



Welcome to [E-XFL.COM](http://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	Active
Core Processor	S08
Core Size	8-Bit
Speed	40MHz
Connectivity	I <sup>2</sup> C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-QFP
Supplier Device Package	44-QFP (10x10)
Purchase URL	<a href="https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc9s08gt16amfbe">https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc9s08gt16amfbe</a>

## Chapter 10 Timer/PWM (S08TPMV2)

10.1	Introduction .....	153
10.1.1	Features .....	153
10.1.2	Features .....	155
10.1.3	Block Diagram .....	155
10.2	External Signal Description .....	157
10.2.1	External TPM Clock Sources .....	157
10.2.2	TPM <sub>x</sub> CH <sub>n</sub> — TPM <sub>x</sub> Channel n I/O Pins .....	157
10.3	Register Definition .....	157
10.3.1	Timer x Status and Control Register (TPM <sub>x</sub> SC) .....	158
10.3.2	Timer x Counter Registers (TPM <sub>x</sub> CNTH:TPM <sub>x</sub> CNTL) .....	159
10.3.3	Timer x Counter Modulo Registers (TPM <sub>x</sub> MODH:TPM <sub>x</sub> MODL) .....	160
10.3.4	Timer x Channel n Status and Control Register (TPM <sub>x</sub> CnSC) .....	161
10.3.5	Timer x Channel Value Registers (TPM <sub>x</sub> CnVH:TPM <sub>x</sub> CnVL) .....	162
10.4	Functional Description .....	163
10.4.1	Counter .....	163
10.4.2	Channel Mode Selection .....	164
10.4.2.1	Input Capture Mode .....	164
10.4.2.2	Output Compare Mode .....	165
10.4.2.3	Edge-Aligned PWM Mode .....	165
10.4.3	Center-Aligned PWM Mode .....	166
10.5	TPM Interrupts .....	167
10.5.1	Clearing Timer Interrupt Flags .....	167
10.5.2	Timer Overflow Interrupt Description .....	167
10.5.3	Channel Event Interrupt Description .....	168
10.5.4	PWM End-of-Duty-Cycle Events .....	168

## Chapter 11 Serial Communications Interface (S08SCIV1)

11.1	Introduction .....	169
11.1.1	Features .....	171
11.1.2	Modes of Operation .....	171
11.1.3	Block Diagram .....	172
11.2	Register Definition .....	174
11.2.1	SCI Baud Rate Registers (SCI <sub>x</sub> BDH, SCI <sub>x</sub> BHL) .....	174
11.2.2	SCI Control Register 1 (SCI <sub>x</sub> C1) .....	175
11.2.3	SCI Control Register 2 (SCI <sub>x</sub> C2) .....	176
11.2.4	SCI Status Register 1 (SCI <sub>x</sub> S1) .....	177
11.2.5	SCI Status Register 2 (SCI <sub>x</sub> S2) .....	179
11.2.6	SCI Control Register 3 (SCI <sub>x</sub> C3) .....	179
11.2.7	SCI Data Register (SCI <sub>x</sub> D) .....	180

only way to disengage security is by mass erasing the FLASH if needed (normally through the background debug interface) and verifying that FLASH is blank. To avoid returning to secure mode after the next reset, program the security bits (SEC01:SEC00) to the unsecured state (1:0).

## 4.3 RAM

The MC9S08GT16A/GT8A includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait, stop2, or stop3 mode. At power-on or after wakeup from stop1, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

For compatibility with older M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08GT16A/GT8A, it is usually best to re-initialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale-provided equate file).

```
LDHX    #RamLast+1    ;point one past RAM
TXS                    ;SP<-(H:X-1)
```

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See [Section 4.5, “Security,”](#) for a detailed description of the security feature.

## 4.4 FLASH

The FLASH memory is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the FLASH memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for FLASH erase and programming operations, in-application programming is also possible through other software-controlled communication paths. For a more detailed discussion of in-circuit and in-application programming, refer to the *HCS08 Family Reference Manual, Volume I*, Freescale Semiconductor document order number HCS08RMV1/D.

### 4.4.1 Features

Features of the FLASH memory include:

- FLASH Size
  - MC9S08GT16A — 16384 bytes (32 pages of 512 bytes each)
  - MC9S08GT8A — 8192 bytes (16 pages of 512 bytes each)
- Single power supply program and erase down to 1.8 V
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature

program time provided that the conditions above are met. In the case the next sequential address is the beginning of a new row, the program time for that byte will be the standard time instead of the burst time. This is because the high voltage to the array must be disabled and then enabled again. If a new burst command has not been queued before the current command completes, then the charge pump will be disabled and high voltage removed from the array.

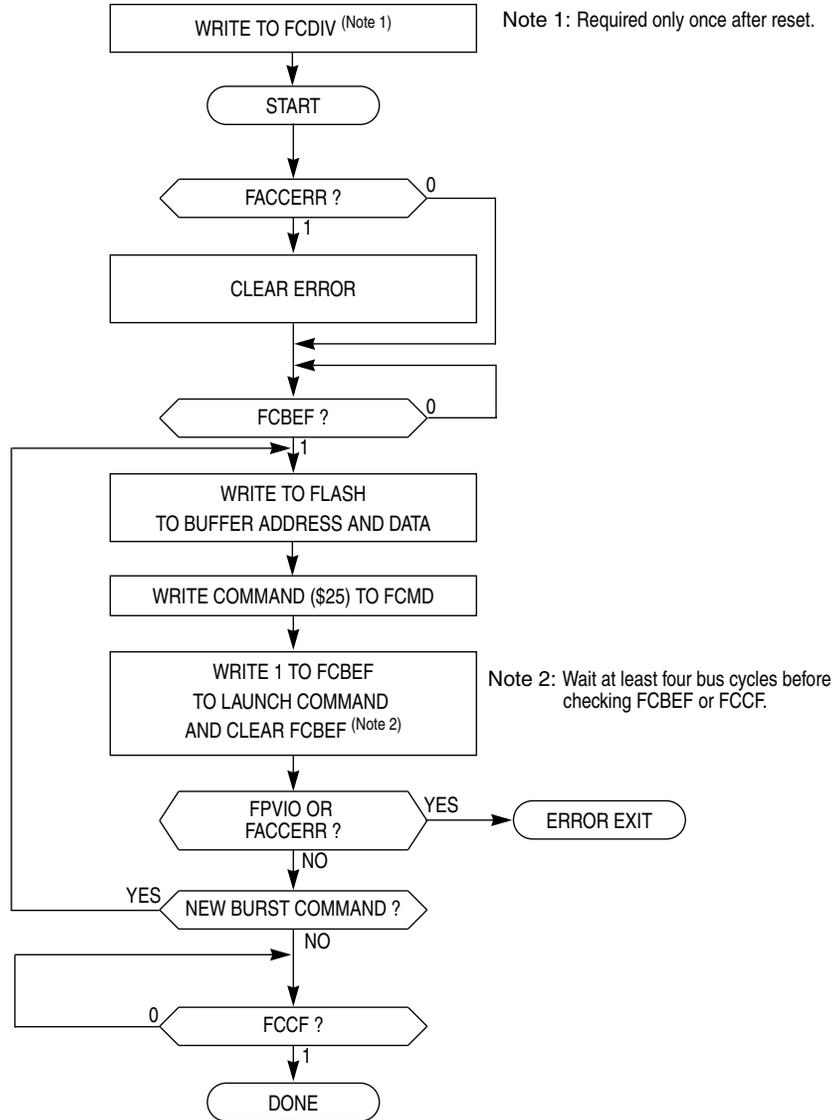


Figure 4-3. FLASH Burst Program Flowchart

## 5.5 Low-Voltage Detect (LVD) System

The MC9S08GT16A/GT8A includes a system to protect against low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system comprises a power-on reset (POR) circuit and an LVD circuit with a user selectable trip voltage, either high ( $V_{LVDH}$ ) or low ( $V_{LVDL}$ ). The LVD circuit is enabled when LVDE in SPMSC1 is high and the trip voltage is selected by LVDV in SPMSC2. The LVD is disabled upon entering any of the stop modes unless the LVDSE bit is set. If LVDSE and LVDE are both set, then the MCU cannot enter stop1 or stop2, and the current consumption in stop3 with the LVD enabled will be greater.

### 5.5.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the  $V_{POR}$  level, the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the chip in reset until the supply has risen above the  $V_{LVDL}$  level. Both the POR bit and the LVD bit in SRS are set following a POR.

### 5.5.2 LVD Reset Operation

The LVD can be configured to generate a reset upon detection of a low voltage condition by setting LVDRE to 1. After an LVD reset has occurred, the LVD system will hold the MCU in reset until the supply voltage has risen above the level determined by LVDV. The LVD bit in the SRS register is set following either an LVD reset or POR.

### 5.5.3 LVD Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF will be set and an LVD interrupt will occur.

### 5.5.4 Low-Voltage Warning (LVW)

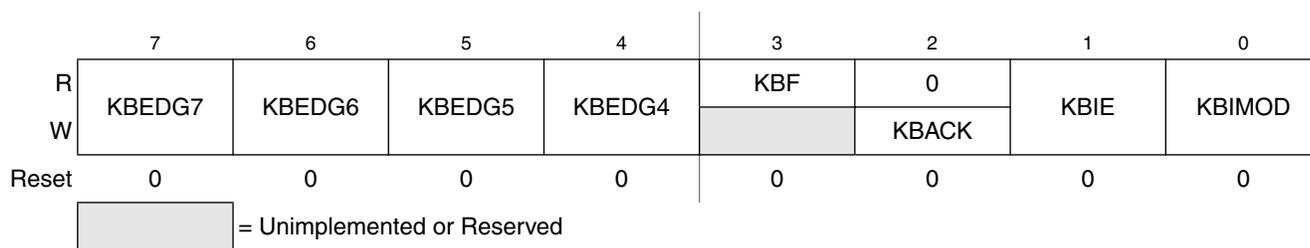
The LVD system has a low voltage warning flag to indicate to the user that the supply voltage is approaching, but is still above, the LVD voltage. The LVW does not have an interrupt associated with it. There are two user selectable trip voltages for the LVW, one high ( $V_{LVWH}$ ) and one low ( $V_{LVWL}$ ). The trip voltage is selected by LVWV in SPMSC2.

## 5.6 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts based on a multiple of the source clock's period. The RTI has two source clock choices, the external clock input (ICGERCLK) to the ICG or the RTI's own internal clock. The RTI can be used in run, wait, stop2 and stop3 modes. It is not available in stop1 mode.

In run and wait modes, only the external clock can be used as the RTI's clock source. In stop2 mode, only the internal RTI clock can be used. In stop3, either the external clock or internal RTI clock can be used.

## 7.2.1 KBI Status and Control Register (KBISC)



**Figure 7-4. KBI Status and Control Register (KBISC)**

**Table 7-1. KBISC Register Field Descriptions**

Field	Description
7:4 KBEDG[7:4]	<p><b>Keyboard Edge Select for KBI Port Bits</b> — Each of these read/write bits selects the polarity of the edges and/or levels that are recognized as trigger events on the corresponding KBI port pin when it is configured as a keyboard interrupt input (KBIPEn = 1). Also see the KBIMOD control bit, which determines whether the pin is sensitive to edges-only or edges and levels.</p> <p>0 Falling edges/low levels 1 Rising edges/high levels</p>
3 KBF	<p><b>Keyboard Interrupt Flag</b> — This read-only status flag is set whenever the selected edge event has been detected on any of the enabled KBI port pins. This flag is cleared by writing a 1 to the KBACK control bit. The flag will remain set if KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level.</p> <p>KBF can be used as a software pollable flag (KBIE = 0) or it can generate a hardware interrupt request to the CPU (KBIE = 1).</p> <p>0 No KBI interrupt pending 1 KBI interrupt pending</p>
2 KBACK	<p><b>Keyboard Interrupt Acknowledge</b> — This write-only bit (reads always return 0) is used to clear the KBF status flag by writing a 1 to KBACK. When KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level, KBF is being continuously set so writing 1 to KBACK does not clear the KBF flag.</p>
1 KBIE	<p><b>Keyboard Interrupt Enable</b> — This read/write control bit determines whether hardware interrupts are generated when the KBF status flag equals 1. When KBIE = 0, no hardware interrupts are generated, but KBF can still be used for software polling.</p> <p>0 KBF does not generate hardware interrupts (use polling) 1 KBI hardware interrupt requested when KBF = 1</p>
KBIMOD	<p><b>Keyboard Detection Mode</b> — This read/write control bit selects either edge-only detection or edge-and-level detection. KBI port bits 3 through 0 can detect falling edges-only or falling edges and low levels. KBI port bits 7 through 4 can be configured to detect either:</p> <ul style="list-style-type: none"> <li>• Rising edges-only or rising edges and high levels (KBEDGn = 1)</li> <li>• Falling edges-only or falling edges and low levels (KBEDGn = 0)</li> </ul> <p>0 Edge-only detection 1 Edge-and-level detection</p>

# Chapter 8

## Central Processor Unit (S08CPUV2)

### 8.1 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).

#### 8.1.1 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

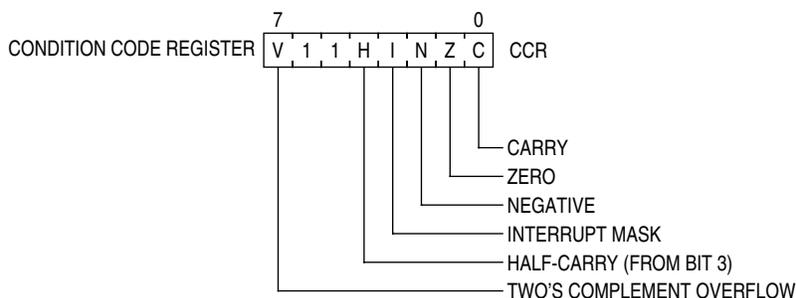


Figure 8-2. Condition Code Register

Table 8-1. CCR Register Field Descriptions

Field	Description
7 V	<b>Two's Complement Overflow Flag</b> — The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag. 0 No overflow 1 Overflow
4 H	<b>Half-Carry Flag</b> — The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C condition code bits to automatically add a correction value to the result from a previous ADD or ADC on BCD operands to correct the result to a valid BCD value. 0 No carry between bits 3 and 4 1 Carry between bits 3 and 4
3 I	<b>Interrupt Mask Bit</b> — When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the first instruction of the interrupt service routine is executed. Interrupts are not recognized at the instruction boundary after any instruction that clears I (CLI or TAP). This ensures that the next instruction after a CLI or TAP will always be executed without the possibility of an intervening interrupt, provided I was set. 0 Interrupts enabled 1 Interrupts disabled
2 N	<b>Negative Flag</b> — The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result. Simply loading or storing an 8-bit or 16-bit value causes N to be set if the most significant bit of the loaded or stored value was 1. 0 Non-negative result 1 Negative result
1 Z	<b>Zero Flag</b> — The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of 0x00 or 0x0000. Simply loading or storing an 8-bit or 16-bit value causes Z to be set if the loaded or stored value was all 0s. 0 Non-zero result 1 Zero result
0 C	<b>Carry/Borrow Flag</b> — The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag. 0 No carry out of bit 7 1 Carry out of bit 7

### 8.3.5 Extended Addressing Mode (EXT)

In extended addressing mode, the full 16-bit address of the operand is located in the next two bytes of program memory after the opcode (high byte first).

### 8.3.6 Indexed Addressing Mode

Indexed addressing mode has seven variations including five that use the 16-bit H:X index register pair and two that use the stack pointer as the base reference.

#### 8.3.6.1 Indexed, No Offset (IX)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction.

#### 8.3.6.2 Indexed, No Offset with Post Increment (IX+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction. The index register pair is then incremented ( $H:X = H:X + 0x0001$ ) after the operand has been fetched. This addressing mode is only used for MOV and CBEQ instructions.

#### 8.3.6.3 Indexed, 8-Bit Offset (IX1)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

#### 8.3.6.4 Indexed, 8-Bit Offset with Post Increment (IX1+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction. The index register pair is then incremented ( $H:X = H:X + 0x0001$ ) after the operand has been fetched. This addressing mode is used only for the CBEQ instruction.

#### 8.3.6.5 Indexed, 16-Bit Offset (IX2)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

#### 8.3.6.6 SP-Relative, 8-Bit Offset (SP1)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

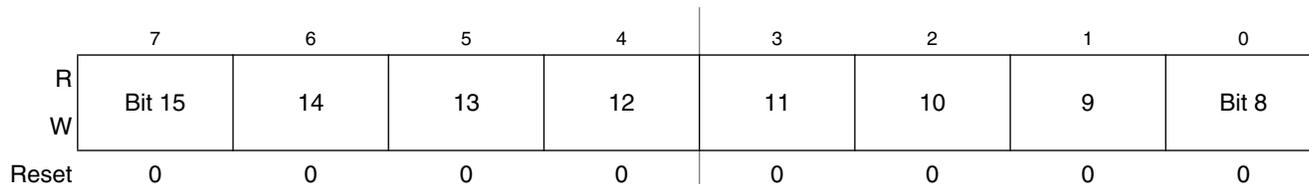
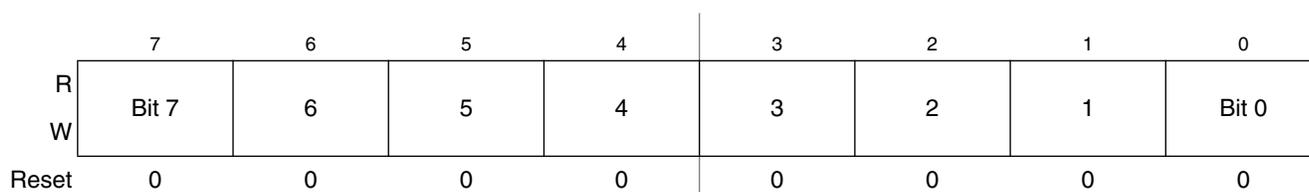
**Table 10-5. Mode, Edge, and Level Selection**

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00		Pin not used for TPM channel; use as an external clock for the TPM or revert to general-purpose I/O
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	00	Output compare	Software compare only
		01		Toggle output on compare
		10		Clear output on compare
1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)	
	X1		Low-true pulses (set output on compare)	
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger. Typically, a program would clear status flags after changing channel configuration bits and before enabling channel interrupts or using the status flags to avoid any unexpected behavior.

### 10.3.5 Timer x Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel value registers are cleared by reset.


**Figure 10-9. Timer x Channel Value Register High (TPMxCnVH)**

**Figure 10-10. Timer Channel Value Register Low (TPMxCnVL)**

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written.

### 10.4.3 Center-Aligned PWM Mode

This type of PWM output uses the up-/down-counting mode of the timer counter ( $CPWMS = 1$ ). The output compare value in  $TPMxCnVH:TPMxCnVL$  determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in  $TPMxMODH:TPMxMODL$ .

$TPMxMODH:TPMxMODL$  should be kept in the range of  $0x0001$  to  $0x7FFF$  because values outside this range can produce ambiguous results.  $ELSnA$  will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPMxCnVH:TPMxCnVL}) \quad \text{Eqn. 10-1}$$

$$\begin{aligned} \text{period} &= 2 \times (\text{TPMxMODH:TPMxMODL}); \\ \text{for } \text{TPMxMODH:TPMxMODL} &= 0x0001\text{--}0x7FFF \end{aligned} \quad \text{Eqn. 10-2}$$

If the channel value register  $TPMxCnVH:TPMxCnVL$  is zero or negative (bit 15 set), the duty cycle will be 0%. If  $TPMxCnVH:TPMxCnVL$  is a positive value (bit 15 clear) and is greater than the (nonzero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is  $0x0001$  through  $0x7FFE$  ( $0x7FFF$  if generation of 100% duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

$TPMxMODH:TPMxMODL = 0x0000$  is a special case that should not be used with center-aligned PWM mode. When  $CPWMS = 0$ , this case corresponds to the counter running free from  $0x0000$  through  $0xFFFF$ , but when  $CPWMS = 1$  the counter needs a valid match to the modulus register somewhere other than at  $0x0000$  in order to change directions from up-counting to down-counting.

Figure 10-12 shows the output compare value in the TPM channel registers (multiplied by 2), which determines the pulse width (duty cycle) of the CPWM signal. If  $ELSnA = 0$ , the compare match while counting up forces the CPWM output signal low and a compare match while counting down forces the output high. The counter counts up until it reaches the modulo setting in  $TPMxMODH:TPMxMODL$ , then counts down until it reaches zero. This sets the period equal to two times  $TPMxMODH:TPMxMODL$ .

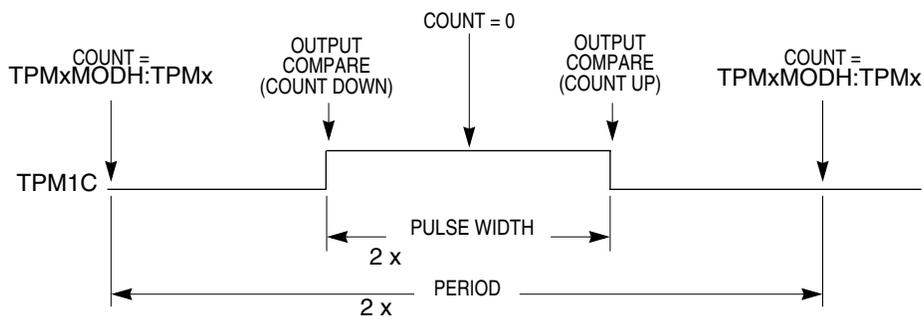


Figure 10-12. CPWM Period and Pulse Width ( $ELSnA = 0$ )

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Because the HCS08 is a family of 8-bit MCUs, 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,  $TPMxMODH$ ,  $TPMxMODL$ ,  $TPMxCnVH$ , and  $TPMxCnVL$ , actually write to buffer registers. Values are

**Table 11-5. SCiXS1 Register Field Descriptions (continued)**

Field	Description
5 RDRF	<p><b>Receive Data Register Full Flag</b> — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCiXD). To clear RDRF, read SCiXS1 with RDRF = 1 and then read the SCI data register (SCiXD).</p> <p>0 Receive data register empty. 1 Receive data register full.</p>
4 IDLE	<p><b>Idle Line Flag</b> — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.</p> <p>To clear IDLE, read SCiXS1 with IDLE = 1 and then read the SCI data register (SCiXD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period.</p> <p>0 No idle line detected. 1 Idle line was detected.</p>
3 OR	<p><b>Receiver Overrun Flag</b> — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCiXD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCiXD. To clear OR, read SCiXS1 with OR = 1 and then read the SCI data register (SCiXD).</p> <p>0 No overrun. 1 Receive overrun (new SCI data lost).</p>
2 NF	<p><b>Noise Flag</b> — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCiXS1 and then read the SCI data register (SCiXD).</p> <p>0 No noise detected. 1 Noise detected in the received character in SCiXD.</p>
1 FE	<p><b>Framing Error Flag</b> — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCiXS1 with FE = 1 and then read the SCI data register (SCiXD).</p> <p>0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.</p>
0 PF	<p><b>Parity Error Flag</b> — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCiXS1 and then read the SCI data register (SCiXD).</p> <p>0 No parity error. 1 Parity error.</p>

When the SPI is configured as a master, the clock output is routed to the SPSCCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

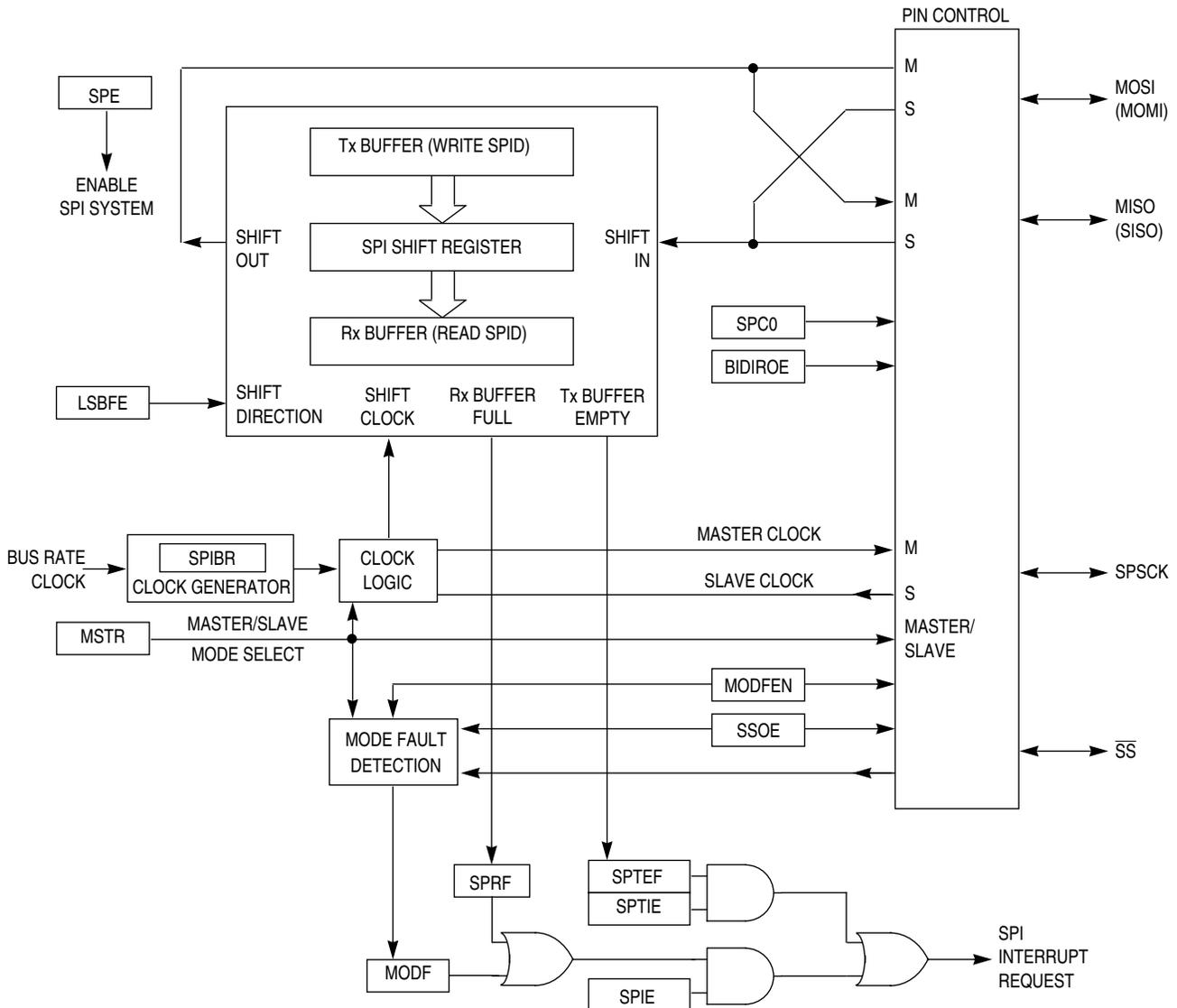


Figure 12-4. SPI Module Block Diagram

### 12.1.3 SPI Baud Rate Generation

As shown in Figure 12-5, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to get the internal SPI master mode bit-rate clock.

**Table 12-1. SPIC1 Field Descriptions (continued)**

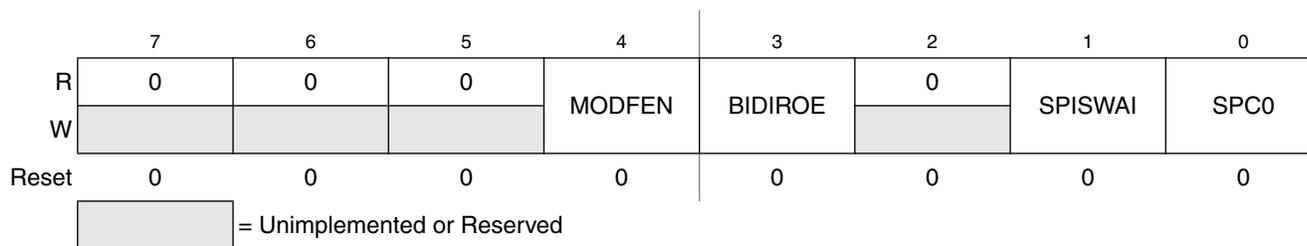
Field	Description
4 MSTR	<b>Master/Slave Mode Select</b> 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	<b>Clock Polarity</b> — This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to <a href="#">Section 12.5.1, “SPI Clock Formats”</a> for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	<b>Clock Phase</b> — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to <a href="#">Section 12.5.1, “SPI Clock Formats”</a> for more details. 0 First edge on SPSCCK occurs at the middle of the first cycle of an 8-cycle data transfer 1 First edge on SPSCCK occurs at the start of the first cycle of an 8-cycle data transfer
1 SSOE	<b>Slave Select Output Enable</b> — This bit is used in combination with the mode fault enable (MODFEN) bit in SPIC2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin as shown in <a href="#">Table 12-2</a> .
0 LSBFE	<b>LSB First (Shifter Direction)</b> 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

**Table 12-2.  $\overline{SS}$  Pin Function**

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

## 12.4.2 SPI Control Register 2 (SPIC2)

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.


**Figure 12-7. SPI Control Register 2 (SPIC2)**

$$E_S = 2^N * (\Delta V_{SAMP} / (V_{REFH} - V_{REFL})) * (C_{AIN} / (C_{AIN} + C_{AS})) \quad \text{Eqn. 14-3}$$

In the case of a 0.1  $\mu\text{F}$   $C_{AS}$ , a worst case sampling error of 0.5 LSB is achieved regardless of  $R_{AS}$ . However, in the case of repeated conversions at a rate of  $f_{SAMP}$ ,  $R_{AS}$  must re-charge  $C_{AS}$ . This recharge is continuous and controlled only by  $R_{AS}$  (not  $R_{AIN}$ ), and reduces the overall sampling error to:

$$E_S = 2^N * \{(\Delta V_{AIN} / (V_{REFH} - V_{REFL})) * e^{-(1 / (f_{SAMP} * R_{AS} * C_{AS}))} + (\Delta V_{SAMP} / (V_{REFH} - V_{REFL})) * \text{Min}[(C_{AIN} / (C_{AIN} + C_{AS})), e^{-(1 / (f_{ATDCLK} * (R_{AS} + R_{AIN}) * C_{AIN}))}]\} \quad \text{Eqn. 14-4}$$

This is a worst case sampling error which does not account for  $R_{AS}$  recharging the combination of  $C_{AS}$  and  $C_{AIN}$  during the sample window. It does illustrate that high values of  $R_{AS}$  ( $>10 \text{ k}\Omega$ ) are possible if a large  $C_{AS}$  is used and sufficient time to recharge  $C_{AS}$  is provided between samples. In order to achieve accuracy specified under the worst case conditions of maximum  $\Delta V_{SAMP}$  and minimum  $C_{AS}$ ,  $R_{AS}$  must be less than the maximum value of 10  $\text{k}\Omega$ . The maximum value of 10  $\text{k}\Omega$  for  $R_{AS}$  is to ensure low sampling error in the worst case condition of maximum  $\Delta V_{SAMP}$  and minimum  $C_{AS}$ .

### 14.4.3 Analog Input Multiplexer

The analog input multiplexer selects one of the eight external analog input channels to generate an analog sample. The analog input multiplexer includes negative stress protection circuitry which prevents cross-talk between channels when the applied input potentials are within specification. Only analog input signals within the potential range of  $V_{REFL}$  to  $V_{REFH}$  (ATD reference potentials) will result in valid ATD conversions.

### 14.4.4 ATD Module Accuracy Definitions

Figure 14-11 illustrates an ideal ATD transfer function. The horizontal axis represents the ATD input voltage in millivolts. The vertical axis the conversion result code. The ATD is specified with the following figures of merit:

- Number of bits (N) — The number of bits in the digitized output
- Resolution (LSB) — The resolution of the ATD is the step size of the ideal transfer function. This is also referred to as the ideal code width, or the difference between the transition voltages to a given code and to the next code. This unit, known as 1LSB, is equal to

$$1\text{LSB} = (V_{REFH} - V_{REFL}) / 2^N \quad \text{Eqn. 14-5}$$

- Inherent quantization error ( $E_Q$ ) — This is the error caused by the division of the perfect ideal straight-line transfer function into the quantized ideal transfer function with  $2^N$  steps. This error is  $\pm 1/2$  LSB.
- Differential non-linearity (DNL) — This is the difference between the current code width and the ideal code width (1LSB). The current code width is the difference in the transition voltages to the current code and to the next code. A negative DNL means the transfer function spends less time at the current code than ideal; a positive DNL, more. The DNL cannot be less than  $-1.0$ ; a DNL of greater than 1.0 reduces the effective number of bits by 1.
- Integral non-linearity (INL) — This is the difference between the transition voltage to the current code and the transition to the corresponding code on the adjusted transfer curve. INL is a measure

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 DBGTC 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.

### 15.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGTC register selects one of nine trigger modes. When TRGSEL = 1 in the DBGTC register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGTC 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 DBGTC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGSR. 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 DBGGEN in DBGTC.

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.

**A-Only** — Trigger when the address matches the value in comparator A

**A OR B** — Trigger when the address matches either the value in comparator A or the value in comparator B

**A Then B** — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.

**A AND B Data (Full Mode)** — This is called a full mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

**A AND NOT B Data (Full Mode)** — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

**Event-Only B (Store Data)** — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

**A Then Event-Only B (Store Data)** — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

**Inside Range ( $A \leq \text{Address} \leq B$ )** — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.

**Outside Range ( $\text{Address} < A$  or  $\text{Address} > B$ )** — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

## 15.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGCR register may be set to 1 to allow any of the trigger conditions described in [Section 15.3.5, “Trigger Modes,”](#) to be used to generate a hardware breakpoint request to the CPU. TAG in DBGCR controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to active background mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to active background mode.

If the background mode has not been enabled (ENBDM = 1) by a serial WRITE\_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to active background mode.

## 15.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

Refer to the high-page register summary in the device overview chapter of this data sheet for the absolute address assignments for all DBG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 15.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE\_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

**Table A-2. Absolute Maximum Ratings**

Rating	Symbol	Value	Unit
Supply voltage	$V_{DD}$	-0.3 to +3.8	V
Maximum current into $V_{DD}$	$I_{DD}$	120	mA
Digital input voltage	$V_{In}$	-0.3 to $V_{DD} + 0.3$	V
Instantaneous maximum current Single pin limit (applies to all port pins) <sup>1, 2, 3</sup>	$I_D$	$\pm 25$	mA
Storage temperature range	$T_{stg}$	-55 to 150	°C
Maximum junction temperature	$T_J$	150	°C

<sup>1</sup> Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive ( $V_{DD}$ ) and negative ( $V_{SS}$ ) clamp voltages, then use the larger of the two resistance values.

<sup>2</sup> All functional non-supply pins are internally clamped to  $V_{SS}$  and  $V_{DD}$ .

<sup>3</sup> Power supply must maintain regulation within operating  $V_{DD}$  range during instantaneous and operating maximum current conditions. If positive injection current ( $V_{In} > V_{DD}$ ) is greater than  $I_{DD}$ , the injection current may flow out of  $V_{DD}$  and could result in external power supply going out of regulation. Ensure external  $V_{DD}$  load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if the clock rate is very low which would reduce overall power consumption.

## A.4 Thermal Characteristics

This section provides information about operating temperature range, power dissipation, and package thermal resistance. Power dissipation on I/O pins is usually small compared to the power dissipation in on-chip logic and voltage regulator circuits and it is user-determined rather than being controlled by the MCU design. In order to take  $P_{I/O}$  into account in power calculations, determine the difference between actual pin voltage and  $V_{SS}$  or  $V_{DD}$  and multiply by the pin current for each I/O pin. Except in cases of unusually high pin current (heavy loads), the difference between pin voltage and  $V_{SS}$  or  $V_{DD}$  will be very small.

## A.11 FLASH Specifications

This section provides details about program/erase times and program-erase endurance for the FLASH memory.

Program and erase operations do not require any special power sources other than the normal  $V_{DD}$  supply. For more detailed information about program/erase operations, see [Chapter 4, “Memory.”](#)

**Table A-15. FLASH Characteristics**

Characteristic	Symbol	Min	Typical	Max	Unit
Supply voltage for program/erase $T \leq 85^{\circ}\text{C}$ $T > 85^{\circ}\text{C}$	$V_{\text{prog/erase}}$	1.8 2.1		3.6 3.6	V V
Supply voltage for read operation $0 < f_{\text{Bus}} < 8 \text{ MHz}$ $0 < f_{\text{Bus}} < 20 \text{ MHz}$	$V_{\text{Read}}$	1.8 2.08		3.6 3.6	V
Internal FCLK frequency <sup>1</sup>	$f_{\text{FCLK}}$	150		200	kHz
Internal FCLK period (1/FCLK)	$t_{\text{Fcy}}c$	5		6.67	$\mu\text{s}$
Byte program time (random location) <sup>(2)</sup>	$t_{\text{prog}}$		9		$t_{\text{Fcy}}c$
Byte program time (burst mode) <sup>(2)</sup>	$t_{\text{Burst}}$		4		$t_{\text{Fcy}}c$
Page erase time <sup>2</sup>	$t_{\text{Page}}$		4000		$t_{\text{Fcy}}c$
Mass erase time <sup>(2)</sup>	$t_{\text{Mass}}$		20,000		$t_{\text{Fcy}}c$
Byte program current <sup>3</sup>	$R_{\text{IDDBP}}$	—	4	—	mA
Page erase current <sup>3</sup>	$R_{\text{IDPE}}$	—	6	—	mA
Program/erase endurance <sup>4</sup> $T_{\text{L}}$ to $T_{\text{H}} = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ $T = 25^{\circ}\text{C}$		10,000	100,000	— —	cycles
Data retention <sup>5</sup>	$t_{\text{D\_ret}}$	15	100	—	years

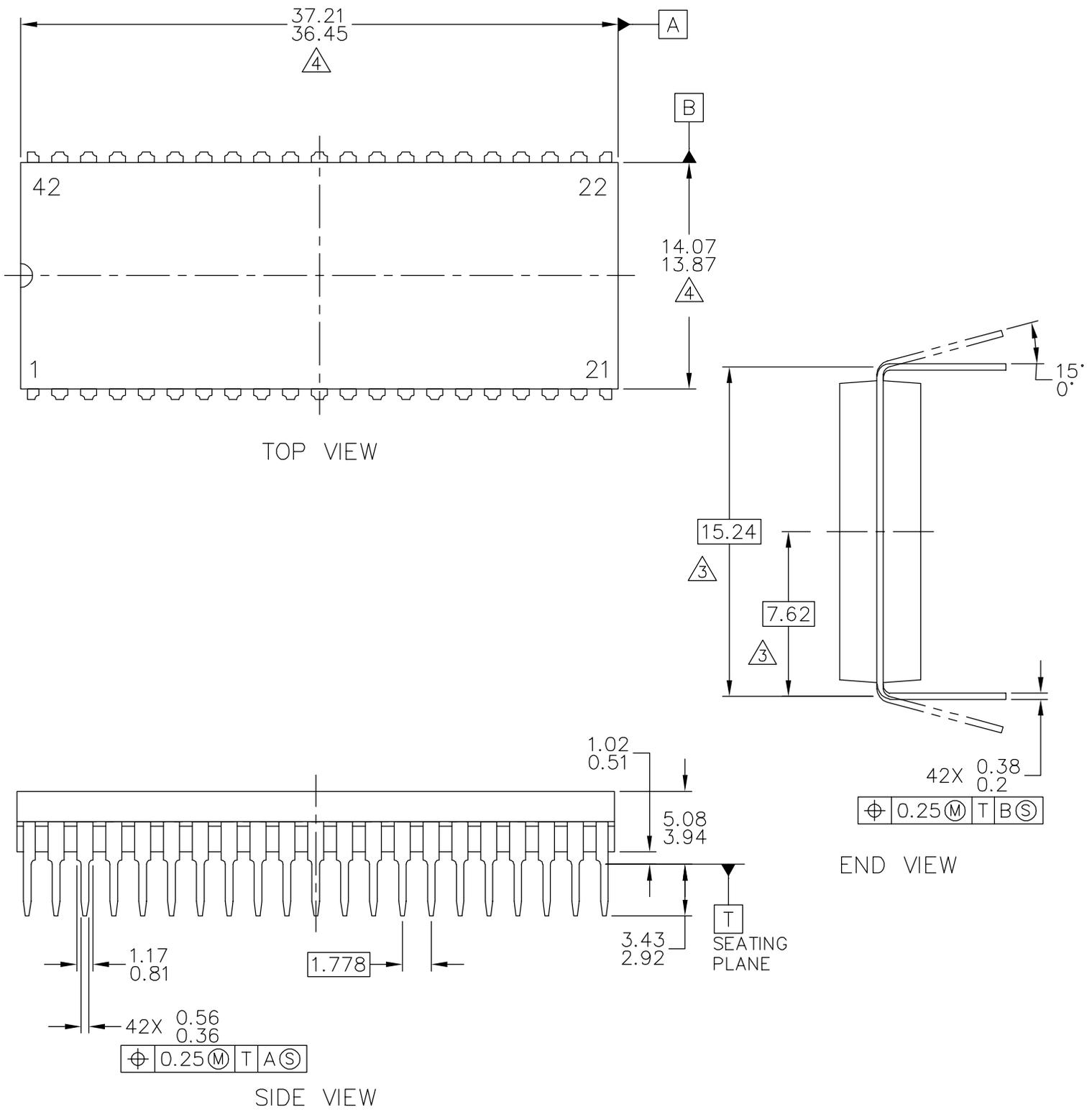
<sup>1</sup> The frequency of this clock is controlled by a software setting.

<sup>2</sup> These values are hardware state machine controlled. User code does not need to count cycles. This information supplied for calculating approximate time to program and erase.

<sup>3</sup> The program and erase currents are additional to the standard run  $I_{\text{DD}}$ . These values were measured at room temperatures with  $V_{\text{DD}} = 3.0 \text{ V}$ , bus frequency = 4.0 MHz.

<sup>4</sup> **Typical endurance for FLASH** was evaluated for this product family on the 9S12Dx64. For additional information on how Freescale Semiconductor defines typical endurance, please refer to Engineering Bulletin EB619/D, *Typical Endurance for Nonvolatile Memory*.

<sup>5</sup> **Typical data retention** values are based on intrinsic capability of the technology measured at high temperature and de-rated to  $25^{\circ}\text{C}$  using the Arrhenius equation. For additional information on how Freescale Semiconductor defines typical data retention, please refer to Engineering Bulletin EB618/D, *Typical Data Retention for Nonvolatile Memory*.



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	<b>MECHANICAL OUTLINE</b>	PRINT VERSION NOT TO SCALE	
TITLE:  42 LD PDIP	DOCUMENT NO: 98ASB42767B	REV: A	
	CASE NUMBER: 858-01	24 OCT 2005	
	STANDARD: NON-JEDEC		