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	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 24x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k40-i-ss

1.2 Other Special Features

- **Memory Endurance:** The Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 10K for program memory and 100K for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-programmability:** These devices can write to their own program memory spaces under internal software control. By using a boot loader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18(L)F2x/4xK40 family introduces an optional extension to the PIC18 instruction set, which adds eight new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced Peripheral Pin Select:** The Peripheral Pin Select (PPS) module connects peripheral inputs and outputs to the device I/O pins. Only digital signals are included in the selections. All analog inputs and outputs remain fixed to their assigned pins.
- **Enhanced Addressable EUSART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-bit A/D Converter with Computation:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead. It has a new module called ADC^2 with computation features, which provides a digital filter and threshold interrupt functions.
- **Windowed Watchdog Timer (WWDT):**
  - Timer monitoring of overflow and underflow events
  - Variable prescaler selection
  - Variable window size selection
  - All sources configurable in hardware or software

1.3 Details on Individual Family Members

Devices in the PIC18(L)F2x/4xK40 family are available in 28-pin and 40/44-pin packages. The block diagram for this device is shown in Figure 1-1.

The devices have the following differences:

1. Program Flash Memory
2. Data Memory SRAM
3. Data Memory EEPROM
4. A/D channels
5. I/O ports
6. Enhanced USART
7. Input Voltage Range/Power Consumption

All other features for devices in this family are identical. These are summarized in Table 1-1.

The pinouts for all devices are listed in the pin summary tables (Table 1 and Table 2).

2.5 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 4.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for details).

The oscillator circuit should be placed on the same side of the board as the device. Place the oscillator circuit close to the respective oscillator pins with no more than 0.5 inch (12 mm) between the circuit components and the pins. The load capacitors should be placed next to the oscillator itself, on the same side of the board.

Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. Also, if using a two-sided board, avoid any traces on the other side of the board where the crystal is placed.

Layout suggestions are shown in Figure 2-3. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

In planning the application's routing and I/O assignments, ensure that adjacent port pins, and other signals in close proximity to the oscillator, are benign (i.e., free of high frequencies, short rise and fall times, and other similar noise).

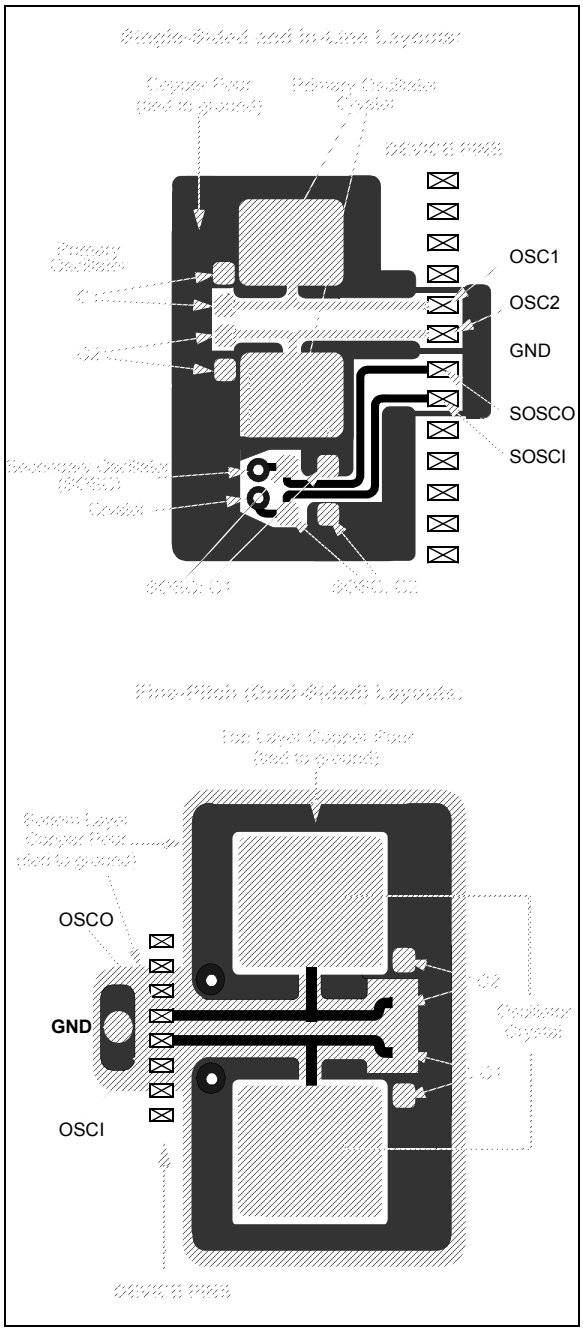
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate website (www.microchip.com):

- *AN826,* "*Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices"*
- *AN849, "Basic PICmicro® Oscillator Design"*
- *AN943, "Practical PICmicro® Oscillator Analysis and Design"*
- *AN949, "Making Your Oscillator Work"*

2.6 Unused I/Os

Unused I/O pins should be configured as outputs and driven to a logic low state. Alternatively, connect a 1 kΩ to 10 kΩ resistor to VSS on unused pins and drive the output to logic low.

FIGURE 2-3: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT



5.1 Clock Source

The input to the reference clock output can be selected using the CLKRCLK register.

5.1.1 CLOCK SYNCHRONIZATION

Once the reference clock enable (EN) is set, the module is ensured to be glitch-free at start-up.

When the reference clock output is disabled, the output signal will be disabled immediately.

Clock dividers and clock duty cycles can be changed while the module is enabled, but glitches may occur on the output. To avoid possible glitches, clock dividers and clock duty cycles should be changed only when the CLKREN is clear.

5.2 Programmable Clock Divider

The module takes the clock input and divides it based on the value of the DIV<2:0> bits of the CLKRCON register (Register 5-1).

The following configurations can be made based on the DIV<2:0> bits:

- Base FOSC value
- FOSC divided by 2
- FOSC divided by 4
- FOSC divided by 8
- FOSC divided by 16
- FOSC divided by 32
- FOSC divided by 64
- FOSC divided by 128

The clock divider values can be changed while the module is enabled; however, in order to prevent glitches on the output, the DIV<2:0> bits should only be changed when the module is disabled (EN = 0).

5.3 Selectable Duty Cycle

The DC<1:0> bits of the CLKRCON register can be used to modify the duty cycle of the output clock. A duty cycle of 25%, 50%, or 75% can be selected for all clock rates, with the exception of the undivided base FOSC value.

The duty cycle can be changed while the module is enabled; however, in order to prevent glitches on the output, the DC<1:0> bits should only be changed when the module is disabled (EN = 0).

Note:	The DC1 bit is reset to '1'. This makes the default duty cycle 50% and not 0%.

5.4 Operation in Sleep Mode

The reference clock output module clock is based on the system clock. When the device goes to Sleep, the module outputs will remain in their current state. This will have a direct effect on peripherals using the reference clock output as an input signal. No change should occur in the module from entering or exiting from Sleep.

7.5 Register Definitions: Peripheral Module Disable

REGISTER 7-1: PMD0: PMD CONTROL REGISTER 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	R/W-0/0
SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
7							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 **SYSCMD:** Disable Peripheral System Clock Network bit(1)
See description in **Section 7.4 "System Clock Disable"**.
1 = System clock network disabled (FOSC)
0 = System clock network enabled

bit 6 **FVRMD:** Disable Fixed Voltage Reference bit
1 = FVR module disabled
0 = FVR module enabled

bit 5 **HLVDMD:** Disable Low-Voltage Detect bit
1 = HLVD module disabled
0 = HLVD module enabled

bit 4 **CRCMD:** Disable CRC Engine bit
1 = CRC module disabled
0 = CRC module enabled

bit 3 **SCANMD:** Disable NVM Memory Scanner bit(2)
1 = NVM Memory Scan module disabled
0 = NVM Memory Scan module enabled

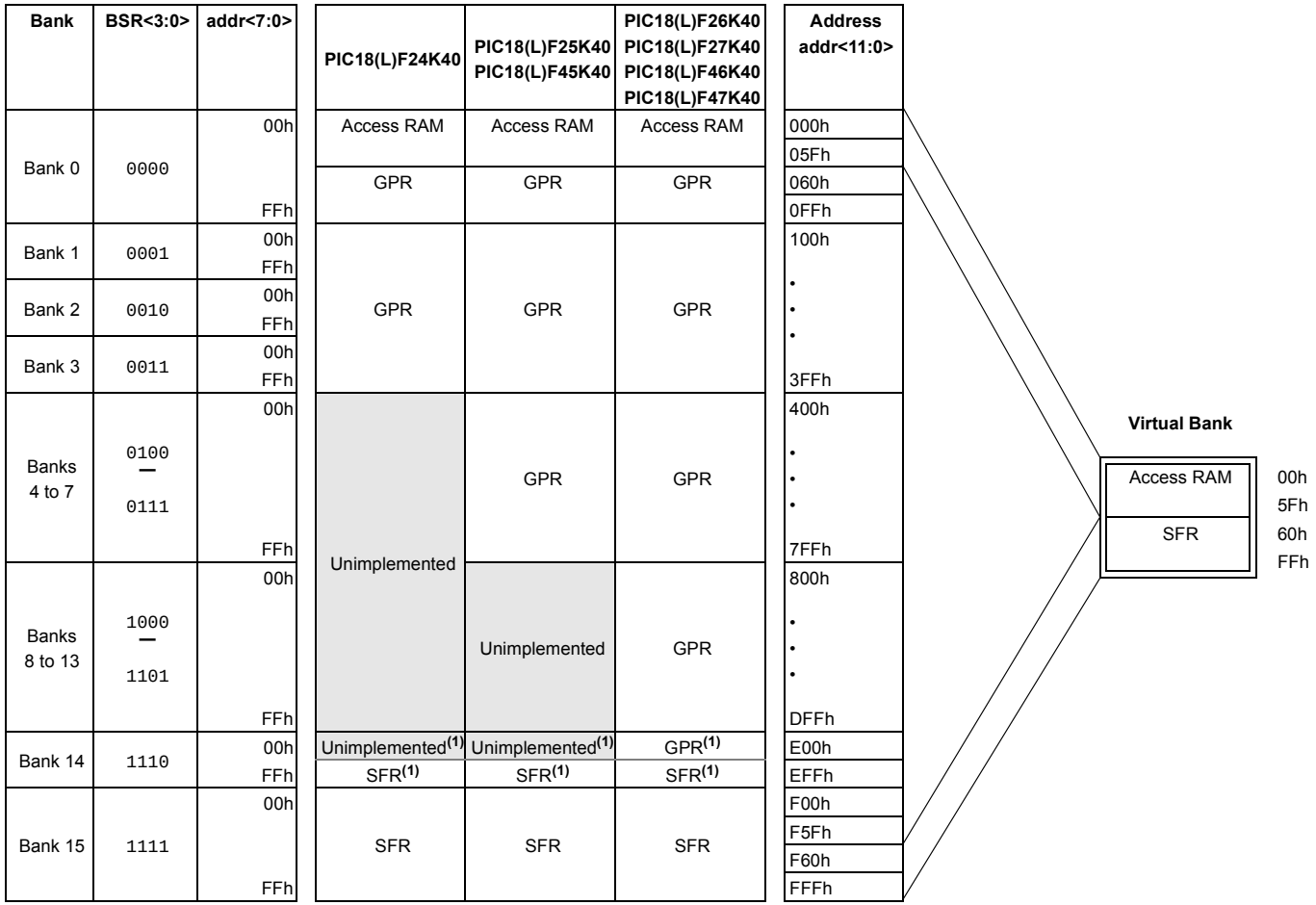
bit 2 **NVMMD:** NVM Module Disable bit(3)
1 = All Memory reading and writing is disabled; NVMCON registers cannot be written
0 = NVM module enabled

bit 1 **CLKRMD:** Disable Clock Reference bit
1 = CLKR module disabled
0 = CLKR module enabled

bit 0 **IOCMD:** Disable Interrupt-on-Change bit, All Ports
1 = IOC module(s) disabled
0 = IOC module(s) enabled

Note 1: Clearing the SYSCMD bit disables the system clock (FOSC) to peripherals, however peripherals clocked by FOSC/4 are not affected.
2: Subject to SCANE bit in CONFIG4H.
3: When enabling NVM, a delay of up to 1 µs may be required before accessing data.

FIGURE 10-4: DATA MEMORY MAP FOR PIC18(L)F2X/4XK40 DEVICES



Note 1: It depends on the number of SFRs. Refer to Table 10-3 and Table 10-4.

10.7.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

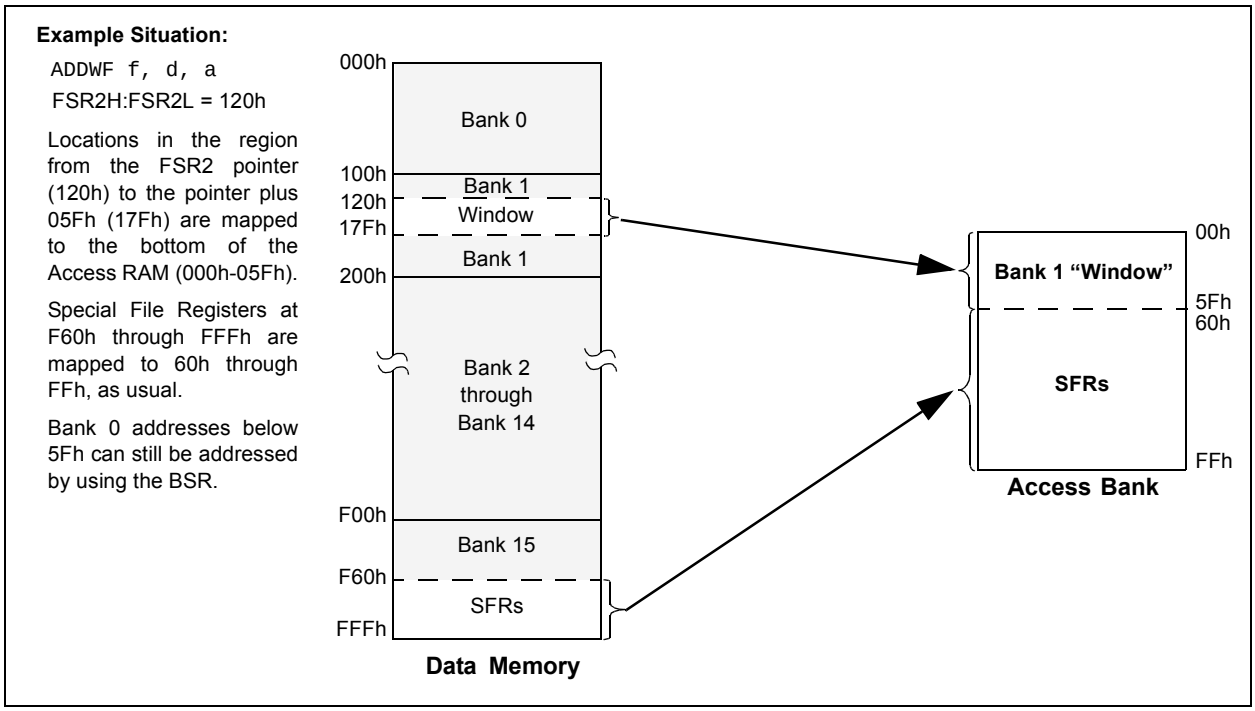
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom section of Bank 0, this mode maps the contents from a user defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 10.4.2 “Access Bank”**). An example of Access Bank remapping in this addressing mode is shown in Figure 10-8.

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is ‘1’) will continue to use direct addressing as before.

10.8 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds eight additional commands to the existing PIC18 instruction set. These instructions are executed as described in **Section 35.2 “Extended Instruction Set”**.

FIGURE 10-8: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING



11.1 Program Flash Memory

The Program Flash Memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed one byte at a time. A write to program memory or program memory erase is executed on blocks of *n* bytes at a time. Refer to Table 11-3 for write and erase block sizes. A Bulk Erase operation cannot be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

It is important to understand the PFM memory structure for erase and programming operations. Program memory word size is 16 bits wide. PFM is arranged in rows. A row is the minimum size that can be erased by user software. Refer to Table 11-3 for the row sizes for the these devices.

After a row has been erased, all or a portion of this row can be programmed. Data to be written into the program memory row is written to 6-bit wide data write latches. These latches are not directly accessible, but may be loaded via sequential writes to the TABLAT register.

Note:	To modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, the new data and retained data can be written into the write latches to reprogram the row of PFM. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations

TABLE 11-2: FLASH MEMORY ORGANIZATION BY DEVICE

Device	Row Erase Size (Words)	Write Latches (Bytes)	Program Flash Memory (Words)	Data Memory (Bytes)
PIC18(L)F45K40	32	64	16384	256
PIC18(L)F26K40			32768	1024
PIC18(L)F46K40				
PIC18(L)F27K40	64	128	65536	
PIC18(L)F47K40				

12.0 8x8 HARDWARE MULTIPLIER

12.1 Introduction

All PIC18 devices include an 8x8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 12-1.

12.2 Operation

Example 12-1 shows the instruction sequence for an 8x8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 12-2 shows the sequence to do an 8x8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 12-1: 8x8 UNSIGNED MULTIPLY ROUTINE

```
MOVF   ARG1, W   ;
MULWF  ARG2      ; ARG1 * ARG2 ->
                 ; PRODH:PRODL
```

EXAMPLE 12-2: 8x8 SIGNED MULTIPLY ROUTINE

```
MOVF   ARG1, W
MULWF  ARG2      ; ARG1 * ARG2 ->
                 ; PRODH:PRODL
BTFSC  ARG2, SB  ; Test Sign Bit
SUBWF  PRODH, F  ; PRODH = PRODH
                 ;          - ARG1
MOVF   ARG2, W
BTFSC  ARG1, SB  ; Test Sign Bit
SUBWF  PRODH, F  ; PRODH = PRODH
                 ;          - ARG2
```

TABLE 12-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time			
				@ 64 MHz	@ 40 MHz	@ 10 MHz	@ 4 MHz
8x8 unsigned	Without hardware multiply	13	69	4.3 μs	6.9 μs	27.6 μs	69 μs
	Hardware multiply	1	1	62.5 ns	100 ns	400 ns	1 μs
8x8 signed	Without hardware multiply	33	91	5.7 μs	9.1 μs	36.4 μs	91 μs
	Hardware multiply	6	6	375 ns	600 ns	2.4 μs	6 μs
16x16 unsigned	Without hardware multiply	21	242	15.1 μs	24.2 μs	96.8 μs	242 μs
	Hardware multiply	28	28	1.8 μs	2.8 μs	11.2 μs	28 μs
16x16 signed	Without hardware multiply	52	254	15.9 μs	25.4 μs	102.6 μs	254 μs
	Hardware multiply	35	40	2.5 μs	4.0 μs	16.0 μs	40 μs

REGISTER 13-6: CRCACCL: CRC ACCUMULATOR LOW BYTE 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
ACC<7:0>							
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 7-0 **ACC<7:0>**: CRC Accumulator Register bits
Writing to this register writes to the CRC accumulator register through the CRC write bus. Reading from this register reads the CRC accumulator.

REGISTER 13-7: CRCSHIFTH: CRC SHIFT HIGH BYTE REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SHIFT<15:8>							
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 7-0 **SHIFT<15:8>**: CRC Shifter Register bits
Reading from this register reads the CRC Shifter.

REGISTER 13-8: CRCSHIFTL: CRC SHIFT LOW BYTE REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SHIFT<7:0>							
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 7-0 **SHIFT<7:0>**: CRC Shifter Register bits
Reading from this register reads the CRC Shifter.

TABLE 13-5: SUMMARY OF REGISTERS ASSOCIATED WITH CRC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CRCACCH	ACC<15:8>								152
CRCACCL	ACC<7:0>								153
CRCCON0	EN	GO	BUSY	ACCM	—	—	SHIFTM	FULL	151
CRCCON1	DLEN<3:0>				PLEN<3:0>				151
CRCDATH	DATA<15:8>								152
CRCDATL	DATA<7:0>								152
CRCSHIFTH	SHIFT<15:8>								153
CRCSHIFTL	SHIFT<7:0>								153
CRCXORH	X<15:8>								154
CRCXORL	X<7:1>							—	154
PMD0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	68
SCANCON0	SCANEN	SCANGO	BUSY	INVALID	INTM	—	MODE<1:0>		155
SCANHADRU	—	—	HADR<21:16>						157
SCANHADRH	HADR<15:8>								158
SCANHADRL	HADR<7:0>								158
SCANLADRU	—	—	LADR<21:16>						156
SCANLADRH	LADR<15:8>								156
SCANLADRL	LADR<7:0>								157
SCANTRIG	—	—	—	—	TSEL<3:0>				159
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	170
PIR7	SCANIF	CRCIF	NVMIF	—	—	—	—	CWG1IF	178
PIE7	SCANIE	CRCIE	NVMIE	—	—	—	—	CWG1IE	186
IPR7	SCANIP	CRCIP	NVMIP	—	—	—	—	CWG1IP	194

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for the CRC module.

REGISTER 15-4: ANSELx: ANALOG SELECT REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
ANSELx7	ANSELx6	ANSELx5	ANSELx4	ANSELx3	ANSELx2	ANSELx1	ANSELx0
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 **ANSELx<7:0>**: Analog Select on Pins Rx<7:0>

1 = Digital Input buffers are disabled.

0 = ST and TTL input devices are enabled

TABLE 15-5: ANALOG SELECT 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								
ANSELA	X	X	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
ANSELB	X	X	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
ANSELC	X	X	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
ANSELD	X		—	—	—	—	—	—	—	—
		X	ANSELD7	ANSELD6	ANSELD5	ANSELD4	ANSELD3	ANSELD2	ANSELD1	ANSELD0
ANSELE	X		—	—	—	—	—	—	—	—
		X	—	—	—	—	—	ANSELE2	ANSELE1	ANSELE0

REGISTER 15-8: INLVLx: INPUT LEVEL CONTROL REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
INLVLx7	INLVLx6	INLVLx5	INLVLx4	INLVLx3	INLVLx2	INLVLx1	INLVLx0
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 **INLVLx<7:0>**: Input Level Select on Pins Rx<7:0>, respectively
1 = ST input used for port reads and interrupt-on-change
0 = TTL input used for port reads and interrupt-on-change

TABLE 15-9: INPUT LEVEL 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								
INLVLA	X	X	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
INLVLB	X	X	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2(1)	INLVLB1(1)	INLVLB0
INLVLC	X	X	INLVLC7	INLVLC6	INLVLC5	INLVLC4(1)	INLVLC3(1)	INLVLC2	INLVLC1	INLVLC0
INLVLD	X		—	—	—	—	—	—	—	—
		X	INLVLD7	INLVLD6	INLVLD5	INLVLD4	INLVLD3	INLVLD2	INLVLD1(1)	INLVLD0(1)
INLVLE	X		—	—	—	—	INLVLE3	—	—	—
		X	—	—	—	—	INLVLE3	INLVLE2	INLVLE1	INLVLE0

Note 1: Pins read the I^2C ST inputs when MSSP inputs select these pins, and I^2C mode is enabled.

22.1.9 SETUP FOR PWM OPERATION USING PWMx PINS

The following steps should be taken when configuring the module for PWM operation using the PWMx pins:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the PR2 register with the PWM period value.
4. Load the PWMxDCH register and bits <7:6> of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the PIR4 register. See Note 1 below.
   - Select the timer clock source to be as FOSC/4 using the TxCLKCON register. This is required for correct operation of the PWM module.
   - Configure the T2CKPS bits of the T2CON register with the Timer2 prescale value.
   - Enable Timer2 by setting the T2ON bit of the T2CON register.
6. Enable PWM output pin and wait until Timer2 overflows, TMR2IF bit of the PIR4 register is set. See note below.
7. Enable the PWMx pin output driver(s) by clearing the associated TRIS bit(s) and setting the desired pin PPS control bits.
8. Configure the PWM module by loading the PWMxCON register with the appropriate values.

Note 1: In order to send a complete duty cycle and period on the first PWM output, the above steps must be followed in the order given. If it is not critical to start with a complete PWM signal, then move Step 8 to replace Step 4.

2: For operation with other peripherals only, disable PWMx pin outputs.

22.1.10 SETUP FOR PWM OPERATION TO OTHER DEVICE PERIPHERALS

The following steps should be taken when configuring the module for PWM operation to be used by other device peripherals:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the PR2 register with the PWM period value.
4. Load the PWMxDCH register and bits <7:6> of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the PIR4 register. See Note 1 below.
   - Select the timer clock source to be as FOSC/4 using the TxCLKCON register. This is required for correct operation of the PWM module.
   - Configure the T2CKPS bits of the T2CON register with the Timer2 prescale value.
   - Enable Timer2 by setting the T2ON bit of the T2CON register.
6. Enable PWM output pin:
   - Wait until Timer2 overflows, TMR2IF bit of the PIR4 register is set. See Note 1 below.
7. Configure the PWM module by loading the PWMxCON register with the appropriate values.

Note 1: In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

REGISTER 22-2: CCPTMRS: CCP TIMERS CONTROL REGISTER

R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1
P4TSEL<1:0>		P3TSEL<1:0>		C2TSEL<1:0>		C1TSEL<1:0>	
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 **P4TSEL<1:0>:** PWM4 Timer Selection bits
11 = PWM4 based on TMR6
10 = PWM4 based on TMR4
01 = PWM4 based on TMR2
00 = Reserved

bit 5-4 **P3TSEL<1:0>:** PWM3 Timer Selection bits
11 = PWM3 based on TMR6
10 = PWM3 based on TMR4
01 = PWM3 based on TMR2
00 = Reserved

bit 3-2 **C2TSEL<1:0>:** CCP2 Timer Selection bits
11 = CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10 = CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01 = CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00 = Reserved

bit 1-0 **C1TSEL<1:0>:** CCP1 Timer Selection bits
11 = CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10 = CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01 = CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00 = Reserved

REGISTER 24-6: CWG1AS0: CWG AUTO-SHUTDOWN CONTROL REGISTER 0

R/W/HS/HC-0/0	R/W-0/0	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	U-0	U-0
SHUTDOWN	REN	LSBD<1:0>		LSAC<1:0>		—	—
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	HS/HC = Bit is set/cleared by hardware
q = Value depends on condition		

bit 7 **SHUTDOWN:** Auto-Shutdown Event Status bit(1,2)
- 1 = An auto-shutdown state is in effect
- 0 = No auto-shutdown event has occurred

bit 6 **REN:** Auto-Restart Enable bit
- 1 = Auto-restart is enabled
- 0 = Auto-restart is disabled

bit 5-4 **LSBD<1:0>:** CWG1B and CWG1D Auto-Shutdown State Control bits
- 11 = A logic '1' is placed on CWG1B/D when an auto-shutdown event occurs.
- 10 = A logic '0' is placed on CWG1B/D when an auto-shutdown event occurs.
- 01 = Pin is tri-stated on CWG1B/D when an auto-shutdown event occurs.
- 00 = The inactive state of the pin, including polarity, is placed on CWG1B/D after the required dead-band interval when an auto-shutdown event occurs.

bit 3-2 **LSAC<1:0>:** CWG1A and CWG1C Auto-Shutdown State Control bits
- 11 = A logic '1' is placed on CWG1A/C when an auto-shutdown event occurs.
- 10 = A logic '0' is placed on CWG1A/C when an auto-shutdown event occurs.
- 01 = Pin is tri-stated on CWG1A/C when an auto-shutdown event occurs.
- 00 = The inactive state of the pin, including polarity, is placed on CWG1A/C after the required dead-band interval when an auto-shutdown event occurs.

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

Note 1: This bit may be written while EN = 0 (Register 24-1), to place the outputs into the shutdown configuration.

2: The outputs will remain in auto-shutdown state until the next rising edge of the CWG data input after this bit is cleared.

26.0 MASTER SYNCHRONOUS SERIAL PORT MODULE

Note: The PIC18(L)F26/45/46K40 devices have two MSSP. Therefore, all information in this section refers to both MSSP1 and MSSP2.

26.1 MSSP Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The PIC18(L)F26/45/46K40 devices have two MSSP modules that can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)

The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Parity
- Slave Select Synchronization (Slave mode only)
- Daisy-chain connection of slave devices

The I^2C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDA hold times

26.2 SPI Mode Overview

The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select ($\overline{SS}$)

Figure 26-1 shows the block diagram of the MSSP module when operating in SPI mode.

31.4.2 PRECHARGE CONTROL

The Precharge stage is an optional period of time that brings the external channel and internal sample and hold capacitor to known voltage levels. Precharge is enabled by writing a non-zero value to the ADPRE register. This stage is initiated when an ADC conversion begins, either from setting the ADGO bit, a special event trigger, or a conversion restart from the computation functionality. If the ADPRE register is cleared when an ADC conversion begins, this stage is skipped.

During the precharge time, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the ADPPOL bit of ADCON1. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is also determined by the ADPPOL bit of ADCON1. The amount of time that this charging needs is controlled by the ADPRE register.

Note:	The external charging overrides the TRIS setting of the respective I/O pin. If there is a device attached to this pin, Precharge should not be used.

31.4.3 ACQUISITION CONTROL

The Acquisition stage is an optional time for the voltage on the internal sample and hold capacitor to charge or discharge from the selected analog channel.This acquisition time is controlled by the ADACQ register. If ADPRE = 0, acquisition starts at the beginning of conversion. When ADPRE = 1, the acquisition stage begins when precharge ends.

At the start of the acquisition stage, the port pin logic of the selected analog channel is overridden to turn off the digital high/low output drivers so they do not affect the final result of the charge averaging. Also, the selected ADC channel is connected to CHOLD. This allows charge averaging to proceed between the precharged channel and the CHOLD capacitor.

Note:	When ADPRE! = 0, acquisition time cannot be '0'. In this case, setting ADACQ to '0' will set a maximum acquisition time (256 ADC clock cycles). When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

31.4.4 GUARD RING OUTPUTS

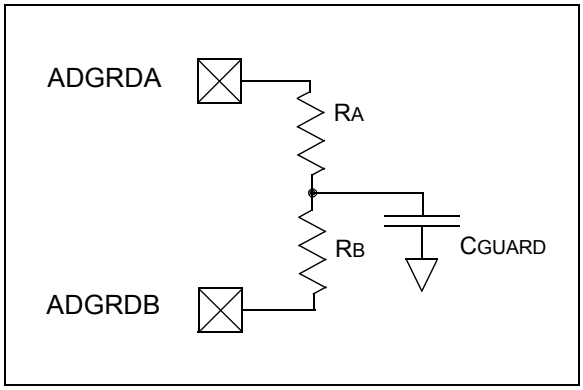
Figure 31-8 shows a typical guard ring circuit. CGUARD represents the capacitance of the guard ring trace placed on the PCB board. The user selects values for RA and RB that will create a voltage profile on CGUARD, which will match the selected acquisition channel.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, "*mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider*" (DS01478).

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs can be routed through PPS controls to I/O pins (see **Section 17.0 "Peripheral Pin Select (PPS) Module"** for details) and the polarity of these outputs are controlled by the ADGPOL and ADIPEN bits of ADCON1.

At the start of the first precharge stage, both outputs are set to match the ADGPOL bit of ADCON1. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the ADIPEN bit of ADCON1 causes both guard ring outputs to transition to the opposite polarity of ADGPOL at the start of the second precharge stage, and ADGRDA toggles again for the second acquisition. For more information on the timing of the guard ring output, refer to Figure 31-8 and Figure 31-9.

FIGURE 31-8: GUARD RING CIRCUIT



LFSR **Load FSR**

Syntax: LFSR f, k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk

Description: The 12-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL

Example: `LFSR 2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

MOVF **Move f**

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

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

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

0101	00da	ffff	ffff

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank.
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 W

Example: `MOVF REG, 0, 0`

Before Instruction
REG = 22h
W = FFh
After Instruction
REG = 22h
W = 22h

RETFIE **Return from Interrupt**

Syntax: RETFIE {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

0000	0000	0001	000s

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	POP PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example:

```
RETFIE  1
```

After Interrupt
PC = TOS
W = WS
BSR = BSRS
Status = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETLW **Return literal to W**

Syntax: RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	1100	kkkk	kkkk

Description: W is loaded with the 8-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	POP PC from stack, Write to W
No operation	No operation	No operation	No operation

Example:

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
    :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
    :
    :
    RETLW kn   ; End of table
```

Before Instruction
W = 07h
After Instruction
W = value of kn

SUBWFB **Subtract W from f with Borrow**

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

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

Operation: (f) – (W) – $(\overline{C})$ → dest

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

Encoding:

0101	10da	ffff	ffff

Description: Subtract W and the CARRY flag (borrow) from register 'f' (2's complement method). 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 1:
```
SUBWFB  REG, 1, 0
```
Before Instruction
```
REG  =  19h   (0001 1001)
W    =  0Dh   (0000 1101)
C    =  1
```
After Instruction
```
REG  =  0Ch   (0000 1100)
W    =  0Dh   (0000 1101)
C    =  1
Z    =  0
N    =  0     ; result is positive
```
Example 2:
```
SUBWFB  REG, 0, 0
```
Before Instruction
```
REG  =  1Bh   (0001 1011)
W    =  1Ah   (0001 1010)
C    =  0
```
After Instruction
```
REG  =  1Bh   (0001 1011)
W    =  00h
C    =  1
Z    =  1     ; result is zero
N    =  0
```
Example 3:
```
SUBWFB  REG, 1, 0
```
Before Instruction
```
REG  =  03h   (0000 0011)
W    =  0Eh   (0000 1110)
C    =  1
```
After Instruction
```
REG  =  F5h   (1111 0101)
                ; [2's comp]
W    =  0Eh   (0000 1110)
C    =  0
Z    =  0
N    =  1     ; result is negative
```

SWAPF **Swap f**

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

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

Operation: (f<3:0>) → dest<7:4>,
(f<7:4>) → dest<3:0>

Status Affected: None

Encoding:

0011	10da	ffff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed 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:
```
SWAPF   REG, 1, 0
```
Before Instruction
```
REG  =  53h
```
After Instruction
```
REG  =  35h
```