Model 7810 8-to-1 HD-15 Switch with 10 BaseT/100 BaseTX LAN Access

• IP Addressable For Critical Network Alternate-Path Switching!

INTRODUCTION:

The Model 7810 8-to-1 HD-15 Switch with 10 BaseT/100 BaseTX LAN Access, allows the user the capability of sharing a single port HD-15 interface device connected to the "COMMON" port among eight other devices connected to the Channel 1-8 ports with local and remote access functionality. Remote access can be accomplished by a 10-Base-T/100 BaseTX connection.

Electro Standards offers hundreds of copper and fiber optic data network switches. Customized switches are manufactured every day. Contact Electro Standards, 401-943-1164, eslab@ElectroStandards.com

Application Note for Model 7810

Remote Control via Ethernet using Visual Studio and Socket Programming

1. Remote Switching of the Model 7810

The 7810 switch has a 10 Base-T/100 interface to the outside world allowing a user to remotely control its function. The user needs to construct the program logic to perform the following functions…

- Open an Ethernet connection to the 7810 switch.
- Send log in command over Ethernet connection
- Send switch commands over Ethernet connection
- Send lockout front panel buttons command over Ethernet connection if it is desired to remove external access to switching
- Send set and confirm password commands if additional security is desired
- Send Status query command to update controlling computer as to the current
  - o Switch position
  - o Front Panel Lockout status

Sending the commands is relatively straightforward. The section entitled "System Details for Programmers" will give you additional information on what to expect for replies from the switch and what you need to do in each case.

1.1 Command Set Table

The table below is the complete command set for the 7810 switch

Command	Parameter	Function	Response	Comments
CTRL-S (13H)	1 digit position code (1-8)	Switch to position	"00x POSITION: x", or "01x POSITION:x, LOCKED", or "099 INVALID POSITION CODE"	Works only when logged in
CTRL-P (10H)	N/A	Query for Status	"00x POSITION: x", or "01x POSITION: x, LOCKED"	Works only when logged in
CTRL-L	N/A	Lockout front panel	"01x POSITION: x, LOCKED"	Works only when logged in
CTRL_U	N/A	Unlock front panel	"00x POSITION: x"	Works only when logged in
CTRL-E	6 digit password	login	"201 WELCOME!", or "102 INVALID PASSWORD!"	Works only when logged out
CTRL-X	N/A	Logout (a.k.a session end)	"104 BYE!"	Works only when logged out
CTRL-N	6 digit password	New password	"301 PLEASE CONFIRM NEW PASSWORD!"	Works only when logged out
CTRL-M	6 digit password	Confirm new password	"303 NEW PASSWORD ACCEPTED!", or"302 PASSWORD MISMATCH!"	Works only when logged in
CTRL-V	N/A	Firmware version and compile date	"901 M7810 firmware ver x.xx, Compiled MMM DD YYYY,"	Works anytime

TABLE 1. Model 7810 Command Set

1.2 Ethernet Access

The 7810 switch is accessible through port 10001 at the IP Address assigned to it in your network configuration. The hardware is configured such that port 10001 will accept control characters and parameters via raw sockets or using the slightly more complex telnet forms of communication. In this way the user has the choice of using, for example, the raw sockets form of communications with the freely available Hyper-Terminal ™ program, or the telnet utility available via the windows command line.

You would configure a Hyper-Terminal session as shown below…

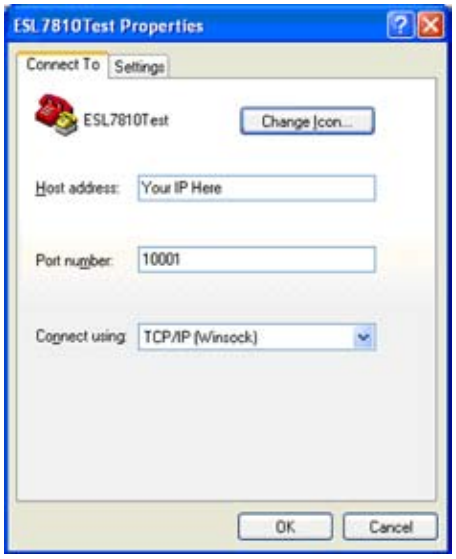

Figure 1. Hyper-Terminal Configuration for Raw Socket Access to the Model 7810

And as soon as you log-in you will see the following in the status windows of the hyperterminal utililty …

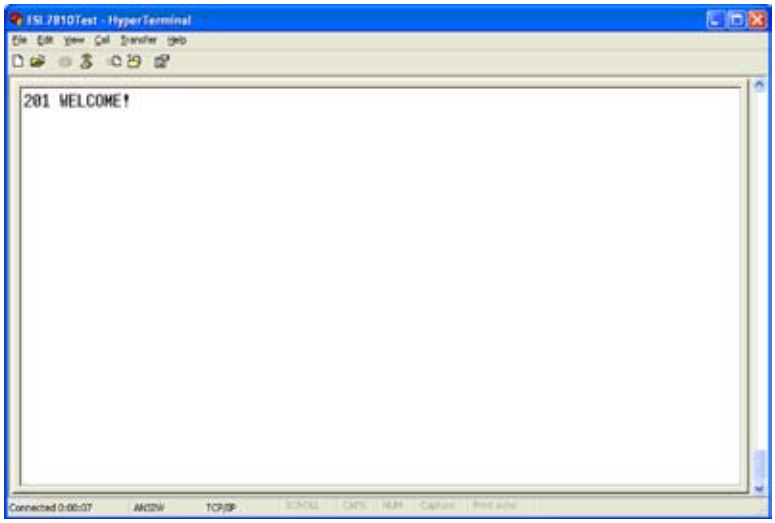

Figure 2. Appearance of Log In response from the Model 7810 on Hyper-Terminal

For telnet access simply do the following…

- Open a command prompt window
- At the prompt type "telnet <IPAddress> 10001
- Followed by the Enter key
- When you logon to the unit you will see…

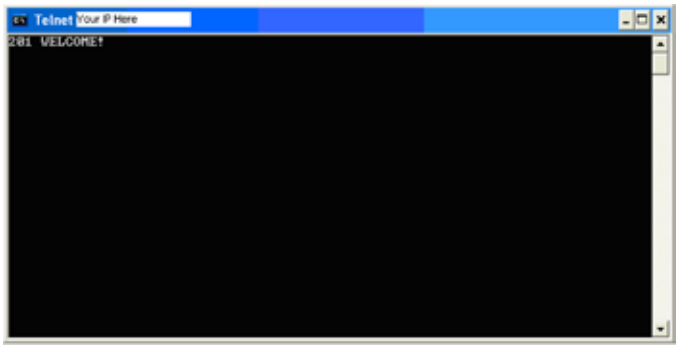

Figure 3. Appearance of Log In response in Telnet Window

As the programmer, having a means to manually send commands to the switch and viewing how it responds provides you with valuable insight on the unit's operation.

1.3 Operational Details

The 7810 switch has two types of timeouts the programmer should be aware of…

- Inter-character timeout – if the time delay between consecutive characters received by the switch for an incomplete command is more than 2.5 seconds, the session will end and the user logged out. The unit will automatically send the following message to the remote user if the inter-character timeout occurs…"401 TIMED OUT, TRY AGAIN".
- Session Timeout – if the time delay between commands exceeds 2 minutes, the session will end and the user logged out. The unit will automatically send the following message to the remote user if the session timeout occurs…"103 SESSION TIMED OUT!".

1.3.1 Real time Switching

If the user transmits the "CTRL-S" character followed by a single digit (1-8) and the operation successful, the unit will reply with one of the responses detailed in Table 1.

1.3.2 Status

If the user transmits the "CTRL-I" character and the operation successful, the unit will reply with one of the responses detailed in Table 1.

1.3.3 Front Panel Buttons Lockout

If the user transmits the "CTRL-L" character and the operation successful, the unit will respond with the response detailed in Table 1.

1.3.4 Restoration of Front Panel Buttons

If the user transmits the "CTRL-U" character and the operation successful, the unit will respond with the response detailed in Table 1.

1.3.5 Login

If the user transmits the "CTRL-E" character followed by six digits, and the operation successful, the unit will respond with the response detailed in Table 1.

1.3.6 LogOut

If the user transmits the "CTRL-X" character and the operation successful, the unit will respond with the response detailed in Table 1.

1.3.7 Set New Password

If the user transmits the "CTRL-M" character followed by six digits, and the operation successful, the unit will respond with the response detailed in Table 1. This action must be successfully followed by the Confirm New Password command in order for the new password to be accepted.

1.3.8 Confirm New Password

If the user transmits the "CTRL-M" character followed by six digits the unit will respond with one of the responses detailed in Table 1.

1.3.9 Retrieving firmware and compilation date

If the user transmits the "CTRL-U" character and the operation successful, the unit will respond as detailed in Table 1.

1.3.10 Reset Password to Default

When logged out, if the user transmits the "CTRL-R" character followed by the six digit character sequence "123456" and the operation successful, the unit will respond as detailed in Table 1.

1.4 System Details for Programmers

At this point the command protocol for the 7810 has been covered and the user has been shown two ways to access the switch remotely in order to manually exercise the command set. Next, comes the task of programmatically performing the same tasks.

1.4.1 Using Sockets in Visual Studio 2005

Visual studio 2005 has simplified the task of permitting the user to communicate with remote devices via Ethernet. The following basic steps are required…

- Open a connection between the local PC and the remote endpoint (7810).
- Specify the receiving data and send data callback functions
- Send commands and receive the replies.
- Close the connection, at your discretion, if it will be a long time between accesses. On the other hand, to avoid the session timeout you may want to send a periodic status command.

The following subsections provide C# examples of code used to facilitate the major core functions of Windows Sockets. Knowledge of Windows™ programming, multi-threading, and callback functions is assumed. If you have programmed Sockets in earlier versions of Visual Studio, you will find the .NET implementation much simpler since it handles much of the threading sub-framework and callback mechanism internally. In Visual C++ 6.0 and earlier the programmer was required to write the threading sub-framework and callback mechanism manually unless they purchased a third-party module.

1.4.1.1 Initialization of the Ethernet Port and opening a connection

The following code can be used to open a connection to a remote Ethernet endpoint…

```
public Int32 OpenConnection( string ipAddress, string port )
        {
            Int32 iRetValue = (Int32)ErrorCodes.NO_ERRORS;
            //
            //  Initialize some parameters.
            //
            iSwitchPort = Int32.Parse(port);
            IPAddress = ipAddress;
            IPHost = Dns.GetHostEntry(IPAddress);
            IPAliases = IPHost.Aliases;
            IPAddresses = IPHost.AddressList;
            // create an instance of Socket object
            eslSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            // Create a new IP endpoint
            iep = new IPEndPoint(IPAddresses[0], iSwitchPort);
            // Be sure we set the IO as non-blocking
            eslSocket.Blocking = false;
            // Set callback to be called once the connection process has completed.
            callbackProc = new AsyncCallback(ESL7810SocketConnectionCallback);
            // Begin the Asynchronous connection to the designated endpoint.
            eslSocket.BeginConnect(iep, callbackProc, eslSocket);

            return iRetValue;
        }
```

Once the connection is successful, the operating system signals the callback function where the user can…

- Successfully complete the connection, and
- Setup the callback used to receive data.

```
private void ESL7810SocketConnectionCallback( IAsyncResult iar )
        {
            Socket ESL7810Socket = (Socket)iar.AsyncState;

            try
            {
                // Successfully complete the connection
                ESL7810Socket.EndConnect(iar);

                // Start the listener thread that listens for incoming data
                ESL7810Socket.BeginReceive( rcvBuffer, 0, rcvBuffer.Length, Socket-
Flags.None, ESL7810ReceiveCallback, ESL7810Socket);
            }
            catch (SocketException exSocket)
            {
                //
                //  Unable to make a connection.
                //
            }
        }
```

When you've completed the operation, the socket connection can be closed and resources released by the command.

```
public void CloseConnection()
 {
      eslSocket.Close();
 }
```

1.4.1.2 Receiving Data

In the previous section, you executed a BeginReceive function which started a listener thread. When data is received the "ESL7810ReceiveCallback" function will be called. An example of such a function is…

```
private void ESL7810ReceiveCallback(IAsyncResult iar)
{
    Socket ESL7810Socket = (Socket)iar.AsyncState;
    int iBytesReceived = ESL7810Socket.EndReceive(iar);

    //
    // We have received a response and now must extract the bytes in the form of a
string
    //

    string strStringData = Encoding.ASCII.GetString(rcvBuffer, 0, iBytesReceived);

    //
    //  Insert your code here to parse received data. Data should be a response
    //  to the command issued.
    //

 }
```

1.4.1.3 Sending Data

Sending data is done at the discretion of the user and can be invoked using the following sample code…

```
private void ESL7810SendMessage( string strMessage )
        {
            Byte[] bMessage = new Byte[strMessage.Length];

            //
            //  Now let's convert the individual characters in the string
            //  to bytes to make the message compatible with the asynchronous
            //  sending framework.
            //

            for (int i = 0; i < strMessage.Length; i++)
            {
                Byte b = Convert.ToByte(strMessage[i]);
                bMessage[i] = b;
            }

            //
            //  If here then we've taken the message and
            //

            IAsyncResult iar2 = eslSocket.BeginSend(bMessage, 0, bMessage.Length,
SocketFlags.None, callbackProc, eslSocket);
            eslSocket.EndSend(iar2);
        }
```

Please Note: The socket functions in this entire section should be wrapped in more extensive exception handling in the event there is a connection problem. The intellisense feature which is part of the Visual Studio code editor will tell you which exceptions could be returned by a function and you should handle each one of them.

1.4.1.4 Implementation Details

In this section, the reader has been introduces to several example blocks of code to facilitate the opening and closing of an Ethernet connection to a remote host. Furthermore, additional code examples were shown to send and receive data from the endpoint.

This section provides the programmer with high level suggestions as to how to integrate this code into a larger framework.

1.4.1.4.1 Sending Commands

Sending commands is a relatively simple task. It is done at the discretion of the master program and normally done by simply making a call to a public function, which in turn calls the private function detailed in section 1.4.1.3. As seen in 1.4.1.3, the send message function expects to receive a string representation of the completed message. Therefore, your public function must…

- know which message is to be constructed,
- assemble the complete message into a string object using the data in Table 1, and
- Call the function in section 1.4.1.3 to have it transmitted to the switch.

The following flowchart describes a sample flow of logic…

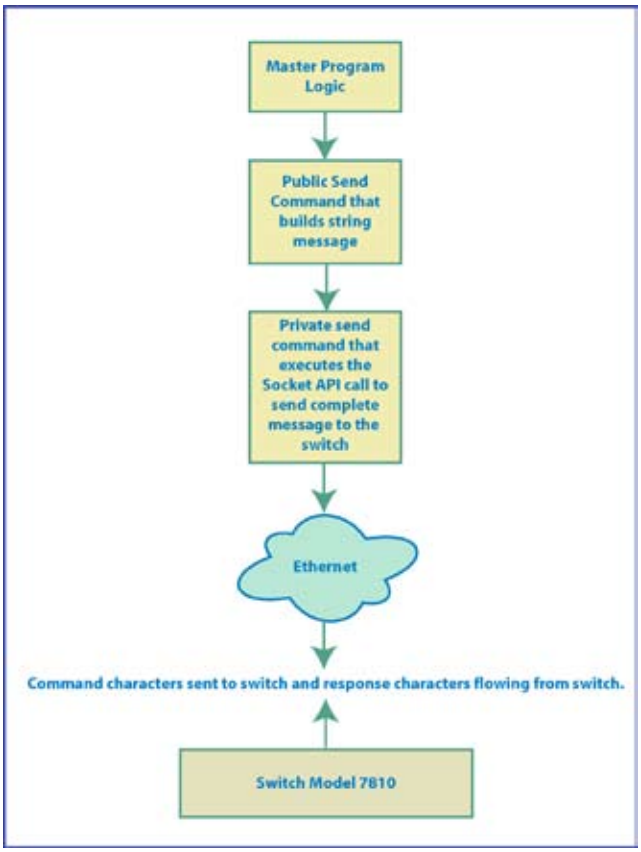

Figure 4. Simple Send Data program flow

1.4.1.4.2 Receiving Data

When using callbacks, the user's code doesn't have to poll the Ethernet receive buffer, but simply wait until the callback is executed where data becomes available in the receive buffer. Please keep in mind, as with any multi-threaded environment, that your master code is designed such that time sensitive tasks are not corrupted when the callback is asserted. Two types of information receipt are possible when dealing with the 7810 switch…

- Intelligent switch feedback
- Simple switch feedback

The following flowchart illustrates an oversimplified view of the way the data receiving code/thread works in partnership with main program code…

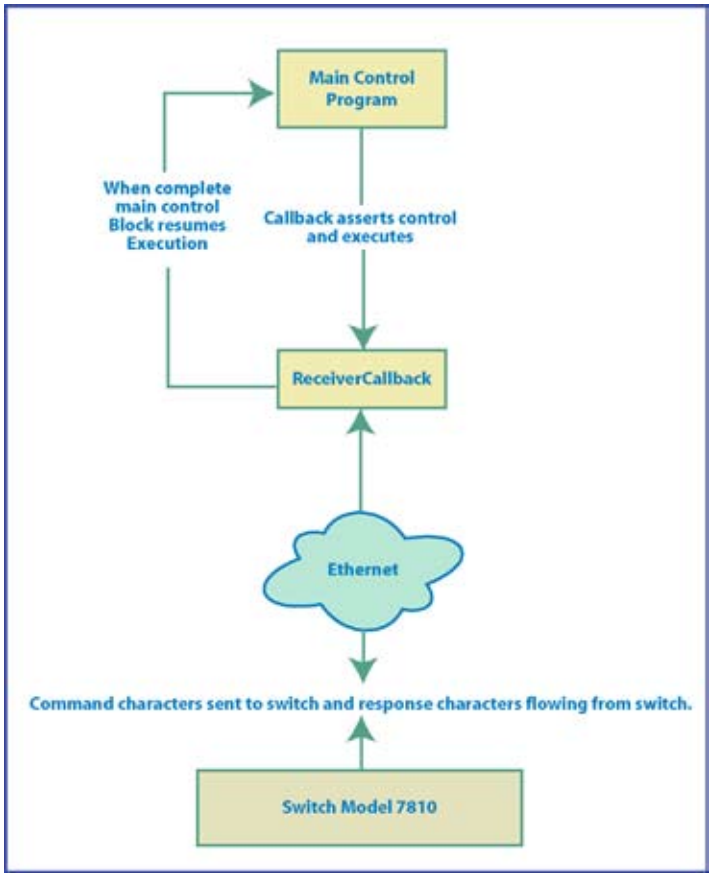

Figure 5. Simple Receive data program flow

1.4.1.4.2.1 Intelligent switch feedback

Intelligent feedback assumes the program will parse the reply transmitted back from the switch and programmatically make decisions based on the findings. For example, consider the timeouts introduced back in section 1.3. The programmer should be mindful of the fact that the switch will timeout if the gap in time is greater than 2 minutes between messages and take the following two possible courses of actions…

- Periodically send a status message to the switch (< 2 minute interval) to prevent the timeout from occurring and simply introduce the desired command in place of the status message.
- Send a command to the switch (switch position, etc) without prior knowledge of the switch timing out…
  - o Parse the response to find out if the switch has ended your session,
  - o If session was ended, Send the login command from Table 1 and verify the response is a successful login. Then follow up with the original command.
  - o If session was not ended then the command should've executed successfully. Parse the response for indications of success or failure.

In an intelligent feedback implementation, much of the switch usage intelligence is programmed into the code and not the user itself. An example of this situation is a monitoring system or simulator.

1.4.1.4.2.2 Simple switch feedback

Simple feedback assumes the master program (A GUI front end for example) simply sends the command and passes the response back to a User Interface (UI). It is then up to the user to decide what to do with the information. The user must interpret the condition of the switch from comparing the response to those found in Table 1 and take the appropriate action via, buttons for example, on the UI. A simple feedback implementation would be a simple GUI to exercise the switch's functionality. The GUI may consist of a group of buttons and other controls to actively issue commands and a window to display the text being transmitted back from the switch. An example of such a GUI is shown below…

- One button for each switch position. The user clicks the button and the command to set the switch position to that particular channel is sent to the switch via a public send message described in section 1.4.1.4.1.
- One button of Log In/Log Off from the box. The GUI would know the current state of the box and decide the appropriate course of action. Clicking this button would send either the log on or log out message. In addition to logging in, the GUI could follow up with sending the Firmware version and compilation date request command and determine, before proceeding further, if the correct firmware is loaded.
- One button represents the action of setting and confirming the establishment of a new password. This means the GUI can manage first setting and then confirmation of a new password.
- One button represents the actions of toggling the front button panel lockout. Click once to send the command to lock the front panel, and again to unlock it.
- One button represents the action of querying the switch for position and lockout status. See table 1 for responses.
- A RichEditControl is the window in which the response from the switch will be displayed and viewed by the user. It occupies the entire upper half of the GUI.

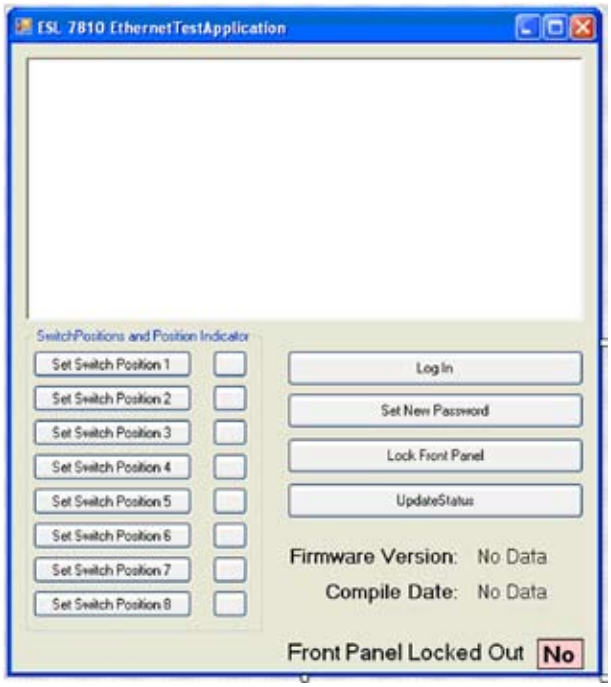

Figure 6. Sample GUI to exercise send/receive data functionality.

In code, when a button is clicked, another callback is typically used to facilitate the user's request to send commands to the switch. The example below is for switching positions…

```
private void butSetSwitchPos1_Click(object sender, EventArgs e)
        {
            //
            //  Set Switch Position to 1
            //

            try
            {
                ESLEthernetCommsLayer.SwitchPosition(1);

            }
            catch
            {
                //
                //  Display Error Message in
                //
            }
        }
```

The function name is "butSetSwitchPos1_Click" which is called after the button click action is complete. The "sender" object contains information about the button itself, and "e", a container for event specific data. The name "ESLEthernetCommsLayer" was assigned the name of the reference to the code blob (in this case a user control) which contains a public function to set the switch position. Naturally, the control handles the process of assembling the string representing the command and passing it to the function that consumes the string by dispatching it to the switch via ethernet Socket API calls seen in the code samples earlier.

1.4.1.4.2.3 Transition from Simple to Intelligent

Once having mastered the simple interface, the user has successfully performed the program logic to send and receive data from the switch. From this point, the GUI can be modified, and thus a stepping stone, to test program logic designed for intelligent control. For example, note the GUI has indicators to the right of each switch button, spaces to display the firmware version and compile date, and lastly, an indicator to show the user whether or not the front panel is locked out. These added indicators would be managed by the "intelligence" code designed to parse the string of characters received from the switch, interpret the data, and set the indicators in the appropriate state.

In this example, once the user logs on, the text of the log in/log out button would change to "Log out". Once a successful log in has occurred, the Firmware version request function could be dispatched and firmware version and compilation data of the 7810 firmware read, parsed, and displayed in the spaces available.

When the user clicks a button to command a switch position, the response would contain the switch position the firmware reports the current switch position and whether or not the switch is locked. In the GUI, the indicator to the right the column of buttons corresponding to the position reported by the firmware should be colored differently than the rest. If different than the requested switch position, then an error reported on the display. In addition to the state of the lockout indicator ("YES"/"NO") updated appropriately. Please note that switch position and lockout statuses can be reconciled against last known states to verify that your code is properly tracking this information.

Deliberately delay the time between commands to more than two minutes in order to test your logic that will respond to the session having ended and not require the user to manually log back in.

Once the logic has been successfully tested in a GUI (responses in the view window match status of indicators), you can incorporate it into the program logic for which it was originally intended and complete testing.

1.4.2 How the 7810 Handles invalid commands

The 7810 will respond with the response "501 INVALID COMMAND" the very instant it recognizes the command to be invalid. This means that for every subsequent character received the entire response will be sent again. If you write your controlling program to send the entire command at once without first checking to see what is being received, then it is possible many invalid command replies could be received before the unit finishes receiving your command.