Tags arduino, hid, keyboard
Edited Mar 22, 2011 7:38 PM by darran...

Arduino UNO Keyboard HID part 2

Part 1 covered getting the avr tools setup and building the Arduino usbserial firmware. Part 2 covers flashing the firmware to the UNO's atmega8u2 and putting together firmware for a Keyboard HID driver.

Updating the UNO's atmega8u2 USB firmware

To do this the atmega8u2 needs to be put into DFU mode. There is a description of how to do this without soldering on the Arduino forums here: msg374201. Note that you can't brick your UNO doing this, it will always support DFU mode so you can recover by flashing it with a working Arduino-usbserial.hex file.

- Put the UNO into DFU mode.
- dfu-prgrammer at90usb82 erase
- dfu-programmer at90usb82 flash --debug 1 Arduino-usbserial.hex
- dfu-programmer at90usb82 reset
- Unplug the UNO's USB cable for a few seconds and plug it back in
- Check that you can still upload a sketch. If you can't upload the sketch (e.g. the usb device hasn't shown up), then put the UNO back into DFU mode and repeat the above procedure using the prebuilt Arduino-usbserial-uno.hex file that is in the Arduino-usbserial directory.

Note you can also flash the firmware by running "make dfu" in the arduino-usbserial directory after you put the UNO into DFU mode.

Creating Keyboard HID firmware

This has been made extremely easy by the excellent work done by Dean Camera with his AVR USB stack LUFA and the excellent demo's he provides which are full drivers for a large collection of USB devices. And, of course, the efforts of the Arduino team and contributors that have made it such an easy platform to develop on.

I put together the Keyboard HID firmware by starting with the Arduino-usbserial source code, adding in the LUFA keyboard HID demo code, and implementing a simple serial protocol to allow communication with the keyboard firmware from the UNO's main processor.

The source code is in the following locations:

- Arduino/hardware/arduino/firmwares/arduino-usbserial
- LUFA 100807/Demos/Device/ClassDriver/Keyboard

I've defined a very simple serial protocol between the keyboard firmware and the UNO's main processor. The keyboard firmware expects to receive 8 bytes formatted as a Keyboard HID report. The format is as follows:

	Bit 0 - Left CTRL Bit 1 - Left SHIFT Bit 2 - Left ALT Bit 3 - Left GUI Bit 4 - Right CTRL Bit 5 - Right SHIFT Bit 6 - Right ALT Bit 7 - Right GUI
1	Not used
2 - 7	HID active key usage codes. This represents up to 6 keys currently being pressed.

The Key Usage codes are documented in chapter 10 of the HID Usage Tables. The letters 'a' to 'z' are codes 4 to 29, and you can indicate upper case by setting the left SHIFT or right SHIFT bit in byte 0 of the report.

The keyboard firmware sends back 1 byte every time it receives 8 bytes. This byte represents the LED status for the keyboard: Bit 0 - NUMLOCK, Bit 1 - CAPSLOCK, Bit 2 SCROLL LOCK. Note this isn't currently working and I haven't worked out why yet. I'll update the page when its fixed.

Example code to send an 'a':

```
uint8_t keyNone[8] = { 0, 0, 0, 0, 0, 0, 0 };
uint8_t keyA[8] = { 0, 0, 4, 0, 0, 0, 0 };

void setup()
{

    Serial.begin(9600);
    delay(2000);

}

void loop()
{

    uint8_t ledStatus;


    /* Send an 'a' every second */

    Serial.write(keyA, 8);

    ledStatus = Serial.read();
    delay(100); // Give the host time to read the key
```

```
    }
```

There is plenty of room for improvement in the keyboard firmware. I'll work on a version that doesn't need the delay(100) to give the host time to read the key. There is some difficulty because of the size of the firmware - its 4040 bytes which is almost at the 4096 byte limit which makes it challenging to add features.

You can download the Keyboard HID firmware source here: arduino-keyboard.tar.gz and the firmware here: Arduino-keyboard.hex