

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

Duane Perkins

Make the SmartPIC Serial Programmer

When picking a PIC programmer, you may find that there are as many choices as there are PICs. The more versatile it is, the more money you have to lay out, right? Not if you follow Duane's plan. He gives you more bang for your buck with his new serial programmer.



IC microcontrollers have become popular because of their speed, versatility, and low cost. There are now so many different types that it's becoming hard to keep track of them, and new types are being offered all the time. One thing they all have in common is the requirement for a programmer because program and configuration memory is contained in the chip. Many PIC programmers on the market are limited with regards to the types they will program. The less costly the programmer, the less versatile it is.

The PIC16Cxx programmer described here is a smart programmer for any serial programmed PIC with a 14-bit instruction word, including the PIC14000 with an adapter for the target socket (not PIC16C5x or PIC17Cxx devices). It requires a master program running on any computer with an RS-232 serial port configured for 9600 bps, 8-bit word, one stop bit,

and no parity. The programmer acts as a DCE slave device that communicates with the computer acting as a DTE device. *16CxxPRG.EXE* is a master program for IBM-compatible PCs. See the 16CxxPRG model A user's manual for details.

SMART FEATURES

The programmer contains an embedded PIC16C71 microcontroller (a PIC16C61 or PIC16C84 will also work) that is programmed to communicate with a master program via asynchronous serial I/O at 9600 bps. Most of the smart features are invoked by the way it communicates with the master program, including hardware handshaking. The master program must act on these signals in an appropriate manner in order to realize their full potential.

When the programmer is connected to an active serial port, it is held in a reset state until DTR is asserted. Once DTR is asserted, it waits for the computer to assert RTS. Once RTS is asserted, it asserts CTS and waits for a command. When a command is received, it returns a transmission error code if a transmission error occurred, or an invalid command error code if it does not recognize the command. If the command is recognized, it either returns a response immediately and waits for further input, or performs the function specified by the command and returns a response. The master program can abort any function at any time by dropping DTR.

One command tells the programmer to identify itself. It responds with an ASCII "A", which is its model designation. The master program should always check to be sure it is communicating with a model A programmer, because a later model (which could be developed) will probably not work correctly.

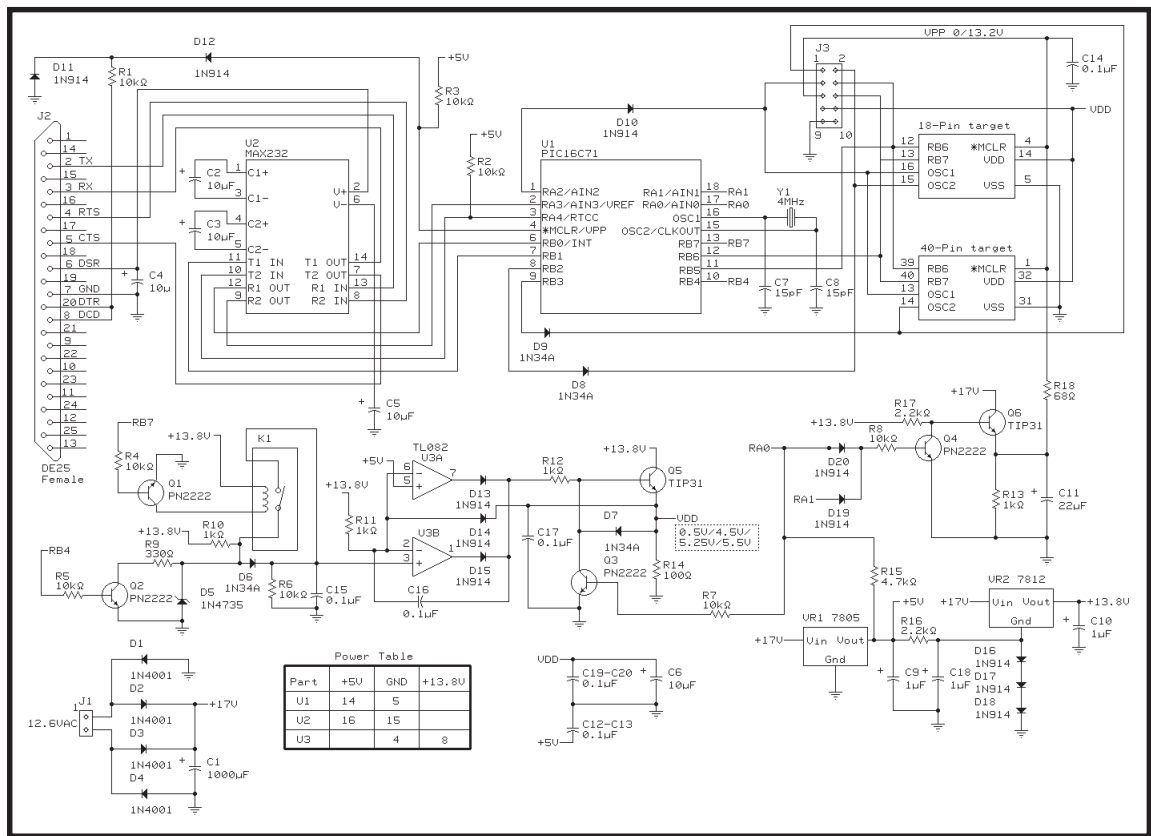


Figure 1—The upper half of the schematic is the digital circuit. The lower half is the voltage control circuit.

The programmer has two sockets for target devices. When told to accept user program and EEPROM data size parameters, the programmer tests to see if a device is in either target socket. If no device or two devices are found, the programmer responds with an error code.

V_{pp} will be applied to the target sockets only when a command is received that tells the programmer to read or program a target device. V_{pp} cannot be applied unless V_{DD} is applied. V_{DD} will be applied prior to V_{pp} and when testing for devices in the target sockets.

When commanded to read or program a target device, no voltages are applied to the target sockets until the master program drops RTS. No voltages are applied to the target sockets when the programmer is powered up, even if it is not connected. It is safe to put a device into either or both sockets at any time. Note: V_{DD} is actually at about 0.5 V when not applied. A target device will not be harmed by this low voltage and it can be safely grounded.

There are three possible levels for V_{DD} —4.5, 5.2, or 5.5 V. The master

program can set the level high (5.5 V) or low (4.5 V) for reading. The level is automatically set to 5.2 V when programming in conformance with Microchip specifications that require V_{DD} to be in the range of 4.75 to 5.25 V. The V_{DD} levels may deviate slightly from those nominal voltages but will be within the limits of Microchip's minimum and maximum specifications.

Hardware handshaking coordinates the computer and the programmer during transfer of data in either direction. The programmer utilizes half-duplex, interrupt-driven serial I/O with a one-character buffer and will accept a character from the computer only when it has asserted CTS. The programmer waits until RTS is asserted (up to 2.3 s) before sending a character. There are no unnecessary delays while the computer or programmer is marking time to give the other time to perform its task.

When programming a non-EEPROM device, words expected to remain blank in either the user program memory or configuration memory are not programmed or verified. The master program can abort

the programming of user program memory when all remaining words to remain blank, saving the time that would otherwise be wasted. The master program should verify the device after programming to be sure that there are no nonblank words where there should be none. When programming an EEPROM device, all of the program memory is programmed to assure that unused words are erased. EEPROM data memory (if any) is always fully programmed and verified.

A device type code can be programmed into configuration memory at 2008h. This word can be left blank by programming it as 3FFFh. When a device is read, the configuration memory can be read first, including the device type code. The master program can then establish the device parameters based on the type code before reading user program memory.

The programmer can read or program up to 31.75 KB of words of user program memory. The programmer can also read or program up to 510 bytes of EEPROM data memory even if the program memory is not EEPROM. In the event that Microchip

develops such a device, the programmer can handle it. The programmer can read and program a device ID at 2000h to 2003h. The master program should limit the ID to four ASCII bytes no greater than 7Fh. All higher-order bits of the word should be 1s.

COMMANDS AND RESPONSE CODES

The command codes are all ASCII decimal digits (0-9). They are listed in Table 1 along with all possible responses. The response codes are also ASCII decimal digits (0-6) with the exception of the model designation, which is an ASCII "A." A transmission error will always result in a response of error code 4. An invalid command (not 0-9) will result in a response of error code 3.

THE ELECTRONICS

Figure 1 shows the schematic diagram for the programmer. External power of 12.6 VAC or 18 VDC is sup-

plied at J1. A bridge rectifier converts this to 17 VDC, which is filtered by C1. VR1 supplies 5 V to U1, U2, and other parts as shown. VR2 has three 1N914 diodes connected in series between its ground terminal and ground, which are supplied with current from R16. The output of this 12-V regulator is at 13.8 V because its ground terminal is held at 1.8 V above ground. VR2 supplies U3, Q5, K1, and other parts as shown.

U3 sets the V_{DD} level for the target sockets by establishing the voltage at the base of Q5. The actual V_{DD} level may deviate slightly from the nominal voltages that are stated in Figure 1. U3A has a reference voltage of 5 V at its noninverting pin. U3B has a reference voltage of 6.05 V (the zener voltage of D5 at low current) if K1 is energized, or 5.8 V if K1 is not energized.

The outputs of U3 are applied to the base of Q5 through D13, D15, and R12. Whichever output is higher will determine the V_{DD} level at the emitter

of Q5. There is a drop of about 0.6 V across the base-emitter junction of Q5 and also across D14, which holds the inverting inputs of U3 at approximately the voltage at the base of Q5, thus applying 100% negative feedback. C6, C15, and C16 prevent oscillation.

When V_{DD} is applied, RA0 is low and Q3 is cut off. When RB4 is low, Q2 is cut off and the V_{DD} level is high. If RB7 is low, Q1 is cut off and K1 is not energized. If RB7 is high, Q1 conducts, energizing K1, which in turn shorts D6. The output of U3B will be higher than the output of U3A, thus D13 will not conduct. Due to negative feedback, U3B will drive the base of Q5 to a voltage approximately equal to the reference voltage at its noninverting input and the emitter of Q5 will be about 0.6 V lower (5.2 or 5.5 V).

When RB4 is high, Q2 conducts. R9 pulls the noninverting input of U3B below 5 V, with the result being that its output is below the output of U3A

Command	Function	Description
0	Identify model	Response is A.
1	Configure	The programmer expects two bytes to follow when it asserts CTS. After these have been received, the programmer sends 0 if one target device is found, 5 if no target device is found, or 6 if two target devices are found. The first byte sent after the command is the high-order byte of the program memory size. The high-order bit = 1 if the device has EEPROM program memory. The second byte is one-half the EEPROM data memory size (zero if none).
2	Set V_{DD} high	Response is 0.
3	Set V_{DD} low	Response is 0.
4	Perform blank check	The programmer reads all user program memory and configuration memory from 2000h to 2008h until a nonblank work is found. Possible responses are: <ul style="list-style-type: none"> • 0, all program memory blank • 1, user program memory not blank • 2, configuration memory not blank
5	Read user program memory	The programmer sends the words read, high-order byte first, and quits when the count is equal to the user program memory size.
6	Read configuration memory	The programmer sends the words read from 2000h to 2008h, high-order byte first.
7	Program the target device	The programmer responds with 0, then expects each word to be received, high-order bit first, when it asserts CTS. If the high-order byte is 1 and the device does not have EEPROM program memory, programming of user program memory is aborted. After the count reaches the user program memory size or programming of user program memory is aborted, the programmer expects to receive nine more bytes for configuration memory. The programmer sends a response after each word. In the event of an error, programming is aborted. <p>The possible responses are:</p> <ul style="list-style-type: none"> • 0, word programmed and verified • 1, programming error
8	Read EEPROM data memory	The programmer sends the bytes read and quits when the count is Equal to the EEPROM data memory size.
9	Program EEPROM data memory	The programmer responds with 0, then expects each byte to be received when it asserts CTS. The programmer sends a response after each byte. In the event of an error, programming is aborted. <p>The possible responses are:</p> <ul style="list-style-type: none"> • 0, byte programmed and verified • 1, programming error

Table 1—All command and response codes are ASCII.

Ground	1 ** 2	Polarizing pin
VDD	3 ** 4	VDD
RB7	5 ** 6	VPP
RB6	7 ** 8	OSC1
OSC2 on 18-pin socket	9 ** 10	OSC2 on 40-pin socket

Table 2—Connections to the target PIC are shown by the pinout of J3.

and D15 does not conduct. Negative feedback forces the base of Q5 to about 5.1 V and the emitter of Q5 to 4.5 V. When RA0 is high, Q3 conducts, pulling the base of Q5 to less than 0.5 V, thus cutting off Q5. Current through R11 and D14 is shunted through D7, holding V_{DD} to about 0.5 V.

V_{PP} is controlled by Q6. When both RA0 and RA1 are low, Q4 is cut off. 13.8 V is applied to the base of Q6 through R17, resulting in 13.2 V at the emitter. When either RA0 or RA1 is high, Q4 conducts, cutting off Q6. R18 limits V_{PP} current to prevent target device latch-up.

U2 converts the TTL-level signals from U1 to RS-232 levels for the serial port lines. The positive voltage generated by the MAX232 is connected directly to the DSR line to signal that the programmer is powered. DTR is connected to DCD and, via R1 and D12, to the NOT-MCLR pin of U1.

When DTR is asserted, D12 does not conduct and 5 V is applied to NOT-MCLR by R3, allowing the PIC16C71 to operate. When DTR is dropped, it is negative and sinks current through R1, D12, and R3, dropping the voltage on NOT-MCLR to ground level, thus holding the PIC16C71 in a reset state. D11 assures that the voltage on NOT-MCLR cannot be negative by holding its cathode and that of D12 at 0.6 V below ground level.

Because the programmer is wired DCE, it sees TX as its data input line, RX as its data output line, RTS as a control input, and CTS as a control output. The master program can determine if the programmer is connected, powered or not, by testing to see if DCD follows DTR. The master program can determine if the programmer is active by checking to see if CTS follows RTS.

U1 acts as a slave that responds to valid commands from the computer. It controls the voltages applied to the

target sockets and the data and clock signals required to read or program a PIC16Cxx. Via D8, D9, and D10, it determines which sockets contain a target device.

V_{DD} is applied, followed by a high on OSC1 from RA2 through D10. RB2 and RB3 are inputs with weak pull-ups enabled. If either is pulled low through D8 or D9, a device is in the respective socket. If the device is configured for a RC oscillator (which is the case for a blank device), OSC2 will be low. If the device is not blank and is not configured for a RC oscillator, an internal inverter outputs the inverse of OSC1 on OSC2, pulling it low. After reading or programming a target device, V_{PP} is dropped, then after a delay of about 100 ms, to allow C18 to discharge, V_{DD} is dropped.

A 4-MHz crystal or ceramic resonator can be used. If a ceramic resonator with integral capacitors is used, do not use C7 and C8. The frequency must be 4 MHz to provide correct timing of various pulses and delays (assuming the PIC16C71 is programmed with *16CxxPRG.HEX*).

CONSTRUCTION

Additional board layout figures can be downloaded. After etching and drilling the main board (a #60 drill bit is a good choice), solder in all the jumpers shown as solid lines on the parts placement diagram. Use bare tinned copper wire, being sure to stretch each one tight and straight to assure that adjacent jumpers cannot touch. Solder in all remaining components, starting with the smallest (D10-D20) and proceed with resistors and the remaining diodes. Continue with the remaining components by size, leaving the largest for last. Use sockets for U1, U2, and U3.

If you plan to use an enclosure, use a 10-pin header (J3) in place of the 40-pin target socket and solder in jumpers where shown by the dashed lines.

For testing purposes, the 18-pin target socket should be soldered in. C19 and C20 are mounted on the solder side and directly to the pins of the target sockets. Solder over unused holes that might otherwise result in a break in the trace.

Be sure all solder joints are good and that there are no solder bridges (solder joints are not always as good as they look). Cleaning off the rosin after soldering is a nice touch and aids visual inspection. Acetone works well, but don't immerse the board.

TESTING

When making the following tests, remove power before changing the test setup. With no ICs in the sockets, apply 12.6 VAC or 18 VDC to J1. Test for 17 V across C1, 13.8 V at pin 8 of U3, 5.8 V at pin 3 of U3, and 5 V at pins 4 and 14 of U1, pin 16 of U2, and pin 5 of U3. You should find about 0.5 V at pin 14 of the 18-pin target socket and pins 11 and 32 of the 40-pin target socket. Pin 4 of the 18-pin target socket and pins 12 and 31 of the 40-pin target socket should be at ground potential (not floating).

Place jumper wires between pins 5 and 17, and between pins 14 and 18 of the U1 socket. Put a TL082 in the U3 socket and test for 5.2 V at pin 14 of the 18-pin target socket and pins 11 and 32 of the 40-pin target socket. Pin 4 of the 18-pin target socket and pins 12 and 31 of the 40-pin target socket should remain at ground potential. Remove the jumper wire from pins 14 and 18. Pin 4 of the 18-pin target socket and pins 12 and 31 of the 40-pin target socket should now measure 13.2 V. Place a jumper wire between pins 14 and 13 of U1. V_{DD} , measured anywhere convenient, should be 5.5 V. Change the jumper to pins 14 and 10. V_{DD} should now measure 4.5 V. Note: slight variances from the V_{DD} levels stated above are acceptable, but V_{DD} should not exceed 5.25 V when programming.

Put a MAX232 in the U2 socket. Pin 6 of J2 (DSR) should measure about +10 V. Pins 3 (RX) and 5 (CTS) should measure about -10 V. Pin 8 (DCD) and pin 20 (DTR) should measure about 4.4 V. Pin 7 should be at

ground potential (not floating).

Put a programmed PIC16C71 in the U1 socket. Pin 4 should measure 5 V, and pin 15 should measure about 2.5 V (indicating oscillation). Connect the programmer to a PC with an RS-232 cable (the cable must have TX, RX, RTS, CTS, DSR, GND, DCD, and DTR lines). Run a terminal emulator program on the PC. Set up for 9600 bps, 8-bit word, one stop bit, and no parity for the port in use. The terminal emulation program must set the IRQ in accordance with the actual hardware configuration. DTR and RTS must be asserted.

Press the 0 key. The programmer should respond with "A." Press 2 then 3. The programmer should respond with "0" after each keypress. Press 1 three times in quick succession. The programmer should respond with "5" after the last keypress. Press 4, 5, 6, and 7. The programmer should not respond. Press any other keys. The programmer should respond with "3" after each keypress. If all these results are obtained, the programmer is basically working.

RUNNING 16CXXPRG

Now is the time for the final testing. Type *FINDPORT A* at the command line. If the programmer is connected and powered, the port it is connected to should be found and displayed. *16CxxPRG.PRT* will be created with one line containing *X:YYY COMZ*, where *X* is the IRQ, *YYY* is the port hex address, and *Z* is the COM port number (1-4). Note that if you are using a port not recognized by DOS (other than 3F8, 2F8, 3E8, or 2E8), or an IRQ other than 2, 3, 4, or 5, you must create *16CxxPRG.PRT* with the parameters for the port. Now run *16CxxPRG*. The display should be as described in the *16CxxPRG* user's manual, and you are ready to start programming PIC16Cxxs.

HOUSING THE PROGRAMMER

For day-in-day-out use, it is nice to have the programmer in an enclosure. A 5.25" floppy drive enclosure is eminently suitable. You will also need a 3.5" floppy drive adapter kit. The

parts list is available for downloading so you can check for a source. The small board (shown in the downloadable figures) will fit in the bezel supplied with the adapter kit.

The cutout for a floppy drive will have to be enlarged to accommodate the ZIF sockets. Cut a piece of sheet metal to serve as a base for the programmer board and a means of attaching it to the floppy drive adapter base that fits into the enclosure. The sheet should measure 5.2" x 5.4". Bend down the tabs on the adapter base so the sheet will lie flat on them, leaving enough space for the heads of sheet metal screws below the tabs and about 1.25" of headroom above the tabs.

Place the sheet on the tabs leaving adequate space in front of the enclosure for a connector cable, then mark and drill the sheet for four sheet metal screws to attach it, using four of the tabs. Align the programmer board squarely over the sheet, then mark and drill the sheet for 6-32 machine screws. Use spacers between the board and the sheet, being sure to allow enough distance so the sheet metal screws will not touch the board.

Use a 10-pin dual-inline header connector and ribbon cable to provide connections between the programmer board and the board with the ZIF sockets. Solder the ribbon cable leads directly to the pads on the solder side of the ZIF socket board. You can devise some means of attaching the cable to the board other than the solder connections (even just a dab of epoxy cement will do).

The ZIF socket board can be attached to the bezel with 6-32 hardware or, if you like, a mini toggle switch and LED holder can be used to provide for power switching and a power-on indicator. Route the leads to the switch, well away from the ZIF sockets. R19 and J4 provide a handy current source for an LED. A Radio Shack 273-1365 transformer fits nicely in the rear end of the enclosure. The pinout of the header is shown in Table 2. Cut off pin 2 and insert it in the header connector so that it cannot be connected incorrectly.

IN CONCLUSION

The design of this programmer provides the most functionality for the least cost and conforms to Microchip specifications within the limits of its intended use. The V_{DD} levels are not precisely controlled and are limited in range, but satisfy the requirements for a developmental programmer that will not be used for production programming where more stringent demands must be met. Users experimenting with PIC16Cxx microcontrollers should find it more than merely adequate.

Microchip recommends that the window of an EPROM device be covered during reading or programming. If you experience problems, especially with a 40-pin device such as a PIC16C74, use a piece of masking tape backed with a small piece of aluminum foil. However, later devices with the A suffix (i.e., PIC16C74A) do not have this problem. ☐

Duane M. Perkins is a self-taught engineer who has made computers and electronics his avocation since retiring in 1980. In recent years he has specialized in PIC microcontrollers. You may reach him at (717) 964-3536 or at dmp Perkins@compuserve.com.

SOURCES

Microchip Technology, Inc.
(888) 628-6247
(480) 786-7200
Fax: (480) 899-9210
www.microchip.com

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, subscribe@circuitscellar.com or www.circuitcellar.com/subscribe.htm.