Welcome to E-XFL.COM

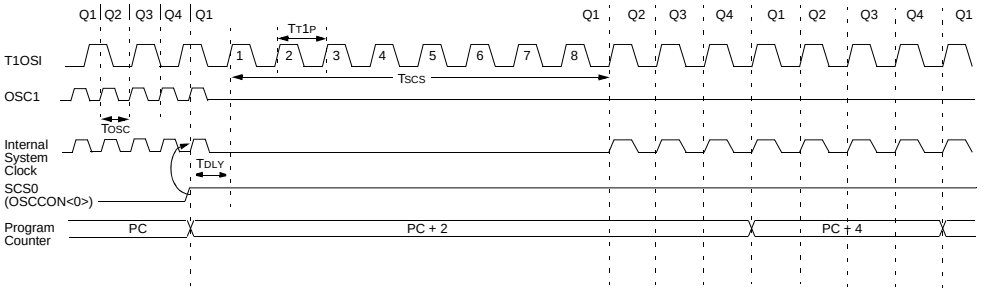
What is "Embedded - Microcontrollers"?

"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"

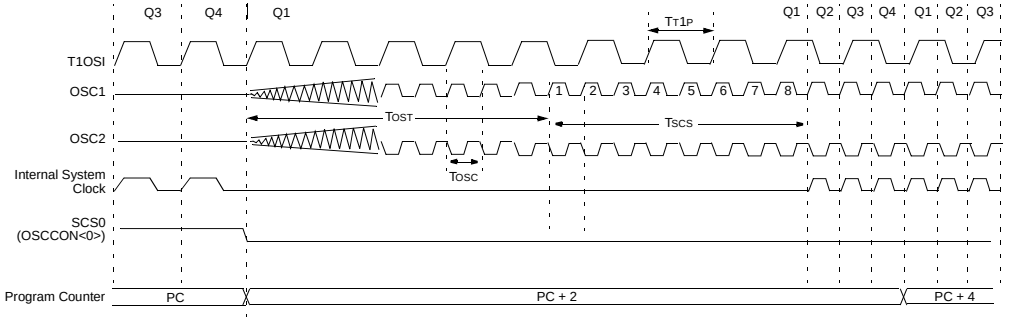
Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	25MHz
Connectivity	EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	37
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	84-LCC (J-Lead)
Supplier Device Package	84-PLCC (29.31x29.31)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18c801-i-l

FIGURE 2-6: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR



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

FIGURE 2-7: TIMING DIAGRAM FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS, LP)



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

4.6 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined, such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), two cycles are required to complete the instruction (Example 4-2).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the “Instruction Register” (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

4.7 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = ’0’). Figure 4-1 shows an example of how instruction words are stored in the program memory. To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read ’0’ (see Section 4.4).

The CALL and GOTO instructions have an absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 4-1 shows how the instruction “GOTO 0x06” is encoded in the program memory. Program branch instructions that encode a relative address offset operate in the same manner. The offset value stored in a branch instruction represents the number of single word instructions by which the PC will be offset. Section 20.0 provides further details of the instruction set.

EXAMPLE 4-2: INSTRUCTION PIPELINE FLOW

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. BRA SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, BIT3 (Forced NOP)				Fetch 4	Flush	
5. Instruction @ address SUB_1					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles, since the fetch instruction is “flushed” from the pipeline, while the new instruction is being fetched and then executed.

TABLE 4-1: INSTRUCTIONS IN PROGRAM MEMORY

Instruction	Opcode	Memory	Address
—	—	—	000007h
MOVLW 055h	0E55h	55h	000008h
		0Eh	000009h
GOTO 000006h	EF03h, F000h	03h	00000Ah
		EFh	00000Bh
		00h	00000Ch
		F0h	00000Dh
MOVFF 123h, 456h	C123h, F456h	23h	00000Eh
		C1h	00000Fh
		56h	000010h
		F4h	000011h
—	—	—	000012h

TABLE 4-2: REGISTER FILE SUMMARY - PIC18C601/801

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS(1)
FFFh	TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					---0 0000	---0 0000
FFEh	TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	0000 0000
FFDh	TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	0000 0000
FFCh	STKPTR	STKOVF	STKUNF	—	Return Stack Pointer					00-0 0000	00-0 0000
FFBh	PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000	---0 0000
FFAh	PCLATH	Holding Register for PC<15:8>								0000 0000	0000 0000
FF9h	PCL	PC Low Byte (PC<7:0>)								0000 0000	0000 0000
FF8h	TBLPTRU	—	—	r	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--r0 0000	--r0 0000
FF7h	TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	0000 0000
FF6h	TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	0000 0000
FF5h	TABLAT	Program Memory Table Latch								0000 0000	0000 0000
FF4h	PRODH	Product Register High Byte								xxxx xxxx	uuuu uuuu
FF3h	PRODL	Product Register Low Byte								xxxx xxxx	uuuu uuuu
FF2h	INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0E	RBIE	TMR0IF	INT0F	RBIF	0000 000x	0000 000u
FF1h	INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	T0IP	—	RBIP	1111 -1-1	1111 -1-1
FF0h	INTCON3	INT2P	INT1P	—	INT2E	INT1E	—	INT2F	INT1F	11-0 0-00	11-0 0-00
FEFh	INDF0	Uses contents of FSR0 to address data memory - value of FSR0 not changed (not a physical register)								N/A	N/A
FEEh	POSTINC0	Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register)								N/A	N/A
FEDh	POSTDEC0	Uses contents of FSR0 to address data memory - value of FSR0 post-decremented (not a physical register)								N/A	N/A
FECh	PREINC0	Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register)								N/A	N/A
FEBh	PLUSW0	Uses contents of FSR0 to address data memory -value of FSR0 offset by WREG (not a physical register)								N/A	N/A
FEAh	FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High				---- xxxx	---- uuuu
FE9h	FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	uuuu uuuu
FE8h	WREG	Working Register								xxxx xxxx	uuuu uuuu
FE7h	INDF1	Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register)								N/A	N/A
FE6h	POSTINC1	Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a physical register)								N/A	N/A
FE5h	POSTDEC1	Uses contents of FSR1 to address data memory - value of FSR1 post-decremented (not a physical register)								N/A	N/A
FE4h	PREINC1	Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register)								N/A	N/A
FE3h	PLUSW1	Uses contents of FSR1 to address data memory - value of FSR1 offset by WREG (not a physical register)								N/A	N/A
FE2h	FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High				---- xxxx	---- uuuu
FE1h	FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	uuuu uuuu
FE0h	BSR	—	—	—	—	Bank Select Register				---- 0000	---- 0000
FDFh	INDF2	Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register)								N/A	N/A
FDEh	POSTINC2	Uses contents of FSR2 to address data memory - value of FSR2 post-incremented (not a physical register)								N/A	N/A
FDDh	POSTDEC2	Uses contents of FSR2 to address data memory - value of FSR2 post-decremented (not a physical register)								N/A	N/A
FDCh	PREINC2	Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register)								N/A	N/A
FDBh	PLUSW2	Uses contents of FSR2 to address data memory -value of FSR2 offset by WREG (not a physical register)								N/A	N/A
FDAh	FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High				---- xxxx	---- uuuu
FD9h	FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	uuuu uuuu
FD8h	STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	---u uuuu

Legend x = unknown, u = unchanged, - = unimplemented, q = value depends on condition, r = reserved

Note 1: Other (non-power-up) RESETS include external RESET through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.
2: These registers can only be modified when the Combination Lock is open.
3: These registers are available on PIC18C801 only.

5.0 EXTERNAL MEMORY INTERFACE

The External Memory Interface is a feature of the PIC18C601/801 that allows the processor to access external memory devices, such as FLASH, EPROM, SRAM, etc. Memory mapped peripherals may also be accessed.

The External Memory Interface physical implementation includes up to 26 pins on the PIC18C601 and up to 38 pins on the PIC18C801. These pins are reserved for external address/data bus functions.

These pins are multiplexed with I/O port pins, but the I/O functions are only enabled when program execution takes place in internal Boot RAM and the EBDIS bit in the MEMCON register is set (see Register 5-1).

5.1 Memory Control Register (MEMCON)

Register 5-1 shows the Memory Control Register (MEMCON). This register contains bits used to control the operation of the External Memory Interface.

REGISTER 5-1: MEMCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0
bit7							bit0

bit 7 **EBDIS**: External Bus Disable

1 = External system bus disabled, all external bus drivers are mapped as I/O ports
0 = External system bus enabled, and I/O ports are disabled

bit 6 **PGRM**: Program RAM Enable

1 = 512 bytes of internal RAM enabled as internal program memory from location 1FFE00h to 1FFFFFh, external program memory at these locations is unused. Internal GPR memory from 400h to 5FFh is disabled and returns 00h.

0 = Internal RAM enabled as internal GPR memory from 400h to 5FFh. Program memory from location 1FFE00h to 1FFFFFh is configured as external program memory.

bit 5-4 **WAIT<1:0>**: Table Reads and Writes Bus Cycle Wait Count

11 = Table reads and writes will wait 0 TCY
10 = Table reads and writes will wait 1 TCY
01 = Table reads and writes will wait 2 TCY
00 = Table reads and writes will wait 3 TCY

bit 3-2 **Unimplemented**: Read as '0'

bit 1-0 **WM<1:0>**: TABLWT Operation with 16-bit Bus

1X = Word Write mode: TABLAT0 and TABLAT1 word output, $\overline{WRH}$ active when TABLAT1 written
01 = Byte Select mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ and ($\overline{UB}$ or $\overline{LB}$) will activate
00 = Byte Write mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ or $\overline{WRL}$ will activate

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

6.3 Table Write

Table Write operations store data from the data memory space into external program memory.

PIC18C601/801devices perform Table Writes one byte at a time. Table Writes to external memory are two-cycle instructions, unless wait states are enabled. The last cycle writes the data to the external memory location.

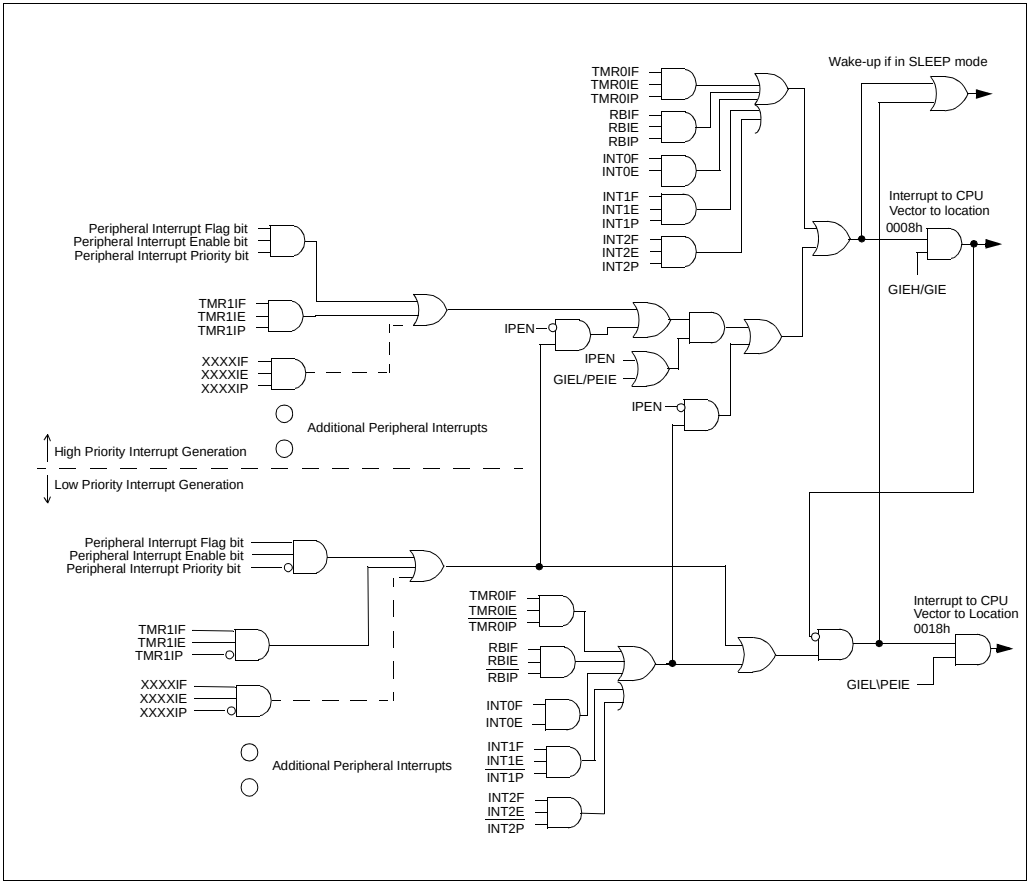
16-bit interface Table Writes depend on the type of external device that is connected and the WM<1:0> bits in the MEMCON register (See Figure 5-2).

Example 6-2 describes how to use TBLWT.

EXAMPLE 6-2: TABLE WRITE CODE EXAMPLE

```
; Write a byte to location 0020h
CLRF    TBLPTRU            ; clear upper 5 bits of TBLPTR
CLRF    TBLPTRH            ; clear higher 8 bits of TBLPTR
MOVLW   20h                ; Load 20h into
MOVWF   TBLPTRL            ; TBLPTRL
MOVLW   55h                ; Load 55h into
MOVWF   TBLAT              ; TBLAT
TBLWT*                     ; Write it
```

FIGURE 8-1: INTERRUPT LOGIC



9.2 PORTB, TRISB and LATB Registers

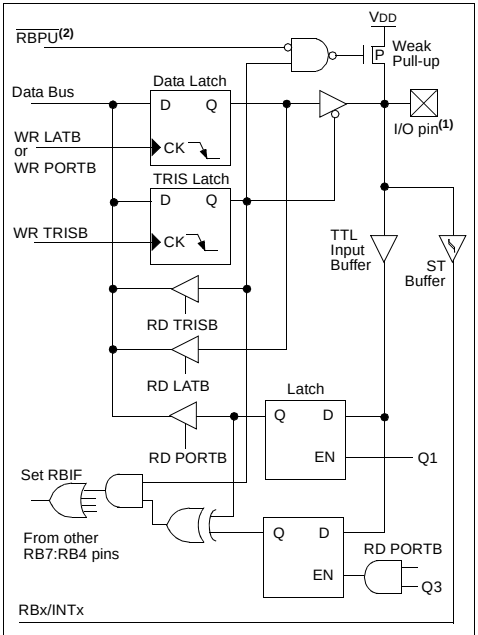
PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 9-2: INITIALIZING PORTB

```
CLRF    PORTB     ; Initialize PORTB by
                  ; clearing output
                  ; data latches
CLRF    LATB      ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW   0CFh      ; Value used to
                  ; initialize data
                  ; direction
MOVWF   TRISB     ; Set RB3:RB0 as inputs
                  ; RB5:RB4 as outputs
                  ; RB7:RB6 as inputs
```

FIGURE 9-3: RB7:RB4 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2 register).

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (INTCON2 register). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Pin RB3 is multiplexed with the CCP input/output. The weak pull-up for RB3 is disabled when the RB3 pin is configured as CCP pin. By disabling the weak pull-up when pin is configured as CCP, allows the remaining weak pull-up devices of PORTB to be used while the CCP is being used.

Four of PORTB's pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'd together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON register).

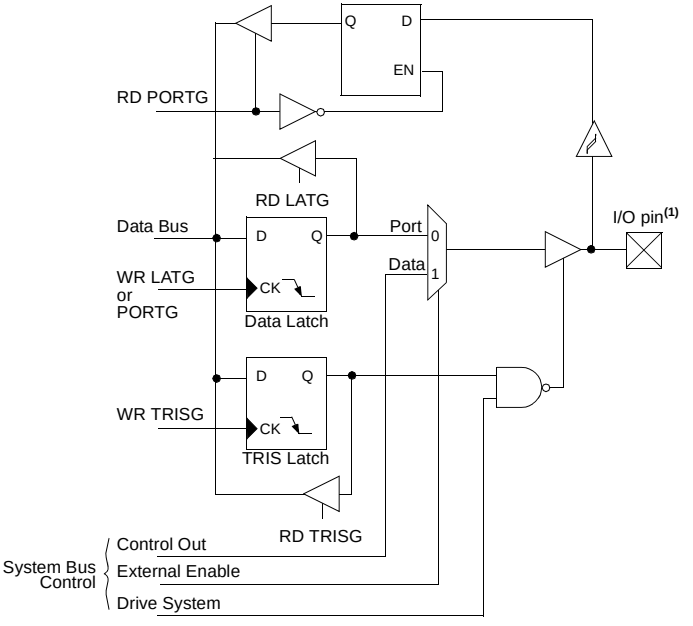
This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the `MOVFF` instruction). This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

FIGURE 9-15: PORTG BLOCK DIAGRAM IN SYSTEM BUS MODE



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

TABLE 9-13: PORTG FUNCTIONS

Name	Bit#	Buffer Type	Function
RG0/ALE	bit0	ST	Input/output port pin or Address Latch Enable signal for external memory
RG1/$\overline{OE}$	bit1	ST	Input/output port pin or Output Enable signal for external memory
RG2/$\overline{WRL}$	bit2	ST	Input/output port pin or Write Low byte signal for external memory
RG3/$\overline{WRH}$	bit3	ST	Input/output port pin or Write High byte signal for external memory
RG4/BA0	bit4	ST	Input/output port pin or Byte Address 0 signal for external memory

Legend: ST = Schmitt Trigger input

TABLE 9-14: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG

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
TRISG	PORTG Data Direction Control Register								---1 1111	---1 1111
PORTG	Read PORTG pin/Write PORTG Data Latch								---x xxxx	---u uuuu
LATG	Read PORTG Data Latch/Write PORTG Data Latch								---x xxxx	---u uuuu
MEMCON	EBDIS	PGRM	WAIT1	WAIT0	—	—	WM1	WM0	0000 --00	0000 --00

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTG.

10.1 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 TMR0L 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 TMR0L 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.

10.2 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.

10.2.1 SWITCHING PRESCALER ASSIGNMENT

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

10.3 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 TMR0IF 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.

10.4 16-Bit Mode Timer Reads and Writes

Timer0 can be set in 16-bit mode by clearing T0CON T08BIT. Registers TMR0H and TMR0L are used to access 16-bit timer value.

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 10-1). 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 the buffered value of TMR0H, when a write occurs to TMR0L. This allows all 16-bits of Timer0 to be updated at once.

TABLE 10-1: 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 11-2: 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	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-000 0000	-000 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	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

NOTES:

15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

15.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface™ (SPI)
- Inter-Integrated Circuit™ (I^2C)
  - Full Master mode
  - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

15.4.12 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated START/STOP condition, de-asserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the baud rate generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the baud rate generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count, in the event that the clock is held low by an external device (Figure 15-19).

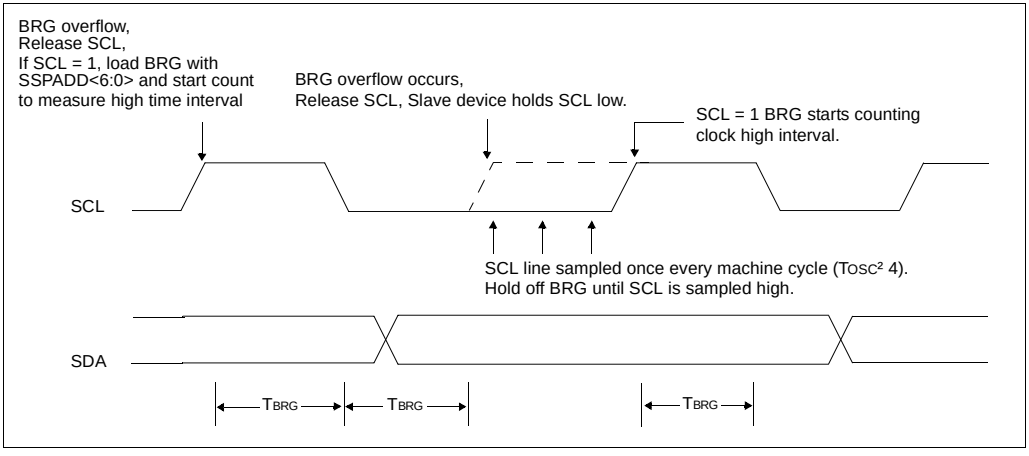
15.4.13 SLEEP OPERATION

While in SLEEP mode, the I^2C module can receive addresses or data, and when an address match or complete byte transfer occurs, wake the processor from SLEEP (if the MSSP interrupt is enabled).

15.4.14 EFFECT OF A RESET

A RESET disables the MSSP module and terminates the current transfer.

FIGURE 15-19: CLOCK ARBITRATION TIMING IN MASTER TRANSMIT MODE



18.1 Control Register

The Low Voltage Detect Control register (Register 18-1) controls the operation of the Low Voltage Detect circuitry.

REGISTER 18-1: LVDCON REGISTER

U-0	U-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit

1 = Indicates that the Low Voltage Detect logic will generate the interrupt flag at the specified voltage range

0 = Indicates that the Low Voltage Detect logic will not generate the interrupt flag at the specified voltage range and the LVD interrupt should not be enabled

bit 4 **LVDEN:** Low Voltage Detect Power Enable bit

1 = Enables LVD, powers up LVD circuit
0 = Disables LVD, powers down LVD circuit

bit 3-0 **LVDL3:LVDL0:** Low Voltage Detection Limit bits

1111 = External analog input is used (input comes from the LVDIN pin)
1110 = 4.5V
1101 = 4.2V
1100 = 4.0V - Reserved on PIC18C601/801
1011 = 3.8V - Reserved on PIC18C601/801
1010 = 3.6V - Reserved on PIC18C601/801
1001 = 3.5V - Reserved on PIC18C601/801
1000 = 3.3V - Reserved on PIC18C601/801
0111 = 3.0V - Reserved on PIC18C601/801
0110 = 2.8V - Reserved on PIC18C601/801
0101 = 2.7V - Reserved on PIC18C601/801
0100 = 2.5V - Reserved on PIC18C601/801
0011 = 2.4V - Reserved on PIC18C601/801
0010 = 2.2V - Reserved on PIC18C601/801
0001 = 2.0V - Reserved on PIC18C601/801
0000 = Reserved on PIC18C601/801 and PIC18LC801/601

LVDL3:LVDL0 modes which result in a trip point below the valid operating voltage of the device are not tested.

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

TABLE 20-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
ACCESS	ACCESS = 0: RAM access bit symbol
BANKED	BANKED = 1: RAM access bit symbol
bbb	Bit address within an 8-bit file register (0 to 7)
BSR	Bank Select Register. Used to select the current RAM bank.
d	Destination select bit; d = 0: store result in WREG, d = 1: store result in file register f.
dest	Destination either the WREG register or the specified register file location
f	8-bit Register file address (00h to FFh)
f_s	12-bit Register file address (000h to FFFh). This is the source address.
f_d	12-bit Register file address (000h to FFFh). This is the destination address.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value)
label	Label name
mm	The mode of the TBLPTR register for the Table Read and Table Write instructions Only used with Table Read and Table Write instructions:
*	No change to register (such as TBLPTR with Table reads and writes)
*+	Post-Increment register (such as TBLPTR with Table reads and writes)
*-	Post-Decrement register (such as TBLPTR with Table reads and writes)
+*	Pre-Increment register (such as TBLPTR with Table reads and writes)
n	The relative address (2's complement number) for relative branch instructions, or the direct address for Call/Branch and Return instructions
PRODH	Product of Multiply high byte (Register at address FF4h)
PRODL	Product of Multiply low byte (Register at address FF3h)
s	Fast Call / Return mode select bit. s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
u	Unused or Unchanged (Register at address FE8h)
W	W = 0: Destination select bit symbol
WREG	Working register (accumulator) (Register at address FE8h)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
TBLPTR	21-bit Table Pointer (points to a Program Memory location) (Register at address FF6h)
TABLAT	8-bit Table Latch (Register at address FF5h)
TOS	Top-of-Stack
PC	Program Counter
PCL	Program Counter Low Byte (Register at address FF9h)
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch (Register at address FFAh)
PCLATU	Program Counter Upper Byte Latch (Register at address FFBh)
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
TO	Time-out bit
PD	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

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 Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDWFC REG, W`

Before Instruction
C = 1
REG = 02h
WREG = 4Dh
N = ?
OV = ?
DC = ?
Z = ?

After Instruction
C = 0
REG = 02h
WREG = 50h
N = 0
OV = 0
DC = 0
Z = 0

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 AND'ed 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 5Fh`

Before Instruction
WREG = 0A3h
N = ?
Z = ?

After Instruction
WREG = 03h
N = 0
Z = 0

SWAPF	Swap nibbles in f
Syntax:	[*label*] SWAPF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>
Status Affected:	None

Encoding:

0011	10da	ffff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `SWAPF REG`

Before Instruction
REG = 53h

After Instruction
REG = 35h

TBLWT **Table Write**

Syntax: [*label*] TBLWT (*; *+; *-; +*)

Operands: None

Operation: if TBLWT*,
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;
TBLPTR - No Change;
if TBLWT*+,
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;
(TBLPTR) +1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;
(TBLPTR) -1 → TBLPTR;
if TBLWT+*,
(TBLPTR) +1 → TBLPTR;
(TABLAT) → Prog Mem (TBLPTR) or Holding Register;

Status Affected: None

Encoding:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction is used to program the contents of Program Memory (P.M.).

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2 MByte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2 (many if long write is to on-chip EPROM program memory)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register or Memory)

TBLWT (Cont.)

Example 1: `TBLWT *+;`

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 0FFh

After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
MEMORY(00A356h) = 55h

Example 2: `TBLWT +*;`

Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
MEMORY(01389Ah) = 0FFh
MEMORY(01389Bh) = 0FFh

After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
MEMORY(01389Ah) = 0FFh
MEMORY(01389Bh) = 34h

22.3 AC (Timing) Characteristics

22.3.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data-in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^2C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	STOP condition
STA	START condition		

NOTES:

