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	Obsolete
Core Processor	-
Core Size	-
Speed	-
Connectivity	-
Peripherals	-
Number of I/O	-
Program Memory Size	-
Program Memory Type	-
EEPROM Size	-
RAM Size	-
Voltage - Supply (Vcc/Vdd)	-
Data Converters	-
Oscillator Type	-
Operating Temperature	-
Mounting Type	-
Package / Case	-
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/renesas-electronics-america/upd78f9489gk-9eu-a

3.4.4 Register addressing

[Function]

In the register addressing mode, general-purpose registers are accessed as operands. The general-purpose register to be accessed is specified by a register specification code or functional name in the instruction code. Register addressing is carried out when an instruction with the following operand format is executed. When an 8-bit register is specified, one of the eight registers is specified with 3 bits in the instruction code.

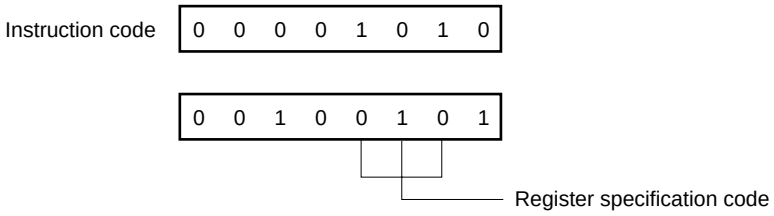
[Operand format]

Identifier	Description
r	X, A, C, B, E, D, L, H
rp	AX, BC, DE, HL

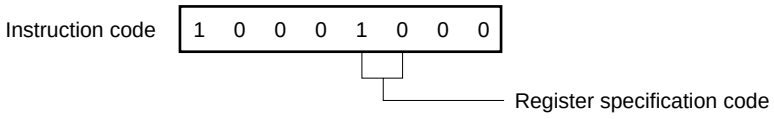
r and rp can be described with absolute names (R0 to R7 and RP0 to RP3) as well as function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL).

[Description example]

MOV A, C; When selecting the C register for r



INCW DE; When selecting the DE register pair for rp



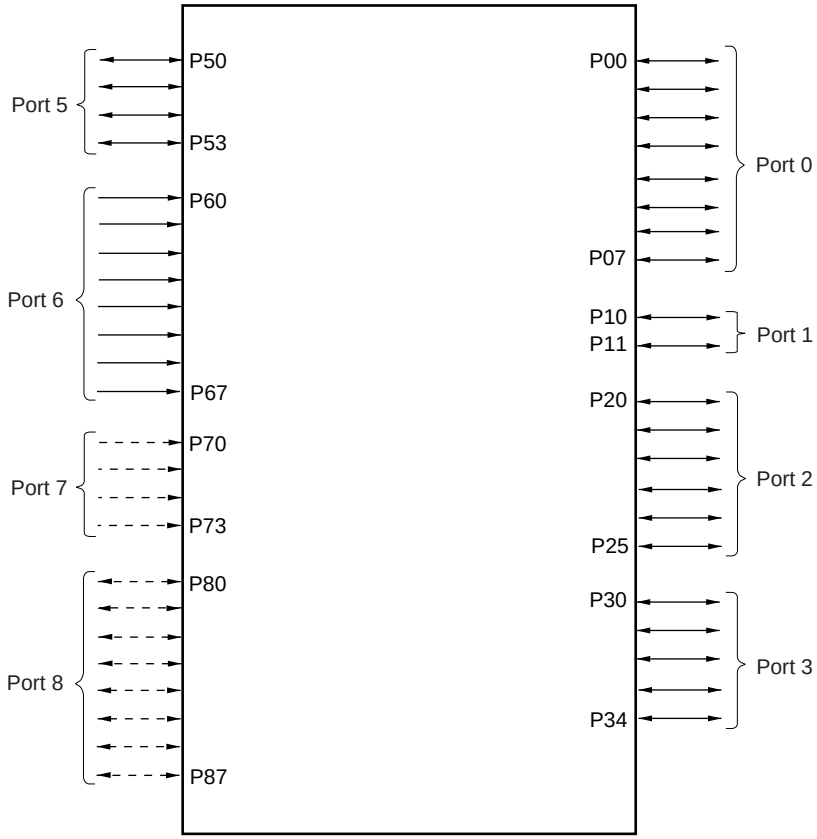
CHAPTER 4 PORT FUNCTIONS

4.1 Port Functions

The μPD789489 Subseries provides the ports shown in Figure 4-1, enabling various methods of control. The functions of each port are shown in Table 4-1.

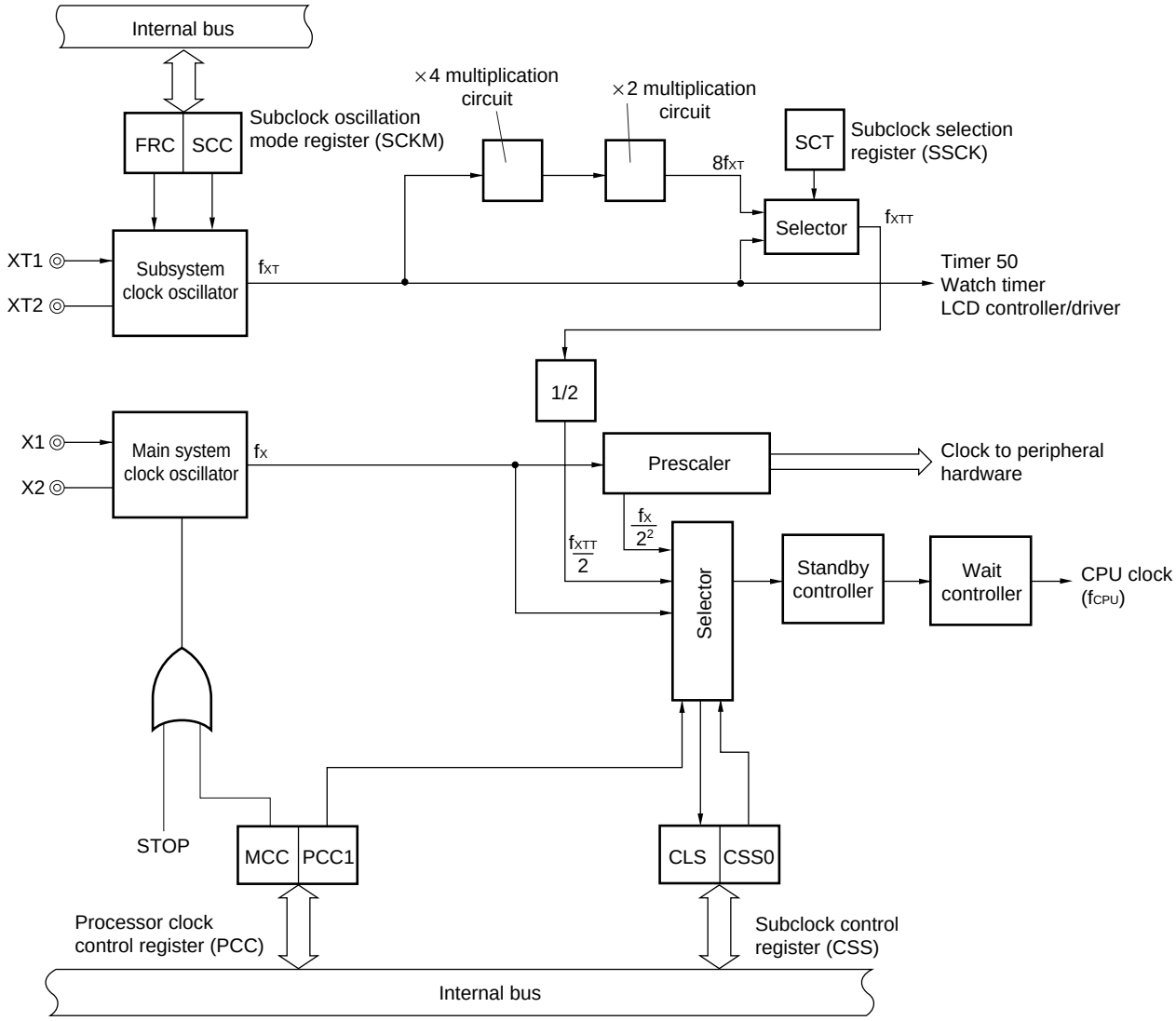
Numerous other functions are provided that can be used in addition to the digital I/O port functions. For more information on these additional functions, see **CHAPTER 2 PIN FUNCTIONS**.

Figure 4-1. Port Types



Remark Ports 7 and 8 are used when the port function is selected by a mask option or port function register.

Figure 5-2. Clock Generator Block Diagram (μPD78F9488, 78F9489)



Remark f_{XTT}: f_{XT} or $8f_{XT}$

5.4 System Clock Oscillators

5.4.1 Main system clock oscillator

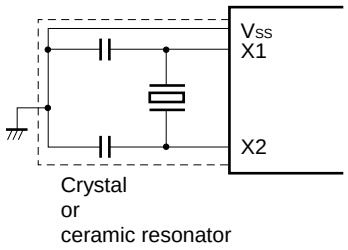
The main system clock oscillator is oscillated by the crystal or ceramic resonator (5.0 MHz TYP.) connected across the X1 and X2 pins.

An external clock can also be input to the circuit. In this case, input the clock signal to the X1 pin, and input the inverted signal to the X2 pin.

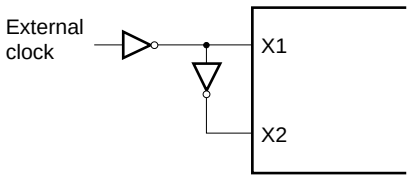
Figure 5-7 shows the external circuit of the main system clock oscillator.

Figure 5-7. External Circuit of Main System Clock Oscillator

(a) Crystal or ceramic oscillation



(b) External clock



Caution When using the main system clock oscillator, wire as follows in the area enclosed by the broken lines in Figure 5-7 to avoid an adverse effect from wiring capacitance.

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with the other signal lines. Do not route the wiring near a signal line through which a high fluctuating current flows.**
- **Always make the ground point of the oscillator capacitor the same potential as V_{SS}. Do not ground the capacitor to a ground pattern through which a high current flows.**
- **Do not fetch signals from the oscillator.**

6.4.4 16-bit timer counter 20 readout

The count value of 16-bit timer counter 20 (TM20) is read out using a 16-bit manipulation instruction.

TM20 readout is performed via the counter read buffer. The counter read buffer latches the TM20 count value, the buffer operation is held pending at the CPU clock falling edge after the read signal of the TM20 lower byte rises, and the count value is retained. The retained counter read buffer value can be read out as the count value.

Cancellation of the pending state is performed at the CPU clock falling edge after the read signal of the TM20 higher byte falls.

$\overline{\text{RESET}}$ input sets TM20 to 0000H and TM20 starts free running.

Figure 6-10 shows the timing of 16-bit timer counter 20 readout.

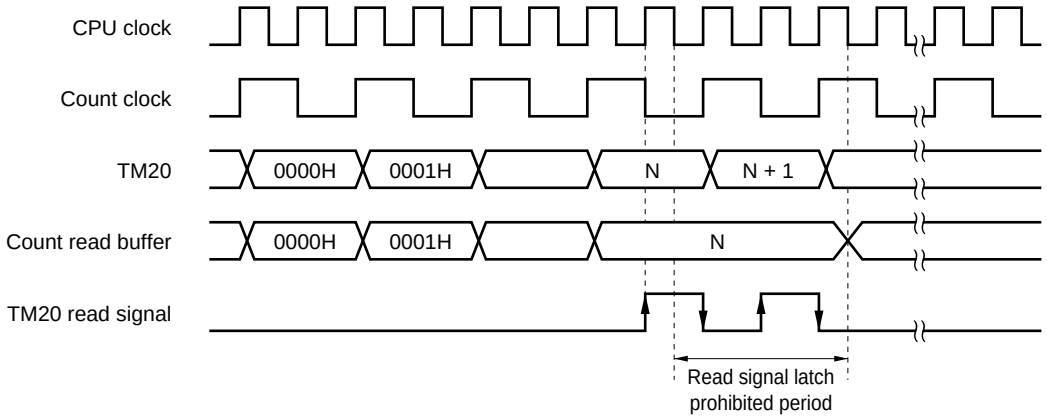
Cautions 1. The count value after releasing stop becomes undefined because the count operation is executed during the oscillation stabilization time.

2. Though TM20 is designed for a 16-bit transfer instruction, an 8-bit transfer instruction can also be used.

When using an 8-bit transfer instruction, execute it by direct addressing.

3. When using an 8-bit transfer instruction, execute in the order from lower byte to higher byte in pairs. If only the lower byte is read, the pending state of the counter read buffer is not canceled, and if only the higher byte is read, an undefined count value is read.

Figure 6-10. 16-Bit Timer Counter 20 Readout Timing



Remark N = 0000H to FFFFH

(5) PPG output mode (PPG: Programmable Pulse Generator)

Pulses are output using any cycle or duty ratio (pulse width) set (both the cycle and pulse width are programmable).

(6) 24-bit event counter mode

Operation as an external event counter with 24-bit resolution is enabled using 16-bit timer 20 and timer 61. However, this mode operates only as a counter read function.

There is no compare, match, or clear function.

<Setting method>

<1> Select the timer 61 interrupt signal for the count clock of 16-bit timer 20 (TCL201 = 0, TCL200 = 0)

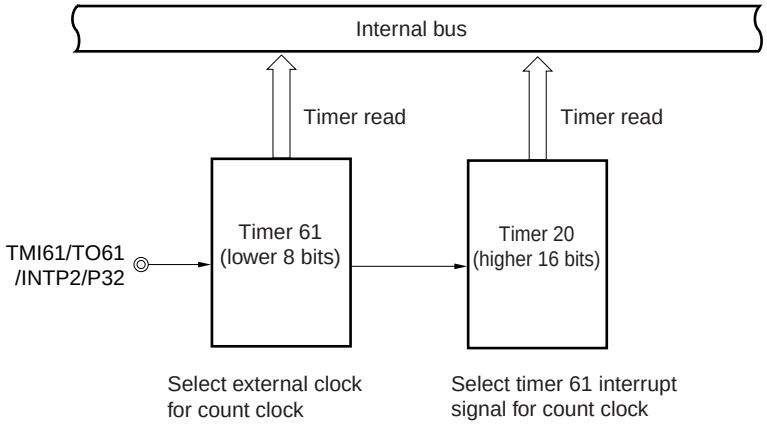
<2> Set timer 61 in stand-alone mode (TMD611 = 0)
Select the external clock input from pin TMI61 for the count clock of timer 61
((TCL612 = 0, TCL611 = 1) or (TCL612 = 1, TCL611 = 0))

<3> Set CR61 to FFH

<4> Read the current count value of 16-bit timer 20
(16-bit timer 20 does not have a count clear function and is counting constantly)

<5> Enable timer 61 count operation (TCE61 = 1)

Figure 7-1. Block Diagram of 24-Bit Event Counter



(3) Operation as square-wave output with 8-bit resolution

Square waves of any frequency can be output at an interval specified by the value preset in 8-bit compare register nm (CRnm).

To operate timer nm for square-wave output, settings must be made in the following sequence.

<1> When using timer 50, set P30 to output mode (PM30 = 0) and the P30 output latch to 0, respectively.
When using timer 60, set P31 to output mode (PM31 = 0) and the P31 output latch to 0, respectively.
When using timer 61, set P32 to output mode (PM32 = 0) and the P32 output latch to 0, respectively.

<2> Disable operation of timer counter nm (TMnm) (TCEnm = 0).

<3> Set a count clock for timer nm (see **Figures 7-6**, **7-7** and **7-9**)

<4> For timer 50, enable timer output of TO50 (TOE50 = 1).
For timer 60, enable timer output of TO60 (TOE600 = 1).
For timer 61, enable timer output of TO61 (TOE610 = 1).

<5> Set a count value in CRnm.

<6> Enable the operation of TMnm (TCEnm0 = 1).

When the count value of TMnm matches the value set in CRnm, the TOnm pin output will be inverted. Through application of this mechanism, square waves of any frequency can be output. As soon as a match occurs, TMnm is cleared to 00H and continues counting. At the same time, an interrupt request signal (INTTMnm) is generated.

The square-wave output is cleared to 0 by setting TCEnm to 0.

Tables 7-6 to 7-8 show the square-wave output range, and Figure 7-18 shows the timing of square-wave output.

Caution Be sure to stop the timer operation before overwriting the count clock with different data.

Remark nm = 50, 60, 61

Table 7-6. Square-Wave Output Range of Timer 50

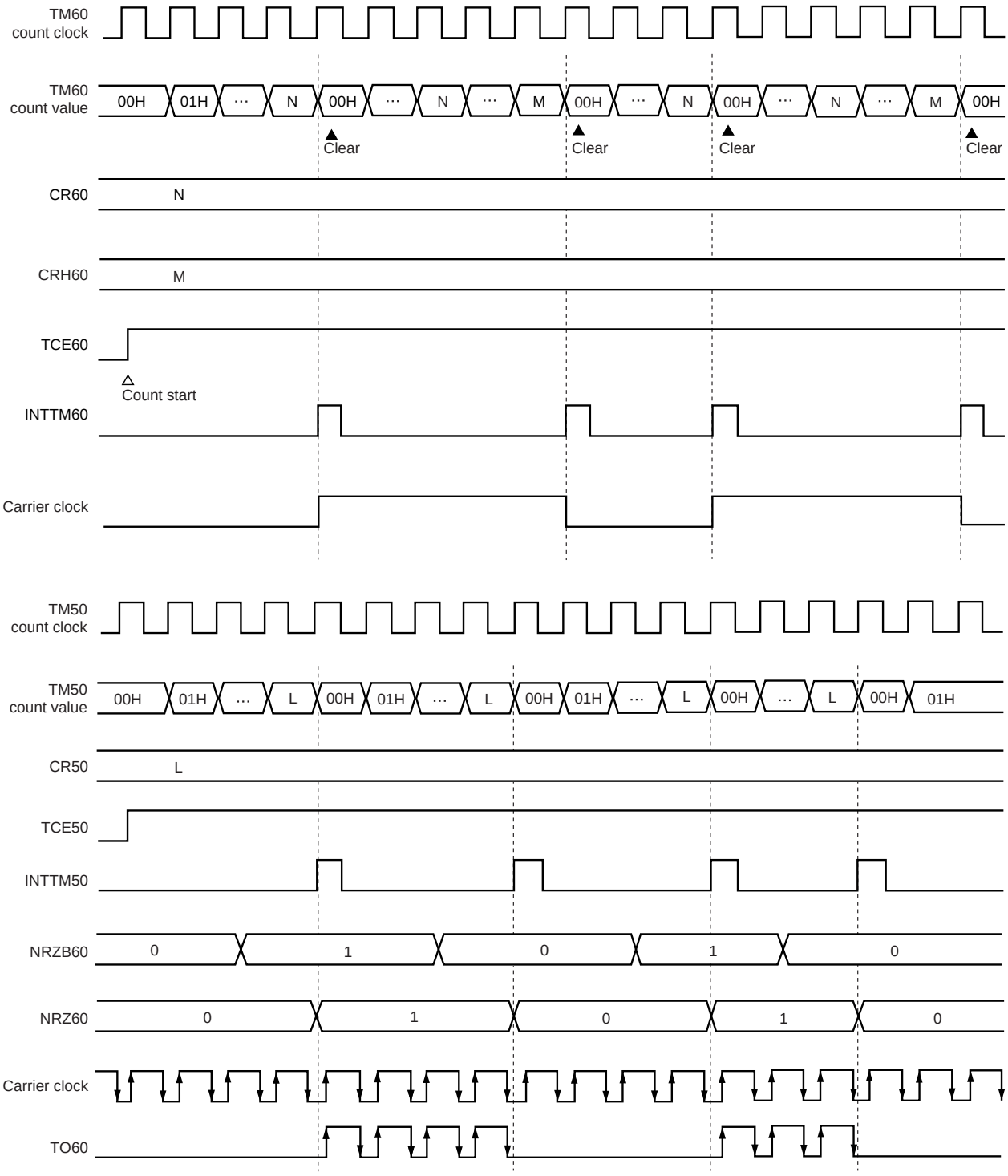
TCL502	TCL501	TCL500	Minimum Pulse Width	Maximum Pulse Width	Resolution
0	0	0	$1/f_X$ (0.2 μs)	$2^8/f_X$ (51.2 μs)	$1/f_X$ (0.2 μs)
0	0	1	$2^3/f_X$ (1.6 μs)	$2^{11}/f_X$ (409.6 μs)	$2^3/f_X$ (1.6 μs)
0	1	0	$2^7/f_X$ (25.6 μs)	$2^{15}/f_X$ (6.55 ms)	$2^7/f_X$ (25.6 μs)
0	1	1	$1/f_{XT}$ (30.5 μs)	$2^8/f_{XT}$ (7.81 ms)	$1/f_{XT}$ (30.5 μs)
1	0	0	Input cycle of timer 60 match signal	Input cycle of timer 60 match signal $\times 2^8$	Input cycle of timer 60 match signal
1	0	1	Input cycle of timer 60 output	Input cycle of timer 60 output $\times 2^8$	Input cycle of timer 60 output

Remarks 1. f_X: Main system clock oscillation frequency

2. f_{XT}: Subsystem clock oscillation frequency

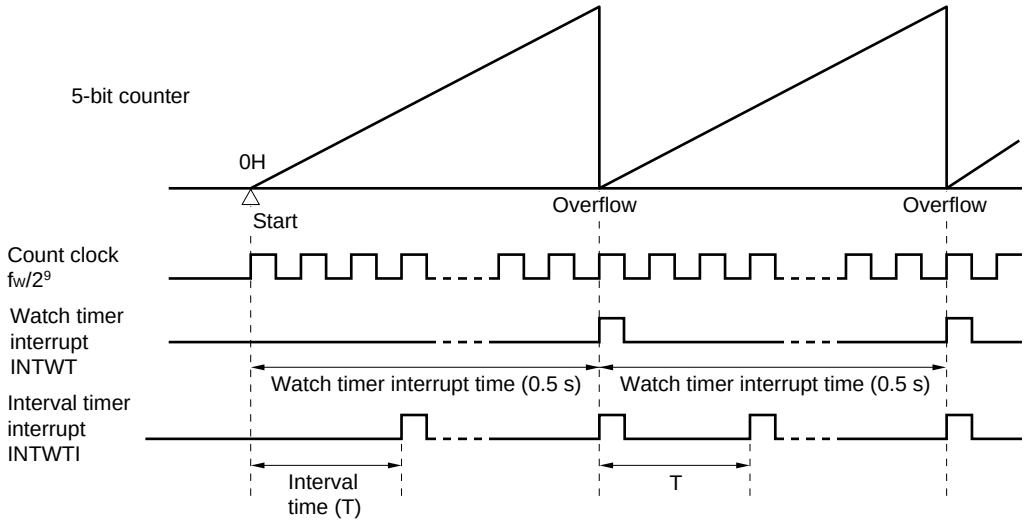
3. The parenthesized values apply to operation at f_X = 5.0 MHz or f_{XT} = 32.768 kHz.

★ **Figure 7-22. Timing of Carrier Generator Operation (When CR60 = N, CRH60 = M (M > N))**



Remark 00H ≤ N < M ≤ FFH, L = 00H to FFH

Figure 8-4. Watch Timer/Interval Timer Operation Timing



Caution When operation of the watch timer and 5-bit counter operation is enabled by setting bit 0 (WTM0) of the watch timer mode control register (WTM) to 1, the interval until the first interrupt request (INTWT) is generated after the register is set does not exactly match the watch timer interrupt time (0.5 s). This is because there is a delay of one 9-bit pre-scaler output cycle until the 5-bit counter starts counting. Subsequently, however, the INTWT signal is generated at the specified intervals.

Remarks 1. f_W: Watch timer clock frequency

2. The parenthesized values apply to operation at f_W = 32.768 kHz.

CHAPTER 10 10-BIT A/D CONVERTER

10.1 10-Bit A/D Converter Functions

The 10-bit A/D converter is a 10-bit resolution converter used to convert analog inputs into digital signals. This converter can control eight channels (ANI0 to ANI7) of analog inputs.

A/D conversion can only be started by software.

One of analog inputs ANI0 to ANI7 is selected for A/D conversion. A/D conversion is performed repeatedly, with an interrupt request (INTAD0) being issued each time A/D conversion is complete.

10.2 10-Bit A/D Converter Configuration

The 10-bit A/D converter includes the following hardware.

Table 10-1. Configuration of 10-Bit A/D Converter

Item	Configuration
Analog inputs	8 channels (ANI0 to ANI7)
Registers	Successive approximation register (SAR) A/D conversion result register 0 (ADCRL0)
Control registers	A/D converter mode register 0 (ADML0) Analog input channel specification register 0 (ADS0)

(3) Asynchronous serial interface status register 20 (ASIS20)

ASIS20 indicates the type of a reception error, if it occurs while asynchronous serial interface mode is set.

ASIS20 is set with a 1-bit or 8-bit memory manipulation instruction.

The contents of ASIS20 are undefined in 3-wire serial I/O mode.

$\overline{\text{RESET}}$ input sets ASIS20 to 00H.

Figure 11-5. Format of Asynchronous Serial Interface Status Register 20

Symbol	7	6	5	4	3	<2>	<1>	<0>	Address	After reset	R/W
ASIS20	0	0	0	0	0	PE20	FE20	OVE20	FF71H	00H	R

PE20	Parity error flag
0	No parity error occurred.
1	A parity error occurred (when the transmit parity and receive parity did not match).

FE20	Framing error flag
0	No framing error occurred.
1	A framing error occurred (when stop bit was not detected).Note 1

OVE20	Overrun error flag
0	No overrun error occurred.
1	An overrun error occurredNote 2 (when the next receive operation was completed before the data was read from receive buffer register 20).

Notes 1. Even when the stop bit length is set to 2 bits by setting bit 2 (SL20) of asynchronous serial interface mode register 20 (ASIM20), the stop bit detection at reception is performed with 1 bit.

2. Be sure to read receive buffer register 20 (RXB20) when an overrun error occurs. If not, an overrun error will occur every time data is received.

(b) Generation of UART baud rate transmit/receive clock from external clock input to ASCK20 pin

The transmit/receive clock is generated by dividing the clock input from the ASCK20 pin. The baud rate of a clock generated from the clock input to the ASCK20 pin is estimated by using the following expression.

$$[\text{Baud rate}] = \frac{f_{ASCK}}{16} \text{ [bps]}$$

f_{ASCK}: Frequency of clock input to the ASCK20 pin

Table 11-4. Relationship Between ASCK20 Pin Input Frequency and Baud Rate (When BRGC20 Is Set to 80H)

Baud Rate (bps)	ASCK20 Pin Input Frequency (kHz)
75	1.2
150	2.4
300	4.8
600	9.6
1,200	19.2
2,400	38.4
4,800	76.8
9,600	153.6
19,200	307.2
31,250	500.0
38,400	614.4

(c) Generation of serial clock from system clock in 3-wire serial I/O

The serial clock is generated by dividing the system clock. The frequency of the serial clock can be obtained by the following expression. If the serial clock is externally input to the $\overline{SCK20}$ pin, it is unnecessary to set BRGC20.

$$\text{Serial clock frequency} = \frac{f_X}{2^{n+1}} \text{ [Hz]}$$

f_X: Main system clock oscillation frequency

n: Values in Figure 11-6 determined by the settings of TPS200 to TPS203 ($1 \leq n \leq 8$)

(c) Baud rate generator control register 20 (BRGC20)

BRGC20 is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BRGC20 to 00H.

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
BRGC20	TPS203	TPS202	TPS201	TPS200	0	0	0	0	FF73H	00H	R/W

TPS203	TPS202	TPS201	TPS200	Selection of baud rate generator source clock	n
0	0	0	0	$f_X/2$ (2.5 MHz)	1
0	0	0	1	$f_X/2^2$ (1.25 MHz)	2
0	0	1	0	$f_X/2^3$ (625 kHz)	3
0	0	1	1	$f_X/2^4$ (313 kHz)	4
0	1	0	0	$f_X/2^5$ (156 kHz)	5
0	1	0	1	$f_X/2^6$ (78.1 kHz)	6
0	1	1	0	$f_X/2^7$ (39.1 kHz)	7
0	1	1	1	$f_X/2^8$ (19.5 kHz)	8
Other than above				Setting prohibited	

Caution When writing to BRGC20 during a communication operation, the baud rate generator output is disrupted and communications cannot be performed normally. Be sure not to write to BRGC20 during a communication operation.

Remarks 1. f_X: Main system clock oscillation frequency

2. n: Values determined by the settings of TPS200 to TPS203 ($1 \leq n \leq 8$)

3. The parenthesized values apply to operation at f_X = 5.0 MHz.

If the internal clock is used as the serial clock for 3-wire serial I/O mode, set bits TPS200 to TPS203 to set the frequency of the serial clock. To obtain the frequency to be set, use the following expression. When an external clock is used, setting BRGC20 is not necessary.

$$\text{Serial clock frequency} = \frac{f_X}{2^{n+1}} \text{ [Hz]}$$

f_X: Main system clock oscillation frequency

n: Values in the above table determined by the settings of TPS200 to TPS203 ($1 \leq n \leq 8$)

Symbol	<7>	6	5	<4>	<3>	<2>	<1>	<0>	Address	After reset	R/W
ADTI0	ADTI07	0	0	ADTI04	ADTI03	ADTI02	ADTI01	ADTI00	FF7BH	00H	R/W

ADTI04	ADTI03	ADTI02	ADTI01	ADTI00	Data transfer interval specification (f_X = 5.0 MHz, f_{SCK} = 1.25 MHz)[Note 2]	n
1	0	0	0	0	13.6 μs + 0.5/f_{SCK}	16
1	0	0	0	1	14.4 μs + 0.5/f_{SCK}	17
1	0	0	1	0	15.2 μs + 0.5/f_{SCK}	18
1	0	0	1	1	16.0 μs + 0.5/f_{SCK}	19
1	0	1	0	0	16.8 μs + 0.5/f_{SCK}	20
1	0	1	0	1	17.6 μs + 0.5/f_{SCK}	21
1	0	1	1	0	18.4 μs + 0.5/f_{SCK}	22
1	0	1	1	1	19.2 μs + 0.5/f_{SCK}	23
1	1	0	0	0	20.0 μs + 0.5/f_{SCK}	24
1	1	0	0	1	20.8 μs + 0.5/f_{SCK}	25
1	1	0	1	0	21.6 μs + 0.5/f_{SCK}	26
1	1	0	1	1	22.4 μs + 0.5/f_{SCK}	27
1	1	1	0	0	23.2 μs + 0.5/f_{SCK}	28
1	1	1	0	1	24.0 μs + 0.5/f_{SCK}	29
1	1	1	1	0	24.8 μs + 0.5/f_{SCK}	30
1	1	1	1	1	25.6 μs + 0.5/f_{SCK}	31

Notes 1. The interval time depends only on the CPU processing.

2. The data transfer interval time is found from the following expressions (n: Value set to ADTI00 to ADTI04).

<1> n = 0

$$\text{Interval time} = \frac{2}{f_{SCK}} + \frac{0.5}{f_{SCK}}$$

<2> n = 1 to 31

$$\text{Interval time} = \frac{n+1}{f_{SCK}} + \frac{0.5}{f_{SCK}}$$

Cautions 1. Do not write to ADTI0 during operation of the automatic transmit/receive function.

2. Be sure to set bits 5 and 6 to 0.

Remark f_X: Main system clock oscillation frequency

f_{SCK}: Serial clock frequency

Figure 15-3. Format of Remote Controller Receive Control Register (2/2)

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
RMCN	RMEN	NCW	PRSEN	RMIN	0	0	RMCK1	RMCK0	FF60H	00H	R/W

RMCK1	RMCK0	Selection of source clock (f_{REM}) of remote controller counter
0	0	$f_X/2^6$ (625 kHz)
0	1	$f_X/2^7$ (313 kHz)
1	0	$f_X/2^8$ (156 kHz)
1	1	f_{XT} (32.768 kHz)

Cautions 1. Always set bits 2 and 3 to 0.

2. To change the values of NCW, PRSEN, RMIN, RMCK1, and RMCK0, disable remote controller reception (RMEN = 0) first.

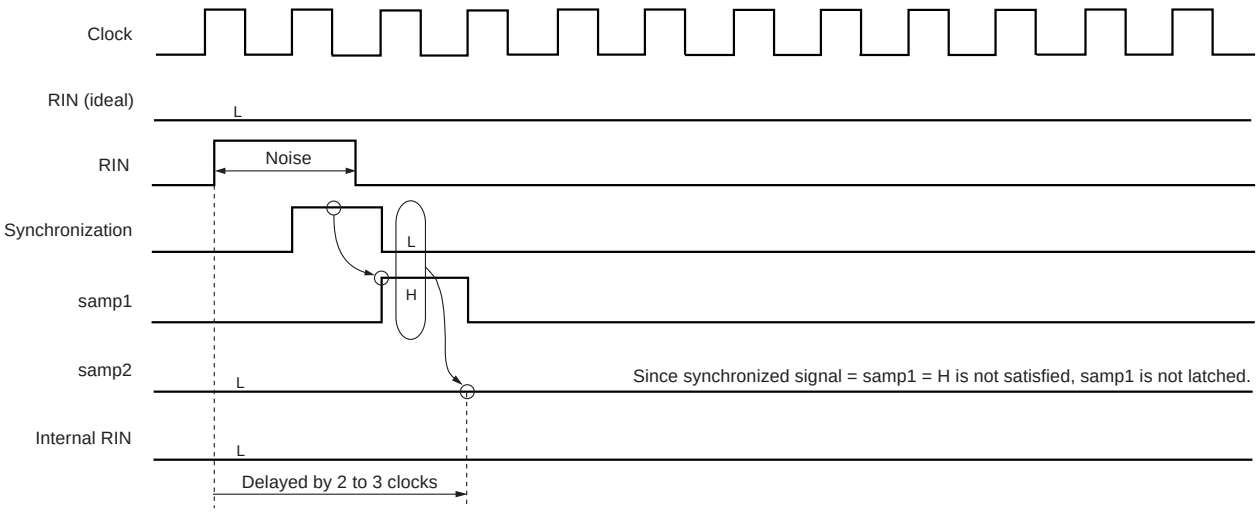
Remarks 1. f_X: Oscillation frequency of main system clock

2. f_{XT}: Oscillation frequency of subsystem clock

3. The parenthesized values apply to operation at f_X = 4.0 MHz and f_{XT} = 32.768 kHz.

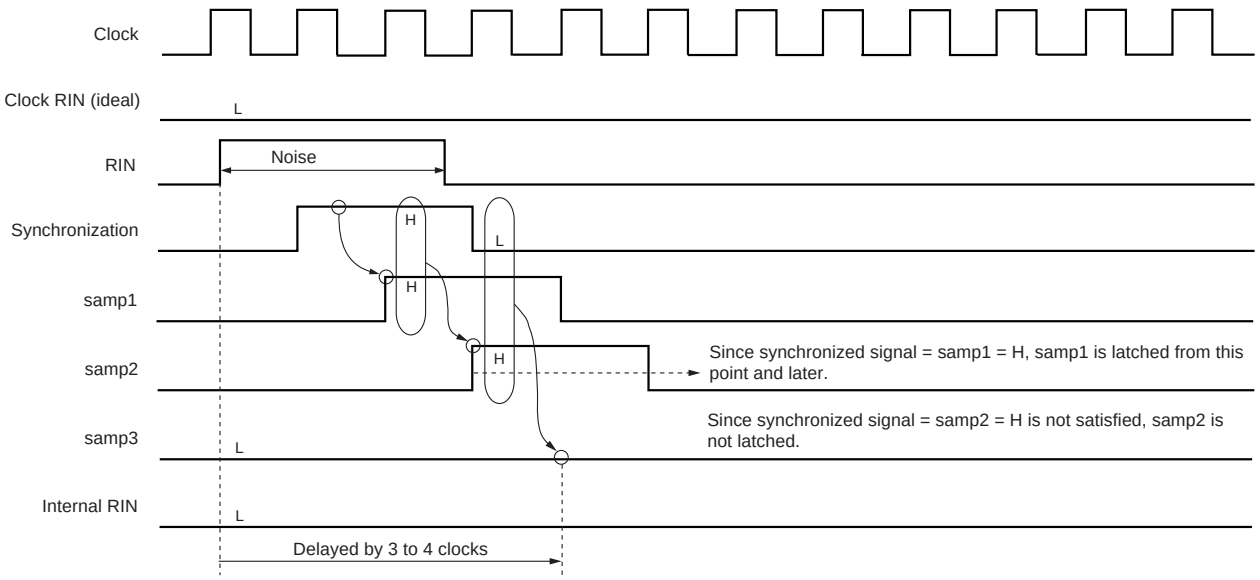
Figure 15-8. Noise Elimination Operation Example (1/2)

(a) 1-clock noise elimination (PRSEN = 0, NCW = 0)



Remark Internal RIN is a signal after synchronization and sampling are performed twice, and is therefore later than the actual signal input from the outside to the RIN pin by two to three clocks.

(b) 2-clock noise elimination (PRSEN = 0, NCW = 1)



Remark Internal RIN is a signal after synchronization and sampling are performed three times, and is therefore later than the actual signal input from the outside to the RIN pin by 3 to 4 clocks.

16.4.2 Maskable interrupt request acknowledgment operation

A maskable interrupt request can be acknowledged when the interrupt request flag is set to 1 and the corresponding interrupt mask flag is cleared to 0. A vectored interrupt is acknowledged in the interrupt enabled status (when the IE flag is set to 1).

The time required to start the interrupt servicing after a maskable interrupt request has been generated is shown in Table 16-4.

Refer to Figures 16-14 and 16-15 for the timing of interrupt request acknowledgement.

Table 16-4. Time from Generation of Maskable Interrupt Request to Servicing

Minimum Time	Maximum Time[Note]
9 clocks	19 clocks

Note The wait time is maximum when an interrupt request is generated immediately before the BT or BF instruction.

Remark 1 clock: $\frac{1}{f_{CPU}}$ (f_{CPU}: CPU clock)

When two or more maskable interrupt requests are generated at the same time, they are acknowledged starting from the one assigned the highest priority by the priority specification flag.

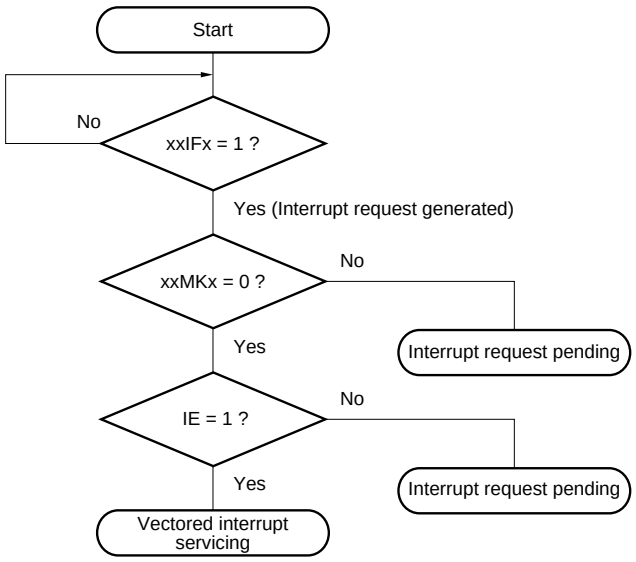
A pending interrupt is acknowledged when the status in which it can be acknowledged is set.

Figure 16-13 shows the algorithm of interrupt request acknowledgment.

When a maskable interrupt request is acknowledged, the PSW and PC are saved to the stack in that order, the IE flag is reset to 0, the data in the vector table determined for each interrupt request is loaded to the PC, and execution branches.

To return from interrupt servicing, use the RETI instruction.

Figure 16-13. Interrupt Request Acknowledgment Program Algorithm



xxIFx: Interrupt request flag

xxMKx: Interrupt mask flag

IE: Flag to control maskable interrupt request acknowledgment (1 = enable, 0 = disable)

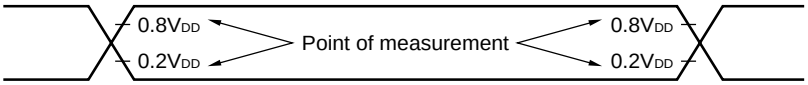
21.1.2 Description of “Operation” column

A:	A register; 8-bit accumulator
X:	X register
B:	B register
C:	C register
D:	D register
E:	E register
H:	H register
L:	L register
AX:	AX register pair; 16-bit accumulator
BC:	BC register pair
DE:	DE register pair
HL:	HL register pair
PC:	Program counter
SP:	Stack pointer
PSW:	Program status word
CY:	Carry flag
AC:	Auxiliary carry flag
Z:	Zero flag
IE:	Interrupt request enable flag
( ):	Memory contents indicated by address or register contents in parenthesis
X_H, X_L:	Higher 8 bits and lower 8 bits of 16-bit register
∧:	Logical product (AND)
∨:	Logical sum (OR)
∀:	Exclusive logical sum (exclusive OR)
¯:	Inverted data
addr16:	16-bit immediate data or label
jdisp8:	Signed 8-bit data (displacement value)

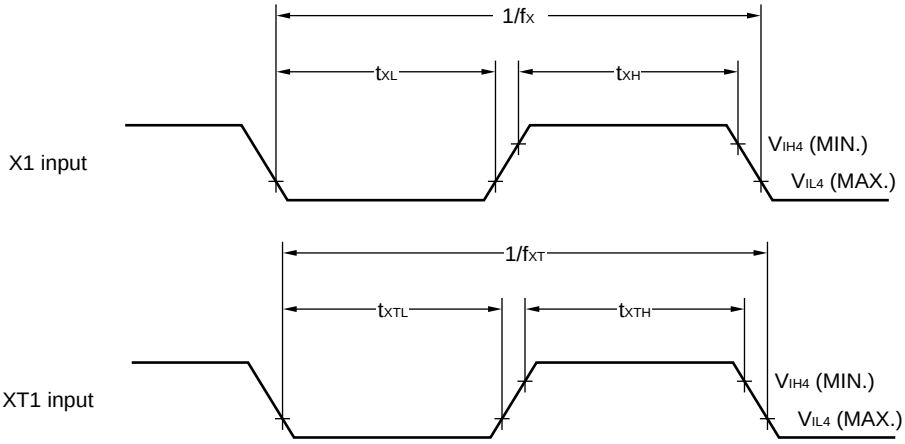
21.1.3 Description of “Flag” column

(Blank):	Unchanged
0:	Cleared to 0
1:	Set to 1
x:	Set/cleared according to the result
R:	Previously saved value is restored

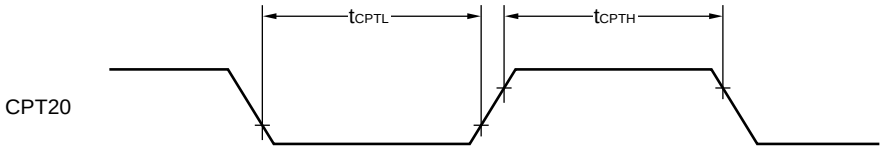
AC Timing Measurement Points (Excluding X1 and XT1 Inputs)



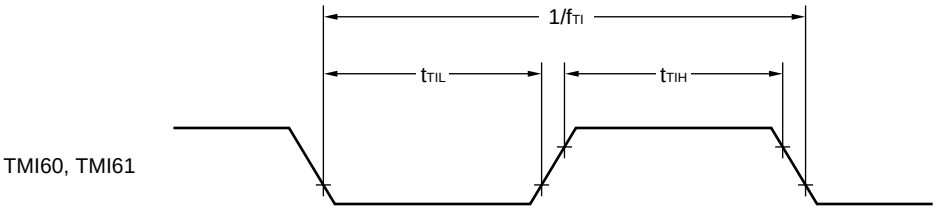
Clock Timing



Capture Input Timing



TMI Timing



Interrupt Input Timing