# Renesas Electronics America Inc - UPD78F0494GC-GAD-AX Datasheet

**Welcome to E-XFL.COM**

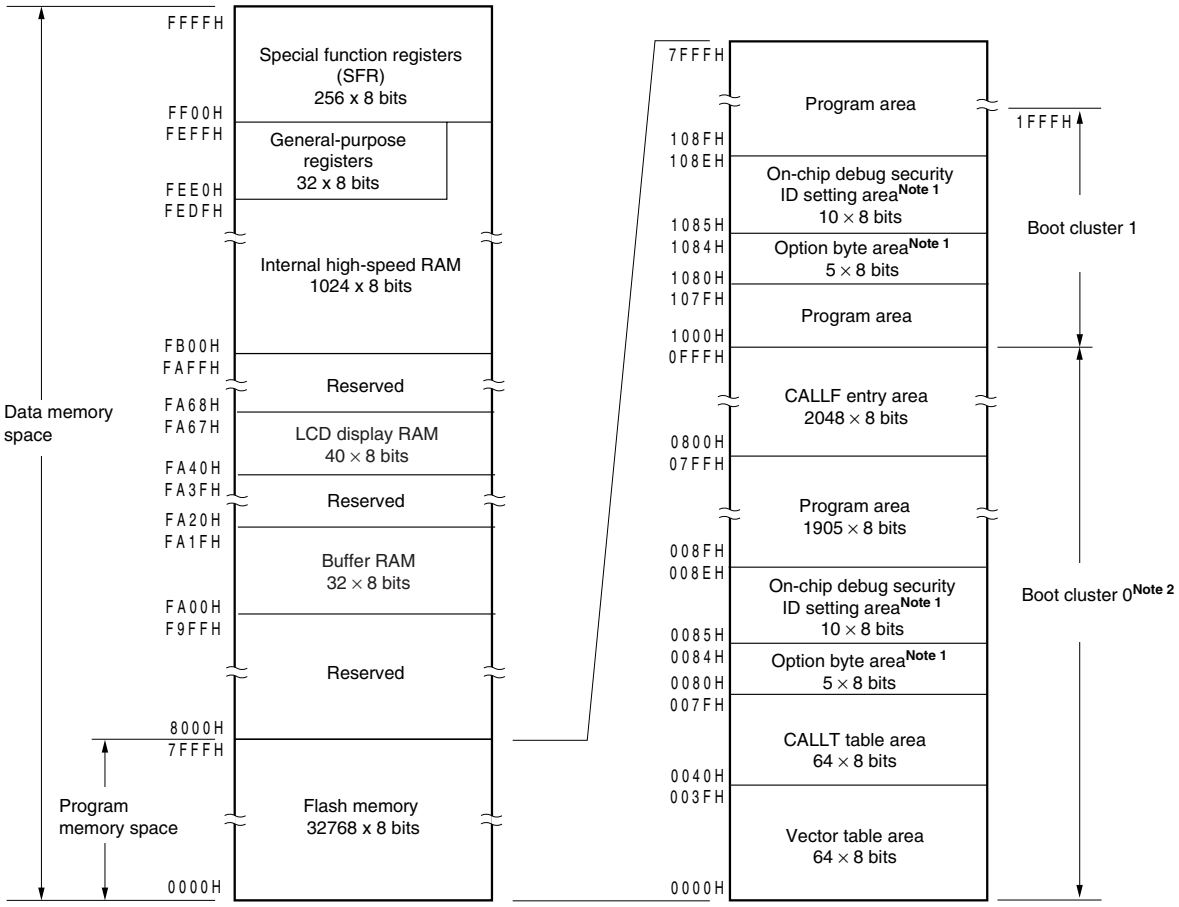
## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | 78K/0 |
| Core Size | 8-Bit |
| Speed | 10MHz |
| Connectivity | 3-Wire SIO, LINbus, UART/USART |
| Peripherals | LCD, LVD, POR, PWM, WDT |
| Number of I/O | 62 |
| Program Memory Size | 48KB (48K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 5.5V |
| Data Converters | A/D 8x10b, 3x16b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 80-LQFP |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/renesas-electronics-america/upd78f0494gc-gad-ax |

**Figure 3-5. Memory Map (μPD78F0473, 78F0483)**

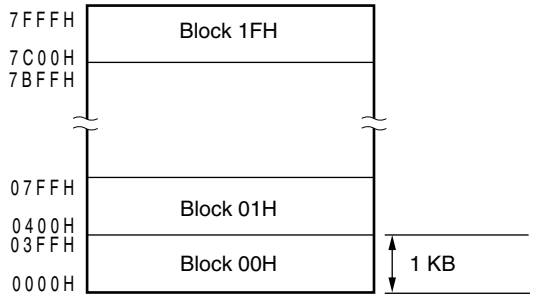



**Notes 1.** When boot swap is not used: Set the option bytes to 0080H to 0084H, and the on-chip debug security IDs to 0085H to 008EH.

When boot swap is used: Set the option bytes to 0080H to 0084H and 1080H to 1084H, and the on-chip debug security IDs to 0085H to 008EH and 1085H to 108EH.

**2.** Writing boot cluster 0 can be prohibited depending on the setting of security (see **28.8 Security Setting**).

**Remark** The flash memory is divided into blocks (one block = 1 KB). For the address values and block numbers, see **Table 3-2 Correspondence Between Address Values and Block Numbers in Flash Memory**.



### 4.2.1 Port 1

Port 1 is an 8-bit I/O port with an output latch. Port 1 can be set to the input mode or output mode in 1-bit units using port mode register 1 (PM1). When the P10 to P17 pins are used as an input port, use of an on-chip pull-up resistor can be specified in 1-bit units by pull-up resistor option register 1 (PU1).

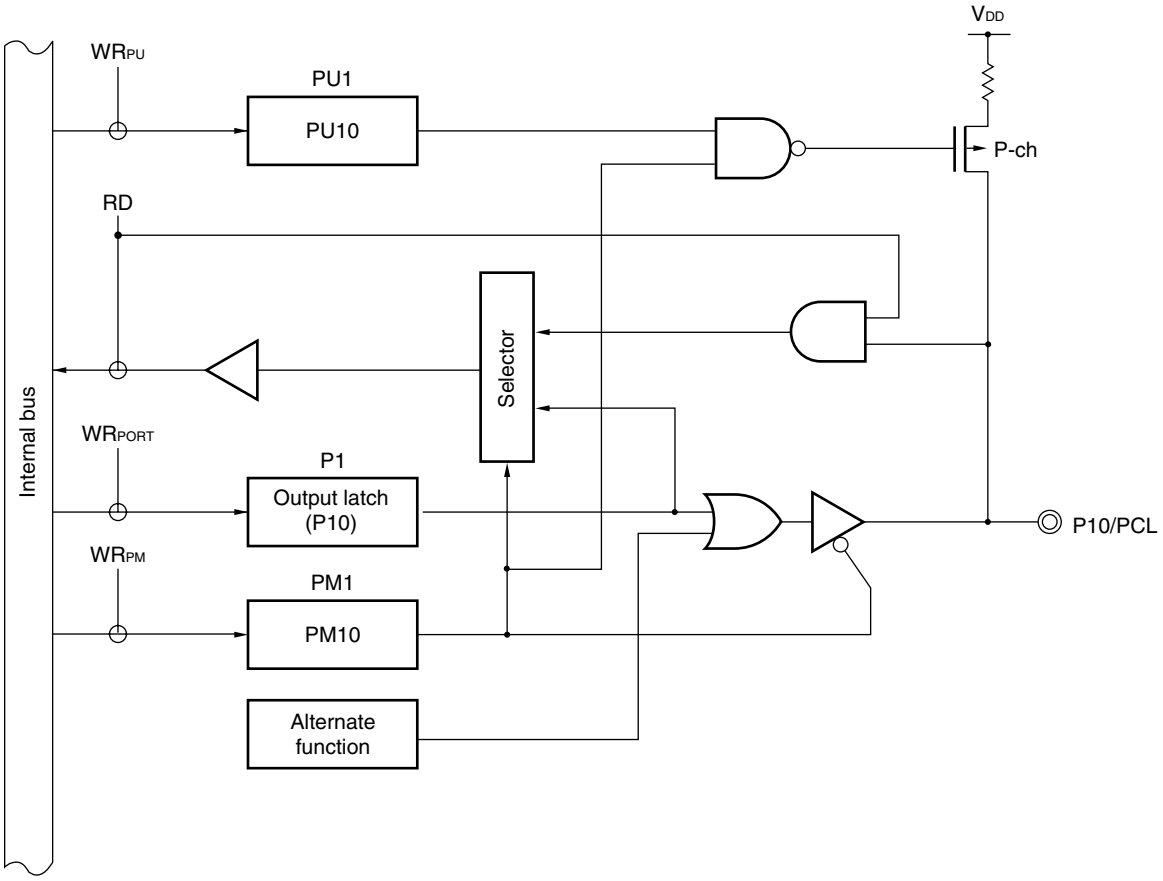
This port can also be used for serial clock I/O, serial interface data I/O, and maskable external interrupt input.

Reset signal generation sets port 1 to input mode.

Figures 4-2 to 4-7 show block diagrams of port 1.

**Caution To use P11/$\overline{\text{SCK10}}$, P12/SI10/RxD0, and P13/SO10/TxD0 as general-purpose ports, set serial operation mode register 10 (CSIM10) and serial clock selection register 10 (CSIC10) to the default status (00H).**

**Figure 4-2. Block Diagram of P10**




P1: Port register 1
PU1: Pull-up resistor option register 1
PM1: Port mode register 1
RD: Read signal
WR××: Write signal

**Remarks 1.** $f_X$: X1 clock oscillation frequency
**2.** $f_{RH}$: Internal high-speed oscillation clock frequency
**3.** $f_{EXCLK}$: External main system clock frequency
**4.** $f_{XH}$: High-speed system clock frequency
**5.** $f_{XP}$: Main system clock frequency
**6.** $f_{PRS}$: Peripheral hardware clock frequency
**7.** $f_{CPU}$: CPU clock frequency
**8.** $f_{XT}$: XT1 clock oscillation frequency
**9.** $f_{SUB}$: Subsystem clock frequency
**10.** $f_{RL}$: Internal low-speed oscillation clock frequency

## 5.3 Registers Controlling Clock Generator

The following eight registers are used to control the clock generator.

- Clock operation mode select register (OSCCTL)
- Processor clock control register (PCC)
- Internal oscillation mode register (RCM)
- Main OSC control register (MOC)
- Main clock mode register (MCM)
- Oscillation stabilization time counter status register (OSTC)
- Oscillation stabilization time select register (OSTS)
- Internal high-speed oscillation trimming register (HIOTRM)

**(1) Clock operation mode select register (OSCCTL)**

This register selects the operation modes of the high-speed system and subsystem clocks, and the gain of the on-chip oscillator.

OSCCTL can be set by a 1-bit or 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

**(3) Example of setting procedure when stopping the internal high-speed oscillation clock**

The internal high-speed oscillation clock can be stopped in the following two ways.

- Executing the STOP instruction to set the STOP mode
- Setting RSTOP to 1 and stopping the internal high-speed oscillation clock

**(a) To execute a STOP instruction**

<1> Setting of peripheral hardware

Stop peripheral hardware that cannot be used in the STOP mode (for peripheral hardware that cannot be used in STOP mode, see **CHAPTER 23 STANDBY FUNCTION**).

<2> Setting the X1 clock oscillation stabilization time after standby release

When the CPU is operating on the X1 clock, set the value of the OSTS register before the STOP instruction is executed.

<3> Executing the STOP instruction

When the STOP instruction is executed, the system is placed in the STOP mode and internal high-speed oscillation clock is stopped.

**(b) To stop internal high-speed oscillation clock by setting RSTOP to 1**

<1> Confirming the CPU clock status (PCC and MCM registers)

Confirm with CLS and MCS that the CPU is operating on a clock other than the internal high-speed oscillation clock.

When CLS = 0 and MCS = 0, the internal high-speed oscillation clock is supplied to the CPU, so change the CPU clock to the high-speed system clock or subsystem clock.

| CLS | MCS | CPU Clock Status |
|---|---|---|
| 0 | 0 | Internal high-speed oscillation clock |
| 0 | 1 | High-speed system clock |
| 1 | × | Subsystem clock |

<2> Stopping the internal high-speed oscillation clock (RCM register)

When RSTOP is set to 1, internal high-speed oscillation clock is stopped.

**Caution Be sure to confirm that MCS = 1 or CLS = 1 when setting RSTOP to 1. In addition, stop peripheral hardware that is operating on the internal high-speed oscillation clock.**

### 5.6.3 Example of controlling subsystem clock

The following two types of subsystem clocks are available.

- XT1 clock: Crystal/ceramic resonator is connected across the XT1 and XT2 pins.

When the subsystem clock is not used, the XT1/P123 and XT2/P124 pins can be used as Input port pins.

**Caution The XT1/P123 and XT2/P124 pins are in the Input port mode after a reset release.**

The following describes examples of setting procedures for the following cases.

(1) When oscillating XT1 clock

(2) When using subsystem clock as CPU clock

(3) When stopping subsystem clock

**Table 5-5. CPU Clock Transition and SFR Register Setting Examples (4/4)**

**(9) CPU clock changing from subsystem clock (D) to high-speed system clock (C)**

(Setting sequence of SFR registers) →

| Setting Flag of SFR Register / Status Transition | EXCLK | OSCSEL | MSTOP | OSTC Register | XSEL[Note] | MCM0 | CSS |
|---|---|---|---|---|---|---|---|
| (D) → (C) (X1 clock) | 0 | 1 | 0 | Must be checked | 1 | 1 | 0 |
| (D) → (C) (external main clock) | 1 | 1 | 0 | Must not be checked | 1 | 1 | 0 |

Unnecessary if these registers are already set

Unnecessary if the CPU is operating with the high-speed system clock

**Note** The value of this flag can be changed only once after a reset release. This setting is not necessary if it has already been set.

**Caution Set the clock after the supply voltage has reached the operable voltage of the clock to be set (see CHAPTER 31 ELECTRICAL SPECIFICATIONS (STANDARD PRODUCTS)).**

**(10) • HALT mode (E) set while CPU is operating with internal high-speed oscillation clock (B)**
**• HALT mode (F) set while CPU is operating with high-speed system clock (C)**
**• HALT mode (G) set while CPU is operating with subsystem clock (D)**

| Status Transition | Setting |
|---|---|
| (B) → (E)<br>(C) → (F)<br>(D) → (G) | Executing HALT instruction |

**(11) • STOP mode (H) set while CPU is operating with internal high-speed oscillation clock (B)**
**• STOP mode (I) set while CPU is operating with high-speed system clock (C)**

(Setting sequence) →

| Status Transition | Setting | |
|---|---|---|
| (B) → (H)<br>(C) → (I) | Stopping peripheral functions that cannot operate in STOP mode | Executing STOP instruction |

**Remarks 1.** (A) to (I) in Table 5-5 correspond to (A) to (I) in Figure 5-15.

**2.** EXCLK, OSCSEL: Bits 7 and 6 of the clock operation mode select register (OSCCTL)
MSTOP: Bit 7 of the main OSC control register (MOC)
XSEL, MCM0: Bits 2 and 0 of the main clock mode register (MCM)
CSS: Bit 4 of the processor clock control register (PCC)

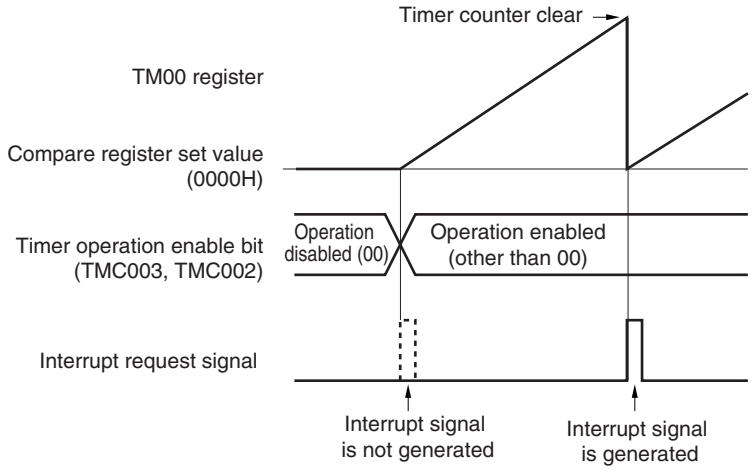
**(iii) Setting range when CR000 or CR010 is used as a compare register**

When CR000 or CR010 is used as a compare register, set it as shown below.

| Operation | CR000 Register Setting Range | CR010 Register Setting Range |
|---|---|---|
| Operation as interval timer | 0000H < N ≤ FFFFH | 0000H[Note] ≤ M ≤ FFFFH<br>Normally, this setting is not used. Mask the match interrupt signal (INTTM010). |
| Operation as square-wave output | | |
| Operation as external event counter | | |
| Operation in the clear & start mode entered by TI000 pin valid edge input | 0000H[Note] ≤ N ≤ FFFFH | 0000H[Note] ≤ M ≤ FFFFH |
| Operation as free-running timer | | |
| Operation as PPG output | M < N ≤ FFFFH | 0000H[Note] ≤ M < N |
| Operation as one-shot pulse output | 0000H[Note] ≤ N ≤ FFFFH (N ≠ M) | 0000H[Note] ≤ M ≤ FFFFH (M ≠ N) |

**Note** When 0000H is set, a match interrupt immediately after the timer operation does not occur and timer output is not changed, and the first match timing is as follows. A match interrupt occurs at the timing when the timer counter (TM00 register) is changed from 0000H to 0001H.

- When the timer counter is cleared due to overflow
- When the timer counter is cleared due to TI000 pin valid edge (when clear & start mode is entered by TI000 pin valid edge input)
- When the timer counter is cleared due to compare match (when clear & start mode is entered by match between TM00 and CR000 (CR000 = other than 0000H, CR010 = 0000H))

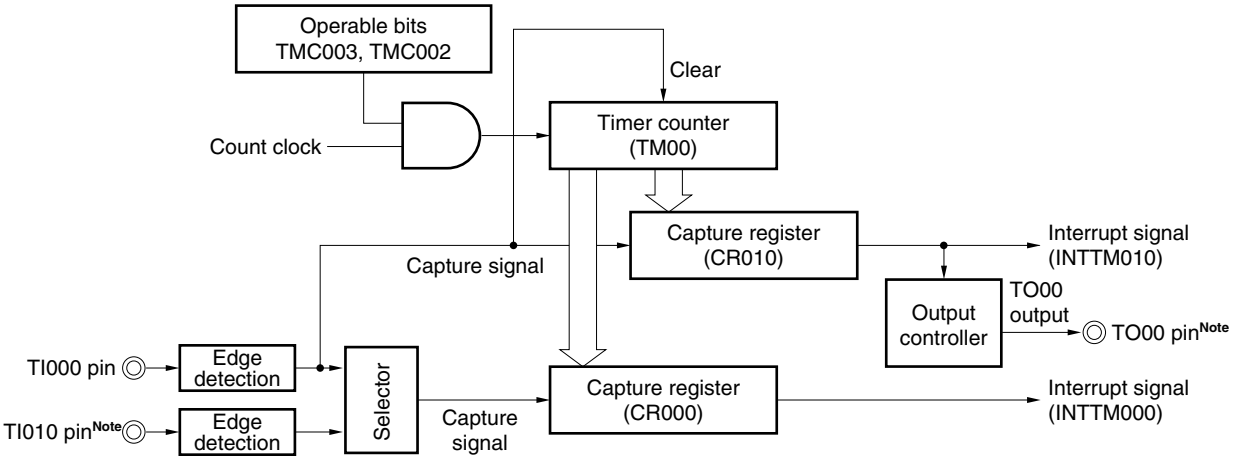



**Remarks 1.** N: CR000 register set value, M: CR010 register set value

**2.** For details of TMC003 and TMC002, see **6.3 (1) 16-bit timer mode control register 00 (TMC00)**.

**(4) Operation in clear & start mode entered by TI000 pin valid edge input (CR000: capture register, CR010: capture register)**

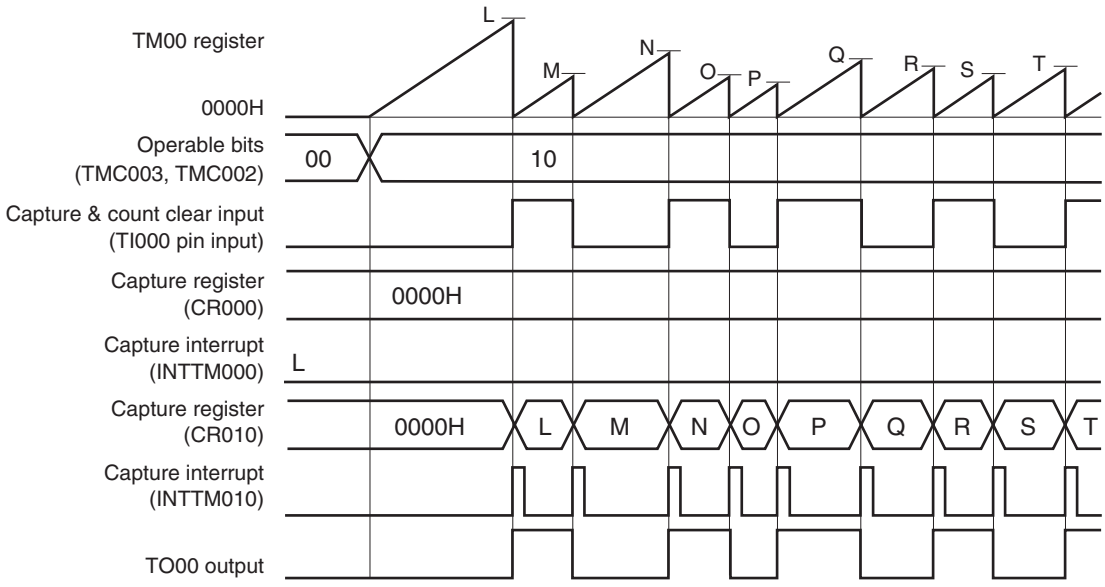
**Figure 6-29. Block Diagram of Clear & Start Mode Entered by TI000 Pin Valid Edge Input (CR000: Capture Register, CR010: Capture Register)**




**Note** The timer output (TO00) cannot be used when detecting the valid edge of the TI010 pin is used.

**Figure 6-30. Timing Example of Clear & Start Mode Entered by TI000 Pin Valid Edge Input (CR000: Capture Register, CR010: Capture Register) (1/3)**

**(a) TOC00 = 13H, PRM00 = 30H, CRC00 = 05H, TMC00 = 0AH**

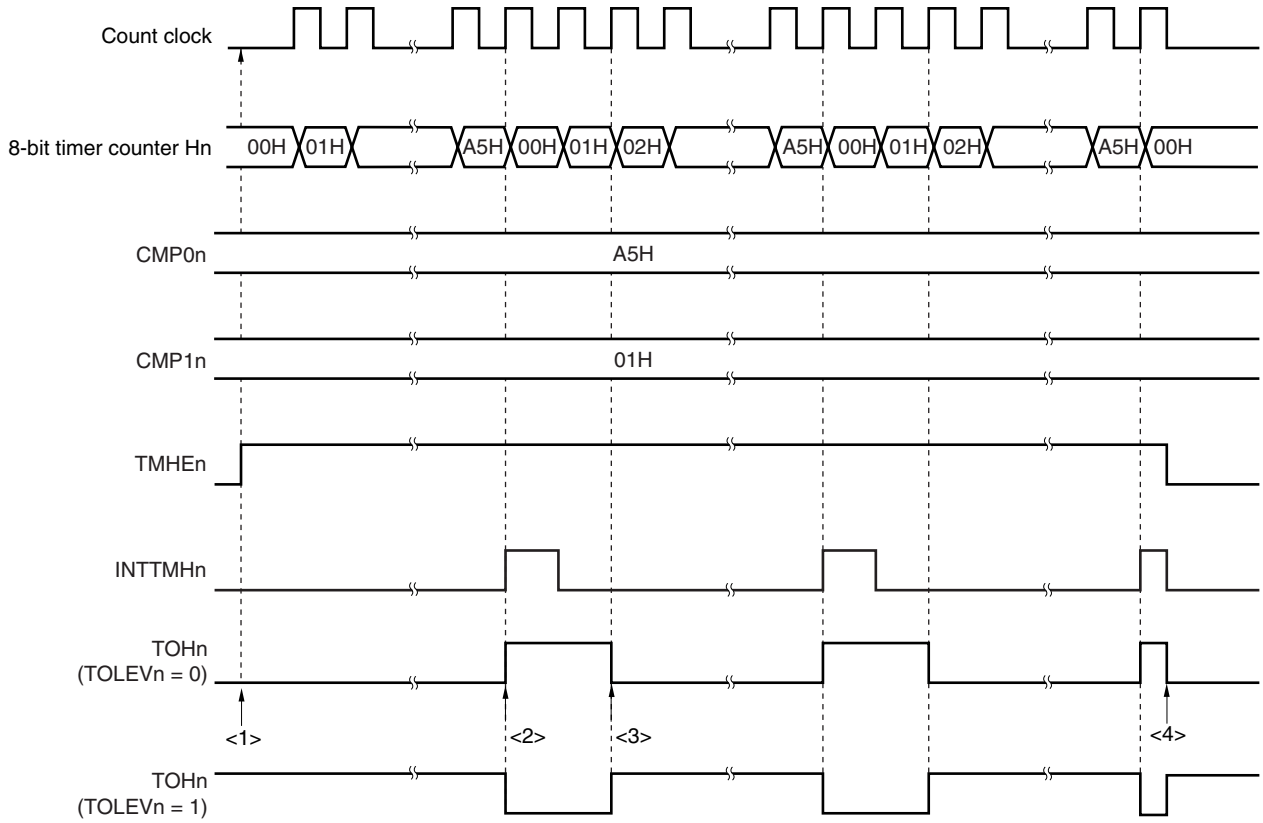



This is an application example where the count value is captured to CR010, TM00 is cleared, and the TO00 output is inverted when the rising or falling edge of the TI000 pin is detected.

When the edge of the TI010 pin is detected, an interrupt signal (INTTM000) is generated. Mask the INTTM000 signal when it is not used.

**Figure 8-14. Operation Timing in PWM Output Mode (1/4)**

**(a) Basic operation**




<1> The count operation is enabled by setting the TMHEn bit to 1. Start 8-bit timer counter Hn by masking one count clock to count up. At this time, PWM output outputs an inactive level.

<2> When the values of 8-bit timer counter Hn and the CMP0n register match, an active level is output. At this time, the value of 8-bit timer counter Hn is cleared, and the INTTMHn signal is output.

<3> When the values of 8-bit timer counter Hn and the CMP1n register match, an inactive level is output. At this time, the 8-bit counter value is not cleared and the INTTMHn signal is not output.

<4> Clearing the TMHEn bit to 0 during timer Hn operation sets the INTTMHn signal to the default and PWM output to an inactive level.

**Remark** n = 0 to 2, however, TOH0 and TOH1 only for TOHn

**(5) Sub-count register (RSUBC)**

The RSUBC register is a 16-bit register that counts the reference time of 1 second of the real-time counter. It takes a value of 0000H to 7FFFH and counts 1 second with a clock of 32.768 kHz.

RSUBC can be set by a 16-bit memory manipulation instruction.

Reset signal generation clears this register to 0000H.

**Cautions 1. When a correction is made by using the SUBCUD register, the value may become 8000H or more.**

**2. This register is also cleared by reset effected by writing the second count register.**

**3. The value read from this register is not guaranteed if it is read during operation, because a value that is changing is read.**

**Figure 9-6. Format of Sub-Count Register (RSUBC)**

Address: FF60H After reset: 0000H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RSUBC | SUBC7 | SUBC6 | SUBC5 | SUBC4 | SUBC3 | SUBC2 | SUBC1 | SUBC0 |

Address: FF61H After reset: 0000H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RSUBC | SUBC15 | SUBC14 | SUBC13 | SUBC12 | SUBC11 | SUBC10 | SUBC9 | SUBC8 |

**(6) Second count register (SEC)**

The SEC register is an 8-bit register that takes a value of 0 to 59 (decimal) and indicates the count value of seconds.

It counts up when the sub-counter overflows.

When data is written to this register, it is written to a buffer and then to the counter up to 2 clocks (32.768 kHz) later. Set a decimal value of 00 to 59 to this register in BCD code. If a value outside this range is set, the register value returns to the normal value after 1 period.

SEC can be set by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 00H.

**Figure 9-7. Format of Second Count Register (SEC)**

Address: FF62H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SEC | 0 | SEC40 | SEC20 | SEC10 | SEC8 | SEC4 | SEC2 | SEC1 |

**(9) Day count register (DAY)**

The DAY register is an 8-bit register that takes a value of 1 to 31 (decimal) and indicates the count value of days.

It counts up when the hour counter overflows.

This counter counts as follows.

- 01 to 31 (January, March, May, July, August, October, December)
- 01 to 30 (April, June, September, November)
- 01 to 29 (February, leap year)
- 01 to 28 (February, normal year)

When data is written to this register, it is written to a buffer and then to the counter up to 2 clocks (32.768 kHz) later. Set a decimal value of 01 to 31 to this register in BCD code. If a value outside this range is set, the register value returns to the normal value after 1 period.

DAY can be set by an 8-bit memory manipulation instruction.

Reset signal generation clears this register to 01H.

**Figure 9-10. Format of Day Count Register (DAY)**

Address: FF66H After reset: 01H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| DAY | 0 | 0 | DAY20 | DAY10 | DAY8 | DAY4 | DAY2 | DAY1 |

**(6) Automatic data transfer interval specification register 0 (ADTI0)**

This is an 8-bit register used to specify the interval time for byte data transfer during automatic data transfer (bit 6 (ATE0) of serial operation mode specification register 0 (CSIMA0) = 1).

Set this register when in master mode (bit 4 (MASTER0) of CSIMA0 = 1) (setting is unnecessary in slave mode). Setting in 1-byte communication mode (bit 6 (ATE0) of CSIMA0 = 0) is also valid. When the interval time specified by ADTI0 after the end of 1-byte communication has elapsed, an interrupt request signal (INTACSI) is output. The number of clocks for the interval can be set to between 0 and 63 clocks.

This register can be set by an 8-bit memory manipulation instruction. However, when bit 0 (TSF0) of serial status register 0 (CSIS0) is 1, rewriting ADTI0 is prohibited.

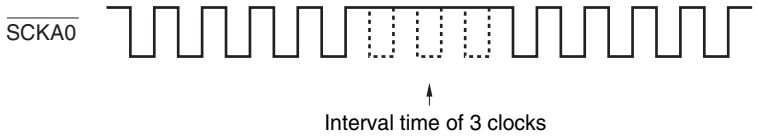
**Figure 17-7. Format of Automatic Data Transfer Interval Specification Register 0 (ADTI0)**

Address: FF95H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADTI0 | 0 | 0 | ADTI05 | ADTI04 | ADTI03 | ADTI02 | ADTI01 | ADTI00 |

The specified interval time is the serial clock (specified by divisor selection register 0 (BRGCA0)) multiplied by an integer value.

**Example** When ADTI0 = 03H



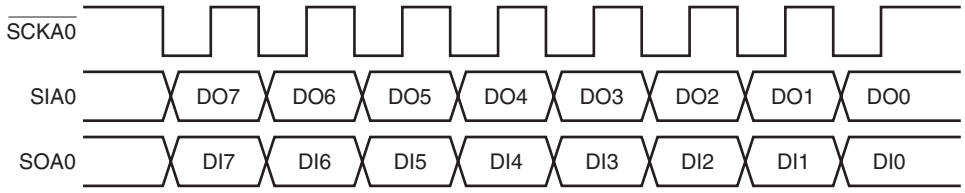
**(d) Data format**

Data is changed in synchronization with the $\overline{\text{SCKA0}}$ falling edge as shown below.

The data length is fixed to 8 bits and the data transfer direction can be switched by the specification of bit 1 (DIR0) of serial operation mode specification register 0 (CSIMA0).

**Figure 17-22. Format of CSIA0 Transmit/Receive Data**

**(a) MSB-first (DIR0 bit = 0)**




**(b) LSB-first (DIR0 bit = 1)**

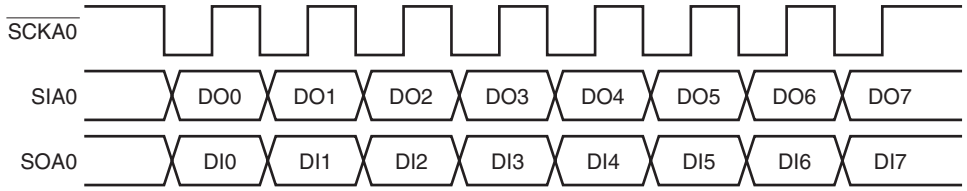
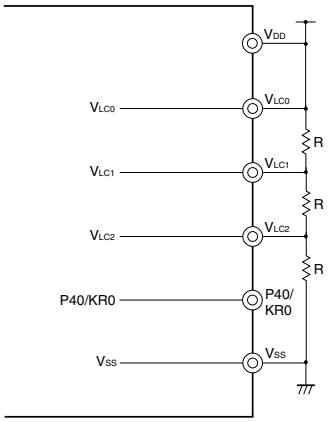


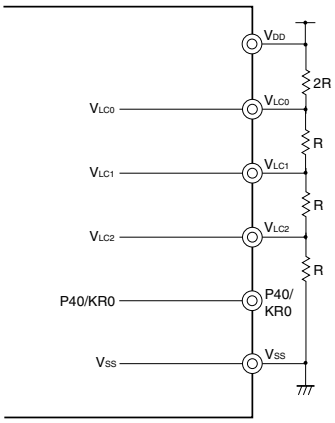
**Figure 18-40. Examples of LCD Drive Power Connections (External Resistance Division Method) (2/2)**

**(e) 1/3 bias method**
**(MDSET1, MDSET0 = 0, 0)**
**(example of $V_{DD}$ = 5 V, $V_{LC0}$ = 5 V)**



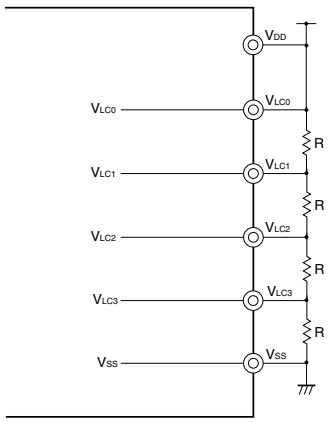

$V_{LC0} = V_{DD}$

**(f) 1/3 bias method**
**(MDSET1, MDSET0 = 0, 0)**
**(example of $V_{DD}$ = 5 V, $V_{LC0}$ = 3 V)**



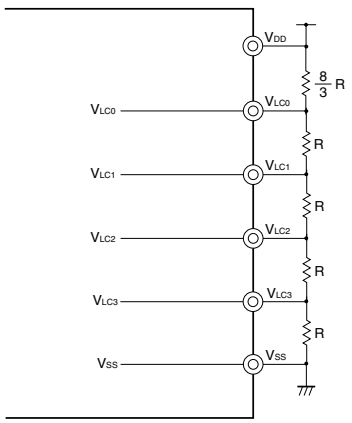

$V_{LC0} = \frac{3}{5} V_{DD}$

**(g) 1/4 bias method**
**(MDSET1, MDSET0 = 0, 0)**
**(example of $V_{DD}$ = 5 V, $V_{LC0}$ = 5 V)**




$V_{LC0} = V_{DD}$

**(h) 1/4 bias method**
**(MDSET1, MDSET0 = 0, 0)**
**(example of $V_{DD}$ = 5 V, $V_{LC0}$ = 3 V)**




$V_{LC0} = \frac{3}{5} V_{DD}$

**(c) MCG control register 2 (MC0CTL2)**

This register is used to set the transmit baud rate.

This register can be set by an 8-bit memory manipulation instruction.

Reset signal generation sets this register to 1FH.

Address: FF4EH After reset: 1FH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MC0CTL2 | 0 | 0 | 0 | MC0BRS4 | MC0BRS3 | MC0BRS2 | MC0BRS1 | MC0BRS0 |

| MC0BRS4 | MC0BRS3 | MC0BRS2 | MC0BRS1 | MC0BRS0 | k | Output clock selection of 5-bit counter |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | × | × | 4 | $f_{XCLK}/4$ |
| 0 | 0 | 1 | 0 | 0 | 4 | $f_{XCLK}/4$ |
| 0 | 0 | 1 | 0 | 1 | 5 | $f_{XCLK}/5$ |
| 0 | 0 | 1 | 1 | 0 | 6 | $f_{XCLK}/6$ |
| 0 | 0 | 1 | 1 | 1 | 7 | $f_{XCLK}/7$ |
| • | • | • | • | • | • | • |
| • | • | • | • | • | • | • |
| • | • | • | • | • | • | • |
| • | • | • | • | • | • | • |
| • | • | • | • | • | • | • |
| 1 | 1 | 1 | 0 | 0 | 28 | $f_{XCLK}/28$ |
| 1 | 1 | 1 | 0 | 1 | 29 | $f_{XCLK}/29$ |
| 1 | 1 | 1 | 1 | 0 | 30 | $f_{XCLK}/30$ |
| 1 | 1 | 1 | 1 | 1 | 31 | $f_{XCLK}/31$ |

**Cautions 1. Clear bit 7 (MC0PWR) of the MC0CTL0 register to 0 before rewriting the MC0BRS4 to MC0BRS0 bits.**

**2. The value from further dividing the output clock of the 5-bit counter by 2 is the baud rate value.**

**Remarks 1.** $f_{XCLK}$: Frequency of the base clock selected by the MC0CKS2 to MC0CKS0 bits of the MC0CTL1 register

**2.** k: Value set by the MC0BRS4 to MC0BRS0 bits (k = 4, 5, 6, 7, …., 31)

**3.** ×: Don't care

**<1> Baud rate**

The baud rate can be calculated by the following expression.

- Baud rate = $\frac{f_{XCLK}}{2 \times k}$ [bps]

$f_{XCLK}$: Frequency of base clock selected by the MC0CKS2 to MC0CKS0 bits of the MC0CTL1 register

k: Value set by the MC0BRS4 to MC0BRS0 bits of the MC0CTL2 register (k = 4, 5, 6, ..., 31)

**<2> Error of baud rate**

The baud rate error can be calculated by the following expression.

- $\text{Error (\%)} = \left( \frac{\text{Actual baud rate (baud rate with error)}}{\text{Desired baud rate (correct baud rate)}} - 1 \right) \times 100 \text{ [\%]}$

**Caution Keep the baud rate error during transmission to within the permissible error range at the reception destination.**

**Example:** Frequency of base clock = 2.5 MHz = 2,500,000 Hz
Set value of MC0BRS4 to MC0BRS0 bits of MC0CTL2 register = 10000B (k = 16)
Target baud rate = 76,800 bps

Baud rate = 2.5 M/(2 × 16)
= 2,500,000/(2 × 16) = 78125 [bps]

Error = (78,125/76,800 − 1) × 100
= 1.725 [%]

**<3> Example of setting baud rate**

| Baud Rate [bps] | $f_{PRS}$ = 10.0 MHz | | | | $f_{PRS}$ = 8.38 MHz | | | | $f_{PRS}$ = 8.0 MHz | | | | $f_{PRS}$ = 6.0 MHz | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MC0CKS2 to MC0CKS0 | k | Calculated Value | ERR [%] | MC0CKS2 to MC0CKS0 | k | Calculated Value | ERR [%] | MC0CKS2 to MC0CKS0 | k | Calculated Value | ERR [%] | MC0CKS2 to MC0CKS0 | k | Calculated Value | ERR [%] |
| 4800 | – | – | – | – | 5, 6, or 7 | 27 | 4850 | 1.03 | 5, 6, or 7 | 26 | 4808 | 0.16 | 5, 6, or 7 | 20 | 4688 | –2.34 |
| 9600 | 5, 6, or 7 | 16 | 9766 | 1.73 | 4 | 27 | 9699 | 1.03 | 5, 6, or 7 | 13 | 9615 | 0.16 | 4 | 20 | 9375 | –2.34 |
| 19200 | 5 | 8 | 19531 | 1.73 | 3 | 27 | 19398 | 1.03 | 4 | 13 | 19231 | 0.16 | 4 | 10 | 18750 | –2.34 |
| 31250 | 4 | 10 | 31250 | 0 | 2 | 17 | 30809 | –1.41 | 4 | 8 | 31250 | 0 | 2 | 24 | 31250 | 0 |
| 38400 | 4 | 8 | 39063 | 1.73 | 2 | 27 | 38796 | 1.03 | 3 | 13 | 38462 | 0.16 | 2 | 20 | 37500 | –2.34 |
| 56000 | 3 | 11 | 56818 | 1.46 | 2 | 19 | 55132 | –1.55 | 3 | 9 | 55556 | –0.79 | 1 | 27 | 55556 | –0.79 |
| 62500 | 2 | 20 | 62500 | 0 | 2 | 17 | 61618 | –1.41 | 3 | 8 | 62500 | 0 | 2 | 12 | 62500 | 0 |
| 76800 | 2 | 16 | 78125 | 1.73 | 1 | 27 | 77592 | 1.03 | 2 | 13 | 76923 | 0.16 | 2 | 10 | 75000 | –2.34 |
| 115200 | 1 | 22 | 113636 | –1.36 | 2 | 9 | 116389 | 1.03 | 1 | 17 | 117647 | 2.12 | 1 | 13 | 115385 | 0.16 |
| 125000 | 1 | 20 | 125000 | 0 | 1 | 17 | 123235 | –1.41 | 1 | 16 | 125000 | 0 | 1 | 12 | 125000 | 0 |
| 153600 | 1 | 16 | 156250 | 1.73 | 2 | 7 | 149643 | –2.58 | 1 | 13 | 153846 | 0.16 | 1 | 10 | 150000 | –2.34 |
| 250000 | 1 | 10 | 250000 | 0 | 1 | 8 | 261875 | 4.75 | 1 | 8 | 250000 | 0 | 1 | 6 | 250000 | 0 |
| | | | | | 0 | 17 | 246471 | –1.41 | | | | | | | | |

**Remark** MC0CKS2 to MC0CKS0: Bits 2 to 0 of MCG control register 1 (MC0CTL1) (setting of base clock ($f_{XCLK}$))
k: Value set by bits 4 to 0 (MC0BRS4 to MC0BRS0) of MCG control register 2 (MC0CTL2) (k = 4, 5, 6, …, 31)
$f_{PRS}$: Peripheral hardware clock frequency
ERR: Baud rate error

## 23.2 Standby Function Operation

### 23.2.1 HALT mode

**(1) HALT mode**

The HALT mode is set by executing the HALT instruction. HALT mode can be set regardless of whether the CPU clock before the setting was the high-speed system clock, internal high-speed oscillation clock, or subsystem clock.

The operating statuses in the HALT mode are shown below.

**(2) When detecting level of input voltage from external input pin (EXLVI)**

- When starting operation
  - <1> Mask the LVI interrupt (LVIMK = 1).
  - <2> Set bit 2 (LVISEL) of the low-voltage detection register (LVIM) to 1 (detects level of input voltage from external input pin (EXLVI)).
  - <3> Set bit 7 (LVION) of LVIM to 1 (enables LVI operation).
  - <4> Use software to wait for an operation stabilization time (10 $\mu$s (MAX.)).
  - <5> Wait until it is checked that (input voltage from external input pin (EXLVI) ≥ detection voltage ($V_{EXLVI}$ = 1.21 V (TYP.))) by bit 0 (LVIF) of LVIM.
  - <6> Set bit 1 (LVIMD) of LVIM to 1 (generates reset signal when the level is detected).

  Figure 26-6 shows the timing of the internal reset signal generated by the low-voltage detector. The numbers in this timing chart correspond to <1> to <6> above.

  **Cautions 1. <1> must always be executed. When LVIMK = 0, an interrupt may occur immediately after the processing in <3>.**
  **2. If input voltage from external input pin (EXLVI) ≥ detection voltage ($V_{EXLVI}$ = 1.21 V (TYP.)) when LVIMD is set to 1, an internal reset signal is not generated.**
  **3. Input voltage from external input pin (EXLVI) must be EXLVI < $V_{DD}$.**

- When stopping operation
  Either of the following procedures must be executed.

  - When using 8-bit memory manipulation instruction:
    Write 00H to LVIM.

  - When using 1-bit memory manipulation instruction:
    Clear LVIMD to 0 and then LVION to 0.

### 28.7.3 Selecting communication mode

In the 78K0/LF3, a communication mode is selected by inputting pulses to the FLMD0 pin after the dedicated flash memory programming mode is entered. These FLMD0 pulses are generated by the flash memory programmer.

The following table shows the relationship between the number of pulses and communication modes.

**Table 28-7. Communication Modes**

| Communication Mode | Standard Setting[Note 1] | | | | Pins Used | Peripheral Clock | Number of FLMD0 Pulses |
|---|---|---|---|---|---|---|---|
| | Port | Speed | Frequency | Multiply Rate | | | |
| UART (UART6) | UART-Ext-OSC | 115,200 bps[Note 3] | 2 M to 10 MHz[Note 2] | 1.0 | TxD6, RxD6 | $f_X$ | 0 |
| | UART-Ext-FP5CLK | | | | | $f_{EXCLK}$ | 3 |
| <R> | UART-Internal-OSC | | – | | | $f_{RH}$ | 5 |
| 3-wire serial I/O (CSI10) | CSI-Internal-OSC | 2.4 kHz to 2.5 MHz | – | | SO10, SI10, $\overline{SCK10}$ | $f_{RH}$ | 8 |

**Notes 1.** Selection items for Standard settings on GUI of the flash memory programmer.

**2.** The possible setting range differs depending on the voltage. For details, see **CHAPTER 31 ELECTRICAL SPECIFICATIONS (STANDARD PRODUCTS)**.

**3.** Because factors other than the baud rate error, such as the signal waveform slew, also affect UART communication, thoroughly evaluate the slew as well as the baud rate error.

**Caution When UART6 is selected, the receive clock is calculated based on the reset command sent from the dedicated flash memory programmer after the FLMD0 pulse has been received.**

**Remark** $f_X$: X1 clock
$f_{EXCLK}$: External main system clock
$f_{RH}$: Internal high-speed oscillation clock

**Table 28-13. Processing Time and Interrupt Acknowledgment (4/4)**
**(When Static Model Library and Entry RAM Are Allocated Within Short Direct Addressing Range)**

| Function Name | | Processing Time (Unit: μs) | | | Interrupt Acknowledgment |
|---|---|---|---|---|---|
| | | RSTOP = 0 and RSTS = 1 (during stable operation of internal high-speed oscillator) | | RSTOP = 1 (internal high-speed oscillator stopped)[Note] | |
| | | MCS = 0 (CPU operates with internal high-speed oscillation clock) | MCS = 1 (CPU operates with high-speed system clock) | MCS = 1 (CPU operates with high-speed system clock) | |
| Self programming start function | | 34/$f_{CPU}$ | 34/$f_{CPU}$ | 34/$f_{CPU}$ | Disabled |
| Self programming end function | | 34/$f_{CPU}$ | 34/$f_{CPU}$ | 34/$f_{CPU}$ | Disabled |
| Initialize function | | 55/$f_{CPU}$+462 | 55/$f_{CPU}$+462 | 55/$f_{CPU}$+473 | Disabled |
| Block erase function | | 136/$f_{CPU}$+352516 | 136/$f_{CPU}$+352516 | 136/$f_{CPU}$+352528 | Enabled |
| Word write function | | 272/$f_{CPU}$+477+2142×W | 272/$f_{CPU}$+477+2142×W | 272/$f_{CPU}$+488+2142×W | Enabled |
| Block verify function | | 136/$f_{CPU}$+24918 | 136/$f_{CPU}$+24918 | 136/$f_{CPU}$+24930 | Enabled |
| Block blank check function | | 136/$f_{CPU}$+12128 | 136/$f_{CPU}$+12128 | 136/$f_{CPU}$+12139 | Enabled |
| Get information function | Option value: 03H | 134/$f_{CPU}$+388 | 134/$f_{CPU}$+388 | 134/$f_{CPU}$+399 | Disabled |
| | Option value: 04H | 144/$f_{CPU}$+378 | 144/$f_{CPU}$+378 | 144/$f_{CPU}$+390 | Disabled |
| | Option value: 05H | 304/$f_{CPU}$+363 | 304/$f_{CPU}$+363 | 304/$f_{CPU}$+375 | Disabled |
| Set information function | | 72/$f_{CPU}$+752540 | 72/$f_{CPU}$+752540 | 72/$f_{CPU}$+753654 | Enabled |
| Mode check function | | 30/$f_{CPU}$+274 | 30/$f_{CPU}$+274 | 30/$f_{CPU}$+286 | Disabled |
| EEPROM write function | | 268/$f_{CPU}$+619+2286×W | 268/$f_{CPU}$+619+2286×W | 268/$f_{CPU}$+630+2286×W | Enabled |

**Note** This is the function processing time when the function is executed immediately after the self programming start function has been executed. The processing time after a function other than the self programming start function has been executed is the same as that of RSTOP = 0.

**Remark** RSTOP: Bit 0 of the internal oscillation mode register (RCM)
RSTS: Bit 7 of RCM
MCS: Bit 1 of main clock mode register (MCM)
$f_{CPU}$: CPU clock frequency
W: Number of words to be written (1 word = 4 bytes)

| Instruction Group | Mnemonic | Operands | Bytes | Clocks | | Operation | Flag |
|---|---|---|---|---|---|---|---|
| | | | | Note 1 | Note 2 | | Z AC CY |
| 8-bit operation | **OR** | A, #byte | 2 | 4 | – | A ← A ∨ byte | × |
| | | saddr, #byte | 3 | 6 | 8 | (saddr) ← (saddr) ∨ byte | × |
| | | A, r Note 3 | 2 | 4 | – | A ← A ∨ r | × |
| | | r, A | 2 | 4 | – | r ← r ∨ A | × |
| | | A, saddr | 2 | 4 | 5 | A ← A ∨ (saddr) | × |
| | | A, !addr16 | 3 | 8 | 9 | A ← A ∨ (addr16) | × |
| | | A, [HL] | 1 | 4 | 5 | A ← A ∨ (HL) | × |
| | | A, [HL + byte] | 2 | 8 | 9 | A ← A ∨ (HL + byte) | × |
| | | A, [HL + B] | 2 | 8 | 9 | A ← A ∨ (HL + B) | × |
| | | A, [HL + C] | 2 | 8 | 9 | A ← A ∨ (HL + C) | × |
| | **XOR** | A, #byte | 2 | 4 | – | A ← A ⊻ byte | × |
| | | saddr, #byte | 3 | 6 | 8 | (saddr) ← (saddr) ⊻ byte | × |
| | | A, r Note 3 | 2 | 4 | – | A ← A ⊻ r | × |
| | | r, A | 2 | 4 | – | r ← r ⊻ A | × |
| | | A, saddr | 2 | 4 | 5 | A ← A ⊻ (saddr) | × |
| | | A, !addr16 | 3 | 8 | 9 | A ← A ⊻ (addr16) | × |
| | | A, [HL] | 1 | 4 | 5 | A ← A ⊻ (HL) | × |
| | | A, [HL + byte] | 2 | 8 | 9 | A ← A ⊻ (HL + byte) | × |
| | | A, [HL + B] | 2 | 8 | 9 | A ← A ⊻ (HL + B) | × |
| | | A, [HL + C] | 2 | 8 | 9 | A ← A ⊻ (HL + C) | × |
| | **CMP** | A, #byte | 2 | 4 | – | A – byte | × × × |
| | | saddr, #byte | 3 | 6 | 8 | (saddr) – byte | × × × |
| | | A, r Note 3 | 2 | 4 | – | A – r | × × × |
| | | r, A | 2 | 4 | – | r – A | × × × |
| | | A, saddr | 2 | 4 | 5 | A – (saddr) | × × × |
| | | A, !addr16 | 3 | 8 | 9 | A – (addr16) | × × × |
| | | A, [HL] | 1 | 4 | 5 | A – (HL) | × × × |
| | | A, [HL + byte] | 2 | 8 | 9 | A – (HL + byte) | × × × |
| | | A, [HL + B] | 2 | 8 | 9 | A – (HL + B) | × × × |
| | | A, [HL + C] | 2 | 8 | 9 | A – (HL + C) | × × × |

**Notes 1.** When the internal high-speed RAM area is accessed or for an instruction with no data access

**2.** When an area except the internal high-speed RAM area is accessed

**3.** Except “r = A”

**Remarks 1.** One instruction clock cycle is one cycle of the CPU clock ($f_{CPU}$) selected by the processor clock control register (PCC).

**2.** This clock cycle applies to the internal ROM program.