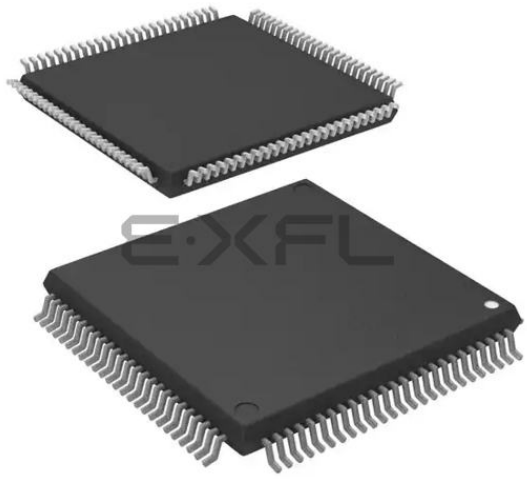
Renesas Electronics America Inc - UPD70F3913GF(R)-GAS-AX Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	V850E1
Core Size	32-Bit Single-Core
Speed	100MHz
Connectivity	CSI, I²C, UART/USART
Peripherals	DMA, LVD, POR, PWM, WDT
Number of I/O	55
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	24K x 8
Voltage - Supply (Vcc/Vdd)	1.35V ~ 1.65V
Data Converters	A/D 12x10b, 7x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/renesas-electronics-america/upd70f3913gf-r-gas-ax

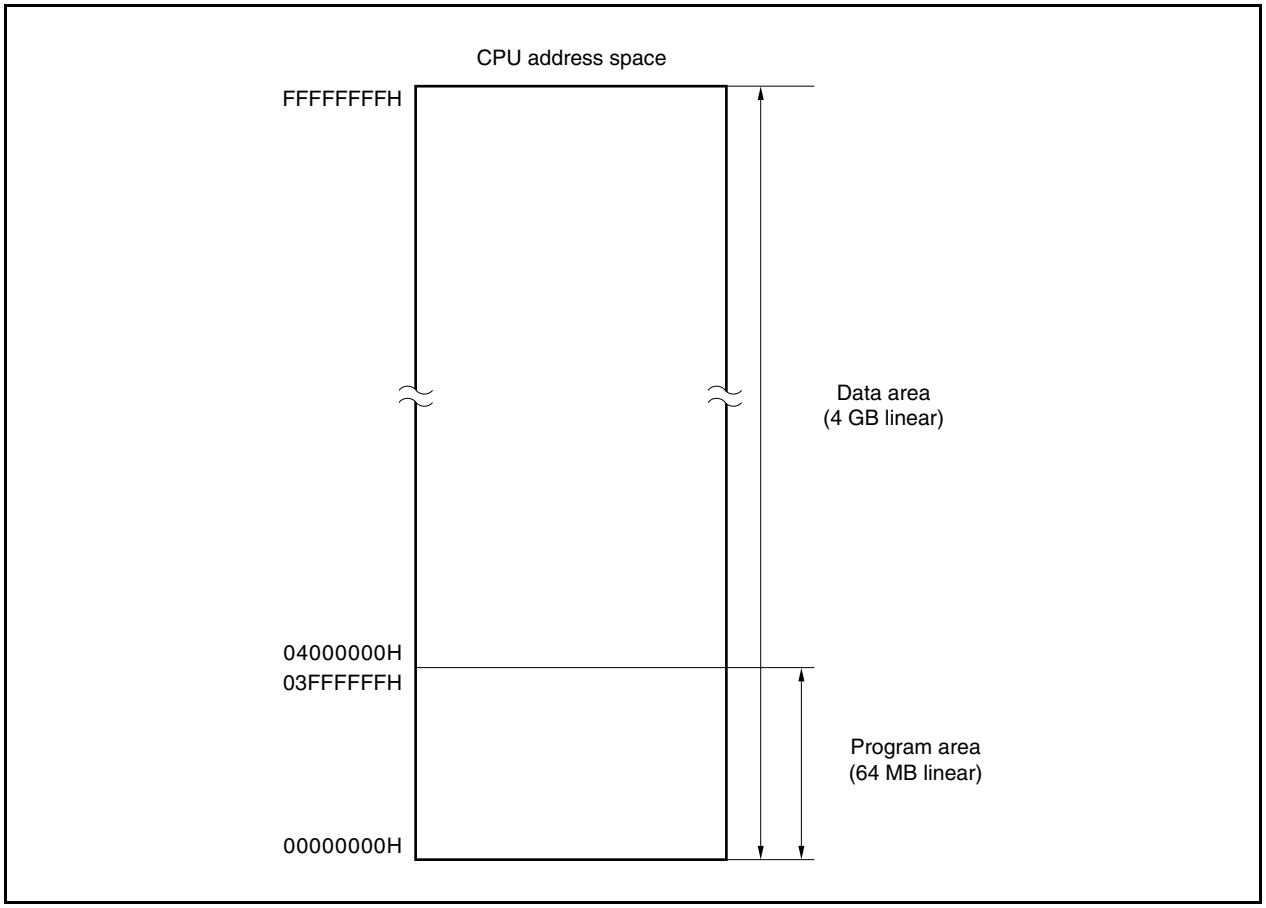
3.4 Address Space

3.4.1 CPU address space

The CPU of the V850E/IG4 and V850E/IH4 has 32-bit architecture and supports up to 4 GB of linear address space (data space) during operand addressing (data access). Also, in instruction address addressing, a maximum of 64 MB of linear address space (program space) is supported.

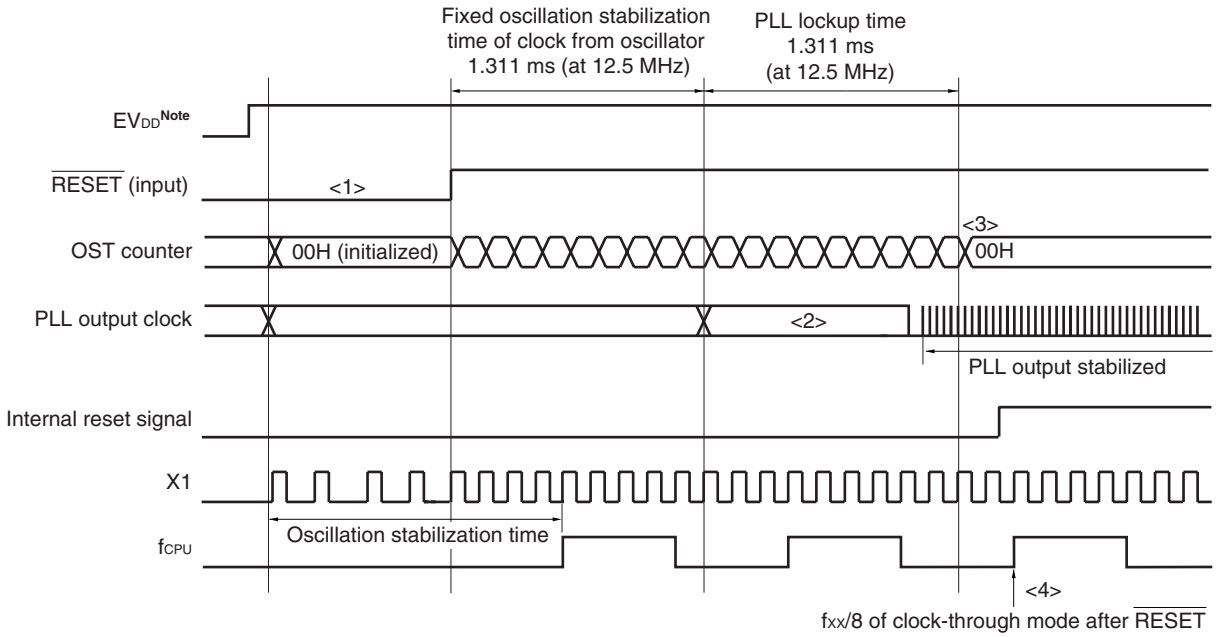
Figure 3-2 shows the CPU address space.

Figure 3-2. CPU Address Space



5.5.2 Operation timing

(1) Power on (power-on reset)



<1> The oscillator is activated during the $\overline{\text{RESET}}$ period that follows power application.
Make sure that the low-level width of the $\overline{\text{RESET}}$ signal is “Oscillation stabilization time of the used resonator – Fixed oscillation stabilization time” or more, taking the oscillation stabilization time into consideration.
PLL stops during the $\overline{\text{RESET}}$ period and fixed oscillation stabilization time.

<2> When the fixed oscillation stabilization time that elapses after the $\overline{\text{RESET}}$ signal is released expires, PLL stop is released, and counting the lockup time starts.

<3> PLL is locked when counting of the lockup time is over. The OST counter is initialized to 00H.

<4> When the lockup time expires, the CPU releases the reset signal and operates in the clock-through mode (f_X). The CPU operation clock (f_{CPU}) is f_{XX}/8. The PLL mode can be set by software.

Note V850E/IG4: EV_{DD0}, EV_{DD1}, EV_{DD2}
V850E/IH4: EV_{DD0}, EV_{DD1}, EV_{DD2}, EV_{DD3}

Cautions 1. The clock generated by the oscillator starts oscillating during the $\overline{\text{RESET}}$ period. After the $\overline{\text{RESET}}$ signal is released, a specific wait time (fixed oscillation stabilization time) elapses.

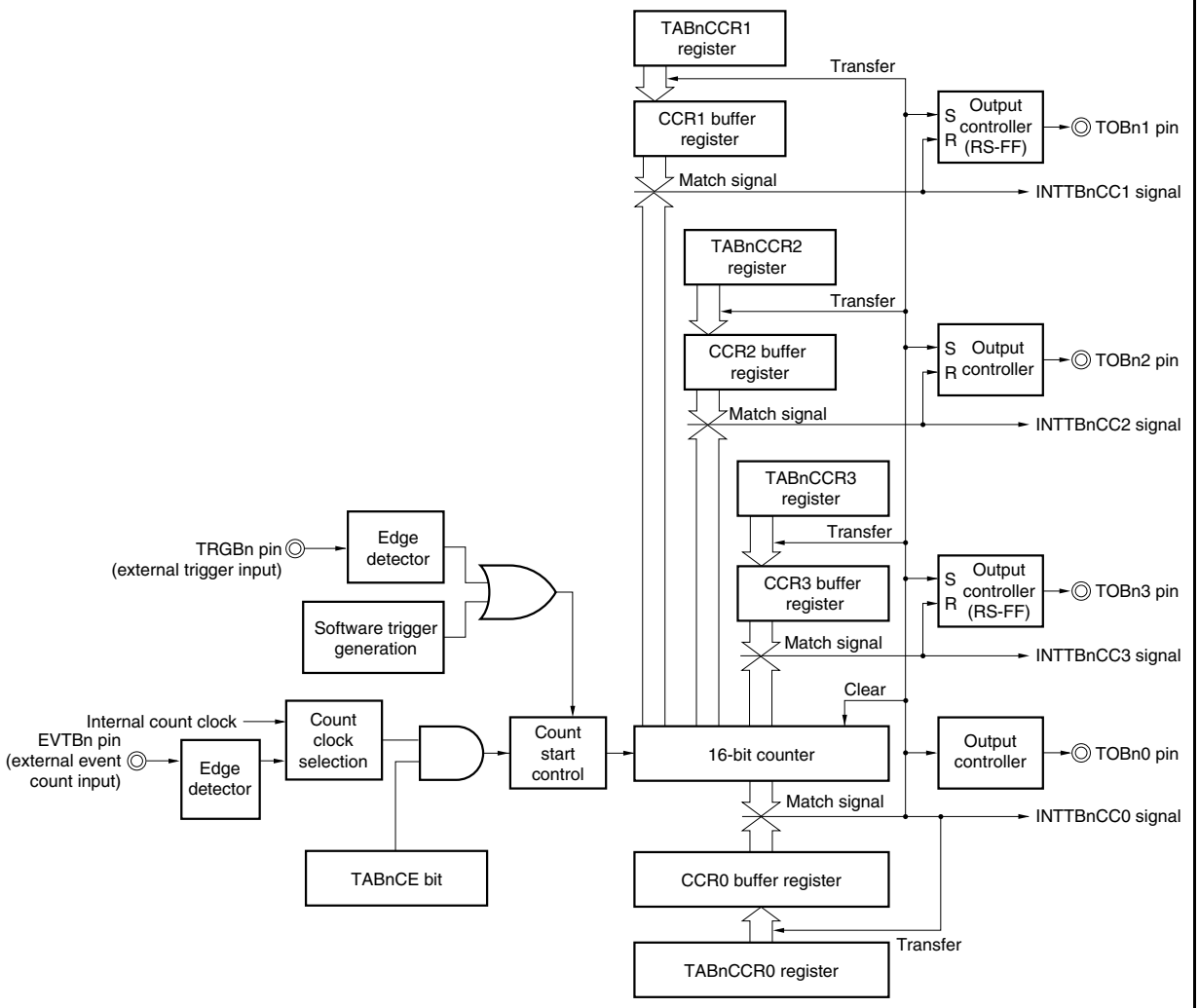
2. To avoid malfunction due to noise, do not change the division ratio of the CPU operation clock (f_{CPU}) by using the PCC register before setting the PLL mode. Before changing the division ratio, be sure to select the PLL mode.

7.6.3 External trigger pulse output mode (TABnMD2 to TABnMD0 bits = 010)

In the external trigger pulse output mode, 16-bit timer/event counter AB waits for a trigger when the TABnCTL0.TABnCE bit is set to 1. When the valid edge of an external trigger input signal (TRGBn) is detected, 16-bit timer/event counter AB starts counting, and outputs a PWM waveform (up to 3-phase) from the TOBn1 to TOBn3 pins. A PWM waveform with a duty factor of 50% whose half cycle is the set value of the TABnCCR0 register + 1 can also be output from the TOBn0 pin.

Pulses can also be output by generating a software trigger instead of using the external trigger input.

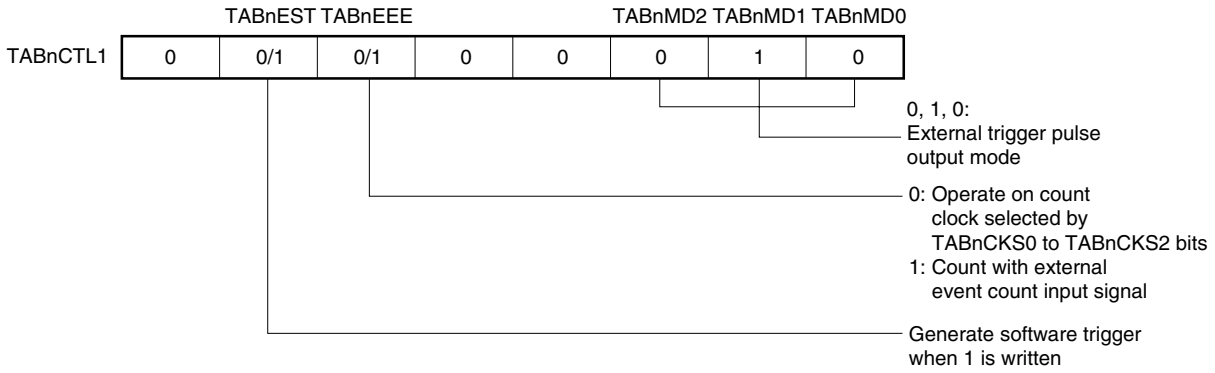
Figure 7-20. Configuration in External Trigger Pulse Output Mode



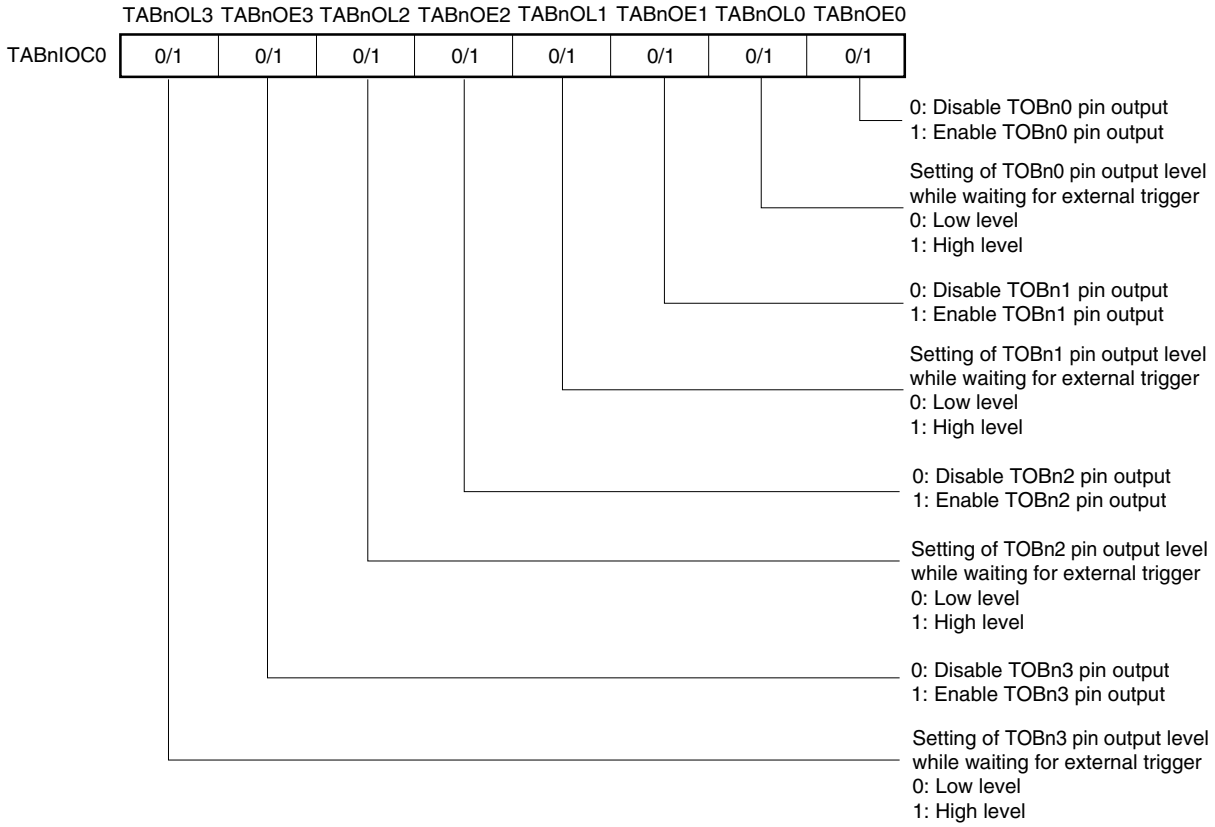
Remark n = 0, 1

Figure 7-22. Setting of Registers in External Trigger Pulse Output Mode (2/3)

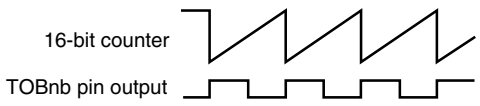
(b) TABn control register 1 (TABnCTL1)



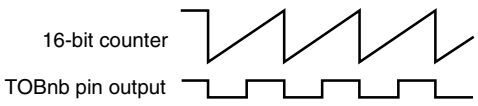
(c) TABn I/O control register 0 (TABnIOC0)



- When TABnOLb bit = 0



- When TABnOLb bit = 1



8.4 Registers

(1) TMTn control register 0 (TTnCTL0)

The TTnCTL0 register is an 8-bit register that controls the operation of TMTn.

This register can be read or written in 8-bit or 1-bit units.

Reset sets this register to 00H.

The same value can always be written to the TTnCTL0 register by software.

After reset: 00H R/W Address: TT0CTL0 FFFFF580H, TT1CTL0 FFFFF5C0H, TT2CTL0 FFFFF780H, TT3CTL0 FFFFF7C0H

	<7>	6	5	4	3	2	1	0
TTnCTL0 (n = 0 to 3)	TTnCE	0	0	0	0	TTnCKS2	TTnCKS1	TTnCKS0

TTnCE	TMTn operation control
0	TMTn operation disabled (TMTn reset asynchronously[Note])
1	TMTn operation enabled. TMTn operation start

TTnCKS2	TTnCKS1	TTnCKS0	Internal count clock selection
0	0	0	$f_{XX}/2$
0	0	1	$f_{XX}/4$
0	1	0	$f_{XX}/8$
0	1	1	$f_{XX}/32$
1	0	0	$f_{XX}/256$
1	0	1	$f_{XX}/1024$
1	1	0	$f_{XX}/2048$
1	1	1	$f_{XX}/4096$

Note The TTnOPT0.TTnOVF bit and the 16-bit counter are reset simultaneously. Moreover, timer outputs (TOTn0 and TOTn1 pins) are reset to the TTnIOC0 register set status at the same time as the 16-bit counter is reset.

Cautions 1. Set the TTnCKS2 to TTnCKS0 bits when the TTnCE bit = 0.
When the value of the TTnCE bit is changed from 0 to 1, the TTnCKS2 to TTnCKS0 bits can be set simultaneously.
2. Be sure to set bits 3 to 6 to "0".

Remark f_{XX}: Peripheral clock

When the TTnCE bit is set to 1, the value of the 16-bit counter is cleared from FFFFH to 0000H. The counter counts each time the valid edge of external event count input is detected. Additionally, the set value of the TTnCCR0 register is transferred to the CCR0 buffer register.

When the count value of the 16-bit counter matches the value of the CCR0 buffer register, the 16-bit counter is cleared to 0000H, and a compare match interrupt request signal (INTTTEQCn0) is generated.

The INTTTEQCn0 signal is generated each time the valid edge of the external event count input has been detected "value set to TTnCCR0 register + 1" times.

Figure 8-17. Register Setting for Operation in External Event Count Mode (1/2)

(a) TMTn control register 0 (TTnCTL0)

	TTnCE					TTnCKS2	TTnCKS1	TTnCKS0
TTnCTL0	0/1	0	0	0	0	0	0	0

0: Stop counting
1: Enable counting

(b) TMTn control register 1 (TTnCTL1)

		TTnEST	TTnEEE		TTnMD3	TTnMD2	TTnMD1	TTnMD0
TTnCTL1	0	0	0	0	0	0	0	1

0, 0, 0, 1:
External event count mode

(c) TMTn I/O control register 2 (TTnIOC2)

					TTnEES1	TTnEES0	TTnETS1	TTnETS0
TTnIOC2	0	0	0	0	0/1	0/1	0	0

Select valid edge
of external event
count input[Note]

Note TMT0, TMT1: EVTTm pin
TMT2, TMT3: TITk0 pin

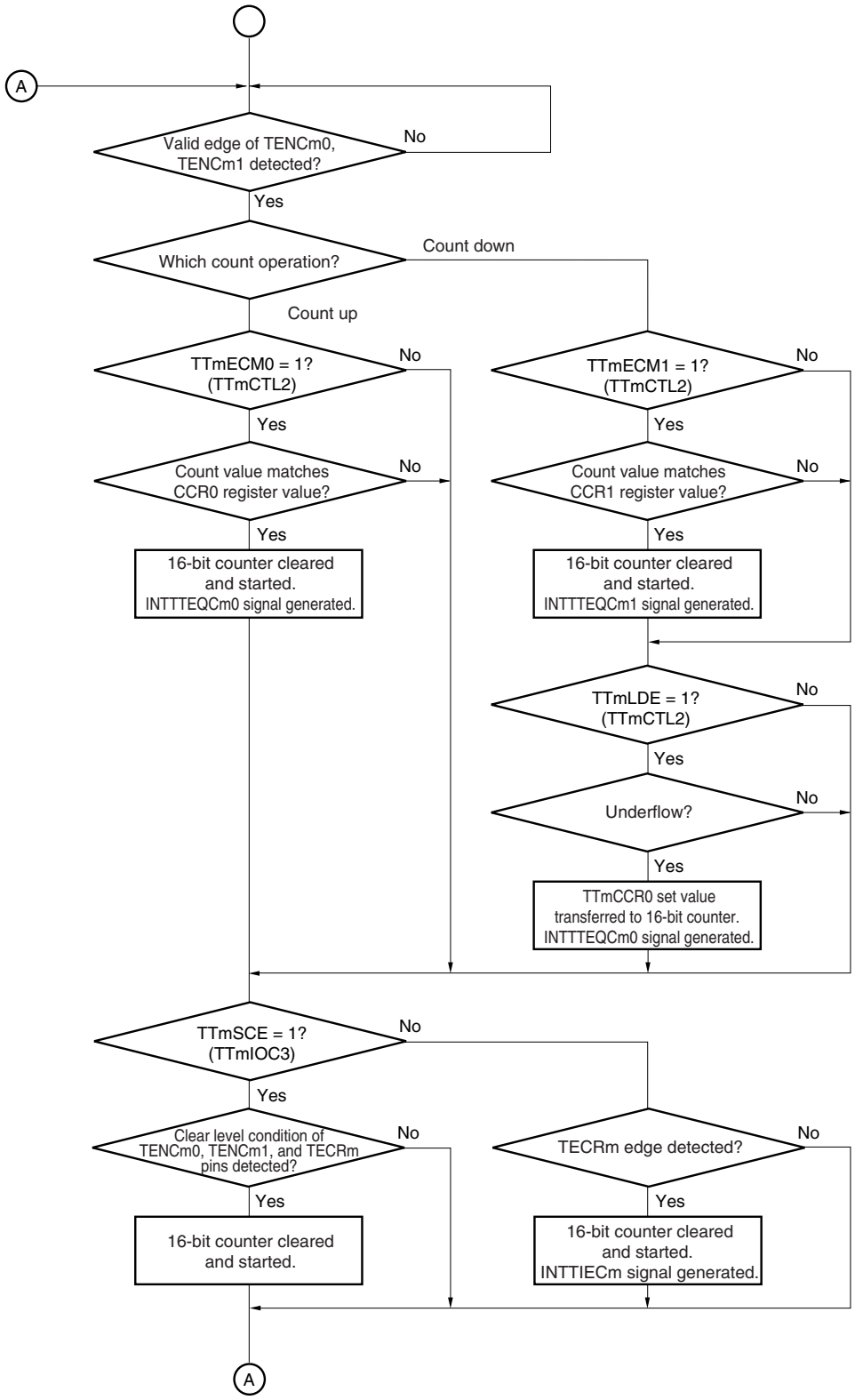
(d) TMTn counter read buffer register (TTnCNT)

The count value of the 16-bit counter can be read by reading the TTnCNT register.

(e) TMTn capture/compare register 0 (TTnCCR0)

If the TTnCCR0 register is set to D_0, the count is cleared when the number of external events has reached (D_0 + 1) and the compare match interrupt request signal (INTTTEQCn0) is generated.

Figure 8-65. Encoder Compare Mode Operation Processing

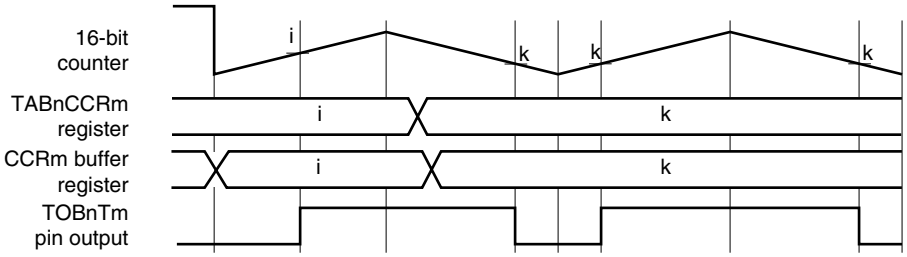


Remark m = 0, 1

Figure 10-24. Example of Rewriting TABnCCR1 to TABnCCR3 Registers (Rewriting Before Match Occurs)

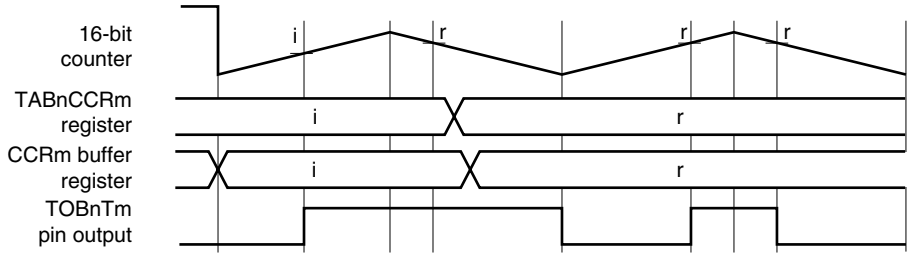
(a)

If the TABnCCRm register is rewritten before its value matches the value of the 16-bit counter, the register value will match the value of the 16-bit counter after the register has been rewritten. Consequently, the new register value is immediately reflected.



(b)

If a value less than the value of the 16-bit counter (greater if the counter is counting down) is written to the TABnCCRm register, the output waveform is as follows because the register value does not match the counter value.



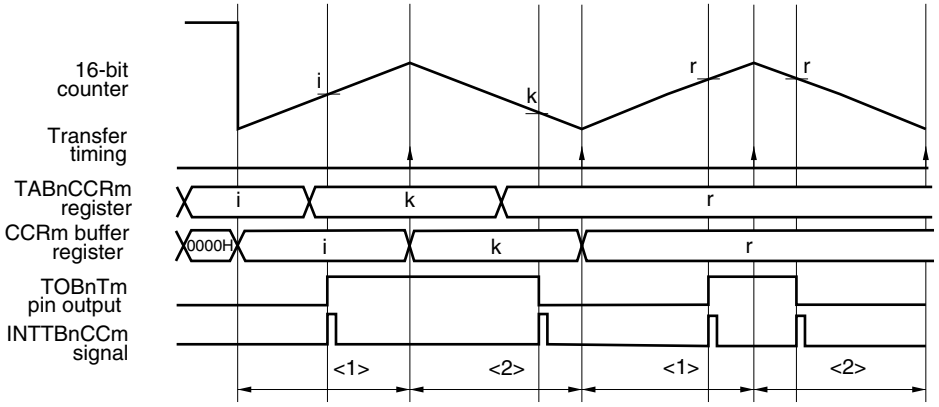
If the register value does not match the counter value, the TOBnTm pin output does not change. Even if the value of the 16-bit counter does not match the value of the TABnCCRm register, the TOBnTm pin output always changes to the high level if the crest interrupt occurs and to the low level if the valley interrupt occurs.
This is a function provided for 0% output and 100% output.
For details, see **10.4.2 (2) PWM output of 0%/100%**.

Remarks 1. i, r, k = Set values of TABnCCRm register

2. n = 0, 1, m = 1 to 3

(c) Rewriting TABnCCRm register

Figure 10-30. Example of Rewriting TABnCCRm Register



Rewriting during period <1> (rewriting during counting up)

Because the TABnCCRm register value is transferred at the transfer timing of the crest (match between the 16-bit counter value and TABnCCR0 register value), an asymmetrical triangular wave is output.

Rewriting during period <2> (rewriting during counting down)

Because the TABnCCRm register value is transferred at the transfer timing of the valley (match between the 16-bit counter value and 0001H), a symmetrical triangular wave is output.

Remark m = 1 to 3

(d) Transferring TABnOPT1 register value

Do not set the TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits to other than 00000. When using the interrupt culling function, rewrite the TABnOPT1 register in the intermittent batch rewrite mode (transfer culling mode).

For details of rewriting the TABnOPT1 register, see **10.4.3 Interrupt culling function**.

(2) Operation of multiple channel conversion

The signals of two or more analog input pins specified by the ADnCHEN register are sequentially converted, starting from the pin with the lowest number, using a signal specified by the ADnTSEL.ADnTRGSEL11 and ADnTSEL.ADnTRGSEL10 bits as a trigger. The result of conversion is stored in the ADnCRk register corresponding to the analog input pin.

When conversion of the signals of all the specified analog input pins is completed, an A/Dn conversion end interrupt request signal (INTADn) is generated. After completion of conversion, the A/D converter waits for the trigger with the ADnSCM.ADnCE bit remaining set to 1.

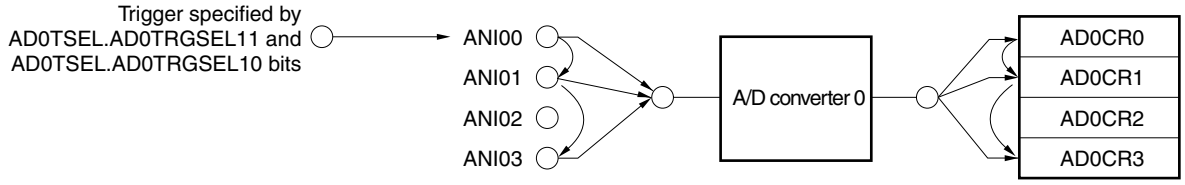
This operation is suitable for an application where two or more analog input signals should be monitored when the trigger is generated.

Analog Input Pin	A/D Conversion Result Register
ANInk[Note]	ADnCRk
⋮	⋮
ANInk[Note]	ADnCRk

Note Two or more can be specified by the ADnCHEN register.
However, A/D conversion is sequentially executed starting from the pin with the lowest number.

Remark A/D converter 0: n = 0
k = 0 to 3, 5 to 7
A/D converter 1: n = 1
V850E/IG4: k = 0 to 2, 5 to 7
V850E/IH4: k = 0 to 3, 5 to 7

Figure 12-19. Example of Multiple Channel Conversion Operation (Hardware Trigger Mode): A/D Converter 0



(1) AD0CE bit = 1 (enable)
(2) Trigger specified by AD0TSEL.AD0TRGSEL11 and AD0TSEL.AD0TRGSEL10 bits is generated
(3) Signal of ANI00 pin is A/D-converted
(4) Conversion result is stored in AD0CR0 register
(5) Signal of ANI01 pin is A/D-converted
(6) Conversion result is stored in AD0CR1 register
(7) Signal of ANI03 pin is A/D-converted
(8) Conversion result is stored in AD0CR3 register
(9) AD0SCM.AD0CS bit = 0
(10) INTAD0 interrupt request signal is generated
(11) Returns to (3) when the next trigger is input
(12) Set AD0CE bit to 0 to end (stop)

Remark This is an operation example when the AD0SCM.AD0PLM, AD0TRG1, and AD0TRG0 bits = 001, AD0CTL0.AD0MD1 and AD0CTL0.AD0MD0 bits = 00, and AD0CHEN register = 000BH.

12.6 Cautions

12.6.1 Stopping conversion operation

The ongoing conversion operation is stopped when 0 is written to the ADnSCM.ADnCE bit. At this time, the conversion result in the A/Dn conversion result register m (ADnCRm) and A/Dn conversion result extension register a (ADnECRa) is undefined. Therefore, read the A/D conversion result after A/D conversion has been completed (after the A/Dn conversion end interrupt request signal (INTADn) has been issued), and then write 0 to the ADnCE bit as necessary.

Note that the ADnCE bit is not cleared to 0 in all the modes even after the INTADn signal is generated.

Remark n = 0, 1
m = 0 to 15

12.6.2 Interval of trigger during conversion operation in hardware trigger mode, conversion channel specification mode, and extension buffer mode

Inputting a trigger during conversion operation is ignored in the hardware trigger mode, conversion channel specification mode, and extension buffer mode. Therefore, the interval of the trigger (input time) in the hardware trigger mode, conversion channel specification mode, and extension buffer mode must be longer than the A/D conversion time specified by the ADnCTC.ADnFR3 to ADnCTC.ADnFR0 bits (see **Table 12-2 Number of A/D Conversion Clocks and A/D Conversion Time**).

Remark n = 0, 1

12.6.3 Writing to ADnSCM register

(1) Restarting A/D conversion

To restart A/D conversion, write the same value to the ADnSCM register. To change the ADnPLM, ADnTRG1, and ADnTRG0 bits, be sure to set the ADnCE bit to 0.

(2) Contention between end of A/D conversion and writing to ADnSCM register

If completion of A/D conversion contends with writing to the ADnSCM register during A/D conversion operation, the conversion result is correctly stored in the ADnCRm and ADnECRa registers, if the A/Dn conversion end interrupt request signal (INTADn) is generated. If the INTADn signal is not generated, the A/D conversion operation is aborted. Therefore, the previous conversion result is held by the ADnCRm and ADnECRa registers.

(3) Successive writing to ADnSCM register

To successively write the ADnSCM register when the conversion operation is enabled (ADnCE bit = 1), be sure to wait for time of at least 5 base clocks (f_{AD01}).

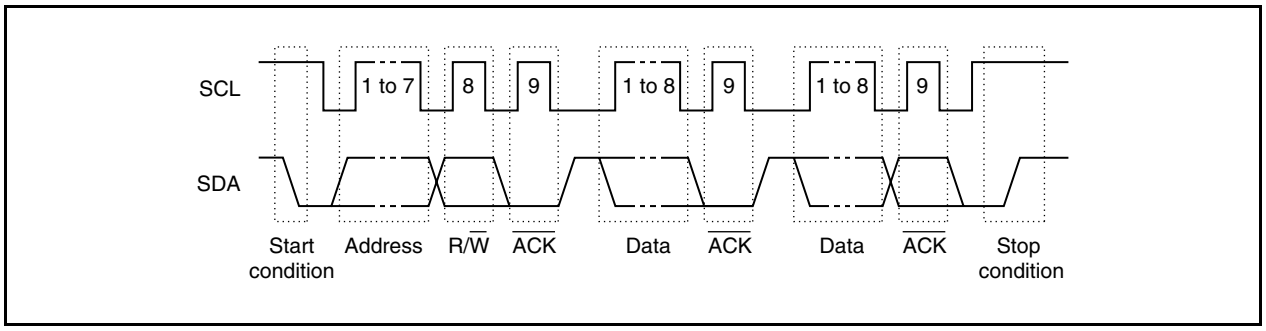
The ADnSCM register can be successively written when the ADnCE bit is set to 1 after the ADnSCM register is written while the ADnCE bit = 0.

Remark n = 0, 1

17.6 I²C Bus Definitions and Control Methods

The following section describes the I²C bus's serial data communication format and the status generated by the I²C bus. The transfer timing for the "start condition", "address", "transfer direction specification", "data", and "stop condition" generated via the I²C bus's serial data bus is shown below.

Figure 17-5. I²C Bus's Serial Data Transfer Timing



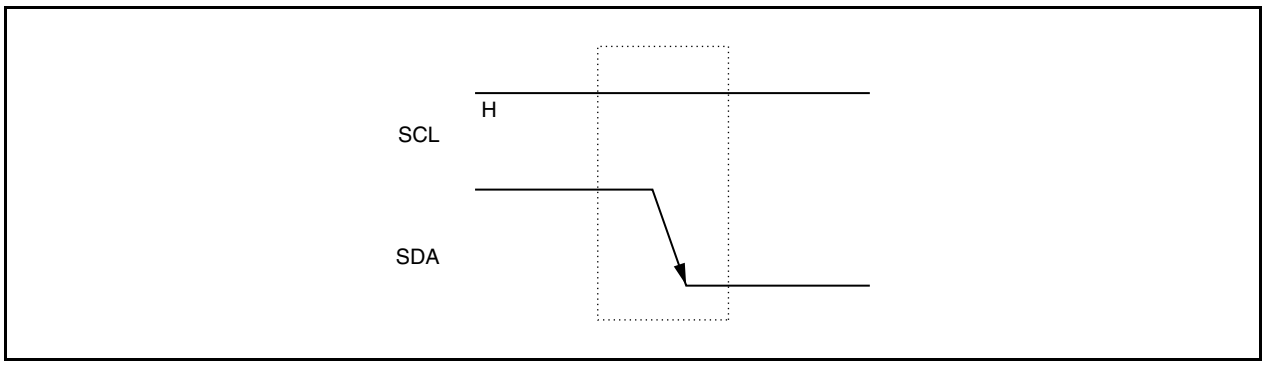
The master device generates the start condition, slave address, and stop condition.
ACK can be generated by either the master or slave device (normally, it is generated by the device that receives 8-bit data).

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

17.6.1 Start condition

A start condition is met when the SCL pin is at high level and the SDA pin changes from high level to low level. The start conditions for the SCL pin and SDA pin are generated when the master device starts a serial transfer to the slave device. Start conditions can be detected when the device is used as a slave.

Figure 17-6. Start Conditions



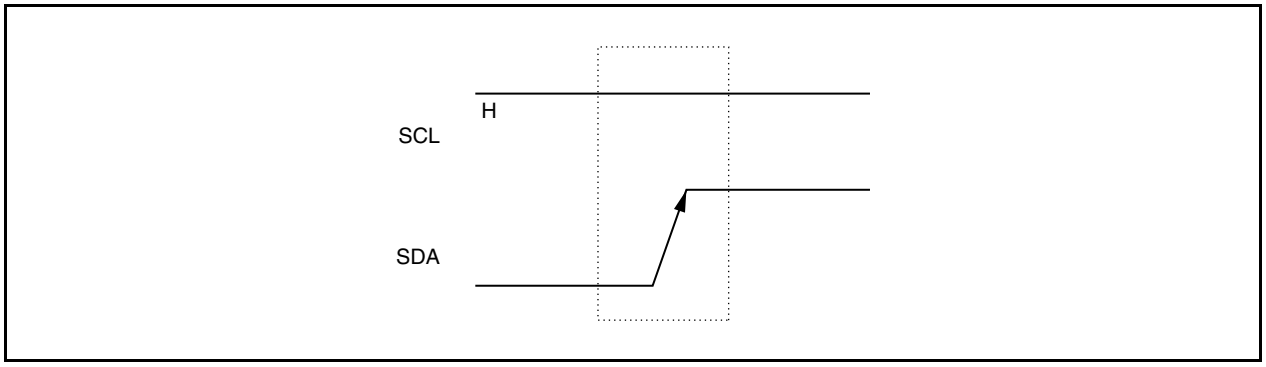
A start condition is generated when the IICC0.STT0 bit is set to 1 after a stop condition has been detected (IICS0.SPD0 bit = 1). When a start condition is detected, IICS0.STD0 bit is set to 1.

17.6.5 Stop condition

When the SCL pin is at high level, changing the SDA pin from low level to high level generates a stop condition.

A stop condition is generated when serial transfer from the master device to the slave device has been completed. Stop conditions can be detected when the device is used as a slave.

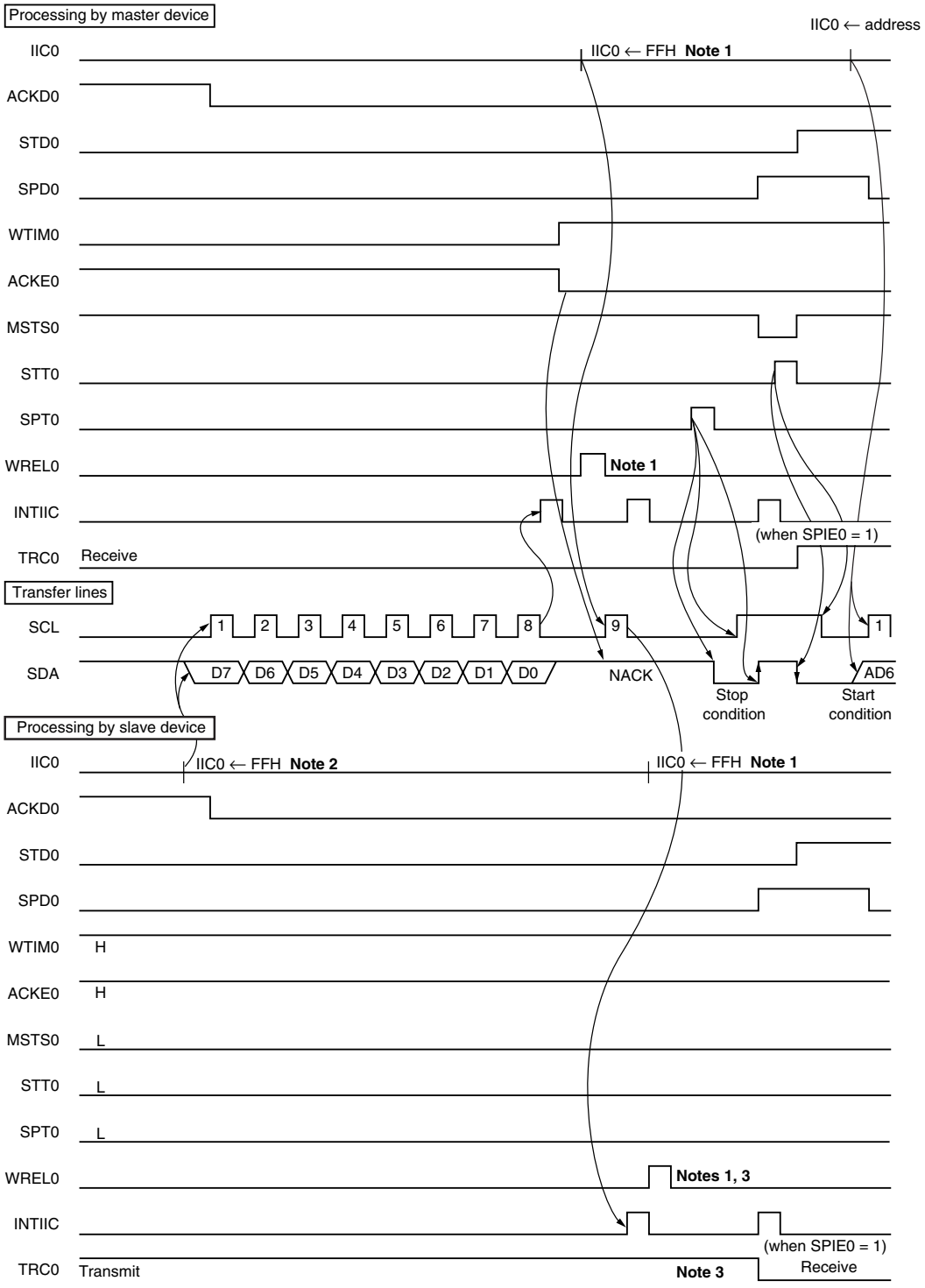
Figure 17-10. Stop Condition



A stop condition is generated when the IICC0.SPT0 bit is set to 1. When the stop condition is detected, the IICS0.SPD0 bit is set to 1 and the interrupt request signal (INTIIC) is generated when the IICC0.SPIE0 bit is set to 1.

Figure 17-22. Example of Slave to Master Communication
(When 8-Clock → 9-Clock Wait for Master and 9-Clock Wait for Slave Are Selected) (3/3)

(c) Stop condition



Notes 1. To cancel master wait, write FFH to IIC0 or set WREL0.

2. Cancel waits during slave transmission by writing data to IIC0, not by setting WREL0.

3. TRC0 is cleared if waits during slave transmission are canceled by setting WREL0.

19.9 Periods in Which CPU Does Not Acknowledge Interrupts

An interrupt is acknowledged by the CPU while an instruction is being executed. However, no interrupt will be acknowledged between an interrupt request non-sample instruction and the next instruction (the interrupt is held pending).

The interrupt request non-sample instructions are as follows.

- EI instruction
- DI instruction
- LDSR reg2, 0x5 instruction (for PSW)
- Store instruction for the command register (PRCMD).
- Store instructions or bit manipulation instructions excluding tst1 instruction for the following registers.
  - Interrupt-related registers:
    Interrupt control register (xxICn) and interrupt mask registers 0 to 6 (IMR0 to IMR6)
  - Power save control register (PSC)

Remark xx: Identification name of each peripheral unit (see **Table 19-2**)
n: Peripheral unit number (see **Table 19-2**)

19.10 Caution

Note that if a port is set to external interrupt input (INTP00 to INTP19, INTADT0, and INTADT1), the timer/counter-related interrupt, serial interface-related interrupt, and A/D converter-related interrupt, which are alternate functions, do not occur.

(2) Releasing HALT mode by $\overline{\text{RESET}}$ pin input or by WDTRES, LVIRES, or POCRES signal generation

The same operation as the normal reset operation is performed.

Table 20-3. Operation Status in HALT Mode

Setting of HALT Mode / Item		Operation Status
Clock generator, PLL		Operates
System clock (f_{XX})		Supply
CPU		Stops operation
DMA		Operable
Interrupt controller		Operable
Timer	TAA0 to TAA2	Operable
	TAB0, TAB1	Operable
	TMT0 to TMT3	Operable
	TMM0 to TMM3	Operable
Watchdog timer		Operable
Serial interface	CSIF0 to CSIF2	Operable
	UARTA0 to UARTA2	Operable
	UARTB	Operable
	I^2C	Operable
A/D converters 0 to 2		Operable
Clock monitor		Operable
Low-voltage detector		Operable
Power-on-clear circuit		Operable
Port function		Retains status before HALT mode was set.
Internal data		The CPU registers, statuses, data, and all other internal data such as the contents of the internal RAM are retained as they were before the HALT mode was set.

CHAPTER 22 LOW-VOLTAGE DETECTOR

22.1 Functions

The low-voltage detector (LVI) has the following functions.

- Compares the supply voltage (V850E/IG4: EV_{DD0}, EV_{DD1}, EV_{DD2}, V850E/IH4: FV_{DD}) and detection voltage (V_{LVI}) and generates an interrupt request signal (INTLVIL, INTLVIH) or internal reset signal (LVIRES) when the supply voltage drops below the detection voltage.
- The level of the supply voltage to be detected can be changed by software (in two steps).
- An interrupt request signal (INTLVIL, INTLVIH) or internal reset signal (LVIRES) can be selected.
- Can operate in STOP mode.
- Operation can be stopped by software.

If the low-voltage detector is used to generate a reset signal, the RESF.LVIRF bit is set to 1 when the reset signal is generated. For details of RESF register, see **CHAPTER 21 RESET FUNCTIONS**.

- Port registers when CSIF0 is used

  When CSIF0 is used, port registers are set to make the SIF0, SOF0, $\overline{\text{SCKF0}}$, and HS (P44) pins valid by the debug monitor program. Do not change the following register settings with the user program during debugging. (The same value can be overwritten.)

(a) SIF0, SOF0, and $\overline{\text{SCKF0}}$ settings

	7	6	5	4	3	2	1	0
PFCE4	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
PFC4	0	0	0	0	×	0	0	0

	7	6	5	4	3	2	1	0
PMC4	0	0	0	×	×	1	1	1

(b) HS (P44 pin) settings

	7	6	5	4	3	2	1	0
PMC4	0	0	0	0	×	×	×	×

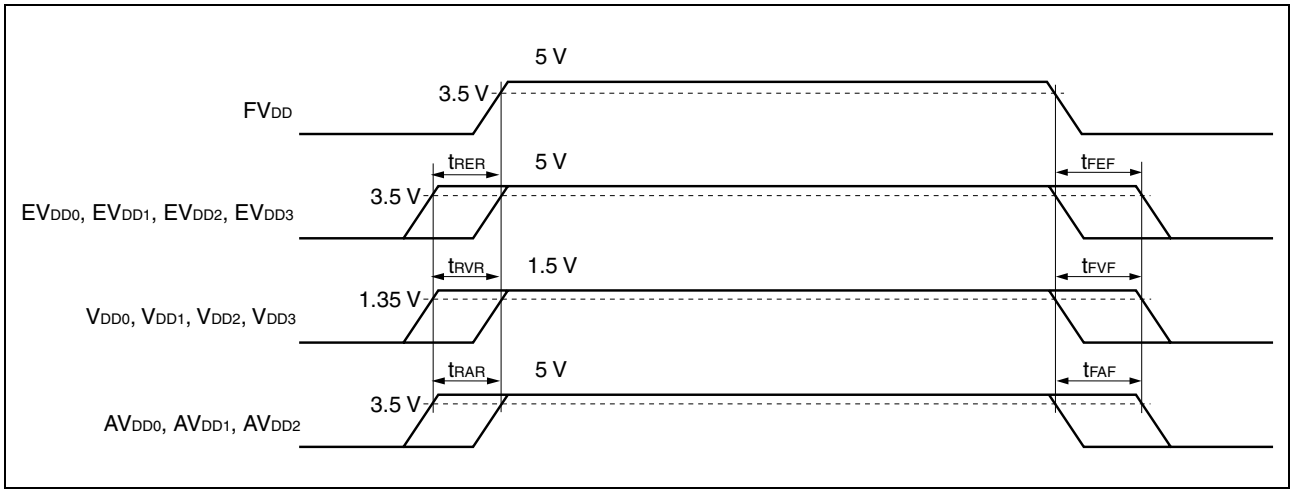
	7	6	5	4	3	2	1	0
PM4	0	0	0	0	×	×	×	×

	7	6	5	4	3	2	1	0
P4	0	0	0	**Note**	×	×	×	×

Note Writing to this bit is prohibited.
The values corresponding to the HS pin are changed by the monitor program according to the debugger status. To perform port register settings in 8-bit units, the user program can usually use read-modify-write. If an interrupt for debugging occurs before writing, however, an unexpected operation may be performed.

Remark ×: 0 or 1

(b) Internal $\overline{\text{RESET}}$ (recommended conditions)



(3/6)

Mnemonic	Operand	Opcode	Operation		Execution Clock			Flags				
					i	r	l	CY	OV	S	Z	SAT
LD.H	disp16[reg1],reg2	rrrrr111001RRRRR ddddddddddddddd0 **Note 8**	adr←GR[reg1]+sign-extend(disp16) GR[reg2]←sign-extend(Load-memory(adr,Halfword))		1	1	Note 11					
LDSR	reg2,regID	rrrrr111111RRRRR 0000000000100000 **Note 12**	SR[regID]←GR[reg2]	Other than regID = PSW	1	1	1					
				regID = PSW	1	1	1	×	×	×	×	×
LD.HU	disp16[reg1],reg2	rrrrr111111RRRRR ddddddddddddddd1 **Note 8**	adr←GR[reg1]+sign-extend(disp16) GR[reg2]←zero-extend(Load-memory(adr,Halfword)		1	1	Note 11					
LD.W	disp16[reg1],reg2	rrrrr111001RRRRR ddddddddddddddd1 **Note 8**	adr←GR[reg1]+sign-extend(disp16) GR[reg2]←Load-memory(adr,Word)		1	1	Note 11					
MOV	reg1,reg2	rrrrr000000RRRRR	GR[reg2]←GR[reg1]		1	1	1					
	imm5,reg2	r r r r r 0 1 0 0 0 0 i i i i i	GR[reg2]←sign-extend(imm5)		1	1	1					
	imm32,reg1	00000110001RRRRR i i i i i i i i i i i i i i i i l l l l l l l l l l l l l l l l	GR[reg1]←imm32		2	2	2					
MOVEA	imm16,reg1,reg2	rrrrr110001RRRRR i i i i i i i i i i i i i i i i	GR[reg2]←GR[reg1]+sign-extend(imm16)		1	1	1					
MOVHI	imm16,reg1,reg2	rrrrr110010RRRRR i i i i i i i i i i i i i i i i	GR[reg2]←GR[reg1]+(imm16 ll 0^{16})		1	1	1					
MUL Note 22	reg1,reg2,reg3	rrrrr111111RRRRR wwwww01000100000	GR[reg3] ll GR[reg2]←GR[reg2]xGR[reg1]		1	2 Note 14	2					
	imm9,reg2,reg3	r r r r r 1 1 1 1 1 i i i i i w w w w w 0 1 0 0 1 l l l l 0 0 **Note 13**	GR[reg3] ll GR[reg2]←GR[reg2]xsign-extend(imm9)		1	2 Note 14	2					
MULH	reg1,reg2	rrrrr000111RRRRR	GR[reg2]←GR[reg2] Note 6 xGR[reg1] Note 6		1	1	2					
	imm5,reg2	r r r r r 0 1 0 1 1 1 i i i i i	GR[reg2]←GR[reg2] Note 6 xsign-extend(imm5)		1	1	2					
MULHI	imm16,reg1,reg2	rrrrr110111RRRRR i i i i i i i i i i i i i i i i	GR[reg2]←GR[reg1] Note 6 ximm16		1	1	2					
MULU Note 22	reg1,reg2,reg3	rrrrr111111RRRRR wwwww01000100010	GR[reg3] ll GR[reg2]←GR[reg2]xGR[reg1]		1	2 Note 14	2					
	imm9,reg2,reg3	r r r r r 1 1 1 1 1 i i i i i w w w w w 0 1 0 0 1 l l l l 1 0 **Note 13**	GR[reg3] ll GR[reg2]←GR[reg2]xzero-extend(imm9)		1	2 Note 14	2					
NOP		0000000000000000	Pass at least one clock cycle doing nothing.		1	1	1					
NOT	reg1,reg2	rrrrr000001RRRRR	GR[reg2]←NOT(GR[reg1])		1	1	1		0	×	×	
NOT1	bit#3,disp16[reg1]	01bbb111110RRRRR ddddddddddddddd	adr←GR[reg1]+sign-extend(disp16) Z flag←Not(Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,Z flag)		3 Note 3	3 Note 3	3 Note 3				×	
	reg2,[reg1]	rrrrr111111RRRRR 0000000011100010	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,Z flag)		3 Note 3	3 Note 3	3 Note 3				×	