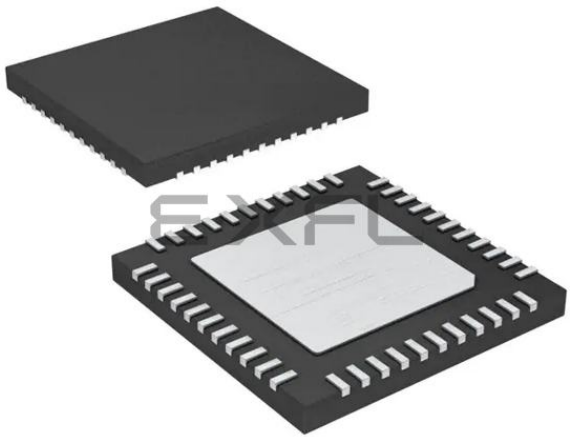


Welcome to E-XFL.COM

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	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f47k40t-i-ml

TABLE 1: 28-PIN ALLOCATION TABLE (PIC18(L)F27K40) (CONTINUED)

I/O(2)	28-Pin SPDIP, SOIC, SSOP	28-Pin QFN	A/D	Reference	Comparator	Timers	CCP	CWG	ZCD	Interrupt	EUSART	DSM	MSSP	Pull-up	Basic
RC0	11	8	ANC0	—	—	T1CKI(1) T3CKI(1) T3G(1)	—	—	—	IOCC0	—	—	—	Y	SOSCO
RC1	12	9	ANC1	—	—	—	CCP2(1)	—	—	IOCC1	—	—	—	Y	SOSCIN SOSCI
RC2	13	10	ANC2	—	—	T5CKI(1)	CCP1(1)	—	—	IOCC2	—	—	—	Y	—
RC3	14	11	ANC3	—	—	T2AIN(1)	—	—	—	IOCC3	—	—	SCK1(1) SCL1(3,4)	Y	—
RC4	15	12	ANC4	—	—	—	—	—	—	IOCC4	—	—	SDI1(1) SDA1(3,4)	Y	—
RC5	16	13	ANC5	—	—	T4AIN(1)	—	—	—	IOCC5	—	—	—	Y	—
RC6	17	14	ANC6	—	—	—	—	—	—	IOCC6	CK1(1)	—	—	Y	—
RC7	18	15	ANC7	—	—	—	—	—	—	IOCC7	RX1/DT1(1)	—	—	Y	—
RE3	1	26	—	—	—	—	—	—	—	IOCE3	—	—	—	Y	VPP/$\overline{\text{MCLR}}$
VSS	19	16	—	—	—	—	—	—	—	—	—	—	—	—	VSS
VDD	20	17	—	—	—	—	—	—	—	—	—	—	—	—	VDD
VSS	8	5	—	—	—	—	—	—	—	—	—	—	—	—	VSS
OUT(2)	—	—	ADGRDA ADGRDB	—	C1OUT C2OUT	TMR0	CCP1 CCP2 PWM3 PWM4	CWG1A CWG1B CWG1C CWG1D	—	—	TX1/CK1(3) DT1(3) TX2/CK2(3) DT2(3)	DSM	SDO1 SCK1 SDO2 SCK2	—	—

Note 1: Default peripheral input. Input can be moved to any other pin with the PPS input selection registers (Register 17-1).

2: All pin outputs default to PORT latch data. Any pin can be selected as a peripheral digital output with the PPS output selection registers.

3: These peripheral functions are bidirectional. The output pin selections must be the same as the input pin selections.

4: These pins are configured for I^2C logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins (e.g., RB1) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I^2C specific or SMBus input buffer thresholds.

10.2 Register Definitions: Stack Pointer

REGISTER 10-1: STKPTR: STACK POINTER REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	STKPTR<4:0>				
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented	C = Clearable only bit
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **STKPTR<4:0>:** Stack Pointer Location bits

10.2.1 STACK OVERFLOW AND UNDERFLOW RESETS

Device Resets on Stack Overflow and Stack Underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a Full or Underflow condition will set the appropriate STKOVF or STKUNF bit and then cause a device Reset. When STVREN is cleared, a Full or Underflow condition will set the appropriate STKOVF or STKUNF bit but not cause a device Reset. The STKOVF or STKUNF bits are cleared by the user software or a Power-on Reset.

10.2.2 FAST REGISTER STACK

A fast register stack is provided for the Status, WREG and BSR registers, to provide a "fast return" option for interrupts. The stack for each register is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the `RETFIE, FAST` instruction is used to return from the interrupt.

If both low and high priority interrupts are enabled, the stack registers cannot be used reliably to return from low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten. In these cases, users must save the key registers by software during a low priority interrupt.

If interrupt priority is not used, all interrupts may use the fast register stack for returns from interrupt. If no interrupts are used, the fast register stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a `CALL label, FAST` instruction must be executed to save the `STATUS`, WREG and BSR registers to the fast register stack. A `RETURN, FAST` instruction is then executed to restore these registers from the fast register stack.

Example 10-1 shows a source code example that uses the fast register stack during a subroutine call and return.

EXAMPLE 10-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
       •
       •
       •
SUB1   •
       •
       •
   RETURN, FAST    ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

TABLE 10-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F27/47K40 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FD7h	PCON0	FAFh	T6TMR	F87h	LATE(2)
FFEh	TOSH	FD6h	T0CON1	FAEh	CCPTMRS	F86h	LATD(2)
FFDh	TOSL	FD5h	T0CON0	FADh	CCP1CAP	F85h	LATC
FFCh	STKPTR	FD4h	TMR0H	FACh	CCP1CON	F84h	LATB
FFBh	PCLATU	FD3h	TMR0L	FABh	CCP1H	F83h	LATA
FFAh	PCLATH	FD2h	T1CLK	FAAh	CCP1L	F82h	NVMCON2
FF9h	PCL	FD1h	T1GATE	FA9h	CCP2CAP	F81h	NVMCON1
FF8h	TBLPTRU	FD0h	T1GCON	FA8h	CCP2CON	F80h	NVMDAT
FF7h	TBLPTRH	FCFh	T1CON	FA7h	CCP2H	F7Fh	NVMADRH
FF6h	TBLPTRL	FCEh	TMR1H	FA6h	CCP2L	F7Eh	NVMADRL
FF5h	TABLAT	FCDh	TMR1L	FA5h	PWM3CON	F7Dh	CRCCON1
FF4h	PRODH	FCCh	T3CLK	FA4h	PWM3DCH	F7Ch	CRCCON0
FF3h	PRODL	FCBh	T3GATE	FA3h	PWM3DCL	F7Bh	CRCXORH
FF2h	INTCON	FCAh	T3GCON	FA2h	PWM4CON	F7Ah	CRCXORL
FF1h	—	FC9h	T3CON	FA1h	PWM4DCH	F79h	CRCSHIFTH
FF0h	—	FC8h	TMR3H	FA0h	PWM4DCL	F78h	CRCSHIFTL
FEFh	INDF0(1)	FC7h	TMR3L	F9Fh	BAUD1CON	F77h	CRCACCH
FEEh	POSTINC0(1)	FC6h	T5CLK	F9Eh	TX1STA	F76h	CRCACCL
FEDh	POSTDEC0(1)	FC5h	T5GATE	F9Dh	RC1STA	F75h	CRCDATH
FECh	PREINC0(1)	FC4h	T5GCON	F9Ch	SP1BRGH	F74h	CRCDATL
FEBh	PLUSW0(1)	FC3h	T5CON	F9Bh	SP1BRGL	F73h	ADFLTRH
FEAh	FSR0H	FC2h	TMR5H	F9Ah	TX1REG	F72h	ADFLTRL
FE9h	FSR0L	FC1h	TMR5L	F99h	RC1REG	F71h	ADACCH
FE8h	WREG	FC0h	T2RST	F98h	SSP1CON3	F70h	ADACCL
FE7h	INDF1(1)	FBFh	T2CLKCON	F97h	SSP1CON2	F6Fh	ADERRH
FE6h	POSTINC1(1)	FBEh	T2HLT	F96h	SSP1CON1	F6Eh	ADERRL
FE5h	POSTDEC1(1)	FBDh	T2CON	F95h	SSP1STAT	F6Dh	ADUTHH
FE4h	PREINC1(1)	FBCh	T2PR	F94h	SSP1MSK	F6Ch	ADUTHL
FE3h	PLUSW1(1)	FBBh	T2TMR	F93h	SSP1ADD	F6Bh	ADLTHH
FE2h	FSR1H	FBAh	T4RST	F92h	SSP1BUF	F6Ah	ADLTHL
FE1h	FSR1L	FB9h	T4CLKCON	F91h	PORTE	F69h	ADSTPTH
FE0h	BSR	FB8h	T4HLT	F90h	PORTD(2)	F68h	ADSTPTL
FDFh	INDF2(1)	FB7h	T4CON	F8Fh	PORTC	F67h	ADCNT
FDEh	POSTINC2(1)	FB6h	T4PR	F8Eh	PORTB	F66h	ADRPT
FDDh	POSTDEC2(1)	FB5h	T4TMR	F8Dh	PORTA	F65h	ADSTAT
FDCh	PREINC2 (1)	FB4h	T6RST	F8Ch	TRISE(2)	F64h	ADRESH
FDBh	PLUSW2(1)	FB3h	T6CLKCON	F8Bh	TRISD(2)	F63h	ADRESL
FDAh	FSR2H	FB2h	T6HLT	F8Ah	TRISC	F62h	ADPREVH
FD9h	FSR2L	FB1h	T6CON	F89h	TRISB	F61h	ADPREVL
FD8h	STATUS	FB0h	T6PR	F88h	TRISA	F60h	ADCON0

Note 1: This is not a physical register.
2: Not available on PIC18(L)F27K40 (28-pin variants).

11.1 Program Flash Memory

The Program Flash Memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed one byte at a time. A write to program memory or program memory erase is executed on blocks of *n* bytes at a time. Refer to Table 11-3 for write and erase block sizes. A Bulk Erase operation cannot be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

It is important to understand the PFM memory structure for erase and programming operations. Program memory word size is 16 bits wide. PFM is arranged in rows. A row is the minimum size that can be erased by user software. Refer to Table 11-3 for the row sizes for the these devices.

After a row has been erased, all or a portion of this row can be programmed. Data to be written into the program memory row is written to 6-bit wide data write latches. These latches are not directly accessible, but may be loaded via sequential writes to the TABLAT register.

Note:	To modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, the new data and retained data can be written into the write latches to reprogram the row of PFM. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations

TABLE 11-2: FLASH MEMORY ORGANIZATION BY DEVICE

Device	Row Erase Size (Words)	Write Latches (Bytes)	Program Flash Memory (Words)	Data Memory (Bytes)
PIC18(L)F45K40	32	64	16384	256
PIC18(L)F26K40			32768	1024
PIC18(L)F46K40				
PIC18(L)F27K40	64	128	65536	
PIC18(L)F47K40				

REGISTER 14-5: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3

R-0/0	R-0/0	R-0/0	R-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **RC2IF:** EUSART2 Receive Interrupt Flag bit
- 1 = The EUSART2 receive buffer, RC1REG, is full (cleared by reading RC2REG)
- 0 = The EUSART2 receive buffer is empty

bit 6 **TX2IF:** EUSART2 Transmit Interrupt Flag bit
- 1 = The EUSART2 transmit buffer, TX2REG, is empty (cleared by writing TX2REG)
- 0 = The EUSART2 transmit buffer is full

bit 5 **RC1IF:** EUSART1 Receive Interrupt Flag bit
- 1 = The EUSART1 receive buffer, RC1REG, is full (cleared by reading RC1REG)
- 0 = The EUSART1 receive buffer is empty

bit 4 **TX1IF:** EUSART1 Transmit Interrupt Flag bit
- 1 = The EUSART1 transmit buffer, TX1REG, is empty (cleared by writing TX1REG)
- 0 = The EUSART1 transmit buffer is full

bit 3 **BCL2IF:** MSSP2 Bus Collision Interrupt Flag bit
- 1 = A bus collision has occurred while the MSSP2 module configured in I^2C master was transmitting (must be cleared in software)
- 0 = No bus collision occurred

bit 2 **SSP2IF:** Synchronous Serial Port 2 Interrupt Flag bit
- 1 = The transmission/reception is complete (must be cleared in software)
- 0 = Waiting to transmit/receive

bit 1 **BCL1IF:** MSSP1 Bus Collision Interrupt Flag bit
- 1 = A bus collision has occurred while the MSSP1 module configured in I^2C master was transmitting (must be cleared in software)
- 0 = No bus collision occurred

bit 0 **SSP1IF:** Synchronous Serial Port 1 Interrupt Flag bit
- 1 = The transmission/reception is complete (must be cleared in software)
- 0 = Waiting to transmit/receive

REGISTER 14-12: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
HLVDIE	ZCDIE	—	—	—	—	C2IE	C1IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **HLVDIE:** HLVD Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **ZCDIE:** Zero-Cross Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5-2 **Unimplemented:** Read as '0'

bit 1 **C2IE:** Comparator 2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **C1IE:** Comparator 1 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 14-22: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4

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
—	—	TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

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

bit 5 **TMR6IP:** TMR6 to PR6 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TMR5IP:** TMR5 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **TMR4IP:** TMR4 to PR4 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

TABLE 14-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

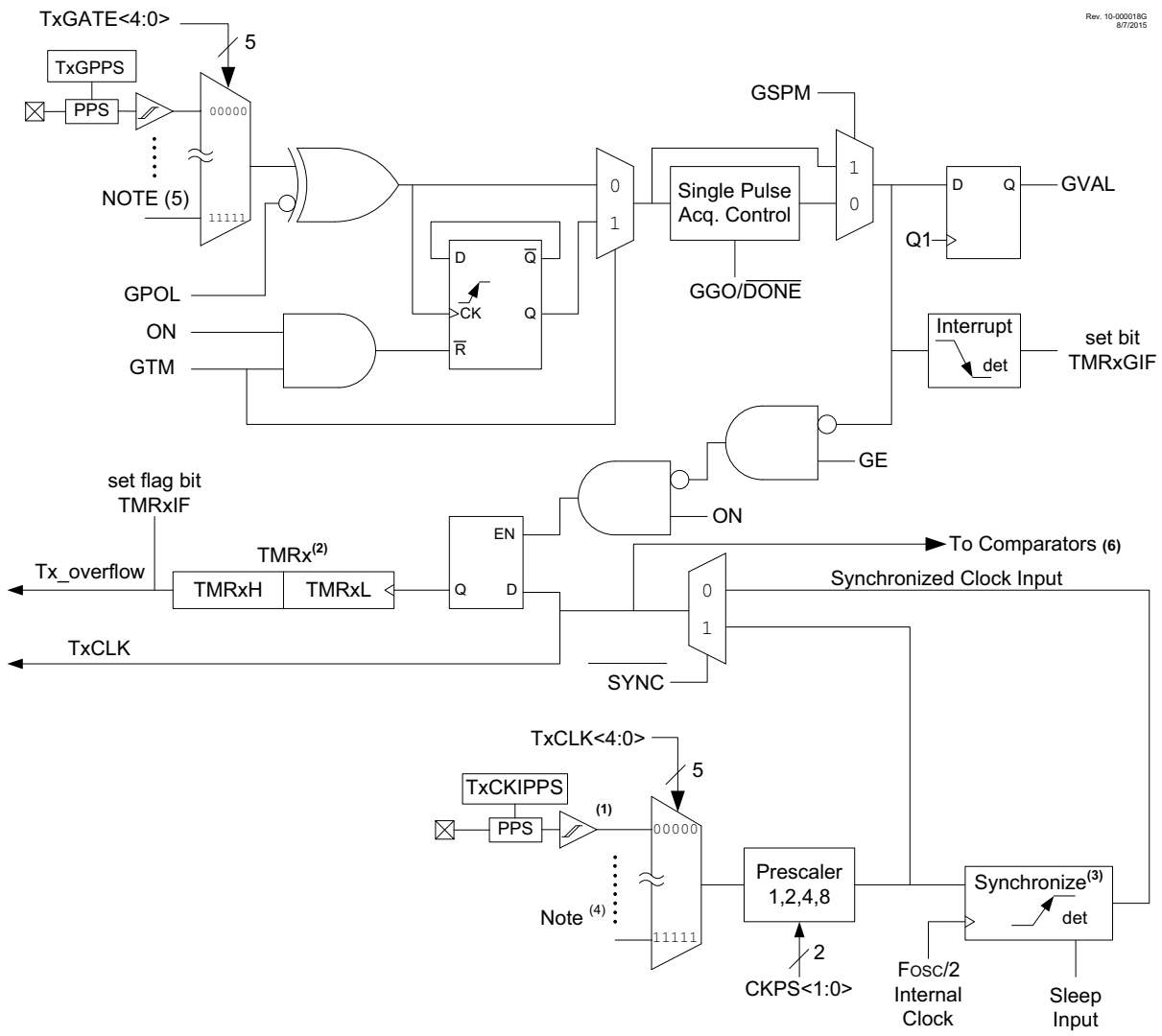
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	170
PIE0	—	—	TMR0IE	IOCIE	—	INT2IE	INT1IE	INT0IE	179
PIE1	OSCFIE	CSWIE	—	—	—	—	ADTIE	ADIE	180
PIE2	HLVDIE	ZCDIE	—	—	—	—	C2IE	C1IE	181
PIE3	RC2IE	TX2IE	RC1IE	TX1IE	BCL2IE	SSP2IE	BCL1IE	SSP1IE	182
PIE4	—	—	TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE	183
PIE5	—	—	—	—	—	TMR5GIE	TMR3GIE	TMR1GIE	184
PIE6	—	—	—	—	—	—	CCP2IE	CCP1IE	185
PIE7	SCANIE	CRCIE	NVMIE	—	—	—	—	CWG1IE	186
PIR0	—	—	TMR0IF	IOCIF	—	INT2IF	INT1IF	INT0IF	171
PIR1	OSCFIF	CSWIF	—	—	—	—	ADTIF	ADIF	172
PIR2	HLVDIF	ZCDIF	—	—	—	—	C2IF	C1IF	173
PIR3	RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF	174
PIR4	—	—	TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF	175
PIR5	—	—	—	—	—	TMR5GIF	TMR3GIF	TMR1GIF	176
PIR6	—	—	—	—	—	—	CCP2IF	CCP1IF	177
PIR7	SCANIF	CRCIF	NVMIF	—	—	—	—	CWG1IF	178
IPR0	—	—	TMR0IP	IOCIP	—	INT2IP	INT1IP	INT0IP	187
IPR1	OSCFIP	CSWIP	—	—	—	—	ADTIP	ADIP	188
IPR2	HLVDIP	ZCDIP	—	—	—	—	C2IP	C1IP	189
IPR3	RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP	190
IPR4	—	—	TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP	191
IPR5	—	—	—	—	—	TMR5GIP	TMR3GIP	TMR1GIP	192
IPR6	—	—	—	—	—	—	CCP2IP	CCP1IP	193
IPR7	SCANIP	CRCIP	NVMIP	—	—	—	—	CWG1IP	194

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for Interrupts.

TABLE 17-1: PPS INPUT REGISTER DETAILS

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	Input Available from Selected PORTx						
				PIC18(L)F27K40			PIC18(L)F47K40			
Interrupt 0	INT0PPS	RB0	5'b0 1000	A	B	—	A	B	—	—
Interrupt 1	INT1PPS	RB1	5'b0 1001	A	B	—	A	B	—	—
Interrupt 2	INT2PPS	RB2	5'b0 1010	A	B	—	A	B	—	—
Timer0 Clock	T0CKIPPS	RA4	5'b0 0100	A	B	—	A	B	—	—
Timer1 Clock	T1CKIPPS	RC0	5'b1 0000	A	—	C	A	—	C	—
Timer1 Gate	T1GPPS	RB5	5'b0 1101	—	B	C	—	B	C	—
Timer3 Clock	T3CKIPPS	RC0	5'b1 0000	—	B	C	—	B	C	—
Timer3 Gate	T3GPPS	RC0	5'b1 0000	A	—	C	A	—	C	—
Timer5 Clock	T5CKIPPS	RC2	5'b1 0010	A	—	C	A	—	C	—
Timer5 Gate	T5GPPS	RB4	5'b0 1100	—	B	C	—	B	—	D
Timer2 Clock	T2INPPS	RC3	5'b1 0011	A	—	C	A	—	C	—
Timer4 Clock	T4INPPS	RC5	5'b1 0101	—	B	C	—	B	C	—
Timer6 Clock	T6INPPS	RB7	5'b0 1111	—	B	C	—	B	—	D
CCP1	CCP1PPS	RC2	5'b1 0010	—	B	C	—	B	C	—
CCP2	CCP2PPS	RC1	5'b1 0001	—	B	C	—	B	C	—
CWG	CWG1PPS	RB0	5'b0 1000	—	B	C	—	B	—	D
DSM Carrier Low	MDCARLPPS	RA3	5'b0 0011	A	—	C	A	—	—	D
DSM Carrier High	MDCARHPPS	RA4	5'b0 0100	A	—	C	A	—	—	D
DSM Source	MDSRCPPS	RA5	5'b0 0101	A	—	C	A	—	—	D
ADC Conversion Trigger	ADACTPPS	RB4	5'b0 1100	—	B	C	—	B	—	D
MSSP1 Clock	SSP1CLKPPS	RC3	5'b1 0011	—	B	C	—	B	C	—
MSSP1 Data	SSP1DATPPS	RC4	5'b1 0100	—	B	C	—	B	C	—
MSSP1 Slave Select	SSP1SSPPS	RA5	5'b0 0101	A	—	C	A	—	—	D
MSSP2 Clock	SSP2CLKPPS	RB1	5'b0 1001	—	B	C	—	B	—	D
MSSP2 Data	SSP2DATPPS	RB2	5'b0 1010	—	B	C	—	B	—	D
MSSP2 Slave Select	SSP2SSPPS	RB0	5'b0 1000	—	B	C	—	B	—	D
EUSART1 Receive	RX1PPS	RC7	5'b1 0111	—	B	C	—	B	C	—
EUSART1 Transmit	TX1PPS	RC6	5'b1 0110	—	B	C	—	B	C	—
EUSART2 Receive	RX2PPS	RB7	5'b0 1111	—	B	C	—	B	—	D
EUSART2 Transmit	TX2PPS	RB6	5'b0 1110	—	B	C	—	B	—	D

FIGURE 19-1: TIMER1/3/5 BLOCK DIAGRAM



Note 1: ST Buffer is high speed type when using TxCKIPPS.

2: TMRx register increments on rising edge.

3: Synchronize does not operate while in Sleep.

4: See Register 19-3 for clock source selections.

5: See Register 19-4 for gate source selection.

6: Synchronized comparator output should not be used in conjunction with synchronized input clock.

19.7 Timer1/3/5 16-Bit Read/Write Mode

Timer1/3/5 can be configured to read and write all 16 bits of data, to and from, the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the RD16 bit of the TxCON register.

To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-Bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

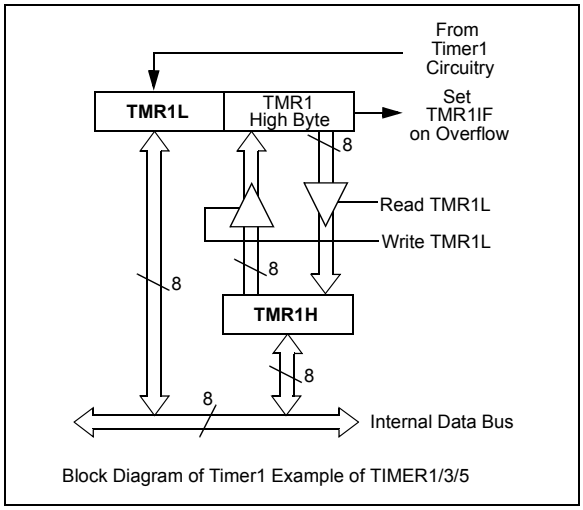
When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1/3/5 value from a single instance in time. Reference the block diagram in Figure 19-2 for more details.

In contrast, when not in 16-Bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRxL:TMRxH register pair at the same time.

Any requests to write to the TMRxH directly does not clear the Timer1/3/5 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.

FIGURE 19-2: TIMER1/3/5 16-BIT READ/WRITE MODE BLOCK DIAGRAM



Block Diagram of Timer1 Example of TIMER1/3/5

19.8 Timer1/3/5 Gate

Timer1/3/5 can be configured to count freely or the count can be enabled and disabled using Timer1/3/5 gate circuitry. This is also referred to as Timer1/3/5 gate enable.

Timer1/3/5 gate can also be driven by multiple selectable sources.

19.8.1 TIMER1/3/5 GATE ENABLE

The Timer1/3/5 Gate Enable mode is enabled by setting the TMRxGE bit of the TxGCON register. The polarity of the Timer1/3/5 Gate Enable mode is configured using the TxGPOL bit of the TxGCON register.

When Timer1/3/5 Gate Enable mode is enabled, Timer1/3/5 will increment on the rising edge of the Timer1/3/5 clock source. When Timer1/3/5 Gate signal is inactive, the timer will not increment and hold the current count. Enable mode is disabled, no incrementing will occur and Timer1/3/5 will hold the current count. See Figure 19-4 for timing details.

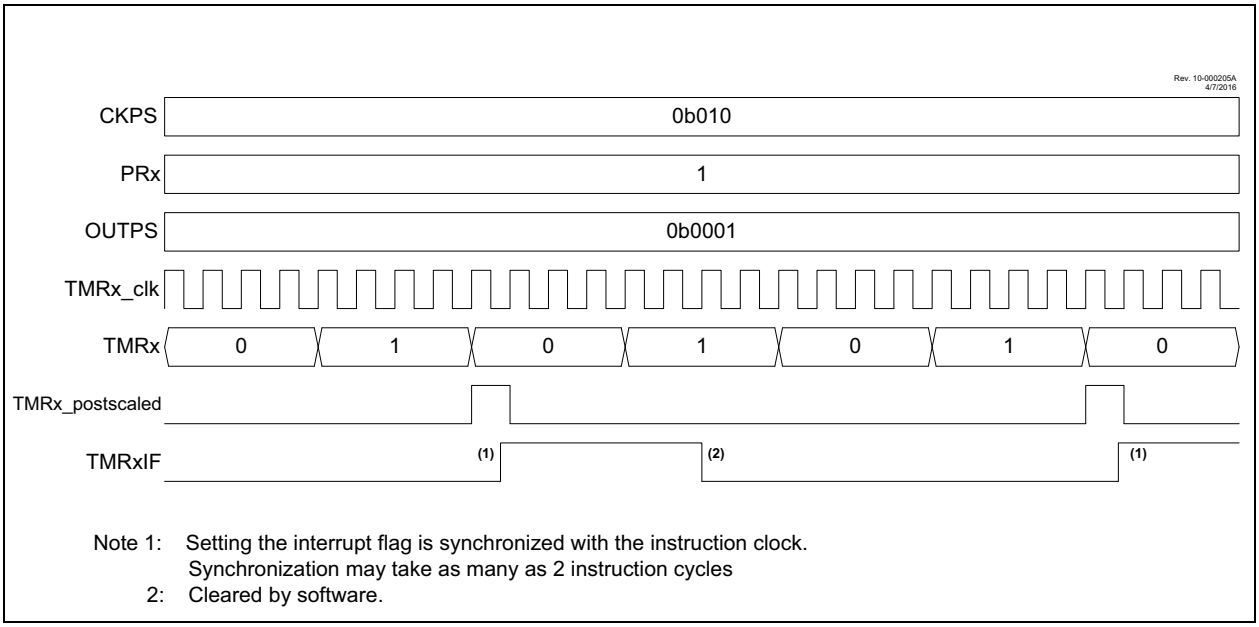
TABLE 19-3: TIMER1/3/5 GATE ENABLE SELECTIONS

TMRxCLK	TxGPOL	TxG	Timer1/3/5 Operation
↑	1	1	Counts
↑	1	0	Holds Count
↑	0	1	Holds Count
↑	0	0	Counts

20.4 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches one of 16 postscale options (from 1:1 through 1:16), which are selected with the postscaler control bits, OUTPS<3:0> of the T2CON register. The interrupt is enabled by setting the TMR2IE interrupt enable bit of the PIE4 register. Interrupt timing is illustrated in Figure 20-3.

FIGURE 20-3: TIMER2 PRESCALER, POSTSCALER, AND INTERRUPT TIMING DIAGRAM



Note 1: Setting the interrupt flag is synchronized with the instruction clock. Synchronization may take as many as 2 instruction cycles
2: Cleared by software.

REGISTER 21-5: CCPRxH: CCPx REGISTER HIGH BYTE

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0

MODE = Capture Mode:
CCPRxH<7:0>: MSB of captured TMR1 value
MODE = Compare Mode:
CCPRxH<7:0>: MSB compared to TMR1 value
MODE = PWM Mode && FMT = 0:
CCPRxH<7:2>: Not used
CCPRxH<1:0>: CCPW<9:8> – Pulse-Width MS 2 bits
MODE = PWM Mode && FMT = 1:
CCPRxH<7:0>: CCPW<9:2> – Pulse-Width MS 8 bits

REGISTER 24-7: CWG1AS1: CWG AUTO-SHUTDOWN CONTROL REGISTER 1

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
—	—	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
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 **AS5E:** CWG Auto-shutdown Source 5 (CMP2 OUT) Enable bit
- 1 = Auto-shutdown for CMP2 OUT is enabled
- 0 = Auto-shutdown for CMP2 OUT is disabled

bit 4 **AS4E:** CWG Auto-shutdown Source 4 (CMP1 OUT) Enable bit
- 1 = Auto-shutdown for CMP1 OUT is enabled
- 0 = Auto-shutdown for CMP1 OUT is disabled

bit 3 **AS3E:** CWG Auto-shutdown Source 3 (TMR6_Postscaled) Enable bit
- 1 = Auto-shutdown for TMR6_Postscaled is enabled
- 0 = Auto-shutdown for TMR6_Postscaled is disabled

bit 2 **AS2E:** CWG Auto-shutdown Source 2 (TMR4_Postscaled) Enable bit
- 1 = Auto-shutdown for TMR4_Postscaled is enabled
- 0 = Auto-shutdown for TMR4_Postscaled is disabled

bit 1 **AS1E:** CWG Auto-shutdown Source 1 (TMR2_Postscaled) Enable bit
- 1 = Auto-shutdown for TMR2_Postscaled is enabled
- 0 = Auto-shutdown for TMR2_Postscaled is disabled

bit 0 **AS0E:** CWG Auto-shutdown Source 0 (Pin selected by CWG1PPS) Enable bit
- 1 = Auto-shutdown for CWG1PPS Pin is enabled
- 0 = Auto-shutdown for CWG1PPS Pin is disabled

The transition of a data bit is always performed while the SCL line is held low. Transitions that occur while the SCL line is held high are used to indicate Start and Stop bits.

If the master intends to write to the slave, then it repeatedly sends out a byte of data, with the slave responding after each byte with an ACK bit. In this example, the master device is in Master Transmit mode and the slave is in Slave Receive mode.

If the master intends to read from the slave, then it repeatedly receives a byte of data from the slave, and responds after each byte with an ACK bit. In this example, the master device is in Master Receive mode and the slave is Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop bit. If the master device is in Receive mode, it sends the Stop bit in place of the last ACK bit. A Stop bit is indicated by a low-to-high transition of the SDA line while the SCL line is held high.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send another Start bit in place of the Stop bit or last ACK bit when it is in receive mode.

The I^2C bus specifies three message protocols;

- Single message where a master writes data to a slave.
- Single message where a master reads data from a slave.
- Combined message where a master initiates a minimum of two writes, or two reads, or a combination of writes and reads, to one or more slaves.

When one device is transmitting a logical one, or letting the line float, and a second device is transmitting a logical zero, or holding the line low, the first device can detect that the line is not a logical one. This detection, when used on the SCL line, is called clock stretching. Clock stretching gives slave devices a mechanism to control the flow of data. When this detection is used on the SDA line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

26.6.1 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCL connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

FIGURE 33-2: LOW-VOLTAGE DETECT OPERATION (HLVDINTL = 1)

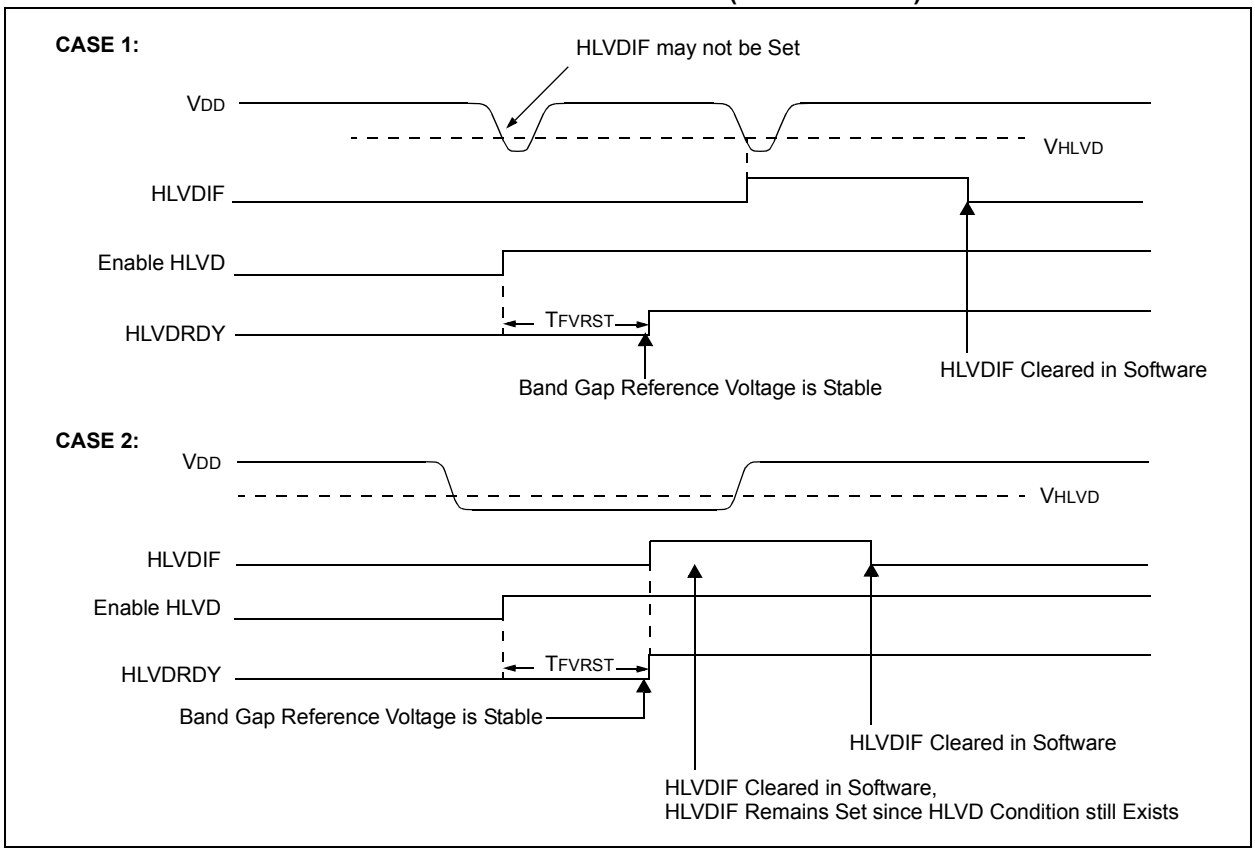
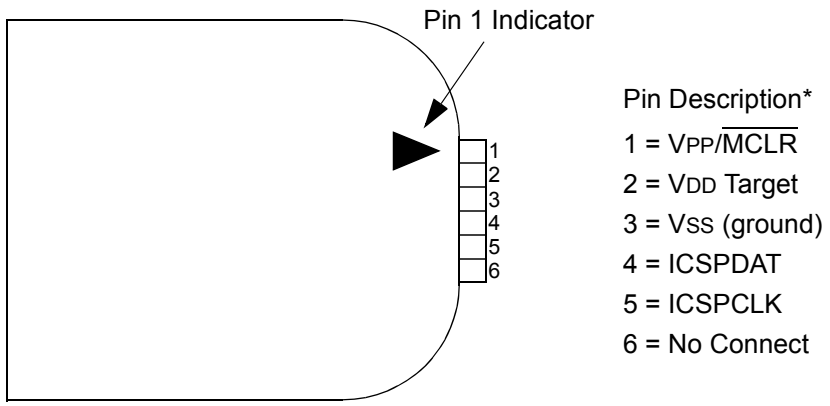
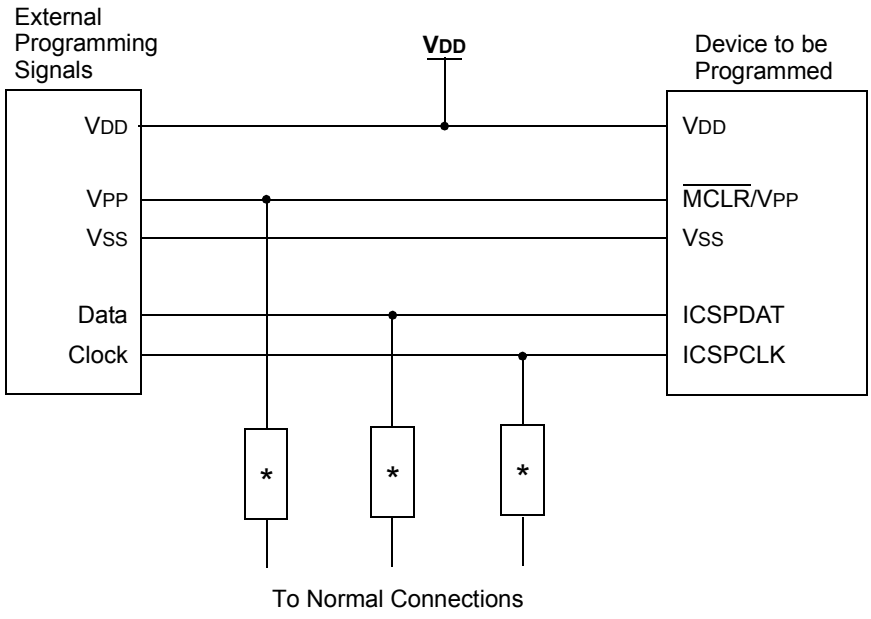


FIGURE 34-2: PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE



* The 6-pin header (0.100" spacing) accepts 0.025" square pins.

FIGURE 34-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING



* Isolation devices (as required).

MOVSS **Move Indexed to Indexed**

Syntax: MOVSS [z_s], [z_d]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: ((FSR2) + z_s) → ((FSR2) + z_d)

Status Affected: None

Encoding:

1st word (source)	1110	1011	1zzz	$zzzz_s$
2nd word (dest.)	1111	xxxx	xzzz	$zzzz_d$

Description: The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets 'z_s' or 'z_d', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).
The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	Determine dest addr	Determine dest addr	Write to dest reg

Example: MOVSS [05h], [06h]

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h
After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

PUSHL **Store Literal at FSR2, Decrement FSR2**

Syntax: PUSHL k

Operands: $0 \le k \le 255$

Operation: k → (FSR2),
FSR2 – 1 → FSR2

Status Affected: None

Encoding:

1111	1010	kkkk	kkkk

Description: The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read 'k'	Process data	Write to destination

Example: PUSHL 08h

Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h

36.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

36.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

36.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

36.5 MPLAB Assembler, Linker and Librarian for Various Device Families

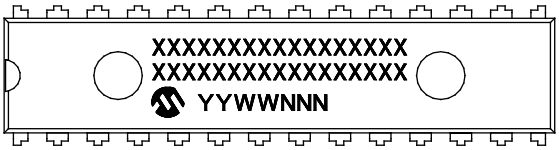
MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

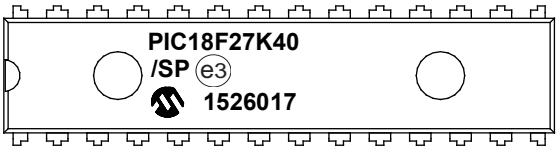
39.0 PACKAGING INFORMATION

Package Marking Information

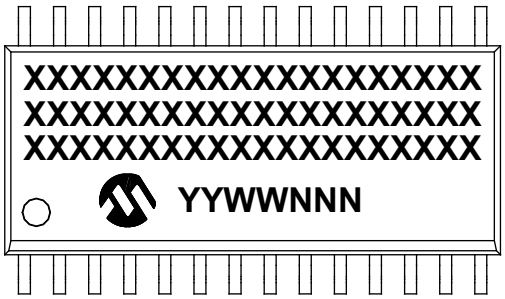
28-Lead SPDIP (.300”)



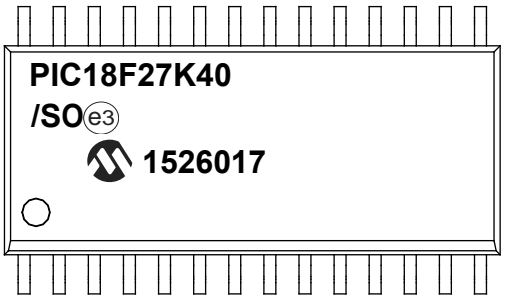
Example



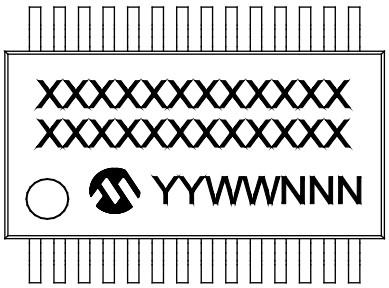
28-Lead SOIC (7.50 mm)



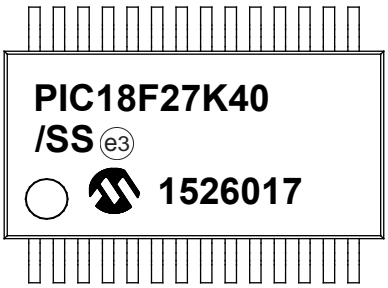
Example



28-Lead SSOP (5.30 mm)



Example



Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC® designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.