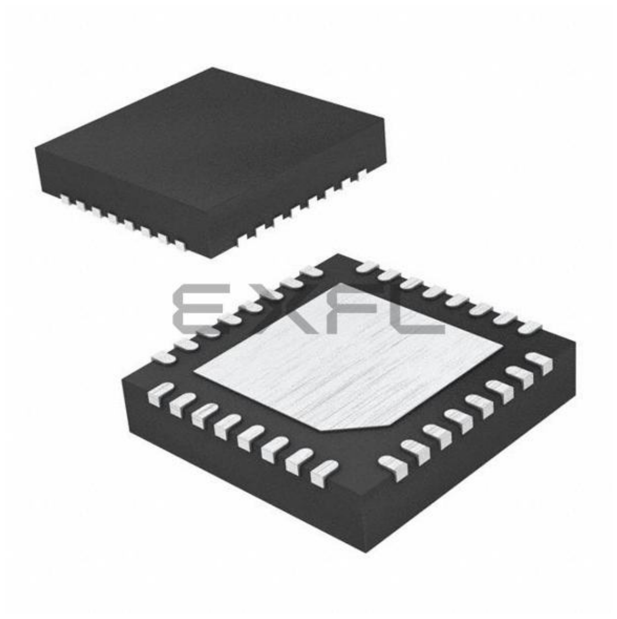
Microchip Technology - PIC18F25K80-H/MM Datasheet



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)	4V ~ 5.5V
Data Converters	A/D 8x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN-S (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25k80-h-mm

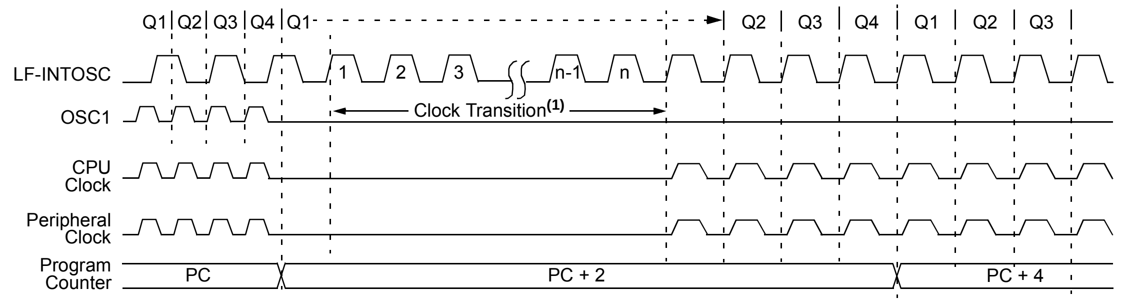
TABLE 1-5: PIC18F4XK80 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	PDIP	QFN/ TQFP			
RB6/PGC/KBI2	39	16			
RB6			I/O	ST/ CMOS	Digital I/O.
PGC			I	ST	In-Circuit Debugger and ICSP™ programming clock input pin.
KBI2			I	ST	Interrupt-on-change pin.
RB7/PGD/T3G/KBI3	40	17			
RB7			I/O	ST/ CMOS	Digital I/O.
PGD			I/O	ST	In-Circuit Debugger and ICSP™ programming data pin.
T3G			I	ST	Timer3 external clock gate input.
KBI3			I	ST	Interrupt-on-change pin.

Legend:			
I^2C™	= I^2C/SMBus input buffer	CMOS	= CMOS compatible input or output
ST	= Schmitt Trigger input with CMOS levels	Analog	= Analog input
I	= Input	O	= Output
P	= Power		

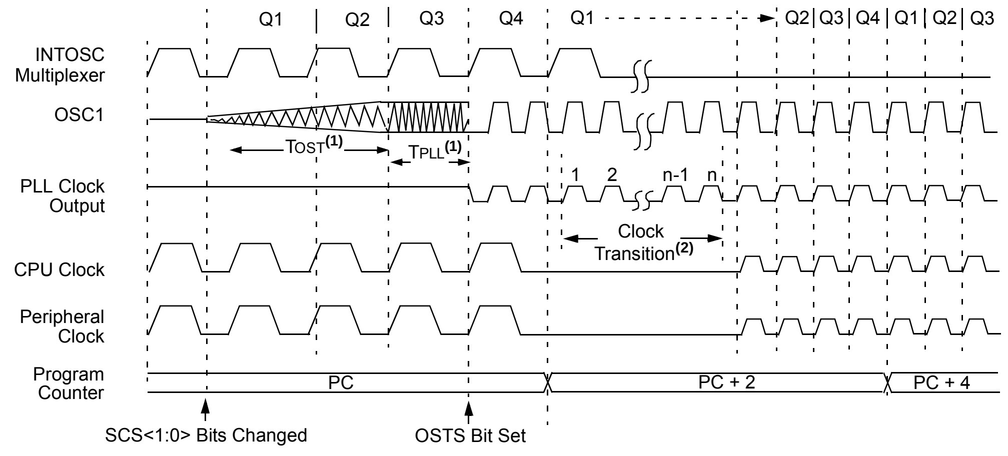
NOTES:

FIGURE 4-3: TRANSITION TIMING TO RC_RUN MODE



Note 1: Clock transition typically occurs within 2-4 TOSC.

FIGURE 4-4: TRANSITION TIMING FROM RC_RUN MODE TO PRI_RUN MODE



Note 1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.
2: Clock transition typically occurs within 2-4 TOSC.

TABLE 5-4: 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
PMD1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0000 0000`	`0000 0000`	`uuuu uuuu`
PMD2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`---- 0000`	`---- 0000`	`---- uuuu`
PADCFG1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0000 ---0`	`0000 ---0`	`uuuu ---u`
CTMUCONH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0-00 0000`	`0-00 0000`	`u-uu uuuu`
CTMUCONL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0000 0000`	`0000 0000`	`uuuu uuuu`
CTMUICON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0000 0000`	`0000 0000`	`uuuu uuuu`
CCPR2H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCPR2L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCP2CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR3H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCPR3L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCP3CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR4H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCPR4L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCP4CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR5H	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCPR5L	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`xxxx xxxx`	`uuuu uuuu`
CCP5CON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`--00 0000`	`--00 0000`	`--uu uuuu`
PSPCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0000 ----`	`0000 ----`	`uuuu ----`
MDCON	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0010 0--0`	`0010 0--0`	`uuuu u--u`
MDSRC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0--- xxxx`	`0--- xxxx`	`u--- uuuu`
MDCARH	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0xx- xxxx`	`0xx- xxxx`	`uuu- uuuu`
MDCARL	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`0xx- xxxx`	`0xx- xxxx`	`uuu- uuuu`
CANCON_RO0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`1000 0000`	`1000 0000`	`uuuu uuuu`
CANSTAT_RO0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`1000 0000`	`1000 0000`	`uuuu uuuu`
RXB1D7	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1D6	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1D5	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1D4	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1D3	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1D2	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1D1	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1D0	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXB1DLC	PIC18F2XK80	PIC18F4XK80	PIC18F6XK80	`xxxx xxxx`	`uuuu uuuu`	`xxxx xxxx`

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`'.

6.1.2 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and 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 and 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 6.1.5.1 "Computed `GOTO`"**).

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

6.1.3 RETURN ADDRESS STACK

The return address stack enables execution of any combination of up to 31 program calls and interrupts. 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. The value is also pulled off the stack on `ADDULNK` and `SUBULNK` instructions if the extended instruction set is enabled. 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 (TOS) Special Function 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.

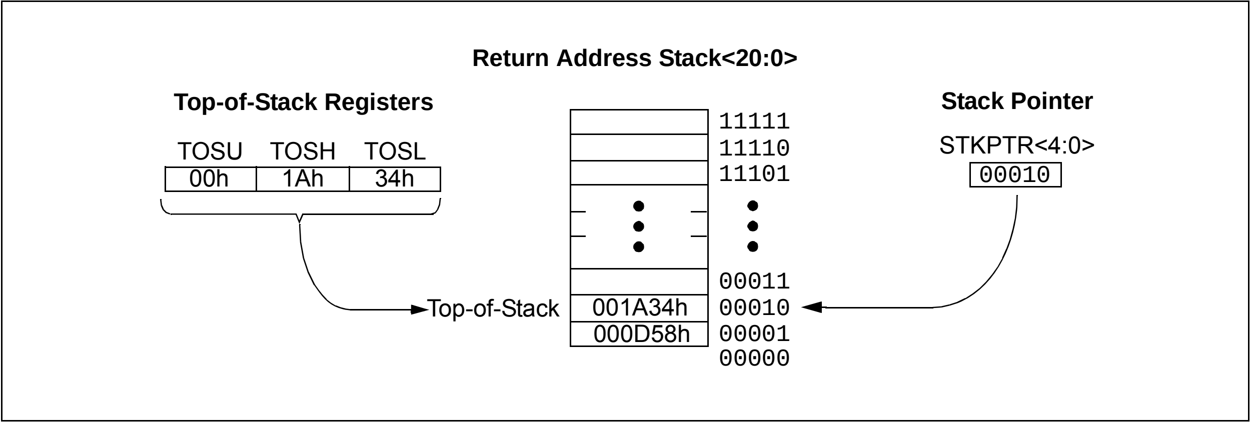
The Stack Pointer is initialized to '`00000`' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '`00000`'; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

6.1.3.1 Top-of-Stack Access

Only the top of the return address stack is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register (Figure 6-3). This allows users to implement a software stack, if necessary. After a `CALL`, `RCALL` or interrupt (or `ADDULNK` and `SUBULNK` instructions, if the extended instruction set is enabled), 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.

While accessing the stack, users must disable the Global Interrupt Enable bits to prevent inadvertent stack corruption.

FIGURE 6-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two or four bytes in program memory. The Least Significant Byte (LSB) 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 6.1.2 “Program Counter”**).

Figure 6-5 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 6-5 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. For more details on the instruction set, see **Section 29.0 “Instruction Set Summary”**.

FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY

			LSB = `1`	LSB = `0`	Word Address ↓
Program Memory Byte Locations →					000000h
					000002h
					000004h
					000006h
Instruction 1:	`MOVLW`	`055h`	0Fh	55h	000008h
Instruction 2:	`GOTO`	`0006h`	EFh	03h	00000Ah
			F0h	00h	00000Ch
Instruction 3:	`MOVFF`	`123h, 456h`	C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

6.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four, two-word instructions: `CALL`, `MOVFF`, `GOTO` and `LSFR`. In all cases, the second word of the instructions always has ‘`1111`’ as its four Most Significant bits (MSbs). The other 12 bits are literal data, usually a data memory address.

The use of ‘`1111`’ in the 4 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 word is executed by itself, a `NOP` is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 6-4 shows how this works.

Note: For information on two-word instructions in the extended instruction set, see **Section 6.5 “Program Memory and the Extended Instruction Set”**.

EXAMPLE 6-4: TWO-WORD INSTRUCTIONS

CASE 1:

Object Code	Source Code
`0110 0110 0000 0000`	`TSTFSZ    REG1        ; is RAM location 0?`
`1100 0001 0010 0011`	`MOVFF     REG1, REG2  ; No, skip this word`
`1111 0100 0101 0110`	`                      ; Execute this word as a NOP`
`0010 0100 0000 0000`	`ADDWF     REG3        ; continue code`

CASE 2:

Object Code	Source Code
`0110 0110 0000 0000`	`TSTFSZ    REG1        ; is RAM location 0?`
`1100 0001 0010 0011`	`MOVFF     REG1, REG2  ; Yes, execute this word`
`1111 0100 0101 0110`	`                      ; 2nd word of instruction`
`0010 0100 0000 0000`	`ADDWF     REG3        ; continue code`

REGISTER 10-3: INTCON3: INTERRUPT CONTROL REGISTER 3

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
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	**INT2IP:** INT2 External Interrupt Priority bit 1 = High priority 0 = Low priority
bit 6	**INT1IP:** INT1 External Interrupt Priority bit 1 = High priority 0 = Low priority
bit 5	**INT3IE:** INT3 External Interrupt Enable bit 1 = Enables the INT3 external interrupt 0 = Disables the INT3 external interrupt
bit 4	**INT2IE:** INT2 External Interrupt Enable bit 1 = Enables the INT2 external interrupt 0 = Disables the INT2 external interrupt
bit 3	**INT1IE:** INT1 External Interrupt Enable bit 1 = Enables the INT1 external interrupt 0 = Disables the INT1 external interrupt
bit 2	**INT3IF:** INT3 External Interrupt Flag bit 1 = The INT3 external interrupt occurred (must be cleared in software) 0 = The INT3 external interrupt did not occur
bit 1	**INT2IF:** INT2 External Interrupt Flag bit 1 = The INT2 external interrupt occurred (must be cleared in software) 0 = The INT2 external interrupt did not occur
bit 0	**INT1IF:** INT1 External Interrupt Flag bit 1 = The INT1 external interrupt occurred (must be cleared in software) 0 = The INT1 external interrupt did not occur

Note: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 12-4: MDCARL: MODULATION LOW CARRIER CONTROL REGISTER

R/W-0	R/W-x	R/W-x	U-0	R/W-x	R/W-x	R/W-x	R/W-x
MDCLODIS	MDCLPOL	MDCLSYNC	—	MDCL3(1)	MDCL2(1)	MDCL1(1)	MDCL0(1)
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 **MDCLODIS:** Modulator Low Carrier Output Disable bit
1 = Output signal driving the peripheral output pin (selected by MDCL<3:0> of the MDCARL register) is disabled
0 = Output signal driving the peripheral output pin (selected by MDCL<3:0> of the MDCARL register) is enabled

bit 6 **MDCLPOL:** Modulator Low Carrier Polarity Select bit
1 = Selected low carrier signal is inverted
0 = Selected low carrier signal is not inverted

bit 5 **MDCLSYNC:** Modulator Low Carrier Synchronization Enable bit
1 = Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier
0 = Modulator output is not synchronized to the low time carrier signal(1)

bit 4 **Unimplemented:** Read as ‘0’

bit 3-0 **MDCL<3:0>** Modulator Data High Carrier Selection bits(1)
1111-1001 = Reserved
1000 = CCP5 output (PWM Output mode only)
0111 = CCP4 output (PWM Output mode only)
0110 = CCP3 output (PWM Output mode only)
0101 = CCP2 output (PWM Output mode only)
0100 = ECCP1 output (PWM Output mode only)
0011 = Reference clock module signal
0010 = MDCIN2 port pin
0001 = MDCIN1 port pin
0000 = VSS

Note 1: Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

TABLE 12-1: SUMMARY OF REGISTERS ASSOCIATED WITH DATA SIGNAL MODULATOR MODE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MDCARH	MDCHODIS	MDCHPOL	MDCHSYNC	—	MDCH3	MDCH2	MDCH1	MDCH0
MDCARL	MDCLODIS	MDCLPOL	MDCLSYNC	—	MDCL3	MDCL2	MDCL1	MDCL0
MDCON	MDEN	MDOE	MDSLR	MDOPOL	MDO	—	—	MDBIT
MDSRC	MDSODIS	—	—	—	MDSRC3	MDSRC2	MDSRC1	MDSRC0
PMD2	—	—	—	—	MODMD	ECANMD	CMP2MD	CMP1MD

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used in the Data Signal Modulator mode.

For more details on selecting the optimum C1 and C2 for a given crystal, see the crystal manufacture's applications information. The optimum value depends in part on the amount of parasitic capacitance in the circuit, which is often unknown. For that reason, it is highly recommended that thorough testing and validation of the oscillator be performed after values have been selected.

14.5.1 USING SOSC AS A CLOCK SOURCE

The SOSC oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS<1:0> (OSCCON<1:0>), to '`01`', the device switches to SEC_RUN mode and both the CPU and peripherals are clocked from the SOSC oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a `SLEEP` instruction is executed, the device enters SEC_IDLE mode. Additional details are available in **Section 4.0 "Power-Managed Modes"**.

Whenever the SOSC oscillator is providing the clock source, the SOSC System Clock Status flag, SOSCRUN (OSCCON2<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source currently being used by the Fail-Safe Clock Monitor.

If the Clock Monitor is enabled and the SOSC oscillator fails while providing the clock, polling the SOCSRUN bit will indicate whether the clock is being provided by the SOSC oscillator or another source.

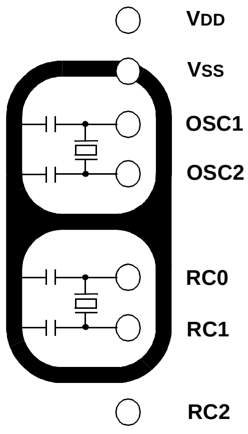
14.5.2 SOSC OSCILLATOR LAYOUT CONSIDERATIONS

The SOSC oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity. This is especially true when the oscillator is configured for extremely Low-Power mode, SOSCSEL<1:0> (CONFIG1L<4:3>) = `01`.

The oscillator circuit, displayed in Figure 14-2, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator, it may help to have a grounded guard ring around the oscillator circuit. The guard, as displayed in Figure 14-3, could be used on a single-sided PCB or in addition to a ground plane. (Examples of a high-speed circuit include the ECCP1 pin, in Output Compare or PWM mode, or the primary oscillator, using the OSC2 pin.)

FIGURE 14-3: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING



Note: Not drawn to scale.

In the Low Drive Level mode, SOSCSEL<1:0> = `01`, it is critical that RC2 I/O pin signals be kept away from the oscillator circuit. Configuring RC2 as a digital output, and toggling it, can potentially disturb the oscillator circuit, even with a relatively good PCB layout. If possible, either leave RC2 unused or use it as an input pin with a slew rate limited signal source. If RC2 must be used as a digital output, it may be necessary to use the Higher Drive Level Oscillator mode (SOSCSEL<1:0> = `11`) with many PCB layouts.

Even in the Higher Drive Level mode, careful layout procedures should still be followed when designing the oscillator circuit.

In addition to dV/dt induced noise considerations, it is important to ensure that the circuit board is clean. Even a very small amount of conductive, soldering flux residue can cause PCB leakage currents that can overwhelm the oscillator circuit.

14.6 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

EXAMPLE 18-2: CURRENT CALIBRATION ROUTINE

```
#include "p18cxxx.h"

#define COUNT 500                        //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define RCAL .027                        //R value is 4200000 (4.2M)
                                         //scaled so that result is in
                                         //1/100th of uA
#define ADSCALE 1023                     //for unsigned conversion 10 sig bits
#define ADREF 3.3                        //Vdd connected to A/D Vr+

int main(void)
{
    int i;
    int j = 0; //index for loop
    unsigned int Vread = 0;
    double VTot = 0;
    float Vavg=0, Vcal=0, CTMUISrc = 0;  //float values stored for calcs

//assume CTMU and A/D have been setup correctly
//see Example 25-1 for CTMU & A/D setup
setup();

CTMUCONHbits.CTMUEN = 1;                 //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;        //drain charge on the circuit
        DELAY;                           //wait 125us
        CTMUCONHbits.IDISSEN = 0;        //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;       //Begin charging the circuit
                                         //using CTMU current source
        DELAY;                           //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;       //Stop charging circuit

        PIR1bits.ADIF = 0;               //make sure A/D Int not set
        ADCON0bits.GO=1;                 //and begin A/D conv.
        while(!PIR1bits.ADIF);           //Wait for A/D convert complete

        Vread = ADRES;                   //Get the value from the A/D
        PIR1bits.ADIF = 0;               //Clear A/D Interrupt Flag
        VTot += Vread;                   //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);         //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL;                //CTMUISrc is in 1/100ths of uA

}
```

REGISTER 21-4: SSPCON1: MSSP CONTROL REGISTER 1 (I^{2}C™ 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
WCOL	SSPOV	SSPEN(1)	CKP	SSPM3(2)	SSPM2(2)	SSPM1(2)	SSPM0(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 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
`1` = A write to the SSPBUF register was attempted while the I^{2}C conditions were not valid for a transmission to be started (must be cleared in software)
`0` = No collision

In Slave Transmit mode:
`1` = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
`0` = No collision

In Receive mode (Master or Slave modes):
This is a “don’t care” bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:
`1` = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)
`0` = No overflow

In Transmit mode:
This is a “don’t care” bit in Transmit mode.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(1)

`1` = Enables the serial port and configures the SDA and SCL pins as the serial port pins
`0` = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCK Release Control bit

In Slave mode:
`1` = Releases clock
`0` = Holds clock low (clock stretch), used to ensure data setup time

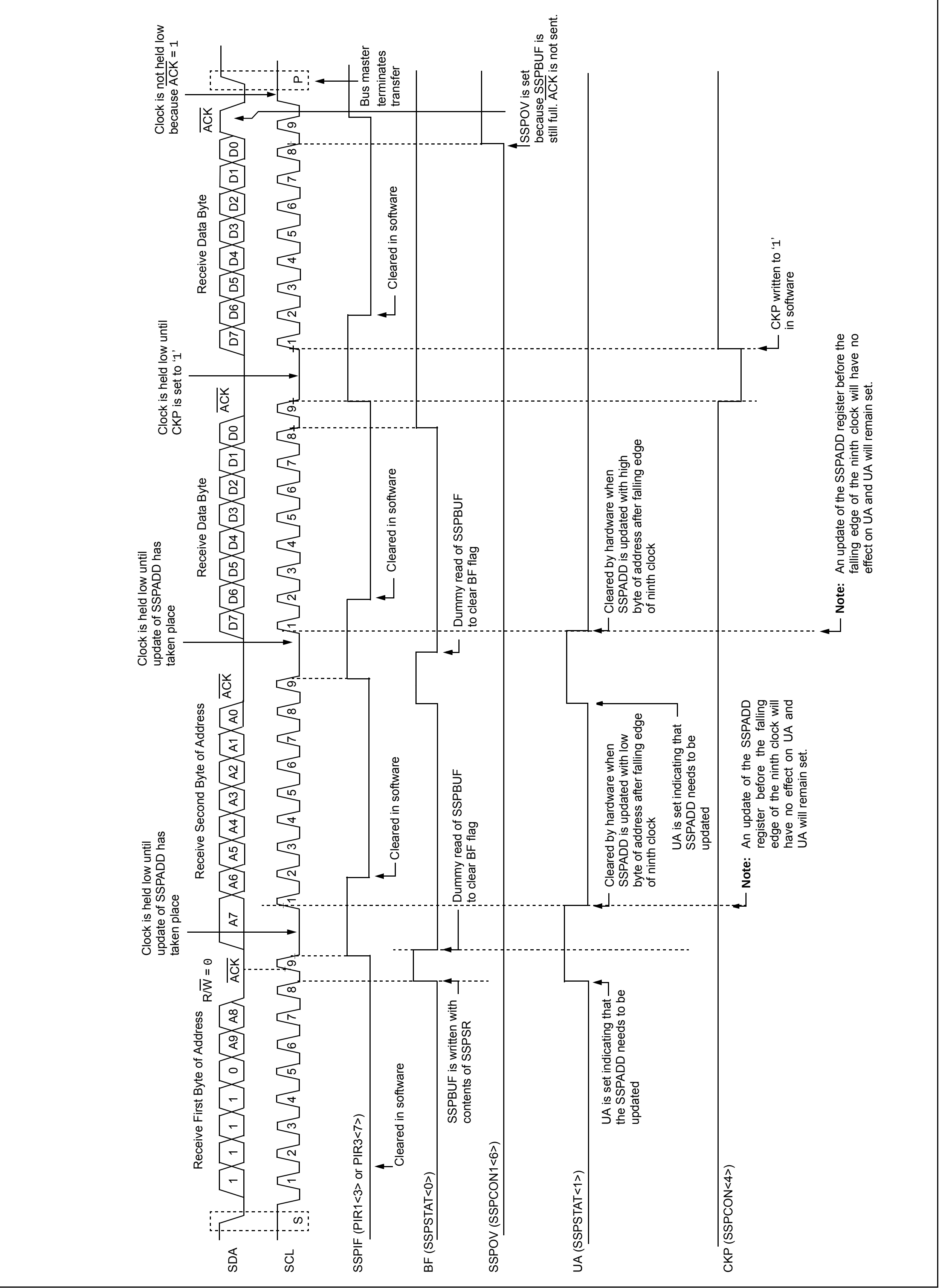
In Master mode:
Unused in this mode.

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(2)

`1111` = I^{2}C Slave mode, 10-bit address with Start and Stop bit interrupts enabled
`1110` = I^{2}C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
`1011` = I^{2}C Firmware Controlled Master mode (slave Idle)
`1001` = Load SSPMSK register at SSPADD SFR address(3,4)
`1000` = I^{2}C Master mode, clock = FOSC/(4 * (SSPADD + 1))
`0111` = I^{2}C Slave mode, 10-bit address
`0110` = I^{2}C Slave mode, 7-bit address

Note 1: When enabled, the SDA and SCL pins must be configured as inputs.
2: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.
3: When SSPM<3:0> = `1001`, any reads or writes to the SSPADD SFR address actually access the SSPMSK register.
4: This mode is only available when 7-Bit Address Masking mode is selected (MSSPMSK Configuration bit is ‘`1`’).

FIGURE 21-16: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)


SDA
SCL
SSPIF (PIR1<3> or PIR3<7>)
BF (SSPSTAT<0>)
SSPOV (SSPCON1<6>)
UA (SSPSTAT<1>)
CKP (SSPCON<4>)
S
P
Receive First Byte of Address
1 1 1 1 0 A9 A8
R/W = 0
ACK
Receive Second Byte of Address
A7 A6 A5 A4 A3 A2 A1 A0
Receive Data Byte
D7 D6 D5 D4 D3 D2 D1 D0
1 2 3 4 5 6 7 8 9
Clock is held low until update of SSPADD has taken place
Clock is held low until CKP is set to '1'
Clock is not held low because ACK = 1
Cleared in software
SSPBUF is written with contents of SSPSR
Dummy read of SSPBUF to clear BF flag
UA is set indicating that the SSPADD needs to be updated
Cleared by hardware when SSPADD is updated with low byte of address after falling edge of ninth clock
Cleared by hardware when SSPADD is updated with high byte of address after falling edge of ninth clock
Note: An update of the SSPADD register before the falling edge of the ninth clock will have no effect on UA and UA will remain set.
Note: An update of the SSPADD register before the falling edge of the ninth clock will have no effect on UA and UA will remain set.
CKP written to '1' in software
Bus master terminates transfer
SSPOV is set because SSPBUF is still full. ACK is not sent.

27.15.6.1 Receiver Overflow

An overflow condition occurs when the MAB has assembled a valid received message (the message meets the criteria of the acceptance filters) and the receive buffer associated with the filter is not available for loading of a new message. The associated RXBnOVFL bit in the COMSTAT register will be set to indicate the overflow condition. This bit must be cleared by the MCU.

27.15.6.2 Receiver Warning

The receive error counter has reached the MCU warning limit of 96.

27.15.6.3 Transmitter Warning

The transmit error counter has reached the MCU warning limit of 96.

27.15.6.4 Receiver Bus Passive

This will occur when the device has gone to the error-passive state because the receive error counter is greater or equal to 128.

27.15.6.5 Transmitter Bus Passive

This will occur when the device has gone to the error-passive state because the transmit error counter is greater or equal to 128.

27.15.6.6 Bus-Off

The transmit error counter has exceeded 255 and the device has gone to bus-off state.

TABLE 29-2: PIC18F66K80 FAMILY INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	`0010`	`01da`	`ffff`	`ffff`	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	`0010`	`00da`	`ffff`	`ffff`	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	`0001`	`01da`	`ffff`	`ffff`	Z, N	1,2
CLRF	f, a	Clear f	1	`0110`	`101a`	`ffff`	`ffff`	Z	2
COMF	f, d, a	Complement f	1	`0001`	`11da`	`ffff`	`ffff`	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, Skip =	1 (2 or 3)	`0110`	`001a`	`ffff`	`ffff`	None	4
CPFSGT	f, a	Compare f with WREG, Skip >	1 (2 or 3)	`0110`	`010a`	`ffff`	`ffff`	None	4
CPFSLT	f, a	Compare f with WREG, Skip <	1 (2 or 3)	`0110`	`000a`	`ffff`	`ffff`	None	1, 2
DECF	f, d, a	Decrement f	1	`0000`	`01da`	`ffff`	`ffff`	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	`0010`	`11da`	`ffff`	`ffff`	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	`0100`	`11da`	`ffff`	`ffff`	None	1, 2
INCF	f, d, a	Increment f	1	`0010`	`10da`	`ffff`	`ffff`	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	`0011`	`11da`	`ffff`	`ffff`	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	`0100`	`10da`	`ffff`	`ffff`	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	`0001`	`00da`	`ffff`	`ffff`	Z, N	1, 2
MOVF	f, d, a	Move f	1	`0101`	`00da`	`ffff`	`ffff`	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	`1100` `1111`	`ffff` `ffff`	`ffff` `ffff`	`ffff` `ffff`	None	
MOVWF	f, a	Move WREG to f	1	`0110`	`111a`	`ffff`	`ffff`	None	
MULWF	f, a	Multiply WREG with f	1	`0000`	`001a`	`ffff`	`ffff`	None	1, 2
NEGF	f, a	Negate f	1	`0110`	`110a`	`ffff`	`ffff`	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	`0011`	`01da`	`ffff`	`ffff`	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	`0100`	`01da`	`ffff`	`ffff`	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	`0011`	`00da`	`ffff`	`ffff`	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	`0100`	`00da`	`ffff`	`ffff`	Z, N	
SETF	f, a	Set f	1	`0110`	`100a`	`ffff`	`ffff`	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with Borrow	1	`0101`	`01da`	`ffff`	`ffff`	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	`0101`	`11da`	`ffff`	`ffff`	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with Borrow	1	`0101`	`10da`	`ffff`	`ffff`	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap Nibbles in f	1	`0011`	`10da`	`ffff`	`ffff`	None	4
TSTFSZ	f, a	Test f, Skip if 0	1 (2 or 3)	`0110`	`011a`	`ffff`	`ffff`	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	`0001`	`10da`	`ffff`	`ffff`	Z, N	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '`1`' for a pin configured as an input and is driven low by an external device, the data will be written back with a '`0`'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = `1`), the prescaler will be cleared if assigned.

3: If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

RCALL	**Relative Call**
Syntax:	RCALL n
Operands:	-1024 ≤ n ≤ 1023
Operation:	(PC) + 2 → TOS, (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	`1101` `1nnn` `nnnn` `nnnn`
Description:	Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' 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 a two-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

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

Example: `HERE     RCALL Jump`

Before Instruction
PC = Address (`HERE`)
After Instruction
PC = Address (`Jump`)
TOS = Address (`HERE + 2`)

RESET	**Reset**
Syntax:	RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.
Status Affected:	All
Encoding:	`0000` `0000` `1111` `1111`
Description:	This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset in software.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start reset	No operation	No operation

Example: `RESET`

After Instruction
Registers = Reset Value
Flags* = Reset Value

30.7 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

30.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

30.9 MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE).

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

30.10 PICkit 3 In-Circuit Debugger/ Programmer and PICkit 3 Debug Express

The MPLAB PICkit 3 allows debugging and programming of PIC® and dsPIC® Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via an Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

30.11 PICkit 2 Development Programmer/Debugger and PICkit 2 Debug Express

The PICkit™ 2 Development Programmer/Debugger is a low-cost development tool with an easy to use interface for programming and debugging Microchip's Flash families of microcontrollers. The full featured Windows® programming interface supports baseline (PIC10F, PIC12F5xx, PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33, and PIC32 families of 8-bit, 16-bit, and 32-bit microcontrollers, and many Microchip Serial EEPROM products. With Microchip's powerful MPLAB Integrated Development Environment (IDE) the PICkit™ 2 enables in-circuit debugging on most PIC® microcontrollers. In-Circuit-Debugging runs, halts and single steps the program while the PIC microcontroller is embedded in the application. When halted at a breakpoint, the file registers can be examined and modified.

The PICkit 2 Debug Express include the PICkit 2, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

30.12 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular, detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an MMC card for file storage and data applications.

30.13 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (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)[2,3]						
	PIC18**LF**XXK80	4	8	μA	-40°C	VDD = 1.8V[4] Regulator Disabled	FOSC = 31 kHz (**RC_RUN** mode, LF-INTOSC)
		4	8	μA	+25°C		
		4	8	μA	+60°C		
		5	9	μA	+85°C		
		9	12	μA	+125°C		
	PIC18**LF**XXK80	7	11	μA	-40°C	VDD = 3.3V[4] Regulator Disabled	
		7	11	μA	+25°C		
		7	11	μA	+60°C		
		8	12	μA	+85°C		
		13	15	μA	+125°C		
	PIC18**F**XXK80	51	150	μA	-40°C	VDD = 3.3V[5] Regulator Enabled	
		70	150	μA	+25°C		
		75	150	μA	+60°C		
		80	170	μA	+85°C		
		88	190	μA	+125°C		
	PIC18**F**XXK80	75	180	μA	-40°C	VDD = 5V[5] Regulator Enabled	
		75	180	μA	+25°C		
		75	180	μA	+60°C		
		80	190	μA	+85°C		
		95	200	μA	+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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, SOSC oscillator, BOR, etc.).

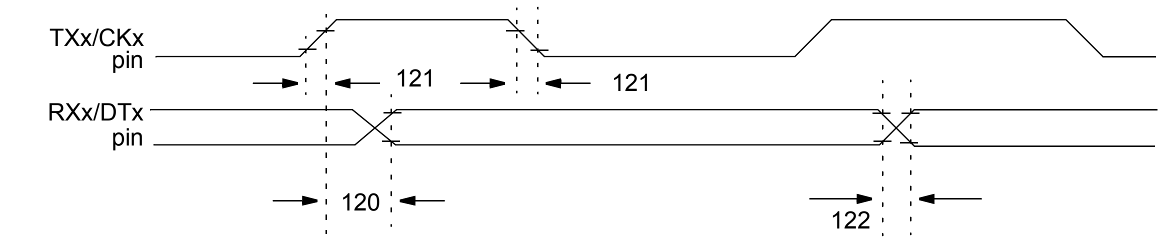
2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD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

3: Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4: For LF devices, $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 1.

5: For F devices, SRETEN (WDTCON<4>) = 1 and $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 0.

FIGURE 31-19: EUSARTx SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

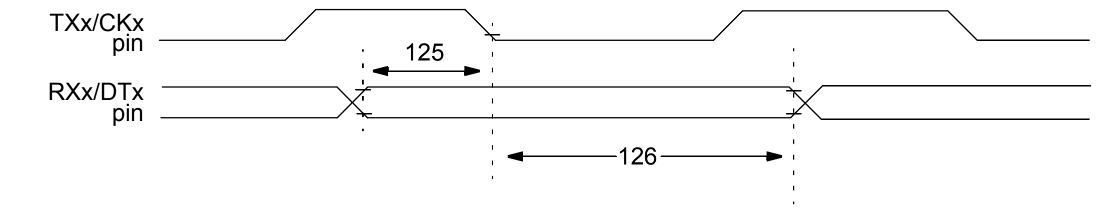


Note: Refer to Figure 31-3 for load conditions.

TABLE 31-23: EUSART/AUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	TCKH2DTV	<u>SYNC XMIT (MASTER and SLAVE)</u> Clock High to Data Out Valid	—	40	ns	
121	TCKRF	Clock Out Rise Time and Fall Time (Master mode)	—	20	ns	
122	TDTRF	Data Out Rise Time and Fall Time	—	20	ns	

FIGURE 31-20: EUSART/AUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 31-3 for load conditions.

TABLE 31-24: EUSART/AUSART SYNCHRONOUS RECEIVE REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TDTV2CKL	<u>SYNC RCV (MASTER and SLAVE)</u> Data Hold before CKx ↓ (DTx hold time)	10	—	ns	
126	TCKL2DTL	Data Hold after CKx ↓ (DTx hold time)	15	—	ns	

NOTES: