CT Ethernet Serial Data Input

1 Introduction

This document describes receiving character data from an external Ethernet device to the ScanMaster Controller (SMC). The data can then be used as data input to a ScanMaster Designer (SMD) ScanScript program.

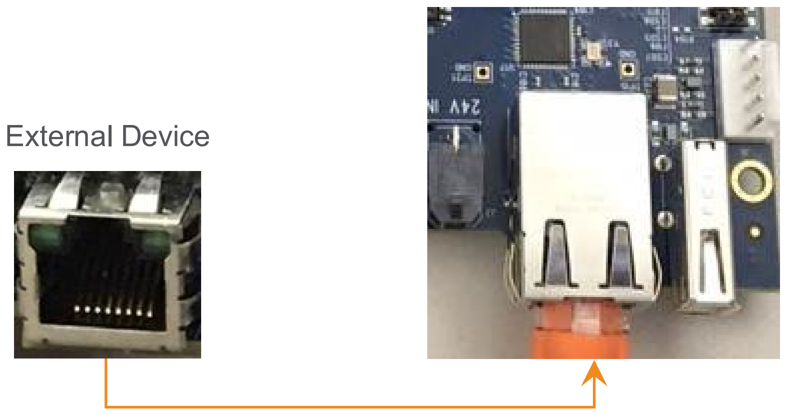


Figure 1 - **External device connecting to SMC**

In this document, we will use SMC to generate the test data and then use an Echo Program on the PC to echo the test data back to the SMC via Ethernet. With this method, we simulate the situation of an external Ethernet device sending serial data through the Ethernet to SMC. This lets you test the Ethernet serial data input to SMC without an external Ethernet device.

NOTE: For the purposes of this document, we will define our test data as 4 groups of data. Each group will contain an x, y, and z coordinate character string data with a line feed character after each group.

2 Prerequisites

1. Be sure your ScanMaster Controller (SMC) is operational. You can refer to section 5 of the SMC Quick Start Guide.

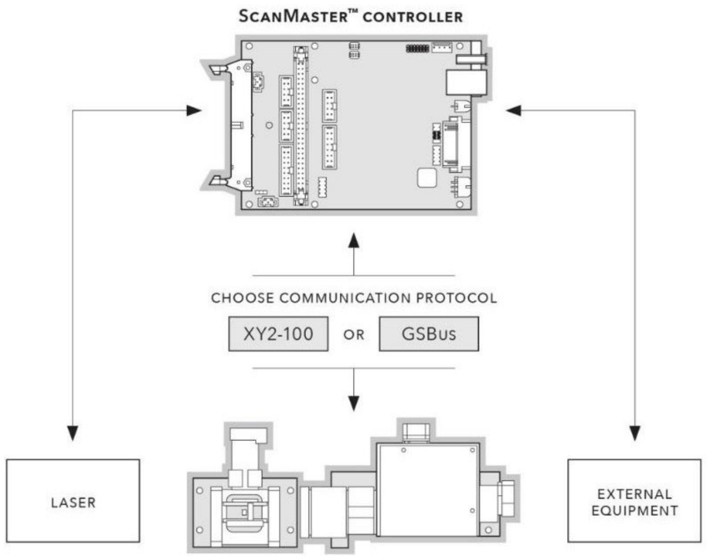


Figure 2 - **SMC Controller, laser and external equipment**

2. Be sure ScanMaster Designer (SMD) and the SMC SDK is installed. Refer to the Software Suite Setup Guide. CT Website Password: Contact CTI
3. Set the IP address of the external Ethernet device to 192.168.100.2 and port to 5033 so it talks to SMC properly. Refer to the documentation for your external device for details.
4. If you plan to use your PC instead of your external Ethernet device for this test, you will need to use the Ethernet Echo program (see below). Make sure Microsoft Visual Studio (Version 10 or greater) is installed (see the Microsoft website).

3 Ethernet Echo Program

The Ethernet Echo program is a console application that monitors the Ethernet port on your PC (port 5033).

It takes the place of having an actual external device connected to the SMC by echoing back data sent to it over the Ethernet connection. The source code is provided below for testing purposes. This is only used for testing your ScanScript code.

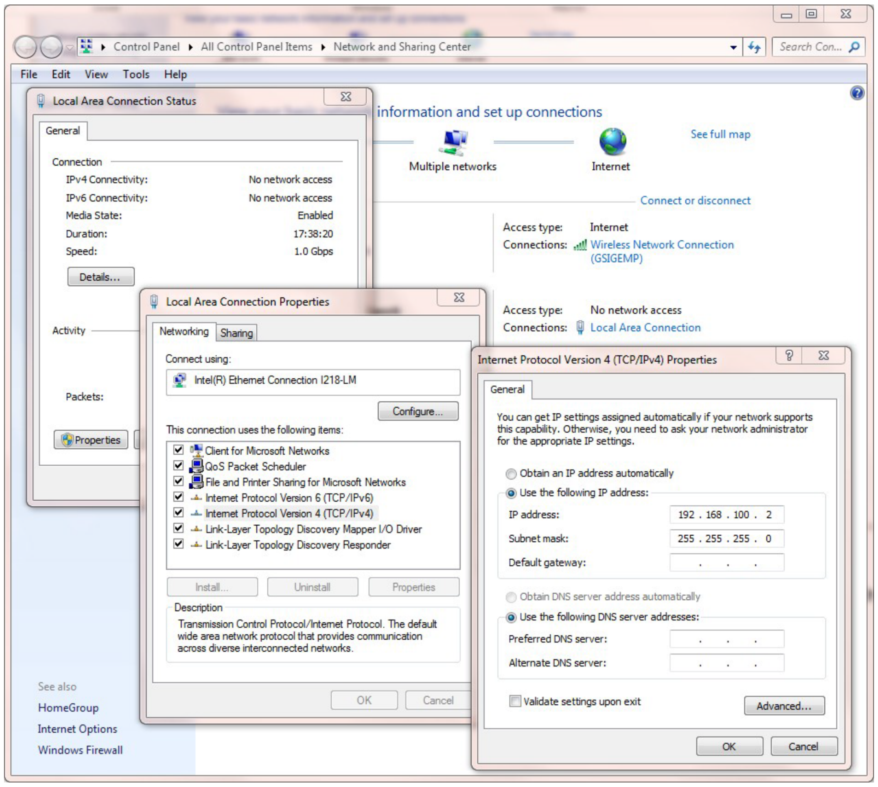
If you have an external device that can generate the serial data and send it to the SMC via Ethernet, the Ethernet Echo program is not needed. You can proceed to section 4.

Here is the procedure to use the Ethernet Echo program to test communication:

1. For the purposes of this document you should have your PC IP address set to a static IP of 192.168.100.2 or change the `targetAddress` variable in line 31 of the code example shown below to match your PC's IP address.

   Typical Ethernet Controller configuration process:

   a. Go to Control Panel\All Control Panel Items\Network and Sharing Center.
   b. Click to open Local Area Connection.
   c. Click on Properties on the Local Area Connection Status window.
   d. Select Internet Protocol Version 4 and then click on Properties.
   e. Enter the static IP Address 192.168.100.2.
   f. After you are done, close all the control panel windows.


Control Panel ▸ All Control Panel Items ▸ Network and Sharing Center
Search Con...
File Edit View Tools Help
information and set up connections
See full map
Multiple networks
Internet
Connect or disconnect
See also
HomeGroup
Internet Options
Windows Firewall

Access type: Internet
Connections: Wireless Network Connection (GSIGEMP)

Access type: No network access
Connections: Local Area Connection

Local Area Connection Status — General

Connection
IPv4 Connectivity: No network access
IPv6 Connectivity: No network access
Media State: Enabled
Duration: 17:38:20
Speed: 1.0 Gbps

Details...
Activity
Packets:
Properties

Local Area Connection Properties — Networking | Sharing

Connect using:
Intel(R) Ethernet Connection I218-LM
Configure...

This connection uses the following items:
☑ Client for Microsoft Networks
☑ QoS Packet Scheduler
☑ File and Printer Sharing for Microsoft Networks
☑ Internet Protocol Version 6 (TCP/IPv6)
☑ Internet Protocol Version 4 (TCP/IPv4)
☑ Link-Layer Topology Discovery Mapper I/O Driver
☑ Link-Layer Topology Discovery Responder

Install... Uninstall Properties

Description
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

OK Cancel

Internet Protocol Version 4 (TCP/IPv4) Properties — General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:
IP address: 192 . 168 . 100 . 2
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

○ Obtain DNS server address automatically
● Use the following DNS server addresses:
Preferred DNS server: . . .
Alternate DNS server: . . .

☐ Validate settings upon exit
Advanced...
OK Cancel

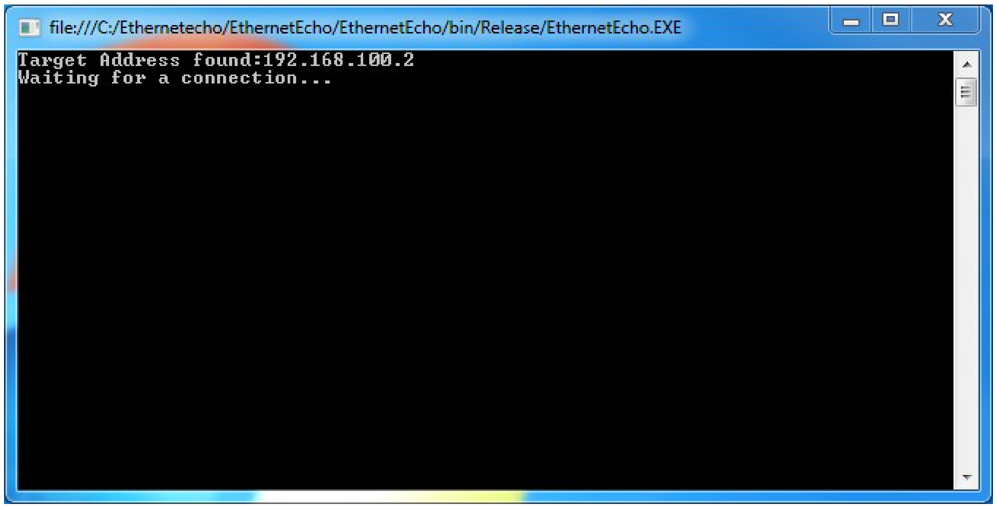
Figure 3 - **Typical local adapter IP address settings**

2. Directly connect your SMC (static IP address of 192.168.100.20) to your PC (static IP address of 192.168.100.2).
3. Compile the following Ethernet Echo C# Console application in Visual Studio.

Ethernet Echo source code:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;

// State object for reading client data asynchronously
public class StateObject
{
    // Client  socket.
    public Socket workSocket = null;
    // Size of receive buffer.
    public const int BufferSize = 1024;
    // Receive buffer.
    public byte[] buffer = new byte[BufferSize];
    // Received data string.
    public StringBuilder sb = new StringBuilder();
}

public class AsynchronousSocketListener
{
    // Thread signal.
    public static ManualResetEvent allDone = new ManualResetEvent(false);

    public AsynchronousSocketListener()
    {
    }

    public static void StartListening()
    {
        string targetAddress = "192.168.100.2";
        // Data buffer for incoming data.
        byte[] bytes = new Byte[1024];

        // Establish the local endpoint for the socket.
        // The DNS name of the computer
        // running the listener is "host.contoso.com".
        IPHostEntry ipHostInfo = Dns.Resolve(Dns.GetHostName());
        int addressListCount = 0;
        try
        {
            for (int i = 0; i < 5; i++) // 5 or the number of Ethernet Controllers in your PC
            {
                string currAddress = ipHostInfo.AddressList[i].ToString();
                if (string.Equals(targetAddress, currAddress) == true)
                {
                    addressListCount = i;
                    break;
                }
            }
            if (ipHostInfo.AddressList[addressListCount].ToString() == targetAddress)
            {
                Console.WriteLine("Target Address found:" + targetAddress);
            }
        }

        catch
        {
            Console.WriteLine("Controller with targetaddress not found");
            Console.WriteLine("\nPress ENTER to continue...");
            Console.Read();
            return;
        }
        IPAddress ipAddress = ipHostInfo.AddressList[addressListCount];
        IPEndPoint localEndPoint = new IPEndPoint(ipAddress, 5033);

        // Create a TCP/IP socket.
        Socket listener = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);

        // Bind the socket to the local endpoint and listen for incoming connections.
        try
        {
            listener.Bind(localEndPoint);
            listener.Listen(100);

            while (true)
            {
                // Set the event to nonsignaled state.
                allDone.Reset();

                // Start an asynchronous socket to listen for connections.
                Console.WriteLine("Waiting for a connection...");
                listener.BeginAccept(
                    new AsyncCallback(AcceptCallback),
                    listener);

                // Wait until a connection is made before continuing.
                allDone.WaitOne();
            }

        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }

        Console.WriteLine("\nPress ENTER to continue...");
        Console.Read();

    }

    public static void AcceptCallback(IAsyncResult ar)
    {
        // Signal the main thread to continue.
        allDone.Set();

        // Get the socket that handles the client request.
        Socket listener = (Socket)ar.AsyncState;
        Socket handler = listener.EndAccept(ar);

        // Create the state object.
        StateObject state = new StateObject();
        state.workSocket = handler;
        handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
            new AsyncCallback(ReadCallback), state);
    }

    public static void ReadCallback(IAsyncResult ar)
    {
        String content = String.Empty;

        // Retrieve the state object and the handler socket
        // from the asynchronous state object.
        StateObject state = (StateObject)ar.AsyncState;
        Socket handler = state.workSocket;

        // Read data from the client socket.
        int bytesRead = handler.EndReceive(ar);

        if (bytesRead > 0)
        {
            // There  might be more data, so store the data received so far.
            state.sb.Append(Encoding.ASCII.GetString(
                state.buffer, 0, bytesRead));

            // Check for end-of-file tag. If it is not there, read
            // more data.
            content = state.sb.ToString();
            //commented test data line below
            //content = "10,10,0\n";
            //Waits for a line feed ("\n") character before echoing the data back
            if (content.IndexOf("\n") > -1)
            {
                // All the data has been read from the
                // client. Display it on the console.
                Console.WriteLine("Read {0} bytes from socket. \n Data : {1}",
                    content.Length + 1, content + "\n");
                // Echo the data back to the client.
                Send(handler, content);
            }
            else
            {
                // Not all data received. Get more.
                handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
                new AsyncCallback(ReadCallback), state);
            }



        }
    }

    private static void Send(Socket handler, String data)
    {
        // Convert the string data to byte data using ASCII encoding.
        byte[] byteData = Encoding.ASCII.GetBytes(data);

        // Begin sending the data to the remote device.
        handler.BeginSend(byteData, 0, byteData.Length, 0,
            new AsyncCallback(SendCallback), handler);
    }

    private static void SendCallback(IAsyncResult ar)
    {
        try
        {
            // Retrieve the socket from the state object.
            Socket handler = (Socket)ar.AsyncState;

            // Complete sending the data to the remote device.
            int bytesSent = handler.EndSend(ar);
            Console.WriteLine("Sent {0} bytes to client.", bytesSent);

            handler.Shutdown(SocketShutdown.Both);
            handler.Close();

        }
        catch (Exception e)
        {
            Console.WriteLine(e.ToString());
        }
    }


    public static int Main(String[] args)
    {
        StartListening();
        return 0;
    }
}
```

When compiled in Visual Studio and then run, the Ethernet Echo program will look like the following:


file:///C:/Ethernetecho/EthernetEcho/EthernetEcho/bin/Release/EthernetEcho.EXE

```
Target Address found:192.168.100.2
Waiting for a connection...
```

Figure 4 - **Ethernet Echo program**

4 ScanScript Code

Follow the steps below to create an SMD project that will accept serial data from the SMC Ethernet connection to be used as data inside your other SMD projects as a starting point.

1. Open SMD.
2. Start a new project by selecting New from the menu.

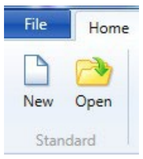


Figure 5 - **In SMD, select File then New from menu**

3. Select the Script1 tab.

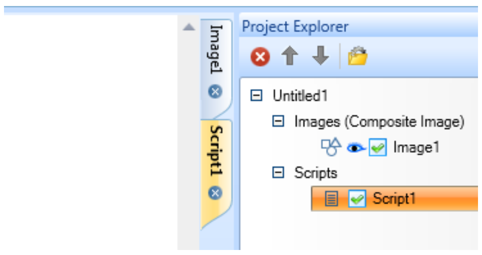
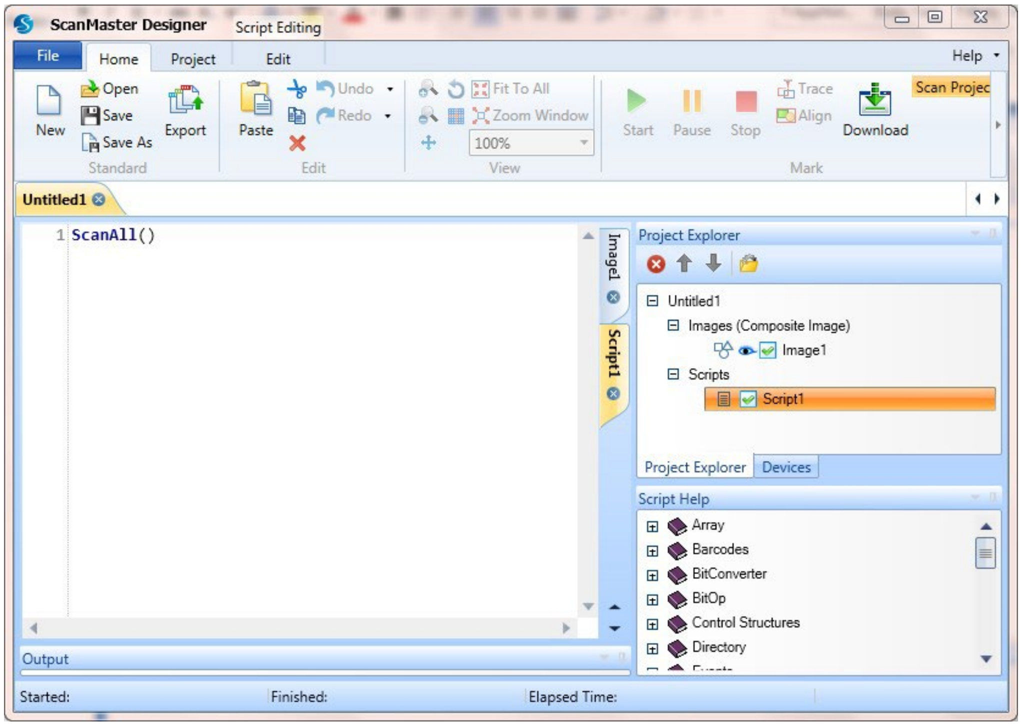


Figure 6 - **ScanScript (Script1) window tab in SMD**



```
ScanAll()
```

Figure 7 - **SMD Script1 window**

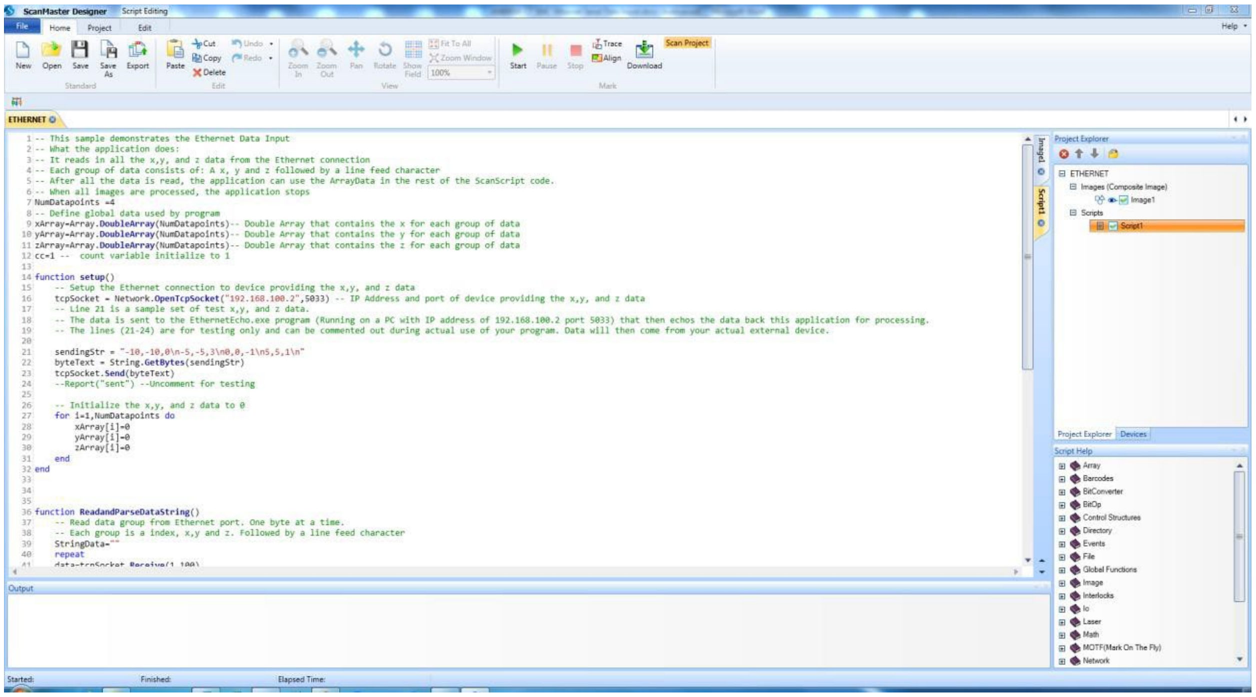
4. Replace the ScanAll() line of code with the following code and then run.

NOTE: The lines (20-24) are used to send data to be echoed back to simulate serial data input. This is for testing only and should be commented out during actual use of your program. Data will then come from your actual external device.

```
-- This sample demonstrates the Ethernet Data Input
-- What the application does:
-- It reads in all the x, y, and z data from the Ethernet connection
-- Each group of data consists of: A x, y and z followed by a line feed character
-- After all the data is read, the application can use the ArrayData in the rest of the ScanScript code.


-- Define global data used by program
xArray=Array.DoubleArray(NumDatapoints)-- Double Array that contains the x for each group of data
yArray=Array.DoubleArray(NumDatapoints)-- Double Array that contains the y for each group of data
zArray=Array.DoubleArray(NumDatapoints)-- Double Array that contains the z for each group of data
cc=1 --  count variable initialize to 1

function setup()
   -- Setup the Ethernet connection to device providing the x,y, and z data
   tcpSocket = Network.OpenTcpSocket("192.168.100.2",5033) -- IP Address and port of device
providing the x,y, and z data
   -- Line 21 is a sample set of test x,y, and z data.
```

```
    -- The data is sent to the EthernetEcho.exe program (Running on a PC with IP address of
192.168.100.2 port 5033) that then echos the data back this application for processing.
    -- The lines (21-24) are for testing only and can be commented out during actual use of your
program. Data will then come from your actual external device.

    sendingStr = "-10,-10,0\n-5,-5,3\n0,0,-1\n5,5,1\n"
    byteText = String.GetBytes(sendingStr)
    tcpSocket.Send(byteText)
    --Report("sent") --Uncomment for testing

    -- Initialize the x,y, and z data to 0
    for i=1,NumDatapoints do
        xArray[i]=0
        yArray[i]=0
        zArray[i]=0
    end
end



function ReadandParseDataString()
    -- Read data group from Ethernet port. One byte at a time.
    -- Each group is a index, x,y and z. Followed by a line feed character
    StringData=""
    repeat
    data=tcpSocket.Receive(1,100)
    if (data ~= nil) then
        StringData=String.Concat(StringData,data.getstring())
    end
    -- keep adding byte characters (data) to cellStringData until line feed
    until(String.EndsWith(data.getstring(),"\n"))
    -- Split StringData into it's three parts and assign each StringData item into it's corresponding array.
    StringData=String.Split(StringData,",")
    xArray[cc]=ToNumber(StringData[1])
    yArray[cc]=ToNumber(StringData[2])
    zArray[cc]=ToNumber(StringData[3])
    -- return StringData from the ReadDataString function
    cc=cc+1
        return(StringData)
end

--  This is the beginning of the  main part of program
setup()
-- Populate data arrays with data from Ethernet data strings
for i=1,NumDatapoints do
```

```
    rdata=ReadandParseDataString()
-- Report(rdata)  --  uncomment for testing purposes
-- Now the Ethernet data is populated in the xArray, yArray and zArray and is ready to be used as
data in your ScanScript program.
-- The following reconstructs the string from the data for testing
    xyz=""
    xyz=String.Concat(xyz,xArray[i])
    xyz=String.Concat(xyz,",")
    xyz=String.Concat(xyz,yArray[i])
    xyz=String.Concat(xyz,",")
    xyz=String.Concat(xyz,zArray[i])
    Report(xyz)
end
-- The rest of your app can start here using the data arrays
```



```
 1 -- This sample demonstrates the Ethernet Data Input
 2 -- What the application does:
 3 -- It reads in all the x,y, and z data from the Ethernet connection
 4 -- Each group of data consists of: A x, y and z followed by a line feed character
 5 -- After all the data is read, the application can use the ArrayData in the rest of the ScanScript code.
 6 -- When all images are processed, the application stops
 7 NumDatapoints =4
 8 -- Define global data used by program
 9 xArray=Array.DoubleArray(NumDatapoints)-- Double Array that contains the x for each group of data
10 yArray=Array.DoubleArray(NumDatapoints)-- Double Array that contains the y for each group of data
11 zArray=Array.DoubleArray(NumDatapoints)-- Double Array that contains the z for each group of data
12 cc=1 --  count variable initialize to 1
13
14 function setup()
15     -- Setup the Ethernet connection to device providing the x,y, and z data
16     tcpSocket = Network.OpenTcpSocket("192.168.100.2",5033) -- IP Address and port of device providing the x,y, and z data
17     -- Line 21 is a sample set of test x,y, and z data.
18     -- The data is sent to the EthernetEcho.exe program (Running on a PC with IP address of 192.168.100.2 port 5033) that then echos the data back this application for processing.
19     -- The lines (21-24) are for testing only and can be commented out during actual use of your program. Data will then come from your actual external device.
20
21     sendingStr = "-10,-10,0\n-5,-5,3\n0,0,-1\n5,5,1\n"
22     byteText = String.GetBytes(sendingStr)
23     tcpSocket.Send(byteText)
24     --Report("sent") --Uncomment for testing
25
26     -- Initialize the x,y, and z data to 0
27     for i=1,NumDatapoints do
28         xArray[i]=0
29         yArray[i]=0
30         zArray[i]=0
31     end
32 end
33
34
35
36 function ReadandParseDataString()
37     -- Read data group from Ethernet port. One byte at a time.
38     -- Each group is a index, x,y and z. Followed by a line feed character
39     StringData=""
40     repeat
41     data=tcpSocket.Receive(1,100)
```

Figure 8 - **ScanScript window with the above code installed**

5 ScanScript code Description

1. xArray, yArray and zArray are declared to store the test data (Script1 Lines 9-11).
2. The Setup function first opens the Ethernet port (Script 1 Line 16).
3. The Setup function then sends the test data from the SMC to the Ethernet Echo Program (will be commented out if you use the external Ethernet device to send data).
4. It then sets the Array data to an initial value of 0 for each data element (Script1 Lines 14-31).
5. The SMC then reads in the data sent to the Ethernet port (Script 1 Lines 36–55, ReadandParseDataString function).
6. Inside the for loop (Line 61) the ReadandParseDataString function is called to read each group of data and parse the corresponding x, y and z array data in to the x,y and z arrays.