



Welcome to [E-XFL.COM](#)

### What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

### Applications of "[Embedded - Microcontrollers](#)"

#### Details

Product Status	Active
Core Processor	S08
Core Size	8-Bit
Speed	40MHz
Connectivity	I <sup>2</sup> C, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	16
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/s9s08sg32e1vtjr">https://www.e-xfl.com/product-detail/nxp-semiconductors/s9s08sg32e1vtjr</a>

Section Number	Title	Page
8.5	External Signal Description .....	119
8.6	Memory Map .....	119
8.6.1	Register Descriptions .....	119
8.7	Functional Description .....	121

## Chapter 9

### Analog-to-Digital Converter (S08ADC10V1)

9.1	Introduction .....	123
9.1.1	Channel Assignments.....	123
9.1.2	Analog Power and Ground Signal Names .....	124
9.1.3	Alternate Clock .....	124
9.1.4	Hardware Trigger.....	124
9.1.5	Temperature Sensor .....	124
9.1.6	Features .....	127
9.1.7	ADC Module Block Diagram .....	127
9.2	External Signal Description .....	128
9.2.1	Analog Power ( $V_{DDA}$ ) .....	129
9.2.2	Analog Ground ( $V_{SSA}$ ).....	129
9.2.3	Voltage Reference High ( $V_{REFH}$ ) .....	129
9.2.4	Voltage Reference Low ( $V_{REFL}$ ).....	129
9.2.5	Analog Channel Inputs ( $ADx$ ).....	129
9.3	Register Definition .....	129
9.3.1	Status and Control Register 1 (ADCSC1) .....	130
9.3.2	Status and Control Register 2 (ADCSC2) .....	131
9.3.3	Data Result High Register (ADCRH).....	132
9.3.4	Data Result Low Register (ADCRL).....	132
9.3.5	Compare Value High Register (ADCCVH).....	133
9.3.6	Compare Value Low Register (ADCCVL).....	133
9.3.7	Configuration Register (ADCCFG) .....	133
9.3.8	Pin Control 1 Register (APCTL1) .....	135
9.3.9	Pin Control 2 Register (APCTL2) .....	136
9.3.10	Pin Control 3 Register (APCTL3) .....	137
9.4	Functional Description .....	138
9.4.1	Clock Select and Divide Control .....	138
9.4.2	Input Select and Pin Control .....	139
9.4.3	Hardware Trigger.....	139
9.4.4	Conversion Control .....	139
9.4.5	Automatic Compare Function.....	142
9.4.6	MCU Wait Mode Operation.....	143
9.4.7	MCU Stop3 Mode Operation.....	143
9.4.8	MCU Stop2 Mode Operation.....	144
9.5	Initialization Information .....	144

Section Number	Title	Page
17.3	On-Chip Debug System (DBG) .....	279
17.3.1	Comparators A and B.....	279
17.3.2	Bus Capture Information and FIFO Operation .....	279
17.3.3	Change-of-Flow Information .....	280
17.3.4	Tag vs. Force Breakpoints and Triggers .....	280
17.3.5	Trigger Modes .....	281
17.3.6	Hardware Breakpoints .....	283
17.4	Register Definition .....	283
17.4.1	BDC Registers and Control Bits .....	283
17.4.2	System Background Debug Force Reset Register (SBDFFR) .....	285
17.4.3	DBG Registers and Control Bits .....	286

## Appendix A

### Electrical Characteristics

A.1	Introduction .....	291
A.2	Parameter Classification.....	291
A.3	Absolute Maximum Ratings.....	291
A.4	Thermal Characteristics.....	293
A.5	ESD Protection and Latch-Up Immunity .....	295
A.6	DC Characteristics.....	296
A.7	Supply Current Characteristics.....	302
A.8	External Oscillator (XOSC) Characteristics .....	306
A.9	Internal Clock Source (ICS) Characteristics .....	308
A.10	Analog Comparator (ACMP) Electricals .....	309
A.11	ADC Characteristics.....	310
A.12	AC Characteristics.....	316
A.12.1	Control Timing .....	316
A.12.2	TPM/MTIM Module Timing .....	318
A.12.3	SPI.....	319
A.13	Flash Specifications.....	323
A.14	EMC Performance.....	324
A.14.1	Radiated Emissions .....	324

## Appendix B

### Ordering Information and Mechanical Drawings

B.1	Ordering Information .....	325
B.1.1	Device Numbering Scheme .....	326
B.2	Package Information and Mechanical Drawings .....	326

Most background commands are not available in stop mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop and enter active background mode if the ENBDM bit is set. After entering background debug mode, all background commands are available.

### 3.6.2 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in [Table 3-1](#). Most of the internal circuitry of the MCU is powered off in stop2 with the exception of the RAM. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting the wake-up pin ( $\overline{\text{RESET}}$ ) on the MCU.

In addition, the real-time counter (RTC) can wake the MCU from stop2, if enabled.

Upon wake-up from stop2 mode, the MCU starts up as from a power-on reset (POR):

- All module control and status registers are reset
- The LVD reset function is enabled and the MCU remains in the reset state if  $V_{DD}$  is below the LVD trip point (low trip point selected due to POR)
- The CPU takes the reset vector

In addition to the above, upon waking up from stop2, the PPDF bit in SPMSC2 is set. This flag is used to direct user code to go to a stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to PPDACK in SPMSC2.

To maintain I/O states for pins that were configured as general-purpose I/O before entering stop2, the user must restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, then the pins will switch to their reset states when PPDACK is written.

For pins that were configured as peripheral I/O, the user must reconfigure the peripheral module that interfaces to the pin before writing to the PPDACK bit. If the peripheral module is not enabled before writing to PPDACK, the pins will be controlled by their associated port control registers when the I/O latches are opened.

### 3.6.3 On-Chip Peripheral Modules in Stop Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case ( $\text{ENBDM} = 1$ ), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to [Section 3.6.2, “Stop2 Mode,”](#) and [Section 3.6.1, “Stop3 Mode,”](#) for specific information on system behavior in stop modes.

## 4.5.1 Features

Features of the FLASH memory include:

- FLASH size
  - MC9S08SG32: 32,768 bytes (64 pages of 512 bytes each)
  - MC9S08SG16: 16,384 bytes (32 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection and vector redirection
- Security feature for FLASH and RAM
- Auto power-down for low-frequency read accesses

## 4.5.2 Program and Erase Times

Before any program or erase command can be accepted, the FLASH clock divider register (FCDIV) must be written to set the internal clock for the FLASH module to a frequency ( $f_{FCLK}$ ) between 150 kHz and 200 kHz (see [Section 4.7.1, “FLASH Clock Divider Register \(FCDIV\)”](#)). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ( $1/f_{FCLK}$ ) is used by the command processor to time program and erase pulses. An integer number of these timing pulses are used by the command processor to complete a program or erase command.

[Table 4-5](#) shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK ( $f_{FCLK}$ ). The time for one cycle of FCLK is  $t_{FCLK} = 1/f_{FCLK}$ . The times are shown as a number of cycles of FCLK and as an absolute time for the case where  $t_{FCLK} = 5 \mu s$ . Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

**Table 4-5. Program and Erase Times**

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 $\mu s$
Byte program (burst)	4	20 $\mu s$ <sup>1</sup>
Page erase	4000	20 ms
Mass erase	20,000	100 ms

<sup>1</sup> Excluding start/end overhead

Table 4-11. FPROT Register Field Descriptions

Field	Description
7:1 FPS	<b>FLASH Protect Select Bits</b> — When FPDIS = 0, this 7-bit field determines the ending address of unprotected FLASH locations at the high address end of the FLASH. Protected FLASH locations cannot be erased or programmed.
0 FPDIS	<b>FLASH Protection Disable</b> 0 FLASH block specified by FPS[7:1] is block protected (program and erase not allowed). 1 No FLASH block is protected.

### 4.7.5 FLASH Status Register (FSTAT)

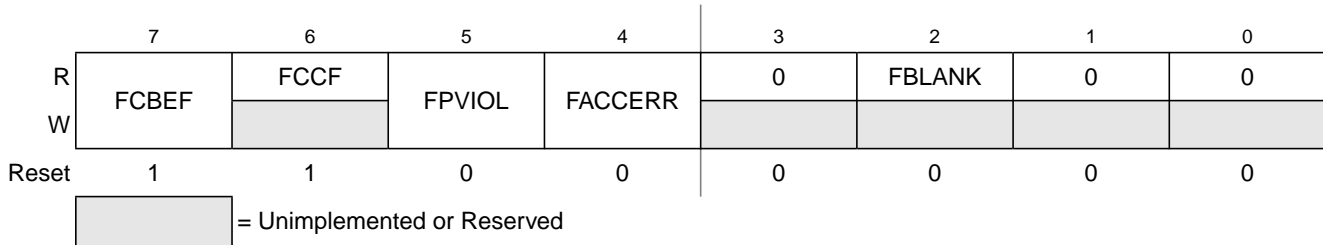


Figure 4-9. FLASH Status Register (FSTAT)

Table 4-12. FSTAT Register Field Descriptions

Field	Description
7 FCBEF	<b>FLASH Command Buffer Empty Flag</b> — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a 1 to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered. 0 Command buffer is full (not ready for additional commands). 1 A new burst program command can be written to the command buffer.
6 FCCF	<b>FLASH Command Complete Flag</b> — FCCF is set automatically when the command buffer is empty and no command is being processed. FCCF is cleared automatically when a new command is started (by writing 1 to FCBEF to register a command). Writing to FCCF has no meaning or effect. 0 Command in progress 1 All commands complete
5 FPVIOL	<b>Protection Violation Flag</b> — FPVIOL is set automatically when a command is written that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL. 0 No protection violation. 1 An attempt was made to erase or program a protected location.

### 6.4.2 Edge and Level Sensitivity

A valid edge or level on an enabled pin interrupt sets PTxIF in PTxSC. If PTxIE in PTxSC is set, an interrupt request is presented to the CPU. To clear PTxIF, write a 1 to PTxACK in PTxSC provided all enabled pin interrupt inputs are at their de-asserted levels. PTxIF remains set if any enabled pin interrupt is asserted while attempting to clear by writing a 1 to PTxACK.

### 6.4.3 Pull-up/Pull-down Resistors

The pin interrupts can be configured to use an internal pull-up/pull-down resistor using the associated I/O port pull-up enable register. If an internal resistor is enabled, the PTxES register is used to select whether the resistor is a pull-up (PTxESn = 0) or a pull-down (PTxESn = 1).

### 6.4.4 Pin Interrupt Initialization

When a pin interrupt is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization, the user should do the following:

1. Mask interrupts by clearing PTxIE in PTxSC.
2. Select the pin polarity by setting the appropriate PTxESn bits in PTxES.
3. If using internal pull-up/pull-down device, configure the associated pull enable bits in PTxPE.
4. Enable the interrupt pins by setting the appropriate PTxPSn bits in PTxPS.
5. Write to PTxACK in PTxSC to clear any false interrupts.
6. Set PTxIE in PTxSC to enable interrupts.

## 6.5 Pin Behavior in Stop Modes

Pin behavior following execution of a STOP instruction depends on the stop mode that is entered. An explanation of pin behavior for the various stop modes follows:

- Stop2 mode is a partial power-down mode, whereby I/O latches are maintained in their state as before the STOP instruction was executed. CPU register status and the state of I/O registers should be saved in RAM before the STOP instruction is executed to place the MCU in stop2 mode. Upon recovery from stop2 mode, before accessing any I/O, the user should examine the state of the PPDF bit in the SPMSC2 register. If the PPDF bit is 0, I/O must be initialized as if a power on reset had occurred. If the PPDF bit is 1, I/O data previously stored in RAM, before the STOP instruction was executed, peripherals may require being initialized and restored to their pre-stop condition. The user must then write a 1 to the PPDACK bit in the SPMSC2 register. Access to I/O is now permitted again in the user application program.
- In stop3 mode, all I/O is maintained because internal logic circuitry stays powered up. Upon recovery, normal I/O function is available to the user.

Table 7-2. Instruction Set Summary (Sheet 3 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affecton CCR	
						V 1 1 H	I N Z C
BPL <i>rel</i>	Branch if Plus (if N = 0)	REL	2A rr	3	ppp	- 1 1 -	- - - -
BRA <i>rel</i>	Branch Always (if I = 1)	REL	20 rr	3	ppp	- 1 1 -	- - - -
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear (if (Mn) = 0)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 dd rr 03 dd rr 05 dd rr 07 dd rr 09 dd rr 0B dd rr 0D dd rr 0F dd rr	5 5 5 5 5 5 5 5	rpppp rpppp rpppp rpppp rpppp rpppp rpppp rpppp	- 1 1 -	- - - - ↑
BRN <i>rel</i>	Branch Never (if I = 0)	REL	21 rr	3	ppp	- 1 1 -	- - - -
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set (if (Mn) = 1)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 dd rr 02 dd rr 04 dd rr 06 dd rr 08 dd rr 0A dd rr 0C dd rr 0E dd rr	5 5 5 5 5 5 5 5	rpppp rpppp rpppp rpppp rpppp rpppp rpppp rpppp	- 1 1 -	- - - - ↑
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory (Mn ← 1)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 dd 12 dd 14 dd 16 dd 18 dd 1A dd 1C dd 1E dd	5 5 5 5 5 5 5 5	rFWpp rFWpp rFWpp rFWpp rFWpp rFWpp rFWpp rFWpp	- 1 1 -	- - - -
BSR <i>rel</i>	Branch to Subroutine PC ← (PC) + \$0002 push (PCL); SP ← (SP) – \$0001 push (PCH); SP ← (SP) – \$0001 PC ← (PC) + <i>rel</i>	REL	AD rr	5	ssppp	- 1 1 -	- - - -
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 (A) = (M) Branch if (A) = (M) Branch if (X) = (M) Branch if (A) = (M) Branch if (A) = (M) Branch if (A) = (M)	DIR IMM IMM IX1+ IX+ SP1	31 dd 41 rrii 51 rrii 61 rrff 71 rrrrf 9E 61 f rr	5 4 4 5 5 6	rpppp pppp pppp rpppp rfppp prpppp	- 1 1 -	- - - -
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	- 1 1 -	- - - 0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	- 1 1 -	0 - - -
CLR <i>opr8a</i> CLRA CLR X CLR H CLR <i>opr8,X</i> CLR <i>,X</i> CLR <i>opr8,SP</i>	Clear M ← \$00 A ← \$00 X ← \$00 H ← \$00 M ← \$00 M ← \$00 M ← \$00	DIR INH INH INH IX1 IX SP1	3F dd 4F 5F 8C 6F ff 7F 9E 6F ff	5 1 1 1 5 4 6	rFWpp p p p rFWpp rFWp prFWpp	0 1 1 -	- 0 1 -



converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around  $1/2\text{LSB}$  and increases with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in [Section 9.6.2.3](#) reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

### 10.3.5 IIC Data I/O Register (IICD)

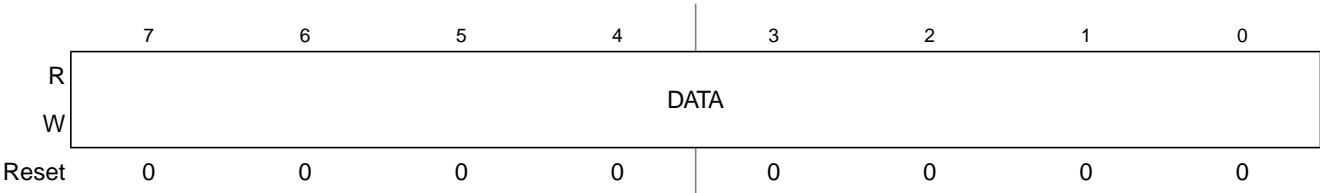


Figure 10-7. IIC Data I/O Register (IICD)

Table 10-8. IICD Field Descriptions

Field	Description
7–0 DATA	<b>Data</b> — In master transmit mode, when data is written to the IICD, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data.

#### NOTE

When transitioning out of master receive mode, the IIC mode should be switched before reading the IICD register to prevent an inadvertent initiation of a master receive data transfer.

In slave mode, the same functions are available after an address match has occurred.

The TX bit in IICC must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, reading the IICD does not initiate the receive.

Reading the IICD returns the last byte received while the IIC is configured in master receive or slave receive modes. The IICD does not reflect every byte transmitted on the IIC bus, nor can software verify that a byte has been written to the IICD correctly by reading it back.

In master transmit mode, the first byte of data written to IICD following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required  $R/\overline{W}$  bit (in position bit 0).

### 10.3.6 IIC Control Register 2 (IICC2)

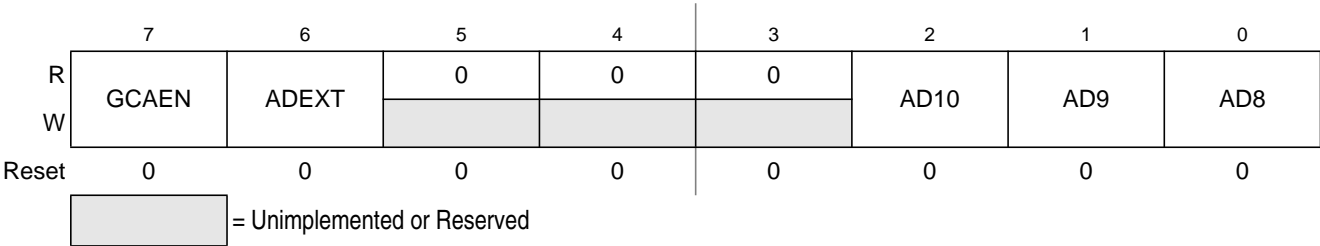


Figure 10-8. IIC Control Register (IICC2)

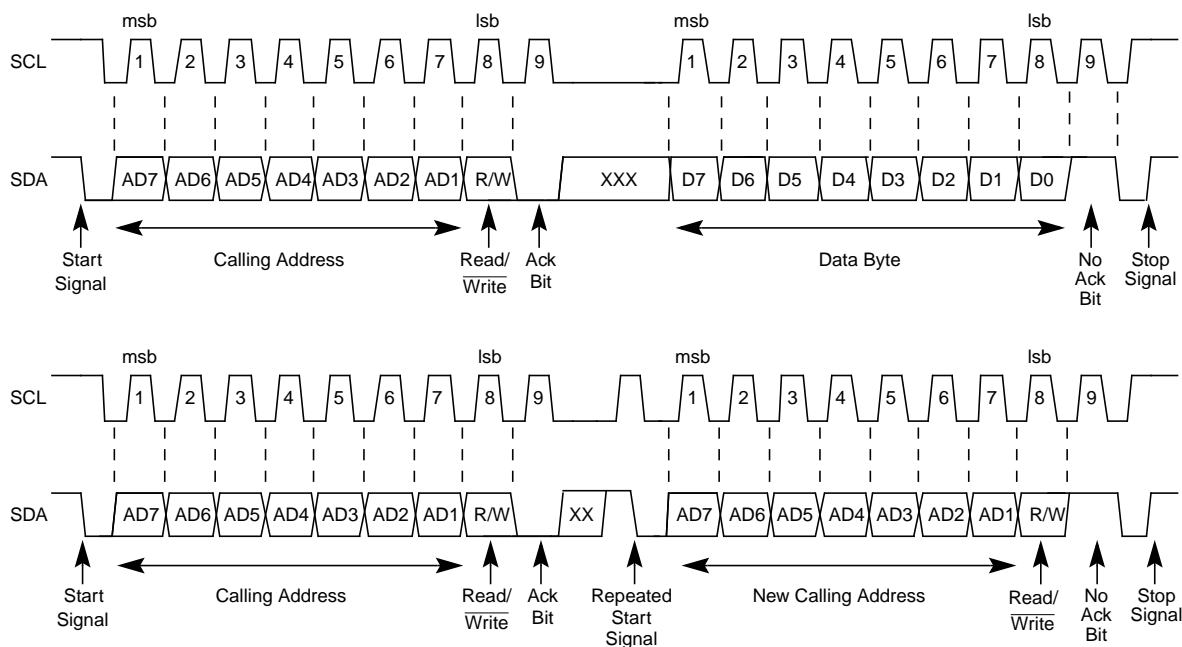


Figure 10-9. IIC Bus Transmission Signals

#### 10.4.1.1 Start Signal

When the bus is free, no master device is engaging the bus (SCL and SDA lines are at logical high), a master may initiate communication by sending a start signal. As shown in Figure 10-9, 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.

#### 10.4.1.2 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  $R/\overline{W}$  bit. The  $R/\overline{W}$  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 responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 10-9).

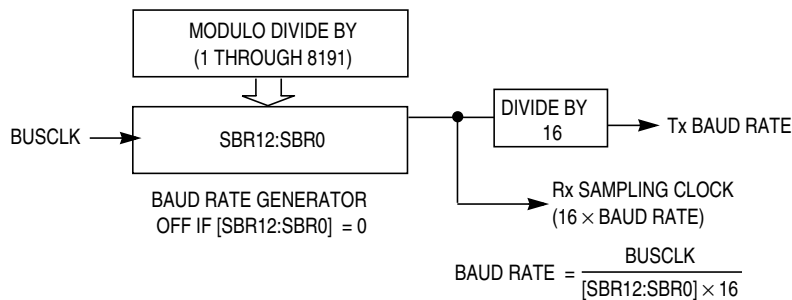
No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC reverts to slave mode and operates correctly even if it is being addressed by another master.

## Chapter 14

# Serial Communications Interface (S08SCIV4)

### 14.1 Introduction

[Figure 14-1](#) shows the MC9S08SG32 Series block diagram with the SCI module highlighted.



**Figure 14-12. SCI Baud Rate Generation**

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about 4.5percent for 8-bit data format and about 4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

### 14.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in [Figure 14-2](#).

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCIC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCID).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume M = 0, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCID.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

pin from a master and the MISO waveform applies to the MISO output from a slave. The  $\overline{SS}$  OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master  $\overline{SS}$  output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The  $\overline{SS}$  IN waveform applies to the slave select input of a slave.

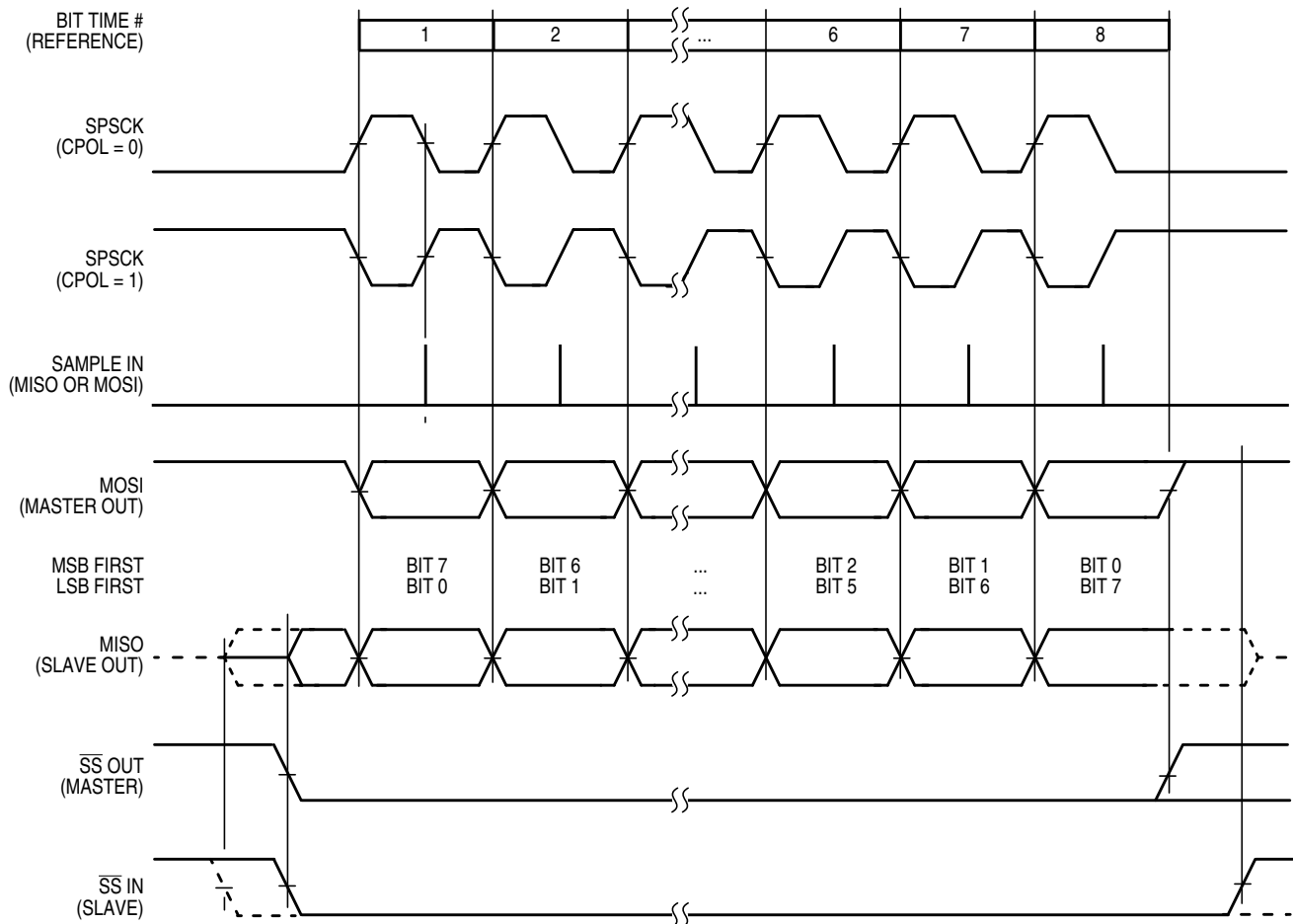
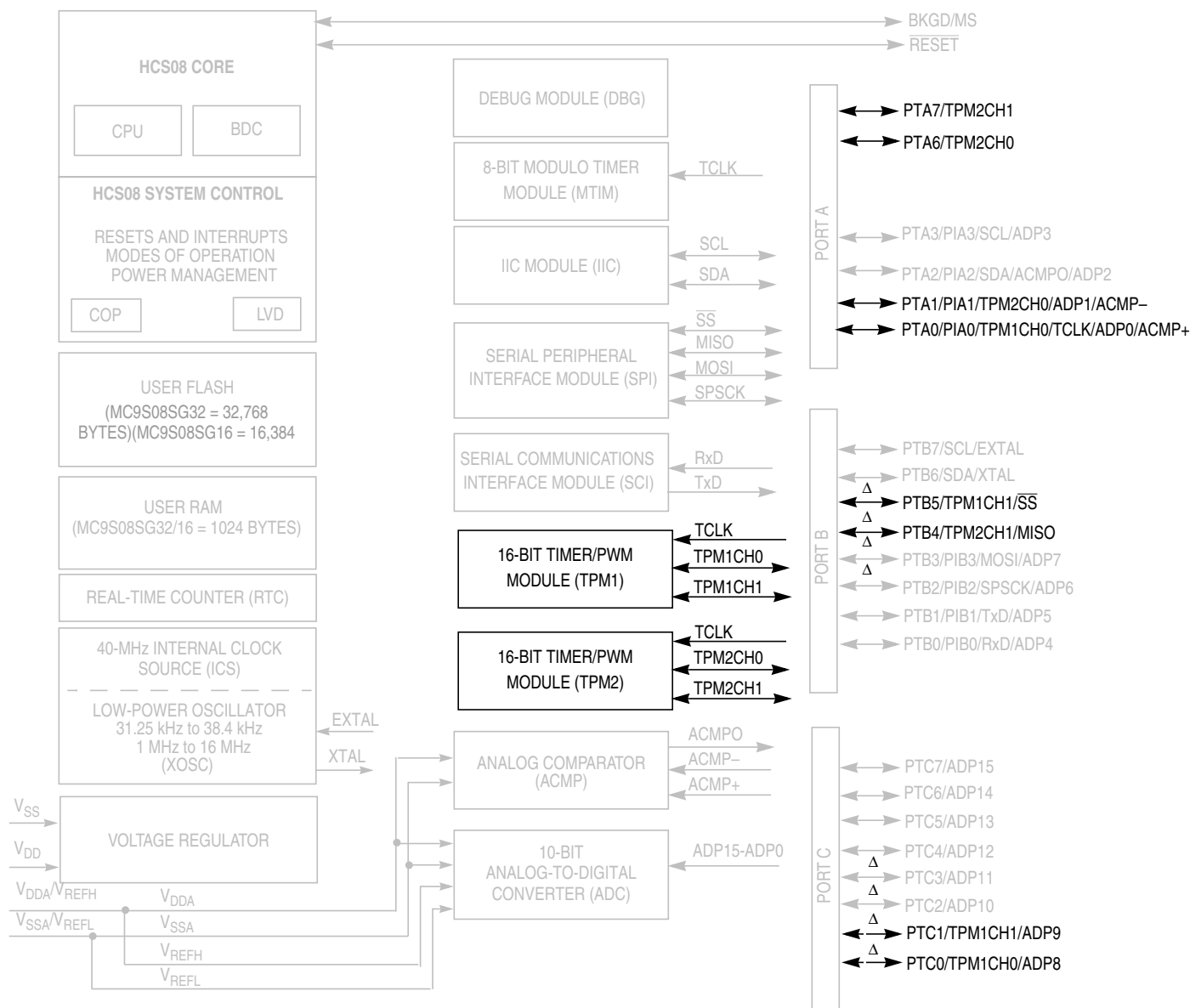


Figure 15-10. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when  $\overline{SS}$  goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 1, the slave's  $\overline{SS}$  input is not required to go to its inactive high level between transfers.

Figure 15-11 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ( $\overline{SS}$  IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting



**Figure 16-1. MC9S08SG32 Series Block Diagram Highlighting TPM Block and Pins**

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.

TPMxMODH:TPMxMODL = 0x0008  
TPMxCnVH:TPMxCnVL = 0x0005

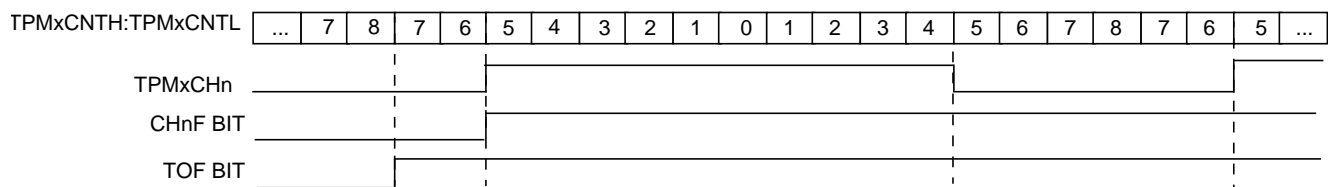


Figure 16-5. High-True Pulse of a Center-Aligned PWM

TPMxMODH:TPMxMODL = 0x0008  
TPMxCnVH:TPMxCnVL = 0x0005

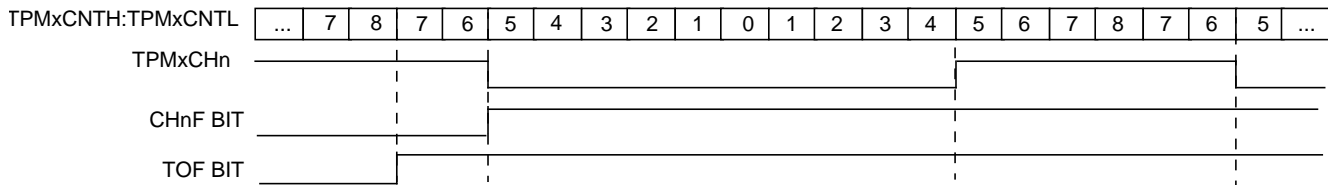


Figure 16-6. Low-True Pulse of a Center-Aligned PWM



are used for PWM & output compare operation once normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

## 16.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

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, CLKS<sub>B</sub>:CLKS<sub>A</sub>, 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, CLKS<sub>B</sub>:CLKS<sub>A</sub>=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 CLKS<sub>B</sub>:CLKS<sub>A</sub> 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).

### 17.3.6 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 17.3.5, “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.

## 17.4 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.

### 17.4.1 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.

### 17.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

### 17.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn't necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.

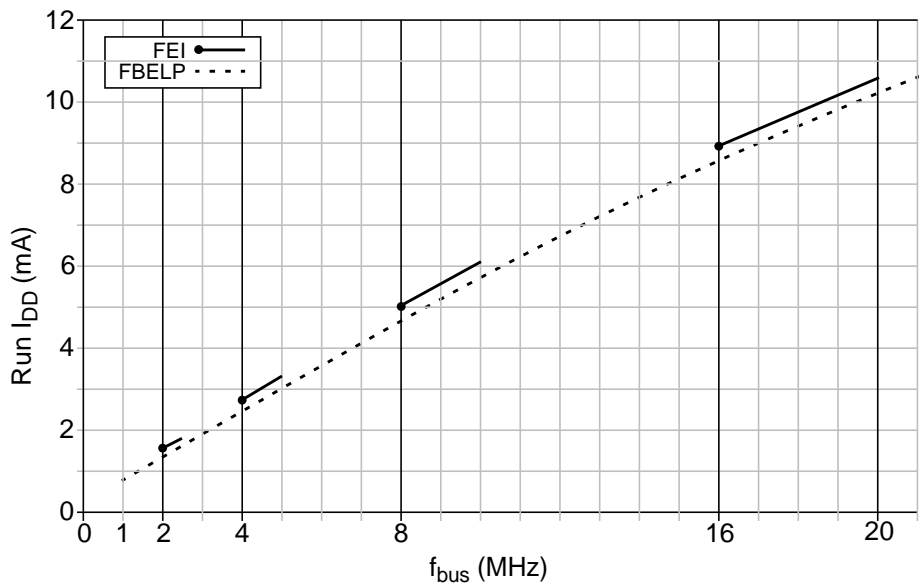


Figure A-5. Typical Run  $I_{DD}$  vs. Bus Frequency ( $V_{DD} = 5V$ )

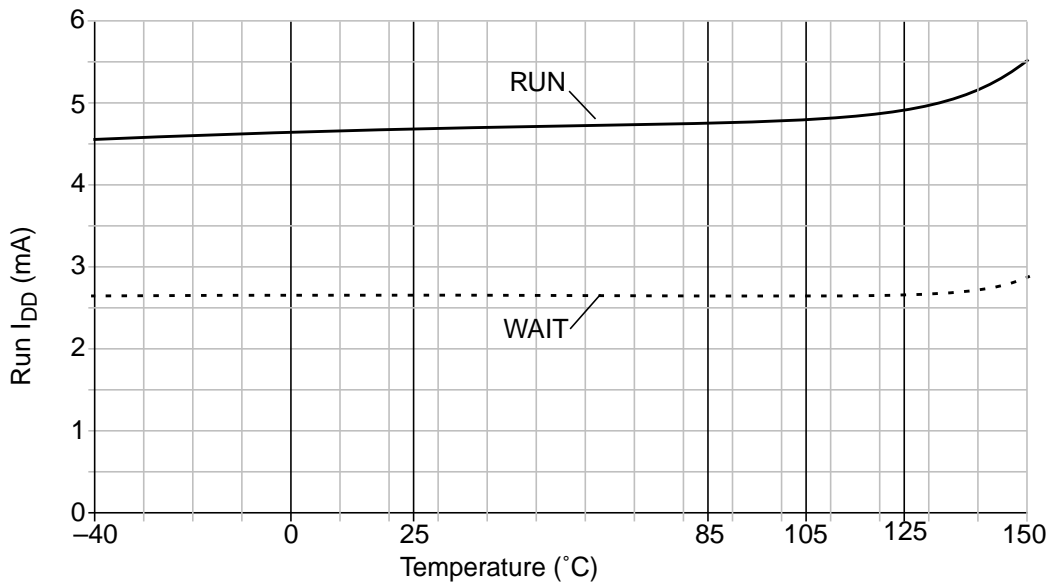


Figure A-6. Typical Run and Wait  $I_{DD}$  vs. Temperature ( $V_{DD} = 5V$ ;  $f_{bus} = 8MHz$ )