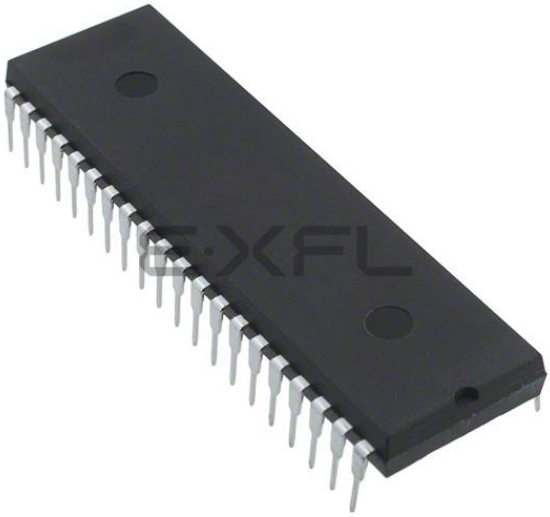


Welcome to E-XFL.COM

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	ECANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 11x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf46k80-i-p

TABLE 1-6: PIC18F6XK80 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Num	Pin Type	Buffer Type	Description
$\overline{\text{MCLR}}$/RE3	28			
$\overline{\text{MCLR}}$		I	ST	Master Clear (input) or programming voltage (input). This pin is an active-low Reset to the device.
RE3		I	ST	General purpose, input only pin.
OSC1/CLKIN/RA7	46			
OSC1		I	ST	Oscillator crystal input.
CLKIN		I	CMOS	External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7		I/O	ST/ CMOS	General purpose I/O pin.
OSC2/CLKOUT/RA6	47			
OSC2		O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKOUT		O	—	In certain oscillator modes, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	ST/ CMOS	General purpose I/O pin.

Legend: I^2C™ = I^2C/SMBus input buffer
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output

REGISTER 3-4: REFOCON: REFERENCE OSCILLATOR CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ROON	—	ROSSLP	ROSEL(1)	RODIV3	RODIV2	RODIV1	RODIV0
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 **ROON:** Reference Oscillator Output Enable bit
1 = Reference oscillator output is available on REFO pin
0 = Reference oscillator output is disabled

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

bit 5 **ROSSLP:** Reference Oscillator Output Stop in Sleep bit
1 = Reference oscillator continues to run in Sleep
0 = Reference oscillator is disabled in Sleep

bit 4 **ROSEL:** Reference Oscillator Source Select bit(1)
1 = Primary oscillator (EC or HS) is used as the base clock
0 = System clock is used as the base clock; base clock reflects any clock switching of the device

bit 3-0 **RODIV<3:0>:** Reference Oscillator Divisor Select bits
1111 = Base clock value divided by 32,768
1110 = Base clock value divided by 16,384
1101 = Base clock value divided by 8,192
1100 = Base clock value divided by 4,096
1011 = Base clock value divided by 2,048
1010 = Base clock value divided by 1,024
1001 = Base clock value divided by 512
1000 = Base clock value divided by 256
0111 = Base clock value divided by 128
0110 = Base clock value divided by 64
0101 = Base clock value divided by 32
0100 = Base clock value divided by 16
0011 = Base clock value divided by 8
0010 = Base clock value divided by 4
0001 = Base clock value divided by 2
0000 = Base clock value

Note 1: For ROSEL (REFOCON<4>), the primary oscillator is available only when configured as the default via the FOSCx settings. This is regardless of whether the device is in Sleep mode.

5.6.2 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-up Timer (OST) provides a 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over (Parameter 33). This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP, HS and HSPLL modes and only on Power-on Reset or on exit from most power-managed modes.

5.6.3 PLL LOCK TIME-OUT

With the PLL enabled in its PLL mode, the time-out sequence following a Power-on Reset is slightly different from other oscillator modes. A separate timer is used to provide a fixed time-out that is sufficient for the PLL to lock to the main oscillator frequency. This PLL lock time-out (T_{PLL}) is typically 2 ms and follows the oscillator start-up time-out.

5.6.4 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows:

1. After the POR pulse has cleared, PWRT time-out is invoked (if enabled).
2. Then, the OST is activated.

The total time-out will vary based on oscillator configuration and the status of the PWRT. Figure 5-3, Figure 5-4, Figure 5-5, Figure 5-6 and Figure 5-7 all depict time-out sequences on power-up, with the Power-up Timer enabled and the device operating in HS Oscillator mode. Figures 5-3 through 5-6 also apply to devices operating in XT or LP modes. For devices in RC mode and with the PWRT disabled, on the other hand, there will be no time-out at all.

Since the time-outs occur from the POR pulse, if $\overline{MCLR}$ is kept low long enough, all time-outs will expire. Bringing $\overline{MCLR}$ high will begin execution immediately (Figure 5-5). This is useful for testing purposes or to synchronize more than one PIC18FXXXX device operating in parallel.

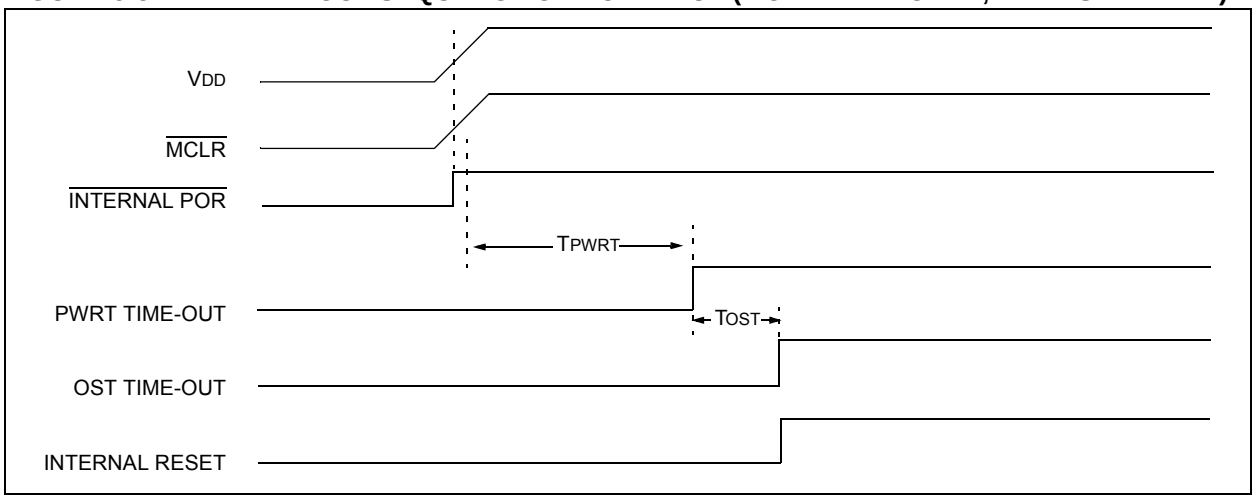
TABLE 5-2: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up and Brown-out		Exit from Power-Managed Mode
	$\overline{PWRTEN}$ = 0	$\overline{PWRTEN}$ = 1	
HSPLL	66 ms(1) + 1024 TOSC + 2 ms(2)	1024 TOSC + 2 ms(2)	1024 TOSC + 2 ms(2)
HS, XT, LP	66 ms(1) + 1024 TOSC	1024 TOSC	1024 TOSC
EC, ECIO	66 ms(1)	—	—
RC, RCIO	66 ms(1)	—	—
INTIO1, INTIO2	66 ms(1)	—	—

Note 1: 66 ms (65.5 ms) is the nominal Power-up Timer (PWRT) delay.

2: 2 ms is the nominal time required for the PLL to lock.

FIGURE 5-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ TIED TO V_{DD}, V_{DD} RISE < T_{PWRT})



6.1.3.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 6-1) contains the Stack Pointer value, the STKFUL (Stack Full) status bit and the STKUNF (Stack Underflow) status bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off of the stack. On Reset, the Stack Pointer value will be zero.

The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit is cleared by software or by a POR.

What happens when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (For a description of the device Configuration bits, see **Section 28.1 "Configuration Bits"**.) If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the Stack Pointer will increment to 31. Any additional pushes will not overwrite the 31st push and the STKPTR will remain at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and set the STKUNF bit, while the Stack Pointer remains at zero. The STKUNF bit will remain set until cleared by software or until a POR occurs.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

6.1.3.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off of the stack, without disturbing normal program execution, is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

REGISTER 6-1: STKPTR: STACK POINTER REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL(1)	STKUNF(1)	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

Legend:	C = Clearable bit		
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 **STKFUL:** Stack Full Flag bit(1)
1 = Stack has become full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF:** Stack Underflow Flag bit(1)
1 = Stack underflow has occurred
0 = Stack underflow did not occur

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

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

Note 1: Bit 7 and bit 6 are cleared by user software or by a POR.

6.1.3.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit (CONFIG4L<0>). When STVREN is set, a full or underflow condition will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

6.1.4 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers to provide a “fast return” option for interrupts. This stack 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 the working 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 in 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 6-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

EXAMPLE 6-1: 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
```

6.1.5 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

6.1.5.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the Program Counter. An example is shown in Example 6-2.

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 6-2: COMPUTED GOTO USING AN OFFSET VALUE

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

6.1.5.2 Table Reads

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 while programming. The Table Pointer (TBLPTR) specifies the byte address and the Table Latch (TABLAT) contains the data that is read from the program memory. Data is transferred from program memory one byte at a time.

The table read operation is discussed further in **Section 7.1 “Table Reads and Table Writes”**.

REGISTER 10-15: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

R/W-1	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	—	—	—	BCLIP	HLVDIP	TMR3IP	TMR3GIP
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 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6-4 **Unimplemented:** Read as ‘0’

bit 3 **BCLIP:** Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **HLVDIP:** High/Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 0 **TMR3GIP:** TMR3 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

11.7 PORTF, LATF and TRISF Registers

PORTF is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISF and LATF. All pins on PORTF are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Note: PORTF is only available on 64-pin devices.

Each of the PORTF pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is done by clearing bit, RFPU (PADCFG1<5>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

Note: On device Resets, pins, RF<7:1>, are configured as analog inputs and are read as '0'.

EXAMPLE 11-6: INITIALIZING PORTF

```
CLRF    PORTF   ; Initialize PORTF by
                ; clearing output
                ; data latches
CLRF    LATF    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CEh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISF   ; Set RF3:RF1 as inputs
                ; RF5:RF4 as outputs
                ; RF7:RF6 as inputs
```

TABLE 11-11: PORTF FUNCTIONS

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RF0/MDMIN	RF0	0	O	DIG	LATF<0> data output.
		1	I	ST	PORTF<0> data input.
	MDMIN	1	I	ST	Modulator source input.
RF1	RF1	0	O	DIG	LATF<1> data output.
		1	I	ST	PORTF<1> data input.
RF2/MDCIN1	RF2	0	O	DIG	LATF<2> data output.
		1	I	ST	PORTF<2> data input.
	MDCIN1	1	I	ST	Modulator Carrier Input 1.
RF3	RF3	0	O	DIG	LATF<3> data output.
		1	I	ST	PORTF<3> data input.
RF4/MDCIN2	RF4	0	O	DIG	LATF<4> data output.
		1	I	ST	PORTF<4> data input.
	MDCIN2	1	I	ST	Modulator Carrier Input 2.
RF5	RF5	0	O	DIG	LATF<5> data output.
		1	I	ST	PORTF<5> data input.
RF6/MDOUT	RF6	0	O	DIG	LATF<6> data output.
		1	I	ST	PORTF<6> data input.
	MDOUT	0	O	DIG	Modulator output.
RF7	RF7	0	O	DIG	LATF<7> data output.
		1	I	ST	PORTF<7> data input.

Legend: O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input; x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

TABLE 11-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0
PADCFG1	RDPU	REPU	RFPU(1)	RGPU(1)	—	—	—	CTMUDS

Legend: — = unimplemented, read as '0'. Shaded cells are not used by PORTF.

Note 1: These bits are unimplemented on 28-pin devices, read as '0'.

17.2 Timer4 Interrupt

The Timer4 module has an eight-bit Period register, PR4, that is both readable and writable. Timer4 increment from 00h until it matches PR4 and then resets to 00h on the next increment cycle. The PR4 register is initialized to FFh upon Reset.

17.3 Output of TMR4

The outputs of TMR4 (before the postscaler) are used only as a PWM time base for the ECCP modules. They are not used as baud rate clocks for the MSSP module as is the Timer2 output.

FIGURE 17-1: TIMER4 BLOCK DIAGRAM

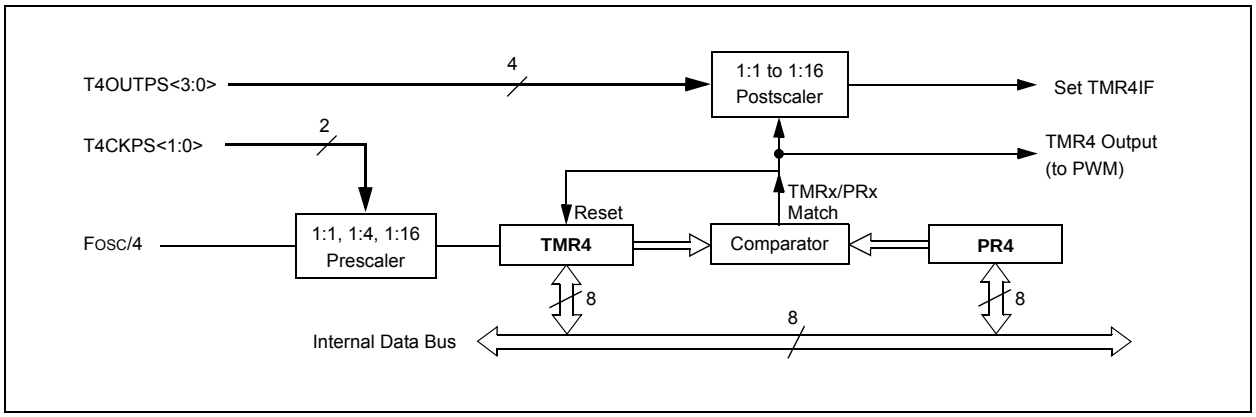


TABLE 17-1: REGISTERS ASSOCIATED WITH TIMER4 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
TMR4	Timer4 Register							
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
PR4	Timer4 Period Register							
PMD1	PSPMD	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer4 module.

EXAMPLE 18-1: SETUP FOR CTMU CALIBRATION ROUTINES

```
#include "p18cxxx.h"
/*****************************************************************************/
/*Setup CTMU *****************************************************************/
/*****************************************************************************/
void setup(void)

{ //CTMUCON - CTMU Control register

    CTMUCONH = 0x00;              //make sure CTMU is disabled
    CTMUCONL = 0x90;
    //CTMU continues to run when emulator is stopped,CTMU continues
    //to run in idle mode,Time Generation mode disabled, Edges are blocked
    //No edge sequence order, Analog current source not grounded, trigger
    //output disabled, Edge2 polarity = positive level, Edge2 source =
    //source 0, Edge1 polarity = positive level, Edge1 source = source 0,
    // Set Edge status bits to zero


    //CTMUICON - CTMU Current Control Register
    CTMUICON = 0x01;              //0.55uA, Nominal - No Adjustment

/*****************************************************************************/
//Setup AD converter;
/*****************************************************************************/

    TRISA=0x04;                   //set channel 2 as an input

    // Configured AN2 as an analog channel
    // ANCON1
    ANCON1 = 0x04;

    // ADCON1
    ADCON2bits.ADFM=1;            // Result format 1= Right justified
    ADCON2bits.ACQT=1;            // Acquisition time 7 = 20TAD 2 = 4TAD 1=2TAD
    ADCON2bits.ADCS=2;            // Clock conversion bits 6= FOSC/64 2=FOSC/32

    // ADCON1
    ADCON1bits.VCFG0 =0;          // Vref+ = AVdd
    ADCON1bits.VCFG1 =0;          // Vref+ = AVdd
    ADCON1bits.VNCFG = 0;         // Vref- = AVss
    ADCON1bits.CHS=2;             // Select ADC channel

    ADCON0bits.ADON=1;            // Turn on ADC

}
```

19.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE bit (PIE4<x>) clear to avoid false interrupts and should clear the flag bit, CCPxIF, following any such change in operating mode.

19.2.4 CCP PRESCALER

There are four prescaler settings in Capture mode. They are specified as part of the operating mode selected by the mode select bits (CCPxM<3:0>). Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Doing that also will not clear the prescaler counter – meaning the first capture may be from a non-zero prescaler.

Example 19-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

EXAMPLE 19-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF   CCPxCON      ; Turn CCP module off
MOVLW  NEW_CAPT_PS  ; Load WREG with the
                    ; new prescaler mode
                    ; value and CCP ON
MOVWF  CCPxCON      ; Load CCPxCON with
                    ; this value
```

19.2.5 CAN MESSAGE TIME-STAMP (CCP2 ONLY)

For CCP2, only the CAN capture event occurs when a message is received in any of the receive buffers. When configured, the CAN module provides the trigger to the CCP2 module to cause a capture event. This feature is provided to “time-stamp” the received CAN messages.

This feature is enabled by setting the CANCAP bit of the CAN I/O Control register (CIOCON<4>). The message receive signal from the CAN module then takes the place of the events on RC2/CCP2.

If this feature is selected, then four different capture options for CCP2M<3:0> are available:

- 0100 – Every time a CAN message is received
- 0101 – Every time a CAN message is received
- 0110 – Every 4th time a CAN message is received
- 0111 – Capture mode, every 16th time a CAN message is received

20.4.2.1 Direction Change in Full-Bridge Mode

In the Full-Bridge mode, the P1M1 bit in the CCP1CON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will change to the new direction on the next PWM cycle.

A direction change is initiated in software by changing the P1M1 bit of the CCP1CON register. The following sequence occurs prior to the end of the current PWM period:

- The modulated outputs (P1B and P1D) are placed in their inactive state.
- The associated unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction.
- PWM modulation resumes at the beginning of the next period.

For an illustration of this sequence, see Figure 20-10.

The Full-Bridge mode does not provide a dead-band delay. As one output is modulated at a time, a dead-band delay is generally not required. There is a situation where a dead-band delay is required. This situation occurs when both of the following conditions are true:

- The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
- The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

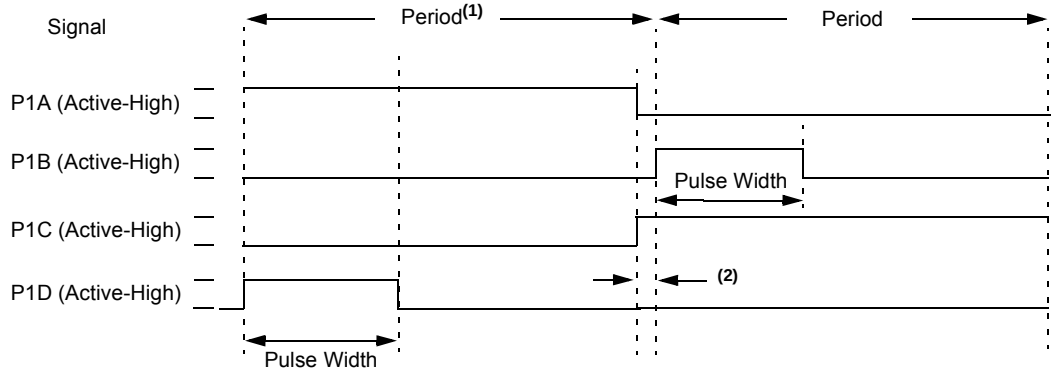
Figure 20-11 shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time, t1, the P1A and P1D outputs become inactive, while the P1C output becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices, QC and QD (see Figure 20-8), for the duration of ‘t’. The same phenomenon will occur to power devices, QA and QB, for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

- Reduce PWM duty cycle for one PWM period before changing directions.
- Use switch drivers that can drive the switches off faster than they can drive them on.

Other options to prevent shoot-through current may exist.

FIGURE 20-10: EXAMPLE OF PWM DIRECTION CHANGE



Note 1: The direction bit, P1M1 of the CCP1CON register, is written any time during the PWM cycle.

2: When changing directions, the P1A and P1C signals switch before the end of the current PWM cycle. The modulated P1B and P1D signals are inactive at this time. The length of this time is: (1/FOSC) • TMR2 Prescale Value.

22.4 EUSARTx Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTAx<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTAx<4>). In addition, enable bit, SPEN (RCSTAx<7>), is set in order to configure the TXx and RXx pins to CKx (clock) and DTx (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CKx line. Clock polarity is selected with the TXCKP bit (BAUDCONx<4>). Setting TXCKP sets the Idle state on CKx as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

22.4.1 EUSARTx SYNCHRONOUS MASTER TRANSMISSION

The EUSARTx transmitter block diagram is shown in Figure 22-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The shift register obtains its data from the Read/Write Transmit Buffer register, TXREGx. The TXREGx register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREGx (if available).

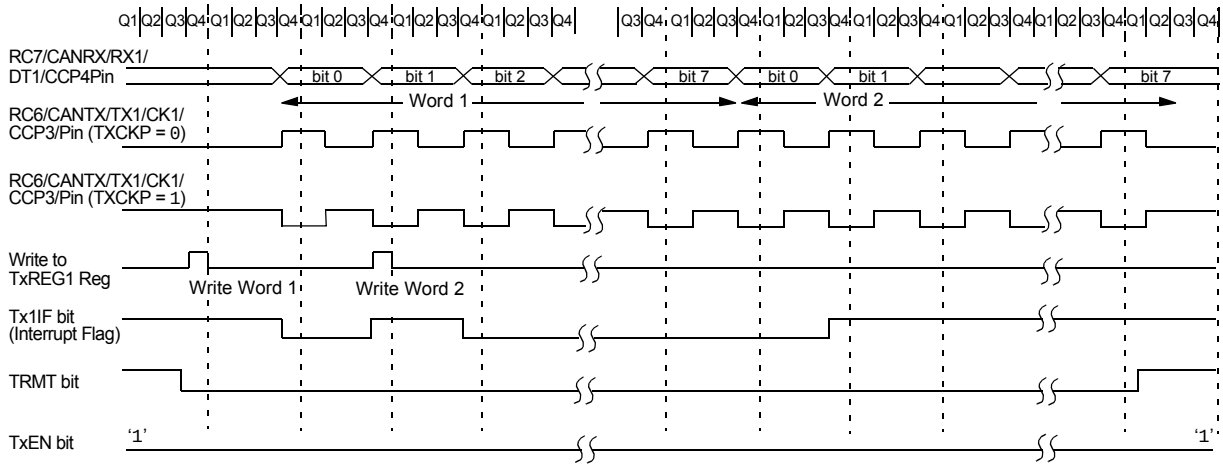
Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxIF is set regardless of the state of enable bit, TXxIE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREGx register.

While flag bit, TXxIF, indicates the status of the TXREGx register, another bit, TRMT (TXSTAx<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user must poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TXxIE.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits (INTCON<7:6>) are set.

FIGURE 22-11: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode, SPBRGx = 0; continuous transmission of two 8-bit words. This example is equally applicable to EUSART2 (RB6/PGC/TX2/CK2/KBI2 and RB7/PGD/T3G/RX2/DT2/KBI3).

REGISTER 23-9: ANCON1: A/D PORT CONFIGURATION REGISTER 1

U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ANSEL14(1)	ANSEL13(1)	ANSEL12(1)	ANSEL11(1)	ANSEL10	ANSEL9	ANSEL8
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 **ANSEL14:** RD3/C2INB Pin Analog Enable bit(1)
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'
0 = Pin is configured as a digital port

bit 5 **ANSEL13:** RD2/C2INA Pin Analog Enable bit(1)
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'
0 = Pin is configured as a digital port

bit 4 **ANSEL12:** RD1/C1INB Pin Analog Enable bit(1)
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'
0 = Pin is configured as a digital port

bit 3 **ANSEL11:** RD0/C1INA Pin Analog Enable bit(1)
1 = Pin is configured as an analog channel: digital input disabled and any inputs read as '0'
0 = Pin is configured as a digital port

bit 2-0 **ANSEL11<10:8>:** Analog Port Configuration bits (AN10 through AN8)
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'
0 = Pin configured as a digital port

Note 1: AN14 through AN11 and AN7 to AN5 are implemented only on 40/44-pin and 64-pin devices. For 28-pin devices, the corresponding ANSELx bits are still implemented for these channels, but have no effect.

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (AV_{DD} and AV_{SS}) or the voltage level on the RA3/V_{REF}+/AN3 and RA2/V_{REF}-/AN2 pins. V_{REF}+ has two additional internal voltage reference selections: 2.0V and 4.1V.

The A/D Converter can uniquely operate while the device is in Sleep mode. To operate in **Sleep**, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

Each port pin associated with the A/D Converter can be configured as an analog input or a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/$\overline{\text{DONE}}$ bit (ADCON0<1>) is cleared and the A/D Interrupt Flag bit, ADIF (PIR1<6>), is set.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted. The value in the ADRESH:ADRESL register pair is not modified for a Power-on Reset. These registers will contain unknown data after a Power-on Reset.

The block diagram of the A/D module is shown in Figure 23-4.

REGISTER 27-34: BnDLC: TX/RX BUFFER 'n' DATA LENGTH CODE REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0](1)

U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0
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 **RXRTR:** Receiver Remote Transmission Request bit
1 = This is a remote transmission request
0 = This is not a remote transmission request

bit 5 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as '0'.

bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as '0'.

bit 3-0 **DLC<3:0>:** Data Length Code bits
1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = Reserved
1011 = Reserved
1010 = Reserved
1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 byte
0000 = Data length = 0 bytes

Note 1: These registers are available in Mode 1 and 2 only.

BTG **Bit Toggle f**

Syntax: BTG f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff

Description: Bit ‘b’ in data memory location ‘f’ is inverted.

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

BOV **Branch if Overflow**

Syntax: BOV n

Operands: $-128 \le n \le 127$

Operation: if Overflow bit is ‘1’,
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn

Description: If the Overflow bit is ‘1’, then the program will branch.

The 2’s complement number, ‘2n’, is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal ‘n’	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal ‘n’	Process Data	No operation

Example:
```
HERE    BOV   Jump
```
Before Instruction
PC = address (HERE)
After Instruction
If Overflow = 1;
PC = address (Jump)
If Overflow = 0;
PC = address (HERE + 2)

CLRF **Clear f**

Syntax: CLRF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: 000h → f,
1 → Z

Status Affected: Z

Encoding:

0110	101a	ffff	ffff

Description: Clears the contents of the specified register.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: CLRF FLAG_REG,1

Before Instruction
FLAG_REG = 5Ah
After Instruction
FLAG_REG = 00h

CLRWDT **Clear Watchdog Timer**

Syntax: CLRWDT

Operands: None

Operation: 000h → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0100

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: CLRWDT

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 00h
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

IORLW **Inclusive OR Literal with W**

Syntax: IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → W

Status Affected: N, Z

Encoding:

0000	1001	kkkk	kkkk

Description: The contents of W are ORed with the eight-bit literal 'k'. 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: IORLW 35h

Before Instruction
W = 9Ah
After Instruction
W = BFh

IORWF **Inclusive OR W with f**

Syntax: IORWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .OR. (f) → dest

Status Affected: N, Z

Encoding:

0001	00da	ffff	ffff

Description: Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: IORWF RESULT, 0, 1

Before Instruction
RESULT = 13h
W = 91h
After Instruction
RESULT = 13h
W = 93h

TABLE 31-22: MSSP I²C™ BUS DATA REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	2(TOSC)(BRG + 1)	—	—	
			400 kHz mode	2(TOSC)(BRG + 1)	—	—	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	—	
101	TLOW	Clock Low Time	100 kHz mode	2(TOSC)(BRG + 1)	—	—	
			400 kHz mode	2(TOSC)(BRG + 1)	—	—	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	—	
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode(1)	—	300	ns	
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode(1)	—	100	ns	
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	—	Only relevant for Repeated Start condition
			400 kHz mode	2(TOSC)(BRG + 1)	—	—	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	—	
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	—	After this period, the first clock pulse is generated
			400 kHz mode	2(TOSC)(BRG + 1)	—	—	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	—	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	—	
			400 kHz mode	0	0.9	μs	
			1 MHz mode(1)	—	μs	ns	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
			1 MHz mode(1)	—	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	—	
			400 kHz mode	2(TOSC)(BRG + 1)	—	—	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	—	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	
			400 kHz mode	—	1000	ns	
			1 MHz mode(1)	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
			1 MHz mode(1)	—	—	μs	
D102	CB	Bus Capacitive Loading		—	400	pF	

Note 1: Maximum pin capacitance = 10 pF for all I²C™ pins.

2: A Fast mode I²C bus device can be used in a Standard mode I²C bus system, but Parameter #107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line, Parameter #102 + Parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode), before the SCL line is released.