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	16MHz
Connectivity	-
Peripherals	LVD, POR, PWM, WDT
Number of I/O	4
Program Memory Size	2KB (2K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	8-DIP (0.300", 7.62mm)
Supplier Device Package	8-PDIP
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s08qd2mpc

Chapter 9
Internal Clock Source (S08ICSV1)

After entering active background mode, the CPU is held in a suspended state waiting for serial background commands rather than executing instructions from the user's application program.

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 when 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 only be executed 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 MC9S08QD4 series devices are shipped from the Freescale Semiconductor factory, the flash program memory is erased by default unless specifically noted, so no program can 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 12, "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 two stop modes is entered upon execution of a STOP instruction when the STOPE bit in the system option register is set. In both stop modes, all internal clocks are halted. If the STOPE bit is not set when

program command is issued, the charge pump is enabled and then remains enabled after completion of the burst program operation if these two conditions are met:

- The next burst program command has been queued before the current program operation has completed.
- The next sequential address selects a byte on the same physical row as the current byte being programmed. A row of flash memory consists of 64 bytes. A byte within a row is selected by addresses A5 through A0. A new row begins when addresses A5 through A0 are all 0s.

The first byte of a series of sequential bytes being programmed in burst mode will take the same amount of time to program as a byte programmed in standard mode. Subsequent bytes will program in the burst 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.

- Writing a second time to a flash address before launching the previous command (There is only one write to flash for every command.)
- Writing a second time to FCMD before launching the previous command (There is only one write to FCMD for every command.)
- Writing to any flash control register other than FCMD after writing to a flash address
- Writing any command code other than the five allowed codes (0x05, 0x20, 0x25, 0x40, or 0x41) to FCMD
- Writing any flash control register other than to write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD.
- The MCU enters stop mode while a program or erase command is in progress (The command is aborted.)
- Writing the byte program, burst program, or page erase command code (0x20, 0x25, or 0x40) with a background debug command while the MCU is secured (The background debug controller can do blank check and mass erase commands only when the MCU is secure.)
- Writing 0 to FCBEF to cancel a partial command

4.5.6 Flash Block Protection

The block protection feature prevents the protected region of flash from program or erase changes. Block protection is controlled through the flash protection register (FPROT). When enabled, block protection begins at any 512 byte boundary below the last address of flash, 0xFFFF. (see Section 4.7.4, “Flash 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 512 bytes 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 allows a way to erase and reprogram a protected flash memory.

The block protection mechanism is illustrated in Figure 4-4. The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits as shown. For example, in order to protect the last 8192 bytes of memory (addresses 0xE000 through 0xFFFF), the FPS bits must be set to 1101 111, which results in the value 0xDFFF as the last address of unprotected memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT) must be programmed to logic 0 to enable block protection. Therefore the value 0xDE must be programmed into NVPROT to protect addresses 0xE000 through 0xFFFF.

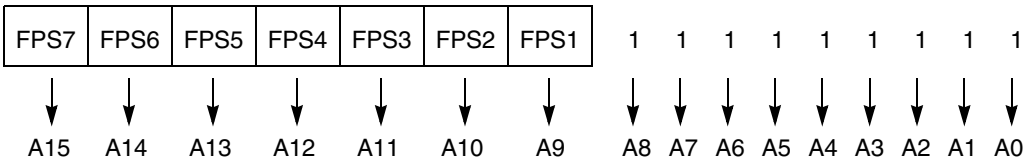

Figure 4-4. Block Protection Mechanism

is no way to disengage security without completely erasing all flash locations. If KEYEN is 1, a secure user program can temporarily disengage security by:

1. Writing 1 to KEYACC in the FCNFG register. This makes the flash module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a flash program or erase command.
2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be performed in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX must not be used for these writes because these writes cannot be performed on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was written matches the key stored in the flash locations, SEC01:SEC00 are automatically changed to 1:0 and security will be disengaged until the next reset.

The security key can be written only from secure memory (either RAM or flash), so it cannot be entered through background commands without the cooperation of a secure user program.

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in flash memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other flash memory location. The nonvolatile registers are in the same 512-byte block of flash as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key. Block protects cannot be changed from user application programs, so if the vector space is block protected, the backdoor security key mechanism cannot permanently change the block protect, security settings, or the backdoor key.

Security can always be disengaged through the background debug interface by taking these steps:

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase flash if necessary.
3. Blank check flash. Provided flash is completely erased, security is disengaged until the next reset.

To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

4.7 Flash Registers and Control Bits

The flash module has nine 8-bit registers in the high-page register space, two locations (NVOPT, NVPROT) in the nonvolatile register space in flash memory are copied into corresponding high-page control registers (FOPT, FPROT) at reset. There is also an 8-byte comparison key in flash memory. Refer to Table 4-3 and Table 4-4 for the absolute address assignments for all flash registers. This section refers to registers and control bits only by their names. A Freescale Semiconductor-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

4.7.1 Flash Clock Divider Register (FCDIV)

Before any erase or programming operations are possible, write to this register to set the frequency of the clock for the nonvolatile memory system within acceptable limits.

4.7.2 Flash Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from flash into FOPT. 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 4-6. 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 4.6, “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. SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash. For more detailed information about security, refer to Section 4.6, “Security.”

Table 4-9. Security States[1]

SEC01:SEC00	Description
0:0	secure
0:1	secure
1:0	unsecured
1:1	secure

[1] SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash.

4.7.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in Table 4-13. Refer to Section 4.5.3, "Program and Erase Command Execution," for a detailed discussion of flash programming and erase operations.

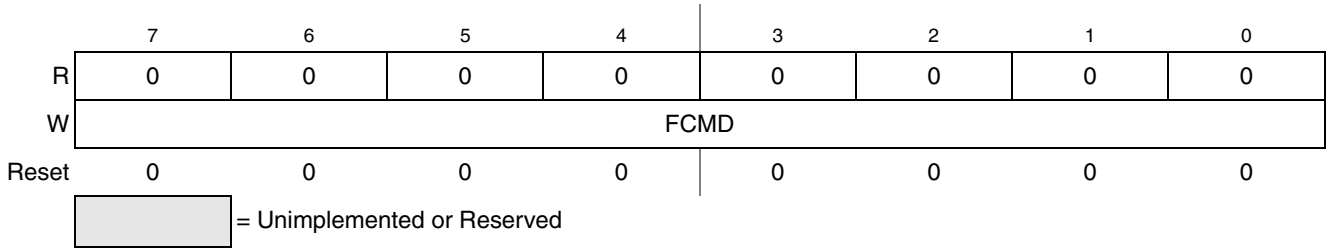

Figure 4-10. Flash Command Register (FCMD)

Table 4-13. Flash Commands

Command	FCMD	Equate File Label
Blank check	0x05	mBlank
Byte program	0x20	mByteProg
Byte program — burst mode	0x25	mBurstProg
Page erase (512 bytes/page)	0x40	mPageErase
Mass erase (all flash)	0x41	mMassErase

All other command codes are illegal and generate an access error.

It is not necessary to perform a blank check command after a mass erase operation. Only blank check is required as part of the security unlocking mechanism.

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 was 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 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 can 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 M68HC08 devices, 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 5-2).

Table 6-1. PTAD Register Field Descriptions

Field	Description
5:0 PTAD[5:0]	**Port A Data Register Bits** — For port A pins that are inputs, reads return the logic level on the pin. For port A 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 A pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTAD 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.

6.4.1.2 Port A Data Direction (PTADD)

	7	6	5	4	3	2	1	0
R	0	0	PTADD5[1]	PTADD4[2]	PTADD3	PTADD2	PTADD1	PTADD0
W								
Reset:	0	0	0	0	0	0	0	0

1 PTADD5 has no effect on the input-only PTA5 pin. Read this bit is always equal to zero.

2 PTADD4 has no effect on the output-only PTA4 pin.

Figure 6-3. Port A Data Direction Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
5:0 PTADD[5:0]	**Data Direction for Port A Bits** — These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit n and PTAD reads return the contents of PTADn.

6.4.2 Port A Control Registers

The pins associated with port A are controlled by the registers in this section. These registers control the pin pullup, slew rate and drive strength of the Port A pins independent of the parallel I/O register.

6.4.2.3 Port A Drive Strength Select (PTADS)

An output pin can be selected to have high output drive strength by setting the corresponding bit in the drive strength select register (PTADSn). When high drive is selected a pin is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, the user must ensure that the total current source and sink limits for the chip are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this the EMC emissions may be affected by enabling pins as high drive.

6.4.2.4 Port A Drive Strength Select (PTADS)

	7	6	5	4	3	2	1	0
R	0	0	PTADS5[1]	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
W								
Reset:	0	0	0	0	0	0	0	0

1 PTADS5 has no effect on the input-only PTA5 pin.

Figure 6-6. Drive Strength Selection for Port A Register (PTADS)

Table 6-5. PTADS Register Field Descriptions

Field	Description
5:0 PTADS[5:0]	**Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output drive for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port A bit n. 1 High output drive strength selected for port A bit n.

Table 7-2. Instruction Set Summary (Sheet 4 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C

CMP #*opr8i*
CMP *opr8a*
CMP *opr16a*
CMP *oprx16*,X
CMP *oprx8*,X
CMP ,X
CMP *oprx16*,SP
CMP *oprx8*,SP
	Compare Accumulator with Memory
	A – M
	(CCR Updated But Operands Not Changed)
		IMM
		DIR
		EXT
		IX2
		IX1
		IX
		SP2
		SP1
			A1 ii
			B1 dd
			C1 hh ll
			D1 ee ff
			E1 ff
			F1
			9E D1 ee ff
			9E E1 ff
				2
				3
				4
				4
				3
				3
				5
				4
					pp
					rpp
					prpp
					prpp
					rpp
					rfp
					pprpp
					prpp
						↕ –	– ↕ ↕ ↕

COM *opr8a* COMA COMX COM *oprx8*,X COM ,X COM *oprx8*,SP	Complement (One's Complement) M ← ($\overline{M}$)= $FF – (M) A ← ($\overline{A}$) = $FF – (A) X ← ($\overline{X}$) = $FF – (X) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	DIR INH INH IX1 IX SP1	33 dd 43 53 63 ff 73 9E 63 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	0 –	– ↕ ↕ 1
CPHX *opr16a* CPHX #*opr16i* CPHX *opr8a* CPHX *oprx8*,SP	Compare Index Register (H:X) with Memory (H:X) – (M:M + $0001) (CCR Updated But Operands Not Changed)	EXT IMM DIR SP1	3E hh ll 65 jj kk 75 dd 9E F3 ff	6 3 5 6	prrfpp ppp rrfpp prrfpp	↕ –	– ↕ ↕ ↕
CPX #*opr8i* CPX *opr8a* CPX *opr16a* CPX *oprx16*,X CPX *oprx8*,X CPX ,X CPX *oprx16*,SP CPX *oprx8*,SP	Compare X (Index Register Low) with Memory X – M (CCR Updated But Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 ii B3 dd C3 hh ll D3 ee ff E3 ff F3 9E D3 ee ff 9E E3 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↕ –	– ↕ ↕ ↕
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	INH	72	1	p	U–	– ↕ ↕ ↕
DBNZ *opr8a*,*rel* DBNZA *rel* DBNZX *rel* DBNZ *oprx8*,X,*rel* DBNZ ,X,*rel* DBNZ *oprx8*,SP,*rel*	Decrement A, X, or M and Branch if Not Zero (if (result) ≠ 0) DBNZX Affects X Not H	DIR INH INH IX1 IX SP1	3B dd rr 4B rr 5B rr 6B ff rr 7B rr 9E 6B ff rr	7 4 4 7 6 8	rfwpppp fppp fppp rfwpppp rfwppp prfwpppp	– –	– – – –
DEC *opr8a* DECA DECX DEC *oprx8*,X DEC ,X DEC *oprx8*,SP	Decrement M ← (M) – $01 A ← (A) – $01 X ← (X) – $01 M ← (M) – $01 M ← (M) – $01 M ← (M) – $01	DIR INH INH IX1 IX SP1	3A dd 4A 5A 6A ff 7A 9E 6A ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ –
DIV	Divide A ← (H:A)÷(X); H ← Remainder	INH	52	6	fffffp	– –	– – ↕ ↕
EOR #*opr8i* EOR *opr8a* EOR *opr16a* EOR *oprx16*,X EOR *oprx8*,X EOR ,X EOR *oprx16*,SP EOR *oprx8*,SP	Exclusive OR Memory with Accumulator A ← (A ⊕ M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 ii B8 dd C8 hh ll D8 ee ff E8 ff F8 9E D8 ee ff 9E E8 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 –	– ↕ ↕ –

8.1.2 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 10 bits resolution.
- Up to 28 analog inputs.
- Output formatted in 10- or 8-bit right-justified format.
- Single or continuous conversion (automatic return to idle after single conversion).
- Configurable sample time and conversion speed/power.
- Conversion complete flag and interrupt.
- Input clock selectable from up to four sources.
- Operation in wait or stop3 modes for lower noise operation.
- Asynchronous clock source for lower noise operation.
- Selectable asynchronous hardware conversion trigger.
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value.

8.1.3 Block Diagram

Figure 8-2 provides a block diagram of the ADC module

are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

8.4.2 Input Select and Pin Control

The pin control registers (APCTL3, APCTL2, and APCTL1) are used to disable the I/O port control of the pins used as analog inputs.When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

8.4.3 Hardware Trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

8.4.4 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

8.4.4.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

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 8-12.

Table 8-12. 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.

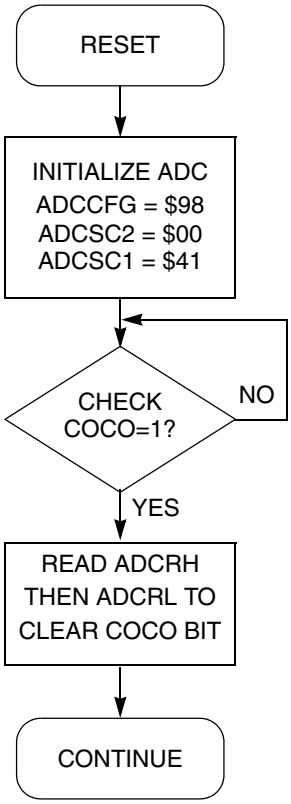

Figure 8-14. Initialization Flowchart for Example

8.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

8.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they must be used for best results.

8.6.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies (V_{DDAD} and V_{SSAD}) which are available as separate pins on some devices. On other devices, V_{SSAD} is shared on the same pin as the MCU digital V_{SS}, and on others, both V_{SSAD} and V_{DDAD} are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies which are bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both V_{DDAD} and V_{SSAD} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

9.1.3.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is supplied from the FLL.

9.1.3.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is not available.

9.1.3.7 Stop (STOP)

In stop mode the FLL is disabled and the internal or external reference clocks can be selected to be enabled or disabled. The BDC clock is not available and the ICS does not provide an MCU clock source.

9.1.4 Block Diagram

Figure 9-2 is the ICS block diagram.

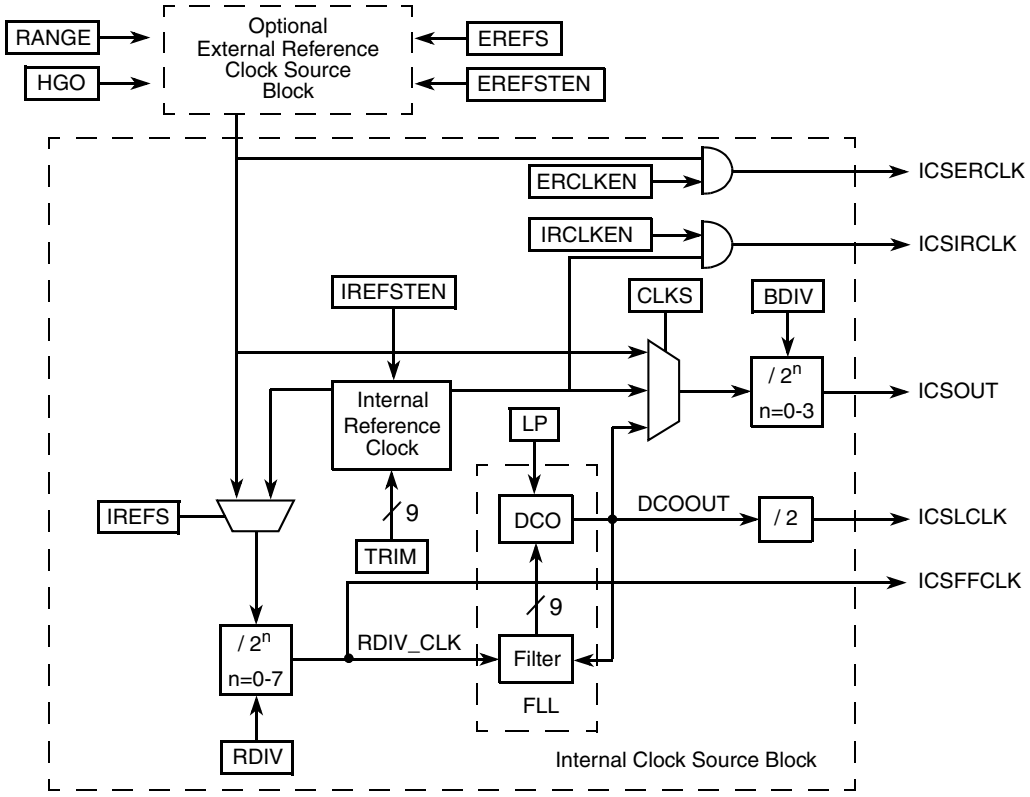

Figure 9-2. Internal Clock Source (ICS) Block Diagram

10.1.1 Features

The KBI features include:

- Up to eight keyboard interrupt pins with individual pin enable bits.
- Each keyboard interrupt pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity.
- One software enabled keyboard interrupt.
- Exit from low-power modes.

10.1.2 Modes of Operation

This section defines the KBI operation in wait, stop, and background debug modes.

10.1.2.1 KBI in Wait Mode

The KBI continues to operate in wait mode if enabled before executing the WAIT instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of wait mode if the KBI interrupt is enabled (KBIE = 1).

10.1.2.2 KBI in Stop Modes

The KBI operates asynchronously in stop3 mode if enabled before executing the STOP instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of stop3 mode if the KBI interrupt is enabled (KBIE = 1).

During either stop1 or stop2 mode, the KBI is disabled. In some systems, the pins associated with the KBI may be sources of wakeup from stop1 or stop2, see the stop modes section in the Modes of Operation chapter. Upon wake-up from stop1 or stop2 mode, the KBI module will be in the reset state.

10.1.2.3 KBI in Active Background Mode

When the microcontroller is in active background mode, the KBI will continue to operate normally.

10.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 10-2.

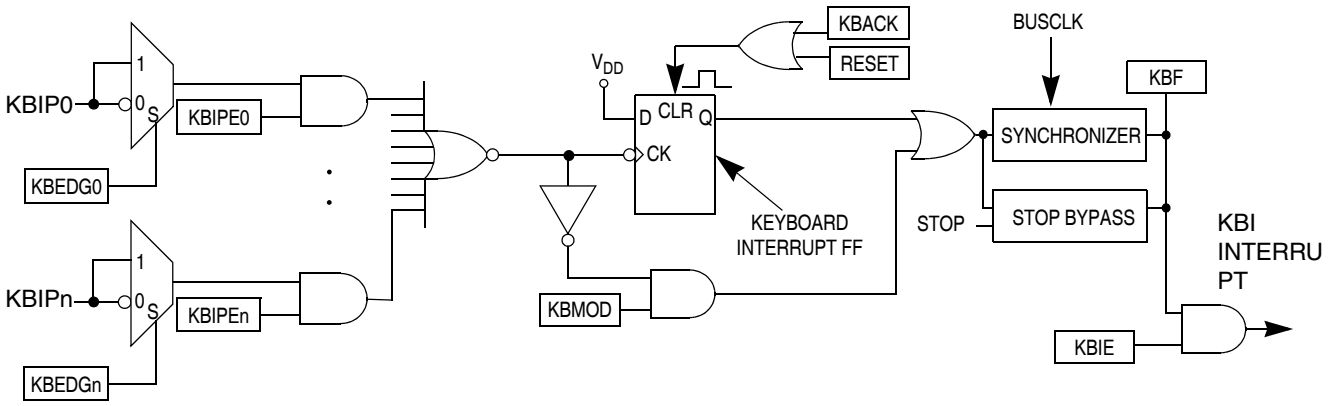

Figure 10-2. KBI Block Diagram

10.2 External Signal Description

The KBI input pins can be used to detect either falling edges, or both falling edge and low level interrupt requests. The KBI input pins can also be used to detect either rising edges, or both rising edge and high level interrupt requests.

The signal properties of KBI are shown in Table 10-1.

Table 10-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

10.3 Register Definition

The KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

Refer to the direct-page register summary in the Memory chapter for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

Some MCUs may have more than one KBI, so register names include placeholder characters to identify which KBI is being referenced.

10.3.1 KBI Status and Control Register (KBISC)

KBISC contains the status flag and control bits, which are used to configure the KBI.

Chapter 11
Timer/Pulse-Width Modulator (S08TPMV2)

11.1 Introduction

Figure 11-1 shows the MC9S08QD4 series block diagram with the TPM module and pins highlighted.

11.1.1 TPM2 Configuration Information

The TPM2 module consist of a single channel, TPM2CH0, that is multiplexed with input pin PTA4 and output pin PTA5. When TPM2 is configured for input capture, the TPM2CH0 will connect to the PTA5 (TPM2CH0I). When TPM2 is configured for output compare, the TPM2CH0 will connect to the PTA4 (TPM2CH0O). When TPM2 is disabled, PTA4 and PTA5 function as standard port pins.

11.1.2 TCLK1 and TCLK2 Configuration Information

The TCLK1 and TCLK2 are the external clock source inputs for TPM1 and TPM2 respectively.

Table A-10. ADC Characteristics (continued)

Characteristic	Conditions	Symb	Min	Typ[1]	Max	Unit	Comment
Zero-Scale Error	10 bit mode	E_{ZS}	0	±1.5	±3.1	LSB	$V_{ADIN} = V_{SSA}$
	8 bit mode		0	±0.5	±0.7		
Full-Scale Error	10 bit mode	E_{FS}	0	±1.0	±1.5	LSB	$V_{ADIN} = V_{DDA}$
	8 bit mode		0	±0.5	±0.5		
Quantization Error	10 bit mode	E_Q	—	—	±0.5	LSB	8 bit mode is not truncated

1 Typical values assume V_{DDAD} = 5.0 V, Temp = 25°C, f_{ADCK}=1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

2 At 4 MHz, for maximum frequency, use proportionally lower source impedance.

A.10 Flash Specifications

This section provides details about program/erase times and program-erase endurance for the flash memory.

Program and erase operations do not require any special power sources other than the normal V_{DD} supply. For more detailed information about program/erase operations, see the Memory section.

Table A-11. Flash Characteristics

Characteristic	Symbol	Min	Typical	Max	Unit
Supply voltage for program/erase –40°C to 125°C	$V_{prog/erase}$	2.7		5.5	V
Supply voltage for read operation	V_{Read}	2.7		5.5	V
Internal FCLK frequency[1]	f_{FCLK}	150		200	kHz
Internal FCLK period (1/FCLK)	t_{Fcyc}	5		6.67	µs
Byte program time (random location)(2)	t_{prog}	9			t_{Fcyc}
Byte program time (burst mode)(2)	t_{Burst}	4			t_{Fcyc}
Page erase time[2]	t_{Page}	4000			t_{Fcyc}
Mass erase time(2)	t_{Mass}	20,000			t_{Fcyc}
Program/erase endurance[3] T_L to T_H = –40°C to + 125°C T = 25°C		10,000	— 100,000	— —	cycles
Data retention[4]	t_{D_ret}	15	100	—	years

1 The frequency of this clock is controlled by a software setting.

2 These values are hardware state machine controlled. User code does not need to count cycles. This information supplied for calculating approximate time to program and erase.

3 **Typical endurance for flash** was evaluated for this product family on the 9S12Dx64. For additional information on how Freescale defines typical endurance, please refer to Engineering Bulletin EB619/D, *Typical Endurance for Nonvolatile Memory.*

4 **Typical data retention** values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale defines typical data retention, please refer to Engineering Bulletin EB618/D, *Typical Data Retention for Nonvolatile Memory.*