David Bern, W2LNX

8809 Cold Spring Rd, Potomac, MD 20854; w2lnx@arrl.net

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.[1] 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.

[1]Notes appear on page 8.

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.[2, 3] 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 *dot*s and *dashe*s 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.[4, 5] Articles on the Internet about the PS/2 protocol, the PS/2 keyboard protocol, and the PS/2 mouse protocol were most helpful.[6, 7, 8] More recently, an article describing a keyboard-game interface using the PS/2 protocol appeared in *Nuts and Volts* magazine.[9]

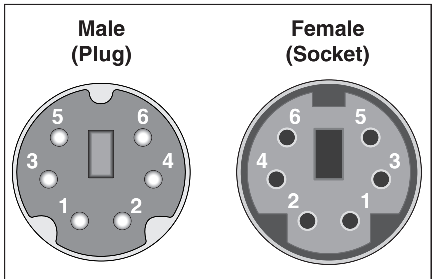

Figure 1 — The PS/2 ports for the keyboard and mouse on personal computers use six pin mini-DIN connectors.

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 signal. 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.[10] 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.

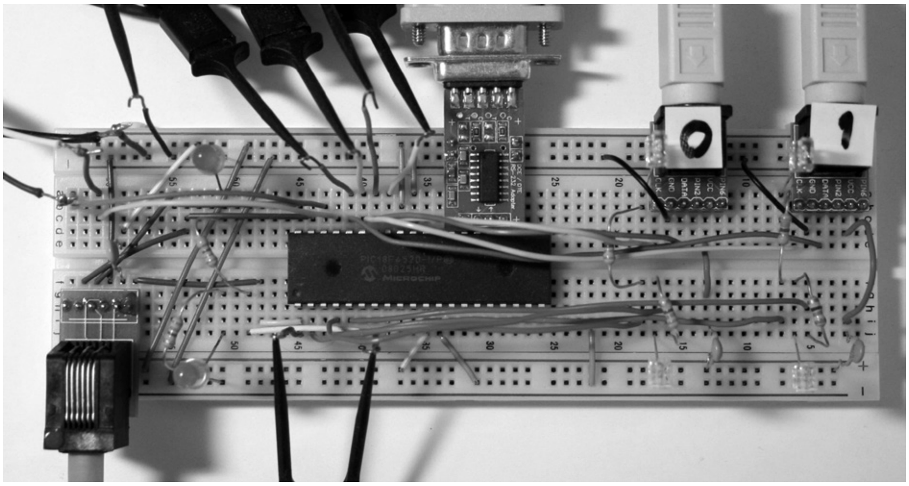

Figure 2 — This was my development breadboard. At the top of the photo, the four logic analyzer probes are connected to the clock and data lines of the keyboard and mouse connectors. The other two logic analyzer probes are connected to the –MCLR pin and a mark pin.

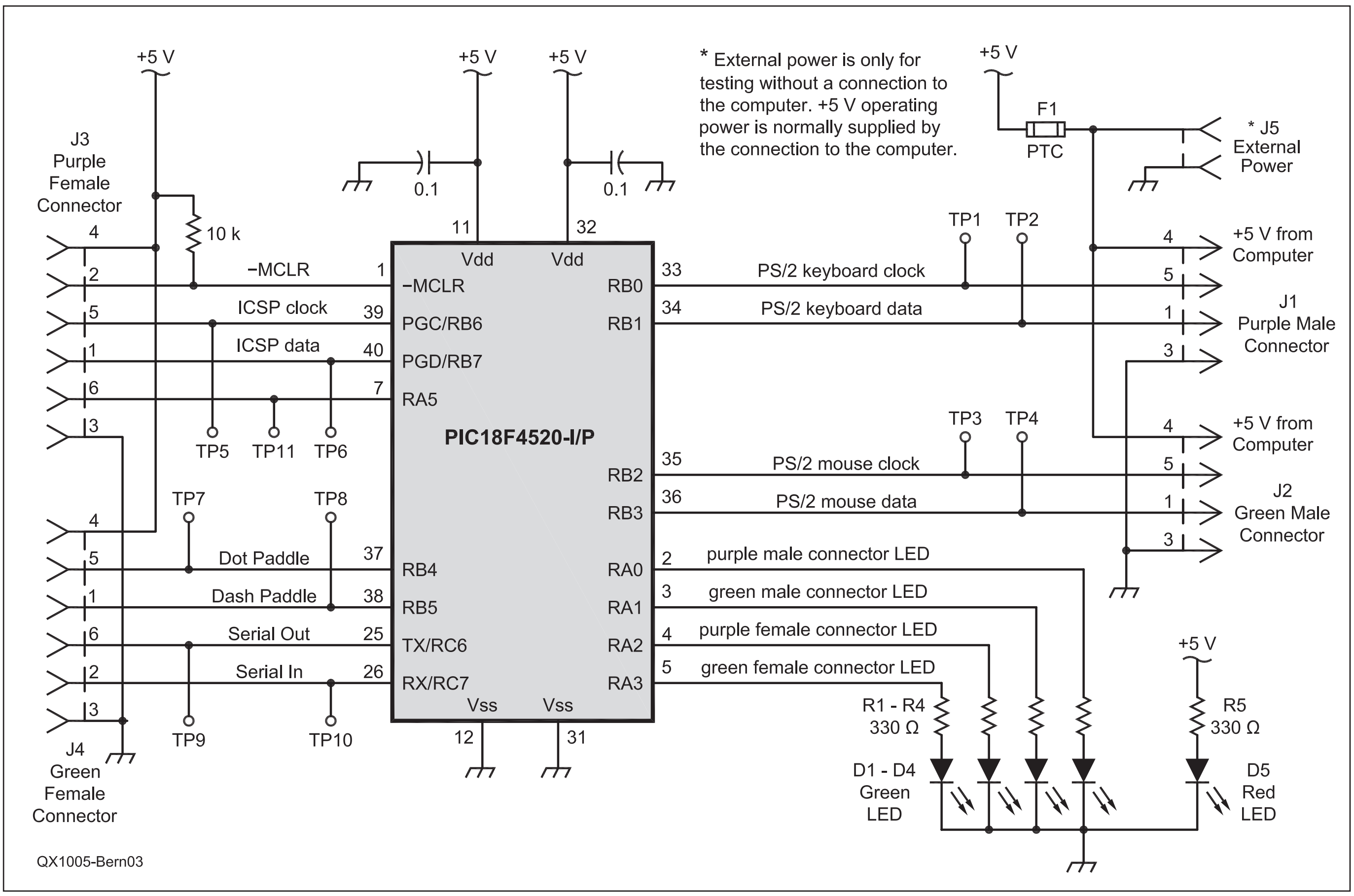

Figure 3 — This is the schematic diagram of the PS/2 Keyer.

I assembled a six pin mini-DIN connector breakout board from parts obtained from SparkFun Electronics, a wonderful resource for digital electronics hobbyists.[11, 12] 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.[13] 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.[14]

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.[15] 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.[16, 17, 18] A USB version of the Olimex 40 pin bare prototyping board is also available.[19]

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.[20, 21, 22, 23] I connected the 5 V power supply to the prototyping board with a positive temperature coefficient (PTC) resettable fuse device, available from Spark Fun, at both mini-DIN connectors and at the ICD programming connector.[24] 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.[25, 26] The ICD connector is connected to a PIC programmer with a short cable. As luck would have it, the 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 PIC. 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.[27] The PICkit 2 is inexpensive and was adequate for this project. It has a convenient power management feature: it provides power to the development board if it detects that the board has no power. Also, power to the board can be turned on and off from a software menu item.

After evaluating free demonstration versions of several PIC *C* compilers, I chose the CCS *C* compiler since it appears to best hide hardware details of PIC microcontrollers with library functions.[28] The port input/output library functions are easy to use and interrupt service routines are easy to write. Also, Rick Hambly, W2GPS, recommended the CCS *C* compiler since he uses it extensively for his work.

For the actual development environment, I preferred to use the Microchip MPLAB® integrated development environment (IDE) instead of the one provided by CCS with their *C* compiler suite.[29] Note that the less expensive CCS PCH *C* compiler instead of the full CCS *C* compiler suite is sufficient for this project since PCH integrates with the MPLAB IDE.[30]

Learning PIC *C* Programming

My first goal was to repeatedly blink an LED on and off and to display "hello, world" on a *HyperTerminal* serial terminal window. I read several introductory books on PIC *C* programming to get me started.[31, 32, 33] There are several other good books to read, as well.[34, 35, 36] *Nuts & Volts* and *Circuit Cellar* magazines also have good articles on PIC microcontroller development.[37, 38]

For program testing and debugging, I learned to use LEDs and timing pulses on an output pin to observe with a logic analyzer. I used the `printf()` function to print configuration and debugging information during 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

Part	*Quantity*	*Part Name*	*Vendor*	*Part Number*
C1, C2	2	0.1 µF 50 V 10% PC-Mount Capacitor	Digi-Key	BC1148CT-ND
D5	1	Red round diffused lens LED	Digi-Key	P589-ND
R1-R5	5	330 Ω ¼ W resistor	Digi-Key	P330BACT-ND
R6	1	10 kΩ ¼ W resistor	Digi-Key	10KQBK-ND
U1	1	40 pin IC socket	Digi-Key	3M5471-ND
U1	1	PIC18F4520-I/P	Digi-Key	PIC18F4520-I/P-ND
	4	Self-Stick Rubber Feet	Digi-Key	SJ5012-0-ND
J1, J3	1	PS/2 keyboard extension cable with purple connectors	Micro Center	SKU: 133314
J2, J4	1	PS/2 mouse extension cable with green connectors	Micro Center	SKU: 133272
J4	1	¼ inch In-Line Stereo Audio Jack	RadioShack	Catalog #: 274-141
D1-D4	4	Green rectangular clear lens LED	SparkFun	SKU: COM-08532
		Microchip PICkit 2 microcontroller programmer	Digi-Key	PG164120-ND
		Saleae Logic logic analyzer	SparkFun	SKU: TOL-08938
		Pro USB Converter: USB to PS/2 Keyboard and Mouse	Micro Center	SKU: 919712

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.[39, 40] An internal timer interrupt is used to measure time between *dot*s and *dashe*s. 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

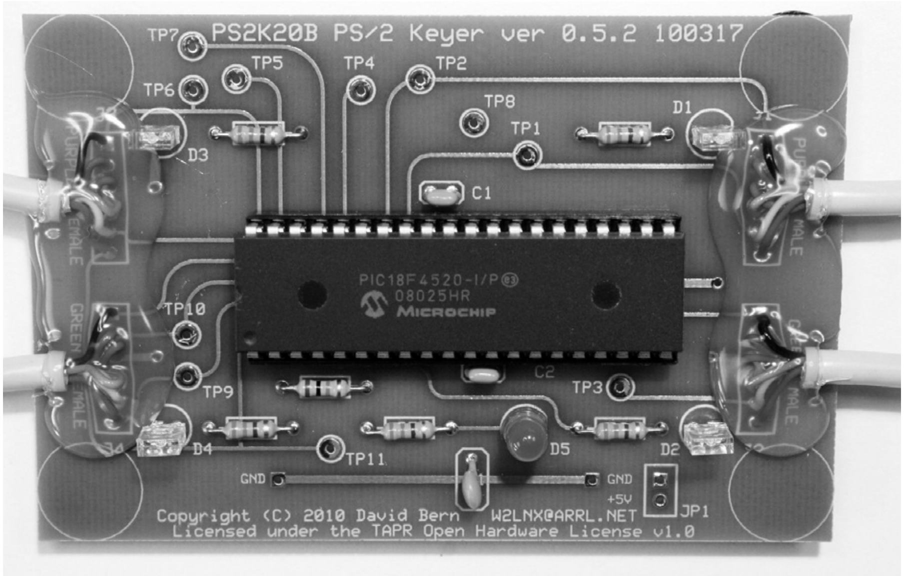

Figure 4 — On the printed circuit board, there is a green LED in a transparent plastic lens next to each cable connection. Hot glue is used to protect and secure the connections from the PS/2 cables to the circuit board. The circles in the corners indicate where rubber feet are placed under the circuit board.

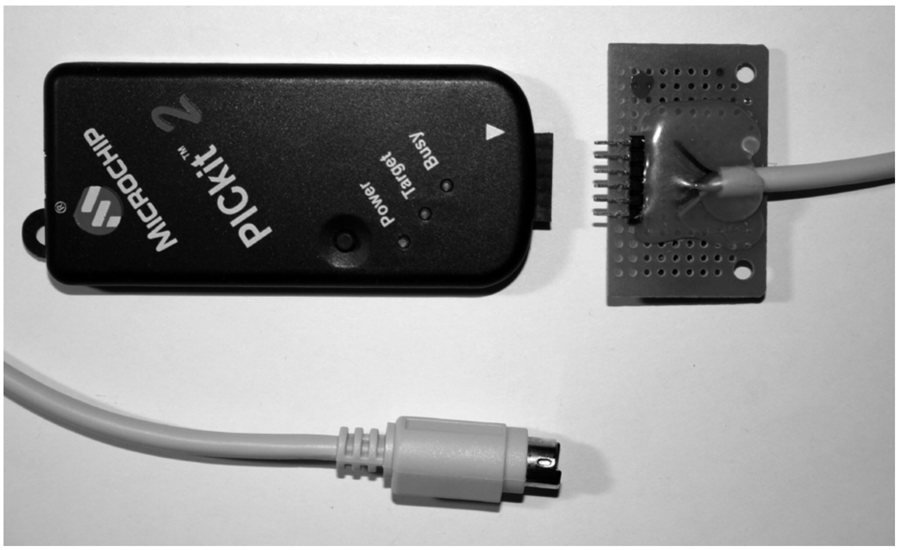

Figure 5 — This photo shows a PICkit 2 programming cable wired using the connections shown in Table 2. Hot glue protects the cable connections to a small prototyping board. The dark colored dot on the top left corner of the perf board is oriented with the white triangle on the PICkit 2.

Table 2A

PS/2 Mouse Extension Cable with Green Connectors and PS/2 Keyboard Extension Cable with Purple Connectors

Pin	*Wire Color*
1	Black
2	Violet
3	Red
4	Green
5	Yellow
6	Blue

Table 2B

Connections from the PICkit 2 Programmer Male Connector to a PS/2 Male Connector PICkit 2 Programming Cable

Pin	*PICkit 2 Male Connector*	*Pin*	*PS/2 Male Connector*
1	–MCLR	2	–MCLR
2	V_{DD}	4	V_{DD}
3	ground	3	ground
4	ICSP PGD	1	ICSP data
5	ICSP PGC	5	ICSP clock
6	not connected	6	not connected

Table 3A

Cable From Bencher CW Paddle to a ¼ Inch Stereo Phone Plug, RadioShack Catalog no.: 274-139.

Paddle Contact		*¼ Inch Stereo Phone Plug*
Dot	Left	Tip
Dash	Right	Ring
Ground		Sleeve

Table 3B

Connections Between a ¼ Inch Stereo In-Line Jack, RadioShack Catalog no.: 274-141, and a PS/2 Male Connector.

CW Paddle Cable

¼ Inch Stereo In-Line Jack		*PS/2 Male Connector*
Dot	Brass Tab	5
Dash	Silver Tab	1
Ground		3

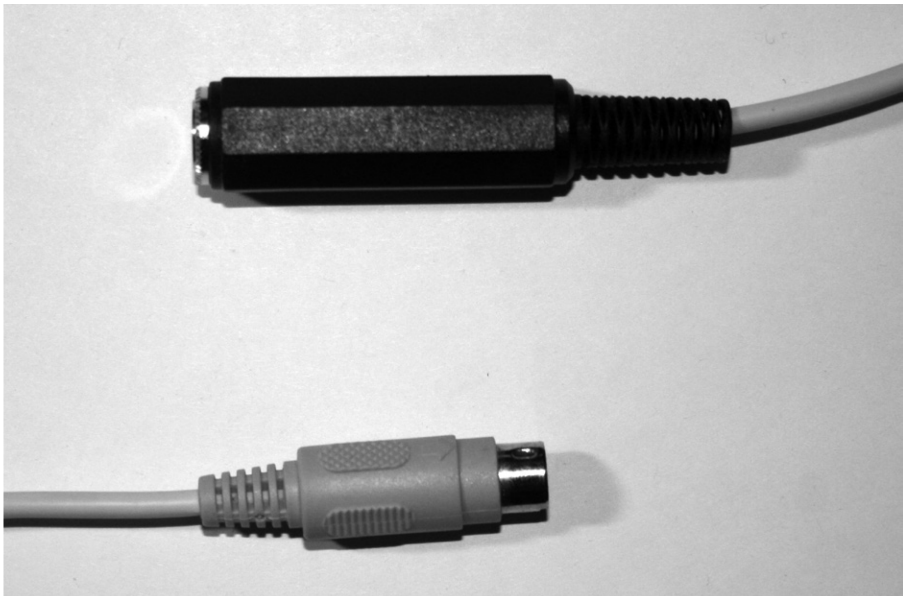

Figure 6 — This shows a CW paddle cable wired using Table 3B.

octagon. 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 device configuration command codes. This made it unnecessary to repeatedly plug and unplug the adapter USB connector in and out of the USB port on the computer. I used a logic analyzer to help my understanding of the host-to-device initial configuration dialog and for debugging my generated PS/2 keyboard scan codes and PS/2 mouse packets. I connected the logic analyzer probes to the PS/2 keyboard and the PS/2 mouse clock and data lines. Also, additional mark pulses were generated at points of interest in the code to identify logic analyzer points of interest. LEDs connected to pins on port A indicated program activity. A general status LED, an LED for the PS/2 keyboard port, and an LED for the PS/2 mouse port were used.

Building a Circuit Board

After the 2009 *Digital Communications Conference,* I designed a printed circuit board as shown in Figure 4. A PS/2 mouse extension cable with green connectors and a PS/2 keyboard extension cable with purple connectors were cut in half and attached to the circuit board with hot glue. The male green and purple connectors are connected to the corresponding purple and green female connectors of the PS/2 keyboard-and-mouse-to-USB adapter.

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.

Figure 6 shows a CW paddle cable. On one end of a connecting cable, I wired an in-line ¼ inch stereo phone jack and on the other end I used a PS/2 male connector, wired according to Table 3B.

A sample program is shown in Figure 7. It continuously blinks all the port LEDs. This program is useful for doing a basic test of the circuit board after its construction.

Future Enhancements

The current version uses a PS/2 keyboard-and-mouse-to-USB adapter to connect the PIC18F4520 microcontroller to the host computer. My next version will use a PIC18F4550 microcontroller, since it contains an internal USB interface, eliminating the need for a PS/2-to-USB adapter.

Availability

The source code and the Express PCB circuit board files for this project are available for download from the ARRL *QEX* Web page.[41] The source code is released under the GNU General Public License, version 2 (GPLv2); the files for the printed circuit board are released under The TAPR Open Hardware License Version 1.0 (May 25, 2007).

Conclusion

This is an interesting and enjoyable project. I hope that this article inspires you to start a Microchip PIC microcontroller programming project. You can incrementally develop a microcontroller project of moderate complexity by systematically testing at every step of the way. Microchip PIC microcontrollers, programming tools and software development tools are readily available and are relatively inexpensive.

This project may be of possible use by people who have limited physical mobility who cannot use a conventional keyboard or a mouse. This could allow them to type on a computer or to manipulate a mouse pointer using just two switch contacts. Coupled with a "Puff and Sip Key," described in the March 2004 issue of *QST,* the PS/2 keyer could allow hands free operation of a computer.[42]

Acknowledgements

I would like to thank the following people for helping me with this project in some way: Adam Bern, KB3KVD, for being my keyer paddle mouse tester, Jim Johns, KAØIQT, for enlightening conversations about Microchip tools, Joe Julicher, N9WXU, for suggesting the PICkit 2 programmer, Larry Wolfgang, WR1B, for convincing me to work on this project, Milt Cram, W8NUE, and George Heron, N2APB, for developing their NUE-

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

[1]Milton Cram, W8NUE, and George L. Heron, N2APB, "NUE-PSK31: A digital modem for PSK31 Field Operation...Without Using a PC!," 2007 *TAPR and ARRL 26th Digital Communications Conference Proceedings*, Hartford, Connecticut: ARRL.

[2]International Morse code, "RECOMMENDATION ITU-R M.1677," International Telecommunication Union, 2004; available at **www.godfreydykes.info/international morse code.pdf**

[3]Morse code, Wikipedia, The Free Encyclopedia; **http://en.wikipedia.org/wiki/Morse_code**

[4]Keyboard 101- and 102-Key, *IBM Personal System/2 Hardware Interface Technical Reference — Common Interfaces*, 1990; **www.mcamafia.de/pdf/pdfref.htm**

[5]Keyboard and Auxiliary Device Controller, *IBM Personal System/2 Hardware Interface Technical Reference — Common Interfaces*, 1990, **www.mcamafia.de/pdf/pdfref.htm**

[6]Adam Chapweske, *The PS/2 Mouse/Keyboard Protocol*, 2003; **www.computer-engineering.org/ps2protocol/**

[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/**

[9]Jochen Jahn, "Keyboard-Game Interface," *Nuts & Volts Magazine*, Dec 2009, pp 30-34.

[10]Adam Chapweske, *Keyboard Scan Codes: Set 2*; **www.computer-engineering.org/ps2keyboard/scancodes2.html**

[11]SparkFun Electronics, MiniDIN 6-Pin Connector, SKU: PRT-08509; **www.sparkfun.com/commerce/product_info.php?products_id=8509**

[12]SparkFun Electronics, MiniDIN 6-Pin Connector Breakout, SKU: PRT-08651; **www.sparkfun.com/commerce/product_info.php?products_id=8651**

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

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

[15]CCS, Inc., PIC18F4520 Development Kit; **www.ccsinfo.com/product_info.php?products_id=18F452kit**

[16]Microchip Technology Inc., PICDEM 2 Plus, Part Number: DM163022; **www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010072**

[17]Sure Electronics Co, Hybrid of PIC18F4520 Dem2PLUS and Low Power Demo Board; **www.sure-electronics.com/product/goods.php?id=24** and **www.sureelectronics.net/goods.php?id=24**

[18]SparkFun Electronics, 40 Pin PIC Development Board, SKU: DEV-00021; **www.sparkfun.com/commerce/product_info.php?products_id=21**

[19]SparkFun Electronics, 40 Pin PIC

```c
#include <18F4520.h>
#use delay(internal=8MHZ)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#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);

/* --------------------------------------------------- *
 * this program continuously blinks all the port LEDs *
 * --------------------------------------------------- */

void main(void)
{
  delay_ms(DELAY);
  printf("Figure 7 sample program... hello\r\n");

  for (;;) {
    blink_port_LEDs();
  }
}


/* --------------------------------- *
 * this routine blinks each port LED *
 * --------------------------------- */

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(PORT3_LED); delay_ms(DELAY);
  output_low(PORT3_LED); delay_ms(DELAY);

  output_high(PORT2_LED); delay_ms(DELAY);
  output_low(PORT2_LED); delay_ms(DELAY);
}
```

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

Development Board with USB, SKU: DEV-00022; **www.sparkfun.com/commerce/product_info.php?products_id=22**
[20]Beginner Electronics, Breadboard and Wire Kit; **www.beginnerelectronics.com/beginner/Products.php**
[21]Microchip Technology Inc, PIC18F2420/2520/4420/4520 Data Sheet: 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nano-watt Technology (document DS39631E), 2008; **www.microchip.com**
[22]SparkFun Electronics, Adapter board for Microchip ICD and ICD2; **www.sparkfun.com/commerce/product_info.php?products_id=193**
[23]HVW Technologies, RS-232 Driver Module — DCE; **www.hvwtech.com/products_view.asp?ProductID=289**
[24]SparkFun Electronics, PTC Resettable Fuse, SKU: COM-08357; **www.sparkfun.com/commerce/product_info.php?products_id=8357**
[25]Inland Pro USB Converter USB to PS/2 Keyboard and Mouse, SKU: 919712; **www.microcenter.com/single_product_results.phtml?product_id=0230515**
[26]Cable Club, 6 ft PS/2 Keyboard and Mouse Interface Cable (Male/Male), part: BC20277-6; **www.cableclub.com/keyboard-mouse-interface-cable-malemale-p-797.html**
[27]Microchip Technology Inc, PICkit 2 Development Programmer/Debugger; **www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805**
[28]CCS, Inc, Compiler Exclusively for Microchip PIC® MCUs, **www.ccsinfo.com/content.php?page=compilers**
[29]Microchip Technology Inc, MPLAB Integrated Development Environment; **www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002**
[30]CCS, Inc, PCH Command-line Compiler for PIC18 MCU parts; **www.ccsinfo.com/product_info.php?cPath=Store_Software&products_id=PCH_full**
[31]Benson, David, C *What Happens, Using PIC® Microcontrollers and the CCS* C *Compiler*, Hayden, Idaho; Square 1 Electronics, 2008; **www.sq-1.com/cwhtoc.html**
[32]Chuck Hellebuyck, *Beginner's Guide To Embedded* C *Programming: Using the PIC® Microcontroller and the HITECH PICC-Lite™* C *Compiler*, Milford, Michigan; Electronic Products, 2008, ISBN: 978-1438231594; **www.elproducts.com/embeddedcbook.htm**
[33]Martin P Bates, *Programming 8-bit PIC Microcontrollers in* C*: with Interactive Hardware Simulation*, Oxford, UK: Newnes Press, 2008; ISBN: 9780750689601
[34]Chuck Hellebuyck, *Beginner's Guide to Embedded* C *Programming — Volume 2: Timers, Interrupts, Communication, Displays and More*, Milford, Michigan: Electronic Products, 2009; ISBN: 978-1448628148, **www.elproducts.com/embeddedcbook2.htm**
[35]Dogan Ibrahim, *Advanced PIC Microcontroller Projects in* C*: From USB to RTOS with the PIC 18F Series*, Oxford, UK: Newnes Press, 2008; ISBN: 978-0750686112
[36]Richard H. Barnett, Sarah Cox, Larry O'Cull, *Embedded* C *Programming and the Microchip PIC*, Clifton Park, New York: Delmar Learning, 2004; ISBN: 978-1401837488
[37]*Nuts & Volts Magazine, The Magazine for the Electronics Hobbyist*, ISSN 1528-9885; **www.nutsvolts.com/**
[38]*Circuit Cellar, The Magazine for Computer Applications***,** ISSN 1528-0608; **www.cuitcellar.com/**
[39]Jack G. Ganssle, *A Guide to Debouncing*, Rev 3: June, 2008; **www.ganssle.com/debouncing.pdf**
[40]CW Touchkeyer touch paddle, model P1PADW; **www.cwtouchkeyer.com/P1PADW.htm**
[41]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**.
[42]Gary Gordon, K6KV, "Build a Puff-and-Sip Key," March 2004 *QST*, pp 31-32.

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.

QEX