A PS/2 Keyer: Using a Keyer Paddle to Emulate a PS/2 Keyboard and Mouse

Would you like to operate your computer without a keyboard? How about RTTY or PSK-31 without a keyboard? Now you can.

This article describes my first programming project with a Microchip PIC® microcontroller. The program for this project, written in C, emulates a PS/2 keyboard and a PS/2 mouse using a CW keyer paddle for input.

Note: an earlier version of this article was published in The 28th Annual ARRL and TAPR Digital Communications Conference Proceedings. The 28th Annual DCC was held in Chicago, Illinois on September 25-27, 2009.

Over the past several years, I became interested in learning how to program Microchip PIC microcontrollers and I began to look for an interesting project. I am an experienced C programmer, but I knew nothing about PIC microcontroller programming. My criteria for the project was that it needed to have a well defined input, a well defined output, and the circuit needed to consist solely of a PIC microcontroller with some light emitting diodes (LED). And, importantly, it needed to be written in C.

At the 26th DCC, Milt Cram, W8NUE and George Heron, N2APB introduced their NUE-PSK digital modem. The full details of the NUE-PSK modem were published in the Mar/Apr 2009 issue of QEX (NUE-PSK Digital Modem Enables PSK31 Field Operation Without Using a PC!), along with a summary article in the March 2008 issue of QST. The NUE-PSK modem is a small device that provides portable PSK31 (and now RTTY) operation without the use of a personal computer. It does, however, require a PS/2 keyboard for entering text. It occurred to me that it could be more portable if the large PS/2 keyboard were replaced with a CW keyer paddle. A PIC microcontroller would translate CW input sent on the paddle into the output from a standard PS/2 keyboard. Hence, I found the idea for the project for which I was looking. I would write a program that runs on a PIC, and that emulates a PS/2 keyboard using a keyer paddle for input. Later on in the project, I wondered if it was also possible to emulate a PS/2 mouse with only two switch contacts and no other moving parts.

Background

Morse Code

Morse code input is defined in the ITU recommendation on the international Morse code and is further described in an article in Wikipedia. The CW character and word timings needed for this project are the length of time of a dash relative to a dot, the length of time between dots and dashes in a letter, the length of time between letters and the length of time between words.

The PS/2 Protocol

The output of a PS/2 keyboard and a PS/2 mouse follow the PS/2 protocol, which was originally described in the IBM Personal System/2 Hardware Technical Reference, in the sections on the “101 and 102 Key Keyboard” and the “Keyboard and Auxiliary Device Controller” sections of the computer reference manual, respectively. Articles on the Internet about the PS/2 protocol, the PS/2 keyboard protocol, and the PS/2 mouse protocol were most helpful. More recently, an article describing a keyboard-game interface using the PS/2 protocol appeared in Nuts and Volts magazine.

The physical PS/2 interface, shown in Figure 1, used for the keyboard and mouse connectors uses the PS/2 protocol. The PS/2 protocol is a two way synchronous protocol used to communicate between a host and a device. (See Note 6.) A host, typically, is a personal computer and a device, typically, is a keyboard or a mouse. In addition to a clock line and a data line, there is a 5 V line and a ground line. The PS/2 protocol uses the clock line and the data line for sending data. The frequency of the clock is within the range of 10 to 16.7 kHz. Data is sent as an 11 bit frame, starting with a zero start bit, the eight data bits, with the least significant bit first, an odd parity bit and one stop bit. The device always generates the clock sig-
nal. The host can request to communicate with the device by pulling the clock line low for at least 100 µs, then pulling the data line low and then releasing the clock line back high. The device responds by generating the clock signal for the host to send its data to the device. Finally, the device acknowledges that it has received the complete data from the host.

**The PS/2 Keyboard Protocol**

Each key on a PS/2 keyboard device is identified by a unique scan code. Scan code set 2 is commonly used today and it was originally developed by IBM for the AT keyboard. The keyboard device sends to the host the scan code of a key that is pressed; a break code and its scan code is then sent when that key is released. The host computer can communicate with the keyboard device. For example, the host commands the keyboard to light its Caps Lock LED when the Caps Lock key is pressed.

**The PS/2 Mouse Protocol**

The standard PS/2 mouse device sends the host its movement and button information as a three byte packet. Mouse movement information is sent as a relative position change and is a signed nine bit two’s complement binary number. Also, a standard PS/2 mouse has a left, a middle, and a right button. Initially, the host communicates with the mouse to configure it after the host has determined whether the mouse is a standard PS/2 mouse or an enhanced PS/2 mouse.

---

**The Project**

**Learning the PS/2 Protocol**

To gain an understanding of the PS/2 protocol, I needed to see the data and the clock lines at the wire level on a PS/2 cable.

---

**Figure 2** — This was my development breadboard. At the top of the photo, the four logic analyzer probes are connected to the clock and data lines of the keyboard and mouse connectors. The other two logic analyzer probes are connected to the –MCLR pin and a mark pin.

**Figure 3** — This is the schematic diagram of the PS/2 Keyer.
I assembled a six pin mini-DIN connector breakout board from parts obtained from SparkFun Electronics, a wonderful resource for digital electronics hobbyists. Sample PS/2 protocol data came from an original IBM AT computer PS/2 keyboard and a Logitech PS/2 three button mouse. I purchased an inexpensive logic analyzer from Saleae. Since the PS/2 data bits are sent in “reverse” order — least significant bit first — I created a paper form to record the start bit, eight data bits, parity bit, stop bit and any acknowledgment bit. Later in the project, I purchased a USBee SX logic analyzer, since it can decode the PS/2 protocol for the host and the device.

### PIC18F4520 Prototyping Boards

I needed to choose a PIC microcontroller for my project. Rick Hambly, W2GPS, develops and sells time related products using PIC microcontrollers and he recommended that I use the 18F series PIC microcontrollers. I purchased the PIC18F4520 development kit from CCS: it includes a prototyping board, and most valuably, an exercise booklet to help get started. Note that the PIC18F4520 prototyping board and exercise booklet can be purchased separately from the complete kit. Other prototyping and demonstration boards I have tried that use the PIC18F4520 are the PICDEM 2 Plus demonstration board from Microchip, the Dem2PLUS demonstration board from Sure Electronics, and the Olimex 40 pin bare prototyping board with RS-232 from SparkFun. A USB version of the Olimex 40 pin bare prototyping board is also available.

I assembled a prototyping board as shown in Figure 2 by using a breadboard purchased from Beginner Electronics, a PIC18F4520 microcontroller in the PDIP form factor, two six pin mini-DIN connectors, an ICD programming connector and a serial TTL to RS-232 adapter. I connected the 5 V power supply to the prototyping board with a positive temperature coefficient (PTC) resettable fuse device, available from SparkFun, at both mini-DIN connectors and at the ICD programming connector. These devices protect the prototyping board and power supply sources against an accidental short circuit. The two mini-DIN connectors are connected to a personal computer using a PS/2 keyboard-and-mouse-to-USB adapter and two PS/2 male-to-male cables. The ICD connector is connected to a PIC programmer with a short cable. As luck would have it, we receive and transmit pins of the RS-232 adapter match the corresponding input/output pins on the PIC18F4520. Note the probes connected to the logic analyzer in Figure 2.

The circuit schematic diagram shown in Figure 3 consists of essentially one component — the PIC18F4520 microcontroller. The PS/2 keyboard connector, the PS/2 mouse connector and the CW keyer paddle is connected to port B pins of the PIC to take advantage of the internal pull-up resistors provided within the microcontroller. No external clock crystal or resonator is needed since the internal 8 MHz clock within the PIC is used instead.

### PIC Development Tools

For the PIC programmer, I used the Microchip PICkit™ 2 programmer/debugger. The PICkit 2 is inexpensive and was easy to use for program development and testing. I also quickly learned not to introduce a timing error in the code by putting a `printf()` in the wrong place.

### Writing the Program

This program is organized into three software modules. The first module decodes CW characters keyed in with the paddle. The invalid CW character `di di dah dah` is used to indicate that the next character entered is a command code. Some command codes are keyboard Enter, keyboard Caps Lock, an

---

**Table 1**

The Parts List to Build the PS/2 Keyer on a Circuit Board

<table>
<thead>
<tr>
<th>Part</th>
<th>Quantity</th>
<th>Part Name</th>
<th>Vendor</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1, C2</td>
<td>2</td>
<td>0.1 μF 50 V 10% PC-Mount Capacitor</td>
<td>Digi-Key</td>
<td>BC1148CT-ND</td>
</tr>
<tr>
<td>D5</td>
<td>1</td>
<td>Red round diffused lens LED</td>
<td>Digi-Key</td>
<td>P589-ND</td>
</tr>
<tr>
<td>R1-R5</td>
<td>5</td>
<td>330 Ω ½ W resistor</td>
<td>Digi-Key</td>
<td>P330BACT-ND</td>
</tr>
<tr>
<td>R6</td>
<td>1</td>
<td>10 kΩ ½ W resistor</td>
<td>Digi-Key</td>
<td>10KQBK-ND</td>
</tr>
<tr>
<td>U1</td>
<td>1</td>
<td>40 pin IC socket</td>
<td>Digi-Key</td>
<td>3M5471-ND</td>
</tr>
<tr>
<td>U1</td>
<td>1</td>
<td>PIC18F4520-I/P</td>
<td>Digi-Key</td>
<td>PIC18F4520-I/P-ND</td>
</tr>
<tr>
<td>F</td>
<td>4</td>
<td>Self-Stick Rubber Feet</td>
<td>Micro Center</td>
<td>SKU: 133314</td>
</tr>
<tr>
<td>J1, J3</td>
<td>1</td>
<td>PS/2 keyboard extension cable with purple connectors</td>
<td>Micro Center</td>
<td>SKU: 133272</td>
</tr>
<tr>
<td>J2, J4</td>
<td>1</td>
<td>PS/2 mouse extension cable with green connectors</td>
<td>Micro Center</td>
<td>Catalog #: 274-141</td>
</tr>
<tr>
<td>J4</td>
<td>1</td>
<td>¼ inch In-Line Stereo Audio Jack</td>
<td>RadioShack</td>
<td>SKU: COM-08532</td>
</tr>
<tr>
<td>D1-D4</td>
<td>4</td>
<td>Green rectangular clear lens LED</td>
<td>SparkFun</td>
<td>SKU: 919712</td>
</tr>
</tbody>
</table>

---

Reprinted with permission © ARRL
error code and a switch to mouse mode. The error code generates the appropriate number of backspace characters to the beginning of a word just entered. CW is translated into ASCII text characters using a lookup table.

The paddle dot and dash contacts are connected to pins RB4 and RB5, respectively, of port B to take advantage of the change-on-input interrupt feature. This allows the PIC to go to sleep and consume practically no power while waiting for an interrupt to occur when a paddle lever is pressed.

Keyer paddle switch contact bounce was probably the most challenging problem of this project. A paper on switch bounce convinced me that a keyer paddle without any switch contacts, such as the Touchkeyer paddle is necessary for this project. An internal timer interrupt is used to measure time between dots and dashes. Using port B input interrupts and timer interrupts simplified the code.

The second module emulates a PS/2 keyboard using the PS/2 protocol. ASCII characters are translated into PS/2 keyboard scan codes using a lookup table. The lookup table has every character of a PS/2 keyboard, including those not found in Morse code. It was a thrill to first see the letter Q generated by the PIC microcontroller appear in a Notepad text editor window!

The third module emulates a PS/2 standard three-button mouse. It generates PS/2 mouse button clicks and mouse pointer movement from the keyer paddle input. Clicking — briefly pressing — the left paddle lever generates a left mouse button click and, correspondingly, clicking the right paddle lever generates a right mouse button click. Mouse pointer movement is controlled by pressing and holding the paddle lever: the left contact controls the left-right mouse pointer movement; the right contact controls the up-down mouse pointer movement. Pressing both sides moves the mouse pointer along one diagonal direction or the other diagonal direction. Hence, there are eight possible mouse movement directions, and it is possible to move the mouse pointer in an...
Pressing for longer than about three seconds causes the mouse pointer to move faster on the screen. A command code of three left clicks switches the program back to keyboard mode. Again, it was fun to see the mouse pointer move by itself in a continuous circle on the screen during initial testing. It took about two weeks of testing and experimenting with different mouse pointer movement policies to get something reasonable to control mouse pointer movement.

Testing during software development was not difficult. A PS/2 keyboard and mouse to USB adapter, a logic analyzer and LEDs were used. During program startup, the PIC would send a PS/2 keyboard and a PS/2 mouse device reset code via the PS/2 to USB adapter to the host computer. The computer would then respond with PS/2 keyboard and PS/2 mouse packets. In the circuit board after its construction. I used a logic analyzer to help my understanding of what was happening at points of interest in the code. Table 1 provides the parts list of readily available components to construct a circuit board. Figure 5 shows a PICkit 2 programming cable wired according to the information in Table 2.

Testing during software development was not difficult. A PS/2 keyboard and mouse to USB adapter, a logic analyzer and LEDs were used. During program startup, the PIC would send a PS/2 keyboard and a PS/2 mouse device reset code via the PS/2 to USB adapter to the host computer. The computer would then respond with PS/2 keyboard and PS/2 mouse packets. In the circuit board after its construction. I used a logic analyzer to help my understanding of what was happening at points of interest in the code. Table 1 provides the parts list of readily available components to construct a circuit board. Figure 5 shows a PICkit 2 programming cable wired according to the information in Table 2.

Testing during software development was not difficult. A PS/2 keyboard and mouse to USB adapter, a logic analyzer and LEDs were used. During program startup, the PIC would send a PS/2 keyboard and a PS/2 mouse device reset code via the PS/2 to USB adapter to the host computer. The computer would then respond with PS/2 keyboard and PS/2 mouse packets. In the circuit board after its construction. I used a logic analyzer to help my understanding of what was happening at points of interest in the code. Table 1 provides the parts list of readily available components to construct a circuit board. Figure 5 shows a PICkit 2 programming cable wired according to the information in Table 2.

Testing during software development was not difficult. A PS/2 keyboard and mouse to USB adapter, a logic analyzer and LEDs were used. During program startup, the PIC would send a PS/2 keyboard and a PS/2 mouse device reset code via the PS/2 to USB adapter to the host computer. The computer would then respond with PS/2 keyboard and PS/2 mouse packets. In the circuit board after its construction. I used a logic analyzer to help my understanding of what was happening at points of interest in the code. Table 1 provides the parts list of readily available components to construct a circuit board. Figure 5 shows a PICkit 2 programming cable wired according to the information in Table 2.

Testing during software development was not difficult. A PS/2 keyboard and mouse to USB adapter, a logic analyzer and LEDs were used. During program startup, the PIC would send a PS/2 keyboard and a PS/2 mouse device reset code via the PS/2 to USB adapter to the host computer. The computer would then respond with PS/2 keyboard and PS/2 mouse packets. In the circuit board after its construction. I used a logic analyzer to help my understanding of what was happening at points of interest in the code. Table 1 provides the parts list of readily available components to construct a circuit board. Figure 5 shows a PICkit 2 programming cable wired according to the information in Table 2.

Testing during software development was not difficult. A PS/2 keyboard and mouse to USB adapter, a logic analyzer and LEDs were used. During program startup, the PIC would send a PS/2 keyboard and a PS/2 mouse device reset code via the PS/2 to USB adapter to the host computer. The computer would then respond with PS/2 keyboard and PS/2 mouse packets. In the circuit board after its construction. I used a logic analyzer to help my understanding of what was happening at points of interest in the code. Table 1 provides the parts list of readily available components to construct a circuit board. Figure 5 shows a PICkit 2 programming cable wired according to the information in Table 2.

Testing during software development was not difficult. A PS/2 keyboard and mouse to USB adapter, a logic analyzer and LEDs were used. During program startup, the PIC would send a PS/2 keyboard and a PS/2 mouse device reset code via the PS/2 to USB adapter to the host computer. The computer would then respond with PS/2 keyboard and PS/2 mouse packets. In the circuit board after its construction. I used a logic analyzer to help my understanding of what was happening at points of interest in the code. Table 1 provides the parts list of readily available components to construct a circuit board. Figure 5 shows a PICkit 2 programming cable wired according to the information in Table 2.
PSK modem, Rick Hambly, W2GPS, for giving me guidance, and Steve Bible, N7HPR, for encouraging me to write up this project for the 2009 Digital Communications Conference.

Notes


7 Adam Chapweske, The PS/2 Keyboard Interface, 2003; www.computer-engineering.org/ps2keyboard/

8 Adam Chapweske, The PS/2 Mouse Interface, 2003; www.computer-engineering.org/ps2mouse/


10 Adam Chapweske, Keyboard Scan Codes: Set 2, www.computer-engineering.org/ps2keyboard/scan_codes2.html


13 Saleae LLC, Saleae logic analyzer; www.saleae.com/logic/

14 CWAV, Inc, USBee SX logic analyzer; www.usbee.com/sx.html


19 SparkFun Electronics, 40 Pin PIC

---

```c
#include <18F4520.h>

#define PORT0_LED PIN_A0
#define PORT1_LED PIN_A1
#define PORT2_LED PIN_A2
#define PORT3_LED PIN_A3
#define DELAY 250

void blink_port_LEDS(void)
{
    output_high(PORT0_LED); delay_ms(DELAY);
    output_low(PORT0_LED); delay_ms(DELAY);
    output_high(PORT1_LED); delay_ms(DELAY);
    output_low(PORT1_LED); delay_ms(DELAY);
    output_high(PORT2_LED); delay_ms(DELAY);
    output_low(PORT2_LED); delay_ms(DELAY);
    output_high(PORT3_LED); delay_ms(DELAY);
    output_low(PORT3_LED); delay_ms(DELAY);
}
```

**Figure 7** — This is a sample C program for the PIC18F4520 using the CCS C compiler.

Beginner Electronics, Breadboard and Wire Kit; www.beginnerelectronics.com/beginner/Products.php


SparkFun Electronics, Adapter board for Microchip ICD and ICD2; www.sparkfun.com/commerce/product_info.php?products_id=193


Inland Pro USB Converter USB to PS/2 Keyboard and Mouse, SKU: 919712; www.microcenter.com/single_product_results.phtml?product_id=0230515

Cable Club, 6 ft PS/2 Keyboard and Mouse Interface Cable (Male/Male), part: BC20277-6; www.cableclub.com/keyboard-mouse-interface-cable-male-male-p-797.html

Microchip Technology Inc, PICKit 2 Development Programmer/Debugger; www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&DDocName=en023805

CCS, Inc, Compiler Exclusively for Microchip PIC® MCUs; www.ccsinfo.com/content.php?page=comilers


Nuts & Volts Magazine, The Magazine for the Electronics Hobbyist, ISSN 1528-9885; www.nutsvolts.com

Circuit Cellar, The Magazine for Computer Applications, ISSN 1528-0608; www.circuitcellar.com


CW Touchkeyer touch paddle, model P1PADW; www.cwtouchkeyer.com/P1PADW.htm

The program source code file and circuit board pattern files in ExpressPCB format are available for download from the ARRL QEX Web site. Go to www.arrl.org/qexfiles and look for the file 5x10_Bern.zip.


David Bern, W2LNX, was first licensed in 1979 as N2AER with an Advanced Class and upgraded to an Amateur Extra Class in 2000. Later, he obtained his W2LNX vanity call sign since he is also an avid Linux enthusiast. As a high school student, he earned his First-Class Radiotelephone Operator License with Ship Radar Endorsement. In 1977, he earned a BS in Computer Science from City College of New York and then in 1983 earned an MS in Computer Science from New York University. He is a professional software developer and also an adjunct professor of engineering at Montgomery College, Rockville, Maryland. He prefers building and experimenting with ham radio projects to operating but enjoys operating QRP digital modes in the summer with his son Adam, KB3KVD, in the Virginia mountains. Currently, he is learning microcontroller programming and digital signal processing.