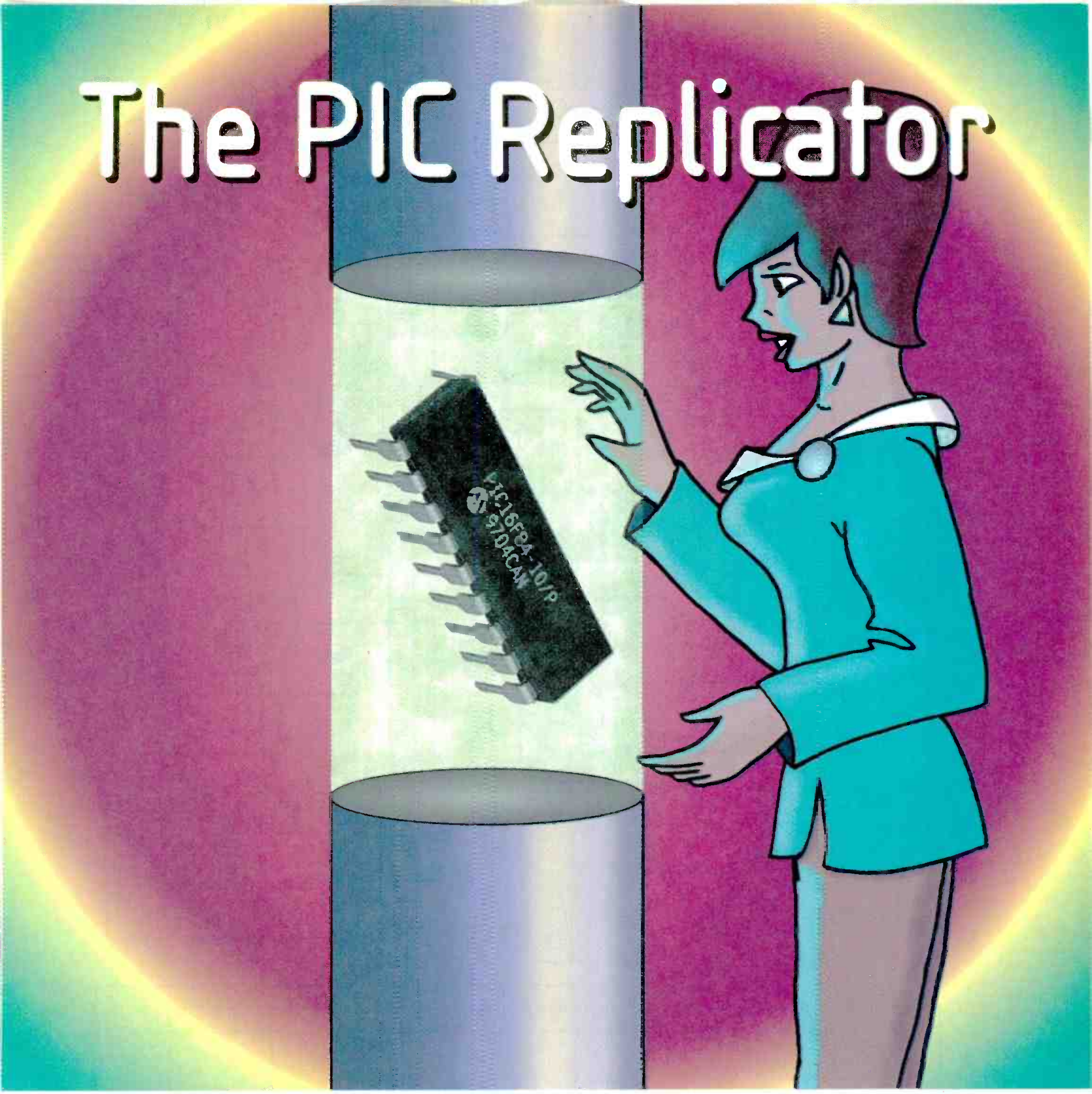
The PIC Replicator



Want to program PICs, but you don't have the required hardware? Here's a programmer that doesn't require a programmer to build. What's more, you can update the unit's software to take advantage of future developments in programmable microcontrollers

PETER BEST

Microchip's PIC microcontroller—cheap, easy to find, easy to program, and easy to use—is probably the most popular microcontroller in the world today. Relatively simple to implement, the only complexity that PICs have are the many available variants.

If you use or plan to use differing types of PICs, supporting their different requirements on the programming-hardware side can get expensive. For most of these programmers, a base unit needs a separate plug-in module for each PIC type to be programmed. The other end of the PIC-programmer spectrum is the really cheap unit that will empty your pockets by requiring software upgrades, special adapter sockets, and cables for each additional part to be programmed.

With thousands of engineers, designers, and hobbyists discovering the power of the PIC daily, many are waiting for a inexpensive but viable PIC programming platform—and

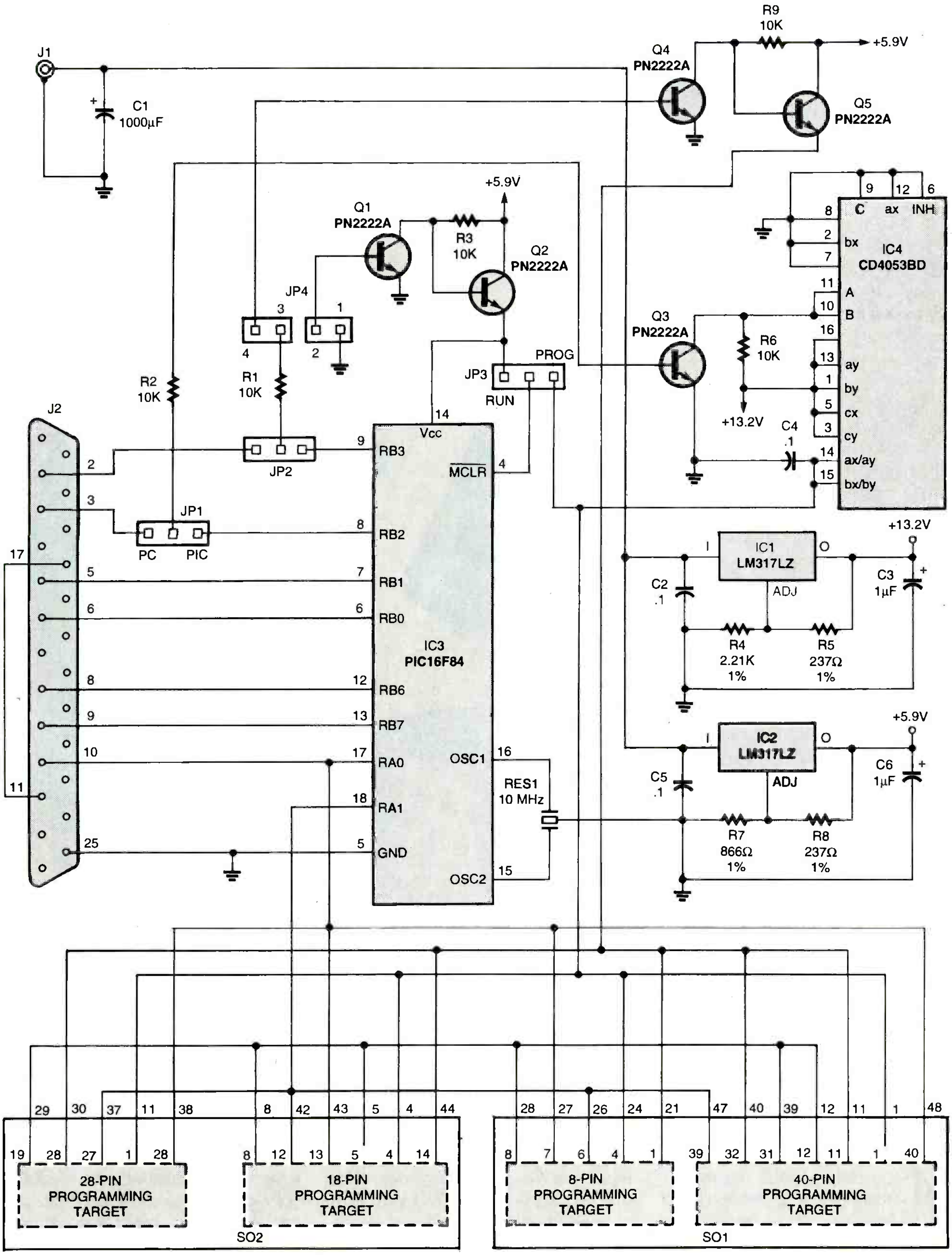


Fig. 1. Like many "intelligent" programmers, the PIC Replicator is built around a PIC microcontroller. However, by setting jumpers JP1–JP4 to the appropriate settings, you can use the basic hardware and host software running on a PC to program the controller. If you have the skill, you can even modify the controller's software to suit your needs.

someone's got to help these folks get started on the right foot!

The "perfect" PIC programmer would be able to program most (if not all) of the PICs currently available. It should be inexpensive, easy to build, and, best of all, easy to maintain. By that we mean you should be able to update and customize its functionality to take advantage of future PIC developments. Nothing can be quite as frustrating as investing time and money into a piece of equipment only to see it become obsolete in a matter of months or weeks!

To that end, we are proud to present the *PIC Replicator*. It has been designed for people who like to fool around with microprocessors like the PIC without burning a hole in their wallet. With about an hour's worth of construction effort, you'll be able to program and read all of the serially-programmable PICs such as the 12C5xx, 12C67x, 12CE67x, 16C6x/7x/9xx, and 16C8x/16F8x PIC devices either in or out of circuit. No special integrated circuits are required and all the parts are common off-the-shelf types that can be obtained from the various advertisers in this magazine. With this low-cost device, you can program your code into at least 40 different types of PICs, including the full line of the exciting new 8-pin "baby PICs."

Have special programming needs? No problem—you have full access to all of the project's source code in addition to the hardware schematics. The PIC Replicator is not locked into a single mode of operation. While it uses a preprogrammed microcontroller for on-board "intelligence," the best part is that the PIC Replicator can program its own controller! In fact, it was designed around the "catch-22" situation of "how do you program a programming controller when you don't have a programmer to start with?" For a custom need or to simply support a new type of PIC in the future, simply modify the PIC Replicator source code, compile it, and reprogram the on-board controller.

How It Works. The PIC Replicator is really a very simple piece of hardware that can be divided into four major parts: the programmer engine, the 5-volt supply and switch circuit, the 13-volt supply and switch circuit, and the PIC target sockets.

We'll discuss each section in turn. To follow these discussions, refer to the schematic diagram shown in Fig. 1.

There is only one "intelligent" component in the PIC Replicator circuit: IC3, a PIC16F84. The program that is stored inside it is what gives the PIC Replicator its "personality;" we'll talk about that program in a moment. With the proper program, IC3 acts as an interface to the printer port and host software running on the PC. It interprets and transfers data and commands, controls the programming voltages, and controls some aspects of the programming process.

What makes the PIC Replicator different from other programmers is the choice of microcontroller for IC3. Thanks to the way the 16F84 series was designed, it is a simple task to program it with very little hardware beyond a PC and some clever software. To see how little hardware is needed, look at the "No Parts PIC Programmer" article that appeared in the September 1998 issue of **Electronics Now**. A sim-

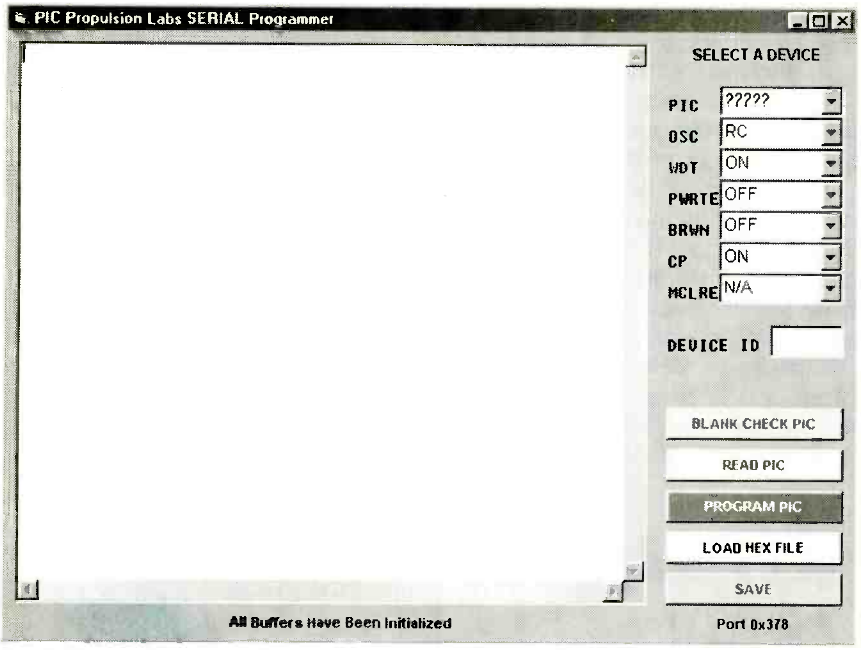


Fig. 2. The PIC Replicator software sports a clean, simple interface that makes it easy to manage your PIC projects. With the click of a button, you can read, write, or check the contents of any supported microcontroller.

ilar approach is used in the PIC Replicator. By setting jumper blocks JP1 and JP2 to "PC CONTROL" and JP3 to "PROGRAM," IC3 can be programmed with the appropriate control program. That way, you can build the programmer without needing access to an already-existing programmer. Normally, having JP3 in the "RUN" position applies 5 volts to pin 4 of IC3, allowing normal operation of IC3. Setting JP3 to the "PROGRAM" position lets us raise pin 4 to 13 volts, putting IC3 into its programming mode.

Pins 6 and 7 of IC3 receive clocked data from J2, which is connected to a PC's printer port. Keep in mind that when we use the printer port to transfer information between the PC and the PIC Replicator, we're sticking with the traditional output-only and input-only lines of the port. The bi-directional features of the parallel port are not used in the PIC Replicator as some of the parallel port lines are dedicated to output functions only. Using the bi-directional capability in that instance could cause port-pin conflicts that could result in unpredictable logic levels and damage to IC3 or the parallel-port circuitry. The big picture here is to use the available hardware resources in their simplest manner without adding additional gates and buffers that would drive up the cost and complexity of the circuit. To sum up the whole idea in a word or two, we're using the basic hardware as is and making the software and firmware do the work.

This approach guarantees that the PIC Replicator is compatible with every type of printer port in existence. While IBM did have fully bi-directional circuitry in their original PC, that feature was not officially documented. That omission put the clone makers in the delicate position of whether to follow the specification or the implemented hardware.

Since the incoming data from the PC is only connected to IC3 and not to the programming sockets, SO1 and SO2, you would think that the incoming clock and data information from the parallel port would be buffered and translated by IC3's internal software before being passed to the target PIC being programmed. That is true to some extent. In reality, the PIC Replicator does not buffer every bit of data that it sees—it only captures data from those pins when commands are being issued to it. Otherwise, the actual values of pins 6 and 7 are read and immediately transferred to pins 17 and 18, which connect to SO1 and SO2. Thus, the data information from the PC program is never captured and is "passed through" to the target as if the parallel-port pins were directly

PARTS LIST FOR THE PIC REPLICATOR

SEMICONDUCTORS

IC1, IC2—LM317LZ adjustable voltage regulator, integrated circuit
IC3—PIC16F84 microcontroller, integrated circuit
IC4—CD4053 analog multiplexor, integrated circuit
Q1–Q5—PN2222A NPN silicon transistor

RESISTORS

(All resistors are ¼-watt, 1% metal-film units unless otherwise noted.)
R1–R3, R6, R9—10,000-ohm, ¼-watt, 5% tolerance
R4—2210-ohm
R5, R8—237-ohm
R7—866-ohm

CAPACITORS

C1—1000-µF, 35-WVDC, electrolytic
C2, C4, C5—0.1-µF, ceramic-disc
C3, C6—1-µF, 16-WVDC, tantalum electrolytic

ADDITIONAL PARTS AND MATERIALS

J1—Co-axial power connector, PC-mount
J2—25-pin connector, PC-mount
JP1–JP3—Jumper pins, 3-position, 0.1-inch spacing
JP4—Jumper pin, 4-position, 0.1-inch spacing
RES1—Ceramic resonator, 10-MHz
SO1, SO2—48-pin zero-insertion-force socket
Jumper blocks, 15-volt DC wall-mounted transformer, IC sockets, wire, hardware, etc.

Note: The following items are available from E D Technical Publications, P.O. Box 541222, Merritt Island, FL 32954; 800-499-3387 (Orders); 321-454-9905 (Technical Support); Fax: 321-454-3198; Web: *www.edtp.com*; e-mail: *peter@edtp.com*: Complete kit of all parts, $89.95; PC board only, $25. Please add $7.50 for shipping and handling on complete kit. MC and VISA are accepted; no COD orders will be filled. FL residents must add appropriate sales tax.

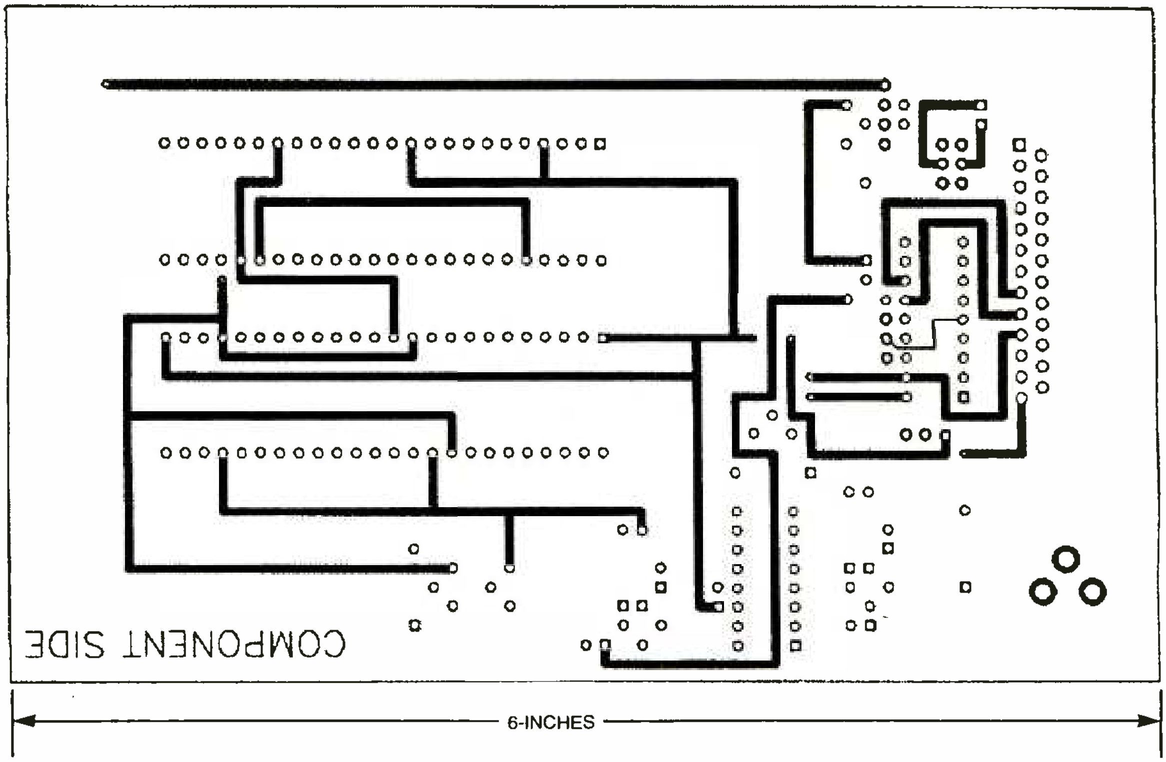


Fig. 3. Although the PIC Replicator can be built on a piece of perfboard, a PC board makes for a more robust unit while reducing wiring errors.

connected to the target sockets. The PIC16F84 with a 10-MHz clock is quick enough to read and transfer the data and clock information to the target PIC without missing any incoming data and clock signals from the PC's parallel port.

When reading information back from a programmed PIC, note that pin 17 of IC3 might cause problems due to conflicting signal levels. The PIC Replicator knows to put that pin in a high-impedance state, called a "tri-state," so that as far as the programmed PIC in SO1 or SO2 and the printer port are concerned, IC3 doesn't exist. The way that the hardware is arranged here, we could use that pin on IC3 to read the incoming data from the target PIC and hold it for later transmission. However, there is no need to do that. Since the PC's host software is reading the data directly from the target PIC, a simple modification of IC3's internal software is all that's necessary.

Powering the PIC Replicator. Overall power for the PIC Replicator is supplied by a 15-volt DC wall-mounted transformer through J1. Capacitor C1 provides some additional filtering. Power is applied to the inputs of IC1 and IC2, a pair of LM317-series adjustable voltage regulators. Precision resistors R4 and R5, along with capacitors C2 and C3, set the output from IC1 to the 13.2 volts needed for programming the microcontrollers.

During normal programming operations, pins 8 and 9 of IC3 control the power sequencing to the target sockets. Logic levels from those pins switch Q3 and Q4, respectively. That control is needed because the power and programming voltages to the PICs being programmed must be switched in a certain order; detailed information on the requirements for a particular microcontroller can be found in its data sheet.

Transistor Q3, along with R6, shifts the logic level from IC3 to allow 0 or 13.2 volts to pass through IC4, a CD4053 CMOS multiplexer. Note that the inputs and outputs of IC4 are connected in parallel to pass the required maximum programming current of 50 milliamps. The CD4053 is a good choice here in that the 0- and 13-volt circuits are completely isolated and there is little if any voltage drop between the chip's input and output pins. Capacitor C4 provides filtering and stability for the 13.2 volts going to the target sockets

Transistor Q4, along with Q5 and R9, switches the 5.9-volt supply generated by IC2, R7, R8, C5, and C6 to the target sockets. If you're wondering why such an "oddball" voltage of 5.9 volts DC is used, remember that the PN2222 transistor used for Q5 is not a perfect electronic device; the transistor's junction resistance will cause a slight voltage drop between the collector and emitter. The goal is to have the transistor switch pair provide a nominal 5.5 volts DC to the target sockets.

In the "replicate" mode, all of the programming voltage switching activity is aimed toward IC3, which could be an unprogrammed microcontroller or one that is about to be electrically erased and reprogrammed. We've already talked a bit about the role of the various jumper blocks; let's look at them in a bit more detail from a voltage-steering point of view.

Under normal conditions, JP4 has a pair of jumper blocks on it that connects R1 to the base of Q4 and grounds the base of Q1. We've

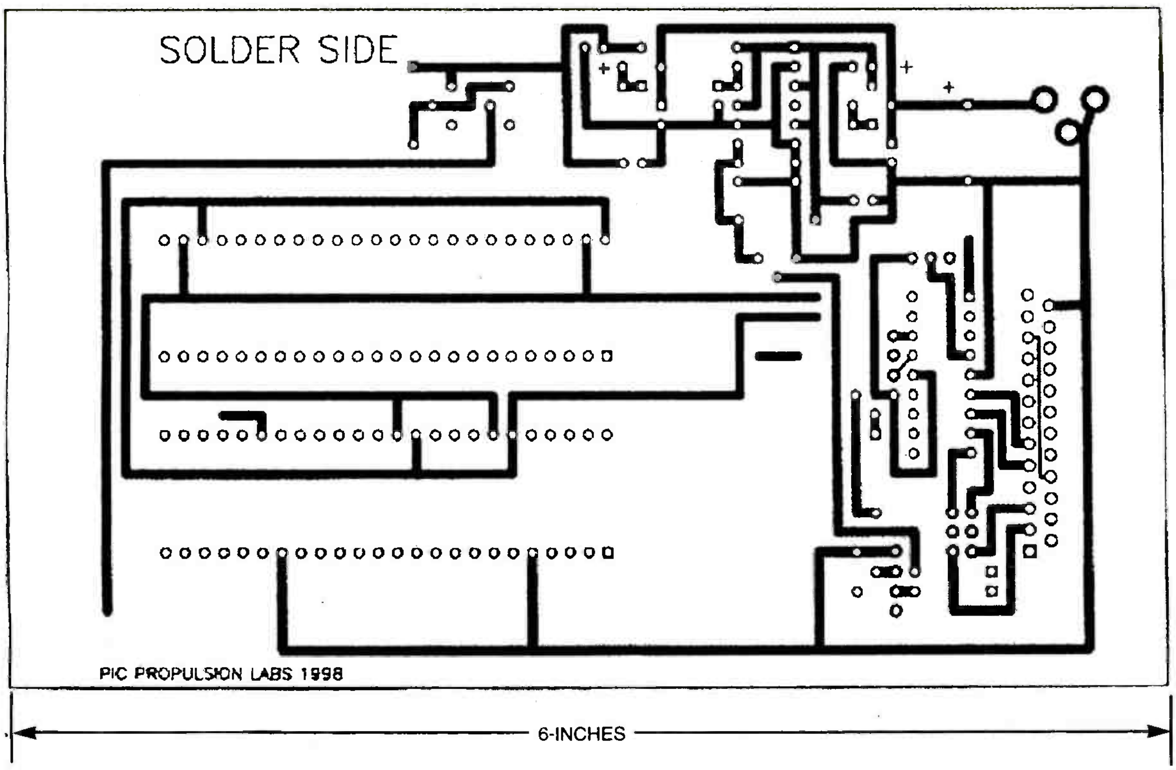


Fig. 4. Here's the foil pattern for the solder side of the PIC Replicator PC board.

already seen how the signal from pin 9 of IC3 passes through JP2, R1, and JP4 to Q4. With Q1's base grounded, it remains cut off, letting the voltage passing through R3 turn on Q2; the result is power for IC3.

During replication, the 5.5-volt DC source and the 13.2-volt programming voltage to IC3 must be controlled. To allow control of the 5.5-volt source, simply remove the two jumper blocks on JP4 and place one across the middle two pins. Note how the logic level from R1 is now applied to the base of Q1. The combination of Q1 and Q2 can now be used to turn the 5.5-volt power to IC3 on and off.

Now that Q1 has control of IC3's power, the next step is to ensure that IC3 itself doesn't try to turn its own power on and off! That is the reason for JP1 and JP2. Switching those jumpers so that the printer port has direct control of R1 and R2 instead of pins 8 and 9 of IC3 ensures that IC3 won't try to accidentally perform "brain surgery" on itself during programming.

The only control left to adjust is to allow 13.2 volts to flow to IC3's reset pin, pin 4. That is done by moving JP3 to the appropriate pins. Normally, JP3 shorts the reset IC3's pin to its power-supply pin. With the jumpers arranged as described, a new IC3 can be created or a current one modified. Moving the jumpers back to their "normal" position makes the PIC Replicator ready to program other microcontrollers in the "socket farm."

The PIC Replicator "Socket Farm." There is no need to go into explicit details on how to program a PIC chip; that information is well documented by Microchip and is available for anyone to download through Microchip's Web site (*www.microchip.com*). Of course, if you want to know what the PIC Replicator's code is doing and why it is doing it, read the Microchip programming specifications. The PIC Replicator's software and firmware was designed using the "development programmer" rules.

You're probably wondering what the PIC Replicator's target sockets has to do with that. It's quite simple: documentation on what signals connect to what pins on the sockets is available in Fig. 1 (how) and Microchip's rules of programming PICs (why). If you consult the Microchip programming specifications and follow the PIC Replicatior's source code, you'll find all of the PIC Replicator signals in their right places being driven by the correct voltages and waveforms.

Note that SO1 and SO2 each serve two types of PIC devices. They are grouped by pin count: 18- and 28-pin devices go with SO1 while 40- and 8-pin devices use SO2. It would be safe to say that, for example, any 18-pin serially-programmed device could go into SO1 at the designated pins and that any 8-pin device could do the same in SO2. To make that a safe bet, you can only select compatible devices from the PC host program. That goes for all of the socket-farm locations. If you need to program parts that are not in the host program's list, consult the appropriate Microchip datasheets for locations of programming voltages and data/clock pins before you attempt to read or program a nonsupported device. Should Microchip modify their programming requirements with future products, you can use the code for IC3 as a basis to do custom coding to handle any new or custom serially-programmable PIC parts.

PIC Replicator Firmware. The

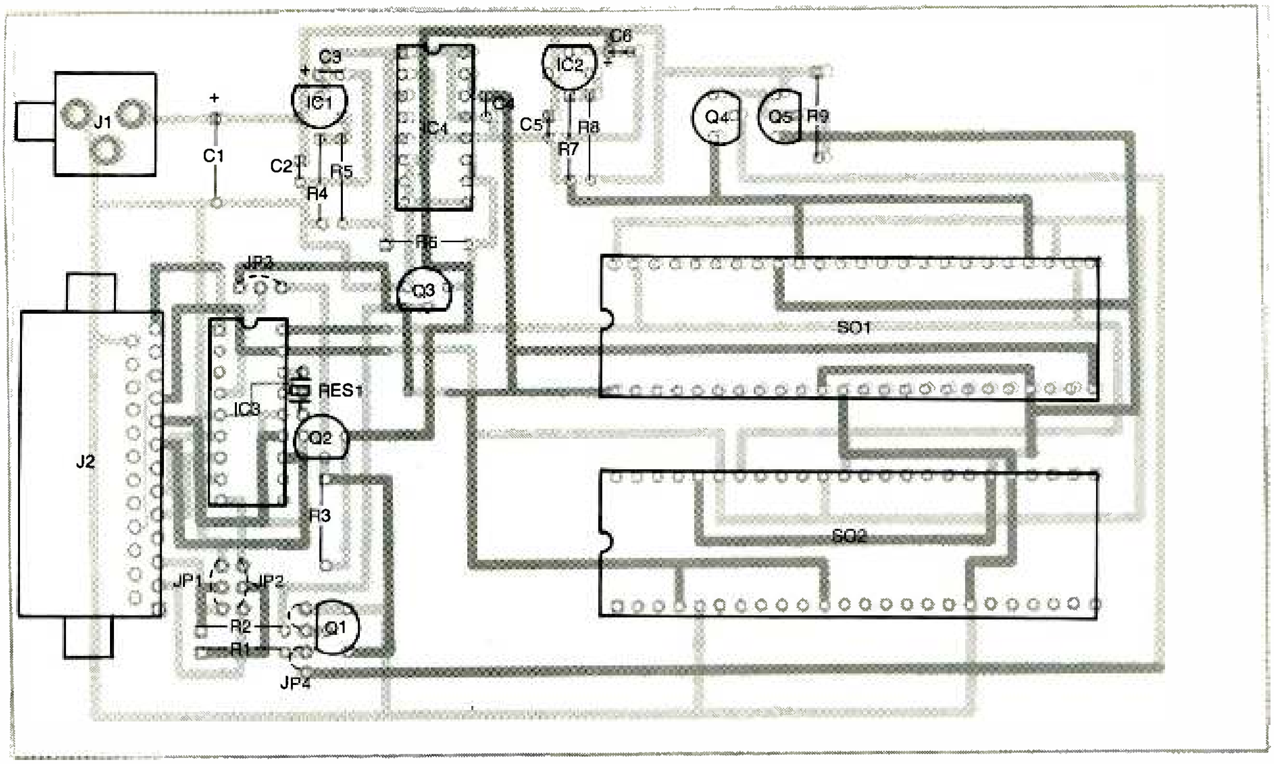


Fig. 5. The PIC Replicator is easily built on a double-sided PC board. The zero-insertion-force (ZIF) "socket farm" lets you program PICs of any size or pin count—including future models.

actual code length for IC3's program is exactly 200 instructions. The source code, as well as the various compiled object code formats, is bundled with the host software. That bundle can be downloaded from the **Poptronics** FTP site; it can be found at *ftp.gernsback.com/pub/pop/pic_replicator.zip*.

The program begins by powering off any target-socket programming voltages that might be present and waits for pin 9 (signifying "power on") from the parallel port to go high. When that happens, the firmware enters an "internal command" mode. Internal commands substitute for using hardwired control bits like the power-on bit we just used. In fact, that power bit could have been implemented as a firmware command as well, but the port pin was available.

The use of internal commands lets us "talk" to the PIC Replicator firmware before it enters the program/read phase of operation. Thus, we can add commands and control the PIC Replicator at will in that no hardware needs to be added or changed when an internal command is added or deleted.

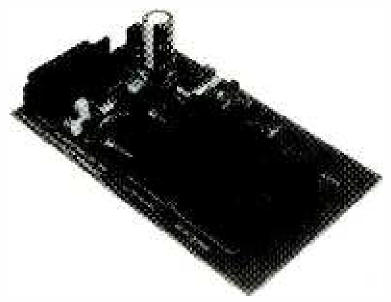
Currently, there are only three internal commands.

Those commands are six bits long and inform the PIC Replicator as to what type of power sequencing to perform. A normal power-up command tells the firmware that when the power bit is high to raise the power voltage to 5.5 volts DC and two milliseconds later to raise the programming voltage to 13.2 volts. That puts most of the parts into what is termed a "test" or "program" mode. The reverse-power-up command is used for the 12C6XXX baby PIC parts only. Those parts require that the programming voltage be raised *before* the power voltage with the same timing and prerequisites as the normal power-up. The final internal command tells the firmware that a "flash" or EEPROM (*E*lectrically *E*rasable *P*rogrammable *R*ead-*O*nly *M*emory) device is mounted in the target socket. The "flash" command tells the firmware to allow 10 milliseconds for the program cycle. Otherwise, the program cycle lasts 100 microseconds. The power-up sequence for the flash command is defined as normal.

After the internal command phase is complete, the firmware enters the "command cycle" phase. This phase constantly monitors the data and clock inputs from the parallel port in search of a six-bit programming command. There are eight commands that can be issued:

- Load Configuration
- Load Data
- Read Data
- Increment Address
- Begin Programming
- End Programming (not used for PIC16X8XX devices)
- Load Data for EEPROM Data Memory (PIC16X8XX only)
- Read Data from EEPROM Data Memory (PIC16X8XX only)

There are actually two more commands: Bulk-Erase Program Memory and Bulk-Erase Data Memory. They are not implemented because the PIC Replicator programs all of the memory area of the PIC16X8XX devices on every program pass. If you need to bulk erase the flash and EEPROM parts, program the parts with a file that



The completed PIC Replicator is not a "dead-end" device—it can be updated and customized as needed.

contains all ones.

The PC Host Software. Written in Visual Basic, the PIC Replicator host program sends commands to and receives data from the target PIC device being programmed through the PIC Replicator hardware. The hardware is invisible as far as the host program is concerned; the host program thinks that it is speaking directly to the target PIC device. To accomplish that, signal timings that cannot be performed by the host program are handled by the PIC Replicator hardware. That technique combined with special software technology in the host software lets the entire PIC Replicator system run on any Windows-based system at any processor speed.

Although we're using a parallel port on a PC, the PIC Replicator software and hardware communicate with each other in a serial fashion. Using a serial port would have added to the complexity of the project—something that we're trying to avoid. Since the PIC Replicator hardware doesn't have any RS-232 conversion ICs or special serial-to-parallel chips (that's done in the PIC Replicator firmware), a "bit-bang" technique is used for communication.

Target PICs were abused unmercifully during the development of the software so that the final product would be smart enough not to let you do so. While we can't guarantee that you won't hurt a PIC at some point either by design or accident, rest assured that the PIC Replicator software will attempt to insure that you don't.

When you start the PIC Replicator host software on your PC, the first task that it does is to look for the PIC Replicator hardware. Note that in the schematic diagram in Fig. 1, pins 11 and 17 of J2 are tied together. It is that short that the host software looks for on every properly installed printer port on the PC. When the PIC hardware is identified, the software will look like the screen shot shown in Fig. 2. As you can see, it is quite simple. All you have to do is select the PIC type and click on a command button. All of the PIC-programmable settings, such as the watchdog timer (WDT), oscillator type (OSC), and so on, should be built into the compiled code that you will be programming into the target PIC; such an arrangement is easy to do when you write your source code under Microchip's program-development software, MPLAB. Those settings are loaded into the PIC Replicator buffers from the file that MPLAB generates during code compilation. While you don't have to use Microchip's software to create your PIC program, using MPLAB eliminates having to manually enter configuration settings into the PIC Replicator host software.

MPLAB has many convenient features such as simulation and error checking; it's a free download that you can get from Microchip's Web site (*www.microchip.com*).

Let's get back to the host software. One of its useful features is performing a "blank check" on a part that the user knows is not blank. Doing this will yield the contents of every non-blank location including the configuration word and the customer i.d. locations. Got a "configuration unknown" PIC? Use the PIC Replicator to read and blank-check it to get the "full Monty" of everything that is stored in it. You can even use the PIC Replicator to "replicate" other non-code-protected PIC parts.

Speaking of replication, the very first action that the PIC Replicator will do is to create its initial programmed IC3. Of course, before you can do that, you have to build the PIC Replicator first.

Assembling the PIC Replicator. This is it—it's time to put the parts together that talk to the software and firmware and bring your PIC Replicator to life. While you can use perfboard and standard construction techniques, a printed-circuit board yields a neater unit with less chance for miswiring errors.

Foil patterns for a PC board are shown in Figs. 3 and 4. If you don't want to attempt to etch a double-sided board, one can be purchased from the source given in the Parts List.

For those assembly methods, follow the parts-placement diagram shown in Fig. 5 for component location. As we build the board we will be testing it; that will help catch mistakes such as wrong parts and components installed backwards before something expensive gets destroyed.

Begin by building up the power-supply regulator circuits. Install J1 and C1, plug in the wall transformer, and test for 15 volts across C1. With the power disconnected and C1 discharged, install the regulator circuits: IC1, IC2, and their associated resistors and capacitors. With power reapplied, you should get 5.9 volts DC from the output of IC2 and +13.2 volts from the output of IC1. Remove power before installing any other parts.

If all is well, install all of the jumpers and assemble the remaining transistor switch pairs and their associated resistors. When it's powered, you should be able to toggle the output voltages of each transistor pair. Apply 5 volts to the base of Q1 through R1. Place a jumper between pins 2 and 3 of JP4 and apply the voltage to the center pin of JP2. The emitter of Q2 (on JP3) should switch to about 5.5 volts. Grounding R1 (jumpering pins 1 and 2 of JP4) should remove that voltage. The same test is done with Q4; a jumper on pins 3 and 4 of JP4 lets you use R1 again. The emitter of Q5 should switch to 5.5 volts in a similar way as before.

Install IC4. Doing the same switching test as before with the base of Q3 (use JP1 and R2 instead) should toggle 13.2 volts at pins 14 and 15 of IC4.

Use a socket to mount RES1—it must be removed when you are programming IC3.

Once you're sure that all of the

voltages are within tolerances and that they can be controlled from both the pins of IC3 and the parallel-port connector pins, mount the remaining sockets and connectors. It is easier to troubleshoot the parallel-port connections and the socket farm before you mount the connector and sockets.

Now we're ready to "burn" the PIC Replicator firmware into IC3. Set JP1 and JP2 to "PC," short pins 2 and 3 of JP4, and set JP3 to "PROG." Remove RES1 and insert a blank PIC16F84 into the socket for IC3. Connect the PIC Replicator to the PC's parallel port with a 25-pin cable; the wires must go "straight through" to all 25 pins.

Start the host program and choose "ENGINE" for the PIC type. On-screen instructions will appear to walk you through the process. It's important that the ENGINE.COD file that was bundled with the software be available; that file contains the actual instructions for IC3.

Once IC3 is programmed, remove power. Set JP1 and JP2 to "PIC," JP3 to "RUN," and two jumpers on pins 1–2 and 3–4 of JP4. Insert RES1 into its socket, and the PIC Replicator is ready to start replicating PICs.

A Quick Review. Let's recap what you now know about the PIC Replicator. Basically, the unit has two modes of operation: "normal" mode in which IC3 is in control of programming and read operations and "replicate" mode that puts a blank PIC16F84 into the socket for IC3 for replication (programming) through the PC's host software and parallel port.

The PIC Replicator is a simple electronic device that depends heavily upon the PC host program and its internal firmware. If you lean towards having the hardware in control, that might be a scary concept for you. The bottom line is that the PIC Replicator hardware is standard stuff and constant. The software and firmware are the variables that make the PIC Replicator useful.

I purposely didn't mention this before, but IC3 can be a PIC16C83, a PIC16C84, a PIC16F84A, a PIC16C61, a PIC16C71, a PIC16C710, a PIC-16C711, a PIC16C620, a PIC16C621, or a PIC16C622. Once the initial PIC16F84 is replicated, you can port the source code to any of the above processors with MPLAB, program them with your PIC16F84 version of the PIC Replicator, and plug the newly-programmed PIC into IC3 as the new PIC Replicator engine. In fact, I designed and tested the original PIC Replicator using a PIC16C61 because I didn't have the tools handy for emulating a PIC16F84.

There are gains and losses for substituting processors, but the point is that the PIC Replicator is not a dead-end device. The PIC Replicator was designed to be a low-cost and powerful general-purpose PIC tool with the capability of being upgraded or modified without the need for special hardware or significant hardware changes.

If you have other jobs for the PIC Replicator, all of the programming power you need is located in the socket farm. There's no reason why you could not design a daughter board to plug into SO1 or SO2 that would hold components to interface to other programmable devices. With the PIC Replicator source code and hardware specifications at your disposal, as far as your special PIC Replicator applications are concerned, the sky is the limit.

If you're a beginner to PICs, you have just built a very useful PIC tool that will help you understand all of the nuances of the PIC family as well as provide a means to put your newly-acquired talent to work. If you've been around PICs, the design techniques found in the host software and IC3 firmware might just provide the impetus, and possibly an answer, for that next PIC design.

The specifics of writing PIC programs is beyond the scope of this article. There are not enough pages in this magazine to cover the topic adequately—entire books have been written on the subject. If you need help, there are many sources at your disposal: Internet forums and newsgroups, magazine articles, books, public and university libraries...the list is endless.

If you have any questions concerning the PIC Replicator, the author can be reached through his Web site (*www.edtp.com*). The PIC Replicator PC program has been designed to help you along the way. Don't be surprised if you soon find yourself in the midst of designing that next big "killer PIC project" that just might appear someday in the pages of this magazine

What are you waiting for? A whole new world of microprosessor-based electronics awaits your next discovery! P