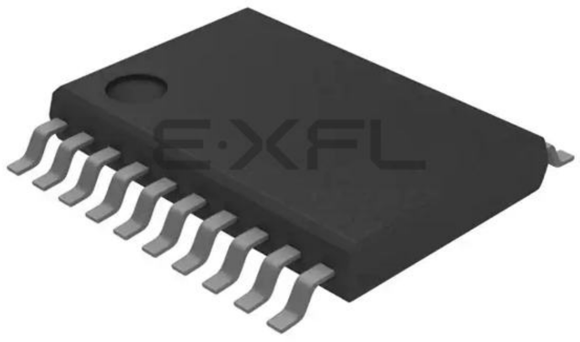
NXP USA Inc. - S9S08EL32F1MTJR Datasheet



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, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	16
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/s9s08el32f1mtjr

MC9S08EL32 Data Sheet

Covers MC9S08EL32
MC9S08EL16
MC9S08SL16
MC9S08SL8

MC9S08EL32
Rev. 3
7/2008

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/MS 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 application program (GO)

The active background mode is used to program a bootloader or user application program into the FLASH program memory before the MCU is operated in run mode for the first time. When the MC9S08EL32 Series and MC9S08SL16 Series is shipped from the Freescale Semiconductor factory, the FLASH program memory is erased by default unless specifically noted so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The active background mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the active background mode, refer to the Development Support chapter.

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 SOPT1 register is set. In both stop modes, all internal clocks are halted. The ICS module can be configured to leave the reference clocks running. See Chapter 8, “Internal Clock Source (S08ICSV2),” for more information.

Chapter 4
Memory

4.1 MC9S08EL32 Series and MC9S08SL16 Series Memory Map

As shown in Figure 4-1, on-chip memory in the MC9S08EL32 Series and MC9S08SL16 Series consists of RAM, EEPROM, and FLASH program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers (0x0000 through 0x007F)
- High-page registers (0x1800 through 0x18FF)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

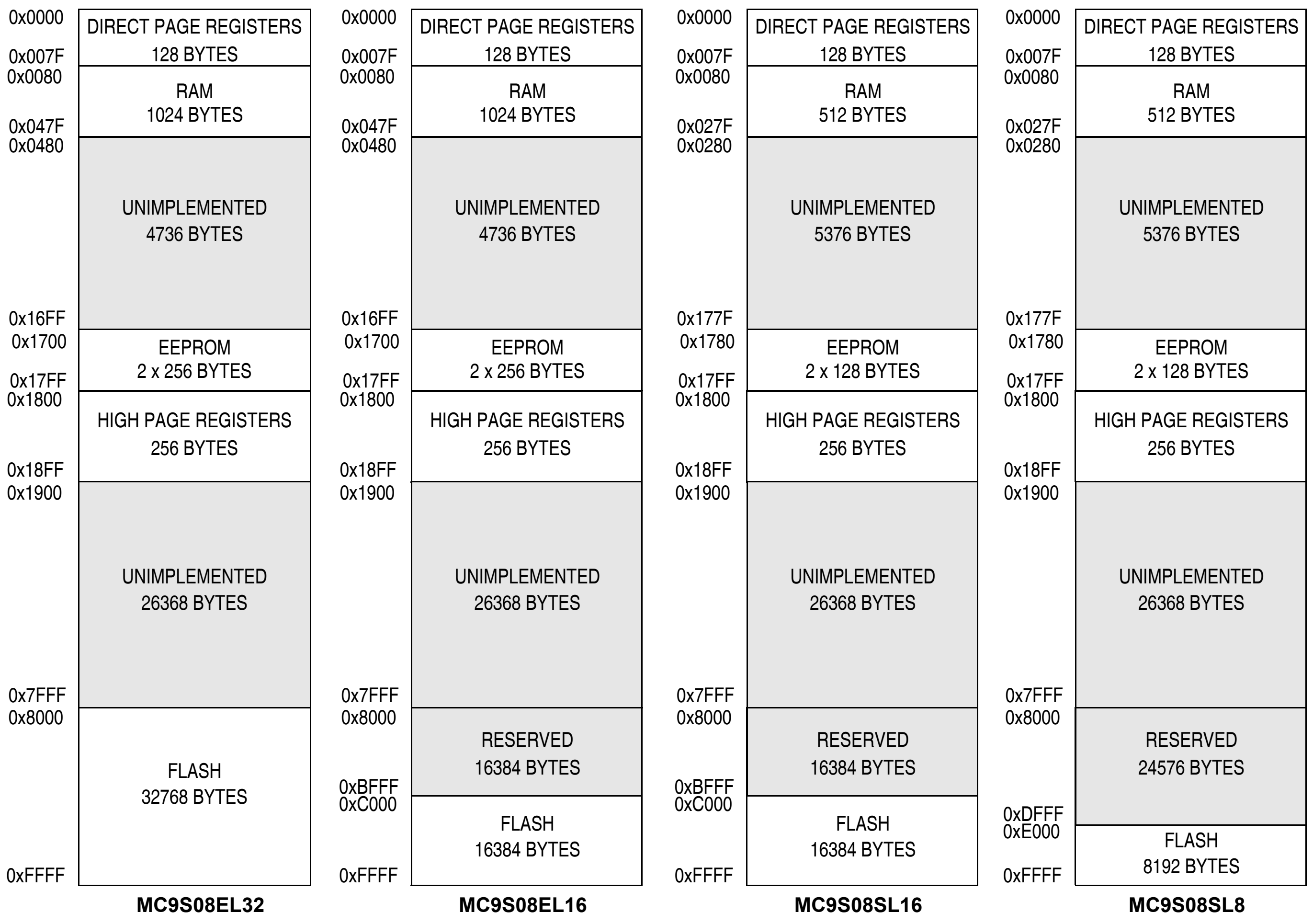


Figure 4-1. MC9S08EL32 Series and MC9S08SL16 Series Memory Map

Table 4-13. FLASH Block Protection

FPS	FPOPEN	Address Area Protected	Memory Size Protected (bytes)	Number of Sectors Protected
0x1F	1	N/A	0	0
0x1E		0xFC00–0xFFFF	1K	2
0x1D		0xF800–0xFFFF	2K	4
0x1C		0xF400–0xFFFF	3K	6
0x1B		0xF000–0xFFFF	4K	8
0x1A		0xEC00–0xFFFF	5K	10
0x19		0xE800–0xFFFF	6K	12
0x18		0xE400–0xFFFF	7K	14
0x17		0xE000–0xFFFF	8K	16
...		...	...	18
0x07		0xA000–0xFFFF	24K	48
0x06		0x9C00–0xFFFF	25K	50
0x05		0x9800–0xFFFF	26K	52
0x04		0x9400–0xFFFF	27K	54
0x03		0x9000–0xFFFF	28K	56
0x02		0x8C00–0xFFFF	29K	58
0x01		0x8800–0xFFFF	30K	60
0x00		0x8400–0xFFFF	31K	62
N/A	0	0x8000–0xFFFF	32K	64

The COP counter is initialized by the first writes to the SOPT1 and SOPT2 registers after any system reset. Subsequent writes to SOPT1 and SOPT2 have no effect on COP operation. Even if the application will use the reset default settings of COPT, COPCLKS, and COPW bits, the user should write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. This will prevent accidental changes if the application program gets lost.

The write to SRS that services (clears) the COP counter 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.

If the bus clock source is selected, the COP counter does not increment while the MCU is in background debug mode or while the system is in stop mode. The COP counter resumes when the MCU exits background debug mode or stop mode.

If the 1-kHz clock source is selected, the COP counter is re-initialized to zero upon entry to either background debug mode or stop mode and begins from zero upon exit from background debug mode or stop mode.

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 an external interrupt 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 unless the local interrupt enable is a 1 to enable the interrupt and 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 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

5.7.7 System Power Management Status and Control 2 Register (SPMSC2)

This register is used to report the status of the low voltage warning function, and to configure the stop mode behavior of the MCU.

	7	6	5	4	3	2	1	0
R	0	0	LVDV[1]	LVWV	PPDF	0	0	PPDC[2]
W	▨	▨			▨	PPDACK	▨	
Power-on Reset:	0	0	0	0	0	0	0	0
LVD Reset:	0	0	u	u	0	0	0	0
Any other Reset:	0	0	u	u	0	0	0	0

▨ = Unimplemented or Reserved u = Unaffected by reset

[1] This bit can be written only one time after power-on reset. Additional writes are ignored.
[2] This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-9. System Power Management Status and Control 2 Register (SPMSC2)

Table 5-10. SPMSC2 Register Field Descriptions

Field	Description
5 LVDV	**Low-Voltage Detect Voltage Select** — This write-once bit selects the low voltage detect (LVD) trip point setting. It also selects the warning voltage range. See Table 5-11.
4 LVWV	**Low-Voltage Warning Voltage Select** — This bit selects the low voltage warning (LVW) trip point voltage. See Table 5-11.
3 PPDF	**Partial Power Down Flag** — This read-only status bit indicates that the MCU has recovered from stop2 mode. 0 MCU has not recovered from stop2 mode. 1 MCU recovered from stop2 mode.
2 PPDACK	**Partial Power Down Acknowledge** — Writing a 1 to PPDACK clears the PPDF bit
0 PPDC	**Partial Power Down Control** — This write-once bit controls whether stop2 or stop3 mode is selected. 0 Stop3 mode enabled. 1 Stop2, partial power down, mode enabled.

Table 5-11. LVD and LVW trip point typical values[1]

LVDV:LVWV	LVW Trip Point	LVD Trip Point
0:0	V_{LVW0} = 2.74 V	V_{LVD0} = 2.56 V
0:1	V_{LVW1} = 2.92 V	
1:0	V_{LVW2} = 4.3 V	V_{LVD1} = 4.0 V
1:1	V_{LVW3} = 4.6 V	

[1] See Electrical Characteristics appendix for minimum and maximum values.

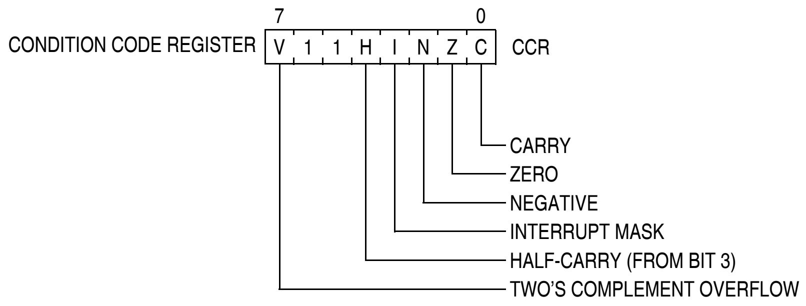
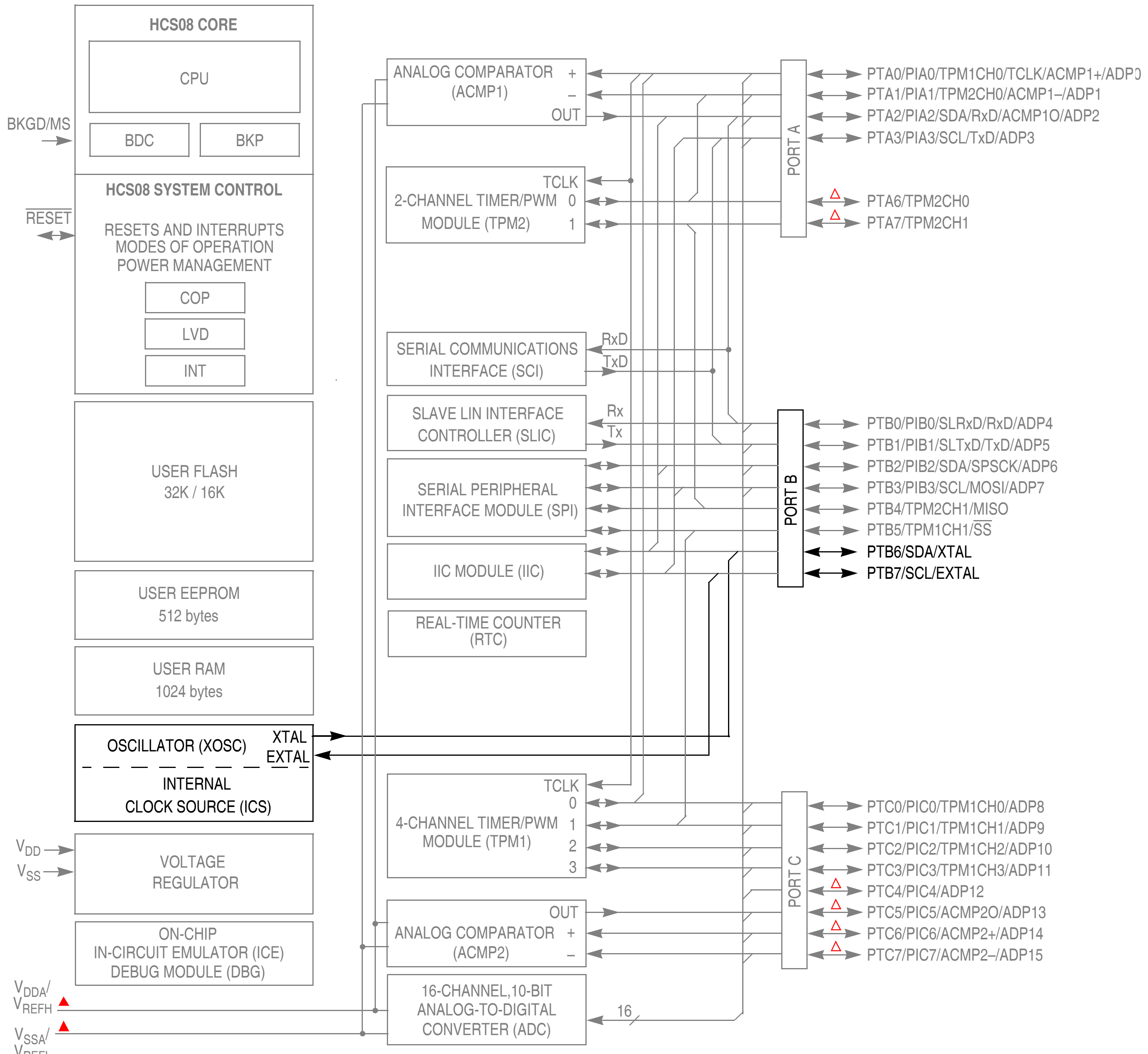


Figure 7-2. Condition Code Register

Table 7-1. CCR Register Field Descriptions

Field	Description
7 V	**Two's Complement Overflow Flag** — The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag. 0 No overflow 1 Overflow
4 H	**Half-Carry Flag** — The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C condition code bits to automatically add a correction value to the result from a previous ADD or ADC on BCD operands to correct the result to a valid BCD value. 0 No carry between bits 3 and 4 1 Carry between bits 3 and 4
3 I	**Interrupt Mask Bit** — When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the first instruction of the interrupt service routine is executed. Interrupts are not recognized at the instruction boundary after any instruction that clears I (CLI or TAP). This ensures that the next instruction after a CLI or TAP will always be executed without the possibility of an intervening interrupt, provided I was set. 0 Interrupts enabled 1 Interrupts disabled
2 N	**Negative Flag** — The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result. Simply loading or storing an 8-bit or 16-bit value causes N to be set if the most significant bit of the loaded or stored value was 1. 0 Non-negative result 1 Negative result
1 Z	**Zero Flag** — The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of 0x00 or 0x0000. Simply loading or storing an 8-bit or 16-bit value causes Z to be set if the loaded or stored value was all 0s. 0 Non-zero result 1 Zero result
0 C	**Carry/Borrow Flag** — The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag. 0 No carry out of bit 7 1 Carry out of bit 7



△ = Not bonded to pins in 20-pin package

▲ = In 20-pin packages, V_{DDA}/V_{REFH} is internally connected to V_{DD} and V_{SSA}/V_{REFL} is internally connected to V_{SS}.

Figure 8-1. Block Diagram Highlighting ICS Block and Pins

10.1.5 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.

10.1.6 Block Diagram

Figure 10-2 provides a block diagram of the ADC module

10.4.7.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

It is possible for the ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure that the data transfer blocking mechanism (discussed in Section 10.4.4.2, "Completing Conversions) is cleared when entering stop3 and continuing ADC conversions.

10.4.8 MCU Stop1 and Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters either stop1 or stop2 mode. All module registers contain their reset values following exit from stop1 or stop2. Therefore the module must be re-enabled and re-configured following exit from stop1 or stop2.

10.5 Initialization Information

This section gives an example which provides some basic direction on how a user would initialize and configure the ADC module. The user has the flexibility of choosing between configuring the module for 8-bit or 10-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 10-6, Table 10-7, and Table 10-8 for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

10.5.1 ADC Module Initialization Example

10.5.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.

converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 LSB and will increase with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 10.6.2.3 will reduce this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values which are never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and to have no missing codes.

method was employed for this message frame. Refer to the LIN specification for more details on the calculations.

- **Byte Framing Error**
  This error comes from the standard UART definition for byte encoding and occurs when the STOP bit is sampled and reads back as a 0Ø0. STOP should always read as 1.

NOTE

A byte framing error can also be an indication that the number of data bytes received in a LIN message frame does not match the value written to the SLCDLC register. See Section 12.6.7, "Handling LIN Message Headers," for more details.

- **Identifier Received Successfully**
  This interrupt source indicates that a LIN identifier byte has been received with correct parity and is waiting in the LIN identifier buffer (SLCID). Upon reading this interrupt source from SLCSV, the user can then decode the identifier in software to determine the nature of the LIN message frame. To clear this source, SLCID must be read.
- **Identifier-Parity-Error**
  A parity error in the identifier (i.e., corrupted identifier) will be flagged. Typical LIN slave applications do not distinguish between an unknown but valid identifier, and a corrupted identifier. However, it is mandatory for all slave nodes to evaluate in case of a known identifier all eight bits of the ID-Field and distinguish between a known and a corrupted identifier. The received identifier value is reported in SLCID so that the user software can choose to acknowledge or ignore the parity error message. Once the ID parity error has been detected, the SLIC will begin looking for another LIN header and will not receive message data, even if it appears on the bus.
- **Wakeup**
  The wakeup interrupt source indicates that the SLIC module has entered SLIC run mode from SLIC stop mode.

12.3.5.2 Byte Transfer Mode Operation

When byte transfer mode is enabled (BTM = 1), many of the interrupt sources for the SLCSV no longer apply, as they are specific to LIN operations. Table 12-9 shows those interrupt sources which are applicable to BTM operations. The value of the SLCSV for each interrupt source remains the same, as well as the priority of the interrupt source.

Table 12-9. Interrupt Sources Summary (BTM = 1)

SLCSV	I3	I2	I1	I0	Interrupt Source	Priority
0x00	0	0	0	0	No Interrupts Pending	0 (Lowest)
0x0C	0	0	1	1	TX Message Buffer Empty	3
0x14	0	1	0	1	RX Data Buffer Full No Errors	5
0x18	0	1	1	0	Bit-Error	6
0x1C	0	1	1	1	Receiver Buffer Overrun	7

Table 13-3. SPIC2 Register Field Descriptions

Field	Description
4 MODFEN	**Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 13-2 for more details). 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	**Bidirectional Mode Output Enable** — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	**SPI Stop in Wait Mode** 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	**SPI Pin Control 0** — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin. 0 SPI uses separate pins for data input and data output 1 SPI configured for single-wire bidirectional operation

13.4.3 SPI Baud Rate Register (SPIBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

	7	6	5	4	3	2	1	0
R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
W	▨				▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 13-7. SPI Baud Rate Register (SPIBR)

Table 13-4. SPIBR Register Field Descriptions

Field	Description
6:4 SPPR[2:0]	**SPI Baud Rate Prescale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 13-5. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 13-4).
2:0 SPR[2:0]	**SPI Baud Rate Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 13-6. The input to this divider comes from the SPI baud rate prescaler (see Figure 13-4). The output of this divider is the SPI bit rate clock for master mode.

Table 14-7. SCIxC3 Field Descriptions (continued)

Field	Description
4 TXINV[1]	**Transmit Data Inversion** — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	**Overrun Interrupt Enable** — This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR = 1.
2 NEIE	**Noise Error Interrupt Enable** — This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF = 1.
1 FEIE	**Framing Error Interrupt Enable** — This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE = 1.
0 PEIE	**Parity Error Interrupt Enable** — This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF = 1.

[1] Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

14.2.7 SCI Data Register (SCIxD)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 14-11. SCI Data Register (SCIxD)

14.3 Functional Description

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

14.3.1 Baud Rate Generation

As shown in Figure 14-12, the clock source for the SCI baud rate generator is the bus-rate clock.

17.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

	7	6	5	4	3	2	1	0
R	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 17-10. Debug Status Register (DBGS)

Table 17-6. DBGS Register Field Descriptions

Field	Description
7 AF	**Trigger Match A Flag** — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	**Trigger Match B Flag** — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	**Arm Flag** — While DBGEN = 1, this status bit is a read-only image of ARM in DBGC. This bit is set by writing 1 to the ARM control bit in DBGC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGC. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	**FIFO Valid Count** — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

Table A-7. Supply Current Characteristics (continued)

Num	C	Parameter	Symbol	V_{DD} (V)	Typ[1]	Max[2]	Unit
5		Stop2 mode supply current					
	C	–40°C (C,M, & V suffix)	$S2I_{DD}$	5	0.9	–	μA
	P	25°C (All parts)			0.9	–	
	P[5]	85°C (C suffix only)			5.0	40.0	
	P[5]	105°C (V suffix only)			11.0	50.0	
	P[5]	125°C (M suffix only)			29.1	65.0	
	C	–40°C (C,M, & V suffix)		3	0.9	–	μA
	P	25°C (All parts)			0.9	–	
	P[5]	85°C (C suffix only)			4.2	35.0	
	P[5]	105°C (V suffix only)			8.8	45.0	
	P[5]	125°C (M suffix only)			25	60.0	
6	C	RTC adder to stop2 or stop3[6]	$S23I_{DDRTI}$	5	300	500	nA
				3	300	500	nA
7	C	LVD adder to stop3 (LVDE = LVDSE = 1)	$S3I_{DDLVD}$	5	110	180	μA
				3	90	160	μA
8	C	Adder to stop3 for oscillator enabled[7] (EREFSTEN =1)	$S3I_{DDOSC}$	5,3	5	8	μA

1 Typical values for specs 1, 2, 3, 6, 7, and 8 are based on characterization data at 25°C. See Figure A-5 through Figure A-7 for typical curves across temperature and voltage.

2 Max values in this column apply for the full operating temperature range of the device unless otherwise noted.

3 All modules except ADC active, ICS configured for FBELP, and does not include any dc loads on port pins

4 All modules except ADC active, ICS configured for FEI, and does not include any dc loads on port pins

5 Stop currents are tested in production for 25°C on all parts. Tests at other temperatures depend upon the part number suffix and maturity of the product. Freescale may eliminate a test insertion at a particular temperature from the production test flow once sufficient data has been collectd and is approved.

6 Most customers are expected to find that auto-wakeup from stop2 or stop3 can be used instead of the higher current wait mode.

7 Values given under the following conditions: low range operation (RANGE = 0) with a 32.768kHz crystal and low power mode (HGO = 0).

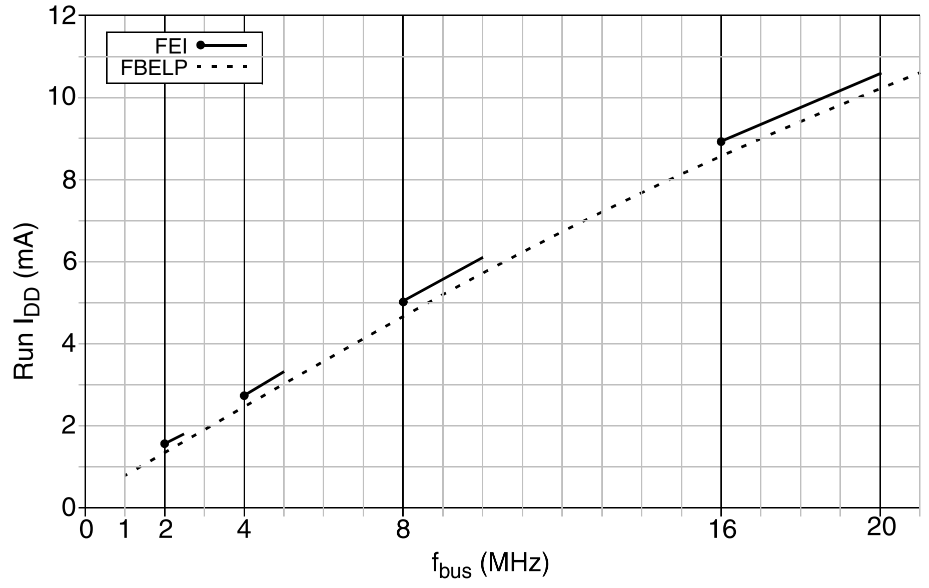


Figure A-5. Typical Run I_{DD} vs. Bus Frequency (V_{DD} = 5V)

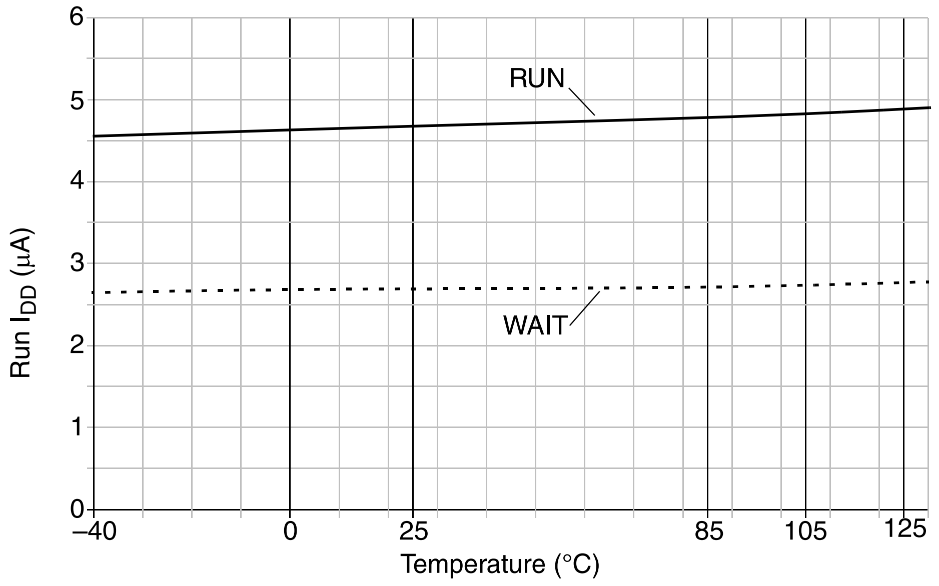


Figure A-6. Typical Run and Wait I_{DD} vs. Temperature (V_{DD} = 5V; f_{bus} = 8MHz)

Table A-10. Analog Comparator Electrical Specifications (continued)

Num	C	Rating	Symbol	Min	Typical	Max	Unit
4	D	Analog input offset voltage	V_{AIO}		20	40	mV
5	D	Analog Comparator hysteresis	V_H	3.0	6.0	20.0	mV
6	D	Analog input leakage current	I_{ALKG}	—	—	1.0	μA
7	D	Analog Comparator initialization delay	t_{AINIT}	—	—	1.0	μs

A.11 ADC Characteristics

Table A-11. ADC Operating Conditions

Num	Characteristic	Conditions	Symb	Min	Typ[1]	Max	Unit	Comment
1	Supply voltage	Absolute	V_{DDAD}	2.7	—	5.5	V	
2	Input Voltage		V_{ADIN}	V_{REFL}	—	V_{REFH}	V	
3	Input Capacitance		C_{ADIN}	—	4.5	5.5	pF	
4	Input Resistance		R_{ADIN}	—	3	5	kΩ	
5	Analog Source Resistance	10 bit mode $f_{ADCK} > 4MHz$ $f_{ADCK} < 4MHz$	R_{AS}	 — —	 — —	 5 10	kΩ	External to MCU
6		8 bit mode (all valid f_{ADCK})		—	—	10		
7	ADC Conversion Clock Freq.	High Speed (ADLPC=0)	f_{ADCK}	0.4	—	8.0	MHz	
8		Low Power (ADLPC=1)		0.4	—	4.0		

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