Enabling Low-Power Windows 8 HID Over I2C Applications Using MSP430™ Microcontrollers

ABSTRACT

The Human Interface Device (HID) protocol was originally targeted at human interface devices such as keyboards, touchpads, mice, and joysticks. It was originally developed to run over USB and *Bluetooth®*. For Windows 8®, Microsoft created a new HID miniport driver that allows devices to communicate to SoC over an Inter-Integrated Circuit (I2C™) bus. Power consumption and design simplicity are key advantages of HID over I2C compared to HID over USB.

This application report provides an overview of HID over I2C and its advantages. It provides detailed power consumption analysis of a keyboard controller application using both mechanisms. Also discussed is how HID over I2C can be implemented using the MSP430F5229, a device that features 1.8-V I/O rails to allow direct interface to application processors.

Contents

List of Figures

List of Tables

1 Introduction

Human Interface Device (HID) was a protocol developed to simplify the process of connecting accessories such as mouse, keyboard and joystick to the PC.

The HID protocol makes implementation of devices very simple. Devices define their data packets and then present a "HID descriptor" to the host. The HID descriptor is a hard coded array of bytes that describes the device's data packets. This information includes how many packets the device supports, how large are the packets, and the purpose of each byte and bit in the packet. A HID driver running on the processor OS parses data and enables application functionality.

HID has enabled rapid innovation and development, and prolific diversification of new human interface devices. Most operating systems recognize standard USB HID devices, such as keyboards and mice, without needing a specialized driver.

HID was originally developed to run over USB and Bluetooth. For Windows 8, Microsoft created a new HID miniport driver that allows devices to communicate to SoC over an Inter-Integrated Circuit (I2C bus). I2C is an extremely simple, efficient and low-power protocol and has been used for over a decade in phone and embedded platforms.

A key advantage of using HID over I2C in comparison to HID over USB is the power saving, which is of paramount importance in battery powered applications such as tablets or smart phones. In a keyboard controller application using a MSP430, a HID over I2C solution showed 87-99% reduction in power consumption relative to a HID over USB solution. The same power advantages can be extended to other applications such as sensor hubs, touchpads, or HID touch screens.

This application report introduces HID over I2C and discusses its power advantages over HID over USB in a keyboard control application. This paper also discusses how HID over I2C can be implemented using the MSP430F522x microcontroller. MSP430F522x features a 1.8-V IO rail that allows interface to application processors without the need for level translators and offering system-level advantages such as reduced system cost and increased flexibility.

2 HID Over I2C Architecture Overview

The HID over I2C architecture is described in depth in the Microsoft HID Over I2C Protocol specification [8]. The following is an excerpt of the high level architecture overview.

The HID I2C driver stack consists of existing and new components supplied by Microsoft, as well as components provided by the I2C silicon manufacturer. Figure 1 shows the stack and these components.
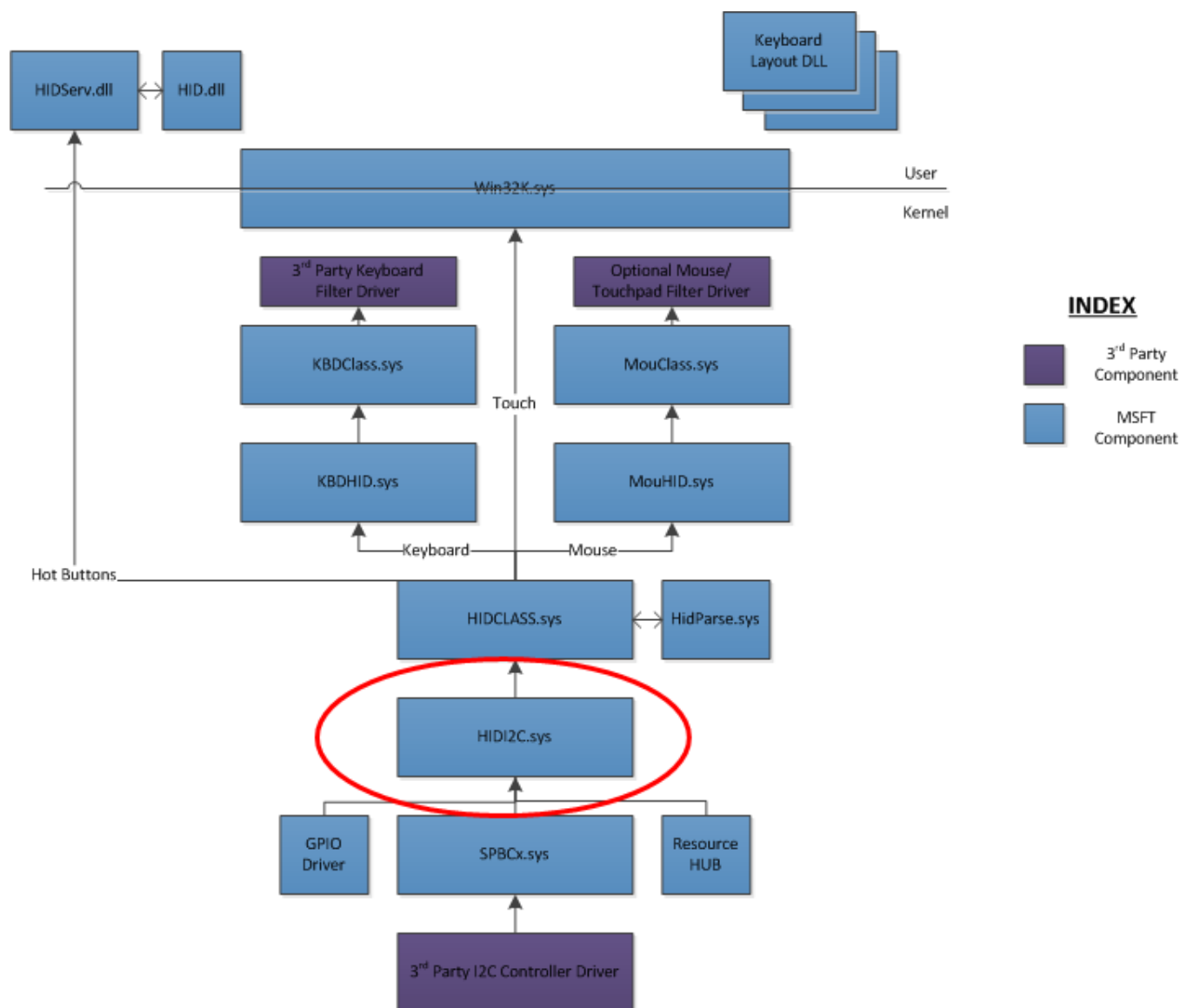
Figure 1. HID Over I2C Architecture Overview (From Microsoft)

Windows 8 provides an interface for low-power, simple buses to communicate effectively with the operating system. This interface is referred to as simple peripheral bus (SPB), and it supports buses like Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI). Windows 8 provides a KMDF-based HID miniport driver that implements version 1.0 of the protocol specification for HID over I2C. This driver is named HIDI2C.sys. Windows loads this driver based on a compatible ID match, which is exposed by the Advanced Configuration and Power Interface (ACPI). The driver ensures that applications that use HID IOCTLs application level compatibility for software that leverages the HID IOCTLs and API set. A device asserts the host when it requires attention or has data. However, before the assertion occurs, a GPIO connection must exist.

Third party devices like keyboard, touch pad, sensor hubs attaching to a Windows 8 processor over I2C, need to implement a communication mechanism compliant with the Windows 8 drivers. A fully compliant HID over I2C software solution for MSP430 MCUs is available from TI. Contact your TI sales representative for more information.

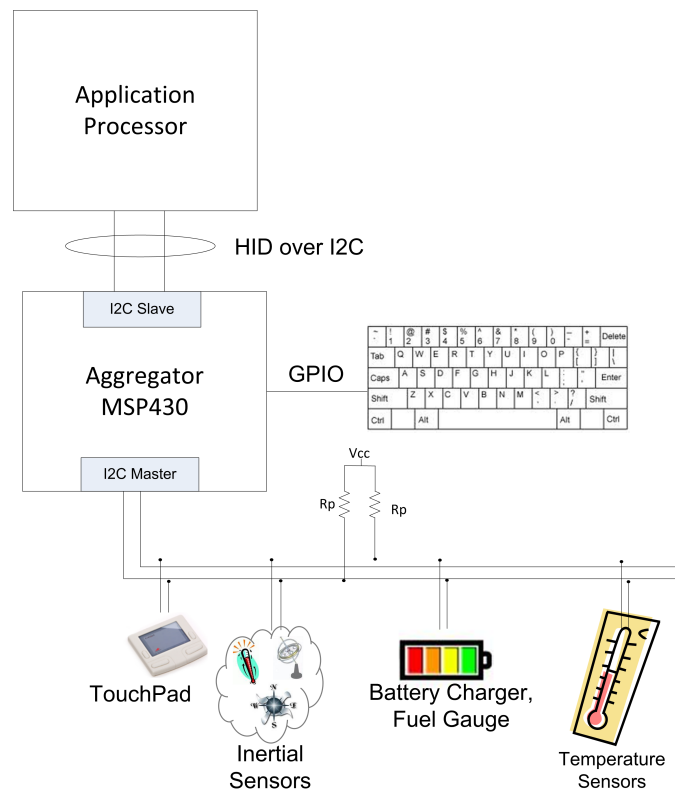## 3   Use Cases for HID Over I2C

Figure 2. Typical HID Over I2C Implementation

HID over I2C can be used for modules that typically reside inside a low-power system such as tablet, handset, ultra-book to communicate with the application processor using a low power interface. Examples of these are touchpad, inertial sensors, temperature sensors, battery charger, and fuel gauge.

In Figure 2, an MSP430 is used as a ultralow-power aggregator for all these modules. The reason for doing so rather than connecting devices directly to the application processor is as follows.

First, using an ultra-low power MCU to read data from various devices in the system allows the power hungry application processor to be in sleep saving power and extending battery life. For instance the MCU can read the battery gauge or temperature sensors periodically and wake up the application processor only when needed. A similar case can be made about using an ultra-low power MCU to aggregate data from sensors. An application note on use of MSP430 as a nine-axis sensor fusion processor is available [9].

Second, many devices inside a tablet interface over I2C, USB, UART, or GPIO and do not support HID over I2C. Using an MSP430 as a "protocol translator" allows device manufacturers and OEMs to comply with Windows 8 HID over I2C specifications. Also, manufacturers of modules such as touchpad can embed a MSP430 inside their module to natively support HID over I2C. This allows seamless integration with an application processor without the need to develop custom drivers for their solution.

In general, modules internal to the device requiring slower interfaces are best suited for HID over I2C. Modules that need faster communication, robust error checking mechanisms, and connection to external peripherals (outside the device) are better suited for HID over USB.

4   Comparing Keyboard Controller Implementations: HID Over I2C vs HID Over USB

HID over I2C provides several advantages when compared to the HID over USB implementation, but one of the most important features is the lower power consumption.

This section analyzes a keyboard implementation using USB and I2C interfaces and compares the expected power consumption.

4.1   *HID USB Keyboard Implementation*

For the purpose of this analysis, a USB keyboard was implemented using a MSP430F5510 as shown in the application report *USB Keyboard Using MSP430 Microcontrollers* (SLAA514).

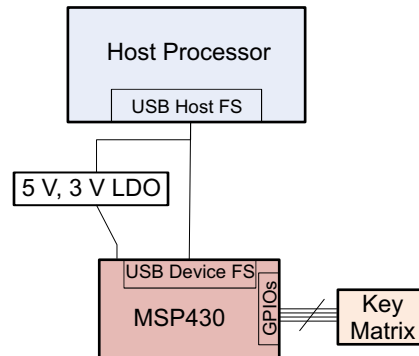This implementation can be simplified as shown in Figure 3.

Figure 3. HID USB Keyboard Block Diagram

Because USB provides a 5-V rail, the same bus is used to power the microcontroller through an external LDO. Note that this section analyzes the power consumption of the MSP430 as a keyboard device, without considering the power consumption of the rest of the system (for example, the host processor and LDOs).

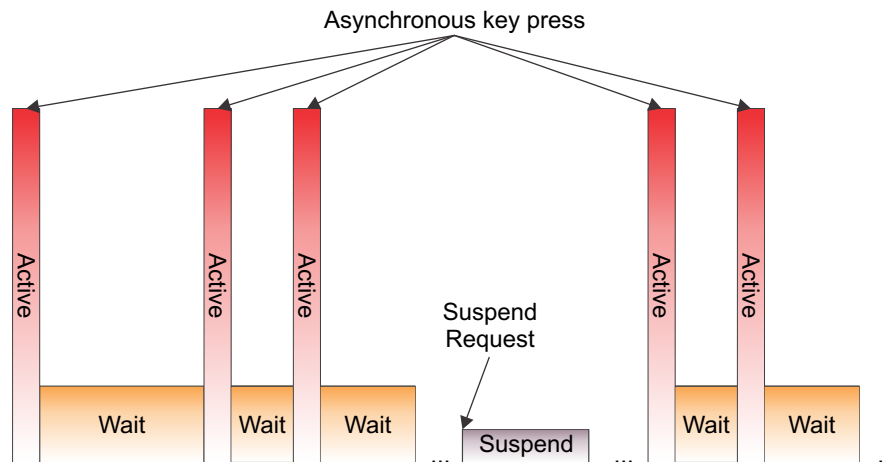Figure 4 shows the power profile of the USB implementation:

Figure 4. Power Profile of MSP430 HID USB Keyboard Implementation

After it is enumerated and functional, the MSP430 is in one of the following states:

*   **Active**: The device is in Active mode when a key is pressed, forcing the CPU to wake-up and scan the key matrix. The MSP430 runs an algorithm to detect the pressed keys and sends the result to the host
*   **Wait**: This is the idle state of the keyboard, with USB enabled but waiting for a key press. Because the USB is enabled, the MSP430 cannot go to the lowest power LPM3 mode, but it can go to LPM0. In this mode, the CPU is not executing code but peripherals are running from the high-frequency bus clock (SMCLK).
*   **Suspend**: This mode is typically implemented when a host (that is, the Windows based host) is in a Standby mode and it issues a Suspend request to the MSP430 through the USB interface. In this mode, the USB is suspended and the MCU is in a lower power mode LPM3. The keyboard is not used for regular typing while the host is sleeping, but the MSP430 can still detect key presses and wake up the system to resume the OS operation. Table 1 shows measurements taken using the evaluation board from SLAA514 [4].

Copyright © 2012, Texas Instruments Incorporated

Table 1. MSP430F5510 Power Consumption of USB HID Keyboard

	Active	Wait	Suspend
I_{MCU}	1.91 mA [1]	260 µA [2]	6 µA [3]
I_{USB_VCC} [4]	713 µA	766 µA	-
TOTAL I_{CC}	2.623 mA	1026 µA	6 µA
TOTAL I_{VBUS} [5]	7.798 mA	6.209 mA	360 µA
TOTAL P_{MSP430} [6]	38.99 mW	31.045 mW	1.8 mW

[1]   Active Mode at 8 MHz and PMMCOREVx = 2 (minimum for USB use)
[2]   LPM0 at 8 MHz and PMMCOREVx = 2
[3]   LPM3 using REFO and SVSH in Full Performance mode
[4]   Current consumption adder for XT2, PLL and USB
[5]   Current consumption measured on USB VBUS (5V)
[6] I_{VBUS} × 5V

4.2   HID Over I2C Keyboard Implementation

To run HID over I2C tests, a similar system was implemented using the same MSP430F5510 and keyboard implementation but replacing the USB interface with I2C, as shown in Figure 5.
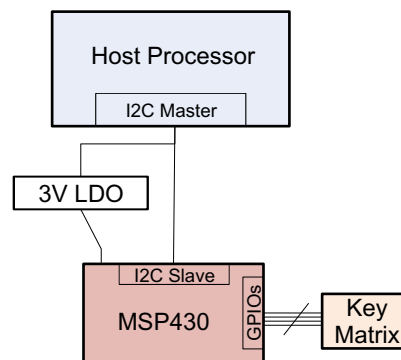
Figure 5. HID Over I2C Keyboard Block Diagram

An external supply provides power to the MSP430, and it is important to note that a host processor and some other parts of the system can use the same rail (for example, it might be derived from a laptop battery using an LDO).This section analyzes the power consumption of the MSP430 as a keyboard device, without considering the power consumption of the rest of the system (for example, the host processor and LDOs).

Figure 6 shows the power profile of the I2C implementation:
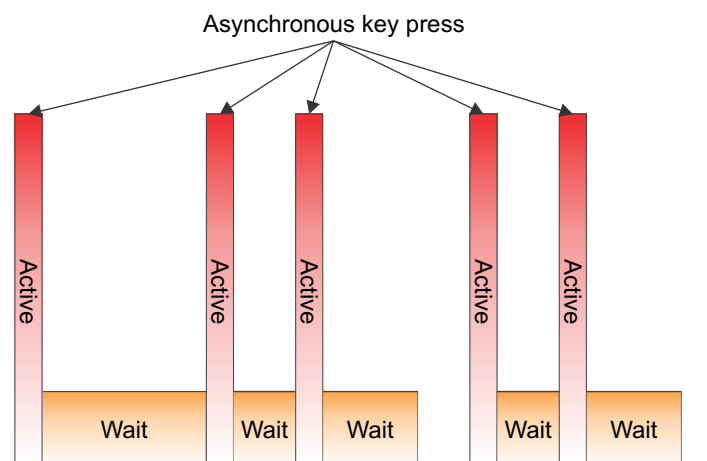
Figure 6. Power Profile of MSP430 HID Over I2C Keyboard Implementation

The MSP430 is in one of the following states during normal operation of a keyboard using I2C:

- **Active**: Similar to the USB implementation, the device is in Active mode when a key is pressed, forcing the CPU to scan the matrix and decode the pressed key(s).

- **Wait or Suspend**: Contrary to USB, in the I2C implementation the MSP430 can go to the lower power mode LPM3 even in Wait mode, because it does not require high-frequency clocks to remain active. The MSP430 can wake up on a key press or when receiving I2C messages from the Master.

Table 2 shows measurements taken using a modified version of the evaluation board from SLAA514 without USB and with I2C.

**Table 2. MSP430F5510 Power Consumption for HID
Over I2C Keyboard**

	Active	Wait or Suspend
I_{MCU}	1.60 mA[1]	6 µA[2]
TOTAL P_{MSP430}[3]	4.8 mW	.018 mW

[1]   Active Mode at 8 MHz and PMMCOREVx = 0
[2]   LPM3 using REFO and SVSH in Full Performance mode
[3] I_{MCU} x 3V

4.3   *Summary of MCU Power Comparison*

As can be seen from Figure 7, MCU running HID over I2C keyboard consumes just 12% of the Active power and 0.06% of the Wait power of a HID over USB keyboard. As the keyboard is mostly waiting for a key press, the average power consumption is usually close to Wait mode, but it can vary depending on the frequency and number of pressed keys. The worst case being a power consumption close to Active Mode.

Despite the fact that in Active Mode both CPUs are executing code at the same frequency, the USB implementation requires a 5-V rail, a high-frequency crystal, a 48-MHz PLL, and the USB PHY, and the CPU must execute at a minimum of PMMCOREVx = 2. Because the I2C implementation does not have these requirements, the difference is evident.

The difference in Wait mode is even more noticeable, because the MSP430 needs to remain at least in LPM0 when the USB is active, but it can go to the lower power LPM3 mode in the I2C implementation.

The I2C implementation makes no distinction between Wait and Suspend mode, because the functionality is the same with the device in LPM3 and the device can still wake-up on I2C activity or key presses. In the USB implementation of Suspend Mode, the MSP430 is also in LPM3 but part of the USB circuitry needs to remain active to detect activity, thus consuming more power.

For the purpose of this comparison, LPM3 is used as the lowest power mode, but depending on the application, LPM3 could be replaced by LPM4 or LPMx.5 thus reducing the power consumption of Wait and Suspend modes even more.
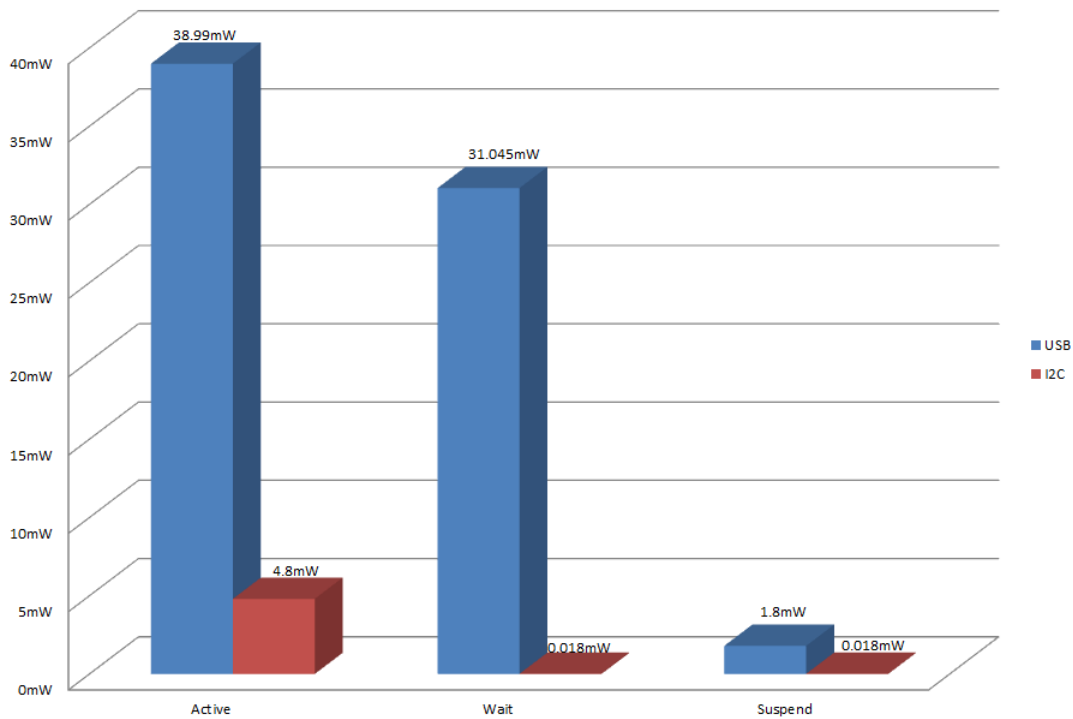
Figure 7. MSP430 Power Consumption of HID Keyboard on USB vs I2C

4.4 *Application Processor Power Consumption*

Considering power consumption of the application processor, the benefit of HID over I2C is even more significant. The following is an analysis of the power consumption on an application processor (TI TMS320AM3517 Cortex-A8 MPU) in a keyboard controller example. The values may vary depending on the choice of processor, bus load, and other system variables but the underlying analysis is still applicable. The power consumption on AM3517 was estimated using a power estimation tool [5].

Figure 8 shows the estimated average power consumption of a keyboard implementation:
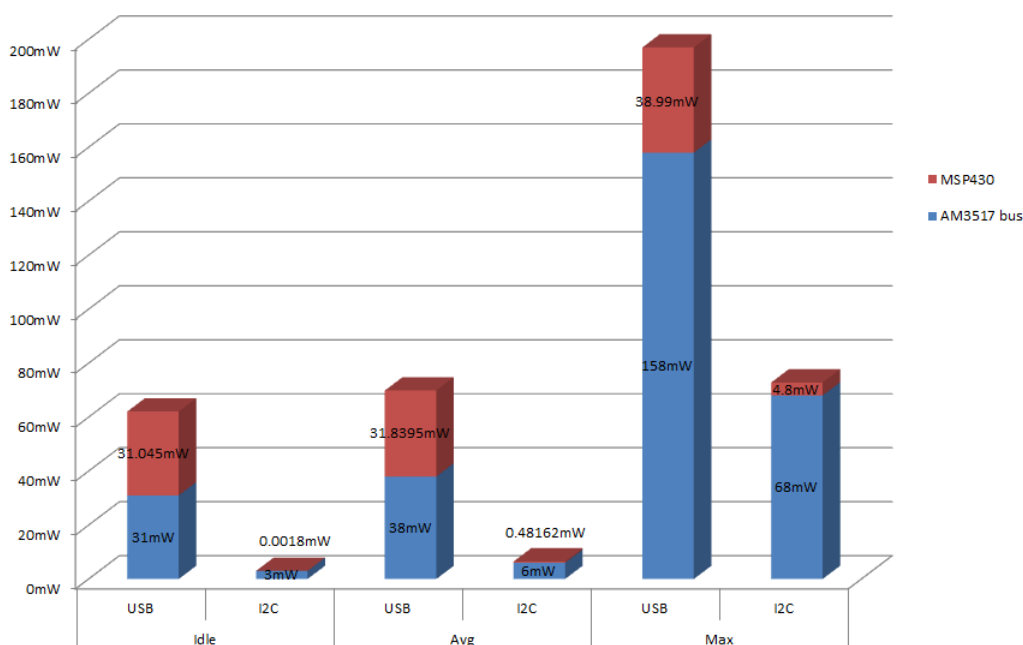
A   AM3517 power is only the adder for the corresponding module (I2C/USB), not for the whole processor as estimated
        by PET (http://www.ti.com/tool/powerest).

B   AM3517 USB power includes PHY based on percentage of use.

C   AM3517 I2C power is part of the core domain and cannot be powered independently. The estimates are based on
        use for all miscellaneous peripherals including GPIOs and McSPI. The individual adder of I2C can be considered
        insignificant.

D   Idle power assumes "wait" mode for MSP430, 0% bus use for AM3517.

E   Average power consumption considers 10% Active mode for MSP430, 5% bus use for AM3517.

F   Max power consumption assumes 100% Active mode for MSP430 and 100% bus use for AM3517.

Figure 8. Power Consumption Estimates of Keyboard Implementation

Powering the application processor's USB bus for a keyboard control application considerably increases
the system power consumption. In idle mode, USB module and PHY on application processor consumes
31 mW whereas the I2C module consumes less than 3 mW. In active mode, the USB module and PHY on
the application processor can consume up to 158 mW compared to 68 mW for an I2C module.

In a typical use case (Average power), if we consider both MCU and applications processor, power, the
HID over I2C keyboard consumes only 3% of the power consumed by the USB keyboard.

Note that for the purpose of this comparison, the estimates are only considering the effect of enabling the
corresponding module on the AM3517, not the whole processor. Also, the I2C module is part of the core
domain and cannot be powered independently, so the estimate is based on use for miscellaneous
peripherals including GPIOs, 4xMcSPIs and 3xI2Cs. The individual adder for a single I2C module cannot
be estimated but it can be considered insignificant.

The estimated average power considers 10% active time for MSP430 and 5% bus use, and a real
application can vary depending on usage.

4.5   *Communication Bus*

We have seen the significant difference in power consumption when using the MSP430 as an HID device
over I2C instead of USB and the effect of this change on a host. In addition, an important part of the
overall system power consumption is the bus implementation and topology.

A typical application requires multiple HID devices (for example, keyboard, mouse, touch sensors, and accelerometers) sharing the bus. Figure 9 shows a basic representation of both USB-based and I2C-based systems.
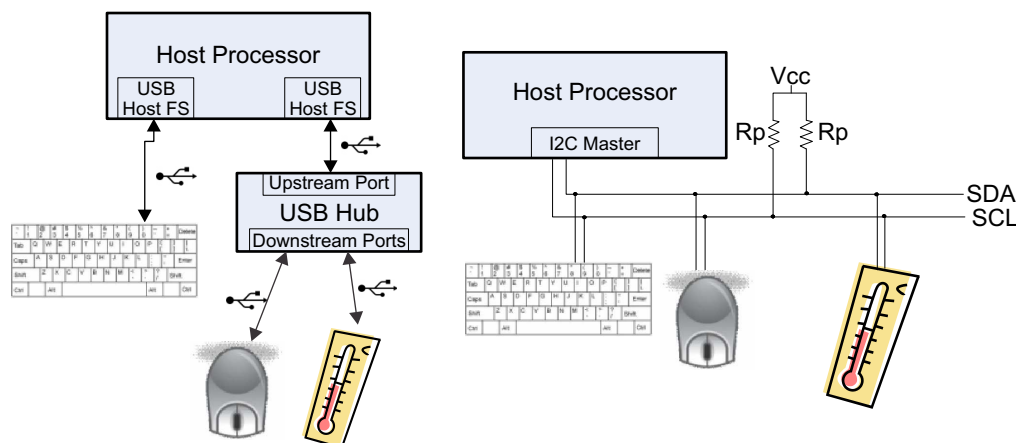

Figure 9. Block Diagram of USB and I2C HID Systems

USB implements a tiered-star topology requiring a single dedicated connection from a downstream port to an upstream port. This requires connecting a single device to the USB Host port, or using a USB hub to connect multiple devices.
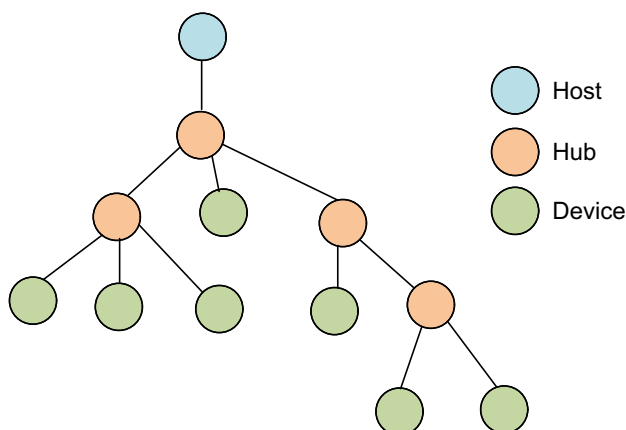

Figure 10. Tiered Star Topology

This is a proven topology that brings advantages like hot-plug detection, but the circuitry required for the USB connection must be implemented for each device. Additionally, the number of hubs, host ports, and device ports is directly proportional to the current consumption.

On the other hand, I2C basically implements a linear bus topology (although different topologies can be implemented if required) where multiple devices can act as host or slaves.
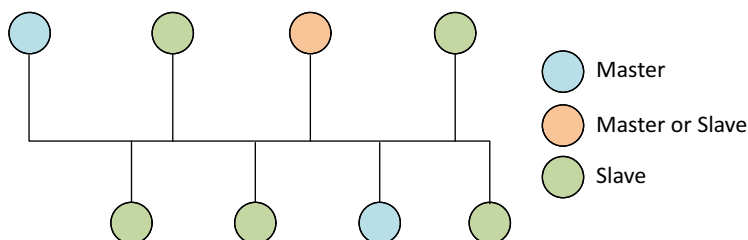
Figure 11. Linear Bus Topology

One important consideration of the I2C bus is that it uses only two bidirectional open drain lines so it requires a pair of pullup resistors (R_p) which set the idle state of the bus as high. Master and Slave devices can only drive the lines low (sending a logical low) or leave them open (sending a logical high).

When a device pulls the line low, the pullup resistor finds a path to ground and the current consumption increases. A worst case scenario happens when both lines are pulled low, which consumes a sink current of V_{CC}/R_P.

Because these resistors are usually in a range between 1 kΩ and 10 kΩ, the current consumption increase can be significant, but it is also important to note that a typical I2C bus is not low all of the time, and the resistor pair is shared by all devices in the bus. Table 3 shows measurements that were taken from a very active I2C bus with a Master device sending data continuously to a Slave device.

Table 3. Current Consumption of I2C Pullup Resistors

R_p: Baud Rate:	10 kΩ 100 kbps	4.7 kΩ 100 kbps	2 kΩ 100 kbps	1 kΩ 100 kbps	1 kΩ 400 kbps
I_{Rp} (V_{CC} = 3 V)	0.334 mA	0.643 mA	1.601 mA	3.122 mA	3.188 mA

The current is approximately:

$$I_{RP} = \left(\frac{V_{CC}}{R_P} \times 2 \right) \times 50\% = \frac{V_{CC}}{R_P}$$

(1)

This is because the I2C bus has two pullup resistors, but a typical I2C bus has a approximately 50% idle time. A clock line usually has equal high and low times, but a packet of data consisting mostly of logical lows consumes more power on the data line than a packet consisting mostly of logical highs. A more accurate scenario would consist on an even mix of 1 and 0, thus totaling approximately 50%.

I2C also allows for clock-stretching, which is forced when devices are not ready to process data. This can have a negative impact on the overall power consumption as seen in the previous measurements comparing 100 kbps and 400 kbps.

Even when no device is pulling the line low and the line is at a logical high, the pullups consume some power, but it is much lower because it mostly depends on the leakage current of the devices. Leakage current can be typically a few microamps, but in the case of the MSP430, each pin has a maximum leakage current of ±50 nA.

An important consideration is that contrary to USB, in which a direct connection and circuitry is required for each device, in I2C the bus is shared, so the current consumption attributed to the pullup resistors is for the whole bus, not for each device.

5   Implementing HID Over I2C Using MSP430F5229 and Its Distinct Advantages

As mentioned previously, HID over I2C is implemented in Windows 8 devices such as laptops, tablets, and cell phones that usually use application processors at a lower voltage of 1.8 V. Interfacing to a microcontroller with 3.3-V IOs require external level translators.

The Texas Instruments MSP430F522x and MSP430F521x devices, which are part of the MSP430 family of ultra-low power microcontrollers, support a main supply rail (1.8 V to 3.6 V) and a separate I/O supply rail (1.8 V ± 10%), eliminating the need for level translators and offering system-level advantages such as reduced system cost and increased flexibility. Featuring this new dual voltage rail capability, the F521x and F522x microcontrollers can operate as an ultralow-power coprocessor to higher power and higher performance devices such as TI's OMAP™ processor and Sitara platforms. For example, the new MSP430 microcontrollers can be used to offload functions such as sensor hub, keyboard control, battery and power management, capacitive touch, haptics, and proximity detection with lower power consumption compared to running these within an application processor. F521x and F522x also has fast wake-up time at 3.5 µs and as little as 1.4-µA power consumption in standby mode.

Figure 12 shows an implementation of an HID over I2C system using the MSP430F5229.
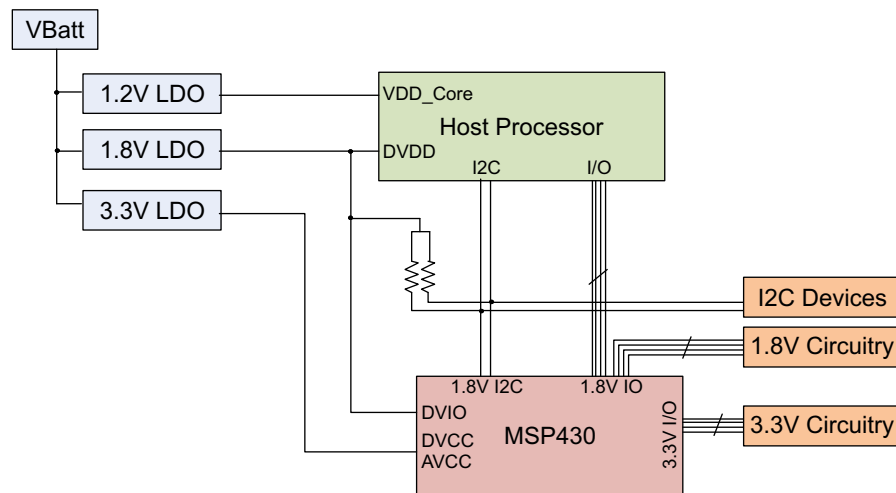
Figure 12. HID Over I2C System Using MSP430F5229

Figure 12 shows the MSP430F5229 in a dual supply configuration, which allows it to interface directly with 1.8-V devices while still being able to run at the maximum frequency and interface with 3.3-V circuitry such as other microcontrollers or sensors.

6 Summary

Tablets, ultrabooks, and laptops using the latest Windows 8 operating system can leverage the new HID over I2C protocol to conserve power in applications like sensor hub, keyboard controller and touchpad. Relative to the legacy HID over USB protocol, HID over I2C can offer up to 99% power savings for devices that attach to application processors.

With 1.8V and 3.3V split IOs, the new MSP430F522x MCU offers a glue-less interface to application processors, eliminating the need for level translators and offering system-level advantages.

TI also offers a fully functional and tested HID over I2C software stack that can run on MSP430F522x and many other MSP430s allowing device manufacturers and system integrators to easily develop devices that can interface to a Windows 8 system. Please contact TI for more information on this solution.

7 References

1. USB HID Specification (http://www.usb.org/developers/devclass_docs/HID1_11.pdf)
2. *MSP430x5xx and MSP430x6xx Family User's Guide* (SLAU208)
3. MSP430F550x, MSP430F5510 data sheet (SLAS645)
4. *USB Keyboard Using MSP430 Microcontrollers* (SLAA514)
5. AM35x Power Estimation Spreadsheet: http://processors.wiki.ti.com/index.php/AM35x_Power_Estimation_Tool
6. http://en.wikipedia.org/wiki/Human_interface_device
7. MSDN HID Over I2C Architectural Overview
8. Microsoft HID over I2C Specification
9. *Nine-Axis Sensor Fusion Using the Direction Cosine Matrix Algorithm on the MSP430F5xx Family* (SLAA518)

IMPORTANT NOTICE

Products		Applications	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		