# HD65901
# MCU (Microcomputer Unit)

HD65901 is a CMOS 8-bit microcomputer, with integrated CPU, 2k-bytes of EEPROM, 3k-bytes of ROM, 128 bytes of RAM, and one Input/Output line on a single chip.

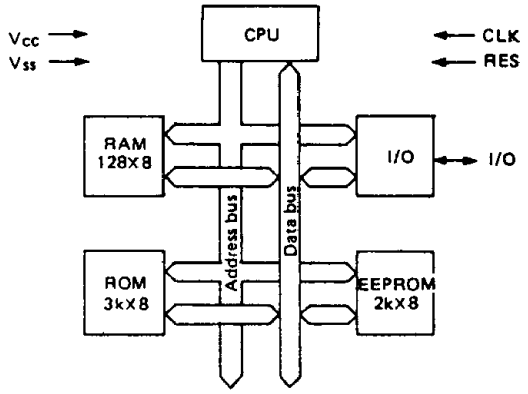
## ■ FEATURES

- CMOS EEPROM Process
- 8-bit CPU
  - —27 Instructions
  - —Five Addressing Modes:
    Immediate, Register Direct, Register Indirect, Relative, Implied Register
  - —Internal Registers:
    Program Counter (PC), General Purpose Register ($R_0$–$R_{15}$)
    Condition Code Register (CCR: N, Z, and C flags)
- Memory
  - —3k-byte ROM
  - —128 byte RAM
  - —2k-byte EEPROM
  - —EEPROM is located in internal memory address space
  - —Byte write/erase
  - —Page write/erase
  - —Single 5V Power Supply
- General Purpose I/O
  One I/O Line
- Data Security
  - —ROM Data Security
  - —EEPROM Data Security
- Operating Range
  f = 3 to 10 MHz ($V_{CC}$ = 5V ± 10%)
- Available in die form for smart card applications.

## ■ PROGRAM DEVELOPOMENT SUPPORT TOOLS

- Cross assembler and simulator software for use with IBM PC or compatibles
- Hardware evaluation board

## ■ BLOCK DIAGRAM



# Section 1. Overview

This MCU is a CMOS 8-bit microcomputer, integrating a CPU, 2k bytes of EEPROM, 3k bytes of ROM, 128 bytes of RAM, and one I/O line on a single chip.

## 1.1 Features

. CMOS EEPROM Process

. 8-bit CPU

27 instructions

Five addressing modes;

Immediate, Register direct, Register indirect, Relative, Implied register

Internal Registers;

Program Counter (PC), General Purpose Register (R0-R15)

Condition Code Register (CCR: N, Z, and C flags)

. Memory

3k-byte ROM

128-byte RAM

. EEPROM

2k-byte EEPROM

EEPROM is located in internal memory address space

Byte erase/write

Page erase/write

Single 5V power supply

. General purpose I/O

One I/O line

. Data Security

ROM data security

EEPROM data security

. Operating range

f = 3 to 5 MHz ($V_{CC}$ = 5V ± 10%)

## 1.2 Block Diagram

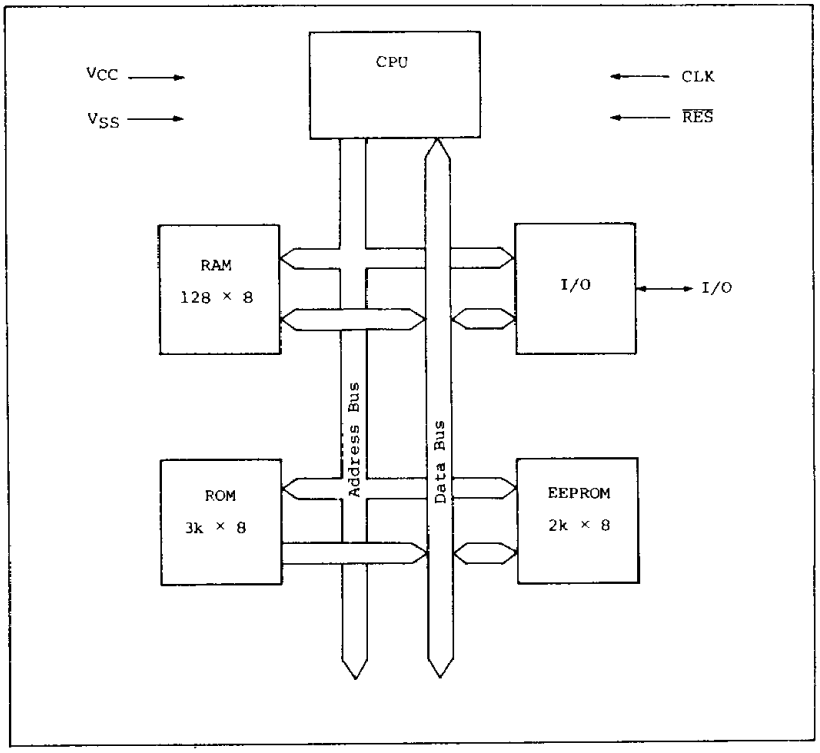
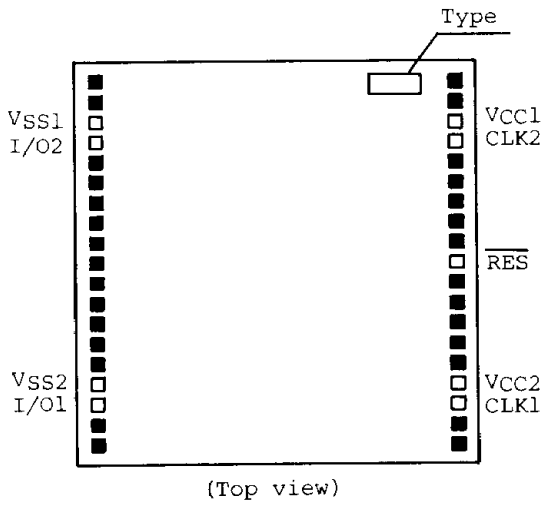



Figure 1-1 Block Diagram

## 1.3 Pad Arrangement




Note: (1) For each VCC, VSS and CLK, one of the pins or each should be connected. For I/O, one of the pins but not the both should be connected.

(2) Pins other than the ones specified above should be open.

(3) The substrate of each die should be connected to VCC or floated.

Figure 1-2 Pad Arrangement

## 1.4 Pin Description

Table 1-1 Pin Description

| Pin | Symbol | Input/Output | Pin Name and Function |
|---|---|---|---|
| Power Supply | $V_{CC}$ | | Power supply: Connected to Power. (5V±10%) |
| GND | $V_{SS}$ | | Ground: Tied to ground. (0V) |
| Clock | CLK | Input | Clock: External clock is input. |
| Reset | $\overline{RES}$ | Input | Reset: When asserted LOW, initializes the MCU*. |
| Port | I/O | Input/Output | I/O port: 1-bit I/O line. Can be softwarely programmed as input or output. |

* MCU: Microcomputer Unit

# Section 2. CPU Architecture

## 2.1 CPU Architecture

This section explains the CPU architecture.

### 2.1.1 CPU registers

The CPU contains sixteen 8-bit General Purpose Registers (R0-R15), a 3-bit Condition Code Register (CCR) and a 16-bit Program Counter (PC). The CPU register configuration is shown in Figure 2-1.

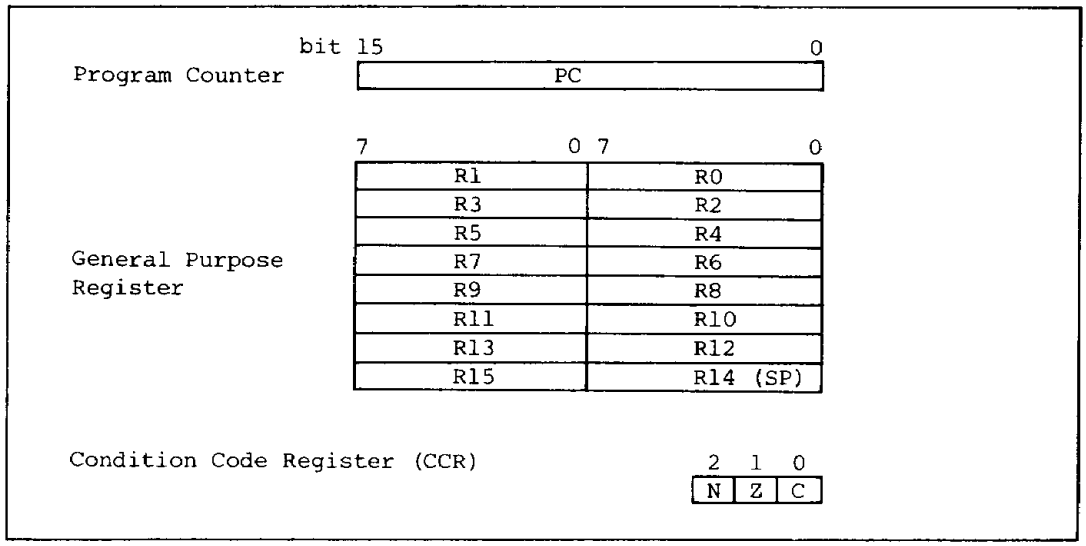



Figure 2-1 CPU Registers

### 2.1.2 Addressing mode

The CPU has five addressing modes as follows.

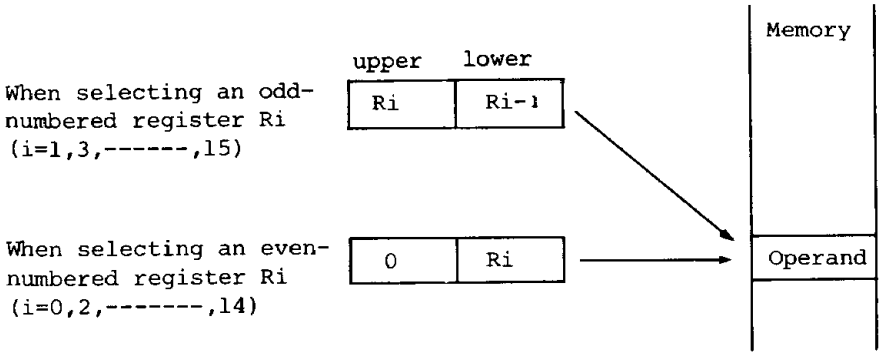
Register direct: The operand is located in one of the 8-bit or 16-bit General Purpose Registers specified by the register field in the op-code.

8-bit register : R0 - R15
16-bit register pair: R1R0 - R15R14

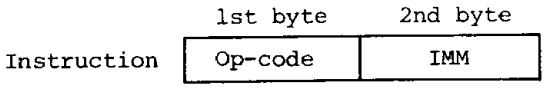
When selecting a 16-bit register pair in ADDD and SUBD, the odd-numbered register should be selected by the register field. For example, when selecting R1R0, R1 should be selected by the register field.

Register indirect: The effective address is located in one of the 8-bit or 16-bit General Purpose Register.

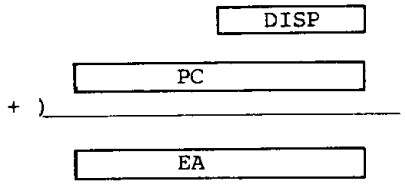
When selecting an odd-numbered register Ri (i=1, 3, -----, 15), an effective address is contained in the 16-bit register pair RiRi-1.
When selecting an even-numbered register Ri (i=0, 2, ------, 14), the lower byte of the effective address is contained in the 8-bit register Ri and the upper byte of that becomes 0. Therefore, the effective address ranges from $0000 to $00FF. In this case, register Ri+1 can be used as a data register.




Immediate: An 8-bit operand is included in the 2nd byte of the instruction.




Relative: The relative addressing mode is used by jump and call instructions. An effective address is calculated by summing the PC and a 8-bit signed displacement in the 2nd byte of the instruction. The relative addressing span is from -126 to +129 from the op-code address.




Implied Register: Certain op-codes automatically imply register usage, such as CTR which inherently reference the CPU register.

The CPU register functions are summarized in Table 2-1.

Table 2-1 CPU Register Function

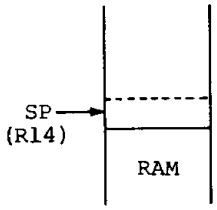
| Register | Symbol | Function |
|---|---|---|
| Program Counter | PC | 16 bit PC contains the address of instruction to be executed. Lower 14 bits of the PC effectively support 16k-byte address space, while upper 2 bits are ignored. PC is initialized to $3400 during Reset. |
| General Purpose Register | R0-R15 | General Purpose Registers can be used as both address and data registers. Depending on the instruction, these registers can be used as individual 8-bit registers (R0, R1, ----, R15) or 16-bit register pairs (R1R0, R3R2, ----, R15R14). R14 also functions as Stack Pointer(SP) during subroutine calls. Registers are not initialized by reset. |
| Condition Code Register | CCR | CCR consists of Carry (C), Zero (Z), and Negative (N), and reflects the result of logical and arithmetic instructions. Each function is described in Table 2-2. They are not initialized by reset. |

Table 2-2 Condition Code Register Description

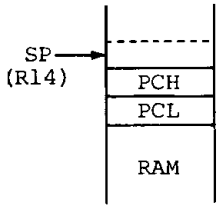
| Condition Code | Symbol | Function |
|---|---|---|
| Negative | N | Bit 2. Set if the result is negative (MSB=1); reset otherwise. |
| Zero | Z | Bit 1. Set if the result is zero; reset otherwise. |
| Carry | C | Bit 0. Set if a "carry" or "borrow" occurs from the MSB of the result; reset otherwise. A "carry" or "borrow" occurs after the following instruction executions.<br>(a) Addition<br>(b) Subtraction<br>(c) Shift and Rotation |

### 2.1.3 Subroutine calls

CALL: CALL pushes the PC onto the stack and then the CPU begins the subroutine. Then R14 functions as Stack Pointer for stacking.




Before CALL execution
(After RET execution)




After CALL execution
(Before RET execution)

Because the SP (R14) is an 8-bit register, the stack area ranges from $00 to $FF in the memory address space. The SP pointing to the next available address of the stack is decremented by two after CALL execution.

RET: RET pops the PC from the stack. Then, the SP is incremented by two.

### 2.1.4 Instruction format

| | | | |
|---|---|---|---|
| 1 FN D | IMM | | Immediate |
| 0 0 0 FN | S | D | Register-Register |
| 0 0 1 FN | S | D | Memory-Register |
| 0 0 1 0 1 1 1 0 | D | S | ST (Store) |
| 0 1 0 0 0 COND | DISP | | Jump (Relative) |
| 0 1 0 1 0 COND | R | 0 0 0 0 | Jump (Indirect) |
| 0 1 0 0 1 1 1 0 | DISP | | CALL (Relative) |
| 0 1 0 1 1 1 1 0 | R | 1 1 1 0 | CALL (Indirect) |
| 0 1 1 1 1 1 1 0 | 0 0 0 0 1 1 1 0 | | RET |

Note: FN : Function field
IMM : Immediate data
COND: Jump condition
DISP: Displacement
S : Source register field
D : Destination register field
R : Register field

## 2.2 CPU Basic Timing

### 2.2.1 Machine cycle

The external clock (CLK) is divided by four to generate an S clock. One cycle of S clock makes up one machine cycle (one memory cycle). The machine cycle is shown in **Figure 2-2.**

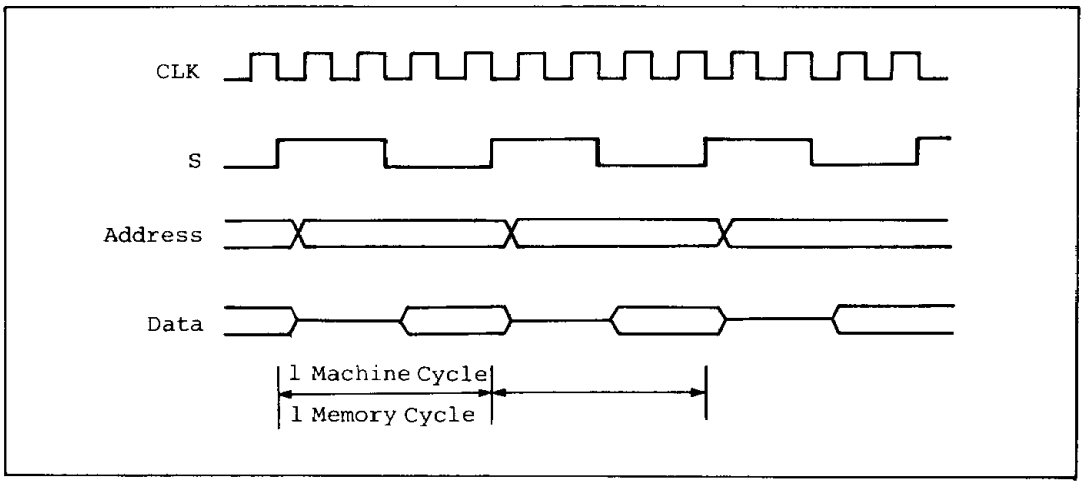



Figure 2-2 Machine Cycle

### 2.2.2 Instruction execution timing

All instructions are executed in four machine cycles except CALL and RET instructions which require five machine cycles for execution. **Figures 2-3** 2-11 illustrate basic instruction execution timings.

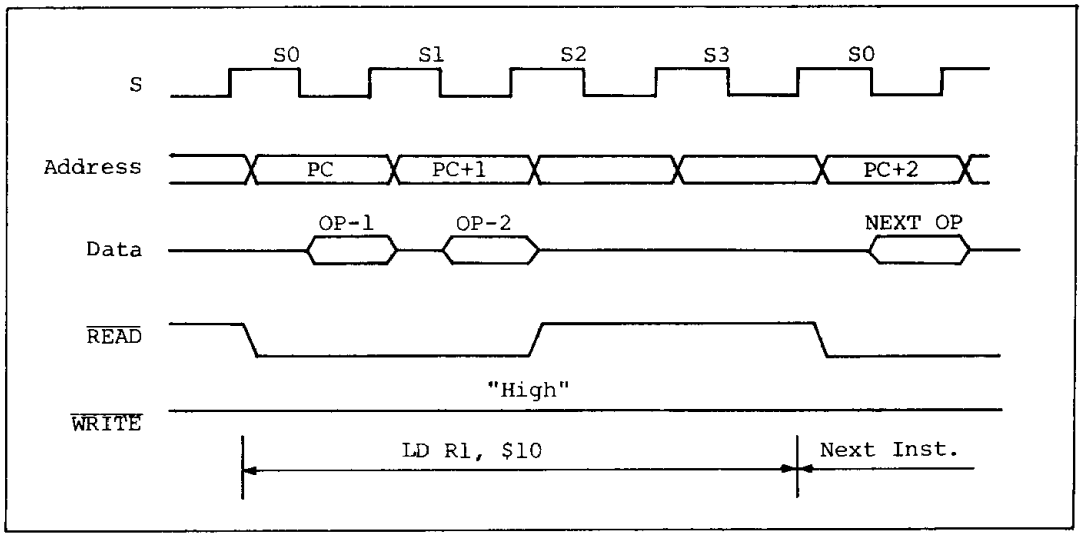



Figure 2-3 LD R1, $10 Execution Timing (Immediate)

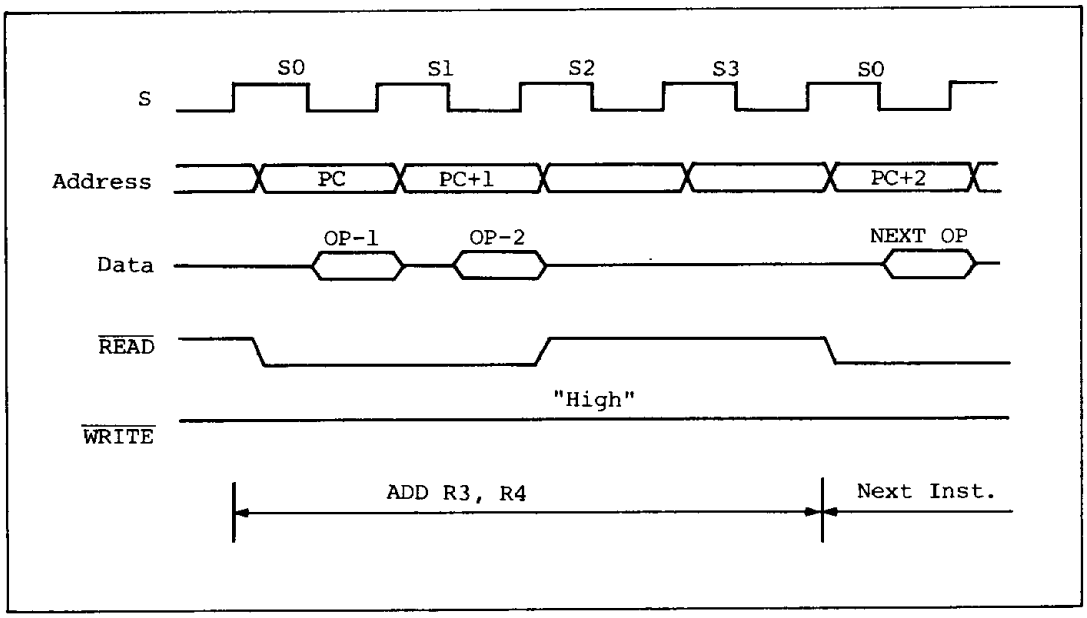



Figure 2-4 ADD R3, R4 Execution Timing (Register Direct)

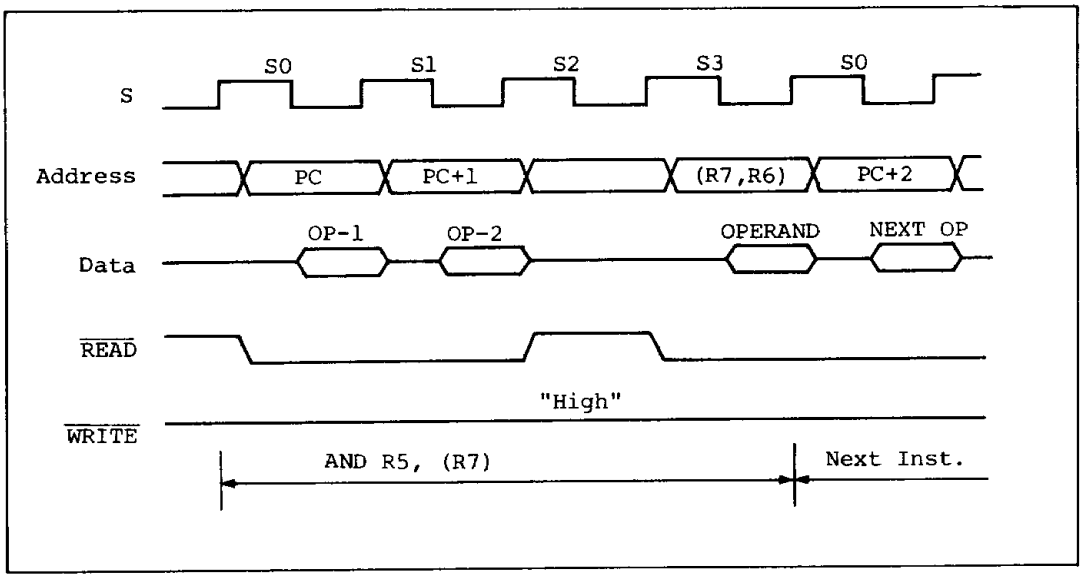



Figure 2-5 AND R5, (R7) Execution Timing (Register Indirect)

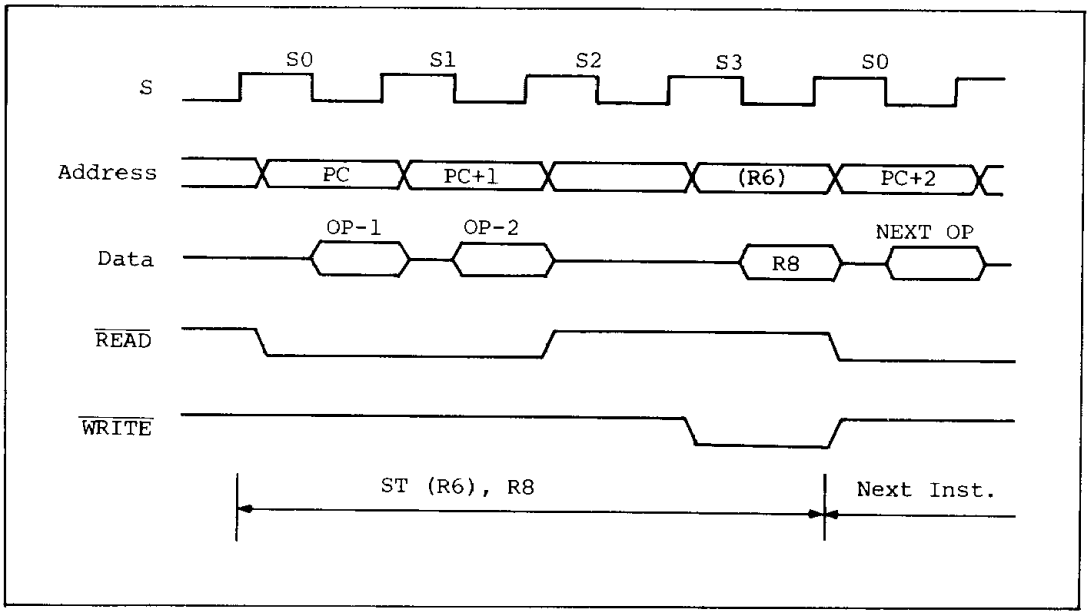



Figure 2-6 ST (R6), R8 Execution Timing (Register Indirect)

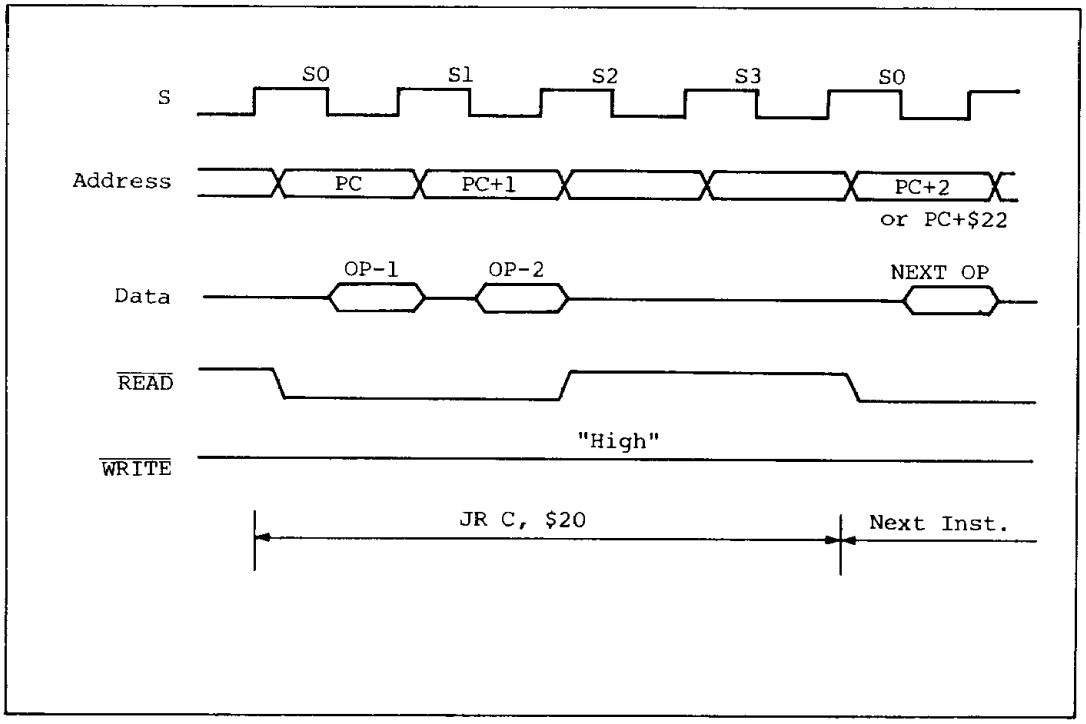



Figure 2-7 JR C, $20 Execution Timing (Relative)

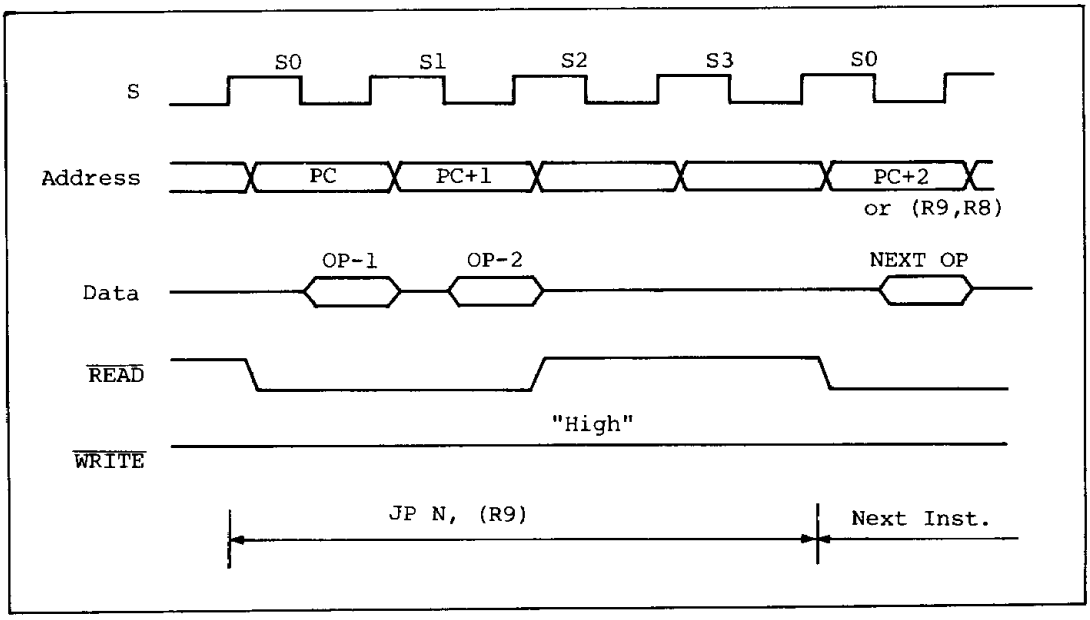



Figure 2-8 JP N, (R9) Execution Timing (Register Indirect)

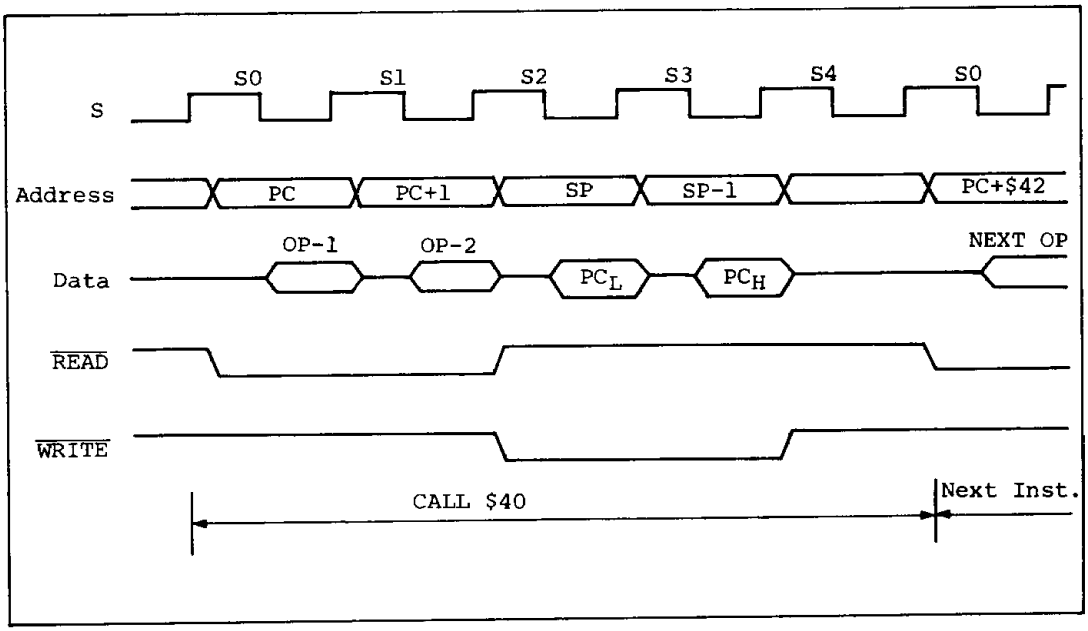



Figure 2-9 CALL $40 Execution Timing (Relative)

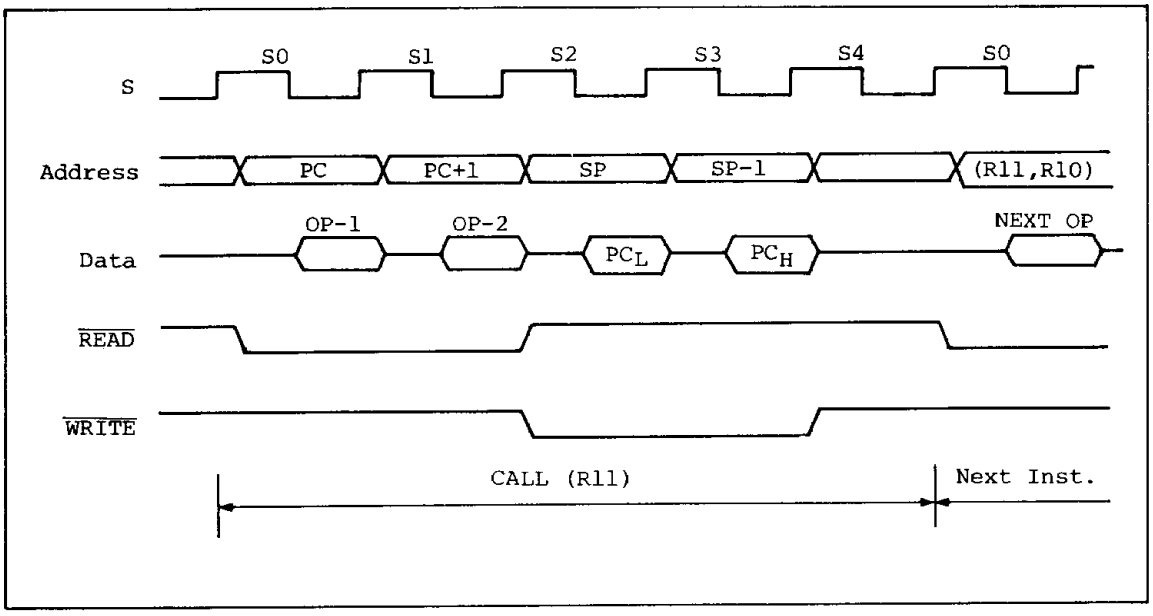



Figure 2-10 CALL (R11) Execution Timing (Register Indirect)

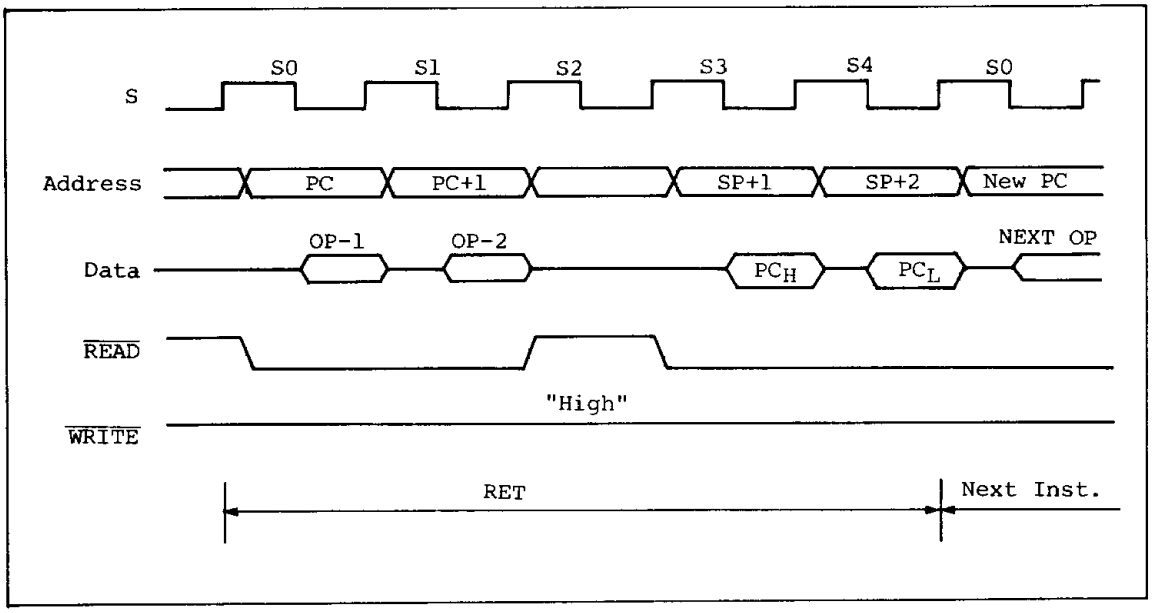



Figure 2-11 RET Execution Timing

### 2.2.3 Reset and restart timing

When $\overline{RES}$ is asserted LOW, the MCU enters the reset mode. To complete reset, $\overline{RES}$ must be asserted LOW for at least 5 machine cycles (20 external clock cycles).

When $\overline{RES}$ is brought HIGH again, the CPU restarts operations from $3400 (Starting address of ROM).

$\overline{RES}$ must also be brought LOW to facilitate power-on reset.

Reset and restart timing is shown in Figure 2-12.

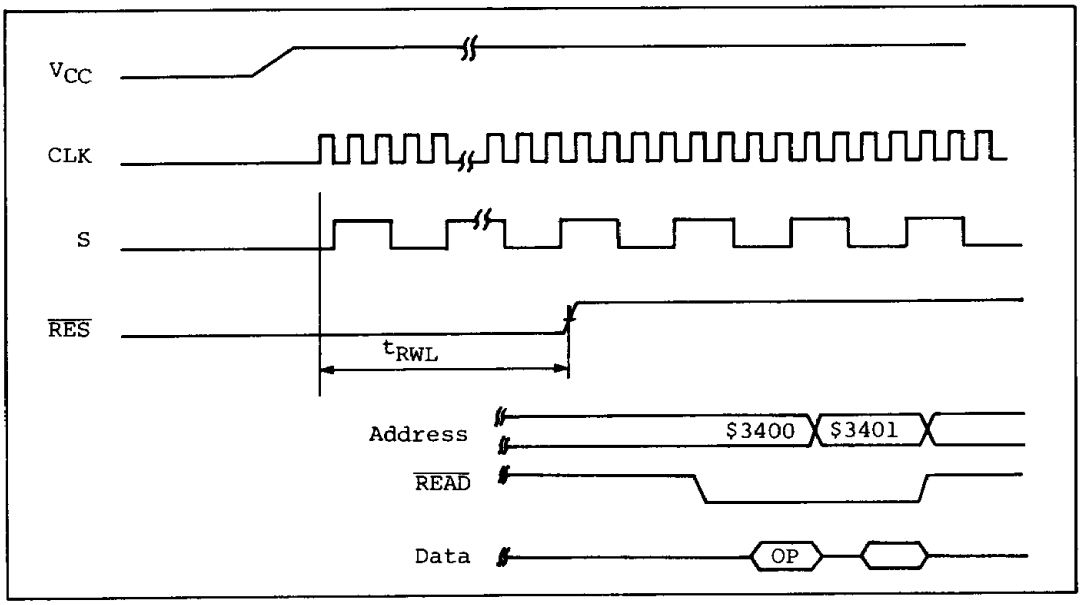
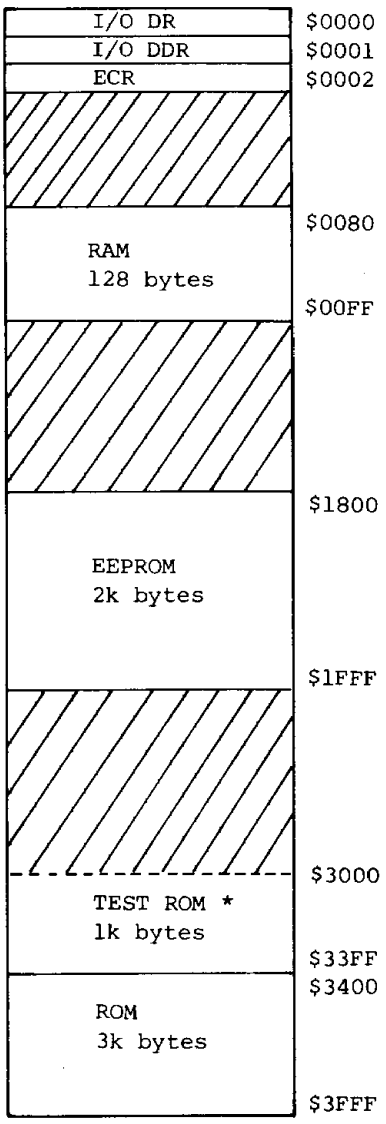



Figure 2-12 Reset Timing

## 2.3 Memory Map

Figure 2-13 shows the memory configuration.




NOTE: * TEST ROM is used only for testing and users cannot access the TEST ROM.

DR (Data Register)
DDR (Data Direction Register)
ECR (EEPROM Control Resister)

Figure 2-13 Memory Map

# Section 3. On-Chip Peripherals

## 3.1 EEPROM

### 3.1.1 On-chip EEPROM overview

The MCU incorporates EEPROM; electrically erasable and programmable ROM. The on-chip EEPROM features are as follows.

- 2k bytes
- Located in memory address space
- Byte erase/write
- Page erase/write
- Address, data, and control latches for erase/write
- On-chip high voltage generation circuit
- On-chip clock generator and timer
- Maximum $10^4$ erase/write times
- 10 year data retention

### 3.1.2 EEPROM block diagram

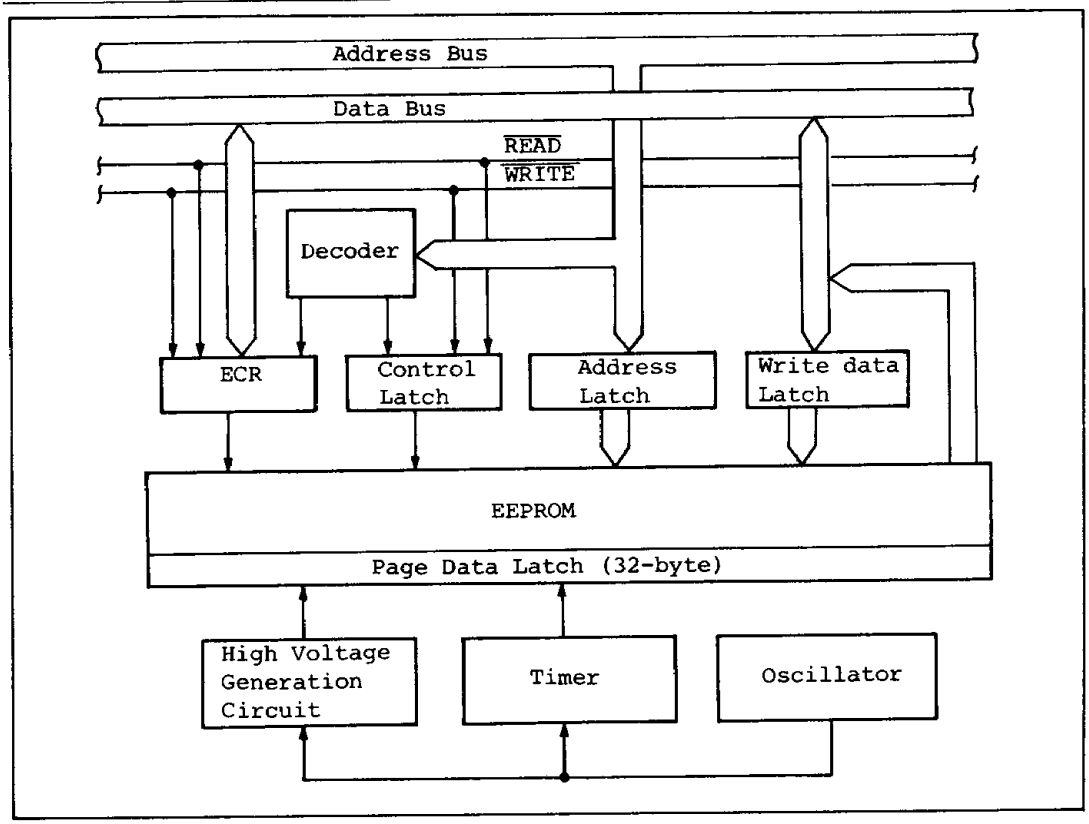



Figure 3-1 EEPROM Block Diagram

Memory mat: The EEPROM is a 2048 × 8 bits matrix with 64 pages (32 byte/page) for page erase/write. The EEPROM configuration is shown in Figure 3-2.

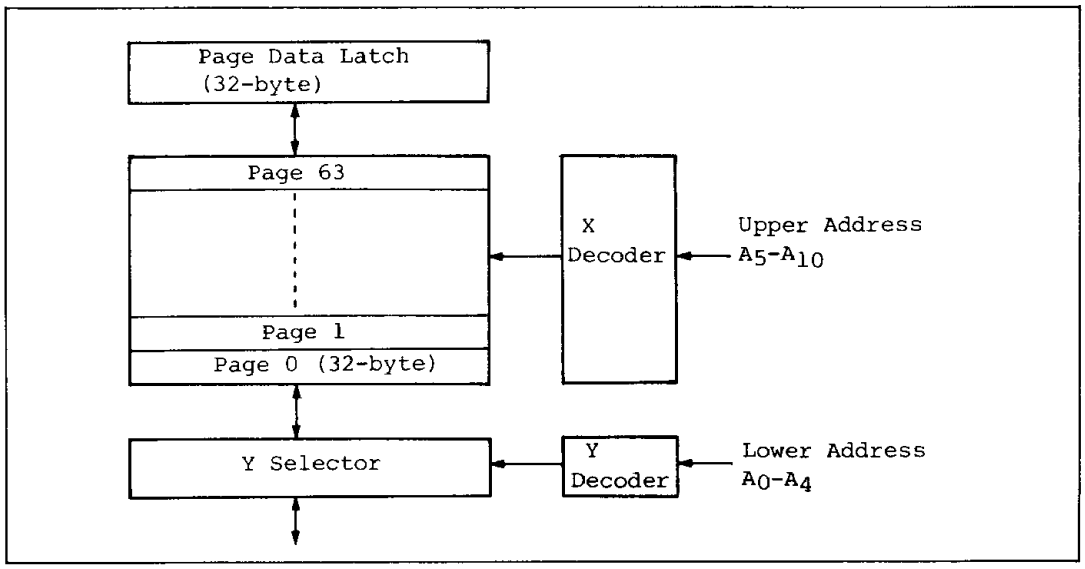



Figure 3-2 EEPROM Memory Mat

Address, data, and control latches: The EEPROM incorporates address, data and control latches. The latched data can be retained during programming. This enables EEPROM programming to be carried out in the same way as RAM programming. (CPU need not hold address and data during erase/write cycles.)

Clock generator and timer: The EEPROM combines a clock generator and timer to control EEPROM erase/write operations. CLK (the CPU system clock) is independent of EEPROM clock so that the EEPROM erase/write timing will not be affected by the CPU operating frequency.

High voltage generation circuit: A high voltage generation circuit for programming is incorporated in the EEPROM.

EEPROM Control Register (ECR): The EEPROM Control Register (ECR) controls EEPROM erasures and programmings. The ECR configuration is illustrated in Figure 3-3.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| ECR | $\overline{BSY}$ | - | - | - | - | - | WI | EI | (Address=$0002) |

Figure 3-3 EEPROM Control Register

Table 3-1 summarizes the ECR functions.

Table 3-1 ECR Function

| Bit | Bit Name | Symbol | Function | Read/Write | Initial Value |
|---|---|---|---|---|---|
| 7 | READY/$\overline{\text{BUSY}}$ | $\overline{\text{BSY}}$ | Reflects the condition of EEPROM. If $\overline{\text{BSY}}$=0, EEPROM is in the erase/write cycle. If $\overline{\text{BSY}}$=1, EEPROM erase/write cycle is completed. Note that EEPROM can be accessed only when $\overline{\text{BSY}}$=1. | Read | 1 *1 |
| 1 | Write Inhibit | WI | Inhibits the EEPROM writing. If WI=0, EEPROM writing is enabled. If WI=1, EEPROM writing is prevented. WI is utilized for page erasure. | *2 Read/Write | 0 *1 |
| 0 | Erase Inhibit | EI | Inhibits the EEPROM erasure. If EI=0, EEPROM erasure is enabled. If EI=1, EEPROM erasure is inhibited. The EEPROM data can be programmed from 1 to 0, but it cannot be reprogrammed from 0 to 1 (equivalent to PROM). | *2 Read/Write | 0 *1 |

Notes: *1. If the $\overline{\text{RES}}$ pin is asserted LOW during the erase-write cycle, EEPROM stops the erase-write operation. And then the $\overline{\text{BSY}}$, WI and EI bits are initialized by reset. It may cause a wrong erase or a wrong write. So it should not be reset during the erase/write operation.

*2. WI and EI are programmable only when $\overline{\text{BSY}}$=1.

*3. Unused bits of the ECR (bit 2-6) are always "1".

### 3.1.3 EEPROM read operation

The EEPROM can be directly read by the CPU in the same way as ROM and RAM only when $\overline{\text{BSY}}$ in the ECR is set to 1. If the EEPROM is read when $\overline{\text{BSY}}$=0, the MCU performs data polling to check whether the erase/write operation is completed or not. Refer to 3.1.4 (7) Data polling for further details. Figure 3-4 shows the EEPROM read timing.

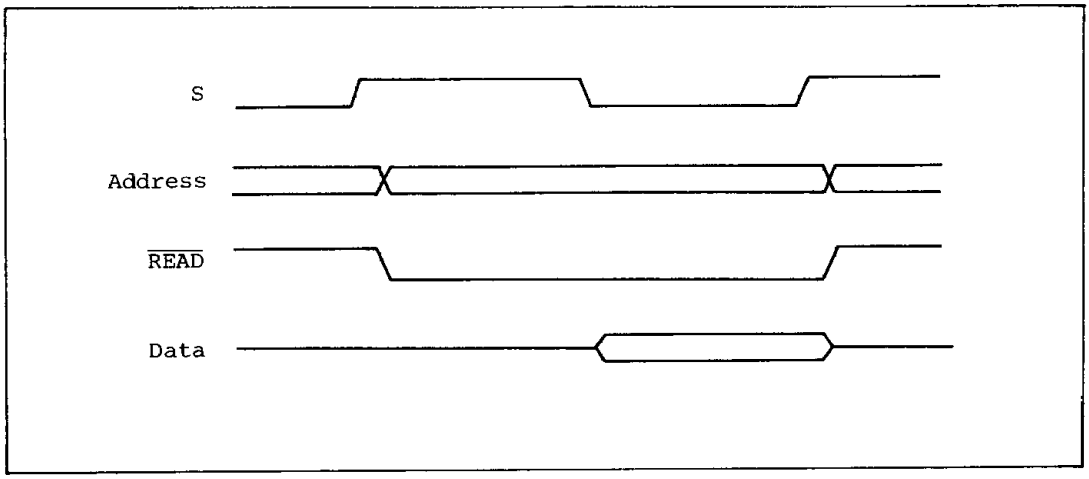



Figure 3-4 EEPROM Read Timing

### 3.1.4 EEPROM erase/write

(1) EEPROM erase/write sequence: EEPROM erase/write is done by the ST instruction like RAM. Once the ST instruction is executed, EEPROM begins the erase and write operation.
First, EEPROM latches the address and data, second, it erases the old data and finally it writes the new data which is latched in the first.
These sequence is controlled and done automatically by the internal control circuit.
Figure 3-5 summarizes EEPROM erase/write sequence.

(2) Byte erase/write (EI=0): The EEPROM can be programmed on a byte basis with ST instruction. The EEPROM erase/write time specification is 15ms (max.). $\overline{\text{BSY}}$ in the ECR can be used to check whether the erase/write cycle has been completed or not. Figure 3-6 shows byte erase/write timing.

(3) Byte write (EI=1): If EI=1, the EEPROM can be written on a byte basis but it can not be erased. Because byte locations are not erased prior to a new data write, the existing data and new data are logically ANDed to obtain new programmed data.

| | | |
|---|---|---|
| Ex. | existing data | 10010110 |
| | new data | 10001111 |
| | new programmed data | 10000110 |

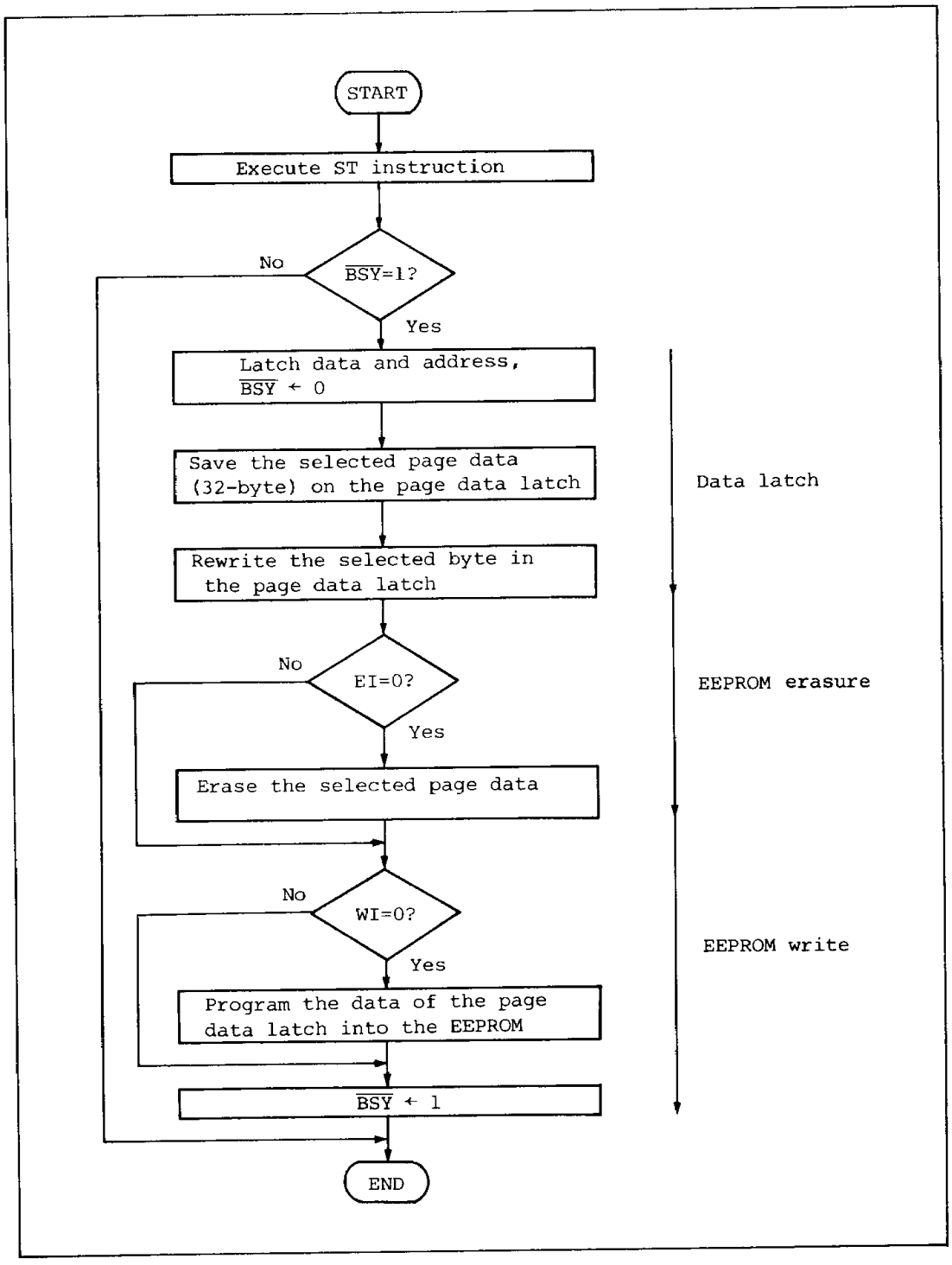



Figure 3-5 EEPROM Erase/Write Sequence

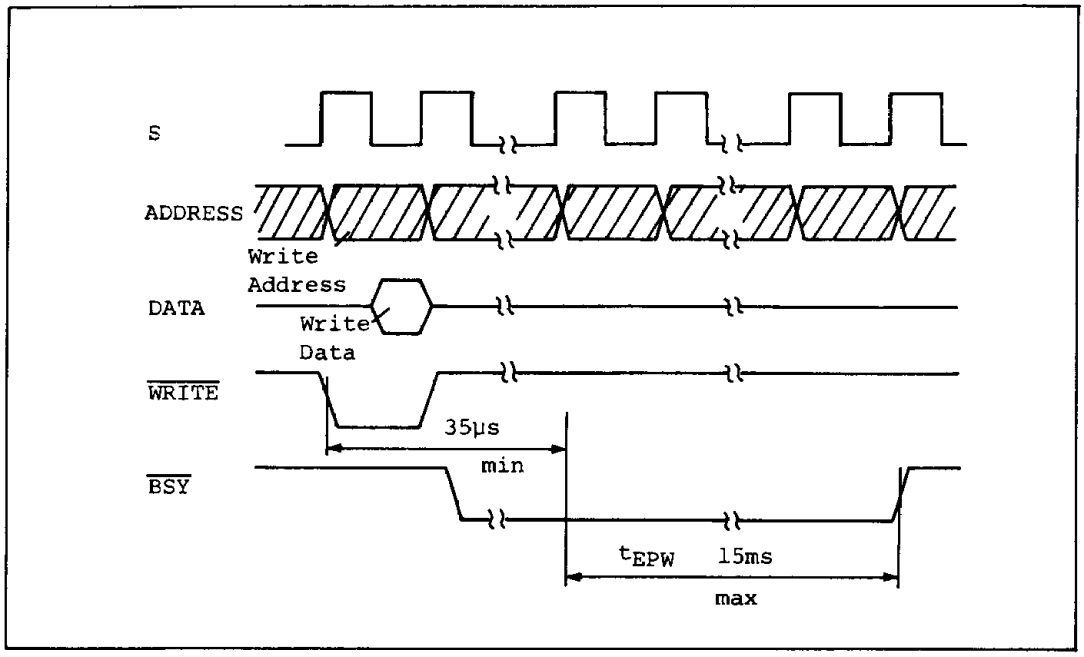



Figure 3-6 Byte Erase/Write Timing

(4) Page write (EI=0): Write operations are done on page basis as shown in Figure 3-5. Therefore more than one byte(2 to 32 bytes)can be written in one write cycle if writing data are existing in a same page. The operations are performed easily by writing the data consecutively during data latching. The intervals of each written data must be less than 35μs. Figure 3-7. shows the timing required for page write. It takes a maximum of 15ms to write one page.

Data bytes written at one time must be on the same page. If data are on the same page, the order in which the bytes are written is unrestricted.

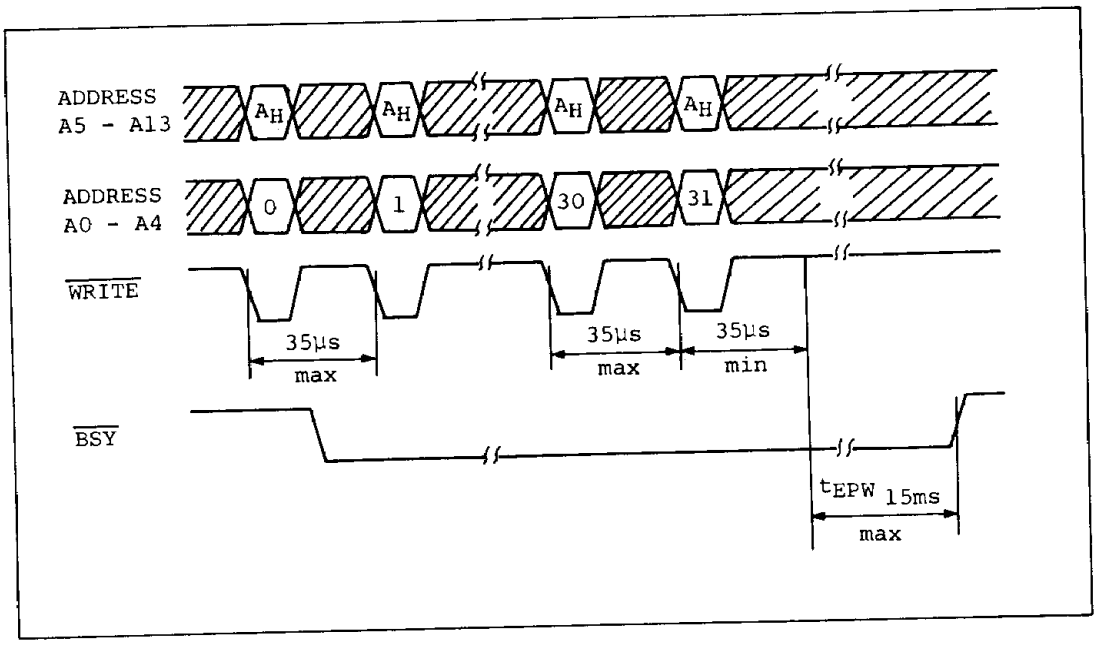



Figure 3-7 Page Write Timing

(5) Page write (EI=1): Since erase operation is not performed before the write operation, the writing data is ANDed with the last data before writing.

It takes a maximum of 15ms to writing one page.

(6) Page erase: Page erase operations are performed by writing any data of one byte with WI bit of ECR "1". The page to be erased are selected by the upper address (A5 to A13).

The maximum time required to erase one page is 15ms. Completion of erasure can be determined by monitoring the $\overline{BSY}$ flag.

(7) Data polling: Besides monitoring the $\overline{BSY}$ flag, data polling can be used to determine when erase/write operations have been completed.

When reading EEPROM during erase/write operations, an inverted data of the last written data is read out to the MSB. "0s" are read to the lower 7 bits regardless of write data. (Refer to Figure 3-8.)

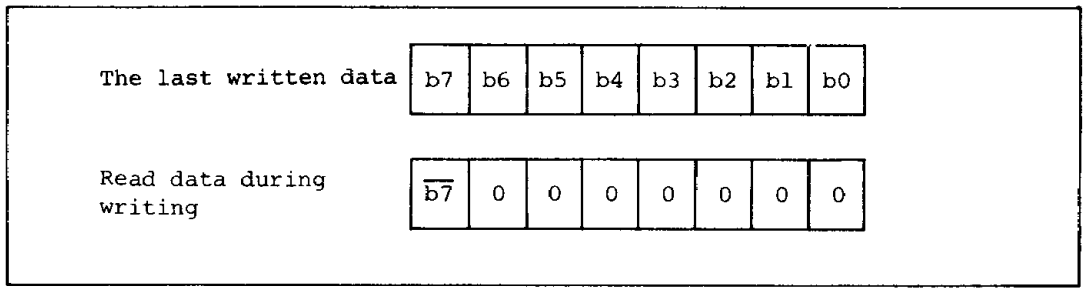



Figure 3-8 Data Polling

(8) Others: When write operations are performed with WI and EI of ECR set, the $\overline{BSY}$ is held at "0" for a maximum of 15ms. The memory contents are not changed. (Data polling also functions.)

EEPROM operation modes are summerised in Table 3-2.

Table 3-2 EEPROM Operation Mode

| | Control | | | | |
|---|---|---|---|---|---|
| | | | ECR | | |
| Mode | $\overline{READ}$ | $\overline{WRITE}$ | WI | EI | $\overline{BSY}$ |
| Read | 0 | 1 | - | - | 1 |
| Data Polling | 0 | 1 | - | - | 0 |
| Erase-Write * | 1 | 0 | 0 | 0 | 1→0 |
| Write | 1 | 0 | 0 | 1 | 1→0 |
| Page Erasure | 1 | 0 | 1 | 0 | 1→0 |
| Erase Write Inhibit | 1 | 0 | 1 | 1 | 1→0 |

Notes: * Byte write and page write operations are the same in terms of operation mode. The difference is that in number of bytes to be written at one time.

## 3.2 I/O

This pin is a 1-bit data I/O pin which is set to input or output by the software. It is TTL compatible.

### 3.2.1 I/O pin configuration

Figure 3-9 shows I/O pin configuration.

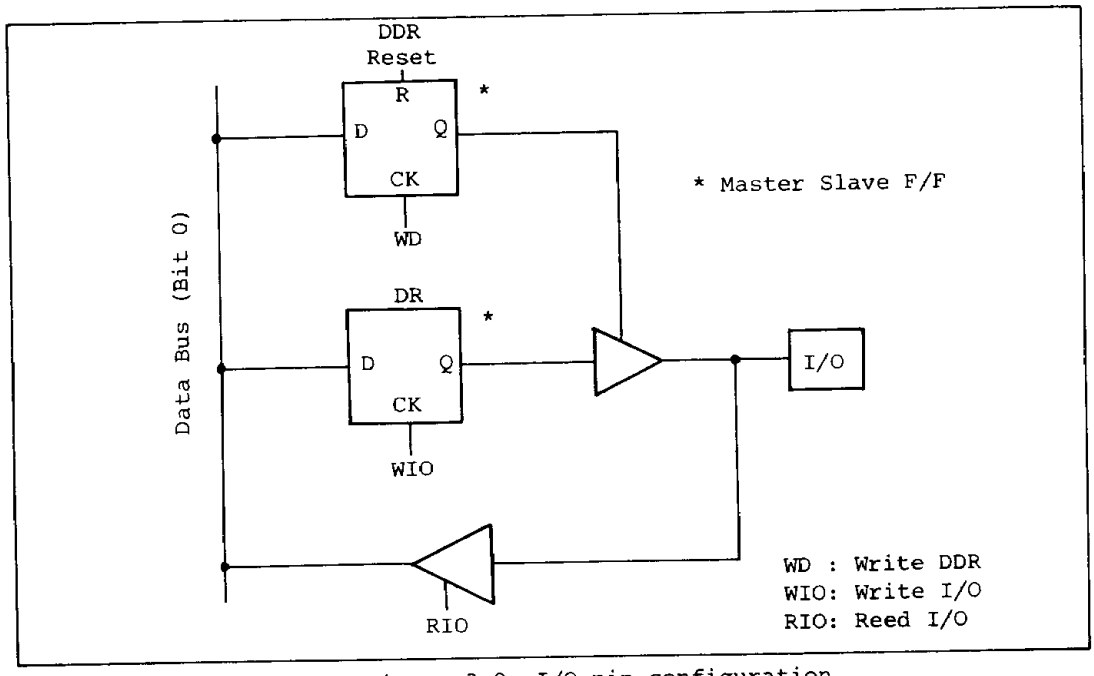



Figure 3-9 I/O pin configuration

The I/O pin includes a data register (DR) which latches output data, and a data direction register (DDR) which designates input and output direction. Their configurations are shown in Figure 3-10.

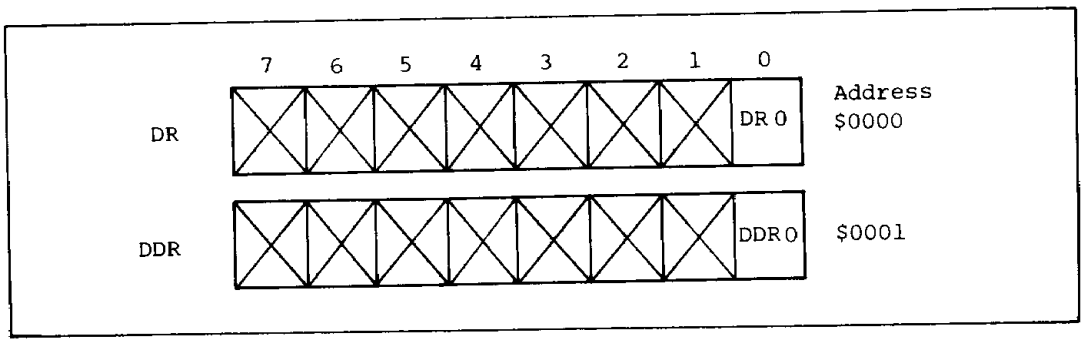



Figure 3-10 I/O Register

Data Direction Register (DDR): DDR is a 1-bit register. When writing "1" to bit 0, the I/O pin is used as output, while writing "0", I/O pin becomes input. DDR is a "write-only" register. If it is read, $FF appears.

DDR becomes "0" (input) by reset.

Data Register (DR): DR is a 1-bit "write-only" register. The content of bit 0 is output to the I/O pin. If it is read, the I/O pin status is read instead of the contents of DR ("1s" are read except for bit 0).

DR is not initialized by reset. (Undefined after power-on.)

### 3.2.2 I/O pin timing

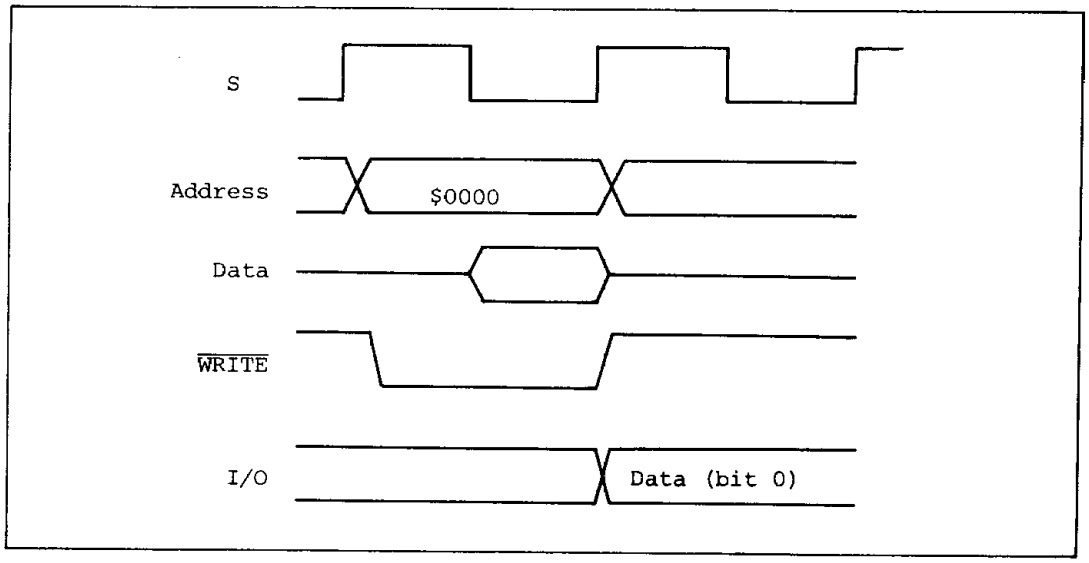



Figure 3-11 I/O pin Output Timing

# Section 4. Instruction Set Overview

## 4.1 Instruction Set Details

The symbols used in the Instruction Set Details and Instruction Set Summary are described as follows.

### 4.1.1 Specifying register

ri and rj are symbols for specifying a register. A register is specified (8-bit register or 16-bit register pair) by an instruction. The corresponding register is as follows.

8-bit

| ri | Ri |
|---|---|
| 0 | R0 |
| 1 | R1 |
| : | : |
| F | R15 |

16-bit

| ri | Ri Ri-1 |
|---|---|
| 1 | R1 R0 |
| 3 | R3 R2 |
| : | : |
| F | R15 R14 |

### 4.1.2 Specifying conditions

f is a symbol which specifies a condition when executing a jump instruction. The conditions corresponding to each f are as follows.

| f | Condition |
|---|---|
| NZ | non zero |
| Z | zero |
| NC | non carry |
| C | carry |
| P | positive |
| N | negative |

### 4.1.3 Addressing mode

Refer to "2.1.2 Addressing mode" for details.

REG : Register Direct

REGI: Register Indirect

IMM : Immediate

REL : Relative

IMP : Implied Register

### 4.1.4 Condition code

The following symbols are used with respect to condition code changes.

• : Not changed.

↕ : Changed according to operation results.

### 4.1.5 Others

( R )$_M$ : content in the memory addressed by R

m : 8-bit immediate data

g : 8-bit displacement

H or L : "H", attached to m or g, shows upper 4 bits and "L" shows lower 4 bits.

S : Source addressing mode

D : Destination addressing mode

### 4.1.6 Precautions

For (Rj)$_M$, the contents of the register pair Rj Rj-1 are used as address if "j" is an odd number, while the content of the Register Rj is used as lower 8 bit address and upper 8 bit address is "0" if "j" is an even number.

When using the instruction ADDD, SUBD, JP, or CALL (Indirect), designate "i" of Ri as an odd number because the instructions specify a register pair (16 bits).

ARITHMETIC OPERATION

ADC

## ADC (Add with Carry)

### FORMAT

ADC Ri, m

### CONDITION CODES

C: Set if a carry is generated from the most significant bit of the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

### OPERATION

Ri + m + C → Ri

### DESCRIPTION

Adds the contents of the carry bit C to the sum of the contents of the Ri and an 8-bit data m, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/IMM | REG | ADC | Ri, m | 9ri | mHmL | 4 |

ARITHMETIC OPERATION

ADC

## ADC (Add with Carry)

**FORMAT**

ADC Ri, Rj

**CONDITION CODES**

C: Set if a carry is generated from the most significant bit of the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

**OPERATION**

Ri + Rj + C → Ri

**DESCRIPTION**

Adds the contents of the carry bit C to the sum of the contents of register Ri and Rj, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | ADC | Ri, Rj | 01 | rjri | 4 |

ARITHMETIC OPERATION

ADC

## ADC (Add with Carry)

### FORMAT

ADC Ri, (Rj)

### CONDITION CODES

C: Set if a carry is generated from the most significant bit of the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

### OPERATION

$Ri + (Rj)_M + C \rightarrow Ri$

### DESCRIPTION

Adds the contents of the carry bit C to the sum of the contents of register Ri, and the data addressed by the contents of the Rj.
The result is placed in Ri.
When the Rj is an odd-numbered register (j=1,3,5,....,15), the memory address is contained in the register pair RjRj-1.
While, when the Rj is an even-numbered register, the lower byte of memory address is contained in the Rj and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | ADC | Ri, (Rj) | 21 | rjri | 4 |

ARITHMETIC OPERATION

ADD

ADD (Add without Carry)

FORMAT

ADD Ri, m

OPERATION

Ri + m → Ri

CONDITION CODES

C: Set if a carry is generated from the most significant bit of the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

DESCRIPTION

Adds an 8-bit data m, to the contents of register Ri (i=0,1,2,3, ....., 15) and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/IMM | REG | ADD | Ri, m | 8ri | mHmL | 4 |

ARITHMETIC OPERATION

ADD

## ADD (Add without Carry)

### FORMAT

ADD Ri, Rj

### OPERATION

Ri + Rj → Ri

### CONDITION CODES

C: Set if a carry is generated from the most significant bit of the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

### DESCRIPTION

Adds the contents of register Rj to the contents of register Ri, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | ADD | Ri, Rj | 00 | rjri | 4 |

ARITHMETIC OPERATION

ADD

ADD (Add without Carry)

FORMAT

ADD Ri, (Rj)

OPERATION

$Ri + (Rj)_M \rightarrow Ri$

CONDITION CODES

C: Set if a carry is generated from the most significant bit of the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

DESCRIPTION

Adds the data in the memory addressed by the contents of register Rj, to the contents of register Ri, and place the result in the Ri.
When the Rj is an odd-numbered register (j=1,3,5,....,15), the memory address is contained in the register pair RjRj-1.
While, when the Rj is an even-numbered register, the lower byte of memory address is contained in the Rj(j=0,2,.....,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | ADD | Ri, (Rj) | 20 | rjri | 4 |

ARITHMETIC OPERATION

ADDD

ADDD (Add Double)

FORMAT

ADDD Ri, Rj (i=1,3,5,....,15)

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared other wise

N: Not affected

OPERATION

Ri-1 + Rj → Ri-1

Ri + Carry* → Ri

*: Carry from the result of lower byte addition.
C flag of CCR will not change.

DESCRIPTION

Adds the contents of register Rj to the contents of register pair RiRi-1 (i=1,3,5,....., 15), and places the result in the register pair RiRi-1.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | ADDD | Ri, Rj | 10 | rjri | 4 |

ARITHMETIC OPERATION

ADDD

ADDD (Add Double)

FORMAT

ADDD Ri, (Rj) (i=1,3,5,...., 15)

CONDITION CODES

C: Not affected

Z: Set if the results 0;
cleared otherwise

N: Not affected

OPERATION

$Ri-1 + (Rj)_M \rightarrow Ri-1$

$Ri + Carry^* \rightarrow Ri$

*:Carry from the result of lower byte addition.
C flag of CCR will not change.

DESCRIPTION

Adds the data in the memory addressed by the contents of register Rj, to the contents of register pair RiRi-1(i=1,3,5, ...., 15), and places the result in the register pair RiRi-1(i=1,3,5, ...., 15).
When the Rj is an odd-numbered register (j=1,3,5,....,15), the memory address is specified by contents of register pair $RjR_{j-1}$.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,.....,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | ADDD | Ri, (Rj) | 30 | rjri | 4 |

LOGICAL OPERATION

AND

## AND (Logical AND)

### FORMAT

AND Ri, m

### CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

### OPERATION

Ri ∧ m → Ri

### DESCRIPTION

Performs logical AND operation between the contents of register Ri and an 8-bit data m, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/IMM | REG | AND | Ri, m | Eri | mHmL | 4 |

| LOGICAL OPERATION |
|---|
| AND |

AND (Logical AND)

**FORMAT**

AND Ri, Rj

**CONDITION CODES**

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

**OPERATION**

Ri ∧ Rj → Ri

**DESCRIPTION**

Performs logical AND operation between the contents of registers Ri and Rj, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | AND | Ri,Rj | 06 | rjri | 4 |

LOGICAL OPERATION

AND

AND (Logical AND)

FORMAT

AND Ri, (Rj)

CONDITION CODES

C: Not affected.

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

$Ri \wedge (Rj)_M \rightarrow Ri$

DESCRIPTION

Performs logical AND operation between the contents of register Ri and the data in the memory addressed by the contents of register Rj.
When the Rj is an odd-numbered register (j=1,3,5,....,15), the memory address is specified by contents of register pair $RjR_{j-1}$.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,.....,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | AND | Ri, (Rj) | 26 | rjri | 4 |

PROGRAM CONTROL

CALL

## CALL (Call Subroutine)

### FORMAT

CALL g

### CONDITION CODES

C: Not affected

Z: Not affected

N: Not affected

### OPERATION

PCL → $(SP)_M$

PCH → $(SP - 1)_M$

SP - 2 → SP

PC + g → PC

### DESCRIPTION

The contents of the PC is pushed onto the stack in order of lower 8-bit and upper 8-bit.

The program counter PC that shows the address of instruction followed by CALL is added with displacement g.

Then a branch occurs to the address specified by the PC.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REL | | CALL | g | 4E | gHgL | 5 |

PROGRAM CONTROL

CALL

## CALL (Call Subroutine)

### FORMAT

CALL (Ri) (i=1,3,5,...., 15)

### CONDITION CODES

C: Not affected

Z: Not affected

N: Not affected

### OPERATION

PCL → $(SP)_M$

PCH → $(SP - 1)_M$

SP - 2 → SP

$RiRi-_1$ → PC

### DESCRIPTION

The contents of program counter PC, which shows the address of instruction followed by CALL, is pushed onto the stack in order of lower 8-bit and upper 8-bit.
A branch occurs to the address specified by the contents of register pair $RiRi-_1$ (i=1,3,5,...., 15).

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REGI | | CALL | (Ri) | 5E | riE | 5 |

ARITHMETIC OPERATION

CMP

CMP (Compare)

FORMAT

CMP Ri, m

OPERATION

Ri - m

CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

DESCRIPTION

Compares the contents of register Ri and an 8-bit data m.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/IMM | - | CMP | Ri, m | Ari | mHmL | 4 |

ARITHMETIC OPERATION

CMP

CMP (Compare)

FORMAT

CMP Ri, Rj

OPERATION

Ri - Rj

CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

DESCRIPTION

Compares the contents of registers Ri and Rj.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | - | CMP | Ri, Rj | 13 | rjri | 4 |

ARITHMETIC OPERATION

CMP

CMP (Compare)

FORMAT

CMP Ri, (Rj)

OPERATION

$Ri - (Rj)_M$

CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

DESCRIPTION

Compares the contents of register Ri and the data in the memory addressed by the contents of register Rj.
When the Rj is an odd-numbered register (j=1,3,5,....,15), the memory address is specified by contents of register pair $RjR_{j-1}$.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,.....,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | - | CMP | Ri, (Rj) | 33 | rjri | 4 |

TRANSFER

CTR

CTR (Condition Code to Register)

FORMAT

CTR Ri

CONDITION CODES

C: Not affected

Z: Not affected

N: Not affected

OPERATION

CCR → Ri

DESCRIPTION

Transfer the contents of condition code register to the register Ri.

The upper 5 bits (bit 3 to bit 7) are cleared.

C → Ri bit 0

Z → Ri bit 1

N → Ri bit 2

"0" → Ri bit 3 to 7

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| IMP | REG | CTR | Ri | 1A | 0ri | 4 |

ARITHMETIC OPERATION

DEC

DEC (Decrement)

FORMAT

DEC Ri

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

Ri - 1 → Ri

DESCRIPTION

Subtracts one from the contents of register Ri (i=0,1,2,..., 15), and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | DEC | Ri | 09 | 0ri | 4 |

LOGICAL OPERATION

EOR

EOR (Exclusive OR)

FORMAT

EOR Ri, m

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

Ri ⊕ m → Ri

DESCRIPTION

Performs the logical EXCLUSIVE OR operation between the contents of register Ri and an 8-bit data m, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/IMM | REG | EOR | Ri,m | Dri | mHmL | 4 |

| LOGICAL OPERATION |
|---|
| EOR |

EOR (Exclusive OR)

**FORMAT**

EOR Ri, Rj

**CONDITION CODES**

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

**OPERATION**

Ri ⊕ Rj → Ri

**DESCRIPTION**

Performs the logical EXCLUSIVE OR operation between the contents of registers Ri, and the Rj, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | EOR | Ri, Rj | 05 | rjri | 4 |

LOGICAL OPERATION

EOR

EOR (Exclusive OR)

FORMAT

EOR Ri, (Rj)

CONDITION CODES

C: Not affected.

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

$Ri \oplus (Rj)_M \rightarrow Ri$

DESCRIPTION

Performs the logical EXCLUSIVE OR operation between the contents of register Ri and the data in the memory addressed by the contents of register Rj. The result is placed in the Ri.
When the Rj is an odd-numbered register (j=1,3,5,....,15), the memory address is specified by the contents of register pair $RjR_{j-1}$.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,.....,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | EOR | Ri, (Rj) | 25 | rjri | 4 |

ARITHMETIC OPERATION

INC

INC (Increment)

FORMAT

INC Ri

CONDITION CODES

C: Not affected.

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

Ri + 1 → Ri

DESCRIPTION

Adds one to the contents of register Ri, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | INC | Ri | 0B | 0ri | 4 |

PROGRAM CONTROL

JP

JP (Jump)

FORMAT

JP (Ri)

(i=1,3,5,····, 15)

CONDITION CODES

C: Not affected

Z: Not affected

N: Not affected

OPERATION

RiRi-1 → PC

DESCRIPTION

Stores the contents of register pair RiRi-1 (i=1,3,5,····, 15) to the program counter PC.

A branch occures to the address specified by the contents of register pair RiRi-1.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REGI | | JP | (Ri) | 50 | ri0 | 4 |

PROGRAM CONTROL

JP

JP (Jump on Condition)

FORMAT

JP f, (Ri)

(i=1,3,5,····, 15)

CONDITION CODES

C: Not affected

Z: Not affected

N: Not affected

OPERATION

f is true : RiRi-1 → PC

f is false : Continue

DESCRIPTION

Tests the status of condition codes (C,NC,N,P,Z,NZ).
When the condition is true, a branch occurs to the address specified by the contents of register pair RiRi-1 (i=1,3,5,····, 15), otherwise the next sequential instruction is executed.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REGI | | JP | P, (Ri) | 52 | ri0 | 4 |
| | | JP | N, (Ri) | 53 | ri0 | 4 |
| | | JP | NZ, (Ri) | 54 | ri0 | 4 |
| | | JP | Z, (Ri) | 55 | ri0 | 4 |
| | | JP | NC, (Ri) | 56 | ri0 | 4 |
| | | JP | C, (Ri) | 57 | ri0 | 4 |

PROGRAM CONTROL

JR

## JR (Jump Relative)

### FORMAT

JR g

### CONDITION CODES

C: Not affected

Z: Not affected

N: Not affected

### OPERATION

PC + g → PC

### DESCRIPTION

The program counter PC, which shows the address of instruction followed by JR, is added with displacement g.
Then, a branch occurs to the address specified by the PC.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REL | | JR | g | 40 | gHgL | 4 |

PROGRAM CONTROL

JR

## JR (Jump Relative on Condition)

### FORMAT

JR f, g

### CONDITION CODES

C: Not affected

Z: Not affected

N: Not affected

### OPERATION

f is true : PC + g → PC

f is false : Continue

### DESCRIPTION

Tests the status of condition codes (C, NC, N, P, Z, NZ).
If the condition is true, the program counter, which shows the address of instruction followed by JR, is added with displacement g, and then branches to the address specified by the PC.
If the condition is false, the next sequential instruction is executed.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REL | | JR | P, g | 42 | gHgL | 4 |
| | | JR | N, g | 43 | gHgL | 4 |
| | | JR | NZ, g | 44 | gHgL | 4 |
| | | JR | Z, g | 45 | gHgL | 4 |
| | | JR | NC, g | 46 | gHgL | 4 |
| | | JR | C, g | 47 | gHgL | 4 |

TRANSFER

LD

## LD (Load)

### FORMAT

LD Ri, m

### CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

### OPERATION

m → Ri

### DESCRIPTION

Loads an 8-bit data m into the register Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| IMM | REG | LD | Ri, m | Fri | mHmL | 4 |

TRANSFER

LD

LD (Load)

FORMAT

LD Ri, (Rj)

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

$(Rj)_M \rightarrow Ri$

DESCRIPTION

Loads the data in the memory addressed by the contents of register Rj into register Ri.
When the Rj is an odd-numbered register (j=1,3,5,····,15), the memory address is specified by contents of register pair RjRj-1.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,·····,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REGI | REG | LD | Ri, (Rj) | 28 | rjri | 4 |

TRANSFER

MV

MV (Move)

FORMAT

MV Ri, Rj

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

Rj → Ri

DESCRIPTION

Moves the contents of register Rj into the register Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | MV | Ri, Rj | 08 | rjri | 4 |

LOGICAL OPERATION

OR

OR (Inclusive OR)

FORMAT

OR Ri, m

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

Ri V m → Ri

DESCRIPTION

Performs logical OR operation between the contents of register Ri and an 8-bit data m, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/IMM | REG | OR | Ri, m | Cri | mHmL | 4 |

| LOGICAL OPERATION |
| --- |
| OR |

OR (Inclusive OR)

**FORMAT**

OR Ri, Rj

**CONDITION CODES**

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

**OPERATION**

Ri ∨ Rj → Ri

**DESCRIPTION**

Performs logical OR operation between the contents of registers Ri and Rj, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
| --- | --- | --- | --- | --- | --- | --- |
| S | D | | | Byte 1 | Byte 2 | |
| REG | | OR | Ri, Rj | 04 | rjri | 4 |

| LOGICAL OPERATION |
|---|
| OR |

OR (Inclusive OR)

**FORMAT**

OR Ri, (Rj)

**CONDITION CODES**

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

**OPERATION**

$Ri \vee (Rj)_M \rightarrow Ri$

**DESCRIPTION**

Performs logical OR operation between the contents of register Ri and the data in memory addressed by contents of register Rj. The result is placed in the Ri.
When the Rj is an odd-numbered register (j=1,3,5,····, 15), the memory address is specified by contents of register pair $RjRj-1$.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,·····,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | OR | Ri, (Rj) | 24 | rjri | 4 |

PROGRAM CONTROL

RET

RET (Return)

**FORMAT**

RET

**CONDITION CODES**

C: Not affected

Z: Not affected

N: Not affected

**OPERATION**

$(SP + 1)_M \rightarrow PCH$

$(SP + 2)_M \rightarrow PCL$

$SP + 2 \rightarrow SP$

**DESCRIPTION**

Returns the PC from stack and causes a branch to the address specified by the PC.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| IMP | | RET | | 7E | 0E | 5 |

SHIFT OPERATION

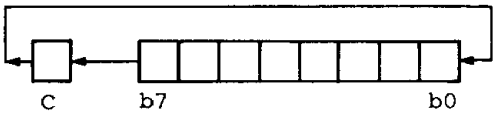
ROL

ROL (Rotate Left)

FORMAT

ROL Ri

OPERATION




CONDITION CODES

C: Set if the most significant bit of the register Ri is 1 before executing the instruction; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

DESCRIPTION

Rotates left the contents of register Ri by one bit.
Bit 7 is transferred to the carry bit C, and C is transferred to bit 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | ROL | Ri | 1F | 0ri | 4 |

SHIFT OPERATION

ROR

ROR (Rotate Right)

FORMAT

ROR Ri

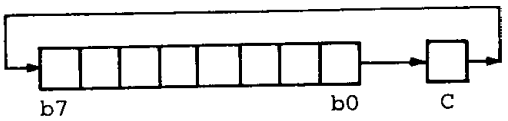
CONDITION CODES

C: Set if the bit 0 of register Ri is 1 before executing the instruction;
cleared otherwise

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant bit of the result is 1;
cleared otherwise

OPERATION




DESCRIPTION

Rotates right the contents of register Ri by one bit.
Bit 0 is transferred to the carry bit C, and C is transferred to bit 7.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | ROR | Ri | 1C | 0ri | 4 |

TRANSFER

RTC

RTC (Register to Condition Code)

FORMAT

RTC Ri

CONDITION CODES

C: Bit 0 of register Ri

Z: Bit 1 of register Ri

N: Bit 2 of register Ri

OPERATION

Ri → CCR

DESCRIPTION

Transfers the contents of register Ri to the condition code register.

Ri bit 0 → C
bit 1 → Z
bit 2 → N

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | IMP | RTC | Ri | 1B | 0ri | 4 |

ARITHMETIC OPERATION

SBC

SBC (Subtract with Carry)

FORMAT

SBC Ri, m

CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

OPERATION

Ri - m - C → Ri

DESCRIPTION

Subtracts an 8-bit data m and carry bit C from the contents of register Ri, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/IMM | REG | SBC | Ri, m | Bri | mHmL | 4 |

ARITHMETIC OPERATION

SBC

SBC (Subtract with Carry)

FORMAT

SBC Ri, Rj

CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

OPERATION

Ri - Rj - C → Ri

DESCRIPTION

Subtracts the contents of register Rj and carry bit C from the contents of register Ri, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | SBC | Ri, Rj | 03 | rjri | 4 |

ARITHMETIC OPERATION

SBC

## SBC (Subtract with Carry)

### FORMAT

SBC Ri, (Rj)

### CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

### OPERATION

$Ri - (Rj)_M - C \rightarrow Ri$

### DESCRIPTION

Subtracts the carry bit C and the data in the memory addressed by the contents of register Rj from register Ri. The result is placed in the Ri.
When the Rj is an odd-numbered register (j=1,3,5,····,15), the memory address is specified by contents of register pair RjRj-1.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,·····,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | SBC | Ri, (Rj) | 23 | rjri | 4 |

SHIFT OPERATION

SL

SL (Shift Left)

FORMAT

SL Ri

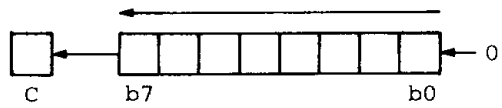
CONDITION CODES

C: Set if the most significant bit of register Ri is 1 before executing the instruction;
cleared otherwise

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant bit of the result is 1;
cleared otherwise

OPERATION




DESCRIPTION

Shifts the contents of register Ri by one bit to the left.
Bit 0 is cleared, and bit 7 is transferred to the carry bit C.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | SL | Ri | 0F | 0ri | 4 |

SHIFT OPERATION

SRA

SRA (Shift Right Arithmetic)

FORMAT

SRA Ri

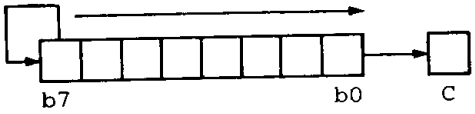
CONDITION CODES

C: Set if bit 0 of register Ri is 1 before executing the instruction; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

OPERATION




DESCRIPTION

Shifts the contents of register Ri by one bit to the right.
Bit 0 is transferred to the carry bit C, while bit 7 is not affected.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | SRA | Ri | 0D | 0ri | 4 |

SHIFT OPERATION

SRL

SRL (Shift Right Logical)

FORMAT

SRL Ri

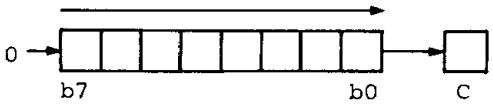
CONDITION CODES

C: Set if bit 0 of register Ri is 1 before executing the instruction; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Cleared

OPERATION




DESCRIPTION

Shift the contents of register Ri by one bit to the right.
Bit 0 is transferred to the carry bit C, while bit 7 is cleared.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | SRL | Ri | 0C | 0ri | 4 |

TRANSFER

ST

ST (Store)

FORMAT

ST (Ri), Rj

CONDITION CODES

C: Not affected

Z: Set if the data is 0;
cleared otherwise

N: Set if the most significant
bit of the data is 1;
cleared otherwise

OPERATION

$Rj \rightarrow (Ri)_M$

DESCRIPTION

Stores the contents of register Rj into the memory addressed by the contents of register Ri.
When the Ri is an odd-numbered register (i=1,3,5,····,15), the memory address is specified by the contents of register pair RiRi-1.
While, when the Ri is an even-numbered register, the lower byte of memory address is specified by the contents of Ri (i=0,2,·····,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | REGI | ST | (Ri),Rj | 2E | rirj | 4 |

ARITHMETIC OPERATION

SUB

SUB (Subtract without Carry)

FORMAT

SUB Ri, Rj

CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

OPERATION

Ri - Rj → Ri

DESCRIPTION

Subtracts the contents of register Rj from the contents of register Ri, and places the result in the Ri.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | SUB | Ri,Rj | 02 | rjri | 4 |

ARITHMETIC OPERATION

SUB

SUB (Subtract without Carry)

FORMAT

SUB Ri, (Rj)

OPERATION

$Ri - (Rj)_M \rightarrow Ri$

CONDITION CODES

C: Set if a borrow is generated from the result; cleared otherwise

Z: Set if the result is 0; cleared otherwise

N: Set if the most significant bit of the result is 1; cleared otherwise

DESCRIPTION

Subtracts the data in the memory addressed by the contents of register Rj from the contents of register Ri. The result is placed in Ri.
When the Rj is an odd-numbered register (j=1,3,5,····,15), the memory address is specified by contents of register pair RjRj-1.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,·····,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | SUB | Ri, (Rj) | 22 | rjri | 4 |

ARITHMETIC OPERATION

SUBD

SUBD (Subtract Double)

FORMAT

SUBD Ri, Rj

(i=1,3,5,····,15)

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Not affected

OPERATION

Ri-1 - Rj → Ri-1

Ri - Borrow* → Ri

*:Borrow from the result of lower byte subtraction.
C flag of CCR will not change.

DESCRIPTION

Subtracts the contents of register Rj from the contents of register pair RiRi-1 (i=1,3,5,...., 15),and places the result in RiRi-1.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | | SUBD | Ri, Rj | 12 | rjri | 4 |

ARITHMETIC OPERATION

SUBD

SUBD (Subtract Double)

FORMAT

SUBD Ri, (Rj)

(i=1,3,5,····, 15)

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Not affected

OPERATION

$Ri-1 - (Rj)_M \rightarrow Ri-1$

$Ri - Borrow^* \rightarrow Ri$

*:Borrow from the result of lower byte subtraction.
C flag of CCR will not change.

DESCRIPTION

Subtracts the data in the memory addressed by the contents of register Rj, from the contents of register pair RiRi-1 (i=1,3,5,···,15). The result is placed in RiRi-1. When the Rj is an odd-numbered register (j=1,3,5,····, 15), the memory address is specified by contents of register pair RjRj-1.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,·····,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | REG | SUBD | Ri, (Rj) | 32 | rjri | 4 |

LOGICAL OPERATION

TST

TST (Test)

FORMAT

TST Ri, Rj

CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

OPERATION

Ri ∧ Rj

DESCRIPTION

Performs logical AND operation between the contents of register Ri and register Rj.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG | - | TST | Ri, Rj | 16 | rjri | 4 |

LOGICAL OPERATION

TST

## TST (Test)

### FORMAT

TST Ri, (Rj)

### CONDITION CODES

C: Not affected

Z: Set if the result is 0;
cleared otherwise

N: Set if the most significant
bit of the result is 1;
cleared otherwise

### OPERATION

$Ri \wedge (Rj)_M$

### DESCRIPTION

Performs logical AND operation between the contents of register Ri and the data in the memory addressed by the contents of register Rj.
When the Rj is an odd-numbered register (j=1,3,5,····,15), the memory address is specified by contents of register pair RjRj-1.
While, when the Rj is an even-numbered register, the lower byte of memory address is specified by the contents of Rj (j=0,2,·····,14) and the upper byte becomes 0.

ADDRESSING MODE & NUMBER OF MACHINE CYCLES

| ADDRESSING MODE | | MNEMONIC | OPERAND TYPE | INSTRUCTION CODE | | NUMBER OF MACHINE CYCLES |
|---|---|---|---|---|---|---|
| S | D | | | Byte 1 | Byte 2 | |
| REG/REGI | - | TST | Ri, (Rj) | 36 | rjri | 4 |

## 4.2 Instruction Set Summary

| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 .0 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | N Z C |
| ADD | ADD Ri, m | 8 $r_i$ mHmL | O | | | | | | Ri + m → Ri | ↕ ↕ ↕*1 |
| | ADD Ri, Rj | 0 0 $r_j r_i$ | | O | | | | | Ri + Rj → Ri | ↕ ↕ ↕ |
| | ADD Ri,(Rj) | 2 0 $r_j r_i$ | | | | O | | | Ri + $(Rj)_M^{*2}$ → Ri | ↕ ↕ ↕ |
| ADC | ADC Ri, m | 9 $r_i$ mHmL | O | | | | | | Ri + m + C → Ri | ↕ ↕ ↕ |
| | ADC Ri, Rj | 0 1 $r_j r_i$ | | O | | | | | Ri + Rj + C → Ri | ↕ ↕ ↕ |
| | ADC Ri,(Rj) | 2 1 $r_j r_i$ | | | | O | | | Ri + $(Rj)_M^{*2}$ + C → Ri | ↕ ↕ ↕ |
| SUB | SUB Ri, Rj | 0 2 $r_j r_i$ | | O | | | | | Ri - Rj → Ri | ↕ ↕ ↕ |
| | SUB Ri,(Rj) | 2 2 $r_j r_i$ | | | | O | | | Ri - $(Rj)_M^{*2}$ → Ri | ↕ ↕ ↕ |
| SBC | SBC Ri, m | B $r_i$ mHmL | O | | | | | | Ri - m - C → Ri | ↕ ↕ ↕ |
| | SBC Ri, Rj | 0 3 $r_j r_i$ | | O | | | | | Ri - Rj - C → Ri | ↕ ↕ ↕ |
| | SBC Ri,(Rj) | 2 3 $r_j r_i$ | | | | O | | | Ri - $(Rj)_M^{*2}$ - C → Ri | ↕ ↕ ↕ |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕ Set or reset depending on the result of instruction execution.  ● : Not change

*2 : j = odd; $(Rj, Rj-1)_M$  j = even ; $(0, Rj)_M$



Hitachi America, Ltd. • Hitachi Plaza • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | N Z C | |
| ADDD | *3 ADDD Ri, Rj | 1 0 rjri | | O | | | | | $R_{i-1}$ + Rj → $R_{i-1}$<br>$R_i$ + Carry → Ri | • ↕ | *1 • |
| | *3 ADDD Ri,(Rj) | 3 0 rjri | | | | O | | | $R_{i-1}$ + $(Rj)_M$ *2 → $R_{i-1}$<br>Ri + Carry → Ri | • ↕ | • |
| SUBD | *3 SUBD Ri, Rj | 1 2 rjri | | O | | | | | $R_{i-1}$ - Rj → $R_{i-1}$<br>$R_i$ - Borrow → $R_i$ | • ↕ | • |
| | *3 SUBD Ri,(Rj) | 3 2 rjri | | | | O | | | $R_{i-1}$ - $(Rj)_M$ *2 → $R_{i-1}$<br>$R_i$ - Borrow → $R_i$ | • ↕ | • |
| TST | TST Ri, Rj | 1 6 rjri | | O | | | | | Ri ∧ Rj | ↕ ↕ | • |
| | TST Ri,(Rj) | 3 6 rjri | | | | O | | | Ri ∧ $(Rj)_M$ *2 | ↕ ↕ | • |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕ Set or reset depending on the result of instruction execution.  • : Not change

*2 : j = odd; $(Rj, Rj-1)_M$  j = even; $(0, Rj)_M$

*3 : Ri; i = odd

HITACHI
Hitachi America, Ltd. • Hitachi Plaza • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 0 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | N | Z | C |
| AND | AND Ri, m | E $r_i$ mHmL | O | | | | | | Ri ∧ m → Ri | ↕ | ↕ | ● *1 |
| | AND Ri, Rj | 0 6 $r_j r_i$ | | O | | | | | Ri ∧ Rj → Ri | ↕ | ↕ | ● |
| | AND Ri,(Rj) | 2 6 $r_j r_i$ | | | | O | | | Ri ∧ $(Rj)_M^{*2}$ → Ri | ↕ | ↕ | ● |
| OR | OR Ri, m | C $r_i$ mHmL | O | | | | | | Ri ∨ m → Ri | ↕ | ↕ | ● |
| | OR Ri, Rj | 0 4 $r_j r_i$ | | O | | | | | Ri ∨ Rj → Ri | ↕ | ↕ | ● |
| | OR Ri,(Rj) | 2 4 $r_j r_i$ | | | | O | | | Ri ∨ $(Rj)_M^{*2}$ → Ri | ↕ | ↕ | ● |
| EOR | EOR Ri, m | D $r_i$ mHmL | O | | | | | | Ri ⊕ m → Ri | ↕ | ↕ | ● |
| | EOR Ri, Rj | 0 5 $r_j r_i$ | | O | | | | | Ri ⊕ Rj → Ri | ↕ | ↕ | ● |
| | EOR Ri,(Rj) | 2 5 $r_j r_i$ | | | | O | | | Ri ⊕ $(Rj)_M^{*2}$ → Ri | ↕ | ↕ | ● |
| CMP | CMP Ri, m | A $r_i$ mHmL | O | | | | | | R:- m | ↕ | ↕ | ↕ |
| | CMP Ri, Rj | 1 3 $r_j r_i$ | | O | | | | | Ri - Rj | ↕ | ↕ | ↕ |
| | CMP Ri, (Rj) | 3 3 $r_j r_i$ | | | | O | | | Ri - $(Rj)_M^{*2}$ | ↕ | ↕ | ↕ |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕:Set or reset depending on the result of instruction execution.  ● : Not change

*2 : j = odd; $(Rj, Rj-1)_M$  j = even; $(0, Rj)_M$



Hitachi America, Ltd. • Hitachi Plaza • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 0 N Z C |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | |
| Shift Left | SL Ri | 0 F 0 $r_i$ | | ○ | | | | |  | *1 ↕ ↕ ↕ |
| Shift Right | SRL Ri | 0 C 0 $r_i$ | | ○ | | | | |  | ↕ ↕ ↕ |
| | SRA Ri | 0 D 0 ri | | ○ | | | | | | ↕ ↕ ↕ |
| Rotate | ROL Ri | 1 F 0 ri | | ○ | | | | |  | ↕ ↕ ↕ |
| | ROR Ri | 1 C 0 $r_i$ | | ○ | | | | | | ↕ ↕ ↕ |
| Increment | INC Ri | 0 B 0 ri | | ○ | | | | | Ri + 1 → Ri | ↕ ↕ • |
| Decrement | DEC Ri | 0 9 0 ri | | ○ | | | | | Ri - 1 → Ri | ↕ ↕ • |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕ Set or reset depending on the result of instruction execution.  • : Not change

| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | N Z | C |
| LOAD | LD Ri, m | F $r_i$ mHmL | O | | | | | | m → Ri | ↕ ↕ | ● *1 |
| | LD Ri,(Rj) | 2 8 $r_j r_i$ | | | | O | | | $(Rj)_M$ *2 → Ri | ↕ ↕ | ● |
| MOVE | MV Ri, Rj | 0 8 $r_j r_i$ | | O | | | | | Rj → Ri | ↕ ↕ | ● |
| STORE | ST (Ri), Rj | 2 E $r_i r_j$ | | | O | | | | Rj → $(Ri)_M$ *2 | ↕ ↕ | ● |
| CCR to R | CTR Ri | 1 A 0 $r_i$ | | O | | | | | CCR → Ri | ● ● | ● |
| R to CCR | RTC Ri | 1 B 0 $r_i$ | | O | | | | | Ri → CCR | ↕ ↕ | ↕ |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕:Set or reset depending on the result of instruction execution.  ● : Not change

*2 : j = odd; $(Rj, Rj-1)_M$  j = even; $(0, Rj)_M$



Hitachi America, Ltd. • Hitachi Plaza • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | N Z C | |
| JUMP | JR g | 4 0 gHgL | | | | | O | | PC + g → PC | • • • | *1 |
| | JR P,g | 4 2 gHgL | | | | | O | | N=0 : PC + g → PC<br>N-1 : continue | • • • | |
| | JR N,g | 4 3 gHgL | | | | | O | | N=1 : PC + g → PC<br>N=0 : continue | • • • | |
| | JR NZ,g | 4 4 gHgL | | | | | O | | Z=0 : PC + g → PC<br>Z=1 : continue | • • • | |
| | JR Z,g | 4 5 gHgL | | | | | O | | Z=1 : PC + g → PC<br>Z=0 : continue | • • • | |
| | JR NC,g | 4 6 gHgL | | | | | O | | C=0 : PC + g → PC<br>C=1 : continue | • • • | |
| | JR C,g | 4 7 gHgL | | | | | O | | C=1 : PC + g → PC<br>C=0 : continue | • • • | |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕:Set or reset depending on the result of instruction execution.  • : Not change

| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 0 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | N | Z | C |
| JUMP | JP (Ri) | 5 0 $r_i$0 | | | | | | O | $R_i$, $R_{i-1}$ *2 → PC | • | • | • *1 |
| | JP P, (Ri) | 5 2 $r_i$0 | | | | | | O | N=0: $R_i$, $R_{i-1}$ *2 → PC<br>N=1: continue | • | • | • |
| | JP N, (Ri) | 5 3 $r_i$0 | | | | | | O | N=1: $R_i$, $R_{i-1}$ *2 → PC<br>N=0: continue | • | • | • |
| | JP NZ, (Ri) | 5 4 $r_i$0 | | | | | | O | Z=0: $R_i$, $R_{i-1}$ *2 → PC<br>Z=1: continue | • | • | • |
| | JP Z, (Ri) | 5 5 $r_i$0 | | | | | | O | Z=1: $R_i$, $R_{i-1}$ *2 → PC<br>Z=0: continue | • | • | • |
| | JP NC, (Ri) | 5 6 $r_i$0 | | | | | | O | C=0: $R_i$, $R_{i-1}$ *2 → PC<br>C=1: continue | • | • | • |
| | JP C, (Ri) | 5 7 $r_i$0 | | | | | | O | C=1: $R_i$, $R_{i-1}$ *2 → PC<br>C=0: continue | • | • | • |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕:Set or reset depending on the result of instruction execution.  • : Not change

*2 : $R_i$, $R_{i-1}$; i = odd



Hitachi America, Ltd. • Hitachi Plaza • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

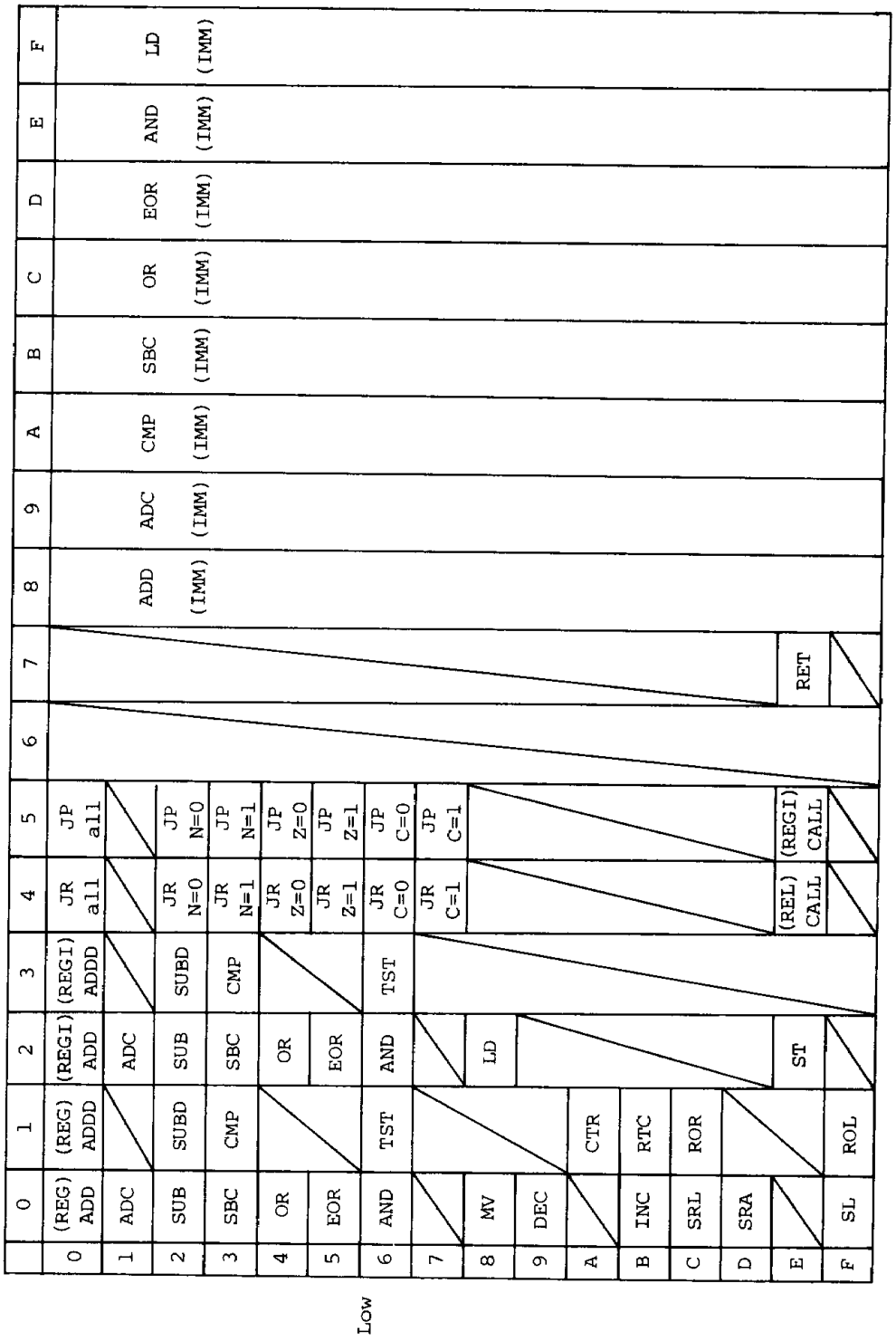
| Operation Name | Mnemonics | OP-code | Addressing | | | | | | Operation | CCR 2 1 0 N Z C |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | IMM → R | R → R | R → M | M → R | REL | IND | | |
| CALL | CALL g | 4 E gHgL | | | | | O | | $PC_L \rightarrow (SP)_M$<br>$PC_H \rightarrow (SP-1)_M$<br>SP - 2 → SP<br>PC + g → PC | *1<br>● ● ● |
| | CALL (Ri) | 5 E $r_i$E | | | | | | O | $PC_L \rightarrow (SP)_M$<br>$PC_H \rightarrow (SP-1)_M$<br>SP - 2 → SP<br>*2<br>$R_i, R_{i-1} \rightarrow PC$ | ● ● ● |
| Return from Subroutine | RET | 7 E 0 E | | | | | | | $(SP+1)_M \rightarrow PC_H$<br>$(SP+2)_M \rightarrow PC_L$<br>SP+2 → SP | ● ● ● |

ri, rj : Register  mHmL : Immediate Data  gHgL : Relative Data

*1 : ↕:Set or reset depending on the result of instruction execution.  ● : Not change

*2 : Ri, Ri-1; i = odd

4.3 Op-code Map






HITACHI
Hitachi America, Ltd. • Hitachi Plaza • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

## 4.4 Precautions for Programming

NOP (No operation): The MCU doesn't have NOP (No operation) instruction. However, the same operation as NOP instruction can be realized by JP instruction. Refer to Figure 4-1 for details.

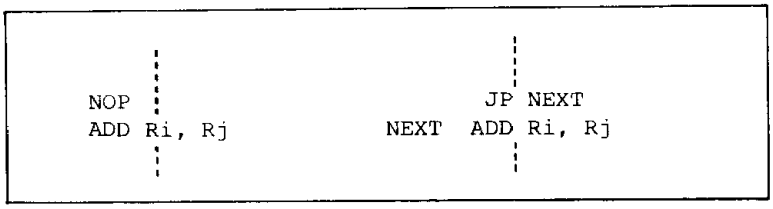



Figure 4-1 No Operation by JP Instruction

Power ON/OFF sequence: Power ON/OFF must be done as follows to prevent a wrong write of EEPROM or latch-up.

Power ON

. Power ON with $\overline{RES}$=CLK="LOW" and I/O="LOW" (or high-impedance).
. Supply the CLK after power up.
. $\overline{RES}$ must be held "LOW" more than 20 CLK cycles.

Power OFF

. $\overline{RES}$="LOW" and I/O="LOW" (or high-impedance)
. CLK="LOW" after holding the $\overline{RES}$="LOW" more than 20 CLK cycles.
. Power OFF within 1ms after CLK="LOW"

Page write of EEPROM: In page write operation, all data written at one time must be in the same page. If the data in different pages are written at one time, all data will be written in one page which is selected by the first data. For example, if the addresses from $1810 to $182F are written at one time, addresses from $1810 to $181F will be written correctly, but the data which must be written to the addresses from $1820 to $182F properly, will be written in $1800 to $180F.

In this case the addresses from $1810 to $181F and $1820 to $182F must be written separately.

# Section 5. ROM Ordering

Concerning ROM ordering, please refer "Single Chip Microcomputer ROM Ordering Manual".

If the ROM code media is an EPROM, please pay attention to the program addresses of the EPROM.

Table 5-1 provides the address correspondence between mask ROM and EPROM. All data in unused addresses must be $FF.

Table 5-1 Address correspondence between mask ROM and EPROM

| Mask ROM Addresses in MCU | | EPROM Addresses | EPROM Type |
|---|---|---|---|
| $3400 | | $1400 | HN482764 |
| to | ⟺ | to | HN27C64 |
| $3FFF | | $1FFF | or equivalents |

# Section 6. Electrical Characteristics

Absolute Maximum Ratings

| Item | Symbol | Value | Unit |
|---|---|---|---|
| Supply voltage | $V_{CC}$ | -0.3 to +7.0 | V |
| Input voltage | Vin | -0.3 to $V_{CC}$+0.3 | V |
| Operating temperature range | Topr | 0 to +50 | °C |
| Storage temperature | Tstg | 0 to +50 | °C |

DC Characteristics ($V_{CC}$=5V±10%, $V_{SS}$=0V, Ta=0 to +50°C)

| Item | | Symbol | Min. | Typ. | Max. | Unit | Test Condition |
|---|---|---|---|---|---|---|---|
| Input high voltage | $\overline{RES}$ | $V_{IH}$ | 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 voltage | $\overline{RES}$ | $V_{IL}$ | -0.3 | | 0.6 | V | |
| | CLK | | -0.3 | | 0.5 | | |
| | I/O Port | | -0.3 | | 0.8 | | |
| Output high voltage | | $V_{OH}$ | 2.4 | | $V_{CC}$ | V | $I_{OH}$ = -100μA |
| | | | 3.8 | | $V_{CC}$ | | $I_{OH}$ = -20μA |
| Output low voltage | | $V_{OL}$ | 0 | | 0.4 | V | $I_{OL}$ = 1mA |
| Input leakage current ($\overline{RES}$, CLK) | | $\|Iin\|$ | | | 10.0 | μA | Vin=0.5 to $V_{CC}$-0.5V |
| Three state leakage current (I/O) | | $\|I_{TL}\|$ | | | 10.0 | μA | Vin=0.5 to $V_{CC}$-0.5V |
| Power dissipation | | Icc | | | 10 | mA | f=5MHz |
| Pin capacitance | | Cp | | | 15 | pF | Vin=0V, f=1MHz Ta=25°C |

# ELECTRICAL CHARACTERISTICS

AC Characteristics ($V_{CC}$=5V±10%, $V_{SS}$=0V, Ta=0 to +50°C)

| Item | Symbol | Min. | Typ. | Max. | Unit | Test Condition |
|---|---|---|---|---|---|---|
| Clock cycle time | $t_{cyc}$ | 0.2 | | 0.33 | µs | Fig. 6-1 |
| Clock pulse width high | $t_{CH}$ | 0.4 | | 0.6 | $t_{cyc}$ | Fig. 6-1 |
| Clock pulse width low | $t_{CL}$ | 0.4 | | 0.6 | $t_{cyc}$ | Fig. 6-1 |
| Clock fall time | $t_{Cf}$ | | | 20 | ns | Fig. 6-1 |
| Clock rise time | $t_{Cr}$ | | | 20 | ns | Fig. 6-1 |
| I/O port fall time | $t_f$ | | | 1.0 | µs | Fig. 6-2 |
| I/O port rise time | $t_r$ | | | 1.0 | µs | Fig. 6-2 |
| $\overline{RES}$ pulse width | $t_{RWL}$ | 20 | | | $t_{cyc}$ | Fig. 2-12 |
| EEPROM erase/write time | $t_{EPW}$ | | 10 | 15 | ms | Fig. 3-6 |

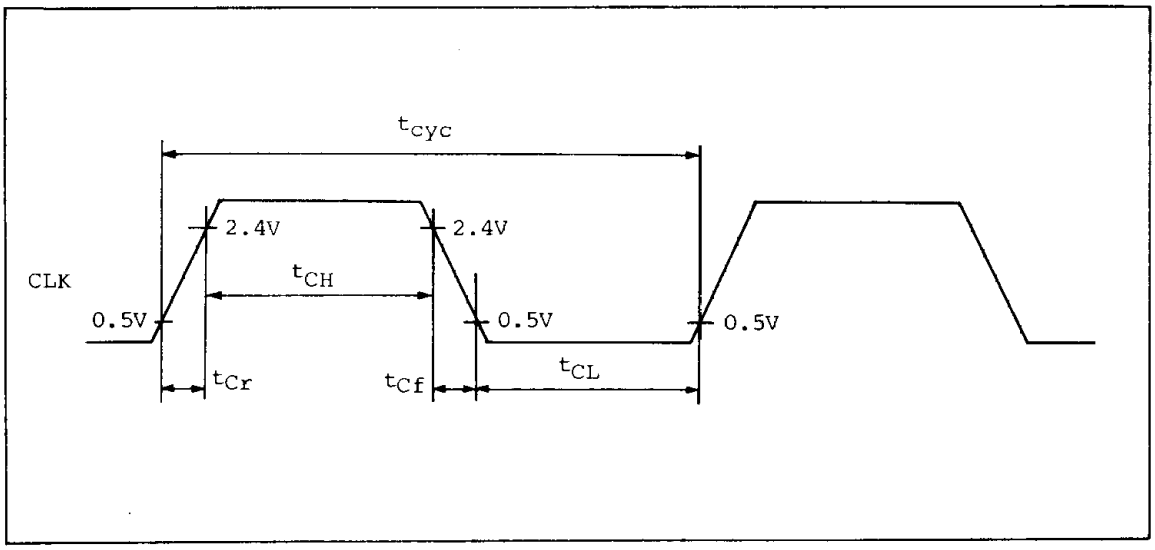



Figure 6-1 CLK Input Wave Form

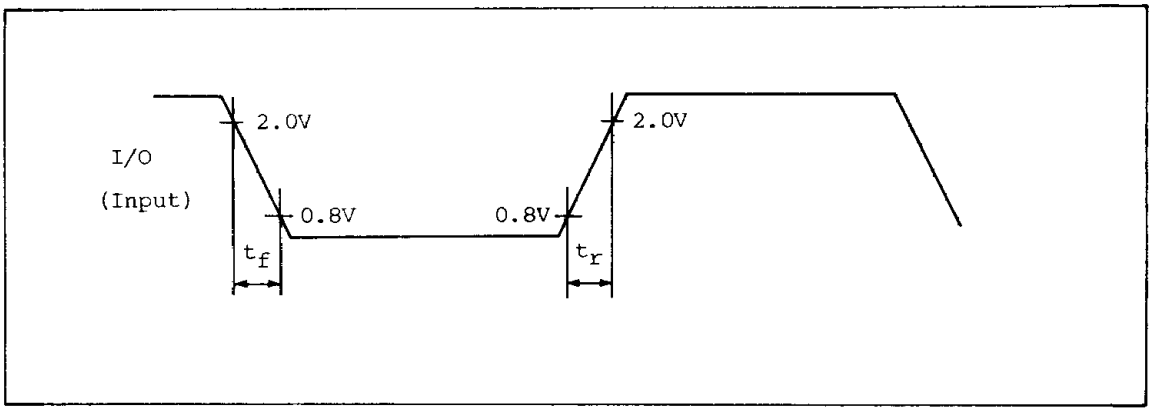



Figure 6-2 I/O Input Wave Form