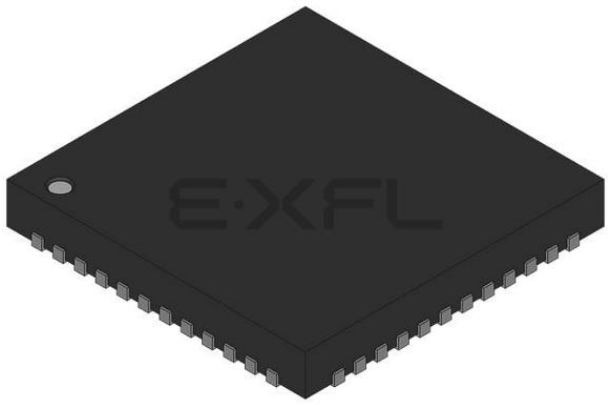


Welcome to E-XFL.COM

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	SPI, USB
Peripherals	LED, LVD, POR, PWM
Number of I/O	29
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	48-VFQFN Exposed Pad
Supplier Device Package	48-QFN (7x7)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mchc908jw32fc

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

MC68HC908JW32

Data Sheet

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://www.freescale.com

- External asynchronous interrupt pin with internal pull-up ($\overline{IRQ}$)
- 48-pin quad flat non-leaded package (QFN)

1.3 MCU Block Diagram

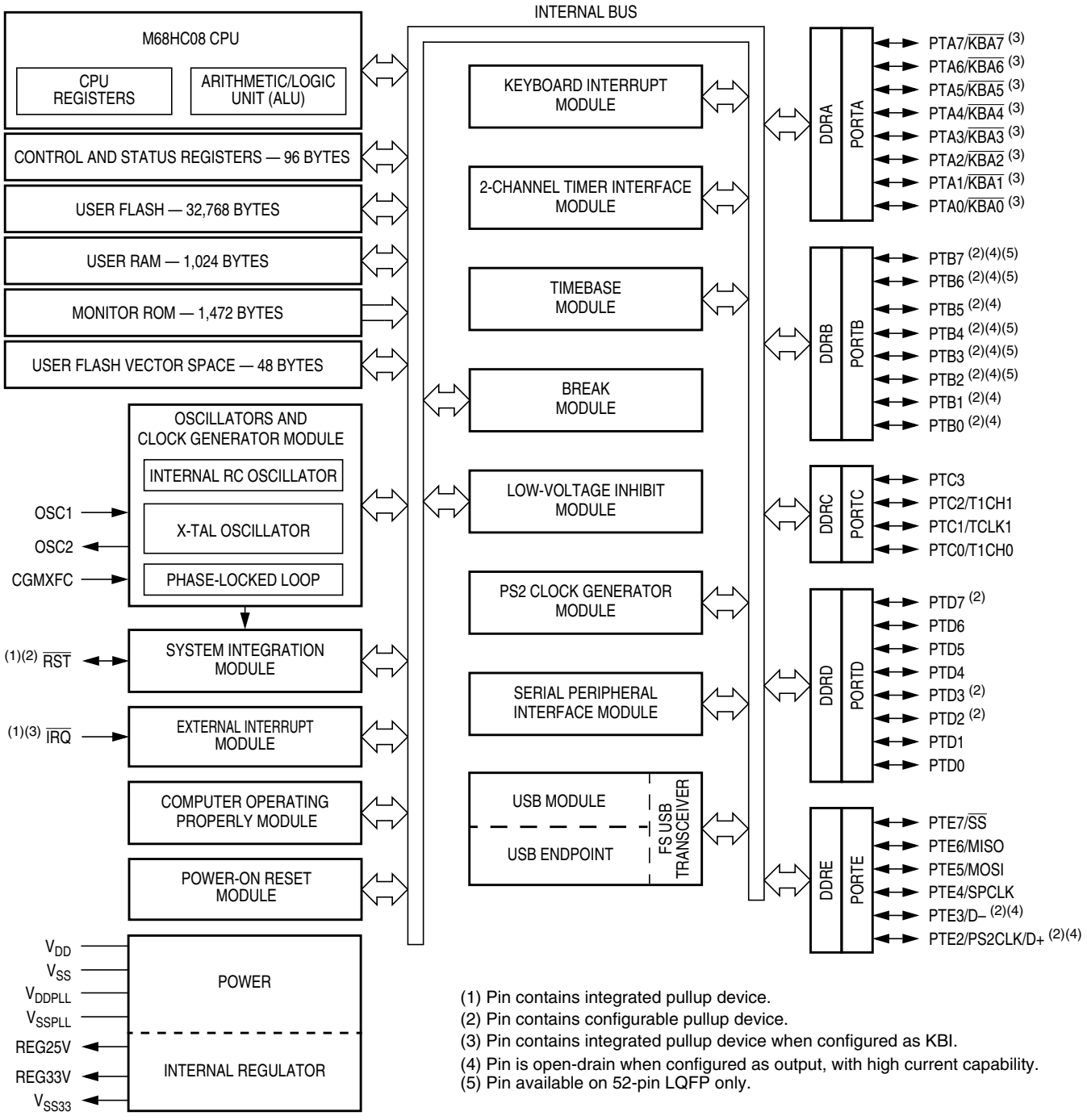


Figure 1-1. MC68HC908JW32 Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000D	Timer 1 Counter Register Low (T1CNTL)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000E	Timer 1 Counter Modulo Register High (T1MODH)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$000F	Timer 1 Counter Modulo Register Low (T1MODL)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0010	Timer 1 Channel 0 Status and Control Register (T1SC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0011	Timer 1 Channel 0 Register High (T1CH0H)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	Indeterminate after reset							
$0012	Timer 1 Channel 0 Register Low (T1CH0L)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	Indeterminate after reset							
$0013	Timer 1 Channel 1 Status and Control Register (T1SC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0014	Timer 1 Channel 1 Register High (T1CH1H)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	Indeterminate after reset							
$0015	Timer 1 Channel 1 Register Low (T1CH1L)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	Indeterminate after reset							
$0016	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
$0017	Keyboard Interrupt Enable Register (KBIER)	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	Timebase Control Register (TBCR)	Read:	TBIF	TBR2	TBR1	TBR0	0	TBIE	TBON	R
		Write:					TACK			
		Reset:	0	0	0	0	0	0	0	0
$0019	PS2 Clock Generator Control and Satus Register (PS2CSR)	Read:	PSTATUS	PS2IF	PRE	CSEL1	CSEL0	PS2IEN	CLKEN	PS2EN
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected by reset

Figure 2-2. Control, Status, and Data Registers (Sheet 2 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0051	USB Control Register (USBCR)	Read:	USBEN	USBCLKEN	TFC4IE	TFC3IE	TFC2IE	TFC1IE	TFC0IE	0
		Write:								RESUME
		Reset:	0	0	0	0	0	0	0	0
$0052	USB Status Register (USBSR)	Read:	CONFIG		SETUP	SOF	CONFIG_CHG	USBRST	RESUMEF	SUSPND
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0053	USB Status Interrupt Mask Register (USIMR)	Read:		0	SETUPIE	SOFIE	CONFIG_ CHGIE	USBRE- SETIE	RESUME- FIE	SUSPNDIE
		Write:		EP0_STALL						
		Reset:	0	0	0	0	0	0	0	0
$0054	USB EPO Control/Status Register (UEP0CSR)	Read:	DSIZE3_OUT	DSIZE2_OUT	DSIZE1_OUT	DSIZE0_OUT	DVALID_IN	TFRC_IN	DVALID_OUT	TFRC_OUT
		Write:	DSIZE3_IN	DSIZE2_IN	DSIZE1_IN	DSIZE0_IN				
		Reset:	0	0	0	0	0	0	0	0
$0055	USB EP1 Control/Status Register (UEP1CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0056	USB EP2 Control/Status Register (UEP2CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0057	USB EP3 Control/Status Register (UEP3CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0058	USB EP4 Control/Status Register (UEP4CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0059	USB EP1 Data Size Register (UEP1DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005A	USB EP2 Data Size Register (UEP2DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005B	USB EP3 Data Size Register (UEP3DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005C	USB EP4 Data Size Register (UEP4DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005D	USB EP 1/2 Base Pointer Register (UEP12BPR)	Read:		BASE22	BASE21	BASE20		BASE12	BASE11	BASE10
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected by reset

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 7)

This program sequence is repeated throughout the memory until all data is programmed.

NOTE

The time between each FLASH address change (step 6 to step 6), or the time between the last FLASH addressed programmed to clearing the PGM bit (step 6 to step 9), must not exceed the maximum programming time, t_{prog} max.

NOTE

Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order shown, other unrelated operations may occur between the steps.

2.5.6 FLASH Protection

Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made to protect pages of memory from unintentional erase or program operations due to system malfunction. This protection is done by use of a FLASH block protect register (FLBPR). The FLBPR determines the range of the FLASH memory which is to be protected. The range of the protected area starts from a location defined by FLBPR and ends to the bottom of the FLASH memory ($FFFF). When the memory is protected, the HVEN bit cannot be set in either erase or program operations.

NOTE

The 48 bytes of user interrupt vectors ($FFD0–$FFFF) are always protected, regardless of the value in the FLASH block protect register. A mass erase is required to erase these locations.

Table 4-1. Instruction Set Summary (Sheet 6 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		9
TAP	Transfer A to CCR	CCR ← (A)	↕	↕	↕	↕	↕	↕	INH	84		2
TAX	Transfer A to X	X ← (A)	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to A	A ← (CCR)	–	–	–	–	–	–	INH	85		1
TST *opr* TSTA TSTX TST *opr*,X TST ,X TST *opr*,SP	Test for Negative or Zero	(A) – $00 or (X) – $00 or (M) – $00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	3 1 1 3 2 4
TSX	Transfer SP to H:X	H:X ← (SP) + 1	–	–	–	–	–	–	INH	95		2
TXA	Transfer X to A	A ← (X)	–	–	–	–	–	–	INH	9F		1
TXS	Transfer H:X to SP	(SP) ← (H:X) – 1	–	–	–	–	–	–	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	I bit ← 0; Inhibit CPU clocking until interrupted	–	–	0	–	–	–	INH	8F		1

A	Accumulator
C	Carry/borrow bit
CCR	Condition code register
dd	Direct address of operand
dd rr	Direct address of operand and relative offset of branch instruction
DD	Direct to direct addressing mode
DIR	Direct addressing mode
DIX+	Direct to indexed with post increment addressing mode
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing
EXT	Extended addressing mode
ff	Offset byte in indexed, 8-bit offset addressing
H	Half-carry bit
H	Index register high byte
hh ll	High and low bytes of operand address in extended addressing
I	Interrupt mask
ii	Immediate operand byte
IMD	Immediate source to direct destination addressing mode
IMM	Immediate addressing mode
INH	Inherent addressing mode
IX	Indexed, no offset addressing mode
IX+	Indexed, no offset, post increment addressing mode
IX+D	Indexed with post increment to direct addressing mode
IX1	Indexed, 8-bit offset addressing mode
IX1+	Indexed, 8-bit offset, post increment addressing mode
IX2	Indexed, 16-bit offset addressing mode
M	Memory location
N	Negative bit
n	Any bit
opr	Operand (one or two bytes)
PC	Program counter
PCH	Program counter high byte
PCL	Program counter low byte
REL	Relative addressing mode
rel	Relative program counter offset byte
rr	Relative program counter offset byte
SP1	Stack pointer, 8-bit offset addressing mode
SP2	Stack pointer 16-bit offset addressing mode
SP	Stack pointer
U	Undefined
V	Overflow bit
X	Index register low byte
Z	Zero bit
&	Logical AND
|	Logical OR
⊕	Logical EXCLUSIVE OR
( )	Contents of
–( )	Negation (two's complement)
#	Immediate value
«	Sign extend
←	Loaded with
?	If
:	Concatenated with
↕	Set or cleared
—	Not affected

4.8 Opcode Map

See Table 4-2.

6.2.2 Clock Start-up from POR or LVI Reset

When the power-on reset module or the low-voltage inhibit module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after the 4096 CGMXCLK cycle POR timeout has completed. The $\overline{RST}$ pin is driven low by the SIM during this entire period. The IBUS clocks start upon completion of the timeout.

6.2.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. (See 6.6.2 Stop Mode.)

In wait mode, the CPU clocks are inactive. The SIM also produces two sets of clocks for other modules. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

6.3 Reset and System Initialization

The MCU has these reset sources:

- Power-on reset module (POR)
- External reset pin ($\overline{RST}$)
- Computer operating properly module (COP)
- Low-voltage inhibit module (LVI)
- Illegal opcode
- Illegal address
- Universal serial bus module (USB)

All of these resets produce the vector $FFFE:$FFFF ($FEFE:$FEFF in monitor mode) and assert the internal reset signal (IRST). IRST causes all registers to be returned to their default values and all modules to be returned to their reset states.

An internal reset clears the SIM counter (see 6.4 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See 6.7 SIM Registers.)

6.3.1 External Pin Reset

The $\overline{RST}$ pin circuit includes an internal pull-up device. Pulling the asynchronous $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for at least the minimum t_{RL} time and no other reset sources are present. See Table 6-2 for details. Figure 6-4 shows the relative timing.

Table 6-2. Reset Recovery

Reset Recovery Type	Actual Number of Cycles
POR/LVI	4163 (4096 + 64 + 3)
All others	67 (64 + 3)

6.6.2 Stop Mode

In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

The SIM disables the clock generator module output (CGMOUT) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the configuration register 1 (CONFIG1). If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode.

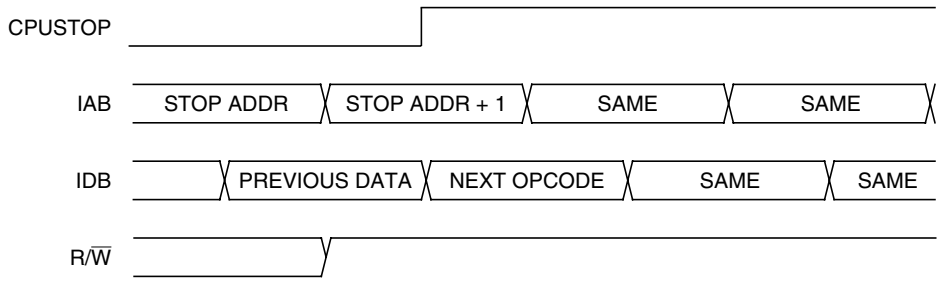
NOTE

External crystal applications should use the full stop recovery time by clearing the SSREC bit.

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 6-18 shows stop mode entry timing.

NOTE

To minimize stop current, all pins configured as inputs should be driven to a logic 1 or logic 0.



NOTE: Previous data can be operand data or the STOP opcode, depending on the last instruction.

Figure 6-18. Stop Mode Entry Timing

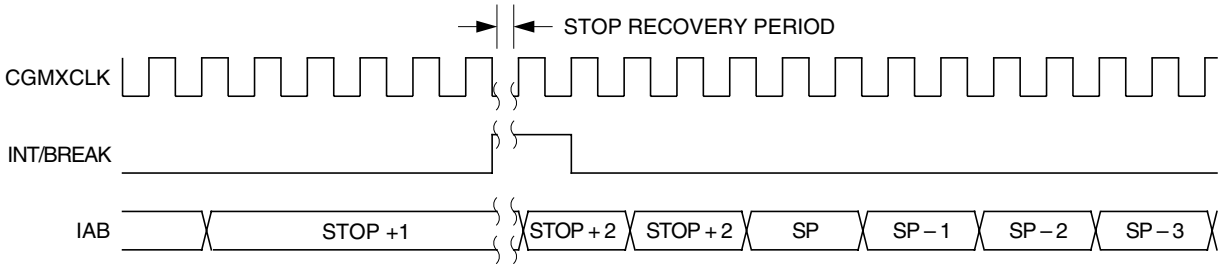


Figure 6-19. Stop Mode Recovery from Interrupt

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all ROM locations and execute code from ROM. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. (See Figure 7-7.)

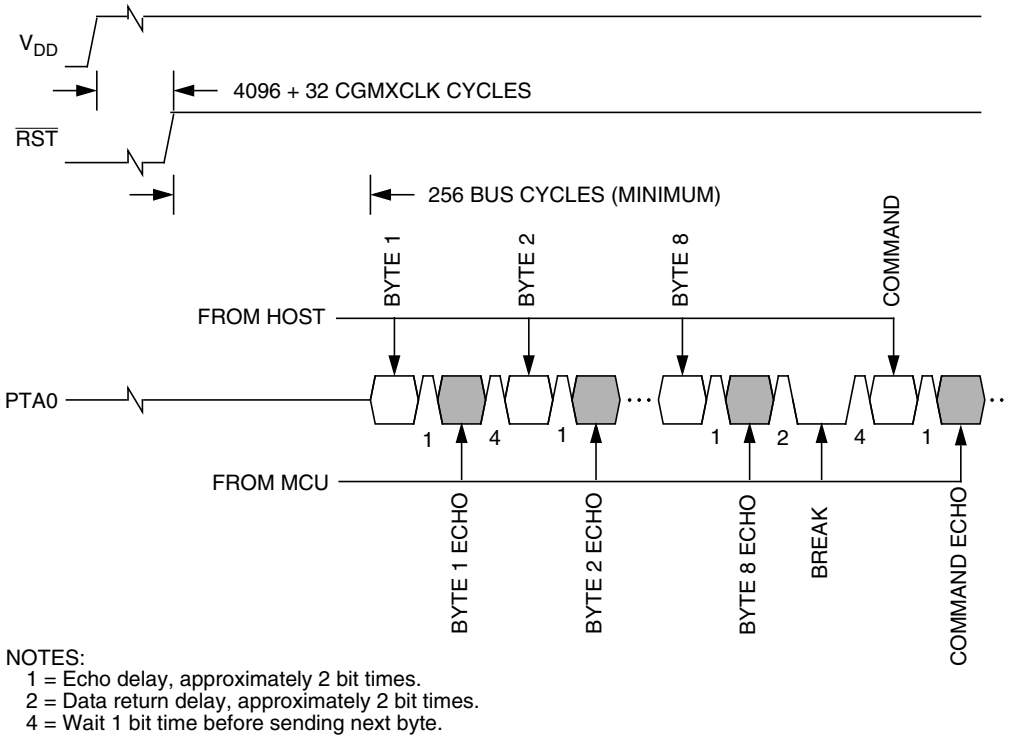


Figure 7-7. Monitor Mode Entry Timing

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 ROM location returns an invalid value and trying to execute code from ROM 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 ROM can be accessed.

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. Routines ERARNGE and MON_ERARNGE 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, MON_PRGRNGE. For the read routines: LDRNGE and data is read from FLASH and stored in this array.

7.5.1 PRGRNGE

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

Table 7-11. PRGRNGE Routine

Routine Name	PRGRNGE
Routine Description	Program a range of locations
Calling Address	$FE10
Stack Used	16 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.

```
                ORG     RAM
        :
FILE_PTR:
BUS_SPD         DS.B    1       ; Indicates 4x bus frequency
DATASIZE        DS.B    1       ; Data size to be programmed
START_ADDR      DS.W    1       ; FLASH start address
DATAARRAY       DS.B    64      ; Reserved data array

PRGRNGE         EQU     $FE10
FLASH_START     EQU     $EE00

                ORG     FLASH
INITIALISATION:
        MOV     #20,    BUS_SPD
        MOV     #64,    DATASIZE
        LDHX    #FLASH_START
        STHX    START_ADDR
        RTS
MAIN:
        BSR     INITIALISATION
        :
        :
        LDHX    #FILE_PTR
        JSR     PRGRNGE
```

7.5.2 ERARNGE

ERARNGE is used to erase a range of locations in FLASH.

Table 7-12. ERARNGE Routine

Routine Name	ERARNGE
Routine Description	Erase a page or the entire array
Calling Address	$FE13
Stack Used	10 bytes
Data Block Format	Bus speed (BUS_SPD) Data size (DATASIZE) Starting address (ADDRH) Starting address (ADDRL)

There are two sizes of erase ranges: a page or the entire array. The ERARNGE will erase the page (512 consecutive bytes) in FLASH specified by the address ADDRH:ADDRL. This address can be any address within the page. Calling ERARNGE with ADDRH:ADDRL equal to $FFFF will erase the entire FLASH array (mass erase). Therefore, care must be taken when calling this routine to prevent an accidental mass erase.

The ERARNGE routine do not use a data array. The DATASIZE byte is a dummy byte that is also not used.

10.5.3 Transmission Format When CPHA = 1

Figure 10-6 shows an SPI transmission in which CPHA is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 10.7.2 Mode Fault Error.) When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

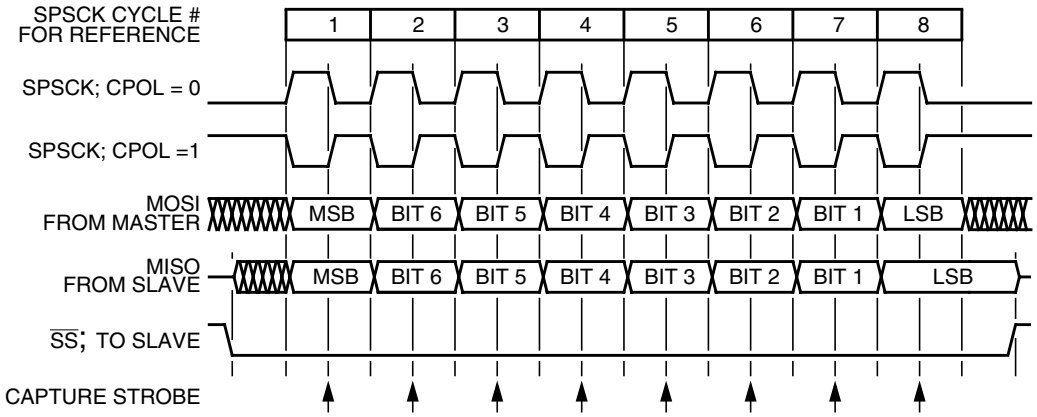


Figure 10-6. Transmission Format (CPHA = 1)

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SPSCK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transmission.

10.5.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCK signal. When CPHA = 0, the SPSCK signal remains inactive for the first half of the first SPSCK cycle. When CPHA = 1, the first SPSCK cycle begins with an edge on the SPSCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See Figure 10-7.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. SPSCK edges occur halfway through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR occurs relative to the slower SPSCK. This uncertainty causes the variation in the initiation delay shown in Figure 10-7. This delay is no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

SPWOM — SPI Wired-OR Mode Bit

This read/write bit disables the pullup devices on pins SPSCK, MOSI, and MISO so that those pins become open-drain outputs.

1 = Wired-OR SPSCK, MOSI, and MISO pins
0 = Normal push-pull SPSCK, MOSI, and MISO pins

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See 10.9 Resetting the SPI.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

This read/write bit enables CPU interrupt requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register. Reset clears the SPTIE bit.

1 = SPTE CPU interrupt requests enabled
0 = SPTE CPU interrupt requests disabled

10.13.2 SPI Status and Control Register

The SPI status and control register contains flags to signal these conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received (overflow error)
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)
- Transmit data register empty

The SPI status and control register also contains bits that perform these functions:

- Enable error interrupts
- Enable mode fault error detection
- Select master SPI baud rate

Address $004D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
Write:								
Reset:	0	0	0	0	1	0	0	0

= Unimplemented

Figure 10-14. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a CPU interrupt request if the SPRIE bit in the SPI control register is set also. During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Reset clears the SPRF bit.

1 = Receive data register full
0 = Receive data register not full

11.3.4.1 Configuration Process

All USB devices must be configured before used. The host will configure the device according to the configuration process defined by the USB specification 2.0 Chapter 9. During the process most of the USB commands issued by the host are responded automatically except GET_DESCRIPTOR, SYNC_FRAME, vendor specific and class specific commands where user interaction is required. These are known as the user commands. The number of configurations and interfaces supported is limited by the module. This module can support a single configuration and maximum of two interfaces. No alternate setting is allowed.

Upon the reception of the user commands, no module level decoding is done instead user is notified by the SETUP flag and TFRC_OUT flag. User can then decode the command through the dedicated 8-byte endpoint 0 buffer. For instance, when a valid GET_DESCRIPTOR command is detected, user is notified by the SETUP, TFRC_OUT flag and DVALID_OUT flag. User should decode the command via the 8-byte endpoint 0 OUT buffer. Corresponding return descriptor is written to the endpoint 0 IN buffer 8 bytes at a time. By setting the DVALID_IN bit, the data is sent to the host in the next IN packet. Otherwise, the module will return NAK to all IN packet. If ACK is not returned from the host, the data is re-sent automatically in the next IN packet until ACK is returned from the host, then transfer complete flag TFRC_IN is set, the next 8 bytes of data can be written to the endpoint 0 IN buffer. The process continues until the requested descriptor is sent completely.

NOTE

Please note the module will return ACK to all valid SETUP packet. No software attention is required.

Endpoint 0 buffer and endpoint 0 data size register (DSIZE) will be updated on every incoming SETUP packet. However, SETUP or TFRC_OUT will not be set unless the SETUP packet is a valid GET_DESCRIPTOR, SYNC_FRAME or class/vendor specific SETUP command.

11.3.4.2 Control Endpoint 0

Endpoint 0 is always treated as control endpoint. It has eight bytes dedicated buffer for device transmit (IN packet) and eight bytes dedicated buffer for device receive (OUT packet). Most of the host requests is handled by the requestor processor excepts the class/vendor specified request, GET_DESCRIPTOR request and the SYNC_FRAME request. If the user is notified by the module about the arrival of such requests, user can decode the request command by reading the endpoint 0 data register.

The SETUP flag will be set if the 8-byte setup packet is received without CRC/Token/EOP error for Vendor/Class/SetDescriptor/SynchFrame commands only.

NOTE

For any OUT data received in the 8-byte endpoint OUT buffer, they are only valid until the start of any SETUP packet addressed to the device, even if the packet is corrupted the 8-byte OUT buffer may still be overwritten by this new SETUP packet. There is no indication of the corruption built into this module.

11.3.5 Endpoint Controller

The module has four independent endpoint controllers that managed the data transfer between CPU and the USB host. Each of these endpoint can be configured to either one of the two modes - bulk or interrupt.

Table 13-1. Port Control Register Bits Summary (Continued)

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
B	0	DDRB0	LED	POCR1 ($1A)	LEDB0	PTB0
			PULLUP	PULLCR ($3E)	PULL0EN	
	1	DDRB1	LED	POCR1 ($1A)	LEDB1	PTB1
			PULLUP	PULLCR ($3E)	PULL1EN	
	2	DDRB2	LED	POCR1 ($1A)	LEDB2	PTB2
			PULLUP	PULLCR ($3E)	PULL2EN	
	3	DDRB3	LED	POCR1 ($1A)	LEDB3	PTB3
			PULLUP	PULLCR ($3E)	PULL3EN	
	4	DDRB4	LED	POCR1 ($1A)	LEDB4	PTB4
			PULLUP	PULLCR ($3E)	PULL4EN	
	5	DDRB5	LED	POCR1 ($1A)	LEDB5	PTB5
			PULLUP	PULLCR ($3E)	PULL5EN	
	6	DDRB6	LED	POCR1 ($1A)	LEDB6	PTB6
			PULLUP	PULLCR ($3E)	PULL6EN	
	7	DDRB7	LED	POCR1 ($1A)	LEDB7	PTB7
			PULLUP	PULLCR ($3E)	PULL7EN	
C	0	DDRC0	TIM1	T1SC0 ($10)	ELS0B:ELS0A	PTC0/T1CH0
	1	DDRC1		T1SC ($0A)	PS[2:0]	PTC1/TCLK1
	2	DDRC2		T1SC1 ($13)	ELS1B:ELS1A	PTC2/T1CH1
	3	DDRC3	—	—	—	PTC3
D	0	DDRD0	—	—	—	PTD0
	1	DDRD1	—	—	—	PTD1
	2	DDRD2	PULLUP	POCR2 ($1B)	PTD2PD	PTD2
	3	DDRD3			PTD3PD	PTD3
	4	DDRD4	—	—	—	PTD4
	5	DDRD5				PTD5
	6	DDRD6				PTD6
	7	DDRD7	PULLUP	POCR2 ($1B)	PTD7PD	PTD7

Address:	$0005							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Reset:	0	0	0	0	0	0	0	0

Figure 13-6. Data Direction Register D (DDRD)

DDRB[7:0] — Data Direction Register B Bits

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

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

NOTE

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

Figure 13-7 shows the port B I/O circuit logic.

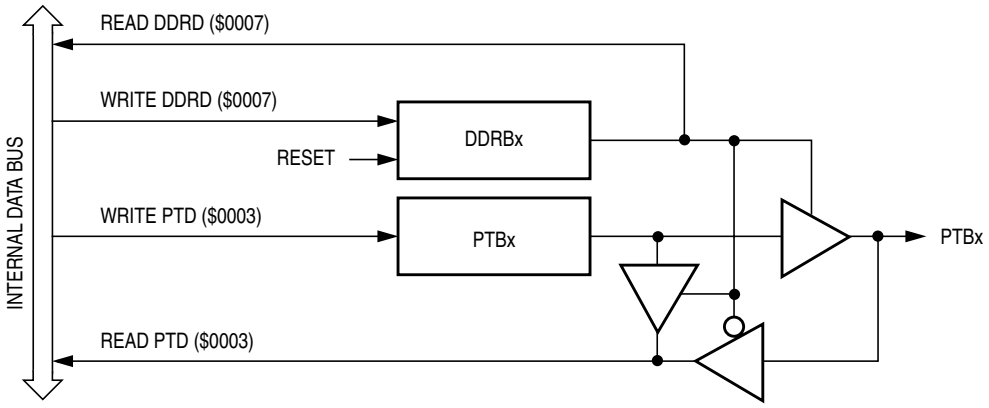


Figure 13-7. Port B I/O Circuit

When bit DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-3 summarizes the operation of the port B pins.

Table 13-3. Port B Pin Functions

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

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

Figure 13-13 shows the port D I/O circuit logic.

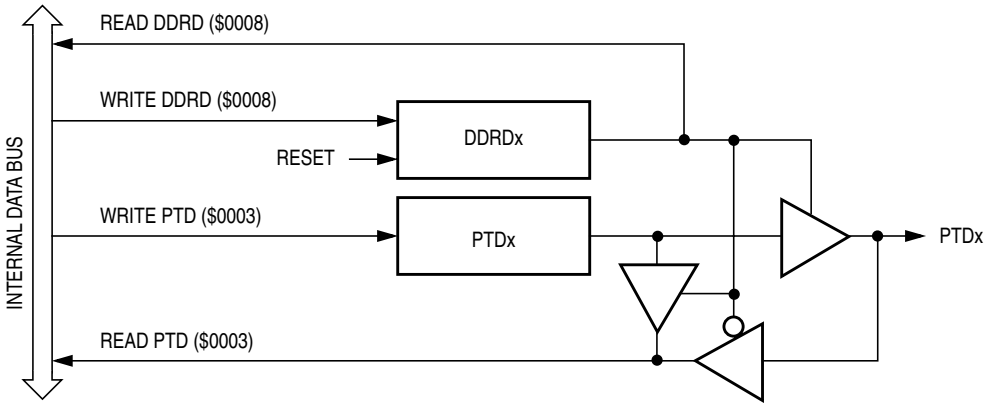


Figure 13-13. Port D I/O Circuit

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 13-6 summarizes the operation of the port D pins.

Table 13-6. 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.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic 1 may occur in any order.

If the MODEK bit is clear, the keyboard interrupt pin is falling-edge-sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

The keyboard flag bit (KEYF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYF bit is not affected by the keyboard interrupt mask bit (IMASKK) which makes it useful in applications where polling is preferred.

To determine the logic level on a keyboard interrupt pin, use the data direction register to configure the pin as an input and read the data register.

NOTE

Setting a keyboard interrupt enable bit (KBIEx) forces the corresponding keyboard interrupt pin to be an input, overriding the data direction register. However, the data direction register bit must be a logic 0 for software to read the pin.

15.4.1 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the pullup device to reach a logic 1. Therefore, a false interrupt can occur as soon as the pin is enabled.

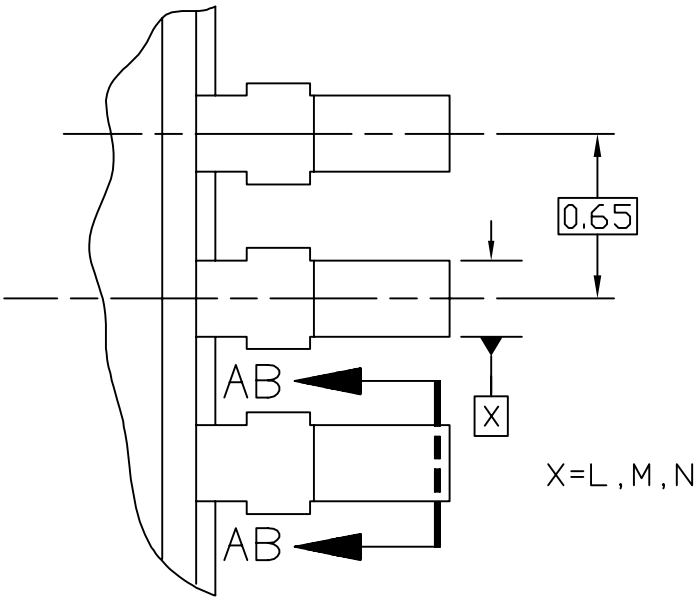
To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.
3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

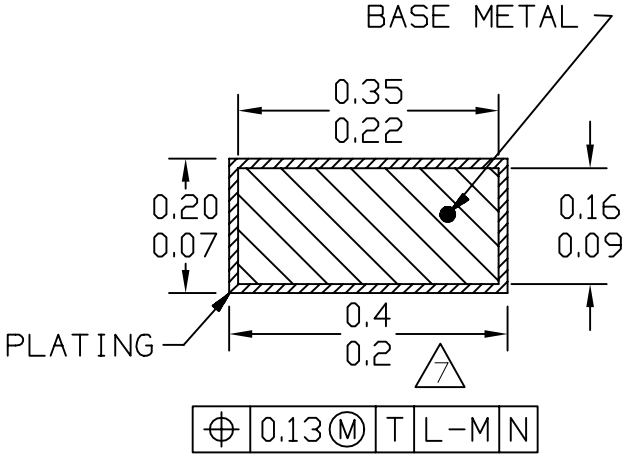
An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

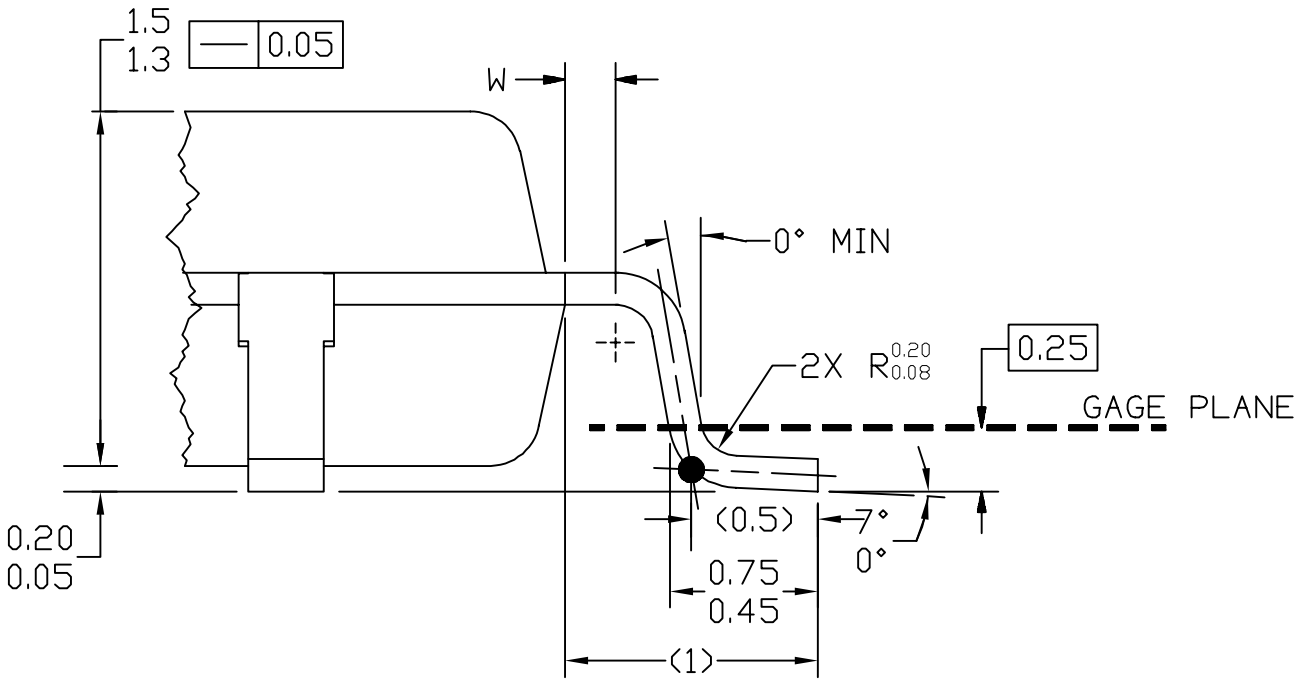
1. Configure the keyboard pins as outputs by setting the appropriate DDRA bits in data direction register A.
2. Write logic 1s to the appropriate port A data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.



VIEW Y



SECTION AB-AB
ROTATED 90° CLOCKWISE



VIEW AA

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE:	DOCUMENT NO: 98ASS23228W	REV: F
52LD TQFP 10 X 10 PKG, 0.65 PITCH, 1.4 THICK	CASE NUMBER: 848D-03	05 MAY 2005
	STANDARD: NON-JEDEC	