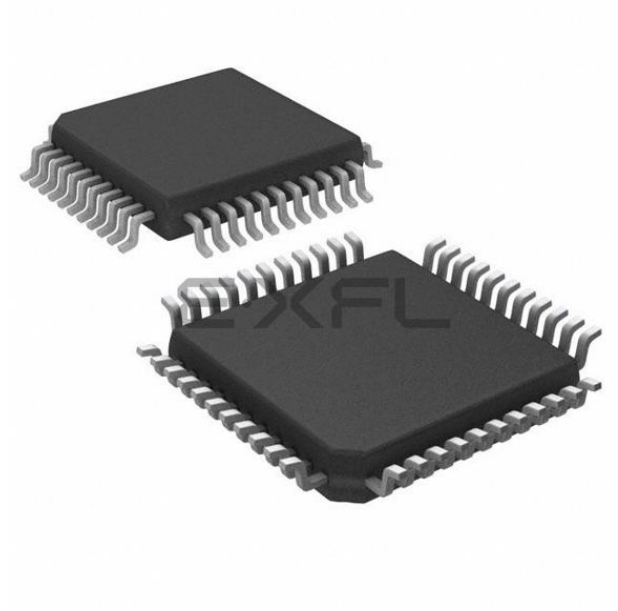

Welcome to E-XFL.COM

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	Not For New Designs
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	I²C, IRSCI, SCI, SPI
Peripherals	LED, LVD, POR, PWM
Number of I/O	32
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 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-QFP
Supplier Device Package	44-QFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908ap16cfbe

Table 2-1. Vector Addresses

Priority	INT Flag	Address	Vector
Lowest	—	$FFD0	Reserved
		$FFD1	Reserved
	IF21	$FFD2	TBM Vector (High)
		$FFD3	TBM Vector (Low)
	IF20	$FFD4	SCI2 (IRSCI) Transmit Vector (High)
		$FFD5	SCI2 (IRSCI) Transmit Vector (Low)
	IF19	$FFD6	SCI2 (IRSCI) Receive Vector (High)
		$FFD7	SCI2 (IRSCI) Receive Vector (Low)
	IF18	$FFD8	SCI2 (IRSCI) Error Vector (High)
		$FFD9	SCI2 (IRSCI) Error Vector (Low)
	IF17	$FFDA	SPI Transmit Vector (High)
		$FFDB	SPI Transmit Vector (Low)
	IF16	$FFDC	SPI Receive Vector (High)
		$FFDD	SPI Receive Vector (Low)
	IF15	$FFDE	ADC Conversion Complete Vector (High)
		$FFDF	ADC Conversion Complete Vector (Low)
	IF14	$FFE0	Keyboard Vector (High)
		$FFE1	Keyboard Vector (Low)
	IF13	$FFE2	SCI Transmit Vector (High)
		$FFE3	SCI Transmit Vector (Low)
	IF12	$FFE4	SCI Receive Vector (High)
		$FFE5	SCI Receive Vector (Low)
	IF11	$FFE6	SCI Error Vector (High)
		$FFE7	SCI Error Vector (Low)
	IF10	$FFE8	MMIIC Interrupt Vector (High)
		$FFE9	MMIIC Interrupt Vector (Low)
	IF9	$FFEA	TIM2 Overflow Vector (High)
		$FFEB	TIM2 Overflow Vector (Low)

NOTE

For correct operation, the stack pointer must point only to RAM locations.

Within page zero are 160 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for I/O control and user data or code. When the stack pointer is moved from its reset location at $00FF, direct addressing mode instructions can access efficiently all page zero RAM locations. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

Before processing an interrupt, the CPU uses five bytes of the stack to save the contents of the CPU registers.

NOTE

For M6805 compatibility, the H register is not stacked.

During a subroutine call, the CPU uses two bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

NOTE

Be careful when using nested subroutines. The CPU may overwrite data in the RAM during a subroutine or during the interrupt stacking operation.

2.5 FLASH Memory

This sub-section describes the operation of the embedded FLASH memory. This memory can be read, programmed, and erased from a single external supply. The program and erase operations are enabled through the use of an internal charge pump. The following table shows the FLASH memory size and address range:

Device	FLASH Size (Bytes)	Memory Address Range
MC68HC908AP64	62,368	$0860–$FBFF
MC68HC908AP32	32,768	$0860–$885F
MC68HC908AP16	16,384	$0860–$485F
MC68HC908AP8	8,192	$0860–$285F

2.5.1 Functional Description

The FLASH memory consists of an array for user memory plus a block of 48 bytes for user interrupt vectors and one byte for the mask option register. *An erased bit reads as logic 1 and a programmed bit reads as a logic 0.* The FLASH memory page size is defined as 512 bytes, and is the minimum size that can be erased in a page erase operation. Program and erase operations are facilitated through control bits in FLASH control register (FLCR). The address ranges for the FLASH memory are:

- $0860–$FBFF; user memory, 62,368 / 32,768 / 16,384 / 8,192 bytes
- $FFD0–$FFFF; user interrupt vectors, 48 bytes
- $FFCF; mask option register

Programming tools are available from Freescale. Contact your local Freescale representative for more information.

NOTE

A security feature prevents viewing of the FLASH contents.[1]

3.4 Configuration Register 2 (CONFIG2)

Address:	$001D							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	STOP_ ICLKDIS	STOP_ RCLKEN	STOP_ XCLKEN	OSCCLK1	OSCCLK0	0	0	SCIBD- SRC
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 3-3. Configuration Register 2 (CONFIG2)

STOP_ICLKDIS — Internal Oscillator Stop Mode Disable

STOP_ICLKDIS disables the internal oscillator during stop mode. Setting the STOP_ICLKDIS bit disables the oscillator during stop mode. (See Chapter 5 Oscillator (OSC).)

Reset clears this bit.

1 = Internal oscillator disabled during stop mode

0 = Internal oscillator enabled to operate during stop mode

STOP_RCLKEN — RC Oscillator Stop Mode Enable Bit

STOP_RCLKEN enables the RC oscillator to continue operating during stop mode. Setting the STOP_RCLKEN bit allows the oscillator to operate continuously even during stop mode. This is useful for driving the timebase module to allow it to generate periodic wake up while in stop mode. (See Chapter 5 Oscillator (OSC).)

Reset clears this bit.

1 = RC oscillator enabled to operate during stop mode

0 = RC oscillator disabled during stop mode

STOP_XCLKEN — X-tal Oscillator Stop Mode Enable Bit

STOP_XCLKEN enables the crystal (x-tal) oscillator to continue operating during stop mode. Setting the STOP_XCLKEN bit allows the x-tal oscillator to operate continuously even during stop mode. This is useful for driving the timebase module to allow it to generate periodic wake up while in stop mode. (See Chapter 5 Oscillator (OSC).) Reset clears this bit.

1 = X-tal oscillator enabled to operate during stop mode

0 = X-tal oscillator disabled during stop mode

OSCCLK1, OSCCLK0 — Oscillator Output Control Bits

OSCCLK1 and OSCCLK0 select which oscillator output to be driven out as OSCCLK to the timebase module (TBM). Reset clears these two bits.

OSCCLK1	OSCCLK0	Timebase Clock Source
0	0	Internal oscillator (ICLK)
0	1	RC oscillator (RCCLK)
1	0	X-tal oscillator (XTAL)
1	1	Not used

SCIBDSRC — SCI Baud Rate Clock Source

SCIBDSRC selects the clock source used for the standard SCI module (non-infrared SCI). The setting of this bit affects the frequency at which the SCI operates.

1 = Internal data bus clock, f_{BUS}, is used as clock source for SCI

0 = Oscillator clock, CGMXCLK, is used as clock source for SCI

3.5 Mask Option Register (MOR)

The mask option register (MOR) is used for selecting one of the three clock options for the MCU. The MOR is a byte located in FLASH memory, and is written to by a FLASH programming routine.

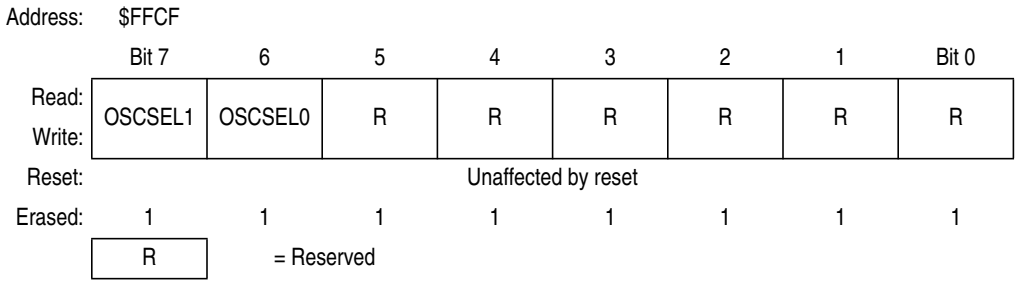

Figure 3-4. Mask Option Register (MOR)

OSCSEL1, OSCSEL0 — Oscillator Selection Bits

OSCSEL1 and OSCSEL0 select which oscillator is used for the MCU CGMXCLK clock. The erase state of these two bits is logic 1. These bits are unaffected by reset. (See Table 3-1).

Bits 5–0 — Should be left as 1's

Table 3-1. CGMXCLK Clock Selection

OSCSEL1	OSCSEL0	CGMXCLK	OSC2 pin	Comments
0	0	—	—	Not used
0	1	ICLK	f_{BUS}	Internal oscillator generates the CGMXCLK.
1	0	RCCLK	f_{BUS}	RC oscillator generates the CGMXCLK. Internal oscillator is available after each POR or reset.
1	1	X-TAL	Inverting output of XTAL	X-tal oscillator generates the CGMXCLK. Internal oscillator is available after each POR or reset.

NOTE

The internal oscillator is a free running oscillator and is available after each POR or reset. It is turned-off in stop mode by setting the STOP_ICLKDIS bit in CONFIG2.

5.3 Internal Oscillator

The internal oscillator clock (ICLK), with a frequency of f_{ICLK}, is a free running clock that requires no external components. It can be selected as the CGMXCLK for the CGM and MCU sub-systems; and the OSCCLK clock for the TBM. The ICLK is also the reference clock input to the computer operating properly (COP) module.

Due to the simplicity of the internal oscillator, it does not have the accuracy and stability of the RC oscillator or the x-tal oscillator. Therefore, the ICLK is not suitable where an accurate bus clock is required and it should not be used as the CGMRCLK to the CGM PLL.

The internal oscillator by default is always available and is free running after POR or reset. It can be turned-off in stop mode by setting the STOP_ICLKDIS bit before executing the STOP instruction.

Figure 5-4 shows the logical representation of components of the internal oscillator circuitry.

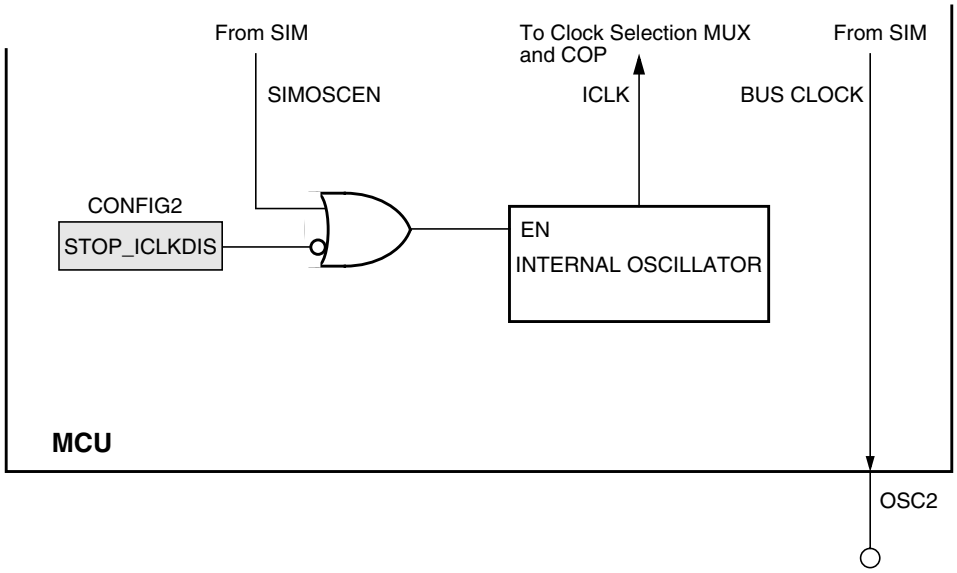

Figure 5-4. Internal Oscillator

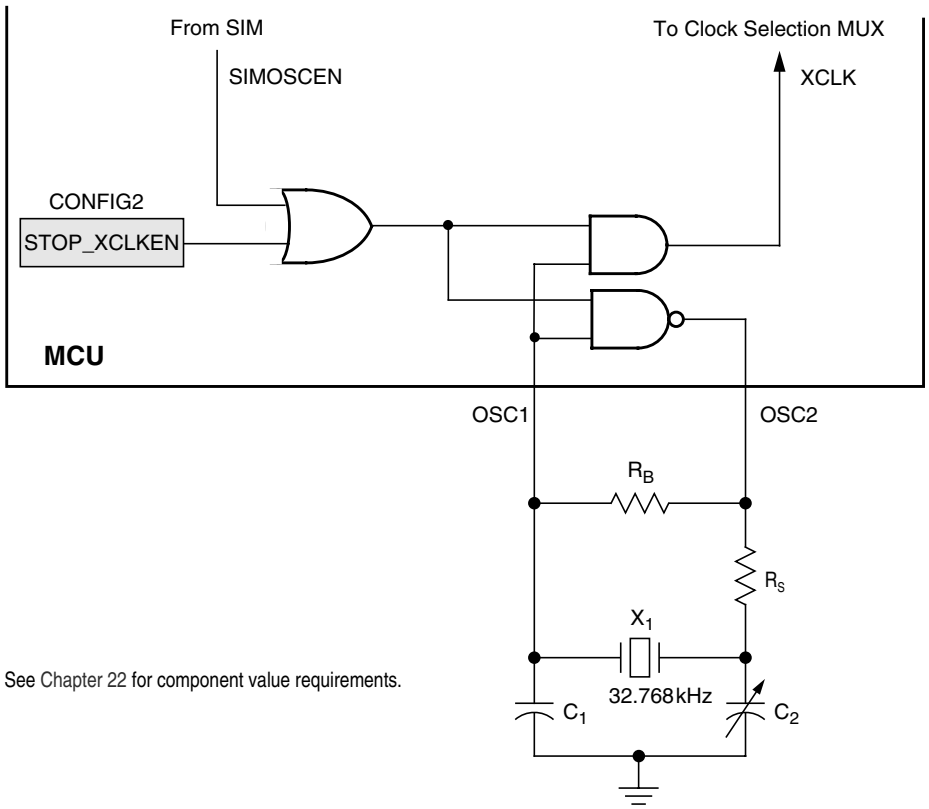

Figure 5-6. Crystal Oscillator

The series resistor (R_S) is included in the diagram to follow strict Pierce oscillator guidelines and may not be required for all ranges of operation, especially with high frequency crystals. Refer to the crystal manufacturer's data for more information.

5.6 I/O Signals

The following paragraphs describe the oscillator I/O signals.

5.6.1 Crystal Amplifier Input Pin (OSC1)

OSC1 pin is an input to the crystal oscillator amplifier or the input to the RC oscillator circuit.

5.6.2 Crystal Amplifier Output Pin (OSC2)

When the x-tal oscillator is selected, OSC2 pin is the output of the crystal oscillator inverting amplifier.

When the RC oscillator or internal oscillator is selected, OSC2 pin is the output of the internal bus clock.

5.6.3 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal from the system integration module (SIM) enables/disables the x-tal oscillator, the RC-oscillator, or the internal oscillator circuit.

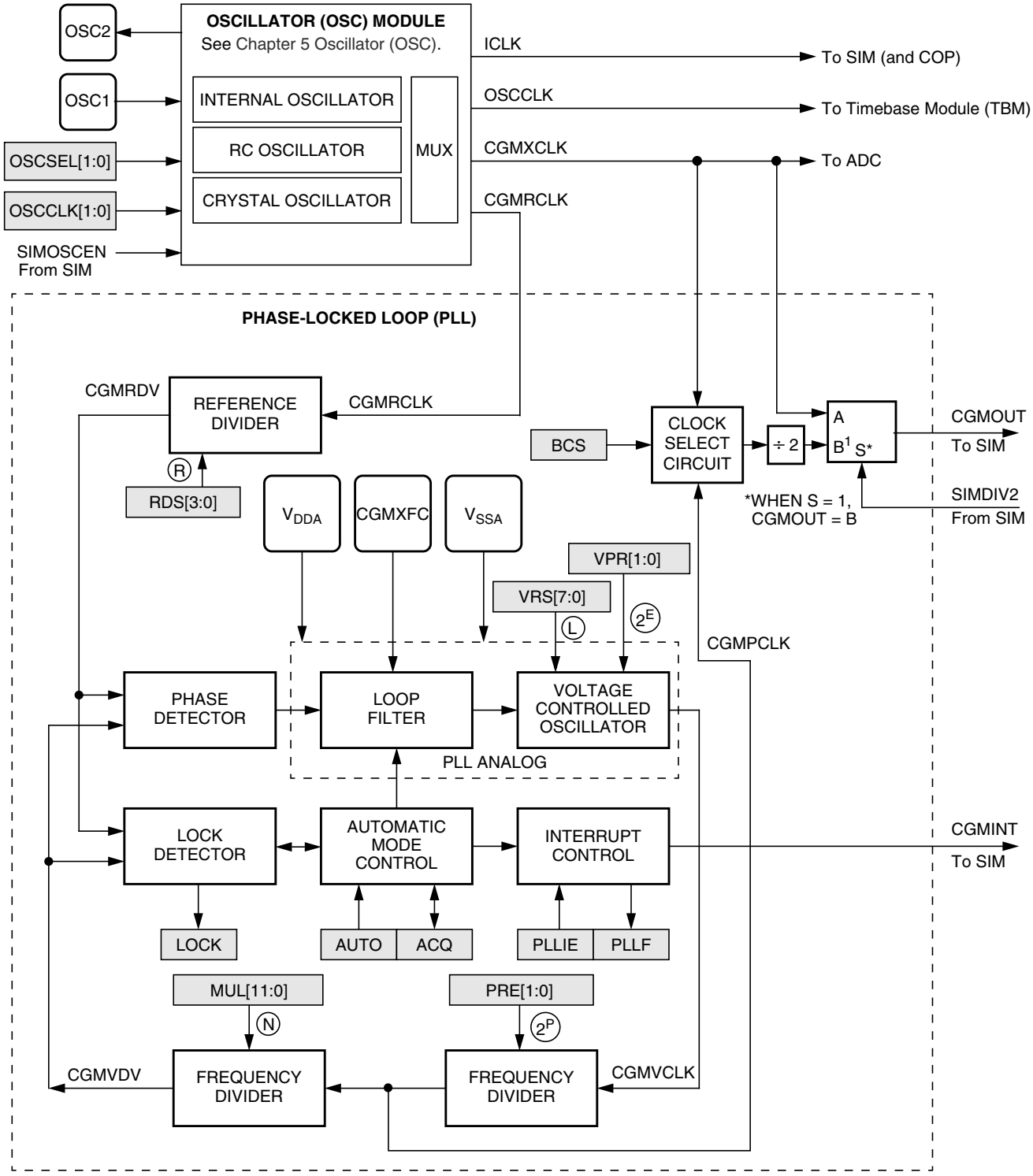

Figure 6-1. CGM Block Diagram

7.5.2.1 Interrupt Status Register 1

Address: $FE04

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-12. Interrupt Status Register 1 (INT1)

IF6–IF1 — Interrupt Flags 6–1

These flags indicate the presence of interrupt requests from the sources shown in Table 7-3.

1 = Interrupt request present
0 = No interrupt request present

Bit 0 and Bit 1 — Always read 0

7.5.2.2 Interrupt Status Register 2

Address: $FE05

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-13. Interrupt Status Register 2 (INT2)

IF14–IF7 — Interrupt Flags 14–7

These flags indicate the presence of interrupt requests from the sources shown in Table 7-3.

1 = Interrupt request present
0 = No interrupt request present

7.5.2.3 Interrupt Status Register 3

Address: $FE06

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	IF21	IF20	IF19	IF18	IF17	IF16	IF15
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-14. Interrupt Status Register 3 (INT3)

IF21–IF15 — Interrupt Flags 21–15

These flags indicate the presence of an interrupt request from the source shown in Table 7-3.

1 = Interrupt request present
0 = No interrupt request present

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

NOTE

The MCU does not transmit a break character until after the host sends the eight security bits.

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $60 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

8.5 ROM-Resident Routines

Seven routines stored in the monitor ROM area (thus ROM-resident) are provided for FLASH memory manipulation. Five of the seven routines are intended to simplify FLASH program, erase, and load operations. The other two routines are intended to simplify the use of the FLASH memory as EEPROM. Table 8-10 shows a summary of the ROM-resident routines.

Table 8-10. Summary of ROM-Resident Routines

Routine Name	Routine Description	Call Address	Stack Used (bytes)
PRGRNGE	Program a range of locations	$FC34	15
ERARNGE	Erase a page or the entire array	$FCE4	9
LDRNGE	Loads data from a range of locations	$FC00	7
MON_PRGRNGE	Program a range of locations in monitor mode	$FF24	17
MON_ERARNGE	Erase a page or the entire array in monitor mode	$FF28	11
EE_WRITE	Emulated EEPROM write. Data size ranges from 7 to 15 bytes at a time.	$FF36	30
EE_READ	Emulated EEPROM read. Data size ranges from 7 to 15 bytes at a time.	$FD5B	18

The routines are designed to be called as stand-alone subroutines in the user program or monitor mode. The parameters that are passed to a routine are in the form of a contiguous data block, stored in RAM. The index register (H:X) is loaded with the address of the first byte of the data block (acting as a pointer), and the subroutine is called (JSR). Using the start address as a pointer, multiple data blocks can be used, any area of RAM be used. A data block has the control and data bytes in a defined order, as shown in Figure 8-9.

Chapter 10
Timebase Module (TBM)

10.1 Introduction

This section describes the timebase module (TBM). The TBM will generate periodic interrupts at user selectable rates using a counter clocked by the selected OSCCLK clock from the oscillator module. This TBM version uses 18 divider stages, eight of which are user selectable.

10.2 Features

Features of the TBM module include:

- Software programmable 8s, 4s, 2s, 1s, 2ms, 1ms, 0.5ms, and 0.25ms periodic interrupt using 32.768-kHz OSCCLK clock
- User selectable oscillator clock source enable during stop mode to allow periodic wake-up from stop

10.3 Functional Description

This module can generate a periodic interrupt by dividing the oscillator clock frequency, OSCCLK. The counter is initialized to all 0s when TBON bit is cleared. The counter, shown in Figure 10-1, starts counting when the TBON bit is set. When the counter overflows at the tap selected by TBR[2:0], the TBIF bit gets set. If the TBIE bit is set, an interrupt request is sent to the CPU. The TBIF flag is cleared by writing a 1 to the TACK bit. The first time the TBIF flag is set after enabling the timebase module, the interrupt is generated at approximately half of the overflow period. Subsequent events occur at the exact period.

The reference clock OSCCLK is derived from the oscillator module, see 5.2.2 TBM Reference Clock Selection.

Table 12-7. SCI Baud Rate Prescaling

SCP1 and SCP0	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in Table 12-8. Reset clears SCR2–SCR0.

Table 12-8. IRSCI Baud Rate Selection

SCR2, SCR1, and SCR0	Baud Rate Divisor (BD)
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Use this formula to calculate the SCI baud rate:

$$\text{baud rate} = \frac{\text{SCI clock source}}{16 \times \text{PD} \times \text{BD}}$$

where:

SCI clock source = f_{BUS} or CGMXCLK
(selected by CKS bit)
PD = prescaler divisor
BD = baud rate divisor

Table 12-9 shows the SCI baud rates that can be generated with a 4.9152-MHz bus clock when f_{BUS} is selected as SCI clock source.

13.12.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transmission. (See 13.5 Transmission Formats.) Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format. See Figure 13-12.

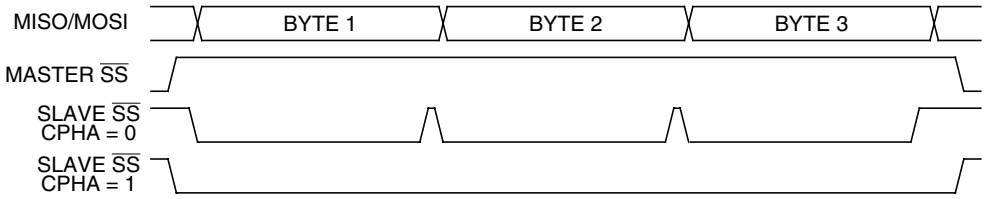

Figure 13-12. CPHA/$\overline{SS}$ Timing

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{SS}$ from creating a MODF error. (See 13.13.2 SPI Status and Control Register.)

NOTE

A logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See 13.7.2 Mode Fault Error.) For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the port data register. (See Table 13-3.)

Table 13-3. SPI Configuration

SPE	SPMSTR	MODFEN	SPI Configuration	State of $\overline{SS}$ Logic
0	X(1)	X	Not enabled	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

Note 1. X = Don't care

14.8.4 Write Byte/Word

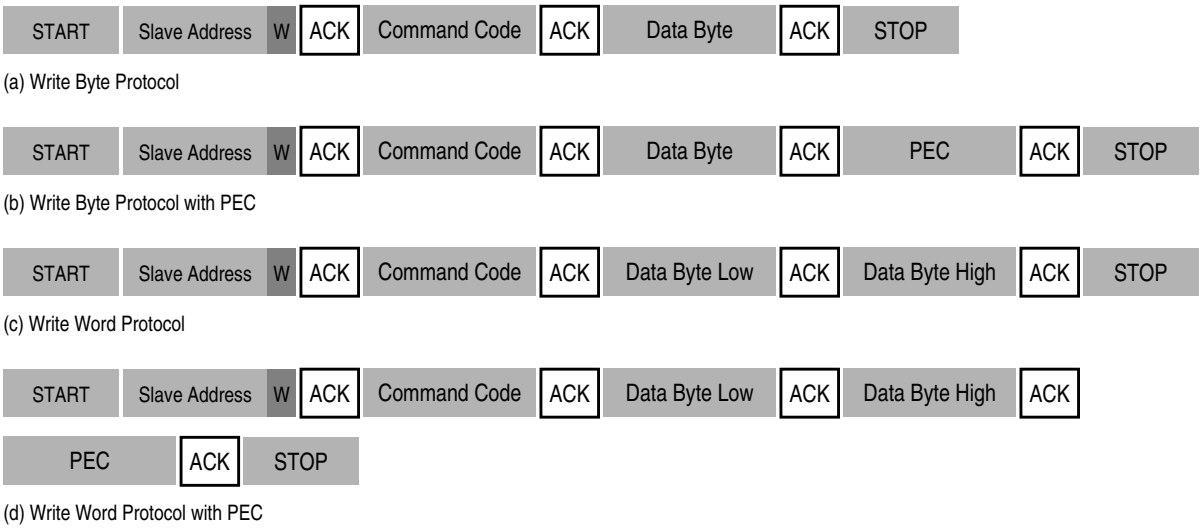

Figure 14-16. Write Byte/Word

14.8.5 Read Byte/Word

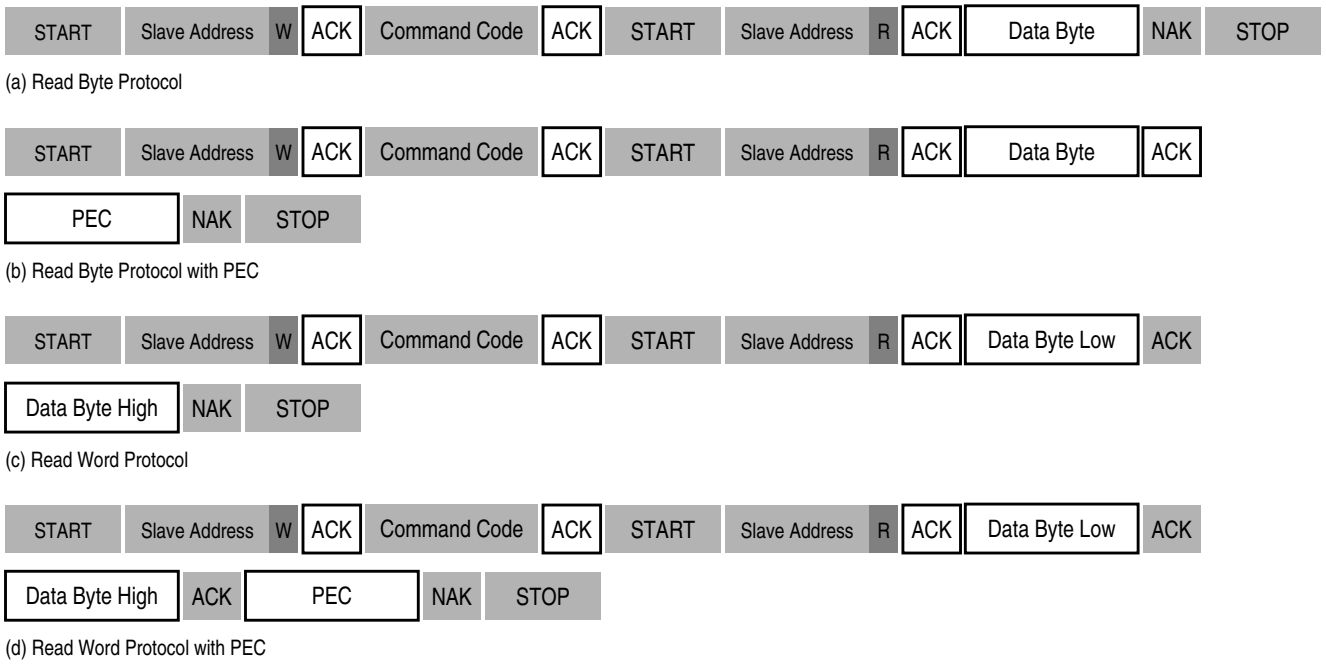

Figure 14-17. Read Byte/Word

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0059	ADC Data Register High 0 (ADRH0)	Read:	$\overline{AD9}$	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005A	ADC Data Register Low 0 (ADRL0)	Read:	AD1	AD0	0	0	0	0	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

Figure 15-8 ADRH0 and ADRL0 in Left Justified Sign Data Mode

15.7.4 ADC Auto-Scan Mode Data Registers (ADRL1–ADRL3)

The ADC data registers 1 to 3 (ADRL1–ADRL3), are 8-bit registers for conversion results in 8-bit truncated mode, for channels ADC1 to ADC3, when the ADC is operating in auto-scan mode (MODE[1:0] = 00).

Address: ADRL1, $005B; ADRL2, $005C; and ADRL3, $005D

Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 15-9. ADC Data Register Low 1 to 3 (ADRL1–ADRL3)

15.7.5 ADC Auto-Scan Control Register (ADASCR)

The ADC auto-scan control register (ADASCR) enables and controls the ADC auto-scan function.

Address: $005E

Read:	0	0	0	0	0	AUTO1	AUTO0	ASCAN
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 15-10. ADC Scan Control Register (ADASCR)

AUTO[1:0] — Auto-Scan Mode Channel Select Bits

AUTO1 and AUTO0 form a 2-bit field which is used to define the number of auto-scan channels used when in auto-scan mode. Reset clears these bits.

Table 15-4. Auto-scan Mode Channel Select

AUTO1	AUTO0	Auto-Scan Channels
0	0	ADC0 only
0	1	ADC0 to ADC1
1	0	ADC0 to ADC2
1	1	ADC0 to ADC3

16.2.3 Port-A LED Control Register (LEDA)

The port-A LED control register (LEDA) controls the direct LED drive capability on PTA7–PTA0 pins. Each bit is individually configurable and requires that the data direction register, DDRA, bit be configured as an output.

Address: $000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	LEDA7	LEDA6	LEDA5	LEDA4	LEDA3	LEDA2	LEDA1	LEDA0
Reset:	0	0	0	0	0	0	0	0

Figure 16-5. Port A LED Control Register (LEDA)

LEDA[7:0] — Port A LED Drive Enable Bits

These read/write bits are software programmable to enable the direct LED drive on an output port pin.

1 = Corresponding port A pin is configured for direct LED drive,
with 15mA current sinking capability

0 = Corresponding port A pin is configured for standard drive

16.3 Port B

Port B is an 8-bit special-function port that shares two of its pins with the multi-master IIC (MMIIC) module, two of its pins with SCI module, and four of its pins with two timer interface (TIM1 and TIM2) modules.

NOTE

PTB3–PTB0 are open-drain pins when configured as outputs regardless whether the pins are used as general purpose I/O pins, MMIIC pins, or SCI pins. Therefore, when configured as general purpose output pins, MMIIC pins, or SCI pins (the TxD pin), pullup resistors must be connected to these pins.

16.3.1 Port B Data Register (PTB)

The port B data register contains a data latch for each of the eight port B pins.

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Reset:	Unaffected by reset							
Alternative Function:	T2CH1	T2CH0	T1CH1	T1CH0	RxD	TxD	SCL	SDA

Figure 16-6. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

These read/write bits are software-programmable. Data direction of each port B pin is under the control of the corresponding bit in data direction register B. Reset has no effect on port B data.

22.16 3V SPI Characteristics

Table 22-16. SPI Characteristics (3V)

Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	 $f_{OP(M)}$ $f_{OP(S)}$	 $f_{OP}/128$ dc	 $f_{OP}/2$ f_{OP}	 MHz MHz
1	Cycle time Master Slave	 $t_{CYC(M)}$ $t_{CYC(S)}$	 2 1	 128 —	 t_{CYC} t_{CYC}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{CYC}
3	Enable lag time	$t_{Lag(S)}$	1	—	t_{CYC}
4	Clock (SPSCK) high time Master Slave	 $t_{SCKH(M)}$ $t_{SCKH(S)}$	 t_{CYC} –25 1/2 t_{CYC} –25	 64 t_{CYC} —	 ns ns
5	Clock (SPSCK) low time Master Slave	 $t_{SCKL(M)}$ $t_{SCKL(S)}$	 t_{CYC} –25 1/2 t_{CYC} –25	 64 t_{CYC} —	 ns ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 40 40	 — —	 ns ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 40 40	 — —	 ns ns
8	Access time, slave[3] CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 50 50	 ns ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	50	ns
10	Data valid time, after enable edge Master Slave[5]	 $t_{V(M)}$ $t_{V(S)}$	 — —	 60 60	 ns ns
11	Data hold time, outputs, after enable edge Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 0	 — —	 ns ns

1. Numbers refer to dimensions in Figure 22-3 and Figure 22-4.
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins