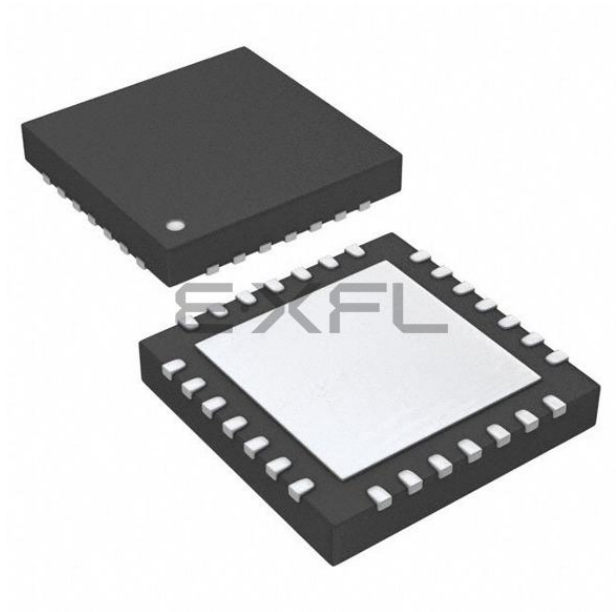


Welcome to E-XFL.COM

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	25
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)	1.8V ~ 3.6V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-UFQFN Exposed Pad
Supplier Device Package	28-UQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf25k40-e-mv

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F24K40
- PIC18LF24K40
- PIC18F25K40
- PIC18LF25K40

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Program Flash Memory. In addition to these features, the PIC18(L)F2x/4xK40 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 XLP TECHNOLOGY

All of the devices in the PIC18(L)F2x/4xK40 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the secondary oscillator or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Peripheral Module Disable:** Modules that are not being used in the code can be selectively disabled using the PMD module. This further reduces the power consumption.

1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18(L)F2x/4xK40 family offer several different oscillator options. The PIC18(L)F2x/4xK40 family can be clocked from several different sources:

- HFINTOSC
  - 1-64 MHz precision digitally controlled internal oscillator
- LFINTOSC
  - 31 kHz internal oscillator
- EXTOSC
  - External clock (EC)
  - Low-power oscillator (LP)
  - Medium power oscillator (XT)
  - High-power oscillator (HS)
- SOSC
  - Secondary oscillator circuit operating at 31 kHz
- A Phase Lock Loop (PLL) frequency multiplier (4x) is available to the External Oscillator modes enabling clock speeds of up to 64 MHz
- Fail-Safe Clock Monitor: This option constantly monitors the main clock source against a reference signal provided by the LFINTOSC. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued operation or a safe application shutdown.

REGISTER 3-4: Configuration Word 2H (30 0003h): Supervisor

R/W-1	U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{XINST}}$	—	$\overline{\text{DEBUG}}$	STVREN	PPS1WAY	$\overline{\text{ZCD}}$	BORV<1:0>	
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 **$\overline{\text{XINST}}$:** Extended Instruction Set Enable bit
1 = Extended Instruction Set and Indexed Addressing mode disabled (Legacy mode)
0 = Extended Instruction Set and Indexed Addressing mode enabled

bit 6 **Unimplemented:** Read as '1'

bit 5 **$\overline{\text{DEBUG}}$:** Debugger Enable bit
1 = Background debugger disabled
0 = Background debugger enabled

bit 4 **STVREN:** Stack Overflow/Underflow Reset Enable bit
1 = Stack Overflow or Underflow will cause a Reset
0 = Stack Overflow or Underflow will not cause a Reset

bit 3 **PPS1WAY:** PPSLOCKED bit One-Way Set Enable bit
1 = The PPSLOCKED bit can only be set once after an unlocking sequence is executed; once PPSLOCK is set, all future changes to PPS registers are prevented
0 = The PPSLOCKED bit can be set and cleared as needed (provided an unlocking sequence is executed)

bit 2 **$\overline{\text{ZCD}}$:** ZCD Disable bit
1 = ZCD disabled. ZCD can be enabled by setting the ZCDSEN bit of ZCDCON
0 = ZCD always enabled, ZCDMD bit is ignored

bit 1-0 **BORV<1:0>:** Brown-out Reset Voltage Selection bit(1)
PIC18F2xK40 device:
11 = Brown-out Reset Voltage (V_{BOR}) set to 2.45V
10 = Brown-out Reset Voltage (V_{BOR}) set to 2.45V
01 = Brown-out Reset Voltage (V_{BOR}) set to 2.7V
00 = Brown-out Reset Voltage (V_{BOR}) set to 2.85V

PIC18LF2xK40 device:
11 = Brown-out Reset Voltage (V_{BOR}) set to 1.90V
10 = Brown-out Reset Voltage (V_{BOR}) set to 2.45V
01 = Brown-out Reset Voltage (V_{BOR}) set to 2.7V
00 = Brown-out Reset Voltage (V_{BOR}) set to 2.85V

Note 1: The higher voltage setting is recommended for operation at or above 16 MHz.

REGISTER 4-6: OSCTUNE: HFINTOSC TUNING REGISTER

U-0	U-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
—	—	HFTUN<5:0>					
bit 7							bit 0

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

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-0 **HFTUN<5:0>:** HFINTOSC Frequency Tuning bits

01 1111 = Maximum frequency

•

•

•

00 0000 = Center frequency. Oscillator module is running at the calibrated frequency (default value).

•

•

•

10 0000 = Minimum frequency

TABLE 6-1: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE	PEIE	—	—	—	—	—	INTEDG	166
PIE0	—	—	TMR0IE	IOCIE	—	INT2IE	INT1IE	INT0IE	175
PIE1	OSCFIE	CSWIE	—	—	—	—	ADTIE	ADIE	176
PIE2	HLVDIE	ZCDIE	—	—	—	—	C2IE	C1IE	177
PIE3	—	—	RC1IE	TX1IE	—	—	BCL1IE	SSP1IE	178
PIE4	—	—	TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE	179
PIE5	—	—	—	—	—	TMR5GIE	TMR3GIE	TMR1GIE	180
PIE6	—	—	—	—	—	—	CCP2IE	CCP1IE	181
PIE7	SCANIE	CRCIE	NVMIE	—	—	—	—	CWG1IE	182
PIR0	—	—	TMR0IF	IOCIF	—	INT2IF	INT1IF	INT0IF	167
PIR1	OSCFIF	CSWIF	—	—	—	—	ADTIF	ADIF	168
PIR2	HLVDIF	ZCDIF(1)	—	—	—	—	C2IF	C1IF	169
PIR3	—	—	RC1IF	TX1IF	—	—	BCL1IF	SSP1IF	170
PIR4	—	—	TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF	170
IOCAP	—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0	206
IOCAN	—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0	206
IOCAF	—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0	206
IOCCP(1)	—	—	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0	206
IOCCN(1)	—	—	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0	206
IOCCF(1)	—	—	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0	206
STATUS	—	—	—	$\overline{TO}$	$\overline{PD}$	Z	DC	C	114
VREGCON	—	—	—	—	—	—	VREGPM	Reserved	60
CPUDOZE	IDLEN	DOZEN	ROI	DOE	—	DOZE<2:0>			61
WDTCON0	—	—	WDTPS<4:0>					SEN	81
WDTCON1	—	WDTPS<2:0>			—	WINDOW<2:0>			82

Note 1: — = unimplemented location, read as '0'. Shaded cells are not used in Power-Down mode.

REGISTER 7-4: PMD3: PMD CONTROL REGISTER 3

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	PWM4MD	PWM3MD	CCP2MD	CCP1MD
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3 **PWM4MD:** Disable Pulse-Width Modulator PWM4 bit
1 = PWM4 module disabled
0 = PWM4 module enabled

bit 2 **PWM3MD:** Disable Pulse-Width Modulator PWM3 bit
1 = PWM3 module disabled
0 = PWM3 module enabled

bit 1 **CCP2MD:** Disable Pulse-Width Modulator CCP2 bit
1 = CCP2 module disabled
0 = CCP2 module enabled

bit 0 **CCP1MD:** Disable Pulse-Width Modulator CCP1 bit
1 = CCP1 module disabled
0 = CCP1 module enabled

9.1 Register Definitions: Windowed Watchdog Timer Control

REGISTER 9-1: WDTCON0: WATCHDOG TIMER CONTROL REGISTER 0

U-0	U-0	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W-0/0
—	—	WDTPS<4:0>					SEN
bit 7							bit 0

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

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

bit 5-1 **WDTPS<4:0>:** Watchdog Timer Prescale Select bits(1)

Bit Value = Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)

•

•

•

10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 (2^{23}) (Interval 256s nominal)
10001 = 1:4194304 (2^{22}) (Interval 128s nominal)
10000 = 1:2097152 (2^{21}) (Interval 64s nominal)
01111 = 1:1048576 (2^{20}) (Interval 32s nominal)
01110 = 1:524288 (2^{19}) (Interval 16s nominal)
01101 = 1:262144 (2^{18}) (Interval 8s nominal)
01100 = 1:131072 (2^{17}) (Interval 4s nominal)
01011 = 1:65536 (Interval 2s nominal) (Reset value)
01010 = 1:32768 (Interval 1s nominal)
01001 = 1:16384 (Interval 512 ms nominal)
01000 = 1:8192 (Interval 256 ms nominal)
00111 = 1:4096 (Interval 128 ms nominal)
00110 = 1:2048 (Interval 64 ms nominal)
00101 = 1:1024 (Interval 32 ms nominal)
00100 = 1:512 (Interval 16 ms nominal)
00011 = 1:256 (Interval 8 ms nominal)
00010 = 1:128 (Interval 4 ms nominal)
00001 = 1:64 (Interval 2 ms nominal)
00000 = 1:32 (Interval 1 ms nominal)

bit 0 **SEN:** Software Enable/Disable for Watchdog Timer bit

If WDTE<1:0> = 1x:
This bit is ignored.
If WDTE<1:0> = 01:
1 = WDT is turned on
0 = WDT is turned off
If WDTE<1:0> = 00:
This bit is ignored.

Note 1: Times are approximate. WDT time is based on 31 kHz LFINTOSC.

2: When WDTCPS <4:0> in CONFIG3L = 11111, the Reset value of WDTPS<4:0> is 01011. Otherwise, the Reset value of WDTPS<4:0> is equal to WDTCPS<4:0> in CONFIG3L.

3: When WDTCPS <4:0> in CONFIG3L ≠ 11111, these bits are read-only.

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F24/25K40 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F26h to F22h	—	Unimplemented								—
F21h	ANSELC	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0	11111111
F20h	WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0	00000000
F1Fh	ODCONC	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0	00000000
F1Eh	SLRCONC	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0	11111111
F1Dh	INLVLC	INLVLC7	INLVLC6	INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0	11111111
F1Ch	IOCCP	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0	00000000
F1Bh	IOCCN	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0	00000000
F1Ah	IOCCF	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0	00000000
F19h	ANSELB	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0	11111111
F18h	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
F17h	ODCONB	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0	00000000
F16h	SLRCONB	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0	11111111
F15h	INLVLB	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0	11111111
F14h	IOCBP	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0	00000000
F13h	IOCBN	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0	00000000
F12h	IOCBF	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0	00000000
F11h	ANSELA	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0	11111111
F10h	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
F0Fh	ODCONA	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0	00000000
F0Eh	SLRCONA	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0	11111111
F0Dh	INLVLA	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0	11111111
F0Ch	IOCAP	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0	00000000
F0Bh	IOCAN	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0	00000000
F0Ah	IOCAF	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0	00000000
F09h to EFFh	—	Unimplemented								—
EFEh	RC7PPS	—	—	—	RC7PPS<4:0>					---00000
EFDh	RC6PPS	—	—	—	RC6PPS<4:0>					---00000
EFCh	RC5PPS	—	—	—	RC5PPS<4:0>					---00000
EFBh	RC4PPS	—	—	—	RC4PPS<4:0>					---00000
EFAh	RC3PPS	—	—	—	RC3PPS<4:0>					---00000
EF9h	RC2PPS	—	—	—	RC2PPS<4:0>					---00000
EF8h	RC1PPS	—	—	—	RC1PPS<4:0>					---00000
EF7h	RC0PPS	—	—	—	RC0PPS<4:0>					---00000
EF6h	RB7PPS	—	—	—	RB7PPS<4:0>					---00000
EF5h	RB6PPS	—	—	—	RB6PPS<4:0>					---00000
EF4h	RB5PPS	—	—	—	RB5PPS<4:0>					---00000
EF3h	RB4PPS	—	—	—	RB4PPS<4:0>					---00000

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not available on LF devices.

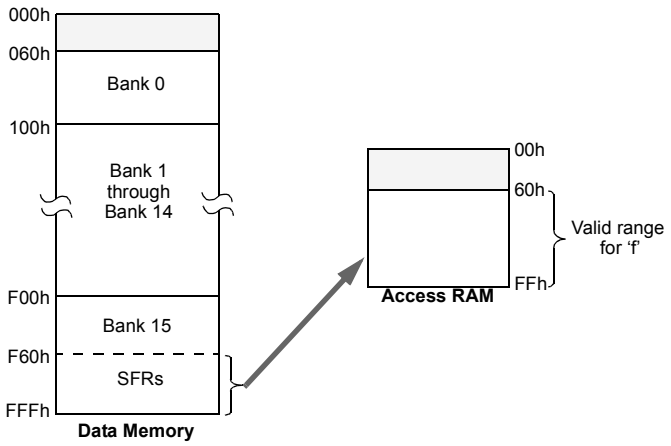
FIGURE 10-7: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

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 0FFh. This is the same as locations F60h to FFFh (Bank 15) of data memory.

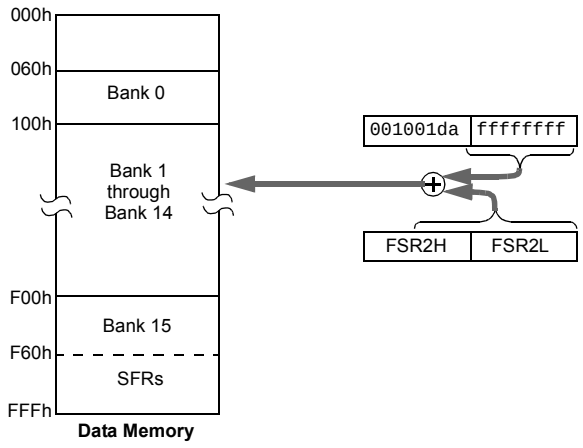
Locations below 060h are not available in this addressing mode.



When 'a' = 0 and f ≤ 5Fh:

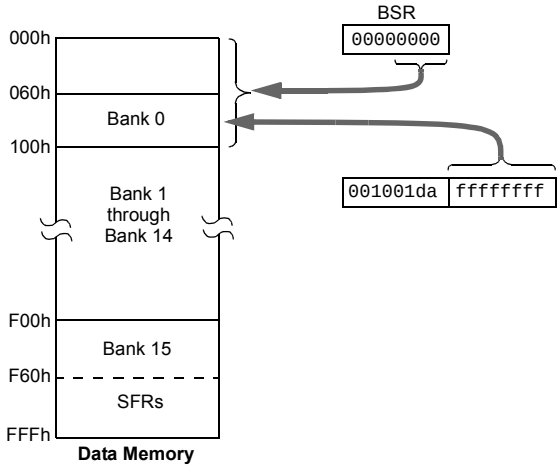
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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.



11.1.5 ERASING PROGRAM FLASH MEMORY

The minimum erase block is 32 or 64 words (refer to Table 11-3). Only through the use of an external programmer, or through ICSP™ control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

For example, when initiating an erase sequence from a microcontroller with erase row size of 32 words, a block of 32 words (64 bytes) of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The NVMCON1 register commands the erase operation. The NVMREG<1:0> bits must be set to point to the Program Flash Memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

The NVM unlock sequence described in **Section 11.1.4 "NVM Unlock Sequence"** should be used to guard against accidental writes. This is sometimes referred to as a long write.

A long write is necessary for erasing the internal Flash. Instruction execution is halted during the long write cycle. The long write is terminated by the internal programming timer.

11.1.5.1 Program Flash Memory Erase Sequence

The sequence of events for erasing a block of internal program memory is:

1. NVMREG bits of the NVMCON1 register point to PFM
2. Set the FREE and WREN bits of the NVMCON1 register
3. Perform the unlock sequence as described in **Section 11.1.4 "NVM Unlock Sequence"**

If the PFM address is write-protected, the WR bit will be cleared and the erase operation will not take place, WRERR is signaled in this scenario.

The operation erases the memory row indicated by masking the LSBs of the current TBLPTR.

While erasing PFM, CPU operation is suspended and it resumes when the operation is complete. Upon completion the WR bit is cleared in hardware, the NVMIF is set and an interrupt will occur if the NVMIE bit is also set.

Write latch data is not affected by erase operations and WREN will remain unchanged.

Note 1:	If a write or erase operation is terminated by an unexpected event, WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.
2:	WRERR is set if WR is written to '1' while TBLPTR points to a write-protected address.
3:	WRERR is set if WR is written to '1' while TBLPTR points to an invalid address location (Table 10-2 and Table 11-1).

14.0 INTERRUPTS

The PIC18(L)F2x/4xK40 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high or low priority level. The high priority interrupt vector is at 0008h and the low priority interrupt vector is at 0018h. A high priority interrupt event will interrupt a low priority interrupt that may be in progress.

The registers for controlling interrupt operation are:

- INTCON
- PIR1, PIR2, PIR3, PIR4, PIR5, PIR6, PIR7
- PIE1, PIE2, PIE3, PIE4, PIE5, PIE6, PIE7
- IPR1, IPR2, IPR3, IPR4, IPR5, IPR6, IPR7

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/ compiler to automatically take care of the placement of these bits within the specified register.

In general, interrupt sources have three bits to control their operation. They are:

- **Flag bit** to indicate that an interrupt event occurred
- **Enable bit** that allows program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** to select high priority or low priority

14.1 Mid-Range Compatibility

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® microcontroller mid-range devices. In Compatibility mode, the interrupt priority bits of the IPRx registers have no effect. The PEIE/GIEL bit of the INTCON register is the global interrupt enable for the peripherals. The PEIE/GIEL bit disables only the peripheral interrupt sources and enables the peripheral interrupt sources when the GIE/GIEH bit is also set. The GIE/GIEH bit of the INTCON register is the global interrupt enable which enables all non-peripheral interrupt sources and disables all interrupt sources, including the peripherals. All interrupts branch to address 0008h in Compatibility mode.

14.2 Interrupt Priority

The interrupt priority feature is enabled by setting the IPEN bit of the INTCON register. When interrupt priority is enabled the GIE/GIEH and PEIE/GIEL Global Interrupt Enable bits of Compatibility mode are replaced by the GIEH high priority, and GIEL low priority, global interrupt enables. When the IPEN bit is set, the GEIH bit of the INTCON register enables all interrupts which have their associated bit in the IPRx register set. When the GEIH bit is cleared, then all interrupt sources including those selected as low priority in the IPRx register are disabled.

When both GIEH and GIEL bits are set, all interrupts selected as low priority sources are enabled.

A high priority interrupt will vector immediately to address 00 0008h and a low priority interrupt will vector to address 00 0018h.

14.3 Interrupt Response

When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. The GIE/GIEH bit is the Global Interrupt Enable when the IPEN bit is cleared. When the IPEN bit is set, enabling interrupt priority levels, the GIEH bit is the high priority Global Interrupt Enable and the GIEL bit is the low priority Global Interrupt Enable. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits in the INTCONx and PIRx registers. The interrupt flag bits must be cleared by software before re-enabling interrupts to avoid repeating the same interrupt.

The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE/GIEH bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the Interrupt-on-change pins, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one-cycle or two-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding enable bits or the Global Interrupt Enable bit.

Note:	Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

14.8 Register Definitions: Interrupt Control

REGISTER 14-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1
GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG
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 **GIE/GIEH:** Global Interrupt Enable bit

If IPEN = 1:
- 1 = Enables all unmasked interrupts and cleared by hardware for high-priority interrupts only
- 0 = Disables all interrupts

If IPEN = 0:
- 1 = Enables all unmasked interrupts and cleared by hardware for all interrupts
- 0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit

If IPEN = 1:
- 1 = Enables all low-priority interrupts and cleared by hardware for low-priority interrupts only
- 0 = Disables all low-priority interrupts

If IPEN = 0:
- 1 = Enables all unmasked peripheral interrupts
- 0 = Disables all peripheral interrupts

bit 5 **IPEN:** Interrupt Priority Enable bit
- 1 = Enable priority levels on interrupts
- 0 = Disable priority levels on interrupts

bit 4-3 **Unimplemented**: Read as '0'

bit 2 **INT2EDG:** External Interrupt 2 Edge Select bit
- 1 = Interrupt on rising edge of INT2 pin
- 0 = Interrupt on falling edge of INT2 pin

bit 1 **INT1EDG:** External Interrupt 1 Edge Select bit
- 1 = Interrupt on rising edge of INT1 pin
- 0 = Interrupt on falling edge of INT1 pin

bit 0 **INT0EDG:** External Interrupt 0 Edge Select bit
- 1 = Interrupt on rising edge of INT0 pin
- 0 = Interrupt on falling edge of INT0 pin

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

21.3.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE Interrupt Priority bit of the PIE6 register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIR6 register following any change in Operating mode.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

21.3.4 CCP PRESCALER

There are four prescaler settings specified by the MODE<3:0> bits of the CCPxCON register. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. Example 21-1 demonstrates the code to perform this function.

EXAMPLE 21-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
BANKSEL CCPxCON     ;Set Bank bits to point
                    ;to CCPxCON
CLRF    CCPxCON     ;Turn CCP module off
MOVLW   NEW_CAPT_PS ;Load the W reg with
                    ;the new prescaler
                    ;move value and CCP ON
MOVWF   CCPxCON     ;Load CCPxCON with this
                    ;value
```

21.3.5 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock (FOSC/4), or by an external clock source.

When Timer1 is clocked by FOSC/4, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep when Timer1 is clocked by an external clock source.

21.4 Compare Mode

The Compare mode function described in this section is available and identical for all CCP modules.

Compare mode makes use of the 16-bit Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMRxH:TMRxL register pair. When a match occurs, one of the following events can occur:

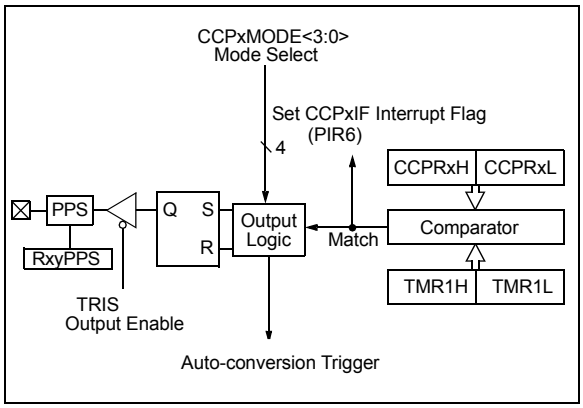
- Toggle the CCPx output, clear TMRx
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Pulse output
- Pulse output, clear TMRx

The action on the pin is based on the value of the MODE<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set, and an ADC conversion can be triggered, if selected.

All Compare modes can generate an interrupt and trigger an ADC conversion. When MODE = `4'b0001` or `4'b1011`, the CCP resets the TMR register pair.

Figure 21-2 shows a simplified diagram of the compare operation.

FIGURE 21-2: COMPARE MODE OPERATION BLOCK DIAGRAM



REGISTER 25-3: MDCARH: MODULATION HIGH CARRIER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	CHS<2:0>(1)		
bit 7							bit 0

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

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **CHS<2:0>:** Modulator Carrier High Selection bits
See Table 25-2 for signal list

REGISTER 25-4: MDCARL: MODULATION LOW CARRIER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	CLS<2:0>(1)		
bit 7							bit 0

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

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **CLS<2:0>:** Modulator Carrier Low Input Selection bits
See Table 25-2 for signal list

TABLE 25-2: MDCARH/MDCARL SELECTION MUX CONNECTIONS

MDCARH			MDCARL		
CHS<2:0>		Connection	CLS<2:0>		Connection
111	7	PWM4 OUT	111	7	PWM4 OUT
110	6	PWM3 OUT	110	6	PWM3 OUT
101	5	CCP2 OUT	101	5	CCP2 OUT
100	4	CCP1 OUT	100	4	CCP1 OUT
011	3	CLKREF output	011	3	CLKREF output
010	2	HFINTOSC	010	2	HFINTOSC
001	1	FOSC (system clock)	001	1	FOSC (system clock)
000	0	Pin selected by MDCARHPPS	000	0	Pin selected by MDCARLPPS

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 35-1: 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
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU Status bits: **C**arry, **D**igit **C**arry, **Z**ero, **Ov**erflow, **N**egative.
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) or 2-bit FSR designator (0h to 3h).
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.
GIE	Global Interrupt Enable bit.
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.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
$\overline{PD}$	Power-down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
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)
TBLPTR	21-bit Table Pointer (points to a Program Memory location).
TABLAT	8-bit Table Latch.
$\overline{TO}$	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
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.
z_s	7-bit offset value for indirect addressing of register files (source).
z_d	7-bit offset value for indirect addressing of register files (destination).
{ }	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr]<n>	Specifies bit n of the register indicated by the pointer expr.
→	Assigned to.
< >	Register bit field.
∈	In the set of.
italics	User defined term (font is Courier).

BNC **Branch if Not Carry**

Syntax: BNC n

Operands: -128 ≤ n ≤ 127

Operation: if CARRY bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0011	nnnn	nnnn

Description: If the CARRY bit is '0', 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 BNC Jump

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

BNN **Branch if Not Negative**

Syntax: BNN n

Operands: -128 ≤ n ≤ 127

Operation: if NEGATIVE bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0111	nnnn	nnnn

Description: If the NEGATIVE bit is '0', 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 BNN Jump

Before Instruction
PC = address (HERE)
After Instruction
If NEGATIVE = 0;
PC = address (Jump)
If NEGATIVE = 1;
PC = address (HERE + 2)

BTG **Bit Toggle f**

Syntax: BTG f, b {,a}

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

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff

Description: Bit 'b' in data memory location 'f' is inverted.
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: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

BOV **Branch if Overflow**

Syntax: BOV n

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

Operation: if OVERFLOW bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn

Description: If the OVERFLOW 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    BOV   Jump
Before Instruction
    PC          =   address (HERE)
After Instruction
    If OVERFLOW =   1;
        PC      =   address (Jump)
    If OVERFLOW =   0;
        PC      =   address (HERE + 2)
```

35.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F2x/4xK40 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDULNK and SUBULNK) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 35-3. Detailed descriptions are provided in **Section 35.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 35-1 apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

35.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets ("[ ]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 35.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }").

TABLE 35-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
ADDFSR	f, k	Add literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK	k	Add literal to FSR2 and return	2	1110	1000	11kk	kkkk	None
CALLW		Call subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word	2	1110	1011	0zzz	zzzz	None
		f_d (destination) 2nd word		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	Move z_s (source) to 1st word	2	1110	1011	1zzz	zzzz	None
		z_d (destination) 2nd word		1111	xxxx	xzzz	zzzz	
PUSHL	k	Store literal at FSR2, decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR	f, k	Subtract literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK	k	Subtract literal from FSR2 and return	2	1110	1001	11kk	kkkk	None

CALLW **Subroutine Call Using WREG**

Syntax: CALLW

Operands: None

Operation: (PC + 2) → TOS,
(W) → PCL,
(PCLATH) → PCH,
(PCLATU) → PCU

Status Affected: None

Encoding:

0000	0000	0001	0100

Description: First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched. Unlike CALL, there is no option to update W, Status or BSR.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read WREG	PUSH PC to stack	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CALLW
```
Before Instruction
```
PC      =   address (HERE)
PCLATH  =   10h
PCLATU  =   00h
W       =   06h
```
After Instruction
```
PC      =   001006h
TOS     =   address (HERE + 2)
PCLATH  =   10h
PCLATU  =   00h
W       =   06h
```

MOVSF **Move Indexed to f**

Syntax: MOVSF [z_s], f_d

Operands: $0 \le z_s \le 127$
$0 \le f_d \le 4095$

Operation: ((FSR2) + z_s) → f_d

Status Affected: None

Encoding:

1st word (source)	1110	1011	0zzz	$zzzz_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of the source register are moved to destination register 'f_d'. The actual address of the source register is determined by adding the 7-bit literal offset 'z_s' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal 'f_d' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).
The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
If the resultant source address points to an indirect addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example:
```
MOVSF   [05h], REG2
```
Before Instruction
```
FSR2        =   80h
Contents
of 85h      =   33h
REG2        =   11h
```
After Instruction
```
FSR2        =   80h
Contents
of 85h      =   33h
REG2        =   33h
```

TABLE 37-23: SPI MODE REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
SP70*	TssL2scH, TssL2scL	$\overline{SS}\downarrow$ to SCK↓ or SCK↑ input	2.25*Tcy	—	—	ns	
SP71*	TscH	SCK input high time (Slave mode)	Tcy + 20	—	—	ns	
SP72*	TscL	SCK input low time (Slave mode)	Tcy + 20	—	—	ns	
SP73*	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge	100	—	—	ns	
SP74*	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge	100	—	—	ns	
SP75*	TdoR	SDO data output rise time	—	10	25	ns	3.0V ≤ VDD ≤ 5.5V
			—	25	50	ns	1.8V ≤ VDD ≤ 5.5V
SP76*	TdoF	SDO data output fall time	—	10	25	ns	
SP77*	TssH2doZ	$\overline{SS}\uparrow$ to SDO output high-impedance	10	—	50	ns	
SP78*	TscR	SCK output rise time (Master mode)	—	10	25	ns	3.0V ≤ VDD ≤ 5.5V
			—	25	50	ns	1.8V ≤ VDD ≤ 5.5V
SP79*	TscF	SCK output fall time (Master mode)	—	10	25	ns	
SP80*	TscH2doV, TscL2doV	SDO data output valid after SCK edge	—	—	50	ns	3.0V ≤ VDD ≤ 5.5V
			—	—	145	ns	1.8V ≤ VDD ≤ 5.5V
SP81*	TdoV2scH, TdoV2scL	SDO data output setup to SCK edge	1 Tcy	—	—	ns	
SP82*	TssL2doV	SDO data output valid after $\overline{SS}\downarrow$ edge	—	—	50	ns	
SP83*	TscH2ssH, TscL2ssH	$\overline{SS}\uparrow$ after SCK edge	1.5 Tcy + 40	—	—	ns	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.