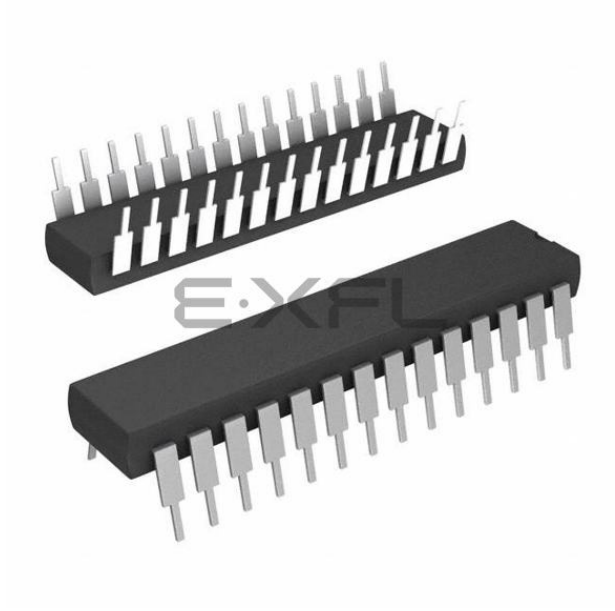


Welcome to E-XFL.COM

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	ECANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	24
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 8x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25k80-e-sp

TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices			Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
B4D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	-uuu uuuu
B4EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu
B4SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CANCON_RO6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
CANSTAT_RO6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
B3D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	-xxx xxxx	-uuu uuuu	-uuu uuuu
B3EIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3EIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3SIDL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx x-xx	uuuu u-uu	uuuu u-uu
B3SIDH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	0000 0000	0000 0000	uuuu uuuu
CANCON_RO7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	1000 0000	1000 0000	uuuu uuuu
B2D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	xxxx xxxx	uuuu uuuu	uuuu 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 5-3 for Reset value for specific conditions.

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 as '0'.

6.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy all of Bank 15 (F00h to FFFh) and the top part of Bank 14 (EF4h to EFFh).

A list of these registers is given in Table 6-1 and Table 6-2.

The SFRs can be classified into two sets: those associated with the “core” device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and Interrupt registers are described in their respective chapters, while the ALU’s STATUS register is described later in this section. Registers related to the operation of the peripheral features are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as ‘0’s.

TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F66K80 FAMILY

Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name	Addr.	Name
FFFh	TOSU	FDFh	INDF2(1)	FBFh	ECCP1AS	F9Fh	IPR1	F7Fh	EECON1	F5Fh	CM1CON(5)
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	ECCP1DEL	F9Eh	PIR1	F7Eh	EECON2	F5Eh	CM2CON(5)
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCPR1H	F9Dh	PIE1	F7Dh	SPBRGH1	F5Dh	ANCON0(5)
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR1L	F9Ch	PSTR1CON	F7Ch	SPBRGH2	F5Ch	ANCON1(5)
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCP1CON	F9Bh	OSCTUNE	F7Bh	SPBRG2	F5Bh	WPUB(5)
FFAh	PCLATH	FDAh	FSR2H	FBAh	TXSTA2	F9Ah	REFOCON	F7Ah	RCREG2	F5Ah	IOCB(5)
FF9h	PCL	FD9h	FSR2L	FB9h	BAUDCON2	F99h	CCPTMRS	F79h	TXREG2	F59h	PMD0(5)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	IPR4	F98h	TRISG(3)	F78h	IPR5	F58h	PMD1(5)
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PIR4	F97h	TRISF(3)	F77h	PIR5	F57h	PMD2(5)
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	PIE4	F96h	TRISE(4)	F76h	PIE5	F56h	PADCFG1(5)
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD(4)	F75h	EEADRH	F55h	CTMUCONH(5)
FF4h	PRODH	FD4h	—(2)	FB4h	CMSTAT	F94h	TRISC	F74h	EEADR	F54h	CTMUCONL(5)
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	EEDATA	F53h	CTMUICONH(5)
FF2h	INTCON	FD2h	OSCCON2	FB2h	TMR3L	F92h	TRISA	F72h	ECANCON	F52h	CCPR2H(5)
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	ODCON	F71h	COMSTAT	F51h	CCPR2L(5)
FF0h	INTCON3	FD0h	RCON	FB0h	T3GCON	F90h	SLRCON	F70h	CIOCON	F50h	CCP2CON(4,5)
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG1	F8Fh	LATG(3)	F6Fh	CANCON	F4Fh	CCPR3H(4,5)
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG1	F8Eh	LATF(3)	F6Eh	CANSTAT	F4Eh	CCPR3L(4,5)
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG1	F8Dh	LATE(4)	F6Dh	RXB0D7	F4Dh	CCP3CON(5)
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA1	F8Ch	LATD(4)	F6Ch	RXB0D6	F4Ch	CCPR4H(5)
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA1	F8Bh	LATC	F6Bh	RXB0D5	F4Bh	CCPR4L(5)
FEAh	FSR0H	FCAh	T2CON	FAAh	T1GCON	F8Ah	LATB	F6Ah	RXB0D4	F4Ah	CCP4CON(5)
FE9h	FSR0L	FC9h	SSPBUF	FA9h	PR4	F89h	LATA	F69h	RXB0D3	F49h	CCPR5H(5)
FE8h	WREG	FC8h	SSPADD	FA8h	HLVDCON	F88h	T4CON	F68h	RXB0D2	F48h	CCPR5L(5)
FE7h	INDF1(1)	FC8h	SSPMSK	FA7h	BAUDCON1	F87h	TMR4	F67h	RXB0D1	F47h	CCP5CON(5)
FE6h	POSTINC1(1)	FC7h	SSPSTAT	FA6h	RCSTA2	F86h	PORTG(3)	F66h	RXB0D0	F46h	PSPCON(4,5)
FE5h	POSTDEC1(1)	FC6h	SSPCON1	FA5h	IPR3	F85h	PORTF(3)	F65h	RXB0DLC	F45h	MDCON(3,5)
FE4h	PREINC1(1)	FC5h	SSPCON2	FA4h	PIR3	F84h	PORTE	F64h	RXB0EIDL	F44h	MDSRC(3,5)
FE3h	PLUSW1(1)	FC4h	ADRESH	FA3h	PIE3	F83h	PORTD(4)	F63h	RXB0EIDH	F43h	MDCARH(3,5)
FE2h	FSR1H	FC3h	ADRESL	FA2h	IPR2	F82h	PORTC	F62h	RXB0SIDL	F42h	MDCARL(3,5)
FE1h	FSR1L	FC2h	ADCON0	FA1h	PIR2	F81h	PORTB	F61h	RXB0SIDH	F41h	—(2)
FE0h	BSR	FC1h	ADCON1	FA0h	PIE2	F80h	PORTA	F60h	RXB0CON	F40h	—(2)
		FC0h	ADCON2								

Note 1: This is not a physical register.
2: Unimplemented registers are read as ‘0’.
3: This register is only available on devices with 64 pins.
4: This register is not available on devices with 28 pins.
5: Addresses, E41h through F5Fh, are also used by the SFRs, but are not part of the Access RAM. To access these registers, users must always load the proper BSR value.

6.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action on its stored value.

These operands are:

- POSTDEC – Accesses the FSR value, then automatically decrements it by ‘1’ afterwards
- POSTINC – Accesses the FSR value, then automatically increments it by ‘1’ afterwards
- PREINC – Increments the FSR value by ‘1’, then uses it in the operation
- PLUSW – Adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the new value in the operation

In this context, accessing an INDF register uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value, offset by the value in the W register, with neither value actually changed in the operation. Accessing the other virtual registers changes the value of the FSR registers.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair. Rollovers of the FSRnL register, from FFh to 00h, carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (for example, Z, N and OV bits).

The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

6.4.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations.

As a specific case, assume that the FSR0H:FSR0L registers contain FE7h, the address of INDF1. Attempts to read the value of the INDF1, using INDF0 as an operand, will return 00h. Attempts to write to INDF1, using INDF0 as the operand, will result in a NOP.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair, but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, however, particularly if their code uses Indirect Addressing.

Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution, so that they do not inadvertently change settings that might affect the operation of the device.

6.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds five additional two-word commands to the existing PIC18 instruction set: ADDFSR, CALLW, MOVSF, MOVSS and SUBFSR. These instructions are executed as described in **Section 6.2.4 “Two-Word Instructions”**.

10.0 INTERRUPTS

Members of the PIC18F66K80 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.

The registers for controlling interrupt operation are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3, PIR4 and PIR5
- PIE1, PIE2, PIE3, PIE4 and PIE5
- IPR1, IPR2, IPR3, IPR4 and IPR5

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** – Indicating that an interrupt event occurred
- **Enable bit** – Enabling program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** – Specifying 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 that 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 that enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit that 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 (ISR), 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) that 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.

REGISTER 10-7: PIR4: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 4

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
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 **TMR4IF:** TMR4 Overflow Interrupt Flag bit
1 = TMR4 register overflowed (must be cleared in software)
0 = TMR4 register did not overflow

bit 6 **EEIF:** Data EEDATA/Flash Write Operation Interrupt Flag bit
1 = The write operation is complete (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 5 **CMP2IF:** CMP2 Interrupt Flag bit
1 = CMP2 interrupt occurred (must be cleared in software)
0 = CMP2 interrupt did not occur

bit 4 **CMP1IF:** CMP1 Interrupt Flag bit
1 = CMP1 interrupt occurred (must be cleared in software)
0 = CMP1 interrupt did not occur

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

bit 2 **CCP5IF:** CCP5 Interrupt Flag bit
Capture Mode
1 = A TMR register capture occurred (bit must be cleared in software)
0 = No TMR register capture occurred
Compare Mode
1 = A TMR register compare match occurred (must be cleared in software)
0 = No TMR register compare match occurred
PWM Mode
Not used in PWM mode.

bit 1 **CCP4IF:** CCP4 Interrupt Flag bit
Capture Mode
1 = A TMR register capture occurred (bit must be cleared in software)
0 = No TMR register capture occurred
Compare Mode
1 = A TMR register compare match occurred (must be cleared in software)
0 = No TMR register compare match occurred
PWM Mode
Not used in PWM mode.

bit 0 **CCP3IF:** CCP3 Interrupt Flag bit
Capture Mode
1 = A TMR register capture occurred (bit must be cleared in software)
0 = No TMR register capture occurred
Compare Mode
1 = A TMR register compare match occurred (must be cleared in software)
0 = No TMR register compare match occurred
PWM Mode
Not used in PWM mode.

11.2 PORTA, TRISA and LATA Registers

PORTA is a seven-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISA and LATA.

RA5 and RA<3:0> are multiplexed with analog inputs for the A/D Converter.

The operation of the analog inputs as A/D Converter inputs is selected by clearing or setting the ANSELx control bits in the ANCON1 register. The corresponding TRISA bits control the direction of these pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

Note:	RA5 and RA<3:0> are configured as analog inputs on any Reset and are read as '0'.

OSC2/CLKO/RA6 and OSC1/CLKI/RA7 normally serve as the external circuit connections for the external (primary) oscillator circuit (HS Oscillator modes) or the external clock input and output (EC Oscillator modes). In these cases, RA6 and RA7 are not available as digital I/O and their corresponding TRIS and LAT bits are read as '0'. When the device is configured to use HF-INTOSC, MF-INTOSC or LF-INTOSC as the default oscillator mode, RA6 and RA7 are automatically configured as digital I/O; the oscillator and clock in/clock out functions are disabled.

RA5 has additional functionality for Timer1 and Timer3. It can be configured as the Timer1 clock input or the Timer3 external clock gate input.

EXAMPLE 11-1: INITIALIZING PORTA

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output latches
CLRF    LATA    ; Alternate method to
                ; clear output data latches
MOVLW   00h     ; Configure A/D
MOVWF   ANCON1  ; for digital inputs
MOVLW   0BFh    ; Value used to initialize
                ; data direction
MOVWF   TRISA   ; Set RA<7, 5:0> as inputs,
                ; RA<6> as output
```

TABLE 19-5: REGISTERS ASSOCIATED WITH PWM AND TIMERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
RCON	IPEN	SBOREN	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
PIR4	TMR4IF	EEIF	CMP2IF	CMP1IF	—	CCP5IF	CCP4IF	CCP3IF
PIE4	TMR4IE	EEIE	CMP2IE	CMP1IE	—	CCP5IE	CCP4IE	CCP3IE
IPR4	TMR4IP	EEIP	CMP2IP	CMP1IP	—	CCP5IP	CCP4IP	CCP3IP
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
TMR2	Timer2 Register							
TMR4	Timer4 Register							
PR2	Timer2 Period Register							
PR4	Timer4 Period Register							
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
CCPR2L	Capture/Compare/PWM Register 2 Low Byte							
CCPR2H	Capture/Compare/PWM Register 2 High Byte							
CCPR3L	Capture/Compare/PWM Register 3 Low Byte							
CCPR3H	Capture/Compare/PWM Register 3 High Byte							
CCPR4L	Capture/Compare/PWM Register 4 Low Byte							
CCPR4H	Capture/Compare/PWM Register 4 High Byte							
CCPR5L	Capture/Compare/PWM Register 5 Low Byte							
CCPR5H	Capture/Compare/PWM Register 5 High Byte							
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0
CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0
CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0
CCPTMRS	—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD

Legend: — = unimplemented, read as '0'. Shaded cells are not used by PWM or Timer2/4.

REGISTER 20-2: CCPTMRS: CCP TIMER SELECT REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	C5TSEL	C4TSEL	C3TSEL	C2TSEL	C1TSEL
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 7-5 **Unimplemented:** Read as '0'

bit 4 **C5TSEL:** CCP5 Timer Selection bit
0 = CCP5 is based off of TMR1/TMR2
1 = CCP5 is based off of TMR3/TMR4

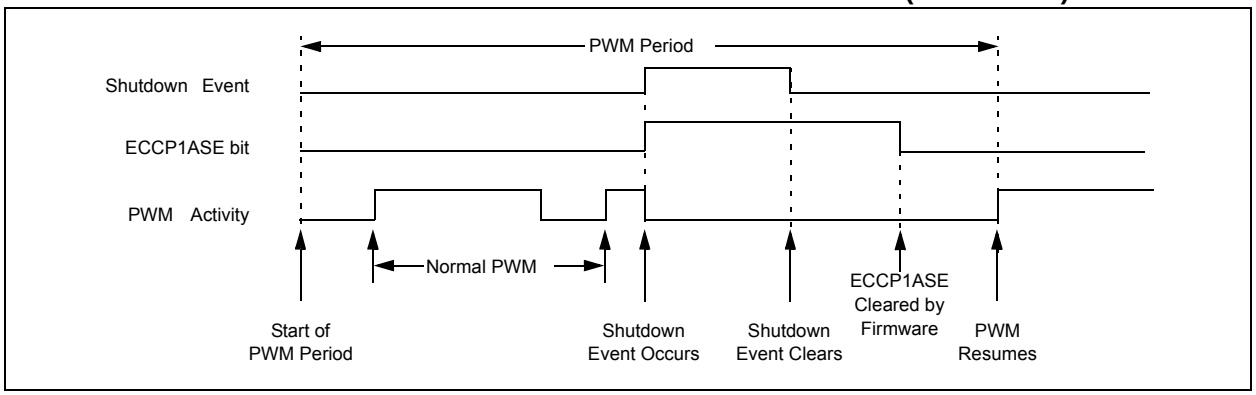
bit 3 **C4TSEL:** CCP4 Timer Selection bit
0 = CCP4 is based off of TMR1/TMR2
1 = CCP4 is based off of TMR3/TMR4

bit 2 **C3TSEL:** CCP3 Timer Selection bit
0 = CCP3 is based off of TMR1/TMR2
1 = CCP3 is based off of TMR3/TMR4

bit 1 **C2TSEL:** CCP2 Timer Selection bit
0 = CCP2 is based off of TMR1/TMR2
1 = CCP2 is based off of TMR3/TMR4

bit 0 **C1TSEL:** CCP1 Timer Selection bit
0 = ECCP1 is based off of TMR1/TMR2
1 = ECCP1 is based off of TMR3/TMR4

FIGURE 20-12: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (P1RSEN = 0)



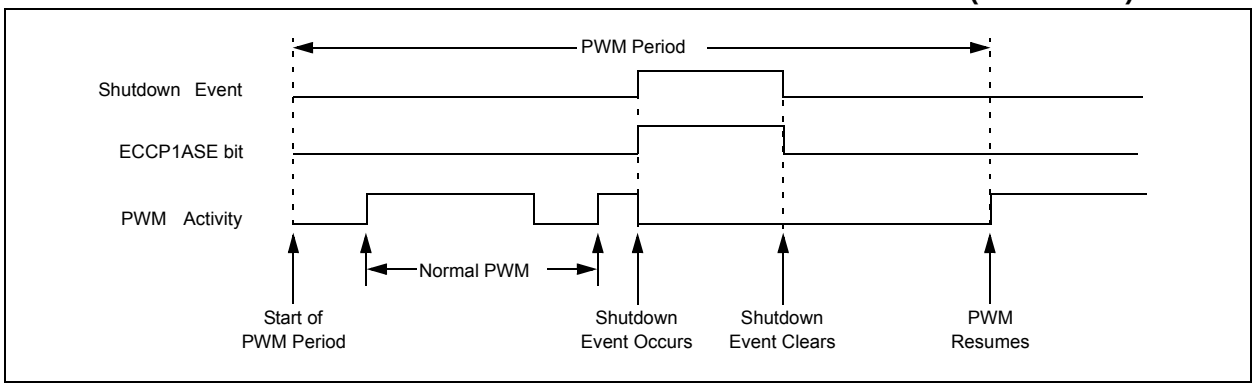
20.4.5 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the P1RSEN bit (ECCP1DEL<7>).

If auto-restart is enabled, the ECCP1ASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the ECCP1ASE bit will be cleared via hardware and normal operation will resume.

The module will wait until the next PWM period begins, however, before re-enabling the output pin. This behavior allows the auto-shutdown with auto-restart features to be used in applications based on current mode of PWM control.

FIGURE 20-13: PWM AUTO-SHUTDOWN WITH AUTO-RESTART ENABLED (P1RSEN = 1)



21.4 I²C Mode

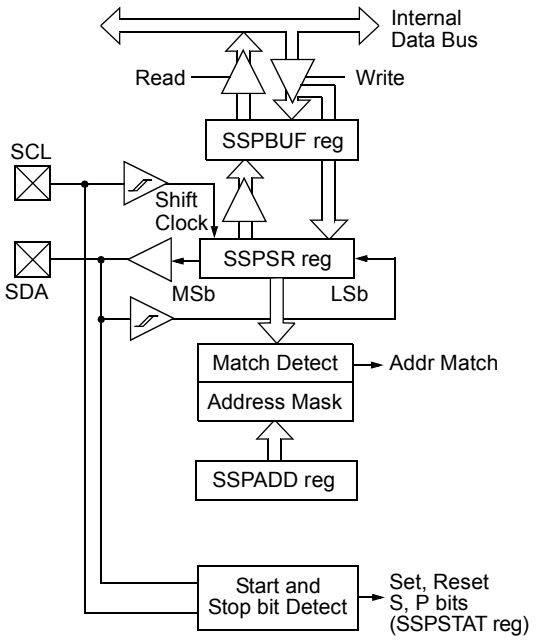
The MSSP module in I²C mode fully implements all master and slave functions (including general call support), and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial Clock (SCL) – RC3/REFO/SCL/SCK
- Serial Data (SDA) – RC4/SDA/SDI

The user must configure these pins as inputs by setting the associated TRIS bits.

FIGURE 21-7: MSSP BLOCK DIAGRAM (I²C™ MODE)



Note: Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

21.4.1 REGISTERS

The MSSP module has seven registers for I²C operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)
- I²C Slave Address Mask Register (SSPMSK)

SSPCON1, SSPCON2 and SSPSTAT are the control and status registers in I²C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

SSPADD contains the slave device address when the MSSP is configured in I²C Slave mode. When the MSSP is configured in Master mode, all eight bits of SSPADD act as the Baud Rate Generator reload value.

SSPMSK holds the slave address mask value when the module is configured for 7-Bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPADD; it is only accessible when the SSPM<3:0> bits are specifically set to permit access. Additional details are provided in **Section 21.4.3.4 “7-Bit Address Masking Mode”**.

In receive operations, SSPSR and SSPBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

21.4.7 BAUD RATE

In I^2C Master mode, the Baud Rate Generator (BRG) reload value is placed in the 8 bits of the SSPADD register (Figure 21-19). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In I^2C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by $\overline{ACK}$), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 21-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD. The SSPADD BRG value of 00h is not supported.

FIGURE 21-19: BAUD RATE GENERATOR BLOCK DIAGRAM

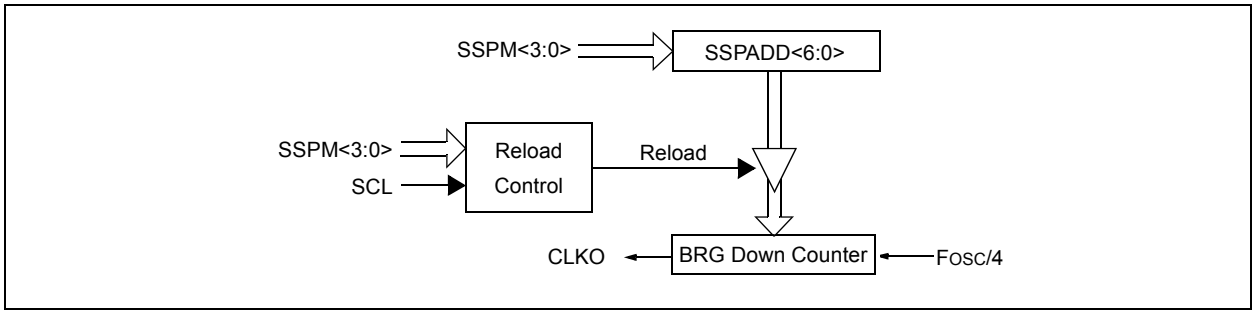


TABLE 21-3: I^2C™ CLOCK RATE w/BRG

FOSC	FCY	FCY * 2	BRG Value	FSCL (2 Rollovers of BRG)
40 MHz	10 MHz	20 MHz	18h	400 kHz(1)
40 MHz	10 MHz	20 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	20 MHz	63h	100 kHz
16 MHz	4 MHz	8 MHz	09h	400 kHz(1)
16 MHz	4 MHz	8 MHz	0Ch	308 kHz
16 MHz	4 MHz	8 MHz	27h	100 kHz
4 MHz	1 MHz	2 MHz	02h	333 kHz(1)
4 MHz	1 MHz	2 MHz	09h	100 kHz
16 MHz(2)	4 MHz	8 MHz	03h	1 MHz(1)

Note 1: The I^2C interface does not conform to the 400 kHz I^2C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

2: A minimum 16-MHz FOSC is required for 1 MHz I^2C.

22.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of two serial I/O modules. (Generically, the EUSART is also known as a Serial Communications Interface or SCI.)

The EUSART can be configured as a full-duplex, asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced USARTx modules implement additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN/J2602 bus) systems.

All members of the PIC18F66K80 family are equipped with two independent EUSART modules, referred to as EUSART1 and EUSART2. They can be configured in the following modes:

- Asynchronous (full duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit Break character transmission
- Synchronous – Master (half duplex) with selectable clock polarity
- Synchronous – Slave (half duplex) with selectable clock polarity

The pins of EUSART1 and EUSART2 are multiplexed with the functions with the following ports, depending on the device pin count. See Table 22-1.

TABLE 22-1: CONFIGURING EUSARTx PINS(1)

Pin Count	EUSART1		EUSART2	
	Port	Pins	Port	Pins
28-pin	PORTC	RC6/TX1/CK1 and RC7/RX1/DT1	PORTB	RB6/PGC/TX2/CK2/KBI2 and RB7/PGD/T3G/RX2/DT2/KBI3
40/44-pin	PORTC	RC6/TX1/CK1 and RC7/RX1/DT1	PORTD	RD6/TX2/CK2/P1C/PSP6 and RD7/RX2/DT2/P1D/PSP7
64-pin	PORTG	RG3/TX1/CK1 and RG0/RX1/DT1	PORTE	RE7/TX2/CK2 and RE6/RX2/DT2

Note 1: The EUSARTx control will automatically reconfigure the pin from input to output as needed.

In order to configure the pins as an EUSARTx:

- For EUSART1:
  - SPEN (RCSTA1<7>) must be set (= 1)
  - TRISx<x> must be set (= 1)
  - For Asynchronous and Synchronous Master modes, TRISx<x> must be cleared (= 0)
  - For Synchronous Slave mode, TRISx<x> must be set (= 1)
- For EUSART2:
  - SPEN (RCSTA2<7>) must be set (= 1)
  - TRISx<x> must be set (= 1)
  - For Asynchronous and Synchronous Master modes, TRISx<x> must be cleared (= 0)
  - For Synchronous Slave mode, TRISx<x> must be set (= 1)

22.5.2 EUSARTx SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep, or any Idle mode and bit, SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREGx register. If the RCxIE enable bit is set, the interrupt generated will wake the chip from the low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RCxIE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RCxIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCxIE, was set.
6. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREGx register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits (INTCON<7:6>) are set.

TABLE 22-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	TMR1GIF	TMR2IF	TMR1IF
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	TMR1GIE	TMR2IE	TMR1IE
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	TMR1GIP	TMR2IP	TMR1IP
PIR3	—	—	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	—
PIE3	—	—	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	—
IPR3	—	—	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	—
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG1	EUSART1 Receive Register							
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte							
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte							
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREG2	EUSART2 Receive Register							
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGH2	EUSART2 Baud Rate Generator Register High Byte							
SPBRG2	EUSART2 Baud Rate Generator Register Low Byte							
PMD0	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	UART2MD	UART1MD	SSPMD
ODCON	SSPOD	CCP5OD	CCP4OD	CCP3OD	CCP2OD	CCP1OD	U2OD	U1OD

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

Table 27-2 shows the relation between the clock generated by the PLL and the frequency error from jitter (measured jitter-induced error of 2%, Gaussian distribution, within 3 standard deviations), as a percentage of the nominal clock frequency.

This is clearly smaller than the expected drift of a crystal oscillator, typically specified at 100 ppm or 0.01%. If we add jitter to oscillator drift, we have a total frequency drift of 0.0132%. The total oscillator frequency errors for common clock frequencies and bit rates, including both drift and jitter, are shown in Table 27-3.

TABLE 27-2: FREQUENCY ERROR FROM JITTER AT VARIOUS PLL GENERATED CLOCK SPEEDS

PLL Output	P_{jitter}	T_{jitter}	Frequency Error at Various Nominal Bit Times (Bit Rates)			
			8 μs (125 Kb/s)	4 μs (250 Kb/s)	2 μs (500 Kb/s)	1 μs (1 Mb/s)
40 MHz	0.5 ns	1 ns	0.00125%	0.00250%	0.005%	0.01%
24 MHz	0.83 ns	1.67 ns	0.00209%	0.00418%	0.008%	0.017%
16 MHz	1.25 ns	2.5 ns	0.00313%	0.00625%	0.013%	0.025%

TABLE 27-3: TOTAL FREQUENCY ERROR AT VARIOUS PLL GENERATED CLOCK SPEEDS (100 PPM OSCILLATOR DRIFT, INCLUDING ERROR FROM JITTER)

Nominal PLL Output	Frequency Error at Various Nominal Bit Times (Bit Rates)			
	8 μs (125 Kb/s)	4 μs (250 Kb/s)	2 μs (500 Kb/s)	1 μs (1 Mb/s)
40 MHz	0.01125%	0.01250%	0.015%	0.02%
24 MHz	0.01209%	0.01418%	0.018%	0.027%
16 MHz	0.01313%	0.01625%	0.023%	0.035%

MOVLW **Move Literal to W**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk

Description: The eight-bit literal ‘k’ is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example: MOVLW 5Ah

After Instruction
W = 5Ah

MOVWF **Move W to f**

Syntax: MOVWF f {,a}

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

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank.

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write register ‘f’

Example: MOVWF REG, 0

Before Instruction
W = 4Fh
REG = FFh
After Instruction
W = 4Fh
REG = 4Fh

RLNCF **Rotate Left f (No Carry)**

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

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

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

Status Affected: N, Z

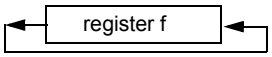
Encoding:

0100	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. 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 $f \le 95$ (5Fh). See **Section 29.2.3 "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:
```
RLNCF   REG, 1, 0
```
Before Instruction
```
REG  =  1010 1011
```
After Instruction
```
REG  =  0101 0111
```

RRCF **Rotate Right f through Carry**

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

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

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

Status Affected: C, N, Z

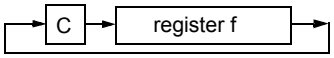
Encoding:

0011	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed 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 $f \le 95$ (5Fh). See **Section 29.2.3 "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:
```
RRCF    REG, 0, 0
```
Before Instruction
```
REG  =  1110 0110
C    =  0
```
After Instruction
```
REG  =  1110 0110
W    =  0111 0011
C    =  0
```

RRNCF **Rotate Right f (No Carry)**

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

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

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

Status Affected: N, Z

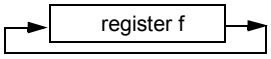
Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value.

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



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:
```
RRNCF   REG, 1, 0
Before Instruction
    REG     =   1101 0111
After Instruction
    REG     =   1110 1011
```

Example 2:
```
RRNCF   REG, 0, 0
Before Instruction
    W       =   ?
    REG     =   1101 0111
After Instruction
    W       =   1110 1011
    REG     =   1101 0111
```

SETF **Set f**

Syntax: SETF f {,a}

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

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh.

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

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example:
```
SETF    REG,1
Before Instruction
    REG     =   5Ah
After Instruction
    REG     =   FFh
```

31.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on $\overline{MCLR}$ with respect to V_{SS} ..... -0.3V to 9.0V
Voltage on any digital only I/O pin with respect to V_{SS} (except V_{DD}) ..... -0.3V to 7.5V
Voltage on any combined digital and analog pin with respect to V_{SS} (except V_{DD} and $\overline{MCLR}$) ..... -0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS} (PIC18F66K80) ..... -0.3V to 7.5V
Voltage on V_{DD} with respect to V_{SS} (PIC18LF66K80) ..... -0.3V to 3.66V
Total power dissipation **(Note 1)** 1W
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ±20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ±20 mA
Maximum output current sunk by PORTA<7:6> and any PORTB and PORTC I/O pins ..... 25 mA
Maximum output current sunk by any PORTD and PORTE I/O pins ..... 8 mA
Maximum output current sunk by PORTA<5:0> and any PORTF and PORTG I/O pins ..... 2 mA
Maximum output current sourced by PORTA<7:6> and any PORTB and PORTC I/O pins ..... 25 mA
Maximum output current sourced by any PORTD, PORTE and PORTJ I/O pins ..... 8 mA
Maximum output current sourced by PORTA<5:0> and any PORTF, PORTG and PORTH I/O pins ..... 2 mA
Maximum current sunk by all ports combined ..... 200 mA

Note 1: Power dissipation is calculated as follows:
$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

† **NOTICE:** Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 31-8: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

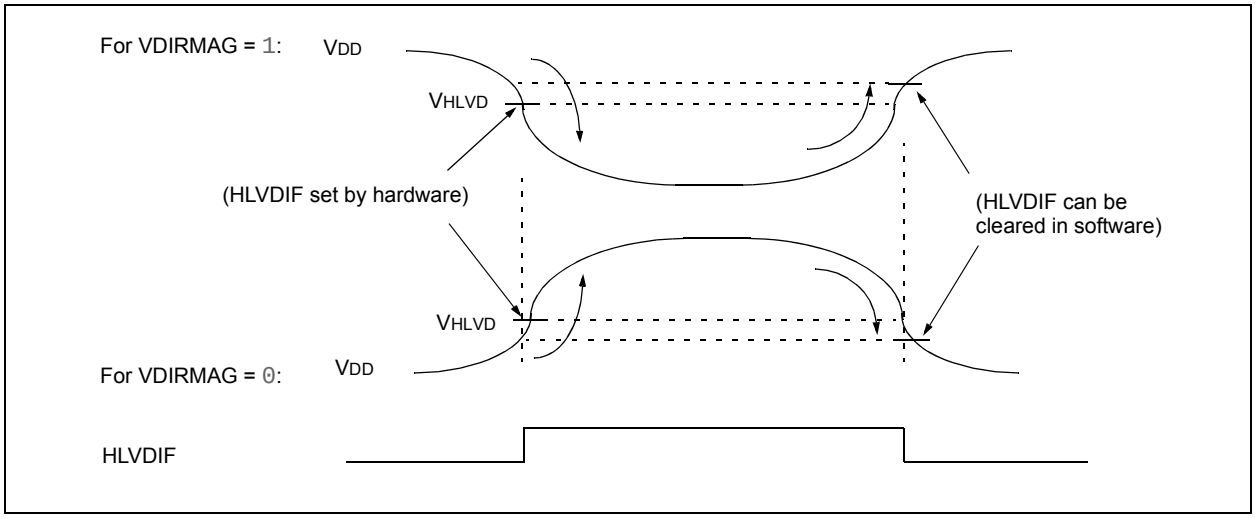


TABLE 31-12: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

Standard Operating Conditions (unless otherwise stated)
Operating temperature -40°C ≤ TA ≤ +85°C for industrial
-40°C ≤ TA ≤ +125°C for extended

Param No.	Sym	Characteristic		Min	Typ	Max	Units	Conditions
D420		HLVD Voltage on VDD Transition High-to-Low	HLVDL<3:0> = 0000	1.80	1.85	1.90	V	
			HLVDL<3:0> = 0001	2.03	2.08	2.13	V	
			HLVDL<3:0> = 0010	2.24	2.29	2.35	V	
			HLVDL<3:0> = 0011	2.40	2.46	2.53	V	
			HLVDL<3:0> = 0100	2.50	2.56	2.62	V	
			HLVDL<3:0> = 0101	2.70	2.77	2.84	V	
			HLVDL<3:0> = 0110	2.82	2.89	2.97	V	
			HLVDL<3:0> = 0111	2.95	3.02	3.10	V	
			HLVDL<3:0> = 1000	3.24	3.32	3.41	V	
			HLVDL<3:0> = 1001	3.42	3.50	3.59	V	
			HLVDL<3:0> = 1010	3.61	3.70	3.79	V	
			HLVDL<3:0> = 1011	3.82	3.91	4.10	V	
			HLVDL<3:0> = 1100	4.06	4.16	4.26	V	
			HLVDL<3:0> = 1101	4.33	4.44	4.55	V	
			HLVDL<3:0> = 1110	4.64	4.75	4.87	V	

D

E