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	16MHz
Connectivity	I²C, SCI
Peripherals	PWM, WDT
Number of I/O	27
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
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°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/hd64n3664fpiv

Table 2.2 Data Transfer Instructions

Instruction	Size*	Function
MOV	B/W/L	(EAs) → Rd, Rs → (EAd) Moves data between two general registers or between a general register and memory, or moves immediate data to a general register.
MOVFPE	B	(EAs) → Rd, Cannot be used in this LSI.
MOVTPE	B	Rs → (EAs) Cannot be used in this LSI.
POP	W/L	@SP+ → Rn Pops a general register from the stack. POP.W Rn is identical to MOV.W @SP+, Rn. POP.L ERn is identical to MOV.L @SP+, ERn.
PUSH	W/L	Rn → @–SP Pushes a general register onto the stack. PUSH.W Rn is identical to MOV.W Rn, @–SP. PUSH.L ERn is identical to MOV.L ERn, @–SP.

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

Table 2.4 Logic Operations Instructions

Instruction	Size*	Function
AND	B/W/L	Rd ∧ Rs → Rd, Rd ∧ #IMM → Rd Performs a logical AND operation on a general register and another general register or immediate data.
OR	B/W/L	Rd ∨ Rs → Rd, Rd ∨ #IMM → Rd Performs a logical OR operation on a general register and another general register or immediate data.
XOR	B/W/L	Rd ⊕ Rs → Rd, Rd ⊕ #IMM → Rd Performs a logical exclusive OR operation on a general register and another general register or immediate data.
NOT	B/W/L	¬ (Rd) → (Rd) Takes the one's complement of general register contents.

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

Table 2.5 Shift Instructions

Instruction	Size*	Function
SHAL SHAR	B/W/L	Rd (shift) → Rd Performs an arithmetic shift on general register contents.
SHLL SHLR	B/W/L	Rd (shift) → Rd Performs a logical shift on general register contents.
ROTL ROTR	B/W/L	Rd (rotate) → Rd Rotates general register contents.
ROTXL ROTXR	B/W/L	Rd (rotate) → Rd Rotates general register contents through the carry flag.

Note: * Refers to the operand size.
B: Byte
W: Word
L: Longword

(5) Absolute Address—@aa:8, @aa:16, @aa:24

The instruction code contains the absolute address of a memory operand. The absolute address may be 8 bits long (@aa:8), 16 bits long (@aa:16), 24 bits long (@aa:24)

For an 8-bit absolute address, the upper 16 bits are all assumed to be 1 (H'FFFF). For a 16-bit absolute address the upper 8 bits are a sign extension. A 24-bit absolute address can access the entire address space.

The access ranges of absolute addresses for the group of this LSI are those shown in table 2.11, because the upper 8 bits are ignored.

Table 2.11 Absolute Address Access Ranges

Absolute Address	Access Range
8 bits (@aa:8)	H'FF00 to H'FFFF
16 bits (@aa:16)	H'0000 to H'FFFF
24 bits (@aa:24)	H'0000 to H'FFFF

(6) Immediate—#xx:8, #xx:16, or #xx:32

The instruction contains 8-bit (#xx:8), 16-bit (#xx:16), or 32-bit (#xx:32) immediate data as an operand.

The ADDS, SUBS, INC, and DEC instructions contain immediate data implicitly. Some bit manipulation instructions contain 3-bit immediate data in the instruction code, specifying a bit number. The TRAPA instruction contains 2-bit immediate data in its instruction code, specifying a vector address.

(7) Program-Counter Relative—@(d:8, PC) or @(d:16, PC)

This mode is used in the BSR instruction. An 8-bit or 16-bit displacement contained in the instruction is sign-extended and added to the 24-bit PC contents to generate a branch address. The PC value to which the displacement is added is the address of the first byte of the next instruction, so the possible branching range is –126 to +128 bytes (–63 to +64 words) or –32766 to +32768 bytes (–16383 to +16384 words) from the branch instruction. The resulting value should be an even number.

7.4.2 Erase/Erase-Verify

When erasing flash memory, the erase/erase-verify flowchart shown in figure 7.4 should be followed.

1. Prewriting (setting erase block data to all 0s) is not necessary.
2. Erasing is performed in block units. Make only a single-bit specification in the erase block register (EBR1). To erase multiple blocks, each block must be erased in turn.
3. The time during which the E bit is set to 1 is the flash memory erase time.
4. The watchdog timer (WDT) is set to prevent overerasing due to program runaway, etc. An overflow cycle of approximately 19.8 ms is allowed.
5. For a dummy write to a verify address, write 1-byte data H'FF to an address whose lower two bits are B'00. Verify data can be read in longwords from the address to which a dummy write was performed.
6. If the read data is not erased successfully, set erase mode again, and repeat the erase/erase-verify sequence as before. The maximum number of repetitions of the erase/erase-verify sequence is 100.

7.4.3 Interrupt Handling when Programming/Erasing Flash Memory

All interrupts, including the NMI interrupt, are disabled while flash memory is being programmed or erased, or while the boot program is executing, for the following three reasons:

1. Interrupt during programming/erasing may cause a violation of the programming or erasing algorithm, with the result that normal operation cannot be assured.
2. If interrupt exception handling starts before the vector address is written or during programming/erasing, a correct vector cannot be fetched and the CPU malfunctions.
3. If an interrupt occurs during boot program execution, normal boot mode sequence cannot be carried out.

Section 8 RAM

This LSI has an on-chip high-speed static RAM. The RAM is connected to the CPU by a 16-bit data bus, enabling two-state access by the CPU to both byte data and word data.

Product Classification		RAM Size	RAM Address
Flash memory version (F-ZTAT™ version)	H8/3664N	2 kbytes	H'F780 to H'FF7F*
	H8/3664F	2 kbytes	H'F780 to H'FF7F*
Mask ROM version	H8/3664	1 kbyte	H'FB80 to H'FF7F
	H8/3663	1 kbyte	H'FB80 to H'FF7F
	H8/3662	512 bytes	H'FD80 to H'FF7F
	H8/3661	512 bytes	H'FD80 to H'FF7F
	H8/3660	512 bytes	H'FD80 to H'FF7F

Note: * Area H'F780 to H'FB7F must not be accessed.

12.4 Operation

The timer W has the following operating modes.

- Normal Operation
- PWM Operation

12.4.1 Normal Operation

TCNT performs free-running or periodic counting operations. After a reset, TCNT is set as a free-running counter. When the CTS bit in TMRW is set to 1, TCNT starts incrementing the count. When the count overflows from H'FFFF to H'0000, the OVF flag in TSRW is set to 1. If the OVIE in TIERW is set to 1, an interrupt request is generated. Figure 12.2 shows free-running counting.

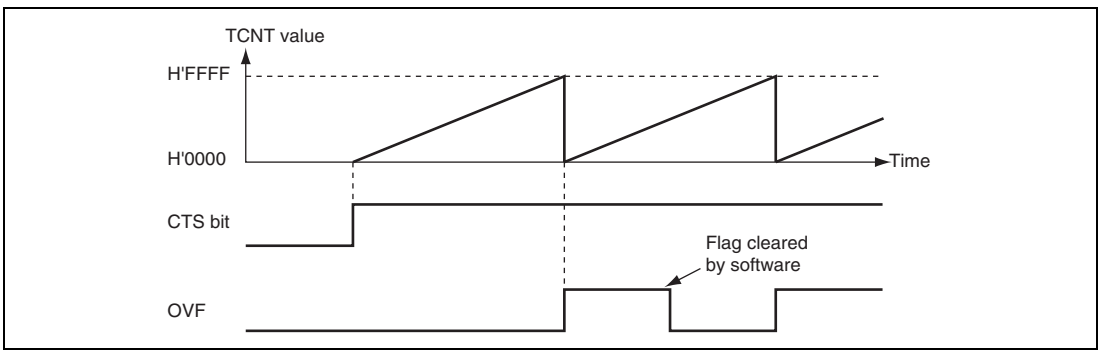


Figure 12.2 Free-Running Counter Operation

Periodic counting operation can be performed when GRA is set as an output compare register and bit CCLR in TCRW is set to 1. When the count matches GRA, TCNT is cleared to H'0000, the IMFA flag in TSRW is set to 1. If the corresponding IMIEA bit in TIERW is set to 1, an interrupt request is generated. TCNT continues counting from H'0000. Figure 12.3 shows periodic counting.

Section 13 Watchdog Timer

The watchdog timer is an 8-bit timer that can generate an internal reset signal for this LSI if a system crash prevents the CPU from writing to the timer counter, thus allowing it to overflow.

The block diagram of the watchdog timer is shown in figure 13.1.

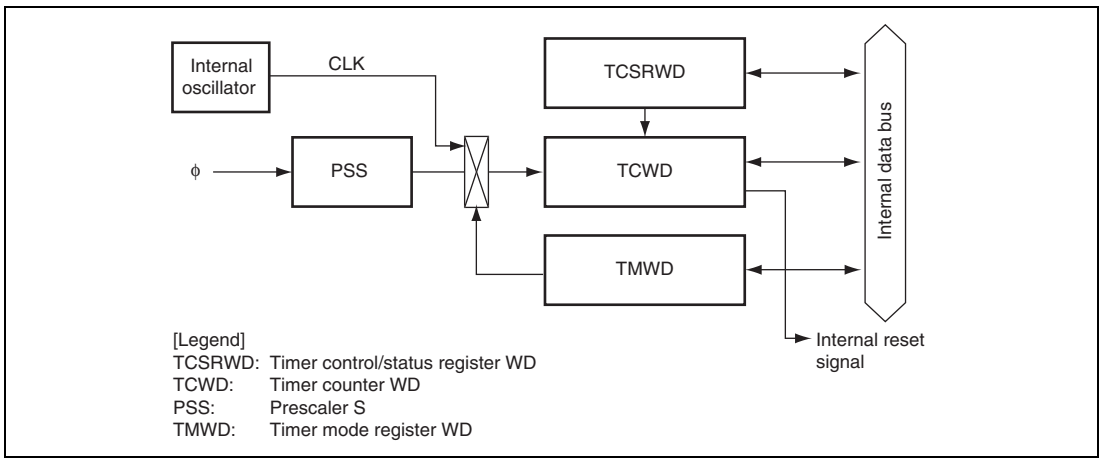


Figure 13.1 Block Diagram of Watchdog Timer

13.1 Features

- Selectable from nine counter input clocks.
  Eight clock sources (ϕ/64, ϕ/128, ϕ/256, ϕ/512, ϕ/1024, ϕ/2048, ϕ/4096, and ϕ/8192) or the internal oscillator can be selected as the timer-counter clock. When the internal oscillator is selected, it can operate as the watchdog timer in any operating mode.
- Reset signal generated on counter overflow
  An overflow period of 1 to 256 times the selected clock can be set.

13.2 Register Descriptions

The watchdog timer has the following registers.

- Timer control/status register WD (TCSRWD)
- Timer counter WD (TCWD)
- Timer mode register WD (TMWD)

14.4.3 Data Transmission

Figure 14.5 shows an example of operation for transmission in asynchronous mode. In transmission, the SCI3 operates as described below.

1. The SCI3 monitors the TDRE flag in SSR. If the flag is cleared to 0, the SCI3 recognizes that data has been written to TDR, and transfers the data from TDR to TSR.
2. After transferring data from TDR to TSR, the SCI3 sets the TDRE flag to 1 and starts transmission. If the TIE bit is set to 1 at this time, a TXI interrupt request is generated. Continuous transmission is possible because the TXI interrupt routine writes next transmit data to TDR before transmission of the current transmit data has been completed.
3. The SCI3 checks the TDRE flag at the timing for sending the stop bit.
4. If the TDRE flag is 0, the data is transferred from TDR to TSR, the stop bit is sent, and then serial transmission of the next frame is started.
5. If the TDRE flag is 1, the TEND flag in SSR is set to 1, the stop bit is sent, and then the “mark state” is entered, in which 1 is output. If the TEIE bit in SCR3 is set to 1 at this time, a TEI interrupt request is generated.
6. Figure 14.6 shows a sample flowchart for transmission in asynchronous mode.

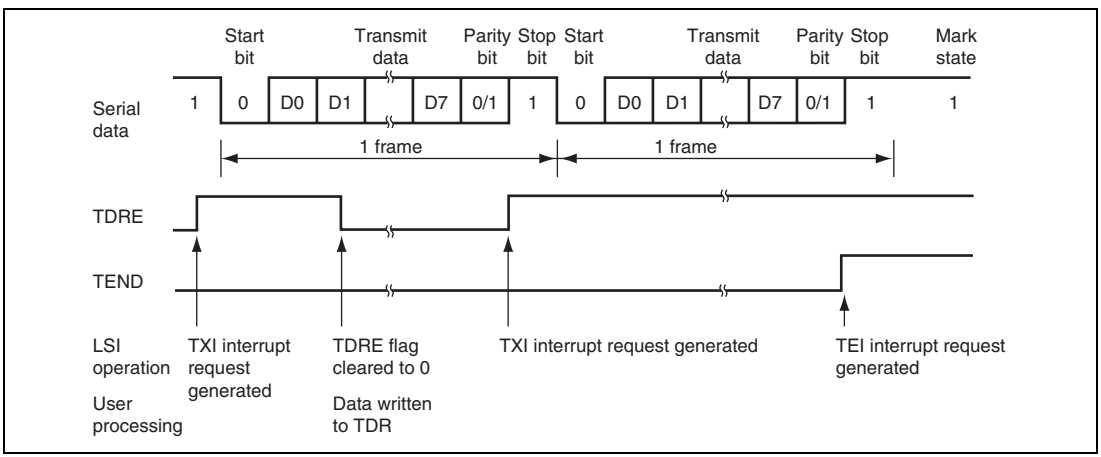
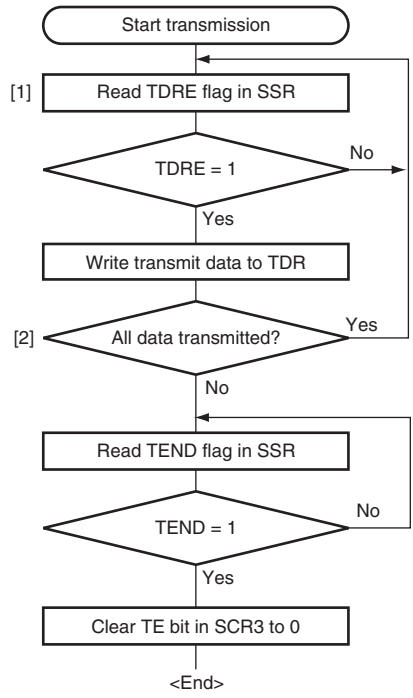


Figure 14.5 Example SCI3 Operation in Transmission in Asynchronous Mode (8-Bit Data, Parity, One Stop Bit)



[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 and clocks are output to start the data transmission.

[2] To continue serial transmission, be sure to 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

Figure 14.11 Sample Serial Transmission Flowchart (Clocked Synchronous Mode)

9. Write the transmit data to ICDR. As indicating the end of the transfer, and so the IRIC flag is cleared to 0. Perform the ICDR write and the IRIC flag clearing sequentially, just as in the step [6]. Transmission of the next frame is performed in synchronization with the internal clock.
10. When one frame of data has been transmitted, the IRIC flag is set to 1 at the rise of the 9th transmit clock pulse. After one frame has been transmitted SCL is automatically fixed low in synchronization with the internal clock until the next transmit data is written.
11. Read the ACKB bit in ICSR. Confirm that the slave device has been acknowledged (ACKB bit is 0). When there is data to be transmitted, go to the step [9] to continue next transmission. When the slave device has not acknowledged (ACKB bit is set to 1), operate the step [12] to end transmission.
12. Clear the IRIC flag to 0. And write 0 to BBSY and SCP in ICCR. This changes SDA from low to high when SCL is high, and generates the stop condition.

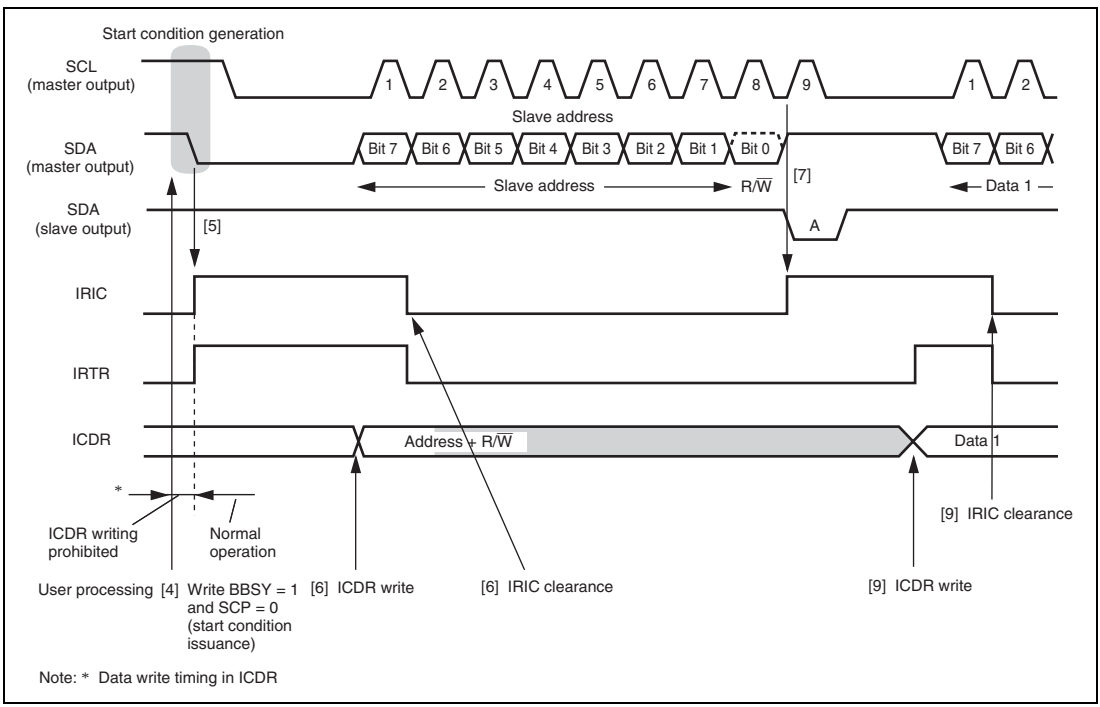


Figure 15.5 Master Transmit Mode Operation Timing Example (MLS = WAIT = 0)

17.4.6 Slave Addressing

The EEPROM device receives a 7-bit slave address and a 1-bit R/$\overline{W}$ code following the generation of the start conditions. The EEPROM enables the chip for a read or a write operation with this operation.

The slave address consists of a former 4-bit device code and latter 3-bit slave address as shown in table 17.2. The device code is used to distinguish device type and this LSI uses "1010" fixed code in the same manner as in a general-purpose EEPROM. The slave address code selects one device out of all devices with device code 1010 (8 devices in maximum) which are connected to the I^2C bus. This means that the device is selected if the inputted slave address code received in the order of A2, A1, A0 is equal to the corresponding slave address reference register (ESAR).

The slave address code is stored in the address H'FF09 in the EEPROM. It is transferred to ESAR from the slave address register in the memory array during 10 ms after the reset is released. An access to the EEPROM is not allowed during transfer.

The initial value of the slave address code written in the EEPROM is H'00. It can be written in the range of H'00 to H'07. Be sure to write the data by the byte write method.

The next one bit of the slave address is the R/$\overline{W}$ code. 0 is for a write and 1 is for a read.

The EEPROM turns to a standby state if the device code is not "1010" or slave address code doesn't coincide.

Table 17.2 Slave Addresses

Bit	Bit name	Initial Value	Setting Value	Remarks
7	Device code D3	—	1	
6	Device code D2	—	0	
5	Device code D1	—	1	
4	Device code D0	—	0	
3	Slave address code A2	0	A2	The initial value can be changed
2	Slave address code A1	0	A1	The initial value can be changed
1	Slave address code A0	0	A0	The initial value can be changed

20.2.7 EEPROM Characteristics

Table 20.9 EEPROM Characteristics

V_{CC} = 3.0 V to 5.5 V, V_{SS} = 0.0 V, T_a = –20°C to +75°C, unless otherwise specified.

Item	Symbol	Test Condition	Values Min	Typ	Max	Unit	Reference Figure
SCL input cycle time	t_{SCL}		2500	—		ns	Figure 20.7
SCL input high pulse width	t_{SCLH}		600	—	—	µs	
SCL input low pulse width	t_{SCLL}		1200	—	—	ns	
SCL, SDA input spike pulse removal time	t_{SP}		—	—	50	ns	
SDA input bus-free time	t_{BUF}		1200	—	—	ns	
Start condition input hold time	t_{STAH}		600	—	—	ns	
Retransmit start condition input setup time	t_{STAS}		600	—	—	ns	
Stop condition input setup time	t_{STOS}		600	—	—	ns	
Data input setup time	t_{SDAS}		160	—	—	ns	
Data input hold time	t_{SDAH}		0	—	—	ns	
SCL, SDA input fall time	t_{Sf}		—	—	300	ns	
SDA input rise time	t_{Sr}		—	—	300	ns	
Data output hold time	t_{DH}		50	—	—	ns	
SCL, SDA capacitive load	C_b		0	—	400	pF	
Access time	t_{AA}		100	—	900	ns	
Cycle time at writing*	t_{WC}		—	—	10	ms	
Reset release time	t_{RES}		—	—	13	ms	

Note: * Cycle time at writing is a time from the stop condition to write completion (internal control).

20.3.3 AC Characteristics

Table 20.11 AC Characteristics

V_{CC} = 2.7 V to 5.5 V, V_{SS} = 0.0 V, T_a = –20°C to +75°C, unless otherwise specified.

Item	Symbol	Applicable Pins	Test Condition	Values Min	Typ	Max	Unit	Reference Figure
System clock oscillation frequency	f_{OSC}	OSC1, OSC2	V_{CC} = 4.0 V to 5.5 V	2.0	—	16.0	MHz	*1
				2.0		10.0		
System clock (ϕ) cycle time	t_{cyc}			1	—	64	t_{OSC}	*2
				—	—	12.8	µs	
Subclock oscillation frequency	f_W	X1, X2		—	32.768	—	kHz	
Watch clock (ϕ_W) cycle time	t_W	X1, X2		—	30.5	—	µs	
Subclock (ϕ_{SUB}) cycle time	t_{subcyc}			2	—	8	t_W	*2
Instruction cycle time				2	—	—	t_{cyc} t_{subcyc}	
Oscillation stabilization time (crystal resonator)	t_{rc}	OSC1, OSC2		—	—	10.0	ms	
Oscillation stabilization time (ceramic resonator)	t_{rc}	OSC1, OSC2		—	—	5.0	ms	
Oscillation stabilization time	t_{rcx}	X1, X2		—	—	2.0	s	
External clock high width	t_{CPH}	OSC1	V_{CC} = 4.0 V to 5.5 V	25.0	—	—	ns	Figure 20.1
				40.0	—	—	ns	
External clock low width	t_{CPL}	OSC1	V_{CC} = 4.0 V to 5.5 V	25.0	—	—	ns	
				40.0	—	—	ns	
External clock rise time	t_{CPr}	OSC1	V_{CC} = 4.0 V to 5.5 V	—	—	10.0	ns	
				—	—	15.0	ns	
External clock fall time	t_{CPf}	OSC1	V_{CC} = 4.0 V to 5.5 V	—	—	10.0	ns	
				—	—	15.0	ns	

8. Block transfer instructions

Mnemonic		Operand Size	Addressing Mode and Instruction Length (bytes)									Operation	Condition Code						No. of States*1	
			#xx	Rn	@ERn	@(d, ERn)	@–ERn/@ERn+	@aa	@(d, PC)	@@aa	|		I	H	N	Z	V	C	Normal	Advanced
EEPMOV	EEPMOV. B	—									4	if R4L ≠ 0 then repeat @R5 → @R6 R5+1 → R5 R6+1 → R6 R4L–1 → R4L until R4L=0 else next	—	—	—	—	—	—	8+ 4n*2	
	EEPMOV. W	—									4	if R4 ≠ 0 then repeat @R5 → @R6 R5+1 → R5 R6+1 → R6 R4–1 → R4 until R4=0 else next	—	—	—	—	—	—	8+ 4n*2	

Notes: 1. The number of states in cases where the instruction code and its operands are located in on-chip memory is shown here. For other cases see appendix A.3, Number of Execution States.

2. n is the value set in register R4L or R4.

(1) Set to 1 when a carry or borrow occurs at bit 11; otherwise cleared to 0.

(2) Set to 1 when a carry or borrow occurs at bit 27; otherwise cleared to 0.

(3) Retains its previous value when the result is zero; otherwise cleared to 0.

(4) Set to 1 when the adjustment produces a carry; otherwise retains its previous value.

(5) The number of states required for execution of an instruction that transfers data in synchronization with the E clock is variable.

(6) Set to 1 when the divisor is negative; otherwise cleared to 0.

(7) Set to 1 when the divisor is zero; otherwise cleared to 0.

(8) Set to 1 when the quotient is negative; otherwise cleared to 0.

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 = $2 \times 2 + 1 \times 2 + 1 \times 2 = 8$

Table A.4 Number of Cycles in Each Instruction

Instruction	Mnemonic	Instruction Fetch I	Branch Addr. Read J	Stack Operation K	Byte Data Access L	Word Data Access M	Internal Operation N
ADD	ADD.B #xx:8, Rd	1					
	ADD.B Rs, Rd	1					
	ADD.W #xx:16, Rd	2					
	ADD.W Rs, Rd	1					
	ADD.L #xx:32, ERd	3					
	ADD.L ERs, ERd	1					
ADDS	ADDS #1/2/4, ERd	1					
ADDX	ADDX #xx:8, Rd	1					
	ADDX Rs, Rd	1					
AND	AND.B #xx:8, Rd	1					
	AND.B Rs, Rd	1					
	AND.W #xx:16, Rd	2					
	AND.W Rs, Rd	1					
	AND.L #xx:32, ERd	3					
	AND.L ERs, ERd	2					
ANDC	ANDC #xx:8, CCR	1					
BAND	BAND #xx:3, Rd	1					
	BAND #xx:3, @ERd	2			1		
	BAND #xx:3, @aa:8	2			1		
Bcc	BRA d:8 (BT d:8)	2					
	BRN d:8 (BF d:8)	2					
	BHI d:8	2					
	BLS d:8	2					
	BCC d:8 (BHS d:8)	2					
	BCS d:8 (BLO d:8)	2					
	BNE d:8	2					
	BEQ d:8	2					
	BVC d:8	2					
	BVS d:8	2					
	BPL d:8	2					
	BMI d:8	2					
	BGE d:8	2					

B.2 Port States in Each Operating State

Port	Reset	Sleep	Subsleep	Standby	Subactive	Active
P17 to P14, P12 to P10	High impedance	Retained	Retained	High impedance*	Functioning	Functioning
P22 to P20	High impedance	Retained	Retained	High impedance*	Functioning	Functioning
P57 to P50 (P55 to P50 for H8/3664N)	High impedance	Retained	Retained	High impedance	Functioning	Functioning
P76 to P74	High impedance	Retained	Retained	High impedance	Functioning	Functioning
P87 to P80	High impedance	Retained	Retained	High impedance	Functioning	Functioning
PB7 to PB0	High impedance	High impedance	High impedance	High impedance	High impedance	High impedance

Note: * High level output when the pull-up MOS is in on state.