AN950

Power Management for PIC18 USB Microcontrollers with nanoWatt Technology

Author: *Rawin Rojvanit*
Microchip Technology Inc.

INTRODUCTION

Reducing power consumption is always desirable in systems where resources may be limited. This is particularly true in USB systems, where designers must not only understand the rules imposed by the USB specification, but know when and how to use which of the available power-saving features.

This application note discusses ways to manage power on the members of the PIC18 family of microcontrollers with nanoWatt technology in USB applications. In this document, we will present a summary of the USB specification in regards to power management and distribution and to describe when and how to use power-saving features in response to different USB events. We will also discuss how the advanced power-saving features of enhanced PIC18 devices with nanoWatt technology can be used to meet these goals.

Our focus in this document is on PIC18 devices with the USB peripheral and uses the PIC18F2455/2550/4455/4550 family of microcontrollers as an example. All references to the device data sheet in this document refer to the data sheet for this family (DS39632).

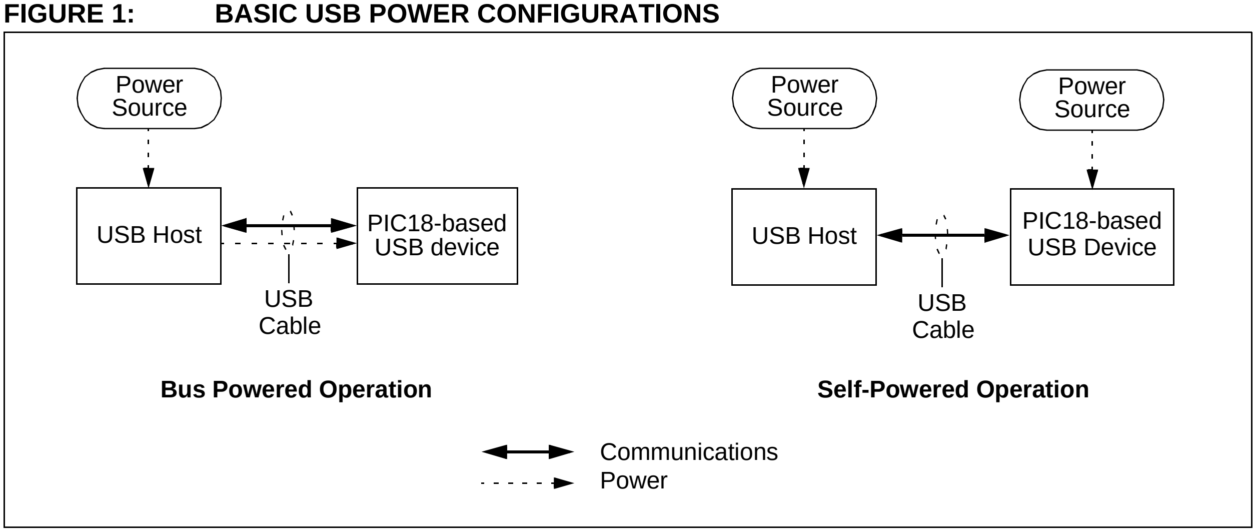
This discussion is limited to USB devices only and is not intended for a USB host. The author assumes that the readers have some basic knowledge about the USB standard. All references to the USB specification in this document refer to *“USB Specification Revision 2.0”*.

OVERVIEW

A basic USB system consists of a USB host and a USB device. The PIC18 family of microcontrollers with the USB peripheral has no USB host functionality and can only assume the role of a USB device.

A USB device can draw power from its host through a USB cable, or from an external power source, to power its functions. These are referred to as bus powered and self-powered configurations, respectively. Figure 1 shows the two basic power configurations in a USB system. In either version, available power may be limited because the host and the device are both battery operated. Reducing the power consumption of the USB device in these cases can prolong the operational life of both the host and the device. Some common power supply circuits are discussed in **Section 17.6 “USB Power Modes”** of the device data sheet.

FIGURE 1: BASIC USB POWER CONFIGURATIONS



USB POWER SPECIFICATION

Information in this section is a summary of topics from the USB specification. Restrictions and rules presented here are not imposed by Microchip Technology, but rather, by the USB specification.

Power Management

The USB bus is said to be Idle when there are no transitions on the differential data lines. This is the same as saying there is no bus traffic.

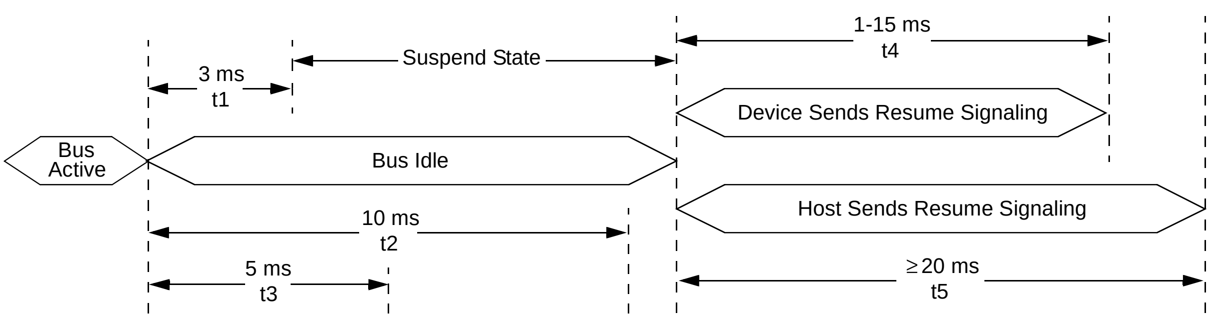
Suspend state is a state in which the device is still attached to the USB and is powered and has not seen bus activity (bus is Idle) for 3 milliseconds. In this state, the device must reduce its current consumption within 10 milliseconds of bus inactivity. The maximum suspend current is dependent on the type of power configuration; these are explained in the next section.

Bus inactivity could be caused by one of two reasons. First, the host itself goes into a Suspend mode and therefore, every USB device attached to it must also suspend. Second, the host can selectively suspend a USB device if it decides that the device is not currently active; this allows the host and other peripherals to continue to function. An example of this is a USB mouse that has not been moved for a period of time; the host can suspend the mouse until it moves again.

The ability of a device to notify the host to re-establish a USB communication link while suspended is called remote wake-up. This action is typically caused by an external stimulus, such as a button press or a mouse movement. The device notifies the host to provide service by sending a resume signal. Remote wake-up support is not mandatory for USB devices. A device specifies if it supports remote wake-up in a configuration descriptor. Before suspending a device, the host must send a request to enable the remote wake-up function. Without a request from the host to enable this function, the device may not send a resume signaling, even if it has the capability to do so. The default mode for devices that support remote wake-up is disabled. Once a device has successfully exited suspend and wakens its host, the host may send a request to disable the remote wake-up function. The enabling and disabling of the remote wake-up function are done through the SET_FEATURE and CLEAR_FEATURE requests.

Re-establishing USB communications can be done in both directions. A host can also wake-up a suspended device by sending a resume signaling. This action is typically caused by a software request on the host computer, such as opening a file on a USB disk drive. A USB device exits Suspend mode when there is bus activity. The resume signaling causes the bus to become active. The details of suspend and resume events timing can be found in Figure 2.

FIGURE 2: SUSPEND AND RESUME EVENTS TIMING



Key:

t1: The bus has to be in the Idle state for 3 ms before the device enters suspend state.

t2: A device may take up to 10 ms of bus inactivity to reduce current consumption.

t3: A device with remote wake-up capability may not send a resume signal in the first 5 ms of bus inactivity. This allows the hubs to get into their suspend state and prepare for propagating resume signaling.

t4: A device with remote wake-up capability must hold the resume signaling for at least 1 ms, but for no more than 15 ms.

t5: The host may signal resume at any time and must hold the resume signaling for at least 20 ms.

Power Distribution

The maximum current a USB device can draw from its host is 500 mA. A device is considered low power if it consumes 100 mA or less and high power if it consumes more than 100 mA. Every device defaults to low-power operation and can only transition to high-power operation under software control. A configured device can have many different power settings depending on the values in the Self-Powered, Remote Wake-up and bMaxPower fields in the configuration descriptor. Additional details about these fields can be found in Chapter 9 of the USB specification. Table 1 shows the maximum supply current under various states and power configurations.

The high-power, bus powered device, with remote wake-up function enabled, has a higher suspend current because its function may require more power to monitor for an external stimulus. On top of complying with the maximum current limitations, all USB devices must be able to provide configuration information with as little as 4.40V from their upstream cables. A USB device can expect the voltage supplied to be in the range of 4.40 to 5.25V. Note that under transient conditions, supply at the hub can drop from 4.40V to 4.07V.

The USB interface in high-power, bus powered and self-powered devices should be able to operate independently while other functions are turned off. This allows the device to operate in Low-Power mode by consuming no more than 100 mA. Details regarding power regulation can be found in sections 7.2.1.3, 7.2.1.4 and 7.2.1.5 of the USB specification. In the event that the external power source is lost, a USB device that consumes more power than specified in bMaxPower must be able to shut down its function and not draw extra power from the host.

TABLE 1: MAXIMUM SUPPLY CURRENT FOR USB DEVICES

State	Maximum Supply Current	
	Not Suspended	**Suspended**
Unconfigured		
	100 mA	500 µA
Configured		
Self-Powered:[1] - Self-Powered = 1 - Remote Wake-up = 0 or 1 - bMaxPower ≤ 100 mA	100 mA	500 µA
Low-Power, Bus Powered: - Self-Powered = 0 - Remote Wake-up = 0 or 1 - bMaxPower ≤ 100 mA	100 mA	500 µA
High-Power, Bus Powered: - Self-Powered = 0 - Remote Wake-up = 0 - bMaxPower > 100 mA	500 mA	500 µA
High-Power, Bus Powered and Remote Wake-up Source: - Self-Powered = 0 - Remote Wake-up = 1 - bMaxPower > 100 mA	500 mA	2.5 mA

Note 1: Self-powered device may draw up to 100 mA from a USB host to allow the USB interface to function when the remainder of the function is powered down.

HANDLING USB INTERRUPTS

There are four basic scenarios to consider when deciding what actions to take to save power.

1. **Bus powered, bus not attached:**

   Since the bus is not attached, there is no power to the device.

2. **Bus powered, bus attached:**

   The USB device is dependent on the host for power. Upon detection of the suspend signaling, the USB device shall reduce its current consumption. This can be done by switching off unnecessary functions, running the CPU from the internal oscillator block, or putting the CPU into Sleep or Idle mode. The microcontroller will maintain a valid voltage level on the USB differential data lines while in Sleep mode to stay attached to the host.

3. **Self-powered, bus not attached:**

   The USB communication interface is inactive in this scenario. The device can go into Power-Saving mode whenever appropriate to conserve external power source.

4. **Self-powered, bus attached:**

   The USB interface shall be active while not in the Suspend mode. Other functions can go into Power-Saving mode whenever appropriate to conserve the external power source.

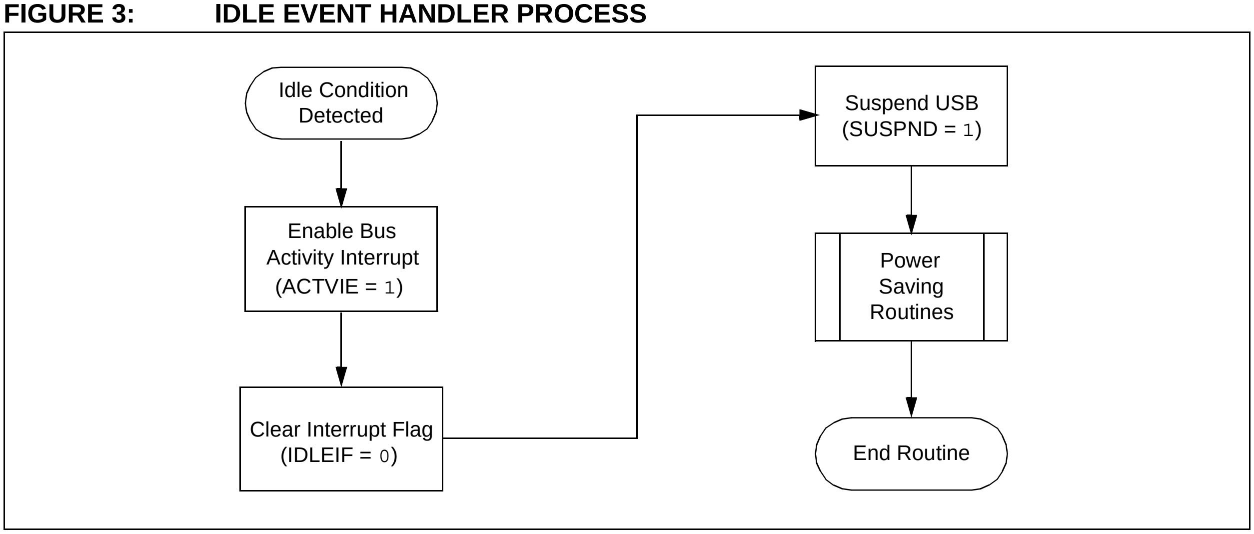
Power-saving procedures are controlled by peripheral control registers. There are eight control bits that are used during USB suspend and resume signaling. These bits are listed in Table 2. Descriptions and in-depth details about each of the control bits listed can be found in the device data sheet.

The IDLEIF, ACTVIF and URSTIF bits are interrupt sources. They are enabled by setting the corresponding IDLEIE, ACTVIE and URSTIE enable bits (= 1). After bus Reset, the Idle detect interrupt should be **enabled** and the bus activity interrupt should be **disabled**. This setup prepares the device to be able to detect the suspend signaling from the host. When IDLEIF is set, it indicates that an Idle condition has been detected for 3 milliseconds or more and the USB module should go into the Suspend mode. Figure 3 shows a typical program flow for servicing an IDLEIF event.

TABLE 2: CONTROL BITS USED IN SUSPENDING AND RESUMING USB OPERATION

Register	Control Bits	Function
USB Interrupt (UIR)	IDLEIF	Idle Detect Interrupt bit (Bus Idle for ≥ 3ms)
	ACTVIF	Bus Activity Interrupt bit
	URSTIF	USB Reset Interrupt bit
USB Interrupt Enable (UIE)	IDLEIE	Idle Detect Interrupt Enable bit
	ACTVIE	Bus Activity Interrupt Enable bit
	URSTIE	USB Reset Interrupt Enable bit
USB Control (UCON)	RESUME	Resume Signaling Enable bit
	SUSPND	Suspend USB bit

FIGURE 3: IDLE EVENT HANDLER PROCESS

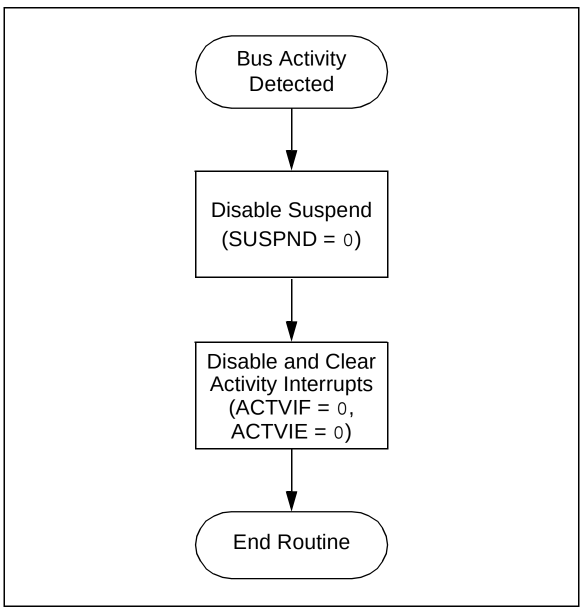


While the USB interface is suspended, only the bus activity interrupt is active; all other USB interrupts are inactive. The power-saving routines block will be discussed in more detail in later sections. Note that power-saving routines can be bypassed if the device is self-powered. Suspending the USB interface itself reduces some current consumption.

When the USB interface is suspended, the USB peripheral clock is disabled and the internal USB voltage regulator and transceivers go into Low-Power mode. In this mode, the voltage regulator, the transceivers, the 1.5 kΩ pull-up resistor and the 15 kΩ pull-down resistor consume approximately 340 μA out of the 500 μA typically allowed by the USB specification. This leaves approximately 160 μA for other functions, which is sufficient to operate the CPU using the internal oscillator block. When an external USB voltage regulator or transceivers are used, it is important to make sure that the suspend current restrictions are met.

While suspended, a USB device can be awakened by its host. A resume signaling from the host causes the ACTVIF bit to set. Figure 4 shows a typical program flow for servicing an ACTVIF event.

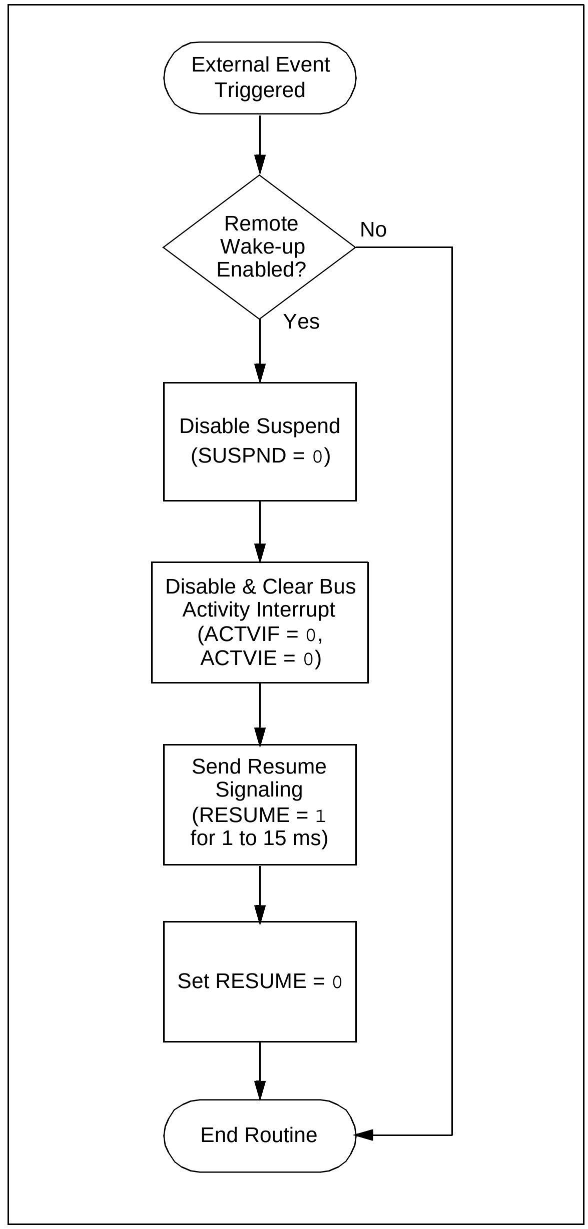
FIGURE 4: BUS ACTIVITY HANDLER PROCESS FLOW DIAGRAM



If a USB device is suspended and the remote wake-up function is enabled, it can wake-up the host. Figure 5 shows a typical program flow for waking up the host.

In the event that the host sends out a Reset signal to a suspended device, the bus activity interrupt will be set first, giving an opportunity for the firmware to clear the SUSPND bit. Only after the SUSPND bit is cleared will the URSTIF bit be set. The URSTIE bit should always be set to '1' to allow the host to reset the device at any time. The difference between the Reset and resume signaling is the state of the differential data lines. Reset signaling has both data lines driven low, while the resume signaling toggles both data lines, transitioning the bus to a state known as K state. Refer to the USB specification for more details regarding bus states.

FIGURE 5: REMOTE WAKE-UP PROCESS FLOW DIAGRAM



POWER-SAVING ROUTINES

The process flows shown in this section are equivalent to the power-saving routine block shown in Figure 3. Information presented in this section is intended to be a brief discussion summarizing key issues. Detailed information regarding the Power Managed mode usage, limitation and restriction can be found in **Section 3 "Power Managed Modes"** of the device data sheet.

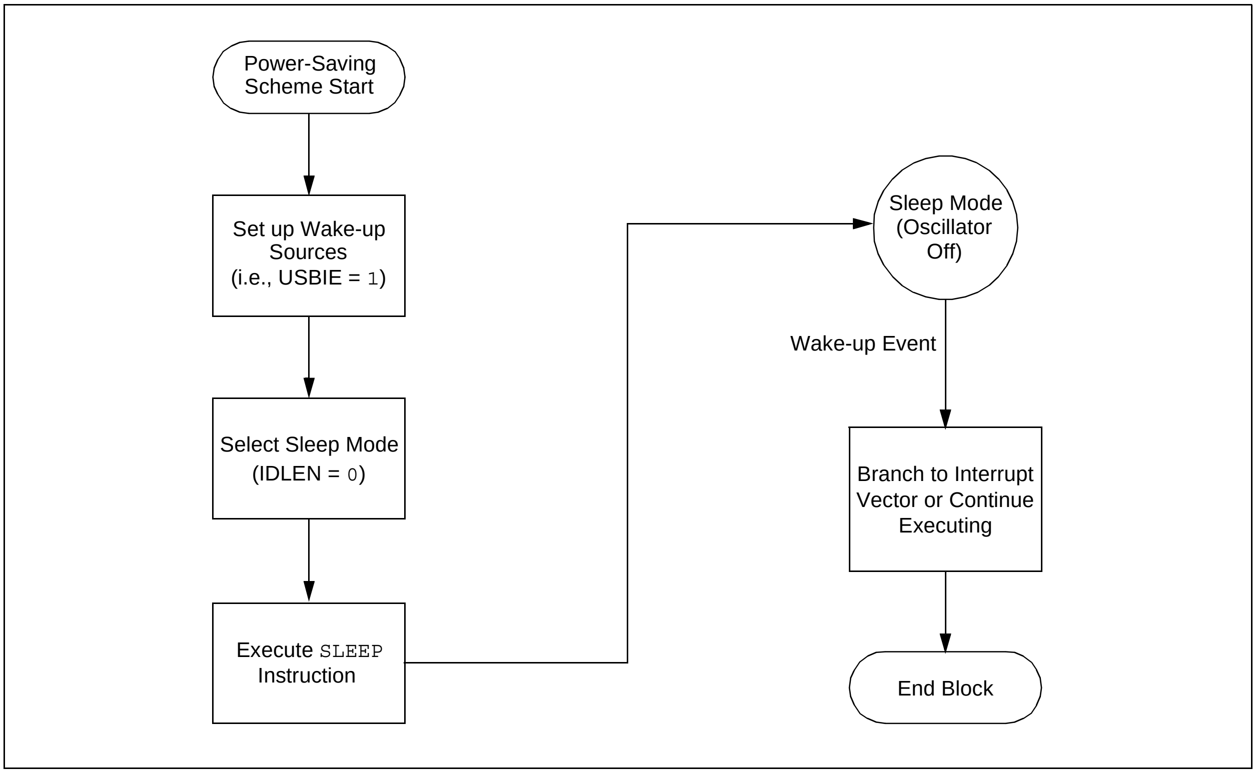
Sleep Mode

The most basic power-saving method is to put the device into Sleep mode. In this mode, the selected oscillator is shut down. Since the CPU is not executing instructions, the only exits from the Sleep mode are by interrupt, Reset or WDT time-out.

The bus activity interrupt can be a wake-up source when USBIE is set. This is necessary because all USB interrupts are funneled to a single interrupt, USBIF. If the microcontroller's global interrupt (GIE/PEIE) is not enabled, the CPU will resume operations upon waking by executing the instruction immediately following the `SLEEP` instruction. If the interrupt is enabled, on the other hand, the CPU will branch to an interrupt vector, service the wake-up event and then return to the normal program code.

Figure 6 shows a typical program flow for entering and exiting the Sleep mode. This is only recommended when the USB module is disabled or suspended.

FIGURE 6: SLEEP MODE PROCESS FLOW



Microcontroller Idle Mode

Another power-saving method is to put the device into the Idle mode. This is not to be confused with the USB bus Idle; for clarity, we shall refer to it as the Microcontroller Idle mode. In this mode, the CPU is not clocked but the peripherals are clocked. It can be used just like the Sleep mode when the USB module is suspended. The advantage of using Microcontroller Idle mode is that other peripherals can continue operating as long as the overall current consumption is less than the maximum suspended current allowed. However, the mode is most advantageous when the device is not in the USB Suspend mode.

Consider a full speed USB application that is designed to use the Interrupt transfer method and be polled once every 2 ms. The CPU can go into the Microcontroller Idle mode most of the time to save power while the USB interface is still active. Remember that to set TRNIF (UIR<3>) as a wake-up source, both TRNIE (UIE<3>) and USBIE (PIE2<5>) must be set (= `1`). To prevent the Start-of-Frame Token Interrupt (SOFIF) from waking up the system, just clear the Start-of-Frame Token Interrupt Enable bit (SOFIE).

Figure 7 illustrates the sequence of some common events when using the Microcontroller Idle mode. Figure 8 shows a typical program flow for entering and exiting the Microcontroller Idle mode.

FIGURE 7: TIMING OF IDLE MODE ENTRY AND EXIT EVENTS

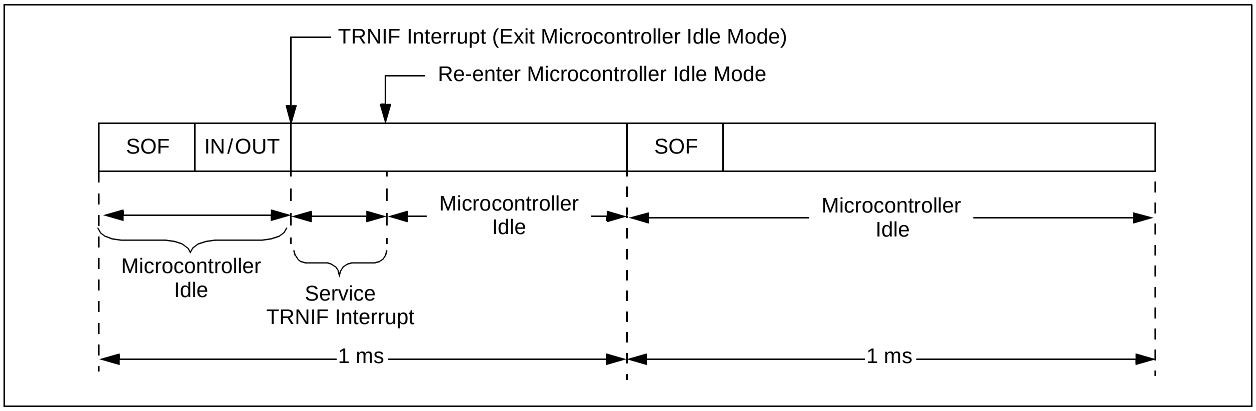
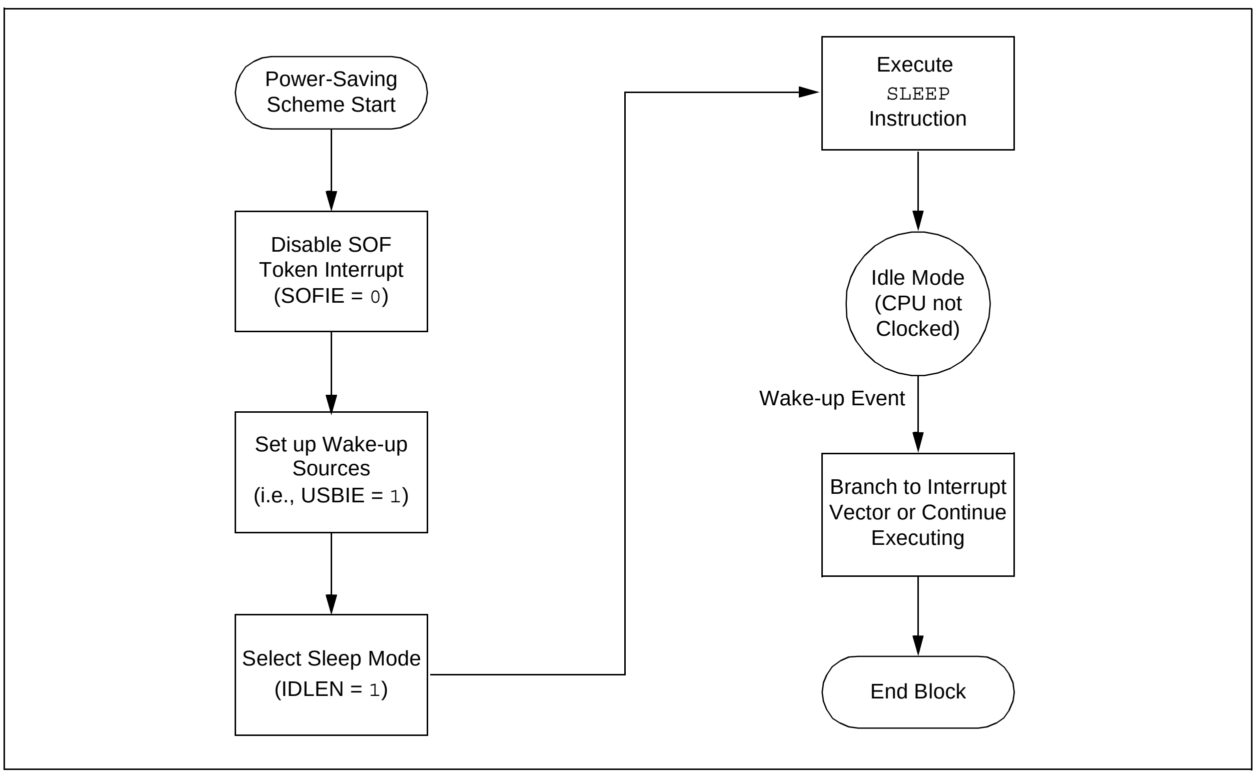


FIGURE 8: MICROCONTROLLER IDLE MODE HANDLER PROCESS



Clock Switching

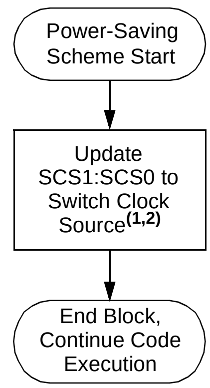
The microcontroller has the option to switch to a slower clock source to reduce power consumption. This has the advantage of allowing the CPU to continue executing instructions, even when the device is in the Suspend mode. An example would be switching from the primary oscillator running at 48 MHz to the internal oscillator block running at 31 kHz. The clock speed can be lowered because the high-frequency clock to the USB module is not needed during suspend. Refer to **Section 2.4 "Clock Sources and Oscillator Switching"** of the data sheet for more details regarding clock switching.

Figure 9 shows an example of clock switching as a power-saving scheme. Figure 10 shows a modified bus activity interrupt handler. An extra step is required to restore the original clock source to the CPU.

Using the 96 MHz PLL

The 48 MHz clock required by the USB module operating at full speed can be clocked from either the 96 MHz PLL, with a divide-by-two module, or directly from a 48 MHz external clock. In bench measurements, the PLL module consumes approximately 2 mA at 5V when enabled. Even so, power consumption using the PLL is much less than that required by an external clock. Therefore, more power can be saved by using the 96 MHz PLL. Also, the frequency input required by the 96 MHz PLL is 4 MHz. This means a cheaper crystal oscillator can be used in place of an expensive high-frequency external clock. A range of oscillator frequencies can be selected as long as it can be divided by a prescalar to 4 MHz. **Section 2.0 "Oscillator Configurations"** of the device data sheet discusses these options in much greater detail.

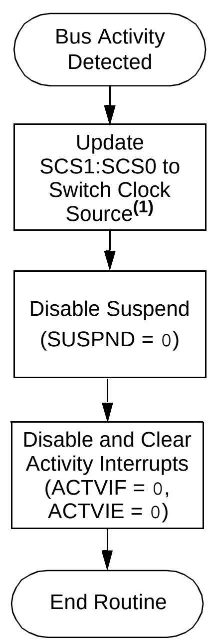
FIGURE 9: CLOCK SWITCHING PROCESS FLOW DIAGRAM



Note 1: The default internal oscillator frequency is 1 MHz. The OSCCON register should be modified during start-up to set the desired internal oscillator frequency (i.e., OSCCON<IRCF2:0> = `000` for 31 kHz).

2: There are start-up times associated with the switching system clock. Care should be taken to make sure that the oscillator and PLL are stable before carrying out time sensitive tasks (i.e., check OSCCON<OSTS> and OSCCON<IOFS> flags).

FIGURE 10: BUS ACTIVITY WITH CLOCK SWITCHING HANDLER



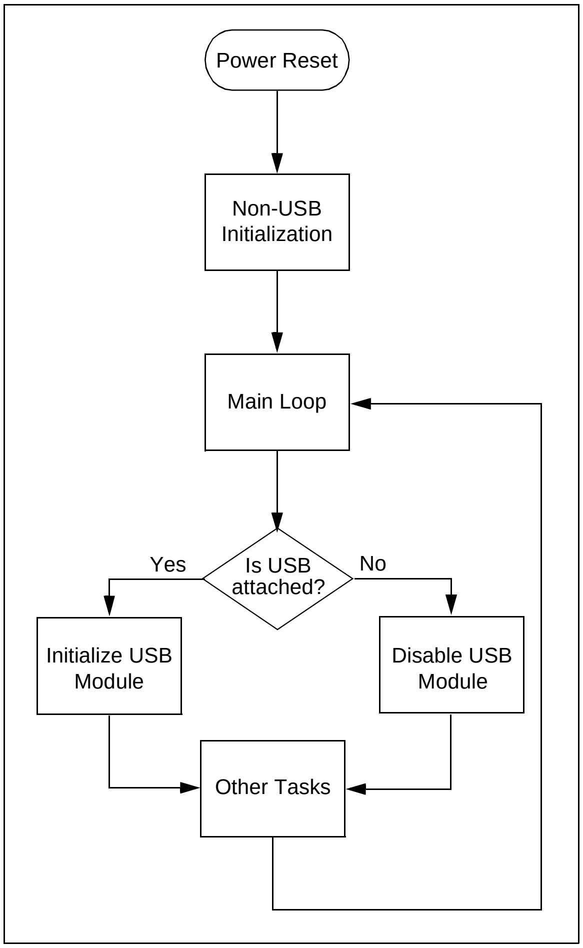
Note 1: There are start-up times associated with the switching system clock. Care should be taken to make sure that the oscillator and PLL are stable before carrying out time sensitive tasks (i.e., check OSCCON<OSTS> and OSCCON<IOFS> flags).

DETECTING BUS ATTACHMENT AND DETACHMENT

If a self-powered device is not attached, enabling the USB module and waiting for a USB Reset event is inefficient because the module consumes power unnecessarily. Bus attachment and detachment events can be detected in many ways. One way is to use an external interrupt pin to detect the change in the bus power supply input. After powering up, a device can check if the bus is attached by polling the external interrupt. If the bus is attached, the USB module is enabled. If not, the device should wait for an external interrupt event before polling the pin again. Only when the input is high should the USB module be enabled.

Figure 11 shows a simple USB bus detection and module initialization program flow.

FIGURE 11: BUS DETECTION PROCESS



SUMMARY

Designing a power efficient USB system depends on evaluating all of the application's requirements. There is no single power-saving design solution. Nonetheless, every system must comply with the USB specification. Knowing the rules and limitations is the first step in the design process.

It is obvious that any functions that are not being used in a given state should be disabled or switched to use a slower clock to conserve power. The whole system can be put into Sleep mode if the USB interface is suspended and if no other tasks need to be serviced. The Microcontroller Idle mode plays an important role in conserving system power because it can be entered even if the USB module is still active.

Finally, to effectively conserve power is to strike a balance between meeting design requirements, utilizing power-saving features where possible and staying compliant to the USB specification.

REFERENCES

"PIC18F2455/2550/4455/4550 Device Data Sheet" (DS39632), Microchip Technology Inc., 2004, http://www.microchip.com

"USB Specification Revision 2.0", USB Implementers Forum Inc., 2000, http://www.usb.org

APPENDIX A: DESIGN EXAMPLE

A.1 Requirements:

Assume that a USB device is self-powered and requires a maximum of 400 mA of current. Its power source is a battery which should be conserved whenever possible to maximize the operational life. The device has to carry out certain tasks, whether it is connected to the USB or not. These few tasks consume over 100 mA.

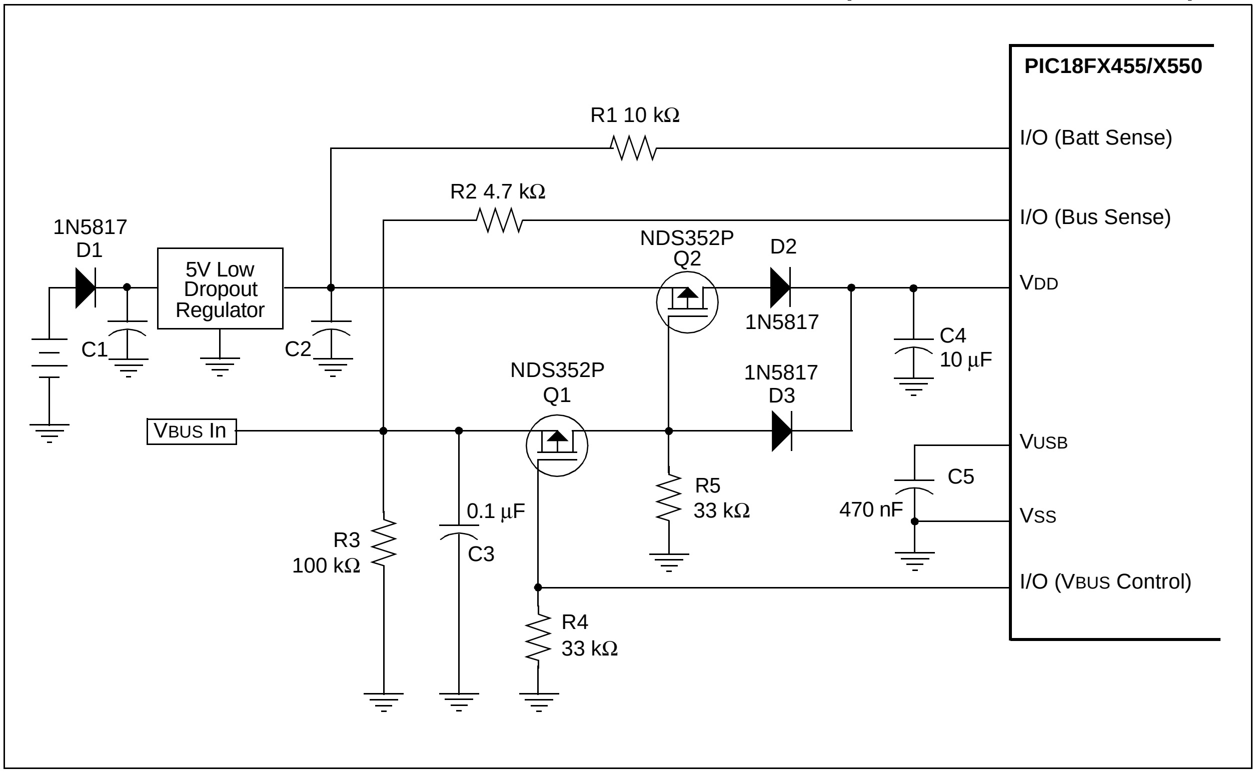
A.2 Things you can do:

- Since the maximum power requirement is under 500 mA, the device could report to the host that it is a bus powered device even though it has its own power. This way, the device could draw up to 500 mA of current from the host. If it was to report itself as a self-powered device, it will only be allowed to consume up to 100 mA (see Table 1). When the device is attached to the bus, it should switch its power source from the battery to the USB cable.

  There is a catch to this scheme, however. When switching power to the USB cable, the device can only consume up to 100 mA initially until it is configured. During this low-power period, the device must power down some of the functions to lower the current consumption.

  Figure A-1 shows a circuit which can sense the bus attachment, power down some functions, then turn on Q1 allowing VBUS to propagate which causes Q2 to turn off.
- Implement a battery recharging circuit, allowing the battery to be recharged while the device is connected to the bus.
- Use other firmware control power-saving schemes as discussed in this application note.

FIGURE A-1: AN EXAMPLE FOR IMPLEMENTING DUAL POWER (BUS POWER DOMINANCE)



Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westford, MA
Tel: 978-692-3848
Fax: 978-692-3821

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

Japan - Kanagawa
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark - Ballerup
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Massy
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Ismaning
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/20/04