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

David Bern, W2LNX
Montgomery College, Rockville, Maryland
w2lnx@arrl.net

This paper describes my experience learning to program a Microchip PIC® microcontroller in C. The program for this project emulates a PS/2 keyboard and a PS/2 mouse using CW keyer paddles for input.

INTRODUCTION

In the past several years, I became interested in learning how to program Microchip PIC microcontrollers and I began to look for an interesting project. As an experienced C programmer, 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 LEDs. And, importantly, it needed to be written in C.

At the 26th DCC, Milt, W8NUE and George, N2APB introduced their NUE-PSK digital modem [Cram 2007]. The NUE-PSK is a small device that provides portable PSK31 operation without the use of a personal computer. However, it does require a PS/2 keyboard for entering text. It occurred to me that it could be more portable if the large PS/2 keyboard is replaced with CW keyer paddles. A PIC microcontroller would translate CW input sent on the paddles into output that comes out of 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 that emulates a PS/2 keyboard using keyer paddles for input. Later on in the project, I wondered if it was possible to emulate a PS/2 mouse with only two switches and no other moving parts.

BACKGROUND

Morse code

Morse code input is defined in the ITU recommendation on the international Morse code [M.1677] and is further described in an article in Wikipedia [Wiki M]. The CW character and word timings needed for this project are the length of time of a *dah* relative to a *dit*, the length of time between *dit*s and *dah*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 uses the PS/2 protocol and was originally described in the IBM PS/2 computer reference manuals [PS/2 k] and [PS/2 m], respectively. Articles on the Web about the PS/2 protocol [Chap p], the PS/2 keyboard protocol [Chap k], and the PS/2 mouse protocol [Chap m] were most helpful.

The PS/2 interface on personal computers with six pin mini-DIN keyboard and mouse connectors uses the PS/2 protocol. This is a two way synchronous protocol used to communicate between a host and a device [Chap p]. 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 five volt 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 a one stop bit. The device always generates the clock signal. The host can request to communicate with the device. The device responds by generating the clock signal for the host to send its data to the device. Then, the device acknowledges that it received the complete data from the host.

The PS/2 keyboard protocol

Each key on a PS/2 keyboard device is identified by its unique scan code. Scan code set 2 [Chap s2] is commonly used today and it was originally developed by IBM for the AT keyboard. The keyboard device sends the host the scan code of a key that is pressed; a break code and its scan code is also sent when that key is released. The host computer can communicate with the keyboard device. For example, the host lights the keyboard 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 a good 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. I assembled a six pin mini-DIN connector breakout board from parts [Spark 1] [Spark 2] 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 [Saleae]. Since the PS/2 data bits are sent in “reverse” order, i. e. least significant bit first, I created a 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 [USBee] 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, 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 [CCS 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 [PICDEM], the Dem2PLUS demonstration board from Sure Electronics [Dem2PLUS], and the Olimex 40 pin bare prototyping board with RS-232 [Spark 4] from SparkFun. A USB version of the Olimex 40 pin bare prototyping board [Spark 5] is also available.

I assembled a prototyping board by using a breadboard [bread] purchased from Beginner Electronics, a PIC18F4520 [18F4520] microcontroller in the PDIP form factor, two six pin mini-DIN connectors, an ICD programming connector [Spark 3] and a serial TTL to RS-232 adapter [HVW Tech]. Five volt power was connected to the prototyping board with positive temperature coefficient (PTC) resettable fuse devices [Spark 6] 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.

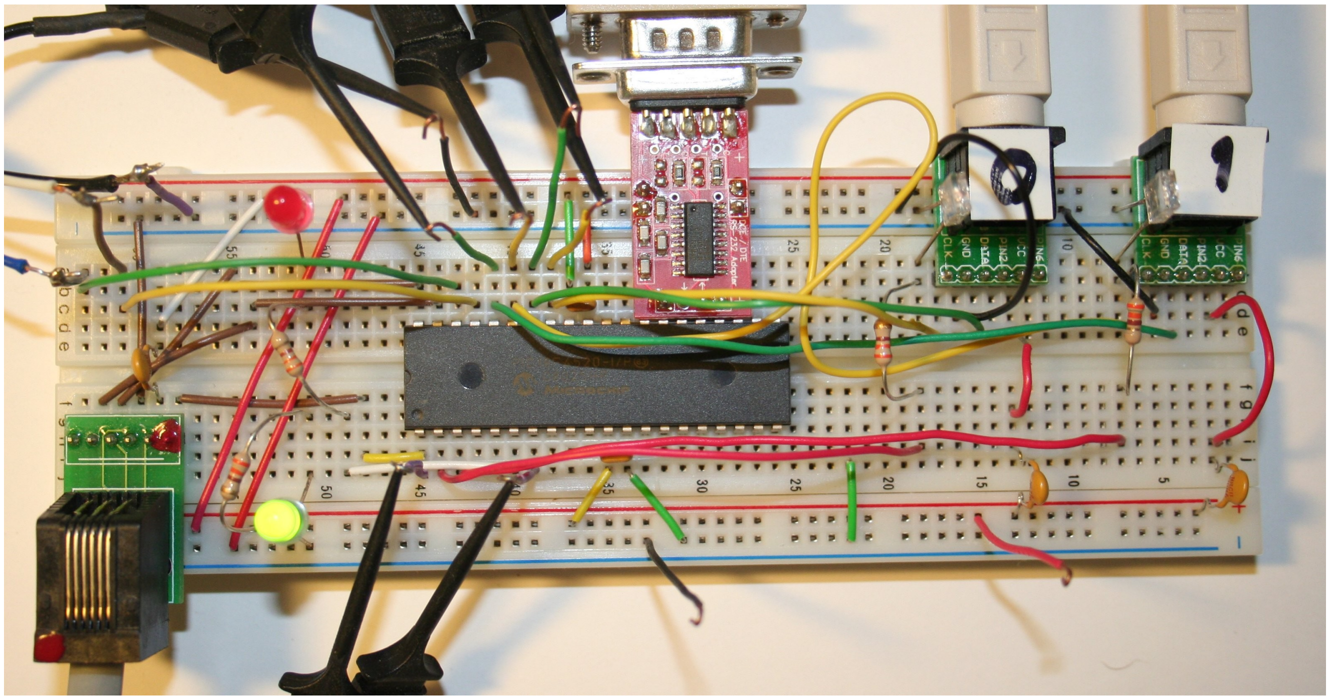

Figure 1. my development breadboard

In Figure 1, the two mini-DIN connectors are connected to a personal computer using a PS/2 keyboard and mouse to USB adapter [Inland] and two PS/2 male-to-male cables [PS/2 m/m]. 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 matches the corresponding input/output pins on the PIC18F4520. Note the probes connected to the logic analyzer.

The circuit in Figure 2 consists of essentially one component, the PIC18F4520 microcontroller. The PS/2 keyboard connector, the PS/2 mouse connector, and the CW keyer paddles are 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. Not shown are the test LEDs and the usual 0.1 uF power bypass capacitors connected between the V_{DD} and V_{SS} power pins of the PIC. Further wiring details are provided in the wiring list in the appendix.

PIC development tools

For the PIC programmer, I used the Microchip PICkit™ 2 programmer/debugger [PICkit]. 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.

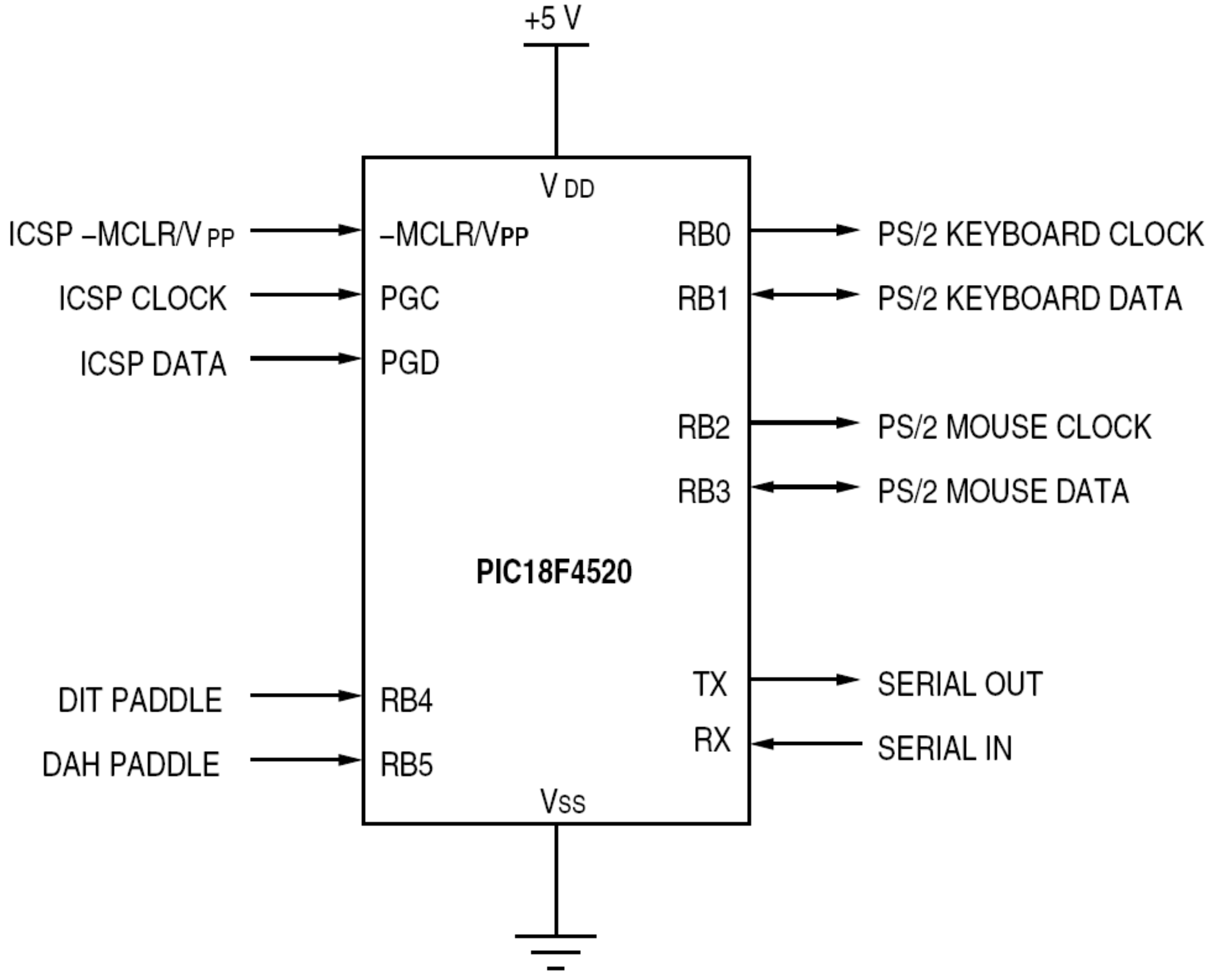

Figure 2: abbreviated schematic. Test LEDs and power supply bypass capacitors are not shown.

After evaluating free demonstration versions of several PIC C compilers, I chose the CCS C [CCS C] compiler since it appears to best hide hardware details of PIC microcontrollers with with library functions. The port input/output library functions are easy to use and interrupt service routines are easy to write. Also, Rick, 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) [MPLAB] instead of the one provided by CCS with their C compiler suite. Note that the less expensive CCS PCH C compiler [CCS PCH] instead of the full CCS C compiler suite is sufficient for this project since PCH integrates with the MPLAB IDE.

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 [Benson 2008] [Helle 2008] [Bates 2008] on PIC C programming to get me started. [Helle 2009] [Ibrahim 2008] [Barnett 2004] are other good books to read. Also [N & V] and [Celler] have good articles on PIC microcontroller development.

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 **`printf()`** to print configuration and debugging information during program development and testing. And, I 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 CW keyer paddles. The invalid CW character *dih dih dah dah* is used as a command code character. Some command codes are keyboard Enter, keyboard Caps Lock and switch to mouse mode. Entering six or more *dits* in a row 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 *dit* paddle and *dah* paddles are connected to pins RB4 and RB5, respectively, of port B to take advantage of the provided 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 keyer paddle is pressed. Keyer paddle switch contact bounce was probably most challenging problem of this program. A paper on switch bounce [Ganssle 2008] convinced me that CW keyer paddles without any switch contacts such as the CW Touchkeyer paddles [P1PADW] are necessary for this project. An internal timer interrupt is used to measure time between *dit*s and *dah*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 on 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 keyer paddle input. Clicking, i. e. briefly pressing, the left keyer paddle generates a left mouse button click and, correspondingly, clicking the right keyer paddle generates a right mouse button click. Clicking both paddles simultaneously generates a middle mouse button click. Mouse pointer movement is controlled by pressing and holding the keyer paddles: the left paddle controls the left-right mouse pointer movement; the right paddles controls the up-down mouse pointer movement. Pressing both paddles 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 octagon. Pressing for longer than about three seconds causes the mouse pointer to move faster on the screen. A command code, i. e. two left clicks followed by two right clicks, switches the program back to keyboard mode. 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. A logic analyzer was used for understanding the host to device initial configuration dialog and for debugging my generated PS/2 keyboard scan codes and PS/2 mouse packets. Logic analyzer probes were connected 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 output 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.

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 using a PIC18F4550 will eliminate the need of the PS/2 to USB adapter since it contains an internal USB interface.

Availability

I am releasing the source code for this project using the GNU General Public License, version 2 (GPLv2). Please contact me at **`w2lnx@arrl.net`** to obtain a copy of the source code and additional documentation.

CONCLUSION

This was an interesting and fun project. I hope that this paper inspires and motivates others to get started with a Microchip PIC microcontroller project. A microcontroller project of modest complexity can be developed incrementally by systematically testing at every step of the way. Microchip PIC microcontrollers, programing tools and software development tools are readily available and are relatively inexpensive.

This project may be of possible use to people who have limited physical mobility and 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 switches.

ACKNOWLEDGEMENTS

Adam Bern, KB3KVD, for being my CW keyer paddles mouse tester
Jim Johns, KA0IQT, 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
Steve Bible, N7HPR, for encouraging me to write up this project for the next DCC

July 31, 2009

APPENDIX: WIRING LIST

PIC18F4520-I/P				
pin number	**pin name**	**connected to**	**pin name**	**function**
1	-MCLR/VPP	PICkit 2 ICSP	VPP/-MCLR	ICSP program and reset
39	PGC	PICkit 2 ICSP	ICSPCLK	ICSP clock
40	PGD	PICkit 2 ICSP	ICSPDAT	ICSP data
		PICkit 2 ICSP	VDD Target	+5 V
		PICkit 2 ICSP	VSS	ground
33	RB0	keyboard PS/2	CLK	PS/2 keyboard clock
34	RB1	keyboard PS/2	DATA	PS/2 keyboard data
		keyboard PS/2	VCC	+5 V
		keyboard PS/2	GND	ground
35	RB2	mouse PS/2	CLK	PS/2 mouse clock
36	RB3	mouse PS/2	DATA	PS/2 mouse data
		mouse PS/2	VCC	+5 V
		mouse PS/2	GND	ground
37	RB4	*dit* paddle		
38	RB5	*dah* paddle		
		paddle ground		ground
25	TX	RS-232 DCE	<-	serial output
26	RX	RS-232 DCE	->	serial input
		RS-232 DCE	+	+5 V
		RS-232 DCE	-	ground
2	RA0			PS/2 keyboard LED
3	RA1			PS/2 mouse LED
6	RA4			status LED
7	RA5			mark pin
11	VDD			+5 V
32	VDD			+5 V
12	VSS			ground
21	VSS			ground
note: each LED is connected in series to a 330 ohm resistor				

REFERENCES and RESOURCES

[18F4520] Microchip Technology Inc., **PIC18F2420/2520/4420/4520 Data Sheet: 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology** (document DS39631E), 2008, **`http://www.microchip.com`**

[Barnett 2004] Barnett, Richard H., Sarah Cox, Larry O'Cull, **Embedded C Programming and the Microchip PIC**, Clifton Park, New York: Delmar Learning, 2004, ISBN: 9781401837488

[Bates 2008] Bates, Martin P., **Programming 8-bit PIC Microcontrollers in C: with Interactive Hardware Simulation**, Oxford, UK: Newnes Press, 2008, ISBN: 9780750689601

[Benson 2008] Benson, David, **C What Happens, Using PIC® Microcontrollers and the CCS C Compiler**, Hayden, Idaho: Square 1 Electronics, 2008, **`http://www.sq-1.com/cwhtoc.html`**

[bread] Beginner Electronics, **Breadboard and Wire Kit**, **`http://www.beginnerelectronics.com/beginner/Products.php`**

[CCS C] CCS, Inc., **Compiler Exclusively for Microchip PIC® MCUs**, **`http://www.ccsinfo.com/content.php?page=compilers`**

[CCS kit] CCS, Inc., **PIC18F4520 Development Kit**, **`http://www.ccsinfo.com/product_info.php?products_id=18F452kit`**

[CCS PCH] CCS, Inc., **PCH Command-line Compiler for PIC18 MCU parts**, **`http://www.ccsinfo.com/product_info.php?cPath=Store_Software&products_id=PCH_full`**

[Cellar] **Circuit Cellar, The Magazine for Computer Applications,** **`http://www.circuitcellar.com/`**

[Chap k] Chapweske, Adam, **The PS/2 Keyboard Interface**, 2003, **`http://www.computer-engineering.org/ps2keyboard/`**

[Chap m] Chapweske, Adam, **The PS/2 Mouse Interface**, 2003, **`http://www.computer-engineering.org/ps2mouse/`**

[Chap p] Chapweske, Adam, **The PS/2 Mouse/Keyboard Protocol**, 2003, **`http://www.computer-engineering.org/ps2protocol/`**

[Chap s2] Chapweske, Adam, **Keyboard Scan Codes: Set 2**, **`http://www.computer-engineering.org/ps2keyboard/scancodes2.html`**

[Cram 2007] Cram, Milton, W8NUE, and George L. Heron, N2APB, **NUE-PSK31: A digital modem for PSK31 field operation...without using a PC!**, *TAPR and ARRL 26th Digital Communications Conference 2007 Proceedings*. Hartford, Connecticut: ARRL.

[Cram 2008] Cram, Milton, W8NUE, and George L. Heron, N2APB, **NUE-PSK Digital: Modem Enables PSK31 field operation... without using a PC!**, *QEX*, March/April 2008. Newington, Connecticut: ARRL.

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

[Ganssle 2008] Ganssle, Jack G., **A Guide to Debouncing**, Rev 3: June, 2008, **`http://www.ganssle.com/debouncing.pdf`**

[Helle 2008] Hellebuyck, Chuck, **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, **`http://www.elproducts.com/embeddedcbook.htm`**

[Helle 2009] Hellebuyck, Chuck, **Beginner's Guide to Embedded C Programming - Volume 2: Timers, Interrupts, Communication, Displays and More**, Milford, Michigan: Electronic Products, 2009, ISBN: 9781448628148, **`http://www.elproducts.com/embeddedcbook2.htm`**

[HVW Tech] HVW Technologies, **RS-232 Driver Module - DCE**, **`http://www.hvwtech.com/products_view.asp?ProductID=289`**

[Ibrahim 2008] Ibrahim, Dogan, **Advanced PIC Microcontroller Projects in C: From USB to RTOS with the PIC 18F Series**, Oxford, UK: Newnes Press, 2008, ISBN: 9780750686112

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

[M.1677] **International Morse code**, RECOMMENDATION ITU-R M.1677, International Telecommunication Union, 2004, **`http://www.godfreydykes.info/international%20morse%20code.pdf`**

[MPLAB] Microchip Technology Inc., **MPLAB Integrated Development Environment**, **`http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002`**

[N & V] **Nuts & Volts Magazine, The Magazine for the Electronics Hobbyist**, **`http://www.nutsvolts.com/`**

[P1PADW] **CW Touchkeyer touch paddles**, model P1PADW, **`http://www.cwtouchkeyer.com/P1PADW.htm`**

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

[PICkit] Microchip Technology Inc., **PICkit 2 Development Programmer/Debugger**, **`http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805`**

[PS/2 k] **Keyboard 101- and 102-Key**, *IBM Personal System/2 Hardware Interface Technical Reference - Common Interfaces*, 1990, **`http://www.mcamafia.de/pdf/pdfref.htm`**

[PS/2 m] **Keyboard and Auxiliary Device Controller**, *IBM Personal System/2 Hardware Interface Technical Reference - Common Interfaces*, 1990, **`http://www.mcamafia.de/pdf/pdfref.htm`**

[PS/2 m/m] Cable Club, **6 Ft PS/2 Keyboard & Mouse Interface Cable (Male/Male)**, part: BC20277-6, **`http://www.cableclub.com/keyboard-mouse-interface-cable-malemale-p-797.html`**

[Saleae] Saleae LLC, **Saleae logic analyzer**, **`http://www.saleae.com/logic/`**

[Spark 1] SparkFun Electronics, **MiniDIN 6-Pin Connector**, SKU: PRT-08509, **`http://www.sparkfun.com/commerce/product_info.php?products_id=8509`**

[Spark 2] SparkFun Electronics, **MiniDIN 6-Pin Connector Breakout**, SKU: PRT-08651, **`http://www.sparkfun.com/commerce/product_info.php?products_id=8651`**

[Spark 3] SparkFun Electronics, **Adapter board for Microchip ICD and ICD2**, **`http://www.sparkfun.com/commerce/product_info.php?products_id=193`**

[Spark 4] SparkFun Electronics, **40 Pin PIC Development Board**, SKU: DEV-00021, **`http://www.sparkfun.com/commerce/product_info.php?products_id=21`**

[Spark 5] SparkFun Electronics, **40 Pin PIC Development Board with USB**, SKU: DEV-00022, **`http://www.sparkfun.com/commerce/product_info.php?products_id=22`**

[Spark 6] SparkFun Electronics, **PTC Resettable Fuse**, SKU: COM-08357, **`http://www.sparkfun.com/commerce/product_info.php?products_id=8357`**

[USBee] CWAV, Inc., **USBee SX logic analyzer**, **`http://www.usbee.com/sx.html`**

[Wiki M] **Morse code**, *Wikipedia, The Free Encyclopedia*, **`http://en.wikipedia.org/wiki/Morse_code`**