User Manual **Version 2.0.0**

Smart Motherboard SDK

for MB-SMT-4, MB-SMT-8 and MB-SMT-D

MicroStrain® Micro Sensors, Big Ideas®

www.microstrain.com

459 Hurricane Lane, Suite 102
Williston, VT 05495
Phone 802-862-6629
Fax 802-863-4093
www.microstrain.com
support@microstrain.com

ISSUED: 10 February 2009

Table of Contents

Overview

The MicroStrain Smart Motherboard Software Development Kit (SDK) is a set of development tools that allows a user to create applications which communicate with MicroStrain's 1-4 channel MB-SMT-4, 5-8 channel MB-SMT-8 and 1-2 channel MB-SMT-D Smart Motherboards.

The SDK is targeted at providing the user with the tools required to build Windows applications for the RS-232 (wired) physical communication interfaces of the MicroStrain Smart Motherboards.

The SDK consists of the following major components:

- Smart Motherboard SDK manual
- Sample Code:
    - Visual Studio C++ 2003
    - LabVIEW 7.1.1
    - Visual Studio Visual Basic 2005
    - Visual Basic 6.0

It is expected that the user will have:

- a working knowledge of the MicroStrain Smart Motherboards;
- experience coding in the programming language in use;
- a working knowledge of communication ports.

Please review the Disclaimer and SDK Support on the following page.

Good luck and good coding.

Disclaimer

The SDK is provided “as is” and any expressed or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall MicroStrain or its contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this SDK, even if advised of the possibility of such damage. MicroStrain will make every effort to amplify the instructions contained in the SDK but will neither undertake to detail the functioning of the hardware or firmware in the Smart Motherboard family nor debug the user’s code.

SDK Support

MicroStrain will only accept requests for support for this SDK via email to support@microstrain.com. Please include the following information to expedite the support:

- Your name
- Your company or organization
- Model name, serial number, firmware version of the orientation sensor(s)
- Physical interface
- Coding language
- Hardware platform
- Operating System
- Your issue(s) in detail including screen shots, data files, links and any other attachments that will aid in understanding

.

Data Communications Protocol

The Smart Motherboard is capable of communicating with a host system using the RS-232 communications standard.

RS-232 Signals Definition

Signal	Name	Direction	Function
TxD	Transmit Data	Host to Smart Motherboard	Asynchronous Serial Data from Host
RxD	Receive Data	Smart Motherboard to Host	Asynchronous Serial Data to Host
GND	Signal Ground	N/A	Signal Ground Reference

RS-232 Asynchronous Character Format

Baud Rate	9600
Parity	None
Data Bits	8
Stop Bits	1

Polling Command

Each of the Smart Motherboard channels may be polled for a single data packet by issuing a one byte command. The table below shows the command byte and response packet for each channel.

Function	The Smart Motherboard will transmit the current reading on the requested channel
Command Byte	1 byte as follows: Channel 0 = 0x66 Channel 1 = 0x78 Channel 2 = 0x79 Channel 3 = 0x77 Channel 4 = 0x6D Channel 5 = 0x6E Channel 6 = 0x6F Channel 7 = 0x70
Response Packet	3 bytes as follows:
Byte 1	Header Byte = 0xFF
Byte 2	Most Significant Byte (MSB)
Byte 3	Least Significant Byte (LSB)

Scaling to Volts

The MSB and LSB are a digital representation of the current channel reading in volts. To scale to volts, use the following formula:

Volts = (((MSB * 256) + LSB) – 8192) * 0.0006103

Visual Studio C++ 2003 Code Samples

The C++ Code Samples were constructed with the Microsoft C++ Development Environment 2003 version 7.1.6030 with Microsoft .NET Framework 1.1 version 1.1.4322 SP1 using only objects and components native to the IDEs. No third party components or objects were used.

The Visual Studio C++ 2003 Code Samples are written directly from the Data Communications Protocol.

The code shown and explained below is reflective of the entire code sample offering.

The code sample spawns a Console window as shown in **Figure 1**. The Console window prompts for the comm port of the Smart Motherboard followed by a prompt for the Smart Motherboard channel to be polled. The data packet returned is parsed and scaled into volts.

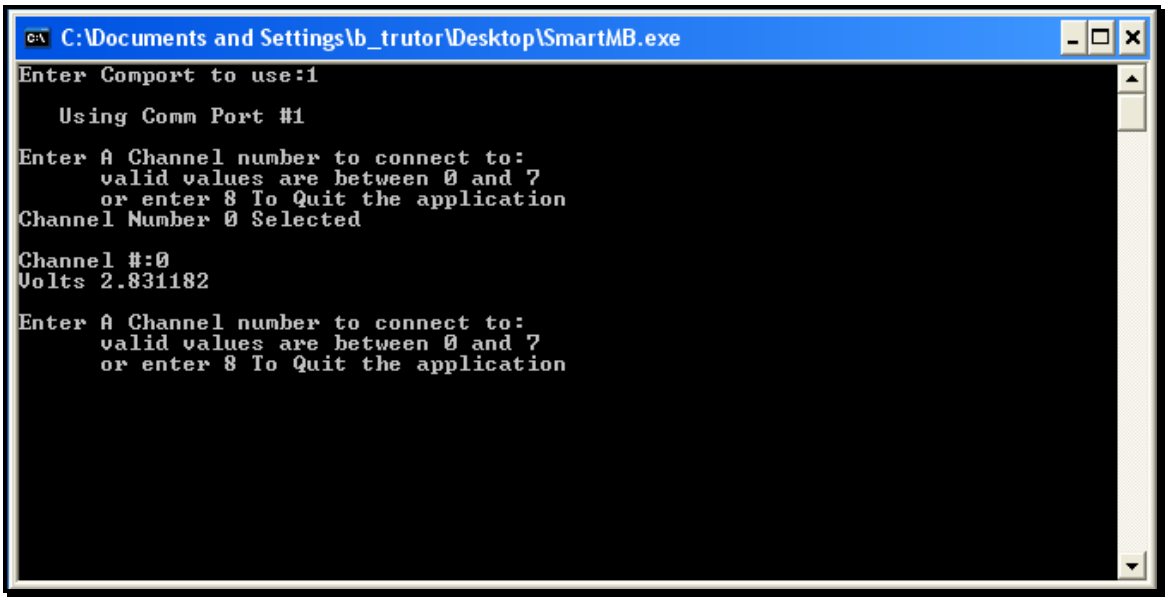

Figure 1

```
/*---------------------------------------------------------------------
 * Main for the Smart Motherboard console application.
 *---------------------------------------------------------------------
*/
#include <stdio.h>
#include <sys/timeb.h>
#include <time.h>
#include "i3dmgx2Errors.h"
#include "i3dmgx2_Cont.h"
#include "i3dmgx2Utils.h"
#include "i3dmgx2Utils_CM.h"
```

```
int GetComPort(); //prompt user for comport and opens it
void PollData(int portNum);
void Channel_connect(int portNum, char C_ID, int chOption);


int main(int argc, char* argv[])
{
      BOOL bQuit = FALSE;
      BOOL bPrintHeader = TRUE;
    int portNum = 0;
      int Ccount = 0;
      I3dmgx2Set pRecord;
      BYTE Ch_Choice;
      while (portNum <=0){
        portNum = GetComPort();
            if (Ccount++ > 4)
                  exit(1);
      }

      while(!bQuit){
            int chOption = 0;

            if(bPrintHeader)
            {
                  printf("\n");
                  printf("Enter A Channel number to connect to:\n");
                  printf("      valid values are between 0 and 7\n");
                  printf("      or enter 8 To Quit the application\n");

                  bPrintHeader = FALSE;
            }

            //read option from the keyboard
            while(!ReadCharNoReturn(&chOption))
            {
                  Sleep(50);
            }

        if(chOption > 55)
                    bQuit = TRUE;
            else{
                  switch (chOption){
                        case 48: chOption = 0;
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'f',
chOption);
                                 bPrintHeader = TRUE;
                                      break;
                        case 49: chOption = 1;
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'x',
chOption);
                                 bPrintHeader = TRUE;
                                      break;
                        case 50: chOption = 2;
```

```
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'y',
chOption);
                              bPrintHeader = TRUE;
                                    break;
                        case 51: chOption = 3;
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'w',
chOption);
                              bPrintHeader = TRUE;
                                    break;
                        case 52: chOption = 4;
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'm',
chOption);
                              bPrintHeader = TRUE;
                                    break;
                        case 53: chOption = 5;
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'n',
chOption);
                              bPrintHeader = TRUE;
                                    break;
                        case 54: chOption = 6;
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'o',
chOption);
                              bPrintHeader = TRUE;
                                    break;
                        case 55: chOption = 7;
                                    printf("Channel Number %d
Selected\n",chOption);
                                      Channel_connect(portNum, 'p',
chOption);
                              bPrintHeader = TRUE;
                                    break;
                        case 56: chOption = 0;
                                    printf("Channel Number
%d\n",chOption);
                                      bQuit = TRUE;
                                    break;
                        default: printf("Invalid option");
                                    bPrintHeader = TRUE;
                                      break;
                  }

            }

      }

      return 0;
}
```

```
//=====================================================================
======
// GetComPort
//---------------------------------------------------------------------
------
// Description: Prompt user for the comport and then opens it.  The
user is
//              prompted until a valid comport is selected and
successfully
//              opened.
//
// Return: HANDLE - handle to the opened comport
//=====================================================================
======
int GetComPort()
{
      int portNum   = 0;
      int iComPort  = 0;
      int errCount = 0;
      int MaxFail = 5;
      //Get comport number ask user for the comport number and open it
      while(portNum == 0)
      {
            printf("Enter Comport to use:");
            scanf("%d", &iComPort);

            /* open a port, map a device */
        portNum = i3dmgx2_openPort(iComPort, 9600, 8, 0, 1, 1024,
1024); //115200, 8, 0, 1, 1024, 1024);
        if (portNum<0) {
            printf("port open failed. ");
            printf("Comm error %d, %s:\n", portNum,
explainError(portNum));
                  if (errCount++ >= MaxFail)
                        exit(-1);
                  iComPort = 0;
            }
            else {
             printf("\n   Using Comm Port #%d \n", portNum);

            }

      }

      return portNum;
}

//=====================================================================
======
// Channel_connect(portNum, chOption)
//---------------------------------------------------------------------
------
// Description: Connects to a user specified channel on the smart
motherboard
//
// Return: voltage
```

```
//=====================================================================
======
void Channel_connect(int portNum, char C_ID, int chOption)
{
      int status = 0, i = 0;
      BYTE RBuff[3];
      BYTE code_SM;
      BYTE SM_MSB;
      BYTE SM_LSB;
      float SM_Volts;

      status = Get_Volts(portNum, C_ID, &RBuff[0]);
      if(status == 0)
      {
            if(RBuff[0] > 1){
               SM_Volts = (((RBuff[1]*256+RBuff[2])-8192)*0.0006103);
               printf("\n");
               printf("Channel #:%d\n", chOption);
               printf("Volts %f\n", SM_Volts);
               }
            else
                  printf("Channel not available\n");
      }
      else
      {
            printf("Get Volts() on Channel %d Failed\n", chOption);
            printf("Error Code: %d\n", status);
      }
                                    }
```

Code Sample 1

LabVIEW 7.1.1 Code Samples

The LabVIEW Code Samples were constructed with the National Instruments LabVIEW version 7.1.1 Interface Development Environment (IDE) using only objects and components native to the IDE. No third party components or objects were used.

The LabVIEW Code Samples are written directly from the Data Communications Protocol.

The code shown and explained below is reflective of the entire code sample offering.

The code sample contains a Front Panel as shown in **Figure 2**. The Front Panel allows selection of a Smart Motherboard channel. The Front Panel displays the data packet breakout (in this case, the 0x66 packet). The Front Panel has a red/green packet header indicator and a waveform chart which displays the data.

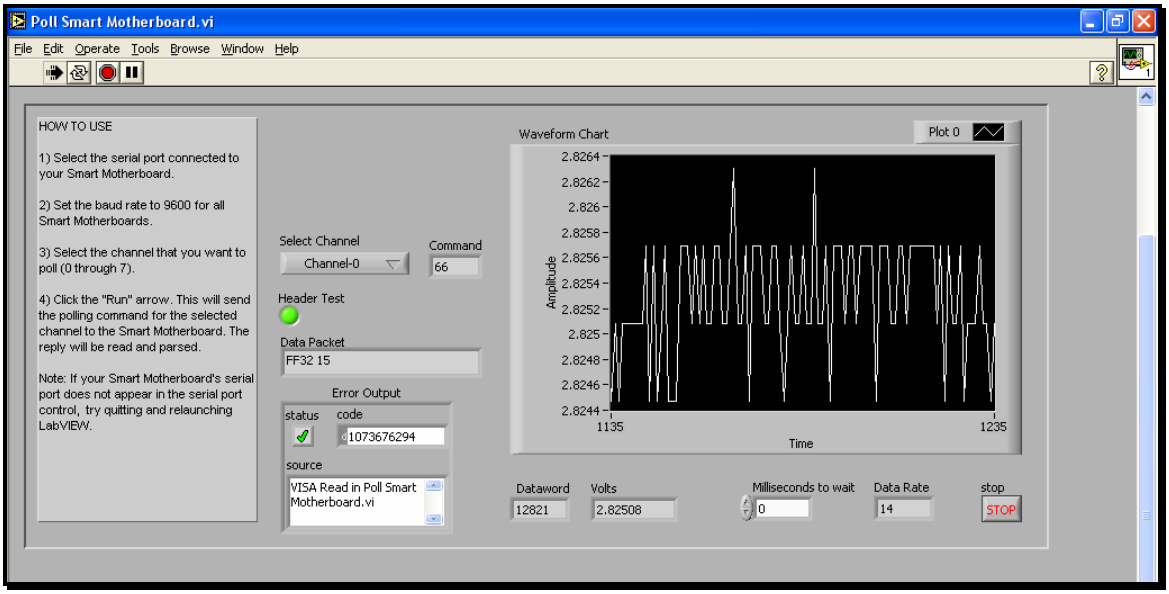

Figure 2

The Block Diagram is show in **Figure 3 and Figure 3A**. Baud rate and serial port are selected. The Run arrow is clicked. The serial port is configured and opened. The polling command is sent, the receive threshold is met, and the returned bytes are read from the serial port. The packet header is evaluated, the packet is parsed and scaled into volts, and the results are displayed on the Front Panel.

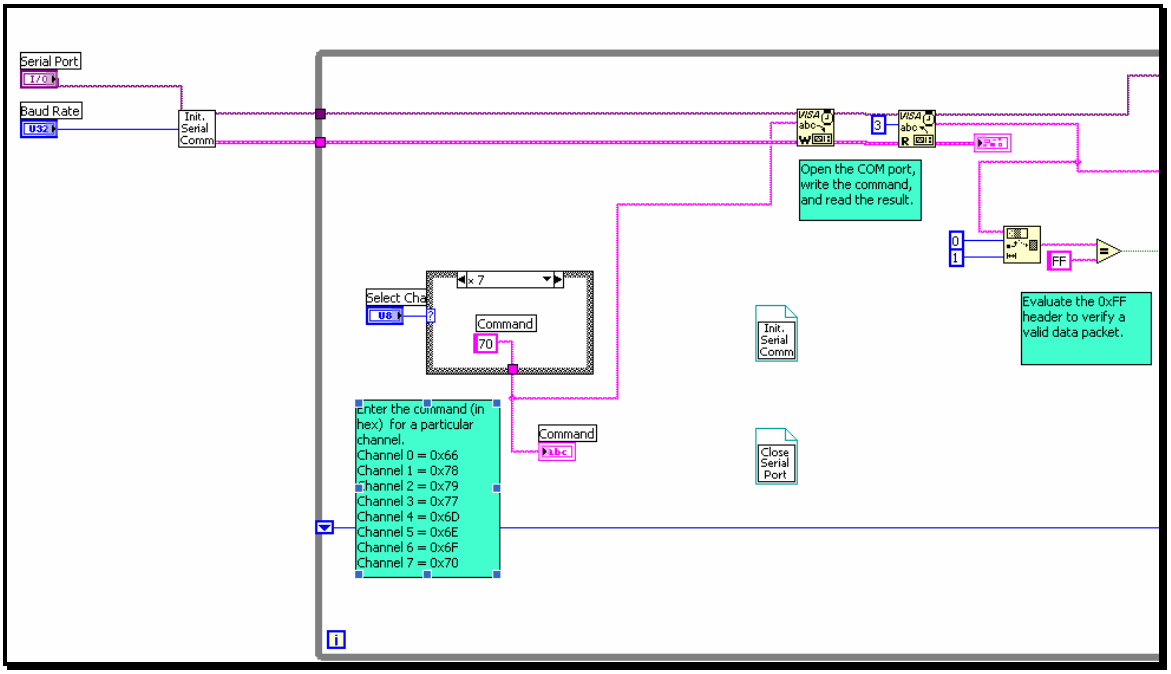

Figure 3

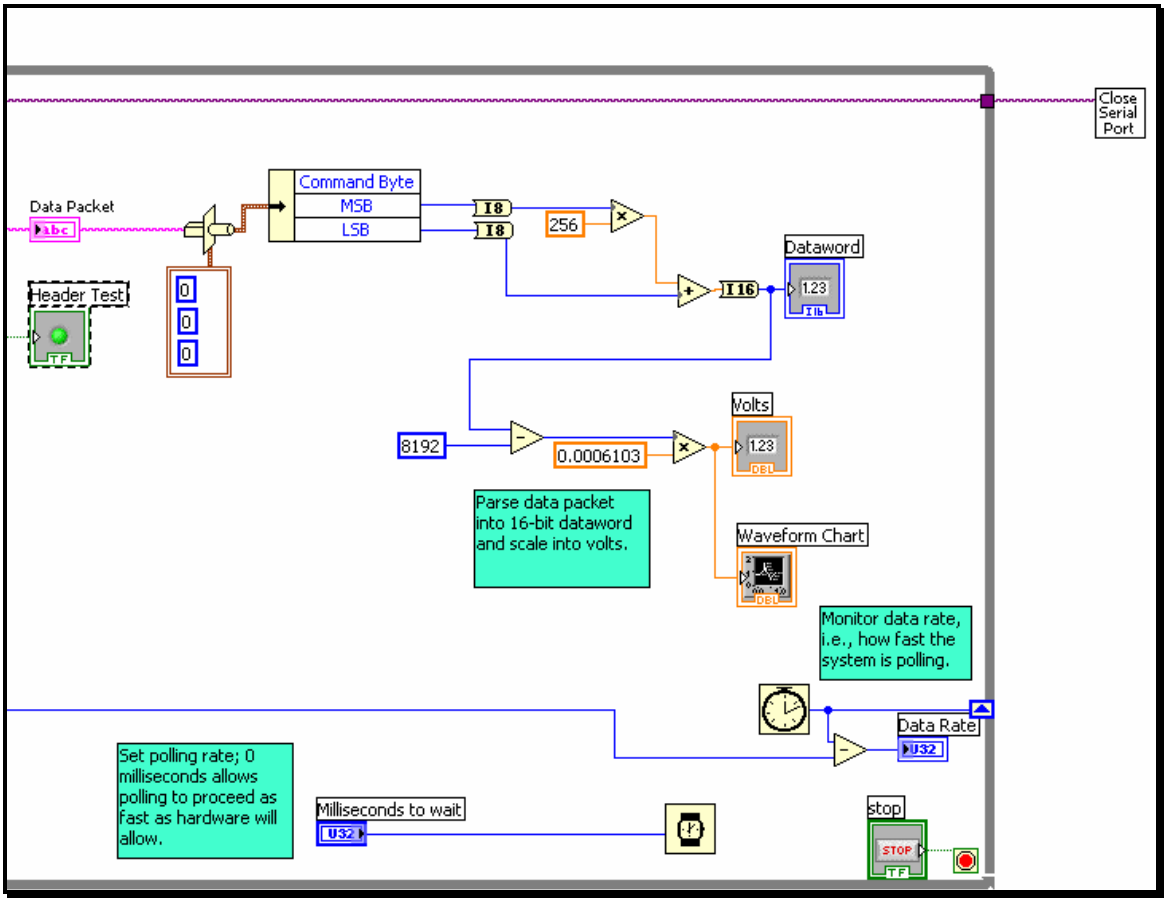

Figure 3A

Visual Studio Visual Basic 2005 Code Samples

The Visual Basic 2005 Code Samples were constructed with the Microsoft Visual Basic 2005 version 8.0.50727.42 Interface Development Environment (IDE) and Microsoft .NET Framework version 2.0.50727 SP1 using only objects and components native to the IDE. No third party components or objects were used.

The Visual Basic 2005 Code Samples are written directly from the Data Communications Protocol.

The code shown and explained below is reflective of the entire code sample offering.

The code sample contains a Form as shown in **Figure 4**. The Form allows selection of a communication port and a Smart Motherboard channel. The Form displays the data packet breakout (in this case, the 0x66 packet). The Form has a red/green packet indicator and an Event textbox which displays messages to the user.

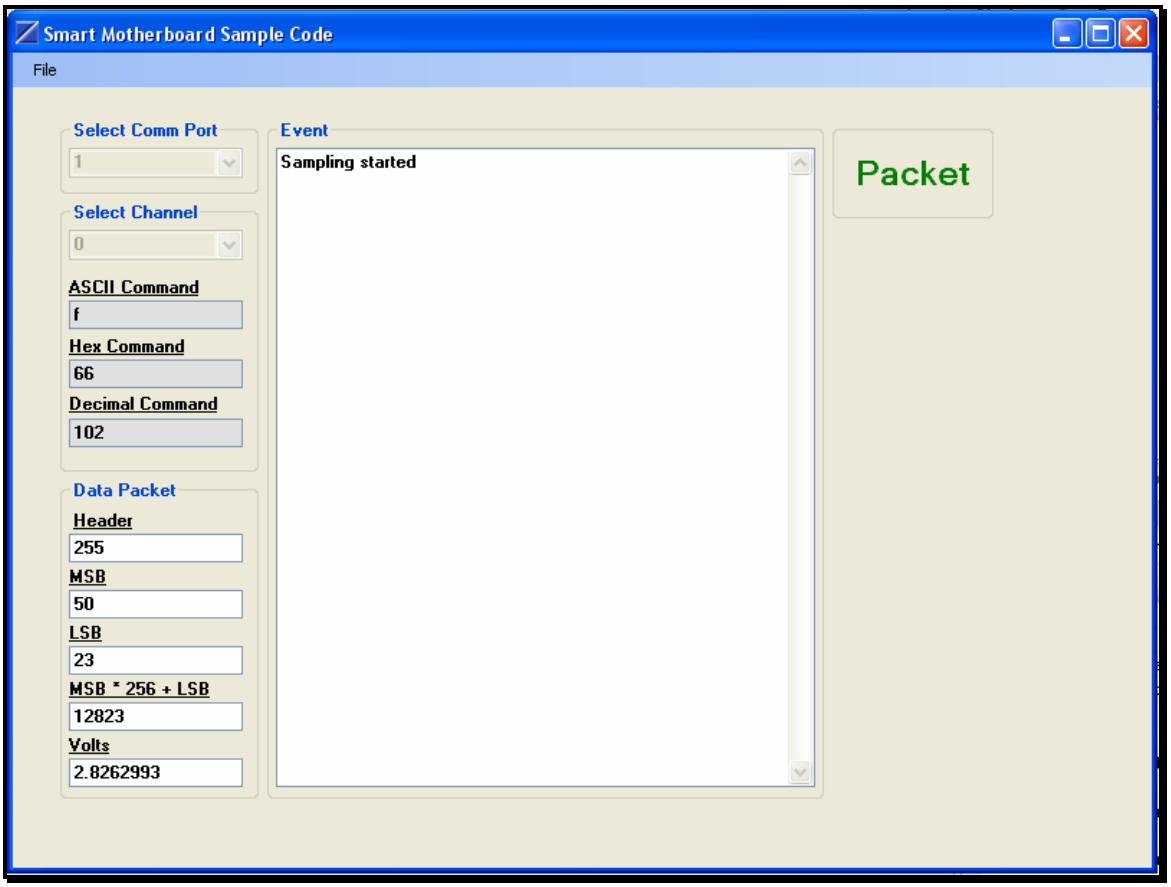

Figure 4

Pseudo code

- On **Main_Load**, the combo box for the comm port and channel selections are populated.

- The user clicks File and Sample on the menu bar.
- **SampleToolStripMenuItem_Click** configures the comm port and sends the first polling command.
- **SerialPort1_DataReceived** fires as the result of the receive threshold hitting 3 bytes and checks the data packet integrity. If the packet is bad, the packet is not parsed, the packet indicator is flashed red and a new polling command is sent. If the packet is good, the packet is parsed and displayed, the packet indicator is flashed green and the next polling command is sent.
- The cycle continues until the user intervenes by clicking File and Sample to stop the process.
- The **ReportEvent** function is used to populate user messages in the Event textbox.
- **ExitToolStripMenuItem_Click** terminates the application.

```
Public Class Main
```

```
    Private Sub Main_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'populate comm port combo box
        Dim X As Integer
        For X = 1 To 32
            cmbCommPort.Items.Add(Trim(Str(X)))
        Next
        'populate channel combo box
        Dim Y As Integer
        For Y = 0 To 7
            cmbChannel.Items.Add(Trim(Str(Y)))
        Next
        'Sets a value indicating whether to catch calls on the wrong
thread that access a control's Handle property
        txtHeader.CheckForIllegalCrossThreadCalls = False
        txtMSB.CheckForIllegalCrossThreadCalls = False
        txtLSB.CheckForIllegalCrossThreadCalls = False
        txtDataWord.CheckForIllegalCrossThreadCalls = False
        txtVolts.CheckForIllegalCrossThreadCalls = False
        txtASCII.CheckForIllegalCrossThreadCalls = False
        txtHex.CheckForIllegalCrossThreadCalls = False
        txtDecimal.CheckForIllegalCrossThreadCalls = False

    End Sub
```

```
    Private Sub SampleToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SampleToolStripMenuItem.Click
        'set error handler
        On Error GoTo ErrorHandler

        'test for comm port selection
        If cmbCommPort.Text = "" Then
            'event
            Call ReportEvent("Comm Port not selected")
            'exit
            Exit Sub
        End If
```

```
        'test for channel selection
        If cmbChannel.Text = "" Then
            'event
            Call ReportEvent("Channel not selected")
            'exit
            Exit Sub
        End If

        'determine action based on menu check
        If SampleToolStripMenuItem.Checked = False Then
            'event
            Call ReportEvent("Sampling started")
            'set menu check
            SampleToolStripMenuItem.Checked = True
            'disable combos
            cmbCommPort.Enabled = False
            cmbChannel.Enabled = False
            'set serial port number
            SerialPort1.PortName() = "COM" + Trim(cmbCommPort.Text)
            'set receive threshold
            SerialPort1.ReceivedBytesThreshold = 3
            'open serial port
            SerialPort1.Open()
            'populate byte array with polling command for channel
            ReDim arrBytes(0 To 0)
            arrBytes(0) = MyCommandByte
            'send command
            SerialPort1.Write(arrBytes, 0, 1)
        ElseIf SampleToolStripMenuItem.Checked = True Then
            'open serial port
            SerialPort1.Close()
            'event
            Call ReportEvent("Sampling stopped")
            'set menu check
            SampleToolStripMenuItem.Checked = False
            'set packet indicator
            lblPacket.ForeColor = Color.Red
            'enable combos
            cmbCommPort.Enabled = True
            cmbChannel.Enabled = True
        End If

        Exit Sub
ErrorHandler:
        'open serial port
        If SerialPort1.IsOpen = True Then SerialPort1.Close()
        'event
        Call ReportEvent("Sampling errored: " + Err.Description)
        'set menu check
        SampleToolStripMenuItem.Checked = False
        'set packet indicator
        lblPacket.ForeColor = Color.Red
        'enable combos
        cmbCommPort.Enabled = True
        cmbChannel.Enabled = True
        'exit
```

```
        Exit Sub
    End Sub
```

```
    Private Sub cmbChannel_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbChannel.SelectedIndexChanged
        'determine command
        If Val(cmbChannel.Text) = 0 Then
            MyCommandByte = 102
        ElseIf Val(cmbChannel.Text) = 1 Then
            MyCommandByte = 120
        ElseIf Val(cmbChannel.Text) = 2 Then
            MyCommandByte = 121
        ElseIf Val(cmbChannel.Text) = 3 Then
            MyCommandByte = 119
        ElseIf Val(cmbChannel.Text) = 4 Then
            MyCommandByte = 109
        ElseIf Val(cmbChannel.Text) = 5 Then
            MyCommandByte = 110
        ElseIf Val(cmbChannel.Text) = 6 Then
            MyCommandByte = 111
        ElseIf Val(cmbChannel.Text) = 7 Then
            MyCommandByte = 112
        End If
        'populate ascii, hex and decimal equivalent in textboxes
        txtASCII.Text = Chr(MyCommandByte)
        txtHex.Text = Hex(MyCommandByte)
        txtDecimal.Text = MyCommandByte
    End Sub
```

```
    Private Sub ExitToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ExitToolStripMenuItem.Click
        'quit app
        End
    End Sub
```

```
    Private Function ReportEvent(ByVal MyEvent As String) As Boolean
        'add event to text
        If txtEvent.Text = "" Then
            txtEvent.Text = MyEvent
        Else
            txtEvent.Text = txtEvent.Text + Chr(13) + Chr(10) + MyEvent
        End If
    End Function
```

```
    Private Sub SerialPort1_DataReceived(ByVal sender As System.Object,
ByVal e As System.IO.Ports.SerialDataReceivedEventArgs) Handles
SerialPort1.DataReceived
        'set error handler
        On Error GoTo ErrorHandler

        'set variable
        UpperLimitOfArray = SerialPort1.BytesToRead - 1
        'prep array
        ReDim arrBytes(0 To UpperLimitOfArray)
```

```
        'get comm buffer
        SerialPort1.Read(arrBytes, 0, SerialPort1.BytesToRead)

        'test for no byte return
        If UBound(arrBytes) = -1 Then
            'set packet indicator
            lblPacket.ForeColor = Color.Red
            'populate byte array with polling command for channel
            ReDim arrBytes(0 To 0)
            arrBytes(0) = MyCommandByte
            'send command
            SerialPort1.Write(arrBytes, 0, 1)
            'exit
            Exit Sub
        End If
        'test 3 byte return and header
        If Not UBound(arrBytes) = 2 And Not arrBytes(0) = 255 Then
            'set packet indicator
            lblPacket.ForeColor = Color.Red
            'populate byte array with polling command for channel
            ReDim arrBytes(0 To 0)
            arrBytes(0) = MyCommandByte
            'send command
            SerialPort1.Write(arrBytes, 0, 1)
            'exit
            Exit Sub
        End If
        'set packet indicator
        lblPacket.ForeColor = Color.Green
        'display data
        txtHeader.Text = arrBytes(0)
        txtMSB.Text = arrBytes(1)
        txtLSB.Text = arrBytes(2)
        MyDataWord = (CDbl(arrBytes(1)) * 256) + arrBytes(2)
        txtDataWord.Text = MyDataWord
        MyVolts = (MyDataWord - 8192) * 0.0006103
        txtVolts.Text = MyVolts
        'populate byte array with polling command for channel
        ReDim arrBytes(0 To 0)
        arrBytes(0) = MyCommandByte
        'send command
        SerialPort1.Write(arrBytes, 0, 1)
        Exit Sub
ErrorHandler:
        'close serial port
        If SerialPort1.IsOpen = True Then SerialPort1.Close()
        'exit
        Exit Sub
    End Sub
End Class
```

Code Sample 2

Visual Basic 6.0 Code Samples

The Visual Basic 6.0 Code Samples were constructed with the Microsoft Visual Basic 6.0 Interface Development Environment (IDE), Service Pack 5, using only objects and components native to the IDE. No third party components or objects were used.

The Visual Basic 6.0 Code Samples are written directly from the Data Communications Protocol.

The code shown and explained below is reflective of the entire code sample offering.

The code sample contains a Form as shown in **Figure 5**. The Form allows selection of a communication port and a Smart Motherboard channel. The Form displays the data packet breakout (in this case, the 0x66 packet). The Form has a red/green packet indicator and an Event textbox which displays messages to the user.

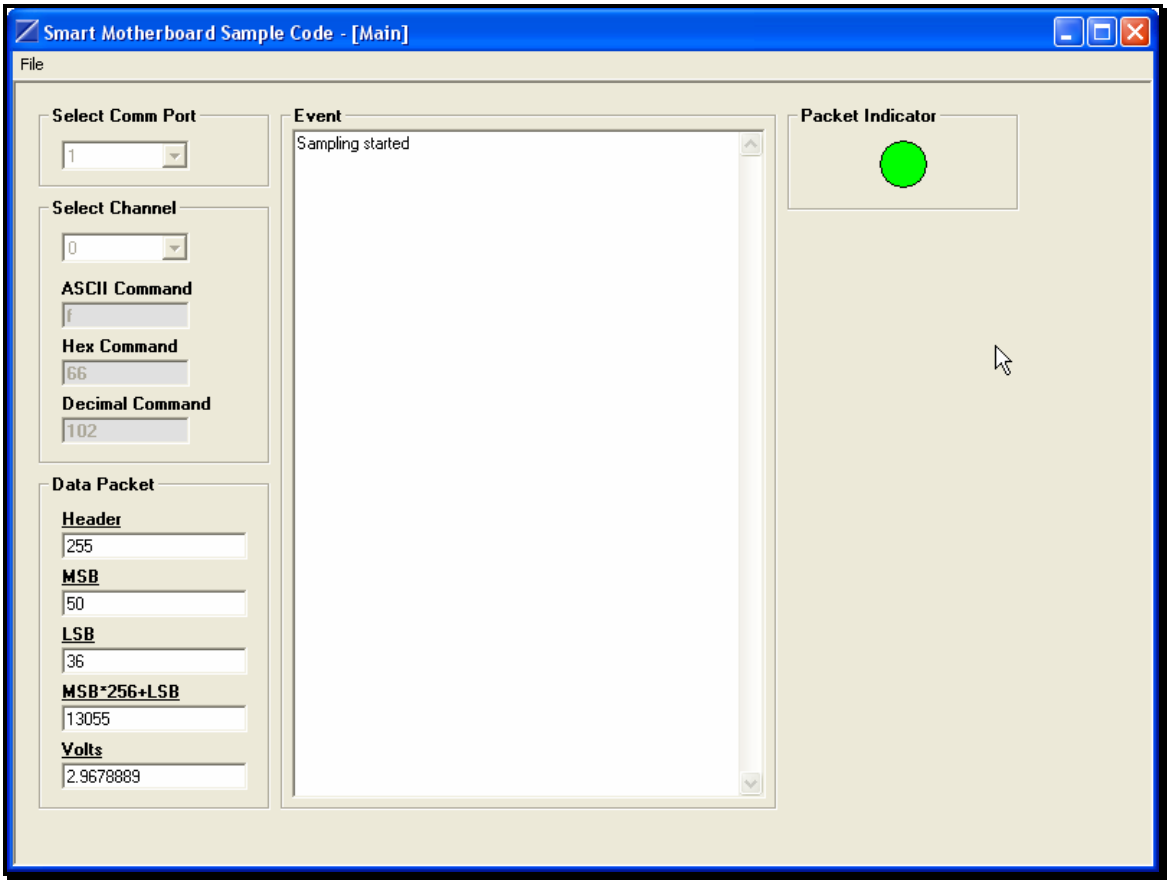

Figure 5

Pseudo code

- On **Form_Load**, the combo boxes for the comm port and channel selections are populated.

- The user selects a comm port and channel.
- The user clicks File and Poll on the menu bar.
- **mnuPoll_Click** configures the comm port and sends the first polling command.
- **MSComm1_OnComm** fires as a result of the receive threshold hitting 3 bytes and checks the data packet integrity. If the packet is bad, the packet is not parsed, the packet indicator is flashed red and a new polling command is sent. If the packet is good, the packet is parsed and displayed, the packet indicator is flashed green and the next polling command is sent.
- The cycle continues until the user intervenes by clicking File and Poll to stop the process.
- The **ReportEvent** function is used to populate user messages in the Event textbox.
- **mnuExit_Click** terminates the application.

```
Private Sub cmbChannel_Click()
'determine command
If Val(cmbChannel.Text) = 0 Then
   MyCommandByte = 102
ElseIf Val(cmbChannel.Text) = 1 Then
   MyCommandByte = 120
ElseIf Val(cmbChannel.Text) = 2 Then
   MyCommandByte = 121
ElseIf Val(cmbChannel.Text) = 3 Then
   MyCommandByte = 119
ElseIf Val(cmbChannel.Text) = 4 Then
   MyCommandByte = 109
ElseIf Val(cmbChannel.Text) = 5 Then
   MyCommandByte = 110
ElseIf Val(cmbChannel.Text) = 6 Then
   MyCommandByte = 111
ElseIf Val(cmbChannel.Text) = 7 Then
   MyCommandByte = 112
End If
'populate ascii, hex and decimal equivalent in textboxes
txtASCII.Text = Chr(MyCommandByte)
txtHex.Text = Hex(MyCommandByte)
txtDecimal.Text = MyCommandByte
End Sub
```

```
Private Sub Form_Load()
'populate comm port combo box
Dim X As Integer
For X = 1 To 16
   cmbCommPort.AddItem X
Next X
'populate channel combo box
```

```
Dim Y As Integer
For Y = 0 To 7
   cmbChannel.AddItem Y
Next Y
End Sub
```

```
Private Sub mnuExit_Click()
'quit app
End
End Sub
```

```
Private Sub mnuPoll_Click()
'set error handler
On Error GoTo ErrorHandler

'test for comm port selection
If cmbCommPort.Text = "" Then
   'event
   Call ReportEvent("Comm Port not selected")
   'exit
   Exit Sub
End If

'test for channel selection
If cmbChannel.Text = "" Then
   'event
   Call ReportEvent("Channel not selected")
   'exit
   Exit Sub
End If

'determine action based on menu check
If mnuPoll.Checked = False Then
   'event
   Call ReportEvent("Sampling started")
   'set packet indicator
   shpLED.FillColor = vbRed
   'set menu check
   mnuPoll.Checked = True
   'disable combos
   cmbCommPort.Enabled = False
   cmbChannel.Enabled = False
   'set comm port
   MSComm1.CommPort = Val(cmbCommPort.Text)
   'set receive threshold to 3 bytes
   MSComm1.RThreshold = 3
```

```
    'open comm port
    MSComm1.PortOpen = True
    'populate byte array with polling command for selected channel
    arrBytes = ChrB(MyCommandByte)
    'send polling command
    MSComm1.Output = arrBytes
ElseIf mnuPoll.Checked = True Then
    'event
    Call ReportEvent("Sampling stopped")
    'set menu check
    mnuPoll.Checked = False
    'close comm port
    MSComm1.PortOpen = False
    'enable combos
    cmbCommPort.Enabled = True
    cmbChannel.Enabled = True
    'set packet indicator
    shpLED.FillColor = vbRed
End If

Exit Sub
ErrorHandler:
    'set menu check
    mnuPoll.Checked = False
    'close comm port
    If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
    'enable combo
    cmbCommPort.Enabled = True
    cmbChannel.Enabled = True
    'set packet indicator
    shpLED.FillColor = vbRed
    'event
    Call ReportEvent("Sampling errored: " + Err.Description)
    'exit
    Exit Sub
End Sub
```

```
Private Function ReportEvent(MyEvent As String)
'add event to text
If txtEvent.Text = "" Then
    txtEvent.Text = MyEvent
Else
    txtEvent.Text = txtEvent.Text + Chr(13) + Chr(10) + MyEvent
End If
End Function
```

```
Private Sub MSComm1_OnComm()

'get all bytes in comm buffer
arrBytes = MSComm1.Input

'test 3 byte return and header
If Not UBound(arrBytes) = 2 Or Not arrBytes(0) = 255 Then
   'set packet indicator
   shpLED.FillColor = vbRed
   'populate byte array with polling command for selected channel
   arrBytes = ChrB(MyCommandByte)
   'send polling command
   MSComm1.Output = arrBytes
   'exit sub
   Exit Sub
End If

'set packet indicator
shpLED.FillColor = vbGreen

'display data
txtHeader.Text = arrBytes(0)
txtMSB.Text = arrBytes(1)
txtLSB.Text = arrBytes(2)
MyDataWord = (CDbl(arrBytes(1)) * 256) + arrBytes(0)
txtDataWord.Text = MyDataWord
MyVolts = (MyDataWord - 8192) * 0.0006103
txtVolts.Text = MyVolts

'populate byte array with polling command for selected channel
arrBytes = ChrB(MyCommandByte)
'send polling command
MSComm1.Output = arrBytes

End Sub
```

Code Sample 3

Suggested Debugging Tools

We have found that software port *sniffers* are incredibly valuable when building applications with serial communication. Here are some of our favorites; we highly recommend their use.

LookRS232

http://www.lookrs232.com/

- Look RS232 is a tool for debugging computer connection with peripheral devices using COM port, such as modem, mini-ATS, projector etc. Its easy to use interface guarantees an easy work with Com-port.
- Look RS232 can send data through COM port, receive data from an outer device through COM port, it has port indication as well.
- Look RS232 supports connection at the standard speed of 110…115200 kbit/s, it supports commands of COM port program control.
- Look RS232 supports data input in ASCII, BIN, HEX, OCT formats, keeps log of the sent and received data and commands in ASCII, BIN, HEX, OCT formats, it has an option of time link (relating to COM port activation).
- Look RS232 works with system initiated COM ports regardless of whether they are installed on motherboard or on supplementary in/output boards, e.g. VS-Com (Roadrunner) PCI-800H 8-port PCI.

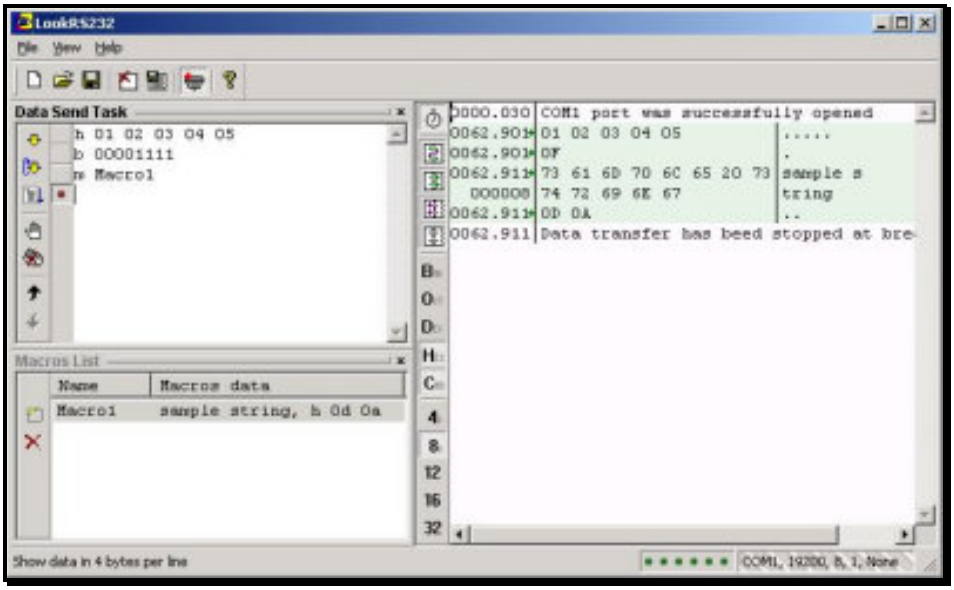

Figure 6

Serial Port Monitor

http://www.serial-port-monitor.com/index.html

Free Serial Port Monitor allows you to intercept, display and analyze all data exchanged between the Windows application and the serial device. It can be successfully used in application development, device driver or serial hardware development and offers the powerful platform for effective coding, testing and optimization.

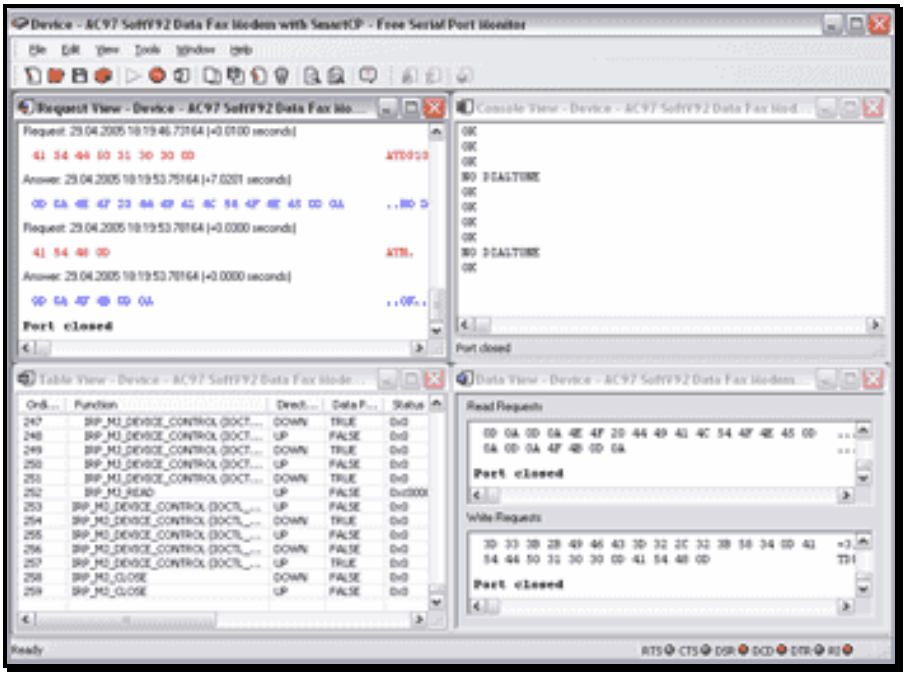

Figure 7

Comm Operator

http://www.serialporttool.com/CommOpV2Info.htm

- Comm Operator is a powerful tool for serial port communication. It can act as peripheral device emulator related RS232 COM port. It can also act as a debug tool for device's test and development.
- The built-in event driven auto send mechanics makes Comm Operator a full functioned device simulator with just a few mouse clicks. By this way, RS232 related software can be developed offsite, no need for device, only the rule script is enough.
- Comm Operator can send data through COM port, receive data from an outer device through COM port. The speed it supports range from 110 to 921600. It can detect all COM ports automatically regardless of whether they are real COM ports on motherboard or on supplementary in/output boards, or virtual COM ports created by special drivers.
- Comm Operator supports data input in ASCII, HEX as well as Decimal formats. It keeps log of the sent and received data in ASCII, HEX as well as Decimal formats. It also supports SYMBOL with which the invisible ASCII code became meaningful immediately. All setting can be saved to file and loaded later.

Support

Overview

- MicroStrain is committed to providing timely, knowledgeable, world-class support to its customers.
- We are open 24 X 7 through our web portal.
- We make every attempt to respond to your email the same business day.
- We are always available by telephone during business hours.
- We provide in-depth FAQs, manuals, quick start guides and technical notes.
- Firmware and software upgrades are made available on-line as they become available.
- Code samples in several languages are posted to aid your development.
- We support our customers as we would want to be supported.

Web

Our home page is at URL: www.microstrain.com
Our support page is at URL: http://www.microstrain.com/support_overview.aspx

Email

MicroStrain's Support Engineers make every attempt to respond to emails requesting product support within the same business day. The more detail you can provide, the quicker we will be able to understand your issues and find solutions. Data files, pictures, screen grabs, etc. are all very helpful in generating a well-thought-out solution.

Please email us at: support@microstrain.com

Telephone

MicroStrain's Support Engineers are available by phone Monday through Friday 9:00AM to 5:00PM local time. When calling MicroStrain, indicate to the receptionist that you are calling for product support and you will be promptly routed to a Support Engineer. Please have your equipment ready to test. Every attempt will be made to solve issues while you are on the line.

1.800.449.DVRT(3878) Toll Free in US
1.802.862.6629 telephone
1.802.863.4093 fax

Local time = GMT -05:00 (Eastern Time US & Canada)

SKYPE

MicroStrain's Support Engineers are available by SKYPE Monday through Friday 9:00AM to 5:00PM local. SKYPE name: **microstrain.orientation.support**