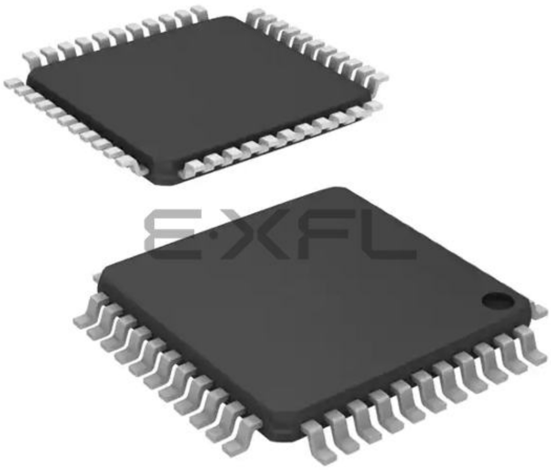


What is "Embedded - Microcontrollers"?

"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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, Power Control PWM, QEI, POR, PWM, WDT
Number of I/O	36
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 9x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf4431t-i-pt

NOTES:

6.5 Data Memory Organization

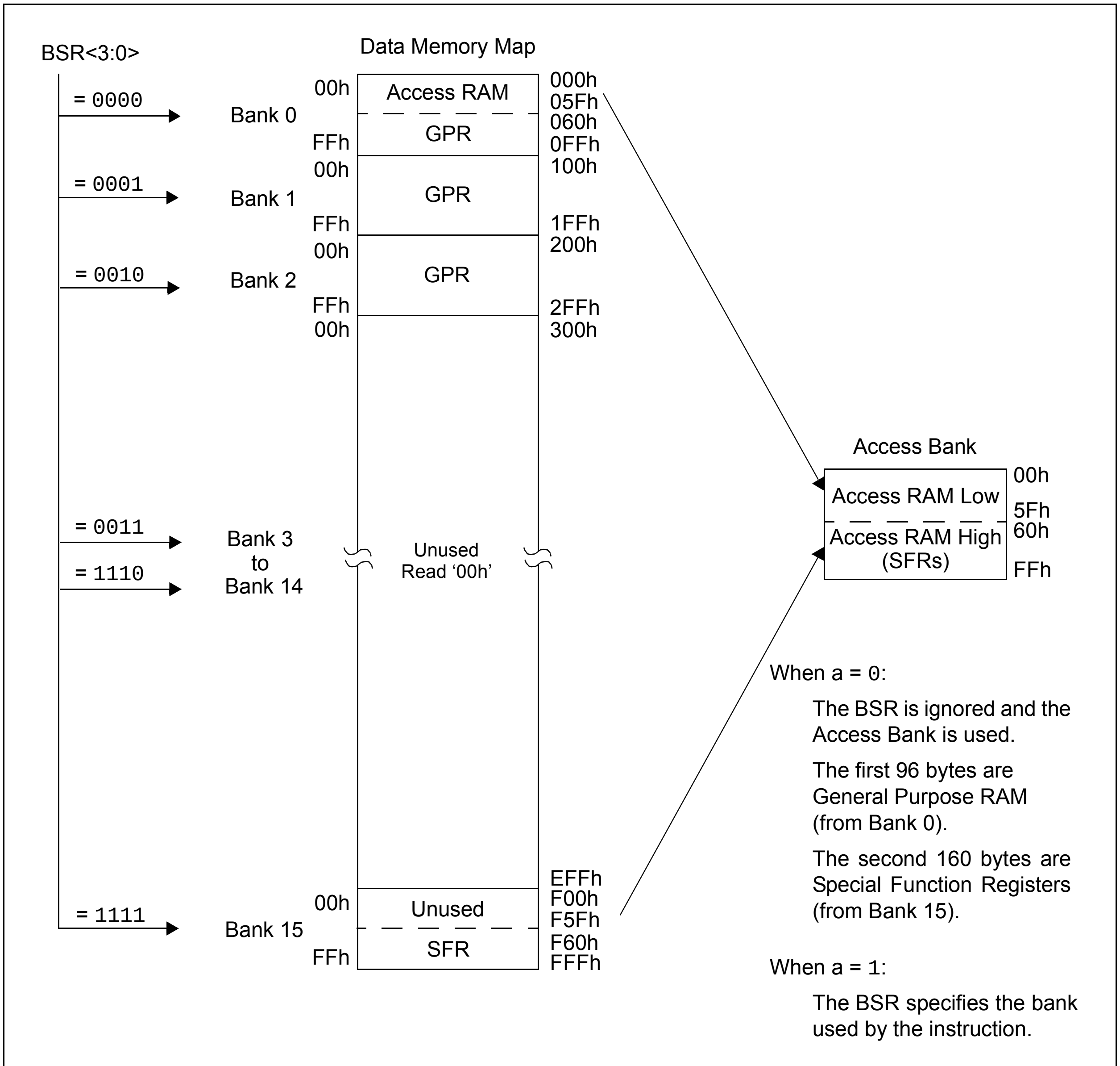
The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4,096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. PIC18F2331/2431/4331/4431 devices implement all 16 banks.

Figure 6-6 shows the data memory organization for the PIC18F2331/2431/4331/4431 devices. The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 6.5.2 "Access Bank"** provides a detailed description of the Access RAM.

FIGURE 6-6: DATA MEMORY MAP FOR PIC18F2331/2431/4331/4431 DEVICES



6.5.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 6-1 and Table 6-2.

The SFRs can be classified into two sets: those associated with the “core” function and those related to the peripheral functions. Those registers related to the “core” are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature.

The SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations will be unimplemented and read as ‘0’s.

TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2331/2431/4331/4431 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1	F7Fh	PTCON0
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1	F7Eh	PTCON1
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1	F7Dh	PTMRL
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	—(2)	F7Ch	PTMRH
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE	F7Bh	PTPERL
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	ADCON3	F7Ah	PTPERH
FF9h	PCL	FD9h	FSR2L	FB9h	ANSEL1	F99h	ADCHS	F79h	PDC0L
FF8h	TBLPTRU	FD8h	STATUS	FB8h	ANSEL0	F98h	—(2)	F78h	PDC0H
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	T5CON	F97h	—(2)	F77h	PDC1L
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	QEICON	F96h	TRISE(3)	F76h	PDC1H
FF5h	TABLAT	FD5h	T0CON	FB5h	—(2)	F95h	TRISD(3)	F75h	PDC2L
FF4h	PRODH	FD4h	—(2)	FB4h	—(2)	F94h	TRISC	F74h	PDC2H
FF3h	PRODL	FD3h	OSCCON	FB3h	—(2)	F93h	TRISB	F73h	PDC3L(3)
FF2h	INTCON	FD2h	LVDCON	FB2h	—(2)	F92h	TRISA	F72h	PDC3H(3)
FF1h	INTCON2	FD1h	WDTCON	FB1h	—(2)	F91h	PR5H	F71h	SEVTCMPL
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	PR5L	F70h	SEVTCMPH
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—(2)	F6Fh	PWMCON0
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—(2)	F6Eh	PWMCON1
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(3)	F6Dh	DTCON
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(3)	F6Ch	FLTCONFIG
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC	F6Bh	OVDCOND
FEAh	FSR0H	FCAh	T2CON	FAAh	BAUDCON	F8Ah	LATB	F6Ah	OVDCONS
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA	F69h	CAP1BUFH
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	TMR5H	F68h	CAP1BUFL
FE7h	INDF1(1)	FC7h	SSPSTAT	FA7h	EECON2	F87h	TMR5L	F67h	CAP2BUFH
FE6h	POSTINC1(1)	FC6h	SSPCON	FA6h	EECON1	F86h	—(2)	F66h	CAP2BUFL
FE5h	POSTDEC1(1)	FC5h	—(2)	FA5h	IPR3	F85h	—(2)	F65h	CAP3BUFH
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE	F64h	CAP3BUFL
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD(3)	F63h	CAP1CON
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	CAP2CON
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	CAP3CON
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA	F60h	DFLTCON

Note 1: This is not a physical register.
2: Unimplemented registers are read as ‘0’.
3: This register is not available on 28-pin devices.

NOTES:

10.2 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Request (Flag) Registers (PIR1, PIR2 and PIR3).

> **Note 1:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE (INTCON<7>).
>
> **2:** User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

REGISTER 10-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
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 **Unimplemented:** Read as ‘0’

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF:** EUSART Receive Interrupt Flag bit
1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The EUSART receive buffer is empty

bit 4 **TXIF:** EUSART Transmit Interrupt Flag bit
1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The EUSART transmit buffer is full

bit 3 **SSPIF:** Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

REGISTER 10-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PTIE	IC3DRIE	IC2QEIE	IC1IE	TMR5IE
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 7-5 **Unimplemented:** Read as ‘0’

bit 4 **PTIE:** PWM Time Base Interrupt Enable bit
1 = PTIF enabled
0 = PTIF disabled

bit 3 **IC3DRIE:** IC3 Interrupt Enable/Direction Change Interrupt Enable bit
IC3 Enabled (CAP3CON<3:0>):
1 = IC3 interrupt enabled
0 = IC3 interrupt disabled
QEI Enabled (QEIM<2:0>):
1 = Change of direction interrupt enabled
0 = Change of direction interrupt disabled

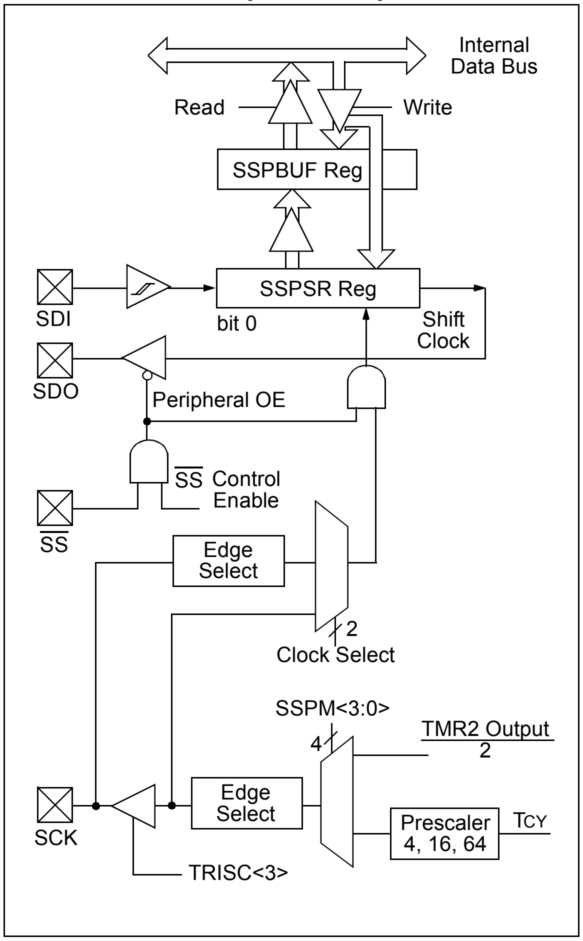
bit 2 **IC2QEIE:** IC2 Interrupt Flag/QEI Interrupt Flag Enable bit
IC2 Enabled (CAP2CON<3:0>):
1 = IC2 interrupt enabled)
0 = IC2 interrupt disabled
QEI Enabled (QEIM<2:0>):
1 = QEI interrupt enabled
0 = QEI interrupt disabled

bit 1 **IC1IE:** IC1 Interrupt Enable bit
1 = IC1 interrupt enabled
0 = IC1 interrupt disabled

bit 0 **TMR5IE:** Timer5 Interrupt Enable bit
1 = Timer5 interrupt enabled
0 = Timer5 interrupt disabled

NOTES:

FIGURE 19-1: SSP BLOCK DIAGRAM (SPI MODE)



To enable the serial port, SSP Enable bit, SSPEN (SSPCON<5>), must be set. To reset or reconfigure SPI mode, clear bit SSPEN, reinitialize the SSPCON register and then set bit SSPEN. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, they must have their data direction bits (in the TRISC register) appropriately programmed. That is:

- Serial Data Out (SDO) – RC7/RX/DT/SDO or RD1/SDO
- SDI must have TRISC<4> or TRISD<2> set
- SDO must have TRISC<7> or TRISD<1> cleared
- SCK (Master mode) must have TRISC<5> or TRISD<3> cleared
- SCK (Slave mode) must have TRISC<5> or TRISD<3> set
- $\overline{SS}$ must have TRISA<6> set

Note 1: When the SPI is in Slave mode, with the $\overline{SS}$ pin control enabled, (SSPCON<3:0> = `0100`), the SPI module will reset if the $\overline{SS}$ pin is set to VDD.

2: If the SPI is used in Slave mode with CKE = `1`, then the $\overline{SS}$ pin control must be enabled.

3: When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON<3:0> = `0100`), the state of the $\overline{SS}$ pin can affect the state read back from the TRISC<6> bit. The peripheral OE signal from the SSP module into PORTC controls the state that is read back from the TRISC<6> bit (see **Section 11.3 "PORTC, TRISC and LATC Registers"** for information on PORTC). If Read-Modify-Write instructions, such as BSF, are performed on the TRISC register while the $\overline{SS}$ pin is high, this will cause the TRISC<6> bit to be set, thus disabling the SDO output.

20.2 EUSART Baud Rate Generator (BRG)

The BRG is a dedicated 8-bit or 16-bit generator, that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit (BAUDCON<3>) selects 16-bit mode.

The SPBRGH:SPBRG register pair controls the period of a free-running timer. In Asynchronous mode, bits BRGH (TXSTA<2>) and BRG16 also control the baud rate. In Synchronous mode, bit BRGH is ignored. Table 20-1 shows the formula for computation of the baud rate for different EUSART modes, which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 20-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 20-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 20-2. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG, to reduce the baud rate error or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGH:SPBRG registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

20.2.1 POWER-MANAGED MODE OPERATION

The system clock is used to generate the desired baud rate. However, when a power-managed mode is entered, the clock source may be operating at a different frequency than in PRI_RUN mode. In Sleep mode, no clocks are present and in PRI_IDLE, the primary clock source continues to provide clocks to the Baud Rate Generator. However, in other power-managed modes, the clock frequency will probably change. This may require the value in SPBRG to be adjusted.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RCIDL bit and make sure that the receive operation is Idle before changing the system clock.

20.2.2 SAMPLING

The data on the RC7/RX/DT/SDO pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 20-1: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	**BRG16**	**BRGH**		
0	0	0	8-Bit/Asynchronous	FOSC/[64 (n + 1)]
0	0	1	8-Bit/Asynchronous	FOSC/[16 (n + 1)]
0	1	0	16-Bit/Asynchronous	
0	1	1	16-Bit/Asynchronous	FOSC/[4 (n + 1)]
1	0	x	8-Bit/Synchronous	
1	1	x	16-Bit/Synchronous	

Legend: x = Don’t care, n = value of SPBRGH:SPBRG register pair

FIGURE 20-2: EUSART TRANSMIT BLOCK DIAGRAM

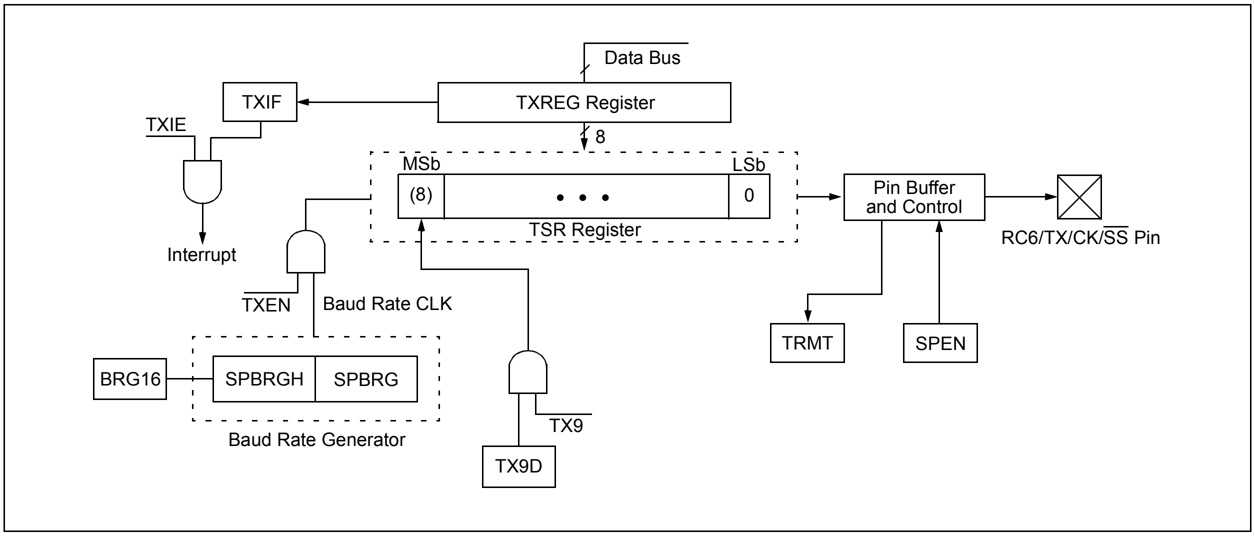


FIGURE 20-3: ASYNCHRONOUS TRANSMISSION

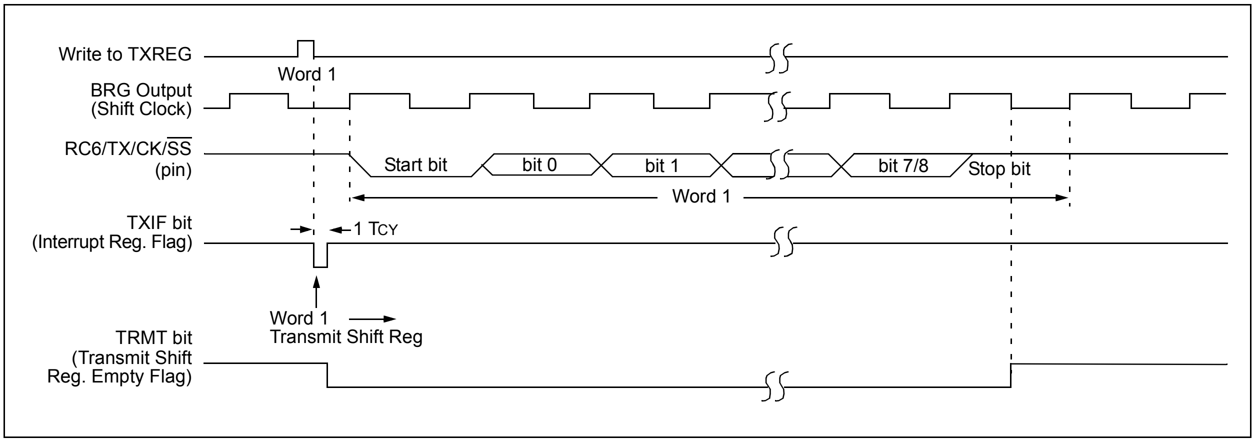
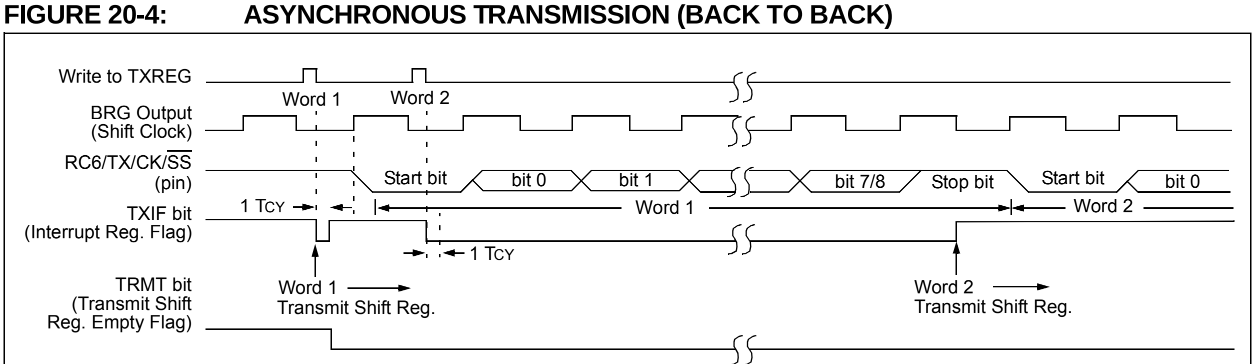


FIGURE 20-4: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

20.3.5 BREAK CHARACTER SEQUENCE

The Enhanced USART module has the capability of sending the special Break character sequences that are required by the LIN/J2602 bus standard. The Break character transmit consists of a Start bit, followed by twelve ‘0’ bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all ‘0’s will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN/J2602 specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 20-9 for the timing of the Break character sequence.

20.3.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN/J2602 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to setup the Break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write ‘55h’ to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

20.3.6 RECEIVING A BREAK CHARACTER

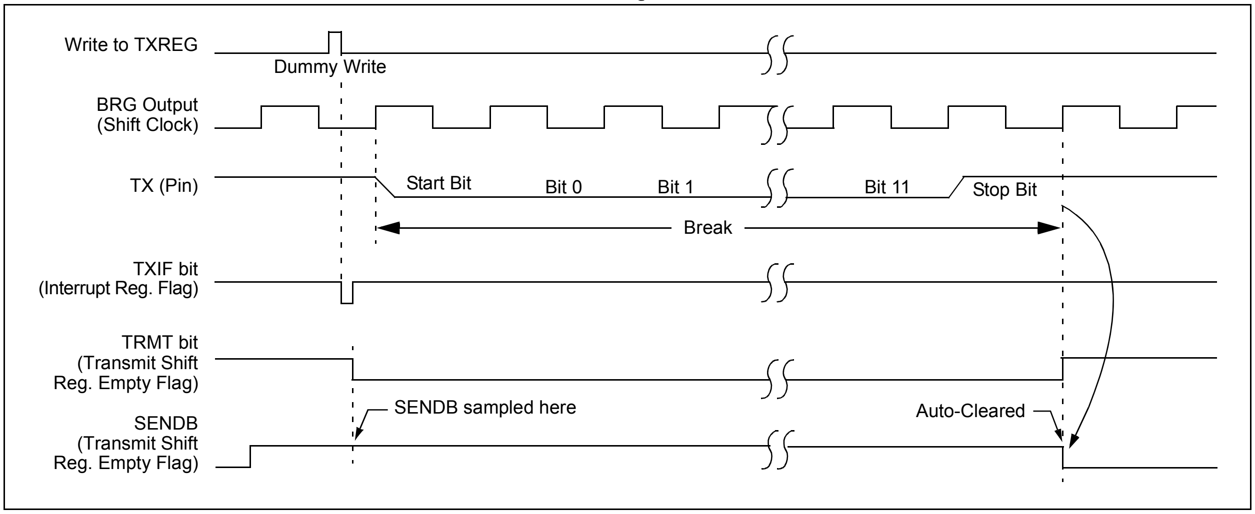
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 of the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 20.3.4 “Auto-Wake-up on Sync Break Character”**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABD bit before placing the EUSART in its Sleep mode.

FIGURE 20-9: SEND BREAK CHARACTER SEQUENCE



24.2 Instruction Set

ADDLW — ADD Literal to W

Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) + k → W
Status Affected:	N, OV, C, DC, Z
Encoding:	`0000` `1111` `kkkk` `kkkk`
Description:	The contents of W are added to the 8-bit literal 'k' and the result is placed in W.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `ADDLW 0x15`

Before Instruction
W = 0x10
After Instruction
W = 0x25

ADDWF — ADD W to f

Syntax:	[*label*] ADDWF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) + (f) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0010` `01da` `ffff` `ffff`
Description:	Add W to register, 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register, 'f'. If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR is used.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWF REG, W`

Before Instruction
W = 0x17
REG = 0xC2
After Instruction
W = 0xD9
REG = 0xC2

RETURN — **Return from Subroutine**

Syntax: [*label*] RETURN [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC;
if s = 1:
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	0000	0001	001s

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If ‘s’= 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If ‘s’ = 0, no update of these registers occurs.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	POP PC from stack
No operation	No operation	No operation	No operation

Example: `RETURN`

After Interrupt
PC = TOS

RLCF — **Rotate Left f through Carry**

Syntax: [*label*] RLCF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

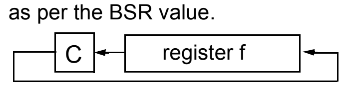
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

0011	01da	ffff	ffff

Description: The contents of register, ‘f’, are rotated one bit to the left through the Carry flag. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register, ‘f’. If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

Example: `RLCF REG, W`

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
W = 1100 1100
C = 1

RLNCF — Rotate Left f (No Carry)

Syntax:	[*label*] RLNCF f [,d [,a]]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<n>) → dest<n + 1>, (f<7>) → dest<0>
Status Affected:	N, Z
Encoding:	`0100` `01da` `ffff` `ffff`
Description:	The contents of register, 'f', are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register, 'f'. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value.

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RLNCF REG`

Before Instruction
REG = `1010 1011`

After Instruction
REG = `0101 0111`

RRCF — Rotate Right f through Carry

Syntax:	[*label*] RRCF f [,d [,a]]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<n>) → dest<n – 1>, (f<0>) → C, (C) → dest<7>
Status Affected:	C, N, Z
Encoding:	`0011` `00da` `ffff` `ffff`
Description:	The contents of register, 'f', are rotated one bit to the right through the Carry Flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register, 'f'. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value.

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RRCF REG, W`

Before Instruction
REG = `1110 0110`
C = `0`

After Instruction
REG = `1110 0110`
W = `0111 0011`
C = `0`

FIGURE 26-6: CLKO AND I/O TIMING

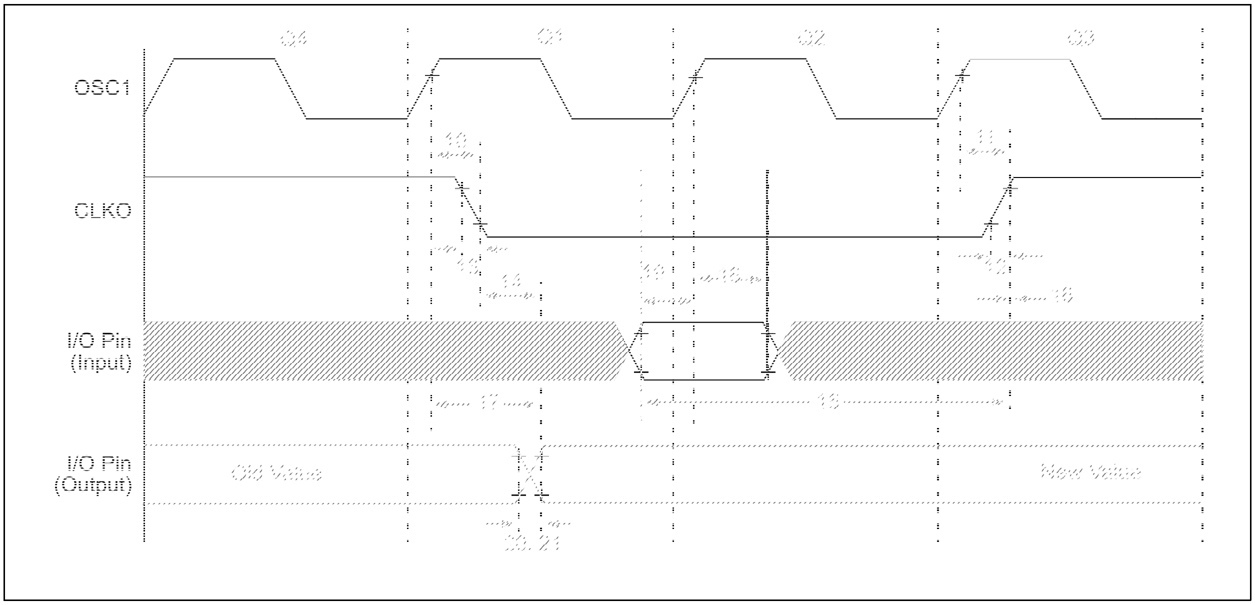


TABLE 26-7: CLKO AND I/O TIMING REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1 ↑ to CLKO ↓		—	75	200	ns	**(Note 1)**
11	TosH2ckH	OSC1 ↑ to CLKO ↑		—	75	200	ns	**(Note 1)**
12	TckR	CLKO Rise Time		—	35	100	ns	**(Note 1)**
13	TckF	CLKO Fall Time		—	35	100	ns	**(Note 1)**
14	TckL2ioV	CLKO ↓ to Port Out Valid		—	—	0.5 TCY + 20	ns	**(Note 1)**
15	TioV2ckH	Port In Valid before CLKO ↑		0.25 TCY + 25	—	—	ns	**(Note 1)**
16	TckH2ioI	Port In Hold after CLKO ↑		0	—	—	ns	**(Note 1)**
17	TosH2ioV	OSC1 ↑ (Q1 cycle) to Port Out Valid		—	50	150	ns	
18	TosH2ioI	OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time)	PIC18FXX31	100	—	—	ns	
18A			PIC18LFXX31	200	—	—	ns	
19	TioV2osH	Port Input Valid to OSC1 ↑ (I/O in setup time)		0	—	—	ns	
20	TioR	Port Output Rise Time	PIC18FXX31	—	10	25	ns	
20A			PIC18LFXX31	—	—	60	ns	
21	TioF	Port Output Fall Time	PIC18FXX31	—	10	25	ns	
21A			PIC18LFXX31	—	—	60	ns	
22†	TINP	INTx Pin High or Low Time		TCY	—	—	ns	
23†	TRBP	RB<7:4> Change INTx High or Low Time		TCY	—	—	ns	

† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode, where CLKO output is 4 x TOSC.

FIGURE 26-11: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

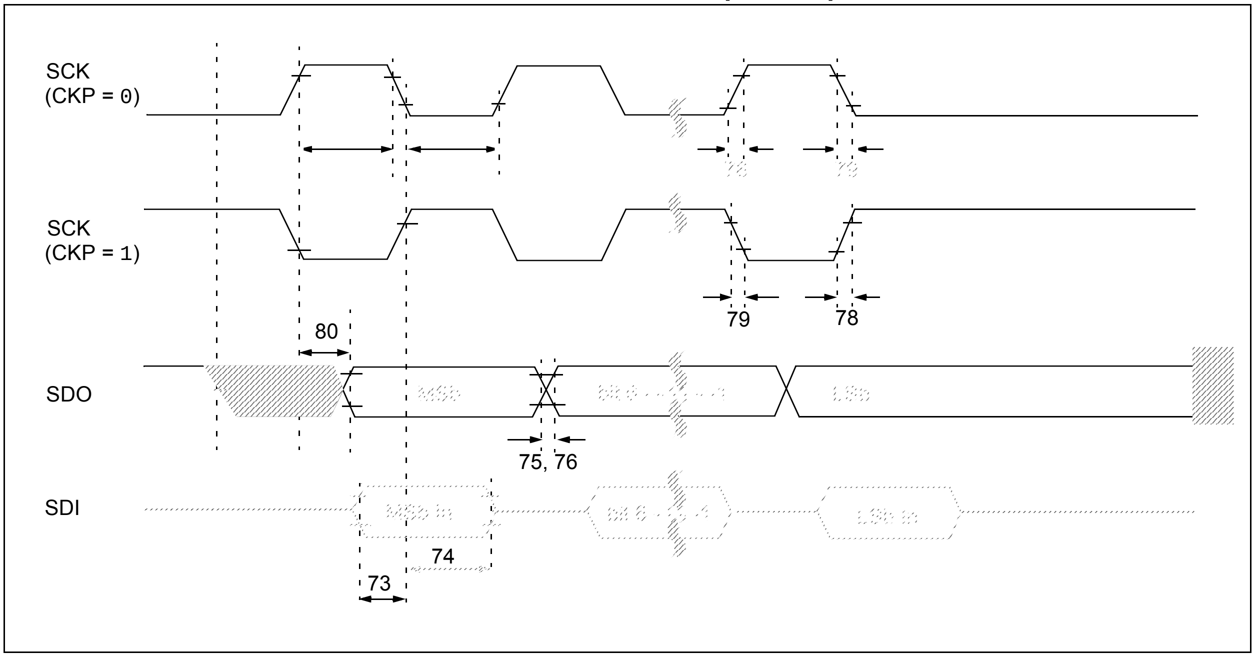
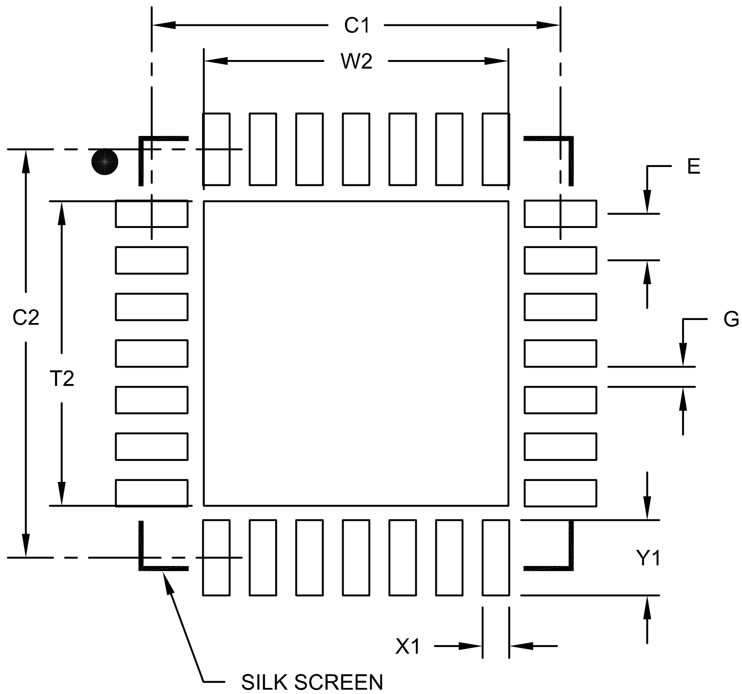


TABLE 26-11: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
73	TdiV2scH, TdiV2scL	Setup Time of SDI Data Input to SCK Edge		20	—	ns	
73A	Tb2b	Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2		1.5 TCY + 40	—	ns	
74	TscH2diL, TscL2diL	Hold Time of SDI Data Input to SCK Edge		40	—	ns	
75	TdoR	SDO Data Output Rise Time	PIC18FXX31	—	25	ns	
			PIC18LFXX31	—	45	ns	
76	TdoF	SDO Data Output Fall Time		—	25	ns	
78	TscR	SCK Output Rise Time	PIC18FXX31	—	25	ns	
			PIC18LFXX31	—	45	ns	
79	TscF	SCK Output Fall Time		—	25	ns	
80	TscH2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXX31	—	50	ns	
			PIC18LFXX31	—	100	ns	

28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Optional Center Pad Width	W2			4.25
Optional Center Pad Length	T2			4.25
Contact Pad Spacing	C1		5.70	
Contact Pad Spacing	C2		5.70	
Contact Pad Width (X28)	X1			0.37
Contact Pad Length (X28)	Y1			1.00
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A

D

E

L

M

NOTES: