AVR271: USB Keyboard Demonstration

Features
• Supported by Windows®98 or later, Linux and MAC OS
• No driver installation
• Display a simple text message
• Does not support keyboard LEDs management

1. Introduction
The PS/2 interface is disappearing from the new generation PCs being replaced by the USB interface, which has become the standard interface between the PCs and peripherals. This change must be followed by keyboard designers, who must integrate the USB interface to connect the keyboard to the PC.

The aim of this document is to describe how to start and implement a USB keyboard application using the STK525 starter kit and FLIP in-system programming software.

A familiarity with USB Software Library for AT90USBxxx Microcontrollers (Doc 7675, Included in the CD-ROM & Atmel website) and the HID specification (http://www.usb.org/developers/hidpage) is assumed.

Figure 1-1. PC to Keyboard Interface
2. Hardware Requirements

The USB keyboard application requires the following hardware:

1. AVR USB evaluation board (STK525, AT90USBKey, STK526...or your own board)
2. AVR USB microcontroller
3. USB cable (Standard A to Mini B)
4. PC running on Windows® (98SE, ME, 2000, XP, Vista), Linux® or MAC® OS with USB 1.1 or 2.0 host

3. In system programming and Device Firmware Upgrade

To program the device you can use the following methods:

- The JTAG interface using the JTAGICE MKII
- The SPI interface using the AVRISP MKII
- The USB interface thanks to the factory DFU bootloader and FLIP software
- The parallel programming using the STK500 or the STK600

Please refer to the hardware user guide of the board you are using (if you are using Atmel starter kit) to see how to program the device using these different methods.

Please refer to FLIP(1) help content to see how to install the USB driver and program the device through the USB interface.

Note: 1. FLIP is a software provided by atmel to allow the user to program the AVR USB devices through the USB interface (No external hardware required) thanks to the factory DFU bootloader.

4. Quick start

Once your device is programmed with usb_keyboard.a90 file, you can start the keyboard demonstration. Check that your device is enumerated as keyboard (see Figure 4-1), then you can use the kit to send characters to the PC.
Figure 4-1. Keyboard enumeration

The figure below shows the STK525 used by the demo (you may use another kit: AT90USBKey, STK526, depending on the AVR USB product you are working with):
The purpose of the keyboard demonstration is to send a character string to the PC.
Follow the instructions below to start the demo:

1. Open the Notepad application or any text editor.
2. Set your keyboard to QWERTY configuration (Otherwise, you’ll see the wrong characters on your text editor).
3. Connect the STK525.
4. Push the joystick button.
Welcome to HID Keyboard demo, and congratulations for selecting the AT90USBxxx microcontroller family.
The AT90USBxxx is based on the AVR core with the following features:
- USB OTG (low/full speed controller with 7 endpoints)
- Up to 128 Kbytes on-chip flash memory
- USB bootloader
- UART, SPI, TWI
  and more...
  enjoy your time with ATMEL Products!!!
## 5. Application overview

The USB Keyboard application is a simple data exchange between the PC and the keyboard. The PC asks the keyboard if there is new data available each \( P \) time (polling interval time), the keyboard will send the data if it is available, otherwise, it will send a NAK (No Acknowledge) to tell the PC that there is no data available.

The data exchanges between the PC and the keyboard are called reports. The report which contains the keys pressed is the report IN (Keyboard to PC). The report which contains the LEDs status (NUM LOCK, CAPS LOCK, SCROLL LOCK...) is the report OUT (PC to Keyboard). The figure below shows the structure of these reports:

**Figure 5-1. USB Report Structure**

<table>
<thead>
<tr>
<th>Report IN</th>
<th>6 Bytes</th>
<th>1 Byte</th>
<th>1 Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keycode 1..6</td>
<td>Reserved for DEM</td>
<td>Modifier keys</td>
<td></td>
</tr>
</tbody>
</table>

**Report OUT**

Leds management

---

Note: This demonstration manages the report IN only.
6. **Firmware**

As explained in the *USB Software Library for AT90USBxxx Microcontrollers* (Doc 7675), all USB firmware packages are based on the same architecture (please refer to this document for more details).
This section is dedicated to the keyboard module only. The customization of the files described hereafter allow the user to build his own keyboard Application.
6.1 keyboard_task.c

This file contains the functions to initialize the hardware which will be used as a keyboard, collect the report data and put it in the endpoint FIFO to be ready to be sent to the PC.

**Figure 6-2. Keyboard Application**

6.1.1 keyboard_task_init

This function performs the initialization of the keyboard parameters and hardware resources (joystick...).

6.1.2 kbd_test_hit

This function checks if there is a key pressed and sets the `key_hit` variable to true.

6.1.3 keyboard_task

This function checks if any key is pressed (`key_hit == true`). If it is the case, the report IN is filled out with the related values and loaded in the USB endpoint FIFO to be transmitted to the host.

6.2 stk_52x.c

This file contains all the routines to manage the STK52x board resources (Joystick, potentiometer, Temperature sensor, LEDs...). The user should not modify this file when using the STK52x board. Otherwise he has to build his own hardware management file.

6.3 How to manage the CAPS, NUMLOCK... LEDs

The keyboard LEDs (CAPS, NUMLOCK...) are managed by the host when the corresponding key is pressed. When receiving the keycode of CAPS or NUMLOCK... the host sends a Set_Report request (Out Report) to turn on/off the related LED of the keyboard.

This request is send through the endpoint 0 (control transfer) and has to be managed as a Set_Configuration request, as shown below:

First the host will send the set_report as showing below:
This request is specific to the HID class, this is why it is not managed by the `usb_standard_request.c` file but with the `usb_specific_request.c`. In this file the request is decoded following the value of the `bmRequest` and the `bRequest` using the `usb_user_read_request()` function. The report type (0x02) corresponds to an Out Report. To handle this request the `usb_user_read_request()` will call the `hid_set_report()` function. This function will acknowledge the setup request and than allow the user to get the one byte data (you can check the size using the `wLength` parameter) to know which LED has to be turned on/off (please refer to the HID specification for further information regarding the LEDs usage values).

```c
void hid_set_report (void)
{
    U16 wLength;
    U8 CAPS_LED = 0;
    U8 REPORT_ID;

    LSB(wInterface)=Usb_read_byte();
    MSB(wInterface)=Usb_read_byte();

    LSB(wLength) = Usb_read_byte();      //!< read wLength
    MSB(wLength) = Usb_read_byte();
    Usb_ack_receive_setup();

    while(!Is_usb_receive_out());
    REPORT_ID = Usb_read_byte();
    CAPS_LED = Usb_read_byte();// get the value of the CAPS LED status sent by the host
    Usb_ack_receive_out();
    Usb_send_control_in();
    while(!Is_usb_in_ready());
    //Send a report to clear the CAPS request
    Usb_select_endpoint(EP_KBD_IN);
    Usb_write_byte(0);// Byte0: Modifier
    Usb_write_byte(0); // Byte1: Reserved
    Usb_write_byte(0); // Byte2: Keycode 0
    Usb_write_byte(0); // Byte2: Keycode 1
    Usb_write_byte(0); // Byte2: Keycode 2
    Usb_write_byte(0); // Byte2: Keycode 3
    Usb_write_byte(0); // Byte2: Keycode 4
    Usb_write_byte(0); // Byte2: Keycode 5
```
Usb_ack_in_ready();

// Turn ON/OFF the LED0 following the host request
if (CAPS_LED == 0)
    Led3_off();
else
    Led3_on();

6.4 How to modify my device from non-bootable to bootable device

Please note that HID device may be bootable or non-bootable. By default, the HID demo provided by Atmel are non-bootable device. If your application need to be bootable, you have to modify the sub-class parameter (usb_descriptors.h):

    // USB Interface descriptor Keyboard
    #define INTERFACE_NB_KEYBOARD 0
    #define ALTERNATE_KEYBOARD 0
    #define NB_ENDPOINT_KEYBOARD 1
    #define INTERFACE_CLASS_KEYBOARD 0x03    // HID Class
    #define INTERFACE_SUB_CLASS_KEYBOARD 0x00 // Non-bootable
    #define INTERFACE_PROTOCOL_KEYBOARD 0x01 //Keyboard
    #define INTERFACE_INDEX_KEYBOARD 0

Set the INTERFACE_SUB_CLASS_KEYBOARD to 1 to convert the keyboard to a bootable device.

7. PC Software

The USB keyboard application doesn’t require any PC software. Limitations

The demonstration does not manage the OUT report. You have to add the required code to handle this feature (refer to the section 6.3 for further details)

8. Related Documents

- AVR USB Datasheet (the related to the part number you are using)
- USB Software Library for AT90USBxxx Microcontrollers (Doc 7675)
- USB HID class specification