CHAPTER 2
PIC® Microcontroller Low Power Tips ‘n Tricks

Table Of Contents

TIPS ‘N TRICKS INTRODUCTION

Microchip continues to provide innovative products that are smaller, faster, easier to use and more reliable. The Flash-based PIC® microcontrollers (MCUs) are used in an wide range of everyday products, from smoke detectors, hospital ID tags and pet containment systems, to industrial, automotive and medical products.

PIC MCUs featuring nanoWatt technology implement a variety of important features which have become standard in PIC microcontrollers. Since the release of nanoWatt technology, changes in MCU process technology and improvements in performance have resulted in new requirements for lower power. PIC MCUs with nanoWatt eXtreme Low Power (nanoWatt XLP™) improve upon the original nanoWatt technology by dramatically reducing static power consumption and providing new flexibility for dynamic power management.

The following series of Tips n’ Tricks can be applied to many applications to make the most of PIC MCU nanoWatt and nanoWatt XLP devices.

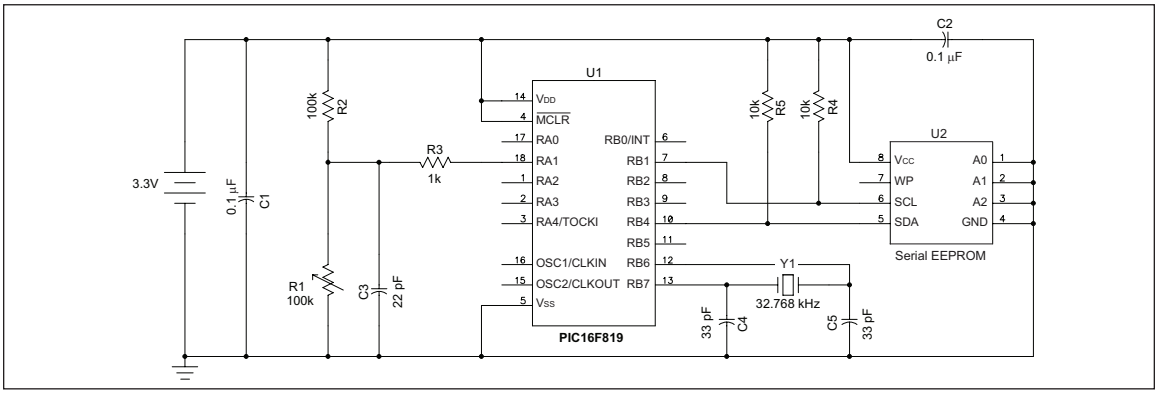
GENERAL LOW POWER TIPS ‘N TRICKS

The following tips can be used with all PIC MCUs to reduce the power consumption of almost any application.

TIP #1 Switching Off External Circuits/Duty Cycle

All the low power modes in the world won't help your application if you are unable to control the power used by circuits external to the microprocessor. Lighting an LED is equivalent to running most PIC MCUs at 5V-20 MHz. When you are designing your circuitry, decide what physical modes or states are required and partition the electronics to shutdown unneeded circuitry.

Figure: 1-1



The system shown above is very simple and clearly has all the parts identified in the requirements. Unfortunately, it has a few problems in that the EEPROM, the sensor, and its bias circuit, are energized all the time. To get the minimum current draw for this design, it would be advantageous to shutdown these circuits when they are not required.

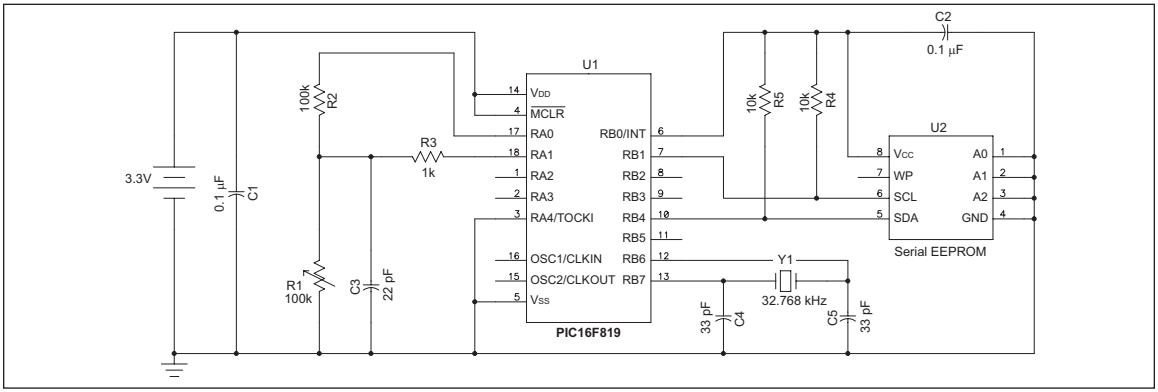
Figure: 1-2

Example:

The application is a long duration data recorder. It has a sensor, an EEPROM, a battery and a microprocessor. Every two seconds, it must take a sensor reading, scale the sensor data, store the scaled data in EEPROM and wait for the next sensor reading.

In Figure 1-2, I/O pins are used to power the EEPROM and the sensor. Many PIC MCU devices can source up to 20 mA of current from each I/O, so there is no need to provide additional components to switch the power.

If more current than can be sourced by the PIC MCU is required, the PIC MCU can instead enable and disable a MOSFET to power the circuit. Refer to the data sheet for drive capabilities for a specific device.



TIP #2 Power Budgeting

Power budgeting is a technique that is critical to predicting current consumption and battery life. Power budgeting is performed by calculating the total charge for each mode of operation of an application by multiplying that mode’s current consumption by the time in the mode for a single application loop. The charge for each mode is added, then averaged over the total loop time to get average current. Table 1 calculates a power budget using the application from Figure 2 in Tip #1 using a typical nanoWatt XLP device.

Mode	Time in Mode (mS)	Current (mA)		Charge Current * Time (mA * Sec)
		By Device	Mode Total	
Sleep				
MCU Sleep	1989	0.00005	5.00E-05	9.95E-05
Sensor Off		0		
EEPROM Off		0		
Initialize				
MCU Sleep	1	0.00005	1.66E-02	1.66E-05
Sensor On		0.0165		
EEPROM Off		0		
Sample Sensor				
MCU Run	1	0.048	6.45E-02	6.45E-05
Sensor On		0.0165		
EEPROM Off		0		
Scaling				
MCU Run	1	0.048	4.80E-02	4.80E-05
Sensor Off		0		
EEPROM Off		0		
Storing				
MCU Run	8	0.048	1.05E+00	8.38E-03
Sensor Off		0		
EEPROM On		1		
Total	2000	—	—	8.61E-03

Average Current

$$= \frac{8.61e\text{-}3}{2000e\text{-}3} \ \frac{mA*Sec}{Sec}$$

$$= 0.0043 \text{ mA}$$

Peak Current 1.05 mA

Computing Battery Life

Using the average current from the calculated power budget, it is possible to determine how long a battery will be able to power the application. Table 2 shows lifetimes for typical battery types using the average power from Table 1.

Battery	Capacity (mAh)	Life			
		Hours	Days	Months	Years
CR1212	18	4180	174	5.8	.48
CR1620	75	17417	726	24.2	1.99
CR2032	220	51089	2129	71.0	5.83
Alkaline AAA	1250	290276	12095	403.2	33.14
Alkaline AA	2890	671118	27963	932.1	76.61
Li-ion*	850	197388	8224	274.1	22.53

NOTE: Calculations are based on average current draw only and do not include battery self-discharge.

*Varies by size; value used is typical.

After completing a power budget, it is very easy to determine the battery size required to meet the application requirements. If too much power is consumed, it is simple to determine where additional effort needs to be placed to reduce the power consumption.

TIP #3 Configuring Port Pins

All PIC MCUs have bidirectional I/O pins. Some of these pins have analog input capabilities. It is very important to pay attention to the signals applied to these pins so the least amount of power will be consumed.

Unused Port Pins

If a port pin is unused, it may be left unconnected but configured as an output pin driving to either state (high or low), or it may be configured as an input with an external resistor (about 10 kΩ) pulling it to V_{DD} or V_{SS}. If configured as an input, only the pin input leakage current will be drawn through the pin (the same current would flow if the pin was connected directly to V_{DD} or V_{SS}). Both options allow the pin to be used later for either input or output without significant hardware modifications.

Digital Inputs

A digital input pin consumes the least amount of power when the input voltage is near V_{DD} or V_{SS}. If the input voltage is near the midpoint between V_{DD} and V_{SS}, the transistors inside the digital input buffer are biased in a linear region and they will consume a significant amount of current. If such a pin can be configured as an analog input, the digital buffer is turned off, reducing both the pin current as well as the total controller current.

Analog Inputs

Analog inputs have a very high-impedance so they consume very little current. They will consume less current than a digital input if the applied voltage would normally be centered between V_{DD} and V_{SS}. Sometimes it is appropriate and possible to configure digital inputs as analog inputs when the digital input must go to a low power state.

Digital Outputs

There is no additional current consumed by a digital output pin other than the current going through the pin to power the external circuit. Pay close attention to the external circuits to minimize their current consumption.

TIP #4 Use High-Value Pull-Up Resistors

It is more power efficient to use larger pull-up resistors on I/O pins such as MCLR, I^2C™ signals, switches and for resistor dividers. For example, a typical I^2C pull-up is 4.7k. However, when the I^2C is transmitting and pulling a line low, this consumes nearly 700 uA of current for each bus at 3.3V. By increasing the size of the I^2C pull-ups to 10k, this current can be halved. The tradeoff is a lower maximum I^2C bus speed, but this can be a worthwhile trade in for many low power applications. This technique is especially useful in cases where the pull-up can be increased to a very high resistance such as 100k or 1M.

TIP #5 Reduce Operating Voltage

Reducing the operating voltage of the device, V_{DD}, is a useful step to reduce the overall power consumption. When running, power consumption is mainly influenced by the clock speed. When sleeping, the most significant factor is leakage in the transistors. At lower voltages, less charge is required to switch the system clocks and transistors leak less current.

It is important to pay attention to how reducing the operating voltage reduces the maximum allowed operating frequency. Select the optimum voltage that allows the application to run at its maximum speed. Refer to the device data sheet for the maximum operating frequency of the device at the given voltage.

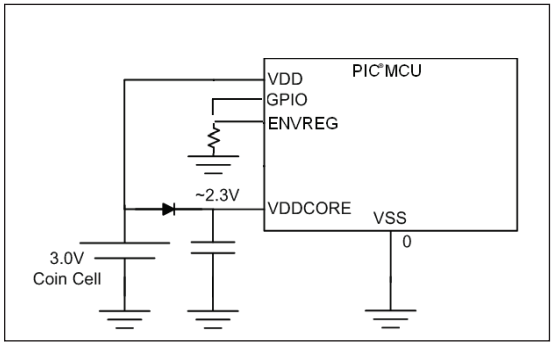
TIP #6 Use an External Source for CPU Core Voltage

Some PIC MCUs such as “J” type devices (ex. PIC18F87J90 or PIC24FJ64GA004) use separate power for CPU core. These devices have an internal voltage regulator that can be used to provide the core voltage. Alternatively, the core voltage can be provided externally by disabling the internal regulator. In some cases, it is more power efficient to use an external source for the core. This is because the internal regulator powers the core at the nominal voltage that allows full speed operation. However, if an application doesn’t require full speed, it is beneficial to use lower voltage to power the core. Disabling the internal regulator also turns off the BOR and LVD circuits, which saves power as well. The following examples show two different battery powered applications where it can be beneficial to disable the internal regulator.

Example 1: Constant Voltage Source

When using a regulated power source or a battery with a flat discharge curve, such as a lithium coin cell, the regulator can be disabled and the core powered directly from the battery through a diode. The diode provides the voltage drop necessary to power the core at the correct voltage. It may be necessary to use a zener diode with a higher forward voltage for applications using sleep mode, as the current consumed in sleep is too low to cause the full forward voltage drop which can result in applying a voltage too high for the core.

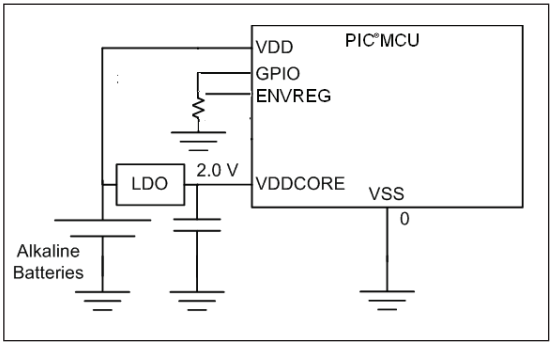
Figure 6-1:



Example 2: Non-Constant Voltage Source

If the source for V_{DD} is not constant, a regulator will be required. It can be beneficial to use an external low quiescent current regulator, which can be selected to provide lower voltage to the core than the internal regulator. Additionally, devices such as the MCP1700, which consumes 1 uA quiescent current while asleep, require less power than the internal regulator.

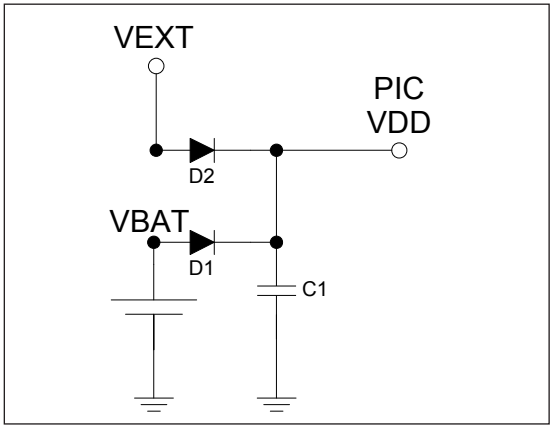
Figure 6-2:



TIP #7 Battery Backup for PIC MCUs

For an application that can operate from either an external supply or a battery backup, it is necessary to be able to switch from one to the other without user intervention. This can be accomplished with battery backup ICs, but it is also possible to implement with a simple diode OR circuit, shown in Figure 7-1. Diode D1 prevents current from flowing into the battery from VEXT when the external power is supplied. D2 prevents current from flowing into any external components from the battery if VEXT is removed. As long as the external source is present and higher voltage than the battery, no current from the battery will be used. When VEXT is removed and the voltage drops below VBAT, the battery will start powering the MCU. Low forward voltage Schottky diodes can be used in order to minimize the voltage dropout from the diodes. Additionally, inputs can be referenced to VEXT and VBAT in order to monitor the voltage levels of the battery and the external supply. This allows the micro to enter lower power modes when the supply is removed or the battery is running low. In order to avoid glitches on V_{DD} caused by the diode turn-on delay when switching supplies, ensure enough decoupling capacitance is used on V_{DD} (C1).

Figure 7-1:



Dynamic Operation Tips n’ Tricks

The following tips and tricks apply to methods of improving the dynamic operating current consumption of an application. This allows an application to get processing done quicker which enables it to sleep more and will help reduce the current consumed while processing.

TIP #8 Enhanced PIC16 Mid-Range Core

The Enhanced PIC16 mid-range core has a few features to assist in low power. New instructions allow many applications to execute in less time. This allows the application to spend more time asleep and less time processing and can provide considerable power savings. It is important not to overlook these new instructions when designing with devices that contain the new core. The Timer1 oscillator and WDT have also been improved, now meeting nanoWatt XLP requirements and drawing much less current than in previous devices.

TIP #9 Two-Speed Start-Up

Two-speed startup is a useful feature on some nanoWatt and all nanoWatt XLP devices which helps reduce power consumption by allowing the device to wake up and return to sleep faster. Using the internal oscillator, the user can execute code while waiting for the Oscillator Start-up (OST) timer to expire (LP, XT or HS modes). This feature (called “Two-Speed Start-up”) is enabled using the IESO configuration bit. A Two-Speed Start-up will clock the device from an internal RC oscillator until the OST has expired. Switching to a faster internal oscillator frequency during start-up is also possible using the OSCCON register. The example below shows several stages on how this can be achieved. The number of frequency changes is dependent upon the designer’s discretion. Assume a 20 MHz crystal (HS Mode) in the PIC16F example below.

Example:

```
TCY
(Instruction Time)      Instruction
                        ORG   0x05                ;Reset vector

125 µs @ 32 kHz         BSF   STATUS,RP0         ;bank1
125 µs @ 32 kHz         BSF   OSCCON,IRCF2      ;switch to 1 MHz

4 µs @ 1 MHz            BSF   OSCCON,IRCF1      ;switch to 4 MHz

1 µs @ 4 MHz            BSF   OSCCON,IRCF0      ;switch to 8 MHz

500 ns                  application code
500 ns                  application code
…                       ….
..                      …
(eventually OST expires, 20 MHz crystal clocks the device)

200 ns                  application code
…                       ….
..                      …
```

TIP #10 Clock Switching

Some nanoWatt devices and all nanoWatt XLP devices have multiple internal and external clock sources, as well as logic to allow switching between the available clock sources as the main system clock. This allows for significant power savings by choosing different clocks for different portions of code. For example, an application can use the slower internal oscillator when executing non-critical code and then switch to a fast high-accuracy oscillator for time or frequency sensitive code. Clock switching allows much more flexible applications than being stuck with a single clock source. Clock switching sequences vary by device family, so refer to device data sheets or Family Reference Manuals for the specific clock switching sequences.

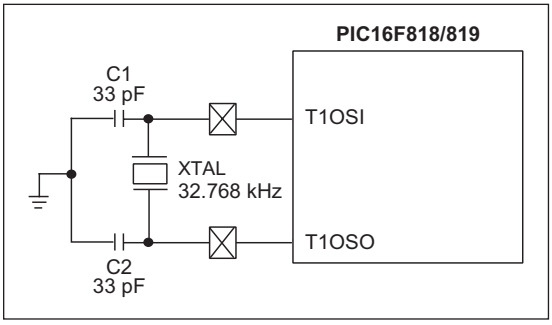
TIP #11 Use Internal RC Oscillators

If frequency precision better than ±5% is not required, it is best to utilize the internal RC oscillators inside all nanoWatt and nanoWatt XLP devices. The internal RC oscillators have better frequency stability than external RC oscillators, and consume less power than external crystal oscillators. Additionally, the internal clock can be configured for many frequency ranges using the internal PLL module to increase frequency and the postscaler to reduce it. All these options can be configured in firmware.

TIP #12 Internal Oscillator Calibration

An internal RC oscillator calibrated from the factory may require further calibration as the temperature or V_{DD} change. Timer1/SOSC can be used to calibrate the internal oscillator by connecting a 32.768 kHz clock crystal. Refer to AN244, “*Internal RC Oscillator Calibration*” for the complete application details. Calibrating the internal oscillator can help save power by allowing for use of the internal RC oscillator in applications which normally require higher accuracy crystals

Figure 12-1: Timer1 Used to Calibrate an Internal Oscillator



The calibration is based on the measured frequency of the internal RC oscillator. For example, if the frequency selected is 4 MHz, we know that the instruction time is 1 µs (F_{OSC}/4) and Timer1 has a period of 30.5 µs (1/32.768 kHz). This means within one Timer1 period, the core can execute 30.5 instructions. If the Timer1 registers are preloaded with a known value, we can calculate the number of instructions that will be executed upon a Timer1 overflow.

This calculated number is then compared against the number of instructions executed by the core. With the result, we can determine if re-calibration is necessary, and if the frequency must be increased or decreased. Tuning uses the OSCTUNE register, which has a ±12% tuning range in 0.8% steps.

TIP #13 Idle and Doze Modes

nanoWatt and nanoWatt XLP devices have an Idle mode where the clock to the CPU is disconnected and only the peripherals are clocked. In PIC16 and PIC18 devices, Idle mode can be entered by setting the Idle bit in the OSCON register to ‘1’ and executing the `SLEEP` instruction. In PIC24, dsPIC® DSCs, and PIC32 devices, Idle mode can be entered by executing the instruction “`PWRSAV #1`”. Idle mode is best used whenever the CPU needs to wait for an event from a peripheral that cannot operate in Sleep mode. Idle mode can reduce power consumption by as much as 96% in many devices.

Doze mode is another low power mode available in PIC24, dsPIC DSCs, and PIC32 devices. In Doze mode, the system clock to the CPU is postscaled so that the CPU runs at a lower speed than the peripherals. If the CPU is not tasked heavily and peripherals need to run at high speed, then Doze mode can be used to scale down the CPU clock to a slower frequency. The CPU clock can be scaled down from 1:1 to 1:128. Doze mode is best used in similar situations to Idle mode, when peripheral operation is critical, but the CPU only requires minimal functionality.

TIP #14 Use NOP and Idle Mode

When waiting on a blocking loop (e.g. waiting for an interrupt), instead put the device into Idle mode to disable the CPU. The peripheral interrupt will wake up the device. Idle mode consumes much less current than constantly reading RAM and jumping back. If the CPU cannot be disabled because the loop required some calculations, such as incrementing a counter, instead of doing a very tight loop that loops many times, add NOPs into the loop. See the code example below. A NOP requires less current to execute than reading RAM or branching operations, so current can be reduced. The overall loop count can be adjusted to account for the extra instructions for the NOPs.

Example:

Replace:

```
while(!_T1IF);
```

with Idle mode:

```
IEC0bits.T1IE = 1;
Idle();
```

and replace:

```
while(!_T1IF){
    i++;
}
```

with extra NOP instructions:

```
while(!_T1IF){
   i++;
    Nop();
    Nop();
    Nop();
    Nop();
    Nop();
}
```

TIP #15 Peripheral Module Disable (PMD) Bits

PIC24, dsPIC DSCs, and PIC32 devices have PMD bits that can be used to disable peripherals that will not be used in the application. Setting these bits disconnects all power to the module as well as SFRs for the module. Because power is completely removed, the PMD bits offer additional power savings over disabling the module by turning off the module’s enable bit. These bits can be dynamically changed so that modules which are only used periodically can be disabled for the remainder of the application. The PMD bits are most effective at high clock speeds and when operating at full speed allowing the average power consumption to be significantly reduced.

Static Power Reduction Tips n' Tricks

The following tips and tricks will help reduce the power consumption of a device while it is asleep. These tips allow an application to stay asleep longer and to consume less current while sleeping.

TIP #16 Deep Sleep Mode

In Deep Sleep mode, the CPU and all peripherals except RTCC, DSWDT and LCD (on LCD devices) are not powered. Additionally, Deep Sleep powers down the Flash, SRAM, and voltage supervisory circuits. This allows Deep Sleep mode to have lower power consumption than any other operating mode. Typical Deep Sleep current is less than 50 nA on most devices. Four bytes of data are retained in the DSGPRx registers that can be used to save some critical data required for the application. While in Deep Sleep mode, the states of I/O pins and 32 kHz crystal oscillator (Timer1/SOSC) are maintained so that Deep Sleep mode does not interrupt the operation of the application. The RTCC interrupt, Ultra Low Power Wake-up, DSWDT time-out, External Interrupt 0 (INT0), MCLR or POR can wake-up the device from Deep Sleep. Upon wake-up the device resumes operation at the reset vector.

Deep Sleep allows for the lowest possible static power in a device. The trade-off is that the firmware must re-initialize after wake-up. Therefore, Deep Sleep is best used in applications that require long battery life and have long sleep times. Refer to the device datasheets and Family Reference Manuals for more information on Deep Sleep and how it is used.

TIP #17 Extended WDT and Deep Sleep WDT

A commonly used source to wake-up from Sleep or Deep Sleep is the Watchdog Timer (WDT) or Deep Sleep Watchdog Timer (DSWDT). The longer the PIC MCU stays in Sleep or Deep Sleep, the less power consumed. Therefore, it is appropriate to use as long a timeout period for the WDT as the application will allow.

The WDT runs in all modes except for Deep Sleep. In Deep Sleep, the DSWDT is used instead. The DSWDT uses less current and has a longer timeout period than the WDT. The timeout period for the WDT varies by device, but typically can vary from a few milliseconds to up to 2 minutes. The DSWDT time-out period can be programmed from 2.1ms to 25.7days

TIP #18 Low Power Timer1 Oscillator and RTCC

nanoWatt XLP microcontrollers all have a robust Timer1 oscillator (SOSC on PIC24) which draws less than 800 nA. nanoWatt technology devices offer a low power Timer1 oscillator which draws 2-3 uA. Some devices offer a selectable oscillator which can be used in either a low-power or high-drive strength mode to suit both low power or higher noise applications. The Timer1 counter and oscillator can be used to generate interrupts for periodic wakes from Sleep and other power managed modes, and can be used as the basis for a real-time clock. Timer1/SOSC wake-up options vary by device. Many nanoWatt XLP devices have a built-in hardware Real-Time Clock and Calendar (RTCC), which can be configured for wake-up periods from 1 second to many years.

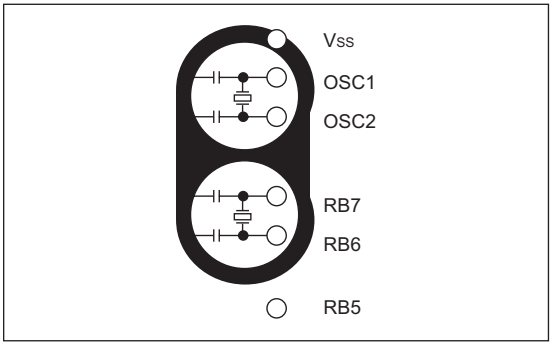
Some nanoWatt devices and all nanoWatt XLP devices can also use the Timer1/SOSC oscillator as the system clock source in place of the main oscillator on the OSC1/OSC2 pins. By reducing execution speed, total current consumption can be reduced.

TIP #19 Low Power Timer1 Oscillator Layout

Applications requiring very low power Timer1/SOSC oscillators on nanoWatt and nanoWatt XLP devices must take PCB layout into consideration. The very low power Timer1/SOSC oscillators on nanoWatt and nanoWatt XLP devices consume very little current, and this sometimes makes the oscillator circuit sensitive to neighboring circuits. The oscillator circuit (crystal and capacitors) should be located as close as possible to the microcontroller.

No circuits should be passing through the oscillator circuit boundaries. If it is unavoidable to have high-speed circuits near the oscillator circuit, a guard ring should be placed around the oscillator circuit and microcontroller pins similar to the figure below. Placing a ground plane under the oscillator components also helps to prevent interaction with high speed circuits.

Figure 19-1: Guard Ring Around Oscillator Circuit and MCU Pins



TIP #20 Use LVD to Detect Low Battery

The Low Voltage Detect (LVD) interrupt present in many PIC MCUs is critical in battery based systems. It is necessary for two reasons. First, many devices cannot run full speed at the minimum operating voltage. In this case, the LVD interrupt indicates when the battery voltage is dropping so that the CPU clock can be slowed down to an appropriate speed, preventing code misexecution. Second, it allows the MCU to detect when the battery is nearing the end of its life, so that a low battery indication can be provided and a lower power state can be entered to maximize battery lifetime. The LVD allows these functions to be implemented without requiring the use of extra analog channels to measure the battery level.

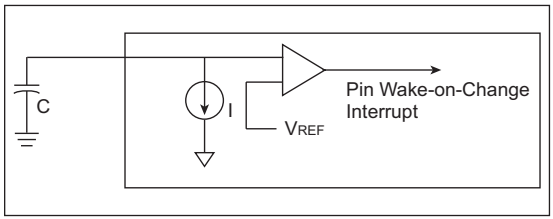
TIP #21 Use Peripheral FIFO and DMA

Some devices have peripherals with DMA or FIFO buffers. These features are not just useful to improve performance; they can also be used to reduce power. Peripherals with just one buffer register require the CPU to stay operating in order to read from the buffer so it doesn’t overflow. However, with a FIFO or DMA, the CPU can go to sleep or idle until the FIFO fills or DMA transfer completes. This allows the device to consume a lot less average current over the life of the application.

TIP #22 Ultra Low-Power Wake-Up Peripheral

Newer devices have a modification to PORTA that creates an Ultra Low-Power Wake-Up (ULPWU) peripheral. A small current sink and a comparator have been added that allows an external capacitor to be used as a wake-up timer. This feature provides a low-power periodic wake-up source which is dependent on the discharge time of the external RC circuit.

Figure 22-1: Ultra Low-Power Wake-Up Peripheral



If the accuracy of the Watchdog Timer is not required, this peripheral can save a lot of current.

Visit the low power design center at: www.microchip.com/lowpower for additional design resources.