# 4508 Group

SINGLE-CHIP 4-BIT CMOS MICROCOMPUTER

REJ03B0148-0103
Rev.1.03
2009.07.27

## DESCRIPTION

The 4508 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 two 8-bit timers (each timer has two reload registers), interrupts, 10-bit A/D converter, Serial interface and oscillation circuit switch function.

## FEATURES

- Minimum instruction execution time ..... 0.5 $\mu$s
  (at 6 MHz oscillation frequency, in through-mode)
- Supply voltage ..... 1.8 V to 5.5 V
  (It depends on operation source clock, oscillation frequency and operating mode.)
- Timers
  Timer 1 ..... 8-bit timer with two reload registers
  Timer 2 ..... 8-bit timer with two reload registers
- Interrupt ..... 5 sources
- Key-on wakeup function pins ..... 12
- Input/Output port ..... 14
- A/D converter
  10-bit successive comparison method ..... 4 channel
- Serial intereface ..... 8-bit × 1
- Voltage drop detection circuit (only for H version)
  Reset occurrence ..... Typ. 2.6 V (Ta = 25 °C)
  Reset release ..... Typ. 2.7 V (Ta = 25 °C)
- Power-on reset circuit (only for H version)
- Watchdog timer
- Clock generating circuit (on-chip oscillator/ceramic resonator/RC oscillation)
- LED drive directly enabled (port D)

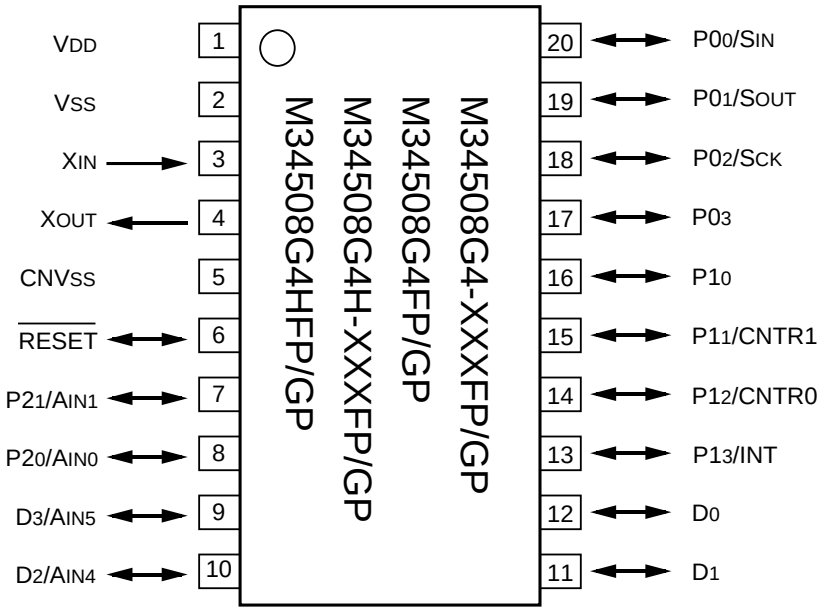
## APPLICATION

Electrical household appliance, consumer electronic products, office automation equipment, etc.

| Part number | ROM (PROM) size (× 10 bits) | RAM size (× 4 bits) | Package | ROM type |
|---|---|---|---|---|
| M34508G4FP (**Note**) | 4096 words | 256 words | PRSP0020DA-A | QzROM |
| M34508G4-XXXFP | 4096 words | 256 words | PRSP0020DA-A | QzROM |
| M34508G4HFP (**Note**) | 4096 words | 256 words | PRSP0020DA-A | QzROM |
| M34508G4H-XXXFP | 4096 words | 256 words | PRSP0020DA-A | QzROM |
| M34508G4GP (**Note**) | 4096 words | 256 words | PLSP0020JB-A | QzROM |
| M34508G4-XXXGP | 4096 words | 256 words | PLSP0020JB-A | QzROM |
| M34508G4HGP (**Note**) | 4096 words | 256 words | PLSP0020JB-A | QzROM |
| M34508G4H-XXXGP | 4096 words | 256 words | PLSP0020JB-A | QzROM |

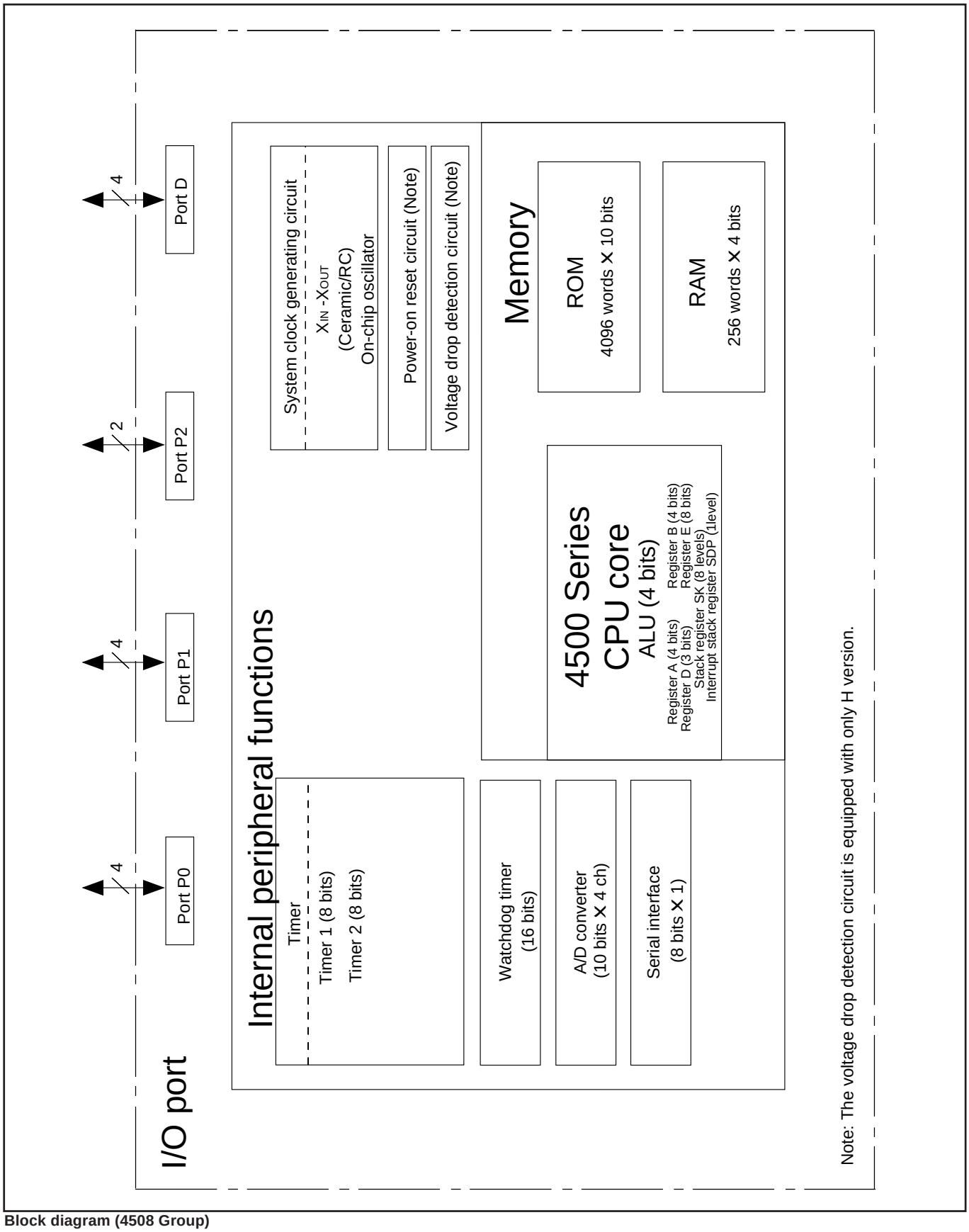
**Note:** Shipped in blank

## PIN CONFIGURATION




Outline FP: PRSP0020DA-A (20P2N-A)
GP: PLSP0020JB-A (20P2F-A)

**Pin configuration (top view) (4508 Group)**




**Block diagram (4508 Group)**

## PERFORMANCE OVERVIEW

| Parameter | | | Function |
|---|---|---|---|
| Number of basic instructions | | M34508G4 | 131 |
| | | M34508G4H | 132 |
| Minimum instruction execution time | | | 0.5 μs (at 6 MHz oscillation frequency, in through mode) |
| Memory sizes | ROM | | 4096 words × 10 bits |
| | RAM | | 256 words × 4 bits |
| Input/Output ports | $D_0$–$D_3$ | I/O | Four independent I/O ports.<br>Input is examined by skip decision.<br>Ports $D_2$ and $D_3$ are equipped with a pull-up function and a key-on wakeup function. Both functions can be switched by software.<br>Ports $D_2$ and $D_3$ are also used as $A_{IN4}$, and $A_{IN5}$, respectively. |
| | $P0_0$–$P0_3$ | I/O | 4-bit I/O port; each pin is equipped with a pull-up function and a key-on wakeup function. Both functions and output structure can be switched by software.<br>Ports $P0_0$, $P0_1$ and $P0_2$ are also used as $S_{IN}$, $S_{OUT}$ and $S_{CK}$, respectively. |
| | $P1_0$–$P1_3$ | I/O | 4-bit I/O port; each pin is equipped with a pull-up function and a key-on wakeup function. Both functions and output structure can be switched by software.<br>Ports $P1_1$, $P1_2$ and $P1_3$ are also used as CNTR1, CNTR0 and INT, respectively. |
| | $P2_0$, $P2_1$ | I/O | 2-bit I/O port; each pin is equipped with a pull-up function and a key-on wakeup function. Both functions and output structure can be switched by software.<br>Ports $P2_0$ and $P2_1$ are also used as $A_{IN0}$ and $A_{IN1}$, respectively. |
| | CNTR0, CNTR1 | Timer I/O | Two independent I/O; CNTR1 and CNTR0 pins are also used as ports $P1_1$ and $P1_2$, respectively. |
| | INT | Interrupt input | 1-bit input; INT pin is also used as port $P1_3$. |
| | $S_{IN}$, $S_{OUT}$, $S_{CK}$ | Serial interface input/output | Three independent I/O;<br>$S_{IN}$, $S_{OUT}$, and $S_{CK}$ are also used as ports $P0_0$, $P0_1$, and $P0_2$, respectively. |
| | $A_{IN0}$, $A_{IN1}$, $A_{IN4}$, $A_{IN5}$ | Analog input | Four independent input; $A_{IN0}$, $A_{IN1}$, $A_{IN4}$, $A_{IN5}$ are also used as $P2_0$, $P2_1$, $D_2$ and $D_3$, respectively. |
| Timers | Timer 1 | | 8-bit programmable timer/event counter with two reload registers and PWM output function. |
| | Timer 2 | | 8-bit programmable timer/event counter with two reload registers and PWM output function. |
| | Watchdog timer function | | 16-bit timer (fixed dividing frequency) (for watchdog) |
| A/D converter | | | 10-bit wide, This is equipped with an 8-bit comparator function. |
| | Analog input | | 4 channel ($A_{IN0}$, $A_{IN1}$, $A_{IN4}$, $A_{IN5}$ pins) |
| Serial interface | | | 8-bit × 1 |
| Voltage drop detection circuit (Note) | Reset occurrence | | Typ. 2.6 V (Ta = 25 °C) |
| | Reset release | | Typ. 2.7 V (Ta = 25 °C) |
| Power-on reset circuit (Note) | | | Built-in type |
| Interrupt | Sources | | 5 (one for external, two for timer, one for A/D, one for Serial interface) |
| | Nesting | | 1 level |
| Subroutine nesting | | | 8 levels |
| Device structure | | | CMOS silicon gate |
| Package | | | FP: 20-pin plastic molded SOP (PRSP0020DA-A) |
| | | | GP: 20-pin plastic molded SSOP (PLSP0020JB-A) |
| Operating temperature range | | | –20 °C to 85 °C |
| Supply voltage | | | 1.8 V to 5.5 V (It depends on operation source clock, oscillation frequency and operating mode.) |
| Power dissipation (typical value) | Active mode | | 2.2 mA (Ta = 25°C, $V_{DD}$ = 5.0 V, f($X_{IN}$) = 6.0 MHz, f(STCK) = f($X_{IN}$)/1) |
| | RAM back-up mode | | 0.1 μA (Ta = 25°C, $V_{DD}$ = 5.0 V, output transistors in the cut-off state) |

Note: These circuits are equipped with only the H version.

## 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. |
| $CNV_{SS}$ | $CNV_{SS}$ | — | Connect $CNV_{SS}$ to $V_{SS}$ and apply "L" (0V) to $CNV_{SS}$ certainly. |
| $\overline{RESET}$ | Reset input/output | I/O | An N-channel open-drain I/O pin for a system reset. When the SRST instruction, watchdog timer, the voltage drop detection circuit (only for H version) or the built-in power-on reset (only for H version) causes the system to be reset, the $\overline{RESET}$ pin outputs "L" level. |
| $X_{IN}$ | System clock input | Input | I/O pins of the main clock generating circuit. When using a ceramic resonator, connect it between pins $X_{IN}$ and $X_{OUT}$. A feedback resistor is built-in between them. When using the RC oscillation, connect a resistor and a capacitor to $X_{IN}$, and leave $X_{OUT}$ pin open. |
| $X_{OUT}$ | System clock output | Output | |
| $D_0$–$D_3$ | I/O port D<br>Input is examined by skip decision. | I/O | Each pin of port D has an independent 1-bit wide I/O function.<br>The output structure can be switched to N-channel open-drain or CMOS by software.<br>For input use, set the latch of the specified bit to "1" and select the N-channel open-drain.<br>Ports $D_2$ and $D_3$ are equipped with a pull-up function and a key-on wakeup function. Both functions can be switched by software.<br>Ports $D_2$ and $D_3$ are also used as $A_{IN4}$ and $A_{IN5}$, respectively. |
| $P0_0$–$P0_3$ | I/O port P0 | I/O | Port P0 serves as a 4-bit I/O port.<br>The output structure can be switched to N-channel open-drain or CMOS by software.<br>For input use, set the latch of the specified bit to "1" and select the N-channel open-drain.<br>Port P0 has a key-on wakeup function and a pull-up function. Both functions can be switched by software.<br>Ports $P0_0$, $P0_1$ and $P0_2$ are also used as $S_{IN}$, $S_{OUT}$ and $S_{CK}$, respectively. |
| $P1_0$–$P1_3$ | I/O port P1 | I/O | Port P1 serves as a 4-bit I/O port.<br>The output structure can be switched to N-channel open-drain or CMOS by software.<br>For input use, set the latch of the specified bit to "1" and select the N-channel open-drain.<br>Port P1 has a key-on wakeup function and a pull-up function. Both functions can be switched by software.<br>Ports $P1_1$, $P1_2$ and $P1_3$ are also used as CNTR1, CNTR0 and INT, respectively. |
| $P2_0$, $P2_1$ | I/O port P2 | I/O | Port P2 serves as a 2-bit I/O port.<br>The output structure can be switched to N-channel open-drain or CMOS by software.<br>For input use, set the latch of the specified bit to "1" and select the N-channel open-drain.<br>Port P2 has a key-on wakeup function and a pull-up function. Both functions can be switched by software.<br>Ports $P2_0$ and $P2_1$ are also used as $A_{IN0}$ and $A_{IN1}$, respectively. |
| CNTR0 | Timer input/output | I/O | CNTR0 pin has the function to input the clock for the timer 2 event counter, and to output the PWM signal generated by timer 1.<br>This pin is also used as port $P1_2$. |
| CNTR1 | Timer input/output | I/O | CNTR1 pin has the function to input the clock for the timer 1 event counter, and to output the PWM signal generated by timer 2.<br>This pin is also used as port $P1_1$. |
| INT | Interrupt input | Input | INT pin accepts external interrupts. It has the key-on wakeup function which can be switched by software.<br>This pin is also used as port $P1_3$. |
| $A_{IN0}$, $A_{IN1}$, $A_{IN4}$, $A_{IN5}$, | Analog input | Input | A/D converter analog input pins.<br>$A_{IN0}$, $A_{IN1}$, $A_{IN4}$, $A_{IN5}$ are also used as ports $P2_0$, $P2_1$, $D_2$ and $D_3$, respectively. |
| $S_{CK}$ | Serial interface clock I/O | I/O | Serial interface data transfer synchronous clock I/O pin. $S_{CK}$ pin is also used as port $P0_2$. |
| $S_{OUT}$ | Serial interface data output | Output | Serial interface data output pin. $S_{OUT}$ pin is also used as port $P0_1$. |
| $S_{IN}$ | Serial interface data input | Input | Serial interface data input pin. $S_{IN}$ pin is also used as port $P0_0$. |

## MULTIFUNCTION

| Pin | Multifunction | Pin | Multifunction | Pin | Multifunction | Pin | Multifunction |
|---|---|---|---|---|---|---|---|
| $P0_0$ | SIN | SIN | $P0_0$ | $P2_0$ | $AIN_0$ | $AIN_0$ | $P2_0$ |
| $P0_1$ | SOUT | SOUT | $P0_1$ | $P2_1$ | $AIN_1$ | $AIN_1$ | $P2_1$ |
| $P0_2$ | SCK | SCK | $P0_2$ | $D_2$ | $AIN_4$ | $AIN_4$ | $D_2$ |
| $P1_1$ | CNTR1 | CNTR1 | $P1_1$ | $D_3$ | $AIN_5$ | $AIN_5$ | $D_3$ |
| $P1_2$ | CNTR0 | CNTR0 | $P1_2$ | | | | |
| $P1_3$ | INT | INT | $P1_3$ | | | | |

Notes 1: Pins except above have just single function.

2: The input/output of $P0_0$ can be used even when SIN is used. Be careful when using inputs of both SIN and $P0_0$ since the input threshold value of SIN pin is different from that of port $P0_0$.

3: The input of $P0_1$ can be used even when SOUT is used.

4: The input of $P0_2$ can be used even when SCK is used. Be careful when using inputs of both SCK and $P0_2$ since the input threshold value of SCK pin is different from that of port $P0_2$.

5: The input of $P1_1$ can be used even when CNTR1 (output) is selected.
The input/output of $P1_1$ can be used even when CNTR1 (input) is selected. Be careful when using inputs of both CNTR1 and $P1_1$ since the input threshold value of CNTR1 pin is different from that of port $P1_1$.

6: The input of $P1_2$ can be used even when CNTR0 (output) is selected.
The input/output of $P1_2$ can be used even when CNTR0 (input) is selected. Be careful when using inputs of both CNTR0 and $P1_2$ since the input threshold value of CNTR0 pin is different from that of port $P1_2$.

7: The input/output of $P1_3$ can be used even when INT is used. Be careful when using inputs of both INT and $P1_3$ since the input threshold value of INT pin is different from that of port $P1_3$.

8: The input/output of $P2_0$, $P2_1$, $D_2$, $D_3$ can be used even when $AIN_0$, $AIN_1$, $AIN_4$, $AIN_5$ are used.

## PORT FUNCTION

| Port | Pin | Input Output | Output structure | I/O unit | Control instructions | Control registers | Remark |
|---|---|---|---|---|---|---|---|
| Port D | $D_0$, $D_1$ | I/O (4) | N-channel open-drain/ CMOS | 1 | SD, RD SZD, CLD | FR3 | Programmable output structure selection function |
| | $D_2$/$AIN_4$ $D_3$/$AIN_5$ | | | | | FR3, PU2 K2 Q1 | Programmable pull-up function Programmable key-on wakeup function Programmable output structure selection function |
| Port P0 | $P0_0$/SIN, $P0_1$/SOUT, $P0_2$/SCK, $P0_3$ | I/O (4) | N-channel open-drain/ CMOS | 4 | OP0A IAP0 | FR0, PU0 K0 J1 | Programmable pull-up function Programmable key-on wakeup function Programmable output structure selection function |
| Port P1 | $P1_0$, $P1_1$/CNTR1, $P1_2$/CNT0, $P1_3$/INT | I/O (4) | N-channel open-drain/ CMOS | 4 | OP1A IAP1 | FR1, PU1 K1, L1, I1 W1, W2 W5, W6 | Programmable pull-up function Programmable key-on wakeup function Programmable output structure selection function |
| Port P2 | $P2_0$/$AIN_0$ $P2_1$/$AIN_1$ | I/O (2) | N-channel open-drain/ CMOS | 2 | OP2A IAP2 | FR2, PU2 Q1 K2 | Programmable pull-up function Programmable key-on wakeup function Programmable output structure selection function |

## DEFINITION OF CLOCK AND CYCLE

● Operation source clock

The operation source clock is the source clock to operate this product. In this product, the following clocks are used.

- Clock (f(XIN)) by the external ceramic resonator
- Clock (f(XIN)) by the external RC oscillation
- Clock (f(XIN)) by the external input
- Clock (f(RING)) of the on-chip oscillator which is the internal oscillator.

● System clock

The system clock is the basic clock for controlling this product. The system clock is selected by the register MR and register RG.

● Instruction clock

The instruction clock is a signal derived by dividing the system clock by 3. The one instruction clock cycle generates the one machine cycle.

● Machine cycle

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

**Table Selection of system clock**

| Register MR, RG | | | | | System clock | Operation mode |
|---|---|---|---|---|---|---|
| $MR_3$ | $MR_2$ | $MR_1$ | $MR_0$ | $RG_0$ | | |
| 1 | 1 | – | 1 | 0 | f(STCK) = f(RING)/8 | Internal frequency divided by 8 mode |
| 1 | 0 | – | 1 | 0 | f(STCK) = f(RING)/4 | Internal frequency divided by 4 mode |
| 0 | 1 | – | 1 | 0 | f(STCK) = f(RING)/2 | Internal frequency divided by 2 mode |
| 0 | 0 | – | 1 | 0 | f(STCK) = f(RING) | Internal frequency through mode |
| 1 | 1 | 0 | 0 | – | f(STCK) = f(XIN)/8 | High-speed frequency divided by 8 mode |
| 1 | 0 | 0 | 0 | – | f(STCK) = f(XIN)/4 | High-speed frequency divided by 4 mode |
| 0 | 1 | 0 | 0 | – | f(STCK) = f(XIN)/2 | High-speed frequency divided by 2 mode |
| 0 | 0 | 0 | 0 | – | f(STCK) = f(XIN) | High-speed through mode |

**Note:** The internal frequency divided by 8 is selected after system is released from reset.

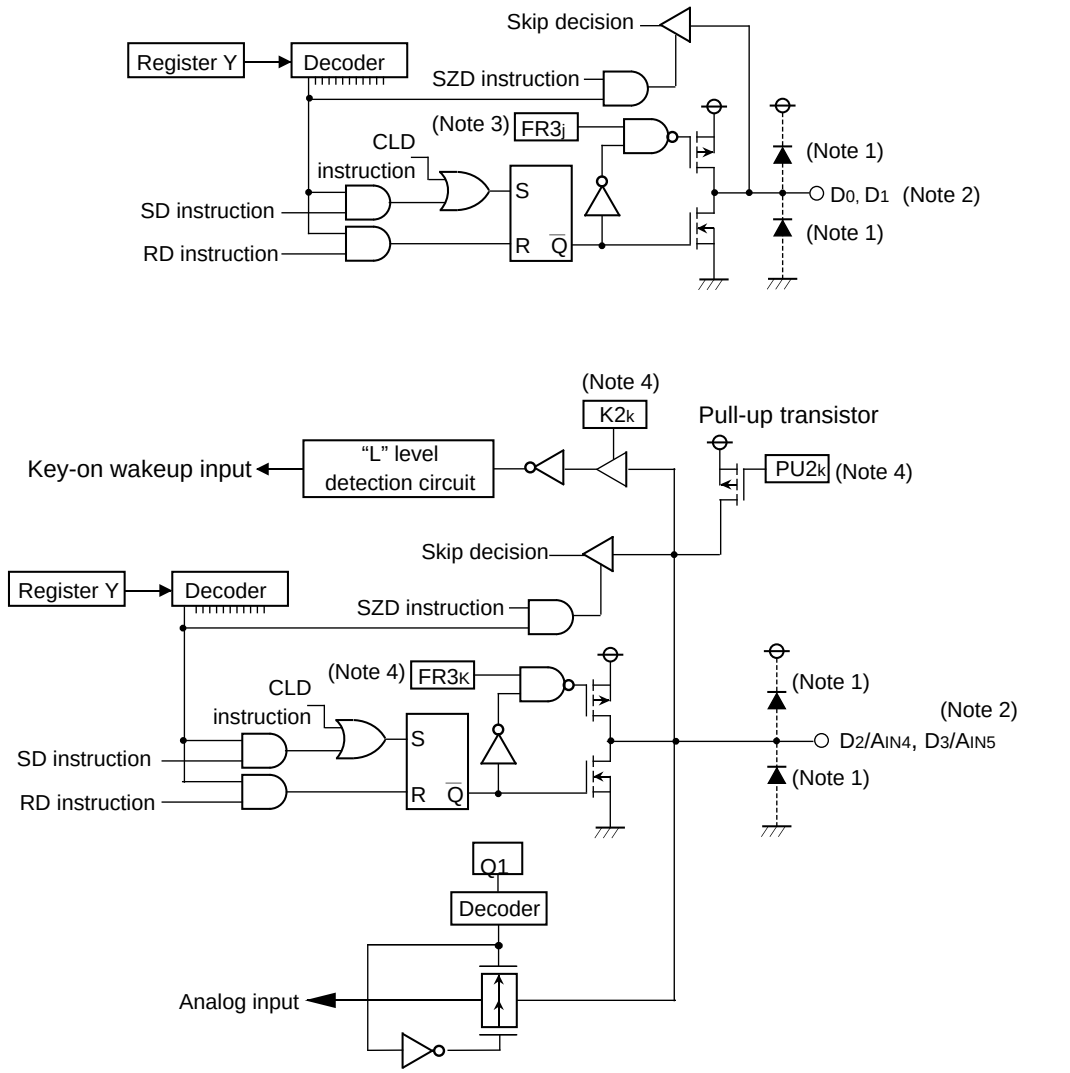
## CONNECTIONS OF UNUSED PINS

| Pin | Connection | Usage condition |
|---|---|---|
| $X_{IN}$ | Connect to $V_{SS}$. | RC oscillation circuit is not selected. (CRCK instruction is not executed.) |
| $X_{OUT}$ | Open. | ———— |
| $D_0$, $D_1$ | Open. | ———— |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR3_0$, $FR3_1$ = "0"). |
| $D_2/A_{IN4}$, $D_3/A_{IN5}$ | Open. | The key-on wakeup function is invalid ($K2_2$, $K2_3$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR3_2$, $FR3_3$ = "0").<br>Pull-up transistor is OFF ($PU2_2$, $PU2_3$ = "0").<br>The key-on wakeup function is invalid ($K2_2$, $K2_3$ = "0"). |
| $P0_0/S_{IN}$ | Open. | $S_{IN}$ pin is not selected ($J1_1$ = "0").<br>The key-on wakeup function is invalid ($K0_0$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR0_0$ = "0").<br>Pull-up transistor is OFF ($PU0_0$ = "0").<br>The key-on wakeup function is invalid ($K0_0$ = "0"). |
| $P0_1/S_{OUT}$ | Open. | The key-on wakeup function is invalid ($K0_1$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR0_1$ = "0").<br>Pull-up transistor is OFF ($PU0_1$ = "0").<br>The key-on wakeup function is invalid ($K0_1$ = "0"). |
| $P0_2/S_{CK}$ | Open. | $S_{CK}$ pin is not selected ($J1_1J1_0$ = "00").<br>The key-on wakeup function is invalid ($K0_2$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR0_2$ = "0").<br>Pull-up transistor is OFF ($PU0_2$ = "0").<br>The key-on wakeup function is invalid ($K0_2$ = "0"). |
| $P0_3$ | Open. | The key-on wakeup function is invalid ($K0_3$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR0_3$ = "0").<br>Pull-up transistor is OFF ($PU0_3$ = "0").<br>The key-on wakeup function is invalid ($K0_3$ = "0"). |
| $P1_0$ | Open. | The key-on wakeup function is invalid ($K1_0$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR1_0$ = "0").<br>Pull-up transistor is OFF ($PU1_0$ = "0").<br>The key-on wakeup function is invalid ($K1_0$ = "0"). |
| $P1_1$/CNTR1 | Open. | CNTR1 input is not selected for the timer 1 count source ($W1_1$, $W1_0 \neq$ "10").<br>The key-on wakeup function is invalid ($K1_1$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR1_1$ = "0").<br>Pull-up transistor is OFF ($PU1_1$ = "0").<br>The key-on wakeup function is invalid ($K1_1$ = "0"). |
| $P1_2$/CNTR0 | Open. | CNTR0 input is not selected for the timer 2 count source ($W2_1$, $W2_0 \neq$ "10").<br>The key-on wakeup function is invalid ($K1_2$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR1_2$ = "0").<br>Pull-up transistor is OFF ($PU1_2$ = "0").<br>The key-on wakeup function is invalid ($K1_2$ = "0"). |
| $P1_3$/INT | Open. | INT pin input is disabled ($I1_3$ = "0").<br>The key-on wakeup function is invalid ($K1_3$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR1_3$ = "0").<br>Pull-up transistor is OFF ($PU1_3$ = "0").<br>The key-on wakeup function is invalid ($K1_3$ = "0"). |
| $P2_0/A_{IN0}$, $P2_1/A_{IN1}$ | Open. | The key-on wakeup function is invalid ($K2_0$, $K2_1$ = "0"). |
| | Connect to $V_{SS}$. | N-channel open-drain is selected for the output structure ($FR2_0$, $FR2_1$ = "0").<br>Pull-up transistor is OFF ($PU2_0$, $PU2_1$ = "0").<br>The key-on wakeup function is invalid ($K2_0$, $K2_1$ = "0"). |

(Note when connecting to $V_{SS}$ or $V_{DD}$)

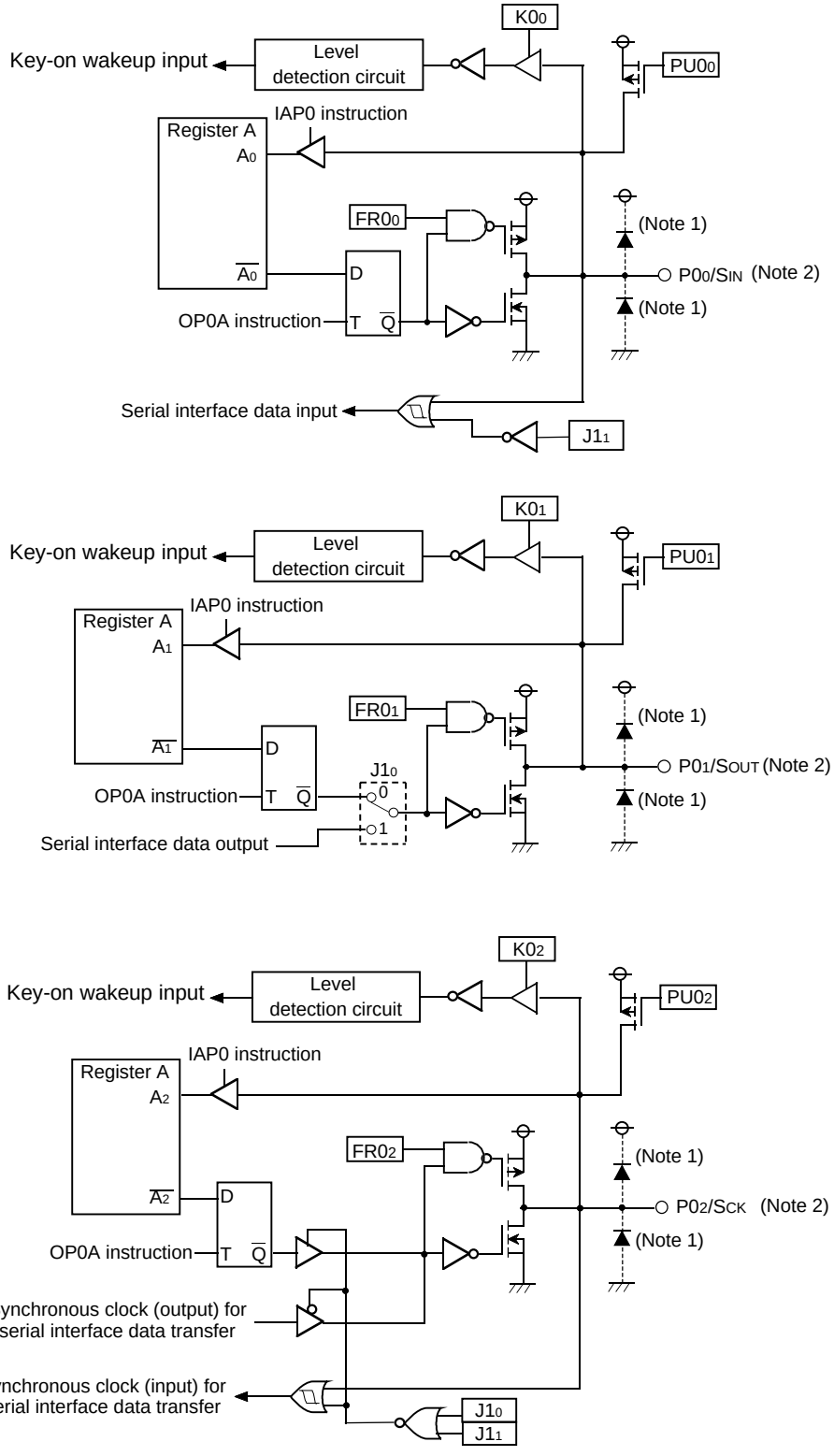
● Connect the unused pins to $V_{SS}$ using the thickest wire at the shortest distance against noise.

## PORT BLOCK DIAGRAMS



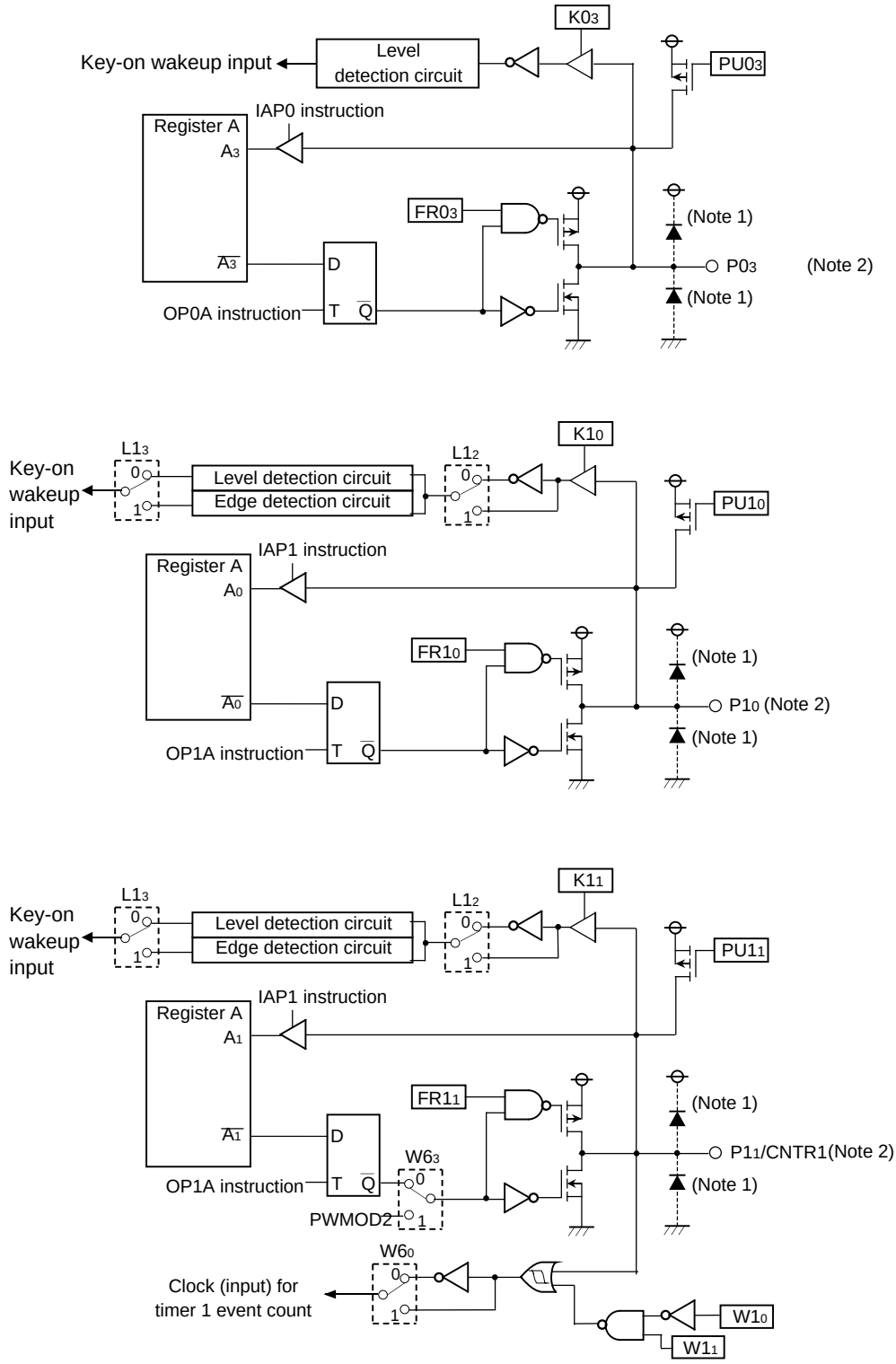

Notes 1: ----◄---- This symbol represents a parasitic diode on the port.
2: Applied potential to these ports must be $V_{DD}$ or less.
3: j represents bits 0 or 1.
4: k represents bits 2 or 3.

**Port block diagram (1)**



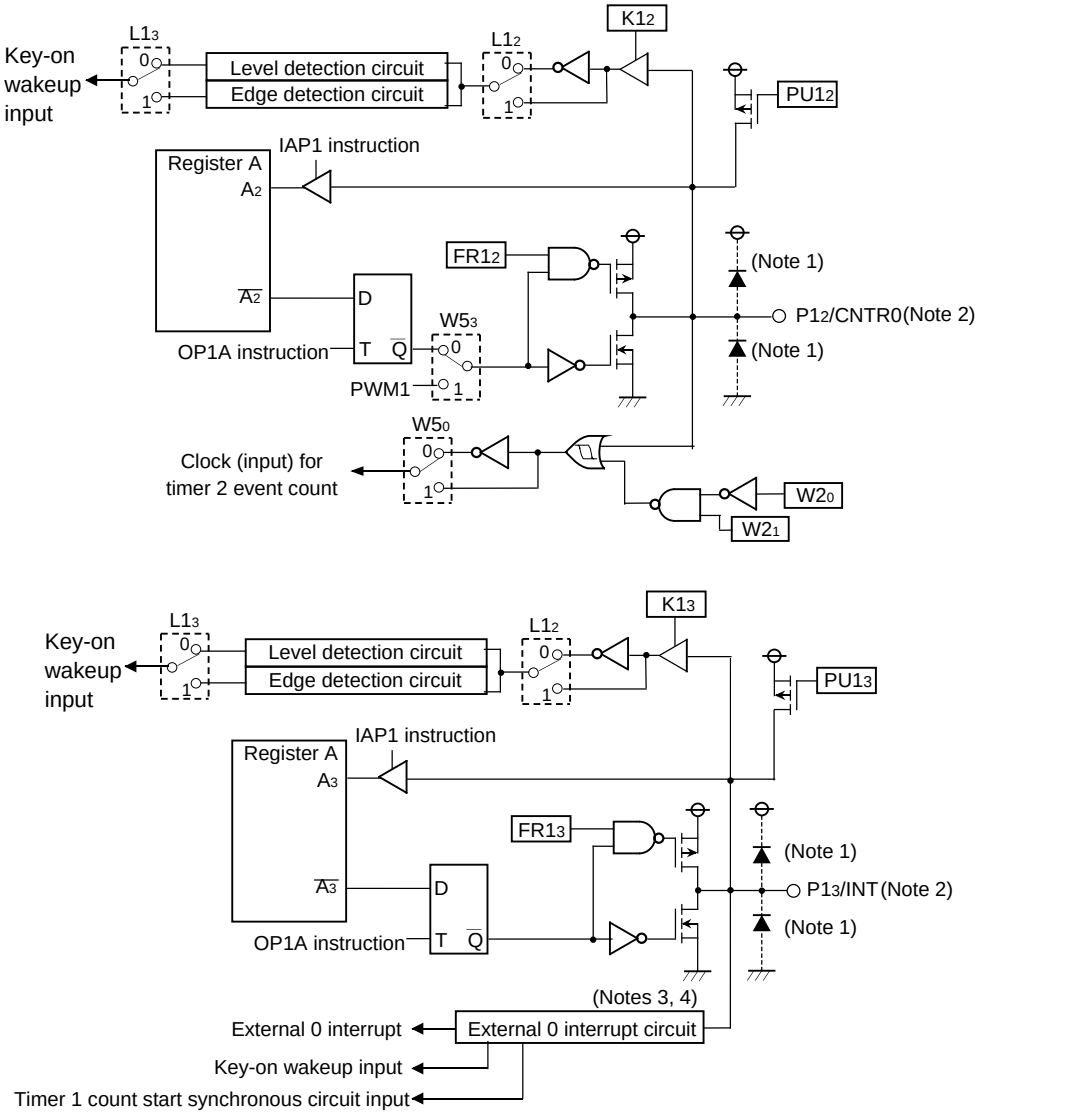

Notes 1: ----|◄---- This symbol represents a parasitic diode on the port.
2: Applied potential to these ports must be $V_{DD}$ or less.

**Port block diagram (2)**



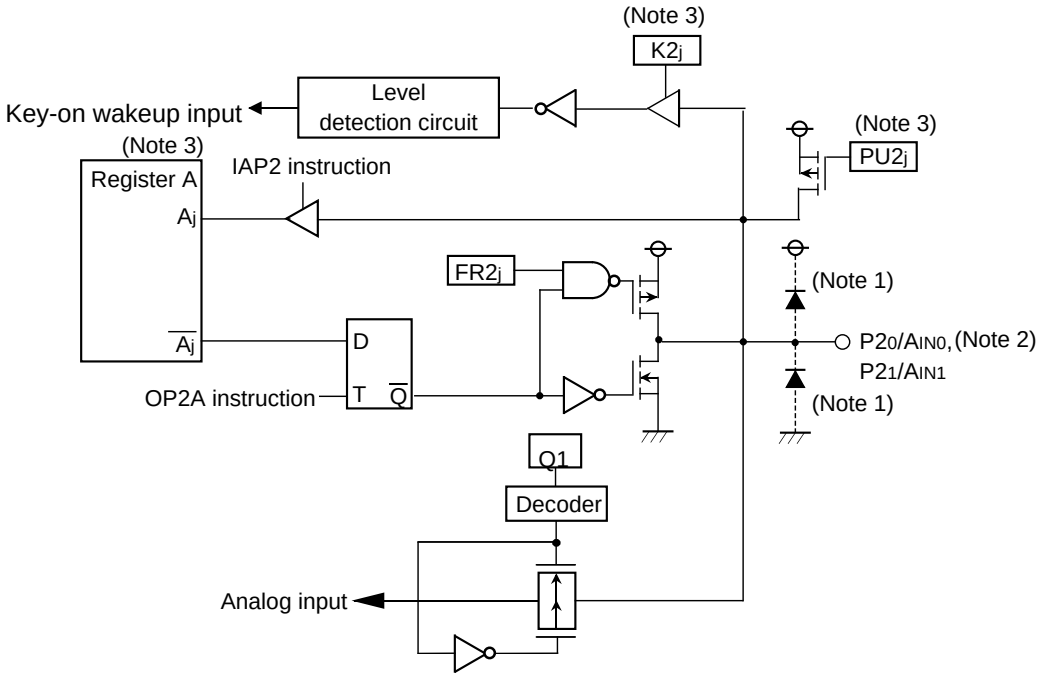

Notes 1: ----|◄---- This symbol represents a parasitic diode on the port.
2: Applied potential to these ports must be $V_{DD}$ or less.

**Port block diagram (3)**



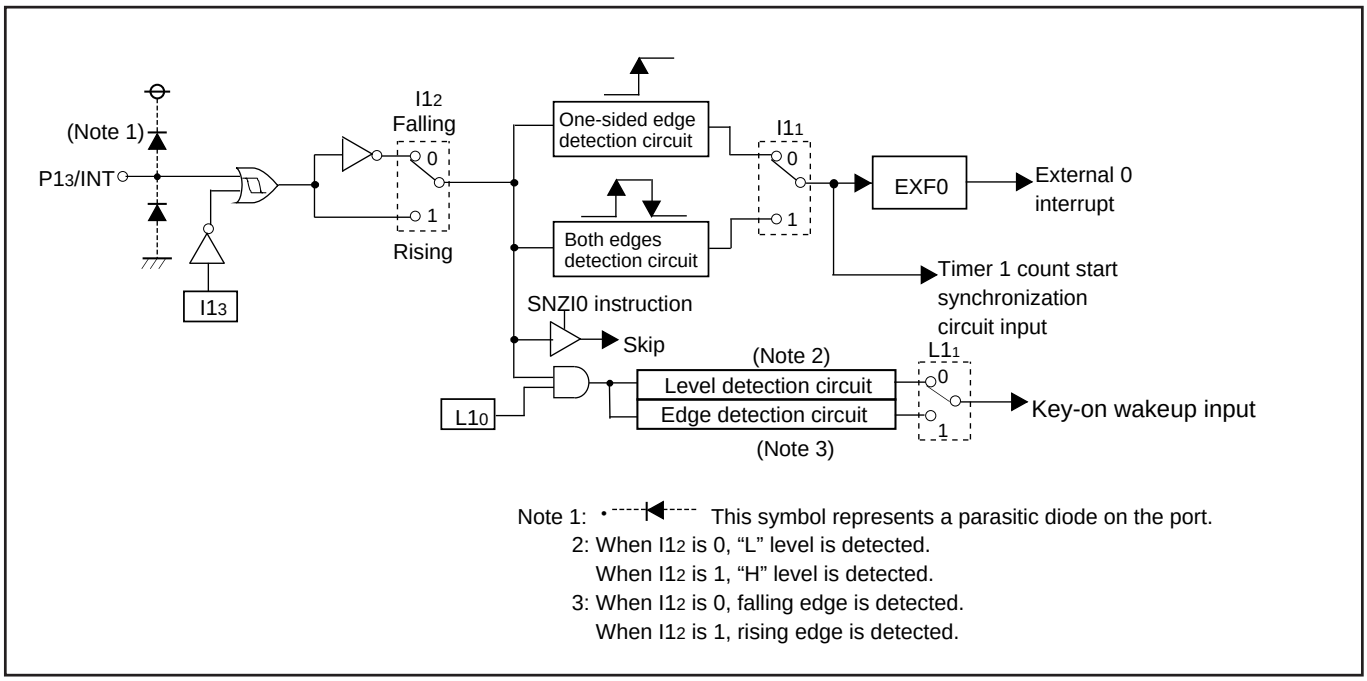

Notes 1: This symbol represents a parasitic diode on the port.
2: Applied potential to these ports must be $V_{DD}$ or less.
3: As for details, refer to the external interrupt structure.
4: The threshold value of port input is different from that of external interrupt input.

**Port block diagram (4)**




Notes 1: This symbol represents a parasitic diode on the port.
2: Applied potential to these ports must be VDD or less.
3: j represents 0 or 1.

**Port block diagram (5)**




**External interrupt circuit structure**

## 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

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).

Register E is undefined after system is released from reset and returned from the RAM back-up. Accordingly, set the initial value.

### (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).

Also, when the TABP p instruction is executed at UPTF flag = “1”, the high-order 2 bits of ROM reference data is stored to the low-order 2 bits of register D, the high-order 1 bit of register D is “0”. When the TABP p instruction is executed at UPTF flag = “0”, the contents of register D remains unchanged. The UPTF flag is set to “1” with the SUPT instruction and cleared to “0” with the RUPT instruction. The initial value of UPTF flag is “0”.

Register D is undefined after system is released from reset and returned from the RAM back-up. Accordingly, set the initial value.

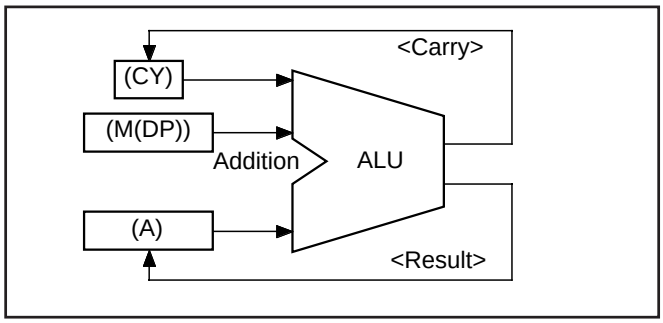



**Fig. 1 AMC instruction execution example**

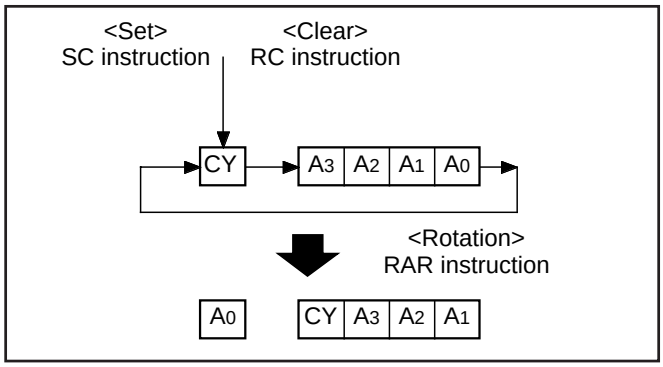



**Fig. 2 RAR instruction execution example**

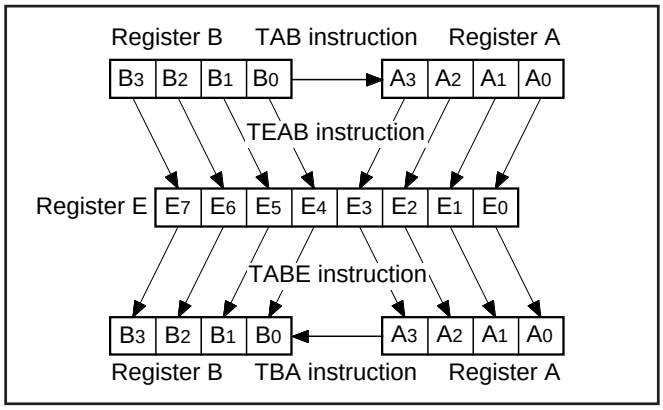



**Fig. 3 Registers A, B and register E**

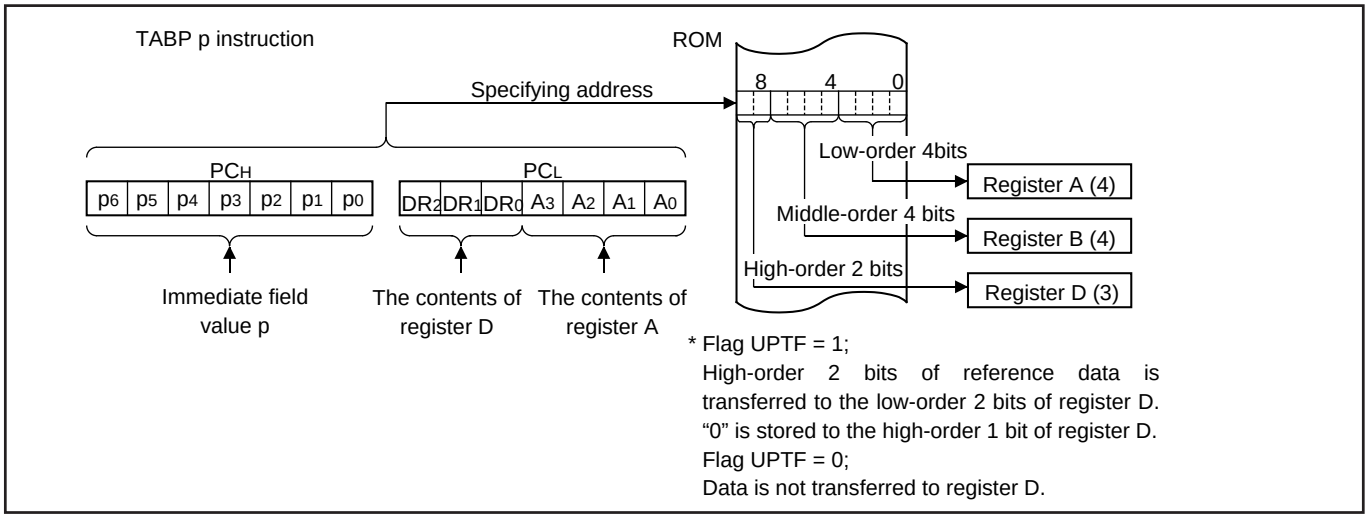



**Fig. 4 TABP p instruction execution example**

## (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 respectively when using an interrupt service routine and when executing a table reference instruction. Accordingly, be careful not to over the stack 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.

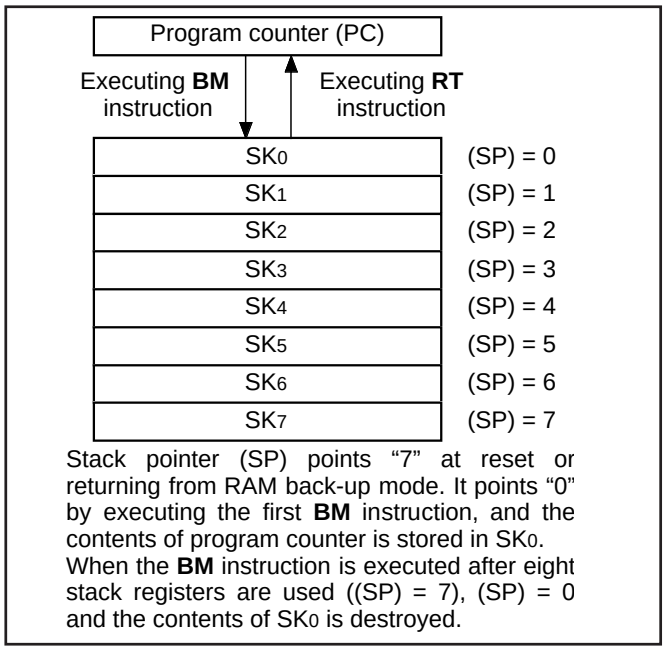



**Fig. 5 Stack registers (SKs) structure**

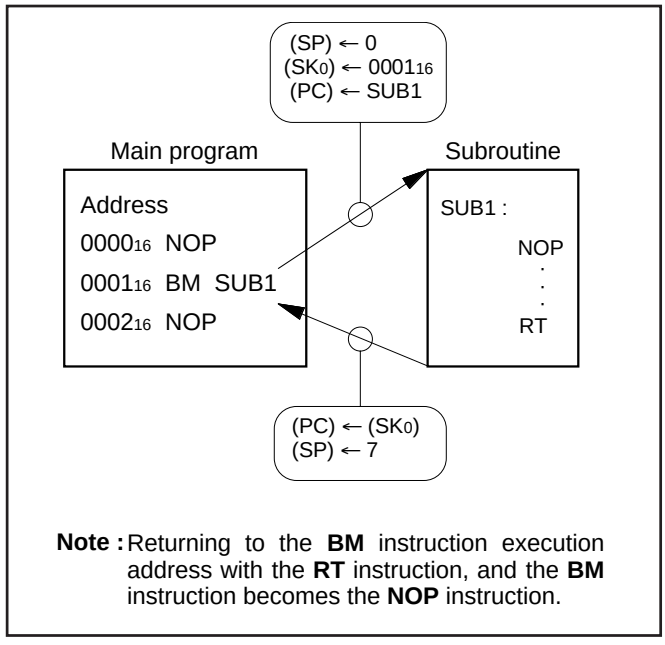



**Fig. 6 Example of operation at subroutine call**

## (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.

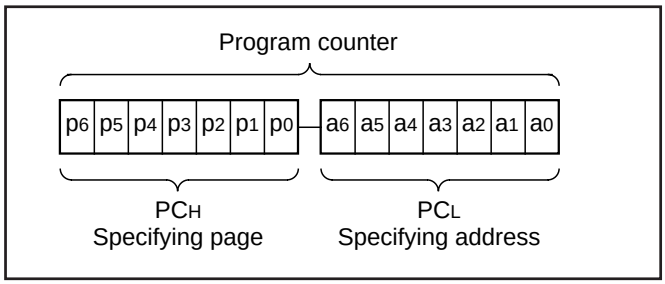



**Fig. 7 Program counter (PC) structure**

## (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, RD, or SZD instruction (Figure 9).

• Note

Register Z of data pointer is undefined after system is released from reset.

Also, registers Z, X and Y are undefined in the RAM back-up. After system is returned from the RAM back-up, set these registers.

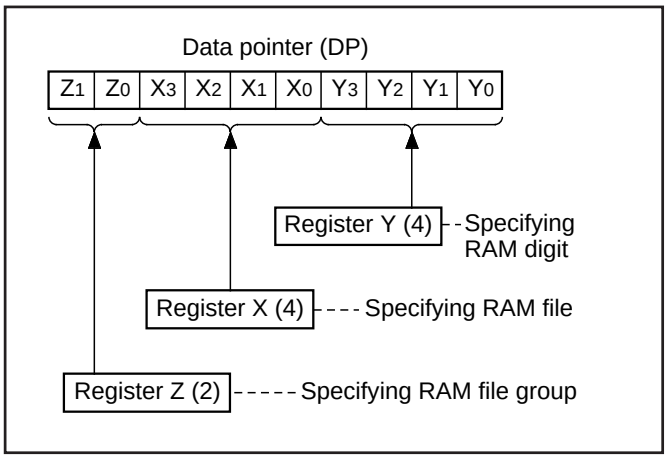



**Fig. 8 Data pointer (DP) structure**

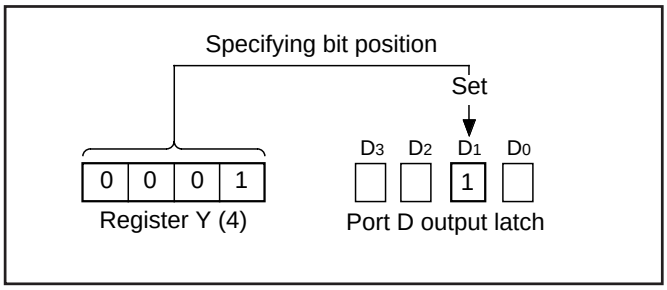



**Fig. 9 SD instruction execution example**

## PROGRAM MEMOY (ROM)

1 word of program memory 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 M34508G4.

**Table 1 ROM size and pages**

| Part number | ROM (PROM) size (× 10 bits) | Pages |
|---|---|---|
| M34508G4 | 4096 words | 32 (0 to 31) |
| M34508G4H | 4096 words | 32 (0 to 31) |

A 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 7 to 0) of all addresses can be used as data areas with the TABP p instruction.

## ROM Code Protect Address

When selecting the protect bit write by using a serial programmer or selecting protect enabled for writing shipment by Renesas Technology corp., reading or writing from/to QzROM is disabled by a serial programmer.

As for the QzROM product in blank, the ROM code is protected by selecting the protect bit write at ROM writing with a serial programmer.

As for the QzROM product shipped after writing, whether the ROM code protect is used or not can be selected as ROM option setup (“MASK option” written in the mask file converter) when ordering.

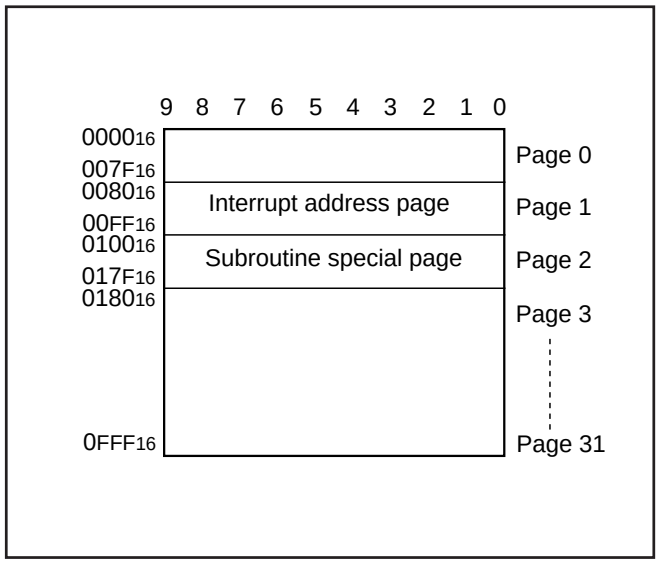



**Fig. 10 ROM map of M34508G4**

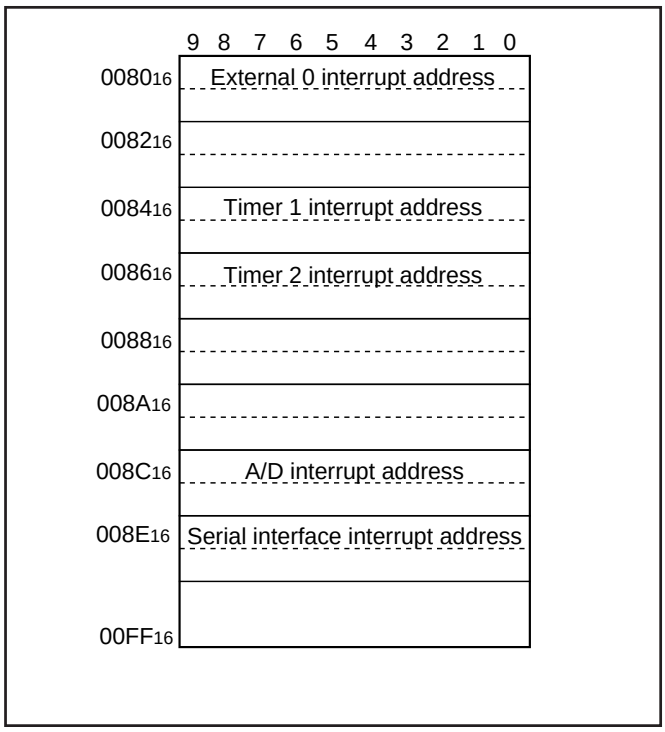



**Fig. 11 Page 1 (addresses $0080_{16}$ to $00FF_{16}$) structure**

## 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.

• Note
Register Z of data pointer is undefined after system is released from reset.
Also, registers Z, X and Y are undefined in the RAM back-up. After system is returned from the RAM back-up, set these registers.

**Table 2 RAM size**

| Part number | RAM size |
|---|---|
| M34508G4 | 256 words × 4 bits (1024 bits) |
| M34508G4H | 256 words × 4 bits (1024 bits) |

RAM 256 words × 4 bits (1024 bits)

| | Register Z | 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Register X | 0 | 1 | 2 | 3 | ··· | 6 | 7 | ····· | 15 |
| Register Y | 0 | | | | | | | | | |
| | 1 | | | | | | | | | |
| | 2 | | | | | | | | | |
| | 3 | | | | | | | | | |
| | 4 | | | | | | | | | |
| | 5 | | | | | | | | | |
| | 6 | | | | | | | | | |
| | 7 | | | | | | | | | |
| | 8 | | | | | | | | | |
| | 9 | | | | | | | | | |
| | 10 | | | | | | | | | |
| | 11 | | | | | | | | | |
| | 12 | | | | | | | | | |
| | 13 | | | | | | | | | |
| | 14 | | | | | | | | | |
| | 15 | | | | | | | | | |

**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.

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

Use an interrupt enable bit of interrupt control registers V1 and V2 to select the corresponding interrupt 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 | A/D interrupt | Completion of A/D conversion | Address C in page 1 |
| 5 | Serial interface interrupt | Completion of serial interface transmit/ recieve | Address E in page 1 |

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

| Interrupt name | Interrupt request flag | Skip instruction | Interrupt enable bit |
|---|---|---|---|
| External 0 interrupt | EXF0 | SNZ0 | $V1_0$ |
| Timer 1 interrupt | T1F | SNZT1 | $V1_2$ |
| Timer 2 interrupt | T2F | SNZT2 | $V1_3$ |
| A/D interrupt | ADF | SNZAD | $V2_2$ |
| Serial interface interrupt | SIOF | SNZSI | $V2_3$ |

**Table 5 Interrupt enable bit function**

| Interrupt enable bit | Occurrence of interrupt | 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 branching a data store sequence to stack register. Write the branch instruction to an interrupt service routine at an interrupt address.

Use the RTI instruction to return from an interrupt service 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 the main routine. (Refer to Figure 13)

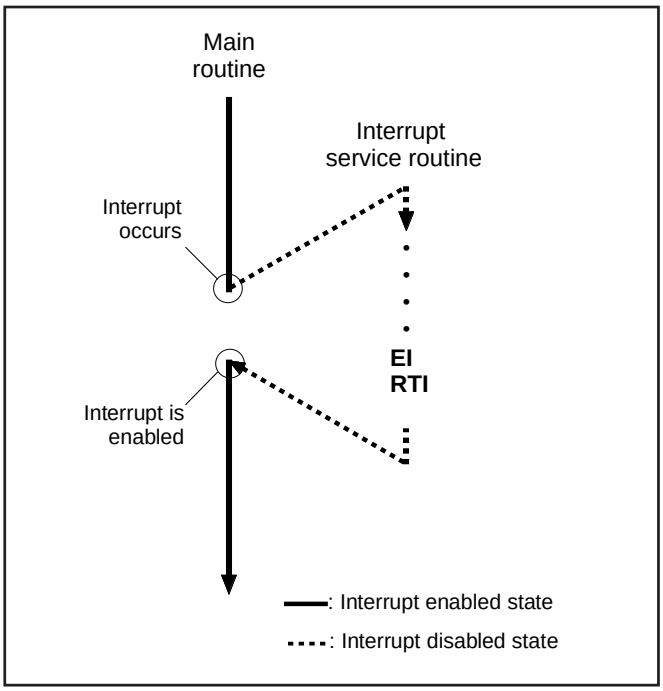



**Fig. 13 Program example of interrupt processing**

- 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**

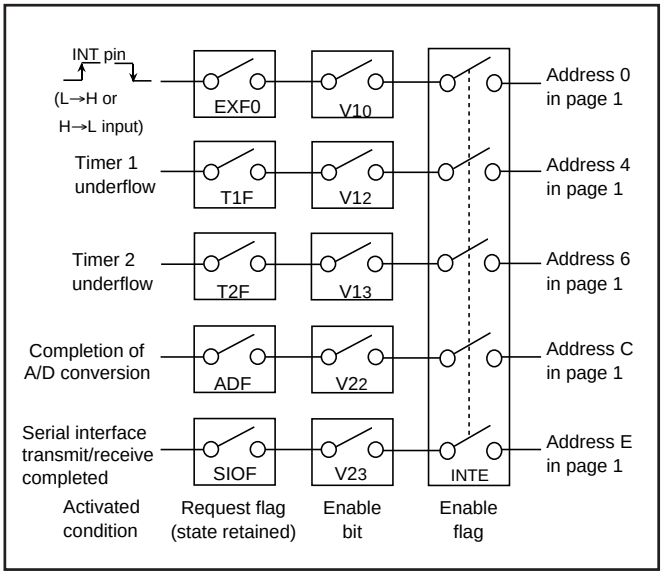



**Fig. 15 Interrupt system diagram**

## (6) Interrupt control registers

- 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
  The A/D interrupt enable bit and serial interface interrupt enable bit are 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 registers**

| Interrupt control register V1 | | at reset : $0000_2$ | at RAM back-up : $0000_2$ | R/W TAV1/TV1A |
|---|---|---|---|---|
| $V1_3$ | 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 TAV2/TV2A |
|---|---|---|---|---|
| $V2_3$ | Serial interface interrupt enable bit | 0 | Interrupt disabled (SNZSI instruction is valid) | |
| | | 1 | Interrupt enabled (SNZSI instruction is invalid) | |
| $V2_2$ | A/D interrupt enable bit | 0 | Interrupt disabled (SNZAD instruction is valid) | |
| | | 1 | Interrupt enabled (SNZAD instruction is invalid) | |
| $V2_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_0$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |

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

## (7) Interrupt sequence

Interrupts only occur when the respective INTE flag, interrupt enable bits ($V1_0$, $V1_2$, $V1_3$, $V2_2$, $V2_3$), and interrupt request flag 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 other than one-cycle instructions (Refer to Figure 16).

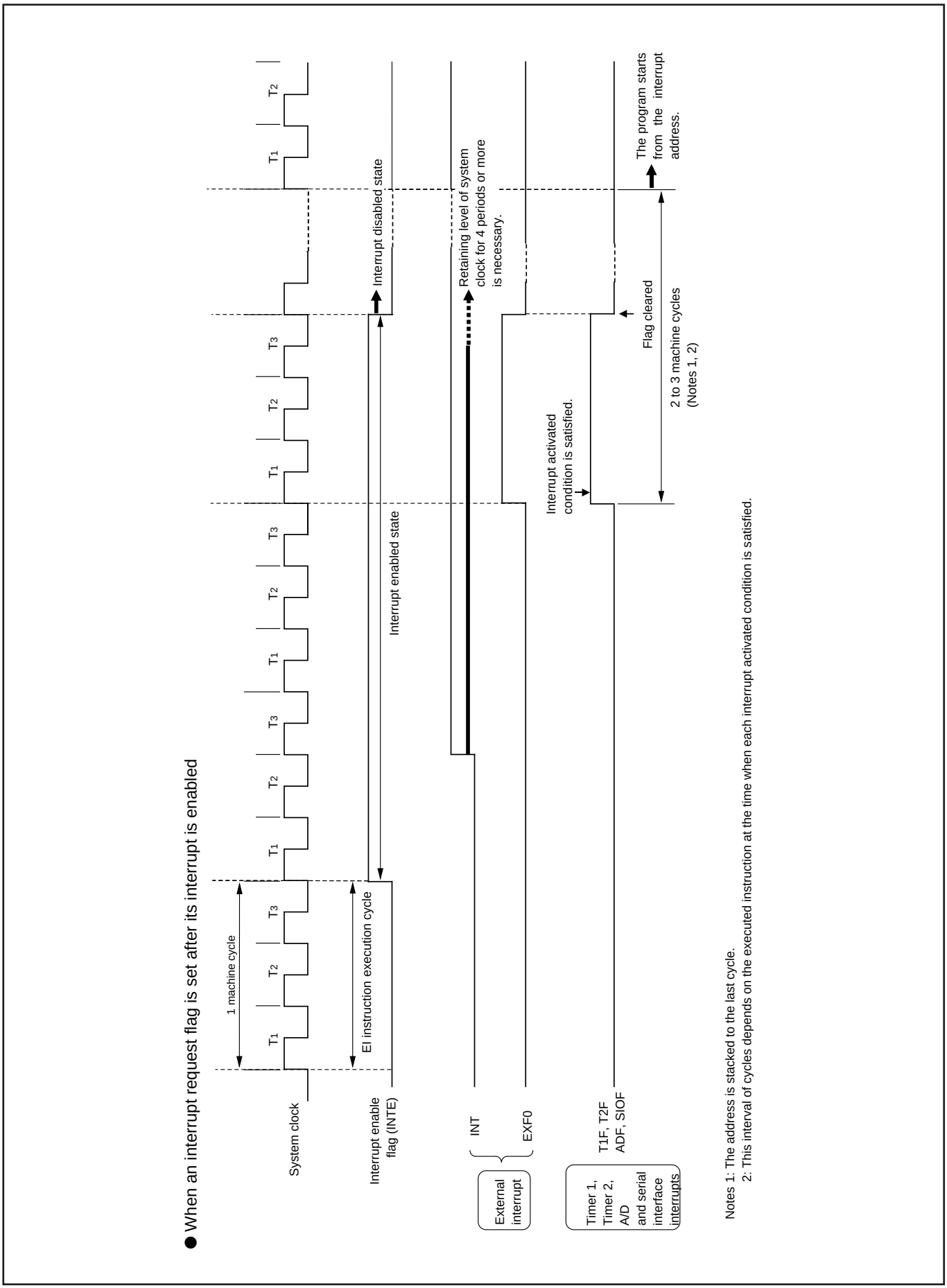



**Fig. 16 Interrupt sequence**

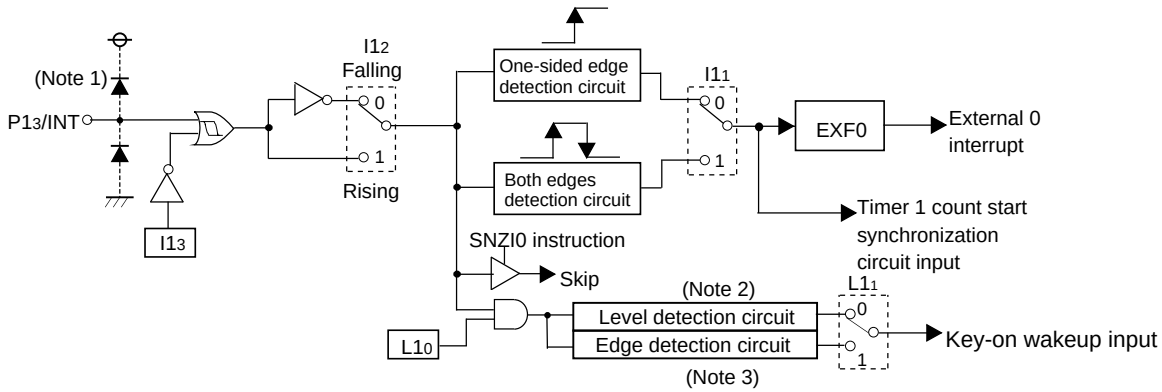
## EXTERNAL INTERRUPTS

The 4508 Group has the external 0 interrupt. An external interrupt request occurs when a valid waveform is input to an interrupt input pin (edge detection).

The external interrupt can be controlled with the interrupt control register I1.

**Table 7 External interrupt activated conditions**

| Name | Input pin | Activated condition | Valid waveform selection bit |
|---|---|---|---|
| External 0 interrupt | P13/INT | When the next waveform is input to P13/INT pin<br>• Falling waveform (“H”→“L”)<br>• Rising waveform (“L”→“H”)<br>• Both rising and falling waveforms | I11<br>I12 |




Note 1: • ⊣ This symbol represents a parasitic diode on the port.
2: When I12 is 0, “L” level is detected.
When I12 is 1, “H” level is detected.
3: When I12 is 0, falling edge is detected.
When I12 is 1, rising edge is detected.

**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 P13/INT pin.

The valid waveforms causing the interrupt must be retained at their level for 4 clock 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.

• External 0 interrupt activated condition

External 0 interrupt activated condition is satisfied when a valid waveform is input to P13/INT pin.

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

① Set the bit 3 of register I1 to “1” for the INT pin to be in the input enabled state.
② Select the valid waveform with the bits 1 and 2 of register I1.
③ Clear the EXF0 flag to “0” with the SNZ0 instruction.
④ Set the NOP instruction for the case when a skip is performed with the SNZ0 instruction.
⑤ Set both the external 0 interrupt enable bit (V10) and the INTE flag to “1.”

The external 0 interrupt is now enabled. Now when a valid waveform is input to the P13/INT pin, the EXF0 flag is set to “1” and the external 0 interrupt occurs.

## (2) External interrupt control registers

• Interrupt control register I1

Register I1 controls the valid waveform for the external 0 interrupt. 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 TAI1/TI1A |
|---|---|---|---|---|---|
| $I1_3$ | INT pin input control bit (Note 2) | 0 | INT pin input disabled | | |
| | | 1 | INT pin input enabled | | |
| $I1_2$ | 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 | | |
| $I1_1$ | INT pin edge detection circuit control bit | 0 | One-sided edge detected | | |
| | | 1 | Both edges detected | | |
| $I1_0$ | INT pin timer 1 control enable bit | 0 | Disabled | | |
| | | 1 | Enabled | | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.
2: When the contents of $I1_2$ and $I1_3$ are changed, the external interrupt request flag EXF0 may be set.

## (3) Notes on interrupts

① Note [1] on bit 3 of register I1

When the input of the INT pin is controlled with the bit 3 of register I1 in software, be careful about the following notes.

• Depending on the input state of the P13/INT pin, the external 0 interrupt request flag (EXF0) may be set when the bit 3 of register I1 is changed. In order to avoid the occurrence of an unexpected interrupt, clear the bit 0 of register V1 to "0" (refer to Figure 18①) and then, change the bit 3 of register I1.
In addition, execute the SNZ0 instruction to clear the EXF0 flag to "0" after executing at least one instruction (refer to Figure 18②).
Also, set the NOP instruction for the case when a skip is performed with the SNZ0 instruction (refer to Figure 18③).

```
    ⋮
LA    4       ; (XXX0₂)
TV1A          ; The SNZ0 instruction is valid ..... ①
LA    8       ; (1XXX₂)
TI1A          ; Control of INT pin input is changed
NOP           ..... ②
SNZ0          ; The SNZ0 instruction is executed
              (EXF0 flag cleared)
NOP           ..... ③
    ⋮
```

X : these bits are not used here.

**Fig. 18 External 0 interrupt program example-1**

② Note [2] on bit 3 of register I1

When the bit 3 of register I1 is cleared to "0", the RAM back-up mode is selected and the input of INT pin is disabled, be careful about the following notes.

• When the INT pin input is disabled (register I1$_3$ = "0"), set the key-on wakeup of INT pin to be invalid (register L1$_0$ = "0") before system enters to the RAM back-up mode. (refer to Figure 19①).

```
    ⋮
LA    0       ; (XXX0₂)
TI1A          ; INT key-on wakeup disabled ..... ①
DI
EPOF
POF           ; RAM back-up
    ⋮
```

X : these bits are not used here.

**Fig. 19 External 0 interrupt program example-2**

③ Note [3] on bit 2 of register I1

When the interrupt valid waveform of the P13/INT pin is changed with the bit 2 of register I1 in software, be careful about the following notes.

• Depending on the input state of the P13/INT pin, the external 0 interrupt request flag (EXF0) may be set when the bit 2 of register I1 is changed. In order to avoid the occurrence of an unexpected interrupt, clear the bit 0 of register V1 to "0" (refer to Figure 20①) and then, change the bit 2 of register I1 is changed.
In addition, execute the SNZ0 instruction to clear the EXF0 flag to "0" after executing at least one instruction (refer to Figure 20②).
Also, set the NOP instruction for the case when a skip is performed with the SNZ0 instruction (refer to Figure 20③).

```
    ⋮
LA    4       ; (XXX0₂)
TV1A          ; The SNZ0 instruction is valid ..... ①
LA    12      ; (1XXX₂)
TI1A          ; Interrupt valid waveform is changed
NOP           ..... ②
SNZ0          ; The SNZ0 instruction is executed
              (EXF0 flag cleared)
NOP           ..... ③
    ⋮
```

X : these bits are not used here.

**Fig. 20 External 0 interrupt program example-3**

## TIMERS

The 4508 Group has the following timers.

- Programmable timer
  The programmable timer has a reload register and enables the frequency dividing ratio to be set. It is decremented from a setting 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” after every n count of a count pulse.

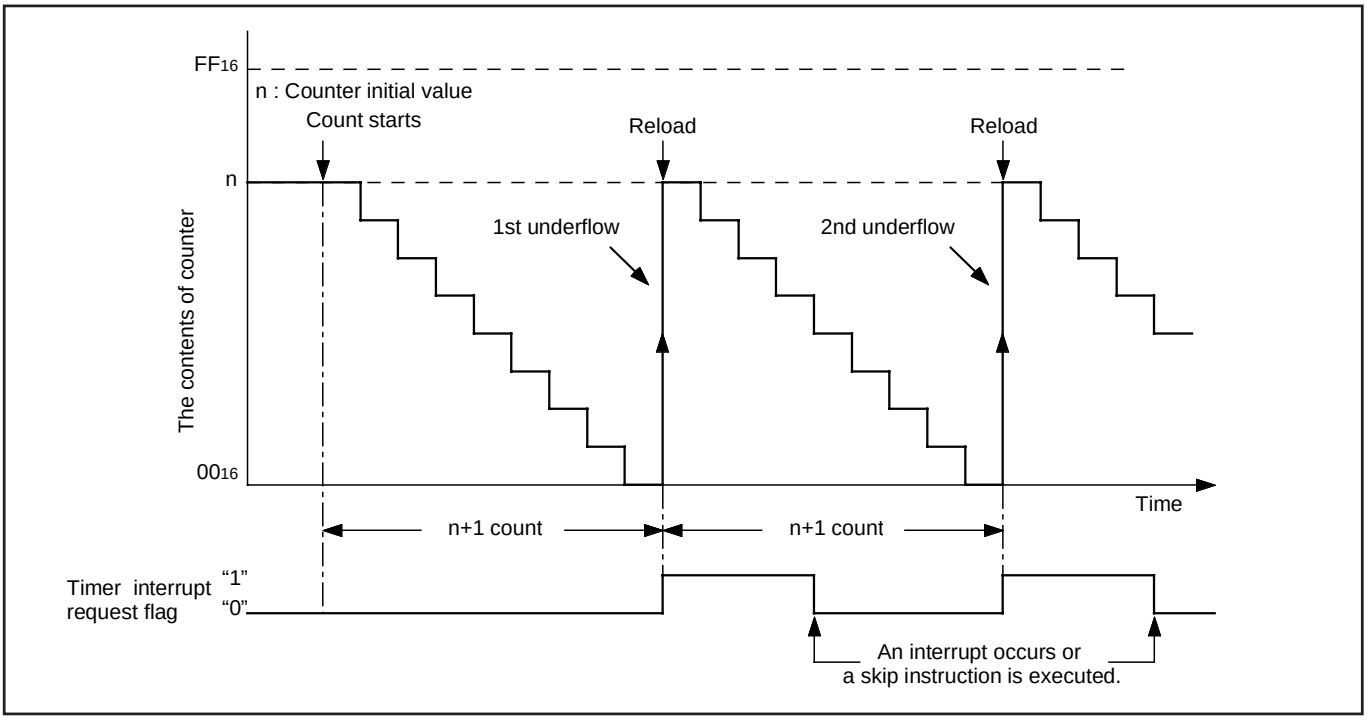



**Fig. 21 Auto-reload function**

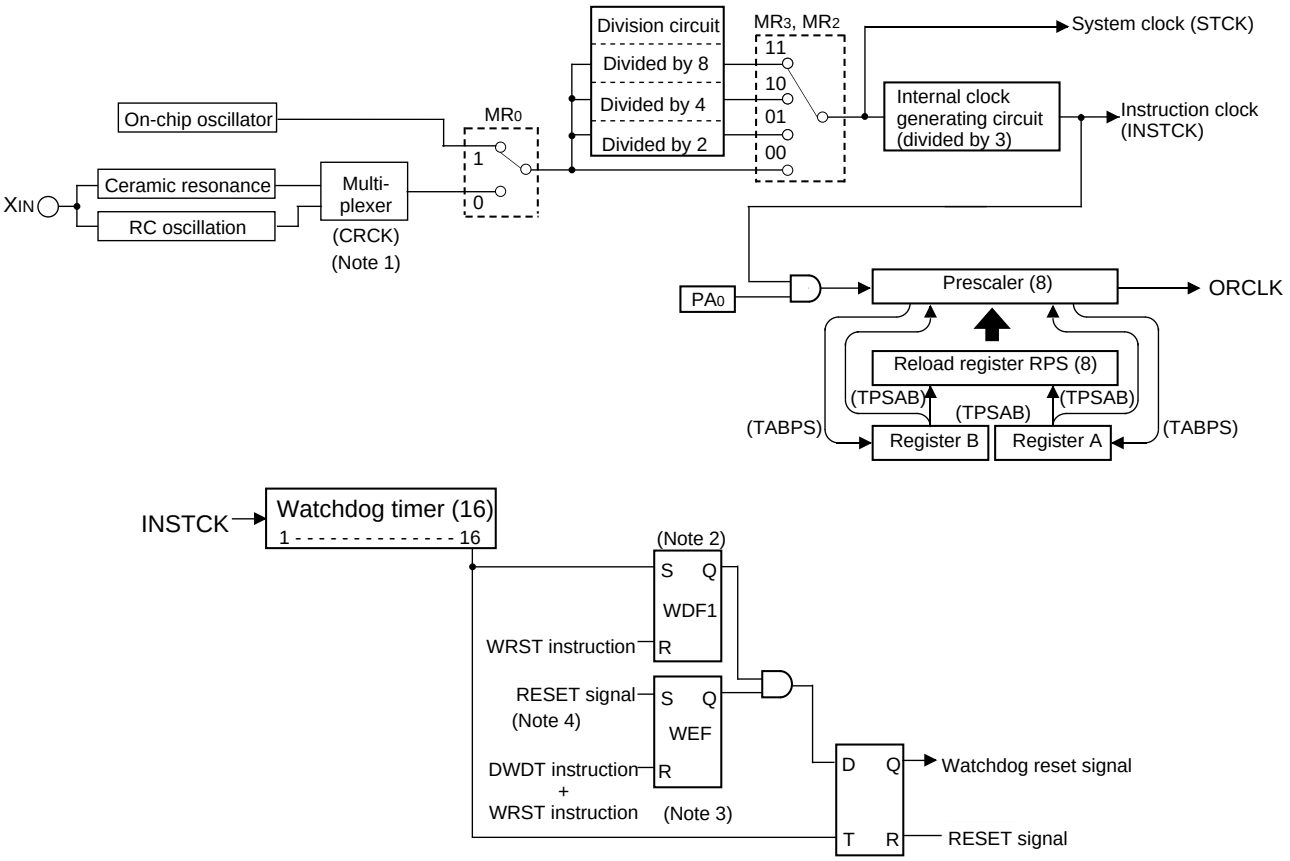
The 4508 Group timer consists of the following circuits.

- Prescaler : 8-bit programmable timer
- Timer 1 : 8-bit programmable timer
- Timer 2 : 8-bit programmable timer
  (Timers 1 and 2 have the interrupt function, respectively)
- 16-bit timer

Prescaler and timers 1 and 2 can be controlled with the timer control registers PA, W1, W2 , W5 and W6. The 16-bit timer is a free counter which is not controlled with the control register.
Each function is described below.

**Table 9 Function related timers**

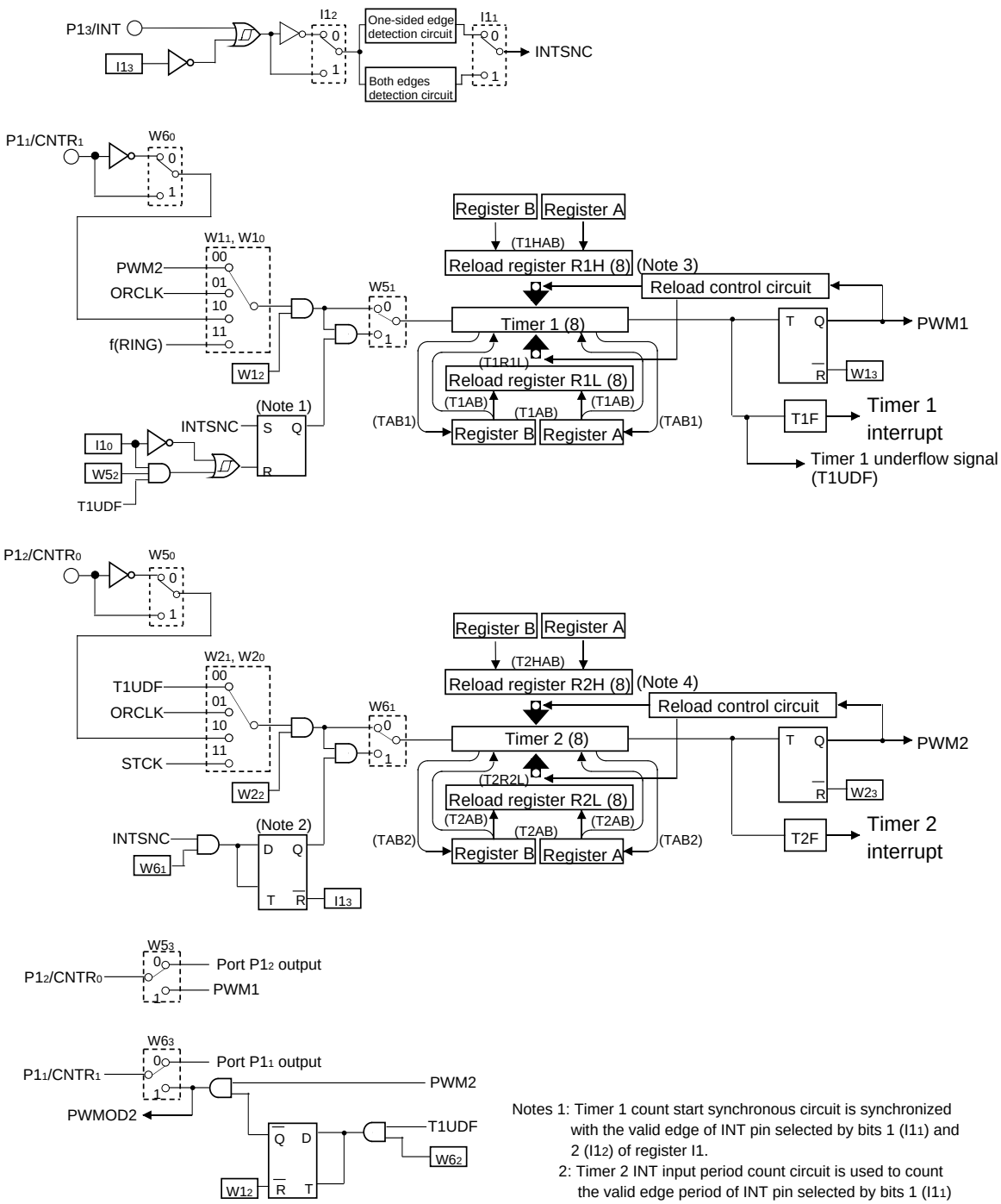
| Circuit | Structure | Count source | Frequency dividing ratio | Use of output signal | Control register |
|---|---|---|---|---|---|
| Prescaler | 8-bit programmable binary down counter | • Instruction clock (INSTCK) | 1 to 256 | • Timer 1 and 2 count sources | PA |
| Timer 1 | 8-bit programmable binary down counter (link to INT input) (with PWM output function) | • PWM2 signal (PWMOD2)<br>• Prescaler output (ORCLK)<br>• CNTR1 input<br>• On-chip oscillator clock (f(RING)) | 1 to 256 | • Timer 2 count source<br>• CNTR0 output<br>• Timer 1 interrupt | W1<br>W5<br>W6 |
| Timer 2 | 8-bit programmable binary down counter (INT input period count function) (with PWM output function) | • Timer 1 underflow (T1UDF)<br>• Prescaler output (ORCLK)<br>• CNTR0 input<br>• System clock (STCK) | 1 to 256 | • Timer 1 count source<br>• CNTR1 output<br>• Timer 2 interrupt | W2<br>W5<br>W6 |
| Watchdog timer | 16-bit fixed dividing frequency | • Instruction clock (INSTCK) | 65536 | • System reset (counting twice)<br>• Decision of flag WDF1 | - |




Data is set automatically from each reload register when timer underflows (auto-reload function).

Notes 1: When CRCK instruction is executed, RC oscillation is selected.
When CRCK instruction is not executed, ceramic resonance is selected.
2: Flag WDF1 is cleared to “0” and the next instruction is skipped when the WRST instruction is executed while flag WDF1 = “1”.
The next instruction is not skipped even when the WRST instruction is executed while flag WDF1 = “0”.
3: Flag WEF is cleared to “0” and watchdog timer reset does not occur when the DWDT instruction and WRST instruction are executed continuously.
4: The WEF flag is set to “1” at system reset or RAM back-up mode.

**Fig. 22 Timers structure (1)**




T1R1L: This instruction is used to transfer the contents of reload register R1L to timer 1.
T2R2L: This instruction is used to transfer the contents of reload register R2L to timer 2.
STCK: System clock
ORCLK: Prescaler output
▲ Data is set automatically from each reload register when timer underflows (auto-reload function).

Notes 1: Timer 1 count start synchronous circuit is synchronized with the valid edge of INT pin selected by bits 1 ($I1_1$) and 2 ($I1_2$) of register I1.
2: Timer 2 INT input period count circuit is used to count the valid edge period of INT pin selected by bits 1 ($I1_1$) and 2 ($I1_2$) of register I1.
3: When the PWM1 function is valid ($W1_3$="1"), the value is auto-reloaded alternately from reload register R1L and R1H every timer 1 underflow.
When the PWM1 function is invalid ($W1_3$="0"), the value is auto-reloaded from reload register R1L only.
4: When the PWM2 function is valid ($W2_3$="1"), the value is auto-reloaded alternately from reload register R2L and R2H every timer 2 underflow.
When the PWM2 function is invalid ($W2_3$="0"), the value is auto-reloaded from reload register R2L only.

**Fig. 23 Timers structure (2)**

**Table 10 Timer control registers**

| Timer control register PA | | at reset : $0_2$ | at RAM back-up : $0_2$ | W<br>TPAA |
|---|---|---|---|---|
| $PA_0$ | Prescaler control bit | 0 | Stop (state initialized) | |
| | | 1 | Operating | |

| Timer control register W1 | | at reset : $0000_2$ | | at RAM back-up : $0000_2$ | R/W<br>TAW1/TW1A |
|---|---|---|---|---|---|
| $W1_3$ | PWM1 function control bit | 0 | | PWM1 function invalid | |
| | | 1 | | PWM1 function valid | |
| $W1_2$ | Timer 1 control bit | 0 | | Stop (state retained) | |
| | | 1 | | Operating | |
| $W1_1$ | Timer 1 count source selection bits | $W1_1$ | $W1_0$ | Count source | |
| | | 0 | 0 | PWM2 signal | |
| $W1_0$ | | 0 | 1 | Prescaler output (ORCLK) | |
| | | 1 | 0 | CNTR1 input | |
| | | 1 | 1 | On-chip oscillator clock (f(RING)) | |

| Timer control register W2 | | at reset : $0000_2$ | | at RAM back-up : $0000_2$ | R/W<br>TAW2/TW2A |
|---|---|---|---|---|---|
| $W2_3$ | PWM2 function control bit | 0 | | PWM2 function invalid | |
| | | 1 | | PWM2 function valid | |
| $W2_2$ | Timer 2 control bit | 0 | | Stop (state retained) | |
| | | 1 | | Operating | |
| $W2_1$ | Timer 2 count source selection bits | $W2_1$ | $W2_0$ | Count source | |
| | | 0 | 0 | Timer 1 underflow signal (T1UDF) | |
| $W2_0$ | | 0 | 1 | Prescaler output (ORCLK) | |
| | | 1 | 0 | CNTR0 input | |
| | | 1 | 1 | System clock (STCK) | |

| Timer control register W5 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W<br>TAW5/TW5A |
|---|---|---|---|---|
| $W5_3$ | $P12$/CNTR0 pin function selection bit | 0 | $P12$ (I/O) / CNTR0 (input) | |
| | | 1 | $P12$ (input) /CNTR0 (I/O) | |
| $W5_2$ | Timer 1 count auto-stop circuit selection bit (Note 2) | 0 | Count auto-stop circuit not selected | |
| | | 1 | Count auto-stop circuit selected | |
| $W5_1$ | Timer 1 count start synchronous circuit selection bit (Note 3) | 0 | Count start synchronous circuit not selected | |
| | | 1 | Count start synchronous circuit selected | |
| $W5_0$ | CNTR0 pin input count edge selection bit | 0 | Falling edge | |
| | | 1 | Rising edge | |

| Timer control register W6 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W<br>TAW6/TW6A |
|---|---|---|---|---|
| $W6_3$ | $P11$/CNTR1 pin function selection bit | 0 | $P11$ (I/O) / CNTR1 (input) | |
| | | 1 | $P11$ (input) /CNTR1 (I/O) | |
| $W6_2$ | CNTR 1 pin output auto-control circuit selection bit | 0 | Output auto-control circuit not selected | |
| | | 1 | Output auto-control circuit selected | |
| $W6_1$ | Timer 2<br>INT pin input period count circuit selection bit | 0 | INT pin input period count circuit not selected | |
| | | 1 | INT pin input period count circuit selected | |
| $W6_0$ | CNTR1 pin input count edge selection bit | 0 | Falling edge | |
| | | 1 | Rising edge | |

Notes 1: "R" represents read enabled, and "W" represents write enabled.
2: This function is valid only when the INT pin/timer 1 control is enabled ($I1_0$="1") and the timer 1 count start synchronous circuit is selected ($W5_1$="1").
3: This function is valid only when the INT pin/timer 1 control is enabled ($I1_0$="1").

## (1) Timer control registers

- Timer control register PA
  Register PA controls the count operation of prescaler. Set the contents of this register through register A with the TPAA instruction.
- Timer control register W1
  Register W1 controls the count operation and count source of timer 1, and PWM1 function. 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 PWM2 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 W5
  Register W5 controls the input count edge of CNTR0 pin, timer 1 count start synchronous circuit, timer 1 auto-stop circuit and $P1_2$/CNTR0 pin function. Set the contents of this register through register A with the TW5A instruction. The TAW5 instruction can be used to transfer the contents of register W5 to register A.
- Timer control register W6
  Register W6 controls the input count edge of CNTR1 pin, the INT pin input count start synchronous circuit and CNTR1 pin output auto-control circuit and the $P1_1$/CNTR1 pin function. Set the contents of this register through register A with the TW6A instruction. The TAW6 instruction can be used to transfer the contents of register W6 to register A.

## (2) Prescaler

Prescaler is an 8-bit binary down counter with the prescaler reload register RPS. Data can be set simultaneously in prescaler and the reload register RPS with the TPSAB instruction. Data can be read from reload register RPS with the TABPS instruction.
Stop counting and then execute the TPSAB or TABPS instruction to read or set prescaler data.
Prescaler starts counting after the following process;
① set data in prescaler, and
② set the bit 0 of register PA to “1.”
When a value set in reload register RPS is n, prescaler divides the count source signal by n + 1 (n = 0 to 255).
Count source for prescaler is the instruction clock (INSTCK).
Once count is started, when prescaler underflows (the next count pulse is input after the contents of prescaler becomes “0”), new data is loaded from reload register RPS, and count continues (auto-reload function).
The output signal (ORCLK) of prescaler can be used for timer 1 and 2 count sources.

## (3) Timer 1 (interrupt function)

Timer 1 is an 8-bit binary down counter with two timer 1 reload registers (R1L, R1H). Data can be set simultaneously in timer 1 and the reload register R1L with the T1AB instruction. Data can be set in the reload register R1H with the T1HAB instruction. The contents of reload register R1L set with the T1AB instruction can be set to timer 1 again with the T1R1L instruction. Data can be read from timer 1 with the TAB1 instruction.
Stop counting and then execute the T1AB or TAB1 instruction to read or set timer 1 data.
When executing the T1HAB instruction to set data to reload register R1H while timer 1 is operating, avoid a timing when timer 1 underflows.
Timer 1 starts counting after the following process;
① set data in timer 1
② set count source by bits 0 and 1 of register W1, and
③ set the bit 2 of register W1 to “1.”

When a value set in reload register R1L is n and a value set in reload register R1H is m, timer 1 divides the count source signal by n + 1 or m + 1 (n = 0 to 255, m = 0 to 255).
<Bit 3 of register W1 = “0” (PWM1 function invalid)>
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 R1L, and count continues (auto-reload function).
<Bit 3 of register W1 = “1” (PWM1 function valid)>
Timer 1 generates the PWM1 signal of the “L” interval set as reload register R1L, and the “H” interval set as reload register R1H. The PWM1 signal generated by timer 1 is output from CNTR0 pin by setting “1” to bit 3 of register W5.
After timer 1 control by INT pin is enabled by setting the bit 0 of register I1 to “1”, INT pin input can be used as the start trigger for timer 1 count operation by setting the bit 1 of register W5 to “1”.
Also, in this time, the auto-stop function by timer 1 underflow can be performed by setting the bit 2 of register W5 to “1.”

## (4) Timer 2 (interrupt function)

Timer 2 is an 8-bit binary down counter with two timer 2 reload registers (R2L, R2H). Data can be set simultaneously in timer 2 and the reload register R2L with the T2AB instruction. Data can be set in the reload register R2H with the T2HAB instruction. The contents of reload register R2L set with the T2AB instruction can be set to timer 2 again with the T2R2L instruction. Data can be read from timer 2 with the TAB2 instruction.

Stop counting and then execute the T2AB or TAB2 instruction to read or set timer 2 data.

When executing the T2HAB instruction to set data to reload register R2H while timer 2 is operating, avoid a timing when timer 2 underflows.

Timer 2 starts counting after the following process;

① set data in timer 2

② set count source by bits 0 and 1 of register W2, and

③ set the bit 2 of register W2 to "1."

When a value set in reload register R2L is n and a value set in reload register R2H is m, timer 2 divides the count source signal by n + 1 or m + 1 (n = 0 to 255, m = 0 to 255).

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 R2L, and count continues (auto-reload function).

<Bit 3 of register W2 = "0" (PWM2 function invalid)>

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 R2L, and count continues (auto-reload function).

<Bit 3 of register W2 = "1" (PWM2 function valid)>

Timer 2 generates the PWM2 signal of the "L" interval set as reload register R2L, and the "H" interval set as reload register R2H. The PWM2 signal generated by timer 2 is output from CNTR1 pin by setting "1" to bit 3 of register W6.

PWM2 output to CNTR1 pin combined with timer 1 can be controlled by setting the bit 2 of register W6 to "1."

Input period of INT pin by timer 2 can be counted by setting the bit 1 of register W6 to "1."

## (5) Count start synchronization circuit (timer 1)

Timer 1 has the count start synchronous circuit which synchronizes the input of INT pin, and can start the timer count operation.

Timer 1 count start synchronous circuit function can be selected after timer 1 control by INT pin is enabled by setting the bit 0 of register I1 to "1" and its function is selected by setting the bit 1 of register W5 to "1".

When timer 1 count start synchronous circuit is used, the count start synchronous circuit is set, the count source is input to timer by inputting valid waveform to INT pin.

The valid waveform of INT pin to set the count start synchronous circuit is the same as the external interrupt activated condition.

Once set, the count start synchronous circuit is cleared by clearing the bit $I1_0$ to "0" or system reset.

However, when the count auto-stop circuit is selected ($W2_2$ = "1"), the count start synchronous circuit is cleared (auto-stop) at the timer 1 underflow.

## (6) Count auto-stop circuit (timer 1)

Timer 1 has the count auto-stop circuit which is used to stop timer 1 automatically by the timer 1 underflow when the count start synchronous circuit is used.

The count auto-stop circuit is valid by setting the bit 2 of register W5 to "1". It is cleared by the timer 1 underflow and the count source to timer 1 is stopped.

This function is valid only when the timer 1 count start synchronous circuit is selected.

## (7) INT pin input period count circuit (timer 2)

Timer 2 has the INT pin input period count circuit to count the valid waveform input interval of the INT pin.

When bit 1 of register W6 is set to "1", the INT pin input period count circuit of timer 2 becomes valid, and the count source is input. The count source input is stopped by the next input of valid waveform to the INT pin.

Then, every a valid waveform is input to the INT pin, start/stop of the count source input is alternately repeated.

A valid waveform of the INT pin input is the same as the activated condition of an external interrupt.

The INT pin input period count circuit set once is cleared by setting the INT pin input to be disabled state. The INT pin input can be disabled by clearing bit 3 of register I1 to "0".

## (8) Timer input/output pin ($P12$/CNTR0 pin, $P11$/CNTR1 pin)

CNTR0 pin is used to input the timer 2 count source and output the PWM1 signal generated by timer 1.

CNTR1 pin is used to input the timer 1 count source and output the PWM2 signal generated by timer 2.

The $P12$/CNTR0 pin function can be selected by bit 3 of register W5.

The $P11$/CNTR1 pin function can be selected by bit 3 of register W6.

When the CNTR0 input is selected for timer 2 count source, timer 2 counts the falling or rising waveform of CNTR0 input. The count edge is selected by bit 0 of register W5.

When the CNTR1 input is selected for timer 1 count source, timer 1 counts the falling or rising waveform of CNTR1 input. The count edge is selected by bit 0 of register W6.

## (9) PWM1 output function (P12/CNTR0, timer 1)

When bit 3 of register W1 is set to “1”, the data is reloaded alternately from reload register R1L and R1H every timer 1 underflow.
Timer 1 generates the PWM1 signal of the “L” interval set as reload register R1L, and the “H” interval set as reload register R1H.
In this time, the PWM1 signal generated by timer 1 is output from CNTR0 pin by setting “1” to bit 3 of register W5.
When the TW1A instruction is executed while the PWM1 signal is “H”, the contents of register W1 is changed after the “H” interval of the PWM1 signal is ended.

## (10) PWM2 output function (P11/CNTR1, timer 1, timer 2)

When bit 3 of register W2 is set to “1”, the data is reloaded alternately from reload register R2L and R2H every timer 2 underflow.
Timer 2 generates the PWM2 signal of the “L” interval set as reload register R2L, and the “H” interval set as reload register R2H.
In this time, the PWM2 signal generated by timer 2 is output from CNTR1 pin by setting “1” to bit 3 of register W6.
When bit 2 of register W6 is set to “1”, the PWM2 signal output to CNTR1 pin is switched to valid/invalid alternately each timer 1 underflow. However, when timer 1 is stopped (bit 2 of register W1 is cleared to “0”), this function is canceled.
When the TW2A instruction is executed while the PWM2 signal is “H”, the contents of register W2 is changed after the “H” interval of the PWM2 signal is ended.

## (11) Timer interrupt request flags (T1F, T2F)

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).
Use the interrupt control register V1, 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.

## (12) Precautions

- Prescaler
  Stop prescaler counting and then execute the TABPS instruction to read its data.
  Stop prescaler counting and then execute the TPSAB instruction to write data to prescaler.

- Timer count source
  Stop timer 1 or 2 counting to change its count source.

- Reading the count value
  Stop timer 1 or 2 counting and then execute the TAB1 or TAB2 instruction to read its data.

- Writing to the timer
  Stop timer 1 or 2 counting and then execute the T1AB, T1R1L, T2AB or T2R2L instruction to write data to timer.

- Writing to reload register
  In order to write a data to the reload register R1H while the timer 1 is operating, execute the T1HAB instruction except a timing of the timer 1 underflow.
  In order to write a data to the reload register R2H while the timer 2 is operating, execute the T2HAB instruction except a timing of the timer 2 underflow.

- PWM signal (PWM1, PWM2)
  If the timer 1 count stop timing and the timer 1 underflow timing overlap during output of the PWM1 signal, a hazard may occur in the PWM1 output waveform.
  If the timer 2 count stop timing and the timer 2 underflow timing overlap during output of the PWM2 signal, a hazard may occur in the PWM2 output waveform.

- Prescaler, timer 1 and timer 2 count start timing and count time when operation starts
  Count starts from the first rising edge of the count source (2) after prescaler and timer operations start (1).
  Time to first underflow (3) is shorter (for up to 1 period of the count source) than time among next underflow (4) by the timing to start the timer and count source operations after count starts.
  When selecting CNTR input as the count source of timer, timer operates synchronizing with the count edge (falling edge or rising edge) of CNTR input selected by software.

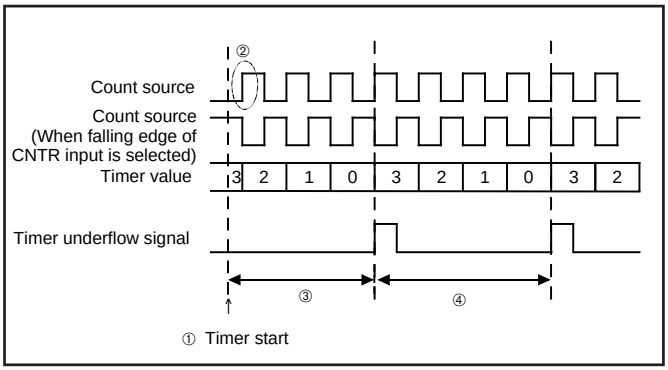



**Fig. 24 Timer count start timing and count time when operation starts**

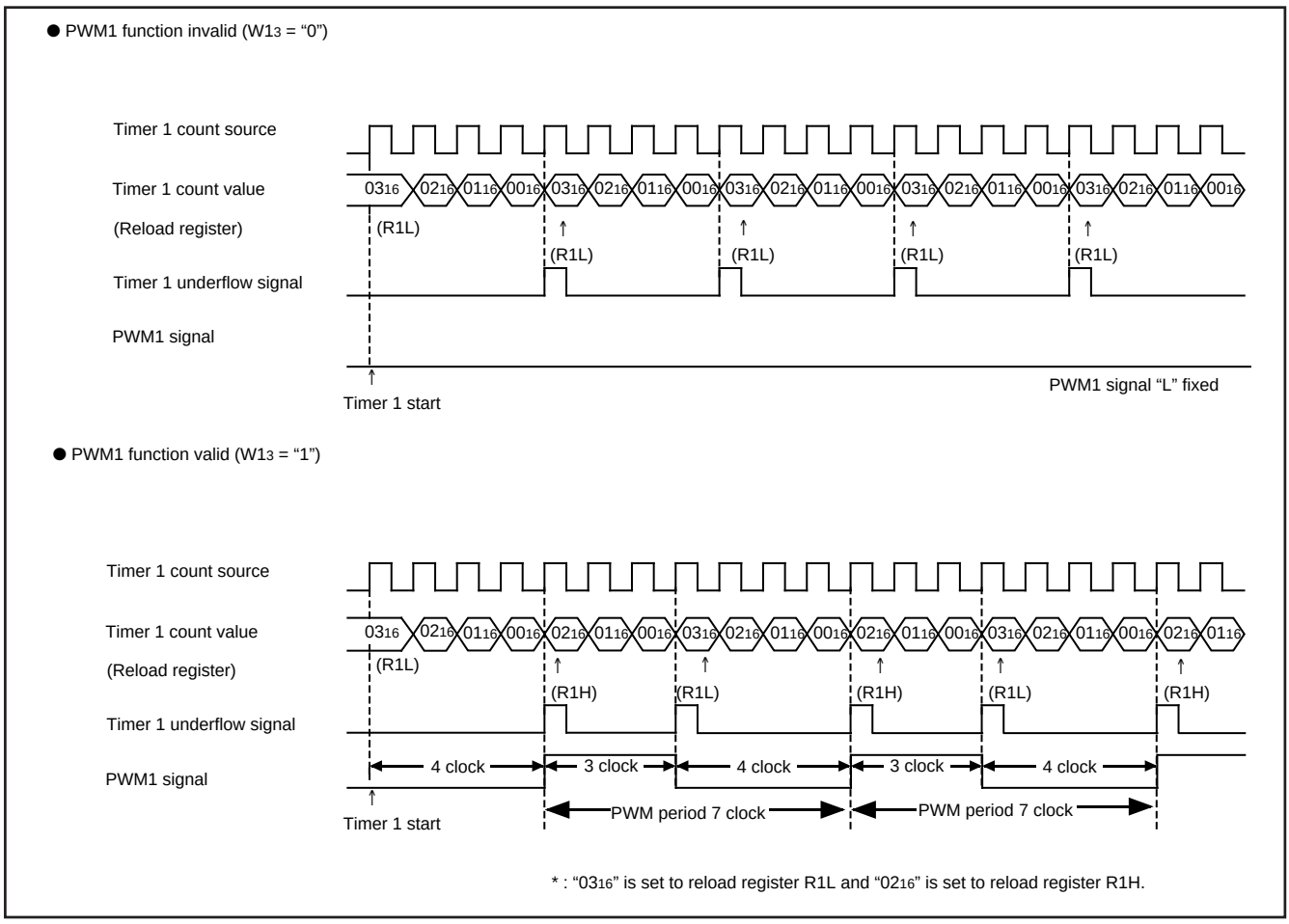
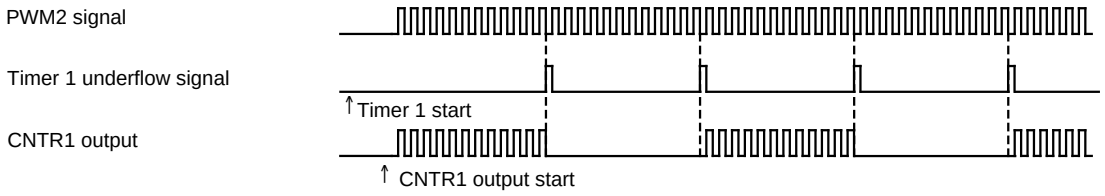


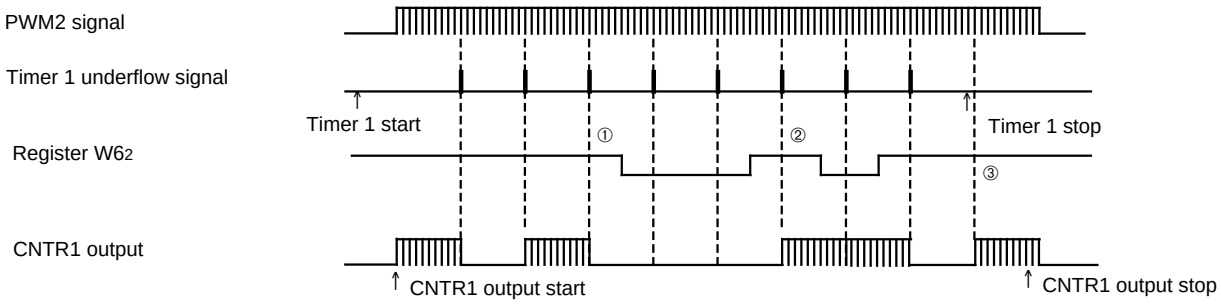

**Fig. 25 Timer 1 operation example**

● CNTR1 output auto-control circuit operation example 1 ($W2_3$ = "1", $W6_3$ = "1", $W6_2$ = "1")




* When the CNTR1 output auto-control circuit is selected, valid/invalid of CNTR1 output is repeated every timer 1 underflows.

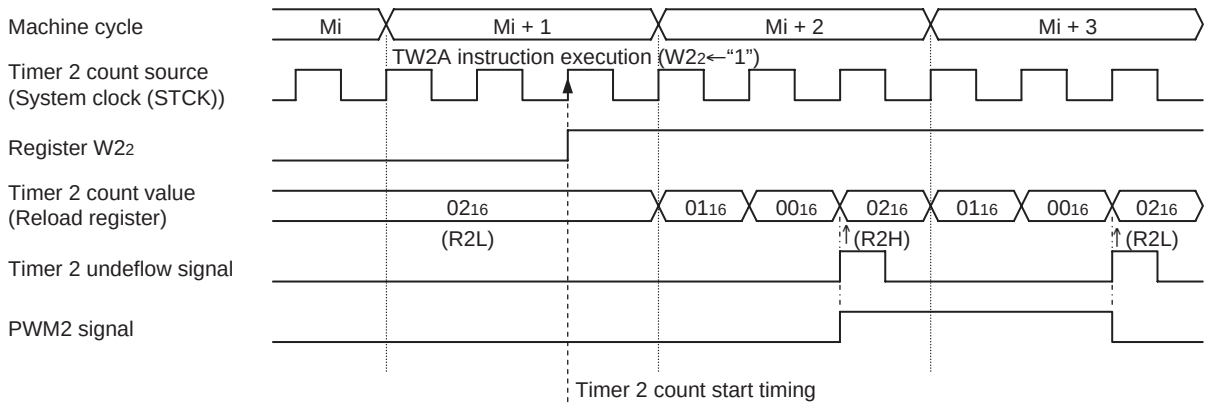
● CNTR1 output auto-control circuit operation example 2 ($W2_3$ = "1", $W6_3$ = "1")



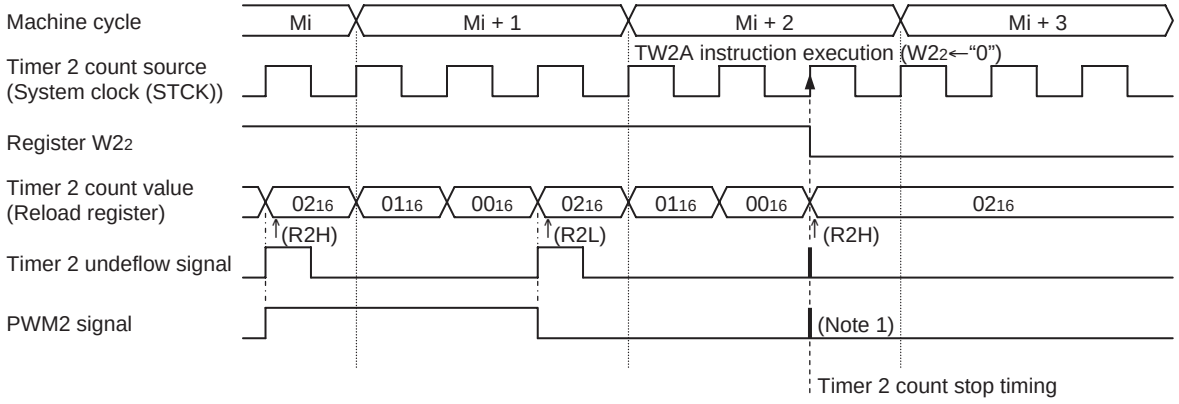

① When the CNTR1 output auto-control function is not selected while the CNTR output is invalid, CNTR1 output invalid state is retained.
② When the CNTR1 output auto-control function is not selected while the CNTR output is valid, CNTR1 output valid state is retained.
③ When the timer 1 is stopped, the CNTR1 output auto-control function becomes invalid.

**Fig. 26 CNTR1 output auto-control function by timer 1**

● Timer 2 count start timing (R2L = “$02_{16}$”, R2H = ”$02_{16}$”, $W2_3$ = “1”)




● Timer 2 count stop timing (R2L = “$02_{16}$”, R2H = ”$02_{16}$”, $W2_3$ = “1”)




Notes 1: If the timer count stop timing and the timer underflow timing overlap while the PWM function is valid ($W1_3$=“1” or $W2_3$=“1”), a hazard may occur in the PWM signal waveform.
2: When timer count is stopped during “H” duration of the PWM signal, timer is stopped after the end of the “H” output duration.

**Fig. 27 Timer count start/stop timing**

## WATCHDOG TIMER

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

The timer WDT downcounts the instruction clocks as the count source from "$FFFF_{16}$" after system is released from reset.

After the count is started, when the timer WDT underflow occurs (after the count value of timer WDT reaches "$FFFF_{16}$," the next count pulse is input), the WDF1 flag is set to "1."

If the WRST instruction is never executed until the timer WDT underflow occurs (until timer WDT counts 65534), WDF2 flag is set to "1," and the $\overline{\text{RESET}}$ pin outputs "L" level to reset the microcomputer.

Execute the WRST instruction at each period of 65534 machine cycle or less by software when using watchdog timer to keep the microcomputer operating normally.

When the WEF flag is set to "1" after system is released from reset, the watchdog timer function is valid.

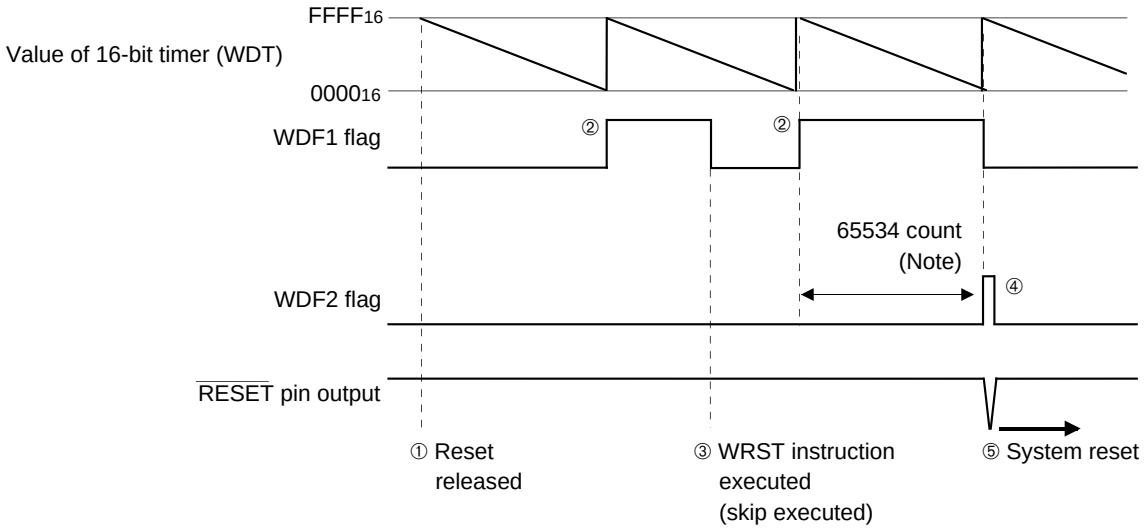
When the DWDT instruction and the WRST instruction are executed continuously, the WEF flag is cleared to "0" and the watchdog timer function is invalid.

The WEF flag is set to "1" at system reset or RAM back-up mode.

The WRST instruction has the skip function. When the WRST instruction is executed while the WDF1 flag is "1", the WDF1 flag is cleared to "0" and the next instruction is skipped.

When the WRST instruction is executed while the WDF1 flag is "0", the next instruction is not skipped.

The skip function of the WRST instruction can be used even when the watchdog timer function is invalid.




① After system is released from reset (= after program is started), timer WDT starts count down.
② When timer WDT underflow occurs, WDF1 flag is set to "1."
③ When the WRST instruction is executed while the WDF1 flag is "1", WDF1 flag is cleared to "0," the next instruction is skipped.
④ When timer WDT underflow occurs while WDF1 flag is "1," WDF2 flag is set to "1" and the watchdog reset signal is output.
⑤ The output transistor of RESET pin is turned "ON" by the watchdog reset signal and system reset is executed.

Note: The number of count is equal to the number of machine cycle because the count source of watchdog timer is the instruction clock.

**Fig. 28 Watchdog timer function**

When the watchdog timer is used, clear the WDF1 flag at the period of 65534 machine cycles or less with the WRST instruction.
When the watchdog timer is not used, execute the DWDT instruction and the WRST instruction continuously (refer to Figure 29).
The watchdog timer is not stopped with only the DWDT instruction.
The contents of WDF1 flag and timer WDT are initialized at the RAM back-up mode.
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 state (refer to Figure 30)
Also, set the NOP instruction after the WRST instruction, for the case when a skip is performed with the WRST instruction.

```
    ⋮
WRST        ; WDF1 flag cleared
    ⋮
DI
DWDT        ; Watchdog timer function enabled/disabled
WRST        ; WEF and WDF1 flags cleared
    ⋮
```

**Fig. 29 Program example to start/stop watchdog timer**

```
    ⋮
WRST        ; WDF1 flag cleared
NOP
DI          ; Interrupt disabled
EPOF        ; POF instruction enabled
POF         ; RAM back-up mode
  ↓
Oscillation stop
    ⋮
```

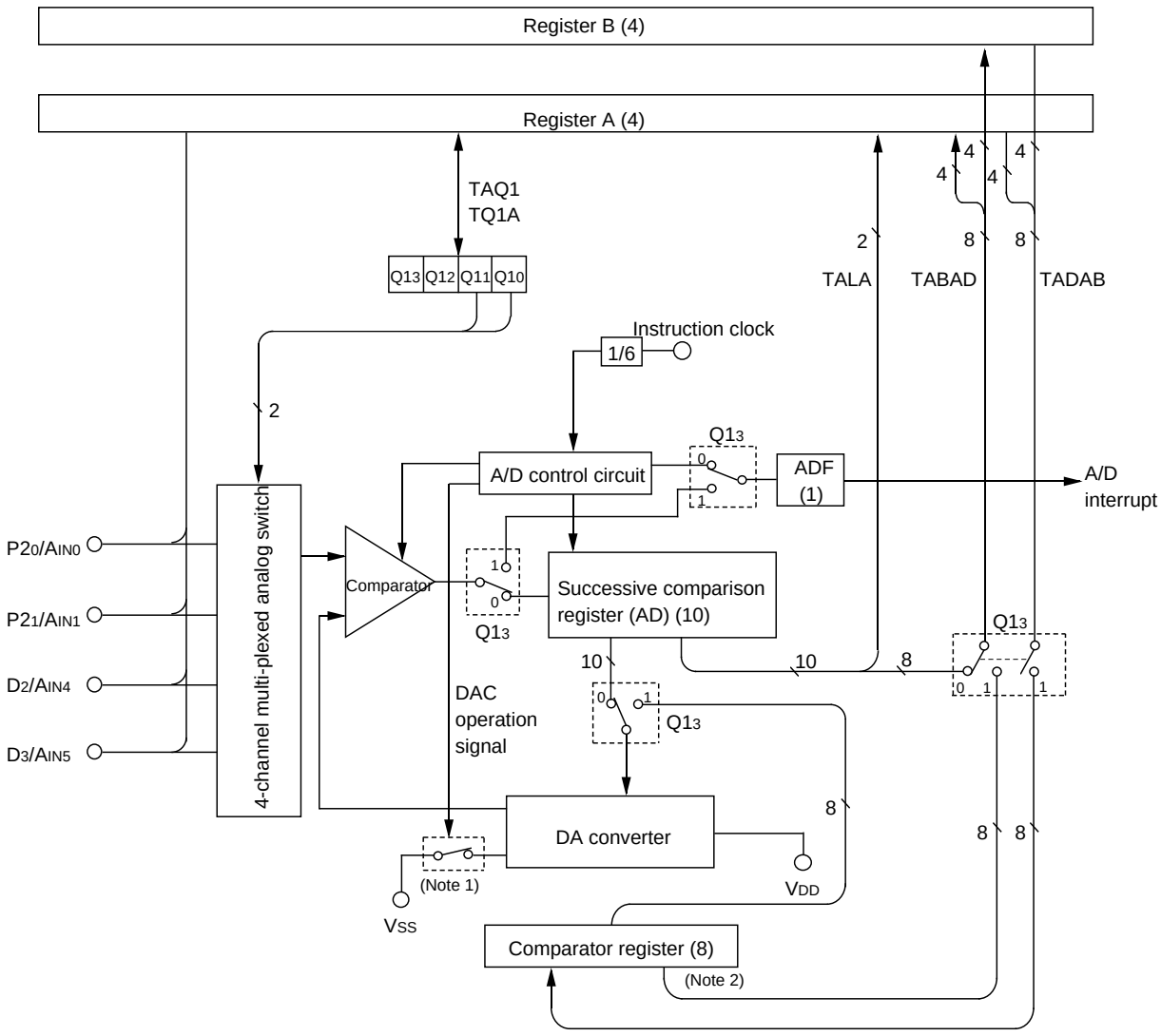
**Fig. 30 Program example to enter the RAM back-up mode when using the watchdog timer**

## A/D CONVERTER

The 4508 Group has a built-in A/D conversion circuit that performs conversion by 10-bit successive comparison method. Table 11 shows the characteristics of this A/D converter. This A/D converter can also be used as an 8-bit comparator to compare analog voltages input from the analog input pin with preset values.

**Table 11 A/D converter characteristics**

| Parameter | Characteristics |
|---|---|
| Conversion format | Successive comparison method |
| Resolution | 10 bits |
| Relative accuracy | Linearity error: ±2LSB ($V_{DD}$=2.7 to 5.5 V) |
| | Differential non-linearity error: ±0.9LSB ($V_{DD}$=2.7 to 5.5 V) |
| Conversion speed | 31 $\mu$s (f($X_{IN}$)=6 MHz, f(STCK)=f($X_{IN}$)) |
| Analog input pin | 4 |




**Notes 1:** This switch is turned ON only when A/D converter is operating and generates the comparison voltage.

**2:** Writing/reading data to the comparator register is possible only in the comparator mode (Q13=1). The value of the comparator register is retained even when the mode is switched to the A/D conversion mode (Q13=0) because it is separated from the successive comparison register (AD). Also, the resolution in the comparator mode is 8 bits because the comparator register consists of 8 bits.

**Fig. 31 A/D conversion circuit structure**

**Table 12 A/D control registers**

| A/D control register Q1 | | at reset : $0000_2$ | | | at RAM back-up : state retained | R/W TAQ1/TQ1A |
|---|---|---|---|---|---|---|
| $Q1_3$ | A/D operation mode selection bit | 0 | A/D conversion mode | | | |
| | | 1 | Comparator mode | | | |
| $Q1_2$ | Analog input pin selection bits | $Q1_2$ | $Q1_1$ | $Q1_0$ | Selected pins | |
| | | 0 | 0 | 0 | $A_{IN0}$ | |
| | | 0 | 0 | 1 | $A_{IN1}$ | |
| $Q1_1$ | | 0 | 1 | 0 | Not available | |
| | | 0 | 1 | 1 | Not available | |
| | | 1 | 0 | 0 | $A_{IN4}$ | |
| $Q1_0$ | | 1 | 0 | 1 | $A_{IN5}$ | |
| | | 1 | 1 | 0 | Not available | |
| | | 1 | 1 | 1 | Not available | |

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

## (1) A/D control register Q1

Register Q1 is used to select the operation mode and one of analog input pins. Set the contents of this register through register A with the TQ1A instruction. The TAQ1 instruction can be used to transfer the contents of register Q1 to register A.

## (2) Operating at A/D conversion mode

The A/D conversion mode is set by setting the bit 3 of register Q1 to "0."

## (3) Successive comparison register AD

Register AD stores the A/D conversion result of an analog input in 10-bit digital data format. The contents of the high-order 8 bits of this register can be stored in register B and register A with the TABAD instruction. The contents of the low-order 2 bits of this register can be stored into the high-order 2 bits of register A with the TALA instruction. However, do not execute these instructions during A/D conversion.

When the contents of register AD is n, the logic value of the comparison voltage $V_{ref}$ generated from the built-in DA converter can be obtained with the reference voltage $V_{DD}$ by the following formula:

Logic value of comparison voltage $V_{ref}$

$$V_{ref} = \frac{V_{DD}}{1024} \times n$$

n: The value of register AD (n = 0 to 1023)

## (4) A/D conversion completion flag (ADF)

A/D conversion completion flag (ADF) is set to "1" when A/D conversion completes. The state of ADF flag can be examined with the skip instruction (SNZAD). Use the interrupt control register V2 to select the interrupt or the skip instruction.

The ADF flag is cleared to "0" when the interrupt occurs or when the next instruction is skipped with the skip instruction.

## (5) A/D conversion start instruction (ADST)

A/D conversion starts when the ADST instruction is executed. The conversion result is automatically stored in the register AD.

## (6) Operation description

A/D conversion is started with the A/D conversion start instruction (ADST). The internal operation during A/D conversion is as follows:

① When the A/D conversion starts, the register AD is cleared to "$000_{16}$."
② Next, the topmost bit of the register AD is set to "1," and the comparison voltage $V_{ref}$ is compared with the analog input voltage $V_{IN}$.
③ When the comparison result is $V_{ref} < V_{IN}$, the topmost bit of the register AD remains set to "1." When the comparison result is $V_{ref} > V_{IN}$, it is cleared to "0."

The 4508 Group repeats this operation to the lowermost bit of the register AD to convert an analog value to a digital value. A/D conversion stops after 62 machine cycles (31 $\mu$s when f($X_{IN}$) = 6.0 MHz in high-speed mode) from the start, and the conversion result is stored in the register AD. An A/D interrupt activated condition is satisfied and the ADF flag is set to "1" as soon as A/D conversion completes (Figure 32).

**Table 13 Change of successive comparison register AD during A/D conversion**

| At starting conversion | Change of successive comparison register AD | Comparison voltage ($V_{ref}$) value |
|---|---|---|
| 1st comparison | 1 0 0 ----- 0 0 0 | $\frac{V_{DD}}{2}$ |
| 2nd comparison | ✻1 1 0 ----- 0 0 0 | $\frac{V_{DD}}{2} \pm \frac{V_{DD}}{4}$ |
| 3rd comparison | ✻1 ✻2 1 ----- 0 0 0 | $\frac{V_{DD}}{2} \pm \frac{V_{DD}}{4} \pm \frac{V_{DD}}{8}$ |
| After 10th comparison completes | A/D conversion result<br>✻1 ✻2 ✻3 ----- ✻8 ✻9 ✻A | $\frac{V_{DD}}{2} \pm \cdots \pm \frac{V_{DD}}{1024}$ |

✻1: 1st comparison result ✻2: 2nd comparison result
✻3: 3rd comparison result ✻8: 8th comparison result
✻9: 9th comparison result ✻A: 10th comparison result

## (7) A/D conversion timing chart

Figure 32 shows the A/D conversion timing chart.

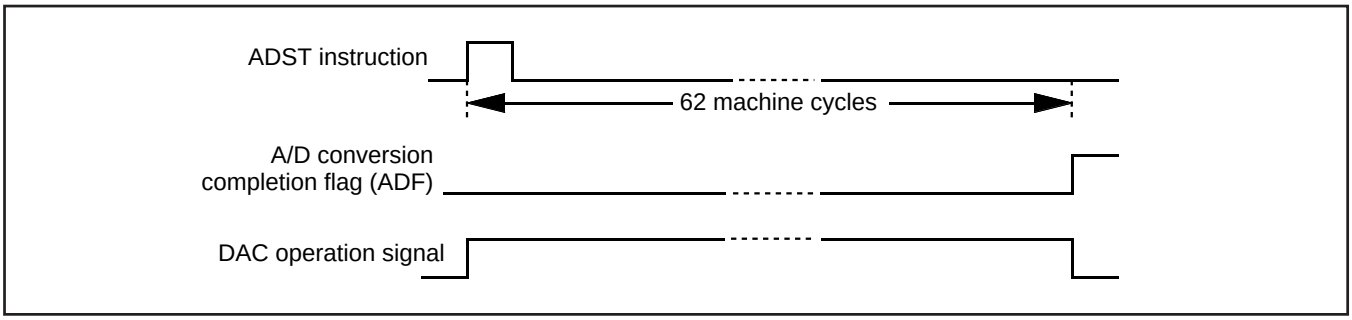



**Fig. 32 A/D conversion timing chart**

## (8) How to use A/D conversion

How to use A/D conversion is explained using as example in which the analog input from P2$_0$/AIN0 pin is A/D converted, and the high-order 4 bits of the converted data are stored in address M(Z, X, Y) = (0, 0, 0), the middle-order 4 bits in address M(Z, X, Y) = (0, 0, 1), and the low-order 2 bits in address M(Z, X, Y) = (0, 0, 2) of RAM. The A/D interrupt is not used in this example.

① Select the AIN0 pin function and A/D conversion mode with the register Q1 (refer to Figure 33).
② Execute the ADST instruction and start A/D conversion.
③ Examine the state of ADF flag with the SNZAD instruction to determine the end of A/D conversion.
④ Transfer the low-order 2 bits of converted data to the high-order 2 bits of register A (TALA instruction).
⑤ Transfer the contents of register A to M (Z, X, Y) = (0, 0, 2).
⑥ Transfer the high-order 8 bits of converted data to registers A and B (TABAD instruction).
⑦ Transfer the contents of register A to M (Z, X, Y) = (0, 0, 1).
⑧ Transfer the contents of register B to register A, and then, store into M(Z, X, Y) = (0, 0, 0).

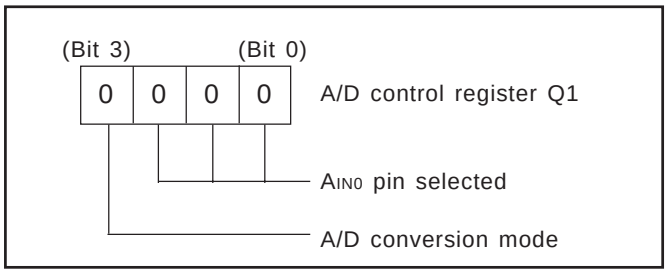



**Fig. 33 Setting registers**

## (9) Operation at comparator mode

The A/D converter is set to comparator mode by setting bit 3 of the register Q1 to “1.”

Below, the operation at comparator mode is described.

## (10) Comparator register

In comparator mode, the built-in DA comparator is connected to the 8-bit comparator register as a register for setting comparison voltages. The contents of register B is stored in the high-order 4 bits of the comparator register and the contents of register A is stored in the low-order 4 bits of the comparator register with the TADAB instruction.

When changing from A/D conversion mode to comparator mode, the result of A/D conversion (register AD) is undefined.

However, because the comparator register is separated from register AD, the value is retained even when changing from comparator mode to A/D conversion mode. Note that the comparator register can be written and read at only comparator mode.

If the value in the comparator register is n, the logic value of comparison voltage $V_{ref}$ generated by the built-in DA converter can be determined from the following formula:

Logic value of comparison voltage $V_{ref}$

$$V_{ref} = \frac{V_{DD}}{256} \times n$$

n: The value of register AD (n = 0 to 255)

## (11) Comparison result store flag (ADF)

In comparator mode, the ADF flag, which shows completion of A/D conversion, stores the results of comparing the analog input voltage with the comparison voltage. When the analog input voltage is lower than the comparison voltage, the ADF flag is set to “1.” The state of ADF flag can be examined with the skip instruction (SNZAD). Use the interrupt control register V2 to select the interrupt or the skip instruction.

The ADF flag is cleared to “0” when the interrupt occurs or when the next instruction is skipped with the skip instruction.

## (12) Comparator operation start instruction (ADST instruction)

In comparator mode, executing ADST starts the comparator operating.

The comparator stops 8 machine cycles after it has started (6 $\mu$s at f($X_{IN}$) = 4.0 MHz in high-speed through mode). When the analog input voltage is lower than the comparison voltage, the ADF flag is set to “1.”

## (13) Notes for the use of A/D conversion 1

- TALA instruction

  When the TALA instruction is executed, the low-order 2 bits of register AD is transferred to the high-order 2 bits of register A, simultaneously, the low-order 2 bits of register A is “0.”

- Operating mode of A/D converter

  Do not change the operating mode (both A/D conversion mode and comparator mode) of A/D converter with the bit 3 of register Q1 while the A/D converter is operating.

  Clear the bit 2 of register V2 to “0” to change the operating mode from the comparator mode to A/D conversion mode.

  The A/D conversion completion flag (ADF) may be set when the operating mode of the A/D converter is changed from the comparator mode to the A/D conversion mode. Accordingly, set a value to the bit 3 of register Q1, and execute the SNZAD instruction to clear the ADF flag.

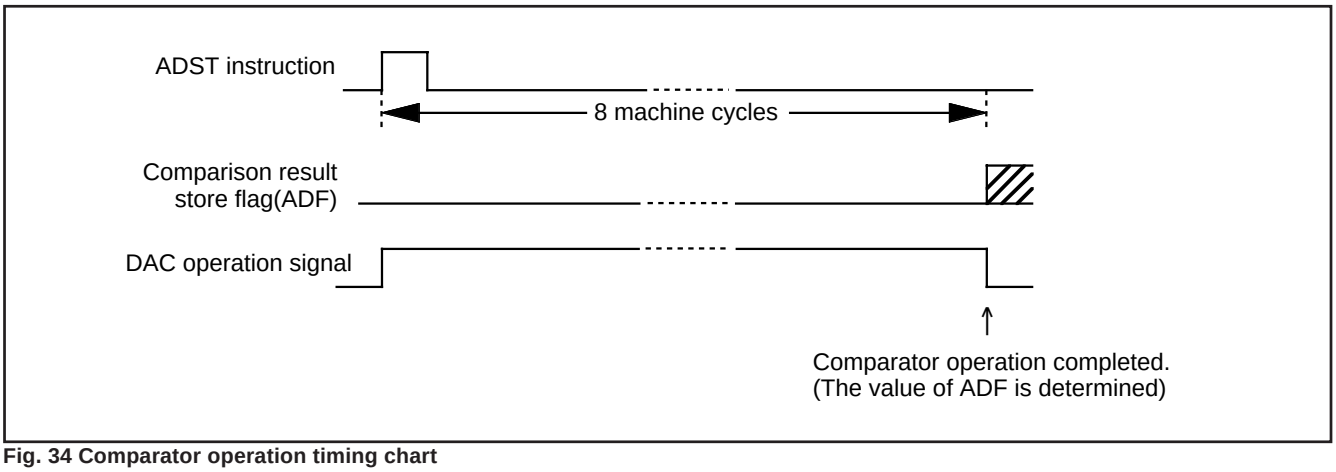



**Fig. 34 Comparator operation timing chart**

## (14) Definition of A/D converter accuracy

The A/D conversion accuracy is defined below (refer to Figure 35).

- Relative accuracy
  - ① Zero transition voltage ($V_{0T}$)
    This means an analog input voltage when the actual A/D conversion output data changes from “0” to “1.”
  - ② Full-scale transition voltage ($V_{FST}$)
    This means an analog input voltage when the actual A/D conversion output data changes from “1023” to ”1022.”
  - ③ Linearity error
    This means a deviation from the line between $V_{0T}$ and $V_{FST}$ of a converted value between $V_{0T}$ and $V_{FST}$.
  - ④ Differential non-linearity error
    This means a deviation from the input potential difference required to change a converter value between $V_{0T}$ and $V_{FST}$ by 1 LSB at the relative accuracy.

- Absolute accuracy
  This means a deviation from the ideal characteristics between 0 to $V_{DD}$ of actual A/D conversion characteristics.

Vn: Analog input voltage when the output data changes from “n” to “n+1” (n = 0 to 1022)

- 1LSB at relative accuracy → $\frac{V_{FST}-V_{0T}}{1022}$ (V)

- 1LSB at absolute accuracy → $\frac{V_{DD}}{1024}$ (V)

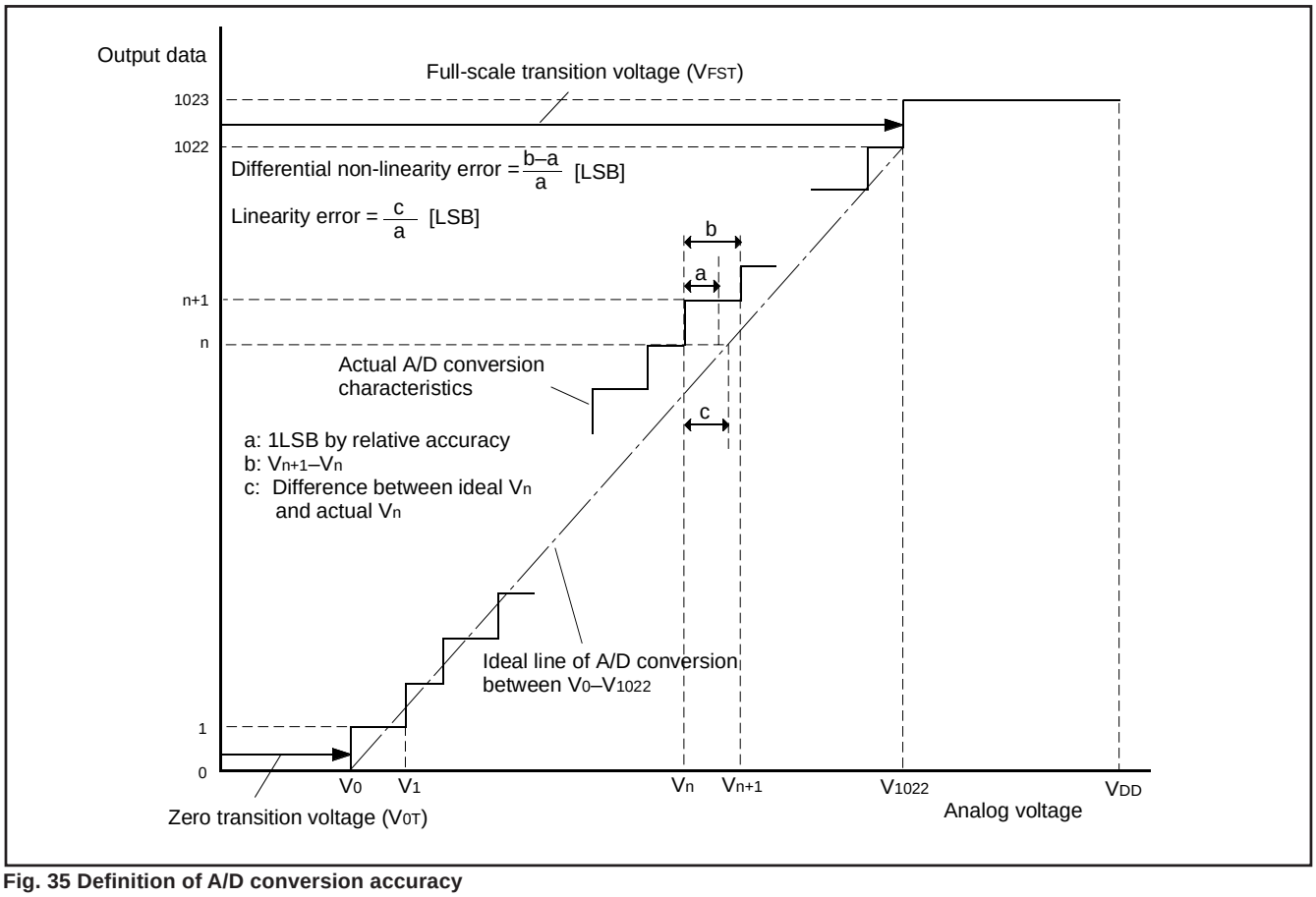



**Fig. 35 Definition of A/D conversion accuracy**

## SERIAL INTERFACE

The 4508 Group has a built-in clock synchronous serial interface which can serially transmit or receive 8-bit data.

Serial interface consists of;

- Serial interface register SI
- Serial interface control register J1
- Serial interface transmit/receive completion flag (SIOF)
- Serial interface counter

Registers A and B are used to perform data transfer with internal CPU.

The pin functions of the serial interface pins can be set with the register J1.

**Table 14 Serial interface pins**

| Pin | Pin function when selecting serial interface |
|---|---|
| P02/SCK | Clock I/O (SCK) |
| P01/SOUT | Serial data output (SOUT) |
| P03/SIN | Serial data input (SIN) |

Note: Even when the SIN pin function is used, the I/O of port P00 is valid.
Even when the SOUT pin function is used, the input of port P01 is valid.
Even when the SCK pin function is used, the input of P02 is valid.
Be careful when using inputs of both SCK and P02 since the input threshold value of SCK pin is different from that of port P02.

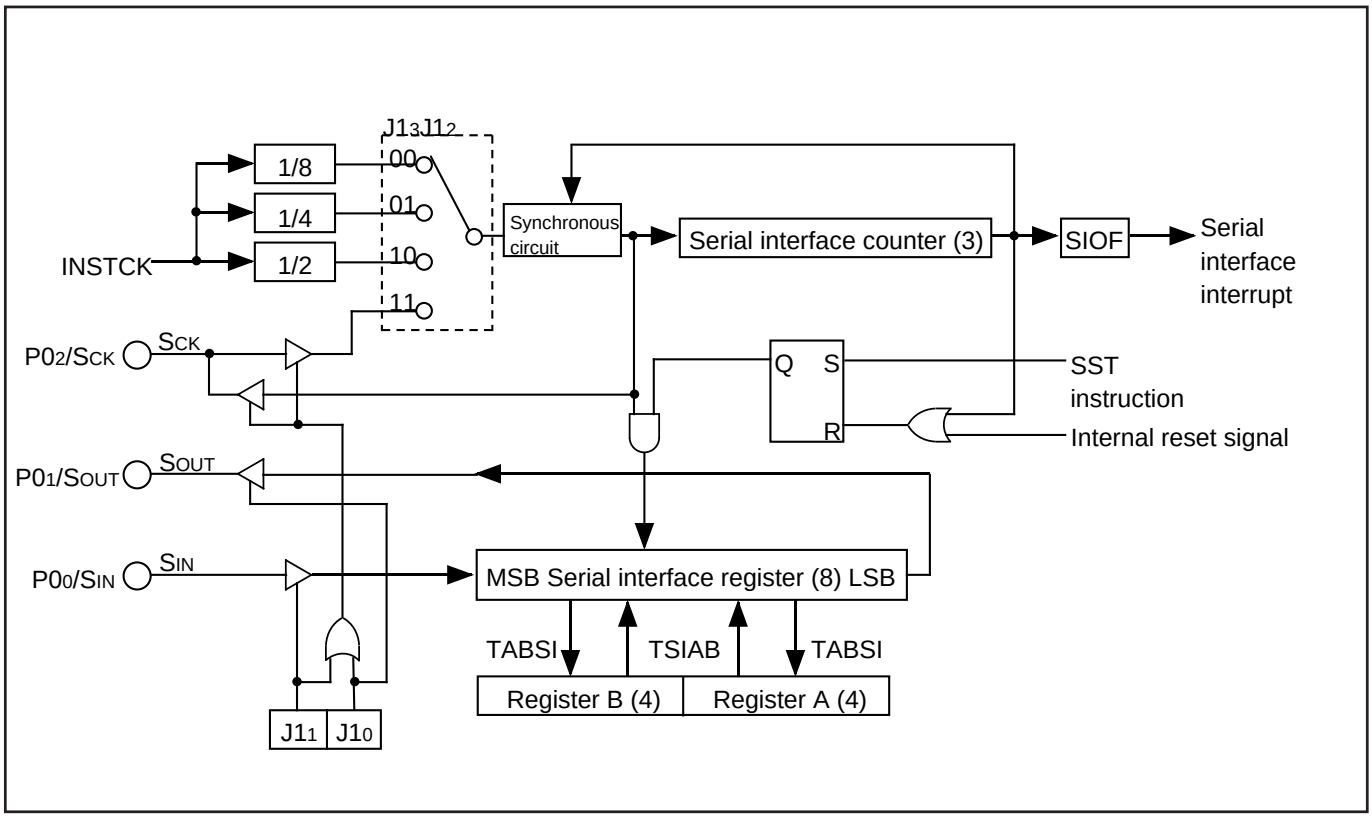



**Fig. 36 Serial interface structure**

**Table 15 Serial interface control register**

| Serial interface control register J1 | | at reset : 00002 | | at RAM back-up : state retained | R/W TAJ1/TJ1A |
|---|---|---|---|---|---|
| J13 | Serial interface synchronous clock selection bits | J13 | J12 | Synchronous clock | |
| | | 0 | 0 | Instruction clock (INSTCK) divided by 8 | |
| J12 | | 0 | 1 | Instruction clock (INSTCK) divided by 4 | |
| | | 1 | 0 | Instruction clock (INSTCK) divided by 2 | |
| | | 1 | 1 | External clock (SCK input) | |
| J11 | Serial interface port function selection bits | J11 | J10 | Port function | |
| | | 0 | 0 | P00, P01,P02 selected/SIN, SOUT, SCK not selected | |
| J10 | | 0 | 1 | P00, SOUT, SCK selected/SIN, P01, P02 not selected | |
| | | 1 | 0 | SIN, P01, SCK selected/P00, SOUT, P02 not selected | |
| | | 1 | 1 | SIN, SOUT, SCK selected/P00, P01,P02 not selected | |

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

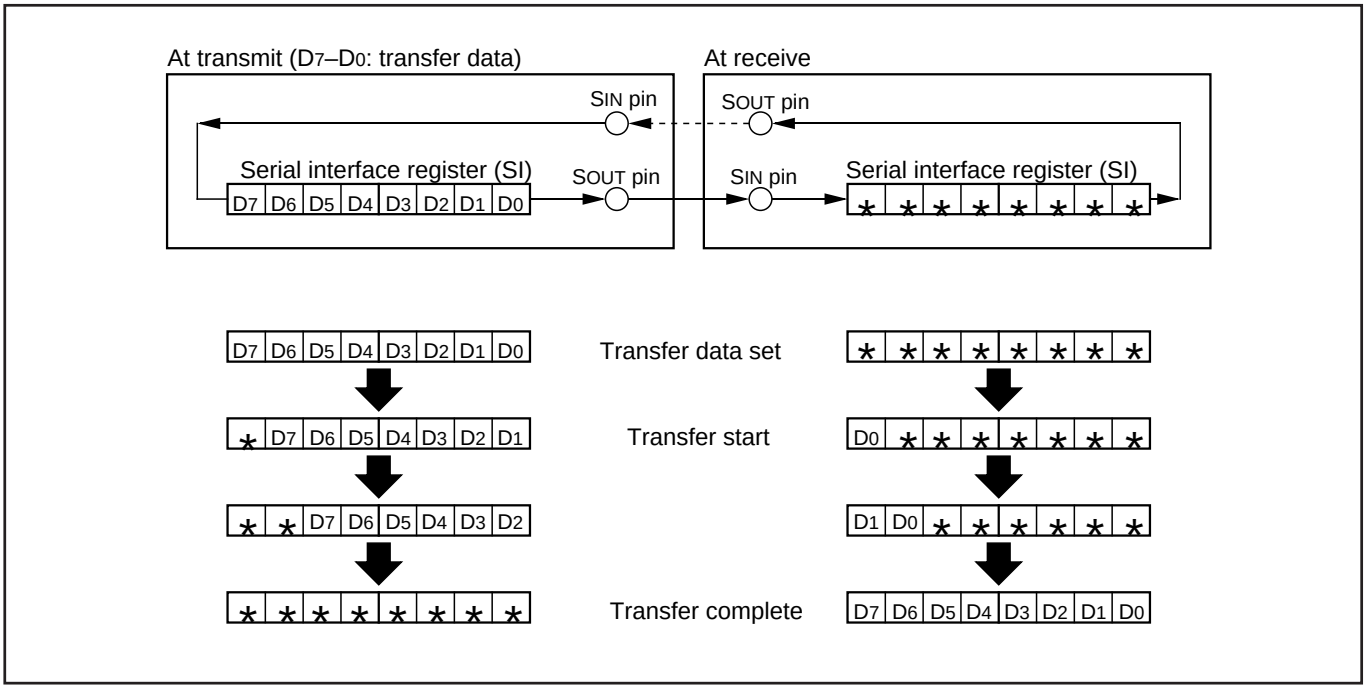



**Fig. 37 Serial interface register state when transferring**

## (1) Serial interface register SI

Serial interface register SI is the 8-bit data transfer serial/parallel conversion register. Data can be set to register SI through registers A and B with the TSIAB instruction. The contents of register A is transmitted to the low-order 4 bits of register SI, and the contents of register B is transmitted to the high-order 4 bits of register SI.

During transmission, each bit data is transmitted LSB first from the lowermost bit (bit 0) of register SI, and during reception, each bit data is received LSB first to register SI starting from the topmost bit (bit 7).

When register SI is used as a work register without using serial interface, do not select the SCK pin.

## (2) Serial interface transmit/receive completion flag (SIOF)

Serial interface transmit/receive completion flag (SIOF) is set to "1" when serial data transmission or reception completes. The state of SIOF flag can be examined with the skip instruction (SNZSI). Use the interrupt control register V2 to select the interrupt or the skip instruction.

The SIOF flag is cleared to "0" when the interrupt occurs or when the next instruction is skipped with the skip instruction.

## (3) Serial interface start instruction (SST)

When the SST instruction is executed, the SIOF flag is cleared to "0" and then serial interface transmission/reception is started.

## (4) Serial interface control register J1

Register J1 controls the synchronous clock, P02/SCK, P01/SOUT and P00/SIN pin function. Set the contents of this register through register A with the TJ1A instruction. The TAJ1 instruction can be used to transfer the contents of register J1 to register A.

## (5) How to use serial interface

Figure 38 shows the serial interface connection example. Serial interface interrupt is not used in this example. In the actual wiring, pull up the wiring between each pin with a resistor. Figure 38 shows the data transfer timing and Table 16 shows the data transfer sequence.

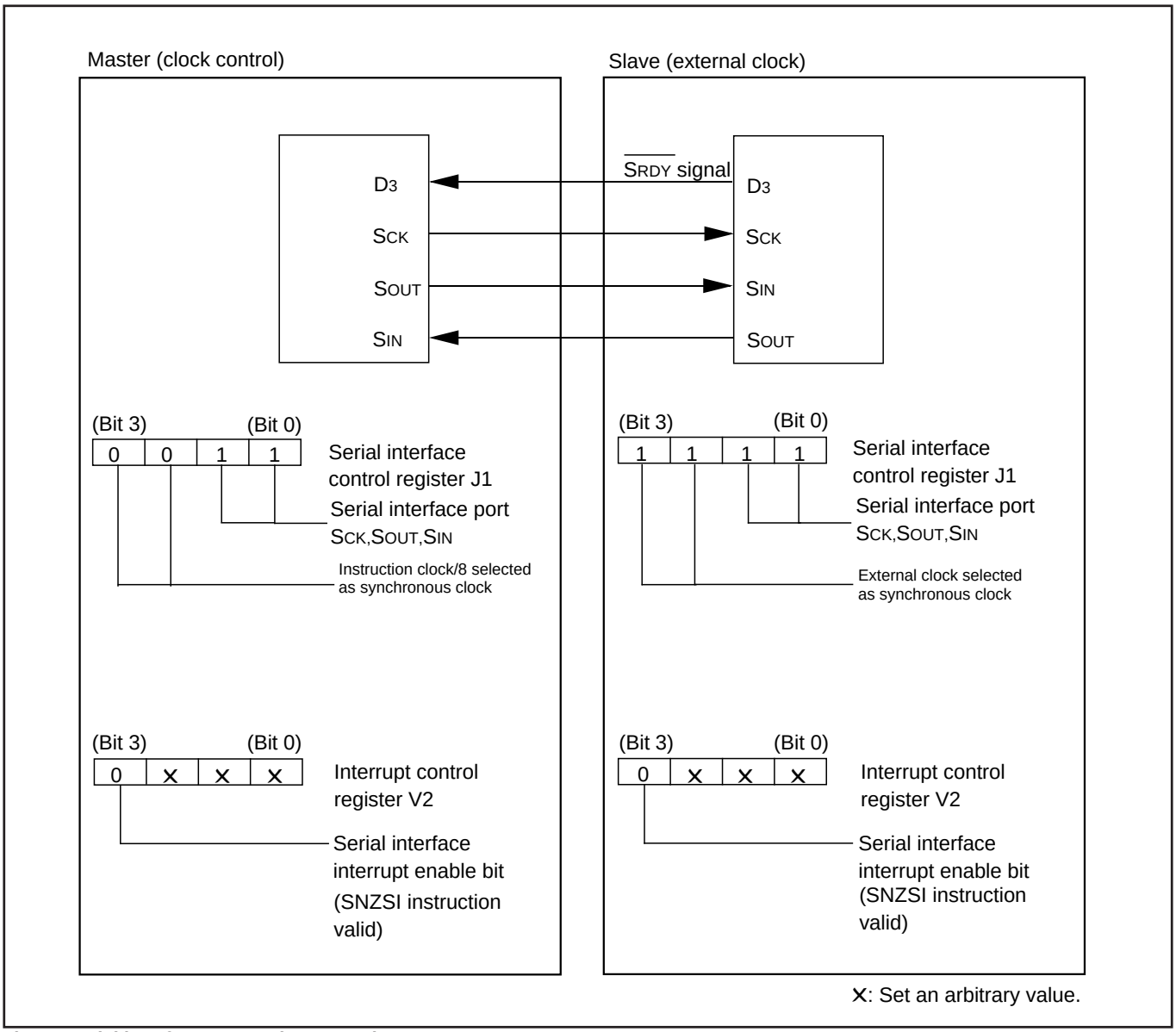



**Fig. 38 Serial interface connection example**

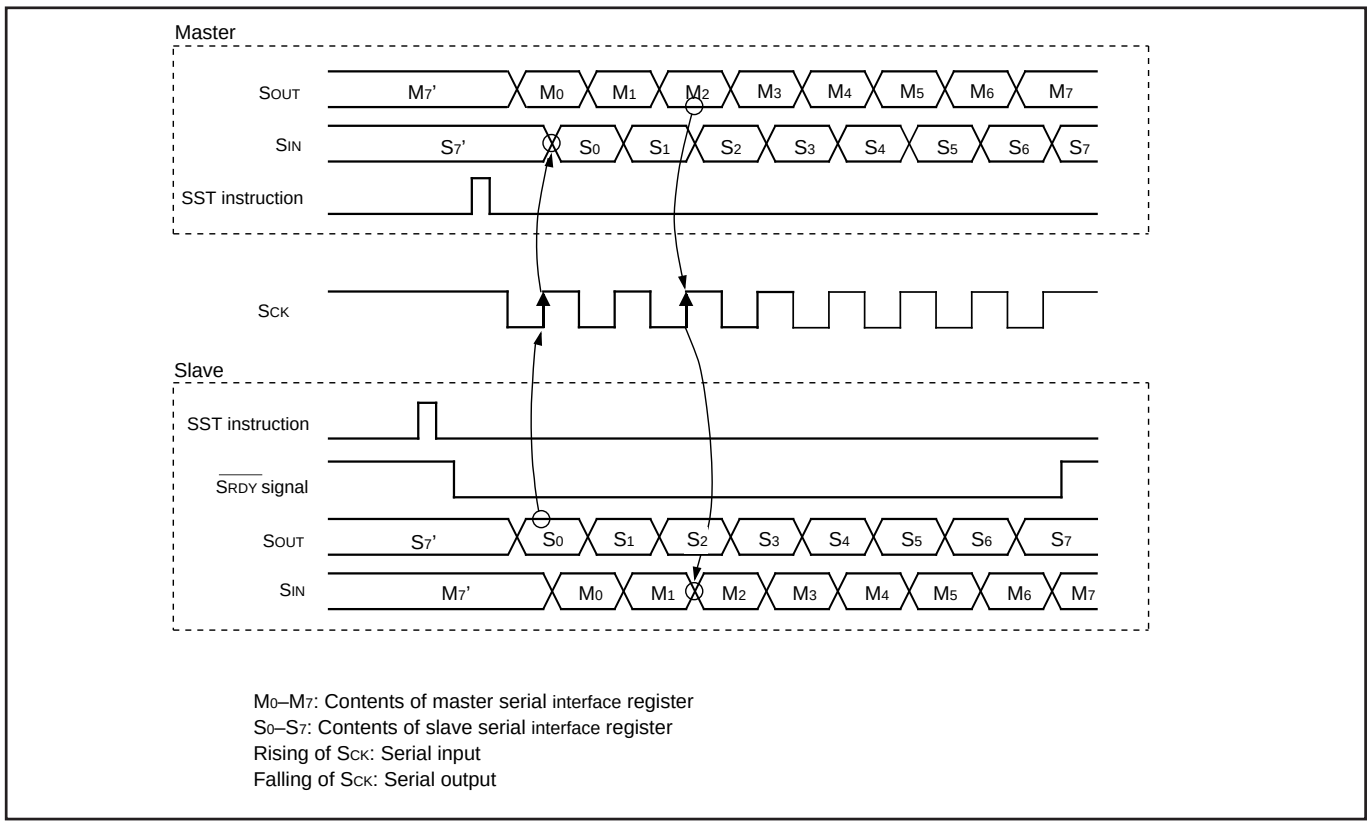



**Fig. 39 Timing of serial interface data transfer**

**Table 16 Processing sequence of data transfer from master to slave**

| Master (transmission) | Slave (reception) |
|---|---|
| [Initial setting]<br>• Setting the serial interface control register J1 and interrupt control register V2 shown in Figure 38. | [Initial setting]<br>• Setting serial interface control register J1, and interrupt control register V2 shown in Figure 38. |
| TJ1A and TV2A instructions | TJ1A and TV2A instructions |
| • Setting the port received the reception enable signal ($\overline{SRDY}$) to the input mode.<br>(Port D3 is used in this example) | • Setting the port transmitted the reception enable signal ($\overline{SRDY}$) and outputting "H" level.<br>(Port D3 is used in this example) |
| SD instruction | SD instruction |
| * [Transmission enable state]<br>• Storing transmission data to serial interface register SI. | *[Reception enable state]<br>• The SIOF flag is cleared to "0." |
| TSIAB instruction | SST instruction |
| | • "L" level (reception possible) is output from port D3. |
| | RD instruction |
| [Transmission]<br>•Check port D3 is "L" level. | [Reception] |
| SZD instruction | |
| •Serial transfer starts. | |
| SST instruction | |
| •Check transmission completes. | • Check reception completes. |
| SNZSI instruction | SNZSI instruction |
| •Wait (timing when continuously transferring) | • "H" level is output from port D3. |
| | SD instruction |
| | [Data processing] |

1-byte data is serially transferred on this process. Subsequently, data can be transferred continuously by repeating the process from *.

When an external clock is selected as a synchronous clock, control the clock externally because serial transfer is performed as long as clock is externally input. (Unlike an internal clock, an external clock is not stopped when serial transfer is completed.) However, the SIOF flag is set to "1" when the clock is counted 8 times after executing the SST instruction. Be sure to set the initial level of the external clock to "H."

## RESET FUNCTION

System reset is performed by the followings:

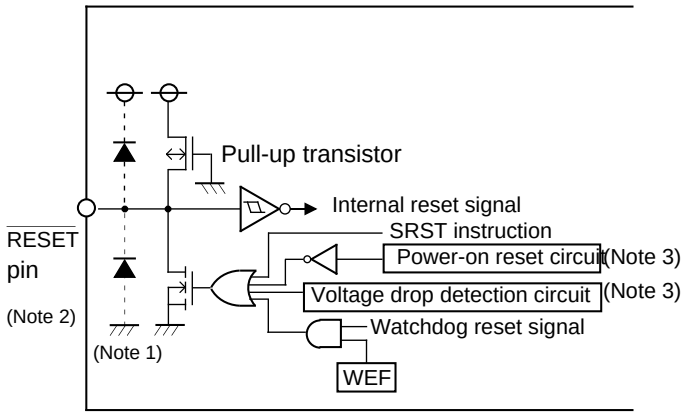
- "L" level is applied to the $\overline{\text{RESET}}$ pin externally,
- System reset instruction (SRST) is executed,
- Reset occurs by watchdog timer,
- Reset occurs by built-in power-on reset (only for H version)
- Reset occurs by voltage drop detection circuit (only for H version)

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

### (1) $\overline{\text{RESET}}$ pin input

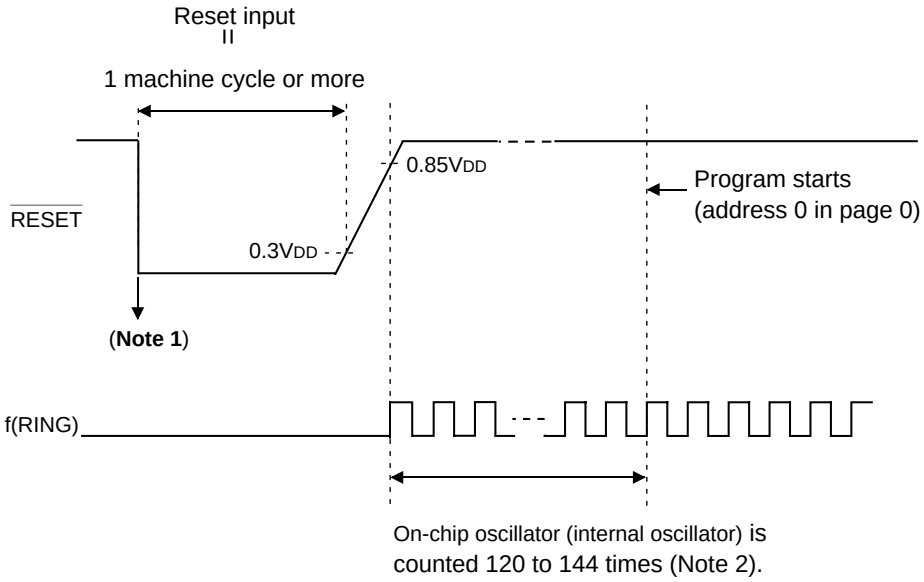
System reset is performed certainly by applying "L" level to $\overline{\text{RESET}}$ pin for 1 machine cycle or more when the following condition is satisfied;

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




**Notes 1:** This symbol represents a parasitic diode.
**2:** Applied potential to $\overline{\text{RESET}}$ pin must be $V_{DD}$ or less.
**3:** These are equipped with only H version.

**Fig. 40 Structure of reset pin and its peripherals**




**Notes 1:** Keep the value of supply voltage to the minimum value or more of the recommended operating conditions.
**2:** It depends on the internal state at reset.

**Fig. 41 $\overline{\text{RESET}}$ pin input waveform and reset release timing**

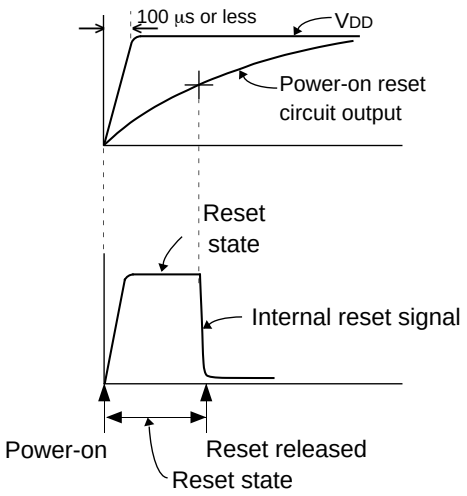
## (2) Power-on reset (only for H version)

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, set the time for the supply voltage to rise from 0 V to the minimum voltage of recommended operating conditions to 100 μs or less.

If the rising time exceeds 100 μs, 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.

## (3) System reset instruction (SRST)

By executing the SRST instruction, "L" level is output to $\overline{\text{RESET}}$ pin and system reset is performed.




Note: Keep the value of supply voltage to the minimum value or more of the recommended operating conditions.

**Fig. 42 Power-on reset operation**

**Table 17 Port state at reset**

| Name | Function | State |
|---|---|---|
| $D_0$, $D_1$ | $D_0$, $D_1$ | High-impedance (Notes 1, 2) |
| $D_2$/$A_{IN4}$, $D_3$/$A_{IN5}$ | $D_2$, $D_3$ | High-impedance (Notes 1, 2, 3) |
| $P0_0$/$S_{IN}$, $P0_1$/$S_{OUT}$, $P0_2$/$S_{CK}$ | $P0_0$, $P0_1$, $P0_2$ | High-impedance (Notes 1, 2, 3) |
| $P0_3$ | $P0_3$ | High-impedance (Notes 1, 2, 3) |
| $P1_0$ | $P1_0$ | High-impedance (Notes 1, 2, 3) |
| $P1_1$/CNTR1 | $P1_1$ | High-impedance (Notes 1, 2, 3) |
| $P1_2$/CNTR0 | $P1_2$ | High-impedance (Notes 1, 2, 3) |
| $P1_3$/INT | $P1_3$ | High-impedance (Notes 1, 2, 3) |
| $P2_0$/$A_{IN0}$, $P2_1$/$A_{IN1}$ | $P2_0$, $P2_1$ | High-impedance (Notes 1, 2, 3) |

Notes 1: Output latch is set to "1."
2: The output structure is N-channel open-drain.
3: Pull-up transistor is turned OFF.

## (4) Internal state at reset

Figure 43 shows internal state at reset (they are the same after system is released from reset). The contents of timers, registers, flags and RAM except shown in Figure 43 are undefined, so set the initial value to them.

- 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
- Watchdog timer flags (WDF1, WDF2) ..... 0
- Watchdog timer enable flag (WEF) ..... 1
- Timer control register PA ..... 0 (Prescaler stopped)
- Timer control register W1 ..... 0 0 0 0 (Timer 1 stopped)
- Timer control register W2 ..... 0 0 0 0 (Timer 2 stopped)
- Timer control register W5 ..... 0 0 0 0
- Timer control register W6 ..... 0 0 0 0
- Clock control register MR ..... 1 1 0 1
- Clock control register RG ..... 0 (On-chip oscillator operating)
- Serial interface transmit/receive completion flag (SIOF) ..... 0
- Serial interface control register J1 ..... 0 0 0 0 (Serial interface port not selected)
- Serial interface register SI ..... X X X X X X X X
- A/D conversion completion flag (ADF) ..... 0
- A/D control register Q1 ..... 0 0 0 0
- Successive comparison register AD ..... X X X X X X X X X X
- Comparator register ..... X X X X X X X X
- Key-on wakeup control register K0 ..... 0 0 0 0
- Key-on wakeup control register K1 ..... 0 0 0 0
- Key-on wakeup control register K2 ..... 0 0 0 0
- Key-on wakeup control register L1 ..... 0 0 0 0
- Pull-up control register PU0 ..... 0 0 0 0
- Pull-up control register PU1 ..... 0 0 0 0
- Pull-up control register PU2 ..... 0 0 0 0
- Port output structure control register FR0 ..... 0 0 0 0
- Port output structure control register FR1 ..... 0 0 0 0
- Port output structure control register FR2 ..... 0 0 0 0
- Port output structure control register FR3 ..... 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
- Operation source clock ..... On-chip oscillator (operating)
- Ceramic resonator circuit ..... Operating
- RC oscillation circuit ..... Stop

"X" represents undefined.

**Fig. 43 Internal state at reset**

## VOLTAGE DROP DETECTION CIRCUIT (only for H version)

The built-in voltage drop detection circuit is designed to detect a drop in voltage and to reset the microcomputer by outputting "L" level to RESET pin if the supply voltage drops below a set value.

(1) SVDE instruction

If the SVDE instruction is not executed (initial state), the voltage drop detection circuit becomes invalid at RAM back-up mode.

When the SVDE instruction is executed, the voltage drop deteciton circuit is valid even after system enters into the RAM back-up mode.

The SVDE instruction can be executed only once.

In order to release the execution of the SVDE instruction, the system reset is required.

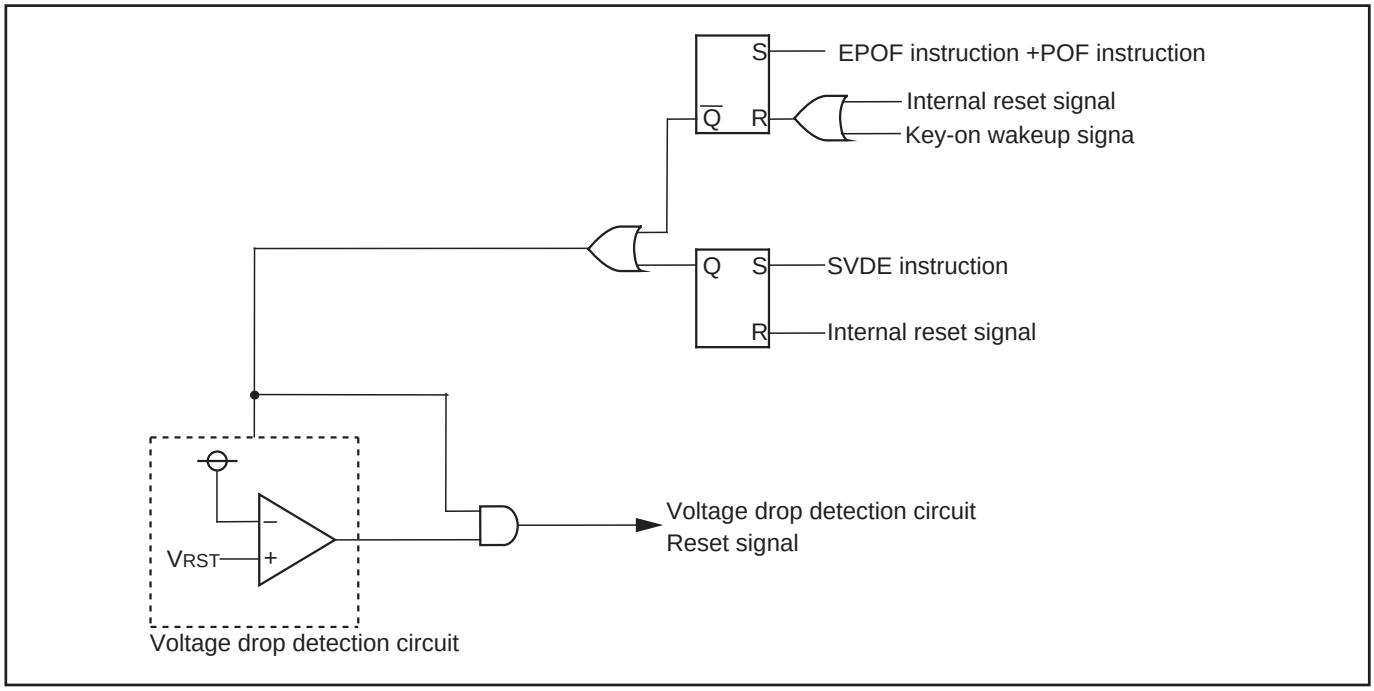



**Fig. 44 Voltage drop detection reset circuit**

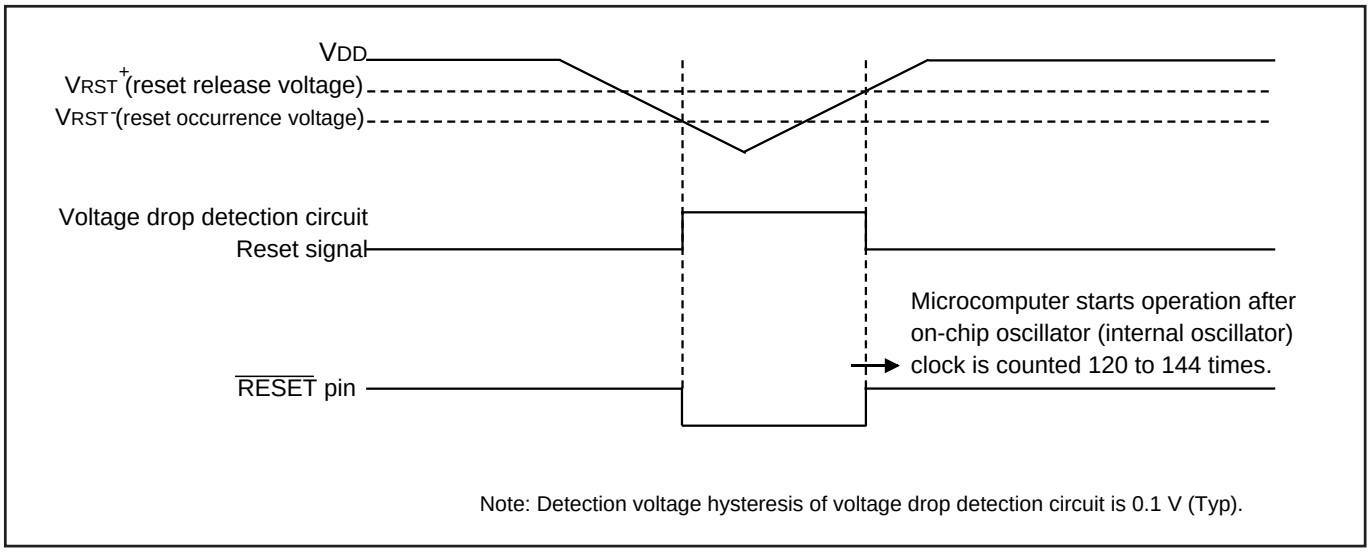



**Fig. 45 Voltage drop detection circuit operation waveform**

**Table 18 Voltage drop detection circuit operation state**

| | At CPU operating | At RAM back-up mode |
|---|---|---|
| SVDE instruction not executed | Valid | Invalid |
| SVDE instruction executed | Valid | Valid |

## RAM BACK-UP MODE

The 4508 Group has the RAM back-up mode.
When the POF instruction is executed continuously after the EPOF instruction, system enters the RAM back-up state.
The POF instruction is equal to the NOP instruction when the EPOF instruction is not executed before the POF instruction.
As oscillation stops retaining RAM, the function of reset circuit and states at RAM back-up mode, current dissipation can be reduced without losing the contents of RAM.
Table 19 shows the function and states retained at RAM back-up. Figure 46 shows the state transition.

### (1) Identification of the start condition

Warm start (return from the RAM back-up state) 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 state by executing the EPOF instruction and POF instruction continuously, the CPU starts executing the program from address 0 in page 0. In this case, the P flag is "1."

### (3) Cold start condition

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

- "L" level is applied to $\overline{\text{RESET}}$ pin,
- system reset (SRST) is performed,
- reset by watchdog timer is performed,
- reset by the built-in power-on reset circuit is performed (only for H version), or
- reset by the voltage drop detection circuit is performed (only for H version).

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

**Table 19 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 |
| Interrupt control registers V1, V2 | × |
| Interrupt control register I1 | O |
| Selected oscillation circuit (execution of CRCK) | O |
| Clock control register MR | × |
| Clock control register RG | × |
| Timer 1, Timer 2 function | (Note 3) |
| Watchdog timer function | × (Note 4) |
| Timer control register PA | × |
| Timer control registers W1, W2 | × |
| Timer control registers W5, W6 | O |
| Serial interface function | × |
| Serial interface control register J1 | O |
| A/D conversion function | × |
| A/D control register Q1 | O |
| Voltage drop detection circuit | (Note 5) |
| Port level | O |
| Key-on wakeup control registers K0 to K2, L1 | O |
| Pull-up control registers PU0 to PU2 | O |
| Port output structure control registers FR0 to FR3 | O |
| External interrupt request flag (EXF0) | × |
| Timer interrupt request flags (T1F, T2F) | (Note 3) |
| A/D conversion completion flag (ADF) | × |
| Serial interface transmit/receive completion flag (SIOF) | × |
| Interrupt enable flag (INTE) | × |
| Watchdog timer flags (WDF1, WDF2) | × (Note 4) |
| Watchdog timer enable flag (WEF) | × (Note 4) |

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 "7" at RAM back-up.
3: The state of the timer is undefined.
4: Initialize the watchdog timer flag WDF1 with the WRST instruction, and then set the system to be in the RAM back-up mode.
5: The voltage drop detection circuit is equipped with only H version. In the RAM back-up mode, when the SVDE instruction is not executed, the voltage drop detection circuit is invalid, and when the SVDE instruction is executed, the voltage drop detection circuit is valid.

## (4) Return signal

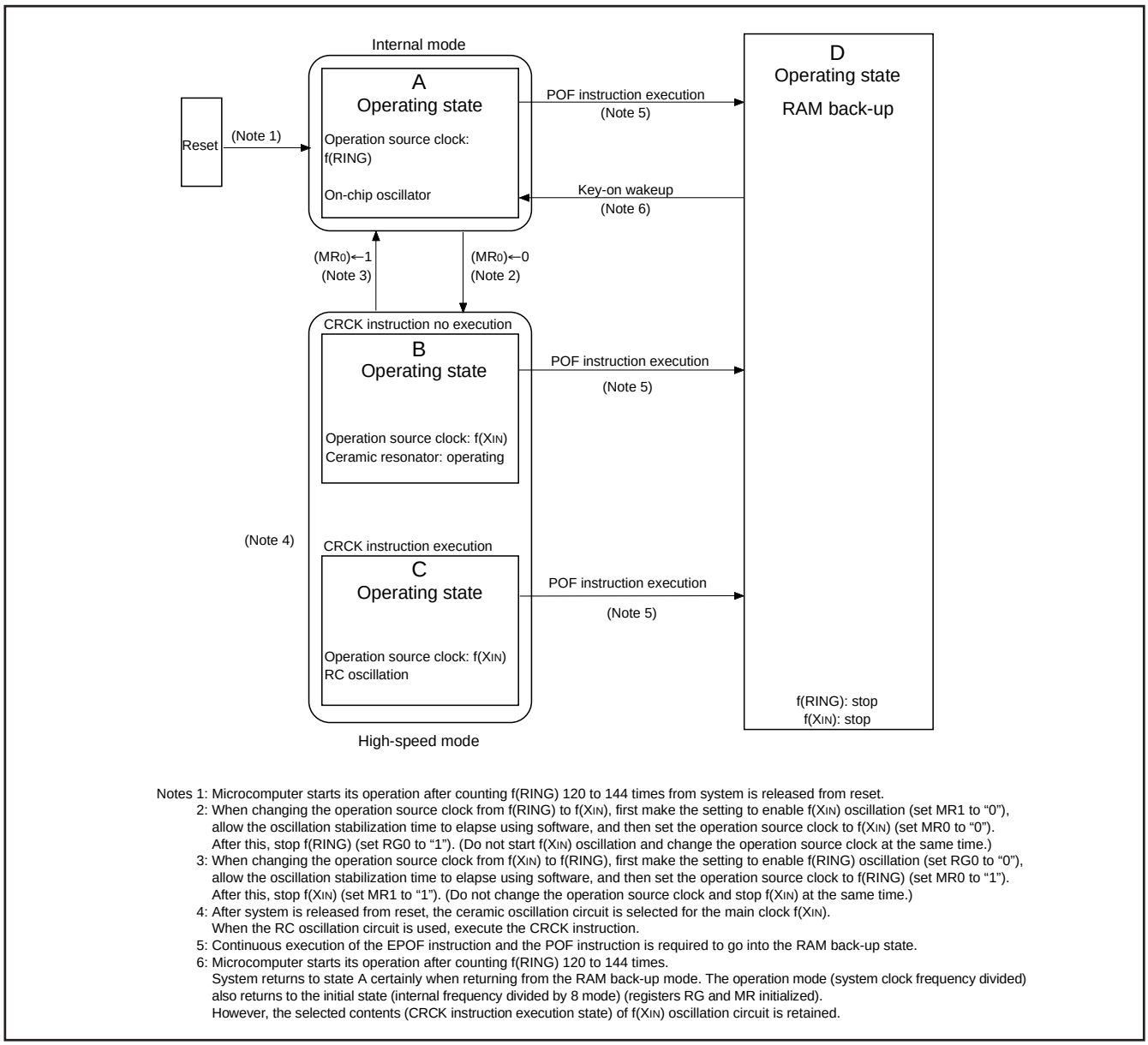
An external wakeup signal is used to return from the RAM back-up mode because the oscillation is stopped. Table 20 shows the return condition for each return source.

## (5) Control registers

- Key-on wakeup control register K0
  Register K0 controls the port P0 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.
- Key-on wakeup control register K1
  Register K1 controls the port P1 key-on wakeup function. Set the contents of this register through register A with the TK1A instruction. In addition, the TAK1 instruction can be used to transfer the contents of register K1 to register A.
- Key-on wakeup control register K2
  Register K2 controls the ports P2, $D_2$ and $D_3$ key-on wakeup function. Set the contents of this register through register A with the TK2A instruction. In addition, the TAK2 instruction can be used to transfer the contents of register K2 to register A.
- Key-on wakeup control register L1
  Register L1 controls the selection of the return condition and valid waveform/level of port P1, and the selection of the INT pin return condition and INT pin key-on wakeup function. Set the contents of this register through register A with the TL1A instruction. In addition, the TAL1 instruction can be used to transfer the contents of register L1 to register A.
- Pull-up control register PU0
  Register PU0 controls the ON/OFF of the port P0 pull-up transistor. Set the contents of this register through register A with the TPU0A instruction. In addition, the TAK1 instruction can be used to transfer the contents of register K0 to register A.
- Pull-up control register PU1
  Register PU1 controls the ON/OFF of the port P1 pull-up transistor. Set the contents of this register through register A with the TPU1A instruction. In addition, the TAPU1 instruction can be used to transfer the contents of register PU1 to register A.
- Pull-up control register PU2
  Register PU2 controls the ON/OFF of the ports P2, $D_2$ and $D_3$ pull-up transistor. Set the contents of this register through register A with the TPU2A instruction. In addition, the TAPU2 instruction can be used to transfer the contents of register PU2 to register A.
- Interrupt control register I1
  Register I1 controls the valid waveform/level of the external 0 interrupt and the input control of INT pin. Set the contents of this register through register A with the TI1A instruction. In addition, the TAI1 instruction can be used to transfer the contents of register I1 to register A.

**Table 20 Return source and return condition**

| Return source | | Return condition | Remarks |
|---|---|---|---|
| External wakeup signal | Port $P0_0$–$P0_3$<br>Port $P2_0$, $P2_1$<br>Port $D_2$, $D_3$ | Return by an external “L” level input. | The key-on wakeup function can be selected by one port unit. Set the port using the key-on wakeup function to “H” level before going into the RAM back-up state. |
| | Port $P1_0$–$P1_3$ | Return by an external “H” level or “L” level input, or falling edge (“H”→“L”) or rising edge (“L”→“H”). | The key-on wakeup function can be selected by one port unit. Select the return level (“L” level or “H” level) and return condition (level or edge) with the register L1 according to the external state before going into the RAM back-up state.<br>Before going into the RAM backup state, set an opposite level of the selected return level (edge) to the port using the key-on wakeup function. |
| | INT pin | Return by an external “H” level or “L” level input, or falling edge (“H”→“L”) or rising edge (“L”→“H”). When the return level is input, the EXF0 flag is not set. | The key-on wakeup function can be selected by one port unit. Select the return level (“L” level or “H” level) with the register I1 and return condition (level or edge) with the register L1 according to the external state before going into the RAM back-up state. |




Notes 1: Microcomputer starts its operation after counting f(RING) 120 to 144 times from system is released from reset.
2: When changing the operation source clock from f(RING) to f(XIN), first make the setting to enable f(XIN) oscillation (set MR1 to "0"), allow the oscillation stabilization time to elapse using software, and then set the operation source clock to f(XIN) (set MR0 to "0"). After this, stop f(RING) (set RG0 to "1"). (Do not start f(XIN) oscillation and change the operation source clock at the same time.)
3: When changing the operation source clock from f(XIN) to f(RING), first make the setting to enable f(RING) oscillation (set RG0 to "0"), allow the oscillation stabilization time to elapse using software, and then set the operation source clock to f(RING) (set MR0 to "1"). After this, stop f(XIN) (set MR1 to "1"). (Do not change the operation source clock and stop f(XIN) at the same time.)
4: After system is released from reset, the ceramic oscillation circuit is selected for the main clock f(XIN). When the RC oscillation circuit is used, execute the CRCK instruction.
5: Continuous execution of the EPOF instruction and the POF instruction is required to go into the RAM back-up state.
6: Microcomputer starts its operation after counting f(RING) 120 to 144 times. System returns to state A certainly when returning from the RAM back-up mode. The operation mode (system clock frequency divided) also returns to the initial state (internal frequency divided by 8 mode) (registers RG and MR initialized). However, the selected contents (CRCK instruction execution state) of f(XIN) oscillation circuit is retained.

**Fig. 46 State transition**

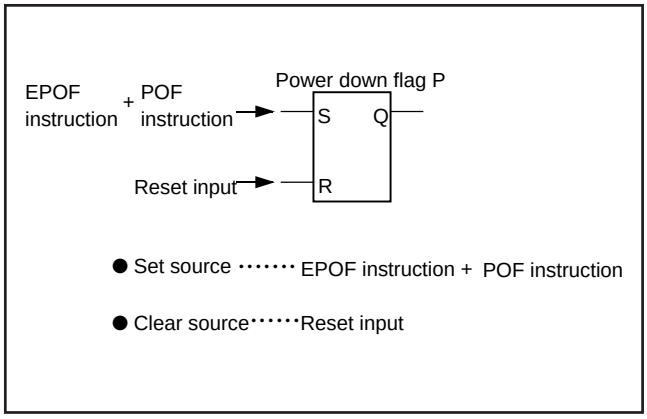



**Fig. 47 Set source and clear source of the P flag**

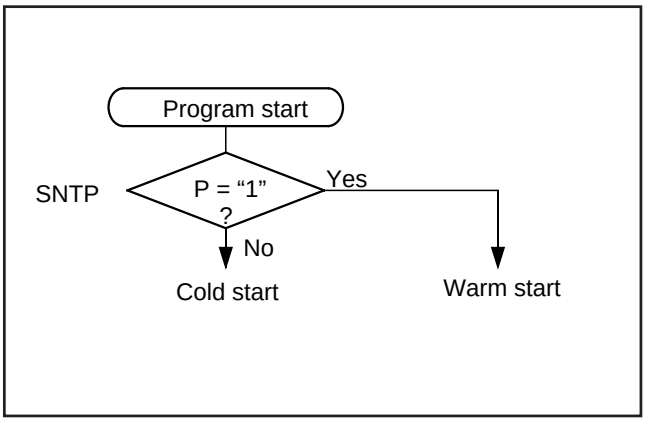



**Fig. 48 Start condition identified example using the SNZP instruction**

**Table 21 Key-on wakeup control register**

| Key-on wakeup control register K0 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W TAK0/TK0A |
|---|---|---|---|---|
| $K0_3$ | Port $P0_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K0_2$ | Port $P0_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K0_1$ | Port $P0_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K0_0$ | Port $P0_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |

| Key-on wakeup control register K1 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W TAK1/TK1A |
|---|---|---|---|---|
| $K1_3$ | Port $P1_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K1_2$ | Port $P1_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K1_1$ | Port $P1_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K1_0$ | Port $P1_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |

| Key-on wakeup control register K2 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W TAK2/TK2A |
|---|---|---|---|---|
| $K2_3$ | Port $D_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K2_2$ | Port $D_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K2_1$ | Port $P2_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |
| $K2_0$ | Port $P2_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |

| Key-on wakeup control register L1 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W TAL1/TL1A |
|---|---|---|---|---|
| $L1_3$ | Ports $P1_0$–$P1_3$ return condition selection bit | 0 | Return by level | |
| | | 1 | Return by edge | |
| $L1_2$ | Ports $P1_0$–$P1_3$ valid waveform/ level selection bit | 0 | Falling waveform/“L” level | |
| | | 1 | Rising waveform/“H” level | |
| $L1_1$ | INT pin return condition selection bit | 0 | Return by level | |
| | | 1 | Return by edge | |
| $L1_0$ | INT pin key-on wakeup control bit | 0 | Key-on wakeup not used | |
| | | 1 | Key-on wakeup used | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.

**Table 22 Pull-up control register and interrupt control register**

| Pull-up control register PU0 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAPU0/TPU0A |
|---|---|---|---|---|---|
| $PU0_3$ | Port $P0_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_2$ | Port $P0_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_1$ | Port $P0_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_0$ | Port $P0_0$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |

| Pull-up control register PU1 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAPU1/TPU1A |
|---|---|---|---|---|---|
| $PU1_3$ | Port $P1_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU1_2$ | Port $P1_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU1_1$ | Port $P1_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU1_0$ | Port $P1_0$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |

| Pull-up control register PU2 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAPU2/TPU2A |
|---|---|---|---|---|---|
| $PU2_3$ | Port $D_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU2_2$ | Port $D_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU2_1$ | Port $P2_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU2_0$ | Port $P2_0$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.

## CLOCK CONTROL

The clock control circuit consists of the following circuits.

- On-chip oscillator (internal oscillator)
- Ceramic oscillation circuit
- RC oscillation circuit
- Multi-plexer (clock selection circuit)
- Frequency divider
- Internal clock generating circuit

The system clock and the instruction clock are generated as the source clock for operation by these circuits.
Figure 49 shows the structure of the clock control circuit.
The 4508 Group operates by the on-chip oscillator clock (f(RING)) which is the internal oscillator after system is released from reset.
Also, the ceramic resonator or the RC oscillation can be used for the source oscillation (f(XIN)) of the 4508 Group.

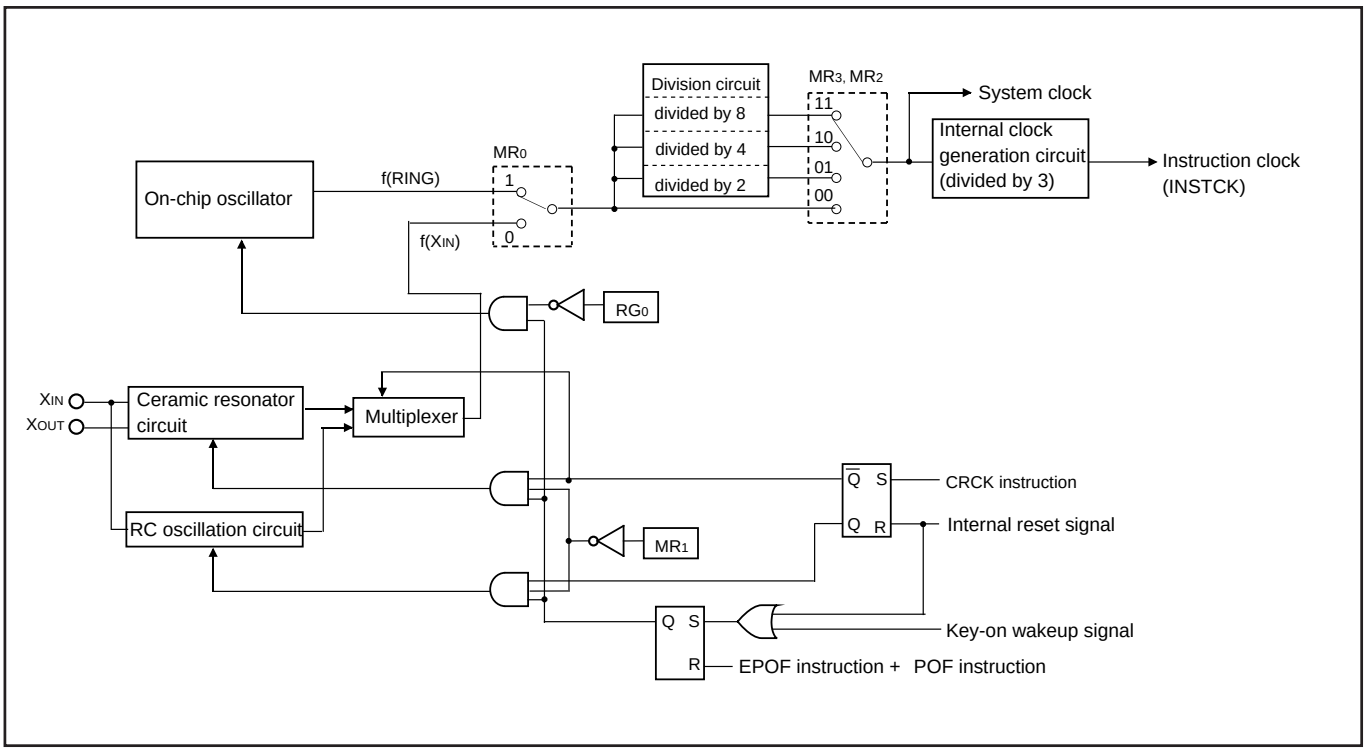



**Fig. 49 Clock control circuit structure**

## (1) On-chip oscillator operation

After system is released from reset, the MCU starts operation by the clock output from the on-chip oscillator which is the internal oscillator.

The clock frequency of the on-chip oscillator depends on the supply voltage and the operation temperature range.

Be careful that variable frequencies when designing application products.

## (2) Main clock generating circuit (f(XIN))

The ceramic resonator or RC oscillation can be used for the main clock of this product.

After system is released from reset, the ceramic oscillation is active for main clock.

The ceramic oscillation is invalid and the RC oscillation circuit is valid with the CRCK instruction.

Execute the CRCK instruction in the initial setting routine of program (executing it in address 0 in page 0 is recommended).

The execution of the CRCK instruction can be valid only once.

Register MR controls the enable/disable of the oscillation and the selection of the operation source clock.

Also, when the MCU operates only by the on-chip oscillator without using main clock f(XIN), connect XIN pin to Vss and leave XOUT pin open, and do not execute the CRCK instruction (Figure 51).

## (3) Ceramic resonator

When the ceramic resonator is used as the main clock (f(XIN)), connect the ceramic resonator and the external circuit to pins XIN and XOUT at the shortest distance. A feedback resistor is built in between pins XIN and XOUT (Figure 52).

Do not execute the CRCK instruction.

Set “0” to bit 0 of register MR after the oscillation stabilizing wait time is generated by software to select the clock generated by the ceramic oscillation circuit for the source oscillation clock.

## (4) RC oscillation

When the RC oscillation is used as the main clock (f(XIN)), connect the XIN pin to the external circuit of resistor R and the capacitor C at the shortest distance and leave XOUT pin open. Then, execute the CRCK instruction (Figure 53).

The frequency is affected by a capacitor, a resistor and a microcomputer. So, set the constants within the recommended operating condition of the frequency limits.

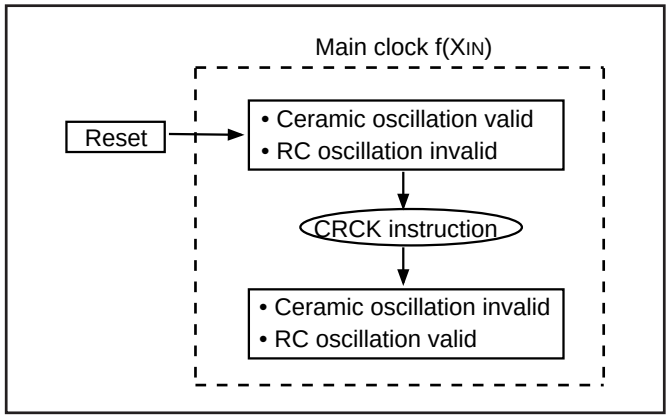



**Fig. 50 Switch to ceramic oscillation/RC oscillation**

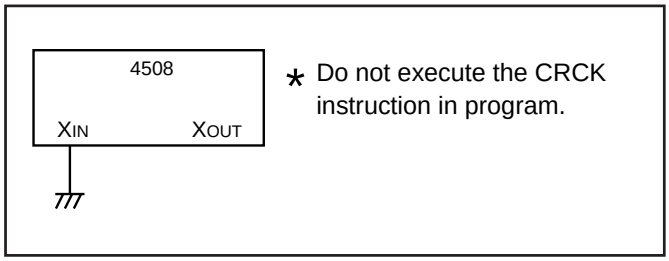



**Fig. 51 Handling of XIN and XOUT when main clock is not used**

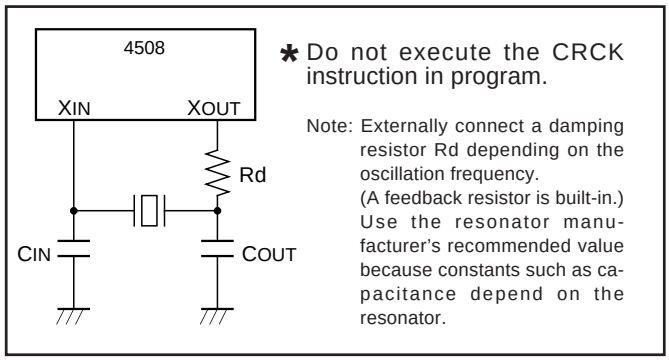



**Fig. 52 Ceramic resonator external circuit**

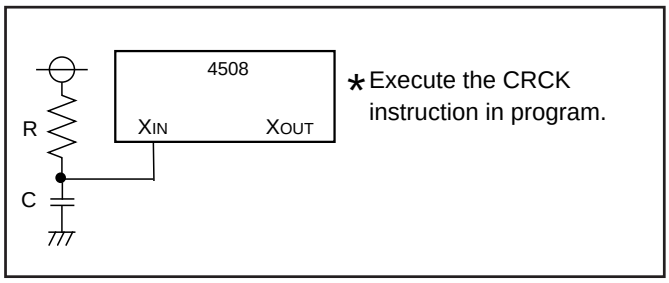



**Fig. 53 External RC circuit**

## (5) External clock

When the external signal clock is used for the main clock (f(XIN)), connect the XIN pin to the clock source and leave XOUT pin open (Figure 54). Do not execute the CRCK instruction in program.

Be careful that the maximum value of the oscillation frequency when using the external clock differs from the value when using the ceramic resonator (refer to the recommended operating condition).

Also, note that the RAM back-up mode (POF instruction) cannot be used when using the external clock.

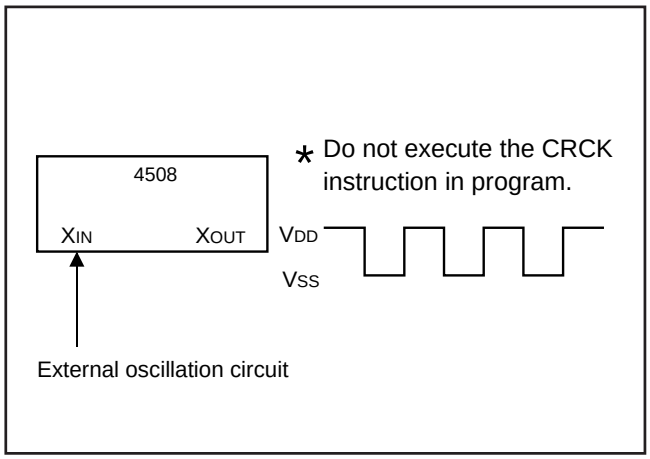



**Fig. 54 External clock input circuit**

## (6) Clock control register MR

Register MR controls the selection of operation mode and the operation source clock, and enable/stop of main clock. Set the contents of this register through register A with the TMRA instruction. In addition, the TAMR instruction can be used to transfer the contents of register MR to register A.

## (7) Clock control register RG

Register RG controls the on-chip oscillator. Set the contents of this register through register A with the TRGA instruction.

**Table 23 Clock control register MR**

| Clock control register MR | | at reset : $1101_2$ | | at RAM back-up : $1101_2$ | R/W TAMR/TMRA |
|---|---|---|---|---|---|
| MR3 | Operation mode selection bits | MR3 | MR2 | Operation mode | |
| | | 0 | 0 | Through mode (frequency not divided) | |
| MR2 | | 0 | 1 | Frequency divided by 2 mode | |
| | | 1 | 0 | Frequency divided by 4 mode | |
| | | 1 | 1 | Frequency divided by 8 mode | |
| MR1 | Main clock f(XIN) control bit (Notes 2, 5) | 0 | | Main clock (f(XIN)) oscillation enabled | |
| | | 1 | | Main clock (f(XIN)) oscillation stop | |
| MR0 | Operation source clock selection bit (Notes 3, 5) | 0 | | Main clock (f(XIN)) | |
| | | 1 | | On-chip oscillator clock (f(RING)) | |

| Clock control register RG | | at reset : $0_2$ | at RAM back-up : $0_2$ | W TRGA |
|---|---|---|---|---|
| RG0 | On-chip oscillator (f(RING)) control bit (Note 4) | 0 | On-chip oscillator (f(RING)) oscillation enabled | |
| | | 1 | On-chip oscillator (f(RING)) oscillation stop | |

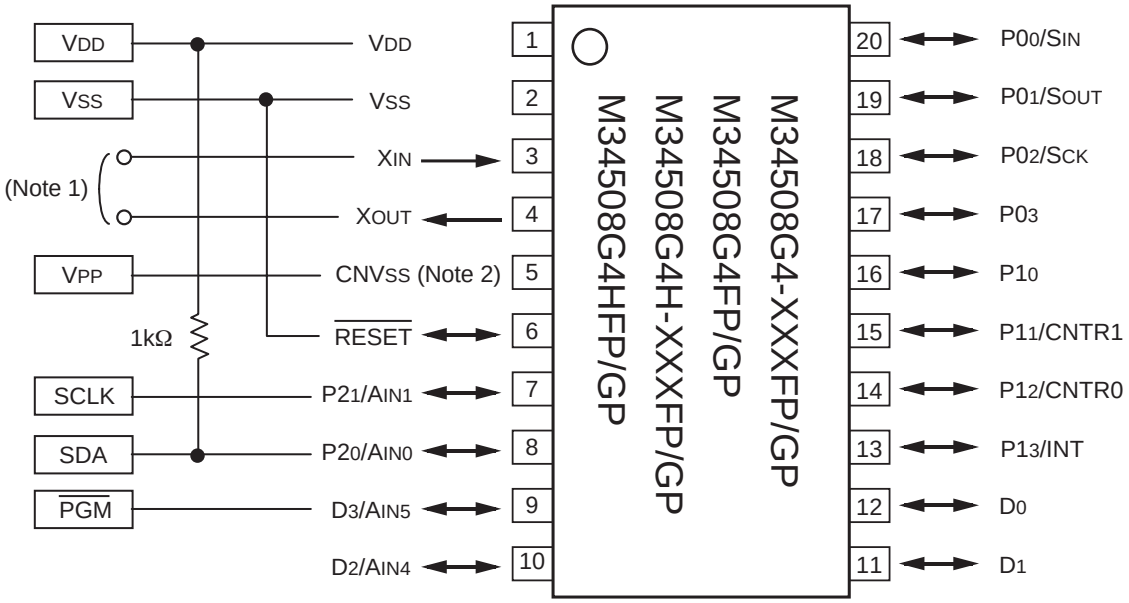
Notes 1: "R" represents read enabled, and "W" represents write enabled.
2: Main clock cannot be stopped when the main clock is selected for the operation source clock.
3: The stopped clock cannot be selected for the operation source clock. In order to switch the operation source clock, generate the oscillation stabilizing wait time by software first and set the oscillation of the destination clock to be enabled.
4: On-chip oscillator cannot be stopped when the on-chip oscillator is selected for the operation source clock.
5: When changing the setting of MR1 and MR0 from "00" to "11", make settings in the sequence "00" → "01" → "11".
When changing the setting of MR1 and MR0 from "11" to "0", make settings in the sequence "11" → "01" → "00".

## QzROM Writing Mode

In the QzROM writing mode, the user ROM area can be rewritten while the microcomputer is mounted on-board by using a serial programmer which is applicable for this microcomputer.
Table 24 lists the pin description (QzROM writing mode) and Figure 55 shows the pin connections.
Refer to Figure 56 for examples of a connection with a serial programmer.
Contact the manufacturer of your serial programmer for serial programmer. Refer to the user's manual of your serial programmer for details on how to use it.

**Table 24 Pin description (QzROM writing mode)**

| Pin | Name | I/O | Function |
|---|---|---|---|
| $V_{DD}$ | Power source | — | • Power supply voltage pin. |
| $V_{SS}$ | GND | — | • GND pin. |
| CNV$_{SS}$ | $V_{PP}$ input | — | • QzROM programmable power source pin.<br>• $V_{PP}$ input is possible with $V_{SS}$ connected via a resistor of about 5 kΩ. |
| P2$_0$/AIN$_0$ | SDA input/output | I/O | • QzROM serial data I/O pin. |
| P2$_1$/AIN$_1$ | SCLK input | Input | • QzROM serial clock input pin. |
| D$_3$/AIN$_5$ | $\overline{\text{PGM}}$ input | Input | • QzROM read/program pulse input pin. |
| $\overline{\text{RESET}}$ | Reset input | Input | • Reset input pin.<br>• Input "L" level signal. |
| $X_{IN}$ | Clock input | — | • Either connect an oscillation circuit or connect $X_{IN}$ pin to $V_{SS}$ and leave the $X_{OUT}$ pin open. |
| $X_{OUT}$ | Clock output | — | |
| D$_0$, D$_1$, D$_2$/AIN$_4$, P0$_0$/S$_{IN}$, P0$_1$/S$_{OUT}$, P0$_2$/S$_{CK}$, P0$_3$, P1$_0$, P1$_1$/CNTR1, P1$_2$/CNTR0, P1$_3$/INT | I/O port | I/O | • Input "H" or "L" level signal or leave the pin open. |

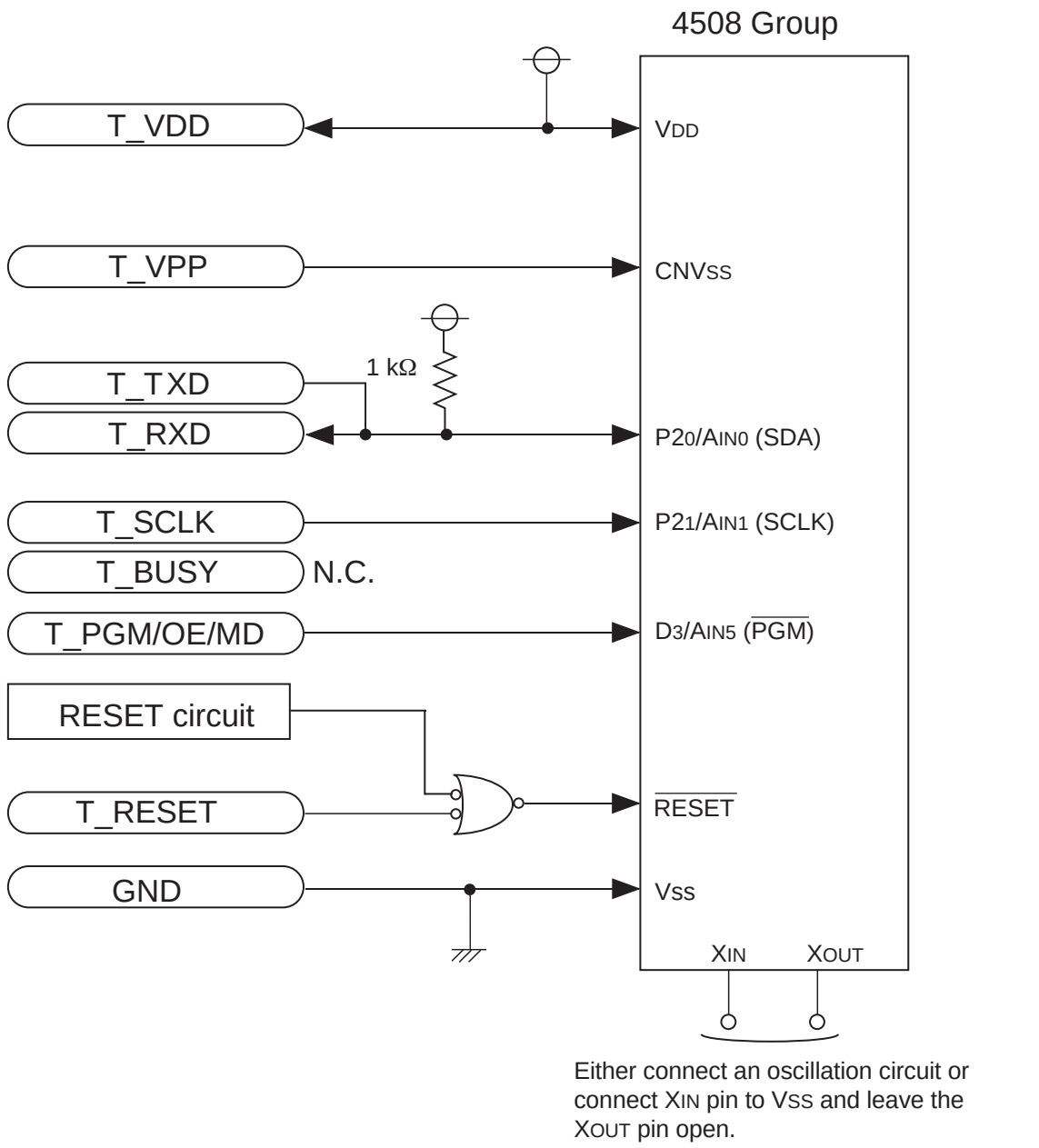



Package type FP: PRSP0020DA-A (20P2N-A)
GP: PLSP0020JB-A (20P2F-A)

Notes 1: Either connect an oscillation circuit or connect XIN pin to VSS and leave the XOUT pin open.
2: VPP input is possible with VSS connected via a resistor of about 5 kΩ.

: QzROM pin

**Fig. 55 Pin connection diagram**




Note: For the programming circuit, the wiring capacity of each signal pin must not exceed 47 pF.

**Fig. 56 When using programmer of Suisei Electronics System Co., LTD, connection example**

## DATA REQUIRED FOR QzROM WRITING ORDERS

The following are necessary when ordering a QzROM product shipped after writing:

1. QzROM Writing Confirmation Form*
2. Mark Specification Form*
3. ROM data.....Mask file

* For the QzROM writing confirmation form and the mark specification form, refer to the “Renesas Technology Corp.” Homepage (http://www.renesas.com/homepage.jsp).

Note that we cannot deal with special font marking (customer's trademark etc.) in QzROM microcomputer.

## LIST OF PRECAUTIONS

①Noise and latch-up prevention

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

- connect a bypass capacitor (approx. 0.1 μF) between pins $V_{DD}$ and $V_{SS}$ at the shortest distance,
- equalize its wiring in width and length, and
- use relatively thick wire.

$CNV_{SS}$ 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).

②Note on Power Source Voltage

When the power source voltage value of a microcomputer is less than the value which is indicated as the recommended operating conditions, the microcomputer does not operate normally and may perform unstable operation.

In a system where the power source voltage drops slowly when the power source voltage drops or the power supply is turned off, reset a microcomputer when the supply voltage is less than the recommended operating conditions and design a system not to cause errors to the system by this unstable operation.

③Register initial values 1

The initial value of the following registers are undefined after system is released from reset. After system is released from reset, set initial values.

- Register Z (2 bits)
- Register D (3 bits)
- Register E (8 bits)

④Register initial values 2

The initial value of the following registers are undefined at RAM back-up. After system is returned from RAM back-up, set initial values.

- Register Z (2 bits)
- Register X (4 bits)
- Register Y (4 bits)
- Register D (3 bits)
- Register E (8 bits)

⑤Program counter

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

⑥Stack registers (SKs) and stack pointer (SP)

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

⑦Multifunction

- The input/output of $P0_0$ can be used even when $S_{IN}$ is used. Be careful when using inputs of both $S_{IN}$ and $P0_0$ since the input threshold value of $S_{IN}$ pin is different from that of port $P0_0$.
- The input of $P0_1$ can be used even when $S_{OUT}$ is used.
- The input of $P0_2$ can be used even when $S_{CK}$ is used. Be careful when using inputs of both $S_{CK}$ and $P0_2$ since the input threshold value of $S_{CK}$ pin is different from that of port $P0_2$.
- The input of $P1_1$ can be used even when CNTR1 (output) is selected.

  The input/output of $P1_1$ can be used even when CNTR1 (input) is selected. Be careful when using inputs of both CNTR1 and $P1_1$ since the input threshold value of CNTR1 pin is different from that of port $P1_1$.
- The input of $P1_2$ can be used even when CNTR0 (output) is selected.

  The input/output of $P1_2$ can be used even when CNTR0 (input) is selected. Be careful when using inputs of both CNTR0 and $P1_2$ since the input threshold value of CNTR0 pin is different from that of port $P1_2$.
- The input/output of $P1_3$ can be used even when INT is used. Be careful when using inputs of both INT and $P1_3$ since the input threshold value of INT pin is different from that of port $P1_3$.
- The input/output of $P2_0$, $P2_1$, $D_2$, $D_3$ can be used even when $A_{IN0}$, $A_{IN1}$, $A_{IN4}$ or $A_{IN5}$ are used.

⑧Power-on reset (only for H version)

When the built-in power-on reset circuit is used, set the time for the supply voltage to rise from 0 V to the minimum voltage of recommended operating conditions to 100 μs or less.

If the rising time exceeds 100 μs, connect a capacitor between the $\overline{RESET}$ pin and Vss at the shortest distance, and input “L” level to $\overline{RESET}$ pin until the value of supply voltage reaches the minimum operating voltage.

⑨POF instruction

When the POF instruction is executed continuously after the EPOF instruction, system enters the RAM back-up state.

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

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

⑩ P13/INT pin

Note [1] on bit 3 of register I1

When the input of the INT pin is controlled with the bit 3 of register I1 in software, be careful about the following notes.

• Depending on the input state of the P13/INT pin, the external 0 interrupt request flag (EXF0) may be set when the bit 3 of register I1 is changed. In order to avoid the occurrence of an unexpected interrupt, clear the bit 0 of register V1 to "0" (refer to Figure 57①) and then, change the bit 3 of register I1.
In addition, execute the SNZ0 instruction to clear the EXF0 flag to "0" after executing at least one instruction (refer to Figure 57②).
Also, set the NOP instruction for the case when a skip is performed with the SNZ0 instruction (refer to Figure 57③).

```
    ⋮
LA    4        ; (XXX0₂)
TV1A           ; The SNZ0 instruction is valid ..... ①
LA    8        ; (1XXX₂)
TI1A           ; Control of INT pin input is changed
NOP            ..... ②
SNZ0           ; The SNZ0 instruction is executed
                 (EXF0 flag cleared)
NOP            ..... ③
    ⋮
X : these bits are not used here.
```

**Fig. 57 External 0 interrupt program example-1**

② Note [2] on bit 3 of register I1

When the bit 3 of register I1 is cleared to "0", the RAM back-up mode is selected and the input of INT pin is disabled, be careful about the following notes.

• When the INT pin input is disabled (register $I1_3$ = "0"), set the key-on wakeup of INT pin to be invalid (register $L1_0$ = "0") before system enters to the RAM back-up mode. (refer to Figure 58①).

```
    ⋮
LA    0        ; (XXX0₂)
TI1A           ; INT key-on wakeup disabled ..... ①
DI
EPOF
POF2           ; RAM back-up
    ⋮
X : these bits are not used here.
```

**Fig. 58 External 0 interrupt program example-2**

Note [3] on bit 2 of register I1

When the interrupt valid waveform of the P13/INT pin is changed with the bit 2 of register I1 in software, be careful about the following notes.

• Depending on the input state of the P13/INT pin, the external 0 interrupt request flag (EXF0) may be set when the bit 2 of register I1 is changed. In order to avoid the occurrence of an unexpected interrupt, clear the bit 0 of register V1 to "0" (refer to Figure 59①) and then, change the bit 2 of register I1.
In addition, execute the SNZ0 instruction to clear the EXF0 flag to "0" after executing at least one instruction (refer to Figure 59②).
Also, set the NOP instruction for the case when a skip is performed with the SNZ0 instruction (refer to Figure 59③).

```
    ⋮
LA    4        ; (XXX0₂)
TV1A           ; The SNZ0 instruction is valid ..... ①
LA    12       ; (1XXX₂)
TI1A           ; Interrupt valid waveform is changed
NOP            ..... ②
SNZ0           ; The SNZ0 instruction is executed
                 (EXF0 flag cleared)
NOP            ..... ③
    ⋮
X : these bits are not used here.
```

**Fig. 59 A/D conversion interrupt program example**

⑪ Prescaler

Stop prescaler counting and then execute the TABPS instruction to read its data.

Stop prescaler counting and then execute the TPSAB instruction to write data to prescaler.

⑫ Timer count source

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

⑬ Reading the count value

Stop timer 1 or 2 counting and then execute the TAB1 or TAB2 instruction to read its data.

⑭ Writing to the timer

Stop timer 1 or 2 counting and then execute the T1AB, T1R1L, T2AB or T2R2L instruction to write data to timer.

⑮ Writing to reload register

In order to write a data to the reload register R1H while the timer 1 is operating, execute the T1HAB instruction except a timing of the timer 1 underflow.

In order to write a data to the reload register R2H while the timer 2 is operating, execute the T2HAB instruction except a timing of the timer 2 underflow.

⑯ Prescaler, timer 1 and timer 2 count start timing and count time when operation starts

Count starts from the first rising edge of the count source (2) after prescaler and timer operations start (1).

Time to first underflow (3) is shorter (for up to 1 period of the count source) than time among next underflow (4) by the timing to start the timer and count source operations after count starts.

When selecting CNTR input as the count source of timer, timer operates synchronizing with the count edge (falling edge or rising edge) of CNTR input selected by software.

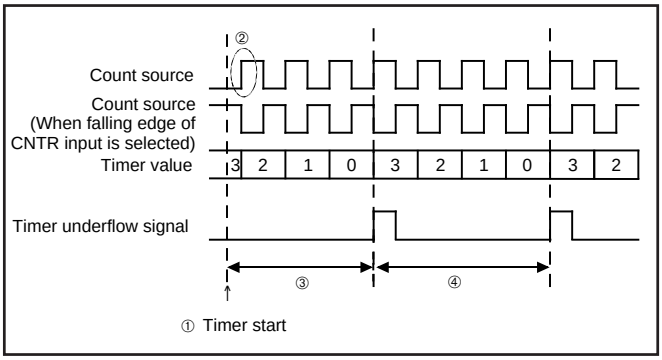



**Fig. 60 Timer count start timing and count time when operation starts**

⑰ PWM signal (PWM1, PWM2)

If the timer 1 count stop timing and the timer 1 underflow timing overlap during output of the PWM1 signal, a hazard may occur in the PWM1 output waveform.

If the timer 2 count stop timing and the timer 2 underflow timing overlap during output of the PWM2 signal, a hazard may occur in the PWM2 output waveform.

⑱ Watchdog timer

- The watchdog timer function is valid after system is released from reset. When not using the watchdog timer function, execute the DWDT instruction and the WRST instruction continuously, and clear the WEF flag to “0” to stop the watchdog timer function.
- The contents of WDF1 flag and timer WDT are initialized at the RAM back-up mode.
- 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 state.

  Also, set the NOP instruction after the WRST instruction, for the case when a skip is performed with the WRST instruction.

⑲ Clock control

When the RC oscillation is used as the main clock f(XIN), execute the CRCK instruction in the initial setting routine of program (executing it in address 0 in page 0 is recommended).

The oscillation circuit by the CRCK instruction can be selected only once. When the CRCK instruction is not executed, the ceramic oscillation is selected for the main clock f(XIN).

Also, when the MCU operates only by the on-chip oscillator without using main clock f(XIN), connect XIN pin to Vss and leave XOUT pin open, and do not execute the CRCK instruction.

In order to switch the operation source clock (f(RING)) or f(XIN)), generate the oscillation stabilizing wait time by software first and set the oscillation of the destination clock to be enabled.

Registers RG and MR are initialized when system returns from RAM back-up mode.

However, the selected contents (CRCK instruction execution state) of main clock (f(XIN)) oscillation circuit is retained.

⑳ On-chip oscillator

The clock frequency of the on-chip oscillator depends on the supply voltage and the operation temperature range.

Be careful that variable frequencies when designing application products.

Also, when considering the oscillation stabilize wait time for switching clock, be careful that the variable frequency of the on-chip oscillator clock.

㉑ External clock

When the external clock is used for the main clock (f(XIN)), connect the XIN pin to the clock source and leave XOUT pin open. Do not execute the CRCK instruction in program.

Be careful that the maximum value of the oscillation frequency when using the external clock differs from the value when using the ceramic resonator (refer to the recommended operating condition).

Also, note that the RAM back-up mode (POF instruction) cannot be used when using the external clock.

㉒ Notes for the use of A/D conversion 1

- TALA instruction
  When the TALA instruction is executed, the low-order 2 bits of register AD is transferred to the high-order 2 bits of register A, simultaneously, the low-order 2 bits of register A is “0.”
- Do not change the operating mode (both A/D conversion mode and comparator mode) of A/D converter with the bit 3 of register Q1 while the A/D converter is operating.
- Clear the bit 2 of register V2 to “0” to change the operating mode from the comparator mode to A/D conversion mode.
- The A/D conversion completion flag (ADF) may be set when the operating mode of the A/D converter is changed from the comparator mode to the A/D conversion mode. Accordingly, set a value to the bit 3 of register Q1, and execute the SNZAD instruction to clear the ADF flag.

```
      ⋮
LA    8       ; (X0XX2)
TV2A          ; The SNZAD instruction is valid ..... ①
LA    0       ; (0XXX2)
TQ1A          ; Operation mode of A/D converter is
                changed from comparator mode to A/D
                conversion mode.
SNZAD
NOP
      ⋮
```

X : this bit is not related to change the operation mode of A/D converter.

**Fig. 61 External 0 interrupt program example-3**

㉓ Notes for the use of A/D conversion 2

Each analog input pin is equipped with a capacitor which is used to compare the analog voltage. Accordingly, when the analog voltage is input from the circuit with high-impedance and, charge/discharge noise is generated and the sufficient A/D accuracy may not be obtained. Therefore, reduce the impedance or, connect a capacitor (0.01 μF to 1 μF) to analog input pins (Figure 62).
When the overvoltage applied to the A/D conversion circuit may occur, connect an external circuit in order to keep the voltage within the rated range as shown the Figure 61. In addition, test the application products sufficiently.

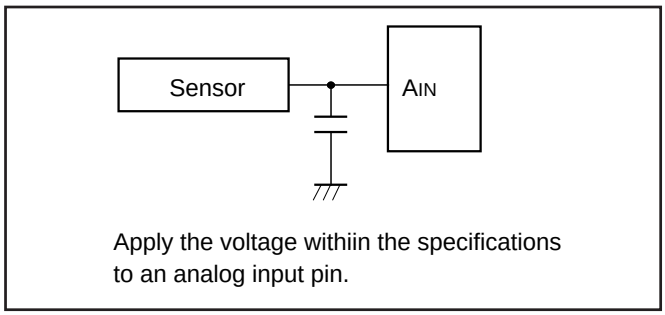



**Fig. 62 Analog input external circuit example-1**

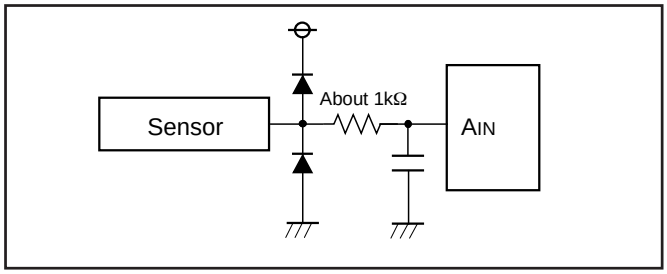



**Fig. 63 Analog input external circuit example-2**

㉔ QzROM

(1) Be careful not to apply overvoltage to MCU. The contents of QzROM may be overwritten because of overvoltage. Take care especially at turning on the power.

(2) As for the product shipped in blank, Renesas does not perform the writing test to user ROM area after the assembly process though the QzROM writing test is performed enough before the assembly process. Therefore, a writing error of approx.0.1 % may occur. Moreover, please note the contact of cables and foreign bodies on a socket, etc. because a writing environment may cause some writing errors.

㉕ Notes On ROM Code Protect
(QzROM product shipped after writing)

As for the QzROM product shipped after writing, the ROM code protect is specified according to the ROM option setup data in the mask file which is submitted at ordering.

The ROM option setup data in the mask file is “$00_{16}$” for protect enabled or “$FF_{16}$” for protect disabled.

Note that the mask file which has nothing at the ROM option data or has the data other than “$00_{16}$” and “$FF_{16}$” can not be accepted.

## NOTES ON NOISE

Countermeasures against noise are described below.
The following countermeasures are effective against noise in theory, however, it is necessary not only to take measures as follows but to evaluate before actual use.

**1. Shortest wiring length**

(1) Wiring for $\overline{\text{RESET}}$ pin
Make the length of wiring which is connected to the $\overline{\text{RESET}}$ pin as short as possible. Especially, connect a capacitor across the $\overline{\text{RESET}}$ pin and the Vss pin with the shortest possible wiring.

<Reason>
In order to reset a microcomputer correctly, 1 machine cycle or more of the width of a pulse input into the $\overline{\text{RESET}}$ pin is required.
If noise having a shorter pulse width than this is input to the $\overline{\text{RESET}}$ input pin, the reset is released before the internal state of the microcomputer is completely initialized.
This may cause a program runaway.

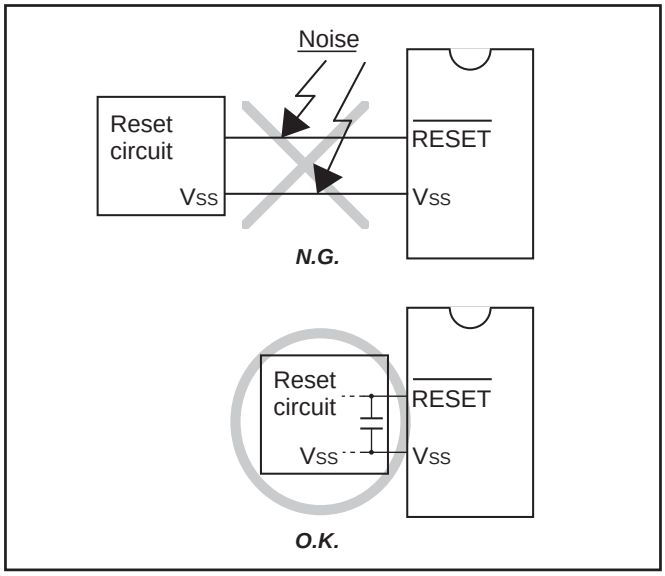



**Fig. 64 Wiring for the $\overline{\text{RESET}}$ pin**

(2) Wiring for clock input/output pins
- Make the length of wiring which is connected to clock I/O pins as short as possible.
- Make the length of wiring across the grounding lead of a capacitor which is connected to an oscillator and the Vss pin of a microcomputer as short as possible.
- Separate the Vss pattern only for oscillation from other Vss patterns.

<Reason>
If noise enters clock I/O pins, clock waveforms may be deformed. This may cause a program failure or program runaway. Also, if a potential difference is caused by the noise between the Vss level of a microcomputer and the Vss level of an oscillator, the correct clock will not be input in the microcomputer.

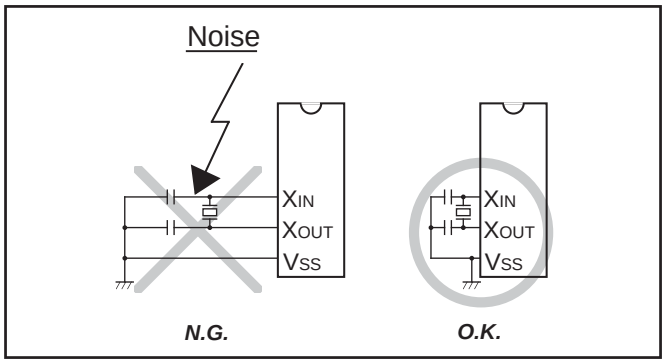



**Fig. 65 Wiring for clock I/O pins**

(3) Wiring to CNVss pin
Connect CNVss pin to a GND pattern at the shortest distance.
The GND pattern is required to be as close as possible to the GND supplied to Vss.
In order to improve the noise reduction, to connect a 5 kΩ resistor serially to the CNVss pin - GND line may be valid.
As well as the above-mentioned, in this case, connect to a GND pattern at the shortest distance. The GND pattern is required to be as close as possible to the GND supplied to Vss.

<Reason>
The CNVss pin of the QzROM is the power source input pin for the built-in QzROM. When programming in the built-in QzROM, the impedance of the CNVss pin is low to allow the electric current for writing flow into the QzROM. Because of this, noise can enter easily. If noise enters the CNVss pin, abnormal instruction codes or data are read from the built-in QzROM, which may cause a program runaway.

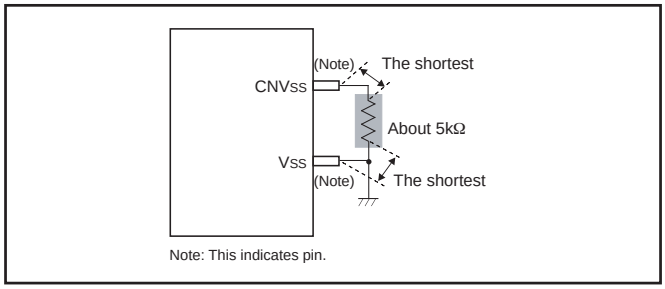



**Fig. 66 Wiring for the CNVss pin of the QzPROM**

**2. Connection of bypass capacitor across Vss line and VDD line**

Connect an approximately 0.1 $\mu$F bypass capacitor across the Vss line and the VDD line as follows:

- Connect a bypass capacitor across the Vss pin and the VDD pin at equal length.
- Connect a bypass capacitor across the Vss pin and the VDD pin with the shortest possible wiring.
- Use lines with a larger diameter than other signal lines for Vss line and VDD line.
- Connect the power source wiring via a bypass capacitor to the Vss pin and the VDD pin.

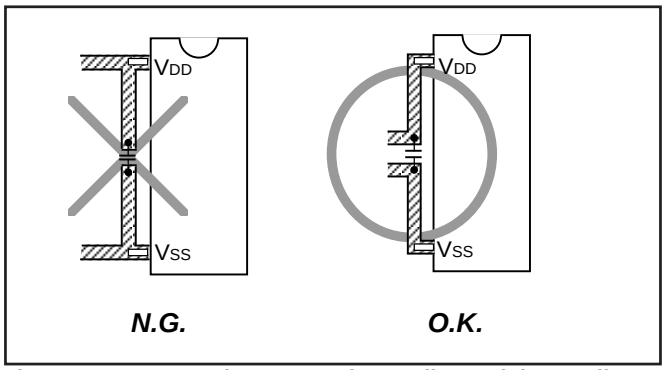



**Fig. 67 Bypass capacitor across the Vss line and the VDD line**

**3. Wiring to analog input pins**

- Connect an approximately 100 Ω to 1 kΩ resistor to an analog signal line which is connected to an analog input pin in series. Besides, connect the resistor to the microcomputer as close as possible.
- Connect an approximately 1000 pF capacitor across the Vss pin and the analog input pin. Besides, connect the capacitor to the Vss pin as close as possible. Also, connect the capacitor across the analog input pin and the Vss pin at equal length.

<Reason>

Signals which is input in an analog input pin (such as an A/D converter/comparator input pin) are usually output signals from sensor. The sensor which detects a change of event is installed far from the printed circuit board with a microcomputer, the wiring to an analog input pin is longer necessarily. This long wiring functions as an antenna which feeds noise into the microcomputer, which causes noise to an analog input pin.

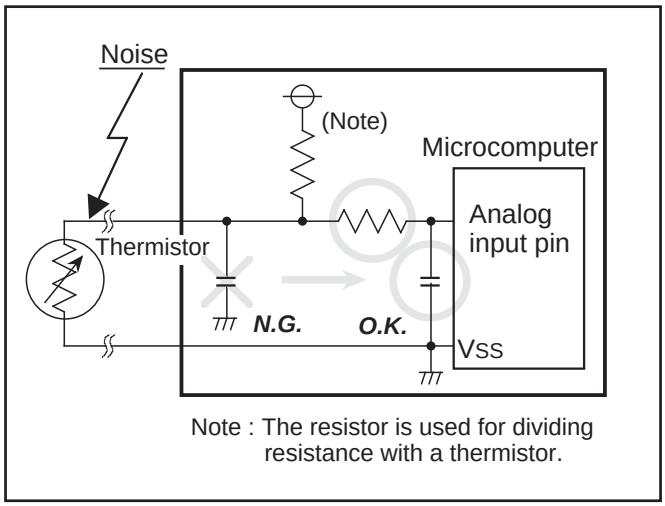



**Fig. 68 Analog signal line and a resistor and a capacitor**

**4. Oscillator concerns**

Take care to prevent an oscillator that generates clocks for a microcomputer operation from being affected by other signals.

(1) Keeping oscillator away from large current signal lines

Install a microcomputer (and especially an oscillator) as far as possible from signal lines where a current larger than the tolerance of current value flows.

<Reason>

In the system using a microcomputer, there are signal lines for controlling motors, LEDs, and thermal heads or others. When a large current flows through those signal lines, strong noise occurs because of mutual inductance.

(2) Installing oscillator away from signal lines where potential levels change frequently

Install an oscillator and a connecting pattern of an oscillator away from signal lines where potential levels change frequently. Also, do not cross such signal lines over the clock lines or the signal lines which are sensitive to noise.

<Reason>

Signal lines where potential levels change frequently (such as the CNTR pin signal line) may affect other lines at signal rising edge or falling edge. If such lines cross over a clock line, clock waveforms may be deformed, which causes a microcomputer failure or a program runaway.

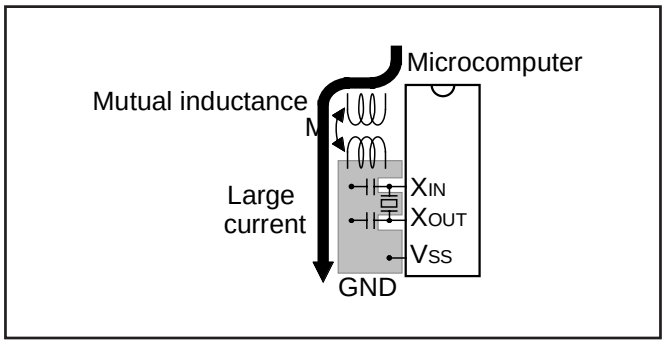



**Fig. 69 Wiring for a large current signal line**

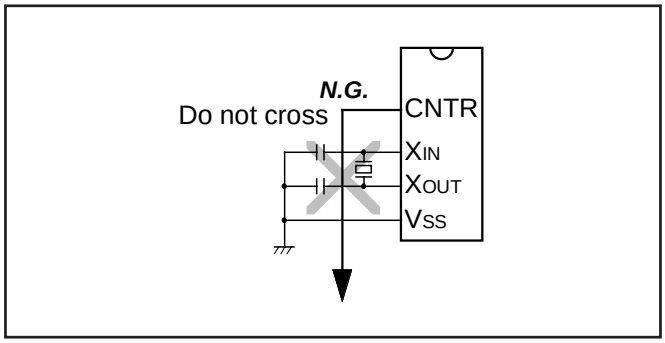



**Fig. 70 Wiring to a signal line where potential levels change frequently**

(3) Oscillator protection using Vss pattern

As for a two-sided printed circuit board, print a Vss pattern on the underside (soldering side) of the position (on the component side) where an oscillator is mounted.

Connect the Vss pattern to the microcomputer Vss pin with the shortest possible wiring. Besides, separate this Vss pattern from other Vss patterns.

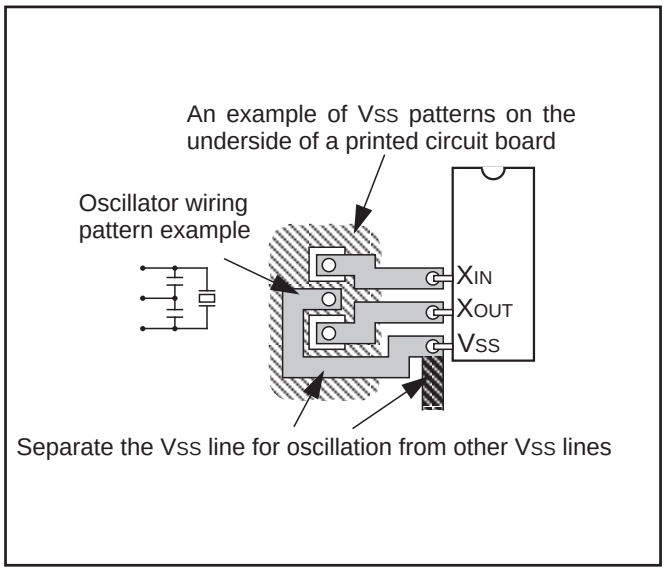



**Fig. 71 Vss pattern on the underside of an oscillator**

**5. Setup for I/O ports**

Setup I/O ports using hardware and software as follows:

<Hardware>

- Connect a resistor of 100 Ω or more to an I/O port in series.

<Software>

- As for an input port, read data several times by a program for checking whether input levels are equal or not.
- As for an output port or an I/O port, since the output data may reverse because of noise, rewrite data to its port latch at fixed periods.
- Rewrite data to pull-up control registers at fixed periods.

**6. Providing of watchdog timer function by software**

If a microcomputer runs away because of noise or others, it can be detected by a software watchdog timer and the microcomputer can be reset to normal operation. This is equal to or more effective than program runaway detection by a hardware watchdog timer. The following shows an example of a watchdog timer provided by software. In the following example, to reset a microcomputer to normal operation, the main routine detects errors of the interrupt processing routine and the interrupt processing routine detects errors of the main routine.

This example assumes that interrupt processing is repeated multiple times in a single main routine processing.

<The main routine>

- Assigns a single word of RAM to a software watchdog timer (SWDT) and writes the initial value N in the SWDT once at each execution of the main routine. The initial value N should satisfy the following condition:
  N+1 ≥ (Counts of interrupt processing executed in each main routine)
  As the main routine execution cycle may change because of an interrupt processing or others, the initial value N should have a margin.
- Watches the operation of the interrupt processing routine by comparing the SWDT contents with counts of interrupt processing after the initial value N has been set.
- Detects that the interrupt processing routine has failed and determines to branch to the program initialization routine for recovery processing in the following case:
  If the SWDT contents do not change after interrupt processing.

<The interrupt processing routine>

- Decrements the SWDT contents by 1 at each interrupt processing.
- Determines that the main routine operates normally when the SWDT contents are reset to the initial value N at almost fixed cycles (at the fixed interrupt processing count).
- Detects that the main routine has failed and determines to branch to the program initialization routine for recovery processing in the following case:
  If the SWDT contents are not initialized to the initial value N but continued to decrement and if they reach 0 or less.

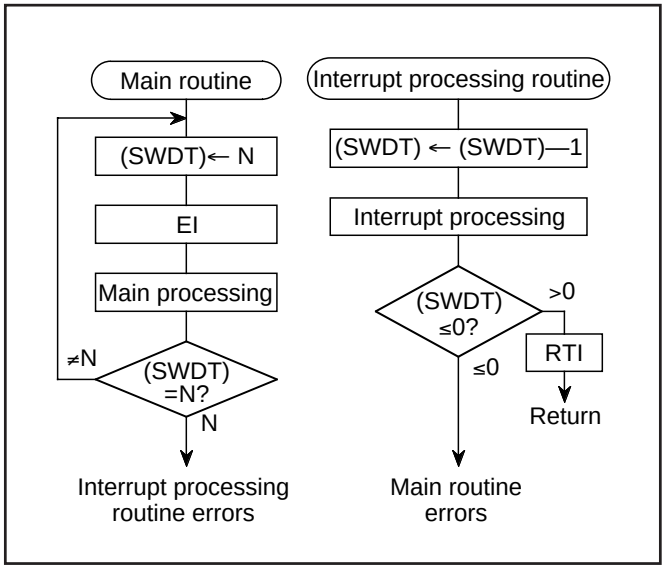



**Fig. 72 Watchdog timer by software**

## CONTROL REGISTERS

| Interrupt control register V1 | | at reset : $0000_2$ | at RAM back-up : $0000_2$ | R/W TAV1/TV1A |
|---|---|---|---|---|
| $V1_3$ | 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 TAV2/TV2A |
|---|---|---|---|---|
| $V2_3$ | Serial interface interrupt enable bit | 0 | Interrupt disabled (SNZSI instruction is valid) | |
| | | 1 | Interrupt enabled (SNZSI instruction is invalid) | |
| $V2_2$ | A/D interrupt enable bit | 0 | Interrupt disabled (SNZAD instruction is valid) | |
| | | 1 | Interrupt enabled (SNZAD instruction is invalid) | |
| $V2_1$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $V2_0$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |

| Interrupt control register I1 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W TAI1/TI1A |
|---|---|---|---|---|
| $I1_3$ | INT pin input control bit (Note 2) | 0 | INT pin input disabled | |
| | | 1 | INT pin input enabled | |
| $I1_2$ | 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 | |
| $I1_1$ | INT pin edge detection circuit control bit | 0 | One-sided edge detected | |
| | | 1 | Both edges detected | |
| $I1_0$ | INT pin timer 1 control enable bit | 0 | Disabled | |
| | | 1 | Enabled | |

| Clock control register MR | | at reset : $1101_2$ | | at RAM back-up : $1101_2$ | R/W TAMR/TMRA |
|---|---|---|---|---|---|
| $MR_3$ | Operation mode selection bits | $MR_3$ | $MR_2$ | Operation mode | |
| | | 0 | 0 | Through mode (frequency not divided) | |
| $MR_2$ | | 0 | 1 | Frequency divided by 2 mode | |
| | | 1 | 0 | Frequency divided by 4 mode | |
| | | 1 | 1 | Frequency divided by 8 mode | |
| $MR_1$ | Main clock f($X_{IN}$) control bit (Note 3) | 0 | | Main clock (f($X_{IN}$)) oscillation enabled | |
| | | 1 | | Main clock (f($X_{IN}$)) oscillation stop | |
| $MR_0$ | Operation source clock selection bit (Note 4) | 0 | | Main clock (f($X_{IN}$)) | |
| | | 1 | | On-chip oscillator clock (f(RING)) | |

| Clock control register RG | | at reset : $0_2$ | at RAM back-up : $0_2$ | W TRGA |
|---|---|---|---|---|
| $RG_0$ | On-chip oscillator (f(RING)) control bit (Note 5) | 0 | On-chip oscillator (f(RING)) oscillation enabled | |
| | | 1 | On-chip oscillator (f(RING)) oscillation stop | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.
2: When the contents of $I1_2$ and $I1_3$ are changed, the external interrupt request flag EXF0 may be set.
3: Main clock cannot be stopped when the main clock is selected for the operation source clock.
4: The stopped clock cannot be selected for the operation source clock. In order to switch the operation source clock, generate the oscillation stabilizing wait time by software first and set the oscillation of the destination clock to be enabled.
5: On-chip oscillator cannot be stopped when the on-chip oscillator is selected for the operation source clock.

| Timer control register PA | | at reset : $0_2$ | at RAM back-up : $0_2$ | W<br>TPAA |
|---|---|---|---|---|
| $PA_0$ | Prescaler control bit | 0 | Stop (state initialized) | |
| | | 1 | Operating | |

| Timer control register W1 | | at reset : $0000_2$ | | at RAM back-up : $0000_2$ | R/W<br>TAW1/TW1A |
|---|---|---|---|---|---|
| $W1_3$ | PWM1 function control bit | 0 | | PWM1 function invalid | |
| | | 1 | | PWM1 function valid | |
| $W1_2$ | Timer 1 control bit | 0 | | Stop (state retained) | |
| | | 1 | | Operating | |
| $W1_1$ | Timer 1 count source selection bits | $W1_1$ | $W1_0$ | Count source | |
| | | 0 | 0 | PWM2 signal | |
| $W1_0$ | | 0 | 1 | Prescaler output (ORCLK) | |
| | | 1 | 0 | CNTR1 input | |
| | | 1 | 1 | On-chip oscillator clock (f(RING)) | |

| Timer control register W2 | | at reset : $0000_2$ | | at RAM back-up : $0000_2$ | R/W<br>TAW2/TW2A |
|---|---|---|---|---|---|
| $W2_3$ | PWM2 function control bit | 0 | | PWM2 function invalid | |
| | | 1 | | PWM2 function valid | |
| $W2_2$ | Timer 2 control bit | 0 | | Stop (state retained) | |
| | | 1 | | Operating | |
| $W2_1$ | Timer 2 count source selection bits | $W2_1$ | $W2_0$ | Count source | |
| | | 0 | 0 | Timer 1 underflow signal (T1UDF) | |
| $W2_0$ | | 0 | 1 | Prescaler output (ORCLK) | |
| | | 1 | 0 | CNTR0 input | |
| | | 1 | 1 | System clock (STCK) | |

| Timer control register W5 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W<br>TAW5/TW5A |
|---|---|---|---|---|
| $W5_3$ | $P12_2$/CNTR0 pin function selection bit | 0 | $P12_2$ (I/O) / CNTR0 (input) | |
| | | 1 | $P12_2$ (input) /CNTR0 (I/O) | |
| $W5_2$ | Timer 1 count auto-stop circuit selection bit (Note 2) | 0 | Count auto-stop circuit not selected | |
| | | 1 | Count auto-stop circuit selected | |
| $W5_1$ | Timer 1 count start synchronous circuit selection bit (Note 3) | 0 | Count start synchronous circuit not selected | |
| | | 1 | Count start synchronous circuit selected | |
| $W5_0$ | CNTR0 pin input count edge selection bit | 0 | Falling edge | |
| | | 1 | Rising edge | |

| Timer control register W6 | | at reset : $0000_2$ | at RAM back-up : state retained | R/W<br>TAW6/TW6A |
|---|---|---|---|---|
| $W6_3$ | $P11_1$/CNTR1 pin function selection bit | 0 | $P11_1$ (I/O) / CNTR1 (input) | |
| | | 1 | $P11_1$ (input) /CNTR1 (I/O) | |
| $W6_2$ | CNTR 1 pin output auto-control circuit selection bit | 0 | Output auto-control circuit not selected | |
| | | 1 | Output auto-control circuit selected | |
| $W6_1$ | Timer 2<br>INT pin input period count circuit selection bit | 0 | INT pin input period count circuit not selected | |
| | | 1 | INT pin input period count circuit selected | |
| $W6_0$ | CNTR1 pin input count edge selection bit | 0 | Falling edge | |
| | | 1 | Rising edge | |

Notes 1: "R" represents read enabled, and "W" represents write enabled.
2: This function is valid only when the INT pin/timer 1 control is enabled ($I1_0$="1") and the timer 1 count start synchronous circuit is selected ($W5_1$="1").
3: This function is valid only when the INT pin/timer 1 control is enabled ($I1_0$="1").

| A/D control register Q1 | | at reset : $0000_2$ | | | at RAM back-up : state retained | R/W TAQ1/TQ1A |
|---|---|---|---|---|---|---|
| $Q1_3$ | A/D operation mode selection bit | 0 | A/D conversion mode | | | |
| | | 1 | Comparator mode | | | |
| $Q1_2$ | Analog input pin selection bits | $Q1_2$ | $Q1_1$ | $Q1_0$ | Selected pins | |
| | | 0 | 0 | 0 | $A_{IN0}$ | |
| | | 0 | 0 | 1 | $A_{IN1}$ | |
| $Q1_1$ | | 0 | 1 | 0 | Not available | |
| | | 0 | 1 | 1 | Not available | |
| | | 1 | 0 | 0 | $A_{IN4}$ | |
| $Q1_0$ | | 1 | 0 | 1 | $A_{IN5}$ | |
| | | 1 | 1 | 0 | Not available | |
| | | 1 | 1 | 1 | Not available | |

| Serial interface control register J1 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAJ1/TJ1A |
|---|---|---|---|---|---|
| $J1_3$ | Serial interface synchronous clock selection bits | $J1_3$ | $J1_2$ | Synchronous clock | |
| | | 0 | 0 | Instruction clock (INSTCK) divided by 8 | |
| $J1_2$ | | 0 | 1 | Instruction clock (INSTCK) divided by 4 | |
| | | 1 | 0 | Instruction clock (INSTCK) divided by 2 | |
| | | 1 | 1 | External clock ($S_{CK}$ input) | |
| $J1_1$ | Serial interface port function selection bits | $J1_1$ | $J1_0$ | Port function | |
| | | 0 | 0 | $P0_0$, $P0_1$, $P0_2$ selected/$S_{IN}$, $S_{OUT}$, $S_{CK}$ not selected | |
| $J1_0$ | | 0 | 1 | $P0_0$, $S_{OUT}$, $S_{CK}$ selected/$S_{IN}$, $P0_1$, $P0_2$ not selected | |
| | | 1 | 0 | $S_{IN}$, $P0_1$, $S_{CK}$ selected/$P0_0$, $S_{OUT}$, $P0_2$ not selected | |
| | | 1 | 1 | $S_{IN}$, $S_{OUT}$, $S_{CK}$ selected/$P0_0$, $P0_1$, $P0_2$ not selected | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.

| Key-on wakeup control register K0 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAK0/TK0A |
|---|---|---|---|---|---|
| $K0_3$ | Port $P0_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K0_2$ | Port $P0_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K0_1$ | Port $P0_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K0_0$ | Port $P0_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |

| Key-on wakeup control register K1 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAK1/TK1A |
|---|---|---|---|---|---|
| $K1_3$ | Port $P1_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K1_2$ | Port $P1_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K1_1$ | Port $P1_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K1_0$ | Port $P1_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |

| Key-on wakeup control register K2 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAK2/TK2A |
|---|---|---|---|---|---|
| $K2_3$ | Port $D_3$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K2_2$ | Port $D_2$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K2_1$ | Port $P2_1$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |
| $K2_0$ | Port $P2_0$ key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |

| Key-on wakeup control register L1 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAL1/TL1A |
|---|---|---|---|---|---|
| $L1_3$ | Ports $P1_0$–$P1_3$ return condition selection bit | 0 | Return by level | | |
| | | 1 | Return by edge | | |
| $L1_2$ | Ports $P1_0$–$P1_3$ valid waveform/ level selection bit | 0 | Falling waveform/“L” level | | |
| | | 1 | Rising waveform/“H” level | | |
| $L1_1$ | INT pin return condition selection bit | 0 | Return by level | | |
| | | 1 | Return by edge | | |
| $L1_0$ | INT pin key-on wakeup control bit | 0 | Key-on wakeup not used | | |
| | | 1 | Key-on wakeup used | | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.

| Pull-up control register PU0 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAPU0/TPU0A |
|---|---|---|---|---|---|
| $PU0_3$ | Port $P0_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_2$ | Port $P0_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_1$ | Port $P0_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU0_0$ | Port $P0_0$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |

| Pull-up control register PU1 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAPU1/TPU1A |
|---|---|---|---|---|---|
| $PU1_3$ | Port $P1_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU1_2$ | Port $P1_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU1_1$ | Port $P1_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU1_0$ | Port $P1_0$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |

| Pull-up control register PU2 | | at reset : $0000_2$ | | at RAM back-up : state retained | R/W TAPU2/TPU2A |
|---|---|---|---|---|---|
| $PU2_3$ | Port $D_3$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU2_2$ | Port $D_2$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU2_1$ | Port $P2_1$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |
| $PU2_0$ | Port $P2_0$ pull-up transistor control bit | 0 | Pull-up transistor OFF | | |
| | | 1 | Pull-up transistor ON | | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.

| Port output structure control register FR0 | | at reset : $0000_2$ | at RAM back-up : state retained | W<br>TFR0A |
|---|---|---|---|---|
| $FR0_3$ | Port $P0_3$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR0_2$ | Port $P0_2$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR0_1$ | Port $P0_1$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR0_0$ | Port $P0_0$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |

| Port output structure control register FR1 | | at reset : $0000_2$ | at RAM back-up : state retained | W<br>TFR1A |
|---|---|---|---|---|
| $FR1_3$ | Port $P1_3$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR1_2$ | Port $P1_2$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR1_1$ | Port $P1_1$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR1_0$ | Port $P1_0$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |

| Port output structure control register FR2 | | at reset : $0000_2$ | at RAM back-up : state retained | W<br>TFR2A |
|---|---|---|---|---|
| $FR2_3$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $FR2_2$ | Not used | 0 | This bit has no function, but read/write is enabled. | |
| | | 1 | | |
| $FR2_1$ | Port $P2_1$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR2_0$ | Port $P2_0$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |

| Port output structure control register FR3 | | at reset : $0000_2$ | at RAM back-up : state retained | W<br>TFR3A |
|---|---|---|---|---|
| $FR3_3$ | Port $D_3$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR3_2$ | Port $D_2$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR3_1$ | Port $D_1$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |
| $FR3_0$ | Port $D_0$ output structure selection bit | 0 | N-channel open-drain output | |
| | | 1 | CMOS output | |

Notes 1: “R” represents read enabled, and “W” represents write enabled.

## INSTRUCTIONS

Each instruction is described as follows;
(1) Index list of instruction function
(2) Machine instructions (index by alphabet)
(3) Machine instructions (index by function)
(4) Instruction code table

## SYMBOL

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

| Symbol | Contents | Symbol | Contents |
|---|---|---|---|
| A | Register A (4 bits) | RPS | Prescaler reload register (8 bits) |
| B | Register B (4 bits) | R1L | Timer 1 reload register (8 bits) |
| DR | Register D (3 bits) | R1H | Timer 1 reload register (8 bits) |
| E | Register E (8 bits) | R2L | Timer 2 reload register (8 bits) |
| Q1 | A/D control register Q1 (4 bits) | R2H | Timer 2 reload register (8 bits) |
| V1 | Interrupt control register V1 (4 bits) | PS | Prescaler |
| V2 | Interrupt control register V2 (4 bits) | T1 | Timer 1 |
| I1 | Interrupt control register I1 (4 bits) | T2 | Timer 2 |
| W1 | Timer control register W1 (4 bits) | T1F | Timer 1 interrupt request flag |
| W2 | Timer control register W2 (4 bits) | T2F | Timer 2 interrupt request flag |
| W5 | Timer control register W5 (4 bits) | WDF1 | Watchdog timer flag |
| W6 | Timer control register W6 (4 bits) | WEF | Watchdog timer enable flag |
| FR0 | Port output structure control register FR0 (4 bits) | INTE | Interrupt enable flag |
| FR1 | Port output structure control register FR1 (4 bits) | EXF0 | External 0 interrupt request flag |
| FR2 | Port output structure control register FR2 (4 bits) | P | Power down flag |
| FR3 | Port output structure control register FR3 (4 bits) | ADF | A/D conversion completion flag |
| J1 | Serial interface control register J1 (4 bits) | SIOF | Serial interface transmit/receive completion flag |
| MR | Clock control register MR (4 bits) | | |
| K0 | Key-on wakeup control register K0 (4 bits) | D | Port D (4 bits) |
| K1 | Key-on wakeup control register K1 (4 bits) | P0 | Port P0 (4 bits) |
| K2 | Key-on wakeup control register K2 (4 bits) | P1 | Port P1 (4 bits) |
| L1 | Key-on wakeup control register L1 (4 bits) | P2 | Port P2 (2 bits) |
| PU0 | Pull-up control register PU0 (4 bits) | | |
| PU1 | Pull-up control register PU1 (4 bits) | x | Hexadecimal variable |
| PU2 | Pull-up control register PU2 (4 bits) | y | Hexadecimal variable |
| X | Register X (4 bits) | z | Hexadecimal variable |
| Y | Register Y (4 bits) | p | Hexadecimal variable |
| Z | Register Z (2 bits) | n | Hexadecimal constant |
| DP | Data pointer (10 bits) | i | Hexadecimal constant |
| | (It consists of registers X, Y, and Z) | j | Hexadecimal constant |
| PC | Program counter (14 bits) | $A_3A_2A_1A_0$ | Binary notation of hexadecimal variable A |
| $PC_H$ | High-order 7 bits of program counter | | (same for others) |
| $PC_L$ | Low-order 7 bits of program counter | | |
| SK | Stack register (14 bits × 8) | ← | Direction of data movement |
| SP | Stack pointer (3 bits) | ↔ | Data exchange between a register and memory |
| CY | Carry flag | ? | Decision of state shown before “?” |
| | | ( ) | Contents of registers and memories |
| | | — | Negate, Flag unchanged after executing instruction |
| | | M(DP) | RAM address pointed by the data pointer |
| | | a | Label indicating address $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$ |
| | | p, a | Label indicating address $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$ in page $p_6$ $p_5$ $p_4$ $p_3$ $p_2$ $p_1$ $p_0$ |
| | | C | Hex. C + Hex. number x (also same for others) |
| | | + | |
| | | x | |

Note : The 4508 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.

## INDEX 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 (Note)<br>($PC_L$) ← ($DR_2$–$DR_0$, $A_3$–$A_0$)<br>(UPTF) = 1,<br>($DR_1$, $DR_0$) ← (ROM(PC))$_{9, 8}$<br>($DR_2$) ← 0<br>(B) ← (ROM(PC))$_{7–4}$<br>(A) ← (ROM(PC))$_{3–0}$<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 → $A_3A_2A_1A_0$ → |

Note: p is 0 to 31.

## INDEX LIST OF INSTRUCTION FUNCTION (continued)

| Grouping | Mnemonic | Function |
|---|---|---|
| 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 (Note)<br>($PC_L$) ← $a_6$–$a_0$ |
| | BLA p | ($PC_H$) ← p (Note)<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 (Note)<br>($PC_L$) ← $a_6$–$a_0$ |
| | BMLA p | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>($PC_H$) ← p (Note)<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 |

| Grouping | Mnemonic | Function |
|---|---|---|
| Interrupt operation | DI | (INTE) ← 0 |
| | EI | (INTE) ← 1 |
| | SNZ0 | $V1_0$ = 0: (EXF0) = 1 ?<br>(EXF0) ← 0<br>$V1_0$ = 1: SNZ0 = NOP |
| | SNZI0 | $I1_2$ = 0 : (INT) = "L" ?<br>$I1_2$ = 1 : (INT) = "H" ? |
| | TAV1 | (A) ← (V1) |
| | TV1A | (V1) ← (A) |
| | TAV2 | (A) ← (V2) |
| | TV2A | (V2) ← (A) |
| | TAI1 | (A) ← (I1) |
| | TI1A | (I1) ← (A) |
| Timer operation | TPAA | (PA) ← (A) |
| | TAW1 | (A) ← (W1) |
| | TW1A | (W1) ← (A) |
| | TAW2 | (A) ← (W2) |
| | TW2A | (W2) ← (A) |
| | TAW5 | (A) ← (W5) |
| | TW5A | (W5) ← (A) |
| | TAW6 | (A) ← (W6) |
| | TW6A | (W6) ← (A) |
| | TABPS | (B) ← ($TPS_7$–$TPS_4$)<br>(A) ← ($TPS_3$–$TPS_0$) |
| | TPSAB | ($RPS_7$–$RPS_4$) ← (B)<br>($TPS_7$–$TPS_4$) ← (B)<br>($RPS_3$–$RPS_0$) ← (A)<br>($TPS_3$–$TPS_0$) ← (A) |
| | TAB1 | (B) ← ($T1_7$–$T1_4$)<br>(A) ← ($T1_3$–$T1_0$) |

Note: p is 0 to 31.

## INDEX LIST OF INSTRUCTION FUNCTION (continued)

| Grouping | Mnemonic | Function |
|---|---|---|
| Timer operation | T1AB | (R1L7–R1L4) ← (B)<br>(T17–T14) ← (B)<br>(R1L3–R1L0) ← (A)<br>(T13–T10) ← (A) |
| | T1HAB | (R1H7–R1H4) ← (B)<br>(R1H3–R1H0) ← (A) |
| | TAB2 | (B) ← (T27–T24)<br>(A) ← (T23–T20) |
| | T2AB | (R2L7–R2L4) ← (B)<br>(T27–T24) ← (B)<br>(R2L3–R2L0) ← (A)<br>(T23–T20) ← (A) |
| | T2HAB | (R2H7–R2H4) ← (B)<br>(R2H3–R2H0) ← (A) |
| | T1R1L | (T17–T10) ← (R1L7–R1L0) |
| | T2R2L | (T27–T20) ← (R2L7–R2L0) |
| | SNZT1 | V12 = 0: (T1F) = 1 ?<br>(T1F) ← 0<br>V12 = 1: SNZT1 = NOP |
| | SNZT2 | V13 = 0: (T2F) = 1 ?<br>(T2F) ← 0<br>V13 = 1: SNZT2 = NOP |
| Input/Output operation | IAP0 | (A) ← (P0) |
| | OP0A | (P0) ← (A) |
| | IAP1 | (A) ← (P1) |
| | OP1A | (P1) ← (A) |
| | IAP2 | (A1, A0) ← (P21, P20)<br>(A3, A2) ← 0 |
| | OP2A | (P21, P20) ← (A1, A0) |
| | CLD | (D) ← 1 |
| | RD | (D(Y)) ← 0<br>(Y) = 0 to 3 |
| | SD | (D(Y)) ← 1<br>(Y) = 0 to 3 |
| Input/Output operation | SZD | (D(Y)) = 0 ?<br>(Y) = 0 to 3 |
| | TFR0A | (FR0) ← (A) |
| | TFR1A | (FR1) ← (A) |
| | TFR2A | (FR2) ← (A) |
| | TFR3A | (FR3) ← (A) |
| | TK0A | (K0) ← (A) |
| | TAK0 | (A) ← (K0) |
| | TK1A | (K1) ← (A) |
| | TAK1 | (A) ← (K1) |
| | TK2A | (K2) ← (A) |
| | TAK2 | (A) ← (K2) |
| | TPU0A | (PU0) ← (A) |
| | TAPU0 | (A) ← (PU0) |
| | TPU1A | (PU1) ← (A) |
| | TAPU1 | (A) ← (PU1) |
| | TPU2A | (PU2) ← (A) |
| | TAPU2 | (A) ← (PU2) |
| | TL1A | (L1) ← (A) |
| | TAL1 | (A) ← (L1) |

## INDEX LIST OF INSTRUCTION FUNCTION (continued)

| Grouping | Mnemonic | Function |
|---|---|---|
| Serial interface operation | TABSI | (B) ← (SI7–SI4) (A) ← (SI3–SI0) |
| | TSIAB | (SI7–SI4) ← (B) (SI3–SI0) ← (A) |
| | SST | (SIOF) ← 0<br>Serial interface transmit/receive starting |
| | SNZSI | V23=0: (SIOF)=1?<br>(SIOF) ← 0<br>V23 = 1: SNZSI = NOP |
| | TAJ1 | (A) ← (J1) |
| | TJ1A | (J1) ← (A) |
| Clock operation | CRCK | RC oscillator selected |
| | TRGA | (RG0) ← (A0) |
| | TAMR | (A) ← (MR) |
| | TMRA | (MR) ← (A) |
| A/D conversion operation | TABAD | Q13 = 0,<br>(B) ← (AD9–AD6)<br>(A) ← (AD5–AD2)<br>Q13 = 1,<br>(B) ← (AD7–AD4)<br>(A) ← (AD3–AD0) |
| | TALA | (A3, A2) ← (AD1, AD0)<br>(A1, A0) ← 0 |
| | TADAB | Q13 = 1 : (AD7–AD4) ← (B)<br>(AD3–AD0) ← (A)<br>Q13 = 0 : TABAD = NOP |
| | TAQ1 | (A) ← (Q1) |
| | TQ1A | (Q1) ← (A) |
| | ADST | (ADF) ← 0<br>Q13 = 0 : A/D conversion starting<br>Q13 = 1 : Comparator operation starting |
| | SNZAD | V22 = 0: (ADF) = 1 ?<br>(ADF) ← 0<br>V22 = 1: SNZAD = NOP |

| Grouping | Mnemonic | Function |
|---|---|---|
| Other operation | NOP | (PC) ← (PC) + 1 |
| | POF | RAM back-up |
| | EPOF | POF instruction valid |
| | SNZP | (P) = 1 ? |
| | DWDT | Stop of watchdog timer function enabled |
| | WRST | (WDF1) = 1 ?,<br>(WDF1) ← 0 |
| | SRST | System reset |
| | RUPT | (UPTF) ← 0 |
| | SUPT | (UPTF) ← 1 |
| | SVDE** | Voltage drop detection circuit valid at RAM back-up |

Note: The SVDE instruction can be used only in the H version.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET)

### **A n** (Add n and accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | n | n | n | n | 2 | 0 6 n | 16 | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | Overflow = 0 |

**Operation:** (A) ← (A) + n
n = 0 to 15

**Grouping:** Arithmetic operation

**Description:** Adds the value n in the immediate field to register A, and stores a result in register A.
The contents of carry flag CY remains unchanged.
Skips the next instruction when there is no overflow as the result of operation.
Executes the next instruction when there is overflow as the result of operation.

### **ADST** (A/D conversion STart)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 9 F | 16 | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (ADF) ← 0
$Q1_3$ = 0: A/D conversion starting
$Q1_3$ = 1: Comparator operation starting
($Q1_3$ : bit 3 of A/D control register Q1)

**Grouping:** A/D conversion operation

**Description:** Clears (0) to A/D conversion completion flag ADF, and the A/D conversion at the A/D conversion mode ($Q1_3$ = 0) or the comparator operation at the comparator mode ($Q1_3$ = 1) is started.

### **AM** (Add accumulator and Memory)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 2 | 0 0 A | 16 | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (A) + (M(DP))

**Grouping:** Arithmetic operation

**Description:** Adds the contents of M(DP) to register A. Stores the result in register A. The contents of carry flag CY remains unchanged.

### **AMC** (Add accumulator, Memory and Carry)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | 0 0 B | 16 | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | 0/1 | – |

**Operation:** (A) ← (A) + (M(DP)) + (CY)
(CY) ← Carry

**Grouping:** Arithmetic operation

**Description:** Adds the contents of M(DP) and carry flag CY to register A. Stores the result in register A and carry flag CY.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### AND (logical AND between accumulator and memory)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 8 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (A) AND (M(DP))

**Grouping:** Arithmetic operation

**Description:** Takes the AND operation between the contents of register A and the contents of M(DP), and stores the result in register A.

### B a (Branch to address a)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 1 | 8+a | a |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($PC_L$) ← $a_6$ to $a_0$

**Grouping:** Branch operation

**Description:** Branch within a page : Branches to address a in the identical page.

**Note:** Specify the branch address within the page including this instruction.

### BL p, a (Branch Long to address a in page p)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ | 0 | E+p | p |

$_2$ $_{16}$

| 1 | 0 | 0 | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 2 | a | a |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 2 | 2 | – | – |

**Operation:** ($PC_H$) ← p
($PC_L$) ← $a_6$ to $a_0$

**Grouping:** Branch operation

**Description:** Branch out of a page : Branches to address a in page p.

**Note:** p is 0 to 31.

### BLA p (Branch Long to address (D) + (A) in page p)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

$_2$ $_{16}$

| 1 | 0 | 0 | $p_4$ | 0 | 0 | $p_3$ | $p_2$ | $p_1$ | $p_0$ | 2 | p | p |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 2 | 2 | – | – |

**Operation:** ($PC_H$) ← p
($PC_L$) ← ($DR_2$–$DR_0$, $A_3$–$A_0$)

**Grouping:** Branch operation

**Description:** 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.

**Note:** p is 0 to 31.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### BM a (Branch and Mark to address a in page 2)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | a6 | a5 | a4 | a3 | a2 | a1 | a0 | 2 | 1 | a | a | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:**
(SP) ← (SP) + 1
(SK(SP)) ← (PC)
(PCH) ← 2
(PCL) ← a6–a0

**Grouping:** Subroutine call operation

**Description:** Call the subroutine in page 2 : Calls the subroutine at address a in page 2.

**Note:** Subroutine extending from page 2 to another page can also be called with the BM instruction when it starts on page 2.
Be careful not to over the stack because the maximum level of subroutine nesting is 8.

### BML p, a (Branch and Mark Long to address a in page p)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | p4 | p3 | p2 | p1 | p0 | 2 | 0 | C +p | p | 16 |
| 1 | 0 | 0 | a6 | a5 | a4 | a3 | a2 | a1 | a0 | 2 | 2 | a | a | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 2 | 2 | – | – |

**Operation:**
(SP) ← (SP) + 1
(SK(SP)) ← (PC)
(PCH) ← p
(PCL) ← a6–a0

**Grouping:** Subroutine call operation

**Description:** Call the subroutine : Calls the subroutine at address a in page p.

**Note:** p is 0 to 31.
Be careful not to over the stack because the maximum level of subroutine nesting is 8.

### BMLA p (Branch and Mark Long to address (D) + (A) in page p)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 3 | 0 | 16 |
| 1 | 0 | 0 | p4 | 0 | 0 | p3 | p2 | p1 | p0 | 2 | 2 | p | p | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 2 | 2 | – | – |

**Operation:**
(SP) ← (SP) + 1
(SK(SP)) ← (PC)
(PCH) ← p
(PCL) ← (DR2–DR0, A3–A0)

**Grouping:** Subroutine call operation

**Description:** Call the subroutine : Calls the subroutine at address (DR2 DR1 DR0 A3 A2 A1 A0)2 specified by registers D and A in page p.

**Note:** p is 0 to 31.
Be careful not to over the stack because the maximum level of subroutine nesting is 8.

### CLD (CLear port D)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 1 | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (D) ← 1

**Grouping:** Input/Output operation

**Description:** Sets (1) to port D.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### CMA (CoMplement of Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | $_2$ |
| 0 | 1 | C | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← $\overline{(A)}$

**Grouping:** Arithmetic operation

**Description:** Stores the one's complement for register A's contents in register A.

### CRCK (Clock select: Rc oscillation ClocK)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | $_2$ |
| 2 | 9 | B | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** RC oscillation circuit selected

**Grouping:** Other operation

**Description:** Selects the RC oscillation circuit for main clock f(XIN).

### DEY (DEcrement register Y)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | $_2$ |
| 0 | 1 | 7 | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (Y) = 15 |

**Operation:** (Y) ← (Y) – 1

**Grouping:** RAM addresses

**Description:** 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. When the contents of register Y is not 15, the next instruction is executed.

### DI (Disable Interrupt)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | $_2$ |
| 0 | 0 | 4 | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (INTE) ← 0

**Grouping:** Interrupt control operation

**Description:** Clears (0) to interrupt enable flag INTE, and disables the interrupt.

**Note:** Interrupt is disabled by executing the DI instruction after executing 1 machine cycle.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### **DWDT** (Disable WatchDog Timer)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 2 | 9 | C | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** Stop of watchdog timer function enabled

**Grouping:** Other operation

**Description:** Stops the watchdog timer function by the WRST instruction after executing the DWDT instruction.

### **EI** (Enable Interrupt)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 5 | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (INTE) ← 1

**Grouping:** Interrupt control operation

**Description:** Sets (1) to interrupt enable flag INTE, and enables the interrupt.

**Note:** Interrupt is enabled by executing the EI instruction after executing 1 machine cycle.

### **EPOF** (Enable POF instruction)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 5 | B | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** POF instruction valid

**Grouping:** Other operation

**Description:** Makes the immediate after POF instruction valid by executing the EPOF instruction.

### **IAP0** (Input Accumulator from port P0)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 6 | 0 | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (P0)

**Grouping:** Input/Output operation

**Description:** Transfers the input of port P0 to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### IAP1 (Input Accumulator from port P1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | $_2$ | 2 6 1 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (P1)

**Grouping:** Input/Output operation

**Description:** Transfers the input of port P1 to register A.

### IAP2 (Input Accumulator from port P2)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | $_2$ | 2 6 2 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($A_1$, $A_0$) ← ($P2_1$, $P2_0$)
($A_3$, $A_2$) ← 0

**Grouping:** Input/Output operation

**Description:** Transfers the input of port P2 to the low-order 2 bits ($A_1$, $A_0$) of register A.

**Note:** After this instruction is executed, "0" is stored to the high-order 2 bits ($A_3$, $A_2$) of register A.

### INY (INcrement register Y)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | $_2$ | 0 1 3 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (Y) = 0 |

**Operation:** (Y) ← (Y) + 1

**Grouping:** RAM addresses

**Description:** 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. When the contents of register Y is not 0, the next instruction is executed.

### LA n (Load n in Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | n | n | n | n | $_2$ | 0 7 n | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | Continuous description |

**Operation:** (A) ← n
n = 0 to 15

**Grouping:** Arithmetic operation

**Description:** Loads the value n in the immediate field to register A.
When the LA instructions are continuously coded and executed, only the first LA instruction is executed and other LA instructions coded continuously are skipped.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### LXY x, y (Load register X and Y with x and y)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | x3 | x2 | x1 | x0 | y3 | y2 | y1 | y0 | 3 | x | y |
| 2 | | | | | | | | | | 16 | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | Continuous description |

**Operation:** (X) ← x x = 0 to 15
(Y) ← y y = 0 to 15

**Grouping:** RAM addresses

**Description:** Loads the value x in the immediate field to register X, and the value y in the immediate field to register Y. 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 (Load register Z with z)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | z1 | z0 | 0 | 4 | 8 +z |
| 2 | | | | | | | | | | 16 | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (Z) ← z z = 0 to 3

**Grouping:** RAM addresses

**Description:** Loads the value z in the immediate field to register Z.

### NOP (No OPeration)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | | | | | | | | | | 16 | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (PC) ← (PC) + 1

**Grouping:** Other operation

**Description:** No operation; Adds 1 to program counter value, and others remain unchanged.

### OP0A (Output port P0 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 |
| 2 | | | | | | | | | | 16 | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (P0) ← (A)

**Grouping:** Input/Output operation

**Description:** Outputs the contents of register A to port P0.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### OP1A (Output port P1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | $_2$ | 2 | 2 | 1 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (P1) ← (A)

**Grouping:** Input/Output operation

**Description:** Outputs the contents of register A to port P1.

### OP2A (Output port P2 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | $_2$ | 2 | 2 | 2 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($P2_1$, $P2_0$) ← ($A_1$, $A_0$)

**Grouping:** Input/Output operation

**Description:** Outputs the contents of the low-order 2 bits ($A_1$, $A_0$) of register A to port P2.

### OR (logical OR between accumulator and memory)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | $_2$ | 0 | 1 | 9 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (A) OR (M(DP))

**Grouping:** Arithmetic operation

**Description:** Takes the OR operation between the contents of register A and the contents of M(DP), and stores the result in register A.

### POF (Power OFF)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | $_2$ | 0 | 0 | 2 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** RAM back-up

**Grouping:** Other operation

**Description:** Puts the system in RAM back-up state by executing the POF instruction after executing the EPOF instruction.

**Note:** If the EPOF instruction is not executed just before this instruction, this instruction is equivalent to the NOP instruction.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### RAR (Rotate Accumulator Right)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | $_2$ |
| 0 | 1 | D | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | 0/1 | – |

**Operation:** → CY → $A_3A_2A_1A_0$ →

**Grouping:** Arithmetic operation

**Description:** Rotates 1 bit of the contents of register A including the contents of carry flag CY to the right.

### RB j (Reset Bit)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | j | j | $_2$ |
| 0 | 4 | C+j | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (Mj(DP)) ← 0
j = 0 to 3

**Grouping:** Bit operation

**Description:** Clears (0) the contents of bit j (bit specified by the value j in the immediate field) of M(DP).

### RC (Reset Carry flag)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | $_2$ |
| 0 | 0 | 6 | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | 0 | – |

**Operation:** (CY) ← 0

**Grouping:** Arithmetic operation

**Description:** Clears (0) to carry flag CY.

### RD (Reset port D specified by register Y)

**Instruction code**

| D9 | | | | | | | | | D0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | $_2$ |
| 0 | 1 | 4 | | | | | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (D(Y)) ← 0
However,
(Y) = 0 to 3

**Grouping:** Input/Output operation

**Description:** Clears (0) to a bit of port D specified by register Y.

**Note:** (Y) = 0 to 3.
Do not execute this instruction if values except above are set to register Y.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### **RT** (ReTurn from subroutine)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | $_2$ | 0 4 4 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 2 | – | – |

**Operation:** (PC) ← (SK(SP))
(SP) ← (SP) – 1

**Grouping:** Return operation

**Description:** Returns from subroutine to the routine called the subroutine.

### **RTI** (ReTurn from Interrupt)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | $_2$ | 0 4 6 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (PC) ← (SK(SP))
(SP) ← (SP) – 1

**Grouping:** Return operation

**Description:** Returns from interrupt service routine to main routine.
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.

### **RTS** (ReTurn from subroutine and Skip)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | $_2$ | 0 4 5 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 2 | – | Skip at uncondition |

**Operation:** (PC) ← (SK(SP))
(SP) ← (SP) – 1

**Grouping:** Return operation

**Description:** Returns from subroutine to the routine called the subroutine, and skips the next instruction at uncondition.

### **RUPT** (Reset UPT flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | $_2$ | 0 5 8 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (UPTF) ← 0

**Grouping:** Other operation

**Description:** Clears (0) to the high-order bit reference enable flag UPTF.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### SB j (Set Bit)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | j | j | $_2$ | 0 | 5 | C+j $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (Mj(DP)) ← 0
j = 0 to 3

**Grouping:** Bit operation

**Description:** Sets (1) the contents of bit j (bit specified by the value j in the immediate field) of M(DP).

### SC (Set Carry flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | $_2$ | 0 | 0 | 7 $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | 1 | – |

**Operation:** (CY) ← 1

**Grouping:** Arithmetic operation

**Description:** Sets (1) to carry flag CY.

### SD (Set port D specified by register Y)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | $_2$ | 0 | 1 | 5 $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (D(Y)) ← 1
(Y) = 0 to 3

**Grouping:** Input/Output operation

**Description:** Sets (1) to a bit of port D specified by register Y.

**Note:** (Y) = 0 to 3.
Do not execute this instruction if values except above are set to register Y.

### SEA n (Skip Equal, Accumulator with immediate data n)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | $_2$ | 0 | 2 | 5 $_{16}$ |
| 0 | 0 | 0 | 1 | 1 | 1 | n | n | n | n | $_2$ | 0 | 7 | n $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 2 | 2 | – | (A) = n |

**Operation:** (A) = n ?
n = 0 to 15

**Grouping:** Comparison operation

**Description:** Skips the next instruction when the contents of register A is equal to the value n in the immediate field.
Executes the next instruction when the contents of register A is not equal to the value n in the immediate field.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### SEAM (Skip Equal, Accumulator with Memory)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | $_2$ | 0 2 6 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (A) = (M(DP)) |

**Operation:** (A) = (M(DP)) ?

**Grouping:** Comparison operation

**Description:** Skips the next instruction when the contents of register A is equal to the contents of M(DP).
Executes the next instruction when the contents of register A is not equal to the contents of M(DP).

### SNZ0 (Skip if Non Zero condition of external 0 interrupt request flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | $_2$ | 0 3 8 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | $V1_0$ = 0: (EXF0) = 1 |

**Operation:**
$V1_0$ = 0: (EXF0) = 1 ?
(EXF0) ← 0
$V1_0$ = 1: SNZ0 = NOP
($V1_0$ : bit 0 of the interrupt control register V1)

**Grouping:** Interrupt operation

**Description:** When $V1_0$ = 0 : Clears (0) to the EXF0 flag and skips the next instruction when external 0 interrupt request flag EXF0 is “1.” When the EXF0 flag is “0,” executes the next instruction.
When $V1_0$ = 1 : This instruction is equivalent to the NOP instruction.

### SNZAD (Skip if Non Zero condition of A/D conversion completion flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | $_2$ | 2 8 7 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | $V2_2$ = 0: (ADF) = 1 |

**Operation:**
$V2_2$ = 0: (ADF) = 1 ?
(ADF) ← 0
$V2_2$ = 1: SNZAD = NOP
($V2_2$ : bit 2 of the interrupt control register V2)

**Grouping:** A/D conversion operation

**Description:** When $V2_2$ = 0 : Clears (0) to the ADF flag and skips the next instruction when A/D conversion completion flag ADF is “1.” When the ADF flag is “0,” executes the next instruction.
When $V2_2$ = 1 : This instruction is equivalent to the NOP instruction.

### SNZI0 (Skip if Non Zero condition of external 0 Interrupt input pin)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | $_2$ | 0 3 A | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | $I1_2$ = 0 : (INT) = “L”<br>$I1_2$ = 1 : (INT) = “H” |

**Operation:**
$I1_2$ = 0 : (INT) = “L” ?
$I1_2$ = 1 : (INT) = “H” ?
($I1_2$ : bit 2 of the interrupt control register I1)

**Grouping:** Interrupt operation

**Description:** When $I1_2$ = 0 : Skips the next instruction when the level of INT pin is “L.” Executes the next instruction when the level of INT pin is “H.”
When $I1_2$ = 1 : Skips the next instruction when the level of INT pin is “H.” Executes the next instruction when the level of INT pin is “L.”

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### SNZP (Skip if Non Zero condition of Power down flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | $_2$ | 0 | 0 | 3 $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (P) = 1 |

**Operation:** (P) = 1 ?

**Grouping:** Other operation

**Description:** Skips the next instruction when the P flag is "1".
After skipping, the P flag remains unchanged.
Executes the next instruction when the P flag is "0."

### SNZSI (Skip if Non Zero condition of Serial Interface interrupt request flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | $_2$ | 2 | 8 | 8 $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | $V2_3$ = 0: (SIOF) =1 |

**Operation:** $V2_3$=0: (SIOF)=1?
(SIOF) ← 0
$V2_3$ = 1: SNZSI = NOP

**Grouping:** Serial interface operation

**Description:** Clears (0) to SIOF flag and skips the next instruction when the contents of bit 3 ($V2_3$) of interrupt control register V2 is "0" and contents of SIOF flag is "1."
When $V2_3$ = 1: This instruction is equivalent to the NOP instruction.

### SNZT1 (Skip if Non Zero condition of Timer 1 interrupt request flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $_2$ | 2 | 8 | 0 $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | $V1_2$ = 0: (T1F) = 1 |

**Operation:** $V1_2$ = 0: (T1F) = 1 ?
(T1F) ← 0
$V1_2$ = 1: SNZT1 = NOP
($V1_2$ = bit 2 of interrupt control register V1)

**Grouping:** Timer operation

**Description:** When $V1_2$ = 0 : Clears (0) to the T1F flag and skips the next instruction when timer 1 interrupt request flag T1F is "1." When the T1F flag is "0," executes the next instruction.
When $V1_2$ = 1 : This instruction is equivalent to the NOP instruction.

### SNZT2 (Skip if Non Zero condition of Timer 2 interrupt request flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | $_2$ | 2 | 8 | 1 $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | $V1_3$ = 0: (T2F) = 1 |

**Operation:** $V1_3$ = 0: (T2F) = 1 ?
(T2F) ← 0
$V1_3$ = 1: SNZT2 = NOP
($V1_3$ = bit 3 of interrupt control register V1)

**Grouping:** Timer operation

**Description:** When $V1_3$ = 0 : Clears (0) to the T2F flag and skips the next instruction when timer 2 interrupt request flag T2F is "1." When the T2F flag is "0," executes the next instruction.
When $V1_3$ = 1 : This instruction is equivalent to the NOP instruction.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### SRST (System ReSet)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | $_2$ | 0 | 0 | 1 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** System reset

**Grouping:** Other operation

**Description:** System reset occurs.

### SST (Serial interface transmission/reception STart)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | $_2$ | 2 | 9 | E | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (SIOF) ← 0
Serial interface transmit/receive starting

**Grouping:** Serial interface operation

**Description:** Clears (0) to SIOF flag and starts serial interface.

### SUPT (Set UPT flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | $_2$ | 0 | 5 | 9 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (UPTF) ← 1

**Grouping:** Other operation

**Description:** Sets (1) to the high-order bit reference enable flag UPTF. When the table reference instruction (TABP p) is executed, the high-order 2 bits of ROM reference data is transferred to the low-order 2 bits of register D.

### SVDE (Set Voltage Detector Enable flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | $_2$ | 2 | 9 | 3 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** Voltage drop detection circuit valid at RAM back-up

**Grouping:** Other operation

**Description:** Validates the voltage drop detection circuit at RAM back-up.

**Note:** This instruction can be executed only for the H version.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### SZB j (Skip if Zero, Bit)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | j | j | $_2$ | 0 2 j | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (Mj(DP)) = 0<br>j = 0 to 3 |

**Operation:** (Mj(DP)) = 0 ?
j = 0 to 3

**Grouping:** Bit operation

**Description:** 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.”
Executes the next instruction when the contents of bit j of M(DP) is “1.”

### SZC (Skip if Zero, Carry flag)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | $_2$ | 0 2 F | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (CY) = 0 |

**Operation:** (CY) = 0 ?

**Grouping:** Arithmetic operation

**Description:** Skips the next instruction when the contents of carry flag CY is “0.”
After skipping, the CY flag remains unchanged.
Executes the next instruction when the contents of the CY flag is “1.“

### SZD (Skip if Zero, port D specified by register Y)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | $_2$ | 0 2 4 | $_{16}$ | |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | $_2$ | 0 2 B | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 2 | 2 | – | (D(Y)) = 0<br>(Y) = 0 to 3 |

**Operation:** (D(Y)) = 0 ?
(Y) = 0 to 3

**Grouping:** Input/Output operation

**Description:** Skips the next instruction when a bit of port D specified by register Y is “0.” Executes the next instruction when the bit is “1.”

**Note:** (Y) = 0 to 3.
Do not execute this instruction if values except above are set to register Y.

### T1AB (Transfer data to timer 1 and register R1L from Accumulator and register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | $_2$ | 2 3 0 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($R1L_7$–$R1L_4$) ← (B)
($T1_7$–$T1_4$) ← (B)
($R1L_3$–$R1L_0$) ← (A)
($T1_3$–$T1_0$) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register B to the high-order 4 bits of timer 1 and timer 1 reload register R1L. Transfers the contents of register A to the low-order 4 bits of timer 1 and timer 1 reload register R1L.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### T1HAB (Transfer data to register R1H from Accumulator and register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | $_2$ | 2 | 9 | 2 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (R1H7–R1H4) ← (B)
(R1H3–R1H0) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register B to the high-order 4 bits of timer 1 reload register R1H. Transfers the contents of register A to the low-order 4 bits of timer 1 reload register R1H.

### T1R1L (Transfer data to timer 1 from register R1L)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | $_2$ | 2 | A | 7 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (T17–T10) ← (R1L7–R1L0)

**Grouping:** Timer operation

**Description:** Transfers the contents of timer 1 reload register R1L to timer 1.

### T2AB (Transfer data to timer 2 and register R2L from Accumulator and register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | $_2$ | 2 | 3 | 1 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (R2L7–R2L4) ← (B)
(T27–T24) ← (B)
(R2L3–R2L0) ← (A)
(T23–T20) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register B to the high-order 4 bits of timer 2 and timer 2 reload register R2L. Transfers the contents of register A to the low-order 4 bits of timer 2 and timer 2 reload register R2L.

### T2HAB (Transfer data to register R2H from Accumulator and register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | $_2$ | 2 | 9 | 4 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (R2H7–R2H4) ← (B)
(R2H3–R2H0) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register B to the high-order 4 bits of timer 2 reload register R2H. Transfers the contents of register A to the low-order 4 bits of timer 2 reload register R2H.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### T2R2L (Transfer data to timer 2 from register R2L)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | $_2$ | 2 | 9 | 5 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($T2_7$–$T2_0$) ← ($R2L_7$–$R2L_0$)

**Grouping:** Timer operation

**Description:** Transfers the contents of timer 2 reload register R2L to timer 2.

### TAB (Transfer data to Accumulator from register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | $_2$ | 0 | 1 | E | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (B)

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register B to register A.

### TAB1 (Transfer data to Accumulator and register B from timer 1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | $_2$ | 2 | 7 | 0 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (B) ← ($T1_7$–$T1_4$)
(A) ← ($T1_3$–$T1_0$)

**Grouping:** Timer operation

**Description:** Transfers the high-order 4 bits ($T1_7$–$T1_4$) of timer 1 to register B.
Transfers the low-order 4 bits ($T1_3$–$T1_0$) of timer 1 to register A.

### TAB2 (Transfer data to Accumulator and register B from timer 2)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | $_2$ | 2 | 7 | 1 | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (B) ← ($T2_7$–$T2_4$)
(A) ← ($T2_3$–$T2_0$)

**Grouping:** Timer operation

**Description:** Transfers the high-order 4 bits ($T2_7$–$T2_4$) of timer 2 to register B.
Transfers the low-order 4 bits ($T2_3$–$T2_0$) of timer 2 to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TABAD (Transfer data to Accumulator and register B from register AD)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 2 | 7 | 9 |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:**
In A/D conversion mode (Q1$_3$ = 0),
(B) ← (AD$_9$–AD$_6$)
(A) ← (AD$_5$–AD$_2$)
In comparator mode (Q1$_3$ = 1),
(B) ← (AD$_7$–AD$_4$)
(A) ← (AD$_3$–AD$_0$)
(Q1$_3$ : bit 3 of A/D control register Q1)

**Grouping:** A/D conversion operation

**Description:** In the A/D conversion mode (Q1$_3$ = 0), transfers the high-order 4 bits (AD$_9$–AD$_6$) of register AD to register B, and the middle-order 4 bits (AD$_5$–AD$_2$) of register AD to register A. In the comparator mode (Q1$_3$ = 1), transfers the high-order 4 bits (AD$_7$–AD$_4$) of comparator register to register B, and the low-order 4 bits (AD$_3$–AD$_0$) of comparator register to register A.

### TABE (Transfer data to Accumulator and register B from register E)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | A |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:**
(B) ← (E$_7$–E$_4$)
(A) ← (E$_3$–E$_0$)

**Grouping:** Register to register transfer

**Description:** Transfers the high-order 4 bits (E$_7$–E$_4$) of register E to register B, and low-order 4 bits of register E to register A.

### TABP p (Transfer data to Accumulator and register B from Program memory in page p)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | p$_4$ | p$_3$ | p$_2$ | p$_1$ | p$_0$ | 0 | 8 +p | p |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 3 | – | – |

**Operation:**
(SP) ← (SP) + 1
(SK(SP)) ← (PC)
(PC$_H$) ← p
(PC$_L$) ← (DR$_2$–DR$_0$, A$_3$–A$_0$)
(B) ← (ROM(PC))$_{7–4}$
(A) ← (ROM(PC))$_{3–0}$
(UPTF) ← 1
(DR$_1$, DR$_0$) ← (ROM(PC))$_{9, 8}$
(DR$_2$) ← 0
(PC) ← (SK(SP))
(SP) ← (SP) – 1

**Grouping:** Arithmetic operation

**Description:** Transfers bits 7 to 4 to register B and bits 3 to 0 to register A. These bits 7 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. When UPTF is 1, Transfers bits 9, 8 to the low-order 2 bits (DR$_1$, DR$_0$) of register D, and "0" is stored to the least significant bit (DR$_2$) of register D.
When this instruction is executed, 1 stage of stack register (SK) is used.

**Note:** p is 0 to 31.
When this instruction is executed, be careful not to over the stack because 1 stage of stack register is used.

### TABPS (Transfer data to Accumulator and register B from Pre-Scaler)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 2 | 7 | 5 |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:**
(B) ← (TPS$_7$–TPS$_4$)
(A) ← (TPS$_3$–TPS$_0$)

**Grouping:** Timer operation

**Description:** Transfers the high-order 4 bits of prescaler to register B.
Transfers the low-order 4 bits of prescaler to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TABSI (Transfer data to Accumulator and register B from register SI)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 2 | 7 | 8 |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (B) ← ($SI_7$–$SI_4$) (A) ← ($SI_3$–$SI_0$)

**Grouping:** Serial interface operation

**Description:** Transfers the high-order 4 bits of serial interface register SI to register B, and transfers the low-order 4 bits of serial interface register SI to register A.

### TAD (Transfer data to Accumulator from register D)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 5 | 1 |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($A_2$–$A_0$) ← ($DR_2$–$DR_0$)
($A_3$) ← 0

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register D to the low-order 3 bits ($A_2$–$A_0$) of register A.

**Note:** When this instruction is executed, "0" is stored to the bit 3 ($A_3$) of register A.

### TADAB (Transfer data to register AD from Accumulator from register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 2 | 3 | 9 |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** $Q1_3$ = 1: ($AD_7$–$AD_4$) ← (B)
($AD_3$–$AD_0$) ← (A)
$Q1_3$ = 0: TADAB = NOP

**Grouping:** A/D conversion operation

**Description:** In the comparator mode ($Q1_3$ = 1), transfers the contents of register B to the high-order 4 bits ($AD_7$–$AD_4$) of comparator register, and the contents of register A to the low-order 4 bits ($AD_3$–$AD_0$) of comparator register.
In the A/D conversion mode ($Q1_3$ = 0), this instruction is equivalent to the NOP instruction.
($Q1_3$ = bit 3 of A/D control register Q1)

### TAI1 (Transfer data to Accumulator from register I1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 2 | 5 | 3 |
| $_2$ | | | | | | | | | | $_{16}$ | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (I1)

**Grouping:** Interrupt operation

**Description:** Transfers the contents of interrupt control register I1 to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TAJ1 (Transfer data to Accumulator from register J1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 4 | 2 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (J1)

**Grouping:** Serial interface operation

**Description:** Transfers the contents of serial interface control register J1 to register A.

### TAK0 (Transfer data to Accumulator from register K0)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 2 | 5 | 6 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (K0)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of key-on wakeup control register K0 to register A.

### TAK1 (Transfer data to Accumulator from register K1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 2 | 5 | 9 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (K1)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of key-on wakeup control register K1 to register A.

### TAK2 (Transfer data to Accumulator from register K2)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 | 5 | A |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (K2)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of key-on wakeup control register K2 to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TAL1 (Transfer data to Accumulator from register L1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 2 | 4 | A |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (L1)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of key-on wakeup control register L1 to register A.

### TALA (Transfer data to Accumulator from register LA)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 4 | 9 |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($A_3$, $A_2$) ← ($AD_1$, $AD_0$)
($A_1$, $A_0$) ← 0

**Grouping:** A/D conversion operation

**Description:** Transfers the low-order 2 bits ($AD_1$, $AD_0$) of register AD to the high-order 2 bits ($A_3$, $A_2$) of register A. “0” is stored to the low-order 2 bits ($A_1$, $A_0$) of register A.

### TAM j (Transfer data to Accumulator from Memory)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | j | j | j | j | 2 | C | j |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (M(DP))
(X) ← (X)EXOR(j)
j = 0 to 15

**Grouping:** RAM to register transfer

**Description:** 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.

### TAMR (Transfer data to Accumulator from register MR)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 5 | 2 |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (MR)

**Grouping:** Clock operation

**Description:** Transfers the contents of clock control register MR to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TAPU0 (Transfer data to Accumulator from register PU0)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 2 | 5 | 7 | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (PU0)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of pull-up control register PU0 to register A.

### TAPU1 (Transfer data to Accumulator from register PU1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 2 | 5 | E | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (PU1)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of pull-up control register PU1 to register A.

### TAPU2 (Transfer data to Accumulator from register PU2)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 5 | F | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (PU2)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of pull-up control register PU2 to register A.

### TAQ1 (Transfer data to Accumulator from register Q1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 4 | 4 | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (Q1)

**Grouping:** A/D conversion operation

**Description:** Transfers the contents of A/D control register Q1 to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TASP (Transfer data to Accumulator from Stack Pointer)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | $_2$ | 0 5 0 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($A_2$–$A_0$) ← ($SP_2$–$SP_0$)
($A_3$) ← 0

**Grouping:** Register to register transfer

**Description:** Transfers the contents of stack pointer (SP) to the low-order 3 bits ($A_2$–$A_0$) of register A. "0" is stored to the bit 3 ($A_3$) of register A.

### TAV1 (Transfer data to Accumulator from register V1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | $_2$ | 0 5 4 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (V1)

**Grouping:** Interrupt operation

**Description:** Transfers the contents of interrupt control register V1 to register A.

### TAV2 (Transfer data to Accumulator from register V2)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | $_2$ | 0 5 5 | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (V2)

**Grouping:** Interrupt operation

**Description:** Transfers the contents of interrupt control register V2 to register A.

### TAW1 (Transfer data to Accumulator from register W1)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | $_2$ | 2 4 B | $_{16}$ | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (W1)

**Grouping:** Timer operation

**Description:** Transfers the contents of timer control register W1 to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TAW2 (Transfer data to Accumulator from register W2)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 4 | C | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (W2)

**Grouping:** Timer operation

**Description:** Transfers the contents of timer control register W2 to register A.

### TAW5 (Transfer data to Accumulator from register W5)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 4 | F | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (W5)

**Grouping:** Timer operation

**Description:** Transfers the contents of timer control register W5 to register A.

### TAW6 (Transfer data to Accumulator from register W6)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 5 | 0 | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (W6)

**Grouping:** Timer operation

**Description:** Transfers the contents of timer control register W6 to register A.

### TAX (Transfer data to Accumulator from register X)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5 | 2 | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (X)

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register X to register A.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TAY (Transfer data to Accumulator from register Y)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 0 | 1 | F | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ← (Y)

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register Y to register A.

### TAZ (Transfer data to Accumulator from register Z)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 2 | 0 | 5 | 3 | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($A_1$, $A_0$) ← ($Z_1$, $Z_0$)
($A_3$, $A_2$) ← 0

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register Z to the low-order 2 bits ($A_1$, $A_0$) of register A. "0" is stored to the high-order 2 bits ($A_3$, $A_2$) of register A.

### TBA (Transfer data to register B from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 2 | 0 | 0 | E | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (B) ← (A)

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register A to register B.

### TDA (Transfer data to register D from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2 | 0 | 2 | 9 | 16 |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($DR_2$–$DR_0$) ← ($A_2$–$A_0$)

**Grouping:** Register to register transfer

**Description:** Transfers the contents of the low-order 3 bits ($A_2$–$A_0$) of register A to register D.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TEAB (Transfer data to register E from Accumulator and register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | A | |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($E_7$–$E_4$) ← (B)
($E_3$–$E_0$) ← (A)

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register B to the high-order 4 bits ($E_3$–$E_0$) of register E, and the contents of register A to the low-order 4 bits ($E_3$–$E_0$) of register E.

### TFR0A (Transfer data to register FR0 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 2 | 2 | 8 | |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (FR0) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to port output structure control register FR0.

### TFR1A (Transfer data to register FR1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2 | 2 | 9 | |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (FR1) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to port output structure control register FR1.

### TFR2A (Transfer data to register FR2 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 2 | 2 | A | |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (FR2) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to port output structure control register FR2.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TFR3A (Transfer data to register FR3 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 2 | 2 | B |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (FR3) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to port output structure control register FR3.

### TI1A (Transfer data to register I1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 2 | 1 | 7 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (I1) ← (A)

**Grouping:** Interrupt operation

**Description:** Transfers the contents of register A to interrupt control register I1.

### TJ1A (Transfer data to register J1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 2 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (J1) ← (A)

**Grouping:** Serial interface operation

**Description:** Transfers the contents of register A to serial interface control register J1.

### TK0A (Transfer data to register K0 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | B |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (K0) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to key-on wakeup control register K0.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TK1A (Transfer data to register K1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 1 | 4 |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (K1) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to key-on wakeup control register K1.

### TK2A (Transfer data to register K2 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 2 | 1 | 5 |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (K2) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to key-on wakeup control register K2.

### TL1A (Transfer data to register L1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 2 | 0 | A |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (L1) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to key-on wakeup control register L1.

### TMA j (Transfer data to Memory from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 | j | j | j | j | 2 | B | j |
| $_2$ | | | | | | | | | | $_{16}$ | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (M(DP)) ← (A)
(X) ← (X)EXOR(j)
j = 0 to 15

**Grouping:** RAM to register transfer

**Description:** 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.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TMRA (Transfer data to register MR from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 2 | 1 | 6 | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (MR) ← (A)

**Grouping:** Clock operation

**Description:** Transfers the contents of register A to clock control register MR.

### TPAA (Transfer data to register PA from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 2 | A | A | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($PA_0$) ← ($A_0$)

**Grouping:** Timer operation

**Description:** Transfers the least significant bit of register A to timer control register PA.

### TPSAB (Transfer data to Pre-Scaler and register RPS from Accumulator and register B)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 2 | 3 | 5 | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:**
($RPS_7$–$RPS_4$) ← (B)
($TPS_7$–$TPS_4$) ← (B)
($RPS_3$–$RPS_0$) ← (A)
($TPS_3$–$TPS_0$) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register B to the high-order 4 bits of prescaler and prescaler reload register RPS. Transfers the contents of register A to the low-order 4 bits of prescaler and prescaler reload register RPS.

### TPU0A (Transfer data to register PU0 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 2 | 2 | D | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (PU0) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to pull-up control register PU0.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TPU1A (Transfer data to register PU1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 2 | 2 | E | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (PU1) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to pull-up control register PU1.

### TPU2A (Transfer data to register PU2 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | F | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (PU2) ← (A)

**Grouping:** Input/Output operation

**Description:** Transfers the contents of register A to pull-up control register PU2.

### TQ1A (Transfer data to register Q1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 4 | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (Q1) ← (A)

**Grouping:** A/D conversion operation

**Description:** Transfers the contents of register A to A/D control register Q1.

### TRGA (Transfer data to register RG from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 0 | 9 | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($RG_0$) ← ($A_0$)

**Grouping:** Clock operation

**Description:** Transfers the least significant bit (A0) of register A to clock control regiser RG.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TSIAB (Transfer data to register SI from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 2 | 3 | 8 | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** ($SI_7$–$SI_4$) ← (B) ($SI_3$–$SI_0$) ← (A)

**Grouping:** Serial interface operation

**Description:** Transfers the contents of register B to the high-order 4 bits of serial interface register SI, and transfers the contents of register A to the low-order 4 bits of serial interface register SI.

### TV1A (Transfer data to register V1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 3 | F | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (V1) ← (A)

**Grouping:** Interrupt operation

**Description:** Transfers the contents of register A to interrupt control register V1.

### TV2A (Transfer data to register V2 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 3 | E | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (V2) ← (A)

**Grouping:** Interrupt operation

**Description:** Transfers the contents of register A to interrupt control register V2.

### TW1A (Transfer data to register W1 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 2 | 0 | E | 16 |
| | | | | | | | | | 2 | | | | |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (W1) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register A to timer control register W1.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### TW2A (Transfer data to register W2 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 0 | F |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (W2) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register A to timer control register W2.

### TW5A (Transfer data to register W5 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 1 | 2 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (W5) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register A to timer control register W5.

### TW6A (Transfer data to register W6 from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 2 | 1 | 3 |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (W6) ← (A)

**Grouping:** Timer operation

**Description:** Transfers the contents of register A to timer control register W6.

### TYA (Transfer data to register Y from Accumulator)

**Instruction code**

| D9 | | | | | | | | | D0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | C |

$_2$ $_{16}$

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (Y) ← (A)

**Grouping:** Register to register transfer

**Description:** Transfers the contents of register A to register Y.

## MACHINE INSTRUCTIONS (INDEX BY ALPHABET) (continued)

### WRST (Watchdog timer ReSeT)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | A | 0 | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (WDF1) = 1 |

**Operation:** (WDF1) = 1 ?
(WDF1) ← 0

**Grouping:** Other operation

**Description:** Clears (0) to the WDF1 flag and skips the next instruction when watchdog timer flag WDF1 is "1." When the WDF1 flag is "0," executes the next instruction. Also, stops the watchdog timer function when executing the WRST instruction immediately after the DWDT instruction.

### XAM j (eXchange Accumulator and Memory data)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 | j | j | j | j | 2 | D | j | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | – |

**Operation:** (A) ←→ (M(DP))
(X) ← (X)EXOR(j)
j = 0 to 15

**Grouping:** RAM to register transfer

**Description:** 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 (eXchange Accumulator and Memory data and Decrement register Y and skip)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 | j | j | j | j | 2 | F | j | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (Y) = 15 |

**Operation:** (A) ←→ (M(DP))
(X) ← (X)EXOR(j)
j = 0 to 15
(Y) ← (Y) – 1

**Grouping:** RAM to register transfer

**Description:** 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. When the contents of register Y is not 15, the next instruction is executed.

### XAMI j (eXchange Accumulator and Memory data and Increment register Y and skip)

**Instruction code**

| D9 | | | | | | | | | D0 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | j | j | j | j | 2 | E | j | |
| | | | | | | | | | $_2$ | | | | $_{16}$ |

| Number of words | Number of cycles | Flag CY | Skip condition |
|---|---|---|---|
| 1 | 1 | – | (Y) = 0 |

**Operation:** (A) ←→ (M(DP))
(X) ← (X)EXOR(j)
j = 0 to 15
(Y) ← (Y) + 1

**Grouping:** RAM to register transfer

**Description:** 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. When the contents of register Y is not 0, the next instruction is executed.

## MACHINE INSTRUCTIONS (INDEX BY TYPES)

| Type of instructions | Mnemonic | Instruction code | | | | | | | | | | Hexadecimal notation | Number of words | Number of cycles | Function |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $D_9$ | $D_8$ | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | | | | |
| Register to register transfer | TAB | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 1 E | 1 | 1 | (A) ← (B) |
| | TBA | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 0 E | 1 | 1 | (B) ← (A) |
| | TAY | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 1 F | 1 | 1 | (A) ← (Y) |
| | TYA | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 0 C | 1 | 1 | (Y) ← (A) |
| | 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) |
| | TABE | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 2 A | 1 | 1 | (B) ← ($E_7$–$E_4$)<br>(A) ← ($E_3$–$E_0$) |
| | TDA | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 2 9 | 1 | 1 | ($DR_2$–$DR_0$) ← ($A_2$–$A_0$) |
| | 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 |
| | 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 |
| | TAX | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 5 2 | 1 | 1 | (A) ← (X) |
| | 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 |
| 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 |
| | 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 |
| | INY | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 1 3 | 1 | 1 | (Y) ← (Y) + 1 |
| | DEY | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 1 7 | 1 | 1 | (Y) ← (Y) – 1 |
| 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 |
| | 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 |
| | 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 |
| | 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 |
| | 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 |

| Skip condition | Carry flag CY | Datailed description |
|---|---|---|
| – | – | Transfers the contents of register B to register A. |
| – | – | Transfers the contents of register A to register B. |
| – | – | Transfers the contents of register Y to register A. |
| – | – | Transfers the contents of register A to register Y. |
| – | – | Transfers the contents of register B to the high-order 4 bits ($E_7$–$E_4$) of register E, and the contents of register A to the low-order 4 bits ($E_3$–$E_0$) of register E. |
| – | – | Transfers the high-order 4 bits ($E_7$–$E_4$) of register E to register B, and low-order 4 bits of register E to register A. |
| – | – | Transfers the contents of the low-order 3 bits ($A_2$–$A_0$) of register A to register D. |
| – | – | Transfers the contents of register D to the low-order 3 bits ($A_2$–$A_0$) of register A.<br>"0" is stored to the bit 3 ($A_3$) of register A. |
| – | – | Transfers the contents of register Z to the low-order 2 bits ($A_1$, $A_0$) of register A.<br>"0" is stored to the high-order 2 bits ($A_3$, $A_2$) of register A. |
| – | – | Transfers the contents of register X to register A. |
| – | – | Transfers the contents of stack pointer (SP) to the low-order 3 bits ($A_2$–$A_0$) of register A.<br>"0" is stored to the bit 3 ($A_3$) of register A. |
| 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. |
| – | – | Loads the value z in the immediate field to register Z. |
| (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. When the contents of register Y is not 0, the next instruction is executed. |
| (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. When the contents of register Y is not 15, the next instruction is executed. |
| – | – | 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. |
| – | – | 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. |
| (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.<br>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. When the contents of register Y is not 15, the next instruction is executed. |
| (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.<br>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. when the contents of register Y is not 0, the next instruction is executed. |
| – | – | 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. |

## MACHINE INSTRUCTIONS (INDEX BY TYPES) (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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| | TABP p | 0 | 0 | 1 | 0 | 0 | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ | 0 8 p+p | 1 | 3 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>($PC_H$) ← p (Note)<br>($PC_L$) ← ($DR_2$–$DR_0$, $A_3$–$A_0$)<br>(B) ← (ROM(PC))$_{7–4}$<br>(A) ← (ROM(PC))$_{3–0}$<br>(UPTF) = 1<br>($DR_1$, $DR_0$) ← (ROM(PC))$_{9, 8}$<br>($DR_2$) ← 0<br>(PC) ← (SK(SP))<br>(SP) ← (SP) – 1 |
| | AM | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 0 A | 1 | 1 | (A) ← (A) + (M(DP)) |
| | AMC | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 0 B | 1 | 1 | (A) ← (A) + (M(DP)) +(CY)<br>(CY) ← Carry |
| | 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 |
| | AND | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 1 8 | 1 | 1 | (A) ← (A) AND (M(DP)) |
| | OR | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 1 9 | 1 | 1 | (A) ← (A) OR (M(DP)) |
| | SC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 0 7 | 1 | 1 | (CY) ← 1 |
| | RC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 0 6 | 1 | 1 | (CY) ← 0 |
| | SZC | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 2 F | 1 | 1 | (CY) = 0 ? |
| | CMA | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 1 C | 1 | 1 | (A) ← ($\overline{A}$) |
| | RAR | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 1 D | 1 | 1 | → CY → $A_3A_2A_1A_0$ → |
| 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 |
| | 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 |
| | 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 |
| Comparison operation | SEAM | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 2 6 | 1 | 1 | (A) = (M(DP)) ? |
| | SEA n | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 2 5 | 2 | 2 | (A) = n ?<br>n = 0 to 15 |
| | | 0 | 0 | 0 | 1 | 1 | 1 | n | n | n | n | 0 7 n | | | |

Note : p is 0 to 31.

| Skip condition | Carry flag CY | Datailed description |
|---|---|---|
| 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. |
| – | – | Transfers bits 7 to 4 to register B and bits 3 to 0 to register A. These bits 7 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. When UPTF is 1, Transfers bits 9, 8 to the low-order 2 bits ($DR_1$, $DR_0$) of register D, and "0" is stored to the least significant bit ($DR_2$) of register D.<br>When this instruction is executed, 1 stage of stack register (SK) is used. |
| – | – | Adds the contents of M(DP) to register A. Stores the result in register A. The contents of carry flag CY remains unchanged. |
| – | 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. |
| Overflow = 0 | – | Adds the value n in the immediate field to register A, and stores a result in 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.<br>Executes the next instruction when there is overflow as the result of operation. |
| – | – | Takes the AND operation between the contents of register A and the contents of M(DP), and stores the result in register A. |
| – | – | Takes the OR operation between the contents of register A and the contents of M(DP), and stores the result in register A. |
| – | 1 | Sets (1) to carry flag CY. |
| – | 0 | Clears (0) to carry flag CY. |
| (CY) = 0 | – | Skips the next instruction when the contents of carry flag CY is "0." |
| – | – | Stores the one's complement for register A's contents in register A. |
| – | 0/1 | Rotates 1 bit of the contents of register A including the contents of carry flag CY to the right. |
| – | – | Sets (1) the contents of bit j (bit specified by the value j in the immediate field) of M(DP). |
| – | – | Clears (0) the contents of bit j (bit specified by the value j in the immediate field) of M(DP). |
| (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."<br>Executes the next instruction when the contents of bit j of M(DP) is "1." |
| (A) = (M(DP)) | – | Skips the next instruction when the contents of register A is equal to the contents of M(DP).<br>Executes the next instruction when the contents of register A is not equal to the contents of M(DP). |
| (A) = n<br>n = 0 to 15 | – | Skips the next instruction when the contents of register A is equal to the value n in the immediate field.<br>Executes the next instruction when the contents of register A is not equal to the value n in the immediate field. |

## MACHINE INSTRUCTIONS (INDEX BY TYPES) (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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Branch operation | B a | 0 | 1 | 1 | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 1 8 a +a | 1 | 1 | $(PC_L) \leftarrow a_6$–$a_0$ |
| | BL p, a | 0 | 0 | 1 | 1 | 1 | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ | 0 E p +p | 2 | 2 | $(PC_H) \leftarrow$ p (Note)<br>$(PC_L) \leftarrow a_6$–$a_0$ |
| | | 1 | 0 | 0 | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 2 a a | | | |
| | BLA p | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 1 0 | 2 | 2 | $(PC_H) \leftarrow$ p (Note)<br>$(PC_L) \leftarrow (DR_2$–$DR_0, A_3$–$A_0)$ |
| | | 1 | 0 | 0 | $p_4$ | 0 | 0 | $p_3$ | $p_2$ | $p_1$ | $p_0$ | 2 p p | | | |
| Subroutine operation | BM a | 0 | 1 | 0 | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 1 a a | 1 | 1 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>$(PC_H) \leftarrow 2$<br>$(PC_L) \leftarrow a_6$–$a_0$ |
| | BML p, a | 0 | 0 | 1 | 1 | 0 | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ | 0 C p +p | 2 | 2 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC)<br>$(PC_H) \leftarrow$ p (Note) |
| | | 1 | 0 | 0 | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 2 a a | | | $(PC_L) \leftarrow a_6$–$a_0$ |
| | BMLA p | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 3 0 | 2 | 2 | (SP) ← (SP) + 1<br>(SK(SP)) ← (PC) |
| | | 1 | 0 | 0 | $p_4$ | 0 | 0 | $p_3$ | $p_2$ | $p_1$ | $p_0$ | 2 p p | | | $(PC_H) \leftarrow$ p (Note)<br>$(PC_L) \leftarrow (DR_2$–$DR_0, A_3$–$A_0)$ |
| 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 |
| | RT | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 4 4 | 1 | 2 | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 |
| | RTS | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 4 5 | 1 | 2 | (PC) ← (SK(SP))<br>(SP) ← (SP) – 1 |

Note : p is 0 to 31.

| Skip condition | Carry flag CY | Datailed description |
|---|---|---|
| – | – | Branch within a page : Branches to address a in the identical page. |
| – | – | Branch out of a page : Branches to address a in page p. |
| – | – | 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. |
| – | – | Call the subroutine in page 2 : Calls the subroutine at address a in page 2. |
| – | – | Call the subroutine : Calls the subroutine at address a in page p. |
| – | – | 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. |
| – | – | 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. |
| – | – | Returns from subroutine to the routine called the subroutine. |
| Skip at uncondition | – | Returns from subroutine to the routine called the subroutine, and skips the next instruction at uncondition. |

## MACHINE INSTRUCTIONS (INDEX BY TYPES) (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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Interrupt operation | DI | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 0 4 | 1 | 1 | (INTE) ← 0 |
| | EI | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 0 5 | 1 | 1 | (INTE) ← 1 |
| | SNZ0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 3 8 | 1 | 1 | $V1_0$ = 0: (EXF0) = 1 ?<br>(EXF0) ← 0<br>$V1_0$ = 1: SNZ0 = NOP |
| | SNZI0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 3 A | 1 | 1 | $I1_2$ = 0 : (INT) = "L" ?<br>$I1_2$ = 1 : (INT) = "H" ? |
| | TAV1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 5 4 | 1 | 1 | (A) ← (V1) |
| | TV1A | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 3 F | 1 | 1 | (V1) ← (A) |
| | TAV2 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 5 5 | 1 | 1 | (A) ← (V2) |
| | TV2A | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 3 E | 1 | 1 | (V2) ← (A) |
| | TAI1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 2 5 3 | 1 | 1 | (A) ← (I1) |
| | TI1A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 2 1 7 | 1 | 1 | (I1) ← (A) |
| Timer operation | TPAA | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 2 A A | 1 | 1 | ($PA_0$) ← ($A_0$) |
| | TAW1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 2 4 B | 1 | 1 | (A) ← (W1) |
| | TW1A | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 2 0 E | 1 | 1 | (W1) ← (A) |
| | TAW2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 2 4 C | 1 | 1 | (A) ← (W2) |
| | TW2A | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 0 F | 1 | 1 | (W2) ← (A) |
| | TAW5 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 2 4 F | 1 | 1 | (A) ← (W5) |
| | TW5A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 1 2 | 1 | 1 | (W5) ← (A) |
| | TAW6 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 2 5 0 | 1 | 1 | (A) ← (W6) |
| | TW6A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 2 1 3 | 1 | 1 | (W6) ← (A) |
| | TABPS | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 2 7 5 | 1 | 1 | (B) ← ($TPS_7$–$TPS_4$)<br>(A) ← ($TPS_3$–$TPS_0$) |
| | TPSAB | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 2 3 5 | 1 | 1 | ($RPS_7$–$RPS_4$) ← (B)<br>($TPS_7$–$TPS_4$) ← (B)<br>($RPS_3$–$RPS_0$) ← (A)<br>($TPS_3$–$TPS_0$) ← (A) |
| | TAB1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 2 7 0 | 1 | 1 | (B) ← ($T1_7$–$T1_4$)<br>(A) ← ($T1_3$–$T1_0$) |
| | T1AB | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 2 3 0 | 1 | 1 | ($R1L_7$–$R1L_4$) ← (B)<br>($T1_7$–$T1_4$) ← (B)<br>($R1L_3$–$R1L_0$) ← (A)<br>($T1_3$–$T1_0$) ← (A) |
| | T1HAB | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 9 2 | 1 | 1 | ($R1H_7$–$R1H_4$) ← (B)<br>($R1H_3$–$R1H_0$) ← (A) |

| Skip condition | Carry flag CY | Datailed description |
|---|---|---|
| – | – | Clears (0) to interrupt enable flag INTE, and disables the interrupt. |
| – | – | Sets (1) to interrupt enable flag INTE, and enables the interrupt. |
| $V1_0$ = 0: (EXF0) = 1 | – | When $V1_0$ = 0 : Clears (0) to the EXF0 flag and skips the next instruction when external 0 interrupt request flag EXF0 is “1.” When the EXF0 flag is “0,” executes the next instruction.<br>When $V1_0$ = 1 : This instruction is equivalent to the NOP instruction. ($V1_0$: bit 0 of interrupt control register V1) |
| (INT) = “L”<br>However, $I1_2$ = 0 | – | When $I1_2$ = 0 : Skips the next instruction when the level of INT pin is “L.” Executes the next instruction when the level of INT pin is “H.” |
| (INT) = “H”<br>However, $I1_2$ = 1 | | When $I1_2$ = 1 : Skips the next instruction when the level of INT pin is “H.” Executes the next instruction when the level of INT pin is “L.” ($I1_2$: bit 2 of interrupt control register I1) |
| – | – | Transfers the contents of interrupt control register V1 to register A. |
| – | – | Transfers the contents of register A to interrupt control register V1. |
| – | – | Transfers the contents of interrupt control register V2 to register A. |
| – | – | Transfers the contents of register A to interrupt control register V2. |
| – | – | Transfers the contents of interrupt control register I1 to register A. |
| – | – | Transfers the contents of register A to interrupt control register I1. |
| – | – | Transfers the contents of register A to timer control register PA. |
| – | – | Transfers the contents of timer control register W1 to register A. |
| – | – | Transfers the contents of register A to timer control register W1. |
| – | – | Transfers the contents of timer control register W2 to register A. |
| – | – | Transfers the contents of register A to timer control register W2. |
| – | – | Transfers the contents of timer control register W5 to register A. |
| – | – | Transfers the contents of register A to timer control register W5. |
| – | – | Transfers the contents of timer control register W6 to register A. |
| – | – | Transfers the contents of register A to timer control register W6. |
| – | – | Transfers the high-order 4 bits of prescaler to register B.<br>Transfers the low-order 4 bits of prescaler to register A. |
| – | – | Transfers the contents of register B to the high-order 4 bits of prescaler and prescaler reload register RPS.<br>Transfers the contents of register A to the low-order 4 bits of prescaler and prescaler reload register RPS. |
| – | – | Transfers the high-order 4 bits ($T1_7$–$T1_4$) of timer 1 to register B.<br>Transfers the low-order 4 bits ($T1_3$–$T1_0$) of timer 1 to register A. |
| – | – | Transfers the contents of register B to the high-order 4 bits of timer 1 and timer 1 reload register R1L.<br>Transfers the contents of register A to the low-order 4 bits of timer 1 and timer 1 reload register R1L. |
| – | – | Transfers the contents of register B to the high-order 4 bits of timer 1 reload register R1H. Transfers the contents of register A to the low-order 4 bits of timer 1 reload register R1H. |

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Timer operation | TAB2 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2 7 1 | 1 | 1 | (B) ← ($T2_7$–$T2_4$)<br>(A) ← ($T2_3$–$T2_0$) |
| | T2AB | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 2 3 1 | 1 | 1 | ($R2L_7$–$R2L_4$) ← (B)<br>($T2_7$–$T2_4$) ← (B)<br>($R2L_3$–$R2L_0$) ← (A)<br>($T2_3$–$T2_0$) ← (A) |
| | T2HAB | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 9 4 | 1 | 1 | ($R2H_7$–$R2H_4$) ← (B)<br>($R2H_3$–$R2H_0$) ← (A) |
| | T1R1L | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 2 A 7 | 1 | 1 | (T1) ← (R1L) |
| | T2R2L | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 2 9 5 | 1 | 1 | (T2) ← (R2L) |
| | SNZT1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 8 0 | 1 | 1 | $V1_2$ = 0: (T1F) = 1 ?<br>(T1F) ← 0<br>$V1_2$ = 1: SNZT1 = NOP |
| | SNZT2 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 8 1 | 1 | 1 | $V1_3$ = 0: (T2F) = 1 ?<br>(T2F) ← 0<br>$V1_3$ = 1: SNZT2 = NOP |
| Input/Output operation | IAP0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 2 6 0 | 1 | 1 | (A) ← (P0) |
| | OP0A | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 2 0 | 1 | 1 | (P0) ← (A) |
| | IAP1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 2 6 1 | 1 | 1 | (A) ← (P1) |
| | OP1A | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 2 1 | 1 | 1 | (P1) ← (A) |
| | 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 |
| | OP2A | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 2 2 2 | 1 | 1 | ($P2_1$, $P2_0$) ← ($A_1$, $A_0$) |
| | CLD | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 1 1 | 1 | 1 | (D) ← 1 |
| | RD | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 1 4 | 1 | 1 | (D(Y)) ← 0<br>(Y) = 0 to 3 |
| | SD | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 1 5 | 1 | 1 | (D(Y)) ← 1<br>(Y) = 0 to 3 |
| | SZD | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 2 4 | 2 | 2 | (D(Y)) = 0 ?<br>(Y) = 0 to 3 |
| | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 2 B | | | |
| | TFR0A | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 2 2 8 | 1 | 1 | (FR0) ← (A) |
| | TFR1A | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2 2 9 | 1 | 1 | (FR1) ← (A) |
| | TFR2A | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 2 2 A | 1 | 1 | (FR2) ← (A) |
| | TFR3A | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 2 2 B | 1 | 1 | (FR3) ← (A) |
| | TK0A | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 2 1 B | 1 | 1 | (K0) ← (A) |
| | TAK0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 2 5 6 | 1 | 1 | (A) ← (K0) |

| Skip condition | Carry flag CY | Datailed description |
|---|---|---|
| – | – | Transfers the high-order 4 bits ($T2_7$–$T2_4$) of timer 2 to register B.<br>Transfers the low-order 4 bits ($T2_3$–$T2_0$) of timer 2 to register A. |
| – | – | Transfers the contents of register B to the high-order 4 bits of timer 2 and timer 2 reload register R2L. Transfers the contents of register A to the low-order 4 bits of timer 2 and timer 2 reload register R2L. |
| – | – | Transfers the contents of register B to the high-order 4 bits of timer 2 reload register R2H. Transfers the contents of register A to the low-order 4 bits of timer 2 reload register R2H. |
| – | – | Transfers the contents of timer 1 reload register R1L to timer 1. |
| – | – | Transfers the contents of timer 2 reload register R2L to timer 2. |
| $V1_2$ = 0: (T1F) = 1 | – | When $V1_2$ = 0 : Clears (0) to the T1F flag and skips the next instruction when timer 1 interrupt request flag T1F is “1.” When the T1F flag is “0,” executes the next instruction.<br>When $V1_2$ = 1 : This instruction is equivalent to the NOP instruction. ($V1_2$: bit 2 of interrupt control register V1) |
| $V1_3$ = 0: (T2F) =1 | – | When $V1_3$ = 0 : Clears (0) to the T2F flag and skips the next instruction when timer 2 interrupt request flag T2F is “1.” When the T2F flag is “0,” executes the next instruction.<br>When $V1_3$ = 1 : This instruction is equivalent to the NOP instruction. ($V1_3$: bit 3 of interrupt control register V1) |
| – | – | Transfers the input of port P0 to register A. |
| – | – | Outputs the contents of register A to port P0. |
| – | – | Transfers the input of port P1 to register A. |
| – | – | Outputs the contents of register A to port P1. |
| – | – | Transfers the input of port P2 to the low-order 2 bits ($A_1$, $A_0$) of register A.<br>“0” is stored to the bit 3 ($A_3$) of register A. |
| – | – | Outputs the contents of the low-order 2 bits ($A_1$, $A_0$) of register A to port P2. |
| – | – | Sets (1) to port D. |
| – | – | Clears (0) to a bit of port D specified by register Y. |
| – | – | Sets (1) to a bit of port D specified by register Y. |
| (D(Y)) = 0 ? | – | Skips the next instruction when a bit of port D specified by register Y is “0.” Executes the next instruction when a bit of port D specified by register Y is “1.” |
| – | – | Transfers the contents of register A to port output structure control register FR0. |
| – | – | Transfers the contents of register A to port output structure control register FR1. |
| – | – | Transfers the contents of register A to port output structure control register FR2. |
| – | – | Transfers the contents of register A to port output structure control register FR3. |
| – | – | Transfers the contents of register A to key-on wakeup control register K0. |
| – | – | Transfers the contents of key-on wakeup control register K0 to register A. |

## MACHINE INSTRUCTIONS (INDEX BY TYPES) (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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Input/Output operation | TK1A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 1 4 | 1 | 1 | (K1) ← (A) |
| | TAK1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 2 5 9 | 1 | 1 | (A) ← (K1) |
| | TK2A | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 2 1 5 | 1 | 1 | (K2) ← (A) |
| | TAK2 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 5 A | 1 | 1 | (A) ← (K2) |
| | TPU0A | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 2 2 D | 1 | 1 | (PU0) ← (A) |
| | TAPU0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 2 5 7 | 1 | 1 | (A) ← (PU0) |
| | TPU1A | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 2 2 E | 1 | 1 | (PU1) ← (A) |
| | TAPU1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 2 5 E | 1 | 1 | (A) ← (PU1) |
| | TPU2A | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 2 2 F | 1 | 1 | (PU2) ← (A) |
| | TAPU2 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 2 5 F | 1 | 1 | (A) ← (PU2) |
| | TL1A | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 2 0 A | 1 | 1 | (L1) ← (A) |
| | TAL1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 2 4 A | 1 | 1 | (A) ← (L1) |
| Serial interface operation | TABSI | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 2 7 8 | 1 | 1 | (B) ← ($SI_7$–$SI_4$) (A) ← ($SI_3$–$SI_0$) |
| | TSIAB | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 2 3 8 | 1 | 1 | ($SI_7$–$SI_4$) ← (B) ($SI_3$–$SI_0$) ← (A) |
| | SST | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 2 9 E | 1 | 1 | (SIOF) ← 0<br>Serial interface transmit/receive starting |
| | SNZSI | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 8 8 | 1 | 1 | $V2_3$=0: (SIOF)=1?<br>(SIOF) ← 0<br>$V2_3$ = 1: SNZSI = NOP |
| | TAJ1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 4 2 | 1 | 1 | (A) ← (J1) |
| | TJ1A | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 0 2 | 1 | 1 | (J1) ← (A) |
| Clock operation | CRCK | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 2 9 B | 1 | 1 | RC oscillator selected |
| | TRGA | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 0 9 | 1 | 1 | ($RG_0$) ← ($A_0$) |
| | TAMR | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 2 5 2 | 1 | 1 | (A) ← (MR) |
| | TMRA | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 2 1 6 | 1 | 1 | (MR) ← (A) |

| Skip condition | Carry flag CY | Datailed description |
|---|---|---|
| – | – | Transfers the contents of register A to key-on wakeup control register K1. |
| – | – | Transfers the contents of key-on wakeup control register K1 to register A. |
| – | – | Transfers the contents of register A to key-on wakeup control register K2. |
| – | – | Transfers the contents of key-on wakeup control register K2 to register A. |
| – | – | Transfers the contents of register A to pull-up control register PU0. |
| – | – | Transfers the contents of pull-up control register PU0 to register A. |
| – | – | Transfers the contents of register A to pull-up control register PU1. |
| – | – | Transfers the contents of pull-up control register PU1 to register A. |
| – | – | Transfers the contents of register A to pull-up control register PU2. |
| – | – | Transfers the contents of pull-up control register PU2 to register A. |
| – | – | Transfers the contents of register A to key-on wakeup control register L1. |
| – | – | Transfers the contents of key-on wakeup control register L1 to register A. |
| – | – | Transfers the high-order 4 bits of serial interface register SI to register B, and transfers the low-order 4 bits of serial interface register SI to register A. |
| – | – | Transfers the contents of register B to the high-order 4 bits of serial interface register SI, and transfers the contents of register A to the low-order 4 bits of serial interface register SI. |
| – | – | Clears (0) to SIOF flag and starts serial interface transmit/receive. |
| $V2_3$ = 0: (SIOF) =1 | – | Clears (0) to SIOF flag and skips the next instruction when the contents of bit 3 ($V2_3$) of interrupt control register V2 is “0” and contents of SIOF flag is “1.” When $V2_3$ = 1: This instruction is equivalent to the NOP instruction. |
| – | – | Transfers the contents of serial interface control register J1 to register A. |
| – | – | Transfers the contents of register A to serial interface control register J1. |
| – | – | Selects the RC oscillation circuit for main clock f($X_{IN}$). |
| – | – | Transfers the least significant bit ($A_0$) of register A to clock control regiser RG. |
| – | – | Transfers the contents of clock control regiser MR to register A. |
| – | – | Transfers the contents of register A to clock control register MR. |

## MACHINE INSTRUCTIONS (INDEX BY TYPES) (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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A/D conversion operation | TABAD | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 2 7 9 | 1 | 1 | Q13 = 0:<br>(B) ← (AD9–AD6)<br>(A) ← (AD5–AD2)<br>Q13 = 1:<br>(B) ← (AD7–AD4)<br>(A) ← (AD3–AD0) |
| | TALA | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 2 4 9 | 1 | 1 | (A3, A2) ← (AD1, AD0)<br>(A1, A0) ← 0 |
| | TADAB | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 2 3 9 | 1 | 1 | Q13 = 0:<br>(AD7–AD4) ← (B)<br>(AD3–AD0) ← (A)<br>Q13 = 1: TADAB = NOP |
| | TAQ1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 2 4 4 | 1 | 1 | (A) ← (Q1) |
| | TQ1A | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 0 4 | 1 | 1 | (Q1) ← (A) |
| | ADST | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 9 F | 1 | 1 | (ADF) ← 0<br>Q13 = 0: A/D conversion starting<br>Q13 = 1: Comparator operation starting |
| | SNZAD | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 2 8 7 | 1 | 1 | V22 = 0: (ADF) = 1 ?<br>(ADF) ← 0<br>V22 = 1: SNZAD = NOP |
| Other operation | NOP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 0 0 | 1 | 1 | (PC) ← (PC) + 1 |
| | POF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 0 2 | 1 | 1 | RAM back-up |
| | EPOF | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 5 B | 1 | 1 | POF instruction valid |
| | SNZP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 0 3 | 1 | 1 | (P) = 1 ? |
| | DWDT | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 2 9 C | 1 | 1 | Stop of watchdog timer function enabled |
| | WRST | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 A 0 | 1 | 1 | (WDF1) = 1 ?,<br>(WDF1) ← 0 |
| | SRST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 0 1 | 1 | 1 | System reset |
| | RUPT | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 5 8 | 1 | 1 | (UPTF) ← 0 |
| | SUPT | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 5 9 | 1 | 1 | (UPTF) ← 1 |
| | SVDE** | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 2 9 3 | 1 | 1 | Voltage drop detection circuit valid at RAM back-up |

Note: The SVDE instruction can be used only in the H version.

| Skip condition | Carry flag CY | Datailed description |
|---|---|---|
| – | – | In the A/D conversion mode ($Q1_3$ = 0), transfers the high-order 4 bits ($AD_9$–$AD_6$) of register AD to register B, and the middle-order 4 bits ($AD_5$–$AD_2$) of register AD to register A.<br>In the comparator mode ($Q1_3$ = 1), transfers the high-order 4 bits ($AD_7$–$AD_4$) of comparator register to register B, and the low-order 4 bits ($AD_3$–$AD_0$) of comparator register to register A.<br>($Q1_3$: bit 3 of A/D control register Q1) |
| – | – | Transfers the low-order 2 bits ($AD_1$, $AD_0$) of register AD to the high-order 2 bits ($A_3$, $A_2$) of register A.<br>"0" is stored to the least significant bit ($A_0$) of register A. |
| – | – | In the comparator mode ($Q1_3$ = 1), transfers the contents of register B to the high-order 4 bits ($AD_7$–$AD_4$) of comparator register, and the contents of register A to the low-order 4 bits ($AD_3$–$AD_0$) of comparator register.<br>In the A/D conversion mode ($Q1_3$ = 0), this instruction is equivalent to the NOP instruction.<br>($Q1_3$ = bit 3 of A/D control register Q1) |
| – | – | Transfers the contents of A/D control register Q1 to register A. |
| – | – | Transfers the contents of register A to A/D control register Q1. |
| – | – | Clears (0) to A/D conversion completion flag ADF, and the A/D conversion at the A/D conversion mode ($Q1_3$ = 0) or the comparator operation at the comparator mode ($Q1_3$ = 1) is started.<br>($Q1_3$ = bit 3 of A/D control register Q1) |
| $V2_2$ = 0: (ADF) = 1 | – | When $V2_2$ = 0 : Skips the next instruction when A/D conversion completion flag ADF is "1." After skipping, clears (0) to the ADF flag. When the ADF flag is "0," executes the next instruction.<br>When $V2_2$ = 1 : This instruction is equivalent to the NOP instruction. ($V2_2$: bit 2 of interrupt control register V2) |
| – | – | No operation; Adds 1 to program counter value, and others remain unchanged. |
| – | – | Puts the system in RAM back-up state by executing the POF instruction after executing the EPOF instruction.<br>Operations of all functions are stopped. |
| – | – | Makes the immediate after POF instruction valid by executing the EPOF instruction. |
| (P) = 1 | – | Skips the next instruction when the P flag is "1".<br>After skipping, the P flag remains unchanged.<br>Executes the next instruction when the P flag is "0." |
| – | – | Stops the watchdog timer function by the WRST instruction after executing the DWDT instruction. |
| (WDF1) = 1 | – | Clears (0) to the WDF1 flag and skips the next instruction when watchdog timer flag WDF1 is "1." When the WDF1 flag is "0," executes the next instruction. Also, stops the watchdog timer function when executing the WRST instruction immediately after the DWDT instruction. |
| – | – | System reset occurs. |
| – | – | Clears (0) to the high-order bit reference enable flag UPTF. |
| – | – | Sets (1) to the high-order bit reference enable flag UPTF. |
| – | – | Validates the voltage drop detection circuit at RAM back-up (only for the H version). |

## INSTRUCTION CODE TABLE

| D9–D4 / D3–D0 | Hex. notation | 000000 | 000001 | 000010 | 000011 | 000100 | 000101 | 000110 | 000111 | 001000 | 001001 | 001010 | 001011 | 001100 | 001101 | 001110 | 001111 | 010000 010111 | 011000 011111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D3–D0 | Hex. notation | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10–17 | 18–1F |
| 0000 | 0 | NOP | BLA | SZB 0 | BMLA | – | TASP | A 0 | LA 0 | TABP 0 | TABP 16 | – | – | BML | BML | BL | BL | BM | B |
| 0001 | 1 | SRST | CLD | SZB 1 | – | – | TAD | A 1 | LA 1 | TABP 1 | TABP 17 | – | – | BML | BML | BL | BL | BM | B |
| 0010 | 2 | POF | – | SZB 2 | – | – | TAX | A 2 | LA 2 | TABP 2 | TABP 18 | – | – | BML | BML | BL | BL | BM | B |
| 0011 | 3 | SNZP | INY | SZB 3 | – | – | TAZ | A 3 | LA 3 | TABP 3 | TABP 19 | – | – | BML | BML | BL | BL | BM | B |
| 0100 | 4 | DI | RD | SZD | – | RT | TAV1 | A 4 | LA 4 | TABP 4 | TABP 20 | – | – | BML | BML | BL | BL | BM | B |
| 0101 | 5 | EI | SD | SEAn | – | RTS | TAV2 | A 5 | LA 5 | TABP 5 | TABP 21 | – | – | BML | BML | BL | BL | BM | B |
| 0110 | 6 | RC | – | SEAM | – | RTI | – | A 6 | LA 6 | TABP 6 | TABP 22 | – | – | BML | BML | BL | BL | BM | B |
| 0111 | 7 | SC | DEY | – | – | – | – | A 7 | LA 7 | TABP 7 | TABP 23 | – | – | BML | BML | BL | BL | BM | B |
| 1000 | 8 | – | AND | – | SNZ0 | LZ 0 | RUPT | A 8 | LA 8 | TABP 8 | TABP 24 | – | – | BML | BML | BL | BL | BM | B |
| 1001 | 9 | – | OR | TDA | – | LZ 1 | SUPT | A 9 | LA 9 | TABP 9 | TABP 25 | – | – | BML | BML | BL | BL | BM | B |
| 1010 | A | AM | TEAB | TABE | SNZI0 | LZ 2 | – | A 10 | LA 10 | TABP 10 | TABP 26 | – | – | BML | BML | BL | BL | BM | B |
| 1011 | B | AMC | – | – | – | LZ 3 | EPOF | A 11 | LA 11 | TABP 11 | TABP 27 | – | – | BML | BML | BL | BL | BM | B |
| 1100 | C | TYA | CMA | – | – | RB 0 | SB 0 | A 12 | LA 12 | TABP 12 | TABP 28 | – | – | BML | BML | BL | BL | BM | B |
| 1101 | D | – | RAR | – | – | RB 1 | SB 1 | A 13 | LA 13 | TABP 13 | TABP 29 | – | – | BML | BML | BL | BL | BM | B |
| 1110 | E | TBA | TAB | – | TV2A | RB 2 | SB 2 | A 14 | LA 14 | TABP 14 | TABP 30 | – | – | BML | BML | BL | BL | BM | B |
| 1111 | F | – | TAY | SZC | TV1A | RB 3 | SB 3 | A 15 | LA 15 | TABP 15 | TABP 31 | – | – | 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 "–."

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

| | The second word | | |
|---|---|---|---|
| BL | 10 | 0aaa | aaaa |
| BML | 10 | 0aaa | aaaa |
| BLA | 10 | 0p00 | pppp |
| BMLA | 10 | 0p00 | pppp |
| SEA | 00 | 0111 | nnnn |
| SZD | 00 | 0010 | 1011 |

## INSTRUCTION CODE TABLE (continued)

| $D_3$–$D_0$ \ $D_9$–$D_4$ | 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 | – | – | OP0A | T1AB | – | TAW6 | 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 | TJ1A | TW5A | OP2A | – | TAJ1 | TAMR | IAP2 | – | – | T1HAB | – | TMA 2 | TAM 2 | XAM 2 | XAMI 2 | XAMD 2 | LXY |
| 0011 | 3 | – | TW6A | – | – | – | TAI1 | – | – | – | SVDE* | – | TMA 3 | TAM 3 | XAM 3 | XAMI 3 | XAMD 3 | LXY |
| 0100 | 4 | TQ1A | TK1A | – | – | TAQ1 | – | – | – | – | T2HAB | – | TMA 4 | TAM 4 | XAM 4 | XAMI 4 | XAMD 4 | LXY |
| 0101 | 5 | – | TK2A | – | TPSAB | – | – | – | TABPS | – | T2R2L | – | 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 | – | – | SNZAD | – | T1R1L | TMA 7 | TAM 7 | XAM 7 | XAMI 7 | XAMD 7 | LXY |
| 1000 | 8 | – | – | TFR0A | TSIAB | – | – | – | TABSI | SNZSI | – | – | TMA 8 | TAM 8 | XAM 8 | XAMI 8 | XAMD 8 | LXY |
| 1001 | 9 | TRGA | – | TFR1A | TADAB | TALA | TAK1 | – | TABAD | – | – | – | TMA 9 | TAM 9 | XAM 9 | XAMI 9 | XAMD 9 | LXY |
| 1010 | A | TL1A | – | TFR2A | – | TAL1 | TAK2 | – | – | – | – | TPAA | TMA 10 | TAM 10 | XAM 10 | XAMI 10 | XAMD 10 | LXY |
| 1011 | B | – | TK0A | TFR3A | – | TAW1 | – | – | – | – | CRCK | – | TMA 11 | TAM 11 | XAM 11 | XAMI 11 | XAMD 11 | LXY |
| 1100 | C | – | – | – | – | TAW2 | – | – | – | – | DWDT | – | TMA 12 | TAM 12 | XAM 12 | XAMI 12 | XAMD 12 | LXY |
| 1101 | D | – | – | TPU0A | – | – | – | – | – | – | – | – | TMA 13 | TAM 13 | XAM 13 | XAMI 13 | XAMD 13 | LXY |
| 1110 | E | TW1A | – | TPU1A | – | – | TAPU1 | – | – | – | SST | – | TMA 14 | TAM 14 | XAM 14 | XAMI 14 | XAMD 14 | LXY |
| 1111 | F | TW2A | – | TPU2A | – | TAW5 | TAPU2 | – | – | – | ADST | – | 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 | 10 0aaa aaaa |
| BML | 10 0aaa aaaa |
| BLA | 10 0p00 pppp |
| BMLA | 10 0p00 pppp |
| SEA | 00 0111 nnnn |
| SZD | 00 0010 1011 |

• * can be used only in the H version.

# Electrical characteristics

## Absolute maximum ratings

| Symbol | Parameter | Conditions | Ratings | Unit |
|---|---|---|---|---|
| $V_{DD}$ | Supply voltage | – | –0.3 to 6.5 | V |
| $V_I$ | Input voltage P0, P1, P2, $D_0$–$D_3$, $\overline{RESET}$, $X_{IN}$ | – | –0.3 to $V_{DD}$+0.3 | V |
| $V_I$ | Input voltage INT, CNTR0, CNTR1, $S_{IN}$, $S_{CK}$ | – | –0.3 to $V_{DD}$+0.3 | V |
| $V_I$ | Input voltage $A_{IN0}$, $A_{IN1}$, $A_{IN4}$, $A_{IN5}$ | – | –0.3 to $V_{DD}$+0.3 | V |
| $V_O$ | Output voltage P0, P1, P2, $D_0$–$D_3$, $\overline{RESET}$ | Output transistors in cut-off state | –0.3 to $V_{DD}$+0.3 | V |
| $V_O$ | Output voltage CNTR0, CNTR1, $S_{OUT}$, $S_{CK}$ | Output transistors in cut-off state | –0.3 to $V_{DD}$+0.3 | V |
| $V_O$ | Output voltage $X_{OUT}$ | – | –0.3 to $V_{DD}$+0.3 | V |
| $P_d$ | Power dissipation | Ta = 25 °C | 300 | mW |
| Topr | Operating temperature range | – | –20 to 85 | °C |
| Tstg | Storage temperature range | – | –40 to 125 | °C |

## Recommended operating conditions 1

(Ta = –20 °C to 85 °C, $V_{DD}$ = 1.8 to 5.5 V, unless otherwise noted)

| Symbol | Parameter | Conditions | | Limits | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | Min. | Typ. | Max. | |
| $V_{DD}$ | Supply voltage (with a ceramic resonator) | f(STCK) ≤ 6 MHz | | 4 | | 5.5 | V |
| | | f(STCK) ≤ 4.4 MHz | | 2.7 | | 5.5 | |
| | | f(STCK) ≤ 2.2 MHz | | 2.0 | | 5.5 | |
| | | f(STCK) ≤ 1.1 MHz | | 1.8 | | 5.5 | |
| $V_{DD}$ | Supply voltage (with RC oscillation) | f(STCK) ≤ 4.4 MHz | | 2.7 | | 5.5 | V |
| $V_{DD}$ | Supply voltage (with an on-chip oscillator) | | | 1.8 | | 5.5 | V |
| $V_{RAM}$ | RAM back-up voltage | (at RAM back-up) | | 1.6 | | 5.5 | V |
| $V_{SS}$ | Supply voltage | | | | 0 | | V |
| $V_{IH}$ | "H" level input voltage | P0, P1, P2, $D_0$–$D_3$ | | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| | | $X_{IN}$ | | 0.7$V_{DD}$ | | $V_{DD}$ | |
| | | $\overline{RESET}$ | | 0.85$V_{DD}$ | | $V_{DD}$ | |
| | | INT, CNTR0, CNTR1, $S_{IN}$, $S_{CK}$ | | 0.85$V_{DD}$ | | $V_{DD}$ | |
| $V_{IL}$ | "L" level input voltage | P0, P1, P2, $D_0$–$D_3$ | | 0 | | 0.2$V_{DD}$ | V |
| | | $X_{IN}$ | | 0 | | 0.3$V_{DD}$ | |
| | | $\overline{RESET}$ | | 0 | | 0.3$V_{DD}$ | |
| | | INT, CNTR0, CNTR1, $S_{IN}$, $S_{CK}$ | | 0 | | 0.15$V_{DD}$ | |
| $I_{OH}$(peak) | "H" level peak output current | P0, P1, P2, $D_0$–$D_3$ | $V_{DD}$ = 5.0 V | | | –20 | mA |
| | | CNTR0, CNTR1, $S_{OUT}$, $S_{CK}$ | $V_{DD}$ = 3.0 V | | | –10 | |
| $I_{OH}$(avg) | "H" level average output current (Note) | P0, P1, P2, $D_0$–$D_3$ | $V_{DD}$ = 5.0 V | | | –10 | mA |
| | | CNTR0, CNTR1, $S_{OUT}$, $S_{CK}$ | $V_{DD}$ = 3.0 V | | | –5 | |
| $I_{OL}$(peak) | "L" level peak output current | P0, P1 | $V_{DD}$ = 5.0 V | | | 24 | mA |
| | | CNTR0, CNTR1, $S_{OUT}$, $S_{CK}$ | $V_{DD}$ = 3.0 V | | | 12 | |
| | | P2, $\overline{RESET}$ | $V_{DD}$ = 5.0 V | | | 10 | |
| | | | $V_{DD}$ = 3.0 V | | | 4.0 | |
| | | $D_0$, $D_1$ | $V_{DD}$ = 5.0 V | | | 40 | |
| | | | $V_{DD}$ = 3.0 V | | | 30 | |
| | | $D_2$, $D_3$ | $V_{DD}$ = 5.0 V | | | 24 | |
| | | | $V_{DD}$ = 3.0 V | | | 12 | |
| $I_{OL}$(avg) | "L" level average output current | P0, P1 | $V_{DD}$ = 5.0 V | | | 12 | mA |
| | | CNTR0, CNTR1, $S_{OUT}$, $S_{CK}$ | $V_{DD}$ = 3.0 V | | | 6.0 | |
| | | P2, $\overline{RESET}$ | $V_{DD}$ = 5.0 V | | | 5.0 | |
| | | | $V_{DD}$ = 3.0 V | | | 2.0 | |
| | | $D_0$, $D_1$ | $V_{DD}$ = 5.0 V | | | 30 | |
| | | | $V_{DD}$ = 3.0 V | | | 15 | |
| | | $D_2$, $D_3$ | $V_{DD}$ = 5.0 V | | | 15 | |
| | | | $V_{DD}$ = 3.0 V | | | 7.0 | |
| $\Sigma I_{OH}$(avg) | "H" level total average current | P0, P1, CNTR0, CNTR1, $S_{OUT}$, $S_{CK}$ | | | | –40 | mA |
| | | P2, $D_0$–$D_3$ | | | | –40 | |
| $\Sigma I_{OL}$(avg) | "L" level total average current | P0, P1, CNTR0, CNTR1, $S_{OUT}$, $S_{CK}$ | | | | 60 | mA |
| | | P2, $D_0$–$D_3$, $\overline{RESET}$ | | | | 60 | |

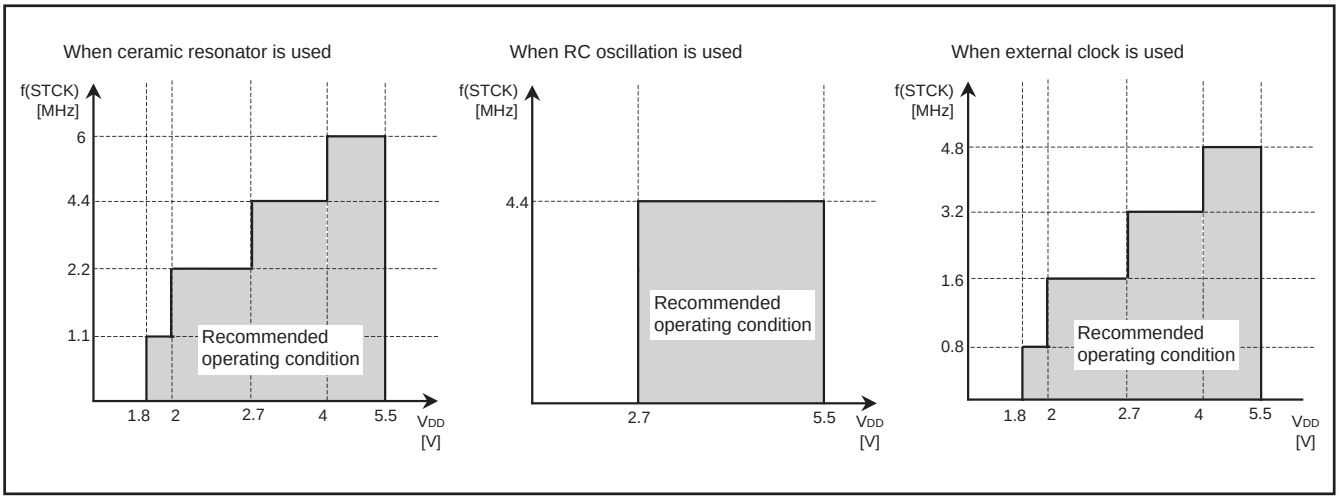
Notes 1: The average output current ($I_{OH}$, $I_{OL}$) is the average value during 100 ms.

## Recommended operating conditions 2

(Ta = –20 °C to 85 °C, $V_{DD}$ = 1.8 to 5.5 V, unless otherwise noted)

| Symbol | Parameter | Conditions | | Limits | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | Min. | Typ. | Max. | |
| f(XIN) | Oscillation frequency (with a ceramic resonator) | Through mode | $V_{DD}$ = 4.0 V to 5.5 V | | | 6 | MHz |
| | | | $V_{DD}$ = 2.7 V to 5.5 V | | | 4.4 | |
| | | | $V_{DD}$ = 2.0 V to 5.5 V | | | 2.2 | |
| | | | $V_{DD}$ = 1.8 V to 5.5 V | | | 1.1 | |
| | | Internal frequency divided by 2 | $V_{DD}$ = 2.7 V to 5.5 V | | | 6 | |
| | | | $V_{DD}$ = 2.0 V to 5.5 V | | | 4.4 | |
| | | | $V_{DD}$ = 1.8 V to 5.5 V | | | 2.2 | |
| | | Internal frequency divided by 4, 8 | $V_{DD}$ = 2.0 V to 5.5 V | | | 6 | |
| | | | $V_{DD}$ = 1.8 V to 5.5 V | | | 4.4 | |
| f(XIN) | Oscillation frequency (with RC oscillation) (Note 1) | $V_{DD}$ = 2.7 V to 5.5 V | | | | 4.4 | MHz |
| f(XIN) | Oscillation frequency (with a ceramic oscillation selected, external clock input) | Through mode | $V_{DD}$ = 4.0 V to 5.5 V | | | 4.8 | MHz |
| | | | $V_{DD}$ = 2.7 V to 5.5 V | | | 3.2 | |
| | | | $V_{DD}$ = 2.0 V to 5.5 V | | | 1.6 | |
| | | | $V_{DD}$ = 1.8 V to 5.5 V | | | 0.8 | |
| | | Internal frequency divided by 2 | $V_{DD}$ = 2.7 V to 5.5 V | | | 4.8 | |
| | | | $V_{DD}$ = 2.0 V to 5.5 V | | | 3.2 | |
| | | | $V_{DD}$ = 1.8 V to 5.5 V | | | 1.6 | |
| | | Internal frequency divided by 4, 8 | $V_{DD}$ = 2.0 V to 5.5 V | | | 4.8 | |
| | | | $V_{DD}$ = 1.8 V to 5.5 V | | | 3.2 | |
| f(CNTR) | Timer external input frequency | CNTR0, CNTR1 | | | | f(STCK)/6 | Hz |
| tw(CNTR) | Timer external input period (“H” and “L” pulse width) | CNTR0, CNTR1 | | 3/f(STCK) | | | s |
| f(SCK) | Serial interface external input frequency | SCK | | 3/f(STCK) | | f(STCK)/6 | Hz |
| tw(SCK) | Serial interface external input period (“H” and “L” pulse width) | SCK | | | | | s |
| TPON | Power-on reset circuit valid supply voltage rising time (Note 2) | $V_{DD}$ = 0 → 1.8 V | | | | 100 | µs |

Notes 1: The frequency at RC oscillation is affected by a capacitor, a resistor and a microcomputer. So, set the constants within the range of the frequency limits.
2: If the rising time exceeds the maximum rating value, 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.




**System clock (STCK) operating condition map**

**Electrical characteristics 1** (Ta = –20 °C to 85 °C, $V_{DD}$ = 1.8 to 5.5 V, unless otherwise noted)

| Symbol | Parameter | Test conditions | | Limits | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | Min. | Typ. | Max. | |
| $V_{OH}$ | "H" level output voltage<br>P0, P1, P2, $D_0$–$D_3$<br>CNTR0, CNTR1, SOUT, SCK | $V_{DD}$ = 5.0 V | $I_{OH}$ = –10 mA | 3.0 | | | V |
| | | | $I_{OH}$ = –3.0 mA | 4.1 | | | |
| | | $V_{DD}$ = 3.0 V | $I_{OH}$ = –5.0 mA | 2.1 | | | |
| | | | $I_{OH}$ = –1.0 mA | 2.4 | | | |
| $V_{OL}$ | "L" level output voltage<br>P0, P1<br>CNTR0, CNTR1, SOUT, SCK | $V_{DD}$ = 5.0 V | $I_{OL}$ = 12 mA | | | 2.0 | V |
| | | | $I_{OL}$ = 4.0 mA | | | 0.9 | |
| | | $V_{DD}$ = 3.0 V | $I_{OL}$ = 6.0 mA | | | 0.9 | |
| | | | $I_{OL}$ = 2.0 mA | | | 0.6 | |
| $V_{OL}$ | "L" level output voltage<br>P2, $\overline{\text{RESET}}$ | $V_{DD}$ = 5.0 V | $I_{OL}$ = 5.0 mA | | | 2.0 | V |
| | | | $I_{OL}$ = 1.0 mA | | | 0.6 | |
| | | $V_{DD}$ = 3.0 V | $I_{OL}$ = 2.0 mA | | | 0.9 | |
| $V_{OL}$ | "L" level output voltage<br>$D_0$, $D_1$ | $V_{DD}$ = 5.0 V | $I_{OL}$ = 30 mA | | | 2.0 | V |
| | | | $I_{OL}$ = 10 mA | | | 0.9 | |
| | | $V_{DD}$ = 3.0 V | $I_{OL}$ = 15 mA | | | 2.0 | |
| | | | $I_{OL}$ = 5.0 mA | | | 0.9 | |
| $V_{OL}$ | "L" level output voltage<br>$D_2$, $D_3$ | $V_{DD}$ = 5.0 V | $I_{OL}$ = 15 mA | | | 2.0 | V |
| | | | $I_{OL}$ = 5.0 mA | | | 0.9 | |
| | | $V_{DD}$ = 3.0 V | $I_{OL}$ = 9.0 mA | | | 1.4 | |
| | | | $I_{OL}$ = 3.0 mA | | | 0.9 | |
| $I_{IH}$ | "H" level input current<br>P0, P1, P2, $D_0$–$D_3$<br>$\overline{\text{RESET}}$, INT<br>CNTR0, CNTR1, SIN, SCK | $V_I$ = $V_{DD}$ | | | | 2.0 | μA |
| $I_{IL}$ | "L" level input current<br>P0, P1, P2, $D_0$–$D_3$<br>$\overline{\text{RESET}}$, INT<br>CNTR0, CNTR1, SIN, SCK | $V_I$ = 0 V P0, P1, P2, $D_2$, $D_3$ No pull-up | | | | –2.0 | μA |
| $R_{PU}$ | Pull-up resistor value<br>P0, P1, P2, $D_2$, $D_3$, $\overline{\text{RESET}}$ | $V_I$ = 0 V | $V_{DD}$ = 5.0 V | 30 | 60 | 125 | kΩ |
| | | | $V_{DD}$ = 3.0 V | 50 | 120 | 250 | |
| $V_{T+} – V_{T–}$ | Hysteresis $\overline{\text{RESET}}$ | $V_{DD}$ = 5.0 V | | | 1.0 | | V |
| | | $V_{DD}$ = 3.0 V | | | 0.4 | | |
| $V_{T+} – V_{T–}$ | Hysteresis INT, CNTR0, CNTR1<br>SIN, SCK | $V_{DD}$ = 5.0 V | | | 0.2 | | V |
| | | $V_{DD}$ = 3.0 V | | | 0.2 | | |
| f(RING) | On-chip oscillator clock frequency | $V_{DD}$ = 5.0 V | | 200 | 500 | 700 | kHz |
| | | $V_{DD}$ = 3.0 V | | 100 | 250 | 400 | |
| | | $V_{DD}$ = 1.8 V | | 30 | 120 | 200 | |
| Δ f($X_{IN}$) | Oscillation frequency error (Note 1)<br>(at RC oscillation, error value of external R, C not included) | $V_{DD}$ = 5.0 V ± 10 %, Ta = center 25 °C | | | | ±17 | % |
| | | $V_{DD}$ = 3.0 V ± 10 %, Ta = center 25 °C | | | | ±17 | |

Notes 1: When the RC oscillation is used, use a 33 pF capacitor externally.

**Electrical characteristics 2** (Ta = –20 °C to 85 °C, $V_{DD}$ = 1.8 to 5.5 V, unless otherwise noted)

| Symbol | Parameter | | Test conditions | | Limits | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | | | | Min. | Typ. | Max. | |
| $I_{DD}$ | Supply current | at active mode (with a ceramic resonator) (Notes 1, 2) | $V_{DD}$ = 5.0 V, f($X_{IN}$) = 6.0 MHz, f(RING) = stop | f(STCK) = f($X_{IN}$)/8 | | 1.2 | 2.4 | mA |
| | | | | f(STCK) = f($X_{IN}$)/4 | | 1.3 | 2.6 | |
| | | | | f(STCK) = f($X_{IN}$)/2 | | 1.6 | 3.2 | |
| | | | | f(STCK) = f($X_{IN}$) | | 2.2 | 4.4 | |
| | | | $V_{DD}$ = 5.0 V, f($X_{IN}$) = 4.0 MHz, f(RING) = stop | f(STCK) = f($X_{IN}$)/8 | | 0.9 | 1.8 | mA |
| | | | | f(STCK) = f($X_{IN}$)/4 | | 1 | 2 | |
| | | | | f(STCK) = f($X_{IN}$)/2 | | 1.2 | 2.4 | |
| | | | | f(STCK) = f($X_{IN}$) | | 1.6 | 3.2 | |
| | | | $V_{DD}$ = 3.0 V, f($X_{IN}$) = 2.0 MHz, f(RING) = stop | f(STCK) = f($X_{IN}$)/8 | | 0.2 | 0.4 | mA |
| | | | | f(STCK) = f($X_{IN}$)/4 | | 0.25 | 0.5 | |
| | | | | f(STCK) = f($X_{IN}$)/2 | | 0.3 | 0.6 | |
| | | | | f(STCK) = f($X_{IN}$) | | 0.4 | 0.8 | |
| | | at active mode (with an on-chip oscillator) (Notes 1, 2) | $V_{DD}$ = 5.0 V, f($X_{IN}$) = stop, f(RING) = operating | f(STCK) = f(RING)/8 | | 50 | 100 | μA |
| | | | | f(STCK) = f(RING)/4 | | 60 | 120 | |
| | | | | f(STCK) = f(RING)/2 | | 80 | 160 | |
| | | | | f(STCK) = f(RING) | | 120 | 240 | |
| | | | $V_{DD}$ = 3.0 V, f($X_{IN}$) = stop, f(RING) = opertaing | f(STCK) = f(RING)/8 | | 10 | 20 | μA |
| | | | | f(STCK) = f(RING)/4 | | 13 | 26 | |
| | | | | f(STCK) = f(RING)/2 | | 19 | 38 | |
| | | | | f(STCK) = f(RING) | | 31 | 62 | |
| | | at RAM back-up mode (POF instruction execution) (Note 3) | Ta = 25 °C | | | 0.1 | 3 | μA |
| | | | $V_{DD}$ = 5.0 V | | | | 10 | |
| | | | $V_{DD}$ = 3.0 V | | | | 6 | |

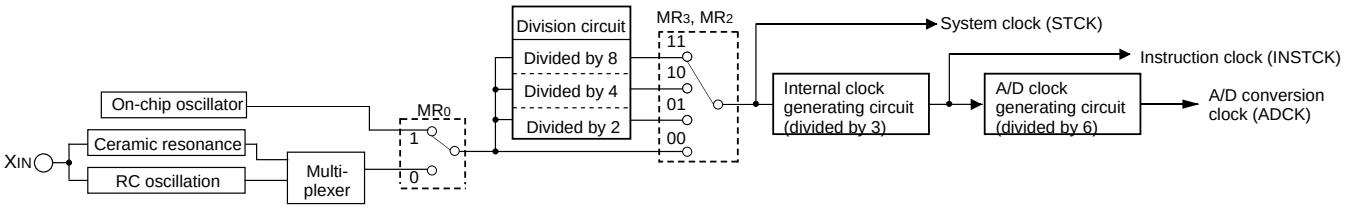
Notes 1: When the A/D converter is used, the A/D operation current ($I_{ADD}$) is added.
2: In the M34508G4H, the voltage drop detection circuit operation current ($I_{RST}$) is added.
3: In the M34508G4H, when the SVDE instruction is executed, the voltage drop detection circuit operation current ($I_{RST}$) is added.

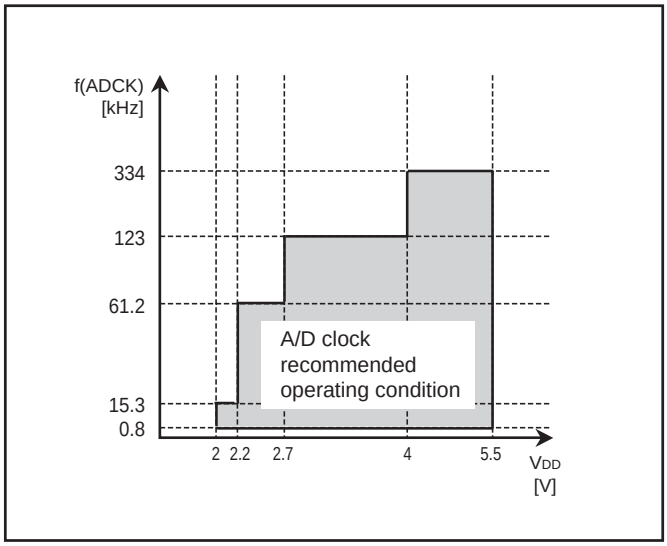
## A/D converter recommended operating conditions

(Comparator mode included, Ta = –20 °C to 85 °C, unless otherwise noted)

| Symbol | Parameter | Conditions | Limits | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $V_{DD}$ | Supply voltage | Ta = 0 °C to 50 °C | 2.0 | | 5.5 | V |
| | | Ta = –20 °C to 85 °C | 2.7 | | 5.5 | |
| $V_{IA}$ | Analog input voltage | | 0 | | $V_{DD}$ | V |
| f(ADCK) | A/D clock frequency (Note) | $V_{DD}$ = 4.0 V to 5.5 V | 0.8 | | 334 | kHz |
| | | $V_{DD}$ = 2.7 V to 5.5 V | 0.8 | | 123 | |
| | | $V_{DD}$ = 2.2 V to 5.5 V | 0.8 | | 61.2 | |
| | | $V_{DD}$ = 2.0 V to 5.5 V | 0.8 | | 15.3 | |

Note: Definition of A/D conversion clock (ADCK)




**A/D clock (ADCK) operating condition map**

## A/D converter characteristcs

(Ta = –20 °C to 85 °C, unless otherwise noted)

| Symbol | Parameter | Test conditions | Limits | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| – | Resolution | | | | 10 | bits |
| – | Linearity error | Ta = 0 °C to 50 °C, 2.2 V ≤ $V_{DD}$ < 2.7 V | | | ±4.0 | LSB |
| | | Ta = –20 °C to 85 °C, 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | | | ±2.0 | |
| – | Differential non-linearity error | Ta = 0 °C to 50 °C, 2.2 V ≤ $V_{DD}$ < 2.7 V | | | ±0.9 | LSB |
| | | Ta = –20 °C to 85 °C, 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | | | ±0.9 | |
| $V_{0T}$ | Zero transition voltage | $V_{DD}$ = 2.56 V | 0 | 7.5 | 15 | mV |
| | | $V_{DD}$ = 3.075 V | 0 | 7.5 | 15 | |
| | | $V_{DD}$ = 5.12 V | 0 | 10 | 20 | |
| $V_{FST}$ | Full-scale transition voltage | $V_{DD}$ = 2.56 V | 2552.5 | 2560 | 2567.5 | mV |
| | | $V_{DD}$ = 3.075 V | 3064.5 | 3072 | 3079.5 | |
| | | $V_{DD}$ = 5.12 V | 5100 | 5110 | 5120 | |
| – | Absolute accuracy (Quantization error excluded) | Ta = 0 °C to 50 °C, 2.0 V ≤ $V_{DD}$ < 2.2 V | | | ±8.0 | LSB |
| $I_{ADD}$ | A/D operating current (Note 1) | $V_{DD}$ = 5.0 V | | 300 | 900 | μA |
| | | $V_{DD}$ = 3.0 V | | 100 | 300 | |
| $T_{CONV}$ | A/D conversion time | f(ADCK) = 334 kHz | | | 31 | μs |
| | | f(ADCK) = 123 kHz | | | 85 | |
| | | f(ADCK) = 61.2 kHz | | | 169 | |
| | | f(ADCK) = 15.3 kHz | | | 676 | |
| – | Comparator resolution | | | | 8 | bits |
| – | Comparator error (Note 2) | $V_{DD}$ = 2.56 V | | | ± 15 | mV |
| | | $V_{DD}$ = 3.072 V | | | ± 15 | |
| | | $V_{DD}$ = 5.12 V | | | ± 20 | |
| – | Comparator comparison time | f(ADCK) = 334 kHz | | | 4 | μs |
| | | f(ADCK) = 123 kHz | | | 11 | |
| | | f(ADCK) = 61.2 kHz | | | 22 | |
| | | f(ADCK) = 15.3 kHz | | | 88 | |

Notes 1: When the A/D converter is used, the $I_{ADD}$ is added to $I_{DD}$.

2: As for the error from the logic value in the comparator mode, when the contents of the comparator register is n, the logic value of the comparison voltage $V_{ref}$ which is generated by the built-in DA converter can be obtained by the following formula.

Logic value of comparison voltage $V_{ref}$

$$V_{ref} = \frac{V_{DD}}{256} \times n$$

n = Value of register AD (n = 0 to 255)

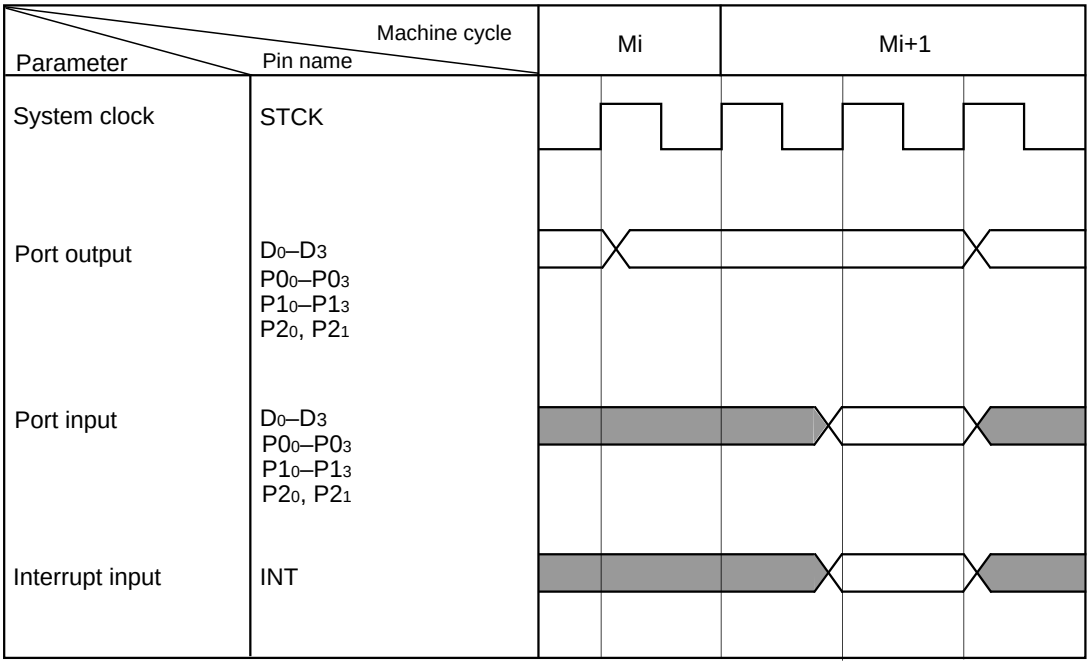
## VOLTAGE DROP DETECTION CIRCUIT CHARACTERISTICS

(Ta = –20 °C to 85 °C, unless otherwise noted)

| Symbol | Parameter | Test conditions | Limits | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $V_{RST^-}$ | Detection voltage (reset occurs) (Note 2) | Ta = 25 °C | | 2.6 | | V |
| | | -20 °C ≤ Ta < 0 °C | 2.5 | | 3.1 | |
| | | 0 °C ≤ Ta < 50 °C | 2.2 | | 3 | |
| | | 50 °C ≤ Ta ≤ 85 °C | 2 | | 2.7 | |
| $V_{RST^+}$ | Detection voltage (reset release) (Note 3) | Ta = 25 °C | | 2.7 | | V |
| | | -20 °C ≤ Ta < 0 °C | 2.6 | | 3.2 | |
| | | 0 °C ≤ Ta < 50 °C | 2.3 | | 3.1 | |
| | | 50 °C ≤ Ta ≤ 85 °C | 2.1 | | 2.8 | |
| $V_{RST^+} – V_{RST^-}$ | Detection voltage hysteresis | | | 0.1 | | V |
| $I_{RST}$ | Operation current (Note 4) | $V_{DD}$ = 5 V | | 50 | 100 | μA |
| | | $V_{DD}$ = 3 V | | 30 | 60 | |
| $T_{RST}$ | Detection time (Note 5) | $V_{DD} \rightarrow (V_{RST^-} – 0.1\ V)$ | | 0.2 | 1.2 | ms |

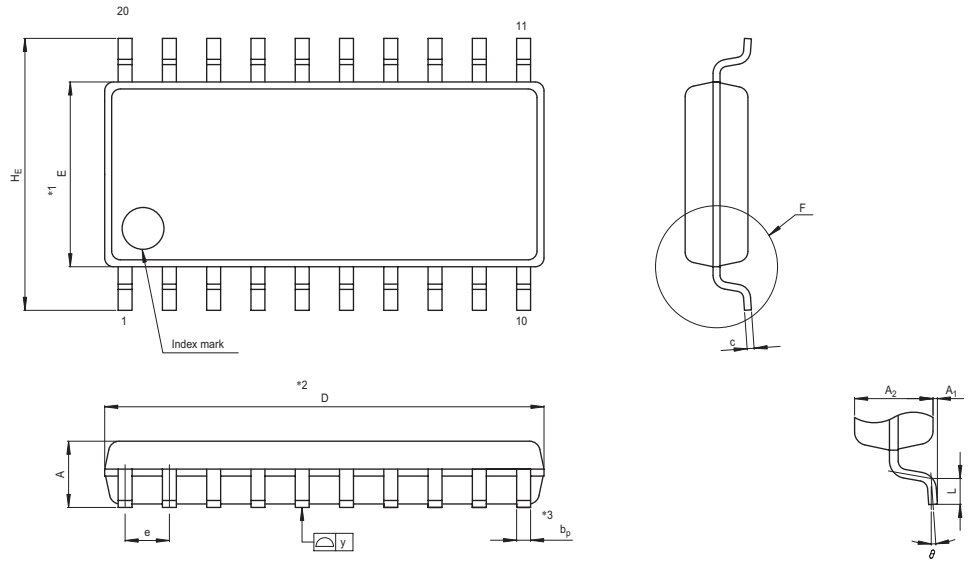
Notes 1: The voltage drop detection circuit is equipped with only the M34508G4H.
2: The detection voltage ($V_{RST^-}$) is defined as the voltage when reset occurs when the supply voltage ($V_{DD}$) is falling.
3: The detection voltage ($V_{RST^+}$) is defined as the voltage when reset is released when the supply voltage ($V_{DD}$) is rising from reset occurs.
4: In the M34508G4H, $I_{RST}$ is added to $I_{DD}$ (supply current).
5: The detection time ($T_{RST}$) is defined as the time until reset occurs when the supply voltage ($V_{DD}$) is falling to [$V_{RST^-}$ – 0.1 V].

### Basic timing diagram



# Package outline

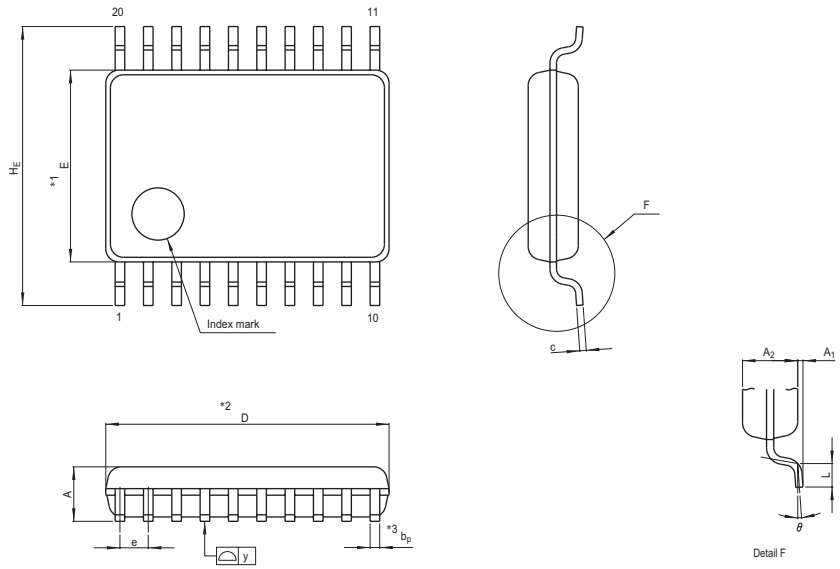
| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-SOP20-5.3x12.6-1.27 | PRSP0020DA-A | 20P2N-A | 0.3g |




NOTE)
1. DIMENSIONS "*1" AND "*2" DO NOT INCLUDE MOLD FLASH.
2. DIMENSION "*3" DOES NOT INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | 12.5 | 12.6 | 12.7 |
| E | 5.2 | 5.3 | 5.4 |
| $A_2$ | —— | 1.8 | —— |
| $A_1$ | 0 | 0.1 | 0.2 |
| A | —— | —— | 2.1 |
| $b_p$ | 0.35 | 0.4 | 0.5 |
| c | 0.18 | 0.2 | 0.25 |
| θ | 0° | —— | 8° |
| $H_E$ | 7.5 | 7.8 | 8.1 |
| e | 1.12 | 1.27 | 1.42 |
| y | —— | —— | 0.1 |
| L | 0.4 | 0.6 | 0.8 |

| JEITA Package Code | RENESAS Code | Previous Code | MASS[Typ.] |
|---|---|---|---|
| P-LSSOP20-4.4x6.5-0.65 | PLSP0020JB-A | 20P2F-A | 0.1g |




NOTE)
1. DIMENSIONS "*1" AND "*2" DO NOT INCLUDE MOLD FLASH.
2. DIMENSION "*3" DOES NOT INCLUDE TRIM OFFSET.

| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | 6.4 | 6.5 | 6.6 |
| E | 4.3 | 4.4 | 4.5 |
| $A_2$ | —— | 1.15 | —— |
| A | —— | —— | 1.45 |
| $A_1$ | 0 | 0.1 | 0.2 |
| $b_p$ | 0.17 | 0.22 | 0.32 |
| c | 0.13 | 0.15 | 0.2 |
| θ | 0° | —— | 10° |
| $H_E$ | 6.2 | 6.4 | 6.6 |
| e | 0.53 | 0.65 | 0.77 |
| y | —— | —— | 0.10 |
| L | 0.3 | 0.5 | 0.7 |

| REVISION HISTORY | 4508 Group Data Sheet |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Mar. 25, 2005 | – | First edition issued |
| 1.01 | Aug. 12, 2005 | 1 | Information about the PLSP0020JB-A package products. |
| | | | "PLSP0020JB-A (20P2F-A)" added to PIN CONFIGURATION. |
| | | 17 | ROM Code Protect Address added. |
| | | 52 | Table 20: Some description about Port P1 added. |
| | | 57 | Fig.52 revised. |
| | | 58 | Fig.54 revised. |
| | | | "DATA REQUIRED FOR QzROM WRITING ORDERS" added. |
| | | 62 | Notes On ROM Code Protect added. |
| | | 130 | A/D converter characteristics: |
| | | | Linearity error, Differential non-linearity error and Absolute accuracy |
| | | | → Parameters and Test conditions revised. |
| | | 131 | Voltage drop detection circuit characteristics: $V_{RST}^-$, $V_{RST}^+$ → Test conditions revised. |
| | | 132 | PLSP0020JB-A package added. |
| 1.02 | Dec. 22, 2006 | 2 | Block diagram: "Power-on reset circuit" added. |
| | | 5, 63 | Description (Note 4) about SCK pin revised. |
| | | 26 | TIMER: Description revised and Structure of Timer 2 in Table 9 revised. |
| | | 28 | Fig.23: INSTCK (wrong) → INTSNC (correct) |
| | | 30 | (2) Prescaler: PRS → RPS |
| | | | (3) Timer 3 → Timer 1 |
| | | 43 | SERIAL I/O: Table 14: Note revised. |
| | | 53 | Fig. 46: Notes revised. |
| | | 58 | Table 23: Changes referring ahead and note 5 added. |
| | | 59 to 61 | QzROM Writing Mode added. |
| | | 63 | LIST OF PRECAUTIONS: Mulfunction revised. |
| | | 68 to 71 | NOTES ON NOISE added. |
| | | 77 | Description of Port output structure control register FR2 and FR3 revised. |
| | | 103 | Instruction code of TAL1 revised. Description of TALA revised. |
| | | 107 | TC1A eliminated. |
| | | 117 | Detailed description of TEAB revised. |
| | | 134 | f(SCK): Serial interface external input frequency → |
| | | | Serial interface external input period |
| | | 135 | Δ f(XIN): Ta = around 25 °C → center 25 °C |
| | | 137 | Figure title revised, "When ceramic resonator is used" deleted. |
| | | 139 | Note 4: (power current) → (supply current) |
| | | → | Pages 80–82, 94, 95, 115, 122–129: |
| | | | Description of SNZ0, SNZT1, SNZT2, SNZAD, SNZSI and WRST instructions revised. |

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.03 | Jul. 27, 2009 | 1 | M34508G4GP, M34508G4-XXXGP, M34508G4HGP, M34508G4H-XXXGP<br>"Under development" deleted. |
| | | 50 | Fig 45: Note revised. |
| | | 134 | f(SCK): Serial interface external input period →<br>Serial interface external input frequency |
| | | 136 | Note 1: ...., the A/D operation current (IADD) is included. →<br>...., the A/D operation current (IADD) is added. |
| | | 138 | Note 1: ...., the IADD is included to IDD. → ...., the IADD is added to IDD. |

Renesas Technology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan



## RENESAS SALES OFFICES

http://www.renesas.com

Refer to "**http://www.renesas.com/en/network**" for the latest and detailed information.

**Renesas Technology America, Inc.**
450 Holger Way, San Jose, CA 95134-1368, U.S.A
Tel: <1> (408) 382-7500, Fax: <1> (408) 382-7501

**Renesas Technology Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

**Renesas Technology (Shanghai) Co., Ltd.**
Unit 204, 205, AZIACenter, No.1233 Lujiazui Ring Rd, Pudong District, Shanghai, China 200120
Tel: <86> (21) 5877-1818, Fax: <86> (21) 6887-7858/7898

**Renesas Technology Hong Kong Ltd.**
7th Floor, North Tower, World Finance Centre, Harbour City, Canton Road, Tsimshatsui, Kowloon, Hong Kong
Tel: <852> 2265-6688, Fax: <852> 2377-3473

**Renesas Technology Taiwan Co., Ltd.**
10th Floor, No.99, Fushing North Road, Taipei, Taiwan
Tel: <886> (2) 2715-2888, Fax: <886> (2) 3518-3399

**Renesas Technology Singapore Pte. Ltd.**
1 Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: <65> 6213-0200, Fax: <65> 6278-8001

**Renesas Technology Korea Co., Ltd.**
Kukje Center Bldg. 18th Fl., 191, 2-ka, Hangang-ro, Yongsan-ku, Seoul 140-702, Korea
Tel: <82> (2) 796-3115, Fax: <82> (2) 796-2145

**Renesas Technology Malaysia Sdn. Bhd**
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No.18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: <603> 7955-9390, Fax: <603> 7955-9510



Colophon .7.2