET32		
	...		
	.ALIGN 2		; Align on even address
SET32:	MOV:S.W R1, @H'A2:8		; 2 Bytes, 9 states
	BSR WAIT:8		; 2 Bytes, 9 states
	MOV:S.W R2, @H'B2:8		
	RTS		
	.ALIGN 2		; Align on even address
WAIT:	NOP		; 2 States
	XCH R1, R1		; 4 States
	RTS		; 8 States

Total 32 states

Note: The stack is assumed to be in a memory area accessed in two states via a 16-bit bus.

Example d: $\phi/4$ clock source, 20-state write interval ($n = 5$), 8-bit bus, three-state-access memory

LD:	LDC.B #H'FE, BR		
	LDC.W #H'0700, SR		; Set interrupt mask level to 7
	CLR.B @H'F8:8		; Disable wait states
	MOV.W #m, R1		
	MOV.W #m+5, R2		
	MOV:S.W R1, @H'A2:8		; 13 States
	BRN LD:8		; 2 Bytes, 7 states
	MOV:S.W R2, @H'B2:8		

Total 20 states

Example e: $\phi/8$ clock source, 24-state write interval ($n = 3$), 8-bit bus, three-state-access memory

LE:	LDC.B #H'FE, BR			
	LDC.W #H'0700, SR			
	CLR.B @H'F8:8			
	MOV.W #m, R1			
	MOV.W #m+3, R2			
	MOV:S.W R1, @H'A2:8	; 13 States	—————	} Total 24 states
	BRN LE:8	; 2 Bytes,	7 states ———	
	NOP	; 1 Byte,	4 states ———	
	MOV:S.W R2, @H'B2:8			

Example f: $\phi/32$ clock source, 32-state write interval ($n = 1$), 8-bit bus, three-state-access memory

LF:	LDC.B #H'FE, BR			
	LDC.W #H'0700, SR			
	CLR.B @H'F8:8			
	MOV.W #m, R1			
	MOV.W #m+1, R2			
	MOV:S.W R1, @H'A2:8	;	13 states ———	} Total 32 states
	XCH R0, R0	;	8 states ———	
	BRN LF:8	; 2 Bytes,	7 states ———	
	NOP	;	4 states ———	
	MOV:S.W R2, @H'B2:8			

Synchronization on External Clock Source: When the external clock source is selected, the free-running timers can be synchronized by halting their external clock inputs, then writing identical values in their free-running counters.

10.7 Sample Application

In the example below, one free-running timer channel is used to generate two square-wave outputs with a 50% duty factor and arbitrary phase relationship. The programming is as follows:

1. The CCLRA bit in the TCSR is set to 1.
2. Each time a compare-match interrupt occurs, software inverts the corresponding output level bit in the TCSR.

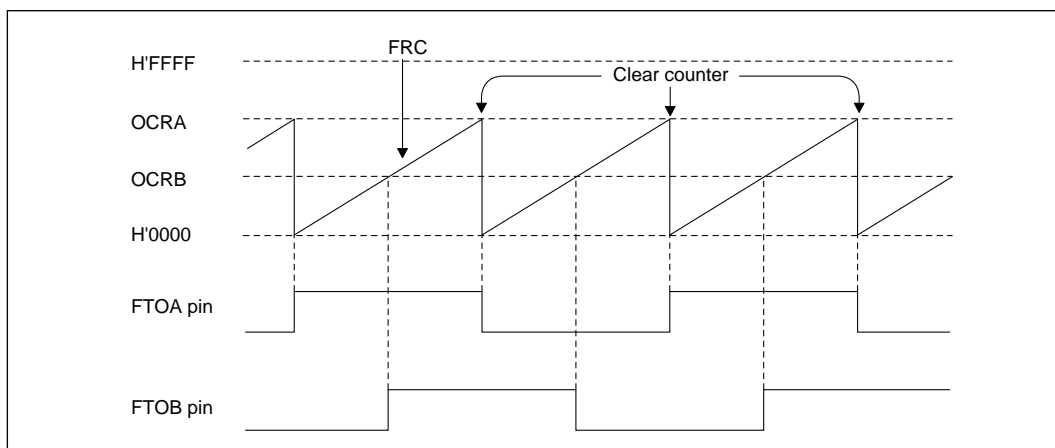


Figure 10-11 Square-Wave Output (Example)

10.8 Application Notes

Application programmers should note that the following types of contention can occur in the free-running timers.

Contention between FRC Write and Clear: If an internal counter clear signal is generated during the T3 state of a write cycle to the lower byte of a free-running counter, the clear signal takes priority and the write is not performed.

Figure 10-12 shows this type of contention.

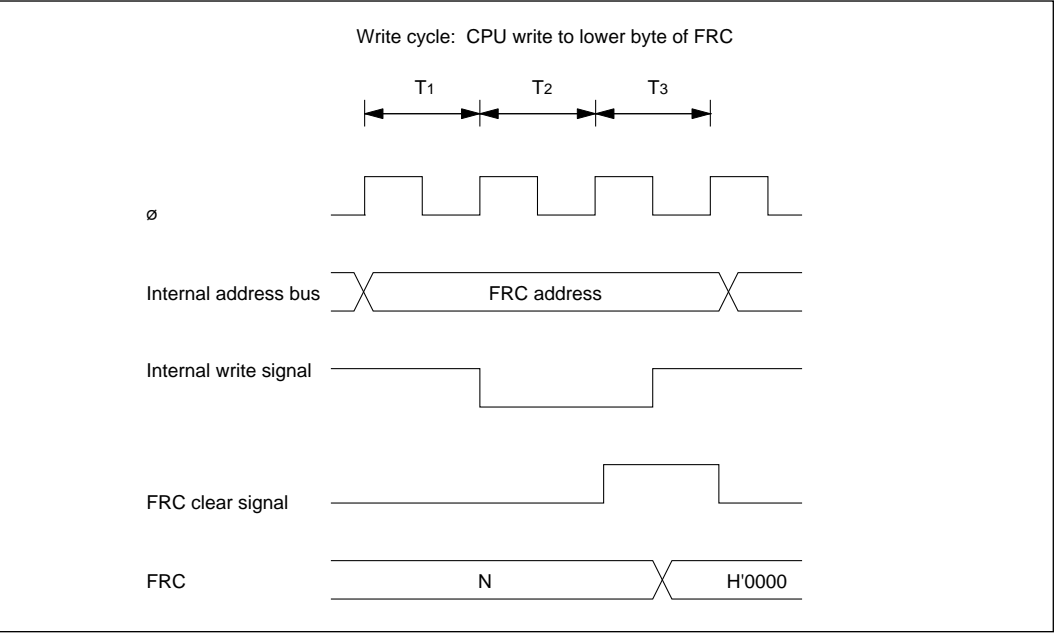


Figure 10-12 FRC Write-Clear Contention

Contention between FRC Write and Increment: If an FRC increment pulse is generated during the T3 state of a write cycle to the lower byte of a free-running counter, the write takes priority and the FRC is not incremented.

Figure 10-13 shows this type of contention.

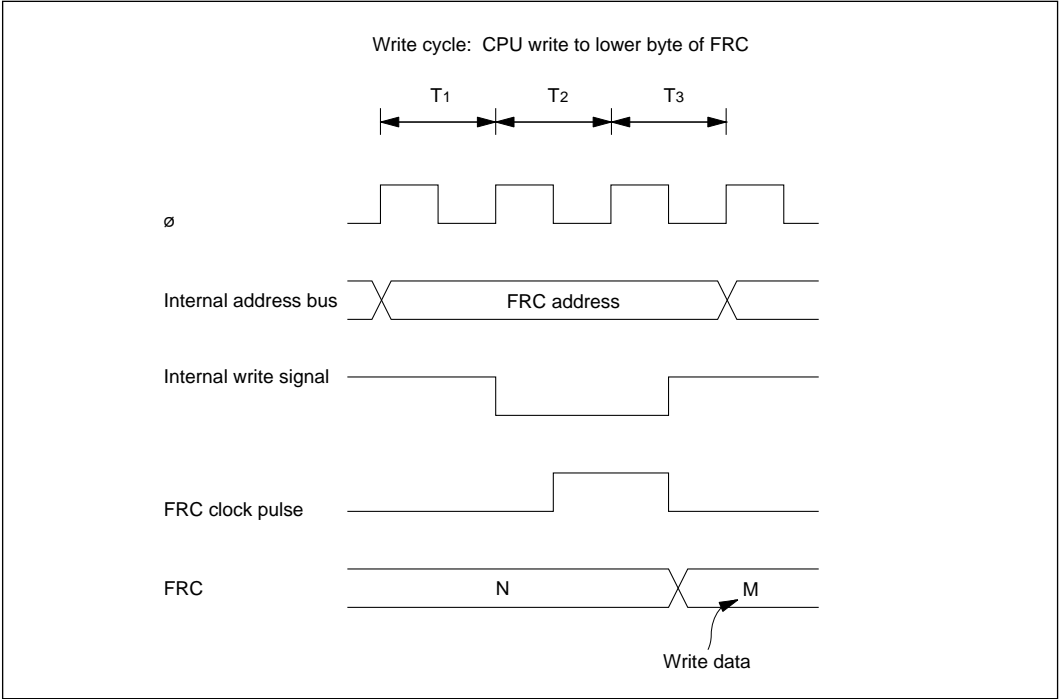


Figure 10-13 FRC Write-Increment Contention

Contention between OCR Write and Compare-Match: If a compare-match occurs during the T3 state of a write cycle to the lower byte of OCRA or OCRB, the write takes precedence and the compare-match signal is inhibited.

Figure 10-14 shows this type of contention.

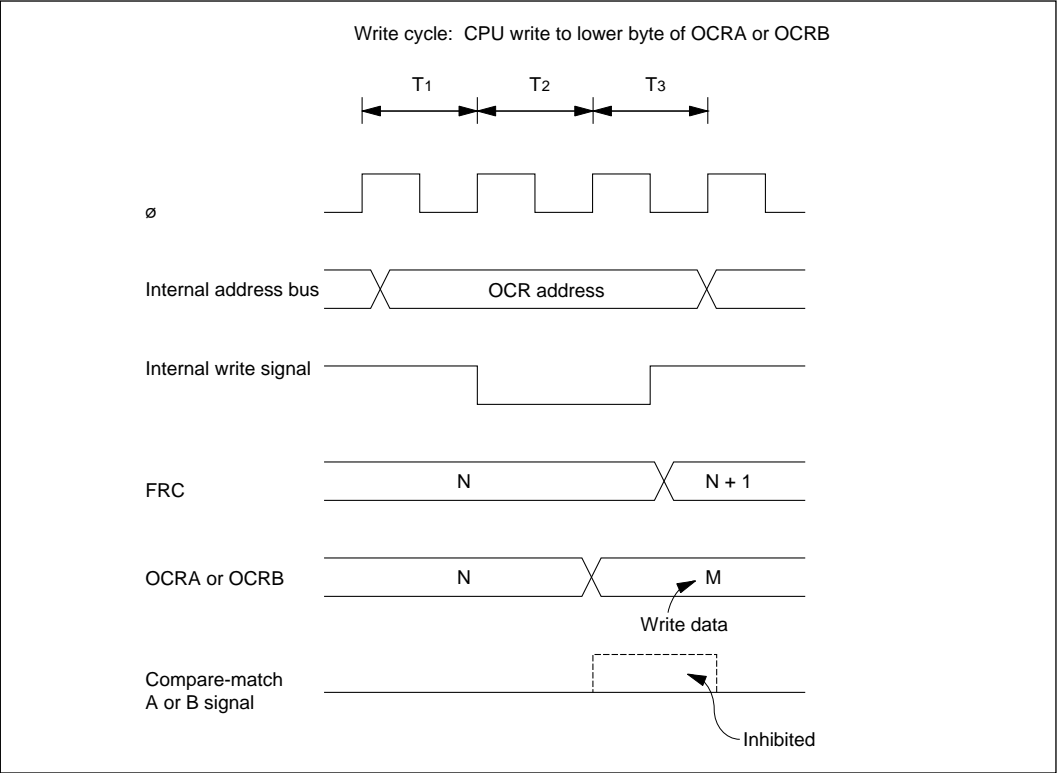


Figure 10-14 Contention between OCR Write and Compare-Match

Incrementation Caused by Changing of Internal Clock Source: When an internal clock source is changed, the changeover may cause the FRC to increment. This depends on the time at which the clock select bits (CKS1 and CKS0) are rewritten, as shown in table 10-5.

The pulse that increments the FRC is generated at the falling edge of the internal clock source. If clock sources are changed when the old source is High and the new source is Low, as in case No. 3 in table 10-5, the changeover generates a falling edge that triggers the FRC increment pulse.

Switching between an internal and external clock source can also cause the FRC to increment.

Table 10-5 Effect of Changing Internal Clock Sources

No.	Description	Timing Chart
1	Low → Low: CKS1 and CKS0 are rewritten while both clock sources are Low.	<p>Old clock source</p> <p>New clock source</p> <p>FRC clock pulse</p> <p>FRC</p> <p>N N + 1</p> <p>CKS rewrite</p>
2	Low → High: CKS1 and CKS0 are rewritten while old clock source is Low and new clock source is High.	<p>Old clock source</p> <p>New clock source</p> <p>FRC clock pulse</p> <p>FRC</p> <p>N N + 1 N + 2</p> <p>CKS rewrite</p>
3	High → Low: CKS1 and CKS0 are rewritten while old clock source is High and new clock source is Low.	<p>Old clock source</p> <p>New clock source</p> <p>FRC clock pulse</p> <p>FRC</p> <p>N N + 1 N + 2</p> <p>CKS rewrite</p>

* The switching of clock sources is regarded as a falling edge that increments the FRC.

Table 10-5 Effect of Changing Internal Clock Sources (cont)

No.	Description	Timing Chart
4	High → High: CKS1 and CKS0 are rewritten while both clock sources are High.	<p>The timing chart illustrates the transition from an old clock source to a new clock source. The 'Old clock source' and 'New clock source' signals are both high during the transition. The 'FRC clock pulse' is generated during the transition. The 'FRC' signal shows three clock cycles labeled N, N+1, and N+2. A vertical dashed line labeled 'CKS rewrite' is positioned between N+1 and N+2.</p>

Section 11 8-Bit Timer

11.1 Overview

The H8/510 chip includes a single 8-bit timer based on an 8-bit counter (TCNT). The timer has two time constant registers (TCORA and TCORB) that are constantly compared with the TCNT value to detect compare-match events. One application of the 8-bit timer is to generate a rectangular-wave output with an arbitrary duty factor.

11.1.1 Features

The features of the 8-bit timer are listed below.

- Selection of four clock sources
The counter can be driven by an internal clock signal ($\phi/8$, $\phi/64$, or $\phi/1024$) or an external clock input (enabling use as an external event counter).
- Selection of three ways to clear the counter
The counter can be cleared on compare-match A or B, or by an external reset signal.
- Timer output controlled by two time constants
The single timer output (TMO) is controlled by two independent time constants, enabling the timer to generate output waveforms with an arbitrary duty factor.
- Three types of interrupts
Compare-match A and B and overflow interrupts can be requested independently.
The compare match interrupts can be served by the data transfer controller (DTC), enabling interrupt-driven data transfer with minimal CPU programming.

11.1.2 Block Diagram

Figure 11-1 shows a block diagram of 8-bit timer.

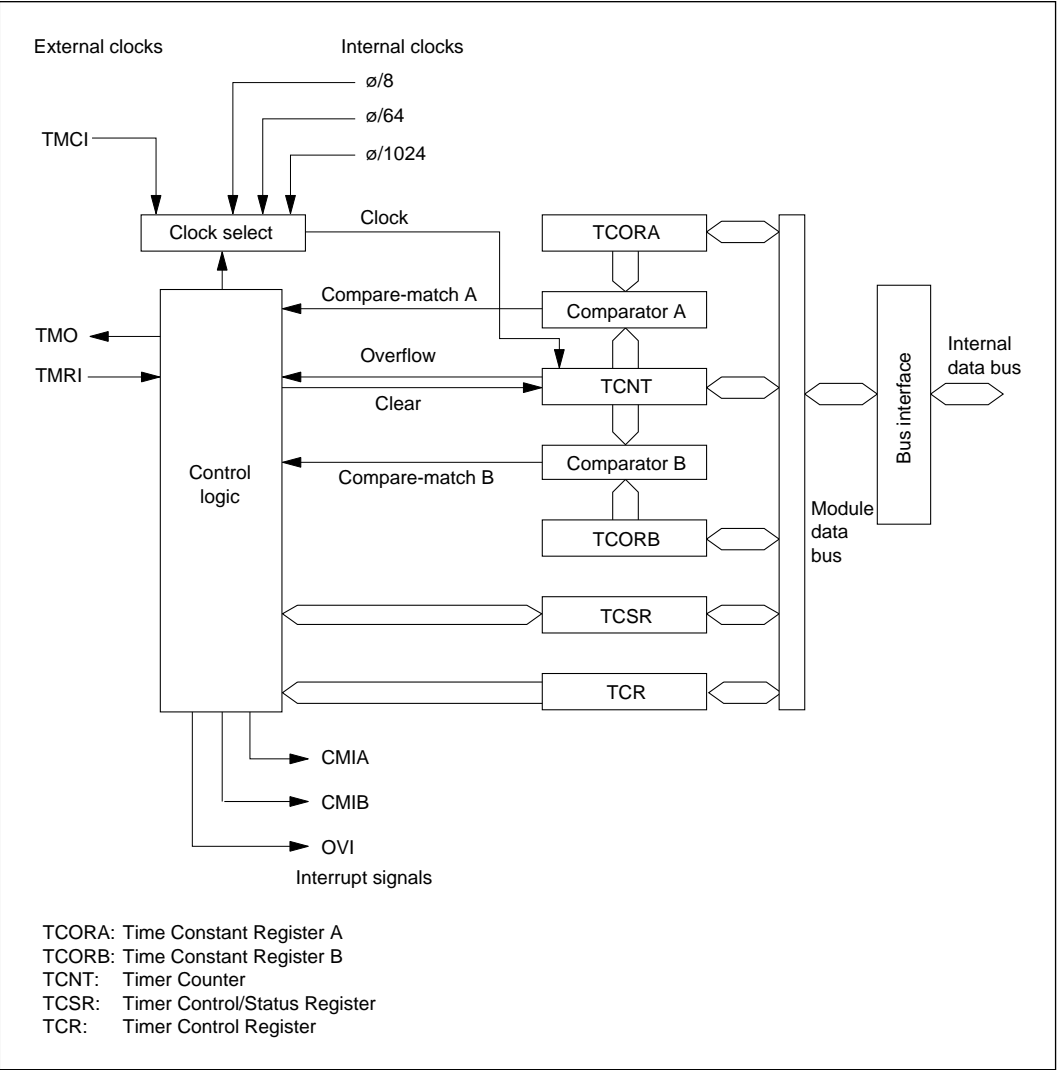


Figure 11-1 Block Diagram of 8-Bit Timer

11.1.3 Input and Output Pins

Table 11-1 lists the input and output pins of the 8-bit timer.

Table 11-1 Input and Output Pins of 8-Bit Timer

Name	Abbreviation	I/O	Function
Timer output	TMO	Output	Output controlled by compare-match
Timer clock input	TMCI	Input	External clock source for the counter
Timer reset input	TMRI	Input	External reset signal for the counter

11.1.4 Register Configuration

Table 11-2 lists the registers of the 8-bit timer.

Table 11-2 8-Bit Timer Registers

Name	Abbreviation	R/W	Initial Value	Address
Timer control register	TCR	R/W	H'00	H'FEC0
Timer control/status register	TCSR	R/(W)*	H'10	H'FEC1
Timer constant register A	TCORA	R/W	H'FF	H'FEC2
Timer constant register B	TCORB	R/W	H'FF	H'FEC3
Timer counter	TCNT	R/W	H'00	H'FEC4

* Software can write a “0” to clear bits 7 to 5, but cannot write a “1” in these bits.

11.2 Register Descriptions

11.2.1 Timer Counter (TCNT)—H'FEC4

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The timer counter (TCNT) is an 8-bit up-counter that increments on a pulse generated from one of four clock sources. The clock source is selected by clock select bits 2 to 0 (CKS2 to CKS0) of the timer control register (TCR). The CPU can always read or write the timer counter.

The timer counter can be cleared by an external reset input or by an internal compare-match signal generated at a compare-match event. Clock clear bits 1 and 0 (CCLR1 and CCLR0) of the timer control register select the method of clearing.

When the timer counter overflows from H'FF to H'00, the overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

The timer counter is initialized to H'00 at a reset and in the standby modes.

11.2.2 Time Constant Registers A and B (TCORA and TCORB)—H'FEC2 and H'FEC3

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TCORA and TCORB are 8-bit readable/writable registers. The timer count is continually compared with the constants written in these registers. When a match is detected, the corresponding compare-match flag (CMFA or CMFB) is set in the timer control/status register (TCSR).

The timer output signal (TMO) is controlled by these compare-match signals as specified by output select bits 1 to 0 (OS1 to OS0) in the timer status/control register (TCSR).

TCORA and TCORB are initialized to H'FF at a reset and in the standby modes.

11.2.3 Timer Control Register (TCR)—H'FEC0

Bit	7	6	5	4	3	2	1	0
	CMIEB	CMIEA	OVIE	CCLR1	CCLR0	CKS2	CKS1	CKS0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The TCR is an 8-bit readable/writable register that selects the clock source and the time at which the timer counter is cleared, and enables interrupts.

The TCR is initialized to H'00 at a reset and in the standby modes.

Bit 7—Compare-Match Interrupt Enable B (CMIEB): This bit selects whether to request compare-match interrupt B (CMIB) when compare-match flag B (CMFB) in the timer control/status register (TCSR) is set to 1.

Bit 7

CMIEB	Description
0	Compare-match interrupt request B (CMIB) is disabled. (Initial value)
1	Compare-match interrupt request B (CMIB) is enabled.

Bit 6—Compare-Match Interrupt Enable A (CMIEA): This bit selects whether to request compare-match interrupt A (CMIA) when compare-match flag A (CMFA) in the timer control/status register (TCSR) is set to 1.

Bit 6

CMIEA	Description
0	Compare-match interrupt request A (CMIA) is disabled. (Initial value)
1	Compare-match interrupt request A (CMIA) is enabled.

Bit 5—Timer Overflow Interrupt Enable (OVIE): This bit selects whether to request a timer overflow interrupt (OVI) when the overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

Bit 5

OVIE	Description
0	The timer overflow interrupt request (OVI) is disabled. (Initial value)
1	The timer overflow interrupt request (OVI) is enabled.

Bits 4 and 3—Counter Clear 1 and 0 (CCLR1 and CCLR0): These bits select how the timer counter is cleared: by compare-match A or B or by an external reset input.

Bit 4 CCLR1	Bit 3 CCLR0	Description
0	0	Not cleared. (Initial value)
0	1	Cleared on compare-match A.
1	0	Cleared on compare-match B.
1	1	Cleared on rising edge of external reset input signal.

Bits 2, 1, and 0—Clock Select (CKS2, CKS1, and CKS0): These bits select the internal or external clock source for the timer counter. For the external clock source they select whether to increment the count on the rising or falling edge of the clock input, or on both edges.

Bit 2 CKS2	Bit 1 CKS1	Bit 0 CKS0	Description
0	0	0	No clock source (timer stopped). (Initial value)
0	0	1	Internal clock source ($\phi/8$).
0	1	0	Internal clock source ($\phi/64$).
0	1	1	Internal clock source ($\phi/1024$).
1	0	0	No clock source (timer stopped).
1	0	1	External clock source, counted on the rising edge.
1	1	0	External clock source, counted on the falling edge.
1	1	1	External clock source, counted on both the rising and falling edges.

11.2.4 Timer Control/Status Register (TCSR)—H'FEC1

Bit	7	6	5	4	3	2	1	0
	CMFB	CMFA	OVF	—	OS3	OS2	OS1	OS0
Initial value	0	0	0	1	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	—	R/W	R/W	R/W	R/W

The TCSR is an 8-bit readable and partially writable* register that indicates compare-match and overflow status and selects the effect of compare-match events on the timer output signal (TMO).

The TCSR is initialized to H'10 at a reset and in the standby modes.

* Software can write a “0” in bits 7 to 5 to clear the flags, but cannot write a 1 in these bits.

Bit 7—Compare-Match Flag B (CMFB): This status flag is set to 1 when the timer count matches the time constant set in TCORB.

Bit 7**CMFB Description**

0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the CMFB bit after it has been set to 1, then writes a 0 in this bit. 2. Compare-match interrupt B is served by the data transfer controller (DTC).
1	This bit is set to 1 when TCNT = TCORB.

Bit 6—Compare-Match Flag A (CMFA): This status flag is set to 1 when the timer count matches the time constant set in TCORA.

Bit 6**CMFA Description**

0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the CMFA bit after it has been set to 1, then writes a 0 in this bit. 2. Compare-match interrupt A is served by the data transfer controller (DTC).
1	This bit is set to 1 when TCNT = TCORA.

Bit 5—Timer Overflow Flag (OVF): This status flag is set to 1 when the timer count overflows (changes from H'FF to H'00).

Bit 5**OVF Description**

0	This bit is cleared from 1 to 0 when the CPU reads the OVF bit after it has been set to 1, then writes a 0 in this bit.
1	This bit is set to 1 when TCNT changes from H'FF to H'00.

Bit 4—Reserved: This bit cannot be modified and is always read as 1.

Bits 3 to 0—Output Select 3 to 0 (OS3 to OS0): These bits specify the effect of compare-match events on the timer output signal (TMO). Bits OS3 and OS2 control the effect of compare-match B on the output level. Bits OS1 and OS0 control the effect of compare-match A on the output level.

When all four output select bits are cleared to 0 the TMO signal is not output. The TMO output is 0 before the first compare-match.

Bit 3 OS3	Bit 2 OS2	Description
0	0	No change when compare-match B occurs. (Initial value)
0	1	Output changes to 0 when compare-match B occurs.
1	0	Output changes to 1 when compare-match B occurs.
1	1	Output inverts (toggles) when compare-match B occurs.

Bit 1 OS1	Bit 0 OS0	Description
0	0	No change when compare-match A occurs. (Initial value)
0	1	Output changes to 0 when compare-match A occurs.
1	0	Output changes to 1 when compare-match A occurs.
1	1	Output inverts (toggles) when compare-match A occurs.

11.3 Operation

11.3.1 TCNT Incrementation Timing

The timer counter increments on a pulse generated once for each period of the selected (internal or external) clock source.

If external clock input (TMCI) is selected, the timer counter can increment on the rising edge, the falling edge, or both edges of the external clock signal.

The external clock pulse width must be at least $1.5 \cdot \phi$ clock periods for incrementation on a single edge, and at least $2.5 \cdot \phi$ clock periods for incrementation on both edges. The counter will not increment correctly if the pulse width is shorter than these values.

Figure 11-2 shows the count timing for the case of incrementation on both edges of an external clock input.

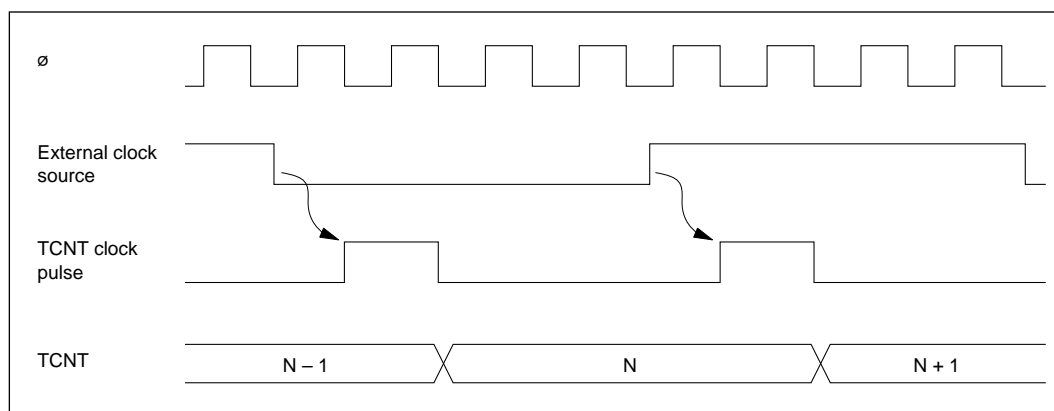


Figure 11-2 Count Timing for External Clock Input

11.3.2 Compare Match Timing

Setting of Compare-Match Flags A and B (CMFA and CMFB): The compare-match flags are set to 1 by an internal compare-match signal generated when the timer count matches the time constant in TCORA or TCORB. The compare-match signal is generated at the last state in which the match is true, just before the timer counter increments to a new value.

Accordingly, when the timer count matches one of the time constants, the compare-match signal is not generated until the next period of the clock source. Figure 11-3 shows the timing of the setting of the compare-match flags.

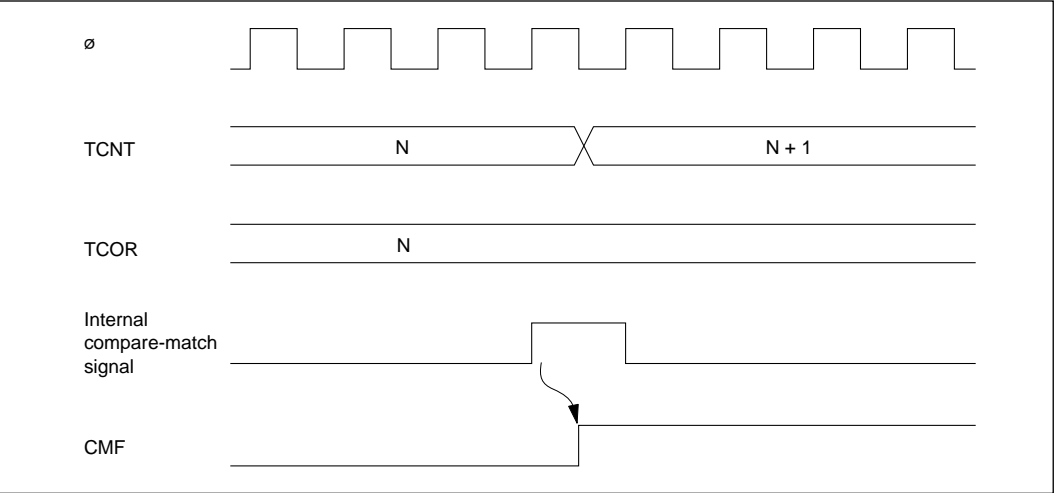


Figure 11-3 Setting of Compare-Match Flags

Output Timing: When a compare-match event occurs, the timer output (TMO) changes as specified by the output select bits (OS3 to OS0) in the TCSR. Depending on these bits, the output can remain the same, change to 0, change to 1, or toggle.

Figure 11-4 shows the timing when the output is set to toggle on compare-match A.

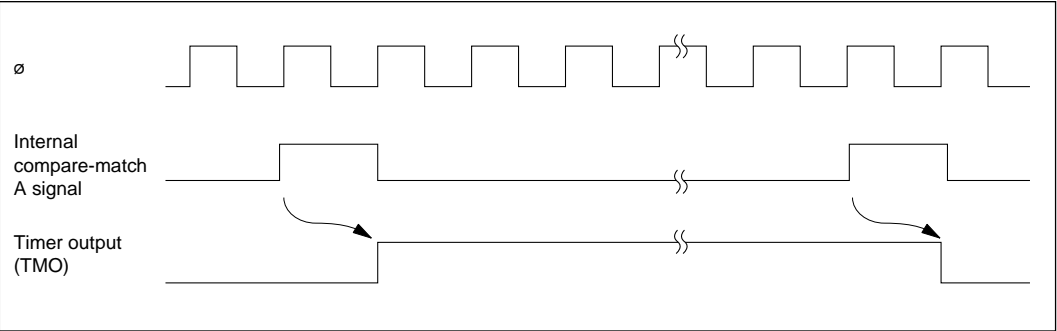


Figure 11-4 Timing of Timer Output

Timing of Compare-Match Clear: Depending on the CCLR1 and CCLR0 bits in the TCR, the timer counter can be cleared when compare-match A or B occurs. Figure 11-5 shows the timing of this operation.

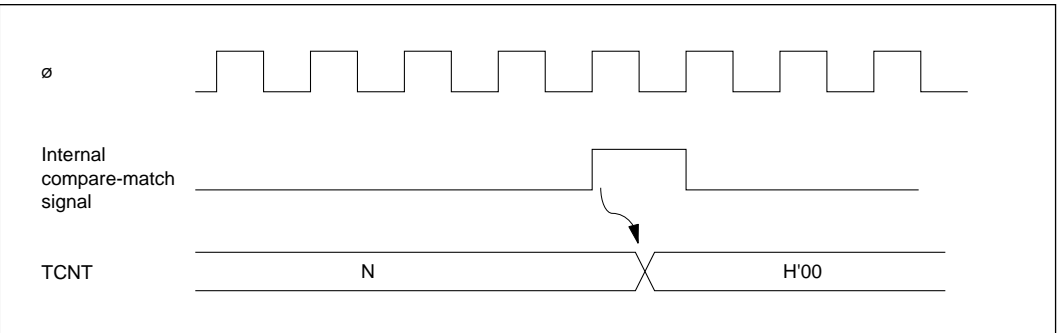


Figure 11-5 Timing of Compare-Match Clear

11.3.3 External Reset of TCNT

When the CCLR1 and CCLR0 bits in the TCR are both set to 1, the timer counter is cleared on the rising edge of an external reset input. Figure 11-6 shows the timing of this operation.

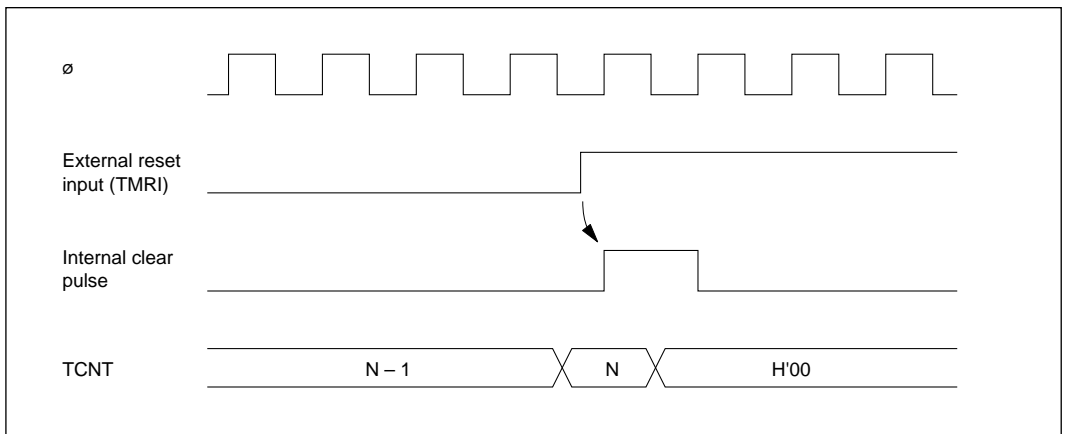


Figure 11-6 Timing of External Reset

11.3.4 Setting of TCNT Overflow Flag

The overflow flag (OVF) is set to 1 when the timer count overflows (changes from H'FF to H'00). Figure 11-7 shows the timing of this operation.

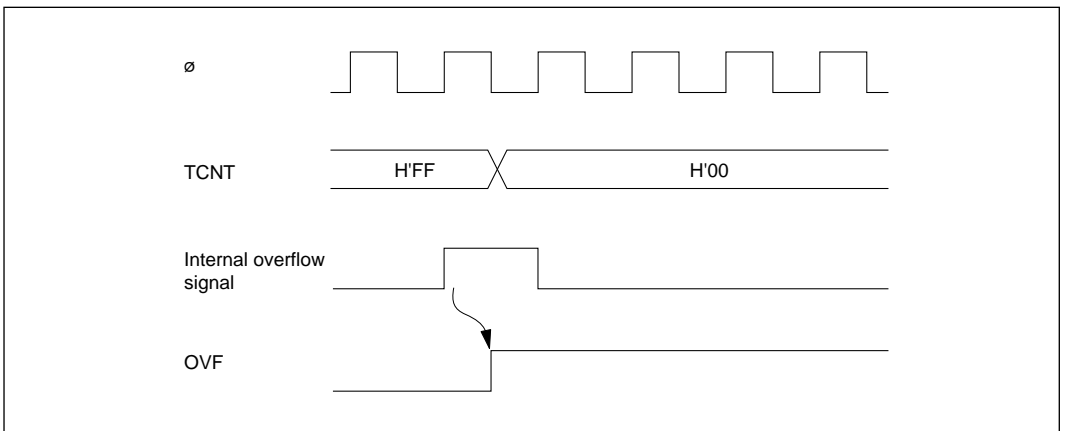


Figure 11-7 Setting of Overflow Flag (OVF)

11.4 CPU Interrupts and DTC Interrupts

The 8-bit timer can generate three types of interrupts: compare-match A and B (CMIA and CMIB), and overflow (OVI). Each interrupt is requested when the corresponding enable and flag bits are set in the TCR and TCSR. Independent signals are sent to the interrupt controller for each type of interrupt. Table 11-3 lists information about these interrupts.

Table 11-3 8-Bit Timer Interrupts

Interrupt	Description	DTC Service Available?	Priority
CMIA	Requested when CMFA is set	Yes	High
CMIB	Requested when CMFB is set	Yes	▲
OVI	Requested when OVF is set	No	Low

The CMIA and CMIB interrupts can be served by the data transfer controller (DTC) to have a data transfer performed.

When the DTC serves one of these interrupts, it automatically clears the CMFA or CMFB flag to 0. See section 6, “Data Transfer Controller,” for further information on the DTC.

11.5 Sample Application

In the example below, the 8-bit timer is used to generate a pulse output with a selected duty factor. The control bits are set as follows:

1. In the TCR, CCLR1 is cleared to 0 and CCLR0 is set to 1 so that the timer counter is cleared when its value matches the constant in TCORA.
2. In the TCSR, bits OS3 to OS0 are set to 0110, causing the output to change to 1 on compare-match A and to 0 on compare-match B.

With these settings, the 8-bit timer provides output of pulses at a rate determined by TCORA with a pulse width determined by TCORB. No software intervention is required.

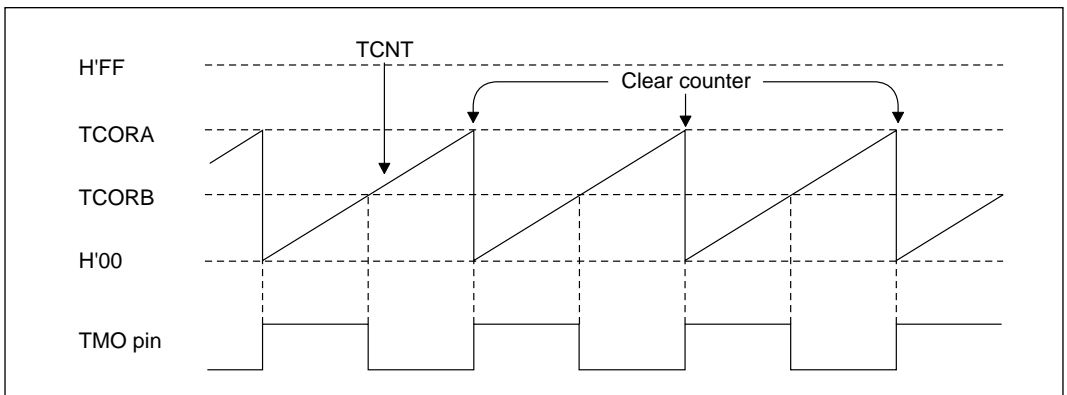


Figure 11-8 Example of Pulse Output

11.6 Application Notes

Application programmers should note that the following types of contention can occur in the 8-bit timer.

Contention between TCNT Write and Clear: If an internal counter clear signal is generated during the T3 state of a write cycle to the timer counter, the clear signal takes priority and the write is not performed.

Figure 11-9 shows this type of contention.

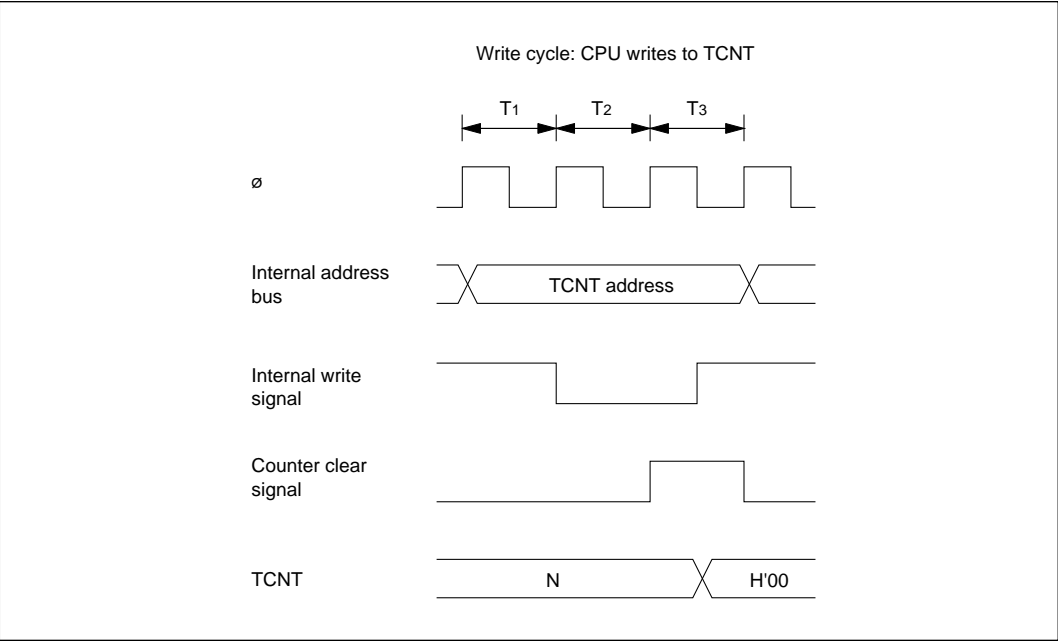


Figure 11-9 TCNT Write-Clear Contention

Contention between TCNT Write and Increment: If a timer counter increment pulse is generated during the T3 state of a write cycle to the timer counter, the write takes priority and the timer counter is not incremented.

Figure 11-10 shows this type of contention.

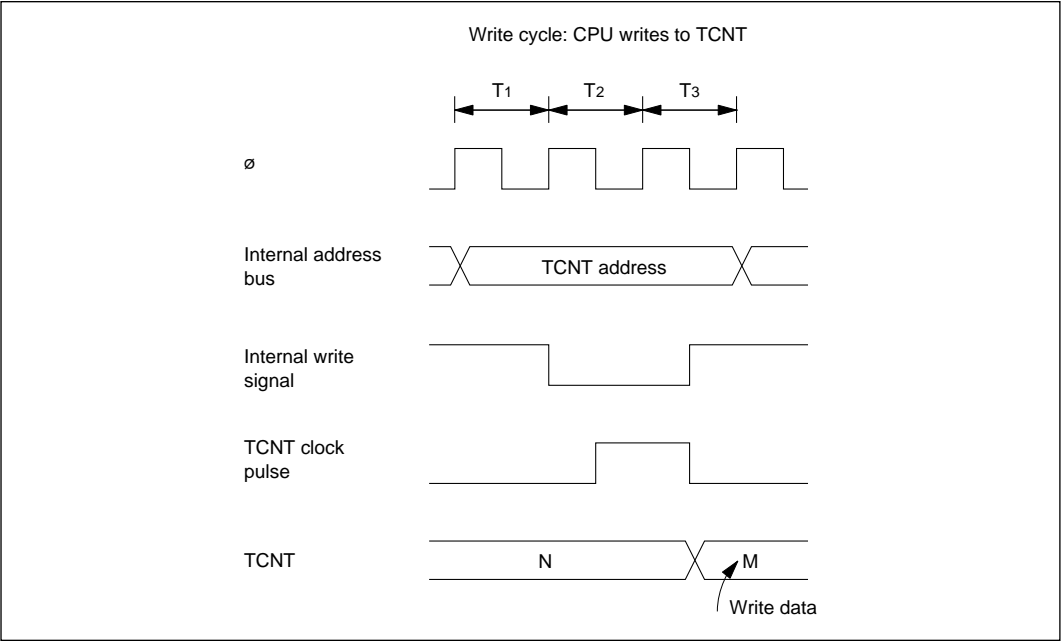


Figure 11-10 TCNT Write-Increment Contention

Contention between TCOR Write and Compare-Match: If a compare-match occurs during the T3 state of a write cycle to TCORA or TCORB, the write takes precedence and the compare-match signal is inhibited.

Figure 11-11 shows this type of contention.

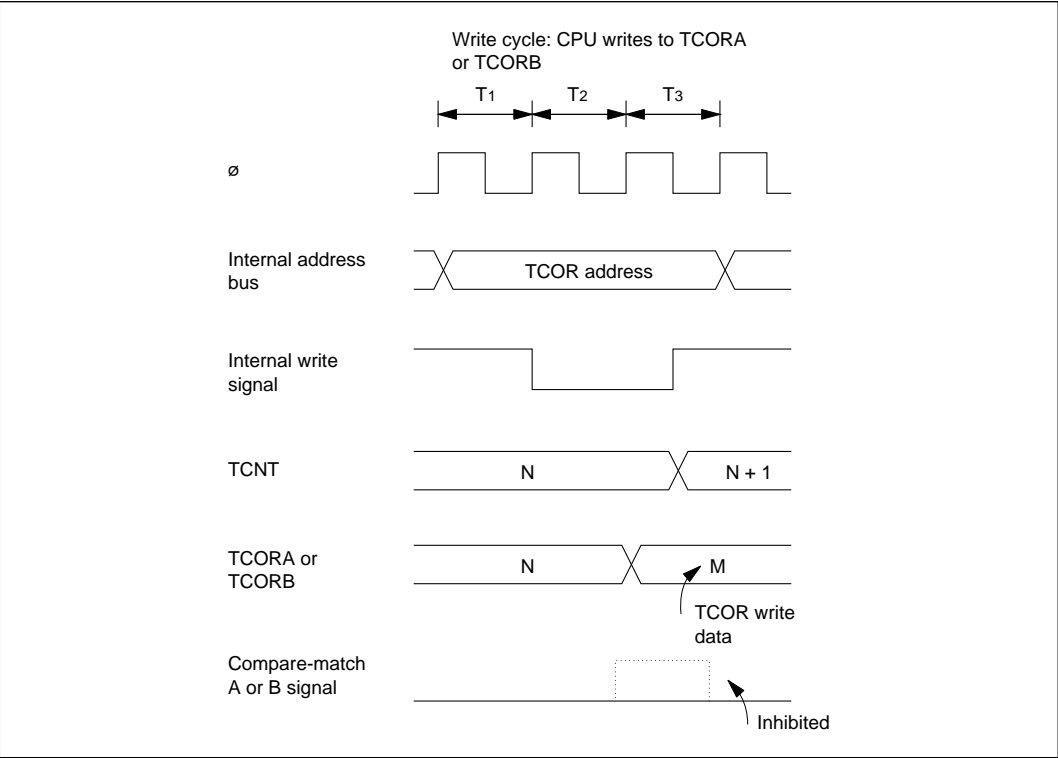


Figure 11-11 Contention between TCOR Write and Compare-Match

Contention between Compare-Match A and Compare-Match B: If identical time constants are written in TCORA and TCORB, causing compare-match A and B to occur simultaneously, any conflict between the output selections for compare-match A and B is resolved by following the priority order in table 11-4.

Table 11-4 Priority Order of Timer Output

Output Selection	Priority
Toggle	High
1 Output	↑
0 Output	↑
No change	Low

Incrementation Caused by Changing of Internal Clock Source: When an internal clock source is changed, the changeover may cause the timer counter to increment. This depends on the time at which the clock select bits (CKS2 to CKS0) are rewritten, as shown in table 11-5.

The pulse that increments the timer counter is generated at the falling edge of the internal clock source signal. If clock sources are changed when the old source is High and the new source is Low, as in case No. 3 in table 11-5, the changeover generates a falling edge that triggers the TCNT clock pulse and increments the timer counter.

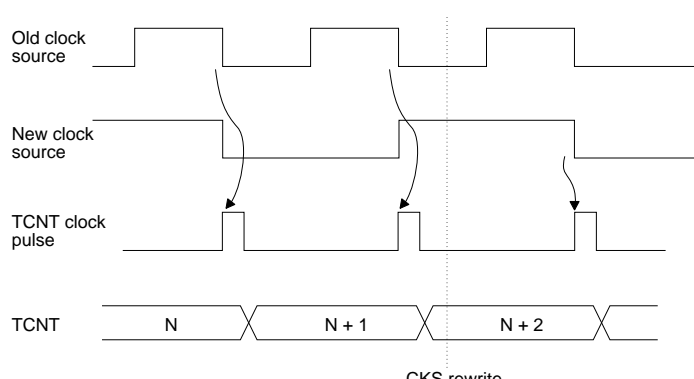
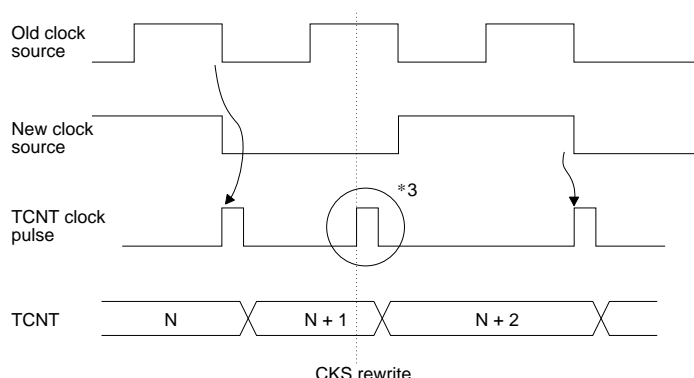
Switching between an internal and external clock source can also cause the timer counter to increment.

Table 11-5 Effect of Changing Internal Clock Sources

No.	Description	Timing Chart
1	Low → Low*1: CKS1 and CKS0 are rewritten while both clock sources are Low.	<p>The timing chart illustrates the transition from one Low clock source to another. The 'Old clock source' signal starts High and falls to Low. The 'New clock source' signal starts Low and rises to High. The 'TCNT clock pulse' signal shows a sharp spike at the falling edge of the old source and a smaller spike at the falling edge of the new source. The 'TCNT' counter value is N during the first clock cycle and N+1 during the second, with a vertical dashed line labeled 'CKS rewrite' between the two cycles.</p>

Note: *1 Including a transition from Low to the stopped state (CKS1 = 0, CKS0 = 0), or a transition from the stopped state to Low.

Table 11-5 Effect of Changing Internal Clock Sources (cont)

No.	Description	Timing Chart
2	Low → High*1: CKS1 and CKS0 are rewritten while old clock source is Low and new clock source is High.	
3	High → Low*2: CKS1 and CKS0 are rewritten while old clock source is High and new clock source is Low.	

Notes: *1 Including a transition from the stopped state to High.
*2 Including a transition from High to the stopped state.
*3 The switching of clock sources is regarded as a falling edge that increments the TCNT.

Table 11-5 Effect of Changing Internal Clock Sources (cont)

No.	Description	Timing Chart
4	High → High: CKS1 and CKS0 are rewritten while both clock sources are High.	<p>The timing chart illustrates the effect of changing internal clock sources (CKS1 and CKS0) while both are High. It shows four signals over time:</p> <ul style="list-style-type: none">Old clock source: A square wave that is High during the first two TCNT counter periods (N and N+1) and Low during the third (N+2).New clock source: A square wave that is Low during the first two TCNT counter periods (N and N+1) and High during the third (N+2).TCNT clock pulse: A series of three narrow pulses, each occurring at the rising edge of the Old clock source. The first two pulses occur during periods N and N+1, and the third pulse occurs during period N+2.TCNT: A counter that increments from N to N+1 during the first period, and from N+1 to N+2 during the second period. The counter value N+2 is shown during the third period. <p>A vertical dashed line labeled "CKS rewrite" is positioned at the rising edge of the Old clock source during the third TCNT counter period (N+2). Arrows indicate that the Old clock source is rewritten to Low and the New clock source is rewritten to High at this point.</p>

Section 12 Refresh Controller

12.1 Overview

To simplify interfacing to dynamic RAM, the H8/510 has an on-chip refresh control circuit. Insertion of refresh cycles can be inhibited in systems not using dynamic RAM.

12.1.1 Features

The refresh controller has the following features:

- Programmable refresh interval
Eight refresh intervals can be selected (from 32 to 256 states)
- 12-Bit refresh addresses
- Refresh cycle length: 2 to 5 states (selectable)
- Precharge states (TP) can be inserted

12.1.2 Block Diagram

Figure 12-1 is a block diagram of the refresh controller.

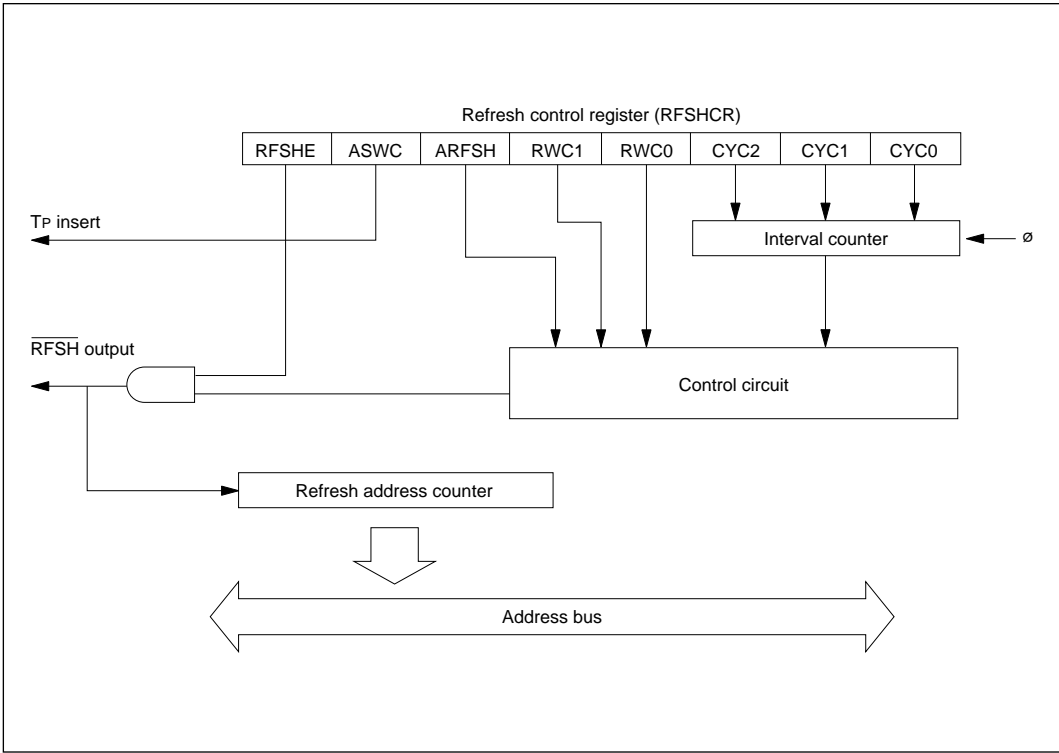


Figure 12-1 Refresh Controller Block Diagram

12.1.3 Register Configuration

The refresh controller has one control register, described in table 12-1.

Table 12-1 Refresh Control Register

Name	Abbreviation	Read/Write	Initial Value	Address
Refresh control register	RFSHCR	R/W	H'D8	H'FED8

12.2 Refresh Control Register (RFSHCR)—H'FED8

Bit	7	6	5	4	3	2	1	0
	RFSHE	ASWC	ARFSH	RWC1	RWC0	CYC2	CYC1	CYC0
Initial value	1	1	0	1	1	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The refresh control register (RFSHCR) is an 8-bit register that controls the operating modes of the refresh controller.

The refresh control register is initialized to H'D8 at a reset and in the hardware standby mode. It is not initialized in the software standby mode.

Bit 7—Refresh Enable (RFSHE): Specifies whether or not to insert refresh cycles.

Bit 7

RFSHE	Description
0	Refresh cycles are not inserted.
1	Refresh cycles are inserted. (Initial value)

Bit 6—As Wait Control (ASWC): Specifies whether or not to insert a precharge (Tp) state immediately before the T1 state of a three-state-access bus cycle.

Bit 6

ASWC	Description
0	No Tp state is inserted.
1	Tp state is inserted. (Initial value)

Bit 5—Auto-Refresh (ARFSH): Specifies whether or not to generate an auto-refresh pulse for pseudo-static RAM.

Bit 5

ARFSH	Description
0	\overline{RD} is always 1 during refresh cycles. (Initial value)
1	\overline{RD} is output as an auto-refresh pulse for pseudo-static RAM.

Bits 4 and 3—Refresh Wait Cycle (RWC1 and RWC0): Specify the number of wait states inserted in a refresh bus cycle.

Bit 4	Bit 3	Description	
RWC1	RWC0	Wait States	Refresh States
0	0	0	2
0	1	1	3
1	0	2	4
1	1	3	5

(Initial value)

Bits 2 to 0—Refresh Cycle 2 to 0 (CYC2 to CYC0):

Bit 2	Bit 1	Bit 0	Refresh Request Interval (States)	Time Interval (Examples) for Typical Frequencies of System Clock (ø)		
CYC2	CYC1	CYC0		10 MHz	8 MHz	6 MHz
0	0	0	32	3.2 µs	4.0 µs	5.3 µs
0	0	1	64	6.4 µs	8.0 µs	10.6 µs
0	1	0	96	9.6 µs	12.0 µs	16.0 µs
0	1	1	128	12.8 µs	16.0 µs	21.3 µs
1	0	0	160	16.0 µs	20.0 µs	26.6 µs
1	0	1	192	19.2 µs	24.0 µs	32.0 µs
1	1	0	224	22.4 µs	28.0 µs	37.3 µs
1	1	1	256	25.6 µs	32.0 µs	42.6 µs

(Initial value)

Dynamic RAM that requires 128 refresh cycles over a 2-ms period (or 256 refresh cycles over a 4-ms period) has a refresh interval of:

$$2 \text{ ms}/128 = 4 \text{ ms}/256 = 15.625 \text{ µs}$$

If the H8/510 is operating at 10 MHz, the refresh cycle can be set to 128 states (12.8 µs).

Refresh cycles are inserted at the ends of other bus cycles, so the actual interval between refresh cycles may differ slightly from the interval selected with CYC2 to CYC0. When wait states are inserted, the interval may also differ for the same reason.

12.3 Operation

The refresh controller sends the CPU a refresh request signal at fixed intervals. When it receives this signal, the CPU waits for the end of the current bus cycle, then executes a refresh cycle.

Figure 12-2 shows an example of the timing of a refresh cycle. During a refresh cycle the $\overline{\text{RFSH}}$ signal goes Low to identify the cycle as a refresh cycle, and a refresh address is output. The number of bits in the refresh address varies depending on the MCU mode as shown in table 12-2.

The refresh operation is not executed while the bus is released, or during wait states.

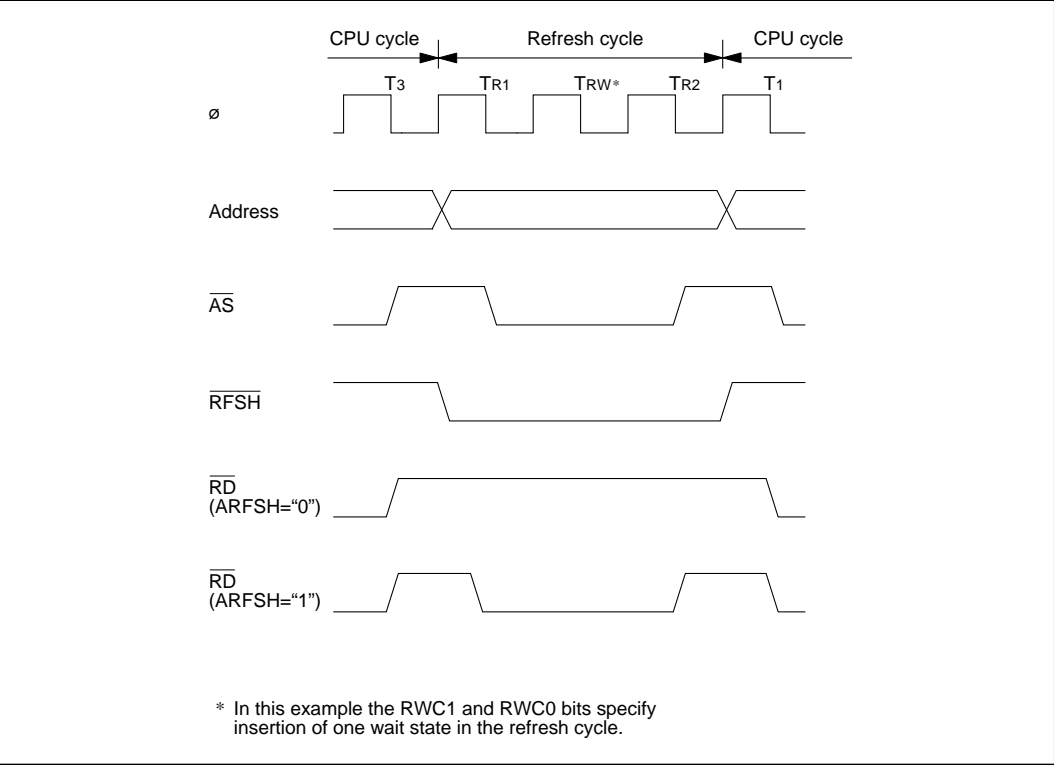


Figure 12-2 Refresh Timing

Table 12-2 MCU Modes and Refresh Addresses

MCU Mode	Refresh Address
Modes 1 and 3 (8-bit bus)	Output on A0 to A11; A12 to A23 are all 0
Modes 2 and 4 (16-bit bus)	Output on A1 to A12; A0 and A13 to A23 are all 0

If the ARFSH bit in the refresh control register (RFSHCR) is set to 1, a pseudo-static RAM auto-refresh cycle is executed. In the auto-refresh cycle, the $\overline{\text{RFSH}}$ signal sits Low while a Low pulse is output on the $\overline{\text{RD}}$ signal line. Refresh addresses are output even though they are not needed in an auto-refresh.

12.3.1 Wait State Insertion

One or more TRW states can be inserted in a refresh cycle before the TR2 state, depending on the RWC1 and RWC0 bits. TRW states can also be inserted by $\overline{\text{WAIT}}$ input. When the WMS1 bit in the wait control register (WCR) is set to 1, if one or more programmable wait states are inserted by RWC1 and RWC0, the $\overline{\text{WAIT}}$ signal is sampled on the falling edge of the state before TR2. If the $\overline{\text{WAIT}}$ signal is Low at this time, a TRW state is inserted. The $\overline{\text{WAIT}}$ signal is sampled again on the falling edge of each TRW state. Figure 12-3 shows the timing.

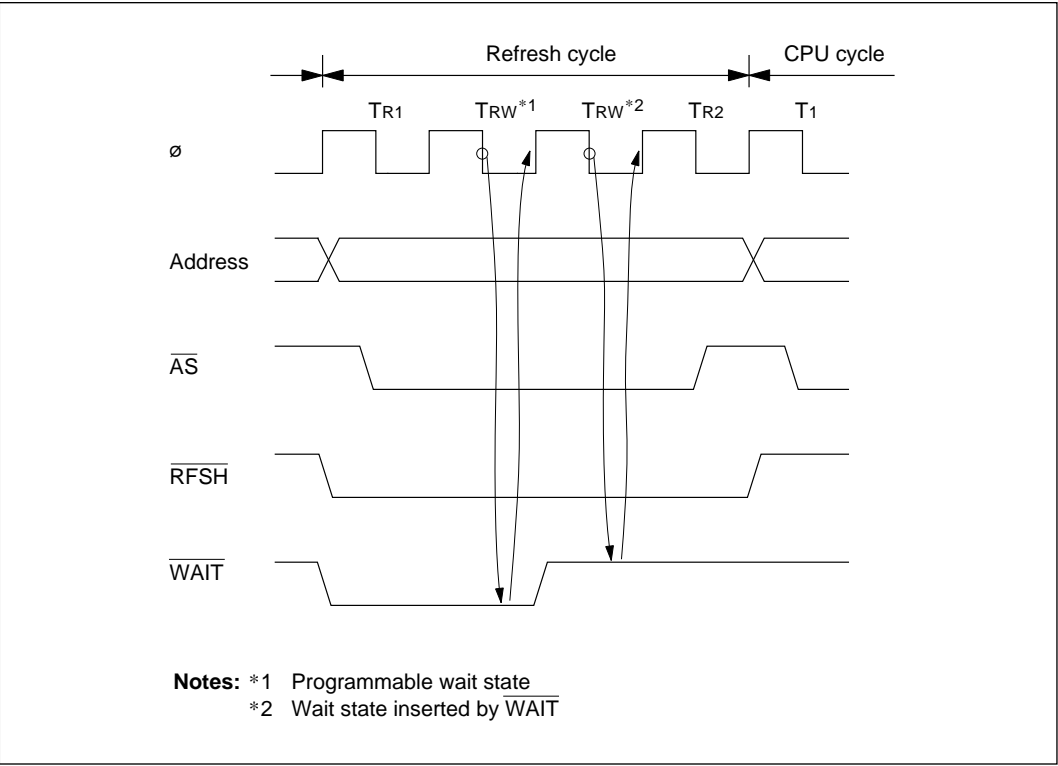


Figure 12-3 Insertion of Wait State by $\overline{\text{WAIT}}$

12.3.2 TP Insertion

A TP state can be inserted to satisfy the $\overline{\text{RAS}}$ precharge requirements of dynamic RAM.

When the ASWC bit in the refresh control register (RFSHCR) is set to 1, a TP state is inserted before the T1 state in CPU bus cycles and before the TR1 state in refresh bus cycles.

Figure 12-4 shows the insertion of a TP state in a refresh bus cycle.

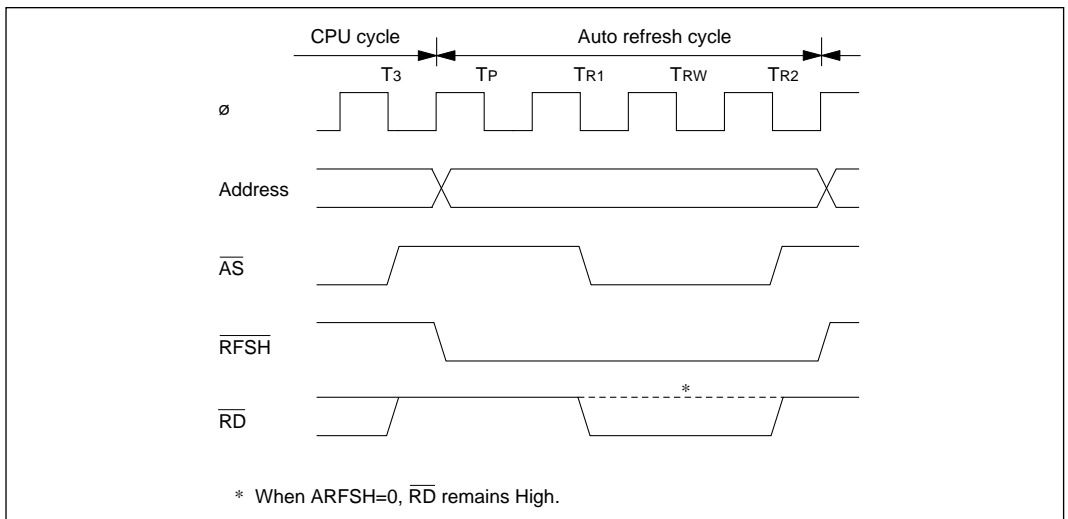


Figure 12-4 Refresh Timing when TP States are Inserted (Three-State Refresh)

TP state can be inserted in three-state-access bus cycles and in refresh cycles with three states or more. They cannot be inserted in two-state-access bus cycles and two-state refresh cycles.

Figure 12-5 shows the insertion of a TP state in a CPU bus cycle.

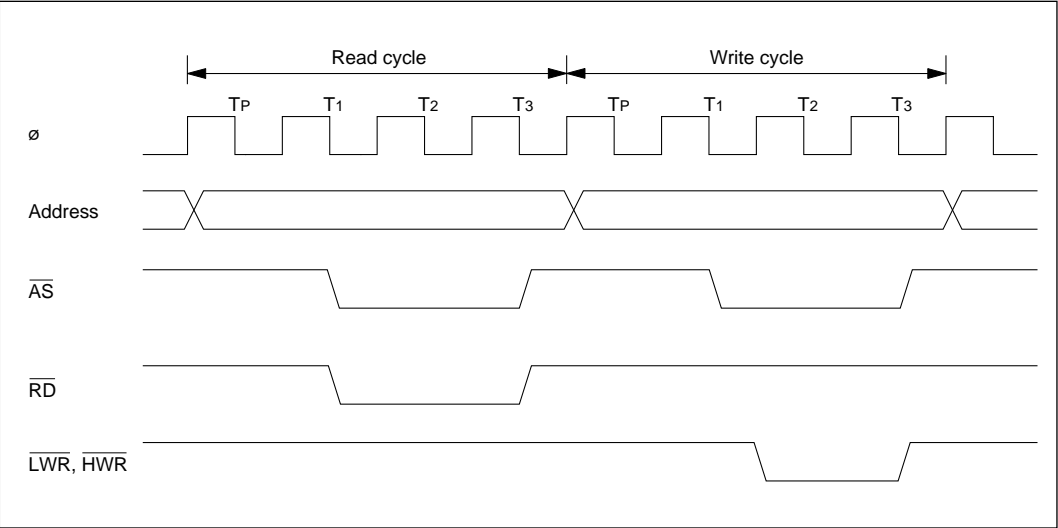


Figure 12-5 CUP Bus Cycle Timing when TP States are Inserted

12.4 Operation in Power-Down State

The refresh controller continues to operate in sleep mode.

The refresh controller halts in the standby modes. In software standby mode the refresh control register (RFSHCR) is not initialized; it retains the values set before the standby began. If the chip recovers from software standby mode by an NMI interrupt, however, the refresh address is modified unpredictably.

12.5 Operation in Reset State

The refresh controller halts during the reset state. The refresh control register (RFSHCR) is initialized to H'D8 (enabling refresh operations). The refresh address is initialized to H'000.

12.6 Application Notes

The following points require attention when the refresh controller is used.

1. Refresh cycles are not executed when the CPU released the bus, in the software standby mode, in the hardware standby mode, and during wait states. If any of these conditions continues for a long time, memory must be refreshed by other means.
2. If refresh requests are generated internally while the bus is released, one request is held pending, causing one refresh cycle to be executed after the CPU regains control of the bus. Figure 12-6 shows an example of bus cycles in this case.
3. If a refresh request is generated internally during a wait state, the request is held until the next refresh request is generated, and a refresh cycle is executed at the first opportunity after the wait state is released.
4. If refresh cycles are not executed for a long time because the chip is in the bus-released state or a long wait state, when this state ends the refresh address output in the next refresh cycle is still the next refresh address after the preceding refresh address.

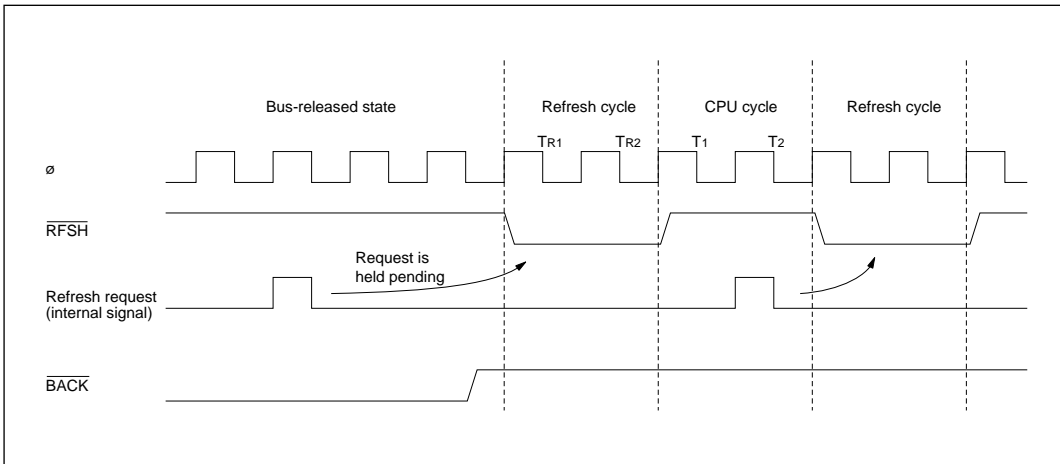


Figure 12-6 Refresh Request Generated while Bus is Released

Section 13 Serial Communication Interface

13.1 Overview

The H8/510 chip includes two serial communication interface channels (SCI1 and SCI2) for transferring serial data to and from other chips. Both channels are identical. Each channel supports both synchronous and asynchronous data transfer. Communication control functions are provided by eight internal registers.

13.1.1 Features

The features of the on-chip serial communication interface channels are:

- Selection of asynchronous or synchronous mode
 - Asynchronous mode

The H8/510 can communicate with a UART (Universal Asynchronous Receiver/Transmitter), ACIA (Asynchronous Communication Interface Adapter), or other chip that employs standard asynchronous serial communication. Eight data formats are available.

 - Data length: 7 or 8 bits
 - Stop bit length: 1 or 2 bits
 - Parity: Even, odd, or none
 - Error detection: Parity, overrun, and framing errors
 - Synchronous mode

The H8/510 can communicate with chips able to synchronize data transfers with clock pulses.

 - Data length: 8 bits
 - Error detection: Overrun errors
- Full duplex communication

The transmitting and receiving sections are independent, so each SCI can transmit and receive simultaneously. Both the transmit and receive sections use double buffering, so continuous data transfer is possible in either direction.
- Built-in baud rate generator

Any specified bit rate can be generated.
- Internal or external clock source

The baud rate generator can operate on an internal clock source, or an external clock signal input at the SCK pin.
- Three interrupts

Transmit-end, receive-end, and receive-error interrupts are requested independently. The transmit-end and receive-end interrupts can be served by the on-chip data transfer controller (DTC), providing a convenient way to transfer data with minimal CPU programming.

13.1.2 Block Diagram

Figure 13-1 shows a block diagram of the serial communication interface for one channel.

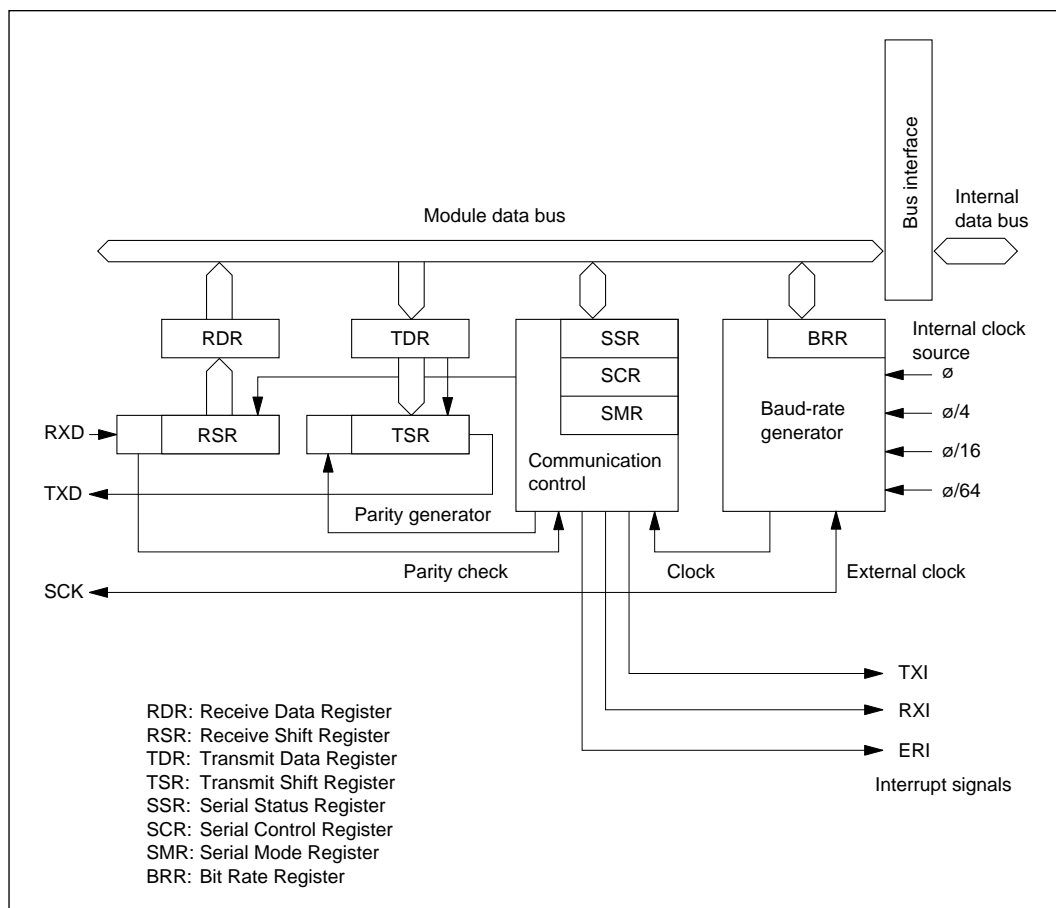


Figure 13-1 Block Diagram of Serial Communication Interface

13.1.3 Input and Output Pins

Table 13-1 lists the input and output pins used by each SCI channel.

Table 13-1 SCI Input/Output Pins

Name	Abbreviation	I/O	Function
Serial clock	SCK	Input/output	Serial clock input and output
Receive data	RXD	Input	Receive data input
Transmit data	TXD	Output	Transmit data output

13.1.4 Register Configuration

Table 13-2 lists the SCI registers.

Table 13-2 SCI Registers

Channel	Name	Abbreviation	R/W	Initial Value	Address
1	Receive shift register	RSR	—	—	—
	Receive data register	RDR	R	H'00	H'FECD
	Transmit shift register	TSR	—	—	—
	Transmit data register	TDR	R/W	H'FF	H'FECB
	Serial mode register	SMR	R/W	H'04	H'FEC8
	Serial control register	SCR	R/W	H'0C	H'FECA
	Serial status register	SSR	R/(W)*	H'87	H'FECC
	Bit rate register	BRR	R/W	H'FF	H'FEC9
2	Receive shift register	RSR	—	—	—
	Receive data register	RDR	R	H'00	H'FED5
	Transmit shift register	TSR	—	—	—
	Transmit data register	TDR	R/W	H'FF	H'FED3
	Serial mode register	SMR	R/W	H'04	H'FED0
	Serial control register	SCR	R/W	H'0C	H'FED2
	Serial status register	SSR	R/(W)*	H'87	H'FED4
	Bit rate register	BRR	R/W	H'FF	H'FED1

* Software can write a 0 to clear the status flag bits, but cannot write a 1.

13.2 Register Descriptions

13.2.1 Receive Shift Register (RSR)

Bit	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—

The RSR receives incoming data bits. When one data character has been received, it is transferred to the receive data register (RDR).

The CPU cannot read or write the RSR directly.

13.2.2 Receive Data Register (RDR)—H'FECD and H'FED5

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

The RDR stores received data. As each character is received, it is transferred from the RSR to the RDR, enabling the RSR to receive the next character. This double-buffering allows the SCI to receive data continuously.

The CPU can read but not write the RDR. The RDR is initialized to H'00 at a reset and in the standby modes.

13.2.3 Transmit Shift Register (TSR)

Bit	7	6	5	4	3	2	1	0
Read/Write	—	—	—	—	—	—	—	—

The TSR holds the character currently being transmitted. When transmission of this character is completed, the next character is moved from the transmit data register (TDR) to the TSR and transmission of that character begins. If the TDR does not contain valid data, the SCI stops transmitting.

The CPU cannot read or write the TSR directly.

13.2.4 Transmit Data Register (TDR)—H'FECB and H'FED3

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The TDR is an 8-bit readable/writable register that holds the next character to be transmitted. When the TSR becomes empty, the character written in the TDR is transferred to the TSR.

Continuous data transmission is possible by writing the next byte in the TDR while the current byte is being transmitted from the TSR.

The TDR is initialized to H'FF at a reset and in the standby modes.

13.2.5 Serial Mode Register (SMR)—H'FEC8 and H'FED0

Bit	7	6	5	4	3	2	1	0
	C/ \bar{A}	CHR	PE	O/ \bar{E}	STOP	—	CKS1	CKS0
Initial value	0	0	0	0	0	1	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	—	R/W	R/W

The SMR is an 8-bit readable/writable register that controls the communication format and selects the clock rate for the internal clock source. It is initialized to H'04 at a reset and in the standby modes.

Bit 7—Communication Mode (C/ \bar{A}): This bit selects the asynchronous or synchronous communication mode.

Bit 7

C/ \bar{A}	Description
0	Asynchronous communication. (Initial value)
1	Communication is synchronized with the serial clock.

Bit 6—Character Length (CHR): This bit selects the character length in asynchronous mode. It is ignored in synchronous mode.

Bit 6

CHR	Description
0	8 Bits per character. (Initial value)
1	7 Bits per character.

Bit 5—Parity Enable (PE): This bit selects whether to add a parity bit in asynchronous mode. It is ignored in synchronous mode.

Bit 5

PE	Description
0	Transmit: No parity bit is added. (Initial value) Receive: Parity is not checked.
1	Transmit: A parity bit is added. Receive: Parity is checked.

Bit 4—Parity Mode (O/E): In asynchronous mode, when parity is enabled (PE = 1), this bit selects even or odd parity.

Even parity means that a parity bit is added to the data bits for each character to make the total number of 1's even. Odd parity means that the total number of 1's is made odd.

This bit is ignored when PE = 0 and in the synchronous mode.

Bit 4

O/E	Description
0	Even parity. (Initial value)
1	Odd parity.

Bit 3—Stop Bit Length (STOP): This bit selects the number of stop bits. It is ignored in the synchronous mode.

Bit 3

STOP	Description
0	1 Stop bit. (Initial value)
1	2 Stop bits.

Bit 2—Reserved: This bit cannot be modified and is always read as 1.

Bits 1 and 0—Clock Select 1 and 0 (CKS1 and CKS0): These bits select the internal clock source when the baud rate generator is clocked from within the H8/510 chip.

Bit 1 CKS1	Bit 0 CKS0	Description
0	0	∅ clock (Initial value)
0	1	∅/4 clock
1	0	∅/16 clock
1	1	∅/64 clock

13.2.6 Serial Control Register (SCR)—H'FECA and H'FED2

Bit	7	6	5	4	3	2	1	0
	TIE	RIE	TE	RE	—	—	CKE1	CKE0
Initial value	0	0	0	0	1	1	0	0
Read/Write	R/W	R/W	R/W	R/W	—	—	R/W	R/W

The SCR is an 8-bit readable/writable register that enables or disables various SCI functions. It is initialized to H'0C at a reset and in the standby modes.

Bit 7—Transmit Interrupt Enable (TIE): This bit enables or disables the transmit-end interrupt (TXI) requested when the transmit data register empty (TDRE) bit in the serial status register (SSR) is set to 1.

Bit 7 TIE	Description
0	The transmit-end interrupt request (TXI) is disabled. (Initial value)
1	The transmit-end interrupt request (TXI) is enabled.

Bit 6—Receive Interrupt Enable (RIE): This bit enables or disables the receive-end interrupt (RXI) requested when the receive data register full (RDRF) bit in the serial status register (SSR) is set to 1. It also enables and disables the receive-error interrupt (ERI) request.

Bit 6 RIE	Description
0	The receive-end interrupt (RXI) and receive-error interrupt (ERI) requests are disabled. (Initial value)
1	The receive-end interrupt (RXI) and receive-error interrupt (ERI) requests are enabled.

Bit 5—Transmit Enable (TE): This bit enables or disables the transmit function. When the transmit function is enabled, the TXD pin is automatically used for output. When the transmit function is disabled, the TXD pin can be used as a general-purpose I/O port.

Bit 5

TE	Description
0	The transmit function is disabled. The TXD pin can be used as a general-purpose I/O port. (Initial value)
1	The transmit function is enabled. The TXD pin is used for output.

Bit 4—Receive Enable (RE): This bit enables or disables the receive function. When the receive function is enabled, the RXD pin is automatically used for input. When the receive function is disabled, the RXD pin is available as a general-purpose I/O port.

Bit 4

RE	Description
0	The receive function is disabled. The RXD pin can be used as a general-purpose I/O port. (Initial value)
1	The receive function is enabled. The RXD pin is used for input.

Bits 3 and 2—Reserved: These bits cannot be modified and are always read as 1.

Bit 1—Clock Enable 1 (CKE1): This bit selects the internal or external clock source for the baud rate generator. When the external clock source is selected, the SCK pin is automatically used for input of the external clock signal.

Bit 1

CKE1	Description
0	Internal clock source. (Initial value)
1	External clock source. (The SCK pin is used for input.)

Bit 0—Clock Enable 0 (CKE0): When an internal clock source is used in synchronous mode, this bit enables or disables serial clock output at the SCK pin.

This bit is ignored when the external clock is selected, or when the asynchronous mode is selected.

For further information on the communication format and clock source selection, see tables 13-5 and 13-6 in section 13.3, “Operation.”

Bit 0**CKE0 Description**

0	The SCK pin is not used by the SCI (and is available as a general-purpose I/O port).	(Initial value)
1	The SCK pin is used for serial clock output.	

13.2.7 Serial Status Register (SSR)—H'FECC and H'FED4

Bit	7	6	5	4	3	2	1	0
	TDRE	RDRF	ORER	FER	PER	—	—	—
Initial value	1	0	0	0	0	1	1	1
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	—	—	—

* Software can write a 0 to clear the flags, but cannot write a 1 in these bits.

The SSR is an 8-bit register that indicates transmit and receive status. It is initialized to H'87 at a reset and in the standby modes.

Bit 7—Transmit Data Register Empty (TDRE): This bit indicates when the TDR contents have been transferred to the TSR and the next character can safely be written in the TDR.

Bit 7**TDRE Description**

0	This bit is cleared from 1 to 0 when: 1. The CPU reads the TDRE bit after it has been set to 1, then writes a 0 in this bit. 2. The data transfer controller (DTC) writes data in the TDR.	
1	This bit is set to 1 at the following times: 1. The chip is reset or enters a standby mode. 2. When TDR contents are transferred to the TSR. 3. When TDRE = 0 and the TE bit is cleared to 0.	(Initial value)

Bit 6—Receive Data Register Full (RDRF): This bit indicates when one character has been received and transferred to the RDR.

Bit 6

RDRF	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the RDRF bit after it has been set to 1, then writes a 0 in this bit. 2. The data transfer controller (DTC) reads the RDR. 3. The chip is reset or enters a standby mode.
1	This bit is set to 1 when one character is received without error and transferred from the RSR to the RDR.

Bit 5—Overrun Error (ORER): This bit indicates an overrun error during reception.

Bit 5

ORER	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the ORER bit after it has been set to 1, then writes a 0 in this bit. 2. The chip is reset or enters a standby mode.
1	This bit is set to 1 if reception of the next character ends while the receive data register is still full (RDRF = 1).

Bit 4—Framing Error (FER): This bit indicates a framing error during data reception in the synchronous mode. It has no meaning in the asynchronous mode.

Bit 4

FER	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the FER bit after it has been set to 1, then writes a 0 in this bit. 2. The chip is reset or enters a standby mode.
1	This bit is set to 1 if a framing error occurs (stop bit = 0).

Bit 3—Parity Error (PER): This bit indicates a parity error during data reception in the asynchronous mode, when a communication format with parity bits is used.

This bit has no meaning in the synchronous mode, or when a communication format without parity bits is used.

Bit 3

PER	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The CPU reads the PER bit after it has been set to 1, then writes a 0 in this bit. 2. The chip is reset or enters a standby mode.
1	This bit is set to 1 when a parity error occurs (the parity of the received data does not match the parity selected by the bit in the SMR).

Bits 2 to 0—Reserved: These bits cannot be modified and are always read as 1.

13.2.8 Bit Rate Register (BRR)—H'FEC9 and H'FED1

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The BRR is an 8-bit register that, together with the CKS1 and CKS0 bits in the SMR, determines the bit rate output by the baud rate generator.

The BRR is initialized to H'FF (the slowest rate) at a reset and in the standby modes.

Tables 13-3 and 13-4 show examples of BRR (N) and CKS (n) settings for commonly used bit rates.

Table 13-3 Examples of BRR Settings in Asynchronous Mode (1)

Bit Rate	XTAL Frequency (MHz)											
	2			2.4576			4			4.194304		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	1	70	+0.03	1	86	+0.31	1	141	+0.03	1	148	−0.04
150	0	207	+0.16	0	255	0	1	103	+0.16	1	108	+0.21
300	0	103	+0.16	0	127	0	0	207	+0.16	0	217	+0.21
600	0	51	+0.16	0	63	0	0	103	+0.16	0	108	+0.21
1200	0	25	+0.16	0	31	0	0	51	+0.16	0	54	−0.70
2400	0	12	+0.16	0	15	0	0	25	+0.16	0	26	+1.14
4800	—	—	—	0	7	0	0	12	+0.16	0	13	−2.48
9600	—	—	—	0	3	0	—	—	—	—	—	—
19200	—	—	—	0	1	0	—	—	—	—	—	—
31250	—	—	—	—	—	—	0	1	0	—	—	—
38400	—	—	—	0	0	0	—	—	—	—	—	—

Table 13-3 Examples of BRR Settings in Asynchronous Mode (2)

Bit Rate	XTAL Frequency (MHz)											
	4.9152			6			7.3728			8		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	1	174	−0.26	2	52	+0.50	2	64	+0.70	2	70	+0.03
150	1	127	0	1	155	+0.16	1	191	0	1	207	+0.16
300	0	255	0	1	77	+0.16	1	95	0	1	103	+0.16
600	0	127	0	0	155	+0.16	0	191	0	0	207	+0.16
1200	0	63	0	0	77	+0.16	0	95	0	0	103	+0.16
2400	0	31	0	0	38	+0.16	0	47	0	0	51	+0.16
4800	0	15	0	0	19	−2.34	0	23	0	0	25	+0.16
9600	0	7	0	—	—	—	0	11	0	0	12	+0.16
19200	0	3	0	—	—	—	0	5	0	—	—	—
31250	—	—	—	0	2	0	—	—	—	0	3	0
38400	0	1	0	—	—	—	0	2	0	—	—	—

Table 13-3 Examples of BRR Settings in Asynchronous Mode (3)

Bit Rate	XTAL Frequency (MHz)											
	9.8304			10			12			12.288		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	86	+0.31	2	88	−0.25	2	106	−0.44	2	108	+0.08
150	1	255	0	2	64	+0.16	2	77	0	2	79	0
300	1	127	0	1	129	+0.16	1	155	0	1	159	0
600	0	255	0	1	64	+0.16	1	77	0	1	79	0
1200	0	127	0	0	129	+0.16	0	155	+0.16	0	159	0
2400	0	63	0	0	64	+0.16	0	77	+0.16	0	79	0
4800	0	31	0	0	32	−1.36	0	38	+0.16	0	39	0
9600	0	15	0	0	15	+1.73	0	19	−2.34	0	19	0
19200	0	7	0	0	7	+1.73	—	—	—	0	9	0
31250	0	4	−1.70	0	4	0	0	5	0	0	5	+2.40
38400	0	3	0	0	3	+1.73	—	—	—	0	4	0

Table 13-3 Examples of BRR Settings in Asynchronous Mode (4)

Bit Rate	XTAL Frequency (MHz)											
	14.7456			16			19.6608			20		
	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)	n	N	Error (%)
110	2	130	-0.07	2	141	+0.03	2	174	-0.26	3	43	+0.88
150	2	95	0	2	103	+0.16	2	127	0	2	129	+0.16
300	1	191	0	1	207	+0.16	1	255	0	2	64	+0.16
600	1	95	0	1	103	+0.16	1	127	0	1	129	+0.16
1200	0	191	0	0	207	+0.16	0	255	0	1	64	+0.16
2400	0	95	0	0	103	+0.16	0	127	0	0	129	+0.16
4800	0	47	0	0	51	+0.16	0	63	0	0	64	+0.16
9600	0	23	0	0	25	+0.16	0	31	0	0	32	-1.36
19200	0	11	0	0	12	+0.16	0	15	0	0	15	+1.73
31250	—	—	—	0	7	0	0	9	-1.70	0	9	0
38400	0	5	0	—	—	—	0	7	0	0	7	+1.73

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

B : Bit rate

N : BRR value ($0 \leq N \leq 255$)

OSC : Crystal oscillator frequency in MHz

n : Internal clock source (0, 1, 2, or 3)

The meaning of n is given by the table below:

n	CKS1	CKS0	Clock
0	0	0	ϕ
1	0	1	$\phi/4$
2	1	0	$\phi/16$
3	1	1	$\phi/64$

Table 13-4 Examples of BRR Settings in Synchronous Mode

Bit Rate	XTAL Frequency (MHz)											
	2		4		8		10		16		20	
	n	N	n	N	n	N	n	N	n	N	n	N
100	—	—	—	—	—	—	—	—	—	—	—	—
250	1	249	2	124	2	249	—	—	3	124	—	—
500	1	124	1	249	2	124	—	—	2	249	—	—
1k	0	249	1	124	1	249	—	—	2	124	—	—
2.5k	0	99	0	199	1	99	1	124	1	199	1	249
5k	0	49	0	99	0	199	0	249	1	99	1	124
10k	0	24	0	49	0	99	0	124	0	199	0	249
25k	0	9	0	19	0	39	0	49	0	79	0	99
50k	0	4	0	9	0	19	0	24	0	39	0	49
100k	—	—	0	4	0	9	—	—	0	19	0	24
250k	0	0	0	1	0	3	0	4	0	7	0	9
500k			0	0	0	1	—	—	0	3	0	4
1M					0	0	—	—	0	1	—	—
2.5M											0	0

Notes:

Blank: No setting is available.

—: A setting is available, but the bit rate is inaccurate.

$$B = OSC / [8 \times 2^{2n} \times (N + 1)]$$

B : Bit rate

N : BRR value ($0 \leq N \leq 255$)

OSC : Crystal oscillator frequency in MHz

n : Internal clock source (0, 1, 2, or 3)

The meaning of n is given by the table below:

n	CKS1	CKS0	Clock
0	0	0	Ø
1	0	1	Ø/4
2	1	0	Ø/16
3	1	1	Ø/64

13.3 Operation

13.3.1 Overview

The SCI supports serial data transfer in both asynchronous and synchronous modes.

The communication format depends on settings in the SMR as indicated in table 13-5. The clock source and usage of the SCK pin depend on settings in the SMR and SCR as indicated in table 13-6.

Table 13-5 Communication Formats Used by SCI

SMR				Mode	Format	Parity	Stop Bit
C/A	CHR	PE	STOP				Length
0	0	0	0	Asynchronous	8-Bit data	None	1
			1				2
		1	0			Yes	1
			1				2
	1	0	0		7-Bit data	None	1
			1				2
		1	0			Yes	1
			1				2
1	—	—	—	Synchronous	8-Bit data	—	—

Table 13-6 SCI Clock Source Selection

SMR	SCR		Clock	
C/A	CKE1	CKE0	Source	SCK Pin
0 (Async mode)	0	0	Internal	I/O port*
		1		Clock output at same frequency as baud rate
	1	0	External	Clock input at 16 times the baud rate frequency
		1		
1 (Sync mode)	0	0	Internal	Serial clock output
		1		
	1	0	External	Serial clock input
		1		

* Cannot be used by the SCI.

Transmitting and receiving operations in the two modes are described next.

13.3.2 Asynchronous Mode

In asynchronous mode, each character is individually synchronized by framing it with a start bit and stop bit.

Full duplex data transfer is possible because the SCI has independent transmit and receive sections. Double buffering in both sections enables the SCI to be programmed for continuous data transfer.

Figure 13-2 shows the general format of one character sent or received in the asynchronous mode. The communication channel is normally held in the mark state (High). Character transmission or reception starts with a transition to the space state (Low).

The first bit transmitted or received is the start bit (Low). It is followed by the data bits, in which the least significant bit (LSB) comes first. The data bits are followed by the parity bit, if present, then the stop bit or bits (High) confirming the end of the frame.

In receiving, the SCI synchronizes on the falling edge of the start bit, and samples each bit at the center of bit (at the 8th cycle of the internal serial clock, which runs at 16 times the bit rate).

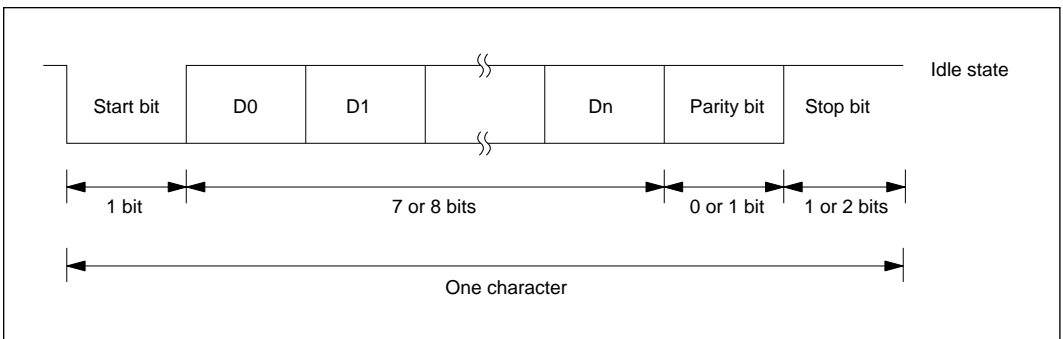


Figure 13-2 Data Format in Asynchronous Mode

1. Data Format

Table 13-7 lists the data formats that can be sent and received in asynchronous mode. Eight formats can be selected by bits in the SMR.

Table 13-7 Data Formats in Asynchronous Mode

SMR Bits									
CHR	PE	STOP	Data Format						
0	0	0	START	8-Bit data		STOP			
0	0	1	START	8-Bit data		STOP	STOP		
0	1	0	START	8-Bit data		P	STOP		
0	1	1	START	8-Bit data		P	STOP	STOP	
1	0	0	START	7-Bit data	STOP				
1	0	1	START	7-Bit data	STOP	STOP			
1	1	0	START	7-Bit data	P	STOP			
1	1	1	START	7-Bit data	P	STOP	STOP		

Note:

START: Start bit

STOP: Stop bit

P: Parity bit

2. Clock

In the asynchronous mode it is possible to select either an internal clock created by the on-chip baud rate generator, or an external clock input at the SCK pin. Refer to table 13-6.

If an external clock is input at the SCK pin, its frequency should be 16 times the desired baud rate.

If the internal clock provided by the on-chip baud rate generator is selected and the SCK pin is used for clock output, the output clock frequency is equal to the baud rate, and the clock pulse rises at the center of the transmit data bits. Figure 13-3 shows the phase relationship between the output clock and transmit data.

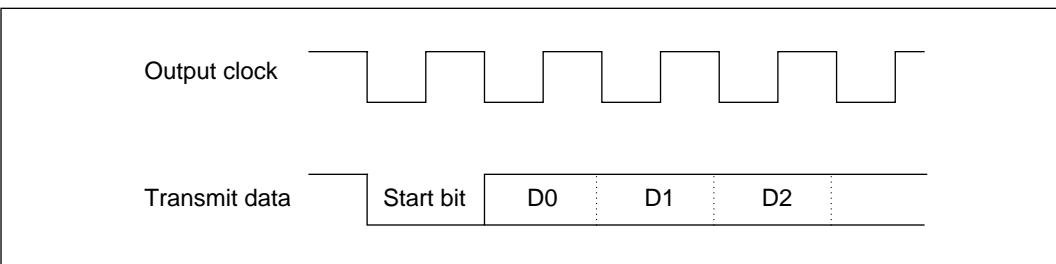


Figure 13-3 Phase Relationship between Clock Output and Transmit Data

3. Data Transmission and Reception

SCI Initialization: Before data can be transmitted or received, the SCI must be initialized by software. To initialize the SCI, software must clear the TE and RE bits to 0, then execute the following procedure.

1. Set the desired communication format in the SMR.
2. Write the value corresponding to the desired bit rate in the BRR. (This step is not necessary if an external clock is used.)
3. Select the clock and enable desired interrupts in the SCR.
4. Set the TE and/or RE bit in the SCR to 1.

The TE and RE bits must both be cleared to 0 whenever the operating mode or data format is changed.

After changing the operating mode or data format, before setting the TE and RE bits to 1 software must wait for at least the transfer time for 1 bit at the selected baud rate, to make sure the SCI is initialized. If an external clock is used, the clock must not be stopped.

When clearing the TDRE bit during data transmission, to assure transfer of the correct data, do not clear the TDRE bit until after writing data in the TDR. Similarly, in receiving data, do not clear the RDRF bit until after reading data from the RDR.

Data Transmission: The procedure for transmitting data is as follows.

1. Set up the desired transmitting conditions in the SMR, SCR, and BRR.
2. Set the TE bit in the SCR to 1.
The TXD pin will automatically be switched to output and one frame* of all 1's will be transmitted, after which the SCI is ready to transmit data.
3. Check that the TDRE bit is set to 1, then write the first byte of transmit data in the TDR. Next clear the TDRE bit to 0.
4. The first byte of transmit data is transferred from the TDR to the TSR and sent in the designated format as follows.
 - i) Start bit (one 0 bit)
 - ii) Transmit data (seven or eight bits, starting from bit 0)
 - iii) Parity bit (odd or even parity bit, or no parity bit)
 - iv) Stop bit (one or two consecutive 1 bits)
5. Transfer of the transmit data from the TDR to the TSR makes the TDR empty, so the TDRE bit is set to 1.

If the TIE bit is set to 1, a transmit-end interrupt (TXI) is requested.

When the transmit function is enabled but the TDR is empty (TDRE = 1), the output at the TXD pin is held at 1 until the TDRE bit is cleared to 0.

* A frame is the data for one character, including the start bit and stop bit(s).

Data Reception: The procedure for receiving data is as follows.

1. Set up the desired receiving conditions in the SMR, SCR, and BRR.
2. Set the RE bit in the SCR to 1.
The RXD pin will automatically be switched to input and the SCI is ready to receive data.
3. The SCI synchronizes with the incoming data by detecting the start bit, and places the received bits in the RSR. At the end of the data, the SCI checks that the stop bit is 1.
If the stop bit length is 2 bits, the SCI checks that both bits are 1.
4. When a complete frame has been received, the SCI transfers the received data to the RDR so that it can be read. If the character length is 7 bits, the most significant bit of the RDR is cleared to 0. At the same time, the SCI sets the RDRF bit in the SSR to 1. If the RIE bit is set to 1, a receive-end interrupt (RXI) is requested.
5. The RDRF bit is cleared to 0 when the CPU reads the SSR, then writes a 0 in the RDRF bit, or when the RDR is read by the data transfer controller (DTC). The RDR is then ready to receive the next character from the RSR.

When a frame is not received correctly, a receive error occurs. There are three types of receive errors, listed in table 13-8.

If a receive error occurs, the RDRF bit in the SSR is not set to 1. The corresponding error flag is set to 1 instead. If the RIE bit in the SCR is set to 1, a receive-error interrupt (ERI) is requested.

When a framing or parity error occurs, the RSR contents are transferred to the RDR. If an overrun error occurs, however, the RSR contents are not transferred to the RDR.

If multiple receive errors occur simultaneously, all the corresponding error flags are set to 1.

To clear a receive-error flag (ORER, FER, or PER), software must read the SSR, then write a 0 in the flag bit.

Table 13-8 Receive Errors

Name	Abbreviation	Description
Overrun error	ORER	Reception of the next frame ends while the RDRF bit is still set to 1. The RSR contents are not transferred to the RDR.
Framing error	FER	A stop bit is 0. The RSR contents are transferred to the RDR.
Parity error	PER	The parity of a frame does not match the value selected by the bit in the SMR. The RSR contents are transferred to the RDR.

13.3.3 Synchronous Mode

The synchronous mode is suited for high-speed, continuous data transfer. Each bit of data is synchronized with a serial clock pulse.

Continuous data transfer is enabled by the double buffering employed in both the transmit and receive sections of the SCI. Full duplex communication is possible because the transmit and receive sections are independent.

1. Data Format

Figure 13-4 shows the communication format used in the synchronous mode. The data length is 8 bits for both the transmit and receive directions. The least significant bit (LSB) is sent and received first. Each bit of transmit data is output from the falling edge of the serial clock pulse to the next falling edge. Received bits are latched on the rising edge of the serial clock pulse.

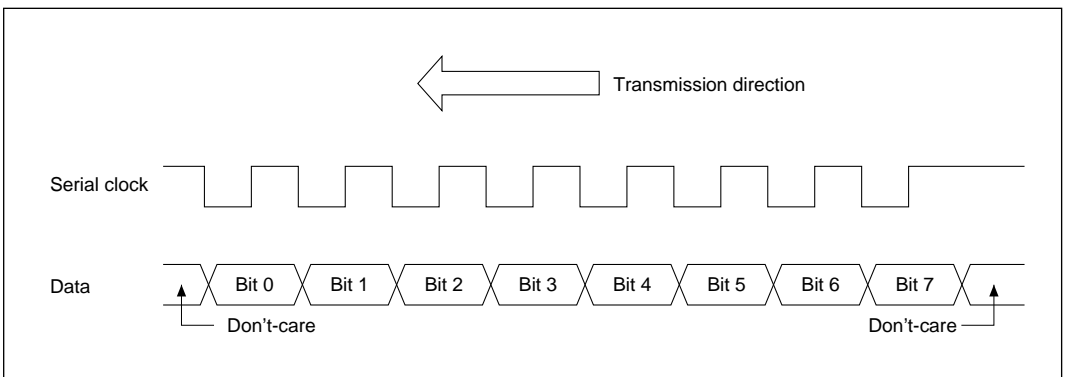


Figure 13-4 Data Format in Synchronous Mode

2. Clock

Either the internal serial clock created by the on-chip baud rate generator or an external clock input at the SCK pin can be selected in the synchronous mode. See table 13-6 for details.

3. Data Transmission and Reception

SCI Initialization: Before data can be transmitted or received, the SCI must be initialized by software. To initialize the SCI, software must clear the TE and RE bits to 0 to disable both the transmit and receive functions, then execute the following procedure.

1. Write the value corresponding to the desired bit rate in the BRR. (This step is not necessary if an external clock is used.)
2. Select the clock in the SCR.
3. Select the synchronous mode in the SMR*.
4. Set the TE and/or RE bit to 1, and enable desired interrupts in the SCR.

The TE and RE bits must both be cleared to 0 whenever the operating mode or data format is changed. After changing the operating mode or data format, before setting the TE and RE bits to 1 software must wait for at least 1 bit transfer time at the selected communication speed, to make sure the SCI is initialized.

* The SCK pin is used for input or output according to the $\overline{C/A}$ bit in the serial mode register (SMR) and the CKE0 and CKE1 bits in the serial control register (SCR). (See table 13-6.) To prevent unwanted output at the SCK pin, pay attention to the order in which you set SMR and SCR.

When clearing the TDRE bit during data transmission, to assure correct data transfer, do not clear the TDRE bit until after writing data in the TDR. Similarly, in receiving data, do not clear the RDRF bit until after reading data from the RDR.

Data Transmission: The procedure for transmitting data is as follows.

1. Set up the desired transmitting conditions in the SMR, BRR, and SCR.
2. Set the TE bit in the SCR to 1.
The TXD pin will automatically be switched to output, after which the SCI is ready to transmit data.
3. Check that the TDRE bit is set to 1, then write the first byte of transmit data in the TDR. Next clear the TDRE bit to 0.
4. The first byte of transmit data is transferred from the TDR to the TSR and sent, each bit synchronized with a clock pulse. Bit 0 is sent first.
Transfer of the transmit data from the TDR to the TSR makes the TDR empty, so the TDRE bit is set to 1. If the TIE bit is set to 1, a transmit-end interrupt (TXI) is requested.

The TDR and TSR function as a double buffer. Continuous data transmission can be achieved by writing the next transmit data in the TDR and clearing the TDRE bit to 0 while the SCI is transmitting the current data from the TSR.

If an internal clock source is selected, after transferring the transmit data from the TDR to the TSR, while transmitting the data from the TSR the SCI also outputs a serial clock signal at the SCK pin. When all data bits in the TSR have been transmitted, if the TDR is empty (TDRE = 1), serial clock output is suspended until the next data byte is written in the TDR and the TDRE bit is cleared to 0. During this interval the TXD pin is held at the value of the last bit transmitted.

If the external clock source is selected, data transmission is synchronized with the clock signal input at the SCK pin. When all data bits in the TSR have been transmitted, if the TDR is empty (TDRE = 1) but external clock pulses continue to arrive, the TXD pin outputs a string of bits equal to the last bit transmitted.

Data Reception: The procedure for receiving data is as follows.

1. Set up the desired receiving conditions in the SMR, BRR, and SCR.
2. Set the RE bit in the SCR to 1.
The RXD pin will automatically be switched to input and the SCI is ready to receive data.
3. Incoming data bits are latched in the RSR on eight clock pulses.
When 8 bits of data have been received, the SCI sets the RDRF bit in the SSR to 1. If the RIE bit is set to 1, a receive-end interrupt (RXI) is requested.
4. The SCI transfers the received data byte to the RDR so that it can be read.
The RDRF bit is cleared when the program reads the RDRF bit in the SSR, then writes a 0 in the RDRF bit, or when the data transfer controller (DTC) reads the RDR.

The RDR and RSR function as a double buffer. Data can be received continuously by reading each byte of data from the RDR and clearing the RDRF bit to 0 before the last bit of the next byte is received.

In general, an external clock source should be used for receiving data.

If an internal clock source is selected, the SCI starts receiving data as soon as the RE bit is set to 1. The serial clock is also output at the SCK pin. The SCI continues receiving until the RE bit is cleared to 0.

If the last bit of the next data byte is received while the RDRF bit is still set to 1, an overrun error occurs and the ORER bit is set to 1. If the RIE bit is set to 1, a receive-error interrupt (ERI) is requested. The data received in the RSR are not transferred to the RDR when an overrun error occurs.

After an overrun error, reception of the next data is enabled when the ORER bit is cleared to 0.

Simultaneous Transmit and Receive: The procedure for transmitting and receiving simultaneously is as follows:

1. Set up the desired communication conditions in the SMR, BRR, and SCR.
2. Set the TE and RE bits in the SCR to 1.
The TXD and RXD pins are automatically switched to output and input, respectively, and the SCI is ready to transmit and receive data.
3. Data transmitting and receiving start when the TDRE bit in the SSR is cleared to 0.
4. Data are sent and received in synchronization with eight clock pulses.
5. First, the transmit data are transferred from the TDR to the TSR. This makes the TDR empty, so the TDRE bit is set to 1. If the TIE bit is set to 1, a transmit-end interrupt (TXI) is requested.
If continuous data transmission is desired, the CPU must read the TDRE bit in the SSR, write the next transmit data in the TDR, then clear the TDRE bit to 0. Alternatively, the DTC can write the next transmit data in the TDR, in which case the TDRE bit is cleared automatically. If the TDRE bit is not cleared to 0 by the time the SCI finishes sending the current byte from the TSR, the TXD pin continues to output the last bit in the TSR.
6. In the receiving section, when 8 bits of data have been received they are transferred from the RSR to the RDR and the RDRF bit in the SSR is set to 1. If the RIE bit is set to 1, a receive-end interrupt (RXI) is requested.
7. To clear the RDRF bit, software read the RDRF bit in the SSR, read the data in the RDR, then write a 0 in the RDRF bit. Alternatively, the DTC can read the RDR, in which case the RDRF bit is cleared automatically.
For continuous data reception, the RDRF bit must be cleared to 0 before the last bit of the next byte of data is received.

If the last bit of the next byte is received while the RDRF bit is still set to 1, an overrun error occurs. The error is handled as described under “Data Reception” above. The overrun error does not affect the transmit section of the SCI, which continues to transmit normally.

13.4 CPU Interrupts and DTC Interrupts

The SCI can request three types of interrupts: transmit-end (TXI), receive-end (RXI), and receive-error (ERI). Interrupt requests are enabled or disabled by the TIE and RIE bits in the SCR. Independent signals are sent to the interrupt controller for each type of interrupt. The transmit-end and receive-end interrupt request signals are obtained from the TDRE and RDRF flags. The receive-error interrupt request signal is the logical OR of the three error flags: overrun error (ORER), framing error (FER), and parity error (PER). Table 13-9 lists information about these interrupts.

Table 13-9 SCI Interrupts

Interrupt	Description	DTC Service Available?	Priority
ERI	Receive-error interrupt, requested when ORER, FER, or PER is set.	No	High
RXI	Receive-end interrupt, requested when RDRF is set.	Yes	↑ Low
TXI	Transmit-end interrupt, requested when TDRE is set.	Yes	

The TXI and RXI interrupts can be served by the data transfer controller (DTC) to have a data transfer performed. When the DTC serves one of these interrupts, it clears the TDRE or RDRF bit to 0 under the following conditions, which differ between the two bits.

When invoked by a TXI request, if the DTC writes to the TDR, it automatically clears the TDRE bit to 0. When invoked by an RXI request, if the DTC reads from the RDR, it automatically clears the RDRF bit to 0.

See section 6, “Data Transfer Controller” for further information on the DTC.

13.5 Application Notes

Application programmers should note the following features of the SCI.

TDR Write: The TDRE bit in the SSR is simply a flag that indicates that the TDR contents have been transferred to the TSR. The TDR contents can be rewritten regardless of the TDRE value. If a new byte is written in the TDR while the TDRE bit is 0, before the old TDR contents have been moved into the TSR, the old byte will be lost. Normally, software should check that the TDRE bit is set to 1 before writing to the TDR.

Multiple Receive Errors: Table 13-10 lists the values of flag bits in the SSR when multiple receive errors occur, and indicates whether the RSR contents are transferred to the RDR.

Table 13-10 SSR Bit States and Data Transfer When Multiple Receive Errors Occur

Receive Error	SSR Bits				RSR to RDR*2
	RDRF	ORER	FER	PER	
Overrun error	1*1	1	0	0	No
Framing error	0	0	1	0	Yes
Parity error	0	0	0	1	Yes
Overrun + framing errors	1*1	1	1	0	No
Overrun + parity errors	1*1	1	0	1	No
Framing + parity errors	0	0	1	1	Yes
Overrun + framing + parity errors	1*1	1	1	1	No

Notes: *1 Set to 1 before the overrun error occurs.

*2 Yes: The RSR contents are transferred to the RDR.

No: The RSR contents are not transferred to the RDR.

Line Break Detection: When the RXD pin receives a continuous stream of 0's in the asynchronous mode (line-break state), a framing error occurs because the SCI detects a 0 stop bit. The value H'00 is transferred from the RSR to the RDR. Software can detect the line-break state as a framing error accompanied by H'00 data in the RDR.

The SCI continues to receive data, so if the FER bit is cleared to 0 another framing error will occur.

Sampling Timing and Receive Margin in Asynchronous Mode: The serial clock used by the SCI in asynchronous mode runs at 16 times the bit rate. The falling edge of the start bit is detected by sampling the RXD input on the falling edge of this clock. After the start bit is detected, each bit of receive data in the frame (including the start bit, parity bit, and stop bit or bits) is sampled on the rising edge of the serial clock pulse at the center of the bit. See figure 13-5.

It follows that the receive margin can be calculated as in equation (1).

When the absolute frequency deviation of the clock signal is 0 and the clock duty factor is 0.5, data can theoretically be received with distortion up to the margin given by equation (2). This is a theoretical limit, however. In practice, system designers should allow a margin of 20% to 30%.

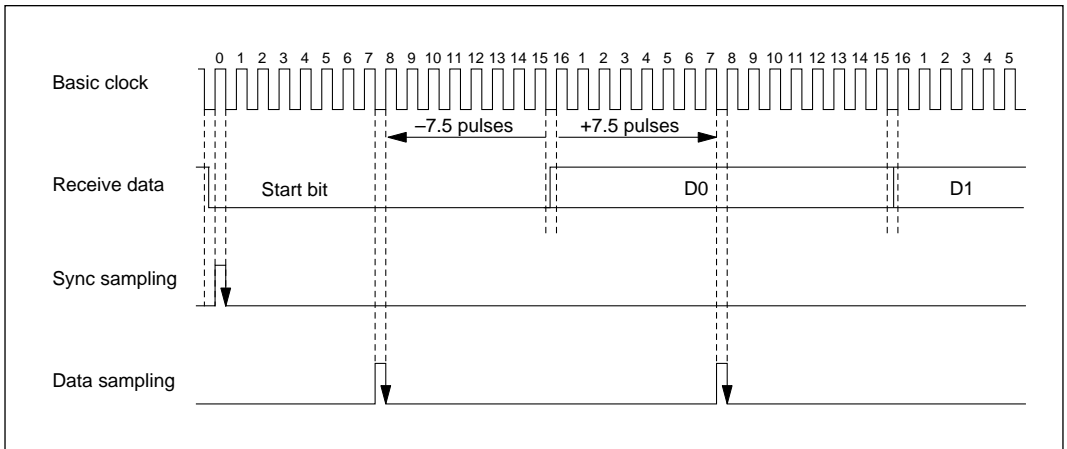


Figure 13-5 Sampling Timing (Asynchronous Mode)

$$M = \{ (0.5 - 1/2N) - (D - 0.5)/N - (L - 0.5)F \} \times 100 [\%] \quad (1)$$

M: Receive margin

N: Ratio of basic clock to bit rate (16)

D: Duty factor of clock—ratio of High pulse width to Low width (0.5 to 1.0)

L: Frame length (9 to 12)

F: Absolute clock frequency deviation

When $D = 0.5$ and $F = 0$

$$M = (0.5 - 1/2 \times 16) \times 100 [\%] = 46.875\% \quad (2)$$

Section 14 A/D Converter

14.1 Overview

The H8/510 chip includes an analog-to-digital converter module which can be programmed for input of analog signal on up to four channels. A/D conversion is performed by the successive approximations method with 10-bit resolution.

14.1.1 Features

The features of the on-chip A/D module are:

- Four analog input channels
- Sample and hold circuit
- 10-Bit resolution
- Rapid conversion
Conversion time is 13.4 μ s per channel (at $\phi = 10$ MHz)
- Single and scan modes
 - Single mode: A/D conversion is performed once.
 - Scan mode: A/D conversion is performed in a repeated cycle on one to four channels.
- Four 16-bit data registers
These registers store A/D conversion results for up to four channels.
- A CPU interrupt (ADI) can be requested at the completion of each A/D conversion cycle.
This interrupt can also be served by the on-chip data transfer controller (DTC), providing a convenient way to move results into memory.
- The start of A/D conversion can be externally triggered.

14.1.2 Block Diagram

Figure 14-1 shows a block diagram of A/D converter.

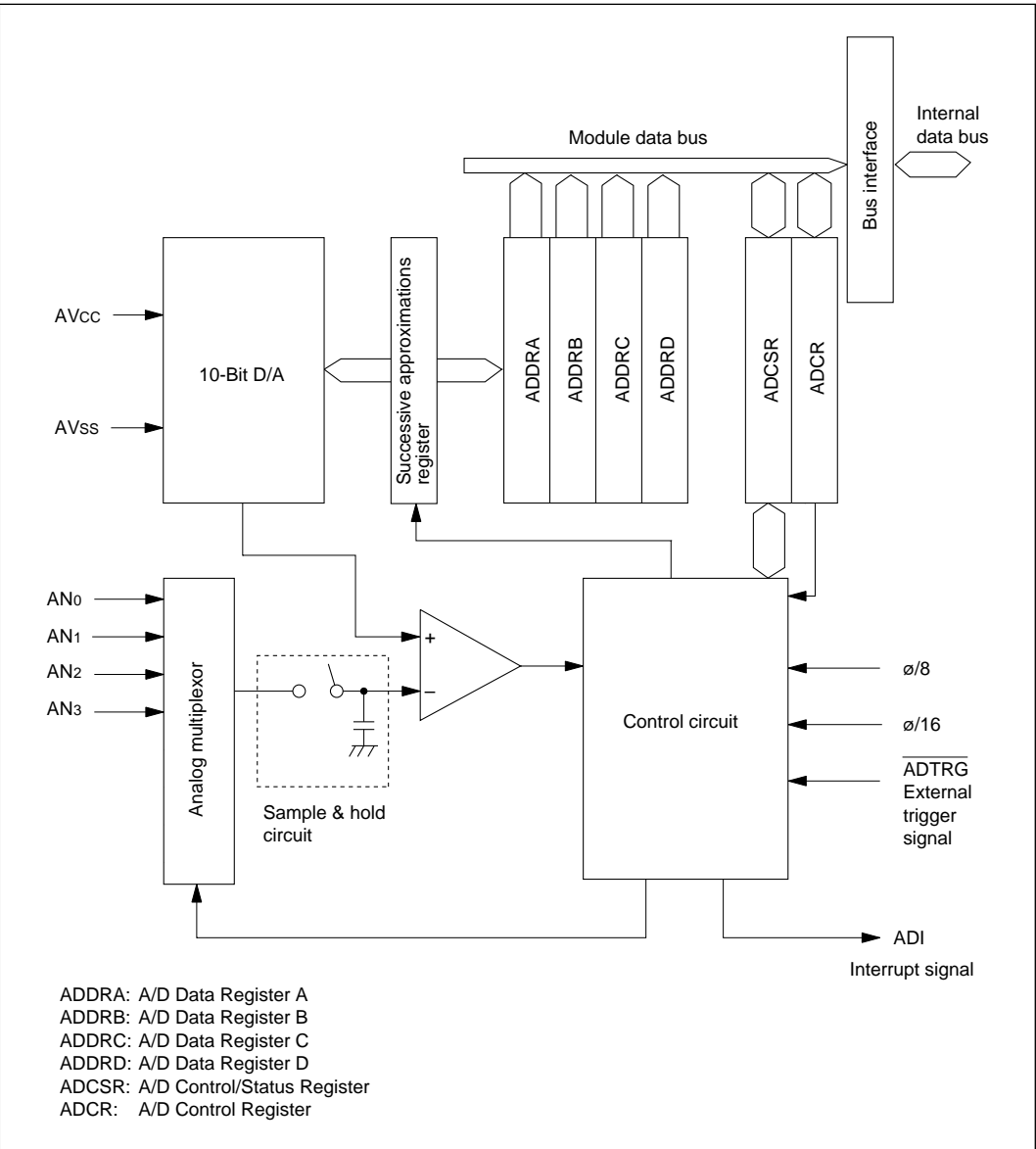


Figure 14-1 Block Diagram of A/D Converter

14.1.3 Input Pins

Table 14-1 lists the input pins used by the A/D converter module.

Table 14-1 A/D Input Pins

Name	Abbreviation	I/O	Function
Analog supply voltage	AVcc	Input	Power supply and reference voltage for the analog circuits.
Analog ground	AVss	Input	Ground and reference voltage for the analog circuits.
Analog input 0	AN0	Input	Analog input pins
Analog input 1	AN1	Input	
Analog input 2	AN2	Input	
Analog input 3	AN3	Input	
A/D trigger	ADTRG	Input	Trigger input for start of A/D conversion

14.1.4 Register Configuration

Table 14-2 lists the registers of the A/D converter module.

Table 14-2 A/D Registers

Name	Abbreviation	R/W	Initial Value	Address
A/D data register A (High)	ADDRA (H)	R	H'00	H'FE90
A/D data register A (Low)	ADDRA (L)	R	H'00	H'FE91
A/D data register B (High)	ADDRB (H)	R	H'00	H'FE92
A/D data register B (Low)	ADDRB (L)	R	H'00	H'FE93
A/D data register C (High)	ADDRC (H)	R	H'00	H'FE94
A/D data register C (Low)	ADDRC (L)	R	H'00	H'FE95
A/D data register D (High)	ADDRD (H)	R	H'00	H'FE96
A/D data register D (Low)	ADDRD (L)	R	H'00	H'FE97
A/D control/status register	ADCSR	R/(W)*	H'00	H'FE98
A/D control register	ADCR	R/W	H'7F	H'FE99

* Software can write 0 to clear the status flag bits but cannot write 1.

14.2 Register Descriptions

14.2.1 A/D Data Registers (ADDR)—H'FE90 to H'FE97

Bit	7	6	5	4	3	2	1	0
ADDRn H	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R
(n = A to D)								

Bit	7	6	5	4	3	2	1	0
ADDRn L	AD1	AD0	—	—	—	—	—	—
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R
(n = A to D)								

The four A/D data registers (ADDRA to ADDR D) are 16-bit read-only registers that store the results of A/D conversion.

Each result consist of 10 bits. The first 8 bits are stored in the upper byte of the data register corresponding to the selected channel. The last two bits are stored in the lower data register byte. The data registers are assigned to analog input channels as indicated in table 14-3.

The A/D data registers are always readable by the CPU. The upper byte can be read directly. The lower byte is read via a temporary register. See section 14-3, “CPU Interface” for details.

The unused bits (bits 5 to 0) of the lower data register byte are always read as 0.

The A/D data registers are initialized to H'0000 at a reset and in the standby modes.

Table 14-3 Assignment of Data Registers to Analog Input Channels

Analog Input Channel	A/D Data Register
AN0	ADDRA
AN1	ADDRB
AN2	ADDRC
AN3	ADDRD

14.2.2 A/D Control/Status Register (ADCSR)—H'FE98

Bit	7	6	5	4	3	2	1	0
	ADF	ADIE	ADST	SCAN	CKS	CH2*2	CH1	CH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*1	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Notes: *1 Software can write a 0 in bit 7 to clear the flag, but cannot write a 1 in this bit.

*2 The CH2 bit should always be cleared to 0.

The A/D control/status register (ADCSR) is an 8-bit readable/writable register that controls the operation of the A/D converter module.

The ADCSR is initialized to H'00 at a reset and in the standby modes.

Bit 7—A/D End Flag (ADF): This status flag indicates the end of one cycle of A/D conversion.

Bit 7

ADF	Description
0	This bit is cleared from 1 to 0 when: (Initial value) 1. The chip is reset or placed in a standby mode. 2. The CPU reads the ADF bit after it has been set to 1, then writes a 0 in this bit. 3. An A/D interrupt is served by the data transfer controller (DTC).
1	This bit is set to 1 at the following times: 1. Single mode: when one A/D conversion is completed. 2. Scan mode: when inputs on all selected channels have been converted.

Bit 6—A/D Interrupt Enable (ADIE): This bit selects whether to request an A/D interrupt (ADI) when A/D conversion is completed.

Bit 6

ADIE	Description
0	The A/D interrupt request (ADI) is disabled. (Initial value)
1	The A/D interrupt request (ADI) is enabled.

Bit 5—A/D Start (ADST): The A/D converter operates while this bit is set to 1. In the single mode, this bit is automatically cleared to 0 at the end of each A/D conversion.

Bit 5

ADST	Description
0	A/D conversion is halted. (Initial value)
1	<ol style="list-style-type: none"> 1. Single mode: One A/D conversion is performed. The ADST bit is automatically cleared to 0 at the end of the conversion. 2. Scan mode: A/D conversion starts and continues cyclically on the selected channels until the ADST bit is cleared to 0.

Bit 4—Scan Mode (SCAN): This bit selects the scan mode or single mode of operation.

See section 14.4, “Operation” for descriptions of these modes.

The mode should be changed only when the ADST bit is cleared to 0.

Bit 4

SCAN	Description
0	Single mode (Initial value)
1	Scan mode

Bit 3—Clock Select (CKS): This bit controls the A/D conversion time.

The conversion time should be changed only when the ADST bit is cleared to 0.

Bit 3

CKS	Description
0	Conversion time = 266 states (Initial value)
1	Conversion time = 134 states

Bits 2 to 0—Channel Select 2 to 0 (CH2 to CH0): These bits and the SCAN bit combine to select one or more analog input channels.

The channel selection should be changed only when the ADST bit is cleared to 0.

Group Select	Channel Select		Selected Channels	
CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	AN ₀	AN ₀
	0	1	AN ₁	AN ₀ and AN ₁
	1	0	AN ₂	AN ₀ to AN ₂
	1	1	AN ₃	AN ₀ to AN ₃

14.2.3 A/D Control Register (ADCR)—H'FE99

Bit	7	6	5	4	3	2	1	0
	TRGE	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

The A/D control register (ADCR) is an 8-bit readable/writable register that enables or disables the A/D external trigger signal.

The ADCR is initialized to H'7F at a reset and in the standby modes.

Bit 7—Trigger Enable (TRGE): The bit enables the $\overline{\text{ADTRG}}$ (A/D external trigger) signal. A High-to-Low transition of $\overline{\text{ADTRG}}$ sets the ADST bit, starting A/D conversion.

Bit 7

TRGE	Description
0	A/D external trigger is disabled. $\overline{\text{ADTRG}}$ does not set the ADST bit. (Initial value)
1	A/D external trigger is enabled. A High-to-Low transition of $\overline{\text{ADTRG}}$ sets the ADST bit. Pin P40 is set to input and used for $\overline{\text{ADTRG}}$ input.

Bit 6 to 0—Reserved: These bits cannot be modified and are always read as 1.

14.2.4 External Triggering of A/D Conversion

External trigger input is enabled at the $\overline{\text{ADTRG}}$ pin when the TRGE bit in the ADCR is set to 1.

One and one-half system clock cycles after the $\overline{\text{ADTRG}}$ input goes Low, the ADST bit in the ADCSR is set to 1 and A/D conversion commences.

The timing of external triggering is shown in figure 14-2.

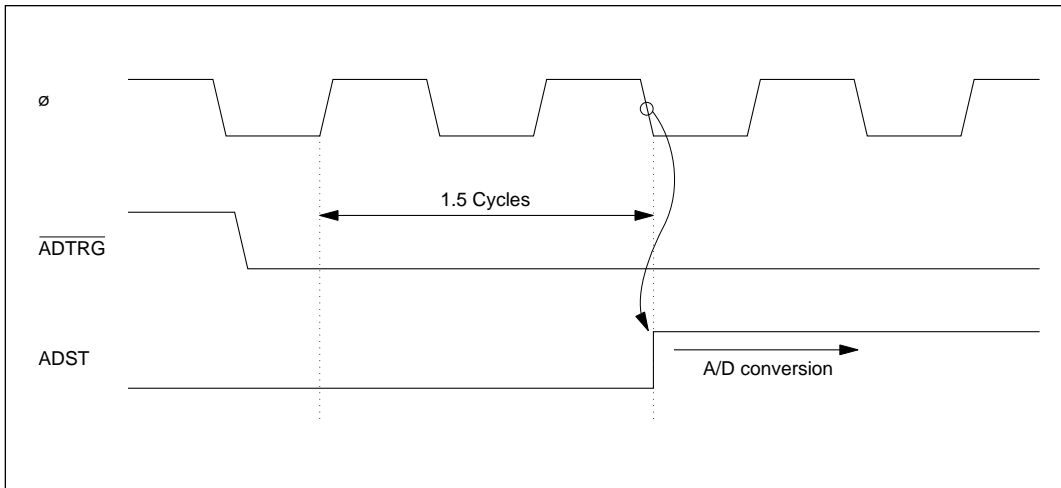


Figure 14-2 External Trigger Input Timing

14.3 CPU Interface

The A/D data registers (ADDRA to ADDR D) are 16-bit registers. The upper byte of each register can be read directly, but the lower byte is accessed through an 8-bit temporary register (TEMP).

When the CPU or DTC reads the upper byte of an A/D data register, at the same time as the upper byte is placed on the internal data bus, the lower byte is transferred to TEMP. When the lower byte is accessed, the value in TEMP is placed on the internal data bus.

A program that requires all 10 bits of an A/D result should perform word access, or should read first the upper byte, then the lower byte of the A/D data register. Either way, it is assured of obtaining consistent data. Consistent data are not assured if the program reads the lower byte first.

A program that requires only 8-bit A/D accuracy should perform byte access to the upper byte of the A/D data register. The value in TEMP can be left unread.

Figure 14-3 shows the data flow when the CPU (or DTC) reads an A/D data register.

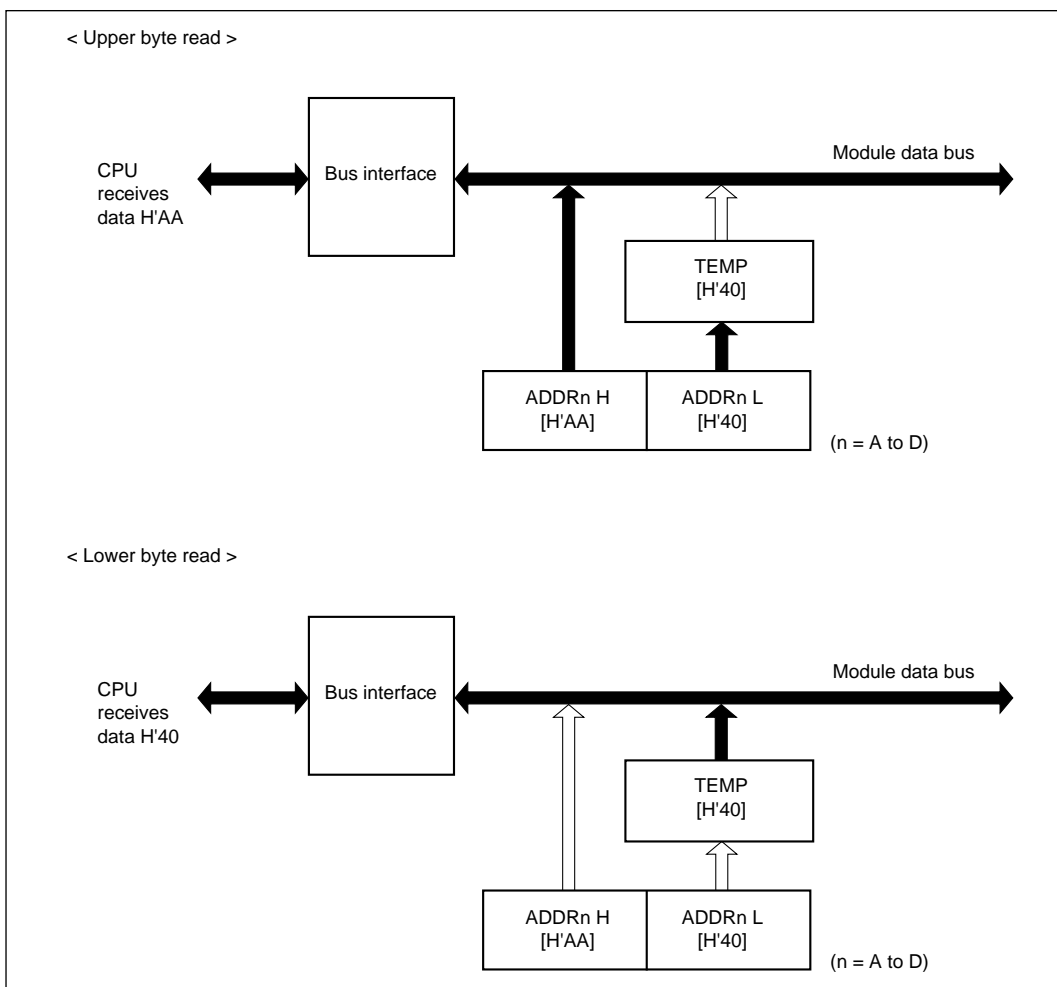


Figure 14-3 Read Access to A/D Data Register (When Register Contains H'AA40)

14.4 Operation

The A/D converter performs 10 successive approximations to obtain a result ranging from H'0000 (corresponding to AVSS) to H'FFC0 (corresponding to AVCC). Only the first 10 bits of the result are significant.

The A/D converter module can be programmed to operate in single mode or scan mode as explained below.

14.4.1 Single Mode

The single mode is suitable for obtaining a single data value from a single channel. A/D conversion starts when the ADST bit is set to 1. During the conversion process the ADST bit remains set to 1. When conversion is completed, the ADST bit is automatically cleared to 0.

When the conversion is completed, the ADF bit is set to 1. If the interrupt enable bit (ADIE) is also set to 1, an A/D conversion end interrupt (ADI) is requested, so that the converted data can be processed by an interrupt-handling routine. Alternatively, the interrupt can be served by the data transfer controller (DTC).

When an A/D interrupt is served by the DTC, the DTC automatically clears the ADF bit to 0. When an A/D interrupt is served by the CPU, however, the ADF bit remains set until the CPU reads the ADCSR, then writes a 0 in the ADF bit.

Before selecting the single mode, clock, and analog input channel, software should clear the ADST bit to 0 to make sure the A/D converter is stopped. Changing the mode, clock, or channel selection while A/D conversion is in progress can lead to conversion errors.

The following example explains the A/D conversion process in single mode when channel 1 (AN1) is selected. Figure 14-4 shows the corresponding timing chart.

1. Software clears the ADST bit to 0, then selects the single mode (SCAN = 0) and channel 1 (CH2 to CH0 = “001”), enables the A/D interrupt request (ADIE = 1), and sets the ADST bit to 1 to start A/D conversion. (Selection of mode, clock channel and setting the ADST bit can be done at same time.)

Coding Example: (when using the slow clock, CKS = 0)

```
BCLR #5, @H'FE98
```

```
MOV.B #H'61, @H'FE98
```

2. The A/D converter samples the AN1 input and converts the voltage level to a digital value. At the end of the conversion process the A/D converter transfers the result to register ADDR_B, sets the ADF bit is set to 1, clears the ADST bit to 0, and halts.
3. ADF = 1 and ADIE = 1, so an A/D interrupt is requested.
4. The user-coded A/D interrupt-handling routine is started.
5. The interrupt-handling routine reads the ADCSR value, then writes a 0 in the ADF bit to clear this bit to 0.
6. The interrupt-handling routine reads and processes the A/D conversion result.
7. The routine ends.

Steps 2 to 7 can now be repeated by setting the ADST bit to 1 again.

If the data transfer enable (DTE) bit is set to 1, the interrupt is served by the data transfer controller (DTC). Steps 4 to 7 then change as follows.

4'. The DTC is started.

5'. The DTC automatically clears the ADF bit to 0.

6'. The DTC transfers the A/D conversion result from ADDR0 to a specified destination address.

7'. The DTC ends.

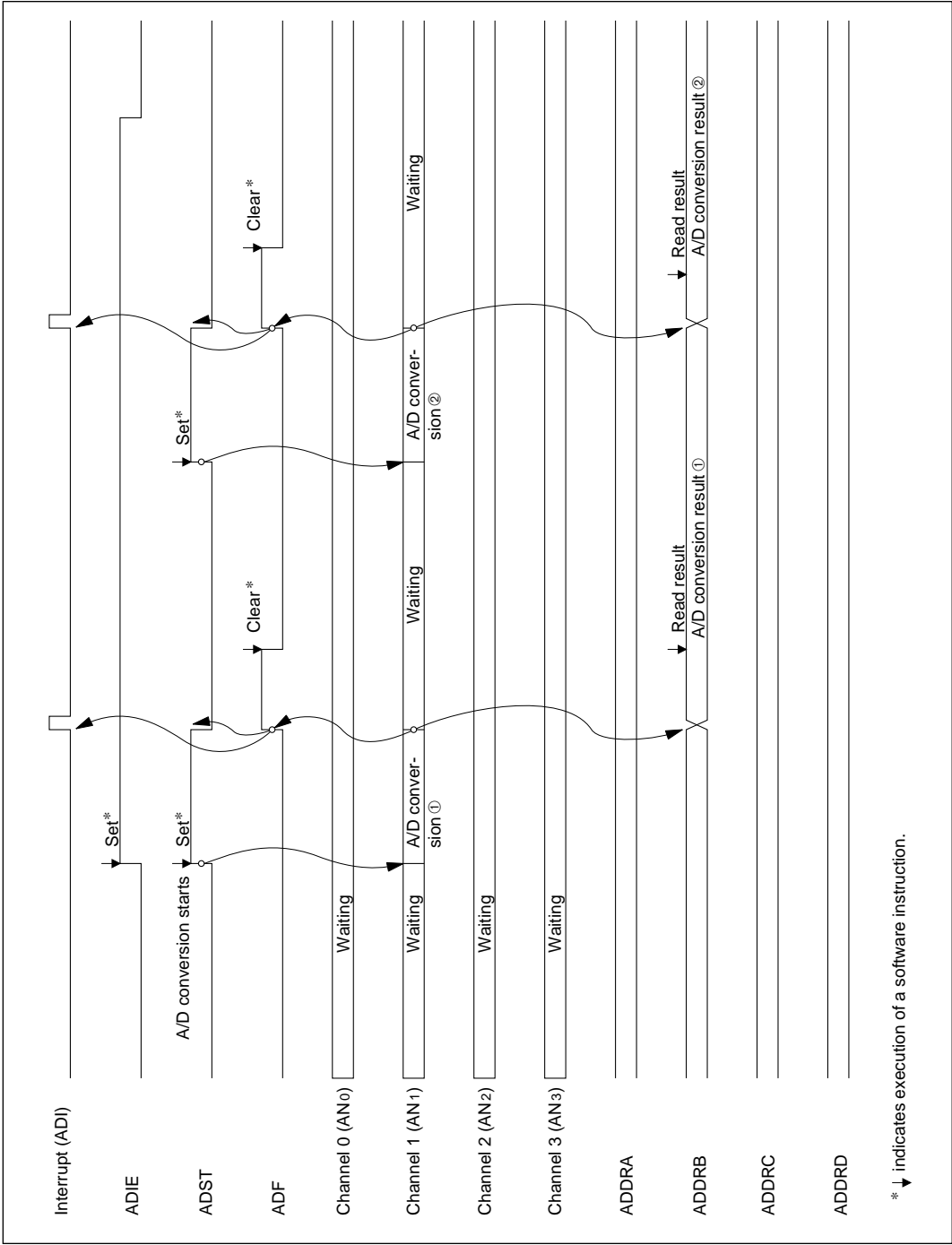


Figure 14-4 A/D Operation in Single Mode (When Channel 1 is Selected)

14.4.2 Scan Mode

The scan mode can be used to monitor analog inputs on one or more channels. When the ADST bit is set to 1, A/D conversion starts from the first channel (AN0).

If the scan group includes more than one channel (i.e. if bit CH1 or CH0 is set), conversion of the next channel begins as soon as conversion of the first channel ends.

Conversion of the selected channels continues cyclically until the ADST bit is cleared to 0. The conversion results are placed in the data registers corresponding to the selected channels.

Before selecting the scan mode, clock, and analog input channels, software should clear the ADST bit to 0 to make sure the A/D converter is stopped. Changing the mode, clock, or channel selection while A/D conversion is in progress can lead to conversion errors.

The following example explains the A/D conversion process when three channels are selected (AN0, AN1, and AN2). Figure 14-5 shows the corresponding timing chart.

1. Software clears the ADST bit to 0, then selects the scan mode (SCAN = 1), scan group 0 (CH2 = 0), and analog input channels AN0 to AN2 (CH1 and CH0 = 0) and sets the ADST bit to 1 to start A/D conversion.

Coding Example: (with slow clock and ADI interrupt enabled)

```
BCLR #5, @H'FE98  
MOV.B #H'72, @FE98
```

2. The A/D converter samples the input at AN0, converts the voltage level to a digital value, and transfers the result to register ADDRA.
3. Next the A/D converter samples and converts AN1 and transfers the result to ADDRb. Then it samples and converts AN2 and transfers the result to ADDRc.
4. After all selected channels (AN0 to AN2) have been converted, the AD converter sets the ADF bit to 1. If the ADIE bit is set to 1, an A/D interrupt (ADI) is requested. Then the A/D converter begins converting AN0 again.
5. Steps 2 to 4 are repeated cyclically as long as the ADST bit remains set to 1.

To stop the A/D converter, software must clear the ADST bit to 0.

Regardless of which channel is being converted when the ADST bit is cleared to 0, when the ADST bit is set to 1 again, conversion begins from the first selected channel (AN0).

14.5 Input Sampling Time and A/D Conversion Time

The A/D converter includes a built-in sample-and-hold circuit. Sampling of the input starts at a time t_D after the ADST bit is set to 1. The sampling process lasts for a time t_{SPL} . The actual A/D conversion begins after sampling is completed. Figure 14-6 shows the timing of these steps, and table 14-4 lists the total conversion times (t_{CONV}) for the single mode.

The total conversion time includes t_D and t_{SPL} . The purpose of t_D is to synchronize the ADCSR write time with the A/D conversion process, so the length of t_D is variable. The total conversion time therefore varies within the minimum to maximum ranges indicated in table 14-4.

In the scan mode, the ranges given in table 14-4 apply to the first conversion. The length of the second and subsequent conversion processes is fixed at 256 states (when $CKS = 0$) or 128 states (when $CKS = 1$).

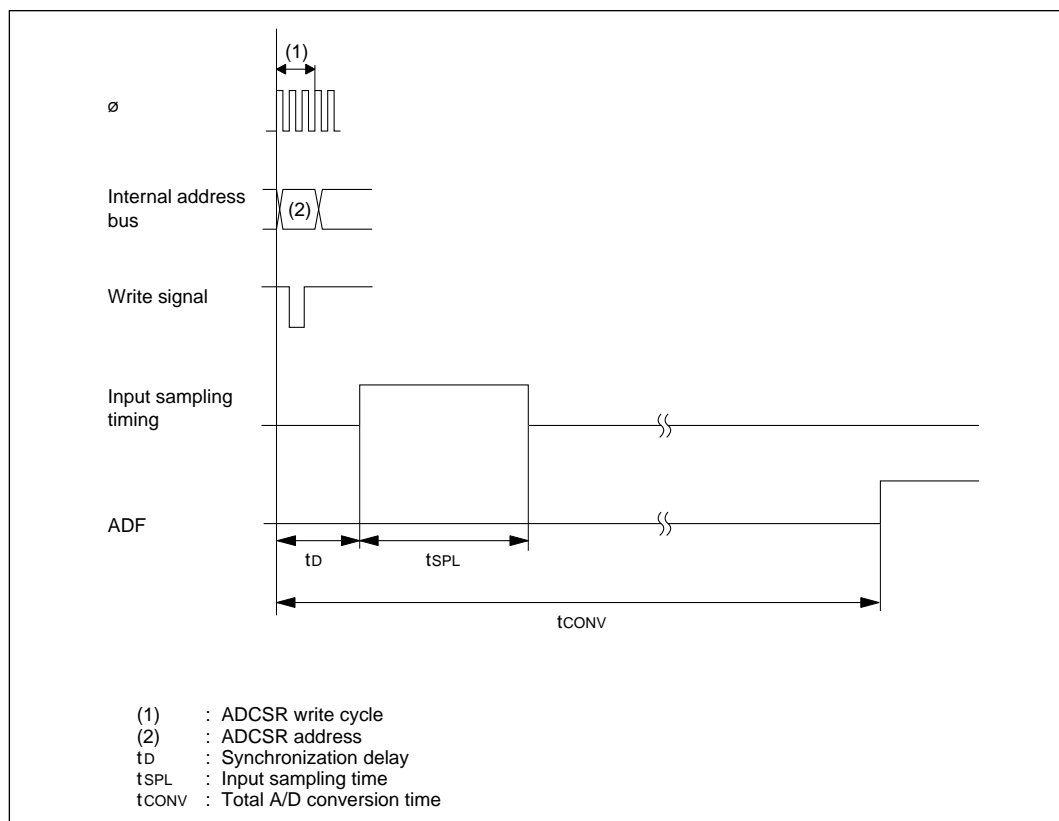


Figure 14-6 A/D Conversion Timing

Table 14-4 A/D Conversion Time (Single Mode)

Item	Symbol	CKS = 0			CKS = 1		
		Min	Typ	Max	Min	Typ	Max
Synchronization delay	t_D	10	—	17	6	—	9
Input sampling time	t_{SPL}	—	80	—	—	40	—
Total A/D conversion time	t_{CONV}	259	—	266	131	—	134

Note: Values in the table are numbers of states.

14.6 Interrupts and the Data Transfer Controller

The ADI interrupt request is enabled or disabled by the ADIE bit in the ADCSR.

When the ADI bit in data transfer enable register DTED (bit 0 at address H'FF0B) is set to 1, the ADI interrupt is served by the data transfer controller. The DTC can be used to transfer A/D results to a buffer in memory, or to an I/O port. The DTC automatically clears the ADF bit to 0.

Note: In scan mode, the DTC can transfer data for only one channel per interrupt, even if two or more channels are selected.

Section 15 Bus Controller

15.1 Overview

The H8/510 has an on-chip bus controller that enables the bus width and bus cycle length to be altered dynamically.

When a 16-bit bus width is selected by the inputs at the mode pins, part of the address space can be reserved for access via an 8-bit bus (byte-access area). The bus controller can also switch an area between 8-bit and 16-bit access, and shorten the bus cycle from three states to two states for high-speed access.

15.1.1 Features

The bus controller has the following features:

- Setting of 8-bit data bus access area (in modes 2 and 4)
Addresses greater than the address set in the byte area top register (ARBT) are designated for 8-bit access. (This area does not include the address set in ARBT, which is the boundary address of the word area.)
When an address greater than the address set in ARBT is accessed, only the upper data bus lines (D15 to D8) are used, so the access is carried out with an 8-bit bus width. The bus width of the internal and external I/O areas, however, is not changed by the ARBT setting.
- Setting of two-state access area
Addresses equal to or greater than the address set in the three-state area top register (AR3T) are designated for three-state access. (This area includes the address set in AR3T, which is the boundary of the three-state area.)
When an address less than the address set in AR3T is accessed, it is accessed using a two-state bus cycle. Wait states cannot be inserted into two-state access cycles. The bus cycle length of the internal and external I/O areas is not changed by the AR3T setting.
- The boundaries of the word and three-state areas are set to multiples of 4 kbytes in minimum mode, and multiples of 64 kbytes in maximum mode.

15.1.2 Block Diagram

Figure 15-1 is a block diagram of the bus controller.

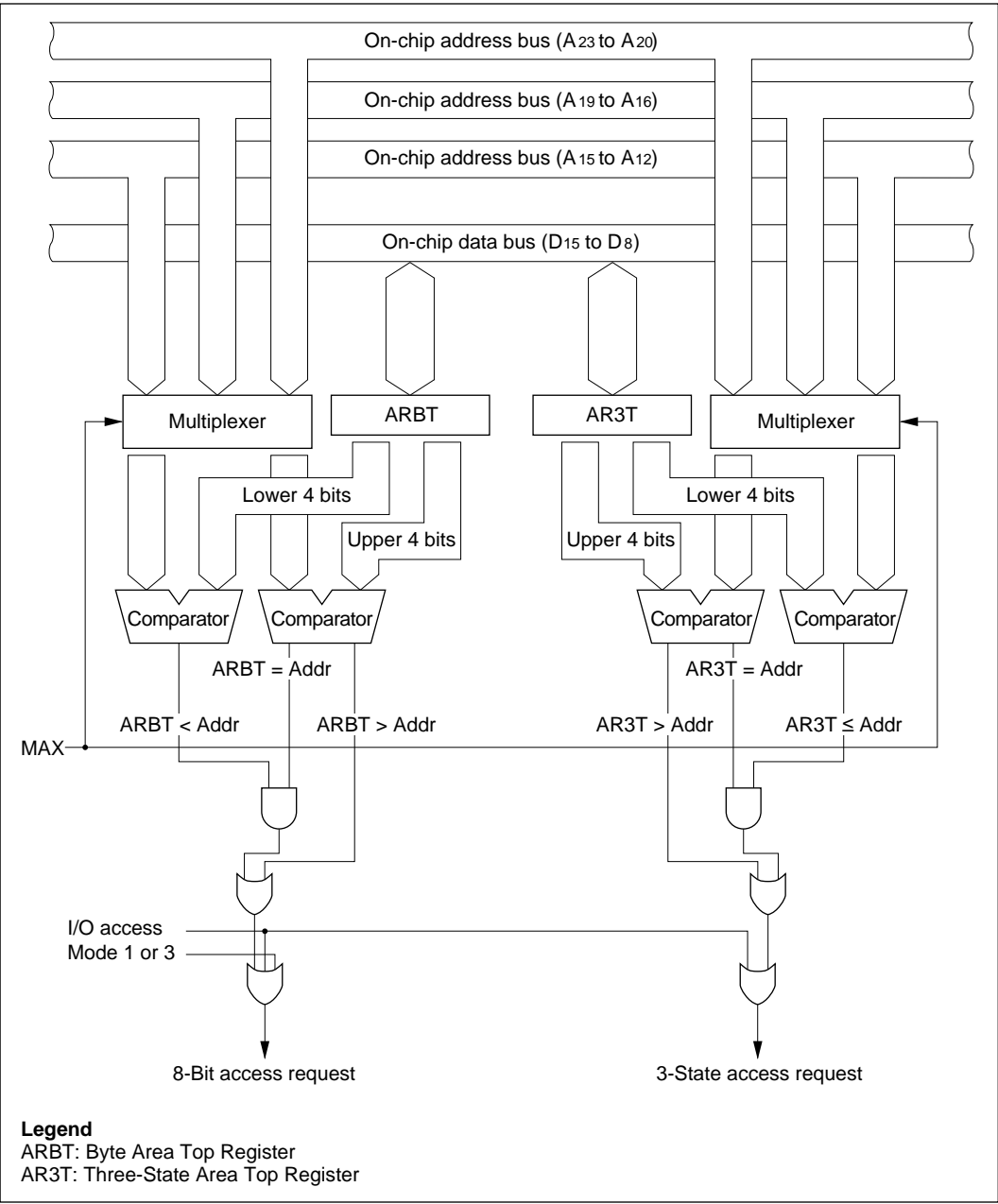


Figure 15-1 Block Diagram of Bus Controller

15.1.3 Register Configuration

Table 15-1 describes the bus controller registers.

The Bus Controller Has Two 8-Bit Registers: A byte area top register (ARBT) that designates the boundary of the word area, and a three-state area top register (AR3T) that designates the boundary of the three-state area.

Table 15-1 Bus Controller Registers

Name	Abbreviation	R/W	Initial Value	Address
Byte area top register	ARBT	R/W	H'FF	H'FF16
Three-state area top register	AR3T	R/W	H'00	H'FF17

15.2 Register Descriptions

15.2.1 Byte Area Top Register (ARBT)—H'FF16

The ARBT register designates the boundary between addresses that are accessed via a 16-bit data bus and addresses that are accessed using only the upper 8 bits of the 16-bit bus. The address set in ARBT is the last address accessed via a 16-bit-wide bus. This address is referred to as the word area boundary.

Bit:	7	6	5	4	3	2	1	0
Initial value:	1	1	1	1	1	1	1	1
Read/Write:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bus controller controls the CPU so that external addresses greater than the ARBT value are accessed via a 8-bit-wide bus.

In the expanded maximum modes the value in ARBT is used as the upper eight bits (A23 to A16) of the word area boundary address. The word area boundary address is therefore settable to a multiple of 64 kbytes. Note that in the expanded maximum modes addresses H'000000 to H'00FE7F are always located in the word access area.

In the expanded minimum modes only the four lowest ARBT bits are valid. They designate the upper four bits (A15 to A12) of the word area boundary address. The boundary address is therefore settable to a multiple of 4 kbytes. In the expanded minimum modes addresses H'0000 to H'0FFF are always located in the word access area.

The ARBT setting affects only the external address space. It does not alter the bus width of the internal and external I/O areas. In modes 1 and 3 the entire address space is accessed via an 8-bit data bus, so the ARBT setting is ignored.

ARBT is initialized to H'FF by a reset and in the hardware standby mode. It is not initialized in the software standby mode.

15.2.2 Three-State Area Top Register (AR3T)—H'FF17

The AR3T register designates the boundary between the two-state access area and the three-state access area. The value set in AR3T, referred to as the three-state area boundary, is the first address to be accessed in three states.

Bit:	7	6	5	4	3	2	1	0
Initial value:	0	0	0	0	0	0	0	0
Read/Write:	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bus controller controls the CPU so that external addresses equal to or greater than the AR3T value are accessed in three states. Wait states cannot be inserted in two-state access.

In the expanded maximum modes the AR3T value designates the upper eight bits (A23 to A16) of the three-state area boundary address. The three-state area boundary address is therefore settable as a multiple of 64 kbytes. Note that in the expanded maximum modes addresses H'F00000 to H'FFFFFF are always accessed in three states.

In the expanded minimum modes only the four lowest AR3T bits are valid. They designate the upper four bits (A15 to A12) of the three-state area boundary address. The three-state area boundary address is therefore settable as a multiple of 4 kbytes. In the expanded minimum modes addresses H'F000 to H'FFFF are always accessed in three states.

The AR3T setting affects only the external address space. It does not alter the bus cycle length of the external and internal I/O areas.

AR3T is initialized to H'00 by a reset and in the hardware standby mode. It is not initialized in the software standby mode.

15.3 Operation

1. Operation in Each Mode after a Reset: See figure 15-2.

Mode 1: The bus is 8 bits wide. Addresses H'0000 to H'FFFF are all accessed at 8 bits per three states.

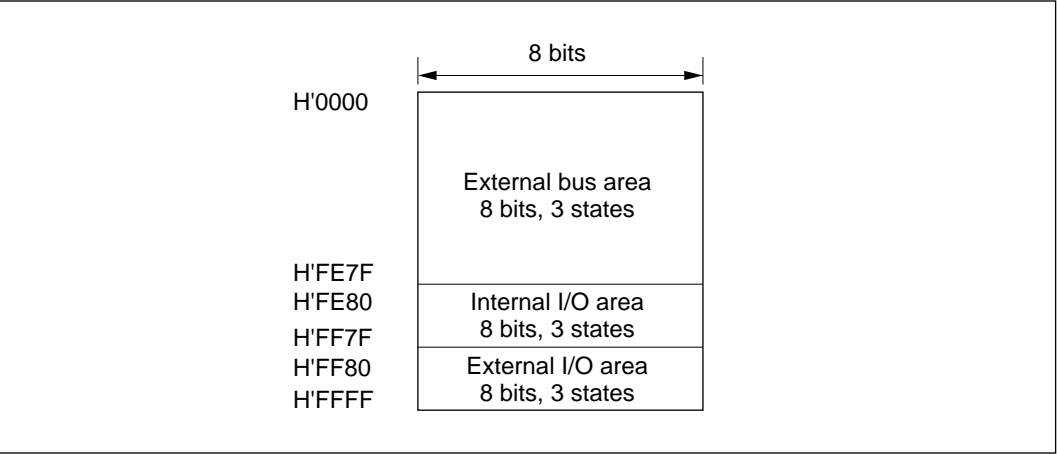


Figure 15-2 Bus Width and Cycle Length after Reset (Mode 1)

Mode 2: The bus is 16 bits wide. Addresses H'0000 to H'FE7F are accessed in three states via the 16-bit bus. Addresses H'FE80 to H'FF7F are the internal I/O area, accessed at 8 bits per three states. Addresses H'FF80 to H'FFFF are the external I/O area, also accessed at 8 bits per three states.

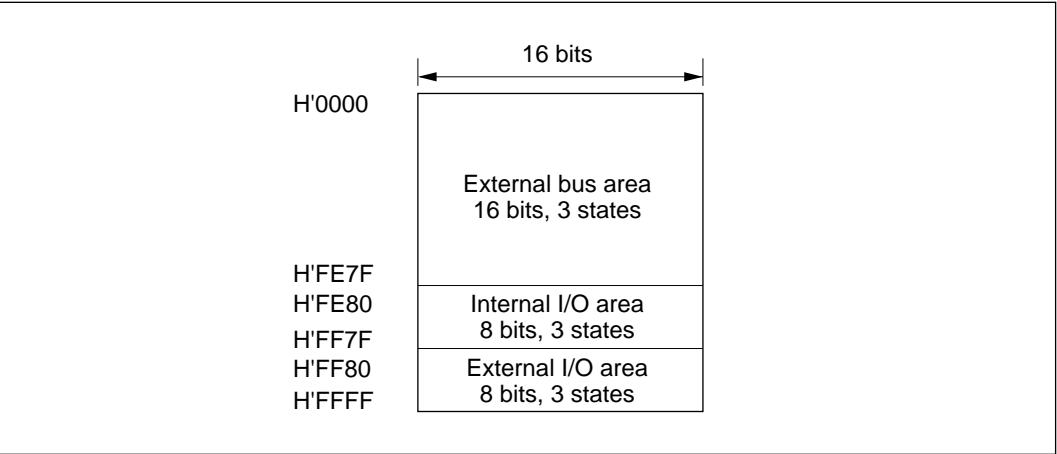


Figure 15-2 Bus Width and Cycle Length after Reset (Mode 2)

Mode 3: The bus is 8 bits wide. Addresses H'000000 to H'FFFFFF are all accessed at 8 bits per three states.

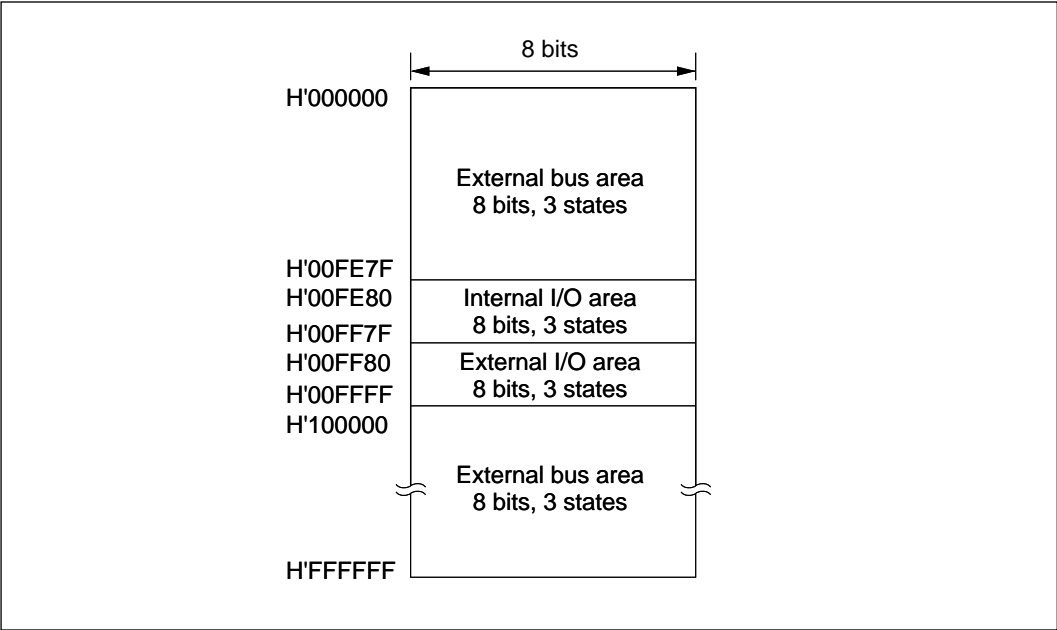


Figure 15-2 Bus Width and Cycle Length after Reset (Mode 3)

Mode 4: The bus is 16 bits wide. Addresses H'000000 to H'00FE7F and H'010000 to H'FEFFFF are accessed in three states via the 16-bit bus. Addresses H'00FE80 to H'00FF7F are the internal I/O area, accessed at 8 bits per three states. Addresses H'00FF80 to H'00FFFF are the external I/O area, also accessed at 8 bits per three states.

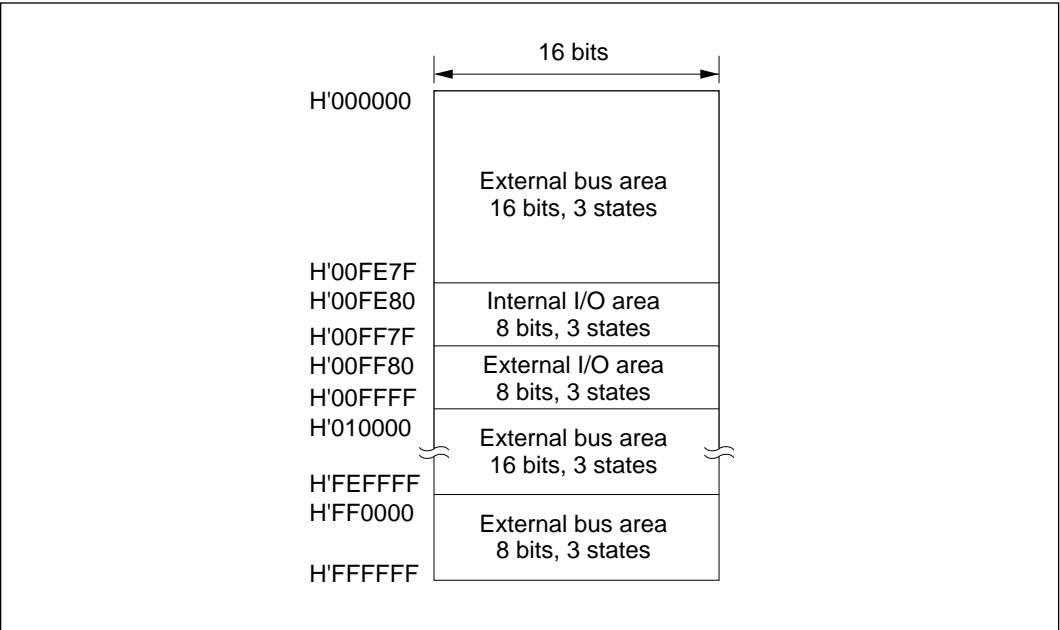


Figure 15-2 Bus Width and Cycle Length after Reset (Mode 4)

2. Timing of Changes in Bus Parameters: Changes in the bus width or bus cycle length take effect starting in the next bus cycle after the ARBT or AR3T write cycle.

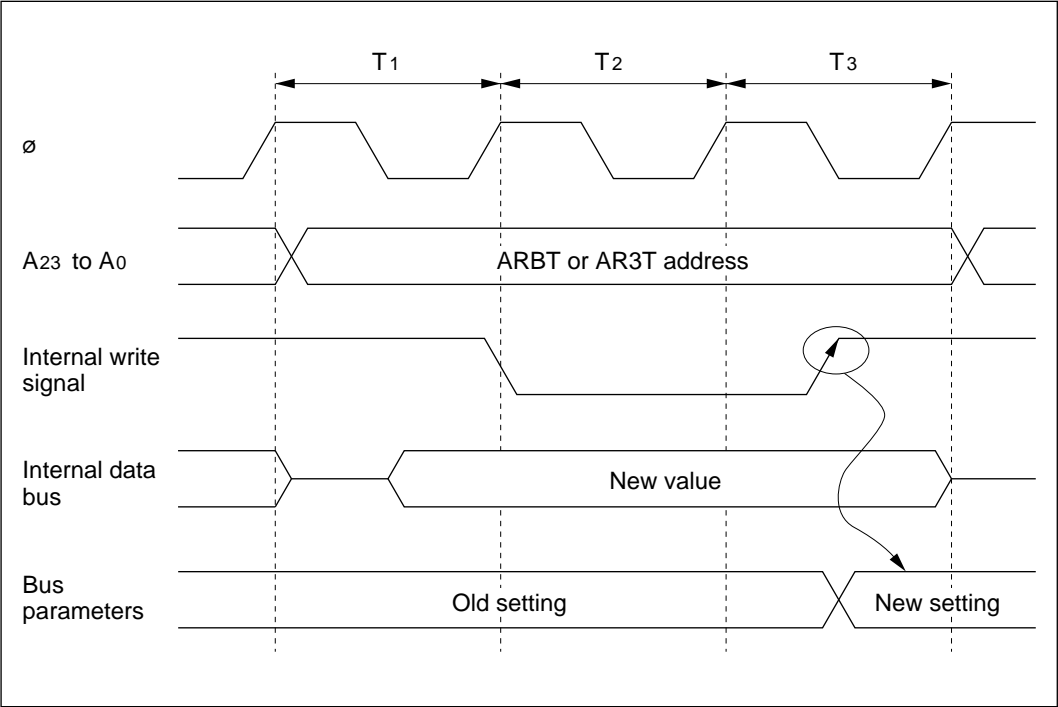


Figure 15-3 Time at which Bus Controller Setting Takes Effect (Byte Write)

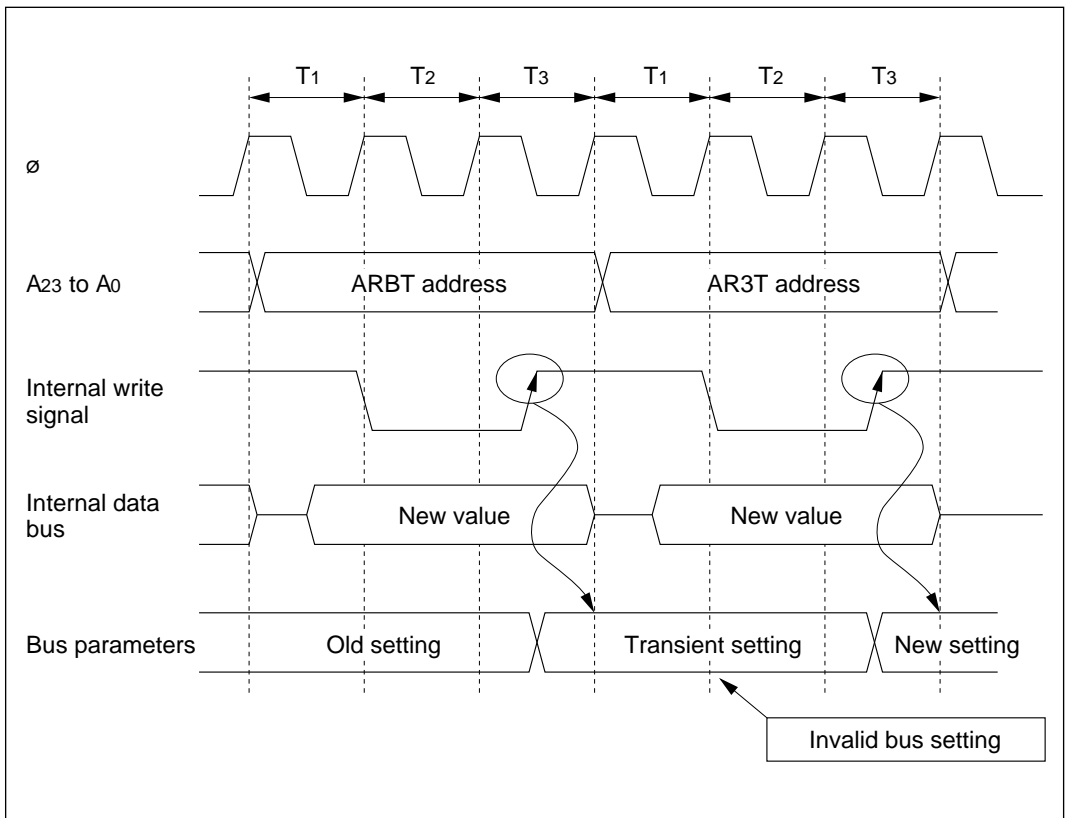


Figure 15-4 Time at which Bus Controller Setting Takes Effect (Word Write)

15.4 Notes and Precautions

When using the bus controller, note the following points.

Rewriting ARBT and AR3T: When ARBT and AR3T are rewritten, the bus parameters may become temporarily invalid, preventing normal program execution. This situation should be prevented as follows.

Solution: Place a branch instruction after any instruction that rewrites ARBT or AR3T. The branch instruction clears the instruction fetch performed using the temporarily invalid bus parameters, thereby preventing incorrect operation.

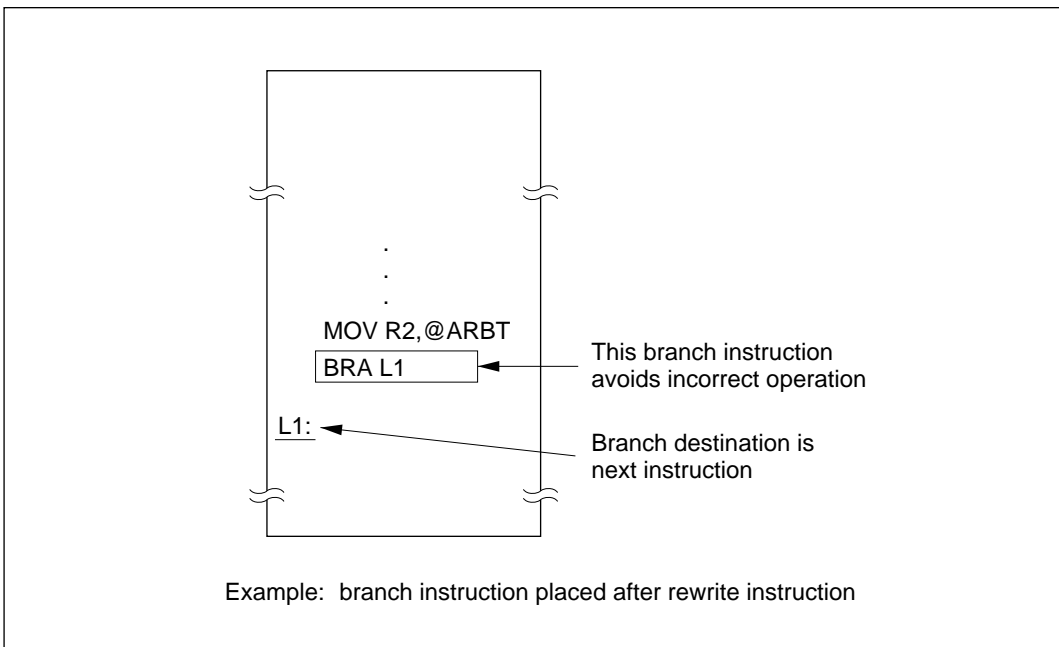


Figure 15-5 Example of Program that Rewrites ARBT or AR3T

Data Bus and Control Signals for Different Types of Access: The data bus and control signals vary depending on the type of access as indicated in table 15-2.

Table 15-2 Data Bus and Control Signals for Different Types of Access

Access Type	A0	Data Bus		Control Signals		
		D15 to D8	D7 to D0	\overline{RD}	\overline{HWR}	\overline{LWR}
8-bit bus	0	MSB	Not used	H	L	H
CPU → external address (write)	1	LSB	(I/O port)	H	L	H
8-bit bus	0	MSB	Not used	L	H	H
CPU ← external address (read)	1	LSB	(I/O port)	L	H	H
16-bit bus	0	MSB	LSB	H	L	L
Word area access	1	—	—	—	—	—
CPU ← external address (write)						
16-bit bus	0	MSB	LSB	L	H	H
Word area access	1	—	—	—	—	—
CPU ← external address (read)						
16-bit bus	0	MSB	Hi-Z	H	L	H
Byte area access	1	LSB	Hi-Z	H	L	H
CPU → external address (write)						
16-bit bus	0	MSB	Don't care	L	H	H
Byte area access	1	LSB	Don't care	L	H	H
CPU ← external address (read)						

Figures 15-6 and 15-7 show examples of bus controller settings made in mode 4.

1. $AR3T \leq ARBT + 1$

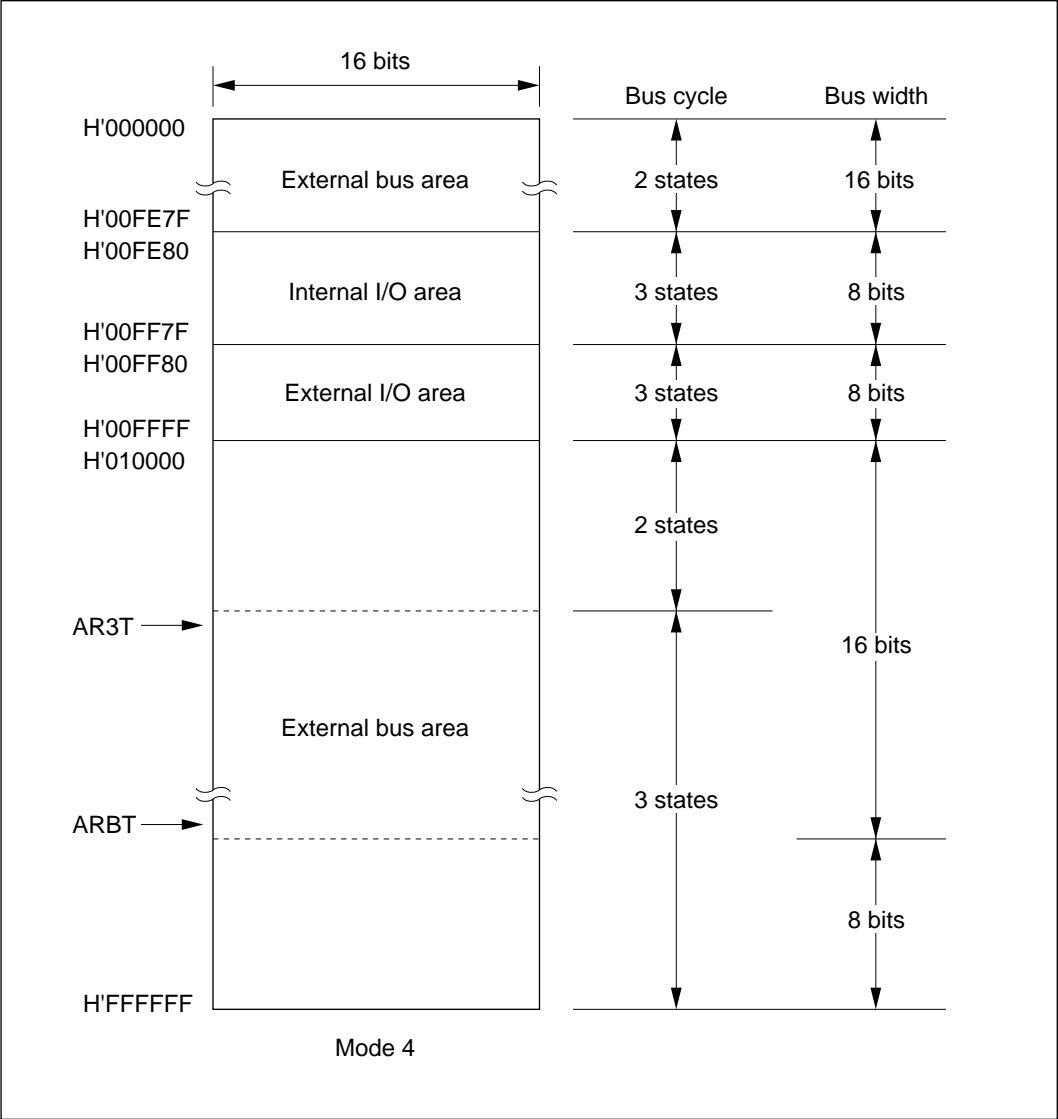


Figure 15-6 Example of Bus Controller Usage (Mode 4)

2. $AR3T > ARBT + 1$

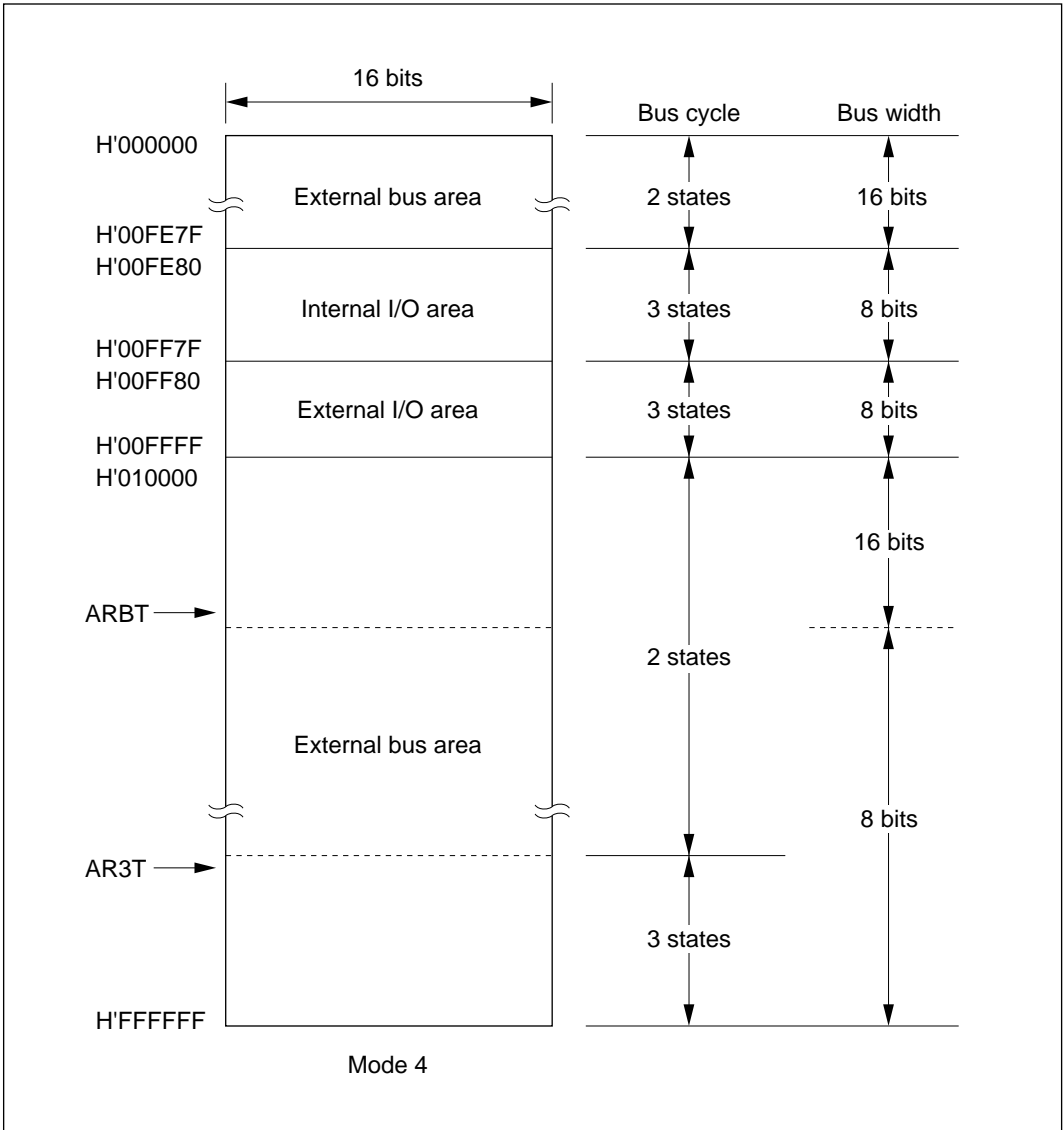


Figure 15-7 Example of Bus Controller Usage (Mode 4)

Section 16 Watchdog Timer

16.1 Overview

The H8/510 has an on-chip watchdog timer (WDT) module. This module can monitor system operation by generating a signal that resets the H8/510 chip if a system crash allows the timer count to overflow.

When this watchdog function is not needed, the WDT module can be used as an interval timer. In the interval timer mode, an IRQ0 interrupt is requested at each counter overflow.

The WDT module is also used in recovering from the software standby mode.

16.1.1 Features

The basic features of the watchdog timer module are summarized as follows:

- Selection of eight clock sources
- Selection of two modes: watchdog timer mode and interval timer mode
- Counter overflow generates a reset signal or interrupt request
Reset signal in the watchdog timer mode; IRQ0 request in the interval timer mode.
- External output of reset signal

The reset signal generated when the watchdog timer overflows resets the entire H8/510 chip. Depending on a reset output enable bit, the reset signal can also be output from the $\overline{\text{RES}}$ pin to reset devices controlled by the H8/510.

16.1.2 Block Diagram

Figure 16-1 is a block diagram of the watchdog timer.

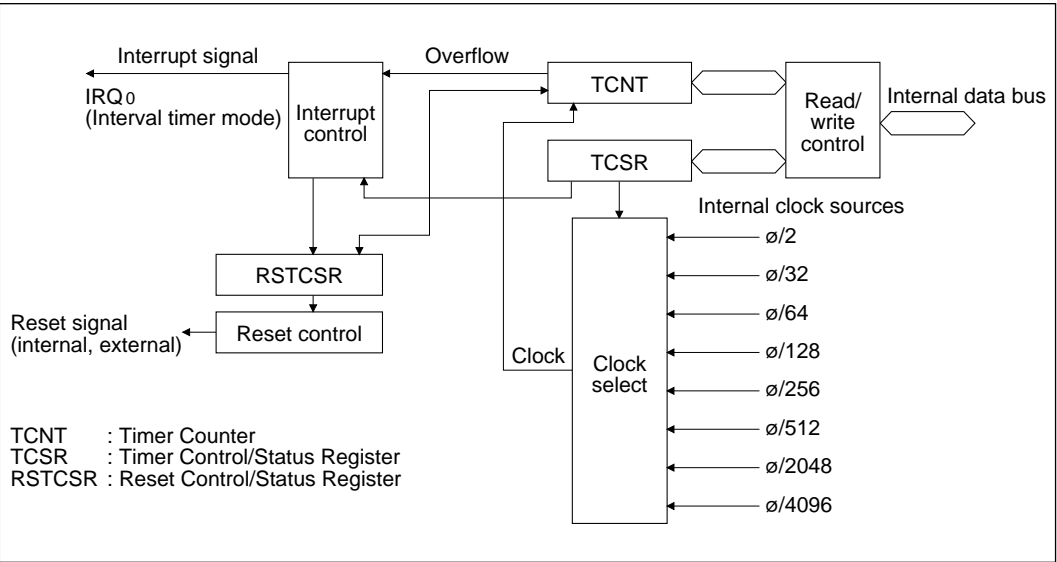


Figure 16-1 Block Diagram of Timer Counter

16.1.3 Register Configuration

Table 16-1 lists information on the watchdog timer registers.

Table 16-1 Register Configuration

Name	Abbreviation	R/W	Initial Value	Addresses	
				Write	Read
Timer control/status register	TCSR	R/(W)*	H'18	H'FF10	H'FF10
Timer counter	TCNT	R/W	H'00	H'FF10	H'FF11
Reset control/status register	RSTCSR	R/(W)*	H'3F	H'FF1E	H'FF1F

* Software can write a 0 to clear bit 7, but cannot write a 1.

16.2 Register Descriptions

16.2.1 Timer Counter TCNT—H'FF10 (Write), H'FF11 (Read)

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The watchdog timer counter (TCNT) is a readable/writable* 8-bit up-counter. When the timer enable bit (TME) in the timer control/status register (TCSR) is set to 1, the timer counter starts counting pulses of an internal clock source selected by clock select bits 2 to 0 (CKS2 to CKS0) in the TCSR. When the count overflows (changes from H'FF to H'00), the overflow flag (OVF) in the timer control/status register (TCSR) is set to 1.

The watchdog timer counter is initialized to H'00 at a reset and when the TME bit is cleared to 0.

* TCNT is write-protected by a password. See section 16.2.4, “Notes on Register Access” for details.

16.2.2 Timer Control/Status Register (TCSR)—H'FF10

Bit	7	6	5	4	3	2	1	0
	OVF	WT/IT	TME	—	—	CKS2	CKS1	CKS0
Initial value	0	0	0	1	1	0	0	0
Read/Write	R/(W)*1	R/W	R/W	—	—	R/W	R/W	R/W

The watchdog timer control/status register (TCSR) is an 8-bit readable/writable*2 register that selects the timer mode and clock source and performs other functions.

Bits 7 to 5 are initialized to 0 at a reset and in the standby modes. Bits 2 to 0 are initialized to 0 at a reset, but retain their values in the software standby mode.

Notes: *1 Software can write a 0 in bit 7 to clear the flag, but cannot set this bit to 1.

*2 The TCSR is write-protected by a password. See section 16.2.4, “Notes on Register Access” for details.

Bit 7—Overflow Flag (OVF): This bit indicates that the watchdog timer count has overflowed.

Bit 7

OVF	Description
0	This bit is cleared from 1 to 0 when the CPU reads (Initial value) the OVF bit after it has been set to 1, then writes a 0 in this bit.
1	This bit is set to 1 when TCNT changes from H'FF to H'00.

* The OVF bit is not set in the watchdog timer mode.

Bit 6—Timer Mode Select (WT/IT): This bit selects whether to operate in the watchdog timer mode or interval timer mode. If the watchdog timer mode is selected, a watchdog timer overflow resets the chip. If the interval timer mode is selected, a watchdog timer overflow generates an IRQ0 interrupt request.

Bit 6

WT/IT	Description
0	Interval timer mode (IRQ0 request) (Initial value)
1	Watchdog timer mode (Reset)

Bit 5—Timer Enable (TME): This bit enables or disables the timer.

Bit 5

TME	Description
0	TCNT is initialized to H'00 and stopped. (Initial value)
1	TCNT runs. A reset or interrupt is requested when the count overflows.

Bits 4 and 3—Reserved: These bits cannot be modified and are always read as 1.

Bits 2, 1, and 0—Clock Select (CKS2, CKS1, and CKS0): These bits select one of eight clock sources obtained by dividing the system clock (ϕ).

The overflow interval listed in the table below is the time from when the watchdog timer counter begins counting from H'00 until an overflow occurs.

In the interval timer mode, IRQ0 interrupts are requested at this interval.

Bit 2	Bit 1	Bit 0	Description	
CKS2	CKS1	CKS0	Clock Source	Overflow Interval ($\phi = 10\text{ MHz}$)
0	0	0	$\phi/2$	51.2 μs (Initial value)
0	0	1	$\phi/32$	819.2 μs
0	1	0	$\phi/64$	1.6 ms
0	1	1	$\phi/128$	3.3 ms
1	0	0	$\phi/256$	6.6 ms
1	0	1	$\phi/512$	13.1 ms
1	1	0	$\phi/2048$	52.4 ms
1	1	1	$\phi/4096$	104.9 ms

16.2.3 Reset Control/Status Register (RSTCSR)—H'FF1F (Read), H'FF1E (Write)

Bit	7	6	5	4	3	2	1	0
	WRST	RSTOE	—	—	—	—	—	—
Initial value	0	0	1	1	1	1	1	1
Read/Write	R/(W)*1	R/W	—	—	—	—	—	—

The reset control/status register (RSTCSR) is an 8-bit readable/writable*2 register that indicates when a reset has been caused by a watchdog timer overflow, and controls external output of the reset signal.

Bit 6 is not initialized by the reset caused by the watchdog timer overflow. It is initialized, however, by a reset caused by input at the $\overline{\text{RES}}$ pin.

Notes: *1 Software can write a 0 in bit 7 to clear the flag, but cannot set this bit to 1.

*2 RSTCSR is write-protected by a password. See section 16.2.4, “Notes on Register Access” for details.

Bit 7—Watchdog Timer Reset (WRST): This bit indicates that a reset signal has been generated by a watchdog timer overflow in the watchdog timer mode.

The reset signal generated by the overflow resets the entire H8/510 chip. In addition, if the reset output enable (RSTOE) bit is set to 1, a reset signal (Low) is output at the $\overline{\text{RES}}$ pin to reset devices connected to the H8/510.

The WRST bit can be cleared by software by writing a 0. It is also cleared when a reset signal from an external device is received at the $\overline{\text{RES}}$ pin.

Bit 7

WRST	Description
0	This bit is cleared to 0 by a reset signal input from the $\overline{\text{RES}}$ pin, or (Initial state) when software reads the WRST bit after it has been set to 1, then writes a 0 in this bit.
1	This bit is set to 1 when the watchdog timer overflows in the watchdog timer mode and an internal reset signal is generated.

Bit 6—Reset Output Enable (RSTOE): This bit selects whether the reset signal generated by a watchdog timer overflow in the watchdog timer mode is output from the $\overline{\text{RES}}$ pin.

Bit 6

RSTOE	Description
0	The reset signal generated by a watchdog timer overflow is not output to external devices. (Initial state)
1	The reset signal generated by a watchdog timer overflow is output to external devices.

Bit 5 to 0—Reserved: These bits cannot be modified and are always read as 1.

16.2.4 Notes on Register Access

The watchdog timer’s TCNT, TCSR, and RSTCSR registers differ from other registers in being more difficult to write. The procedures for writing and reading these registers are given below.

Writing to TCNT and TCSR: These registers must be written by word access. Programs cannot write to them by byte access. The word must contain the write data and a password.

The watchdog timer’s TCNT and TCSR registers both have the same write address. The write data must be contained in the lower byte of the word written at this address. The upper byte must contain H'5A (password for TCNT) or H'A5 (password for TCSR). See figure 16-2.

The result of the access depicted in figure 16-2 is to transfer the write data from the lower byte to TCNT or TCSR.

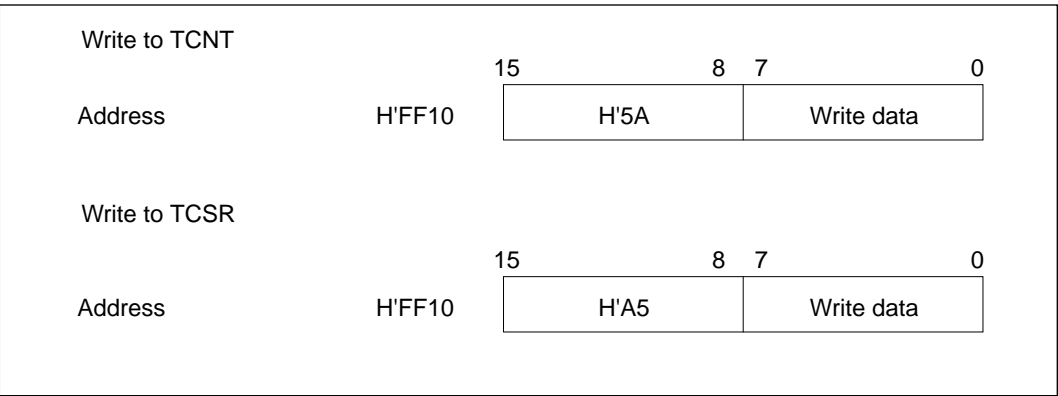


Figure 16-2 Writing to TCNT and TCSR

Writing to RSTCSR: RSTCSR must be written by moving word data to address H'FF1E. It cannot be written by byte access.

The upper byte of the word must contain a password. Separate passwords are used for clearing the WRST bit and for writing a 1 or 0 to the RSTOE bit.

To clear the WRST bit, the word written at address H'FFFE must contain the password H'A5 in the upper byte and the data H'00 in the lower byte. This clears the WRST bit to 0 without affecting other bits.

To set or clear the RSTOE bit, the word written at address H'FF1E must contain the password H'5A in the upper byte and the write data in the lower byte. This writes the desired data in the RSTOE bit without affecting other bits.

These write operations are illustrated in figure 16-3.

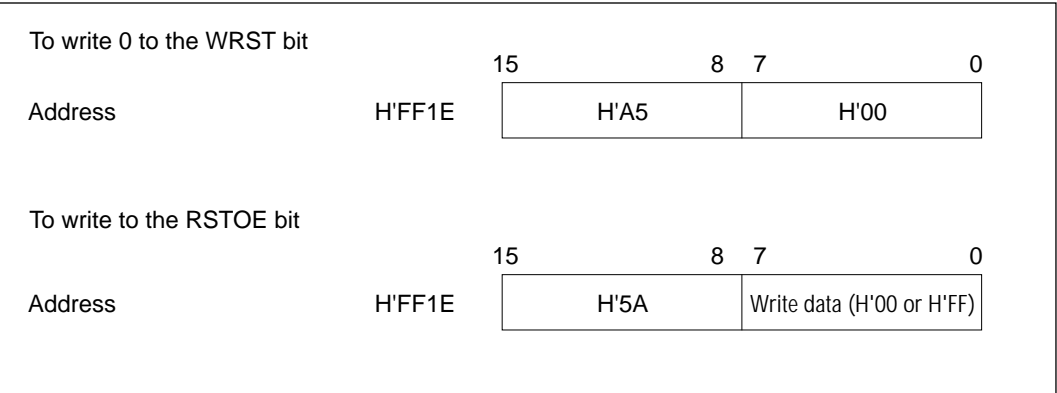


Figure 16-3 Writing to RSTCSR

Reading TCNT, TCSR, and RSTCSR: The read addresses are H'FF10 for TCSR, H'FF11 for TCNT, and H'FF1F for RSTCSR as indicated in table 16-2.

These three registers are read like other registers. Byte access instructions can be used.

Table 16-2 Read Addresses of TCNT and TCSR

Read Address	Register
H'FF10	TCSR
H'FF11	TCNT
H'FF1F	RSTCSR

16.3 Operation

16.3.1 Watchdog Timer Mode

The watchdog timer function begins operating when software sets the $\overline{WT/IT}$ and TME bits to 1 in the timer control/status register (TCSR). Thereafter, software should periodically rewrite the contents of the timer counter (normally by writing H'00) to prevent the count from overflowing. If a program crash allows the timer count to overflow, the watchdog timer generates a reset as shown in figure 16-4.

The reset signal from the watchdog timer can also be output from the \overline{RES} pin to reset external devices. This reset output signal is a Low pulse with a duration of 132 system clock cycles. The reset signal is output only if the RSTOE bit in the timer control/status register is set to 1.

The reset generated by the watchdog timer has the same vector as a reset generated by Low input at the RES pin. Software should check the WRST bit in the reset control/status register (RSTCSR) to determine the source of the reset.

If a watchdog timer overflow occurs at the same time as a Low input at the \overline{RES} pin, priority is given to one type of reset or the other depending on the value of the RSTOE bit in the reset control/status register.

If the RSTOE bit is set to 1 when both types of reset occur simultaneously, the watchdog timer's reset signal takes precedence. The internal state of the H8/510 chip is reset, the RSTOE bit remains set to 1, the WRST bit is also set to 1, and the \overline{RES} pin is held Low for 132 system clock cycles. If at the end of 520 system clock cycles there is still an external Low input to the \overline{RES} pin, the external reset takes effect, clearing the WRST and RSTOE bits to 0. Note that if the external reset occurs before the watchdog timer overflows, it takes effect immediately and clears the RSTOE bit.

If the RSTOE bit is cleared to 0 when both types of reset occur simultaneously, the reset signal input from the \overline{RES} pin takes precedence and the WRST bit is cleared to 0.

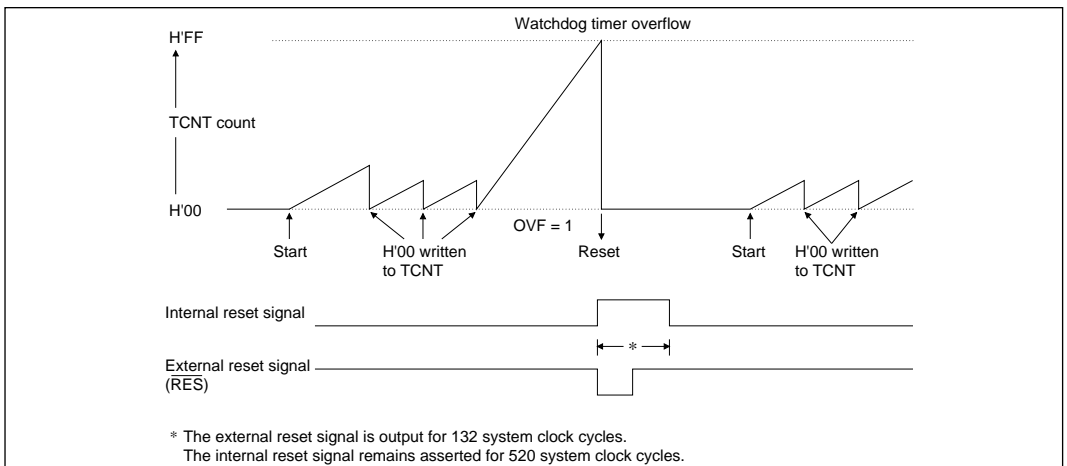


Figure 16-4 Operation in Watchdog Timer Mode

16.3.2 Interval Timer Mode

Interval timer operation begins when the $\overline{WT/IT}$ bit is cleared to 0 and the TME bit is set to 1.

In the interval timer mode, an IRQ0 request is generated each time the timer count overflows. This function can be used to generate IRQ0 requests at regular intervals. See figure 16-5.

IRQ0 requests from the watchdog timer module have a different vector as IRQ0 requests from the IRQ0 pin, so the IRQ0 interrupt-handling routine does not have to determine the source of the interrupt (which it could do by checking the OVF bit).

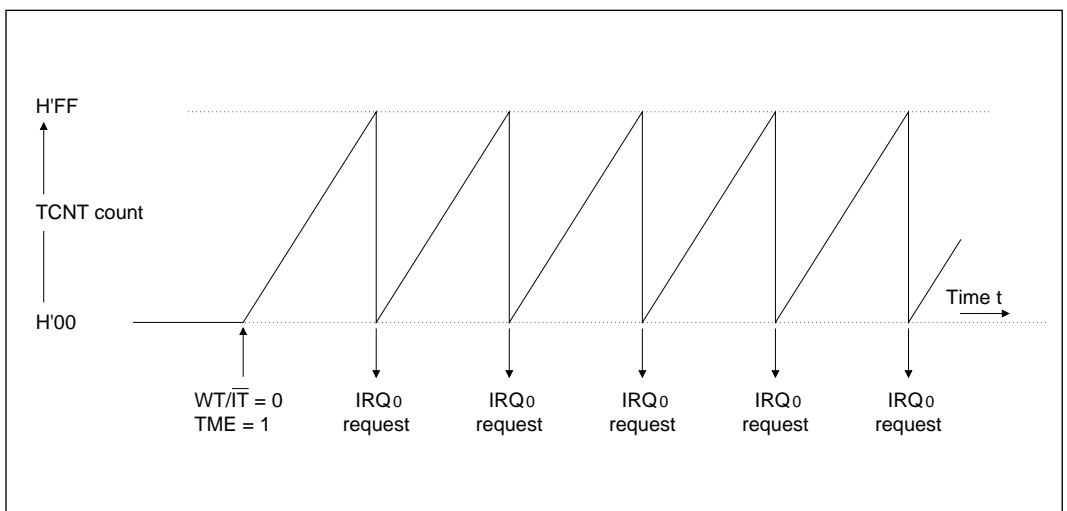


Figure 16-5 Operation in Interval Timer Mode

16.3.3 Operation in Software Standby Mode

The watchdog timer has a special function in the software standby mode. Specific watchdog timer settings are required when the software standby mode is used.

Before Transition to the Software Standby Mode: The TME bit must be cleared to 0 to stop the watchdog timer counter before a transition to the software standby mode. The chip cannot enter the software standby mode while the TME bit is set to 1. Before entering the software standby mode, software should also set the clock select bits (CKS2 to CKS0) to a value that makes the timer overflow interval equal to or greater than the settling time of the clock oscillator.

Recovery from the Software Standby Mode: Recovery from the software standby mode can be triggered by an NMI request. In this case the recovery proceeds as follows:

When an NMI request signal is received, the clock oscillator starts running and the watchdog timer starts counting at the rate selected by the clock select bits before the software standby mode was entered. When the count overflows from H'FF to H'00, the system clock is presumed to be stable and usable, clock signals are supplied to all modules on the chip, and the NMI interrupt-handling routine starts executing.

16.3.4 Setting of Overflow Flag

The OVF bit is set to 1 when the timer count overflows in interval timer mode. Simultaneously, the WDT module requests an IRQ0 interrupt. The timing is shown in figure 16-6.

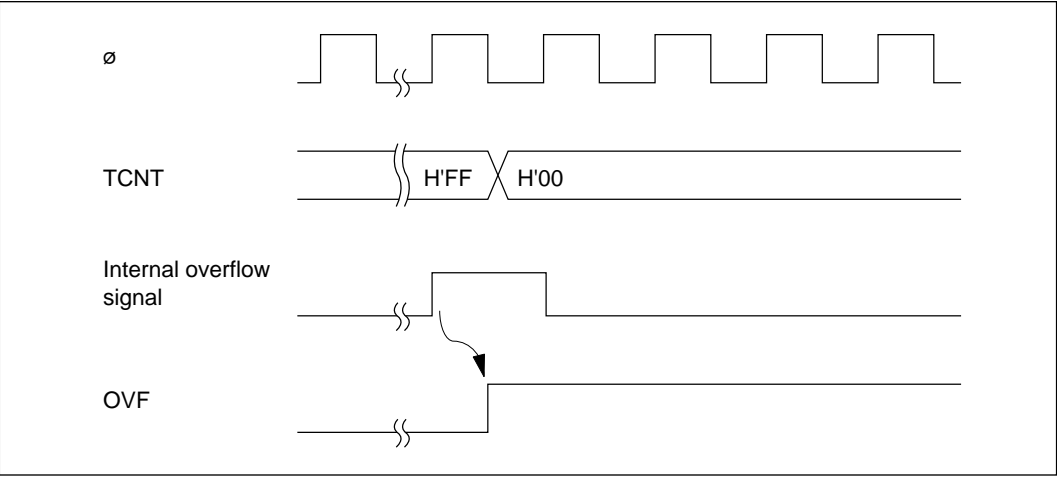


Figure 16-6 Setting of OVF Bit

16.3.5 Setting of Watchdog Timer Reset (WRST) Bit

The WRST bit is valid when $WT/\overline{IT} = 1$ and $TME = 1$.

The WRST bit is set to 1 when the timer count overflows. An internal reset signal is simultaneously generated for the entire H8/510 chip. The timing is shown in figure 16-7.

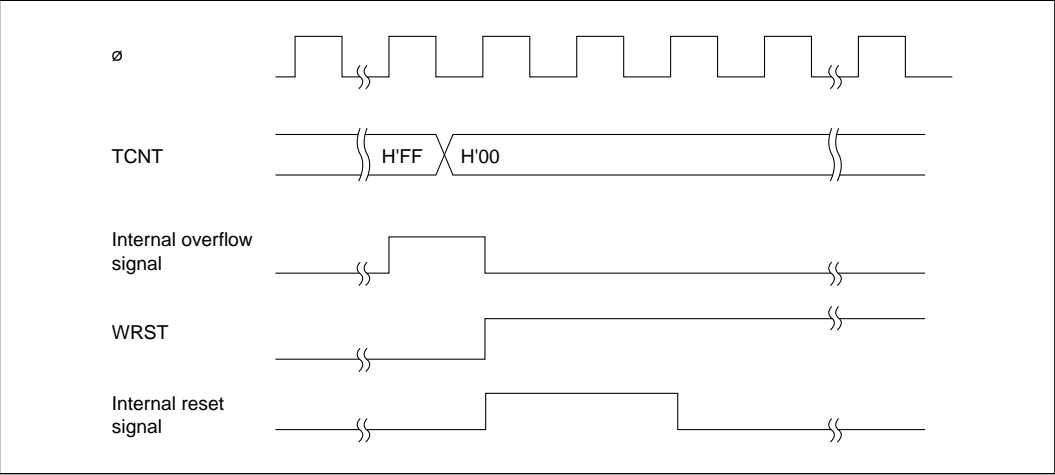


Figure 16-7 Setting of WRST Bit and Internal Reset Signal

16.4 Application Notes

Contention between TCNT Write and Increment: If a timer counter clock pulse is generated during the T3 state of a write cycle to the timer counter, the write takes priority and the timer counter is not incremented. See figure 16-8.

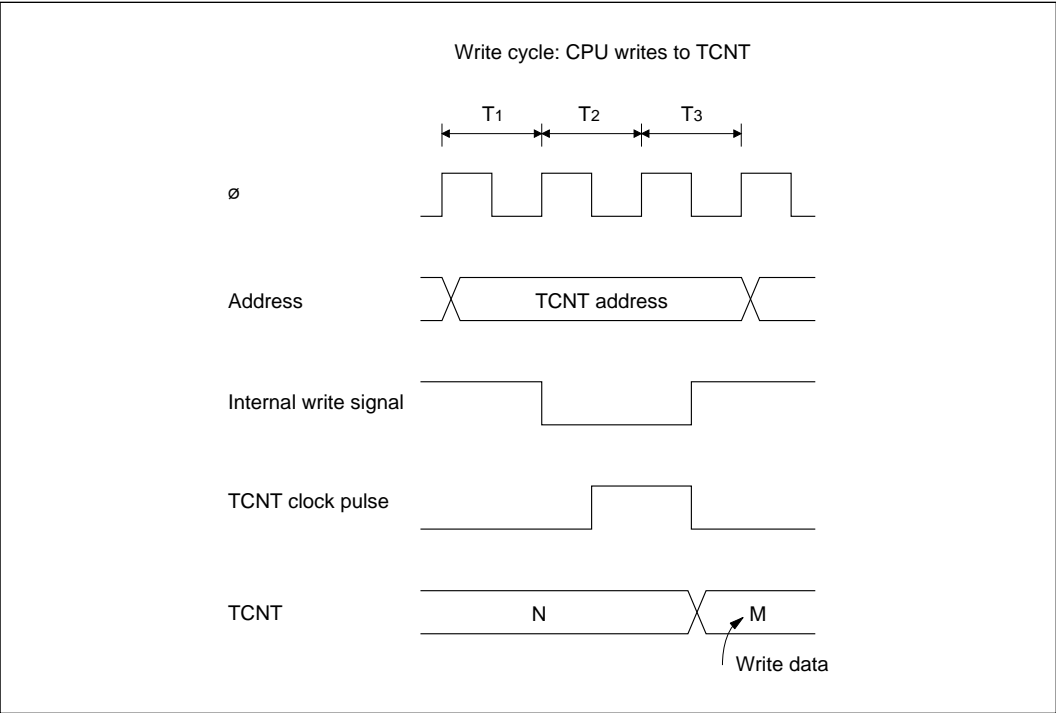


Figure 16-8 TCNT Write-Increment Contention

Changing the Clock Select Bits (CKS2 to CKS0): Software should stop the watchdog timer (by clearing the TME bit to 0) before changing the value of the clock select bits. If the clock select bits are modified while the watchdog timer is running, the timer count may be incremented incorrectly.

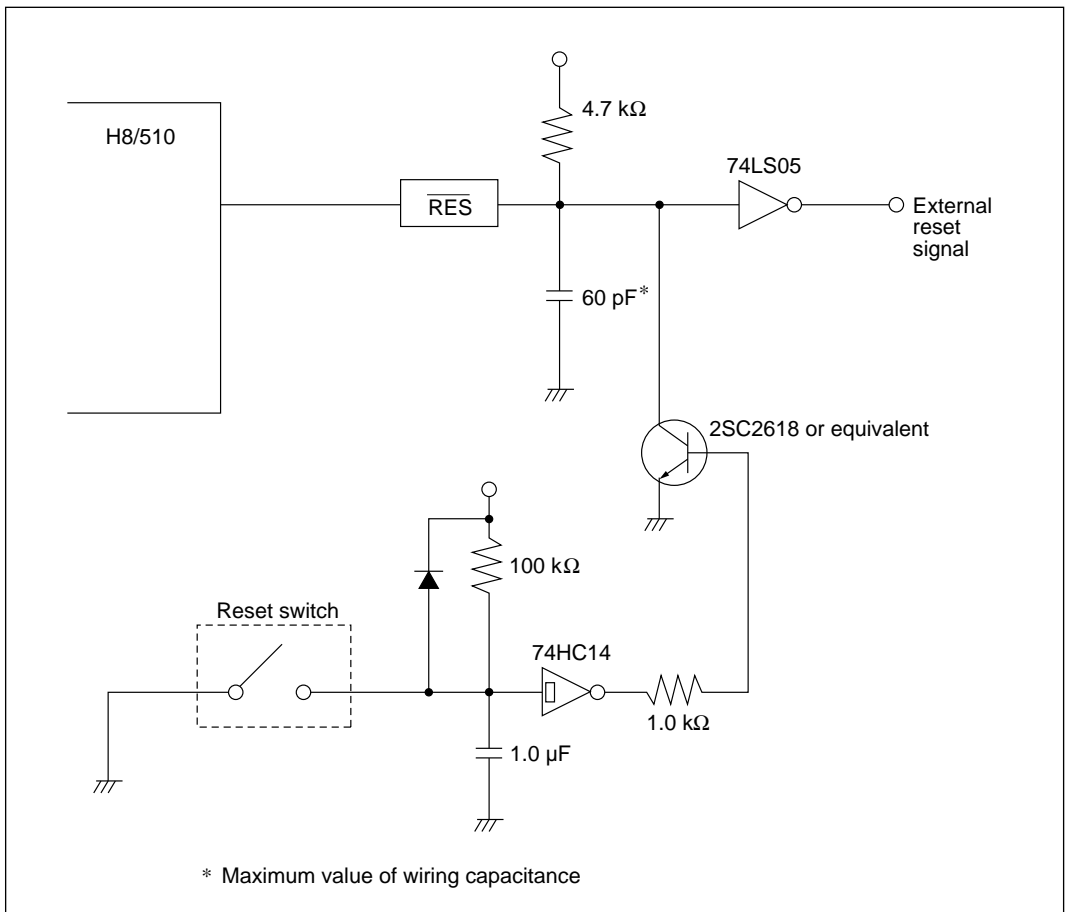


Figure 16-9 Reset Circuit (Example)

Section 17 Power-Down State

17.1 Overview

The H8/510 has a power-down state that greatly reduces power consumption by stopping the CPU functions. The power-down state includes three modes:

1. Sleep mode
2. Software standby mode
3. Hardware standby mode

The sleep mode and software standby mode are entered from the program execution state by executing the SLEEP instruction under the conditions given in table 17-1. The hardware standby mode is entered from any other state by a Low input at the STBY pin.

Table 17-1 lists the conditions for entering and leaving the power-down modes. It also indicates the status of the CPU, on-chip supporting modules, etc., in each power-down mode.

Table 17-1 Power-Down State

Mode	Entering Procedure	Clock	CPU	CPU Reg's.	Sup. Mod's.	I/O Ports	Exiting Methods
Sleep mode	Execute SLEEP instruction	Run	Halt	Held	Run	Held	<ul style="list-style-type: none"> • Interrupt • $\overline{\text{RES}}$ Low • $\overline{\text{STBY}}$ Low
Software standby mode	Set SSBY bit in SBYCR to 1, then execute SLEEP instruction*	Halt	Halt	Held	Halt and partly initialized	Held	<ul style="list-style-type: none"> • NMI • $\overline{\text{RES}}$ Low • $\overline{\text{STBY}}$ Low
Hardware standby mode	Set STBY pin to Low level	Halt	Halt	Not held	Halt and initialized	High impedance state	<ul style="list-style-type: none"> • $\overline{\text{STBY}}$ High, then $\overline{\text{RES}}$ Low → High

* The watchdog timer must also be stopped.

Notes: SBYCR: Software standby control register
SSBY: Software standby bit

17.2 Sleep Mode

17.2.1 Transition to Sleep Mode

Execution of the SLEEP instruction causes a transition from the program execution state to the sleep mode. After executing the SLEEP instruction, the CPU halts, but the contents of its internal registers remain unchanged. The functions of the on-chip supporting modules do not stop in the sleep mode.

17.2.2 Exit from Sleep Mode

The chip wakes up from the sleep mode when it receives an internal or external interrupt request, or a Low input at the $\overline{\text{RES}}$ or $\overline{\text{STBY}}$ pin.

Wake-Up by Interrupt: An interrupt releases the sleep mode and starts either the CPU's interrupt-handling sequence or the data transfer controller (DTC).

If the interrupt is served by the DTC, after the data transfer is completed the CPU executes the instruction following the SLEEP instruction, unless the count in the data transfer count register (DTCR) is 0.

If an interrupt on a level equal to or less than the mask level in the CPU's status register (SR) is requested, the interrupt is left pending and the sleep mode continues. Also, if an interrupt from an on-chip supporting module is disabled by the corresponding enable/disable bit in the module's control register, the interrupt cannot be requested, so it cannot wake the chip up.

Wake-Up by $\overline{\text{RES}}$ Pin: When the $\overline{\text{RES}}$ pin goes Low, the chip exits from the sleep mode to the reset state.

Wake-Up by $\overline{\text{STBY}}$ Pin: When the $\overline{\text{STBY}}$ pin goes Low, the chip exits from the sleep mode to the hardware standby mode.

17.3 Software Standby Mode

17.3.1 Transition to Software Standby Mode

A program enters the software standby mode by setting the standby bit (SSBY) in the software standby control register (SBYCR) to 1, then executing the SLEEP instruction. Table 17-2 lists the attributes of the software standby control register.

Table 17-2 Software Standby Control Register

Name	Abbreviation	R/W	Initial Value	Address
Software standby control register	SBYCR	R/W	H'7F	H'FF1A

In the software standby mode, the CPU, clock, and the on-chip supporting module functions all stop, reducing power consumption to an extremely low level. The on-chip supporting modules and their registers are reset to their initial state, but as long as a minimum necessary voltage supply is maintained (at least 2 V), the contents of the CPU registers remain unchanged. The I/O ports also remain in their current states.

17.3.2 Software Standby Control Register (SBYCR)

Bit	7	6	5	4	3	2	1	0
	SSBY	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

The software standby control register (SBYCR) is an 8-bit register that controls the action of the SLEEP instruction.

Bit 7—Software Standby (SSBY): This bit enables or disables the transition to the software standby mode.

Bit 7

SSBY	Description
0	The SLEEP instruction causes a transition to the sleep mode. (Initial value)
1	The SLEEP instruction causes a transition to the software standby mode.

The watchdog timer must be stopped before the chip can enter the software standby mode. To stop the watchdog timer, clear the timer enable bit (TME) in the watchdog timer's timer control/status register (TCSR) to 0. The SSBY bit cannot be set to 1 while the TME bit is set to 1.

When the chip is recovered from the software standby mode by a nonmaskable interrupt (NMI), the SSBY bit is automatically cleared to 0. It is also cleared to 0 by a reset or transition to the hardware standby mode.

Bits 6 to 0—Reserved: These bits cannot be modified and are always read as 1.

17.3.3 Exit from Software Standby Mode

The chip can be brought out of the software standby mode by an input at one of three pins: the NMI pin, $\overline{\text{RES}}$ pin, or $\overline{\text{STBY}}$ pin.

Recovery by NMI Pin: When an NMI request signal is received, the clock oscillator begins operating but clock pulses are supplied only to the watchdog timer (WDT). The watchdog timer begins counting from H'00 at the rate determined by the clock select bits (CKS2 to CKS0) in its timer status/control register (TCSR). This rate should be set slow enough to allow the clock oscillator to stabilize before the count reaches H'FF. When the count overflows from H'FF to H'00, clock pulses are supplied to the whole chip, the software standby mode ends, and execution of the NMI interrupt-handling sequence begins.

The clock select bits (CKS2 to CKS0) should be set as follows.

- **Crystal Oscillator:** Set CKS2 to CKS0 to a value that makes the watchdog timer interval equal to or greater than 10ms, which is the clock stabilization time.
- **External Clock Input:** CKS2 to CKS0 can be set to any value. The minimum value (CKS2 = CKS1 = CKS0 = 0) is recommended.

Recovery by $\overline{\text{RES}}$ Pin: When the $\overline{\text{RES}}$ pin goes Low, the clock oscillator starts. Next, when the RES pin goes High, the CPU begins executing the reset sequence.

When the chip recovers from the software standby mode by a reset, clock pulses are supplied to the entire chip at once. Be sure to hold the RES pin Low long enough for the clock to stabilize.

Recovery by $\overline{\text{STBY}}$ Pin: When $\overline{\text{STBY}}$ the pin goes Low, the chip exits from the software standby mode to the hardware standby mode.

17.3.4 Sample Application of Software Standby Mode

In this example the chip enters the software standby mode on the falling edge of the NMI input and recovers from the software standby mode on the rising edge of NMI. Figure 17-1 shows a timing chart of the transitions.

The nonmaskable interrupt edge bit (NMIEG) in the NMI control register (NMICR) is originally cleared to 0, selecting the falling edge as the NMI trigger. After accepting an NMI interrupt in this condition, software changes the NMIEG bit to 1, sets the SSBY bit to 1, and executes the SLEEP instruction to enter the software standby mode. The chip recovers from the software standby mode on the next rising edge at the NMI pin.

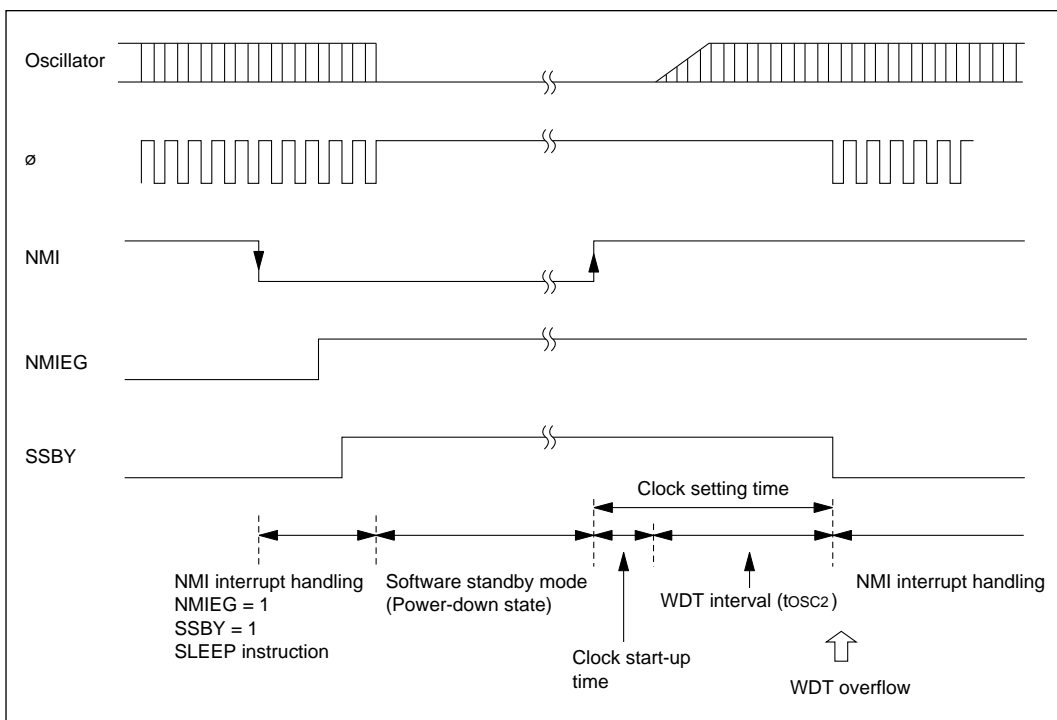


Figure 17-1 NMI Timing of Software Standby Mode (Application Example)

17.3.5 Application Notes

The I/O ports remain in their current states in the software standby mode. If a port is in the High output state, the output current is not reduced in the software standby mode.

17.4 Hardware Standby Mode

17.4.1 Transition to Hardware Standby Mode

Regardless of its current state, the chip enters the hardware standby mode whenever the $\overline{\text{STBY}}$ pin goes Low.

The hardware standby mode reduces power consumption drastically by halting the CPU, stopping all the functions of the on-chip supporting modules, and placing I/O ports in the high-impedance state. The on-chip supporting modules go into the reset state.

Note: Do not change the inputs at the mode pins (MD2, MD1, MD0) during hardware standby mode. Be particularly careful not to let all three mode inputs go Low, since that would cause increased current dissipation.

17.4.2 Recovery from Hardware Standby Mode

Recovery from the hardware standby mode requires inputs at both the $\overline{\text{STBY}}$ and $\overline{\text{RES}}$ pins.

When the $\overline{\text{STBY}}$ pin goes High, the clock oscillator begins running. The $\overline{\text{RES}}$ pin should be Low at this time and should be held Low long enough for the clock to stabilize. When the $\overline{\text{RES}}$ pin changes from Low to High, the reset sequence is executed and the chip returns to the program execution state.

17.4.3 Timing Sequence of Hardware Standby Mode

Figure 17-2 shows the usual sequence for entering and leaving the hardware standby mode.

First the $\overline{\text{RES}}$ pin goes Low, placing the chip in the reset state. Then the $\overline{\text{STBY}}$ pin goes Low, placing the chip in the hardware standby mode and stopping the clock. In the recovery sequence first the $\overline{\text{STBY}}$ pin goes High; then after the clock stabilizes, the $\overline{\text{RES}}$ pin is returned to the High level.

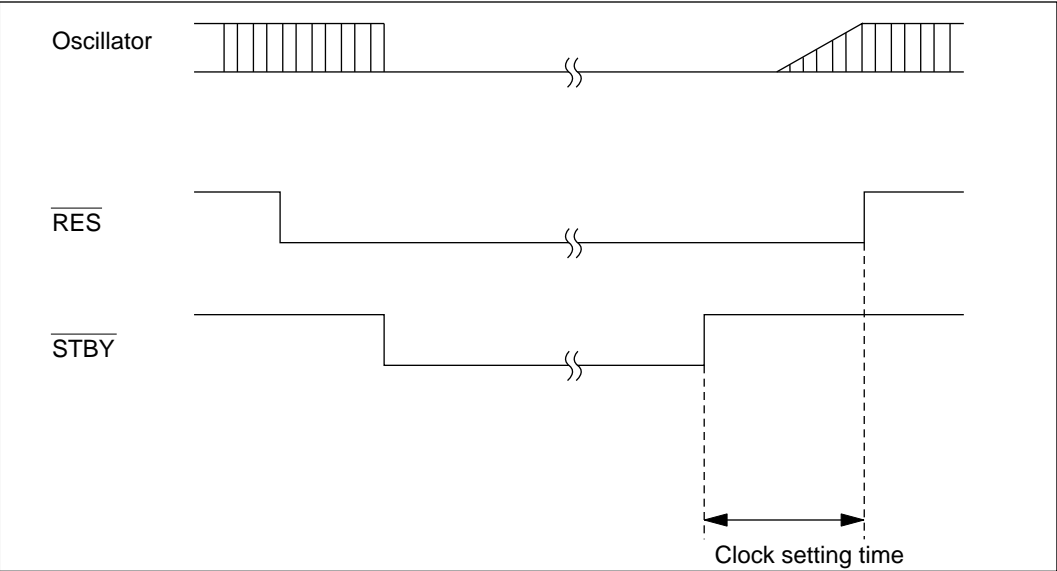


Figure 17-2 Hardware Standby Sequence

Section 18 E Clock Interface

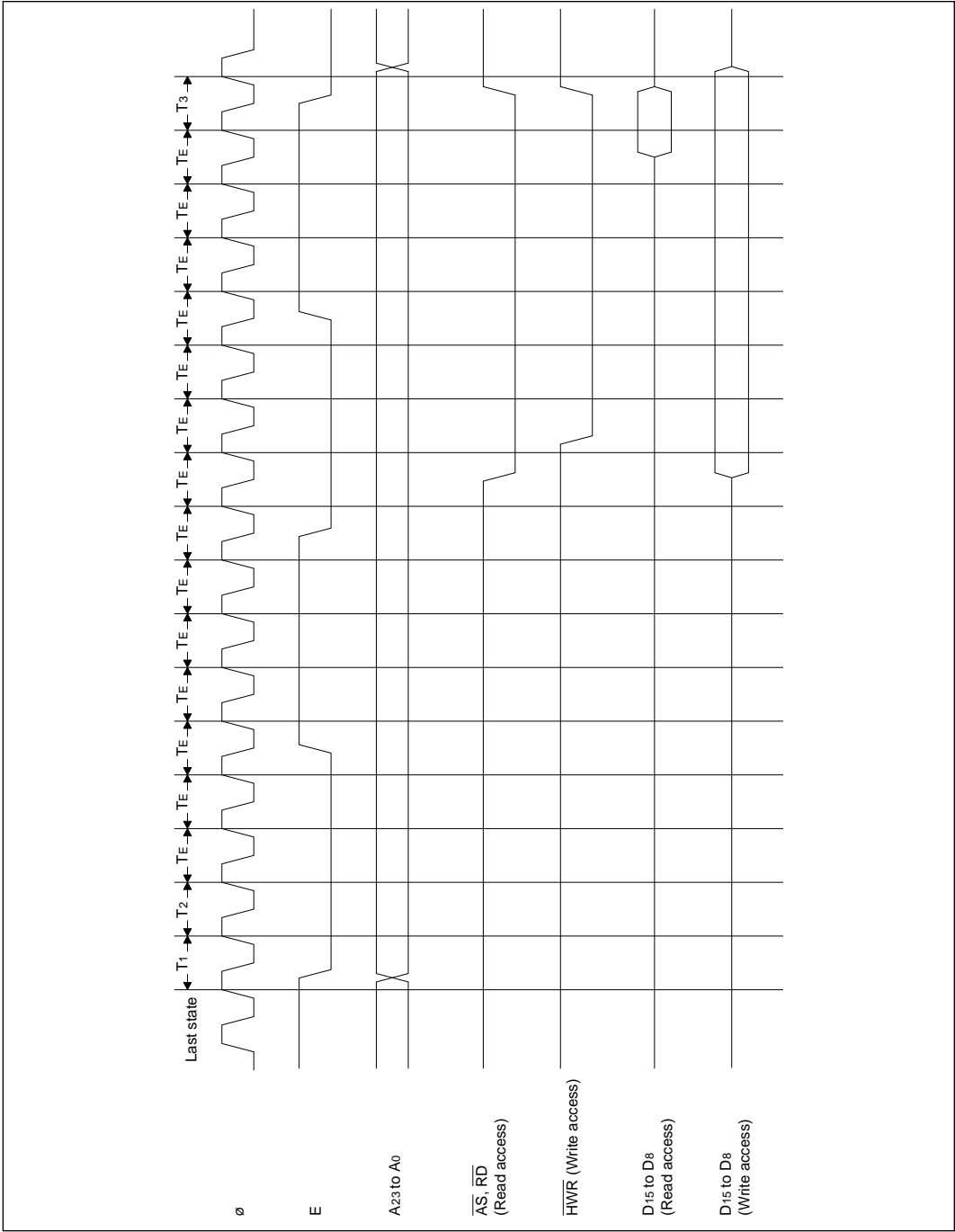
18.1 Overview

For interfacing to E clock based peripheral devices, the H8/510 can generate an E clock output. Special instructions (MOVTPE, MOVFPE) perform data transfers synchronized with the E clock.

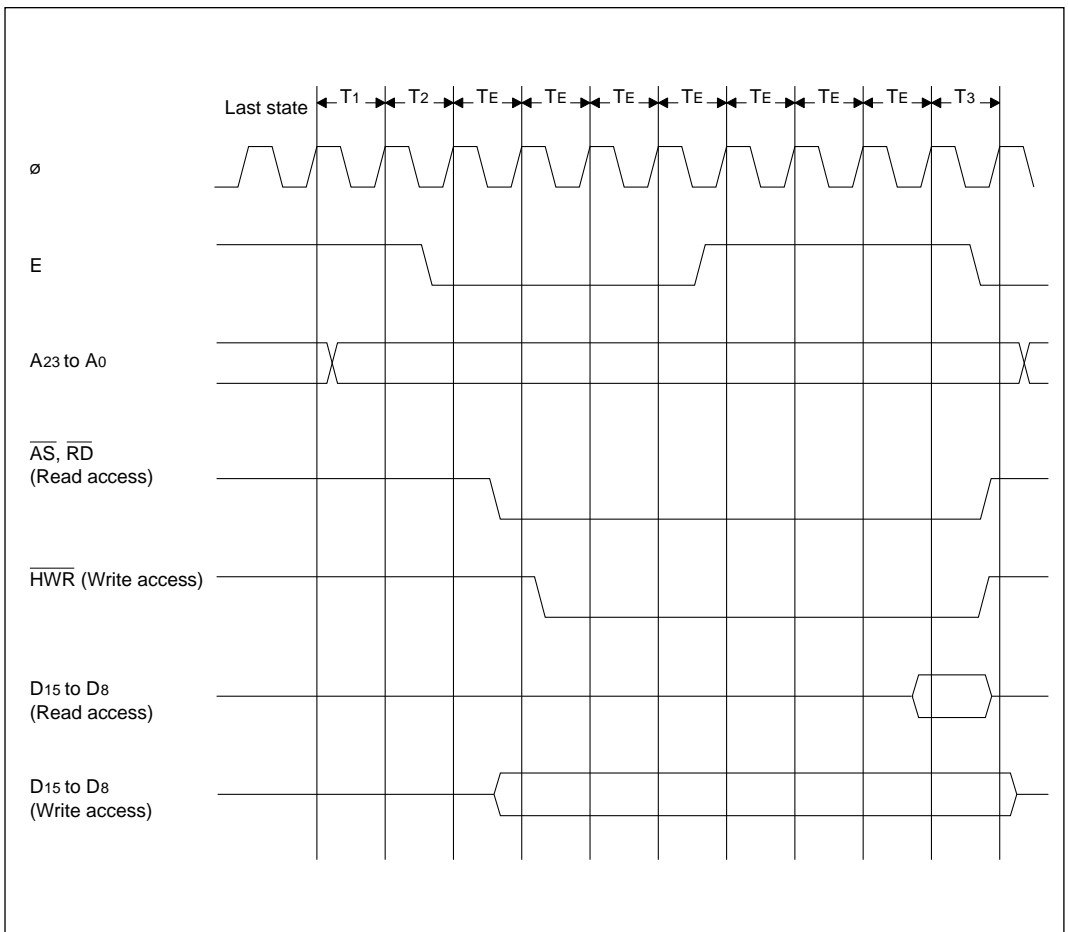
The E clock is created by dividing the system clock (ϕ) by 8.

When the CPU executes an instruction that synchronizes with the E clock, the address is output on the address bus as usual, but the data bus and the \overline{RD} and \overline{HWR} signal lines do not become active until the falling edge of the E clock is detected. The length of the access cycle for an instruction synchronized with the E clock is accordingly variable. Figures 18-1 and 18-2 show the timing in the cases of maximum and minimum synchronization delay.

The wait state controller (WSC) does not insert any wait states (T_w) during the execution of an instruction synchronized with the E clock.



**Figure 18-1 Execution Cycle of Instruction Synchronized with E Clock
(Maximum Synchronization Delay)**



**Figure 18-2 Execution Cycle of Instruction Synchronized with E Clock
(Minimum Synchronization Delay)**

Section 19 Electrical Specifications

19.1 Absolute Maximum Ratings

Table 19-1 lists the absolute maximum ratings.

Table 19-1 Absolute Maximum Ratings

Item	Symbol	Rating	Unit
Supply voltage	V _{CC}	−0.3 to +7.0	V
Input voltage (except port 7)	V _{in}	−0.3 to V _{CC} + 0.3	V
(port 7)	V _{in}	−0.3 to AV _{CC} + 0.3	V
Analog supply voltage	AV _{CC}	−0.3 to +7.0	V
Analog input voltage	VA _N	−0.3 to AV _{CC} + 0.3	V
Operating temperature	T _{opr}	Regular specifications: −20 to +75	°C
		Wide-range specifications: −40 to +85	°C
Storage temperature	T _{stg}	−55 to +125	°C

Note: Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions.

19.2 Electrical Characteristics

19.2.1 DC Characteristics

Table 19-2 lists the DC characteristics.

Table 19-2 DC Characteristics

Conditions: $V_{CC} = 5.0\text{ V} \pm 10\%$, $AV_{CC} = 5.0\text{ V} \pm 10\%$ *1, $V_{SS} = AV_{SS} = 0\text{ V}$,
 $T_a = -20\text{ to }+75^\circ\text{C}$ (Regular Specifications)
 $T_a = -40\text{ to }+85^\circ\text{C}$ (Wide-Range Specifications)

Item		Sym- bol	Min	Typ	Max	Unit	Measurement Conditions
Input High voltage	RES, STBY, EXTAL	V_{IH}	$V_{CC} - 0.7$	—	$V_{CC} + 0.3$	V	
	Port 7		$V_{CC} \times 0.7$	—	$V_{CC} + 0.3$	V	
	Other input pins (except port 4)		2.2	—	$AV_{CC} + 0.3$	V	
			2.2	—	$V_{CC} + 0.3$	V	
Input Low voltage	RES, STBY, MD2, MD1, MD0	V_{IL}	−0.3	—	0.5	V	
	Other input pins (except port 4)		−0.3	—	0.8	V	
Schmitt trigger input voltage	Port 4	V_{T-}	1.0	—	2.5	V	
		V_{T+}	2.0	—	3.5	V	
		$V_{T+} - V_{T-}$	0.4	—	—	V	
Input leakage current	RES	$ I_{in} $	—	—	10.0	μA	$V_{in} = 0.5\text{ to}$
	STBY, NMI, MD2, MD1, MD0, port 7		—	—	1.0	μA	$V_{CC} - 0.5\text{ V}$
Leakage current in 3-state (off state)	Port 8, ports 6 to 1	$ I_{TSI} $	—	—	1.0	μA	$V_{in} = 0.5\text{ to}$ $V_{CC} - 0.5\text{ V}$
Output High voltage	All output pins	V_{OH}	$V_{CC} - 0.5$	—	—	V	$I_{OH} = -200\text{ }\mu\text{A}$
			3.5	—	—	V	$I_{OH} = -1\text{ mA}$
Output Low voltage	All output pins	V_{OL}	—	—	0.4	V	$I_{OL} = 1.6\text{ mA}$
	RES		—	—	0.4	V	$I_{OL} = 2.6\text{ mA}$
Input capacitance	All input pins	C_{in}	—	—	20	pF	$V_{in} = 0\text{ V}$ $f = 1\text{ MHz}$ $T_a = 25^\circ\text{C}$

Note: *1 AV_{CC} must be connected to the power supply even when the A/D converter is not used.

Table 19-2 DC Characteristics (cont)

Item		Sym- bol	Min	Typ	Max	Unit	Measurement Conditions
Current dissipation*1	Normal operation	I _{CC}	—	15	30	mA	f = 6 MHz
			—	20	40	mA	f = 8 MHz
			—	25	50	mA	f = 10 MHz
	Sleep mode		—	9	20	mA	f = 6 MHz
			—	12	25	mA	f = 8 MHz
			—	15	30	mA	f = 10 MHz
	Standby		—	0.01	5.0	μA	T _a ≤ 50°C
			—	—	20.0	μA	T _a > 50°C
Analog supply current	During A/D conversion	A _I CC	—	1.0	2.0	mA	
	While waiting		—	0.01	5.0	μA	

Note: *1 Current dissipation values assume that V_{IH} min = V_{CC} – 0.5 V, V_{IL} max = 0.5 V and all output pins are in the no-load state.

Conditions: V_{CC} = 3.0 V to 5.5 V, V_{SS} = AV_{SS} = 0 V, T_a = –20 to +75°C (Regular Specifications)
AV_{CC} = 5.0 V ±10%*1

Item		Sym- bol	Min	Typ	Max	Unit	Measurement Conditions
Input High voltage	RES, STBY,	V _{IH}	V _{CC} × 0.9	—	V _{CC} + 0.3	V	
	EXTAL		V _{CC} × 0.9	—	V _{CC} + 0.3	V	
	Port 7		V _{CC} × 0.7	—	AV _{CC} + 0.3	V	
	Other input pins (except port 4)		V _{CC} × 0.7	—	V _{CC} + 0.3	V	
Input Low voltage	RES, STBY,	V _{IL}	–0.3	—	V _{CC} × 0.1	V	
	MD2, MD1, MD0						
	Other input pins (except port 4)		–0.3	—	V _{CC} × 0.15	V	
Schmitt trigger input voltage	Port 4	V _T [–]	V _{CC} × 0.2	—	V _{CC} × 0.6	V	
		V _T ⁺	V _{CC} × 0.4	—	V _{CC} × 0.7	V	
		V _T ⁺ –V _T [–]	V _{CC} × 0.07	—	—	V	
Input leakage current	RES	I _{in}	—	—	10.0	μA	V _{in} = 0.5 to
	STBY, NMI, MD2, MD1, MD0, port 7		—	—	1.0	μA	V _{CC} – 0.5 V

Note: *1 AV_{CC} must be connected to the power supply even when the A/D converter is not used.

Table 19-2 DC Characteristics (cont)

Item		Sym- bol	Min	Typ	Max	Unit	Measurement Conditions
Leakage current in 3-state (off state)	Port 8, ports 6 to 1	I _{TSI}	–	–	1.0	μA	V _{in} = 0.5 to V _{CC} – 0.5 V
Output High voltage	All output pins	V _{OH}	V _{CC} – 0.4	–	–	V	I _{OH} = –200 μA
			V _{CC} – 0.9	–	–	V	I _{OH} = –1 mA
Output Low voltage	All output pins	V _{OL}	–	–	0.4	V	I _{OL} = 1.6 mA
	RES		–	–	0.4	V	I _{OL} = 2.6 mA
Input capacitance	All input pins	C _{in}	–	–	20	pF	V _{in} = 0 V f = 1 MHz T _a = 25°C
Current dissipation*1	Normal operation	I _{CC}	–	20	28	mA	f = 5 MHz
	Sleep mode		–	12	18	mA	f = 5 MHz
	Standby		–	0.01	5.0	μA	T _a ≤ 50°C
			–	–	20.0	μA	T _a > 50°C
Analog supply current	During A/D conversion	A _{ICC}	–	1.0	2.0	mA	
	While waiting		–	0.01	5.0	μA	

Note: *1 Current dissipation values assume that V_{IH} min = V_{CC} × 0.9 V, V_{IL} max = 0.3 V and all output pins are in the no-load state.

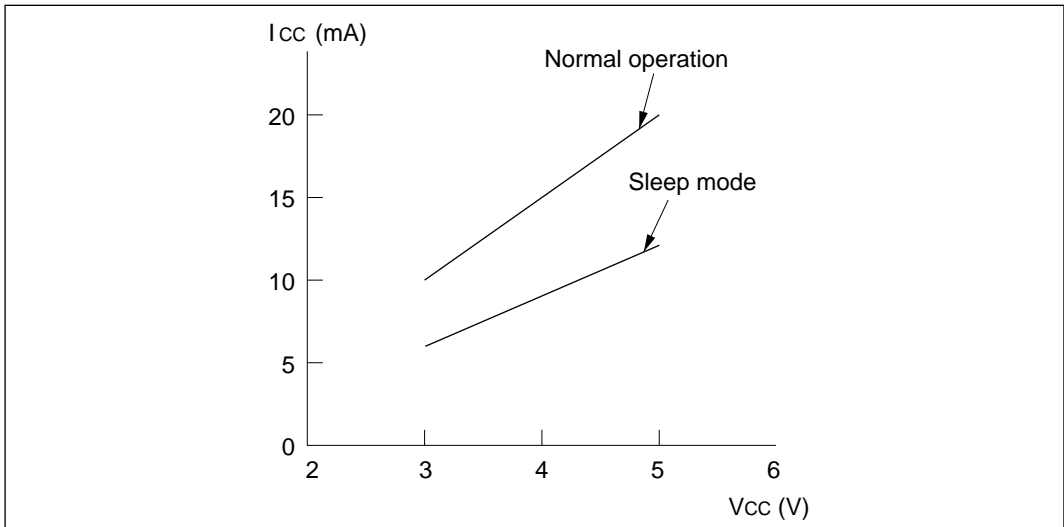


Figure 19-1 Relation between I_{CC} and V_{CC}

Table 19-3 Allowable Output Current Sink Values

Conditions: $V_{CC} = 5.0 \text{ V} \pm 10\%$, $AV_{CC} = 5.0 \text{ V} \pm 10\%$, $V_{SS} = AV_{SS} = 0 \text{ V}$,

$T_a = -20 \text{ to } +75^\circ\text{C}$ (Regular Specifications)

$T_a = -40 \text{ to } +85^\circ\text{C}$ (Wide-Range Specifications)

Item		Symbol	Min	Typ	Max	Unit
Allowable output Low current sink (per pin)	Per output pin	I_{OL}	—	—	2.0	mA
Allowable output Low current sink (total)	Total of all output pins	ΣI_{OL}	—	—	80	mA
Allowable output High current sink (per pin)	All output pins	$-I_{OH}$	—	—	2.0	mA
Allowable output High current sink (total)	Total of all output pins	$\Sigma -I_{OH}$	—	—	25	mA

Note: To avoid degrading the reliability of the chip, be careful not to exceed the output current sink values in table 19-3. In particular, when driving a Darlington transistor pair directly, be sure to insert a current-limiting resistor in the output path. See figure 19-2.

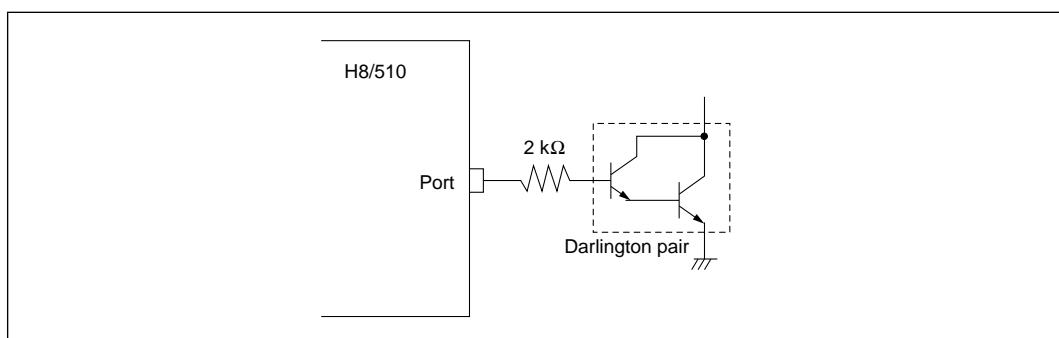


Figure 19-2 Example of Circuit for Driving a Darlington Transistor Pair

19.2.2 AC Characteristics

The AC characteristics of the H8/510 chip are listed in three tables. Bus timing parameters are given in table 20-4, control signal timing parameters in table 20-5, and timing parameters of the on-chip supporting modules in table 20-6.

Table 19-4 Bus Timing

Conditions: $V_{CC} = 5.0\text{ V} \pm 10\%$, $AV_{CC} = 5.0\text{ V} \pm 10\%$, $\phi = 0.5$ to 10 MHz, $V_{SS} = 0\text{ V}$

$T_a = -20$ to $+75^\circ\text{C}$ (Regular Specifications)

$T_a = -40$ to $+85^\circ\text{C}$ (Wide-Range Specifications)

Item	Symbol	6 MHz		8 MHz		10 MHz		Unit	Measurement Conditions
		Min	Max	Min	Max	Min	Max		
Clock cycle time	t _{cyc}	166.7	2000	125	2000	100	2000	ns	See figures 19-4 and 19-5
Clock pulse width Low	t _{CL}	65	–	45	–	40	–	ns	
Clock pulse width High	t _{CH}	65	–	45	–	40	–	ns	
Clock rise time	t _{Cr}	–	15	–	15	–	10	ns	
Clock fall time	t _{Cf}	–	15	–	15	–	10	ns	
Address delay time	t _{AD}	–	50	–	40	–	30	ns	
Address hold time	t _{AH1}	30	–	25	–	15	–	ns	
Read strobe delay time 1	t _{RDD1}	–	50	–	40	–	30	ns	
Read strobe delay time 2	t _{RDD2}	–	50	–	40	–	40	ns	
Read strobe width High 1	t _{ASH1}	110	–	85	–	70	–	ns	
Read strobe width High 2	t _{ASH2}	110	–	85	–	70	–	ns	
Write strobe delay time 1	t _{WRD1}	–	50	–	40	–	30	ns	
Write strobe delay time 2	t _{WRD2}	–	50	–	40	–	30	ns	
Write strobe delay time 3	t _{WRD3}	–	50	–	40	–	30	ns	
Write data strobe pulse width1	t _{WRW1}	150	–	110	–	90	–	ns	
Write data strobe pulse width2	t _{WRW2}	200	–	150	–	120	–	ns	
Address setup time 1	t _{AS1}	25	–	20	–	15	–	ns	
Address setup time 2	t _{AS2}	25	–	20	–	15	–	ns	
Address setup time 3	t _{AS3}	105	–	80	–	65	–	ns	
Read data setup time	t _{RDS}	40	–	30	–	20	–	ns	
Read data hold time	t _{RDH}	0	–	0	–	0	–	ns	
Read data access time 1	t _{ACC1}	–	160	–	125	–	100	ns	
Read data access time 2	t _{ACC2}	–	300	–	230	–	180	ns	
Read data access time 3	t _{ACC3}	–	81.7	–	60	–	55	ns	
Read data access time 4	t _{ACC4}	–	230	–	165	–	135	ns	
Write data delay time	t _{WDD}	–	70	–	60	–	50	ns	
Write data setup time	t _{WDS}	30	–	15	–	10	–	ns	
Write data hold time	t _{WDH}	30	–	25	–	20	–	ns	
Wait setup time	t _{WTS}	40	–	40	–	40	–	ns	See figure 19-6
Wait hold time	t _{WTH}	10	–	10	–	10	–	ns	
Bus request setup time	t _{BRQS}	40	–	40	–	40	–	ns	See figure 19-11
Bus acknowledge delay time 1	t _{BACD1}	–	70	–	60	–	50	ns	
Bus acknowledge delay time 2	t _{BACD2}	–	70	–	60	–	50	ns	
Bus floating delay time	t _{BZD}	–	t _{BACD1}	–	t _{BACD1}	–	t _{BACD1}	ns	

Table 19-4 Bus Timing (cont)

Item	Symbol	6 MHz		8 MHz		10 MHz		Unit	Measurement Conditions
		Min	Max	Min	Max	Min	Max		
E clock delay time	tED	–	20	–	15	–	15	ns	See figure 19-12
E clock rise time	tER	–	15	–	15	–	15	ns	
E clock fall time	tEF	–	15	–	15	–	15	ns	
Read data hold time (E clock sync)	trDHE	0	–	0	–	0	–	ns	See figure 19-7
Write data hold time (E clock sync)	tWDHE	50	–	40	–	30	–	ns	

Conditions: VCC = 3.0 V to 5.5 V, ϕ = 0.5 to 5.0 MHz, VSS = 0 V

Ta = –20 to +75°C (Regular Specifications)

Item	Symbol	5 MHz		Unit	Measurement Conditions
		Min	Max		
Clock cycle time	tcyc	200	2000	ns	See figures 19-4 and 19-5
Clock pulse width Low	tCL	60	–	ns	
Clock pulse width High	tCH	60	–	ns	
Clock rise time	tCr	–	25	ns	
Clock fall time	tCf	–	25	ns	
Address delay time	tAD	–	80	ns	
Address hold time (read)	tAH1	5	–	ns	
Address hold time (write)	tAH2	20	–	ns	
Read strobe delay time 1	trDD1	–	80	ns	
Read strobe delay time 2	trDD2	–	120	ns	
Write strobe delay time 1	tWRD1	–	80	ns	
Write strobe delay time 2	tWRD2	–	80	ns	
Write strobe delay time 3	tWRD3	–	80	ns	
Write data strobe pulse width1	tWRW1	150	–	ns	
Write data strobe pulse width2	tWRW2	220	–	ns	
Address setup time 1	tAS1	30	–	ns	
Address setup time 2	tAS2	30	–	ns	
Address setup time 3	tAS3	130	–	ns	
Read data setup time	trDS	50	–	ns	
Read data hold time	trDH	0	–	ns	
Read data access time 1	tACC1	–	180	ns	
Read data access time 2	tACC2	–	350	ns	
Write data delay time	tWDD	–	120	ns	
Write data setup time	tWDS	10	–	ns	
Write data hold time	tWDH	40	–	ns	

Table 19-4 Bus Timing (cont)

Item	Symbol	5 MHz		Unit	Measurement Conditions
		Min	Max		
Wait setup time	tWTS	60	—	ns	See figure 19-6
Wait hold time	tWTH	20	—	ns	
Bus request setup time	tBRQS	80	—	ns	See figure 19-11
Bus acknowledge delay time 1	tBACD1	—	80	ns	
Bus acknowledge delay time 2	tBACD2	—	80	ns	
Bus floating delay time	tBZD	—	tBACD1	ns	See figure 19-12
E clock delay time	tED	—	50	ns	
E clock rise time	tER	—	25	ns	
E clock fall time	tEF	—	25	ns	
Read data hold time (E clock sync)	tRDHE	30	—	ns	See figure 19-7
Write data hold time (E clock sync)	tWDHE	40	—	ns	

Table 19-5 Control Signal Timing

Conditions: VCC = 5.0 V \pm 10%, AVCC = 5.0 V \pm 10%, ϕ = 0.5 to 10 MHz, VSS = 0 V

Ta = –20 to +75°C (Regular Specifications)

Ta = –40 to +85°C (Wide-Range Specifications)

Item	Symbol	6 MHz		8 MHz		10 MHz		Unit	Measurement Conditions
		Min	Max	Min	Max	Min	Max		
RES setup time	tRESS	200	—	200	—	200	—	ns	See figure 19-8
RES pulse width	tRESW	6.0	—	6.0	—	6.0	—	t _{cyc}	
RES output delay time	tRESO	—	100	—	100	—	100	ns	See figure 19-8
RES output pulse width	tRESOW	132	—	132	—	132	—	t _{cyc}	
NMI setup time	tNMIS	150	—	150	—	150	—	ns	See figure 19-10
NMI hold time	tNMIH	10	—	10	—	10	—	ns	
IRQ ₀ setup time	tIRQ0S	50	—	50	—	50	—	ns	
IRQ ₁ setup time	tIRQ1S	50	—	50	—	50	—	ns	
IRQ ₁ hold time	tIRQ1H	10	—	10	—	10	—	ns	
NMI pulse width (for recovery from software standby mode)	tNMIW	200	—	200	—	200	—	ns	
Crystal oscillator settling time (reset)	tOSC1	20	—	20	—	20	—	ms	See figure 19-11
Crystal oscillator settling time (software standby)	tOSC2	10	—	10	—	10	—	ms	See figure 18-1

Table 19-5 Control Signal Timing (cont)

Conditions: $V_{CC} = 3.0\text{ V to }5.5\text{ V}$, $\phi = 0.5\text{ to }5.0\text{ MHz}$, $V_{SS} = 0\text{ V}$

$T_a = -20\text{ to }+75^\circ\text{C}$ (Regular Specifications)

Item	Symbol	5 MHz		Unit	Measurement Conditions
		Min	Max		
RES setup time	tRESS	300	–	ns	See figure 19-8
RES pulse width	tRESW	6.0	–	t _{cyc}	
RES output delay time	tRESD	–	150	ns	See figure 19-8
RES output pulse width	tRESOW	132	–	t _{cyc}	
NMI setup time	tNMIS	300	–	ns	See figure 19-10
NMI hold time	tNMIH	10	–	ns	
IRQ ₀ setup time	tIRQ0S	100	–	ns	
IRQ ₁ setup time	tIRQ1S	100	–	ns	
IRQ ₁ hold time	tIRQ1H	10	–	ns	
NMI pulse width (for recovery from software standby mode)	tNMIW	200	–	ns	
Crystal oscillator settling time (reset)	tOSC1	20	–	ms	See figure 19-13
Crystal oscillator settling time (software standby)	tOSC2	10	–	ms	See figure 18-1

Table 19-6 Timing Conditions of On-Chip Supporting Modules

Conditions: $V_{CC} = 5.0\text{ V} \pm 10\%$, $AV_{CC} = 5.0\text{ V} \pm 10\%$, $\phi = 0.5$ to 10 MHz , $V_{SS} = 0\text{ V}$

$T_a = -20$ to $+75^\circ\text{C}$ (Regular Specifications)

$T_a = -40$ to $+85^\circ\text{C}$ (Wide-Range Specifications)

			6 MHz		8 MHz		10 MHz		Measurement	
Item		Symbol	Min	Max	Min	Max	Min	Max	Unit	Conditions
FRT	Timer output delay time	tFTOD	—	100	—	100	—	100	ns	See figure 19-15
	Timer input setup time	tFTIS	50	—	50	—	50	—	ns	
	Timer clock input setup time	tFTCS	50	—	50	—	50	—	ns	See figure 19-16
	Timer clock pulse width	tFTCW	1.5	—	1.5	—	1.5	—	t _{cyc}	
TMR	Timer output delay time	tTMOD	—	100	—	100	—	100	ns	See figure 19-17
	Timer clock input setup time	tTMCS	50	—	50	—	50	—	ns	
	Timer clock pulse width	tTMCW	1.5	—	1.5	—	1.5	—	t _{cyc}	See figure 19-18
	Timer reset input setup time	tTMRS	50	—	50	—	50	—	ns	
SCI	Input clock cycle	(Async) t _{Scyc}	2	—	2	—	2	—	t _{cyc}	See figure 19-20
		(Sync)	4	—	4	—	4	—	t _{cyc}	
	Input clock pulse width	tSCKW	0.4	0.6	0.4	0.6	0.4	0.6	t _{Scyc}	See figure 19-21
	Transmit data delay time	(Sync) tTXD	—	100	—	100	—	100	ns	
	Receive data setup time	(Sync) tRXS	100	—	100	—	100	—	ns	
	Receive data hold time	(Sync) tRXH	100	—	100	—	100	—	ns	
Port	Output data delay time	tPWD	—	100	—	100	—	100	ns	See figure 19-14
	Input data setup time	tPRS	50	—	50	—	50	—	ns	
	Input data hold time	tPRH	50	—	50	—	50	—	ns	
RFSH	Refresh output delay time 1	tRFD1	—	50	—	45	—	40	ns	See figure 19-22
	Refresh output delay time 2	tRFD2	—	50	—	45	—	40	ns	

Table 19-6 Timing Conditions of On-Chip Supporting Modules (cont)

Conditions: $V_{CC} = 3.0\text{ V to }5.5\text{ V}$, $\phi = 0.5\text{ to }5.0\text{ MHz}$, $V_{SS} = 0\text{ V}$

$T_a = -20\text{ to }+75^\circ\text{C}$ (Regular Specifications)

$T_a = -40\text{ to }+85^\circ\text{C}$ (Wide-Range Specifications)

Item		Symbol	5 MHz		Unit	Measurement
			Min	Max		Conditions
FRT	Timer output delay time	tFTOD	—	150	ns	See figure 19-15
	Timer input setup time	tFTIS	80	—	ns	
	Timer clock input setup time	tFTCS	80	—	ns	See figure 19-16
	Timer clock pulse width	tFTCW	1.5	—	t _{cyc}	
TMR	Timer output delay time	tTMOD	—	150	ns	See figure 19-17
	Timer clock input setup time	tTMCS	80	—	ns	See figure 19-18
	Timer clock pulse width	tTMCW	1.5	—	t _{cyc}	
	Timer reset input setup time	tTMRS	80	—	ns	See figure 19-19
SCI	Input clock cycle	(Async) tScyc	2	—	t _{cyc}	See figure 19-20
		(Sync)	4	—	t _{cyc}	
	Input clock pulse width	tSCKW	0.4	0.6	tScyc	
	Transmit data delay time	(Sync) tTXD	—	200	ns	See figure 19-21
	Receive data setup time	(Sync) tRXS	150	—	ns	
	Receive data hold time	(Sync) tRXH	150	—	ns	
Port	Output data delay time	tPWD	—	150	ns	See figure 19-14
	Input data setup time	tPRS	80	—	ns	
	Input data hold time	tPRH	80	—	ns	
RFSH	Refresh output delay time 1	tRFD1	—	80	ns	See figure 19-22
	Refresh output delay time 2	tRFD2	—	80	ns	

• **Measurement Conditions for AC Characteristics**

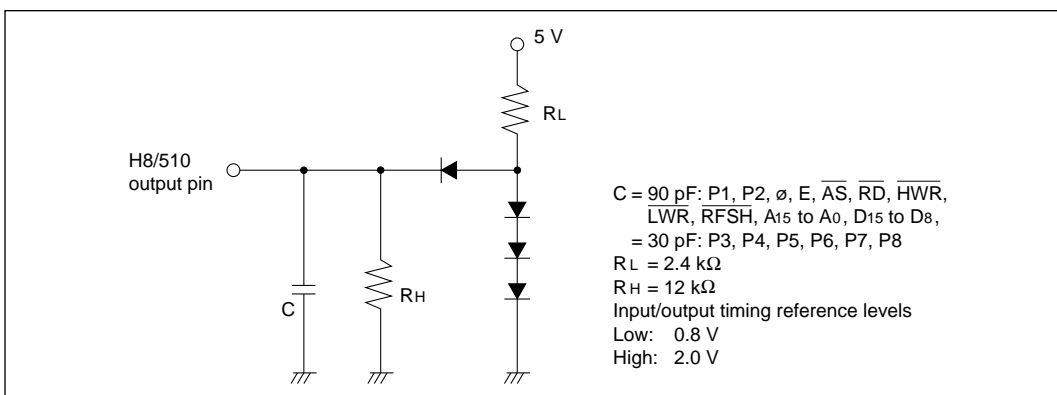


Figure 19-3 Output Load Circuit

19.2.3 A/D Converter Characteristics

Table 19-7 lists the characteristics of the on-chip A/D converter.

Table 19-7 A/D Converter Characteristics

Conditions: $V_{CC} = 5.0\text{ V} \pm 10\%$, $AV_{CC} = 5.0\text{ V} \pm 10\%$, $V_{SS} = AV_{SS} = 0\text{ V}$,

$T_a = -20\text{ to }+75^\circ\text{C}$ (Regular Specifications)

$T_a = -40\text{ to }+85^\circ\text{C}$ (Wide-Range Specifications)

Item	6 MHz			8 MHz			10 MHz			Unit
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Resolution	10	10	10	10	10	10	10	10	10	Bits
Conversion time	—	—	22.33	—	—	16.75	—	—	13.4	μs
Analog input capacitance	—	—	20	—	—	20	—	—	20	pF
Allowable signal-source impedance	—	—	10	—	—	10	—	—	10	k Ω
Nonlinearity error	—	—	± 3	—	—	± 3	—	—	± 3	LSB
Offset error	—	—	± 2	—	—	± 2	—	—	± 2	LSB
Full-scale error	—	—	± 2	—	—	± 2	—	—	± 2	LSB
Quantizing error	—	—	$\pm 1/2$	—	—	$\pm 1/2$	—	—	$\pm 1/2$	LSB
Absolute accuracy	—	—	± 4	—	—	± 4	—	—	± 4	LSB

Conditions: $V_{CC} = 3.0\text{ V to }5.5\text{ V}$, $V_{SS} = AV_{SS} = 0\text{ V}$, $T_a = -20\text{ to }+75^\circ\text{C}$ (Regular Specifications)

$AV_{CC} = 5.0\text{ V} \pm 10\%$

Item	5 MHz			Unit
	Min	Typ	Max	
Resolution	10	10	10	Bits
Conversion time	—	—	26.8	μs
Analog input capacitance	—	—	20	pF
Allowable signal-source impedance	—	—	10	k Ω
Nonlinearity error	—	—	± 3	LSB
Offset error	—	—	± 2	LSB
Full-scale error	—	—	± 2	LSB
Quantizing error	—	—	$\pm 1/2$	LSB
Absolute accuracy	—	—	± 4	LSB

19.3 MCU Operational Timing

This section provides the following timing charts:

19.3.1 Bus timing	Figures 19-4 to 19-7
19.3.2 Control Signal Timing	Figures 19-8 to 19-11
19.3.3 Clock Timing	Figures 19-12 and 19-13
19.3.4 I/O Port Timing	Figure 19-14
19.3.5 16-Bit Free-Running Timer Timing	Figures 19-15 and 19-16
19.3.6 8-Bit Timer Timing	Figures 19-17 to 19-19
19.3.7 Serial Communication Interface Timing	Figures 19-20 and 19-21
19.3.8 Refresh Timing	Figures 19-22 and 19-23

19.3.1 Bus Timing

1. Basic Bus Cycle (Two-State Mode)

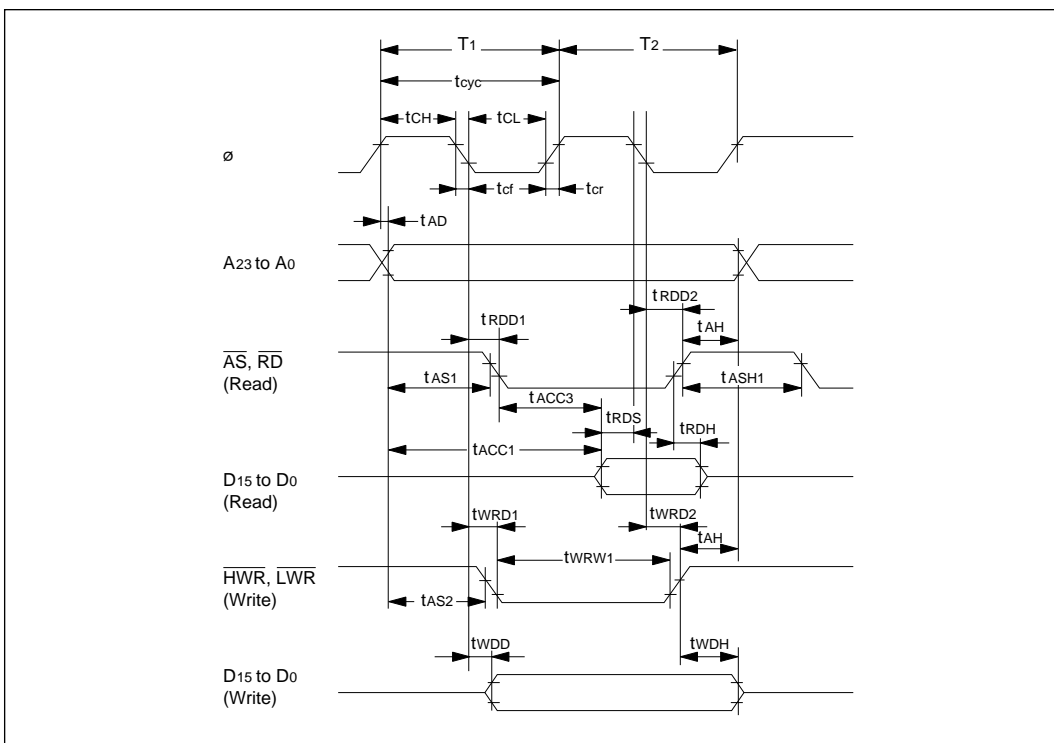


Figure 19-4 Basic Bus Cycle (Two-State Mode)

2. Basic Bus Cycle (Three-State Mode)

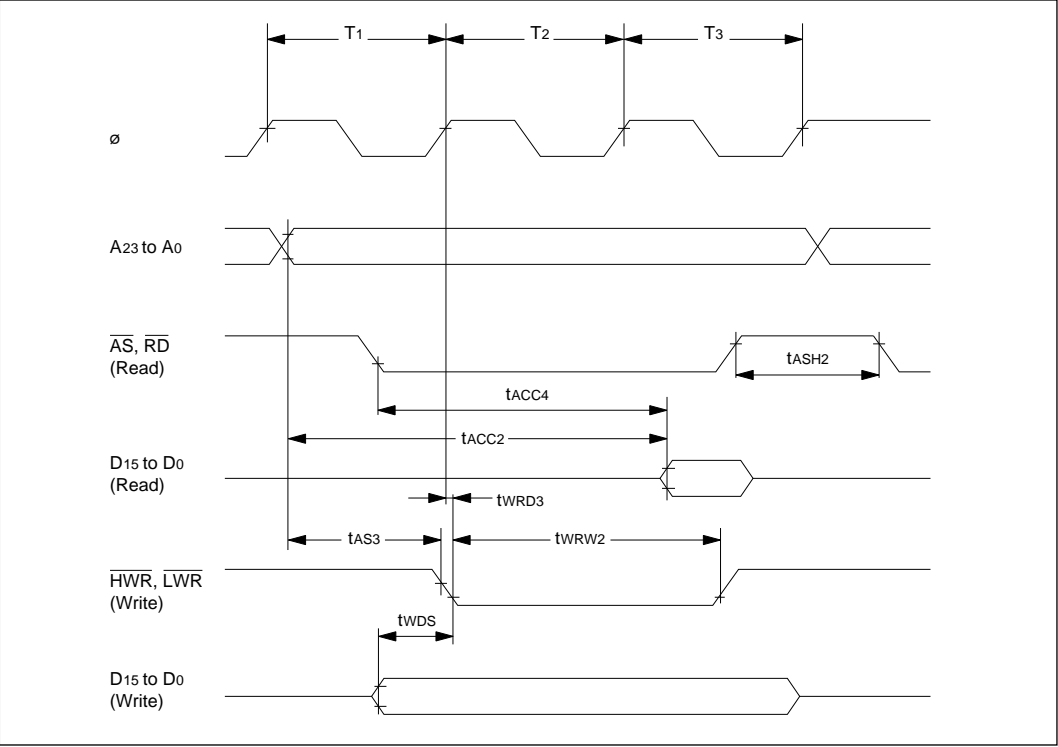


Figure 19-5 Basic Bus Cycle (Three-State Mode)

3. Basic Bus Cycle (Three-State Mode with One Wait State)

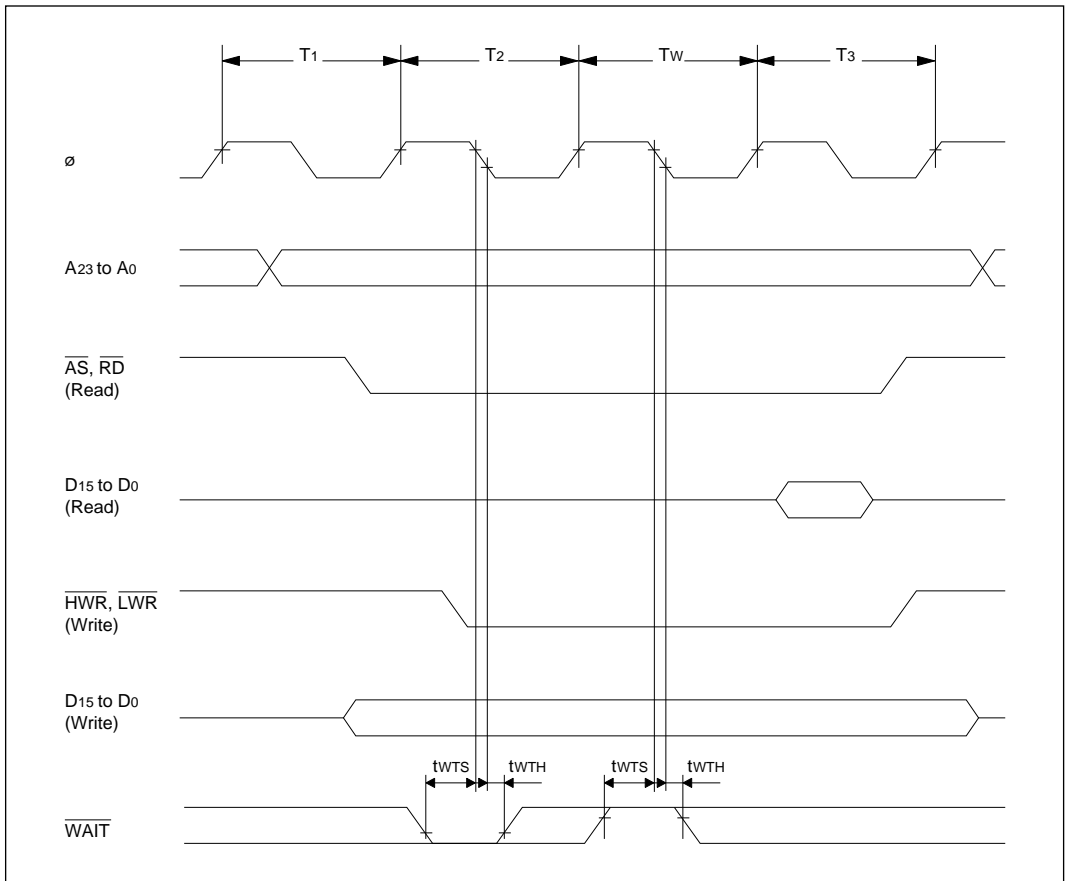


Figure 19-6 Basic Bus Cycle (Three-State Mode with One Wait State)

4. Bus Cycle Synchronized with E Clock

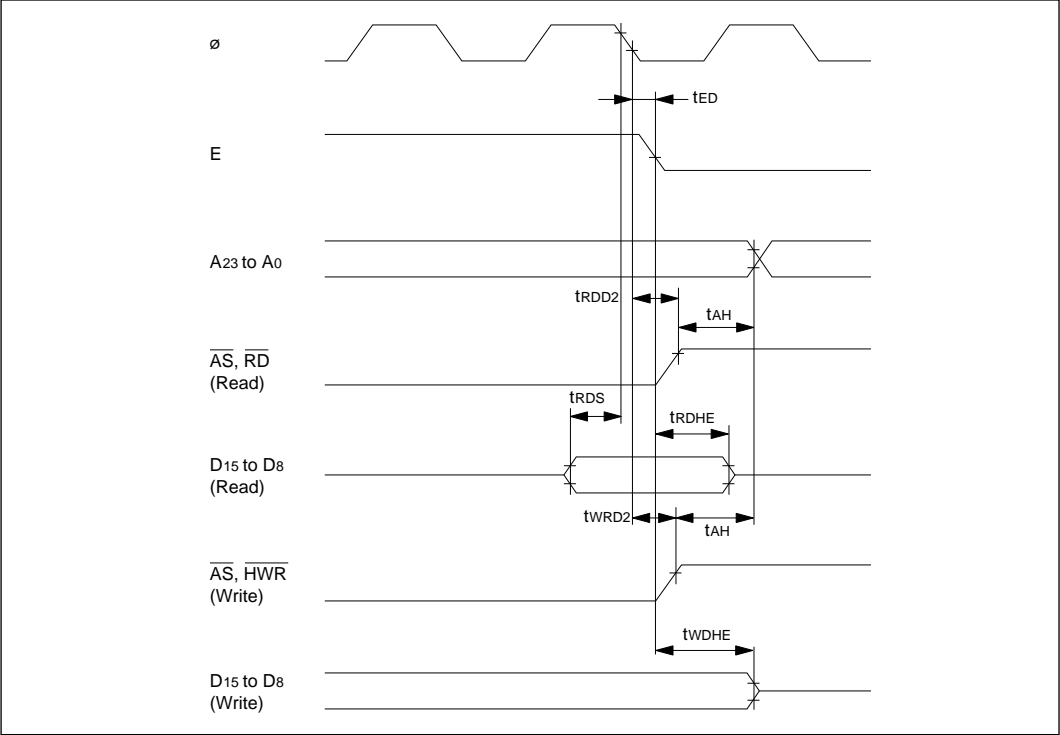


Figure 19-7 Bus Cycle Synchronized with E Clock

19.3.2 Control Signal Timing

1. Reset Input Timing

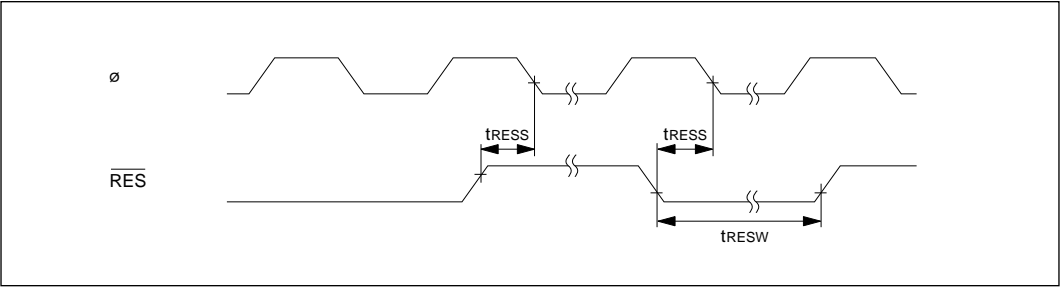


Figure 19-8 Reset Input Timing

2. Reset Output Timing

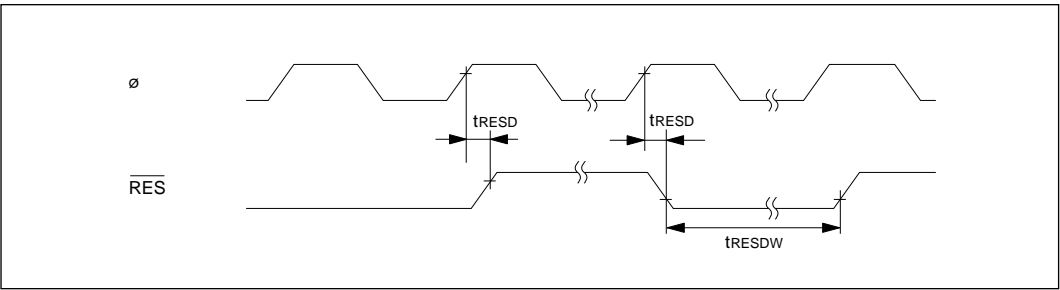


Figure 19-9 Reset Output Timing

3. Interrupt Input Timing

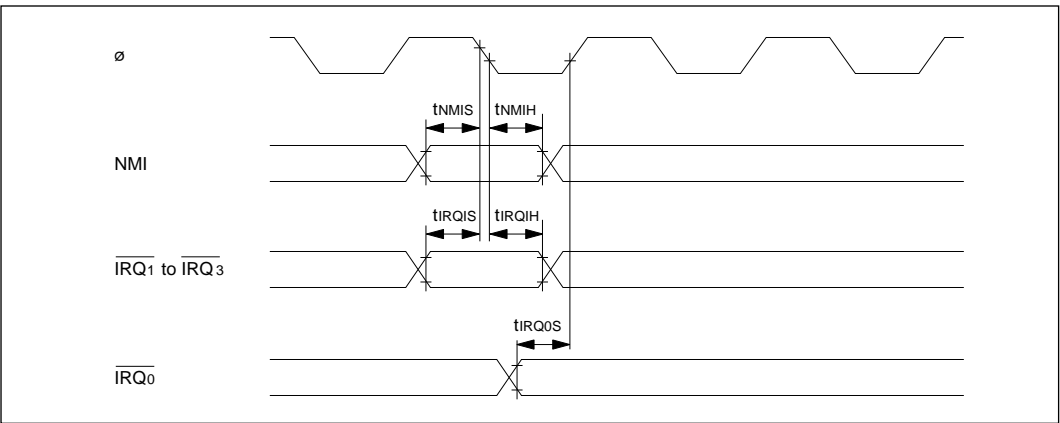


Figure 19-10 Interrupt Input Timing

4. Bus Release State Timing

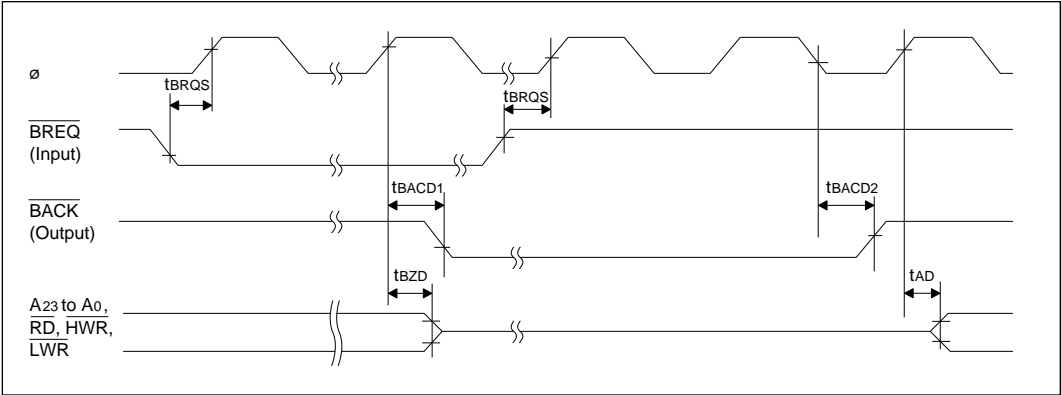


Figure 19-11 Bus Release State Timing

19.3.3 Clock Timing

1. E Clock Timing

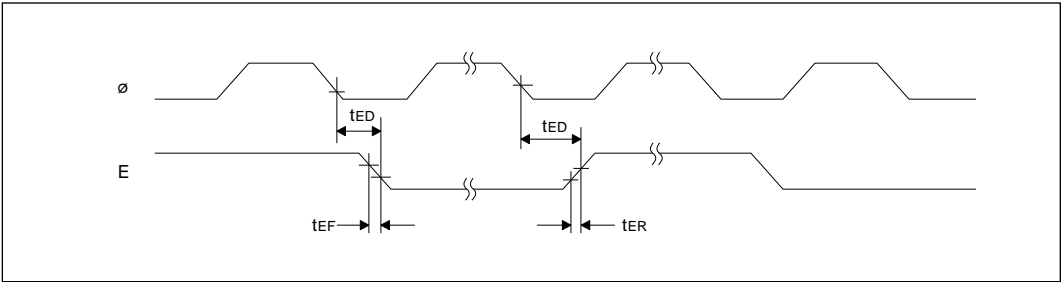


Figure 19-12 E Clock Timing

2. Clock Oscillator Stabilization Timing

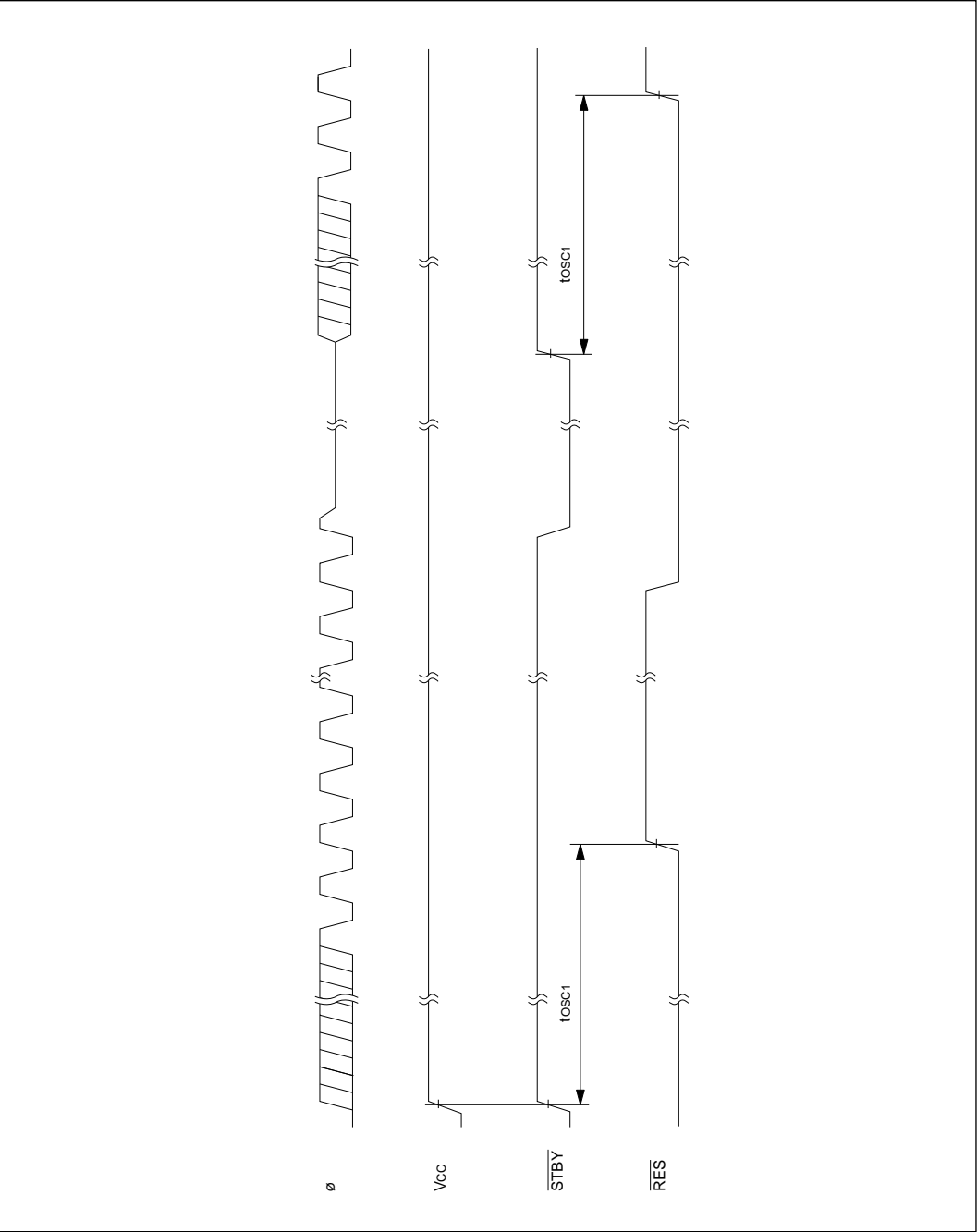


Figure 19-13 Clock Oscillator Stabilization Timing

19.3.4 I/O Port Timing

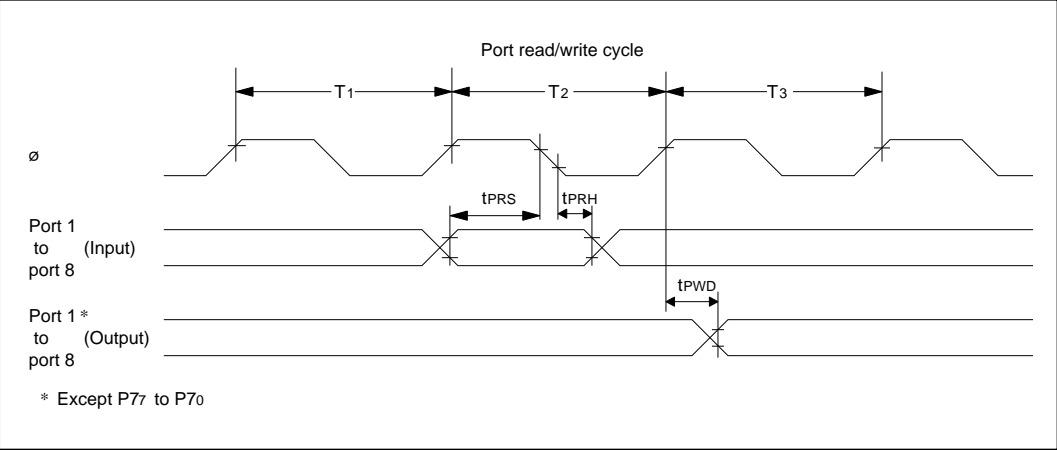


Figure 19-14 I/O Port Input/Output Timing

19.3.5 16-Bit Free-Running Timer Timing

1. Free-Running Timer Input/Output Timing

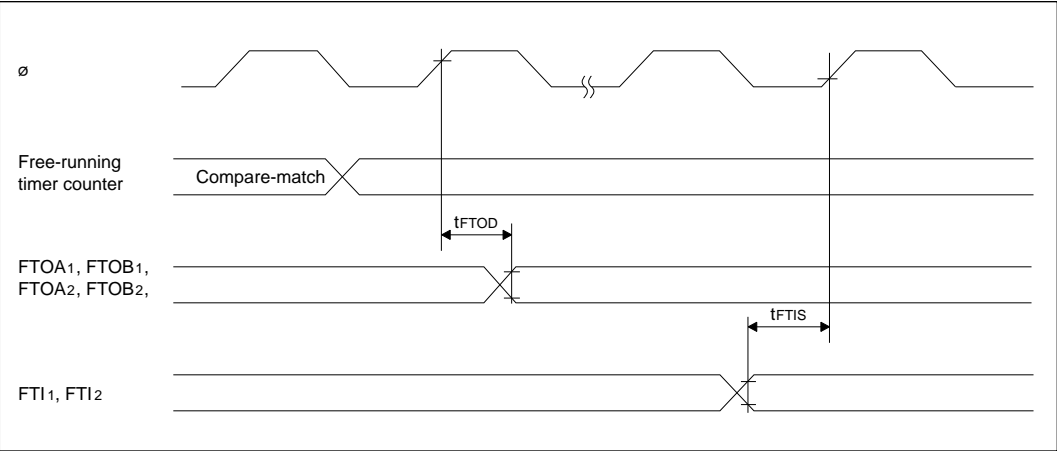


Figure 19-15 Free-Running Timer Input/Output Timing

2. External Clock Input Timing for Free-Running Timers

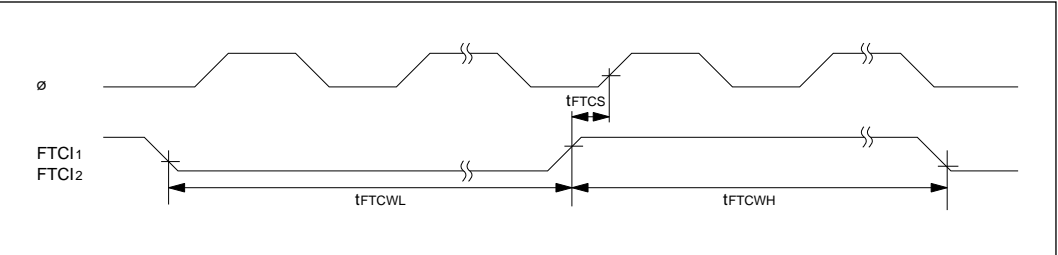


Figure 19-16 External Clock Input Timing for Free-Running Timers

19.3.6 8-Bit Timer Timing

1. 8-Bit Timer Output Timing

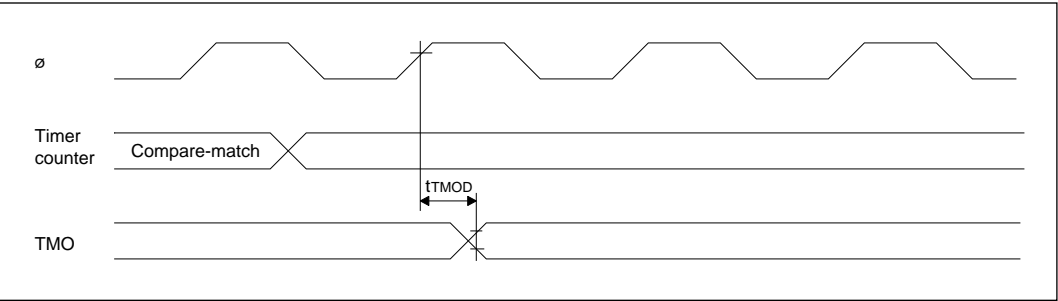


Figure 19-17 8-Bit Timer Output Timing

2. 8-Bit Timer Clock Input Timing

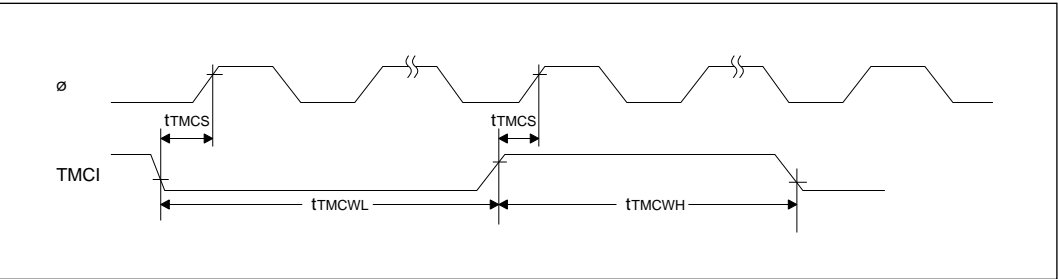


Figure 19-18 8-Bit Timer Clock Input Timing

3. 8-Bit Timer Reset Input Timing

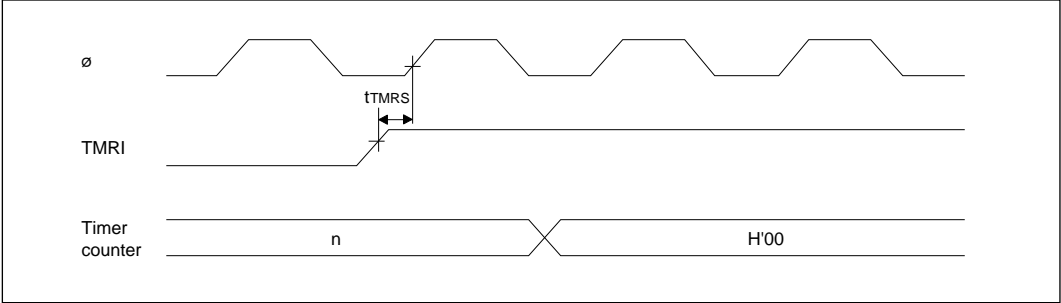


Figure 19-19 8-Bit Timer Reset Input Timing

19.3.7 Serial Communication Interface Timing

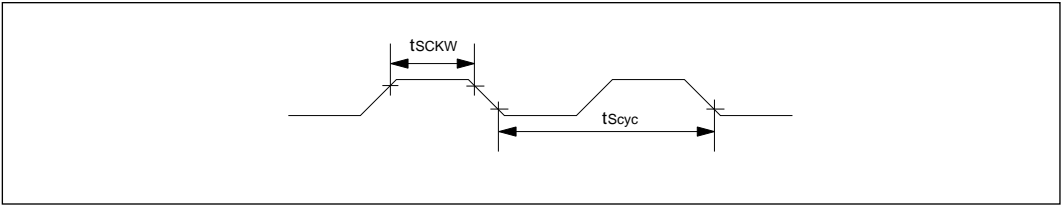


Figure 19-20 SCI Input Clock Timing

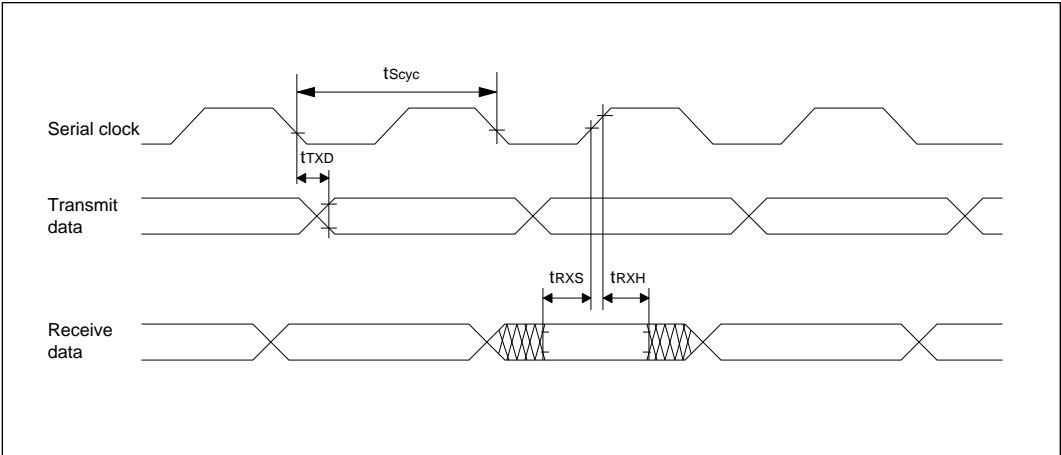


Figure 19-21 SCI Input/Output Timing (Synchronous Mode)

19.3.8 Refresh Timing

1. Basic Refresh Bus Cycle

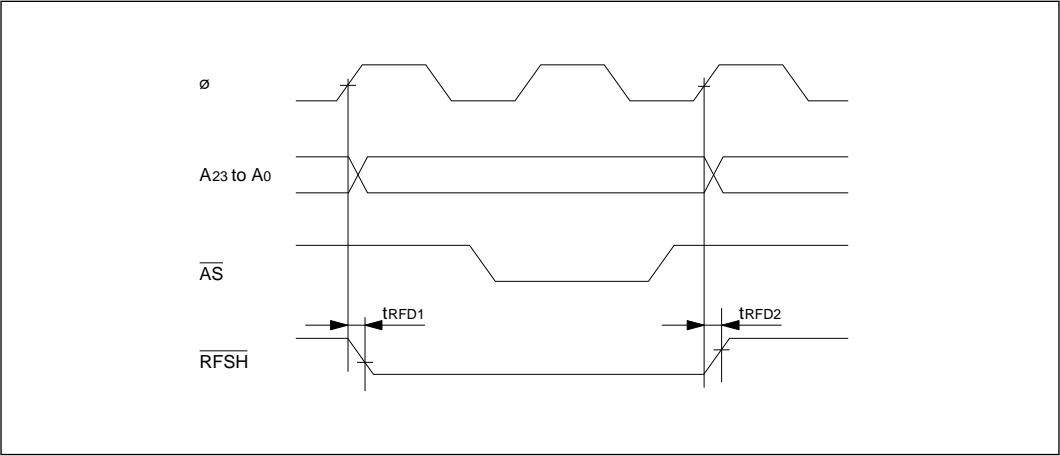


Figure 19-22 Basic Refresh Bus Cycle

2. Refresh Timing (Wait Cycle)

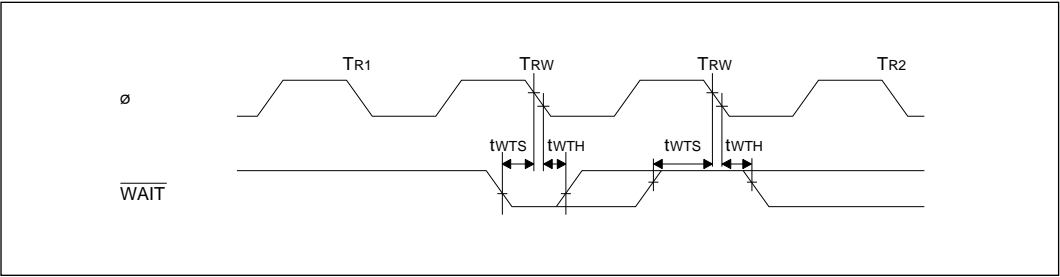


Figure 19-23 Refresh Timing (Wait Cycle)

Appendix A Instructions

A.1 Instruction Set

Operation Notation

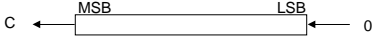
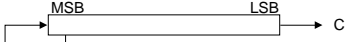
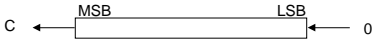
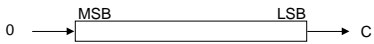
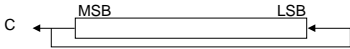
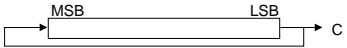
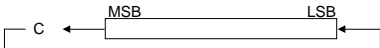
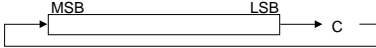
Rd	General register (destination operand)
Rs	General register (source operand)
Rn	General register
(EAd)	Destination operand
(EAs)	Source operand
CCR	Condition code register
N	N (Negative) flag in CCR
Z	Z (Zero) flag in CCR
V	V (Overflow) flag in CCR
C	C (Carry) flag in CCR
CR	Control register
PC	Program counter
CP	Code page register
SP	Stack pointer

FP	Frame pointer
#IMM	Immediate data
disp	Displacement
+	Add
−	Subtract
×	Multiply
÷	Divide
^	Logical AND
∨	Logical OR
⊕	Logical exclusive OR
→	Move
↔	Swap
¬	Logical NOT

Condition Code Notation

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

Mnemonic	Operation	Size	CCR Bits			
		B/W	N	Z	V	C
Data transfer	MOV: G (EAs) \longrightarrow Rd Rs \longrightarrow (EAd) #IMM \longrightarrow (EAd)	B/W	\updownarrow	\updownarrow	0	—
	MOV: E #IMM \longrightarrow Rd (short format)	B	\updownarrow	\updownarrow	0	—
	MOV: F @ (d: 8, FP) \longrightarrow Rd Rs \longrightarrow @ (d: 8, FP) (short format)	B/W	\updownarrow	\updownarrow	0	—
	MOV: I #IMM \longrightarrow Rd (short format)	W	\updownarrow	\updownarrow	0	—
	MOV: L (@aa: 8) \longrightarrow Rd (short format)	B/W	\updownarrow	\updownarrow	0	—
	MOV: S Rs \longrightarrow (@aa: 8) (short format)	B/W	\updownarrow	\updownarrow	0	—
	LDM @ SP + \longrightarrow Rn (register list)	W	—	—	—	—
	STM Rn (register list) \longrightarrow @ – SP	W	—	—	—	—
	XCH Rs \longleftrightarrow Rd	W	—	—	—	—
	SWAP Rd (upper byte) \longleftrightarrow Rd (lower byte)	B	\updownarrow	\updownarrow	0	—
	MOVTPE Rs \longrightarrow (EAd) Synchronized with E clock	B	—	—	—	—
	MOVFPPE (EAs) \longrightarrow Rd Synchronized with E clock	B	—	—	—	—
Arithmetic operations	ADD: G Rd + (EAs) \longrightarrow Rd	B/W	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	ADD: Q (EAd) + #IMM \longrightarrow (EAd) (#IMM = $\pm 1, \pm 2$) (short format)	B/W	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	ADDS Rd + (EAs) \longrightarrow Rd (Rd is always word size)	B/W	—	—	—	—
	ADDX Rd + (EAs) + C \longrightarrow Rd	B/W	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	DADD (Rd) ₁₀ + (Rs) ₁₀ + C \longrightarrow (Rd) ₁₀	B	—	\updownarrow	—	\updownarrow
	SUB Rd – (EAs) \longrightarrow Rd	B/W	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	SUBS Rd – (EAs) \longrightarrow Rd	B/W	—	—	—	—
	SUBX Rd – (EAs) – C \longrightarrow Rd	B/W	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	DSUB (Rd) ₁₀ – (Rs) ₁₀ – C \longrightarrow (Rd) ₁₀	B	—	\updownarrow	—	\updownarrow
	MULXU Rd \times (EAs) \longrightarrow Rd 8 \times 8 (Unsigned) 16 \times 16	B/W	\updownarrow	\updownarrow	0	0
	DIVXU Rd \div (EAs) \longrightarrow Rd 16 \div 8 (Unsigned) 32 \div 16	B/W	\updownarrow	\updownarrow	\updownarrow	0
	CMP: G Rd – (EAs), Set CCR (EAd) – #IMM, Set CCR	B/W	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	CMP: E Rd – #IMM, Set CCR (short format)	B	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	CMP: I Rd – #IMM, Set CCR (short format)	W	\updownarrow	\updownarrow	\updownarrow	\updownarrow

			Size	CCR Bits				
Mnemonic		Operation	B/W	N	Z	V	C	
Arithmetic operations	EXTS	(< Bit 7 > of < Rd >) → (< Bit 15 to 8 > of < Rd >)	B	↑	↑	0	0	
	EXTU	0 → (< Bit 15 to 8 > of < Rd >)	B	0	↑	0	0	
	TST	(EAd) – 0, Set CCR	B/W	↑	↑	0	0	
	NEG	0 – (EAd) → (EAd)	B/W	↑	↑	0	↑	
	CLR	0 → (EAd)	B/W	0	1	0	0	
	TAS	(EAd) – 0, Set CCR (1) ₂ → (< Bit 7 > of < EAd >)	B	↑	↑	0	0	
Shift operations	SHAL		B/W	↑	↑	↑	↑	
	SHAR		B/W	↑	↑	0	↑	
	SHLL		B/W	↑	↑	0	↑	
	SHLR		B/W	0	↑	0	↑	
	ROTL		B/W	↑	↑	0	↑	
	ROTR		B/W	↑	↑	0	↑	
	ROTXL		B/W	↑	↑	0	↑	
	ROTXR		B/W	↑	↑	0	↑	
Logic operations	AND	Rd ∧ (EAs) → Rd	B/W	↑	↑	0	—	
	OR	Rd ∨ (EAs) → Rd	B/W	↑	↑	0	—	
	XOR	Rd ⊕ (EAs) → Rd	B/W	↑	↑	0	—	
	NOT	¬ (EAd) → (EAd)	B/W	↑	↑	0	—	
Bit manipulations	BSET	¬ (< Bit number > of < EAd >) → Z 1 → (< Bit number > of < EAd >)	B/W	—	↑	—	—	
	BCLR	¬ (< Bit number > of < EAd >) → Z 0 → (< Bit number > of < EAd >)	B/W	—	↑	—	—	
	BTST	¬ (< Bit number > of < EAd >) → Z	B/W	—	↑	—	—	
	BNOT	¬ (< Bit number > of < EAd >) → Z → (< Bit number > of < EAd >)	B/W	—	↑	—	—	

Mnemonic	Operation	Size		CCR Bits			
		B/W	N	Z	V	C	
Branching instructions	If condition is true then PC + disp \longrightarrow PC else next;	—	—	—	—	—	
	Mnemonic	Description		Condition			
	BRA (BT)	Always (True)		True			
	BRN (BF)	Never (False)		False			
	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		$Z \vee (N \oplus V) = 1$			
JMP	Effective address \longrightarrow PC	—	—	—	—	—	
PJMP	Effective address \longrightarrow CP, PC	—	—	—	—	—	
BSR	PC \longrightarrow @ – SP PC + disp \longrightarrow PC	—	—	—	—	—	
JSR	PC \longrightarrow @ – SP Effective address \longrightarrow PC	—	—	—	—	—	
PJSR	PC \longrightarrow @ – SP CP \longrightarrow @ – SP Effective address \longrightarrow CP, PC	—	—	—	—	—	
RTS	@ SP + \longrightarrow PC	—	—	—	—	—	
PRTS	@ SP + \longrightarrow CP @ SP + \longrightarrow PC	—	—	—	—	—	
RTD	@ SP + \longrightarrow PC SP + #IMM \longrightarrow SP	—	—	—	—	—	
PRTD	@ SP + \longrightarrow CP @ SP + \longrightarrow PC SP + #IMM \longrightarrow SP	—	—	—	—	—	
SCB	If condition is true then next; SCB/F else Rn – 1 \longrightarrow Rn; SCB/NE If Rn = –1 then next; SCB/EQ else PC + disp \longrightarrow PC;	—	—	—	—	—	
	Mnemonic	Description		Condition			
	SCB/F			False			
	SCB/NE	Not Equal		$Z = 0$			
	SCB/EQ	Equal		$Z = 1$			

Mnemonic	Operation	Size	CCR Bits			
		B/W	N	Z	V	C
System control	TRAPA PC \longrightarrow @ – SP (If MAX MODE CP \longrightarrow @ – SP) SR \longrightarrow @ – SP (If MAX MODE < vector > \longrightarrow CP) < vector > \longrightarrow PC	—	—	—	—	—
	TRAP/VS If V bit = 1 then TRAP else next;	—	—	—	—	—
	RTE @ SP + \longrightarrow SR (If MAX MODE @ SP + \longrightarrow CP) @ SP + \longrightarrow PC	—	\updownarrow	\updownarrow	\updownarrow	\updownarrow
	LINK FP (R6) \longrightarrow @ – SP SP \longrightarrow FP (R6) SP + #IMM \longrightarrow SP	—	—	—	—	—
	UNLK FP (R6) \longrightarrow SP @SP + \longrightarrow FP	—	—	—	—	—
	SLEEP Normal running mode \longrightarrow power-down state	—	—	—	—	—
	LDC (EAs) \longrightarrow CR	B/W*	\triangle	\triangle	\triangle	\triangle
	STC CR \longrightarrow (EAd)	B/W*	—	—	—	—
	ANDC CR \wedge #IMM \longrightarrow CR	B/W*	\triangle	\triangle	\triangle	\triangle
	ORC CR \vee #IMM \longrightarrow CR	B/W*	\triangle	\triangle	\triangle	\triangle
	XORC CR \oplus #IMM \longrightarrow CR	B/W*	\triangle	\triangle	\triangle	\triangle
	NOP PC + 1 \longrightarrow PC	—	—	—	—	—

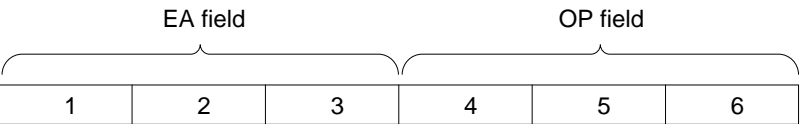
* Depends on the CR.

A.2 Instruction Codes

Table A-1 shows the machine-language coding of each instruction.

- **How to Read Table A-1 (a) to (d)**

The general operand format consists of an effective address (EA) field and operation-code (OP) field specified in the following order.



Bytes 2, 3, 5, 6 are not present in all instructions.

Instruction		Operation code (EA)										Operation code (OP)			
		Address- ing mode		1			2			3					
		Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16	4	5	6	
Instruction	MOV:G.B <EA _s >, R _d	2	2	3	4	2	2	3	4	3		1 0 0 0 0	r _d r _d r _d		
	MOV:G.W <EA _s >, R _d	2	2	3	4	2	2	3	4		4	1 0 0 0 0	r _d r _d r _d		
	MOV:G.B R _s , <EA _d >		2	3	4	2	2	3	4			1 0 0 1 0	r _s r _s r _s		
	MOV:G.W R _s , <EA _d >		2	3	4	2	2	3	4			1 0 0 1 0	r _s r _s r _s		

Byte length of instruction

Shading indicates addressing modes not available for this instruction.

Some instructions have a special format in which the operation code comes first.

The following notation is used in the tables.

- Sz: Operand size (byte or word)
 Byte: Sz = 0
 Word: Sz = 1

- rrr : General register number field

rrr	Sz = 0 (Byte)			Sz = 1 (Word)	
	15	8 7	0	15	0
000	Not used	R0		R0	
001	Not used	R1		R1	
010	Not used	R2		R2	
011	Not used	R3		R3	
100	Not used	R4		R4	
101	Not used	R5		R5	
110	Not used	R6		R6	
111	Not used	R7		R7	

- ccc : Control register number field

ccc	Sz = 0 (Byte)			Sz = 1 (Word)	
	15	8 7	0	15	0
000	(Not allowed*)			SR	
001	Not used	CCR		(Not allowed)	
010	(Not allowed)			(Not allowed)	
011	Not used	BR		(Not allowed)	
100	Not used	EP		(Not allowed)	
101	Not used	DP		(Not allowed)	
110	(Not allowed)			(Not allowed)	
111	Not used	TP		(Not allowed)	

* "Not allowed" means that this combination of bits must not be specified. Specifying a disallowed combination may cause abnormal results.

- d: Transfer direction
Load when d = 0
Store when d = 1
- register list: A byte in which bits indicate general registers as follows

Bit	7	6	5	4	3	2	1	0
	R7	R6	R5	R4	R3	R2	R1	R0

- #VEC: Four bits designating a vector number from 0 to 15. The vector numbers correspond to addresses of entries in the exception vector table as follows:

Vector Address			Vector Address		
#VEC	Minimum Mode	Maximum Mode	#VEC	Minimum Mode	Maximum Mode
0	H'0020 – H'0021	H'0040 – H'0043	8	H'0030 – H'0031	H'0060 – H'0063
1	H'0022 – H'0023	H'0044 – H'0047	9	H'0032 – H'0033	H'0064 – H'0067
2	H'0024 – H'0025	H'0048 – H'004B	A	H'0034 – H'0035	H'0068 – H'006B
3	H'0026 – H'0027	H'004C – H'004F	B	H'0036 – H'0037	H'006C – H'006F
4	H'0028 – H'0029	H'0050 – H'0053	C	H'0038 – H'0039	H'0070 – H'0073
5	H'002A – H'002B	H'0054 – H'0057	D	H'003A – H'003B	H'0074 – H'0077
6	H'002C – H'002D	H'0058 – H'005B	E	H'003C – H'003D	H'0078 – H'007B
7	H'002E – H'002F	H'005C – H'005F	F	H'003E – H'003F	H'007C – H'007F

• Example of machine-language coding

Example 1: ADD:G.B @R0, R1

	EA Field	OP Field	
Table A-1 (a)	1101Szrrr	00100rrr	
Machine code	11010000	00100001	Sz = 0 (Byte)
	H'D021		Rs = R0, Rd = R1

Example 2: ADD:G.W @H'11:8, R1

	EA Field	OP Field	
Table A-1 (a)	0000Sz101	00010001	00100rrr
Machine code	00001101	00010001	00100001
	H'0D1121		

Table A-1 (a) Machine Language Coding [General Format]

Instruction		Operation code (EA)			Address- ing mode	Operation code (OP)					
		1		2							
		Rn				4		5		6	
Data transfer instruction	MOV:G.B <EAs>, Rd	2	2	3	4	2	2	3	4	3	1 0 0 0 0 Rd Rd Rd
	MOV:G.W <EAs>, Rd	2	2	3	4	2	2	3	4	4	1 0 0 0 0 Rd Rd Rd
	MOV:G.B Rs, <EA _d >		2	3	4	2	2	3	4		1 0 0 1 0 Rs Rs Rs
	MOV:G.W Rs, <EA _d >		2	3	4	2	2	3	4		1 0 0 1 0 Rs Rs Rs
	MOV:G.B #xx:8, <EA _d >		3	4	5	3	3	4	5		0 0 0 0 0 1 1 0 data
	MOV:G.W #xx:8, <EA _d >		3	4	5	3	3	4	5		0 0 0 0 0 1 1 0 data
	MOV:G.W #xx:16, <EA _d >		4	5	6	4	4	5	6		0 0 0 0 0 1 1 1 data (H) data (L)
	LDM.W @SP+, <register list>					2					0 0 0 0 0 0 1 0 register list
	STM.W <register list>, @-SP					2					0 0 0 1 0 0 1 0 register list
	XCH.W Rs, Rd	2									1 0 0 1 0 Rd Rd Rd
Arithmetic operation instruction	SWAP.B Rd	2									0 0 0 1 0 0 0 0
	MOVTPE.B Rs, <EA _d >		3	4	5	3	3	4	5		0 0 0 0 0 0 0 0 1 0 0 1 0 Rs Rs Rs
	MOVFPPE.B <EAs>, Rd		3	4	5	3	3	4	5		0 0 0 0 0 0 0 0 1 0 0 0 0 Rd Rd Rd
	ADD:G.B <EAs>, Rd	2	2	3	4	2	2	3	4	3	0 0 1 0 0 Rd Rd Rd
	ADD:G.W <EAs>, Rd	2	2	3	4	2	2	3	4	4	0 0 1 0 0 Rd Rd Rd
	ADD:Q.B #1, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 0 0 0
	ADD:Q.W #1, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 0 0 0
	ADD:Q.B #2, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 0 0 1
	ADD:Q.W #2, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 0 0 1
	ADD:Q.B #-1, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 1 0 0
	ADD:Q.W #-1, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 1 0 0
	ADD:Q.B #-2, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 1 0 1
	ADD:Q.W #-2, <EA _d >*	2	2	3	4	2	2	3	4		0 0 0 0 1 1 0 1
	ADDS.B <EAs>, Rd	2	2	3	4	2	2	3	4	3	0 0 1 0 1 Rd Rd Rd
	ADDS.W <EAs>, Rd	2	2	3	4	2	2	3	4	4	0 0 1 0 1 Rd Rd Rd
	ADDX.B <EAs>, Rd	2	2	3	4	2	2	3	4	3	1 0 1 0 0 Rd Rd Rd
	ADDX.W <EAs>, Rd	2	2	3	4	2	2	3	4	4	1 0 1 0 0 Rd Rd Rd

* Short format instruction

Table A-1 (a) Machine Language Coding [General Format] (cont)

Instruction		Operation code (EA)			Operation code (OP)				
		Address- ing mode	1					2	3
			Rn	@Rn				disp	disp (L)
Arithmetic operation instruction	DADD.B Rs,Rd	3	1 0 1 0 S z r r r						
	SUB.B <EAs>, Rd	2 2 3 4 2 2 3 4 3	1 1 0 1 S z r r r						
	SUB.W <EAs>, Rd	2 2 3 4 2 2 3 4	1 1 0 S z r r r	disp					
	SUBS.B <EAs>, Rd	2 2 3 4 2 2 3 4 3	1 1 1 S z r r r	disp (H)					
	SUBS.W <EAs>, Rd	2 2 3 4 2 2 3 4	1 0 1 1 S z r r r						
	SUBX.B <EAs>, Rd	2 2 3 4 2 2 3 4 3	1 0 1 1 S z r r r						
	SUBX.W <EAs>, Rd	2 2 3 4 2 2 3 4	1 1 0 S z r r r						
	DSUB.B Rs,Rd	3	1 1 0 S z r r r						
	MULXU.B <EAs>, Rd	2 2 3 4 2 2 3 4 3	0 0 0 S z 1 0 1	address					
	MULXU.W <EAs>, Rd	2 2 3 4 2 2 3 4	0 0 0 1 S z 1 0 1	address (H)					
	DIVXU.B <EAs>, Rd	2 2 3 4 2 2 3 4 3	0 0 0 0 1 0 0	data					
	DIVXU.W <EAs>, Rd	2 2 3 4 2 2 3 4	0 0 0 0 1 1 0	data (H)					
	CMP:G.B <EAs>, Rd	2 2 3 4 2 2 3 4 3	0 0 0 0 1 0 0	data (H)					
	CMP:G.W <EAs>, Rd	2 2 3 4 2 2 3 4	0 0 0 0 1 1 0	data (H)					
	CMP:G.B #xx,<EA d>	3 4 5 3 3 4 5							
	CMP:G.W #xx,<EA d>	4 5 6 4 4 5 6							
	EXTS.B Rd	2							
	EXTU.B Rd	2							
	TST.B <EA d>	2 2 3 4 2 2 3 4							
	TST.W <EA d>	2 2 3 4 2 2 3 4							
	NEG.B <EA d>	2 2 3 4 2 2 3 4							
	NEG.W <EA d>	2 2 3 4 2 2 3 4							
	CLR.B <EA d>	2 2 3 4 2 2 3 4							
	CLR.W <EA d>	2 2 3 4 2 2 3 4							
	TAS.B <EA d>	2 2 3 4 2 2 3 4							

Table A-1 (a) Machine Language Coding [General Format] (cont)

Instruction		Address- ing mode		Operation code (EA)								Operation code (OP)		
				1				2						
				Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16	4
Shift instruction	SHAL.B <EA _d >	2	2	3	4	2	2	3	4			00011000		
	SHAL.W <EA _d >	2	2	3	4	2	2	3	4			00011000		
	SHAR.B <EA _d >	2	2	3	4	2	2	3	4			00011001		
	SHAR.W <EA _d >	2	2	3	4	2	2	3	4			00011001		
	SHLL.B <EA _d >	2	2	3	4	2	2	3	4			00011010		
	SHLL.W <EA _d >	2	2	3	4	2	2	3	4			00011010		
	SHLR.B <EA _d >	2	2	3	4	2	2	3	4			00011011		
	SHLR.W <EA _d >	2	2	3	4	2	2	3	4			00011011		
	ROTL.B <EA _d >	2	2	3	4	2	2	3	4			00011100		
	ROTL.W <EA _d >	2	2	3	4	2	2	3	4			00011100		
	ROTR.B <EA _d >	2	2	3	4	2	2	3	4			00011101		
	ROTR.W <EA _d >	2	2	3	4	2	2	3	4			00011101		
Logic operation instruction	ROTXL.B <EA _d >	2	2	3	4	2	2	3	4			00011110		
	ROTXL.W <EA _d >	2	2	3	4	2	2	3	4			00011110		
	ROTXR.B <EA _d >	2	2	3	4	2	2	3	4			00011111		
	ROTXR.W <EA _d >	2	2	3	4	2	2	3	4			00011111		
	AND.B <EA _s >, R _d	2	2	3	4	2	2	3	4	3		01010rdrd		
	AND.W <EA _s >, R _d	2	2	3	4	2	2	3	4		4	01010rdrd		
	OR.B <EA _s >, R _d	2	2	3	4	2	2	3	4	3		01000rdrd		
	OR.W <EA _s >, R _d	2	2	3	4	2	2	3	4		4	01000rdrd		
Logic operation instruction	XOR.B <EA _s >, R _d	2	2	3	4	2	2	3	4	3		01100rdrd		
	XOR.W <EA _s >, R _d	2	2	3	4	2	2	3	4		4	01100rdrd		
	NOT.B <EA _d >	2	2	3	4	2	2	3	4			00010101		
	NOT.W <EA _d >	2	2	3	4	2	2	3	4			00010101		

Table A-1 (a) Machine Language Coding [General Format] (cont)

Instruction		Operation code (EA)										Operation code (OP)		
		Address- ing mode	1			2			3					
			Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16	4	5
Bit manipulate instruction	BSET.B #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 0 0 (data)		
	BSET.W #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 0 0 (data)		
	BSET.B R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 0 0 1 r _s r _s r _s		
	BSET.W R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 0 0 1 r _s r _s r _s		
	BCLR.B #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 0 1 (data)		
	BCLR.W #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 0 1 (data)		
	BCLR.B R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 0 1 1 r _s r _s r _s		
	BCLR.W R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 0 1 1 r _s r _s r _s		
	BTST.B #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 1 1 (data)		
	BTST.W #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 1 1 (data)		
	BTST.B R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 1 1 1 r _s r _s r _s r _s		
	BTST.W R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 1 1 1 r _s r _s r _s r _s		
System control instruction	BNOT.B #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 1 0 (data)		
	BNOT.W #xx, <EA _d >	2	2	3	4	2	2	3	4			1 1 1 0 (data)		
	BNOT.B R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 1 0 1 r _s r _s r _s		
	BNOT.W R _s , <EA _d >	2	2	3	4	2	2	3	4			0 1 1 0 1 r _s r _s r _s		
	LDC.B <EA _s >, CR	2	2	3	4	2	2	3	4	3		1 0 0 0 1 c c c		
	LDC.W <EA _s >, CR	2	2	3	4	2	2	3	4		4	1 0 0 0 1 c c c		
	STC.B CR, <EA _d >	2	2	3	4	2	2	3	4			1 0 0 1 1 c c c		
	STC.W CR, <EA _d >	2	2	3	4	2	2	3	4			1 0 0 1 1 c c c		
	ANDC.B #xx:8, CR										3	0 1 0 1 1 c c c		
	ANDC.W #xx:16, CR											4 0 1 0 1 1 c c c		
	ORC.B #xx:8, CR										3	0 1 0 0 1 c c c		
	ORC.W #xx:16, CR											4 0 1 0 0 1 c c c		
XORC.B #xx:8, CR										3	0 1 1 0 1 c c c			
XORC.W #xx:16, CR											4 0 1 1 0 1 c c c			

Table A-1 (b) Machine Language Coding [Special Format: Short Format]

Instrunction	Byte	Operation code			
		1	2	3	4
MOV:E.B #xx:8,Rd	2	01010rdrdrd	data		
MOV:I.W #xx:16,Rd	3	01011rdrdrd	data (H)	data (L)	
MOV:L.B @aa:8,Rd	2	01100rdrdrd	address (L)		
MOV:L.W @aa:8,Rd	2	01101rdrdrd	address (L)		
MOV:S.B Rs,@aa:8	2	01110rsfsfs	address (L)		
MOV:S.W Rs,@aa:8	2	01111rsfsfs	address (L)		
MOV:F.B @(d:8,R6),Rd	2	10000rdrdrd	disp		
MOV:F.W @(d:8,R6),Rd	2	10001rdrdrd	disp		
MOV:F.B Rs,@(d:8,R6)	2	10010rsfsfs	disp		
MOV:F.W Rs,@(d:8,R6)	2	10011rsfsfs	disp		
CMP:E #xx:8,Rd	2	01000rdrdrd	data		
CMP:I #xx:16,Rd	3	01001rdrdrd	data (H)	data (L)	

Table A-1 (c) Machine Language Coding [Special Format: Branch Instruction]

Instruction		Byte	Operation code			
			1	2	3	4
Bcc d:8	BRA (BT)	2	00100000	disp		
	BRN (BF)		00100001	disp		
	BHI		00100010	disp		
	BLS		00100011	disp		
	BCC (BHS)		00100100	disp		
	BCS (BLO)		00100101	disp		
	BNE		00100110	disp		
	BEQ		00100111	disp		
	BVC		00101000	disp		
	BVS		00101001	disp		
	BPL		00101010	disp		
	BMI		00101011	disp		
	BGE		00101100	disp		
	BLT		00101101	disp		
	BGT		00101110	disp		
	BLE		00101111	disp		
Bcc d:16	BRA (BT)	3	00110000	disp (H)	disp (L)	
	BRN (BF)		00110001	disp (H)	disp (L)	
	BHI		00110010	disp (H)	disp (L)	
	BLS		00110011	disp (H)	disp (L)	
	BCC (BHS)		00110100	disp (H)	disp (L)	
	BCS (BLO)		00110101	disp (H)	disp (L)	
	BNE		00110110	disp (H)	disp (L)	
	BEQ		00110111	disp (H)	disp (L)	
	BVC		00111000	disp (H)	disp (L)	
	BVS		00111001	disp (H)	disp (L)	
	BPL		00111010	disp (H)	disp (L)	
	BMI		00111011	disp (H)	disp (L)	
	BGE		00111100	disp (H)	disp (L)	
	BLT		00111101	disp (H)	disp (L)	
	BGT		00111110	disp (H)	disp (L)	
	BLE		00111111	disp (H)	disp (L)	
JMP @Rn		2	00010001	11010rrr		
JMP @aa:16		3	00010000	address (H)	address (L)	

Table A-1 (c) Machine Language Coding [Special Format: Branch Instruction]

Instruction	Byte	Operation code			
		1	2	3	4
JMP @(d:8,Rn)	3	00010001	11100rrr	disp	
JMP @(d:16,Rn)	4	00010001	11110rrr	disp (H)	disp (L)
BSR d:8	2	00001110	disp		
BSR d:16	3	00011110	disp (H)	disp (L)	
JSR @Rn	2	00010001	11011rrr		
JSR @aa:16	3	00011000	address (H)	address (L)	
JSR @(d:8,Rn)	3	00010001	11101rrr	disp	
JSR @(d:16,Rn)	4	00010001	11111rrr	disp (H)	disp (L)
RTS	1	00011001			
RTD #xx:8	2	00010100	data		
RTD #xx:16	3	00011100	data (H)	data (L)	
SCB/cc Rn,disp	SCB/F SCB/NE SCB/EQ	3	00000001	10111rrr	disp
			00000110	10111rrr	disp
			00000111	10111rrr	disp
PJMP @aa:24	4	00010011	page	address (H)	address (L)
PJMP @Rn	2	00010001	11000rrr		
PJSR @aa:24	4	00000011	page	address (H)	address (L)
PJSR @Rn	2	00010001	11001rrr		
PRTS	2	00010001	00011001		
PRTD #xx:8	3	00010001	00010100	data	
PRTD #xx:16	4	00010001	00011100	data (H)	data (L)

Table A-1 (d) Machine Language Coding [Special Format: System Control Instructions]

Instruction	Byte	Operation code			
		1	2	3	4
TRAPA #xx	2	00001000	0001 #VEC		
TRAP/VS	1	00001001			
RTE	1	00001010			
LINK FP,#xx:8	2	00010111	data		
LINK FP,#xx:16	3	00011111	data (H)	data (L)	
UNLK FP	1	00001111			
SLEEP	1	00011010			
NOP	1	00000000			

Tables A-2 through A-6 are maps of the operation codes. Table A-2 shows the meaning of the first byte of the instruction code, indicating both operation codes and addressing modes. Tables A-2 through A-6 indicate the meanings of operation codes in the second and third bytes.

Table A-2 Operation Code in Byte 1

LO		HI															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NOP	SCB/F See Tbl. A-6	LDM	PJSR @aa:24	#xx:8 See Tbl. A-5	#aa:8:B See Tbl. A-4	SCB/NE See Tbl. A-6	SCB/EQ See Tbl. A-6	TRAPA	TRAPVS	RTE		#xx:16 See Tbl. A-5	@aa:8:W See Tbl. A-4	BSR d:8	UNLK	
1	JMP	See Tbl. A-6 *	STM	PJMP @aa:24	RTD #xx:8 Tbl. A-4	@aa:16:B See Tbl. A-4		LINK #xx:8	JSR	RTS	SLEEP		RTD #xx:16 Tbl. A-4	@aa:16:W See Tbl. A-4	BSR d:16	LINK #xx:16	
2	BRA d:8	BRN	BHI	BLS	Bcc	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE	
3	BRA d:16	BRN	BHI	BLS	Bcc	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE	
4	R0	R1	R2	R3	R4	R5	R6	R7	R0	R1	R2	R3	R4	R5	R6	R7	
5		CMP/E #xx:8, Rn															
6		MOV.LB @aa:8, Rn															
7		MOV.SB Rn, @aa:8															
8		MOV.FB @d:8, R6, Rn															
9		MOV.FB Rn, @d:8, R6															
A		Rn															
B		@-Rn															
C		@Rn+															
D		@Rn															
E		@d:8, Rn															
F		@d:16, Rn															

Notes References to tables A-3 through A-6 indicate that the instruction code has one or more additional bytes, described in those tables.

* H'11 is the first operation code byte of the following instructions:

JMP,JSR, PJSR (register indirect addressing mode)

JMP,JSR (register indirect addressing mode with displacement)

PRTS, PRTD (all addressing modes)

0	See 10h, A-6*								ADD:Q #1	ADD:Q #2	ADD:Q #1	ADD:Q #2				
1	SWAP	EXTS	EXTU	CLR	NEG	NOT	TST	TAS	SHAL	SHAR	SHLL	SHLR	ROTL	ROTR	ROTXL	ROTXR
2	R0	R1	R2	ADD				R0	R1	ADDS						
3				SUB						R2	R3	R4	R5	R6	R7	
4				OR						BSET (Register indirect specification of bit number)						
5				AND						BCLR (Register indirect specification of bit number)						
6				XOR						BNOT (Register indirect specification of bit number)						
7				CMP						BTST (Register indirect specification of bit number)						
8				MOV						LDC						
9				XCH						STC						
A				ADDX						MULXU						
B				SUBX						DIVXU						
C	b0	b1	b2	b3	b4	b5	b6	b7	BSET (immediate specification of bit number)							
D							BCLR (immediate specification of bit number)									
E							BNOT (immediate specification of bit number)									
F							BTST (immediate specification of bit number)									

Note:* The operation code is in byte 3, given in table A-6.

HI	LO																		
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
0	See Tbl. A-6*				CMP #xx:8	CMP #xx:16	MOV #xx:8	MOV #xx:16	ADD:Q #1	ADD:Q #2			ADD:Q #-1	ADD:Q #-2					
1			CLR	NEG	NOT	TST	TAS	SHAL	SHAR	SHL	SHLR	ROTL	ROTR	ROTXL	ROTXR				
2	R0		R1	R2	ADD R3		R4	R5	R6	R7	ADDS R0		R1	R2	R3	R4	R5	R6	R7
3				SUB							SUBS								
4				OR							BSET (Register indirect specification of bit number)								
5				AND							BCLR (Register indirect specification of bit number)								
6				XOR							BNOT (Register indirect specification of bit number)								
7				CMP							BTST (Register indirect specification of bit number)								
8				MOV (load)							LDC								
9				MOV (store)							STC								
A				ADDX							MULXU								
B				SUBX							DIVXU								
C	b0		b1	b2	b3	b4	b5	BSET (immediate b6		b7	BCLR (immediate b8		b9	b10	b11	b12	b13	b14	b15
D								BCLR (immediate specification of bit number)											
E								BNOT (immediate specification of bit number)											
F								BTST (immediate specification of bit number)											

Note: * The operation code is in byte 3, given in table A-6.

[illegible]

	HI																LO																																															
	0				1				2				3				4				5				6				7				8				9				A				B				C				D				E				F			
0																																																																
1																	PRTD #xx:8																PRTS																PRTD #xx:16															
2																																																																
3																																																																
4																																																																
5																																																																
6																																																																
7																																																																
8	MOVEPE																																																															
	R0				R1				R2				R3				R4				R5				R6				R7																																			
9																	MOVTR																																															
A																	DADD																																															
B																	DSUB																																															
	R0				R1				R2				R3				R4				R5				R6				R7				SCB																															
C																	PJMP @Rn																																															
D																	JMP @Rn																																															
																	JSR @Rn																																															
E																	JMP @(d:8,Rn)																																															
																	JSR @(d:8,Rn)																																															
F																	JMP @(d:16,Rn)																																															
																	JSR @(d:16,Rn)																																															

A.4 Instruction Execution Cycles

Tables A-7 (1) through (6) list the number of cycles required by the CPU to execute each instruction in each addressing mode.

The meaning of the symbols in the tables is explained below. The values of I, J, and K are used to calculate the number of execution cycles needed to fetch an instruction or read or write an operand that is not located in the memory area accessed in two states via a 16-bit bus. The formulas for these calculations are given next.

- **Calculation on Instruction Execution States**

One state is one cycle of the system clock (ϕ). When $\phi = 10\text{ MHz}$, one state is 100 ns.

Instruction Fetch	Operand Read/Write	Number of States	
16-Bit bus, 2-state-access memory area	16-Bit bus, 2-state-access memory area or general register		(Value in table A-7) + (Value in table A-8)
	16-Bit bus, 3-state-access memory area	Byte	(Value in table A-7) + (Value in table A-8) + I
		Word	(Value in table A-7) + (Value in table A-8) + I/2
	8-Bit bus, 2-state-access memory area	Byte	(Value in table A-7) + (Value in table A-8)
		Word	((Value in table A-7) + (Value in table A-8) + I
	8-Bit bus, 3-state-access memory area or on-chip supporting module	Byte	(Value in table A-7) + (Value in table A-8) + I
		Word	(Value in table A-7) + (Value in table A-8) + 2I

Instruction Fetch	Operand Read/Write	Number of States	
16-Bit bus, 3-state-access memory area	16-Bit bus, 2-state-access memory area or general register		(Value in table A-7) + (Value in table A-8) + $(J + K) / 2$
		Byte	(Value in table A-7) + (Value in table A-8) + $I + (J + K) / 2$
	16-Bit bus, 3-state-access memory area	Word	(Value in table A-7) + (Value in table A-8) + $(I + J + K) / 2$
	8-Bit bus, 2-state-access memory area	Byte	(Value in table A-7) + (Value in table A-8) + $(J + K) / 2$
		Word	(Value in table A-7) + (Value in table A-8) + $I + (J + K) / 2$
	8-Bit bus, 3-state-access memory area or on-chip supporting module	Byte	(Value in table A-7) + (Value in table A-8) + $I + (J + K) / 2$
		Word	(Value in table A-7) + (Value in table A-8) + $2I + (J + K) / 2$
8-Bit bus, 2-state-access memory area	16-Bit bus, 2-state-access memory area or general register		(Value in table A-7) + $J + K$
	16-Bit bus, 3-state-access memory area	Byte	(Value in table A-7) + $I + J + K$
		Word	(Value in table A-7) + $I/2 + J + K$
	8-Bit bus, 2-state-access memory area	Byte	(Value in table A-7) + $J + K$
		Word	(Value in table A-7) + $I + J + K$
	8-Bit bus, 3-state-access memory area or on-chip supporting module	Byte	(Value in table A-7) + $I + J + K$
		Word	(Value in table A-7) + $2I + J + K$

Instruction Fetch	Operand Read/Write	Number of States	
8-Bit bus, 3-state-access memory area	16-Bit bus, 2-state-access memory area or general register	(Value in table A-7) + 2(J + K)	
	16-Bit bus, 3-state-access memory area	Byte	(Value in table A-7) + I + 2(J + K)
		Word	(Value in table A-7) + I/2 + 2(J + K)
	8-Bit bus, 2-state-access memory area	Byte	(Value in table A-7) + 2(J + K)
		Word	(Value in table A-7) + I + 2(J + K)
	8-Bit bus, 3-state-access memory area or on-chip supporting module	Byte	(Value in table A-7) + I + 2(J + K)
		Word	(Value in table A-7) + 2(I + J + K)

- Notes:**
1. When an instruction is fetched via a 16-bit bus, the number of execution states varies by 1 or 2 depending on whether the instruction is stored at an even or odd address. This difference must be noted when software is used for timing, and in other cases in which the exact number of states is important.
 2. If wait states or TP states are inserted in access to the three-state-access memory area, add the necessary number of states.
 3. When an instruction is fetched from a memory area that is accessed via a 16-bit bus in three states, fractions in the term (J + K)/2 should be rounded down.

• **Tables of Instruction Execution Cycles**

Tables A-7 (1) through (6) should be read as shown below:

J + K: Number of instruction fetch cycles.

I: Total number of bytes written and read when operand is in memory.

				Addressing mode									
				Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
Instruction	I	J	K	1	1	2	3	1	1	2	3	2	3
ADD.B	1	1		2	5	5	6	5	6	5	6	3	
ADD.W	2	1		2	5	5	6	5	6	5	6		4
ADD:Q.B	2	1		2	7	7	8	7	8	7	8		
ADD:Q.W	4	1		2	7	7	8	7	8	7	8		
DADD		2		4									

Shading in the I column means the operand cannot be in memory.

Shading indicates addressing modes that cannot be used with this instruction.

- Examples of Calculation of Number of States Required for Execution

Example 1: Instruction fetch from memory area accessed via 16-bit bus in 2 states

Operand Read/Write	Start Addr.	Assembler Notation			Table A-7 + Table A-8	Number of States
		Address	Code	Mnemonic		
16-Bit bus, 2-state access memory area or general register	Even	H'0100	H'D821	ADD @R0, R1	5 + 1	6
	Odd	H'0101	H'D821	ADD @R0, R1	5 + 0	5

Example 2: Instruction fetch from memory area accessed via 16-bit bus in 2 states when stack is in area accessed via 8-bit bus in 3 states

Operand Read/Write	Start Addr.	Assembler Notation			Table A-7 + Table A-8 + 2I	Number of States
		Address	Code	Mnemonic		
On-chip supporting module or 8-bit bus, 3-state-access memory area (word)	Even	H'FC00	H'11D8	JSR @R0	9 + 0 + 2 × 2	13
	Odd	H'FC01	H'11D8	JSR @R0	9 + 1 + 2 × 2	14

Example 3: Instruction fetch from memory area accessed via 8-bit bus in 3 states

Operand Read/Write	Assembler Notation			Table A-7 + 2(J + K)	Number of States
	Address	Code	Mnemonic		
16-Bit bus, 2-state-access memory area or general register	H'9002	H'D821	ADD @R0, R1	5 + 2 × (1 + 1)	9

Example 4: Instruction fetch from memory area accessed via 16-bit bus in 3 states

Operand Read/Write	Start Addr.	Assembler Notation			Table A-7 + Table A-8 + (J + K)/2	Number of States
		Address	Code	Mnemonic		
16-Bit bus, 2-state access memory area or general register	Even	H'0100	H'D821	ADD @R0, R1	5 + 1 + (1 + 1)/2	7
	Odd	H'0101	H'D821	ADD @R0, R1	5 + 0 + (1 + 1)/2	6

Table A-7 Instruction Execution Cycles (1)

Instruction	I	J	K	Addressing mode									
				Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
				1	1	2	3	1	1	2	3	2	3
ADD:G.B <EAs>, Rd	1	1		2	5	5	6	5	6	5	6	3	
ADD:G.W <EAs>, Rd	2	1		2	5	5	6	5	6	5	6		4
ADD:Q.B #xx, <EAd>	2	1		2	7	7	8	7	8	7	8		
ADD:Q.W #xx, <EAd>	4	1		2	7	7	8	7	8	7	8		
ADDS.B <EAs>, Rd	1	1		3	5	5	6	5	6	5	6	3	
ADDS.W <EAs>, Rd	2	1		3	5	5	6	5	6	5	6		4
ADDX.B <EAs>, Rd	1	1		2	5	5	6	5	6	5	6	3	
ADDX.W <EAs>, Rd	2	1		2	5	5	6	5	6	5	6		4
AND.B <EAs>, Rd	1	1		2	5	5	6	5	6	5	6	3	
AND.W <EAs>, Rd	2	1		2	5	5	6	5	6	5	6		4
ANDC #xx, CR		1										5	9
BCLR.B #xx, <EAd>	*	2	1	4	7	7	8	7	8	7	8		
BCLR.W #xx, <EAd>	*	4	1	4	7	7	8	7	8	7	8		
BNOT.B #xx, <EAd>	*	2	1	4	7	7	8	7	8	7	8		
BNOT.W #xx, <EAd>	*	4	1	4	7	7	8	7	8	7	8		
BSET.B #xx, <EAd>	*	2	1	4	7	7	8	7	8	7	8		
BSET.W #xx, <EAd>	*	4	1	4	7	7	8	7	8	7	8		
BTST.B #xx, <EAd>	*	1	1	3	5	5	6	5	6	5	6		
BTST.W #xx, <EAd>	*	2	1	3	5	5	6	5	6	5	6		
CLR.B <EAd>	1	1		2	5	5	6	5	6	5	6		
CLR.W <EAd>	2	1		2	5	5	6	5	6	5	6		
CMP:G.B <EAs>, Rd	1	1		2	5	5	6	5	6	5	6	3	
CMP:G.W <EAs>, Rd	2	1		2	5	5	6	5	6	5	6		4
CMP:G.B #XX:8, <EA>	1	2			6	6	7	6	7	6	7		
CMP:G.B #XX:16, <EA>	2	3			7	7	8	7	8	7	8		

* Rs can also be coded as the source operand.

Table A-7 Instruction Execution Cycles (2)

Instruction	I	J K			Addressing mode									
					Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
					1	1	2	3	1	1	2	3	2	3
CMP:E #xx:8, Rd		0											2	
CMP:I #xx:16, Rd		0												3
DADD Rs, Rd		2	4											
DIVXU.B <EAs>, Rd	1	1	20	23	23	24	23	24	23	24	23	24	21	
DIVXU.W <EAs>, Rd	2	1	26	29	29	30	29	30	29	30	29	30		28
DSUB Rs, Rd		2	4											
EXTS Rd		1	3											
EXTU Rd		1	3											
LDC.B <EAs>, CR	1	1	3	6	6	7	6	7	6	7	6	7	4	
LDC.W <EAs>, CR	2	1	4	7	7	8	7	8	7	8	7	8		6
MOV:G.B	1	1	2	5	5	6	5	6	5	6	5	6	3	
MOV:G.W	2	1	2	5	5	6	5	6	5	6	5	6		4
MOV.G.B #xx:8, <EAd>	1	2		7	7	8	7	8	7	8	7	8		
MOV.G.W #xx:16, <EAd>	2	3		8	8	9	8	9	8	9	8	9		
MOV:E #xx:8, Rd		0											2	
MOV:I #xx:16, Rd		0												3
MOV:L.B @aa:8, Rd	1	0									5			
MOV:L.W @aa:8, Rd	2	0									5			
MOV:S.B Rs, @aa:8	1	0									5			
MOV:S.W Rs, @aa:8	2	0									5			
MOV:F.B @(d:8, R6), Rd	1	0				5								
MOV:F.W @(d:8, R6), Rd	2	0				5								
MOV:F.B Rs, @(d:8, R6)	1	0				5								
MOV:F.W Rs, @(d:8, R6)	2	0				5								

Table A-7 Instruction Execution Cycles (3)

Instruction	I	J	K	Addressing mode									
				Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
				1	1	2	3	1	1	2	3	2	3
MOVFPPE * <EAs>, RD	0	2			13 └ 20	13 └ 20	14 └ 21	13 └ 20	14 └ 21	13 └ 20	14 └ 21		
MOVTPPE * <EAs>, RD	0	2			13 └ 20	13 └ 20	14 └ 21	13 └ 20	14 └ 21	13 └ 20	14 └ 21		
MULXU.B <EAs>, RD	1	1	16	19	19	20	19	20	19	20	18		
MULXU.W <EAs>, RD	2	1	23	25	25	26	25	26	25	26			25
NEG.B <EAd>	2	1	2	7	7	8	7	8	7	8			
NEG.W <EAd>	4	1	2	7	7	8	7	8	7	8			
NOT.B <EAd>	2	1	2	7	7	8	7	8	7	8			
NOT.W <EAd>	4	1	2	7	7	8	7	8	7	8			
OR.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3		
OR.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6			4
ORC #xx, CR		1										5	9
ROTL.B <EAd>	2	1	2	7	7	8	7	8	7	8			
ROTL.W <EAd>	4	1	2	7	7	8	7	8	7	8			
ROTR.B <EAd>	2	1	2	7	7	8	7	8	7	8			
ROTR.W <EAd>	4	1	2	7	7	8	7	8	7	8			
ROTXL.B <EAd>	2	1	2	7	7	8	7	8	7	8			
ROTXL.W <EAd>	4	1	2	7	7	8	7	8	7	8			
ROTXR.B <EAd>	2	1	2	7	7	8	7	8	7	8			
ROTXR.W <EAd>	4	1	2	7	7	8	7	8	7	8			
SHAL.B <EAd>	2	1	2	7	7	8	7	8	7	8			
SHAL.W <EAd>	4	1	2	7	7	8	7	8	7	8			
SHAR.B <EAd>	2	1	2	7	7	8	7	8	7	8			
SHAR.W <EAd>	4	1	2	7	7	8	7	8	7	8			
SHLL.B <EAd>	2	1	2	7	7	8	7	8	7	8			
SHLL.W <EAd>	4	1	2	7	7	8	7	8	7	8			

* MOVFPPE and MOVTPPE are executed synchronous with the E-clock, so the number of execution states will change depending on the timing of execution.

Table A-7 Instruction Execution Cycles (4)

Instruction	I	J	K	Addressing mode									
				Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
SHLR.B <EAd>	2	1	2	7	7	8	7	8	7	8			
SHLR.W <EAd>	4	1	2	7	7	8	7	8	7	8			
STC.B CR, <EAd>	1	1	4	7	7	8	7	8	7	8			
STC.W CR, <EAd>	2	1	4	7	7	8	7	8	7	8			
SUB.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3		
SUB.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6			4
SUBS.B <EAs>, Rd	1	1	3	5	5	6	5	6	5	6	3		
SUBS.W <EAs>, Rd	2	1	3	5	5	6	5	6	5	6			4
SUBX.B <EAs>, Rd	1	1	2	5	5	6	5	6	5	6	3		
SUBX.W <EAs>, Rd	2	1	2	5	5	6	5	6	5	6			4
SWAP Rd		1	3										
TAS <EAd>	2	1	4	7	7	8	7	8	7	8			
TST.B <EAd>	1	1	2	5	5	6	5	6	5	6			
TST.W <EAd>	2	1	2	5	5	6	5	6	5	6			
XCH Rs, Rd		1	4										
XOR.B <EAs>, Rd	1	1	2	5	6	5	5	6	5	6	3		
XOR.W <EAs>, Rd	2	1	2	5	6	5	5	6	5	6			4
XORC #xx, CR		1										5	9

* 7

DIVXU.B	Zero divide, minimum mode	6/7	1	20	23	23	24	23	24	23	24	21	
DIVXU.B	Zero divide, maximum mode	10/11	1	25	28	28	29	28	29	28	29	21	
DIVXU.W	Zero divide, minimum mode	6/8	1	20	23	23	24	23	24	23	24		27
DIVXU.W	Zero divide, maximum mode	10/12	1	25	28	28	29	28	29	28	29		27
DIVXU.B	Overflow	1	1	8	11	11	12	11	12	11	12	9	
DIVXU.W	Overflow	2	1	8	11	11	12	11	12	11	12		10

* 7 For register and immediate operands



For memory operand

Table A-7 Instruction Execution Cycles (5)

Instruction	(Condition)	Execution Cycles	I	J + K
Bcc d:8	Condition false, branch not taken	3		2
	Condition true, branch taken	7		5
Bcc d:16	Condition false, branch not taken	3		3
	Condition true, branch taken	7		6
BSR	d:8	9	2	4
	d:16	9	2	5
JMP	@aa:16	7		5
	@Rn	6		5
	@(d:8, Rn)	7		5
	@(d:16, Rn)	8		6
JSR	@aa:16	9	2	5
	@Rn	9	2	5
	@(d:8, Rn)	9	2	5
	@(d:16, Rn)	10	2	6
LDM		$6 + 4n^*$	2n	2
LINK	#xx:8	6	2	2
	#xx:16	7	2	3
NOP		2		1
RTD	#xx:8	9	2	4
	#xx:16	9	2	5
RTE	Minimum mode	13	4	4
	Maximum mode	15	6	4
RTS		8	2	4
SCB	Condition false, branch not taken	3		3
	Count = -1, branch not taken	4		3
	Other than the above, branch taken	8		6
SLEEP	Cycles preceding transition to power-down mode	2		0
STM		$6 + 3n^*$	2n	2

* n is the number of registers specified in the register list.

Table A-7 Instruction Execution Cycles (6)

Instruction	(Condition)	Execution Cycles	I	J + K
TRAPA	Minimum mode	17	6	4
	Maximum mode	22	10	4
TRAP/VS	V = 0, trap not taken	3		1
	V = 1, trap taken, minimum mode	18	6	4
	V = 1, trap taken, maximum mode	23	10	4
UNLK		5	2	1
PJMP	@aa:24	9		6
	@Rn	8		5
PJSR	@aa:24	15	4	6
	@Rn	13	4	5
PRTS		12	4	5
PRTD	#xx:8	13	4	5
	#xx:16	13	4	6

Table A-8 (a) Adjustment Value (Branch Instruction)

Instruction	Address	Adjustment Value
BSR, JMP, JSR, RTS, RTD, RTE	even	0
TRAPA, PJMP, PJSR, PRTS, PRTD	odd	1
Bcc, SCB, TRAP/VS (branch taken)	even	0
	odd	1

Table A-8 (b) Adjustment Value (Other Instructions by Addressing Modes)

Instructor	Start Address	Rn	@Rn	@(d:8, Rn)	@(d:16, Rn)	@-Rn	@Rn+	@aa:8	@aa:16	#xx:8	#xx:16
MOV.B #xx:8, <EA>	even		1	1	1	1	1	1	1		
MOVTPE, MOVFPE	odd		1	1	1	1	1	1	1		
MOV.W #xx:16, <EA>	even		2	0	2	2	2	0	2		
	odd		0	2	0	0	0	2	0		
Instruction other than above	even	0	1	0	1	1	1	0	1	0	0
	odd	0	0	1	0	0	0	1	0	0	0

Appendix B Register Field

B.1 Register Addresses and Bit Names

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FE80	P1DDR	P17DDR	P16DDR	P15DDR	P14DDR	P13DDR	P12DDR	P11DDR	P10DDR	Port 1
H'FE81	P2DDR	P27DDR	P26DDR	P25DDR	P24DDR	P23DDR	P22DDR	P21DDR	P20DDR	Port 2
H'FE82	P1DR	P17	P16	P15	P14	P13	P12	P11	P10	Port 1
H'FE83	P2DR	P27	P26	P25	P24	P23	P22	P21	P20	Port 2
H'FE84	P3DDR	P37DDR	P36DDR	P35DDR	P34DDR	P33DDR	P32DDR	P31DDR	P30DDR	Port 3
H'FE85	P4DDR	P47DDR	P46DDR	P45DDR	P44DDR	P43DDR	P42DDR	P41DDR	P40DDR	Port 4
H'FE86	P3DR	P37	P36	P35	P34	P33	P32	P31	P30	Port 3
H'FE87	P4DR	P47	P46	P45	P44	P43	P42	P41	P40	Port 4
H'FE88	P5DDR	P57DDR	P56DDR	P55DDR	P54DDR	P53DDR	P52DDR	P51DDR	P50DDR	Port 5
H'FE89	P6DDR	P67DDR	P66DDR	P65DDR	P64DDR	P63DDR	P62DDR	P61DDR	P60DDR	Port 6
H'FE8A	P5DR	P57	P56	P55	P54	P53	P52	P51	P50	Port 5
H'FE8B	P6DR	P67	P66	P65	P64	P63	P62	P61	P60	Port 6
H'FE8C	—	—	—	—	—	—	—	—	—	—
H'FE8D	P8DDR	P87DDR	P86DDR	P85DDR	P84DDR	P83DDR	P82DDR	P81DDR	P80DDR	Port 8
H'FE8E	P7DR	—	—	—	—	P73	P72	P71	P70	Port 7
H'FE8F	P8DR	P87	P86	P85	P84	P83	P82	P81	P80	Port 8
H'FE90	ADDRA H	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	A/D
H'FE91	ADDRA L	AD1	AD0	—	—	—	—	—	—	
H'FE92	ADDRB H	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	
H'FE93	ADDRB L	AD1	AD0	—	—	—	—	—	—	
H'FE94	ADDRC H	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	
H'FE95	ADDRC L	AD1	AD0	—	—	—	—	—	—	
H'FE96	ADDRD H	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	
H'FE97	ADDRD L	AD1	AD0	—	—	—	—	—	—	
H'FE98	ADCSR	ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0	
H'FE99	ADCR	TRGE	—	—	—	—	—	—	—	
H'FE9A	—	—	—	—	—	—	—	—	—	
H'FE9B	—	—	—	—	—	—	—	—	—	
H'FE9C	—	—	—	—	—	—	—	—	—	
H'FE9D	—	—	—	—	—	—	—	—	—	
H'FE9E	—	—	—	—	—	—	—	—	—	
H'FE9F	—	—	—	—	—	—	—	—	—	

Note: A/D: A/D converter

(Continued on next page)

(Continued from preceding page)

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEA0	TCR	ICIE	OCIEB	OCIEA	OVIE	OEB	OEA	CKS1	CKS0	FRT1
H'FEA1	TCSR	ICF	OCFB	OCFA	OVF	OLVLB	OLVLA	IEDG	CCLRA	
H'FEA2	FRC H									
H'FEA3	FRC L									
H'FEA4	OCRA H									
H'FEA5	OCRA L									
H'FEA6	OCRB H									
H'FEA7	OCRB L									
H'FEA8	ICR H									
H'FEA9	ICR L									
H'FEAA	—	—	—	—	—	—	—	—	—	
H'FEAB	—	—	—	—	—	—	—	—	—	
H'FEAC	—	—	—	—	—	—	—	—	—	
H'FEAD	—	—	—	—	—	—	—	—	—	
H'FEAE	—	—	—	—	—	—	—	—	—	
H'FEAF	—	—	—	—	—	—	—	—	—	
H'FEB0	TCR	ICIE	OCIEB	OCIEA	OVIE	OEB	OEA	CKS1	CKS0	FRT 2
H'FEB1	TCSR	ICF	OCFB	OCFA	OVF	OLVLB	OLVLA	IEDG	CCLRA	
H'FEB2	FRC H									
H'FEB3	FRC L									
H'FEB4	OCRA H									
H'FEB5	OCRA L									
H'FEB6	OCRB H									
H'FEB7	OCRB L									
H'FEB8	ICR H									
H'FEB9	ICR L									
H'FEBA	—	—	—	—	—	—	—	—	—	
H'FEBB	—	—	—	—	—	—	—	—	—	
H'FEBBC	—	—	—	—	—	—	—	—	—	
H'FEBD	—	—	—	—	—	—	—	—	—	
H'FEBE	—	—	—	—	—	—	—	—	—	
H'FEBF	—	—	—	—	—	—	—	—	—	

Notes: FRT1: Free-running timer channel 1
FRT2: Free-running timer channel 2

(Continued on next page)

(Continued from preceding page)

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FEC0	TCR	CMIEB	CMIEA	OVIE	CCLR1	CCLR0	CKS2	CKS1	CKS0	TMR
H'FEC1	TCSR	CMFB	CMFA	OVF	—	OS3	OS2	OS1	OS0	
H'FEC2	TCORA									
H'FEC3	TCORB									
H'FEC4	TCNT									
H'FEC5	—	—	—	—	—	—	—	—	—	
H'FEC6	—	—	—	—	—	—	—	—	—	
H'FEC7	—	—	—	—	—	—	—	—	—	SCI1
H'FEC8	SMR	C/A	CHR	PE	O/E	STOP	—	CKS1	CKS0	
H'FEC9	BRR									
H'FECA	SCR	TIE	RIE	TE	RE	—	—	CKE1	CKE0	
H'FECB	TDR									
H'FECC	SSR	TDRE	RDRF	ORER	FER	PER	—	—	—	
H'FECD	RDR									
H'FECE	—	—	—	—	—	—	—	—	—	SCI2
H'FECF	—	—	—	—	—	—	—	—	—	
H'FED0	SMR	C/A	CHR	PE	O/E	STOP	—	CKS1	CKS0	
H'FED1	BRR									
H'FED2	SCR	TIE	RIE	TE	RE	—	—	CKE1	CKE0	
H'FED3	TDR									
H'FED4	SSR	TDRE	RDRF	ORER	FER	PER	—	—	—	
H'FED5	RDR									
H'FED6	—	—	—	—	—	—	—	—	—	RFSHC
H'FED7	—	—	—	—	—	—	—	—	—	
H'FED8	RFSHCR	RFSHE	ASWC	ARFSH	RWC1	RWC0	CYC2	CYC1	CYC0	
H'FED9										
H'FEDA										
H'FEDB										
H'FEDC										
H'FEDD										
H'FEDE										
H'FEDF										

Notes: TMR: 8-Bit timer

(Continued on next page)

SCI1: Serial communication interface channel 1

SCI2: Serial communication interface channel 2

RFSHC: Refresh controller

(Continued from preceding page)

Addr.	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FF00	IPRA	—				—				INTC
H'FF01	IPRB	—				—				
H'FF02	IPRC	—				—				
H'FF03	IPRD	—				—				
H'FF04	—	—	—	—	—	—	—	—	—	
H'FF05	—	—	—	—	—	—	—	—	—	
H'FF06	—	—	—	—	—	—	—	—	—	
H'FF07	—	—	—	—	—	—	—	—	—	
H'FF08	DTEA	—	—	—		—				
H'FF09	DTEB	—				—				
H'FF0A	DTEC	—	—			—			—	
H'FF0B	DTED	—			—	—	—	—		
H'FF0C	—	—	—	—	—	—	—	—	—	
H'FF0D	—	—	—	—	—	—	—	—	—	
H'FF0E	—	—	—	—	—	—	—	—	—	
H'FF0F	—	—	—	—	—	—	—	—	—	
H'FF10	TCSR*	OVF	WT/IT	TME	—	—	CKS2	CKS1	CKS0	WDT
H'FF11	TCNT*									
H'FF12										
H'FF13										
H'FF14	WCR	—	—	—	—	WMS1	WMS0	WC1	WC0	WSC
H'FF15										BSC
H'FF16	ARBT									
H'FF17	AR3T									
H'FF18										
H'FF19	MDCR	—	—	—	—	—	MDS2	MDS1	MDS0	
H'FF1A	SBYCR	SSBY	—	—	—	—	—	—	—	
H'FF1B	BRCR	—	—	—	—	—	—	—	BRLE	
H'FF1C	NMICR	—	—	—	—	—	—	—	NMIEG	
H'FF1D	IRQCR	—	—	—	—	IRQ3E	IRQ2E	IRQ1E	IRQ0E	
H'FF1E	RSTCSR*									WDT
H'FF1F	RSTCSR*	WRST	RSTOE	—	—	—	—	—	—	

Notes: INTC: Interrupt controller

(Continued on next page)

WDT: Watchdog timer

WSC: Wait state controller

BSC: Bus controller

* Read addresses of TCSR and TCNT are shown. Write addresses of both TCSR and TCNT are H'FF10. RSTCSR is written at H'FF1E and read at H'FF1F. These three registers are password-protected. See section 16.2.4, "Notes on Register Access" for details.

B.2 Register Descriptions

Acronym of the register

Register name

Address to which the register is mapped

Name of the on-chip supporting module

ADCSR—A/D Control/Status Register

H'FE98

A/D

Bit numbers

Bit

Initial bit values

Initial value

Read/Write

Type of access permitted

7	6	5	4	3	2	1	0
ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0
0	0	0	0	0	0	0	0
R/(W)*	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Names of the bits. Dashes (—) indicate reserved bits.

Full name of the bit

Functions of the bit settings

Channel Select

CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	AN ₀	AN ₀
0	0	1	AN ₁	AN ₀ , AN ₁
0	1	0	AN ₂	AN ₀ to AN ₂
0	1	1	AN ₃	AN ₀ to AN ₃

Clock Select

0	Conversion time = 247 states (max)
1	Conversion time = 138 states (min)

Scan Mode

0	Single mode
1	Scan mode

Type of access permitted

R	Read only
W	Write only
R/W	Both read and write

P1DDR—Port 1 Data Direction Register**H'FE80****Port 1**

Bit	7	6	5	4	3	2	1	0
	P17DDR	P16DDR	P15DDR	P14DDR	P13DDR	P12DDR	P11DDR	P10DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 1 Input/Output Selection

0	Input port
1	Output port

P2DDR—Port 2 Data Direction Register**H'FE81****Port 2**

Bit	7	6	5	4	3	2	1	0
	P27DDR	P26DDR	P25DDR	P24DDR	P23DDR	P22DDR	P21DDR	P20DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 2 Input/Output Selection

0	Input port
1	Output port

P1DR—Port 1 Data Register**H'FE82****Port 1**

Bit	7	6	5	4	3	2	1	0
	P17	P16	P15	P14	P13	P12	P11	P10
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P2DR—Port 2 Data Register**H'FE83****Port 2**

Bit	7	6	5	4	3	2	1	0
	P27	P26	P25	P24	P23	P22	P21	P20
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P3DDR—Port 3 Data Direction Register**H'FE84****Port 3**

Bit	7	6	5	4	3	2	1	0
	P37DDR	P36DDR	P35DDR	P34DDR	P33DDR	P32DDR	P31DDR	P30DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 3 Input/Output Selection

0	Input port
1	Output port

P4DDR—Port 4 Data Direction Register**H'FE85****Port 4**

Bit	7	6	5	4	3	2	1	0
	P47DDR	P46DDR	P45DDR	P44DDR	P43DDR	P42DDR	P41DDR	P40DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 4 Input/Output Selection

0	Input port
1	Output port

P3DR—Port 3 Data Register

H'FE86

Port 3

Bit	7	6	5	4	3	2	1	0
	P37	P36	P35	P34	P33	P32	P31	P30
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P4DR—Port 4 Data Register

H'FE87

Port 4

Bit	7	6	5	4	3	2	1	0
	P47	P46	P45	P44	P43	P42	P41	P40
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P5DDR—Port 5 Data Direction Register

H'FE88

Port 5

Bit	7	6	5	4	3	2	1	0
	P57DDR	P56DDR	P55DDR	P54DDR	P53DDR	P52DDR	P51DDR	P50DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 5 Input/Output Selection

0	Input port
1	Output port

P6DDR—Port 6 Data Direction Register**H'FE89****Port 6**

Bit	7	6	5	4	3	2	1	0
	P67DDR	P66DDR	P65DDR	P64DDR	P63DDR	P62DDR	P61DDR	P60DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 6 Input/Output Selection

0	Input port
1	Output port

P5DR—Port 5 Data Register**H'FE8A****Port 5**

Bit	7	6	5	4	3	2	1	0
	P57	P56	P55	P54	P53	P52	P51	P50
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P6DR—Port 6 Data Register**H'FE8B****Port 6**

Bit	7	6	5	4	3	2	1	0
	P67	P66	P65	P64	P63	P62	P61	P60
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P8DDR—Port 8 Data Direction Register**H'FE8D****Port 8**

Bit	7	6	5	4	3	2	1	0
	P87DDR	P86DDR	P85DDR	P84DDR	P83DDR	P82DDR	P81DDR	P80DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 8 Input/Output Selection

0	Input port
1	Output port

P7DR—Port 7 Data Register**H'FE8E****Port 7**

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	P7 ₃	P7 ₂	P7 ₁	P7 ₀

Read/Write	—	—	—	—	R	R	R	R
------------	---	---	---	---	---	---	---	---

P8DR—Port 8 Data Register**H'FE8F****Port 8**

Bit	7	6	5	4	3	2	1	0
	P8 ₇	P8 ₆	P8 ₅	P8 ₄	P8 ₃	P8 ₂	P8 ₁	P8 ₀

Initial value	0	0	0	0	0	0	0	0
---------------	---	---	---	---	---	---	---	---

Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
------------	-----	-----	-----	-----	-----	-----	-----	-----

ADDRn (H)—A/D Data Register n (High)**A/D****(n = A, B, C, D) H'FE90, H'FE92, H'FE94, H'FE96**

Bit	7	6	5	4	3	2	1	0
	AD ₉	AD ₈	AD ₇	AD ₆	AD ₅	AD ₄	AD ₃	AD ₂

Initial value	0	0	0	0	0	0	0	0
---------------	---	---	---	---	---	---	---	---

Read/Write	R	R	R	R	R	R	R	R
------------	---	---	---	---	---	---	---	---

Upper 8 bits of 10-bit A/D conversion result

ADDRn (L)—A/D Data Register n (Low)**A/D****(n = A, B, C, D) H'FE91, H'FE93, H'FE95, H'FE97**

Bit	7	6	5	4	3	2	1	0
	AD ₁	AD ₀	—	—	—	—	—	—

Initial value	0	0	0	0	0	0	0	0
---------------	---	---	---	---	---	---	---	---

Read/Write	R	R	R	R	R	R	R	R
------------	---	---	---	---	---	---	---	---

Lower 2 bits of 10-bit A/D conversion result

ADCSR—A/D Control/Status Register

H'FE98

A/D

Bit	7	6	5	4	3	2	1	0
	ADF	ADIE	ADST	SCAN	CKS	CH2	CH1	CH0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Channel Select				
CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	AN0	AN0
0	0	1	AN1	AN0, AN1
0	1	0	AN2	AN0 to AN2
0	1	1	AN3	AN0 to AN3

Clock Select	
0	Conversion time = 266 states (max)
1	Conversion time = 134 states (max)

Scan Mode	
0	Single mode
1	Scan mode

A/D Start	
0	A/D conversion is halted.
1	1. Single mode: One A/D conversion is performed, then this bit is automatically cleared to 0. 2. Scan mode: A/D conversion starts and continues cyclically on all selected channels until 0 is written in this bit.

A/D Interrupt Enable	
0	A/D-end interrupt is disabled.
1	A/D-end interrupt is enabled.

A/D End Flag	
0	Cleared from 1 to 0 when: 1. The chip is reset or enters a standby mode. 2. CPU reads ADF = 1, then writes 0 in ADF. 3. ADI interrupt is served by DTC.
1	Set to 1 at the following times: 1. Single mode: at the completion of A/D conversion. 2. Scan mode: when all selected channels have been converted.

* Only writing of 0 to clear the flag is enabled.

ADCR—A/D Control Register

H'FE99

A/D

Bit	7	6	5	4	3	2	1	0
	TRGE							
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—

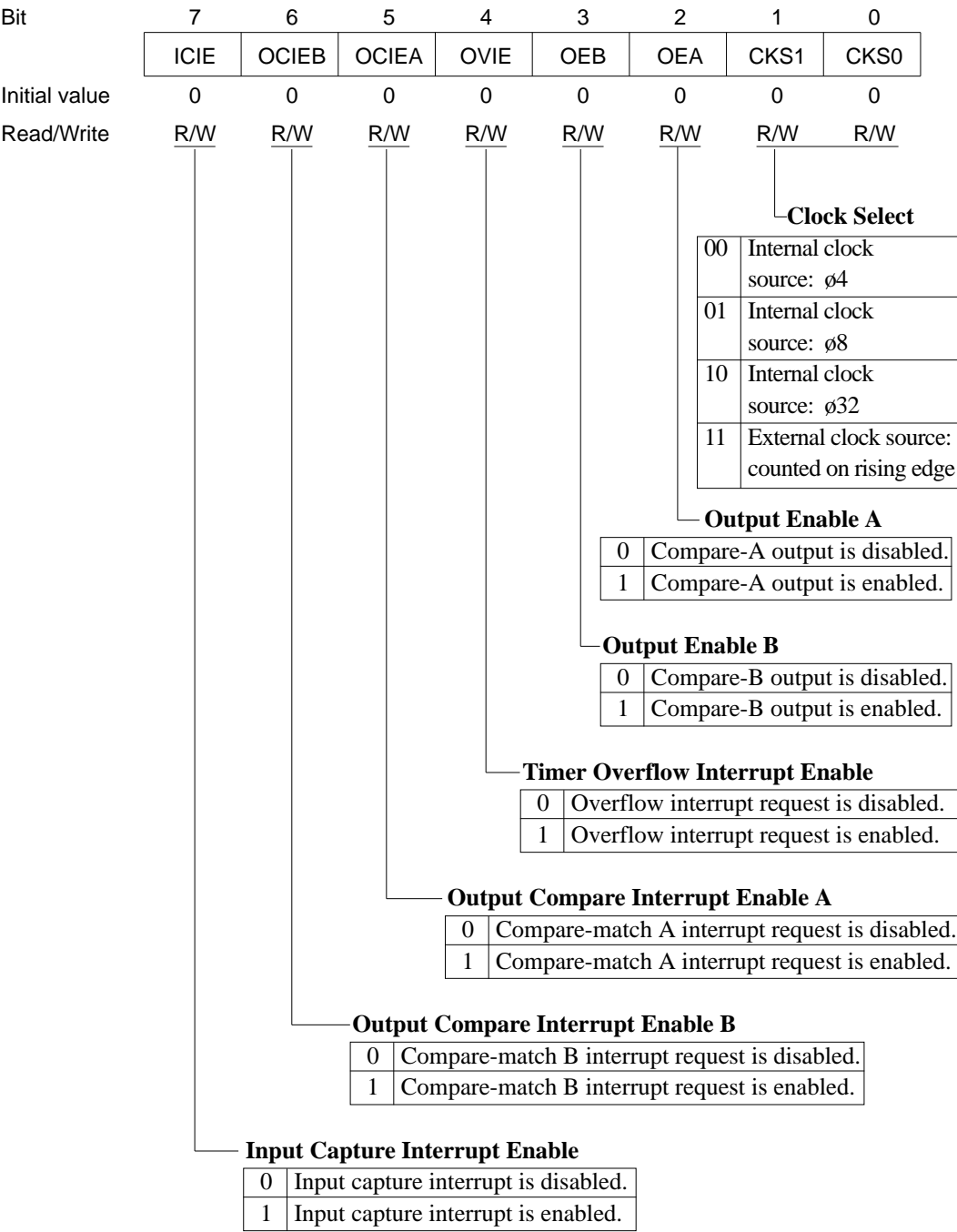
Trigger Enable

0	The A/D external trigger is disabled.
1	The A/D external trigger is enabled and P40 is set for input. A/D conversion starts on the falling edge of the ADTRG signal input at P40.

TCR—Timer Control Register

H'FEA0

FRT1



TCSR—Timer Control/Status Register

H'FEA1

FRT1

Bit	7	6	5	4	3	2	1	0
	ICF	OCFB	OCFA	OVF	OLVLB	OLVLA	IEDG	CCLRA
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/W	R/W	R/W	R/W

Counter Clear A	
0	FRC count is not cleared.
1	FRC count is cleared by compare-match A.

Input Edge Select	
0	Count is captured on falling edge of input capture signal (FTI).
1	Count is captured on rising edge of input capture signal.

Output Level A	
0	Compare-match A causes 0 output.
1	Compare-match A causes 1 output.

Output Level B	
0	Compare-match B causes 0 output.
1	Compare-match B causes 1 output.

Timer Overflow	
0	Cleared from 1 to 0 when CPU reads OVF = 1, then writes 0 in OVF.
1	Set to 1 when FRC changes from H'FFFF to H'0000.

Output Compare Flag A	
0	Cleared from 1 to 0 when: 1. CPU reads OCFA = 1, then writes 0 in OCFA. 2. OCIA interrupt is served by DTC.
1	Set to 1 when FRC = OCRA.

Output Compare Flag B	
0	Cleared from 1 to 0 when: 1. CPU reads OCFB = 1, then writes 0 in OCFB. 2. OCIB interrupt is served by DTC.
1	Set to 1 when FRC = OCRB.

Input Capture Flag	
0	Cleared from 1 to 0 when: 1. CPU reads ICF = 1, then writes 0 in ICF. 2. ICI interrupt is served by DTC.
1	Set to 1 when input capture signal is received and FRC count is copied to ICR.

* Only writing of a 0 to clear the flag is enabled.

FRC (H and L)—Free-Running Counter**H'FEA2, H'FEA3****FRT 1**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Count value

OCRA (H and L)—Output Compare Register A**H'FEA4, H'FEA5****FRT 1**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Continually compared with FRC. OCFA is set to 1 when OCRA = FRC.

OCRB (H and L)—Output Compare Register B**H'FEA6, H'FEA7****FRT 1**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Continually compared with FRC. OCFB is set to 1 when OCRB = FRC.

ICR (H and L)—Input Capture Register**H'FEA8, H'FEA9****FRT 1**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

Contains FRC count captured when external input capture signal changes.

TCR—Timer Control Register**H'FEB0****FRT 2**

Bit	7	6	5	4	3	2	1	0
	ICIE	OCIEB	OCIEA	OVIE	OEB	OEA	CKS1	CKS0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: Bit functions are the same as for FRT1.

TCSR—Timer Control/Status Register**H'FEB1****FRT 2**

Bit	7	6	5	4	3	2	1	0
	ICF	OCFB	OCFA	OVF	OLVLB	OLVLA	IEDG	CCLRA
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/W	R/W	R/W	R/W

Note: Bit functions are the same as for FRT1.

* Only writing of a 0 to clear the flag is enabled.

FRC (H and L)—Free-Running Counter**H'FEB2, H'FEB3****FRT 2**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: Bit functions are the same as for FRT1.

OCRA (H and L)—Output Compare Register A H'FEB4, H'FEB5 FRT 2

Bit	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: Bit functions are the same as for FRT1.

OCRB (H and L)—Output Compare Register B H'FEB6, H'FEB7 FRT 2

Bit	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: Bit functions are the same as for FRT1.

ICR (H and L)—Input Capture Register H'FEB8, H'FEB9 FRT 2

Bit	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

Note: Bit functions are the same as for FRT1.

TCR—Timer Control Register**H'FEC0****TMR**

Bit	7	6	5	4	3	2	1	0
	CMIEB	CMIEA	OVIE	CCLR1	CCLR0	CKS2	CKS1	CKS0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

			Clock Select		
0	0	0	No clock source; timer stops.		
0	0	1	Internal clock source: $\phi/8$, counted on falling edge.		
0	1	0	Internal clock source: $\phi/64$, counted on falling edge.		
0	1	1	Internal clock source: $\phi/1024$, counted on falling edge.		
1	0	0	No clock source; timer stops.		
1	0	1	External clock source, counted on rising edge.		
1	1	0	External clock source, counted on falling edge.		
1	1	1	External clock source, counted on both rising and falling edges.		

			Counter Clear		
0	0		Counter is not cleared.		
0	1		Cleared by compare-match A.		
1	0		Cleared by compare-match B.		
1	1		Cleared on rising edge of external reset input.		

			Timer Overflow Interrupt Enable		
0			Overflow interrupt request is disabled.		
1			Overflow interrupt request is enabled.		

			Compare-Match Interrupt Enable A		
0			Compare-match A interrupt request is disabled.		
1			Compare-match A interrupt request is enabled.		

			Compare-Match Interrupt Enable B		
0			Compare-match B interrupt request is disabled.		
1			Compare-match B interrupt request is enabled.		

TCSR—Timer Control/Status Register

H'FEC1

TMR

Bit	7	6	5	4	3	2	1	0
	CMFB	CMFA	OVF	—	OS3*2	OS2*2	OS1*2	OS0*2
Initial value	0	0	0	1	0	0	0	0
Read/Write	R/(W)*1	R/(W)*1	R/(W)*1	—	R/W	R/W	R/W	R/W

Output Select

0	0	No change on compare-match A.
0	1	Output 0 on compare-match A.
1	0	Output 1 on compare-match A.
1	1	Invert (toggle) output on compare-match A.

Output Select

0	0	No change on compare-match B.
0	1	Output 0 on compare-match B.
1	0	Output 1 on compare-match B.
1	1	Invert (toggle) output on compare-match B.

Timer Overflow Flag

0	Cleared from 1 to 0 when CPU reads OVF = 1, then writes 0 in OVF.
1	Set to 1 when TCNT changes from H'FF to H'00.

Compare-Match Flag A

0	Cleared from 1 to 0 when: 1. CPU reads CMFA = 1, then writes 0 in CMFA. 2. CMIA interrupt is served by the DTC.
1	Set to 1 when TCNT = TCORA.

Compare-Match Flag B

0	Cleared from 1 to 0 when: 1. CPU reads CMFB = 1, then writes 0 in CMFB. 2. CMIB interrupt is served by the DTC.
1	Set to 1 when TCNT = TCORB.

Notes: *1 Only writing of 0 to clear the flag is enabled.

*2 When all four bits (OS3 to OS0) are cleared to 0, output is disabled.

TCORA—Time Constant Register A

H'FEC2

TMR

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The CMFA bit is set to 1 when TCORA = TCNT.

TCORB—Time Constant Register B

H'FEC3

TMR

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The CMFB bit is set to 1 when TCORB = TCNT.

TCNT—Timer Counter

H'FEC4

TMR

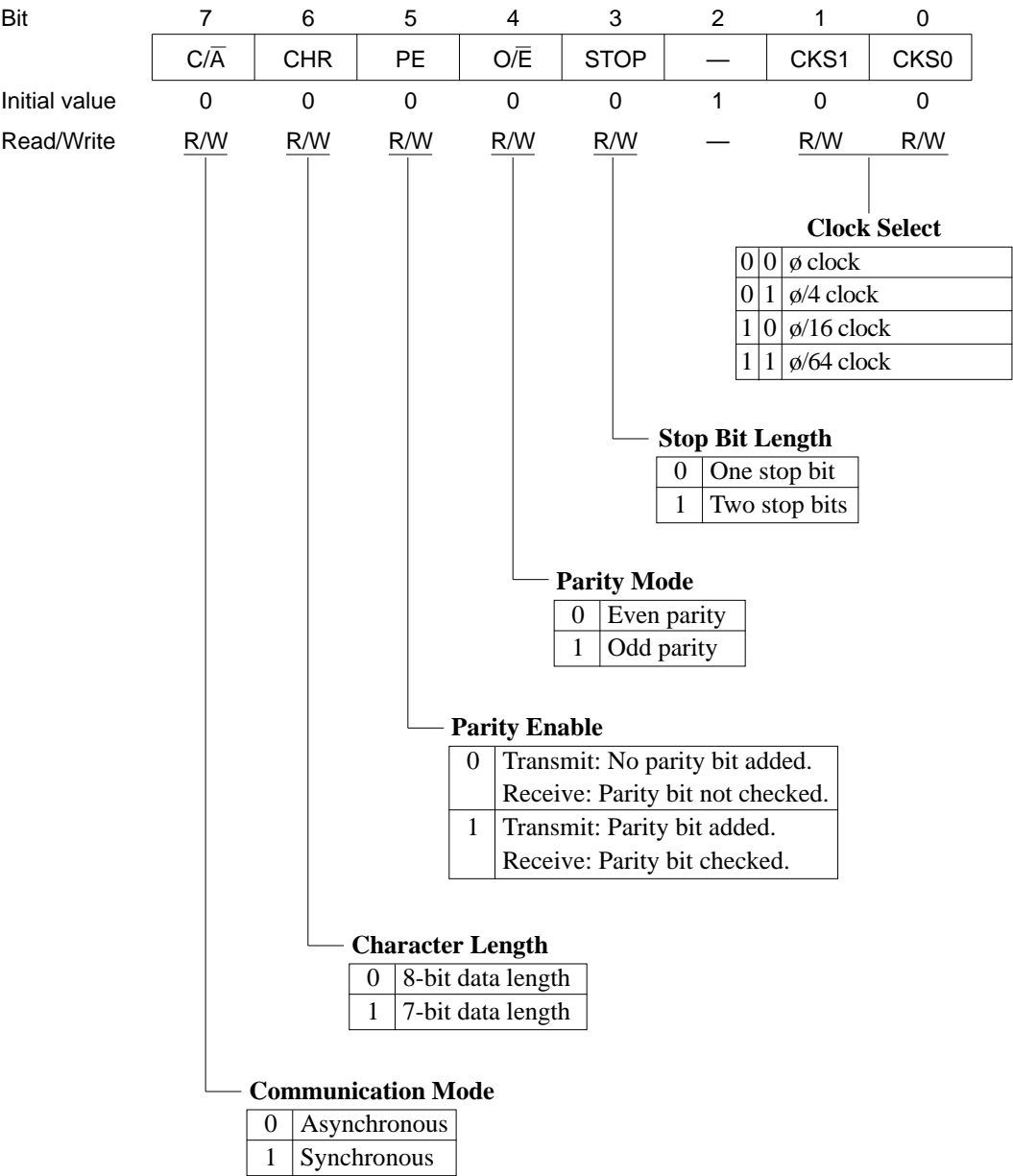
Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Count value

SMR—Serial Mode Register

H'FEC8

SCI1



BRR—Bit Rate Register**H'FEC9****SCI1**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Constant that determines the baud rate

SCR—Serial Communication Register**H'FECA****SCI1**

Bit	7	6	5	4	3	2	1	0
	TIE	RIE	TE	RE	—	—	CKE1	CKE0
Initial value	0	0	0	0	1	1	0	0
Read/Write	R/W	R/W	R/W	R/W	—	—	R/W	R/W

Clock Enable 0

0	SCK pin is not used.
1	SCK pin is used for output.

Clock Enable 1

0	Internal clock
1	External clock, input at SCK pin

Receive Enable

0	Receive disabled
1	Receive enabled

Transmit Enable

0	Transmit disabled
1	Transmit enabled

Receive Interrupt Enable

0	Receive interrupt request (RXI) is disabled.
1	Receive interrupt request (RXI) is enabled.

Transmit Interrupt Enable

0	Transmit interrupt request (TXI) is disabled.
1	Transmit interrupt request (TXI) is enabled.

TDR—Transmit Data Register

H'FECB

SCI1

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Transmit data

SSR—Serial Status Register

H'FECC

SCI1

Bit	7	6	5	4	3	2	1	0
	TDRE	RDRF	ORER	FER	PER	—	—	—
Initial value	1	0	0	0	0	1	1	1
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	—	—	—

Parity Error

0	Cleared from 1 to 0 when: 1. CPU reads PER = 1, then writes 0 in PER. 2. The chip is reset or enters a standby mode.
1	Set to 1 when a parity error occurs (parity of receive data does not match parity selected by O/E bit).

Framing Error

0	Cleared from 1 to 0 when: 1. CPU reads FER = 1, then writes 0 in FER. 2. The chip is reset or enters a standby mode.
1	Set to 1 when a framing error occurs (stop bit is 0).

Overrun Error

0	Cleared from 1 to 0 when: 1. CPU reads ORER = 1, then writes 0 in ORER. 2. The chip is reset or enters a standby mode.
1	Set to 1 when an overrun error occurs (next data is completely received while RDRF bit is set to 1).

Receive Data Register Full

0	Cleared from 1 to 0 when: 1. CPU reads RDRF = 1, then writes 0 in RDRF. 2. RDR is read by the DTC. 3. The chip is reset or enters a standby mode.
1	Set to 1 when one character is received normally and transferred from RSR to RDR.

Transmit Data Register Empty

0	Cleared from 1 to 0 when: 1. CPU reads TDRE = 1, then writes 0 in TDRE. 2. The DTC writes data in TDR.
1	Set to 1 when: 1. The chip is reset or enters a standby mode. 2. Data is transferred from TDR to TSR. 3. TE is cleared to 0 when TDRE = 0.

* Only writing of 0 to clear the flag is enabled.

RDR—Receive Data Register**H'FECD****SCI1**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

Receive data

SMR—Serial Mode Register**H'FED0****SCI2**

Bit	7	6	5	4	3	2	1	0
	C/ \bar{A}	CHR	PE	O/ \bar{E}	STOP	—	CKS1	CKS0
Initial value	0	0	0	0	0	1	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	—	R/W	R/W

Note: Bit functions are the same as for SCI1.**BRR—Bit Rate Register****H'FED1****SCI2**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: Bit functions are the same as for SCI1.**SCR—Serial Control Register****H'FED2****SCI2**

Bit	7	6	5	4	3	2	1	0
	TIE	RIE	TE	RE	—	—	CKE1	CKE0
Initial value	0	0	0	0	1	1	0	0
Read/Write	R/W	R/W	R/W	R/W	—	—	R/W	R/W

Note: Bit functions are the same as for SCI1.

TDR—Transmit Data Register**H'FED3****SCI2**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: Bit functions are the same as for SCI1.

SSR—Serial Status Register**H'FED4****SCI2**

Bit	7	6	5	4	3	2	1	0
	TDRE	RDRF	ORER	FER	PER	—	—	—
Initial value	1	0	0	0	0	1	1	1
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	—	—	—

Note: Bit functions are the same as for SCI1.

* Only writing of 0 to clear the flag is enabled.

RDR—Receive Data Register**H'FED5****SCI2**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R

Note: Bit functions are the same as for SCI1.

RFSHC377

IPRA—Interrupt Priority Register A**H'FF00****INTC**

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		IRQ0				IRQ1 to IRQ3		
		Interrupt priority level (0 to 7)				Interrupt priority level (0 to 7)		

IPRB—Interrupt Priority Register B**H'FF01****INTC**

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		FRT1				FRT2		
		Interrupt priority level (0 to 7)				Interrupt priority level (0 to 7)		

IPRC—Interrupt Priority Register C

H'FF02

INTC

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		8-bit timer				SCI1		
		Interrupt priority level (0 to 7)				Interrupt priority level (0 to 7)		

IPRD—Interrupt Priority Register D

H'FF03

INTC

Bit	7	6	5	4	3	2	1	0
	—				—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W
		SCI2				A/D converter		
		Interrupt priority level (0 to 7)				Interrupt priority level (0 to 7)		

DTEA—Data Transfer Enable Register A

H'FF08

INTC

Bit	7	6	5	4	3	2	1	0
	—	—	—		—			
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IRQ0

0	Served by CPU
1	Served by DTC

IRQ3

0	Served by CPU
1	Served by DTC

IRQ2

0	Served by CPU
1	Served by DTC

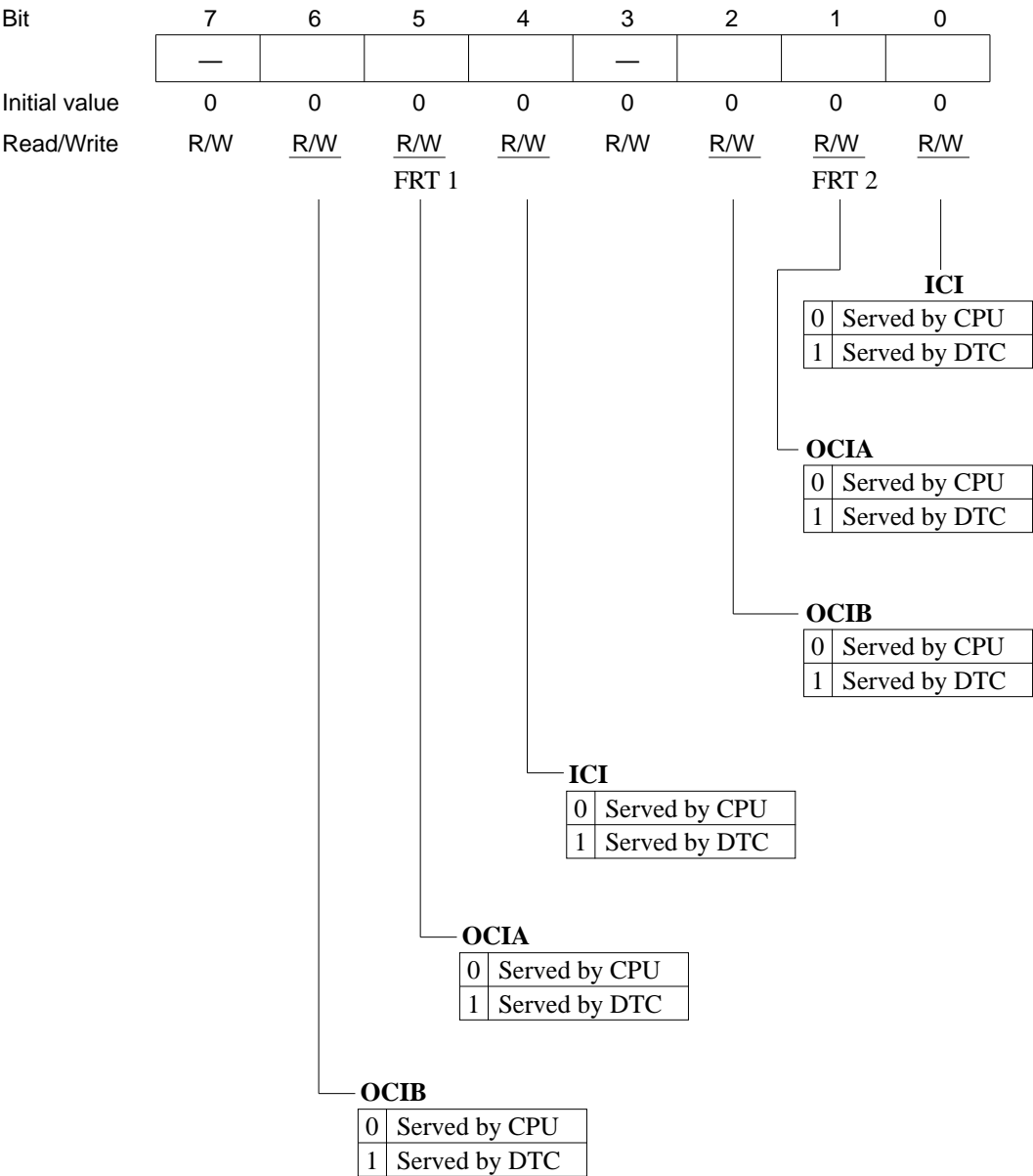
IRQ1

0	Served by CPU
1	Served by DTC

DTEB—Data Transfer Enable Register B

H'FF09

INTC



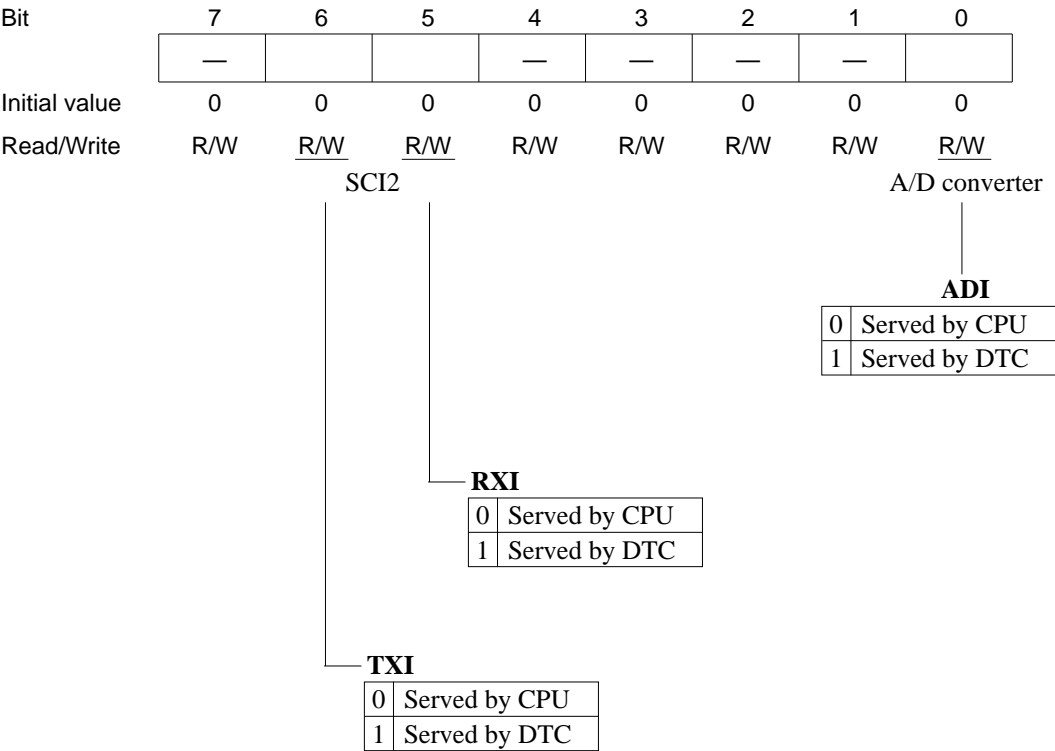
INTC

CMIA	
0	Served by CPU
1	Served by DTC

DTED—Data Transfer Enable Register D

H'FF0B

INTC



TCSR—Timer Control/Status Register**H'FF10*1, H'FF11*2****WDT**

Bit	7	6	5	4	3	2	1	0
	OVF	WT/IT	TME	—	—	CKS2	CKS1	CKS0
Initial value	0	0	0	1	1	0	0	0
Read/Write	R/(W)*3	R/W	R/W	—	—	R/W	R/W	R/W

Clock Select

0	0	0	$\phi/2$	(51.2 μs)*4
0	0	1	$\phi/32$	(819.2 μs)
0	1	0	$\phi/64$	(1.6 ms)
0	1	1	$\phi/128$	(3.3 ms)
1	0	0	$\phi/256$	(6.6 ms)
1	0	1	$\phi/512$	(13.1 ms)
1	1	0	$\phi/2048$	(52.4 ms)
1	1	1	$\phi/4096$	(104.9 ms)

Timer Enable

0	Timer is disabled. • TCNT is initialized to H'00 and stopped.
1	Timer is enabled. • TCNT starts incrementing. • CPU interrupt request is enabled.

Timer Mode Select

0	Interval timer mode (IRQ0 interrupt request)
1	Watchdog timer mode (Reset)

Overflow Flag

0	Cleared from 1 to 0 when CPU reads OVF = 1, then writes 0 in OVF.
1	Set to 1 when TCNT changes from H'FF to H'00.

Notes: *1 Read address

*2 Write address

*3 Only writing of 0 to clear the flag is enabled.

*4 Times in parentheses are the times for TCNT to increment from H'00 to H'FF and change to H'00 again when $\phi = 10 \text{ MHz}$.

TCNT—Timer Counter**H'FF11****WDT**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Count value

WCR—Wait-State Control Register**H'FF14****WSC**

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	WMS1	WMS0	WC1	WC0
Initial value	1	1	1	1	0	0	1	1
Read/Write	—	—	—	—	R/W	R/W	R/W	R/W

Wait Count 1 and 0

0	0	No wait states (Tw) are inserted.
0	1	1 wait state is inserted.
1	0	2 wait states are inserted.
1	1	3 wait states are inserted.

Wait Mode Select 1 and 0

0	0	Programmable wait mode
0	1	No wait states are inserted, regardless of the wait count.
1	0	Pin wait mode
1	1	Pin auto-wait mode

ARBT—Byte Area Top Register**H'FF16****BSC**

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

AR3T—3-State Area Top Register**H'FF17****BSC**

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MDCR—Mode Control Register**H'FF19**

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	MDS2	MDS1	MDS0
Initial value	1	1	0	0	0	—*	—*	—*
Read/Write	—	—	—	—	—	R	R	R

Mode Select

Value input at mode pins

* Initialized according to the inputs at pins MD2, MD1, and MD0.

H'FF1A

Bit	7	6	5	4	3	2	1	0
	SSBY	—	—	—	—	—	—	—
Initial value	0	1	1	1	1	1	1	1
Read/Write	R/W	—	—	—	—	—	—	—


Software Standby

0	SLEEP instruction causes transition to sleep mode.
1	SLEEP instruction causes transition to software standby mode.

H'FF1B

Port 3

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	BRLE
Initial value	1	1	1	1	1	1	1	0
Read/Write	—	—	—	—	—	—	—	R/W




Bus Release Enable	
0	P32 and P31 are input and output pins.
1	P32 is the $\overline{\text{BREQ}}$ input pin and P31 is the $\overline{\text{BACK}}$ output pin.

H'FF1C

INTC

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	NMIEG
Initial value	1	1	1	1	1	1	1	0
Read/Write	R	R	R	R	R	R	R	R/W



Nonmaskable Interrupt Edge	
0	Interrupt requested on falling edge of NMI signal.
1	Interrupt requested on rising edge of NMI signal.

IRQCR—IRQ Control Register**H'FF1D****INTC**

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	IRQ3E	IRQ2E	IRQ1E	IRQ0E
Initial value	1	1	1	1	0	0	0	0
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W

Interrupt Request 3 Enable

0	P83 is an input/output pin.
1	P83 is used for $\overline{\text{IRQ3}}$ signal input, regardless of P83DDR. (The pin level can also be read.)

Interrupt Request 2 Enable

0	P82 is an input/output pin.
1	P82 is used for $\overline{\text{IRQ2}}$ signal input, regardless of P82DDR. (The pin level can also be read.)

Interrupt Request 1 Enable

0	P81 is an input/output pin.
1	P81 is used for $\overline{\text{IRQ1}}$ signal input, regardless of P81DDR. (The pin level can also be read.)

Interrupt Request 0 Enable

0	P80 is an input/output pin.
1	P80 is used for $\overline{\text{IRQ0}}$ signal input, regardless of P80DDR. (The pin level can also be read.)

RSTCSR—Reset Status/Control Register **H'FF1F** **WDT**

Bit	7	6	5	4	3	2	1	0
	WRST	RSTOE	—	—	—	—	—	—
Initial value	0	0	1	1	1	1	1	1
Read/Write	R/(W)*1	R/W	—	—	—	—	—	—

Reset Output Enable

0	The reset signal generated when the watchdog timer overflows is not output externally.
1	The reset signal generated when the watchdog timer overflows is output externally.

Watchdog Timer Reset

0	Cleared from 1 to 0 by software, or by a Low input at the $\overline{\text{RES}}$ pin.
1	Set to 1 when TCNT overflows in watchdog timer mode and a reset signal is generated.

Note: *1 Software can write a 0 in bit 7 to clear the flag but cannot write a 1.

Appendix C I/O Port Schematic Diagrams

C.1 Schematic Diagram of Port 1

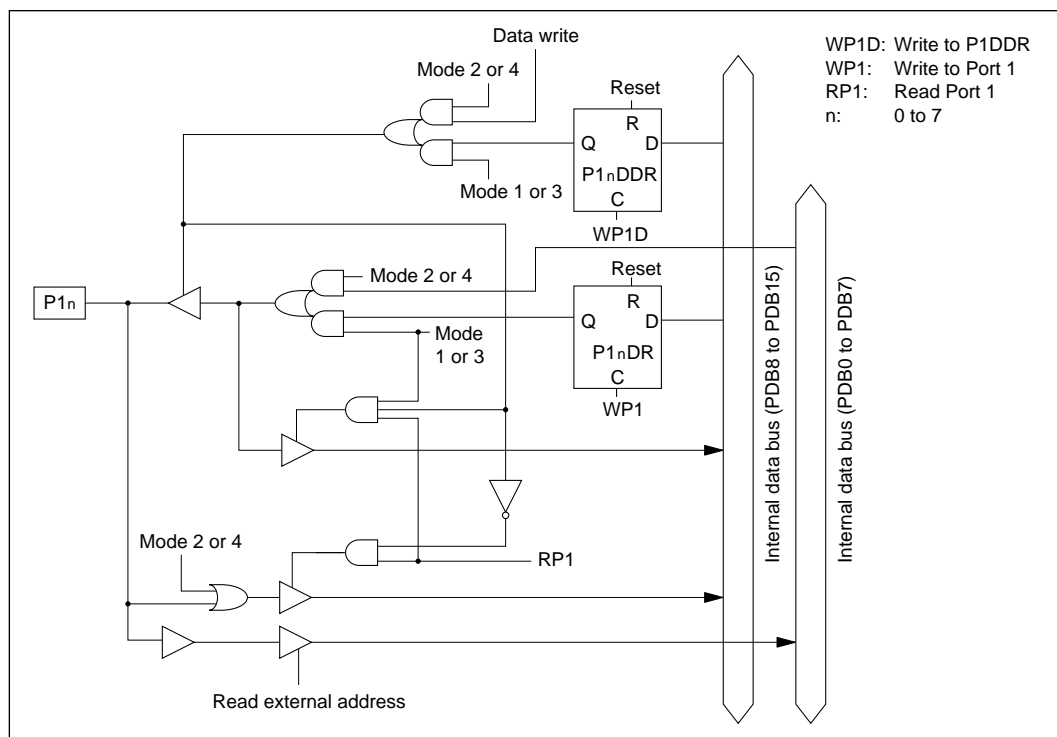


Figure C-1 Schematic Diagram of Port 1

Table C-1 Data Read from Port 1

Mode	Data
2 or 4	Always 1
1 or 3	DDR = 0 Logic level at pin
	DDR = 1 DR value

C.2 Schematic Diagram of Port 2

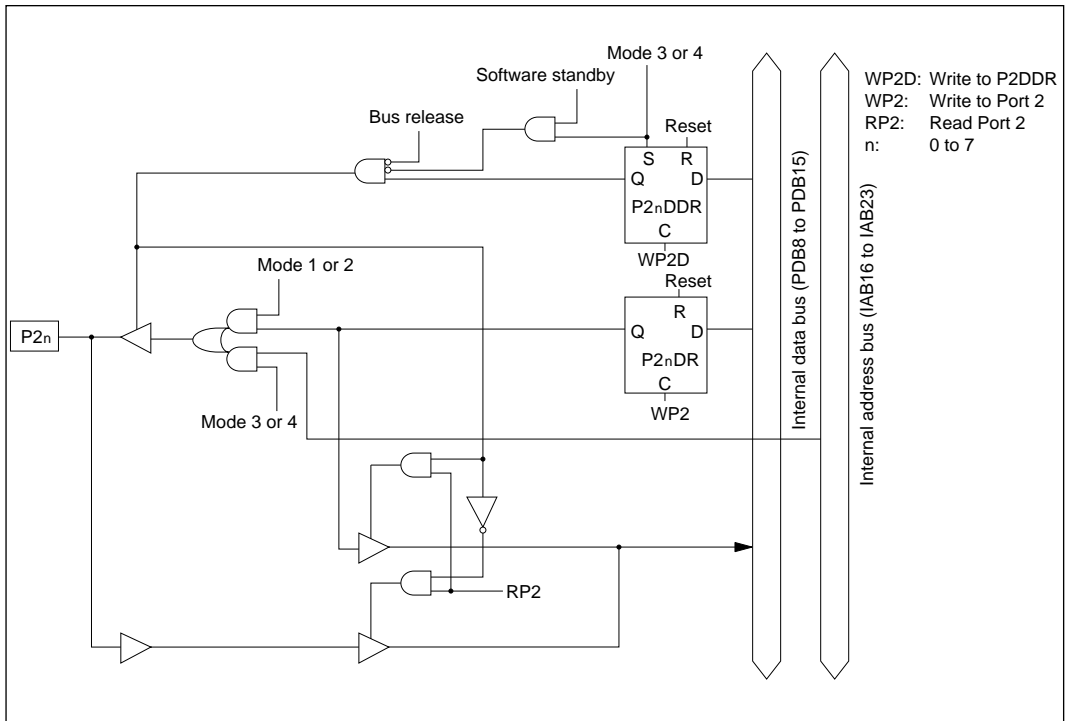


Figure C-2 Schematic Diagram of Port 2

Table C-2 Data Read from Port 2

Mode		Data
3 or 4		DR value
1 or 2	DDR = 0	Logic level at pin
	DDR = 1	DR value

C.3 Schematic Diagrams of Port 3

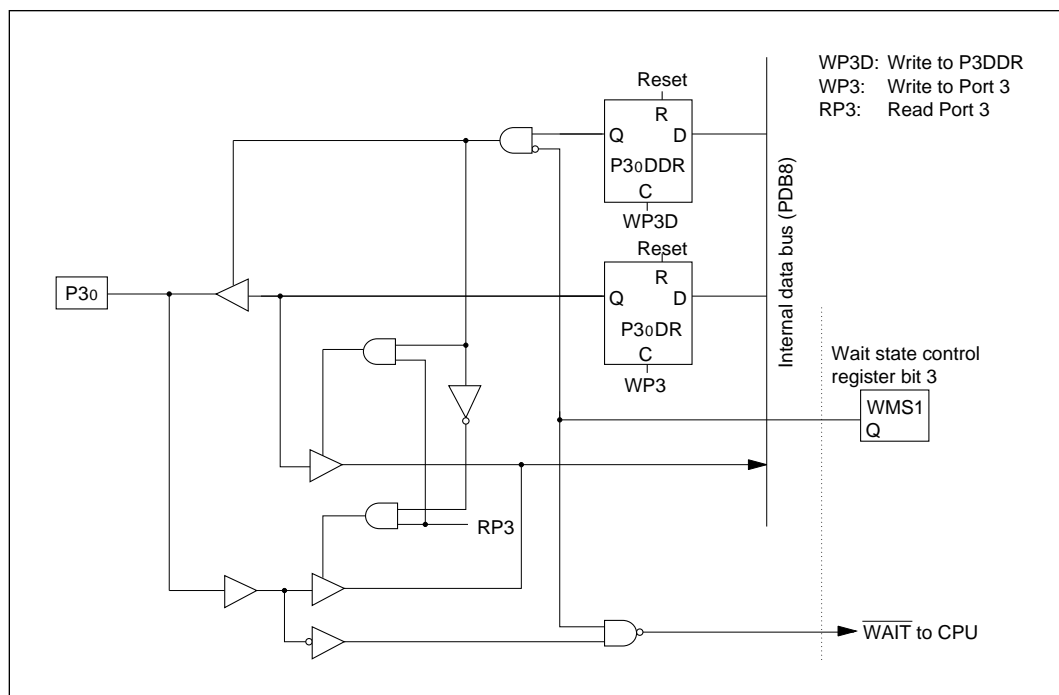


Figure C-3 (a) Schematic Diagram of Port 3, Pin P30

Table C-3 (a) Data Read from Port 3, Pin P30

Mode		Data
WMS1 = 1		Logic level at pin
WMS1 = 0	DDR = 0	Logic level at pin
	DDR = 1	DR value

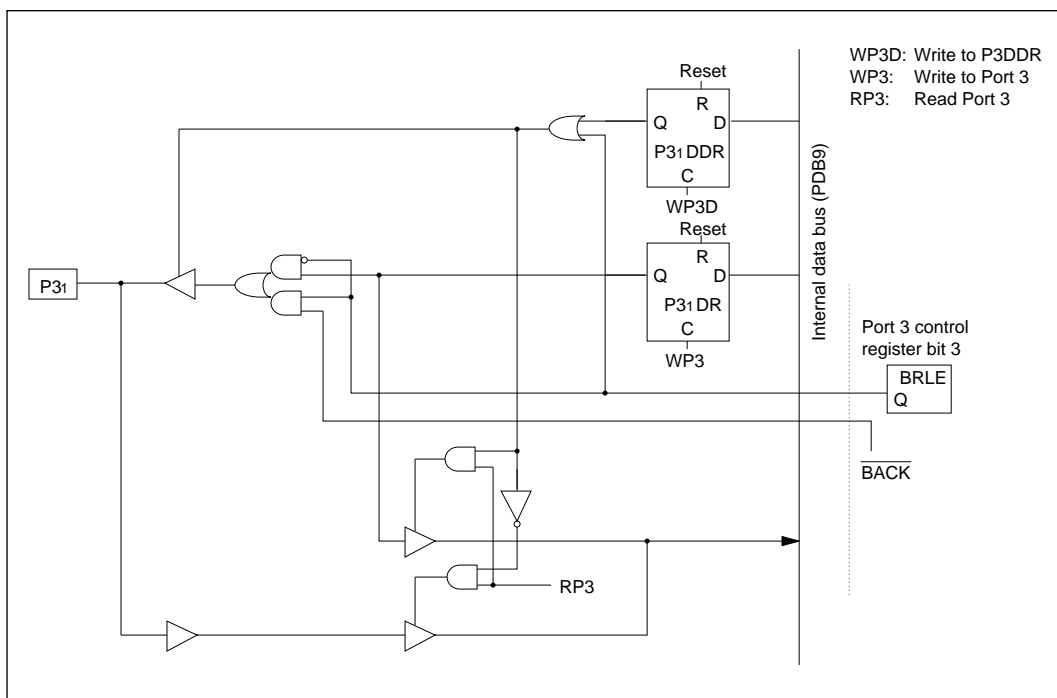


Figure C-3 (b) Schematic Diagram of Port 3, Pin P31

Table C-3 (b) Data Read from Port 3, Pin P31

Mode		Data
BRLE = 1		DR value
BRLE = 0	DDR = 0	Logic level at pin
	DDR = 1	DR value

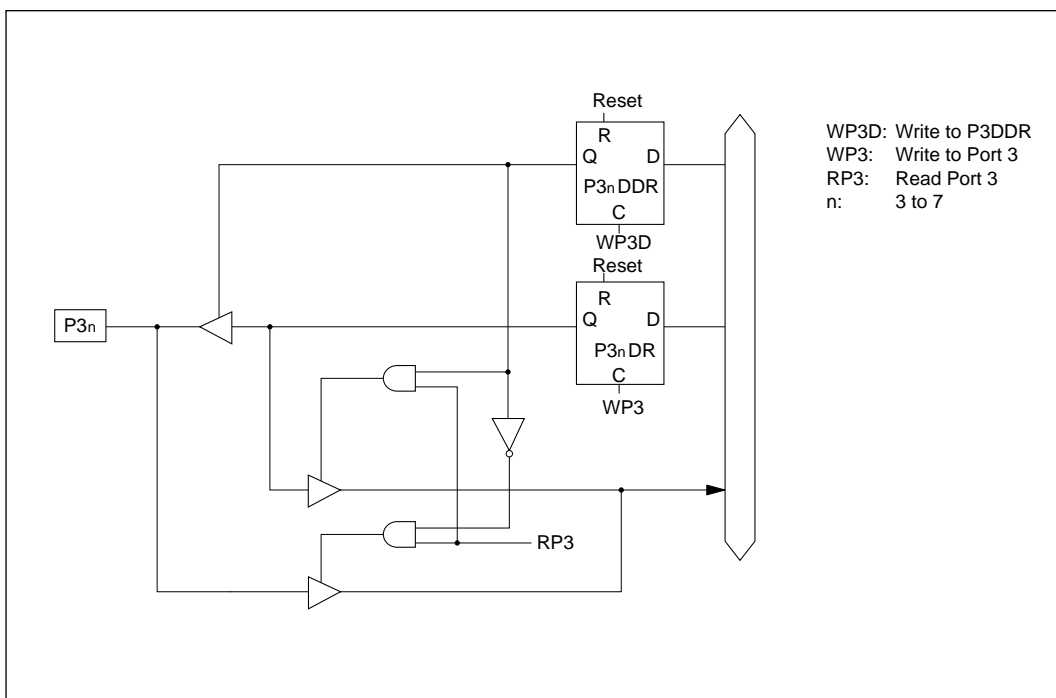


Figure C-3 (d) Schematic Diagram of Port 3, Pins P33 to P37

Table C-3 (d) Data Read from Port 3, Pins P33 to P37

Mode	Data
DDR = 0	Logic level at pin
DDR = 1	DR value

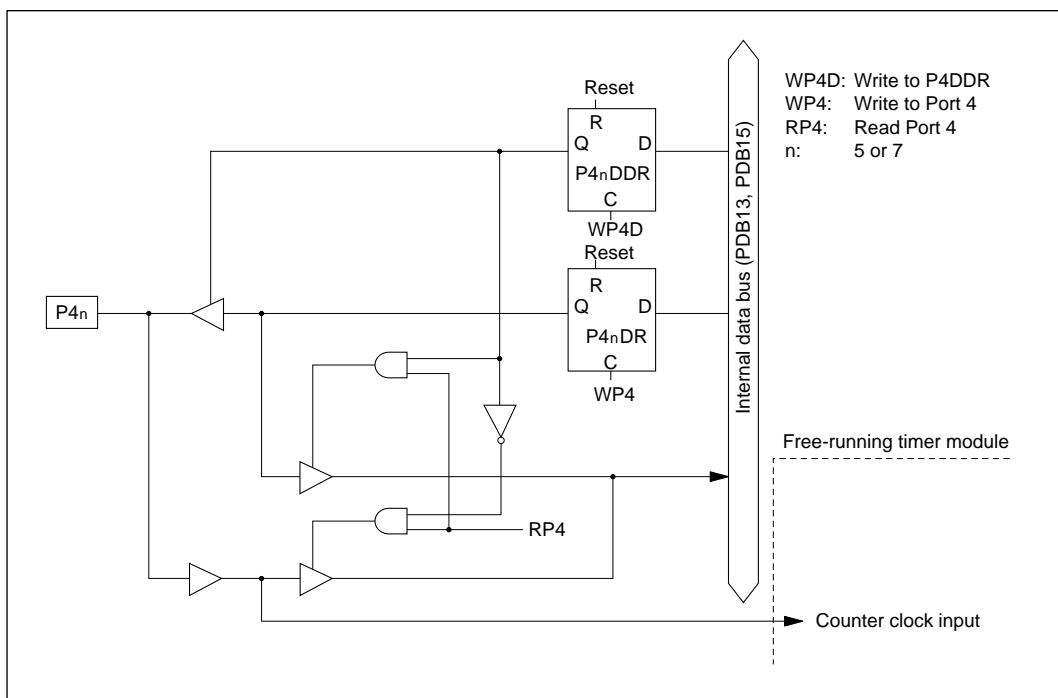


Figure C-4 (f) Schematic Diagram of Port 4, Pins P45 and P47

Table C-4 (f) Data Read from Port 4, Pins P45 and P47

Mode	Data
DDR = 0	Logic level at pin
DDR = 1	DR value

C.5 Schematic Diagrams of Port 5

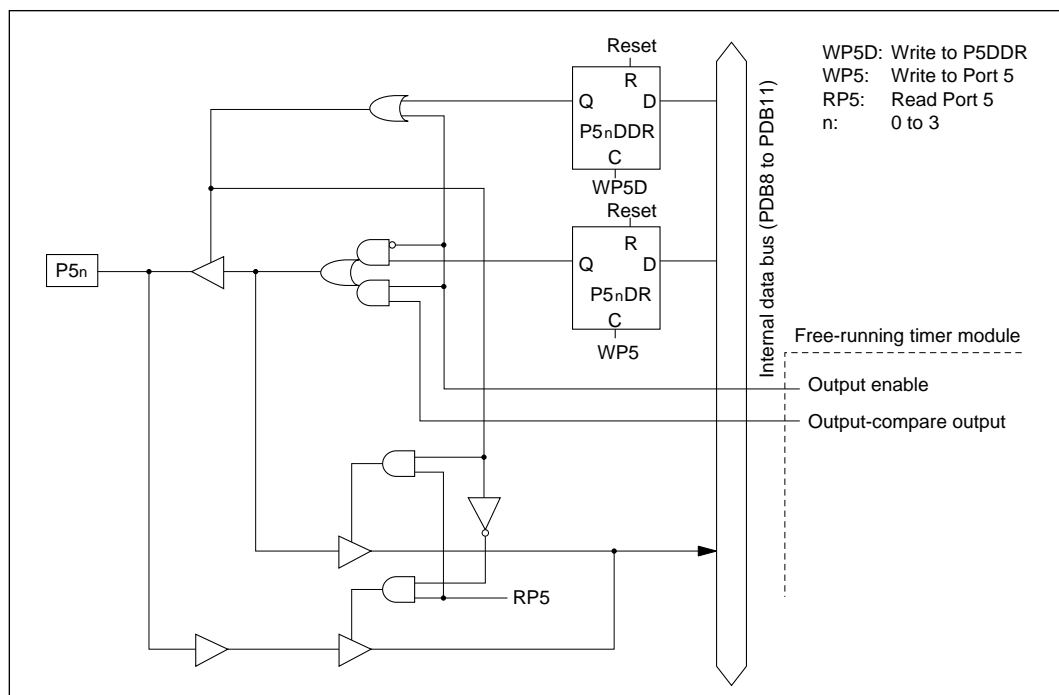


Figure C-5 (a) Schematic Diagram of Port 5, Pins P50 to P53

Table C-5 (a) Data Read from Port 5, Pins P50 to P53

Mode		Data
Output enabled		Output-compare output
Output disabled	DDR = 0	Logic level at pin
	DDR = 1	DR value

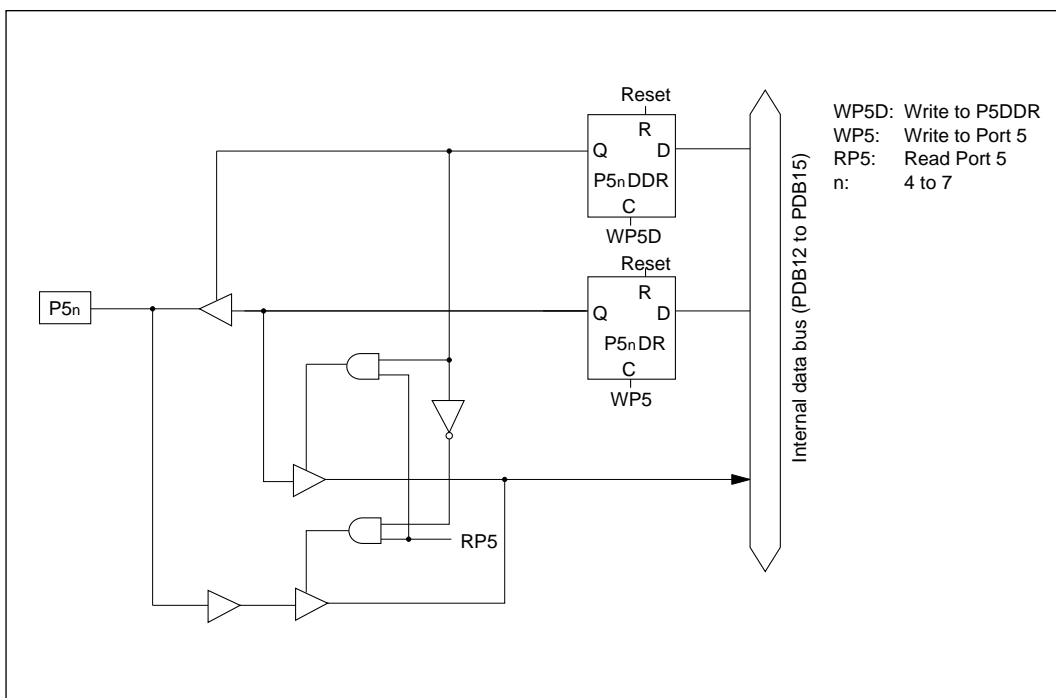


Figure C-5 (b) Schematic Diagram of Port 5, Pins P54 to P57

Table C-5 (b) Data Read from Port 5, Pins P54 to P57

Mode	Data
DDR = 0	Logic level at pin
DDR = 1	DR value

C.6 Schematic Diagram of Port 6

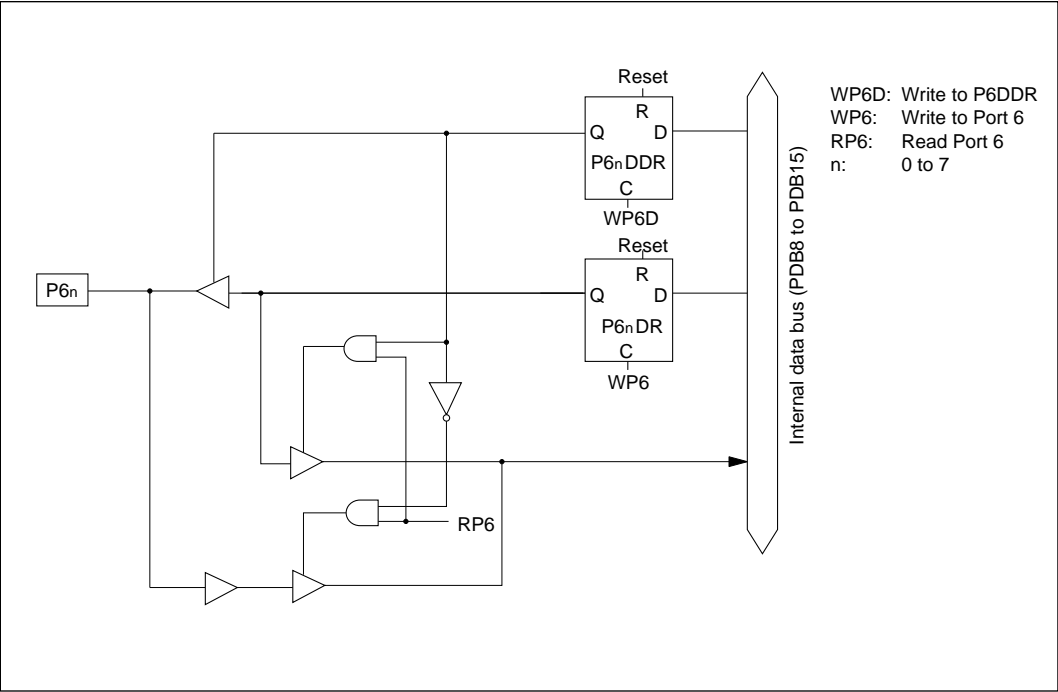


Figure C-6 Schematic Diagram of Port 6, Pins P60 to P67

Table C-6 Data Read from Port 6, Pins P60 and P67

Mode	Data
DDR = 0	Logic level at pin
DDR = 1	DR value

C.7 Schematic Diagram of Port 7

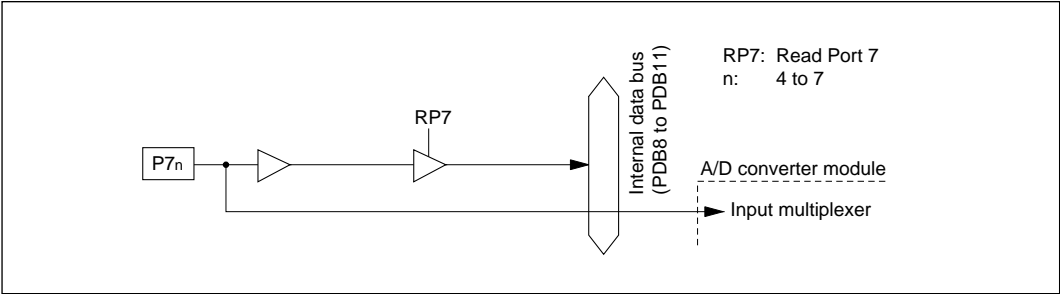


Figure C-7 Schematic Diagram of Port 7

C.8 Schematic Diagrams of Port 8

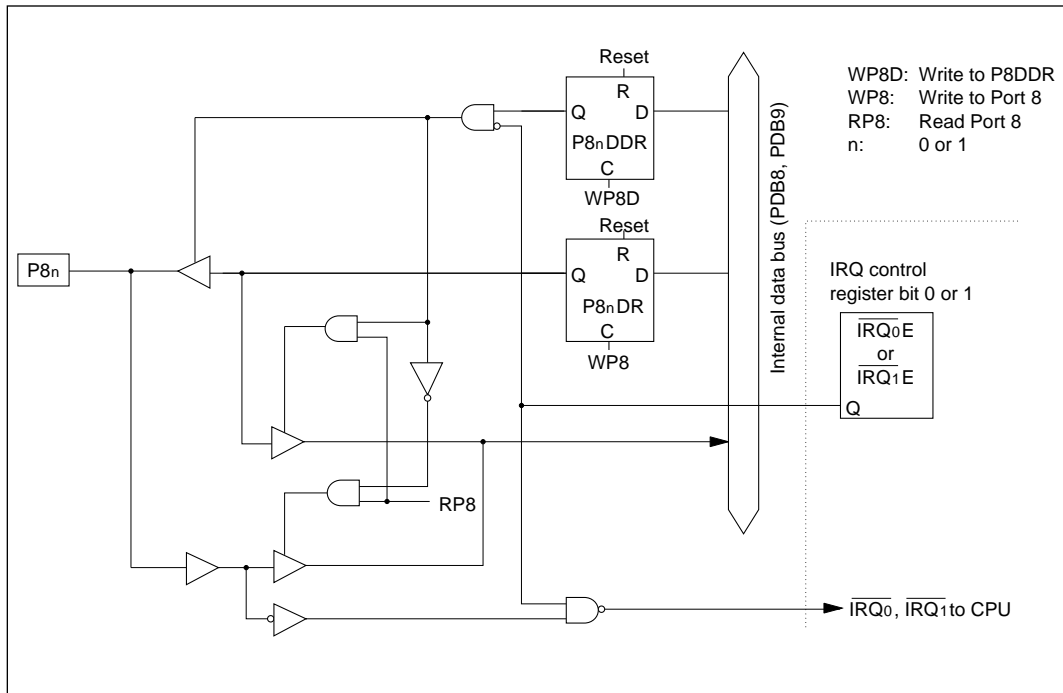


Figure C-8 (a) Schematic Diagram of Port 8, Pins P80 and P81

Table C-7 (a) Data Read from Port 8, Pins P80 and P81

Mode	Data
IRQ0E or IRQ1E = 1	Logic level at pin
IRQ0E or IRQ1E = 1	DDR = 0 Logic level at pin
	DDR = 1 DR value

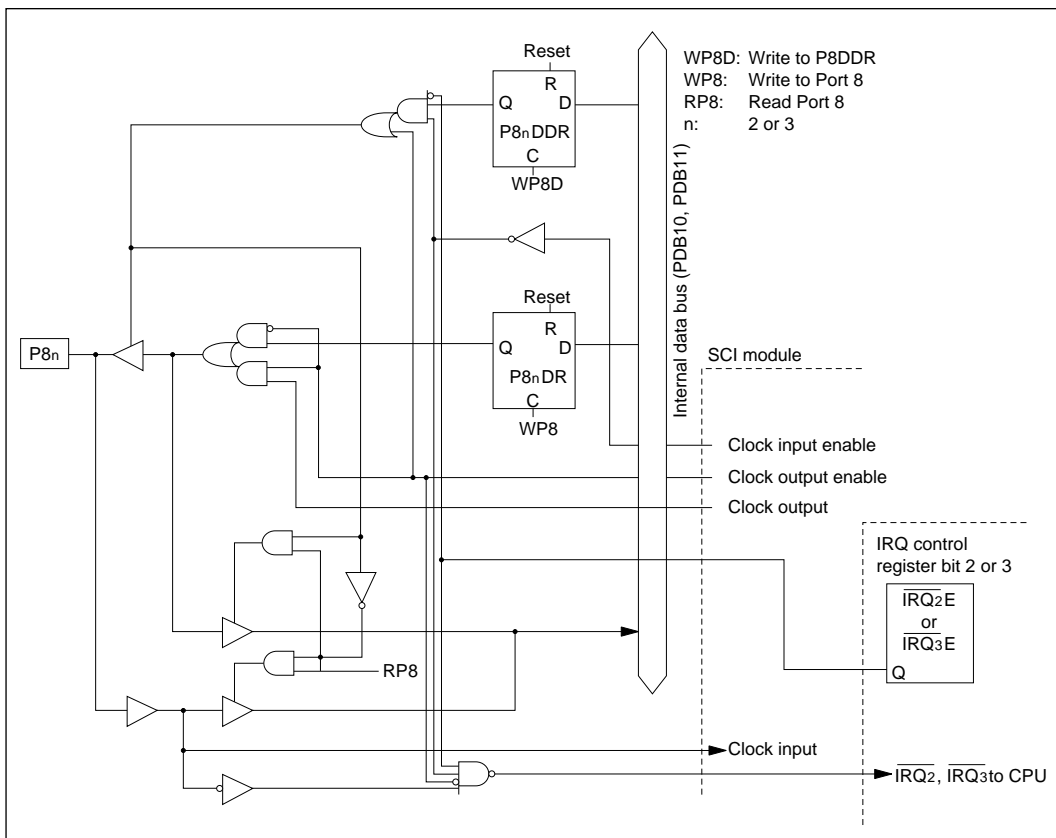
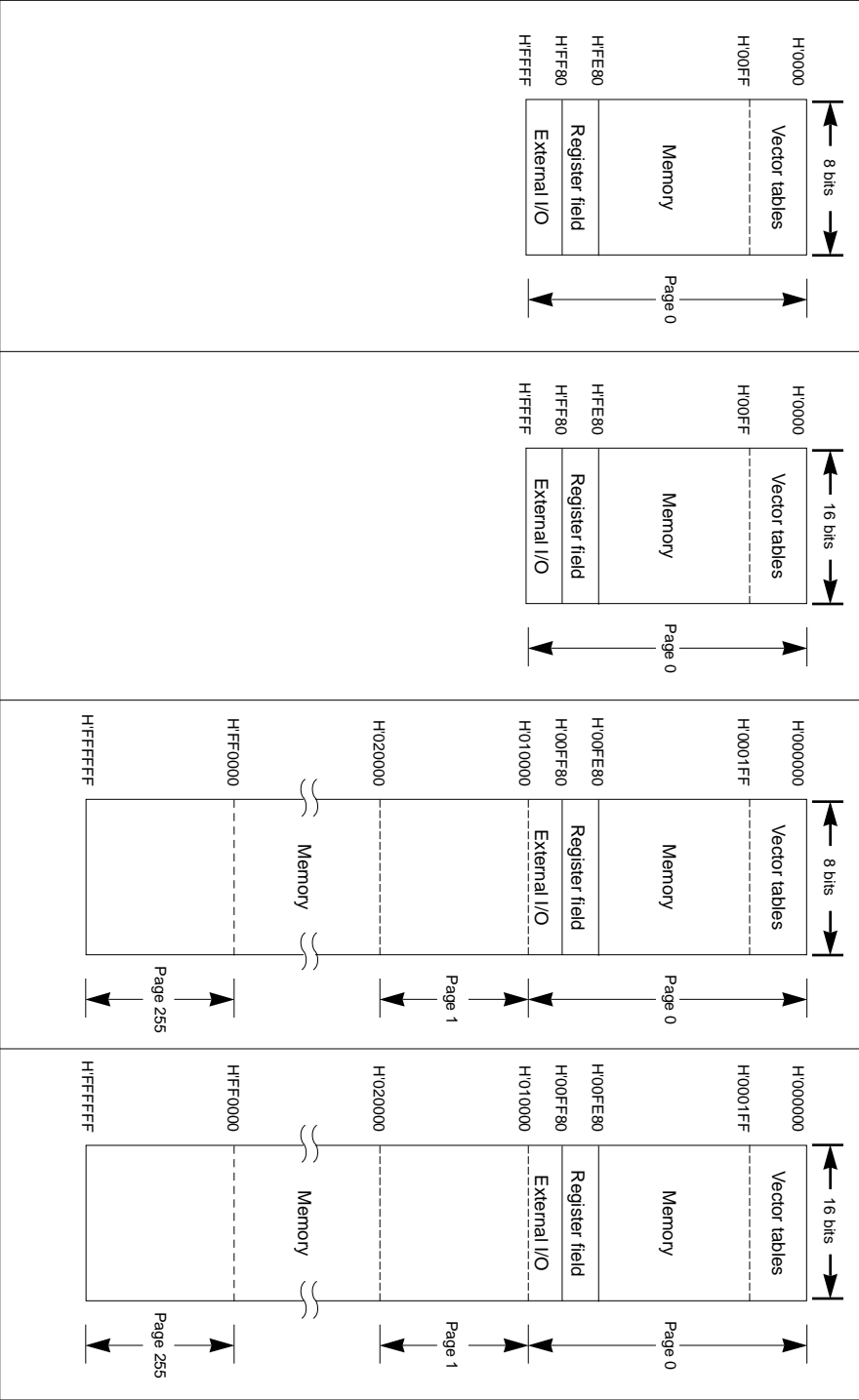


Figure C-8 (b) Schematic Diagram of Port 8, Pins P82 and P83

Table C-7 (b) Data Read from Port 8, Pins P82 and P83

Mode		Data
Clock input enabled		Clock input value
Clock output enabled		Clock output value
Clock input and output disabled	IRQ2E or IRQ3E = 1	Logic level at pin
	IRQ2E or IRQ3E = 0	DDR = 0 Logic level at pin
		DDR = 1 DR value



Appendix E Pin States

E.1 States of I/O Ports

Table E-1 States of I/O Ports

Pin Name	Mode	Reset	Hardware Standby Mode	Software Standby Mode	Sleep Mode	Bus Release Mode	Program Execution Mode (Normal Operation)
\emptyset , E	—	Clock output	3-state	\emptyset =High E=Low	Clock output	Clock output	Clock output
\overline{RD} , \overline{AS} , \overline{HWR} , \overline{LWR}	—	High	3-state	3-state	High*5	3-state	\overline{RD} , \overline{AS} , \overline{HWR} , \overline{LWR}
\overline{RFSH}	—	High	3-state	3-state	High*5	High	\overline{RFSH}
D15 to D8	—	3-state	3-state	3-state	3-state	3-state	D15 to D8
A15 to A0	—	Low	3-state	3-state	Low	3-state	A15 to A0
P17 to P10	1, 3	3-state	3-state	Prev. state	Prev. state	Prev. state	I/O port
	2, 4			3-state	3-state	3-state	D7 to D0
P27 to P20	1, 2	3-state	3-state	Prev. state	Prev. state	Prev. state	I/O port
	3, 4	Low		3-state	3-state	3-state	A23 to A16
P37 to P30	—	3-state	3-state	Prev. state *1	Prev. state *2	Prev. state *3	I/O port or control input/output
P47 to P40 P57 to P50 P67 to P60	—	3-state	3-state	Prev. state *4	Prev. state	Prev. state	I/O port
P73 to P70	—	3-state	3-state	3-state	3-state	3-state	Input port
P87 to P80	—	3-state	3-state	Prev. state *4	Prev. state	Prev. state	I/O port

Notes: 3-state: High-impedance state

Prev. state: Input pins are in the high-impedance state; output pins maintain their previous state.

*1 If P32 is set for \overline{BACK} output, it goes to the high-impedance state.

*2 \overline{BREQ} can be received, and \overline{BACK} is high.

*3 \overline{BACK} is low.

*4 The on-chip supporting modules are reset, so these pins become input or output pins according to their DDR and DR bits.

*5 During refresh cycles, \overline{RFSH} and \overline{AS} (and \overline{RD}) are low.

E.2 Pin Status in the Reset State

1. Mode 1

Figure E-1 shows how the pin states change when the $\overline{\text{RES}}$ pin goes low during access to a three-state-access area in mode 1.

As soon as $\overline{\text{RES}}$ goes low, all ports are initialized to the input (high-impedance) state. The $\overline{\text{AS}}$, $\overline{\text{RD}}$, $\overline{\text{HWR}}$, and $\overline{\text{LWR}}$ signals all go high. The data bus (D15 to D8) is placed in the high-impedance state.

The address bus is initialized 1.5 system clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled. All address bus signals are made low.

The clock output pins ϕ and E are initialized 0.5 system clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled. Both pins are initialized to the output state.

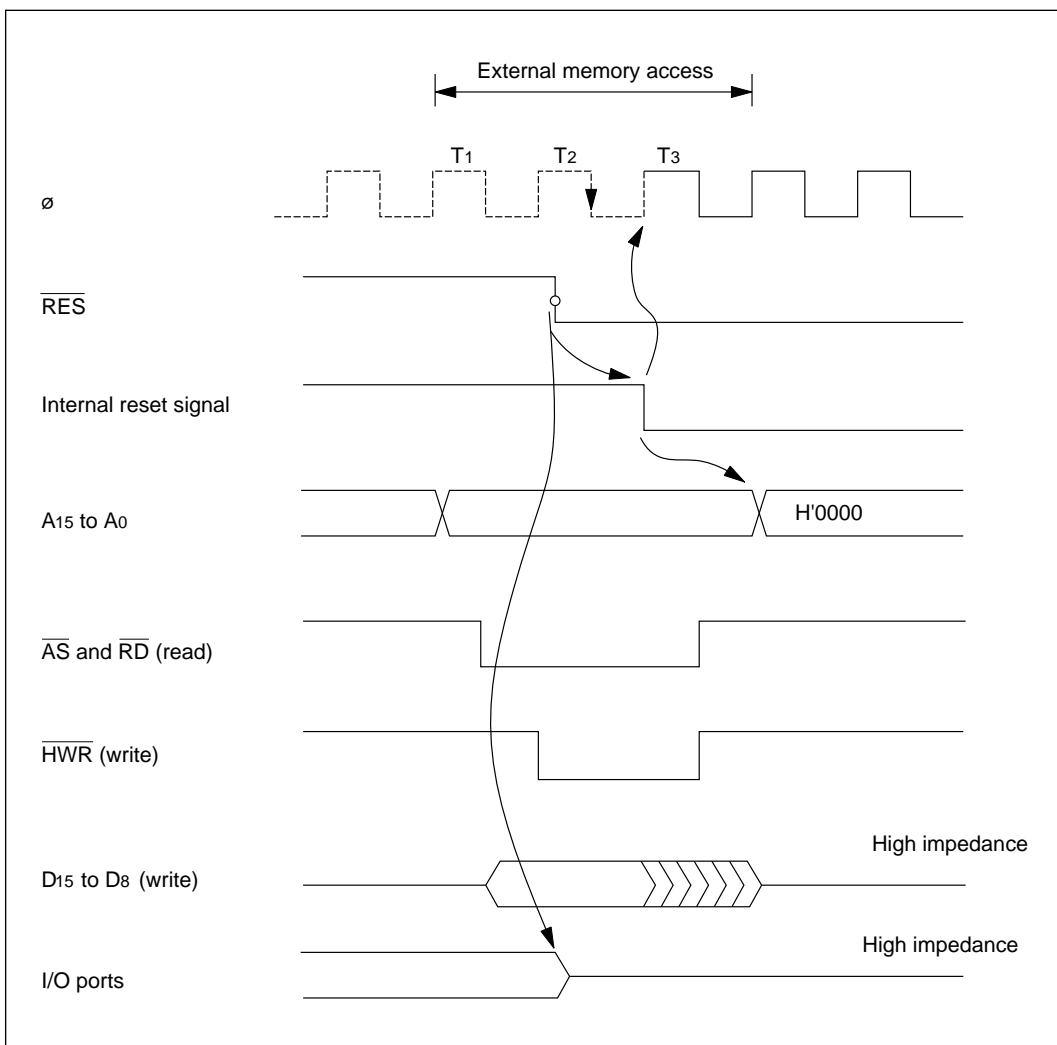


Figure E-1 Reset During Access to Three-State-Access Area (Mode 1)

2. Mode 2

Figure E-2 shows how the pin states change when the $\overline{\text{RES}}$ pin goes low during access to a three-state-access area in mode 2.

As soon as $\overline{\text{RES}}$ goes low, all ports are initialized to the input (high-impedance) state. The $\overline{\text{AS}}$, $\overline{\text{RD}}$, $\overline{\text{HWR}}$, and $\overline{\text{LWR}}$ signals all go high. The data bus (D15 to D0) is placed in the high-impedance state.

Pins A15 to A0 of the address bus are initialized to the Low state 1.5 system clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled.

The clock output pins ϕ and E are initialized 0.5 ϕ clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled. Both pins are initialized to the output state.

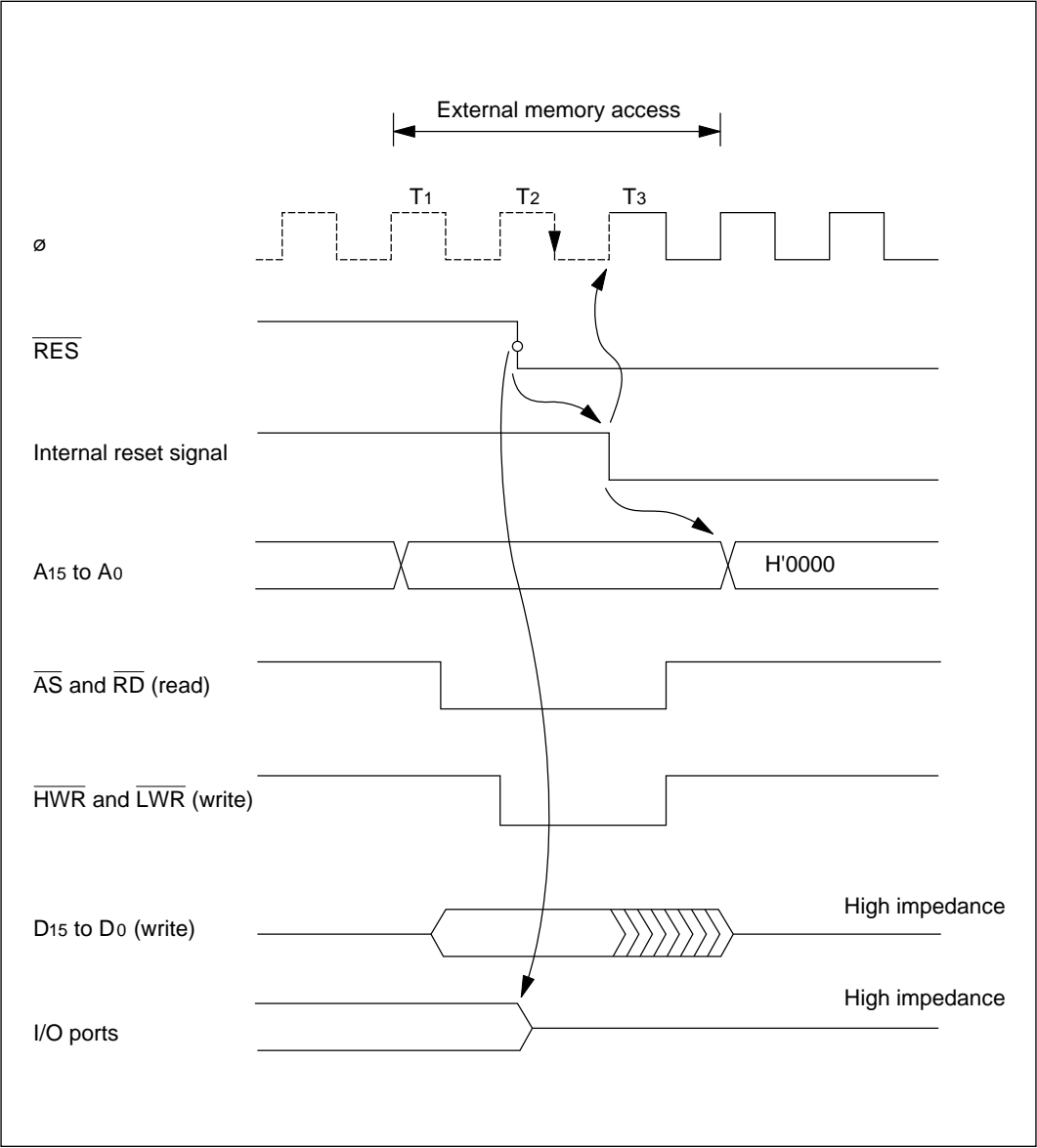


Figure E-2 Reset During Access to Three-State-Access Area (Mode 2)

3. Mode 3

Figure E-3 shows how the pin states change when the $\overline{\text{RES}}$ pin goes low during access to a three-state-access area in mode 3.

As soon as $\overline{\text{RES}}$ goes low, all ports are initialized to the input (high-impedance) state. The $\overline{\text{AS}}$, $\overline{\text{RD}}$, $\overline{\text{HWR}}$, and $\overline{\text{LWR}}$ signals all go high. The data bus (D15 to D8) is placed in the high-impedance state.

The address bus is initialized to the low state 1.5 system clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled.

The clock output pins ϕ and E are initialized 0.5 ϕ clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled. Both pins are initialized to the output state.

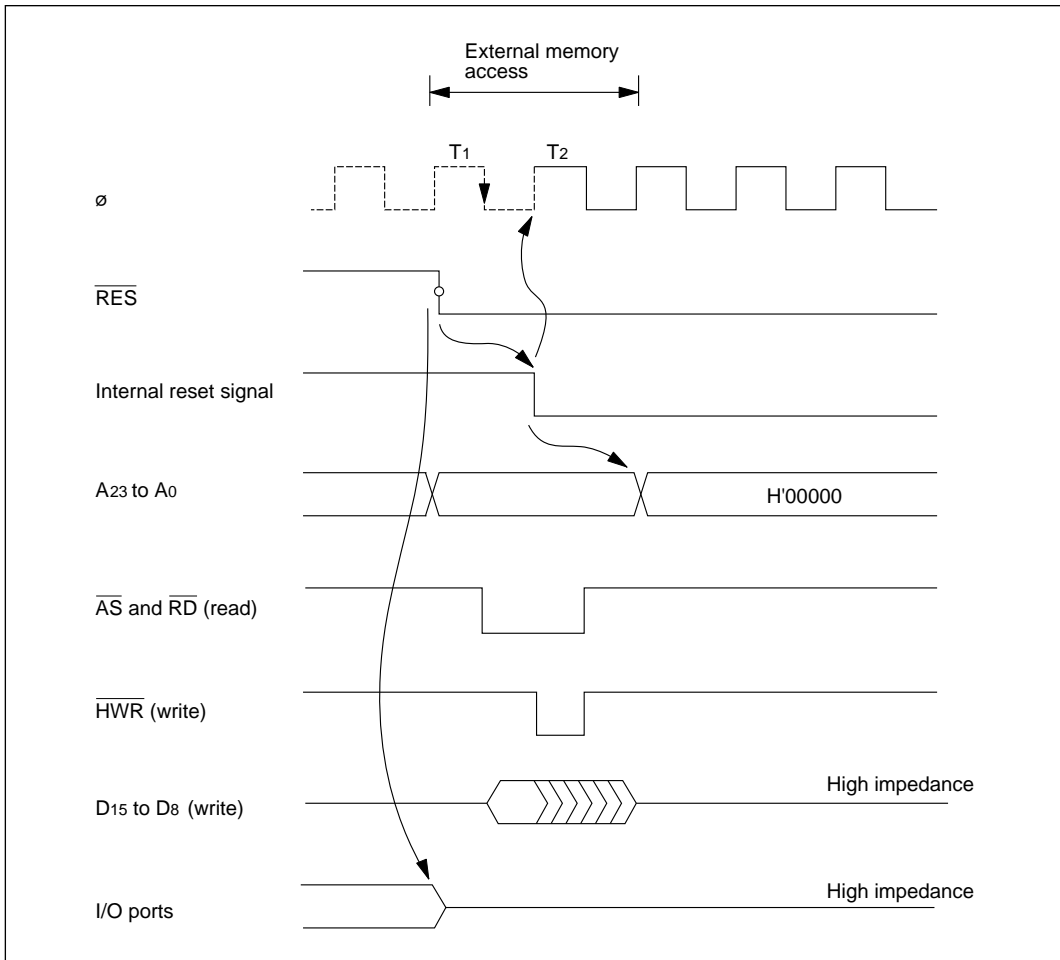


Figure E-3 Reset During Access to Three-State-Access Area (Mode 3)

4. Mode 4

Figure E-4 shows how the pin states change when the \overline{RES} pin goes low during access to a three-state-access area in mode 4.

As soon as \overline{RES} goes low, all ports are initialized to the input (high-impedance) state. The \overline{AS} , \overline{RD} , \overline{HWR} , and \overline{LWR} signals all go high. The data bus (D15 to D0) is placed in the high-impedance state.

Pins A23 to A0 of the address bus are initialized to the Low state 1.5 system clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled.

The clock output pins ϕ and E are initialized 0.5 system clock periods after the low state of the $\overline{\text{RES}}$ pin is sampled. Both pins are initialized to the output state.

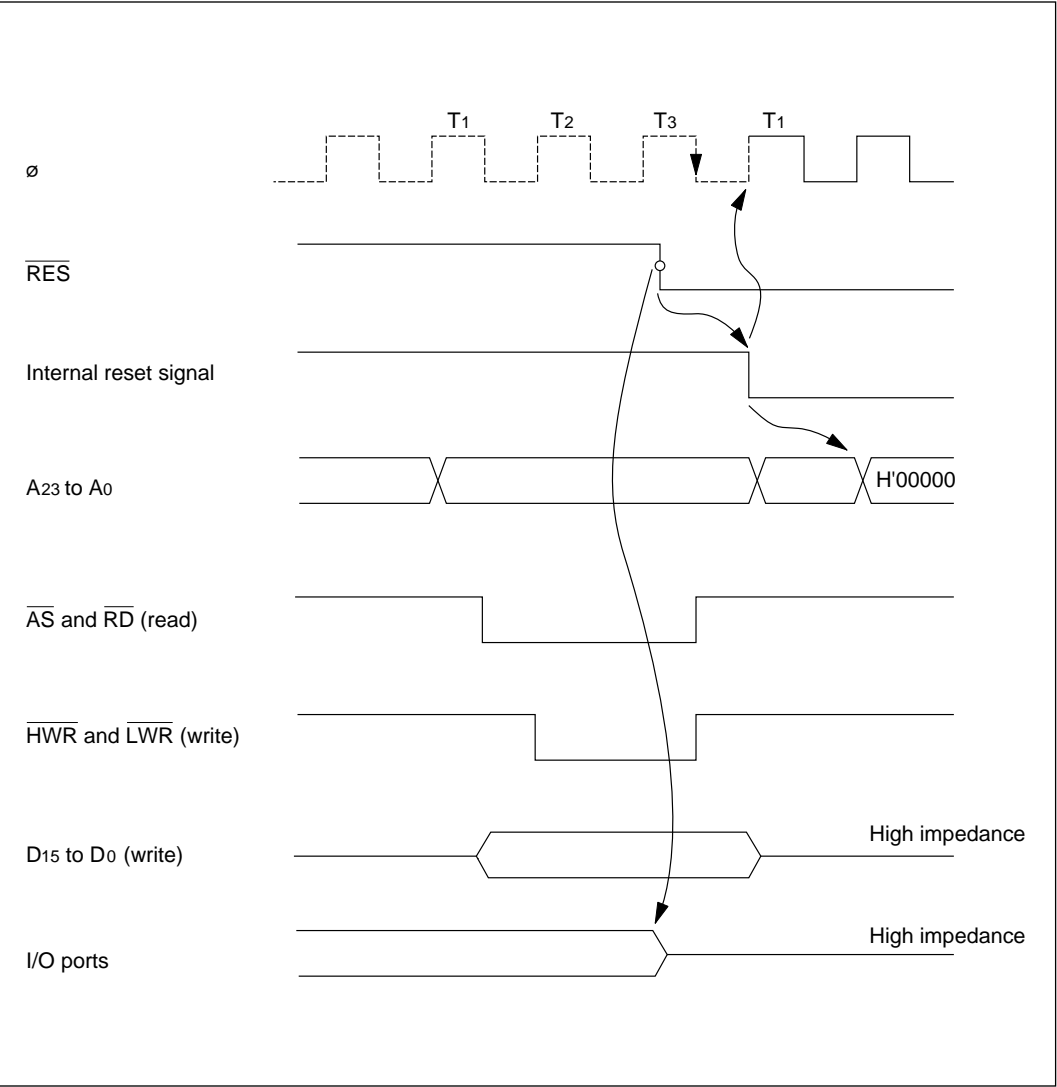


Figure E-4 Reset During Access to Three-State-Area (Mode 4)

Appendix F Package Dimensions

Figure F-1 shows the dimensions of the QFP-112 package.

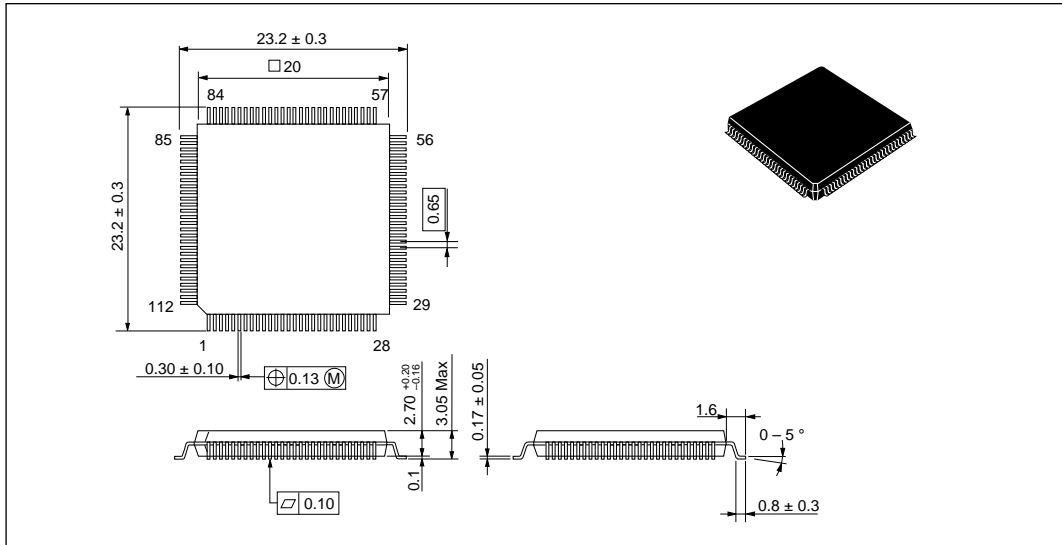


Figure F-1 Package Dimensions (QFP-112)