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	I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	54
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 16x10b
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/mc9s08aw16cpue

Part Number	Package Description	Original (gold wire) package document number	Current (copper wire) package document number
MC68HC908JW32	48 QFN	98ARH99048A	98ASA00466D
MC9S08AC16			
MC9S908AC60			
MC9S08AC128			
MC9S08AW60			
MC9S08GB60A			
MC9S08GT16A			
MC9S08JM16			
MC9S08JM60			
MC9S08LL16			
MC9S08QE128			
MC9S08QE32			
MC9S08RG60			
MCF51CN128			
MC9RS08LA8	48 QFN	98ARL10606D	98ASA00466D
MC9S08GT16A	32 QFN	98ARH99035A	98ASA00473D
MC9S908QE32	32 QFN	98ARE10566D	98ASA00473D
MC9S908QE8	32 QFN	98ASA00071D	98ASA00736D
MC9S08JS16	24 QFN	98ARL10608D	98ASA00734D
MC9S08QB8			
MC9S08QG8	24 QFN	98ARL10605D	98ASA00474D
MC9S08SH8	24 QFN	98ARE10714D	98ASA00474D
MC9RS08KB12	24 QFN	98ASA00087D	98ASA00602D
MC9S08QG8	16 QFN	98ARE10614D	98ASA00671D
MC9RS08KB12	8 DFN	98ARL10557D	98ASA00672D
MC9S08QG8			
MC9RS08KA2	6 DFN	98ARL10602D	98ASA00735D

  - The output of the digitally-controlled oscillator (DCO) in the frequency-locked loop sub-module
  - Control bits inside the ICG determine which source is connected.
- FFE is a control signal generated inside the ICG. If the frequency of ICGOUT > 4 × the frequency of ICGERCLK, this signal is a logic 1 and the fixed-frequency clock will be ICGERCLK/2. Otherwise the fixed-frequency clock will be BUSCLK.
- ICGLCLK — Development tools can select this internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.
- ICGERCLK — External reference clock can be selected as the real-time interrupt clock source. Can also be used as the ALTCLK input to the ADC module.

2.3.1 Power (V_{DD}, 2 x V_{SS}, V_{DDAD}, V_{SSAD})

V_{DD} and V_{SS} are the primary power supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides regulated lower-voltage source to the CPU and other internal circuitry of the MCU.

Typically, application systems have two separate capacitors across the power pins. In this case, there should be a bulk electrolytic capacitor, such as a 10-µF tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1-µF ceramic bypass capacitor located as near to the paired V_{DD} and V_{SS} power pins as practical to suppress high-frequency noise. The MC9S08AW60 has a second V_{SS} pin. This pin should be connected to the system ground plane or to the primary V_{SS} pin through a low-impedance connection.

V_{DDAD} and V_{SSAD} are the analog power supply pins for the MCU. This voltage source supplies power to the ADC module. A 0.1-µF ceramic bypass capacitor should be located as near to the analog power pins as practical to suppress high-frequency noise.

2.3.2 Oscillator (XTAL, EXTAL)

Out of reset, the MCU uses an internally generated clock (self-clocked mode — f_{Self_reset}) equivalent to about 8-MHz crystal rate. This frequency source is used during reset startup and can be enabled as the clock source for stop recovery to avoid the need for a long crystal startup delay. This MCU also contains a trimmable internal clock generator (ICG) module that can be used to run the MCU. For more information on the ICG, see the Chapter 8, “Internal Clock Generator (S08ICGV4).”

The oscillator amplitude on XTAL and EXTAL is gain limited for low-power oscillation. Typically, these pins have a 1-V peak-to-peak signal. For noisy environments, the high gain output (HGO) bit can be set to enable rail-to-rail oscillation.

The oscillator in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator in either of two frequency ranges selected by the RANGE bit in the ICGC1 register. Rather than a crystal or ceramic resonator, an external oscillator can be connected to the EXTAL input pin.

Refer to Figure 2-4 for the following discussion. R_S (when used) and R_F should be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally should be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

R_F is used to provide a bias path to keep the EXTAL input in its linear range during crystal startup and its value is not generally critical. Typical systems use 1 MΩ to 10 MΩ. Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5-pF to 25-pF range and are chosen to match the requirements of a specific crystal or resonator. Be sure to take into account printed circuit board (PCB) capacitance and MCU pin capacitance when sizing C1 and C2. The crystal manufacturer typically specifies a load capacitance which is the series combination of C1 and C2 which are usually the same size. As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

Table 4-7. FLASH Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV5:DIV0 (Decimal)	f_{FCLK}	Program/Erase Timing Pulse (5 µs Min, 6.7 µs Max)
20 MHz	1	12	192.3 kHz	5.2 µs
10 MHz	0	49	200 kHz	5 µs
8 MHz	0	39	200 kHz	5 µs
4 MHz	0	19	200 kHz	5 µs
2 MHz	0	9	200 kHz	5 µs
1 MHz	0	4	200 kHz	5 µs
200 kHz	0	0	200 kHz	5 µs
150 kHz	0	0	150 kHz	6.7 µs

4.6.2 FLASH Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into FOPT. Bits 5 through 2 are not used and always read 0. This register may be read at any time, but writes have no meaning or effect. To change the value in this register, erase and reprogram the NVOPT location in FLASH memory as usual and then issue a new MCU reset.

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
W								
Reset	This register is loaded from nonvolatile location NVOPT during reset.							

= Unimplemented or Reserved

Figure 4-7. FLASH Options Register (FOPT)

Table 4-8. FOPT Register Field Descriptions

Field	Description
7 KEYEN	**Backdoor Key Mechanism Enable** — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to Section 4.5, “Security.” 0 No backdoor key access allowed. 1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset.
6 FNORED	**Vector Redirection Disable** — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 SEC0[1:0]	**Security State Code** — This 2-bit field determines the security state of the MCU as shown in Table 4-9. When the MCU is secure, the contents of RAM and FLASH memory cannot be accessed by instructions from any unsecured source including the background debug interface. For more detailed information about security, refer to Section 4.5, “Security.”

- Illegal opcode detect
- Background debug forced reset
- The reset pin ($\overline{\text{RESET}}$)
- Clock generator loss of lock and loss of clock reset

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 bus cycles where the internal bus frequency is half the ICG frequency. After the 34 bus 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 bus 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 5.9.4, “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 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 a logic 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.

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence obeys the same cycle-by-cycle sequence as the SWI instruction and consists of:

- Saving the CPU registers on the stack
- Setting the I bit in the CCR to mask further interrupts
- Fetching the interrupt vector for the highest-priority interrupt that is currently pending
- Filling the instruction queue with the first three bytes of program information starting from the address fetched from the interrupt vector locations

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is restored to 0 when the CCR is restored from the value stacked on entry to the ISR. In rare cases, the I bit may be cleared inside an ISR (after clearing the status flag that generated the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not recommended for anyone other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

The interrupt service routine ends with a return-from-interrupt (RTI) instruction which restores the CCR, A, X, and PC registers to their pre-interrupt values by reading the previously saved information off the stack.

NOTE

For compatibility with the M68HC08, the H register is not automatically saved and restored. It is good programming practice to push H onto the stack at the start of the interrupt service routine (ISR) and restore it immediately before the RTI that is used to return from the ISR.

When two or more interrupts are pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-1).

5.5.1 Interrupt Stack Frame

Figure 5-1 shows the contents and organization of a stack frame. Before the interrupt, the stack pointer (SP) points at the next available byte location on the stack. The current values of CPU registers are stored on the stack starting with the low-order byte of the program counter (PCL) and ending with the CCR. After stacking, the SP points at the next available location on the stack which is the address that is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

Table 5-1. Vector Summary

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lower	26 through 31	$FFC0/FFC1 through $FFCA/FFCB	Unused Vector Space (available for user program)				
	25	$FFCC/FFCD	Vrti	System control	RTIF	RTIE	Real-time interrupt
	24	$FFCE/FFCF	Viic1	IIC1	IICIF	IICIE	IIC1
	23	$FFD0/FFD1	Vadc1	ADC1	COCO	AIEN	ADC1
	22	$FFD2/FFD3	Vkeyboard 1	KBI1	KBF	KBIE	KBI1 pins
	21	$FFD4/FFD5	Vsci2tx	SCI2	TDRE TC	TIE TCIE	SCI2 transmit
	20	$FFD6/FFD7	Vsci2rx	SCI2	IDLE RDRF	ILIE RIE	SCI2 receive
	19	$FFD8/FFD9	Vsci2err	SCI2	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI2 error
	18	$FFDA/FFDB	Vsci1tx	SCI1	TDRE TC	TIE TCIE	SCI1 transmit
	17	$FFDC/FFDD	Vsci1rx	SCI1	IDLE RDRF	ILIE RIE	SCI1 receive
	16	$FFDE/FFDF	Vsci1err	SCI1	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI1 error
	15	$FFE0/FFE1	Vspi1	SPI1	SPIF MODF SPTEF	SPIE SPIE SPTIE	SPI1
	14	$FFE2/FFE3	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow
	13	$FFE4/FFE5	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1
	12	$FFE6/FFE7	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0
	11	$FFE8/FFE9	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow
	10	$FFEA/FFEB	Vtpm1ch5	TPM1	CH5F	CH5IE	TPM1 channel 5
	9	$FFEC/FFED	Vtpm1ch4	TPM1	CH4F	CH4IE	TPM1 channel 4
	8	$FFEE/FFEF	Vtpm1ch3	TPM1	CH3F	CH3IE	TPM1 channel 3
	7	$FFF0/FFF1	Vtpm1ch2	TPM1	CH2F	CH2IE	TPM1 channel 2
	6	$FFF2/FFF3	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1
	5	$FFF4/FFF5	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0
	4	$FFF6/FFF7	Vicg	ICG	ICGIF (LOLS/LOCS)	LOLRE/LOCRE	ICG
	3	$FFF8/FFF9	Vlvd	System control	LVDF	LVDIE	Low-voltage detect
	2	$FFFA/FFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	1	$FFFC/FFFD	Vswi	Core	SWI Instruction	—	Software interrupt
Higher	0	$FFFE/FFFF	Vreset	System control	COP LVD $\overline{\text{RESET}}$ pin Illegal opcode	COPE LVDRE — —	Watchdog timer Low-voltage detect External pin Illegal opcode

5.9.6 System Device Identification Register (SDIDH, SDIDL)

This read-only register is included so host development systems can identify the HCS08 derivative. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.

	7	6	5	4	3	2	1	0
R					ID11	ID10	ID9	ID8
W								
Reset	—	—	—	—	0	0	0	0

= Unimplemented or Reserved

Figure 5-7. System Device Identification Register — High (SDIDH)

Table 5-7. SDIDH Register Field Descriptions

Field	Description
7:4 Reserved	**Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.**
3:0 ID[11:8]	**Part Identification Number** — Each derivative in the HCS08 family has a unique identification number. The MC9S08AW60 Series is hard coded to the value $008. See also ID bits in Table 5-8.

	7	6	5	4	3	2	1	0
R	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
W								
Reset	0	0	0	0	1	0	0	0

= Unimplemented or Reserved

Figure 5-8. System Device Identification Register — Low (SDIDL)

Table 5-8. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	**Part Identification Number** — Each derivative in the HCS08 family has a unique identification number. The MC9S08AW60 Series is hard coded to the value $008. See also ID bits in Table 5-7.

6.7.5 Port C I/O Registers (PTCD and PTCDD)

Port C parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W		PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
Reset	0	0	0	0	0	0	0	0

Figure 6-19. Port C Data Register (PTCD)

Table 6-12. PTCD Register Field Descriptions

Field	Description
6:0 PTCD[6:0]	**Port C Data Register Bits** — For port C pins that are inputs, reads return the logic level on the pin. For port C 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 C pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTCD 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.

	7	6	5	4	3	2	1	0
R W		PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-20. Data Direction for Port C (PTCDD)

Table 6-13. PTCDD Register Field Descriptions

Field	Description
6:0 PTCDD[6:0]	**Data Direction for Port C Bits** — These read/write bits control the direction of port C pins and what is read for PTCD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port C bit n and PTCD reads return the contents of PTCDn.

6.7.9 Port E I/O Registers (PTED and PTEDD)

Port E parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
Reset	0	0	0	0	0	0	0	0

Figure 6-29. Port E Data Register (PTED)

Table 6-22. PTED Register Field Descriptions

Field	Description
7:0 PTED[7:0]	**Port E Data Register Bits** — 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 of 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.

	7	6	5	4	3	2	1	0
R W	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-30. Data Direction for Port E (PTEDD)

Table 6-23. PTEDD Register Field Descriptions

Field	Description
7:0 PTEDD[7:0]	**Data Direction for Port E Bits** — These read/write bits control the direction of port E pins and what is read for PTED reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port E bit n and PTED reads return the contents of PTEDn.

6.7.10 Port E Pin Control Registers (PTEPE, PTESE, PTEDS)

In addition to the I/O control, port E pins are controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-31. Internal Pullup Enable for Port E (PTEPE)

Table 6-24. PTEPE Register Field Descriptions

Field	Description
7:0 PTEPE[7:0]	**Internal Pullup Enable for Port E Bits**— Each of these control bits determines if the internal pullup device is enabled for the associated PTE pin. For port E pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port E bit n. 1 Internal pullup device enabled for port E bit n.

	7	6	5	4	3	2	1	0
R W	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
Reset	0	0	0	0	0	0	0	0

Figure 6-32. Output Slew Rate Control Enable for Port E (PTESE)

Table 6-25. PTESE Register Field Descriptions

Field	Description
7:0 PTESE[7:0]	**Output Slew Rate Control Enable for Port E Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTE pin. For port E pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port E bit n. 1 Output slew rate control enabled for port E bit n.

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

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
BCLR *n,opr8a*	Clear Bit n in Memory	Mn ← 0	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	Branch if (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *rel*	Branch if Greater Than or Equal To (Signed Operands)	Branch if (N ⊕ V) = 0	–	–	–	–	–	–	REL	90	rr	3
BGND	Enter Active Background if ENBDM = 1	Waits For and Processes BDM Commands Until GO, TRACE1, or TAGGO	–	–	–	–	–	–	INH	82		5+
BGT *rel*	Branch if Greater Than (Signed Operands)	Branch if (Z) ǀ (N ⊕ V) = 0	–	–	–	–	–	–	REL	92	rr	3
BHCC *rel*	Branch if Half Carry Bit Clear	Branch if (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS *rel*	Branch if Half Carry Bit Set	Branch if (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI *rel*	Branch if Higher	Branch if (C) ǀ (Z) = 0	–	–	–	–	–	–	REL	22	rr	3
BHS *rel*	Branch if Higher or Same (Same as BCC)	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if IRQ Pin High	Branch if IRQ pin = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if IRQ Pin Low	Branch if IRQ pin = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr8i* BIT *opr8a* BIT *opr16a* BIT *oprx16*,X BIT *oprx8*,X BIT ,X BIT *oprx16*,SP BIT *oprx8*,SP	Bit Test	(A) & (M) (CCR Updated but Operands Not Changed)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
BLE *rel*	Branch if Less Than or Equal To (Signed Operands)	Branch if (Z) ǀ (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	Branch if (C) ǀ (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *rel*	Branch if Less Than (Signed Operands)	Branch if (N ⊕ V) = 1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	Branch if (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	Branch if (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	Branch if (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	Branch if (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	Branch if (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	No Test	–	–	–	–	–	–	REL	20	rr	3

ICGTRM = $xx

Bit 7:0 TRIM — Only need to write when trimming internal oscillator; done in separate operation (see example #4)

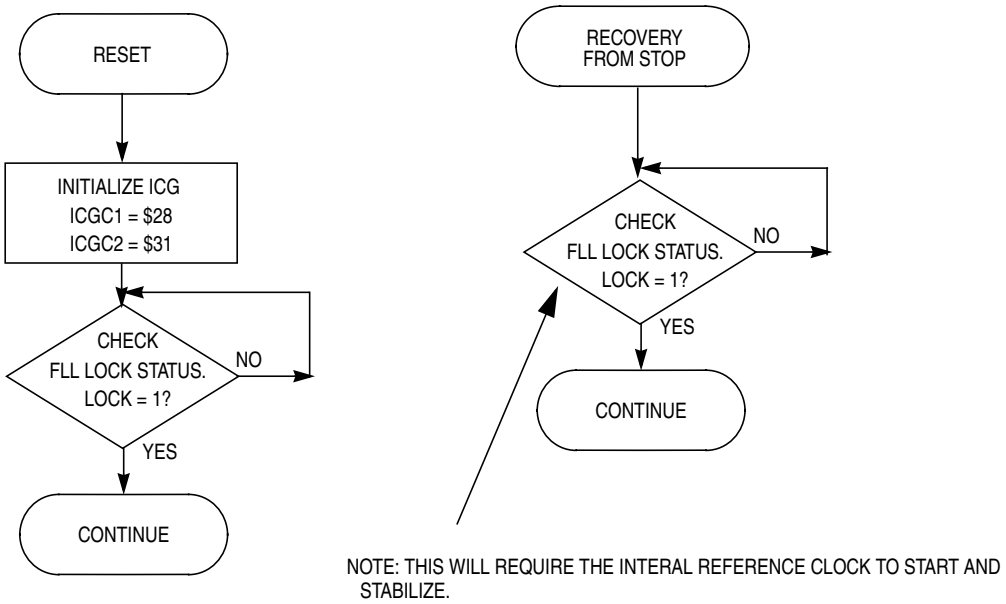

Figure 8-16. ICG Initialization and Stop Recovery for Example #3

Table 10-5. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM channel; use as an external clock for the TPM or revert to general-purpose I/O	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	00	Output compare	Software compare only
		01		Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger. Typically, a program would clear status flags after changing channel configuration bits and before enabling channel interrupts or using the status flags to avoid any unexpected behavior.

10.4.5 Timer x Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel value registers are cleared by reset.

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 10-9. Timer x Channel Value Register High (TPMxCnVH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 10-10. Timer Channel Value Register Low (TPMxCnVL)

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

11.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes.

No SCI module registers are affected in stop3 mode.

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

11.3.5.3 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

11.3.5.4 Single-Wire Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter.

14.5.4.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADC1RH and ADC1RL. This is indicated by the setting of COCO. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADC1RH and ADC1RL if the previous data is in the process of being read while in 10-bit MODE (the ADC1RH register has been read but the ADC1RL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

14.5.4.3 Aborting Conversions

Any conversion in progress will be aborted when:

- A write to ADC1SC1 occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADC1SC2, ADC1CFG, ADC1CVH, or ADC1CVL occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset.
- The MCU enters stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADC1RH and ADC1RL, are not altered but continue to be the values transferred after the completion of the last successful conversion. In the case that the conversion was aborted by a reset, ADC1RH and ADC1RL return to their reset states.

14.5.4.4 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).

14.5.4.5 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 or 10-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins. ADLSMP is used to select between short and long 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 ADC1RH and ADC1RL 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 14-12.

Table 14-12. 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	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	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 µs + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	0	5 µs + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 µs + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	1	5 µs + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-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
Subsequent continuous 10-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits. For example, in 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, then the conversion time for a single conversion is:

$$\text{Conversion time} = \frac{23 \text{ ADCK cyc}}{8 \text{ MHz}/1} + \frac{5 \text{ bus cyc}}{8 \text{ MHz}} = 3.5 \text{ µs}$$

$$\text{Number of bus cycles} = 3.5 \text{ µs} \times 8 \text{ MHz} = 28 \text{ cycles}$$

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

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.

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.

15.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R W	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
Reset	0	0	0	0	0	0	0	0

Figure 15-7. Debug Control Register (DBGC)

Table 15-4. DBGC Register Field Descriptions

Field	Description
7 DBGEN	**Debug Module Enable** — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure. 0 DBG disabled 1 DBG enabled
6 ARM	**Arm Control** — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN. 0 Debugger not armed 1 Debugger armed
5 TAG	**Tag/Force Select** — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect. 0 CPU breaks requested as force type requests 1 CPU breaks requested as tag type requests
4 BRKEN	**Break Enable** — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests. 0 CPU break requests not enabled 1 Triggers cause a break request to the CPU
3 RWA	**R/W Comparison Value for Comparator A** — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A. 0 Comparator A can only match on a write cycle 1 Comparator A can only match on a read cycle
2 RWAEN	**Enable R/W for Comparator A** — Controls whether the level of R/W is considered for a comparator A match. 0 R/W is not used in comparison A 1 R/W is used in comparison A
1 RWB	**R/W Comparison Value for Comparator B** — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B. 0 Comparator B can match only on a write cycle 1 Comparator B can match only on a read cycle
0 RWBEN	**Enable R/W for Comparator B** — Controls whether the level of R/W is considered for a comparator B match. 0 R/W is not used in comparison B 1 R/W is used in comparison B