Programming the BetaBrite® LED electronic sign

by Gary Peek

It seems like nearly every microcontroller project needs some kind of user display. The LCD is one of the most common types of small displays microcontroller designers use for a number of reasons: LCD's are inexpensive, use very little power, and have many sources. Many LCD's have backlighting systems to allow viewing in a wide range of lighting conditions. Plus, a number of companies make interface boards to convert the parallel interface of an LCD into a much simpler serial port connection and protocol.

So it would seem like the LCD is the ideal display to use, and often it is. But what do you use when your microcontroller (or even your PC) needs a really large and bright display that can be seen from across the room? At that point you may need to consider an LED display. But large LED displays are not as common as LCD's, and most are expensive industrial units.

The BetaBrite electronic message display

Fortunately there is a large LED display that is not so expensive, and that is the BetaBrite

electronic message display. You may have already seen BetaBrite displays in operation because many businesses use them for promotional messages. The Missouri Lottery for example uses these displays on top of their lottery ticket vending machines, and many bars and taverns use them to announce specials as well.

The BetaBrite has a large display area 24 inches wide by 2 inches tall and uses an 80 by 7 array of bi-color LED's, allowing a number of colors to be displayed. It is reasonably priced considering its large size and numerous features, and it can be purchased locally in many cases. But most importantly, it has an RS-232 serial port.

So what's the catch, you ask? Well, the BetaBrite uses a special proprietary protocol with its serial port to display messages. This protocol is not terribly difficult to decipher, but it is definitely more difficult than connecting a serial LCD display to your microcontroller or PC.

About the BetaBrite

The BetaBrite® electronic message display is manufactured by Adaptive Micro Systems, Inc. (www.adaptivemicro.com) in Milwaukee, Wisconsin. The BetaBrite comes complete with a cable to connect it to your PC serial port, an AC power supply, and Windows software that allows you to easily display your messages. It also comes with a handheld remote control unit to program messages into the display without a computer. (The "Programming Manual" that comes with the BetaBrite does not describe the serial protocol. It's simply a user manual that tells you how to hook it up and program it with the handheld remote control.)

The BetaBrite can be purchased directly (www.BetaBrite.com) for $199, and is also available at Sams Club (item number 78029) for $169. Adaptive Micro Systems also manufactures a full line of industrial displays called the Alpha® Series, which is oriented toward distributorship and the resale of the displays, however, the closest equivalent Alpha Series display to the BetaBrite has a suggested price of $499! The Alpha Series displays may be the choice of discriminating industrial users, but for its low price, the BetaBrite is a very useful alternative for microcontroller users.

Programming the BetaBrite

Some years ago Adaptive Micro Systems did not include a serial port cable or software with the BetaBrite, but instead sold them as accessories that cost nearly as much as the BetaBrite. This prompted a number of curious individuals to "hack" the BetaBrite protocol, which was based on the Alpha Series displays. If you search on the web for information on the BetaBrite you will find a number of web sites that were created in the past that contain this information. Eventually

Adaptive Micro Systems made the Alpha protocol available and it is now on their web site.

The following information about the BetaBrite/Alpha protocol is based on the Alpha protocol document and is a small subset of these commands needed for simple message display. After trying some of the simple message display techniques described in this article, you may want to take a look at the full protocol and learn to program the BetaBrite in more complex ways. Keep in mind however that the Alpha protocol is not exactly the same as the BetaBrite protocol, and that some Alpha protocol commands do not work with the BetaBrite.

The protocol

The hardware communication parameters for the BetaBrite are 1200, 2400, 4800, or 9600 baud, 1 start bit, 7 data bits, 1 stop bit, and even parity. The reason that a number of baud rates can be used is that the first part of a command is to send a number of "null" characters so that the display can lock on to the baud rate.

Once the BetaBrite begins receiving characters, it must continue to receive characters at least once per second, or the command will timeout and be ignored. While the display is receiving characters the display will blank, and when the command is complete the new message will be displayed.

The Alpha protocol includes command options for addressing multiple displays connected in a network and displays that have more than one line, so some of the command options in this simplified protocol are set to default selections or selections that have no effect.

And lastly, BetaBrite displays can contain multiple messages in its memory. This simplified protocol uses only "Message A", meaning the other messages can be entered and saved using the handheld remote control unit. The BetaBrite displays the last message to be entered or selected, so the two methods of entering and displaying messages can co-exist to some degree.

Some control characters defined

Since a BetaBrite message command includes ASCII control characters that cannot be printed, we must represent them in another manner. The following is a list of the control codes used and how we will represent them. Also included is a representation of options and control sequences that have a number of possible values.

<NUL> is the "null" character, 0 decimal or 00 hexadecimal
<SOH> is the "start of heading" character, 1 decimal, 01 hexadecimal

<STX> is the “start of text” character, 2 decimal, 02 hexadecimal
<ETX> is the “end of text” character, 3 decimal, 03 hexadecimal
<EOT> is the “end of transmission” character, 4 decimal, 04 hexadecimal
<ESC> is “escape”, 27 decimal, 1B hexadecimal
<FS> is “file separator”, 28 decimal, 1C hexadecimal
<SP> is the “space” character, 32 decimal, 20 hexadecimal

<DM> is the “Display Mode” option character. The following modes are the most likely modes to be used for simple messages. See the Alpha protocol document for an explanation of all of the modes.

a (a = rotate, decimal 97, hex 61)
b (b = hold, decimal 98, hex 62)
c (c = flash, decimal 99, hex 63)
m (m = scroll, 109, 6D hex)
o (o = auto, 111 decimal, 6F hex)
t (t = compressed rotate, 116 decimal, 74 hex)

<MSG> is the text message you wish to be displayed and consists of up to 125 normal text characters (minus the characters used by special control sequences).

The following pairs of characters can be inserted anywhere in the message and will change the color of the text following it. The message can contain multiple color changes.

<FS>1 (file separator and the number 1, 49 decimal, 31 hex = red)
<FS>2 (file separator and the number 2, 50 decimal, 32 hex = green)
<FS>3 (file separator and the number 3, 51 decimal, 33 hex = amber)
<FS>4 (file separator and the number 4, 52 decimal, 34 hex = dim red)
<FS>5 (file separator and the number 5, 53 decimal, 35 hex = dim green)
<FS>6 (file separator and the number 6, 54 decimal, 36 hex = brown)
<FS>7 (file separator and the number 7, 55 decimal, 37 hex = orange)
<FS>8 (file separator and the number 8, 56 decimal, 38 hex = yellow)
<FS>9 (file separator and the number 9, 57 decimal, 39 hex = rainbow 1)
<FS>A (file separator and the letter A, 65 decimal, 41 hex = rainbow 2)
<FS>B (file separator and the letter B, 66 decimal, 42 hex = mixed)
<FS>C (file separator and the letter C, 67 decimal, 43 hex = auto)

There are many types of control sequences besides the color changes. See the Alpha protocol document for an explanation of all of the control modes and sequences.

The message command

Now that we have identified a way to represent the individual characters and control sequences in a message command, we can show the entire command. A command for displaying a message will

contain the following:

<NUL>	(5 to 20 null characters)
<NUL>	
<NUL>	
<NUL>	
<NUL>	
<SOH>	(start of heading character)
Z	(Type Code, Z = broadcast to all displays, 90 decimal, 5A hex)
00	(Address Field, does not matter if Type Code is "broadcast", this is two “zero” characters, 48 decimal, 30 hex)
<STX>	(start of transmission character)
A	(Command Code, A = write and display message, 65 decimal, 41 hex)
A	(Message number, A = Message “A”)
<ESC>	(Display Line on multiline display, does not matter on BetaBrite)
<SP>	(space character = “middle line”)
<DM>	(display mode character)
<MSG>	(your message including modifying control sequences)
<ETX>	(end of transmission character)

To clear the display simply send a message command with no characters in <MSG>.

A BASIC program to send messages

One way to make sure that you understand how to send message commands to the BetaBrite before connecting it to a microcontroller is to replace the software that is included with the display with your own program. The following is a simple BASIC program that can be used with QBasic, QuickBasic, or GWBasic. Connect the cable that is included with the BetaBrite display between the display and a serial port on your PC. Com 1 is used in the program but can of course be changed if required. With this program you can easily experiment with the many variations in the message command, and it can become a good base for when you begin to experiment with other commands listed in the Alpha protocol document.

```
10 OPEN "COM1:2400,E,7,2,CS,DS,CD" AS 1     :REM OPEN COM PORT
20 FOR A=1 TO 10:PRINT #1,CHR$(0);:NEXT A   :REM DETERMINE BAUD RATE
30 PRINT #1,CHR$(1);                        :REM SOH
40 PRINT #1,"Z00";                          :REM BROADCAST TO ALL, ADDRESS 0
50 PRINT #1,CHR$(2);                        :REM STX
60 PRINT #1,"AA";                           :REM WRITE AND DISPLAY MESSAGE A
70 PRINT #1,CHR$(27)+" ";                   :REM DISPLAY LINE = CENTER
80 PRINT #1,"b";                            :REM DISPLAY MODE = HOLD
90 PRINT #1,CHR$(28)+"2";                   :REM COLOR = GREEN (OPTIONAL)
100 PRINT #1, "TIME=";                      :REM PART OF MESSAGE
110 PRINT #1,CHR$(28)+"1";                  :REM COLOR = RED (OPTIONAL)
120 PRINT #1,TIME$;                         :REM REST OF MESSAGE IS PC'S TIME
130 PRINT #1,CHR$(4);                       :REM EOT
140 IN$=INKEY$:IF IN$=CHR$(27) THEN 200     :REM EXIT IF ESCAPE KEY HIT
```

```
150 T$=RIGHT$(TIME$,1)                          :REM GET CURRENT SECOND
160 IF T$=RIGHT$(TIME$,1) THEN 160              :REM WAIT FOR NEXT SECOND
170 GOTO 20
200 CLOSE
```

Connecting a BetaBrite to your microcontroller

The cable that comes with the BetaBrite is designed to easily connect it to the serial port of an IBM PC compatible. It has an RJ11 style connector to plug into the BetaBrite, and a DB9F connector for the PC serial port. The BetaBrite requires only that the transmit, receive, and ground signals are connected to work properly, (even if you include the commands that tell it to send information back). Connecting it successfully to any other host device with an RS-232 serial connection simply requires that the transmit and receive signals go to the proper signals on the device and are not reversed.

Making it even simpler

For those of you who are interested in making the BetaBrite even easier to program through its serial port, Industrologic, Inc. (www.industrologic.com) makes an interface board called the RS51-SPR-BETA that takes care of the special protocol for you. Instead of sending control characters, you can send simple character strings followed by a carriage return. Commands exist for changing display colors and other display features, but they too are much simpler than the BetaBrite protocol. The RS51-SPR-BETA also accepts commands at a number of user selectable baud rates at 8 data bits, 1 stop bit, and no parity, which may make interfacing to your host device much easier. The RS51-SPR-BETA has DB9 connectors and is designed to make it easy to place between a PC and the BetaBrite.

About the author

Gary Peek is the President and co-founder of Industrologic, Inc., a manufacturer of microcontroller based industrial data acquisition and control products. He can be contacted at Industrologic at (636) 723-4000 or peek@industrologic.com. Other Industrologic products can be found on their web site at www.industrologic.com.