



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	CANbus, I ² C, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	87
Program Memory Size	96KB (96K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 24x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=s9s08dv96f2vll

MC9S08DZ128

MC9S08DZ96

MC9S08DV128

MC9S08DV96

Data Sheet

HCS08
Microcontrollers

MC9S08DZ128
Rev. 1
5/2008

freescale.com

Section Number	Title	Page
9.3	Memory Map	203
9.3.1	Register Descriptions	203
9.4	Functional Description	205

Chapter 10 Analog-to-Digital Converter (S08ADC12V1)

10.1	Introduction	207
10.1.1	Channel Assignments	207
10.1.2	Analog Power and Ground Signal Names	207
10.1.3	Alternate Clock	208
10.1.4	Hardware Trigger	208
10.1.5	Temperature Sensor	209
10.1.6	Features	211
10.1.7	ADC Module Block Diagram	211
10.2	External Signal Description	212
10.2.1	Analog Power (V_{DDAD})	213
10.2.2	Analog Ground (V_{SSAD})	213
10.2.3	Voltage Reference High (V_{REFH})	213
10.2.4	Voltage Reference Low (V_{REFL})	213
10.2.5	Analog Channel Inputs (ADx)	213
10.3	Register Definition	213
10.3.1	Status and Control Register 1 (ADCSC1)	213
10.3.2	Status and Control Register 2 (ADCSC2)	215
10.3.3	Data Result High Register (ADCRH)	215
10.3.4	Data Result Low Register (ADCRL)	216
10.3.5	Compare Value High Register (ADCCVH)	216
10.3.6	Compare Value Low Register (ADCCVL)	217
10.3.7	Configuration Register (ADCCFG)	217
10.3.8	Pin Control 1 Register (APCTL1)	218
10.3.9	Pin Control 2 Register (APCTL2)	219
10.3.10	Pin Control 3 Register (APCTL3)	220
10.4	Functional Description	221
10.4.1	Clock Select and Divide Control	222
10.4.2	Input Select and Pin Control	222
10.4.3	Hardware Trigger	222
10.4.4	Conversion Control	222
10.4.5	Automatic Compare Function	225
10.4.6	MCU Wait Mode Operation	225
10.4.7	MCU Stop3 Mode Operation	226
10.4.8	MCU Stop2 Mode Operation	226
10.5	Initialization Information	227
10.5.1	ADC Module Initialization Example	227

Section Number	Title	Page
13.2	External Signal Description	312
13.2.1	SPSCK — SPI Serial Clock	312
13.2.2	MOSI — Master Data Out, Slave Data In	312
13.2.3	MISO — Master Data In, Slave Data Out	312
13.2.4	\overline{SS} — Slave Select	312
13.3	Modes of Operation.....	313
13.3.1	SPI in Stop Modes	313
13.4	Register Definition	313
13.4.1	SPI Control Register 1 (SPIxC1)	313
13.4.2	SPI Control Register 2 (SPIxC2)	314
13.4.3	SPI Baud Rate Register (SPIxBR)	315
13.4.4	SPI Status Register (SPIxS)	316
13.4.5	SPI Data Register (SPIxD)	317
13.5	Functional Description	318
13.5.1	SPI Clock Formats	318
13.5.2	SPI Interrupts	321
13.5.3	Mode Fault Detection	321

Chapter 14

Serial Communications Interface (S08SCIV4)

14.1	Introduction	323
14.1.1	SCI2 Configuration Information	323
14.1.2	Features	325
14.1.3	Modes of Operation	325
14.1.4	Block Diagram	326
14.2	Register Definition	328
14.2.1	SCI Baud Rate Registers (SCIxBDH, SCIxBDL)	328
14.2.2	SCI Control Register 1 (SCIxC1)	329
14.2.3	SCI Control Register 2 (SCIxC2)	330
14.2.4	SCI Status Register 1 (SCIxS1)	331
14.2.5	SCI Status Register 2 (SCIxS2)	333
14.2.6	SCI Control Register 3 (SCIxC3)	334
14.2.7	SCI Data Register (SCIxD)	335
14.3	Functional Description	335
14.3.1	Baud Rate Generation	335
14.3.2	Transmitter Functional Description	336
14.3.3	Receiver Functional Description	337
14.3.4	Interrupts and Status Flags	339
14.3.5	Additional SCI Functions	340

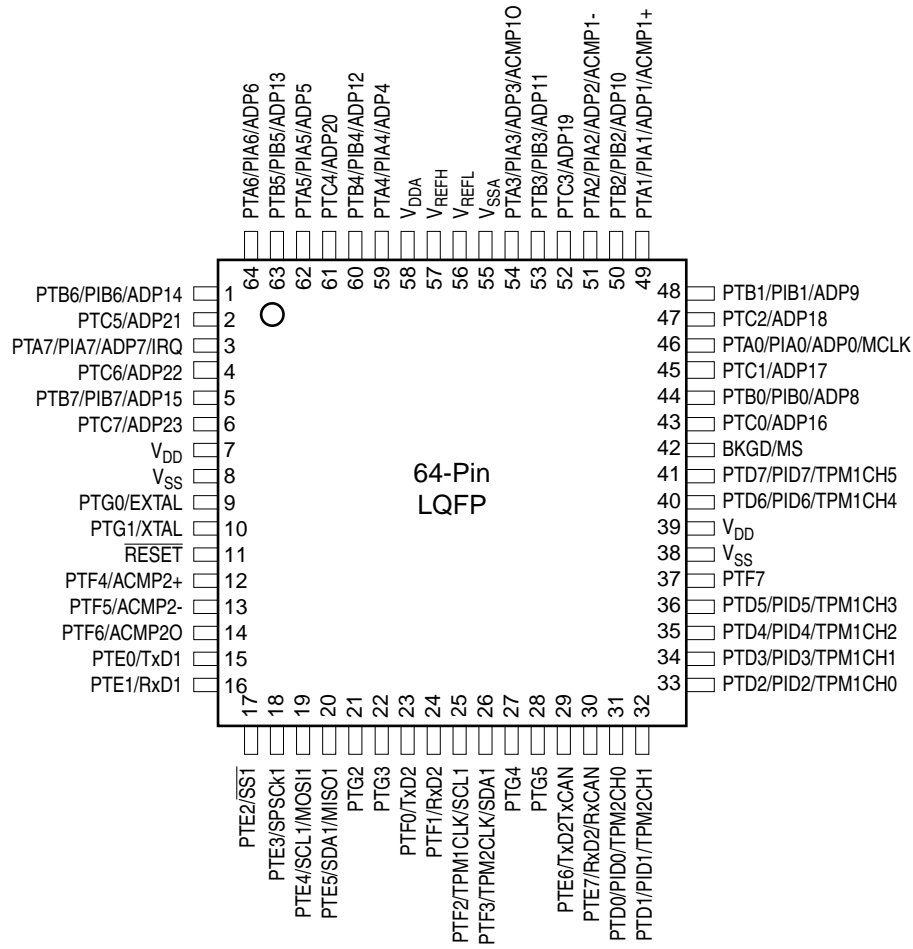


Figure 2-2. MC9S08DZ128 Series in 64-Pin LQFP Package

4.6.7 Block Protection

The block protection feature prevents the protected region of FLASH or EEPROM from program or erase changes. Block protection is controlled through the FLASH and EEPROM protection register (FPROT). The EPS bits determine the protected region of EEPROM and the FPS bits determine the protected region of FLASH. See [Section 4.6.11.4, “FLASH and EEPROM Protection Register \(FPROT and NVPROT\).”](#)

After exit from reset, FPROT is loaded with the contents of the NVPROT location, which is in the nonvolatile register block of the FLASH memory. FPROT cannot be changed directly from application software so a runaway program cannot alter the block protection settings. Because NVPROT is within the last sector of FLASH, if any amount of memory is protected, NVPROT is itself protected and cannot be altered (intentionally or unintentionally) by the application software. FPROT can be written through background debug commands, which provides a way to erase and reprogram protected FLASH memory.

One use for block protection is to block protect an area of FLASH memory for a bootloader program. This bootloader program then can be used to erase the rest of the FLASH memory and reprogram it. The bootloader is protected even if MCU power is lost during an erase and reprogram operation.

4.6.8 Vector Redirection

Whenever any FLASH is block protected, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register to 0. For redirection to occur, at least some portion of the FLASH memory must be block protected by programming the FPS bits in the NVPROT register. All interrupt vectors (memory locations 0x0_FF80 through 0x0_FFFD) are redirected, though the reset vector (0x0_FFFE:0x0_FFFF) is not.

For example, if 8192 bytes of FLASH are protected, the protected address region is from 0x0_E000 through 0x0_FFFF. The interrupt vectors (0x0_FF80 through 0x0_FFFD) are redirected to the locations 0x0_DF80 through 0x0_DFFD. If vector redirection is enabled and an interrupt occurs, the values in the locations 0x0_DFE0:0x0_DFE1 are used for the vector instead of the values in the locations 0x0_FFE0:0x0_FFE1. This allows the user to reprogram the unprotected portion of the FLASH with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

4.6.9 Security

The MC9S08DZ128 Series includes circuitry to prevent unauthorized access to the contents of FLASH, EEPROM, and RAM memory. When security is engaged, FLASH, EEPROM, and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two register bits (SEC[1:0]) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into the working FOPT register in high-page register space. A user engages security by programming the NVOPT location,

Chapter 7

Central Processor Unit (S08CPUV5)

7.1 Introduction

This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family. For a more detailed discussion, refer to the *HCS08 Family Reference Manual, volume 1*, Freescale Semiconductor document order number HCS08RMV1/D.

The HCS08 CPU is fully source- and object-code-compatible with the M68HC08 CPU. Several instructions and enhanced addressing modes were added to improve C compiler efficiency and to support a new background debug system which replaces the monitor mode of earlier M68HC08 microcontrollers (MCU).

7.1.1 Features

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- 64-KB CPU address space with banked memory management unit for greater than 64 KB
- 16-bit stack pointer (any size stack anywhere in 64-KB CPU address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
 - Inherent — Operands in internal registers
 - Relative — 8-bit signed offset to branch destination
 - Immediate — Operand in next object code byte(s)
 - Direct — Operand in memory at 0x0000–0x00FF
 - Extended — Operand anywhere in 64-Kbyte address space
 - Indexed relative to H:X — Five submodes including auto increment
 - Indexed relative to SP — Improves C efficiency dramatically
- Memory-to-memory data move instructions with four address mode combinations
- Overflow, half-carry, negative, zero, and carry condition codes support conditional branching on the results of signed, unsigned, and binary-coded decimal (BCD) operations
- Efficient bit manipulation instructions
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- STOP and WAIT instructions to invoke low-power operating modes

7.3 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the HCS08, memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte CPU address space. This arrangement means that the same instructions that access variables in RAM can also be used to access I/O and control registers or nonvolatile program space.

NOTE

For more information about extended addressing modes, see the Memory Management Unit section in the Memory chapter.

MCU derivatives with more than 64-Kbytes of memory also include a memory management unit (MMU) to support extended memory space. A PPAGE register is used to manage 16-Kbyte pages of memory which can be accessed by the CPU through a 16-Kbyte window from 0x8000 through 0xBFFF. The CPU includes two special instructions (CALL and RTC). CALL operates like the JSR instruction except that CALL saves the current PPAGE value on the stack and provides a new PPAGE value for the destination. RTC works like the RTS instruction except RTC restores the old PPAGE value in addition to the PC during the return from the called routine. The MMU also includes a linear address pointer register and data access registers so that the extended memory space operates as if it was a single linear block of memory. For additional information about the MMU, refer to the Memory chapter of this data sheet.

Some instructions use more than one addressing mode. For instance, move instructions use one addressing mode to specify the source operand and a second addressing mode to specify the destination address. Instructions such as BRCLR, BRSET, CBEQ, and DBNZ use one addressing mode to specify the location of an operand for a test and then use relative addressing mode to specify the branch destination address when the tested condition is true. For BRCLR, BRSET, CBEQ, and DBNZ, the addressing mode listed in the instruction set tables is the addressing mode needed to access the operand to be tested, and relative addressing mode is implied for the branch destination.

7.3.1 Inherent Addressing Mode (INH)

In this addressing mode, operands needed to complete the instruction (if any) are located within CPU registers so the CPU does not need to access memory to get any operands.

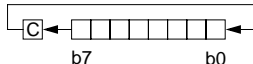
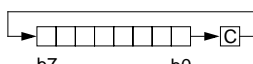
7.3.2 Relative Addressing Mode (REL)

Relative addressing mode is used to specify the destination location for branch instructions. A signed 8-bit offset value is located in the memory location immediately following the opcode. During execution, if the branch condition is true, the signed offset is sign-extended to a 16-bit value and is added to the current contents of the program counter, which causes program execution to continue at the branch destination address.

7.3.3 Immediate Addressing Mode (IMM)

In immediate addressing mode, the operand needed to complete the instruction is included in the object code immediately following the instruction opcode in memory. In the case of a 16-bit immediate operand,

Table 7-2. Instruction Set Summary (Sheet 6 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V I 1 H	I N Z C
MOV <i>opr8a,opr8a</i> MOV <i>opr8a,X+</i> MOV <i>#opr8i,opr8a</i> MOV <i>,X+,opr8a</i>	Move $(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ In IX+/DIR and DIR/IX+ Modes, $H:X \leftarrow (H:X) + \$0001$	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E dd dd 5E dd 6E ii dd 7E dd	5 5 4 5	rpwpp rfwpp pwpp rfwpp	0 1 1 -	- \uparrow \downarrow \uparrow -
MUL	Unsigned multiply $X:A \leftarrow (X) \times (A)$	INH	42	5	ffffp	- 1 1 0	- - - - 0
NEG <i>opr8a</i> NEGA NEGX NEG <i>opr8,X</i> NEG <i>,X</i> NEG <i>opr8,SP</i>	Negate Two's Complement $M \leftarrow -(M) = \$00 - (M)$ $A \leftarrow -(A) = \$00 - (A)$ $X \leftarrow -(X) = \$00 - (X)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$	DIR INH INH IX1 IX SP1	30 dd 40 50 60 ff 70 9E 60 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	\uparrow 1 1 -	- \uparrow \downarrow \uparrow \downarrow
NOP	No Operation — Uses 1 Bus Cycle	INH	9D	1	p	- 1 1 -	- - - - -
NSA	Nibble Swap Accumulator $A \leftarrow (A[3:0]:A[7:4])$	INH	62	1	p	- 1 1 -	- - - - -
ORA <i>#opr8i</i> ORA <i>opr8a</i> ORA <i>opr16a</i> ORA <i>opr8,X</i> ORA <i>opr8,X</i> ORA <i>,X</i> ORA <i>opr8,SP</i> ORA <i>opr8,SP</i>	Inclusive OR Accumulator and Memory $A \leftarrow (A) (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA ii BA dd CA hh ll DA ee ff EA ff FA 9E DA ee ff 9E EA ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 1 1 -	- \uparrow \downarrow \uparrow -
PSHA	Push Accumulator onto Stack Push (A); $SP \leftarrow (SP) - \$0001$	INH	87	2	sp	- 1 1 -	- - - - -
PSHH	Push H (Index Register High) onto Stack Push (H); $SP \leftarrow (SP) - \$0001$	INH	8B	2	sp	- 1 1 -	- - - - -
PSHX	Push X (Index Register Low) onto Stack Push (X); $SP \leftarrow (SP) - \$0001$	INH	89	2	sp	- 1 1 -	- - - - -
PULA	Pull Accumulator from Stack $SP \leftarrow (SP + \$0001)$; Pull (A)	INH	86	3	ufp	- 1 1 -	- - - - -
PULH	Pull H (Index Register High) from Stack $SP \leftarrow (SP + \$0001)$; Pull (H)	INH	8A	3	ufp	- 1 1 -	- - - - -
PULX	Pull X (Index Register Low) from Stack $SP \leftarrow (SP + \$0001)$; Pull (X)	INH	88	3	ufp	- 1 1 -	- - - - -
ROL <i>opr8a</i> ROLA ROLX ROL <i>opr8,X</i> ROL <i>,X</i> ROL <i>opr8,SP</i>	Rotate Left through Carry 	DIR INH INH IX1 IX SP1	39 dd 49 59 69 ff 79 9E 69 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	\uparrow 1 1 -	- \uparrow \downarrow \uparrow \downarrow
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR <i>,X</i> ROR <i>opr8,SP</i>	Rotate Right through Carry 	DIR INH INH IX1 IX SP1	36 dd 46 56 66 ff 76 9E 66 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	\uparrow 1 1 -	- \uparrow \downarrow \uparrow \downarrow

ADHTS, in the SOPT2 register. The RTC or IRQ can be configured to cause a hardware trigger in run, wait, and stop3 modes.

10.1.5 Temperature Sensor

To use the on-chip temperature sensor, the user must perform the following:

- Configure ADC for long sample with a maximum of 1 MHz clock
- Convert the bandgap voltage reference channel (AD27)
 - By converting the digital value of the bandgap voltage reference channel using the value of V_{BG} the user can determine V_{DD} . For value of bandgap voltage, see [Section A.6, “DC Characteristics”](#).
- Convert the temperature sensor channel (AD26)
 - By using the calculated value of V_{DD} , convert the digital value of AD26 into a voltage, V_{TEMP}

[Equation 10-1](#) provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \quad \text{Eqn. 10-1}$$

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP} and compares to V_{TEMP25} . If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in [Equation 10-1](#). If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in [Equation 10-1](#).

To improve accuracy the user should calibrate the bandgap voltage reference and temperature sensor.

Calibrating at 25°C will improve accuracy to $\pm 4.5^\circ\text{C}$.

Calibration at three points, -40°C, 25°C, and 125°C will improve accuracy to $\pm 2.5^\circ\text{C}$. Once calibration has been completed, the user will need to calculate the slope for both hot and cold. In application code, the user would then calculate the temperature using [Equation 10-1](#) as detailed above and then determine if the temperature is above or below 25°C. Once determined if the temperature is above or below 25°C, the user can recalculate the temperature using the hot or cold slope value obtained during calibration.

When a conversion is aborted, the contents of the data registers, ADCRH and ADCRL, are not altered. However, they continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset, ADCRH and ADCRL return to their reset states.

10.4.4.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Power consumption when active can be reduced by setting ADLPC. This results in a lower maximum value for f_{ADCK} (see the electrical specifications).

10.4.4.5 Sample Time and Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit, 10-bit or 12-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins. ADLSMP selects between short (3.5 ADCK cycles) and long (23.5 ADCK cycles) sample times. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0). If the bus frequency is less than 1/11th of the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADLSMP=1).

The maximum total conversion time for different conditions is summarized in [Table 10-13](#).

Table 10-13. Total Conversion Time vs. Control Conditions

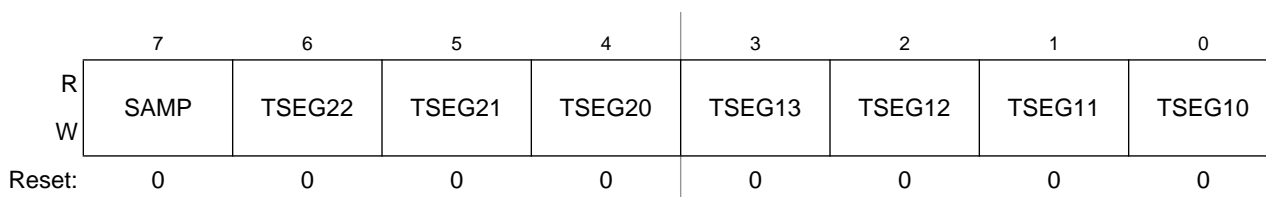
Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μ s + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 μ s + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μ s + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 μ s + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles

Table 12-5. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
1	1	1	1	1	1	64

12.3.4 MSCAN Bus Timing Register 1 (CANBTR1)

The CANBTR1 register configures various CAN bus timing parameters of the MSCAN module.


Figure 12-7. MSCAN Bus Timing Register 1 (CANBTR1)

Read: Anytime

Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 12-6. CANBTR1 Register Field Descriptions

Field	Description
7 SAMP	Sampling — This bit determines the number of CAN bus samples taken per bit time. 0 One sample per bit. 1 Three samples per bit ¹ . If SAMP = 0, the resulting bit value is equal to the value of the single bit positioned at the sample point. If SAMP = 1, the resulting bit value is determined by using majority rule on the three total samples. For higher bit rates, it is recommended that only one sample is taken per bit time (SAMP = 0).
6:4 TSEG2[2:0]	Time Segment 2 — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 12-43). Time segment 2 (TSEG2) values are programmable as shown in Table 12-7 .
3:0 TSEG1[3:0]	Time Segment 1 — Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point (see Figure 12-43). Time segment 1 (TSEG1) values are programmable as shown in Table 12-8 .

¹ In this case, PHASE_SEG1 must be at least 2 time quanta (Tq).

bit position in the filter register. Finally, registers CANIDAR0/1/2/3 determine the value of those bits determined by CANIDMR0/1/2/3.

For instance in the case of the filter value of:

0001x1001x0

The CANIDMR0/1/2/3 register would be configured as:

00001000010

and so all message identifier bits except bit 1 and bit 6 would be compared against the CANIDAR0/1/2/3 registers. These would be configured as:

00010100100

In this case bits 1 and 6 are set to '0', but since they are ignored it is equally valid to set them to '1'.

12.5.3.1 Identifier Acceptance Filters example

As described above, filters work by comparisons to individual bits in the CAN message identifier field. The filter will check each one of the eleven bits of a standard CAN message identifier. Suppose a filter value of 0001x1001x0. In this simple example, there are only three possible CAN messages.

Filter value: 0001x1001x0

Message 1: 00011100110

Message 2: 00110100110

Message 3: 00010100100

Message 2 will be rejected since its third most significant bit is not '0' - 001. The filter is simply a convenient way of defining the set of messages that the CPU must receive. For full 29-bits of an extended CAN message identifier, the filter identifies two sets of messages: one set that it receives and one set that it rejects. Alternatively, the filter may be split into two. This allows the MSCAN to examine only the first 16 bits of a message identifier, but allows two separate filters to perform the checking. See the example below:

Filter value A: 0001x1001x0

Filter value B: 00x101x01x0

Message 1: 00011100110

Message 2: 00110100110

Message 3: 00010100100

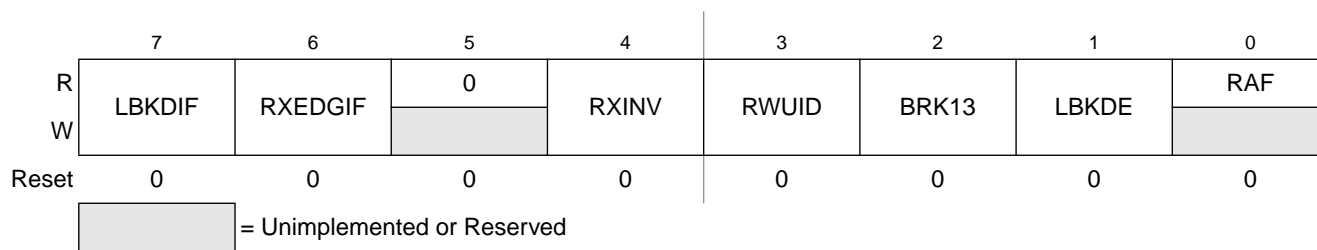
MSCAN will accept all three messages. Filter A will accept messages 1 and 3 as before and filter B will accept message 2. In practice, it is unimportant which filter accepts the message - messages accepted by either will be placed in the input buffer. A message may be accepted by more than one filter.

Table 14-6. SC1xS1 Field Descriptions (continued)

Field	Description
1 FE	Framing Error Flag — 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 SC1xS1 with FE = 1 and then read the SCI data register (SCIxD). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	Parity Error Flag — 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 SC1xS1 and then read the SCI data register (SCIxD). 0 No parity error. 1 Parity error.

14.2.5 SCI Status Register 2 (SC1xS2)

This register has one read-only status flag.


Figure 14-9. SCI Status Register 2 (SC1xS2)
Table 14-7. SC1xS2 Field Descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV ¹	Receive Data Inversion — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	Receive Wake Up Idle Detect — RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full ($RDRF = 1$), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to [Section 14.3.4, "Interrupts and Status Flags"](#) for more details about flag clearing.

14.3.3.1 Data Sampling Technique

The SCI receiver uses a $16\times$ baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The $16\times$ baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

14.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant

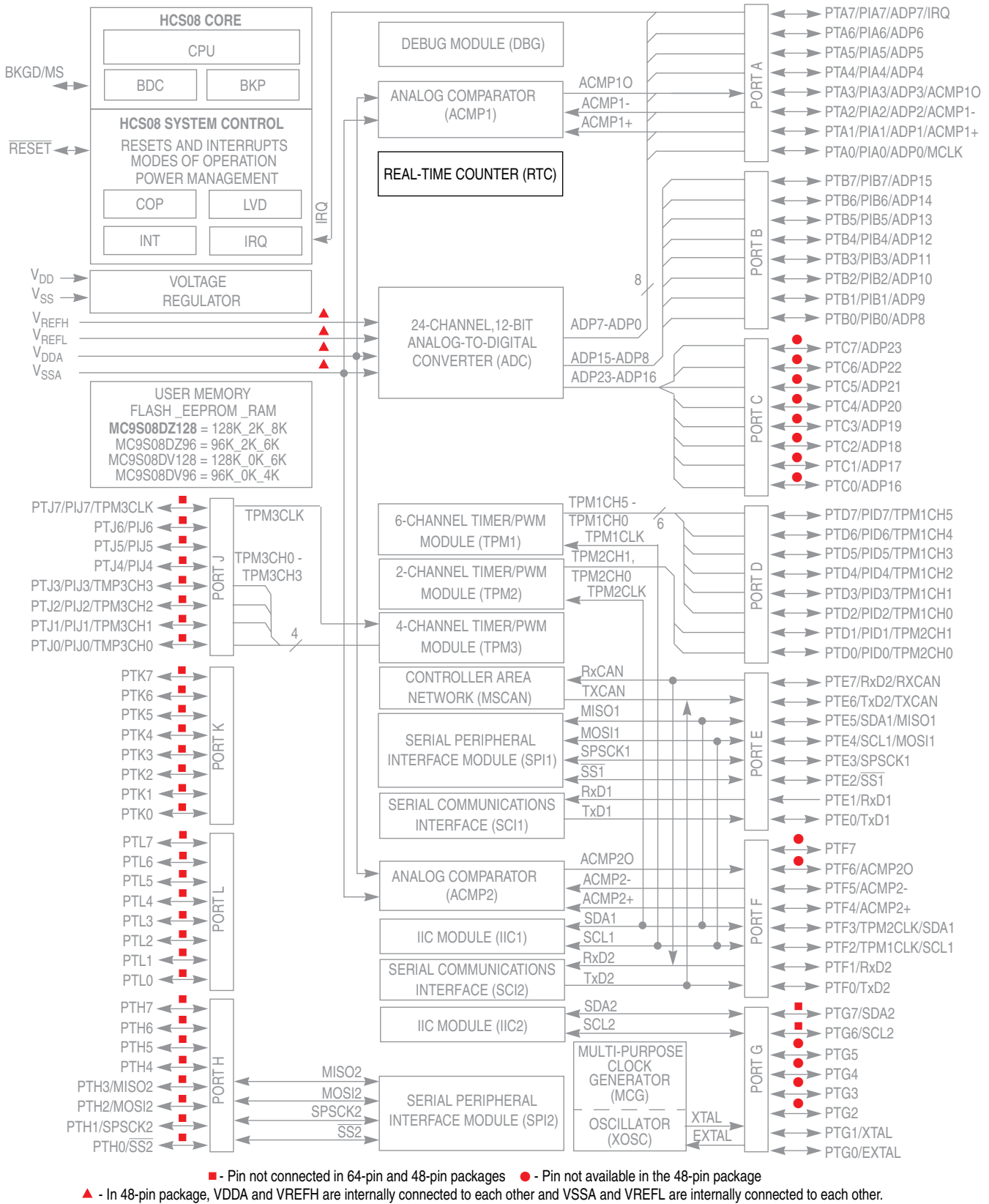


Figure 15-1. MC9S08DZ128 Block Diagram with RTC Highlighted

15.3.2 RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 8-bit counter.

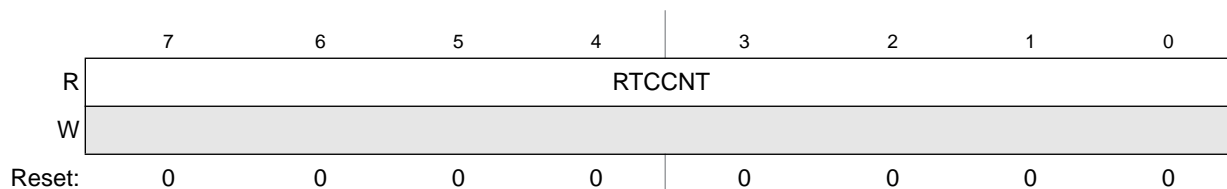


Figure 15-4. RTC Counter Register (RTCCNT)

Table 15-4. RTCCNT Field Descriptions

Field	Description
7:0 RTCCNT	RTC Count. These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset, writing to RTCMOD, or writing different values to RTCLKS and RTCPS clear the count to 0x00.

15.3.3 RTC Modulo Register (RTCMOD)

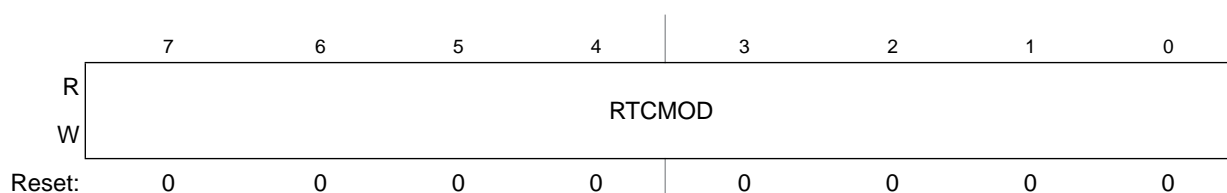


Figure 15-5. RTC Modulo Register (RTCMOD)

Table 15-5. RTCMOD Field Descriptions

Field	Description
7:0 RTCMOD	RTC Modulo. These eight read/write bits contain the modulo value used to reset the count to 0x00 upon a compare match and set the RTIF status bit. A value of 0x00 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00. Reset sets the modulo to 0x00.

15.4 Functional Description

The RTC is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with binary-based and decimal-based selectable values. The module also contains software selectable interrupt logic.

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1-kHz internal oscillator clock is selected as the default clock source. To start the prescaler, write any value other than zero to the prescaler select bits (RTCPS).

Three clock sources are software selectable: the low power oscillator clock (LPO), the external clock (ERCLK), and the internal clock (IRCLK). The RTC clock select bits (RTCLKS) select the desired clock source. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

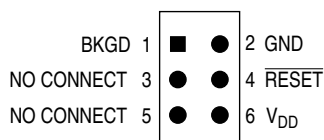


Figure 17-1. BDM Tool Connector

17.2.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to [Section 17.2.2, "Communication Details."](#)

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to [Section 17.2.2, "Communication Details,"](#) for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the HCS08 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

17.2.2 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

Chapter 18

Debug Module (S08DBGV3) (128K)

18.1 Introduction

The DBG module implements an on-chip ICE (in-circuit emulation) system and allows non-intrusive debug of application software by providing an on-chip trace buffer with flexible triggering capability. The trigger also can provide extended breakpoint capacity. The on-chip ICE system is optimized for the HCS08 8-bit architecture and supports 64K bytes or 128K bytes of memory space.

18.1.1 Features

The on-chip ICE system includes these distinctive features:

- Three comparators (A, B, and C) with ability to match addresses in 128K space
 - Dual mode, Comparators A and B used to compare addresses
 - Full mode, Comparator A compares address and Comparator B compares data
 - Can be used as triggers and/or breakpoints
 - Comparator C can be used as a normal hardware breakpoint
 - Loop1 capture mode, Comparator C is used to track most recent COF event captured into FIFO
- Tag and Force type breakpoints
- Nine trigger modes
 - A
 - A Or B
 - A Then B
 - A And B, where B is data (Full mode)
 - A And Not B, where B is data (Full mode)
 - Event Only B, store data
 - A Then Event Only B, store data
 - Inside Range, $A \leq \text{Address} \leq B$
 - Outside Range, $\text{Address} < A$ or $\text{Address} > B$
- FIFO for storing change of flow information and event only data
 - Source address of conditional branches taken
 - Destination address of indirect JMP and JSR instruction
 - Destination address of interrupts, RTI, RTC, and RTS instruction
 - Data associated with Event B trigger modes

¹ In the case of an end-trace to reset where DBGGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 18-6. DBGCBL Field Descriptions

Field	Description
Bits 7–0	<p>Comparator B Low Compare Bits — The Comparator B Low compare bits control whether Comparator B will compare the address bus or data bus bits [7:0] to a logic 1 or logic 0.</p> <p>0 Compare corresponding address bit to a logic 0, compares to data if in Full mode</p> <p>1 Compare corresponding address bit to a logic 1, compares to data if in Full mode</p>

18.3.3.5 Debug Comparator C High Register (DBGCCCH)

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R								
W								
	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run ¹	U	U	U	U	U	U	U	U

Figure 18-6. Debug Comparator C High Register (DBGCCCH)

¹ In the case of an end-trace to reset where DBGGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 18-7. DBGCCCH Field Descriptions

Field	Description
Bits 15–8	<p>Comparator C High Compare Bits — The Comparator C High compare bits control whether Comparator C will compare the address bus bits [15:8] to a logic 1 or logic 0.</p> <p>0 Compare corresponding address bit to a logic 0</p> <p>1 Compare corresponding address bit to a logic 1</p>

Table 18-9. DBGFH Field Descriptions

Field	Description
Bits 15–8	FIFO High Data Bits — The FIFO High data bits provide access to bits [15:8] of data in the FIFO. This register is not used in event only modes and will read a \$00 for valid FIFO words.

18.3.3.8 Debug FIFO Low Register (DBGFL)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W								
POR or non- end-run	0	0	0	0	0	0	0	0
Reset end-run ¹	U	U	U	U	U	U	U	U

= Unimplemented or Reserved

Figure 18-9. Debug FIFO Low Register (DBGFL)

¹ In the case of an end-trace to reset where DBGGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 18-10. DBGFL Field Descriptions

Field	Description
Bits 7–0	FIFO Low Data Bits — The FIFO Low data bits contain the least significant byte of data in the FIFO. When reading FIFO words, read DBGFX and DBGFH before reading DBGFL because reading DBGFL causes the FIFO pointers to advance to the next FIFO location. In event-only modes, there is no useful information in DBGFX and DBGFH so it is not necessary to read them before reading DBGFL.