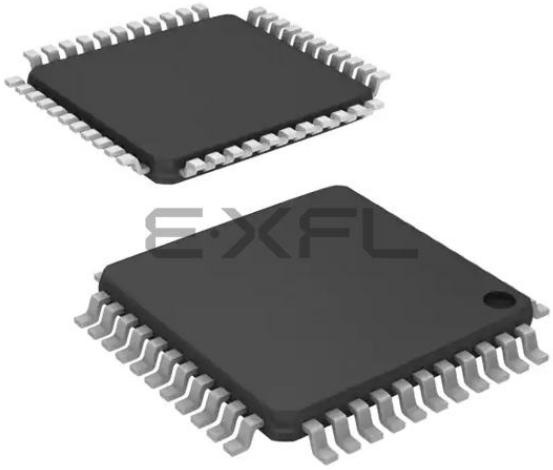


Welcome to E-XFL.COM

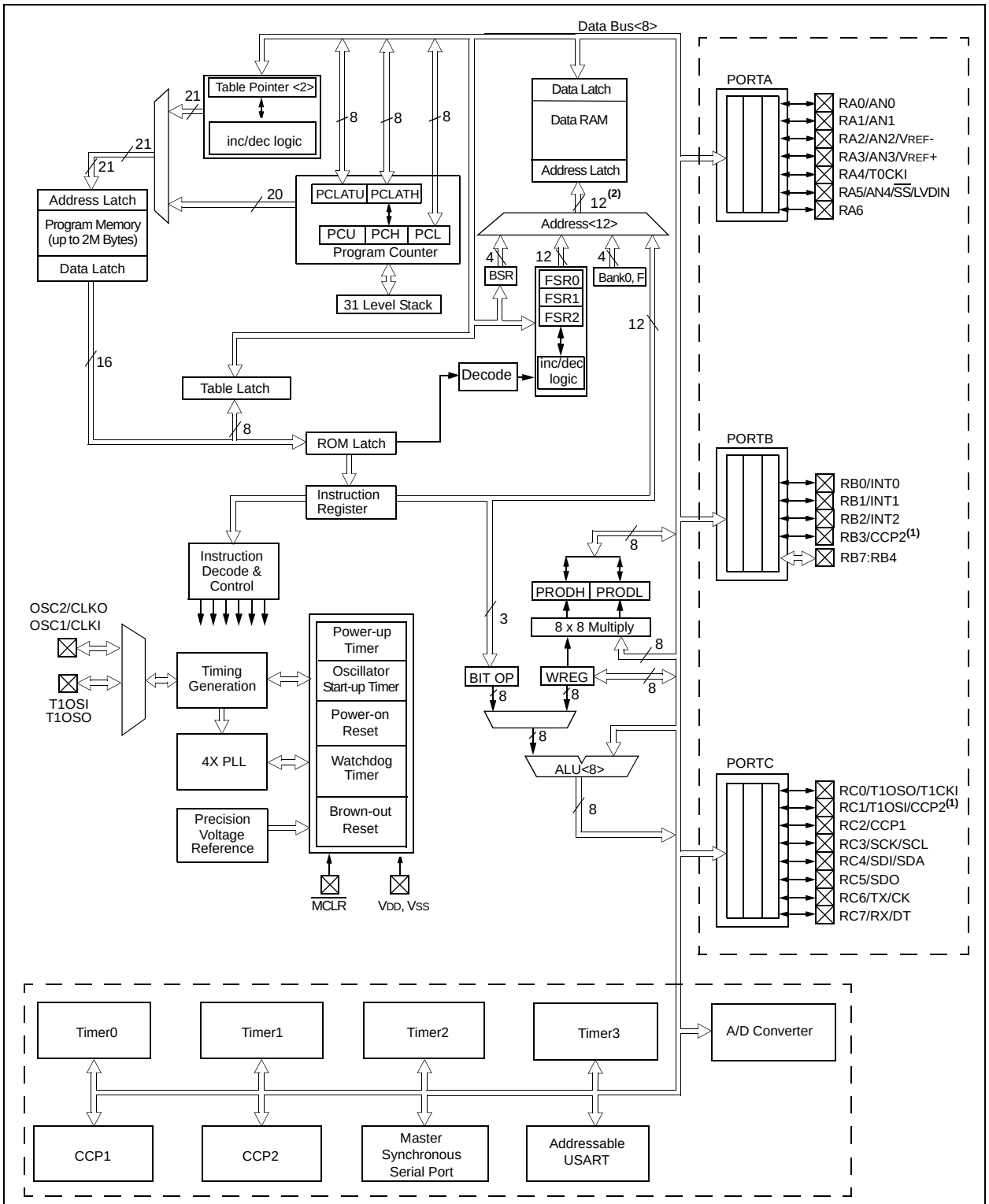
What is "Embedded - Microcontrollers"?

"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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	33
Program Memory Size	16KB (8K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	External
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/pic18lc442-i-pt

FIGURE 1-1: PIC18C2X2 BLOCK DIAGRAM

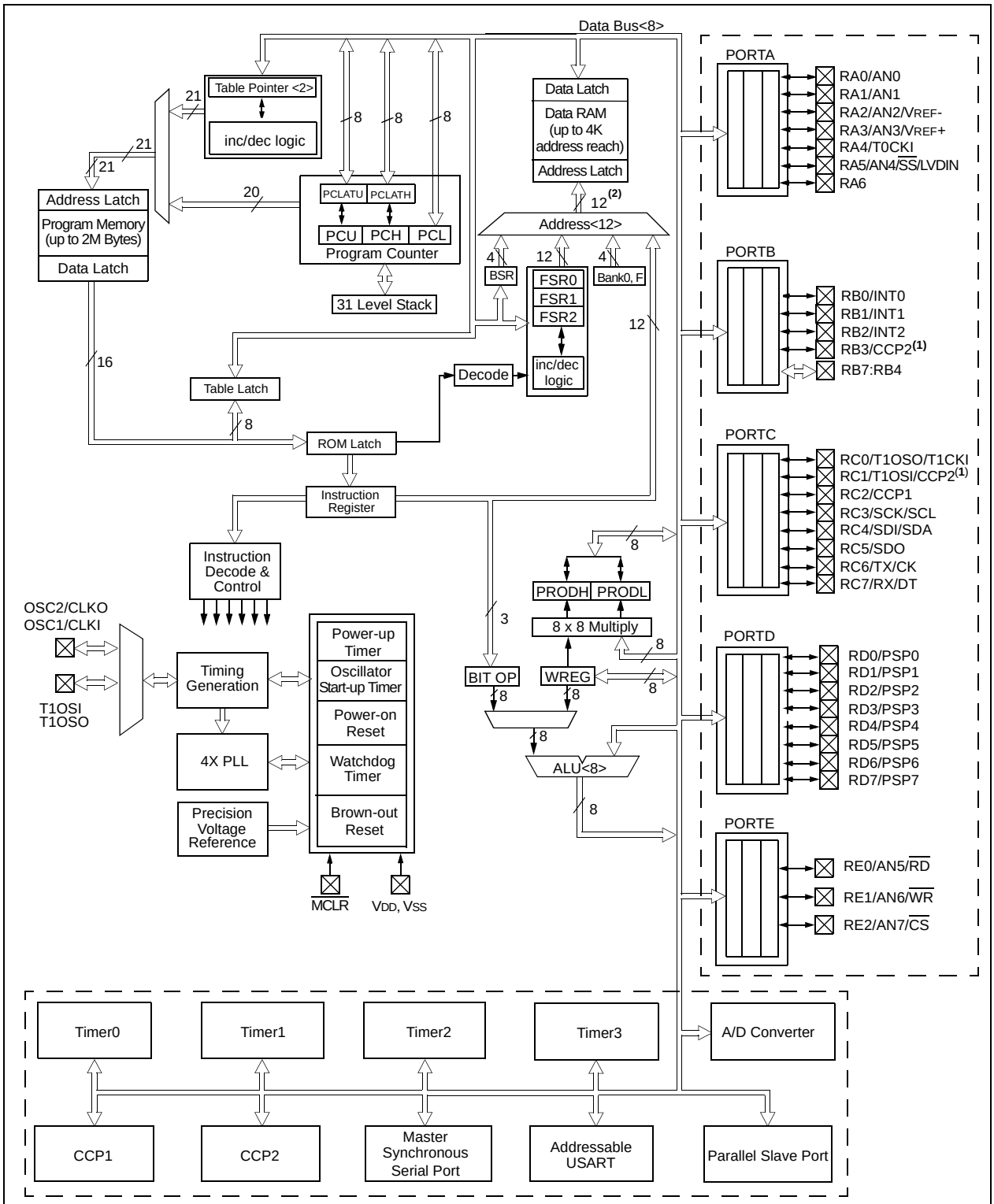


Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.

2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the `MOVFF` instruction).

3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

FIGURE 1-2: PIC18C4X2 BLOCK DIAGRAM



Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.

2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the `MOVFF` instruction).

3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

7.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Enable Registers (PIE1, PIE2). When IPEN = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 7-6: PERIPHERAL INTERRUPT ENABLE REGISTER 1 (PIE1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

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

7.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Priority Registers (IPR1, IPR2). The operation of the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 7-8: PERIPHERAL INTERRUPT PRIORITY REGISTER 1 (IPR1)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

bit 7 **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **ADIP**: A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP**: USART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP**: USART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP**: Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP**: CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

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

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

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

9.1 Timer0 Operation

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing the T0CS bit. In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting the T0CS bit. In Counter mode, Timer0 will increment either on every rising, or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed below.

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

9.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not readable or writable.

The PSA and T0PS2:T0PS0 bits determine the prescaler assignment and prescale ratio.

Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4,..., 1:256 are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g. `CLRF TMR0, MOVWF TMR0, BSF TMR0, x`....etc.) will clear the prescaler count.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment.

9.2.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed “on-the-fly” during program execution).

9.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF bit. The interrupt can be masked by clearing the TMR0IE bit. The TMR0IE bit must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut-off during SLEEP.

9.4 16-Bit Mode Timer Reads and Writes

TMR0H is not the high byte of the timer/counter in 16-bit mode, but is actually a buffered version of the high byte of Timer0 (refer to Figure 9-2). The high byte of the Timer0 counter/timer is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16-bits of Timer0 without having to verify that the read of the high and low byte were valid due to a rollover between successive reads of the high and low byte.

A write to the high byte of Timer0 must also take place through the TMR0H buffer register. Timer0 high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16-bits of Timer0 to be updated at once.

TABLE 9-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
TMR0L	Timer0 Module’s Low Byte Register								xxxx xxxx	uuuu uuuu
TMR0H	Timer0 Module’s High Byte Register								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

TABLE 10-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18C2X2 devices. Always maintain these bits clear.

REGISTER 14-2: SSPCON1: MSSP CONTROL REGISTER1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit

Master mode:

1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started

0 = No collision

Slave mode:

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit

In SPI mode:

1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode.
In Slave mode, the user must read the SSPBUF, even if only transmitting data to avoid setting overflow.
In Master mode, the overflow bit is not set, since each new reception (and transmission) is initiated by writing to the SSPBUF register (must be cleared in software).

0 = No overflow

In I^2C mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a “don’t care” in Transmit mode (must be cleared in software).

0 = No overflow

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

In both modes when enabled, these pins must be properly configured as input or output.

In SPI mode:

1 = Enables serial port and configures SCK, SDO, SDI, and $\overline{SS}$ as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

In I^2C mode:

1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

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

FIGURE 14-18: I²C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

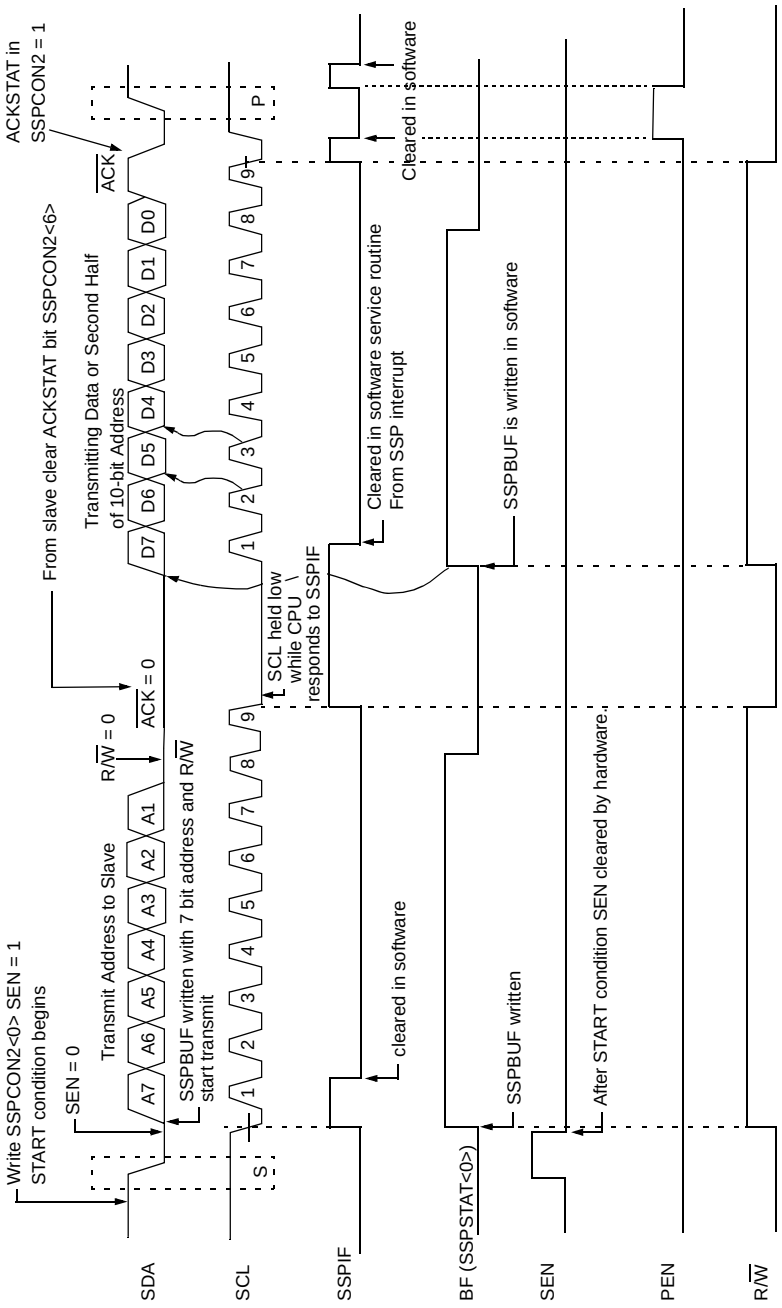
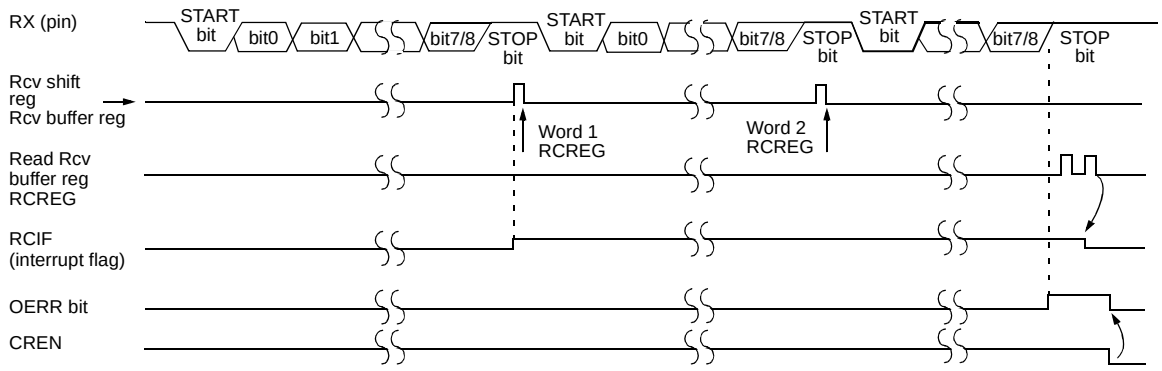


FIGURE 15-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

TABLE 15-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Asynchronous Reception.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18C2X2 devices. Always maintain these bits clear.

ADDWFC — ADD WREG and Carry bit to f

Syntax:	[*label*] ADDWFC f [,d [,a]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(WREG) + (f) + (C) → dest
Status Affected:	N,OV, C, DC, Z
Encoding:	0010 | 00da | ffff | ffff
Description:	Add WREG, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR will not be overridden.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ADDWFC REG, 0, 1`

Before Instruction

```
Carry bit= 1
REG    =   0x02
WREG   =   0x4D
```

After Instruction

```
Carry bit= 0
REG    =   0x02
WREG   =   0x50
```

ANDLW — AND literal with WREG

Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(WREG) .AND. k → WREG
Status Affected:	N,Z
Encoding:	0000 | 1011 | kkkk | kkkk
Description:	The contents of WREG are ANDed with the 8-bit literal 'k'. The result is placed in WREG.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ANDLW 0x5F`

Before Instruction

```
WREG   =   0xA3
```

After Instruction

```
WREG   =   0x03
```

GOTO — Unconditional Branch

Syntax:	[*label*] GOTO k
Operands:	$0 \le k \le 1048575$
Operation:	$k \rightarrow PC<20:1>$
Status Affected:	None
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 1111 / 1111 k_{19}kkk / $k_7$kkk kkkk / kkkk$_0$ kkkk$_8$
Description:	GOTO allows an unconditional branch anywhere within entire 2 Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `GOTO THERE`

After Instruction

```
PC  =  Address (THERE)
```

INCF — Increment f

Syntax:	[*label*] INCF f [,d [,a]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(f) + 1 \rightarrow dest$
Status Affected:	C,DC,N,OV,Z
Encoding:	0010 10da ffff ffff
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `INCF CNT, 1, 0`

Before Instruction

```
CNT  =  0xFF
Z    =  0
C    =  ?
DC   =  ?
```

After Instruction

```
CNT  =  0x00
Z    =  1
C    =  1
DC   =  1
```

RCALL	Relative Call
Syntax:	[*label*] RCALL n
Operands:	$-1024 \le n \le 1023$
Operation:	(PC) + 2 → TOS, (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1101 | 1nnn | nnnn | nnnn
Description:	Subroutine call with a jump up to 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' 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 a two-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE RCALL` Jump

Before Instruction

```
PC  =   Address(HERE)
```

After Instruction

```
PC  =   Address(Jump)
TOS =   Address (HERE+2)
```

RESET	Reset
Syntax:	[*label*] RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ reset.
Status Affected:	All
Encoding:	0000 | 0000 | 1111 | 1111
Description:	This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset in software.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start reset	No operation	No operation

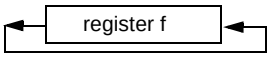
Example: `RESET`

After Instruction

```
Registers  =   Reset Value
Flags*     =   Reset Value
```

RLNCF — Rotate Left f (no carry)

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



Words:	1
Cycles:	1

Q Cycle Activity:

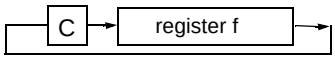
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RLNCF REG, 1, 0`

Before Instruction
```
REG     =   1010 1011
```
After Instruction
```
REG     =   0101 0111
```

RRCF — Rotate Right f through Carry

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



Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `RRCF REG, 0, 0`

Before Instruction
```
REG     =   1110 0110
C       =   0
```
After Instruction
```
REG     =   1110 0110
WREG    =   0111 0011
C       =   0
```

SUBLW — Subtract WREG from literal

Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k – (WREG) → WREG
Status Affected:	N,OV, C, DC, Z
Encoding:	0000 | 1000 | kkkk | kkkk
Description:	WREG is subtracted from the eight-bit literal 'k'. The result is placed in WREG.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example 1: `SUBLW 0x02`

Before Instruction

```
WREG    =   1
C       =   ?
```

After Instruction

```
WREG    =   1
C       =   1       ; result is positive
Z       =   0
N       =   0
```

Example 2: `SUBLW 0x02`

Before Instruction

```
WREG    =   2
C       =   ?
```

After Instruction

```
WREG    =   0
C       =   1       ; result is zero
Z       =   1
N       =   0
```

Example 3: `SUBLW 0x02`

Before Instruction

```
WREG    =   3
C       =   ?
```

After Instruction

```
WREG    =   FF    ; (2's complement)
C       =   0     ; result is negative
Z       =   0
N       =   1
```

SUBWF — Subtract WREG from f

Syntax:	[*label*] SUBWF f [,d [,a]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – (WREG) → dest
Status Affected:	N,OV, C, DC, Z
Encoding:	0101 | 11da | ffff | ffff
Description:	Subtract WREG from register 'f' (2's complement method). If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example 1: `SUBWF REG, 1, 0`

Before Instruction

```
REG     =   3
WREG    =   2
C       =   ?
```

After Instruction

```
REG     =   1
WREG    =   2
C       =   1       ; result is positive
Z       =   0
N       =   0
```

Example 2: `SUBWF REG, 0, 0`

Before Instruction

```
REG     =   2
WREG    =   2
C       =   ?
```

After Instruction

```
REG     =   2
WREG    =   0
C       =   1       ; result is zero
Z       =   1
N       =   0
```

Example 3: `SUBWF REG, 1, 0`

Before Instruction

```
REG     =   1
WREG    =   2
C       =   ?
```

After Instruction

```
REG     =   FFh     ;(2's complement)
WREG    =   2
C       =   0       ; result is negative
Z       =   0
N       =   1
```

TABLE 20-1: DEVELOPMENT TOOLS FROM MICROCHIP

		PIC12CXXX	PIC14000	PIC16C5X	PIC16C6X	PIC16CXXX	PIC16F62X	PIC16C7X	PIC16C7XX	PIC16C8X	PIC16F8XX	PIC16C9XX	PIC17C4X	PIC17C7XX	PIC18CXX2	24CXX/ 25CXX/ 93CXX	HCSXXX	MCRFXXX	MCP2510
Software Tools	MPLAB® Integrated Development Environment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	MPLAB® C17 C Compiler												✓	✓					
	MPLAB® C18 C Compiler														✓				
	MPASM™ Assembler/ MPLINK™ Object Linker	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Emulators	MPLAB® ICE In-Circuit Emulator	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓				
	ICEPIC™ In-Circuit Emulator	✓		✓	✓	✓		✓	✓	✓		✓							
Debugger	MPLAB® ICD In-Circuit Debugger				✓*			✓*			✓								
Programmers	PICSTART® Plus Entry Level Development Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓				
	PRO MATE® II Universal Device Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Demo Boards and Eval Kits	PICDEM™ 1 Demonstration Board			✓		✓		✓†		✓			✓						
	PICDEM™ 2 Demonstration Board				✓†			✓†							✓				
	PICDEM™ 3 Demonstration Board											✓							
	PICDEM™ 14A Demonstration Board		✓																
	PICDEM™ 17 Demonstration Board													✓					
	KEELOQ® Evaluation Kit																✓		
	KEELOQ® Transponder Kit																✓		
	microID™ Programmer's Kit																	✓	
	125 kHz microID™ Developer's Kit																	✓	
	125 kHz Anticollision microID™ Developer's Kit																	✓	
	13.56 MHz Anticollision microID™ Developer's Kit																	✓	
	MCP2510 CAN Developer's Kit																		✓

* Contact the Microchip Technology Inc. web site at www.microchip.com for information on how to use the MPLAB® ICD In-Circuit Debugger (DV164001) with PIC16C62, 63, 64, 65, 72, 73, 74, 76, 77.
** Contact Microchip Technology Inc. for availability date.
† Development tool is available on select devices.

FIGURE 21-14: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)

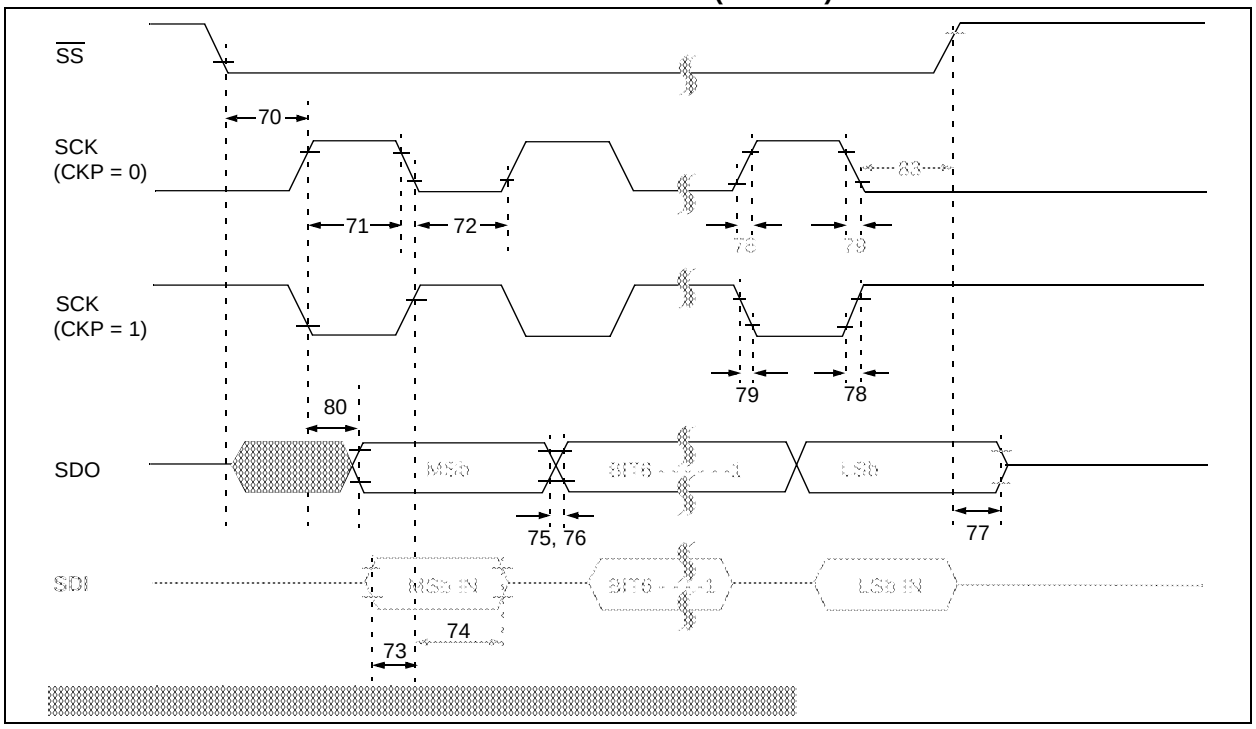
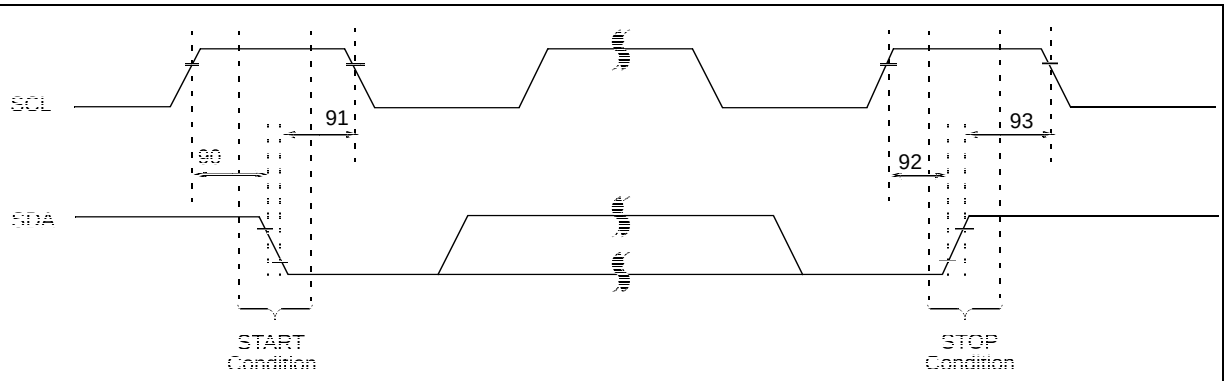


TABLE 21-13: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING (CKE = 0))

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$↓ to SCK↓ or SCK↑ input		TCY	—	ns	
71	TscH	SCK input high time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TscL	SCK input low time (Slave mode)	Continuous	1.25TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	ns	
73A	TB2B	Last clock edge of Byte1 to the first clock edge of Byte2		1.5TCY + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18**C**XXX	—	25	ns	
			PIC18**LC**XXX		45	ns	
76	TdoF	SDO data output fall time		—	25	ns	
77	TssH2doZ	$\overline{SS}$↑ to SDO output hi-impedance		10	50	ns	
78	TscR	SCK output rise time (Master mode)	PIC18**C**XXX	—	25	ns	
			PIC18**LC**XXX		45	ns	
79	TscF	SCK output fall time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18**C**XXX	—	50	ns	
			PIC18**LC**XXX		100	ns	
83	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK edge		1.5TCY + 40	—	ns	

Note 1: Requires the use of Parameter # 73A.
2: Only if Parameter # 71A and # 72A are used.

FIGURE 21-16: I²C BUS START/STOP BITS TIMING

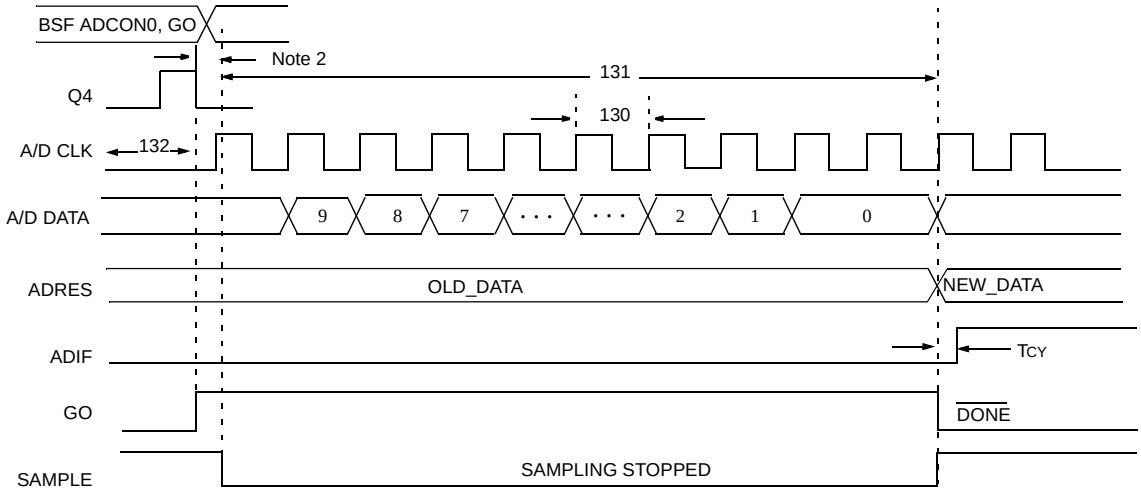


Note: Refer to Figure 21-4 for load conditions.

TABLE 21-15: I²C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	Tsu:sta	START condition Setup time	100 kHz mode	4700	—	ns	Only relevant for Repeated START condition
			400 kHz mode	600	—		
91	Thd:sta	START condition Hold time	100 kHz mode	4000	—	ns	After this period the first clock pulse is generated
			400 kHz mode	600	—		
92	Tsu:sto	STOP condition Setup time	100 kHz mode	4700	—	ns	
			400 kHz mode	600	—		
93	Thd:sto	STOP condition Hold time	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—		

FIGURE 21-22: A/D CONVERSION TIMING



Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

2: This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.

TABLE 21-22: A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D clock period	PIC18**C**XXX	1.6	20(5)	μs	TOSC based, VREF ≥ 3.0V
			PIC18**LC**XXX	3.0	20(5)	μs	TOSC based, VREF full range
			PIC18**C**XXX	2.0	6.0	μs	A/D RC mode
			PIC18**LC**XXX	3.0	9.0	μs	A/D RC mode
131	TCNV	Conversion time (not including acquisition time) **(Note 1)**		11	12	TAD	
132	TACQ	Acquisition time **(Note 3)**		15 10	— —	μs μs	-40°C ≤ Temp ≤ 125°C 0°C ≤ Temp ≤ 125°C
135	TSWC	Switching Time from convert → sample		—	**(Note 4)**		
136	TAMP	Amplifier settling time **(Note 2)**		1	—	μs	This may be used if the “new” input voltage has not changed by more than 1 LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).

Note 1: ADRES register may be read on the following TCY cycle.

2: See Section 16.0 for minimum conditions, when input voltage has changed more than 1 LSb.

3: The time for the holding capacitor to acquire the “New” input voltage, when the voltage changes full scale after the conversion (AVDD to AVSS, or AVSS to AVDD). The source impedance (*RS*) on the input channels is 50 Ω.

4: On the next Q4 cycle of the device clock.

5: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

APPENDIX A: REVISION HISTORY

Revision A (July 1999)

Original data sheet for PIC18CXX2 family.

Revision B (March 2001)

Added DC and AC characteristics graphs (Section 22.0).

Revision C (January 2013)

Added a note to each package outline drawing.

APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table 1.

TABLE 1: DEVICE DIFFERENCES

Feature	PIC18C242	PIC18C252	PIC18C442	PIC18C452
Program Memory (Kbytes)	16	32	16	32
Data Memory (Bytes)	512	1536	512	1536
A/D Channels	5	5	8	8
Parallel Slave Port (PSP)	No	No	Yes	Yes
Package Types	28-pin DIP 28-pin SOIC 28-pin JW	28-pin DIP 28-pin SOIC 28-pin JW	40-pin DIP 44-pin PLCC 44-pin TQFP 40-pin JW	40-pin DIP 44-pin PLCC 44-pin TQFP 40-pin JW

K

L