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	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	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K 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/pro/item?MUrl=&PartUrl=mc908ap64acfbe

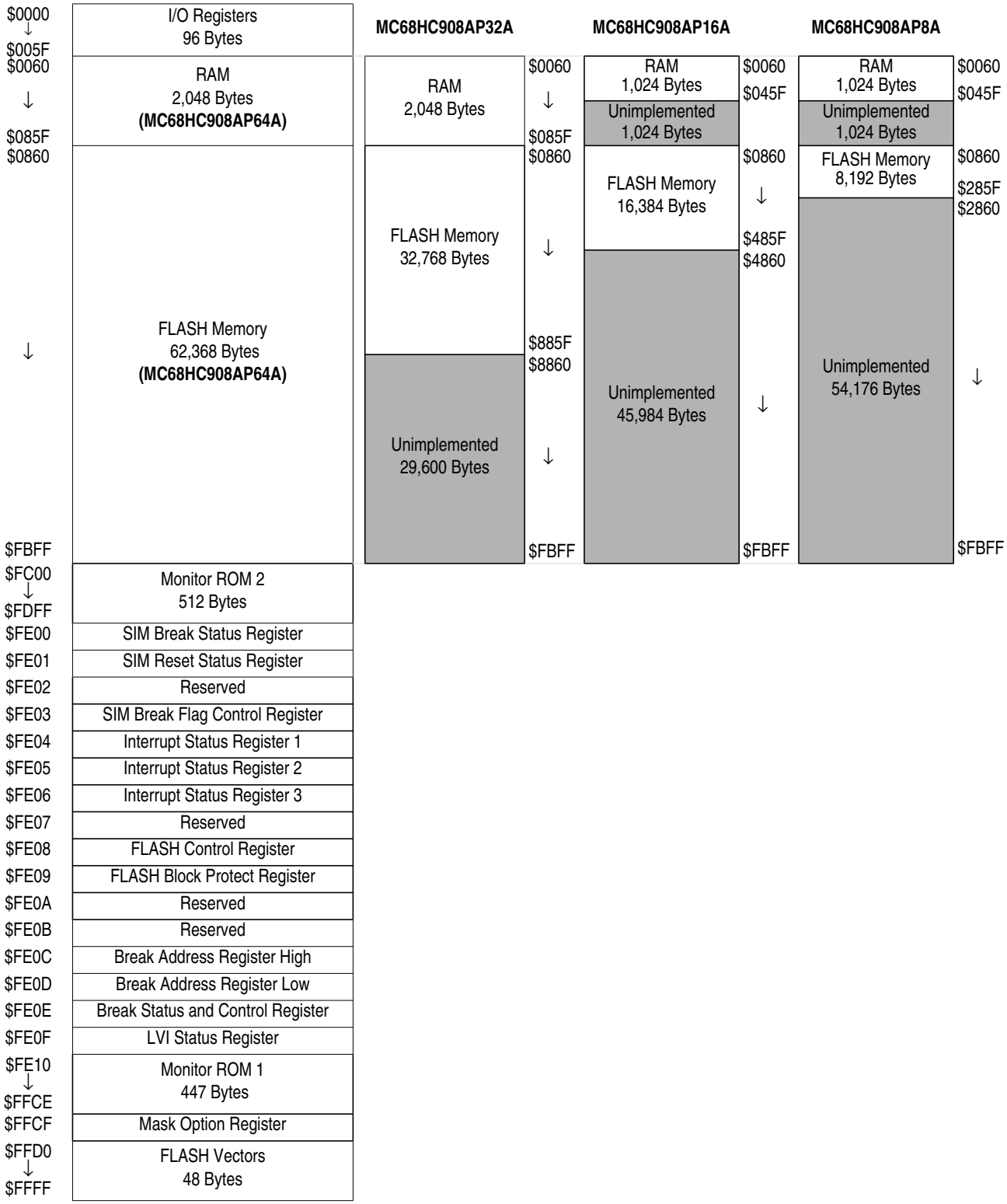

Figure 2-1. Memory Map

2.5.2 FLASH Control Register

The FLASH control register (FLCR) controls FLASH program and erase operation.

Address: $FE08

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	HVEN	MASS	ERASE	PGM
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 2-3. FLASH Control Register (FLCR)

HVEN — High Voltage Enable Bit

This read/write bit enables the charge pump to drive high voltages for program and erase operations in the array. HVEN can only be set if either PGM = 1 or ERASE = 1 and the proper sequence for program or erase is followed.

1 = High voltage enabled to array and charge pump on
0 = High voltage disabled to array and charge pump off

MASS — Mass Erase Control Bit

This read/write bit configures the memory for mass erase operation or page erase operation when the ERASE bit is set.

1 = Mass erase operation selected
0 = Page erase operation selected

ERASE — Erase Control Bit

This read/write bit configures the memory for erase operation. ERASE is interlocked with the PGM bit such that both bits cannot be equal to 1 or set to 1 at the same time.

1 = Erase operation selected
0 = Erase operation not selected

PGM — Program Control Bit

This read/write bit configures the memory for program operation. PGM is interlocked with the ERASE bit such that both bits cannot be equal to 1 or set to 1 at the same time.

1 = Program operation selected
0 = Program operation not selected

2.5.3 FLASH Page Erase Operation

Use the following procedure to erase a page of FLASH memory. A page consists of 512 consecutive bytes starting from addresses $X000, $X200, $X400, $X600, $X800, $XA00, $XC00, or $XE00. *The 48-byte user interrupt vectors cannot be erased by the page erase operation because of security reasons. Mass erase is required to erase this page.*

1. Set the ERASE bit and clear the MASS bit in the FLASH control register.
2. Write any data to any FLASH location within the page address range desired.
3. Wait for a time, t_{nvs} (5 µs).
4. Set the HVEN bit.
5. Wait for a time t_{erase} (20 ms).
6. Clear the ERASE bit.
7. Wait for a time, t_{nvh} (5 µs).

Table 4-1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
DBNZ *opr,rel* DBNZA *rel* DBNZX *rel* DBNZ *opr*,X,*rel* DBNZ X,*rel* DBNZ *opr*,SP,*rel*	Decrement and Branch if Not Zero	A ← (A)–1 or M ← (M)–1 or X ← (X)–1 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 4 + *rel* ? (result) ≠ 0	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6
DEC *opr* DECA DECX DEC *opr*,X DEC ,X DEC *opr*,SP	Decrement	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1 M ← (M) – 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	4 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		7
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
INC *opr* INCA INCX INC *opr*,X INC ,X INC *opr*,SP	Increment	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1 M ← (M) + 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff 	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff 	4 5 6 5 4
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X LDA *opr*,SP LDA *opr*,SP	Load A from M	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9EE6 9ED6	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LDHX #*opr* LDHX *opr*	Load H:X from M	H:X ← (M:M + 1)	0	–	–	↕	↕	–	IMM DIR	45 55	ii jj dd	3 4

Chapter 5
Oscillator (OSC)

5.1 Introduction

The oscillator module consist of three types of oscillator circuits:

- Internal oscillator
- RC oscillator
- 1 MHz to 8 MHz crystal (x-tal) oscillator

The reference clock for the CGM and other MCU sub-systems is selected by programming the mask option register located at $FFCF.

The reference clock for the timebase module (TBM) is selected by the two bits, OSCCLK1 and OSCCLK0, in the CONFIG2 register.

The internal oscillator runs continuously after a POR or reset, and is always available. The RC and crystal oscillator cannot run concurrently; one is disabled while the other is selected; because the RC and x-tal circuits share the same OSC1 pin.

NOTE

The oscillator circuits are powered by the on-chip V_{REG} regulator, therefore, the output swing on OSC1 and OSC2 is from V_{SS} to V_{REG}.

Figure 5-1. shows the block diagram of the oscillator module.

5.2 Clock Selection

Reference clocks are selectable for the following sub-systems:

- CGMXCLK and CGMRCLK — Reference clock for clock generator module (CGM) and other MCU sub-systems other than TBM and COP. This is the main reference clock for the MCU.
- OSCCLK — Reference clock for timebase module (TBM).

At power-on, these events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables CGMOUT.
- Internal clocks to the CPU and modules are held inactive for 4096 ICLK cycles to allow stabilization of the oscillator.
- The pin is driven low during the oscillator stabilization time.
- The POR bit of the SIM reset status register (SRSR) is set and all other bits in the register are cleared.

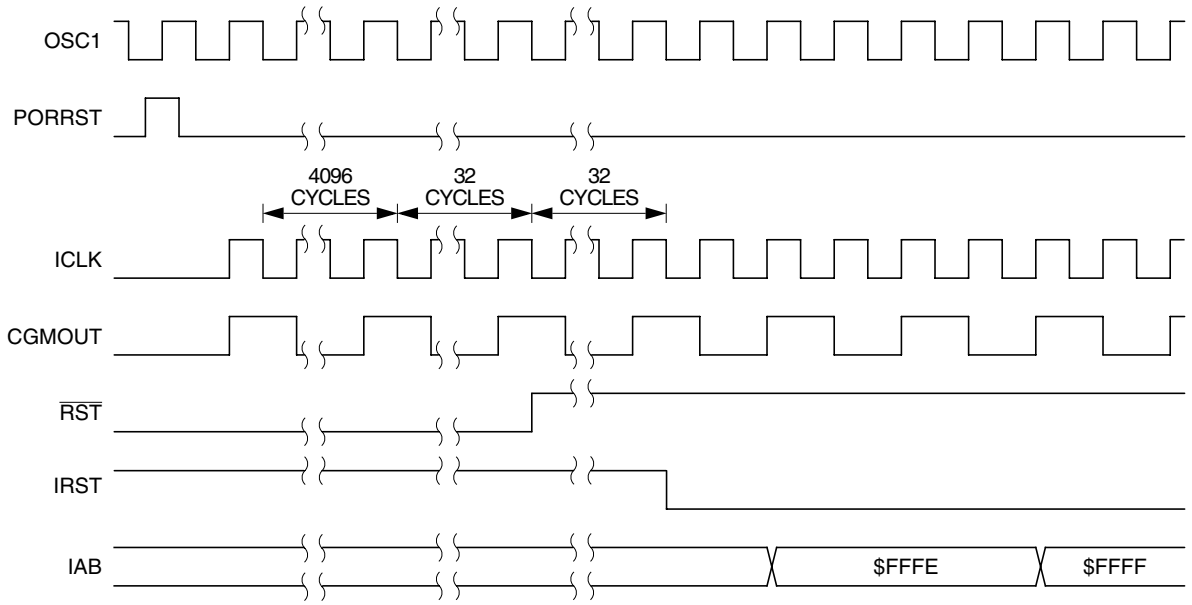

Figure 7-7. POR Recovery

7.3.2.2 Computer Operating Properly (COP) Reset

An input to the SIM is reserved for the COP reset signal. The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR). The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

To prevent a COP module timeout, write any value to location $FFFF. Writing to location $FFFF clears the COP counter and bits 12 through 5 of the SIM counter. The SIM counter output, which occurs at least every $2^{13} - 2^4$ ICLK cycles, drives the COP counter. The COP should be serviced as soon as possible out of reset to guarantee the maximum amount of time before the first timeout.

The COP module is disabled if the $\overline{\text{RST}}$ pin or the $\overline{\text{IRQ1}}$ pin is held at V_{TST} while the MCU is in monitor mode. The COP module can be disabled only through combinational logic conditioned with the high voltage signal on the $\overline{\text{RST}}$ or the $\overline{\text{IRQ1}}$ pin. This prevents the COP from becoming disabled as a result of external noise. During a break state, V_{TST} on the $\overline{\text{RST}}$ pin disables the COP module.

7.3.2.3 Illegal Opcode Reset

The SIM decodes signals from the CPU to detect illegal instructions. An illegal instruction sets the ILOP bit in the SIM reset status register (SRSR) and causes a reset.

7.7.2 SIM Reset Status Register

This register contains six flags that show the source of the last reset provided all previous reset status bits have been cleared. Clear the SIM reset status register by reading it. A power-on reset sets the POR bit and clears all other bits in the register.

The register is initialized on power up with the POR bit set and all other bits cleared. During a POR or any other internal reset, the $\overline{RST}$ pin is pulled low. After the pin is released, it will be sampled 32 CGMXCLK cycles later. If the pin is not above V_{IH} at this time, then the PIN bit may be set, in addition to whatever other bits are set.

Address: $FE01

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
Write:								
Reset:	1	0	0	0	0	0	0	0

= Unimplemented

Figure 7-21. SIM Reset Status Register (SRSR)

POR — Power-On Reset Bit
1 = Last reset caused by POR circuit
0 = Read of SRSR

PIN — External Reset Bit
1 = Last reset caused by external reset pin ($\overline{RST}$)
0 = POR or read of SRSR

COP — Computer Operating Properly Reset Bit
1 = Last reset caused by COP counter
0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit
1 = Last reset caused by an illegal opcode
0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (opcode fetches only)
1 = Last reset caused by an opcode fetch from an illegal address
0 = POR or read of SRSR

MODRST — Monitor Mode Entry Module Reset Bit
1 = Last reset caused by monitor mode entry when vector locations $FFFE and $FFFF are $FF after POR while $\overline{IRQ1}$ = V_{DD}
0 = POR or read of SRSR

LVI — Low-Voltage Inhibit Reset Bit
1 = Last reset caused by the LVI circuit
0 = POR or read of SRSR

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

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.

During the software execution, it does not consume any dedicated RAM location, the run-time heap will extend the system stack, all other RAM location will not be affected.

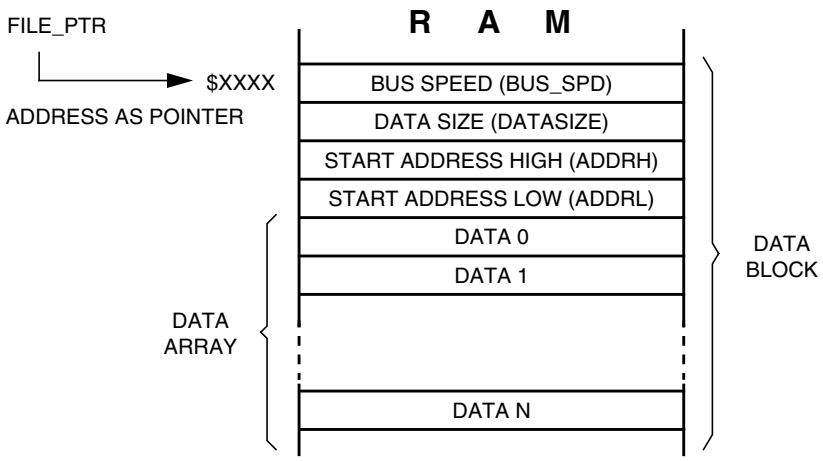

Figure 8-9. Data Block Format for ROM-Resident Routines

The control and data bytes are described below.

- **Bus speed** — This one byte indicates the operating bus speed of the MCU. The value of this byte should be equal to 4 times the bus speed. E.g., for a 4MHz bus, the value is 16 ($10). This control byte is useful where the MCU clock source is switched between the PLL clock and the crystal clock.
- **Data size** — This one byte indicates the number of bytes in the data array that are to be manipulated. The maximum data array size is 255. ERARNGE and MON_ERARNGE routines do not manipulate a data array, thus, this data size byte has no meaning.
- **Start address** — These two bytes, high byte followed by low byte, indicate the start address of the FLASH memory to be manipulated.
- **Data array** — This data array contains data that are to be manipulated. Data in this array are programmed to FLASH memory by the programming routines, PRGRNGE and MON_PRGRNGE. For the read routine, LDRNGE, data is read from FLASH and stored in this array.

8.5.1 PRGRNGE

PRGRNGE is used to program a range of FLASH locations with data loaded into the data array.

Table 8-11. PRGRNGE Routine

Routine Name	PRGRNGE
Routine Description	Program a range of locations
Calling Address	$FC34
Stack Used	15 bytes
Data Block Format	Bus speed (BUS_SPD) Data size (DATASIZE) Start address high (ADDRH) Start address (ADDRL) Data 1 (DATA1) : Data N (DATAN)

The start location of the FLASH to be programmed is specified by the address ADDRH:ADDRL and the number of bytes from this location is specified by DATASIZE. The maximum number of bytes that can be programmed in one routine call is 255 bytes (max. DATASIZE is 255).

ADDRH:ADDRL do not need to be at a page boundary, the routine handles any boundary misalignment during programming. A check to see that all bytes in the specified range are erased is not performed by this routine prior programming. Nor does this routine do a verification after programming, so there is no return confirmation that programming was successful. User must assure that the range specified is first erased.

The coding example below is to program 64 bytes of data starting at FLASH location $EE00, with a bus speed of 4.9152 MHz. The coding assumes the data block is already loaded in RAM, with the address pointer, FILE_PTR, pointing to the first byte of the data block.

The coding example below is to perform a page erase, from $EE00–$EFFF. The Initialization subroutine is the same as the coding example for PRGRNGE (see 8.5.1 PRGRNGE).

```
ERARNGE         EQU     $FCE4
MAIN:
        BSR     INITIALISATION
        :
        :
        LDHX    #FILE_PTR
        JSR     ERARNGE
        :
```

8.5.3 LDRNGE

LDRNGE is used to load the data array in RAM with data from a range of FLASH locations.

Table 8-13. LDRNGE Routine

Routine Name	LDRNGE
Routine Description	Loads data from a range of locations
Calling Address	$FC00
Stack Used	7 bytes
Data Block Format	Bus speed (BUS_SPD) Data size (DATASIZE) Starting address (ADDRH) Starting address (ADDRL) Data 1 : Data N

The start location of FLASH from where data is retrieved is specified by the address ADDRH:ADDRL and the number of bytes from this location is specified by DATASIZE. The maximum number of bytes that can be retrieved in one routine call is 255 bytes. The data retrieved from FLASH is loaded into the data array in RAM. Previous data in the data array will be overwritten. User can use this routine to retrieve data from FLASH that was previously programmed.

The coding example below is to retrieve 64 bytes of data starting from $EE00 in FLASH. The Initialization subroutine is the same as the coding example for PRGRNGE (see 8.5.1 PRGRNGE).

```
LDRNGE          EQU     $FC00
MAIN:
        BSR     INITIALIZATION
        :
        :
        LDHX    #FILE_PTR
        JSR     LDRNGE
        :
```

12.5.3.2 Character Reception

During an SCI reception, the receive shift register shifts characters in from the RxD pin. The SCI data register (IRSCDR) is the read-only buffer between the internal data bus and the receive shift register.

After a complete character shifts into the receive shift register, the data portion of the character transfers to the IRSCDR. The SCI receiver full bit, SCRF, in IRSCI status register 1 (IRSCS1) becomes set, indicating that the received byte can be read. If the SCI receive interrupt enable bit, SCRIE, in IRSCC2 is also set, the SCRF bit generates a receiver CPU interrupt request.

12.5.3.3 Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see Figure 12-9):

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

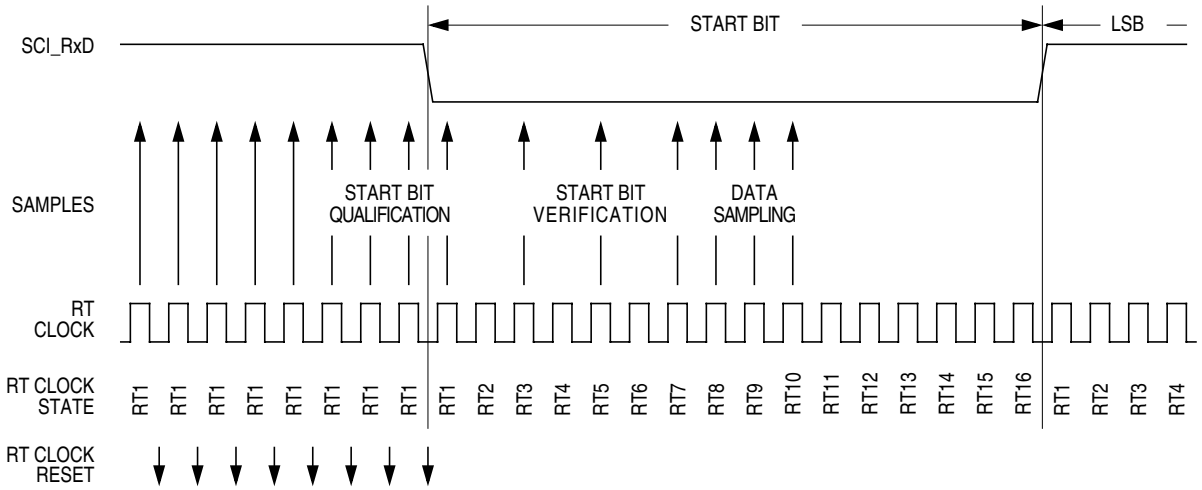

Figure 12-9. Receiver Data Sampling

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 12-2 summarizes the results of the start bit verification samples.

Table 12-2. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1

NOTE

Writing to the TE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.

RE — Receiver Enable Bit

Setting this read/write bit enables the receiver. Clearing the RE bit disables the receiver but does not affect receiver interrupt flag bits. Reset clears the RE bit.

1 = Receiver enabled
0 = Receiver disabled

NOTE

Writing to the RE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.

RWU — Receiver Wakeup Bit

This read/write bit puts the receiver in a standby state during which receiver interrupts are disabled. The WAKE bit in IRSCC1 determines whether an idle input or an address mark brings the receiver out of the standby state and clears the RWU bit. Reset clears the RWU bit.

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

Setting and then clearing this read/write bit transmits a break character followed by a logic 1. The logic 1 after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no logic 1s between them. Reset clears the SBK bit.

1 = Transmit break characters
0 = No break characters being transmitted

NOTE

Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK before the preamble begins causes the SCI to send a break character instead of a preamble.

12.9.3 IRSCI Control Register 3

IRSCI control register 3:

- Stores the ninth SCI data bit received and the ninth SCI data bit to be transmitted
- Enables the following interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts

Address: $0042

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
Write:								
Reset:	U	U	0	0	0	0	0	0

= Unimplemented U = Unaffected

Figure 12-14. IRSCI Control Register 3 (IRSCC3)

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the transmit data register in break mode does not initiate a transmission nor is this data transferred into the shift register. Therefore, a write to the SPDR in break mode with the BCFE bit cleared has no effect.

13.12 I/O Signals

The SPI module has five I/O pins and shares four of them with a parallel I/O port. They are:

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select
- CGND — Clock ground (internally connected to V_{SS})

The SPI has limited inter-integrated circuit (I^2C) capability (requiring software support) as a master in a single-master environment. To communicate with I^2C peripherals, MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In I^2C communication, the MOSI and MISO pins are connected to a bidirectional pin from the I^2C peripheral and through a pullup resistor to V_{DD}.

13.12.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmits serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic 0 and its $\overline{SS}$ pin is at logic 0. To support a multiple-slave system, a logic 1 on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

When enabled, the SPI controls data direction of the MISO pin regardless of the state of the data direction register of the shared I/O port.

13.12.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits serial data. In full-duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

When enabled, the SPI controls data direction of the MOSI pin regardless of the state of the data direction register of the shared I/O port.

13.12.3 SPSCK (Serial Clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SPSCK pin is the clock output. In a slave MCU, the SPSCK pin is the clock input. In full-duplex operation, the master and slave MCUs exchange a byte of data in eight serial clock cycles.

When enabled, the SPI controls data direction of the SPSCK pin regardless of the state of the data direction register of the shared I/O port.

The ADC conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is either the bus clock or CGMXCLK and is selectable by the ADICLK bit located in the ADC clock register. The divide ratio is selected by the ADIV[2:0] bits.

For example, if a 4MHz CGMXCLK is selected as the ADC input clock source, with a divide-by-four prescale, and the bus speed is set at 2MHz:

$$\text{Conversion time} = \frac{16 \text{ to} 17 \text{ ADC cycles}}{4\text{MHz} \div 4} = 16 \text{ to } 17\ \mu\text{s}$$

$$\text{Number of bus cycles} = 16\ \mu\text{s} \times 2\text{MHz} = 32 \text{ to } 34 \text{ cycles}$$

NOTE

The ADC frequency must be between f_{ADIC} minimum and f_{ADIC} maximum to meet A/D specifications. (See 22.5 5V DC Electrical Characteristics.).

Since an ADC cycle may be comprised of several bus cycles (four in the previous example) and the start of a conversion is initiated by a bus cycle write to the ADSCR, from zero to four additional bus cycles may occur before the start of the initial ADC cycle. This results in a fractional ADC cycle and is represented as the 17th cycle.

15.3.4 Continuous Conversion

In the continuous conversion mode, the ADC continuously converts the selected channel, filling the ADC data register with new data after each conversion. Data from the previous conversion will be overwritten whether that data has been read or not. Conversions will continue until the ADCO bit is cleared. The COCO bit is set after each conversion and can be cleared by writing to the ADC status and control register or reading of the ADRL0 data register.

15.3.5 Auto-Scan Mode

In auto-scan mode, the ADC input channel is selected by the value of the 2-bit up-counter, instead of the channel select bits, ADCH[4:0]. The value of the counter also defines the data register ADRLx to be used to store the conversion result. When ASCAN bit is set, a write to ADC status and control register (ADSCR) will reset the auto-scan up-counter and ADC conversion will start on the channel 0 up to the channel number defined by the integer value of AUTO[1:0]. After a channel conversion is completed, data is stored in ADRLx and the COCO-bit will be set. The counter value will be incremented by 1 and a new conversion will start. This process will continue until the counter value reaches the value of AUTO[1:0]. When this happens, it indicates that the current channel is the last channel to be converted. Upon the completion on the last channel, the counter value will not be incremented and no further conversion will be performed. To start another auto-scan cycle, a write to ADSCR must be performed.

NOTE

The system only provides 8-bit data storage in auto-scan code, user must clear MODE[1:0] bits to select 8-bit truncation mode before entering auto-scan mode.

It is recommended that user should disable the auto-scan function before switching channel and also before entering STOP mode.

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

PTD[7:0] — Port D Data Bits

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

KBI7–KBI0 — Keyboard Interrupt Inputs

The keyboard interrupt enable bits, KBIE[7:0], in the keyboard interrupt enable register (KBIER), enable the port D pins as external interrupt pins. See Chapter 18 Keyboard Interrupt Module (KBI).

16.5.2 Data Direction Register D (DDRD)

Data direction register D determines whether each port D pin is an input or an output. Writing a logic 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a logic 0 disables the output buffer.

Address: $0007

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Reset:	0	0	0	0	0	0	0	0

Figure 16-13. Data Direction Register D (DDRD)

DDRD[7:0] — Data Direction Register D Bits

These read/write bits control port D data direction. Reset clears DDRD[7:0], configuring all port D pins as inputs.

1 = Corresponding port D pin configured as output
0 = Corresponding port D pin configured as input

NOTE

Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1.

Figure 16-14 shows the port D I/O logic.

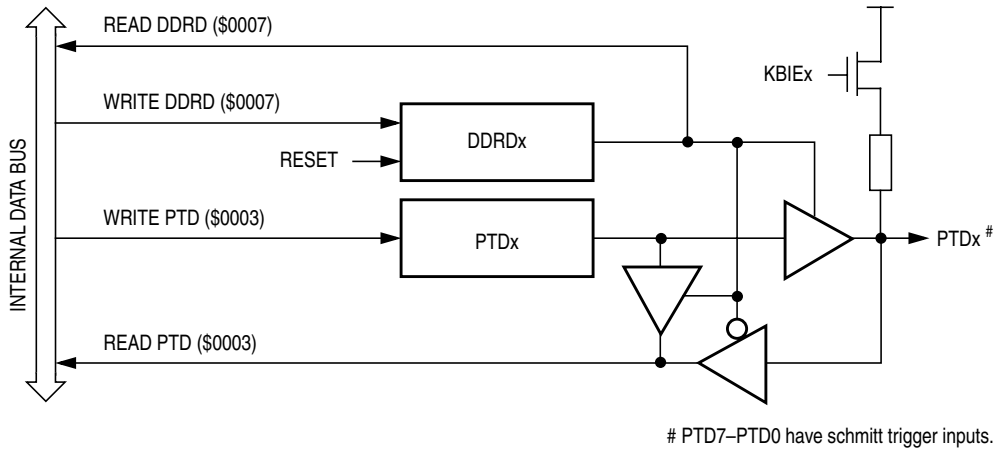

Figure 16-14. Port D I/O Circuit

When bit DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 16-5 summarizes the operation of the port D pins.

Table 16-5. Port D Pin Functions

DDRD Bit	PTD Bit	I/O Pin Mode	Accesses to DDRD	Accesses to PTD	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRD[7:0]	Pin	PTD[7:0](3)
1	X	Output	DDRD[7:0]	PTD[7:0]	PTD[7:0]

1. X = don't care.
2. Hi-Z = high impedance.
3. Writing affects data register, but does not affect input.

Chapter 18
Keyboard Interrupt Module (KBI)

18.1 Introduction

The keyboard interrupt module (KBI) provides eight independently maskable external interrupts which are accessible via PTD0–PTD7. When a port pin is enabled for keyboard interrupt function, an internal 30kΩ pullup device is also enabled on the pin.

18.2 Features

Features of the keyboard interrupt module include the following:

- Eight keyboard interrupt pins with pullup devices
- Separate keyboard interrupt enable bits and one keyboard interrupt mask
- Programmable edge-only or edge- and level- interrupt sensitivity
- Exit from low-lower modes

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:						ACKK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Interrupt Enable Register (KBIER)	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-1. KBI I/O Register Summary

18.3 I/O Pins

The eight keyboard interrupt pins are shared with standard port I/O pins. The full name of the KBI pins are listed in Table 18-1. The generic pin name appear in the text that follows.

Table 18-1. Pin Name Conventions

KBI Generic Pin Name	Full MCU Pin Name	Pin Selected for KBI Function by KBIEx Bit in KBIER
KBI0–KBI7	PTD0/KBI0–PTD7/KBI7	KBIE0–KBIE7

20.6 Low-Power Modes

The STOP and WAIT instructions put the MCU in low power-consumption standby modes.

20.6.1 Wait Mode

If enabled, the LVI module remains active in wait mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of wait mode.

20.6.2 Stop Mode

If enabled in stop mode (LVISTOP = 1), the LVI module remains active in stop mode. If enabled to generate resets (LVIRSTD = 0), the LVI module can generate a reset and bring the MCU out of stop mode.