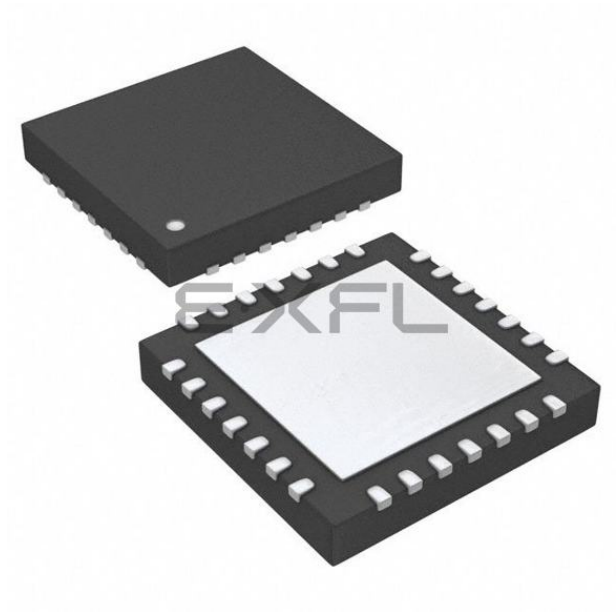


Welcome to E-XFL.COM

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, DMA, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf27k42-e-ml

4.4.2 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as either two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = 0). To maintain alignment with instruction boundaries, the PC increments in steps of two and the LSb will always read '0' (see **Section 4.2.4 "Program Counter"**).

Figure 4-2 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 4-2 shows how the instruction GOTO 0006h is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 41.0 "Instruction Set Summary"** provides further details of the instruction set.

4.4.3 MULTI-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: CALL, MOVFF, GOTO and LFSR and two three-word instructions: MOVFFL and MOVSFL. In all cases, the second and the third word of the instruction always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

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

FIGURE 4-2: INSTRUCTIONS IN PROGRAM MEMORY

Program Memory Byte Locations →

Instruction 1: MOVLW 055h
Instruction 2: GOTO 0006h
Instruction 3: MOVFF 123h, 456h
Instruction 4: MOVFFL 123h, 456h

LSB = 1	LSB = 0	Word Address ↓
		000000h
		000002h
		000004h
		000006h
0Fh	55h	000008h
EFh	03h	00000Ah
F0h	00h	00000Ch
C1h	23h	00000Eh
F4h	56h	000010h
00h	60h	000012h
F4h	8Ch	000014h
F4h	56h	000016h
		000018h
		00001Ah

REGISTER 7-7: OSCEN: OSCILLATOR MANUAL ENABLE 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	U-0	U-0
EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN	—	—

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 **EXTOEN:** External Oscillator Manual Request Enable bit
1 = EXTOSC is explicitly enabled, operating as specified by FEXTOSC
0 = EXTOSC could be enabled by requesting peripheral

bit 6 **HFOEN:** HFINTOSC Oscillator Manual Request Enable bit
1 = HFINTOSC is explicitly enabled, operating as specified by OSCFRQ (Register 7-5)
0 = HFINTOSC could be enabled by requesting peripheral

bit 5 **MFOEN:** MFINTOSC (500 kHz/31.25 kHz) Oscillator Manual Request Enable bit (Derived from HFINTOSC)
1 = MFINTOSC is explicitly enabled
0 = MFINTOSC could be enabled by requesting peripheral

bit 4 **LFOEN:** LFINTOSC (31 kHz) Oscillator Manual Request Enable bit
1 = LFINTOSC is explicitly enabled
0 = LFINTOSC could be enabled by requesting peripheral

bit 3 **SOSCEN:** Secondary Oscillator Manual Request Enable bit
1 = Secondary Oscillator is explicitly enabled, operating as specified by SOSCPWR
0 = Secondary Oscillator could be enabled by requesting peripheral

bit 2 **ADOEN:** ADC Oscillator Manual Request Enable bit
1 = ADC oscillator is explicitly enabled
0 = ADC oscillator could be enabled by requesting peripheral

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

9.3 Interrupt Priority

The final priority level for any pending source of interrupt is determined first by the user-assigned priority of that source in the IPRx register, then by the natural order priority within the IVT. The sections below detail the operation of Interrupt priorities.

9.3.1 USER (SOFTWARE) PRIORITY

User-assigned interrupt priority is enabled by setting the IPEN bit in the INTCON0 register (Register 9-1). Each peripheral interrupt source can be assigned a high or low priority level by the user. The user-assignable interrupt priority control bits for each interrupt are located in the IPRx registers (Registers 9-25 through 9-35).

The interrupts are serviced based on predefined interrupt priority scheme defined below.

1. Interrupts set by the user as high-priority interrupt have higher precedence of execution. High-priority interrupts will override a low-priority request when:
   a) A low priority interrupt has been requested or its request is already pending.
   b) A low- and high-priority interrupt are triggered concurrently, i.e., on the same instruction cycle**(1)**.
   c) A low-priority interrupt was requested and the corresponding Interrupt Service Routine is currently executing. In this case, the lower priority interrupt routine will complete executing after the high-priority interrupt has been serviced**(2)**.
2. Interrupts set by the user as a low priority have the lower priority of execution and are preempted by any high-priority interrupt.
3. Interrupts defined with the same software priority cannot preempt or interrupt each other. Concurrent pending interrupts with the same user priority are resolved using the natural order priority. (when MVECEN = ON) or in the order the interrupt flag bits are polled in the ISR (when MVECEN = OFF).

Note 1: When a high priority interrupt preempts a concurrent low priority interrupt, the GIEL bit may be cleared in the high priority Interrupt Service Routine. If the GIEL bit is cleared, the low priority interrupt will NOT be serviced even if it was originally requested. The corresponding interrupt flag needs to be cleared in user code.

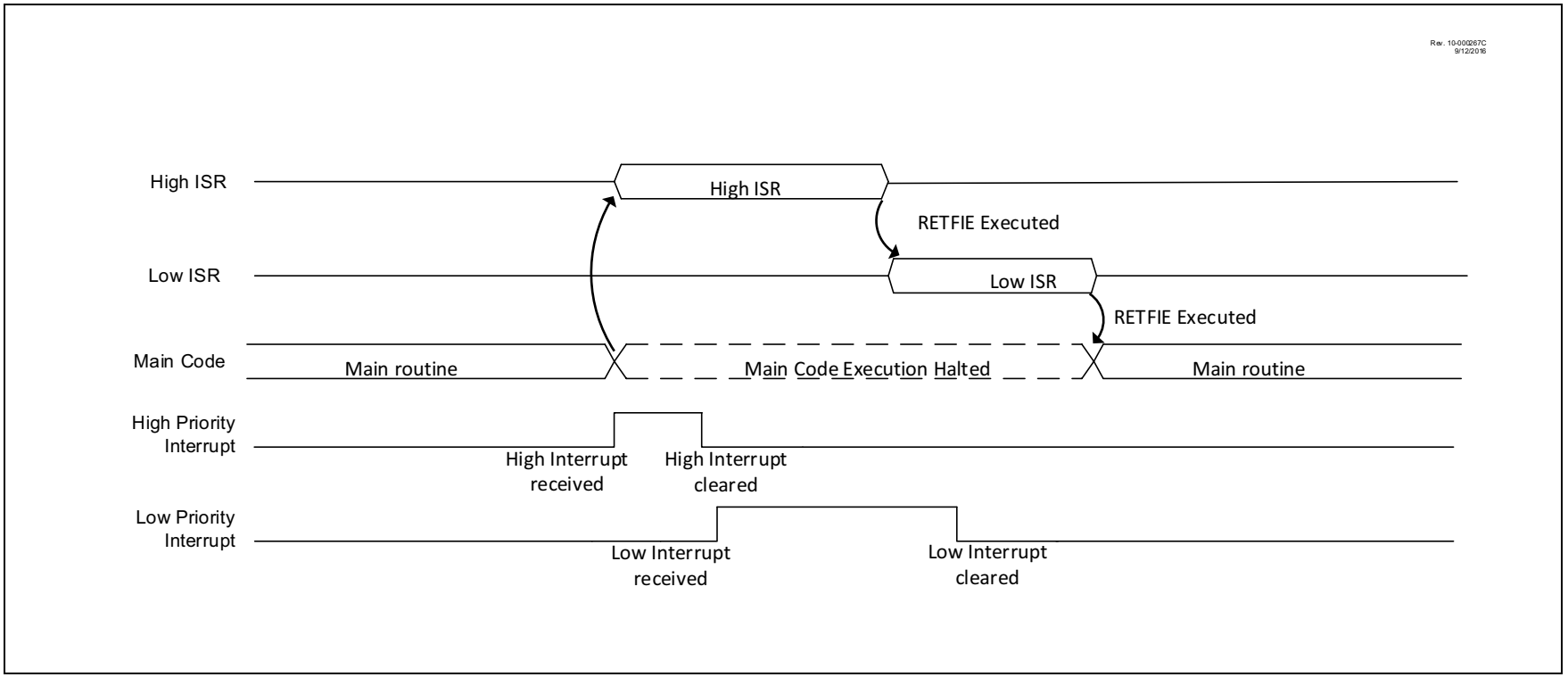
2: When a high priority interrupt is requested while a low priority Interrupt Service Routine is executing, the GIEL bit may be cleared in the high priority Interrupt Service Routine. The pending low priority interrupt will resume even if the GIEL bit is cleared.

9.4.2 SERVING A HIGH PRIORITY INTERRUPT WHILE A LOW PRIORITY INTERRUPT PENDING

A high priority interrupt request will always take precedence over any interrupt of a lower priority. The high priority interrupt is acknowledged first, then the low-priority interrupt is acknowledged. Upon a return from the high priority ISR (by executing the `RETFIE` instruction), the low priority interrupt is serviced, see Figure 9-3.

If any other high priority interrupts are pending and enabled, then they are serviced before servicing the pending low priority interrupt. If no other high priority interrupt requests are active, the low priority interrupt is serviced.

FIGURE 9-3: INTERRUPT EXECUTION: HIGH PRIORITY INTERRUPT WITH A LOW PRIORITY INTERRUPT PENDING



9.5 Context Saving

The Interrupt controller supports a two-level deep context saving (Main routine context and Low ISR context). Refer to state machine shown in Figure 9-6 for details.

The Program Counter (PC) is saved on the dedicated device PC stack. CPU registers saved include STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U.

After WREG has been saved to the context registers, the resolved vector number of the interrupt source to be serviced is copied into WREG. Context save and restore operation is completed by the interrupt controller based on current state of the interrupts and the order in which they were sent to the CPU.

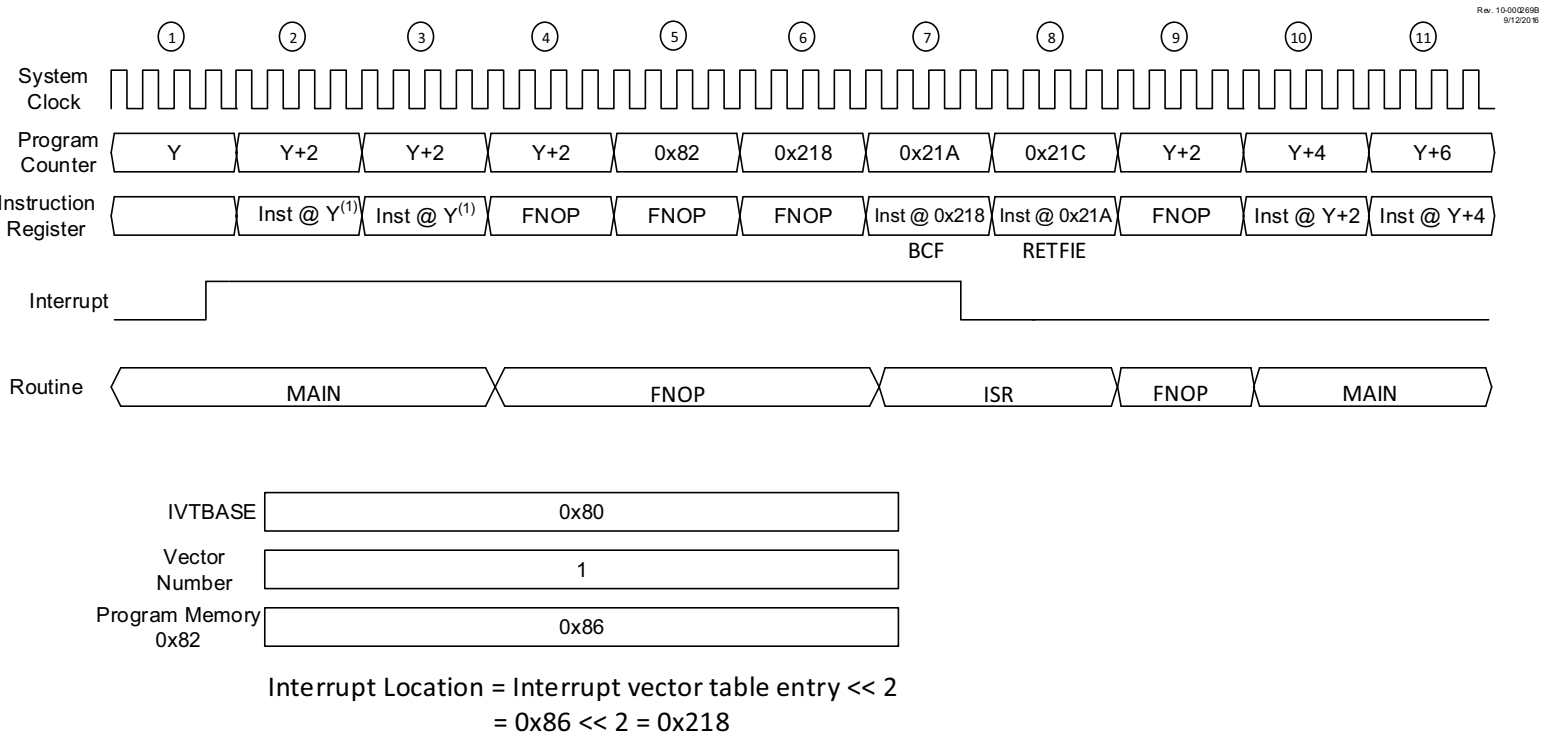
Context save/restore works the same way in both states of MVECEN. When IPEN = 0, there is only one level interrupt active. Hence, only the main context is saved when an interrupt is received.

9.5.1 ACCESSING SHADOW REGISTERS

The Interrupt controller automatically saves the context information in the shadow registers available in Bank 56. Both the saved context values (i.e., main routine and low ISR) can be accessed using the same set of shadow registers. By clearing the SHADLO bit in the SHADCON register (Register 9-43), the CPU register values saved for main routine context can accessed, and by setting the SHADLO bit of the CPU register, values saved for low ISR context can accessed. Low ISR context is automatically restored to the CPU registers upon exiting the high ISR. Similarly, the main context is automatically restored to the CPU registers upon exiting the low ISR.

The Shadow registers in Bank 56 are readable and writable, so if the user desires to modify the context, then the corresponding shadow register should be modified and the value will be restored when exiting the ISR. Depending on the user's application, other registers may also need to be saved.

FIGURE 9-8: INTERRUPT TIMING DIAGRAM - TWO WORD INSTRUCTION



Note 1: Instruction @ Y is a Two-cycle instruction.

REGISTER 9-11: PIR8: PERIPHERAL INTERRUPT REGISTER 8(1)

R/W/HS-0/0	R/W/HS-0/0	U-0	U-0	U-0	U-0	U-0	U-0
TMR5GIF	TMR5IF	—	—	—	—	—	—
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 = Bit is set in hardware

bit 7 **TMR5GIF:** TMR5 Gate Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 6 **TMR5IF:** TMR5 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 5-0 **Unimplemented:** Read as '0'

Note 1: Interrupt flag bits get 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.

REGISTER 9-12: PIR9: PERIPHERAL INTERRUPT REGISTER 9(1)

U-0	U-0	U-0	U-0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
—	—	—	—	CLC3IF	CWG3IF	CCP3IF	TMR6IF
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 7-4 **Unimplemented:** Read as '0'

bit 3 **CLC3IF:** CLC3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **CWG3IF:** CWG3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **CCP3IF:** CCP3 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **TMR6IF:** TMR6 Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get 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.

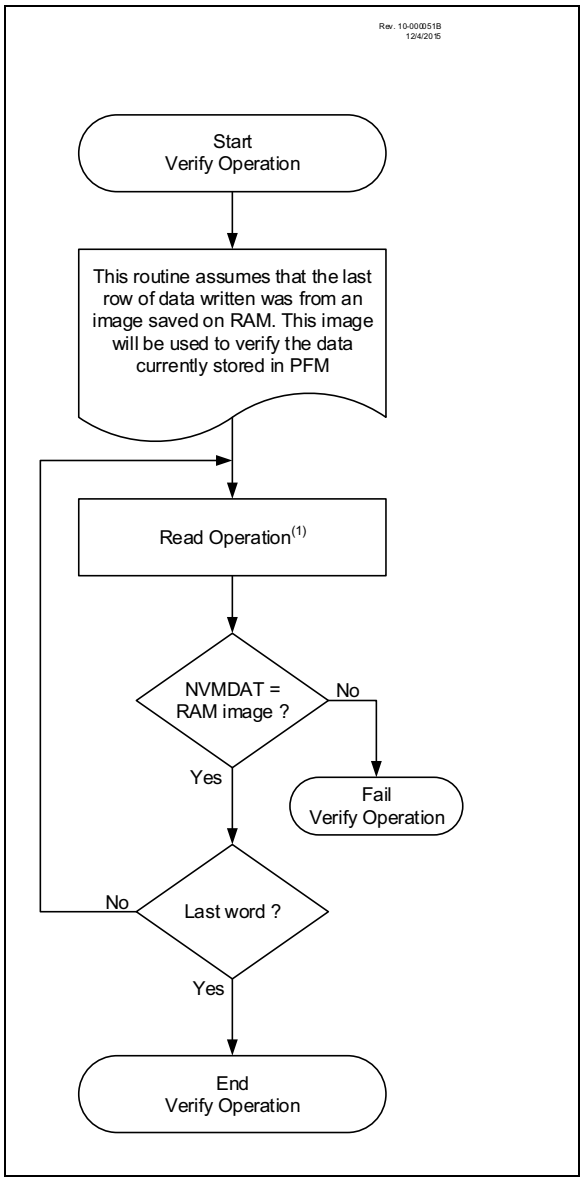
EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY

```
                MOVLW     D'64'                 ; number of bytes in erase block
                MOVWF     COUNTER
                MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                MOVWF     FSR0H
                MOVLW     BUFFER_ADDR_LOW
                MOVWF     FSR0L
                MOVLW     CODE_ADDR_UPPER       ; Load TBLPTR with the base
                MOVWF     TBLPTRU               ; address of the memory block
                MOVLW     CODE_ADDR_HIGH
                MOVWF     TBLPTRH
                MOVLW     CODE_ADDR_LOW
                MOVWF     TBLPTRL
READ_BLOCK
                TBLRD*+                         ; read into TABLAT, and inc
                MOVF      TABLAT, W             ; get data
                MOVWF     POSTINC0              ; store data
                DECFSZ    COUNTER               ; done?
                BRA       READ_BLOCK            ; repeat
MODIFY_WORD
                MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                MOVWF     FSR0H
                MOVLW     BUFFER_ADDR_LOW
                MOVWF     FSR0L
                MOVLW     NEW_DATA_LOW          ; update buffer word
                MOVWF     POSTINC0
                MOVLW     NEW_DATA_HIGH
                MOVWF     INDF0
ERASE_BLOCK
                MOVLW     CODE_ADDR_UPPER       ; load TBLPTR with the base
                MOVWF     TBLPTRU               ; address of the memory block
                MOVLW     CODE_ADDR_HIGH
                MOVWF     TBLPTRH
                MOVLW     CODE_ADDR_LOW
                MOVWF     TBLPTRL
                BCF       NVMCON1, REG0         ; point to Program Flash Memory
                BSF       NVMCON1, REG1         ; point to Program Flash Memory
                BSF       NVMCON1, WREN         ; enable write to memory
                BSF       NVMCON1, FREE         ; enable Erase operation
                BCF       INTCON0, GIE          ; disable interrupts
                MOVLW     55h
Required        MOVWF     NVMCON2               ; write 55h
Sequence        MOVLW     AAh
                MOVWF     NVMCON2               ; write 0AAh
                BSF       NVMCON1, WR           ; start erase (CPU stall)
                BSF       INTCON0, GIE          ; re-enable interrupts
                TBLRD*-                         ; dummy read decrement
                MOVLW     BUFFER_ADDR_HIGH      ; point to buffer
                MOVWF     FSR0H
                MOVLW     BUFFER_ADDR_LOW
                MOVWF     FSR0L
WRITE_BUFFER_BACK
                MOVLW     BlockSize             ; number of bytes in holding register
                MOVWF     COUNTER
                MOVLW     D'64'/BlockSize       ; number of write blocks in 64 bytes
                MOVWF     COUNTER2
```

13.1.6.2 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit. Since program memory is stored as a full page, the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

FIGURE 13-10: PROGRAM FLASH MEMORY VERIFY FLOWCHART



13.1.6.3 Unexpected Termination of Write Operation

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. If the write operation is interrupted by a MCLR Reset or a WDT Time-out Reset during normal operation, the WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.

13.1.6.4 Protection Against Spurious Writes

A write sequence is valid only when both the following conditions are met, this prevents spurious writes which might lead to data corruption.

1. The WR bit is gated through the WREN bit. It is suggested to have the WREN bit cleared at all times except during memory writes. This prevents memory writes if the WR bit gets set accidentally.
2. The NVM unlock sequence must be performed each time before a write operation.

13.2 Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access

When REG<1:0> = 0b01 or 0b11 in the NVMCON1 register, the Device Information Area, the Device Configuration Area, the User IDs, Device ID/ Revision ID and Configuration Words can be accessed. Different access may exist for reads and writes (see Table 13-1).

13.2.1 Reading Access

The user can read from these blocks by setting the REG bits to 0b01 or 0b11. The user needs to load the address into the TBLPTR registers. Executing a TBLRD after that moves the byte pointed to the TABLAT register. The CPU operation is suspended during the read and resumes after. When read access is initiated on an address outside the parameters listed in Table 13-1, the TABLAT register is cleared, reading back ‘0’s.

13.2.2 Writing Access

The WREN bit in NVMCON1 must be set to enable writes. This prevents accidental writes to the CONFIG words due to errant (unexpected) code execution. The WREN bit should be kept clear at all times, except when updating the CONFIG words. The WREN bit is not cleared by hardware. The WR bit will be inhibited from being set unless the WREN bit is set.

14.8 Scanner Module Overview

The Scanner allows segments of the Program Flash Memory or Data EEPROM, to be read out (scanned) to the CRC Peripheral. The Scanner module interacts with the CRC module and supplies it data one word at a time. Data is fetched from the address range defined by SCANLADR registers up to the SCANHADR registers.

The Scanner begins operation when the SGO bit is set (SCANCON0 Register) and ends when either SGO is cleared by the user or when SCANLADR increments past SCANHADR. The SGO bit is also cleared by clearing the EN bit (CRCCON0 register).

14.9 Configuring the Scanner

The scanner module may be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory or Data EEPROM addresses. In order to set up the scanner to work with the CRC, perform the following steps:

1. Set up the CRC module (See **Section 14.7 "Configuring the CRC"**) and enable the Scanner module by setting the EN bit in the SCANCON0 register.
2. Choose which memory region the Scanner module should operate on and set the MREG bit of the SCANCON0 register appropriately.
3. If trigger is used for scanner operation, set the TRIGEN bit of the SCANCON0 register and select the trigger source using SCANTRIG register. Select the trigger source using SCANTRIG register and then set the TRIGEN bit of the SCANCON0 register. See Table 14-1 for Scanner Operation.
4. If Burst mode of operation is desired, set the BURSTMD bit (SCANCON0 register). See Table 14-1 for Scanner Operation.
5. Set the SCANLADRL/H/U and SCANHADRL/H/U registers with the beginning and ending locations in memory that are to be scanned.
6. Select the priority level for the Scanner module (See **Section 3.1 "System Arbitration"**) and lock the priorities (See **Section 3.1.1 "Priority Lock"**).
7. Both CRCEN and CRCGO bits must be enabled to use the scanner. Setting the SGO bit will start the scanner operation.

14.10 Scanner Interrupt

The scanner will trigger an interrupt when the SCANLADR increments past SCANHADR. The SCANIF bit can only be cleared in software.

14.11 Scanning Modes

The interaction of the scanner with the system operation is controlled by the priority selection in the System Arbiter (see **Section 3.2 "Memory Access Scheme"**). Additionally, BURSTMD and TRIGEN also determine the operation of the Scanner.

14.11.1 TRIGEN = 0, BURSTMD = 0

In this case, the memory access request is granted to the scanner if no other higher priority source is requesting access.

All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

14.11.2 TRIGEN = 1, BURSTMD = 0

In this case, the memory access request is generated when the CRC module is ready to accept.

The memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

The memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

14.11.3 TRIGEN = x, BURSTMD = 1

In this case, the memory access is always requested by the scanner.

The memory access request is granted to the scanner if no other higher priority source is requesting access. The memory access cycles will not be granted to lower priority sources than the scanner until it completes operation i.e. SGO = 0 (SCANCON0 register)

Note:	If TRIGEN = 1 and BURSTMD = 1, the user should ensure that the trigger source is active for the Scanner operation to complete.

16.0 I/O PORTS

The PIC18(L)F26/27/45/46/47/55/56/57K42 devices have six I/O ports, allocated as shown in Table 16-1.

TABLE 16-1: PORT ALLOCATION TABLE FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

Device	PORTA	PORTB	PORTC	PORTD	PORTE	PORTF
PIC18(L)F26K42	•	•	•		•(1)	
PIC18(L)F27K42	•	•	•		•(1)	
PIC18(L)F45K42	•	•	•	•	•(2)	
PIC18(L)F46K42	•	•	•	•	•(2)	
PIC18(L)F47K42	•	•	•	•	•(2)	
PIC18(L)F55K42	•	•	•	•	•(2)	•
PIC18(L)F56K42	•	•	•	•	•(2)	•
PIC18(L)F57K42	•	•	•	•	•(2)	•

Note 1: Pin RE3 only.
2: Pins RE0, RE1, RE2 and RE3 only.

Each port has ten registers to control the operation. These registers are:

- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)
- TRISx registers (data direction)
- ANSELx registers (analog select)
- WPUx registers (weak pull-up)
- INLVLx (input level control)
- SLRCONx registers (slew rate control)
- ODCONx registers (open-drain control)

Most port pins share functions with device peripherals, both analog and digital. In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output; however, the pin can still be read.

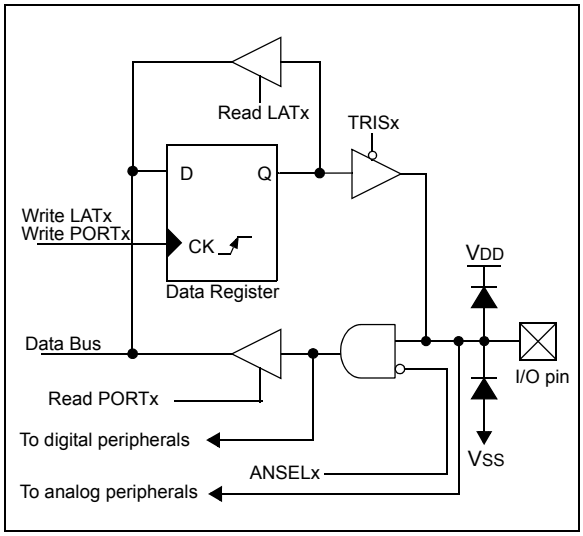
The Data Latch (LATx registers) is useful for read-modify-write operations on the value that the I/O pins are driving.

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSELx bit is set, the digital input buffer associated with that bit is disabled.

Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 16-1.

FIGURE 16-1: GENERIC I/O PORT OPERATION



16.1 I/O Priorities

Each pin defaults to the PORT data latch after Reset. Other functions are selected with the peripheral pin select logic. See **Section 17.0 "Peripheral Pin Select (PPS) Module"** for more information.

Analog input functions, such as ADC and comparator inputs, are not shown in the peripheral pin select lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx register. Digital output functions may continue to control the pin when it is in Analog mode.

Analog outputs, when enabled, take priority over digital outputs and force the digital output driver into a high-impedance state.

The pin function priorities are as follows:

1. Configuration bits
2. Analog outputs (disable the input buffers)
3. Analog inputs
4. Port inputs and outputs from PPS

16.2 PORTx Registers

In this section, the generic names such as PORTx, LATx, TRISx, etc. can be associated with PORTA, PORTB, and PORTC. The functionality of PORTE is different compared to other ports and is explained in a separate section.

REGISTER 19-8: PMD7: PMD CONTROL REGISTER 7

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	DMA2MD	DMA1MD
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 7-2 **Unimplemented:** Read as '0'

bit 1 **DMA2MD:** Disable DMA2 Module bit
1 = DMA2 module disabled
0 = DMA2 module enabled

bit 0 **DMA1MD:** Disable DMA1 Module bit
1 = DMA1 module disabled
0 = DMA1 module enabled

TABLE 19-1: SUMMARY OF REGISTERS ASSOCIATED WITH PERIPHERAL MODULE DISABLE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PMD0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	290
PMD1	NCO1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD	291
PMD2	—	DACMD	ADCMD	—	—	CMP2MD	CMP1MD	ZCDMD	292
PMD3	PWM8MD	PWM7MD	PWM6MD	PWM5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	293
PMD4	CWG3MD	CWG2MD	CWG1MD	—	—	—	—	—	294
PMD5	—	—	U2MD	U1MD	—	SPI1MD	I2C2MD	I2C1MD	295
PMD6	—	—	SMT1MD	CLC4MD	CLC3MD	CLC2MD	CLC1MD	DSMMD	295
PMD7	—	—	—	—	—	—	DMA2MD	DMA1MD	297

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by peripheral module disable.

REGISTER 21-4: TxGATE: TIMERx GATE ISM REGISTER

U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	GSS<4:0>				
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	u = unchanged

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **GSS<4:0>:** Timerx Gate Source Selection bits

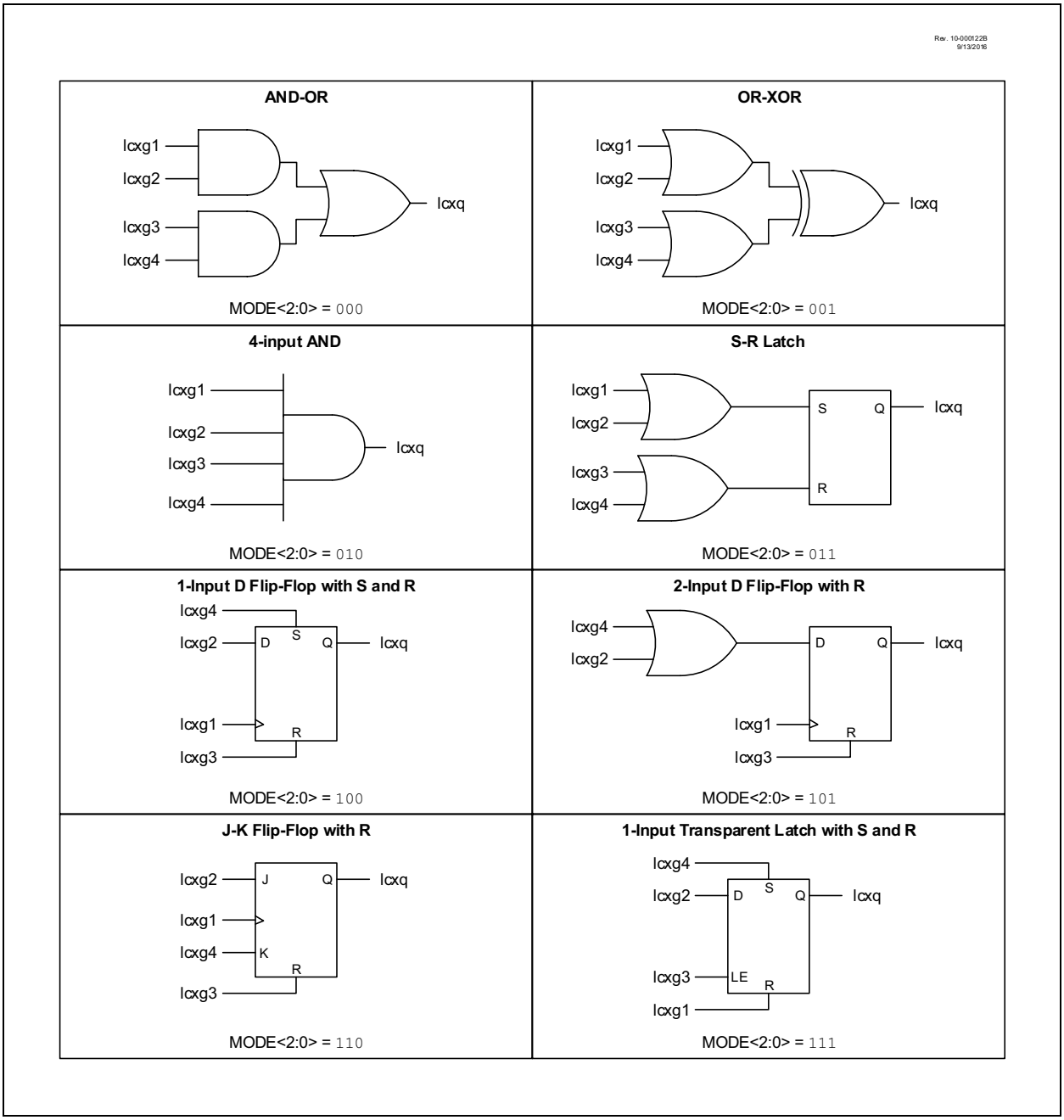
GSS	Timer1	Timer3	Timer5
	Gate Source	Gate Source	Gate Source
11111-11011	Reserved	Reserved	Reserved
11010	CLC4_out	CLC4_out	CLC4_out
11001	CLC3_out	CLC3_out	CLC3_out
11000	CLC2_out	CLC2_out	CLC2_out
10111	CLC1_out	CLC1_out	CLC1_out
10110	ZCDOUT	ZCDOUT	ZCDOUT
10101	CMP2OUT	CMP2OUT	CMP2OUT
10100	CMP1OUT	CMP1OUT	CMP1OUT
10011	NCO1OUT	NCO1OUT	NCO1OUT
10010-10001	Reserved	Reserved	Reserved
10000	PWM8OUT	PWM8OUT	PWM8OUT
01111	PWM7OUT	PWM7OUT	PWM7OUT
01110	PWM6OUT	PWM6OUT	PWM6OUT
01101	PWM5OUT	PWM5OUT	PWM5OUT
01100	CCP4OUT	CCP4OUT	CCP4OUT
01011	CCP3OUT	CCP3OUT	CCP3OUT
01010	CCP2OUT	CCP2OUT	CCP2OUT
01001	CCP1OUT	CCP1OUT	CCP1OUT
01000	SMT1_match	SMT1_match	SMT1_match
00111	TMR6OUT (postscaled)	TMR6OUT (postscaled)	TMR6OUT (postscaled)
00110	TMR5 overflow	TMR5 overflow	Reserved
00101	TMR4OUT (postscaled)	TMR4OUT (postscaled)	TMR4OUT (postscaled)
00100	TMR3 overflow	Reserved	TMR3 overflow
00011	TMR2OUT (postscaled)	TMR2OUT (postscaled)	TMR2OUT (postscaled)
00010	Reserved	TMR1 overflow	TMR1 overflow
00001	TMR0 overflow	TMR0 overflow	TMR0 overflow
00000	Pin selected by T1GPPS	Pin selected by T3GPPS	Pin selected by T5GPPS

TABLE 22-1: TIMER2 OPERATING MODES

Mode	MODE<4:0>		Output Operation	Operation	Timer Control		
	<4:3>	<2:0>			Start	Reset	Stop
Free Running Period	00	000	Period Pulse	Software gate (Figure 22-6)	ON = 1	—	ON = 0
		001		Hardware gate, active-high (Figure 22-7)	ON = 1 & TMRx_ers = 1	—	ON = 0 or TMRx_ers = 0
		010		Hardware gate, active-low	ON = 1 & TMRx_ers = 0	—	ON = 0 or TMRx_ers = 1
		011	Period Pulse with Hardware Reset	Rising or Falling Edge Reset	ON = 1	TMRx_ers ↕	ON = 0
		100		Rising Edge Reset (Figure 22-8)		TMRx_ers ↑	
		101		Falling Edge Reset		TMRx_ers ↓	
		110		Low Level Reset		TMRx_ers = 0	ON = 0 or TMRx_ers = 0
		111		High Level Reset (Figure 22-9)		TMRx_ers = 1	ON = 0 or TMRx_ers = 1
One-shot	01	000	One-Shot	Software Start (Figure 22-10)	ON = 1	—	ON = 0 or Next clock after TMRx = PRx **(Note 2)**
		001	Edge Triggered Start (Note 1)	Rising Edge Start (Figure 22-9)	ON = 1 & TMRx_ers ↑	—	
		010		Falling Edge Start	ON = 1 & TMRx_ers ↓	—	
		011		Any eEdge Start	ON = 1 & TMRx_ers ↕	—	
		100	Edge Triggered Start and Hardware Reset (Note 1)	Rising Edge Start & Rising Edge Reset (Figure 22-12)	ON = 1 & TMRx_ers ↑	TMRx_ers ↑	
		101		Falling Edge Start & Falling Edge Reset	ON = 1 & TMRx_ers ↓	TMRx_ers ↓	
		110		Rising Edge Start & Low Level Reset (Figure 22-13)	ON = 1 & TMRx_ers ↑	TMRx_ers = 0	
		111		Falling Edge Start & High Level Reset	ON = 1 & TMRx_ers ↓	TMRx_ers = 1	
Monostable	10	000	Reserved				
		001	Edge Triggered Start (Note 1)	Rising Edge Start (Figure 22-12)	ON = 1 & TMRx_ers ↑	—	ON=0 or Next clock after TxTMR = TxPR **(Note 3)**
		010		Falling Edge Start	ON = 1 & TMRx_ers ↓	—	
		011		Any Edge Start	ON = 1 & TMRx_ers ↕	—	
Reserved		100	Reserved				
Reserved		101	Reserved				
One-shot		110	Level Triggered Start and Hardware Reset	High Level Start & Low Level Reset (Figure 22-13)	ON = 1 & TMRx_ers = 1	TMRx_ers = 0	ON = 0 or Held in Reset **(Note 2)**
		111		Low Level Start & High Level Reset	ON = 1 & TMRx_ers = 0	TMRx_ers = 1	
Reserved	11	xxx	Reserved				

Note 1: If ON = 0 then an edge is required to restart the timer after ON = 1.
2: When TxTMR = TxPR then the next clock clears ON and stops TxTMR at 00h.
3: When TxTMR = TxPR then the next clock stops TxTMR at 00h but does not clear ON.

FIGURE 27-3: PROGRAMMABLE LOGIC FUNCTIONS



35.2 Temperature Calculation

This section describes the steps involved in calculating the die temperature, TMEAS:

1. Obtain the ADC count value of the measured analog voltage: The analog output voltage, VMEAS is converted to a digital count value by the Analog to Digital Converter (ADC) and is referred to as ADCMEAS.
2. Obtain the ADC count value, ADCDIA at 90 degrees, from the DIA table. This parameter is TSLR2 for the low range setting or TSHR2 for the high range setting of the temperature indicator module.
3. Obtain the output analog voltage (in mV) value of the Fixed Reference Voltage (FVR) for 2x setting, from the DIA Table. This parameter is FVRA2X in the DIA table (Table 5-3).
4. Obtain the value of the temperature indicator voltage sensitivity, parameter Mv, from Table 44-27 for the corresponding range setting.

Equation 35-1 provides an estimate for the die temperature based on the above parameters.

EQUATION 35-1: SENSOR TEMPERATURE

$$T_{MEAS} = 90 + \frac{(ADC_{MEAS} - ADC_{DIA}) \times FVRA2X}{(2^N - 1) \times Mv}$$

Where:
ADCMEAS = ADC reading at temperature being estimated
ADCDIA = ADC reading stored in the DIA
FVRA2X = FVR value stored in the DIA for 2x setting
N = Resolution of the ADC
Mv = Temperature Indicator voltage sensitivity (mV/°C)

Note: It is recommended to take the average of 10 measurements of ADCmeas to reduce noise and improve accuracy.

35.2.1 CALIBRATION

35.2.1.1 Higher-Order Calibration

If the application requires more precise temperature measurement, additional calibrations steps will be necessary. For these applications, two-point or three-point calibration is recommended.

35.2.2 TEMPERATURE RESOLUTION

The resolution of the ADC reading, Ma (°C/count), depends on both the ADC resolution N and the reference voltage used for conversion, as shown in Equation 35-2. It is recommended to use the smallest VREF value, such as the ADC FVR1 Output Voltage for 2x setting (FVRA2X) value from the DIA. Refer to Table 5-3 for DIA location.

Note: Refer to Table 44-19 for FVR reference voltage accuracy.

35.3 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait a certain minimum acquisition time (parameter TS01 in Table 44-27) for the ADC value to settle, after the ADC input multiplexer is connected to the temperature indicator output, before the conversion is performed.

TABLE 35-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR(1)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
FVRCON	EN	RDY	TSEN	TSRNG	CDAFVR<1:0>		ADFVR<1:0>		597

Legend: — = Unimplemented location, read as '0'. Shaded cells are unused by the temperature indicator module.
Note 1: It is recommended to take the average of ten measurements of ADCMEAS to reduce noise and improve accuracy.

REGISTER 36-29: ADERRH: ADC SETPOINT ERROR REGISTER HIGH

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ERR<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 **ERR<15:8>**: ADC Setpoint Error MSB. Upper byte of ADC Setpoint Error. Setpoint Error calculation is determined by CALC bits of ADCON3, see Register 36-4 for more details.

REGISTER 36-30: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ERR<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 **ERR<7:0>**: ADC Setpoint Error LSB. Lower byte of ADC Setpoint Error calculation is determined by CALC bits of ADCON3, see Register 36-4 for more details.

REGISTER 36-31: ADLTHH: ADC LOWER THRESHOLD HIGH 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
LTH<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 **LTH<15:8>**: ADC Lower Threshold MSB. LTH and UTH are compared with ERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

39.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18(L)F26/27/45/46/47/55/56/57K42 family of devices has a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point (positive going, negative going or both). If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

Complete control of the HLVD module is provided through the HLVDCON0 and HLVDCON1 register. This allows the circuitry to be “turned off” by the user under software control, which minimizes the current consumption for the device.

The module’s block diagram is shown in Figure 39-1.

Since the HLVD can be software enabled through the EN bit, setting and clearing the enable bit does not produce a false HLVD event glitch. Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The RDY bit (HLVDCON0<4>) is a read-only bit used to indicate when the band gap reference voltages are stable.

The module can only generate an interrupt after the module is turned ON and the band gap reference voltages are ready.

The INTH and INTL bits determine the overall operation of the module. When INTH is set, the module monitors for rises in V_{DD} above the trip point set by the HLVDCON1 register. When INTL is set, the module monitors for drops in V_{DD} below the trip point set by the HLVDCON1 register. When both the INTH and INTL bits are set, any changes above or below the trip point set by the HLVDCON1 register can be monitored.

The OUT bit can be read to determine if the voltage is greater than or less than the voltage level selected by the HLVDCON1 register.

MOVF **Move f**

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

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

Operation: f → 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 ≤ 95 (5Fh). See **Section 41.2.3 "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

MOVFF **Move f to f**

Syntax: MOVFF f_s,f_d

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	1100	ffff	ffff	$ffff_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh.
`MOVFF` has curtailed the source and destination range to the lower 4 Kbyte space of memory (Banks 1 through 15). For everything else, use `MOVFFL`.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: `MOVFF REG1, REG2`

Before Instruction
REG1 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3BC8h - 3AEBh	—	Unimplemented								
3AEAh	U2CTSPPS	—	—	—	U2CTSPPS					277
3AE8h	U2RXPPS	—	—	—	U2RXPPS					277
3AE7h	U1CTSPPS	—	—	—	U1CTSPPS					277
3AE5h	U1RXPPS	—	—	—	U1RXPPS					277
3AE4h	I2C2SDAPPS	—	—	—	I2C2SDAPPS					277
3AE3h	I2C2SCLPPS	—	—	—	I2C2SCLPPS					277
3AE2h	I2C1SDAPPS	—	—	—	I2C1SDAPPS					277
3AE1h	I2C1SCLPPS	—	—	—	I2C1SCLPPS					277
3AE0h	SPI1SSPPS	—	—	—	SPI1SSPPS					277
3ADFh	SPI1SDIPPS	—	—	—	SPI1SDIPPS					277
3ADEh	SPI1SCKPPS	—	—	—	SPI1SCKPPS					277
3ADDh	ADACTPPS	—	—	—	ADACTPPS					277
3ADCh	CLCIN3PPS	—	—	—	CLCIN3PPS					277
3ADBh	CLCIN2PPS	—	—	—	CLCIN2PPS					277
3ADAh	CLCIN1PPS	—	—	—	CLCIN1PPS					277
3AD9h	CLCIN0PPS	—	—	—	CLCIN0PPS					277
3AD8h	MD1SRCPPS	—	—	—	MD1SRCPPS					277
3AD7h	MD1CARHPPS	—	—	—	MD1CARHPPS					277
3AD6h	MD1CARLPPS	—	—	—	MD1CARLPPS					277
3AD5h	CWG3INPPS	—	—	—	CWG3INPPS					277
3AD4h	CWG2INPPS	—	—	—	CWG2INPPS					277
3AD3h	CWG1INPPS	—	—	—	CWG1INPPS					277
3AD2h	SMT1SIGPPS	—	—	—	SMT1SIGPPS					277
3AD1h	SMT1WINPPS	—	—	—	SMT1WINPPS					277
3AD0h	CCP4PPS	—	—	—	CCP4PPS					277
3ACFh	CCP3PPS	—	—	—	CCP3PPS					277
3ACEh	CCP2PPS	—	—	—	CCP2PPS					277
3ACDh	CCP1PPS	—	—	—	CCP1PPS					277
3ACCh	T6INPPS	—	—	—	T6INPPS					277
3ACBh	T4INPPS	—	—	—	T4INPPS					277
3ACAh	T2INPPS	—	—	—	T2INPPS					277
3AC9h	T5GPPS	—	—	—	T5GPPS					277
3AC8h	T5CLKIPPS	—	—	—	T5CLKIPPS					277
3AC7h	T3GPPS	—	—	—	T3GPPS					277
3AC6h	T3CLKIPPS	—	—	—	T3CLKIPPS					277
3AC5h	T1GPPS	—	—	—	T1GPPS					277
3AC4h	T1CLKIPPS	—	—	—	T1CLKIPPS					277
3AC3h	T0CLKIPPS	—	—	—	T0CLKIPPS					277
3AC2h	INT2PPS	—	—	—	INT2PPS					277
3AC1h	INT1PPS	—	—	—	INT1PPS					277
3AC0h	INT0PPS	—	—	—	INT0PPS					277
3ABFh	PPSLOCK	—	—	—	—	—	—	—	PPSLOCKED	283
3ABEh	—	Reserved, maintain as '0'								
3ABDh - 3A9Ah	—	Unimplemented								
3A99h	—	Reserved, maintain as '0'								

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
3: Unimplemented on PIC18(L)F26/27/45/46/47K42 devices.
4: Unimplemented in PIC18(L)F45/55K42.