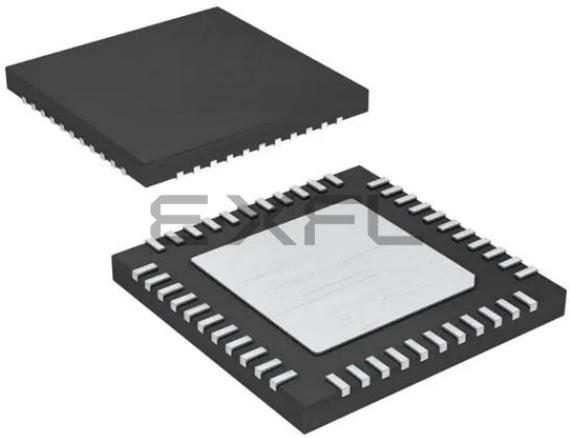


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f45k40-e-ml

REGISTER 3-3: Configuration Word 2L (30 0002h): Supervisor

R/W-1	R/W-1	R/W-1	U-1	U-1	U-1	R/W-1	R/W-1
BOREN<1:0>		$\overline{\text{LPBOREN}}$	—	—	—	$\overline{\text{PWRTE}}$	MCLRE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '1'	
-n = Value for blank device	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 **BOREN<1:0>:** Brown-out Reset Enable bits
When enabled, Brown-out Reset Voltage (V_{BOR}) is set by BORV bit
11 = Brown-out Reset enabled, SBOREN bit is ignored
10 = Brown-out Reset enabled while running, disabled in Sleep; SBOREN is ignored
01 = Brown-out Reset enabled according to SBOREN
00 = Brown-out Reset disabled

bit 5 **$\overline{\text{LPBOREN}}$**: Low-Power BOR Enable bit
1 = Low-Power Brown-out Reset is disabled
0 = Low-Power Brown-out Reset is enabled

bit 4-2 **Unimplemented:** Read as '1'

bit 1 **$\overline{\text{PWRTE}}$:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 0 **MCLRE:** Master Clear ($\overline{\text{MCLR}}$) Enable bit
If LVP = 1
RE3 pin function is $\overline{\text{MCLR}}$
If LVP = 0
1 = $\overline{\text{MCLR}}$ pin is $\overline{\text{MCLR}}$
0 = $\overline{\text{MCLR}}$ pin function is port defined function

4.3 Clock Source Types

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (ECH, ECM, ECL mode), quartz crystal resonators or ceramic resonators (LP, XT and HS modes).

A 4x PLL is provided that can be used in conjunction with the external clock. When used with the HFINTOSC the 4x PLL has input frequency limitations.See **Section 4.3.1.4 "4x PLL"** for more details.

Internal clock sources are contained within the oscillator module. The internal oscillator block has two internal oscillators that are used to generate internal system clock sources. The High-Frequency Internal Oscillator (HFINTOSC) can produce 1, 2, 4, 8, 12, 16, 32, 48 and 64 MHz clock. The frequency can be controlled through the OSCFRQ register (Register 4-5). The Low-Frequency Internal Oscillator (LFINTOSC) generates a fixed 31 kHz frequency.

The system clock can be selected between external or internal clock sources via the NOSC bits in the OSCCON1 register. See **Section 4.4 "Clock Switching"** for additional information. The system clock can be made available on the OSC2/CLKOUT pin for any of the modes that do not use the OSC2 pin. The clock out functionality is governed by the $\overline{\text{CLKOUTEN}}$ bit in the CONFIG1H register (Register 3-2). If enabled, the clock out signal is always at a frequency of FOSC/4.

4.3.1 EXTERNAL CLOCK SOURCES

An external clock source can be used as the device system clock by performing one of the following actions:

- Program the RSTOSC<2:0> and FEXTOSC<2:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Write the NOSC<2:0> and NDIV<3:0> bits in the OSCCON1 register to switch the system clock source.

See **Section 4.4 "Clock Switching"** for more information.

4.3.1.1 EC Mode

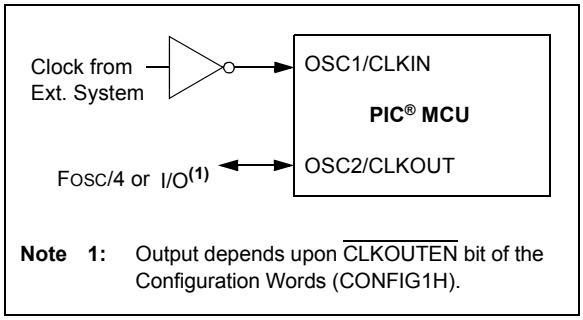
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. Figure 4-2 shows the pin connections for EC mode.

EC mode has three power modes to select from through Configuration Words:

- ECH – High power, above 8 MHz
- ECM – Medium power, 100 kHz-8 MHz
- ECL – Low power, below 100 MHz

The Oscillator Start-up Timer (OST) is disabled when EC mode is selected. Therefore, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

FIGURE 4-2: EXTERNAL CLOCK (EC) MODE OPERATION



Note 1: Output depends upon $\overline{\text{CLKOUTEN}}$ bit of the Configuration Words (CONFIG1H).

4.3.1.2 LP, XT, HS Modes

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 4-3). The three modes select a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification (above 100 kHz - 8 MHz).

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

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

4.3.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision digitally-controlled internal clock source that produces a stable clock up to 64 MHz. The HFINTOSC can be enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits in Configuration Word 1 to '110' (FOSC = 1 MHz) or '000' (FOSC = 64 MHz) to set the oscillator upon device Power-up or Reset.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See **Section 4.4 "Clock Switching"** for more information.

The HFINTOSC frequency can be selected by setting the HFFRQ<3:0> bits of the OSCFRQ register.

The NDIV<3:0> bits of the OSCCON1 register allow for division of the HFINTOSC output from a range between 1:1 and 1:512.

4.3.2.2 MFINTOSC

The module provides two (500 kHz and 31.25 kHz) constant clock outputs. These clocks are digital divisors of the HFINTOSC clock. Dynamic divider logic is used to provide constant MFINTOSC clock rates for all settings of HFINTOSC.

The MFINTOSC cannot be used to drive the system but it is used to clock certain modules such as the Timers and WWDT.

4.3.2.3 Internal Oscillator Frequency Adjustment

The internal oscillator is factory-calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register (Register 4-3).

The default value of the OSCTUNE register is 00h. The value is a 6-bit two's complement number. A value of 1Fh will provide an adjustment to the maximum frequency. A value of 20h will provide an adjustment to the minimum frequency.

When the OSCTUNE register is modified, the oscillator frequency will begin shifting to the new frequency. Code execution continues during this shift. There is no indication that the shift has occurred.

OSCTUNE **does not affect** the LFINTOSC frequency. Operation of features that depend on the LFINTOSC clock source frequency, such as the Power-up Timer (PWRT), WWDT, Fail-Safe Clock Monitor (FSCM) and peripherals, are *not* affected by the change in frequency.

4.3.2.4 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a factory-calibrated 31 kHz internal clock source.

The LFINTOSC is the frequency for the Power-up Timer (PWRT), Windowed Watchdog Timer (WWDT) and Fail-Safe Clock Monitor (FSCM).

The LFINTOSC is enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits of Configuration Word 1 to enable LFINTOSC.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See **Section 4.4, Clock Switching** for more information.

4.3.2.5 ADCRC

The ADCRC is an oscillator dedicated to the ADC^2 module. The ADCRC oscillator can be manually enabled using the ADOEN bit of the OSCEN register. The ADCRC runs at a fixed frequency of 600 kHz. ADCRC is automatically enabled if it is selected as the clock source for the ADC^2 module.

10.5 Register Definitions: Status

REGISTER 10-2: STATUS: STATUS REGISTER

U-0	R-1/q	R-1/q	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	$\overline{TO}$	$\overline{PD}$	N	OV	Z	DC	C
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **$\overline{TO}$:** Time-Out bit
1 = Set at power-up or by execution of CLRWDT or SLEEP instruction
0 = A WDT time-out occurred

bit 5 **$\overline{PD}$:** Power-Down bit
1 = Set at power-up or by execution of CLRWDT instruction
0 = Set by execution of the SLEEP instruction

bit 4 **N:** Negative bit used for signed arithmetic (2’s complement); indicates if the result is negative, (ALU MSb = 1).
1 = The result is negative
0 = The result is positive

bit 3 **OV:** Overflow bit used for signed arithmetic (2’s complement); indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.
1 = Overflow occurred for current signed arithmetic operation
0 = No overflow occurred

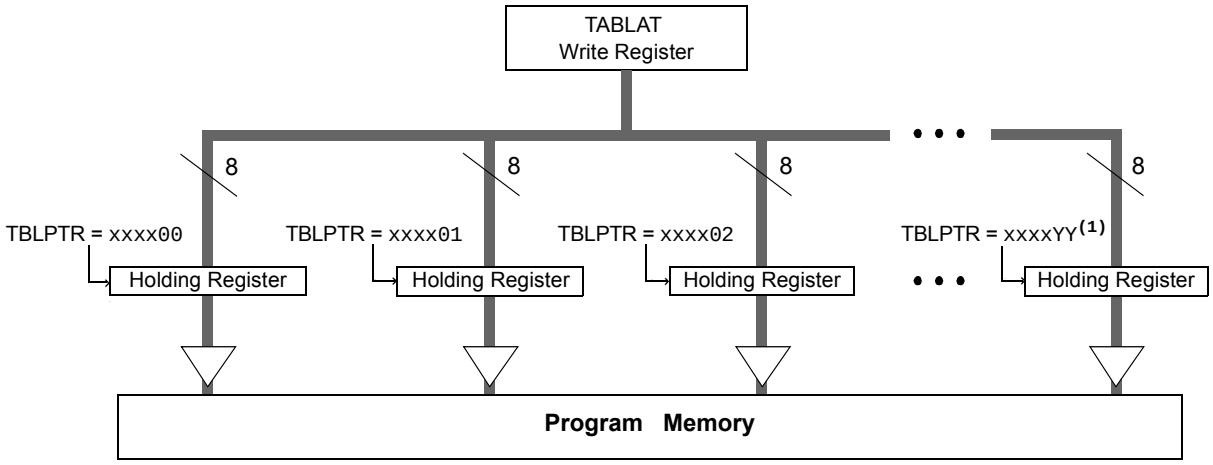
bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{Borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)**(1)**
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/$\overline{Borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)**(1,2)**
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note 1: For $\overline{Borrow}$, the polarity is reversed. A subtraction is executed by adding the two’s complement of the second operand.
2: For Rotate (RRF, RLF) instructions, this bit is loaded with either the high or low-order bit of the Source register.

FIGURE 11-8: TABLE WRITES TO PROGRAM FLASH MEMORY



Note 1: Refer to Table 11-3 for number of holding registers (e.g., YY = 3F for 64 holding registers).

11.1.6.1 Program Flash Memory Write Sequence

The sequence of events for programming an internal program memory location should be:

1. Read appropriate number of bytes into RAM. Refer to Table 11-2 for Write latch size.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the block erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the n-byte block into the holding registers with auto-increment. Refer to Table 11-2 for Write latch size.
7. Set NVMREG<1:0> bits to point to program memory.
8. Clear FREE bit and set WREN bit in NVMCON1 register.
9. Disable interrupts.
10. Execute the unlock sequence (see **Section 11.1.4 "NVM Unlock Sequence"**).
11. WR bit is set in NVMCON1 register.
12. The CPU will stall for the duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Verify the memory (table read).

This procedure will require about 6 ms to update each write block of memory. An example of the required code is given in Example 11-4.

Note:	Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the bytes in the holding registers.

13.11.7 IN-CIRCUIT DEBUG (ICD) INTERACTION

The scanner freezes when an ICD halt occurs, and remains frozen until user-mode operation resumes. The debugger may inspect the SCANCON0 and SCANLADR registers to determine the state of the scan.

The ICD interaction with each operating mode is summarized in Table 13-4.

TABLE 13-4: ICD AND SCANNER INTERACTIONS

ICD Halt	Scanner Operating Mode		
	Peek	**Concurrent Triggered**	**Burst**
External Halt	If scanner would peek an instruction that is not executed (because of ICD entry), the peek will occur after ICD exit, when the instruction executes.	If external halt is asserted during a scan cycle, the instruction (delayed by scan) may or may not execute before ICD entry, depending on external halt timing.	If external halt is asserted during the BSF(SCANCON.GO), ICD entry occurs, and the burst is delayed until ICD exit. Otherwise, the current NVM-access cycle will complete, and then the scanner will be interrupted for ICD entry.
		If external halt is asserted during the cycle immediately prior to the scan cycle, both scan and instruction execution happen after the ICD exits.	If external halt is asserted during the burst, the burst is suspended and will resume with ICD exit.
PC Breakpoint		Scan cycle occurs before ICD entry and instruction execution happens after the ICD exits.	If PCPB (or single step) is on BSF(SCANCON.GO), the ICD is entered before execution; execution of the burst will occur at ICD exit, and the burst will run to completion. Note that the burst can be interrupted by an external halt.
Data Breakpoint		The instruction with the dataBP executes and ICD entry occurs immediately after. If scan is requested during that cycle, the scan cycle is postponed until the ICD exits.	
Single Step		If a scan cycle is ready after the debug instruction is executed, the scan will read PFM and then the ICD is re-entered.	
SWBP and ICDINST		If scan would stall a SWBP, the scan cycle occurs and the ICD is entered.	If SWBP replaces BSF(SCANCON.GO), the ICD will be entered; instruction execution will occur at ICD exit (from ICDINSTR register), and the burst will run to completion.

13.11.8 PERIPHERAL MODULE DISABLE

Both the CRC and scanner module can be disabled individually by setting the CRCMD and SCANMD bits of the PMD0 register (Register 7-1). The SCANMD can be used to enable or disable to the scanner module only if the SCANE bit of Configuration Word 4 is set. If the SCANE bit is cleared, then the scanner module is not available for use and the SCANMD bit is ignored.

REGISTER 14-9: PIR7: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 7

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0
SCANIF	CRCIF	NVMIF	—	—	—	—	CWG1IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **SCANIF:** SCAN Interrupt Flag bit
1 = SCAN interrupt has occurred (must be cleared in software)
0 = SCAN interrupt has not occurred or has not been started

bit 6 **CRCIF:** CRC Interrupt Flag bit
1 = CRC interrupt has occurred (must be cleared in software)
0 = CRC interrupt has not occurred or has not been started

bit 5 **NVMIF:** NVM Interrupt Flag bit
1 = NVM interrupt has occurred (must be cleared in software)
0 = NVM interrupt has not occurred or has not been started

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **CWG1IF:** CWG Interrupt Flag bit
1 = CWG interrupt has occurred (must be cleared in software)
0 = CWG interrupt has not occurred or has not been started

TABLE 14-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	170
PIE0	—	—	TMR0IE	IOCIE	—	INT2IE	INT1IE	INT0IE	179
PIE1	OSCFIE	CSWIE	—	—	—	—	ADTIE	ADIE	180
PIE2	HLVDIE	ZCDIE	—	—	—	—	C2IE	C1IE	181
PIE3	RC2IE	TX2IE	RC1IE	TX1IE	BCL2IE	SSP2IE	BCL1IE	SSP1IE	182
PIE4	—	—	TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE	183
PIE5	—	—	—	—	—	TMR5GIE	TMR3GIE	TMR1GIE	184
PIE6	—	—	—	—	—	—	CCP2IE	CCP1IE	185
PIE7	SCANIE	CRCIE	NVMIE	—	—	—	—	CWG1IE	186
PIR0	—	—	TMR0IF	IOCIF	—	INT2IF	INT1IF	INT0IF	171
PIR1	OSCFIF	CSWIF	—	—	—	—	ADTIF	ADIF	172
PIR2	HLVDIF	ZCDIF	—	—	—	—	C2IF	C1IF	173
PIR3	RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF	174
PIR4	—	—	TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF	175
PIR5	—	—	—	—	—	TMR5GIF	TMR3GIF	TMR1GIF	176
PIR6	—	—	—	—	—	—	CCP2IF	CCP1IF	177
PIR7	SCANIF	CRCIF	NVMIF	—	—	—	—	CWG1IF	178
IPR0	—	—	TMR0IP	IOCIP	—	INT2IP	INT1IP	INT0IP	187
IPR1	OSCFIP	CSWIP	—	—	—	—	ADTIP	ADIP	188
IPR2	HLVDIP	ZCDIP	—	—	—	—	C2IP	C1IP	189
IPR3	RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP	190
IPR4	—	—	TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP	191
IPR5	—	—	—	—	—	TMR5GIP	TMR3GIP	TMR1GIP	192
IPR6	—	—	—	—	—	—	CCP2IP	CCP1IP	193
IPR7	SCANIP	CRCIP	NVMIP	—	—	—	—	CWG1IP	194

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for Interrupts.

REGISTER 15-5: WPUx: WEAK PULL-UP REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
WPUx7	WPUx6	WPUx5	WPUx4	WPUx3	WPUx2	WPUx1	WPUx0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown
-n/n = Value at POR and BOR/Value at all other Resets		

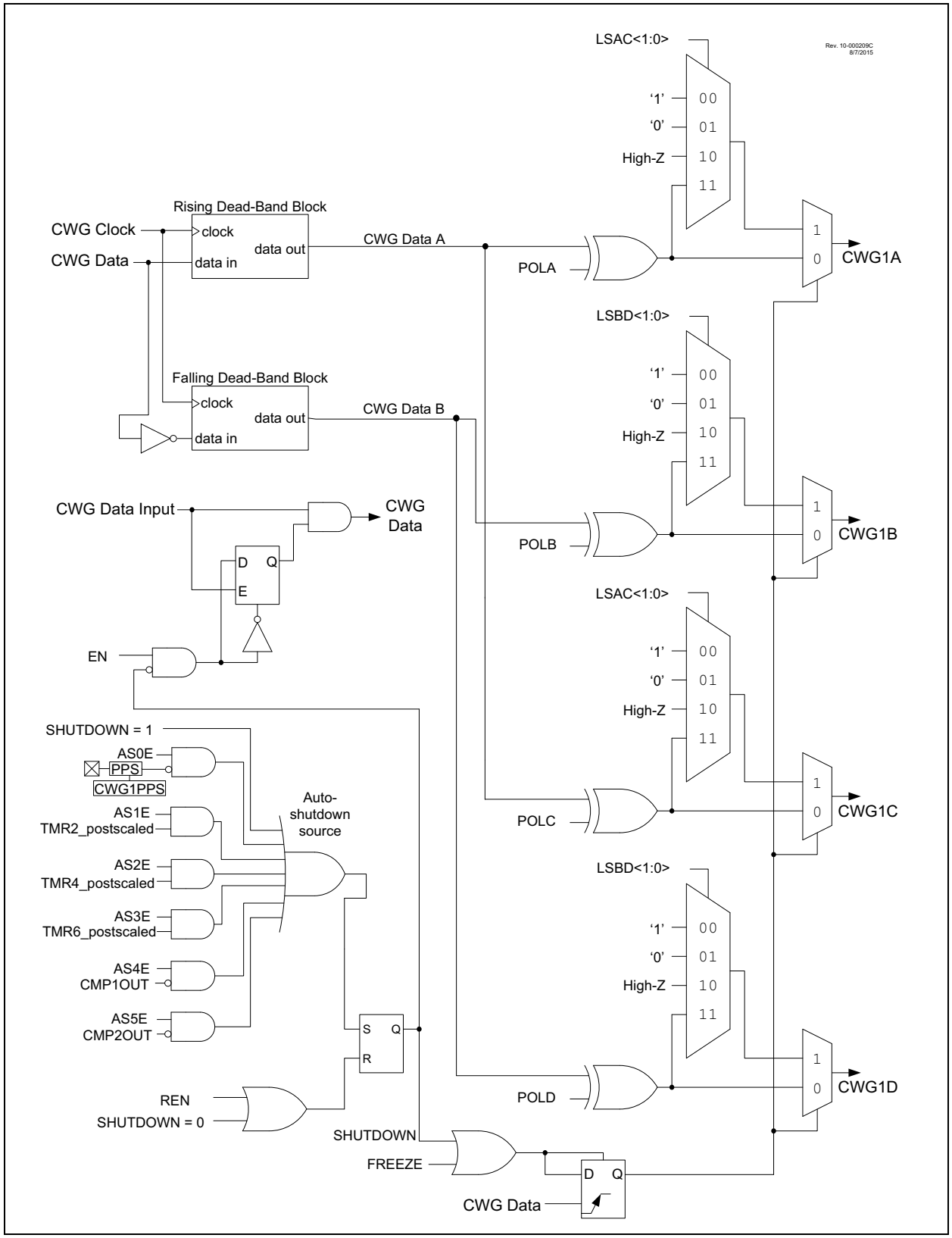
bit 7-0 **WPUx<7:0>:** Weak Pull-up PORTx Control bits
1 = Weak Pull-up enabled
0 = Weak Pull-up disabled

TABLE 15-6: WEAK PULL-UP PORT REGISTERS

Name	Device		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	28 Pins	40/44 Pins								
WPUA	X	X	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
WPUB	X	X	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
WPUC	X	X	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
WPUD	X		—	—	—	—	—	—	—	—
		X	WPUD7	WPUD6	WPUD5	WPUD4	WPUD3	WPUD2	WPUD1	WPUD0
WPUE	X		—	—	—	—	WPUE3(1)	—	—	—
		X	—	—	—	—	WPUE3(1)	WPUE2	WPUE1	WPUE0

Note 1: If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.

FIGURE 24-1: SIMPLIFIED CWG BLOCK DIAGRAM (HALF-BRIDGE MODE, MODE<2:0> = 100)



24.7 Rising Edge and Reverse Dead Band

In Half-Bridge mode, the rising edge dead band delays the turn-on of the CWG1A output after the rising edge of the CWG data input. In Full-Bridge mode, the reverse dead-band delay is only inserted when changing directions from Forward mode to Reverse mode, and only the modulated output CWG1B is affected.

The CWG1DBR register determines the duration of the dead-band interval on the rising edge of the input source signal. This duration is from 0 to 64 periods of the CWG clock.

Dead band is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWG1DBR register value is double-buffered. When EN = 0 (Register 24-1), the buffer is loaded when CWG1DBR is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input, after the LD bit (Register 24-1) is set. Refer to Figure 24-12 for an example.

24.8 Falling Edge and Forward Dead Band

In Half-Bridge mode, the falling edge dead band delays the turn-on of the CWG1B output at the falling edge of the CWG data input. In Full-Bridge mode, the forward dead-band delay is only inserted when changing directions from Reverse mode to Forward mode, and only the modulated output CWG1D is affected.

The CWG1DBF register determines the duration of the dead-band interval on the falling edge of the input source signal. This duration is from zero to 64 periods of CWG clock.

Dead-band delay is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWG1DBF register value is double-buffered. When EN = 0 (Register 24-1), the buffer is loaded when CWG1DBF is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input after the LD (Register 24-1) is set. Refer to Figure 24-13 for an example.

27.5.1.5 Synchronous Master Reception

Data is received at the RXx/DTx pin. The RXx/DTx pin output driver is automatically disabled when the EUSART is configured for synchronous master receive operation.

In Synchronous mode, reception is enabled by setting either the Single Receive Enable bit (SREN of the RCxSTA register) or the Continuous Receive Enable bit (CREN of the RCxSTA register).

When SREN is set and CREN is clear, only as many clock cycles are generated as there are data bits in a single character. The SREN bit is automatically cleared at the completion of one character. When CREN is set, clocks are continuously generated until CREN is cleared. If CREN is cleared in the middle of a character the CK clock stops immediately and the partial character is discarded. If SREN and CREN are both set, then SREN is cleared at the completion of the first character and CREN takes precedence.

To initiate reception, set either SREN or CREN. Data is sampled at the RXx/DTx pin on the trailing edge of the TX/CK clock pin and is shifted into the Receive Shift Register (RSR). When a complete character is received into the RSR, the RCxIF bit is set and the character is automatically transferred to the two character receive FIFO. The Least Significant eight bits of the top character in the receive FIFO are available in RCxREG. The RCxIF bit remains set as long as there are unread characters in the receive FIFO.

Note:	If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

27.5.1.6 Slave Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a slave receives the clock on the TX/CK line. The TXx/CKx pin output driver is automatically disabled when the device is configured for synchronous slave transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One data bit is transferred for each clock cycle. Only as many clock cycles should be received as there are data bits.

Note:	If the device is configured as a slave and the TX/CK function is on an analog pin, the corresponding ANSEL bit must be cleared.

27.5.1.7 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before RCxREG is read to access the FIFO. When this happens the OERR bit of the RCxSTA register is set. Previous data in the FIFO will not be overwritten. The two characters in the FIFO buffer can be read, however, no additional characters will be received until the error is cleared. The OERR bit can only be cleared by clearing the overrun condition. If the overrun error occurred when the SREN bit is set and CREN is clear then the error is cleared by reading RCxREG. If the overrun occurred when the CREN bit is set then the error condition is cleared by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

27.5.1.8 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCxSTA register is set the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCxSTA register is the ninth, and Most Significant, data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCxREG.

27.5.1.9 Synchronous Master Reception Setup:

1. Initialize the SPxBRGH:SPxBRGL register pair for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
4. Ensure bits CREN and SREN are clear.
5. If interrupts are desired, set the RCxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
6. If 9-bit reception is desired, set bit RX9.
7. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
8. Interrupt flag bit RCxIF will be set when reception of a character is complete. An interrupt will be generated if the enable bit RCxIE was set.
9. Read the RCxSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
10. Read the 8-bit received data by reading the RCxREG register.
11. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

TABLE 31-3: COMPUTATION MODES

		Bit Clear Conditions	Value after Trigger completion		Threshold Operations			Value at ADTIF interrupt		
Mode	ADMD	ADACC and ADCNT	ADACC	ADCNT	Retrigger	Threshold Test	Interrupt	ADAOV	ADFLTR	ADCNT
Basic	0	ADACLR = 1	Unchanged	Unchanged	No	Every Sample	If threshold=true	N/A	N/A	count
Accumulate	1	ADACLR = 1	S + ADACC or (S2-S1) + ADACC	If (ADCNT=FF): ADCNT, otherwise: ADCNT+1	No	Every Sample	If threshold=true	ADACC Overflow	ADACC/2^{ADCRS}	count
Average	2	ADACLR = 1 or ADCNT>=ADRPT at ADGO or retrigger	S + ADACC or (S2-S1) + ADACC	If (ADCNT=FF): ADCNT, otherwise: ADCNT+1	No	If ADCNT>= ADRPT	If threshold=true	ADACC Overflow	ADACC/2^{ADCRS}	count
Burst Average	3	ADACLR = 1 or ADGO set or retrigger	Each repetition: same as Average End with sum of all samples	Each repetition: same as Average End with ADCNT=ADRPT	Repeat while ADCNT<ADRPT	If ADCNT>= ADRPT	If threshold=true	ADACC Overflow	ADACC/2^{ADCRS}	ADRPT
Low-pass Filter	4	ADACLR = 1	S+ADACC-ADACC/2^{ADCRS} or (S2-S1)+ADACC-ADACC/2^{ADCRS}	If (ADCNT=FF): ADCNT, otherwise: ADCNT+1	No	If ADCNT>= ADRPT	If threshold=true	ADACC Overflow	Filtered Value	count

Note: S1 and S2 are abbreviations for Sample 1 and Sample 2, respectively. When ADDSEN = 0, S1 = ADRES; When ADDSEN = 1, S1 = ADPREV and S2 = ADRES.

TABLE 35-2: INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and CARRY bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

35.1.1 STANDARD INSTRUCTION SET

ADDLW **ADD literal to W**

Syntax: ADDLW k

Operands: $0 \le k \le 255$

Operation: $(W) + k \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDLW 15h`

Before Instruction
W = 10h
After Instruction
W = 25h

ADDWF **ADD W to f**

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

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

Operation: $(W) + (f) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

0010	01da	ffff	ffff

Description: Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDWF REG, 0, 0`

Before Instruction
W = 17h
REG = 0C2h
After Instruction
W = 0D9h
REG = 0C2h

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

ANDWF **AND W with f**

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

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

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

0001	01da	ffff	ffff

Description: The contents of W are AND'ed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: ANDWF REG, 0, 0

Before Instruction
W = 17h
REG = C2h
After Instruction
W = 02h
REG = C2h

BC **Branch if Carry**

Syntax: BC n

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

Operation: if CARRY bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0010	nnnn	nnnn

Description: If the CARRY bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

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

Example: HERE BC 5

Before Instruction
PC = address (HERE)
After Instruction
If CARRY = 1;
PC = address (HERE + 12)
If CARRY = 0;
PC = address (HERE + 2)

BRA **Unconditional Branch**

Syntax: BRA n

Operands: $-1024 \le n \le 1023$

Operation: $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1101	0nnn	nnnn	nnnn

Description: 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 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

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

Example: HERE BRA Jump

Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

BSF **Bit Set f**

Syntax: BSF f, b {,a}

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

Operation: $1 \rightarrow f<b>$

Status Affected: None

Encoding:

1000	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is set.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: BSF FLAG_REG, 7, 1

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

ADDWF **ADD W to Indexed (Indexed Literal Offset mode)**

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$

Operation: (W) + ((FSR2) + k) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0010	01d0	kkkk	kkkk

Description: The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'.
If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: ADDWF [OFST], 0

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

BSF **Bit Set Indexed (Indexed Literal Offset mode)**

Syntax: BSF [k], b

Operands: $0 \le f \le 95$
$0 \le b \le 7$

Operation: 1 → ((FSR2) + k)<b>

Status Affected: None

Encoding:

1000	bbb0	kkkk	kkkk

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: BSF [FLAG_OFST], 7

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h
After Instruction
Contents of 0A0Ah = D5h

SETF **Set Indexed (Indexed Literal Offset mode)**

Syntax: SETF [k]

Operands: $0 \le k \le 95$

Operation: FFh → ((FSR2) + k)

Status Affected: None

Encoding:

0110	1000	kkkk	kkkk

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: SETF [OFST]

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh

36.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

36.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

36.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

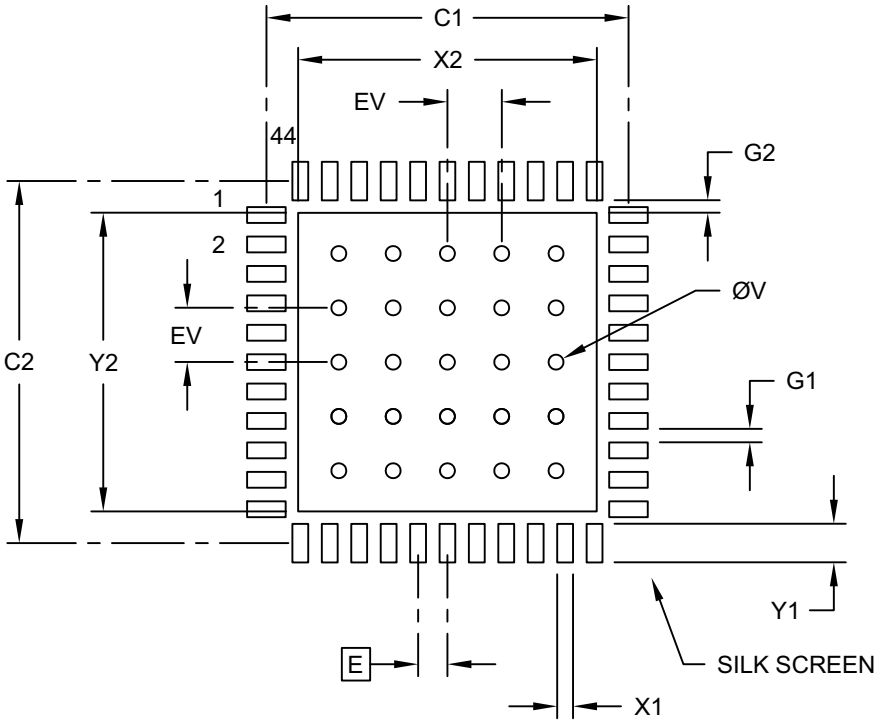
36.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Optional Center Pad Width	X2			6.60
Optional Center Pad Length	Y2			6.60
Contact Pad Spacing	C1		8.00	
Contact Pad Spacing	C2		8.00	
Contact Pad Width (X44)	X1			0.35
Contact Pad Length (X44)	Y1			0.85
Contact Pad to Contact Pad (X40)	G1	0.30		
Contact Pad to Center Pad (X44)	G2	0.28		
Thermal Via Diameter	V		0.33	
Thermal Via Pitch	EV		1.20	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing No. C04-2103C