Introduction

After years of using computers, I started to question the fact that I had to remove my hands from the keyboard in order to operate my standard mouse on the mouse pad. I also questioned my hand position because I am prone to carpal tunnel syndrome. I will develop a new kind of input device that solves both these issues.

I have a plan to consolidate the keyboard and the functions of a mouse into one device that I can use continually without removing my hands. In this document I will refer to this device by the name KeyStick.

The KeyStick consolidates the functions of a keyboard and mouse while occupying roughly the same amount of desktop space. The KeyStick is not designed to replace the existing keyboard and mouse, but will be a concurrent input option.

The user will be able to connect the KeyStick to any USB enabled computer, and after installing the drivers from the included CDROM, use the KeyStick as the only input device.

The user will also have the option to display a virtual keyboard, similar to the “On-Screen Keyboard” that is included in the accessibility package of Windows XP professional edition. This virtual keyboard will graphically display what a key press on the Key Sticks will produce when pressed from any Key Stick position.
Description and Sketch

The KeyStick Project (Fig. 1) is classified as an HID (Human Input Device). The project has three main sections: Input, Output, and USB connectivity.

KeyStick Project

**INPUT**

- Hand Held Key Sticks
- USB Enabled Microcontroller

**OUTPUT**

- USB Enabled Host Computer
- Graphical User’s Interface

Figure 1: KeyStick Project Diagram
The input section is the physical device that a user will simply refer to as the KeyStick. It consists of the two Key Sticks and the base. The two Key Sticks (Fig. 1) are located on the base of the KeyStick, and are held in a user's hand just like he/she was holding a joystick. The Key Sticks are to be no greater than 3” by 2” by 7”. The optimal dimensions will be discovered during the prototyping phase. The base of the KeyStick (Fig. 1) is 14” long, 8” wide, and 2” tall. The base has a USB jack in the back.

The output section includes the virtual keyboard, and the computer’s response to the KeyStick's input. When the user presses a key he/she expects something to happen, whether it is typing a letter in a document editor, or closing a window.

To the user, the USB connectivity section is simply the USB cable that connects from the back of the base to an open USB port on the host computer, or a hub connected to the host computer.

**Detailed Functional Description**

![Figure 2: Left Hand Key Stick (only 2 of eight supports shown)](image)

![Figure 3: Key Stick Operation](image)
The base will house the circuit board and have a USB type “A” receiving port centered on the back. There will be no power cord to the KeyStick, as the USB’s 500mA at 5v provides sufficient power for this product.

**Hardware**

The communication between the KeyStick and the computer will be implemented as wired USB. The USB communication choice has a strong role in almost every aspect of this project (Fig. 1).

The KeyStick will be controlled by the Cypress EZ-USB AN2131QC microcontroller. I chose this chip for five reasons: it has an EZ-USB core, 24 general purpose IO channels, 8k of RAM, and finally the development support for this chip is inexpensive and extensive.

A good analogy is that the EZ-USB core is to the USB as a UART is to serial communication. The core provides a layer of abstraction from the timing issues of USB communication as well as handling enumeration with the host computer.

The 24 gpio channels come out of reset initialized for input. The KeyStick will only be using 16 of these channels. The other 8 channels will be ignored.

The 8k of RAM is also upgradeable, but I will not be upgrading for this project. This RAM will hold the 8051 code as well as the data. The code will be downloaded from the host computer every time the KeyStick is plugged into the USB port. Downloading every time removes the need for flashing EEPROM. I have purchased a development board from www.devasys.com that has the
AN2131QC as its microcontroller. My only debugging mechanism will be watching the communication packets to and from the KeyStick. I cannot afford to purchase a USB analyzer which would be my first choice on a project like this.

The base and Key Sticks will be made out of wood, old keyboard keys, push buttons, wood screws, electrical wire, and springs.

Figure 4 is a Functional Hardware Diagram.
The KeyStick uses the AN2131QC as its microcontroller (Fig. 4). The KeyStick powers the AN2131QC with the USB +5 line regulated to 3.3V. The KeyStick uses 16 GPIO channels (PortA and PortB) for simple logic level detectors. The µC needs a 12Mhz oscillation on the XTAL input which is provided by an external component. Internal circuitry then multiplies the 12Mhz to the 24Mhz that the CPU runs at. The RESET channel will be tied low which will prevent resetting the µC. KeyStick will use most of the 8K of RAM for program code and data.

Software Requirements

There are two different software requirements for this project. The first requirement is the firmware that will run on the EZ-USB chip. The second requirement is the program running on the host computer that communicates with the KeyStick through a WindowsXP HID driver.

The firmware will be written in C and assembler for the 8051 microcontroller. The 8051 code will be responsible for de-bouncing the key states as well as the Key Stick positions and putting this information in the USB endpoint buffers so the host can gather the state of the KeyStick.

The software running on the host computer will be written in Visual Basic from Visual Studio Version 6.0. I choose to write it in Visual Basic because of the fast development, great readability of the code, and the extensive documentation on my particular application already in existence on the internet. This software
will be responsible for opening up communication with the USB and generating appropriate keystrokes and mouse movements from the state of the KeyStick.

The firmware modules are as follows:

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KERNAL</td>
<td>Will be a time slice scheduler based on the full speed 1ms interrupt generated by the host computer.</td>
</tr>
<tr>
<td>MAIN</td>
<td>Handle data gathering and conditioning before filling the endpoint buffers with the KeyStick status bytes. Data conditioning includes de-bouncing all the switches.</td>
</tr>
</tbody>
</table>

The host computer modules are as follows:

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETUP</td>
<td>Open connection with windows driver. Discover if the device is attached. Setup driver to poll USB every 1ms for new key state.</td>
</tr>
<tr>
<td>MAIN</td>
<td>Receive current key states and logically discover what the user wants to happen by referring to input key matrix in memory and loading keystrokes and mouse intelligence into the windows message queues.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>Displaying the On-Screen keyboard for learning as well as a portal for manipulating the KeyStick settings.</td>
</tr>
</tbody>
</table>

**User Interface Requirements**

The left and the right Key Sticks are constructed, and operate in the same way. Each Key Stick (Fig. 2) is suspended about a ¼ inch above the base on eight screw heads that are 45 degrees apart, and around 1.5” from the center of the Key Stick (Fig. 3). There are 4 keys on each Key Stick. The letter shown (Fig. 2) is what the keys represent if the Key Stick is in the neutral position. The
keys will change what character they represent by tilting the Key Stick in a direction that causes 2 of the screws to be contacting while the 2 screws on the opposite side are not touching. For example, if he/she tilts the Key Stick forward the 2 screws that are next to the forward face of the Key Stick will still be making contact, but the back ones will not. The other screws are not important. Each pair of screws on the Key Stick will be the logic pull up point on an input channel. In the neutral position there are 4 channels pulled low. In a tilted position, 1 channel will be pulled high and at least one channel will be pulled low. The keys on the Key Stick are also connected to the logic point of a pulled up channel in a similar fashion. Table 1 shows the matrix of position and keystrokes necessary to simulate a 101 keyboard.

Each Key Stick can be in one of 9 states, neutral and pointing in 1 of the main compass directions. The shift key is made by the opposite hand that the letter is going to be pressed with. When both KeyStick keys are pressed at the same time the KeyStick goes into Mouse Mode. The direction that the Key Sticks are pointing will determine the mouse movements directions. In Mouse Mode, the top 2 keys operate just like the 2 mouse buttons. The bottom 2 keys will provide acceleration to the cursor movement in the direction that is Key Stick is currently pointing to.

<table>
<thead>
<tr>
<th>LEFT HAND</th>
<th>4th Finger</th>
<th>3rd Finger</th>
<th>2nd Finger</th>
<th>1st Finger</th>
<th>1st Finger</th>
<th>2nd Finger</th>
<th>3rd Finger</th>
<th>4th Finger</th>
<th>RIGHT HAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP</td>
<td>q</td>
<td>w</td>
<td>e</td>
<td>R</td>
<td>u</td>
<td>i</td>
<td>o</td>
<td>p</td>
<td>UP</td>
</tr>
<tr>
<td>NEUTRAL</td>
<td>a</td>
<td>s</td>
<td>d</td>
<td>F</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>;</td>
<td>NEUTRAL</td>
</tr>
<tr>
<td>DOWN</td>
<td>z</td>
<td>x</td>
<td>c</td>
<td>V</td>
<td>m</td>
<td>,</td>
<td>.</td>
<td>/</td>
<td>DOWN</td>
</tr>
<tr>
<td>UP+RIGHT</td>
<td>[</td>
<td>WIN</td>
<td>KeyStick</td>
<td>T</td>
<td>y</td>
<td>KeyStick</td>
<td>MENU</td>
<td>j</td>
<td>UP+LEFT</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Ctrl</td>
<td>Shift</td>
<td>ALT</td>
<td>G</td>
<td>h</td>
<td>ALT</td>
<td>Shift</td>
<td>Ctrl</td>
<td>LEFT</td>
</tr>
</tbody>
</table>
I will write a program that teaches how to use the KeyStick. This program will look like the On Screen Virtual Keyboard as mentioned in the Introduction.

This program will show the current virtual keys that the keys on the Key Stick are able to simulate. Figure 5 shows what the program will look like when the Key Sticks are in the neutral position.

Figure 6 shows what the program will look like when the right Key Stick is tilted forward. Notice that the red (for right) is highlighting the letters that are higher up from the neutral position. This will aid in the learning curve of using the KeyStick.
By combining: Figure 5, Figure 6, and the Key Stick matrix table (table 1), it is pretty clear what the On-Screen Keyboard will look like in any particular combination.

**Development Plan**

I will be developing the KeyStick mainly in room 338 at my own workstation. My only requirements for development equipment is a computer with a USB port, Visual Basic installed running Windows98SE or greater, and a multimeter. I have all the wood working equipment at home that I will use. My first step in developing will be to get logical data from my development board. Verifying communication will entail writing a very simple windows interface as well as the firmware for the microcontroller.

Specifically the C will be compiled using SDCC (small device c compiler). This compiler is open source and has been optimized for 8 bit microcontrollers such as the 8051. The assembly code will be assembled using ASEM-51, a two
pass macro assembler. I will be using M-IDE MCS-51 studio from www.opcode.com.

This will be the true test of when this project will be completed. Assuming that I have logical communication between my Windows application and the development board by the start of winter quarter my two quarter schedule will look something like this:

**Winter Quarter 2005:**

Week 1: Produce a reliable prototype physical input device out of wood, springs, screws and wires.

Week 2: Write KERNAL module for μC.

Week 3: Write MAIN module for μC.

Week 4: Write MAIN module for μC.

Week 5: Write MAIN module for host PC.

Week 6: Write MAIN module for host PC.

Week 7: Test thoroughly.

Week 7: Test thoroughly.

Week 8: Cushion for the end of the quarter.

Week 9: Cushion for the end of the quarter.
Spring Quarter 2005:

Week 1: Develop the DISPLAY module.
Week 2: Develop the DISPLAY module.
Week 3: Test thoroughly, In-class hardware review.
Week 4: Test thoroughly.
Week 5: In-class software systems presentation.
Week 6: Hardware documentation due.
Week 7: Cushion for year.
Week 8: Cushion for year.
Week 9: In-class code review.
Week 10: In-class final presentation.

Electrical Specifications

Range ------------------------------> 16’ connected to host or 96’ with 5 hubs
Power Source -----------------------> USB provides up to 500mA @ +5V
Worst case power----------------------> 60.03mA @ +3.3V = 198mW
Operating Temperature Range---------> -10°C to 50°C
PCB Limits ----------------------------> 4”x 4”x1”

Standard Specifications

User Interface------------------------> 101 keys emulation, 2 button mouse emulation.
USB connectivity----------------------> Full-Speed, HID, using interrupt transfers.
## Preliminary Parts List

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>Quantity</th>
<th>Lead Time</th>
<th>Price/unit with Tax</th>
<th>Max Current</th>
<th>Distributor</th>
<th>Picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN2131QC</td>
<td>Microprocessor</td>
<td>1</td>
<td>1 Week</td>
<td>10.53</td>
<td>50 mA</td>
<td>Digikey</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>300-6027-ND</td>
<td>12 Mhz Crystal</td>
<td>1</td>
<td>1 Week</td>
<td>88¢</td>
<td>n/a</td>
<td>Digikey</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>n/a</td>
<td>Resistors 20k</td>
<td>32</td>
<td>0</td>
<td>Free</td>
<td>8 mA</td>
<td>WWU</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>CAT# SPR-5</td>
<td>Spring</td>
<td>8</td>
<td>1 Week</td>
<td>10¢</td>
<td>n/a</td>
<td>All Electronics</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>CAT# MPB-131</td>
<td>Pushbutton</td>
<td>8</td>
<td>1 Week</td>
<td>25¢</td>
<td>n/a</td>
<td>All Electronics</td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td>3957</td>
<td>Philips Flat Head</td>
<td>32</td>
<td>1 Week</td>
<td>8¢</td>
<td>n/a</td>
<td>Bolt Depot</td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>AP1701</td>
<td>Reset IC</td>
<td>1</td>
<td>1 Week</td>
<td>96¢</td>
<td>30μA</td>
<td>Anachip</td>
<td><img src="image7.png" alt="Image" /></td>
</tr>
<tr>
<td>MIC2920A-3.3BS</td>
<td>3.3 Volt Regulator</td>
<td>1</td>
<td>1 Week</td>
<td>$1.62</td>
<td>2mA</td>
<td>Micrel</td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>KeyStick</td>
<td>Scrap wood</td>
<td>n/a</td>
<td>0</td>
<td>Free</td>
<td>n/a</td>
<td>me</td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td></td>
<td>83</td>
<td></td>
<td><strong>$19.35</strong></td>
<td>60.03mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>