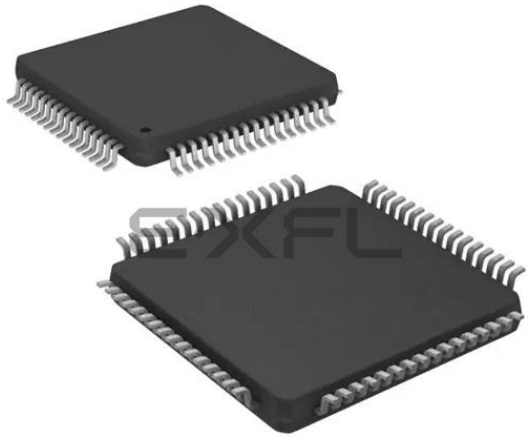


Welcome to E-XFL.COM

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	25MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	54
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f6722-e-pt

3.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

Certain exits from power-managed modes do not invoke the OST at all. There are two cases:

- PRI_IDLE mode, where the primary clock source is not stopped and
- the primary clock source is not any of the LP, XT, HS or HSPLL modes.

In these instances, the primary clock source either does not require an oscillator start-up delay since it is already running (PRI_IDLE), or normally does not require an oscillator start-up delay (RC, EC and INTIO Oscillator modes). However, a fixed delay of interval TCSD following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

TABLE 3-2: EXIT DELAY ON WAKE-UP BY RESET FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)

Clock Source before Wake-up	Clock Source after Wake-up	Exit Delay	Clock Ready Status Bit (OSCCON)
Primary Device Clock (PRI_IDLE mode)	LP, XT, HS	T_{CSD}(1)	OSTS
	HSPLL		
	EC, RC		
	INTOSC(2)		IOFS
T1OSC or INTRC	LP, XT, HS	T_{OST}(3)	OSTS
	HSPLL	$T_{OST} + t_{rc}$(3)	
	EC, RC	T_{CSD}(1)	
	INTOSC(2)	T_{IOBST}(4)	IOFS
INTOSC(2)	LP, XT, HS	T_{OST}(4)	OSTS
	HSPLL	$T_{OST} + t_{rc}$(3)	
	EC, RC	T_{CSD}(1)	
	INTOSC(2)	None	IOFS
None (Sleep mode)	LP, XT, HS	T_{OST}(3)	OSTS
	HSPLL	$T_{OST} + t_{rc}$(3)	
	EC, RC	T_{CSD}(1)	
	INTOSC(2)	T_{IOBST}(4)	IOFS

Note 1: TCSD (parameter 38, Table 28-12) is a required delay when waking from Sleep and all Idle modes and runs concurrently with any other required delays (see **Section 3.4 “Idle Modes”**).

2: Includes both the INTOSC 8 MHz source and postscaler derived frequencies. On Reset, INTOSC defaults to 1 MHz.

3: TOST is the Oscillator Start-up Timer (parameter 32, Table 28-12). t_{rc} is the PLL Lock-out Timer (parameter F12, Table 28-7); it is also designated as TPLL.

4: Execution continues during TIOBST (parameter 39, Table 28-12), the INTOSC stabilization period.

FIGURE 4-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD, VDD RISE < TPWRT)

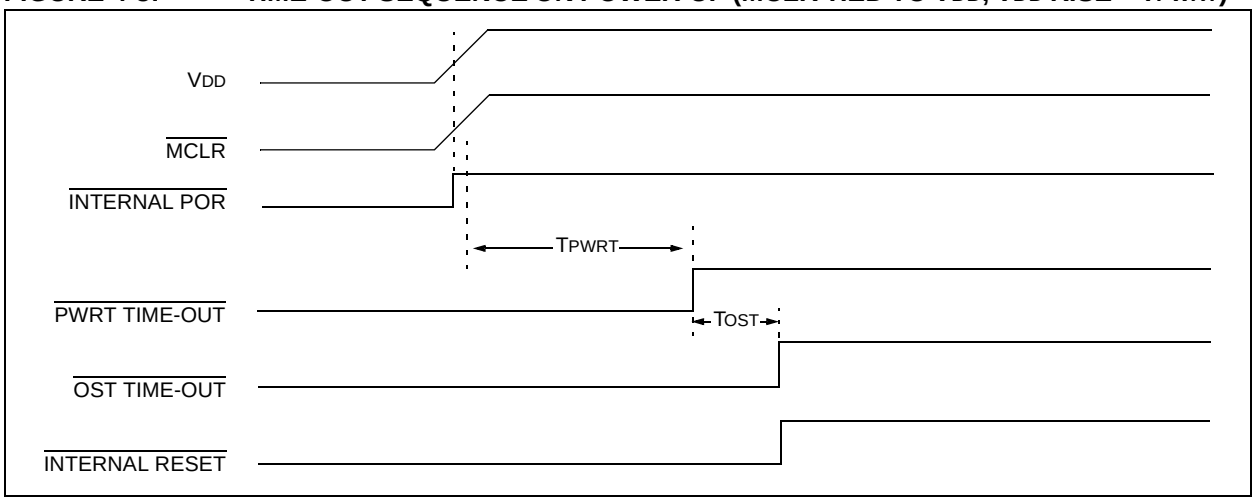


FIGURE 4-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

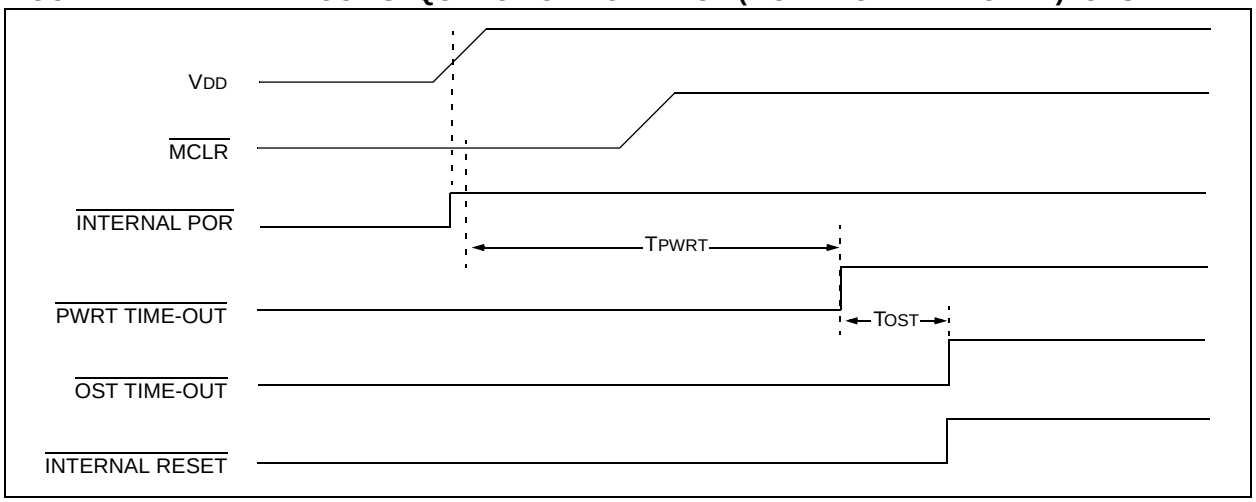


FIGURE 4-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

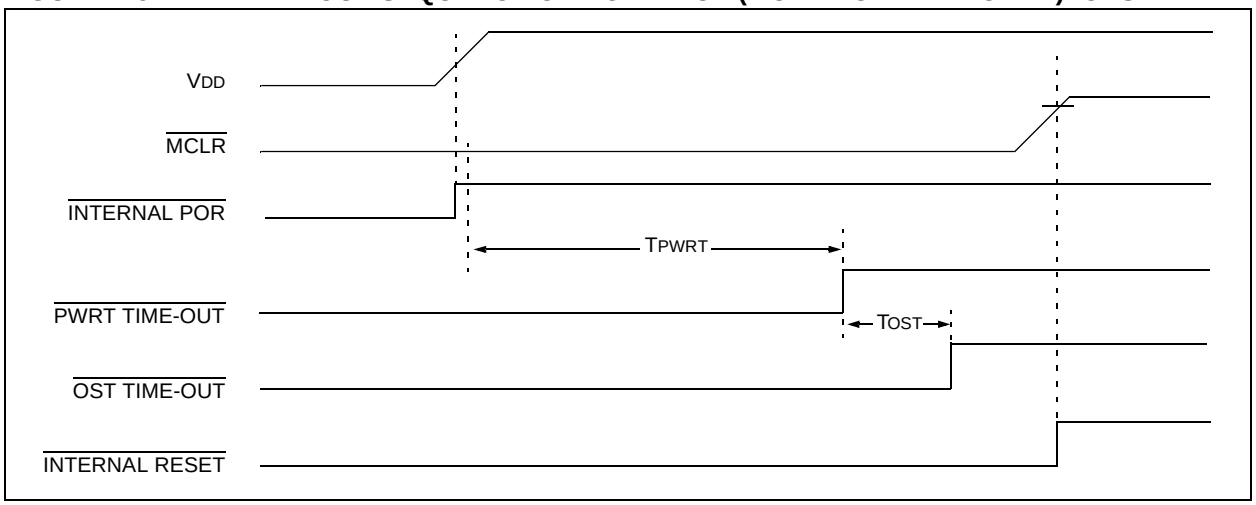


TABLE 4-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices				Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
ADRESH	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	6X27	6X22	8X27	8X22	--00 0000	--00 0000	--uu uuuu
ADCON1	6X27	6X22	8X27	8X22	--00 0000	--00 0000	--uu uuuu
ADCON2	6X27	6X22	8X27	8X22	0-00 0000	0-00 0000	u-uu uuuu
CCPR1H	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
CCPR2H	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
CCPR3H	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR3L	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP3CON	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
ECCP1AS	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
CVRCON	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
CMCON	6X27	6X22	8X27	8X22	0000 0111	0000 0111	uuuu uuuu
TMR3H	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	6X27	6X22	8X27	8X22	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	6X27	6X22	8X27	8X22	0000 0000	uuuu uuuu	uuuu uuuu
PSPCON	6X27	6X22	8X27	8X22	0000 ----	0000 ----	uuuu ----
SPBRG1	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
RCREG1	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
TXREG1	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
TXSTA1	6X27	6X22	8X27	8X22	0000 0010	0000 0010	uuuu uuuu
RCSTA1	6X27	6X22	8X27	8X22	0000 000x	0000 000x	uuuu uuuu
EEADRH	6X27	6X22	8X27	8X22	---- --00	---- --00	---- --uu
EEADR	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
EEDATA	6X27	6X22	8X27	8X22	0000 0000	0000 0000	uuuu uuuu
EECON2	6X27	6X22	8X27	8X22	0000 0000	0000 0000	0000 0000
EECON1	6X27	6X22	8X27	8X22	xx-0 x000	uu-0 u000	uu-u uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition.
Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 4-3 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

5.1.2 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 5.1.5.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

5.1.3 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, STKPTR. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the top-of-stack Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a POP from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

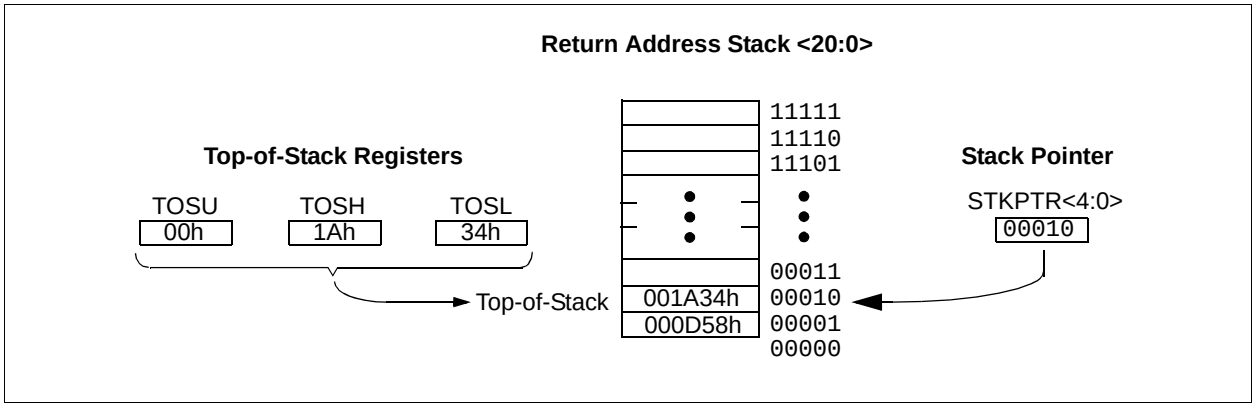
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

5.1.3.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 5-3). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the global interrupt enable bits while accessing the stack to prevent inadvertent stack corruption.

FIGURE 5-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



5.1.5.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to or from program memory one byte at a time.

Table read and table write operations are discussed further in **Section 6.1 "Table Reads and Table Writes"**.

5.2 PIC18 Instruction Cycle

5.2.1 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 5-4.

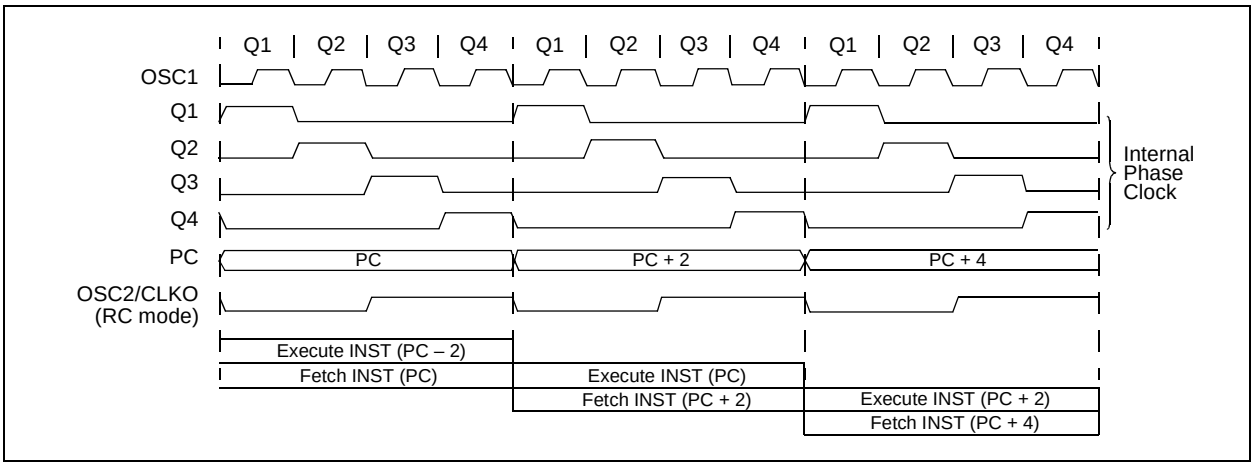
5.2.2 INSTRUCTION FLOW/PIPELINING

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 5-3).

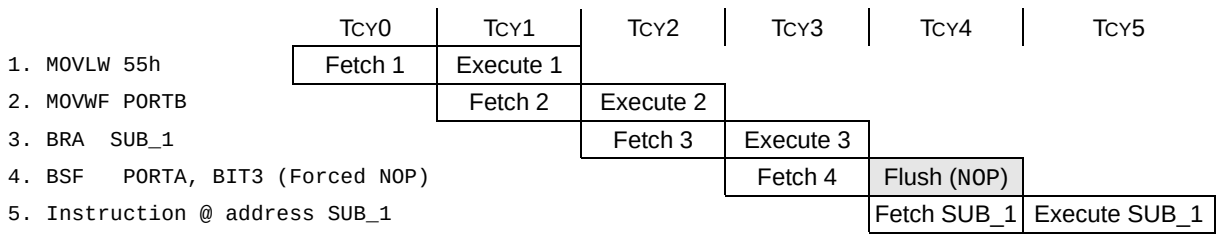
A fetch cycle begins with the program counter incrementing in Q1.

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 5-4: CLOCK/INSTRUCTION CYCLE



EXAMPLE 5-3: INSTRUCTION PIPELINE FLOW



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

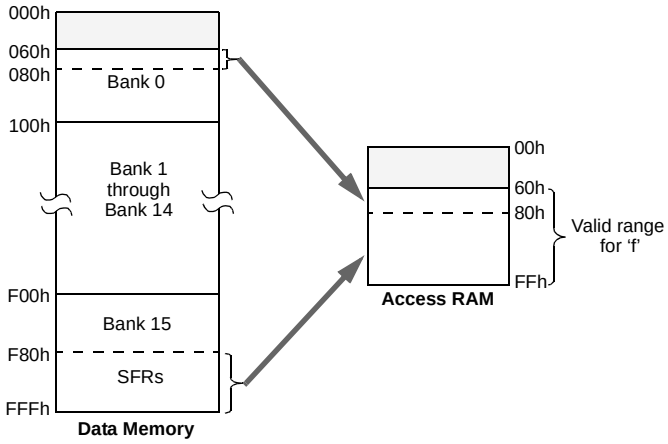
FIGURE 5-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

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

When 'a' = 0 and f ≥ 60h:

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations 060h to 07Fh (Bank 0) and F80h to FFFh (Bank 15) of data memory.

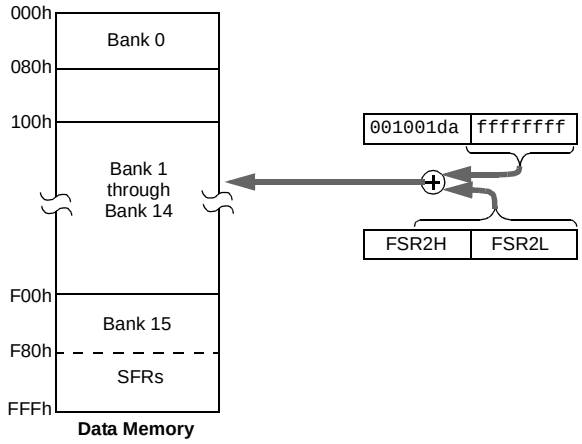
Locations below 60h are not available in this addressing mode.



When 'a' = 0 and f ≤ 5Fh:

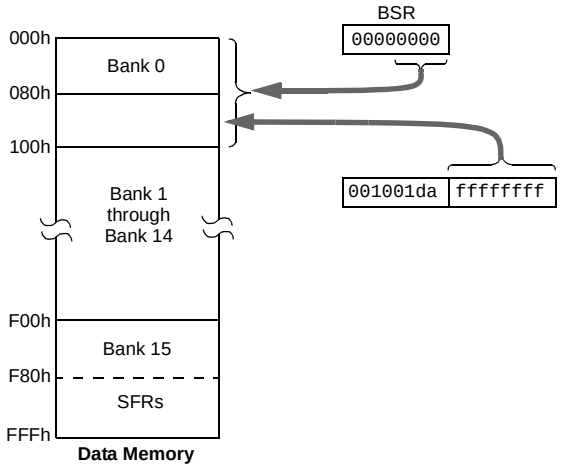
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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



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

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



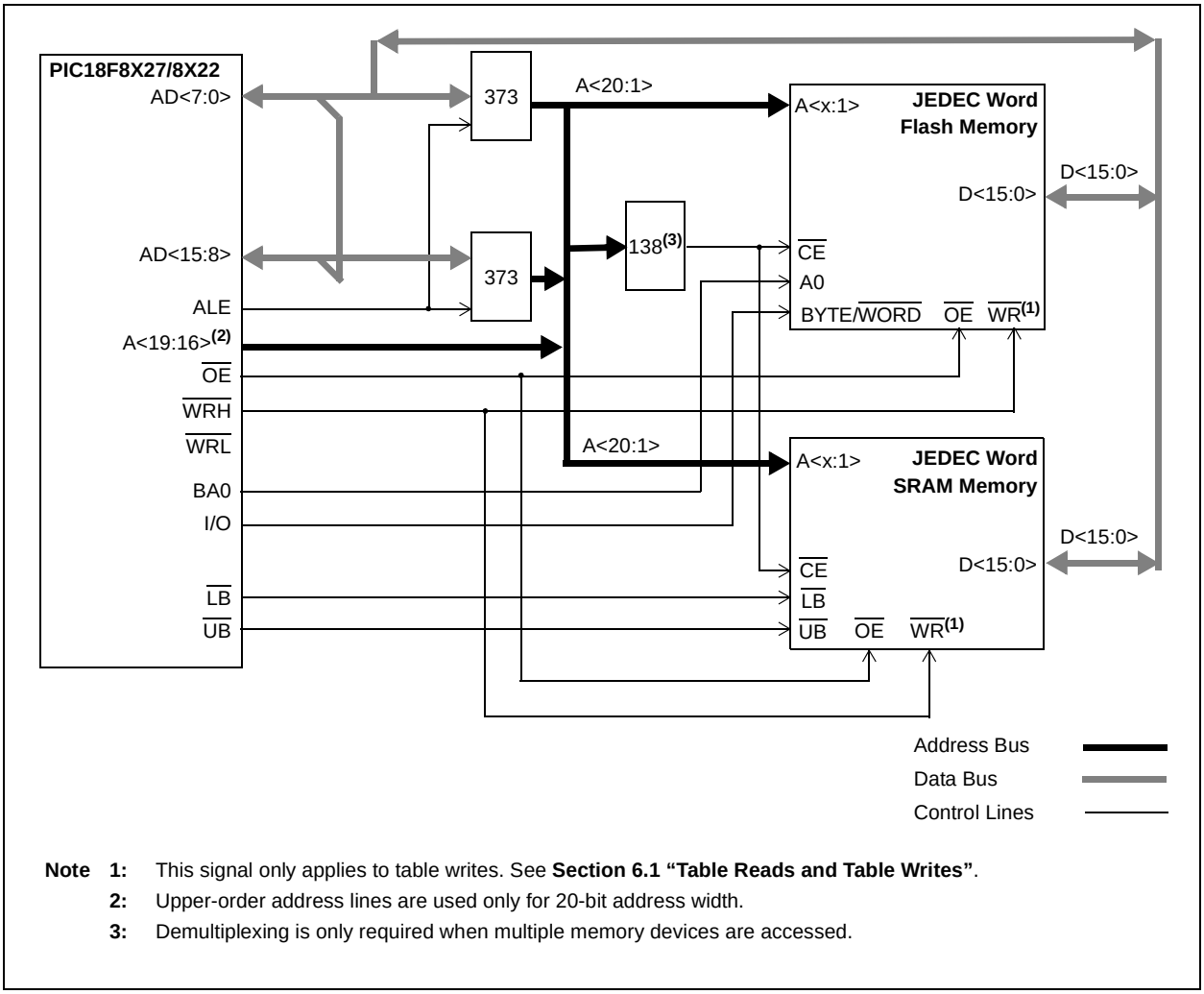
7.5.3 16-BIT BYTE SELECT MODE

Figure 7-3 shows an example of 16-bit Byte Select mode. This mode allows table write operations to word-wide external memories with byte selection capability. This generally includes both word-wide Flash and SRAM devices.

During a TBLWT cycle, the TABLAT data is presented on the upper and lower byte of the AD<15:0> bus. The $\overline{WRH}$ signal is strobed for each write cycle; the $\overline{WRL}$ pin is not used. The BA0 or $\overline{UB}$/$\overline{LB}$ signals are used to select the byte to be written, based on the Least Significant bit of the TBLPTR register.

Flash and SRAM devices use different control signal combinations to implement Byte Select mode. JEDEC standard Flash memories require that a controller I/O port pin be connected to the memory's BYTE/$\overline{WORD}$ pin to provide the select signal. They also use the BA0 signal from the controller as a byte address. JEDEC standard static RAM memories, on the other hand, use the $\overline{UB}$ or $\overline{LB}$ signals to select the byte.

FIGURE 7-3: 16-BIT BYTE SELECT MODE EXAMPLE



Note 1: This signal only applies to table writes. See **Section 6.1 "Table Reads and Table Writes"**.

2: Upper-order address lines are used only for 20-bit address width.

3: Demultiplexing is only required when multiple memory devices are accessed.

10.0 INTERRUPTS

The PIC18F8722 family of devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. High-priority interrupt events will interrupt any low-priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

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

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

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

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 0008h or 0018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit, which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit, which enables/disables all interrupt sources. All interrupts branch to address 0008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High-priority interrupt sources can interrupt a low-priority interrupt. Low-priority interrupts are not processed while high-priority interrupts are in progress.

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

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

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

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

TABLE 17-2: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	57
RCON	IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	56
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	60
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	60
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	60
PIR2	OSCFIF	CMIF	—	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF	60
PIE2	OSCFIE	CMIE	—	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE	60
IPR2	OSCFIP	CMIP	—	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP	60
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CCP5IF	CCP4IF	CCP3IF	60
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CCP5IE	CCP4IE	CCP3IE	60
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CCP5IP	CCP4IP	CCP3IP	60
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	60
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	60
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0	60
TRISG	—	—	—	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	60
TRISH(1)	TRISH7	TRISH6	TRISH5	TRISH4	TRISH3	TRISH2	TRISH1	TRISH0	60
TMR1L	Timer1 Register Low Byte								58
TMR1H	Timer1 Register High Byte								58
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	58
TMR3H	Timer3 Register High Byte								59
TMR3L	Timer3 Register Low Byte								59
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	59
CCPR1L	Enhanced Capture/Compare/PWM Register 1 Low Byte								59
CCPR1H	Enhanced Capture/Compare/PWM Register 1 High Byte								59
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	59
CCPR2L	Enhanced Capture/Compare/PWM Register 2 Low Byte								59
CCPR2H	Enhanced Capture/Compare/PWM Register 2 High Byte								59
CCP2CON	P2M1	P2M0	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	59
CCP3CON	P3M1	P3M0	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	59
CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0	61
CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0	61

Legend: — = unimplemented, read as '0'. Shaded cells are not used by Capture/Compare, Timer1 or Timer3.

Note 1: Implemented on 80-pin devices only.

REGISTER 19-5: SSPxCON2: MSSPx CONTROL REGISTER 2 (I^2C™ MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT(1)	ACKEN(2)	RCEN(2)	PEN(2)	RSEN(2)	SEN(2)
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 **GCEN:** General Call Enable bit (Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPxSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)(1)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)(2)
1 = Initiate Acknowledge sequence on SDAx and SCLx pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master mode only)(2)
1 = Enables Receive mode for I^2C
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)(2)
1 = Initiate Stop condition on SDAx and SCLx pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit (Master mode only)(2)
1 = Initiate Repeated Start condition on SDAx and SCLx pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enable/Stretch Enable bit(2)
In Master mode:
1 = Initiate Start condition on SDAx and SCLx pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is disabled

Note 1: Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
2: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I^2C™ module is active, these bits may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

21.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) converter module has 12 inputs for the 64-pin devices and 16 for the 80-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 21-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 21-2, configures the functions of the port pins. The ADCON2 register, shown in Register 21-3, configures the A/D clock source, programmed acquisition time and justification.

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3(1)	CHS2(1)	CHS1(1)	CHS0(1)	GO/$\overline{\text{DONE}}$	ADON
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 **Unimplemented:** Read as ‘0’

bit 5-2 **CHS<3:0>** Analog Channel Select bits(1)
0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)
0110 = Channel 6 (AN6)
0111 = Channel 7 (AN7)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)(1)
1101 = Channel 13 (AN13)(1)
1110 = Channel 14 (AN14)(1)
1111 = Channel 15 (AN15)(1)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit
1 = A/D converter module is enabled
0 = A/D converter module is disabled

Note 1: These channels are not implemented on 64-pin devices.

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
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 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as ‘0’

bit 5-3 **ACQT<2:0>:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

21.8 Use of the ECCP2 Trigger

An A/D conversion can be started by the Special Event Trigger of the ECCP2 module. This requires that the CCP2M<3:0> bits (CCP2CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ bit will be set, starting the A/D acquisition and conversion and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH:ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition period is either timed by the user, or an appropriate T_{ACQ} time selected before the Special Event Trigger sets the GO/$\overline{\text{DONE}}$ bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the Special Event Trigger will be ignored by the A/D module but will still reset the Timer1 (or Timer3) counter.

TABLE 21-2: REGISTERS ASSOCIATED WITH A/D OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	57
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	60
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	60
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	60
PIR2	OSCFIF	CMIF	—	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF	60
PIE2	OSCFIE	CMIE	—	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE	60
IPR2	OSCFIP	CMIP	—	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP	60
ADRESH	A/D Result Register High Byte								59
ADRESL	A/D Result Register Low Byte								59
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	59
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	59
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	59
TRISA	TRISA7(1)	TRISA6(1)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	60
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	60
TRISH(2)	TRISH7	TRISH6	TRISH5	TRISH4	TRISH3	TRISH2	TRISH1	TRISH0	60

Legend: — = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

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'.

2: These registers are not implemented on 64-pin devices.

22.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 22-4. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 22-4: COMPARATOR ANALOG INPUT MODEL

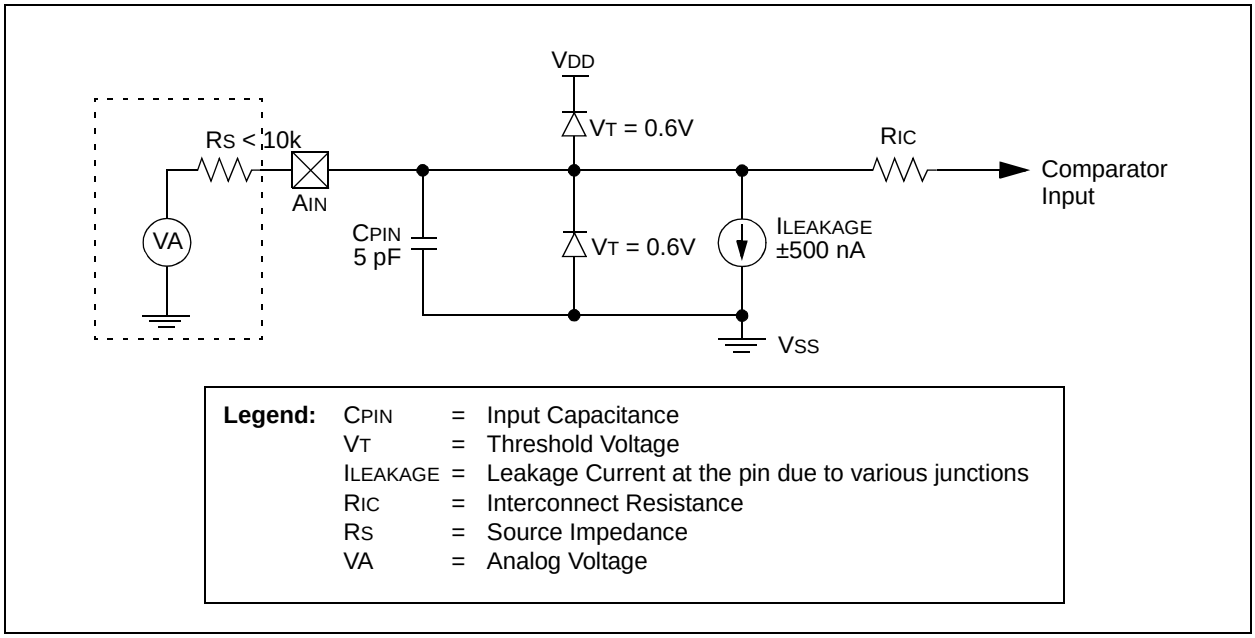


TABLE 22-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	59
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	59
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	60
PIR2	OSCFIF	CMIF	—	EEIF	BCL1IF	HLVDIF	TMR3IF	CCP2IF	60
PIE2	OSCFIE	CMIE	—	EEIE	BCL1IE	HLVDIE	TMR3IE	CCP2IE	60
IPR2	OSCFIP	CMIP	—	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP	60
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	60

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

NOTES:

The module is enabled by setting the HLVDEN bit. Each time that the HLVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit is a read-only bit and is used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

The VDIRMAG bit determines the overall operation of the module. When VDIRMAG is cleared, the module monitors for drops in VDD below a predetermined set point. When the bit is set, the module monitors for rises in VDD above the set point.

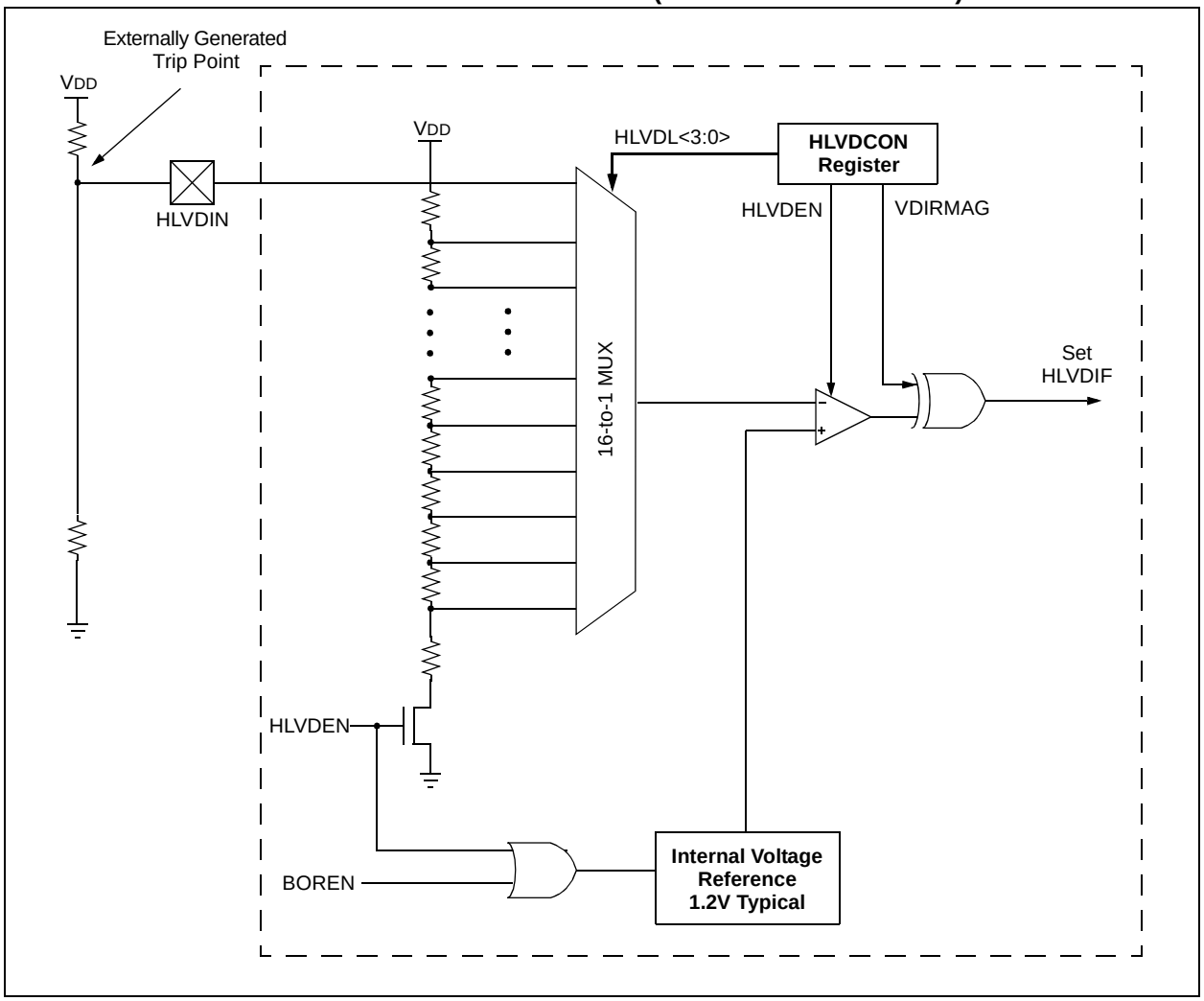
24.1 Operation

When the HLVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any one of 16 values. The trip point is selected by programming the HLVDL<3:0> bits (HLVDCON<3:0>).

The HLVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits HLVDL<3:0> are set to ‘1111’. In this state, the comparator input is multiplexed from the external input pin, HLVDIN. This gives users flexibility because it allows them to configure the High/Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

FIGURE 24-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)



BNOV **Branch if Not Overflow**

Syntax: BNOV n

Operands: -128 ≤ n ≤ 127

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

Status Affected: None

Encoding:

1110	0101	nnnn	nnnn

Description: If the Overflow bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNOV Jump`

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

BNZ **Branch if Not Zero**

Syntax: BNZ n

Operands: -128 ≤ n ≤ 127

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

Status Affected: None

Encoding:

1110	0001	nnnn	nnnn

Description: If the Zero bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNZ Jump`

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

CPFSGT **Compare f with W, Skip if f > W**

Syntax: CPFSGT f {,a}

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

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	010a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.

If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

CPFSLT **Compare f with W, Skip if f < W**

Syntax: CPFSLT f {,a}

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

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	000a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

26.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, the PIC18F8722 family of 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 for many of the standard PIC18 instructions.

The additional features of the extended instruction set are enabled 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 26-3. Detailed descriptions are provided in **Section 26.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 26-1 (page 322) 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.

26.2.1 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. The 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 26.2.3.1 "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 26-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

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

28.2 DC Characteristics: Power-Down and Supply Current PIC18F6X27/6X22/8X27/8X22 (Industrial, Extended) PIC18LF6X27/6X22/8X27/8X22 (Industrial) (Continued)

PIC18LF6X27/6X22/8X27/8X22 (Industrial)		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F6X27/6X22/8X27/8X22 (Industrial, Extended)		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD)(2)						
	PIC18LF6X27/6X22/8X27/8X22	300	350	µA	-40°C	VDD = 2.0V	FOSC = 1 MHz (**PRI_RUN** mode, EC oscillator)
		310	350	µA	+25°C		
		300	350	µA	+85°C		
	PIC18LF6X27/6X22/8X27/8X22	660	800	µA	-40°C	VDD = 3.0V	
		580	700	µA	+25°C		
		550	670	µA	+85°C		
	All devices	1.2	1.75	mA	-40°C	VDD = 5.0V	
		1.1	1.4	mA	+25°C		
		1.0	1.3	mA	+85°C		
	Extended devices only	1.0	1.4	mA	+125°C		
	PIC18LF6X27/6X22/8X27/8X22	0.86	1.2	mA	-40°C	VDD = 2.0V	FOSC = 4 MHz (**PRI_RUN** mode, EC oscillator)
		0.88	1.2	mA	+25°C		
		0.88	1.2	mA	+85°C		
	PIC18LF6X27/6X22/8X27/8X22	1.6	1.9	mA	-40°C	VDD = 3.0V	
		1.6	1.8	mA	+25°C		
		1.6	1.8	mA	+85°C		
	All devices	3.2	3.6	mA	-40°C	VDD = 5.0V	
		3.1	3.5	mA	+25°C		
		3.0	3.5	mA	+85°C		
	Extended devices only	3.1	3.5	mA	+125°C		

Legend: Shading of rows is to assist in readability of the table.

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD OR VSS;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: When operation below -10°C is expected, use T1OSC High-Power mode, where LPT1OSC (CONFIG3H<2>) = 0. When operation will always be above -10°C, then the low-power Timer1 oscillator may be selected.

4: BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.