Large Digits on a 4x20 LCD

By Chuck Hellebuyck

Driving a Liquid Crystal Display (LCD) module has become very easy to do with the various PIC® microcontroller (MCU) options that are available. microEngineering Labs' PICBASIC PRO compiler, Basic Micro's BasicATOM chips and many other PIC MCU-based chips, modules and compilers offer a command dedicated to driving these types of displays. When I've discussed driving an LCD module from a PIC MCU in the past, I received several emails asking where to buy one. I just took it for granted that everybody knew they were practically available anywhere. The secret is in the common driver chip used in the LCD modules. LCD modules are 99% of the time controlled by a Hitachi 44780 chip that handles all of the character generation. If you find an LCD module with that chip, then you are set. You can get that type of LCD module from all of the typical suppliers, such as Digi-Key, Mouser, Jameco and others, but you can usually find a better deal by visiting individual Web sites. One of my favorites is junun.org. They sell robot kits and parts, and also have a great deal on LCDs. You can even get surplus LCDs and LCDs removed from equipment, at Marlin Jones (MPJA.com) and others, at incredibly cheap prices. I use a surplus unit in this month's project.

For this month's article, I wanted to cover a little different feature of the Hitachi LCD chip's character generator, which many may not know about. The first eight locations of character memory can be modified to make custom 5x8 pixel characters. In this month's column I will load custom 5x8 characters into those eight locations and then use those custom 5x8 characters to display large characters on a 4x20 LCD. I will drive the 4x20 LCD from a BasicATOM 28-pin chip, which is just a PIC16F876A with the custom BasicATOM self-programming bootloader installed. To make the connections simpler, I use my Ultimate OEM module with the BasicATOM 28 installed. However, wiring it directly to any PIC MCU will also work. If you want to use the PICBASIC PRO compiler or some other compiler, even the software setup is very similar. Let me show you how it's done.

Project Description

To start, let me explain how the custom characters are set up in the LCD. The LCD control chip has eight locations at the beginning of its character memory, which can be modified to make custom 5x8 characters. Once those are stored in the LCD's chip, these custom 5x8 characters can be called the in the same manner as any standard, pre-stored ASCII character. In Figure 1, a 5x8 character bit map is shown with the first custom character the program will define.

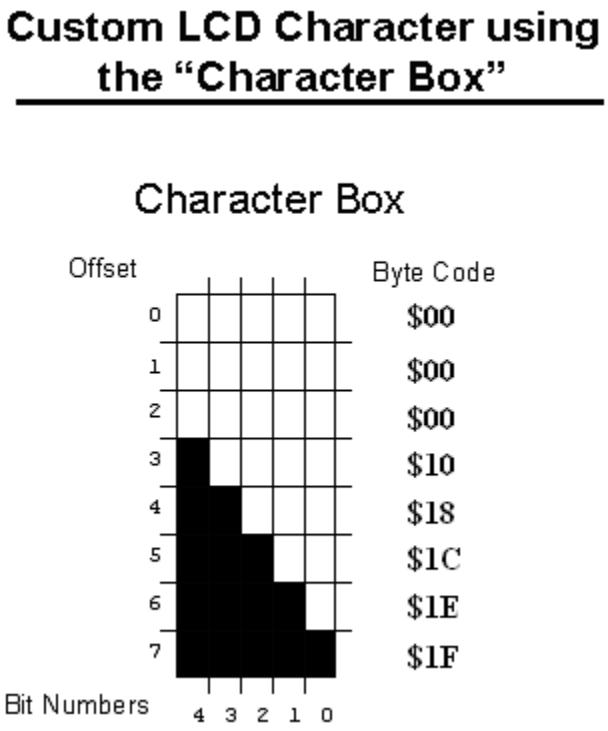

Figure 1: Custom LCD Character Using the "Character Box"

This custom character will be placed at memory location $00 of the LCD character memory. It then forms a small ramp that slopes downward.
Each row of the character has to be defined by a byte value. Since the characters are only 5 bits wide in size, the three most significant bits of the byte value are always zero. The highest bit value is the 5th bit (bit number 4).

The key is to set or clear the proper bits to form the character you want. For example, if you look at the fourth line of the character box (offset row value 3); the byte to the right shows $10 or binary %00010000. This makes the 5th block solid black, but the rest of the row clear. The next line sets two blocks black, by using byte $18 or binary %00011000, and the rest you can see in Figure 1. By setting these bits and storing the data in the LCD's character memory, we have established a new custom character that the software can call.

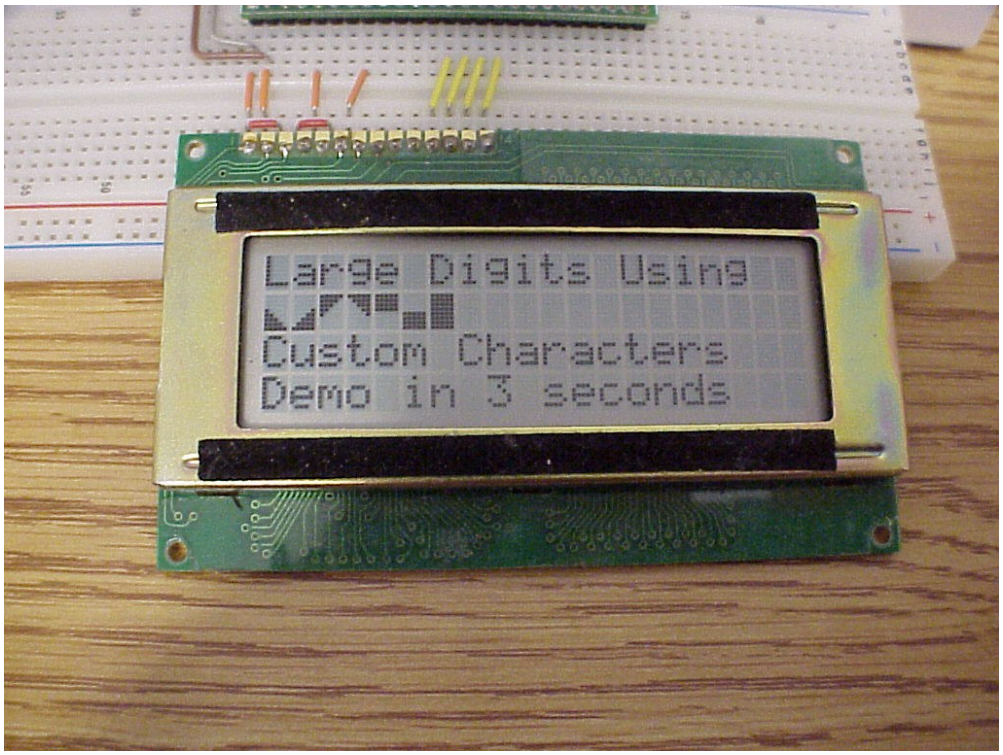

Figure 2: Custom LCD Characters

This doesn't stop us from displaying standard, single-line words on the LCD screen. It does allow us to also create large digits that span all four lines of a 4x20 LCD, by using the custom characters shown in Figure 2. It shows seven custom characters on line 2 of the LCD, but the eighth custom character is a blank so it doesn't appear. The main project software loop, discussed later, will create the hexadecimal number system and display them in sequence, starting with the numbers 0 – 9 and then A – F, all in a large-character format that spans all four lines. The number "1" is displayed in Figure 3 to illustrate what I'm trying to describe. Using large numbers like this makes it very easy to read from across the room.

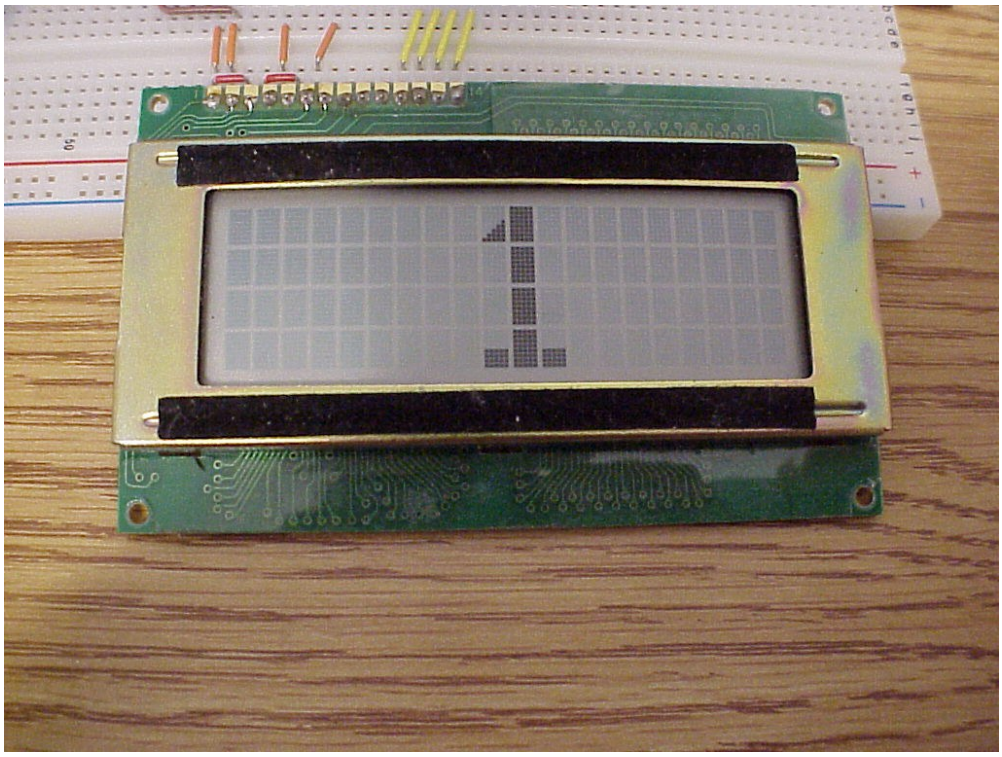

Figure 3: An Example Large Character

Project Setup

If you remember the 2x16 LCD project that I did in a previous column, then you will find the 4x20 LCD has all the same connections. A great advantage to using LCD modules is their common connection system, which makes it easy to change from 2x16 to 4x20. Figure 4 shows the connections to the Ultimate OEM BasicATOM module. You can easily connect the LCD to a PIC16F876A directly, by following the connection names in the schematic.

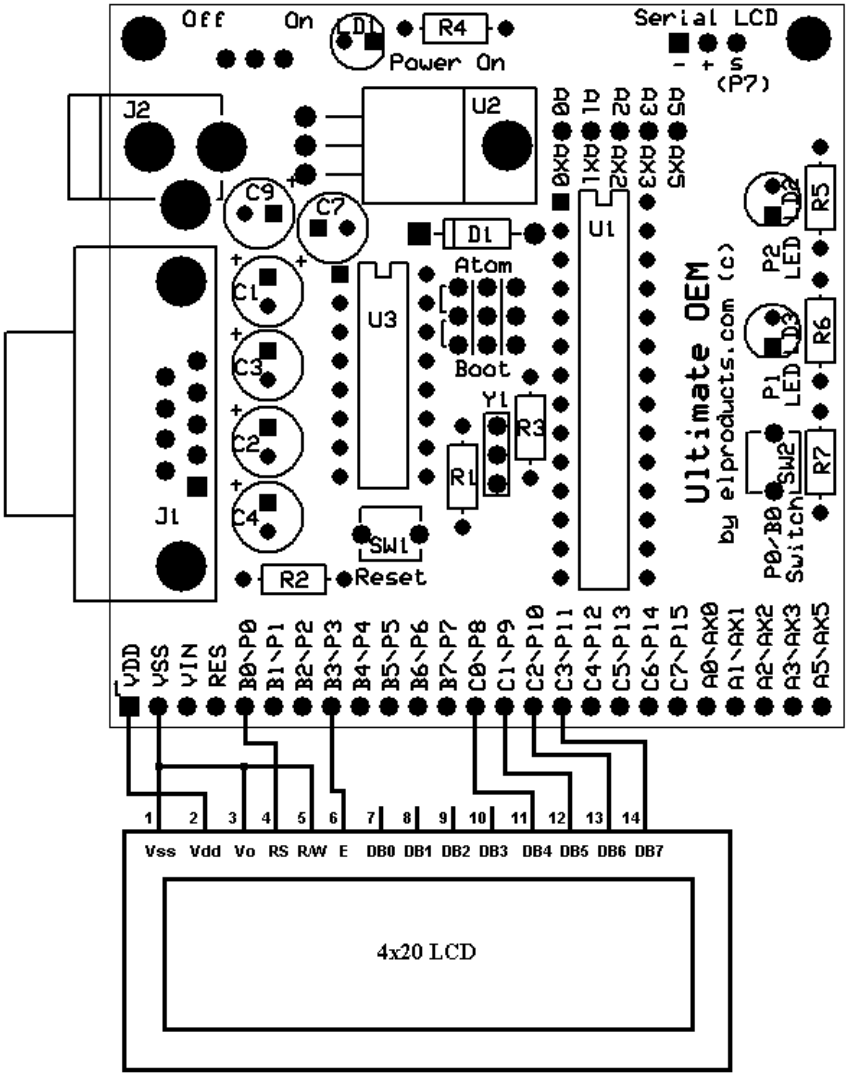

Figure 4: Connections to the Ultimate OEM BasicATOM Module

Software

This software listing is kind of long, but most of the code deals with setting up the LCD to display the large custom characters. In the code, you will notice the "|" pipe character at the end of several lines. This is for line continuation. This is a special character that the BasicATOM compiler recognizes as a continuation message. When the compiler sees that character, it knows the command line was too long for the editor window and continues on the next line. Setting up the characters takes a lot of space, so the line-continuation function is used often.

```
x var byte
char var byte
epin con 3          'Establish nickname for LCD enable pin
rspin con 0         'Establish nickname for LCD Register Select pin

' *** Initialize LCD ***
pause 500
lcdwrite rspin\epin,outc,[initlcd1,initlcd2,twoline,scr,clear,home]

'*** Create Custom Characters in LCD memory locations 0-7 ***

lcdwrite rspin\epin,outc,[CGRAM]
for x = 0 to 63

lookup x,[$00,$00,$00,$10,$18,$1C,$1E,$1F,$00,$00,$00,$01,|
$03,$07,$0F,$1F,$1F,$1E,$1C,$18,$10,$00,$00,$00,$1F,$0F,|
$07,$03,$01,$00,$00,$00,$1F,$1F,$1F,$1F,$00,$00,$00,$00,|
$00,$00,$00,$00,$1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F,|
$1F,$1F,$00,$00,$00,$00,$00,$00,$00,$00], char

lcdwrite rspin\epin,outc,[char]
next

' *** Initial screen with program description ***
main
lcdwrite rspin\epin,outc,[clear,home,scrram,"Large Digits Using"]
lcdwrite rspin\epin,outc,[scrram + $40]

for x = 0 to 7
lcdwrite rspin\epin,outc,[x]
```

```
next
lcdwrite rspin\epin, outc,[scrram + $14, “Custom Characters”]
lcdwrite rspin\epin, outc,[scrram + $54, “Demo in 3 seconds”]
pause 3000

‘ *** LCD control code to display 0 - F large characters ****

‘ *** “0” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,6,scrram+$49,|
6,7,6,scrram+$1D,6,7,6,scrram+$5D,6,5,6]
pause 1000

‘ *** “1” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,1,6,scrram+$4A,|
6,scrram+$1E,6,scrram+$5D,5,6,5]
pause 1000

‘ *** “2” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,4,4,6,scrram+$49,|
5,5,6,scrram+$1D,6,7,7,scrram+$5D,6,5,5]
pause 1000

‘ *** “3” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,4,4,6,scrram+$49,|
5,5,6,scrram+$1D,7,7,6,scrram+$5D,5,5,6]
pause 1000

‘ *** “4” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,7,6,scrram+$49,|
6,7,6,scrram+$1D,4,4,6,scrram+$5D,7,7,6]
pause 1000

‘ *** “5” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,4,scrram+$49,|
6,5,5,scrram+$1D,7,7,6,scrram+$5D,5,5,6]
pause 1000

‘ *** “6” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,4,scrram+$49,|
6,5,5,scrram+$1D,6,7,6,scrram+$5D,6,5,6]
```

```
pause 1000

‘ *** “7” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,4,4,6,scrram+$49,|
7,7,6,scrram+$1D,7,7,6,scrram+$5D,7,7,6]
pause 1000

‘ *** “8” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,6,scrram+$49,|
6,5,6,scrram+$1D,6,4,6,scrram+$5D,6,5,6]
pause 1000

‘ *** “9” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,6,scrram+$49,|
6,5,6,scrram+$1D,7,7,6,scrram+$5D,7,7,6]
pause 1000

‘ *** “A” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,6,scrram+$49,|
6,5,6,scrram+$1D,6,7,6,scrram+$5D,6,7,6]
pause 1000

‘ *** “b” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,7,7,scrram+$49,|
6,5,5,scrram+$1D,6,7,6,scrram+$5D,6,5,6]
pause 1000

‘ *** “C” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,4,scrram+$49,|
6,7,7,scrram+$1D,6,7,7,scrram+$5D,6,5,5]
pause 1000

‘ *** “d” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,7,7,6,scrram+$49,|
5,5,6,scrram+$1D,6,7,6,scrram+$5D,6,5,6]
pause 1000

‘ *** “E” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,4,scrram+$49,|
6,5,5,scrram+$1D,6,7,7,scrram+$5D,6,5,5]
```

```
pause 1000

‘ *** “F” character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,6,4,4,scrram+$49,|
6,5,5,scrram+$1D,6,7,7,scrram+$5D,6,7,7]
pause 1000

‘ **Final message from program before looping back to the top **

lcdwrite rspin\epin,outc,[clear,home,scrram,”Just imagine what”,|
scrram+$40,”you can do!”]
pause 3000
goto main
```

How it Works

First, we establish a few variables and constants. The variables are just temporary storage locations labeled X and Char. The constants define the LCD “E” pin and “RS” pin.

```
x var byte
char var byte
epin con 3          ‘Establish nickname for LCD enable pin
rspin con 0         ‘Establish nickname for LCD Register Select pin
```

The program initializes the LCD by first waiting ½ second for it to warm up, and then it issues the LCDWRITE command to set it up as a two-line LCD. A 4x20 LCD is really a two-line by 40 character LCD, with line 1 broken up between lines 1 and 3, and line 2 broken up between lines 2 and 4, to form a 4x20 LCD.

```
‘ *** Initialize LCD ***
pause 500
lcdwrite rspin\epin,outc,[initlcd1,initlcd2,twoline,scr,clear,home]
```

The next section is the heart of this program. In this block of code, the custom characters are created and stored in the LCD character memory locations 0 thru 7. Each character takes 8 bytes of data, for a total of 64 bytes (8 characters times 8 bytes).

To do this, we first have to point to location zero of the Character RAM. We do this with the LCDWRITE command, again by sending the "CGRAM" pointer. We don't have to add an address value, since it defaults to the zero or first location.

```
'*** Create Custom Characters in LCD memory locations 0-7 ***

lcdwrite rspin\epin,outc,[CGRAM]
```

Now, we send the custom characters to the LCD character RAM by using a FOR-NEXT loop and the LOOKUP command. The FOR-NEXT loop counts from 0 to 63, for a total of 64 loops, and it defaults to stepping one count per loop. The variable "x" stores the present loop count value. The LOOKUP command then takes the value of "x" and jumps that many places, reads the byte value and stores it in the "char" variable. For example, lets assume x = 5 or the 6th time through the loop since the count starts at zero. The value of "char" will equal $1C, since it is the sixth value listed.

```
for x = 0 to 63

lookup x,[$00,$00,$00,$10,$18,$1C,$1E,$1F,$00,$00,$00,$01,|
$03,$07,$0F,$1F,$1F,$1E,$1C,$18,$10,$00,$00,$00,$1F,$0F,|
$07,$03,$01,$00,$00,$00,$1F,$1F,$1F,$1F,$00,$00,$00,$00,|
$00,$00,$00,$00,$1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F,$1F,|
$1F,$1F,$00,$00,$00,$00,$00,$00,$00,$00], char

lcdwrite rspin\epin,outc,[char]
next
```

After the code above executes, the custom characters are now in the LCD character-generator memory. The program can now call them to create the large characters on the LCD. The "main" label starts the central program loop. In the section below "main", we use the LCDWRITE command to display a description of what this program will do, as shown in Figure 2. We display "Large Digits Using" by using the LCDWRITE command.

```
' *** Initial screen with program description ***
main
```

```
lcdwrite rspin\epin,outc,[clear,home,scrram,"Large Digits Using"]
lcdwrite rspin\epin,outc,[scrram + $40]
```

This next section will call up the custom characters just created, one at a time, using a FOR-NEXT loop, and will display them using the LCDWRITE command. The variable "x" holds a value from 0 to 7. LCDWRITE directs the LCD to display characters 0 thru 7. See how easy it is to display custom characters, once they are created?

```
for x = 0 to 7
lcdwrite rspin\epin,outc,[x]
next
```

We finish this block of code by displaying "Custom Characters" and "Demo in 3 seconds" to the display lines 3 and 4. SCRRAM +$14 is the beginning of line 3, and SCRRAM +$54 is the beginning of line 4.

```
lcdwrite rspin\epin, outc,[scrram + $14, "Custom Characters"]
lcdwrite rspin\epin, outc,[scrram + $54, "Demo in 3 seconds"]
```

Finally, we pause 3 seconds so you can read the display and then move on to the next section.

```
pause 3000
```

From here, the program creates the custom large characters using the custom 5x8 characters stored in CGRAM. I'll just describe the digit "1", shown in Figure 3, but all the other large character sections below operate in the same manner.

The #1 character is created by placing custom characters 1 and 6 on line one, character 6 on line two, character 6 on line three, and characters 5,6 and 5 on line four. We then pause one second, so the digit can be read. The scrram is offset with values that center the "1" on the LCD.

```
' *** "1" character
lcdwrite rspin\epin,outc,[clear,home,scrram+$09,1,6,scrram+$4A,|
6,scrram+$1E,6,scrram+$5D,5,6,5]
pause 1000
```

The rest of the large digits (0 thru F) are created in a similar fashion. Each is displayed, and then a final message is displayed. The final section of code displays, “Just imagine what you can do.”

```
‘ *** Final message from program before looping back to the top ***

lcdwrite rspin\epin,outc,[clear,home,scrram,”Just imagine what”,|
scrram+$40,”you can do!”]
pause 3000
goto main
```

With this custom character method, you can create just about anything on an LCD screen.

Next Steps

The projects that can result from this are endless. The thing to remember is, nothing stops you from redefining the custom characters in the middle of the program. For example, let’s say you want to display large characters, initially, and then later in the program want to create an animation using different custom characters.

After completing the large custom number characters, clear the LCD screen and then load new custom characters in CGRAM locations 0 - 7. From these new characters, you can create the animation. Since the custom characters load in CGRAM so fast, the person watching the display just notices a frame change from words to large digits to animation. I’ve seen custom characters that had the old Pacman character eating dots across the screen.

Conclusion

By no means am I declaring that I invented this custom-character method. In fact, there are numerous sites on the Internet that refer to creating custom characters on an LCD. Scott Edwards even incorporates this type of large-character generation into his PIC MCU-driven serial LCD modules. If you are really creative, you can probably create a whole animated cartoon on the LCD by constantly changing the custom characters. It will potentially take a lot of memory, but most graphic programs do.

I once again use the BasicATOM chip, because of the simplicity of the software and the low cost for any reader that wants to follow along by doing the projects. I have received many emails from readers asking me to pick a platform and stick with it. Because Basic Micro (creator of the BasicATOM) also has a Basic compiler, which is called the MBasic Professional compiler, this platform offers the reader the option to start cheap with the BasicATOM software, via a free download and a $20 BasicATOM chip (which should be cheaper by the time you read this). Eventually, most people will move to programming blank PIC MCUs, to save money on larger-volume projects.

Here's a tip for readers; there is a book available at the Nuts & Volts Web site called "Programming the PIC Microcontroller in MBasic" by Jack Smith (this has a similar title and the exact same publisher as my PICBASIC compiler book). Jack does a great job of detailing how to use the PIC16F877A MCU with MBasic Pro. Best of all—his book's CD includes a free version of the MBasic Pro compiler, which is limited to working with the PIC16F876(A). This offers you the option to take a 28-pin BasicATOM chip design and pretty easily move it to PIC16F876A—all for the cost of a book and a programmer (to load the .hex file into the PIC16F876A). Check it out.

Email me with your comments at chuck@elproducts.com, and thanks for all of the feedback I continue to get. I do like reading the feedback and try to respond to all of the emails as quickly as possible. See you next month.

Author Bio

Chuck Hellebuyck has written many articles and two books on PIC microcontrollers, which can be found at his Web site www.elproducts.com. Chuck is also a Field Applications Engineer for Microchip Technology Inc.

Note: PIC is a registered trademark of Microchip Technology Inc. in the USA and other countries. All other trademarks mentioned herein are property of their respective companies.