Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

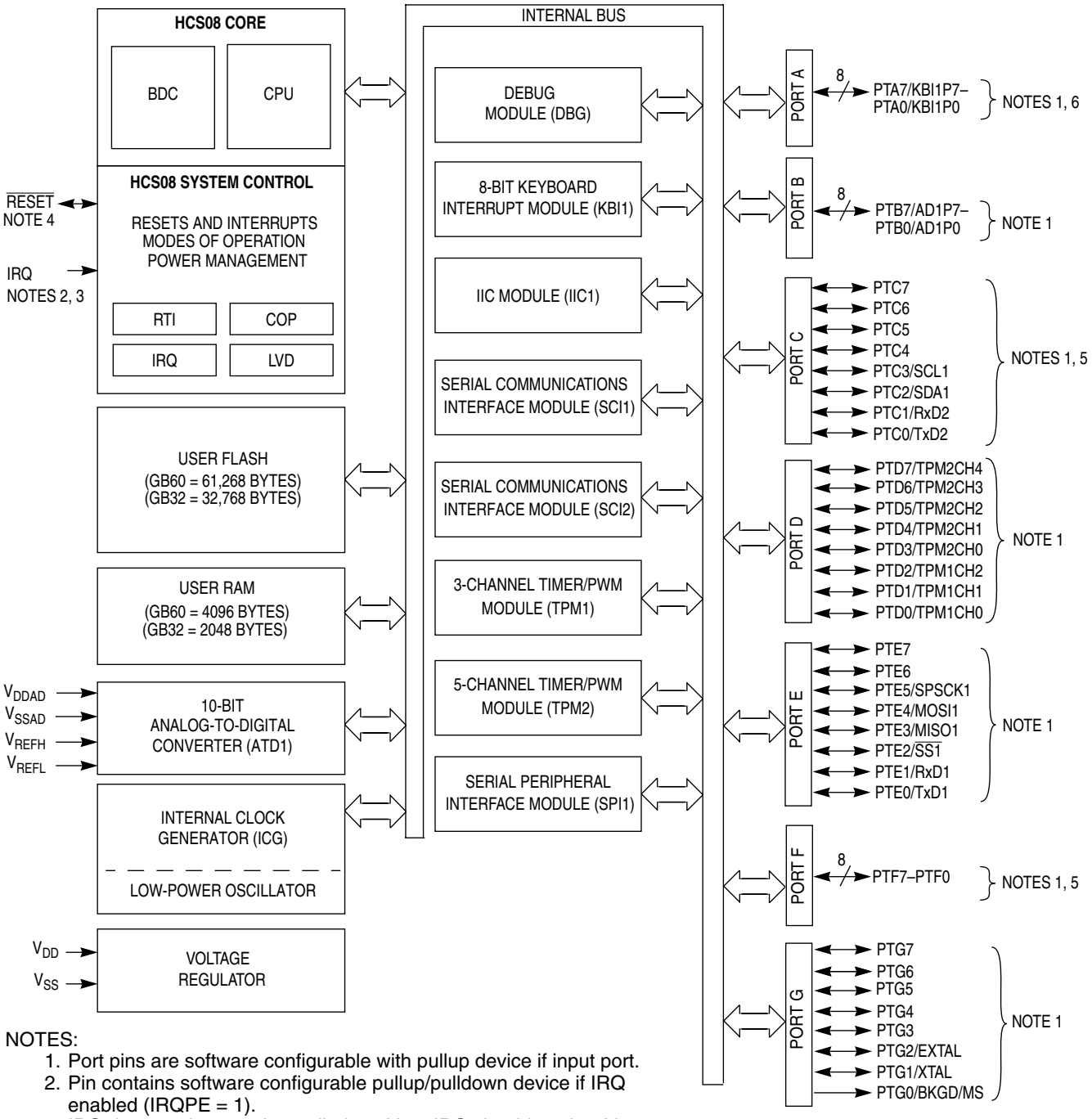
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	Obsolete
Core Processor	S08
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	56
Program Memory Size	60KB (60K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08gb60cfu

1.3 MCU Block Diagrams

These block diagrams show the structure of the MC9S08GB/GT MCUs.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1).
3. IRQ does not have a clamp diode to V_{DD}. IRQ should not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. High current drive
6. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown available when KBI enabled (KBIPn = 1).

Figure 1-1. MC9S08GBxx Block Diagram

Chapter 2 Pins and Connections

2.1 Introduction

This section describes signals that connect to package pins. It includes a pinout diagram, a table of signal properties, and detailed discussion of signals.

2.2 Device Pin Assignment

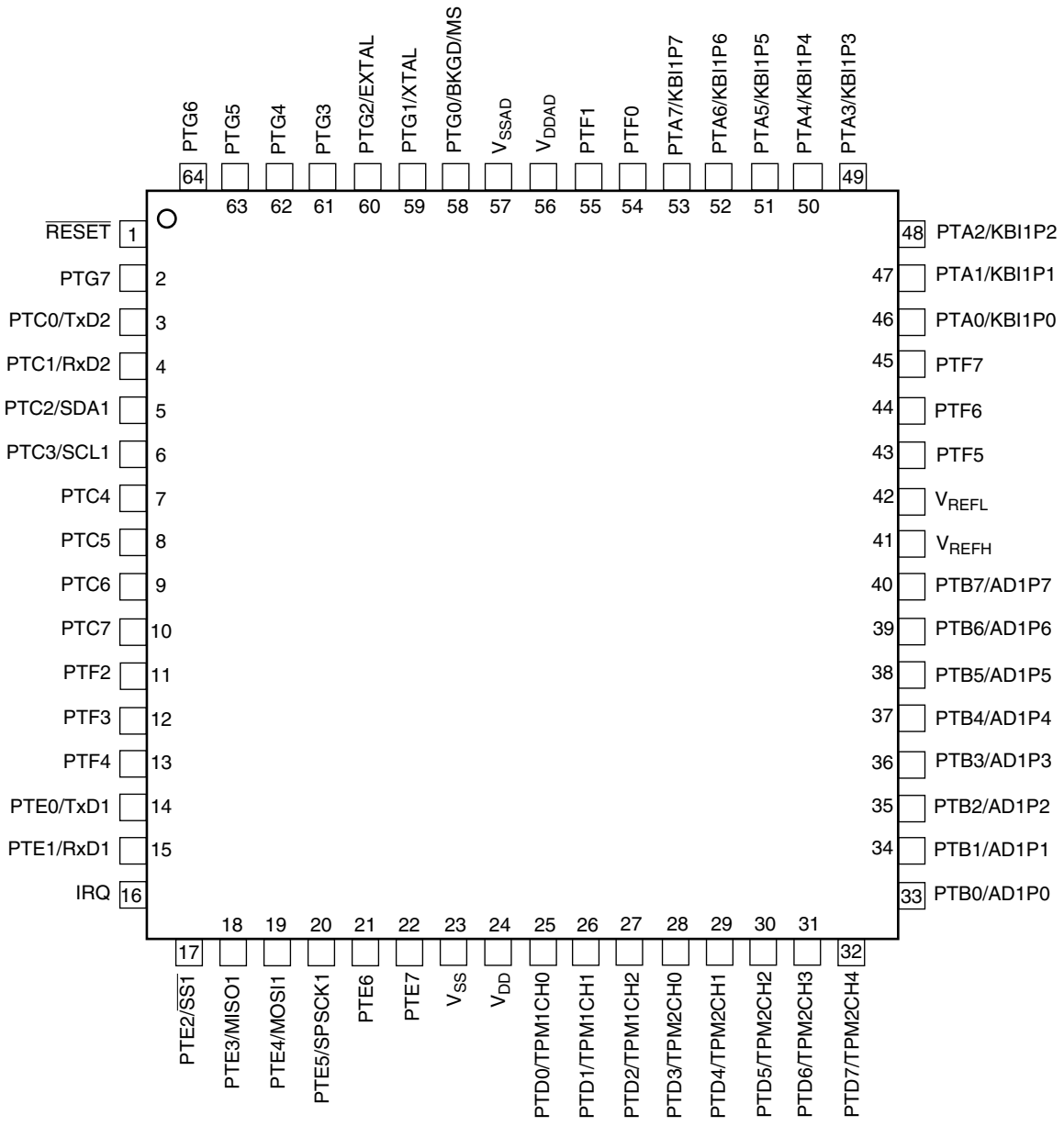


Figure 2-1. MC9S08GBxx in 64-Pin LQFP Package

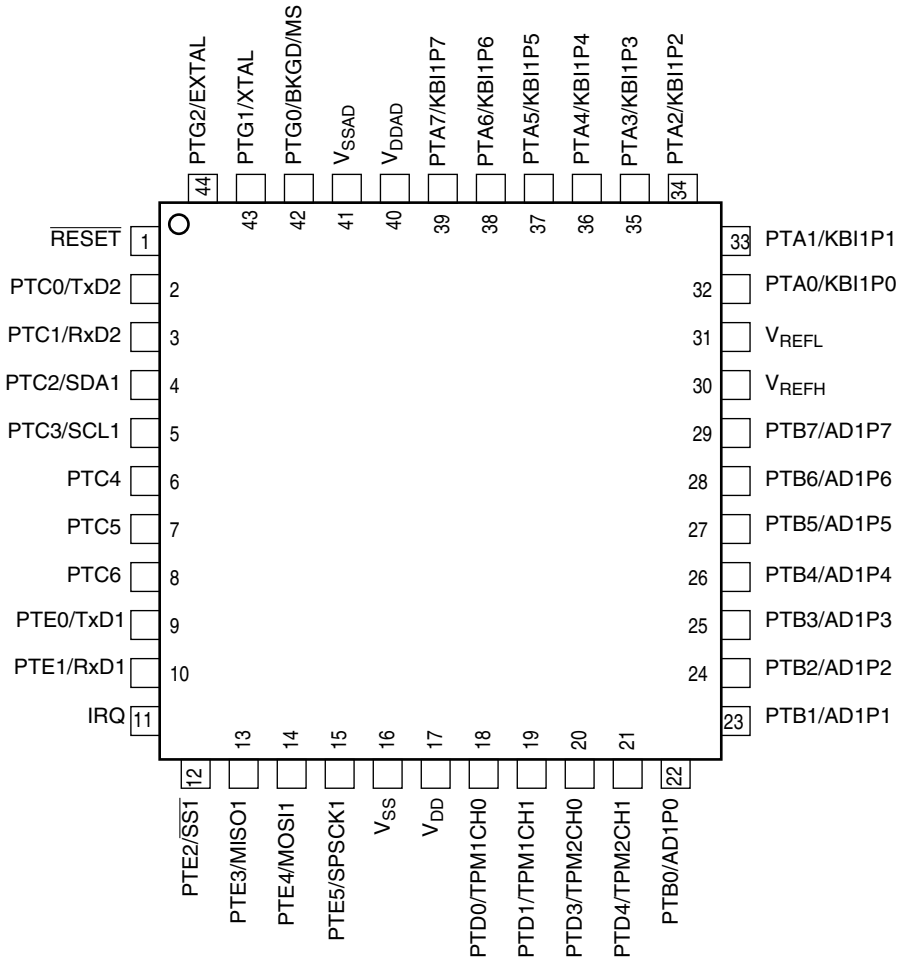


Figure 2-3. MC9S08GTxx in 44-Pin QFP Package

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD pin while the MCU is in run mode; non-intrusive commands can also be executed while the MCU is in the active background mode. Non-intrusive commands include:
  - Memory access commands
  - Memory-access-with-status commands
  - BDC register access commands
  - The BACKGROUND command
- Active background commands, which can be executed only while the MCU is in active background mode. Active background commands include commands to:
  - Read or write CPU registers
  - Trace one user program instruction at a time
  - Leave active background mode to return to the user's application program (GO)

The active background mode is used to program a bootloader or user application program into the FLASH program memory before the MCU is operated in run mode for the first time. When the MC9S08GB/GT is shipped from the Freescale Semiconductor factory, the FLASH program memory is erased by default unless specifically noted so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The active background mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the active background mode, refer to Chapter 15, "Development Support."

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The I bit in CCR is cleared when the CPU enters the wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits the wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available when the MCU is in wait 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 wait mode and enter active background mode.

3.6 Stop Modes

One of three stop modes is entered upon execution of a STOP instruction when the STOPE bit in the system option register is set. In all stop modes, all internal clocks are halted. If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU will not enter any of the stop modes and an illegal opcode reset is forced. The stop modes are selected by setting the appropriate bits in SPMSC2.

Chapter 4 Memory

4.1 MC9S08GB/GT Memory Map

As shown in Figure 4-1, on-chip memory in the MC9S08GB/GT series of MCUs consists of RAM, FLASH program memory for nonvolatile data storage, plus I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers ($0000 through $007F)
- High-page registers ($1800 through $182B)
- Nonvolatile registers ($FFB0 through $FFBF)

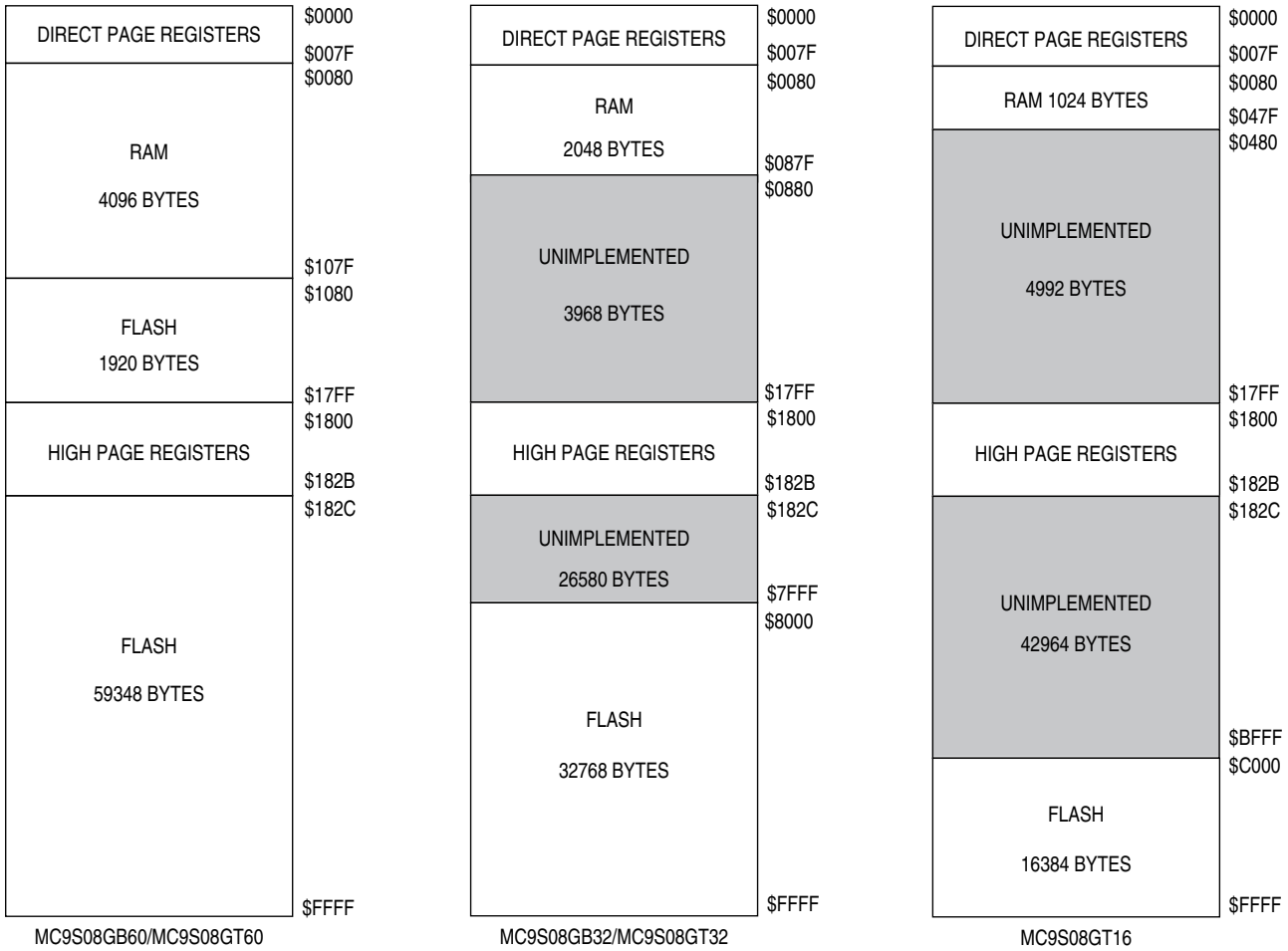


Figure 4-1. MC9S08GB/GT Memory Map

4.1.1 Reset and Interrupt Vector Assignments

Table 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale-provided equate file for the MC9S08GB/GT. For more details about

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$004F	Reserved	0	0	0	0	0	0	0	0
$0050	ATD1C	ATDPU	DJM	RES8	SGN	PRS			
$0051	ATD1SC	CCF	ATDIE	ATDCO	ATDCH				
$0052	ATD1RH	Bit 7	6	5	4	3	2	1	Bit 0
$0053	ATD1RL	Bit 7	6	5	4	3	2	1	Bit 0
$0054	ATD1PE	ATDPE7	ATDPE6	ATDPE5	ATDPE4	ATDPE3	ATDPE2	ATDPE1	ATDPE0
$0055–$0057	Reserved	—	—	—	—	—	—	—	—
$0058	IIC1A	ADDR							0
$0059	IIC1F	MULT		ICR					
$005A	IIC1C	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
$005B	IIC1S	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
$005C	IIC1D	DATA							
$005D–$005F	Reserved	—	—	—	—	—	—	—	—
$0060	TPM2SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
$0061	TPM2CNTH	Bit 15	14	13	12	11	10	9	Bit 8
$0062	TPM2CNTL	Bit 7	6	5	4	3	2	1	Bit 0
$0063	TPM2MODH	Bit 15	14	13	12	11	10	9	Bit 8
$0064	TPM2MODL	Bit 7	6	5	4	3	2	1	Bit 0
$0065	TPM2C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
$0066	TPM2C0VH	Bit 15	14	13	12	11	10	9	Bit 8
$0067	TPM2C0VL	Bit 7	6	5	4	3	2	1	Bit 0
$0068	TPM2C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
$0069	TPM2C1VH	Bit 15	14	13	12	11	10	9	Bit 8
$006A	TPM2C1VL	Bit 7	6	5	4	3	2	1	Bit 0
$006B	TPM2C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
$006C	TPM2C2VH	Bit 15	14	13	12	11	10	9	Bit 8
$006D	TPM2C2VL	Bit 7	6	5	4	3	2	1	Bit 0
$006E	TPM2C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
$006F	TPM2C3VH	Bit 15	14	13	12	11	10	9	Bit 8
$0070	TPM2C3VL	Bit 7	6	5	4	3	2	1	Bit 0
$0071	TPM2C4SC	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	0	0
$0072	TPM2C4VH	Bit 15	14	13	12	11	10	9	Bit 8
$0073	TPM2C4VL	Bit 7	6	5	4	3	2	1	Bit 0
$0074–$007F	Reserved	—	—	—	—	—	—	—	—

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

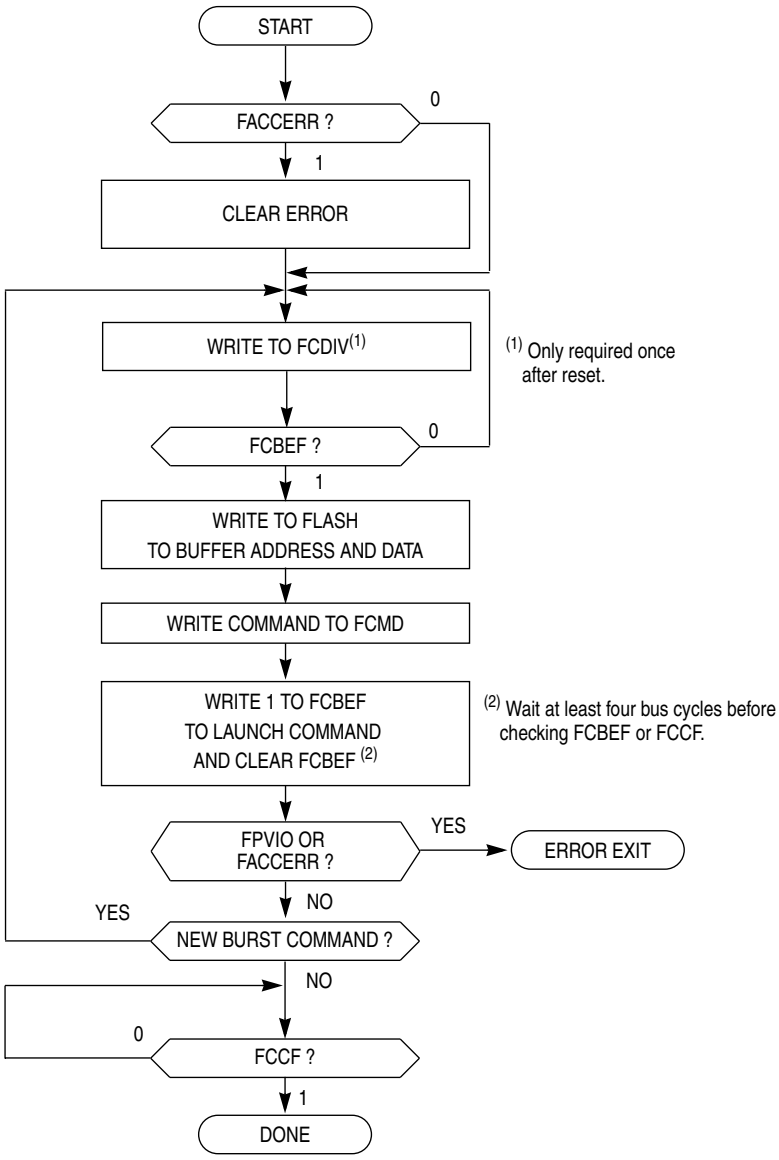


Figure 4-3. FLASH Burst Program Flowchart

4.4.5 Access Errors

An access error occurs whenever the command execution protocol is violated.

- Any of the following specific actions will cause the access error flag (FACCERR) in FSTAT to be set. FACCERR must be cleared by writing a 1 to FACCERR in FSTAT before any command can be processed.

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register. Whenever the MCU enters reset, the internal clock generator (ICG) module switches to self-clocked mode with the frequency of f_{Self_reset} selected. The reset pin is driven low for 34 internal bus cycles where the internal bus frequency is half the ICG frequency. After the 34 cycles are completed, the pin is released and will be pulled up by the internal pullup resistor, unless it is held low externally. After the pin is released, it is sampled after another 38 cycles to determine whether the reset pin is the cause of the MCU reset.

5.4 Computer Operating Properly (COP) Watchdog

The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP timer periodically. If the application program gets lost and fails to reset the COP before it times out, a system reset is generated to force the system back to a known starting point. The COP watchdog is enabled by the COPE bit in SOPT (see Section 5.8.4, “System Options Register (SOPT)” for additional information). The COP timer is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP timer.

After any reset, the COP timer is enabled. This provides a reliable way to detect code that is not executing as intended. If the COP watchdog is not used in an application, it can be disabled by clearing the COPE bit in the write-once SOPT register. Also, the COPT bit can be used to choose one of two timeout periods (2^{18} or 2^{13} cycles of the bus rate clock). Even if the application will use the reset default settings in COPE and COPT, the user should still write to write-once SOPT during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost.

The write to SRS that services (clears) the COP timer should not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

When the MCU is in active background mode, the COP timer is temporarily disabled.

5.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing resumes where it left off before the interrupt. Other than the software interrupt (SWI), which is a program instruction, interrupts are caused by hardware events such as an edge on the IRQ pin or a timer-overflow event. The debug module can also generate an SWI under certain circumstances.

If an event occurs in an enabled interrupt source, an associated read-only status flag will become set. The CPU will not respond until and unless the local interrupt enable is set to 1 to enable the interrupt. The I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset which masks (prevents) all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

Chapter 6 Parallel Input/Output

6.1 Introduction

This section explains software controls related to parallel input/output (I/O). The MC9S08GBxx has seven I/O ports which include a total of 56 general-purpose I/O pins (one of these pins is output only). The MC9S08GTxx has six I/O ports which include a total of up to 39 general-purpose I/O pins, depending on the package (one pin, PTG0, is output only). See Chapter 2, "Pins and Connections," for more information about the logic and hardware aspects of these pins.

Many of these pins are shared with on-chip peripherals such as timer systems, external interrupts, or keyboard interrupts. When these other modules are not controlling the port pins, they revert to general-purpose I/O control. For each I/O pin, a port data bit provides access to input (read) and output (write) data, a data direction bit controls the direction of the pin, and a pullup enable bit enables an internal pullup device (provided the pin is configured as an input), and a slew rate control bit controls the rise and fall times of the pins.

NOTE

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user's reset initialization routine in the application program should either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

PTBDn — Port B Data Register Bit n (n = 0–7)

For port B pins that are inputs, reads return the logic level on the pin. For port B pins that are configured as outputs, reads return the last value written to this register.

Writes are latched into all bits of this register. For port B pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.

Reset forces PTBD to all 0s, but these 0s are not driven out on the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

PTBPEn — Pullup Enable for Port B Bit n (n = 0–7)

For port B pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port B pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.
0 = Internal pullup device disabled.

PTBSEn — Slew Rate Control Enable for Port B Bit n (n = 0–7)

For port B pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port B pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.
0 = Slew rate control disabled.

PTBDDn — Data Direction for Port B Bit n (n = 0–7)

These read/write bits control the direction of port B pins and what is read for PTBD reads.

1 = Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn.
0 = Input (output driver disabled) and reads return the pin value.

6.6.3 Port C Registers (PTCD, PTCPE, PTCSE, and PTCDD)

Port C includes eight general-purpose I/O pins that share with the SCI2 and IIC modules. Port C pins used as general-purpose I/O pins are controlled by the port C data (PTCD), data direction (PTCDD), pullup enable (PTCPE), and slew rate control (PTCSE) registers.

If the SCI2 takes control of a port C pin, the corresponding PTCDD bit is ignored. PTCSE can be used to provide slew rate on the SCI2 transmit pin, TxD2. PTCPE can be used, provided the corresponding PTCDD bit is 0, to provide a pullup device on the SCI2 receive pin, RxD2.

If the IIC takes control of a port C pin, the corresponding PTCDD bit is ignored. PTCSE can be used to provide slew rate on the IIC serial data pin (SDA1), when in output mode and the IIC clock pin (SCL1). PTCPE can be used, provided the corresponding PTCDD bit is 0, to provide a pullup device on the IIC serial data pin, when in receive mode.

Reads of PTCD will return the logic value of the corresponding pin, provided PTCDD is 0.

PTDDn — Port D Data Register Bit n (n = 0–7)

For port D pins that are inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register.

Writes are latched into all bits of this register. For port D pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.

Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

PTDPEn — Pullup Enable for Port D Bit n (n = 0–7)

For port D pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port D pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.
0 = Internal pullup device disabled.

PTDSEn — Slew Rate Control Enable for Port D Bit n (n = 0–7)

For port D pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port D pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.
0 = Slew rate control disabled.

PTDDDn — Data Direction for Port D Bit n (n = 0–7)

These read/write bits control the direction of port D pins and what is read for PTDD reads.

1 = Output driver enabled for port D bit n and PTDD reads return the contents of PTDDn.
0 = Input (output driver disabled) and reads return the pin value.

6.6.5 Port E Registers (PTED, PTEPE, PTESE, and PTEDD)

Port E includes eight general-purpose I/O pins that share with the SCI1 and SPI modules. Port E pins used as general-purpose I/O pins are controlled by the port E data (PTED), data direction (PTEDD), pullup enable (PTEPE), and slew rate control (PTESE) registers.

If the SCI1 takes control of a port E pin, the corresponding PTEDD bit is ignored. PTESE can be used to provide slew rate on the SCI1 transmit pin, TxD1. PTEPE can be used, provided the corresponding PTEDD bit is 0, to provide a pullup device on the SCI1 receive pin, RxD1.

If the SPI takes control of a port E pin, the corresponding PTEDD bit is ignored. PTESE can be used to provide slew rate on the SPI serial output pin (MOSI1 or MISO1) and serial clock pin (SPSCK1) depending on the SPI operational mode. PTEPE can be used, provided the corresponding PTEDD bit is 0, to provide a pullup device on the SPI serial input pins (MOSI1 or MISO1) and slave select pin ($\overline{SS1}$) depending on the SPI operational mode.

Reads of PTED will return the logic value of the corresponding pin, provided PTEDD is 0.

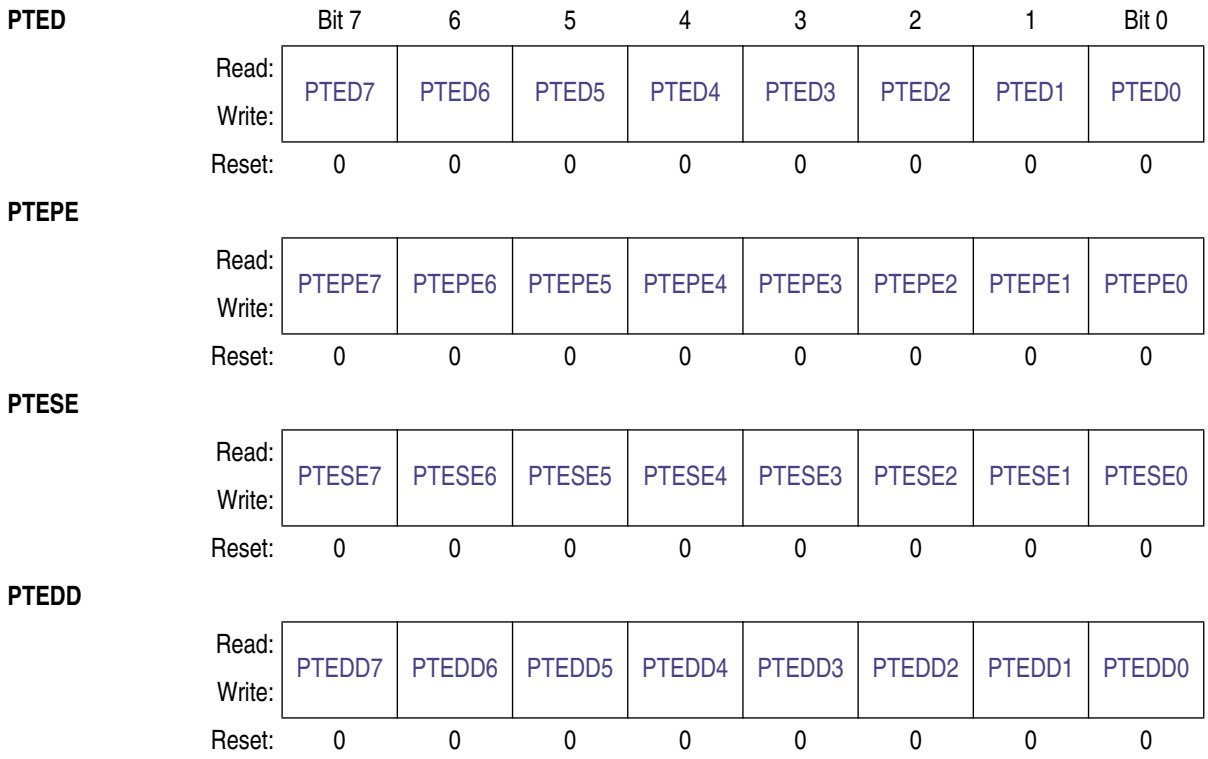


Figure 6-13. Port E Registers

PTEDn — Port E Data Register Bit n (n = 0–7)

For port E pins that are inputs, reads return the logic level on the pin. For port E pins that are configured as outputs, reads return the last value written to this register.

Writes are latched into all bits in this register. For port E pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.

Reset forces PTED to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

PTEPEn — Pullup Enable for Port E Bit n (n = 0–7)

For port E pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port E pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.

1 = Internal pullup device enabled.
0 = Internal pullup device disabled.

PTESEn — Slew Rate Control Enable for Port E Bit n (n = 0–7)

For port E pins that are outputs, these read/write control bits determine whether the slew rate controlled outputs are enabled. For port E pins that are configured as inputs, these bits are ignored.

1 = Slew rate control enabled.
0 = Slew rate control disabled.

8.5.5 BGND Instruction

The BGND instruction is new to the HCS08 compared to the M68HC08. BGND would not be used in normal user programs because it forces the CPU to stop processing user instructions and enter the active background mode. The only way to resume execution of the user program is through reset or by a host debug system issuing a GO, TRACE1, or TAGGO serial command through the background debug interface.

Software-based breakpoints can be set by replacing an opcode at the desired breakpoint address with the BGND opcode. When the program reaches this breakpoint address, the CPU is forced to active background mode rather than continuing the user program.

8.6 HCS08 Instruction Set Summary

Instruction Set Summary Nomenclature

The nomenclature listed here is used in the instruction descriptions in Table 8-1.

Operators

( )	=	Contents of register or memory location shown inside parentheses
←	=	Is loaded with (read: “gets”)
&	=	Boolean AND
|	=	Boolean OR
⊕	=	Boolean exclusive-OR
×	=	Multiply
÷	=	Divide
:	=	Concatenate
+	=	Add
–	=	Negate (two’s complement)

CPU registers

A	=	Accumulator
CCR	=	Condition code register
H	=	Index register, higher order (most significant) 8 bits
X	=	Index register, lower order (least significant) 8 bits
PC	=	Program counter
PCH	=	Program counter, higher order (most significant) 8 bits
PCL	=	Program counter, lower order (least significant) 8 bits
SP	=	Stack pointer

Memory and addressing

M	=	A memory location or absolute data, depending on addressing mode
M:M + $0001	=	A 16-bit value in two consecutive memory locations. The higher-order (most significant) 8 bits are located at the address of M, and the lower-order (least significant) 8 bits are located at the next higher sequential address.

10.6.4 PWM End-of-Duty-Cycle Events

For channels that are configured for PWM operation, there are two possibilities:

- When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period.
- When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle, which are the times when the timer counter matches the channel value register.

The flag is cleared by the 2-step sequence described in Section 10.6.1, "Clearing Timer Interrupt Flags."

10.7 TPM Registers and Control Bits

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 Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some MCU systems have more than one TPM, so register names include placeholder characters to identify which TPM and which channel is being referenced. For example, TPMxCnSC refers to timer (TPM) x, channel n and TPM1C2SC is the status and control register for timer 1, channel 2.

10.7.1 Timer x Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits that are used to configure the interrupt enable, TPM configuration, clock source, and prescale divisor. These controls relate to all channels within this timer module.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 10-5. Timer x Status and Control Register (TPMxSC)

11.2 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

11.3 SCI System Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

11.4 Baud Rate Generation

As shown in Figure 11-2, the clock source for the SCI baud rate generator is the bus-rate clock.

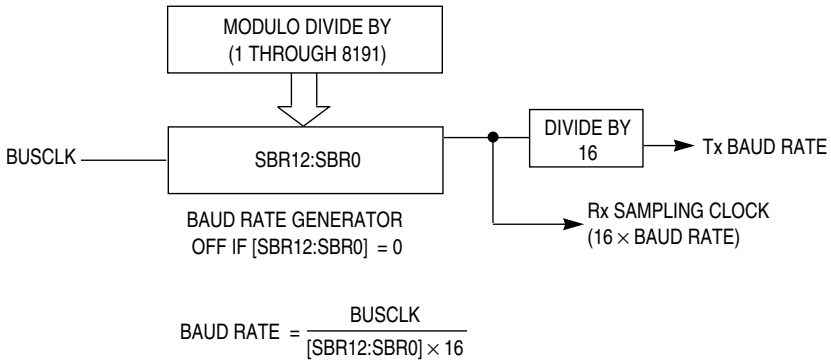


Figure 11-2. SCI Baud Rate Generation

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.

12.4.1 SPI Control Register 1 (SPI1C1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset:	0	0	0	0	0	1	0	0

Figure 12-7. SPI Control Register 1 (SPI1C1)

SPIE — SPI Interrupt Enable (for SPRF and MODF)

This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events.
1 = When SPRF or MODF is 1, request a hardware interrupt.
0 = Interrupts from SPRF and MODF inhibited (use polling).

SPE — SPI System Enable

Disabling the SPI halts any transfer that is in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty.
1 = SPI system enabled.
0 = SPI system inactive.

SPTIE — SPI Transmit Interrupt Enable

This is the interrupt enable bit for SPI transmit buffer empty (SPTEF).
1 = When SPTEF is 1, hardware interrupt requested.
0 = Interrupts from SPTEF inhibited (use polling).

MSTR — Master/Slave Mode Select
1 = SPI module configured as a master SPI device.
0 = SPI module configured as a slave SPI device.

CPOL — Clock Polarity

This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 12.3.1, “SPI Clock Formats,” for more details.
1 = Active-low SPI clock (idles high).
0 = Active-high SPI clock (idles low).

CPHA — Clock Phase

This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 12.3.1, “SPI Clock Formats,” for more details.
1 = First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer.
0 = First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer.

Table 12-2. SPI Baud Rate Prescaler Divisor

SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

SPR2:SPR1:SPR0 — SPI Baud Rate Divisor

This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Figure 12-3. The input to this divider comes from the SPI baud rate prescaler (see Figure 12-4). The output of this divider is the SPI bit rate clock for master mode.

Table 12-3. SPI Baud Rate Divisor

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 15.4.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

15.4.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

15.4.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

CLKSW — Select Source for BDC Communications Clock

CLKSW defaults to 0, which selects the alternate BDC clock source.
1 = MCU bus clock.
0 = Alternate BDC clock source.

WS — Wait or Stop Status

When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands.
1 = Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode.
0 = Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active).

WSF — Wait or Stop Failure Status

This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.)
1 = Memory access command failed because the CPU entered wait or stop mode.
0 = Memory access did not conflict with a wait or stop instruction.

DVF — Data Valid Failure Status

This status bit is not used in the MC9S08GB/GT because it does not have any slow access memory.
1 = Memory access command failed because CPU was not finished with a slow memory access.
0 = Memory access did not conflict with a slow memory access.

15.5.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to Section 15.3.4, “BDC Hardware Breakpoint.”

15.5.2 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial active background mode command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return $00.