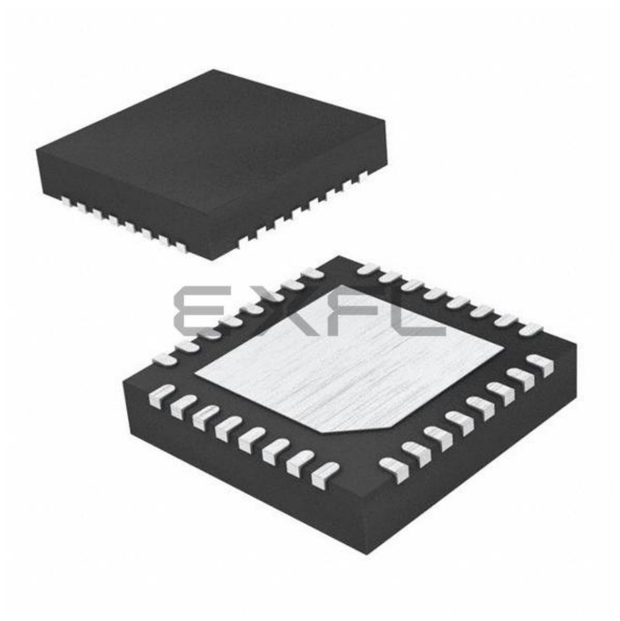


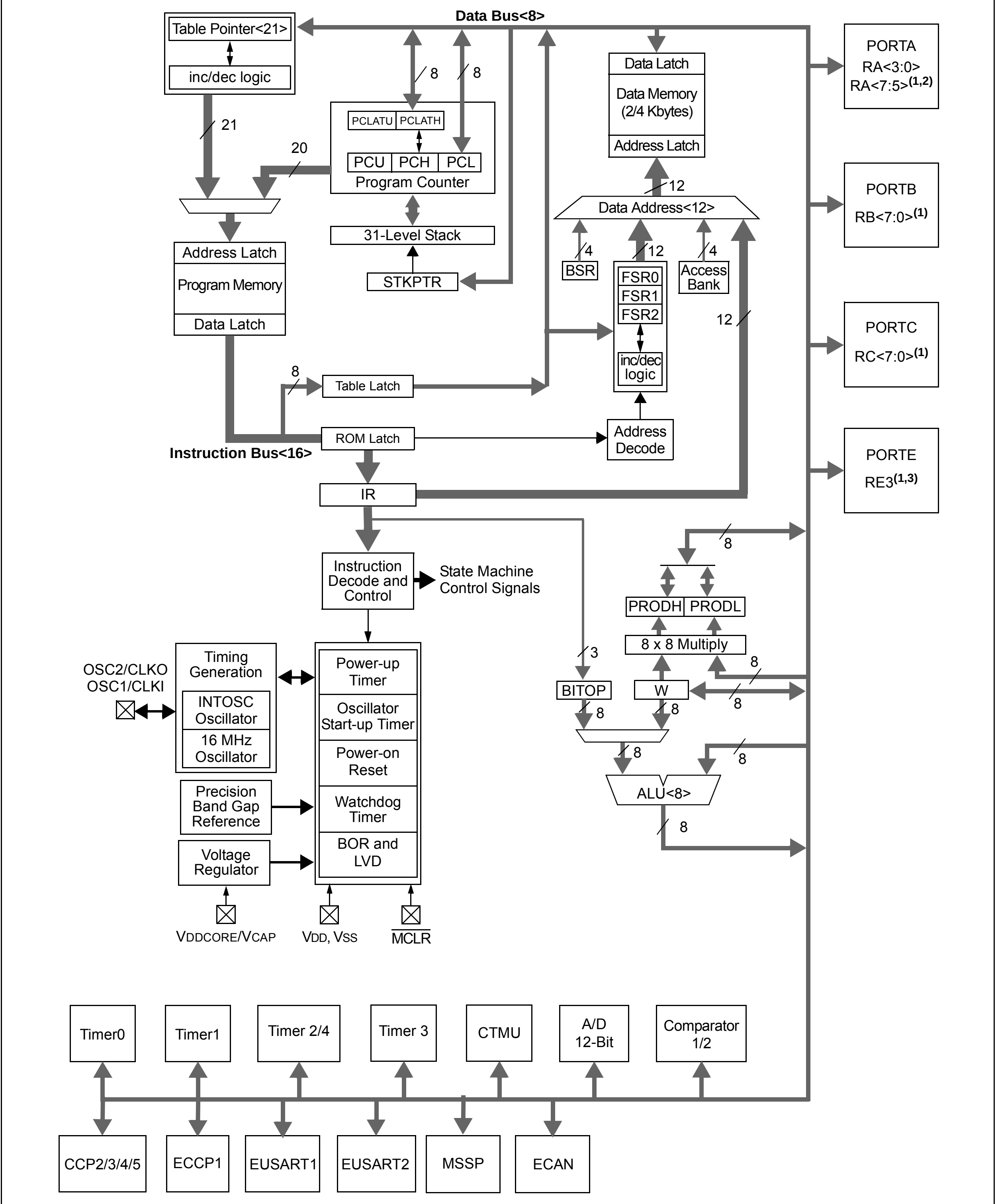
What is "Embedded - Microcontrollers"?

"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	ECANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	24
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 8x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN-S (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k80-i-mm

FIGURE 1-1: PIC18F2XK80 (28-PIN) BLOCK DIAGRAM



Note 1: See Table 1-4 for I/O port pin descriptions.

2: RA6 and RA7 are only available as digital I/O in select oscillator modes. For more information, see **Section 3.0 “Oscillator Configurations”**.

3: RE3 is only available when the MCLRE Configuration bit is cleared (MCLRE = 0).

REGISTER 4-2: PMD1: PERIPHERAL MODULE DISABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPMD[1]	CTMUMD	ADCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
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	**PSPMD:** Peripheral Module Disable bit[1] 1 = The PSP module is disabled; all PSP registers are held in Reset and are not writable 0 = The PSP module is enabled
bit 6	**CTMUMD:** PMD CTMU Disable bit 1 = The CTMU module is disabled; all CTMU registers are held in Reset and are not writable 0 = The CTMU module is enabled
bit 5	**ADCMD:** A/D Module Disable bit 1 = The A/D module is disabled; all A/D registers are held in Reset and are not writable 0 = The A/D module is enabled
bit 4	**TMR4MD:** TMR4MD Disable bit 1 = The Timer4 module is disabled; all Timer4 registers are held in Reset and are not writable 0 = The Timer4 module is enabled
bit 3	**TMR3MD:** TMR3MD Disable bit 1 = The Timer3 module is disabled; all Timer3 registers are held in Reset and are not writable 0 = The Timer3 module is enabled
bit 2	**TMR2MD:** TMR2MD Disable bit 1 = The Timer2 module is disabled; all Timer2 registers are held in Reset and are not writable 0 = The Timer2 module is enabled
bit 1	**TMR1MD:** TMR1MD Disable bit 1 = The Timer1 module is disabled; all Timer1 registers are held in Reset and are not writable 0 = The Timer1 module is enabled
bit 0	**TMR0MD:** Timer0 Module Disable bit 1 = The Timer0 module is disabled; all Timer0 registers are held in Reset and are not writable 0 = The Timer0 module is enabled

Note 1: This bit is unimplemented on 28-pin devices (PIC18F2XK80, PIC18LF2XK80).

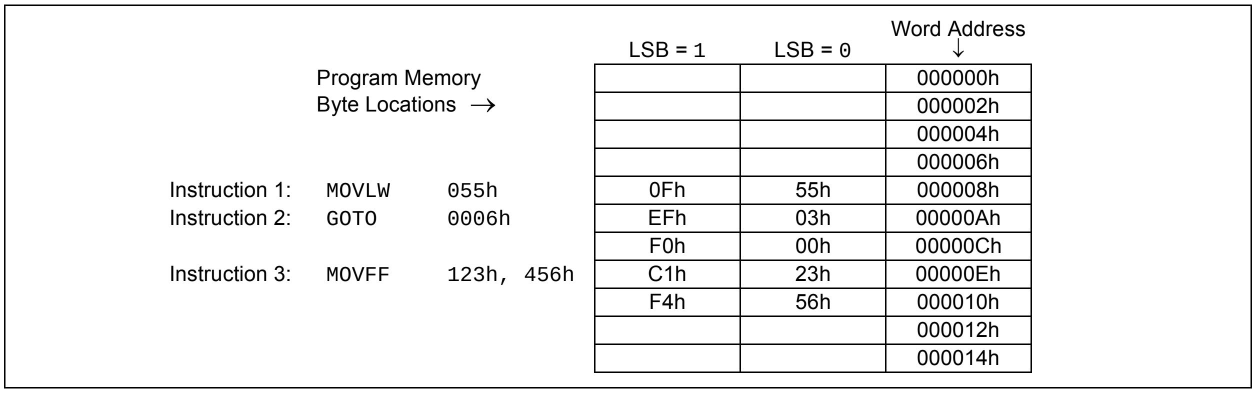
6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two or four bytes in program memory. The Least Significant Byte (LSB) of an instruction word is always stored in a program memory location with an even address (LSB = `0`). To maintain alignment with instruction boundaries, the PC increments in steps of two and the LSB will always read '`0`' (see **Section 6.1.2 "Program Counter"**).

Figure 6-5 shows an example of how instruction words are stored in the program memory.

The `CALL` and `GOTO` instructions have the 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 6-5 shows how the instruction, `GOTO 0006h`, is encoded in the program memory. Program branch instructions, which 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 that the PC will be offset by. For more details on the instruction set, see **Section 29.0 "Instruction Set Summary"**.

FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY



Program Memory Byte Locations →

			LSB = 1	LSB = 0	Word Address ↓
					000000h
					000002h
					000004h
					000006h
Instruction 1:	`MOVLW`	`055h`	0Fh	55h	000008h
Instruction 2:	`GOTO`	`0006h`	EFh	03h	00000Ah
			F0h	00h	00000Ch
Instruction 3:	`MOVFF`	`123h, 456h`	C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

6.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four, two-word instructions: `CALL`, `MOVFF`, `GOTO` and `LSFR`. In all cases, the second word of the instructions always has '`1111`' as its four Most Significant bits (MSbs). The other 12 bits are literal data, usually a data memory address.

The use of '`1111`' in the 4 MSbs of an instruction specifies a special form of `NOP`. If the instruction is executed in proper sequence, immediately after the first word, the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a `NOP` is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 6-4 shows how this works.

Note:	For information on two-word instructions in the extended instruction set, see **Section 6.5 "Program Memory and the Extended Instruction Set"**.

EXAMPLE 6-4: 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, skip this word`
`1111 0100 0101 0110`	`                      ; Execute this word as a NOP`
`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, execute this word`
`1111 0100 0101 0110`	`                      ; 2nd word of instruction`
`0010 0100 0000 0000`	`ADDWF     REG3        ; continue code`

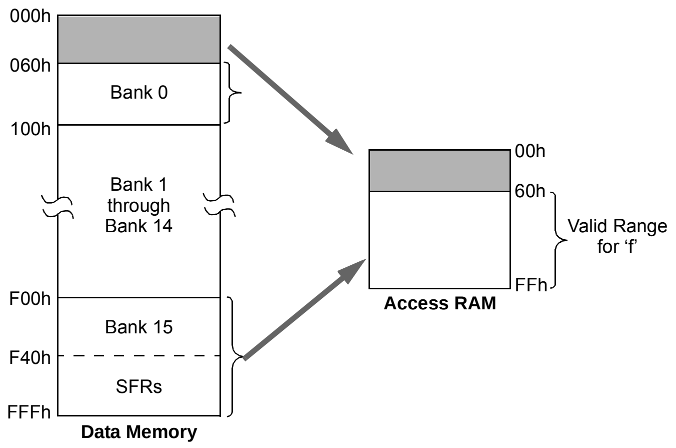
FIGURE 6-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

EXAMPLE INSTRUCTION: `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

When a = `0` and f ≥ 60h:

The instruction executes in Direct Forced mode. ‘f’ is interpreted as a location in the Access RAM between 060h and FFFh. This is the same as locations, F60h to FFFh, (Bank 15) of data memory.

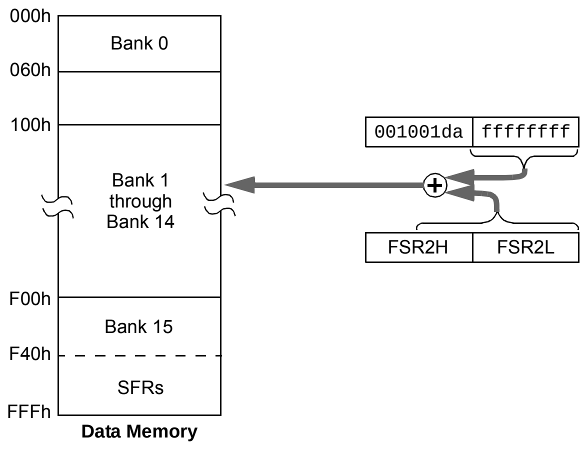
Locations below 060h are not available in this addressing mode.



When a = `0` and f ≤ 5Fh:

The instruction executes in Indexed Literal Offset mode. ‘f’ is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where ‘k’ is the same as ‘f’.



When a = `1` (all values of f):

The instruction executes in Direct mode (also known as Direct Long mode). ‘f’ is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.

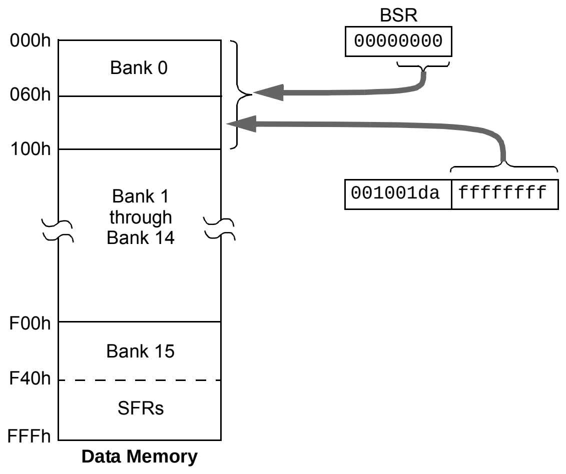


TABLE 11-7: PORTD FUNCTIONS (CONTINUED)

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RD7/RX2/DT2/ P1D/PSP7	RD7	0	O	DIG	LATD<7> data output.
		1	I	ST	PORTD<7> data input.
	RX2[(1)]	1	I	ST	Asynchronous serial receive data input (EUSARTx module).
	DT2[(1)]	1	O	DIG	Synchronous serial data output (EUSARTx module); takes priority over port data.
		1	I	ST	Synchronous serial data input (EUSARTx module); user must configure as an input.
	P1D	0	O	DIG	ECCP1 Enhanced PWM output, Channel D. May be configured for tri-state during Enhanced PWM.
	PSP7	x	I/O	ST	Parallel Slave Port data.

Legend: O = Output; I = Input; ANA = Analog Signal; DIG = CMOS Output; ST = Schmitt Trigger Buffer Input; x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

Note 1: This is the pin assignment for 40 and 44-pin devices (PIC18F4XK80).

TABLE 11-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

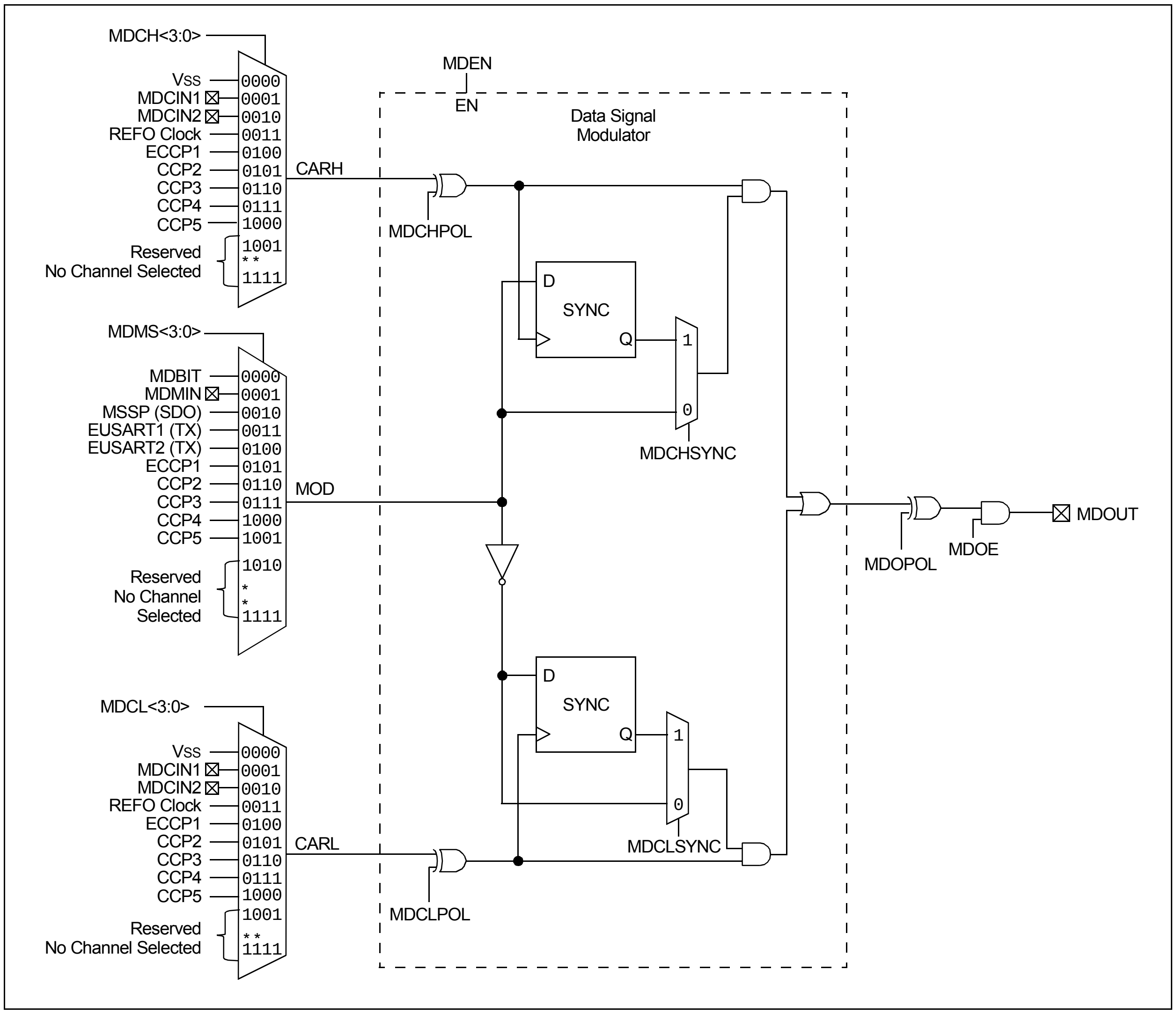
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
LATD	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
PADCFG1	RDPU[(1)]	REPU[(1)]	RFPU[(2)]	RGPU[(2)]	—	—	—	CTMUDS
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD
ANCON1	—	ANSEL14	ANSEL13	ANSEL12	ANSEL11	ANSEL10	ANSEL9	ANSEL8

Legend: Shaded cells are not used by PORTD.

Note 1: These bits are unimplemented on 28-pin devices, read as '0'.

2: These bits are unimplemented on 28/40/44-pin devices, read as '0'.

FIGURE 12-1: SIMPLIFIED BLOCK DIAGRAM OF THE DATA SIGNAL MODULATOR



REGISTER 12-1: MDCON: MODULATION CONTROL REGISTER

R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	U-0	U-0	R/W-0
MDEN	MDOE	MDSLR	MDOPOL	MDO	—	—	MDBIT
bit 7							bit 0

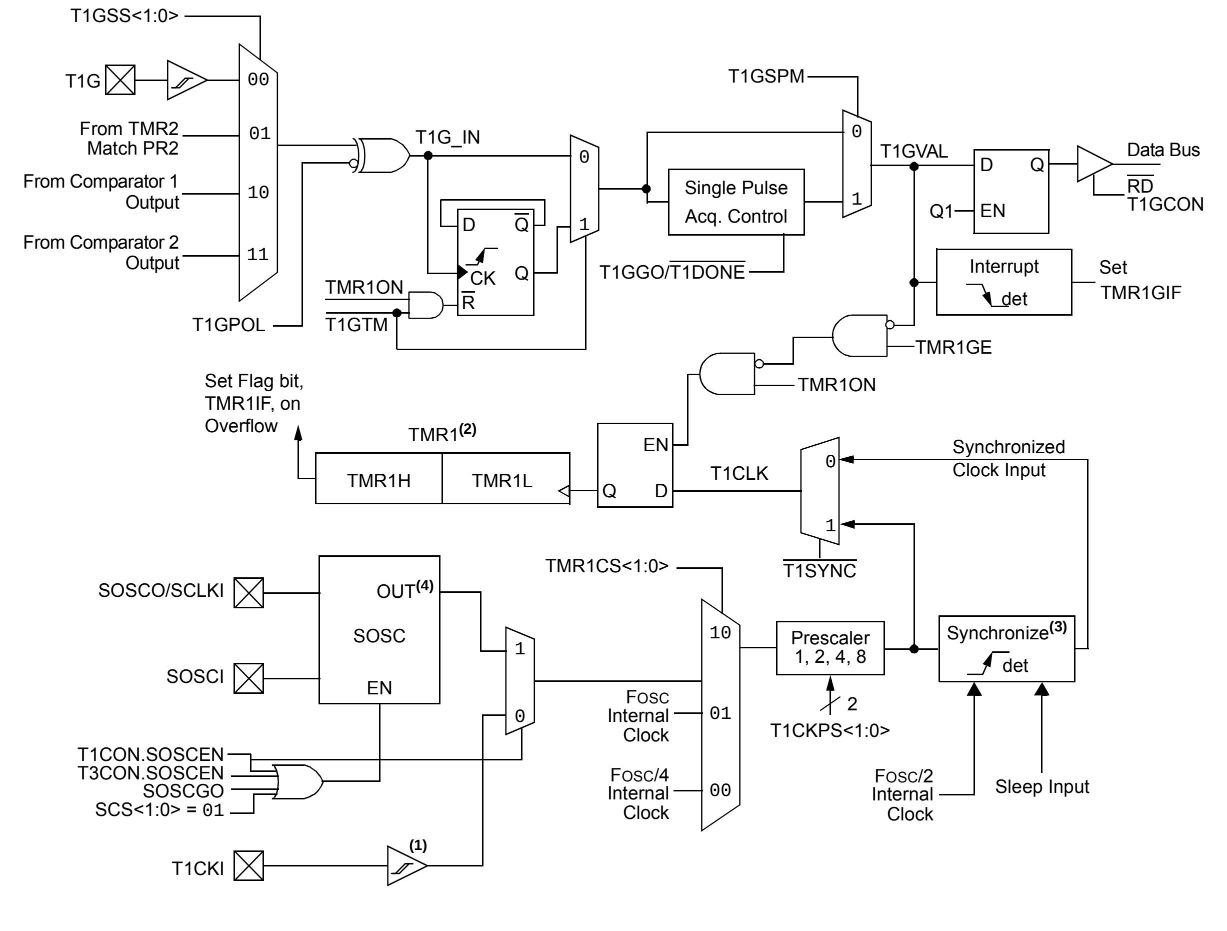
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7	**MDEN:** Modulator Module Enable bit 1 = Modulator module is enabled and mixing input signals 0 = Modulator module is disabled and has no output
bit 6	**MDOE:** Modulator Module Pin Output Enable bit 1 = Modulator pin output is enabled 0 = Modulator pin output is disabled
bit 5	**MDSLR:** MDOUT Pin Slew Rate Limiting bit 1 = MDOUT pin slew rate limiting is enabled 0 = MDOUT pin slew rate limiting is disabled
bit 4	**MDOPOL:** Modulator Output Polarity Select bit 1 = Modulator output signal is inverted 0 = Modulator output signal is not inverted
bit 3	**MDO:** Modulator Output bit Displays the current output value of the modulator module.(2)
bit 2-1	**Unimplemented:** Read as ‘0’
bit 0	**MDBIT:** Modulator Source Input bit Allows software to manually set modulation source input to module.(1)

Note 1: The MDBIT must be selected as the modulation source in the MDSRC register for this operation.

2: The modulated output frequency can be greater and asynchronous from the clock that updates this register bit. The bit value may not be valid for higher speed modulator or carrier signals.

FIGURE 14-1: TIMER1 BLOCK DIAGRAM



Note 1: ST Buffer is high-speed type when using T1CKI.
2: Timer1 register increments on rising edge.
3: Synchronize does not operate while in Sleep.
4: The output of SOSC is determined by the SOSCSEL<1:0> Configuration bits.

16.5.2 TIMER3 GATE SOURCE SELECTION

The Timer3 gate source can be selected from one of four different sources. Source selection is controlled by the T3GSS<1:0> bits (T3GCON<1:0>). The polarity for each available source is also selectable and is controlled by the T3GPOL bit (T3GCON<6>).

TABLE 16-2: TIMER3 GATE SOURCES

T3GSS<1:0>	Timer3 Gate Source
00	Timerx Gate Pin
01	TMR4 to Match PR4 (TMR4 increments to match PR4)
10	Comparator 1 Output (comparator logic high output)
11	Comparator 2 Output (comparator logic high output)

16.5.2.1 T3G Pin Gate Operation

The T3G pin is one source for Timer3 gate control. It can be used to supply an external source to the Timerx gate circuitry.

16.5.2.2 Timer4 Match Gate Operation

The TMR4 register will increment until it matches the value in the PR4 register. On the very next increment cycle, TMR4 will be reset to 00h. When this Reset occurs, a low-to-high pulse will automatically be generated and internally supplied to the Timerx gate circuitry. The pulse will remain high for one instruction cycle and will return back to a low state until the next match.

Depending on T3GPOL, Timerx increments differently when TMR4 matches PR4. When T3GPOL = `1`, Timer3 increments for a single instruction cycle following a TMR4 match with PR4. When T3GPOL = `0`, Timer3 increments continuously, except for the cycle following the match, when the gate signal goes from low-to-high.

16.5.2.3 Comparator 1 Output Gate Operation

The output of Comparator 1 can be internally supplied to the Timer3 gate circuitry. After setting up Comparator 1 with the CM1CON register, Timer3 will increment depending on the transitions of the CMP1OUT (CMSTAT<6>) bit.

16.5.2.4 Comparator 2 Output Gate Operation

The output of Comparator 2 can be internally supplied to the Timer3 gate circuitry. After setting up Comparator 2 with the CM2CON register, Timer3 will increment depending on the transitions of the CMP2OUT (CMSTAT<7>) bit.

16.5.3 TIMER3 GATE TOGGLE MODE

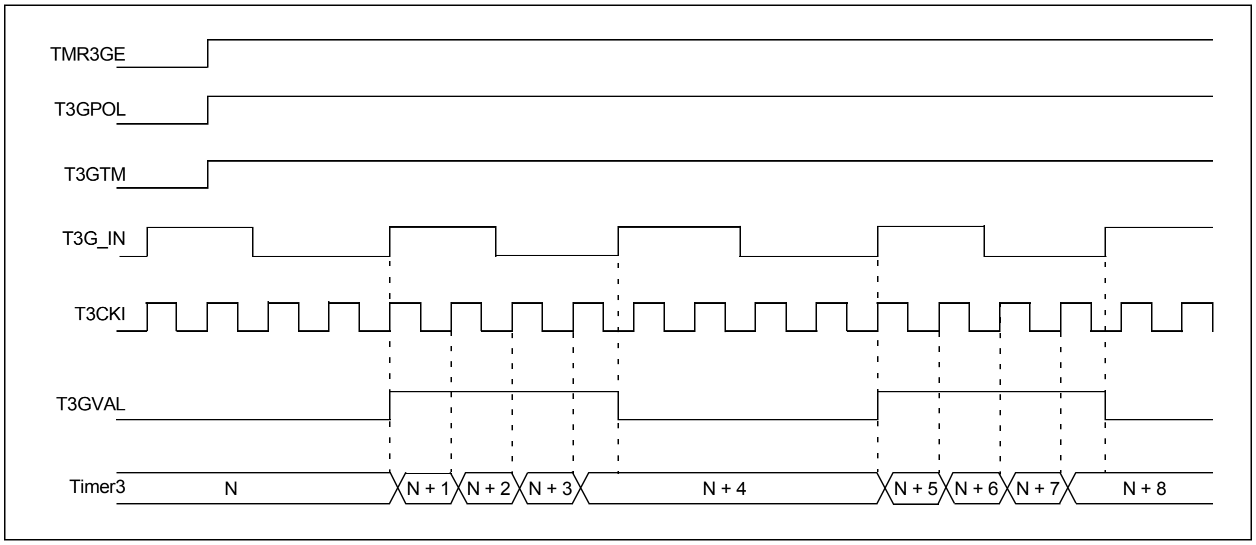
When Timer3 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer3 gate signal, as opposed to the duration of a single level pulse.

The Timer3 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. (For timing details, see Figure 16-3.)

The T3GVAL bit will indicate when the Toggled mode is active and the timer is counting.

Timer3 Gate Toggle mode is enabled by setting the T3GTM bit (T3GCON<5>). When the T3GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

FIGURE 16-3: TIMER3 GATE TOGGLE MODE



20.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

PIC18F66K80 family devices have one Enhanced Capture/Compare/PWM (ECCP) module: ECCP1. These modules contain a 16-bit register, which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register. These ECCP modules are upward compatible with CCP

ECCP1 is implemented as standard CCP modules with enhanced PWM capabilities. These include:

- Provision for two or four output channels
- Output Steering modes
- Programmable polarity
- Programmable dead-band control
- Automatic shutdown and restart

The enhanced features are discussed in detail in **Section 20.4 "PWM (Enhanced Mode)"**.

The ECCP1 module uses the control register, CCP1CON. The control registers, CCP2CON through CCP5CON, are for the modules, CCP2 through CCP5.

20.4 PWM (Enhanced Mode)

The Enhanced PWM mode can generate a PWM signal on up to four different output pins with up to 10 bits of resolution. It can do this through four different PWM Output modes:

- Single PWM
- Half-Bridge PWM
- Full-Bridge PWM, Forward mode
- Full-Bridge PWM, Reverse mode

To select an Enhanced PWM mode, the P1M bits of the CCP1CON register must be set appropriately.

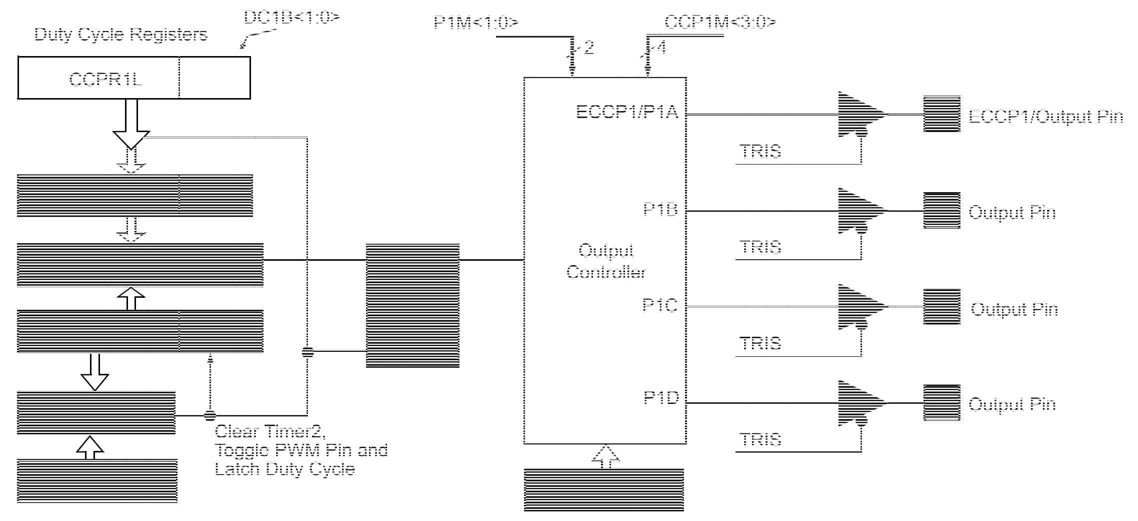
The PWM outputs are multiplexed with I/O pins and are designated: P1A, P1B, P1C and P1D. The polarity of the PWM pins is configurable and is selected by setting the CCP1M bits in the CCP1CON register appropriately.

Table 20-1 provides the pin assignments for each Enhanced PWM mode.

Figure 20-3 provides an example of a simplified block diagram of the Enhanced PWM module.

Note:	To prevent the generation of an incomplete waveform when the PWM is first enabled, the ECCP module waits until the start of a new PWM period before generating a PWM signal.

FIGURE 20-3: EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE



Note 1: The 8-bit TMR2 register is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler, to create the 10-bit time base.

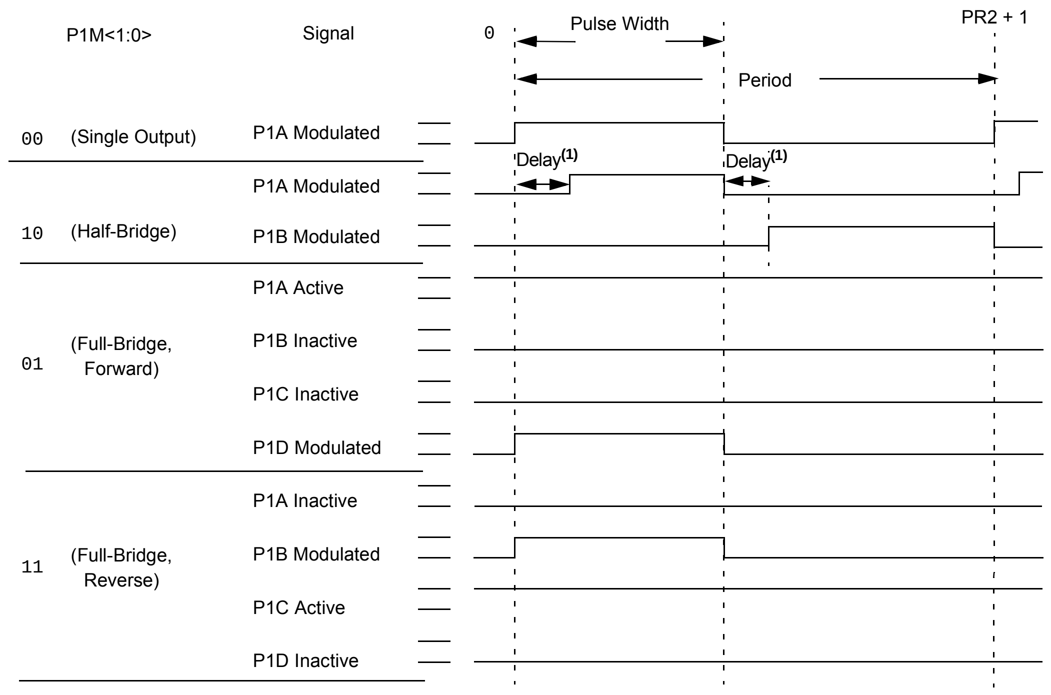
Note 1:	The TRIS register value for each PWM output must be configured appropriately.
2:	Any pin not used by an Enhanced PWM mode is available for alternate pin functions.

TABLE 20-2: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES

ECCP Mode	P1M<1:0>	P1A	P1B	P1C	P1D
Single	00	Yes[1]	Yes[1]	Yes[1]	Yes[1]
Half-Bridge	10	Yes	Yes	No	No
Full-Bridge, Forward	01	Yes	Yes	Yes	Yes
Full-Bridge, Reverse	11	Yes	Yes	Yes	Yes

Note 1: Outputs are enabled by pulse steering in Single mode (see Register 20-5).

FIGURE 20-4: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)



Relationships:
- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Pulse Width = TOSC * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * (ECCP1DEL<6:0>)

Note 1: Dead-band delay is programmed using the ECCP1DEL register (**Section 20.4.6 “Programmable Dead-Band Delay Mode”**).

21.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

REGISTER 21-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE[(1)]	$D/\overline{A}$	P	S	$R/\overline{W}$	UA	BF
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	**SMP:** Sample bit SPI Master mode: 1 = Input data is sampled at the end of data output time 0 = Input data is sampled at the middle of data output time SPI Slave mode: SMP must be cleared when SPI is used in Slave mode.
bit 6	**CKE:** SPI Clock Select bit[(1)] 1 = Transmit occurs on transition from active to Idle clock state 0 = Transmit occurs on transition from Idle to active clock state
bit 5	**$D/\overline{A}$:** Data/$\overline{\text{Address}}$ bit Used in I^{2}C™ mode only.
bit 4	**P:** Stop bit Used in I^{2}C mode only. This bit is cleared when the MSSP module is disabled; SSPEN is cleared.
bit 3	**S:** Start bit Used in I^{2}C mode only.
bit 2	**$R/\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit Used in I^{2}C mode only.
bit 1	**UA:** Update Address bit Used in I^{2}C mode only.
bit 0	**BF:** Buffer Full Status bit (Receive mode only) 1 = Receive is complete, SSPBUF is full 0 = Receive is not complete, SSPBUF is empty

Note 1: Polarity of clock state is set by the CKP bit (SSPCON1<4>).

25.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 32-tap resistor ladder network that provides a selectable reference voltage. Although its primary purpose is to provide a reference for the analog comparators, it may also be used independently of them.

A block diagram of the module is shown in Figure 25-1. The resistor ladder is segmented to provide a range of CVREF values and has a power-down function to conserve power when the reference is not being used. The module's supply reference can be provided from either device VDD/VSS or an external voltage reference.

25.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register (Register 25-1). The comparator voltage reference provides a range of output voltage with 32 levels.

The CVR<4:0> selection bits (CVRCON<4:0>) offer a range of output voltages. Equation 25-1 shows the how the comparator voltage reference is computed.

EQUATION 25-1:

If CVRSS = 1:

$$CV_{REF} = \left(V_{REF-} + \frac{CVR<4:0>}{32}\right) \cdot (V_{REF+} - V_{REF-})$$

If CVRSS = 0:

$$CV_{REF} = \left(AV_{SS} + \frac{CVR<4:0>}{32}\right) \cdot (AV_{DD} - AV_{SS})$$

The comparator reference supply voltage can come from either VDD and VSS, or the external VREF+ and VREF- that are multiplexed with RA3 and RA2. The voltage source is selected by the CVRSS bit (CVRCON<5>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see Table 31-2 in **Section 31.0 "Electrical Characteristics"**).

REGISTER 25-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE	CVRSS	CVR4	CVR3	CVR2	CVR1	CVR0
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

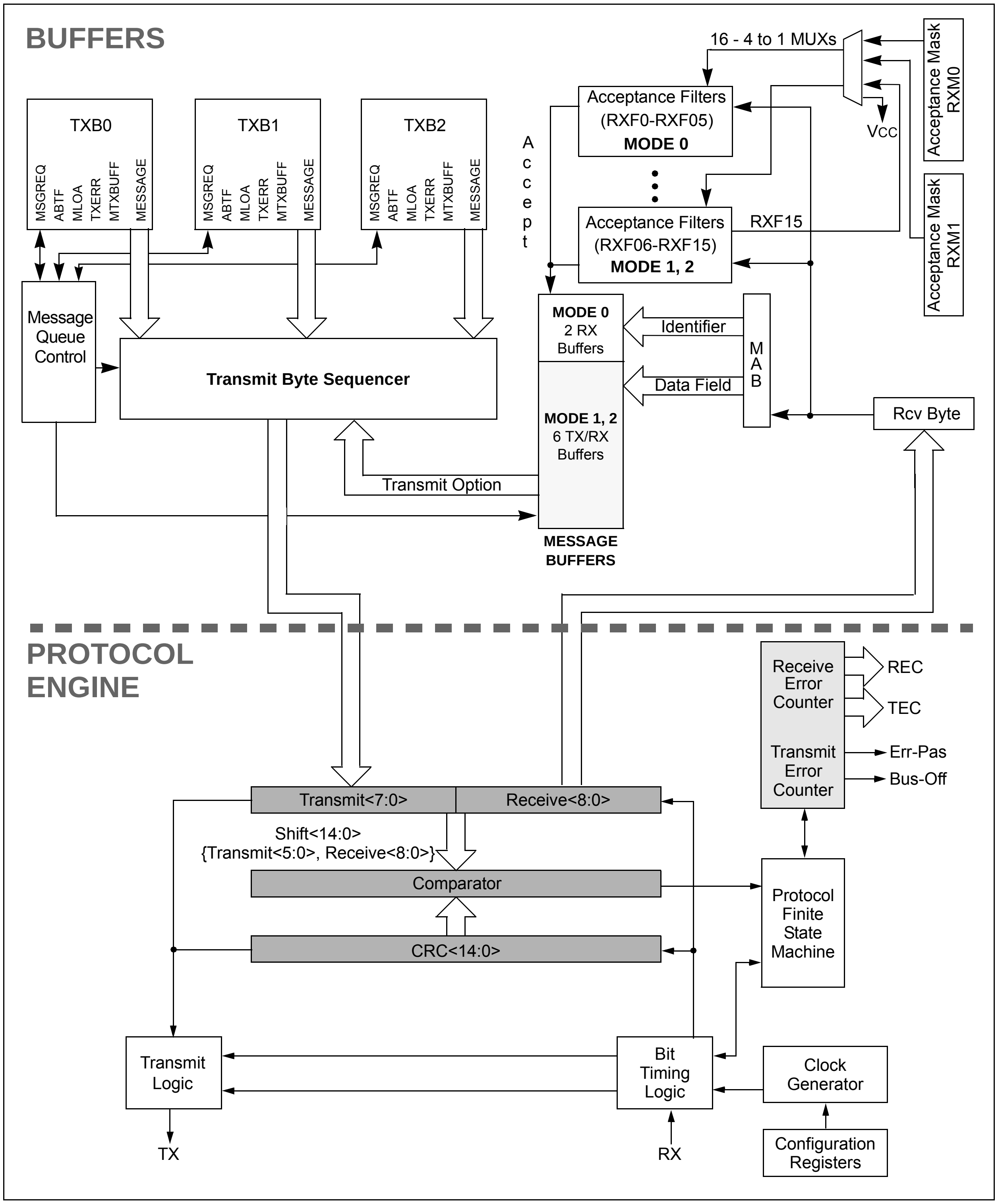
bit 7 **CVREN:** Comparator Voltage Reference Enable bit
1 = CVREF circuit powered on
0 = CVREF circuit powered down

bit 6 **CVROE:** Comparator VREF Output Enable bit
1 = CVREF voltage level is output on CVREF pin
0 = CVREF voltage level is disconnected from CVREF pin

bit 5 **CVRSS:** Comparator VREF Source Selection bit
1 = Comparator reference source, CVRSRC = VREF+ – VREF-
0 = Comparator reference source, CVRSRC = AVDD – AVSS

bit 4-0 **CVR<4:0>:** Comparator VREF Value Selection 0 ≤ CVR<4:0> ≤ 31 bits
When CVRSS = 1:
CVREF = (VREF-) + (CVR<4:0>/32) • (VREF+ – VREF-)
When CVRSS = 0:
CVREF = (AVSS) + (CVR<4:0>/32) • (AVDD – AVSS)

FIGURE 27-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM



REGISTER 27-49: MSEL1: MASK SELECT REGISTER 1[1]

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
FIL7_1	FIL7_0	FIL6_1	FIL6_0	FIL5_1	FIL5_0	FIL4_1	FIL4_0
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 7-6 **FIL7_<1:0>:** Filter 7 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL6_<1:0>:** Filter 6 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL5_<1:0>:** Filter 5 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL4_<1:0>:** Filter 4 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

REGISTER 28-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

R/P-0	R/P-0	U-0	U-0	R/P-1	R/P-0	R/P-0	R/P-0
IESO	FCMEN	—	PLLCFG[1]	FOSC3[2]	FOSC2[2]	FOSC1[2]	FOSC0[2]
bit 7							bit 0

Legend:	P = Programmable bit		
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 **IESO:** Internal/External Oscillator Switchover bit
1 = Two-Speed Start-up is enabled
0 = Two-Speed Start-up is disabled

bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor is enabled
0 = Fail-Safe Clock Monitor is disabled

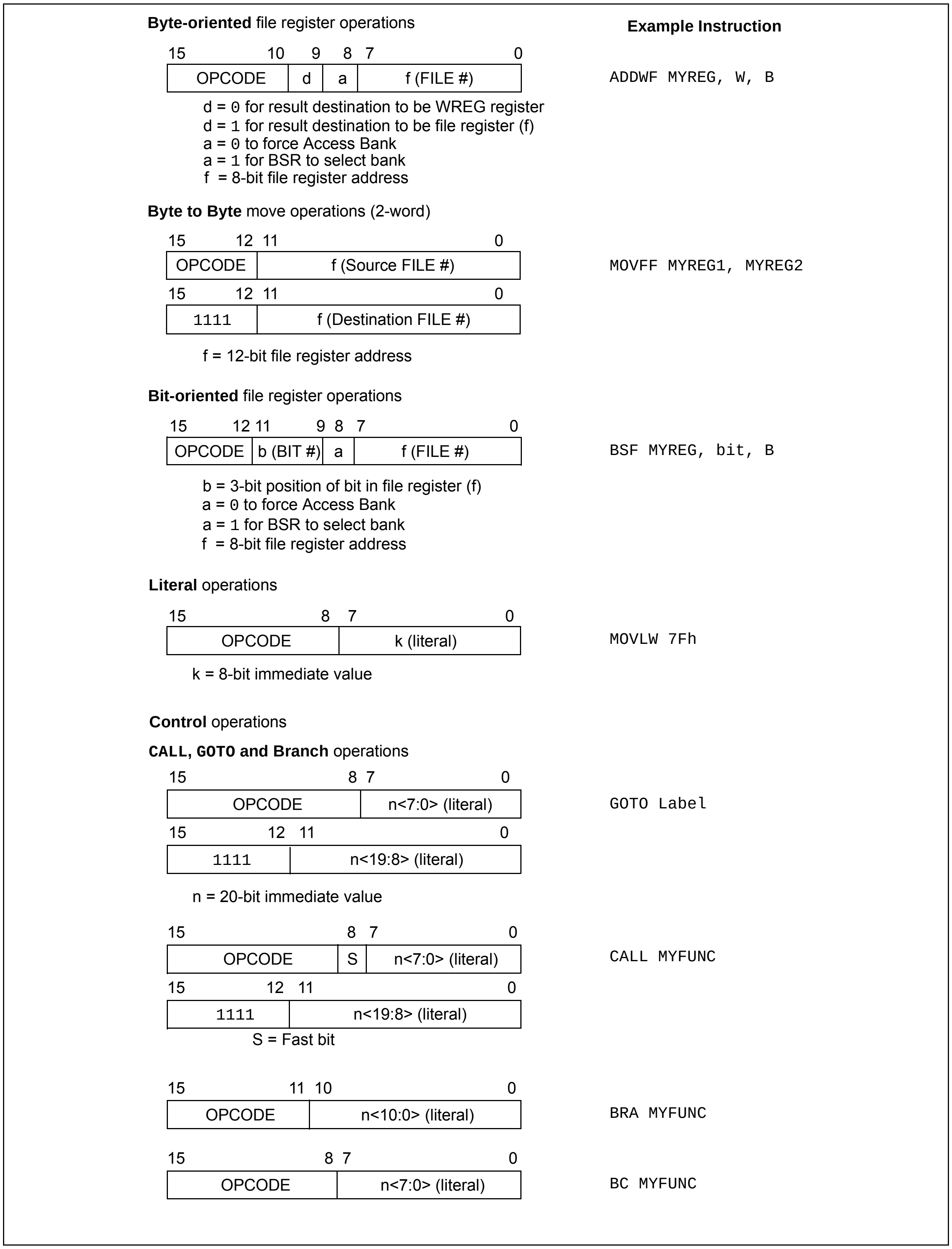
bit 5 **Unimplemented:** Read as ‘0’

bit 4 **PLLCFG:** 4X PLL Enable bit[1]
1 = Oscillator is multiplied by 4
0 = Oscillator is used directly

bit 3-0 **FOSC<3:0>:** Oscillator Selection bits[2]
1101 = EC1, EC oscillator **(low power, DC-160 kHz)**
1100 = EC1IO, EC oscillator with CLKOUT function on RA6 **(low power, DC-160 kHz)**
1011 = EC2, EC oscillator **(medium power, 160 kHz-16 MHz)**
1010 = EC2IO, EC oscillator with CLKOUT function on RA6 **(medium power, 160 kHz-16 MHz)**
0101 = EC3, EC oscillator **(high power, 16 MHz-64 MHz)**
0100 = EC3IO, EC oscillator with CLKOUT function on RA6 **(high power, 16 MHz-64 MHz)**
0011 = HS1, HS oscillator **(medium power, 4 MHz-16 MHz)**
0010 = HS2, HS oscillator **(high power, 16 MHz-25 MHz)**
0001 = XT oscillator
0000 = LP oscillator
0111 = RC, external RC oscillator
0110 = RCIO, external RC oscillator with CKLOUT function on RA6
1000 = INTIO2, internal RC oscillator
1001 = INTIO1, internal RC oscillator with CLKOUT function on RA6

Note 1: Not valid for the INTIOx PLL mode.
2: INTIO + PLL can be enabled only by the PLLEN bit (OSCTUNE<6>). Other PLL modes can be enabled by either the PLLEN bit or the PLLCFG (CONFIG1H<4>) bit.

FIGURE 29-1: GENERAL FORMAT FOR INSTRUCTIONS


Byte-oriented file register operations
Example Instruction
15 10 9 8 7 0
OPCODE | d | a | f (FILE #)
ADDWF MYREG, W, B
d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address
Byte to Byte move operations (2-word)
15 12 11 0
OPCODE | f (Source FILE #)
MOVFF MYREG1, MYREG2
15 12 11 0
1111 | f (Destination FILE #)
f = 12-bit file register address
Bit-oriented file register operations
15 12 11 9 8 7 0
OPCODE | b (BIT #) | a | f (FILE #)
BSF MYREG, bit, B
b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address
Literal operations
15 8 7 0
OPCODE | k (literal)
MOVLW 7Fh
k = 8-bit immediate value
Control operations
CALL, GOTO and Branch operations
15 8 7 0
OPCODE | n<7:0> (literal)
GOTO Label
15 12 11 0
1111 | n<19:8> (literal)
n = 20-bit immediate value
15 8 7 0
OPCODE | S | n<7:0> (literal)
CALL MYFUNC
15 12 11 0
1111 | n<19:8> (literal)
S = Fast bit
15 11 10 0
OPCODE | n<10:0> (literal)
BRA MYFUNC
15 8 7 0
OPCODE | n<7:0> (literal)
BC MYFUNC

XORWF	**Exclusive OR W with f**
Syntax:	XORWF f {,d {,a}}
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) .XOR. (f) $\rightarrow$ dest
Status Affected:	N, Z
Encoding:	`0001` `10da` `ffff` `ffff`
Description:	Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the 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 $f \leq 95$ (5Fh). See **Section 29.2.3 "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: `XORWF REG, 1, 0`

Before Instruction

```
REG   =   AFh
W     =   B5h
```

After Instruction

```
REG   =   1Ah
W     =   B5h
```

29.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

Note:	Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing (**Section 6.6.1 "Indexed Addressing with Literal Offset"**). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank (a = `0`) or in a GPR bank designated by the BSR (a = `1`). When the extended instruction set is enabled and a = `0`, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward-compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see **Section 29.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**).

Although the Indexed Literal Offset mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

29.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument 'f' in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within the brackets, will generate an error in the MPASM™ Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '`0`'. This is in contrast to standard operation (extended instruction set disabled), when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM Assembler.

The destination argument 'd' functions as before.

In the latest versions of the MPASM Assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, `/y`, or the PE directive in the source listing.

29.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F66K80 family, it is very important to consider the type of code. A large, re-entrant application that is written in C and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.