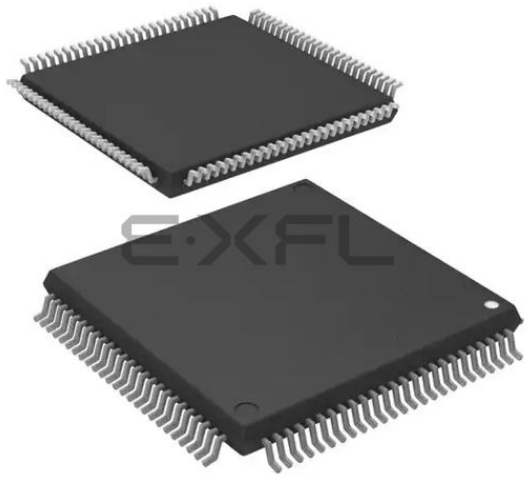


**Welcome to E-XFL.COM**

## 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 | RL78 |
| Core Size | 16-Bit |
| Speed | 24MHz |
| Connectivity | CSI, I²C, IrDA, LINbus, UART/USART |
| Peripherals | Brown-out Detect/Reset, LCD, LVD, POR, PWM, WDT |
| Number of I/O | 69 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.9V ~ 5.5V |
| Data Converters | A/D 6x10b, 4x24b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 100-LQFP |
| Supplier Device Package | 100-LFQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/renesas-electronics-america/r5f10mpgdfb-50 |

### 2.1.2 100-pin products

(1/3)

<R>

| Function Name | Pin Type | I/O | After Reset Release | Alternate Function | Function |
|---|---|---|---|---|---|
| P00 | 8-1-3 | I/O | Input port | RxD2/IrRxD/VCOUT0 | Port 0.<br>8-bit I/O port.<br>Input/output can be specified in 1-bit units.<br>Use of an on-chip pull-up resistor can be specified by a software setting at input port.<br>Input of P00, P03, P05, and P06 can be set to TTL input buffer.<br>Output of P01 to P07 can be set to N-ch open-drain output ($V_{DD}$ tolerance). |
| P01 | 7-1-4 | | | TxD2/IrTxD/VCOUT1 | |
| P02 | | | | SCL10/TI07/TO07/INTP5 | |
| P03 | 8-1-4 | | | RxD1/TI06/TO06/ SDA10/(VCOUT0) | |
| P04 | 7-1-4 | | | TxD1/TI05/TO05/INTP4/ (VCOUT1) | |
| P05 | 8-1-4 | | | SCK00/SCL00/TI04/ TO04/INTP3 | |
| P06 | | | | SI00/RxD0/TI03/TO03/ SDA00/TOOLRxD | |
| P07 | 7-1-4 | | | SO00/TxD0/TI02/TO02/ INTP2/TOOLTxD | |
| P10 | 7-5-4 | I/O | Digital input invalid[Note 1] | SEG4 | Port 1.<br>8-bit I/O port.<br>Input/output can be specified in 1-bit units.<br>Use of an on-chip pull-up resistor can be specified by a software setting at input port.<br>Input of P15 and P16 can be set to TTL input buffer.<br>Output of P15 to P17 can be set to N-ch open-drain output ($V_{DD}$ tolerance).<br>Can be set to LCD output[Note 2]. |
| P11 | | | | SEG5 | |
| P12 | | | | SEG6 | |
| P13 | | | | SEG7 | |
| P14 | | | | SEG8 | |
| P15 | 8-5-10 | | | SEG9/(SCK00)/(SCL00) | |
| P16 | | | | SEG10/(SI00)/(RxD0)/ (SDA00) | |
| P17 | 7-5-10 | | | SEG11/(SO00)/(TxD0) | |
| P20 | 4-3-3 | I/O | Analog input port | $AV_{REFP}$/ANI0 | Port 2.<br>6-bit I/O port.<br>Input/output can be specified in 1-bit units.<br>Can be set to analog input[Note 3]. |
| P21 | | | | $AV_{REFM}$/ANI1 | |
| P22 | 4-9-2 | | | ANI2/IVCMP0/IVREF1 | |
| P23 | | | | ANI3/IVCMP1/IVREF0 | |
| P24 | 4-3-3 | | | ANI4 | |
| P25 | | | | ANI5 | |
| P30 | 7-5-4 | I/O | Digital input invalid[Note 1] | SEG24/(TI07)/(TO07) | Port 3.<br>8-bit I/O port.<br>Input/output can be specified in 1-bit units.<br>Use of an on-chip pull-up resistor can be specified by a software setting at input port.<br>Can be set to LCD output[Note 2]. |
| P31 | | | | SEG25/(TI06)/(TO06) | |
| P32 | | | | SEG26/(PCLBUZ1) | |
| P33 | | | | SEG27/(PCLBUZ0) | |
| P34 | | | | SEG28 | |
| P35 | | | | SEG29 | |
| P36 | | | | SEG30 | |
| P37 | | | | SEG31 | |

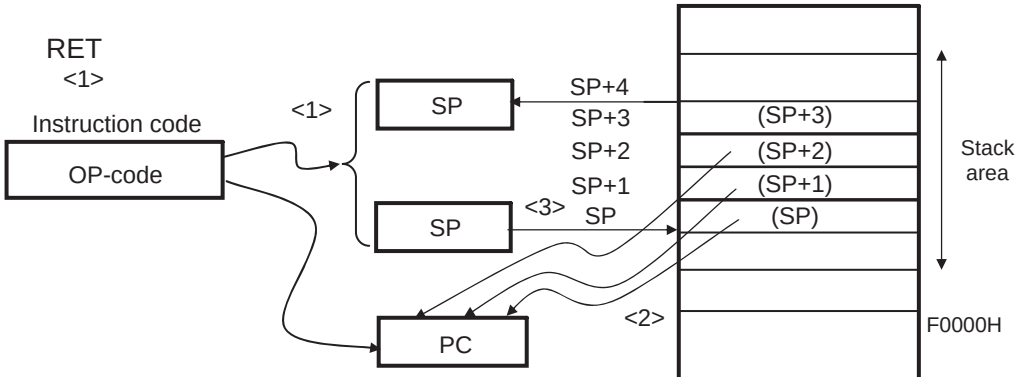
**Notes 1.** “Digital input invalid” refers to the state in which all the digital outputs, digital inputs, and LCD outputs are disabled.

**2.** Digital or LCD for each pin can be selected with the port mode register x (PMx) and the LCD port function register x (PFSEGx) (can be set in 1-bit unit).

**3.** Setting digital or analog to each pin can be done in A/D port configuration register (ADPC).

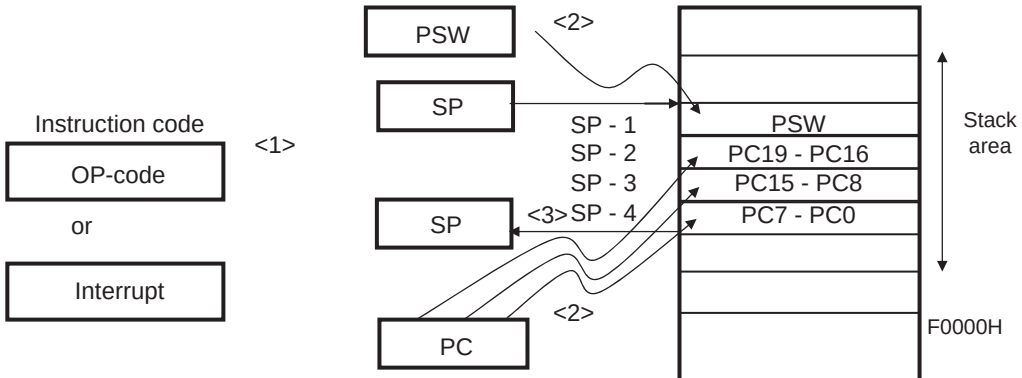
**Remark** Functions in parentheses in the above figure can be assigned via settings in the peripheral I/O redirection register (PIOR). See **Figure 4-8 Format of Peripheral I/O Redirection Register (PIOR)**.

**Figure 3-36. Example of RET**




- Stack addressing is specified <1>.
- The contents of addresses SP, SP + 1, and SP + 2 are stored in PC bits 7 to 0, 15 to 8, and 19 to 16, respectively <2>.
- The value of SP <3> is increased by four.

**Figure 3-37. Example of Interrupt, BRK**




- Stack addressing is specified <1>. In response to a BRK instruction or acceptance of an interrupt, the value of the program counter (PC) changes to indicate the address of the next instruction.
- The values of the PSW, PC bits 19 to 16, 15 to 8, and 7 to 0 are stored in addresses SP - 1, SP - 2, SP - 3, and SP - 4, respectively <2>.
- The value of the SP <3> is decreased by 4.

### 4.3.8 Peripheral I/O redirection register (PIOR)

This register is used to specify whether to enable or disable the peripheral I/O redirect function.

This function is used to switch ports to which alternate functions are assigned.

Use the PIOR register to assign a port to the function to redirect and enable the function.

In addition, can be changed the settings for redirection until its function enable operation.

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

Reset signal generation sets this register to 00H.

**Figure 4-8. Format of Peripheral I/O Redirection Register (PIOR)**

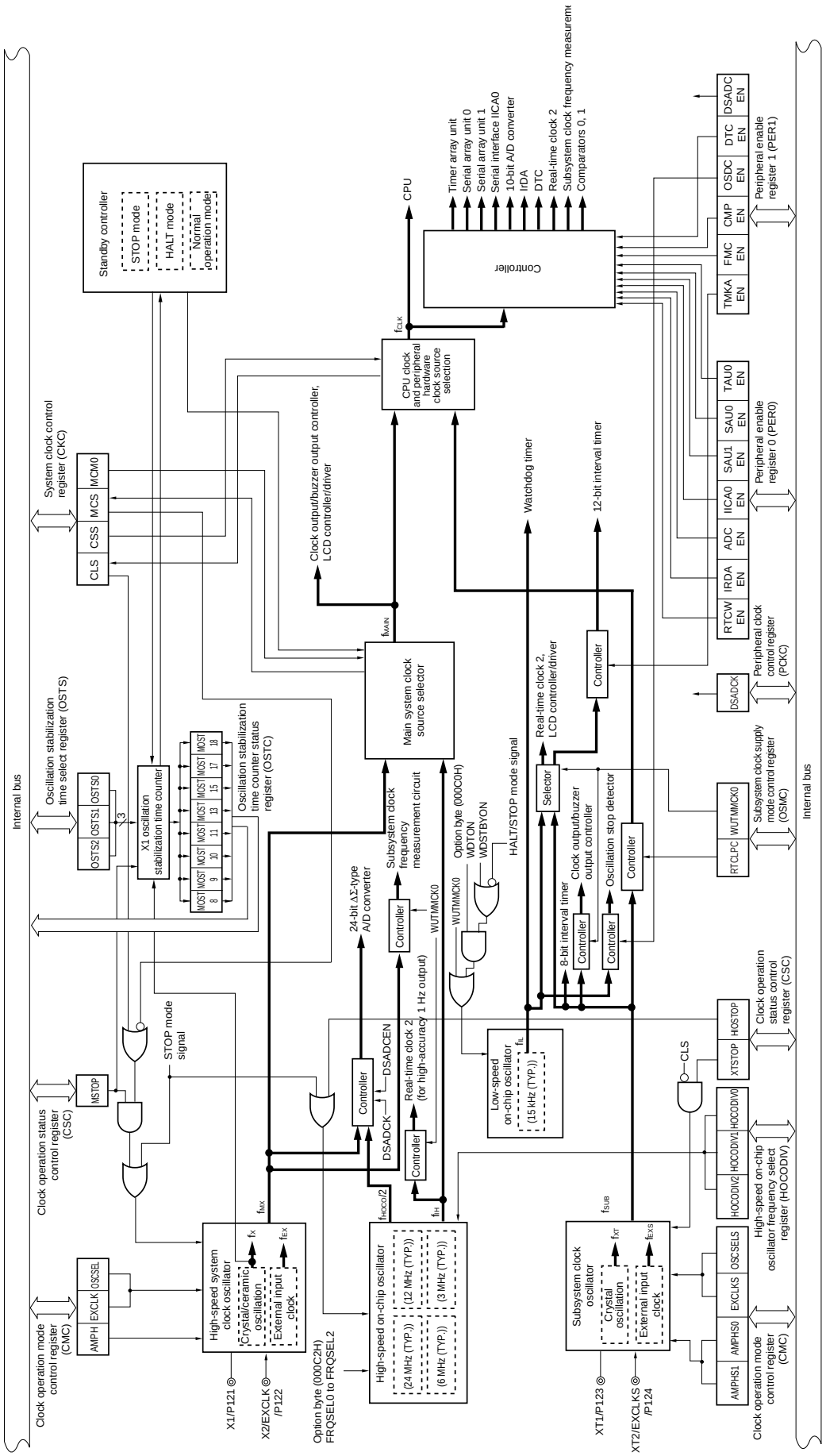
Address: F0077H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PIOR | 0 | 0 | 0 | PIOR4 | PIOR3 | PIOR2 | PIOR1 | PIOR0 |

| Bit | Function | 80-pin | | 100-pin | |
|---|---|---|---|---|---|
| | | Setting value | | Setting value | |
| | | 0 | 1 | 0 | 1 |
| PIOR4 | INTP0[Note] | P137 | P70 | P137 | P70 |
| | INTP1 | P125 | P71 | P125 | P71 |
| | INTP2 | P07 | P72 | P07 | P72 |
| | INTP3 | P05 | P73 | P05 | P73 |
| | INTP4 | P04 | P74 | P04 | P74 |
| | INTP5 | P02 | P75 | P02 | P75 |
| | INTP6 | P44 | P76 | P44 | P76 |
| | INTP7 | P42 | P77 | P42 | P77 |
| PIOR3 | PCLBUZ0 | P43 | P33 | P43 | P33 |
| | PCLBUZ1 | P41 | P32 | P41 | P32 |
| | VCOUT0 | P00 | P03 | P00 | P03 |
| | VCOUT1 | P01 | P04 | P01 | P04 |
| | RTC1HZ | P130 | P62 | P130 | P62 |
| PIOR2 | TxD1 | P04 | P82 | P04 | P82 |
| | RxD1 | P03 | P81 | P03 | P81 |
| | SCL10 | P02 | P80 | P02 | P80 |
| | SDA10 | P03 | P81 | P03 | P81 |
| PIOR1 | TxD0 | P07 | P17 | P07 | P17 |
| | RxD0 | P06 | P16 | P06 | P16 |
| | SCL00 | P05 | P15 | P05 | P15 |
| | SDA00 | P06 | P16 | P06 | P16 |
| | SI00 | P06 | P16 | P06 | P16 |
| | SO00 | P07 | P17 | P07 | P17 |
| | SCK00 | P05 | P15 | P05 | P15 |
| PIOR0 | TI00/TO00 | P43 | P60 | P43 | P60 |
| | TI01/TO01 | P41 | P61 | P41 | P61 |
| | TI02/TO02 | P07 | P62 | P07 | P62 |
| | TI03/TO03 | P06 | P127 | P06 | P127 |
| | TI04/TO04 | P05 | P126 | P05 | P126 |
| | TI05/TO05 | P04 | P125 | P04 | P125 |
| | TI06/TO06 | P03 | P31 | P03 | P31 |
| | TI07/TO07 | P02 | P30 | P02 | P30 |

**Note** Uses a battery backup function and the P137 pin is enabled when supplying power from VBAT.
When the INTP0 function is assigned to P70, note that the interrupt function is disabled when supplying power from VBAT.

**Figure 5-1. Block Diagram of Clock Generator**




**(Remark** is listed on the next page.)

**(1) X1 oscillation:**

As of March, 2014 (1/2)

| Manufacturer | Resonator | Part Number | SMD/ Lead | Frequency (MHz) | Flash operation mode[Note 1] | Recommended Circuit Constants[Note 2] (reference) | | | Oscillation Voltage Range (V) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | C1 (pF) | C2 (pF) | Rd (kΩ) | MIN. | MAX. |
| Murata Manufacturing Co., Ltd.[Note 3] | Ceramic resonator | CSTCC2M00G56-R0 | SMD | 2.0 | LS | (47) | (47) | 0 | 1.9 | 5.5 |
| | | CSTCR4M00G55-R0 | SMD | 4.0 | | (39) | (39) | 0 | | |
| | | CSTLS4M00G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCR4M19G55-R0 | SMD | 4.194 | | (39) | (39) | 0 | | |
| | | CSTLS4M19G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCR4M91G53-R0 | SMD | 4.915 | | (15) | (15) | 0 | | |
| | | CSTLS4M91G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCR5M00G53-R0 | SMD | 5.0 | | (15) | (15) | 0 | | |
| | | CSTLS5M00G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCR6M00G53-R0 | SMD | 6.0 | | (15) | (15) | 0 | | |
| | | CSTLS6M00G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCE8M00G52-R0 | SMD | 8.0 | | (10) | (10) | 0 | | |
| | | CSTLS8M00G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCE8M38G52-R0 | SMD | 8.388 | HS | (10) | (10) | 0 | 2.4 | 5.5 |
| | | CSTLS8M38G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCE10M0G52-R0 | SMD | 10.0 | | (10) | (10) | 0 | | |
| | | CSTLS10M0G53-B0 | Lead | | | (15) | (15) | 0 | | |
| | | CSTCE12M0G52-R0 | SMD | 12.0 | | (10) | (10) | 0 | | |
| | | CSTCE16M0V53-R0 | SMD | 16.0 | | (15) | (15) | 0 | | |
| | | CSTLS16M0X51-B0 | Lead | | | (5) | (5) | 0 | | |
| | | CSTCE20M0V51-R0 | SMD | 20.0 | | (5) | (5) | 0 | 2.7 | 5.5 |
| | | CSTLS20M0X51-B0 | Lead | | | (5) | (5) | 0 | | |

**Notes 1.** Set the flash operation mode by using CMODE1 and CMODE0 bits of the option byte (000C2H).

**2.** Values in parentheses in the C1 and C2 columns indicate an internal capacitance.

**3.** When using this resonator, for details about the matching, contact Murata Manufacturing Co., Ltd. (http://www.murata.com).

**Remark** Relationship between operation voltage width, operation frequency of CPU and operation mode is as below.

HS (high-speed main) mode: $2.7\ V \leq V_{DD} \leq 5.5\ V$@1 MHz to 24 MHz

$2.4\ V \leq V_{DD} \leq 5.5\ V$@1 MHz to 16 MHz

LS (low-speed main) mode: $1.9\ V \leq V_{DD} \leq 5.5\ V$@1 MHz to 8 MHz

### 7.3.12 Timer output mode register m (TOMm)

The TOMm register is used to control the timer output mode of each channel.

When a channel is used for the independent channel operation function, set the corresponding bit of the channel to be used to 0.

When a channel is used for the simultaneous channel operation function (PWM output, one-shot pulse output, or multiple PWM output), set the corresponding bit of the master channel to 0 and the corresponding bit of the slave channel to 1.

The setting of each channel n by this register is reflected at the timing when the timer output signal is set or reset while the timer output is enabled (TOEmn = 1).

The TOMm register can be set by a 16-bit memory manipulation instruction.

The lower 8 bits of the TOMm register can be set with an 8-bit memory manipulation instruction with TOMmL.

Reset signal generation clears this register to 0000H.

**Figure 7-21. Format of Timer Output Mode register m (TOMm)**

Address: F01BEH, F01BFH After reset: 0000H R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TOMm | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TOMm7 | TOMm6 | TOMm5 | TOMm4 | TOMm3 | TOMm2 | TOMm1 | 0 |

| TOMmn | Control of timer output mode of channel n |
|---|---|
| 0 | Master channel output mode (to produce toggle output by timer interrupt request signal (INTTMmn)) |
| 1 | Slave channel output mode (output is set by the timer interrupt request signal (INTTMmn) of the master channel, and reset by the timer interrupt request signal (INTTM0p) of the slave channel) |

**Caution Be sure to clear bits 15 to 8, and 0 to "0".**

**Remark** m: Unit number (m = 0)
n: Channel number
n = 0 to 7 (n = 0, 2, 4, 6 for master channel)
p: Slave channel number
$n < p \leq 7$
(For details of the relation between the master channel and slave channel, see **7.4.1 Basic rules of simultaneous channel operation function**.)

### 7.5.2 Start timing of counter

Timer count register mn (TCRmn) becomes enabled to operation by setting of TSmn bit of timer channel start register m (TSm).

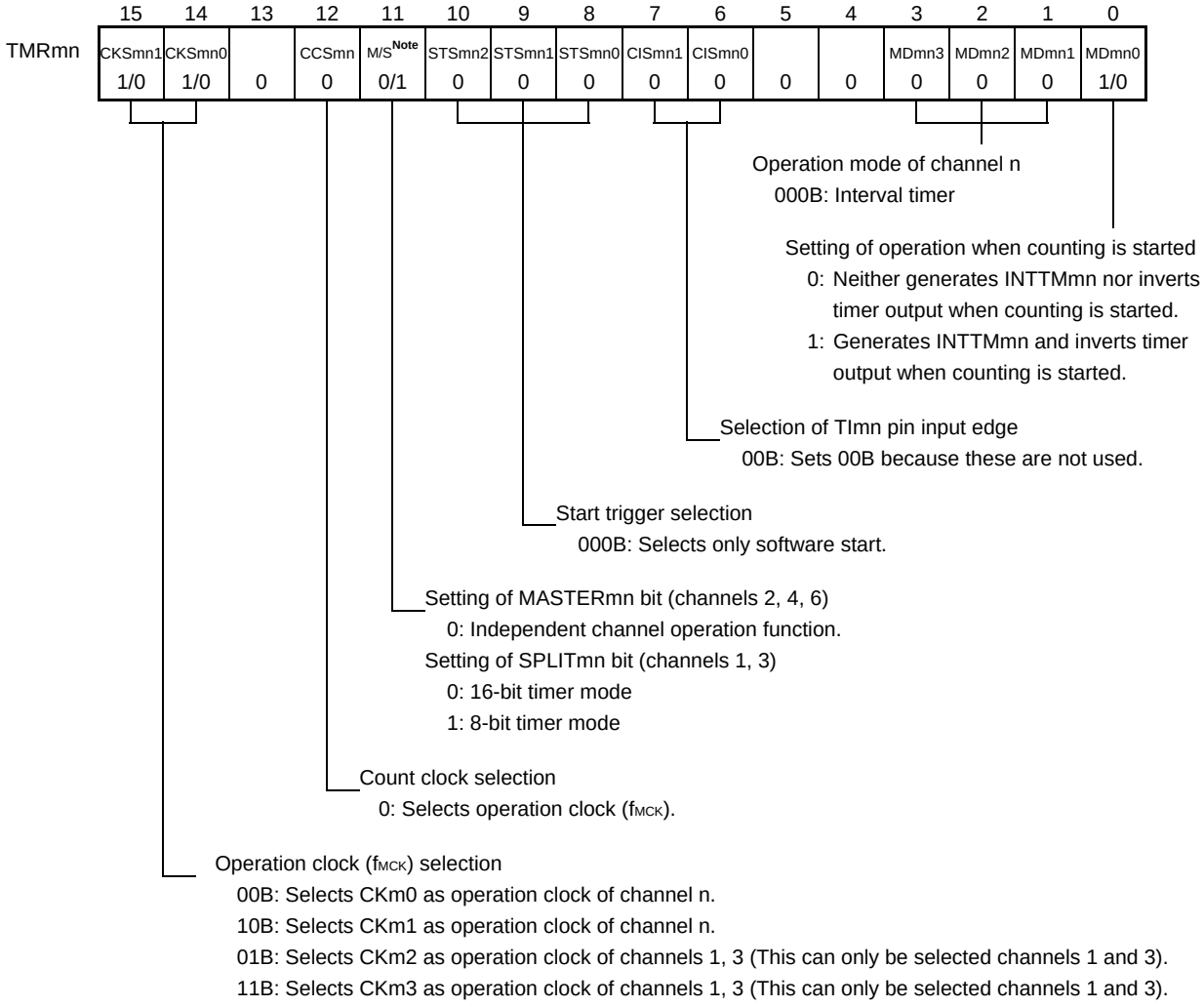
Operations from count operation enabled state to timer count Register mn (TCRmn) count start is shown in Table 7-5.

**Table 7-5. Operations from Count Operation Enabled State to Timer Count Register mn (TCRmn) Count Start**

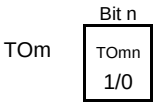
| Timer Operation Mode | Operation When TSmn = 1 Is Set |
|---|---|
| • Interval timer mode | No operation is carried out from start trigger detection (TSmn=1) until count clock generation.<br>The first count clock loads the value of the TDRmn register to the TCRmn register and the subsequent count clock performs count down operation (see **7.5.3 (1) Operation of interval timer mode**). |
| • Event counter mode | Writing 1 to the TSmn bit loads the value of the TDRmn register to the TCRmn register.<br>If detect edge of TImn input. The subsequent count clock performs count down operation (see **7.5.3 (2) Operation of event counter mode**). |
| • Capture mode | No operation is carried out from start trigger detection (TSmn = 1) until count clock generation.<br>The first count clock loads 0000H to the TCRmn register and the subsequent count clock performs count up operation (see **7.5.3 (3) Operation of capture mode (input pulse interval measurement)**). |
| • One-count mode | The waiting-for-start-trigger state is entered by writing 1 to the TSmn bit while the timer is stopped (TEmn = 0).<br>No operation is carried out from start trigger detection until count clock generation.<br>The first count clock loads the value of the TDRmn register to the TCRmn register and the subsequent count clock performs count down operation (see **7.5.3 (4) Operation of one-count mode**). |
| • Capture & one-count mode | The waiting-for-start-trigger state is entered by writing 1 to the TSmn bit while the timer is stopped (TEmn = 0).<br>No operation is carried out from start trigger detection until count clock generation.<br>The first count clock loads 0000H to the TCRmn register and the subsequent count clock performs count up operation (see **7.5.3 (5) Operation of capture & one-count mode (high-level width measurement)**). |

**Figure 7-44. Example of Set Contents of Registers During Operation as Interval Timer/Square Wave Output (1/2)**

**(a) Timer mode register mn (TMRmn)**

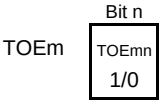



**(b) Timer output register m (TOm)**




0: Outputs 0 from TOmn.
1: Outputs 1 from TOmn.

**(c) Timer output enable register m (TOEm)**



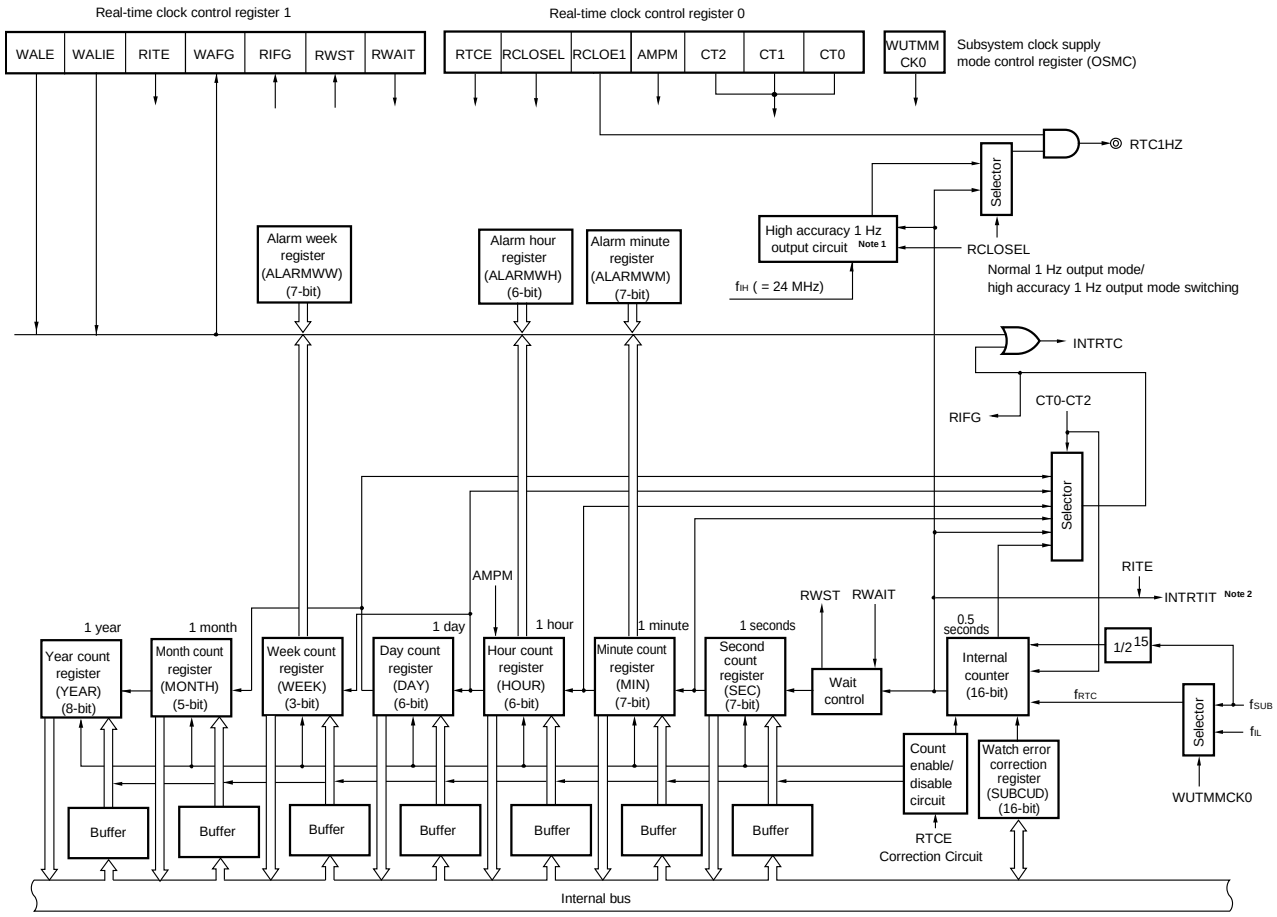

0: Stops the TOmn output operation by counting operation.
1: Enables the TOmn output operation by counting operation.

**Note** TMRm2, TMRm4, TMRm6: MASTERmn bit
TMRm1, TMRm3: SPLITmn bit
TMRm0, TMRm5, TMRm7: Fixed to 0

**Remark** m: Unit number (m = 0), n: Channel number (n = 0 to 7)

<R>

**Figure 8-1. Real-time Clock 2 Diagram**




**Notes 1.** A high-speed on-chip oscillator (HOCO: 24 MHz) can be used for high precision 1 Hz output. HOCO must be set to ON in order to run in high precision 1 Hz output mode. To run in normal 1 Hz mode, there is no need to set HOCO to ON.

**2.** An interrupt that indicates the timing to get the correction value from the clock error correction register (SUBCUD).

The fetch timing is 1 second ($f_{SUB}$ base) interval.

Figure 18-2 shows the block diagram of the serial array unit 1.

<R>

**Figure 18-2. Block Diagram of Serial Array Unit 1**

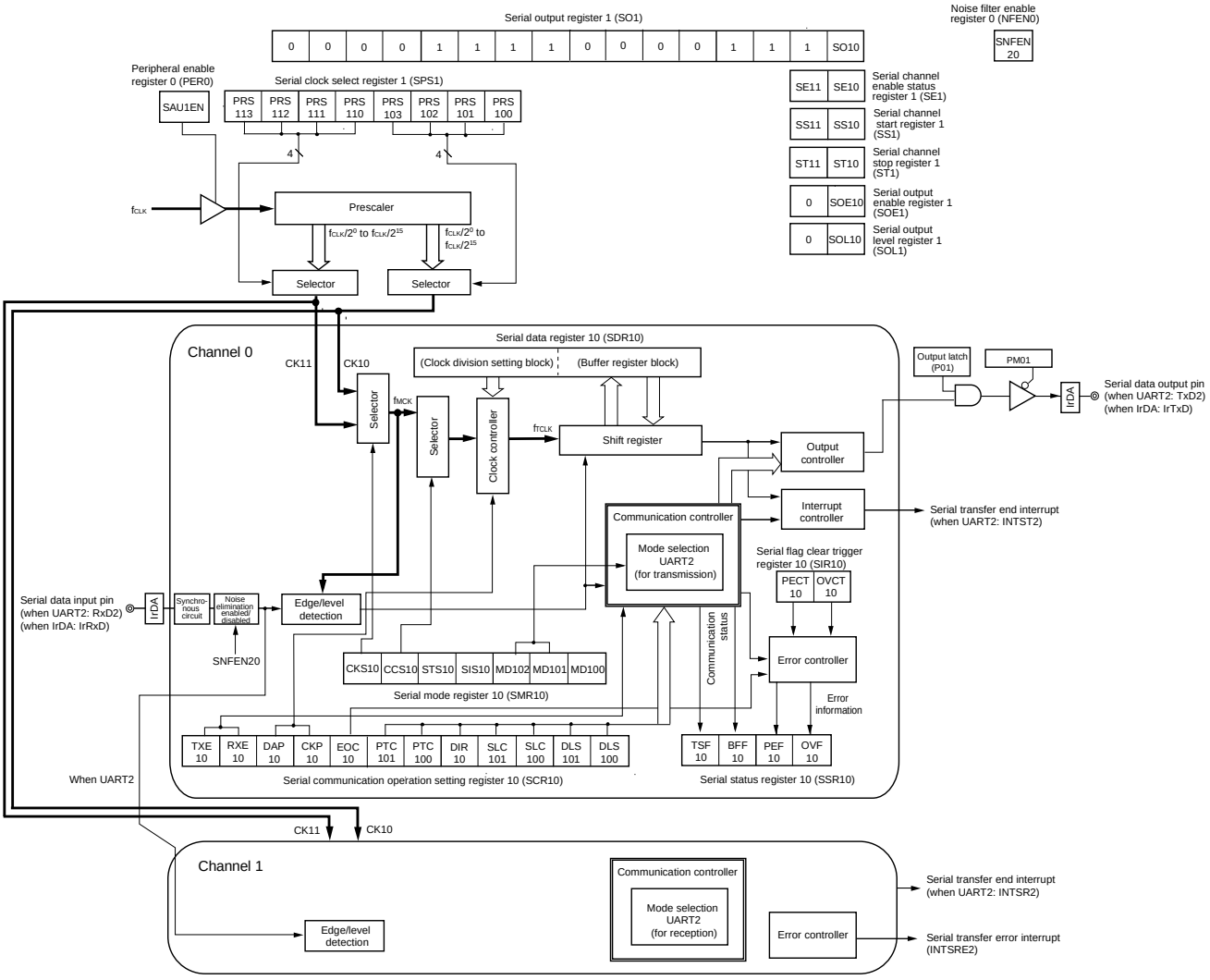
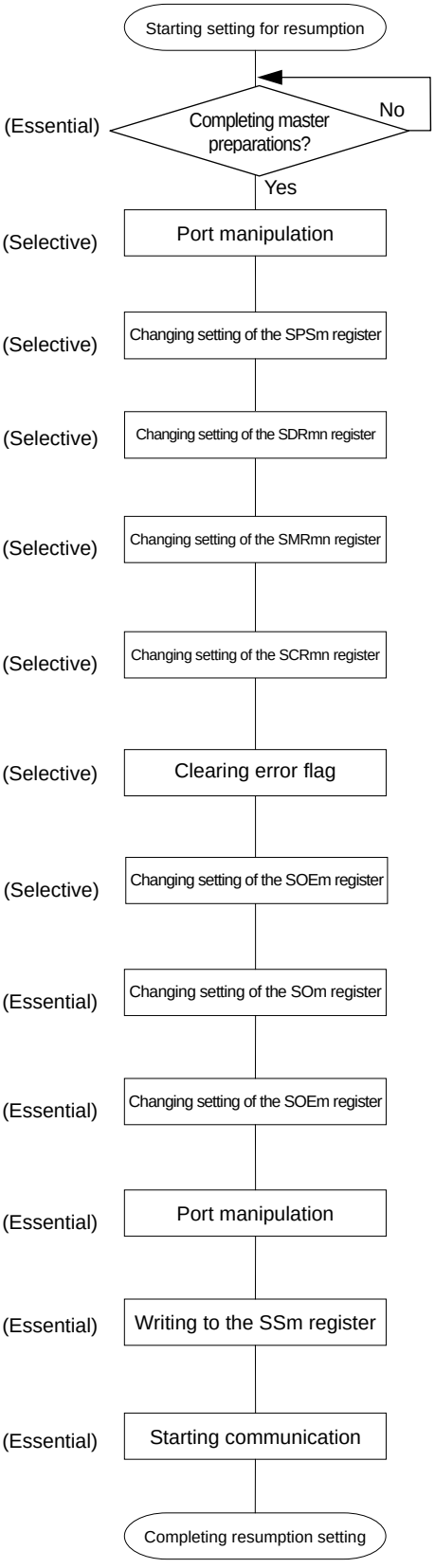


**Figure 18-52. Procedure for Resuming Slave Transmission**

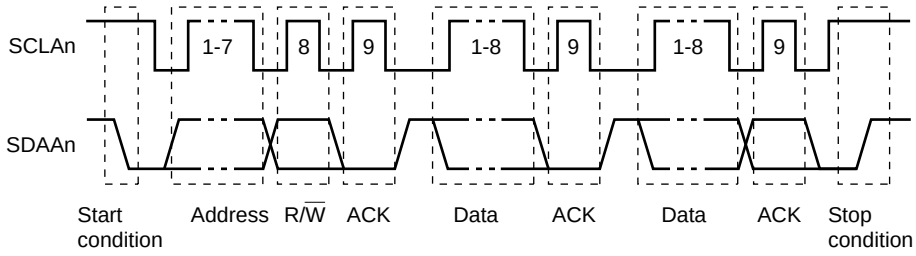



**Remark** If PER0 is rewritten while stopping the master transmission and the clock supply is stopped, wait until the transmission target (master) stops or transmission finishes, and then perform initialization instead of restarting the transmission.

## 19.5 $I^2C$ Bus Definitions and Control Methods

The following section describes the $I^2C$ bus's serial data communication format and the signals used by the $I^2C$ bus. Figure 19-14 shows the transfer timing for the "start condition", "address", "data", and "stop condition" output via the $I^2C$ bus's serial data bus.

**Figure 19-14. $I^2C$ Bus Serial Data Transfer Timing**



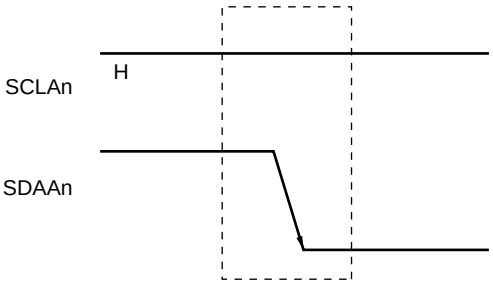

The master device generates the start condition, slave address, and stop condition.
The acknowledge (ACK) can be generated by either the master or slave device (normally, it is output by the device that receives 8-bit data).

The serial clock (SCLAn) is continuously output by the master device. However, in the slave device, the SCLAn pin low level period can be extended and a wait can be inserted.

### 19.5.1 Start conditions

A start condition is met when the SCLAn pin is at high level and the SDAAn pin changes from high level to low level. The start conditions for the SCLAn pin and SDAAn pin are signals that the master device generates to the slave device when starting a serial transfer. When the device is used as a slave, start conditions can be detected.

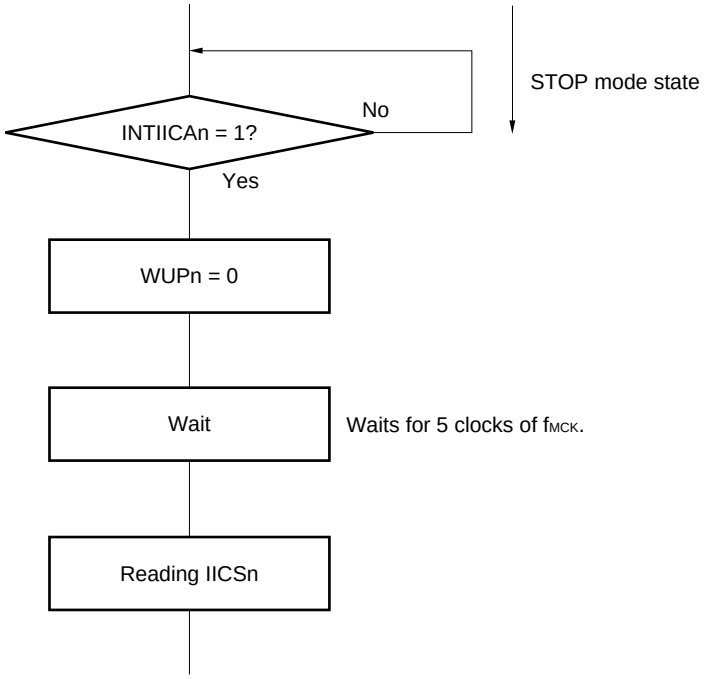
**Figure 19-15. Start Conditions**




A start condition is output when bit 1 (STTn) of IICA control register n0 (IICCTLn0) is set (1) after a stop condition has been detected (SPDn: Bit 0 of the IICA status register n (IICSn) = 1). When a start condition is detected, bit 1 (STDn) of the IICSn register is set (1).

**Remark** n = 0

**Figure 19-23. Flow When Setting WUPn = 0 upon Address Match (Including Extension Code Reception)**




Use the following flows to perform the processing to release the STOP mode other than by an interrupt request (INTIICAn) generated from serial interface IICA.

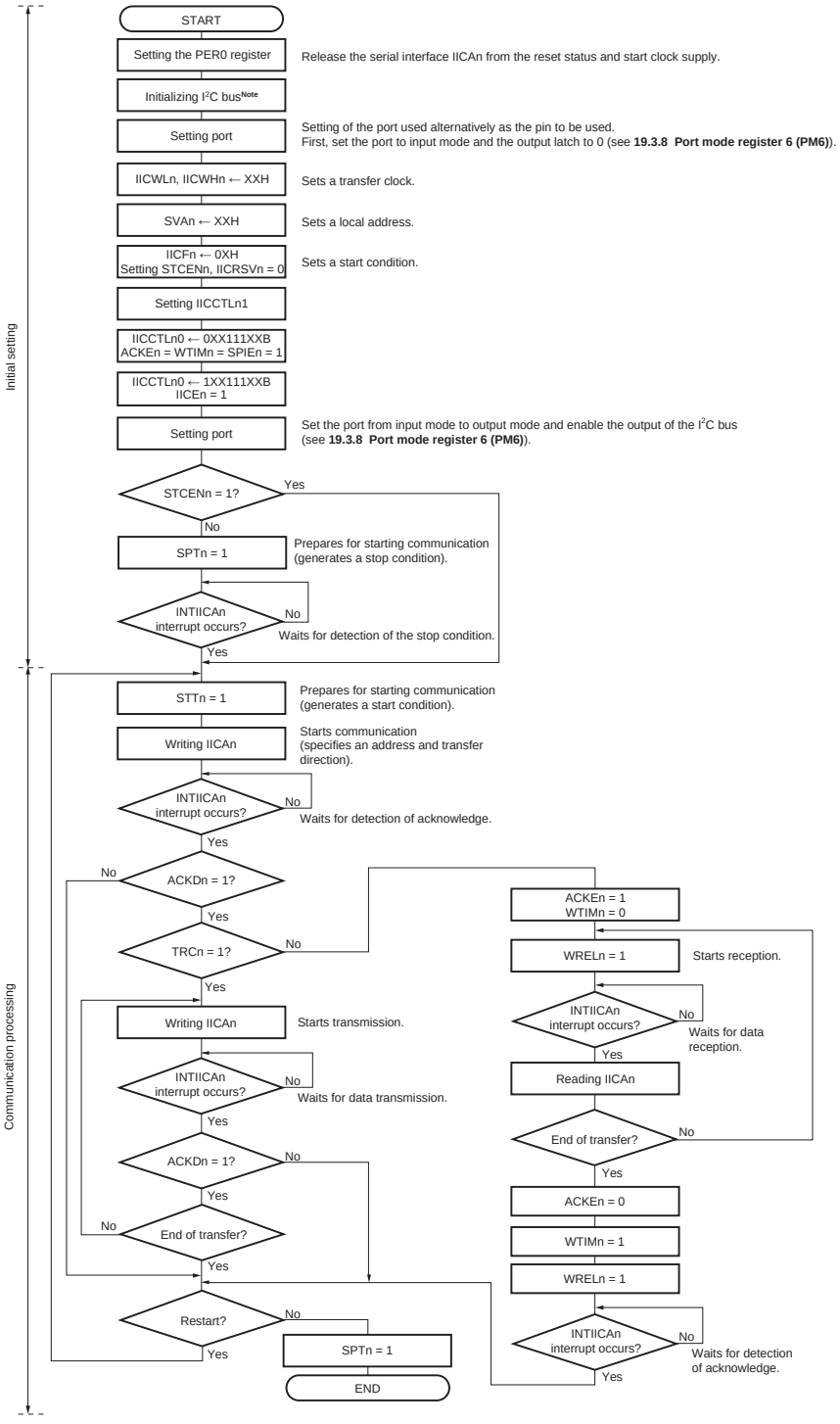
- When operating next IIC communication as master: Flow shown in Figure 19-24
- When operating next IIC communication as slave:
  When restored by INTIICAn interrupt: Same as the flow in Figure 19-23
  When restored by other than INTIICAn interrupt: Until the INTIICAn interrupt occurs, continue operating with WUPn left set to 1

**Remark** n = 0

### (1) Master operation in single-master system

<R>

**Figure 19-28. Master Operation in Single-Master System**



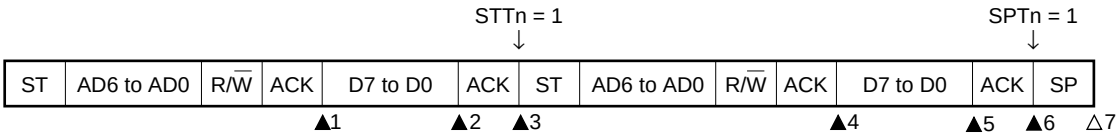

**Note** Release (SCLAn and SDAAn pins = high level) the I²C bus in conformance with the specifications of the product that is communicating. If EEPROM is outputting a low level to the SDAAn pin, for example, set the SCLAn pin in the output port mode, and output a clock pulse from the output port until the SDAAn pin is constantly at high level.

**Remarks 1.** Conform to the specifications of the product that is communicating, with respect to the transmission and reception formats.

**2.** n = 0

**(b) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop (restart)**

**(i) When WTIMn = 0**

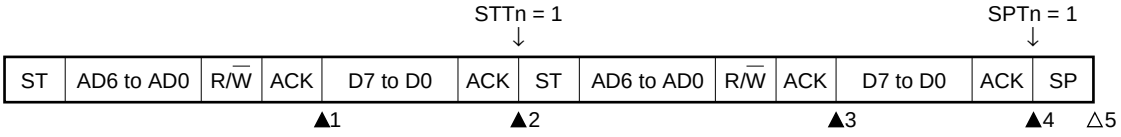



▲1: IICSn = 1000×110B
▲2: IICSn = 1000×000B (Sets the WTIMn bit to 1)[Note 1]
▲3: IICSn = 1000××00B (Clears the WTIMn bit to 0[Note 2], sets the STTn bit to 1)
▲4: IICSn = 1000×110B
▲5: IICSn = 1000×000B (Sets the WTIMn bit to 1)[Note 3]
▲6: IICSn = 1000××00B (Sets the SPTn bit to 1)
△7: IICSn = 00000001B

**Notes 1.** To generate a start condition, set the WTIMn bit to 1 and change the timing for generating the INTIICAn interrupt request signal.
**2.** Clear the WTIMn bit to 0 to restore the original setting.
**3.** To generate a stop condition, set the WTIMn bit to 1 and change the timing for generating the INTIICAn interrupt request signal.

**Remark** ▲: Always generated
△: Generated only when SPIEn = 1
×: Don't care

**(ii) When WTIMn = 1**

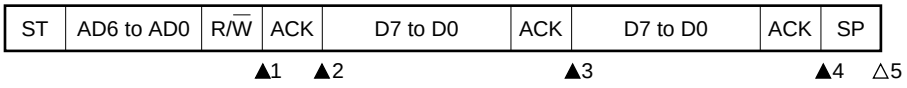



▲1: IICSn = 1000×110B
▲2: IICSn = 1000××00B (Sets the STTn bit to 1)
▲3: IICSn = 1000×110B
▲4: IICSn = 1000××00B (Sets the SPTn bit to 1)
△5: IICSn = 00000001B

**Remark** ▲: Always generated
△: Generated only when SPIEn = 1
×: Don't care

**Remark** n = 0

**(ii) When WTIMn = 1**



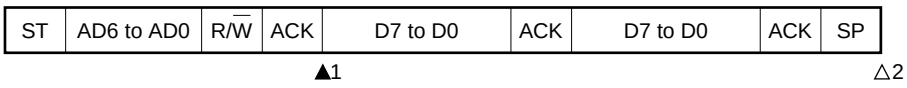

▲1: IICSn = 0110×010B
▲2: IICSn = 0010×110B
▲3: IICSn = 0010×100B
▲4: IICSn = 0010××00B
△5: IICSn = 00000001B

**Remark** ▲: Always generated
△: Generated only when SPIEn = 1
×: Don't care

**(6) Operation when arbitration loss occurs (no communication after arbitration loss)**

When the device is used as a master in a multi-master system, read the MSTSn bit each time interrupt request signal INTIICAn has occurred to check the arbitration result.

**(a) When arbitration loss occurs during transmission of slave address data (when WTIMn = 1)**




▲1: IICSn = 01000110B
△2: IICSn = 00000001B

**Remark** ▲: Always generated
△: Generated only when SPIEn = 1

**Remark** n = 0

The meanings of <7> to <15> in (3) Data ~ data ~ stop condition in Figure 19-32 are explained below.

<7> After data transfer is completed, because of ACKEn = 1, the slave device sends an ACK by hardware to the master device. The ACK is detected by the master device (ACKDn = 1) at the rising edge of the 9th clock.

<8> The master device and slave device set a wait status (SCLAn = 0) at the falling edge of the 9th clock, and both the master device and slave device issue an interrupt (INTIICAn: end of transfer).

<9> The master device writes the data to transmit to the IICA shift register n (IICAn) and releases the wait status that it set by the master device.

<10> The slave device reads the received data and releases the wait status (WRELn = 1). The master device then starts transferring data to the slave device.

<11> When data transfer is complete, the slave device (ACKEn =1) sends an ACK by hardware to the master device. The ACK is detected by the master device (ACKDn = 1) at the rising edge of the 9th clock.

<12> The master device and slave device set a wait status (SCLAn = 0) at the falling edge of the 9th clock, and both the master device and slave device issue an interrupt (INTIICAn: end of transfer).

<13> The slave device reads the received data and releases the wait status (WRELn = 1).

<14> By the master device setting a stop condition trigger (SPTn = 1), the bus data line is cleared (SDAAn = 0) and the bus clock line is set (SCLAn = 1). After the stop condition setup time has elapsed, by setting the bus data line (SDAAn = 1), the stop condition is then generated (i.e. SCLAn =1 changes SDAAn from 0 to 1).

<15> When a stop condition is generated, the slave device detects the stop condition and issues an interrupt (INTIICAn: stop condition).

**Remarks 1.** <1> to <15> in Figure 19-32 represent the entire procedure for communicating data using the $I^2C$ bus.

Figure 19-32 (1) Start condition ~ address ~ data shows the processing from <1> to <6>, Figure 19-32 (2) Address ~ data ~ data shows the processing from <3> to <10>, and Figure 19-32 (3) Data ~ data ~ stop condition shows the processing from <7> to <15>.

**2.** n = 0

# CHAPTER 23 INTERRUPT FUNCTIONS

The interrupt function switches the program execution to other processing. When the branch processing is finished, the program returns to the interrupted processing.

| | | Number of interrupt sources |
|---|---|---|
| Maskable interrupts | External | 10 |
| | Internal | 33 |

## 23.1 Interrupt Function Types

The following two types of interrupt functions are used.

**(1) Maskable interrupts**

These interrupts undergo mask control. Maskable interrupts can be divided into four priority groups by setting the priority specification flag registers (PR00L, PR00H, PR01L, PR01H, PR02L, PR02H, PR03L, PR10L, PR10H, PR11L, PR11H, PR12L, PR12H, PR13L).

Multiple interrupt servicing can be applied to low-priority interrupts when high-priority interrupts are generated. If two or more interrupt requests, each having the same priority, are simultaneously generated, then they are processed according to the default priority of vectored interrupt servicing. Default priority, see **Table 23-1**.

A standby release signal is generated and STOP, HALT, and SNOOZE modes are released.

External interrupt requests and internal interrupt requests are provided as maskable interrupts.

**(2) Software interrupts**

This is a vectored interrupt generated by executing the BRK instruction. It is acknowledged even when interrupts are disabled. The software interrupt does not undergo interrupt priority control.

## 23.2 Interrupt Sources and Configuration

Interrupt sources include maskable interrupts and software interrupts. In addition, they also have up to seven reset sources (see **Table 23-1**). The vector codes that store the program start address when branching due to the generation of a reset or various interrupt requests are two bytes each, so interrupts jump to a 64 K address of 00000H to 0FFFFH.

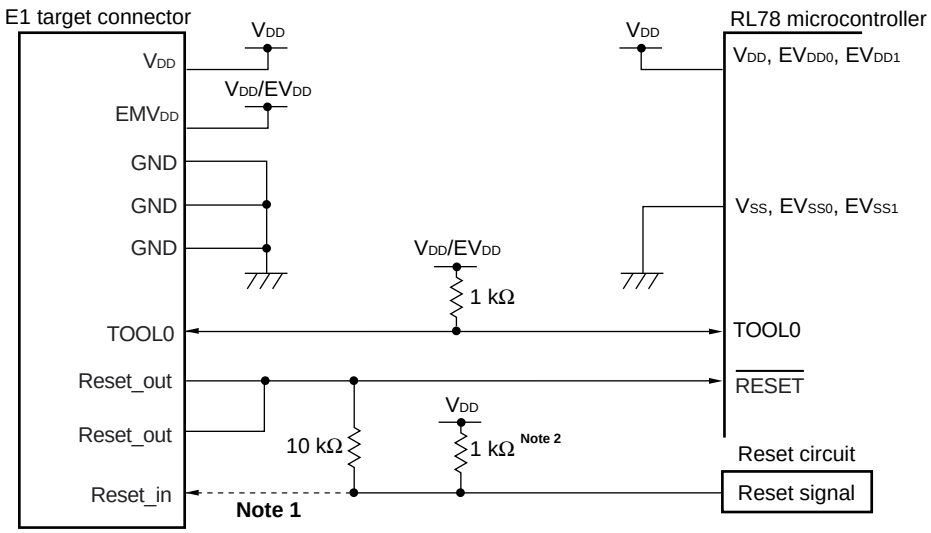
# CHAPTER 34 ON-CHIP DEBUG FUNCTION

## 34.1 Connecting E1 On-chip Debugging Emulator

The RL78 microcontroller uses the $V_{DD}$, $\overline{RESET}$, TOOL0, and $V_{SS}$ pins to communicate with the host machine via an E1 on-chip debugging emulator. Serial communication is performed by using a single-line UART that uses the TOOL0 pin.

**Caution The RL78 microcontroller has an on-chip debug function, which is provided for development and evaluation. Do not use the on-chip debug function in products designated for mass production, because the guaranteed number of rewritable times of the flash memory may be exceeded when this function is used, and product reliability therefore cannot be guaranteed. Renesas Electronics is not liable for problems occurring when the on-chip debug function is used.**
**Also, note that the debug function is disabled when power is supplied from the VBAT pin with the battery backup function.**

**Figure 34-1. Connection Example of E1 On-chip Debugging Emulator**




**Notes 1.** Connecting the dotted line is not necessary during serial programming.

**2.** If the reset circuit on the target system does not have a buffer and generates a reset signal only with resistors and capacitors, this pull-up resistor is not necessary.

**Caution This circuit diagram is assumed that the reset signal outputs from an N-ch O.D. buffer (output resistor: 100 Ω or less)**

**Remark** With products not provided with an $EV_{DD0}$, $EV_{DD1}$, $EV_{SS0}$, or $EV_{SS1}$ pin, replace $EV_{DD0}$ and $EV_{DD1}$ with $V_{DD}$, or replace $EV_{SS0}$ and $EV_{SS1}$ with $V_{SS}$.

## 36.2 Operation List

**Table 36-5. Operation List (1/18)**

| Instruction Group | Mnemonic | Operands | Bytes | Clocks | | Clocks | Flag | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Note 1 | Note 2 | | Z | AC | CY |
| 8-bit data transfer | MOV | r, #byte | 2 | 1 | – | r ← byte | | | |
| | | PSW, #byte | 3 | 3 | – | PSW ← byte | × | × | × |
| | | CS, #byte | 3 | 1 | – | CS ← byte | | | |
| | | ES, #byte | 2 | 1 | – | ES ← byte | | | |
| | | !addr16, #byte | 4 | 1 | – | (addr16) ← byte | | | |
| | | ES:!addr16, #byte | 5 | 2 | – | (ES, addr16) ← byte | | | |
| | | saddr, #byte | 3 | 1 | – | (saddr) ← byte | | | |
| | | sfr, #byte | 3 | 1 | – | sfr ← byte | | | |
| | | [DE+byte], #byte | 3 | 1 | – | (DE+byte) ← byte | | | |
| | | ES:[DE+byte],#byte | 4 | 2 | – | ((ES, DE)+byte) ← byte | | | |
| | | [HL+byte], #byte | 3 | 1 | – | (HL+byte) ← byte | | | |
| | | ES:[HL+byte],#byte | 4 | 2 | – | ((ES, HL)+byte) ← byte | | | |
| | | [SP+byte], #byte | 3 | 1 | – | (SP+byte) ← byte | | | |
| | | word[B], #byte | 4 | 1 | – | (B+word) ← byte | | | |
| | | ES:word[B], #byte | 5 | 2 | – | ((ES, B)+word) ← byte | | | |
| | | word[C], #byte | 4 | 1 | – | (C+word) ← byte | | | |
| | | ES:word[C], #byte | 5 | 2 | – | ((ES, C)+word) ← byte | | | |
| | | word[BC], #byte | 4 | 1 | – | (BC+word) ← byte | | | |
| | | ES:word[BC], #byte | 5 | 2 | – | ((ES, BC)+word) ← byte | | | |
| | | A, r Note 3 | 1 | 1 | – | A ← r | | | |
| | | r, A Note 3 | 1 | 1 | – | r ← A | | | |
| | | A, PSW | 2 | 1 | – | A ← PSW | | | |
| | | PSW, A | 2 | 3 | – | PSW ← A | × | × | × |
| | | A, CS | 2 | 1 | – | A ← CS | | | |
| | | CS, A | 2 | 1 | – | CS ← A | | | |
| | | A, ES | 2 | 1 | – | A ← ES | | | |
| | | ES, A | 2 | 1 | – | ES ← A | | | |
| | | A, !addr16 | 3 | 1 | 4 | A ← (addr16) | | | |
| | | A, ES:!addr16 | 4 | 2 | 5 | A ← (ES, addr16) | | | |
| | | !addr16, A | 3 | 1 | – | (addr16) ← A | | | |
| | | ES:!addr16, A | 4 | 2 | – | (ES, addr16) ← A | | | |
| | | A, saddr | 2 | 1 | – | A ← (saddr) | | | |
| | | saddr, A | 2 | 1 | – | (saddr) ← 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 area is accessed.

**3.** Except r = A

**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.