

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, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	24
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 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-HVQFN (5x5)
Purchase URL	<a href="https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc9s08gt16acfce">https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc9s08gt16acfce</a>

## Chapter 3

# Modes of Operation

### 3.1 Introduction

The operating modes of the MC9S08GT16A/GT8A are described in this section. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

#### 3.1.1 Features

- Active background mode for code development
- Wait mode:
  - CPU shuts down to conserve power
  - System clocks running
  - Full voltage regulation maintained
- Stop modes:
  - Stop1 — Full power down of internal circuits for maximum power savings
  - Stop2 — Partial power down of internal circuits, RAM contents retained
  - Stop3 — All internal circuits powered for fast recovery

### 3.2 Run Mode

This is the normal operating mode for the MC9S08GT16A/GT8A. This mode is selected when the BKGD/MS pin is high at the rising edge of reset. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFFE:0xFFFF after reset.

### 3.3 Active Background Mode

The active background mode functions are managed through the background debug controller (BDC) in the HCS08 core. The BDC, together with the on-chip ICE debug module (DBG), provide the means for analyzing MCU operation during software development.

Active background mode is entered in any of five ways:

- When the BKGD/MS pin is low at the rising edge of reset
- When a BACKGROUND command is received through the BKGD pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint
- When encountering a DBG breakpoint

# Chapter 5

## Resets, Interrupts, and System Configuration

### 5.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08GT16A/GT8A. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this data manual. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog and real-time interrupt (RTI), are not part of on-chip peripheral systems with their own sections but are part of the system control logic.

#### 5.1.1 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation:
  - Power-on detection (POR)
  - Low voltage detection (LVD) with enable
  - External  $\overline{\text{RESET}}$  pin with enable
  - COP watchdog with enable and two timeout choices
  - Illegal opcode
  - Illegal address
  - Serial command from a background debug host
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vectors for each module (reduces polling overhead) (see [Table 5-1](#))

### 5.2 MCU Reset

Resetting the MCU provides a way to start processing from a known set of initial conditions. During reset, most control and status registers are forced to initial values and the program counter is loaded from the reset vector (0xFFFF:0xFFFF). On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup devices disabled. The I bit in the condition code register (CCR) is set to block maskable interrupts so the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to 0x00FF at reset.

The MC9S08GT16A/GT8A has eight sources for reset:

- Power-on reset (POR)
- Low-voltage detect (LVD)

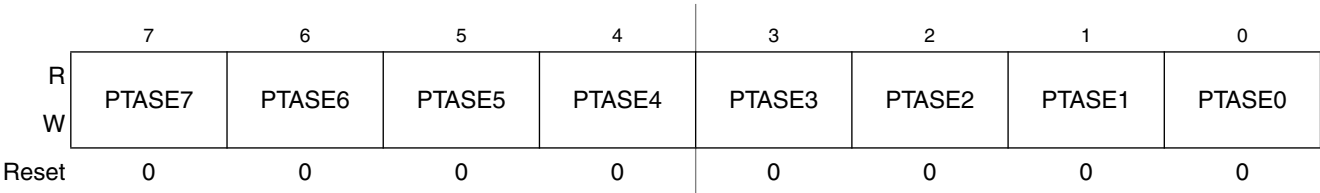


Figure 6-10. Slew Rate Control Enable for Port A (PTASE)

Table 6-3. PTASE Field Descriptions

Field	Description
7:0 PTASE[7:0]	<b>Slew Rate Control Enable for Port A Bits</b> — For port A pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port A pins that are configured as inputs, these bits are ignored. 0 Slew rate control disabled. 1 Slew rate control enabled.

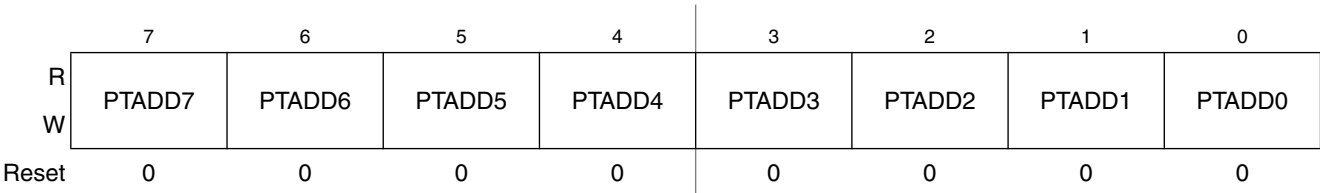


Figure 6-11. Data Direction for Port A (PTADD)

Table 6-4. PTADD Field Descriptions

Field	Description
7:0 PTADD[7:0]	<b>Data Direction for Port A Bits</b> — These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit n and PTAD reads return the contents of PTADn.

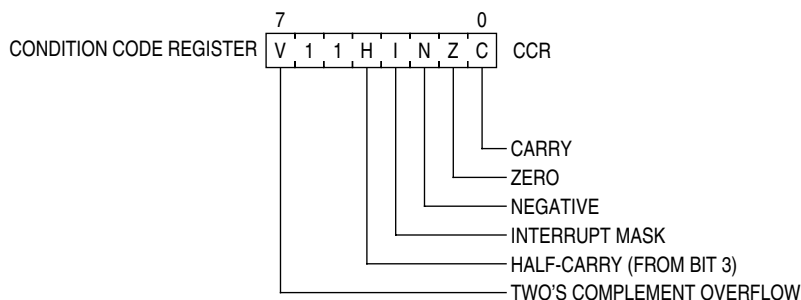


Figure 8-2. Condition Code Register

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.

### 8.3.6.7 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.

## 8.4 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.

### 8.4.1 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.

### 8.4.2 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

Table 9-1. ICGC1 Register Field Descriptions (continued)

Field	Description
2 OSCSTEN	<b>Enable Oscillator in Off Mode</b> — The OSCSTEN bit controls whether or not the oscillator circuit remains enabled when the ICG enters off mode. This bit has no effect if HGO = 1 and RANGE = 1. 0 Oscillator disabled when ICG is in off mode unless ENABLE is high, CLKS = 10, and REFST = 1. 1 Oscillator enabled when ICG is in off mode, CLKS = 1X and REFST = 1.
1 LOCD	<b>Loss of Clock Disable</b> 0 Loss of clock detection enabled. 1 Loss of clock detection disabled.



## Chapter 11

# Serial Communications Interface (S08SCIV1)

### 11.1 Introduction

The MC9S08GT16A/GT8A includes two independent serial communications interface (SCI) modules — sometimes called universal asynchronous receiver/transmitters (UARTs). Typically, these systems are used to connect to the RS232 serial input/output (I/O) port of a personal computer or workstation, and they can also be used to communicate with other embedded controllers.

A flexible, 13-bit, modulo-based baud rate generator supports a broad range of standard baud rates beyond 115.2 kbaud. Transmit and receive within the same SCI use a common baud rate, and each SCI module has a separate baud rate generator.

This SCI system offers many advanced features not commonly found on other asynchronous serial I/O peripherals on other embedded controllers. The receiver employs an advanced data sampling technique that ensures reliable communication and noise detection. Hardware parity, receiver wakeup, and double buffering on transmit and receive are also included.

### 11.1.1 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark

### 11.1.2 Modes of Operation

See [Section 11.3, “Functional Description,”](#) for a detailed description of SCI operation in the different modes.

- 8- and 9- bit data modes
- Stop modes — SCI is halted during all stop modes
- Loop modes

**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>

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

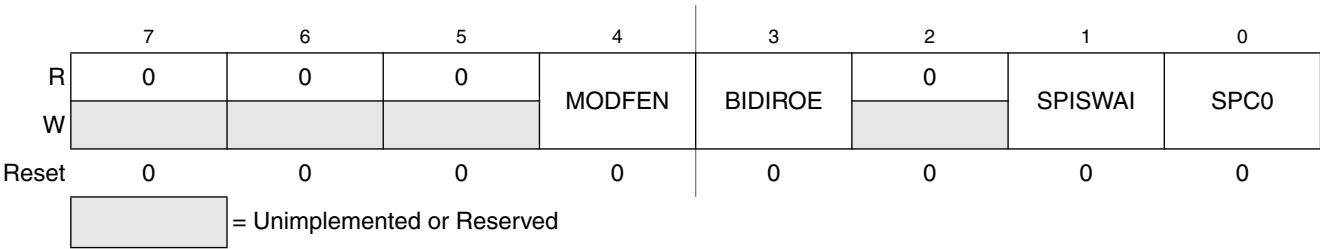
Field	Description
4 MSTR	<b>Master/Slave Mode Select</b> 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	<b>Clock Polarity</b> — This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer 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)**



### 13.3.3 IIC Control Register (IICC)

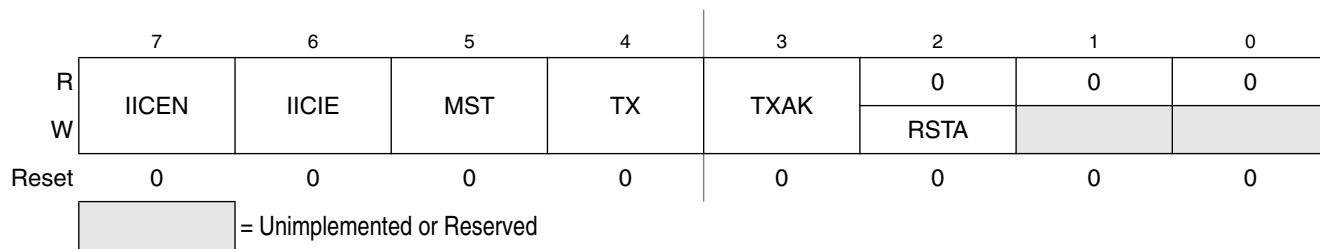


Figure 13-5. IIC Control Register (IICC)

Table 13-4. IICC Register Field Descriptions

Field	Description
7 IICEN	<b>IIC Enable</b> — The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled. 1 IIC is enabled.
6 IICIE	<b>IIC Interrupt Enable</b> — The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled. 1 IIC interrupt request enabled.
5 MST	<b>Master Mode Select</b> — The MST bit is changed from a 0 to a 1 when a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a STOP signal is generated and the mode of operation changes from master to slave. 0 Slave Mode. 1 Master Mode.
4 TX	<b>Transmit Mode Select</b> — The TX bit selects the direction of master and slave transfers. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. 0 Receive. 1 Transmit.
3 TXAK	<b>Transmit Acknowledge Enable</b> — This bit specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. 0 An acknowledge signal will be sent out to the bus after receiving one data byte. 1 No acknowledge signal response is sent.
2 RSTA	<b>Repeat START</b> — Writing a one to this bit will generate a repeated START condition provided it is the current master. This bit will always be read as a low. Attempting a repeat at the wrong time will result in loss of arbitration.

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

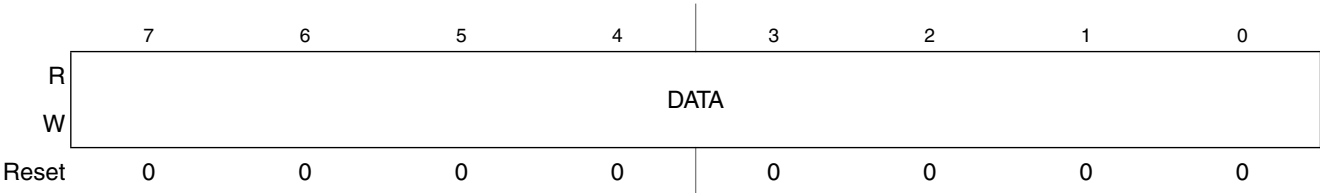


Figure 13-7. IIC Data I/O Register (IICD)

Table 13-6. IICD Register 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 transmitting 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.

Note that 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, then reading the IICD will not initiate the receive.

Reading the IICD will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IICD does not reflect every byte that is 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–bit 1) concatenated with the required R/W bit (in position bit 0).

#### 13.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see [Figure 13-8](#)).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

#### 13.4.1.5 Repeated START Signal

As shown in [Figure 13-8](#), a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

#### 13.4.1.6 Arbitration Procedure

The IIC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

#### 13.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see [Figure 13-9](#)). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.



## Chapter 14

### Analog-to-Digital Converter (S08ATDV3)

The MC9S08GT16A/GT8A provides one 8-channel analog-to-digital (ATD) module. The eight ATD channels share port B. Each channel individually can be configured for general-purpose I/O or for ATD functionality. All features of the ATD module as described in this section are available on the MC9S08GT16A/GT8A. Electrical parametric information for the ATD may be found in [Appendix A](#), “Electrical Characteristics.”

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a development system is connected, it can pull both BKGD and  $\overline{\text{RESET}}$  low, release  $\overline{\text{RESET}}$  to select active background mode rather than normal operating mode, then release BKGD. It is not necessary to reset the target MCU to communicate with it through the background debug interface.

## 15.2.2 Communication Details

The BDC serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

The clock switch (CLKSW) control bit in the BDC status and control register allows the user to select the BDC clock source. The BDC clock source can either be the bus or the alternate BDC clock source.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

## 15.3 On-Chip Debug System (DBG)

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

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

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in [Section 15.3.6, "Hardware Breakpoints."](#)

### 15.3.1 Comparators A and B

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

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

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

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

### 15.3.2 Bus Capture Information and FIFO Operation

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

## A.8 ATD Characteristics

**Table A-8. ATD Electrical Characteristics (Operating)**

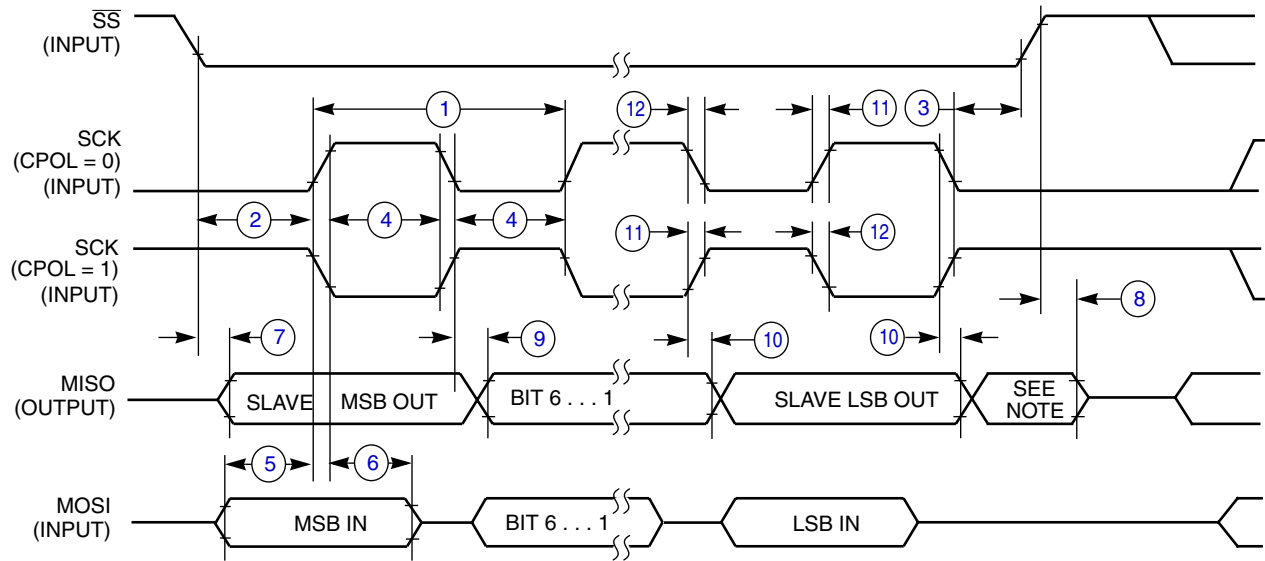
No.	Characteristic	Condition	Symbol	Min	Typ	Max	Unit
1	ATD supply <sup>1</sup>		$V_{DDAD}$	1.80	—	3.6	V
2	ATD supply current	Enabled	$I_{DDADrun}$	—	0.7	1.2	mA
		Disabled (ATDPU = 0 or STOP)	$I_{DDADstop}$	—	0.02	0.6	$\mu$ A
3	Differential supply voltage	$V_{DD} - V_{DDAD}$	$ V_{DDLT} $	—	—	100	mV
4	Differential ground voltage	$V_{SS} - V_{SSAD}$	$ V_{SDLT} $	—	—	100	mV
5	Reference potential, low		$ V_{REFL} $	—	—	$V_{SSAD}$	V
	Reference potential, high	$2.08V \leq V_{DDAD} \leq 3.6V$	$V_{REFH}$	2.08	—	$V_{DDAD}$	V
		$1.80V \leq V_{DDAD} < 2.08V$		$V_{DDAD}$	—	$V_{DDAD}$	
6	Reference supply current ( $V_{REFH}$ to $V_{REFL}$ )	Enabled	$I_{REF}$	—	200	300	$\mu$ A
		Disabled (ATDPU = 0 or STOP)	$I_{REF}$	—	<0.01	0.02	
7	Analog input voltage <sup>2</sup>		$V_{INDC}$	$V_{SSAD} - 0.3$	—	$V_{DDAD} + 0.3$	V

<sup>1</sup>  $V_{DDAD}$  must be at same potential as  $V_{DD}$ .

<sup>2</sup> Maximum electrical operating range, not valid conversion range.

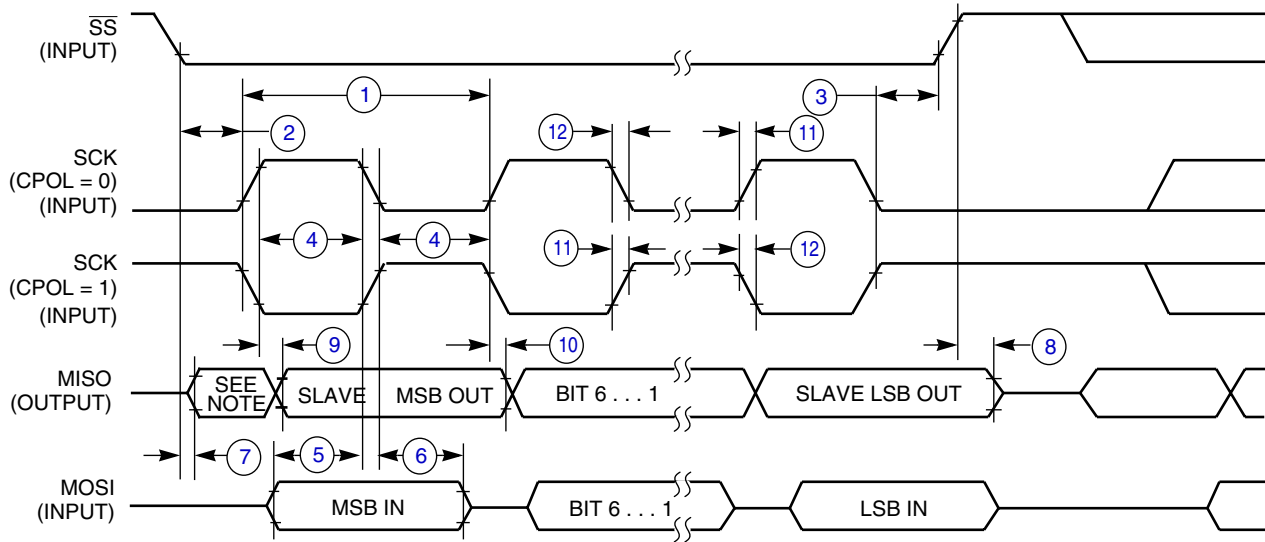
**Table A-9. ATD Timing/Performance Characteristics<sup>1</sup>**

No.	Characteristic	Condition	Symbol	Min	Typ	Max	Unit
1	ATD conversion clock frequency	$2.08V \leq V_{DDAD} \leq 3.6V$	$f_{ATDCLK}$	0.5	—	2.0	MHz
		$1.80V \leq V_{DDAD} < 2.08V$		0.5	—	1.0	
2	Conversion cycles (continuous convert) <sup>2</sup>		CC	28	28	<30	ATDCLK cycles
3	Conversion time (Including sample time)	$2.08V \leq V_{DDAD} \leq 3.6V$	$T_{conv}$	14.0	—	60.0	$\mu$ s
		$1.80V \leq V_{DDAD} < 2.08V$		28.0	—	60.0	
4	ATD sample time	$t_{ADS}$	$t_{ADS}$	—	14	—	ATDCLK cycles
5	Source impedance at input <sup>3</sup>		$R_{AS}$	—	—	10	k $\Omega$
6	Analog Input Voltage <sup>4</sup>		$V_{AIN}$	$V_{REFL}$		$V_{REFH}$	V



NOTE:  
1. Not defined but normally MSB of character just received

Figure A-19. SPI Slave Timing (CPHA = 0)



NOTE:  
1. Not defined but normally LSB of character just received

Figure A-20. SPI Slave Timing (CPHA = 1)