USB Keyboard Using MSP430™ Microcontrollers

David Racine, Luis Reynoso   *MSP430 Apps*

ABSTRACT

This application report describes a low-cost highly-flexible composite USB keyboard implementation based on MSP430F5xx/MSP430F6xx families. Schematics and software are included allowing for an easy implementation and customization.

The document explains basic necessary concepts but familiarity with the MSP430™ USB Developers Package (MSP430USBDEVPACK) and USB HID specification is assumed.

Source code and additional information described in this application report can be downloaded from http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/USBKBD_430/latest/index_FDS.html.

Contents

List of Figures

List of Tables

MSP430, Code Composer Studio are trademarks of Texas Instruments.
IAR Embedded Workbench is a trademark of IAR Systems.

1 Introduction

This application report describes the implementation of a USB keyboard with the following characteristics:

- 101 keys, 2 LEDs: standard HID keyboard and LED usage
- 16x8 matrix: allows for easy customization of different keyboard layouts
- Composite USB device: In addition to the keyboard interface, it includes an HID-datapipe back-channel which can be used to transmit any custom data
- HID boot protocol support, allowing keyboard to be used to interface with a PC's BIOS
- "Ghost" key handling in software, to prevent errors from multiple key presses
- Uses MSP430F550x/5510 low-cost USB family

The Texas Instruments MSP430F550x/5510 devices are ultra-low power microcontrollers featuring a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. In addition, this MSP430 family includes an integrated USB and PHY supporting USB 2.0 full-speed communication, four 16-bit timers, a high-performance 10-bit analog-to-digital converter (ADC), two universal serial communication interfaces (USCI), hardware multiplier, DMA, real-time clock module with alarm capabilities, and 31 or 47 I/O pins.

2 Implementation

2.1 Key Matrix

The USB keyboard presented in this application report implements a key matrix of rows and columns similar to smaller keypads like the one shown in the application report Implementing *An Ultralow-Power Keypad Interface with MSP430* (SLAA139).

This implementation uses a 16 rows x 8 columns matrix, which allows up to 128 keys, but it actually uses only 101 keys in total.
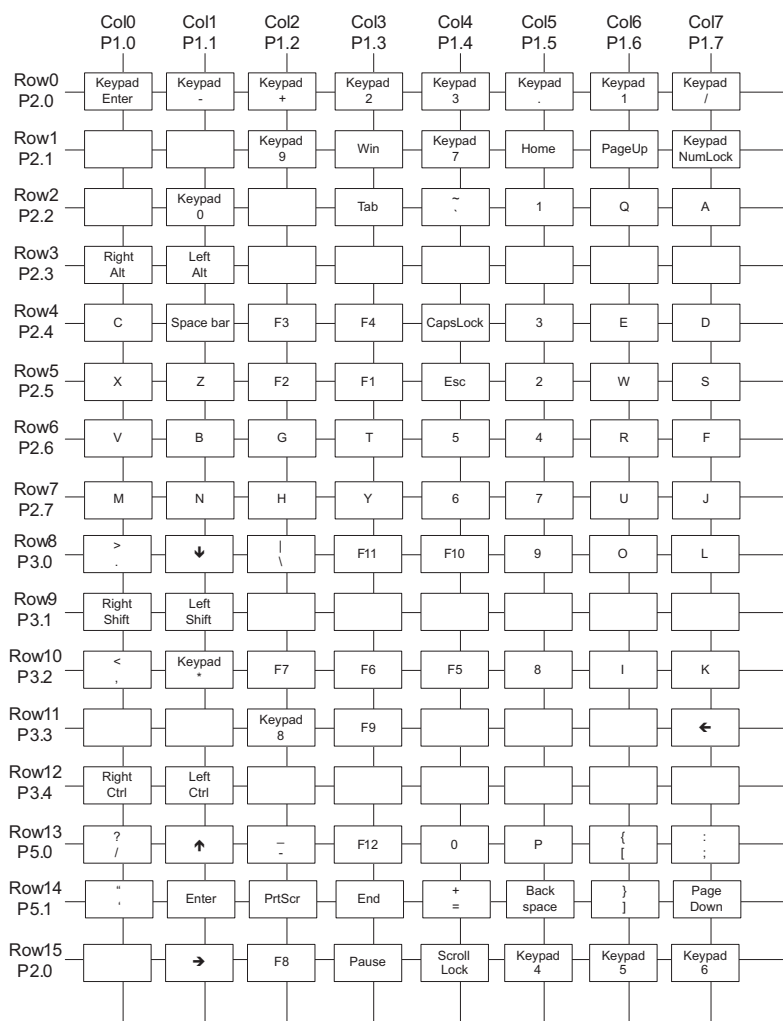
The key matrix is shown in Figure 1.

Figure 1. Key Matrix

Each key works like a switch, and pulldowns are implemented on each column, keeping the idle state low (see Figure 2).
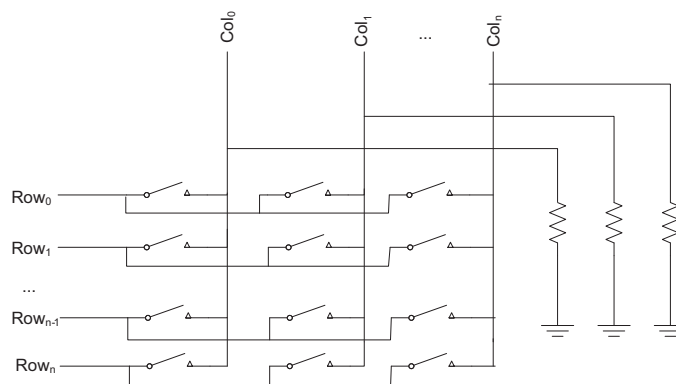

Figure 2. Keyboard Schematic

There are multiple ways to scan a key matrix, but this application report uses two methods, referred in this application report as: column-interrupt and polling.

In the column-interrupt approach, all rows are actively driven at the same time and columns are configured to interrupt the processor when any single key is pressed.

This method is useful in low-power modes, because any key can wake up the microcontroller; however, it is important to remark that the key press is only used for that purpose, because it does not provide the exact key being pressed.

Figure 3 shows the key matrix behavior when the Enter key is pressed in column-interrupt mode. Actively driven rows and columns are shown in red. Notice that the Col1 pin would detect a change when the Enter key is pressed, but the effect would be the same for any other pin pressed in the same column.
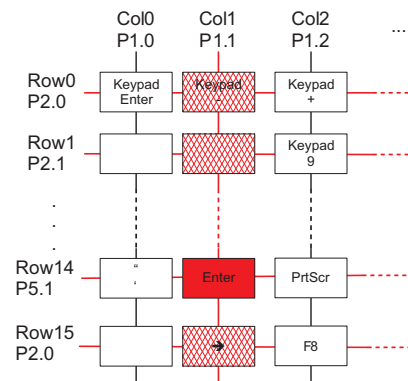
Figure 3. Detection of a Key Using Column-Interrupt Method

After the system is awake due to a key press using the column-interrupt approach, the polling method can be used to determine which key(s) is(are) being pressed (see Figure 4). In the polling method, each row is scanned separately driving one row at a time in sequential order. The columns are then read giving the exact keys being pressed.

Figure 4. Detection of a Key Using Polling Method

One of the caveats when using this method is that particular patterns can cause unwanted connections, known as "ghost" keys. This behavior is caused when three or more keys sharing rows and columns are pressed at the same time (see Figure 5).

Figure 5. "Ghost" Key Detection

The software included in this application report detects potential "ghost" keys and does not report them to the host.

2.2 USB HID
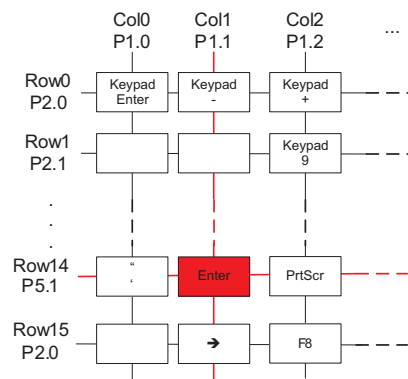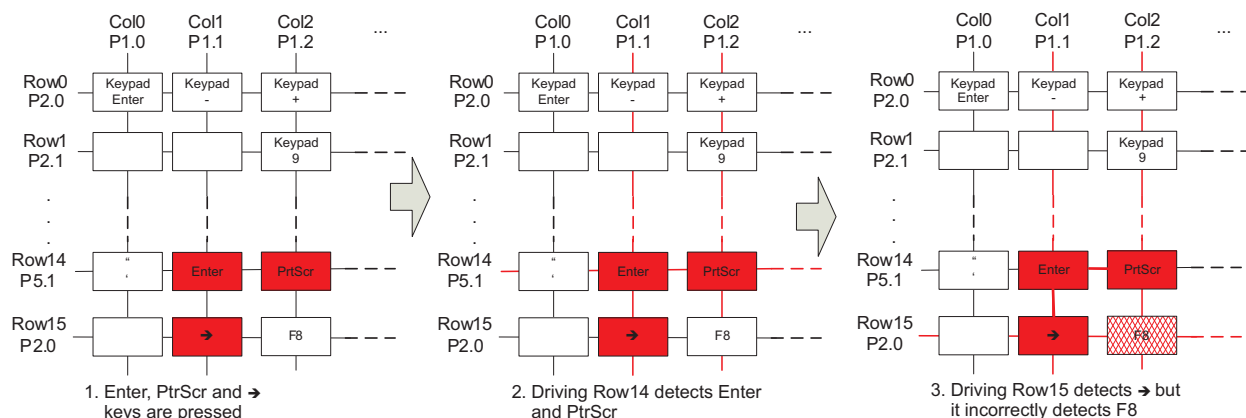
This application report uses the MSP430 application programming interface (API) stack found in the MSP430 USB Developers Package (MSP430USBDEVPACK).

The stack is configured to work as a composite HID-HID interface with the first interface being a standard Keyboard and the second interface used as a DataPipe. One of the advantages of using this implementation, which using only HID interfaces, is that no drivers are required.

Although the relevant code for the keyboard implementation uses the standard keyboard interface, the DataPipe interface was added to provide users with more flexibility and to facilitate customization.

This interface can be used to send or receive any type of data to/from the host, so that the MSP430 microcontroller not only performs the job of a digital keyboard, but it can also be used to perform other jobs taking advantage of the same USB interface and the rest of the peripherals. Some examples include reading sensors using ADC and reporting to PC, controlling actuators using timer PWMs, etc.

It should be noted that while the host OS interprets and uses the data from the standard keyboard interface without additional applications or drivers, in the case of the Datapipe interface, a host application is required. Texas Instruments provides an HID API which enables communication between a PC and a MSP430 microcontroller running the HID API stack. This HID API is available in executable format and source code in the MSP430 USB Developers Package (MSP430USBDEVPACK).

The keyboard interface supports Boot protocol, which allows it to work with HID-limited hosts (such as some BIOS).

VID and PID can be modified according to the particular application but the default code used for this example uses the values shown in Table 1.

Table 1. VID/PID Used by the Device

VID	0x2047
PID	0x0401

3   Software

3.1   Tools

The software included in this application report was built and tested using:

- IAR Embedded Workbench™ for MSP430 5.30.4 IDE
- Code Composer Studio™ (CCS) 5.1.0 IDE

3.2   *Software Implementation*

Figure 6 shows the software layers for the USB keyboard.
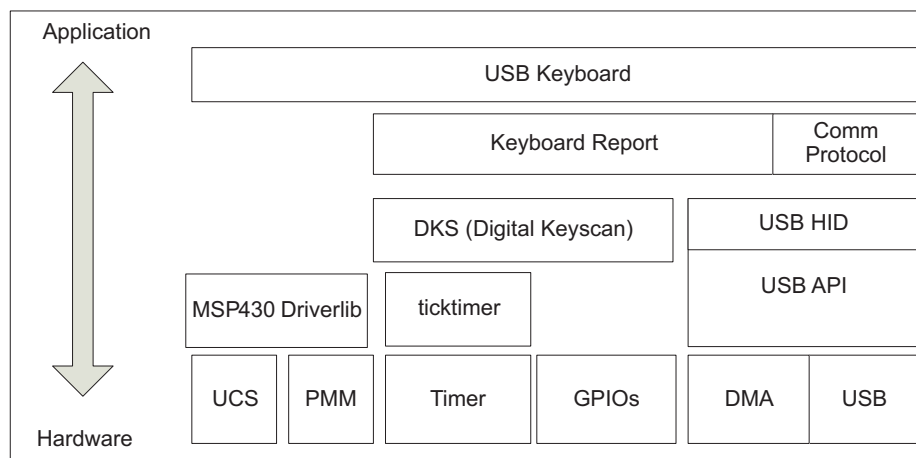
Figure 6. USB Keyboard Software Modules

Software is designed in a modular way, re-using existing TI libraries such as driverlib and the USB API and adding new modules from low-level drivers to application level. These modules include:

- **USB Keyboard**

    Description

    Main application initializing the microcontroller, peripherals, and executing a loop checking and servicing the rest of the modules.

    Files

    Src\TI_USBKBD_main.c

    Flow Diagram

Figure 7. USB Keyboard Flow Diagram

Copyright © 2011, Texas Instruments Incorporated

- **Keyboard Report**

  Description

  Handles the HID Keyboard report, adding and removing keys from the report depending on press/release events and sends the report to the USB Host.

  Files

  Src\TI_USBKBD_HIDKBD_report.c

  Src\Include\ TI_USBKBD_HIDKBD_report_public.h

  HID Keyboard Report Format

Table 2. HID Keyboard Report Format

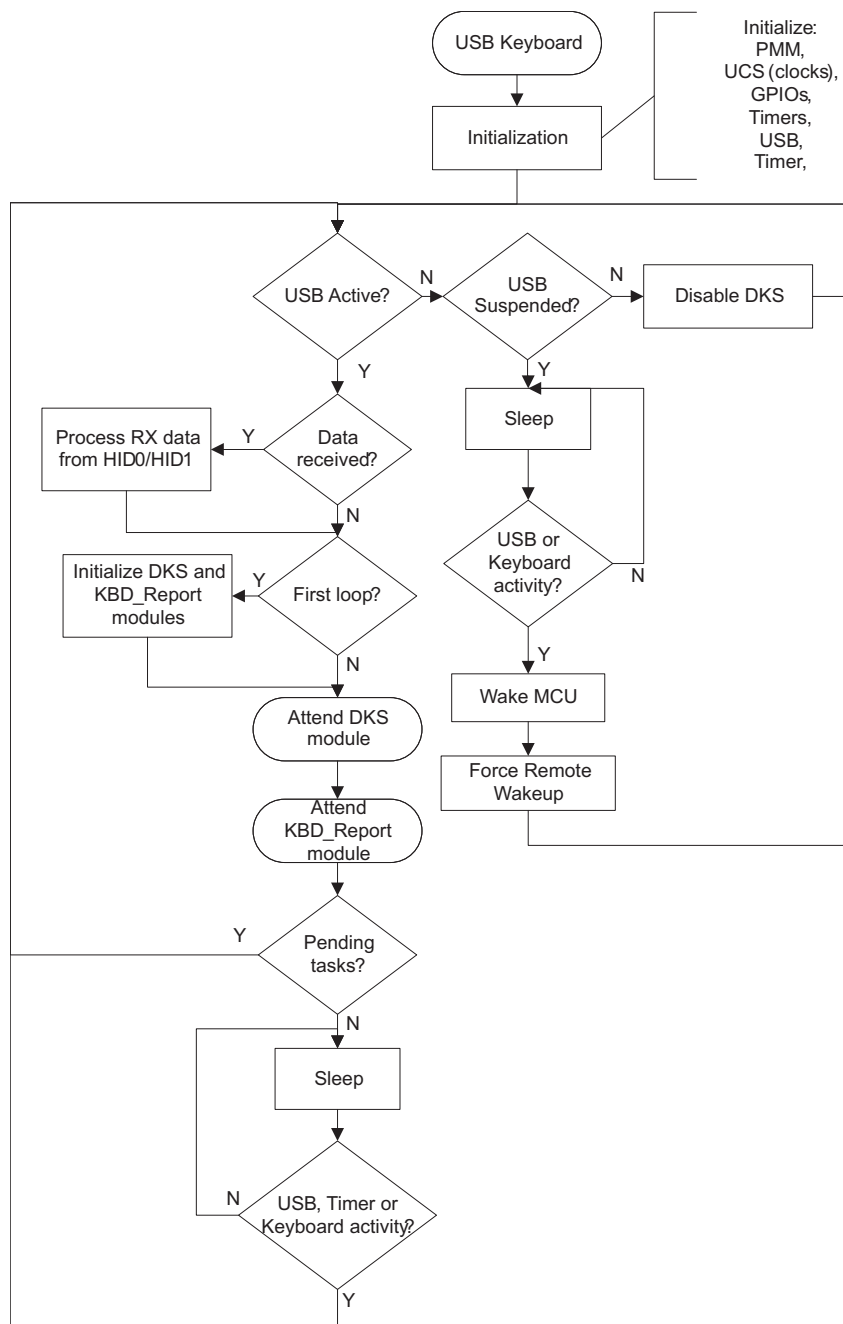|	**Bit7**	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
Byte0	Right GUI	Right Alt	Right Shift	Right Ctrl	Left GUI	Left Alt	Left Shift	Left Ctrl
Byte1	Reserved							
Byte2	Key_array[0]							
Byte3	Key_array[1]							
Byte4	Key_array[2]							
Byte5	Key_array[3]							
Byte6	Key_array[4]							
Byte7	Key_array[5]							

- **Communication Protocol**

  Description

  Handles the HID custom interface, which is used to transfer data to/from an USB host. The current implementation shows a template that can be used for custom development.

  This module uses the HID-Datapipe as defined in the USB API included in MSP430 USB Developers Package (MSP430USBDEVPACK).

  Files

  Src\TI_USBKBD_comm_protocol.c

  Src\Include\ TI_USBKBD_comm_protocol_public.h

  HID Custom Interface Report Descriptor

Table 3. Communication Protocol Report Descriptor

Field	Size	Description
IN Report		
Report ID	1 byte	The report ID of the chosen report (automatically assigned to 0x3F by the HID-Datapipe calls)
Size	1 byte	The number of valid bytes in the data field
Data	62 bytes	Data payload
OUT Report		
Report ID	1 byte	The report ID of the chosen report (must be assigned to 0x3F by the host)
Size	1 byte	The number of valid bytes in the data field
Data	62 bytes	Data payload

Data Payload Protocol

Table 4. Implemented Protocol

Field	Size	Description
CMD	1 byte	1 = Toggle CAPS LED 2 = Toggle NUM LED
Data	61 bytes	Unused

- **DKS (Digital KeyScan)**

  Description

  Handles the digital keyboard scanning, detecting key press/release events, and reporting them to the keyboard report module.

  Files

  Src\TI_USBKBD_DKS.c

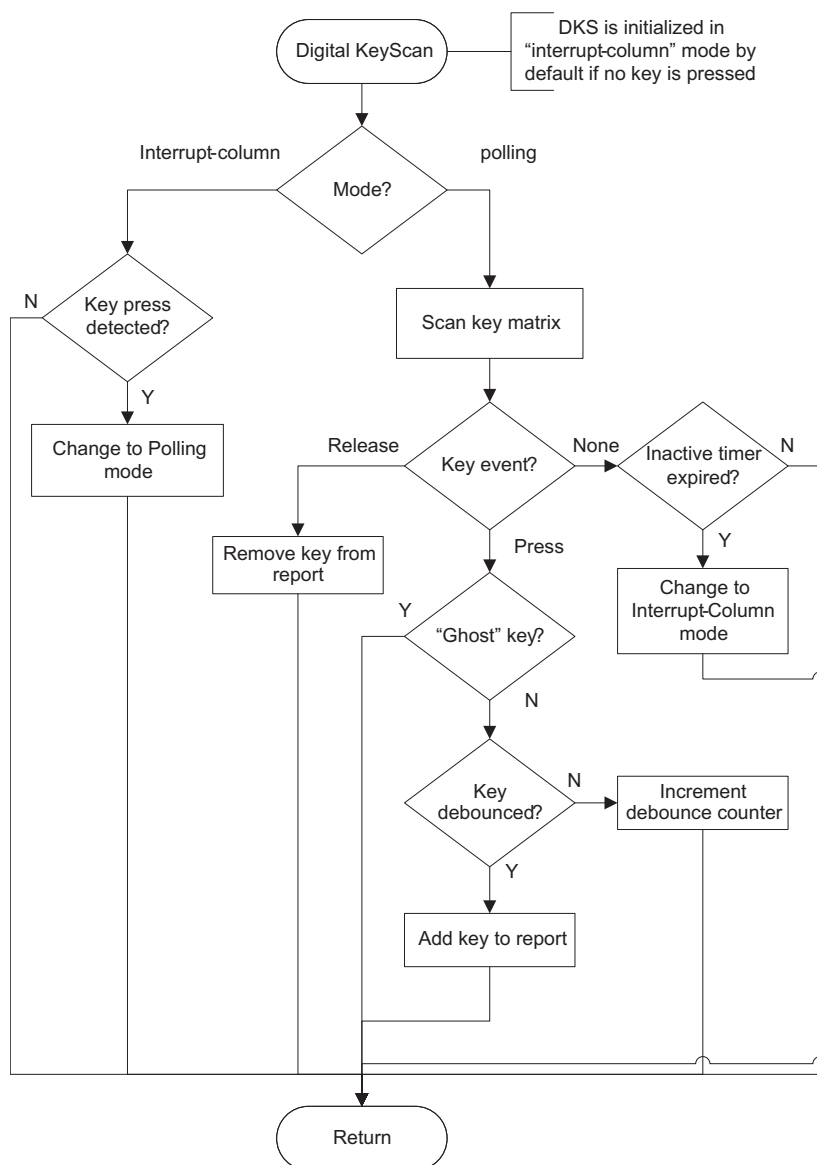 Src\Include\TI_USBKBD_DKS_public.h

  Flow Diagram

Figure 8. Digital Keyscan Flow Diagram

- **USB API / USB HID**

  Description

  The MSP430 USB API stack is a software solution provided by Texas Instruments that includes support for:
  - Communications Device Class (CDC)
  - Human Interface Device class (HID)
  - Mass Storage Class (MSC)
  - Personal HealthCare Device Class (PHDC)

  This software solution, including detailed documentation, is available in the MSP430 USB Developers Package (MSP430USBDEVPACK).

  Files

  Src\USB_API*.*

  Src\USB_config*.*

  Src\USB_App*.*

- **Ticktimer**

  Description

  Handles a general purpose interrupt timer that is used as a timebase, to wake-up the processor, and to trigger a new keyboard scan, among other functions.

  The ticktimer is implemented using TA0.0 with a default time base of 2 ms.

  Files

  Src\TI_USBKBD_ticktimer.c

  Src\Include\TI_USBKBD_ticktimer_public.h

- **MSP430 Driverlib**

  Description

  The Texas Instruments MSP430 Peripheral Driver Library (Driverlib) is a set of drivers that provide an easy mechanism to use the MSP430 peripherals. This software uses Driverlib to initialize the PMM and UCS modules.

  Source code and detailed documentation are available in MSP430Ware (www.ti.com/msp430ware). For simplicity purposes, this project includes only the pre-compiled libraries for IAR and CCS using a small memory model and header files.

  Files

  Src\ driverlib*.h

  Src\ driverlib\driverlib_small_CCS.lib

  Src\ driverlib\driverlib_small_IAR.r43

3.3 Configuration and ScanCode Tables

For modularity purposes and to allow for an easier optimization or upgrade, the USB keyboard software reserves some Flash sectors for constant tables that define some of the functionality of the application and define the ScanCode table.

- **Configuration Constant Table**

  Description

  Contains the USB keyboard version and configuration constants defining the KeyScan functionality, such as debounce counter, ticktimer period, etc.

  Files

  Src\TI_USBKBD_SharedTables.c (declaration)

  Src\Include\TI_USBKBD_public.h (typedef)

  Declaration

  ```
  const USBKBD_config_const_t USBKBD_configconst_s
  ```

Location

USBKBD_CONFIGCONST_SEGMENT (0xFC00-0xFDFF)

Contents

Table 5. Configuration Constant Table

Field	Size	Description
MagicKey	4 bytes	Indicates the start of the table. 0xDEADC0DE is used by default.
Version	2 bytes	USB keyboard version in BCD format: 0x0101 - 1.0.1
ticktimer_div	2 bytes	TickTimer divider (based on ACLK): 66 represents a period of 66 / 32768 = ~2 ms
debounce_cycles	2 bytes	Number of debounce cycles in tick counts: 2 represents a debounce of 4 ms with Ticktimer = 2 ms
inactive_timeout	2 bytes	Number of tick counts before going to interrupt_column mode if no key is detected: 8 represents 16 ms with Ticktimer = 2 ms

- **ScanCode Table**

  Description

  Contains the USB Keyboard scancode table, mapping each row and column to the corresponding value based on HID Usage Tables.

  Files

  Src\TI_USBKBD_SharedTables.c(declaration)

  Src\Include\TI_USBKBD_public.h(typedef)

  Declaration

  ```
  const USBKBD_scancodest_t USBKBD_scancodes_s
  ```

  Location

  USBKBD_SCANCODES_SEGMENT (0xFA00-0xFBFF)

  Contents

Table 6. ScanCodes

Field	Size	Description
MagicKey	4 bytes	Indicates the start of the table. 0xDEADC0DE is used by default.
keycode	128 bytes	Keycodes for each key in the following order: Row0,Col0 Row0,Col1 … Row0,Col7 Row1,Col0 Row1,Col1 … Row15,Col6 Row15,Col7

4   Hardware and Peripheral Usage

In addition to system modules (UCS, PMM), this keyboard implementation uses the peripherals shown in Table 7.

Table 7. MSP430F550x/5510 Peripheral Usage

Peripheral	Usage
USB	Communication with host (Composite HID-HID)
Timer_A0 (TA0.0)	TimerTick used as a time base to perform periodic polling, debounce, etc.

In addition to the circuitry required for USB and common functionality (reset, VCC, VSS, crystal, etc.), the USB keyboard uses the pins shown in Table 8.

Table 8. MSP430F550x/5510 Pinout Usage

Columns			Rows		
Columns	KSO0	P2.0	Rows	KSI0	P1.0
Columns	KSO1	P2.1	Rows	KSI1	P1.1
Columns	KSO2	P2.2	Rows	KSI2	P1.2
Columns	KSO3	P2.3	Rows	KSI3	P1.3
Columns	KSO4	P2.4	Rows	KSI4	P1.4
Columns	KSO5	P2.5	Rows	KSI5	P1.5
Columns	KSO6	P2.6	Rows	KSI6	P1.6
Columns	KSO7	P2.7	Rows	KSI7	P1.7
Columns	KSO8	P3.0	LEDs	LED0 (CAPS)	P4.7
Columns	KSO9	P3.1	LEDs	LED1 (NUM)	P4.6
Columns	KSO10	P3.2			
Columns	KSO11	P3.3			
Columns	KSO12	P3.4			
Columns	KSO13	P5.0			
Columns	KSO14	P5.1			
Columns	KSO15	P5.4			

Schematics showing the implementation on the USB keyboard are found in Section 6.

5   Using the USB Keyboard

When connected to a PC, the USB keyboard should be detected by the operating system and enumerated without drivers. Windows shows three devices in the Device Manager (see Figure 9).

- Human Interface Devices
    - USB Human Interface Device: Standard keyboard in intf0 (MI_00)
    - USB Human Interface Device: Custom interface in intf1 (MI_01)
- Keyboards
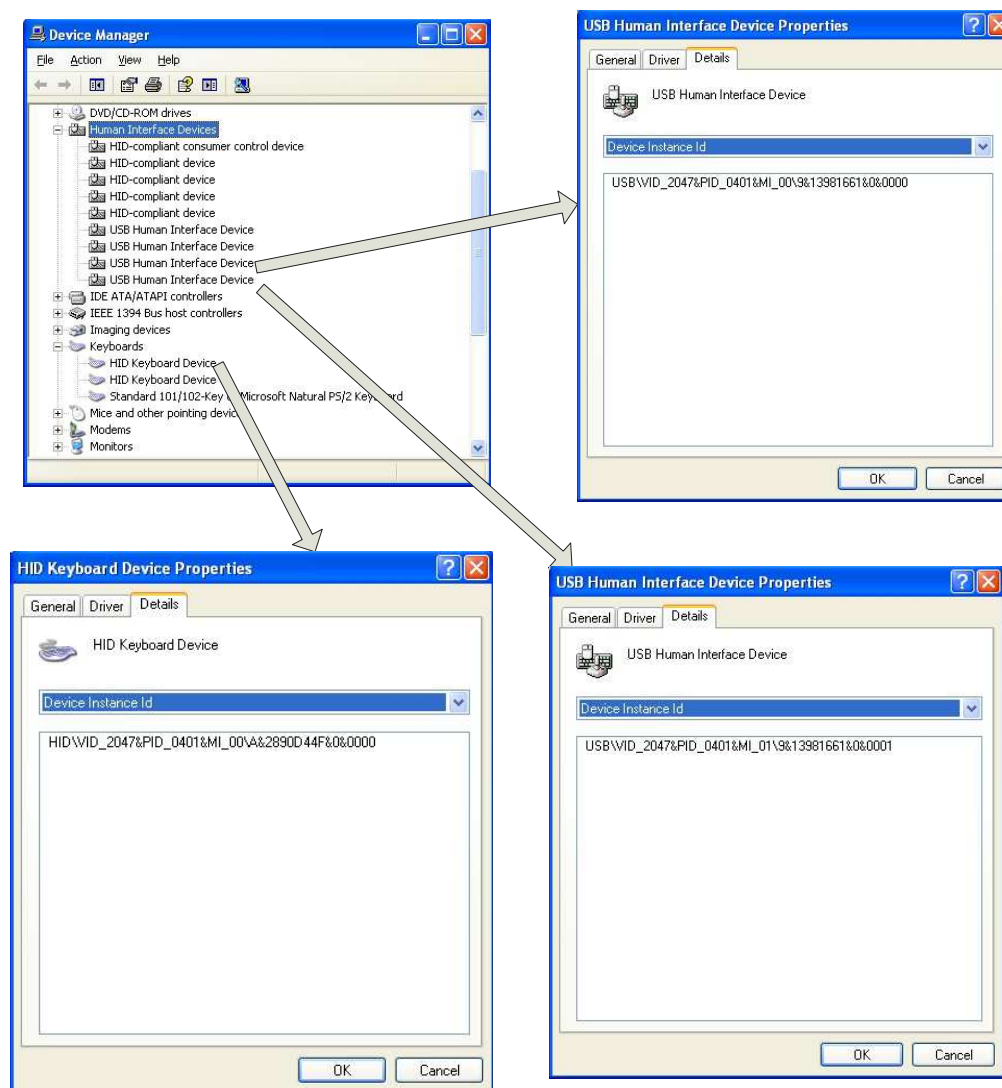   - HID Keyboard Device: Standard keyboard in intf0 (MI_00)

Figure 9. USB Keyboard in Windows Device Manager

The keyboard can now be tested and used as a standard keyboard.

In addition to the regular key functionality, the custom interface can be tested using the MSP430 HID USB Application following these steps (see Figure 10):

1. Select the VID and PID (default: VID = 0x2047, PID = 0x0401).
2. Click Set VID PID.
3. Click Connect.
4. The LED should turn green.
5. Write one of the commands in the Send & Receive field.
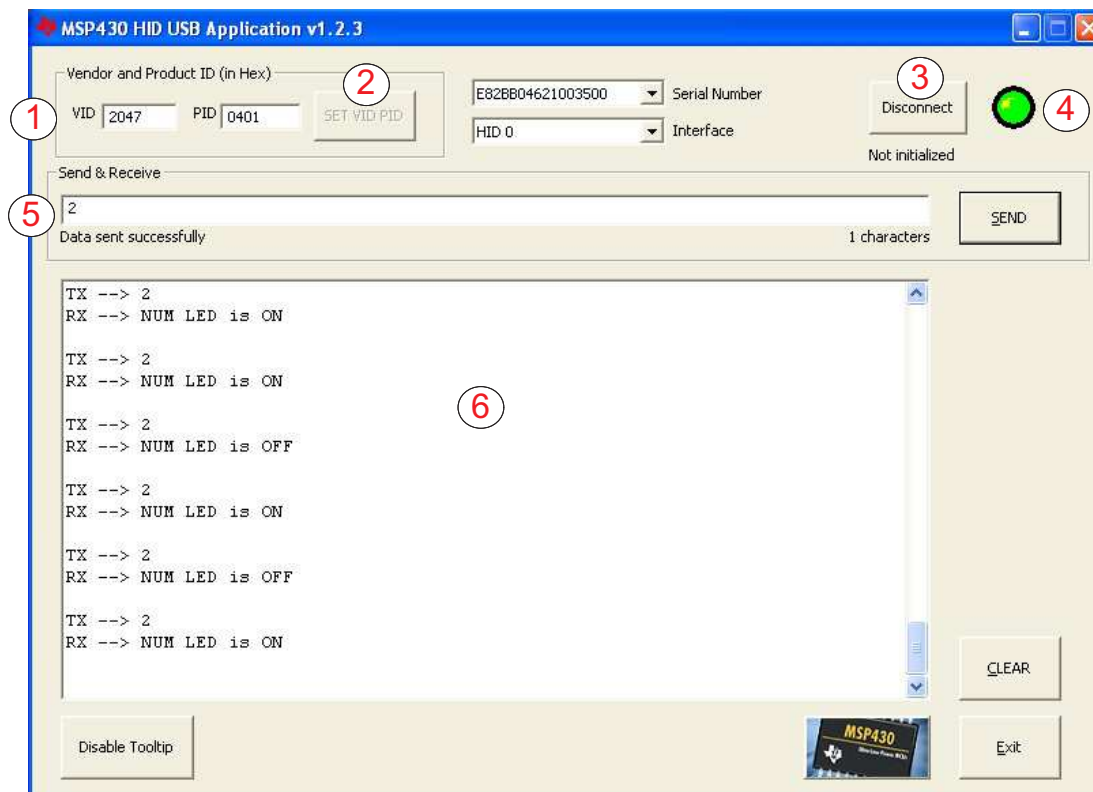6. Observe the response from the USB keyboard.

Figure 10. Testing the HID Custom Interface

The MSP430 HID USB Application is available in the MSP430 USB Developers Package (MSP430USBDEVPACK).

5.1 Performance

The usual response time for keyboards is approximately 5 to 50 ms. While this depends on different factors such as the mechanical implementation of the keyboard, USB bus load, etc., by using this software, developers have more flexibility to customize the application according to their needs. Whether response time, price, or power consumption is the most important requirement, parameters such as debounce time, polling scan rate, USB polling interval, and microcontroller internal frequency can be adjusted to meet particular requirements.

One important factor affecting the response time is the polling rate, which defines the time required to scan all keys. While a key press is detected in a few cycles in column-interrupt mode, the algorithm to recognize the particular pressed key, debounce it, discard "ghost" keys, etc. can take more cycles.

During bench tests, this implementation was measured to take ~1870 cycles (which is equivalent to ~233 µs at 8 MHz) for the first pressed key and ~520 cycles (~65 µs at 8 MHz) for each additional pressed key.

5.2 Memory Footprint

The following memory footprint was obtained using IAR Embedded Workbench 5.30.4 using the maximum optimization level:

Code: 7626 Bytes
Constants: 1096 Bytes
Data: 679 Bytes

6   Schematics

Figure 11. Schematics

7 References

1. USB HID specification (www.usb.org/developers/devclass_docs/HID1_11.pdf)
2. *MSP430x5xx/MSP430x6xx Family User's Guide* (SLAU208)
3. *MSP430F550x/MSP430F5510 Mixed Signal Microcontroller* data sheet (SLAS645)
4. MSP430 USB Developers Package (MSP430USBDEVPACK) (www.ti.com/tool/msp430usbdevpack).
5. MSP430Ware (www.ti.com/msp430ware)
6. *Implementing an Ultralow-Power Keypad Interface With the MSP430* (SLAA139)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

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 Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		

TI E2E Community Home Page   e2e.ti.com