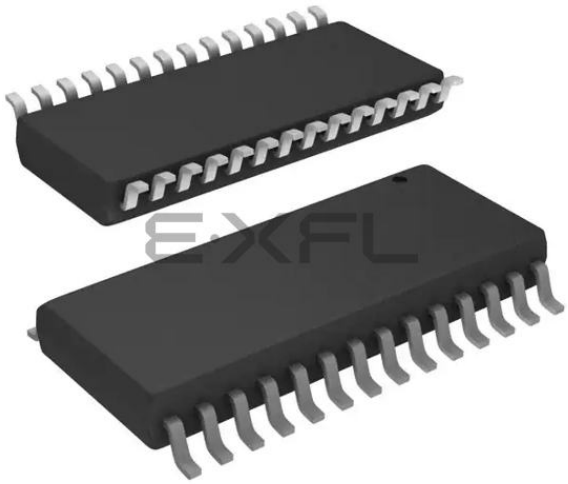


Welcome to E-XFL.COM

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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	25
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 24x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f24q10t-i-so

Bits 0, 1, 2, 3 – WRTn User NVM Self-Write Protection bits

Value	Description
1	Corresponding Memory Block NOT write-protected
0	Corresponding Memory Block write-protected

Related Links

10.1 Program Memory Organization

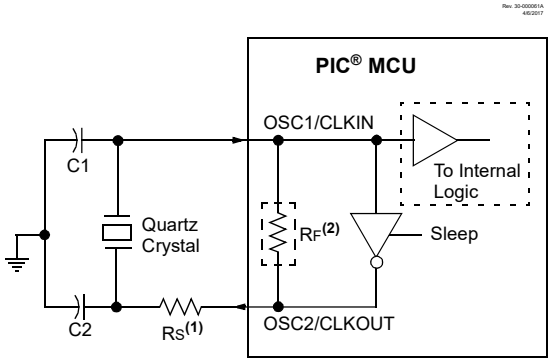
11.3.4 Operation During Code-Protect and Write-Protect

XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification (above 500 kHz - 8 MHz).

HS Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require a high drive setting (above 8 MHz).

Figure 4-3 and Figure 4-4 show typical circuits for quartz crystal and ceramic resonators, respectively.

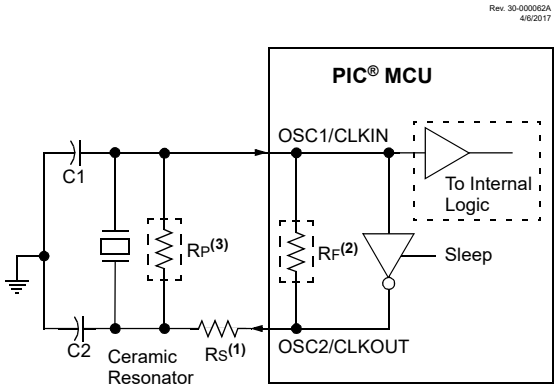
Figure 4-3. Quartz Crystal Operation (LP, XT or HS Mode)



Note:

1. A series resistor (R_S) may be required for quartz crystals with low drive level.
2. The value of R_F varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

Figure 4-4. Ceramic Resonator Operation (XT or HS Mode)



Note:

1. A series resistor (R_S) may be required for ceramic resonators with low drive level.
2. The value of R_F varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).
3. An additional parallel feedback resistor (R_P) may be required for proper ceramic resonator operation.

4.2.1.3 Oscillator Start-up Timer (OST)

If the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) counts 1024 oscillations from OSC1. This occurs following a Power-on Reset (POR), or a wake-up from Sleep. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the oscillator module.

4.6.2 OSCCON2

Name: OSCCON2
Address: 0xED4

Oscillator Control Register 2

Bit	7	6	5	4	3	2	1	0
		COSC[2:0]			CDIV[3:0]			
Access		R	R	R	R	R	R	R
Reset		q	q	q	q	q	q	q

Bits 6:4 – COSC[2:0] Current Oscillator Source Select bits (read-only)(1,2)
Indicates the current source oscillator and PLL combination as shown in the following table.

Table 4-4. COSC Bit Settings

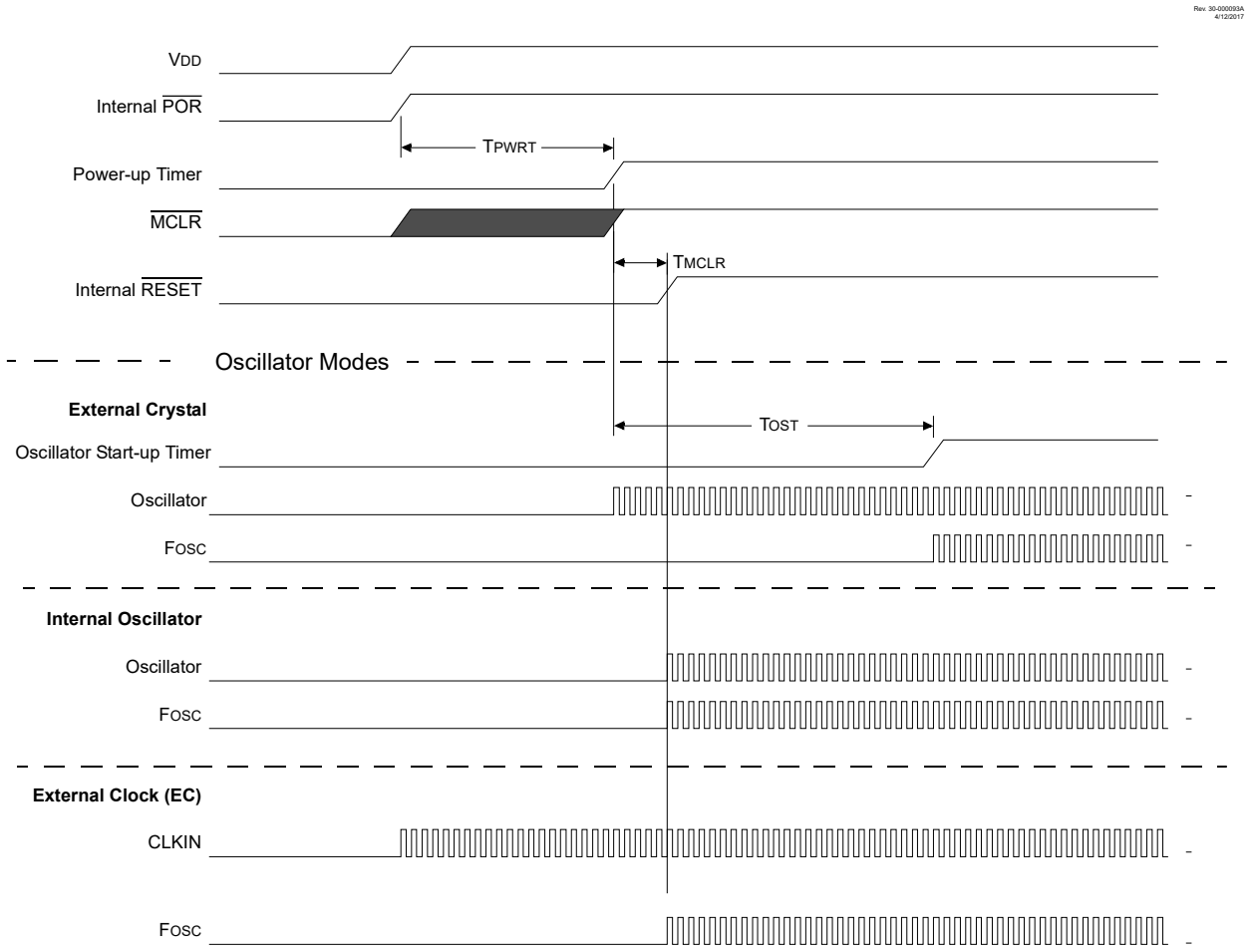
COSC/NOSC	Clock Source
111	EXTOSC(3)
110	HFINTOSC(4)
101	LFINTOSC
100	SOSC
011	Reserved
010	EXTOSC + 4x PLL(5)
001	Reserved
000	Reserved

Bits 3:0 – CDIV[3:0] Current Divider Select bits (read-only)(1,2)
Indicates the current postscaler division ratio as shown in the follwing table.

Table 4-5. CDIV Bit Settings

CDIV/NDIV	Clock Divider
1111-1010	Reserved
1001	512
1000	256
0111	128
0110	64
0101	32
0100	16
0011	8
0010	4

Figure 8-4. Reset Start-up Sequence



Related Links

4. Oscillator Module (with Fail-Safe Clock Monitor)

8.11 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS and PCON0 registers are updated to indicate the cause of the Reset. The following table shows the Reset conditions of these registers.

Table 8-3. Reset Condition for Special Registers

Condition	Program Counter	STATUS Register(2,3)	PCON0 Register	PCON1 Register
Power-on Reset	0	-110 0000	0011 110x	---- -1-1
Brown-out Reset	0	-110 0000	0011 11u0	---- -u-u
$\overline{MCLR}$ Reset during normal operation	0	-uuu uuuu	uuuu 0uuu	uuuu-u-u
$\overline{MCLR}$ Reset during Sleep	0	-10u uuuu	uuuu 0uuu	uuuu-u-u

9.8.4 WDTPSL

Name: WDTPSL
Address: 0xECF

WWDT Prescale Select Low Register (Read-Only)

Bit	7	6	5	4	3	2	1	0
	PSCNTL[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PSCNTL[7:0] Prescale Select Low Byte bits(1)

Note:

1. The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should be read during normal operation.

10.1.2.3 Stack Overflow and Underflow Resets

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

10.1.2.4 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 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.

10.1.2.5 Fast Register Stack

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

Important: The $\overline{TO}$ and $\overline{PD}$ bits of the STATUS register are not copied over in this operation.

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

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

The following example shows a source code example that uses the fast register stack during a subroutine call and return.

Example 10-1. Fast Register Stack Code Example

```
     CALL SUB1, FAST  ;STATUS, WREG, BSR SAVED IN FAST REGISTER STACK
          •
          •
SUB1:
          •
```

specific case, assume that FSR0H:FSR0L contains the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a NOP.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to either the INDF2 or POSTDEC2 register will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses indirect addressing.

Similarly, operations by indirect addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

10.5 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect addressing with FSR0 and FSR1 also remain unchanged.

10.5.1 Indexed Addressing with Literal Offset

Enabling the PIC18 extended instruction set changes the behavior of indirect addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of indexed addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced (‘a’ = 0) and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in direct addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

10.5.2 Instructions Affected by Indexed Literal Offset Mode

Any of the core PIC18 instructions that can use direct addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘1’), or include a file address of 60h or above. Instructions meeting these criteria

13.12.1 CRCCON0

Name: CRCCON0
Address: 0xF77
Reset: 0

CRC Control Register 0

Bit	7	6	5	4	3	2	1	0
	EN	GO	BUSY	ACCM			SHIFTM	FULL
Access	R/W	R/W	RO	R/W			R/W	RO
Reset	0	0	0	0			0	0

Bit 7 – EN CRC Enable bit

Value	Description
1	CRC module is released from Reset
0	CRC is disabled and consumes no operating current

Bit 6 – GO CRC Start bit

Value	Description
1	Start CRC serial shifter
0	CRC serial shifter turned off

Bit 5 – BUSY CRC Busy bit

Value	Description
1	Shifting in progress or pending
0	All valid bits in shifter have been shifted into accumulator and EMPTY = 1

Bit 4 – ACCM Accumulator Mode bit

Value	Description
1	Data is augmented with zeros
0	Data is not augmented with zeros

Bit 1 – SHIFTM Shift Mode bit

Value	Description
1	Shift right (LSb)
0	Shift left (MSb)

Bit 0 – FULL Data Path Full Indicator bit

Value	Description
1	CRCDATH/L registers are full
0	CRCDATH/L registers have shifted their data into the shifter

```
        MOVFF   STATUS, STATUS_TEMP    ; STATUS_TEMP located anywhere
        MOVFF   BSR, BSR_TEMP          ; BSR_TEMP located anywhere
;
; USER ISR CODE
;
        MOVFF   BSR_TEMP, BSR          ; Restore BSR
        MOVF    W_TEMP, W              ; Restore WREG
        MOVFF   STATUS_TEMP, STATUS    ; Restore STATUS
```

Related Links

10.1.2.5 Fast Register Stack

14.13.4 PIR2

Name: PIR2
Address: 0xEC7

Peripheral Interrupt Request (Flag) Register 2

Bit	7	6	5	4	3	2	1	0
	HLVDIF	ZCDIF					C2IF	C1IF
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

Bit 7 – HLVDIF HLVD Interrupt Flag bit

Value	Description
1	HLVD interrupt event has occurred
0	HLVD interrupt event has not occurred or has not been set up

Bit 6 – ZCDIF Zero-Cross Detect Interrupt Flag bit

Value	Description
1	ZCD Output has changed (must be cleared in software)
0	ZCD Output has not changed

Bits 0, 1 – CxIF Comparator ‘x’ Interrupt Flag bit

Value	Description
1	Comparator Cx output has changed (must be cleared by software)
0	Comparator Cx output has not changed

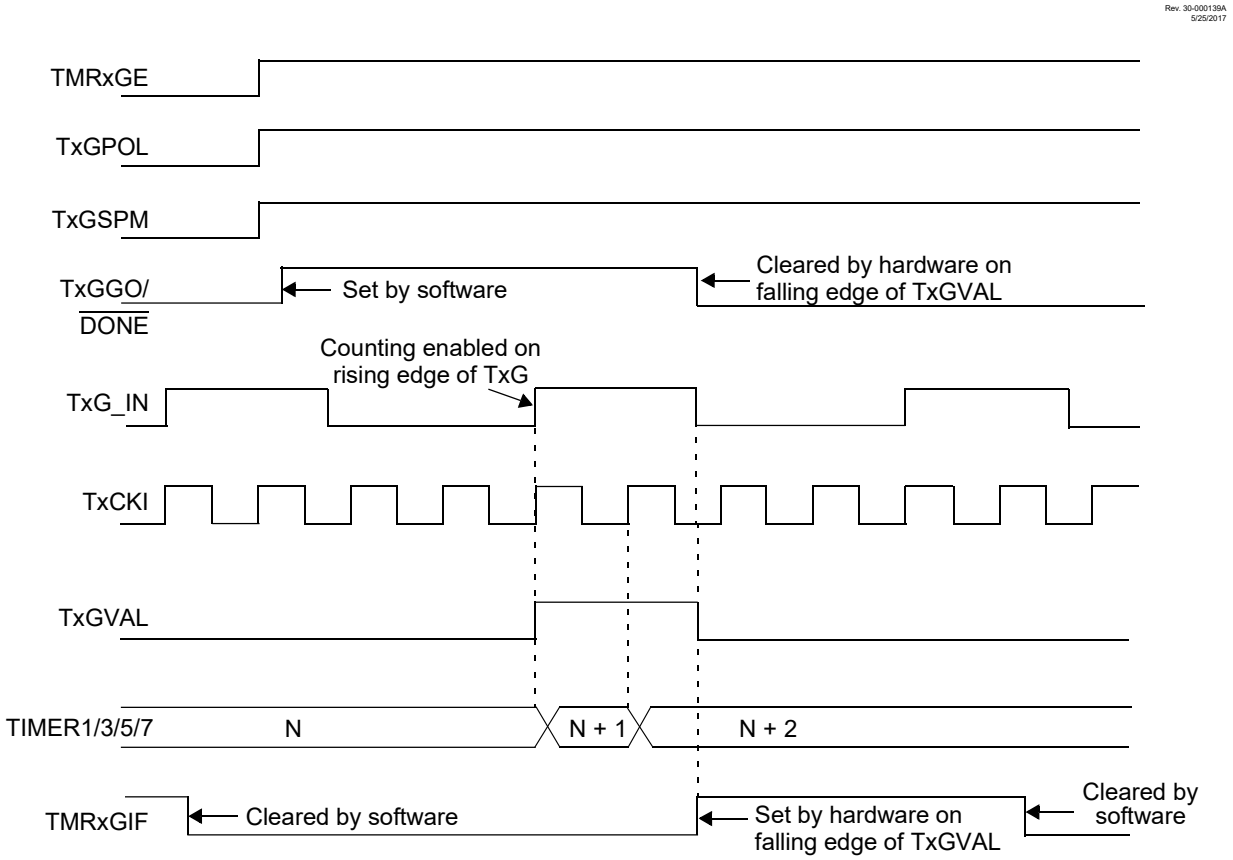
19.7.4 Timer1 Gate Single-Pulse Mode

When Timer1 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1 Gate Single-Pulse mode is first enabled by setting the 19.14.2.4 GSPM bit. Next, the 19.14.2.5 GGO/DONE bit must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the GGO/$\overline{\text{DONE}}$ bit is once again set in software.

Clearing the GSPM bit will also clear the GGO/$\overline{\text{DONE}}$ bit. See figure below for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. See figure below for timing details.

Figure 19-6. TIMER1 GATE SINGLE-PULSE MODE



20.1.3 Monostable Mode

Monostable modes are similar to One-Shot modes except that the ON bit is not cleared and the timer can be restarted by an external Reset event.

20.2 Timer2 Output

The Timer2 module's primary output is TMR2_postscaled, which pulses for a single TMR2_clk period upon each match of the postscaler counter and the OUTPS bits of the T2CON register. The postscaler is incremented each time the T2TMR value matches the T2PR value. This signal can be selected as an input to several other input modules:

- The ADC module, as an auto-conversion trigger
- CWG, as an auto-shutdown source
- The CRC memory scanner, as a trigger for triggered mode
- Gate source for odd numbered timers (Timer1, Timer3, etc.)
- Alternate SPI clock
- Reset signals for other instances of even numbered timers (Timer2, Timer4, etc.)

In addition, the Timer2 is also used by the CCP module for pulse generation in PWM mode. See *"PWM Overview"* and *"Pulse-width Modulation"* sections for more details on setting up Timer2 for use with the CCP and PWM modules.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

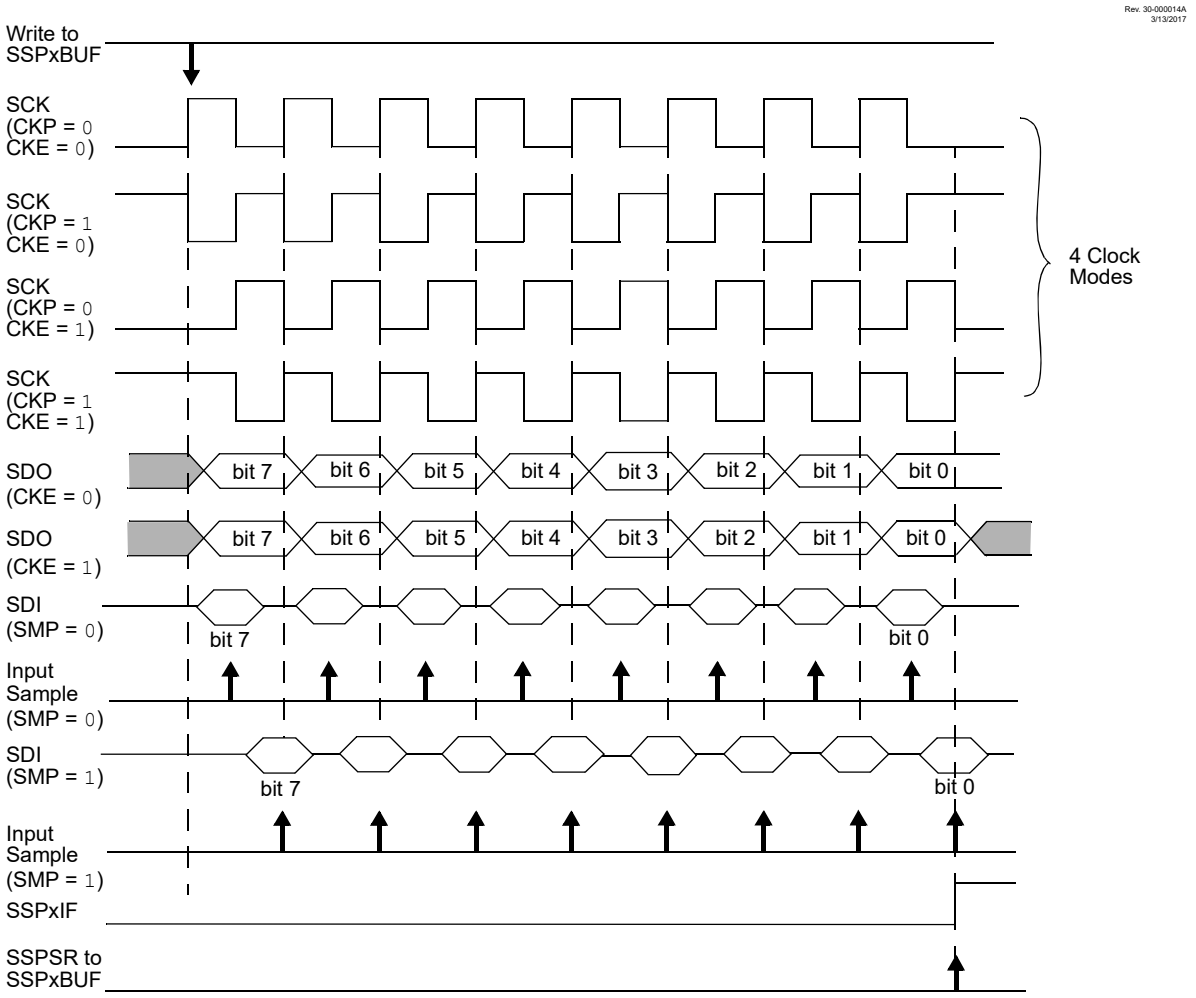
20.3 External Reset Sources

In addition to the clock source, the Timer2 also takes in an external Reset source. This external Reset source is selected for each timer with the corresponding TxRST register. This source can control starting and stopping of the timer, as well as resetting the timer, depending on which mode the timer is in. Reset source selections are shown in the following table.

Table 20-2. External Reset Sources

RSEL<3:0>	Reset Source		
	TMR2	TMR4	TMR6
1011-1111	Reserved	Reserved	Reserved
1010	ZCD_OUT	ZCD_OUT	ZCD_OUT
1001	CMP2OUT	CMP2OUT	CMP2OUT
1000	CMP1OUT	CMP1OUT	CMP1OUT
0111	PWM4OUT	PWM4OUT	PWM4OUT
0110	PWM3OUT	PWM3OUT	PWM3OUT
0101	CCP2OUT	CCP2OUT	CCP2OUT
0100	CCP1OUT	CCP1OUT	CCP1OUT

Figure 26-4. SPI Mode Waveform (Master Mode)



26.2.2 SPI Slave Mode

In Slave mode, the data is transmitted and received as external clock pulses appear on SCK. When the last bit is latched, the SSPxIF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the 26.9.2.4 CKP bit.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

26.2.3 Daisy-Chain Configuration

The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole

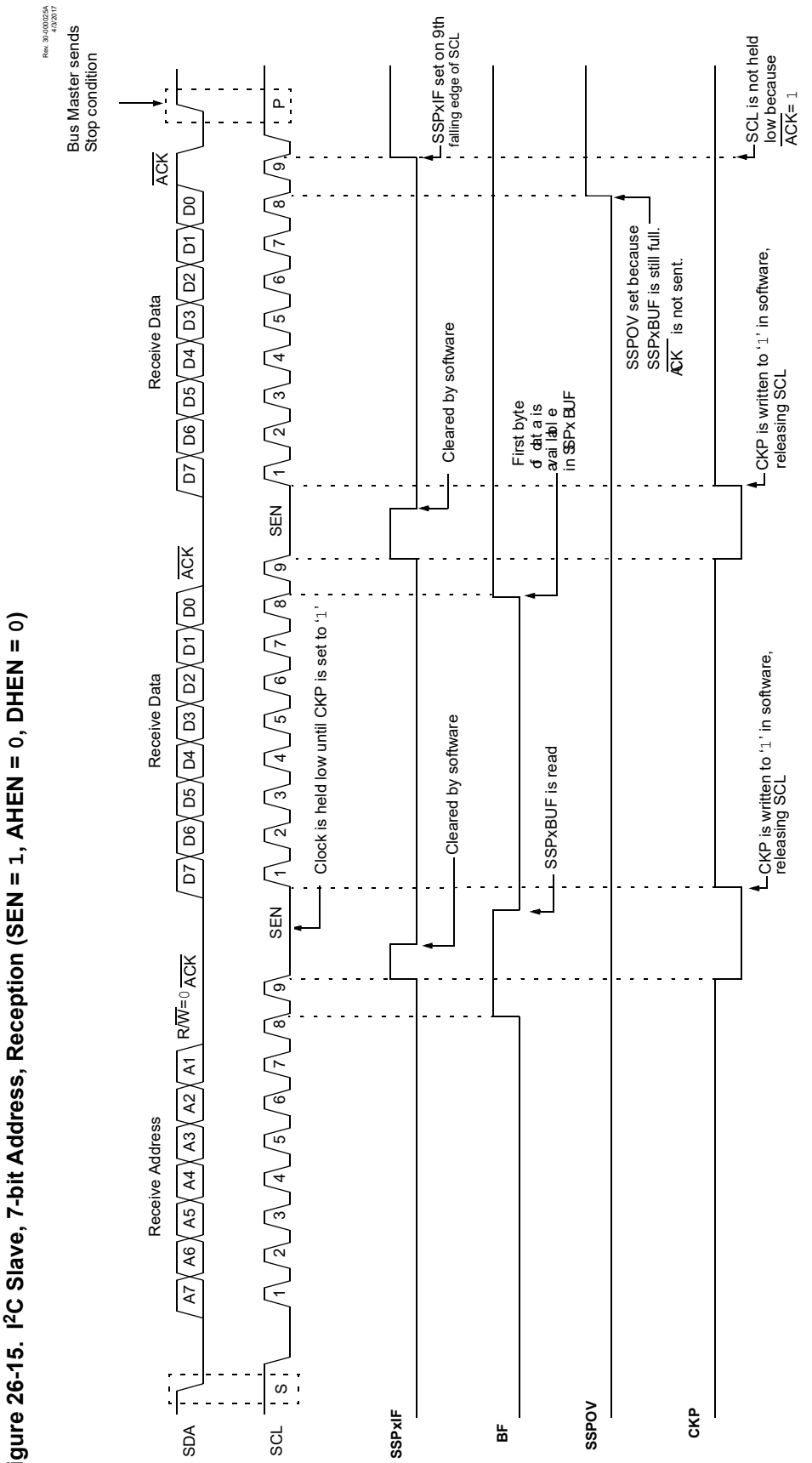


Figure 26-15. I²C Slave, 7-bit Address, Reception (SEN = 1, AHEN = 0, DHEN = 0)

Important:

1. If the master $\overline{\text{ACK}}$s then the clock will be stretched.
2. ACKSTAT is the only bit updated on the rising edge of the ninth SCL clock instead of the falling edge.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not $\overline{\text{ACK}}$; the clock is not held, but SSPxIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

26.9.2 SSPxCON1

Name: SSPxCON1
Address: 0x0F95

MSSP Control Register 1

Bit	7	6	5	4	3	2	1	0
	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
Access	R/W/HS	R/W/HS	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – WCOL
Write Collision Detect bit

Value	Mode	Description
1	SPI	A write to the SSPxBUF register was attempted while the previous byte was still transmitting (must be cleared by software)
1	I^2C Master transmit	A write to the SSPxBUF register was attempted while the I^2C conditions were not valid for a transmission to be started (must be cleared by software)
1	I^2C Slave transmit	The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0	SPI or I^2C Master or Slave transmit	No collision
x	Master or Slave receive	Don't care

Bit 6 – SSPOV
Receive Overflow Indicator bit(1)

Value	Mode	Description
1	SPI Slave	A byte is received while the SSPxBUF register is still holding the previous byte. The user must read SSPxBUF, even if only transmitting data, to avoid setting overflow. (must be cleared in software)
1	I^2C Receive	A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software)
0	SPI Slave or I^2C Receive	No overflow
x	SPI Master or I^2C Master transmit	Don't care

Bit 5 – SSPEN
Master Synchronous Serial Port Enable bit.(2)

recovery circuit samples a '0' in the Stop bit position then a framing error is set for this character, otherwise the framing error is cleared for this character. See the 27.1.2.4 Receive Framing Error section for more information on framing errors.

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the EUSART receive FIFO and the RCxIF interrupt flag bit of the PIRx register is set. The top character in the FIFO is transferred out of the FIFO by reading the RCxREG register.

Important: If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See the 27.1.2.5 Receive Overrun Error section for more information.

27.1.2.3 Receive Interrupts

The RCxIF interrupt flag bit of the PIRx register is set whenever the EUSART receiver is enabled and there is an unread character in the receive FIFO. The RCxIF interrupt flag bit is read-only, it cannot be set or cleared by software.

RCxIF interrupts are enabled by setting all of the following bits:

- RCxIE, Interrupt Enable bit of the PIEx register
- PEIE, Peripheral Interrupt Enable bit of the INTCON register
- GIE, Global Interrupt Enable bit of the INTCON register

The RCxIF interrupt flag bit will be set when there is an unread character in the FIFO, regardless of the state of interrupt enable bits.

27.1.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error Status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCxSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCxREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCxSTA register which resets the EUSART. Clearing the CREN bit of the RCxSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

Important: If all receive characters in the receive FIFO have framing errors, repeated reads of the RCxREG will not clear the FERR bit.

27.1.2.5 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCxSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCxSTA register or by resetting the EUSART by clearing the SPEN bit of the RCxSTA register.

27.5 Register Summary - EUSART

Address	Name	Bit Pos.								
0x0F98	RC1REG	7:0	RCREG[7:0]							
0x0F99	TX1REG	7:0	TXREG[7:0]							
0x0F9A	SP1BRG	7:0	SPBRGL[7:0]							
		15:8	SPBRGH[7:0]							
0x0F9C	RC1STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x0F9D	TX1STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x0F9E	BAUD1CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN

27.6 Register Definitions: EUSART Control

If CNT = 0;

PC = Address (ZERO)

If CNT ≠ 0;

PC = Address (NZERO)

INFSNZ	Increment f, skip if not 0			
Syntax:	INFSNZ f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) + 1 → dest, skip if result ≠ 0			
Status Affected:	None			
Encoding:	0100	10da	ffff	ffff
Description:	The contents of register ‘f’ are incremented. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). If the result is not ‘0’, the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. 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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three cycles if skip and followed by a 2-word instruction.			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

If skip:			
Q1	Q2	Q3	Q4

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Device Characteristics	Min.	Typ†	Max.	Units	Conditions
MEM23	V_{D_RW}	V_{DD} for Read or Erase/Write operation	V_{DDMIN}	—	V_{DDMAX}	V	
MEM24	T_{D_BEW}	Byte Erase and Write Cycle Time	—	10	TBD	ms	
Program Flash Memory Specifications							
MEM30	E_P	Flash Memory Cell Endurance	10k	—	—	E/W	-40°C≤Ta≤+85°C (**Note 1**)
MEM32	T_{P_RET}	Characteristic Retention	—	40	—	Year	Provided no other specifications are violated
MEM33	V_{P_RD}	V_{DD} for Read operation	V_{DDMIN}	—	V_{DDMAX}	V	
MEM34	V_{P_REW}	V_{DD} for Row Erase or Write operation	V_{DDMIN}	—	V_{DDMAX}	V	
MEM35	T_{P_REW}	Self-Timed Sector Write	—	6	TBD	ms	
MEM36	T_{SE}	Self-Timed Sector Erase	—	10	TBD	ms	
MEM37	T_{P_WRD}	Self-Timed Word Write	—	50	TBD	µs	

Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note:

1. Flash Memory Cell Endurance for the Flash memory is defined as: One Row Erase operation and one Self-Timed Write.

38.3.6 Thermal Characteristics

Table 38-6.

Standard Operating Conditions (unless otherwise stated)					
Param No.	Sym.	Characteristic	Typ.	Units	Conditions
TH01	θ_{JA}	Thermal Resistance Junction to Ambient	60	°C/W	28-pin SPDIP package
			80	°C/W	28-pin SOIC package
			90	°C/W	28-pin SSOP package
			27.5	°C/W	28-pin UQFN 4x4 mm package
			27.5	°C/W	28-pin QFN 6x6mm package
TH02	θ_{JC}	Thermal Resistance Junction to Case	31.4	°C/W	28-pin SPDIP package
			24	°C/W	28-pin SOIC package