Game Touch

Project Number MP0012
Entry in SchmartBoard 2010 MCU Challenge – Microchip Segment
Dave Wickliff and Nate Wickliff

1. Project Number

MP0012

Table of Contents

2. Project Description

The Game Touch project is a USB gaming keypad providing customized ergonomics and legends for massively multiplayer online role-playing games (MMORPGs) and other PC games. The project utilizes a SchmartBoard 710-0004-01 board for the Microchip 8 Bit PIC MCU, and the USB interface and capacitive touch sensing features of a Microchip PIC18F26J50 microcontroller.

The Game Touch features 10 capacitive touch sensor pads or “buttons”. These are sensitive to a finger touch. There is no mechanical movement involved like traditional push buttons. Compared to mechanical switches these touch pads provides lower material costs and no mechanical wear out.

When a touch pad is activated it generates a predetermined key-code to the gaming computer via the USB connection. From the computer’s point of view these key-codes are identical to those generated by a standard computer keyboard. For example an “up arrow” key code could be assigned to one of the touch “buttons”. When this touch “button” is activated the computer (and the gaming application running) would react as it would normally to a “up arrow” press on a standard keyboard.

Just about any keypad layout can be created and each touch “button” can generate its own key-code. The Game Touch prototype keypad layout was created to assist game play on a popular MMORPG game. The keypad is used with the left hand for navigation, targeting and most frequently used actions; while the right hand is using a mouse to change viewpoints and other actions. A standard keyboard can still be attached to the gaming computer as well. The keypad legends can easily be customized by printing a new sheet of paper and assembling it into the keypad stack up.

The Game Touch is powered entirely by the USB connection to the gaming computer. The standard 5V USB supply is regulated down to 3.3V for the PIC18F26J50 microcontroller to use. This requires a simple replacement of the regulator device that comes standard with the SchmartModule.

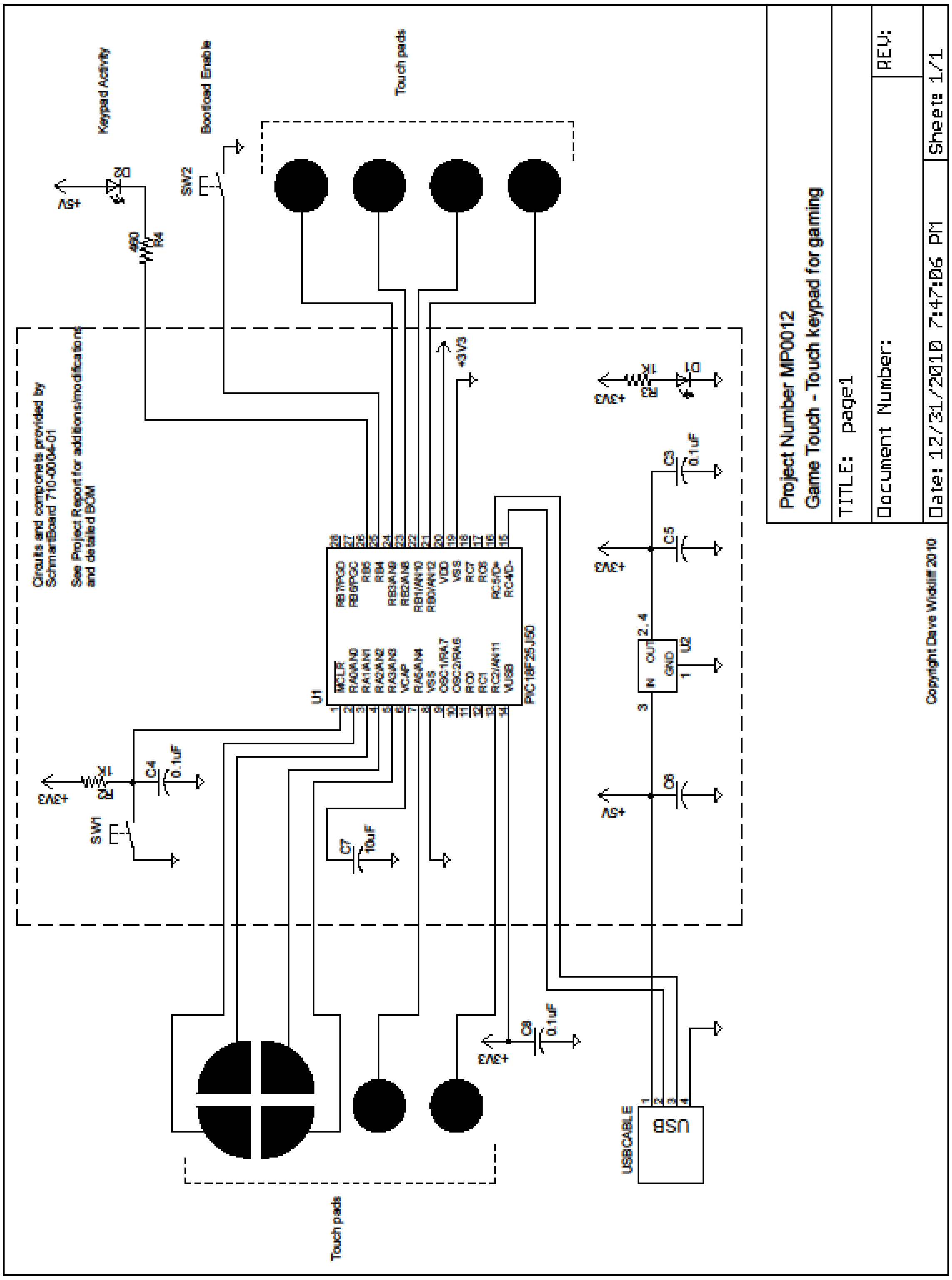
Microchip PIC18F26J50 microcontroller has several key features which are used by the Game Touch:

- The Charge Time Measurement Unit (CTMU) peripheral which provides robust capacitive sensing for the touch pads.
- A Universal Serial Bus (USB) peripheral with internal transceiver and pull-ups
- A high-precision internal oscillator which eliminates the need and cost of an external crystal

Finally, the Game Touch keypad incorporates Microchip’s USB HID Bootloader software. This allows the keypad to be easily reprogrammed with a new software update (e.g. new key-codes) with just a simple download program on the gaming computer via USB. No microcontroller programming tools are required. Holding the Bootloader Enable switch down while plugging in the USB cable puts the keypad in the download mode.

Schematic

Circuit schematic for the project is on the following page. A SchmartBoard 710-0004-01 8 Bit Microchip PIC Development SchmartModule was used to prototype this project. The schematic shows only the devices and circuits of the SchmartModule actually used. See the SchmartBoard documentation for further information. The following subsections list the configuration jumpers, additions and modifications to the standard SchmartModule.


Keypad Activity
Bootload Enable
Touch pads
Circuits and components provided by SchmartBoard 710-0004-01
See Project Report for additions/modifications and detailed BOM
U1
PIC18F25J50
USBCABLE
USB
Touch pads
Project Number MP0012
Game Touch - Touch keypad for gaming
TITLE: page1
Document Number:
REV:
Date: 12/31/2010 7:47:06 PM
Sheet: 1/1
Copyright Dave Widlif 2010

If this is the source MS Word document, then a PDF schematic file is embedded above. Double click above picture to open PDF.

If this is a PDF document or printout, then see the separate PDF schematic file.

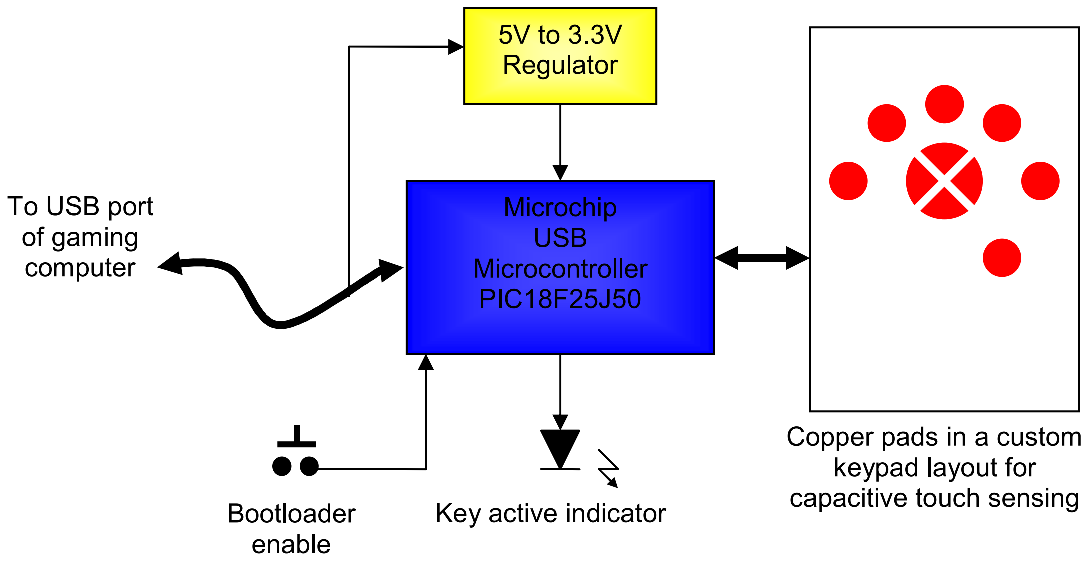
2.1. Configuration jumpers of the 710-0004-01 SchmartModule:

- Jumper pin 5 and pin 6 of J4 “PWR Select” header
- Jumper pin 5 and pin 6 of J5 “GND Select” header
- Jumper pin 7 and pin 8 of J5 “GND Select” header
- Jumper pin 5 and pin 6 of J6 “Reset Select” header
    - Note this jumper should be removed during the use of an ICD debugger

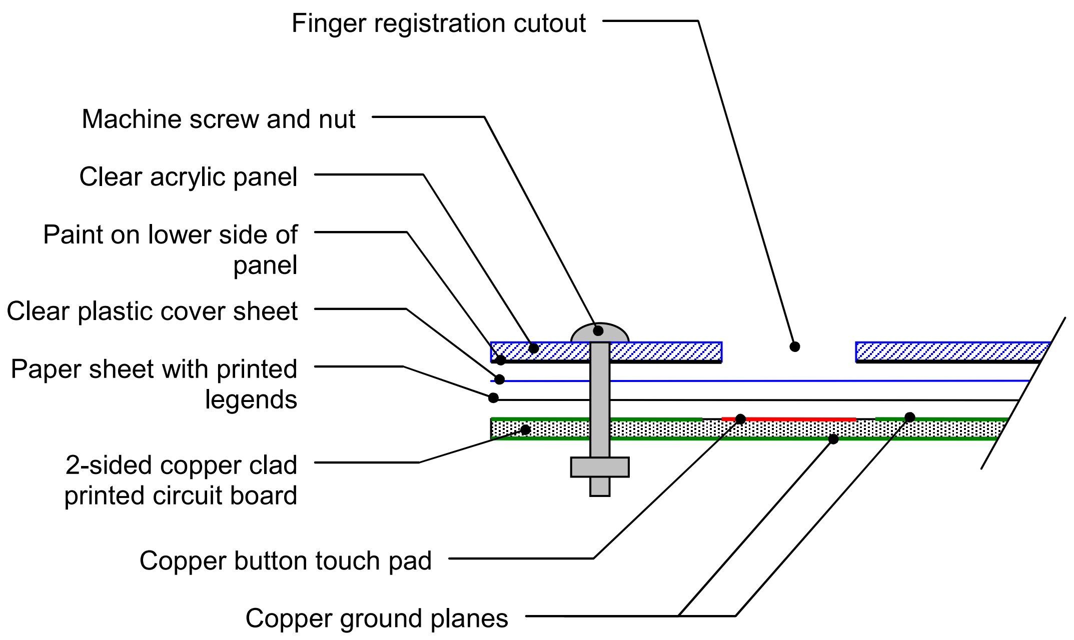
2.2. Additions/Modifications to the 710-0004-01 SchmartModule:

- Add a Microchip PIC18F26J50-I/SO microcontroller to the U1 site
- Replace the surface mounted 5V linear regulator, U2, with a pin-compatible 3.3V linear regulator (e.g. TLV1117-33CDCYRG3 or similar)
- Add a surface mount 10uF capacitor between pin 6 of U1 and ground. Note that the SchmartModule has a PCB site for this capacitor.
- Add a 28-pin header to the J10 “1 to 1 pin-out mapping” site
- Add a 6-pin header to the J3 “28-pin ICSP” site
- Add a 1-pin header to the Vin net of the U2 regulator.
    - This allows easy connection of the +5VDC jumper from the USB cable.
- Add a wire between pin 14 of J10 and VCC.
    - This supplies power to the on-chip USB interface. The wire was connected to the VCC side of the unused R1 site.
- Add a 0.1uF capacitor between pin 14 of J10 and ground.
    - This capacitor is required for the microcontroller’s USB interface. Using a through-hole type capacitor with leads can make the connections easier. The ground lead of the capacitor was connected to the ground side of the unused C2 site.

3. Block Diagram


5V to 3.3V Regulator
Microchip USB Microcontroller PIC18F25J50
To USB port of gaming computer
Bootloader enable
Key active indicator
Copper pads in a custom keypad layout for capacitive touch sensing

3.1. Cross Section Diagram of Keypad Assembly


Finger registration cutout
Machine screw and nut
Clear acrylic panel
Paint on lower side of panel
Clear plastic cover sheet
Paper sheet with printed legends
2-sided copper clad printed circuit board
Copper button touch pad
Copper ground planes

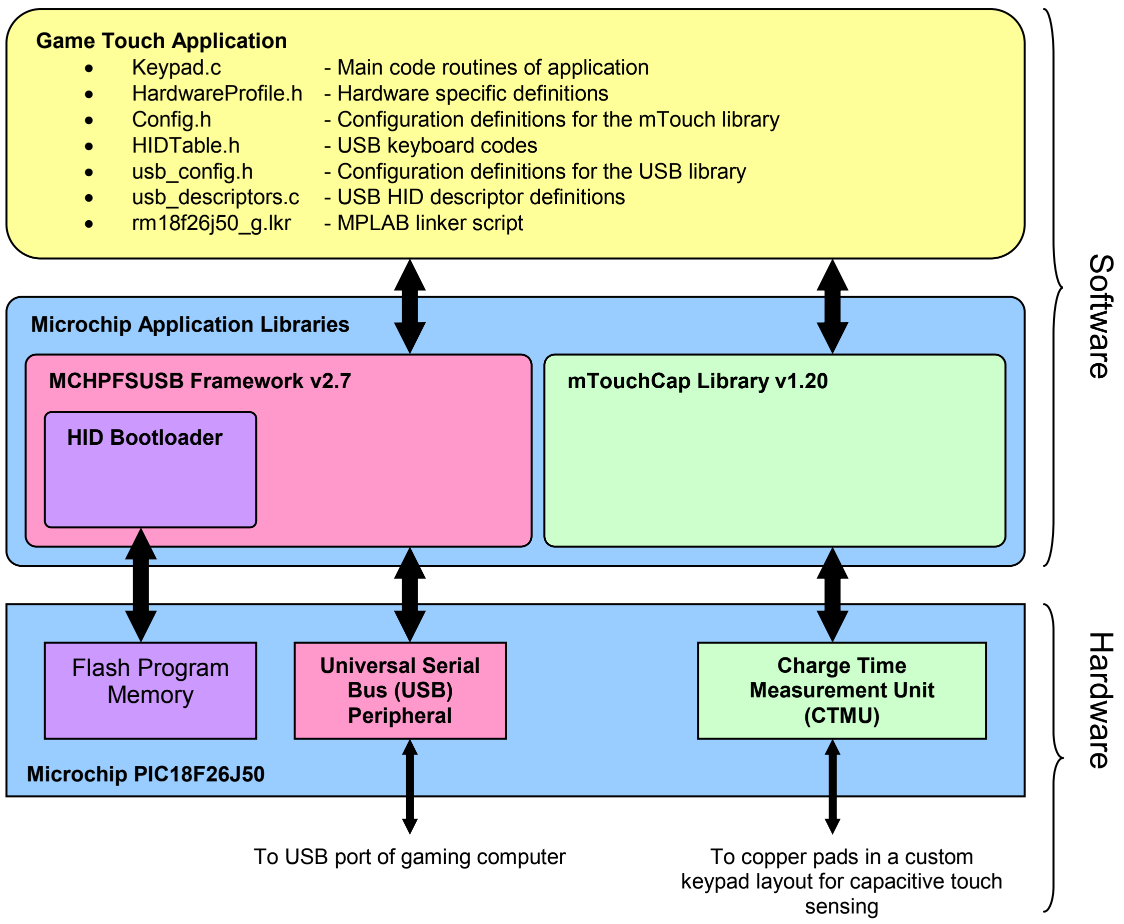
Note that by applying the spray paint on the lower side of the clear acrylic panel a very glossy panel appearance can be achieved on the front side.

4. Source Code

The following tools were used to develop the software:

- Microchip MPLAB Integrated Development Environment (IDE) (version 8.60)
- Microchip MPLAB C Compiler for PIC18 MCUs (version 3.36)
- Microchip MPLAB ICD 3 In-Circuit Debugger System

The following diagram illustrates the overall architecture of the software for the project.


Game Touch Application
Keypad.c - Main code routines of application
HardwareProfile.h - Hardware specific definitions
Config.h - Configuration definitions for the mTouch library
HIDTable.h - USB keyboard codes
usb_config.h - Configuration definitions for the USB library
usb_descriptors.c - USB HID descriptor definitions
rm18f26j50_g.lkr - MPLAB linker script
Microchip Application Libraries
MCHPFSUSB Framework v2.7
HID Bootloader
mTouchCap Library v1.20
Flash Program Memory
Universal Serial Bus (USB) Peripheral
Charge Time Measurement Unit (CTMU)
Microchip PIC18F26J50
To USB port of gaming computer
To copper pads in a custom keypad layout for capacitive touch sensing
Software
Hardware

The Microchip Application Libraries (version 2010-04-28) which includes the USB Framework (version 2.7) and mTouchCap Library (version 1.20) are used to provide the basic HID USB functionality and the basic capacitive touch sensing functionality. The Microchip Application Libraries which includes full source code and documentation is available for download at www.Microchip.com.

The Game Touch Application software uses these Microchip libraries and creates the overall touch keypad functionality. The above diagram lists the individual files that make up the Game Touch Application. These source files of the Game Touch Application can be found the ZIP archive accompanying this report.

4.1. ProcessKeypad – Main processing code routine in Keypad.c

The ProcessKeypad routine provides the core processing for the application:

```
//
*****************************************************************************************************
void ProcessKeypad(void)
//
// Scan touch pads and send USB reports back to host with key pressed information
//
//
*****************************************************************************************************

{
    BOOL SomeButtonsPressed;    // Flag to indicate that there is at least one key pad/button pressed

    if(dataReadyCTMU == 1)     // This flag is set by Timer 4 ISR //when all touch channels have been
read
    {
        dataReadyCTMU = 0;     // Clear touch flag

        // Stop scanning touch pads
           Set_ScanTimer_IF_Bit_State(DISABLE);         // Clear timer 4 SHORT flag
          Set_ScanTimer_IE_Bit_State(DISABLE);         // Disable interrupt
          Set_ScanTimer_ON_Bit_State(DISABLE) ;        // Stop timer 4


        // Check if the IN endpoint is not busy, and if it isn't check if we want to send
        // key touch data to the host.
        if(!HIDTxHandleBusy(lastINTransmission))
        {
            // Build an empty (no keys pressed) USB HID report
            hid_report_in[0] = 0;
            hid_report_in[1] = 0;
            hid_report_in[2] = 0;
            hid_report_in[3] = 0;
            hid_report_in[4] = 0;
            hid_report_in[5] = 0;
            hid_report_in[6] = 0;
            hid_report_in[7] = 0;

            //****************************************************************************
            // Look for any touch buttons pressed and populated the HID buffer with values
            SomeButtonsPressed = FALSE;

            // Look for any arrow touch buttons
            // Report multiple key/button presses (populate a different buffer location for each)
            if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey1))
            {
                hid_report_in[3] = HIDKey_UpArrow;
                SomeButtonsPressed = TRUE;
            }
            if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey2))
            {
                hid_report_in[4] = HIDKey_LeftArrow;
                SomeButtonsPressed = TRUE;
            }
            if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey3))
            {
                hid_report_in[5] = HIDKey_DownArrow;
                SomeButtonsPressed = TRUE;
            }
            if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey4))
            {
                hid_report_in[6] = HIDKey_RightArrow;
                SomeButtonsPressed = TRUE;
            }
```

ProcessKeypad routine continued:

```
            // Look for any "function" touch buttons
            // Report only the first key/button detected (populate the same buffer location)
            if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey5))
            {
                hid_report_in[2] = HIDKey_0;
                SomeButtonsPressed = TRUE;
            }
            else if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey6))
            {
                hid_report_in[2] = HIDKey_9;
                SomeButtonsPressed = TRUE;
            }
            else if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey7))
            {
                hid_report_in[2] = HIDKey_3;
                SomeButtonsPressed = TRUE;
            }
            else if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey8))
            {
                hid_report_in[2] = HIDKey_2;
                SomeButtonsPressed = TRUE;
            }
            else if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey9))
            {
                hid_report_in[2] = HIDKey_1;
                SomeButtonsPressed = TRUE;
            }
            else if(KEY_PRESSED == mTouchCapAPI_GetStatusDirectButton(&DirectKey10))
            {
                hid_report_in[2] = HIDKey_Insert;
                SomeButtonsPressed = TRUE;
            }

            if(SomeButtonsPressed) LEDKeyActivity = 1;        // Turn LED OFF when any keypads are
being pressed
            else LEDKeyActivity = 0;                        // Turn LED back ON when no keypads are
being pressed

               // Send the 8 byte packet over USB to the host with any key data
               lastINTransmission = HIDTxPacket(HID_EP, (BYTE*)hid_report_in, 0x08);

        }

        // Resume scanning touch pads
        Set_ScanTimer_IF_Bit_State(DISABLE);       // Clear flag
           Set_ScanTimer_IE_Bit_State(ENABLE);        // Enable interrupt
           Set_ScanTimer_ON_Bit_State(ENABLE) ;       //Run timer

    } // end if(dataReadyCTMU)

    // Check if any data was sent from the PC to the keypad.
    if(!HIDRxHandleBusy(lastOUTTransmission))
    {
        // Do nothing with this informaiton

        lastOUTTransmission = HIDRxPacket(HID_EP,(BYTE*)&hid_report_out,1);
    }

    return;
}//end ProcessKeypad()
```

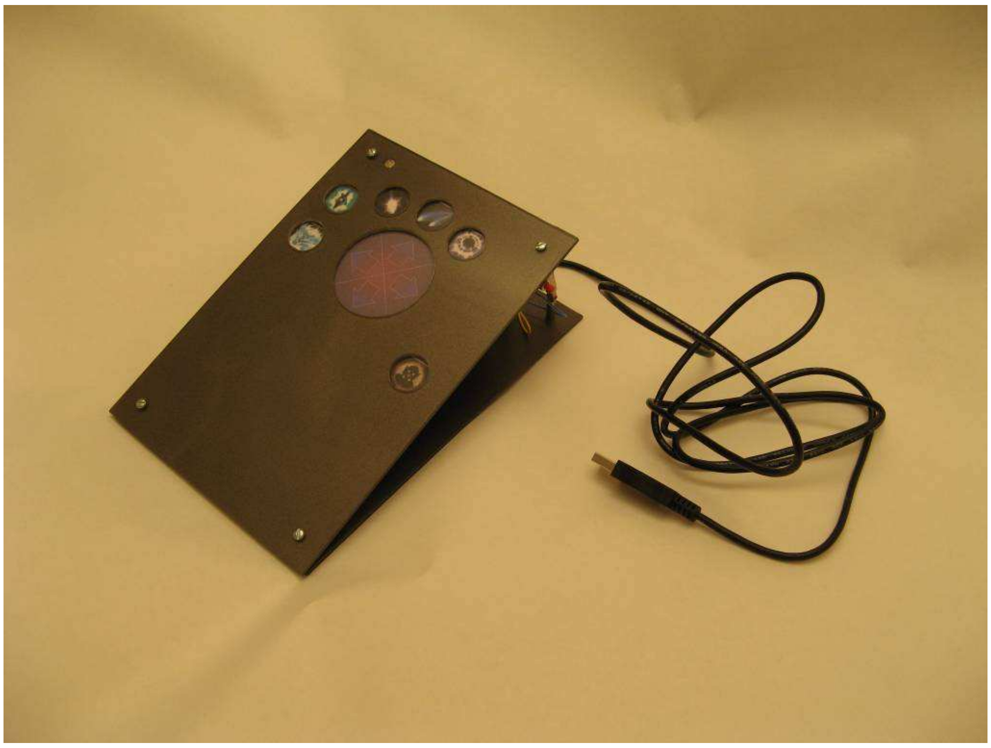
5. Bill of Materials

SchmartModule 710-0004-01 8 Bit Microchip PIC Development Board			
Device	**Value**	**Qty**	**Schematic Part Number**
Push button switch	-	1	SW1
Resistor	1K ohm	2	R2, R3
Capacitor	0.1uF	2	C3, C4
Capacitor	-	2	C5, C6
LED	-	1	D1
Shorting jumpers for configuration	-	4	-

Additional Devices			
Device	**Value**	**Qty**	**Schematic Part Number**
IC, 3.3V LDO regulator	TLV1117-33CDCYRG3 or similar	1	U2
Capacitor	10uF	1	C8
Male USB A cable	~3 feet	1	USBCABLE
LED	blue	1	D2
Resistor	460 ohm	1	R4
Push button switch	-	1	SW2

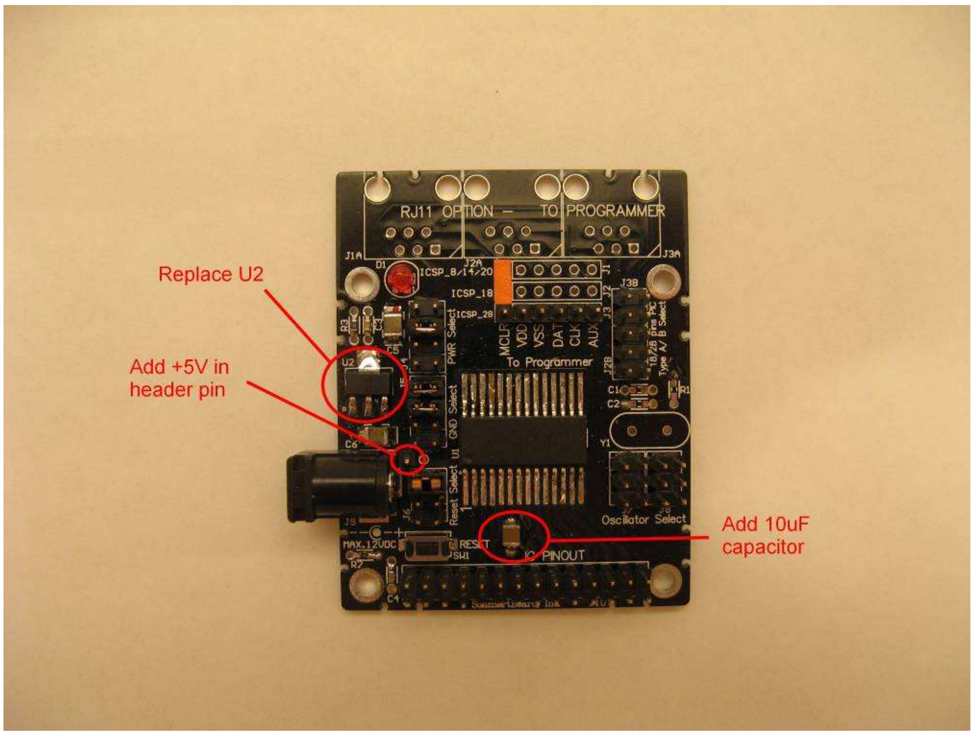
Additional Materials
Two 8”x5.5” x 0.1” acrylic panels
Spray paint in color of choice
One sheet of transparency film
One sheet of paper with printed legends of choice
One 8”x5.5” blank double-side copper-plated circuit board
Assorted machine screws, nuts and standoffs
Assorted wire jumpers and headers

6. Pictures

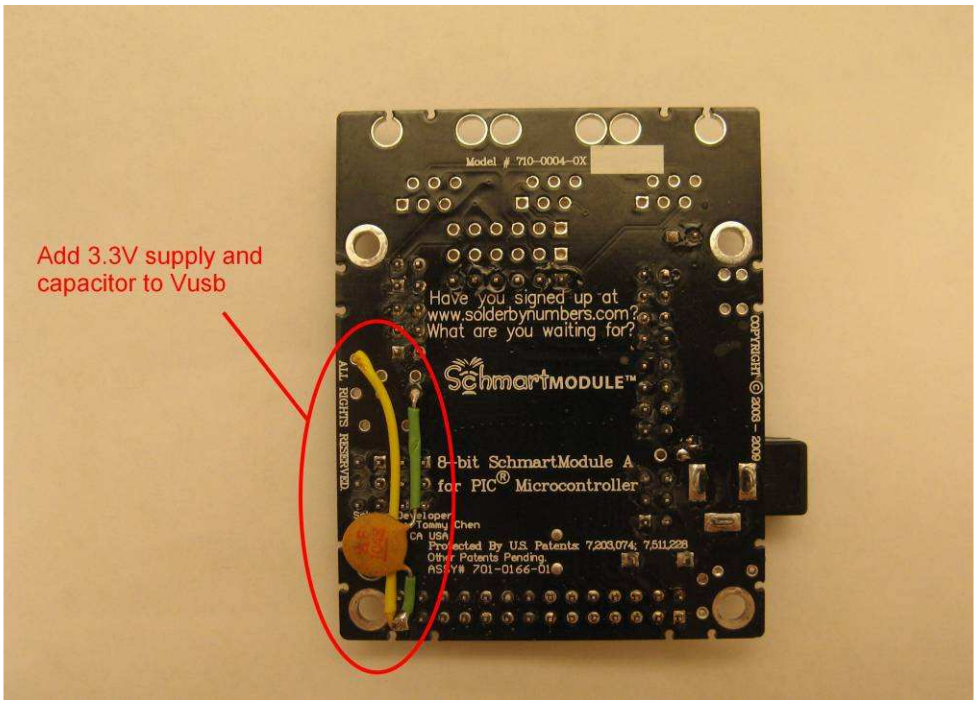

Game Touch - gaming keypad – finished view

Dave Wickliff and Nate Wickliff holding the Game Touch

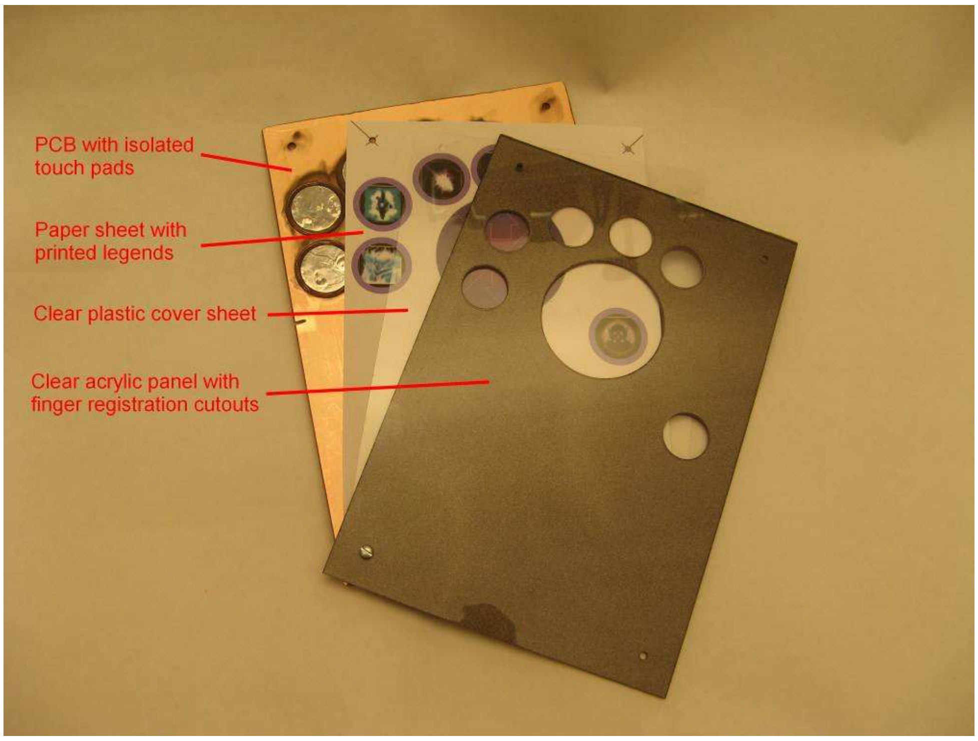
Game Touch in Use



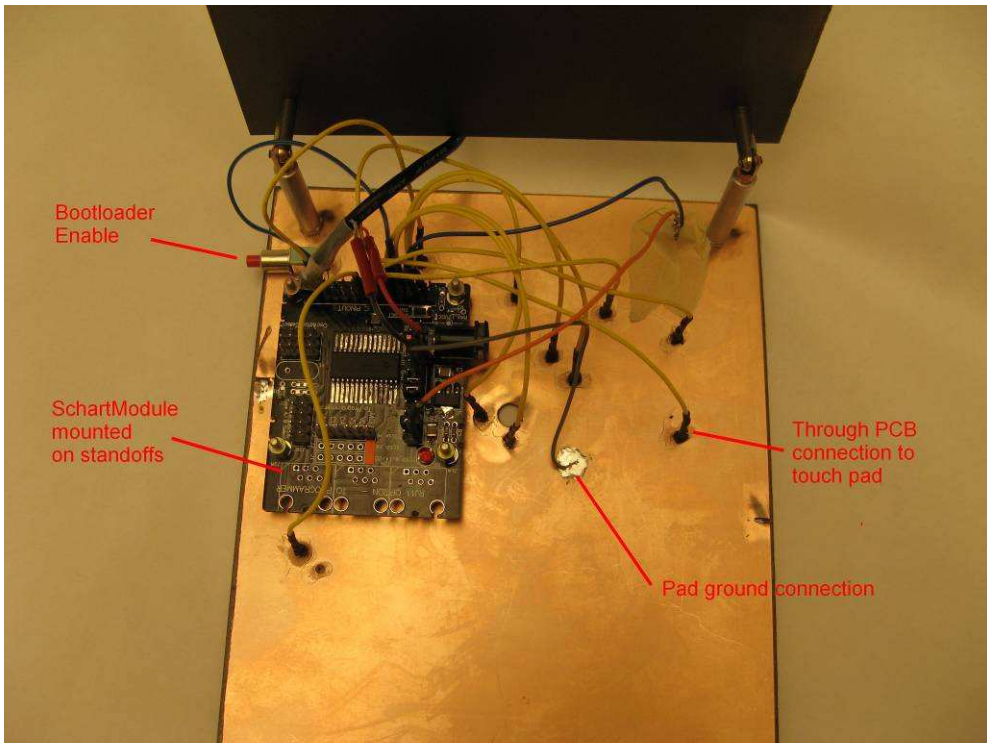
Modifications to front of SchmartModule


Add 3.3V supply and capacitor to Vusb
Have you signed up at www.solderbynumbers.com? What are you waiting for?
SchmartMODULE™
8-bit SchmartModule A for PIC® Microcontroller
ALL RIGHTS RESERVED.
COPYRIGHT © 2003 - 2009

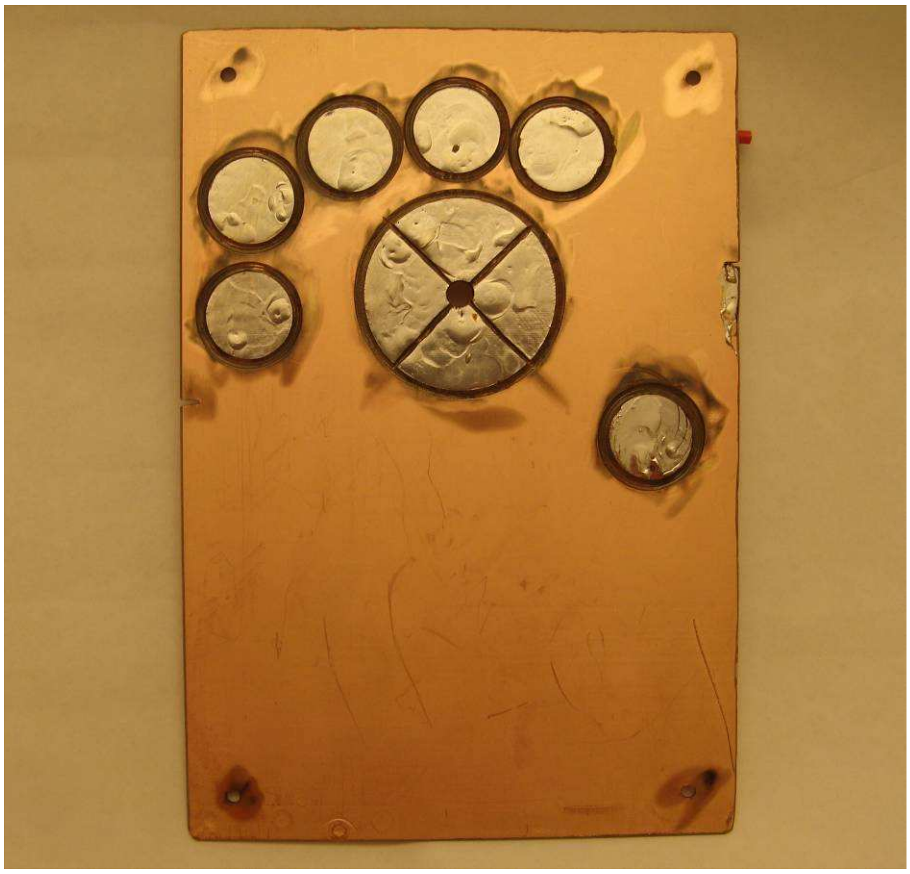
Modifications to rear of SchmartModule


PCB with isolated touch pads
Paper sheet with printed legends
Clear plastic cover sheet
Clear acrylic panel with finger registration cutouts

Keypad stack up view


Bootloader Enable
SchartModule mounted on standoffs
Through PCB connection to touch pad
Pad ground connection

Inside rear view



Front view of bare copper clad board with touch pads

7. References

- Microchip Application Note AN1250, “Microchip CTMU for Capacitive Touch Applications”, www.microchip.com

- Microchip PIC18F46J50 Family Data Sheet, www.microchip.com

- Microchip Application Libraries, “USB MCHPFSUSB Framework v2.7 Help and Documentation”, www.microchip.com

- Microchip Application Libraries, “mTouchCap Library v1.20 Help and Documentation”, www.microchip.com

- Documentation for “8 Bit Microchip PIC Microcontroller Development SchmartModule 710-0004-01”, www.schmartboard.com

- USB Implementers’ Forum, “Device Class Definition for Human Interface Devices (HID)”, Version 1.11, www.usb.org

- USB Implementers’ Forum, “HID Usage Tables”, Version 1.12, www.usb.org

8. Appendix A – Keypad Template

Example template used for the prototype keypad. Printout several sheets at 100% scale and use to mark and drill/cut holes in acrylic panels and printed circuit board. Overall size is 5.5”x8.0”. Also use to place drawings/photos on legend paper sheet.