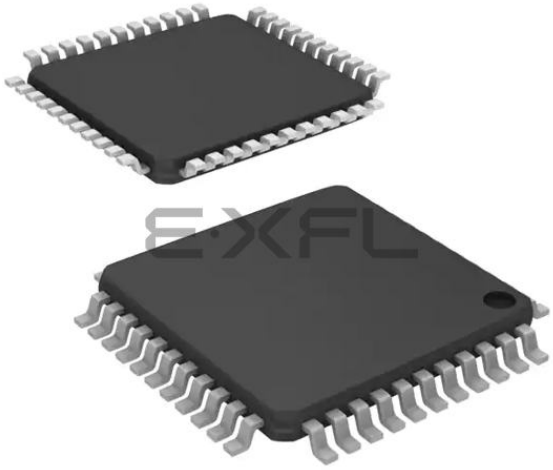


Welcome to E-XFL.COM

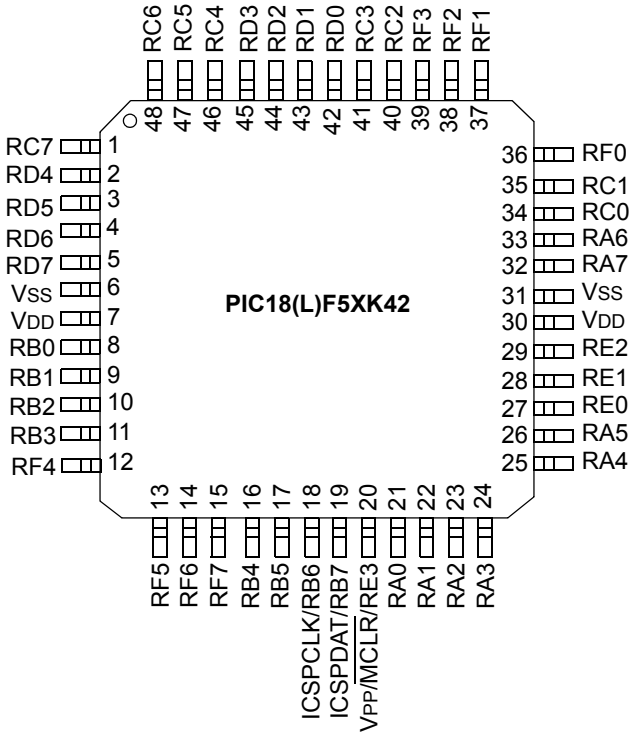
What is "Embedded - Microcontrollers"?

"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, DMA, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x12b; D/A 1x5b
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/pic18f45k42t-i-pt

48-pin TQFP (7x7x1mm)/
48-pin UQFN (6x6x0.5mm)



Note: See Table 3 for location of all peripheral functions.

TABLE 3: 48-PIN ALLOCATION TABLE FOR PIC18(L)F5XK42 (CONTINUED)

I/O	48-Pin TQFP	48-Pin UQFN	ADC	Voltage Reference	DAC	Comparators	Zero Cross Detect	I^2C	SPI	UART	DSM	Timers/SMT	CCP and PWM	CWG	CLC	NCO	Clock Reference (CLKR)	Interrupt-on-Change	Basic
VDD	7, 30	7, 30	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
VSS	6, 31	6, 31	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
OUT(2)	—	—	ADGRDA ADGRDB	—	—	C1OUT C2OUT	—	SDA1 SCL1 SDA2 SCL2	SS1 SCK1 SDO1	DTR1 RTS1 TX1 DTR2 RTS2 TX2	DSM	TMR0	CCP1 CCP2 CCP3 CCP4 PWM5OUT PWM6OUT PWM7OUT PWM8OUT	CWG1A CWG1B CWG1C CWG1D CWG2A CWG2B CWG2C CWG2D CWG3A CWG3B CWG3C CWG3D	CLC1OUT CLC2OUT CLC3OUT CLC4OUT	NCO	CLKR	—	—

Note 1: This is a PPS remappable input signal. The input function may be moved from the default location shown to one of several other PORTx pins.

2: All output signals shown in this row are PPS remappable.

3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.

4: These pins can be configured for I^2C and SMB™ 3.0/2.0 logic levels; The SCLx/SDAx signals may be assigned to any of the RB1/RB2/RC3/RC4/RD0/RD1 pins. PPS assignments to the other pins (e.g., RA5) 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.

4.3.2 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

4.3.2.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 4-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW nn instructions that returns the value 'nn' to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of two (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

EXAMPLE 4-2: COMPUTED GOTO USING AN OFFSET VALUE

```
        MOVF    OFFSET, W
        CALL    TABLE
ORG     nn00h
TABLE   ADDWF   PCL
        RETLW   nnh
        RETLW   nnh
        RETLW   nnh
        .
        .
        .
```

4.3.2.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory.

Table read and table write operations are discussed further in **Section 13.1.1 "Table Reads and Table Writes"**.

REGISTER 7-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

U-0	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f	R-f/f
—	COSC<2:0>			CDIV<3:0>			
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 **Unimplemented:** Read as '0'

bit 6-4 **COSC<2:0>:** Current Oscillator Source Select bits (read-only)(1)
Indicates the current source oscillator and PLL combination per Table 7-1.

bit 3-0 **CDIV<3:0>:** Current Divider Select bits (read-only)(1)
Indicates the current postscaler division ratio per Table 7-1.

Note 1: The POR value is the value present when user code execution begins.

REGISTER 7-3: OSCCON3: OSCILLATOR CONTROL REGISTER 3

R/W/HC-0/0	R/W-0/0	U-0	R-0/0	R-0/0	U-0	U-0	U-0
CSWHOLD	SOSCPWR	—	ORDY	NOSCR	—	—	—
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	HC = Bit is cleared by hardware

bit 7 **CSWHOLD:** Clock Switch Hold bit
1 = Clock switch will hold (with interrupt) when the oscillator selected by NOSC is ready
0 = Clock switch may proceed when the oscillator selected by NOSC is ready; NOSCR becomes '1', the switch will occur

bit 6 **SOSCPWR:** Secondary Oscillator Power Mode Select bit
1 = Secondary oscillator operating in High-Power mode
0 = Secondary oscillator operating in Low-Power mode

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

bit 4 **ORDY:** Oscillator Ready bit (read-only)
1 = OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC
0 = A clock switch is in progress

bit 3 **NOSCR:** New Oscillator is Ready bit (read-only)(1)
1 = A clock switch is in progress and the oscillator selected by NOSC indicates a "ready" condition
0 = A clock switch is not in progress, or the NOSC-selected oscillator is not yet ready

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

Note 1: If CSWHOLD = 0, the user may not see this bit set because, when the oscillator becomes ready there may be a delay of one instruction clock before this bit is set. The clock switch occurs in the next instruction cycle and this bit is cleared.

10.0 POWER-SAVING OPERATION MODES

The purpose of the Power-Down modes is to reduce power consumption. There are three Power-Down modes:

- Doze mode
- Sleep mode
- Idle mode

10.1 Doze Mode

Doze mode saves power by reducing CPU execution and program memory (PFM) access, without affecting peripheral operation.

10.1.1 DOZE OPERATION

When the Doze Enable bit is set (DOZEN = 1), the CPU executes one instruction cycle out of every N cycles as defined by the DOZE<2:0> bits of the CPUDOZE register. FOSC and FOSC/4 clock sources are unaffected in Doze mode and peripherals can continue using these sources.

10.1.2 INTERRUPTS DURING DOZE

When an interrupt occurs during Doze, the system behavior can be configured using the Recover-On-Interrupt bit (ROI) and the Doze-On-Exit bit (DOE). Refer to Table 10-2 for details about system behavior in all the cases for a transition from Main > ISR > Main. For PIC18(L)F26/27/45/46/47/55/56/57 devices, the transition from Main > ISR > Main always happens in Normal operation, regardless of the state of the DOZEN or DOE bits.

TABLE 10-1: SYSTEM BEHAVIOR FOR INTERRUPT DURING DOZE

DOZEN	ROI	Code Flow			
		Main	ISR (1)	Return to Main	
0	0	Normal operation	Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (unchanged)	If DOE = 1 when return from interrupt; Doze operation and DOZEN = 1 (in hardware)	If DOE = 0 when return from interrupt; Normal operation and DOZEN = 0 (in hardware)
0	1	Normal operation	Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (in hardware)		
1	0	Doze operation	Doze operation and DOE = DOZEN (in hardware) DOZEN = 1 (unchanged)		
1	1	Doze operation	Normal operation and DOE = DOZEN (in hardware) DOZEN = 0 (in hardware)		

Note 1: User software can change the DOE bit in ISR.

For example, if ROI = 1 and DOZE<2:0> = 001, the instruction cycle ratio is 1:4. The CPU and memory operate for one instruction cycle and stay idle for the next three instruction cycles. The Doze operation is illustrated in Figure 10-1.

10.2.1 WAKE-UP FROM SLEEP

The device can wake up from Sleep through one of the following events:

1. External Reset input on $\overline{\text{MCLR}}$ pin, if enabled
2. BOR Reset, if enabled
3. Low-Power Brown-Out Reset (LPBOR), if enabled
4. POR Reset
5. Windowed Watchdog Timer, if enabled
6. All interrupt sources except clock switch interrupt can wake up the part.

The first five events will cause a device Reset. The last one event is considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to **Section 6.13 "Power Control (PCON0/PCON1) Register"**.

When the SLEEP instruction is being executed, the next instruction (PC + 2) is prefetched. For the device to wake-up through an interrupt event, the corresponding Interrupt Enable bit must be enabled. Wake-up will occur regardless of the state of the GIE bit. If the GIE bit is disabled, the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is enabled, the device executes the instruction after the SLEEP instruction, the device will then call the Interrupt Service Routine. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

The WDT is cleared when the device wakes-up from Sleep, regardless of the source of wake-up.

Upon a wake from a Sleep event, the core will wait for a combination of three conditions before beginning execution. The conditions are:

- PFM Ready
- COSC-Selected Oscillator Ready
- BOR Ready (unless BOR is disabled)

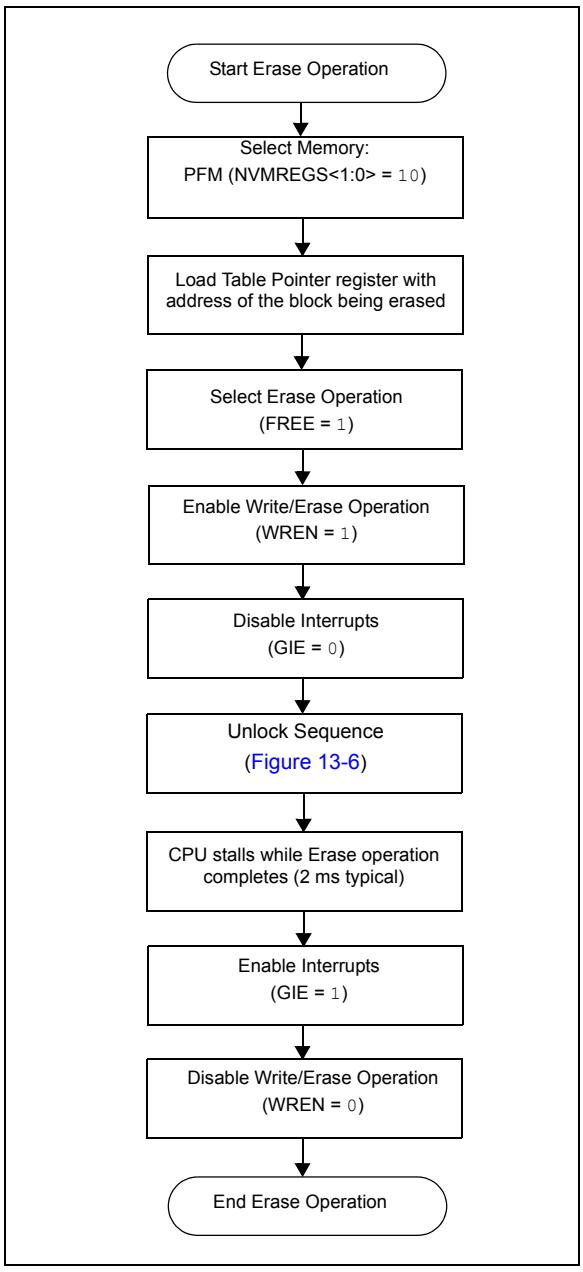
10.2.2 WAKE-UP USING INTERRUPTS

When any interrupt source, with the exception of the clock switch interrupt, has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a SLEEP instruction
  - SLEEP instruction will execute as a NOP
  - WDT and WDT prescaler will not be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will not be set
  - $\overline{\text{PD}}$ bit of the STATUS register will not be cleared
- If the interrupt occurs **during or after** the execution of a SLEEP instruction
  - SLEEP instruction will be completely executed
  - Device will immediately wake-up from Sleep
  - WDT and WDT prescaler will be cleared
  - $\overline{\text{TO}}$ bit of the STATUS register will be set
  - $\overline{\text{PD}}$ bit of the STATUS register will be cleared

Even if the flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{\text{PD}}$ bit. If the $\overline{\text{PD}}$ bit is set, the SLEEP instruction was executed as a NOP.

FIGURE 13-7: PFM ROW ERASE FLOWCHART



13.1.6 WRITING TO PROGRAM FLASH MEMORY

The programming write block size is described in Table 5-4. Word or byte programming is not supported. Table writes are used internally to load the holding registers needed to program the memory. There are only as many holding registers as there are bytes in a write block. Refer to Table 5-4 for write latch size.

Since the table latch (TABLAT) is only a single byte, the TBLWT instruction needs to be executed multiple times for each programming operation. The write protection state is ignored for this operation. All of the table write operations will essentially be short writes because only the holding registers are written. NVMIF is not affected while writing to the holding registers.

After all the holding registers have been written, the programming operation of that block of memory is started by configuring the NVMCON1 register for a program memory write and performing the long write sequence.

If the PFM address in the TBLPTR is write-protected or if TBLPTR points to an invalid location, the WR bit is cleared without any effect and the WRERR is signaled.

The long write is necessary for programming the program memory. CPU operation is suspended during a long write cycle and resumes when the operation is complete. The long write operation completes in one instruction cycle. When complete, WR is cleared in hardware and NVMIF is set and an interrupt will occur if NVMIE is also set. The latched data is reset to all '1s'. WREN is not changed.

The internal programming timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

Note:	The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all holding registers before executing a long write operation.

REGISTER 14-14: SCANLADRL: SCAN LOW ADDRESS LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LADR<7:0>(1, 2)							
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 **LADR<7:0>**: Scan Start/Current Address bits(1, 2)
Least Significant bits of the current address to be fetched from, value increments on each fetch of memory

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-15: SCANHADRU: SCAN HIGH ADDRESS UPPER BYTE 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
—	—	HADR<21:16>					
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 **HADR<21:16>:** Scan End Address bits(1, 2)
Upper bits of the address at the end of the designated scan

Note 1: Registers SCANHADRU/H/L form a 22-bit value but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

21.3 Timer1/3/5 Prescaler

Timer1/3/5 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The CKPS bits of the TxCON register control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRxH or TMRxL.

21.4 Timer1/3/5 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{SYNC}}$ of the TxCON register is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If external clock source is selected then the timer will continue to run during Sleep and can generate an interrupt on overflow, which will wake up the processor. However, special precautions in software are needed to read/write the timer (see **Section 21.4.1 "Reading and Writing Timer1/3/5 in Asynchronous Counter Mode"**).

Note:	When switching from synchronous to asynchronous operation, it is possible to skip an increment. When switching from asynchronous to synchronous operation, it is possible to produce an additional increment.

21.4.1 READING AND WRITING TIMER1/3/5 IN ASYNCHRONOUS COUNTER MODE

Reading TMRxH or TMRxL while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads. For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the TMRxH:TMRxL register pair.

21.5 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 21-2 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.

REGISTER 22-3: TxTMR: TIMERx COUNTER REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TMRx<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 **TMRx<7:0>:** Timerx Counter bits

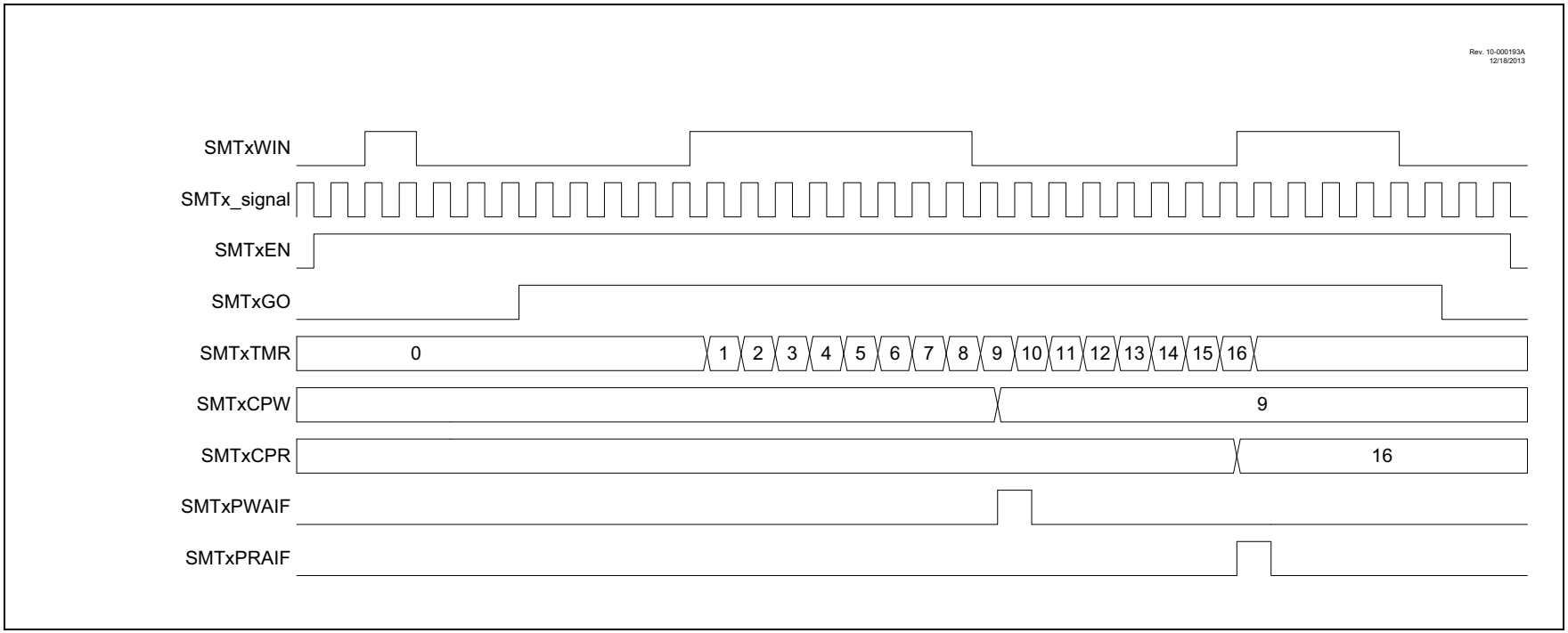
REGISTER 22-4: TxPR: TIMERx PERIOD REGISTER

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
PRx<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 **PRx<7:0>:** Timerx Period Register bits

FIGURE 25-22: WINDOWED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM



31.16 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as VDD or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value of the OSCTUNE register allows for fine resolution changes to the system clock source. See **Section 7.2.2.3 "Internal Oscillator Frequency Adjustment"** for more information.

The other method adjusts the value of the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see **Section 31.17.1 "Auto-Baud Detect"**). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change of the peripheral clock frequency.

31.17 UART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is a 16-bit timer that is dedicated to the support of the UART operation.

The UxBRGH, UxBRGL register pair determines the period of the free running baud rate timer. The multiplier of the baud rate period is determined by the BRGS bit in the UxCON0 register.

Table 31-1 contains the formulas for determining the baud rate. Example 31-1 provides a sample calculation for determining the baud rate and baud rate error.

The high baud rate range (BRGS = 1) is intended to extend the baud rate range up to a faster rate when the desired baud rate is not possible otherwise. Using the normal baud rate range (BRGS = 0) is recommended when the desired baud rate is achievable with either range.

Writing a new value to the UxBRGH, UxBRGL register pair causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

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 RXIDL bit to make sure that the receive operation is idle before changing the system clock.

EXAMPLE 31-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, BRGS = 0:

$$Desired\ Baud\ Rate = \frac{FOSC}{16([UxBRG] + 1)}$$

$$X = \frac{\frac{FOSC}{Desired\ Baud\ Rate}}{16} - 1$$

$$= \frac{\frac{16000000}{9600}}{16} - 1$$

$$= [103.17] = 103$$

$$Calculated\ Baud\ Rate = \frac{16000000}{16(103 + 1)}$$

$$= 9615$$

$$Error = \frac{Calc.\ Baud\ Rate - Desired\ Baud\ Rate}{Desired\ Baud\ Rate}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

TABLE 31-1: BAUD RATE FORMULAS

BRGS	BRG/UART Mode	Baud Rate Formula
1	High Rate	FOSC/[4 (n+1)]
0	Normal Rate	FOSC/[16(n+1)]

Legend: n = value of UxBRGH, UxBRGL register pair.

REGISTER 31-10: UxRXB: UART RECEIVE REGISTER

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
RXB<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 **RXB<7:0>:** Top of Receive Buffer

REGISTER 31-11: UxTXB: UART TRANSMIT REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TXB<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 **TXB<7:0>:** Bottom of Transmit Buffer

REGISTER 31-16: UxP3H: UART PARAMETER 3 HIGH REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	P3<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 7-6 **Unimplemented:** Read as '0'

bit 0 **P3<8>:** Most Significant Bit of Parameter 3
DMX mode:
Most Significant bit of last address of receive block
Other modes:
Not used

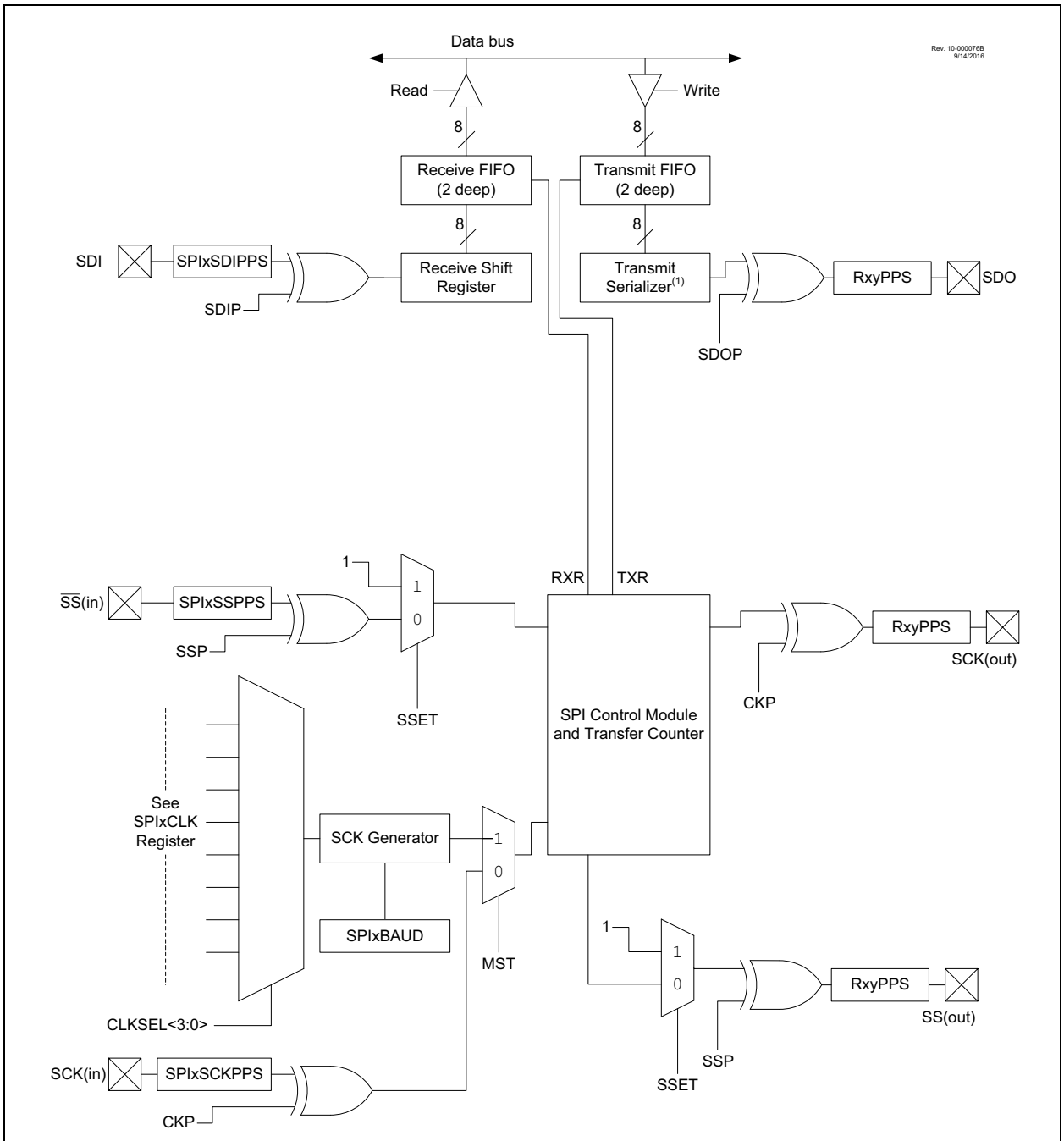
REGISTER 31-17: UxP3L: UART PARAMETER 3 LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
P3<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 **P3<7:0>:** Least Significant Bits of Parameter 3
DMX mode:
Least Significant Byte of last address of receive block
LIN Slave mode:
Number of data bytes to receive
Asynchronous Address mode:
Receiver address mask. Received address is XOR'd with UxP2L then AND'd with UxP3L
Match occurs when result is zero
Other modes:
Not used

FIGURE 32-1: SPI MODULE SIMPLIFIED BLOCK DIAGRAM



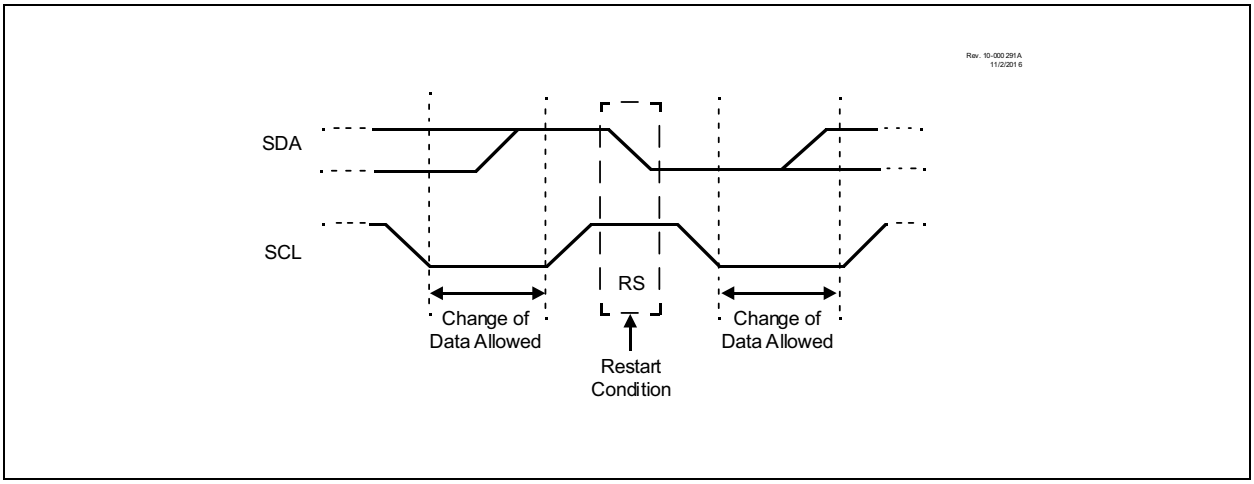
Note 1: If TXR=1 and the transmit FIFO is empty, the previous value of the receive shift register will be sent to the transmit serializer.

33.3.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave. Figure 33-4 shows the waveform for a Restart condition.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes (SMA = 1), the master can issue a Restart and the high address byte with the R/W bit set. The slave logic will then hold the clock and prepare to clock out data.

FIGURE 33-4: RESTART CONDITION



33.3.8 ACKNOWLEDGE SEQUENCE

The ninth SCL pulse for any transferred byte in I^2C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge (ACK) is an active-low signal, pulling the SDA line low indicates to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an ACK is placed in the ACKSTAT bit of the I2CxCON1 register. The ACKSTAT bit is cleared when the receiving device sends an Acknowledge and is set when the receiving device does not Acknowledge. A slave sends an Acknowledge when it has recognized its address. When in a mode that is receiving data, the ACK data being sent to the transmitter depends on the value of I2CxCNT register. ACKDT is the value sent when I2CxCNT! = 0. When I2CxCNT = 0, the ACKCNT value is used instead.

In Slave mode, if the ADRIE or WRIE bits are set, clock stretching is initiated when there is an address match or when there is an attempt to write to slave. This allows the user to set the ACK value sent back to the transmitter. The ACKDT bit of the I2CxCON1 register is set/cleared to determine the response. Slave hardware will generate an ACK response if the ADRIE or WRIE bits are clear.

Certain conditions will cause a not-ACK (NACK) to be sent automatically. If any of the RXRE, TXRE, RXO, or TXU bits is set, the hardware response is forced to NACK. All subsequent responses from the device for address matches or data will be a NACK response.

33.3.9 BUS TIME-OUT

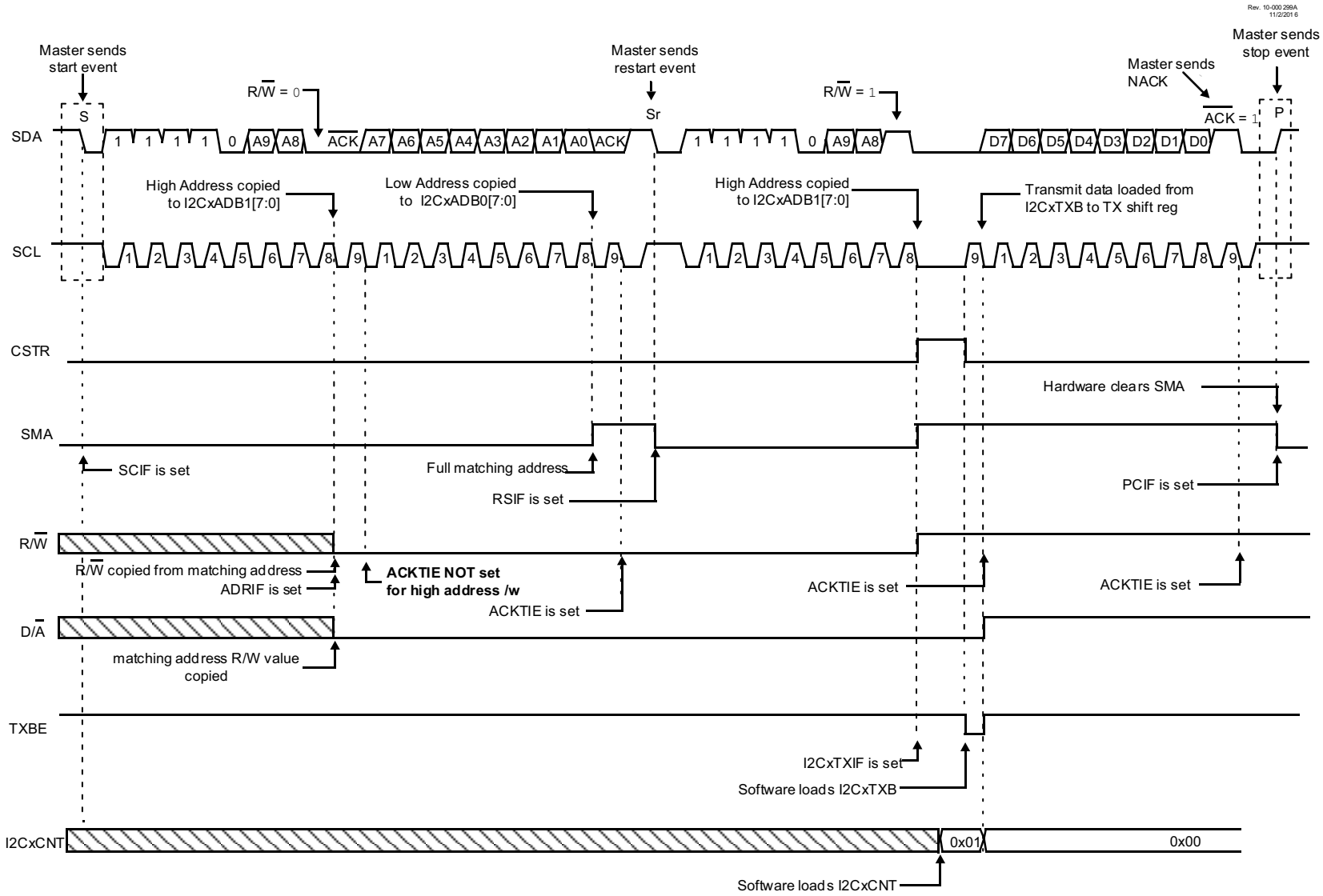
The I2CxBTO register can be used to select the timeout source for the module. The I^2C module is reset when the selected bus time out signal goes high. This feature is useful for SMBus and PMBus™ compatibility.

For example, Timer2 can be selected as the bus timeout source and configured to count when the SCL pin is low. If the timer runs over before the SCL pin transitioned high, the timer-out pulse will reset the module.

Note:	The bus time-out source should produce a rising edge.

If the module is configured as a slave and a BTO event occurs when the slave is active, i.e., the SMA bit is set, the module is immediately reset. The SMA and CSTR bits are also cleared, and the BTOIF bit is set.

FIGURE 33-12: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION



40.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

ICSP™ programming allows customers to manufacture circuit boards with unprogrammed devices. Programming can be done after the assembly process, allowing the device to be programmed with the most recent firmware or a custom firmware. Five pins are needed for ICSP™ programming:

- ICSPCLK
- ICSPDAT
- $\overline{\text{MCLR}}$/VPP
- VDD
- VSS

In Program/Verify mode the program memory, User IDs and the Configuration Words are programmed through serial communications. The ICSPDAT pin is a bidirectional I/O used for transferring the serial data and the ICSPCLK pin is the clock input. For more information on ICSP™ refer to the **"**PIC18F26/27/45/46/47/55/56/57K42 *Memory Programming Specification***"** (DS40001886).

40.1 High-Voltage Programming Entry Mode

The device is placed into High-Voltage Programming Entry mode by holding the ICSPCLK and ICSPDAT pins low then raising the voltage on $\overline{\text{MCLR}}$/VPP to VIHH.

40.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using VDD only, without high voltage. When the LVP bit of Configuration Words is set to '1', the low-voltage ICSP™ programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to '0'.

Entry into the Low-Voltage Programming Entry mode requires the following steps:

1. $\overline{\text{MCLR}}$ is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, $\overline{\text{MCLR}}$ must be held at VIL for as long as Program/Verify mode is to be maintained.

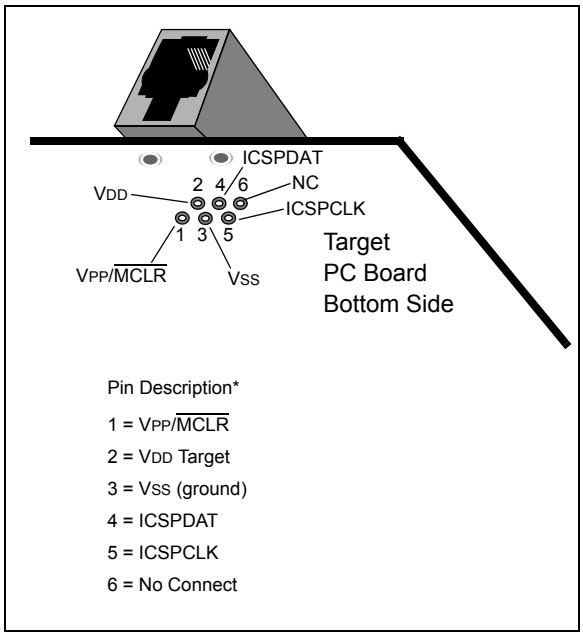
If low-voltage programming is enabled (LVP = 1), the $\overline{\text{MCLR}}$ Reset function is automatically enabled and cannot be disabled. See **Section 6.5 "MCLR"** for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

40.3 Common Programming Interfaces

Connection to a target device is typically done through an ICSP™ header. A commonly found connector on development tools is the RJ-11 in the 6P6C (6-pin, 6-connector) configuration. See Figure 40-1.

FIGURE 40-1: ICD RJ-11 STYLE CONNECTOR INTERFACE



Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 40-2.

For additional interface recommendations, refer to your specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 40-3 for more information.

TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
39DAh	OSCCON2	—	COSC			CDIV				105
39D9h	OSCCON1	—	NOSC			NDIV				104
39D8h	CPUDOZE	IDLEN	DOZEN	ROI	DOE	—	DOZE			177
39D7h - 39D2h	—	Unimplemented								
39D1h	VREGCON(1)	—	—	—	—	—	—	VREGPM	—	176
39D0h	BORCON	SBOREN	—	—	—	—	—	—	BORRDY	85
39CFh - 39C8h	—	Unimplemented								
39C7h	PMD7	—	—	—	—	—	—	DMA2MD	DMA1MD	297
39C6h	PMD6	—	—	SMT1MD	CLC4MD	CLC3MD	CLC2MD	CLC1MD	DSMMD	296
39C5h	PMD5	—	—	U2MD	U1MD	—	SPI1MD	I2C2MD	I2C1MD	295
39C4h	PMD4	CWG3MD	CWG2MD	CWG1MD	—	—	—	—	—	294
39C3h	PMD3	PWM8MD	PWM7MD	PWM6MD	PWM5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	293
39C2h	PMD2	—	DACMD	ADCMD	—	—	CMP2MD	CMP1MD	ZCDMD	292
39C1h	PMD1	NCO1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD	291
39C0h	PMD0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	290
39BFh - 39ABh	—	Unimplemented								
39AAh	PIR10	—	—	—	—	—	—	CLC4IF	CCP4IF	146
39A9h	PIR9	—	—	—	—	CLC3IF	CWG3IF	CCP3IF	TMR6IF	145
39A8h	PIR8	TMR5GIF	TMR5IF	—	—	—	—	—	—	145
39A7h	PIR7	—	—	INT2IF	CLC2IF	CWG2IF	—	CCP2IF	TMR4IF	144
39A6h	PIR6	TMR3GIF	TMR3IF	U2IF	U2EIF	U2TXIF	U2RXIF	I2C2EIF	I2C2IF	143
39A5h	PIR5	I2C2TXIF	I2C2RXIF	DMA2AIF	DMA2ORIF	DMA2DCN-TIF	DMA2SCN-TIF	C2IF	INT1IF	142
39A4h	PIR4	CLC1IF	CWG1IF	NCO1IF	—	CCP1IF	TMR2IF	TMR1GIF	TMR1IF	141
39A3h	PIR3	TMR0IF	U1IF	U1EIF	U1TXIF	U1RXIF	I2C1EIF	I2C1IF	I2C1TXIF	140
39A2h	PIR2	I2C1RXIF	SPI1IF	SPI1TXIF	SPI1RXIF	DMA1AIF	DMA1ORIF	DMA1DCN-TIF	DMA1SCNTIF	138
39A1h	PIR1	SMT1PWAIF	SMT1PRAIF	SMT1IF	C1IF	ADTIF	ADIF	ZCDIF	INT0IF	138
39A0h	PIR0	IOCIF	CRCIF	SCANIF	NVMIF	CSWIF	OSFIF	HLVDIF	SWIF	137
399Fh - 399Bh	—	Unimplemented								
399Ah	PIE10	—	—	—	—	—	—	CLC4IE	CCP4IE	156
3999h	PIE9	—	—	—	—	CLC3IE	CWG3IE	CCP3IE	TMR6IE	155
3998h	PIE8	TMR5GIE	TMR5IE	—	—	—	—	—	—	155
3997h	PIE7	—	—	INT2IE	CLC2IE	CWG2IE	—	CCP2IE	TMR4IE	154
3996h	PIE6	TMR3GIE	TMR3IE	U2IE	U2EIE	U2TXIE	U2RXIE	I2C2EIE	I2C2IE	153
3995h	PIE5	I2C2TXIE	I2C2RXIE	DMA2AIE	DMA2ORIE	DMA2DCN-TIE	DMA2SCN-TIE	C2IE	INT1IE	152
3994h	PIE4	CLC1IE	CWG1IE	NCO1IE	—	CCP1IE	TMR2IE	TMR1GIE	TMR1IE	151
3993h	PIE3	TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE	150
3992h	PIE2	I2C1RXIE	SPI1IE	SPI1TXIE	SPI1RXIE	DMA1AIE	DMA1ORIE	DMA1DCN-TIE	DMA1SCNTIE	149
3991h	PIE1	SMT1PWAIE	SMT1PRAIE	SMT1IE	C1IE	ADTIE	ADIE	ZCDIE	INT0IE	148
3990h	PIE0	IOCIE	CRCIE	SCANIE	NVMIE	CSWIE	OSFIE	HLVDIE	SWIE	147
398Fh - 398Bh	—	Unimplemented								
398Ah	IPR10	—	—	—	—	—	—	CLC4IP	CCP4IP	165

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
3: Unimplemented on PIC18(L)F26/27/45/46/47K42 devices.
4: Unimplemented in PIC18(L)F45/55K42.

THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: http://www.microchip.com/support