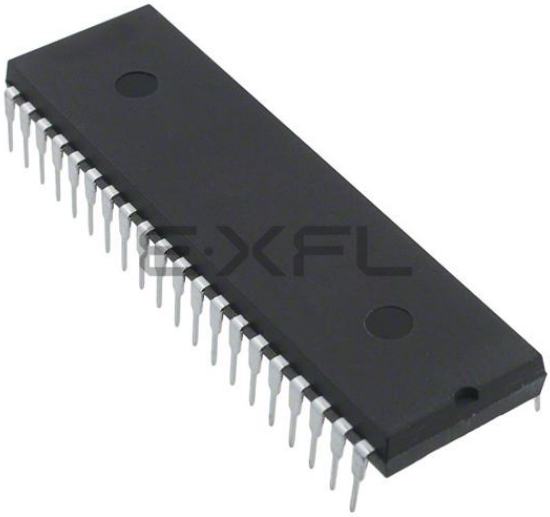


Welcome to E-XFL.COM

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	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	36
Program Memory Size	14KB (8K x 14)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 28x10b; D/A 4x5b, 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1777-e-p

TABLE 3-18: SPECIAL FUNCTION REGISTER SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 0											
00Ch	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx xxxx	uuuu uuuu
00Dh	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
00Eh	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
00Fh	PORTD(3)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
010h	PORTE	—	—	—	—	RE3	RE2(3)	RE1(3)	RE0(3)	---- xxxx	---- uuuu
011h	PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
012h	PIR2	OSFIF	C2IF	C1IF	COG1IF	BCL1IF	C4IF	C3IF	CCP2IF	0000 0000	0000 0000
013h	PIR3	—	—	COG2IF	ZCDIF	CLC4IF	CLC3IF	CLC2IF	CLC1IF	--00 0000	--00 0000
014h	PIR4	—	TMR8IF	TMR5GIF	TMR5IF	TMR3GIF	TMR3IF	TMR6IF	TRM4IF	-000 0000	-000 0000
015h	PIR5	CCP8IF(3)	CCP7IF	COG4IF(3)	COG3IF	C8IF(3)	C7IF(3)	C6IF	C5IF	0000 --00	0000 --00
016h	PIR6	—	—	—	—	PWM12IF(3)	PWM11IF	PWM6IF	PWM5IF	---- 0000	---- 0000
017h	TMR0	Timer0 Module Register								0000 0000	0000 0000
018h	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
019h	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
01Ah	T1CON	CS<1:0>		CKPS<1:0>		OSCEN	$\overline{\text{SYNC}}$	—	ON	0000 00-0	uuuu uu-u
01Bh	T1GCON	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL	GSS<1:0>		0000 0x00	uuuu uxuu
01Ch	TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
01Dh	TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
01Eh	T3CON	CS<1:0>		CKPS<1:0>		OSCEN	$\overline{\text{SYNC}}$	—	ON	0000 00-0	uuuu uu-u
01Fh	T3GCON	GE	GPOL	GTM	GSPM	GGO/$\overline{\text{DONE}}$	GVAL	GSS<1:0>		0000 0x00	uuuu uxuu

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: Unimplemented, read as '1'.
2: Unimplemented on PIC16LF1777/8/9.
3: Unimplemented on PIC16(L)F1778.

TABLE 3-18: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 1											
08Ch	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111
08Dh	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
08Eh	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	1111 1111
08Fh	TRISD(3)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111	1111 1111
090h	TRISE	—	—	—	—	—(1)	TRISE2(3)	TRISE1(3)	TRISE0(3)	---- 1111	---- 1111
091h	PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
092h	PIE2	OSFIE	C2IE	C1IE	COG1IE	BCL1IE	C4IE	C3IE	CCP2IE	0000 0000	0000 0000
093h	PIE3	—	—	COG2IE	ZCDIE	CLC4IE	CLC3IE	CLC2IE	CLC1IE	--00 0000	--00 0000
094h	PIE4	—	TMR8IE	TMR5GIE	TMR5IE	TMR3GIE	TMR3IE	TMR6IE	TRM4IE	-000 0000	-000 0000
095h	PIE5	CCP8IE(3)	CCP7IE	COG4IE(3)	COG3IE	C8IE(3)	C7IE(3)	C6IE	C5IE	0000 0000	0000 0000
096h	PIE6	—	—	—	—	PWM12IE(3)	PWM11IE	PWM6IE	PWM5IE	---- 0000	---- 0000
097h	OPTION_REG	$\overline{\text{WPUEN}}$	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>			1111 1111	1111 1111
098h	PCON	STKOVF	STKUNF	—	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	00-1 11qq	qq-q qquu
099h	WDTCON	—	—	WDTPS<4:0>					SWDTEN	--01 0110	--01 0110
09Ah	OSCTUNE	—	—	TUN<5:0>						--00 0000	--00 0000
09Bh	OSCCON	SPLLEN	IRCF<3:0>				—	SCS<1:0>		0011 1-00	0011 1-00
09Ch	OSCSTAT	SOSCR	PLLR	OSTS	HFIOFR	HFIOFL	MFIOFR	LFIOFR	HFIOFS	0qq0 0q0q	qqqq qq0q
09Dh	BORCON	SBOREN	BORFS	—	—	—	—	—	BORRDY	10-- ---q	uu-- ---u
09Eh	FVRCON	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR<1:0>		ADFVR<1:0>		0q00 0000	0q00 0000
09Fh	ZCD1CON	EN	—	OUT	POL	—	—	INTP	INTN	0-x0 --00	0-x0 --00

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: Unimplemented, read as '1'.
2: Unimplemented on PIC16LF1777/8/9.
3: Unimplemented on PIC16(L)F1778.

TABLE 3-18: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 29											
E8Ch — E8Fh	—	Unimplemented								—	—
E90h	RA0PPS	—	—	RA0PPS<4:0>						--00 0000	--uu uuuu
E91h	RA1PPS	—	—	RA1PPS<4:0>						--00 0000	--uu uuuu
E92h	RA2PPS	—	—	RA2PPS<4:0>						--00 0000	--uu uuuu
E93h	RA3PPS	—	—	RA3PPS<4:0>						--00 0000	--uu uuuu
E94h	RA4PPS	—	—	RA4PPS<4:0>						--00 0000	--uu uuuu
E95h	RA5PPS	—	—	RA5PPS<4:0>						--00 0000	--uu uuuu
E96h	RA6PPS	—	—	RA6PPS<4:0>						--00 0000	--uu uuuu
E97h	RA7PPS	—	—	RA7PPS<4:0>						--00 0000	--uu uuuu
E98h	RB0PPS	—	—	RB0PPS<5:0>						--00 0000	--uu uuuu
E99h	RB1PPS	—	—	RB1PPS<5:0>						--00 0000	--uu uuuu
E9Ah	RB2PPS	—	—	RB2PPS<5:0>						--00 0000	--uu uuuu
E9Bh	RB3PPS	—	—	RB3PPS<5:0>						--00 0000	--uu uuuu
E9Ch	RB4PPS	—	—	RB4PPS<5:0>						--00 0000	--uu uuuu
E9Dh	RB5PPS	—	—	RB5PPS<5:0>						--00 0000	--uu uuuu
E9Eh	RB6PPS	—	—	RB6PPS<5:0>						--00 0000	--uu uuuu
E9Fh	RB7PPS	—	—	RB7PPS<5:0>						--00 0000	--uu uuuu
EA0h	RC0PPS	—	—	RC0PPS<5:0>						--00 0000	--uu uuuu
EA1h	RC1PPS	—	—	RC1PPS<5:0>						--00 0000	--uu uuuu
EA2h	RC2PPS	—	—	RC2PPS<5:0>						--00 0000	--uu uuuu
EA3h	RC3PPS	—	—	RC3PPS<5:0>						--00 0000	--uu uuuu
EA4h	RC4PPS	—	—	RC4PPS<5:0>						--00 0000	--uu uuuu
EA5h	RC5PPS	—	—	RC5PPS<5:0>						--00 0000	--uu uuuu
EA6h	RC6PPS	—	—	RC6PPS<5:0>						--00 0000	--uu uuuu
EA7h	RC7PPS	—	—	RC7PPS<5:0>						--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: Unimplemented, read as '1'.
2: Unimplemented on PIC16LF1777/8/9.
3: Unimplemented on PIC16(L)F1778.

TABLE 3-18: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 29 (Cont.)											
EA8h	RD0PPS(3)	—	—	RD0PPS<5:0>						--00 0000	--uu uuuu
EA9h	RD1PPS(3)	—	—	RD1PPS<5:0>						--00 0000	--uu uuuu
EAAh	RD2PPS(3)	—	—	RD2PPS<5:0>						--00 0000	--uu uuuu
EABh	RD3PPS(3)	—	—	RD3PPS<5:0>						--00 0000	--uu uuuu
EACh	RD4PPS(3)	—	—	RD4PPS<5:0>						--00 0000	--uu uuuu
EADh	RD5PPS(3)	—	—	RD5PPS<5:0>						--00 0000	--uu uuuu
EAEh	RD6PPS(3)	—	—	RD6PPS<5:0>						--00 0000	--uu uuuu
EAFh	RD7PPS(3)	—	—	RD7PPS<5:0>						--00 0000	--uu uuuu
EB0h	RE0PPS(3)	—	—	RE0PPS<5:0>						--00 0000	--uu uuuu
EB1h	RE1PPS(3)	—	—	RE1PPS<5:0>						--00 0000	--uu uuuu
EB2h	RE2PPS(3)	—	—	RE2PPS<5:0>						--00 0000	--uu uuuu
EB3h to EEFh	—	Unimplemented								—	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: Unimplemented, read as '1'.
2: Unimplemented on PIC16LF1777/8/9.
3: Unimplemented on PIC16(L)F1778.

3.6 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figures 3-1 and 3-2). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when CALL or CALLW instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer if the STVREN bit is programmed to '0' (Configuration Words). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

Note:	There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, CALLW, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

3.6.1 ACCESSING THE STACK

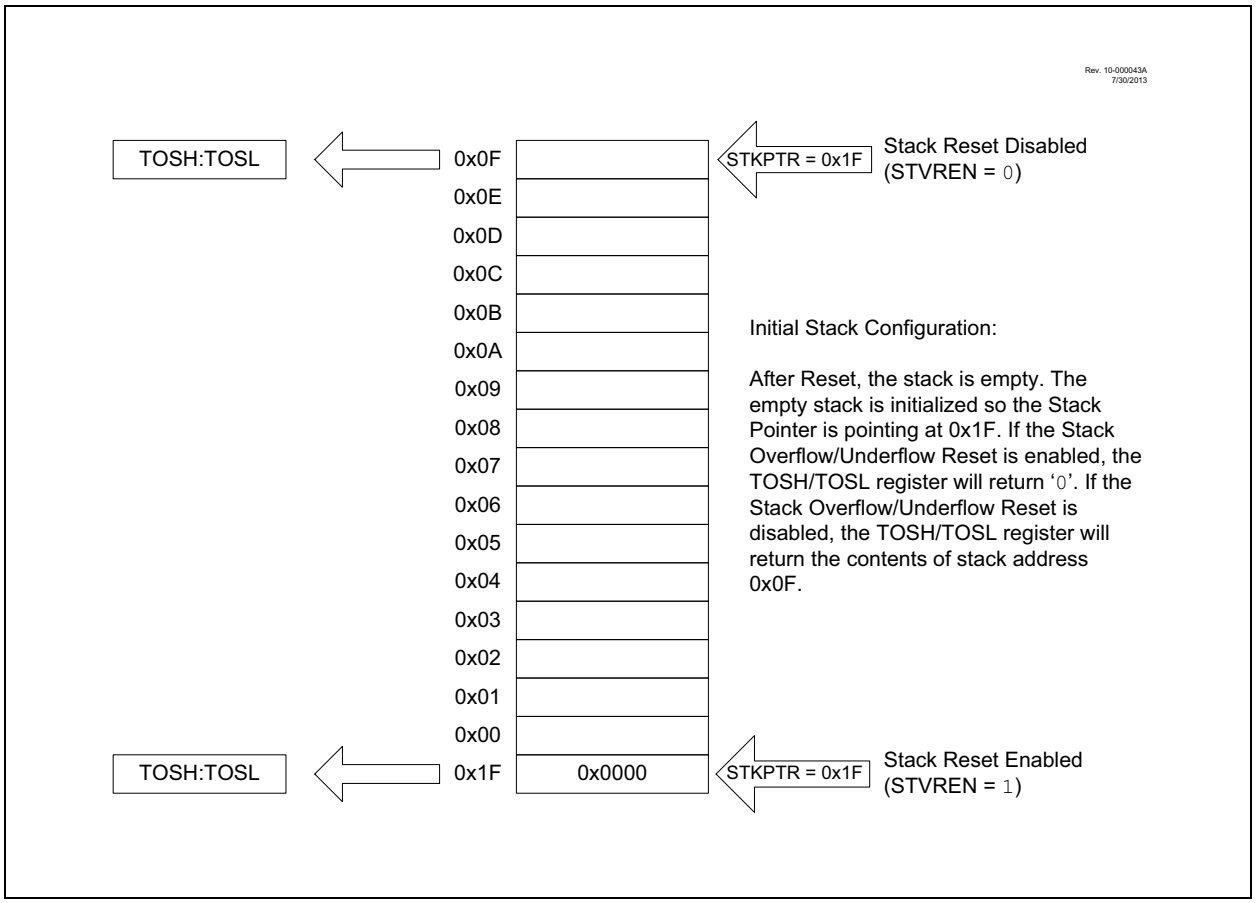
The stack is available through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. The TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is five bits to allow detection of overflow and underflow.

Note:	Care should be taken when modifying the STKPTR while interrupts are enabled.

During normal program operation, CALL, CALLW and Interrupts will increment STKPTR while RETLW, RETURN, and RETFIE will decrement STKPTR. At any time, STKPTR can be inspected to see how much stack is left. The STKPTR always points at the currently used place on the stack. Therefore, a CALL or CALLW will increment the STKPTR and then write the PC, and a return will unload the PC and then decrement the STKPTR.

Reference Figure 3-5 through Figure 3-8 for examples of accessing the stack.

FIGURE 3-5: ACCESSING THE STACK EXAMPLE 1



REGISTER 11-4: ANSELA: PORTA ANALOG SELECT REGISTER

U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	ANSA5	ANSA4	ANSA3	ANSA2	ANSA1	ANSA0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **ANSA<5:0>**: Analog Select between Analog or Digital Function on pins RA<5:0>
1 = Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

Note 1: When setting a pin to an analog input, the corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.

REGISTER 11-5: WPUA: WEAK PULL-UP PORTA REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 **WPUA<7:0>**: Weak Pull-up Register bits(1),(2)
1 = Pull-up enabled
0 = Pull-up disabled

Note 1: Global $\overline{\text{WPUEN}}$ bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.
2: The weak pull-up device is automatically disabled if the pin is configured as an output.

16.2 ADC Operation

16.2.1 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. Setting the GO/$\overline{\text{DONE}}$ bit of the ADCON0 register to a '1' will start the Analog-to-Digital conversion.

Note:	The GO/$\overline{\text{DONE}}$ bit should not be set in the same instruction that turns on the ADC. Refer to **Section 16.2.6 "ADC Conversion Procedure"**.

16.2.2 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/$\overline{\text{DONE}}$ bit
- Set the ADIF Interrupt Flag bit
- Update the ADRESH and ADRESL registers with new conversion result

16.2.3 TERMINATING A CONVERSION

If a conversion must be terminated before completion, the GO/$\overline{\text{DONE}}$ bit can be cleared in software. The ADRESH and ADRESL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.

Note:	A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

16.2.4 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC oscillator source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the ADC module is turned off after the conversion completes, although the ADON bit remains set.

When the ADC clock source is something other than FRC, a SLEEP instruction causes the present conversion to be aborted and the ADC module is turned off, although the ADON bit remains set.

16.2.5 AUTO-CONVERSION TRIGGER

The Auto-conversion Trigger allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the GO/$\overline{\text{DONE}}$ bit is set by hardware.

The Auto-conversion Trigger source is selected with the TRIGSEL<5:0> bits of the ADCON2 register.

Using the Auto-conversion Trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met.

See Table 16-2 for auto-conversion sources.

TABLE 16-2: AUTO-CONVERSION SOURCES

Source Peripheral	Signal Name
CCP1	CCP1_trigger
CCP2	CCP2_trigger
CCP7	CCP7_trigger
CCP8(1)	CCP8_trigger
Timer0	T0_overflow
Timer1	T1_overflow
Timer3	T3_overflow
Timer5	T5_overflow
Timer2	T2_postscaled
Timer4	T4_postscaled
Timer6	T6_postscaled
Timer8	T8_postscaled
Comparator C1	sync_C1OUT
Comparator C2	sync_C2OUT
Comparator C3	sync_C3OUT
Comparator C4	sync_C4OUT
Comparator C5	sync_C5OUT
Comparator C6	sync_C6OUT
Comparator C7(1)	sync_C7OUT
Comparator C8(1)	sync_C8OUT
CLC1	LC1_out
CLC2	LC2_out
CLC3	LC3_out
CLC4	LC4_out
PWM3	PWM3OUT
PWM4	PWM4OUT
PWM9	PWM9OUT
PWM9	PR/PH/OF/DC9_match
PWM5	PR/PH/OF/DC5_match
PWM6	PR/PH/OF/DC6_match
PWM10(1)	PR/PH/OF/DC10_match
PWM11	PR/PH/OF/DC11_match
PWM12(1)	PR/PH/OF/DC12_match
ADCACT	ADCACTPPS Pin

Note 1: PIC16(L)F1777/9 only.

21.1.3 SOFTWARE PROGRAMMABLE PRESCALER

A software programmable prescaler is available for exclusive use with Timer0. The prescaler is enabled by clearing the PSA bit of the OPTION_REG register.

Note:	The Watchdog Timer (WDT) uses its own independent prescaler.

There are eight prescaler options for the Timer0 module ranging from 1:2 to 1:256. The prescale values are selectable via the PS<2:0> bits of the OPTION_REG register. In order to have a 1:1 prescaler value for the Timer0 module, the prescaler must be disabled by setting the PSA bit of the OPTION_REG register.

The prescaler is not readable or writable. All instructions writing to the TMR0 register will clear the prescaler.

21.1.4 TIMER0 INTERRUPT

Timer0 will generate an interrupt when the TMR0 register overflows from FFh to 00h. The TMR0IF interrupt flag bit of the INTCON register is set every time the TMR0 register overflows, regardless of whether or not the Timer0 interrupt is enabled. The TMR0IF bit can only be cleared in software. The Timer0 interrupt enable is the TMR0IE bit of the INTCON register.

Note:	The Timer0 interrupt cannot wake the processor from Sleep since the timer is frozen during Sleep.

21.1.5 8-BIT COUNTER MODE SYNCHRONIZATION

When in 8-Bit Counter mode, the incrementing edge on the T0CKI pin must be synchronized to the instruction clock. Synchronization can be accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the instruction clock. The high and low periods of the external clocking source must meet the timing requirements as shown in Table 36-12: Timer0 and Timer1 External Clock Requirements.

21.1.6 OPERATION DURING SLEEP

Timer0 cannot operate while the processor is in Sleep mode. The contents of the TMR0 register will remain unchanged while the processor is in Sleep mode.

22.7 Timer1 Interrupt

The Timer1 register pair (TMR1H:TMR1L) increments to FFFFh and rolls over to 0000h. When Timer1 rolls over, the Timer1 interrupt flag bit of the PIR1 register is set. To enable the interrupt-on-rollover, you must set these bits:

- ON bit of the T1CON register
- TMR1IE bit of the PIE1 register
- PEIE bit of the INTCON register
- GIE bit of the INTCON register

The interrupt is cleared by clearing the TMR1IF bit in the Interrupt Service Routine.

Note:	The TMR1H:TMR1L register pair and the TMR1IF bit should be cleared before enabling interrupts.

22.8 Timer1 Operation During Sleep

Timer1 can only operate during Sleep when setup in Asynchronous Counter mode. In this mode, an external crystal or clock source can be used to increment the counter. To set up the timer to wake the device:

- ON bit of the T1CON register must be set
- TMR1IE bit of the PIE1 register must be set
- PEIE bit of the INTCON register must be set
- $\overline{\text{SYNC}}$ bit of the T1CON register must be set
- CS bits of the T1CON register must be configured
- OSCEN bit of the T1CON register must be configured

The device will wake-up on an overflow and execute the next instructions. If the GIE bit of the INTCON register is set, the device will call the Interrupt Service Routine.

Secondary oscillator will continue to operate in Sleep regardless of the $\overline{\text{SYNC}}$ bit setting.

22.9 CCP Capture/Compare Time Base

The CCP modules use the TMR1H:TMR1L register pair as the time base when operating in Capture or Compare mode.

In Capture mode, the value in the TMR1H:TMR1L register pair is copied into the CCPR1H:CCPR1L register pair on a configured event.

In Compare mode, an event is triggered when the value in the CCPR1H:CCPR1L register pair matches the value in the TMR1H:TMR1L register pair. This event can be an Auto-conversion Trigger.

For more information, see **Section 24.0 "Capture/Compare/PWM Modules"**.

22.10 CCP Auto-Conversion Trigger

When any of the CCP's are configured to trigger an auto-conversion, the trigger will clear the TMR1H:TMR1L register pair. This auto-conversion does not cause a Timer1 interrupt. The CCP module may still be configured to generate a CCP interrupt.

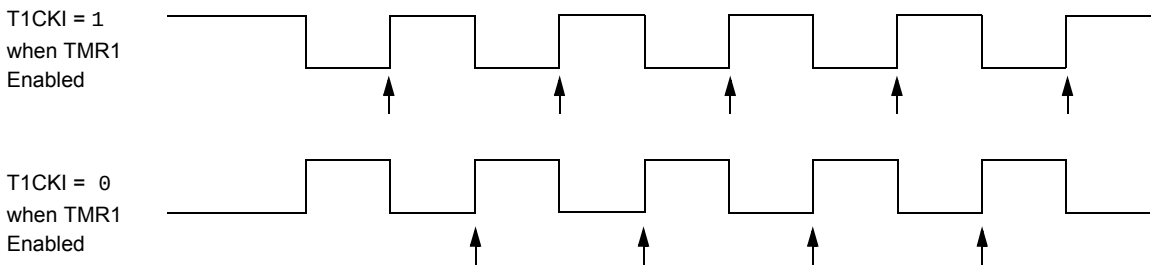
In this mode of operation, the CCPR1H:CCPR1L register pair becomes the period register for Timer1.

Timer1 should be synchronized and FOSC/4 should be selected as the clock source in order to utilize the Auto-conversion Trigger. Asynchronous operation of Timer1 can cause an Auto-conversion Trigger to be missed.

In the event that a write to TMR1H or TMR1L coincides with an Auto-conversion Trigger from the CCP, the write will take precedence.

For more information, see **Section 24.2.1 "Auto-Conversion Trigger"**.

FIGURE 22-2: TIMER1 INCREMENTING EDGE



Note 1: Arrows indicate counter increments.

2: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.

REGISTER 26-13: PWMxOFH: PWMx OFFSET COUNT HIGH REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
OF<15:8>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **OF<15:8>**: PWM Offset High bits
Upper eight bits of PWM offset count

REGISTER 26-14: PWMxOFL: PWMx OFFSET COUNT LOW REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
OF<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **OF<7:0>**: PWM Offset Low bits
Lower eight bits of PWM offset count

27.8.1 FALLING EVENT BLANKING OF RISING EVENT INPUTS

The falling event blanking counter inhibits rising event inputs from triggering a rising event. The falling event blanking time starts when the rising_event output drive goes false.

The falling event blanking time is set by the value contained in the COGxBLKF register (Register 27-17). Blanking times are calculated using the formula shown in Equation 27-1.

When the COGxBLKF value is zero, falling event blanking is disabled and the blanking counter output is true, thereby, allowing the event signal to pass straight through to the event trigger circuit.

27.8.2 RISING EVENT BLANKING OF FALLING EVENT INPUTS

The rising event blanking counter inhibits falling event inputs from triggering a falling event. The rising event blanking time starts when the falling_event output drive goes false.

The rising event blanking time is set by the value contained in the COGxBLKR register (Register 27-16).

When the COGxBLKR value is zero, rising event blanking is disabled and the blanking counter output is true, thereby, allowing the event signal to pass straight through to the event trigger circuit.

27.8.3 BLANKING TIME UNCERTAINTY

When the rising and falling sources that trigger the blanking counters are asynchronous to the COG_clock, it creates uncertainty in the blanking time. The maximum uncertainty is equal to one COG_clock period. Refer to Equation 27-1 and Example 27-1 for more detail.

27.9 Phase Delay

It is possible to delay the assertion of either or both the rising event and falling events. This is accomplished by placing a non-zero value in COGxPHR or COGxPHF phase-delay count registers, respectively (Register 27-18 and Register 27-19). Refer to Figure 27-10 for COG operation with CCP1 and phase delay. The delay from the input rising event signal switching to the actual assertion of the events is calculated the same as the dead-band and blanking delays. Refer to Equation 27-1.

When the phase-delay count value is zero, phase delay is disabled and the phase-delay counter output is true, thereby, allowing the event signal to pass straight through to the complementary output driver flop.

27.9.1 CUMULATIVE UNCERTAINTY

It is not possible to create more than one COG_clock of uncertainty by successive stages. Consider that the phase-delay stage comes after the blanking stage, the dead-band stage comes after either the blanking or phase-delay stages, and the blanking stage comes after the dead-band stage. When the preceding stage is enabled, the output of that stage is necessarily synchronous with the COG_clock, which removes any possibility of uncertainty in the succeeding stage.

EQUATION 27-1: PHASE, DEAD-BAND, AND BLANKING TIME CALCULATION

$$T_{\min} = \frac{\text{Count}}{F_{COG\_\text{clock}}}$$

$$T_{\max} = \frac{\text{Count} + 1}{F_{COG\_\text{clock}}}$$

$$T_{\text{uncertainty}} = T_{\max} - T_{\min}$$

Also:

$$T_{\text{uncertainty}} = \frac{1}{F_{COG\_\text{clock}}}$$

Where:

T	Count
Rising Phase Delay	COGxPHR
Falling Phase Delay	COGxPHF
Rising dead band	COGxDBR
Falling dead band	COGxDBF
Rising Event Blanking	COGxBLKR
Falling Event Blanking	COGxBLKF

REGISTER 27-14: COGxDBR: COG RISING EVENT DEAD-BAND COUNT REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
		DBR<5:0>					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **DBR<5:0>:** Rising Event Dead-Band Count Value bits

RDBS = 1:
= Number of delay chain element periods to delay primary output after rising event

RDBS = 0:
= Number of COGx clock periods to delay primary output after rising event

REGISTER 27-15: COGxDBF: COG FALLING EVENT DEAD-BAND COUNT REGISTER

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
		DBF<5:0>					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **DBF<5:0>:** Falling Event Dead-Band Count Value bits

FDBS = 1:
= Number of delay chain element periods to delay complementary output after falling event input

FDBS = 0:
= Number of COGx clock periods to delay complementary output after falling event input

28.1 CLCx Setup

Programming the CLCx module is performed by configuring the four stages in the logic signal flow. The four stages are:

- Data selection
- Data gating
- Logic function selection
- Output polarity

Each stage is setup at run time by writing to the corresponding CLCx Special Function Registers. This has the added advantage of permitting logic reconfiguration on-the-fly during program execution.

28.1.1 DATA SELECTION

There are 32 signals available as inputs to the configurable logic. Four 32-input multiplexers are used to select the inputs to pass on to the next stage.

Data selection is through four multiplexers as indicated on the left side of Figure 28-2. Data inputs in the figure are identified by a generic numbered input name.

Table 28-1 correlates the generic input name to the actual signal for each CLC module. The column labeled dy indicates the MUX selection code for the selected data input. DxS is an abbreviation for the MUX select input codes: D1S<4:0> through D4S<4:0>.

Data inputs are selected with the CLCxSEL0 through CLCxSEL3 registers (Register 28-3 through Register 28-6).

Note:	Data selections are undefined at power-up.

TABLE 28-1: CLCx DATA INPUT SELECTION

Data Input	dy DxS	CLCx
LCx_in[54]	110110	MD1_OUT OR MD2_OUT OR MD3_OUT
LCx_in[53]	110101	FOSC
LCx_in[52]	110100	HFINTOSC
LCx_in[51]	110011	LFINTOSC
LCx_in[50]	110010	FRC (ADC RC clock)
LCx_in[49]	110001	IOCIF set
LCx_in[48]	110000	Timer8_postscaled
LCx_in[47]	101111	Timer6_postscaled
LCx_in[46]	101110	Timer4_postscaled
LCx_in[45]	101101	Timer2_postscaled
LCx_in[44]	101100	Timer5 overflow
LCx_in[43]	101011	Timer3 overflow
LCx_in[42]	101010	Timer1 overflow
LCx_in[41]	101001	Timer0 overflow
LCx_in[40]	101000	EUSART RX
LCx_in[39]	100111	EUSART TX
LCx_in[38]	100110	ZCD1_output
LCx_in[37]	100101	MSSP1 SDO/SDA
LCx_in[36]	100100	MSSP1 SCL/SCK

TABLE 28-1: CLCx DATA INPUT SELECTION

Data Input	dy DxS	CLCx
LCx_in[35]	100011	PWM12_out(1)
LCx_in[34]	100010	PWM11_out
LCx_in[33]	100001	PWM6_out
LCx_in[32]	100000	PWM5_out
LCx_in[31]	011111	PWM10_out(1)
LCx_in[30]	011110	PWM9_out
LCx_in[29]	011101	PWM4_out
LCx_in[28]	011100	PWM3_out
LCx_in[27]	011011	CCP8_out(1)
LCx_in[26]	011010	CCP7_out
LCx_in[25]	011001	CCP2_out
LCx_in[24]	011000	CCP1_out
LCx_in[23]	010111	COG4B(1)
LCx_in[22]	010110	COG4A(1)
LCx_in[21]	010101	COG3B
LCx_in[20]	010100	COG3A
LCx_in[19]	010011	COG2B
LCx_in[18]	010010	COG2A
LCx_in[17]	010001	COG1B
LCx_in[16]	010000	COG1A
LCx_in[15]	001111	sync_C8OUT(1)
LCx_in[14]	001110	sync_C7OUT(1)
LCx_in[13]	001101	sync_C6OUT
LCx_in[12]	001100	sync_C5OUT
LCx_in[11]	001011	sync_C4OUT
LCx_in[10]	001010	sync_C3OUT
LCx_in[9]	001001	sync_C2OUT
LCx_in[8]	001000	sync_C1OUT
LCx_in[7]	000111	LC4_out from the CLC4
LCx_in[6]	000110	LC3_out from the CLC3
LCx_in[5]	000101	LC2_out from the CLC2
LCx_in[4]	000100	LC1_out from the CLC1
LCx_in[3]	000011	CLCIN3 pin input selected in CLCIN3PPS register
LCx_in[2]	000010	CLCIN2 pin input selected in CLCIN2PPS register
LCx_in[1]	000001	CLCIN1 pin input selected in CLCIN1PPS register
LCx_in[0]	000000	CLCIN0 pin input selected in CLCIN0PPS register

Note 1: PIC16(L)F1777/9 only.

32.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCL line low, effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCL.

The CKP bit of the SSPxCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. Setting CKP will release SCL and allow more communication.

32.5.6.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the R/$\overline{W}$ bit of SSPxSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPxBUF with data to transfer to the master. If the SEN bit of SSPxCON2 is set, the slave hardware will always stretch the clock after the $\overline{ACK}$ sequence. Once the slave is ready; CKP is set by software and communication resumes.

Note 1: The BF bit has no effect on if the clock will be stretched or not. This is different than previous versions of the module that would not stretch the clock, clear CKP, if SSPxBUF was read before the ninth falling edge of SCL.

2: Previous versions of the module did not stretch the clock for a transmission if SSPxBUF was loaded before the ninth falling edge of SCL. It is now always cleared for read requests.

32.5.6.2 10-bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set the clock is always stretched. This is the only time the SCL is stretched without CKP being cleared. SCL is released immediately after a write to SSPxADD.

Note: Previous versions of the module did not stretch the clock if the second address byte did not match.

32.5.6.3 Byte NACKing

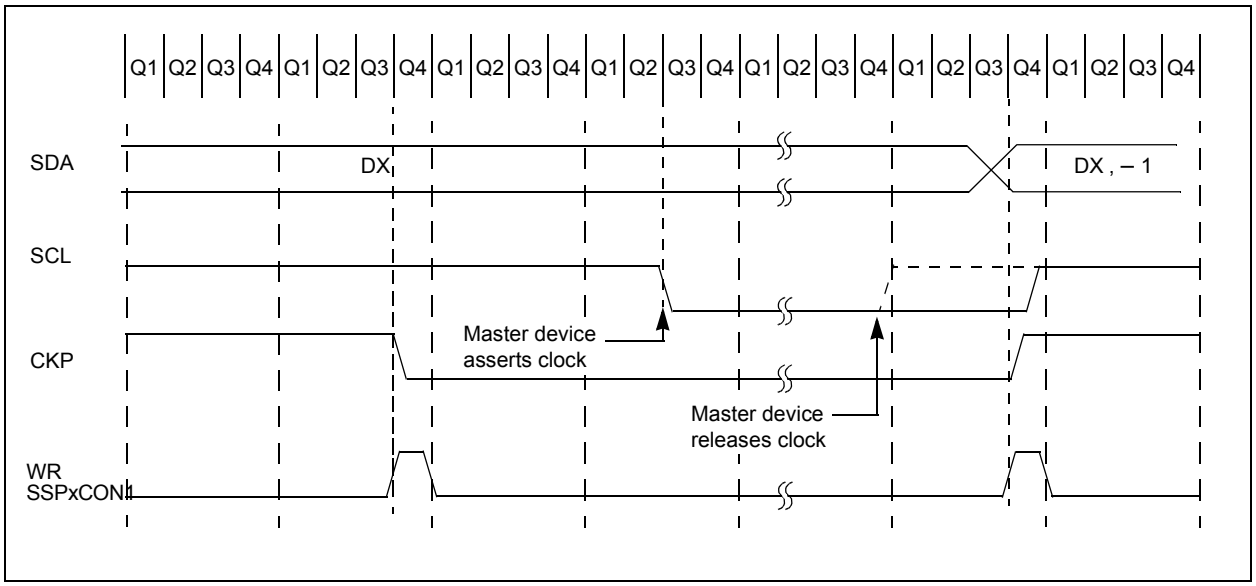
When AHEN bit of SSPxCON3 is set; CKP is cleared by hardware after the eighth falling edge of SCL for a received matching address byte. When DHEN bit of SSPxCON3 is set; CKP is cleared after the eighth falling edge of SCL for received data.

Stretching after the eighth falling edge of SCL allows the slave to look at the received address or data and decide if it wants to ACK the received data.

32.5.6.4 Clock Synchronization and the CKP Bit

Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I^2C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I^2C bus have released SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 32-23).

FIGURE 32-23: CLOCK SYNCHRONIZATION TIMING



33.1 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a VOH Mark state which represents a '1' data bit, and a VOL Space state which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the Mark state. Each character transmission consists of one Start bit followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is eight bits. Each transmitted bit persists for a period of 1/(baud rate). An on-chip dedicated 8-bit/16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See Table 33-5 for examples of baud rate configurations.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

33.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 33-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the TXxREG register.

33.1.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the TXEN bit of the TXxSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXxSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCxSTA register enables the EUSART and automatically configures the TX/CK I/O pin as an output. If the TX/CK pin is shared with an analog peripheral, the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

Note:	The TXIF Transmitter Interrupt flag is set when the TXEN enable bit is set.

33.1.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXxREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXxREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXxREG until the Stop bit of the previous character has been transmitted. The pending character in the TXxREG is then transferred to the TSR in one TCY immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXxREG.

33.1.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the SCKP bit of the BAUDxCON register. The default state of this bit is '0', which selects high true transmit idle and data bits. Setting the SCKP bit to '1' will invert the transmit data resulting in low true idle and data bits. The SCKP bit controls transmit data polarity in Asynchronous mode only. In Synchronous mode, the SCKP bit has a different function. See **Section 33.5.1.2 "Clock Polarity"**.

33.1.1.4 Transmit Interrupt Flag

The TXIF interrupt flag bit of the PIR1 register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXxREG. In other words, the TXIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXxREG. The TXIF flag bit is not cleared immediately upon writing TXxREG. TXIF becomes valid in the second instruction cycle following the write execution. Polling TXIF immediately following the TXxREG write will return invalid results. The TXIF bit is read-only, it cannot be set or cleared by software.

The TXIF interrupt can be enabled by setting the TXIE interrupt enable bit of the PIE1 register. However, the TXIF flag bit will be set whenever the TXxREG is empty, regardless of the state of the TXIE enable bit.

To use interrupts when transmitting data, set the TXIE bit only when there is more data to send. Clear the TXIE interrupt enable bit upon writing the last character of the transmission to the TXxREG.

TABLE 33-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	FOSC = 32.000 MHz			FOSC = 20.000 MHz			FOSC = 18.432 MHz			FOSC = 11.0592 MHz		
	Actual Rate	% Error	SPxBRG value (decimal)	Actual Rate	% Error	SPxBRG value (decimal)	Actual Rate	% Error	SPxBRG value (decimal)	Actual Rate	% Error	SPxBRG value (decimal)
300	300.0	0.00	26666	300.0	0.00	16665	300.0	0.00	15359	300.0	0.00	9215
1200	1200	0.00	6666	1200	-0.01	4166	1200	0.00	3839	1200	0.00	2303
2400	2400	0.01	3332	2400	0.02	2082	2400	0.00	1919	2400	0.00	1151
9600	9604	0.04	832	9597	-0.03	520	9600	0.00	479	9600	0.00	287
10417	10417	0.00	767	10417	0.00	479	10425	0.08	441	10433	0.16	264
19.2k	19.18k	-0.08	416	19.23k	0.16	259	19.20k	0.00	239	19.20k	0.00	143
57.6k	57.55k	-0.08	138	57.47k	-0.22	86	57.60k	0.00	79	57.60k	0.00	47
115.2k	115.9k	0.64	68	116.3k	0.94	42	115.2k	0.00	39	115.2k	0.00	23

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	FOSC = 8.000 MHz			FOSC = 4.000 MHz			FOSC = 3.6864 MHz			FOSC = 1.000 MHz		
	Actual Rate	% Error	SPxBRG value (decimal)	Actual Rate	% Error	SPxBRG value (decimal)	Actual Rate	% Error	SPxBRG value (decimal)	Actual Rate	% Error	SPxBRG value (decimal)
300	300.0	0.00	6666	300.0	0.01	3332	300.0	0.00	3071	300.1	0.04	832
1200	1200	-0.02	1666	1200	0.04	832	1200	0.00	767	1202	0.16	207
2400	2401	0.04	832	2398	0.08	416	2400	0.00	383	2404	0.16	103
9600	9615	0.16	207	9615	0.16	103	9600	0.00	95	9615	0.16	25
10417	10417	0	191	10417	0.00	95	10473	0.53	87	10417	0.00	23
19.2k	19.23k	0.16	103	19.23k	0.16	51	19.20k	0.00	47	19.23k	0.16	12
57.6k	57.14k	-0.79	34	58.82k	2.12	16	57.60k	0.00	15	—	—	—
115.2k	117.6k	2.12	16	111.1k	-3.55	8	115.2k	0.00	7	—	—	—

MOVWI **Move W to INDFn**

Syntax: [*label*] MOVWI ++FSRn
[*label*] MOVWI --FSRn
[*label*] MOVWI FSRn++
[*label*] MOVWI FSRn--
[*label*] MOVWI k[FSRn]

Operands: n ∈ [0,1]
mm ∈ [00,01, 10, 11]
-32 ≤ k ≤ 31

Operation: W → INDFn
Effective address is determined by
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be either:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)

Unchanged

Status Affected: None

Mode	Syntax	mm
Preincrement	++FSRn	00
Predecrement	--FSRn	01
Postincrement	FSRn++	10
Postdecrement	FSRn--	11

Description: This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.

Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.

FSRn is limited to the range 0000h-FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap-around.

The increment/decrement operation on FSRn WILL NOT affect any Status bits.

NOP **No Operation**

Syntax: [*label*] NOP

Operands: None

Operation: No operation

Status Affected: None

Description: No operation.

Words: 1

Cycles: 1

Example:
```
NOP
```

OPTION **Load OPTION_REG Register with W**

Syntax: [*label*] OPTION

Operands: None

Operation: (W) → OPTION_REG

Status Affected: None

Description: Move data from W register to OPTION_REG register.

Words: 1

Cycles: 1

Example:
```
OPTION
```
Before Instruction
OPTION_REG = 0xFF
W = 0x4F
After Instruction
OPTION_REG = 0x4F
W = 0x4F

RESET **Software Reset**

Syntax: [*label*] RESET

Operands: None

Operation: Execute a device Reset. Resets the $\overline{RI}$ flag of the PCON register.

Status Affected: None

Description: This instruction provides a way to execute a hardware Reset by software.

TABLE 36-27: I^2C BUS DATA REQUIREMENTS

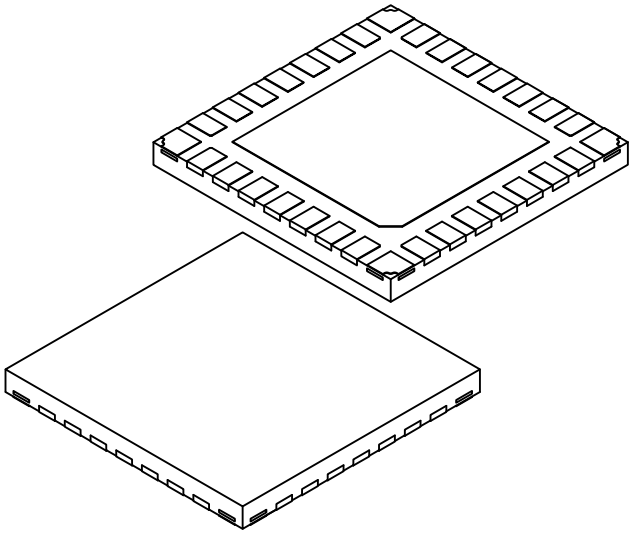
Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Symbol**	**Characteristic**		**Min.**	**Max.**	**Units**	**Conditions**
SP100*	THIGH	Clock high time	100 kHz mode	4.0	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	μs	Device must operate at a minimum of 10 MHz
			SSP module	1.5TCY	—		
SP101*	TLOW	Clock low time	100 kHz mode	4.7	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	μs	Device must operate at a minimum of 10 MHz
			SSP module	1.5TCY	—		
SP102*	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1CB	300	ns	CB is specified to be from 10-400 pF
SP103*	TF	SDA and SCL fall time	100 kHz mode	—	250	ns	
			400 kHz mode	20 + 0.1CB	250	ns	CB is specified to be from 10-400 pF
SP106*	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
SP107*	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
SP109*	TAA	Output valid from clock	100 kHz mode	—	3500	ns	**(Note 1)**
			400 kHz mode	—	—	ns	
SP110*	TBUF	Bus free time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
SP111	CB	Bus capacitive loading		—	400	pF	

* These parameters are characterized but not tested.

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

2: A Fast mode (400 kHz) I^2C bus device can be used in a Standard mode (100 kHz) I^2C bus system, but the requirement $T_{SU:DAT} \geq 250$ ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line T_R max. + $T_{SU:DAT}$ = 1000 + 250 = 1250 ns (according to the Standard mode I^2C bus specification), before the SCL line is released.

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	0.40	0.50	0.60
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	(A3)		0.127 REF	
Overall Width	E		6.00 BSC	
Exposed Pad Width	E2		4.00	
Overall Length	D		6.00 BSC	
Exposed Pad Length	D2		4.00	
Terminal Width	b	0.35	0.40	0.45
Corner Pad	b1	0.55	0.60	0.65
Corner Pad, Metal Free Zone	b2	0.15	0.20	0.25
Terminal Length	L	0.55	0.60	0.65
Terminal-to-Exposed Pad	K	0.20	-	-

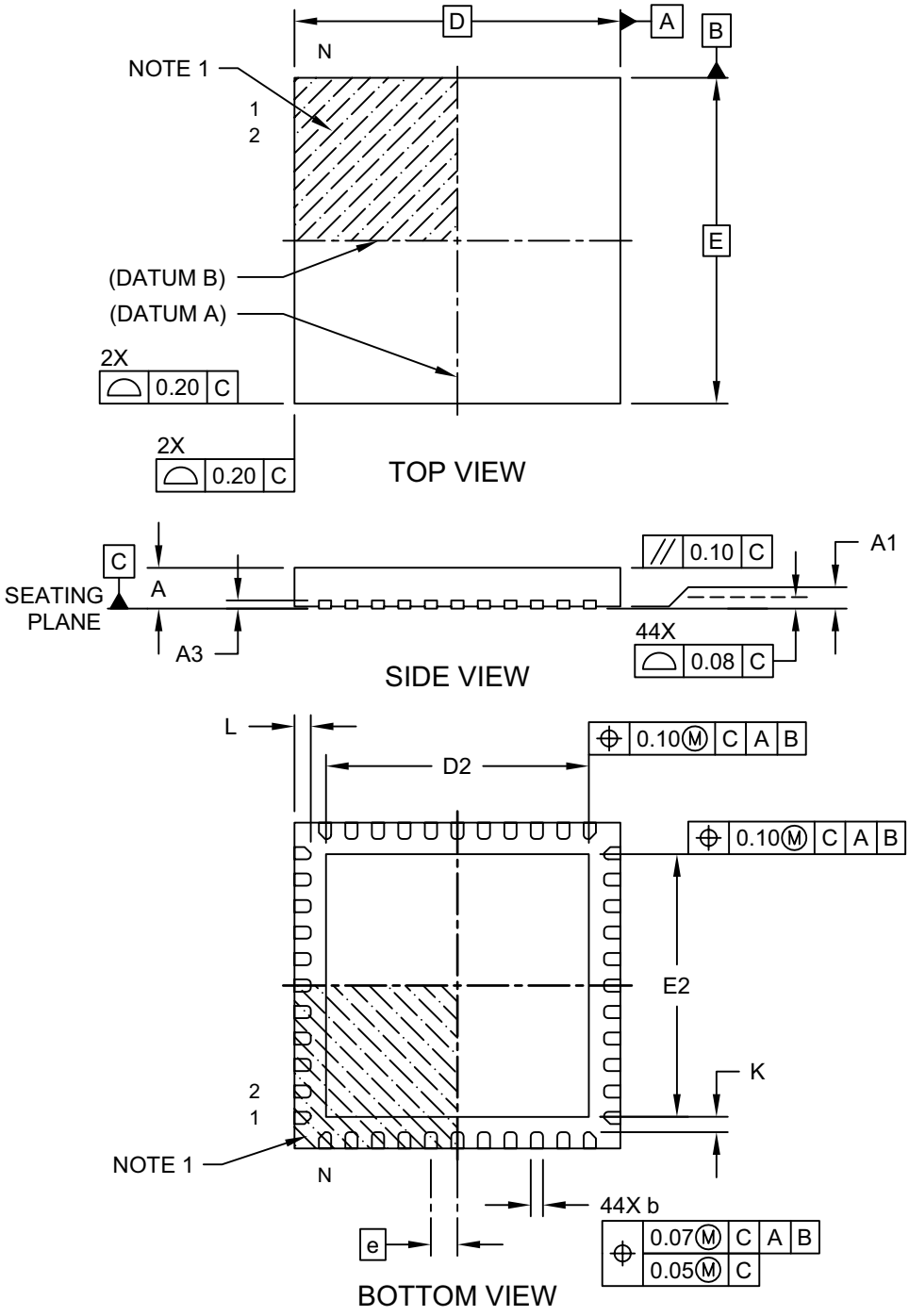
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
4. Outermost portions of corner structures may vary slightly.

Microchip Technology Drawing C04-0209 Rev C Sheet 2 of 2

44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-103D Sheet 1 of 2