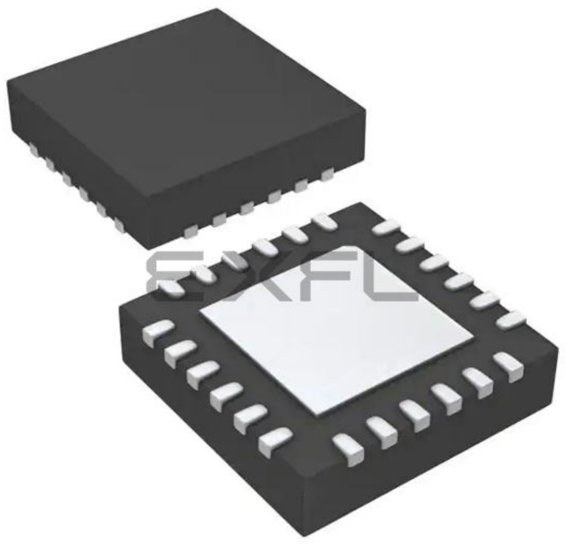
# Renesas Electronics America Inc - R5F10279DNA#W0 Datasheet



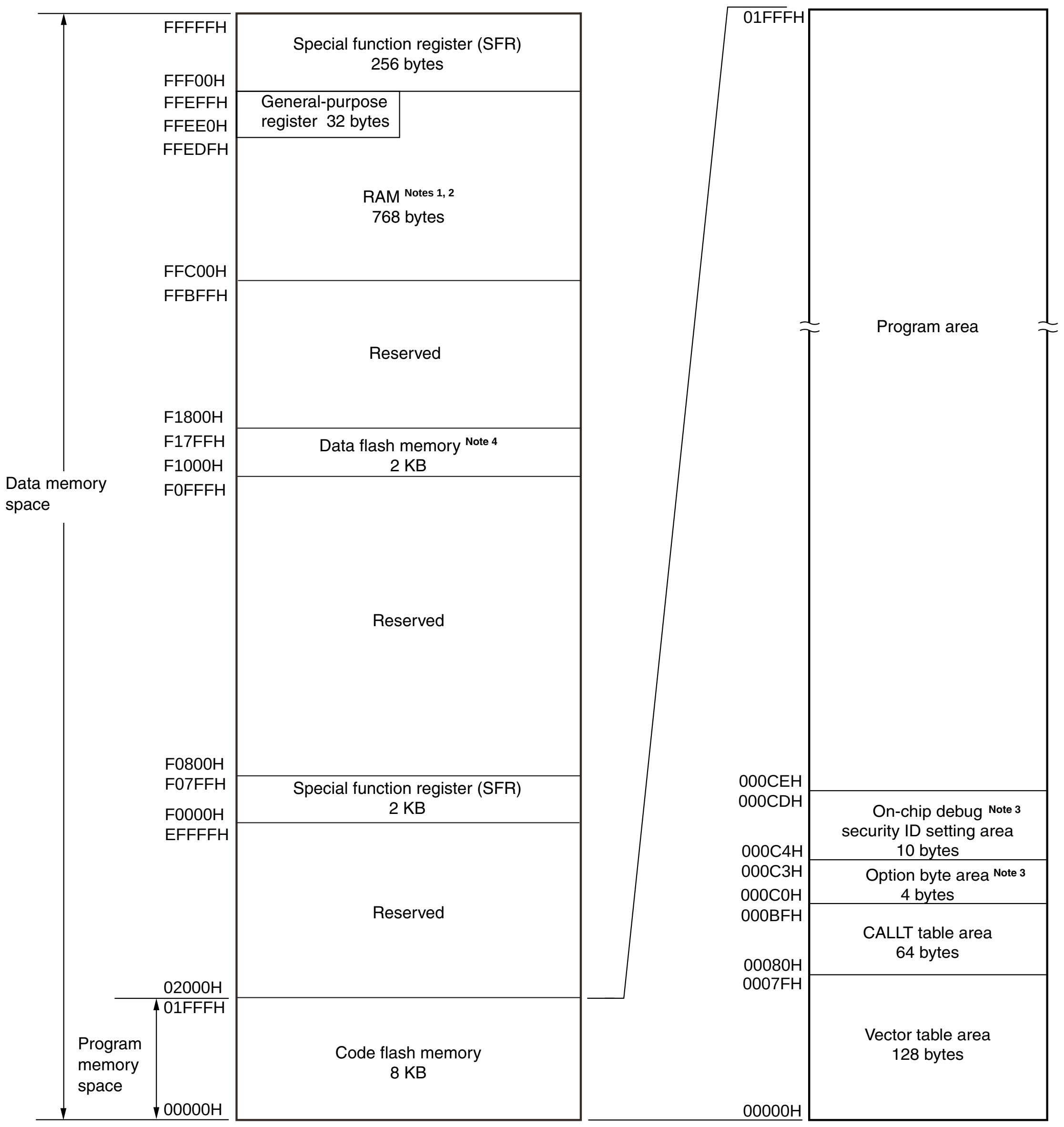
## What is "Embedded - Microcontrollers"?

"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 | Obsolete |
| Core Processor | RL78 |
| Core Size | 16-Bit |
| Speed | 24MHz |
| Connectivity | CSI, I²C, UART/USART |
| Peripherals | DMA, LVD, POR, PWM, WDT |
| Number of I/O | 18 |
| Program Memory Size | 12KB (12K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 2K x 8 |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 5.5V |
| Data Converters | A/D 11x8/10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 24-WFQFN Exposed Pad |
| Supplier Device Package | 24-HWQFN (4x4) |
| Purchase URL | https://www.e-xfl.com/product-detail/renesas-electronics-america/r5f10279dna-w0 |

**Figure 3-3. Memory Map for the R5F10x68, R5F10x78, and R5F10xA8 (x = 2 or 3)**



<R> **Notes 1.** Do not allocate the stack area, data buffers for use by the flash library, arguments of library functions, branch destinations in the processing of vectored interrupts, or destinations or sources for DMA transfer to the area from FFE20H to FFEDFH when performing self-programming or rewriting of the data flash memory. For R5F10x68 and R5F10x78, the RAM area used by the flash library starts at FFC00H. For the RAM areas used by the flash library, see Self RAM list of Flash Self-Programming Library for RL78 Family (R20UT2944).

**2.** Instructions can be executed from the RAM area excluding the general-purpose register area.

**3.** Set the option bytes to 000C0H to 000C3H, and the on-chip debug security IDs to 000C4H to 000CDH.

**4.** The areas are reserved in the R5F10368, R5F10378, and R5103A8.

**Caution** While RAM parity error resets are enabled (RPERDIS = 0), be sure to initialize RAM areas where data access is to proceed and the RAM area +10 bytes when instructions are fetched from RAM areas, respectively. Reset signal generation sets RAM parity error resets to enabled (RPERDIS = 0). For details, see 21.3.2 RAM parity error detection function.

**Table 3-7. Extended SFR (2nd SFR) List (2/5)**

| Address | Special Function Register (SFR) Name | Symbol | | R/W | Manipulable Bit Range | | | After Reset |
|---|---|---|---|---|---|---|---|---|
| | | | | | 1-bit | 8-bit | 16-bit | |
| F00F0H | Peripheral enable register 0 | PER0 | | R/W | √ | √ | – | 00H |
| F00F3H | Operation speed mode control register | OSMC | | R/W | – | √ | – | 00H |
| F00F5H | RAM parity error control register | RPECTL | | R/W | √ | √ | – | 00H |
| F00FEH | BCD adjust result register | BCDADJ | | R | – | √ | – | Undefined |
| F0100H | Serial status register 00 | SSR00L | SSR00 | R | – | √ | √ | 0000H |
| F0101H | | – | | | – | – | | |
| F0102H | Serial status register 01 | SSR01L | SSR01 | R | – | √ | √ | 0000H |
| F0103H | | – | | | – | – | | |
| F0104H | Serial status register 02 | SSR02L | SSR02 | R | – | √ | √ | 0000H |
| F0105H | | – | | | – | – | | 0000H |
| F0106H | Serial status register 03 | SSR03L | SSR03 | R | – | √ | √ | 0000H |
| F0107H | | – | | | – | – | | 0000H |
| F0108H | Serial flag clear trigger register 00 | SIR00L | SIR00 | R/W | – | √ | √ | 0000H |
| F0109H | | – | | | – | – | | |
| F010AH | Serial flag clear trigger register 01 | SIR01L | SIR01 | R/W | – | √ | √ | 0000H |
| F010BH | | – | | | – | – | | |
| F010CH | Serial flag clear trigger register 02 | SIR02L | SIR02 | R/W | – | √ | √ | 0000H |
| F010DH | | – | | | – | – | | |
| F010EH | Serial flag clear trigger register 03 | SIR03L | SIR03 | R/W | – | √ | √ | 0000H |
| F010FH | | – | | | – | – | | |
| F0110H | Serial mode register 00 | SMR00 | | R/W | – | – | √ | 0020H |
| F0111H | | | | | | | | |
| F0112H | Serial mode register 01 | SMR01 | | R/W | – | – | √ | 0020H |
| F0113H | | | | | | | | |
| F0114H | Serial mode register 02 | SMR02 | | R/W | – | – | √ | 0020H |
| F0115H | | | | | | | | |
| F0116H | Serial mode register 03 | SMR03 | | R/W | – | – | √ | 0020H |
| F0117H | | | | | | | | |
| F0118H | Serial communication operation setting register 00 | SCR00 | | R/W | – | – | √ | 0087H |
| F0119H | | | | | | | | |
| F011AH | Serial communication operation setting register 01 | SCR01 | | R/W | – | – | √ | 0087H |
| F011BH | | | | | | | | |
| F011CH | Serial communication operation setting register 02 | SCR02 | | R/W | – | – | √ | 0087H |
| F011DH | | | | | | | | |
| F011EH | Serial communication operation setting register 03 | SCR03 | | R/W | – | – | √ | 0087H |
| F011FH | | | | | | | | |

### 6.8.3 Operation as frequency divider (channel 0 of 30-pin products only)

The timer array unit can be used as a frequency divider that divides a clock input to the TI00 pin and outputs the result from the TO00 pin.

The divided clock frequency output from TO00 can be calculated by the following expression.

- When rising edge/falling edge is selected:
  Divided clock frequency = Input clock frequency/{(Set value of TDR00 + 1) × 2}
- When both edges are selected:
  Divided clock frequency ≅ Input clock frequency/(Set value of TDR00 + 1)

Timer count register 00 (TCR00) operates as a down counter in the interval timer mode.

After the channel start trigger bit (TS00) of timer channel start register 0 (TS0) is set to 1, the TCR00 register loads the value of timer data register 00 (TDR00) when the TI00 valid edge is detected.

If the MD000 bit of timer mode register 00 (TMR00) is 0 at this time, INTTM00 is not output and TO00 is not toggled. If the MD000 bit of timer mode register 00 (TMR00) is 1, INTTM00 is output and TO00 is toggled.

After that, the TCR00 register counts down at the valid edge of the TI00 pin. When TCR00 = 0000H, it toggles TO00. At the same time, the TCR00 register loads the value of the TDR00 register again, and continues counting.

If detection of both the edges of the TI00 pin is selected, the duty factor error of the input clock affects the divided clock period of the TO00 output.

The period of the TO00 output clock includes a sampling error of one period of the operation clock.

Clock period of TO00 output = Ideal TO00 output clock period ± Operation clock period (error)

The TDR00 register can be rewritten at any time. The new value of the TDR00 register becomes valid during the next count period.

**Figure 6-46. Block Diagram of Operation as Frequency Divider**

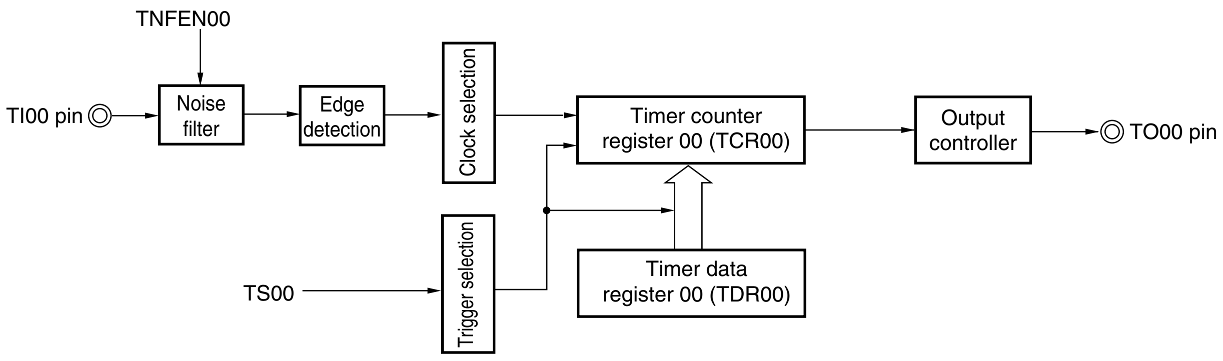
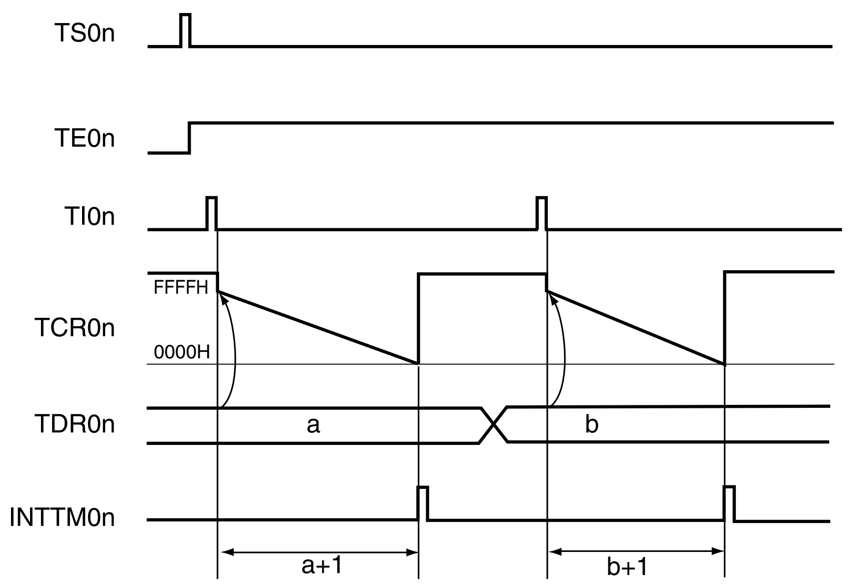


**Figure 6-59. Example of Basic Timing of Operation as Delay Counter**



**Remarks 1.** n: Channel number (n = 0 to 7)

**2.**

| | |
|---|---|
| TS0n: | Bit n of timer channel start register 0 (TS0) |
| TE0n: | Bit n of timer channel enable status register 0 (TE0) |
| TI0n: | TI0n pin input signal |
| TCR0n: | Timer count register 0n (TCR0n) |
| TDR0n: | Timer data register 0n (TDR0n) |

### 10.6.5 Hardware trigger no-wait mode (select mode, sequential conversion mode)

<1> In the stop status, the ADCE bit of A/D converter mode register 0 (ADM0) is set to 1, and the system enters the A/D conversion standby status.

<2> After the software counts up to the stabilization wait time (1 μs), the ADCS bit of the ADM0 register is set to 1 to place the system in the hardware trigger standby status (and conversion does not start at this stage). Note that, while in this status, A/D conversion does not start even if ADCS is set to 1.

<3> If a hardware trigger is input while ADCS = 1, A/D conversion is performed on the analog input specified by the analog input channel specification register (ADS).

<4> When A/D conversion ends, the conversion result is stored in the A/D conversion result register (ADCR, ADCRH), and the A/D conversion end interrupt request signal (INTAD) is generated. After A/D conversion ends, the next A/D conversion immediately starts.

<5> If a hardware trigger is input during conversion operation, the current A/D conversion is interrupted, and conversion restarts. The partially converted data is discarded.

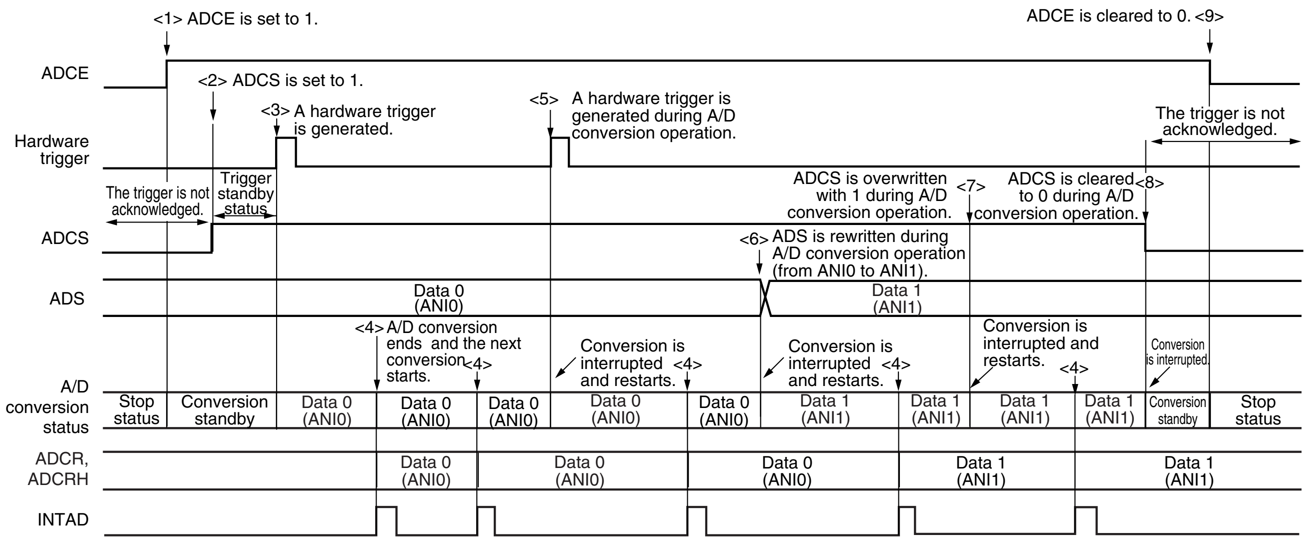
<6> When the value of the ADS register is rewritten or overwritten during conversion operation, the current A/D conversion is interrupted, and A/D conversion is performed on the analog input respecified by the ADS register. The partially converted data is discarded.

<7> When ADCS is overwritten with 1 during conversion operation, the current A/D conversion is interrupted, and conversion restarts. The partially converted data is discarded.

<8> When ADCS is cleared to 0 during conversion operation, the current A/D conversion is interrupted, and the system enters the A/D conversion standby status. However, the A/D converter does not power down in this status.

<9> When ADCE is cleared to 0 while in the A/D conversion standby status, the A/D converter enters the stop status. When ADCS = 0, inputting a hardware trigger is ignored and A/D conversion does not start.

**Figure 10-21. Example of Hardware Trigger No-Wait Mode (Select Mode, Sequential Conversion Mode) Operation Timing**



<1> ADCE is set to 1.
ADCE is cleared to 0. <9>
ADCE
<2> ADCS is set to 1.
<3> A hardware trigger is generated.
<5> A hardware trigger is generated during A/D conversion operation.
The trigger is not acknowledged.
Hardware trigger
The trigger is not acknowledged.
Trigger standby status
ADCS is overwritten with 1 during A/D conversion operation.
<7>
ADCS is cleared to 0 during A/D conversion operation.
<8>
ADCS
<6> ADS is rewritten during A/D conversion operation (from ANI0 to ANI1).
ADS
Data 0 (ANI0)
Data 1 (ANI1)
<4> A/D conversion ends and the next conversion starts.
<4>
Conversion is interrupted <4> and restarts.
Conversion is interrupted <4> and restarts.
Conversion is interrupted and restarts.
<4>
Conversion is interrupted.
A/D conversion status
Stop status
Conversion standby
Data 0 (ANI0)
Data 0 (ANI0)
Data 0 (ANI0)
Data 0 (ANI0)
Data 0 (ANI0)
Data 1 (ANI1)
Data 1 (ANI1)
Data 1 (ANI1)
Data 1 (ANI1)
Conversion standby
Stop status
ADCR, ADCRH
Data 0 (ANI0)
Data 0 (ANI0)
Data 0 (ANI0)
Data 1 (ANI1)
Data 1 (ANI1)
INTAD

### 10.6.9 Hardware trigger wait mode (select mode, sequential conversion mode)

<1> In the stop status, the ADCE bit of A/D converter mode register 0 (ADM0) is set to 1, and the system enters the hardware trigger standby status.

<2> If a hardware trigger is input while in the hardware trigger standby status, A/D conversion is performed on the analog input specified by the analog input channel specification register (ADS). The ADCS bit of the ADM0 register is automatically set to 1 according to the hardware trigger input.

<3> When A/D conversion ends, the conversion result is stored in the A/D conversion result register (ADCR, ADCRH), and the A/D conversion end interrupt request signal (INTAD) is generated. After A/D conversion ends, the next A/D conversion immediately starts. (At this time, no hardware trigger is necessary.)

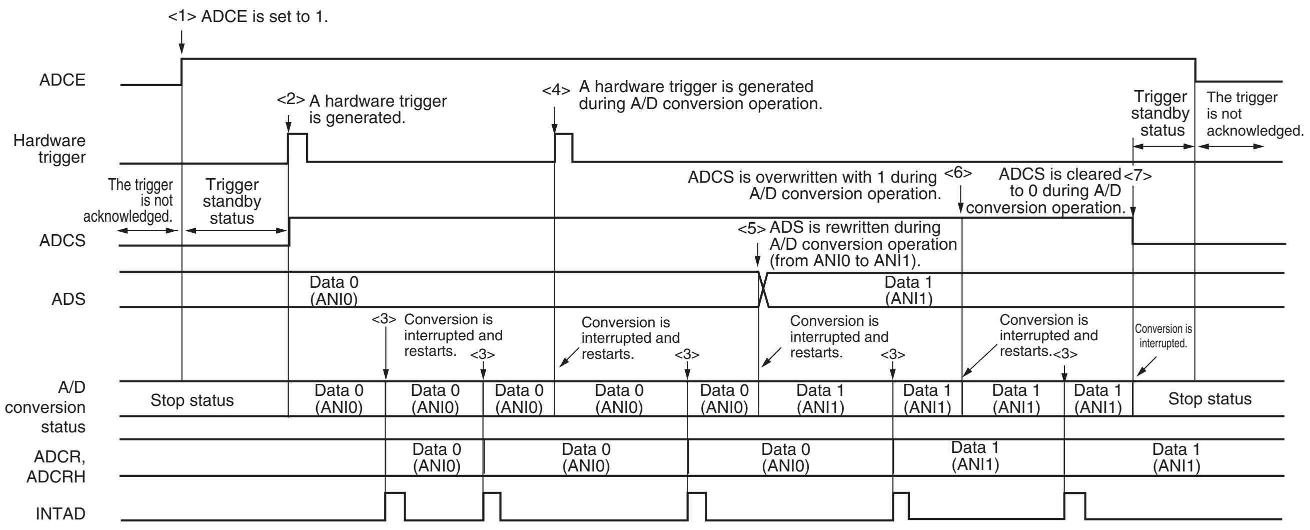
<4> If a hardware trigger is input during conversion operation, the current A/D conversion is interrupted, and conversion restarts. The partially converted data is discarded.

<5> When the value of the ADS register is rewritten or overwritten during conversion operation, the current A/D conversion is interrupted, and A/D conversion is performed on the analog input respecified by the ADS register. The partially converted data is discarded.

<6> When ADCS is overwritten with 1 during conversion operation, the current A/D conversion is interrupted, and conversion restarts. The partially converted data is discarded.

<7> When ADCS is cleared to 0 during conversion operation, the current A/D conversion is interrupted, the system enters the hardware trigger standby status, and the A/D converter enters the stop status. When ADCE = 0, inputting a hardware trigger is ignored and A/D conversion does not start.

**Figure 10-25. Example of Hardware Trigger Wait Mode (Select Mode, Sequential Conversion Mode) Operation Timing**


<1> ADCE is set to 1.
ADCE
Hardware trigger
ADCS
ADS
A/D conversion status
ADCR, ADCRH
INTAD
<2> A hardware trigger is generated.
<4> A hardware trigger is generated during A/D conversion operation.
Trigger standby status
The trigger is not acknowledged.
ADCS is overwritten with 1 during A/D conversion operation. <6>
ADCS is cleared to 0 during A/D conversion operation. <7>
<5> ADS is rewritten during A/D conversion operation (from ANI0 to ANI1).
Data 0 (ANI0)
Data 1 (ANI1)
<3> Conversion is interrupted and restarts.
Conversion is interrupted.
Stop status

## 10.8 SNOOZE mode function

In the SNOOZE mode, A/D conversion is triggered by inputting a hardware trigger in the STOP mode. Normally, A/D conversion is stopped while in the STOP mode, but, by using the SNOOZE mode function, A/D conversion can be performed without operating the CPU by inputting a hardware trigger. This is effective for reducing the operation current.

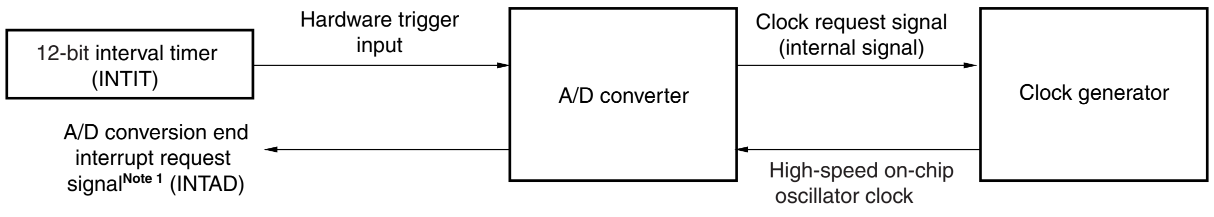
If the A/D conversion result range is specified using the ADUL and ADLL registers, A/D conversion results can be judged at a certain interval of time in SNOOZE mode. Using this function enables power supply voltage monitoring and input key judgment based on A/D inputs.

In the SNOOZE mode, only the following two conversion modes can be used:

- Hardware trigger wait mode (select mode, one-shot conversion mode)
- Hardware trigger wait mode (scan mode, one-shot conversion mode)

**Caution** SNOOZE mode can only be specified when the high-speed on-chip oscillator clock is selected for $f_{CLK}$.

**Figure 10-34. Block Diagram When Using SNOOZE Mode Function**


12-bit interval timer (INTIT)
Hardware trigger input
A/D converter
Clock request signal (internal signal)
Clock generator
A/D conversion end interrupt request signalNote 1 (INTAD)
High-speed on-chip oscillator clock

When using the SNOOZE mode function, the initial setting of each register is specified before switching to the STOP mode (for details about these settings, see **10.7.3 Setting up hardware trigger wait mode**Note 2). Just before switching to the STOP mode, bit 2 (AWC) of the A/D converter mode register 2 (ADM2) is set to 1. After the initial settings are specified, bit 0 (ADCE) of the A/D converter mode register 0 (ADM0) is set to 1.

If a hardware trigger is input after switching to the STOP mode, the high-speed on-chip oscillator clock is supplied to the A/D converter. After supplying this clock, the system automatically counts up to the A/D power supply stabilization wait time, and then A/D conversion starts.

The SNOOZE mode operation after A/D conversion ends differs depending on whether an interrupt signal is generatedNote 1.

**Notes 1.** Depending on the setting of the A/D conversion result comparison function (ADRCK bit, ADUL/ADLL register), there is a possibility of no interrupt signal being generated.

**2.** Be sure to set the ADM1 register to E2H or E3H.

**Remarks 1.** The hardware trigger is INTIT.

**2.** Specify the hardware trigger by using the A/D Converter Mode Register 1 (ADM1)

### 11.5.9 Procedure for processing errors that occurred during 3-wire serial I/O (CSI00, CSI01, CSI11, CSI20) communication

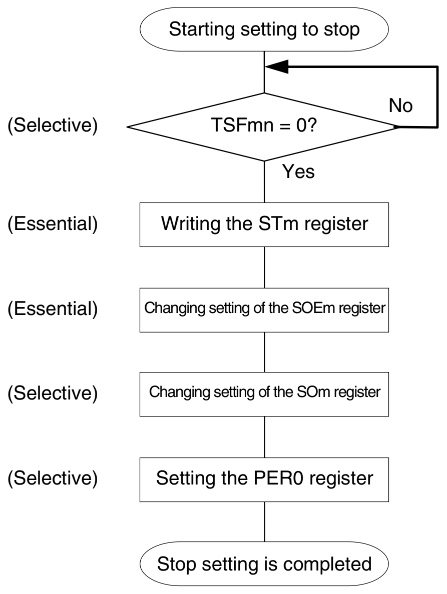
The procedure for processing errors that occurred during 3-wire serial I/O (CSI00, CSI01, CSI11, CSI20) communication is described in Figure 11-75.

**Figure 11-75. Processing Procedure in Case of Overrun Error**

| Software Manipulation | Hardware Status | Remark |
|---|---|---|
| Reads serial data register mn (SDRmn). → | The BFFmn bit of the SSRmn register is set to 0 and channel n is enabled to receive data. | This is to prevent an overrun error if the next reception is completed during error processing. |
| Reads serial status register mn (SSRmn). | | Error type is identified and the read value is used to clear error flag. |
| Writes 1 to serial flag clear trigger register mn (SIRmn). → | Error flag is cleared. | Error can be cleared only during reading, by writing the value read from the SSRmn register to the SIRmn register without modification. |

**Remark** m: Unit number (m = 0, 1), n: Channel number (n = 0 to 3), mn = 00, 01, 03, 10

**Figure 11-78. Procedure for Stopping UART Transmission**


Starting setting to stop
(Selective)
TSFmn = 0?
No
Yes
(Essential)
Writing the STm register
(Essential)
Changing setting of the SOEm register
(Selective)
Changing setting of the SOm register
(Selective)
Setting the PER0 register
Stop setting is completed

If there is any data being transferred, wait for their completion.
(If there is an urgent must stop, do not wait)

Write 1 to the STmn bit of the target channel.
(SEmn = 0 : to operation stop status)

Set the SOEmn bit to 0 and stop the output of the target channel.

The levels of the serial clock (CKOmn) and serial data (SOmn) on the target channel can be changed if necessitated by an emergency.

Reset the serial array unit by stopping the clock supply to it.

The master transmission is stopped.
Go to the next processing.

### 12.4.2 Setting transfer clock by using IICWL0 and IICWH0 registers

**(1) Setting transfer clock on master side**

$$\boxed{\text{Transfer clock} = \frac{f_{MCK}}{\text{IICWL} + \text{IICWH} + f_{MCK}\,(t_R + t_F)}}$$

At this time, the optimal setting values of the IICWL0 and IICWH0 registers are as follows.
(The fractional parts of all setting values are rounded up.)

• When the fast mode

$$\text{IICWL0} = \frac{0.52}{\text{Transfer clock}} \times f_{MCK}$$
$$\text{IICWH0} = \left(\frac{0.48}{\text{Transfer clock}} - t_R - t_F\right) \times f_{MCK}$$

• When the normal mode

$$\text{IICWL0} = \frac{0.47}{\text{Transfer clock}} \times f_{MCK}$$
$$\text{IICWH0} = \left(\frac{0.53}{\text{Transfer clock}} - t_R - t_F\right) \times f_{MCK}$$

**(2) Setting IICWL0 and IICWH0 registers on slave side**

(The fractional parts of all setting values are truncated.)

• When the fast mode

$\text{IICWL0} = 1.3\ \mu\text{s} \times f_{MCK}$

$\text{IICWH0} = (1.2\ \mu\text{s} - t_R - t_F) \times f_{MCK}$

• When the normal mode

$\text{IICWL0} = 4.7\ \mu\text{s} \times f_{MCK}$

$\text{IICWH0} = (5.3\ \mu\text{s} - t_R - t_F) \times f_{MCK}$

**Cautions 1.** The fastest operation frequency of IICA operation clock ($f_{MCK}$) is 20 MHz (max.). When only the $f_{CLK}$ exceeds 20 MHz, set bit 0 (PRSn) of the IICA control register n1 (IICCTLn1) to 1.

**2.** Note the minimum $f_{CLK}$ operation frequency when setting the transfer clock. The minimum $f_{CLK}$ operation frequency for serial interface IICA is determined according to the mode.

| | |
|---|---|
| Fast mode: | $f_{CLK}$ = 3.5 MHz (min.) |
| Normal mode: | $f_{CLK}$ = 1 MHz (min.) |

**Remarks 1.** Calculate the rise time ($t_R$) and fall time ($t_F$) of the SDAA0 and SCLA0 signals separately, because they differ depending on the pull-up resistance and wire load.

**2.**

| | |
|---|---|
| IICWL0: | IICA low-level width setting register 0 |
| IICWH0: | IICA high-level width setting register 0 |
| $t_F$: | SDAA0 and SCLA0 signal falling times |
| $t_R$: | SDAA0 and SCLA0 signal rising times |
| <R> $f_{MCK}$: | IICA operation clock frequency |

The meanings of <8> to <19> in (3) Data ~ data ~ stop condition in Figure 12-33 are explained below.

<8> The master device sets a wait status (SCLA0 = 0) at the falling edge of the 8th clock, and issues an interrupt (INTIICA0: end of transfer). Because of ACKE0 = 0 in the master device, the master device then sends an ACK by hardware to the slave device.
<9> The master device reads the received data and releases the wait status (WREL0 = 1).
<10> The ACK is detected by the slave device (ACKD0 = 1) at the rising edge of the 9th clock.
<11> The slave device set a wait status (SCLA0 = 0) at the falling edge of the 9th clock, and the slave device issue an interrupt (INTIICA0: end of transfer).
<12> By the slave device writing the data to transmit to the IICA register, the wait status set by the slave device is released. The slave device then starts transferring data to the master device.
<13> The master device issues an interrupt (INTIICA0: end of transfer) at the falling edge of the 8th clock, and sets a wait status (SCLA0 = 0). Because ACK control (ACKE0 = 1) is performed, the bus data line is at the low level (SDAA0 = 0) at this stage.
<14> The master device sets NACK as the response (ACKE0 = 0) and changes the timing at which it sets the wait status to the 9th clock (WTIM0 = 1).
<15> If the master device releases the wait status (WREL0 = 1), the slave device detects the NACK (ACK = 0) at the rising edge of the 9th clock.
<16> The master device and slave device set a wait status (SCLA0 = 0) at the falling edge of the 9th clock, and both the master device and slave device issue an interrupt (INTIICA0: end of transfer).
<17> When the master device issues a stop condition (SPT0 = 1), the bus data line is cleared (SDAA0 = 0) and the master device releases the wait status. The master device then waits until the bus clock line is set (SCLA0 = 1).
<18> The slave device acknowledges the NACK, halts transmission, and releases the wait status (WREL0 = 1) to end communication. Once the slave device releases the wait status, the bus clock line is set (SCLA0 = 1).
<19> Once the master device recognizes that the bus clock line is set (SCLA0 = 1) and after the stop condition setup time has elapsed, the master device sets the bus data line (SDAA0 = 1) and issues a stop condition (i.e. SCLA0 =1 changes SDAA0 from 0 to 1). When a stop condition is generated, the slave device detects the stop condition and issues an interrupt (INTIICA0: stop condition).

**Remark** <1> to <19> in Figure 12-33 show descriptions the entire procedure for communicating data using the I$^2$C bus.
Figure 12-33 (1) Start condition ~ address ~ data shows the processing from <1> to <7>, Figure 12-33 (2) Address ~ data ~ data shows the processing from <3> to <12>, and Figure 12-33 (3) Data ~ data ~ stop condition shows the processing from <8> to <19>.

**Figure 13-9. Timing Diagram of Multiply-Accumulation (signed) Operation**
**(2 × 3 + (−4) = 2 → 32767 × (−1) + (−2147483647) = −2147450882 (overflow occurs.))**

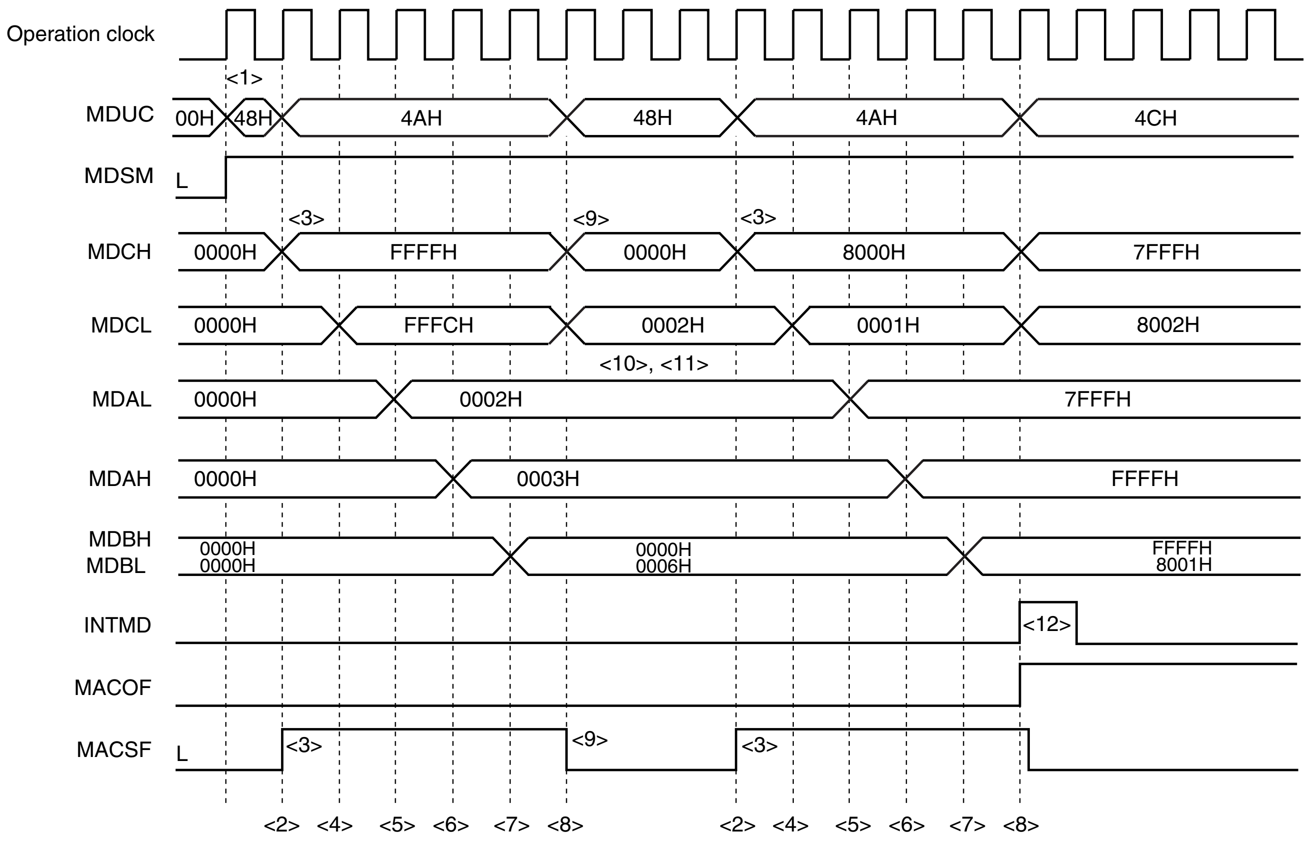


**Table 15-6. Relationship Between Interrupt Requests Enabled for Multiple Interrupt Servicing During Interrupt Servicing**

| Multiple Interrupt Request / Interrupt Being Serviced | | Maskable Interrupt Request | | | | | | | | Software Interrupt Request |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Priority Level 0 (PR = 00) | | Priority Level 1 (PR = 01) | | Priority Level 2 (PR = 10) | | Priority Level 3 (PR = 11) | | |
| | | IE = 1 | IE = 0 | IE = 1 | IE = 0 | IE = 1 | IE = 0 | IE = 1 | IE = 0 | |
| Maskable interrupt | ISP1 = 0 ISP0 = 0 | ○ | × | × | × | × | × | × | × | ○ |
| | ISP1 = 0 ISP0 = 1 | ○ | × | ○ | × | × | × | × | × | ○ |
| | ISP1 = 1 ISP0 = 0 | ○ | × | ○ | × | ○ | × | × | × | ○ |
| | ISP1 = 1 ISP0 = 1 | ○ | × | ○ | × | ○ | × | ○ | × | ○ |
| Software interrupt | | ○ | × | ○ | × | ○ | × | ○ | × | ○ |

**Remarks 1.** ○: Multiple interrupt servicing enabled

**2.** ×: Multiple interrupt servicing disabled

**3.** ISP0, ISP1, and IE are flags contained in the PSW.

ISP1 = 0, ISP0 = 0: An interrupt of level 1 or level 0 is being serviced.

ISP1 = 0, ISP0 = 1: An interrupt of level 2 is being serviced.

ISP1 = 1, ISP0 = 0: An interrupt of level 3 is being serviced.

ISP1 = 1, ISP0 = 1: Wait for an interrupt acknowledgment (all interrupts are enabled).

IE = 0: Interrupt request acknowledgment is disabled.

IE = 1: Interrupt request acknowledgment is enabled.

**4.** PR is a flag contained in the PR00L, PR00H, PR01L, PR10L, PR10H, PR11L registers.

PR = 00: Specify level 0 with ××PR1× = 0, ××PR0× = 0 (higher priority level)

PR = 01: Specify level 1 with ××PR1× = 0, ××PR0× = 1

PR = 10: Specify level 2 with ××PR1× = 1, ××PR0× = 0

PR = 11: Specify level 3 with ××PR1× = 1, ××PR0× = 1 (lower priority level)

### 16.4.2 When using the key interrupt flag (KRMD = 1)

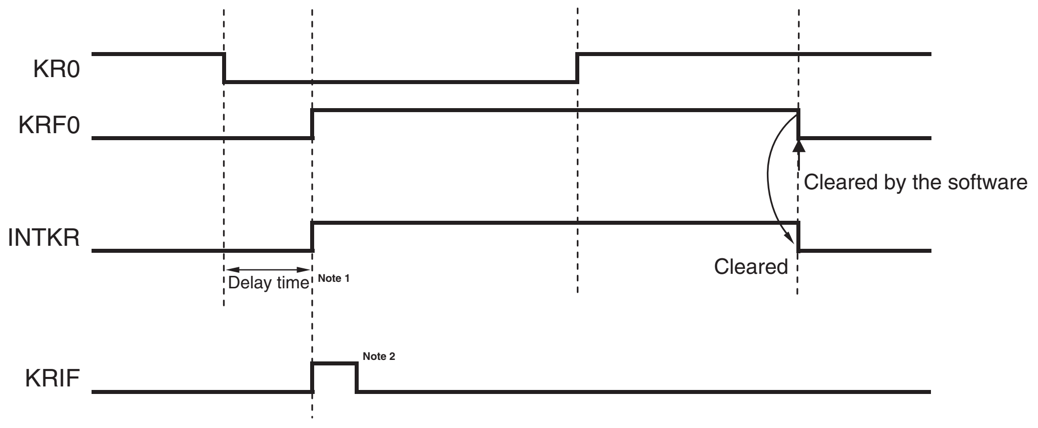
A key interrupt (INTKR) is generated when the valid edge specified by the setting of the KREG bit is input to a key interrupt pin (KR0 to KR5). The channels to which the valid edge was input can be identified by reading the key return flag register (KRF) after the key interrupt (INTKR) is generated.

If the KRMD bit is set to 1, the INTKR signal is cleared by clearing the corresponding bit in the KRF register.

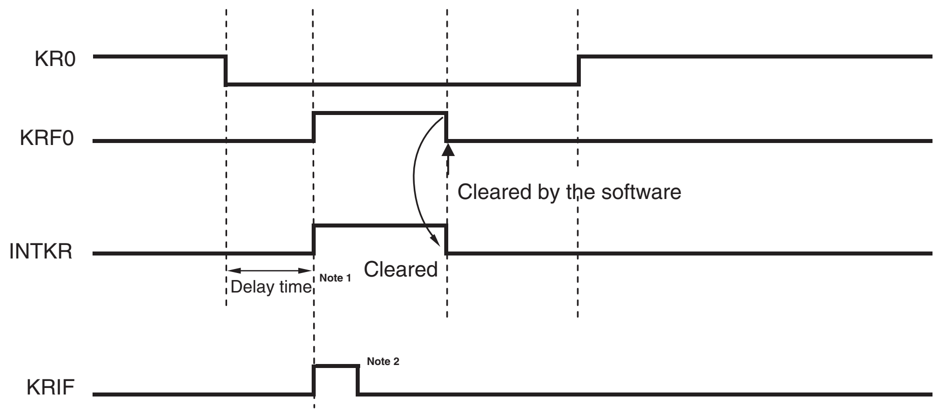
As shown in Figure 16-8, only one interrupt is generated each time a falling edge is input to one channel (when KREG = 0), regardless of whether the KRFn bit is cleared before or after a rising edge is input.

**Figure 16-8. Basic Operation of the INTKR Signal When the Key Interrupt Flag Is Used (When KRMD = 1 and KREG = 0)**

(a) KRF0 is cleared after a rising edge is input to the KR0 pin



(b) KRF0 is cleared before a rising edge is input to the KR0 pin



**Notes 1.** The maximum delay time is the maximum value of the high-level width and low-level width of the key interrupt input (see **28.4 AC Characteristics** or **29.4 AC Characteristics**).

**2.** Cleared by acknowledgment of vectored interrupt request or bit cleared by software.

# CHAPTER 17 STANDBY FUNCTION

## 17.1 Standby Function

The standby function reduces the operating current of the system, and the following three modes are available.

**(1) HALT mode**

HALT instruction execution sets the HALT mode. In the HALT mode, the CPU operation clock is stopped. If the high-speed system clock oscillator or high-speed on-chip oscillator is operating before the HALT mode is set, oscillation of each clock continues. In this mode, the operating current is not decreased as much as in the STOP mode, but the HALT mode is effective for restarting operation immediately upon interrupt request generation and carrying out intermittent operations frequently.

**(2) STOP mode**

STOP instruction execution sets the STOP mode. In the STOP mode, the high-speed system clock oscillator and high-speed on-chip oscillator stop, stopping the whole system, thereby considerably reducing the CPU operating current.

Because this mode can be cleared by an interrupt request, it enables intermittent operations to be carried out. However, because a wait time is required to secure the oscillation stabilization time after the STOP mode is released when the X1 clock is selected, select the HALT mode if it is necessary to start processing immediately upon interrupt request generation.

**(3) SNOOZE mode**

In the case of CSI00 or UART0 data reception and an A/D conversion request by the timer trigger signal (the interrupt request signal (INTIT)), the STOP mode is exited, the CSI00 or UART0 data is received without operating the CPU, and A/D conversion is performed. This can only be specified when the high-speed on-chip oscillator clock is selected for the CPU/peripheral hardware clock ($f_{CLK}$).

In either of these two modes, all the contents of registers, flags and data memory just before the standby mode is set are held. The I/O port output latches and output buffer statuses are also held.

**Cautions 1.** When shifting to the STOP mode, be sure to stop the peripheral hardware operation operating with main system clock before executing STOP instruction (other than SNOOZE mode setting unit).

**2.** When using CSI00, UART0, or the A/D converter in the SNOOZE mode, set up serial standby control register 0 (SSC0) and A/D converter mode register 2 (ADM2) before switching to the STOP mode. For details, see 11.3 Registers Controlling Serial Array Unit and 10.3 Registers Controlling A/D Converter.

**3.** To reduce the current consumption of the A/D converter when the standby function is used, first clear bit 7 (ADCS) and bit 0 (ADCE) of A/D converter mode register 0 (ADM0) to 0 to stop the A/D conversion, and then execute the STOP instruction.

**4.** It can be selected by the option byte whether the low-speed on-chip oscillator continues oscillating or stops in the HALT or STOP mode. For details, see CHAPTER 23 OPTION BYTE.

**Table 27-5. Operation List (2/17)**

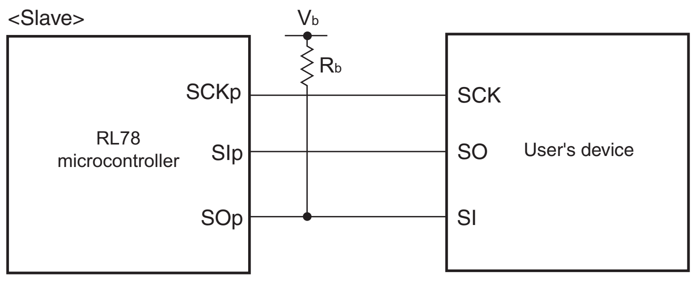
| Instruction Group | Mnemonic | Operands | Bytes | Clocks | | Clocks | Flag | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | **Note 1** | **Note 2** | | Z | AC | CY |
| 8-bit data transfer | MOV | A, sfr | 2 | 1 | – | A ← sfr | | | |
| | | sfr, A | 2 | 1 | – | sfr ← A | | | |
| | | A, [DE] | 1 | 1 | 4 | A ← (DE) | | | |
| | | [DE], A | 1 | 1 | – | (DE) ← A | | | |
| | | A, ES:[DE] | 2 | 2 | 5 | A ← (ES, DE) | | | |
| | | ES:[DE], A | 2 | 2 | – | (ES, DE) ← A | | | |
| | | A, [HL] | 1 | 1 | 4 | A ← (HL) | | | |
| | | [HL], A | 1 | 1 | – | (HL) ← A | | | |
| | | A, ES:[HL] | 2 | 2 | 5 | A ← (ES, HL) | | | |
| | | ES:[HL], A | 2 | 2 | – | (ES, HL) ← A | | | |
| | | A, [DE+byte] | 2 | 1 | 4 | A ← (DE + byte) | | | |
| | | [DE+byte], A | 2 | 1 | – | (DE + byte) ← A | | | |
| | | A, ES:[DE+byte] | 3 | 2 | 5 | A ← ((ES, DE) + byte) | | | |
| | | ES:[DE+byte], A | 3 | 2 | – | ((ES, DE) + byte) ← A | | | |
| | | A, [HL+byte] | 2 | 1 | 4 | A ← (HL + byte) | | | |
| | | [HL+byte], A | 2 | 1 | – | (HL + byte) ← A | | | |
| | | A, ES:[HL+byte] | 3 | 2 | 5 | A ← ((ES, HL) + byte) | | | |
| | | ES:[HL+byte], A | 3 | 2 | – | ((ES, HL) + byte) ← A | | | |
| | | A, [SP+byte] | 2 | 1 | – | A ← (SP + byte) | | | |
| | | [SP+byte], A | 2 | 1 | – | (SP + byte) ← A | | | |
| | | A, word[B] | 3 | 1 | 4 | A ← (B + word) | | | |
| | | word[B], A | 3 | 1 | – | (B + word) ← A | | | |
| | | A, ES:word[B] | 4 | 2 | 5 | A ← ((ES, B) + word) | | | |
| | | ES:word[B], A | 4 | 2 | – | ((ES, B) + word) ← A | | | |
| | | A, word[C] | 3 | 1 | 4 | A ← (C + word) | | | |
| | | word[C], A | 3 | 1 | – | (C + word) ← A | | | |
| | | A, ES:word[C] | 4 | 2 | 5 | A ← ((ES, C) + word) | | | |
| | | ES:word[C], A | 4 | 2 | – | ((ES, C) + word) ← A | | | |
| | | A, word[BC] | 3 | 1 | 4 | A ← (BC + word) | | | |
| | | word[BC], A | 3 | 1 | – | (BC + word) ← A | | | |
| | | A, ES:word[BC] | 4 | 2 | 5 | A ← ((ES, BC) + word) | | | |
| | | ES:word[BC], A | 4 | 2 | – | ((ES, BC) + word) ← A | | | |

**Notes 1.** Number of CPU clocks ($f_{CLK}$) when the internal RAM area, SFR area, or extended SFR area is accessed, or when no data is accessed.

**2.** Number of CPU clocks ($f_{CLK}$) when the code flash memory is accessed, or when the data flash memory is accessed by an 8-bit instruction.

**Remark** Number of clock is when program exists in the internal ROM (flash memory) area. If fetching the instruction from the internal RAM area, the number becomes double number plus 3 clocks at a maximum.

**CSI mode connection diagram (during communication at different potential)**

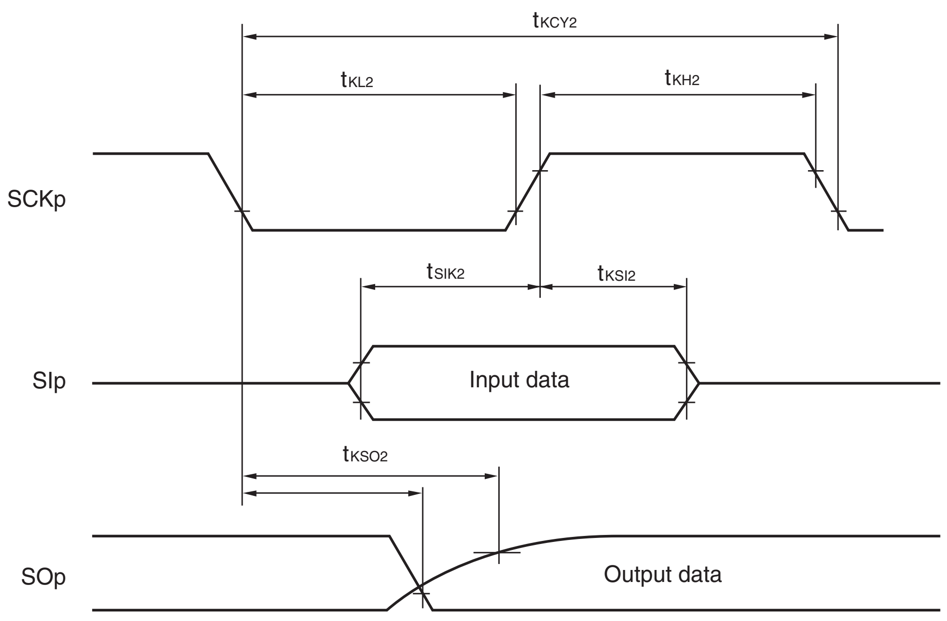


**Remarks 1.** $R_b$ [Ω]: Communication line (SOp) pull-up resistance, $C_b$ [F]: Communication line (SOp) load capacitance, $V_b$ [V]: Communication line voltage

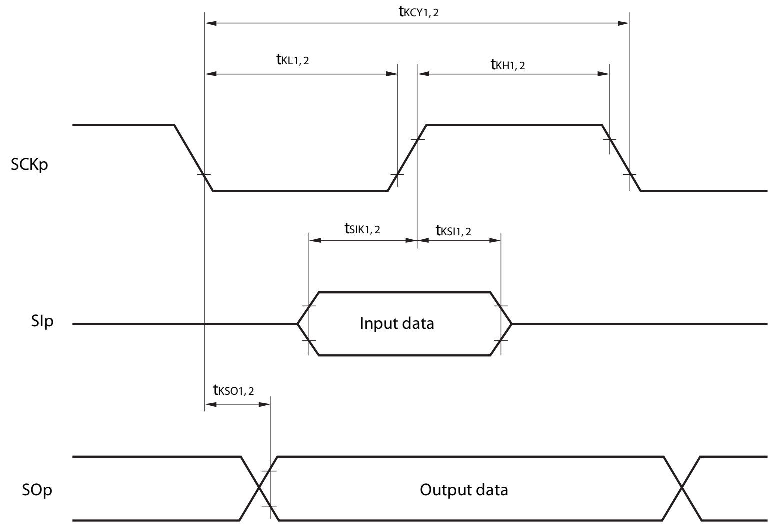
**2.** p: CSI number (p = 00, 20), m: Unit number (m = 0, 1), n: Channel number (n = 0)

**3.** $f_{MCK}$: Serial array unit operation clock frequency
(Operation clock to be set by the serial clock select register m (SPSm) and the CKSmn bit of serial mode register mn (SMRmn). m: Unit number, n: Channel number (mn = 00, 10))

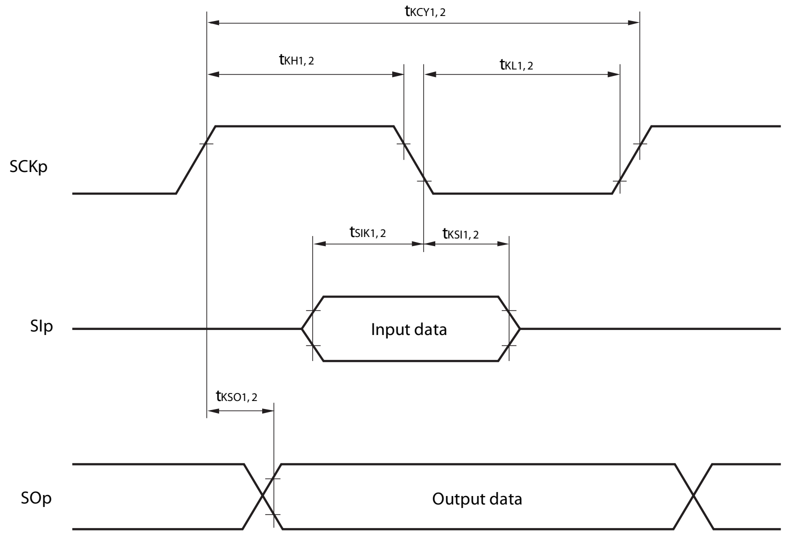
**CSI mode serial transfer timing (slave mode) (during communication at different potential)**
**(When DAPmn = 0 and CKPmn = 0, or DAPmn = 1 and CKPmn = 1.)**



**CSI mode serial transfer timing (during communication at same potential)**
**(When DAPmn = 0 and CKPmn = 0, or DAPmn = 1 and CKPmn = 1.)**



**CSI mode serial transfer timing (during communication at same potential)**
**(When DAPmn = 0 and CKPmn = 1, or DAPmn = 1 and CKPmn = 0.)**



**Remarks 1.** p: CSI number (p = 00, 01, 11, 20), m: Unit number (m = 0, 1), n: Channel number (n = 0, 1, 3)

**2.** $f_{MCK}$: Serial array unit operation clock frequency
(Operation clock to be set by the serial clock select register m (SPSm) and the CKSmn bit of serial mode register mn (SMRmn). m: Unit number (m = 0,1), n: Channel number (n = 0, 1, 3))

(4/17)

| Edition | Description | Chapter |
|---|---|---|
| 2.00 | Modification of Caution in 11.3.14 Serial standby control register 0 (SSC0) | CHAPTER 11 SERIAL ARRAY UNIT |
| | Modification of Figure 11-20 | |
| | Addition of Figure 11-21 | |
| | Modification of description in 11.3.15 Noise filter enable register 0 (NFEN0) | |
| | Modification of Caution in Figure 11-22 | |
| | Modification of description in 11.3.16 Registers controlling port functions of serial input/output pins | |
| | Modification of description of 11.5 Operation of 3-Wire Serial I/O (CSI00, CSI01, CSI11, CSI20) Communication and Note | |
| | Modification of description in 11.5.1 Master transmission and addition of Remark | |
| | Modification of Figures 11-28 to 11-32 | |
| | Modification of description in 11.5.2 Master reception and addition of Remark | |
| | Modification of Figures 11-35 to 11-37, 11-39, and 11-40 | |
| | Modification of description in 11.5.3 Master transmission/reception and addition of Remark | |
| | Modification of Figures 11-43, 11-45, 11-47, and 11-48 | |
| | Modification of description in 11.5.4 Slave transmission, Notes 1 and 2, and Remark 2 | |
| | Modification of Figures 11-49 and 11-51 to 11-56 | |
| | Modification of description in 11.5.5 Slave reception and Notes 1 and 2 | |
| | Modification of Figures 11-57 to 11-59, 11-61, and 11-62 | |
| | Modification of description in 11.5.6 Slave transmission/reception and Notes 1 and 2 | |
| | Modification of Figures 11-65 and 11-67 to 11-70 | |
| | Modification of description in 11.5.7 SNOOZE mode function | |
| | Modification of Figures 11-71 to 11-74 | |
| | Modification of description in 11.6 Operation of UART (UART0 to UART2) Communication and addition of Note 1 | |
| | Modification of description in 11.6.1 UART transmission and addition of Notes 1 and 2 | |
| | Modification of Figures 11-77 to 11-83 | |
| | Modification of description in 11.6.2 UART reception and addition of Notes 1 and 2 | |
| | Modification of Figure 11-86, 11-88, and 11-89 | |
| | Modification of description in 11.6.3 SNOOZE mode function and addition of Cautions 2 to 4 | |
| | Modification of Table 11-3 | |
| | Modification of description in 11. 6. 3 (1) and Figure 11-90 | |
| | Modification of description in 11. 6. 3 (2) and Figure 11-91 | |
| | Modification of Figure 11-92 | |
| | Modification of description in 11. 6. 3 (3) and Figure 11-93 | |
| | Modification of Figure 11-94 | |
| | Modification of description in 11.7 Operation of Simplified I2C (IIC00, IIC01, IIC11, IIC20) Communication | |
| | Modification of description in 11.7.1 Address field transmission and Notes 1 and 2 | |
| | Modification of description in 11.7.2 Data transmission and Notes 1 and 2 | |
| | Modification of description in 11.7.3 Data reception and Notes 1 and 2 | |
| | Modification of Figure 11-107 | |
| | Modification of Figure 11-109 | |
| | Modification of Figure 11-110 | |
| | Modification of Figure 12-1 | CHAPTER 12 SERIAL INTERFACE IICA |
| | Modification of Figure 12-5 | |
| | Modification of Figure 12-6 (3/4) and (4/4) | |
| | Modification of Figure 12-9 (2/2) | |
| | Modification of description in 12.3.6 IICA low-level width setting register 0 (IICWL0) | |
| | Modification of description in 12.3.7 IICA high-level width setting register 0 (IICWH0) | |
| | Modification of description in 12.4.2 Setting transfer clock by using IICWL0 and IICWH0 registers | |