

Control Your World by bulding an Interface to a LED Sign

By Michael Simpson

Important

As featured in September of Nuts and Volts

**Requires Zeus
Version 1.0 or greater**

**Pick up and issue at
<http://www.nutsvolts.com>**



I was doing a robotics show and needed a way to both attract contestants and display contest results. I had seen a couple LED signs and decided one would be perfect for this kind of application. My only requirement was that the cost had to be reasonable and the sign needed to be easy to transport and hookup. I needed a sign with an RS232 interface so I could display real time contest results.

The most prevalent LED sign was one called the BetaBrite. They were stocked at most Office supply stores. The best bargain was when I found them on sale at one of my local Warehouse stores. They cost me less than \$100 each. You can get more information about these signs at www.BetaBrite.com.

The signs are manufactured by a company called Adaptive Micro Systems Inc. They manufacture several signs, everything from single line indoor signs to multi-line outdoor signs.

The BetaBrite Sign comes with an AC adapter for power, IR Remote control and a CD that contains a simple program for uploading messages to the sign from your PC.



Figure 2

Being the kind of person I am I was compelled to write my own interface. I figured that once I learned the protocol I could interface to the display with microcontrollers, Laptops or my Pocket PC.

Most of these signs come with an RS232 interface, which is important if you want to interface with devices like Pocket PC's or microcontrollers. If this is your goal don't purchase a USB controlled sign. The BetaBrite sign comes with a 25 foot 9-pin RS232 connector and is set up as a DCE device so this connector can be plugged directly into the PC.



Figure 3

If you want to connect it to a pocket PC you will need a null modem and gender changer or a Bluetooth RS232 adapter. For a connection to a microcontroller I have created an application note that can be found on the Kronos Robotics Web site at www.kronosrobotics.com

Research the Protocol

I started researching the interface protocol by searching the NET. You can find a great deal of information on various websites, but the best source is from the Adaptive web site at www.adaptivedisplays.com

The signs use a protocol called **Alpha Sign Communications Protocol**. You can find a complete description of the protocol at <http://www.ams-i.com/Pages/97088061.htm>

The protocol can be a bit complicated as it supports networking devices. What I'm going to do in this article is exploit a feature of the protocol to create a very simple interface.

Programming Protocol

I have been doing PC and Pocket PC development for sometime now and a while back I decided to create a simple development platform that would make interfacing to devices and robotics much quicker and easier. With the help of other engineers we came up with a development platform called Zeus. There are both Desktop and Pocket PC versions of Zeus available.

You can compile and run the most of the programs presented in this article with ZeusLite. ZeusLite will let you debug, compile and test the code in this article. You can even create a stand alone executable that you can distribute to others.

ZeusLite has one requirement. You must have the .Net Framework installed on your Windows machine.

The code we used is quite simple and can be adapted to just about any programming language that supports RS232 communications.

The Alpha Protocol

The Alpha protocol has some pretty advanced features such as networking and message labeling. It also supports uploading graphic images call DOTS. In order to make the explanation and examples as simple as possible we will drastically simplify the interface by using certain simple aspects of the protocol.

Before we send a single byte to the sign we need to open up a comport with the correct settings. The protocol supports the following baud rates. 1200, 2400, 4800, and 9600 with the following settings: 7 Bits, 1 Start, 2 Stop and Even Parity. We use the Zeus ComOpen command to open the com port.

Const Ch1 1

ComOpen(Ch1,baud=9600,port=1,parity=2,bits=7,stop=2)

Zeus can control 5 Async connections at once. These are called channels; hence the Ch1 constant. You will need to change the port setting from 1 if you are using anything else. For instance, if you are using Bluetooth on an IPAQ Pocket PC to connect to a Bluetooth RS232 connector, you will use **port=8**.

Once the port is open we have to tell the sign we are ready to communicate with it. I call this the ready header.

```
ComOutput Ch1,chr(0)+chr(0)+chr(0)+chr(0)+chr(0)  
ComOutput Ch1,chr(1)  
ComOutput Ch1,"Z00"
```

We start with 5 null characters (Value 0). This sets the sign's baud rate to the rate you are transmitting. (Note that if you are dealing with a language that can't send Null characters, you may also send 5 1's.) We then send a SOH (Start Of Header, Value =1) character. After this we can do a great many things but in our case we are going to send "Z00". This tells the sign that we are addressing all signs on the bus.

The Alpha protocol supports sending checksums. However to make the interface simpler we are not going to utilize this feature.

With the header sent we can now send a command. The Alpha protocol supports several command types, but we will only be using a couple of them. In this case we are going to send a Priority Text Message. What is neat about this message is that it bypasses the internal file system so it's simple and fast. The down side is that you can only send 125 bytes, but since we are going to be sending real time messages this is perfect.

To send a Priority Text Message you start by sending a STX (Start Text, Value = 2) character. We then send a command code of "A" and a File label of "0".

```
ComOutput Ch1,chr(2)+"A0"
```

This tells the sign that we are sending a Priority Message.

You then send your message text followed by an EOT (End of Transmission, Value = 4)

```
ComOutput Ch1,"Nuts and Volts"  
ComOutput Ch1,chr(4)
```

Once the EOT character has been sent the message will display. By default the colors and display modes will be in auto format so you will see the message change. We will change this setting later.

Zeus will automatically shutdown the comport for you but it's always smart to do it yourself with the CloseCom command.

CloseCom Ch1

Now wasn't that simple? Program 1 shows the complete code example we just discussed.

```
'Simple BetaBrite Priority Message Sample
func main()

  'Open the ComPort
  Const Ch1 1 'Zues Channel 1
  ComOpen(Ch1,baud=9600,port=1,parity=2,bits=7,stop=2)

  'Tell the Sign we are ready to communicate
  ComOutput Ch1,chr(0)+chr(0)+chr(0)+chr(0)+chr(0)
  ComOutput Ch1,chr(1)
  ComOutput Ch1,"Z00"

  'Send Priority Text Message
  ComOutput Ch1,chr(2)+"A0"
  ComOutput Ch1,"Nuts and Volts"
  ComOutput Ch1,chr(4)

  CloseCom Ch1

endfunc
```

Program 1

We are going to add a few new features to the program, so in order to simplify the code let's take all the current code and place it in a function called `BBSendText`. We will set the function up so that we can pass the com port and the text of the message to be displayed. Then all we have to do is call this function each time we want to send new data.

We are also going to add a function called `BBSetTime`. This function lets you send a 4 character string to set the current clock located inside the sign. Yep, that's right, the sign has its own clock. There is even a control code that you can insert into your text and the current time will be substituted for that code.

`BBSetTime` will open the com port, send a Special Function command, and the 4-digit time string (24 hour format). The function will then send the EOT code and close the com port.

Note that you need only set the time once; then you can comment the code out. The clock will keep the current time even when new commands are sent. Once power is

removed the clock will stop. When power is restored the clock will start again where it left off and the last message you sent will be displayed.

```
'BetaBrite Real Time Message Sample
func main()
  BBSendTime(1,"1215")

Loop:

  BBSendText(1,"Nuts and Volts")
  pause 1000
  BBSendText(1,"Is Real Cool")
  pause 1000

  goto Loop

endfunc

func BBSendText(tPort as integer,ttext as string)
  Const Ch1 1 'Zues Channel 1

  'Open the ComPort
  ComOpen(Ch1,baud=9600,port=tPort,parity=2,bits=7,stop=2)

  'Tell the Sign we are ready to communicate
  ComOutput Ch1,chr(0)+chr(0)+chr(0)+chr(0)+chr(0)
  ComOutput Ch1,chr(1)
  ComOutput Ch1,"Z00"

  'Send Priority Text Message
  ComOutput Ch1,chr(2)+"A0"
  ComOutput Ch1,ttext
  ComOutput Ch1,chr(4)

  ComClose Ch1
endfunc

func BBSendTime(tPort as integer,Time as string)
  Const Ch1 1 'Zues Com Channel 1

  'Open the ComPort
  ComOpen(Ch1,baud=9600,port=tPort,parity=2,bits=7,stop=2)

  'Tell the Sign we are ready to communicate
  ComOutput Ch1,chr(0)+chr(0)+chr(0)+chr(0)+chr(0)
  ComOutput Ch1,chr(1)
  ComOutput Ch1,"Z00"

  'Send Time
  ComOutput Ch1,chr(2)+"E "
```

```
ComOutput Ch1,Time
ComOutput Ch1,chr(4)

ComClose Ch1
Endfunc
```

Program 2

You will notice that now all we need to do is use the BBSendText along with various pause commands and we can display whatever we want to in real time.

Notice the second BBSendText function call just after the **Loop** label in Program 2. Change it to:

```
BBSendText(1,chr(19))
```

What we are doing is sending a control code that will tell the sign to display the time. In this instance the control code is 19. Other commands may require more than one sequence of control characters.

Now for Program 3. We will add a very powerful function to the previous program. This function is called ConvertCodes and will take our text message, along with special commands, and create some rather cool effects. You can now control the colors and scrolling affects of the messages.

Here is a list of the commands that you may insert into your message:

Colors

```
[RED] [GREEN] [AMBER] [DRED] [DGREEN] [BROWN] [ORANGE] [YELLOW]
[RAINBOW1] [RAINBOW2] [COLORMIX] [COLORAUTO]
```

Fonts

```
[SMALL] [SMALLWIDE] [LARGE] [LARGEWIDE] [FANCY] [LARGE3D]
[LARGEWIDE3] [LARGEWIDE4] [PROP] [FIXED]
```

Roll

```
[ROLL UP] [ROLL DOWN] [ROLL LEFT] [ROLL RIGHT] [ROLL IN] [ROLL OUT]
```

Wipe

```
[WIPE UP] [WIPE DOWN] [WIPE LEFT] [WIPE RIGHT] [WIPE IN] [WIPE OUT]
```

Other

```
[ROTATE] [ROTATESMALL] [TIME] [HOLD] [AUTOMODE] [FLASH]
[TWINKLE] [SPARKLE] [SNOW] [INTERLOCK] [SWITCH] [SLIDE] [SPRAY]
[STARBURST]
```

Graphics

[SLOTMACHINE] [NOSMOKING] [DRINK] [ANIMAL] [FIREWORKS]
[GRAPHIC1] [BOOM]

Speed

[SPEED1] [SPEED2] [SPEED3] [SPEED4] [SPEED5]

To insert the effect just add it to your messages as in:

BBSendText(1,"[ROTATE]Nuts and Volts")

Program 3 is a bit too large to include here but it is included with all the source code proved in this article.

There is a lot you can do with one of these signs. The animated text and graphics will be sure to attract attention wherever you decide to use them, but the real power is going to be real time updates.

As a bonus I have created a program called BetaBrite TNG shown in Figure 4. It is a cool little program that will make it very easy to setup messages for your sign. In order to compile this program you will need ZeusPro. The compiled programs for both the pocket PC and Desktop are included for those of you who don't want to compile the program.

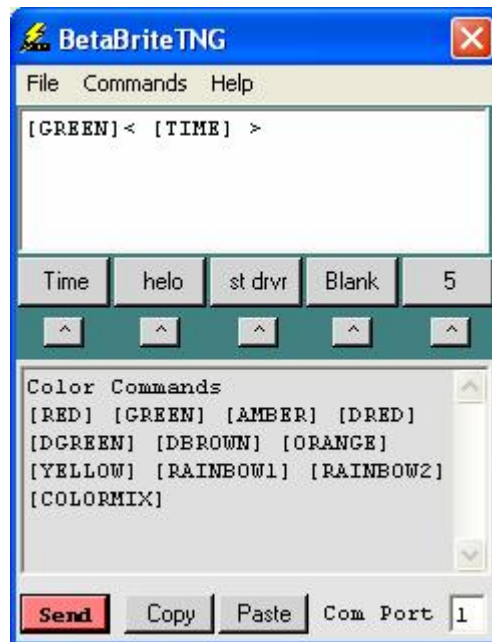


Figure 4

Going Further

The BetaBrite sign comes with its own software so why would we want to create our own interface? Well for a couple of reasons. The most important one is we now have the ability to display real-time information. We could use Zeus to go out and query a web site and then display some important piece of information so the public could see it. Things like stock quotes, web hits, or sales information.

In my lab, I use one to display the time. When a visitor arrives the sign beeps and displays a message “Arrival”.

You could use the interface to display real-time race times or event times.

I recently started teaching my teenage daughter how to drive. I make her keep it under 40 MPH and most other drivers will tailgate even though we are under the posted speed limit. Well I solved the problem. If any one gets too close I just load the message shown in **Figure 5** and you wouldn't believe how fast they back off.



Figure 5

I won't tell you the other messages I have loaded

I have also created an application note showing how to connect a microcontroller to one of these signs. You can find that at the Kronos Robotics website at www.kronosrobotics.com

Experiment and have fun, and be sure to visit the “Control Your World” forum at: <http://www.kronosrobotics.com/forums/viewforum.php?f=21>

Web Links

KronosRobotics website: www.kronosrobotics.com

KRMicros website: www.krmicros.com

Adaptive website: www.adaptivedisplays.com.

BetaBrite website: www.BetaBrite.com.