

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	54
Program Memory Size	60KB (60K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-QFP
Supplier Device Package	64-QFP (14x14)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08aw60cfue">https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08aw60cfue</a>

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.

the MCU secure. During development, whenever the FLASH is erased, it is good practice to immediately program the SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

The on-chip debug module cannot be enabled while the MCU is secure. The separate background debug controller can still be used for background memory access commands, but the MCU cannot enter active background mode except by holding BKGD/MS low at the rising edge of reset.

A user can choose to allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. If the nonvolatile KEYEN bit in NVOPT/FOPT is 0, the backdoor key is disabled and there is no way to disengage security without completely erasing all FLASH locations. If KEYEN is 1, a secure user program can temporarily disengage security by:

1. Writing 1 to KEYACC in the FCNFG register. This makes the FLASH module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a FLASH program or erase command.
2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be done in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX should not be used for these writes because these writes cannot be done on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was just written matches the key stored in the FLASH locations, SEC01:SEC00 are automatically changed to 1:0 and security will be disengaged until the next reset.

The security key can be written only from secure memory (either RAM or FLASH), so it cannot be entered through background commands without the cooperation of a secure user program.

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in FLASH memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other FLASH memory location. The nonvolatile registers are in the same 512-byte block of FLASH as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key. Block protects cannot be changed from user application programs, so if the vector space is block protected, the backdoor security key mechanism cannot permanently change the block protect, security settings, or the backdoor key.

Security can always be disengaged through the background debug interface by taking these steps:

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase FLASH if necessary.
3. Blank check FLASH. Provided FLASH is completely erased, security is disengaged until the next reset.

To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

Either RTI clock source can be used when the MCU is in run, wait or stop3 mode. When using the external oscillator in stop3, it must be enabled in stop (OSCSTEN = 1) and configured for low bandwidth operation (RANGE = 0). Only the internal 1-kHz clock source can be selected to wake the MCU from stop2 mode.

The SRTISC register includes a read-only status flag, a write-only acknowledge bit, and a 3-bit control value (RTIS2:RTIS1:RTIS0) used to disable the clock source to the real-time interrupt or select one of seven wakeup periods. The RTI has a local interrupt enable, RTIE, to allow masking of the real-time interrupt. The RTI can be disabled by writing each bit of RTIS to zeroes, and no interrupts will be generated. See Section 5.9.7, “System Real-Time Interrupt Status and Control Register (SRTISC),” for detailed information about this register.

## 5.8 MCLK Output

The PTC2 pin is shared with the MCLK clock output. Setting the pin enable bit, MPE, causes the PTC2 pin to output a divided version of the internal MCU bus clock. The divide ratio is determined by the MCSEL bits. When MPE is set, the PTC2 pin is forced to operate as an output pin regardless of the state of the port data direction control bit for the pin. If the MCSEL bits are all 0s, the pin is driven low. The slew rate and drive strength for the pin are controlled by PTCSE2 and PTCDS2, respectively. The maximum clock output frequency is limited if slew rate control is enabled, see Appendix A, “Electrical Characteristics and Timing Specifications,” for pin rise and fall times with slew rate enabled.

## 5.9 Reset, Interrupt, and System Control Registers and Control Bits

One 8-bit register in the direct page register space and eight 8-bit registers in the high-page register space are related to reset and interrupt systems.

Refer to the direct-page register summary in Chapter 4, “Memory,” of this data sheet for the absolute address assignments for all 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.

Some control bits in the SOPT and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in Chapter 3, “Modes of Operation.”

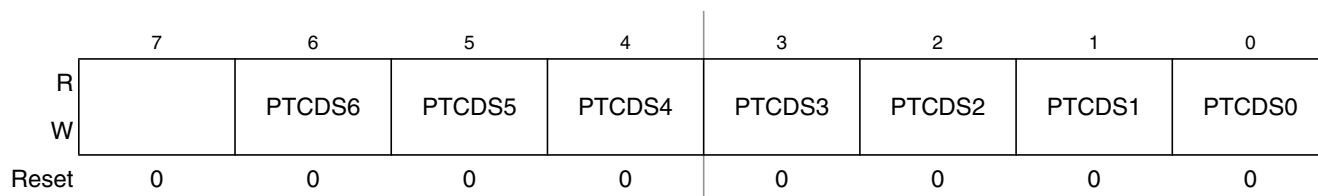


Figure 6-23. Output Drive Strength Selection for Port C (PTCDS)

Table 6-16. PTCDS Register Field Descriptions

Field	Description
6:0 PTCDS[6:0]	<p><b>Output Drive Strength Selection for Port C Bits</b> — Each of these control bits selects between low and high output drive for the associated PTC pin.</p> <p>0 Low output drive enabled for port C bit n.</p> <p>1 High output drive enabled for port C bit n.</p>

### 7.3.6.7 SP-Relative, 16-Bit Offset (SP2)

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

## 7.4 Special Operations

The CPU performs a few special operations that are similar to instructions but do not have opcodes like other CPU instructions. In addition, a few instructions such as STOP and WAIT directly affect other MCU circuitry. This section provides additional information about these operations.

### 7.4.1 Reset Sequence

Reset can be caused by a power-on-reset (POR) event, internal conditions such as the COP (computer operating properly) watchdog, or by assertion of an external active-low reset pin. When a reset event occurs, the CPU immediately stops whatever it is doing (the MCU does not wait for an instruction boundary before responding to a reset event). For a more detailed discussion about how the MCU recognizes resets and determines the source, refer to the Resets, Interrupts, and System Configuration chapter.

The reset event is considered concluded when the sequence to determine whether the reset came from an internal source is done and when the reset pin is no longer asserted. At the conclusion of a reset event, the CPU performs a 6-cycle sequence to fetch the reset vector from 0xFFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

### 7.4.2 Interrupt Sequence

When an interrupt is requested, the CPU completes the current instruction before responding to the interrupt. At this point, the program counter is pointing at the start of the next instruction, which is where the CPU should return after servicing the interrupt. The CPU responds to an interrupt by performing the same sequence of operations as for a software interrupt (SWI) instruction, except the address used for the vector fetch is determined by the highest priority interrupt that is pending when the interrupt sequence started.

The CPU sequence for an interrupt is:

1. Store the contents of PCL, PCH, X, A, and CCR on the stack, in that order.
2. Set the I bit in the CCR.
3. Fetch the high-order half of the interrupt vector.
4. Fetch the low-order half of the interrupt vector.
5. Delay for one free bus cycle.
6. Fetch three bytes of program information starting at the address indicated by the interrupt vector to fill the instruction queue in preparation for execution of the first instruction in the interrupt service routine.

After the CCR contents are pushed onto the stack, the I bit in the CCR is set to prevent other interrupts while in the interrupt service routine. Although it is possible to clear the I bit with an instruction in the

interrupt service routine, this would allow nesting of interrupts (which is not recommended because it leads to programs that are difficult to debug and maintain).

For compatibility with the earlier M68HC05 MCUs, the high-order half of the H:X index register pair (H) is not saved on the stack as part of the interrupt sequence. The user must use a PSHH instruction at the beginning of the service routine to save H and then use a PULH instruction just before the RTI that ends the interrupt service routine. It is not necessary to save H if you are certain that the interrupt service routine does not use any instructions or auto-increment addressing modes that might change the value of H.

The software interrupt (SWI) instruction is like a hardware interrupt except that it is not masked by the global I bit in the CCR and it is associated with an instruction opcode within the program so it is not asynchronous to program execution.

### 7.4.3 Wait Mode Operation

The WAIT instruction enables interrupts by clearing the I bit in the CCR. It then halts the clocks to the CPU to reduce overall power consumption while the CPU is waiting for the interrupt or reset event that will wake the CPU from wait mode. When an interrupt or reset event occurs, the CPU clocks will resume and the interrupt or reset event will be processed normally.

If a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in wait mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in wait mode.

### 7.4.4 Stop Mode Operation

Usually, all system clocks, including the crystal oscillator (when used), are halted during stop mode to minimize power consumption. In such systems, external circuitry is needed to control the time spent in stop mode and to issue a signal to wake up the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the HCS08 can be configured to keep a minimum set of clocks running in stop mode. This optionally allows an internal periodic signal to wake the target MCU from stop mode.

When a host debug system is connected to the background debug pin (BKGD) and the ENBDM control bit has been set by a serial command through the background interface (or because the MCU was reset into active background mode), the oscillator is forced to remain active when the MCU enters stop mode. In this case, if a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in stop mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in stop mode.

Recovery from stop mode depends on the particular HCS08 and whether the oscillator was stopped in stop mode. Refer to the Modes of Operation chapter for more details.

Table 7-2. HCS08 Instruction Set Summary (Sheet 3 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>		
			V	H	I	N	Z	C						
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear	Branch if (Mn) = 0						↓	DIR (b0)	01	dd rr	5		
										DIR (b1)	03	dd rr	5	
											DIR (b2)	05	dd rr	5
											DIR (b3)	07	dd rr	5
											DIR (b4)	09	dd rr	5
											DIR (b5)	0B	dd rr	5
											DIR (b6)	0D	dd rr	5
								DIR (b7)	0F	dd rr	5			
BRN <i>rel</i>	Branch Never	Uses 3 Bus Cycles	-	-	-	-	-	-	REL	21	rr	3		
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set	Branch if (Mn) = 1						↓	DIR (b0)	00	dd rr	5		
										DIR (b1)	02	dd rr	5	
											DIR (b2)	04	dd rr	5
											DIR (b3)	06	dd rr	5
											DIR (b4)	08	dd rr	5
											DIR (b5)	0A	dd rr	5
											DIR (b6)	0C	dd rr	5
								DIR (b7)	0E	dd rr	5			
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory	Mn ← 1	-	-	-	-	-	-	DIR (b0)	10	dd	5		
									DIR (b1)	12	dd	5		
									DIR (b2)	14	dd	5		
									DIR (b3)	16	dd	5		
									DIR (b4)	18	dd	5		
									DIR (b5)	1A	dd	5		
									DIR (b6)	1C	dd	5		
									DIR (b7)	1E	dd	5		
BSR <i>rel</i>	Branch to Subroutine	PC ← (PC) + 0x0002 push (PCL); SP ← (SP) - 0x0001 push (PCH); SP ← (SP) - 0x0001 PC ← (PC) + <i>rel</i>	-	-	-	-	-	-	REL	AD	rr	5		
CBEQ <i>opr8a,rel</i> CBEQA <i>#opr8i,rel</i> CBEQX <i>#opr8i,rel</i> CBEQ <i>opr8,X+,rel</i> CBEQ <i>,X+,rel</i> CBEQ <i>opr8,SP,rel</i>	Compare and Branch if Equal	Branch if (A) = (M)							DIR	31	dd rr	5		
		Branch if (A) = (M)							IMM	41	ii rr	4		
		Branch if (X) = (M)							IMM	51	ii rr	4		
		Branch if (A) = (M)							IX1+	61	ff rr	5		
		Branch if (A) = (M)							IX+	71	rr	5		
		Branch if (A) = (M)							SP1	9E61	ff rr	6		
CLC	Clear Carry Bit	C ← 0	-	-	-	-	0	INH	98		1			
CLI	Clear Interrupt Mask Bit	I ← 0	-	-	0	-	-	INH	9A		1			
CLR <i>opr8a</i> CLRA CLR X CLR X CLR <i>opr8,X</i> CLR <i>,X</i> CLR <i>opr8,SP</i>	Clear	M ← 0x00							DIR	3F	dd	5		
		A ← 0x00							INH	4F		1		
		X ← 0x00							INH	5F		1		
		H ← 0x00					0	1	INH	8C		1		
		M ← 0x00							IX1	6F	ff	5		
		M ← 0x00							IX	7F		4		
		M ← 0x00							SP1	9E6F	ff	6		
CMP <i>#opr8i</i> CMP <i>opr8a</i> CMP <i>opr16a</i> CMP <i>opr16,X</i> CMP <i>opr8,X</i> CMP <i>,X</i> CMP <i>opr16,SP</i> CMP <i>opr8,SP</i>	Compare Accumulator with Memory	(A) - (M) (CCR Updated But Operands Not Changed)	↓	-	-	↓	↓	↓	IMM	A1	ii	2		
									DIR	B1	dd	3		
									EXT	C1	hh ll	4		
									IX2	D1	ee ff	4		
									IX1	E1	ff	3		
									IX	F1		3		
									SP2	9ED1	ee ff	5		
							SP1	9EE1	ff	4				
COM <i>opr8a</i> COMA COM X COM <i>opr8,X</i> COM <i>,X</i> COM <i>opr8,SP</i>	Complement (One's Complement)	M ← (M) = 0xFF - (M)							DIR	33	dd	5		
		A ← (A) = 0xFF - (A)							INH	43		1		
		X ← (X) = 0xFF - (X)							INH	53		1		
		M ← (M) = 0xFF - (M)					↓	↓	IX1	63	ff	5		
		M ← (M) = 0xFF - (M)							IX	73		4		
		M ← (M) = 0xFF - (M)							SP1	9E63	ff	6		
CPHX <i>opr16a</i> CPHX <i>#opr16i</i> CPHX <i>opr8a</i> CPHX <i>opr8,SP</i>	Compare Index Register (H:X) with Memory	(H:X) - (M:M + 0x0001) (CCR Updated But Operands Not Changed)	↓	-	-	↓	↓	↓	EXT	3E	hh ll	6		
									IMM	65	jj kk	3		
									DIR	75	dd	5		
									SP1	9EF3	ff	6		



Table 7-2. HCS08 Instruction Set Summary (Sheet 7 of 7)

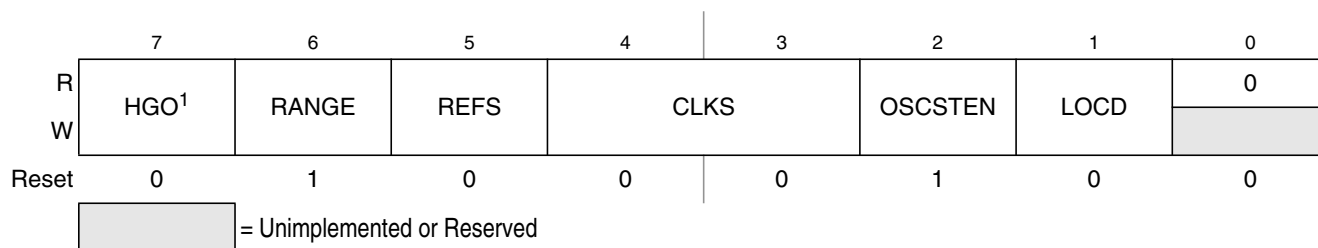
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles <sup>1</sup>
			V	H	I	N	Z	C				
TAP	Transfer Accumulator to CCR	$CCR \leftarrow (A)$	↓	↓	↓	↓	↓	↓	INH	84		1
TAX	Transfer Accumulator to X (Index Register Low)	$X \leftarrow (A)$	-	-	-	-	-	-	INH	97		1
TPA	Transfer CCR to Accumulator	$A \leftarrow (CCR)$	-	-	-	-	-	-	INH	85		1
TST <i>opr8a</i> TSTA TSTX TST <i>opr8,X</i> TST <i>,X</i> TST <i>opr8,SP</i>	Test for Negative or Zero	(M) - 0x00 (A) - 0x00 (X) - 0x00 (M) - 0x00 (M) - 0x00 (M) - 0x00	0	-	-	↓	↓	-	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	4 1 1 4 3 5
TSX	Transfer SP to Index Reg.	$H:X \leftarrow (SP) + 0x0001$	-	-	-	-	-	-	INH	95		2
TXA	Transfer X (Index Reg. Low) to Accumulator	$A \leftarrow (X)$	-	-	-	-	-	-	INH	9F		1
TXS	Transfer Index Reg. to SP	$SP \leftarrow (H:X) - 0x0001$	-	-	-	-	-	-	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	I bit $\leftarrow$ 0; Halt CPU	-	-	0	-	-	-	INH	8F		2+

<sup>1</sup> Bus clock frequency is one-half of the CPU clock frequency.

## 8.3 Register Definition

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all ICG 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.

### 8.3.1 ICG Control Register 1 (ICGC1)



**Figure 8-6. ICG Control Register 1 (ICGC1)**

<sup>1</sup> This bit can be written only once after reset. Additional writes are ignored.

**Table 8-1. ICGC1 Register Field Descriptions**

Field	Description
7 HGO	<b>High Gain Oscillator Select</b> — The HGO bit is used to select between low power operation and high gain operation for improved noise immunity. This bit is write-once after reset. 0 Oscillator configured for low power operation. 1 Oscillator configured for high gain operation.
6 RANGE	<b>Frequency Range Select</b> — The RANGE bit controls the oscillator, reference divider, and FLL loop prescaler multiplication factor (P). It selects one of two reference frequency ranges for the ICG. The RANGE bit is write-once after a reset. The RANGE bit only has an effect in FLL engaged external and FLL bypassed external modes. 0 Oscillator configured for low frequency range. FLL loop prescale factor P is 64. 1 Oscillator configured for high frequency range. FLL loop prescale factor P is 1.
5 REFS	<b>External Reference Select</b> — The REFS bit controls the external reference clock source for ICGERCLK. The REFS bit is write-once after a reset. 0 External clock requested. 1 Oscillator using crystal or resonator requested.
4:3 CLKS	<b>Clock Mode Select</b> — The CLKS bits control the clock mode as described below. If FLL bypassed external is requested, it will not be selected until ERCS = 1. If the ICG enters off mode, the CLKS bits will remain unchanged. Writes to the CLKS bits will not take effect if a previous write is not complete. 00 Self-clocked 01 FLL engaged, internal reference 10 FLL bypassed, external reference 11 FLL engaged, external reference The CLKS bits are writable at any time, unless the first write after a reset was CLKS = 0X, the CLKS bits cannot be written to 1X until after the next reset (because the EXTAL pin was not reserved).

### 9.3.1 KBI Block Diagram

Figure 9-2 shows the block diagram for a KBI module.

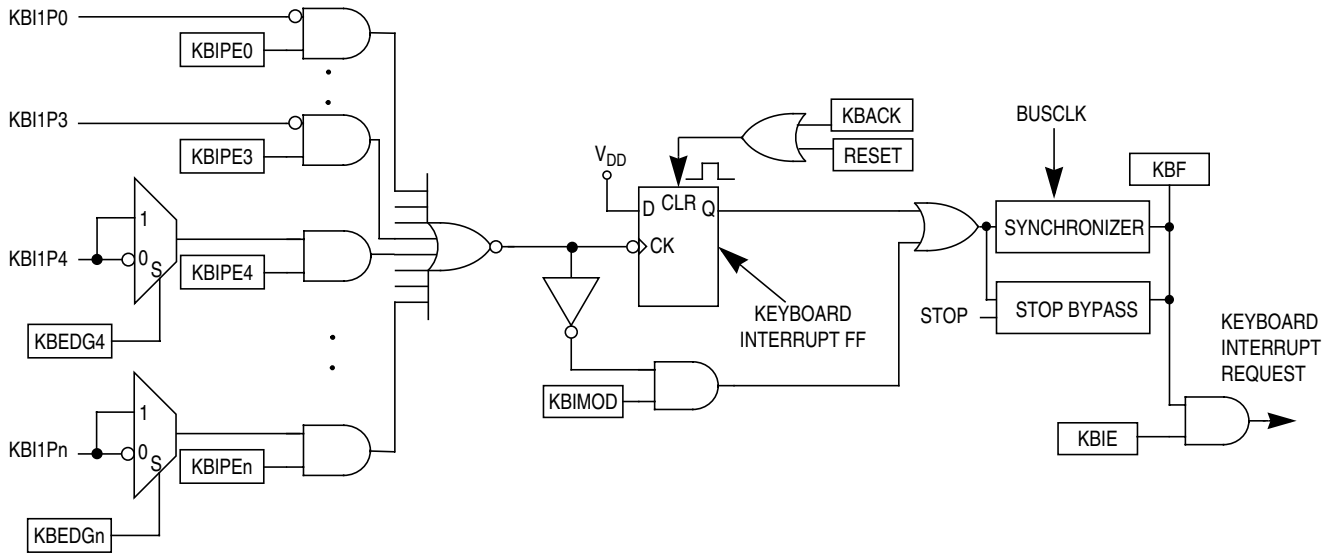


Figure 9-2. KBI Block Diagram

### 9.4 Register Definition

This section provides information about all registers and control bits associated with the KBI module.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all KBI 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.

All TPM channels are programmable independently as input capture, output compare, or buffered edge-aligned PWM channels.

## 10.3 External Signal Description

When any pin associated with the timer is configured as a timer input, a passive pullup can be enabled. After reset, the TPM modules are disabled and all pins default to general-purpose inputs with the passive pullups disabled.

### 10.3.1 External TPM Clock Sources

When control bits CLKSB:CLKSA in the timer status and control register are set to 1:1, the prescaler and consequently the 16-bit counter for TPMx are driven by an external clock source, TPMxCLK, connected to an I/O pin. A synchronizer is needed between the external clock and the rest of the TPM. This synchronizer is clocked by the bus clock so the frequency of the external source must be less than one-half the frequency of the bus rate clock. The upper frequency limit for this external clock source is specified to be one-fourth the bus frequency to conservatively accommodate duty cycle and phase-locked loop (PLL) or frequency-locked loop (FLL) frequency jitter effects.

On some devices the external clock input is shared with one of the TPM channels. When a TPM channel is shared as the external clock input, the associated TPM channel cannot use the pin. (The channel can still be used in output compare mode as a software timer.) Also, if one of the TPM channels is used as the external clock input, the corresponding ELSnB:ELSnA control bits must be set to 0:0 so the channel is not trying to use the same pin.

### 10.3.2 TPMxCHn — TPMx Channel n I/O Pins

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the configuration of the channel. In some cases, no pin function is needed so the pin reverts to being controlled by general-purpose I/O controls. When a timer has control of a port pin, the port data and data direction registers do not affect the related pin(s). See the Pins and Connections chapter for additional information about shared pin functions.

## 10.4 Register Definition

The TPM includes:

- An 8-bit status and control register (TPMxSC)
- A 16-bit counter (TPMxCNTH:TPMxCNTL)
- A 16-bit modulo register (TPMxMODH:TPMxMODL)

Each timer channel has:

- An 8-bit status and control register (TPMxCnSC)
- A 16-bit channel value register (TPMxCnVH:TPMxCnVL)

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all TPM registers. This section refers to registers and control bits only by their names. A

### 10.5.2.2 Output Compare Mode

With the output compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel value registers of an output compare channel, the TPM can set, clear, or toggle the channel pin.

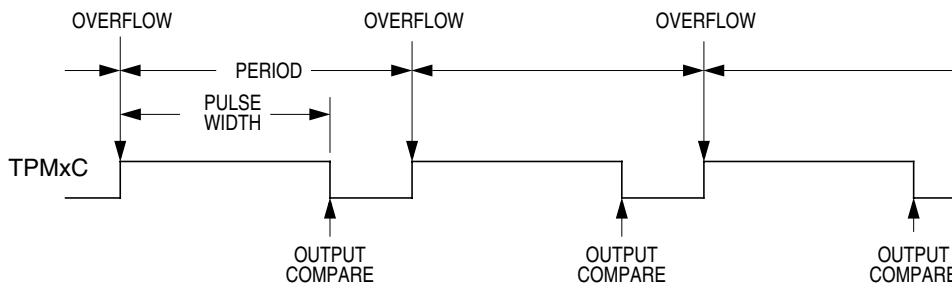
In output compare mode, values are transferred to the corresponding timer channel value registers only after both 8-bit bytes of a 16-bit register have been written. This coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) that can optionally generate a CPU interrupt request.

### 10.5.2.3 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 setting in the modulus register (TPMxMODH:TPMxMODL). The duty cycle is determined by the setting in the timer channel value register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. Duty cycle cases of 0 percent and 100 percent are possible.

As Figure 10-11 shows, the output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal. 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 10-11. PWM Period and Pulse Width (ELSnA = 0)**

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

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 either register, TPMxCnVH or TPMxCnVL, write to buffer registers. In edge-PWM mode, values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the value in the TPMxCnTH:TPMxCnTL counter is 0x0000. (The new duty cycle does not take effect until the next full period.)

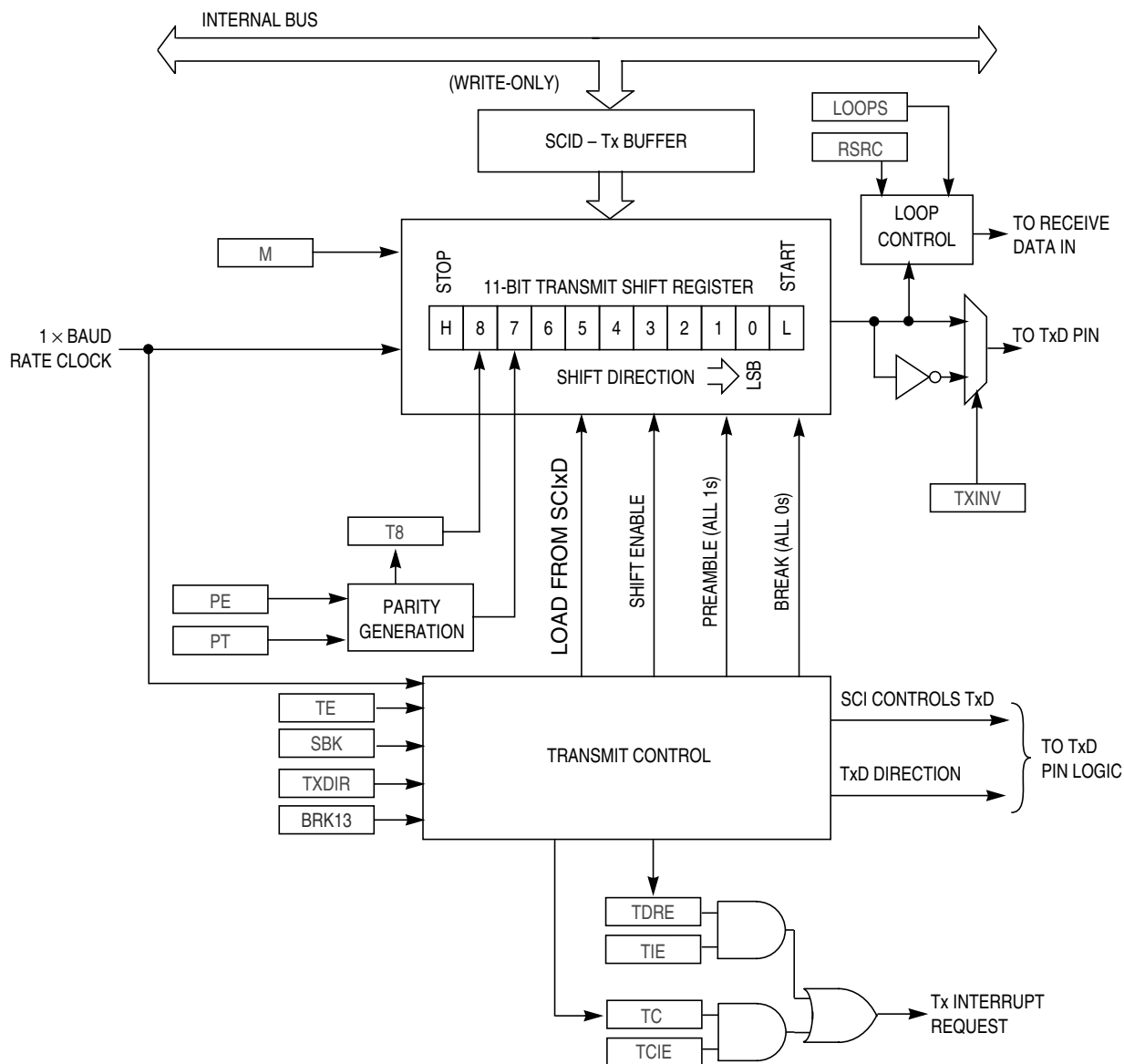


Figure 11-2. SCI Transmitter Block Diagram

systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever  $TC = 1$ . Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared by reading SCIxS1 while  $RDRF = 1$  and then reading SCIxD. If the SCI is configured to operate in 9-bit mode, an additional read to the SCIxC3 register is required to clear RDRF.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIxS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIxS1 while  $IDLE = 1$  and then reading SCIxD. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead and the data and any associated NF, FE, or PF condition is lost.

### 11.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

#### 11.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIxC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIxC3. For the receiver, the ninth bit is held in R8 in SCIxC3.

When transmitting 9-bit data, write to the T8 bit before writing to SCIxD for coherent writes to the transmit data buffer. If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCIxD to the shifter.

When receiving 9-bit data, clear the RDRF bit by reading both R8 and SCIxD. R8 and SCIxD can be read in either order.

### 13.4.1.1 START Signal

When the bus is free; i.e., no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 13-8, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

### 13.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.

0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 13-8).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address that is equal to its own slave address. The IIC cannot be master and slave at the same time.

However, if arbitration is lost during an address cycle, the IIC will revert to slave mode and operate correctly even if it is being addressed by another master.

### 13.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 13-8. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the 9th bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new calling by generating a repeated START signal.



### 15.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ\_STATUS and WRITE\_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0

= Unimplemented or Reserved

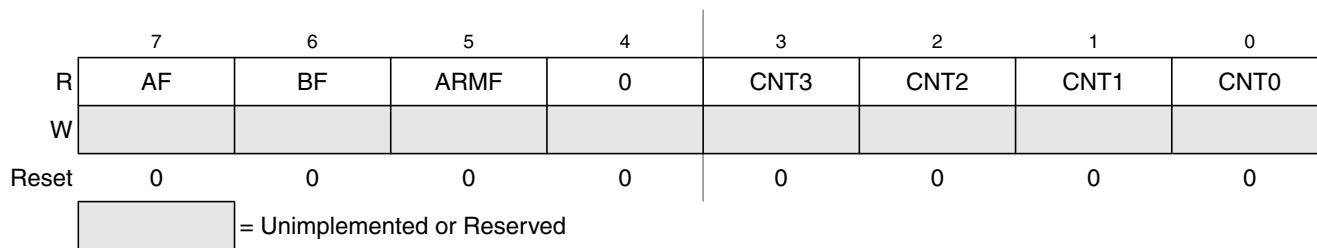
Figure 15-5. BDC Status and Control Register (BDCSCR)

Table 15-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	<b>Enable BDM (Permit Active Background Mode)</b> — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow active background mode commands
6 BDMACT	<b>Background Mode Active Status</b> — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	<b>BDC Breakpoint Enable</b> — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	<b>Force/Tag Select</b> — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode)
3 CLKSW	<b>Select Source for BDC Communications Clock</b> — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

### 15.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.



**Figure 15-9. Debug Status Register (DBGS)**

**Table 15-6. DBGS Register Field Descriptions**

Field	Description
7 AF	<b>Trigger Match A Flag</b> — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	<b>Trigger Match B Flag</b> — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	<b>Arm Flag</b> — While DBGEN = 1, this status bit is a read-only image of ARM in DBGIC. This bit is set by writing 1 to the ARM control bit in DBGIC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGIC. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	<b>FIFO Valid Count</b> — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

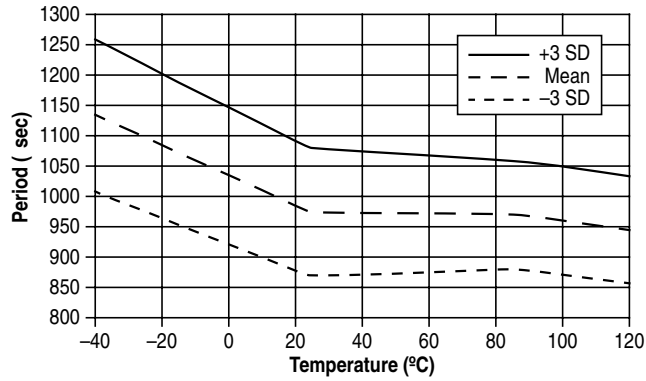


Figure A-10. Typical RTI Clock Period vs. Temperature

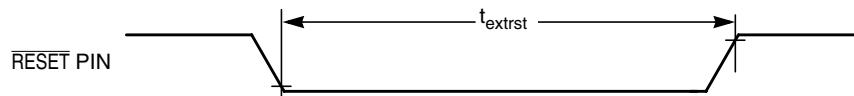


Figure A-11. Reset Timing

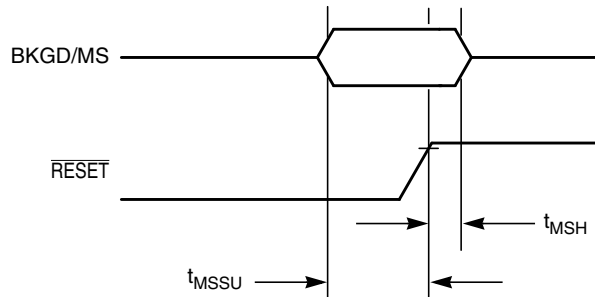


Figure A-12. Active Background Debug Mode Latch Timing

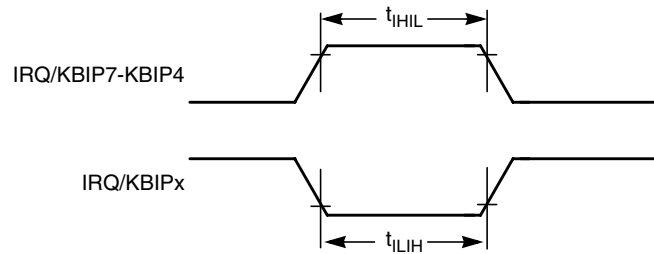


Figure A-13. IRQ/KBIPx Timing

## A.13 EMC Performance

Electromagnetic compatibility (EMC) performance is highly dependant on the environment in which the MCU resides. Board design and layout, circuit topology choices, location and characteristics of external components as well as MCU software operation all play a significant role in EMC performance. The system designer should consult Freescale applications notes such as AN2321, AN1050, AN1263, AN2764, and AN1259 for advice and guidance specifically targeted at optimizing EMC performance.

### A.13.1 Radiated Emissions

Microcontroller radiated RF emissions are measured from 150 kHz to 1 GHz using the TEM/GTEM Cell method in accordance with the IEC 61967-2 and SAE J1752/3 standards. The measurement is performed with the microcontroller installed on a custom EMC evaluation board while running specialized EMC test software. The radiated emissions from the microcontroller are measured in a TEM cell in two package orientations (North and East). For more detailed information concerning the evaluation results, conditions and setup, please refer to the EMC Evaluation Report for this device.

The maximum radiated RF emissions of the tested configuration in all orientations are less than or equal to the reported emissions levels.

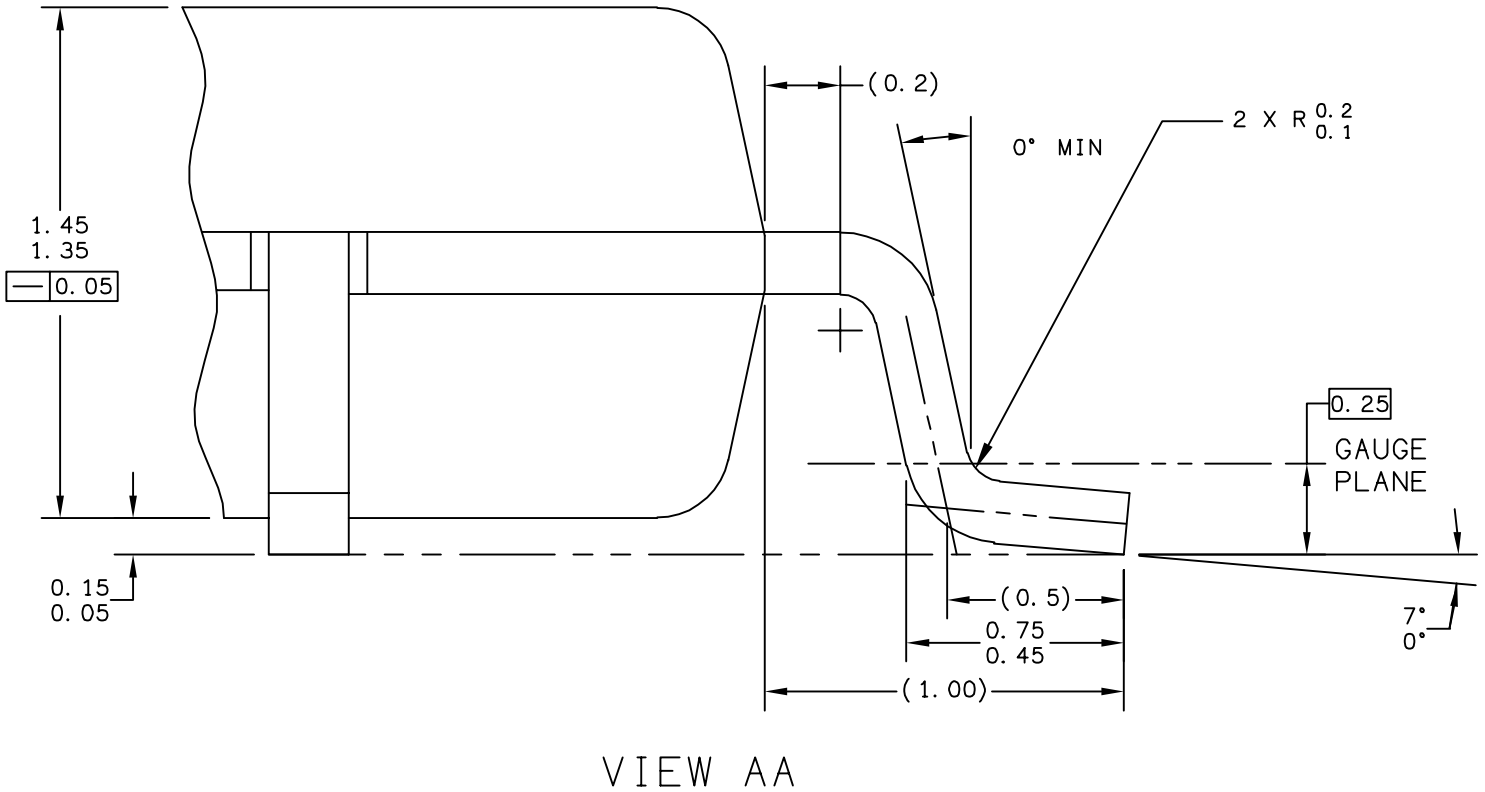
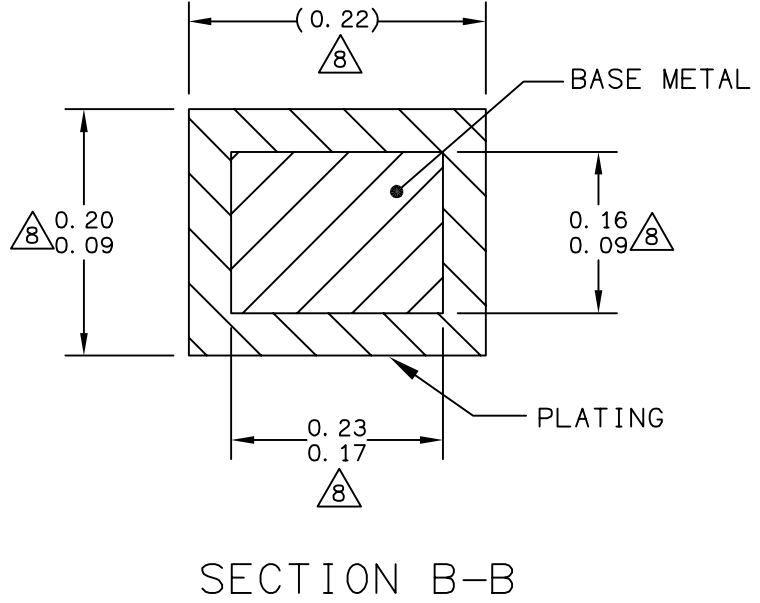
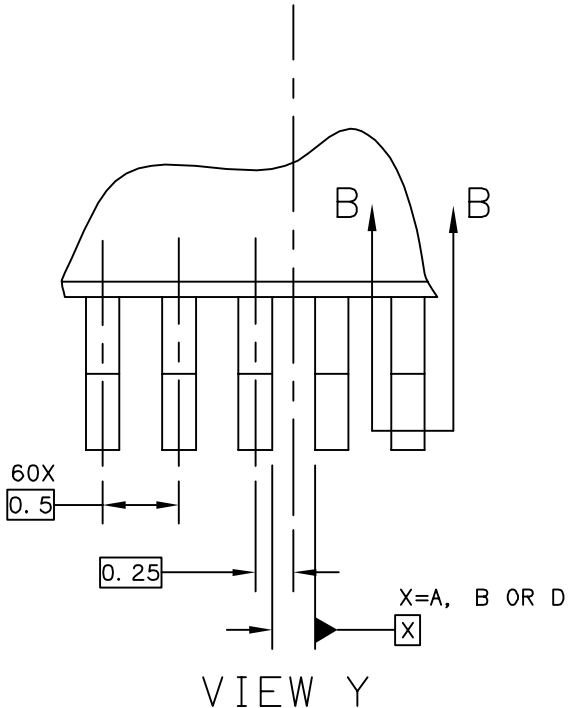
**Table A-17. Radiated Emissions**

Parameter	Symbol	Conditions	$f_{osc}/f_{BUS}$	Frequency	Level <sup>1</sup> (Max)	Unit	
Radiated emissions, electric field	$V_{RE\_TEM}$	$V_{DD} = 5.5V$ $T_A = +25^\circ C$ package type 64 QFP	4 MHz crystal 20 MHz Bus	0.15 – 50 MHz	16	dB $\mu$ V	
				50 – 150 MHz	3		
				150 – 500 MHz	1		
				500 – 1000 MHz	0		
				IEC Level	L		—
				SAE Level	2		—
			32 kHz crystal 8 MHz Bus	0.15 – 50 MHz	-1	dB $\mu$ V	
				50 – 150 MHz	-6		
				150 – 500 MHz	-10		
				500 – 1000 MHz	-10		
				IEC Level	N		—
				SAE Level	1		—

<sup>1</sup> Data based on qualification test results.

### A.13.2 Conducted Transient Susceptibility

Microcontroller transient conducted susceptibility is measured in accordance with an internal Freescale test method. The measurement is performed with the microcontroller installed on a custom EMC evaluation board and running specialized EMC test software designed in compliance with the test method.



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	<b>MECHANICAL OUTLINE</b>	PRINT VERSION NOT TO SCALE	
TITLE:           64LD LQFP, 10 X 10 X 1.4 PKG, 0.5 PITCH, CASE OUTLINE	DOCUMENT NO: 98ASS23234W	REV: E	
	CASE NUMBER: 840F-02	11 AUG 2006	
	STANDARD: JEDEC MS-026 BCD		