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"

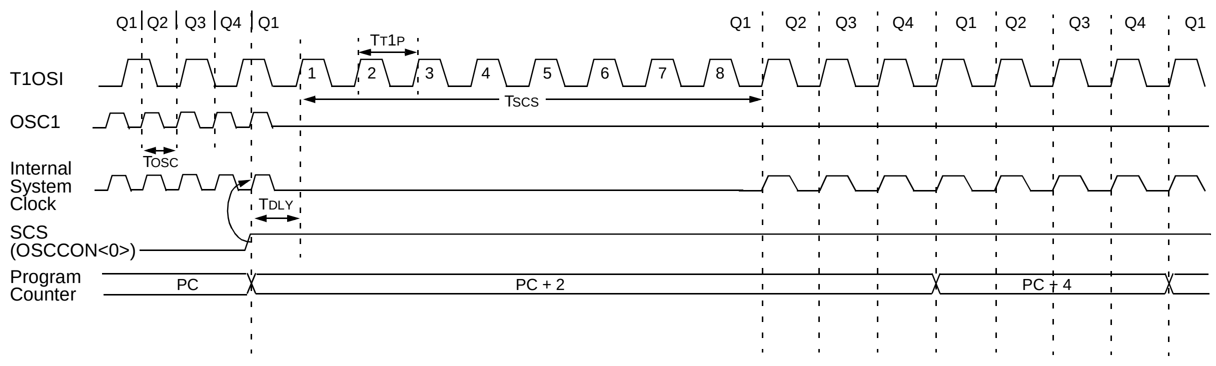
Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f8520t-i-pt

2.6.2 OSCILLATOR TRANSITIONS

PIC18FXX20 devices contain circuitry to prevent "glitches" when switching between oscillator sources. Essentially, the circuitry waits for eight rising edges of the clock source that the processor is switching to. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

A timing diagram indicating the transition from the main oscillator to the Timer1 oscillator is shown in Figure 2-8. The Timer1 oscillator is assumed to be running all the time. After the SCS bit is set, the processor is frozen at the next occurring Q1 cycle. After eight synchronization cycles are counted from the Timer1 oscillator, operation resumes. No additional delays are required after the synchronization cycles.

FIGURE 2-8: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR

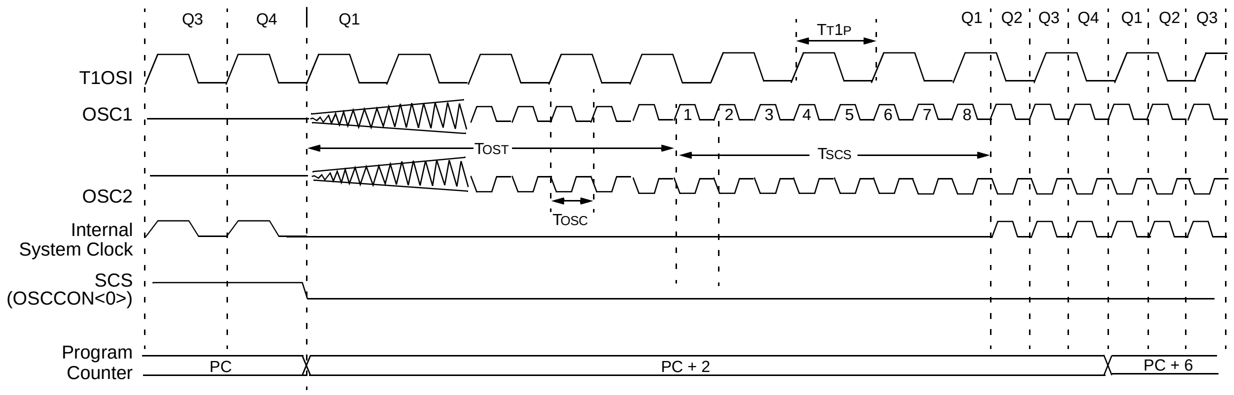


Note 1: Delay on internal system clock is eight oscillator cycles for synchronization.

The sequence of events that takes place when switching from the Timer1 oscillator to the main oscillator will depend on the mode of the main oscillator. In addition to eight clock cycles of the main oscillator, additional delays may take place.

If the main oscillator is configured for an external crystal (HS, XT, LP), then the transition will take place after an oscillator start-up time (TOST) has occurred. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for HS, XT and LP modes, is shown in Figure 2-9.

FIGURE 2-9: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS, XT, LP)



Note 1: TOST = 1024 TOSC (drawing not to scale).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
ADRESH	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	PIC18F6X20	PIC18F8X20	--00 0000	--00 0000	--uu uuuu
ADCON1	PIC18F6X20	PIC18F8X20	--00 0000	--00 0000	--uu uuuu
ADCON2	PIC18F6X20	PIC18F8X20	0--- -000	0--- -000	u--- -uuu
CCPR1H	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	PIC18F6X20	PIC18F8X20	--00 0000	--00 0000	--uu uuuu
CCPR2H	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	PIC18F6X20	PIC18F8X20	--00 0000	--00 0000	--uu uuuu
CCPR3H	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR3L	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP3CON	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
CVRCON	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
CMCON	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
TMR3H	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	PIC18F6X20	PIC18F8X20	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	PIC18F6X20	PIC18F8X20	0000 0000	uuuu uuuu	uuuu uuuu
PSPCON	PIC18F6X20	PIC18F8X20	0000 ----	0000 ----	uuuu ----
SPBRG1	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
RCREG1	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
TXREG1	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
TXSTA1	PIC18F6X20	PIC18F8X20	0000 -010	0000 -010	uuuu -uuu
RCSTA1	PIC18F6X20	PIC18F8X20	0000 000x	0000 000x	uuuu uuuu
EEADRH	PIC18F6X20	PIC18F8X20	---- --00	---- --00	---- --uu
EEADR	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
EEDATA	PIC18F6X20	PIC18F8X20	0000 0000	0000 0000	uuuu uuuu
EECON2	PIC18F6X20	PIC18F8X20	---- ----	---- ----	---- ----
EECON1	PIC18F6X20	PIC18F8X20	xx-0 x000	uu-0 u000	uu-0 u000

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they are read '0'.

4.7.1 TWO-WORD INSTRUCTIONS

The PIC18FXX20 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSBs set to '1's and is a special kind of NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-3. Refer to **Section 24.0 "Instruction Set Summary"** for further details of the instruction set.

EXAMPLE 4-3: TWO-WORD INSTRUCTIONS

CASE 1:	
Object Code	**Source Code**
`0110 0110 0000 0000`	`TSTFSZ    REG1        ; is RAM location 0?`
`1100 0001 0010 0011`	`MOVFF     REG1, REG2  ; No, execute 2-word instruction`
`1111 0100 0101 0110`	`                      ; 2nd operand holds address of REG2`
`0010 0100 0000 0000`	`ADDWF     REG3        ; continue code`
CASE 2:	
Object Code	**Source Code**
`0110 0110 0000 0000`	`TSTFSZ    REG1        ; is RAM location 0?`
`1100 0001 0010 0011`	`MOVFF     REG1, REG2  ; Yes`
`1111 0100 0101 0110`	`                      ; 2nd operand becomes NOP`
`0010 0100 0000 0000`	`ADDWF     REG3        ; continue code`

4.8 Look-up Tables

Look-up tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG 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 0xnn instructions, that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

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

4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Look-up table data may be stored 2 bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) specifies the byte address and the Table Latch (TABLAT) contains the data that is read from, or written to program memory. Data is transferred to/from program memory, one byte at a time.

A description of the table read/table write operation is shown in **Section 5.0 "Flash Program Memory"**.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2[(3)]	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2[(3)]	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2[(3)]	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2[(3)]	FBCh	CCPR2H	F9Ch	MEMCON[(2)]
FFBh	PCLATU	FDBh	PLUSW2[(3)]	FBBh	CCPR2L	F9Bh	—[(1)]
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	TRISJ
FF9h	PCL	FD9h	FSR2L	FB9h	CCPR3H	F99h	TRISH
FF8h	TBLPTRU	FD8h	STATUS	FB8h	CCPR3L	F98h	TRISG
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	CCP3CON	F97h	TRISF
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—[(1)]	F96h	TRISE
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD
FF4h	PRODH	FD4h	—[(1)]	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ
FF0h	INTCON3	FD0h	RCON	FB0h	PSPCON	F90h	LATH
FEFh	INDF0[(3)]	FCFh	TMR1H	FAFh	SPBRG1	F8Fh	LATG
FEEh	POSTINC0[(3)]	FCEh	TMR1L	FAEh	RCREG1	F8Eh	LATF
FEDh	POSTDEC0[(3)]	FCDh	T1CON	FADh	TXREG1	F8Dh	LATE
FECh	PREINC0[(3)]	FCCh	TMR2	FACh	TXSTA1	F8Ch	LATD
FEBh	PLUSW0[(3)]	FCBh	PR2	FABh	RCSTA1	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	EEADRH	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	PORTJ
FE7h	INDF1[(3)]	FC7h	SSPSTAT	FA7h	EECON2	F87h	PORTH
FE6h	POSTINC1[(3)]	FC6h	SSPCON1	FA6h	EECON1	F86h	PORTG
FE5h	POSTDEC1[(3)]	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF
FE4h	PREINC1[(3)]	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE
FE3h	PLUSW1[(3)]	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.
2: This register is unused on PIC18F6X20 devices. Always maintain this register clear.
3: This is not a physical register.

EXAMPLE 5-3: WRITING TO FLASH PROGRAM MEMORY

```
               MOVLW   D'64                    ; number of bytes in erase block
               MOVWF   COUNTER
               MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
               MOVWF   FSR0H
               MOVLW   BUFFER_ADDR_LOW
               MOVWF   FSR0L
               MOVLW   CODE_ADDR_UPPER         ; Load TBLPTR with the base
               MOVWF   TBLPTRU                 ; address of the memory block
               MOVLW   CODE_ADDR_HIGH
               MOVWF   TBLPTRH
               MOVLW   CODE_ADDR_LOW
               MOVWF   TBLPTRL
READ_BLOCK
               TBLRD*+                         ; read into TABLAT, and inc
               MOVF    TABLAT, W               ; get data
               MOVWF   POSTINC0                ; store data
               DECFSZ  COUNTER                 ; done?
               BRA     READ_BLOCK              ; repeat
MODIFY_WORD
               MOVLW   DATA_ADDR_HIGH          ; point to buffer
               MOVWF   FSR0H
               MOVLW   DATA_ADDR_LOW
               MOVWF   FSR0L
               MOVLW   NEW_DATA_LOW            ; update buffer word
               MOVWF   POSTINC0
               MOVLW   NEW_DATA_HIGH
               MOVWF   INDF0
ERASE_BLOCK
               MOVLW   CODE_ADDR_UPPER         ; load TBLPTR with the base
               MOVWF   TBLPTRU                 ; address of the memory block
               MOVLW   CODE_ADDR_HIGH
               MOVWF   TBLPTRH
               MOVLW   CODE_ADDR_LOW
               MOVWF   TBLPTRL
               BSF     EECON1, EEPGD           ; point to Flash program memory
               BCF     EECON1, CFGS            ; access Flash program memory
               BSF     EECON1, WREN            ; enable write to memory
               BSF     EECON1, FREE            ; enable Row Erase operation
               BCF     INTCON, GIE             ; disable interrupts
               MOVLW   55h
               MOVWF   EECON2                  ; write 55H
    Required   MOVLW   AAh
    Sequence   MOVWF   EECON2                  ; write AAH
               BSF     EECON1, WR              ; start erase (CPU stall)
               NOP
               BSF     INTCON, GIE             ; re-enable interrupts
               TBLRD*-                         ; dummy read decrement
WRITE_BUFFER_BACK
               MOVLW   8                       ; number of write buffer groups of 8 bytes
               MOVWF   COUNTER_HI
               MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
               MOVWF   FSR0H
               MOVLW   BUFFER_ADDR_LOW
               MOVWF   FSR0L
PROGRAM_LOOP
               MOVLW   8                       ; number of bytes in holding register
               MOVWF   COUNTER
WRITE_WORD_TO_HREGS
               MOVFF   POSTINC0, WREG          ; get low byte of buffer data
                                               ; present data to table latch
               TBLWT+*                         ; write data, perform a short write
                                               ; to internal TBLWT holding register.
               DECFSZ  COUNTER                 ; loop until buffers are full
               BRA     WRITE_WORD_TO_HREGS
```

REGISTER 7-1: EECON1 REGISTER (ADDRESS FA6h)

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD:** Flash Program/Data EEPROM Memory Select bit
1 = Access Flash program memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
1 = Access configuration or calibration registers
0 = Access Flash program or data EEPROM memory

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

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ or any WDT Reset during self-timed programming in normal operation)
0 = The write operation completed

Note: When a WRERR occurs, the EEPGD or FREE bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
1 = Allows write cycles to Flash program/data EEPROM
0 = Inhibits write cycles to Flash program/data EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle, or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read. (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

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

7.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADRH:EEADR register pair, clear the EEPGD control bit (EECON1<7>), clear the CFGS control bit (EECON1<6>) and then set the RD control bit (EECON1<0>). The data is available for the very next instruction cycle; therefore, the EEDATA register can be read by the next instruction. EEDATA will hold this value until another read operation, or until it is written to by the user (during a write operation).

EXAMPLE 7-1: DATA EEPROM READ

```
MOVLW   DATA_EE_ADDRH  ;
MOVWF   EEADRH         ; Upper bits of Data Memory Address to read
MOVLW   DATA_EE_ADDR   ;
MOVWF   EEADR          ; Lower bits of Data Memory Address to read
BCF     EECON1, EEPGD  ; Point to DATA memory
BCF     EECON1, CFGS   ; Access EEPROM
BSF     EECON1, RD     ; EEPROM Read
MOVF    EEDATA, W      ; W = EEDATA
```

7.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADRH:EEADR register pair and the data written to the EEDATA register. Then the sequence in Example 7-2 must be followed to initiate the write cycle.

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware

After a write sequence has been initiated, EECON1, EEADRH, EEADR and EEDATA cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. Both WR and WREN cannot be set with the same instruction.

At the completion of the write cycle, the WR bit is cleared in hardware and the EEPROM Write Complete Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt, or poll this bit. EEIF must be cleared by software.

EXAMPLE 7-2: DATA EEPROM WRITE

```
           MOVLW   DATA_EE_ADDRH      ;
           MOVWF   EEADRH             ; Upper bits of Data Memory Address to write
           MOVLW   DATA_EE_ADDR       ;
           MOVWF   EEADR              ; Lower bits of Data Memory Address to write
           MOVLW   DATA_EE_DATA       ;
           MOVWF   EEDATA             ; Data Memory Value to write
           BCF     EECON1, EEPGD      ; Point to DATA memory
           BCF     EECON1, CFGS       ; Access EEPROM
           BSF     EECON1, WREN       ; Enable writes

           BCF     INTCON, GIE        ; Disable Interrupts
           MOVLW   55h                ;
Required   MOVWF   EECON2             ; Write 55h
Sequence   MOVLW   AAh                ;
           MOVWF   EECON2             ; Write AAh
           BSF     EECON1, WR         ; Set WR bit to begin write
           BSF     INTCON, GIE        ; Enable Interrupts

                                      ; User code execution
           BCF     EECON1, WREN       ; Disable writes on write complete (EEIF set)
```

REGISTER 9-3: INTCON3 REGISTER

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
bit 7							bit 0

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **INT3IE:** INT3 External Interrupt Enable bit
1 = Enables the INT3 external interrupt
0 = Disables the INT3 external interrupt

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **INT3IF:** INT3 External Interrupt Flag bit
1 = The INT3 external interrupt occurred (must be cleared in software)
0 = The INT3 external interrupt did not occur

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

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

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 9-11: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

U-0	R/W-1	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **CMIP:** Comparator Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 4 **EEIP:** Data EEPROM/Flash Write Operation Interrupt Priority bit
1 = High priority
0 = Low priority

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

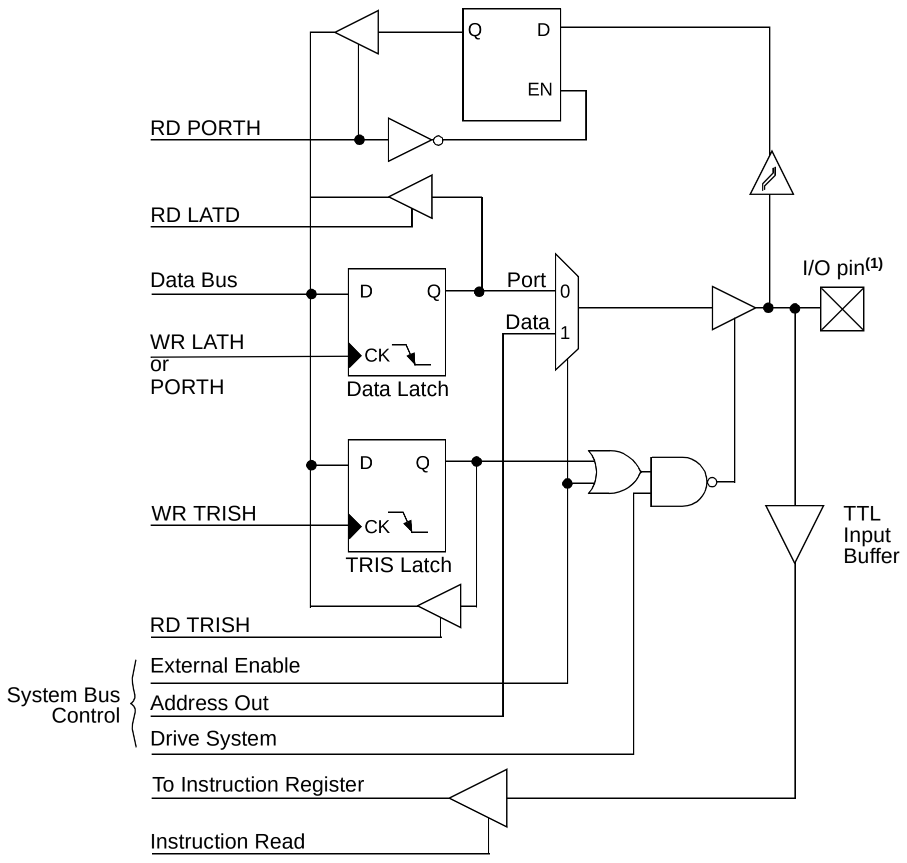
bit 2 **LVDIP:** Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 0 **CCP2IP:** CCP2 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

FIGURE 10-19: RH3:RH0 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

NOTES:

12.0 TIMER1 MODULE

The Timer1 module timer/counter has the following features:

- 16-bit timer/counter (two 8-bit registers: TMR1H and TMR1L)
- Readable and writable (both registers)
- Internal or external clock select
- Interrupt-on-overflow from FFFFh to 0000h
- Reset from CCP module special event trigger

Figure 12-1 is a simplified block diagram of the Timer1 module.

Register 12-1 details the Timer1 Control register. This register controls the operating mode of the Timer1 module and contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications, with only a minimal addition of external components and code overhead.

REGISTER 12-1: T1CON: TIMER1 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
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7	**RD16:** 16-bit Read/Write Mode Enable bit `1` = Enables register read/write of Timer1 in one 16-bit operation `0` = Enables register read/write of Timer1 in two 8-bit operations
bit 6	**Unimplemented:** Read as '`0`'
bit 5-4	**T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits `11` = 1:8 Prescale value `10` = 1:4 Prescale value `01` = 1:2 Prescale value `00` = 1:1 Prescale value
bit 3	**T1OSCEN:** Timer1 Oscillator Enable bit `1` = Timer1 oscillator is enabled `0` = Timer1 oscillator is shut off The oscillator inverter and feedback resistor are turned off to eliminate power drain.
bit 2	**$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit When TMR1CS = `1`: `1` = Do not synchronize external clock input `0` = Synchronize external clock input When TMR1CS = `0`: This bit is ignored. Timer1 uses the internal clock when TMR1CS = `0`.
bit 1	**TMR1CS:** Timer1 Clock Source Select bit `1` = External clock from pin RC0/T1OSO/T13CKI (on the rising edge) `0` = Internal clock (FOSC/4)
bit 0	**TMR1ON:** Timer1 On bit `1` = Enables Timer1 `0` = Stops Timer1

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

16.4 PWM Mode

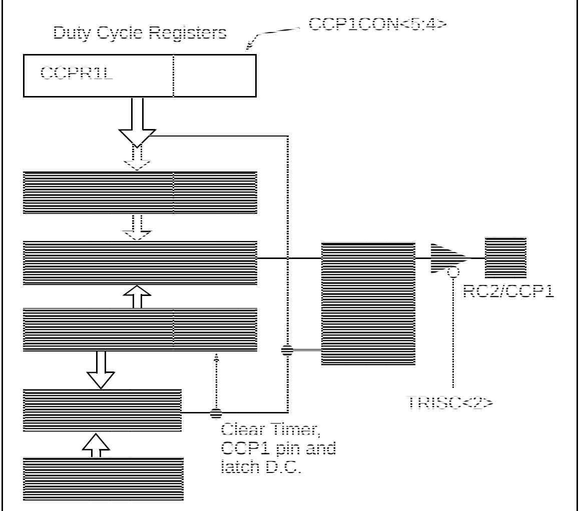
In Pulse Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note:	Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 16-4 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 16.4.3 “Setup for PWM Operation”**.

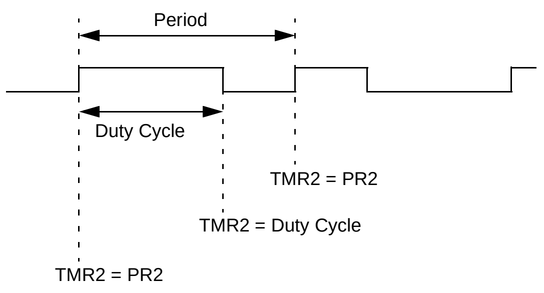
FIGURE 16-4: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time base.

A PWM output (Figure 16-5) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 16-5: PWM OUTPUT



16.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

EQUATION 16-1:

$$\text{PWM Period} = (\text{PR2}) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 and Timer4 postscalers (see **Section 13.0 “Timer2 Module”**) are not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

16.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

EQUATION 16-2:

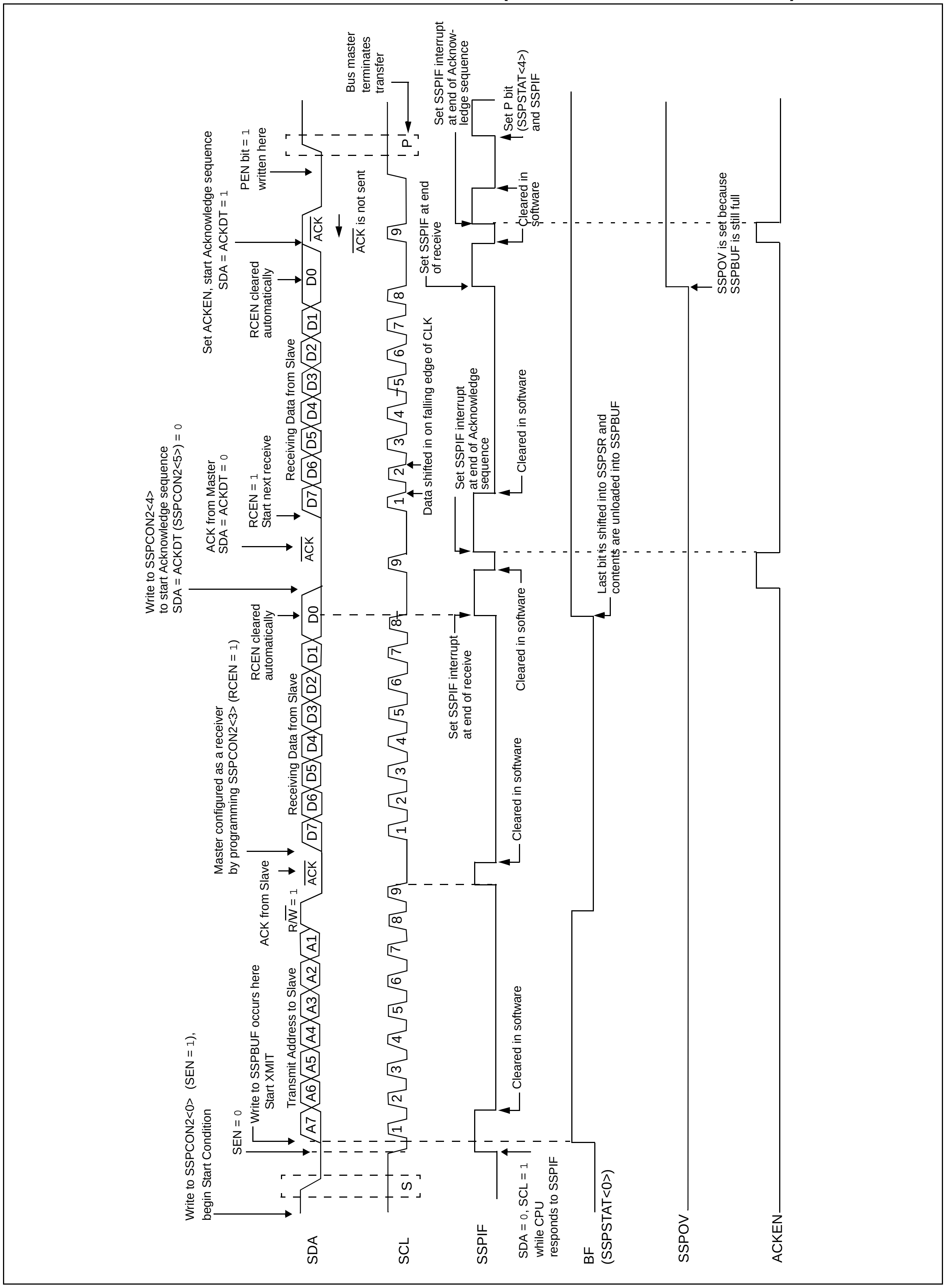
$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This doublebuffering is essential for glitchless PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

FIGURE 17-22: I^2C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



SDA
SCL
SSPIF
SDA = 0, SCL = 1 while CPU responds to SSPIF
BF (SSPSTAT<0>)
SSPOV
ACKEN
Write to SSPCON2<0> (SEN = 1), begin Start Condition
SEN = 0
Write to SSPBUF occurs here Start XMIT
Transmit Address to Slave
ACK from Slave
R/W = 1
Master configured as a receiver by programming SSPCON2<3> (RCEN = 1)
Receiving Data from Slave
RCEN cleared automatically
Write to SSPCON2<4> to start Acknowledge sequence SDA = ACKDT (SSPCON2<5>) = 0
ACK from Master SDA = ACKDT = 0
RCEN = 1 Start next receive
Set ACKEN, start Acknowledge sequence SDA = ACKDT = 1
PEN bit = 1 written here
ACK is not sent
Bus master terminates transfer
S
P
A7 A6 A5 A4 A3 A2 A1
ACK
D7 D6 D5 D4 D3 D2 D1 D0
1 2 3 4 5 6 7 8 9
Cleared in software
Set SSPIF interrupt at end of receive
Set SSPIF interrupt at end of Acknowledge sequence
Data shifted in on falling edge of CLK
Set SSPIF at end of receive
Set SSPIF interrupt at end of Acknow-ledge sequence
Set P bit (SSPSTAT<4>) and SSPIF
Last bit is shifted into SSPSR and contents are unloaded into SSPBUF
SSPOV is set because SSPBUF is still full

MOVFF	Move f to f
Syntax:	[*label*] MOVFF f_s,f_d
Operands:	$0 \le f_s \le 4095$ $0 \le f_d \le 4095$
Operation:	$(f_s) \rightarrow f_d$
Status Affected:	None

Encoding:

1st word (source)	1100	ffff	ffff	$\mathtt{ffff}_s$
2nd word (destin.)	1111	ffff	ffff	$\mathtt{ffff}_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh.
Either source or destination can be W (a useful special situation).
`MOVFF` is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).
The `MOVFF` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation, No dummy read	No operation	Write register 'f' (dest)

Example: `MOVFF REG1, REG2`

Before Instruction

REG1	=	0x33
REG2	=	0x11

After Instruction

REG1	=	0x33,
REG2	=	0x33

MOVLB	Move literal to low nibble in BSR
Syntax:	[*label*] MOVLB k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow$ BSR
Status Affected:	None

Encoding:

0000	0001	kkkk	kkkk

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `MOVLB 5`

Before Instruction

BSR register	=	0x02

After Instruction

BSR register	=	0x05

MOVLW **Move literal to W**

Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → W
Status Affected:	None
Encoding:	`0000` `1110` `kkkk` `kkkk`
Description:	The eight-bit literal 'k' is loaded into W.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVLW 0x5A`

After Instruction

W = 0x5A

MOVWF **Move W to f**

Syntax:	[*label*] MOVWF f [,a]
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) → f
Status Affected:	None
Encoding:	`0110` `111a` `ffff` `ffff`
Description:	Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank. 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 register 'f'

Example: `MOVWF REG, 0`

Before Instruction

W = 0x4F
REG = 0xFF

After Instruction

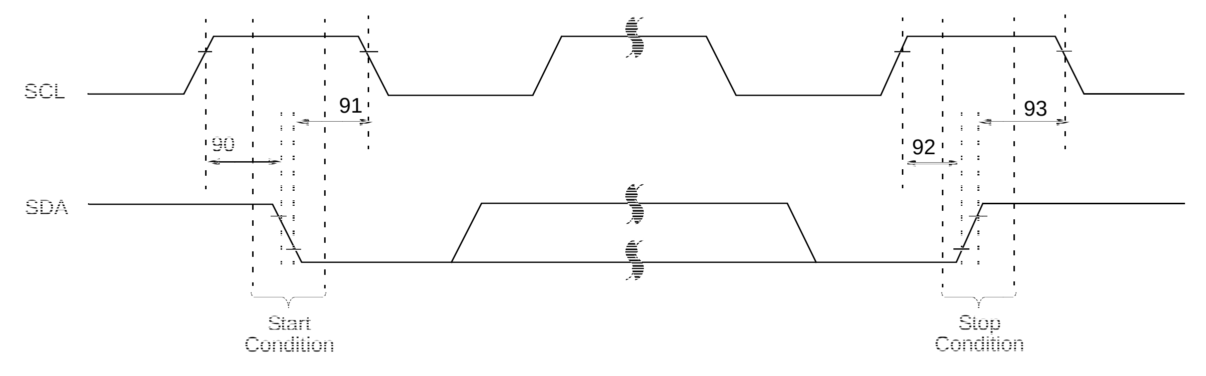
W = 0x4F
REG = 0x4F

TABLE 26-18: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)

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	1.25 TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TSCL	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73A	TB2B	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2		1.5 TCY + 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	PIC18FXX20	—	25	ns	
			PIC18LFXX20	—	45	ns	VDD = 2.0V
76	TDOF	SDO Data Output Fall Time		—	25	ns	
77	TSSH2DOZ	$\overline{SS}$ ↑ to SDO Output High-Impedance		10	50	ns	
78	TSCR	SCK Output Rise Time (Master mode)	PIC18FXX20	—	25	ns	
			PIC18LFXX20	—	45	ns	VDD = 2.0V
79	TSCF	SCK Output Fall Time (Master mode)		—	25	ns	
80	TSCH2DOV, TSCL2DOV	SDO Data Output Valid after SCK Edge	PIC18FXX20	—	50	ns	
			PIC18LFXX20	—	100	ns	VDD = 2.0V
82	TSSL2DOV	SDO Data Output Valid after $\overline{SS}$ ↓ Edge	PIC18FXX20	—	50	ns	
			PIC18LFXX20	—	100	ns	VDD = 2.0V
83	TSCH2SSH, TSCL2SSH	$\overline{SS}$ ↑ after SCK Edge		1.5 TCY + 40	—	ns	

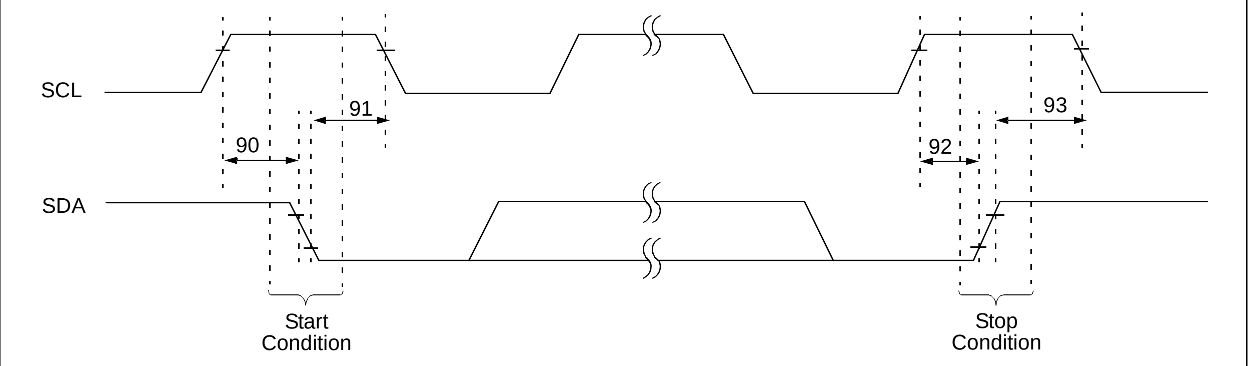
Note 1: Requires the use of Parameter #73A.
2: Only if Parameter #71A and #72A are used.

FIGURE 26-20: I²C BUS START/STOP BITS TIMING



Note: Refer to Figure 26-6 for load conditions.

FIGURE 26-22: MASTER SSP I^{2}C BUS START/STOP BITS TIMING WAVEFORMS



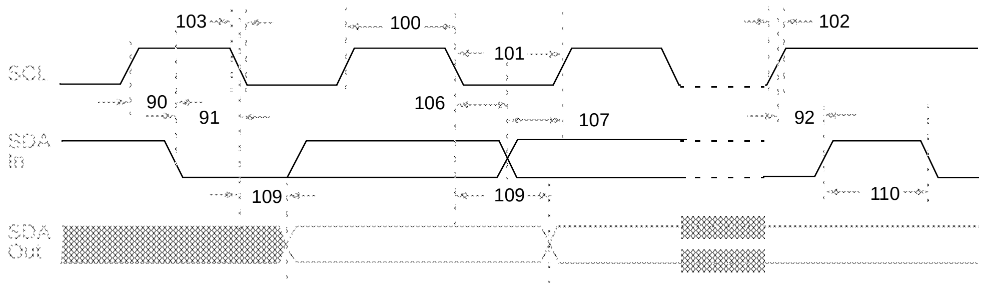
Note: Refer to Figure 26-6 for load conditions.

TABLE 26-21: MASTER SSP I^{2}C BUS START/STOP BITS REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	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)	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
93	THD:STO	Stop Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		

Note 1: Maximum pin capacitance = 10 pF for all I^{2}C pins.

FIGURE 26-23: MASTER SSP I^{2}C BUS DATA TIMING



Note: Refer to Figure 26-6 for load conditions.

NOTES: