

Welcome to [E-XFL.COM](https://www.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	I ² C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
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-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f24k40-i-so

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).

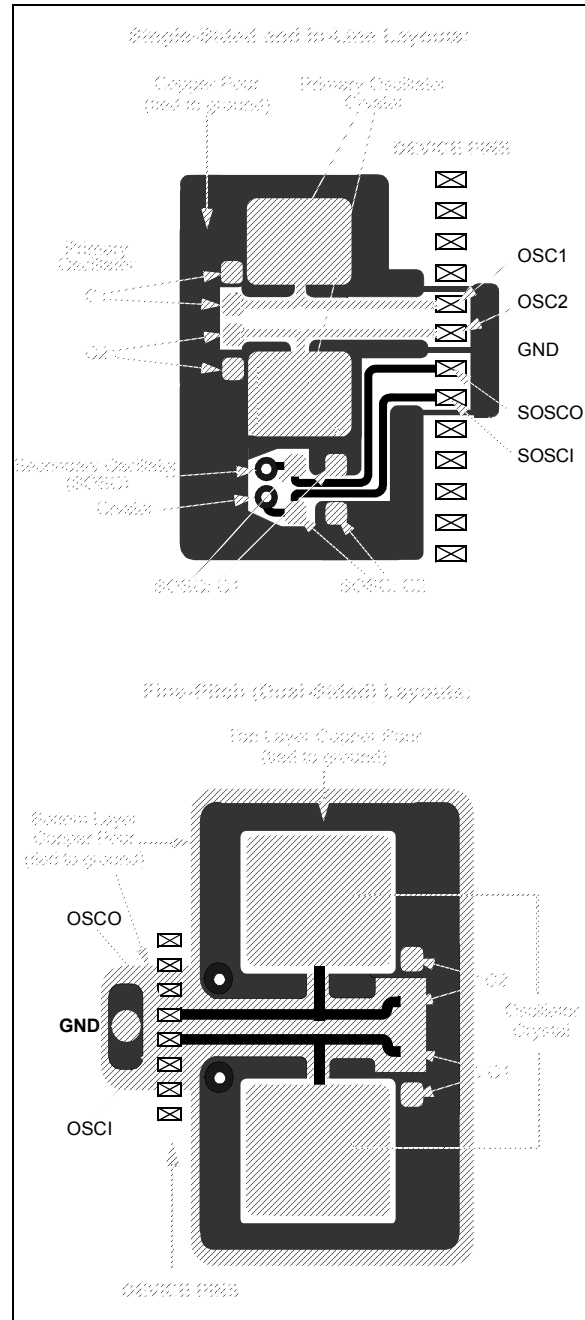
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate website (www.microchip.com):

- AN826, “Crystal Oscillator Basics and Crystal Selection for rPIC™ 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



PIC18(L)F24/25K40

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

R/W-1	R/W-1	R/W-1	U-1	U-1	U-1	R/W-1	R/W-1
BOREN<1:0>		LPBOREN	—	—	—	PWRTE	MCLRE
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6

BOREN<1:0>: Brown-out Reset Enable bits

When enabled, Brown-out Reset Voltage (VBOR) is set by BORV bit

11 = Brown-out Reset enabled, SBOREN bit is ignored

10 = Brown-out Reset enabled while running, disabled in Sleep; SBOREN is ignored

01 = Brown-out Reset enabled according to SBOREN

00 = Brown-out Reset disabled

bit 5

LPBOREN: Low-Power BOR Enable bit

1 = Low-Power Brown-out Reset is disabled

0 = Low-Power Brown-out Reset is enabled

bit 4-2

Unimplemented: Read as '1'

bit 1

PWRTE: Power-up Timer Enable bit

1 = PWRT disabled

0 = PWRT enabled

bit 0

MCLRE: Master Clear ($\overline{\text{MCLR}}$) Enable bit

If LVP = 1

RE3 pin function is $\overline{\text{MCLR}}$

If LVP = 0

1 = $\overline{\text{MCLR}}$ pin is $\overline{\text{MCLR}}$

0 = $\overline{\text{MCLR}}$ pin function is port defined function

3.6 Device ID and Revision ID

The 16-bit device ID word is located at 3F FFEh and the 16-bit revision ID is located at 3F FFFCh. These locations are read-only and cannot be erased or modified.

Development tools, such as device programmers and debuggers, may be used to read the Device ID, Revision ID and Configuration Words. Refer to **11.0 “Nonvolatile Memory (NVM) Control”** for more information on accessing these locations.

3.7 Register Definitions: Device and Revision

REGISTER 3-12: DEVICE ID: DEVICE ID REGISTER

R	R	R	R	R	R	R	R
DEV15	DEV14	DEV13	DEV12	DEV11	DEV10	DEV9	DEV8
bit 15							bit 8

R	R	R	R	R	R	R	R
DEV7	DEV6	DEV5	DEV4	DEV3	DEV2	DEV1	DEV0
bit 7							bit 0

Legend:

R = Readable bit '1' = Bit is set 0' = Bit is cleared x = Bit is unknown

bit 15-0 **DEV<15:0>**: Device ID bits

Device	Device ID
PIC18F24K40	69C0h
PIC18F25K40	69A0h
PIC18LF24K40	6AA0h
PIC18LF25K40	6A80h

10.3 PIC18 Instruction Cycle

10.3.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the program counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the instruction register during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 10-2.

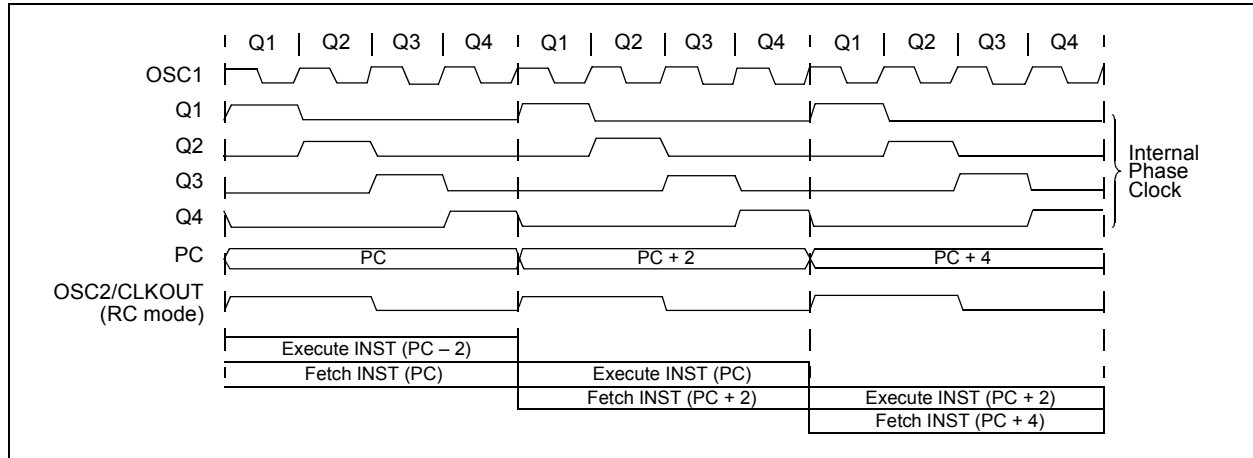
10.3.2 INSTRUCTION FLOW/PIPELINING

An "Instruction Cycle" consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), then two cycles are required to complete the instruction (Example 10-3).

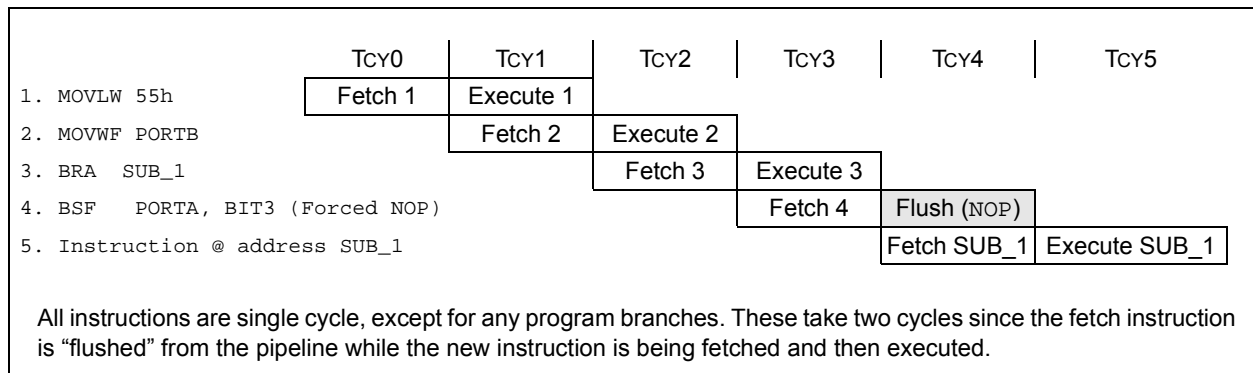
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

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

FIGURE 10-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 10-3: INSTRUCTION PIPELINE FLOW



10.6 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 10.7 “Data Memory and the Extended Instruction Set”** for more information.

While the program memory can be addressed in only one way – through the program counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in **Section 10.7.1 “Indexed Addressing with Literal Offset”**.

10.6.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include `SLEEP`, `RESET` and `DAW`.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include `ADDLW` and `MOVLW`, which respectively, add or move a literal value to the W register. Other examples include `CALL` and `GOTO`, which include a 20-bit program memory address.

10.6.2 DIRECT ADDRESSING

Direct addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 10.4.3 “General Purpose Register File”**) or a location in the Access Bank (**Section 10.4.2 “Access Bank”**) as the data source for the instruction.

The Access RAM bit ‘a’ determines how the address is interpreted. When ‘a’ is ‘1’, the contents of the BSR (**Section 10.4.1 “Bank Select Register (BSR)”**) are used with the address to determine the complete 12-bit address of the register. When ‘a’ is ‘0’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as `MOVFF`, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation’s results is determined by the destination bit ‘d’. When ‘d’ is ‘1’, the results are stored back in the source register, overwriting its original contents. When ‘d’ is ‘0’, the results are stored in the W register. Instructions without the ‘d’ argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

10.6.3 INDIRECT ADDRESSING

Indirect addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations which are to be read or written. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for indirect addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 10-5.

EXAMPLE 10-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```

LFSR   FSR0, 100h ;
NEXT   CLRF POSTINC0 ; Clear INDF
                                ; register then
                                ; inc pointer
        BTFSS FSR0H, 1 ; All done with
                                ; Bank1?
        BRA   NEXT     ; NO, clear next
CONTINUE                                ; YES, continue
    
```

11.1.4 NVM UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the NVM from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- PFM Row Erase
- Write of PFM write latches to PFM memory
- Write of PFM write latches to User IDs
- Write to Data EEPROM Memory
- Write to Configuration Words

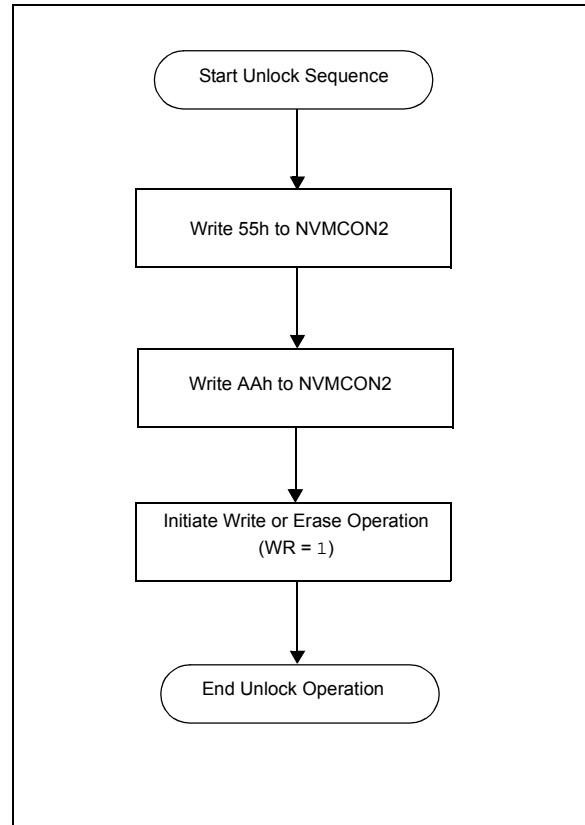
The unlock sequence consists of the following steps and must be completed in order:

- Write 55h to NVMCON2
- Write AAh to NVMCON2
- Set the WR bit of NVMCON1

Once the WR bit is set, the processor will stall internal operations until the operation is complete and then resume with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

FIGURE 11-6: NVM UNLOCK SEQUENCE FLOWCHART



EXAMPLE 11-2: NVM UNLOCK SEQUENCE

```

BCF          INTCON,GIE      ; Recommended so sequence is not interrupted
BANKSEL     NVMCON1
BSF         NVMCON1,WREN     ; Enable write/erase
MOVLW      55h              ; Load 55h

MOVWF      NVMCON2          ; Step 1: Load 55h into NVMCON2
MOVLW      AAh              ; Step 2: Load W with AAh
MOVWF      NVMCON2          ; Step 3: Load AAh into NVMCON2
BSF         INTCON1,WR       ; Step 4: Set WR bit to begin write/erase

BSF         INTCON,GIE      ; Re-enable interrupts
  
```

Note 1: Sequence begins when NVMCON2 is written; steps 1-4 must occur in the cycle-accurate order shown. If the timing of the steps 1 to 4 is corrupted by an interrupt or a debugger Halt, the action will not take place.

2: Opcodes shown are illustrative; any instruction that has the indicated effect may be used.

The user needs to load the TBLPTR and TABLAT register with the address and data byte respectively before executing the write command. An unlock sequence needs to be followed for writing to the USER IDs/DEVICE IDs/CONFIG words (Section 11.1.4, NVM Unlock Sequence). If WRTC = 0 or if TBLPTR points an invalid address location (see Table 11-3), WR bit is cleared without any effect and WRERR is set.

A single CONFIG word byte is written at once and the operation includes an implicit erase cycle for that byte (it is not necessary to set FREE). CPU execution is stalled and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The new CONFIG value takes effect when the CPU resumes operation.

TABLE 11-4: USER ID, DEV/REV ID AND CONFIGURATION WORD ACCESS (NVMREG<1:0> = x1)

Address	Function	Read Access	Write Access
20 0000h-20 000Fh	User IDs	Yes	Yes
3F FFFCh-3F FFFFh	Revision ID/Device ID	Yes	No
30 0000h-30 000Bh	Configuration Words 1-6	Yes	Yes

PIC18(L)F24/25K40

REGISTER 14-15: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	TMR5GIE	TMR3GIE	TMR1GIE
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 7-3 **Unimplemented:** Read as '0'

bit 2 **TMR5GIE:** TMR5 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3GIE:** TMR3 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR1GIE:** TMR1 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

17.3 Bidirectional Pins

PPS selections for peripherals with bidirectional signals on a single pin must be made so that the PPS input and PPS output select the same pin. Peripherals that have bidirectional signals include:

- EUSART (synchronous operation)
- MSSP (I²C)
- CCP module

Note: The I²C default input pins are I²C and SMBus compatible. RB1 and RB2 are additional pins. RC4 and RC3 are default MMP1 pins and are SMBus compatible. Clock and data signals can be routed to any pin, however pins without I²C compatibility will operate at standard TTL/ST logic levels as selected by the INVLV register.

17.4 PPS Lock

The PPS includes a mode in which all input and output selections can be locked to prevent inadvertent changes. PPS selections are locked by setting the PPSLOCKED bit of the PPSLOCK register. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. Examples of setting and clearing the PPSLOCKED bit are shown in Example 17-1.

EXAMPLE 17-1: PPS LOCK SEQUENCE

```

; Disable interrupts:
  BCF     INTCON,GIE

; Bank to PPSLOCK register
  BANKSEL PPSLOCK
  MOVLB  PPSLOCK
  MOVLW  55h

; Required sequence, next 4 instructions
  MOVWF  PPSLOCK
  MOVLW  AAh
  MOVWF  PPSLOCK

; Set PPSLOCKED bit to disable writes
; Only a BSF instruction will work
  BSF    PPSLOCK,0

; Enable Interrupts
  BSF    INTCON,GIE
    
```

EXAMPLE 17-2: PPS UNLOCK SEQUENCE

```

; Disable interrupts:
  BCF     INTCON,GIE

; Bank to PPSLOCK register
  BANKSEL PPSLOCK
  MOVLB  PPSLOCK
  MOVLW  55h

; Required sequence, next 4 instructions
  MOVWF  PPSLOCK
  MOVLW  AAh
  MOVWF  PPSLOCK

; Clear PPSLOCKED bit to enable writes
; Only a BCF instruction will work
  BCF    PPSLOCK,0

; Enable Interrupts
  BSF    INTCON,GIE
    
```

17.5 PPS One-Way Lock

Using the PPS1WAY Configuration bit, the PPS settings can be locked in. When this bit is set, the PPSLOCKED bit can only be cleared and set one time after a device Reset. This allows for clearing the PPSLOCKED bit so that the input and output selections can be made during initialization. When the PPSLOCKED bit is set after all selections have been made, it will remain set and cannot be cleared until after the next device Reset event.

17.6 Operation During Sleep

PPS input and output selections are unaffected by Sleep.

17.7 Effects of a Reset

A device Power-on-Reset (POR) clears all PPS input and output selections to their default values. All other Resets leave the selections unchanged. Default input selections are shown in the **Section “Pin Allocation Tables”**. The PPS one-way is also removed.

PIC18(L)F24/25K40

REGISTER 21-3: CCPxCAP: CAPTURE INPUT SELECTION MULTIPLEXER REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/x	R/W-0/x
—	—	—	—	—	—	CTS<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 7-2 **Unimplemented:** Read as '0'
 bit 1-0 **CTS<1:0>:** Capture Trigger Input Selection bits

CTS<1:0>	Connection	
	CCP1	CCP2
11	IOC_Interrupt	
10	CMP2_output	
01	CMP1_output	
00	Pin selected by CCP1PPS	Pin selected by CCP2PPS

REGISTER 21-4: CCPRxL: CCPx REGISTER LOW BYTE

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
CCPRx<7:0>							
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 7-0 MODE = Capture Mode:
CCPRxL<7:0>: LSB of captured TMR1 value
MODE = Compare Mode:
CCPRxL<7:0>: LSB compared to TMR1 value
MODE = PWM Mode && FMT = 0:
CCPRxL<7:0>: CCPW<7:0> – Pulse-Width LS 8 bits
MODE = PWM Mode && FMT = 1:
CCPRxL<7:6>: CCPW<1:0> – Pulse-Width LS 2 bits
CCPRxL<5:0>: Not used

FIGURE 24-6: SIMPLIFIED CWG BLOCK DIAGRAM (FORWARD AND REVERSE FULL-BRIDGE MODES)

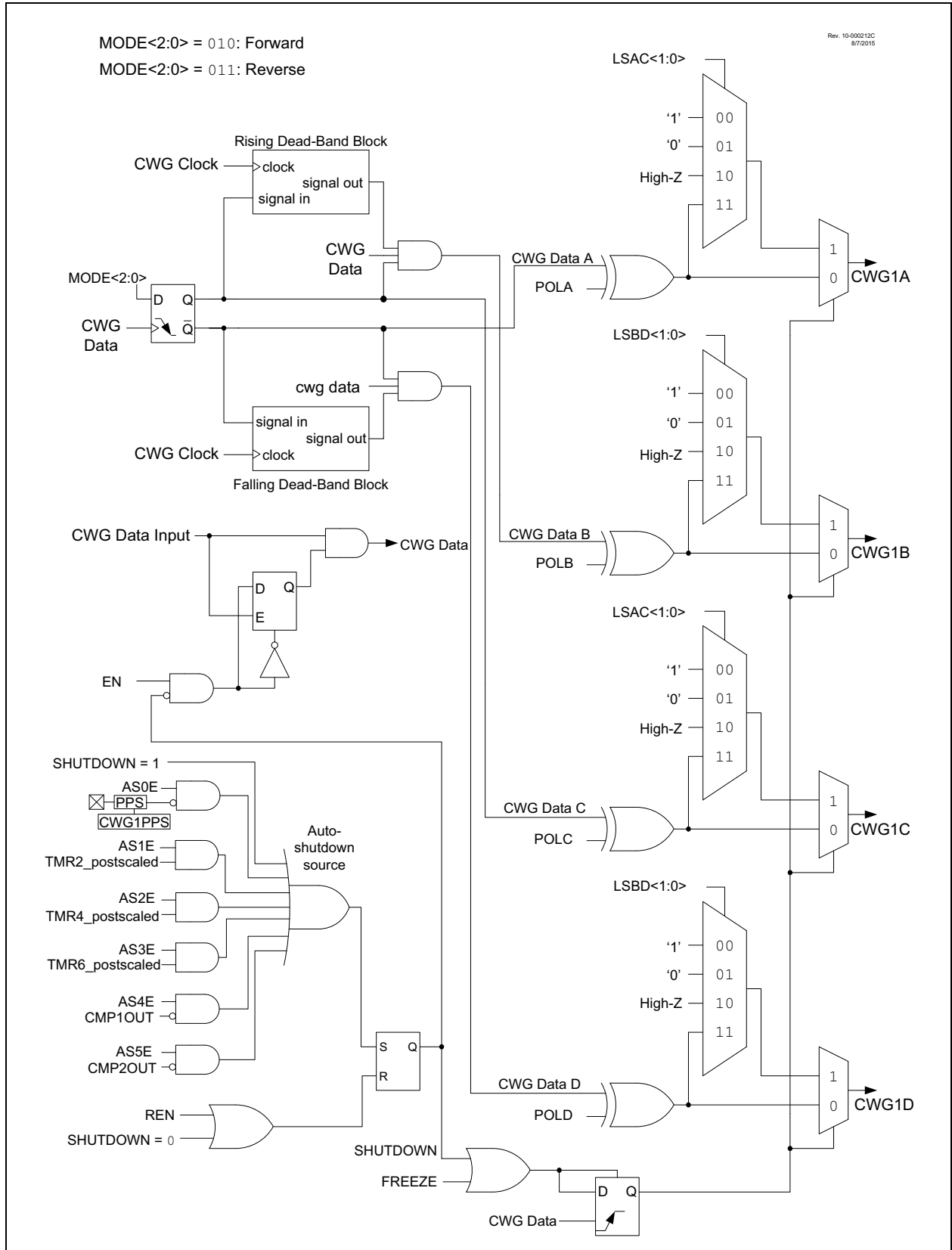
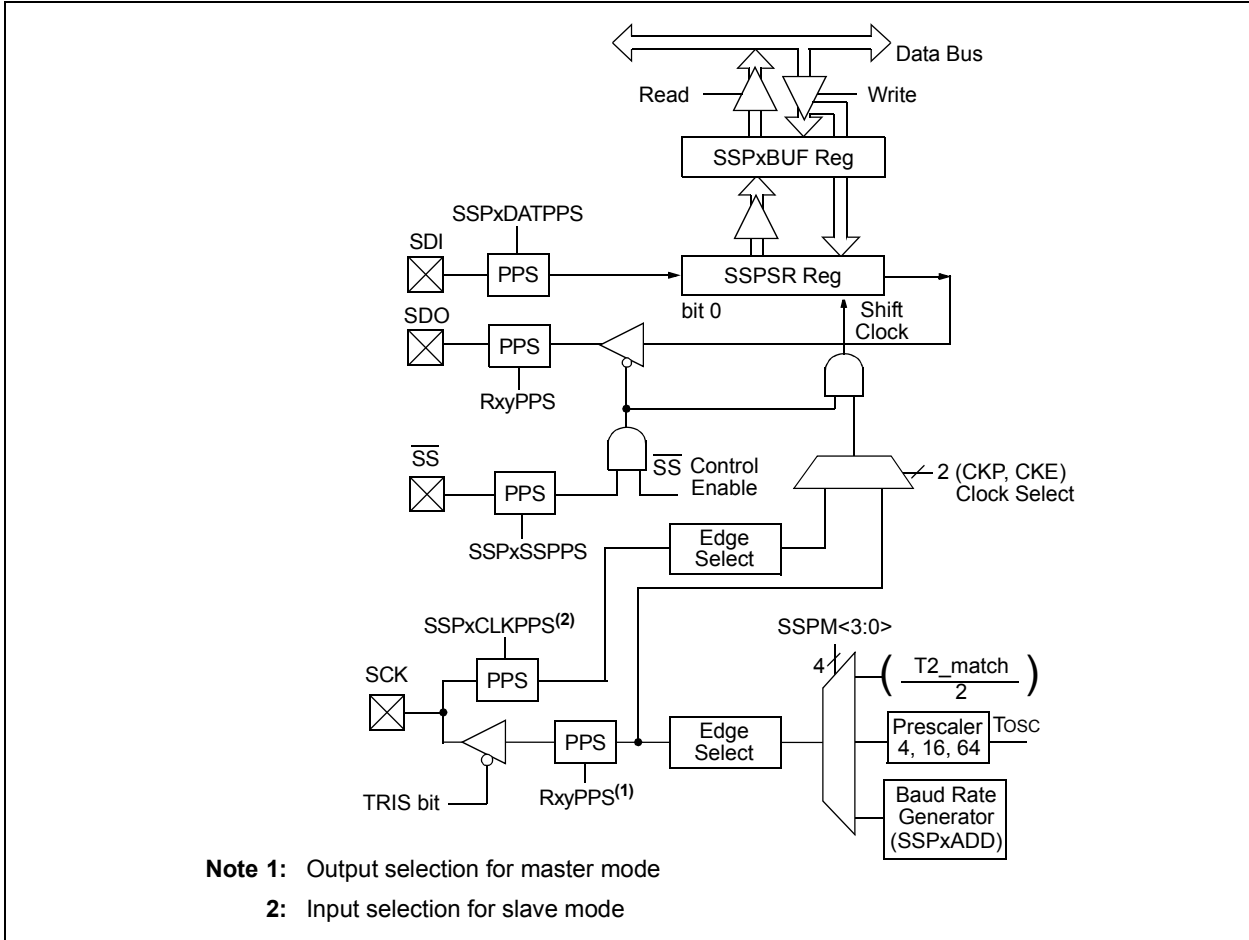


FIGURE 26-1: MSSP BLOCK DIAGRAM (SPI MODE)



The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

Figure 26-2 shows a typical connection between a master device and multiple slave devices.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

FIGURE 26-7: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

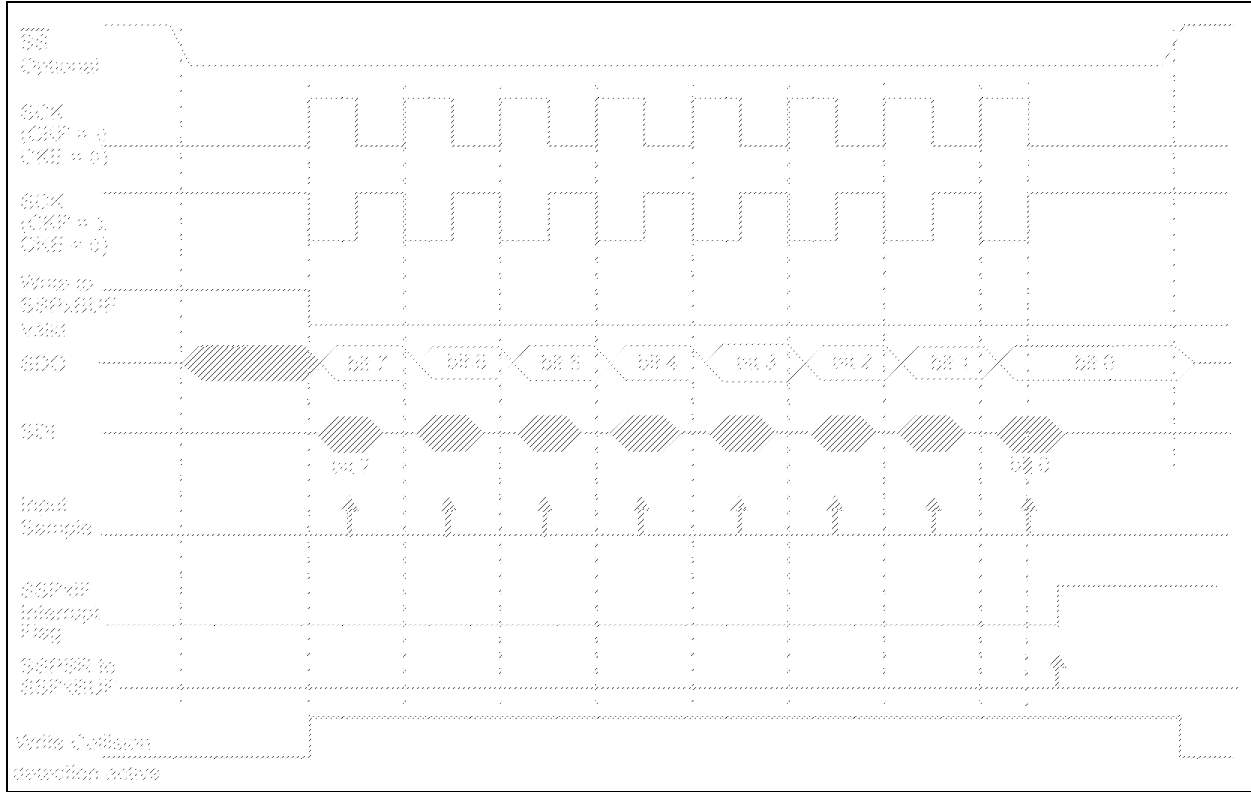
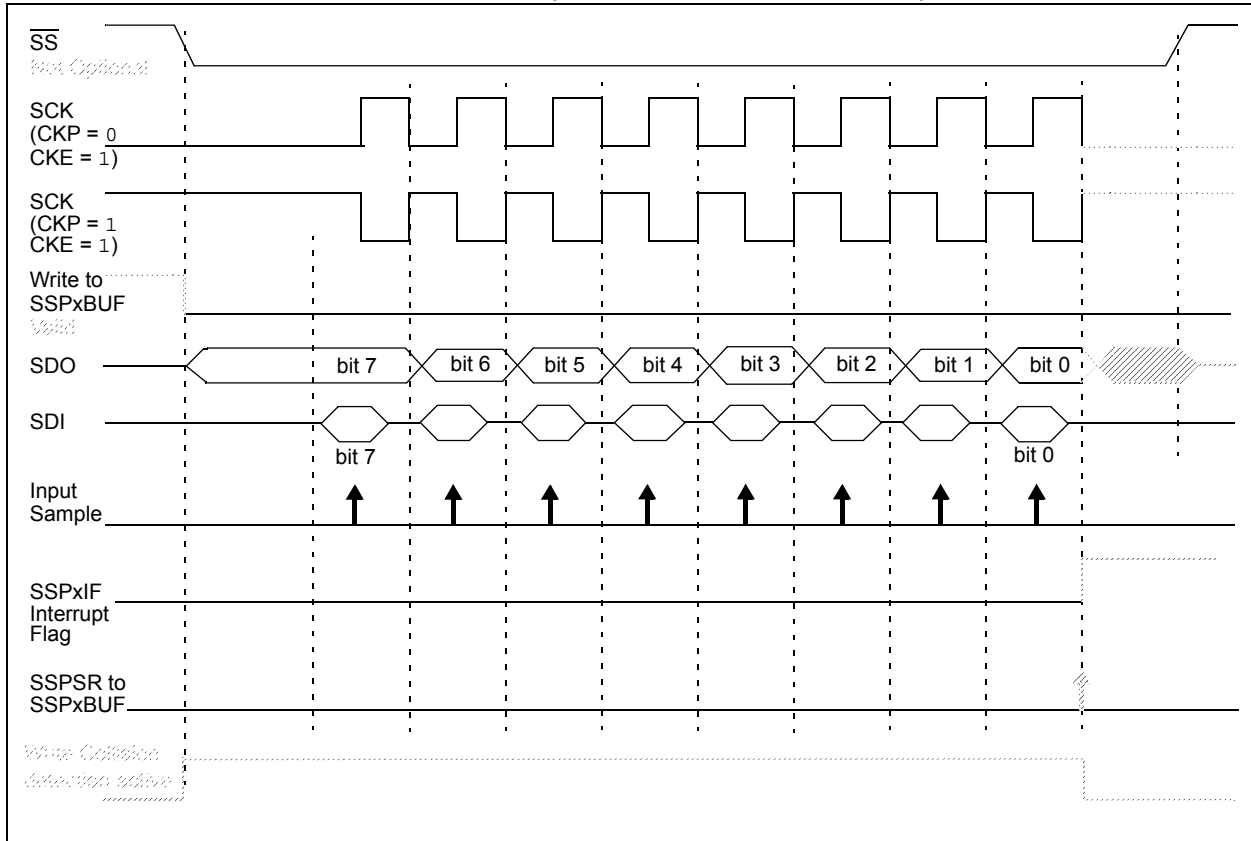


FIGURE 26-8: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



26.10 I²C Master Mode

Master mode is enabled by setting and clearing the appropriate SSPM bits in the SSPxCON1 register and by setting the SSPEN bit. In Master mode, the SDA and SCL pins must be configured as inputs. The MSSP peripheral hardware will override the output driver TRIS controls when necessary to drive the pins low.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop bit condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDA and SCL lines.

The following events will cause the SSP Interrupt Flag bit, SSPxIF, to be set (SSP interrupt, if enabled):

- Start condition detected
- Stop condition detected
- Data transfer byte transmitted/received
- Acknowledge transmitted/received
- Repeated Start generated

Note 1: The MSSP module, when configured in I²C Master mode, does not allow queuing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur

- 2: Master mode suspends Start/Stop detection when sending the Start/Stop condition by means of the SEN/PEN control bits. The SSPxIF bit is set at the end of the Start/Stop generation when hardware clears the control bit.

26.10.1 I²C MASTER MODE OPERATION

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In this case, the R/W bit will be logic '0'. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCL. See **Section 26.11 "Baud Rate Generator"** for more detail.

26.10.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 26-25).

31.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting
- Conversion Trigger Selection
- ADC Acquisition Time
- ADC Precharge Time
- Additional Sample and Hold Capacitor
- Single/Double Sample Conversion
- Guard Ring Outputs

31.1.1 PORT CONFIGURATION

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to **Section 15.0 “I/O Ports”** for more information.

Note: Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

31.1.2 CHANNEL SELECTION

There are several channel selections available:

- Eight PORTA pins (RA<7:0>)
- Eight PORTB pins (RB<7:0>)
- Eight PORTC pins (RC<7:0>)
- Temperature Indicator
- DAC output
- Fixed Voltage Reference (FVR)
- AVss (ground)

The ADPCH register determines which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion. 0

Refer to **Section 31.2 “ADC Operation”** for more information.

Note: It is recommended that when switching from an ADC channel of a higher voltage to a channel of a lower voltage, the software selects the Vss channel before switching. If the ADC does not have a dedicated Vss input channel, the Vss selection (DAC1R<4:0> = b'00000') through the DAC output channel can be used. If the DAC is in use, a free input channel can be connected to Vss, and can be used in place of the DAC.

31.1.3 ADC VOLTAGE REFERENCE

The ADPREF<1:0> bits of the ADREF register provide control of the positive voltage reference. The positive voltage reference can be:

- VREF+ pin
- VDD
- FVR 1.024V
- FVR 2.048V
- FVR 4.096V

The ADNREF bit of the ADREF register provides control of the negative voltage reference. The negative voltage reference can be:

- VREF- pin
- VSS

See **Section 28.0 “Fixed Voltage Reference (FVR)”** for more details on the Fixed Voltage Reference.

31.1.4 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCLK register and the ADCS bits of the ADCON0 register. There are 66 possible clock options:

- Fosc/2
- Fosc/4
- Fosc/6
- Fosc/8
- Fosc/10
- .
- .
- .
- Fosc/128
- FRC (dedicated RC oscillator)

The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11.5 TAD periods as shown in Figure 31-2.

For correct conversion, the appropriate TAD specification must be met. Refer to Table 37-14 for more information. Table 31-1 gives examples of appropriate ADC clock selections.

Note 1: Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.

2: The internal control logic of the ADC runs off of the clock selected by the ADCS bit of ADCON0. What this can mean is when the ADCS bit of ADCON0 is set to '1' (ADC runs on FRC), there may be unexpected delays in operation when setting ADC control bits.

31.5.1 DIGITAL FILTER/AVERAGE

The digital filter/average module consists of an accumulator with data feedback options, and control logic to determine when threshold tests need to be applied. The accumulator is a 16-bit wide register which can be accessed through the ADACCH:ADACCL register pair.

Upon each trigger event (the ADGO bit set or external event trigger), the ADC conversion result is added to the accumulator. If the accumulated value exceeds $2^{(\text{accumulator_width}-1)} = 2^{16} = 65535$, the overflow bit ADAOV in the ADSTAT register is set.

The number of samples to be accumulated is determined by the ADRPT (A/D Repeat Setting) register. Each time a sample is added to the accumulator, the ADCNT register is incremented. Once ADRPT samples are accumulated (ADCNT = ADRPT), an accumulator clear command can be issued by the software by setting the ADACLR bit in the ADCON2 register. Setting the ADACLR bit will also clear the ADAOV (Accumulator overflow) bit in the ADSTAT

register, as well as the ADCNT register. The ADACLR bit is cleared by the hardware when accumulator clearing action is complete.

Note: When ADC is operating from FRC, five FRC clock cycles are required to execute the ADACC clearing operation.

The ADCRS <2:0> bits in the ADCON2 register control the data shift on the accumulator result, which effectively divides the value in accumulator (ADACCH:ADACCL) register pair. For the Accumulate mode of the digital filter, the shift provides a simple scaling operation. For the Average/Burst Average mode, the shift bits are used to determine number of samples for averaging. For the Low-pass Filter mode, the shift is an integral part of the filter, and determines the cut-off frequency of the filter. Table 31-4 shows the -3 dB cut-off frequency in ωT (radians) and the highest signal attenuation obtained by this filter at nyquist frequency ($\omega T = \pi$).

TABLE 31-4: LOW-PASS FILTER -3 dB CUT-OFF FREQUENCY

ADCRS	ωT (radians) @ -3 dB Frequency	dB @ $F_{\text{nyquist}}=1/(2T)$
1	0.72	-9.5
2	0.284	-16.9
3	0.134	-23.5
4	0.065	-29.8
5	0.032	-36.0
6	0.016	-42.0
7	0.0078	-48.1

31.5.2 BASIC MODE

Basic mode (ADMD = 000) disables all additional computation features. In this mode, no accumulation occurs but threshold error comparison is performed. Double sampling, Continuous mode, and all CVD features are still available, but no features involving the digital filter/average features are used.

31.5.3 ACCUMULATE MODE

In Accumulate mode (ADMD = 001), after every conversion, the ADC result is added to the ADACC register. The ADACC register is right-shifted by the value of the ADCRS bits in the ADCON2 register. This right-shifted value is copied in to the ADFLT register. The Formatting mode does not affect the right-justification of the ADACC value. Upon each sample, ADCNT is also incremented, incrementing the number of samples accumulated. After each sample and accumulation, the ADACC value has a threshold comparison performed on it (see **Section 31.5.7 “Threshold Comparison”**) and the ADTIF interrupt may trigger.

31.5.4 AVERAGE MODE

In Average Mode (ADMD = 010), the ADACC registers accumulate with each ADC sample, much as in Accumulate mode, and the ADCNT register increments with each sample. The ADFLT register is also updated with the right-shifted value of the ADACC register. The value of the ADCRS bits governs the number of right shifts. However, in Average mode, the threshold comparison is performed upon ADCNT being greater than or equal to a user-defined ADRPT value. In this mode when $\text{ADRPT} = 2^{\text{ADCNT}}$, then the final accumulated value will be divided by number of samples, allowing for a threshold comparison operation on the average of all gathered samples.

PIC18(L)F24/25K40

REGISTER 32-2: CMxCON1: COMPARATOR x CONTROL REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	INTP	INTN
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 7-2 Unimplemented: Read as '0'

bit 1 **INTP**: Comparator Interrupt on Positive-Going Edge Enable bit

1 = The CxIF interrupt flag will be set upon a positive-going edge of the CxOUT bit

0 = No interrupt flag will be set on a positive-going edge of the CxOUT bit

bit 0 **INTN**: Comparator Interrupt on Negative-Going Edge Enable bit

1 = The CxIF interrupt flag will be set upon a negative-going edge of the CxOUT bit

0 = No interrupt flag will be set on a negative-going edge of the CxOUT bit

REGISTER 32-3: CMxNCH: COMPARATOR x INVERTING CHANNEL SELECT REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	NCH<2:0>		
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 7-3 **Unimplemented**: Read as '0'

bit 2-0 **NCH<2:0>**: Comparator Inverting Input Channel Select bits

111 = AVss

110 = FVR_Buffer2

101 = CxNCH not connected

100 = CxNCH not connected

011 = CxIN3-

010 = CxIN2-

001 = CxIN1-

000 = CxIN0-

PIC18(L)F24/25K40

REGISTER 33-2: HLVDCON0: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER 0

R/W-0/0	U-0	R-x	R-x	U-0	U-0	R/W-0/0	R/W-0/0
EN	—	OUT	RDY	—	—	INTH	INTL
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 **EN:** High/Low-voltage Detect Power Enable bit
 1 = Enables HLVD, powers up HLVD circuit and supporting reference circuitry
 0 = Disables HLVD, powers down HLVD and supporting circuitry
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **OUT:** HLVD Comparator Output bit
 1 = Voltage \leq selected detection limit (HLVDL<3:0>)
 0 = Voltage \geq selected detection limit (HLVDL<3:0>)
- bit 4 **RDY:** Band Gap Reference Voltages Stable Status Flag bit
 1 = Indicates HLVD Module is ready and output is stable
 0 = Indicates HLVD Module is not ready
- bit 3-2 **Unimplemented:** Read as '0'
- bit 1 **INTH:** HLVD Positive going (High Voltage) Interrupt Enable
 1 = HLVDIF will be set when voltage \geq selected detection limit (HLVDSEL<3:0>)
 0 = HLVDIF will not be set
- bit 0 **INTL:** HLVD Negative going (Low Voltage) Interrupt Enable
 1 = HLVDIF will be set when voltage \leq selected detection limit (HLVDSEL<3:0>)
 0 = HLVDIF will not be set

TABLE 33-2: REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
HLVDCON0	EN	—	OUT	RDY	—	—	INTH	INTL	476
HLVDCON1	—	—	—	—	SEL<3:0>				475
INTCON	GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	166
PIR2	HLVDIF	ZCDIF	—	—	—	—	C2IF	C1IF	169
PIE2	HLVDIE	ZCDIE	—	—	—	—	C2IE	C1IE	177
IPR2	HLVDIP	ZCDIP	—	—	—	—	C2IP	C1IP	185
PMD0	SYSCMD	FVRMD	HLVDMMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	64

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.

Note 1: PORTA<7:6> and their direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

PIC18(L)F24/25K40

RETURN Return from Subroutine

Syntax: RETURN {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → W,
(STATUS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	0000	0001	001s
------	------	------	------

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's' = 1, the contents of the shadow registers, WS, STATUS 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	Process Data	POP PC from stack	
No operation	No operation	No operation	No operation	

Example: RETURN

After Instruction:
PC = TOS

RLCF Rotate Left f through Carry

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

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

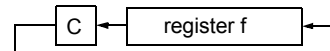
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

0011	01da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the left through the CARRY flag. If 'd' is '0', the result is placed 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: RLCF REG, 0, 0

Before Instruction

REG = 1110 0110
C = 0

After Instruction

REG = 1110 0110
W = 1100 1100
C = 1

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 f _d (destination)	2	1110	1011	0zzz	zzzz	None
MOVSS z _s , z _d	Move z _s (source) to z _d (destination)	2	1110	1011	1zzz	zzzz	None
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