TechNote

QuickBuilder String Handling Example

18

Control Technology Corporation, Hopkinton, MA • 800.282.5008 • www.ctc-control.com

This document shows you an example program that sends and receives a string over an Ethernet Raw Socket connection. This program can be easily modified for serial communications with a simple port set-up change. Simply omit the 22000 Raw Socket Commands (i.e. $REGISTERS[22000] = 7; through $REGISTERS[22007] = 1;) and set $REGISTERS[12000] to the desired comm. port.

For more specific information on general communications and using the message.ini file, refer to the 5300 Enhancements Overview Manual:
http://www.ctc-control.com/customer/techinfo/docs/5300_951/951-530001.pdf

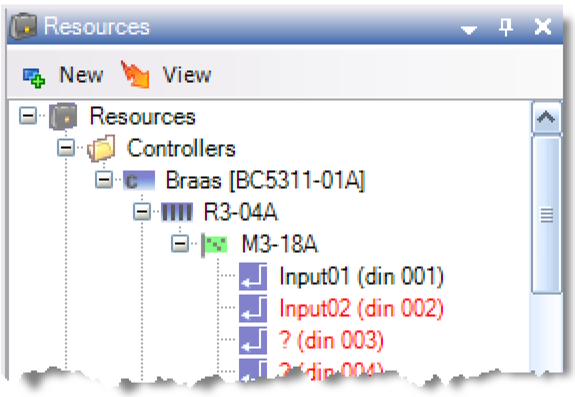
For more information on specific communication register settings, refer to the 5300 Quick Reference Register Guide:
http://www.ctc-control.com/customer/techinfo/docs/5300_951/951-530006.pdf

This program example uses one input to trigger the transmission of an ASCII string and then waits for a response. The string being sent is stored in a file called message.ini and copied to the Messages directory within the _system directory of the controller. This message.ini file can be created using notepad. Each line in the message.ini file represents a different string that you would like to send. Make sure that the last character of this file is a carriage return.

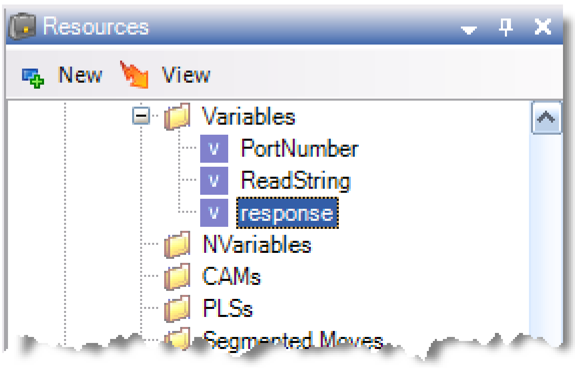
This program does the following:

1) Set-up the Ethernet port settings and initialized the connection.
2) Clears the buffer and waits for input 1.
3) Sends the 1st line of the message.ini file in ASCII format over Ethernet Raw Socket.
4) Gets the response back from the device that the message was sent to.

The system components (Resources) used in the program are shown below
NOTE: Both ReadString and response variables will need to be set as a string types:

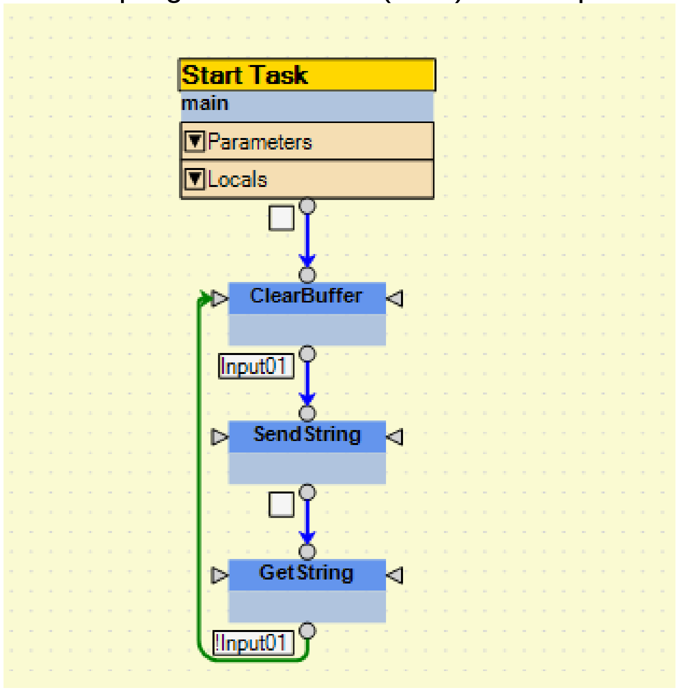


Resources
Controllers
Braas [BC5311-01A]
R3-04A
M3-18A
Input01 (din 001)
Input02 (din 002)
? (din 003)
? (din 004)

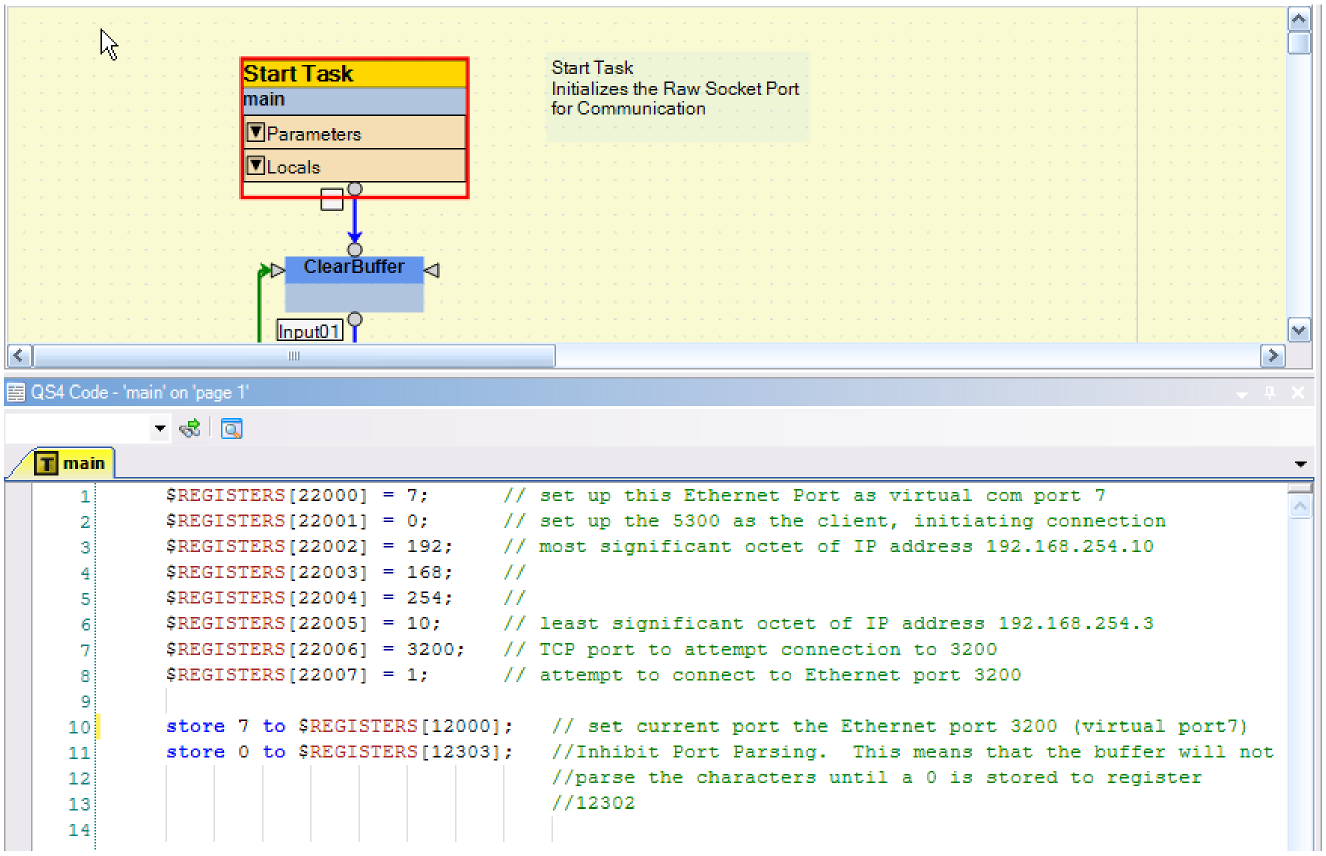


Variables
PortNumber
ReadString
response
NVariables
CAMs
PLSs
Segmented Moves

The overall program structure (SFC) is set-up as shown below:

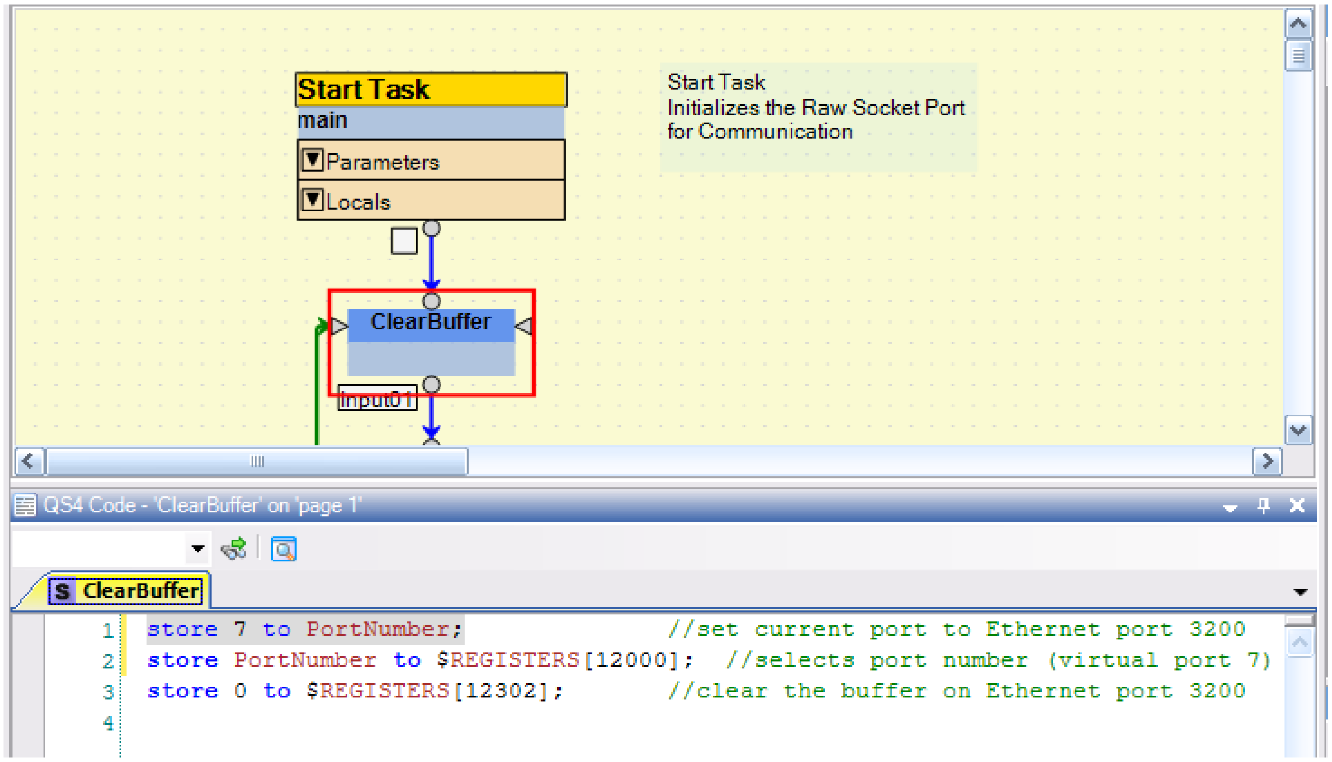


The initial Start Task sets up the Ethernet parameters as described in the code below.



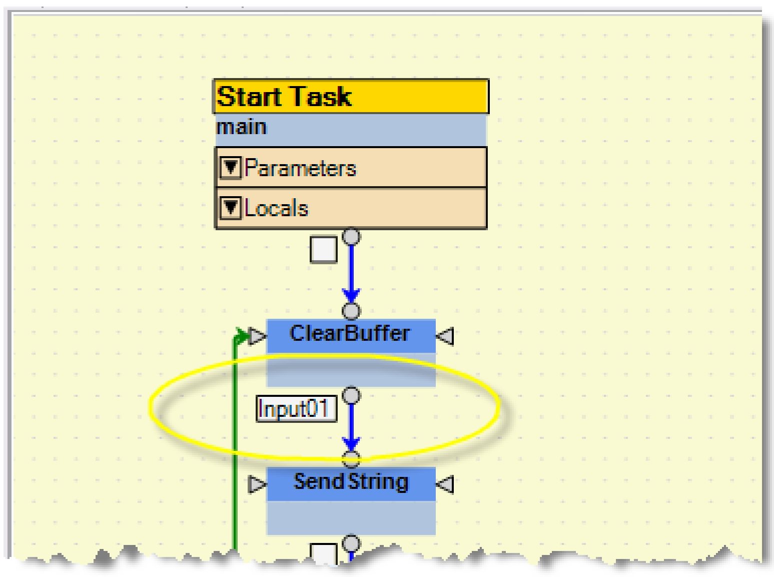
```
        $REGISTERS[22000] = 7;      // set up this Ethernet Port as virtual com port 7
        $REGISTERS[22001] = 0;      // set up the 5300 as the client, initiating connection
        $REGISTERS[22002] = 192;    // most significant octet of IP address 192.168.254.10
        $REGISTERS[22003] = 168;    //
        $REGISTERS[22004] = 254;    //
        $REGISTERS[22005] = 10;     // least significant octet of IP address 192.168.254.3
        $REGISTERS[22006] = 3200;   // TCP port to attempt connection to 3200
        $REGISTERS[22007] = 1;      // attempt to connect to Ethernet port 3200

        store 7 to $REGISTERS[12000];   // set current port the Ethernet port 3200 (virtual port7)
        store 0 to $REGISTERS[12303];   //Inhibit Port Parsing.  This means that the buffer will not
                                        //parse the characters until a 0 is stored to register
                                        //12302

```

The buffer is then cleared in the next step ClearBuffer as shown below:

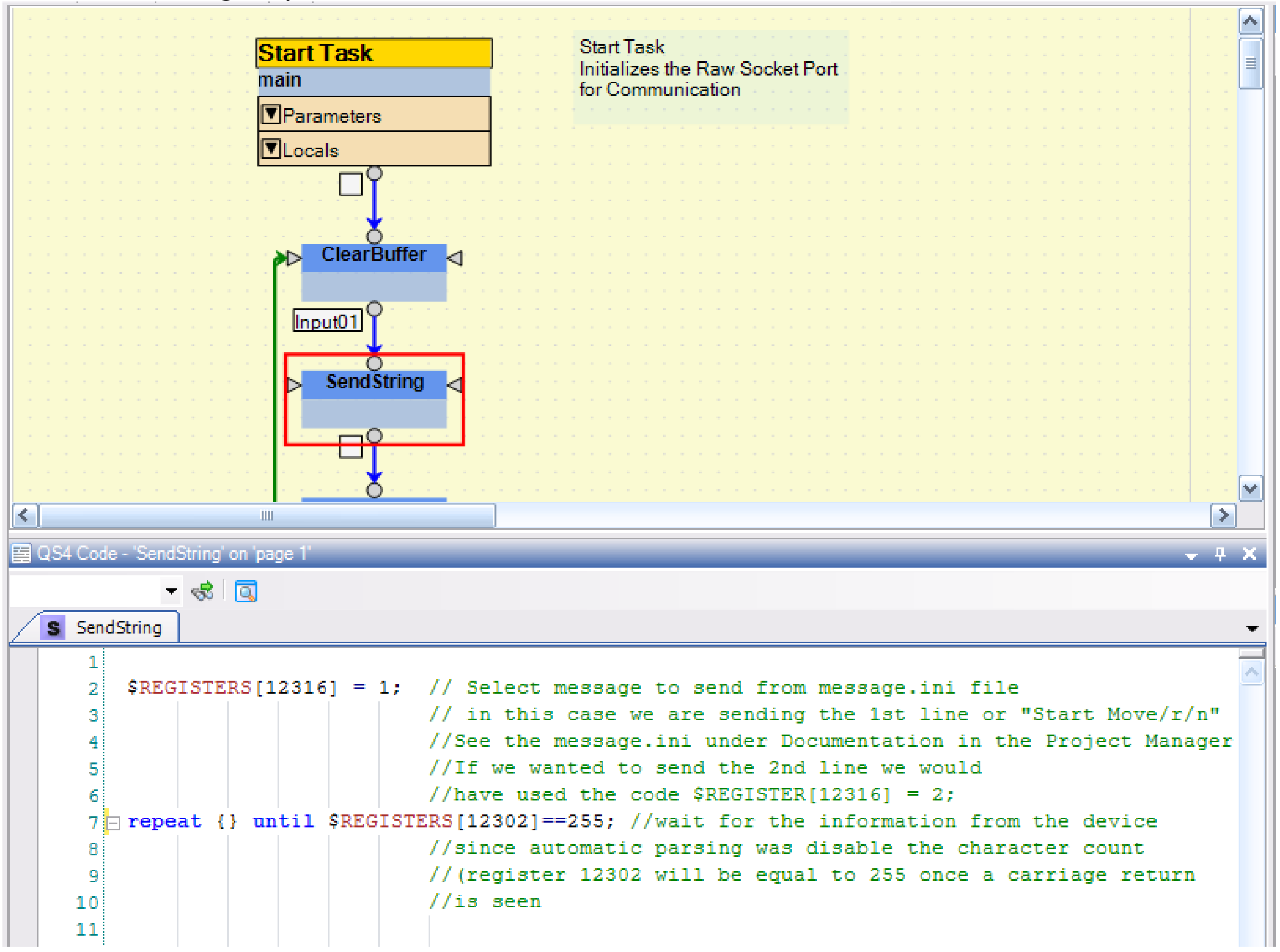


```
store 7 to PortNumber;                //set current port to Ethernet port 3200
store PortNumber to $REGISTERS[12000];  //selects port number (virtual port 7)
store 0 to $REGISTERS[12302];         //clear the buffer on Ethernet port 3200
```

Wait for Input 1 to be on.



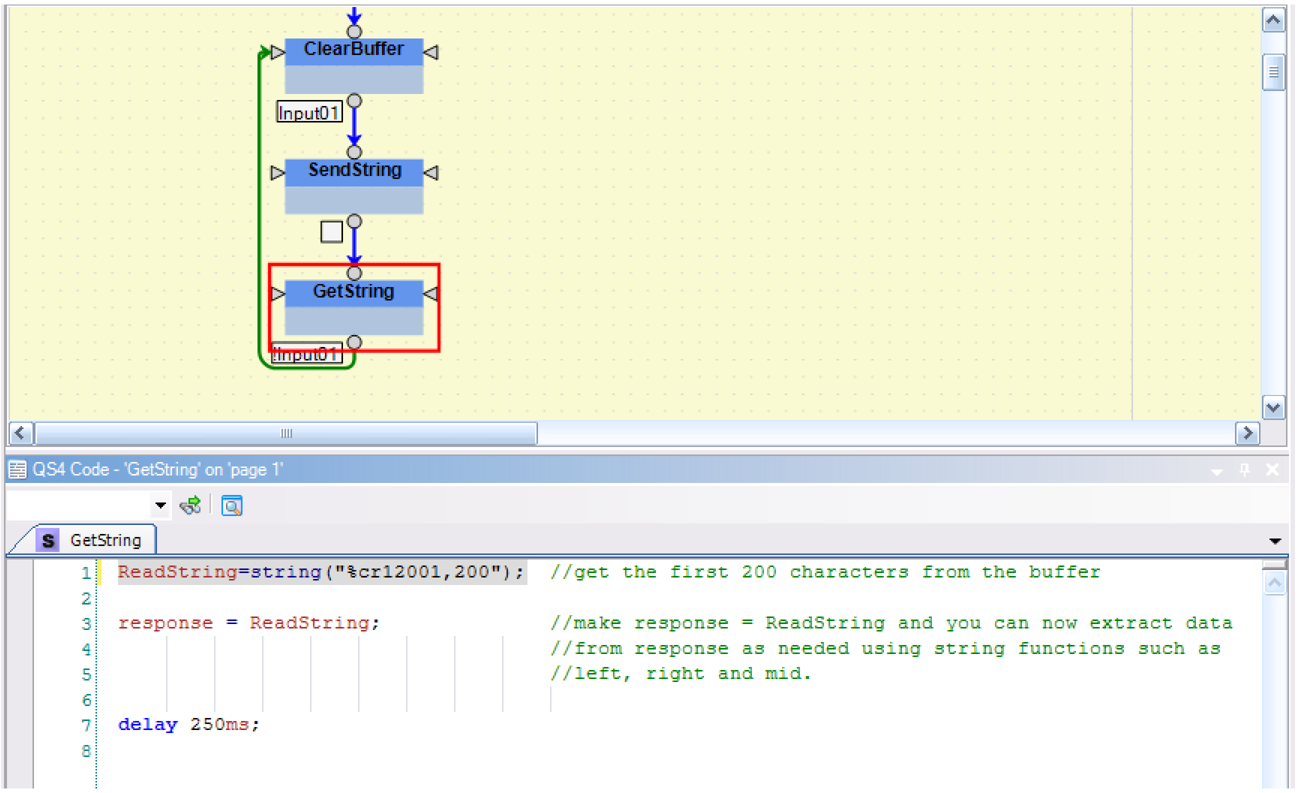
The 1st line of the message.ini is now sent out the Ethernet port as described in the SendString step code below:



Start Task
Initializes the Raw Socket Port
for Communication

```
$REGISTERS[12316] = 1;  // Select message to send from message.ini file
                        // in this case we are sending the 1st line or "Start Move/r/n"
                        //See the message.ini under Documentation in the Project Manager
                        //If we wanted to send the 2nd line we would
                        //have used the code $REGISTER[12316] = 2;
repeat {} until $REGISTERS[12302]==255; //wait for the information from the device
                        //since automatic parsing was disable the character count
                        //(register 12302 will be equal to 255 once a carriage return
                        //is seen

```

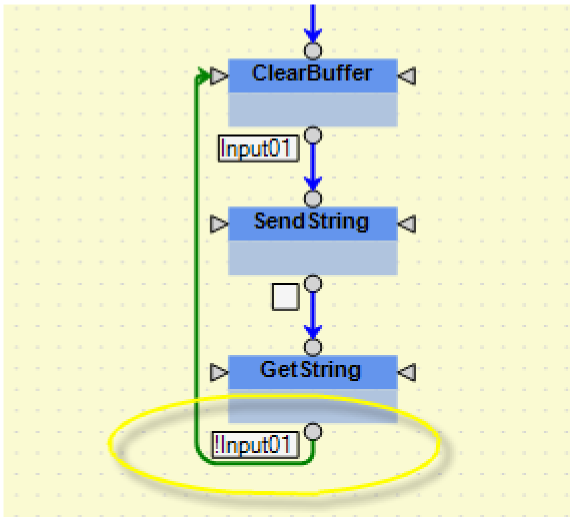
After we see the response from our device is complete we can retrieve the data as described in the GetString step code on the next page:



```
ReadString=string("%cr12001,200");   //get the first 200 characters from the buffer

response = ReadString;               //make response = ReadString and you can now extract data
                                     //from response as needed using string functions such as
                                     //left, right and mid.


delay 250ms;
```

Note: If you wanted to omit the first character received and just read the next 20 characters after that you could use the statement `ReadString=string("%cr12002,19");`, which would get the 2nd through 20th characters of the received string and assign those characters to `ReadString.`

We now wait for Input1 to turn off before we go back to the ClearBuffer step to repeat the whole process.



APPENDIX – MSB CODE

This appendix has the code shown in the steps above but will allow you to cut and past into your documents for your convenience.

StartTask: Main Code:

```
$REGISTERS[22000] = 7;        // set up this Ethernet Port as virtual
com port 7
$REGISTERS[22001] = 0;        // set up the 5300 as the client,
initiating connection
$REGISTERS[22002] = 192;      // most significant octet of IP address
192.168.254.10
$REGISTERS[22003] = 168;      //
$REGISTERS[22004] = 254;      //
$REGISTERS[22005] = 10;     // least significant octet of IP address
192.168.254.3
$REGISTERS[22006] = 3200;     // TCP port to attempt connection to 3200
$REGISTERS[22007] = 1;        // attempt to connect to Ethernet port
3200

store 7 to $REGISTERS[12000]; // set current port the Ethernet port
3200
store 0 to $REGISTERS[12303]; //Inhibit Port Parsing.  This means that
the buffer
                              // will not  parse the characters until a
0 is                                //stored to register 12302
```

Step: ClearBuffer Code:

```
store 7 to PortNumber;                    //set current port to
Ethernet port 3200
store PortNumber to $REGISTERS[12000];    //selects port number
store 0 to $REGISTERS[12302];             //clear the buffer on
Ethernet port 3200
```

Step: SendString Code:

```
$REGISTERS[12316] = 1;        // Select message to send from
message.ini file
                                    // in this case we are sending the
1st line.
                                    //If we wanted to send the 2nd line
we would
                                    //have used the code
$REGISTER[12316] = 2;
repeat {} until $REGISTERS[12302]==255;   //wait for the information
from the
                                    // device since automatic parsing
was disable                               // the character count
```

```
(register 12302 will                                // be equal to
255 once a carriage return
                                    //is seen
```

Step: GetString Code:

```
ReadString=string("%cr12001,200");  //get the first 200 characters from
the buffer

response = ReadString;              //make response = ReadString and
you can now
                                    // extract data from response as

needed using
                                    //string functions such as left,

right and                                 //mid.

delay 250ms;
```