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	H8/300H
Core Size	16-Bit
Speed	20MHz
Connectivity	CANbus, SCI, SSU
Peripherals	LVD, POR, PWM, WDT
Number of I/O	45
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 8x10b SAR; D/A 1x10b
Oscillator Type	External, Internal
Operating Temperature	-20°C ~ 75°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LFQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/renesas-electronics-america/df36034fpv

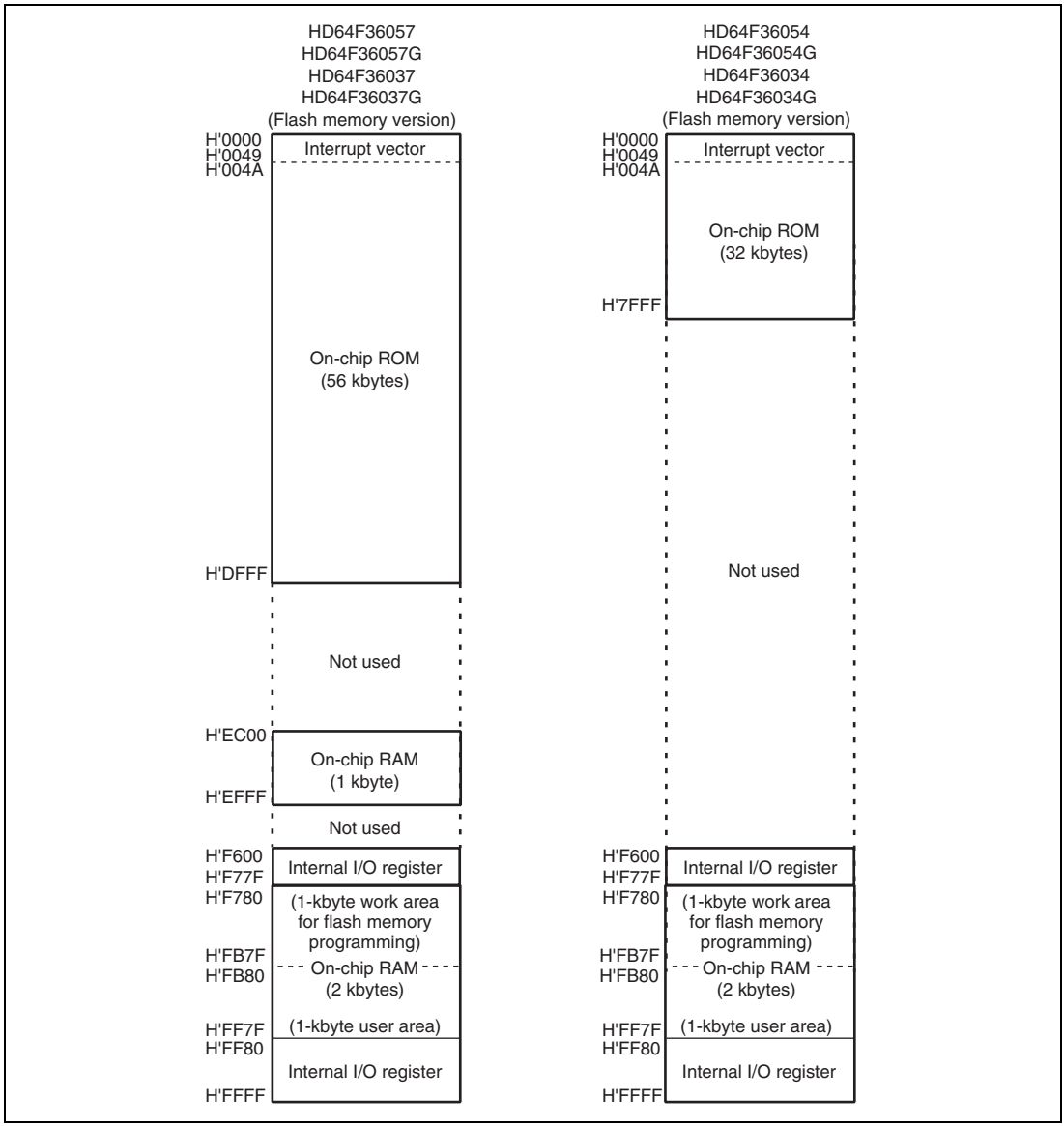


Figure 2.1 Memory Map (1)

- After executing BSET instruction

	P57	P56	P55	P54	P53	P52	P51	P50
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	High level
PCR5	0	0	1	1	1	1	1	1
PDR5	0	1	0	0	0	0	0	1

- Description on operation

1. When the BSET instruction is executed, first the CPU reads port 5.
   Since P57 and P56 are input pins, the CPU reads the pin states (low-level and high-level input).
   P55 to P50 are output pins, so the CPU reads the value in PDR5. In this example PDR5 has a value of H'80, but the value read by the CPU is H'40.
2. Next, the CPU sets bit 0 of the read data to 1, changing the PDR5 data to H'41.
3. Finally, the CPU writes H'41 to PDR5, completing execution of BSET instruction.

As a result of the BSET instruction, bit 0 in PDR5 becomes 1, and P50 outputs a high-level signal. However, bits 7 and 6 of PDR5 end up with different values. To prevent this problem, store a copy of the PDR5 data in a work area in memory. Perform the bit manipulation on the data in the work area, then write this data to PDR5.

- Prior to executing BSET instruction

```
MOV.B   #80,  R0L
MOV.B   R0L,  @RAM0
MOV.B   R0L,  @PDR5
```

The PDR5 value (H'80) is written to a work area in memory (RAM0) as well as to PDR5.

	P57	P56	P55	P54	P53	P52	P51	P50
Input/output	Input	Input	Output	Output	Output	Output	Output	Output
Pin state	Low level	High level	Low level	Low level	Low level	Low level	Low level	Low level
PCR5	0	0	1	1	1	1	1	1
PDR5	1	0	0	0	0	0	0	0
RAM0	1	0	0	0	0	0	0	0

3.4.3 Interrupt Handling Sequence

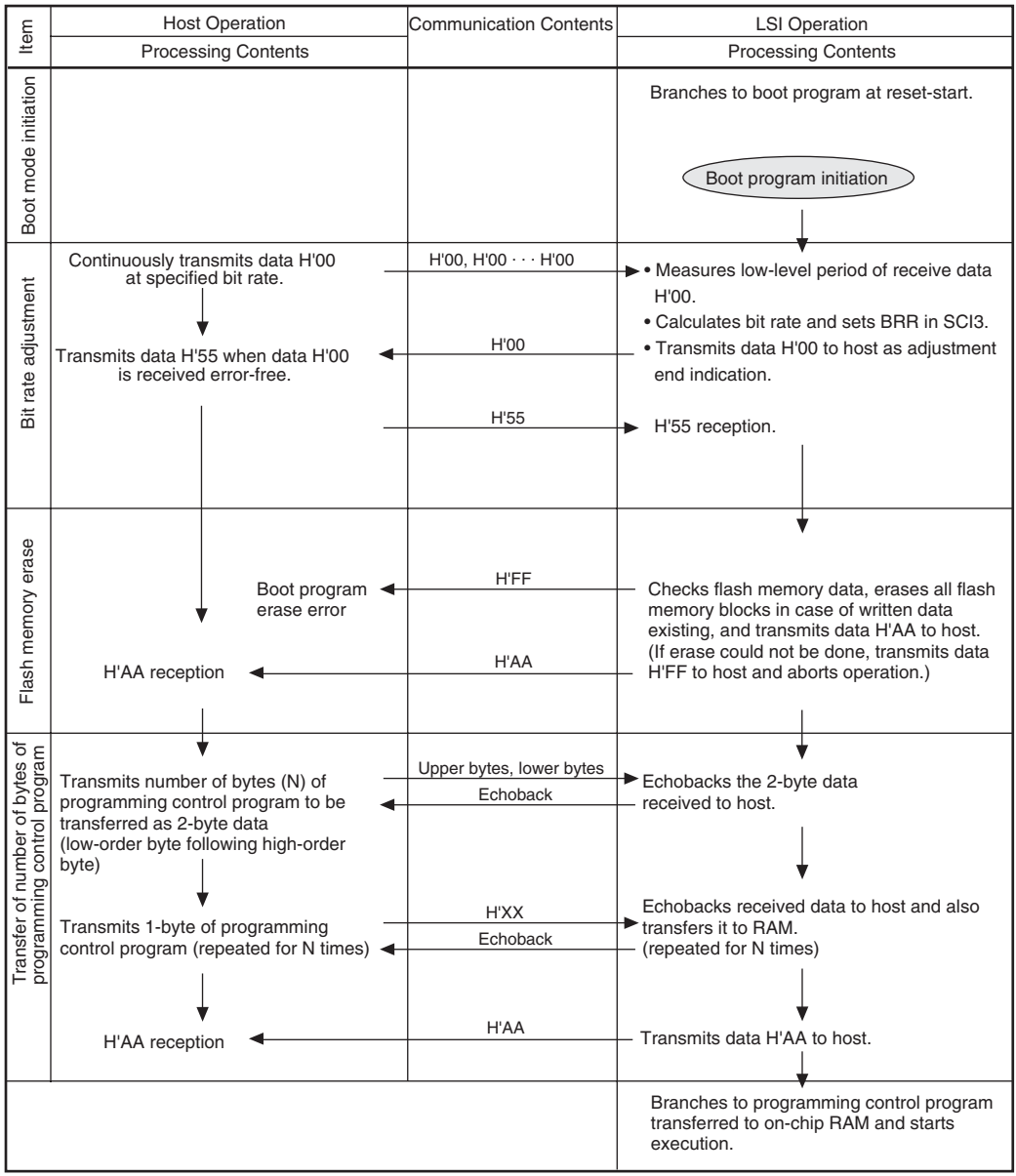
Interrupts are controlled by an interrupt controller.

Interrupt operation is described as follows.

1. If an interrupt occurs while the NMI or interrupt enable bit is set to 1, an interrupt request signal is sent to the interrupt controller.
2. When multiple interrupt requests are generated, the interrupt controller requests to the CPU for the interrupt handling with the highest priority at that time according to table 3.1. Other interrupt requests are held pending.
3. The CPU accepts the NMI and address break without depending on the I bit value. Other interrupt requests are accepted, if the I bit is cleared to 0 in CCR; if the I bit is set to 1, the interrupt request is held pending.
4. If the CPU accepts the interrupt after processing of the current instruction is completed, interrupt exception handling will begin. First, both PC and CCR are pushed onto the stack. The state of the stack at this time is shown in figure 3.2. The PC value pushed onto the stack is the address of the first instruction to be executed upon return from interrupt handling.
5. Then, the I bit of CCR is set to 1, masking further interrupts excluding the NMI and address break. Upon return from interrupt handling, the values of I bit and other bits in CCR will be restored and returned to the values prior to the start of interrupt exception handling.
6. Next, the CPU generates the vector address corresponding to the accepted interrupt, and transfers the address to PC as a start address of the interrupt handling-routine. Then a program starts executing from the address indicated in PC.

Figure 3.3 shows a typical interrupt sequence where the program area is in the on-chip ROM and the stack area is in the on-chip RAM.

Table 7.2 Boot Mode Operation



12.3.14 Interface with CPU

16-Bit Register: TCNT and GR are 16-bit registers. Reading/writing in a 16-bit unit is enabled but disabled in an 8-bit unit since the data bus with the CPU is 16-bit width. These registers must always be accessed in a 16-bit unit. Figure 12.5 shows an example of accessing the 16-bit registers.

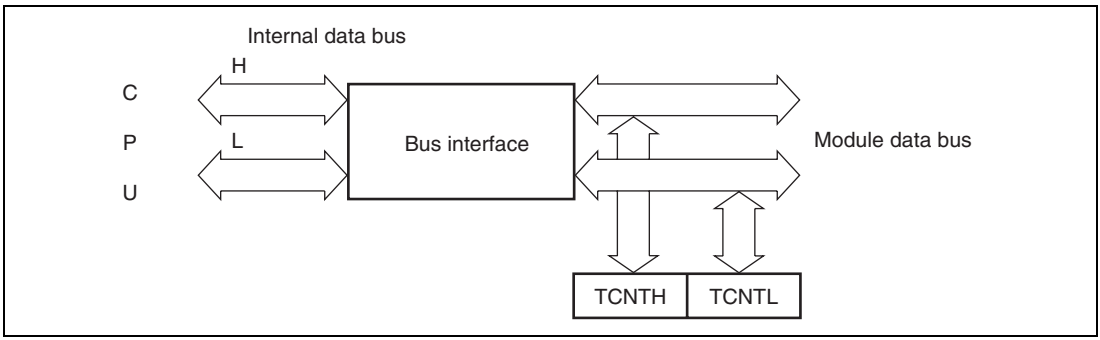


Figure 12.5 Accessing Operation of 16-Bit Register (between CPU and TCNT (16 Bits))

8-Bit Register: Registers other than TCNT and GR are 8-bit registers that are connected internally with the CPU in an 8-bit width. Figure 12.6 shows an example of accessing the 8-bit registers.

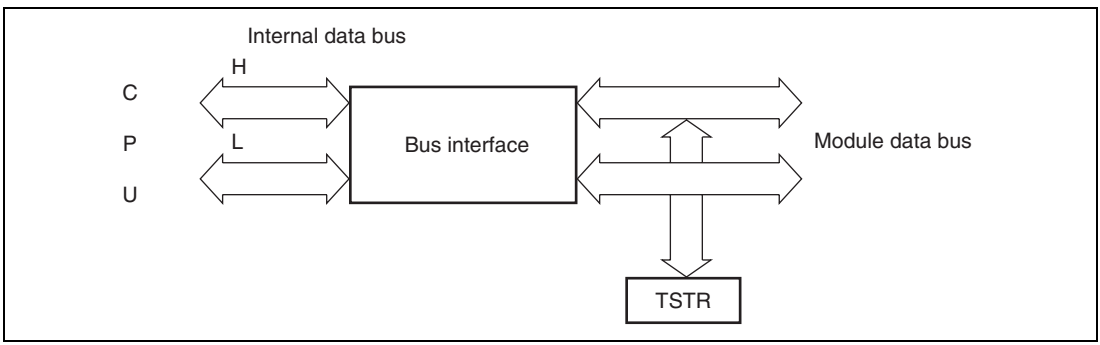


Figure 12.6 Accessing Operation of 8-Bit Register (between CPU and TSTR (8 Bits))

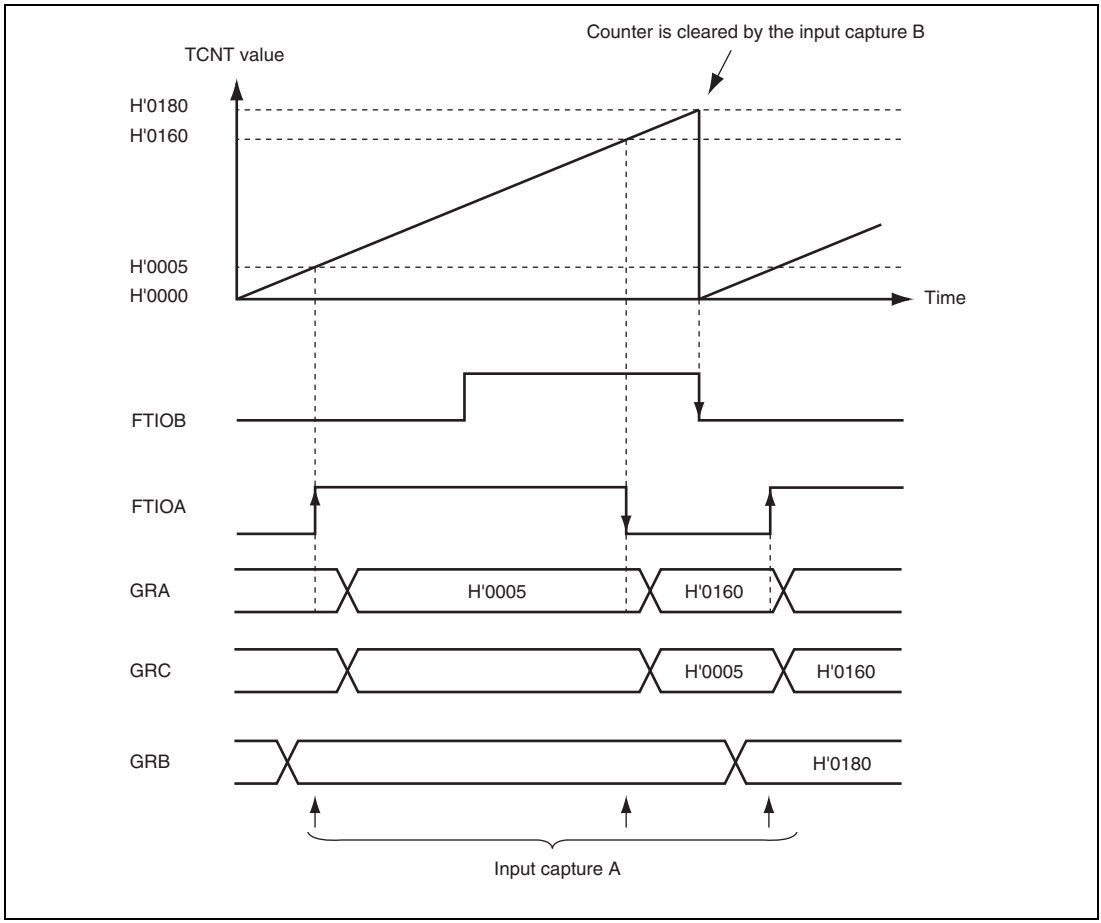


Figure 12.40 Example of Buffer Operation (2)
(Buffer Operation for Input Capture Register)

14.3.8 Bit Rate Register (BRR)

BRR is an 8-bit register that adjusts the bit rate. The initial value of BRR is H'FF. Table 14.3 shows the relationship between the N setting in BRR and the n setting in bits CKS1 and CKS0 of SMR in asynchronous mode. Table 14.4 shows the maximum bit rate for each frequency in asynchronous mode. The values shown in both tables 14.3 and 14.4 are values in active (high-speed) mode. Table 14.5 shows the relationship between the N setting in BRR and the n setting in bits CKS1 and CKS0 of SMR in clocked synchronous mode. The values shown in table 14.5 are values in active (high-speed) mode. The N setting in BRR and error for other operating frequencies and bit rates can be obtained by the following formulas:

[Asynchronous Mode]

$$N = \frac{\phi}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

$$\text{Error (\%)} = \left\{ \frac{\phi \times 10^6}{(N + 1) \times B \times 64 \times 2^{2n-1}} - 1 \right\} \times 100$$

[Clocked Synchronous Mode]

$$N = \frac{\phi}{8 \times 2^{2n-1} \times B} \times 10^6 - 1$$

[Legend]
B: Bit rate (bit/s)
N: BRR setting for baud rate generator ($0 \leq N \leq 255$)
ϕ: Operating frequency (MHz)
n: CSK1 and CSK0 settings in SMR ($0 \leq n \leq 3$)

14.4.4 Serial Data Reception

Figure 14.7 shows an example of operation for reception in asynchronous mode. In serial reception, the SCI3 operates as described below.

1. The SCI3 monitors the communication line. If a start bit is detected, the SCI3 performs internal synchronization, receives receive data in RSR, and checks the parity bit and stop bit.
2. If an overrun error occurs (when reception of the next data is completed while the RDRF flag is still set to 1), the OER bit in SSR is set to 1. If the RIE bit in SCR3 is set to 1 at this time, an ERI interrupt request is generated. Receive data is not transferred to RDR.
3. If a parity error is detected, the PER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR3 is set to 1 at this time, an ERI interrupt request is generated.
4. If a framing error is detected (when the stop bit is 0), the FER bit in SSR is set to 1 and receive data is transferred to RDR. If the RIE bit in SCR3 is set to 1 at this time, an ERI interrupt request is generated.
5. If reception is completed successfully, the RDRF bit in SSR is set to 1, and receive data is transferred to RDR. If the RIE bit in SCR3 is set to 1 at this time, an RXI interrupt request is generated. Continuous reception is possible because the RXI interrupt routine reads the receive data transferred to RDR before reception of the next receive data has been completed.

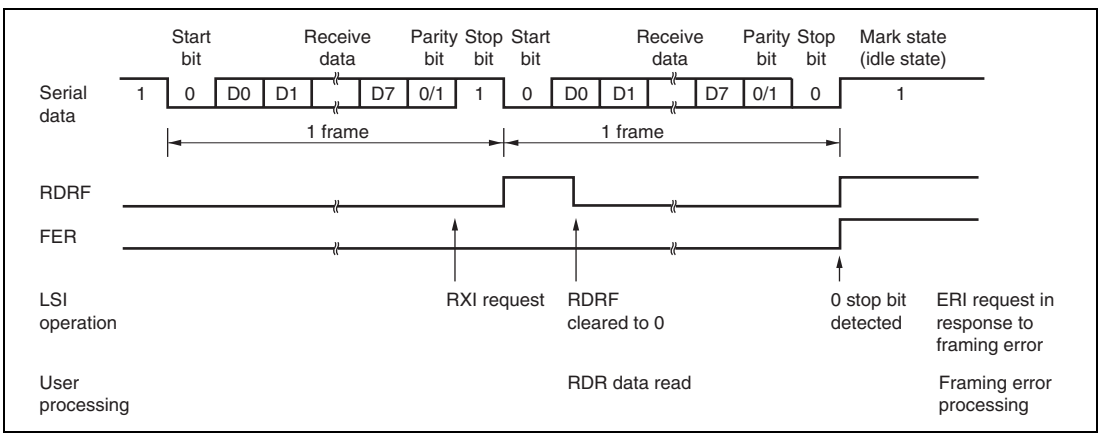


Figure 14.7 Example of SCI3 Reception in Asynchronous Mode (8-Bit Data, Parity, One Stop Bit)

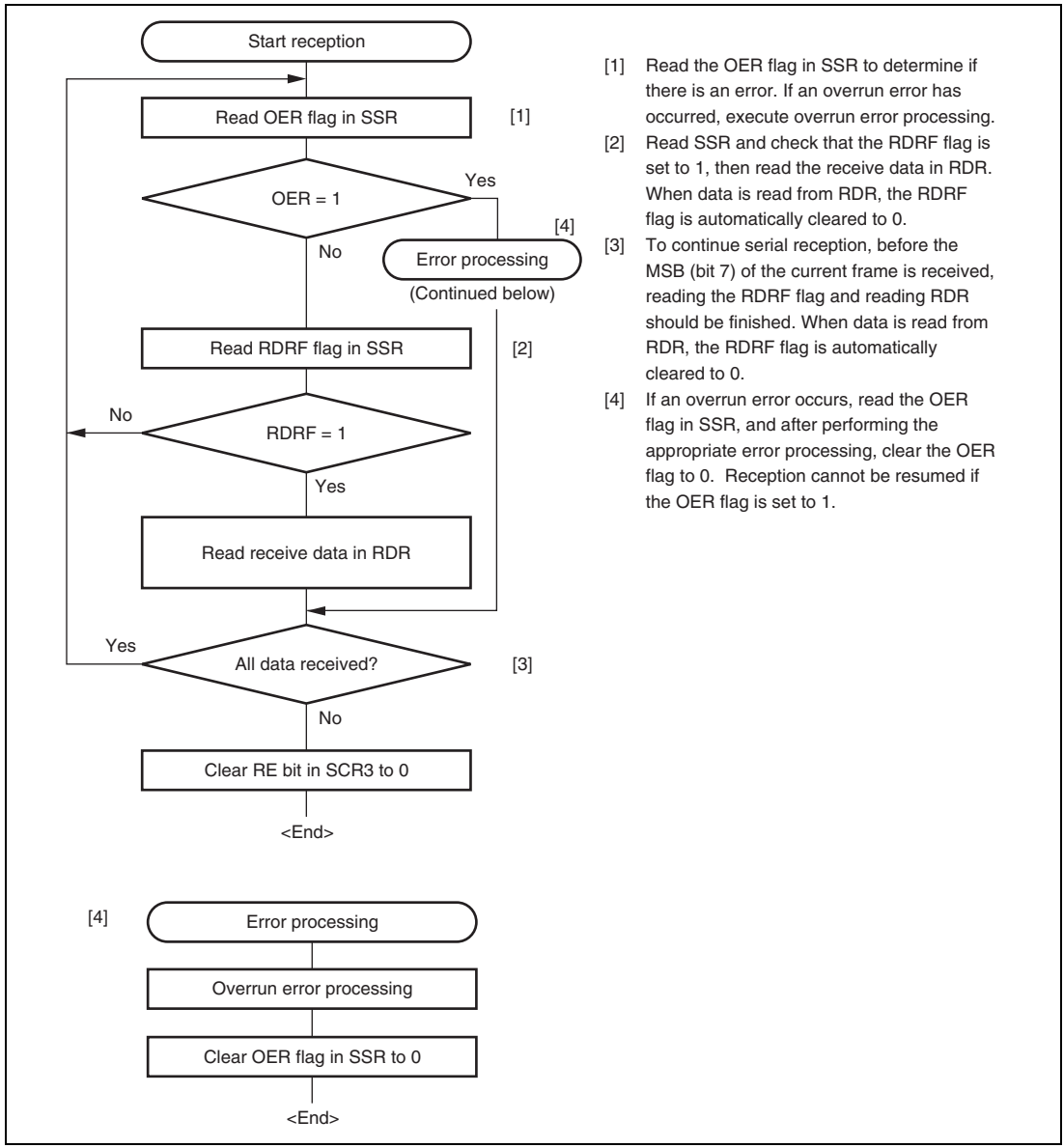
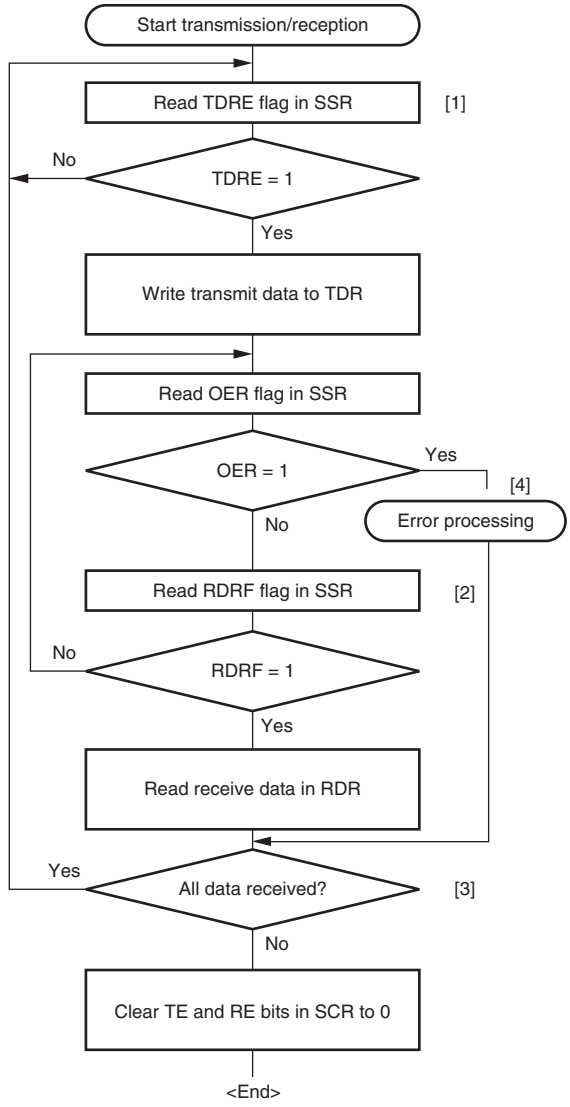


Figure 14.13 Sample Serial Reception Flowchart (Clocked Synchronous Mode)



[1] Read SSR and check that the TDRE flag is set to 1, then write transmit data to TDR.
When data is written to TDR, the TDRE flag is automatically cleared to 0.

[2] Read SSR and check that the RDRF flag is set to 1, then read the receive data in RDR.
When data is read from RDR, the RDRF flag is automatically cleared to 0.

[3] To continue serial transmission/ reception, before the MSB (bit 7) of the current frame is received, finish reading the RDRF flag, reading RDR. Also, before the MSB (bit 7) of the current frame is transmitted, read 1 from the TDRE flag to confirm that writing is possible. Then write data to TDR.
When data is written to TDR, the TDRE flag is automatically cleared to 0. When data is read from RDR, the RDRF flag is automatically cleared to 0.

[4] If an overrun error occurs, read the OER flag in SSR, and after performing the appropriate error processing, clear the OER flag to 0. Transmission/reception cannot be resumed if the OER flag is set to 1. For overrun error processing, see figure 14.13.

Figure 14.14 Sample Flowchart of Simultaneous Serial Transmit and Receive Operations (Clocked Synchronous Mode)

14.6 Multiprocessor Communication Function

Use of the multiprocessor communication function enables data transfer between a number of processors sharing communication lines by asynchronous serial communication using the multiprocessor format, in which a multiprocessor bit is added to the transfer data. When multiprocessor communication is performed, each receiving station is addressed by a unique ID code. The serial communication cycle consists of two component cycles; an ID transmission cycle that specifies the receiving station, and a data transmission cycle. The multiprocessor bit is used to differentiate between the ID transmission cycle and the data transmission cycle. If the multiprocessor bit is 1, the cycle is an ID transmission cycle; if the multiprocessor bit is 0, the cycle is a data transmission cycle. Figure 14.15 shows an example of inter-processor communication using the multiprocessor format. The transmitting station first sends the ID code of the receiving station with which it wants to perform serial communication as data with a 1 multiprocessor bit added. It then sends transmit data as data with a 0 multiprocessor bit added. When data with a 1 multiprocessor bit is received, the receiving station compares that data with its own ID. The station whose ID matches then receives the data sent next. Stations whose IDs do not match continue to skip data until data with a 1 multiprocessor bit is again received.

The SCI3 uses the MPIE bit in SCR3 to implement this function. When the MPIE bit is set to 1, transfer of receive data from RSR to RDR, error flag detection, and setting the SSR status flags, RDRF, FER, and OER, to 1, are inhibited until data with a 1 multiprocessor bit is received. On reception of a receive character with a 1 multiprocessor bit, the MPBR bit in SSR is set to 1 and the MPIE bit is automatically cleared, thus normal reception is resumed. If the RIE bit in SCR3 is set to 1 at this time, an RXI interrupt is generated.

When the multiprocessor format is selected, the parity bit setting is rendered invalid. All other bit settings are the same as those in normal asynchronous mode. The clock used for multiprocessor communication is the same as that in normal asynchronous mode.

15.3.7 Transmit Pending Register (TXPR)

TXPR sets transmit pending (CAN bus arbitration wait) for the transmit message that is stored in a Mailbox. Setting the corresponding bit in TXPR to 1 enables a message to be transmitted. Writing 0 to the bit in TXPR is ignored.

Bit	Bit Name	Initial Value	R/W	Description
7 to 4	—	All 0	—	Reserved These bits are always read as 0.
3	MB3	0	R/W	[Setting condition] When the corresponding MBCR bit for a mailbox is 0, the corresponding bit in TXPR is set to 1 (n = 3 to 1) [Clearing conditions] • When message transmission has completed successfully (TXACKn set) • When transmission cancellation for an untransmitted message has finished (ABACKn set) • When a transmission cancellation request has occurred during message transmission, and an error occurs or arbitration is lost on the CAN bus (ABACKn set) • When a transmit error or arbitration loss occurred with the corresponding DART bit for a message being transmitted set to 1 If the message is not transmitted successfully, the MBn bit is not cleared to 0. If any of these MB bits in TXPR are cleared to 0, the EMPI bit in TCIRR1 is set to 1. The TinyCAN automatically attempts retransmission as long as the DART bit in the message control of the corresponding Mailbox is not set to 1 or the corresponding bit in TXCR is not set to 1. Note: When the MBn bit in MBCR is set to 1, the TinyCAN does not transmit a message even if the MBn bit in TXPR is set to 1. To clear the MBn bit in TXPR to 0, set the MBn bit in TXCR to 1 beforehand.
2	MB2	0	R/W	
1	MB1	0	R/W	
0	—	0	—	Reserved This bit is always read as 0. This bit is relevant to the receive-only Mailbox, and its value cannot be changed.

15.3.13 Unread Message Status Register (UMSR)

UMSR is a status flag that indicates that an unread message in each Mailbox has been overwritten by a new receive message or a new receive message has been discarded.

Bit	Bit Name	Initial Value	R/W	Description
7 to 4	—	All 0	—	Reserved These bits are always read as 0.
3	MB3	0	R/(W)*	Status flags indicating that a new receive message has overwritten/overrun an unread message. [Setting condition] When a new message is received before the corresponding bit in RXPR or RFPR is cleared to 0 [Clearing condition] When 1 is written to these bits
2	MB2	0	R/(W)*	
1	MB1	0	R/(W)*	
0	MB0	0	R/(W)*	

Note: * Only 1 can be written to clear the flag.

Bit	Bit Name	Initial Value	R/W	Description
3	TEND	0	R/W	Transmit End [Setting condition] • When the last bit of data is transmitted, the TDRE bit is 1 [Clearing conditions] • When 0 is written to this bit after reading 1 • When data is written in SSTDR
2	TDRE	1	R/W	Transmit Data Empty [Setting conditions] • When the TE bit in SSER is 0 • When data transfer is performed from SSTDR to SSTRSR and data can be written in SSTDR [Clearing conditions] • When 0 is written to this bit after reading 1 • When data is written in SSTDR
1	RDRF	0	R/W	Receive Data Register Full [Setting condition] • When serial reception is completed normally and receive data is transferred from SSTRSR to SSRDR [Clearing conditions] • When 0 is written to this bit after reading 1 • When data is read from SSRDR
0	CE	0	R/W	Conflict Error Flag [Setting conditions] • When serial communication is started while SSUMS = 1 and MSS =1, the $\overline{SCS}$ pin input is low • When the $\overline{SCS}$ pin level changes from low to high during transfer while SSUMS = 1 and MSS = 0 [Clearing condition] • When 0 is written to this bit after reading 1

21.1 Register Addresses (Address Order)

The data-bus width column indicates the number of bits. The access-state column shows the number of states of the selected basic clock that is required for access to the register.

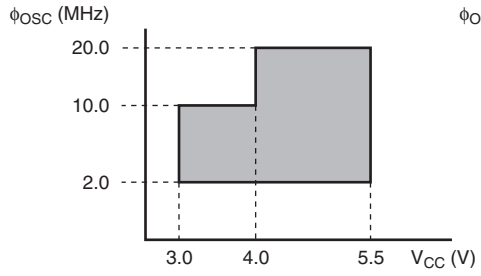
Note: Access to undefined or reserved addresses should not take place. Correct operation of the access itself or later operations is not guaranteed when such a register is accessed.

Register Name	Abbre-viation	Bit No	Address	Module Name	Data Bus Width	Access State
—	—	—	H'F000 to H'F5FF	—	—	—
Master control register	MCR	8	H'F600	TinyCAN	8	4
General status register	GSR	8	H'F601	TinyCAN	8	4
Bit configuration register 1	BCR1	8	H'F602	TinyCAN	8	4
Bit configuration register 0	BCR0	8	H'F603	TinyCAN	8	4
Mailbox configuration register	MBCR	8	H'F604	TinyCAN	8	4
TinyCAN module control register	TCMR	8	H'F605	TinyCAN	8	4
Transmit pending register	TXPR	8	H'F606	TinyCAN	8	4
Transmit pending cancel register	TXCR	8	H'F608	TinyCAN	8	4
Transmit acknowledge register	TXACK	8	H'F60A	TinyCAN	8	4
Abort acknowledge register	ABACK	8	H'F60C	TinyCAN	8	4
Receive complete register	RXPR	8	H'F60E	TinyCAN	8	4
Remote request register	RFPR	8	H'F610	TinyCAN	8	4
TinyCAN interrupt register 1	TCIRR1	8	H'F612	TinyCAN	8	4
TinyCAN interrupt register 0	TCIRR0	8	H'F613	TinyCAN	8	4
Mailbox interrupt mask register	MBIMR	8	H'F614	TinyCAN	8	4
TinyCAN interrupt mask register 1	TCIMR1	8	H'F616	TinyCAN	8	4
TinyCAN interrupt mask register 0	TCIMR0	8	H'F617	TinyCAN	8	4
Receive error counter	REC	8	H'F618	TinyCAN	8	4
Transmit error counter	TEC	8	H'F619	TinyCAN	8	4
Test control register	TCR	8	H'F61A	TinyCAN	8	4
Unread message status register	UMSR	8	H'F61B	TinyCAN	8	4
Message control 0 [0]	MC0[0]	8	H'F620	TinyCAN	8	4

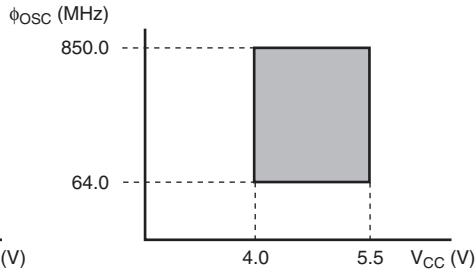
22.2 Electrical Characteristics (F-ZTAT™ Version)

22.2.1 Power Supply Voltage and Operating Ranges

Power Supply Voltage and Oscillation Frequency Range:



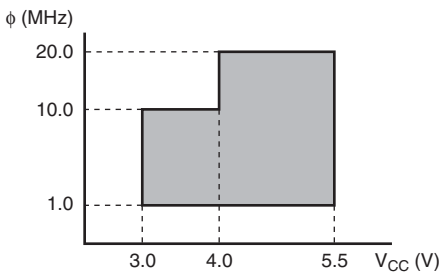
AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode



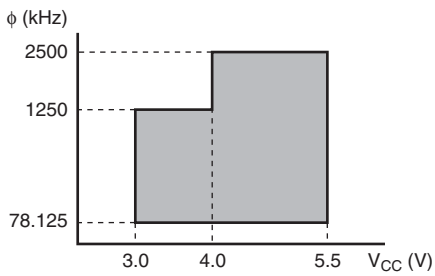
AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode
- Subactive mode
- Subsleep mode

Note: This frequency range is supplied by the on-chip oscillator for the subtimer and is guaranteed.

Power Supply Voltage and Operating Frequency Range:



AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode
  (When MA2 in SYSCR2 = 0)

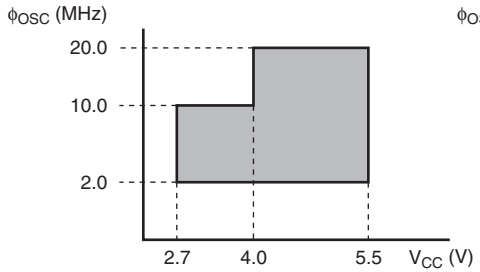


AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode
  (When MA2 in SYSCR2 = 1)

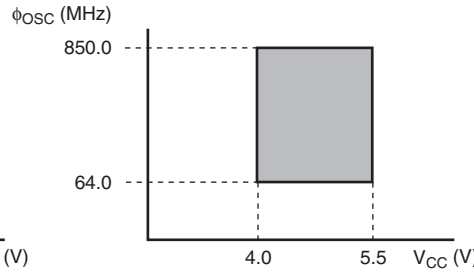
22.3 Electrical Characteristics (Masked ROM Version)

22.3.1 Power Supply Voltage and Operating Ranges

Power Supply Voltage and Oscillation Frequency Range:



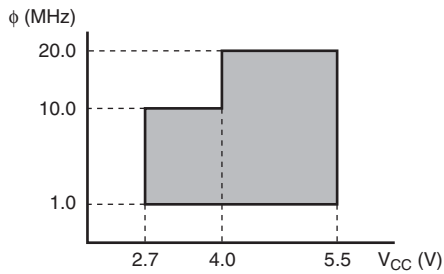
AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode



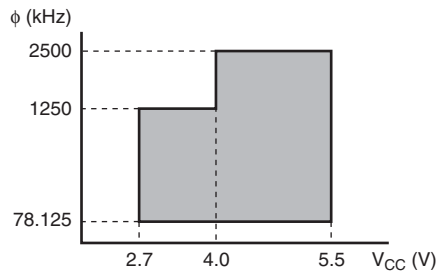
AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode
- Subactive mode
- Subsleep mode

Note: This frequency range is supplied by the on-chip oscillator for the subtimer and is guaranteed.

Power Supply Voltage and Operating Frequency Range:



AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode
  (When MA2 in SYSCR2 = 0)



AV_{CC} = 3.3 to 5.5 V
- Active mode
- Sleep mode
  (When MA2 in SYSCR2 = 1)

22.4 Operation Timing

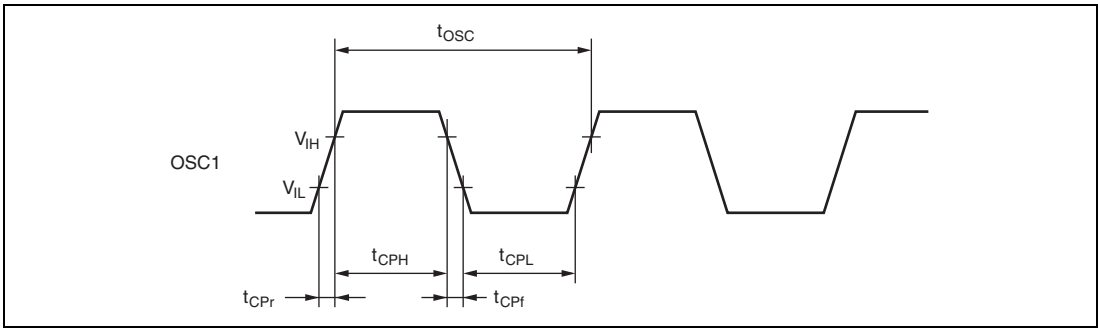


Figure 22.1 System Clock Input Timing

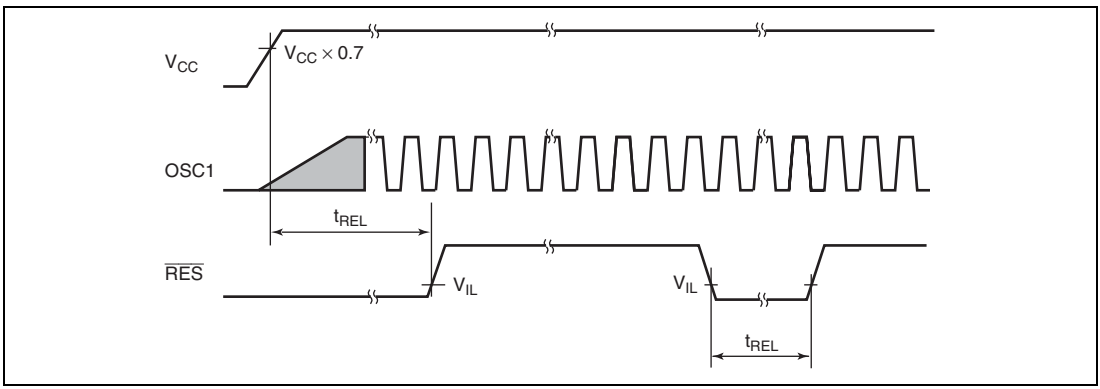


Figure 22.2 $\overline{\text{RES}}$ Low Width Timing

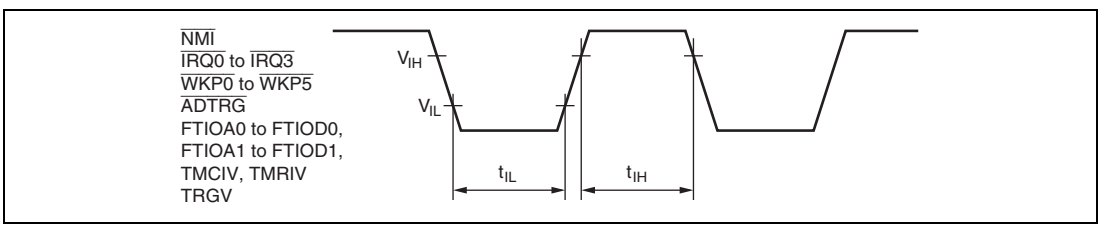


Figure 22.3 Input Timing

A.3 Number of Execution States

The status of execution for each instruction of the H8/300H CPU and the method of calculating the number of states required for instruction execution are shown below. Table A.4 shows the number of cycles of each type occurring in each instruction, such as instruction fetch and data read/write. Table A.3 shows the number of states required for each cycle. The total number of states required for execution of an instruction can be calculated by the following expression:

$$\text{Execution states} = I \times S_I + J \times S_J + K \times S_K + L \times S_L + M \times S_M + N \times S_N$$

Examples: When instruction is fetched from on-chip ROM, and an on-chip RAM is accessed.

BSET #0, @FF00

From table A.4:
I = L = 2, J = K = M = N= 0

From table A.3:
$S_I = 2$, $S_L = 2$

Number of states required for execution = $2 \times 2 + 2 \times 2 = 8$

When instruction is fetched from on-chip ROM, branch address is read from on-chip ROM, and on-chip RAM is used for stack area.

JSR @@ 30

From table A.4:
I = 2, J = K = 1, L = M = N = 0

From table A.3:
$S_I = S_J = S_K = 2$

Number of states required for execution = $2 \times 2 + 1 \times 2 + 1 \times 2 = 8$