Hitachi Single-Chip Microcomputer

H8/3101

HD6483101

User’s Manual

Preface

The H8/3101 is a single-chip microcomputer built around a high-speed H8/300 CPU core. On-chip facilities include 8-kbyte EEPROM, 10-kbyte ROM, 256-byte RAM, and two I/O ports.

On-chip EEPROM makes the H8/3101 ideal for applications requiring nonvolatile data storage, including smart cards and portable data banks. Security functions protect data in the on-chip ROM and EEPROM against external reading and writing.

This manual describes the H8/3101 hardware. For details of the instruction set, refer to the *H8/300 Series Programming Manual*.

Contents

Appendices

Tables

Figures

Section 1. Overview

1.1 Overview

The H8/3101 is a single-chip microcomputer unit (MCU) built around a high-speed H8/300 CPU core. An 8-kbyte EEPROM, 10-kbyte ROM, 256-byte RAM, and 2-bit I/O port are integrated onto the H8/3101 chip.

Operating at a maximum 5-MHz clock rate, the H8/3101 rapidly executes bit-manipulation instructions, arithmetic and logic instructions, and data transfer instructions.

Security functions protect the data in the on-chip ROM and EEPROM.

Table 1-1 lists the features of the H8/3101.

Table 1-1. Features

Item	Specification
CPU	Two-way general register configuration • Sixteen 8-bit registers, or • Eight 16-bit registers High-speed operation • Maximum clock rate: 5 MHz (with 10-MHz external clock input) • Add/subtract: 0.4 µs • Multiply/divide: 2.8 µs Streamlined, concise instruction set • Instruction length: 2 or 4 bytes • Register-register arithmetic and logic operations • MOV instruction for data transfer between registers and memory Instruction set features • Multiply instruction (8 bits × 8 bits) • Divide instruction (16 bits ÷ 8 bits) • Bit-accumulator instructions • Register-indirect specification of bit positions
On-chip supporting modules	EEPROM • 8 kbytes • Written by EEPMOV instruction • Page (32 bytes) write and erase • Protected against accidental writing and erasing • On-chip voltage pumping circuit ROM • 10 kbytes RAM • 256 bytes I/O ports • Two general-purpose input/output ports (one also used for interrupts)

Table 1-1. Features (cont)

Item	Specification
Power	Single-voltage power supply • 5 V ± 10%
Clock frequency range	f_{OP} = 500 kHz to 5 MHz (V_{CC} = 5 V ± 10%) f_{CLK} = 1 MHz to 10 MHz (external clock input)
Interrupts	• One external interrupt line: I/O–1/$\overline{IRQ}$ Used for interrupt input in sleep mode
Power-down state	• Sleep mode

1.2 Block Diagram

Figure 1-1 shows an internal block diagram of the H8/3101.

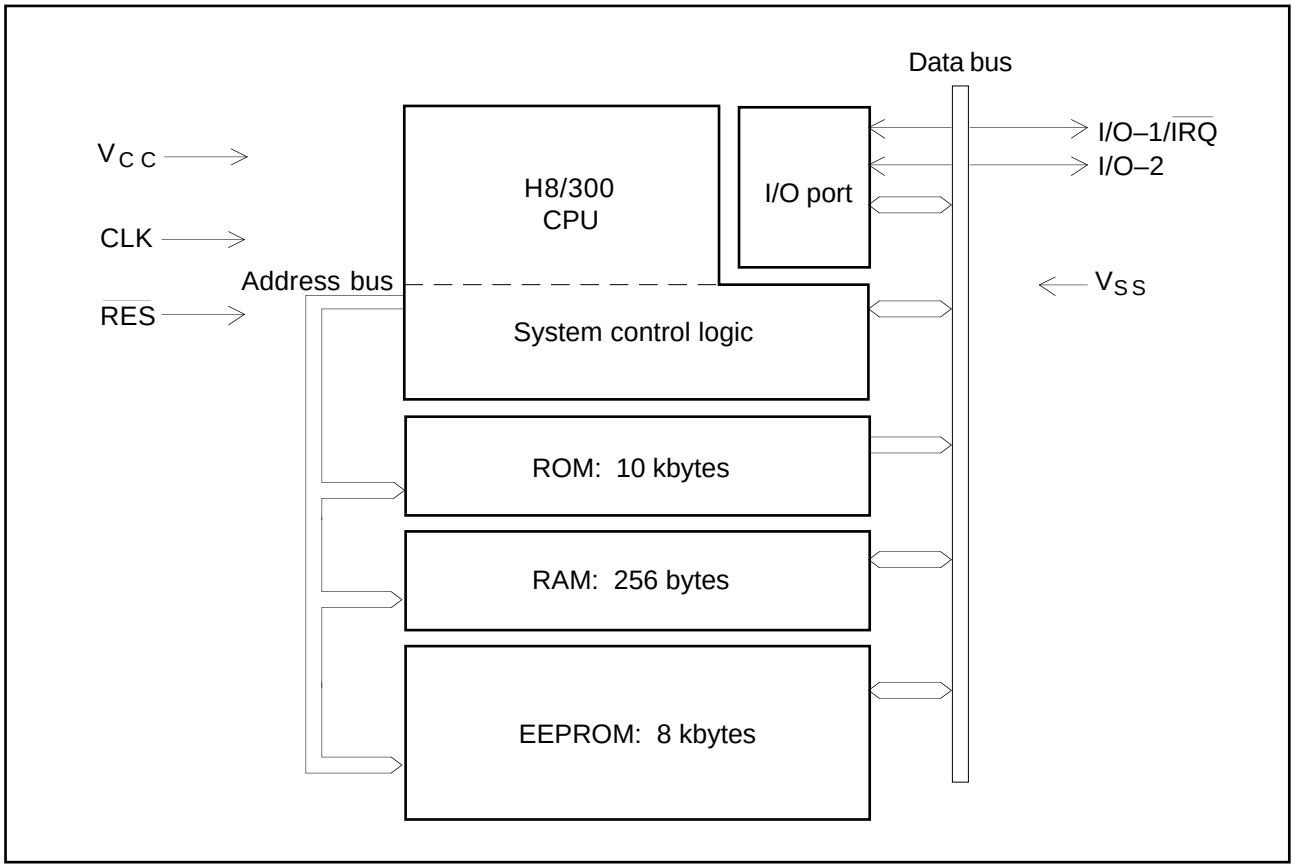
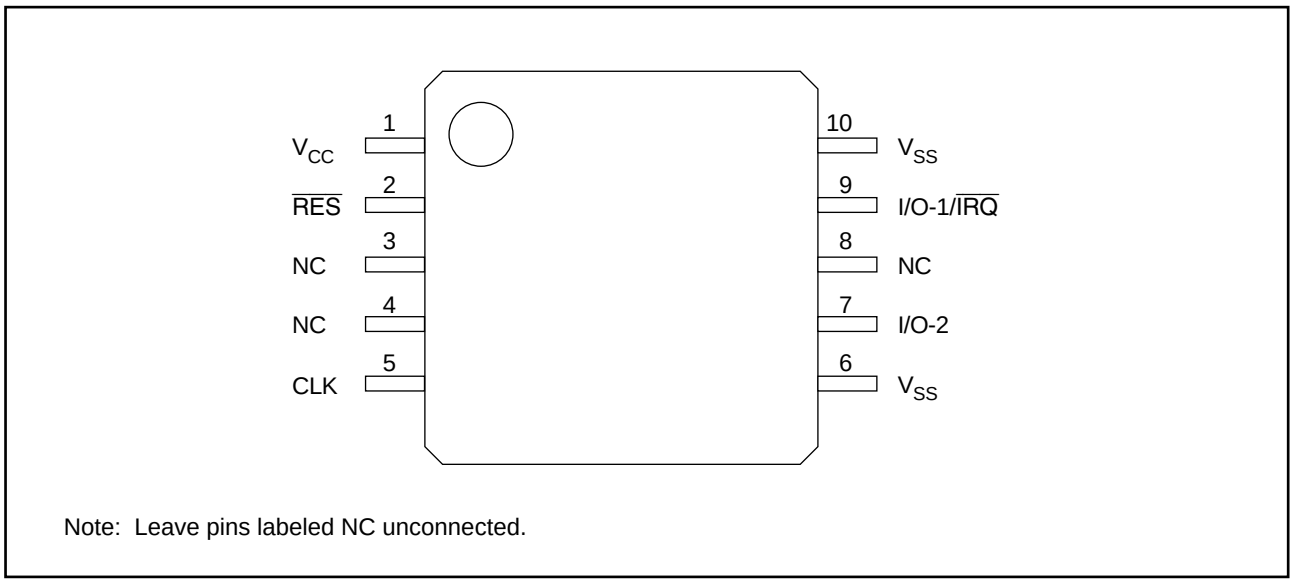


Figure 1-1. Block Diagram

1.3 Pin Arrangement and Functions

1.3.1 Pin Arrangement

Figure 1-2 shows the pin arrangement of the H8/3101 in the SOP-10 package. Figure 1-3 shows the standard COB pattern. Figure 1-4 shows the bonding pad arrangement of the H8/3101 chip.



Note: Leave pins labeled NC unconnected.

Figure 1-2. SOP-10 Pin Arrangement (top view)

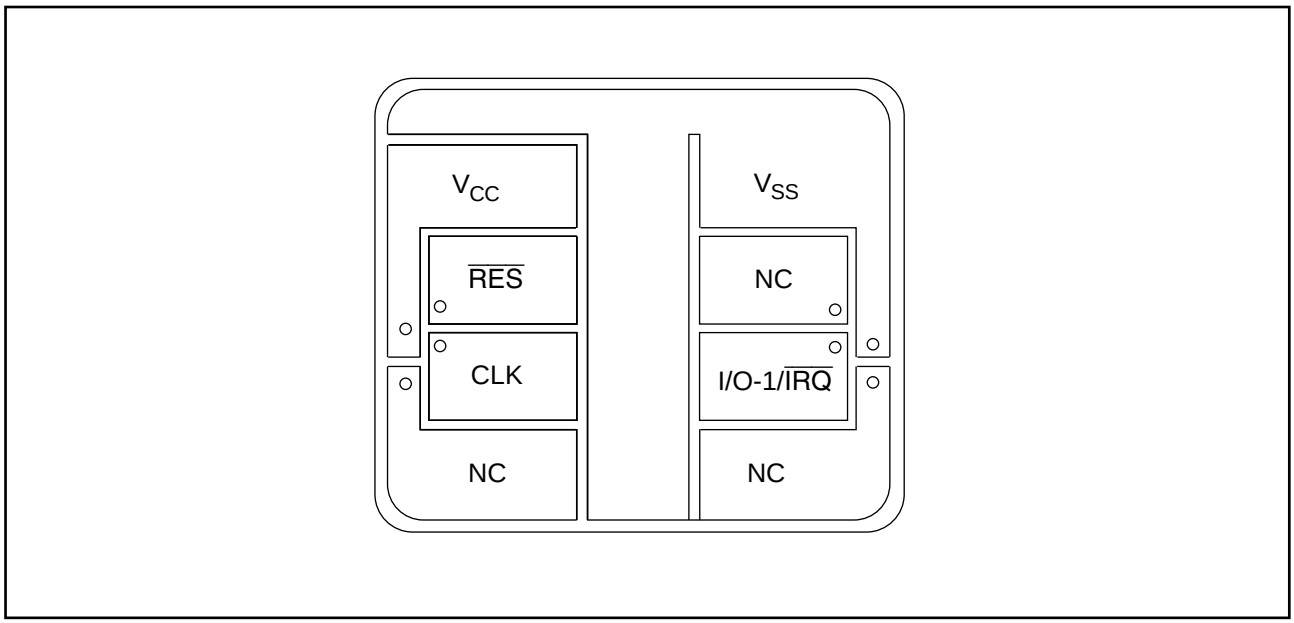


Figure 1-3. Standard COB Pattern (electrode surface)

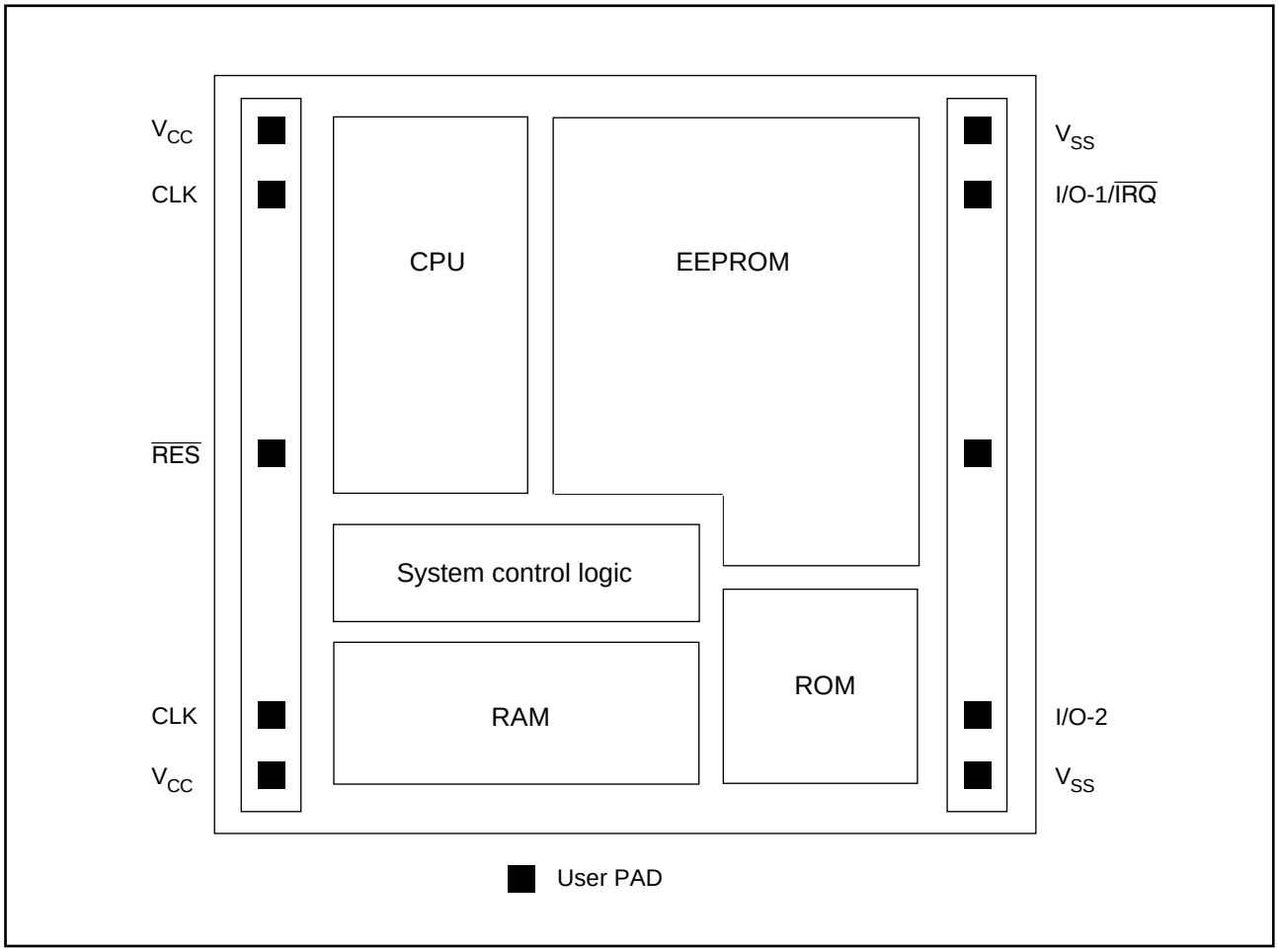


Figure 1-4. Bonding Pad Layout

1.3.2 Pin Functions

Table 1-2 lists the functions of the H8/3101 pins.

Table 1-2. Pin Functions

Type	Symbol	I/O	Name and Description	
Power supply	V_{CC}	I	Power supply: 5 V	
	V_{SS}	I	Ground: 0 V	
Clock	CLK	I	Clock: external clock input	
Reset	$\overline{RES}$	I	Reset: low input resets the chip	
Ports	I/O-1/$\overline{IRQ}$	I/O	I/O port 1:	One-bit data input/output port. Software can select input or output.
			Interrupt:	In sleep mode, this port receives interrupt input.
	I/O-2	I/O	I/O port 2:	One-bit data input/output port. Software can select input or output.

Note: V_{CC}, V_{SS}, and CLK have two bonding pads apiece. When the H8/3101 is mounted as a bare chip, either pad may be used or both pads may be used.

Section 2. CPU

2.1 Overview

The H8/3101 has the generic H8/300 CPU: an 8-bit central processing unit with a speed-oriented architecture featuring sixteen general registers. This section describes the CPU features and functions, including a concise description of the addressing modes and instruction set. For further details on the instructions, see *the H8/300 Series Programming Manual.*

2.1.1 Features

The main features of the H8/300 CPU are listed below.

- Two-way register configuration
  - — Sixteen 8-bit general registers, or
  - — Eight 16-bit general registers
- Instruction set with 55 basic instructions*, including:
  - — Multiply and divide instructions
  - — Powerful bit-manipulation instructions
  - — EEPROM write instruction
- Eight addressing modes

— Register direct	Rn
— Register indirect	@Rn
— Register indirect with displacement	@(d:16, Rn)
— Register indirect with post-increment or pre-decrement	@Rn+ or @–Rn
— Absolute address	@aa:8 or @aa:16
— Immediate	#xx:8 or #xx:16
— Program-counter relative	@(d:8, PC)
— Memory indirect	@@aa:8

- 64-kbyte address space

* The H8/300 CPU has 57 basic instructions, but the H8/3101 uses only 55 of them. The MOVFPE and MOVTPE instructions are not used.

- High-speed operation
  - — All frequently-used instructions are executed in two to four states
  - — Maximum clock rate is 5 MHz (with 10-MHz external clock input)
    - — 8- or 16-bit register-register add or subtract: 0.4 µs
    - — 8 × 8-bit multiply: 2.8 µs
    - — 16 ÷ 8-bit divide: 2.8 µs

- Power-down mode
  - — SLEEP instruction

2.1.2 CPU Registers

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

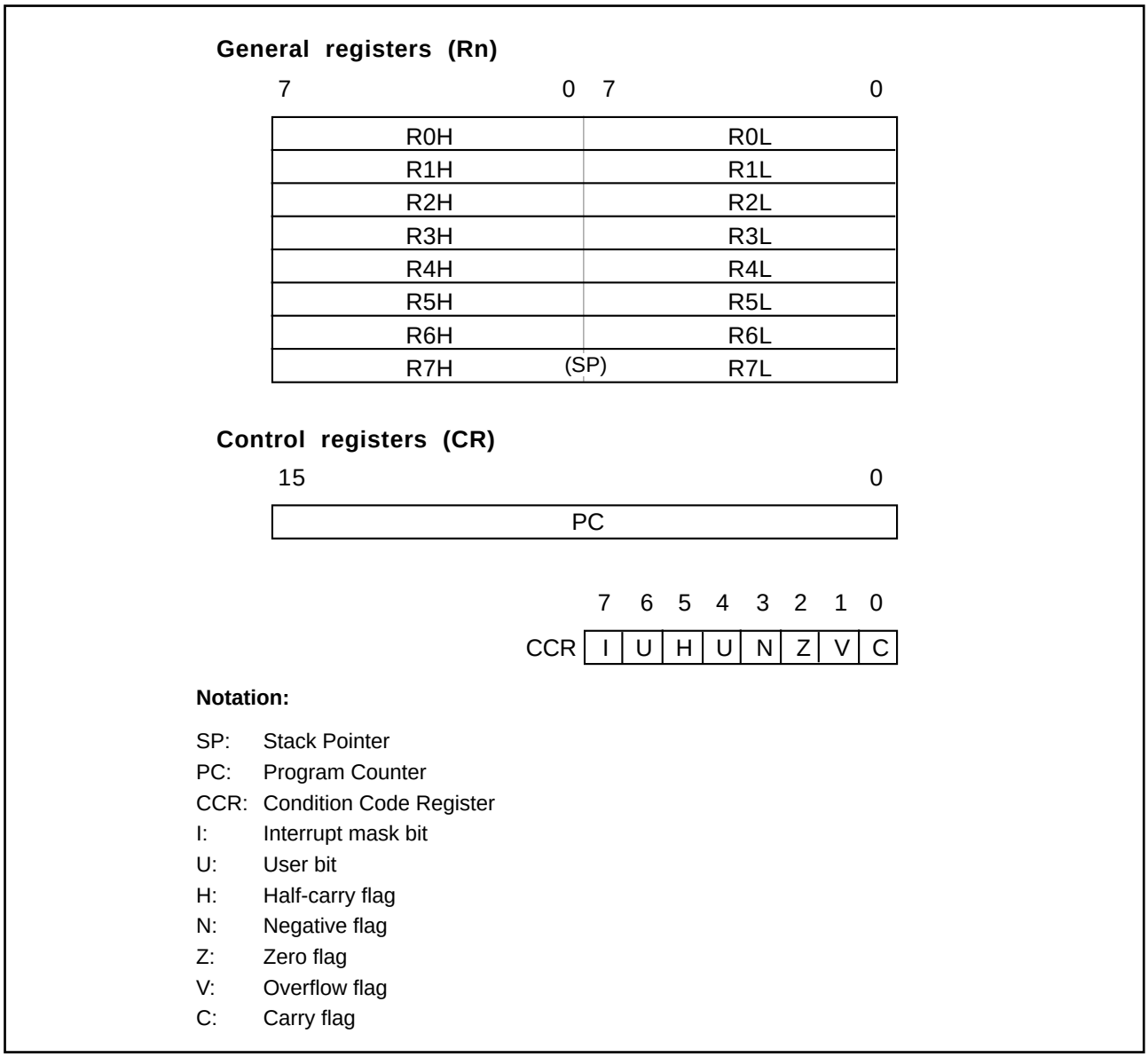


Figure 2-1. CPU Registers

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).

Registers R4L, R5, and R6 have special functions when the EEPMOV (EEPROM write) instruction is executed.

R7 also functions as the stack pointer, used implicitly by hardware in processing exceptions and subroutine calls. In assembly-language coding, R7 can also be denoted by the symbol SP. As indicated in figure 2-2, SP (R7) points to the top of the stack.

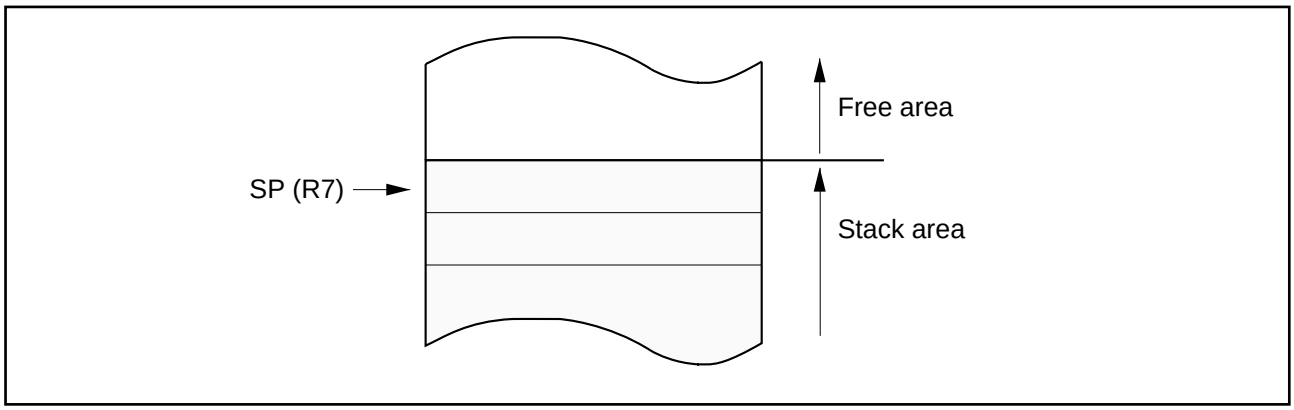


Figure 2-2. Stack Pointer

2.2.2 Control Registers

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

(1) 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).

(2) Condition Code Register (CCR): This 8-bit register contains internal CPU status information, including the interrupt mask bit (I) and half-carry (H), negative (N), zero (Z), overflow (V), and carry (C) flags.

Bit 7—Interrupt Mask Bit (I): Masks interrupts when set to 1. This bit is set to 1 at the beginning of exception processing.

Bit 6—User Bit (U): Can be written and read by software for its own purposes (using the LDC, STC, ANDC, ORC, and XORC instructions).

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 written and read by software for its own purposes (using the LDC, STC, ANDC, ORC, and XORC instructions).

Bit 3—Negative Flag (N): Indicates the most significant bit (sign bit) of data.

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

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

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

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

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

Some instructions leave some or all of the flag bits unchanged. The LDC, STC, ANDC, ORC, and XORC instructions enable the CPU to load and store the CCR, and to set or clear selected bits by logic operations. The N, Z, V, and C flags are used as branching conditions for conditional branching (Bcc) instructions.

Refer to the *H8/300 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 loaded from 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. To prevent program crashes the stack pointer should be initialized by software, by the first instruction executed after a reset.

2.3 Data Formats

The H8/300 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 (n = 0, 1, 2, ..., 7) in a byte operand.
- All arithmetic instructions except ADDS and SUBS can operate on byte data.
- The MOV.W, ADD.W, SUB.W, CMP.W, ADDS, SUBS, MULXU (8 bits × 8 bits), and DIVXU (16 bits ÷ 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.

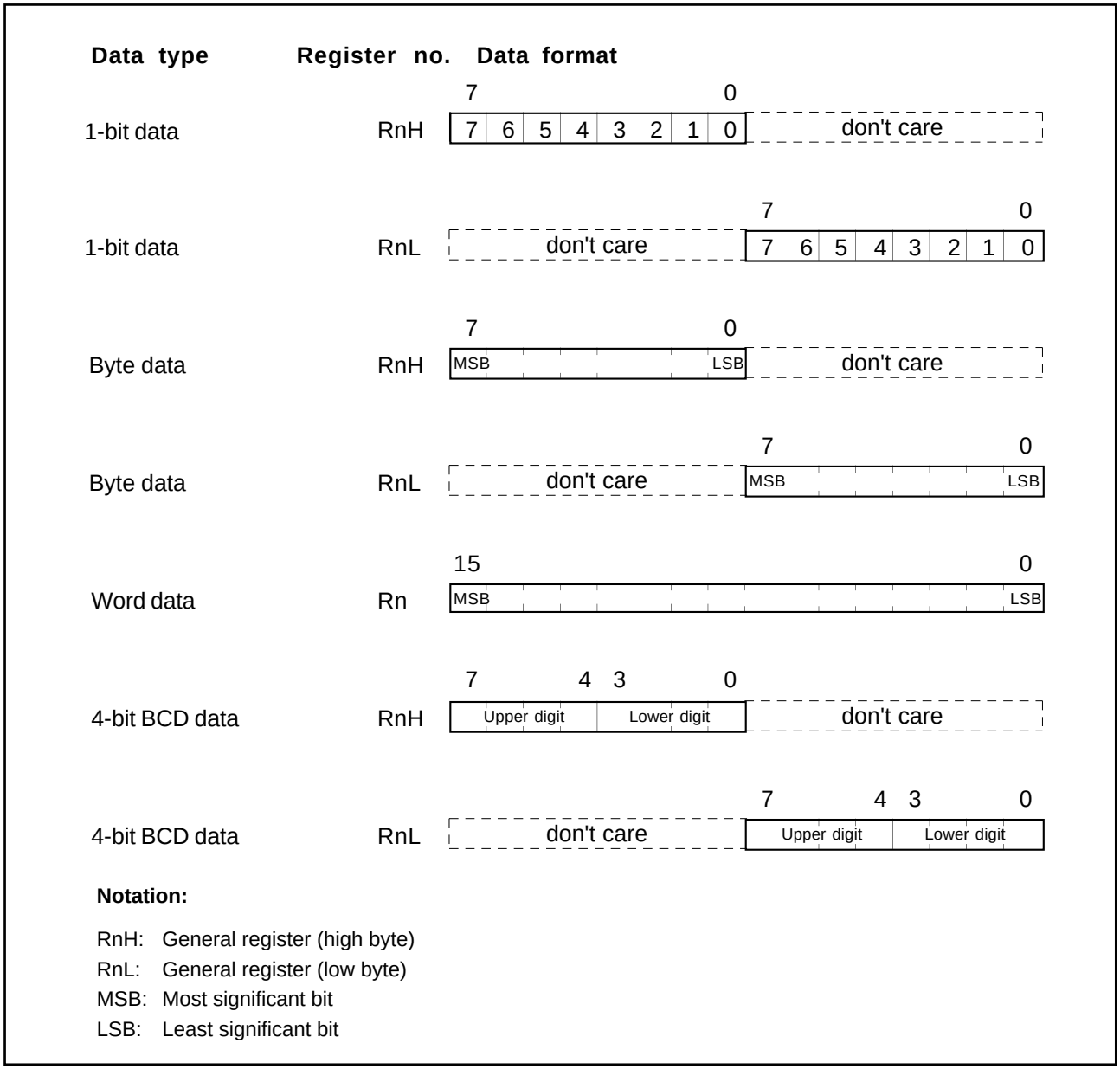


Figure 2-3. Register Data Formats

2.3.2 Memory Data Formats

Figure 2-4 indicates the data formats in memory. 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, no address error occurs but the access is performed at the preceding even address. This rule affects the MOV.W instruction, and also applies to instruction fetching.

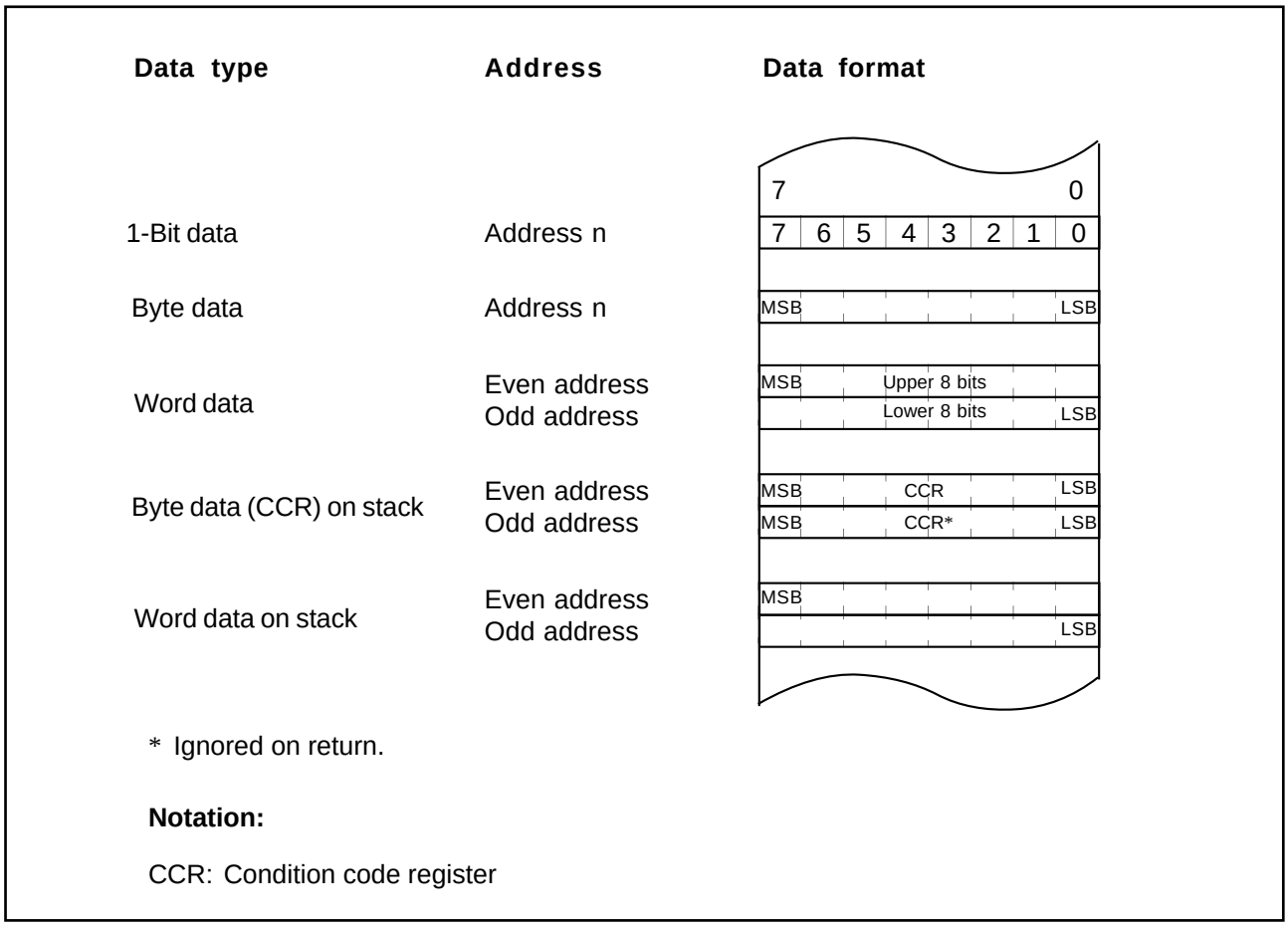


Figure 2-4. Memory Data Formats

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 H8/300 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.	Addressing Mode	Symbol
[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
[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 × 8 bits), and DIVXU (16 bits ÷ 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.

[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.

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 16-bit general 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 addresses H'0000 to H'0007 (0 to 7) are located in the vector table.

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 section 2.3.2, Memory Data Formats for further information.

2.4.2 Effective Address Calculation

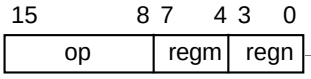
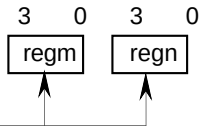
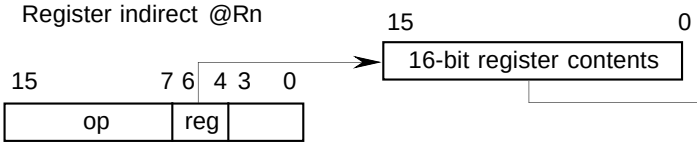
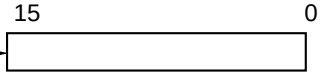
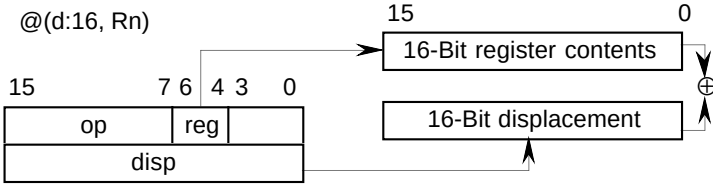
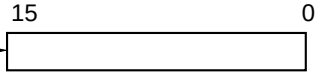
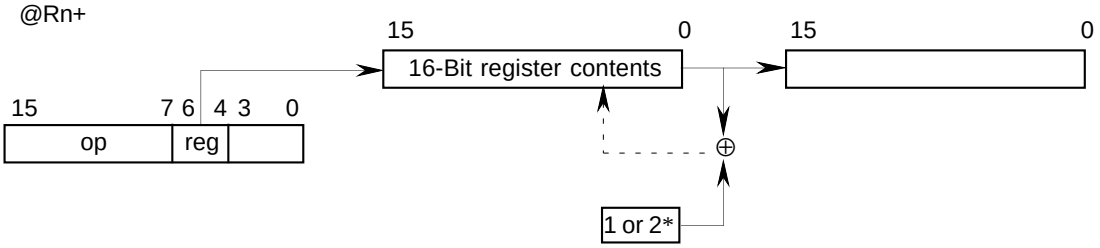
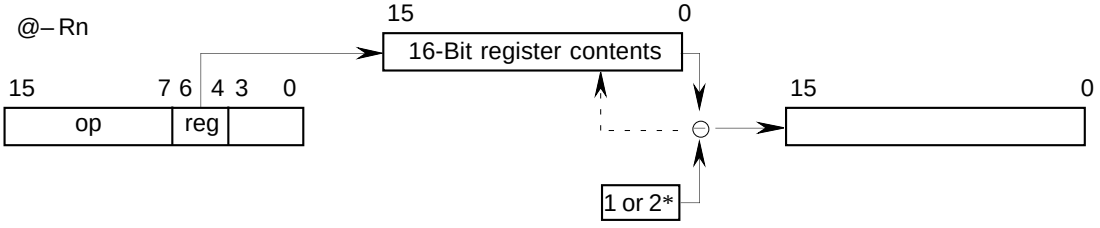
Table 2-2 shows how effective addresses are calculated in each of the addressing modes.

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].

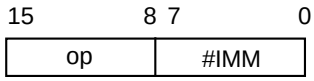
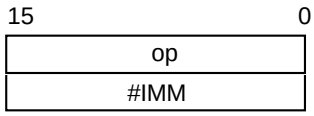
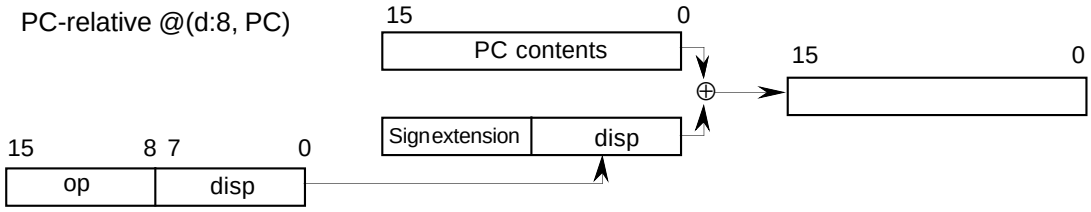
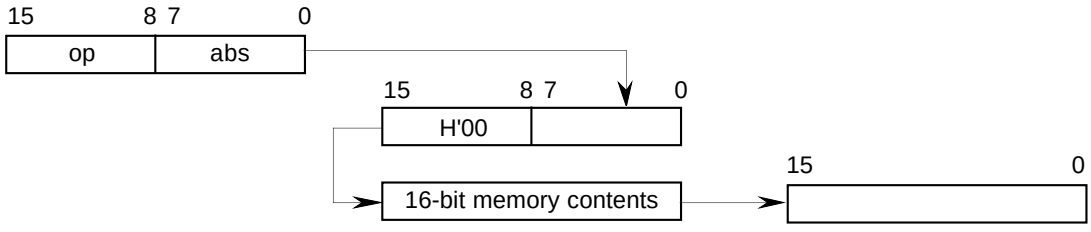
Bit manipulation instructions use register direct [1], register indirect [2], or absolute [5] addressing to specify a byte operand, and 3-bit immediate addressing [6] to specify a bit position in that byte. The BSET, BCLR, BNOT, and BTST instructions can also use register direct addressing [1] to specify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

Table 2-2. Effective Address Calculation

No.	Addressing Mode, Instruction Format	Effective Address Calculation	Effective Address
1	Register direct Rn.		 Operands are contained in registers m and n
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		

* 1 for a byte operand,
2 for a word operand

Table 2-2. Effective Address Calculation (cont)

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

Notation:

reg, regm, regn:	General registers
op:	Operation field
disp:	Displacement
IMM:	Immediate data
abs:	Absolute address

2.5 Instruction Set

The H8/3101 can use a total of 55 instructions, which are grouped by function in table 2-3.

Note: The H8/300 CPU has 57 basic instructions, but the H8/3101 uses only 55 of them. The MOVFPE and MOVTPE instructions are not used.

Table 2-3. Instruction Set

Function	Instructions	Types
Data transfer	MOV, PUSH*1, POP*1	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, BIAND, BOR, BIOR, BXOR, BIXOR, BLD, BILD, BST, BIST	14
Branch	Bcc*2, JMP, BSR, JSR, RTS	5
System control	SLEEP, LDC, STC, ANDC, ORC, XORC, NOP, RTE	8
EEPROM write	EEPMOV	1
		Total 55

Notes: *1 POP Rn is identical to MOV.W @SP+, Rn. PUSH Rn is identical to MOV.W Rn, @–SP.
*2 Bcc is a conditional branch instruction in which cc represents a condition code.

Tables 2-4 to 2-11 give a concise summary of the instructions in each functional group. The following notation is used in these tables to describe the operations performed.

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
PC	Program counter
SP	Stack pointer
#IMM	Immediate data
disp	Displacement
+	Addition
–	Subtraction
×	Multiplication
÷	Division
∧	AND logical
∨	OR logical
⊕	Exclusive OR logical
→	Move
¬	Not
:3, :8, :16	3-, 8-, or 16-bit length

2.5.1 Data Transfer Instructions

Table 2-4 describes the data transfer instructions.

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 #xx:8, @aa:8 addressing mode are available for byte data only. Specify word-size operands for @–R7 and @R7+.
POP	W	@SP+ → Rn Pops a 16-bit general register from the stack. Identical to MOV.W @SP+, Rn.
PUSH	W	Rn → @–SP Pushes a 16-bit general register onto the stack. Identical to MOV.W Rn, @–SP.

* Size: operand size
B: Byte
W: Word

Figure 2-5 shows the object code formats of the data transfer instructions.

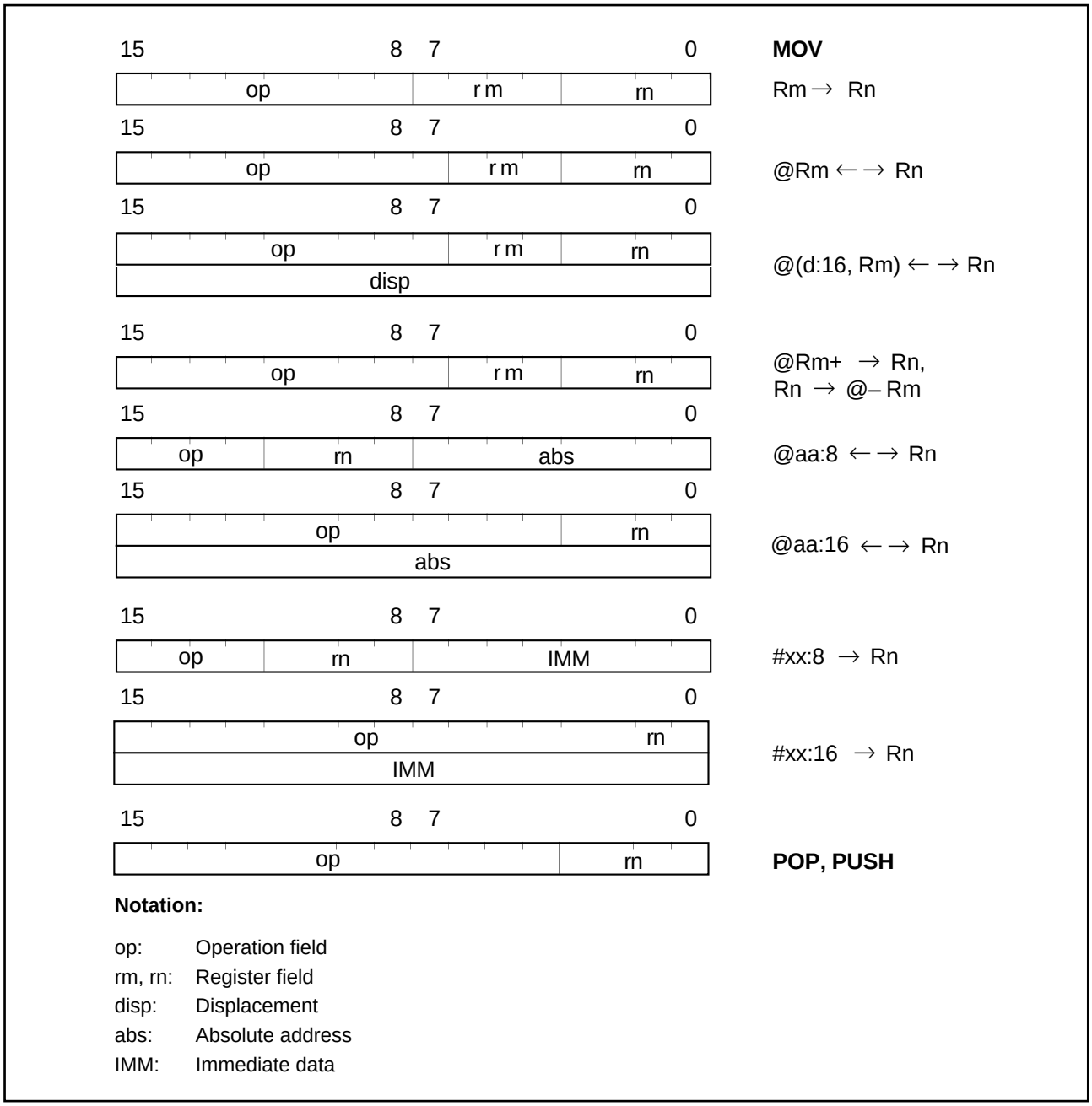


Figure 2-5. Data Transfer Instruction Object Code Formats

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 ± Rs → Rd, Rd + #IMM → 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 ± Rs ± C → Rd, Rd ± #IMM ± C → Rd Performs addition or subtraction with carry or borrow on byte data in two general registers, or on immediate data and data in a general register.
INC DEC	B	Rd ± 1 → Rd Increments or decrements a general register.
ADDS SUBS	W	Rd ± 1 → Rd, Rd ± 2 → Rd Adds or subtracts immediate data to or from data in a general register. The immediate data must be 1 or 2.
DAA DAS	B	Rd decimal adjust → Rd Decimal-adjusts 4-bit BCD data in a general register by referring to the CCR.
MULXU	B	Rd × Rs → Rd Performs 8-bit × 8-bit unsigned multiplication on data in two general registers, providing a 16-bit result.
DIVXU	B	Rd ÷ Rs → Rd Performs 16-bit ÷ 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 sets the CCR according to the result. Word data can be compared only between two general registers.
NEG	B	0 – Rd → Rd Obtains the two's complement (arithmetic complement) of data in a general register.

* Size: operand size
B: Byte
W: Word

2.5.3 Logic Operations

Table 2-6 describes the 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	$\neg$ Rd $\rightarrow$ Rd Obtains the one's complement (logical complement) of general register contents.

* Size: operand size
B: Byte

2.5.4 Shift Operations

Table 2-7 describes the shift instructions.

Table 2-7. Shift Instructions

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

* Size: operand size
B: Byte

Figure 2-6 shows the object code formats of the arithmetic, logic, and shift instructions.

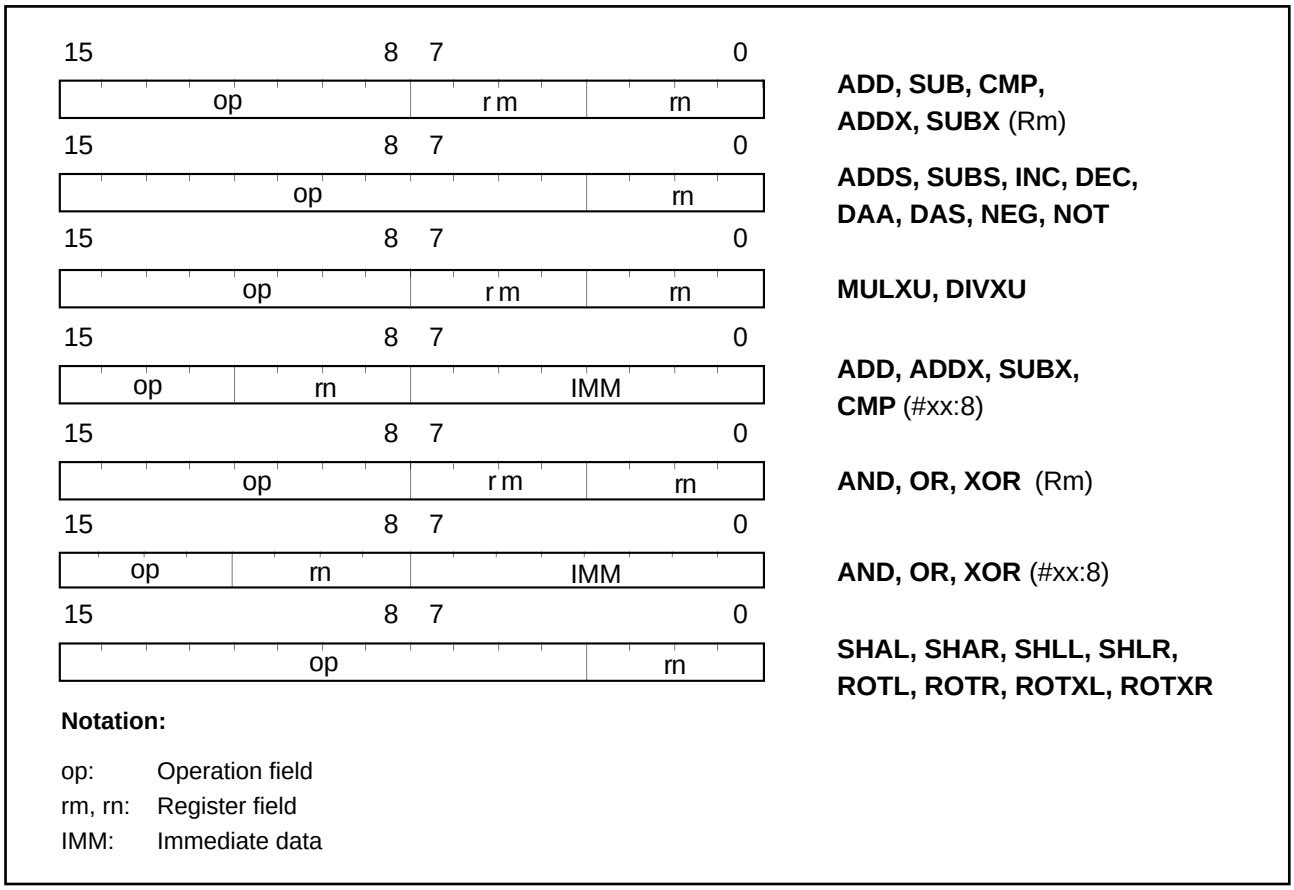


Figure 2-6. Arithmetic, Logic, and Shift Instruction Object Code Formats

2.5.5 Bit Manipulations

Table 2-8 describes the bit-manipulation instructions.

Table 2-8. Bit-Manipulation Instructions

Instruction	Size*	Function
BSET	B	1 → (<bit-No.> of <EAd>) 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 → (<bit-No.> of <EAd>) 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	¬ (<bit-No.> of <EAd>) → (<bit-No.> of <EAd>) 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	¬ (<bit-No.> of <EAd>) → 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 ∧ (<bit-No.> of <EAd>) → 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 ∧ [¬ (<bit-No.> of <EAd>)] → 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 ∨ (<bit-No.> of <EAd>) → 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 ∨ [¬ (<bit-No.> of <EAd>)] → 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.

* Size: operand size
B: Byte

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

Instruction	Size*	Function
BXOR	B	C ⊕ (<bit-No.> of <EAd>) → C Exclusive-ORs the C flag with a specified bit in a general register or memory and stores the result in the C flag.
BIXOR	B	C ⊕ [¬ (<bit-No.> of <EAd>)] → C Exclusive-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.
BLD	B	(<bit-No.> of <EAd>) → C Transfers a specified bit in a general register or memory to the C flag.
BILD	B	¬ (<bit-No.> of <EAd>) → C Transfers 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 → (<bit-No.> of <EAd>) Transfers the C flag value to a specified bit in a general register or memory.
BIST	B	¬ C → (<bit-No.> of <EAd>) Transfers the inverse of the C flag value to a specified bit in a general register or memory. The bit number is specified by 3-bit immediate data.

* Size: operand size
B: Byte

Note on Bit Manipulation Instructions: BSET, BCLR, BNOT, BST, and BIST are read-modify-write instructions. They read a byte of data, modify one bit in the byte, then write the modified byte back to the same address.

Step		Operation
1	Read	Read data (1 byte) at a specified address
2	Modify	Modify one specified bit in the read data
3	Write	Write the modified data back to the specified address

Figure 2-7 shows the object code formats of the bit manipulation instructions.

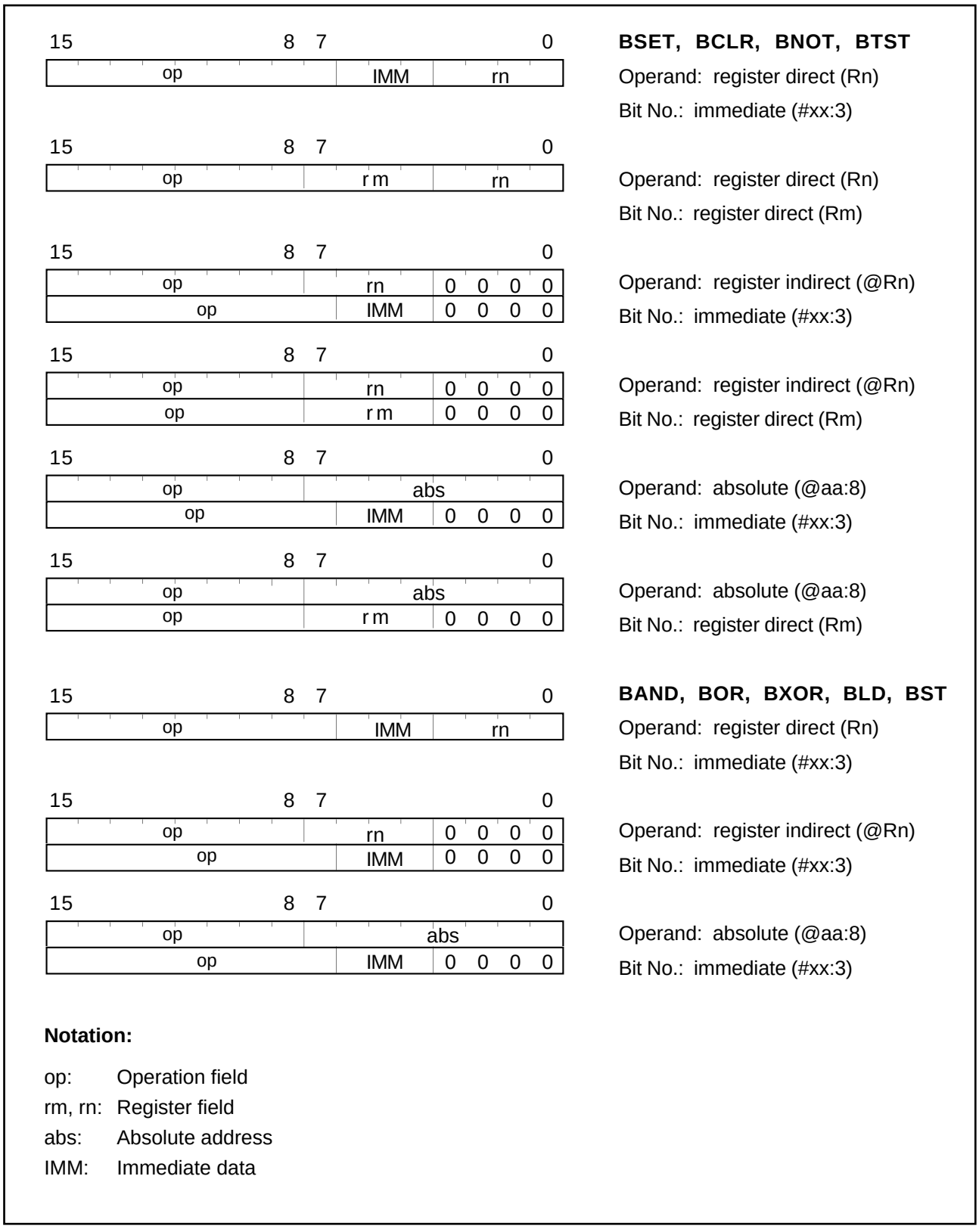
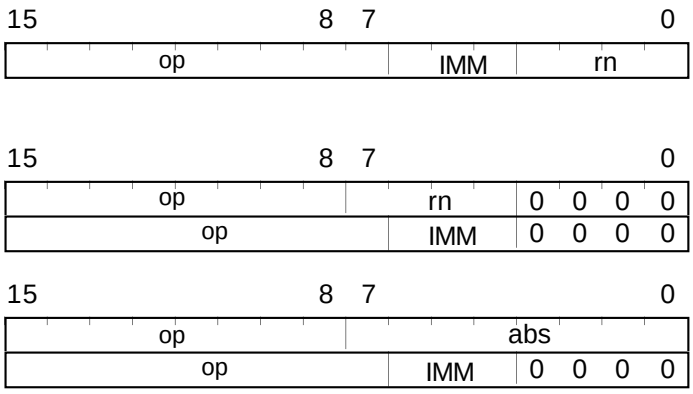


Figure 2-7. Bit Manipulation Instruction Object Code Formats



BIAND, BIOR, BIXOR, BILD, BIST

Operand: register direct (Rn)

Bit No.: immediate (#xx:3)

Operand: register indirect (@Rn)

Bit No.: immediate (#xx:3)

Operand: absolute (@aa:8)

Bit No.: immediate (#xx:3)

Notation:

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

Figure 2-7. Bit Manipulation Instruction Object Code Formats (cont)

2.5.6 Branching Instructions

Table 2-9 describes the branching instructions.

Table 2-9. Branching Instructions

Instruction	Size	Function
Bcc	—	Branches to a specified address if condition cc is true. The branching conditions are listed below.

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 displacement from the current address.
JSR	—	Branches to a subroutine at a specified address.
RTS	—	Returns from a subroutine.

Figure 2-8 shows the object code formats of the branching instructions.

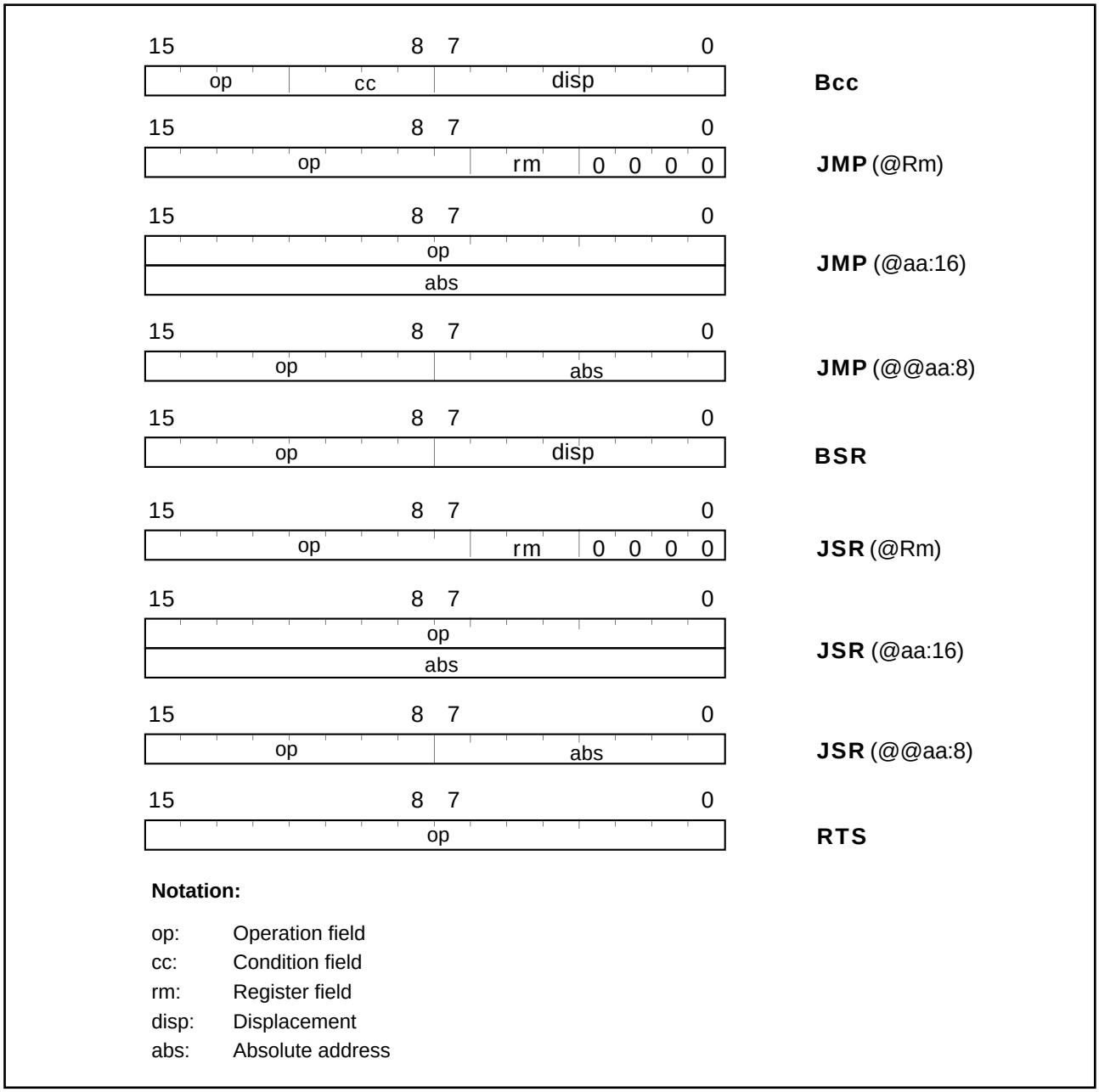


Figure 2-8. Branching Instruction Object Code Formats

2.5.7 System Control Instructions

Table 2-10 describes the system control instructions.

Table 2-10. System Control Instructions

Instruction	Size	Function
RTE	—	Returns from an exception-handling routine.
SLEEP	—	Causes a transition to the power-down state.
LDC	B	Rs → CCR, #IMM → CCR Moves immediate data or general register contents to the condition code register.
STC	B	CCR → Rd Copies the condition code register to a specified general register.
ANDC	B	CCR ∧ #IMM → CCR Logically ANDs the condition code register with immediate data.
ORC	B	CCR ∨ #IMM → CCR Logically ORs the condition code register with immediate data.
XORC	B	CCR ⊕ #IMM → CCR Logically exclusive-ORs the condition code register with immediate data.
NOP	—	PC + 2 → PC Only increments the program counter.

* Size: operand size
  B: Byte

Figure 2-9 shows the object code formats of the system control instructions.

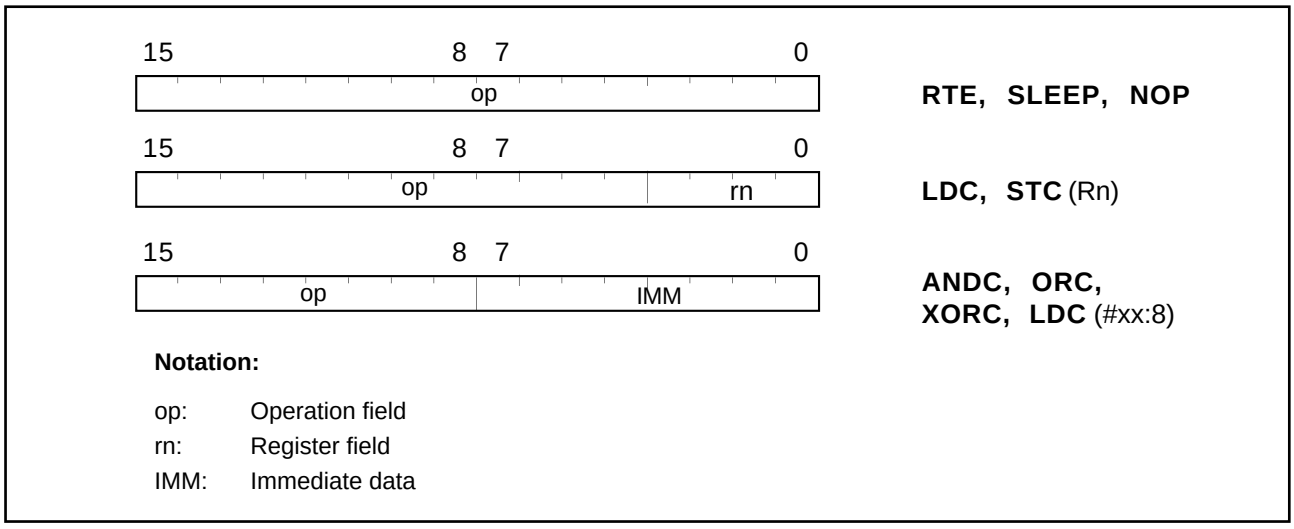


Figure 2-9. System Control Instruction Object Code Formats

2.5.8 EEPROM Write Instruction

Table 2-11 describes the EEPROM write instruction.

Table 2-11. EEPROM Write Instruction

Instruction	Size	Function
EEPMOV	—	If R4L ≠ 0 then repeat @R5+ → @R6+, R4L – 1 → R4L until R4L = 0 else next; Transfers a data block to EEPROM according to parameters set in general registers R4L, R5, and R6. R4L: size of block (bytes) R5: starting source address R6: starting destination address Execution of the next instruction begins as soon as the EEPROM write operation is completed. The transfer cannot cross an EEPROM page boundary.

Figure 2-10 shows the object code format of the EEPROM write instruction.

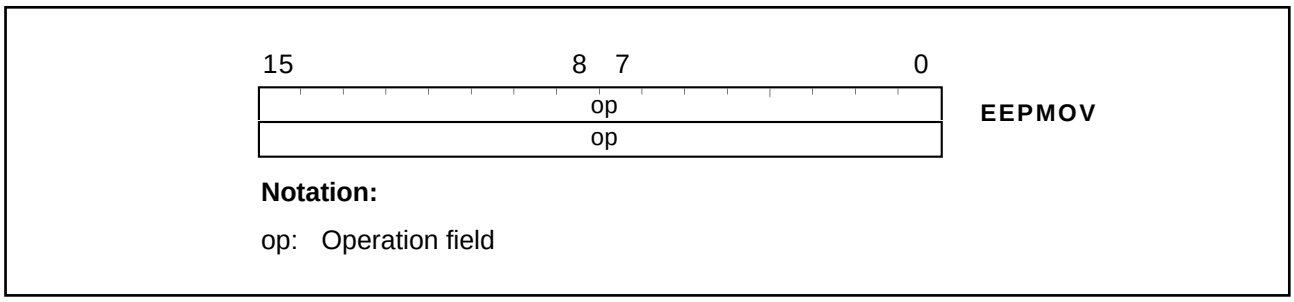


Figure 2-10. EEPROM Write Instruction Object Code Format

2.6 Operating States

2.6.1 Overview

The CPU operates in three states: the program execution state, exception processing state, and power-down state. Figure 2-11 summarizes these states. Figure 2-12 shows the state transitions.

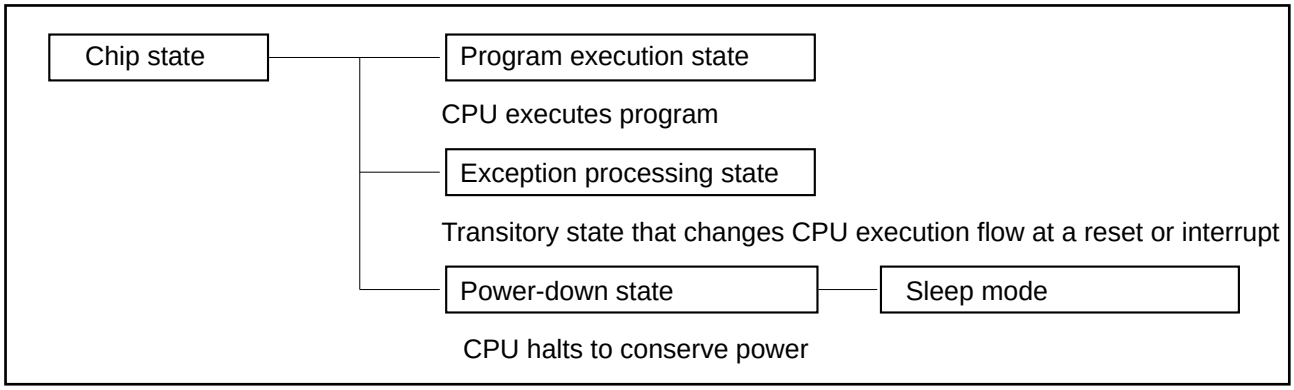


Figure 2-11. Operating States

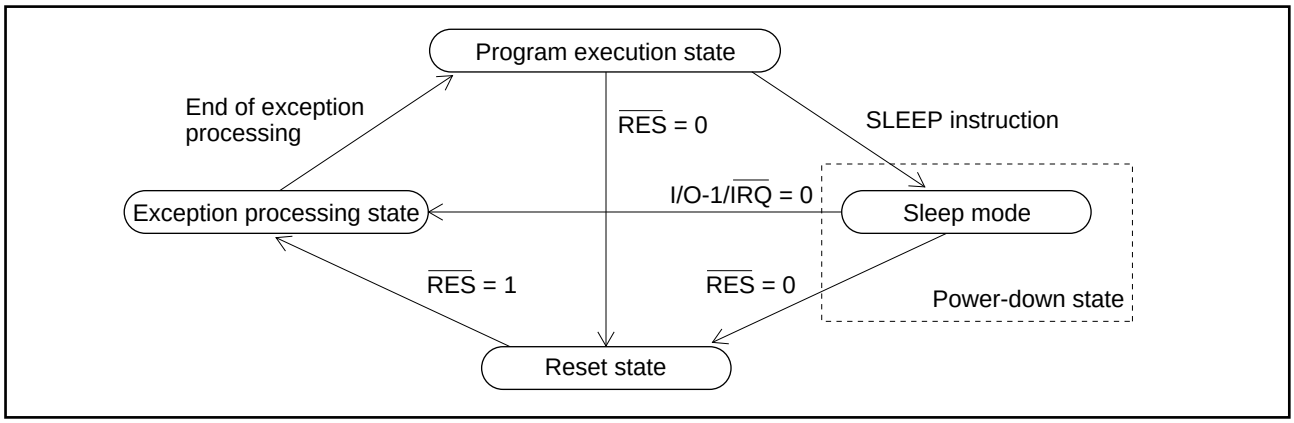


Figure 2-12. State Transitions

2.6.2 Program Execution State

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

2.6.3 Exception Processing State

This is a transitory state entered in response to a reset or interrupt. In interrupt exception processing, the stack pointer is referenced and the program counter and condition code register are saved.

2.6.4 Power Down State

The power-down state is a sleep mode.

Sleep mode is entered when the SLEEP instruction is executed. Sleep mode is exited by low input at the $\overline{\text{RES}}$ or I/O–1/$\overline{\text{IRQ}}$ pin.

2.7 Exception Processing

2.7.1 Overview

In the H8/3101, exception processing is performed in response to a reset or interrupt. Table 2-12 summarizes the exception processing priority and timing. Table 2-13 describes the exception vector table.

Table 2-12. Exception Processing Priority and Timing

Priority	Cause	Detection Timing	Start of Exception Processing Sequence
High	Reset	Synchronized with clock	Instruction execution stops and reset processing starts immediately.
↕	Interrupt (IRQ)	Falling edge is detected	Interrupt exception processing starts immediately.
Low			

Table 2-13. Exception Vector Table

		Vector Address	
Description	**Vector Number**	**PC (high)**	**PC (low)**
Reset	0	H'0000	H'0001
Reserved for system use*	1	H'0002	H'0003
Reserved for system use*	2	H'0004	H'0005
Interrupt (IRQ)	3	H'0006	H'0007

* Software should not access these addresses.

2.7.2 Reset

The H8/3101 begins reset exception processing when the $\overline{\text{RES}}$ input changes from low to high.

At power-up, $\overline{\text{RES}}$ should be held low for at least 20 external clock cycles after the input clock signal (CLK) stabilizes. Similarly, when the chip is reset during operation, $\overline{\text{RES}}$ should be held low for at least 20 external clock cycles. $\overline{\text{RES}}$ should also be low whenever power is switched on or off.

When a low-to-high transition of $\overline{\text{RES}}$ is detected, the CPU begins reset exception processing, which in the H8/3101 consists of the following steps:

1. The low-to-high transition of the $\overline{\text{RES}}$ input is detected.
2. The internal status of the CPU and the registers of the on-chip supporting modules are initialized. In the CCR, the I bit is set to 1 but other bits are left unchanged.
3. The reset vector is read from addresses H'0000 to H'0001 in the vector table and loaded into the program counter. Program execution then starts from the loaded address (start address).

Figure 2-13 shows the timing of the reset sequence.

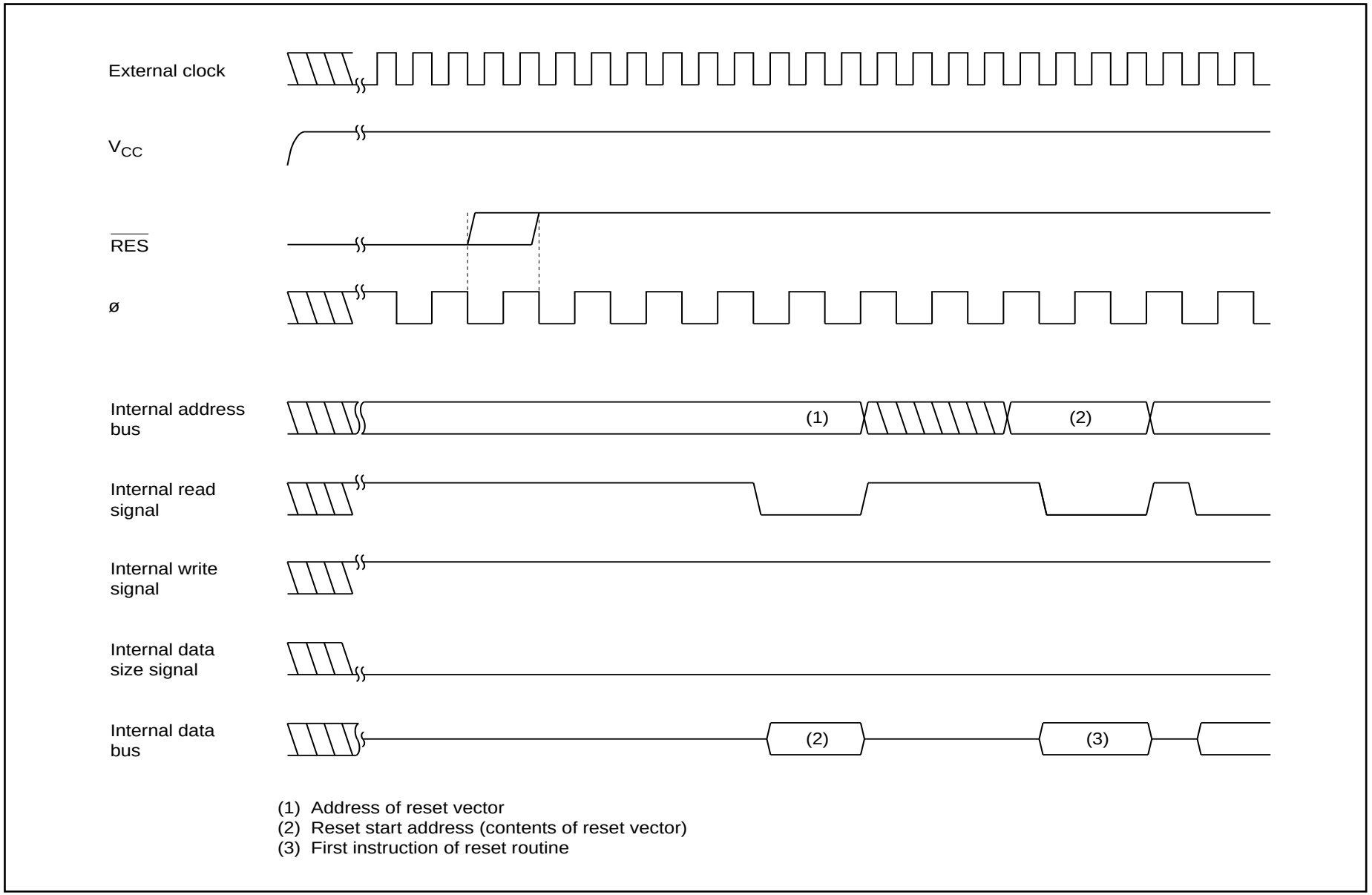


Figure 2-13. Reset Start Timing

2.7.3 Interrupts

The only interrupt source in the H8/3101 is the external interrupt request (IRQ). Interrupt requests can be made only in the sleep mode, when I/O–1/$\overline{\text{IRQ}}$ becomes an interrupt signal input line. An interrupt request is accepted when the falling edge of the $\overline{\text{IRQ}}$ input is detected, provided the I bit in the CCR is cleared to 0.

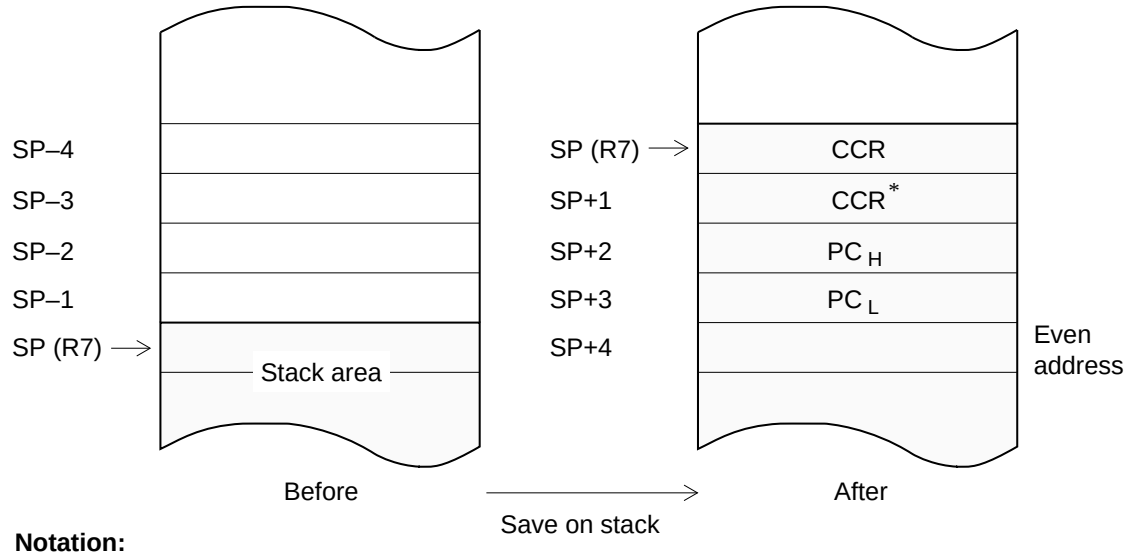
The interrupt sequence consists of the following steps.

1. When the interrupt request is accepted, a transition takes place from the sleep mode to the exception processing state. The program counter and condition code register are saved on the stack as shown in figure 2-14. The program counter address saved on the stack is the address of the first instruction that will be executed after the return from the interrupt-handling routine.

2. The I bit in the condition code register is set to 1.

3. The address of the interrupt-handling routine is read from the vector table entry corresponding to the interrupt vector and loaded into the program counter, and execution of the interrupt-handling routine begins.

Figure 2-18 shows the timing of interrupt sequence.



Notation:
PC_H: Upper 8 bits of program counter (PC)
PC_L: Lower 8 bits of program counter (PC)
CCR: Condition code register
SP: Stack pointer

* Ignored on return

Notes: 1. The program counter indicates the address of the first instruction that will be executed after the return.
2. Registers must be saved and restored by word access starting at an even address.

Figure 2-14. Stack before and after Interrupt Exception-Handling Sequence

2.8 Power-Down State

2.8.1 Overview

The H8/3101 has a sleep mode, a power-down state in which CPU functions are halted to conserve power.

Table 2-14 summarizes the conditions for transition to the sleep mode, the state of the CPU and on-chip supporting modules in sleep mode, and the conditions for exit from sleep mode.

Table 2-14. Power-Down State

		States							
Mode	Entering Procedure	Clock	CPU	CPU Reg's	RAM	DR, DDR	I/O Ports	ECR, EPR	Exiting Methods
Sleep mode	Execute SLEEP instruction	Stop	Stop	Held	Held	Held	High impedance	Initialized	• $\overline{\text{RES}}$ • I/O–1 /$\overline{\text{IRQ}}$

2.8.2 Transition to Sleep Mode

Sleep mode is entered by executing the SLEEP instruction.

In the sleep mode the CPU, clock, and on-chip supporting functions halt, so power consumption is reduced to an extremely low level. As long as the necessary voltage is supplied, however, the contents of CPU registers and RAM and the I/O port registers (DR and DDR) are held. I/O–1/$\overline{\text{IRQ}}$ becomes an interrupt input line. The I/O–1/$\overline{\text{IRQ}}$ line should be kept high during sleep mode.

Figure 2-15 shows the transition sequence to sleep mode.

2.8.3 Exit from Sleep Mode

Exit from the sleep mode takes place by input to the I/O–1/$\overline{\text{IRQ}}$ or $\overline{\text{RES}}$ pin.

1. Exit by interrupt

   In sleep mode the I/O–1/$\overline{\text{IRQ}}$ pin can receive interrupt signals. When a high-to-low transition occurs at this pin, the external clock is supplied to the CPU and on-chip supporting modules, the sleep mode ends, and interrupt exception processing starts. The external clock must be stable when the interrupt signal goes low. Figure 2-16 shows the transition sequence from the sleep mode to interrupt handling. Figure 2-17 shows the timing of an interrupt in sleep mode.

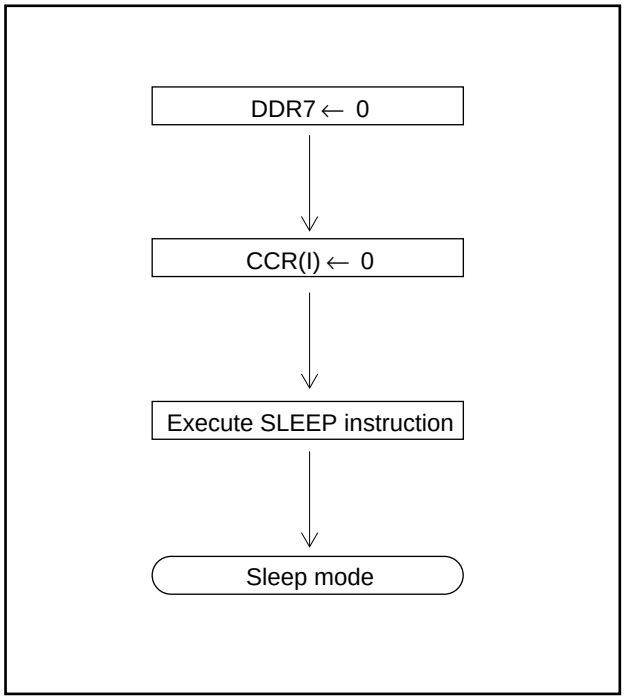


Figure 2-15. Transition Sequence to Sleep Mode (example)

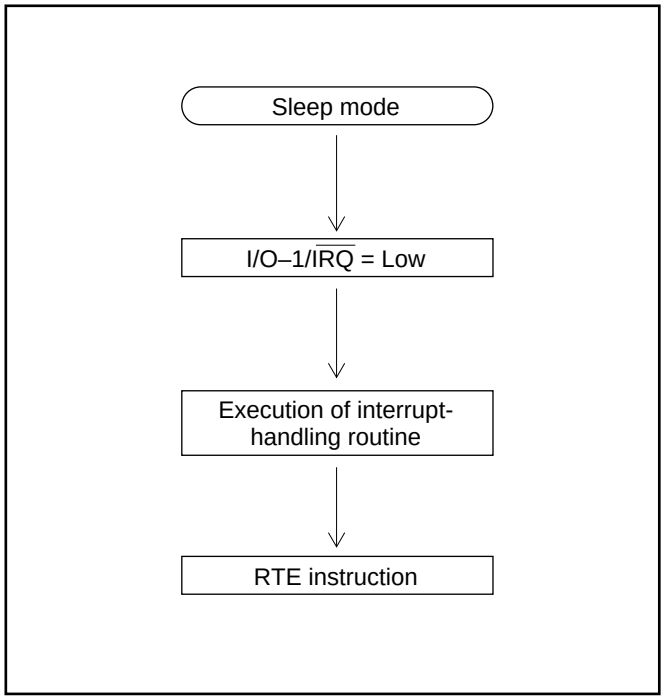


Figure 2-16. Recovery Sequence from Sleep Mode (example)

Note: The I/O–1/$\overline{\text{IRQ}}$ line must be held high during sleep mode.

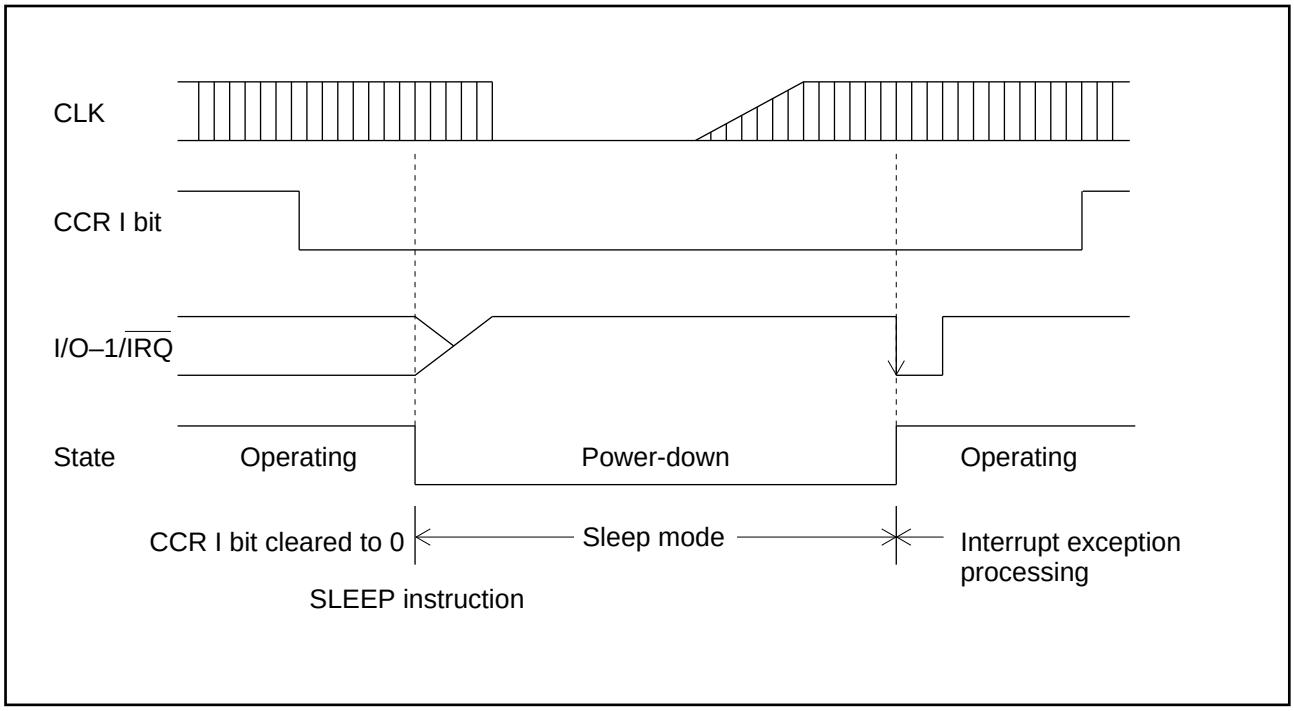


Figure 2-17. Interrupt Timing in Sleep Mode

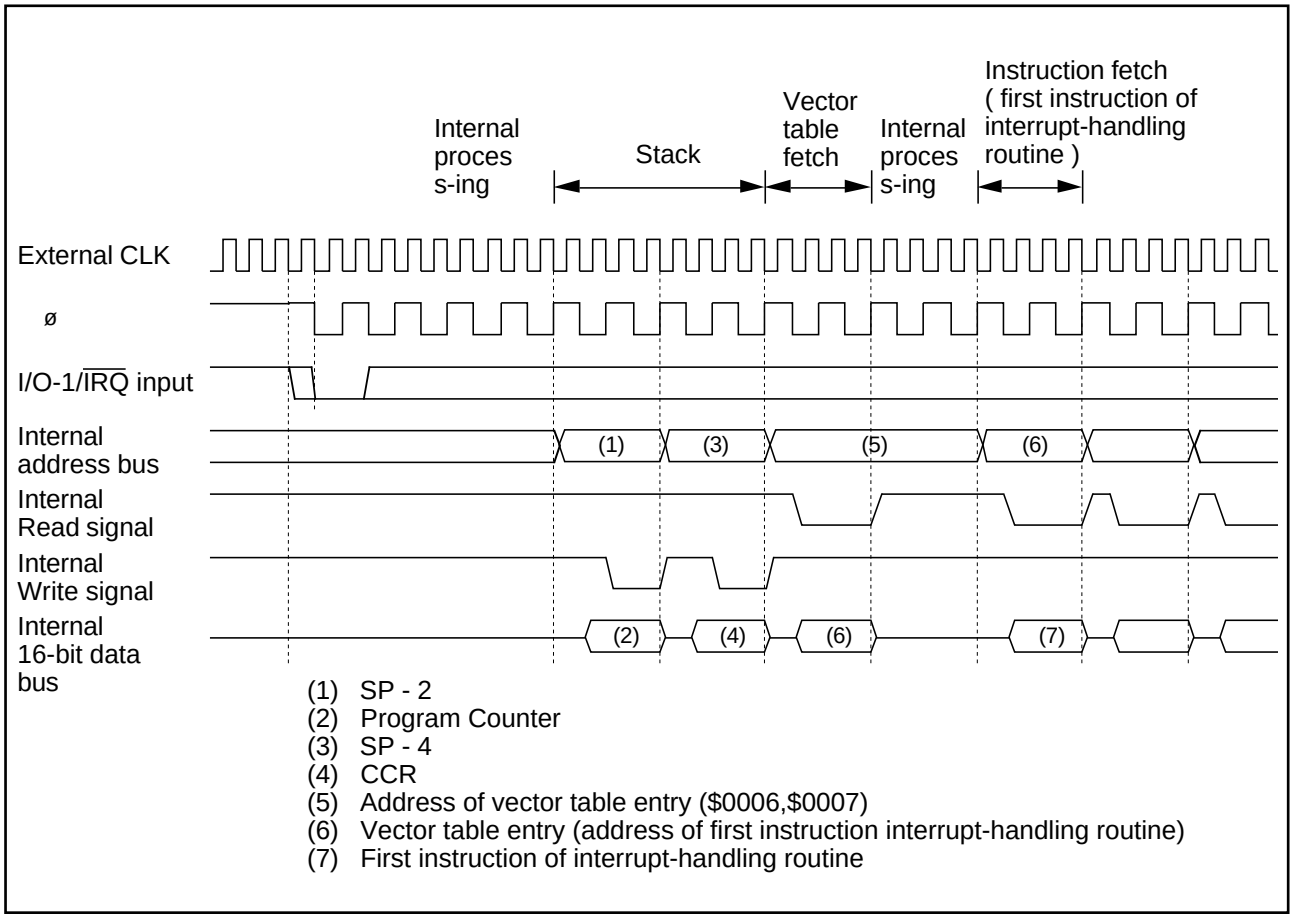


Figure 2-18. Timing of Interrupt Sequence

2. Exit by reset

   If the $\overline{\text{RES}}$ input goes low during sleep mode, the external clock is supplied to the CPU and on-chip supporting modules. Next, when the $\overline{\text{RES}}$ input goes high, the CPU begins reset exception processing. The $\overline{\text{RES}}$ input should be held low for at least 20 stable external clock cycles.

2.9 Basic Timing

The CPU operates on the system clock (ø). The interval from one rising edge of the system clock to the next is called a "state." The memory access cycle or bus cycle consists of two states.

2.9.1 On-Chip Memory (RAM, ROM, EEPROM)

The data bus is 16 bits wide. Both byte and word access are supported.

2.9.2 Register Field (I/O, EEPROM)

The upper 8 bits of the internal data bus are used to access these registers. The data bus is accordingly 8 bits wide.

2.9.3 Non-Existent Addresses

When read, non-existent addresses always return the data H'FF (byte access) or H'FFFF (word access).

Figure 2-18 shows the access timing to on-chip memory and the register field.

Section 3, Memory Map indicates the types of access possible at each address.

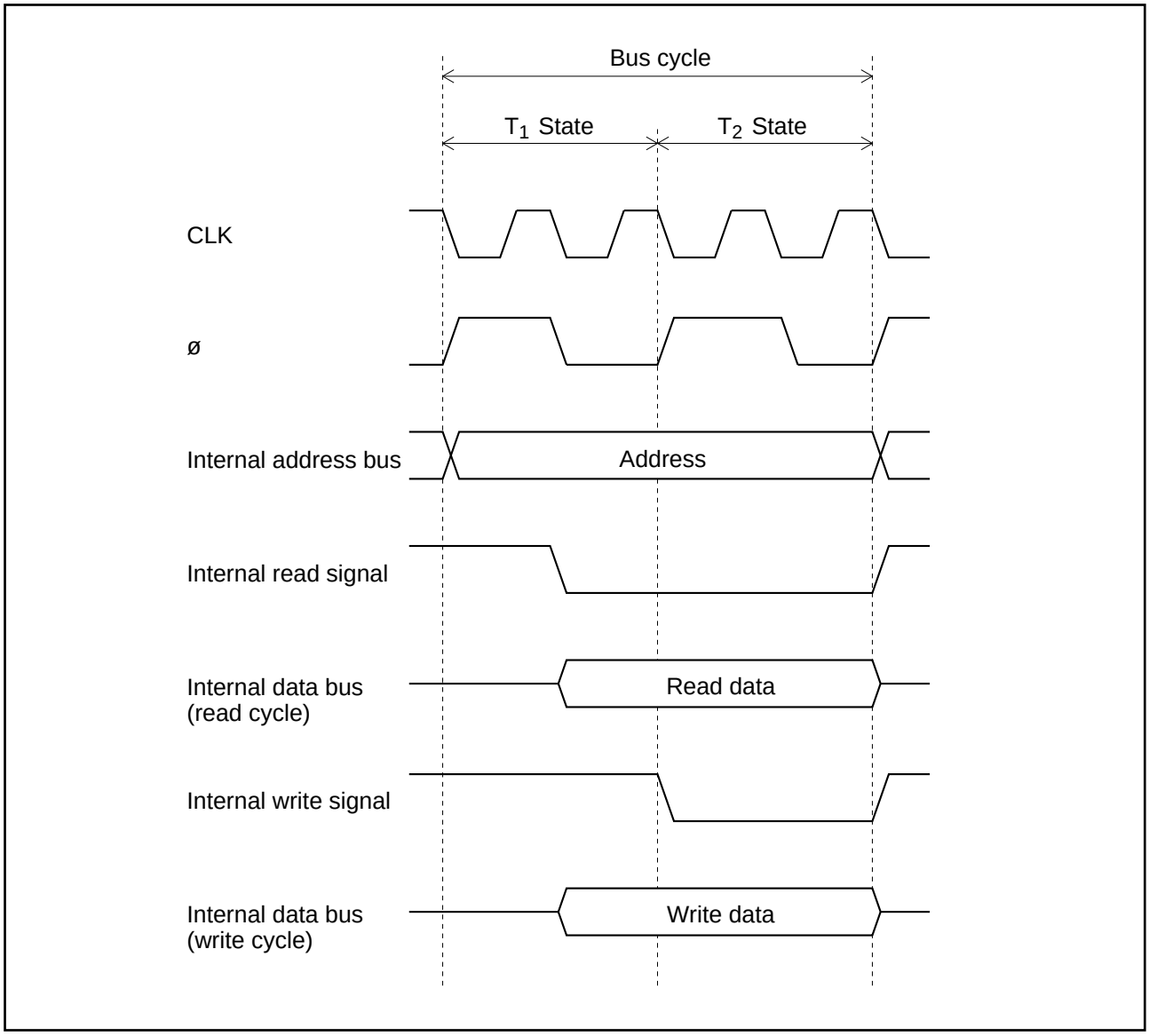
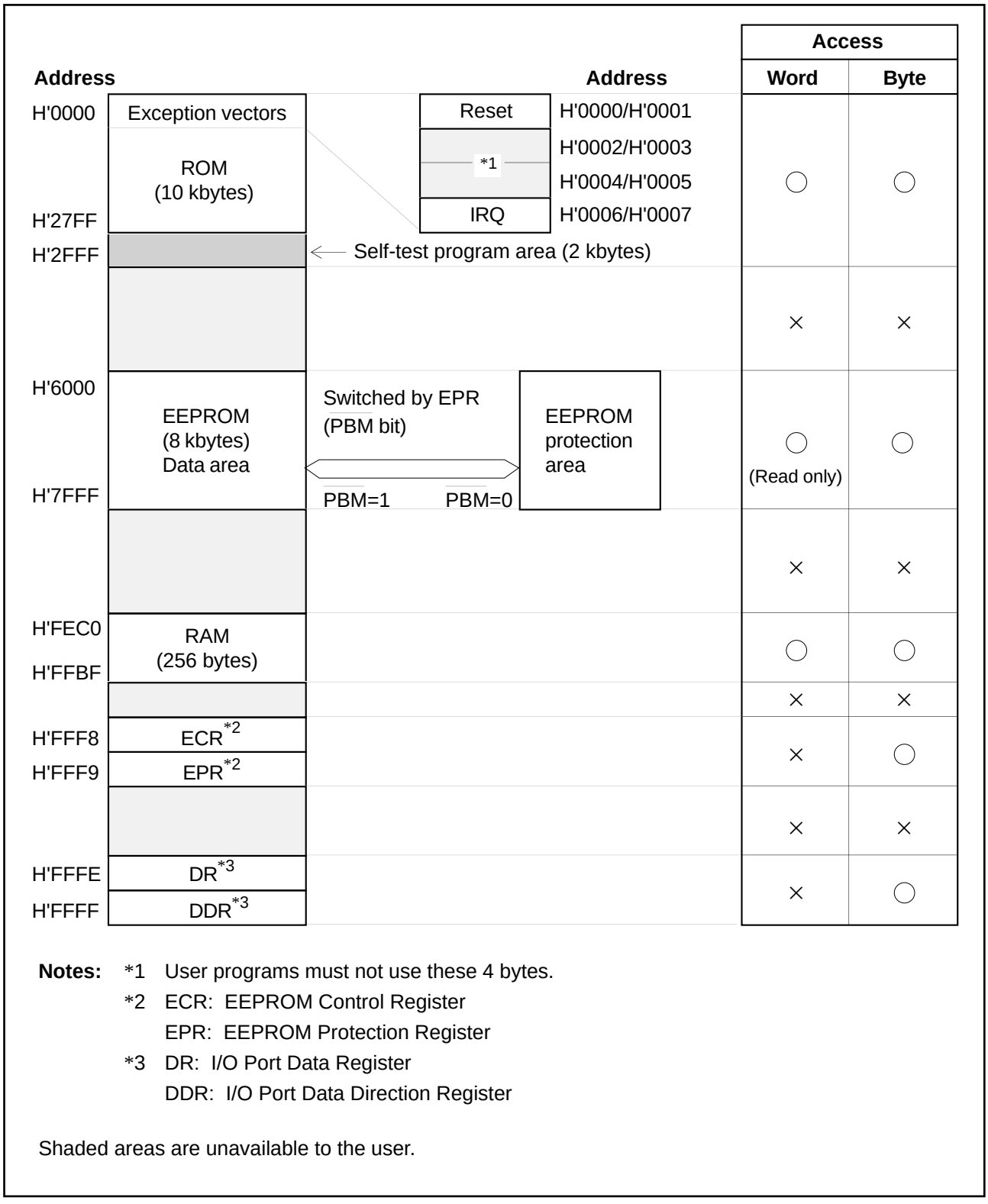


Figure 2-19. Access Timing to On-Chip Memory and Register Field

Section 3. Memory Map

Figure 3-1 shows a memory map of the H8/3101.



Notes: *1 User programs must not use these 4 bytes.
*2 ECR: EEPROM Control Register
EPR: EEPROM Protection Register
*3 DR: I/O Port Data Register
DDR: I/O Port Data Direction Register

Shaded areas are unavailable to the user.

Figure 3-1. Memory Map

Section 4. RAM

4.1 Overview

The H8/3101 has 256 bytes of on-chip static RAM.

The RAM is connected to the CPU by a 16-bit data bus. Both byte data and word data are accessed in two states, enabling rapid data transfer.

If word access is performed at an odd address in RAM, the word at the preceding even address is accessed. Normally an even address should be specified for word data.

4.1.1 Block Diagram

Figure 4-1 shows a block diagram of the RAM.

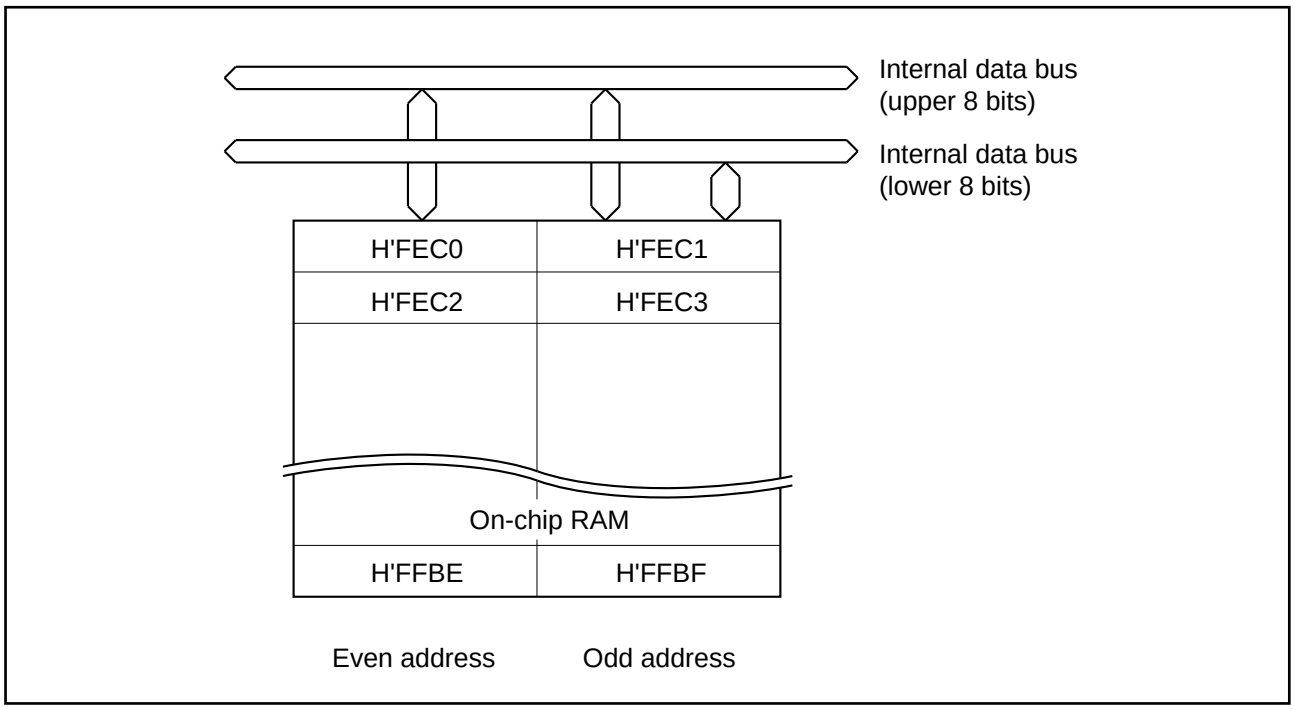


Figure 4-1. RAM Block Diagram

Section 5. ROM

5.1 Overview

The H8/3101 has 10 kbytes of on-chip user ROM. In addition, it has a separate 2-kbyte test program area. The ROM is connected to the CPU by a 16-bit data bus. Both byte and word data are accessed in two states, enabling rapid data transfer.

If word access is performed at an odd address in ROM, the word at the preceding even address is accessed. Normally an even address should be specified for word data.

5.1.1 Block Diagram

Figure 5-1 shows a block diagram of the ROM.

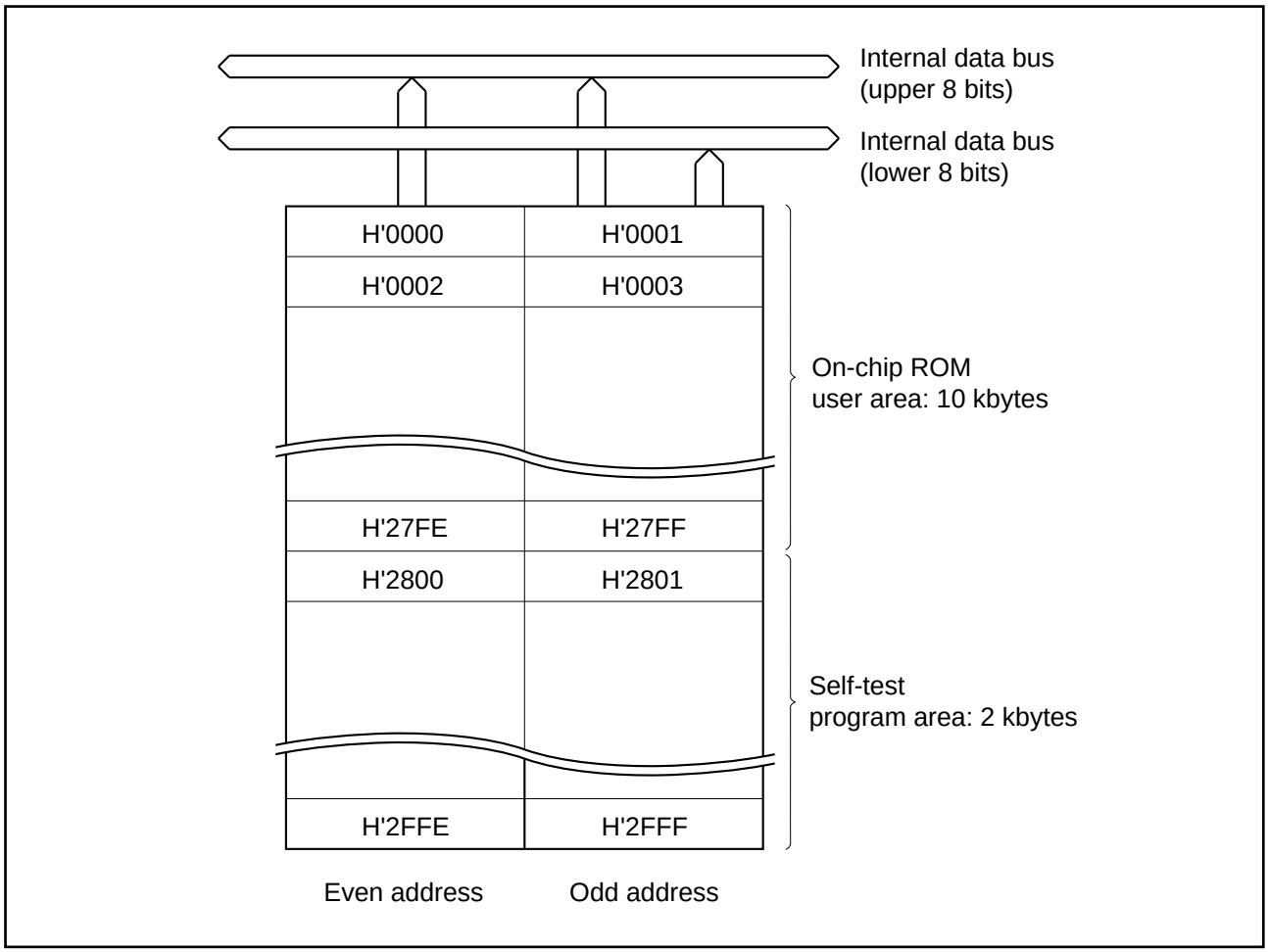


Figure 5-1. ROM Block Diagram

5.1.2 Security

ROM data are security-protected and cannot be read from outside the chip.

Section 6. EEPROM

6.1 Overview

The H8/3101 has 8 kbytes of electrically writable and erasable EEPROM on-chip. Both data and program code can be stored in the EEPROM.

6.1.1 Features

The features of the EEPROM are listed below.

- Capacity: 8 kbytes

  Organization: 32 bytes × 256 pages
  Allocated on the CPU address space

- Written by a special block data transfer instruction

  EEPMOV instruction: rewrites, overwrites, or erases a page (1 to 32 bytes) at a time.

- Protection features prevent accidental writing and erasing

  – Write/erase protection can be designated by protect bits.
  – Low voltage detection
  – Control registers prevent inadvertent writing and erasing.

- On-chip voltage pumping circuit

  Generates the high voltages required for writing and erasing

- Built-in oscillator and timer

  The write/erase sequence is controlled using an independent oscillator. EEPROM write/erase timing does not depend on the external clock.

- Rewrite time: 15 ms (max)

- Rewrite cycles: 10^4 (page rewrite)

- Data retention time: 10 years

6.1.2 Block Diagram

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

The built-in timer generates the write/erase sequence. The clock pulses for this timer are obtained from an on-chip oscillator and are independent of the CPU clock. Changing the CPU clock rate (external clock) does not affect the EEPROM write/erase timing.

The voltage pumping circuit generates the high voltages needed for writing and erasing. No external high-voltage power supply is required.

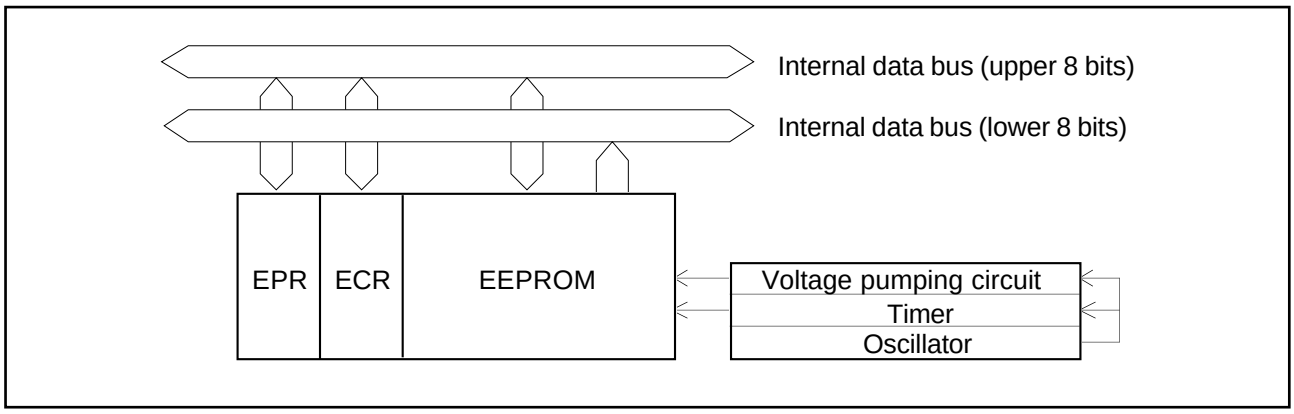


Figure 6-1. EEPROM Block Diagram

6.1.3 Memory Organization

The EEPROM has an 8192 × 8-bit organization, and is further organized into 32-byte pages. There are 256 pages as shown in figure 6-2.

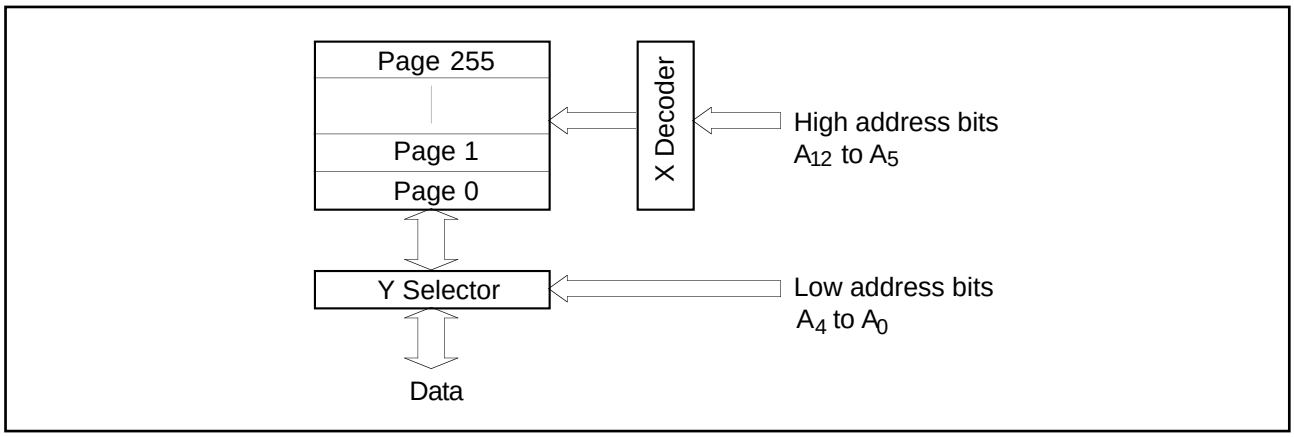


Figure 6-2. EEPROM Memory Organization

6.1.4 Register Configuration

Writing and erasing of the EEPROM are controlled by the registers listed in table 6-1.

Table 6-1. EEPROM Registers

Register	Abbr.	R/W	Initial Value	Address
EEPROM control register	ECR	R/W	H'FF	H'FFF8
EEPROM protection register	EPR	R/W	H'FF	H'FFF9

6.2 Register Descriptions

6.2.1 EEPROM Control Register (ECR)

The ECR is an 8-bit register that indicates power status and controls the type of write or erase operation performed on the EEPROM.

Bit:	7	6	5	4	3	2	1	0
	—	—	—	PWR	—	—	OC1	OC0
Initial value:				1			1	1
R/W:				R			R/W	R/W

Bits 7, 6, and 5—Reserved: These bits cannot be written and are always read as 1.

Although not used at present, reserved bits may be used in the future.

Bit 4—Power (PWR): This bit is set to 1 when an internal voltage drop is detected. The voltage drop detection function operates at all times, regardless of the operating state of the EEPROM.

The PWR bit can be read but not written. It is cleared to 0 when bit OC1 or OC0 is written.

Bits 3 and 2—Reserved: These bits cannot be written and are always read as 1.

Although not used at present, reserved bits may be used in the future.

Bits 1 and 0—Operation Control 1 and 0 (OC1 and OC0): These bits select the type of EEPROM write/erase operation.

Four operations can be selected by OC1 and OC0 as follows.

Bit 1 OC1	Bit 0 OC0	Description	
0	0	Rewrite	
0	1	Overwrite	
1	0	Page erase	
1	1	Write/erase disabled	(Initial value)

To prevent unintended writing and erasing, the OC1 and OC0 bits are both set to 1 automatically at a reset and at the end of a write or erase operation. They are also set to 1 automatically whenever an internal voltage drop is detected. It is accordingly necessary to clear one or both of these bits before every write or erase operation.

6.2.2 EEPROM Protection Register (EPR)

Bit:	7	6	5	4	3	2	1	0
	$\overline{\text{PBM}}$	—	—	—	—	—	—	—
Initial value:	1							
R/W:	R/W							

The EPR is an 8-bit register that enables the writing of EEPROM write/erase protect bits.

Bit 7—Protect Bit Mode ($\overline{\text{PBM}}$): This bit selects the EEPROM data area or protection area.

The protection area is selected when the $\overline{\text{PBM}}$ bit is cleared to 0. The data area is selected when the $\overline{\text{PBM}}$ bit is set to 1.

Writing the $\overline{\text{PBM}}$ bit automatically sets both the OC1 and OC0 bits in the ECR to 1, disabling writing or erasing of the EEPROM. At the end of a write or erase operation in the protection area, the $\overline{\text{PBM}}$ bit itself is automatically set to 1, selecting the data area.

The protect bits are allocated at the same addresses as the first bytes of pages 0 to 255 of the EEPROM data area. Each page of the EEPROM can be protected individually.

See section 6.5, Write/Erase Protection for further information on the protection area and data area.

Bit 7

$\overline{\text{PBM}}$	Description	
0	Protection area is selected	
1	Data area is selected	(Initial value)

Bits 6 to 0—Reserved: These bits cannot be written and are always read as 1.

Although not used at present, reserved bits may be used in the future.

ECR and EPR are initialized by the SLEEP instruction. After exiting sleep mode, software must set up these registers again before writing to EEPROM.

6.3 EEPROM Read Operation

The EEPROM is read directly by the CPU, using the same instructions as for reading ROM or RAM. Figure 6-3 shows the read timing. The read data are sent to the CPU via a 16-bit bus. If word access is performed at an odd address, the word at the preceding even address is read.

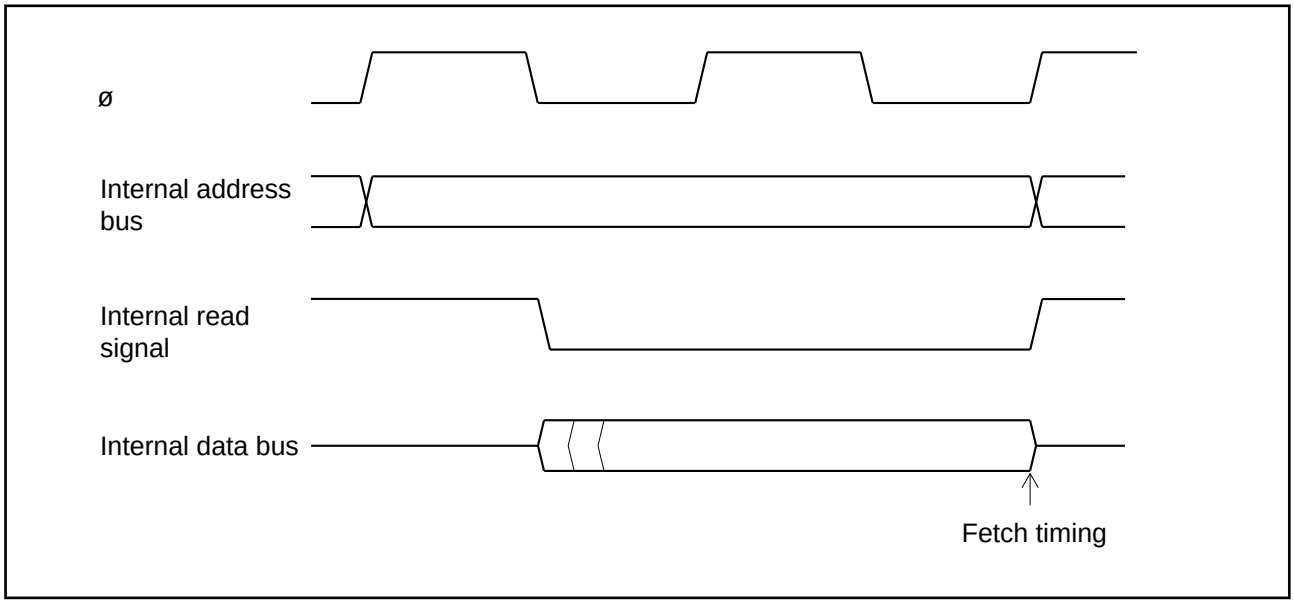


Figure 6-3. EEPROM Read Timing

6.4 EEPROM Write and Erase Operations

6.4.1 Write/Erase Sequence

The EEPROM is written or erased using the EEPMOV block data transfer instruction. The EEPMOV instruction transfers a block of data stored on RAM to a single page in EEPROM. The data transfer from RAM to EEPROM is controlled by parameters set in CPU registers R4L, R5, and R6 as shown in figure 6-4. The transfer is made by first setting parameters in registers R4L, R5, and R6 and control bits in the EPR and ECR, then executing the EEPMOV instruction.

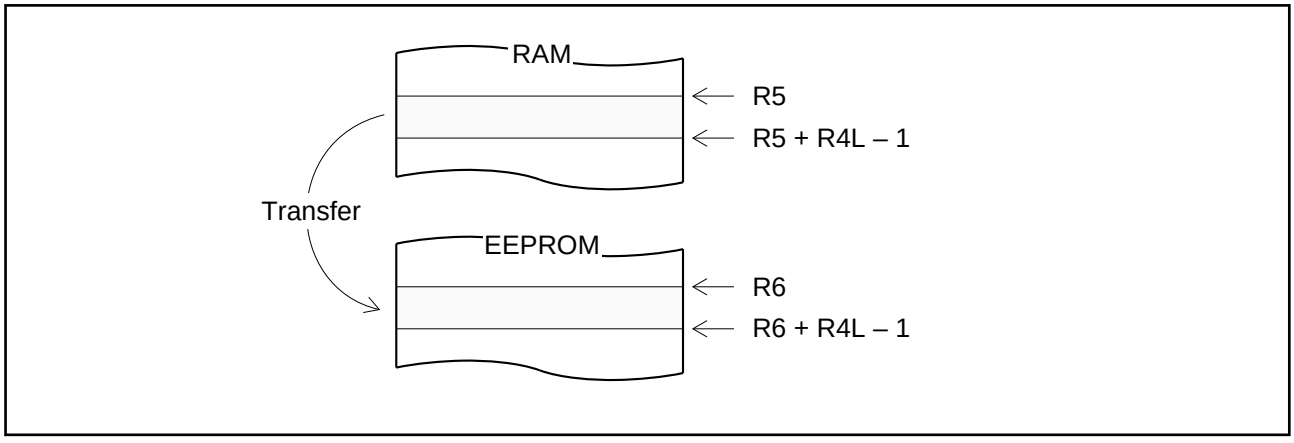


Figure 6-4. Block Transfer to EEPROM

Figure 6-5 indicates the contents of the three parameter registers used by the EEPMOV instruction. Table 6-2 describes the parameters and their valid ranges of values.

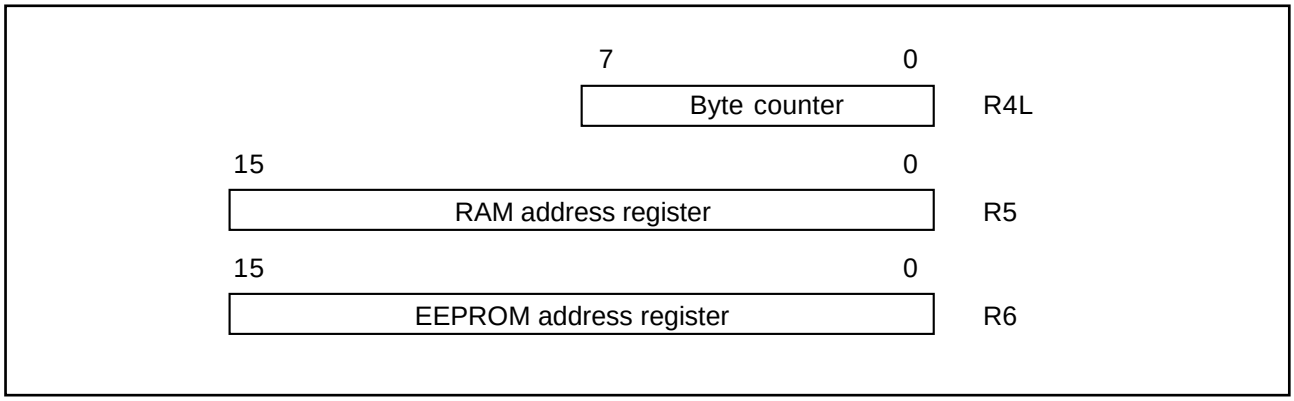


Figure 6-5. EEPMOV Parameters

Table 6-2. EEPMOV Parameters and Their Valid Ranges

Register	Name	Description	Valid Range	Final Value
R4L	Byte counter	Byte length of block to be written in EEPROM	1 to 32 (H'01 to H'20)	H'00
R5	RAM address register	Starting address of source block in RAM	H'FEC0 to H'FFBF	R5 + R4L
R6	EEPROM address register	Starting address of destination block in EEPROM	H'6000 to H'7FFF	R6 + R4L*

* When an EEPROM write operation ends at the last address on a page, the EEPROM address register (R6) reverts to the first address on that page.
Example 1: If R6 = H'6000 and R4L = H'20, the final value of R6 is H'6000.
Example 2: If R6 = H'603F and R4L = H'01, the final value of R6 is H'6020.

If the parameters are set to values outside the valid ranges in table 6-2 when the EEPMOV instruction is executed, or if the byte counter (R4L) and EEPROM address register (R6) are set so as to cross a page boundary, the write or erase operation may not be performed as intended. In addition, the final values left in the registers after instruction execution may not be the values indicated in table 6-2.

Figure 6-6 shows the sequence to be performed by software for writing or erasing the EEPROM.

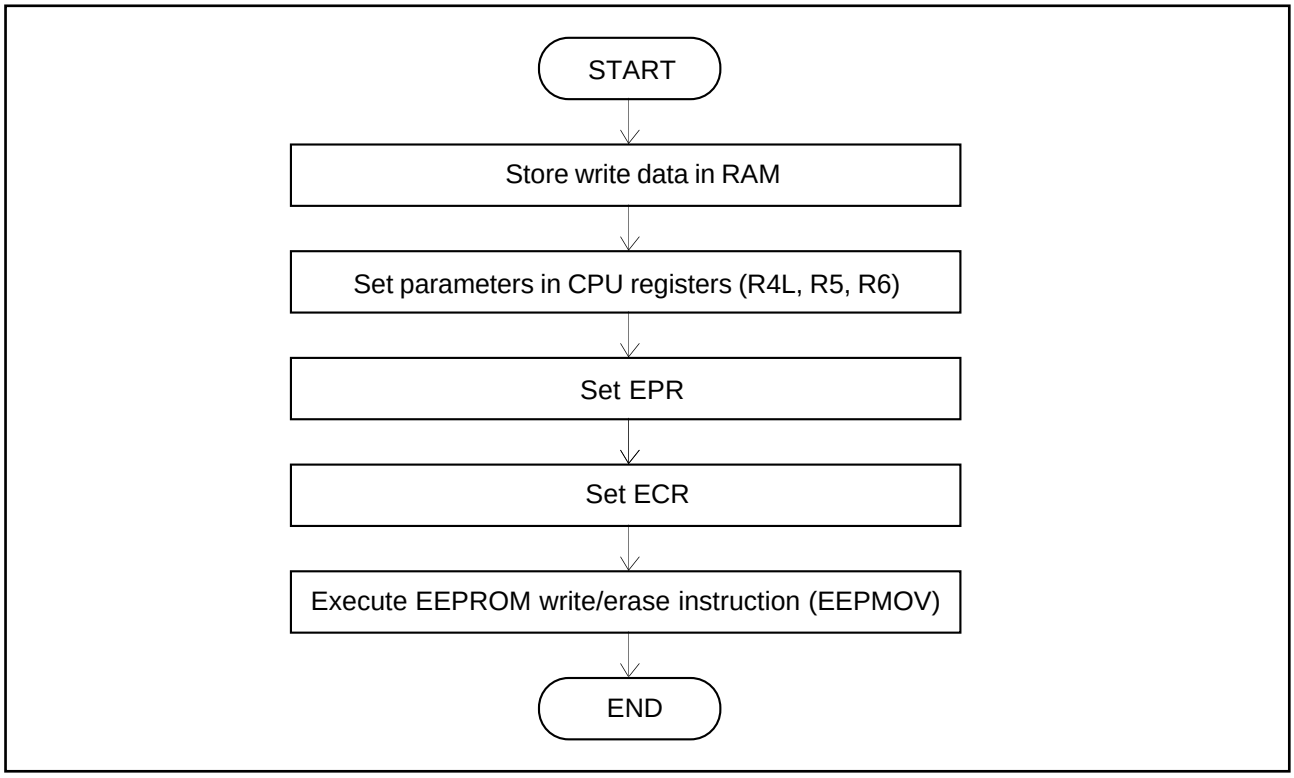


Figure 6-6. EEPROM Write/Erase Sequence

After an EEPMOV instruction, the CPU does not execute the next instruction until the writing or erasing of EEPROM data has ended.

EEPROM data cannot be written or erased by instructions other than EEPMOV.

6.4.2 Rewrite

A single rewrite operation can modify the values of 1 to 32 contiguous bytes located in the same EEPROM page.

A rewrite operation is restricted to a single page. The byte counter (R4L) and EEPROM address register (R6) should be set so that the operation does not cross a page boundary.

To perform a rewrite operation, clear both OC1 and OC0 to 0.

6.4.3 Erase

When the EEPMOV instruction is executed with OC1 = 1 and OC0 = 0, the relevant EEPROM page is erased.

The entire page containing the byte addressed by the EEPROM address register (R6) is erased. All data in the page are changed to 1. The byte counter (R4L) and RAM address register (R5) can be set to any valid values.

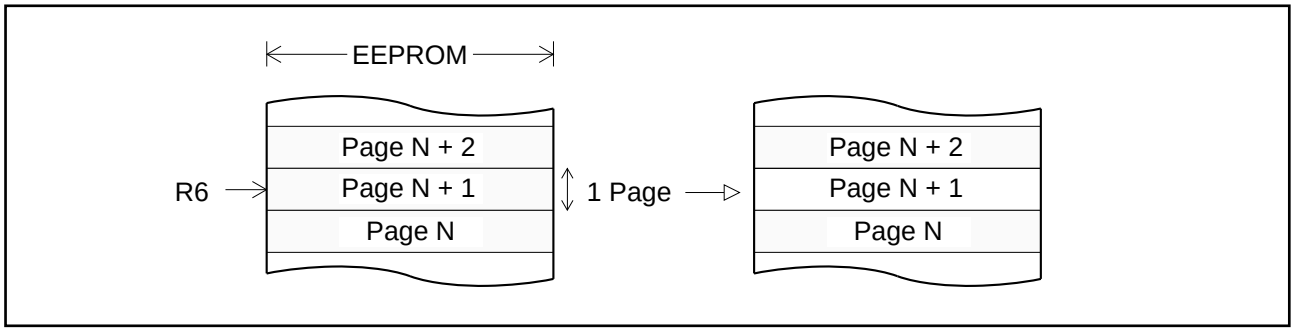


Figure 6-7. EEPROM Erase Operation

6.4.4 Overwrite

When the EEPMOV instruction is executed with OC1 = 0 and OC0 = 1, the transferred data are overwritten on the old data.

After an overwrite operation, the EEPROM contains the logical AND of the old data and the overwritten data.

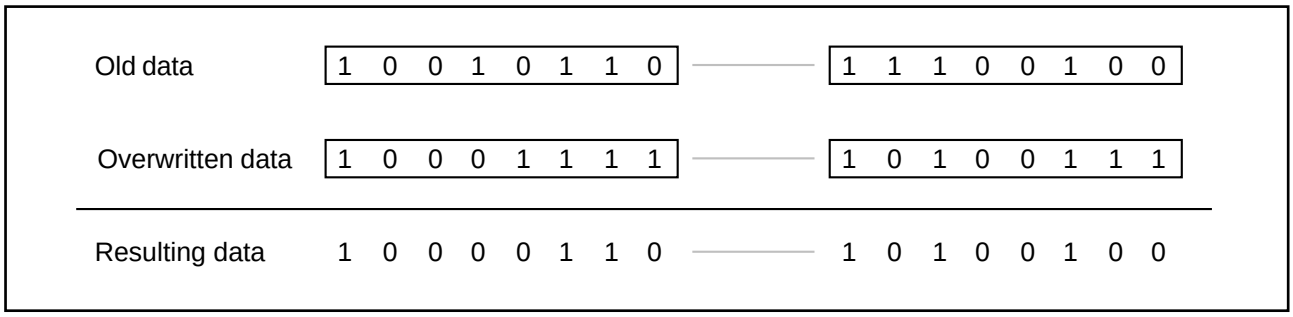


Figure 6-8. Results of Overwrite Operations (examples)

6.5 Write/Erase Protection

6.5.1 Protect Bits

EEPROM data can be protected from accidental writing and erasing. Each 32-byte page can be protected individually.

Each page has its own protect bits. Write/erase protection is conferred by writing a protection code (H'78) in the protect bits.

Once a page is protected, the protection cannot be removed.

The protect bits for a page have the same address as the first data byte in the page. The $\overline{\text{PBM}}$ bit in the EPR selects either the protection or data area: the protection area is selected when $\overline{\text{PBM}}$ = 0; the data area is selected when $\overline{\text{PBM}}$ = 1.

Figure 6-9 shows how the protect bits are allocated to pages. Figure 6-10 shows an example of write/erase protection.

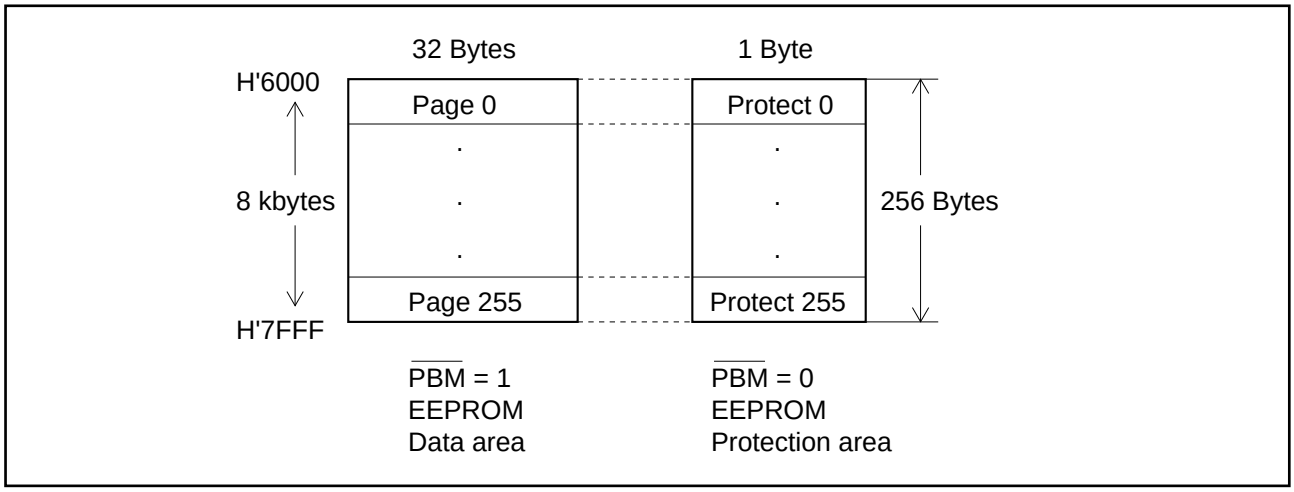


Figure 6-9. Allocation of Protect Bits

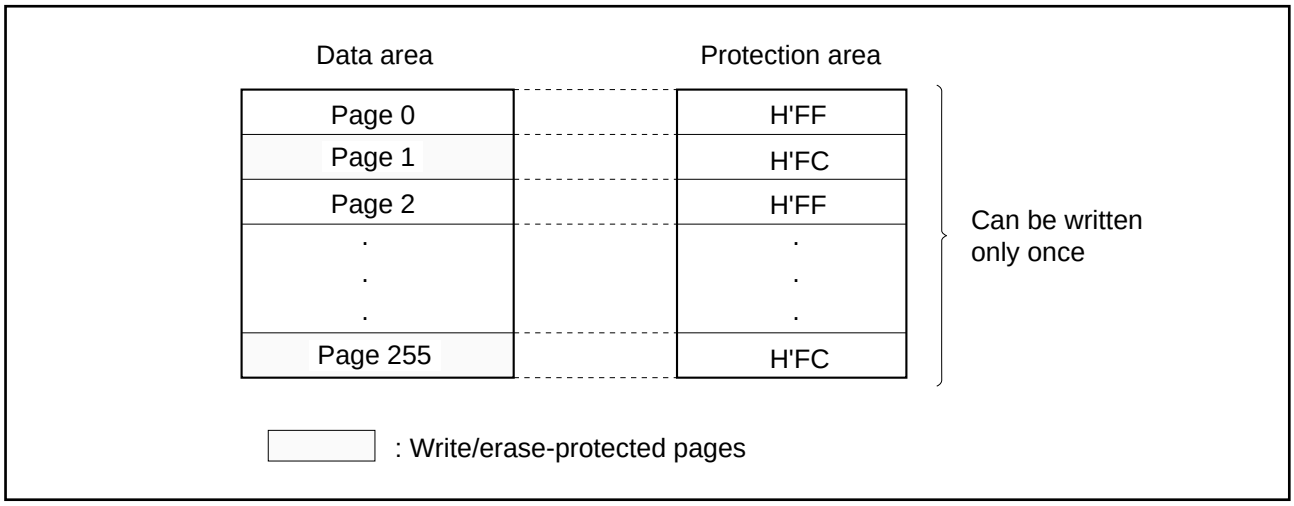


Figure 6-10. Example of Write/Erase Protection

6.5.2 Protection Procedure

To protect a page, software must set the EPR and ECR registers, then write the protection code (H'78) in the protect bits.

The protection procedure is given next. Figure 6-11 shows a flowchart.

1. Clear the $\overline{\text{PBM}}$ bit in the EPR to 0 to select the protection area. The OC1 and OC0 bits in the ECR will then be automatically set to 1, disabling EEPROM writing and erasing.

2. Clear the OC1 bit in the ECR to 0. The OC0 bit may be set to either 1 or 0.

3. Execute the EEPMOV instruction to write the protection code H'78 in the protect bits. The address of the protect bits is the same as the top byte address on the page to be protected.

After the protection code has been written, the EPR automatically reverts to select the data area, and the ECR is set to the write/erase-disabled state (OC1 = OC0 = 1).

Steps 1 to 3 must be carried out for each page protected.

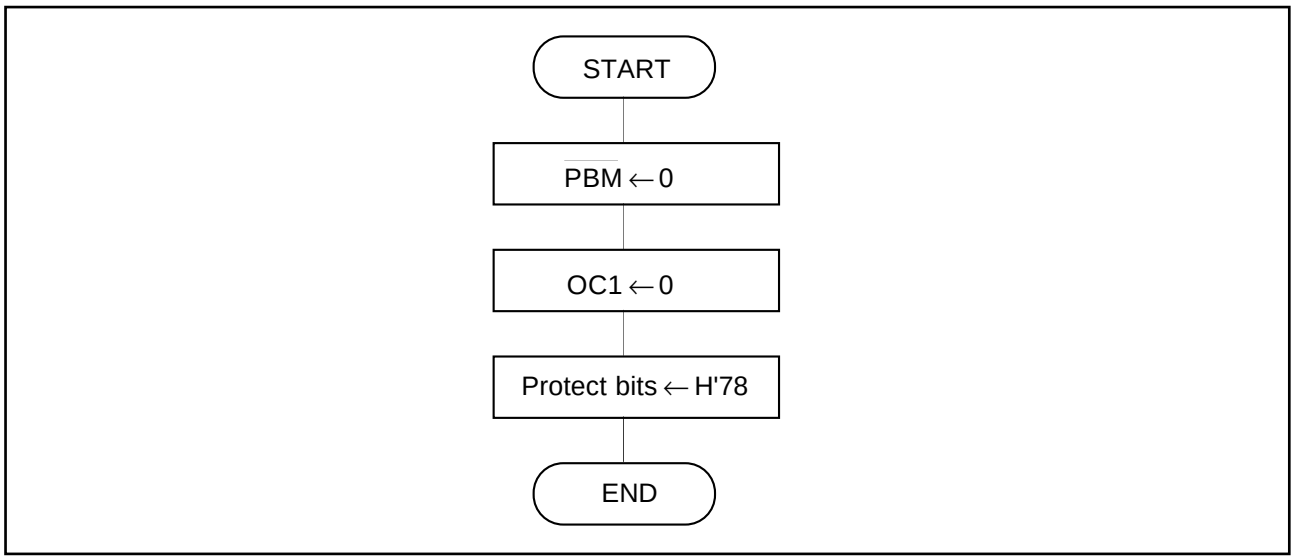


Figure 6-11. Protection Flowchart

6.5.3 Reading the Protect Bits

When the $\overline{\text{PBM}}$ bit in the EPR is cleared to 0, the protect bits can be read.

The protect bits for a protected page are read as H'FC. The protect bits for an unprotected page are read as H'FF.

6.6 Notes

When using the EEPROM, note the following points.

(1) Write/Erase Abort: The reset signal input to circuits other than the EEPROM, including the CPU, is synchronized with the system clock (ø) and processed to eliminate short pulses. The reset signal input to the EEPROM, however, is not synchronized with the system clock and is not processed to eliminate short pulses.

This feature enables the EEPROM to be deactivated by the reset signal alone, even when external clock input is stopped, to prevent EEPROM data from being destroyed due to chip malfunctions.

As a result, if a reset signal shorter than the minimum pulse width is input, only the EEPROM is reset (aborting any write or erase operation in progress and initializing the control registers). The CPU and other circuits may or may not continue to operate correctly.

(2) EEPMOV Execution with Invalid Register Settings: If registers R4L and R6 are set so as to cross a page boundary, the EEPROM write or erase operation is performed within the page including the initial address in R6.

Example: If:

R4L = H'20
R5 = H'FF00
R6 = H'6010

Then the block data transfer is performed as follows:

RAM addresses H'FF00 to H'FF0F → EEPROM addresses H'6010 to H'601F
RAM addresses H'FF10 to H'FF1F → EEPROM addresses H'6000 to H'600F

Section 7. I/O Port

7.1 Overview

The H8/3101 has a two-bit-wide I/O port. Software can select whether to use each I/O bit for data input or output.

The I/O port has a data register (DR) for latching output data, and a data direction register (DDR) for specifying input or output.

7.1.1 Block Diagram

Figure 7-1 shows an I/O port block diagram. The DR and DDR can be accessed only by byte access.

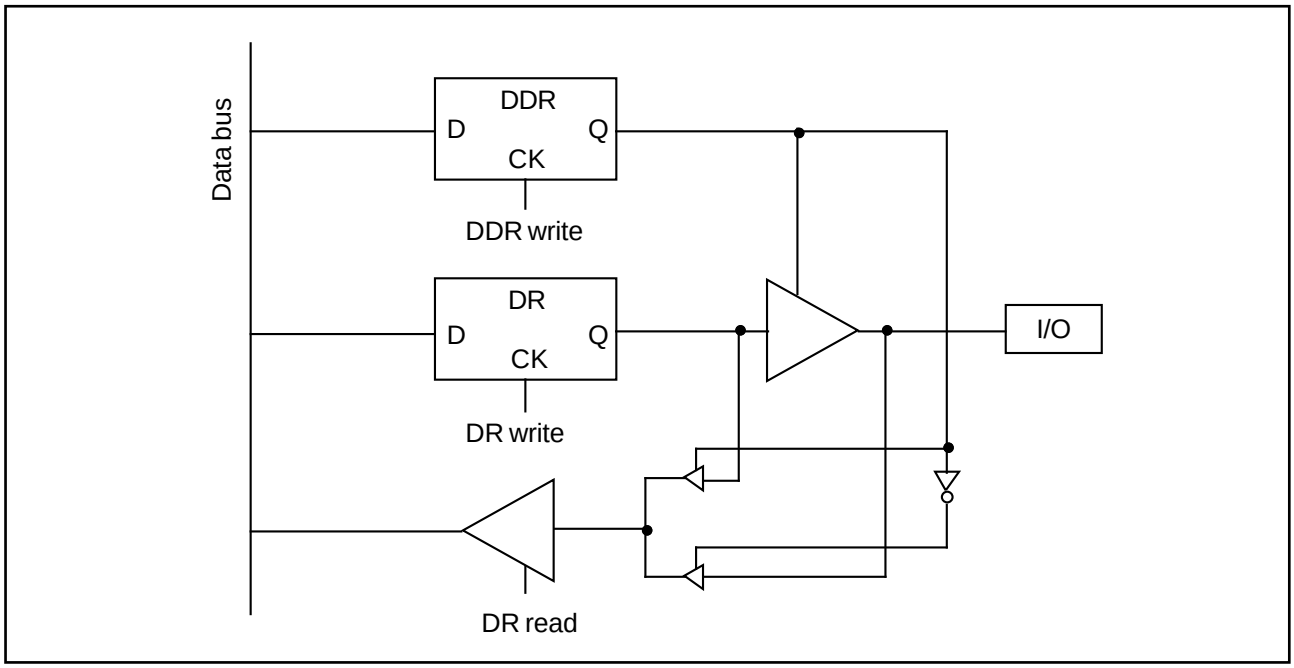


Figure 7-1. I/O Port Block Diagram

7.1.2 Register Configuration

Table 7-1 lists the I/O port registers.

Table 7-1. I/O Port Registers

Name	Abbr.	R/W	Address
Data register	DR	R/W	H'FFFE
Data direction register	DDR	W	H'FFFF

7.2 Register Descriptions

7.2.1 Data Register (DR)

Bit:	7	6	5	4	3	2	1	0
	DR7	DR6	—	—	—	—	—	—
Initial value:	—	—	—	—	—	—	—	—
R/W:	R/W	R/W	—	—	—	—	—	—

The data register latches the output data.

Bit 7—Data Register Bit 7 (DR7): Latches the I/O–1 output data. When DDR7 = 1 (selecting output), the value of the DR7 bit is output on the I/O–1 line.

When the DR is read, if DDR7 = 0 (input), the logic level of the I/O–1 line is read directly. If DDR7 = 1 (output), the value in the DR7 latch is read.

The value of DR7 after a reset is undetermined.

Bit 6—Data Register Bit 6 (DR6): Latches the I/O–2 output data. When DDR6 = 1 (selecting output), the value of the DR6 bit is output on the I/O–2 line.

When the DR is read, if DDR6 = 0 (input), the logic level of the I/O–2 line is read directly. If DDR6 = 1 (output), the value in the DR6 latch is read.

The value of DR6 after a reset is undetermined.

Bits 5 to 0—Reserved: These bits cannot be written, and are always read as 1.

Although not used at present, reserved bits may be used in the future.

7.2.2 Data Direction Register (DDR)

Bit:	7	6	5	4	3	2	1	0
	DDR7	DDR6	—	—	—	—	—	—
Initial value:	0	0	—	—	—	—	—	—
R/W:	W	W	—	—	—	—	—	—

The data direction register specifies the direction (input or output) of the I/O port.

Bit 7—Data Direction Bit 7 (DDR7): Specifies the direction of the I/O–1 line: 1 selects output; 0 selects input.

This bit can be written but not read. If read, it always returns the value 1, regardless of its true value.

A reset clears this bit to 0, making I/O–1 an input port.

Bit 6—Data Direction Bit 6 (DDR6): Specifies the direction of the I/O–2 line: 1 selects output; 0 selects input.

This bit can be written but not read. If read, it always returns the value 1, regardless of its true value.

A reset clears this bit to 0, making I/O–2 an input port.

Bits 5 to 0—Reserved: These bits cannot be written, and are always read as 1.

Although not used at present, reserved bits may be used in the future.

The DR and DDR contents are held in sleep mode as long as the necessary voltage is supplied, but the I/O ports are placed in the high-impedance state.

7.3 I/O Pad Timing

Figure 7-2 shows the output timing on the I/O lines. Figure 7-3 shows the input timing.

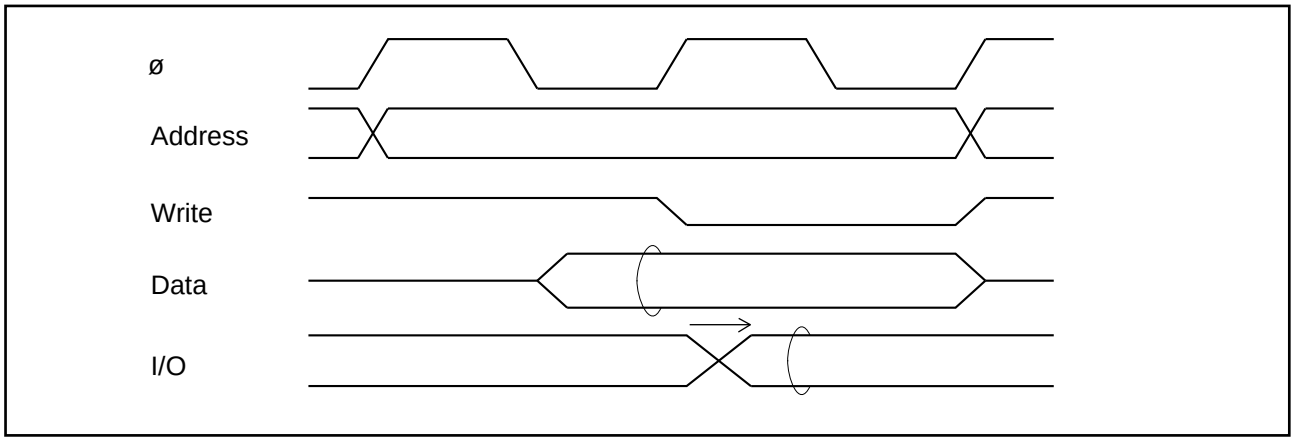


Figure 7-2. I/O Line Output Timing

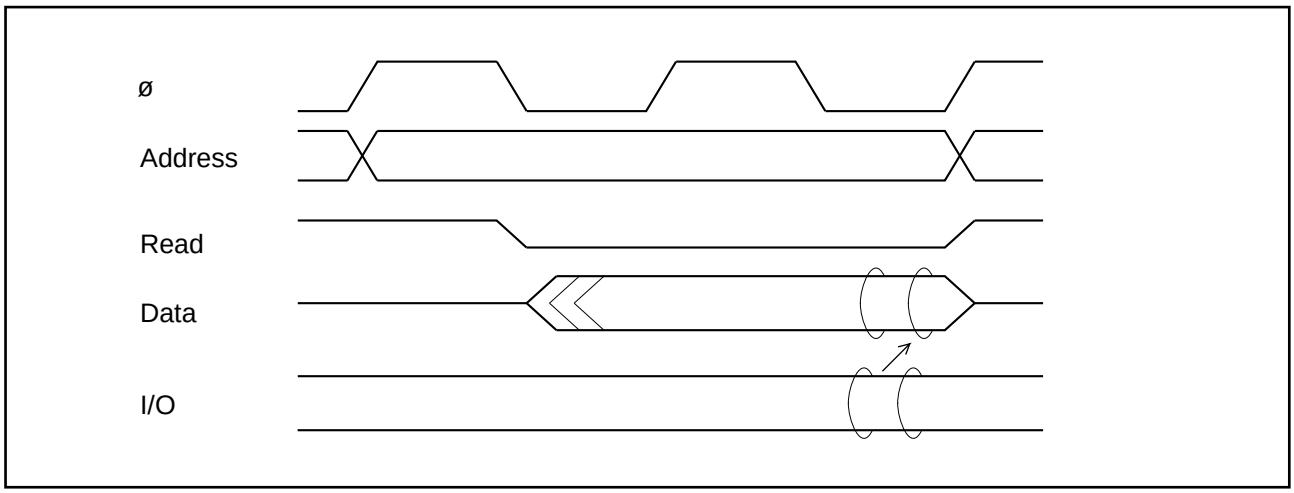


Figure 7-3. I/O Line Input Timing

Section 8. Clock Pulse Generator

8.1 Overview

The H8/3101 includes an on-chip divider circuit that generates the system clock (ø) from an external clock input. The external clock is input at the CLK pin.

The system clock frequency is one-half the external clock frequency.

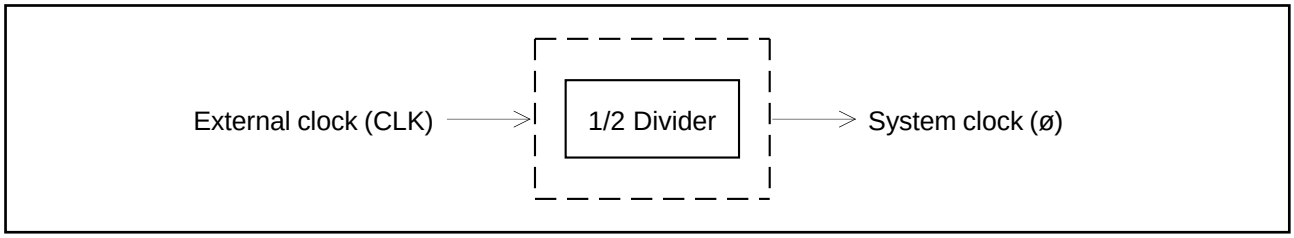


Figure 8-1. Block Diagram of Clock Pulse Generator

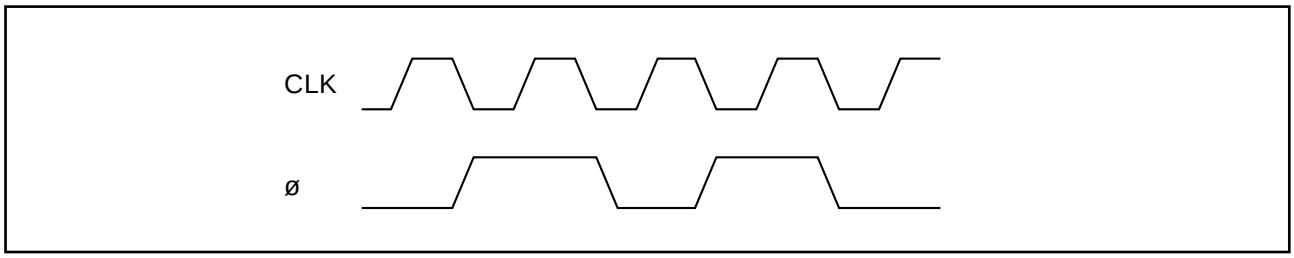


Figure 8-2. Relationship of System Clock and External Clock Input

Section 9. Electrical Characteristics

9.1 Absolute Maximum Ratings

Parameter	Symbol	Rating	Unit
Power supply voltage	V_{CC}	–0.3 to + 7.0	V
Input voltage	V_{in}	–0.3 to V_{CC} + 0.3	V
Operating temperature	T_{opr}	Regular specifications: –20 to 75	°C
		wide-range specifications: –40 to 85	
Storage temperature	T_{stg}*	–55 to +125	°C

* Before writing EEPROM

9.2 Electrical Characteristics

9.2.1 DC Characteristics

Conditions: V_{CC} = 5 V ±10%, V_{SS} = 0 V, T_a = –20 to 75°C (regular specifications),
T_a = –40 to 85°C (wide-range specifications) unless otherwise specified

Parameter		Symbol	Measurement Conditions	Min	Typ	Max	Unit
Input high level	$\overline{RES}$	V_{IH}		4.0	—	V_{CC} + 0.3	V
	CLK			2.4	—	V_{CC} + 0.3	
	I/O port			2.0	—	V_{CC} + 0.3	
Input low level	$\overline{RES}$	V_{IL}		–0.3	—	0.6	V
	CLK			–0.3	—	0.5	
	I/O port			–0.3	—	0.8	
Output high level		V_{OH}	I_{OH} = –100 µA	2.4	—	V_{CC}	V
			I_{OH} = –20 µA	3.8	—	V_{CC}	
Output low level		V_{OL}	I_{OL} = 1 mA	0	—	0.4	V
Input leakage current	CLK	$\|I_{in}\|$	V_{in} = 0.5 to	—	—	10	µA
	$\overline{RES}$		V_{CC} – 0.5 V	—	—	150	
Leakage current in 3-state (off state)	I/O	$\|I_{TSI}\|$	V_{in} = 0.5 to V_{CC} – 0.5 V	—	—	150	µA

Parameter		Symbol	Measurement Conditions	Min	Typ	Max	Unit
Current consumption*2	Normal operation	I_{CC}	f_{CLK}*1 = 10 MHz	—	—	20	mA
			f_{CLK}*1 = 5 MHz	—	—	10	
	Sleep	I_{CC}	—	—	—	100	µA
Pin capacitance		C_p	V_{in} = 0 V, f_{CLK} = 1 MHz, T_a = 25°C	—	—	20	pF

Notes: *1 f_{CLK} is the external clock frequency.
*2 These values assume that $V_{IHmin} = V_{CC} – 0.5$ V, $V_{ILmax} = 0.5$ V, and all output lines are unloaded.

9.2.2 AC Characteristics

Conditions: V_{CC} = 5 V ±10%, V_{SS} = 0 V, T_a = –20 to 75°C (regular specifications), T_a = –40 to 85°C (wide-range specifications) unless otherwise specified

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Clock cycle time	t_{cyc}	Figure 9-1	0.1	—	1.0	µs
Clock high width	t_{CH}	Figure 9-1	0.4	—	0.6	t_{cyc}
Clock low width	t_{CL}	Figure 9-1	0.4	—	0.6	t_{cyc}
Clock fall time	t_{cf}	Figure 9-1	—	—	10	ns
Clock rise time	t_{cr}	Figure 9-1	—	—	10	ns
I/O port fall time	t_f	Figure 9-2	—	—	1.0	µs
I/O port rise time	t_r	Figure 9-2	—	—	1.0	µs
$\overline{RES}$ pulse width	t_{RWL}	Figure 9-3	20	—	—	t_{cyc}
EEPROM write time	t_{EPW}		—	10	15	ms
Clock hold time	t_{CLKH}		20	—	—	t_{cyc}
Clock setup time	t_{CLKS}		20	—	—	t_{cyc}
Interrupt pulse width ($\overline{IRQ}$)	t_{IRQW}	Figure 9-4	200	—	—	ns

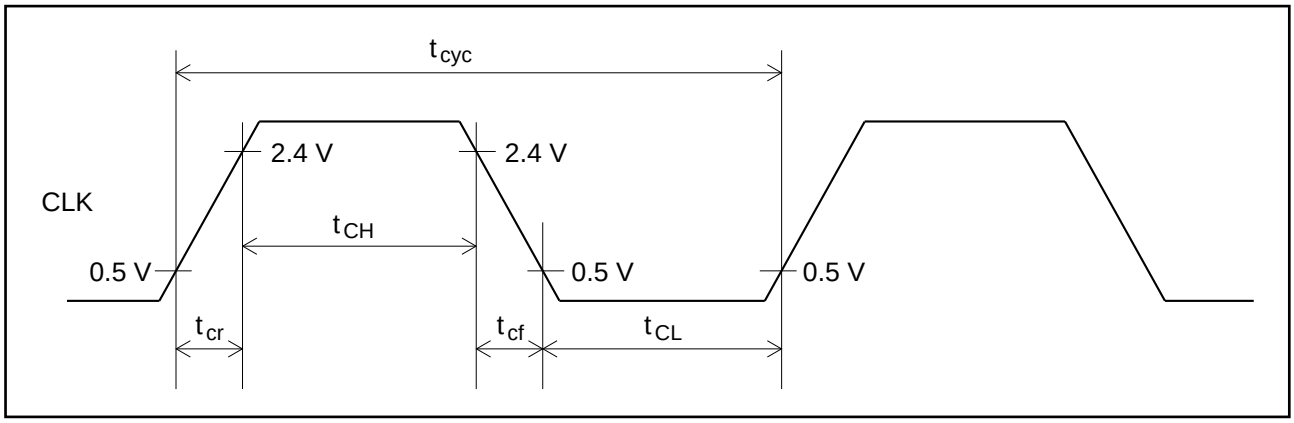


Figure 9-1. CLK Input Waveform

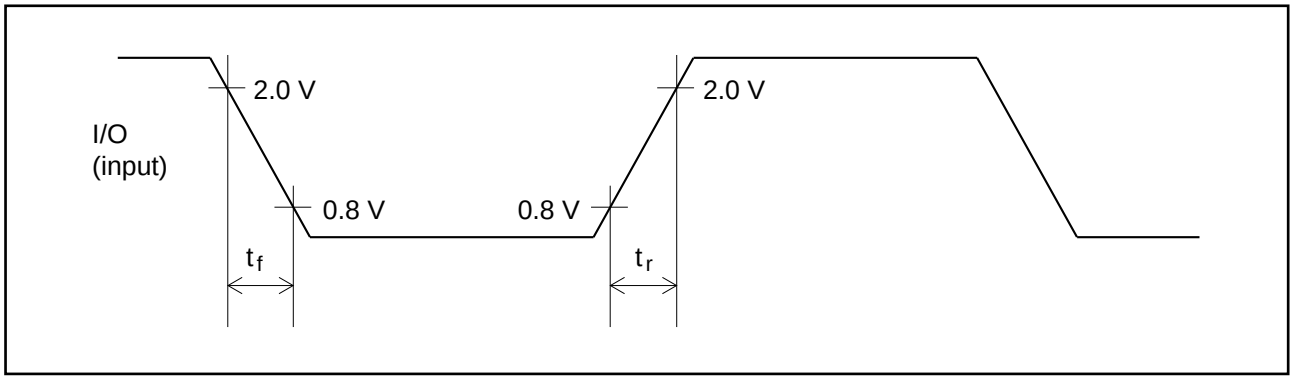


Figure 9-2. I/O Input Waveform

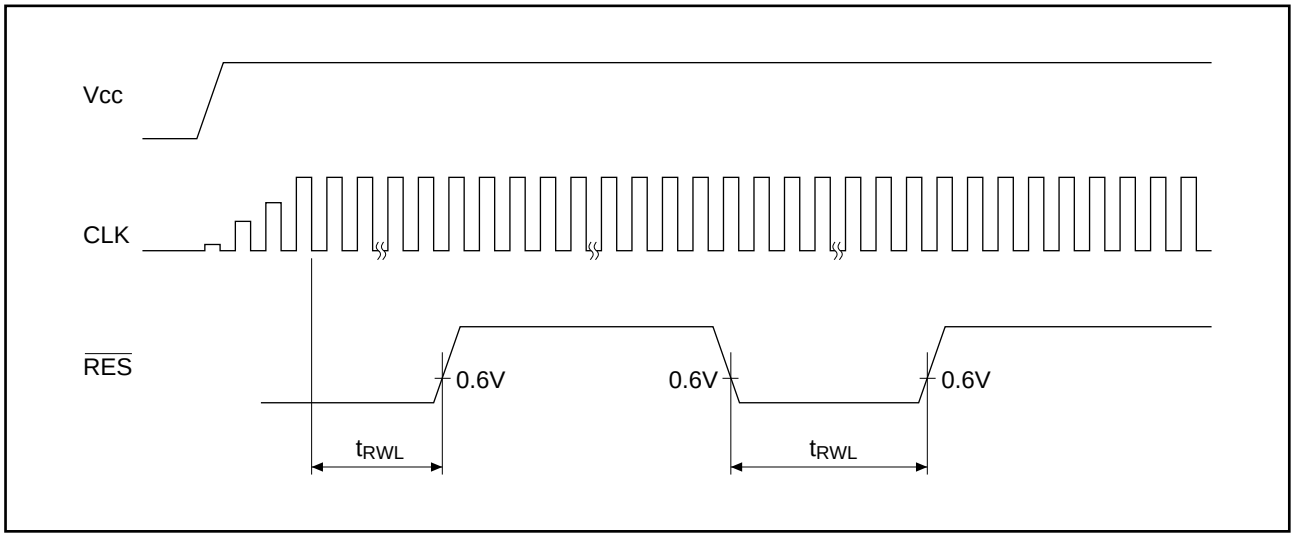


Figure 9-3. RESET Input Timing

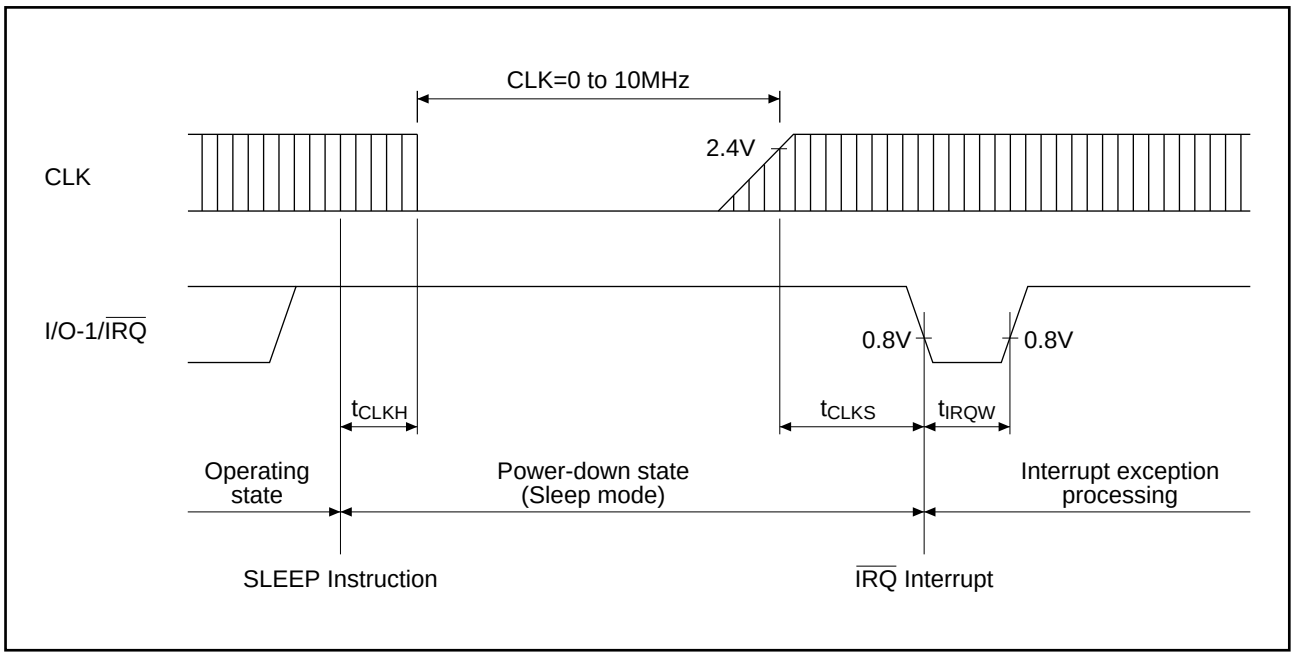


Figure 9-4. Interrupt Timing in Sleep Mode

Appendix A. Instruction Set

Operation Notation

Rd8/16	8- or 16-bit general register (destination)
Rs8/16	8- or 16-bit general register (source)
Rn8/16	8- or 16-bit general register
CCR	Condition code register
N	N (negative) flag of CCR
Z	Z (zero) flag of CCR
V	V (overflow) flag of CCR
C	C (carry) flag of CCR
PC	Program counter
SP	Stack pointer
#xx:3/8/16	3-, 8-, or 16-bit immediate data
d:8/16	8- or 16-bit displacement
@aa:8/16	8- or 16-bit absolute address
+	Addition
–	Subtraction
×	Multiplication
÷	Division
∧	AND logical
∨	OR logical
⊕	Exclusive OR logical
→	Move
¬	Not

Condition Code

Notation

↕	Changed according to the execution result
*	Undetermined value
0	Always cleared to 0
–	Previous value remains unchanged

Table A. Instruction Set

Mnemonic	Operand size	Operation	Addressing mode/ instruction length									Condition code						No. of states
			#xx:8/16	Rn	@Rn	@(d:16,Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8,PC)	@@aa	Implied	I	H	N	Z	V	C	
MOV.B #xx:8,Rd	B	#xx:8 → Rd8	2									–	–	↕	↕	0	–	2
MOV.B Rs,Rd	B	Rs8 → Rd8		2								–	–	↕	↕	0	–	2
MOV.B @Rs,Rd	B	@Rs16 → Rd8			2							–	–	↕	↕	0	–	4
MOV.B @(d:16,Rs),Rd	B	@(d:16,Rs16) → Rd8				4						–	–	↕	↕	0	–	6
MOV.B @Rs+,Rd	B	@Rs16 → Rd8 Rs16+1 → Rs16					2					–	–	↕	↕	0	–	6
MOV.B @aa:8,Rd	B	@aa:8 → Rd8						2				–	–	↕	↕	0	–	4
MOV.B @aa:16,Rd	B	@aa:16 → Rd8						4				–	–	↕	↕	0	–	6
MOV.B Rs,@Rd	B	Rs8 → @Rd16			2							–	–	↕	↕	0	–	4
MOV.B Rs,@(d:16,Rd)	B	Rs8 → @(d:16,Rd16)				4						–	–	↕	↕	0	–	6
MOV.B Rs,@–Rd	B	Rd16–1 → Rd16 Rs8 → @Rd16					2					–	–	↕	↕	0	–	6
MOV.B Rs,@aa:8	B	Rs8 → @aa:8						2				–	–	↕	↕	0	–	4
MOV.B Rs,@aa:16	B	Rs8 → @aa:16						4				–	–	↕	↕	0	–	6
MOV.W #xx:16,Rd	W	#xx:16 → Rd16	4									–	–	↕	↕	0	–	4
MOV.W Rs,Rd	W	Rs16 → Rd16		2								–	–	↕	↕	0	–	2
MOV.W @Rs,Rd	W	@Rs16 → Rd16			2							–	–	↕	↕	0	–	4
MOV.W @(d:16,Rs),Rd	W	@(d:16,Rs16) → Rd16				4						–	–	↕	↕	0	–	6
MOV.W @Rs+,Rd	W	@Rs16 → Rd16 Rs16+2 → Rs16					2					–	–	↕	↕	0	–	6
MOV.W @aa:16,Rd	W	@aa:16 → Rd16						4				–	–	↕	↕	0	–	6
MOV.W Rs,@Rd	W	Rs16 → @Rd16			2							–	–	↕	↕	0	–	4
MOV.W Rs,@(d:16,Rd)	W	Rs16 → @(d:16,Rd16)				4						–	–	↕	↕	0	–	6
MOV.W Rs,@–Rd	W	Rd16–2 → Rd16 Rs16 → @Rd16					2					–	–	↕	↕	0	–	6
MOV.W Rs, @aa:16	W	Rs16 → @aa:16						4				–	–	↕	↕	0	–	6
POP Rd	W	@SP → Rd16 SP+2 → SP					2					–	–	↕	↕	0	–	6
PUSH Rs	W	SP–2 → SP Rs16 → @SP					2					–	–	↕	↕	0	–	6

Table A. Instruction Set (cont)

Mnemonic	Operand size	Operation	Addressing mode/ instruction length								Condition code						No. of states
			#xx:8/16	Rn	@Rn	@(d:16,Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8,PC)	@@aa	I	H	N	Z	V	C	
ADD.B #xx:8,Rd	B	Rd8+#xx:8 → Rd8	2								–	↕	↕	↕	↕	↕	2
ADD.B Rs,Rd	B	Rs8+Rd8 → Rd8		2							–	↕	↕	↕	↕	↕	2
ADD.W Rs,Rd	W	Rs16+Rd16 → Rd16		2							–	[1]	↕	↕	↕	↕	2
ADDX.B #xx:8,Rd	B	Rd8+#xx:8 +C → Rd8	2								–	↕	↕	[2]	↕	↕	2
ADDX.B Rs,Rd	B	Rd8+Rs8 +C → Rd8		2							–	↕	↕	[2]	↕	↕	2
ADDS.W #1,Rd	W	Rd16+1 → Rd16		2							–	–	–	–	–	–	2
ADDS.W #2,Rd	W	Rd16+2 → Rd16		2							–	–	–	–	–	–	2
INC.B Rd	B	Rd8+1 → Rd8		2							–	–	↕	↕	↕	–	2
DAA.B Rd	B	Rd8 decimal adjust → Rd8		2							–	*	↕	↕	*	[3]	2
SUB.B Rs,Rd	B	Rd8–Rs8 → Rd8		2							–	↕	↕	↕	↕	↕	2
SUB.W Rs,Rd	W	Rd16–Rs16 → Rd16		2							–	[1]	↕	↕	↕	↕	2
SUBX.B #xx:8,Rd	B	Rd8–#xx:8 –C → Rd8	2								–	↕	↕	[2]	↕	↕	2
SUBX.B Rs,Rd	B	Rd8–Rs8 –C → Rd8		2							–	↕	↕	[2]	↕	↕	2
SUBS.W #1,Rd	W	Rd16–1 → Rd16		2							–	–	–	–	–	–	2
SUBS.W #2,Rd	W	Rd16–2 → Rd16		2							–	–	–	–	–	–	2
DEC.B Rd	B	Rd8–1 → Rd8		2							–	–	↕	↕	↕	–	2
DAS.B Rd	B	Rd8 decimal adjust → Rd8		2							–	*	↕	↕	*	–	2
NEG.B Rd	B	0–Rd → Rd		2							–	↕	↕	↕	↕	↕	2
CMP.B #xx:8,Rd	B	Rd8–#xx:8	2								–	↕	↕	↕	↕	↕	2
CMP.B Rs,Rd	B	Rd8–Rs8		2							–	↕	↕	↕	↕	↕	2
CMP.W Rs,Rd	W	Rd16–Rs16		2							–	[1]	↕	↕	↕	↕	2
MULXU.B Rs,Rd	B	Rd8×Rs8 → Rd16		2							–	–	–	–	–	–	14
DIVXU.B Rs,Rd	B	Rd16÷Rs8 → Rd16 (RdH:remainder,RdL:quotient)		2							–	–	[5]	[6]	–	–	14
AND.B #xx:8,Rd	B	Rd8∧#xx:8 → Rd8	2								–	–	↕	↕	0	–	2
AND.B Rs,Rd	B	Rd8∧Rs8 → Rd8		2							–	–	↕	↕	0	–	2
OR.B #xx:8,Rd	B	Rd8∨#xx:8 → Rd8	2								–	–	↕	↕	0	–	2
OR.B Rs,Rd	B	Rd8∨Rs8 → Rd8		2							–	–	↕	↕	0	–	2
XOR.B #xx:8,Rd	B	Rd8⊕#xx:8 → Rd8	2								–	–	↕	↕	0	–	2
XOR.B Rs,Rd	B	Rd8⊕Rs8 → Rd8		2							–	–	↕	↕	0	–	2
NOT.B Rd	B	$\overline{\text{Rd}}$ → Rd		2							–	–	↕	↕	0	–	2

Table A. Instruction Set (cont)

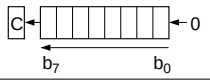
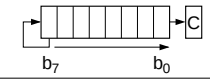
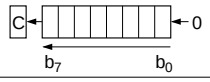
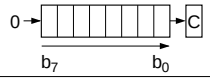
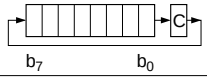
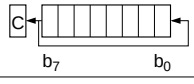
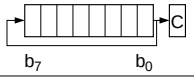
Mnemonic	Operand size	Operation	Addressing mode/ instruction length								Condition code						No. of states
			#xx:8/16	Rn	@Rn	@(d:16,Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8,PC)	@@aa	I	H	N	Z	V	C	
SHAL.B Rd	B			2							–	–	↕	↕	↕	↕	2
SHAR.B Rd	B			2							–	–	↕	↕	0	↕	2
SHLL.B Rd	B			2							–	–	↕	↕	0	↕	2
SHLR.B Rd	B			2							–	–	0	↕	0	↕	2
ROTXL.B Rd	B			2							–	–	↕	↕	0	↕	2
ROTXR.B Rd	B			2							–	–	↕	↕	0	↕	2
ROTL.B Rd	B			2								–	↕	↕	0	↕	2
ROTR.B Rd	B			2							–	–	↕	↕	0	↕	2
BSET #xx:3,Rd	B	(#xx:3 of Rd8) ← 1		2							–	–	–	–	–	–	2
BSET #xx:3,@Rd	B	(#xx:3 of @Rd16) ← 1			4						–	–	–	–	–	–	8
BSET #xx:3,@aa:8	B	(#xx:3 of @aa:8) ← 1						4			–	–	–	–	–	–	8
BSET Rn,Rd	B	(Rn8 of Rd8) ← 1		2							–	–	–	–	–	–	2
BSET Rn,@Rd	B	(Rn8 of @Rd16) ← 1			4						–	–	–	–	–	–	8
BSET Rn,@aa:8	B	(Rn8 of @aa:8) ← 1						4			–	–	–	–	–	–	8
BCLR #xx:3,Rd	B	(#xx:3 of Rd8) ← 0		2							–	–	–	–	–	–	2
BCLR #xx:3,@Rd	B	(#xx:3 of @Rd16) ← 0			4						–	–	–	–	–	–	8
BCLR #xx:3,@aa:8	B	(#xx:3 of @aa:8) ← 0						4			–	–	–	–	–	–	8
BCLR Rn,Rd	B	(Rn8 of Rd8) ← 0		2							–	–	–	–	–	–	2
BCLR Rn,@Rd	B	(Rn8 of @Rd16) ← 0			4						–	–	–	–	–	–	8
BCLR Rn,@aa:8	B	(Rn8 of @aa:8) ← 0						4			–	–	–	–	–	–	8
BNOT #xx:3,Rd	B	(#xx:3 of Rd8) ← ($\overline{\text{#xx:3 of Rd8}}$)		2							–	–	–	–	–	–	2
BNOT #xx:3,@Rd	B	(#xx:3 of @Rd16) ← ($\overline{\text{#xx:3 of @Rd16}}$)			4						–	–	–	–	–	–	8
BNOT #xx:3,@aa:8	B	(#xx:3 of @aa:8) ← ($\overline{\text{#xx:3 of @aa:8}}$)						4			–	–	–	–	–	–	8

Table A. Instruction Set (cont)

Mnemonic	Operand size	Operation	Addressing mode/ instruction length								Condition code						No. of states
			#xx:8/16	Rn	@Rn	@(d:16,Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8,PC)	@@aa	I	H	N	Z	V	C	
BNOT Rn,Rd	B	(Rn8 of Rd8) ← ($\overline{\text{Rn8}}$ $\overline{\text{of}}$ $\overline{\text{Rd8}}$)		2							–	–	–	–	–	–	2
BNOT Rn,@Rd	B	(Rn8 of @Rd16) ← ($\overline{\text{Rn8}}$ $\overline{\text{of}}$ $\overline{\text{@Rd16}}$)			4						–	–	–	–	–	–	8
BNOT Rn,@aa:8	B	(Rn8 of @aa:8) ← ($\overline{\text{Rn8}}$ $\overline{\text{of}}$ $\overline{\text{@aa:8}}$)						4			–	–	–	–	–	–	8
BTST #xx:3,Rd	B	($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{Rd8}}$) → Z		2							–	–	–	↕	–	–	2
BTST #xx:3,@Rd	B	($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{@Rd16}}$) → Z			4						–	–	–	↕	–	–	6
BTST #xx:3,@aa:8	B	($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{@aa:8}}$) → Z						4			–	–	–	↕	–	–	6
BTST Rn,Rd	B	($\overline{\text{Rn8}}$ $\overline{\text{of}}$ $\overline{\text{Rd8}}$) → Z		2							–	–	–	↕	–	–	2
BTST Rn,@Rd	B	($\overline{\text{Rn8}}$ $\overline{\text{of}}$ $\overline{\text{@Rd16}}$) → Z			4						–	–	–	↕	–	–	6
BTST Rn,@aa:8	B	($\overline{\text{Rn8}}$ $\overline{\text{of}}$ $\overline{\text{@aa:8}}$) → Z						4			–	–	–	↕	–	–	6
BLD #xx:3,Rd	B	(#xx:3 of Rd8) → C		2							–	–	–	–	–	↕	2
BLD #xx:3,@Rd	B	(#xx:3 of @Rd16) → C			4						–	–	–	–	–	↕	6
BLD #xx:3,@aa:8	B	(#xx:3 of @aa:8) → C						4			–	–	–	–	–	↕	6
BILD #xx:3,Rd	B	($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{Rd8}}$) → C		2							–	–	–	–	–	↕	2
BILD #xx:3,@Rd	B	($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{@Rd16}}$) → C			4						–	–	–	–	–	↕	6
BILD #xx:3,@aa:8	B	($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{@aa:8}}$) → C						4			–	–	–	–	–	↕	6
BST #xx:3,Rd	B	C → (#xx:3 of Rd8)		2							–	–	–	–	–	–	2
BST #xx:3,@Rd	B	C → (#xx:3 of @Rd16)			4						–	–	–	–	–	–	8
BST #xx:3,@aa:8	B	C → (#xx:3 of @aa:8)						4			–	–	–	–	–	–	8
BIST #xx:3,Rd	B	$\overline{\text{C}}$ → (#xx:3 of Rd8)		2							–	–	–	–	–	–	2
BIST #xx:3,@Rd	B	$\overline{\text{C}}$ → (#xx:3 of @Rd16)			4						–	–	–	–	–	–	8
BIST #xx:3,@aa:8	B	$\overline{\text{C}}$ → (#xx:3 of @aa:8)						4			–	–	–	–	–	–	8
BAND #xx:3,Rd	B	C∧(#xx:3 of Rd8) → C		2							–	–	–	–	–	↕	2
BAND #xx:3,@Rd	B	C∧(#xx:3 of @Rd16) → C			4						–	–	–	–	–	↕	6
BAND #xx:3,@aa:8	B	C∧(#xx:3 of @aa:8) → C						4			–	–	–	–	–	↕	6
BIAND #xx:3,Rd	B	C∧($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{Rd8}}$) → C		2							–	–	–	–	–	↕	2
BIAND #xx:3,@Rd	B	C∧($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{@Rd16}}$) → C			4						–	–	–	–	–	↕	6
BIAND #xx:3, @aa:8	B	C∧($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{@aa:8}}$) → C						4			–	–	–	–	–	↕	6
BOR #xx:3,Rd	B	C∨(#xx:3 of Rd8) → C		2							–	–	–	–	–	↕	2
BOR #xx:3,@Rd	B	C∨(#xx:3 of @Rd16) → C			4						–	–	–	–	–	↕	6
BOR #xx:3,@aa:8	B	C∨(#xx:3 of @aa:8) → C						4			–	–	–	–	–	↕	6
BIOR #xx:3,Rd	B	C∨($\overline{\text{#xx:3}}$ $\overline{\text{of}}$ $\overline{\text{Rd8}}$) → C		2							–	–	–	–	–	↕	2

Table A. Instruction Set (cont)

Mnemonic	Operand size	Operation		Addressing mode/ instruction length								Condition code						No. of states
				#xx:8/16	Rn	@Rn	@(d:16,Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8,PC)	@@aa	I	H	N	Z	V	C	
BIOR #xx:3,@Rd	B	C∨($\overline{\#xx:3}$ $\overline{\text{of}}$ $\overline{@Rd16}$) → C				4						–	–	–	–	–	↕	6
BIOR #xx:3, @aa:8	B	C∨($\overline{\#xx:3}$ $\overline{\text{of}}$ $\overline{@aa:8}$) → C							4			–	–	–	–	–	↕	6
BXOR #xx:3,Rd	B	C⊕(#xx:3 of Rd8) → C			2							–	–	–	–	–	↕	2
BXOR #xx:3,@Rd	B	C⊕(#xx:3 of @Rd16) → C				4						–	–	–	–	–	↕	6
BXOR #xx:3, @aa:8	B	C⊕(#xx:3 of @aa:8) → C							4			–	–	–	–	–	↕	6
BIXOR #xx:3,Rd	B	C⊕($\overline{\#xx:3}$ $\overline{\text{of}}$ $\overline{Rd8}$) → C			2							–	–	–	–	–	↕	2
BIXOR #xx:3,@Rd	B	C⊕($\overline{\#xx:3}$ $\overline{\text{of}}$ $\overline{@Rd16}$) → C				4						–	–	–	–	–	↕	6
BIXOR #xx:3, @aa:8	B	C⊕($\overline{\#xx:3}$ $\overline{\text{of}}$ $\overline{@aa:8}$) → C							4			–	–	–	–	–	↕	6
BRA d:8 (BTd:8)	–	PC ← PC+d:8								2		–	–	–	–	–	–	4
BRNd:8 (BFd:8)	–	PC ← PC+2								2		–	–	–	–	–	–	4
BHI d:8	–	if true then	C∨Z=0							2		–	–	–	–	–	–	4
BLS d:8	–	PC ← PC+d:8	C∨Z=1							2		–	–	–	–	–	–	4
BCC d:8 (BHS d:8)	–	else next	C=0							2		–	–	–	–	–	–	4
BCS d:8 (BLO d:8)	–		C=1							2		–	–	–	–	–	–	4
BNE d:8	–		Z=0							2		–	–	–	–	–	–	4
BEQ d:8	–		Z=1							2		–	–	–	–	–	–	4
BVC d:8	–		V=0							2		–	–	–	–	–	–	4
BVS d:8	–		V=1							2		–	–	–	–	–	–	4
BPL d:8	–		N=0							2		–	–	–	–	–	–	4
BMI d:8	–		N=1							2		–	–	–	–	–	–	4
BGE d:8	–		N⊕V=0							2		–	–	–	–	–	–	4
BLT d:8	–		N⊕V=1							2		–	–	–	–	–	–	4
BGT d:8	–		Z∨(N⊕V)=0							2		–	–	–	–	–	–	4
BLE d:8	–		Z∨(N⊕V)=1							2		–	–	–	–	–	–	4
JMP @Rn	–	PC ← Rn16				2						–	–	–	–	–	–	4
JMP @aa:16	–	PC ← aa:16							4			–	–	–	–	–	–	6
JMP @@aa:8	–	PC ← @aa:8									2	–	–	–	–	–	–	8
BSR d:8	–	SP–2 → SP PC → @SP PC ← PC+d:8								2		–	–	–	–	–	–	6
JSR @Rn	–	SP–2 → SP PC → @SP PC ← Rn16				2						–	–	–	–	–	–	6

Table A. Instruction Set (cont)

Mnemonic	Operand size	Operation	Addressing mode/ instruction length									Condition code						No. of states
			#xx:8/16	Rn	@Rn	@(d:16,Rn)	@-Rn/@Rn+	@aa:8/16	@(d:8,PC)	@@aa	Implied	I	H	N	Z	V	C	
JSR @aa:16	–	SP–2 → SP PC → @SP PC ← aa:16						4				–	–	–	–	–	–	8
JSR @@aa:8	–	SP–2 → SP PC → @SP PC ← @aa:8								2		–	–	–	–	–	–	8
RTS	–	PC ← @SP SP+2 → SP									2	–	–	–	–	–	–	8
RTE	–	CCR ← @SP SP+2 → SP PC ← @SP SP+2 → SP									2	↕	↕	↕	↕	↕	↕	10
SLEEP	–	Transit to sleep mode.									2	–	–	–	–	–	–	2
LDC #xx:8,CCR	B	#xx:8 → CCR	2									↕	↕	↕	↕	↕	↕	2
LDC Rs,CCR	B	Rs8 → CCR		2								↕	↕	↕	↕	↕	↕	2
STC CCR,Rd	B	CCR → Rd8		2								–	–	–	–	–	–	2
ANDC #xx:8,CCR	B	CCR∧#xx:8 → CCR	2									↕	↕	↕	↕	↕	↕	2
ORC #xx:8,CCR	B	CCR∨#xx:8 → CCR	2									↕	↕	↕	↕	↕	↕	2
XORC #xx:8,CCR	B	CCR⊕#xx:8 → CCR	2									↕	↕	↕	↕	↕	↕	2
NOP	–	PC ← PC+2									2	–	–	–	–	–	–	2
EEPMOV	–	if R4L≠0 then Repeat @R5 → @R6 R5+1 → R5 R6+1 → R6 R4L–1 → R4L Until R4L=0 else next									4	–	–	–	–	–	–	[4]

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

[1] Set to 1 when there is a carry or borrow from bit 11; otherwise cleared to 0.
[2] If the result is zero, the previous value of the flag is retained; otherwise the flag is cleared to 0.
[3] Set to 1 if decimal adjustment produces a carry; otherwise cleared to 0.
[4] The maximum write time is 15 ms.
[5] Set to 1 when the divisor is negative; otherwise cleared to 0.
[6] Set to 1 when the divisor is zero; otherwise cleared to 0.

Appendix B. Operation Code Map

Table B is a map of the operation codes contained in the first byte of the instruction code (bits 15 to 8 of the first instruction word).

Some pairs of instructions have identical first bytes. These instructions are differentiated by the first bit of the second byte (bit 7 of the first instruction word).

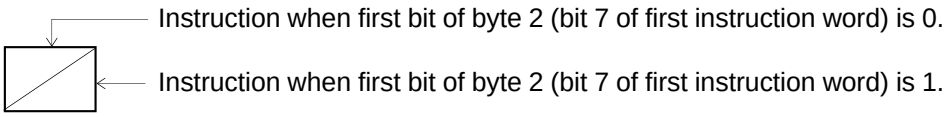
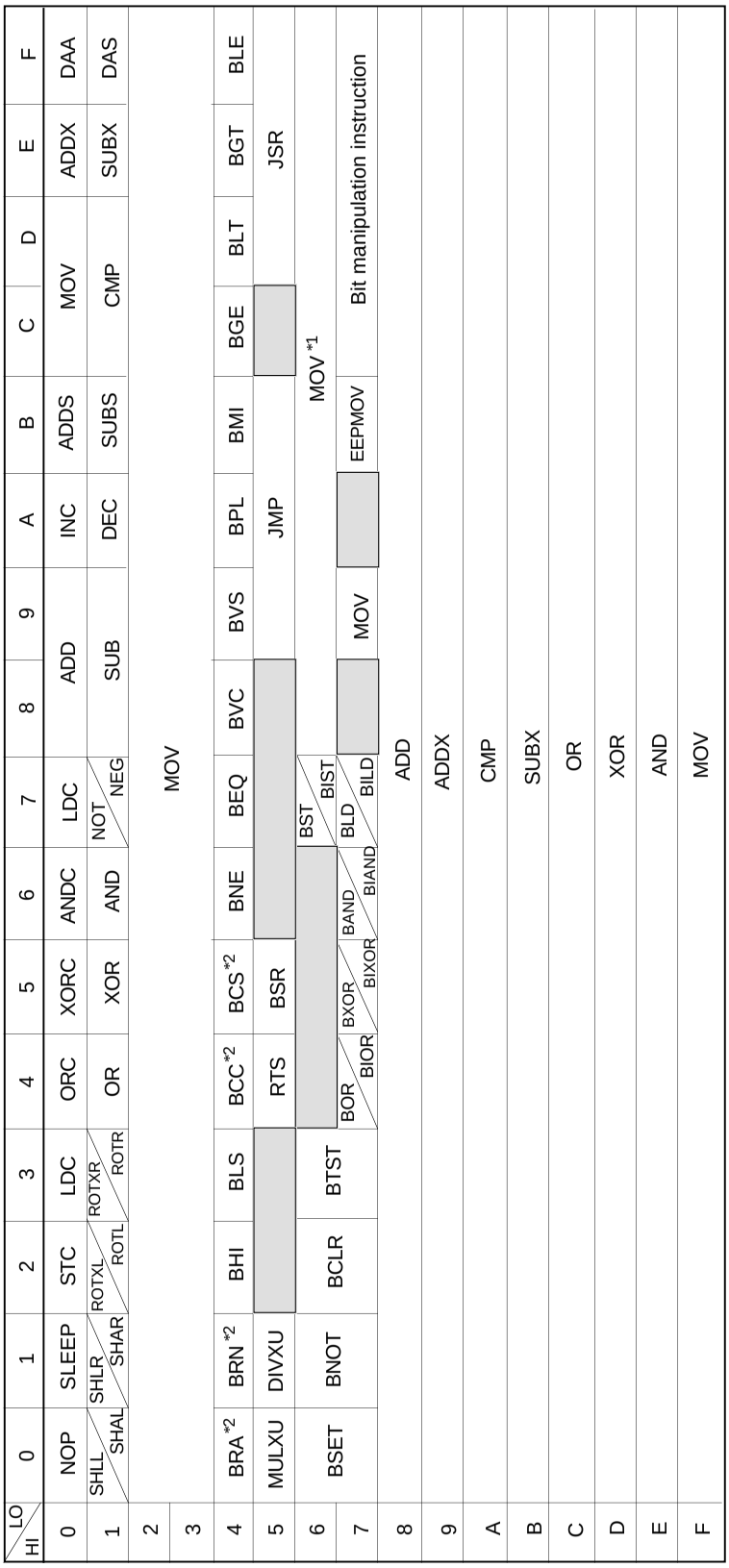


Table B. Operation Code Map



Notes: *1 The PUSH and POP instructions are identical to MOV instructions

*2 The BT, BF, BHS, and BLO instructions are identical to BRA, BRN, BCC, and BCS, respectively.

Appendix C. Register Field

C.1 Register Field (1)

Address	Register Name	Bit Names								Module
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
H'FFF8	ECR	—	—	—	PWR	—	—	OC1	OC0	EEPROM
H'FFF9	EPR	$\overline{\text{PBM}}$	—	—	—	—	—	—	—	
H'FFFE	DR	DR7	DR6	—	—	—	—	—	—	I/O port
H'FFFF	DDR	DDR7	DDR6	—	—	—	—	—	—	

C.2 Register Field (2)

ECR EEPROM Control Register **EEPROM**

Bit:	7	6	5	4	3	2	1	0
	—	—	—	PWR	—	—	OC1	OC0
Initial value:				1			1	1
R/W:				R			R/W	R/W

Operation Control 1 and 0

0	0	Rewrite
0	1	Overwrite
1	0	Page erase
1	1	Write/erase disabled

Power Bit

Set to 1 when supply voltage is abnormal

EPR EEPROM Protect Register **EEPROM**

Bit:	7	6	5	4	3	2	1	0
	$\overline{\text{PBM}}$	—	—	—	—	—	—	—
Initial value:	1							
R/W:	R/W							

Protect Bit Mode

0	Protection area
1	Data area

DR Data Register **I/O**

Bit:	7	6	5	4	3	2	1	0
	DR7	DR6	—	—	—	—	—	—
Initial value:	—	—						
R/W:	R/W	R/W						

Data Register Bit 6
Output data latch

Data Register Bit 7
Output data latch

DDR Data Direction Register **I/O**

Bit:	7	6	5	4	3	2	1	0
	DDR7	DDR6	—	—	—	—	—	—
Initial value:	0	0						
R/W:	W	W						

Data Direction Register Bit 6
Selects the input/output direction of I/O–2

0	Input
1	Output

Data Direction Register Bit 7
Selects the input/output direction of I/O–1

0	Input
1	Output

Appendix D. External Dimensions

Figure D-1 shows the dimensions of the SOP-10 package of the H8/3101. Figure D-2 shows the standard COB pattern.

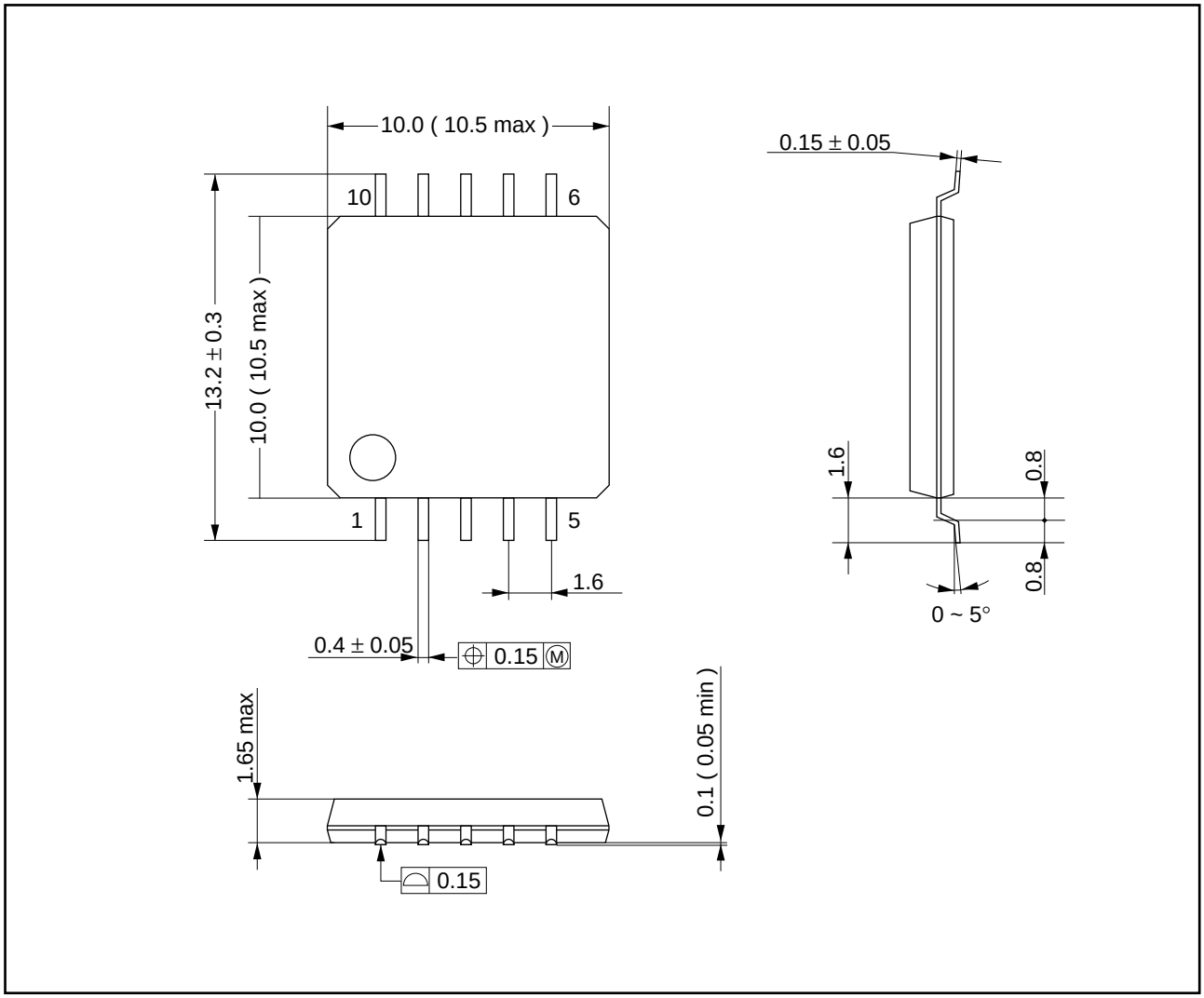


Figure D-1. SOP-10 Package Dimensions (unit: mm)

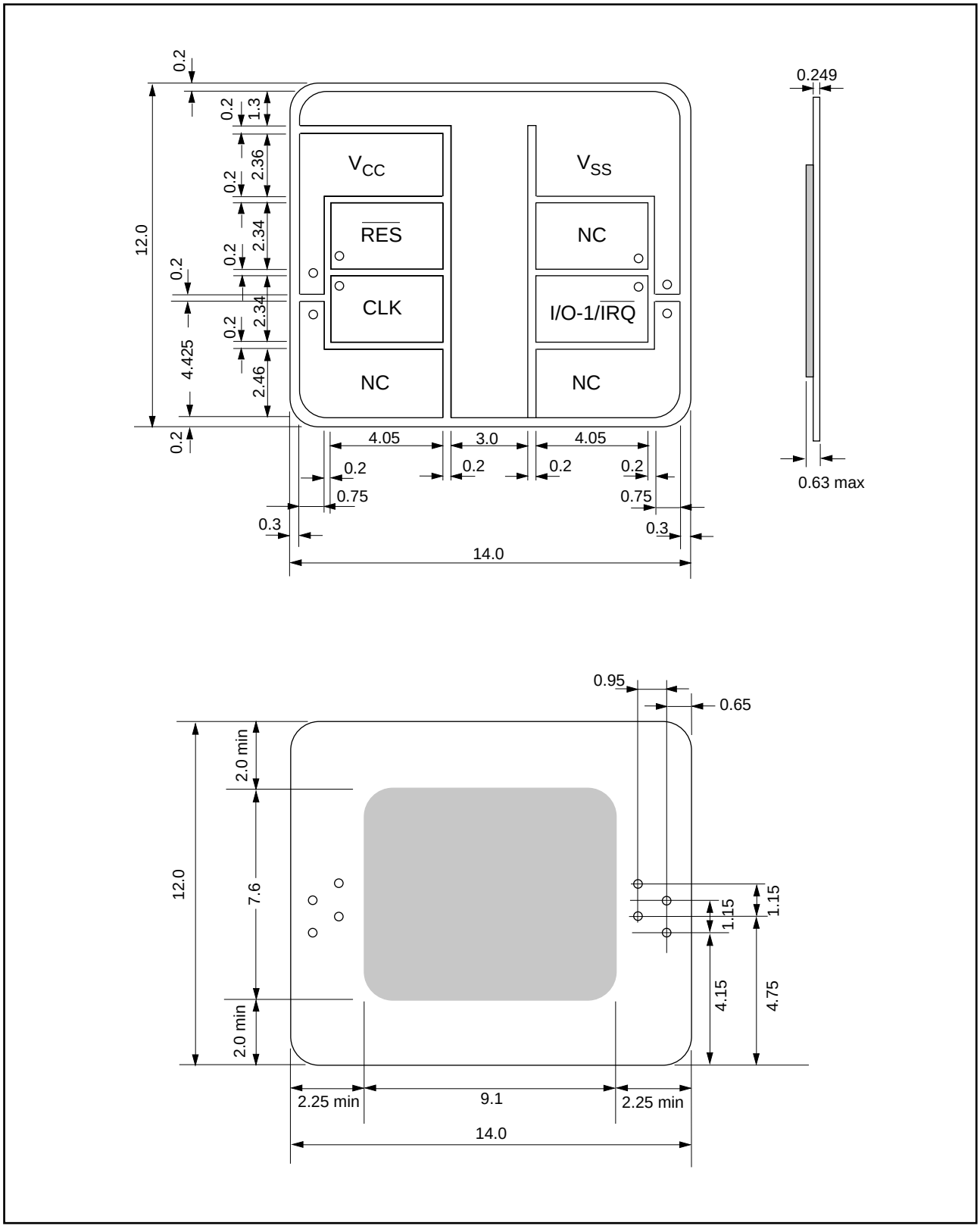


Figure D-2. Standard COB Pattern (unit: mm)