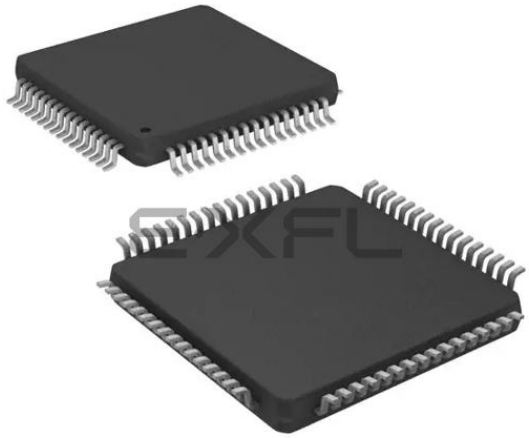
Microchip Technology - PIC18F66K40-I/PT Datasheet



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	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	60
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.5K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 45x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°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/pic18f66k40-i-pt

2.0 GUIDELINES FOR GETTING STARTED WITH PIC18(L)F6XK40 MICROCONTROLLERS

2.1 Basic Connection Requirements

Getting started with the PIC18(L)F6xK40 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins (see **Section 2.2 "Power Supply Pins"**)
- $\overline{\text{MCLR}}$ pin (see **Section 2.3 "Master Clear (MCLR) Pin"**)

These pins must also be connected if they are being used in the end application:

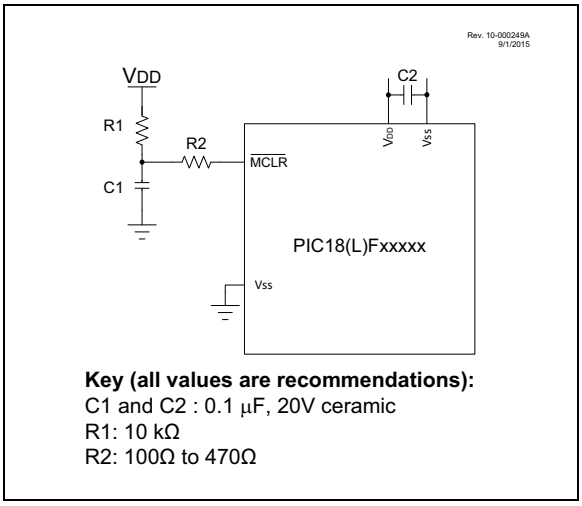
- PGC/PGD pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see **Section 2.4 "ICSP™ Pins"**)
- OSCI and OSCO pins when an external oscillator source is used (see **Section 2.5 "External Oscillator Pins"**)

Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

The minimum mandatory connections are shown in Figure 2-1.

FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS



Key (all values are recommendations):
C1 and C2 : 0.1 μF, 20V ceramic
R1: 10 kΩ
R2: 100Ω to 470Ω

2.2 Power Supply Pins

2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins (VDD and VSS) is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1 μF (100 nF), 10-20V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01 μF to 0.001 μF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 μF in parallel with 0.001 μF).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

2.2.2 TANK CAPACITORS

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7 μF to 47 μF.

REGISTER 7-6: PMD5: PMD CONTROL REGISTER 5

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	USART5MD	USART4MD	USART3MD	USART2MD	USART1MD	MSSP2MD	MSSP1MD
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

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

bit 6 **USART5MD:** Disable USART5 Module bit
1 = USART5 module disabled
0 = USART5 module enabled

bit 5 **USART4MD:** Disable USART4 Module bit
1 = USART4 module disabled
0 = USART4 module enabled

bit 4 **USART3MD:** Disable USART3 Module bit
1 = USART3 module disabled
0 = USART3 module enabled

bit 3 **USART2MD:** Disable USART2 Module bit
1 = USART2 module disabled
0 = USART2 module enabled

bit 2 **USART1MD:** Disable USART4 Module bit
1 = USART1 module disabled
0 = USART1 module enabled

bit 1 **MSSP2MD:** Disable MSSP2 Module bit
1 = MSSP2 module disabled
0 = MSSP2 module enabled

bit 0 **MSSP1MD:** Disable MSSP1 Module bit
1 = MSSP1 module disabled
0 = MSSP1 module enabled

TABLE 7-1: SUMMARY OF REGISTERS ASSOCIATED WITH PMD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
PMD0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	66
PMD1	TMR7MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD	67
PMD2	—	—	CWGMD	—	DSMMD	SMT2MD	SMT1MD	TMR8MD	68
PMD3	—	DACMD	ADCMD	—	CMP3MD	CMP2MD	CMP1MD	ZCDMD	69
PMD4	—	PWM7MD	PWM6MD	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	70
PMD5	—	USART5MD	USART4MD	USART3MD	USART2MD	USART1MD	MSSP2MD	MSSP1MD	71

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the PMD.

TABLE 10-1: PROGRAM AND DATA EEPROM MEMORY MAP

	PIC18(L)F65K40	PIC18(L)F66K40	PIC18(L)F67K40	
	PC<21:0>	PC<21:0>	PC<21:0>	
Note 1	Stack (31 levels)	Stack (31 levels)	Stack (31 levels)	**Note 1**
00 0000h	Reset Vector	Reset Vector	Reset Vector	00 0000h
• • •	• • •	• • •	• • •	• • •
00 0008h	Interrupt Vector High	Interrupt Vector High	Interrupt Vector High	00 0008h
• • •	• • •	• • •	• • •	• • •
00 0018h	Interrupt Vector Low	Interrupt Vector Low	Interrupt Vector Low	00 0018h
00 001Ah • 00 3FFFh	Program Flash Memory (16 KW)	Program Flash Memory (32 KW)	Program Flash Memory (64 KW)	00 001Ah • 00 3FFFh
00 4000h • 00 7FFFh				00 4000h • 00 7FFFh
00 8000h • 00 FFFFh	Not present(2)	Not present(2)		00 8000h • 00 FFFFh
01 0000h • 01 FFFFh				01 0000h • 01 FFFFh
02 0000h • 1F FFFFh			Not present(2)	02 0000h • 1F FFFFh
20 0000h • • • 20 000Fh	User IDs (8 Words)(3)			20 0000h • • • 20 000Fh
20 0010h • • • 2F FFFFh	Reserved			20 0010h • • • 2F FFFFh
30 0000h • • • 30 000Bh	Configuration Words (6 Words)(3)			30 0000h • • • 30 000Bh
30 000Ch • • • 30 FFFFh	Reserved			30 000Ch • • • 30 FFFFh
31 0000h • • • 31 00FFh • • • 31 03FFh	DataEEByte0 • • • DataEEByte1023			31 0000h • • • • • • 31 03FFh
31 0400h • • • 3F FFFBh	Reserved			31 0400h • • • 3F FFFBh
3F FFFCh • • • 3F FFFDh	Revision ID (1 Word)(4)			3F FFFCh • • • 3F FFFDh
3F FFFEh • • • 3F FFFFh	Device ID (1 Word)(4)			3F FFFEh • • • 3F FFFFh

Note 1: The stack is a separate SRAM panel, apart from all user memory panels.
2: The addresses do not roll over. The region is read as '0'.
3: Not code-protected.
4: Device/Revision IDs are hard-coded in silicon.

10.4 Data Memory Organization

Note:	The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 10.7 "Data Memory and the Extended Instruction Set"** for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. Figure 10-4 shows the data memory organization for the PIC18(L)F6xK40 devices.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the Bank Select Register (BSR). **Section 10.4.2 "Access Bank"** provides a detailed description of the Access RAM.

10.4.1 BANK SELECT REGISTER (BSR)

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the four Most Significant bits of a location's address; the instruction itself includes the eight Least Significant bits. Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the MOVLB instruction.

The value of the BSR indicates the bank in data memory; the eight bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 10-4.

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh will end up resetting the program counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory maps in Figure 10-4 indicate which banks are implemented.

In the core PIC18 instruction set, only the MOVFF instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, roll-overs 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 (e.g., Z, N, OV, etc.).

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.

10.6.3.3 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 FSR0H:FSR0L contains 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 either the INDF2 or POSTDEC2 register 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, 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 that they do not inadvertently change settings that might affect the operation of the device.

10.7 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect addressing with FSR0 and FSR1 also remain unchanged.

10.7.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of indirect addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of indexed addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced (‘a’ = 0) and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in direct addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

10.7.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use direct addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘1’), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in Figure 10-7.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 36.2.1 “Extended Instruction Syntax”**.

11.1.3 READING THE PROGRAM FLASH MEMORY

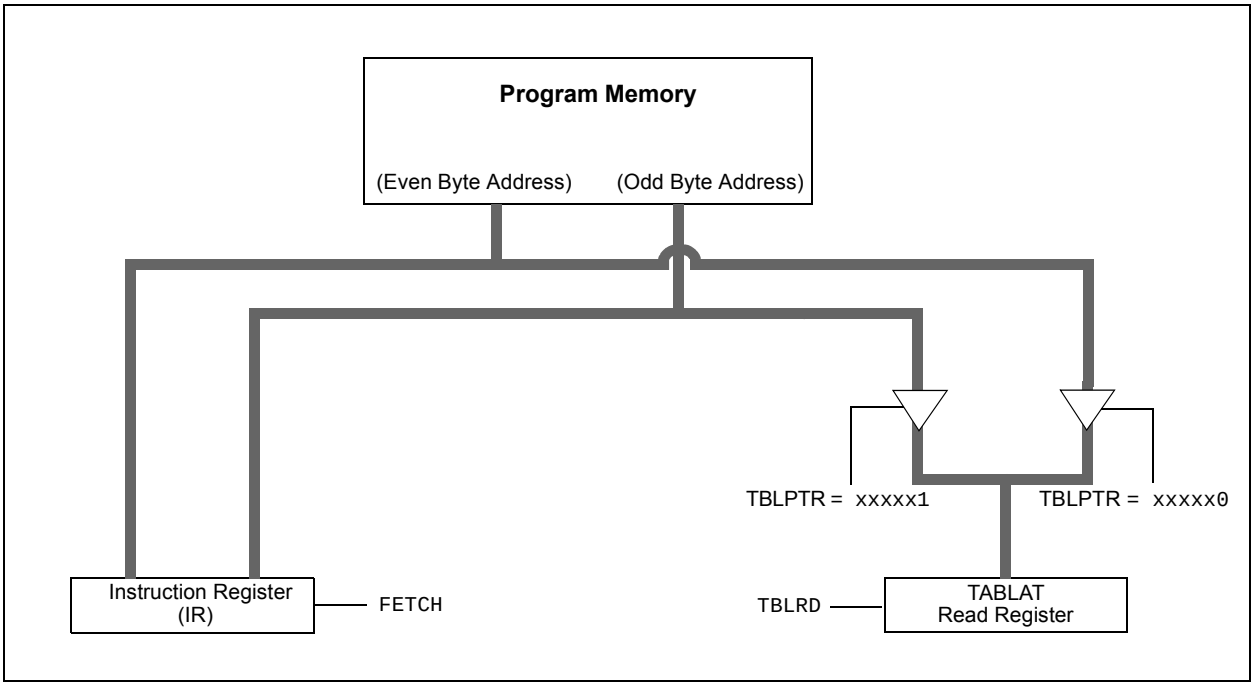
The TBLRD instruction retrieves data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

TBLPTR points to a byte address in program space. Executing TBLRD places the byte pointed to into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation.

The CPU operation is suspended during the read, and it resumes immediately after. From the user point of view, TABLAT is valid in the next instruction cycle.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 11-4 shows the interface between the internal program memory and the TABLAT.

FIGURE 11-4: READS FROM PROGRAM FLASH MEMORY



EXAMPLE 11-1: READING A PROGRAM FLASH MEMORY WORD

```
            MOVLW     CODE_ADDR_UPPER         ; Load TBLPTR with the base
            MOVWF     TBLPTRU                 ; address of the word
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
READ_WORD
            TBLRD*+                           ; read into TABLAT and increment
            MOVF      TABLAT, W               ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                           ; read into TABLAT and increment
            MOVFW     TABLAT, W               ; get data
            MOVF      WORD_ODD
```

11.3 Data EEPROM Memory

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, which is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Four SFRs are used to read and write to the data EEPROM as well as the program memory. They are:

- NVMCON1
- NVMCON2
- NVMDAT
- NVMADRL
- NVMADRH

The data EEPROM allows byte read and write. When interfacing to the data memory block, NVMDAT holds the 8-bit data for read/write and the NVMADRH:NVMADRL register pair hold the address of the EEPROM location being accessed.

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an internal programming timer; it will vary with voltage and temperature as well as from chip-to-chip. Please refer to the Data EEPROM Memory parameters in **Section 37.0 "Electrical Specifications"** for limits.

11.3.1 NVMADRL AND NVMADRH REGISTERS

The NVMADRH:NVMADRL registers are used to address the data EEPROM for read and write operations.

11.3.2 NVMCON1 AND NVMCON2 REGISTERS

Access to the data EEPROM is controlled by two registers: NVMCON1 and NVMCON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The NVMCON1 register (Register 11-1) is the control register for data and program memory access. Control bits NVMREG<1:0> determine if the access will be to program, Data EEPROM Memory or the User IDs, Configuration bits, Revision ID and Device ID.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

The WR control bit initiates write operations. The bit can be set but not cleared by software. It is cleared only by hardware at the completion of the write operation.

The NVMIF interrupt flag bit of the PIR7 register is set when the write is complete. It must be cleared by software.

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

The RD bit cannot be set when accessing program memory (NVMREG<1:0> = 0x10). Program memory is read using table read instructions. See **Section 11.1.1 "Table Reads and Table Writes"** regarding table reads.

11.3.5 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

EXAMPLE 11-5: DATA EEPROM READ

```
; Data Memory Address to read
              BCF NVMCON1, NVMREG0   ; Setup Data EEPROM Access
              BCF NVMCON1, NVMREG1   ; Setup Data EEPROM Access
              MOVF EE_ADDRL, W       ;
              MOVWF NVMADRL          ; Setup Address low byte
              MOVF EE_ADDRH, W       ;
              MOVWF NVMADRH          ; Setup Address high byte (if applicable)
              BSF NVMCON1, RD        ; Issue EE Read
              MOVF NVMDAT, W         ; W = EE_DATA
```

EXAMPLE 11-6: DATA EEPROM WRITE

```
; Data Memory Address to write
            BCF       NVMCON1, NVMREG0 ; Setup Data EEPROM access
            BCF       NVMCON1, NVMREG1 ; Setup Data EEPROM access
            MOVF      EE_ADDRL, W      ;
            MOVWF     NVMADRL          ; Setup Address low byte
            MOVF      EE_ADDRH, W      ;
            MOVWF     NVMADRH          ; Setup Address high byte (if applicable)
; Data Memory Value to write
            MOVF      EE_DATA, W       ;
            MOVWF     NVMDAT           ;
; Enable writes
            BSF       NVMCON1, WREN    ;
; Disable interrupts
            BCF       INTCON, GIE      ;
; Required unlock sequence
            MOVLW     55h              ;
            MOVWF     NVMCON2          ;
            MOVLW     AAh              ;
            MOVWF     NVMCON2          ;
; Set WR bit to begin write
            BSF       NVMCON1, WR      ;
; Wait for write to complete
            BTFSC     NVMCON1, WR
            BRA       $-2
; Enable INT
            BSF       INTCON, GIE      ;
; Disable writes
            BCF       NVMCON1, WREN    ;
```

REGISTER 13-14: SCANLADRL: SCAN LOW ADDRESS LOW BYTE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
LADR<7:0>(1, 2)							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 **LADR<7:0>**: Scan Start/Current Address bits(1, 2)
Least Significant bits of the current address to be fetched from, value increments on each fetch of memory

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).
2: While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 13-15: SCANHADRU: SCAN HIGH ADDRESS UPPER BYTE REGISTER

U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	HADR<21:16>					
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **HADR<21:16>:** Scan End Address bits(1, 2)
Upper bits of the address at the end of the designated scan

Note 1: Registers SCANHADRU/H/L form a 22-bit value but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).
2: While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

20.0 TIMER2/4/6/8 MODULE

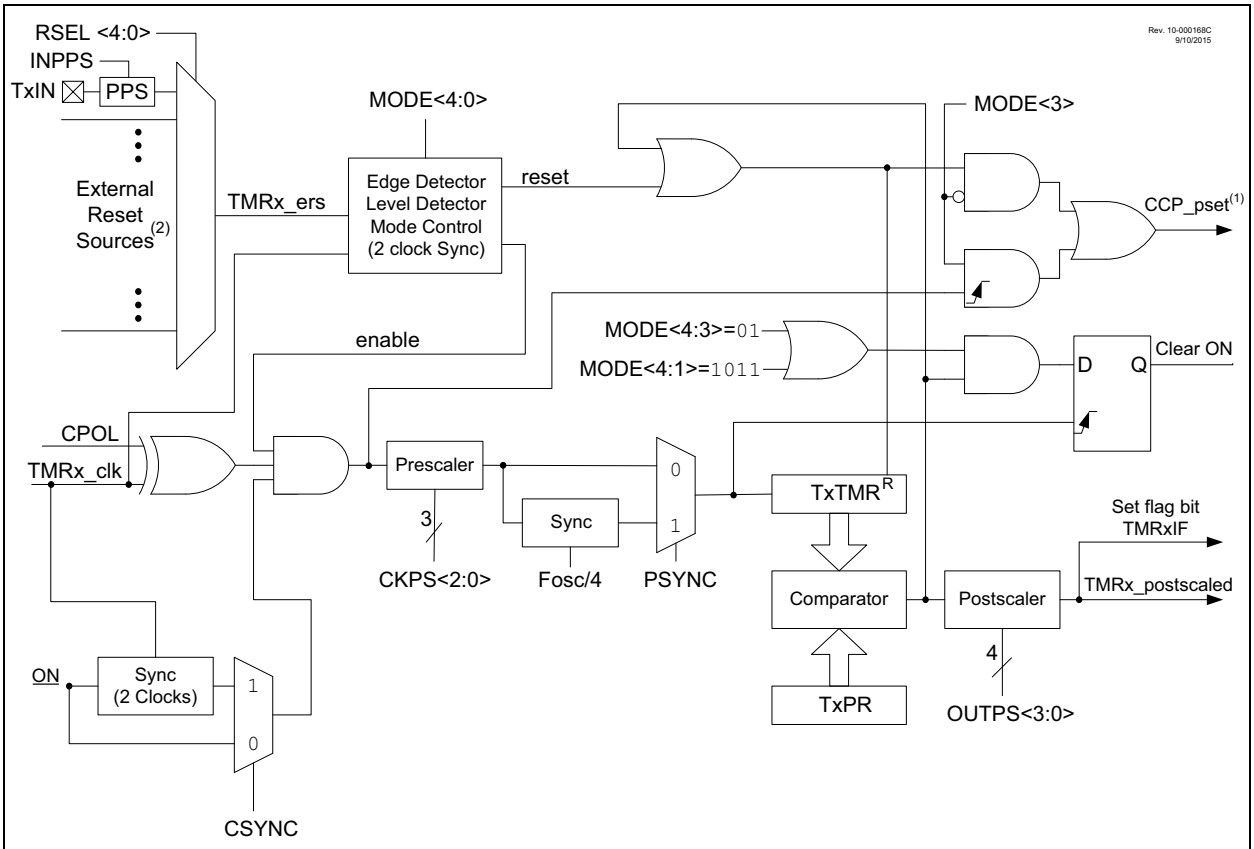
The Timer2/4/6/8 modules are 8-bit timers that can operate as free-running period counters or in conjunction with external signals that control start, run, freeze, and reset operation in One-Shot and Monostable modes of operation. Sophisticated waveform control such as pulse density modulation are possible by combining the operation of these timers with other internal peripherals such as the comparators and CCP modules. Features of the timer include:

- 8-bit timer register
- 8-bit period register
- Selectable external hardware timer Resets
- Programmable prescaler (1:1 to 1:128)
- Programmable postscaler (1:1 to 1:16)
- Selectable synchronous/asynchronous operation
- Alternate clock sources
- Interrupt-on-period
- Three modes of operation:
  - Free Running Period
  - One-shot
  - Monostable

See Figure 20-1 for a block diagram of Timer2. See Figure 20-2 for the clock source block diagram.

Note:	Four identical Timer2 modules are implemented on this device. The timers are named Timer2, Timer4, Timer6 and Timer8. All references to Timer2 apply as well to Timer4, Timer6 and Timer8. All references to PR2 apply equally to other timers as well.

FIGURE 20-1: TIMER2 BLOCK DIAGRAM



Note 1: Signal to the CCP to trigger the PWM pulse.
2: See Table 20.5 for description of CCP interaction in the different TMR modes.
3: See Table 20.5 for external Reset sources.

REGISTER 21-6: CCPRxH: CCPx REGISTER HIGH BYTE

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
CCPRx<15:8>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0

MODE = Capture Mode:
CCPRxH<7:0>: MSB of captured TMR1 value
MODE = Compare Mode:
CCPRxH<7:0>: MSB compared to TMR1 value
MODE = PWM Mode && FMT = 0:
CCPRxH<7:2>: Not used
CCPRxH<1:0>: CCPW<9:8> – Pulse-Width MS 2 bits
MODE = PWM Mode && FMT = 1:
CCPRxH<7:0>: CCPW<9:2> – Pulse-Width MS 8 bits

24.3 Clock Source

The clock source is used to drive the dead-band timing circuits. The CWG module allows the following clock sources to be selected:

- FOSC (system clock)
- HFINTOSC

When the HFINTOSC is selected, the HFINTOSC will be kept running during Sleep. Therefore, CWG modes requiring dead band can operate in Sleep, provided that the CWG data input is also active during Sleep.The clock sources are selected using the CS bit of the CWG1CLKCON register (Register 24-3). The system clock FOSC, is disabled in Sleep and thus dead-band control cannot be used.

24.4 Selectable Input Sources

The CWG generates the output waveforms from the input sources in Table 24-1.

TABLE 24-1: SELECTABLE INPUT SOURCES

Source Peripheral	Signal Name	ISM<3:0>
CWG1PPS	Pin selected by CWG1PPS	0000
CCP1	CCP1 Output	0001
CCP2	CCP2 Output	0010
CCP3	CCP3 Output	0011
CCP4	CCP4 Output	0100
CCP5	CCP5 Output	0101
PWM6	PWM6 Output	0110
PWM7	PWM7 Output	0111
CMP1	Comparator 1 Output	1000
CMP2	Comparator 2 Output	1001
CMP3	Comparator 3 Output	1010
DSM	Data signal modulator output	1011
Reserved		1100-1111

The input sources are selected using the ISM<3:0> bits in the CWG1ISM register (Register 24-4).

24.5 Output Control

24.5.1 CWG OUTPUTS

Each CWG output can be routed to a Peripheral Pin Select (PPS) output via the RxyPPS register (see **Section 17.0 "Peripheral Pin Select (PPS) Module"**).

24.5.2 POLARITY CONTROL

The polarity of each CWG output can be selected independently. When the output polarity bit is set, the corresponding output is active-high. Clearing the output polarity bit configures the corresponding output as active-low. However, polarity does not affect the override levels. Output polarity is selected with the POLy bits of the CWG1CON1. Auto-shutdown and steering options are unaffected by polarity.

24.6 Dead-Band Control

The dead-band control provides non-overlapping PWM signals to prevent shoot-through current in PWM switches. Dead-band operation is employed for Half-Bridge and Full-Bridge modes. The CWG contains two 6-bit dead-band counters. One is used for the rising edge of the input source control in Half-Bridge mode or for reverse dead-band Full-Bridge mode. The other is used for the falling edge of the input source control in Half-Bridge mode or for forward dead band in Full-Bridge mode.

Dead band is timed by counting CWG clock periods from zero up to the value in the rising or falling dead-band counter registers. See CWG1DBR and CWG1DBF registers, respectively.

24.6.1 DEAD-BAND FUNCTIONALITY IN HALF-BRIDGE MODE

In Half-Bridge mode, the dead-band counters dictate the delay between the falling edge of the normal output and the rising edge of the inverted output. This can be seen in Figure 24-2.

24.6.2 DEAD-BAND FUNCTIONALITY IN FULL-BRIDGE MODE

In Full-Bridge mode, the dead-band counters are used when undergoing a direction change. The MODE<0> bit of the CWG1CON0 register can be set or cleared while the CWG is running, allowing for changes from Forward to Reverse mode. The CWG1A and CWG1C signals will change immediately upon the first rising input edge following a direction change, but the modulated signals (CWG1B or CWG1D, depending on the direction of the change) will experience a delay dictated by the dead-band counters.

26.6 Carrier Source Polarity Select

The signal provided from any selected input source for the carrier high and carrier low signals can be inverted. Inverting the signal for the carrier high source is enabled by setting the MDCHPOL bit of the MDCON1 register. Inverting the signal for the carrier low source is enabled by setting the MDCLPOL bit of the MDCON1 register.

26.7 Programmable Modulator Data

The MDBIT of the MDCON0 register can be selected as the source for the modulator signal. This gives the user the ability to program the value used for modulation.

26.8 Modulated Output Polarity

The modulated output signal provided on the DSM pin can also be inverted. Inverting the modulated output signal is enabled by setting the MDOPOL bit of the MDCON0 register.

26.9 Operation in Sleep Mode

The DSM module is not affected by Sleep mode. The DSM can still operate during Sleep, if the Carrier and Modulator input sources are also still operable during Sleep. Refer to **Section 6.0 "Power-Saving Operation Modes"** for more details.

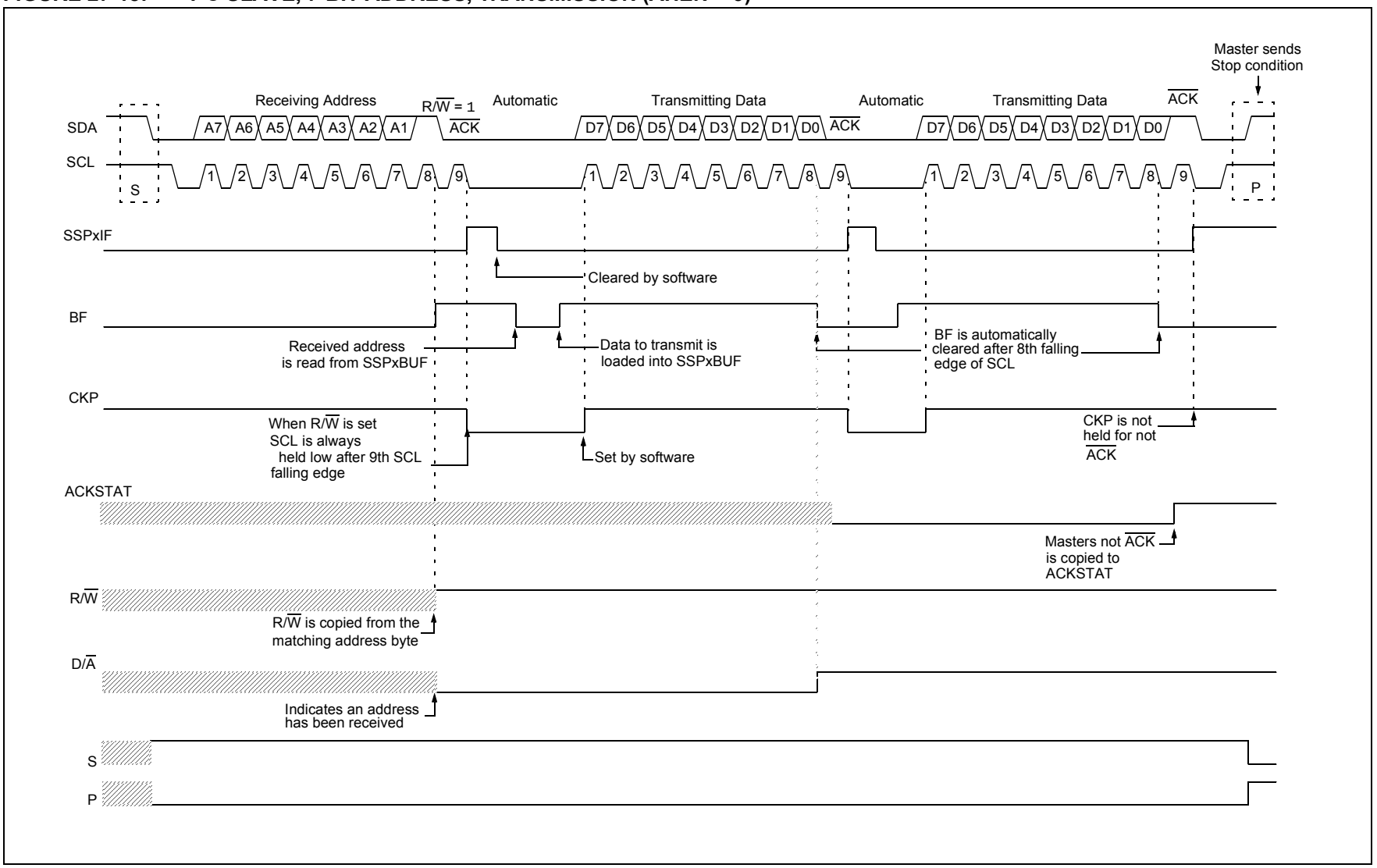
26.10 Effects of a Reset

Upon any device Reset, the DSM module is disabled. The user's firmware is responsible for initializing the module before enabling the output. The registers are reset to their default values.

26.11 Peripheral Module Disable

The DSM module can be completely disabled using the PMD module to achieve maximum power saving. The DSMMD bit of PMD2 (Register 7-3) when set disables the DSM module completely. When enabled again all the registers of the DSM module default to POR status.

FIGURE 27-18: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)



32.2.6 ADC CONVERSION PROCEDURE (BASIC MODE)

This is an example procedure for using the ADC to perform an Analog-to-Digital conversion:

1. Configure Port:
   - Disable pin output driver (Refer to the TRISx register)
   - Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   - Select ADC conversion clock
   - Configure voltage reference
   - Select ADC input channel (precharge+acquisition)
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt (PEIE bit)
   - Enable global interrupt (GIE bit)**(1)**
4. If ADACQ = 0, software must wait the required acquisition time**(2)**.
5. Start conversion by setting the ADGO bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the ADGO bit
   - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

2: Refer to **Section 32.3 "ADC Acquisition Requirements"**.

EXAMPLE 32-1: ADC CONVERSION

```
;This code block configures the ADC
;for polling, VDD and VSS references, FRC
;oscillator and AN0 input.
;
;Conversion start & polling for completion
;are included.
;
BANKSEL   ADCON1       ;
MOVLW     B'11110000'  ;Right justify,
                       ;FRC oscillator
MOVWF     ADCON1       ;Vdd and Vss Vref
BANKSEL   TRISA        ;
BSF       TRISA,0      ;Set RA0 to input
BANKSEL   ANSEL        ;
BSF       ANSEL,0      ;Set RA0 to analog
BANKSEL   ADCON0       ;
MOVLW     B'00000001'  ;Select channel AN0
MOVWF     ADCON0       ;Turn ADC On
CALL      SampleTime   ;Acquisiton delay
BSF       ADCON0,ADGO  ;Start conversion
BTFSC     ADCON0,ADGO  ;Is conversion done?
GOTO      $-1          ;No, test again
BANKSEL   ADRESH       ;
MOVF      ADRESH,W     ;Read upper 2 bits
MOVWF     RESULTHI     ;store in GPR space
BANKSEL   ADRESL       ;
MOVF      ADRESL,W     ;Read lower 8 bits
MOVWF     RESULTLO     ;Store in GPR space
```

32.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 32-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 32-4. **The maximum recommended impedance for analog sources is 10 kΩ.** As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be completed before the conversion can be started. To calculate the minimum acquisition time, Equation 32-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

EQUATION 32-1: ACQUISITION TIME EXAMPLE

Assumptions: $Temperature = 50°C$ *and external impedance of* $10k\Omega$ *5.0V* V_{DD}

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$
$$= 2\mu s + T_C + [(Temperature - 25°C)(0.05\mu s/°C)]$$

The value for T_C *can be approximated with the following equations:*

$$V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) = V_{CHOLD}$$;[1] V_{CHOLD} *charged to within 1/2 lsb*

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD}$$;[2] V_{CHOLD} *charge response to* $V_{APPLIED}$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right)$$;*combining [1] and [2]*

Note: Where n = number of bits of the ADC.

Solving for T_C*:*

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln(1/2047)$$
$$= -10pF(1k\Omega + 7k\Omega + 10k\Omega)\ ln(0.0004885)$$
$$= 1.37\mu s$$

Therefore:

$$T_{ACQ} = 2\mu s + 892ns + [(50°C - 25°C)(0.05\mu s/°C)]$$
$$= 4.62\mu s$$

Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

32.5.5 BURST AVERAGE MODE

The Burst Average mode (ADMD = 011) acts the same as the Average mode in most respects. The one way it differs is that it continuously retriggers ADC sampling until the ADCNT value is greater than or equal to ADRPT, even if Continuous Sampling mode (see **Section 32.5.8 "Continuous Sampling mode"**) is not enabled. This allows for a threshold comparison on the average of a short burst of ADC samples.

32.5.6 LOW-PASS FILTER MODE

The Low-pass Filter mode (ADMD = 100) acts similarly to the Average mode in how it handles samples (accumulates samples until ADCNT value greater than or equal to ADRPT, then triggers threshold comparison), but instead of a simple average, it performs a low-pass filter operation on all of the samples, reducing the effect of high-frequency noise on the average, then performs a threshold comparison on the results. (see Table 32-3 for a more detailed description of the mathematical operation). In this mode, the ADCRS bits determine the cut-off frequency of the low-pass filter (as demonstrated by Table 32-4).

32.5.7 THRESHOLD COMPARISON

At the end of each computation:

- The conversion results are latched and held stable at the end-of-conversion.
- The error is calculated based on a difference calculation which is selected by the ADCALC<2:0> bits in the ADCON3 register. The value can be one of the following calculations (see Register 32-4 for more details):
  - The first derivative of single measurements
  - The CVD result in CVD mode
  - The current result vs. a setpoint
  - The current result vs. the filtered/average result
  - The first derivative of the filtered/average value
  - Filtered/average value vs. a setpoint
- The result of the calculation (ADERR) is compared to the upper and lower thresholds, ADUTH<ADUTHH:ADUTHL> and ADLTH<ADLTHH:ADLTHL> registers, to set the ADUTHR and ADLTHR flag bits. The threshold logic is selected by ADTMD<2:0> bits in the ADCON3 register. The threshold trigger option can be one of the following:
  - Never interrupt
  - Error is less than lower threshold
  - Error is greater than or equal to lower threshold
  - Error is between thresholds (inclusive)
  - Error is outside of thresholds
  - Error is less than or equal to upper threshold
  - Error is greater than upper threshold
  - Always interrupt regardless of threshold test results
  - If the threshold condition is met, the threshold interrupt flag ADTIF is set.

Note 1:	The threshold tests are signed operations.
2:	If ADAOV is set, a threshold interrupt is signaled.

BZ **Branch if Zero**

Syntax: BZ n

Operands: -128 ≤ n ≤ 127

Operation: if ZERO bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0000	nnnn	nnnn

Description: If the ZERO bit is '1', 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 2-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    BZ   Jump
Before Instruction
    PC          =   address (HERE)
After Instruction
    If ZERO     =   1;
        PC      =   address (Jump)
    If ZERO     =   0;
        PC      =   address (HERE + 2)
```

CALL **Subroutine Call**

Syntax: CALL k {,s}

Operands: 0 ≤ k ≤ 1048575
s ∈ [0,1]

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(W) → WS,
(Status) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	110s	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example:
```
HERE    CALL   THERE, 1
Before Instruction
    PC      =   address (HERE)
After Instruction
    PC      =   address (THERE)
    TOS     =   address (HERE + 4)
    WS      =   W
    BSRS    =   BSR
    STATUSS =   Status
```

DAW **Decimal Adjust W Register**

Syntax: DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> + DC > 9] or [C = 1] then
(W<7:4>) + 6 + DC → W<7:4> ;
else
(W<7:4>) + DC → W<7:4>

Status Affected: C

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example1:

```
DAW
```

Before Instruction
W = A5h
C = 0
DC = 0
After Instruction
W = 05h
C = 1
DC = 0

Example 2:

Before Instruction
W = CEh
C = 0
DC = 0
After Instruction
W = 34h
C = 1
DC = 0

DECF **Decrement f**

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

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

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

0000	01da	ffff	ffff

Description: Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

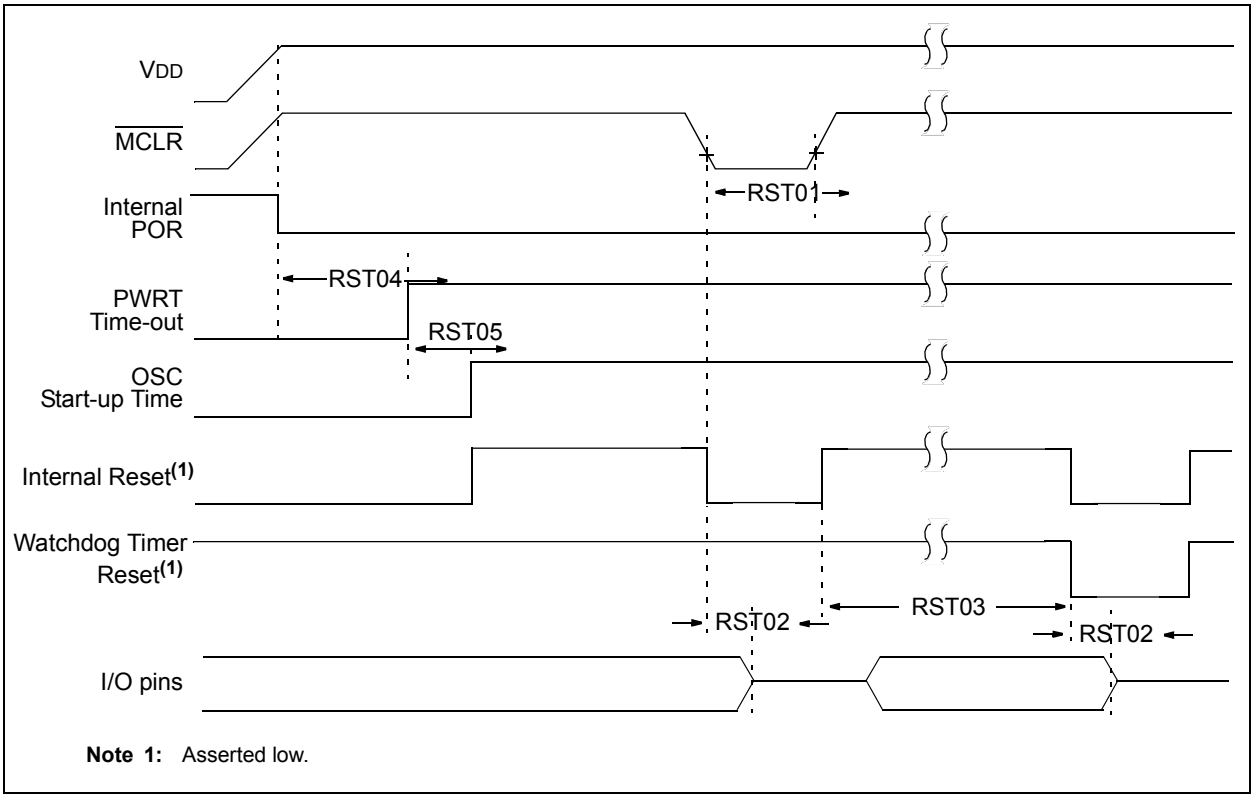
Q Cycle Activity:

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

Example: DECF CNT, 1, 0

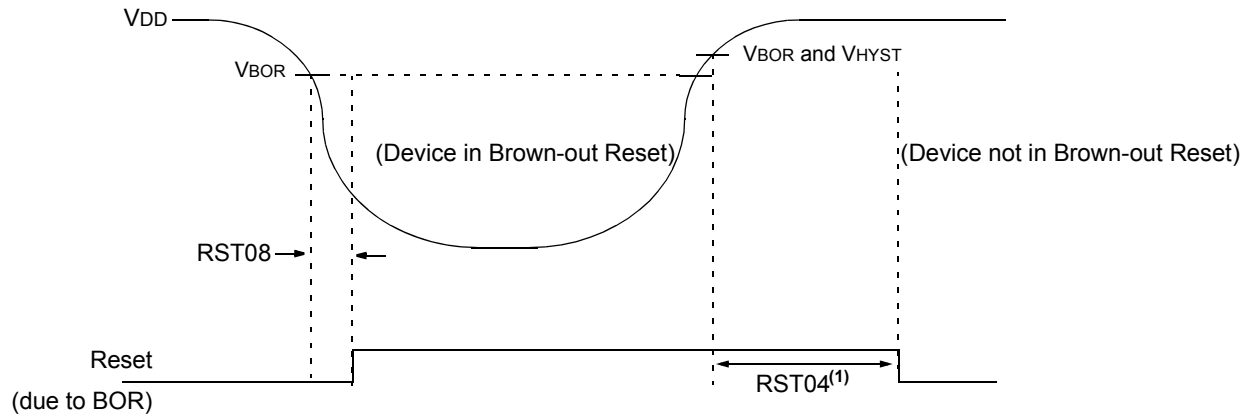
Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1

FIGURE 38-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note 1: Asserted low.

FIGURE 38-9: BROWN-OUT RESET TIMING AND CHARACTERISTICS



Note 1: Only if $\overline{\text{PWRTE}}$ bit in the Configuration Word register is programmed to '1'; 2 ms delay if $\overline{\text{PWRTE}}$ = 0.