

## **CVDM-2000**

### **Cellular Voice/Data/GPS Module**

### **Theory of Operation**

**General:** CVDM-2000 is a class I, 3 Watt cellular transceiver board, which operates according to AMPS cellular Standards EIA/TIA 553 and IS19B, that has been integrated with a global positioning system (GPS) receiver, modem and power management system. It is a mobile communications and tracking device that is used for vehicle and asset management.

The 3-Watt cellular transceiver design is identical to our CVDM-3 module (FCC ID: NJI CVDM-3). The CVDM-2000 module was created by extending the board outline of the existing CVDM-3 PCB design and adding the supplementary circuitry required to support the GPS receiver and power management system. Additionally, the modem was moved off of a daughter board and onto the main PCB and several digital I/O and A/D converter inputs were provided.

The receiver consists of a dual conversion superheterodyne receiver circuit as described in section V.2 below. The transmitter consists of a direct modulation of a VCO, which is upconverted by a second oscillator to transmit at the carrier frequency. A programmable power amplifier increases the power level to up to 3 Watts. More detail description of the transmitter is given in section V.3. The calibration and tune up procedure for the transceiver are described in section VII.

All call processing and control functions are performed by the software running on the ACE9050 CPU as described in section V.4.

The GPS receiver, modem, power management system, digital I/O and A/D converter inputs are all controlled by a dedicated function control processor (FCP), as described in section V.5.

The GPS receiver is an OEM module, purchased from Trimble Navigations, as described in section V.6.

During periods when the vehicle engine is not running, the CVDM-2000 generally operates solely on a 12V, 2AH sealed lead acid back-up battery. When the vehicle engine is running, a built-in battery charger, which uses the vehicle battery as its input voltage source, is used to recharge the back-up battery. When the cellular transmitter is on and operating at the maximum output level of 3 Watts, the CVDM-2000 consumes up to 1.5 amps

The functions of all of the critical circuit devices are described in detail in section VI.

**V.2. Receiver:** Receiver chain includes a front-end RF amplifier, a SAW filter, another RF amplifier, and a mixer, which converts the signal to a 45 MHz IF. This first IF signal is filtered by a 4 pole band pass crystal, the output of which is amplified and mixed down

to a 450 kHz second IF. This signal is digitally demodulated to base band. The base band signal is processed by the audio processor chip described in section VI. See the schematics in section IV for more detail.

**V.3. Transmitter:** The transmit signal is processed in base band by the audio processor chip. It's output modulates the transmit VCO. The output of the VCO is mixed with the local oscillator to up convert to the carrier frequency. It is then filtered by a SAW band pass filter and amplified before feeding the power amplifier. The output of the power amplifier is then filtered by the duplexer, which is connected to the antenna connector. See the schematics in section IV for more detail.

**V.4 CPU:** The AMPS, CPU functions are performed by the ACE9050 chip described in section VI. The software consists of a dual tasking operating system (DTOS), a foreground task performing all of the call processing functions and a background task controlling the test and configuration functions. See the schematics in section IV for more detail.

**V.5 FCP:** The CVDM2000 primary system controller is the Hitachi H8/3644 Function Control Processor (FCP) described in section VI. The FCP is responsible for power management functions, scheduling the various activities such as GPS position fixes and cellular phone calls, maintaining the Real-Time Clock (RTC), and monitoring/controlling external I/O. The software consists of a single main-loop application process. Real-time issues are handled either directly through interrupt handlers or with the use of a 2 msec interval timer which provides a set of software timers used to guarantee periodic service. See the schematics in section IV for more detail.

**V.6 GPS Module:** The GPS module is Lassen-SK8 board which is purchased as an OEM unit from Trimble Navigations. It has an 8 channel, continuous tracking receiver and 32 correlators. The module is screwed onto the CVDM-2000 module and the I/O interface to the Hitachi FCP is made through an 8 pin header. See the applications information provided in section VI for more detail.

## Lassen™-SK *GPS Board for Fast Integratio*

*Highest performance, smaller size, and  
lower power consumption for embedded application*

Introducing the Lassen-SK8 GPS board, Trimble's new OEM GPS module for embedded applications based on Sierra™ GPS technology. Using the latest 8-channel technology, Lassen-SK8 offers the highest performance available through a miniature GPS receiver. Two-thirds the size of a business card, Lassen-SK8's extremely low power consumption, 20-second hot acquisition capabilities, and two-meter differential accuracy readings make it the choice for demanding applications.

Lassen-SK8's power consumption is a mere 0.75 watts. This means that battery-powered GPS applications can start up and calculate positions with less power than any other module available today. With the fastest reacquisition time available,

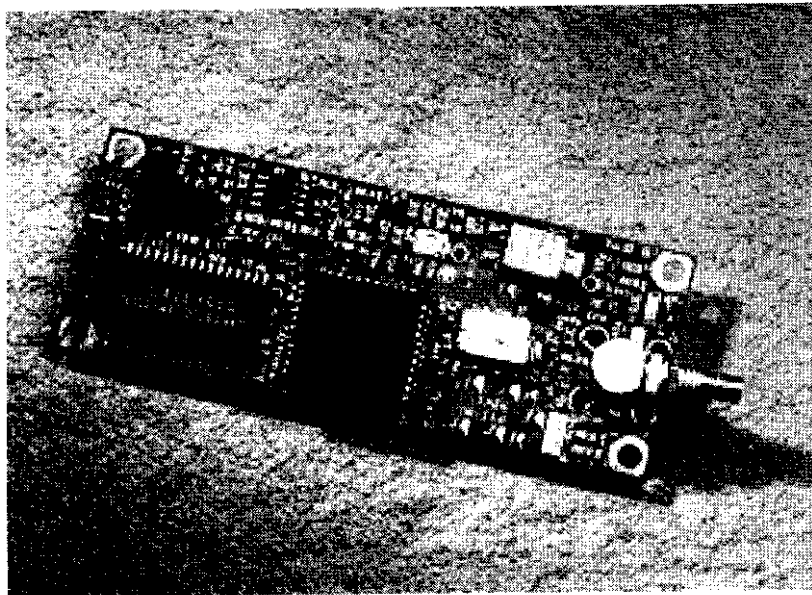
Lassen-SK8 also delivers position data faster and more often. Lassen-SK8 even provides this information in areas where satellite signals are inhibited by terrain and structures.

If your application requires differential GPS accuracy, Lassen-SK8 is more than ready. Using industry standard RTCM SC-104 correction data or TSIP format corrections, Lassen-SK8 provides the highest level of DGPS available in a miniature GPS receiver. The optional full-measurement feature positions you well ahead of the competition.

Dual serial I/O ports mean greater flexibility and faster integration. One serial port uses Trimble's renowned TSIP binary data protocol to provide maximum data and control over your GPS receiver. The second port

outputs your choice of NMEA 0183 Version 2.1 standard data messages and receives RTCM SC-104 differential corrections. While TSIP data packets are output once each second, you can configure NMEA messages to output anywhere from once per second to once every 20 seconds. Use TSIP commands to configure and permanently store your serial port settings and NMEA data selections.

Lassen-SK8's Starter Kit provides everything you need to get started integrating state-of-the-art GPS into your application. The new 12-volt, RS-232 interface module and Toolkit Software included in the Starter Kit make the Lassen-SK8 GPS board computer-ready for your evaluation.



*Actual size*

# Lassen™SK8

## GPS Board for Fast Integration

### Performance Specifications

<b>General:</b>	L1 frequency, C/A code (SPS), 8-channel, continuous tracking receiver, 32 correlators
<b>Update rate:</b>	TSIP @ 1 Hz; NMEA @ 1 Hz
<b>Accuracy:</b>	Position: 25 m CEP (50%) without SA Velocity: 0.1 m/sec without SA Time: ±500 nano-seconds (nominal)
<b>DGPS accuracy:</b>	Position: 2 m CEP (50%) Velocity: 0.05 m/sec Time: ±500 nano-seconds (nominal)
<b>Acquisition (typical):</b>	Cold start: <2 minutes (90%) Warm start: <45 seconds (90%) Hot start: <20 seconds (90%)  <i>Cold start requires no initialization. Warm start implies last position, time and almanac are saved by back-up power. Hot start implies ephemeris also saved.</i>
<b>Reacquisition after signal loss:</b>	< 2 seconds (90%)
<b>Dynamics:</b>	Altitude: -1000 m to +18,000 m Velocity: 515 m/sec maximum Acceleration: 4g (39.2 m/sec²) Motional Jerk: 20 m/sec³

### Environmental Specifications

<b>Operating temp:</b>	-10°C to +60°C (standard) -40°C to +85°C (optional)
<b>Storage temp:</b>	-55°C to +100°C
<b>Vibration:</b>	0.008g²/Hz 5 Hz to 20 Hz 0.05g²/Hz 20 Hz to 100 Hz 3dB/octave 100 Hz to 900 Hz
<b>Operating humidity:</b>	5% to 95% R.H. non-condensing, @60°C
<b>Altitude:</b>	-400 m to +18,000 m

### Physical Characteristics

<b>Prime power:</b>	+5 volts DC, ±5%
<b>Power consumption (nominal):</b>	GPS board only: 150 ma, 0.75 watts With antenna: 175 ma, 0.88 watts
<b>Back-up power:</b>	+3.2 to +5 volts DC 2 micro-amp @ +3.5 volts, +25°C (nominal)
<b>Serial ports/1PPS:</b>	CMOS TTL levels
<b>Protocols:</b>	TSIP @ 9600 baud, 8-Odd-1 NMEA 0183 v2.1 @ 4800 baud, 8-None-1 RTCM SC-104 @ 4800 baud, 8-None-1
<b>NMEA messages:</b>	GGA, VTG, GLL, ZDA, GSV, GSA and RMC messages selectable by TSIP command; selection stored in non-volatile memory
<b>Antenna power:</b>	5V at 25mA available Short circuit protection Feedline fault detection

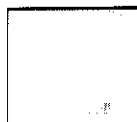
### Physical Characteristics

<b>Dimensions:</b>	3.25" L x 1.25" W x 0.40" H (82.6 mm x 31.2 mm x 10.2 mm) without connectors 3.29" L x 1.30" W x 0.52" H (83.6 mm x 33.0 mm x 13.2 mm) with connectors and optional shield
<b>Weight:</b>	0.7 oz. (19.6 g) without optional shield 1.3 oz. (36.4 g) with optional shield
<b>Connectors:</b>	RF: SMB; I/O: 8-pin (2x4), 0.100" header

### Accessories



**GPS antenna:** Compact, active micropatch antenna with 5-meter cable and magnetic mount. 1.60" x 1.90" x 0.55" high (40.6 mm x 48.3 mm x 13.9 mm)



**Hard mount antenna:** Compact, hard mount, active micropatch antenna with single-hole 0.75" threaded mount and TNC connector. 2.46" diameter x 0.75" high (62.6 mm x 19.0 mm)



**Rooftop antenna:** Bullet antenna with 22-meter cable and SMB adapter



**RF shield:** Optional snap-on metal cover for severe RF environments

### Ordering Information

<b>Modules:</b>	
28835-00	Standard Temperature, TSIP (binary) protocol and NMEA 0183 (ASCII) protocol, DGPS ready
28835-50	Extended Temperature, TSIP (binary) protocol and NMEA 0183 (ASCII) protocol, DGPS ready
<b>Antennas:</b>	
28367-00	26 dB magnetic mount antenna, 5-meter cable
28367-70	26 dB hard mount antenna, TNC connector
23726-00	35 dB rooftop Bullet antenna, 23-meter cable
<b>Starter Kit:</b>	
29467-00	Includes Lassen-SK8 board, interface motherboard in durable metal enclosure with dual DB9, RS-232 interface, AC/DC power converter, magnetic mount antenna, TSIP and NMEA protocols, software toolkit for TSIP, interface cable, and manual
<b>Manual:</b>	
29473-00	Lassen-SK8 System Designer Reference Guide

## Section 1 Overview

### 1.1 Overview

The H8/300L Series is a series of single-chip microcomputers (MCU: microcomputer unit), built around the high-speed H8/300L CPU and equipped with peripheral system functions on-chip.

Within the H8/300L Series, the H8/3644 Series of microcomputers are equipped with a UART (Universal Asynchronous Receiver/Transmitter). Other on-chip peripheral functions include five timers, a 14-bit pulse width modulator (PWM), two serial communication interface channels, and an A/D converter. Together, these functions make the H8/3644 Series ideally suited for embedded applications in advanced control systems. The ZTAT™\* versions of the H8/3644 come with user-programmable PROM. Table 1 summarizes the features of the H8/3644 Series..

Table 1 summarizes the features of the H8/3927 Series.

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

**Table 1-1 Features**

Item	Description
CPU	<div>High-speed H8/300L CPU</div> <ul style="list-style-type: none"><li>• General-register architecture<div>General registers: Sixteen 8-bit registers (can be used as eight 16-bit registers)</div></li><li>• Operating speed<ul style="list-style-type: none"><li>— Max. operating speed: 5 MHz</li><li>— Add/subtract: 0.4 <math>\mu</math>s (operating at 5 MHz)</li><li>— Multiply/divide: 2.8 <math>\mu</math>s (operating at 5 MHz)</li><li>— Can run on 32.768 kHz subclock</li></ul></li><li>• Instruction set compatible with H8/300 CPU<ul style="list-style-type: none"><li>— Instruction length of 2 bytes or 4 bytes</li><li>— Basic arithmetic operations between registers</li><li>— MOV instruction for data transfer between memory and registers</li></ul></li><li>• Typical instructions<ul style="list-style-type: none"><li>— Multiply (8 bits <math>\times</math> 8 bits)</li><li>— Divide (16 bits <math>\div</math> 8 bits)</li><li>— Bit accumulator</li><li>— Register-indirect designation of bit position</li></ul></li></ul>

**Table 1-1 Features (cont)**

<b>Item</b>	<b>Description</b>
Interrupts	33 interrupt sources <ul style="list-style-type: none"> <li>• 12 external interrupt sources (IRQ<sub>3</sub> to IRQ<sub>0</sub>, INT<sub>7</sub> to INT<sub>0</sub>)</li> <li>• 21 internal interrupt sources</li> </ul>
Clock pulse generators	Two on-chip clock pulse generators <ul style="list-style-type: none"> <li>• System clock pulse generator: 1 to 10 MHz</li> <li>• Subclock pulse generator: 32.768 kHz</li> </ul>
Power-down modes	Seven power-down modes <ul style="list-style-type: none"> <li>• Sleep (high-speed) mode</li> <li>• Sleep (medium-speed) mode</li> <li>• Standby mode</li> <li>• Watch mode</li> <li>• Subsleep mode</li> <li>• Subactive mode</li> <li>• Active (medium-speed) mode</li> </ul>
Memory	Large on-chip memory <ul style="list-style-type: none"> <li>• H8/3644: 32-kbyte ROM, 1-kbyte RAM</li> <li>• H8/3643: 24-kbyte ROM, 1-kbyte RAM</li> <li>• H8/3642: 16-kbyte ROM, 512 byte RAM</li> <li>• H8/3641: 12-kbyte ROM, 512 byte RAM</li> <li>• H8/3640: 8-kbyte ROM, 512 byte RAM</li> </ul>
I/O ports	53 pins <ul style="list-style-type: none"> <li>• 45 I/O pins</li> <li>• 8 input pins</li> </ul>
Timers	Five on-chip timers <ul style="list-style-type: none"> <li>• Timer A: 8-bit timer               <p>Count-up timer with selection of eight internal clock signals divided from the system clock (<math>\phi</math>)* and four clock signals divided from the watch clock (<math>\phi_w</math>)*</p> </li> <li>• Timer B1: 8-bit timer               <ul style="list-style-type: none"> <li>— Count-up timer with selection of seven internal clock signals or event input from external pin</li> <li>— Auto-reloading</li> </ul> </li> </ul> <p>Note: * <math>\phi</math> and <math>\phi_w</math> are defined in section 4, Clock Pulse Generators.</p>

**Table 1-1 Features (cont)**

<b>Item</b>	<b>Description</b>
Timers	<ul style="list-style-type: none"> <li>• Timer V: 8-bit timer <ul style="list-style-type: none"> <li>— Count-up timer with selection of six internal clock signals or event input from external pin</li> <li>— Compare-match waveform output</li> <li>— Externally triggerable</li> </ul> </li> <li>• Timer X: 16-bit timer <ul style="list-style-type: none"> <li>— Count-up timer with selection of three internal clock signals or event input from external pin</li> <li>— Output compare (2 output pins)</li> <li>— Input capture (4 input pins)</li> </ul> </li> <li>• Watchdog timer <ul style="list-style-type: none"> <li>— Reset signal generated by 8-bit counter overflow</li> </ul> </li> </ul> <p>Note: * See section 4, Clock Pulse Generator for the definitions of <math>\phi</math> and <math>\phi_{WV}</math>.</p>
Serial communication interface	<p>Two channels on chip</p> <ul style="list-style-type: none"> <li>• SCI1: synchronous serial interface <ul style="list-style-type: none"> <li>Choice of 8-bit or 16-bit data transfer</li> </ul> </li> <li>• SCI3: 8-bit synchronous/asynchronous serial interface <ul style="list-style-type: none"> <li>Incorporates multiprocessor communication function</li> </ul> </li> </ul>
14-bit PWM	<p>Pulse-division PWM output for reduced ripple</p> <ul style="list-style-type: none"> <li>• Can be used as a 14-bit D/A converter by connecting to an external low-pass filter.</li> </ul>
A/D converter	<p>Successive approximations using a resistance ladder</p> <ul style="list-style-type: none"> <li>• 8-channel analog input pins</li> <li>• Conversion time: <math>31/\phi</math> or <math>62/\phi</math> per channel</li> </ul>

**Table 1-1 Features (cont)**

<b>Item</b>	<b>Specification</b>				
Product lineup	Product Code			Package	ROM/RAM Size
	Mask ROM Version	ZTAT™ Version	F-ZTAT® Version		
HD6433644H		HD6473644H	HD64F3644H	64-pin QFP (FP-64A)	ROM: 32 kbytes RAM: 1 kbyte
HD6433643H		—	HD64F3644P		ROM: 24 kbytes RAM: 1 kbyte
HD6433642H		—	—		ROM: 16 kbytes RAM: 1 kbyte
HD643341H		—	—		ROM: 12 kbytes RAM: 512 bytes
HD6433644P		HD6473644P	—	64-pin SDIP (DP-64S)	ROM: 32 kbytes RAM: 1 kbyte
HD6433643P		—	—		ROM: 24 kbytes RAM: 1 kbyte
HD6433642P		—	—		ROM: 16 kbytes RAM: 512 bytes
HD643341P		—	—		ROM: 12 kbytes RAM: 512 bytes
HD6433640H		—	—	64-pin QFP (FP-64A)	ROM: 8 kbytes RAM: 512 bytes
HD6433640P		—	—	64-pin SDIP (DP-64S)	



## 1.2 Internal Block Diagram

Figure 1-1 shows a block diagram of the H8/3644 Series.

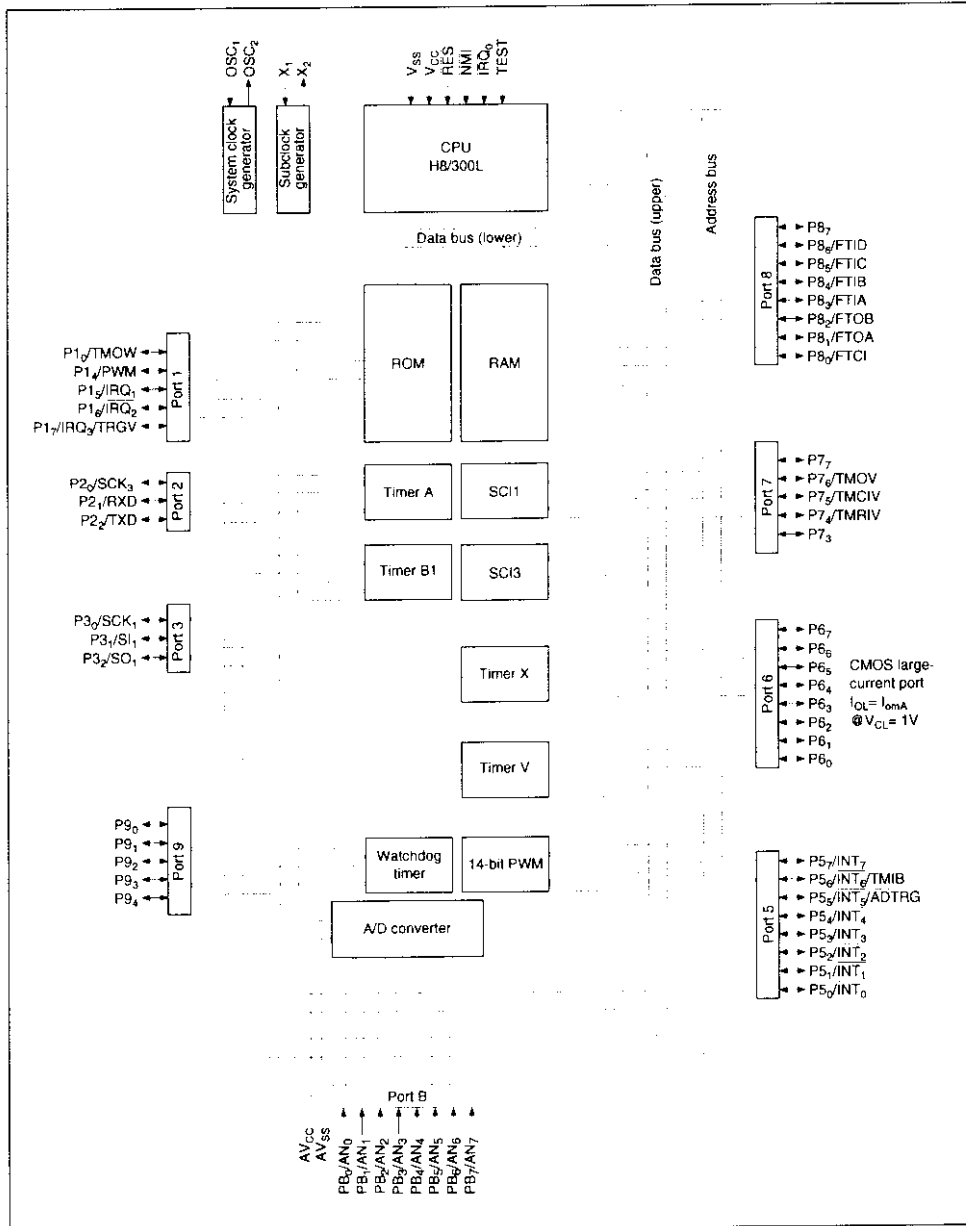


Figure 1-1 Block Diagram



P1 <sub>7</sub> /IRQ <sub>9</sub> /TRGV	1	64	P1 <sub>6</sub> /IRQ <sub>2</sub>
AV <sub>CC</sub>	2	63	P1 <sub>5</sub> /IRQ <sub>1</sub>
PB <sub>7</sub> /AN <sub>7</sub>	3	62	P1 <sub>4</sub> /PWM
PB <sub>6</sub> /AN <sub>6</sub>	4	61	P1 <sub>3</sub> /TMOW
PB <sub>5</sub> /AN <sub>5</sub>	5	60	P3 <sub>0</sub> /SCK <sub>1</sub>
PB <sub>4</sub> /AN <sub>4</sub>	6	59	P3 <sub>1</sub> /SI <sub>1</sub>
PB <sub>3</sub> /AN <sub>3</sub>	7	58	P3 <sub>2</sub> /SO <sub>1</sub>
PB <sub>2</sub> /AN <sub>2</sub>	8	57	P2 <sub>2</sub> /TXD
PB <sub>1</sub> /AN <sub>1</sub>	9	56	P2 <sub>1</sub> /RXD
PB <sub>0</sub> /AN <sub>0</sub>	10	55	P2 <sub>0</sub> /SCK <sub>3</sub>
AV <sub>SS</sub>	11	54	P8 <sub>7</sub>
TEST	12	53	P8 <sub>6</sub> /FTID
X <sub>2</sub>	13	52	P8 <sub>5</sub> /FTIC
X <sub>1</sub>	14	51	P8 <sub>4</sub> /FTIB
V <sub>SS</sub>	15	50	P8 <sub>3</sub> /FTIA
OSC <sub>1</sub>	16	49	P8 <sub>2</sub> /FTOB
OSC <sub>2</sub>	17	48	P8 <sub>1</sub> /FTOA
RES	18	47	P8 <sub>0</sub> /FTCI
P9 <sub>0</sub>	19	46	P7 <sub>7</sub>
P9 <sub>1</sub>	20	45	P7 <sub>6</sub> /TMOV
P9 <sub>2</sub>	21	44	P7 <sub>5</sub> /TMCIV
P9 <sub>3</sub>	22	43	P7 <sub>4</sub> /TMRIV
P9 <sub>4</sub>	23	42	P7 <sub>3</sub>
IRQ <sub>0</sub>	24	41	V <sub>CC</sub>
P6 <sub>0</sub>	25	40	P5 <sub>7</sub> /INT <sub>7</sub>
P6 <sub>1</sub>	26	39	P5 <sub>6</sub> /INT <sub>6</sub> /TMIB
P6 <sub>2</sub>	27	38	P5 <sub>5</sub> /INT <sub>5</sub> /ADTRG
P6 <sub>3</sub>	28	37	P5 <sub>4</sub> /INT <sub>4</sub>
P6 <sub>4</sub>	29	36	P5 <sub>3</sub> /INT <sub>3</sub>
P6 <sub>5</sub>	30	35	P5 <sub>2</sub> /INT <sub>2</sub>
P6 <sub>6</sub>	31	34	P5 <sub>1</sub> /INT <sub>1</sub>
P6 <sub>7</sub>	32	33	P5 <sub>0</sub> /INT <sub>0</sub>

**Figure 1-3 Pin Arrangement (DP-64S: Top View)**

### 1.3.2 Pin Functions

Table 1-2 outlines the pin functions of the H8/3644 Series.

**Table 1-2 Pin Functions**

Type	Symbol	Pin No.		I/O	Name and Functions
		FP-64A	DP-64S		
Power source pins	V <sub>CC</sub>	33	41	Input	<b>Power supply:</b> All V <sub>CC</sub> pins should be connected to the system power supply (+5 V).
	V <sub>SS</sub>	7	15	Input	<b>Ground:</b> All V <sub>SS</sub> pins should be connected to the system power supply (0 V).
	AV <sub>CC</sub>	58	2	Input	<b>Analog power supply:</b> This is the power supply pin for the A/D converter. When the A/D converter is not used, connect this pin to the system power supply (+5 V).
	AV <sub>SS</sub>	3	11	Input	<b>Analog ground:</b> This is the A/D converter ground pin. It should be connected to the system power supply (0 V).
Clock pins	OSC <sub>1</sub>	8	16	Input	<b>System clock:</b> These pins connect to a crystal or ceramic oscillator, or can be used to input an external clock. See section 4, Clock Pulse Generators, for a typical connection diagram.
	OSC <sub>2</sub>	9	17	Output	
	X <sub>1</sub>	6	14	Input	<b>Subclock:</b> These pins connect to a 32.768-kHz crystal oscillator. See section 4, Clock Pulse Generators, for a typical connection diagram.
	X <sub>2</sub>	5	13	Output	
System control	RES	10	18	Input	<b>Reset:</b> When this pin is driven low, the chip is reset.
	TEST	4	12	Input	<b>Test:</b> This is a test pin, not for use in application systems. It should be connected to V <sub>SS</sub> .

**Table 1-2 Pin Functions (cont)**

Type	Symbol	Pin No.		I/O	Name and Functions
		FP-64A	DP-64S		
Interrupt pins	IRQ <sub>0</sub>	16	24	Input	<b>IRQ interrupt request 0 to 3:</b> These are input pins for edge-sensitive external interrupts, with a selection of rising or falling edge
	IRQ <sub>1</sub>	55	63		
	IRQ <sub>2</sub>	56	64		
	IRQ <sub>3</sub>	57	1		
	INT <sub>7</sub> to INT <sub>0</sub>	24 to 17	32 to 25	Input	<b>INT interrupt request 0 to 7:</b> These are input pins for edge-sensitive external interrupts, with a selection of rising or falling edge
Timer pins	TMOW	53	61	Output	<b>Clock output:</b> This is an output pin for waveforms generated by the timer A output circuit
	TMIB	23	31	Input	<b>Timer B1 event counter input:</b> This is an event input pin for input to the timer B1 counter
	TMOV	37	45	Output	<b>Timer V output:</b> This is an output pin for waveforms generated by the timer V output compare function
	TMCIV	36	44	Input	<b>Timer V event input:</b> This is an event input pin for input to the timer V counter
	TMRIV	35	43	Input	<b>Timer V counter reset:</b> This is a counter reset input pin for timer V
	TRGV	57	1	Input	<b>Timer V counter trigger input:</b> This is a trigger input pin for the timer V counter and realtime output port
	FTCI	39	47	Input	<b>Timer X clock input:</b> This is an external clock input pin for input to the timer X counter
	FTOA	40	48	Output	<b>Timer X output compare A output:</b> This is an output pin for timer X output compare A
	FTOB	41	49	Output	<b>Timer X output compare B output:</b> This is an output pin for timer X output compare B
	FTIA	42	50	Input	<b>Timer X input capture A input:</b> This is an input pin for timer X input capture B
	FTIB	43	51	Input	<b>Timer X input capture B input:</b> This is an input pin for timer X input capture B
	FTIC	44	52	Input	<b>Timer X input capture C input:</b> This is an input pin for timer X input capture C

**Table 1-2 Pin Functions (cont)**

Type	Symbol	Pin No.		I/O	Name and Functions
		FP-64A	DP-64S		
Timer pins	FTID	45	53	Input	<b>Timer X input capture C input:</b> This is an input pin for timer X input capture D
14-bit PWM pin	PWM	54	62	Output	<b>14-bit PWM output:</b> This is an output pin for waveforms generated by the 14-bit PWM
I/O ports	PB <sub>7</sub> to PB <sub>0</sub>	59 to 64, 1 to 2	3 to 10	Input	<b>Port B:</b> This is an 8-bit input port
	P1 <sub>7</sub> to P1 <sub>4</sub> , P1 <sub>0</sub>	57 to 53	1, 64 to 61	I/O	<b>Port 1:</b> This is a 5-bit I/O port. Input or output can be designated for each bit by means of port control register 1 (PCR1)
	P2 <sub>2</sub> to P2 <sub>0</sub>	49 to 47	57 to 55	I/O	<b>Port 2:</b> This is a 3-bit I/O port. Input or output can be designated for each bit by means of port control register 2 (PCR2)
	P3 <sub>2</sub> to P3 <sub>0</sub>	50 to 52	58 to 60	I/O	<b>Port 3:</b> This is a 3-bit I/O port. Input or output can be designated for each bit by means of port control register 3 (PCR3)
	P5 <sub>7</sub> to P5 <sub>0</sub>	32 to 25	40 to 33	I/O	<b>Port 5:</b> This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 5 (PCR5)
	P6 <sub>7</sub> to P6 <sub>0</sub>	24 to 17	32 to 25	I/O	<b>Port 6:</b> This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 6 (PCR6)
	P7 <sub>7</sub> to P7 <sub>3</sub>	38 to 34	46 to 42	I/O	<b>Port 7:</b> This is a 5-bit I/O port. Input or output can be designated for each bit by means of port control register 7 (PCR7)
	P8 <sub>7</sub> to P8 <sub>0</sub>	46 to 39	54 to 47	I/O	<b>Port 8:</b> This is an 8-bit I/O port. Input or output can be designated for each bit by means of port control register 8 (PCR8)
	P9 <sub>4</sub> to P9 <sub>0</sub>	15 to 11	23 to 19	I/O	<b>Port 9:</b> This is a 5-bit I/O port. Input or output can be designated for each bit by means of port control register 9 (PCR9)

**Table 1-2 Pin Functions (cont)**

Type	Symbol	Pin No.		I/O	Name and Functions
		FP-64A	DP-64S		
Serial communication interface (SCI)	SI <sub>1</sub>	51	59	Input	<b>SCI1 receive data input:</b> This is the SCI1 data input pin
	SO <sub>1</sub>	50	58	Output	<b>SCI1 send data output:</b> This is the SCI1 data output pin
	SCK <sub>1</sub>	52	60	I/O	<b>SCI1 clock I/O :</b> This is the SCI1 clock I/O pin
	RXD	48	56	Input	<b>SCI3 receive data input:</b> This is the SCI3 data input pin
	TXD	49	57	Output	<b>SCI3 send data output:</b> This is the SCI3 data output pin
	SCK <sub>3</sub>	47	55	I/O	<b>SCI3 clock I/O:</b> This is the SCI3 clock I/O pin
A/D converter	AN7 to AN0	59 to 64, 1 to 2	3 to 10	Input	<b>Analog input channels 7 to 0:</b> These are analog data input channels to the A/D converter
	ADTRG	22	30	Input	<b>A/D converter trigger input:</b> This is the external trigger input pin to the A/D converter

## Section 2 CPU

### 2.1 Overview

The H8/300L CPU has sixteen 8-bit general registers, which can also be paired as eight 16-bit registers. Its concise instruction set is designed for high-speed operation.

#### 2.1.1 Features

Features of the H8/300L CPU are listed below.

- General-register architecture  
Sixteen 8-bit general registers, also usable as eight 16-bit general registers
- Instruction set with 55 basic instructions, including:
  - Multiply and divide instructions
  - Powerful bit-manipulation instructions
- Eight addressing modes
  - Register direct
  - Register indirect
  - Register indirect with displacement
  - Register indirect with post-increment or pre-decrement
  - Absolute address
  - Immediate
  - Program-counter relative
  - Memory indirect
- 64-kbyte address space
- High-speed operation
  - All frequently used instructions are executed in two to four states
  - High-speed arithmetic and logic operations
  - 8- or 16-bit register-register add or subtract: 0.4  $\mu$ s\*
  - 8  $\times$  8-bit multiply: 2.8  $\mu$ s\*
  - 16  $\div$  8-bit divide: 2.8  $\mu$ s\*
- Low-power operation modes  
SLEEP instruction for transfer to low-power operation

Note: \* These values are at  $\phi = 5$  MHz.



### 2.1.2 Address Space

The H8/300L CPU supports an address space of up to 64 kbytes for storing program code and data.

See 2.8, Memory Map, for details of the memory map.

### 2.1.3 Register Configuration

Figure 2-1 shows the register structure of the H8/300L CPU. There are two groups of registers: the general registers and control registers.

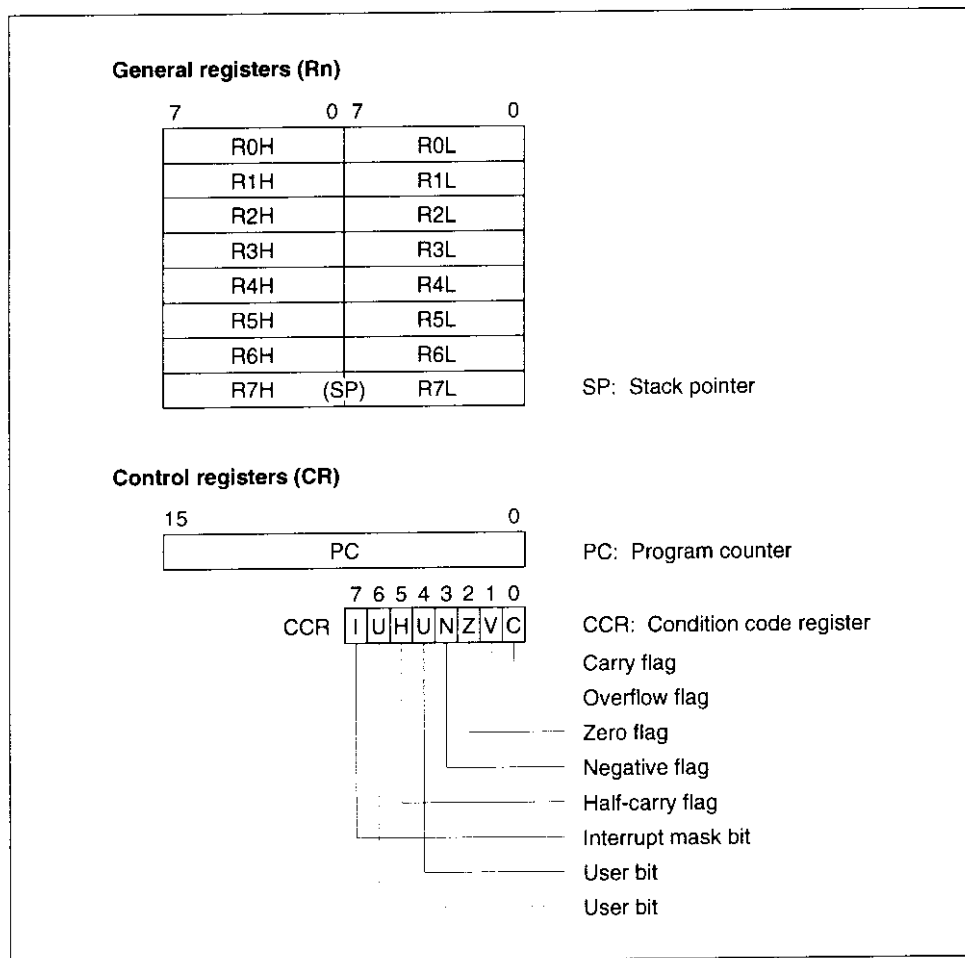


Figure 2-1 CPU Registers

## 2.2 Register Descriptions

### 2.2.1 General Registers

All the general registers can be used as both data registers and address registers.

When used as data registers, they can be accessed as 16-bit registers (R0 to R7), or the high bytes (R0H to R7H) and low bytes (R0L to R7L) can be accessed separately as 8-bit registers.

When used as address registers, the general registers are accessed as 16-bit registers (R0 to R7).

R7 also functions as the stack pointer (SP), used implicitly by hardware in exception processing and subroutine calls. When it functions as the stack pointer, as indicated in figure 2-2, SP (R7) points to the top of the stack.

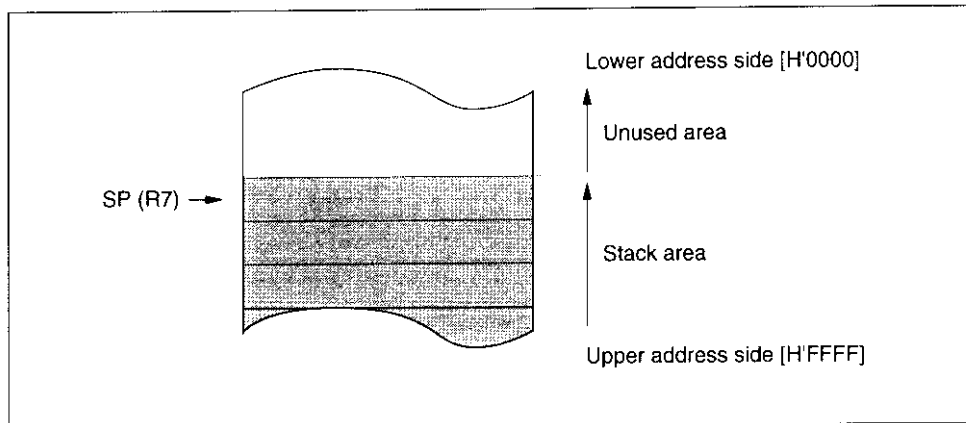


Figure 2-2 Stack Pointer

### 2.2.2 Control Registers

The CPU control registers include a 16-bit program counter (PC) and an 8-bit condition code register (CCR).

**Program Counter (PC):** This 16-bit register indicates the address of the next instruction the CPU will execute. All instructions are fetched 16 bits (1 word) at a time, so the least significant bit of the PC is ignored (always regarded as 0).

**Condition Code Register (CCR):** This 8-bit register contains internal status information, including the interrupt mask bit (I) and half-carry (H), negative (N), zero (Z), overflow (V), and carry (C) flags. These bits can be read and written by software (using the LDC, STC, ANDC, ORC, and XORC instructions). The N, Z, V, and C flags are used as branching conditions for conditional branching (Bcc) instructions.

**Bit 7—Interrupt Mask Bit (I):** When this bit is set to 1, interrupts are masked. This bit is set to 1 automatically at the start of exception handling. The interrupt mask bit may be read and written by software. For further details, see section 3.3, Interrupts.

**Bit 6—User Bit (U):** Can be used freely by the user.

**Bit 5—Half-Carry Flag (H):** When the ADD.B, ADDX.B, SUB.B, SUBX.B, CMP.B, or NEG.B instruction is executed, this flag is set to 1 if there is a carry or borrow at bit 3, and is cleared to 0 otherwise.

The H flag is used implicitly by the DAA and DAS instructions.

When the ADD.W, SUB.W, or CMP.W instruction is executed, the H flag is set to 1 if there is a carry or borrow at bit 11, and is cleared to 0 otherwise.

**Bit 4—User Bit (U):** Can be used freely by the user.

**Bit 3—Negative Flag (N):** Indicates the most significant bit (sign bit) of the result of an instruction.

**Bit 2—Zero Flag (Z):** Set to 1 to indicate a zero result, and cleared to 0 to indicate a non-zero result.

**Bit 1—Overflow Flag (V):** Set to 1 when an arithmetic overflow occurs, and cleared to 0 at other times.

**Bit 0—Carry Flag (C):** Set to 1 when a carry occurs, and cleared to 0 otherwise. Used by:

- Add instructions, to indicate a carry
- Subtract instructions, to indicate a borrow
- Shift and rotate instructions, to store the value shifted out of the end bit

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

Some instructions leave some or all of the flag bits unchanged.

Refer to the *H8/300L Series Programming Manual* for the action of each instruction on the flag bits.

### 2.2.3 Initial Register Values

When the CPU is reset, the program counter (PC) is initialized to the value stored at address H'0000 in the vector table, and the I bit in the CCR is set to 1. The other CCR bits and the general registers are not initialized. In particular, the stack pointer (R7) is not initialized. The stack pointer should be initialized by software, by the first instruction executed after a reset.

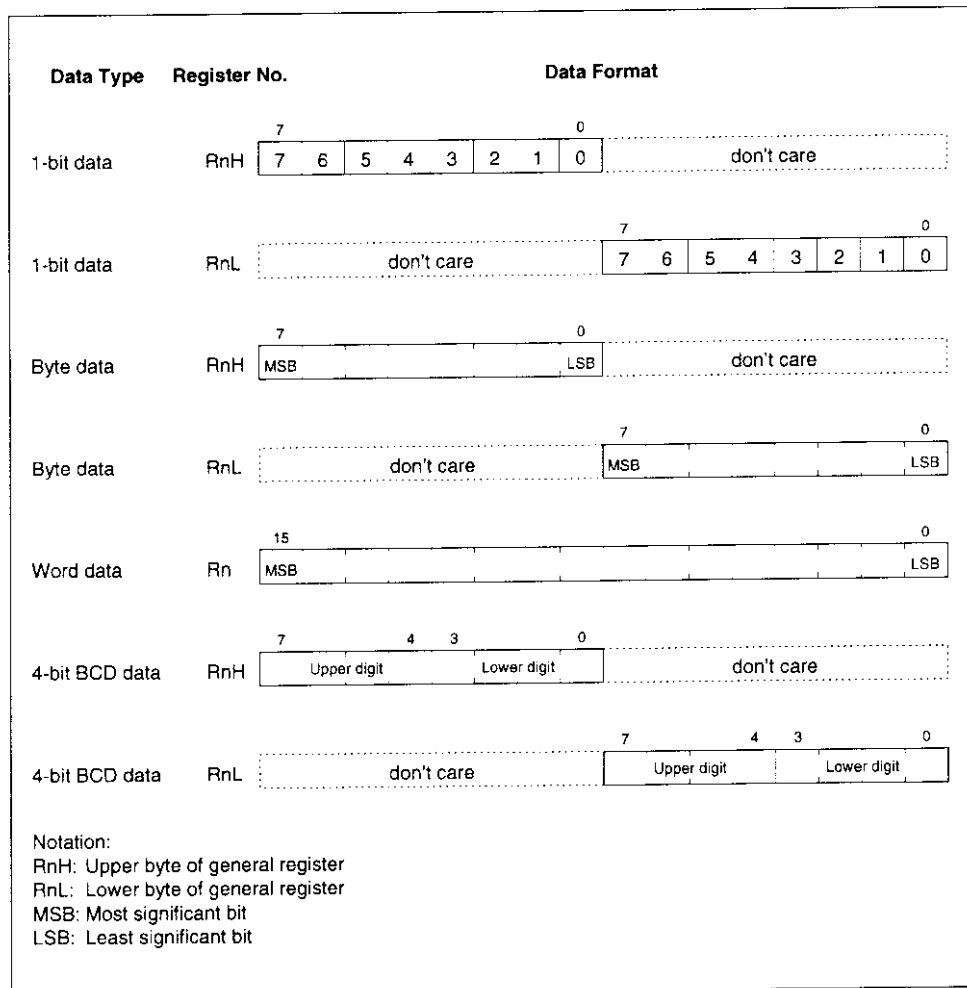
## 2.3 Data Formats

The H8/300L CPU can process 1-bit data, 4-bit (BCD) data, 8-bit (byte) data, and 16-bit (word) data.

- Bit manipulation instructions operate on 1-bit data specified as bit *n* in a byte operand ( $n = 0, 1, 2, \dots, 7$ ).
- All arithmetic and logic instructions except ADDS and SUBS can operate on byte data.
- The MOV.W, ADD.W, SUB.W, CMP.W, ADDS, SUBS, MULXU (8 bits  $\times$  8 bits), and DIVXU (16 bits  $\div$  8 bits) instructions operate on word data.
- The DAA and DAS instructions perform decimal arithmetic adjustments on byte data in packed BCD form. Each nibble of the byte is treated as a decimal digit.

### 2.3.1 Data Formats in General Registers

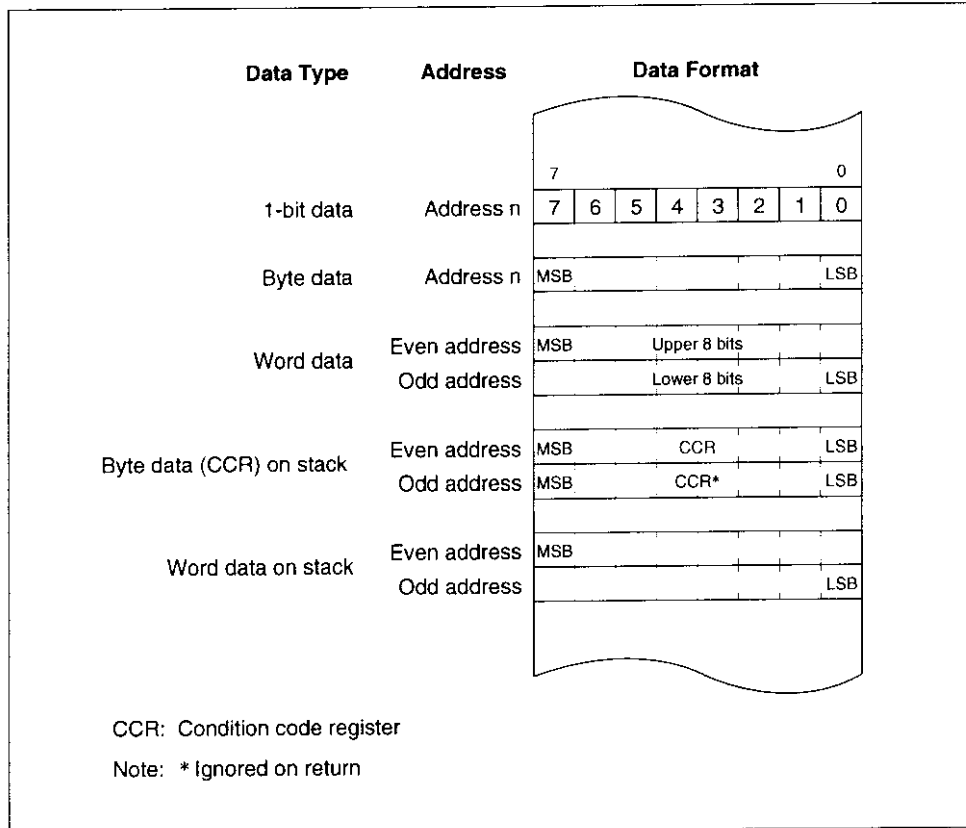
Data of all the sizes above can be stored in general registers as shown in figure 2-3.



**Figure 2-3 Register Data Formats**

### 2.3.2 Memory Data Formats

Figure 2-4 indicates the data formats in memory. For access by the H8/300L CPU, word data stored in memory must always begin at an even address. In word access the least significant bit of the address is regarded as 0. If an odd address is specified, the access is performed at the preceding even address. This rule affects the MOV.W instruction, and also applies to instruction fetching.



**Figure 2-4 Memory Data Formats**

When the stack is accessed using R7 as an address register, word access should always be performed. When the CCR is pushed on the stack, two identical copies of the CCR are pushed to make a complete word. When they are restored, the lower byte is ignored.

## 2.4 Addressing Modes

### 2.4.1 Addressing Modes

The H8/300L CPU supports the eight addressing modes listed in table 2-1. Each instruction uses a subset of these addressing modes.

**Table 2-1 Addressing Modes**

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

1. **Register Direct—Rn:** The register field of the instruction specifies an 8- or 16-bit general register containing the operand.

Only the MOV.W, ADD.W, SUB.W, CMP.W, ADDS, SUBS, MULXU (8 bits  $\times$  8 bits), and DIVXU (16 bits  $\div$  8 bits) instructions have 16-bit operands.

2. **Register Indirect—@Rn:** The register field of the instruction specifies a 16-bit general register containing the address of the operand in memory.
3. **Register Indirect with Displacement—@(d:16, Rn):** The instruction has a second word (bytes 3 and 4) containing a displacement which is added to the contents of the specified general register to obtain the operand address in memory.

This mode is used only in MOV instructions. For the MOV.W instruction, the resulting address must be even.

**4. Register Indirect with Post-Increment or Pre-Decrement—@Rn+ or @-Rn:**

- Register indirect with post-increment—@Rn+

The @Rn+ mode is used with MOV instructions that load registers from memory.

The register field of the instruction specifies a 16-bit general register containing the address of the operand. After the operand is accessed, the register is incremented by 1 for MOV.B or 2 for MOV.W. For MOV.W, the original contents of the 16-bit general register must be even.

- Register indirect with pre-decrement—@-Rn

The @-Rn mode is used with MOV instructions that store register contents to memory.

The register field of the instruction specifies a 16-bit general register which is decremented by 1 or 2 to obtain the address of the operand in memory. The register retains the decremented value. The size of the decrement is 1 for MOV.B or 2 for MOV.W. For MOV.W, the original contents of the register must be even.

**5. Absolute Address—@aa:8 or @aa:16:** The instruction specifies the absolute address of the operand in memory.

The absolute address may be 8 bits long (@aa:8) or 16 bits long (@aa:16). The MOV.B and bit manipulation instructions can use 8-bit absolute addresses. The MOV.B, MOV.W, JMP, and JSR instructions can use 16-bit absolute addresses.

For an 8-bit absolute address, the upper 8 bits are assumed to be 1 (H'FF). The address range is H'FF00 to H'FFFF (65280 to 65535).

**6. Immediate—#xx:8 or #xx:16:** The instruction contains an 8-bit operand (#xx:8) in its second byte, or a 16-bit operand (#xx:16) in its third and fourth bytes. Only MOV.W instructions can contain 16-bit immediate values.

The ADDS and SUBS instructions implicitly contain the value 1 or 2 as immediate data. Some bit manipulation instructions contain 3-bit immediate data in the second or fourth byte of the instruction, specifying a bit number.

**7. Program-Counter Relative—@(d:8, PC):** This mode is used in the Bcc and BSR instructions. An 8-bit displacement in byte 2 of the instruction code is sign-extended to 16 bits and added to the program counter contents to generate a branch destination address. The possible branching range is -126 to +128 bytes (-63 to +64 words) from the current address. The displacement should be an even number.



- 8. Memory Indirect—@@aa:8:** This mode can be used by the JMP and JSR instructions. The second byte of the instruction code specifies an 8-bit absolute address. The word located at this address contains the branch destination address.

The upper 8 bits of the absolute address are assumed to be 0 (H'00), so the address range is from H'0000 to H'00FF (0 to 255). Note that with the H8/300L Series, the lower end of the address area is also used as a vector area. See 3.3, Interrupts, for details on the vector area.

If an odd address is specified as a branch destination or as the operand address of a MOV.W instruction, the least significant bit is regarded as 0, causing word access to be performed at the address preceding the specified address. See 2.3.2, Memory Data Formats, for further information.

#### 2.4.2 Effective Address Calculation

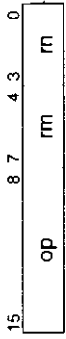
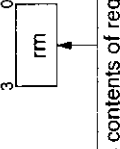
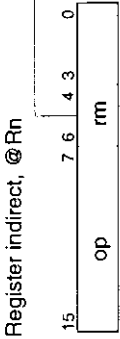
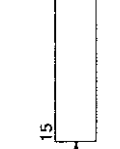
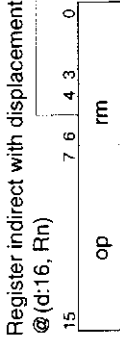
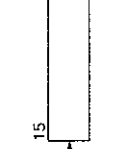
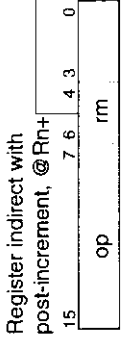

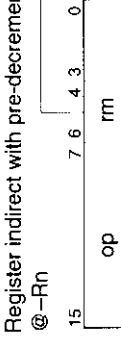
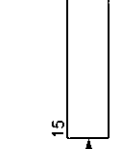
Table 2-2 shows how effective addresses are calculated in each of the addressing modes.

Arithmetic and logic instructions use register direct addressing (1). The ADD.B, ADDX, SUBX, CMP.B, AND, OR, and XOR instructions can also use immediate addressing (6).

Data transfer instructions can use all addressing modes except program-counter relative (7) and memory indirect (8).

Bit manipulation instructions use register direct (1), register indirect (2), or absolute addressing (5) to specify a byte operand, and 3-bit immediate addressing (6) to specify a bit position in that byte. The BSET, BCLR, BNOT, and BTST instructions can also use register direct addressing (1) to specify the bit position.

Table 2-2 Effective Address Calculation

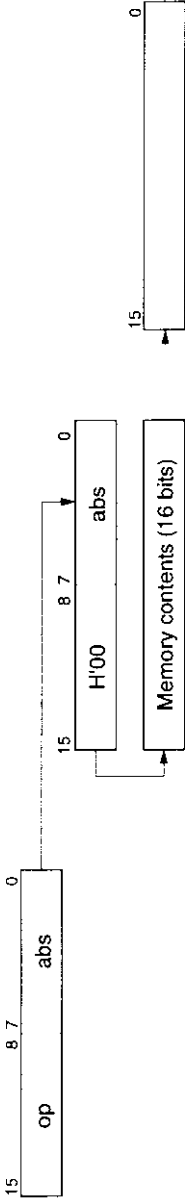
No.	Addressing Mode and Instruction Format	Effective Address Calculation Method	Effective Address (EA)
1	Register direct, Rn		
2	Register indirect, @Rn		
3	Register indirect with displacement, @(d:16, Rn)		
4	Register indirect with post-increment, @Rn+		
	Register indirect with pre-decrement, @-Rn		

Incremented or decremented by 1 if operand is byte size, and by 2 if word size

Table 2-2 Effective Address Calculation (cont)

No.	Addressing Mode and Instruction Format	Effective Address Calculation Method	Effective Address (EA)
5	Absolute address @aa:8		
	@aa:16		
6	Immediate #xx:8		Operand is 1- or 2-byte immediate data
	#xx:16		
7	Program-counter relative @(d:8, PC)		

Table 2-2 Effective Address Calculation (cont)

Addressing Mode and		Effective Address Calculation Method		Effective Address (EA)
No.	Instruction Format			
8	Memory indirect, @aa:8			

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

## 2.5 Instruction Set

The H8/300L Series can use a total of 55 instructions, which are grouped by function in table 2-3.

**Table 2-3 Instruction Set**

Function	Instructions	Number
Data transfer	MOV, PUSH* <sup>1</sup> , POP* <sup>1</sup>	1
Arithmetic operations	ADD, SUB, ADDX, SUBX, INC, DEC, ADDS, SUBS, DAA, DAS, MULXU, DIVXU, CMP, NEG	14
Logic operations	AND, OR, XOR, NOT	4
Shift	SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR	8
Bit manipulation	BSET, BCLR, BNOT, BTST, BAND, BAND, BOR, BIOR, BXOR, BIXOR, BLD, BILD, BST, BIST	14
Branch	Bcc* <sup>2</sup> , JMP, BSR, JSR, RTS	5
System control	RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP	8
Block data transfer	EEPMOV	1
		Total: 55

Notes: 1. PUSH Rn is equivalent to MOV.W Rn, @-SP.

POP Rn is equivalent to MOV.W @SP+, Rn.

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

The following sections give a concise summary of the instructions in each category, and indicate the bit patterns of their object code. The notation used is defined next.

**Notation**

Rd	General register (destination)
Rs	General register (source)
Rn	General register
(EAd), <EAd>	Destination operand
(EAs), <EAs>	Source operand
CCR	Condition code register
N	N (negative) flag of CCR
Z	Z (zero) flag of CCR
V	V (overflow) flag of CCR
C	C (carry) flag of CCR
PC	Program counter
SP	Stack pointer
#IMM	Immediate data
disp	Displacement
+	Addition
-	Subtraction
x	Multiplication
÷	Division
^	AND logical
v	OR logical
⊕	Exclusive OR logical
→	Move
~	Logical negation (logical complement)
:3	3-bit length
:8	8-bit length
:16	16-bit length
( ), < >	Contents of operand indicated by effective address

### 2.5.1 Data Transfer Instructions

Table 2-4 describes the data transfer instructions. Figure 2-5 shows their object code formats.

**Table 2-4 Data Transfer Instructions**

Instruction	Size*	Function
MOV	B/W	(EAs) → Rd, Rs → (EAd)  Moves data between two general registers or between a general register and memory, or moves immediate data to a general register.  The Rn, @Rn, @(d:16, Rn), @aa:16, #xx:16, @-Rn, and @Rn+ addressing modes are available for byte or word data. The @aa:8 addressing mode is available for byte data only.  The @-R7 and @R7+ modes require word operands. Do not specify byte size for these two modes.
POP	W	@SP+ → Rn  Pops a 16-bit general register from the stack. Equivalent to MOV.W @SP+, Rn.
PUSH	W	Rn → @-SP  Pushes a 16-bit general register onto the stack. Equivalent to MOV.W Rn, @-SP.

Notes: \* Size: Operand size  
B: Byte  
W: Word

Certain precautions are required in data access. See 2.9.1, Notes on Data Access, for details.

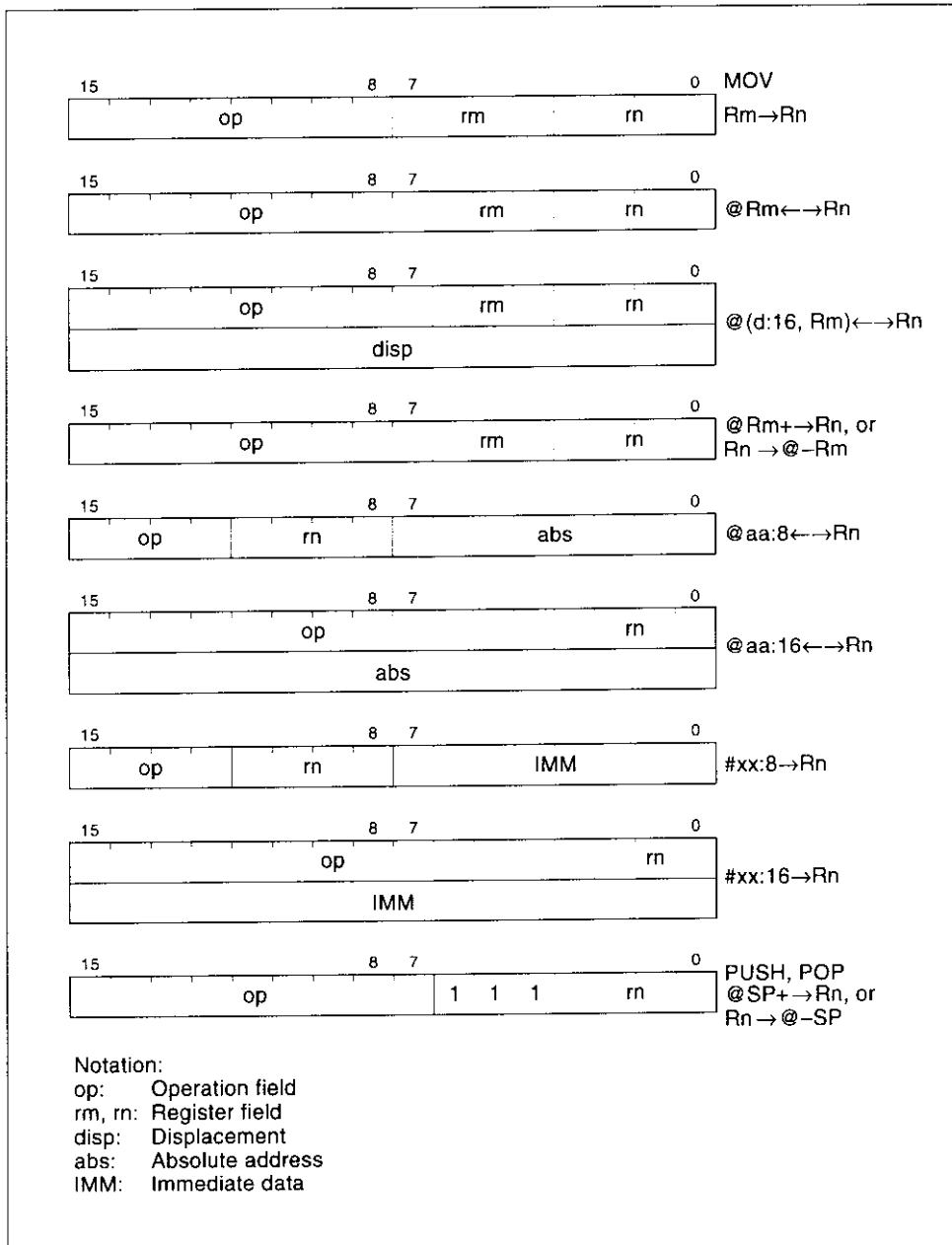


Figure 2-5 Data Transfer Instruction Codes



## 2.5.2 Arithmetic Operations

Table 2-5 describes the arithmetic instructions.

**Table 2-5 Arithmetic Instructions**

Instruction	Size*	Function
ADD SUB	B/W	$Rd \pm Rs \rightarrow Rd$ , $Rd + \#IMM \rightarrow Rd$ Performs addition or subtraction on data in two general registers, or addition on immediate data and data in a general register. Immediate data cannot be subtracted from data in a general register. Word data can be added or subtracted only when both words are in general registers.
ADDX SUBX	B	$Rd \pm Rs \pm C \rightarrow Rd$ , $Rd \pm \#IMM \pm C \rightarrow Rd$ Performs addition or subtraction with carry or borrow on byte data in two general registers, or addition or subtraction on immediate data and data in a general register.
INC DEC	B	$Rd \pm 1 \rightarrow Rd$ Increments or decrements a general register
ADDS SUBS	W	$Rd \pm 1 \rightarrow Rd$ , $Rd \pm 2 \rightarrow Rd$ Adds or subtracts 1 or 2 to or from a general register
DAA DAS	B	$Rd$ decimal adjust $\rightarrow Rd$ Decimal-adjusts (adjusts to packed BCD) an addition or subtraction result in a general register by referring to the CCR
MULXU	B	$Rd \times Rs \rightarrow Rd$ Performs 8-bit $\times$ 8-bit unsigned multiplication on data in two general registers, providing a 16-bit result
DIVXU	B	$Rd \div Rs \rightarrow Rd$ Performs 16-bit $\div$ 8-bit unsigned division on data in two general registers, providing an 8-bit quotient and 8-bit remainder
CMP	B/W	$Rd - Rs$ , $Rd - \#IMM$ Compares data in a general register with data in another general register or with immediate data, and indicates the result in the CCR. Word data can be compared only between two general registers.
NEG	B	$0 - Rd \rightarrow Rd$ Obtains the two's complement (arithmetic complement) of data in a general register

Notes: \* Size: Operand size  
 B: Byte  
 W: Word

### 2.5.3 Logic Operations

Table 2-6 describes the four instructions that perform logic operations.

**Table 2-6 Logic Operation Instructions**

Instruction	Size*	Function
AND	B	$Rd \wedge Rs \rightarrow Rd$ , $Rd \wedge \#IMM \rightarrow Rd$ Performs a logical AND operation on a general register and another general register or immediate data
OR	B	$Rd \vee Rs \rightarrow Rd$ , $Rd \vee \#IMM \rightarrow Rd$ Performs a logical OR operation on a general register and another general register or immediate data
XOR	B	$Rd \oplus Rs \rightarrow Rd$ , $Rd \oplus \#IMM \rightarrow Rd$ Performs a logical exclusive OR operation on a general register and another general register or immediate data
NOT	B	$\sim Rd \rightarrow Rd$ Obtains the one's complement (logical complement) of general register contents

Notes: \* Size: Operand size  
B: Byte

### 2.5.4 Shift Operations

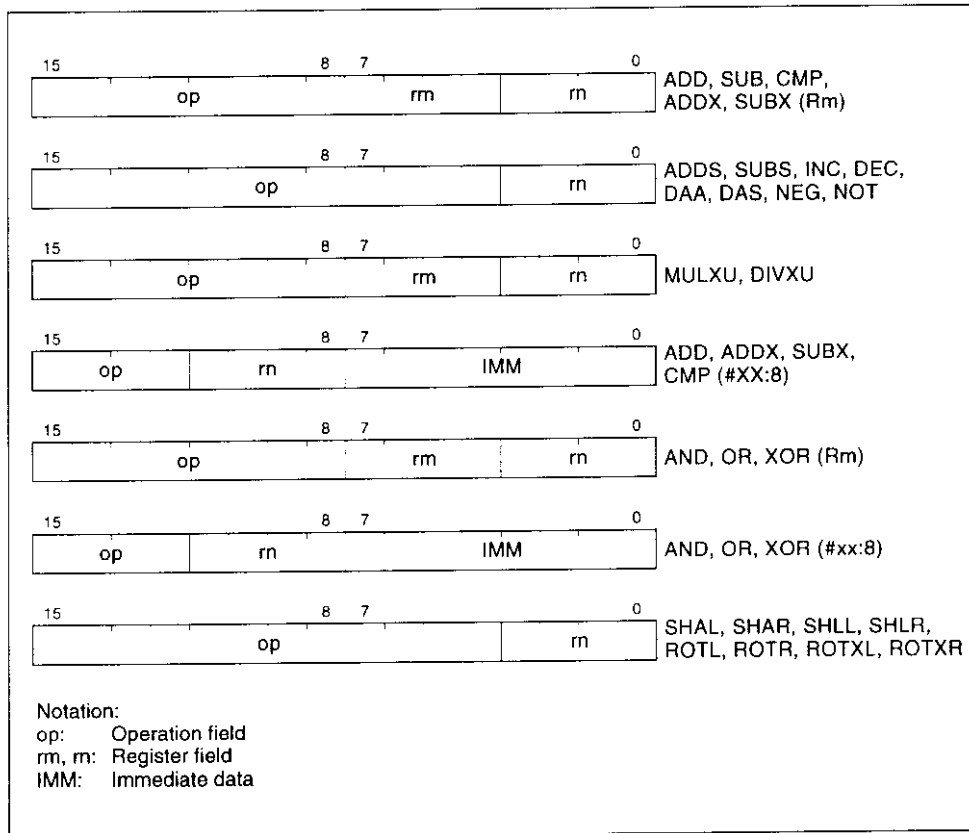
Table 2-7 describes the eight shift instructions.

**Table 2-7 Shift Instructions**

Instruction	Size*	Function
SHAL SHAR	B	$Rd \text{ shift} \rightarrow Rd$ Performs an arithmetic shift operation on general register contents
SHLL SHLR	B	$Rd \text{ shift} \rightarrow Rd$ Performs a logical shift operation on general register contents
ROTL ROTR	B	$Rd \text{ rotate} \rightarrow Rd$ Rotates general register contents
ROTXL ROTXR	B	$Rd \text{ rotate through carry} \rightarrow Rd$ Rotates general register contents through the C (carry) bit

Notes: \* Size: Operand size  
B: Byte

Figure 2-6 shows the instruction code format of arithmetic, logic, and shift instructions.



**Figure 2-6 Arithmetic, Logic, and Shift Instruction Codes**

### 2.5.5 Bit Manipulations

Table 2-8 describes the bit-manipulation instructions. Figure 2-7 shows their object code formats.

**Table 2-8 Bit-Manipulation Instructions**

Instruction	Size*	Function
BSET	B	$1 \rightarrow \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle$ Sets a specified bit in a general register or memory to 1. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BCLR	B	$0 \rightarrow \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle$ Clears a specified bit in a general register or memory to 0. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BNOT	B	$\sim \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle \rightarrow \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle$ Inverts a specified bit in a general register or memory. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BTST	B	$\sim \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle \rightarrow Z$ Tests a specified bit in a general register or memory and sets or clears the Z flag accordingly. The bit number is specified by 3-bit immediate data or the lower three bits of a general register.
BAND	B	$C \wedge \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle \rightarrow C$ ANDs the C flag with a specified bit in a general register or memory, and stores the result in the C flag.
BIAND	B	$C \wedge [\sim \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle] \rightarrow C$ ANDs the C flag with the inverse of a specified bit in a general register or memory, and stores the result in the C flag. The bit number is specified by 3-bit immediate data.
BOR	B	$C \vee \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle \rightarrow C$ ORs the C flag with a specified bit in a general register or memory, and stores the result in the C flag.
BIOR	B	$C \vee [\sim \langle \text{bit-No.} \rangle \text{ of } \langle \text{EAd} \rangle] \rightarrow C$ ORs the C flag with the inverse of a specified bit in a general register or memory, and stores the result in the C flag. The bit number is specified by 3-bit immediate data.

Notes: \* Size: Operand size  
B: Byte

**Table 2-8 Bit-Manipulation Instructions (cont)**

<b>Instruction</b>	<b>Size*</b>	<b>Function</b>
BXOR	B	$C \oplus (\text{<bit-No.> of <EAd>}) \rightarrow C$ XORs the C flag with a specified bit in a general register or memory, and stores the result in the C flag.
BIXOR	B	$C \oplus [\sim(\text{<bit-No.> of <EAd>})] \rightarrow C$ XORs the C flag with the inverse of a specified bit in a general register or memory, and stores the result in the C flag. The bit number is specified by 3-bit immediate data.
BLD	B	$(\text{<bit-No.> of <EAd>}) \rightarrow C$ Copies a specified bit in a general register or memory to the C flag.
BILD	B	$\sim (\text{<bit-No.> of <EAd>}) \rightarrow C$ Copies the inverse of a specified bit in a general register or memory to the C flag. The bit number is specified by 3-bit immediate data.
BST	B	$C \rightarrow (\text{<bit-No.> of <EAd>})$ Copies the C flag to a specified bit in a general register or memory.
BIST	B	$\sim C \rightarrow (\text{<bit-No.> of <EAd>})$ Copies the inverse of the C flag to a specified bit in a general register or memory. The bit number is specified by 3-bit immediate data.

Notes: \* Size: Operand size  
B: Byte

Certain precautions are required in bit manipulation. See 2.9.2, Notes on Bit Manipulation, for details.

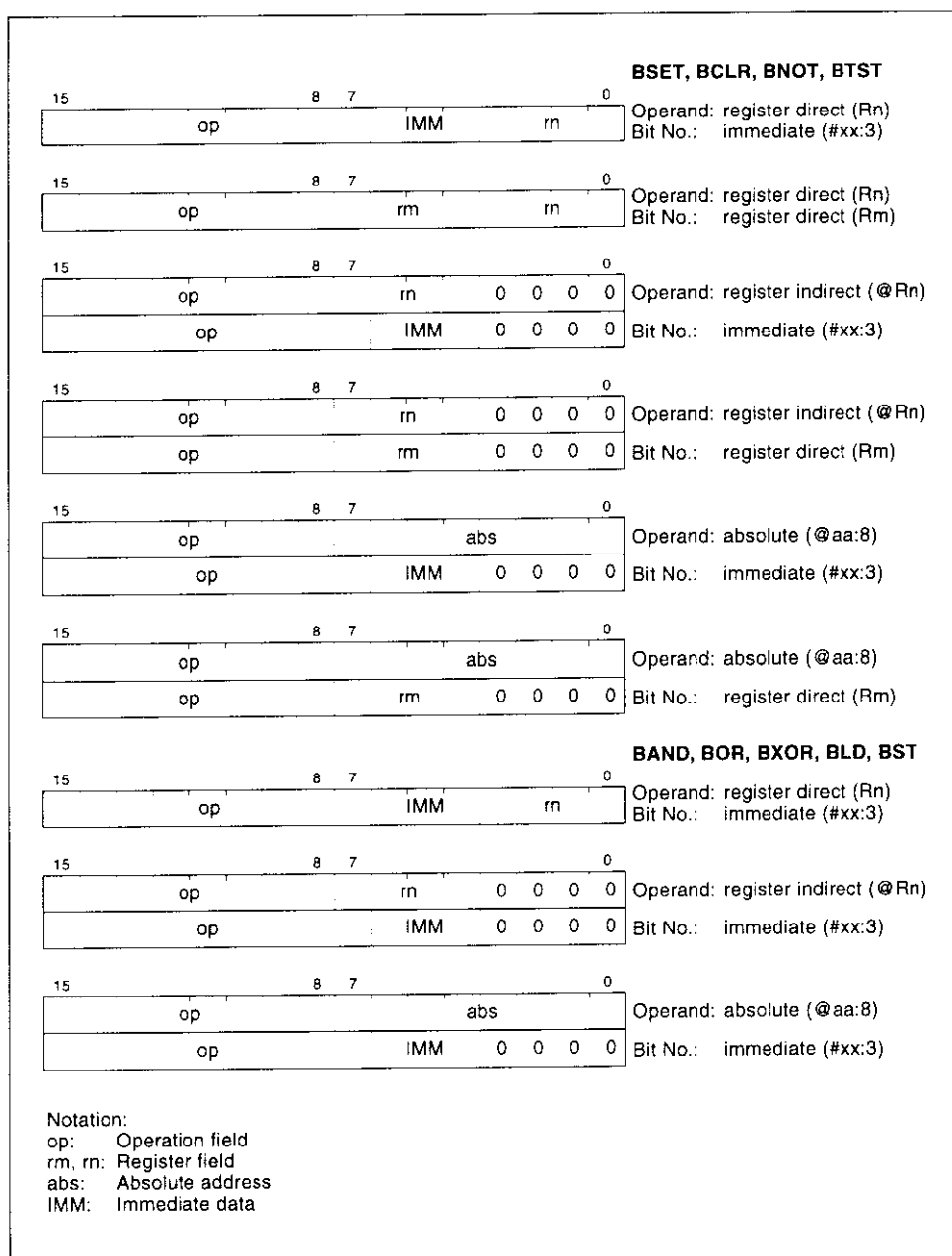
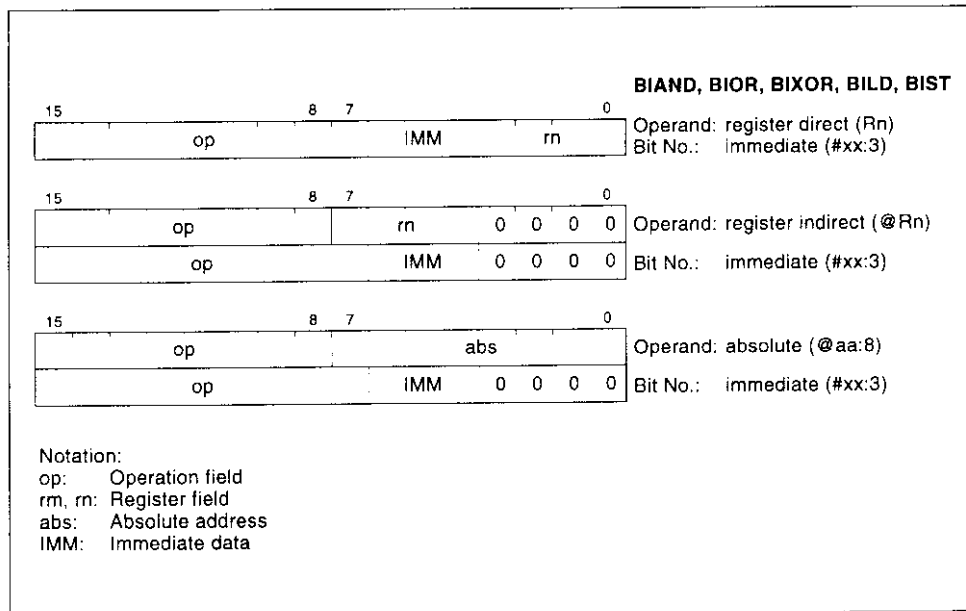


Figure 2-7 Bit Manipulation Instruction Codes



**Figure 2-7 Bit Manipulation Instruction Codes (cont)**

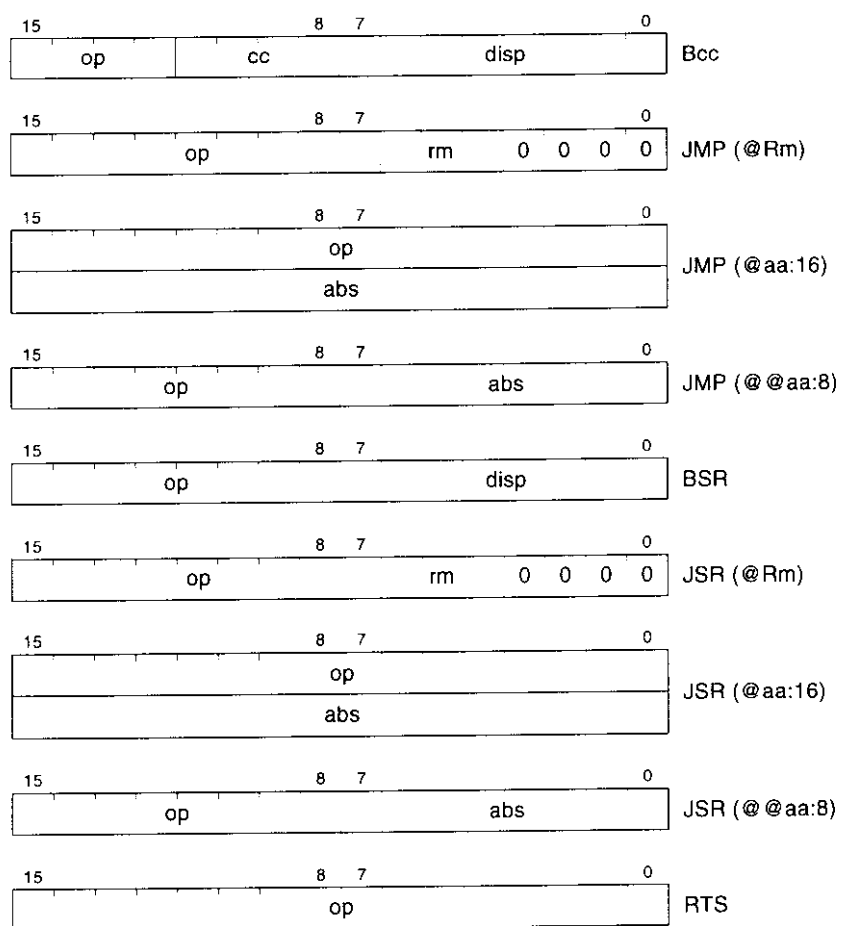
### 2.5.6 Branching Instructions

Table 2-9 describes the branching instructions. Figure 2-8 shows their object code formats.

**Table 2-9 Branching Instructions**

Instruction	Size	Function																																																			
Bcc	—	Branches to the designated address if condition cc is true. The branching conditions are given below.																																																			
		<table> <tr> <th>Mnemonic</th><th>Description</th><th>Condition</th></tr> <tr> <td>BRA (BT)</td><td>Always (true)</td><td>Always</td></tr> <tr> <td>BRN (BF)</td><td>Never (false)</td><td>Never</td></tr> <tr> <td>BHI</td><td>High</td><td><math>C \vee Z = 0</math></td></tr> <tr> <td>BLS</td><td>Low or same</td><td><math>C \vee Z = 1</math></td></tr> <tr> <td>BCC (BHS)</td><td>Carry clear (high or same)</td><td><math>C = 0</math></td></tr> <tr> <td>BCS (BLO)</td><td>Carry set (low)</td><td><math>C = 1</math></td></tr> <tr> <td>BNE</td><td>Not equal</td><td><math>Z = 0</math></td></tr> <tr> <td>BEQ</td><td>Equal</td><td><math>Z = 1</math></td></tr> <tr> <td>BVC</td><td>Overflow clear</td><td><math>V = 0</math></td></tr> <tr> <td>BVS</td><td>Overflow set</td><td><math>V = 1</math></td></tr> <tr> <td>BPL</td><td>Plus</td><td><math>N = 0</math></td></tr> <tr> <td>BMI</td><td>Minus</td><td><math>N = 1</math></td></tr> <tr> <td>BGE</td><td>Greater or equal</td><td><math>N \oplus V = 0</math></td></tr> <tr> <td>BLT</td><td>Less than</td><td><math>N \oplus V = 1</math></td></tr> <tr> <td>BGT</td><td>Greater than</td><td><math>Z \vee (N \oplus V) = 0</math></td></tr> <tr> <td>BLE</td><td>Less or equal</td><td><math>Z \vee (N \oplus V) = 1</math></td></tr> </table>	Mnemonic	Description	Condition	BRA (BT)	Always (true)	Always	BRN (BF)	Never (false)	Never	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	$Z \vee (N \oplus V) = 1$
Mnemonic	Description	Condition																																																			
BRA (BT)	Always (true)	Always																																																			
BRN (BF)	Never (false)	Never																																																			
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																																																			
BSR	—	Branches to a subroutine at a specified address																																																			
JSR	—	Branches to a subroutine at a specified address																																																			
RTS	—	Returns from a subroutine																																																			





Notation:  
 op: Operation field  
 cc: Condition field  
 rm: Register field  
 disp: Displacement  
 abs: Absolute address

**Figure 2-8 Branching Instruction Codes**

### 2.5.7 System Control Instructions

Table 2-10 describes the system control instructions. Figure 2-9 shows their object code formats.

**Table 2-10 System Control Instructions**

Instruction	Size*	Function
RTE	—	Returns from an exception-handling routine
SLEEP	—	Causes a transition from active mode to a power-down mode. See section 5, Power-Down Modes, for details.
LDC	B	$R_s \rightarrow CCR$ , $\#IMM \rightarrow CCR$ Moves immediate data or general register contents to the condition code register
STC	B	$CCR \rightarrow R_d$ Copies the condition code register to a specified general register
ANDC	B	$CCR \wedge \#IMM \rightarrow CCR$ Logically ANDs the condition code register with immediate data
ORC	B	$CCR \vee \#IMM \rightarrow CCR$ Logically ORs the condition code register with immediate data
XORC	B	$CCR \oplus \#IMM \rightarrow CCR$ Logically exclusive-ORs the condition code register with immediate data
NOP	—	$PC + 2 \rightarrow PC$ Only increments the program counter

Notes: \* Size: Operand size  
B: Byte

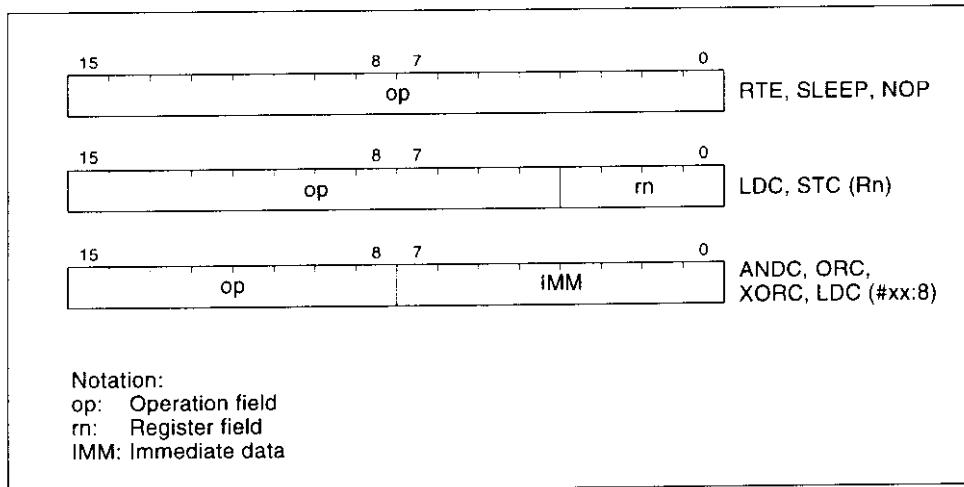


Figure 2-9 System Control Instruction Codes

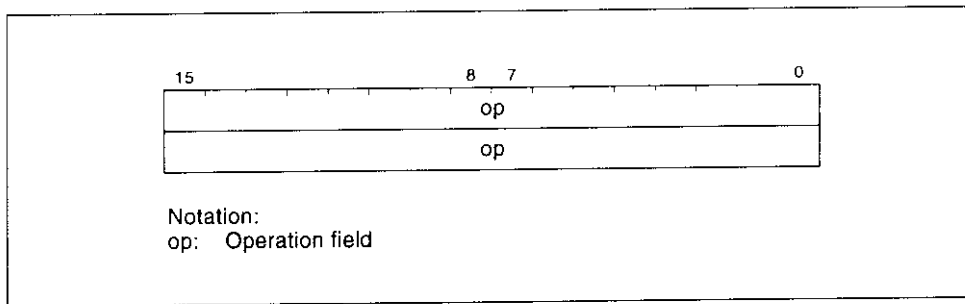
### 2.5.8 Block Data Transfer Instruction

Table 2-11 describes the block data transfer instruction. Figure 2-10 shows its object code format.

Table 2-11 Block Data Transfer Instruction

Instruction	Size	Function
EEPMOV	—	<p>If R4L <math>\neq</math> 0 then</p> <p>repeat @R5+ <math>\rightarrow</math> @R6+ R4L - 1 <math>\rightarrow</math> R4L</p> <p>until R4L = 0</p> <p>else next;</p> <p>Moves a data block according to parameters set in general registers R4L, R5, and R6.</p> <p>R4L: Size of block (bytes)</p> <p>R5: Starting source address</p> <p>R6: Starting destination address</p> <p>Execution of the next instruction starts as soon as the block transfer is completed.</p>

Certain precautions are required in using the EEPMOV instruction. See 2.9.3, Notes on Use of the EEPMOV Instruction, for details.



**Figure 2-10 Block Data Transfer Instruction Code**

## 2.6 Basic Operational Timing

CPU operation is synchronized by a system clock ( $\phi$ ) or a subclock ( $\phi_{SUB}$ ). For details on these clock signals see section 4, Clock Pulse Generators. The period from a rising edge of  $\phi$  or  $\phi_{SUB}$  to the next rising edge is called one state. A bus cycle consists of two states or three states. The cycle differs depending on whether access is to on-chip memory or to on-chip peripheral modules.

### 2.6.1 Access to On-Chip Memory (RAM, ROM)

Access to on-chip memory takes place in two states. The data bus width is 16 bits, allowing access in byte or word size. Figure 2-11 shows the on-chip memory access cycle.

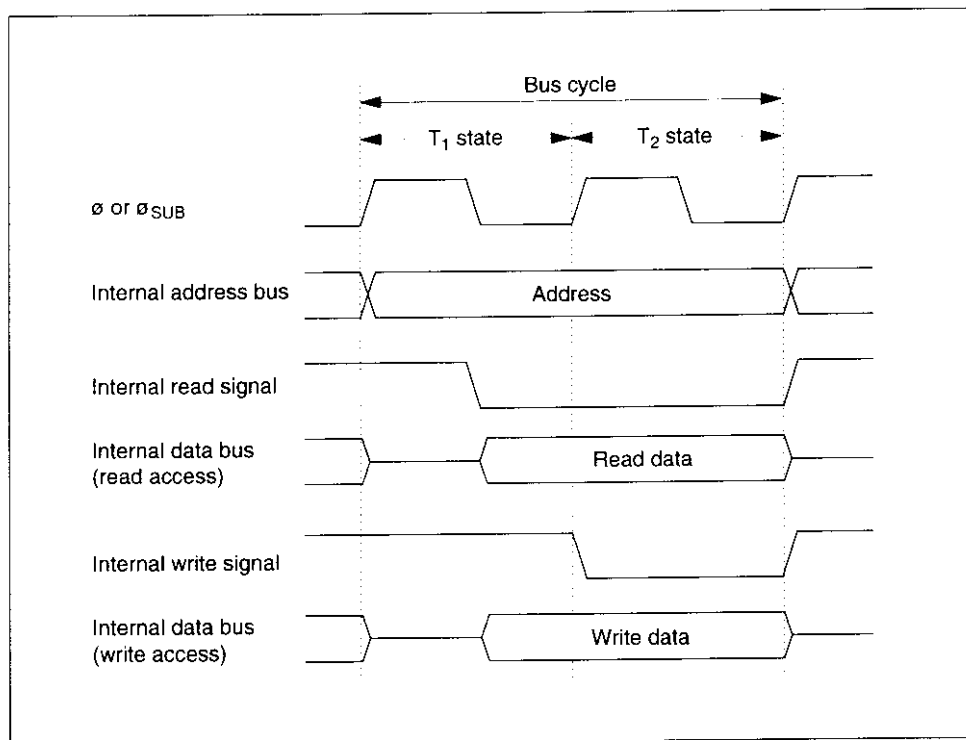


Figure 2-11 On-Chip Memory Access Cycle

### 2.6.2 Access to On-Chip Peripheral Modules

On-chip peripheral modules are accessed in two states or three states. The data bus width is 8 bits, so access is by byte size only. This means that for accessing word data, two instructions must be used. Figures 2-12 and 2-13 show the on-chip peripheral module access cycle.

Two-state access to on-chip peripheral modules

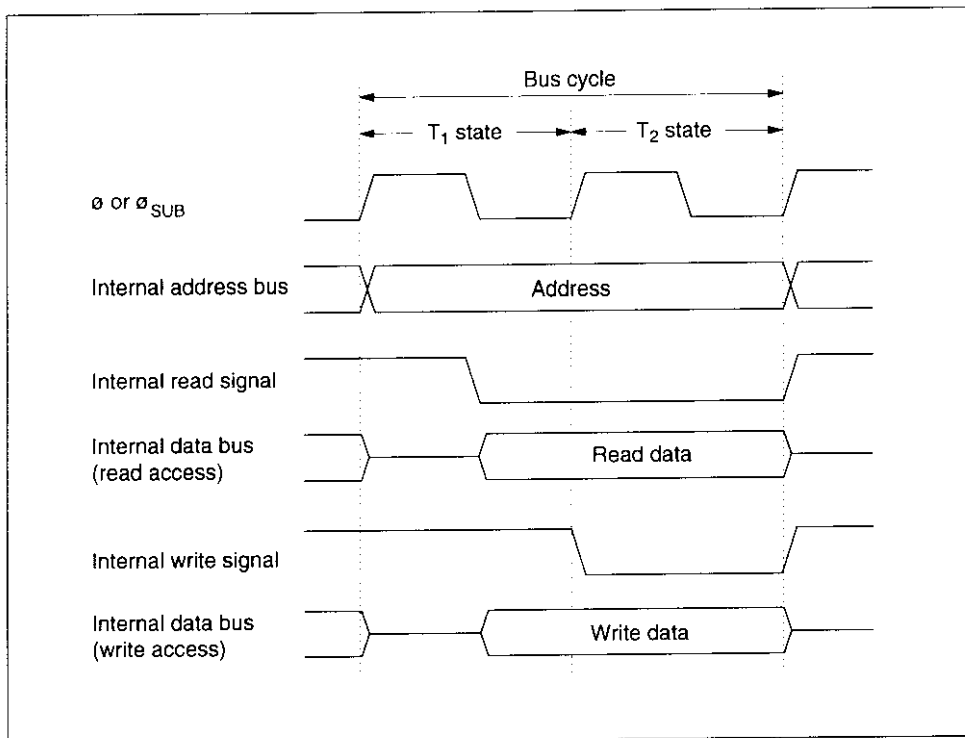
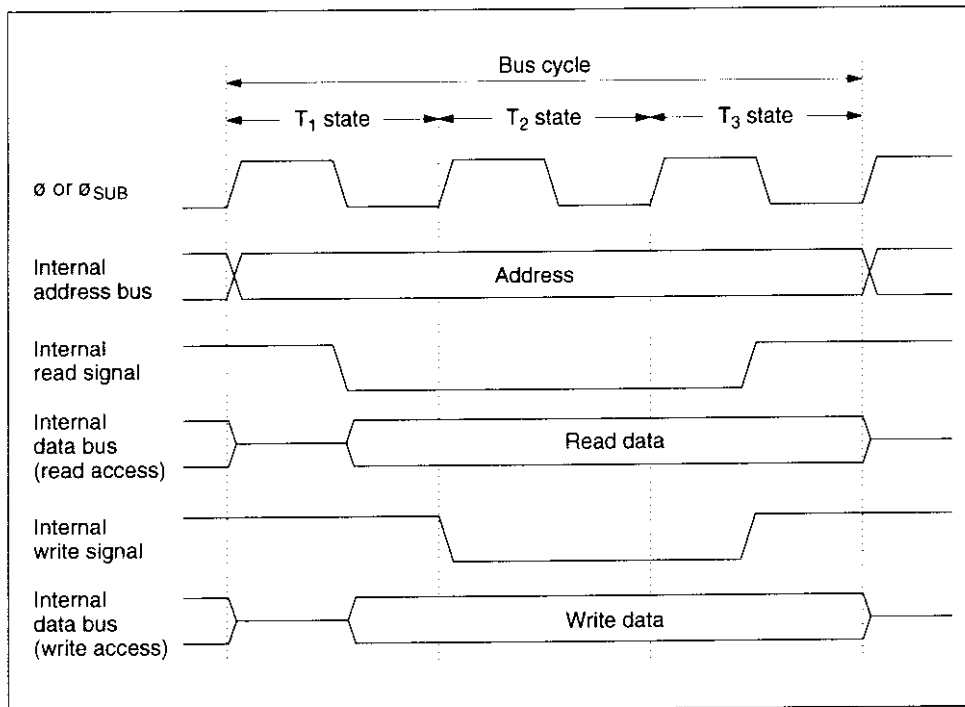


Figure 2-12 On-Chip Peripheral Module Access Cycle (2-State Access)

### Three-state access to on-chip peripheral modules



**Figure 2-13 On-Chip Peripheral Module Access Cycle (3-State Access)**

## 2.7 CPU States

### 2.7.1 Overview

There are four CPU states: the reset state, program execution state, program halt state, and exception-handling state. The program execution state includes active (high-speed or medium-speed) mode and subactive mode. In the program halt state there are a sleep (high-speed or medium-speed) mode, standby mode, watch mode, and sub-sleep mode. These states are shown in figure 2-14. Figure 2-15 shows the state transitions.

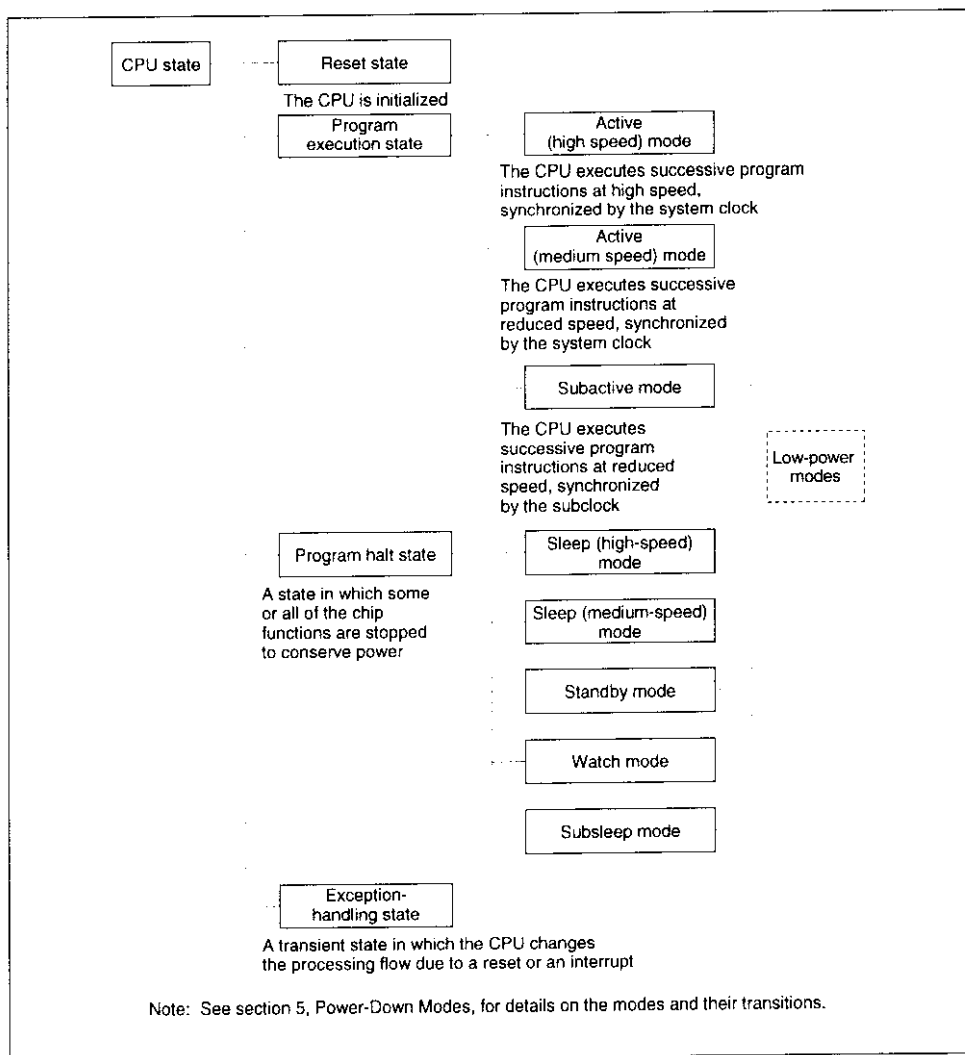
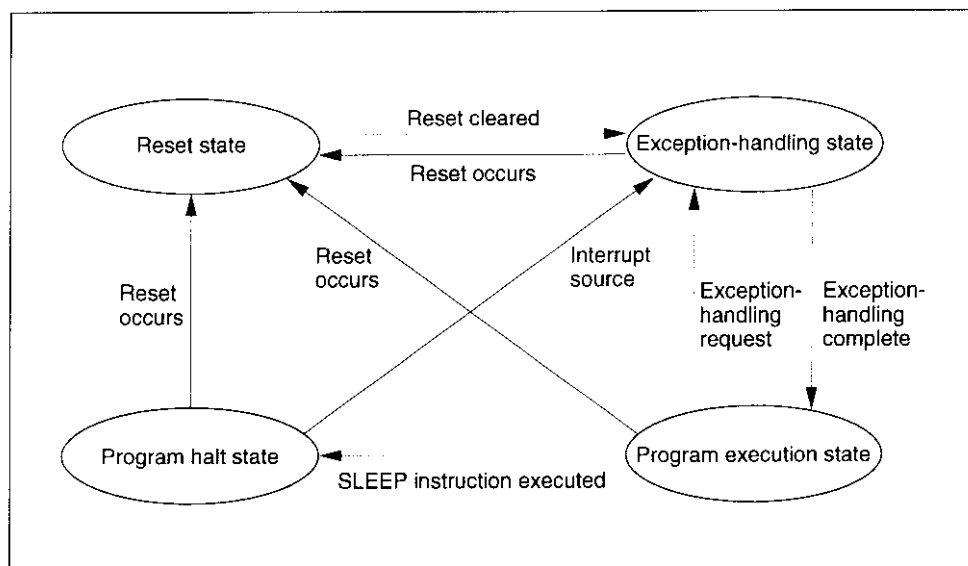


Figure 2-14 CPU Operation States





**Figure 2-15 State Transitions**

### 2.7.2 Program Execution State

In the program execution state the CPU executes program instructions in sequence.

There are three modes in this state, two active modes (high speed and medium speed) and one subactive mode. Operation is synchronized with the system clock in active mode (high speed and medium speed), and with the subclock in subactive mode. See section 5, Power-Down Modes for details on these modes.

### 2.7.3 Program Halt State

In the program halt state there are four modes: two sleep modes (high speed and medium speed), standby mode, watch mode, and subsleep mode. See section 5, Power-Down Modes for details on these modes.

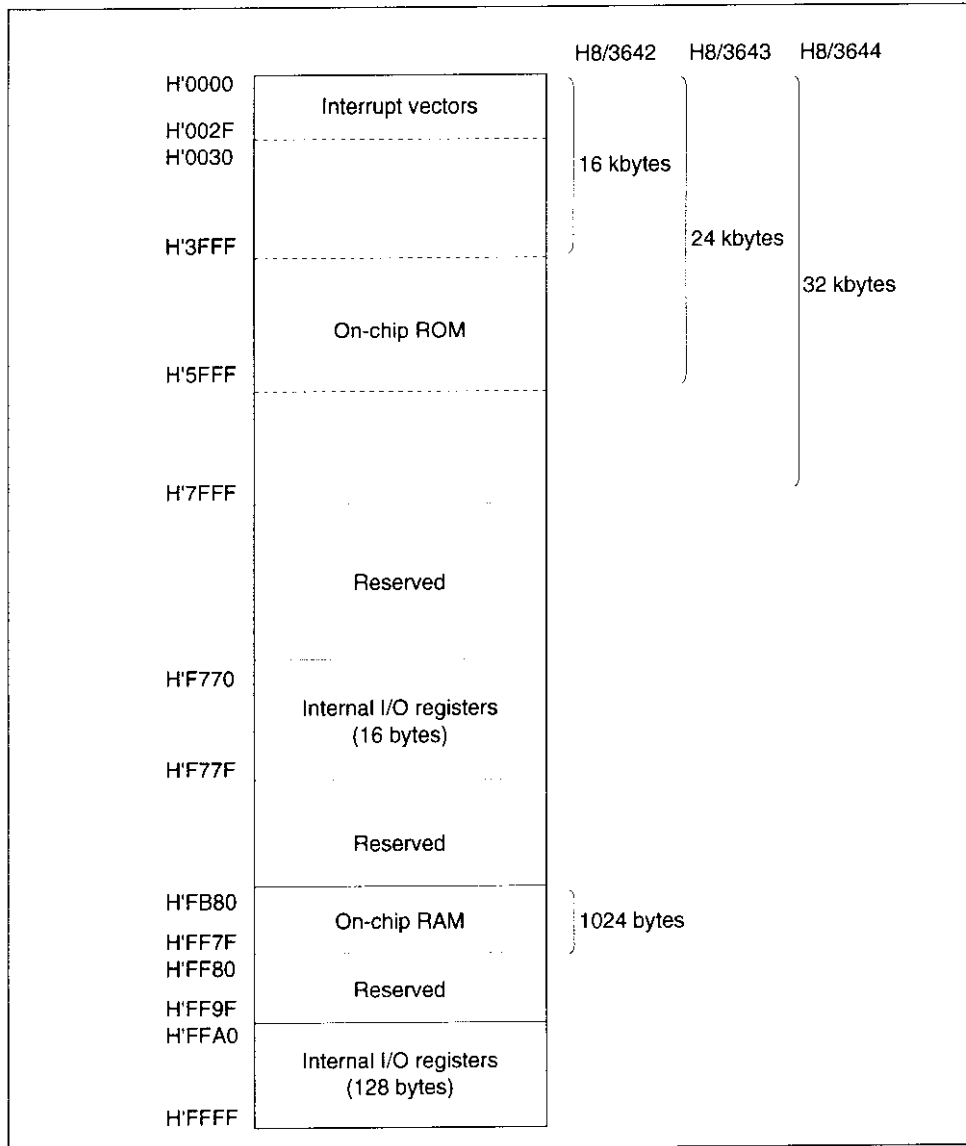
### 2.7.4 Exception-Handling State

The exception-handling state is a transient state occurring when exception handling is started by a reset or interrupt and the CPU changes its normal processing flow. In exception handling caused by an interrupt, SP (R7) is referenced and the PC and CCR values are saved on the stack.

For details on interrupt handling, see section 3.3, Interrupts.

## 2.8 Memory Map

Figure 2-16 shows a memory map of the H8/3644 Series.



**Figure 2-16 H8/3644 Series Memory Map**

## 2.9 Application Notes

### 2.9.1 Notes on Data Access

1. The address space of the H8/300L CPU includes empty areas in addition to the RAM, registers, and ROM areas available to the user. If these empty areas are mistakenly accessed by an application program, the following results will occur.

Data transfer from CPU to empty area:

The transferred data will be lost. This action may also cause the CPU to misoperate.

Data transfer from empty area to CPU:

Unpredictable data is transferred.

2. Internal data transfer to or from on-chip modules other than the ROM and RAM areas makes use of an 8-bit data width. If word access is attempted to these areas, the following results will occur.

Word access from CPU to I/O register area:

Upper byte: Will be written to I/O register.

Lower byte: Transferred data will be lost.

Word access from I/O register to CPU:

Upper byte: Will be written to upper part of CPU register.

Lower byte: Unpredictable data will be written to lower part of CPU register.

Byte size instructions should therefore be used when transferring data to or from I/O registers other than the on-chip ROM and RAM areas. Figure 2-17 shows the data size and number of states in which on-chip peripheral modules can be accessed.

			Access		States
			Word	Byte	
H'0000	Interrupt vector area (48 bytes)				
H'002F					
H'0030					
	On-chip ROM		○	○	2
H'7FFF	Reserved		—	—	—
H'F770	Internal I/O registers (16 bytes)		×	○	3*
H'F77F	Reserved		—	—	—
H'FB80	On-chip RAM	1,024 bytes	○	○	2
H'FF7F	Reserved		×	—	—
H'FF80					
H'FF9F					
H'FFA0	Internal I/O registers (96 bytes)		×	○	2 or 3*2
H'FFFF					

Notes: The H8/3644 is shown as an example.

\* Internal I/O registers in areas assigned to timer X (H'F770 to H'F77F), SCI3 (H'FFA8 to H'FFAD), and timer V (H'FFB8 to H'FFBD) are accessed in three states.

**Figure 2-17 Data Size and Number of States for Access to and from On-Chip Peripheral Modules**

## 2.9.2 Notes on Bit Manipulation

The BSET, BCLR, BNOT, BST, and BIST instructions read one byte of data, modify the data, then write the data byte again. Special care is required when using these instructions in cases where two registers are assigned to the same address, in the case of registers that include write-only bits, and when the instruction accesses an I/O port.

Order of Operation	Operation
1 Read	Read byte data at the designated address
2 Modify	Modify a designated bit in the read data
3 Write	Write the altered byte data to the designated address

1. Bit manipulation in two registers assigned to the same address

Example 1: timer load register and timer counter

Figure 2-18 shows an example in which two timer registers share the same address. When a bit manipulation instruction accesses the timer load register and timer counter of a reloadable timer, since these two registers share the same address, the following operations take place.

Order of Operation	Operation
1 Read	Timer counter data is read (one byte)
2 Modify	The CPU modifies (sets or resets) the bit designated in the instruction
3 Write	The altered byte data is written to the timer load register

The timer counter is counting, so the value read is not necessarily the same as the value in the timer load register. As a result, bits other than the intended bit in the timer load register may be modified to the timer counter value.

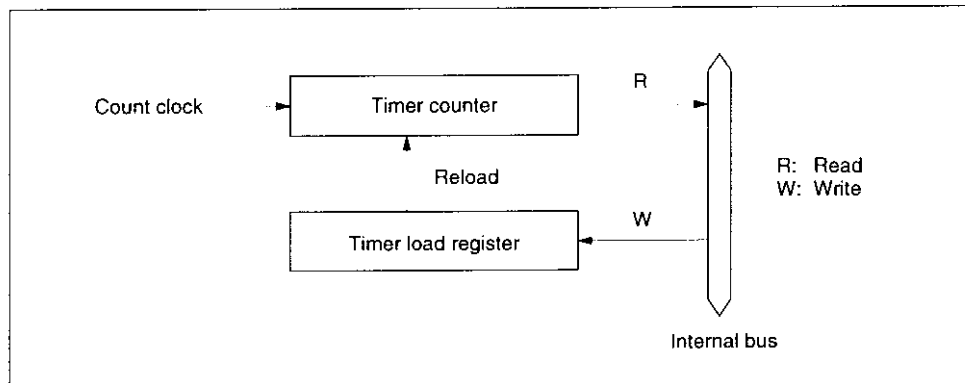


Figure 2-18 Timer Configuration Example

Example 2: BSET instruction executed designating port 3

P3<sub>7</sub> and P3<sub>6</sub> are designated as input pins, with a low-level signal input at P3<sub>7</sub> and a high-level signal at P3<sub>6</sub>. The remaining pins, P3<sub>5</sub> to P3<sub>0</sub>, are output pins and output low-level signals. In this example, the BSET instruction is used to change pin P3<sub>0</sub> to high-level output.

[A: Prior to executing BSET]

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	Low level
PCR3	0	0	1	1	1	1	1	1
PDR3	1	0	0	0	0	0	0	0

[B: BSET instruction executed]

BSET #0, @PDR3

The BSET instruction is executed designating port 3.

[C: After executing BSET]

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	High level
PCR3	0	0	1	1	1	1	1	1
PDR3	0	1	0	0	0	0	0	1

[D: Explanation of how BSET operates]

When the BSET instruction is executed, first the CPU reads port 3.

Since P3<sub>7</sub> and P3<sub>6</sub> are input pins, the CPU reads the pin states (low-level and high-level input). P3<sub>5</sub> to P3<sub>0</sub> are output pins, so the CPU reads the value in PDR3. In this example PDR3 has a value of H'80, but the value read by the CPU is H'40.

Next, the CPU sets bit 0 of the read data to 1, changing the PDR3 data to H'41. Finally, the CPU writes this value (H'41) to PDR3, completing execution of BSET.

As a result of this operation, bit 0 in PDR3 becomes 1, and P3<sub>0</sub> outputs a high-level signal. However, bits 7 and 6 of PDR3 end up with different values.

To avoid this problem, store a copy of the PDR3 data in a work area in memory. Perform the bit manipulation on the data in the work area, then write this data to PDR3.

[A: Prior to executing BSET]

MOV. B #80, R0L
MOV. B R0L, @RAM0
MOV. B R0L, @PDR3

The PDR3 value (H'80) is written to a work area in memory (RAM0) as well as to PDR3.

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	Low level
PCR3	0	0	1	1	1	1	1	1
PDR3	1	0	0	0	0	0	0	0
RAM0	1	0	0	0	0	0	0	0

[B: BSET instruction executed]

BSET	#0	,	@RAM0
------	----	---	-------

The BSET instruction is executed designating the PDR3 work area (RAM0).

[C: After executing BSET]

```
MOV. B  @RAM0, R0L
MOV. B  R0L,  @PDR3
```

The work area (RAM0) value is written to PDR3.

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	High level
PCR3	0	0	1	1	1	1	1	1
PDR3	1	0	0	0	0	0	0	1
RAM0	1	0	0	0	0	0	0	1

## 2. Bit manipulation in a register containing a write-only bit

Example 3: BCLR instruction executed designating port 3 control register PCR3

As in the examples above, P3<sub>7</sub> and P3<sub>6</sub> are input pins, with a low-level signal input at P3<sub>7</sub> and a high-level signal at P3<sub>6</sub>. The remaining pins, P3<sub>5</sub> to P3<sub>0</sub>, are output pins that output low-level signals. In this example, the BCLR instruction is used to change pin P3<sub>0</sub> to an input port. It is assumed that a high-level signal will be input to this input pin.

[A: Prior to executing BCLR]

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	Low level
PCR3	0	0	1	1	1	1	1	1
PDR3	1	0	0	0	0	0	0	0

[B: BCLR instruction executed]

```
BCLR  #0 ,  @PCR3
```

The BCLR instruction is executed designating PCR3.



[C: After executing BCLR]

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Output	Output	Output	Output	Output	Output	Output	Input
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	High level
PCR3	1	1	1	1	1	1	1	0
PDR3	1	0	0	0	0	0	0	0

[D: Explanation of how BCLR operates]

When the BCLR instruction is executed, first the CPU reads PCR3. Since PCR3 is a write-only register, the CPU reads a value of H'FF, even though the PCR3 value is actually H'3F.

Next, the CPU clears bit 0 in the read data to 0, changing the data to H'FE. Finally, this value (H'FE) is written to PCR3 and BCLR instruction execution ends.

As a result of this operation, bit 0 in PCR3 becomes 0, making P3<sub>0</sub> an input port. However, bits 7 and 6 in PCR3 change to 1, so that P3<sub>7</sub> and P3<sub>6</sub> change from input pins to output pins.

To avoid this problem, store a copy of the PCR3 data in a work area in memory. Perform the bit manipulation on the data in the work area, then write this data to PCR3.

[A: Prior to executing BCLR]

```
MOV. B  #3F,  R0L
MOV. B  R0L,  @RAM0
MOV. B  R0L,  @PCR3
```

The PCR3 value (H'3F) is written to a work area in memory (RAM0) as well as to PCR3.

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	Low level
PCR3	0	0	1	1	1	1	1	1
PDR3	1	0	0	0	0	0	0	0
RAM0	0	0	1	1	1	1	1	1

[B: BCLR instruction executed]

BCLR	#0	,	@RAM0
------	----	---	-------

The BCLR instruction is executed designating the PCR3 work area (RAM0).

[C: After executing BCLR]

MOV. B	@RAM0, R0L
MOV. B	R0L, @PCR3

The work area (RAM0) value is written to PCR3.

	P3 <sub>7</sub>	P3 <sub>6</sub>	P3 <sub>5</sub>	P3 <sub>4</sub>	P3 <sub>3</sub>	P3 <sub>2</sub>	P3 <sub>1</sub>	P3 <sub>0</sub>
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	High level
PCR3	0	0	1	1	1	1	1	0
PDR3	1	0	0	0	0	0	0	0
RAM0	0	0	1	1	1	1	1	0

Table 2-12 lists the pairs of registers that share identical addresses. Table 2-13 lists the registers that contain write-only bits.

**Table 2-12 Registers with Shared Addresses**

Register Name	Abbreviation	Address
Output compare register AH and output compare register BH (timer X)	OCRAH/OCRBH	H'F774
Output compare register AL and output compare register BL (timer X)	OCRAL/OCRBL	H'F775
Timer counter B1 and timer load register B1	(timer B1) TCB1/TLB1	H'FFB3
Port data register 1*	PDR1	H'FFD4
Port data register 2*	PDR2	H'FFD5
Port data register 3*	PDR3	H'FFD6
Port data register 5*	PDR5	H'FFD8
Port data register 6*	PDR6	H'FFD9
Port data register 7*	PDR7	H'FFDA
Port data register 8*	PDR8	H'FFDB
Port data register 9*	PDR9	H'FFDC

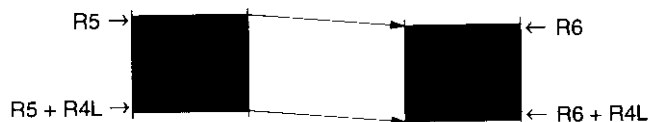
Note: \* Port data registers have the same addresses as input pins.

**Table 2-13 Registers with Write-Only Bits**

Register Name	Abbreviation	Address
Port control register 1	PCR1	H'FFE4
Port control register 2	PCR2	H'FFE5
Port control register 3	PCR3	H'FFE6
Port control register 5	PCR5	H'FFE8
Port control register 6	PCR6	H'FFE9
Port control register 7	PCR7	H'FFEA
Port control register 8	PCR8	H'FFEB
Port control register 9	PCR9	H'FFEC
PWM control register	PWCR	H'FFD0
PWM data register U	PWDRU	H'FFD1
PWM data register L	PWDRL	H'FFD2

### 2.9.3 Notes on Use of the EEPMOV Instruction

- The EEPMOV instruction is a block data transfer instruction. It moves the number of bytes specified by R4L from the address specified by R5 to the address specified by R6.



- When setting R4L and R6, make sure that the final destination address (R6 + R4L) does not exceed H'FFFF. The value in R6 must not change from H'FFFF to H'0000 during execution of the instruction.

