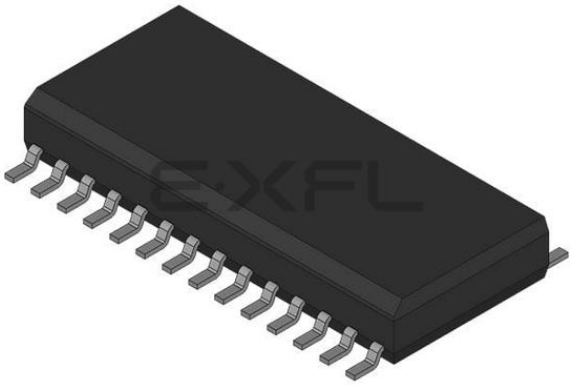


Welcome to E-XFL.COM

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	SCI
Peripherals	LED, LVD, POR, PWM
Number of I/O	23
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 13x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc908jl8cdwer

1.4 Pin Assignments

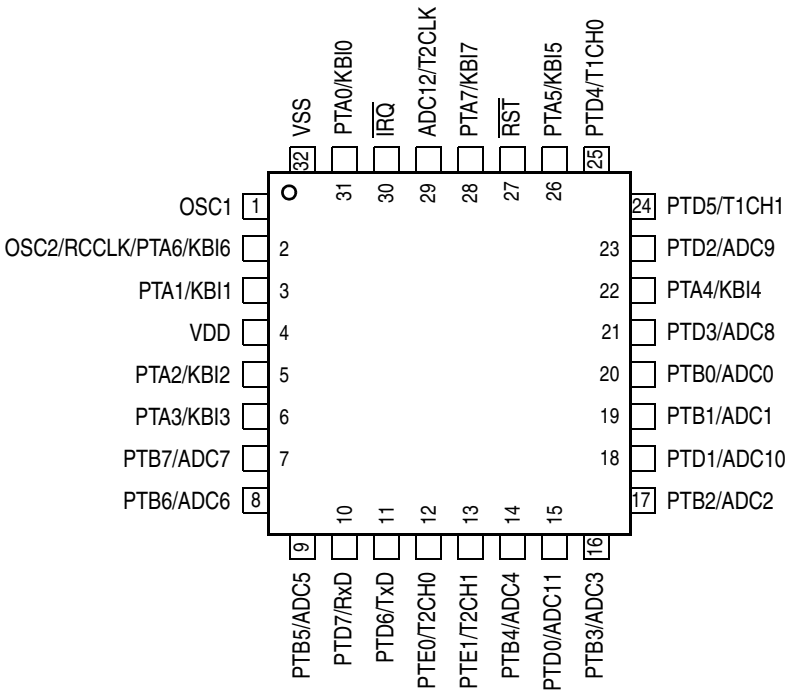


Figure 1-2. 32-Pin LQFP Pin Assignment

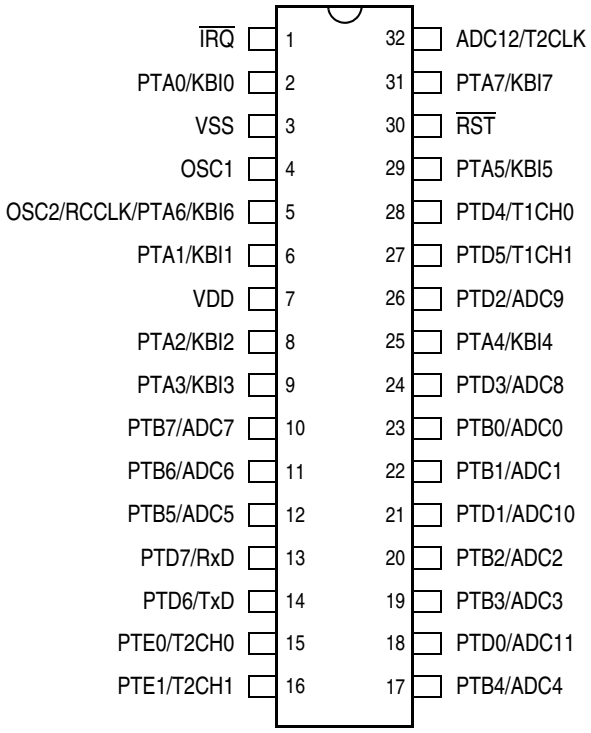


Figure 1-3. 32-Pin SDIP Pin Assignment

The resultant 16-bit address is used for specifying the start address of the FLASH memory for block protection. The FLASH is protected from this start address to the end of FLASH memory, at $FFFF. With this mechanism, the protect start address can be XX00, XX40, XX80, or XXC0 (at page boundaries — 64 bytes) within the FLASH memory.

Examples of protect start address:

BPR[7:0]	Start of Address of Protect Range (1)
$00–$70	The entire FLASH memory is protected.
$71 (**0111 0001**)	$DC40 (11**01 1100 01**00 0000)
$72 (**0111 0010**)	$DC80 (11**01 1100 10**00 0000)
$73 (**0111 0011**)	$DCC0 (11**01 1100 11**00 0000)
and so on...	
$FD (**1111 1101**)	$FF40 (11**11 1111 01**00 0000)
$FE (**1111 1110**)	$FF80 (11**11 1111 10**00 0000)
$FF	The entire FLASH memory is not protected.

1. The end address of the protected range is always $FFFF.

4.3 CPU Registers

Figure 4-1 shows the five CPU registers. CPU registers are not part of the memory map.

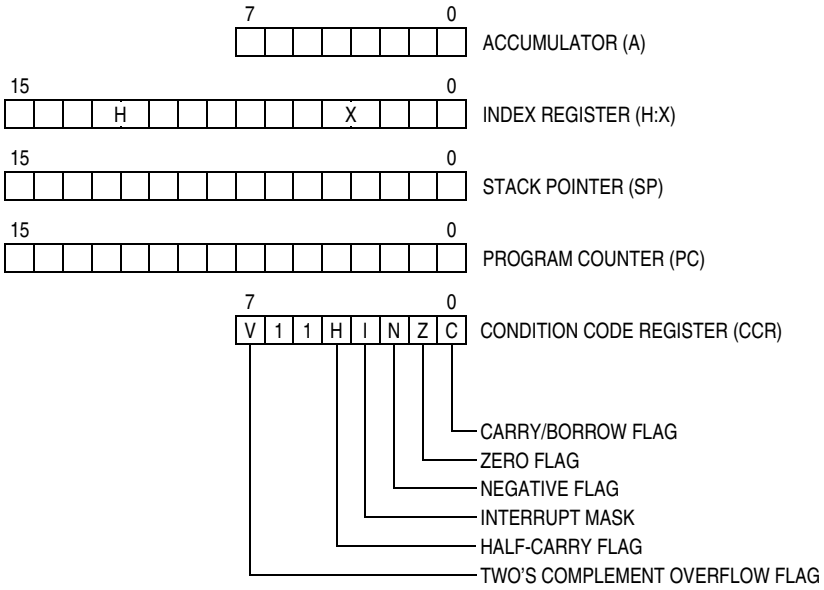


Figure 4-1. CPU Registers

4.3.1 Accumulator

The accumulator is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and the results of arithmetic/logic operations.

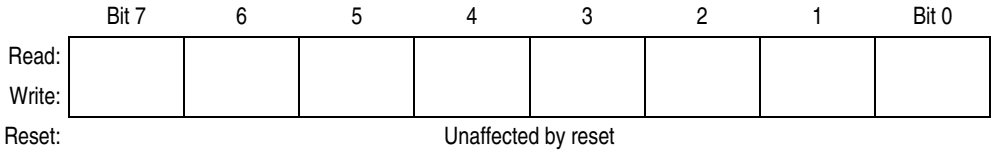


Figure 4-2. Accumulator (A)

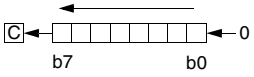
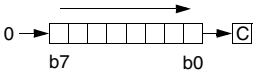
4.3.2 Index Register

The 16-bit index register allows indexed addressing of a 64-Kbyte memory space. H is the upper byte of the index register, and X is the lower byte. H:X is the concatenated 16-bit index register.

In the indexed addressing modes, the CPU uses the contents of the index register to determine the conditional address of the operand.

The index register can serve also as a temporary data storage location.

Table 4-1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X LDX *opr*,SP LDX *opr*,SP	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X LSL *opr*,SP	Logical Shift Left (Same as ASL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr*,X+ MOV #*opr,opr* MOV X+,*opr*	Move	$(M)_{Destination} \leftarrow (M)_{Source}$ H:X ← (H:X) + 1 (IX+D, DIX+)	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X NEG *opr*,SP	Negate (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap A	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		3
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X ORA *opr*,SP ORA *opr*,SP	Inclusive OR A and M	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); SP ← (SP) – 1	–	–	–	–	–	–	INH	87		2
PSHH	Push H onto Stack	Push (H); SP ← (SP) – 1	–	–	–	–	–	–	INH	8B		2
PSHX	Push X onto Stack	Push (X); SP ← (SP) – 1	–	–	–	–	–	–	INH	89		2
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2

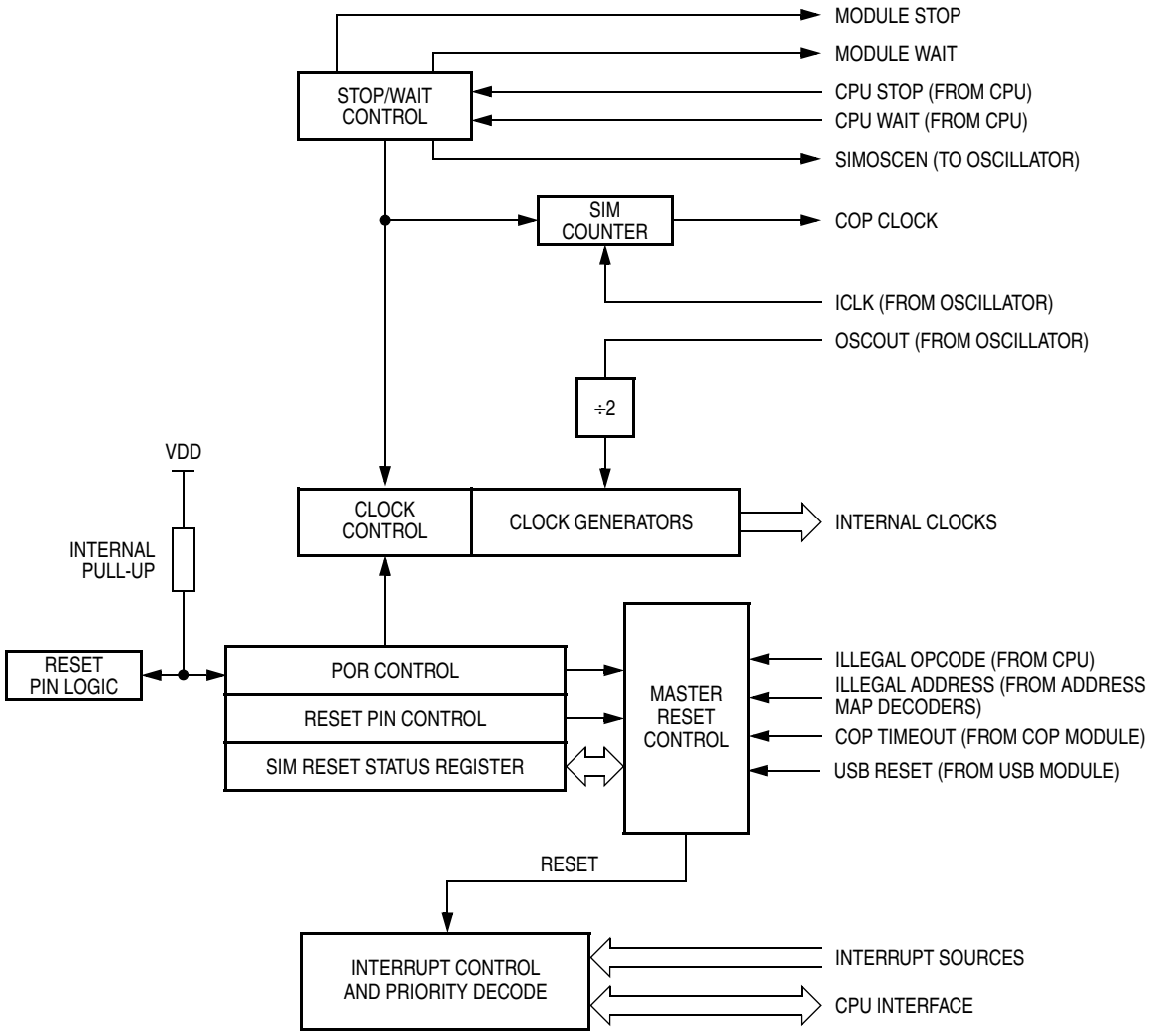


Figure 5-1. SIM Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	Break Status Register (BSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							NOTE	
		Reset:	0	0	0	0	0	0	0	0
	Note: Writing a logic 0 clears SBSW.									
$FE01	Reset Status Register (RSR)	Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
$FE02	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$FE03	Break Flag Control Register (BFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							

Figure 5-2. SIM I/O Register Summary

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 5-9 shows interrupt entry timing.
Figure 5-10 shows interrupt recovery timing.

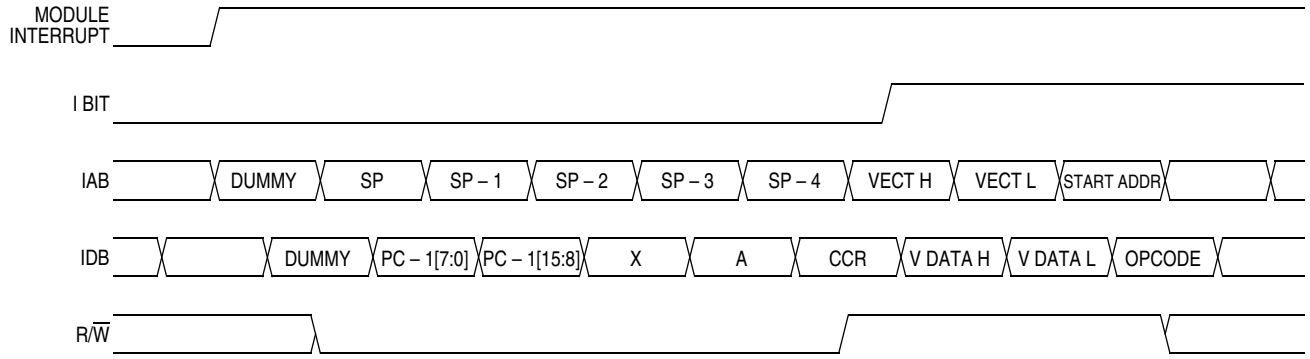


Figure 5-9. Interrupt Entry

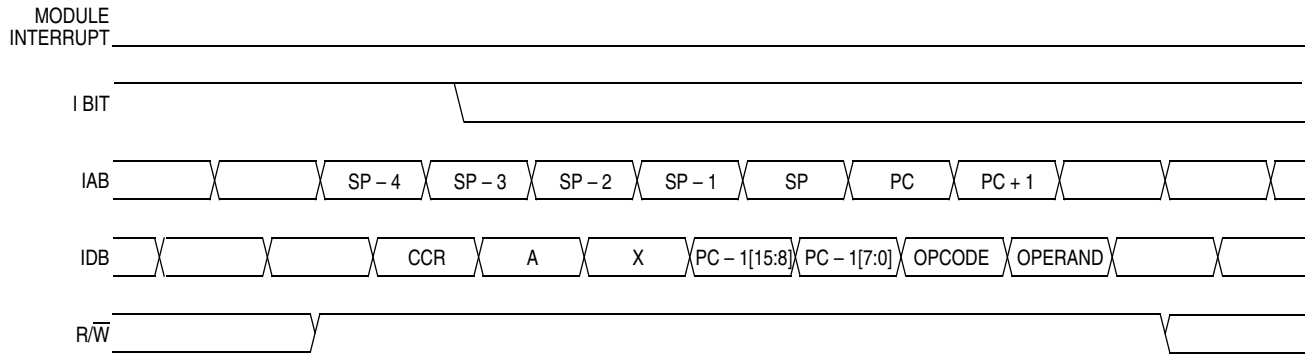


Figure 5-10. Interrupt Recovery

5.5.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register), and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 5-11 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

Chapter 6
Oscillator (OSC)

6.1 Introduction

The oscillator module provides the reference clocks for the MCU system and bus. Two oscillators are running on the device:

Selectable oscillator — for bus clock

- Crystal oscillator (XTAL) — built-in oscillator that requires an external crystal or ceramic-resonator. This option also allows an external clock that can be driven directly into OSC1.
- RC oscillator (RC) — built-in oscillator that requires an external resistor-capacitor connection only.

The selected oscillator is used to drive the bus clock, the SIM, and other modules on the MCU. The oscillator type is selected by programming a bit FLASH memory. The RC and crystal oscillator cannot run concurrently; one is disabled while the other is selected; because the RC and XTAL circuits share the same OSC1 pin.

Non-selectable oscillator — for COP

- Internal oscillator — built-in RC oscillator that requires no external components.

This internal oscillator is used to drive the computer operating properly (COP) module and the SIM. The internal oscillator runs continuously after a POR or reset, and is always available.

6.2 Oscillator Selection

The oscillator type is selected by programming a bit in a FLASH memory location; the mask option register (MOR), at $FFD0.
(See 3.5 Mask Option Register (MOR).)

NOTE

On the ROM device, the oscillator is selected by a ROM-mask layer at factory.

Address: $FFD0

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	OSCSEL	R	R	R	R	R	R	R
Erased:	1	1	1	1	1	1	1	1
Reset:	Unaffected by reset							

Non-volatile FLASH register; write by programming.

R = Reserved

Figure 6-1. Mask Option Register (MOR)

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, and should not be set to less than 4 (i.e. minimum bus speed is 1MHz).
- **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 128. Routines EE_WRITE and EE_READ are restricted to manipulate a data array between 2 to 15 bytes. Whereas 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, EE_WRITE. For the read routines: LDRNGE, MON_LDRNGE, and EE_READ, 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	$FC06
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 128 bytes (max. DATASIZE is 128).

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 32 bytes of data starting at FLASH location $EF00, 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.

7.5.4 MON_PRGRNGE

In monitor mode, MON_PRGRNGE is used to program a range of FLASH locations with data loaded into the data array.

Table 7-14. MON_PRGRNGE Routine

Routine Name	MON_PRGRNGE
Routine Description	Program a range of locations, in monitor mode
Calling Address	$FC28
Stack Used	17 bytes
Data Block Format	Bus speed Data size Starting address (high byte) Starting address (low byte) Data 1 : Data N

The MON_PRGRNGE routine is designed to be used in monitor mode. It performs the same function as the PRGRNGE routine (see 7.5.1 PRGRNGE), except that MON_PRGRNGE returns to the main program via an SWI instruction. After a MON_PRGRNGE call, the SWI instruction will return the control back to the monitor code.

7.5.5 MON_ERARNGE

In monitor mode, ERARNGE is used to erase a range of locations in FLASH.

Table 7-15. MON_ERARNGE Routine

Routine Name	MON_ERARNGE
Routine Description	Erase a page or the entire array, in monitor mode
Calling Address	$FF2C
Stack Used	11 bytes
Data Block Format	Bus speed Data size Starting address (high byte) Starting address (low byte)

The MON_ERARNGE routine is designed to be used in monitor mode. It performs the same function as the ERARNGE routine (see 7.5.2 ERARNGE), except that MON_ERARNGE returns to the main program via an SWI instruction. After a MON_ERARNGE call, the SWI instruction will return the control back to the monitor code.

8.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 8.4.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIM overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIM may pass the new value before it is written.

Use the following methods to synchronize unbuffered changes in the output compare value on channel x:

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

8.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the TIM channel 0 registers initially controls the output on the TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, TCH1, is available as a general-purpose I/O pin.

NOTE

In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.

8.4.4 Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIM can generate a PWM signal. The value in the TIM counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIM counter modulo registers. The time between overflows is the period of the PWM signal.

As Figure 8-3 shows, the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM

NOTE

In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.

8.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the TCH0 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, TCH1, is available as a general-purpose I/O pin.

NOTE

In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.

8.4.4.3 PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use the following initialization procedure:

1. In the TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter and prescaler by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See Table 8-3.)
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See Table 8-3.)

NOTE

In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty

8.9 I/O Registers

NOTE

References to either timer 1 or timer 2 may be made in the following text by omitting the timer number. For example, TSC may generically refer to both T1SC AND T2SC.

These I/O registers control and monitor operation of the TIM:

- TIM status and control register (TSC)
- TIM counter registers (TCNTH:TCNTL)
- TIM counter modulo registers (TMODH:TMODL)
- TIM channel status and control registers (TSC0, TSC1)
- TIM channel registers (TCH0H:TCH0L, TCH1H:TCH1L)

8.9.1 TIM Status and Control Register

The TIM status and control register (TSC):

- Enables TIM overflow interrupts
- Flags TIM overflows
- Stops the TIM counter
- Resets the TIM counter
- Prescales the TIM counter clock

Address: T1SC, $0020 and T2SC, $0030

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				
Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 8-4. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

This read/write flag is set when the TIM counter reaches the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TIM status and control register when TOF is set and then writing a logic 0 to TOF. If another TIM overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIM counter has reached modulo value
0 = TIM counter has not reached modulo value

TOIE — TIM Overflow Interrupt Enable Bit

This read/write bit enables TIM overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIM overflow interrupts enabled
0 = TIM overflow interrupts disabled

The generic pin names appear in the text of this section.

Table 9-1. Pin Name Conventions

Generic Pin Names:	RxD	TxD
Full Pin Names:	PTD7/RxD	PTD6/TxD

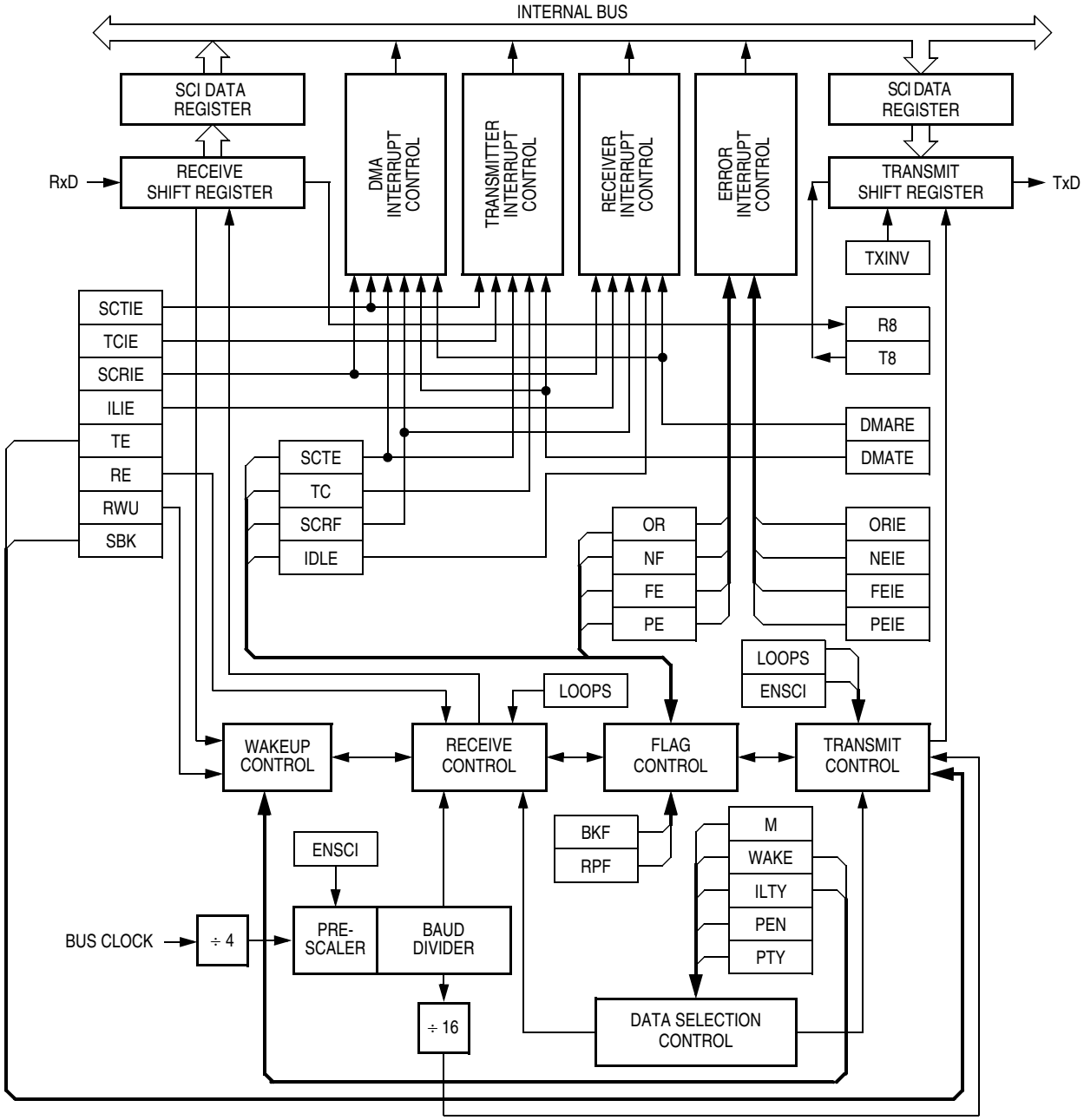


Figure 9-1. SCI Module Block Diagram

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 logic 1s from the transmit shift register to the TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the TxD returns to the idle condition (logic 1). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted. Reset clears the TE bit.

1 = Transmitter enabled
0 = Transmitter disabled

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 SCC1 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.

9.8.3 SCI Control Register 3

SCI control register 3:

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

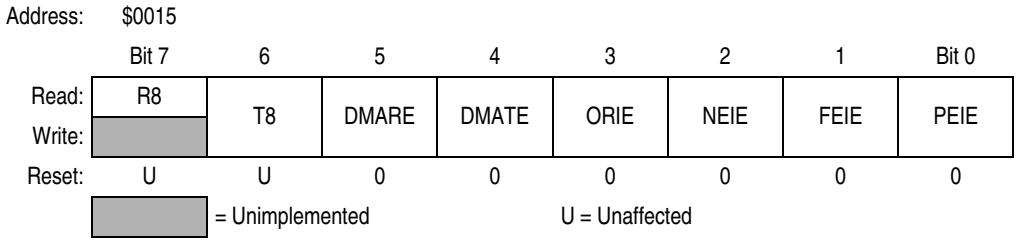


Figure 9-11. SCI Control Register 3 (SCC3)

R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.

When the SCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

T8 — Transmitted Bit 8

When the SCI is transmitting 9-bit characters, T8 is the read/write ninth bit (bit 8) of the transmitted character. T8 is loaded into the transmit shift register at the same time that the SCDR is loaded into the transmit shift register. Reset has no effect on the T8 bit.

DMARE — DMA Receive Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

1 = DMA not enabled to service SCI receiver DMA service requests generated by the SCRF bit (SCI receiver CPU interrupt requests enabled)

0 = DMA not enabled to service SCI receiver DMA service requests generated by the SCRF bit (SCI receiver CPU interrupt requests enabled)

DMATE — DMA Transfer Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

1 = SCTE DMA service requests enabled; SCTE CPU interrupt requests disabled

0 = SCTE DMA service requests disabled; SCTE CPU interrupt requests enabled

ORIE — Receiver Overrun Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the receiver overrun bit, OR.

1 = SCI error CPU interrupt requests from OR bit enabled

0 = SCI error CPU interrupt requests from OR bit disabled

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000D	Port A Input Pull-up Enable Register (PTAPUE)	Read: Write:	PTA6EN	PTAPUE6	PTAPUE5	PTAPUE4	PTAPUE3	PTAPUE2	PTAPUE1	PTAPUE0
		Reset:	0	0	0	0	0	0	0	0
$000E	PTA7 Input Pull-up Enable Register (PTA7PUE)	Read: Write:	PTAPUE7							
		Reset:	0	0	0	0	0	0	0	0

Figure 11-1. I/O Port Register Summary

Table 11-1. Port Control Register Bits Summary

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
A	0	DDRA0	KBI	KBIER ($001B)	KBIE0	PTA0/KBI0
	1	DDRA1			KBIE1	PTA1/KBI1
	2	DDRA2			KBIE2	PTA2/KBI2
	3	DDRA3			KBIE3	PTA3/KBI3
	4	DDRA4			KBIE4	PTA4/KBI4
	5	DDRA5			KBIE5	PTA5/KBI5
	6	DDRA6	OSC KBI	PTAPUE ($000D) KBIER ($001B)	PTA6EN KBIE6	RCCLK/PTA6/KBI6(1)
	7	DDRA7	KBI	KBIER ($001B)	KBIE7	PTA7/KBI7
B	0	DDRB0	ADC	ADSCR ($003C)	ADCH[4:0]	PTB0/ADC0
	1	DDRB1				PTB1/ADC1
	2	DDRB2				PTB2/ADC2
	3	DDRB3				PTB3/ADC3
	4	DDRB4				PTB4/ADC4
	5	DDRB5				PTB5/ADC5
	6	DDRB6				PTB6/ADC6
	7	DDRB7				PTB7/ADC7
D	0	DDRD0	ADC	ADSCR ($003C)	ADCH[4:0]	PTD0/ADC11
	1	DDRD1				PTD1/ADC10
	2	DDRD2				PTD2/ADC9
	3	DDRD3				PTD3/ADC8
	4	DDRD4	TIM1	T1SC0 ($0025)	ELS0B:ELS0A	PTD4/T1CH0
	5	DDRD5		T1SC1 ($0028)	ELS1B:ELS1A	PTD5/T1CH1
	6	DDRD6	SCI	SCC1 ($0013)	ENSCI	PTD6/TxD
	7	DDRD7				PTD7/RxD
E	0	DDRE0	TIM2	T2SC0 ($0035)	ELS0B:ELS0A	PTE0/T2CH0
	1	DDRE1		T2SC1 ($0038)	ELS1B:ELS1A	PTE1/T2CH1

1. RCCLK/PTA6/KBI6 pin is only available when OSCSEL=0 (RC option); PTAPUE register has priority control over the port pin. RCCLK/PTA6/KBI6 is the OSC2 pin when OSCSEL=1 (XTAL option).

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	Break Status Register (BSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							See note	
		Reset:							0	
$FE03	Break Flag Control Register (BFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE0C	Break Address High Register (BRKH)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address low Register (BRKL)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Status and Control Register (BRKSCR)	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Note: Writing a logic 0 clears SBSW.

= Unimplemented R = Reserved

Figure 16-2. Break I/O Register Summary

16.3.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See 5.7.3 Break Flag Control Register (BFCR) and see the Break Interrupts subsection for each module.)

16.3.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with $FFFC:$FFFD ($FEFC:$FEFD in monitor mode)

The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

16.3.3 TIM During Break Interrupts

A break interrupt stops the timer counter.

16.3.4 COP During Break Interrupts

The COP is disabled during a break interrupt when V_{TST} is present on the $\overline{RST}$ pin.

Address:	$FE0D							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 16-5. Break Address Register Low (BRKL)

16.4.3 Break Status Register

The break status register contains a flag to indicate that a break caused an exit from stop or wait mode.

Address:	$FE00							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	R	R	R	R	R	SBSW	R
Write:							Note(1)	
Reset:							0	

R = Reserved

1. Writing a logic zero clears SBSW.

Figure 16-6. Break Status Register (BSR)

SBSW — SIM Break Stop/Wait

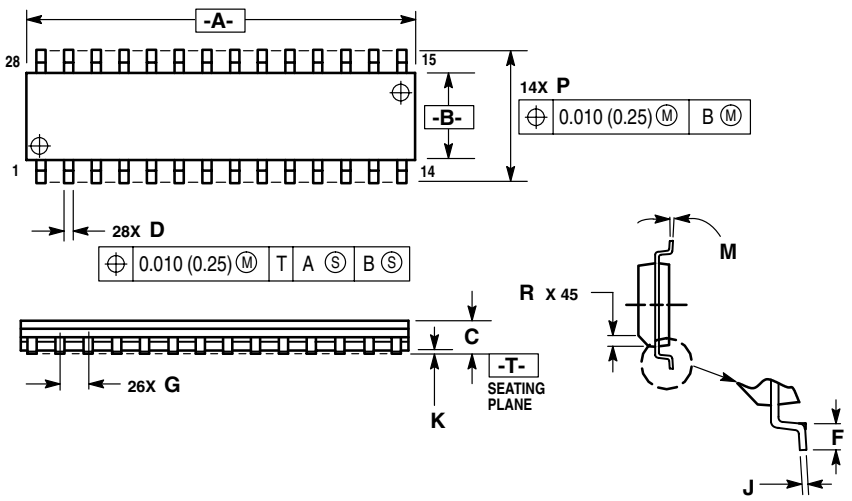
This status bit is useful in applications requiring a return to wait or stop mode after exiting from a break interrupt. Clear SBSW by writing a logic zero to it. Reset clears SBSW.

1 = Stop mode or wait mode was exited by break interrupt
0 = Stop mode or wait mode was not exited by break interrupt

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it. The following code is an example of this.

```
; This code works if the H register has been pushed onto the stack in the break
; service routine software. This code should be executed at the end of the
; break service routine software.

  HIBYTE   EQU    5

  LOBYTE   EQU    6

;          If not SBSW, do RTI

           BRCLR  SBSW,BSR, RETURN   ; See if wait mode or stop mode was exited
                                     ; by break.

           TST    LOBYTE,SP          ; If RETURNLO is not zero,

           BNE    DOLO               ; then just decrement low byte.

           DEC    HIBYTE,SP          ; Else deal with high byte, too.

  DOLO     DEC    LOBYTE,SP          ; Point to WAIT/STOP opcode.

  RETURN   PULH                      ; Restore H register.
           RTI
```

18.5 28-Pin Small Outline Integrated Circuit Package (SOIC)

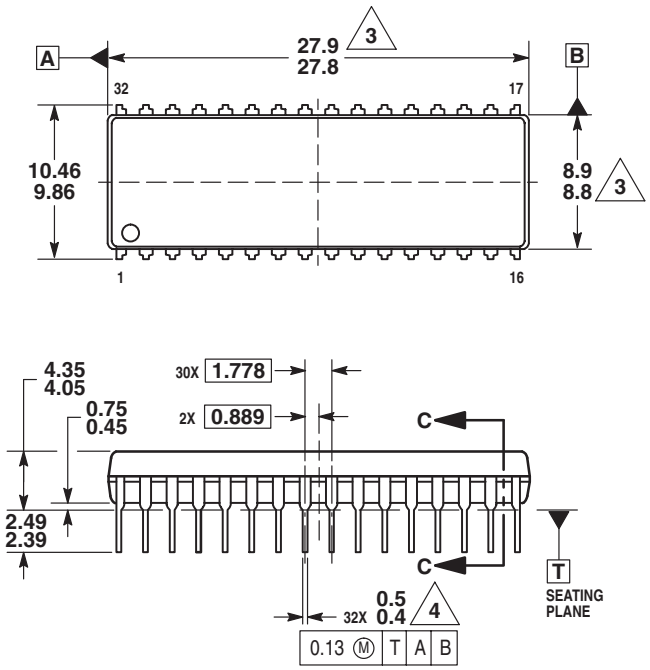


NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006) PER SIDE.
5. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.13 (0.005) TOTAL IN EXCESS OF D DIMENSION AT MAXIMUM MATERIAL CONDITION.

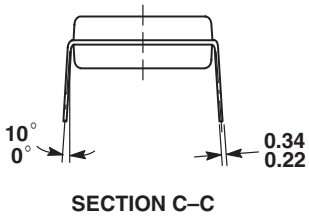
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	17.80	18.05	0.701	0.711
B	7.40	7.60	0.292	0.299
C	2.35	2.65	0.093	0.104
D	0.35	0.49	0.014	0.019
F	0.41	0.90	0.016	0.035
G	1.27 BSC		0.050 BSC	
J	0.23	0.32	0.009	0.013
K	0.13	0.29	0.005	0.011
M	0°	8°	0°	8°
P	10.01	10.55	0.395	0.415
R	0.25	0.75	0.010	0.029

Figure 18-4. 28-Pin SOIC (Case #751F)

18.6 32-Pin Shrink Dual In-Line Package (SDIP)



NOTES:
1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5, 1994.
3. DIMENSIONS DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS.
4. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION.



SECTION C–C

Figure 18-5. 32-Pin SDIP (Case #1376)