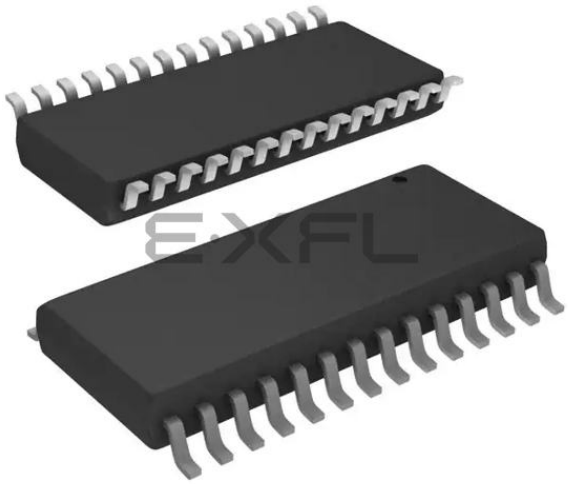


Welcome to E-XFL.COM

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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	21
Program Memory Size	24KB (12K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1408 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f2539-e-so

TABLE 1-2: PIC18F2X39 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	DIP	SOIC			
$\overline{MCLR}$/VPP	1	1			Master Clear (input) or high voltage ICSP programming enable pin.
$\overline{MCLR}$			I	ST	Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP			I	ST	High voltage ICSP programming enable pin.
NC	—	—	—	—	These pins should be left unconnected.
OSC1/CLKI	9	9			Oscillator crystal or external clock input.
OSC1			I	CMOS	Oscillator crystal input or external clock source input.
CLKI			I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
OSC2/CLKO/RA6	10	10			Oscillator crystal or clock output.
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O	—	In EC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
RA6			I/O	TTL	General purpose I/O pin.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2			
RA0			I/O	TTL	Digital I/O.
AN0			I	Analog	Analog input 0.
RA1/AN1	3	3			
RA1			I/O	TTL	Digital I/O.
AN1			I	Analog	Analog input 1.
RA2/AN2/VREF-	4	4			
RA2			I/O	TTL	Digital I/O.
AN2			I	Analog	Analog input 2.
VREF-			I	Analog	A/D Reference Voltage (Low) input.
RA3/AN3/VREF+	5	5			
RA3			I/O	TTL	Digital I/O.
AN3			I	Analog	Analog input 3.
VREF+			I	Analog	A/D Reference Voltage (High) input.
RA4/T0CKI	6	6			
RA4			I/O	ST/OD	Digital I/O. Open drain when configured as output.
T0CKI			I	ST	Timer0 external clock input.
RA5/AN4/$\overline{SS}$/LVDIN	7	7			
RA5			I/O	TTL	Digital I/O.
AN4			I	Analog	Analog input 4.
$\overline{SS}$			I	ST	SPI Slave Select input.
LVDIN			I	Analog	Low Voltage Detect input.
RA6					See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to VDD)

TABLE 1-3: PIC18F4X39 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	QFN	TQFP			
						PORTC is a bi-directional I/O port.
RC0/T13CKI	15	34	32			
RC0				I/O	ST	Digital I/O.
T13CKI				I	ST	Timer1/Timer3 external clock input.
RC3/SCK/SCL	18	37	37			
RC3				I/O	ST	Digital I/O.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	23	42	42			
RC4				I/O	ST	Digital I/O.
SDI				I	ST	SPI Data in.
SDA				I/O	ST	I^2C Data I/O.
RC5/SDO	24	43	43			
RC5				I/O	ST	Digital I/O.
SDO				O	—	SPI Data out.
RC6/TX/CK	25	44	44			
RC6				I/O	ST	Digital I/O.
TX				O	—	USART Asynchronous Transmit.
CK				I/O	ST	USART Synchronous Clock (see related RX/DT).
RC7/RX/DT	26	1	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	USART Asynchronous Receive.
DT				I/O	ST	USART Synchronous Data (see related TX/CK).
PWM1	17	35	36	O	—	PWM Channel 1 (motor control) output.
PWM2	16	36	35	O	—	PWM Channel 2 (motor control) output.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
OD = Open Drain (no P diode to V_{DD})
CMOS = CMOS compatible input or output
I = Input
P = Power

 © 2002-2013 Microchip Technology Inc.

TABLE 1-3: PIC18F4X39 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	QFN	TQFP			
						PORTD is a bi-directional I/O port, or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
RD0/PSP0	19	38	38	I/O		
RD0					ST	Digital I/O.
PSP0					TTL	Parallel Slave Port Data.
RD1/PSP1	20	39	39	I/O		
RD1					ST	Digital I/O.
PSP1					TTL	Parallel Slave Port Data.
RD2/PSP2	21	40	40	I/O		
RD2					ST	Digital I/O.
PSP2					TTL	Parallel Slave Port Data.
RD3/PSP3	22	41	41	I/O		
RD3					ST	Digital I/O.
PSP3					TTL	Parallel Slave Port Data.
RD4/PSP4	27	2	2	I/O		
RD4					ST	Digital I/O.
PSP4					TTL	Parallel Slave Port Data.
RD5/PSP5	28	3	3	I/O		
RD5					ST	Digital I/O.
PSP5					TTL	Parallel Slave Port Data.
RD6/PSP6	29	4	4	I/O		
RD6					ST	Digital I/O.
PSP6					TTL	Parallel Slave Port Data.
RD7/PSP7	30	5	5	I/O		
RD7					ST	Digital I/O.
PSP7					TTL	Parallel Slave Port Data.

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
OD = Open Drain (no P diode to V_{DD})

4.9 Data Memory Organization

The data memory 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 data memory map is divided into 16 banks that contain 256 bytes each. The lower 4 bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits for the BSR are not implemented.

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 scratch pad operations in the user's application. The SFRs start at the last location of Bank 15 (FFFh) and extend downwards. Any remaining space beyond the SFRs in the Bank may be implemented as GPRs. GPRs start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '0's.

The organization of the data memory space for these devices is shown in Figure 4-5 and Figure 4-6. PIC18FX439 devices have 640 bytes of data RAM, extending from Bank 0 to Bank 2 (000h through 27Fh). The block of 128 bytes above this to the top of the bank (280h to 2FFh) is used as data memory for the Motor Control kernel, and is not available to the user. Reading these locations will return random information that reflects the kernel's "scratch" data. Modifying the data in these locations may disrupt the operation of the ProMPT kernel.

PIC18FX539 devices have 1408 bytes of data RAM, extending from Bank 0 to Bank 5 (000h through 57Fh). As with the PIC18FX439 devices, the block of 128 bytes above this to the end of the bank (580h to 5FFh) is used by the Motor Control kernel.

The entire data memory may be accessed directly or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of a File Select Register (FSRn) and a corresponding Indirect File Operand (INDFn). Each FSR holds a 12-bit address value that can be used to access any location in the Data Memory map without banking.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing, or by the use of the `MOVFF` instruction. The `MOVFF` instruction is a two-word/two-cycle instruction that moves a value from one register to another.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. Section 4.10 provides a detailed description of the Access RAM.

4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly. Indirect addressing operates using a File Select Register and corresponding Indirect File Operand. The operation of indirect addressing is shown in Section 4.12.

Enhanced MCU devices may have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other RESETS.

Data RAM is available for use as GPR registers by all instructions. The top half of Bank 15 (F80h to FFFh) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

4.9.2 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. A list of these registers is given in Table 4-1 and Table 4-2.

The SFRs can be classified into two sets; those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature.

The SFRs are typically distributed among the peripherals whose functions they control. The unused SFR locations will be unimplemented and read as '0's. See Table 4-1 for addresses for the SFRs.

Note:	In this chapter and throughout this document, certain SFR names and individual bits are marked with an asterisk (*). This denotes registers that are not implemented in PIC18FXX39 devices, but whose names are retained to maintain compatibility with PIC18FXX2 devices. The designated bits within these registers are reserved and may be used by certain modules or the Motor Control kernel. Users should not write to these registers or alter these bit values. Failure to do this may result in erratic microcontroller operation.

FIGURE 4-8: INDIRECT ADDRESSING OPERATION

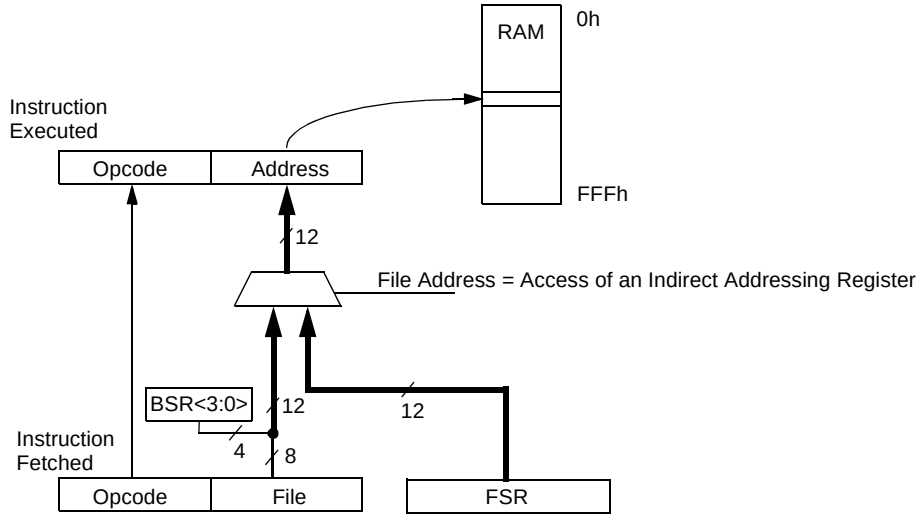
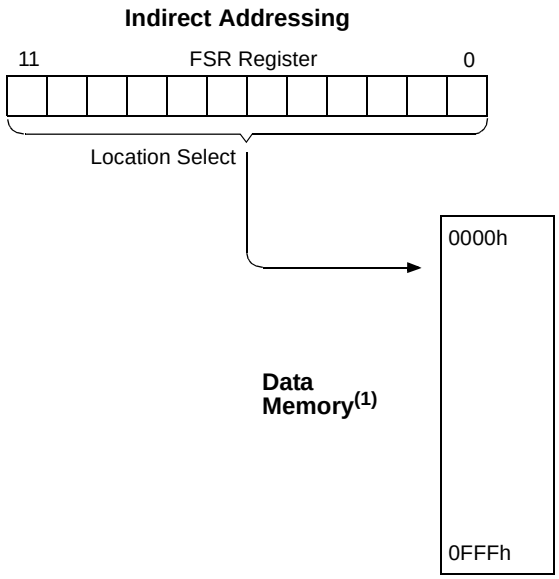


FIGURE 4-9: INDIRECT ADDRESSING



Note 1: For register file map detail, see Table 4-1.

 © 2002-2013 Microchip Technology Inc.

REGISTER 8-9: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	U-1
—	—	—	EEIP(1)	BCLIP(1)	LVDIP(1)	TMR3IP(1)	—
bit 7							bit 0

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

bit 4 **EEIP(1)**: Data EEPROM/FLASH Write Operation Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **BCLIP(1)**: Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **LVDIP(1)**: Low Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP(1)**: TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

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

Note 1: Maintain this bit cleared (= 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

NOTES:

 © 2002-2013 Microchip Technology Inc.

13.1 Timer3 Operation

Timer3 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

The Operating mode is determined by the clock select bit, TMR3CS (T3CON<1>). When TMR3CS = 0, Timer3 increments every instruction cycle. When TMR3CS = 1, Timer3 increments on every rising edge of the Timer1 external clock input.

FIGURE 13-1: TIMER3 BLOCK DIAGRAM

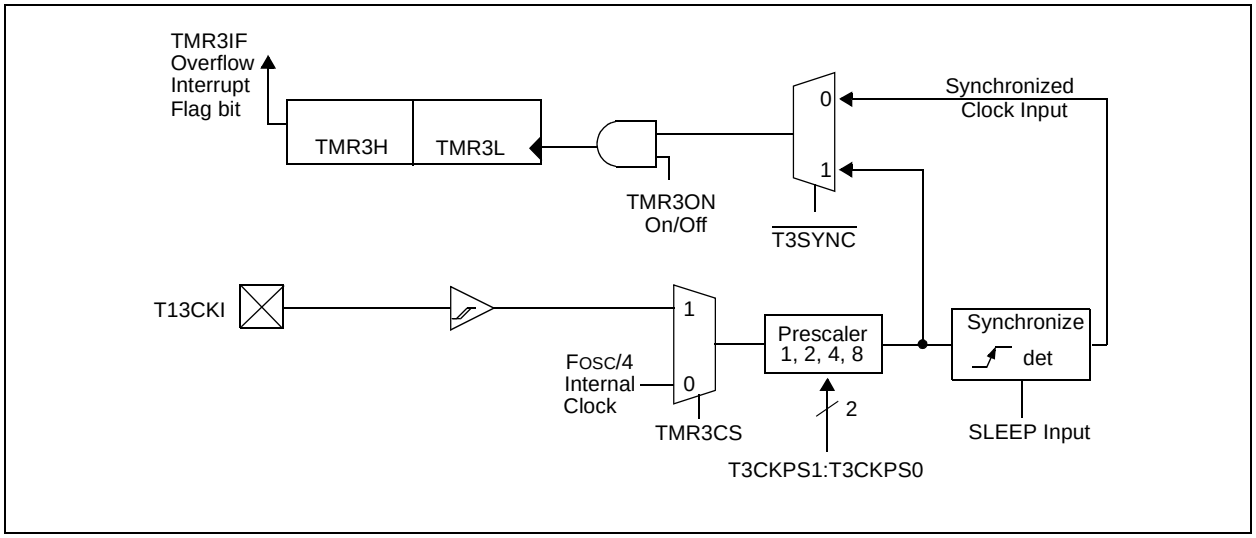
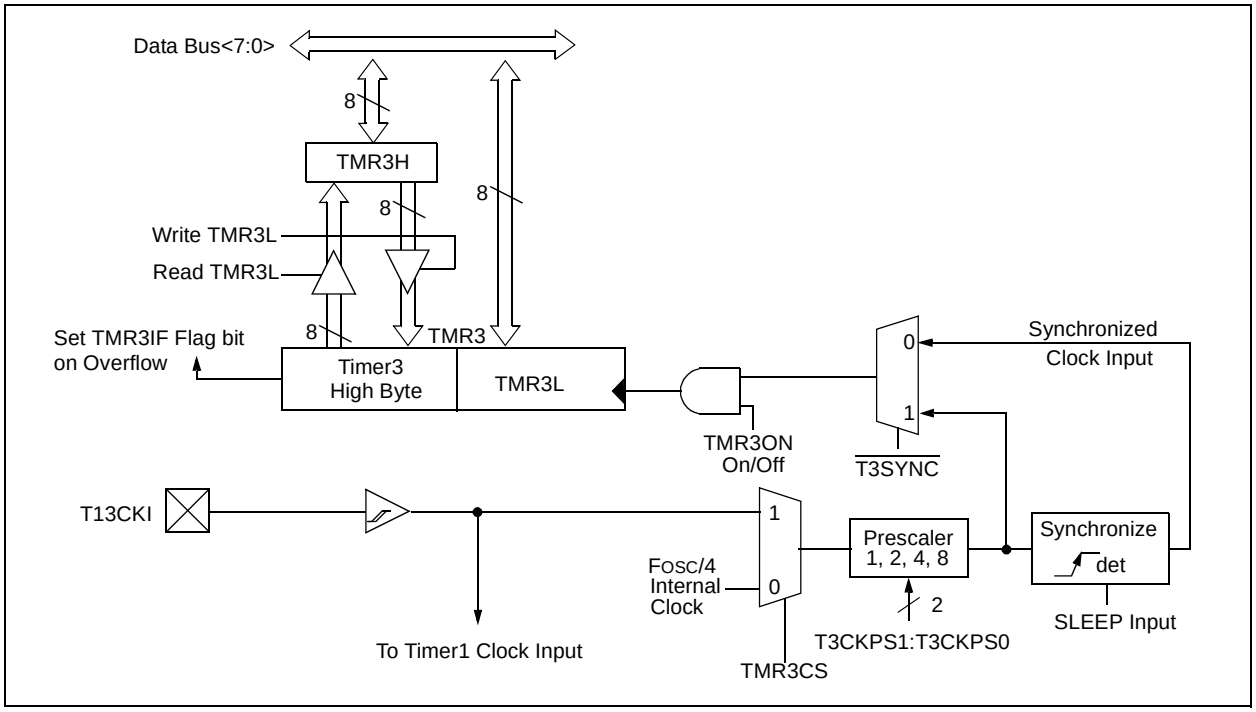


FIGURE 13-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE



© 2002-2013 Microchip Technology Inc.

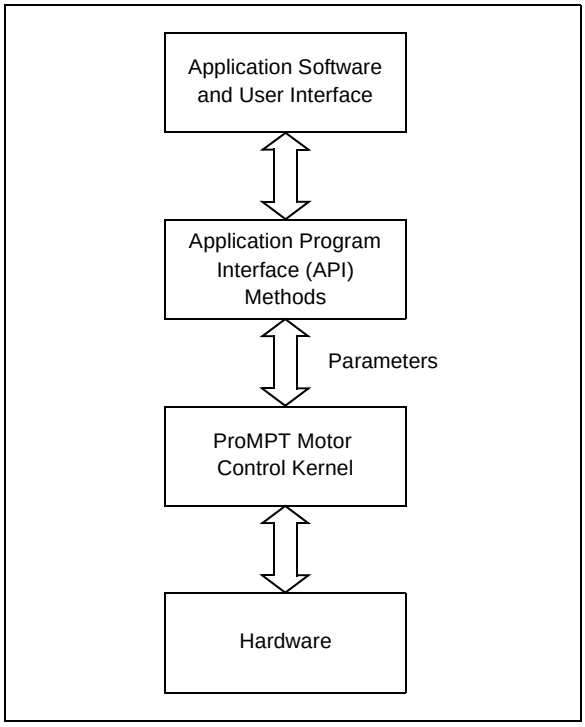
14.4 Developing Applications Using the Motor Control Kernel

The Motor Control kernel allows users to develop their applications without having knowledge of motor control. The key parameters of the motor control kernel can be set and read through the Application Program Interface (API) methods discussed in the previous section.

The overall application can be thought of as a protocol stack, as shown in Figure 14-3. In this case, the API methods reside between the user's application and the ProMPT kernel, and are used to exchange parameter values. The motor control kernel sets the PWM duty cycles based on the inputs from the application software.

A typical motor control routine is shown in Example 14-1. In this case, the motor will run at 20 Hz for 10 seconds, accelerate to 60 Hz at the rate of 10 Hz/s, remain at 60 Hz for 20 seconds, and finally stop.

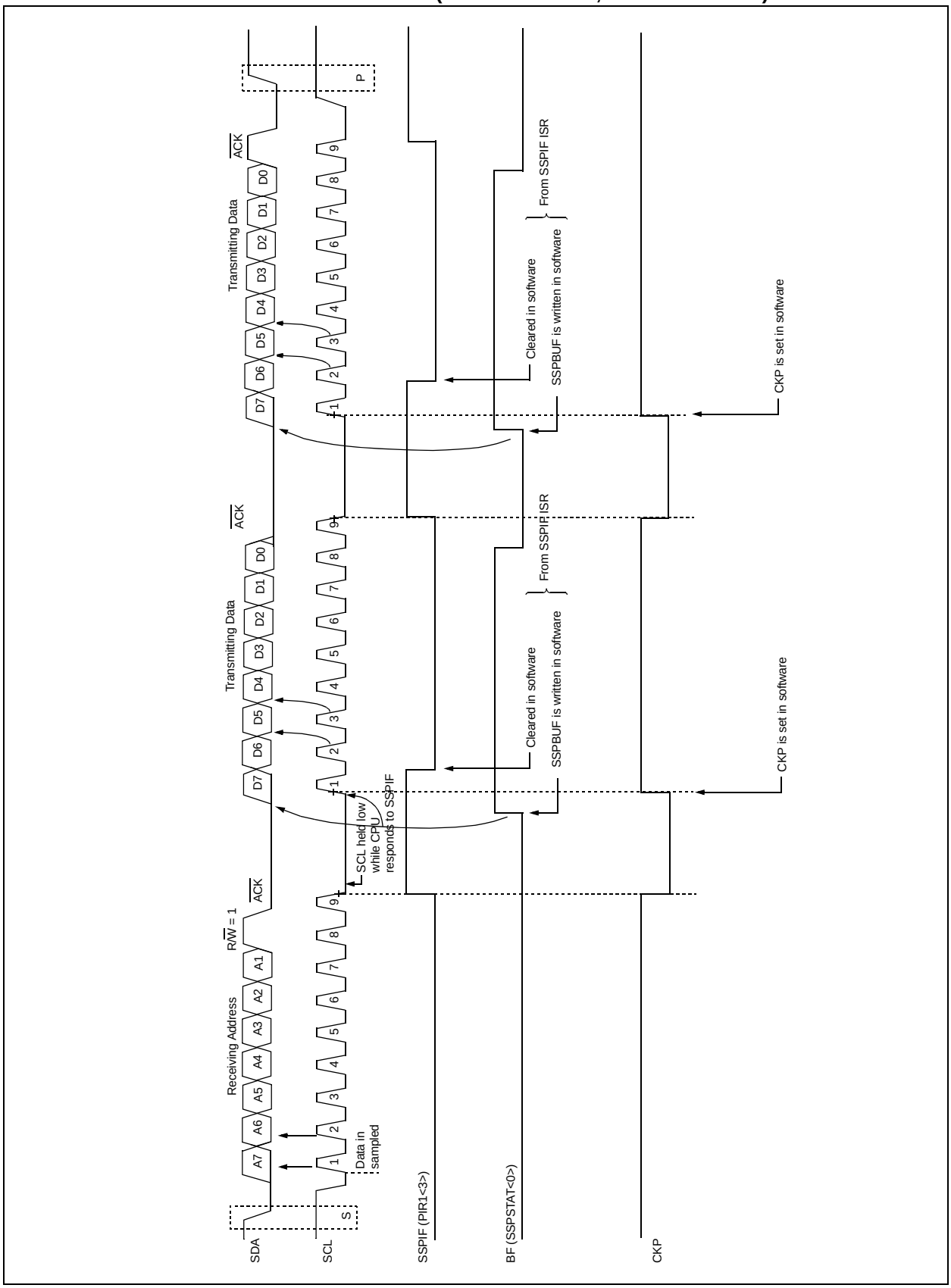
FIGURE 14-3: LAYERS OF THE MOTOR CONTROL ARCHITECTURE STACK



EXAMPLE 14-1: MOTOR CONTROL ROUTINE USING THE ProMPT APIs

```
Void main()
{
    unsigned char i;
    unsigned char j;
    ProMPT_Init(0);                     // Initialize the ProMPT block
    i = ProMPT_SetFrequency(10);        // Set motor frequency to 10Hz

    for (i=0;i<161;i++)                 // Set counter for 10 sec @ 1/16 sec per tick
        {
        j = ProMPT_Tick(void);          // Tick of 1/16 sec
        ProMPT_ClearTick(void);         // Clearing the Tick flag
        }

    ProMPT_SetAccelRate(10);            // Set acceleration rate to 10 Hz/sec

    i = ProMPT_SetFrequency(60);        // Set motor frequency to 60 Hz

    for (i=0;i<161;i++)                 // Set counter for 20 Sec @ 1/16 sec per tick
        {                               // (2 loops of 10 Sec each)
        j = ProMPT_Tick(void);          // Tick of 1/16 Sec
        ProMPT_ClearTick(void);         // Clearing the Tick flag
        j = ProMPT_Tick(void);          // Tick of 1/16 Sec
        ProMPT_ClearTick(void);         // Clearing the Tick flag
        }
    i = ProMPT_SetFrequency(0);         // Set motor frequency to 0 Hz (stop)
    while(1);
}                                       // End of the task
```

FIGURE 16-9: I²C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)



16.4.14 SLEEP OPERATION

While in SLEEP mode, the I^2C module can receive addresses or data, and when an address match or complete byte transfer occurs, wake the processor from SLEEP (if the MSSP interrupt is enabled).

16.4.15 EFFECT OF A RESET

A RESET disables the MSSP module and terminates the current transfer.

16.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the START and STOP conditions allows the determination of when the bus is free. The STOP (P) and START (S) bits are cleared from a RESET, or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit (SSPSTAT<4>) is set, or the bus is IDLE, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the STOP condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration, to see if the signal level is the expected output level. This check is performed in hardware, with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A START Condition
- A Repeated START Condition
- An Acknowledge Condition

16.4.17 MULTI -MASTER COMMUNICATION, BUS COLLISION, AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag BCLIF and reset the I^2C port to its IDLE state (Figure 16-25).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are de-asserted, and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine, and if the I^2C bus is free, the user can resume communication by asserting a START condition.

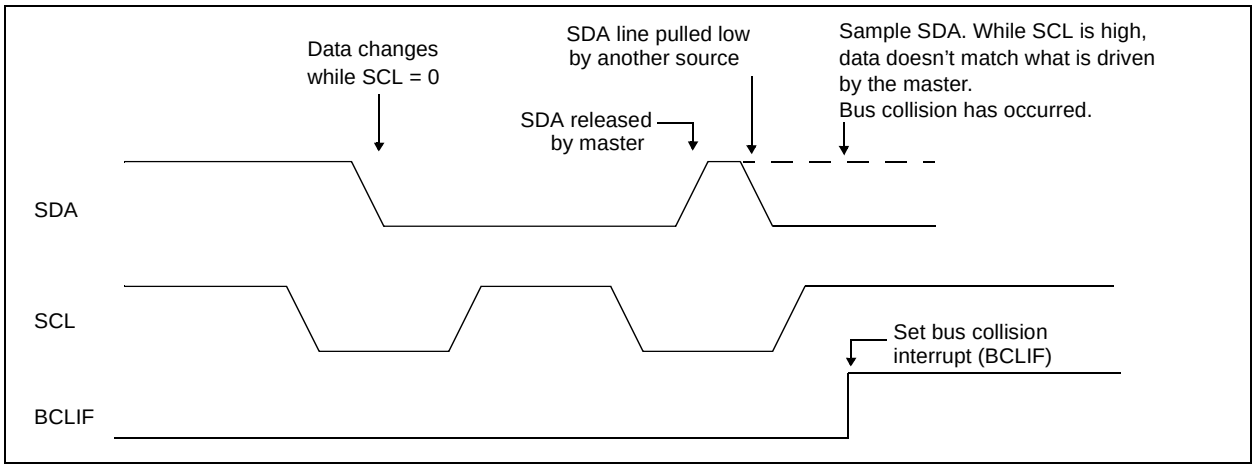
If a START, Repeated START, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted, and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the I^2C bus is free, the user can resume communication by asserting a START condition.

The master will continue to monitor the SDA and SCL pins. If a STOP condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of START and STOP conditions allows the determination of when the bus is free. Control of the I^2C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is IDLE and the S and P bits are cleared.

FIGURE 16-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



19.0 LOW VOLTAGE DETECT

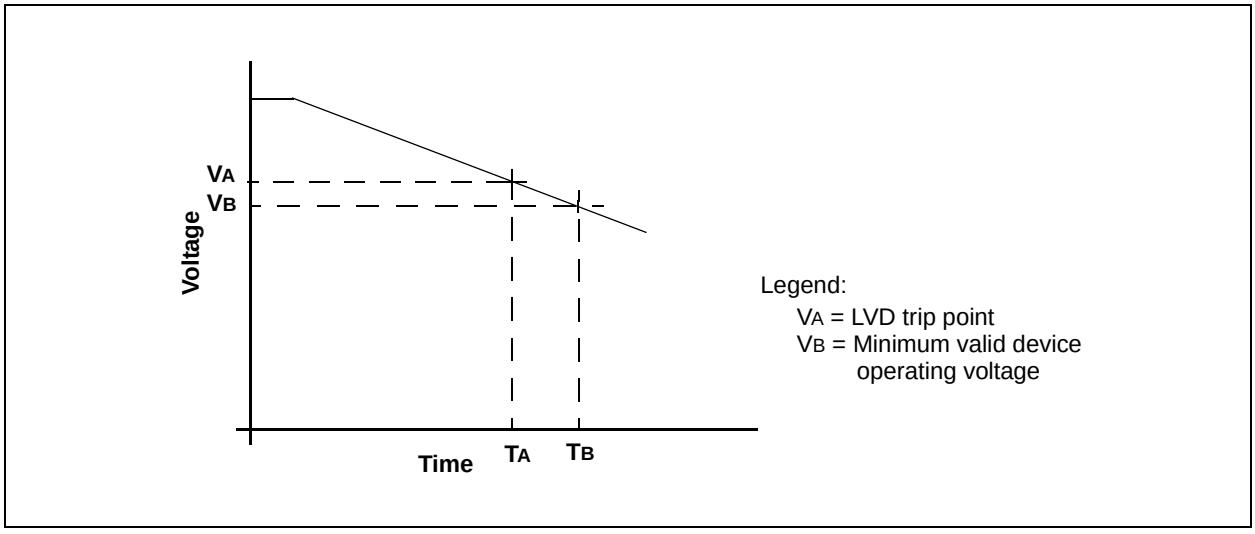
In many applications, the ability to determine if the device voltage (V_{DD}) is below a specified voltage level is a desirable feature. A window of operation for the application can be created, where the application software can do “housekeeping tasks” before the device voltage exits the valid operating range. This can be done using the Low Voltage Detect module.

This module is a software programmable circuitry, where a device voltage trip point can be specified. When the voltage of the device becomes lower then the specified point, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to that interrupt source.

The Low Voltage Detect circuitry is completely under software control. This allows the circuitry to be “turned off” by the software, which minimizes the current consumption for the device.

Figure 19-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage V_A, the LVD logic generates an interrupt. This occurs at time T_A. The application software then has the time, until the device voltage is no longer in valid operating range, to shutdown the system. Voltage point V_B is the minimum valid operating voltage specification. This occurs at time T_B. The difference $T_B - T_A$ is the total time for shutdown.

FIGURE 19-1: TYPICAL LOW VOLTAGE DETECT APPLICATION



The block diagram for the LVD module is shown in Figure 19-2. A comparator uses an internally generated reference voltage as the set point. When the selected tap output of the device voltage crosses the set point (is lower than), the LVDIF bit is set.

Each node in the resistor divider represents a “trip point” voltage. The “trip point” voltage is the minimum supply voltage level at which the device can operate before the LVD module asserts an interrupt. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the 1.2V internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal setting the LVDIF bit. This voltage is software programmable to any one of 16 values (see Figure 19-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

21.1 Instruction Set

ADDLW **ADD literal to W**

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: $(W) + k \rightarrow W$

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

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDLW 0x15`

Before Instruction
W = 0x10

After Instruction
W = 0x25

ADDWF **ADD W to f**

Syntax: [*label*] ADDWF f [,d [,a]

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

Operation: $(W) + (f) \rightarrow$ dest

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

Encoding:

0010	01da	ffff	ffff

Description: Add W to 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 will be selected. If 'a' is 1, the BSR is used.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDWF REG, 0, 0`

Before Instruction
W = 0x17
REG = 0xC2

After Instruction
W = 0xD9
REG = 0xC2

BCF **Bit Clear f**

Syntax: [*label*] BCF f,b[,a]

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

BN **Branch if Negative**

Syntax: [*label*] BN n

Operands: $-128 \le n \le 127$

Operation: if negative bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the Negative 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 two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

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

Example:
```
HERE    BN   Jump
```
Before Instruction
PC = address (HERE)
After Instruction
If Negative = 1;
PC = address (Jump)
If Negative = 0;
PC = address (HERE+2)

 © 2002-2013 Microchip Technology Inc.

DAW **Decimal Adjust W Register**

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

Status Affected: C

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the eight-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 = 0xA5
C = 0
DC = 0

After Instruction
W = 0x05
C = 1
DC = 0

Example 2:

Before Instruction
W = 0xCE
C = 0
DC = 0

After Instruction
W = 0x34
C = 1
DC = 0

DECF **Decrement f**

Syntax: [*label*] 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

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 = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

 © 2002-2013 Microchip Technology Inc.

SUBLW **Subtract W from literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

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

Encoding:

0000	1000	kkkk	kkkk

Description: W is subtracted from the eight-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBLW 0x02`

Before Instruction
W = 1
C = ?

After Instruction
W = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBLW 0x02`

Before Instruction
W = 2
C = ?

After Instruction
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBLW 0x02`

Before Instruction
W = 3
C = ?

After Instruction
W = FF ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF **Subtract W from f**

Syntax: [*label*] SUBWF f [,d [,a]

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

Operation: $(f) - (W) \rightarrow dest$

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

Encoding:

0101	11da	ffff	ffff

Description: Subtract W from register 'f' (2's complement method). 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 will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBWF REG, 1, 0`

Before Instruction
REG = 3
W = 2
C = ?

After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBWF REG, 0, 0`

Before Instruction
REG = 2
W = 2
C = ?

After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBWF REG, 1, 0`

Before Instruction
REG = 1
W = 2
C = ?

After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

TABLE 23-22: A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D clock period	PIC18**F**XXXX	1.6	20(4)	µs	TOSC based
			PIC18**LF**XXXX	2.0	6.0	µs	A/D RC mode
131	TCNV	Conversion time (not including acquisition time) **(Note 1)**		11	12	TAD	
132	TACQ	Acquisition time **(Note 2)**		5 10	— —	µs µs	VREF = VDD = 5.0V VREF = VDD = 2.5V
135	TSWC	Switching Time from convert → sample		—	**(Note 3)**		

Note 1: ADRES register may be read on the following TCY cycle.

2: The time for the holding capacitor to acquire the "New" input voltage, when the new input value has not changed by more than 1 LSB from the last sampled voltage. The source impedance (*RS*) on the input channels is 50Ω. See Section 18.0 for more information on acquisition time consideration.

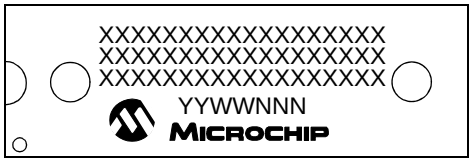
3: On the next Q4 cycle of the device clock.

4: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

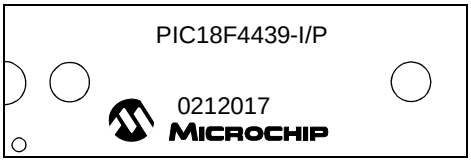
 © 2002-2013 Microchip Technology Inc.

Package Marking Information (Cont'd)

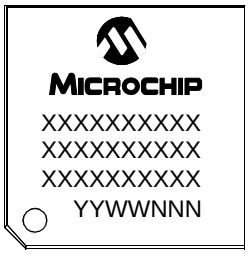
40-Lead PDIP



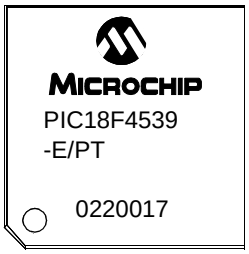
Example



44-Lead TQFP



Example



44-Lead QFN

Example

 © 2002-2013 Microchip Technology Inc.

APPENDIX C: CONVERSION CONSIDERATIONS

The considerations for converting applications from previous versions of PIC18 microcontrollers (i.e., PIC18FXX2 devices) are listed in Table C-1.

A specific list of resources that are unavailable to PIC18FXX2 applications in PIC18FXX39 devices is presented in Table C-2.

TABLE C-1: CONVERSION CONSIDERATIONS BETWEEN PIC18FXX2 AND PIC18FXX39 DEVICES

Characteristic	PIC18FXX2	PIC18FXX39
Pins	28/40/44	28/40/44
Available Packages	DIP, PDIP, SOIC, PLCC, QFN, TQFP	DIP, PDIP, SOIC, QFN, TQFP
Voltage Range	2.0 - 5.5V	2.0 - 5.5V
Frequency Range	DC - 40 MHz	4 - 40 MHz (20 MHz optimal)
Available Program Memory (bytes)	16K or 32K	12K or 24K
Available Data RAM (bytes)	768 or 1536	640 or 1408
Data EEPROM	256	256
Interrupt Sources	17 or 18	15 or 16
Interrupt Priority Levels	Two levels: low priority (vector at 0008h) high priority (vector at 0018h)	One level when using Motor Control: vector at 0008h
Timers (available to users)	4	3
Timer1 Oscillator option	yes	no
Oscillator Switching	yes	no
Capture/Compare/PWM	2 CCP	2 PWM only, available only through Motor Control kernel
Motor Control Kernel	no	yes
A/D	10-bit, 5 or 8 channels, 7 conversion clock selects	10-bit, 5 or 8 channels, 7 conversion clock selects
Communications	PSP, AUSART, MSSP (SPI and I^2C)	PSP, AUSART, MSSP (SPI and I^2C)
Code Protection	By 8K block with separate 512-byte boot block; protection from external reads and writes, Table Read and intra-block Table Read	By 8K block with separate 512-byte boot block; protection from external reads and writes, Table Read and intra-block Table Read; Block 3 not protected on PIC18FX539

TABLE C-2: UNAVAILABLE RESOURCES (COMPARED TO PIC18FXX2)

Resource Type	Item(s)
I/O Resources	RC1; RC2; T1OSO; T1OSI
Registers	CCP1CON; CCP2CON; CCPR1L; CCPR2L; TMR2; PR2; T2CON; OSCCON
SFR bits	CCP1IE; CCP1IF; CCP1IP; CCP21E; CCP21F; CCP2IP; T1OSCEN; T3CCP1; TMR2ON; TOUTPS<3:0>; T2CKPS<1:0>; T3CCP2; SFS; RC1; RC2; TRISC1; TRISC2; LATC1; LATC2
Interrupts and Interrupt Resources	CCP1 Capture/Compare match; CCP2 Capture/Compare match; High priority interrupts (when Motor Control is used; reserved for Timer2)
Timer Resources	Timer2 (available only through the Motor Control kernel); Timer2 as a clock source for MSSP module (SPI mode)
CCP Resources	Capture and Compare functionality; Timer1 reset on special event; Timer3 reset on special event; A/D conversion on special event; Interrupt on special event
Configuration Word bits	OSCEN; CCP2MX; CP3; WRT3; EBTR3

 © 2002-2013 Microchip Technology Inc.

C

CALL 226
Clocking Scheme/Instruction Cycle 36
CLRF 227
CLRWDT 227
Code Examples
16 x 16 Signed Multiply Routine 68
16 x 16 Unsigned Multiply Routine 68
8 x 8 Signed Multiply Routine 67
8 x 8 Unsigned Multiply Routine 67
Data EEPROM Read 63
Data EEPROM Refresh Routine 64
Data EEPROM Write 63
Erasing a FLASH Program Memory Row 56
How to Clear RAM (Bank 1) Using Indirect Addressing 47
Initializing PORTA 83
Initializing PORTB 86
Initializing PORTC 89
Initializing PORTD 91
Initializing PORTE 93
Loading the SSPBUF (SSPSR) Register 128
Motor Control Routine using ProMPT APIs 121
Reading a FLASH Program Memory Word 55
Saving STATUS, WREG and BSR Registers in RAM 81
Writing to FLASH Program Memory 58–59
Code Protection 195
COMF 228
Configuration Bits 195
Context Saving During Interrupts 81
Conversion Considerations 306
CPFSEQ 228
CPFSGT 229
CPFSLT 229

D

Data EEPROM Memory
Associated Registers 65
EEADR Register 61
EECON1 Register 61
EECON2 Register 61
Operation During Code Protect 64
Protection Against Spurious Write 64
Reading 63
Using 64
Write Verify 64
Writing 63
Data Memory 39
General Purpose Registers 39
Map for PIC18FX439 40
Map for PIC18FX539 41
Special Function Registers 39
DAW 230
DC and AC Characteristics
Graphs and Tables 287
DC Characteristics 261, 264
DCFSNZ 231
DECF 230
DECFSZ 231
Developing Applications 121
Development Support 253
Device Differences 305
Device Overview 7
Features 8
Direct Addressing 48
Example 46

E

Electrical Characteristics 259
Errata 5

F

Firmware Instructions 211
FLASH Program Memory 51
Associated Registers 59
Control Registers 52
Erase Sequence 56
Erasing 56
Operation During Code Protection 59
Reading 55
TABLAT Register 54
Table Pointer 54
Boundaries Based on Operation 54
Table Pointer Boundaries 54
Table Reads and Table Writes 51
Writing to 57
Protection Against Spurious Writes 59
Unexpected Termination 59
Write Verify 59

G

GOTO 232

H

Hardware Interface 113
Hardware Multiplier 67
Introduction 67
Operation 67
Performance Comparison 67
HS/PLL 20

I

I/O Ports 83
I^2C Mode
Bus Collision
During a STOP Condition 163
I^2C Mode 134
$\overline{ACK}$ Pulse 138, 139
Acknowledge Sequence Timing 158
Baud Rate Generator 151
Bus Collision
Repeated START Condition 162
START Condition 160
Clock Arbitration 152
Clock Stretching 144
Effect of a RESET 159
General Call Address Support 148
Master Mode 149
Operation 150
Reception 155
Repeated START Condition Timing 154
START Condition Timing 153
Transmission 155
Multi-Master Communication, Bus Collision and Arbitration 159

 © 2002-2013 Microchip Technology Inc.