To all our customers

# Regarding the change of names mentioned in the document, such as Mitsubishi Electric and Mitsubishi XX, to Renesas Technology Corp.

The semiconductor operations of Hitachi and Mitsubishi Electric were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Mitsubishi Electric, Mitsubishi Electric Corporation, Mitsubishi Semiconductors, and other Mitsubishi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Note : Mitsubishi Electric will continue the business operations of high frequency & optical devices and power devices.

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

RENESAS
Renesas Technology Corp.

# 4570 Group

SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## DESCRIPTION

The 4570 Group is a 4-bit single-chip microcomputer designed with CMOS technology. Its CPU is that of the 4500 series using a simple, high-speed instruction set. The computer is equipped with a carrier wave output circuit for remote control, an 8-bit timer with a reload register, a 10-bit timer with a reload register, and an 8-bit timer with two reload registers.

The various microcomputers in the 4570 Group include variations of the built-in memory size. The mask ROM version and One Time PROM version of 4570 Group are produced as shown in the table below.

## FEATURES

- Minimum instruction execution time
  - When f($X_{IN}$) is selected for system clock ..... 1.5$\mu$s ($f(X_{IN})$=2.0 MHz, $V_{DD}$=4.5 V to 5.5 V)
  - When f($X_{IN}$)/4 is selected for system clock ..... 2.86$\mu$s ($f(X_{IN})$=4.2 MHz, $V_{DD}$=2.0 V to 5.5 V)
- Supply voltage
  - ..... 2.5 V to 5.5 V (One Time PROM version)
  - ..... 2.0 V to 5.5 V (Mask ROM version)
- System clock switch function
  - ..... f($X_{IN}$)/4 or not divided
- Timers
  - Timer 1... 10-bit timer with a reload register and carrier wave output auto-control function
  - Timer 2 ..... 8-bit timer with a reload register
  - Timer 3... 8-bit timer with two reload registers and carrier wave generation function
- Interrupt ..... 4 sources
- Power-on reset circuit
- Watchdog timer ..... 16 bits
- Key-on wakeup function (Ports P0, P1, and P4, ON/OFF of port P4 can be switched)
- Pull-up transistor ..... (Ports P0, P1, and P4, ON/OFF of port P4 can be switched)
- Voltage drop detection circuit
- Clock generating circuit (ceramic resonance)

## APPLICATION

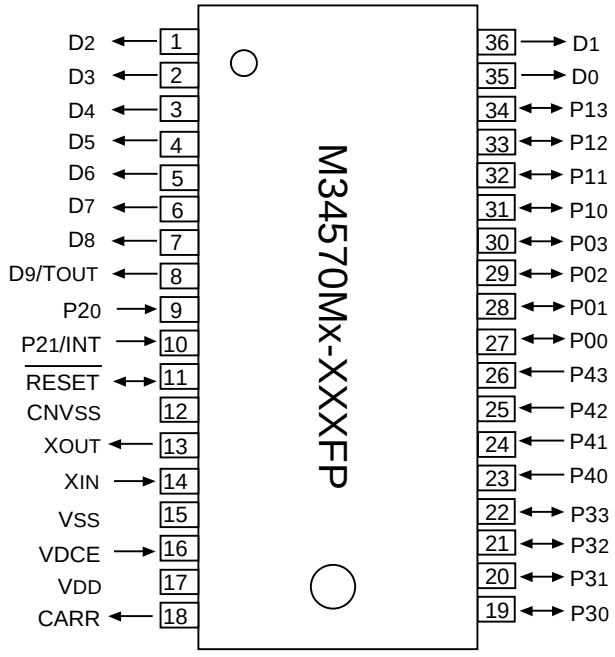
Remote control transmitter

| Product | ROM (PROM) size (× 10 bits) | RAM size (× 4 bits) | Package | ROM type |
|---|---|---|---|---|
| M34570M4-XXXFP | 4096 words | 128 words | 36P2R-A | Mask ROM |
| M34570M8-XXXFP | 8192 words | 128 words | 36P2R-A | Mask ROM |
| M34570MD-XXXFP | 16384 words | 128 words | 36P2R-A | Mask ROM |
| M34570E8FP | 8192 words | 128 words | 36P2R-A | One Time PROM |
| M34570EDFP * | 16384 words | 128 words | 36P2R-A | One Time PROM |

*: Under development (Jan. 1999)

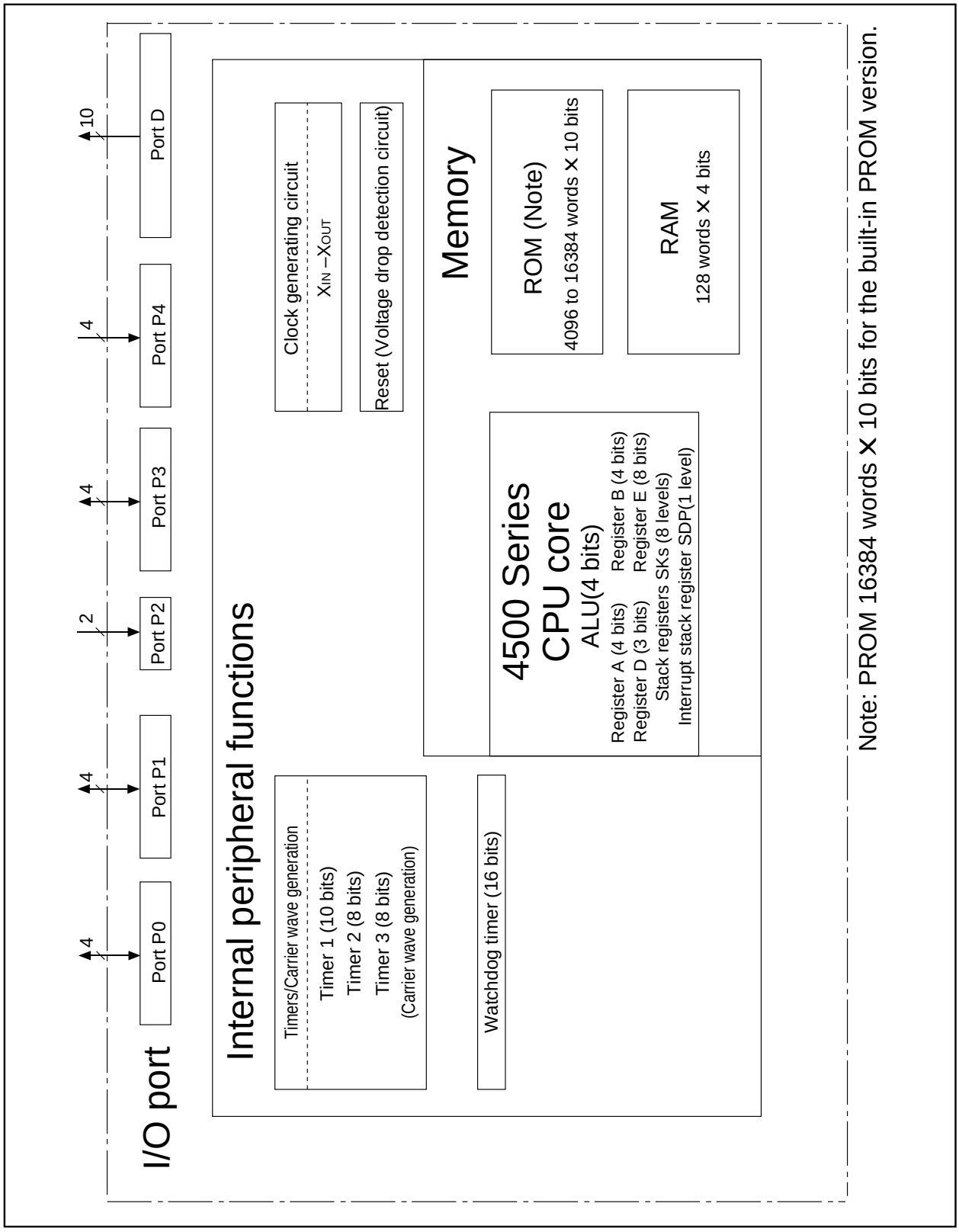
## PIN CONFIGURATION (TOP VIEW)

M34570Mx-XXXFP




Outline 36P2R-A

## BLOCK DIAGRAM



## PERFORMANCE OVERVIEW

| Parameter | | | Function |
|---|---|---|---|
| Number of basic instructions | | | 99 |
| Minimum instruction execution time | | | 1.5 $\mu$s (f(XIN) = 2.0 MHz:system clock = f(XIN): VDD = 5.0 V) |
| | | | 2.86 $\mu$s (f(XIN) = 4.2 MHz:system clock = f(XIN)/4: VDD = 5.0 V) |
| Memory sizes | ROM | M34570M4 | 4096 words × 10 bits |
| | | M34570M8 | 8192 words × 10 bits |
| | | M34570MD | 16384 words × 10 bits |
| | | M34570E8 | 8192 words × 10 bits |
| | | M34570ED | 16384 words × 10 bits |
| | RAM | | 128 words × 4 bits |
| Input/Output ports | D0–D9 | Output | Ten independent output ports; port D9 is also used as the TOUT output pin. |
| | P00–P03 | I/O | 4-bit I/O port; every pin of the ports has a key-on wakeup function and a pull-up function. |
| | P10–P13 | I/O | 4-bit I/O port; every pin of the ports has a key-on wakeup function and a pull-up function. |
| | P20, P21 | Input | 2-bit input port, port P21 is also used as INT input pin. |
| | P30–P33 | I/O | 4-bit I/O port |
| | P40–P43 | Input | 4-bit input port; both pull-up function and key-on wakeup function can be switched by software. |
| | CARR | Output | 1-bit output port (CMOS output) |
| | TOUT | Output | 1-bit output pin; TOUT output pin is also used as port D9. |
| | INT | Input | 1-bit input pin with a key-on wakeup function. INT input pin is also used as port P21. |
| Timers | Timer 1 | | 10-bit timer with a reload register and carrier wave output auto-control function |
| | Timer 2 | | 8-bit timer with a reload register |
| | Timer 3 | | 8-bit timer with two reload registers and carrier wave generation function |
| Interrupt | Sources | | 4 (one for external and three for timer) |
| | Nesting | | 1 level |
| Subroutine nesting | | | 8 levels (however, only 7 levels can be used when an interrupt is used or the TABP p instruction is executed) |
| Device structure | | | CMOS silicon gate |
| Package | | | 36-pin plastic molded SSOP |
| Operating temperature range | | | –20 °C to 70 °C |
| Supply voltage | | | 2.0 V to 5.5 V for mask ROM version (2.5 V to 5.5 V for One Time PROM version) |
| Power dissipation (typical value) | at active | | 1.3 mA (f(XIN) = 4.2 MHz: system clock = f(XIN)/4, VDD=5.0 V) |
| | | | 0.5 mA (f(XIN) = 1.0 MHz: system clock = f(XIN), VDD=3.0 V) |
| | at RAM back-up | | 0.1 $\mu$A (Ta=25 °C, VDD=5V, typical value) |

## DEFINITION OF CLOCK AND CYCLE

● System clock
The system clock is the basic clock for controlling this product. The system clock can be selected by bit 3 of the clock control register MR as shown in the table below.

**Table Selection of system clock**

| MR3 | System clock |
|---|---|
| 0 | f(XIN) |
| 1 | f(XIN)/4 |

Note: f(XIN)/4 is selected immediately after system is released from reset.

● Instruction clock
The instruction clock is the standard clock for controlling CPU. The instruction clock is a signal derived from dividing the system clock by 3. The one cycle of the instruction clock is equivalent to the one machine cycle.

● Machine cycle
The machine cycle is the standard cycle required to execute the instruction.

## PIN DESCRIPTION

| Pin | Name | Input/Output | Function |
|---|---|---|---|
| $V_{DD}$ | Power supply | — | Connected to a plus power supply. |
| $V_{SS}$ | Ground | — | Connected to a 0 V power supply. |
| $CNV_{SS}$ | $CNV_{SS}$ | Input | Connect $CNV_{SS}$ to $V_{SS}$ and apply "L" (0V) to $CNV_{SS}$ certainly. |
| $\overline{RESET}$ | Reset input | I/O | An N-channel open-drain I/O pin for a system reset. A pull-up transistor and a capacitor are built-in this pin. When the watchdog timer causes the system to be reset or the low-supply voltage is detected, the $\overline{RESET}$ pin outputs "L" level. |
| $X_{IN}$ | Clock input | Input | I/O pins of the clock generating circuit. Connect a ceramic resonator between $X_{IN}$ pin and $X_{OUT}$ pin. A feedback resistor is built-in between them. |
| $X_{OUT}$ | Clock output | Output | |
| $D_0$–$D_9$ | Output port D | Output | Each pin of port D has an independent 1-bit wide output function. Port $D_9$ is also used as $T_{OUT}$ output pin. The output structure is N-channel open-drain. |
| $P0_0$–$P0_3$ | I/O port P0 | I/O | 4-bit I/O port. It can be used as an input port when the output latch is set to "1." The output structure is N-channel open-drain. Every pin of the ports has a key-on wakeup function and a pull-up function. |
| $P1_0$–$P1_3$ | I/O port P1 | I/O | 4-bit I/O port. It can be used as an input port when the output latch is set to "1." The output structure is N-channel open-drain. Every pin of the ports has a key-on wakeup function and a pull-up function. |
| $P2_0$, $P2_1$ | Input port P2 | I/O | 2-bit input port. Port $P2_1$ is also used as the INT input pin. |
| $P3_0$–$P3_3$ | I/O port P3 | I/O | 4-bit I/O port. It can be used as an input port when the output latch is set to "1." The output structure is N-channel open-drain. |
| $P4_0$–$P4_3$ | Input port P4 | Input | 4-bit input port. Every pin of the ports has a key-on wakeup function and a pull-up function. Both functions can be switched by software. |
| CARR | Carrier wave output for remote control | Output | Carrier wave output pin for remote control transmit. The output structure is the CMOS circuit. |
| INT | Interrupt input | Input | INT input pin accepts an external interrupt and has a key-on wakeup function. INT input pin is also used as port $P2_1$. |
| $T_{OUT}$ | Timer output | Output | $T_{OUT}$ output pin has the function to output the timer 2 underflow signal divided by 2. $T_{OUT}$ output pin is also used as port $D_9$. |
| VDCE | Voltage drop detection circuit enable | Input | VDCE pin is used to control the operation/stop of the voltage drop detection circuit. The circuit is operating when "H" level is input to the VDCE pin. It is stopped when "L" level is input to this pin. |

## MULTIFUNCTION

| Pin | Multifunction | Pin | Multifunction |
|---|---|---|---|
| $D_9$ | $T_{OUT}$ | $T_{OUT}$ | $D_9$ |
| $P2_1$ | INT | INT | $P2_1$ |

Notes 1: Pins except above have just single function.
2: The port $D_9$ is the output port and port $P2_1$ is the input port.

## CONNECTIONS OF UNUSED PINS

| Pin | Connection | Pin | Connection |
|---|---|---|---|
| $D_0$–$D_8$<br>$D_9/T_{OUT}$ | Connect to $V_{SS}$, or set the output latch to "0" and open. | $P3_0$–$P3_3$ | Connect to $V_{SS}$, or set the output latch to "0" and open. |
| $P0_0$–$P0_3$<br>$P1_0$–$P1_3$ | Set the output latch to "1" and open. | $P4_0$–$P4_3$ | Connect to $V_{SS}$ (Note 2) or open (Note 3). |
| | | CARR | Open. |
| $P2_0$, $P2_1$/INT | Connect to $V_{SS}$ (Note 1). | | |

Notes 1: When the $P2_1$/INT pin is connected to $V_{SS}$ pin, set the return level to "H" level by software (interrupt control register $I1_2$="1"). When the $P2_1$/INT pin is connected to $V_{SS}$ pin while the return level is set to "L" level, system returns from RAM back-up state immediately after system enters the RAM back-up state.
2: In order to connect ports $P4_0$–$P4_3$ to $V_{SS}$, turn off their pull-up transistors (pull-up control register PU0i="0") by software and also invalidate the key-on wakeup functions (key-on wakeup control register K0i="0"). When these pins are connected to $V_{SS}$ while the key-on wakeup functions are left valid, the system fails to return from RAM back-up state. In order to make these pins open, turn on their pull-up transistors (register PU0i="1") by software (i = 0, 1, 2, 3).
Be sure to select the key-on wakeup function and the pull-up function with every one port.
3: In order to make ports $P4_0$–$P4_3$ open, turn on their pull-up transistors (register PU0i = "1") by software (i = 0, 1, 2, 3).

(Note in order to set the output latch to "0" or "1" or make pins open)
- After system is released from reset, a port is in a high-impedance state until the output latch of the port is set to "0" by software. Accordingly, the voltage level of pins is undefined and the excess of the supply current may occur.
- To set the output latch periodically is recommended because the value of output latch may change by noise or a program run away (caused by noise).

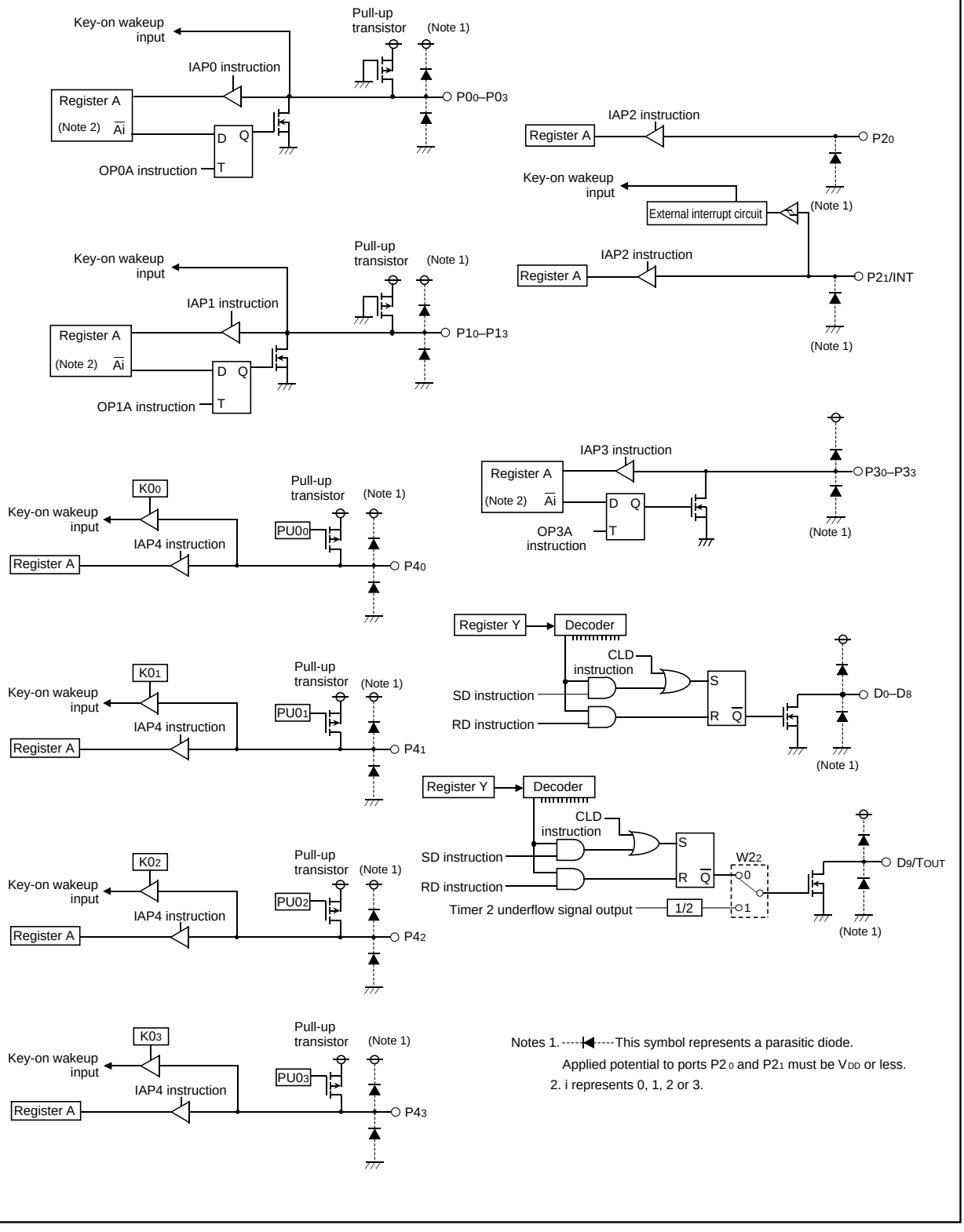
(Note in order to connect unused pins to $V_{SS}$)
- To avoid noise, connect the unused pins to $V_{SS}$ at the shortest distance using a thick wire.

## PORT FUNCTION

| Port | Pin | Input/ Output | Output structure | Control bits | Control instructions | Control registers | Remark |
|---|---|---|---|---|---|---|---|
| Port D | $D_0$–$D_8$, $D_9/T_{OUT}$ | Output (10) | N-channel open-drain | 1 | SD<br>RD<br>CLD | $W2_2$ | $W2_2$ controls the switch of $D_9/T_{OUT}$ pin |
| Port P0 | $P0_0$–$P0_3$ | I/O (4) | N-channel open-drain | 4 | OP0A<br>IAP0 | | Pull-up functions<br>Key-on wakeup functions |
| Port P1 | $P1_0$–$P1_3$ | I/O (4) | N-channel open-drain | 4 | OP1A<br>IAP1 | | Pull-up functions<br>Key-on wakeup functions |
| Port P2 | $P2_0$ | Input (2) | | 2 | IAP2<br>SNZI0<br>(Note) | | |
| | $P2_1$/INT | | | | | | Key-on wakeup function |
| Port P3 | $P3_0$–$P3_3$ | I/O | N-channel open-drain | 4 | OP3A<br>IAP3 | | |
| Port P4 | $P4_0$–$P4_3$ | Input (4) | | 4 | IAP4 | PU0<br>K0 | Pull-up functions (programmable)<br>Key-on wakeup functions (programmable) |

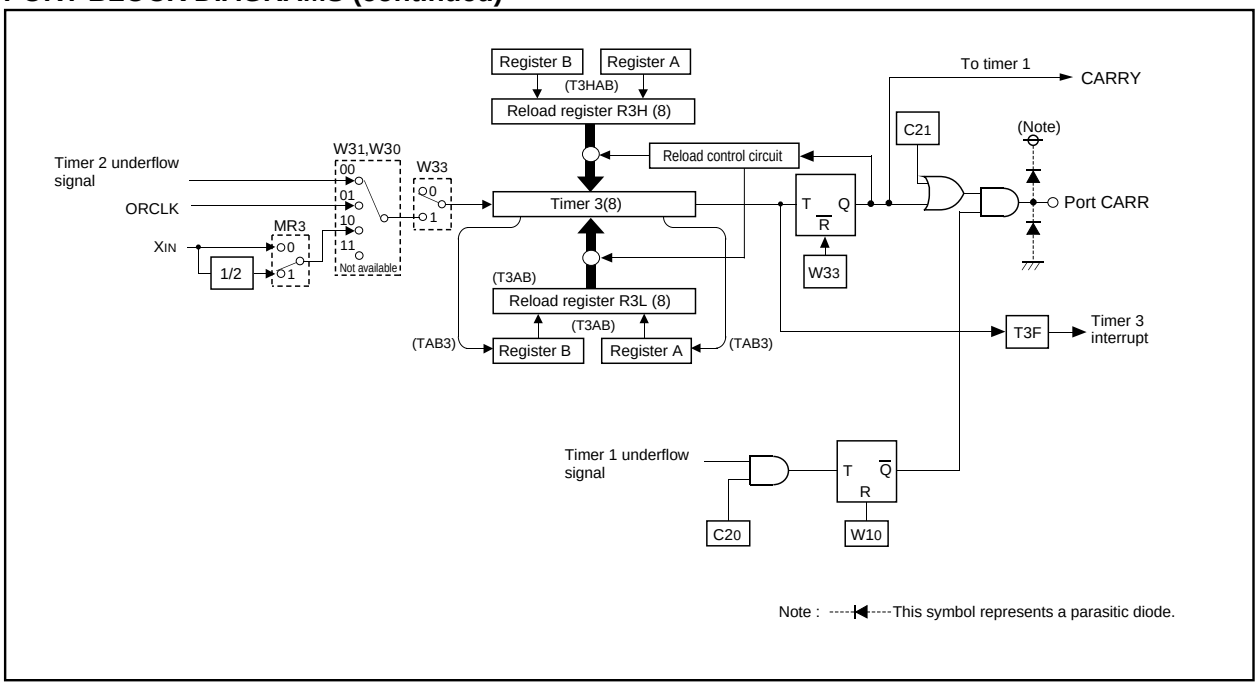
Note: Level of the $P2_1$/INT pin can be examined with the SNZI0 instruction.

## PORT BLOCK DIAGRAMS




Notes 1. This symbol represents a parasitic diode.
Applied potential to ports P20 and P21 must be $V_{DD}$ or less.
2. i represents 0, 1, 2 or 3.

## PORT BLOCK DIAGRAMS (continued)



## FUNCTION BLOCK OPERATIONS
## CPU

**(1) Arithmetic logic unit (ALU)**

The arithmetic logic unit ALU performs 4-bit arithmetic such as 4-bit data addition, comparison, AND operation, OR operation, and bit manipulation.

**(2) Register A and carry flag (CY)**

Register A is a 4-bit register used for arithmetic, transfer, exchange, and I/O operation.

Carry flag CY is a 1-bit flag that is set to "1" when there is a carry with the AMC instruction (Figure 1).

It is unchanged with both A n instruction and AM instruction.

The value of $A_0$ is stored in carry flag CY with the RAR instruction (Figure 2).

Carry flag CY can be set to "1" with the SC instruction and cleared to "0" with the RC instruction.

**(3) Registers B and E**

Register B is a 4-bit register used for temporary storage of 4-bit data, and for 8-bit data transfer together with register A.

Register E is an 8-bit register. It can be used for 8-bit data transfer with register B used as the high-order 4 bits and register A as the low-order 4 bits (Figure 3).

**(4) Register D**

Register D is a 3-bit register.

It is used to store a 7-bit ROM address together with register A and is used as a pointer within the specified page when the TABP p, BLA p, or BMLA p instruction is executed (Figure 4).

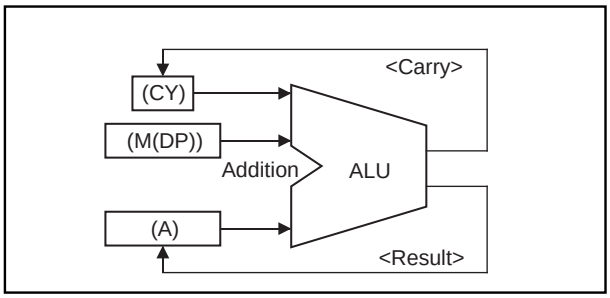



**Fig. 1 AMC instruction execution example**

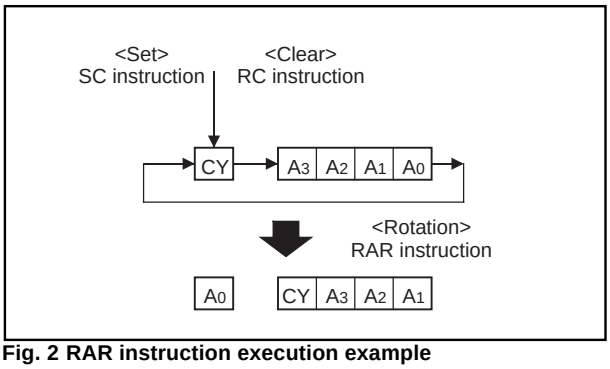



**Fig. 2 RAR instruction execution example**

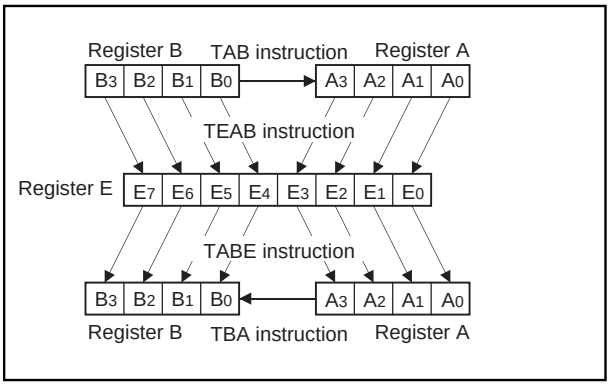



**Fig. 3 Registers A, B and register E**

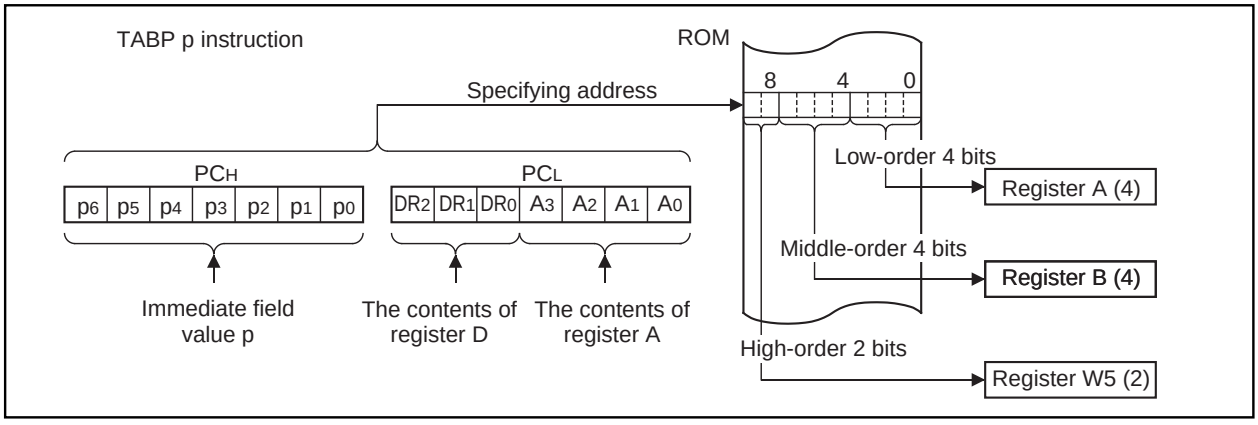



**Fig. 4 TABP p instruction execution example**

**(5) Stack registers (SKs) and stack pointer (SP)**

Stack registers (SKs) are used to temporarily store the contents of program counter (PC) just before branching until returning to the original routine when;

- branching to an interrupt service routine (referred to as an interrupt service routine),
- performing a subroutine call, or
- executing the table reference instruction (TABP p).

Stack registers (SKs) are eight identical registers, so that subroutines can be nested up to 8 levels. However, one of stack registers is used when using an interrupt service routine or when executing a table reference instruction. Accordingly, be careful not to stack over when performing these operations together. The contents of registers SKs are destroyed when 8 levels are exceeded.

The register SK nesting level is pointed automatically by 3-bit stack pointer (SP). The contents of the stack pointer (SP) can be transferred to register A with the TASP instruction.

Figure 5 shows the stack registers (SKs) structure.

Figure 6 shows the example of operation at subroutine call.

**(6) Interrupt stack register (SDP)**

Interrupt stack register (SDP) is a 1-stage register. When an interrupt occurs, this register (SDP) is used to temporarily store the contents of data pointer, carry flag, skip flag, register A, and register B just before an interrupt until returning to the original routine.

Unlike the stack registers (SKs), this register (SDP) is not used when executing the subroutine call instruction and the table reference instruction.

**(7) Skip flag**

Skip flag controls skip decision for the conditional skip instructions and continuous described skip instructions. When an interrupt occurs, the contents of skip flag is stored automatically in the interrupt stack register (SDP) and the skip condition is retained.

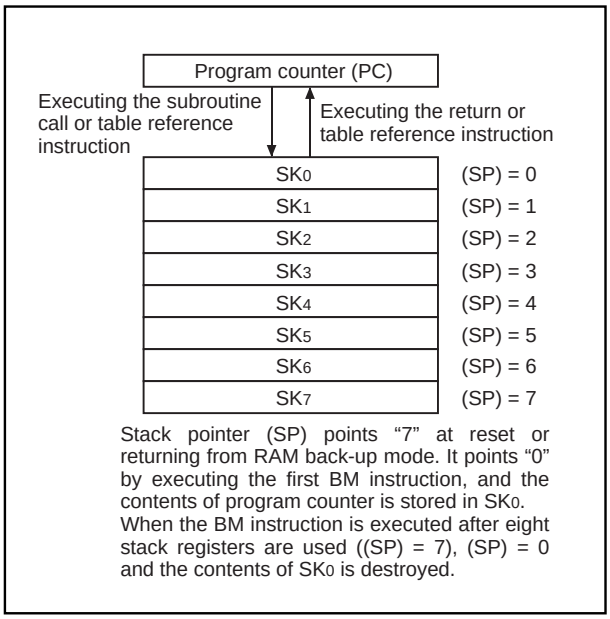



**Fig. 5 Stack registers (SKs) structure**

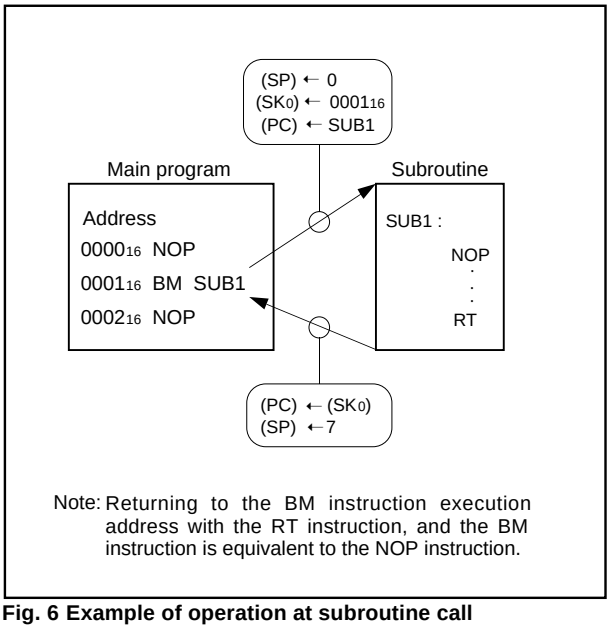



**Fig. 6 Example of operation at subroutine call**

**(8) Program counter (PC)**

Program counter (PC) is used to specify a ROM address (page and address). It determines a sequence in which instructions stored in ROM are read. It is a binary counter that increments the number of instruction bytes each time an instruction is executed. However, the value changes to a specified address when branch instructions, subroutine call instructions, return instructions, or the table reference instruction (TABP p) is executed.

Program counter consists of $PC_H$ (most significant bit to bit 7) which specifies to a ROM page and $PC_L$ (bits 6 to 0) which specifies an address within a page. After it reaches the last address (address 127) of a page, it specifies address 0 of the next page (Figure 7).

Make sure that the $PC_H$ does not specify after the last page of the built-in ROM.

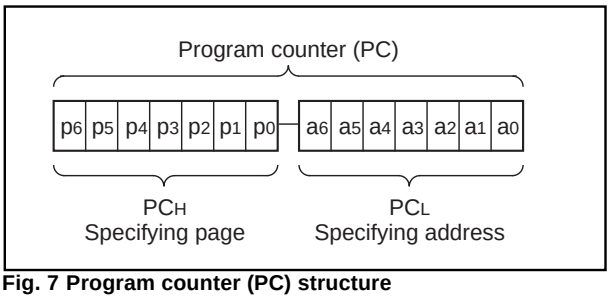



**Fig. 7 Program counter (PC) structure**

**(9) Data pointer (DP)**

Data pointer (DP) is used to specify a RAM address and consists of registers Z, X, and Y. Register Z specifies a RAM file group, register X specifies a file, and register Y specifies a RAM digit (Figure 8).

Register Y is also used to specify the port D bit position. When using port D, set the port D bit position to register Y certainly and execute the SD or RD instruction (Figure 9).

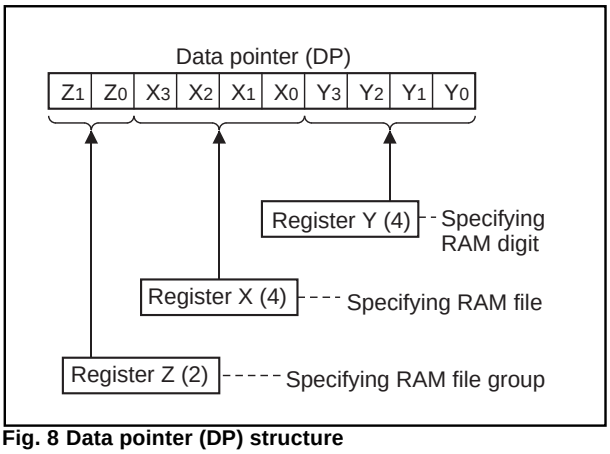



**Fig. 8 Data pointer (DP) structure**

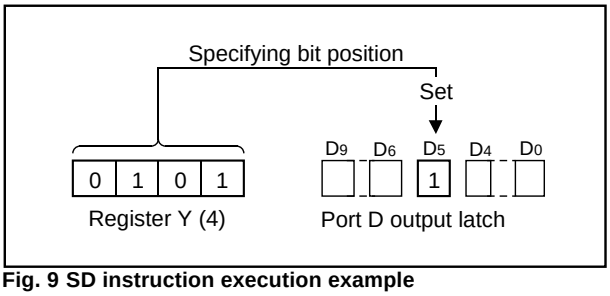



**Fig. 9 SD instruction execution example**

## PROGRAM MEMORY (ROM)

1 word of ROM is composed of 10 bits. ROM is separated every 128 words by the unit of page (addresses 0 to 127). Table 1 shows the ROM size and pages. Figure 10 shows the ROM map of M34570M8.

**Table 1 ROM size and pages**

| Product | ROM size (× 10 bits) | Pages |
|---|---|---|
| M34570M4 | 4096 words | 32 (0 to 31) |
| M34570M8 | 8192 words | 64 (0 to 63) |
| M34570E8 | 8192 words | 64 (0 to 63) |
| M34570MD | 16384 words | 128 (0 to 127) |
| M34570ED | 16384 words | 128 (0 to 127) |

Note: When the TABP instruction is executed after executing the SBK instruction, data in pages 64 to 127 can be referred. When the TABP instruction is executed after executing the RBK instruction, data in pages 0 to 63 can be referred.

A top part of page 1 (addresses $0080_{16}$ to $00FF_{16}$) is reserved for interrupt addresses (Figure 11). When an interrupt occurs, the address (interrupt address) corresponding to each interrupt is set in the program counter, and the instruction at the interrupt address is executed. When using an interrupt service routine, write the instruction generating the branch to that routine at an interrupt address.

Page 2 (addresses $0100_{16}$ to $017F_{16}$) is the special page for subroutine calls. Subroutines written in this page can be called from any page with the 1-word instruction (BM). Subroutines extending from page 2 to another page can also be called with the BM instruction when it starts on page 2.

ROM pattern (bits 9 to 0) of all addresses can be used as data areas with the TABP p instruction.

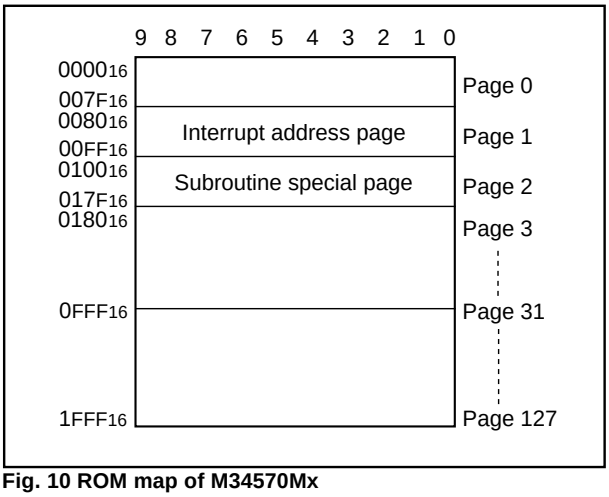



**Fig. 10 ROM map of M34570Mx**

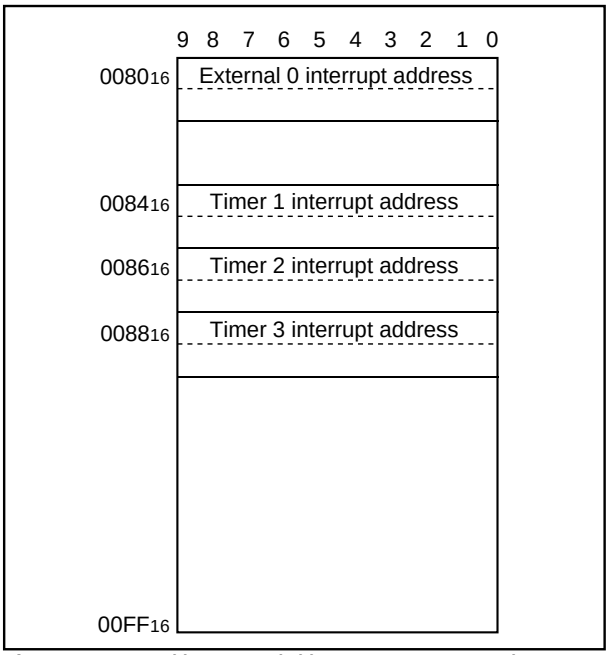



**Fig. 11 Interrupt address page (addresses $0080_{16}$ to $00FF_{16}$) structure**

## DATA MEMORY (RAM)

1 word of RAM is composed of 4 bits, but 1-bit manipulation (with the SB j, RB j, and SZB j instructions) is enabled for the entire memory area. A RAM address is specified by a data pointer. The data pointer consists of registers Z, X, and Y. Set a value to the data pointer certainly when executing an instruction to access RAM.
Table 2 shows the RAM size. Figure 12 shows the RAM map.

**Table 2 RAM size**

| Product | RAM size |
|---|---|
| M34570Mx | 128 words × 4 bits (512 bits) |
| M34570Ex | |

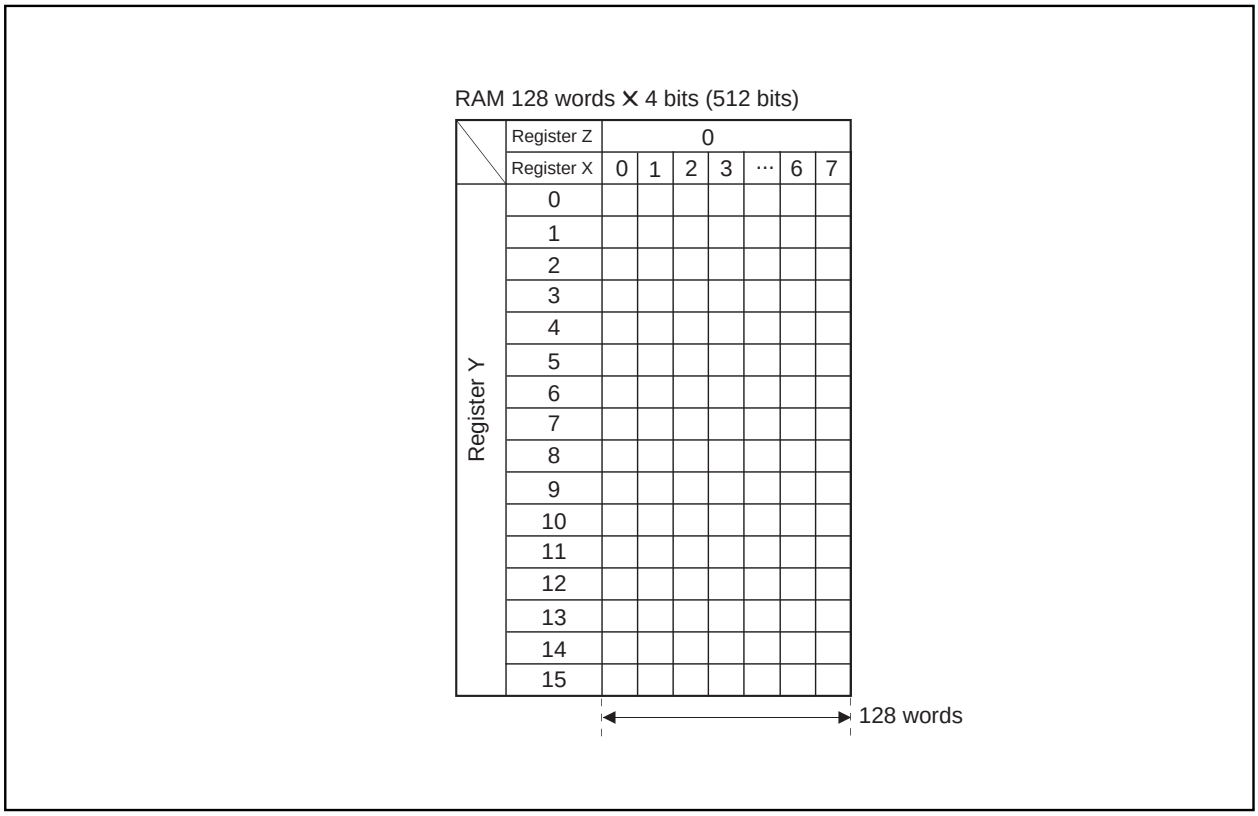



**Fig. 12 RAM map**

## INTERRUPT FUNCTION

The interrupt type is a vectored interrupt branching to an individual address (interrupt address) according to each interrupt source. An interrupt occurs when the following 3 conditions are satisfied.

- Interrupt enable flag (INTE) = "1" (Interrupt enabled)
- Interrupt enable bit = "1" (Interrupt request occurrence enabled)
- An interrupt activated condition is satisfied (request flag = "1")

Table 3 shows interrupt sources. (Refer to each interrupt request flag for details of activated conditions.)

**(1) Interrupt enable flag (INTE)**

The interrupt enable flag (INTE) controls whether the every interrupt enable/disable. Interrupts are enabled when INTE flag is set to "1" with the EI instruction and disabled when INTE flag is cleared to "0" with the DI instruction. When any interrupt occurs, the INTE flag is automatically cleared to "0," so that other interrupts are disabled until the EI instruction is executed.

**(2) Interrupt enable bits ($V1_0$–$V1_3$, $V2_0$–$V2_3$)**

Use an interrupt enable bit of interrupt control registers V1 and V2 to select the corresponding interrupt request or skip instruction.

Table 4 shows the interrupt request flag, interrupt enable bit and skip instruction.

Table 5 shows the interrupt enable bit function.

**(3) Interrupt request flag**

When the activated condition for each interrupt is satisfied, the corresponding interrupt request flag is set to "1." Each interrupt request flag is cleared to "0" when either;

- an interrupt occurs, or
- the next instruction is skipped with a skip instruction.

Each interrupt request flag is set when the activated condition is satisfied even if the interrupt is disabled by the INTE flag or its interrupt enable bit. Once set, the interrupt request flag retains set until a clear condition is satisfied.

Accordingly, an interrupt occurs when the interrupt disable state is released while the interrupt request flag is set.

If more than one interrupt request flag is set when the interrupt disable state is released, the interrupt priority level is as follows shown in Table 3.

**Table 3 Interrupt sources**

| Priority level | Interrupt name | Activated condition | Interrupt address |
|---|---|---|---|
| 1 | External 0 interrupt | Level change of INT pin | Address 0 in page 1 |
| 2 | Timer 1 interrupt | Timer 1 underflow | Address 4 in page 1 |
| 3 | Timer 2 interrupt | Timer 2 underflow | Address 6 in page 1 |
| 4 | Timer 3 interrupt | Timer 3 underflow | Address 8 in page 1 |

**Table 4 Interrupt request flag, interrupt enable bit and skip instruction**

| Interrupt name | Request flag | Enable bit | Skip instruction |
|---|---|---|---|
| External 0 interrupt | EXF0 | $V1_0$ | SNZ0 |
| Timer 1 interrupt | T1F | $V1_2$ | SNZT1 |
| Timer 2 interrupt | T2F | $V1_3$ | SNZT2 |
| Timer 3 interrupt | T3F | $V2_0$ | SNZT3 |

**Table 5 Interrupt enable bit function**

| Interrupt enable bit | Occurrence of interrupt request | Skip instruction |
|---|---|---|
| 1 | Enabled | Invalid |
| 0 | Disabled | Valid |

**(4) Internal state during an interrupt**

The internal state of the microcomputer during an interrupt is as follows (Figure 14).

- Program counter (PC)
  An interrupt address is set in program counter. The address to be executed when returning to the main routine is automatically stored in the stack register (SK).
- Interrupt enable flag (INTE)
  INTE flag is cleared to "0" so that interrupts are disabled.
- Interrupt request flag
  Only the request flag for the current interrupt source is cleared to "0."
- Data pointer, carry flag, skip flag, registers A and B
  The contents of these registers and flags are stored automatically in the interrupt stack register (SDP).

**(5) Interrupt processing**

When an interrupt occurs, a program at an interrupt address is executed after a branch to a sequence for storing data into stack register is performed. Write the branch instruction to an interrupt service routine at an interrupt address.

Use the RTI instruction to return to main routine.

Interrupt enabled by executing the EI instruction is performed after executing 1 instruction (just after the next instruction is executed). Accordingly, when the EI instruction is executed just before the RTI instruction, interrupts are enabled after returning to the main routine. (Refer to Figure 13)

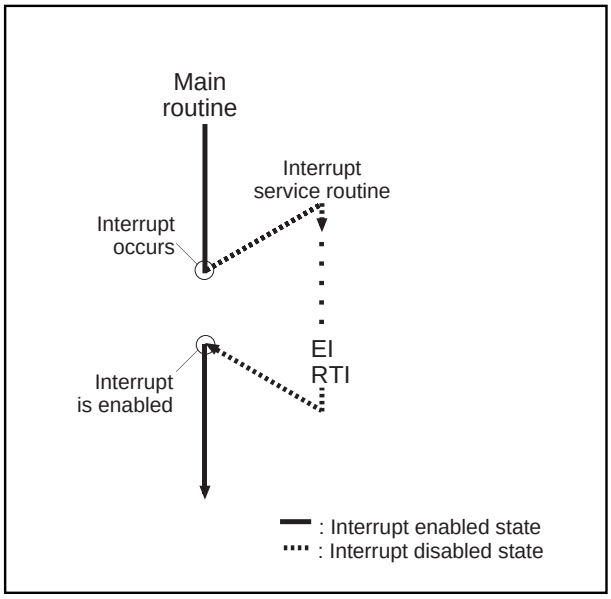



**Fig. 13 Program example of interrupt processing**

•Program counter (PC)
..... Each interrupt address

•Stack register (SK)
..... The address of main routine to be executed when returning

•Interrupt enable flag (INTE)
..... 0 (Interrupt disabled)

•Interrupt request flag (only the flag for the current interrupt source)
..... 0

•Data pointer, carry flag, registers A and B, skip flag
..... Stored in the interrupt stack register (SDP) automatically

**Fig. 14 Internal state when interrupt occurs**

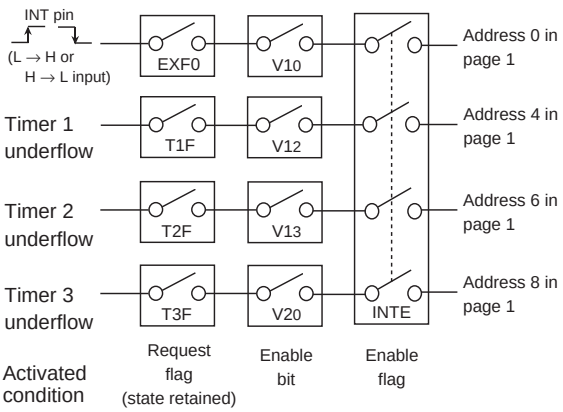



**Fig. 15 Interrupt system diagram**

**(6) Interrupt control register**

● Interrupt control register V1

Interrupt enable bits of external 0, timer 1 and timer 2 are assigned to register V1. Set the contents of this register through register A with the TV1A instruction. The TAV1 instruction can be used to transfer the contents of register V1 to register A.

● Interrupt control register V2

Interrupt enable bit of timer 3 is assigned to register V2. Set the contents of this register through register A with the TV2A instruction. The TAV2 instruction can be used to transfer the contents of register V2 to register A.

**Table 6 Interrupt control register**

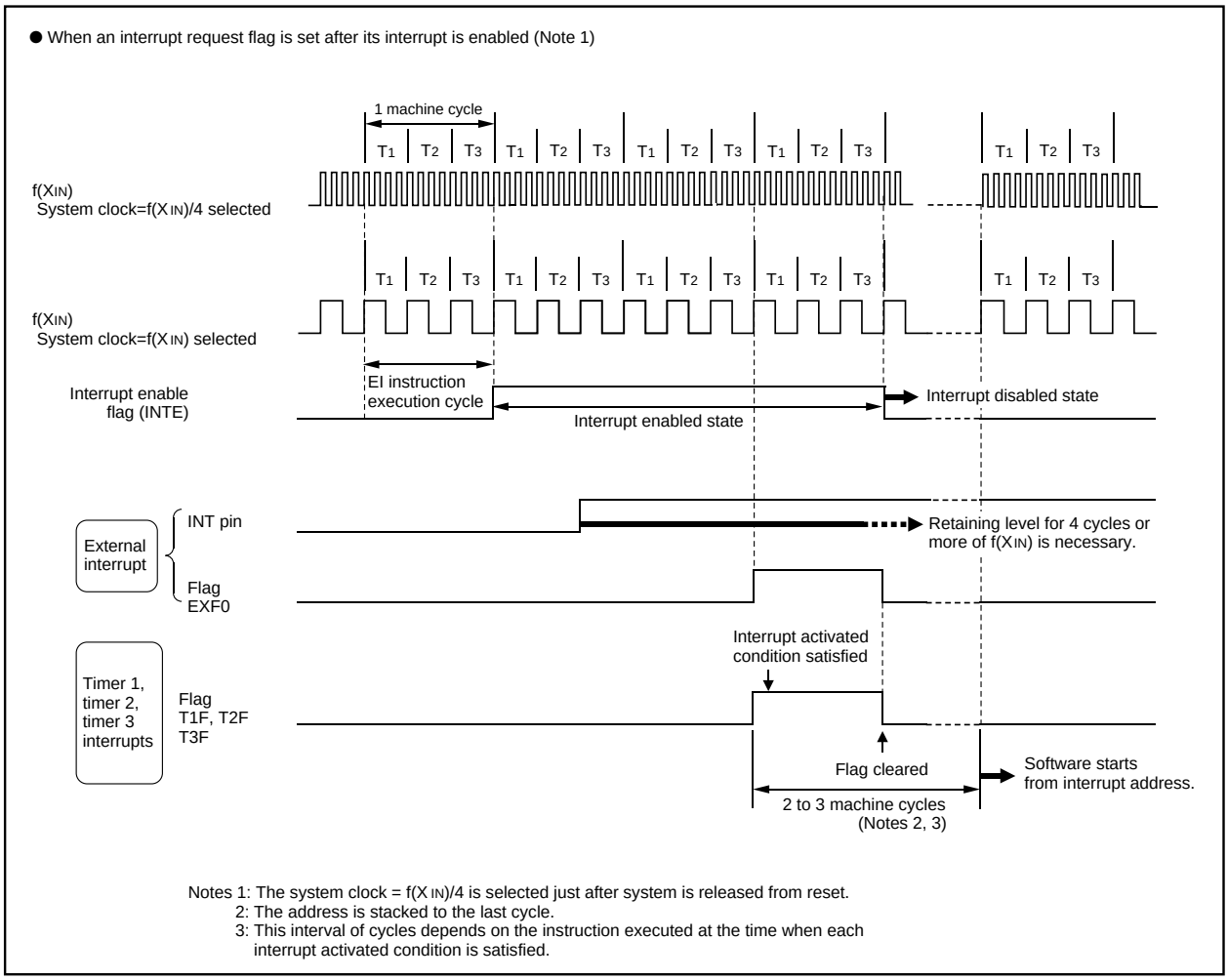
| Interrupt control register V1 | | at reset : $0000_2$ | RAM back-up : $0000_2$ | R/W |
|---|---|---|---|---|
| $V1_3$ | Timer 2 interrupt enable bit | 0 | Interrupt disabled (SNZT2 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZT2 instruction is invalid) | |
| $V1_2$ | Timer 1 interrupt enable bit | 0 | Interrupt disabled (SNZT1 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZT1 instruction is invalid) | |
| $V1_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V1_0$ | External 0 interrupt enable bit | 0 | Interrupt disabled (SNZ0 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZ0 instruction is invalid) | |

| Interrupt control register V2 | | at reset : $0000_2$ | at RAM back-up : $0000_2$ | R/W |
|---|---|---|---|---|
| $V2_3$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_2$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_0$ | Timer 3 interrupt enable bit | 0 | Interrupt disabled (SNZT3 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZT3 instruction is invalid) | |

Note: “R” represents read enabled, and “W” represents write enabled.

**(7) Interrupt sequence**

Interrupts occur only when the respective INTE flag, interrupt enable bits ($V1_0$–$V1_3$ and $V2_0$–$V2_3$), and interrupt request flags (EXF0, T1F, T2F, T3F) are "1." The interrupt actually occurs 2 to 3 machine cycles after the cycle in which all three conditions are satisfied. The interrupt occurs after 3 machine cycles only when the three interrupt conditions are satisfied on execution of instructions other than one-cycle instructions (Refer to Figure 16).




Notes 1: The system clock = f(XIN)/4 is selected just after system is released from reset.
2: The address is stacked to the last cycle.
3: This interval of cycles depends on the instruction executed at the time when each interrupt activated condition is satisfied.

**Fig. 16 Interrupt sequence**

## EXTERNAL INTERRUPTS

An external interrupt request occurs when a valid waveform (= waveform causing the external 0 interrupt) is input to an interrupt input pin (edge detection).
The external 0 interrupt can be controlled with the interrupt control register I1.

**Table 7 External interrupt activated condition**

| Name | Input pin | Valid waveform | Valid waveform selection bit ($I1_2$) |
|---|---|---|---|
| External 0 interrupt | $P2_1$/INT | Falling waveform (“H”→“L”) | 0 |
| | | Rising waveform (“L”→“H”) | 1 |

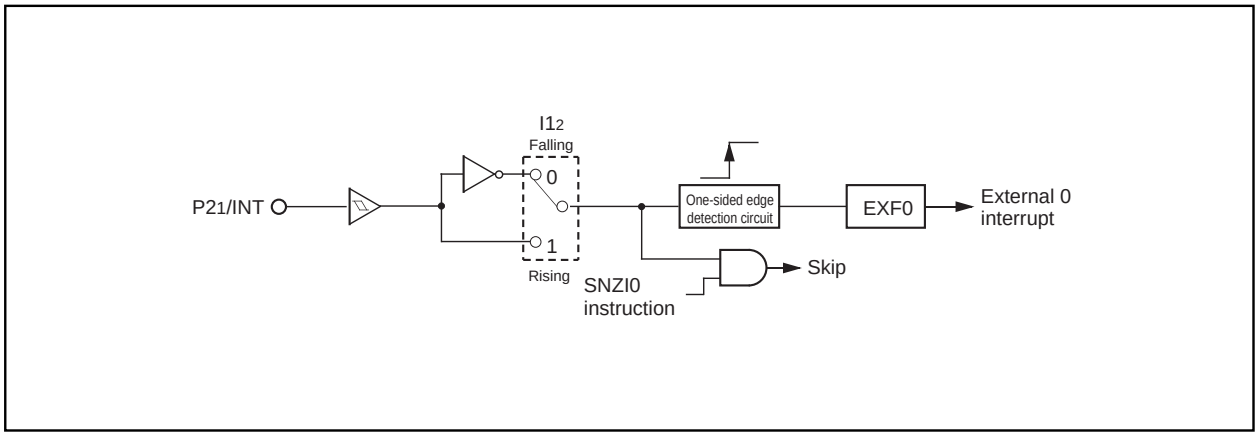



**Fig. 17 External interrupt circuit structure**

**(1) External 0 interrupt request flag (EXF0)**

External 0 interrupt request flag (EXF0) is set to "1" when a valid waveform is input to $P2_1$/INT pin.

The valid waveforms causing the interrupt must be retained at their level for 4 cycles or more of the system clock (Refer to Figure 16).

The state of EXF0 flag can be examined with the skip instruction (SNZ0). Use the interrupt control register V1 to select the interrupt or the skip instruction. The EXF0 flag is cleared to "0" when an interrupt occurs or when the next instruction is skipped with the skip instruction.

The $P2_1$/INT pin need not be selected the external interrupt input INT function or the normal input port $P2_1$ function. However, the EXF0 flag is set to "1" when a valid waveform is input to $P2_1$/INT pin even if it is used as an input port $P2_1$.

● External 0 interrupt activated condition

External 0 interrupt activated condition is satisfied when a valid waveform is input to $P2_1$/INT pin.

The valid waveform can be selected from rising waveform or falling waveform. An example of how to use the external 0 interrupt is as follows.

① Select the valid waveform with the bit 2 of register I1.
② Clear the EXF0 flag to "0" with the SNZ0 instruction.
③ Set the NOP instruction for the case when a skip is performed with the SNZ0 instruction.
④ Set both the external 0 interrupt enable bit ($V1_0$) and the INTE flag to "1."

The external 0 interrupt is now enabled. Now when a valid waveform is input to the $P2_1$/INT pin, the EXF0 flag is set to "1" and the external 0 interrupt occurs.

**(2) External interrupt control register**

● Interrupt control register I1

Register I1 controls the valid waveform for the external 0 interrupt, the return level (valid level of wakeup signal) from the RAM back-up and $P2_1$/INT pin function. Set the contents of this register through register A with the TI1A instruction. The TAI1 instruction can be used to transfer the contents of register I1 to register A.

**Table 8 External interrupt control register**

| Interrupt control register I1 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W |
|---|---|---|---|---|---|
| $I1_3$ | Not used | 0 | This bit has no function, but read/write is enabled. | | |
| | | 1 | | | |
| $I1_2$ | Interrupt valid waveform for INT pin/return level selection bit (Note 2) | 0 | Falling waveform ("L" level of INT pin is recognized with the SNZI0 instruction)/"L" level | | |
| | | 1 | Rising waveform ("H" level of INT pin is recognized with the SNZI0 instruction)/"H" level | | |
| $I1_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | | |
| | | 1 | | | |
| $I1_0$ | Not used | 0 | This bit has no function, but read/write is enabled. | | |
| | | 1 | | | |

Notes 1: "R" represents read enabled, and "W" represents write enabled.
2: Depending on the input state of $P2_1$/INT pin, the external interrupt request flag EXF0 may be set to "1" when the contents of $I1_2$ is changed. Accordingly, set a value to bit 2 of register I1 and execute the SNZ0 instruction to clear the EXF0 flag after executing at least one instruction.

## TIMERS

The 4570 Group has the programmable timers and a fixed dividing frequency timer.

- Programmable timer

  The programmable timer has a reload register and enables the frequency dividing ratio to be set. It is decremented from a set value n. When it underflows (count to n + 1), a timer interrupt request flag is set to "1," new data is loaded from the reload register, and count continues (auto-reload function).

- Fixed dividing frequency timer

  The fixed dividing frequency timer has the fixed frequency dividing ratio (n). An interrupt request flag is set to "1" every n count of a count pulse.

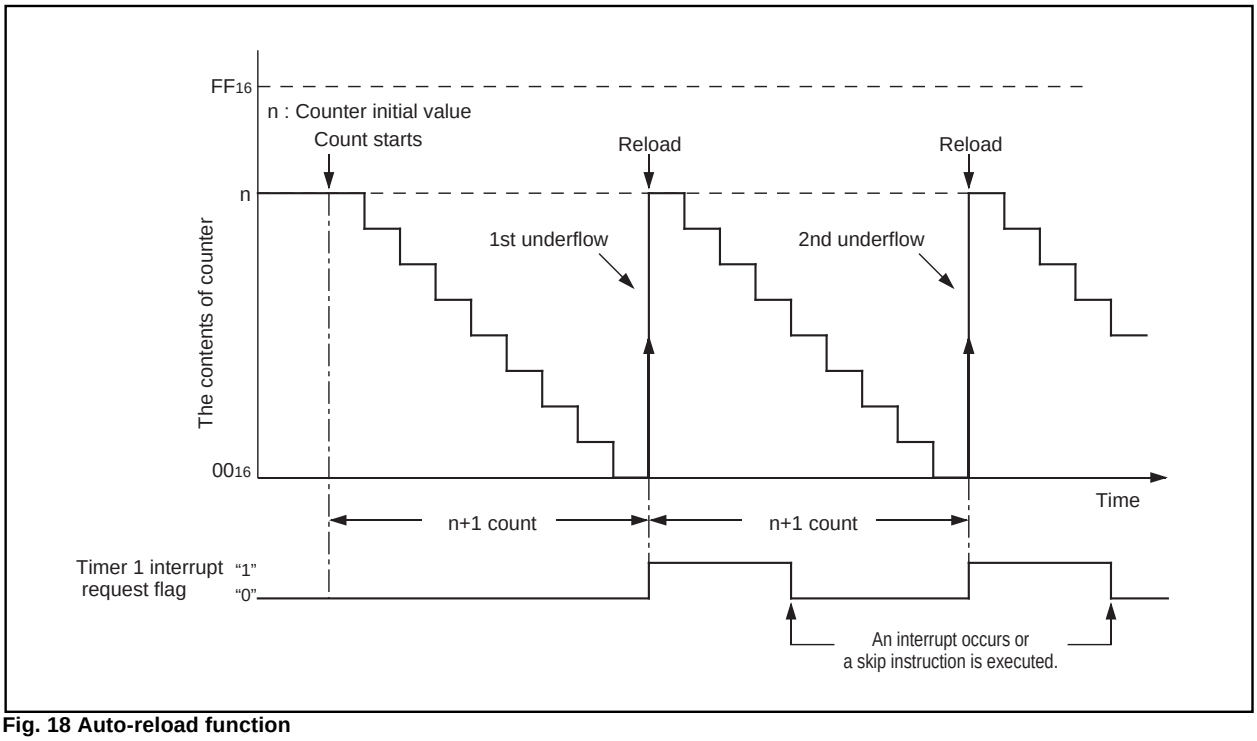



**Fig. 18 Auto-reload function**

The 4570 Group timer consists of the following circuits.
- Prescaler : frequency divider
- Timer 1 : 10-bit programmable timer with the interrupt function and the carrier wave output auto-control function
- Timer 2 : 8-bit programmable timer with the interrupt function
- Timer 3 : 8-bit programmable timer with the interrupt function and the carrier wave generation function
- 16-bit timer

Prescaler, timer 1, timer 2 and timer 3 can be controlled with the timer control registers W1, W2 and W3.
16-bit timer is the free-run counter without the control register.
Each function is described below.

**Table 9 Function related timers**

| Circuit | Structure | Count source | Frequency dividing ratio | Use of output signal | Control register |
|---|---|---|---|---|---|
| Prescaler | Frequency divider | • Instruction clock | 4, 8 | • Timer 1, 2 and 3 count sources | W1 |
| Timer 1 | 10-bit programmable binary down counter | • Prescaler output (ORCLK)<br>• Carrier wave generating circuit output (CARRY) | 1 to 1024 | • Timer 1 interrupt<br>• Carrier wave output auto-control<br>• Timer 2 count source | W1<br>(W5) |
| Timer 2 | 8-bit programmable binary down counter | • Prescaler output (ORCLK)<br>• Timer 1 underflow<br>• Instruction clock<br>• 16-bit timer underflow | 1 to 256 | • Timer 2 interrupt<br>• Timer 3 count source<br>• TOUT output | W2 |
| Timer 3 | 8-bit programmable binary down counter | • Prescaler output (ORCLK)<br>• Timer 2 underflow<br>• f(XIN) or f(XIN)/2 | 1 to 256 | • Timer 3 interrupt<br>• Timer 1 count source<br>• Carrier wave | W3 |
| 16-bit timer | 16-bit fixed dividing frequency | • Instruction clock | 65536 | • Watchdog timer (15-th bit output is counted twice.)<br>• Timer 2 count source (16-bit timer underflow) | |

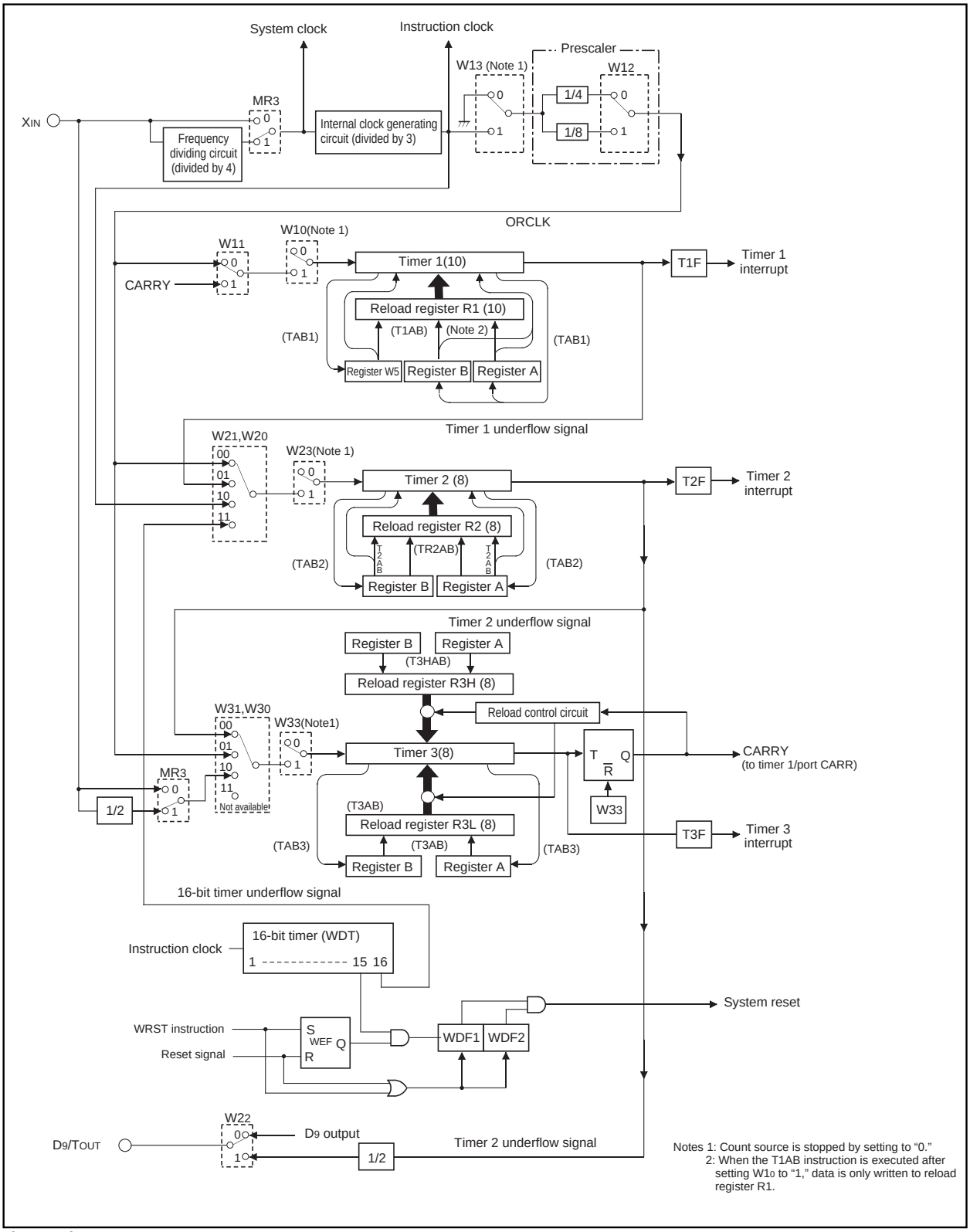



**Fig. 19 Timers structure**

**Table 10 Timer control registers**

| Timer control register W1 | | at reset : $0000_2$ | | at RAM back-up : $0000_2$ | R/W |
|---|---|---|---|---|---|
| $W1_3$ | Prescaler control bit | 0 | Stop (prescaler state initialized) | | |
| | | 1 | Operating | | |
| $W1_2$ | Prescaler dividing ratio selection bit | 0 | Instruction clock divided by 4 | | |
| | | 1 | Instruction clock divided by 8 | | |
| $W1_1$ | Timer 1 count source selection bit | 0 | Prescaler output (ORCLK) | | |
| | | 1 | Carrier output (CARRY) | | |
| $W1_0$ | Timer 1 control bit | 0 | Stop (state retained) | | |
| | | 1 | Operating | | |

| Timer control register W2 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W |
|---|---|---|---|---|---|
| $W2_3$ | Timer 2 control bit | 0 | Stop (state retained) | | |
| | | 1 | Operating | | |
| $W2_2$ | Port $D_9$/$T_{OUT}$ pin function selection bit | 0 | Port $D_9$ | | |
| | | 1 | $T_{OUT}$ pin | | |
| $W2_1$ | Timer 2 count source selection bits | $W2_1$ | $W2_0$ | Count source | |
| | | 0 | 0 | Prescaler output (ORCLK) | |
| $W2_0$ | | 0 | 1 | Timer 1 underflow signal | |
| | | 1 | 0 | Instruction clock | |
| | | 1 | 1 | 16-bit timer underflow signal | |

| Timer control register W3 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W |
|---|---|---|---|---|---|
| $W3_3$ | Timer 3 control bit | 0 | Stop (state retained) | | |
| | | 1 | Operating | | |
| $W3_2$ | Not used | 0 | This bit has no function, but read/write is enabled. | | |
| | | 1 | | | |
| $W3_1$ | Timer 3 count source selection bits | $W3_1$ | $W3_0$ | Count source | |
| | | 0 | 0 | Timer 2 underflow signal | |
| $W3_0$ | | 0 | 1 | Prescaler output (ORCLK) | |
| | | 1 | 0 | f($X_{IN}$) or f($X_{IN}$)/2 | |
| | | 1 | 1 | Not available | |

| Timer count value store register W5 | at reset : $00_2$ | at RAM back-up : state retained | R/W |
|---|---|---|---|
| 2-bit register. The contents of the high-order 2 bits (bits 9 and 8) of the 10-bit ROM pattern at address ($D_2D_1D_0A_3A_2A_1A_0$) in page p specified by registers D and A is stored in this register W5 with the TABP p instruction.<br>In addition, data can be transferred between the low-order 2 bits of register A and this register W5 with the TW5A or TAW5 instruction. Data can be read/written to/from the high-order 2 bits of timer 1 with the T1AB or TAB1 instruction. | | | |

Note: “R” represents read enabled, and “W” represents write enabled.

**(1) Timer control registers**

- Timer control register W1
  Register W1 controls the count source and count operation of timer 1, the frequency dividing ratio and count operation of prescaler. Set the contents of this register through register A with the TW1A instruction. The TAW1 instruction can be used to transfer the contents of register W1 to register A.
- Timer control register W2
  Register W2 controls the count operation and count source of timer 2 and $D_9$/$T_{OUT}$ pin function. Set the contents of this register through register A with the TW2A instruction. The TAW2 instruction can be used to transfer the contents of register W2 to register A.
- Timer control register W3
  Register W3 controls the count operation and count source of timer 3. Set the contents of this register through register A with the TW3A instruction. The TAW3 instruction can be used to transfer the contents of register W3 to register A.
- Timer count value store register W5
  2-bit register. The contents of the high-order 2 bits (bits 9 and 8) of the 10-bit ROM pattern at address in page p specified by registers D and A is stored in this register W5 with the TABP p instruction.
  In addition, data can be transferred between the low-order 2 bits of register A and this register W5 with the TW5A or TAW5 instruction. Data can be read/written to/from the high-order 2 bits of timer 1 with the T1AB or TAB1 instruction.

**(2) Precautions**

Note the following for the use of timers.

- Prescaler
  Stop the prescaler operation to change its frequency dividing ratio.
- Count source
  Stop timer 1, 2 or 3 counting to change its count source.
- Reading the timer count value
  Stop each of the timers and then execute the TAB1, TAB2 or TAB3 instruction to read timer 1, 2 or 3 data.
- Writing to reload register R1
  When writing data to reload register R1 while timer 1 is operating, avoid a timing when timer 1 underflows.
- Writing to reload register R3H
  When writing data to reload register R3H while timer 3 is operating, avoid a timing when timer 3 underflows.

**(3) Prescaler**

Prescaler is a frequency divider. Its frequency dividing ratio can be selected. The count source of prescaler is the instruction clock.

Use the bit 2 of register W1 to select the prescaler dividing ratio and the bit 3 to start and stop its operation. When the bit 3 of register W1 is cleared to “0,” prescaler is initialized, and the output signal (ORCLK) stops.

**(4) Timer 1 (interrupt function)**

Timer 1 is a 10-bit binary down counter with the timer 1 reload register (R1). The 10-bit data can be set in timer 1 through registers A, B and W5. Set bits 0 to 3 to register A, bits 4 to 7 to regiser B and bits 8 to 9 to register W5 to set data to timer 1. Also, ROM pattern (bits 0 to 9) can be set to registers A, B and W5 with the TABP p instruction. Execute the T1AB instruction to set data in timer 1.

When timer 1 stops, 10-bit data can be set simultaneously in timer 1 and the reload register (R1) with the T1AB instruction.

When timer 1 is operating, data can be set only in the reload register (R1) with the T1AB instruction.

When setting the next count data to reload register R1 while timer 1 is operating, be sure to set data before timer 1 underflows.

Timer 1 starts counting after the following process;

① set data in timer 1,
② select the count source with bit 1 of register W1,
③ set the bit 0 of register W1 to “1.”

Once count is started, when timer 1 underflows (the next count pulse is input after the contents of timer 1 becomes “0”), the timer 1 interrupt request flag (T1F) is set to “1,” new data is loaded from reload register R1, and count continues (auto-reload function).

When a value set in reload register R1 is n, timer 1 divides the count source signal by n + 1 (n = 0 to 1023).

Data can be read from timer 1 to registers A, B and W5. Stop counting and then execute the TAB1 instruction to read its data.

**(5) Timer 2 (interrupt function)**

Timer 2 is an 8-bit binary counter with the timer 2 reload register (R2). Data can be set simultaneously in timer 2 and the reload register (R2) with the TAB2 instrucion. Also, data can be set only in the reload register (R2) with the TR2AB instruction.

Timer 2 starts counting after following process;

① set data in timer 2,
② select the count source with bits 0 and 1 of register W2,
③ set the bit 3 of register W2 to “1.”

Once count is started, when timer 2 underflows (the next count pulse is input after the contents of timer 2 becomes “0”), the timer 2 interrupt request flag (T2F) is set to “1,” new data is loaded from reload register R2, and count continues (auto-reload function).

When a value set in reload register R2 is n, timer 2 divides the count source signal by n+1 (n = 0 to 255).

Data can be read from timer 2 to registers A and B with the TAB2 instruction. Stop counting and then execute the TAB2 instruction to read its data.

**(6) Timer 3**

Timer 3 is an 8-bit binary down counter with the timer 3 reload registers (R3H, R3L). Data can be set simultaneously in timer 3 and the reload register (R3L) with the T3AB instruction.

Data can be set in reload register R3H with the T3HAB instruction.

Timer 3 starts counting after the following process;

① set data in timer 3,

② select the count source with the bits 1 and 0 of register W3,

③ set the bit 3 of register W3 to "1."

The f($X_{IN}$) or f($X_{IN}$)/2 is selected as the count source by setting $W3_1$ to "1" and $W3_0$ to "0."

When the f($X_{IN}$) is selected as the system clock (bit 3 of clock control register MR= "0"), f($X_{IN}$) is selected as the count source.

When the f($X_{IN}$)/4 is selected as the system clock (bit 3 of clock control register MR= "1"), f($X_{IN}$)/2 is selected as the count source.

Once count is started, when timer 3 underflows (the next count pulse is input after the contents of timer 3 become "0"), the timer 3 interrupt request flag (T3F) is set to "1," new data is loaded from reload register R3H, and count coutinues (auto-reload function).

When the timer 3 underflows again after auto-reload is performed, the timer 3 interrupt request flag (T3F) is set to "1" and new data is reloaded from the reload register R3L and count continues. Timer 3 reloads data from reload register R3H or R3L alternately every underflow.

When the T3AB instruction is executed while timer 3 is operating, new data is set in timer 3 and reload register R3L, count is started again at the next machine cycle. At the next underflow, data is reloaded from R3H and count continues regardless that auto-reload is performed from reload register R3H or R3L at the previous underflow.

Data can be read from timer 3 through registers A and B. Stop counting and then execute the TAB3 instruction to read its data. Timer 3 can be also used as the carrier wave generating circuit.

**(7) Timer output pin ($D_9$/$T_{OUT}$)**

Timer output pin ($D_9$/$T_{OUT}$) is used to output the timer 2 underflow signal.

The $D_9$/$T_{OUT}$ pin function can be selected by the bit 2 of register W2.

**(8) Timer interrupt request flags (T1F, T2F, T3F)**

Each timer interrupt request flag is set to "1" when each timer underflows. The state of these flags can be examined with the skip instructions (SNZT1, SNZT2, SNZT3).

Use the interrupt control registers V1 and V2 to select an interrupt or a skip instruction.

An interrupt request flag is cleared to "0" when an interrupt occurs or when the next instruction is skipped with a skip instruction.

## WATCHDOG TIMER

Watchdog timer provides a method to reset the system when a program runs wild. Watchdog timer consists of 16-bit timer (WDT), watchdog timer enable flag (WEF), and watchdog timer flags (WDF1, WDF2).

Timer WDT starts downcounting the instruction clocks as the count source immediately after system is released from reset. The underflow signal is generated when the count value reaches "$0000_{16}$." This underflow signal can be used as the timer 2 count source.

When the WRST instruction is executed after system is released from reset, the WEF flag is set to "1." At this time, the watchdog timer starts operating.

When the count value of timer WDT reaches "$BFFF_{16}$" or "$3FFF_{16}$," WDF1 flag is set to "1." Then, if the WRST instruction is not executed while the timer WDT counts 32767, the WDF2 flag is set to "1" and the $\overline{\text{RESET}}$ pin outputs "L" level to reset the microcomputer. In software using the watchdog timer, make sure that the WRST instruction is executed in 32766 machine cycles or less in order to keep the microcomputer operating normally.

To prevent the watchdog timer from stopping in the event of misoperation, the WEF flag is designed not to be initialized once the WRST instruction has been executed. Note also that, if the WRST instruction is never executed, the watchdog timer does not start.

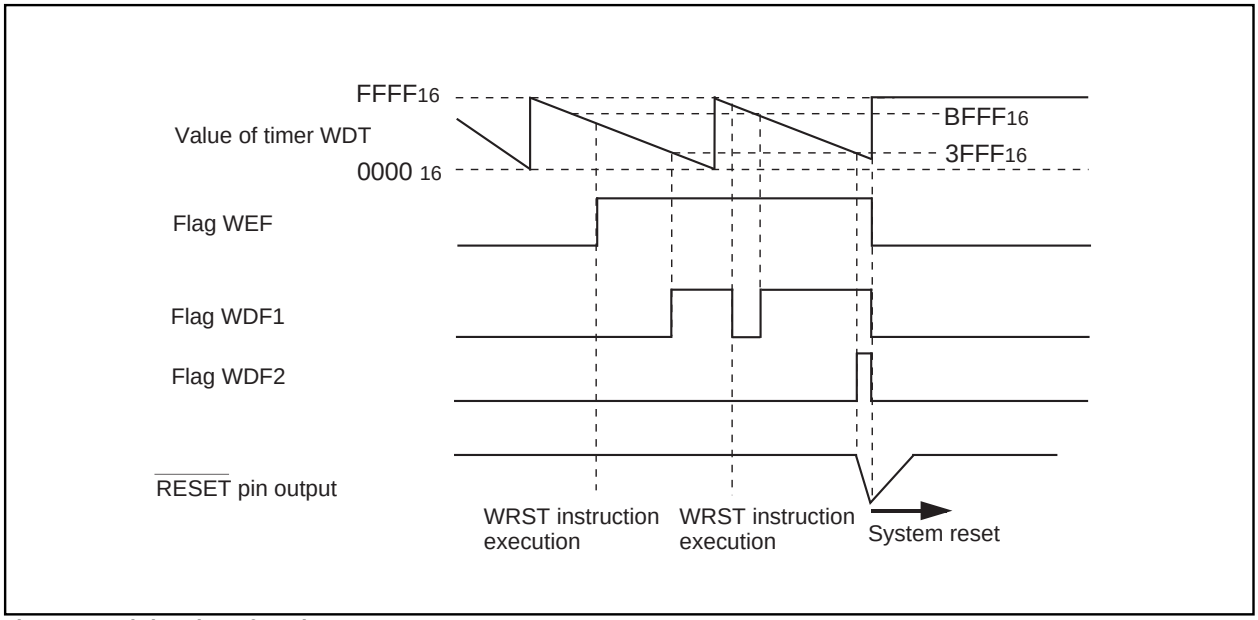



**Fig. 20 Watchdog timer function**

The contents of the WEF flag, the WDF1 and WDF2 flags and the timer WDT are initialized at the RAM back-up mode.

However, if the WDF2 flag is set to "1" at the same time that the microcomputer enters the RAM back-up mode, system reset may be performed.

When using the watchdog timer and the RAM back-up mode, initialize the WDF1 flag with the WRST instruction just before the microcomputer enters the RAM back-up mode (refer to Figure 21).

```
    :
WRST        ; Clear WDF1 flag
EPOF        ; POF instruction execution enabled
POF
↓
Oscillation stop (RAM back-up mode)
```

**Fig. 21 Program example to enter the RAM back-up mode when using the watchdog timer**

## CARRIER WAVE GENERATING CIRCUIT

The 4570 Group has a carrier wave generating circuit that generates the transfer waveform for various remote control carrier wave.

The carrier wave generating circuit outputs the signal inverted every timer 3 underflow (CARRY) from port CARR.

When using the carrier wave generating circuit, select the f($X_{IN}$) or f($X_{IN}$)/2 for the timer 3 count source ($W3_1$="1", $W3_0$="0").

When the bit 3 of the clock control register MR is "0" (system clock=f($X_{IN}$)), f($X_{IN}$) is selected as the count source.

When the bit 3 of the clock control register MR is "1" (system clock=f($X_{IN}$)/4), f($X_{IN}$)/2 is selected as the count source.

Set the count value corresponding to "L" interval of carrier wave output to timer 3 reload register R3L.

Set the count value corresponding to "H" interval of carrier wave output to timer 3 reload register R3H.

Also, timer 1 can auto-control the carrier wave output of port CARR by setting the carrier wave output control register (C2).

When timer 3 is stopped, the output level of port CARR is initialized. ("L" level)

**(1) Carrier wave output control register (C2)**

Timer 1 can auto-control the output enable interval and the output disable interval of the carrier wave output from port CARR by setting the bit 0 of register C2 to "1." Set the contents of this register through register A with the TC2A instruction.

The setting of the output enable/disable interval is described below.

① Validate the carrier wave output auto-control function ($C2_0$="1").
② Set the count value ("L" interval of carrier wave output) to timer 3 and reload register R3L.
③ Set the count value ("H" interval of carrier wave output) to timer 3 reload register R3H.
④ Set the count value (the output enable interval of carrier wave from port CARR) to timer 1.
⑤ Select the carrier wave ($W1_1$ = "1") as the timer 1 count source.
⑥ Operate timer 1 ($W1_0$="1").
⑦ Operate timer 3 ($W3_3$="1").
⑧ Set the next count value (the output disable interval of carrier wave from port CARR) to reload register R1 before timer 1 underflow occurs.

The carrier wave is output from port CARR until the first timer 3 underflow occurs. The output of the carrier wave from port CARR is disabled and the next count value is loaded from reload register R1 to timer 1 by the first timer 1 underflow.

Then, the output of carrier wave is disabled until the second timer 1 underflow occurs. Also, the next enable interval of the carrier wave output can be set by setting the third count value to timer 1 reload register R1 before the second timer 1 underflow occurs.

If the carrier wave output auto-control function is invalidated ($C2_0$="0") while the carrier wave output is auto-controlled, the output of port CARR retains the state when the auto-control is invalidated regardless of timer 1 underflow. This state can be terminated by timer 1 stop ($W1_0$="0").

When the carrier wave output auto-control function is validated ($C2_0$="1") again after it is invalidated ($C2_0$="0"), the auto-control of carrier wave output is started again when the next timer 1 underflow occurs.

Stop the timer 3 and invalidate the auto-control function by timer 1 to use the port CARR output contorl bit ($C2_1$).

**(2) Notes when using the carrier wave output auto-control function**

- Set the timer 1 and register C2 before timer 3 is started to operate ($W3_3$="1").
- Stop the timer 1 ($W1_0$="0") after stopping the timer 3 ($W3_3$="0") while the carrier wave output is disabled in order to stop the carrier wave output auto-control operation.
- If the carrier wave output auto-control function is invalidated ($C2_0$="0") while the carrier wave output is auto-controlled, the output of port CARR retains the state when the auto-control is invalidated regardless of timer 1 underflow.
  When the carrier wave output auto-control function is validated ($C2_0$="1") again after it is invalidated ($C2_0$="0"), the auto-control by timer 1 is validated again when the next timer 1 underflow occurs.
  However, when the carrier wave output auto-control bit ($C2_0$) is changed during timer 1 underflow, the error-operation may occur.
- When the carrier wave output auto-control function is selected, use the carrier wave CARRY as the timer 1 count source.
  If the ORCLK is used as the count source, a short pulse may occur in port CARR output because ORCLK is not synchronized with the carrier wave.
- When the carrier wave output auto-control function is selected and data is set to reload register R1 while timer 1 is operating, avoid the timing that the contents of timer 1 becomes "0" to execute the T1AB instruction.

**Table 11 Carrier wave output control register**

| Carrier wave output control register C2 | | at reset : $00_2$ | | at RAM back-up : $00_2$ | W |
|---|---|---|---|---|---|
| $C2_1$ | Port CARR output control bit | 0 | Port CARR "L" level output | | |
| | | 1 | Port CARR "H" level output | | |
| $C2_0$ | Carrier wave output auto-control bit | 0 | Auto-control output by timer 1 is invalid | | |
| | | 1 | Auto-control output by timer 1 is valid | | |

Note: "W" represents write enabled.

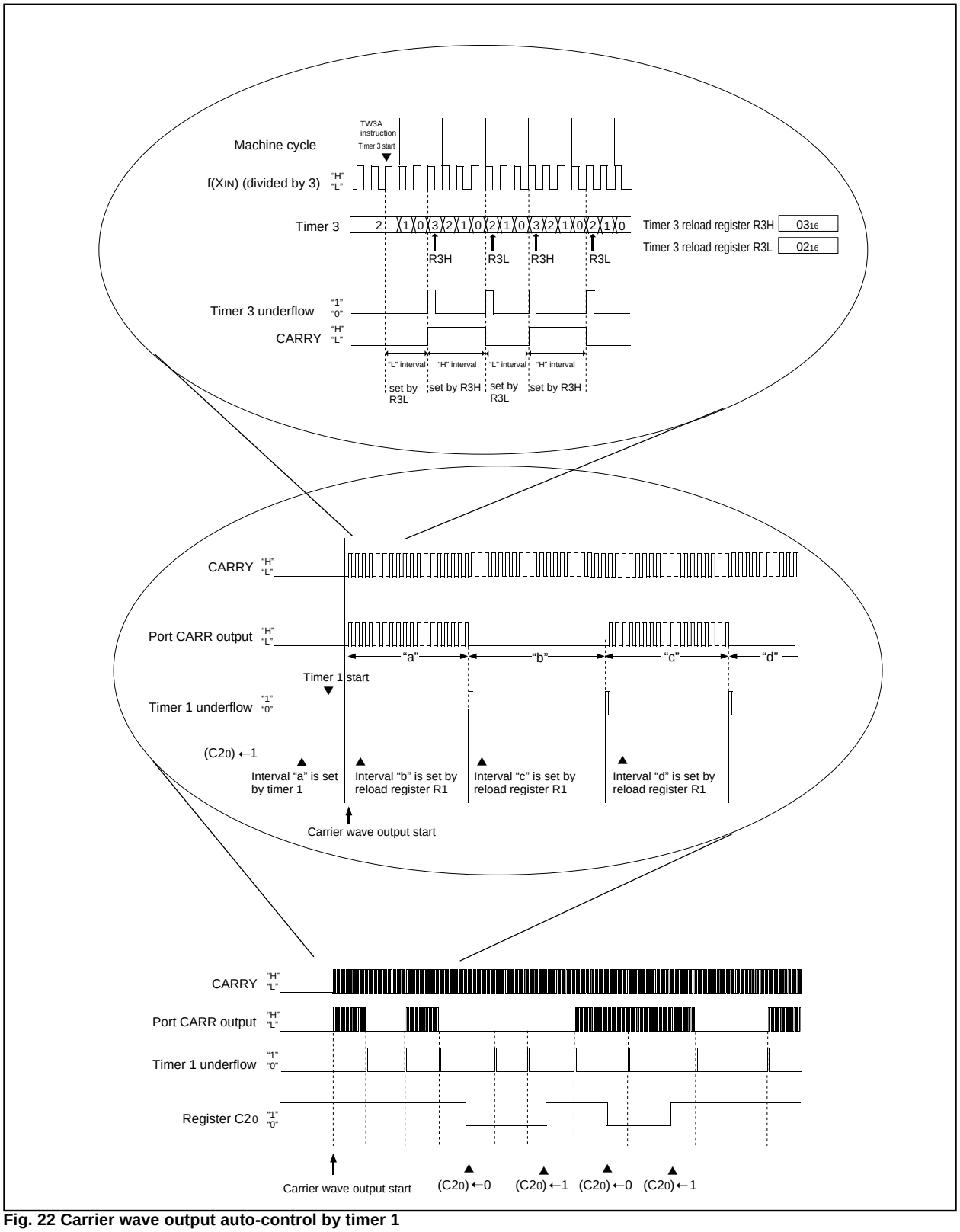



**Fig. 22 Carrier wave output auto-control by timer 1**

## RESET FUNCTION

System reset is performed by applying "L" level to $\overline{\text{RESET}}$ pin for 1 machine cycle or more when the following condition is satisfied;

- the value of supply voltage is the minimum value or more of the recommended operating conditions.

Then when "H" level is applied to $\overline{\text{RESET}}$ pin, software starts from address 0 in page 0.

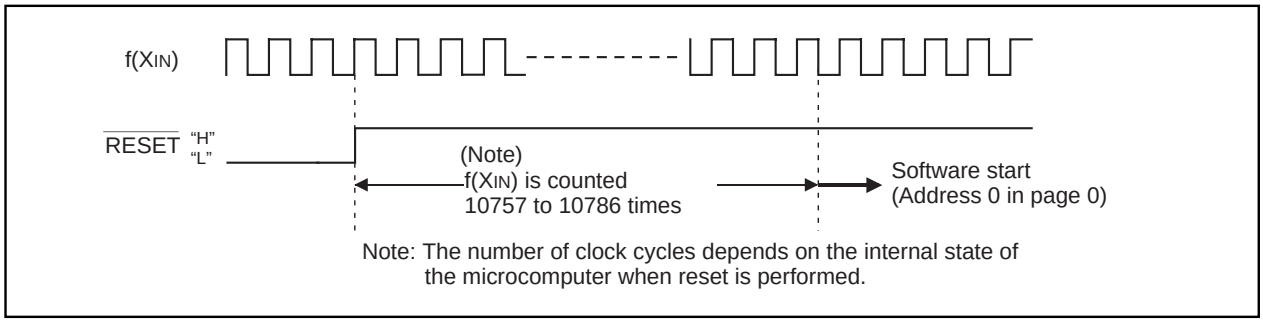



**Fig. 23 Reset release timing**

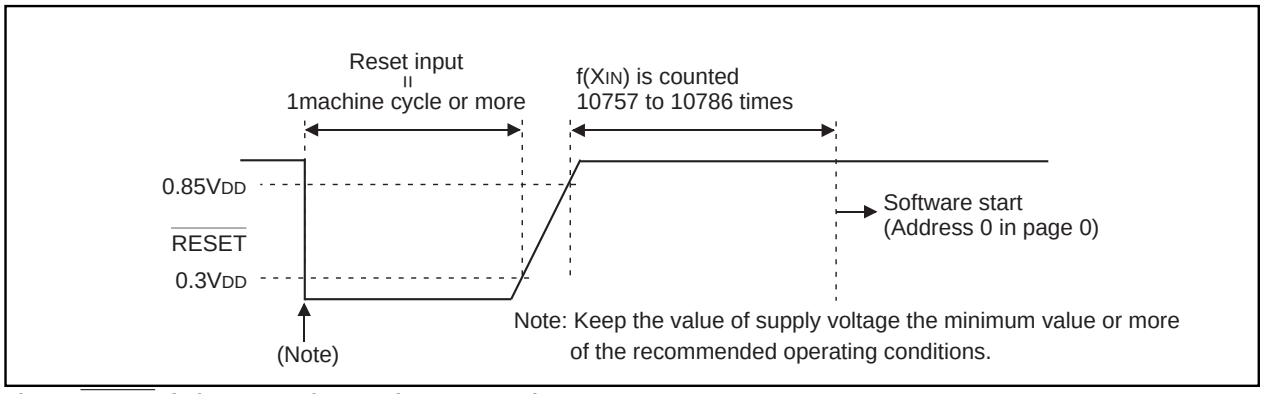



**Fig. 24 $\overline{\text{RESET}}$ pin input waveform and reset operation**

**(1) Power-on reset**

Reset can be automatically performed at power on (power-on reset) by the built-in power-on reset circuit. When the built-in power-on reset circuit is used, the time for the supply voltage to reach the minimum operating voltage must be set to 100 $\mu$s or less. If the rising time exceeds 100 $\mu$s, connect a capacitor between the $\overline{\text{RESET}}$ pin and VSS at the shortest distance, and input "L" level to $\overline{\text{RESET}}$ pin until the value of supply voltage reaches the minimum operating voltage.

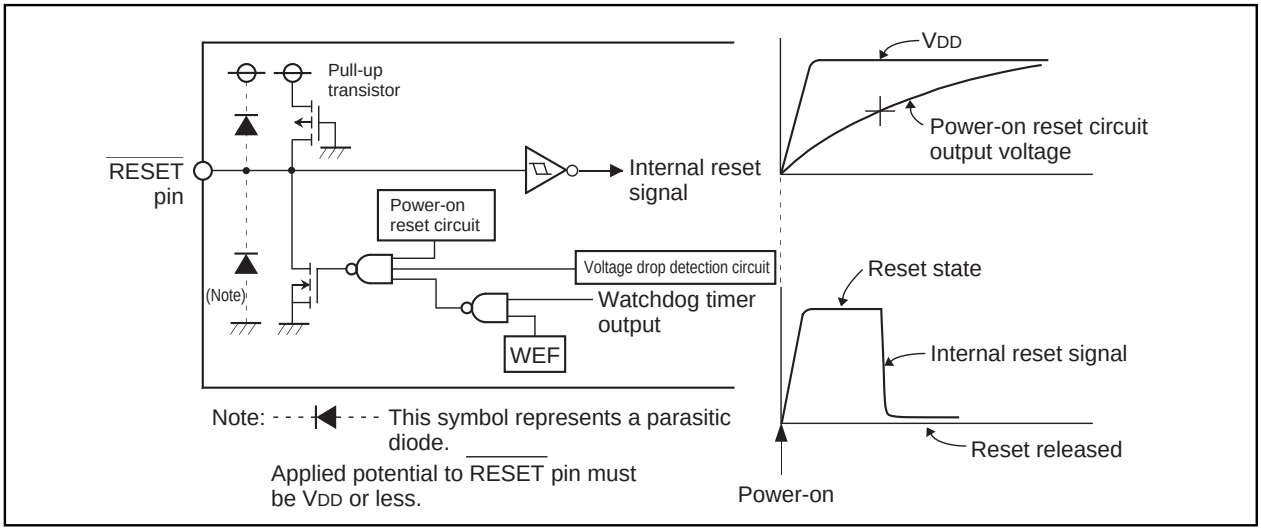



**Fig. 25 Power-on reset circuit example**

**(2) Internal state at reset**

Table 12 shows port state at reset, and Figure 26 shows internal state at reset (they are retained after system is released from reset).

The contents of timers, registers, flags and RAM except those shown in Figure 26 are undefined, so set the initial values to them.

**Table 12 Port state at reset**

| Name | Function | State |
|---|---|---|
| $D_0$–$D_8$, $D_9$/$T_{OUT}$ | $D_0$–$D_8$, $D_9$ | High impedance (Note 1) |
| $P0_0$–$P0_3$ | $P0_0$–$P0_3$ | "H" ($V_{DD}$) level (Note 1) |
| $P1_0$–$P1_3$ | $P1_0$–$P1_3$ | |
| $P2_0$, $P2_1$/INT | $P2_0$, $P2_1$ | High impedance |
| $P3_0$–$P3_3$ | $P3_0$–$P3_3$ | High impedance (Note 1) |
| $P4_0$–$P4_3$ | $P4_0$–$P4_3$ | High impedance (Note 2) |
| CARR | CARR | "L" ($V_{SS}$) level |

Notes 1: Output latch is set to "1."
2: The pull-up transistor is turned off.

- Program counter (PC) ..... 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  Address 0 in page 0 is set to program counter.
- Interrupt enable flag (INTE) ..... 0 (Interrupt disabled)
- Power down flag (P) ..... 0
- External 0 interrupt request flag (EXF0) ..... 0
- Interrupt control register V1 ..... 0 0 0 0 (Interrupt disabled)
- Interrupt control register V2 ..... 0 0 0 0 (Interrupt disabled)
- Interrupt control register I1 ..... 0 0 0 0
- Timer 1 interrupt request flag (T1F) ..... 0
- Timer 2 interrupt request flag (T2F) ..... 0
- Timer 3 interrupt request flag (T3F) ..... 0
- Watchdog timer flags (WDF1, WDF2) ..... 0
- Watchdog timer enable flag (WEF) ..... 0
- Timer control register W1 ..... 0 0 0 0 (Prescaler and timer 1 stopped)
- Timer control register W2 ..... 0 0 0 0 (Timer 2 stopped)
- Timer control register W3 ..... 0 0 0 0 (Timer 3 stopped)
- Timer count value store register W5 ..... 0 0
- Clock control register MR ..... 1 0 0 0
- 8-bit general-purpose register SI ..... 0 0 0 0 0 0 0 0
- Carrier wave output control register C2 ..... 0 0
- Key-on wakeup control register K0 ..... 0 0 0 0
- Pull-up control register PU0 ..... 0 0 0 0
- Carry flag (CY) ..... 0
- Register A ..... 0 0 0 0
- Register B ..... 0 0 0 0
- Register D ..... X X X
- Register E ..... X X X X X X X X
- Register X ..... 0 0 0 0
- Register Y ..... 0 0 0 0
- Register Z ..... X X
- Stack pointer (SP) ..... 1 1 1

"X" represents undefined.

**Fig. 26 Internal state at reset**

## VOLTAGE DROP DETECTION CIRCUIT

The built-in voltage drop detection circuit is designed to detect a drop in voltage and to reset the microcomputer if the supply voltage drops below a set value.

The voltage drop detection circuit is not operated at the RAM back-up mode.

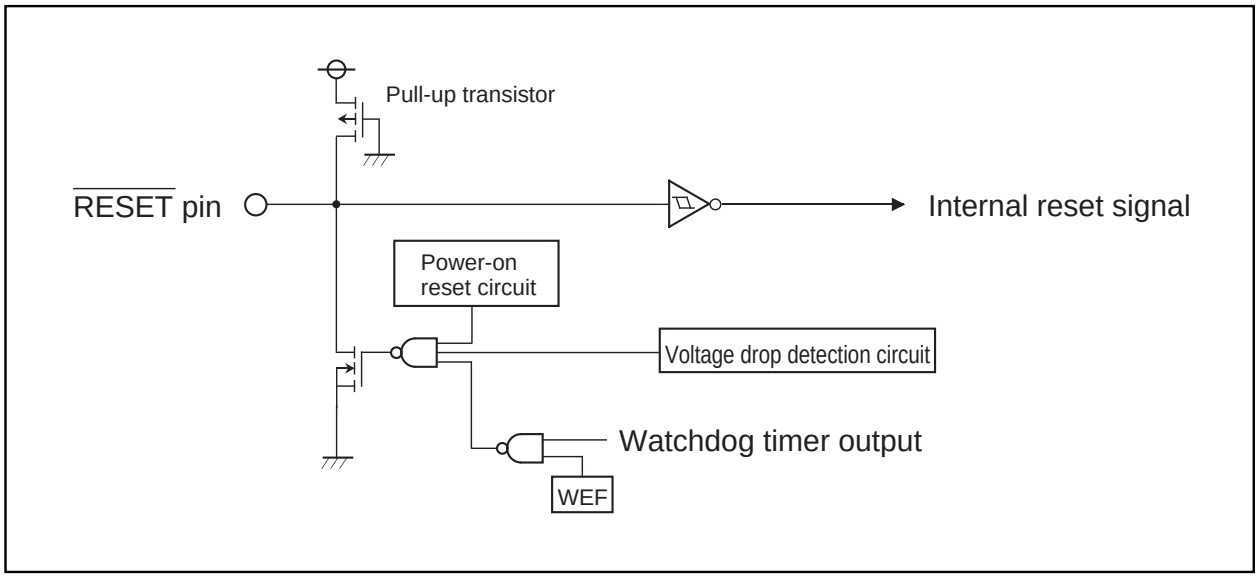



**Fig. 27 Voltage drop detection reset circuit**

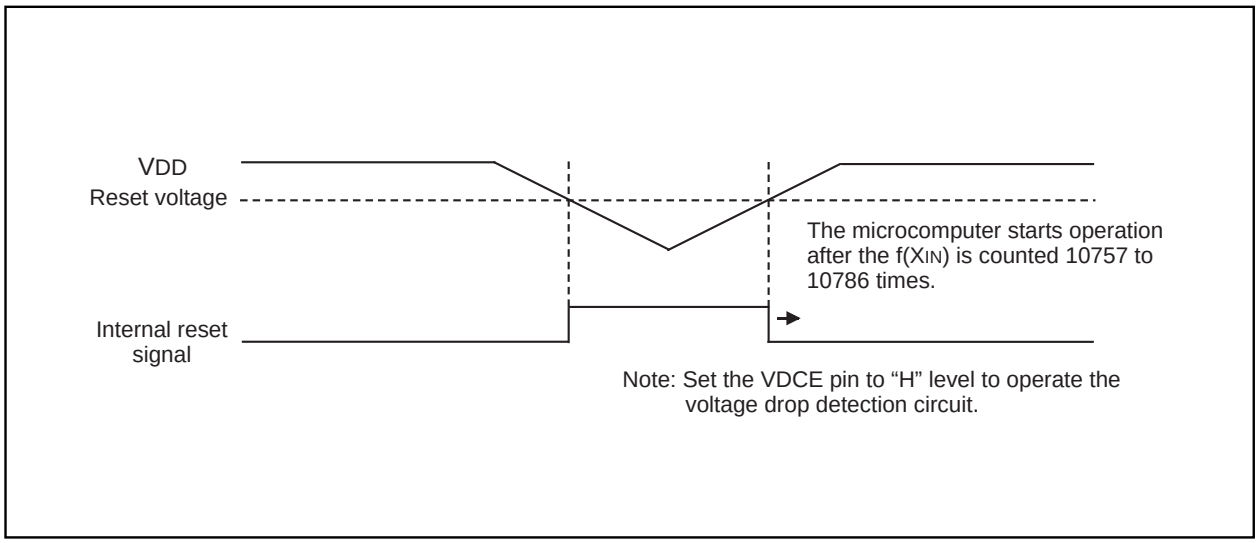



**Fig. 28 Voltage drop detection circuit operation waveform**

## RAM BACK-UP MODE

The 4570 Group has the RAM back-up mode.
When the EPOF and POF instructions are executed continuously, system enters the RAM back-up state.
The POF instruction is equivalent to the NOP instruction when the EPOF instruction is not executed before the POF instruction.
As oscillation is stopped retaining RAM, the function of reset circuit and states at RAM back-up mode, power dissipation can be reduced without losing the contents of RAM.
Table 13 shows the function and states retained at RAM back-up. Figure 29 shows the state transition.

**(1) Identification of the start condition**

Warm start (return from the RAM back-up mode) or cold start (return from the normal reset state) can be identified by examining the state of the power down flag (P) with the SNZP instruction.

**(2) Warm start condition**

When the external wakeup signal is input after the system enters the RAM back-up mode by executing the EPOF and POF instructions continuously, the CPU starts executing the software from address 0 in page 0. In this case, the P flag is "1."

**(3) Cold start condition**

The CPU starts executing the software from address 0 in page 0 when;

- reset pulse is input to $\overline{\text{RESET}}$ pin, or
- reset by watchdog timer is performed, or
- voltage drop detection circuit detects the voltage drop.

In this case, the P flag is "0."

**Table 13 Functions and states retained at RAM back-up**

| Function | RAM back-up |
|---|---|
| Program counter (PC), registers A, B, carry flag (CY), stack pointer (SP) (Note 2) | × |
| Contents of RAM | O |
| Port level | O |
| Clock control register MR | O |
| Timer control register W1 | × |
| Timer control registers W2, W3 | O |
| Timer count value store register W5 | O |
| Interrupt control registers V1, V2 | × |
| Interrupt control register I1 | O |
| Carrier wave output control register C2 | × |
| 8-bit general-purpose register SI | O |
| Timer 1 function | × |
| Timer 2 function | (Note 3) |
| Timer 3 function | (Note 3) |
| Pull-up control register PU0 | O |
| Key-on wakeup control register K0 | O |
| External 0 interrupt request flag (EXF0) | × |
| Timer 1 interrupt request flag (T1F) | × |
| Timer 2 interrupt request flag (T2F) | (Note 3) |
| Timer 3 interrupt request flag (T3F) | (Note 3) |
| Watchdog timer flag 1 (WDF1) | × (Note 4) |
| Watchdog timer flag 2 (WDF2) | × (Note 4) |
| Watchdog timer enable flag (WEF) | × (Note 4) |
| 16-bit timer (WDT) | × (Note 4) |
| Interrupt enable flag (INTE) | × |

Notes 1: "O" represents that the function can be retained, and "×" represents that the function is initialized.
Registers and flags other than the above are undefined at RAM back-up, and set an initial value after returning.
2: The stack pointer (SP) points the level of the stack register and is initialized to "$111_2$" at RAM back-up.
3: The state of the timer is undefined.
4: Initialize the watchdog timer with the WRST instruction, and then execute the EPOF and POF instructions.

**(4) Return signal**

An external wakeup signal is used to return from the RAM back-up mode because the oscillation is stopped. Table 14 shows the return condition for each return source.

**(5) Port P4 control registers**

- Key-on wakeup control register K0
  Register K0 controls the port P4 key-on wakeup function. Set the contents of this register through register A with the TK0A instruction. In addition, the TAK0 instruction can be used to transfer the contents of register K0 to register A.
- Pull-up control register PU0
  Register PU0 controls the ON/OFF of the port P4 pull-up transistor. Set the contents of this register through register A with the TPU0A instruction. In addition, the TAPU0 instruction can be used to transfer the contents of register PU0 to register A.

**Table 14 Return source and return condition**

| Return source | | Return condition | Remarks |
|---|---|---|---|
| External wakeup signal | Ports P0, P1 and P4 | Return by an external falling edge input (“H”→“L”). | Port P0 shares the falling edge detection circuit with ports P1 and P4. Key-on wakeup functions of ports P0 and P1 are always valid. The key-on wakeup function valid/invalid of port P4 can be controlled with register K0. Set the port using the key-on wakeup function selected to “H” level before going into the RAM back-up mode. |
| | $P2_1$/INT pin | Return by an external “H” level or “L” level input.<br>The EXF0 flag is not set. | Select the return level (“L” level or “H” level) with the bit 2 of register I1 according to the external state before going into the RAM back-up mode. |

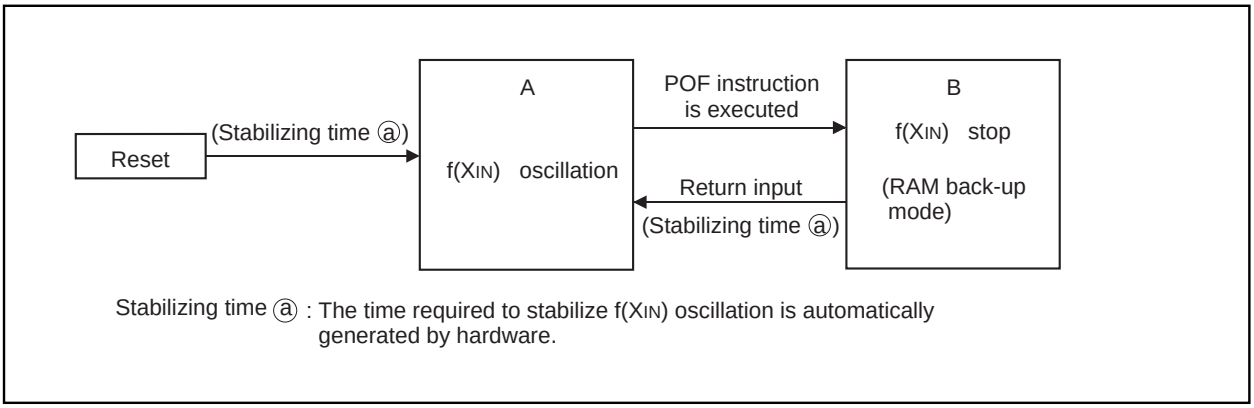



**Fig. 29 State transition**

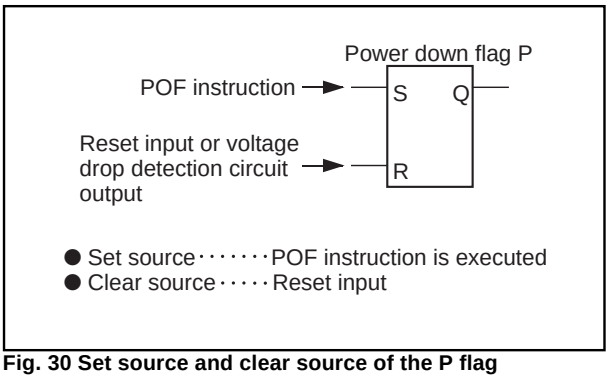



**Fig. 30 Set source and clear source of the P flag**

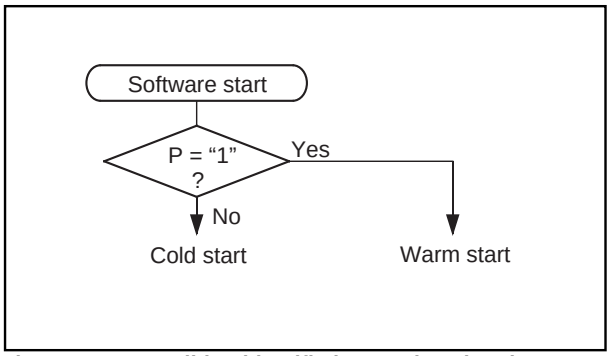



**Fig. 31 Start condition identified example using the SNZP instruction**

**Table 15 Key-on wakeup control register and pull-up control register**

| Key-on wakeup control register K0 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W |
|---|---|---|---|---|---|
| $K0_3$ | Port $P4_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K0_2$ | Port $P4_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K0_1$ | Port $P4_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K0_0$ | Port $P4_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |

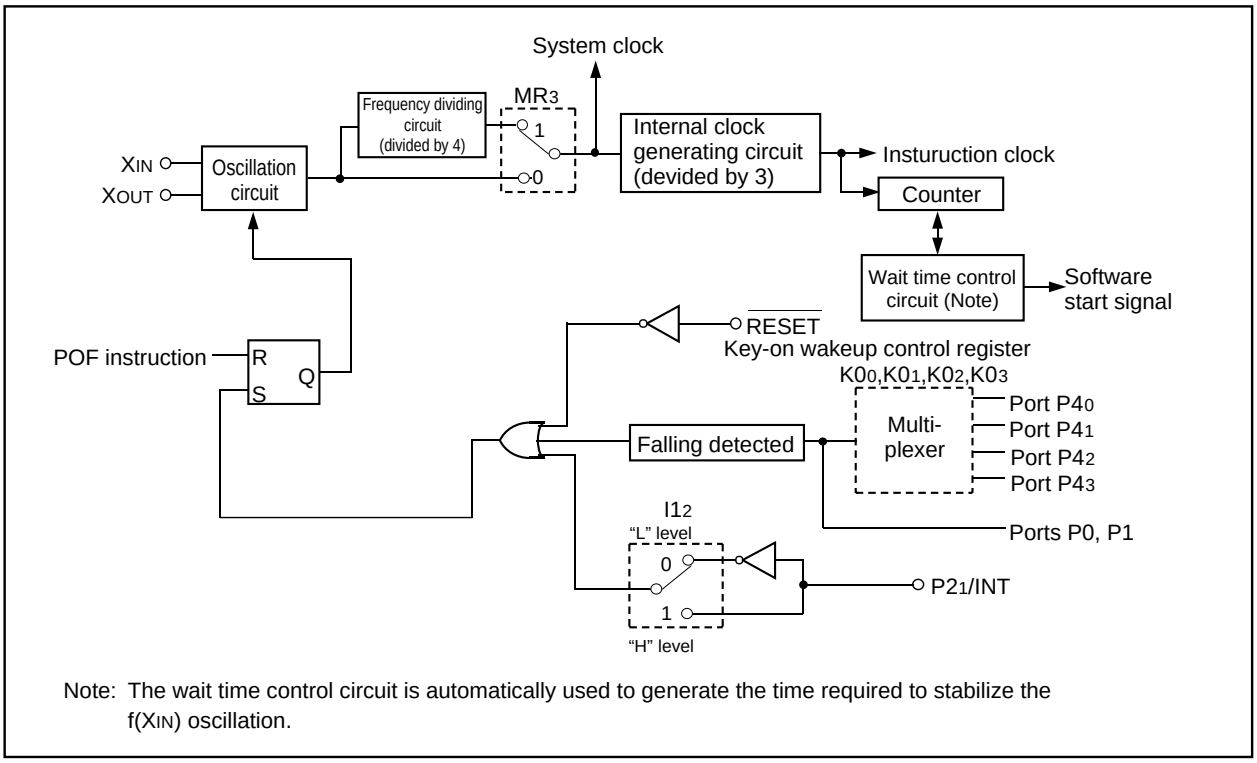
| Pull-up control register PU0 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W |
|---|---|---|---|---|---|
| $PU0_3$ | Port $P4_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_2$ | Port $P4_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_1$ | Port $P4_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_0$ | Port $P4_0$ and $P0_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |

Note: "R" represents read enabled, and "W" represents write enabled.

## CLOCK CONTROL

The clock control circuit consists of the following circuits.

- Clock generating circuit
- Control circuit to stop the clock oscillation
- System clock selection circuit
- Instruction clock generating circuit
- Control circuit to return from the RAM back-up mode




Note: The wait time control circuit is automatically used to generate the time required to stabilize the f(XIN) oscillation.

**Fig. 32 Clock control circuit structure**

Clock signal f(XIN) is obtained by externally connecting a ceramic resonator. Connect this external circuit to pins XIN and XOUT at the shortest distance. A feedback resistor is built-in between pins XIN and XOUT.

## ROM ORDERING METHOD

Please submit the information described below when ordering Mask ROM.

(1) M34570M4-XXXFP Mask ROM Order Confirmation Form, M34570M8-XXXFP Mask ROM Order Confirmation Form, or M34570MD-XXXFP Mask ROM Order Confirmation Form ..... 1

(2) Data to be written into mask ROM ..... EPROM (three sets containing the identical data)

(3) Mark Specification Form ..... 1

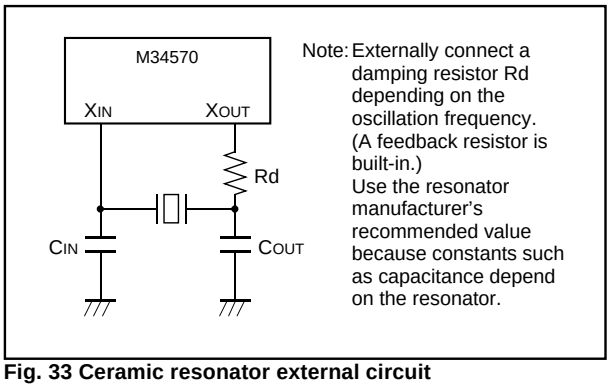



**Fig. 33 Ceramic resonator external circuit**

## LIST OF PRECAUTIONS

① Noise and latch-up prevention

Connect a capacitor on the following condition to prevent noise and latch-up;

- connect a capacitor (approx. 0.1 $\mu$F) between pins $V_{DD}$ and $V_{SS}$ at the shortest distance,
- equalize its wiring in width and length, and
- use the thickest wire.

In the One Time PROM version, $CNV_{SS}$ pin is also used as $V_{PP}$ pin. Accordingly, when using this pin, connect this pin to $V_{SS}$ through a resistor about 5 kΩ (connect this resistor to $CNV_{SS}/V_{PP}$ pin as close as possible).

② Prescaler

Stop the prescaler operation to change its frequency dividing ratio.

③ Count source

Stop timer 1, timer 2 or timer 3 counting to change its count source.

④ Reading the timer count value

Stop each of the timers and then execute the TAB1, TAB2 or TAB3 instruction to read timer 1, 2 or 3 data.

⑤ Writing to reload register R1

When writing the data to reload register R1 while timer 1 is operating, avoid a timing when timer 1 underflows.

⑥ Writing to reload register R3H

When writing the data to reload register R3H while timer 3 is operating, avoid a timing when timer 3 underflows.

⑦ Notes on timer 3 operation start

Set the timer 1 and register C2 before timer 3 is started to operate ($W3_3$="1").

⑧ Notes on carrier wave output auto-control operation stop

Stop the timer 1 ($W1_0$="0") after stopping the timer 3 ($W3_3$="0") while the carrier wave output is disabled in order to stop the carrier wave output auto-control operation.

⑨ Notes on setting carrier wave output control regiter C2

If the carrier wave output auto-control function is invalidated ($C2_0$="0") while the carrier wave output is auto-controlled, the output of port CARR retains the state when the auto-control is invalidated regardless of timer 1 underflow.

When the carrier wave output auto-control function is validated ($C2_0$="1") again after it is invalidated ($C2_0$="0"), the auto-control by timer 1 is validated again when the next timer 1 underflow occurs.

However, when the carrier wave output auto-control bit ($C2_0$) is changed during timer 1 underflow, the error-operation may occur.

⑩ Notes on timer 1 count source

When the carrier wave output auto-control function is selected, use the carrier wave CARRY as the timer 1 count source.

If the ORCLK is used as the count source, a short pulse may occur in port CARR output because ORCLK is not synchronized with the carrier wave.

⑪ Notes on writing to reload register R1 when carrier wave output auto-control operation

When the carrier wave output auto-control function is selected and data is set to reload register R1 while timer 1 is operating, avoid the timing that the contents of timer 1 becomes "0" to execute the T1AB instruction.

⑫ One Time PROM version

The operating power voltage of the One Time PROM version is within the range of 2.5 V to 5.5 V.

⑬ Multifunction

Note that the port $D_9$ output function and $P2_1$ input function can be used even when $T_{OUT}$ and INT pin function is selected.

⑭ POF instruction

Note that system cannot enter the RAM back-up state when executing only the POF instruction.

Execute the POF instruction immediately after executing the EPOF instruction to enter the RAM back-up.

Be sure to disable interrupts by executing the DI instruction before executing the EPOF instruction.

⑮ Program counter

Make sure that the $PC_H$ does not specify after the last page of the built-in ROM.

⑯ $P2_1$/INT pin

When the interrupt valid waveform of $P2_1$/INT pin is changed with the bit 2 of register I1 in software, be careful about the following notes.

- Clear the bit 0 of register V1 to "0" and then change the interrupt valid waveform of $P2_1$/INT pin with the bit 2 of register I1 (refer to Figure 34①).
- Clear the bit 2 of register I1 to "0" and execute the SNZ0 instruction to clear the EXF0 flag after executing at least one instruction (refer to Figure 34②). Depending on the input state of the $P2_1$/INT pin, the external 0 interrupt request flag (EXF0) may be set to "1" when the interrupt valid waveform is changed.

```
    ⋮
    LA      4     ; (XXX0₂)
    TV1A          ; The SNZ0 instruction is valid    ①
    LA      4
    TI1A          ; Change of the interrupt valid waveform
    NOP                                              ②
    SNZ0          ;The SNZ0 instruction is executed
    NOP
    ⋮
```

X : this bit is not related to the setting of INT.

**Fig. 34 External 0 interrupt program example**

## SYMBOL

The symbols shown below are used in the following list of instruction function and machine instructions.

| Symbol | Contents | Symbol | Contents |
|---|---|---|---|
| A | Register A (4 bits) | WDF1 | Watchdog timer flag 1 |
| B | Register B (4 bits) | WDF2 | Watchdog timer flag 2 |
| DR | Register D (3 bits) | WEF | Watchdog timer enable flag |
| E | Register E (8 bits) | INTE | Interrupt enable flag |
| C2 | Carrier wave output control register C2 (2 bits) | EXF0 | External 0 interrupt request flag |
| SI | 8-bit general-purpose register SI (8 bits) | P | Power down flag |
| V1 | Interrupt control register V1 (4 bits) | | |
| V2 | Interrupt control register V2 (4 bits) | D | Port D (10 bits) |
| I1 | Interrupt control register I1 (4 bits) | P0 | Port P0 (4 bits) |
| W1 | Timer control register W1 (4 bits) | P1 | Port P1 (4 bits) |
| W2 | Timer control register W2 (4 bits) | P2 | Port P2 (2 bits) |
| W3 | Timer control register W3 (4 bits) | P3 | Port P3 (4 bits) |
| W5 | Timer count value store register W5 (2 bits) | P4 | Port P4 (4 bits) |
| K0 | Key-on wakeup control register K0 (4 bits) | x | Hexadecimal variable |
| PU0 | Pull-up control register PU0 (4 bits) | y | Hexadecimal variable |
| MR | Clock control register MR (4 bits) | z | Hexadecimal variable |
| X | Register X (4 bits) | p | Hexadecimal variable |
| Y | Register Y (4 bits) | n | Hexadecimal constant which represents the |
| Z | Register Z (2 bits) | | immediate value |
| DP | Data pointer (10 bits) | i | Hexadecimal constant which represents the |
| | (It consists of registers X, Y, and Z) | | immediate value |
| PC | Program counter (14 bits) | j | Hexadecimal constant which represents the |
| $PC_H$ | High-order 7 bits of program counter | | immediate value |
| $PC_L$ | Low-order 7 bits of program counter | $A_3A_2A_1A_0$ | Binary notation of hexadecimal variable A |
| SK | Stack register (14 bits × 8) | | (same for others) |
| SP | Stack pointer (3 bits) | | |
| CY | Carry flag | ← | Direction of data movement |
| R1 | Timer 1 reload register | ↔ | Data exchange between a register and memory |
| R2 | Timer 2 reload register | ? | Decision of state shown before “?” |
| R3H | Timer 3 reload register | ( ) | Contents of registers and memories |
| R3L | Timer 3 reload register | — | Negate, Flag unchanged after executing |
| T1 | Timer 1 | | instruction |
| T2 | Timer 2 | M(DP) | RAM address pointed by the data pointer |
| T3 | Timer 3 | a | Label indicating address $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$ |
| T1F | Timer 1 interrupt request flag | p, a | Label indicating address $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$ |
| T2F | Timer 2 interrupt request flag | | in page $p_5$ $p_4$ $p_3$ $p_2$ $p_1$ $p_0$ |
| T3F | Timer 3 interrupt request flag | C | Hex. C + Hex. number x (also same for others) |
| | | + | |
| | | x | |

Note : The 4570 Group just invalidates the next instruction when a skip is performed. The contents of program counter is not increased by 2. Accordingly, the number of cycles does not change even if skip is not performed. However, the cycle count becomes “1” if the TABP p, RT, or RTS instruction is skipped.

## LIST OF INSTRUCTION FUNCTION

| Grouping | Mnemonic | Function |
|---|---|---|
| Register to register transfer | TAB | (A) ← (B) |
| | TBA | (B) ← (A) |
| | TAY | (A) ← (Y) |
| | TYA | (Y) ← (A) |
| | TEAB | ($E_7$–$E_4$) ← (B)<br>($E_3$–$E_0$) ← (A) |
| | TABE | (B) ← ($E_7$–$E_4$)<br>(A) ← ($E_3$–$E_0$) |
| | TDA | ($DR_2$–$DR_0$) ← ($A_2$–$A_0$) |
| | TAD | ($A_2$–$A_0$) ← ($DR_2$–$DR_0$)<br>($A_3$) ← 0 |
| | TAZ | ($A_1$, $A_0$) ← ($Z_1$, $Z_0$)<br>($A_3$, $A_2$) ← 0 |
| | TAX | (A) ← (X) |
| | TASP | ($A_2$–$A_0$) ← ($SP_2$–$SP_0$)<br>($A_3$) ← 0 |
| RAM addresses | LXY x, y | (X) ← x, x = 0 to 15<br>(Y) ← y, y = 0 to 15 |
| | LZ z | (Z) ← z, z = 0 to 3 |
| | INY | (Y) ← (Y) + 1 |
| | DEY | (Y) ← (Y) – 1 |
| RAM to register transfer | TAM j | (A) ← (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15 |
| | XAM j | (A) ← → (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15 |
| | XAMD j | (A) ← → (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15<br>(Y) ← (Y) – 1 |
| RAM to register transfer | XAMI j | (A) ← → (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15<br>(Y) ← (Y) + 1 |
| | TMA j | (M(DP)) ← (A)<br>(X) ← (X)EXOR(j)<br>j = 0 to 15 |
| Arithmetic operation | LA n | (A) ← n<br>n = 0 to 15 |
| | TABP p | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>($PC_H$) ← p<br>($PC_L$) ← ($DR_2$–$DR_0$, $A_3$–$A_0$)<br>(W5) ← (ROM(PC))$_{9\ to\ 8}$<br>(B) ← (ROM(PC))$_{7\ to\ 4}$<br>(A) ← (ROM(PC))$_{3\ to\ 0}$<br>(PC) ← (SK(SP))<br>(SP) ← (SP) – 1 |
| | AM | (A) ← (A) + (M(DP)) |
| | AMC | (A) ← (A) + (M(DP)) + (CY)<br>(CY) ← Carry |
| | A n | (A) ← (A) + n<br>n = 0 to 15 |
| | AND | (A) ← (A)AND(M(DP)) |
| | OR | (A) ← (A)OR(M(DP)) |
| | SC | (CY) ← 1 |
| | RC | (CY) ← 0 |
| | SZC | (CY) = 0 ? |
| | CMA | (A) ← $\overline{(A)}$ |
| | RAR | → CY → $A_3A_2A_1A_0$ → |
| Bit operation | SB j | (Mj(DP)) ← 1<br>j = 0 to 3 |
| | RB j | (Mj(DP)) ← 0<br>j = 0 to 3 |
| | SZB j | (Mj(DP)) = 0 ?<br>j = 0 to 3 |
| Comparison operation | SEAM | (A) = (M(DP)) ? |
| | SEA n | (A) = n ?<br>n = 0 to 15 |
| Branch operation | B a | ($PC_L$) ← $a_6$–$a_0$ |
| | BL p, a | ($PC_H$) ← p<br>($PC_L$) ← $a_6$–$a_0$ |
| | BLA p | ($PC_H$) ← p<br>($PC_L$) ← ($DR_2$–$DR_0$, $A_3$–$A_0$) |
| Subroutine operation | BM a | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>($PC_H$) ← 2<br>($PC_L$) ← $a_6$–$a_0$ |
| | BML p, a | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>($PC_H$) ← p<br>($PC_L$) ← $a_6$–$a_0$ |
| | BMLA p | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>($PC_H$) ← p<br>($PC_L$) ← ($DR_2$–$DR_0$, $A_3$–$A_0$) |
| Return operation | RTI | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 |
| | RT | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 |
| | RTS | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 |

## LIST OF INSTRUCTION FUNCTION (CONTINUED)

| Grouping | Mnemonic | Function |
|---|---|---|
| Interrupt operation | DI | (INTE) ← 0 |
| | EI | (INTE) ← 1 |
| | SNZ0 | (EXF0) = 1 ?<br>After skipping the next instruction,<br>(EXF0) ← 0 |
| | SNZI0 | $I1_2$ = 1 : (INT0) = "H" ?<br>$I1_2$ = 0 : (INT0) = "L" ? |
| | TAV1 | (A) ← (V1) |
| | TV1A | (V1) ← (A) |
| | TAV2 | (A) ← (V2) |
| | TV2A | (V2) ← (A) |
| | TAI1 | (A) ← (I1) |
| | TI1A | (I1) ← (A) |
| Timer operation | TAW1 | (A) ← (W1) |
| | TW1A | (W1) ← (A) |
| | TAW2 | (A) ← (W2) |
| | TW2A | (W2) ← (A) |
| | TAW3 | (A) ← (W3) |
| | TW3A | (W3) ← (A) |
| | TAW5 | (A) ← (0, 0, $W5_1$, $W5_0$) |
| | TW5A | ($W5_1$, $W5_0$) ← ($A_1$, $A_0$) |
| | TAB1 | (W5) ← ($T1_9$–$T1_8$)<br>(B) ← ($T1_7$–$T1_4$)<br>(A) ← ($T1_3$–$T1_0$) |
| | T1AB | at timer 1 stop ($W1_0$=0)<br>($R1_9$–$R1_8$) ← (W5)<br>($T1_9$–$T1_8$) ← (W5)<br>($R1_7$–$R1_4$) ← (B)<br>($T1_7$–$T1_4$) ← (B)<br>($R1_3$–$R1_0$) ← (A)<br>($T1_3$–$T1_0$) ← (A)<br>At timer 1 operating ($W1_0$=1),<br>($R1_9$–$R1_8$) ← (W5)<br>($R1_7$–$R1_4$) ← (B)<br>($R1_3$–$R1_0$) ← (A) |
| | TAB2 | (B) ← ($T2_7$–$T2_4$)<br>(A) ← ($T2_3$–$T2_0$) |
| | T2AB | ($R2_7$–$R2_4$) ← (B)<br>($T2_7$–$T2_4$) ← (B)<br>($R2_3$–$R2_0$) ← (A)<br>($T2_3$–$T2_0$) ← (A) |
| | TR2AB | ($R2_7$–$R2_4$) ← (B)<br>($R2_3$–$R2_0$) ← (A) |
| Timer operation | TAB3 | (B) ← ($T3_7$–$T3_4$)<br>(A) ← ($T3_3$–$T3_0$) |
| | T3AB | ($R3L_7$–$R3L_4$) ← (B)<br>($T3_7$–$T3_4$) ← (B)<br>($R3L_3$–$R3L_0$) ← (A)<br>($T3_3$–$T3_0$) ← (A) |
| | T3HAB | ($R3H_7$–$R3H_4$) ← (B)<br>($R3H_3$–$R3H_0$) ← (A) |
| | SNZT1 | (T1F) = 1 ?<br>After skipping the next instruction,<br>(T1F) ← 0 |
| | SNZT2 | (T2F) = 1 ?<br>After skipping the next instruction,<br>(T2F) ← 0 |
| | SNZT3 | (T3F) = 1 ?<br>After skipping the next instruction,<br>(T3F) ← 0 |

## LIST OF INSTRUCTION FUNCTION (CONTINUED)

| Grouping | Mnemonic | Function | Grouping | Mnemonic | Function |
|---|---|---|---|---|---|
| Input/Output operation | IAP0 | (A) ← (P0) | Other operation | NOP | (PC) ← (PC) + 1 |
| | OP0A | (P0) ← (A) | | POF | RAM back-up mode |
| | IAP1 | (A) ← (P1) | | EPOF | POF instruction valid |
| | OP1A | (P1) ← (A) | | SNZP | (P) = 1 ? |
| | IAP2 | ($A_1$, $A_0$) ← ($P2_1$, $P2_0$)<br>($A_3$, $A_2$) ← (0) | | WRST | (WDF1) ← 0, (WEF) ←1 |
| | IAP3 | (A) ← (P3) | | TAMR | (A) ← ($MR_3$–$MR_0$) |
| | OP3A | (P3) ← (A) | | TMRA | ($MR_3$–$MR_0$) ← (A) |
| | IAP4 | (A) ← (P4) | | TABSI | (B) ← ($SI_7$–$SI_4$)<br>(A) ← ($SI_3$–$SI_0$) |
| | CLD | (D) ← 1 | | TSIAB | ($SI_7$–$SI_4$) ← (B)<br>($SI_3$–$SI_0$) ← (A) |
| | RD | (D(Y)) ← 0<br>(Y) = 0 to 9 | | SBK | When executing the TABP p instruction,<br>$p_6$ ← 1 |
| | SD | (D(Y)) ← 1<br>(Y) = 0 to 9 | | RBK | When executing the TABP p instruction,<br>$p_6$ ← 0 |
| | TK0A | (K0) ← (A) | | | |
| | TAK0 | (A) ← (K0) | | | |
| | TPU0A | (PU0) ← (A) | | | |
| | TAPU0 | (A) ← (PU0) | | | |
| Carrier wave generating operation | TC2A | ($C2_1$, $C2_0$) ← ($A_1$, $A_0$) | | | |

## INSTRUCTION CODE TABLE

| D9—D4 / D3—D0 Hex. notation | 000000 00 | 000001 01 | 000010 02 | 000011 03 | 000100 04 | 000101 05 | 000110 06 | 000111 07 | 001000 08 | 001001 09 | 001010 0A | 001011 0B | 001100 0C | 001101 0D | 001110 0E | 001111 0F | 010000 \| 010111 10—17 | 011000 \| 011111 18—1F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | 0 | NOP | BLA | SZB 0 | BMLA | RBK | TASP | A 0 | LA 0 | TABP 0* | TABP 16* | TABP 32** | TABP 48** | BML | BML | BL | BL | BM | B |
| 0001 | 1 | — | CLD | SZB 1 | — | SBK | TAD | A 1 | LA 1 | TABP 1* | TABP 17* | TABP 33** | TABP 49** | BML | BML | BL | BL | BM | B |
| 0010 | 2 | POF | — | SZB 2 | — | — | TAX | A 2 | LA 2 | TABP 2* | TABP 18* | TABP 34** | TABP 50** | BML | BML | BL | BL | BM | B |
| 0011 | 3 | SNZP | INY | SZB 3 | — | — | TAZ | A 3 | LA 3 | TABP 3* | TABP 19* | TABP 35** | TABP 51** | BML | BML | BL | BL | BM | B |
| 0100 | 4 | DI | RD | — | — | RT | TAV1 | A 4 | LA 4 | TABP 4* | TABP 20* | TABP 36** | TABP 52** | BML | BML | BL | BL | BM | B |
| 0101 | 5 | EI | SD | SEAn | — | RTS | TAV2 | A 5 | LA 5 | TABP 5* | TABP 21* | TABP 37** | TABP 53** | BML | BML | BL | BL | BM | B |
| 0110 | 6 | RC | — | SEAM | — | RTI | — | A 6 | LA 6 | TABP 6* | TABP 22* | TABP 38** | TABP 54** | BML | BML | BL | BL | BM | B |
| 0111 | 7 | SC | DEY | — | — | — | — | A 7 | LA 7 | TABP 7* | TABP 23* | TABP 39** | TABP 55** | BML | BML | BL | BL | BM | B |
| 1000 | 8 | — | AND | — | SNZ0 | LZ 0 | — | A 8 | LA 8 | TABP 8* | TABP 24* | TABP 40** | TABP 56** | BML | BML | BL | BL | BM | B |
| 1001 | 9 | — | OR | TDA | — | LZ 1 | — | A 9 | LA 9 | TABP 9* | TABP 25* | TABP 41** | TABP 57** | BML | BML | BL | BL | BM | B |
| 1010 | A | AM | TEAB | TABE | SNZI0 | LZ 2 | — | A 10 | LA 10 | TABP 10* | TABP 26* | TABP 42** | TABP 58** | BML | BML | BL | BL | BM | B |
| 1011 | B | AMC | — | — | — | LZ 3 | EPOF | A 11 | LA 11 | TABP 11* | TABP 27* | TABP 43** | TABP 59** | BML | BML | BL | BL | BM | B |
| 1100 | C | TYA | CMA | — | — | RB 0 | SB 0 | A 12 | LA 12 | TABP 12* | TABP 28* | TABP 44** | TABP 60** | BML | BML | BL | BL | BM | B |
| 1101 | D | — | RAR | — | — | RB 1 | SB 1 | A 13 | LA 13 | TABP 13* | TABP 29* | TABP 45** | TABP 61** | BML | BML | BL | BL | BM | B |
| 1110 | E | TBA | TAB | — | TV2A | RB 2 | SB 2 | A 14 | LA 14 | TABP 14* | TABP 30* | TABP 46** | TABP 62** | BML | BML | BL | BL | BM | B |
| 1111 | F | — | TAY | SZC | TV1A | RB 3 | SB 3 | A 15 | LA 15 | TABP 15* | TABP 31* | TABP 47** | TABP 63** | BML | BML | BL | BL | BM | B |

The above table shows the relationship between machine language codes and machine language instructions. D3—D0 show the low-order 4 bits of the machine language code, and D9—D4 show the high-order 6 bits of the machine language code. The hexadecimal representation of the code is also provided. There are one-word instructions and two-word instructions, but only the first word of each instruction is shown. Do not use code marked "—."
** cannot be used at M34570M4.

For M34570M4/M8/E8, the SBK and RBK instructions cannot be used.
For M34570MD/ED, the pages which is referred with the TABP instruction (*, **) can be switched with the SBK and RBK instructions.
After executing the SBK instruction, the pages which can be referred with the TABP instruction are 64 to 127. (ex. TABP 0 →TABP 64)
After executing the RBK instruction, the pages which can be referred with the TABP instruction are 0 to 63.
If the SBK instruction is not executed, the pages which can be referred with the TABP instruction are always 0 to 63.

The codes for the second word of a two-word instruction are described below.

| | The second word |
|---|---|
| BL | 1 p p a a a a a a a a |
| BML | 1 p p a a a a a a a a |
| BLA | 1 p p p 0 0 p p p p |
| BMLA | 1 p p p 0 0 p p p p |
| SEA | 0 0 0 1 1 1 n n n n |
| SZD | 0 0 0 0 1 0 1 0 1 1 |

## INSTRUCTION CODE TABLE (CONTINUED)

| D9—D4 / D3—D0 | Hex. notation | 100000 | 100001 | 100010 | 100011 | 100100 | 100101 | 100110 | 100111 | 101000 | 101001 | 101010 | 101011 | 101100 | 101101 | 101110 | 101111 | 110000 \| 111111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F | 30—3F |
| 0000 | 0 | — | TW3A | OP0A | T1AB | — | — | IAP0 | TAB1 | SNZT1 | — | WRST | TMA 0 | TAM 0 | XAM 0 | XAMI 0 | XAMD 0 | LXY |
| 0001 | 1 | — | — | OP1A | T2AB | — | — | IAP1 | TAB2 | SNZT2 | — | — | TMA 1 | TAM 1 | XAM 1 | XAMI 1 | XAMD 1 | LXY |
| 0010 | 2 | — | TW5A | — | T3AB | — | TAMR | IAP2 | TAB3 | SNZT3 | — | — | TMA 2 | TAM 2 | XAM 2 | XAMI 2 | XAMD 2 | LXY |
| 0011 | 3 | — | — | OP3A | — | — | TAI1 | IAP3 | — | — | — | — | TMA 3 | TAM 3 | XAM 3 | XAMI 3 | XAMD 3 | LXY |
| 0100 | 4 | — | — | — | — | — | — | IAP4 | — | — | — | — | TMA 4 | TAM 4 | XAM 4 | XAMI 4 | XAMD 4 | LXY |
| 0101 | 5 | — | — | — | — | — | — | — | — | — | — | — | TMA 5 | TAM 5 | XAM 5 | XAMI 5 | XAMD 5 | LXY |
| 0110 | 6 | — | TMRA | — | — | — | TAK0 | — | — | — | — | — | TMA 6 | TAM 6 | XAM 6 | XAMI 6 | XAMD 6 | LXY |
| 0111 | 7 | — | TI1A | — | — | — | TAPU0 | — | — | — | — | — | TMA 7 | TAM 7 | XAM 7 | XAMI 7 | XAMD 7 | LXY |
| 1000 | 8 | — | — | — | TSIAB | — | — | — | TABSI | — | — | — | TMA 8 | TAM 8 | XAM 8 | XAMI 8 | XAMD 8 | LXY |
| 1001 | 9 | — | — | — | — | — | — | — | — | — | — | TC2A | TMA 9 | TAM 9 | XAM 9 | XAMI 9 | XAMD 9 | LXY |
| 1010 | A | — | — | — | TR2AB | — | — | — | — | — | — | — | TMA 10 | TAM 10 | XAM 10 | XAMI 10 | XAMD 10 | LXY |
| 1011 | B | — | TK0A | — | — | TAW1 | — | — | — | — | — | — | TMA 11 | TAM 11 | XAM 11 | XAMI 11 | XAMD 11 | LXY |
| 1100 | C | — | — | — | — | TAW2 | — | — | — | — | — | — | TMA 12 | TAM 12 | XAM 12 | XAMI 12 | XAMD 12 | LXY |
| 1101 | D | — | — | TPU0A | T3HAB | TAW3 | — | — | — | — | — | — | TMA 13 | TAM 13 | XAM 13 | XAMI 13 | XAMD 13 | LXY |
| 1110 | E | TW1A | — | — | — | — | — | — | — | — | — | — | TMA 14 | TAM 14 | XAM 14 | XAMI 14 | XAMD 14 | LXY |
| 1111 | F | TW2A | — | — | — | TAW5 | — | — | — | — | — | — | TMA 15 | TAM 15 | XAM 15 | XAMI 15 | XAMD 15 | LXY |

The above table shows the relationship between machine language codes and machine language instructions. $D_{3}$–$D_{0}$ show the low-order 4 bits of the machine language code, and $D_{9}$–$D_{4}$ show the high-order 6 bits of the machine language code. The hexadecimal representation of the code is also provided. There are one-word instructions and two-word instructions, but only the first word of each instruction is shown. Do not use code marked "–."

The codes for the second word of a two-word instruction are described below.

| | The second word |
|---|---|
| BL | 1 p p a a a a a a a a |
| BML | 1 p p a a a a a a a a |
| BLA | 1 p p p p 0 0 p p p p |
| BMLA | 1 p p p p 0 0 p p p p |
| SEA | 0 0 0 1 1 1 n n n n |
| SZD | 0 0 0 0 1 0 1 0 1 1 |

## MACHINE INSTRUCTIONS

| Parameter / Type of instructions | Mnemonic | Instruction code D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|
| Register to register transfer | TAB | 0 0 0 0 0 1 1 1 1 0 | 0 1 E | 1 | 1 | (A) ← (B) | – | – | Transfers the contents of register B to register A. |
| | TBA | 0 0 0 0 0 0 1 1 1 0 | 0 0 E | 1 | 1 | (B) ← (A) | – | – | Transfers the contents of register A to register B. |
| | TAY | 0 0 0 0 0 1 1 1 1 1 | 0 1 F | 1 | 1 | (A) ← (Y) | – | – | Transfers the contents of register Y to register A. |
| | TYA | 0 0 0 0 0 0 1 1 0 0 | 0 0 C | 1 | 1 | (Y) ← (A) | – | – | Transfers the contents of register A to register Y. |
| | TEAB | 0 0 0 0 0 1 1 0 1 0 | 0 1 A | 1 | 1 | ($E_7$–$E_4$) ← (B)<br>($E_3$–$E_0$) ← (A) | – | – | Transfers the contents of registers A and B to register E. |
| | TABE | 0 0 0 0 1 0 1 0 1 0 | 0 2 A | 1 | 1 | (B) ← ($E_7$–$E_4$)<br>(A) ← ($E_3$–$E_0$) | – | – | Transfers the contents of register E to registers A and B. |
| | TDA | 0 0 0 0 1 0 1 0 0 1 | 0 2 9 | 1 | 1 | ($DR_2$–$DR_0$) ← ($A_2$–$A_0$) | – | – | Transfers the contents of register A to register D. |
| | TAD | 0 0 0 1 0 1 0 0 0 1 | 0 5 1 | 1 | 1 | ($A_2$–$A_0$) ← ($DR_2$–$DR_0$)<br>($A_3$) ← 0 | – | – | Transfers the contents of register D to register A. |
| | TAZ | 0 0 0 1 0 1 0 0 1 1 | 0 5 3 | 1 | 1 | ($A_1$, $A_0$) ← ($Z_1$, $Z_0$)<br>($A_3$, $A_2$) ← 0 | – | – | Transfers the contents of register Z to register A. |
| | TAX | 0 0 0 1 0 1 0 0 1 0 | 0 5 2 | 1 | 1 | (A) ← (X) | – | – | Transfers the contents of register X to register A. |
| | TASP | 0 0 0 1 0 1 0 0 0 0 | 0 5 0 | 1 | 1 | ($A_2$–$A_0$) ← ($SP_2$–$SP_0$)<br>($A_3$) ← 0 | – | – | Transfers the contents of stack pointer (SP) to register A. |
| RAM addresses | LXY x, y | 1 1 $x_3$ $x_2$ $x_1$ $x_0$ $y_3$ $y_2$ $y_1$ $y_0$ | 3 x y | 1 | 1 | (X) ← x, x = 0 to 15<br>(Y) ← y, y = 0 to 15 | Continuous description | – | Loads the value x in the immediate field to register X, and the value y in the immediate field to register Y.<br>When the LXY instructions are continuously coded and executed, only the first LXY instruction is executed and other LXY instructions coded continuously are skipped. |
| | LZ z | 0 0 0 1 0 0 1 0 $z_1$ $z_0$ | 0 4 8 +z | 1 | 1 | (Z) ← z, z = 0 to 3 | – | – | Loads the value z in the immediate field to register Z. |
| | INY | 0 0 0 0 0 1 0 0 1 1 | 0 1 3 | 1 | 1 | (Y) ← (Y) + 1 | (Y) = 0 | – | Adds 1 to the contents of register Y. As a result of addition, when the contents of register Y is 0, the next instruction is skipped. |
| | DEY | 0 0 0 0 0 1 0 1 1 1 | 0 1 7 | 1 | 1 | (Y) ← (Y) – 1 | (Y) = 15 | – | Subtracts 1 from the contents of register Y. As a result of subtraction, when the contents of register Y is 15, the next instruction is skipped. |



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## MACHINE INSTRUCTIONS (CONTINUED)

| Type of instructions \ Parameter | Mnemonic | Instruction code D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|
| RAM to register transfer | TAM j | 1 0 1 1 0 0 j j j j | 2 C j | 1 | 1 | (A) ← (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15 | – | – | After transferring the contents of M(DP) to register A, an exclusive OR operation is performed between register X and the value j in the immediate field, and stores the result in register X. |
| | XAM j | 1 0 1 1 0 1 j j j j | 2 D j | 1 | 1 | (A) ← → (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15 | – | – | After exchanging the contents of M(DP) with the contents of register A, an exclusive OR operation is performed between register X and the value j in the immediate field, and stores the result in register X. |
| | XAMD j | 1 0 1 1 1 1 j j j j | 2 F j | 1 | 1 | (A) ← → (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15<br>(Y) ← (Y) – 1 | (Y) = 15 | – | After exchanging the contents of M(DP) with the contents of register A, an exclusive OR operation is performed between register X and the value j in the immediate field, and stores the result in register X. Subtracts 1 from the contents of register Y. As a result of subtraction, when the contents of register Y is 15, the next instruction is skipped. |
| | XAMI j | 1 0 1 1 1 0 j j j j | 2 E j | 1 | 1 | (A) ← → (M(DP))<br>(X) ← (X)EXOR(j)<br>j = 0 to 15<br>(Y) ← (Y) + 1 | (Y) = 0 | – | After exchanging the contents of M(DP) with the contents of register A, an exclusive OR operation is performed between register X and the value j in the immediate field, and stores the result in register X. Adds 1 to the contents of register Y. As a result of addition, when the contents of register Y is 0, the next instruction is skipped. |
| | TMA j | 1 0 1 0 1 1 j j j j | 2 B j | 1 | 1 | (M(DP)) ← (A)<br>(X) ← (X)EXOR(j)<br>j = 0 to 15 | – | – | After transferring the contents of register A to M(DP), an exclusive OR operation is performed between register X and the value j in the immediate field, and stores the result in register X. |
| Arithmetic operation | LA n | 0 0 0 1 1 1 n n n n | 0 7 n | 1 | 1 | (A) ← n<br>n = 0 to 15 | Continuous description | – | Loads the value n in the immediate field to register A.<br>When the LA instructions are continuously coded and executed, only the first LA instruction is executed and other LA instructions coded continuously are skipped. |
| | TABP p | 0 0 1 0 $p_5$ $p_4$ $p_3$ $p_2$ $p_1$ $p_0$ | 0 8 p<br>+p | 1 | 3 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>($PC_H$) ← p<br>($PC_L$) ← ($DR_2$–$DR_0$, $A_3$–$A_0$)<br>(W5) ← (ROM(PC))$_{9\ to\ 8}$<br>(B) ← (ROM(PC))$_{7\ to\ 4}$<br>(A) ← (ROM(PC))$_{3\ to\ 0}$<br>(PC) ← (SK(SP))<br>(SP) ← (SP) – 1<br>(Note) | – | – | Transfers bits 9 and 8 to register W5, bits 7 to 4 to register B and bits 3 to 0 to register A. These bits 9 to 0 are the ROM pattern in address ($DR_2$ $DR_1$ $DR_0$ $A_3$ $A_2$ $A_1$ $A_0$)$_2$ specified by registers A and D in page p.<br>When this instruction is executed, 1 stage of stack register is used.<br>When this instruction is executed after executing the SBK instruction, pages 64 to 127 are specified.<br>When this instruction is executed after executing the RBK instruction, pages 0 to 63 are specified.<br>When this instruction is executed after system is released from reset or returned from RAM back-up, pages 0 to 63 are specified. |

Note: p is 0 to 31 for M34570M4 and p is 0 to 63 for M34570E8 and M34570M8.
p is 0 to 127 for M34570ED and M34570MD, and $p_6$ is specified with the SBK and RBK instructions.



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## MACHINE INSTRUCTIONS (CONTINUED)

| Parameter / Type of instructions | Mnemonic | Instruction code D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|
| Arithmetic operation | AM | 0 0 0 0 0 0 1 0 1 0 | 0 0 A | 1 | 1 | (A) ← (A) + (M(DP)) | – | – | Adds the contents of M(DP) to register A. Stores the result in register A. The contents of carry flag CY remains unchanged. |
| | AMC | 0 0 0 0 0 0 1 0 1 1 | 0 0 B | 1 | 1 | (A) ← (A) + (M(DP))+ (CY)<br>(CY) ← Carry | – | 0/1 | Adds the contents of M(DP) and carry flag CY to register A. Stores the result in register A and carry flag CY. |
| | A n | 0 0 0 1 1 0 n n n n | 0 6 n | 1 | 1 | (A) ← (A) + n<br>n = 0 to 15 | Overflow = 0 | – | Adds the value n in the immediate field to register A.<br>The contents of carry flag CY remains unchanged.<br>Skips the next instruction when there is no overflow as the result of operation. |
| | AND | 0 0 0 0 0 1 1 0 0 0 | 0 1 8 | 1 | 1 | (A) ← (A)AND(M(DP)) | – | – | Performs the AND operation between the contents of register A and the contents of M(DP), and stores the result in register A. |
| | OR | 0 0 0 0 0 1 1 0 0 1 | 0 1 9 | 1 | 1 | (A) ← (A)OR(M(DP)) | – | – | Performs the OR operation between the contents of register A and the contents of M(DP), and stores the result in register A. |
| | SC | 0 0 0 0 0 0 0 1 1 1 | 0 0 7 | 1 | 1 | (CY) ← 1 | – | 1 | Sets carry flag CY to "1." |
| | RC | 0 0 0 0 0 0 0 1 1 0 | 0 0 6 | 1 | 1 | (CY) ← 0 | – | 0 | Clears carry flag CY to "0." |
| | SZC | 0 0 0 0 1 0 1 1 1 1 | 0 2 F | 1 | 1 | (CY) = 0 ? | (CY) = 0 | – | Skips the next instruction when the contents of carry flag CY is "0." |
| | CMA | 0 0 0 0 0 1 1 1 0 0 | 0 1 C | 1 | 1 | (A) ← $\overline{(A)}$ | – | – | Stores the one's complement for register A's contents in register A. |
| | RAR | 0 0 0 0 0 1 1 1 0 1 | 0 1 D | 1 | 1 |  | – | 0/1 | Rotates the contents of register A including the contents of carry flag CY to the right by 1 bit. |
| Bit operation | SB j | 0 0 0 1 0 1 1 1 j j | 0 5 C+j | 1 | 1 | (Mj(DP)) ← 1<br>j = 0 to 3 | – | – | Sets the contents of bit j (bit specified by the value j in the immediate field) of M(DP) to "1." |
| | RB j | 0 0 0 1 0 0 1 1 j j | 0 4 C+j | 1 | 1 | (Mj(DP)) ← 0<br>j = 0 to 3 | – | – | Clears the contents of bit j (bit specified by the value j in the immediate field) of M(DP) to "0." |
| | SZB j | 0 0 0 0 1 0 0 0 j j | 0 2 j | 1 | 1 | (Mj(DP)) = 0 ?<br>j = 0 to 3 | (Mj(DP)) = 0<br>j = 0 to 3 | – | Skips the next instruction when the contents of bit j (bit specified by the value j in the immediate field) of M(DP) is "0." |
| Comparison operation | SEAM | 0 0 0 0 1 0 0 1 1 0 | 0 2 6 | 1 | 1 | (A) = (M(DP)) ? | (A) = (M(DP)) | – | Skips the next instruction when the contents of register A is equal to the contents of M(DP). |
| | SEA n | 0 0 0 0 1 0 0 1 0 1<br>0 0 0 1 1 1 n n n n | 0 2 5<br>0 7 n | 2 | 2 | (A) = n ?<br>n = 0 to 15 | (A) = n | – | Skips the next instruction when the contents of register A is equal to the value n in the immediate field. |



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## MACHINE INSTRUCTIONS (CONTINUED)

| Type of instructions | Mnemonic | Instruction code D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|
| Branch operation | B a | 0 1 1 a6 a5 a4 a3 a2 a1 a0 | 1 8 a<br>+a | 1 | 1 | (PCL) ← a6–a0 | – | – | Branch within a page : Branches to address a in the identical page. |
| | BL p, a | 0 0 1 1 1 p4 p3 p2 p1 p0 | 0 E p<br>+p | 2 | 2 | (PCH) ← p<br>(PCL) ← a6–a0<br>(Note) | – | – | Branch out of a page : Branches to address a in page p. |
| | | 1 0 p5 a6 a5 a4 a3 a2 a1 a0 | 2 p a<br>+a | | | | | | |
| | BLA p | 0 0 0 0 0 1 0 0 0 0 | 0 1 0 | 2 | 2 | (PCH) ← p<br>(PCL) ← (DR2–DR0, A3–A0)<br>(Note) | – | – | Branch out of a page : Branches to address $(DR_2\ DR_1\ DR_0\ A_3\ A_2\ A_1\ A_0)_2$ specified by registers D and A in page p. |
| | | 1 0 p5 p4 0 0 p3 p2 p1 p0 | 2 p p | | | | | | |
| Subroutine operation | BM a | 0 1 0 a6 a5 a4 a3 a2 a1 a0 | 1 a a | 1 | 1 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>(PCH) ← 2<br>(PCL) ← a6–a0 | – | – | Call the subroutine in page 2 : Calls the subroutine at address a in page 2. |
| | BML p, a | 0 0 1 1 0 p4 p3 p2 p1 p0 | 0 C p<br>+p | 2 | 2 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>(PCH) ← p | – | – | Call the subroutine : Calls the subroutine at address a in page p. |
| | | 1 0 p5 a6 a5 a4 a3 a2 a1 a0 | 2 p a<br>+a | | | (PCL) ← a6–a0<br>(Note) | | | |
| | BMLA p | 0 0 0 0 1 1 0 0 0 0 | 0 3 0 | 2 | 2 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC) | – | – | Call the subroutine : Calls the subroutine at address $(DR_2\ DR_1\ DR_0\ A_3\ A_2\ A_1\ A_0)_2$ specified by registers D and A in page p. |
| | | 1 0 p5 p4 0 0 p3 p2 p1 p0 | 2 p p | | | (PCH) ← p<br>(PCL) ← (DR2–DR0, A3–A0)<br>(Note) | | | |
| Return operation | RTI | 0 0 0 1 0 0 0 1 1 0 | 0 4 6 | 1 | 1 | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 | – | – | Returns from interrupt service routine to main routine.<br>Returns each value of data pointer (X, Y, Z), carry flag, skip status, NOP mode status by the continuous description of the LA/LXY instruction, register A and register B to the states just before interrupt. |
| | RT | 0 0 0 1 0 0 0 1 0 0 | 0 4 4 | 1 | 2 | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 | – | – | Returns from subroutine to the routine called the subroutine. |
| | RTS | 0 0 0 1 0 0 0 1 0 1 | 0 4 5 | 1 | 2 | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 | Skip unconditionally | – | Returns from subroutine to the routine called the subroutine, and skips the next instruction unconditionally. |

Note: p is 0 to 31 for M34570M4 and p is 0 to 63 for M34570E8 and M34570M8.
p is 0 to 127 for M34570ED and M34570MD, and $p_6$ is specified with the SBK and RBK instructions.



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## MACHINE INSTRUCTIONS (CONTINUED)

| Type of instructions | Mnemonic | Instruction code D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Interrupt operation | DI | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 0 4 | 1 | 1 | (INTE) ← 0 | – | – | Clears the interrupt enable flag INTE to "0," and disables the interrupt. |
| | EI | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 0 5 | 1 | 1 | (INTE) ← 1 | – | – | Sets the interrupt enable flag INTE to "1," and enables the interrupt. |
| | SNZ0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 3 8 | 1 | 1 | (EXF0) = 1 ?<br>After skipping the next instruction, (EXF0) ← 0 | (EXF0) = 1 | – | Skips the next instruction when the contents of EXF0 flag is "1."<br>After skipping, clears the EXF0 flag to "0." |
| | SNZI0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 3 A | 1 | 1 | $I1_2$ = 1 : (INT) = "H" ? | (INT) = "H"<br>However, $I1_2$ = 1 | – | When bit 2 ($I1_2$) of register I1 is "1" : Skips the next instruction when the level of INT pin is "H." |
| | | | | | | | | | | | | | | | $I1_2$ = 0 : (INT) = "L" ? | (INT) = "L"<br>However, $I1_2$ = 0 | – | When bit 2 ($I1_2$) of register I1 is "0" : Skips the next instruction when the level of INT pin is "L." |
| | TAV1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 5 4 | 1 | 1 | (A) ← (V1) | – | – | Transfers the contents of interrupt control register V1 to register A. |
| | TV1A | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 3 F | 1 | 1 | (V1) ← (A) | – | – | Transfers the contents of register A to interrupt control register V1. |
| | TAV2 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 5 5 | 1 | 1 | (A) ← (V2) | – | – | Transfers the contents of interrupt control register V2 to register A. |
| | TV2A | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 3 E | 1 | 1 | (V2) ← (A) | – | – | Transfers the contents of register A to interrupt control register V2. |
| | TAI1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 2 5 3 | 1 | 1 | (A) ← (I1) | – | – | Transfers the contents of interrupt control register I1 to register A. |
| | TI1A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 2 1 7 | 1 | 1 | (I1) ← (A) | – | – | Transfers the contents of register A to interrupt control register I1. |
| Timer operation | TAW1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 2 4 B | 1 | 1 | (A) ← (W1) | – | – | Transfers the contents of timer control register W1 to register A. |
| | TW1A | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 2 0 E | 1 | 1 | (W1) ← (A) | – | – | Transfers the contents of register A to timer control register W1. |
| | TAW2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 2 4 C | 1 | 1 | (A) ← (W2) | – | – | Transfers the contents of timer control register W2 to register A. |
| | TW2A | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 0 F | 1 | 1 | (W2) ← (A) | – | – | Transfers the contents of register A to timer control register W2. |
| | TAW3 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 2 4 D | 1 | 1 | (A) ← (W3) | – | – | Transfers the contents of timer control register W3 to register A. |
| | TW3A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 1 0 | 1 | 1 | (W3) ← (A) | – | – | Transfers the contents of register A to timer control register W3. |
| | TAW5 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 2 4 F | 1 | 1 | (A) ← (0, 0, $W5_1$, $W5_0$) | – | – | Transfers the contents of timer count value store register W5 to the low-order 2 bits of register A. The contents of the high-order 2 bits of register A is set to "0." |
| | TW5A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 1 2 | 1 | 1 | ($W5_1$, $W5_0$) ← ($A_1$, $A_0$) | – | – | Transfers the contents of the low-order 2 bits of register A to timer count value store register W5. |



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## MACHINE INSTRUCTIONS (CONTINUED)

| Parameter / Type of instructions | Mnemonic | Instruction code D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|
| Timer operation | TAB1 | 1 0 0 1 1 1 0 0 0 0 | 2 7 0 | 1 | 1 | (W5) ← (T1$_9$, T1$_8$)<br>(B) ← (T1$_7$–T1$_4$)<br>(A) ← (T1$_3$–T1$_0$) | – | – | Transfers the contents of the high-order 2 bits of timer 1 to register W5, and transfers the contents of the low-order 8 bits of timer 1 to registers A and B. |
| | T1AB | 1 0 0 0 1 1 0 0 0 0 | 2 3 0 | 1 | 1 | At timer 1 stop (W1$_0$=0),<br>(R1$_9$, R1$_8$) ← (W5)<br>(T1$_9$, T1$_8$) ← (W5)<br>(R1$_7$–R1$_4$) ← (B)<br>(T1$_7$–T1$_4$) ← (B)<br>(R1$_3$–R1$_0$) ← (A)<br>(T1$_3$–T1$_0$) ← (A)<br>At timer 1 operating (W1$_0$=1),<br>(R1$_9$, R1$_8$) ← (W5)<br>(R1$_7$–R1$_4$) ← (B)<br>(R1$_3$–R1$_0$) ← (A) | – | – | When stopping (W1$_0$=0), transfers the contents of register W5 to the contents of the high-order 2 bits of timer 1 and of the timer 1 reload register, and transfers the contents of registers A and B to the contents of the low-order 8 bits of timer 1 and of the timer 1 reload register.<br>When operating (W1$_0$=1), transfers the contents of register W5 to the contents of the high-order 2 bits of the timer 1 reload register, and transfers the contents of registers A and B to the contents of the low-order 8 bits of the timer 1 reload register. |
| | TAB2 | 1 0 0 1 1 1 0 0 0 1 | 2 7 1 | 1 | 1 | (B) ← (T2$_7$–T2$_4$)<br>(A) ← (T2$_3$–T2$_0$) | – | – | Transfers the contents of timer 2 to registers A and B. |
| | T2AB | 1 0 0 0 1 1 0 0 0 1 | 2 3 1 | 1 | 1 | (R2$_7$–R2$_4$) ← (B)<br>(T2$_7$–T2$_4$) ← (B)<br>(R2$_3$–R2$_0$) ← (A)<br>(T2$_3$–T2$_0$) ← (A) | – | – | Transfers the contents of registers A and B to timer 2 and timer 2 reload register. |
| | TR2AB | 1 0 0 0 1 1 1 0 1 0 | 2 3 A | 1 | 1 | (R2$_7$–R2$_4$) ← (B)<br>(R2$_3$–R2$_0$) ← (A) | – | – | Transfers the contents of registers A and B to timer 2 reload register. |
| | TAB3 | 1 0 0 1 1 1 0 0 1 0 | 2 7 2 | 1 | 1 | (B) ← (T3$_7$–T3$_4$)<br>(A) ← (T3$_3$–T3$_0$) | – | – | Transfers the contents of timer 3 to registers A and B. |
| | T3AB | 1 0 0 0 1 1 0 0 1 0 | 2 3 2 | 1 | 1 | (R3L$_7$–R3L$_4$) ← (B)<br>(T3$_7$–T3$_4$) ← (B)<br>(R3L$_3$–R3L$_0$) ← (A)<br>(T3$_3$–T3$_0$) ← (A) | – | – | Transfers the contents of registers A and B to timer 3 and timer 3 reload register R3L. |
| | T3HAB | 1 0 0 0 1 1 1 1 0 1 | 2 3 D | 1 | 1 | (R3H$_7$–R3H$_4$) ← (B)<br>(R3H$_3$–R3H$_0$) ← (A) | – | – | Transfers the contents of registers A and B to timer 3 reload register R3H. |



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## MACHINE INSTRUCTIONS (CONTINUED)

| Type of instructions | Mnemonic | Instruction code D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|
| Timer operation | SNZT1 | 1 0 1 0 0 0 0 0 0 0 | 2 8 0 | 1 | 1 | (T1F) = 1 ?<br>After skipping the next instruction<br>(T1F) ← 0 | (T1F) = 1 | – | Skips the next instruction when the contents of T1F flag is "1."<br>After skipping, clears T1F flag. |
| | SNZT2 | 1 0 1 0 0 0 0 0 0 1 | 2 8 1 | 1 | 1 | (T2F) = 1 ?<br>After skipping the next instruction<br>(T2F) ← 0 | (T2F) = 1 | – | Skips the next instruction when the contents of T2F flag is "1."<br>After skipping, clears T2F flag. |
| | SNZT3 | 1 0 1 0 0 0 0 0 1 0 | 2 8 2 | 1 | 1 | (T3F) = 1 ?<br>After skipping the next instruction<br>(T3F) ← 0 | (T3F) = 1 | – | Skips the next instruction when the contents of T3F flag is "1."<br>After skipping, clears T3F flag. |
| Input/Output operation | IAP0 | 1 0 0 1 1 0 0 0 0 0 | 2 6 0 | 1 | 1 | (A) ← (P0) | – | – | Transfers the input of port P0 to register A. |
| | OP0A | 1 0 0 0 1 0 0 0 0 0 | 2 2 0 | 1 | 1 | (P0) ← (A) | – | – | Outputs the contents of register A to port P0. |
| | IAP1 | 1 0 0 1 1 0 0 0 0 1 | 2 6 1 | 1 | 1 | (A) ← (P1) | – | – | Transfers the input of port P1 to register A. |
| | OP1A | 1 0 0 0 1 0 0 0 0 1 | 2 2 1 | 1 | 1 | (P1) ← (A) | – | – | Outputs the contents of register A to port P1. |
| | IAP2 | 1 0 0 1 1 0 0 0 1 0 | 2 6 2 | 1 | 1 | ($A_1$, $A_0$) ← ($P2_1$, $P2_0$)<br>($A_3$, $A_2$) ← 0 | – | – | Transfers the input of port P2 to register A. |
| | IAP3 | 1 0 0 1 1 0 0 0 1 1 | 2 6 3 | 1 | 1 | (A) ← (P3) | – | – | Transfers the input of port P3 to register A. |
| | OP3A | 1 0 0 0 1 0 0 0 1 1 | 2 2 3 | 1 | 1 | (P3) ← (A) | – | – | Outputs the contents of register A to port P3. |
| | IAP4 | 1 0 0 1 1 0 0 1 0 0 | 2 6 4 | 1 | 1 | (A) ← (P4) | – | – | Transfers the input of port P4 to register A. |
| | CLD | 0 0 0 0 0 1 0 0 0 1 | 0 1 1 | 1 | 1 | (D) ← 1 | – | – | Sets port D to "1." |
| | RD | 0 0 0 0 0 1 0 1 0 0 | 0 1 4 | 1 | 1 | (D(Y)) ← 0<br>(Y) = 0 to 9 | – | – | Clears a bit of port D specified by register Y to "0." |
| | SD | 0 0 0 0 0 1 0 1 0 1 | 0 1 5 | 1 | 1 | (D(Y)) ← 1<br>(Y) = 0 to 9 | – | – | Sets a bit of port D specified by register Y to "1." |
| | TK0A | 1 0 0 0 0 1 1 0 1 1 | 2 1 B | 1 | 1 | (K0) ← (A) | – | – | Transfers the contents of register A to key-on wakeup control register K0. |
| | TAK0 | 1 0 0 1 0 1 0 1 1 0 | 2 5 6 | 1 | 1 | (A) ← (K0) | – | – | Transfers the contents of key-on wakeup control register K0 to register A. |
| | TPU0A | 1 0 0 0 1 0 1 1 0 1 | 2 2 D | 1 | 1 | (PU0) ← (A) | – | – | Transfers the contents of register A to pull-up control register PU0. |
| | TAPU0 | 1 0 0 1 0 1 0 1 1 1 | 2 5 7 | 1 | 1 | (A) ← (PU0) | – | – | Transfers the contents of pull-up control register PU0 to register A. |



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

| Type of instructions \ Parameter | Mnemonic | Instruction code D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Hexadecimal notation | Number of words | Number of cycles | Function | Skip condition | Carry flag CY | Detailed description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Carrier generating circuit operation | TC2A | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2 A 9 | 1 | 1 | ($C2_1$, $C2_0$) ← ($A_1$, $A_0$) | – | – | Transfers the contents of register A to carrier wave output control register C2. |
| Other operation | NOP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 0 0 | 1 | 1 | (PC) ← (PC) + 1 | – | – | No operation |
| | POF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 0 2 | 1 | 1 | Transition to RAM back-up mode | – | – | Puts the system in RAM back-up mode state by executing the POF instruction after executing the EPOF instruction. |
| | EPOF | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 5 B | 1 | 1 | POF instruction valid | – | – | Validates the POF instruction which is executed after the EPOF instruction by executing the EPOF instruction. |
| | SNZP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 0 3 | 1 | 1 | (P) = 1 ? | (P) = 1 | – | Skips the next instruction when P flag is "1."<br>After skipping, P flag remains unchanged. |
| | WRST | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 A 0 | 1 | 1 | (WDF1) ← 0, (WEF) ← 1 | – | – | Operates the watchdog timer and initializes the watchdog timer flag (WDF1). |
| | TABSI | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 2 7 8 | 1 | 1 | (B) ← ($SI_7$–$SI_4$)<br>(A) ← ($SI_3$–$SI_0$) | – | – | Transfers the contents of general-purpose register SI to registers A and B. |
| | TSIAB | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 2 3 8 | 1 | 1 | ($SI_7$–$SI_4$) ← (B)<br>($SI_3$–$SI_0$) ← (A) | – | – | Transfers the contents of registers A and B to general-purpose register SI. |
| | TAMR | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 2 5 2 | 1 | 1 | (A) ← ($MR_3$–$MR_0$) | – | – | Transfers the contents of clock control register MR to register A. |
| | TMRA | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 2 1 6 | 1 | 1 | ($MR_3$–$MR_0$) ← (A) | – | – | Transfers the contents of register A to clock control register MR. |
| | SBK | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 4 1 | 1 | 1 | When executing the TABP p instruction, $p_6$ ← 1 | – | – | Data area which is referred when executing the TABP p instruction is set to pages 64 to 127.<br>This setting is valid only for the TABP p instruction. |
| | RBK | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 4 0 | 1 | 1 | When executing the TABP p instruction, $p_6$ ← 0 | – | – | Data area which is referred when executing the TABP p instruction is set to pages 0 to 63.<br>This setting is valid only for the TABP p instruction.<br>If the SBK instruction is not executed, $p_6$ when executing the TABP p instruction is "0." |



MITSUBISHI MICROCOMPUTERS
4570 Group
SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

## CONTROL REGISTERS

| Interrupt control register V1 | | at reset : $0000_2$ | RAM back-up : $0000_2$ | R/W |
|---|---|---|---|---|
| $V1_3$ | Timer 2 interrupt enable bit | 0 | Interrupt disabled (SNZT2 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZT2 instruction is invalid) | |
| $V1_2$ | Timer 1 interrupt enable bit | 0 | Interrupt disabled (SNZT1 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZT1 instruction is invalid) | |
| $V1_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V1_0$ | External 0 interrupt enable bit | 0 | Interrupt disabled (SNZ0 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZ0 instruction is invalid) | |

| Interrupt control register V2 | | at reset : $0000_2$ | at RAM back-up : $0000_2$ | R/W |
|---|---|---|---|---|
| $V2_3$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_2$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_0$ | Timer 3 interrupt enable bit | 0 | Interrupt disabled (SNZT3 instruction is valid) | |
| | | 1 | Interrupt enabled (SNZT3 instruction is invalid) | |

| Interrupt control register I1 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W |
|---|---|---|---|---|
| $I1_3$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $I1_2$ | Interrupt valid waveform for INT pin /return level selection bit (Note 2) | 0 | Falling waveform (“L” level of INT pin is recognized with the SNZI0 instruction)/“L” level | |
| | | 1 | Rising waveform (“H” level of INT pin is recognized with the SNZI0 instruction)/“H” level | |
| $I1_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $I1_0$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.
2: Depending on the input state of $P2_1$/INT pin, the external interrupt request flag EXF0 may be set to “1” when the contents of $I1_2$ is changed. Accordingly, set a value to bit 2 of register I1 and execute the SNZ0 instruction to clear the EXF0 flag after executing at least one instruction.

## CONTROL REGISTERS (CONTINUED)

| Timer control register W1 | | at reset : $0000_2$ | | at RAM back-up : $0000_2$ | R/W |
|---|---|---|---|---|---|
| $W1_3$ | Prescaler control bit | 0 | Stop (prescaler state initialized) | | |
| | | 1 | Operating | | |
| $W1_2$ | Prescaler dividing ratio selection bit | 0 | Instruction clock divided by 4 | | |
| | | 1 | Instruction clock divided by 8 | | |
| $W1_1$ | Timer 1 count source selection bit | 0 | Prescaler output (ORCLK) | | |
| | | 1 | Carrier output (CARRY) | | |
| $W1_0$ | Timer 1 control bit | 0 | Stop (state retained) | | |
| | | 1 | Operating | | |

| Timer control register W2 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W |
|---|---|---|---|---|---|
| $W2_3$ | Timer 2 control bit | 0 | Stop (state retained) | | |
| | | 1 | Operating | | |
| $W2_2$ | Port $D_9$/$T_{OUT}$ pin function selection bit | 0 | Port $D_9$ | | |
| | | 1 | $T_{OUT}$ pin | | |
| $W2_1$ | Timer 2 count source selection bits | $W2_1$ | $W2_0$ | Count source | |
| | | 0 | 0 | Prescaler output (ORCLK) | |
| $W2_0$ | | 0 | 1 | Timer 1 underflow signal | |
| | | 1 | 0 | Instruction clock | |
| | | 1 | 1 | 16-bit timer underflow signal | |

| Timer control register W3 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W |
|---|---|---|---|---|---|
| $W3_3$ | Timer 3 control bit | 0 | Stop (state retained) | | |
| | | 1 | Operating | | |
| $W3_2$ | Not used | 0 | This bit has no function, but read/write is enabled. | | |
| | | 1 | | | |
| $W3_1$ | Timer 3 count source selection bits | $W3_1$ | $W3_0$ | Count source | |
| | | 0 | 0 | Timer 2 underflow signal | |
| $W3_0$ | | 0 | 1 | Prescaler output (ORCLK) | |
| | | 1 | 0 | f($X_{IN}$) or f($X_{IN}$)/2 | |
| | | 1 | 1 | Not available | |

| Timer count value store register W5 | at reset : $00_2$ | at RAM back-up : state retained | R/W |
|---|---|---|---|
| 2-bit register. The contents of the high-order 2 bits (bits 9 and 8) of the 10-bit ROM pattern at address ($D_2D_1D_0A_3A_2A_1A_0$) in page p specified by registers D and A is stored in this register W5 with the TABP p instruction.<br>In addition, data can be transferred between the low-order 2 bits of register A and this register W5 with the TW5A or TAW5 instruction. Data can be read/written to/from the high-order 2 bits of timer 1 with the T1AB or TAB1 instruction. | | | |

Note: “R” represents read enabled, and “W” represents write enabled.

## CONTROL REGISTERS (CONTINUED)

| Carrier wave output control register C2 | | at reset : $00_2$ | at RAM back-up : $00_2$ | W |
|---|---|---|---|---|
| $C2_1$ | Port CARR output control bit | 0 | Port CARR "L" level output | |
| | | 1 | Port CARR "H" level output | |
| $C2_0$ | Carrier wave output auto-control bit | 0 | Auto-control output by timer 1 is invalid | |
| | | 1 | Auto-control output by timer 1 is valid | |

| Key-on wakeup control register K0 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W |
|---|---|---|---|---|
| $K0_3$ | Port $P4_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K0_2$ | Port $P4_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K0_1$ | Port $P4_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K0_0$ | Port $P4_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |

| Pull-up control register PU0 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W |
|---|---|---|---|---|
| $PU0_3$ | Port $P4_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | |
| | | 1 | Pull-up transistor ON | |
| $PU0_2$ | Port $P4_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | |
| | | 1 | Pull-up transistor ON | |
| $PU0_1$ | Port $P4_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | |
| | | 1 | Pull-up transistor ON | |
| $PU0_0$ | Port $P4_0$ and $P0_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | |
| | | 1 | Pull-up transistor ON | |

| Clock control register MR | | at reset : $1000_2$ | at RAM back-up : state retained | R/W |
|---|---|---|---|---|
| $MR_3$ | System clock selection bit | 0 | $f(X_{IN})$ | |
| | | 1 | $f(X_{IN})/4$ | |
| $MR_2$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $MR_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $MR_0$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |

| 8-bit general purpose register PU0 | at reset : $00_{16}$ | at RAM back-up : state retained | R/W |
|---|---|---|---|
| 8-bit general purpose register.<br>8-bit data can be transferred between this register PU0 and registers A and B with the TSIAB instruction and TABSI instruction. | | | |

Note: "R" represents read enabled, and "W" represents write enabled.

## ABSOLUTE MAXIMUM RATINGS

| Symbol | Parameter | Conditions | Ratings | Unit |
|---|---|---|---|---|
| $V_{DD}$ | Supply voltage | | –0.3 to 7.0 | V |
| $V_I$ | Input voltage<br>P0, P1, P2, P3, P4, $\overline{RESET}$, $X_{IN}$, VDCE | | –0.3 to $V_{DD}$+0.3 | V |
| $V_O$ | Output voltage P0, P1, P3, D | Output transistors in cut-off state | –0.3 to $V_{DD}$+0.3 | V |
| $V_O$ | Output voltage CARR, $X_{OUT}$ | | –0.3 to $V_{DD}$+0.3 | V |
| Pd | Power dissipation | | 300 | mW |
| Topr | Operating temperature range | | –20 to 70 | °C |
| Tstg | Storage temperature range | | –40 to 125 | °C |

## RECOMMENDED OPERATING CONDITIONS1

(Mask ROM version:Ta = –20 °C to 70 °C, $V_{DD}$ = 2.0 V to 5.5 V, unless otherwise noted)
(One Time PROM version:Ta = –20 °C to 70 °C, $V_{DD}$ = 2.5 V to 5.5 V, unless otherwise noted)

| Symbol | Parameter | | Conditions | Limits | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | Min. | Typ. | Max. | |
| $V_{DD}$ | Supply voltage | Mask ROM version<br>System clock<br>=f($X_{IN}$)/4 | f($X_{IN}$) ≤ 4.2 MHz<br>Ceramic resonator | 2.0 | | 5.5 | V |
| | | Mask ROM version<br>System clock<br>=f($X_{IN}$) | f($X_{IN}$) ≤ 2.0 MHz<br>Ceramic resonator | 4.5 | | 5.5 | |
| | | | f($X_{IN}$) ≤ 1.0 MHz<br>Ceramic resonator | 2.0 | | 5.5 | |
| | | One Time PROM version<br>System clock<br>=f($X_{IN}$)/4 | f($X_{IN}$) ≤ 4.2 MHz<br>Ceramic resonator | 2.5 | | 5.5 | |
| | | One Time PROM version<br>System clock<br>=f($X_{IN}$) | f($X_{IN}$) ≤ 2.0 MHz<br>Ceramic resonator | 4.5 | | 5.5 | |
| | | | f($X_{IN}$) ≤ 1.0 MHz<br>Ceramic resonator | 2.5 | | 5.5 | |
| $V_{RAM}$ | RAM back-up voltage | Mask ROM version | RAM back-up | 1.8 | | 5.5 | V |
| | | One Time PROM version | | 2.0 | | 5.5 | V |
| $V_{SS}$ | Supply voltage | | | | 0 | | V |
| f($X_{IN}$) | Oscillation frequency (at ceramic resonance) | Mask ROM version<br>System clock<br>=f($X_{IN}$)/4 | $V_{DD}$=2.0 V to 5.5V | | | 4.2 | MHz |
| | | Mask ROM version<br>System clock<br>=f($X_{IN}$) | $V_{DD}$=4.5 V to 5.5V | | | 2.0 | |
| | | | $V_{DD}$=2.0 V to 5.5V | | | 1.0 | |
| | | One Time PROM version<br>System clock<br>=f($X_{IN}$)/4 | $V_{DD}$=2.5 V to 5.5V | | | 4.2 | |
| | | One Time PROM version<br>System clock<br>=f($X_{IN}$) | $V_{DD}$=4.5 V to 5.5V | | | 2.0 | |
| | | | $V_{DD}$=2.5 V to 5.5V | | | 1.0 | |

## RECOMMENDED OPERATING CONDITIONS 2

(Mask ROM version:Ta = –20 °C to 70 °C, $V_{DD}$ = 2.0 V to 5.5 V, unless otherwise noted)
(One Time PROM version:Ta = –20 °C to 70 °C, $V_{DD}$ = 2.5 V to 5.5 V, unless otherwise noted)

| Symbol | Parameter | Conditions | Limits | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $V_{IH}$ | "H" level input voltage P0, P1, P2, P3, P4, VDCE | | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| $V_{IH}$ | "H" level input voltage $X_{IN}$ | | 0.7$V_{DD}$ | | $V_{DD}$ | V |
| $V_{IH}$ | "H" level input voltage $\overline{RESET}$ | | 0.85$V_{DD}$ | | $V_{DD}$ | V |
| $V_{IH}$ | "H" level input voltage INT | | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| $V_{IL}$ | "L" level input voltage P0, P1, P2, P3, P4, VDCE | | 0 | | 0.3$V_{DD}$ | V |
| $V_{IL}$ | "L" level input voltage $X_{IN}$ | | 0 | | 0.3$V_{DD}$ | V |
| $V_{IL}$ | "L" level input voltage $\overline{RESET}$ | | 0 | | 0.3$V_{DD}$ | V |
| $V_{IL}$ | "L" level input voltage INT | | 0 | | 0.2$V_{DD}$ | V |
| $I_{OL}$(peak) | "L" level peak output current P0, P1, $D_0$–$D_9$, CARR | $V_{DD}$=5.0 V | | | 10 | mA |
| | | $V_{DD}$=3.0 V | | | 4 | |
| $I_{OL}$(peak) | "L" level peak output current P3 | $V_{DD}$=5.0 V | | | 30 | mA |
| | | $V_{DD}$=3.0 V | | | 24 | |
| $I_{OL}$(avg) | "L" level average output current P0, P1, $D_0$–$D_9$, CARR (Note) | $V_{DD}$=5.0 V | | | 5 | mA |
| | | $V_{DD}$=3.0 V | | | 2 | |
| $I_{OL}$(avg) | "L" level average output current P3 (Note) | $V_{DD}$=5.0 V | | | 15 | mA |
| | | $V_{DD}$=3.0 V | | | 12 | |
| $I_{OH}$(peak) | "H" level peak output current CARR | $V_{DD}$=5.0 V | | | –30 | mA |
| | | $V_{DD}$=3.0 V | | | –15 | |
| $I_{OH}$(avg) | "H" level average output current CARR (Note) | $V_{DD}$=5.0 V | | | –15 | mA |
| | | $V_{DD}$=3.0 V | | | –7 | |
| Σ $I_{OL}$ | "L" total current P0, P1, P3 | | | | 30 | mA |
| Σ $I_{OL}$ | "L" total current D | | | | 20 | mA |
| $T_{PON}$ | Power reset circuit valid power rising time | Mask ROM version $V_{DD}$ = 0 to 2.0 V | | | 100 | μs |
| | | One Time PROM version $V_{DD}$ = 0 to 2.5 V | | | | |

Note: The average output current is the average current value at the 100 ms interval.

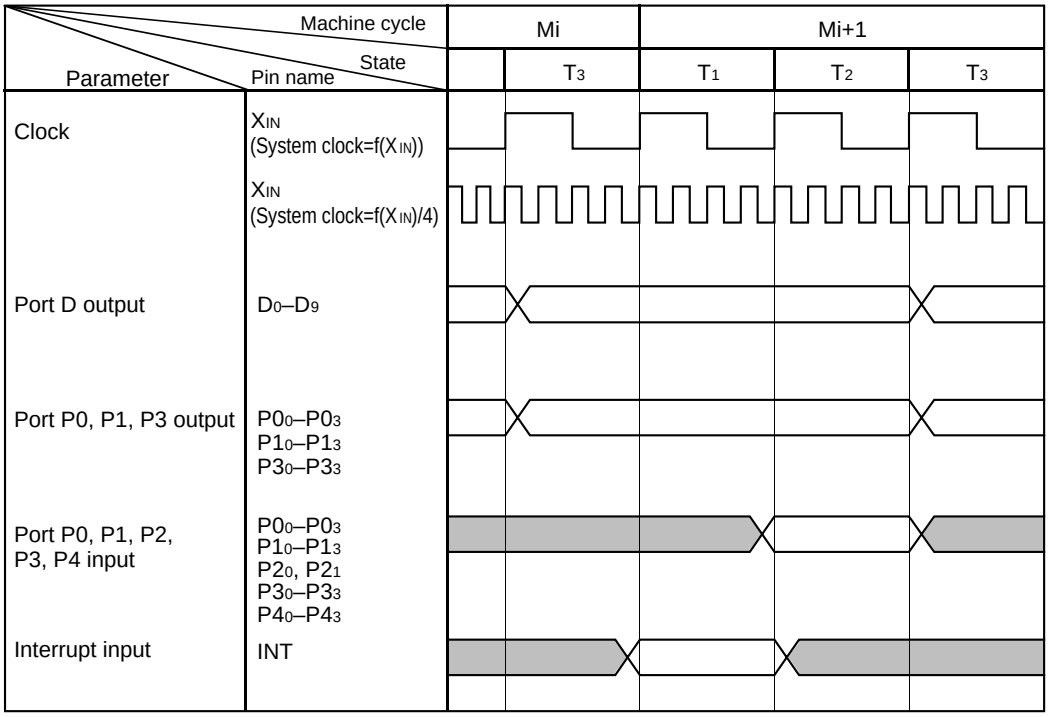
## ELECTRICAL CHARACTERISTICS

(Mask ROM version:Ta = –20 °C to 70 °C, $V_{DD}$ = 2.0 V to 5.5 V, unless otherwise noted)
(One Time PROM version:Ta = –20 °C to 70 °C, $V_{DD}$ = 2.5 V to 5.5 V, unless otherwise noted)

| Symbol | Parameter | | Test conditions | | Limits | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | | | | Min. | Typ. | Max. | |
| $V_{OL}$ | "L" level output voltage P0, P1, $D_0$–$D_9$, CARR, $\overline{RESET}$ | | $I_{OL}$ = 5 mA | $V_{DD}$ = 5.0 V | | | 0.9 | V |
| | | | $I_{OL}$ = 2 mA | $V_{DD}$ = 3.0 V | | | 0.9 | |
| $V_{OL}$ | "L" level output voltage P3 | | $I_{OL}$ = 15 mA | $V_{DD}$ = 5.0 V | | | 1.5 | V |
| | | | $I_{OL}$ = 12 mA | $V_{DD}$ = 3.0 V | | | 1.5 | |
| $V_{OH}$ | "H" level output voltage CARR | | $I_{OH}$ = 15 mA | $V_{DD}$ = 5.0 V | 2.4 | | | V |
| | | | $I_{OH}$ = –7 mA | $V_{DD}$ = 3.0 V | 1.0 | | | |
| $I_{IH}$ | "H" level input current P0, P1, P2, P3, P4, $\overline{RESET}$, VDCE | | $V_I$ = $V_{DD}$ (Note) | | | | 1 | μA |
| $I_{IL}$ | "L" level input current P2, P3, P4, $\overline{RESET}$, VDCE | | $V_I$ = 0 V (Note) | | –1 | | | μA |
| $I_{OZ}$ | Output current at off-state $D_0$–$D_9$ | | $V_O$ = $V_{DD}$ | | | | 1 | μA |
| $I_{DD}$ | Supply current | at CPU operating mode | $V_{DD}$ = 5.0 V, f($X_{IN}$) = 4.2 MHz<br>System clock = f($X_{IN}$)/4 | | | 1.3 | 2.6 | mA |
| | | | $V_{DD}$ = 5.0 V<br>System clock = f($X_{IN}$) | f($X_{IN}$) = 2 MHz | | 1.9 | 3.8 | |
| | | | | f($X_{IN}$) = 1 MHz | | 1.3 | 2.6 | |
| | | | $V_{DD}$ = 3.0 V, f($X_{IN}$) = 4.2 MHz<br>System clock = f($X_{IN}$)/4 | | | 0.6 | 1.2 | |
| | | | $V_{DD}$ = 3.0 V<br>System clock = f($X_{IN}$) | f($X_{IN}$) = 1 MHz | | 0.5 | 1.0 | |
| | | | | f($X_{IN}$) = 500 kHz | | 0.4 | 0.8 | |
| | | at RAM back-up mode | f($X_{IN}$) = stop, typical value at Ta = 25 °C | | | 0.1 | 10 | μA |
| $R_{PH}$ | Pull-up resistor value | P0, P1, P4 | $V_{DD}$ = 5.0 V, $V_I$ = 0 V | | 20 | 50 | 125 | kΩ |
| | | | $V_{DD}$ = 3.0 V, $V_I$ = 0 V | | 40 | 100 | 250 | |
| | | $\overline{RESET}$ | $V_{DD}$ = 5.0 V, $V_I$ = 0 V | | 12 | 30 | 70 | kΩ |
| | | | $V_{DD}$ = 3.0 V, $V_I$ = 0 V | | 25 | 60 | 130 | |
| $V_{T+} – V_{T–}$ | Hysteresis | INT | $V_{DD}$ = 5.0 V | | | 0.5 | | V |
| | | | $V_{DD}$ = 3.0 V | | | 0.4 | | |
| | | $\overline{RESET}$ | $V_{DD}$ = 5.0 V | | | 1.5 | | V |
| | | | $V_{DD}$ = 3.0 V | | | 0.6 | | |

Note: In this case, the pull-up transistor of port P4 is turned off by software.

## BASIC TIMING DIAGRAM



## BUILT-IN PROM VERSION

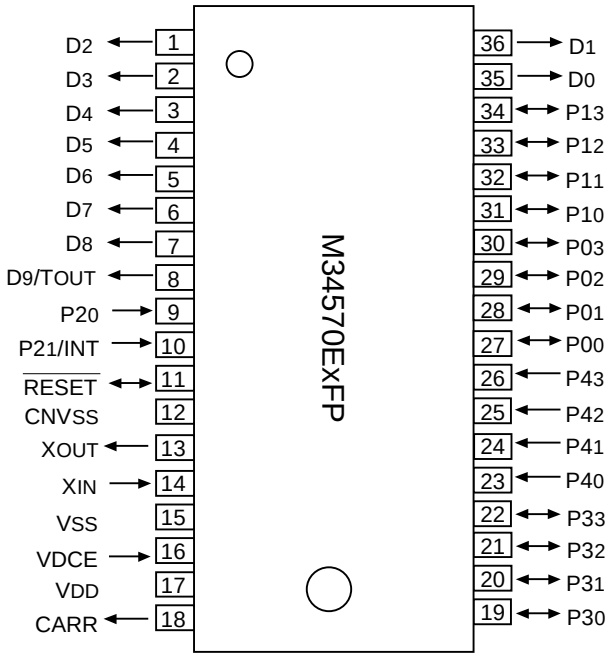
In addition to the mask ROM version, the 4570 Group has the programmable ROM version software compatible with mask ROM. The One Time PROM version has PROM which can only be written to and not be erased.
The built-in PROM version has functions similar to those of the mask ROM version, but it has a PROM mode that enables writing to built-in PROM.

Table 16 shows the product of built-in PROM version. Figure 35 shows the pin configurations of built-in PROM version. The One Time PROM version has pin-compatibility with the mask ROM version.

**Table 16 Product of built-in PROM version**

| Product | PROM size (× 10 bits) | RAM size (× 4 bits) | Package | ROM type |
|---|---|---|---|---|
| M34570E8FP | 8192 words | 128 words | 36P2R-A | One Time PROM |
| M34570EDFP | 16384 words | 128 words | 36P2R-A | |

## PIN CONFIGURATION (TOP VIEW)




Outline 36P2R-A

**Fig. 35 Pin configuration of built-in PROM version**

**(1) PROM mode**

The built-in PROM version has a PROM mode in addition to a normal operation mode. The PROM mode is used to write to and read from the built-in PROM.

In the PROM mode, the programming adapter can be used with a general-purpose PROM programmer to write to or read from the built-in PROM as if it were M5M27C256K. Programming adapter is listed in Table 17. Contact addresses at the end of this book for the appropriate PROM programmer.

- Writing and reading of built-in PROM
  Programming voltage is 12.5 V. Write the program in the PROM of the built-in PROM version as shown in Figure 36.

**(2) Notes on handling**

① A high-voltage is used for writing. Take care that overvoltage is not applied. Take care especially at turning on the power.

② For the One Time PROM version Mitsubishi Electric corp. does not perform PROM writing test and screening in the assembly process and following processes. In order to improve reliability after writing, performing writing and test according to the flow shown in Figure 37 before using is recommended.

**Table 17 Programming adapter**

| Microcomputer | Programming adapter |
|---|---|
| M34570E8FP, M34570EDFP | PCA7425 |

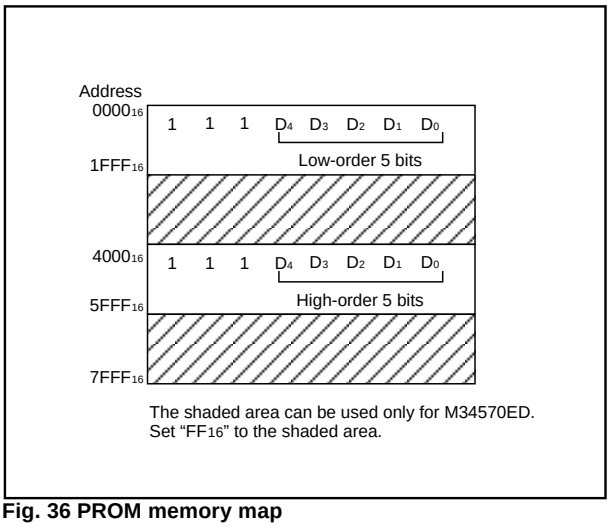



**Fig. 36 PROM memory map**

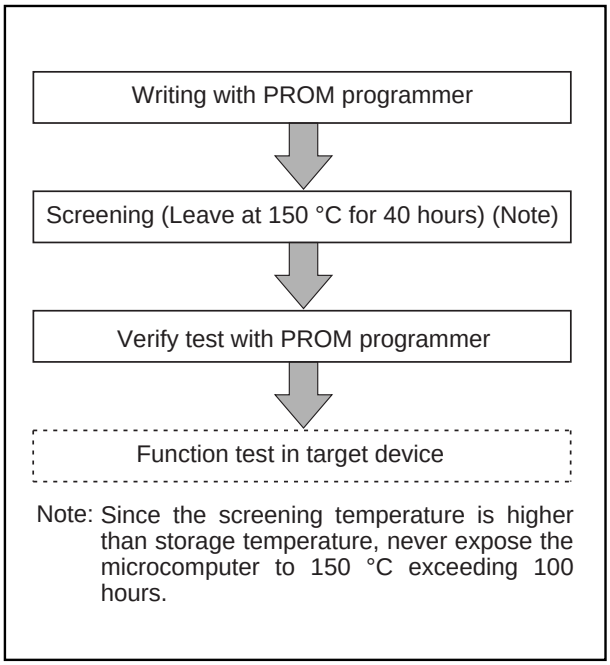



**Fig. 37 Flow of writing and test of the product shipped in blank**

GZZ-SH55-08B <91A0>

| Mask ROM number | |
|---|---|

**4500 SERIES MASK ROM ORDER CONFIRMATION FORM**
**SINGLE-CHIP MICROCOMPUTER M34570M4-XXXFP**
**MITSUBISHI ELECTRIC**

| Receipt | Date: | |
|---|---|---|
| | Section head signature | Supervisor signature |
| | | |

Please fill in all items marked ∗.

| | | | |
|---|---|---|---|
| ∗ Customer | Company name | TEL ( ) | |
| | Date issued | Date: | |

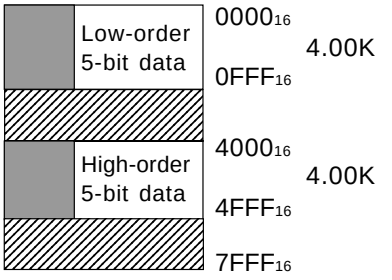
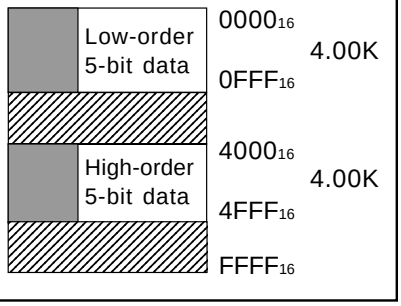
| Issuance signature | Responsible officer | Supervisor |
|---|---|---|
| | | |

∗ 1. Confirmation
Three sets of EPROMs are required for each pattern if this order is performed by EPROMs.
One floppy disk is required for each pattern if this order is performed by floppy disk.

☐ Ordering by the EPROMs
Specify the type of EPROMs submitted (check in the approximate box).
If at least two of the three sets of EPROMs submitted contain the identical data, we will produce masks based on this data. We shall assume the responsibility for errors only if the mask ROM data on the products we produce differ from this data. Thus, the customer must be especially careful in verifying the data contained in the EPROMs submitted.

Checksum code for entire EPROM area ☐☐☐☐ (hexadecimal notation)

EPROM Type:

| ☐ 27C256 | ☐ 27C512 |
|---|---|
|  |  |



Set "FF$_{16}$" in the shaded area.
Set "111$_2$" in the area ▭ of low-order and high-order 5-bit data.

GZZ-SH55-08B <91A0>

| Mask ROM number | |
|---|---|

**4500 SERIES MASK ROM ORDER CONFIRMATION FORM**
**SINGLE-CHIP MICROCOMPUTER M34570M4-XXXFP**
**MITSUBISHI ELECTRIC**

☐ Ordering by floppy disk

We will produce masks based on the mask files generated by the mask file generating utility. We shall assume the responsibility for errors only if the mask ROM data on the products we produce differs from this mask file. Thus, extreme care must be taken to verify the mask file in the submitted floppy disk.
The submitted floppy disk must be-3.5 inch 2HD type and DOS/V format. And the number of the mask files must be 1 in one floppy disk.

File code ☐☐☐☐☐☐☐☐ (hexadecimal notation)

Mask file name ☐☐☐☐☐☐☐☐ .MSK (equal or less than eight characters)

✻ 2. Mark Specification

Mark specification must be submitted using the correct form for the type of package being ordered. Fill out the approximate Mark Specification Form (36P2R-A for M34570M4-XXXFP) and attach to the Mask ROM Order Confirmation Form.

✻ 3. Comments

GZZ-SH55-09B <91A0>

| Mask ROM number | |
|---|---|

**4500 SERIES MASK ROM ORDER CONFIRMATION FORM**
**SINGLE-CHIP MICROCOMPUTER M34570M8-XXXFP**
**MITSUBISHI ELECTRIC**

| Receipt | Date: | |
|---|---|---|
| | Section head signature | Supervisor signature |
| | | |

Please fill in all items marked ∗.

| | | | |
|---|---|---|---|
| ∗ Customer | Company name | TEL ( ) | |
| | Date issued | Date: | |

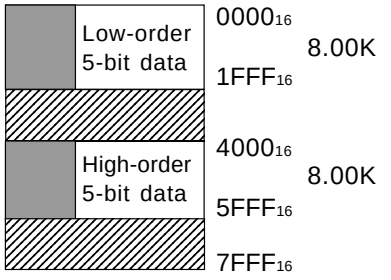
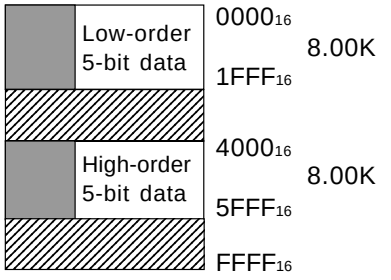
| Issuance signature | Responsible officer | Supervisor |
|---|---|---|
| | | |

∗ 1. Confirmation
Three sets of EPROMs are required for each pattern if this order is performed by EPROMs.
One floppy disk is required for each pattern if this order is performed by floppy disk.

☐ Ordering by the EPROMs
Specify the type of EPROMs submitted (check in the approximate box).
If at least two of the three sets of EPROMs submitted contain the identical data, we will produce masks based on this data. We shall assume the responsibility for errors only if the mask ROM data on the products we produce differ from this data. Thus, the customer must be especially careful in verifying the data contained in the EPROMs submitted.

Checksum code for entire EPROM area ☐☐☐☐ (hexadecimal notation)

EPROM Type:

| ☐ 27C256 | ☐ 27C512 |
|---|---|
|  |  |



Set “FF16” in the shaded area.
Set “1112” in the area ▭ of low-order and high-order 5-bit data.

GZZ-SH55-09B <91A0>

| Mask ROM number | |
|---|---|

**4500 SERIES MASK ROM ORDER CONFIRMATION FORM**
**SINGLE-CHIP MICROCOMPUTER M34570M8-XXXFP**
**MITSUBISHI ELECTRIC**

☐ Ordering by floppy disk

We will produce masks based on the mask files generated by the mask file generating utility. We shall assume the responsibility for errors only if the mask ROM data on the products we produce differs from this mask file. Thus, extreme care must be taken to verify the mask file in the submitted floppy disk.
The submitted floppy disk must be-3.5 inch 2HD type and DOS/V format. And the number of the mask files must be 1 in one floppy disk.

File code | | | | | | | | | (hexadecimal notation)

Mask file name | | | | | | | | | .MSK (equal or less than eight characters)

∗ 2. Mark Specification

Mark specification must be submitted using the correct form for the type of package being ordered. Fill out the approximate Mark Specification Form (36P2R-A for M34570M8-XXXFP) and attach to the Mask ROM Order Confirmation Form.

∗ 3. Comments

GZZ-SH55-10B <91A0>

| Mask ROM number | |
|---|---|

**4500 SERIES MASK ROM ORDER CONFIRMATION FORM**
**SINGLE-CHIP MICROCOMPUTER M34570MD-XXXFP**
**MITSUBISHI ELECTRIC**

| Receipt | Date: | |
|---|---|---|
| | Section head signature | Supervisor signature |
| | | |

Please fill in all items marked ∗.

| | | | |
|---|---|---|---|
| ∗ Customer | Company name | TEL ( ) | |
| | Date issued | Date: | |

| Issuance signature | Responsible officer | Supervisor |
|---|---|---|
| | | |

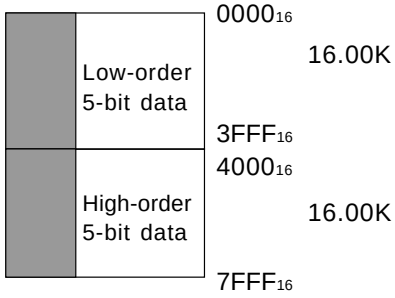
∗ 1. Confirmation
Three sets of EPROMs are required for each pattern if this order is performed by EPROMs.
One floppy disk is required for each pattern if this order is performed by floppy disk.

☐ Ordering by the EPROMs
Specify the type of EPROMs submitted (check in the approximate box).
If at least two of the three sets of EPROMs submitted contain the identical data, we will produce masks based on this data. We shall assume the responsibility for errors only if the mask ROM data on the products we produce differ from this data. Thus, the customer must be especially careful in verifying the data contained in the EPROMs submitted.

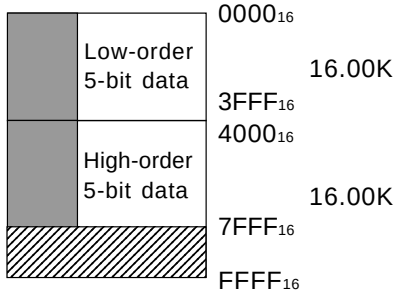
Checksum code for entire EPROM area ☐☐☐☐ (hexadecimal notation)

EPROM Type:

☐ 27C256




☐ 27C512




Set "$FF_{16}$" in the shaded area.
Set "$111_2$" in the area ▒ of low-order and high-order 5-bit data.

GZZ-SH55-10B <91A0>

| Mask ROM number | |
|---|---|

**4500 SERIES MASK ROM ORDER CONFIRMATION FORM**
**SINGLE-CHIP MICROCOMPUTER M34570MD-XXXFP**
**MITSUBISHI ELECTRIC**

☐ Ordering by floppy disk

We will produce masks based on the mask files generated by the mask file generating utility. We shall assume the responsibility for errors only if the mask ROM data on the products we produce differs from this mask file. Thus, extreme care must be taken to verify the mask file in the submitted floppy disk.
The submitted floppy disk must be-3.5 inch 2HD type and DOS/V format. And the number of the mask files must be 1 in one floppy disk.

File code ☐☐☐☐☐☐☐☐ (hexadecimal notation)

Mask file name ☐☐☐☐☐☐☐☐ .MSK (equal or less than eight characters)

✻ 2. Mark Specification

Mark specification must be submitted using the correct form for the type of package being ordered. Fill out the approximate Mark Specification Form (36P2R-A for M34570MD-XXXFP) and attach to the Mask ROM Order Confirmation Form.

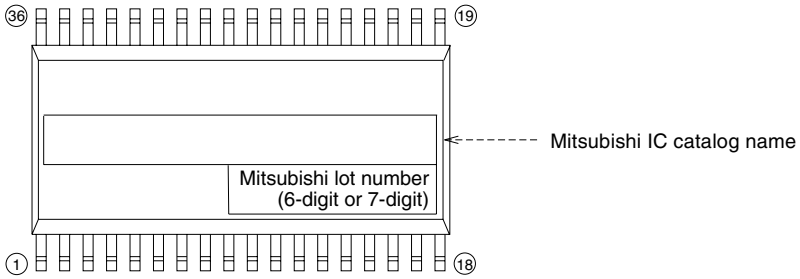
✻ 3. Comments

## MARK SPECIFICATION FORM
## 36P2R-A (36-PIN SHRINK SOP) MARK SPECIFICATION FORM

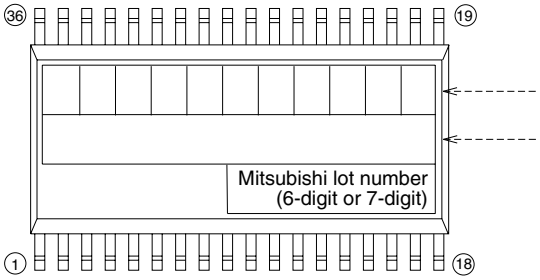
Mitsubishi IC catalog name

Please choose one of the marking types below (A, B, C), and enter the Mitsubishi catalog name and the special mark (if needed).

A. Standard Mitsubishi Mark

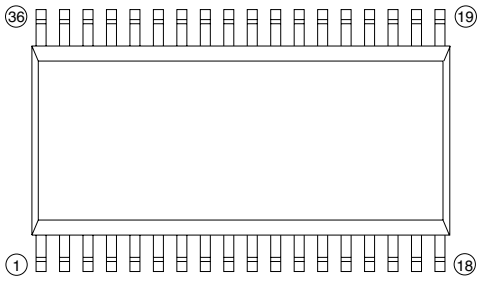



B. Customer's Parts Number + Mitsubishi catalog name




Customer's Parts Number
Note : The fonts and size of characters are standard Mitsubishi type.
Mitsubishi IC catalog name
Note1 : The mark field should be written right aligned.
2 : The fonts and size of characters are standard Mitsubishi type.
3 : Customer's Parts Number can be up to 11 characters : Only 0 ~ 9, A ~ Z, +, –, /, (, ), &, ©, . (periods), , (commas) are usable.

C. Special Mark Required




Note1 : If the Special Mark is to be Printed, indicate the desired layout of the mark in the left figure. The layout will be duplicated as close as possible.
Mitsubishi lot number (6-digit or 7-digit) and Mask ROM number (3-digit) are always marked.
2 : If the customer's trade mark logo must be used in the Special Mark, check the box below.
Please submit a clean original of the logo.
For the new special character fonts a clean font original (ideally logo drawing) must be submitted.

Special logo required ☐

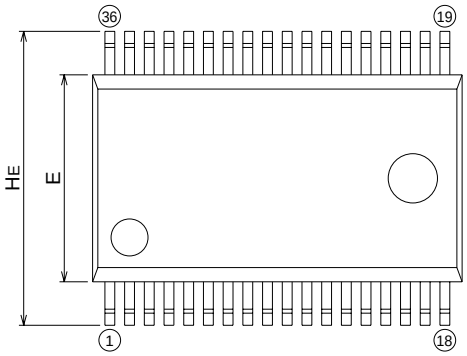
3 : The standard Mitsubishi font is used for all characters except for a logo.

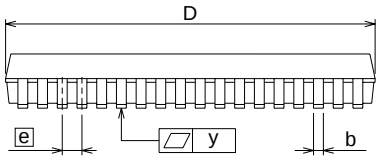
## PACKAGE OUTLINE

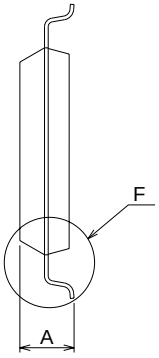
### 36P2R-A

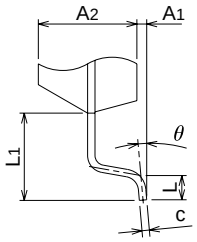
**Plastic 36pin 450mil SSOP**

| EIAJ Package Code | JEDEC Code | Weight(g) | Lead Material |
|---|---|---|---|
| SSOP36-P-450-0.80 | – | 0.53 | Alloy 42 |

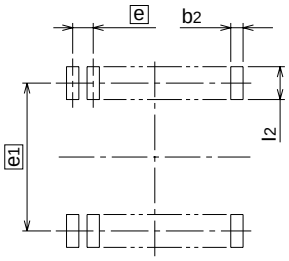



Detail F




Recommended Mount Pad

| Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | – | – | 2.4 |
| $A_1$ | 0.05 | – | – |
| $A_2$ | – | 2.0 | – |
| b | 0.35 | 0.4 | 0.5 |
| c | 0.13 | 0.15 | 0.2 |
| D | 14.8 | 15.0 | 15.2 |
| E | 8.2 | 8.4 | 8.6 |
| e | – | 0.8 | – |
| $H_E$ | 11.63 | 11.93 | 12.23 |
| L | 0.3 | 0.5 | 0.7 |
| $L_1$ | – | 1.765 | – |
| y | – | – | 0.15 |
| $\theta$ | 0° | – | 10° |
| $b_2$ | – | 0.5 | – |
| e1 | – | 11.43 | – |
| $l_2$ | 1.27 | – | – |

MITSUBISHI MICROCOMPUTERS

# 4570 Group

SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

Renesas Technology Corp.
Nippon Bldg.,6-2,Otemachi 2-chome,Chiyoda-ku,Tokyo,100-0004 Japan


New publication, effective April. 1999.
Specifications subject to change without notice.

RENESAS
Renesas Technology Corp.

| REVISION DESCRIPTION LIST | 4570 GROUP DATA SHEET |
|---|---|

| Rev. No. | Revision Description | Rev. date |
|---|---|---|
| 1.0 | First Edition | 971022 |
| 2.0 | Main revision points are described below.<br>•M34570MD-XXXFP and M34570EDFP (ROM expansion products [size: 16K 5 10 bits] ) added.<br>• SBK and RBK instructions added and TABP p instruction function is expanded.<br>(TABP p instruction: When this instruction is executed after executing the SBK instruction, pages 64 to 127 are specified. When this instruction is executed after executing the RBK instruction, pages 0 to 63 are specified. When this instruction is executed after system is released from reset and returned from the RAM back-up mode, pages 0 to 63 are specified.)<br>• BL, BML, BLA and BMLA instructions revised. Referred pages are expanded to pages 0 to 127 (p6 can be used for page specification.) | 990331 |
| | | |