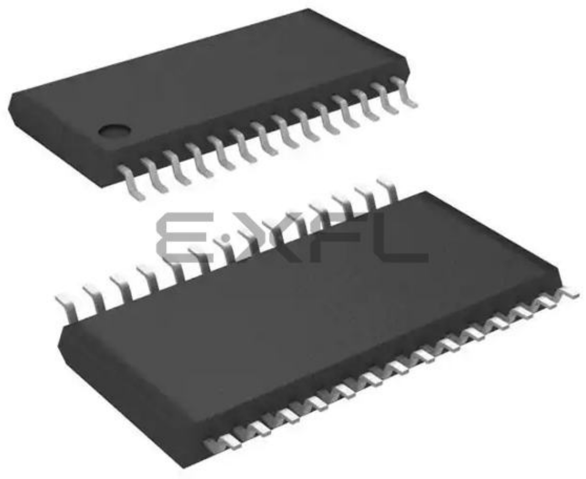

What is "Embedded - Microcontrollers"?

"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	Active
Core Processor	S08
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	22
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	28-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08sl16ctl

Section Number	Title	Page
14.3	Functional Description	261
14.3.1	Baud Rate Generation	261
14.3.2	Transmitter Functional Description	262
14.3.3	Receiver Functional Description	263
14.3.4	Interrupts and Status Flags	265
14.3.5	Additional SCI Functions	266

Chapter 15
Real-Time Counter (S08RTCV1)

15.1 Introduction 269
15.1.1 Features 272
15.1.2 Modes of Operation 272
15.1.3 Block Diagram 273
15.2 External Signal Description 273
15.3 Register Definition 273
15.3.1 RTC Status and Control Register (RTCSC) 274
15.3.2 RTC Counter Register (RTCCNT) 275
15.3.3 RTC Modulo Register (RTCMOD) 275
15.4 Functional Description 275
15.4.1 RTC Operation Example 276
15.5 Initialization/Application Information 277

Chapter 16
Timer Pulse-Width Modulator (S08TPMV2)

16.1 Introduction 279
16.1.1 Features 281
16.1.2 Modes of Operation 281
16.1.3 Block Diagram 282
16.2 Signal Description 284
16.2.1 Detailed Signal Descriptions 284
16.3 Register Definition 288
16.3.1 TPM Status and Control Register (TPMxSC) 288
16.3.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL) 289
16.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL) 290
16.3.4 TPM Channel n Status and Control Register (TPMxCnSC) 291
16.3.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL) 293
16.4 Functional Description 294
16.4.1 Counter 295
16.4.2 Channel Mode Selection 297
16.5 Reset Overview 300
16.5.1 General 300
16.5.2 Description of Reset Operation 300

Table 4-3. High-Page Register Summary (Sheet 2 of 2)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1845	PTAPS	0	0	0	0	PTAPS3	PTAPS2	PTAPS1	PTAPS0
0x1846	PTAES	0	0	0	0	PTAES3	PTAES2	PTAES1	PTAES0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1849	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x184A	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x184B	Reserved	—	—	—	—	—	—	—	—
0x184C	PTBSC	0	0	0	0	PTBIF	PTBACK	PTBIE	PTBMOD
0x184D	PTBPS	0	0	0	0	PTBPS3	PTBPS2	PTBPS1	PTBPS0
0x184E	PTBES	0	0	0	0	PTBES3	PTBES2	PTBES1	PTBES0
0x184F	Reserved	—	—	—	—	—	—	—	—
0x1850	PTCPE	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1851	PTCSE	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x1852	PTCDS	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x1853	Reserved	—	—	—	—	—	—	—	—
0x1854	PTCSC	0	0	0	0	PTCIF	PTCACK	PTCIE	PTCMOD
0x1855	PTCPS	PTCPS7	PTCPS6	PTCPS5	PTCPS4	PTCPS3	PTCPS2	PTCPS1	PTCPS0
0x1856	PTCES	PTCES7	PTCES6	PTCES5	PTCES4	PTCES3	PTCES2	PTCES1	PTCES0
0x1857	Reserved	—	—	—	—	—	—	—	—
0x1858–0x18FF	Reserved	— —	— —	— —	— —	— —	— —	— —	— —

The status flag corresponding to the interrupt source must be acknowledged (cleared) before returning from the ISR. Typically, the flag is cleared at the beginning of the ISR so that if another interrupt is generated by this same source, it will be registered so it can be serviced after completion of the current ISR.

5.5.2 Interrupt Vectors, Sources, and Local Masks

Table 5-2 provides a summary of all interrupt sources. Higher-priority sources are located toward the bottom of the table. The high-order byte of the address for the interrupt service routine is located at the first address in the vector address column, and the low-order byte of the address for the interrupt service routine is located at the next higher address.

When an interrupt condition occurs, an associated flag bit becomes set. If the associated local interrupt enable is 1, an interrupt request is sent to the CPU. Within the CPU, if the global interrupt mask (I bit in the CCR) is 0, the CPU will finish the current instruction; stack the PCL, PCH, X, A, and CCR CPU registers; set the I bit; and then fetch the interrupt vector for the highest priority pending interrupt. Processing then continues in the interrupt service routine.

Chapter 7
Central Processor Unit (S08CPUV3)

7.1 Introduction

This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family. For a more detailed discussion, refer to the *HCS08 Family Reference Manual, volume 1,* Freescale Semiconductor document order number HCS08RMV1/D.

The HCS08 CPU is fully source- and object-code-compatible with the M68HC08 CPU. Several instructions and enhanced addressing modes were added to improve C compiler efficiency and to support a new background debug system which replaces the monitor mode of earlier M68HC08 microcontrollers (MCU).

7.1.1 Features

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- All registers and memory are mapped to a single 64-Kbyte address space
- 16-bit stack pointer (any size stack anywhere in 64-Kbyte address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
  - Inherent — Operands in internal registers
  - Relative — 8-bit signed offset to branch destination
  - Immediate — Operand in next object code byte(s)
  - Direct — Operand in memory at 0x0000–0x00FF
  - Extended — Operand anywhere in 64-Kbyte address space
  - Indexed relative to H:X — Five submodes including auto increment
  - Indexed relative to SP — Improves C efficiency dramatically
- Memory-to-memory data move instructions with four address mode combinations
- Overflow, half-carry, negative, zero, and carry condition codes support conditional branching on the results of signed, unsigned, and binary-coded decimal (BCD) operations
- Efficient bit manipulation instructions
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- STOP and WAIT instructions to invoke low-power operating modes

	7	6	5	4	3	2	1	0
R W	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
Reset:	0	0	0	0	0	0	0	0

Figure 10-10. Configuration Register (ADCCFG)

Table 10-5. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	**Low Power Configuration** — ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: {FC31}The power is reduced at the expense of maximum clock speed.
6:5 ADIV	**Clock Divide Select** — ADIV select the divide ratio used by the ADC to generate the internal clock ADCK. Table 10-6 shows the available clock configurations.
4 ADLSMP	**Long Sample Time Configuration** — ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time
3:2 MODE	**Conversion Mode Selection** — MODE bits are used to select between 10- or 8-bit operation. See Table 10-7.
1:0 ADICLK	**Input Clock Select** — ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 10-8.

Table 10-6. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 10-7. Conversion Modes

MODE	Mode Description
00	8-bit conversion (N=8)
01	Reserved
10	10-bit conversion (N=10)
11	Reserved

For clarification, in this document, “command” messages will refer to any message frame where the SLIC module is receiving data bytes and “request” messages refer to message frames where the SLIC module will be expected to transmit data bytes. This is a generic description and should not be confused with the terminology in the LIN specification. The LIN use of the terms “command” and “request” have the same basic meaning, but are limited in scope to specific identifier values of 0x3C and 0x3D. In the SLIC module documentation, these terms have been used to describe these functional types of messages, regardless of the specific identifier value used.

12.6.7.2 Possible Errors on Message Headers

Possible errors on message headers are:

- Identifier-Parity-Error
- Byte Framing Error

12.6.8 Handling Command Message Frames

Figure 12-15 shows how to handle command message frames, where the SLIC module is receiving data from the master node.

Command message frames refer to LIN messages frames where the master node is “commanding” the slave node to do something. The implication is that the slave will then be receiving data from the master for this message frame. This can be a standard LIN message frame of 1–8 data bytes, a reserved LIN system message (using 0x3C identifier), or an extended command message frame utilizing the reserved 0x3E user defined identifier or perhaps the 0x3F LIN reserved extended identifier. The SLIC module is capable of handling message frames containing up to 64 bytes of data, while still automatically calculating and/or verifying the checksum.

12.6.8.1 Standard Command Message Frames

After the application software has read the incoming identifier and determined that it is a valid identifier which cannot be ignored using IMSG, it must determine if this message frame is a command message frame or a request message frame. (i.e., should the application receive data from the master or send data back to the master?)

The first case, shown in Figure 12-15 deals with command messages, where the SLIC will be receiving data from the master node. If the received identifier corresponds to a standard LIN command frame (i.e., 1–8 data bytes), the user must then write the number of bytes (determined by the system designer and directly linked with this particular identifier) corresponding to the length of the message frame into SLCDLC. The two most significant bits of this register are used for special control bits describing the nature of this message frame.

unmasked, after 8 bytes are received or an error is detected. At this interrupt, the SLCSV will indicate an error condition (in case of byte framing error, idle bus) or that the receive buffer is full. If the data is successfully received, the user must then empty the buffer by reading SLCD7-SLCD0 and then subtract 8 from the software byte count. When this software counter reaches 8 or fewer, the remaining data bytes will fit in the buffer and only one interrupt should occur. At this time, the final interrupt may be handled normally, continuing to use the software counter to read the proper number of bytes from the appropriate SLCD registers.

NOTE

Do not write SLCDLC more than one time per LIN message frame. The SLIC tracks the number of sent or received bytes based on the value written to this register at the beginning of the data field and rewriting this register will corrupt the checksum calculation and cause unpredictable behavior in the SLIC module. The application software must track the number of sent or received bytes to know what the current byte count in the SLIC is. If programming in C, make sure to use the VOLATILE modifier on this variable (or make it a global variable) to ensure that it keeps its value between interrupts.

12.6.8.3 Possible Errors on Command Message Data

Possible errors on command message data are:

- Byte Framing Error
- Checksum-Error (LIN specified error)
- No-Bus-Activity (LIN specified error)
- Receiver Buffer Overrun Error

12.6.9 Handling Request LIN Message Frames

Figure 12-16 shows how to handle request message frames, where the SLIC module is sending data to the master node.

Request message frames refer to LIN messages frames where the master node is “requesting” the slave node to supply information. The implication is that the slave will then be transmitting data to the master for this message frame. This can be a standard LIN message frame of 1–8 data bytes, a reserved LIN system message (using 0x3D identifier), or an extended request message frame utilizing the reserved 0x3E identifier or perhaps the 0x3F LIN reserved extended identifier. The SLIC module is capable of handling request message frames containing up to 64 bytes of data, while still automatically calculating and/or verifying the checksum.

12.6.9.1 Standard Request Message Frames

Dealing with request messages with the SLIC is very similar to dealing with command messages, with one important difference. Because the SLIC is now to be transmitting data in the LIN message frame, the user software must load the data to be transmitted into the message buffer prior to initiating the transmission.

Because perfect conditions are almost impossible to attain, more robust values must be chosen for bit rates. For reliable communication, it is best to ensure that a bit time is no smaller 2x–3x longer than the filter delay on the digital receive filter. This is true in LIN or BTM mode and ensures that valid data bits which have been shortened due to ground shift, asymmetrical rise and fall times, etc., are accepted by the filter without exception. This would translate to 2x to 3x reduction in the maximum speeds shown in Table 12-14. Recommended maximum bit rates are shown in Table 12-15, and ensure that a single bit time is at least twice the length of one filter delay value. If system noise is not adequately filtered out it might be necessary to change the prescaler of the filter and lower the bit rate of the communication. If valid communications are being absorbed by the filter, corrective action must be taken to ensure that either the bit rate is reduced or whatever physical fault is causing bit times to shorten is corrected (ground offset, asymmetrical rise/fall times, insufficient physical layer supply voltage, etc.).

Table 12-15. Recommended Maximum Bit Rates for BTM Operation Due to Digital Filter

SLIC Clock (MHz)	Maximum BTM Bit Rate (kbps)							
	RXFP = ÷8	RXFP = ÷7	RXFP = ÷6	RXFP = ÷5	RXFP = ÷4	RXFP = ÷3	RXFP = ÷2	RXFP = ÷1
20	78.125	89.286	104.167	120.000	120.000	120.000	120.000	120.000
18	70.313	80.357	93.750	112.500	120.000	120.000	120.000	120.000
16	62.500	71.429	83.333	100.000	120.000	120.000	120.000	120.000
14	54.688	62.500	72.917	87.500	109.375	120.000	120.000	120.000
12	46.875	53.571	62.500	75.000	93.750	120.000	120.000	120.000
10	39.063	44.643	52.083	62.500	78.125	104.167	120.000	120.000
8	31.250	35.714	41.667	50.000	62.500	83.333	120.000	120.000
6	23.438	26.786	31.250	37.500	46.875	62.500	93.750	120.000
4	15.625	17.857	20.833	25.000	31.250	41.667	62.500	120.000
2	7.813	8.929	10.417	12.500	15.625	20.833	31.250	62.500

12.6.17 Oscillator Trimming with SLIC

SLCACT can be used as an indicator of LIN bus activity. SLCACT tells the user that the SLIC is currently processing a message header (therefore synchronizing to the bus) or processing a message frame (including checksum). Therefore, at idle times between message frames or during a message frame which has been marked as a "don't care" by writing IMSG, it is possible to trim the oscillator circuit of the MCU with no impact to the LIN communications.

It is important to note the exact mechanisms with which the SLIC sets and clears SLCACT. Any falling edge which successfully passes through the digital receive filter will cause SLCACT to become set. This might even include noise pulses, if they are of sufficient length to pass through the digital RX filter. Although in these cases SLCACT is becoming set on a noise spike, it is very probable that noise of this nature will cause other system issues as well such as corruption of the message frame. The software can then further qualify if it is appropriate to trim the oscillator.

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMxCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMxCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMxCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMxCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMxCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMxCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

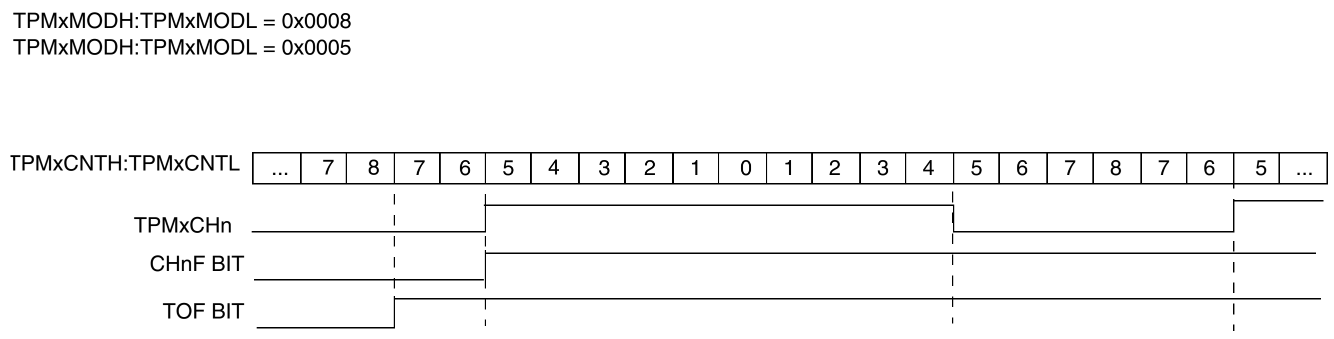

Figure 16-5. High-True Pulse of a Center-Aligned PWM

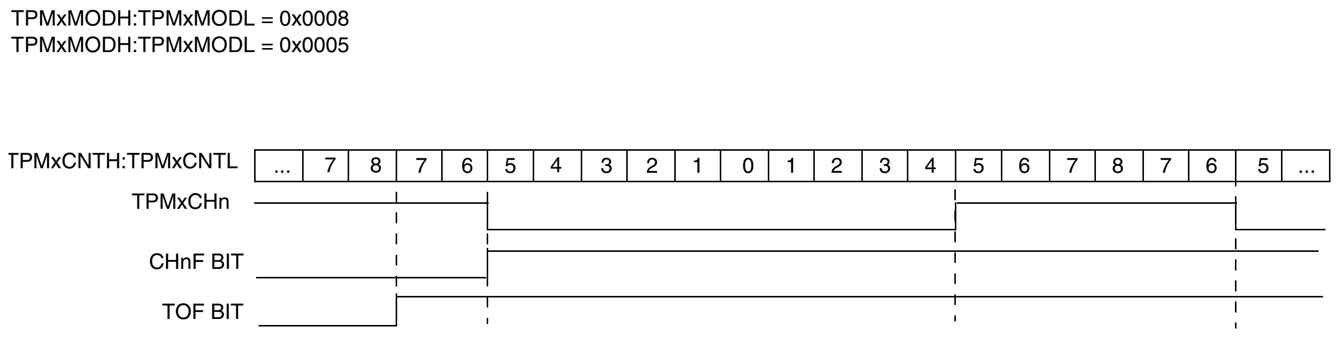

Figure 16-6. Low-True Pulse of a Center-Aligned PWM

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

16.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

16.4.1.1 Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 16-4. After any MCU reset, CLKSB:CLKSA=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKSB:CLKSA field) does not affect the values in the counter or other timer registers.

Table 16-8. TPM Clock Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL and FLL or the PLL and FLL are not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL or FLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user's responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

16.4.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

16.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 16-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.

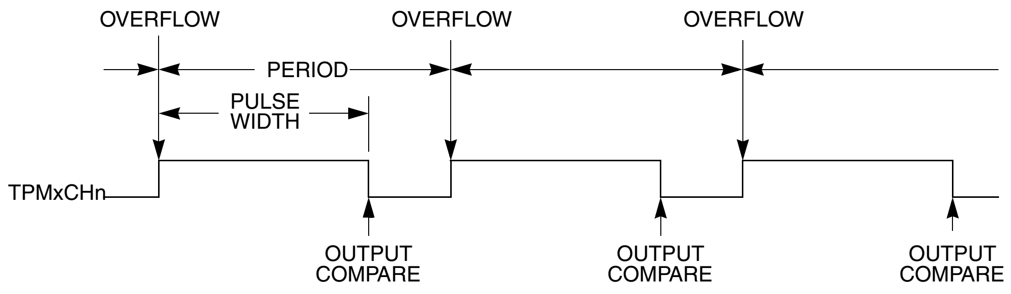

Figure 16-15. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxCnVH and TPMxCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If

BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the frozen TPM counter value.

— This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxSC, TPMxCNTH or TPMxCNTL. Instead, in these conditions the TPM v2 does not clear this read coherency mechanism.

3. Read of TPMxCnVH:L registers (Section 16.3.5, "TPM Channel Value Registers (TPMxCnVH:TPMxCnVL))

   — In TPM v3, any read of TPMxCnVH:L registers during BDM mode returns the value of the TPMxCnVH:L register. In TPM v2, if only one byte of the TPMxCnVH:L registers was read before the BDM mode became active, then any read of TPMxCnVH:L registers during BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the value in the TPMxCnVH:L registers.

   — This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxCnSC. Instead, in this condition the TPM v2 does not clear this read coherency mechanism.

4. Write to TPMxCnVH:L registers

   — Input Capture Mode (Section 16.4.2.1, "Input Capture Mode)

   In this mode the TPM v3 does not allow the writes to TPMxCnVH:L registers. Instead, the TPM v2 allows these writes.

   — Output Compare Mode (Section 16.4.2.2, "Output Compare Mode)

   In this mode and if (CLKSB:CLKSA not = 0:0), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer at the next change of the TPM counter (end of the prescaler counting) after the second byte is written. Instead, the TPM v2 always updates these registers when their second byte is written.

   The following procedure can be used in the TPM v3 to verify if the TPMxCnVH:L registers were updated with the new value that was written to these registers (value in their write buffer).

```
...
write the new value to TPMxCnVH:L;
read TPMxCnVH and TPMxCnVL registers;
while (the read value of TPMxCnVH:L is different from the new value written to
TPMxCnVH:L)
begin
  read again TPMxCnVH and TPMxCnVL;
end
...
```

   In this point, the TPMxCnVH:L registers were updated, so the program can continue and, for example, write to TPMxC0SC without cancelling the previous write to TPMxCnVH:L registers.

   — Edge-Aligned PWM (Section 16.4.2.3, "Edge-Aligned PWM Mode)

   In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the

Figure 17-5 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

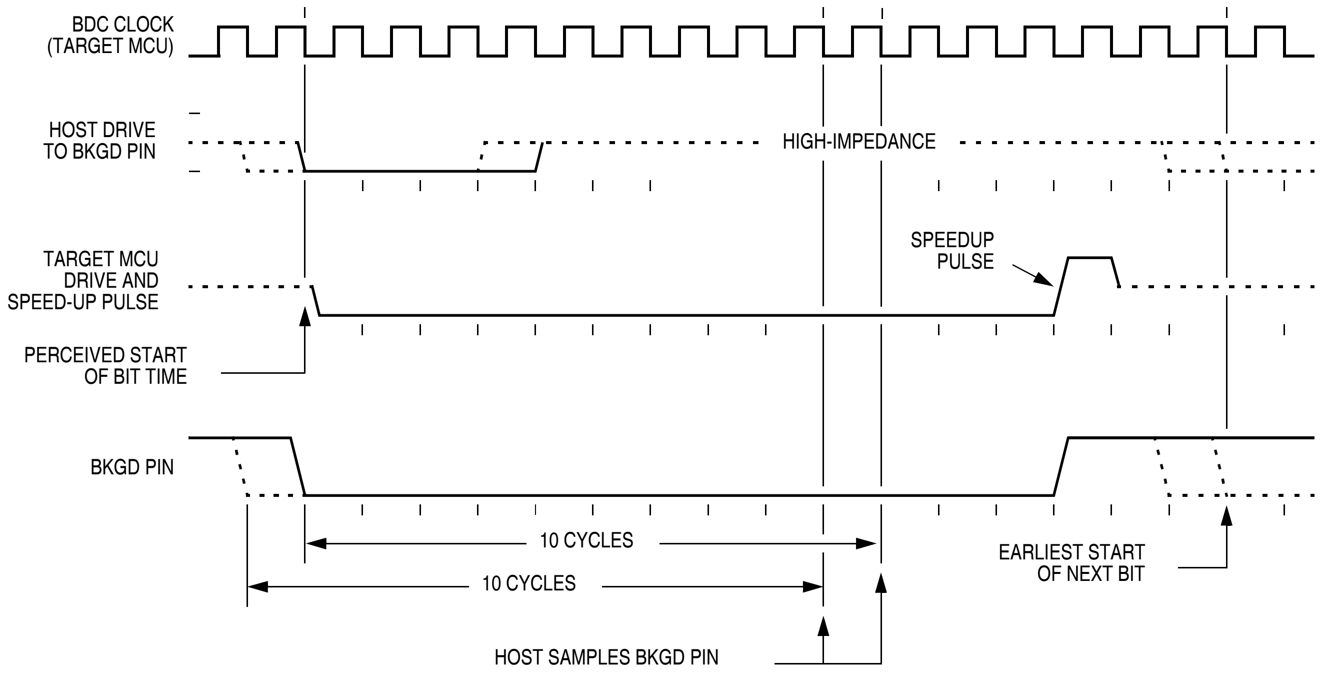

Figure 17-5. BDM Target-to-Host Serial Bit Timing (Logic 0)

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct communications speed to use for BDC communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

The target, upon detecting the SYNC request from the host (which is a much longer low time than would ever occur during normal BDC communications):

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

17.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

17.3 On-Chip Debug System (DBG)

Because HCS08 devices do not have external address and data buses, the most important functions of an in-circuit emulator have been built onto the chip with the MCU. The debug system consists of an 8-stage FIFO that can store address or data bus information, and a flexible trigger system to decide when to capture bus information and what information to capture. The system relies on the single-wire background debug system to access debug control registers and to read results out of the eight stage FIFO.

The debug module includes control and status registers that are accessible in the user's memory map. These registers are located in the high register space to avoid using valuable direct page memory space.

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in Section 17.3.6, "Hardware Breakpoints."

17.3.1 Comparators A and B

Two 16-bit comparators (A and B) can optionally be qualified with the R/W signal and an opcode tracking circuit. Separate control bits allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you to specify that a trigger will occur only if the opcode at the specified address is actually executed as opposed to only being read from memory into the instruction queue. The comparators are also capable of magnitude comparisons to support the inside range and outside range trigger modes. Comparators are disabled temporarily during all BDC accesses.

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If RWAEN = 1 (enabled) and RWA = 0 (write), the CPU's write data bus is used. Otherwise, the CPU's read data bus is used.

The currently selected trigger mode determines what the debugger logic does when a comparator detects a qualified match condition. A match can cause:

- Generation of a breakpoint to the CPU
- Storage of data bus values into the FIFO
- Starting to store change-of-flow addresses into the FIFO (begin type trace)
- Stopping the storage of change-of-flow addresses into the FIFO (end type trace)

17.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGT register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

17.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGT register selects one of nine trigger modes. When TRGSEL = 1 in the DBGT register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGT chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGS. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

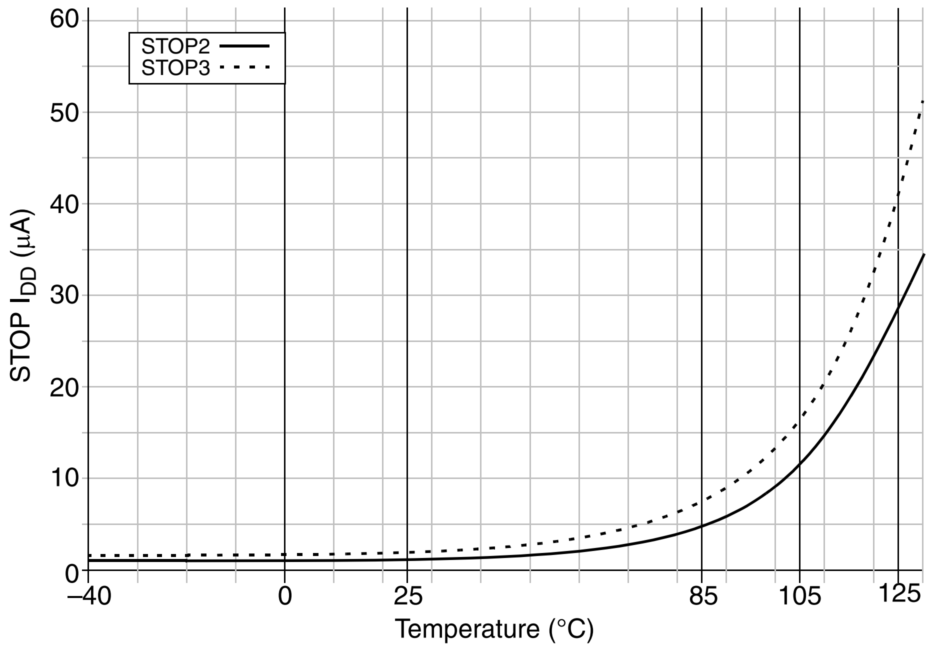

Figure A-7. Typical Stop I_{DD} vs. Temperature (V_{DD} = 5V)

A.8 External Oscillator (XOSC) Characteristics

Table A-8. Oscillator Electrical Specifications (Temperature Range = –40 to 125°C Ambient)

Num	C	Rating	Symbol	Min	Typ[1]	Max	Unit
1	C	Oscillator crystal or resonator (EREFS = 1, ERCLKEN = 1)					
		Low range (RANGE = 0)	f_{lo}	32	—	38.4	kHz
		High range (RANGE = 1) FEE or FBE mode [2]	f_{hi}	1	—	5	MHz
		High range (RANGE = 1, HGO = 1) FBELP mode	$f_{hi\text{-}hgo}$	1	—	16	MHz
		High range (RANGE = 1, HGO = 0) FBELP mode	$f_{hi\text{-}lp}$	1	—	8	MHz
2	—	Load capacitors	C_1, C_2	See crystal or resonator manufacturer's recommendation.			
3	—	Feedback resistor	R_F				MΩ
		Low range (32 kHz to 100 kHz)		—	10	—	
		High range (1 MHz to 16 MHz)		—	1	—	
4	—	Series resistor	R_S				kΩ
		Low range, low gain (RANGE = 0, HGO = 0)		—	0	—	
		Low range, high gain (RANGE = 0, HGO = 1)		—	100	—	
		High range, low gain (RANGE = 1, HGO = 0)		—	0	—	
		High range, high gain (RANGE = 1, HGO = 1)					
		≥ 8 MHz		—	0	0	
		4 MHz		—	0	10	
		1 MHz		—	0	20	