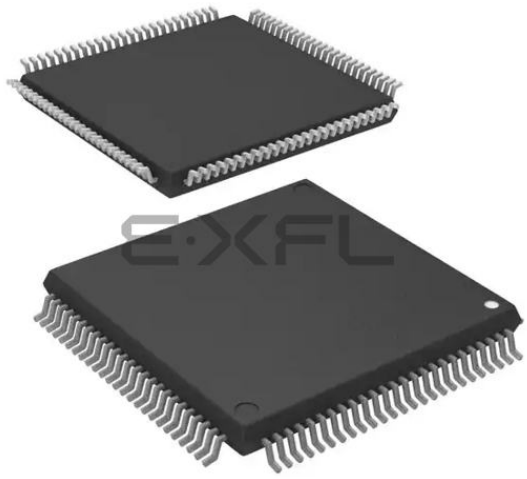
Renesas Electronics America Inc - UPD70F3796GC-UEU-AX Datasheet



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	V850ES
Core Size	32-Bit Single-Core
Speed	20MHz
Connectivity	CSI, EBI/EMI, I²C, UART/USART, USB
Peripherals	DMA, LVD, PWM, WDT
Number of I/O	80
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	40K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 12x10b; D/A 2x8b
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/upd70f3796gc-ueu-ax

(10/11)

Address	Function Register Name	Symbol	R/W	Manipulatable Bits			Default Value
				1	8	16	
FFFFFADDH	RTC control register 0	RC1CC0	R/W	√	√		00H
FFFFFADEH	RTC control register 1	RC1CC1		√	√		00H
FFFFFADFH	RTC control register 2	RC1CC2		√	√		00H
FFFFFAE0H	RTC control register 3	RC1CC3		√	√		00H
FFFFFB00H	RTC backup control register 0	RTCBUMCTL0 Note		√	√		00H
FFFFFB03H	Subclock low-power operation control register	SOSCAMCTL Note		√	√		00H
FFFFFC00H	External interrupt falling edge specification register 0	INTF0		√	√		00H
FFFFFC06H	External interrupt falling edge specification register 3	INTF3		√	√		00H
FFFFFC13H	External interrupt falling edge specification register 9H	INTF9H		√	√		00H
FFFFFC20H	External interrupt rising edge specification register 0	INTR0		√	√		00H
FFFFFC26H	External interrupt rising edge specification register 3	INTR3		√	√		00H
FFFFFC33H	External interrupt rising edge specification register 9H	INTR9H		√	√		00H
FFFFFC60H	Port 0 function register	PF0		√	√		00H
FFFFFC66H	Port 3 function register	PF3				√	0000H
FFFFFC66H	Port 3 function register L	PF3L		√	√		00H
FFFFFC67H	Port 3 function register H	PF3H		√	√		00H
FFFFFC68H	Port 4 function register	PF4		√	√		00H
FFFFFC6AH	Port 5 function register	PF5		√	√		00H
FFFFFC72H	Port 9 function register	PF9				√	0000H
FFFFFC72H	Port 9 function register L	PF9L		√	√		00H
FFFFFC73H	Port 9 function register H	PF9H		√	√		00H
FFFFFD00H	CSIB0 control register 0	CB0CTL0		√	√		01H
FFFFFD01H	CSIB0 control register 1	CB0CTL1		√	√		00H
FFFFFD02H	CSIB0 control register 2	CB0CTL2			√		00H
FFFFFD03H	CSIB0 status register	CB0STR		√	√		00H
FFFFFD04H	CSIB0 receive data register	CB0RX	R			√	0000H
FFFFFD04H	CSIB0 receive data register L	CB0RXL			√		00H
FFFFFD06H	CSIB0 transmit data register	CB0TX	R/W			√	0000H
FFFFFD06H	CSIB0 transmit data register L	CB0TXL			√		00H
FFFFFD10H	CSIB1 control register 0	CB1CTL0		√	√		01H
FFFFFD11H	CSIB1 control register 1	CB1CTL1		√	√		00H
FFFFFD12H	CSIB1 control register 2	CB1CTL2			√		00H
FFFFFD13H	CSIB1 status register	CB1STR		√	√		00H
FFFFFD14H	CSIB1 receive data register	CB1RX	R			√	0000H
FFFFFD14H	CSIB1 receive data register L	CB1RXL			√		00H
FFFFFD16H	CSIB1 transmit data register	CB1TX	R/W			√	0000H
FFFFFD16H	CSIB1 transmit data register L	CB1TXL			√		00H
FFFFFD20H	CSIB2 control register 0	CB2CTL0		√	√		01H
FFFFFD21H	CSIB2 control register 1	CB2CTL1		√	√		00H
FFFFFD22H	CSIB2 control register 2	CB2CTL2			√		00H
FFFFFD23H	CSIB2 status register	CB2STR		√	√		00H
FFFFFD24H	CSIB2 receive data register	CB2RX	R			√	0000H
FFFFFD24H	CSIB2 receive data register L	CB2RXL			√		00H

Note This is a special register.

(2) Conflict between sld instruction and interrupt request

(a) Description

If a conflict occurs between the decode operation of an instruction in <2> immediately before the sld instruction following an instruction in <1> and an interrupt request before the instruction in <1> is complete, the execution result of the instruction in <1> may not be stored in a register.

Instruction <1>

- ld instruction: ld.b, ld.h, ld.w, ld.bu, ld.hu
- sld instruction: sld.b, sld.h, sld.w, sld.bu, sld.hu
- Multiplication instruction: mul, mulh, mulhi, mulu

Instruction <2>

mov reg1, reg2	not reg1, reg2	satsubr reg1, reg2	satsub reg1, reg2
satadd reg1, reg2	satadd imm5, reg2	or reg1, reg2	xor reg1, reg2
and reg1, reg2	tst reg1, reg2	subr reg1, reg2	sub reg1, reg2
add reg1, reg2	add imm5, reg2	cmp reg1, reg2	cmp imm5, reg2
mulh reg1, reg2	shr imm5, reg2	sar imm5, reg2	shl imm5, reg2

<Example>

<i> ld.w [r11], r10
•
•
•
<ii> mov r10, r28
<iii> sld.w 0x28, r10

If the decode operation of the mov instruction <ii> immediately before the sld instruction <iii> and an interrupt request conflict before execution of the ld instruction <i> is complete, the execution result of instruction <i> may not be stored in a register.

(b) Countermeasure

<1> When compiler (CA850) is used

Use CA850 Ver. 2.61 or later because generation of the corresponding instruction sequence can be automatically suppressed.

<2> For assembler

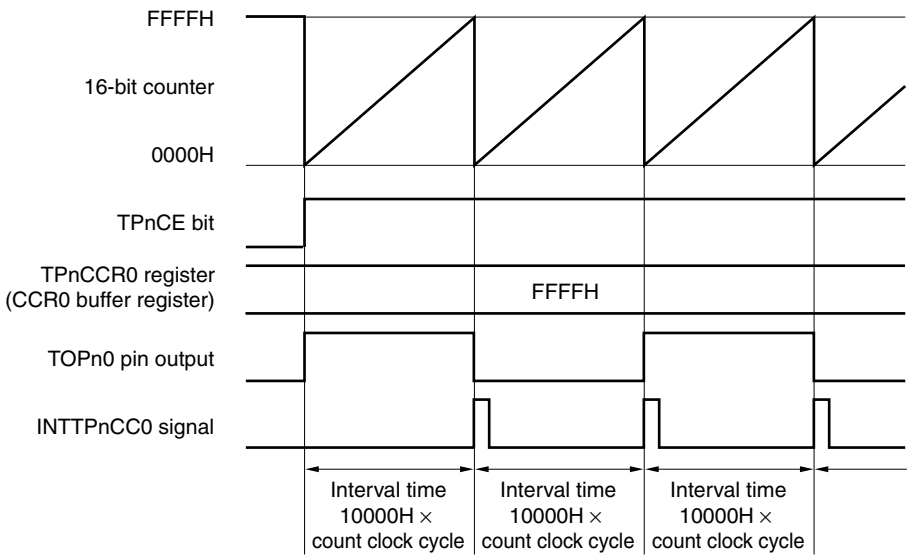
When executing the sld instruction immediately after instruction <ii>, avoid the above operation using either of the following methods.

- Insert a nop instruction immediately before the sld instruction.
- Do not use the same register as the sld instruction destination register in the above instruction <ii> executed immediately before the sld instruction.

(b) Operation when TPnCCR0 register is set to FFFFH

When the TPnCCR0 register is set to FFFFH, the 16-bit counter increments up to FFFFH and is reset to 0000H in synchronization with the next increment timing. The INTTPnCC0 signal is then generated and the output of the TOPn0 pin is inverted. At this time, an overflow interrupt request signal (INTTPnOV) is not generated, nor is the overflow flag (TPnOPT0.TPnOVF bit) set to 1.

Figure 7-11. Operation of Interval Timer When TPnCCR0 Register Is Set to FFFFH



(b) Outputting a 0% or 100% PWM waveform

To output a 0% waveform, clear the TPnCCR1 register to 0000H.

Figure 7-49. Outputting 0% PWM Waveform

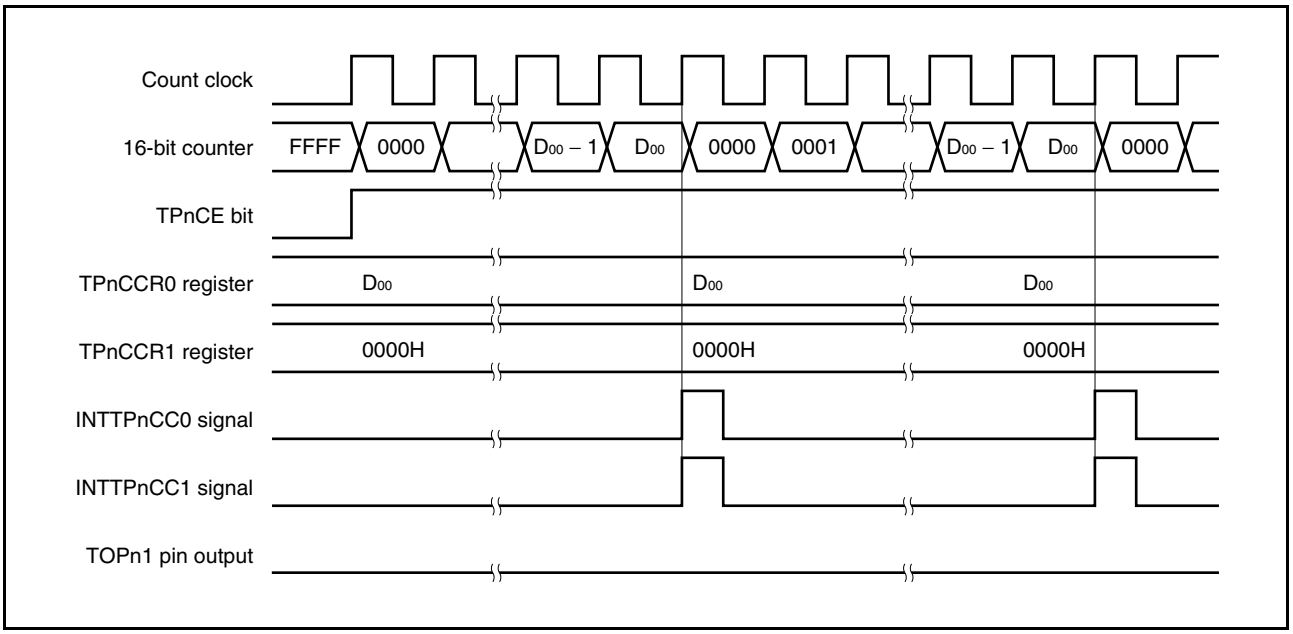
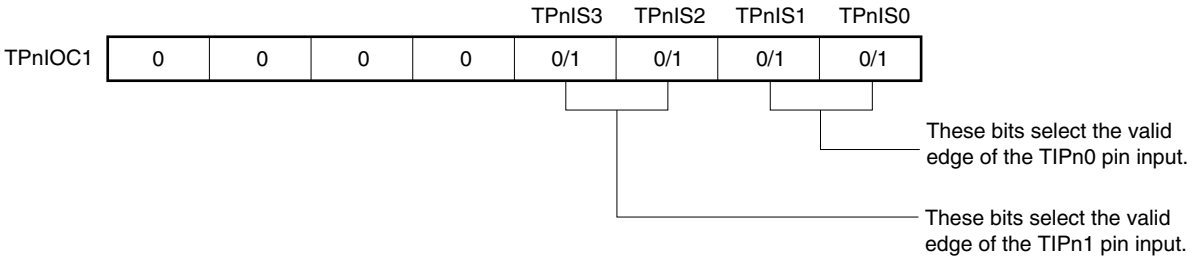
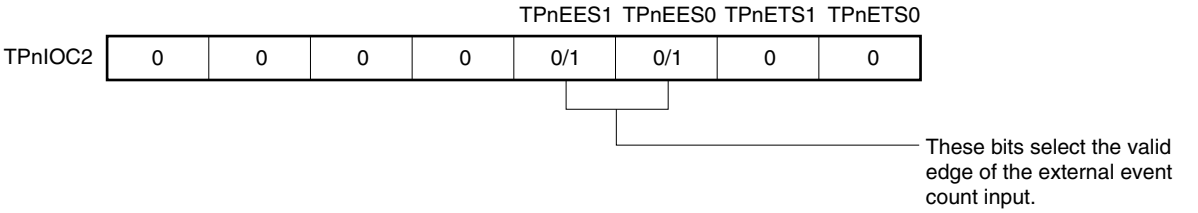


Figure 7-55. Register Settings in Free-Running Timer Mode (2/2)

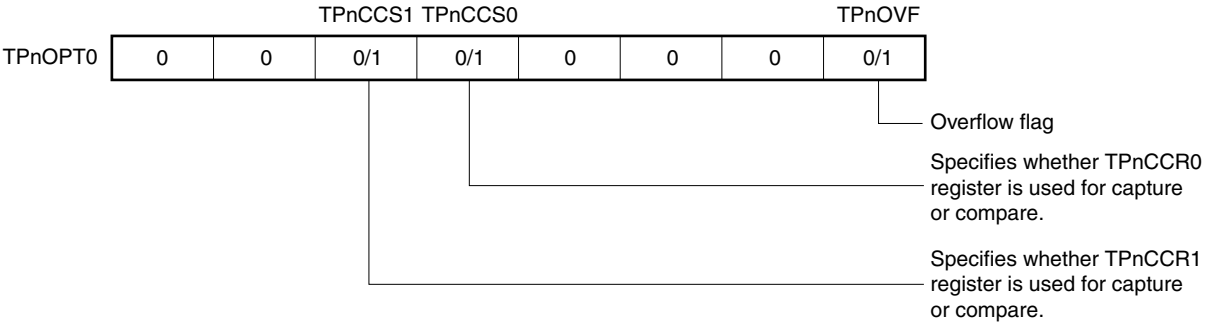
(d) TMPn I/O control register 1 (TPnIOC1)



(e) TMPn I/O control register 2 (TPnIOC2)



(f) TMPn option register 0 (TPnOPT0)



(g) TMPn counter read buffer register (TPnCNT)

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

(h) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

These registers function as capture registers or compare registers according to the setting of the TPnOPT0.TPnCCSa bit.

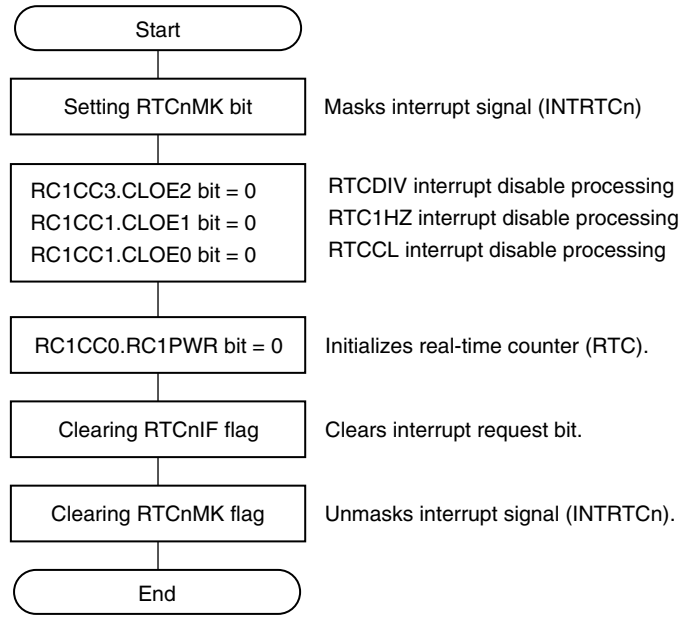
When the registers function as capture registers, they store the value of the 16-bit counter when it is detected that a valid edge has been input to the TIPna pin, after which the INTTPnCCa signal is generated.

When the registers function as compare registers and when the TPnCCRa register is set to Da, the INTTPnCCa signal is generated the when the counter reaches (D_a + 1), and the output signal of the TOPna pin is inverted.

11.4.8 Initializing real-time counter

The procedure for initializing the real-time counter is shown below.

Figure 11-9. Initializing Real-Time Counter



Remarks 1. See **22.3.4 Interrupt control register (xxICn)** for details of the RTCnIF and RTCnMK bits.

2. n = 0 to 2

CHAPTER 13 REAL-TIME OUTPUT FUNCTION (RTO)

13.1 Function

The real-time output function transfers preset data to the real-time output buffer registers (RTBL0 and RTBH0), and then transfers this data by hardware to an external device via the output latches, upon occurrence of a timer interrupt. The pins through which the data is output to an external device constitute a port called the real-time output function (RTO).

Because RTO can output signals without jitter, it is suitable for controlling a stepper motor.

In the V850ES/JG3-L, one 6-bit real-time output port channel is provided.

The real-time output port can be set to the port mode or real-time output port mode in 1-bit units.

Figure 16-8. Continuous Transmission Operation Timing

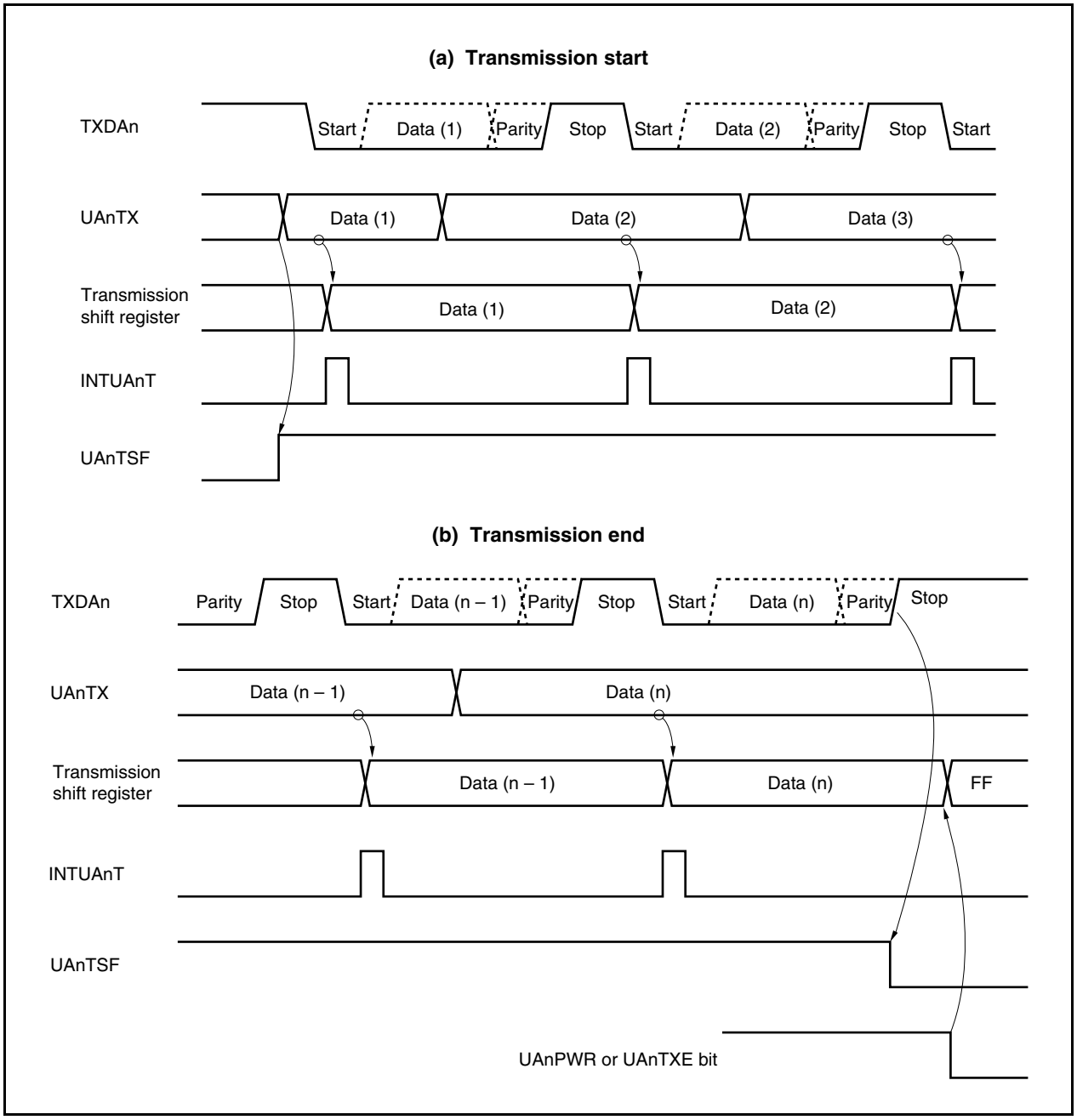
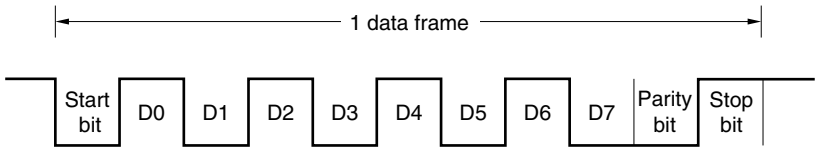
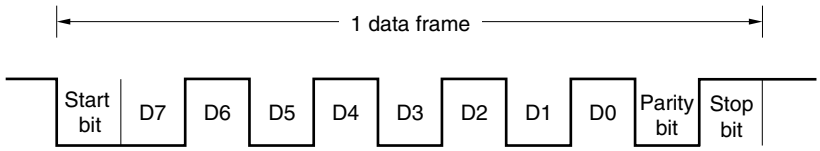


Figure 17-3. UARTC Transmit/Receive Data Format

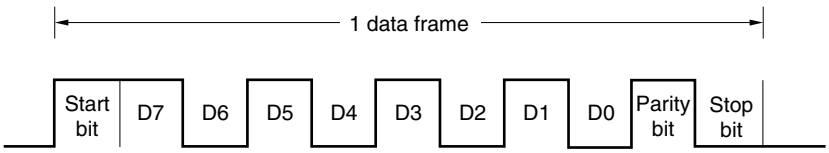
(a) 8-bit data length, LSB first, even parity, 1 stop bit, transfer data: 55H



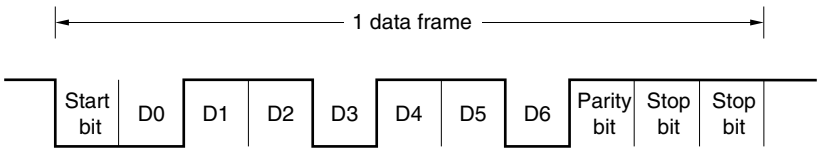
(b) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H



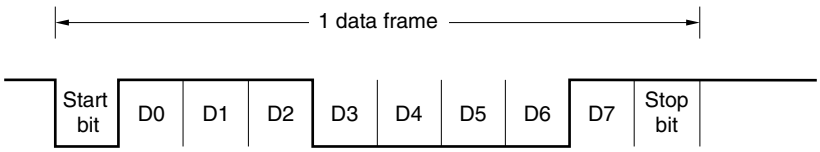
(c) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H, transmit/receive data inverted



(d) 7-bit data length, LSB first, odd parity, 2 stop bits, transfer data: 36H



(e) 8-bit data length, LSB first, no parity, 1 stop bit, transfer data: 87H



17.6.9 SBF reception

The reception enabled status is entered by setting the UC0CTL0.UC0PWR bit to 1 and then setting the UC0CTL0.UC0RXE bit to 1.

The SBF reception wait status is set by setting the SBF reception trigger (UC0OPT0.UC0STR bit) to 1.

In the SBF reception wait status, similarly to the UART reception wait status, the RXDC0 pin is monitored and start bit detection is performed.

Following detection of the start bit, reception is started and the internal counter increments according to the set baud rate.

When a stop bit is received, if the SBF width is 11 or more bits, it is judged as normal processing and a reception complete interrupt request signal (INTUC0R) is output. The UC0OPT0.UC0SRF bit is automatically cleared and SBF reception ends. Error detection for the UC0STR.UC0OVE, UC0STR.UC0PE, and UC0STR.UC0FE bits is suppressed and UART communication error detection processing is not performed. Moreover, data transfer of the UARTC0 reception shift register and UC0RX register is not performed and FFH, the initial value, is held. If the SBF width is 10 or fewer bits, reception is terminated as an error, an interrupt is not generated, and the SBF reception mode is restored. The UC0SRF bit is not cleared at this time.

Cautions 1. If SBF is transmitted during data reception, a framing error occurs.

2. Do not set the SBF reception trigger bit (UC0SRT) and SBF transmission trigger bit (UC0STT) to 1 during SBF reception (UC0SRF = 1).

Figure 17-12. SBF Reception

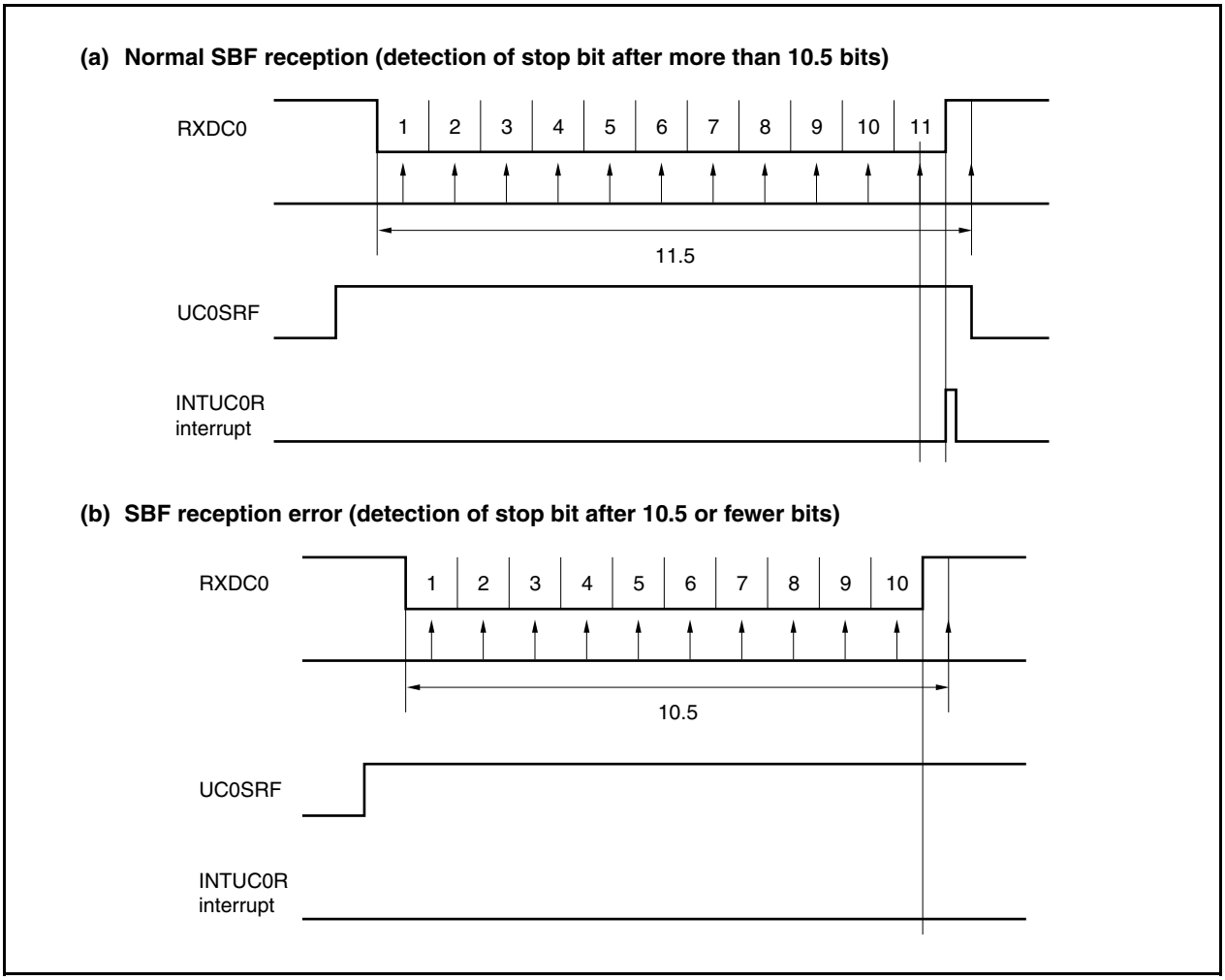


Table 19-6. Wait Periods

Clock Selection	CLXn	SMCn	CLn1	CLn0	Wait Period
fxx (when OCKSm = 18H set)	0	0	0	0	26 clocks
fxx/2 (when OCKSm = 10H set)	0	0	0	0	52 clocks
fxx/3 (when OCKSm = 11H set)	0	0	0	0	78 clocks
fxx/4 (when OCKSm = 12H set)	0	0	0	0	104 clocks
fxx/5 (when OCKSm = 13H set)	0	0	0	0	130 clocks
fxx (when OCKSm = 18H set)	0	0	0	1	47 clocks
fxx/2 (when OCKSm = 10H set)	0	0	0	1	94 clocks
fxx/3 (when OCKSm = 11H set)	0	0	0	1	141 clocks
fxx/4 (when OCKSm = 12H set)	0	0	0	1	188 clocks
fxx	0	0	1	0	47 clocks
fxx (when OCKSm = 18H set)	0	0	1	1	37 clocks
fxx/2 (when OCKSm = 10H set)	0	0	1	1	74 clocks
fxx/3 (when OCKSm = 11H set)	0	0	1	1	111 clocks
fxx (when OCKSm = 18H set)	0	1	0	×	16 clocks
fxx/2 (when OCKSm = 10H set)	0	1	0	×	32 clocks
fxx/3 (when OCKSm = 11H set)	0	1	0	×	48 clocks
fxx/4 (when OCKSm = 12H set)	0	1	0	×	64 clocks
fxx	0	1	1	0	16 clocks
fxx (when OCKSm = 18H set)	0	1	1	1	13 clocks
fxx/2 (when OCKSm = 10H set)	0	1	1	1	26 clocks
fxx/3 (when OCKSm = 11H set)	0	1	1	1	39 clocks
fxx (when OCKSm = 18H set)	1	1	0	×	10 clocks
fxx/2 (when OCKSm = 10H set)	1	1	0	×	20 clocks
fxx/3 (when OCKSm = 11H set)	1	1	0	×	30 clocks
fxx/4 (when OCKSm = 12H set)	1	1	0	×	40 clocks
fxx/5 (when OCKSm = 13H set)	1	1	0	×	50 clocks
fxx	1	1	1	0	10 clocks

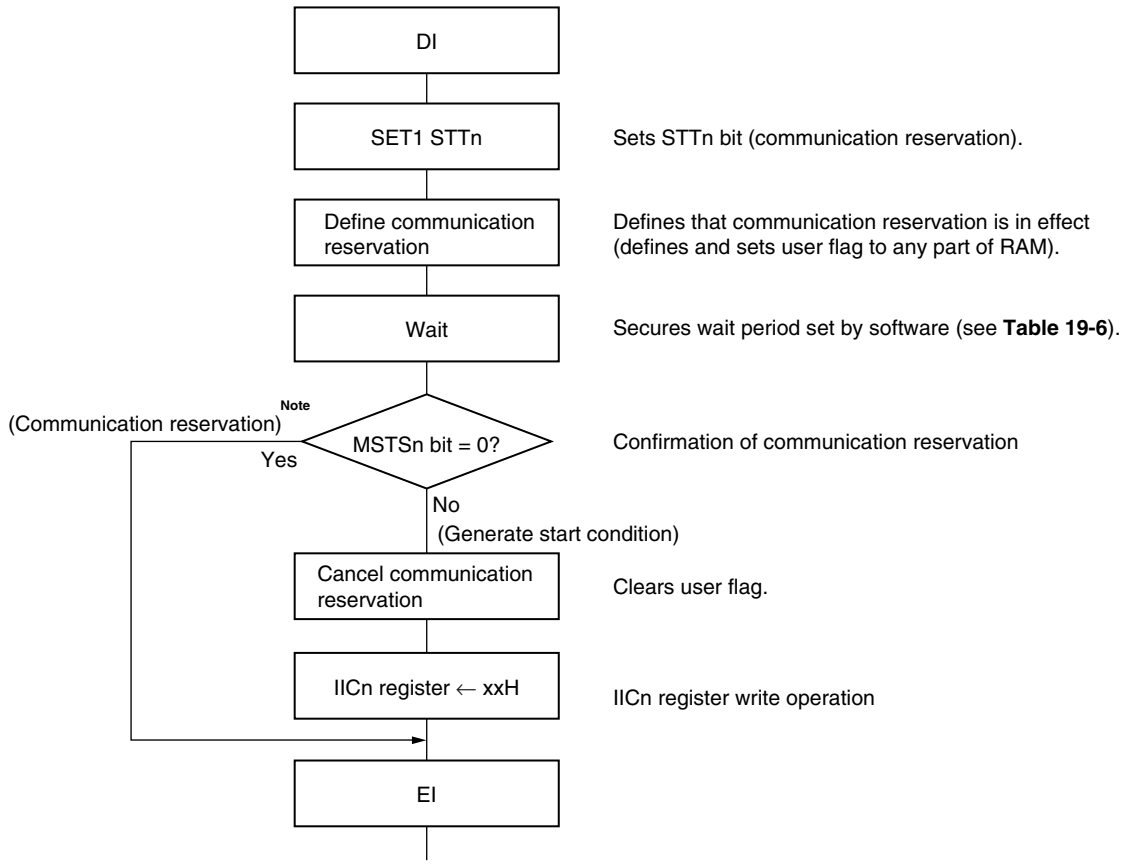
Remarks 1. n = 0 to 2
m = 0, 1

2. × = don't care

The communication reservation timing is shown below.

The communication reservation flowchart is illustrated below.

Figure 19-17. Communication Reservation Flowchart



Note When communication is reserved, the IICn register is written when a stop condition interrupt request occurs.

Remark n = 0 to 2

Table 20-7. Data of UF0CIEn Register

(a) Configuration descriptor (9 bytes)

Offset	Field Name	Contents
0	bLength	Size of this descriptor
1	bDescriptorType	Descriptor type
2	wTotalLength	Lower value of the total number of bytes of Configuration, all Interface, and all Endpoint descriptors
3		Higher value of the total number of bytes of Configuration, all Interface, and all Endpoint descriptors
4	bNumInterface	Number of Interfaces
5	bConfigurationValue	Value to select this Configuration
6	iConfiguration	Index of string descriptor describing this Configuration
7	bmAttributes	Features of this Configuration (self-powered, without remote wakeup)
8	MaxPower	Maximum power consumption of this Configuration (unit: mA)[Note]

Note Shown in 2 mA units. (example: 50 = 100 mA)

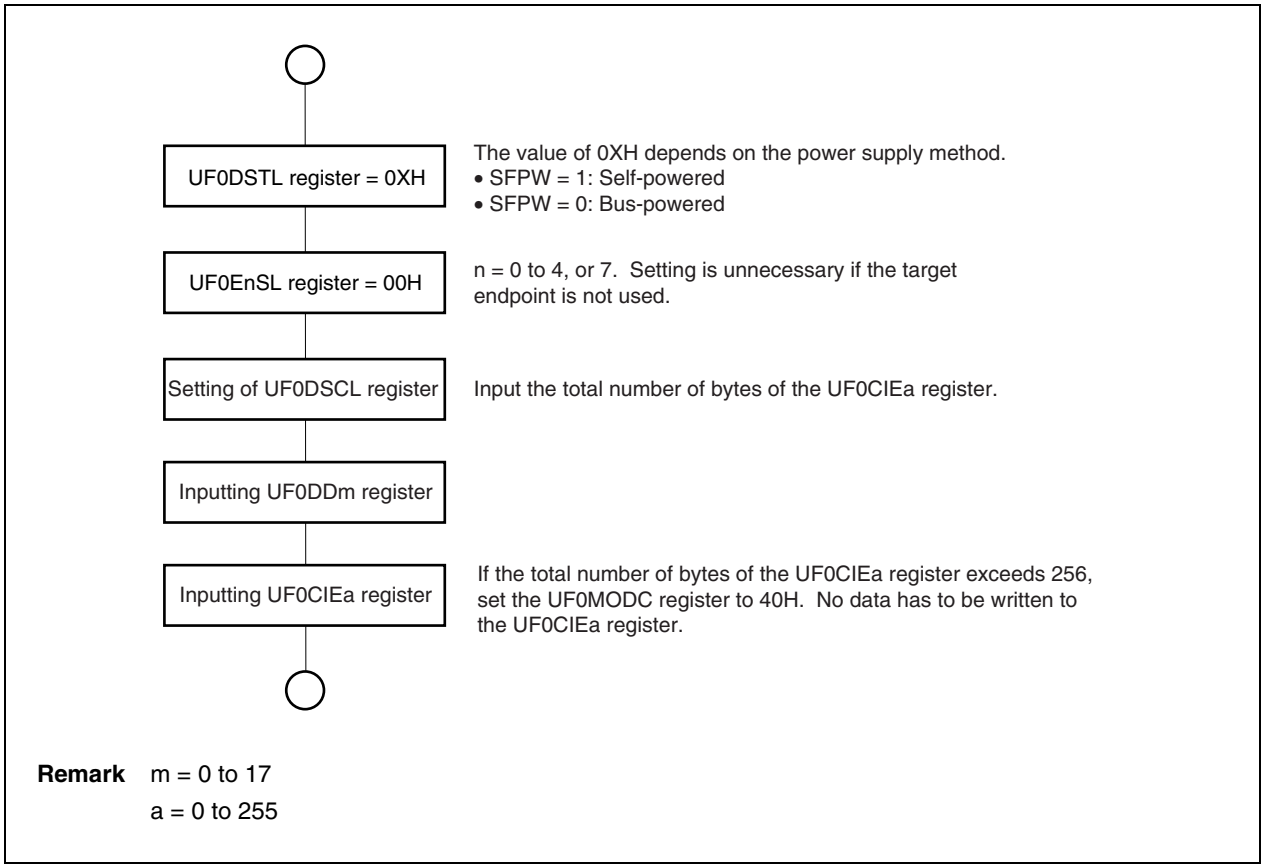
(b) Interface descriptor (9 bytes)

Offset	Field Name	Contents
0	bLength	Size of this descriptor
1	bDescriptorType	Descriptor type
2	bInterfaceNumber	Value of this Interface
3	bAlternateSetting	Value to select alternative setting of Interface
4	bNumEndpoints	Number of usable Endpoints
5	bInterfaceClass	Class code
6	bInterfaceSubClass	Subclass code
7	bInterfaceProtocol	Protocol code
8	Interface	Index of string descriptor describing this Interface

(c) Endpoint descriptor (7 bytes)

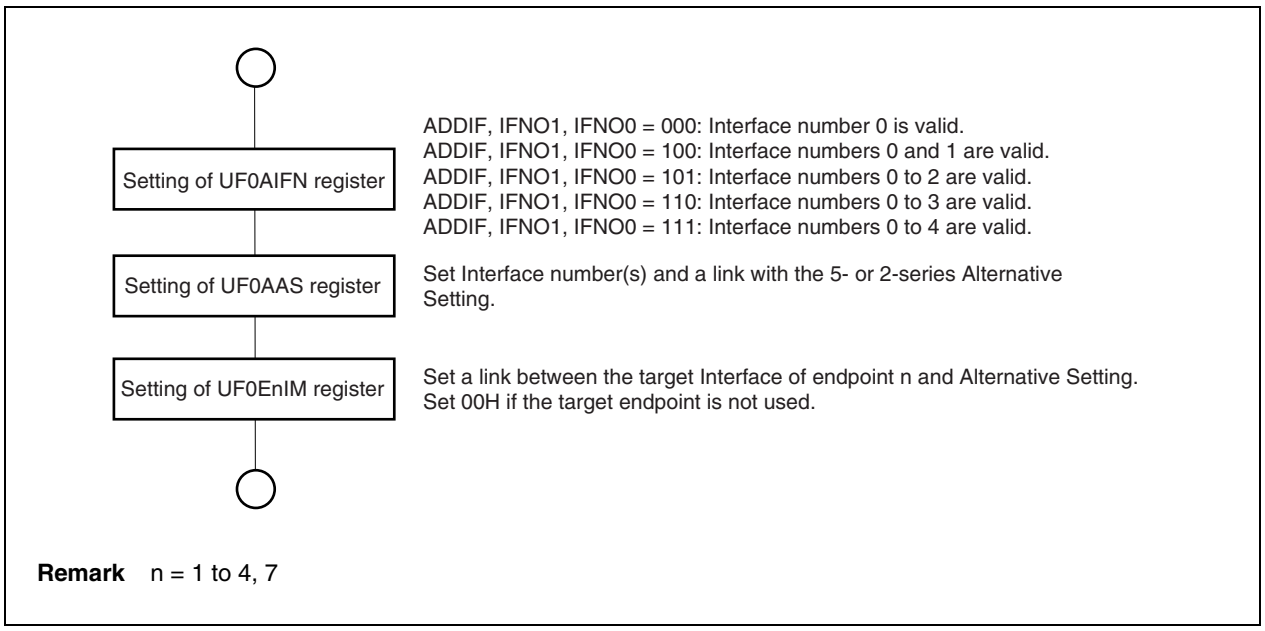
Offset	Field Name	Contents
0	bLength	Size of this descriptor
1	bDescriptorType	Descriptor type
2	bEndpointAddress	Address/transfer direction of this Endpoint
3	bmAttributes	Transfer type
4	wMaxPaketSize	Lower value of maximum number of transfer data
5		Higher value of maximum number of transfer data
6	bInterval	Transfer interval

Figure 20-15. Initialization Settings of Request Data Register



Remark m = 0 to 17
a = 0 to 255

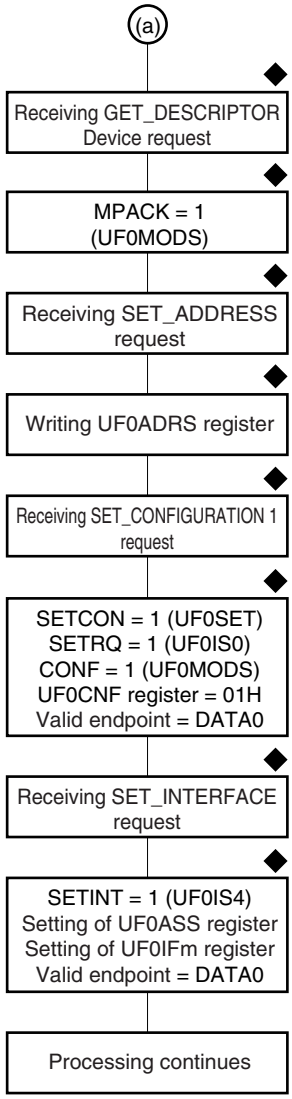
Figure 20-16. Setting of Interface and Endpoint



Remark n = 1 to 4, 7

Figure 20-31. Example of Processing After Power Application/Power Failure (2/3)

(a) Processing after power application (2/2)



Remarks 1. m = 0 to 4

2. ♦: Processing by hardware

(5) DMA channel control registers 0 to 3 (DCHC0 to DCHC3)

The DCHC0 to DCHC3 registers are 8-bit registers that control DMA transfer for DMA channel n.

These registers can be read or written in 8-bit or 1-bit units. (However, bit 7 is read-only and bits 1 and 2 are write-only. If bit 1 or 2 is read, the value read is always 0.)

Reset sets these registers to 00H.

After reset: 00H R/W Address: DCHC0 FFFFF0E0H, DCHC1 FFFFF0E2H, DCHC2 FFFFF0E4H, DCHC3 FFFFF0E6H

	<7>	6	5	4	3	<2>	<1>	<0>
DCHCn	TCn Note 1	0	0	0	0	INITn Note 2	STGn Note 2	Enn

(n = 0 to 3)

TCn Note 1	Status flag indicating whether DMA transfer via DMA channel n is complete
0	DMA transfer is not complete.
1	DMA transfer is complete.
This bit is set to 1 at the last DMA transfer and cleared to 0 when it is read.	

INITn Note 2	If the INITn bit is set to 1 with DMA transfer disabled (Enn bit = 0), the DMA transfer status can be initialized.

STGn Note 2	This is a software startup trigger for DMA transfer. If this bit is set to 1 in the DMA transfer enabled state (TCn bit = 0, Enn bit = 1), DMA transfer is started.

Enn	Setting of whether DMA transfer via DMA channel n is to be enabled or disabled
0	DMA transfer disabled
1	DMA transfer enabled
DMA transfer is enabled when the Enn bit is set to 1. When DMA transfer is completed (when a terminal count is generated), this bit is automatically cleared to 0. To abort DMA transfer, clear the Enn bit to 0 by software. To resume DMA transfer, set the Enn bit to 1 again. When aborting or resuming DMA transfer, be sure to follow the procedure described in **18.13 (5) Procedure for temporarily stopping DMA transfer**.	

Notes 1. The TCn bit is read-only.

2. The INITn and STGn bits are write-only.

Cautions 1. Be sure to clear bits 6 to 3 of the DCHCn register to 0.

2. When DMA transfer is completed (when a terminal count is generated), the Enn bit is cleared to 0 and then the TCn bit is set to 1. If the DCHCn register is read while its bits are being updated, a value indicating "transfer not completed and transfer is disabled" (TCn bit = 0 and Enn bit = 0) may be read.

22.3.4 Interrupt control register (xxICn)

An xxICn register is assigned to each interrupt request signal (maskable interrupt) and sets the control conditions for each maskable interrupt request.

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

Reset sets this register to 47H.

Cautions 1. To mask interrupts, set up the IMR register or use a bit manipulation instruction. The priority levels must be specified at a time when no interrupt will occur.

2. Disable interrupts (DI) before reading the xxICn.xxIFn bit. If the xxIFn bit is read while interrupts are enabled (EI), the correct value may not be read if acknowledging an interrupt and reading the bit conflict.

After reset: 47H R/W Address: FFFFF110H to FFFFF17CH

	<7>	<6>	5	4	3	2	1	0
xxICn	xxIFn	xxMKn	0	0	0	xxPRn2	xxPRn1	xxPRn0

xxIFn	Interrupt request flag[Note]
0	Interrupt request not issued
1	Interrupt request issued

xxMKn	Interrupt mask flag
0	Interrupt servicing enabled
1	Interrupt servicing disabled (pending)

xxPRn2	xxPRn1	xxPRn0	Interrupt priority specification bit
0	0	0	Specifies level 0 (highest).
0	0	1	Specifies level 1.
0	1	0	Specifies level 2.
0	1	1	Specifies level 3.
1	0	0	Specifies level 4.
1	0	1	Specifies level 5.
1	1	0	Specifies level 6.
1	1	1	Specifies level 7 (lowest).

Note The flag xxIFn is reset automatically by the hardware if an interrupt request signal is acknowledged.

Remark xx: Identification name of each peripheral unit (see **Table 22-3 Interrupt Control Registers (xxICn)**)
n: Peripheral unit number (see **Table 22-3 Interrupt Control Registers (xxICn)**).

The addresses and bits of the interrupt control registers are as follows.

22.9 Periods in Which Interrupts Are Not Acknowledged by CPU

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)
- The store instruction for the PRCMD register
- The store, SET1, NOT1, or CLR1 instructions for the following registers.
  - Interrupt-related registers:
    Interrupt control register (xxICn), interrupt mask registers 0 to 3 (IMR0 to IMR3)
  - Power save control register (PSC)
  - On-chip debug mode register (OCDM)

Remarks 1. xx: Identification name of each peripheral unit (see **Table 22-3 Interrupt Control Registers (xxICn)**)
n: Peripheral unit number (see **Table 22-3 Interrupt Control Registers (xxICn)**).

2. For details about the operation of the pipeline, see the **V850ES Architecture User's Manual (U15943E)**.

22.10 Cautions

22.10.1 Restored PC

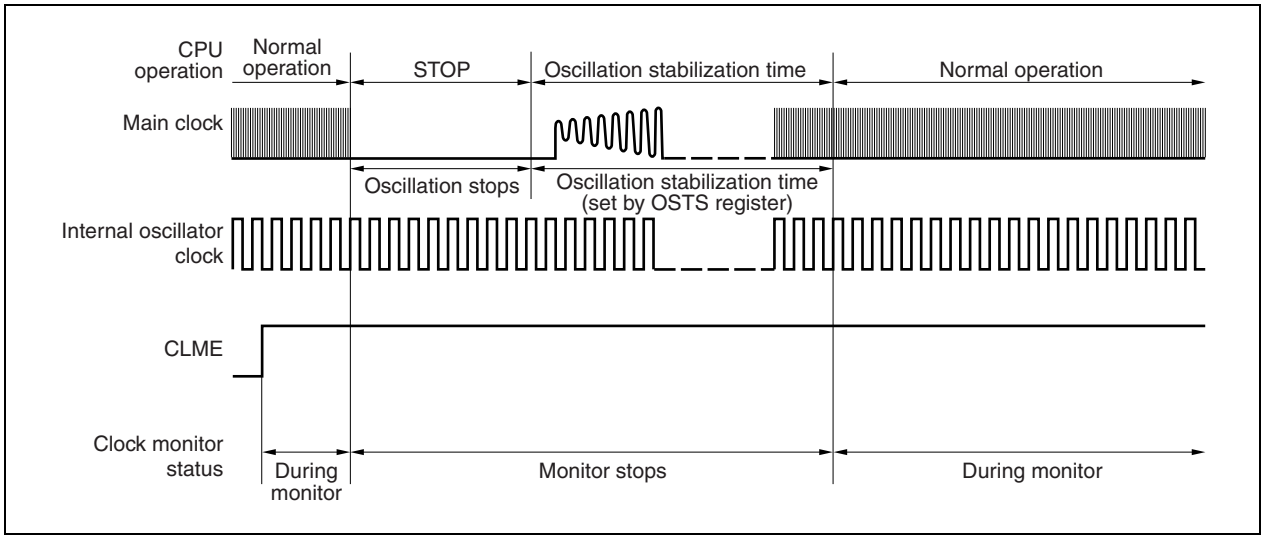
Restored PC is the value of the program counter (PC) saved to EIPC, FEPC, or DBPC when interrupt servicing starts. If a non-maskable or maskable interrupt is acknowledged during the execution of any of the following instructions, the execution of that instruction stops and resumes following completion of interrupt servicing.

- Load instructions (SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W)
- Divide instructions (DIV, DIVH, DIVU, DIVHU)
- PREPARE, DISPOSE instructions (only when an interrupt occurs before the stack pointer is updated)

(3) Operation in STOP mode or after STOP mode is released

If the STOP mode is set with the CLM.CLME bit = 1, the monitor operation is stopped in the STOP mode and while the oscillation stabilization time is being counted. After the oscillation stabilization time, the monitor operation is automatically started.

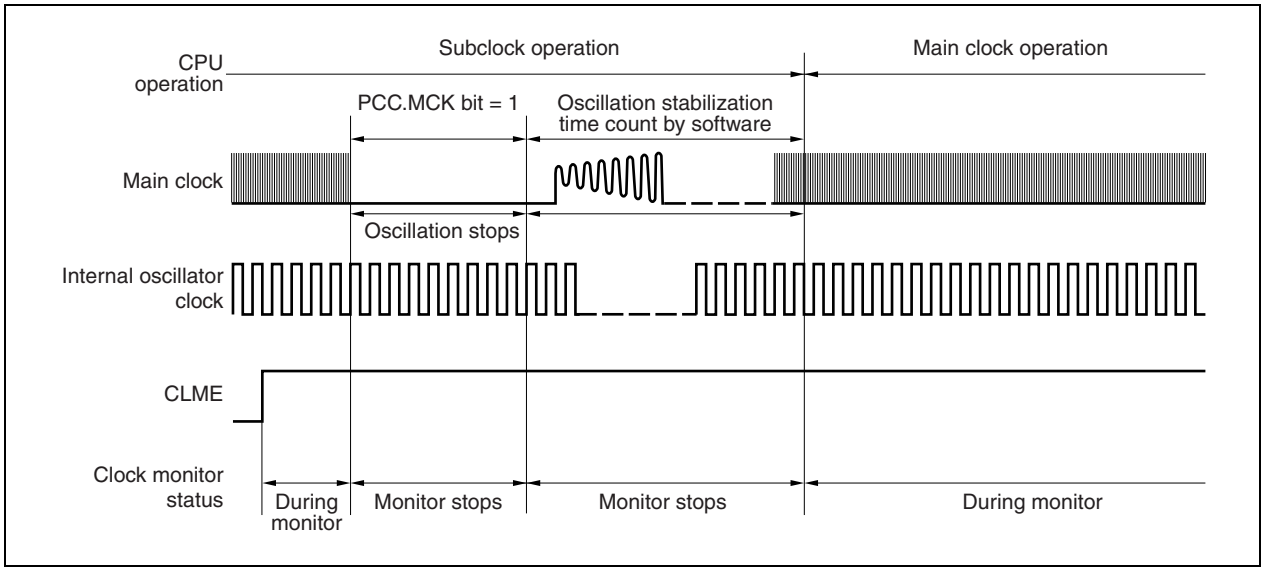
Figure 26-4. Operation in STOP Mode or After STOP Mode Is Released



(4) Operation when main clock is stopped (arbitrary)

During subclock operation (PCC.CLS bit = 1) or when the main clock is stopped by setting the PCC.MCK bit to 1, the monitor operation is stopped until the main clock operation is started (PCC.CLS bit = 0). The monitor operation is automatically started when the main clock operation is started.

Figure 26-5. Operation When Main Clock Is Stopped (Arbitrary)



(5) Operation while CPU is operating on internal oscillator clock (CCLS.CCLSF bit = 1)

The monitor operation is not stopped when the CCLSF bit is 1, even if the CLME bit is set to 1.

(5/6)

Mnemonic	Operand	Opcode	Operation	Execution Clock			Flags				
				i	r	l	CY	OV	S	Z	SAT
SET1	bit#3,disp16[reg1]	00bbb111110RRRRR dddddddddddddddd	adr←GR[reg1]+sign-extend(disp16) Z flag←Not (Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,1)	3 Note 3	3 Note 3	3 Note 3				×	
	reg2,[reg1]	rrrrr111111RRRRR 0000000011100000	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,1)	3 Note 3	3 Note 3	3 Note 3				×	
SHL	reg1,reg2	rrrrr111111RRRRR 0000000011000000	GR[reg2]←GR[reg2] logically shift left by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010110iiiii	GR[reg2]←GR[reg2] logically shift left by zero-extend(imm5)	1	1	1	×	0	×	×	
SHR	reg1,reg2	rrrrr111111RRRRR 0000000010000000	GR[reg2]←GR[reg2] logically shift right by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010100iiiii	GR[reg2]←GR[reg2] logically shift right by zero-extend(imm5)	1	1	1	×	0	×	×	
SLD.B	disp7[ep],reg2	rrrrr0110ddddddd	adr←ep+zero-extend(disp7) GR[reg2]←sign-extend(Load-memory(adr,Byte))	1	1	Note 9					
SLD.BU	disp4[ep],reg2	rrrrr0000110dddd **Note 18**	adr←ep+zero-extend(disp4) GR[reg2]←zero-extend(Load-memory(adr,Byte))	1	1	Note 9					
SLD.H	disp8[ep],reg2	rrrrr1000ddddddd **Note 19**	adr←ep+zero-extend(disp8) GR[reg2]←sign-extend(Load-memory(adr,Halfword))	1	1	Note 9					
SLD.HU	disp5[ep],reg2	rrrrr0000111dddd **Notes 18, 20**	adr←ep+zero-extend(disp5) GR[reg2]←zero-extend(Load-memory(adr,Halfword))	1	1	Note 9					
SLD.W	disp8[ep],reg2	rrrrr1010ddddddd0 **Note 21**	adr←ep+zero-extend(disp8) GR[reg2]←Load-memory(adr,Word)	1	1	Note 9					
SST.B	reg2,disp7[ep]	rrrrr0111ddddddd	adr←ep+zero-extend(disp7) Store-memory(adr,GR[reg2],Byte)	1	1	1					
SST.H	reg2,disp8[ep]	rrrrr1001ddddddd **Note 19**	adr←ep+zero-extend(disp8) Store-memory(adr,GR[reg2],Halfword)	1	1	1					
SST.W	reg2,disp8[ep]	rrrrr1010ddddddd1 **Note 21**	adr←ep+zero-extend(disp8) Store-memory(adr,GR[reg2],Word)	1	1	1					
ST.B	reg2,disp16[reg1]	rrrrr111010RRRRR dddddddddddddddd	adr←GR[reg1]+sign-extend(disp16) Store-memory(adr,GR[reg2],Byte)	1	1	1					
ST.H	reg2,disp16[reg1]	rrrrr111011RRRRR ddddddddddddddd0 **Note 8**	adr←GR[reg1]+sign-extend(disp16) Store-memory (adr,GR[reg2], Halfword)	1	1	1					
ST.W	reg2,disp16[reg1]	rrrrr111011RRRRR ddddddddddddddd1 **Note 8**	adr←GR[reg1]+sign-extend(disp16) Store-memory (adr,GR[reg2], Word)	1	1	1					
STSR	regID,reg2	rrrrr111111RRRRR 0000000001000000	GR[reg2]←SR[regID]	1	1	1					