User Manual

Version 1.0.0

3DM-GX3™ Orientation SDK
for 3DM-GX3™-25

MicroStrain® Little Sensors, Big Ideas®

www.microstrain.com

©2009 by MicroStrain, Inc.
459 Hurricane Lane, Suite 102
Williston, VT 05495
Phone 802-862-6629
Fax 802-863-4093
www.microstrain.com
support@microstrain.com

ISSUED: 1 August 2009

Table of Contents

OVERVIEW **4**
DISCLAIMER **5**
SDK SUPPORT **5**
DATA COMMUNICATIONS PROTOCOL **6**
VISUAL STUDIO C++ 2003/2008 EXPRESS CODE SAMPLES **8**
LABVIEW 7.1.1 CODE SAMPLES **13**
VISUAL STUDIO VISUAL BASIC 2005 CODE SAMPLES **15**
VISUAL BASIC 6.0 CODE SAMPLES **21**
VIRTUAL COMM PORT DRIVER **26**
SUGGESTED DEBUGGING TOOLS **27**
- LOOKRS232 27
- SERIAL PORT MONITOR 28
- COMM OPERATOR 28

SUPPORT **29**
- OVERVIEW 29
- WEB 29
- EMAIL 29
- TELEPHONE 29
- SKYPE 29

Overview

The MicroStrain 3DM-GX3™ Orientation Software Development Kit (SDK) is a set of development tools that allows a user to create applications which communicate with MicroStrain's 3DM-GX3™-25 orientation sensors.

The SDK is targeted at providing the user with the tools required to build Windows applications for the RS-232, USB and TTL physical communication interfaces of the MicroStrain orientation sensor.

The SDK consists of the following major components:

- Data Communications Protocol manual
- Sample Code:
    - Visual Studio C++ 2003/2008 Express
    - 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 3DM-GX3™-25 orientation sensor;
- 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 orientation sensor 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 Data Communications Protocol is a separate document which describes how to communicate with the MicroStrain 3DM-GX3™-25 orientation sensor. The protocol changes from time-to-time in lockstep with firmware changes on the orientation sensor. The most current protocol is maintained on the MicroStrain web site at: http://www.microstrain.com/pdf/3DM-GX3 Data Communications Protocol.pdf.
The Data Communications Protocol manual contains a version number on the cover, a revision date on the second page and a statement concerning which firmwares are supported. The user should confirm the firmware version of the particular orientation sensor being used.

The MicroStrain 3DM-GX3™-25 orientation sensor is available with several communication interface options including USB, RS-232, and TTL. The Data Communications Protocol supports communication through all of these interfaces.

The Data Communications Protocol is a set of serial commands and responses designed specifically for MicroStrain's 3DM-GX3™-25 orientation sensor. For the most part, the communications protocol consists of simple single byte binary commands with fixed length binary data records as replies. Most replies include an "echo" byte and a checksum to do simple data integrity checks. All communications with the sensor is accomplished using a standard serial "COM" port. A typical command and response, as an example Euler angles, is demonstrated in **Table 1** below. A single byte 0xCE is sent by the host to the sensor. The sensor responds with a 19 byte packet containing the header (echo) byte, 4 bytes representing the Roll value as a float, 4 bytes for Pitch, 4 bytes for Yaw, 4 bytes indicating the sample time from the sensor's on-board clock, and a 2 byte checksum to determine packet integrity.

Euler Angles (0xCE) –Single Byte Form

Command	**1 byte**
Command Byte	0xCE
Response	**19 bytes**
Byte 1	Header = 0xCE
Bytes 2-5	Roll
Bytes 6-9	Pitch
Bytes 10-13	Yaw
Bytes 14-17	Timer
Bytes 18-19	Checksum

Table 1

The Data Communications Protocol is the basis of this Software Development Kit. The various sample codes, all later described and detailed in this manual, are constructed directly from the Data Communications Protocol.

We emphasize that careful reading of the Data Communications Protocol manual is good practice and will speed the user's understanding and success in creating applications.

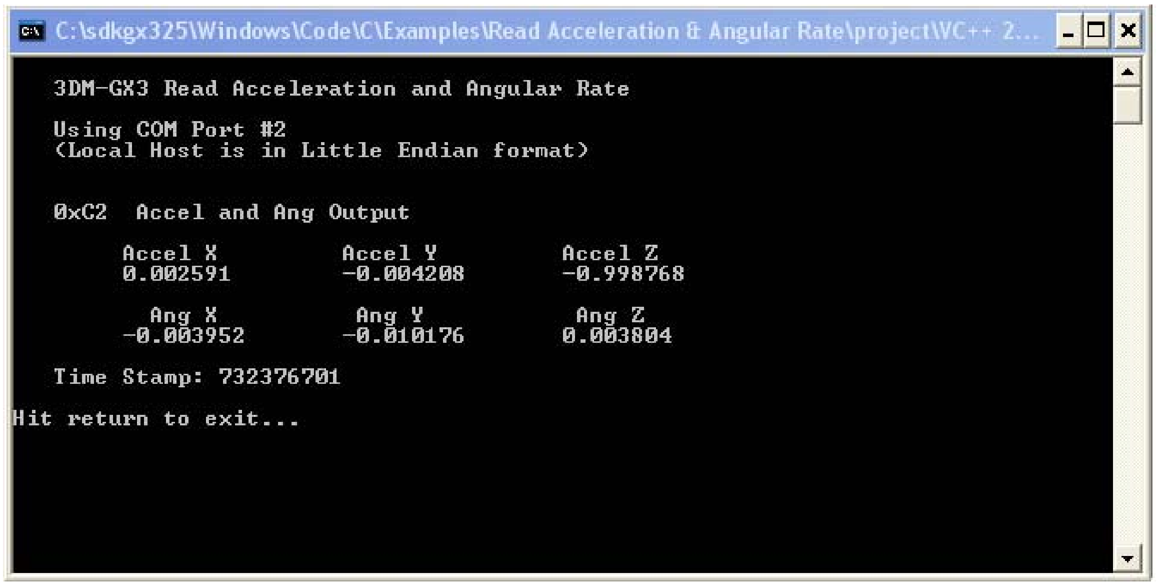
Visual Studio C++ 2003/2008 Express Code Samples

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 and Microsoft C++ 2008 Express Edition version 9.0.30729.1 SP with Microsoft .NET Framework version 3.5 SP1 using only objects and components native to the IDEs. No third party components or objects were used.

The Visual Studio C++ 2003/2008 Express 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 indicates the command being demonstrated (in this case, the 0xC2 packet). The Console window indicates which comm port is being used, whether the host is using Big Endian or Little Endian byte order, the output values of the sensor and the timestamp.


C:\sdkgx325\Windows\Code\C\Examples\Read Acceleration & Angular Rate\project\VC++ 2...

```
   3DM-GX3 Read Acceleration and Angular Rate

   Using COM Port #2
   (Local Host is in Little Endian format)

   0xC2  Accel and Ang Output

        Accel X         Accel Y         Accel Z
        0.002591        -0.004208       -0.998768

          Ang X           Ang Y           Ang Z
        -0.003952       -0.010176       0.003804

   Time Stamp: 732376701

Hit return to exit...
```

Figure 1

Pseudo code

- The application is given either a single argument or run without argument.
- If it is given an argument, it assumes it is a comm port number, attempts to connect to that port number and checks that it is a valid port between 1 and 256.
- If no argument is given, it will scan all ports, looking for valid ports and attempt a connection.
- Upon connection, the application configures the port with baud rate, parity, stop bits, timeout, etc.
- If successful, it sends a polling command to the sensor.

- If a data packet is returned from the sensor, the application will perform a checksum.
- If the checksum is successful, it will parse the data packet into the X, Y, Z acceleration and X, Y, Z angular rate values and display them to the screen.
- The application will terminate.

```
/*----------------------------------------------------------------------
-
* Read Acceleration and Angular Rate
*
* This command line application demonstrates how to use the 3DM-GX3 SDK
to retrieve an Acceleration and Angular Rate data sample from a
MicroStrain orientation sensor.  The only argument is the port number
for the sensor. This can be obtained by using the Windows Device
Manager. Under the "Ports (COM & LPT)" listing, the MicroStrain device
will be listed as either a "CP210x USB to UART Bridge Controller" or a
"MicroStrain Virtual COM Port".  If no port argument is specified, then
a scanport function is called which will look for an attached sensor by
scanning the serial ports.
*----------------------------------------------------------------------
*/
#include <stdio.h>
#include "ms_basic_type.h"
#include "i3dmgx3_Errors.h"
#include "i3dmgx3_Serial.h"
#include "i3dmgx3_Cmd.h"
#include "i3dmgx3_Utils.h"
#include "win_scanports.h"
int GetComPort(); //prompt user for comport and opens it
int OnGetSerialPorts();
/*---------------------------------------------------------------------
*/
int main(int argc, char **argv) {

        s32 zvert=0;
        BOOL endloopy = FALSE;

      s16 portNum;
      s16 deviceNum = 0;
      s16 i;
        s16 Ccount=0;
      u16 value=0;
      s16 id_flag = 0;
      s16 errorCode;
      s16 tryPortNum = 1;
      unsigned char Record[79];                       //record returned
from device read where max size is 79
      C2Accel_AngRecord Accel_AngRecord;

      printf("\n   3DM-GX3 Read Acceleration and Angular Rate\n");

      /*-------- If user specifies a port, then use it */
      if (argc > 1) {
            tryPortNum = atoi(argv[1]);
            if (tryPortNum < 2 || tryPortNum > 256) {
                  printf("   usage:  i3dmgx3 <portNumber>\n");
                  printf("        valid ports are 2..256\n");
                  exit(1);
            }

              /*-------- open a port, map a device */
              portNum = i3dmgx3_openPort(tryPortNum, 115200, 8, 0, 1, 
1024, 1024);
              if (portNum<0) {
                printf("   port open failed.\n");
                printf("   Comm error %d, %s: ", portNum, 
explainError(portNum));
               goto Exit;
              }

        }else{
          portNum=OnGetSerialPorts();
          if(portNum<0)
             goto Exit;

        }
      printf("\n   Using COM Port #%d \n", portNum);

      /*-------- Set Comm Timeout values */
      errorCode = setCommTimeouts(portNum, 50, 50); /* Read & Write 
timeout values */
      if (errorCode!=I3DMGX3_COMM_OK) {
            printf("   setCommTimeouts failed on port:%d with 
errorcode:%d\n",portNum,errorCode);
            goto Exit;
      }

      /*-------- Disclose the byte order of host */
      if( TestByteOrder() !=BIG_ENDIAN)
            printf("   (Local Host is in Little Endian format)\n");
      else
            printf("   (Local Host is in Big Endian format)\n");
      printf("\n");

      /*-------- 0xC2 Accel and Ang rate Output --- Accel x y z and Ang 
x y z */
      printf("\n   0xC2  Accel and Ang Output  \n");


      errorCode = i3dmgx3_AccelAndAngRate(portNum, &Record[0]);
      if (errorCode < 0){
            printf("   Error Accel and AngRate - : %s\n", 
explainError(errorCode));
                endloopy =TRUE;
      }else{
            for (i=0; i<3; i++) {
                  Accel_AngRecord.Accel[i] = FloatFromBytes(&Record[1 + 
i*4]);      // extract float from byte array
                  Accel_AngRecord.AngRt[i] = FloatFromBytes(&Record[13 
+ i*4]);    // extract float from byte array
            }
            printf("\n\tAccel X\t\tAccel Y\t\tAccel Z\n");
            printf("  \t%f\t%f\t%f\n", Accel_AngRecord.Accel[0], 
Accel_AngRecord.Accel[1], Accel_AngRecord.Accel[2]);
            printf("\n\t  Ang X\t\t Ang Y\t\t Ang Z\n");
            printf("  \t%f\t%f\t%f\n", Accel_AngRecord.AngRt[0], 
Accel_AngRecord.AngRt[1], Accel_AngRecord.AngRt[2]);

            Accel_AngRecord.timer = convert2ulong(&Record[25]);
            printf("\n   Time Stamp: %u\n", Accel_AngRecord.timer);
        }

Exit:
      /*-------- close device */
      if (portNum >= 0)
            i3dmgx3_closeDevice(portNum);

      /*-------- wait for user to respond before exiting */
      printf("\nHit return to exit...\n");
      while (getchar() == EOF);
      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(){
      s16 vvportNum   = 0;
      int iComPort  = 0;
      int errCount = 0;
      int MaxFail = 5;
      //Get comport number ask user for the comport number and open it
      while(vvportNum == 0)
      {
            printf("Enter Comport to use:");
            scanf("%d", &iComPort);

            /* open a port, map a device */
                vvportNum = i3dmgx3_openPort(iComPort, 115200, 8, 0, 1, 
1024, 1024);
                if (vvportNum<0) {
                    printf("port open failed. ");
                    printf("Comm error %d, %s:\n", vvportNum, 
explainError(vvportNum));
                  if (errCount++ >= MaxFail)
                        exit(-1);
                  iComPort = 0;
            }
            else {
                        printf("\n   Using Comm Port #%d \n",
vvportNum);
            }
      }

      return vvportNum;
}
/*
***********************************************************************
****************
**  OnGetSerialPorts                   Checks for a valid 3dm-gx3-25
device and if found
**                                     gets a connection and returns
the portnumber
**
***********************************************************************
******************/
int OnGetSerialPorts()
{
      int i_count, t_count=0;
      unsigned char c_name[20];
    int  i_errorCode = 0;
    int i_portNum = 0;
      char c_check[]="3DM-GX3";
      char Verity[20];

      for(i_count=2; i_count<257; i_count++)
      {
            int i_portNum = i3dmgx3_openPort(i_count, 115200, 8, 0, 1,
1024, 1024);
                if (i_portNum>0) {
                    i_errorCode = i3dmgx3_getDeviceIdentiy(i_portNum,
2, &c_name[0]);

                     memcpy(Verity,c_name,20);
                     //if (strstr(c_name,c_check) != NULL){

                              if (strstr(Verity,c_check) != NULL){
                             return i_portNum;
                           } //End string compare

                       else i3dmgx3_closeDevice(i_count);

                 }// end open port

      }// end for loop
         printf("No 3DM-GX3 devices found, please check
connections\n");
       return -1;
 }
```

Code Sample 1

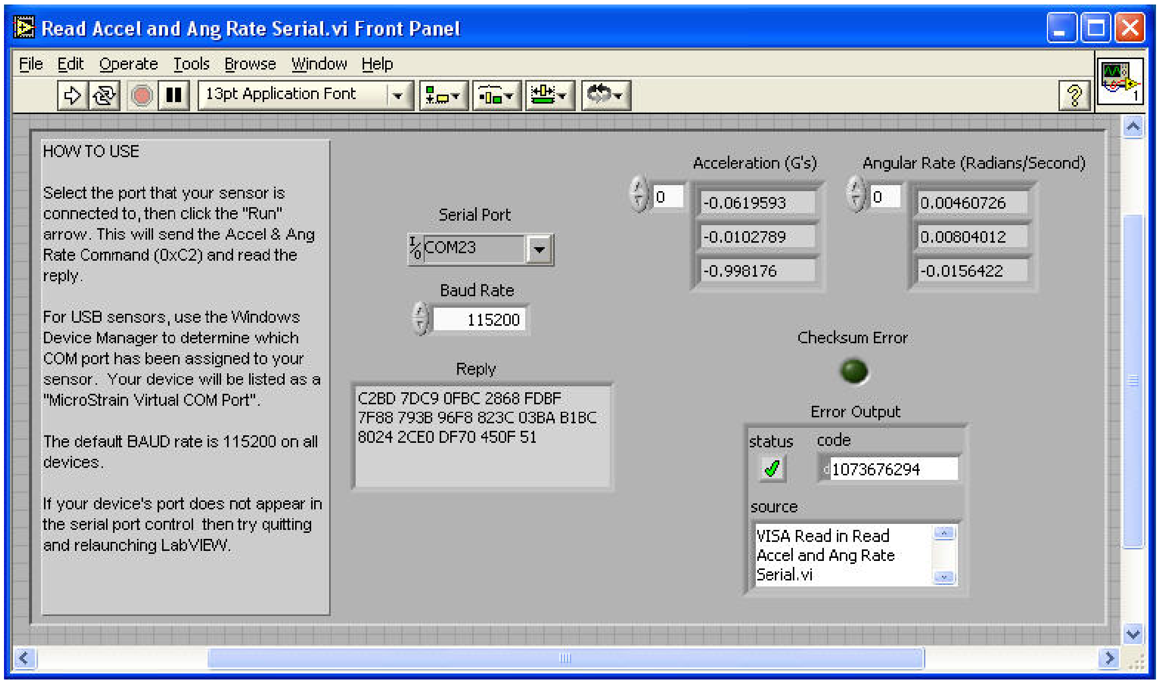
LabVIEW 7.1.1 Code Samples

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 serial port and baud rate. The Front Panel displays the data packet breakout (in this case, the 0xC2 packet). The Form has a red/green checksum indicator and an 'Error Output' box which displays messages to the user.



HOW TO USE

Select the port that your sensor is connected to, then click the "Run" arrow. This will send the Accel & Ang Rate Command (0xC2) and read the reply.

For USB sensors, use the Windows Device Manager to determine which COM port has been assigned to your sensor. Your device will be listed as a "MicroStrain Virtual COM Port".

The default BAUD rate is 115200 on all devices.

If your device's port does not appear in the serial port control then try quitting and relaunching LabVIEW.

Serial Port: COM23

Baud Rate: 115200

Reply:

```
C2BD 7DC9 0FBC 2868 FDBF
7F88 793B 96F8 823C 03BA B1BC
8024 2CE0 DF70 450F 51
```

Acceleration (G's)	Angular Rate (Radians/Second)
-0.0619593	0.00460726
-0.0102789	0.00804012
-0.998176	-0.0156422

Checksum Error ●

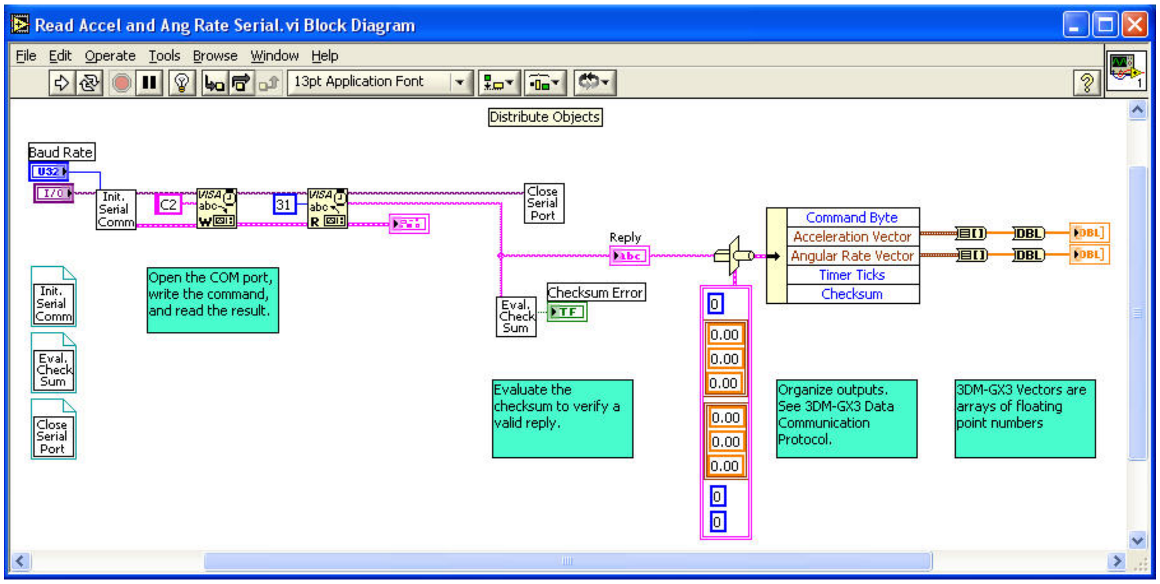
Error Output

status: ✓ code: 1073676294

source: VISA Read in Read Accel and Ang Rate Serial.vi

Figure 2

The Block Diagram is shown in **Figure 3**. The wiring path is linear. 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 checksum is evaluated, the packet is parsed and scaled, and the results are displayed on the Front Panel.



Open the COM port, write the command, and read the result.

Evaluate the checksum to verify a valid reply.

Organize outputs. See 3DM-GX3 Data Communication Protocol.

3DM-GX3 Vectors are arrays of floating point numbers

Figure 3

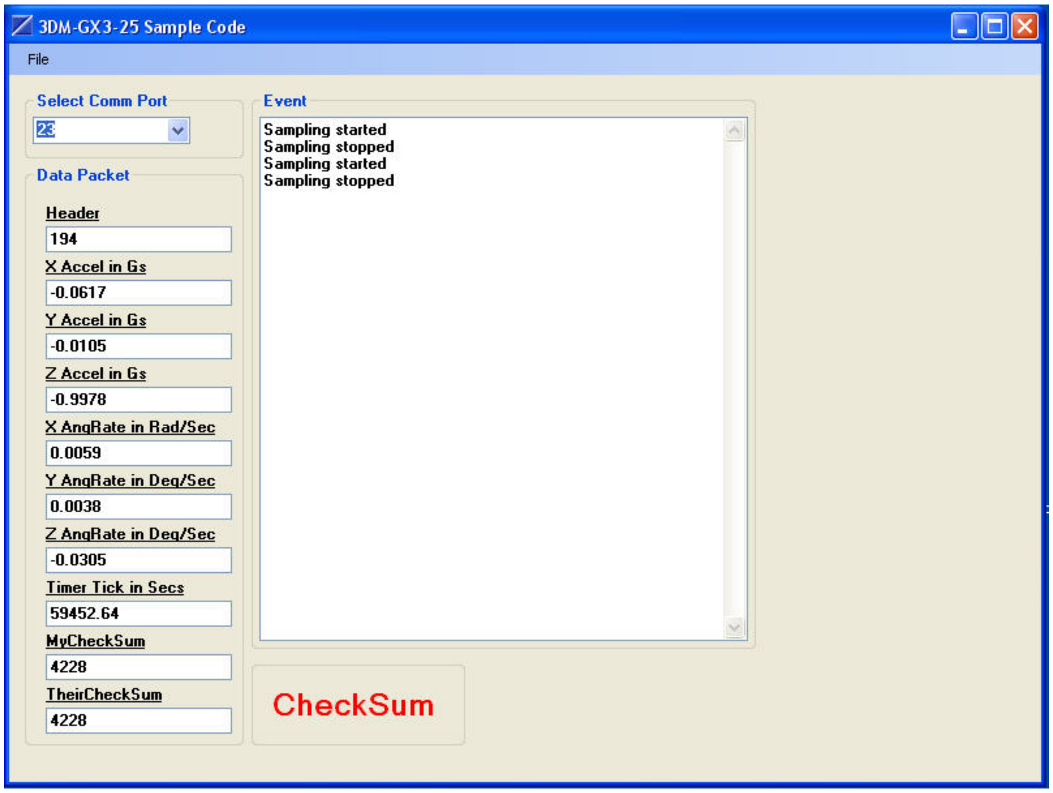
Visual Studio Visual Basic 2005 Code Samples

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. The Form displays the data packet breakout (in this case, the 0xC2 packet). The Form has a red/green checksum indicator and an 'Event' textbox which displays messages to the user.



Sampling started
Sampling stopped
Sampling started
Sampling stopped

Header	194
X Accel in Gs	-0.0617
Y Accel in Gs	-0.0105
Z Accel in Gs	-0.9978
X AngRate in Rad/Sec	0.0059
Y AngRate in Deg/Sec	0.0038
Z AngRate in Deg/Sec	-0.0305
Timer Tick in Secs	59452.64
MyCheckSum	4228
TheirCheckSum	4228

Figure 4

Pseudo code

- On **Main_Load**, the combo box for the comm port selection is populated.
- The user clicks File and Sample on the menu bar.
- **SampleToolStripMenuItem_Click** configures the comm port and sends the continuous mode command.

- **SerialPort1_DataReceived** fires as the result of the receive threshold hitting 31 bytes, checks the data packet integrity and sends the packet to the **CheckSum** function.
- **CheckSum** tests the checksum and if good, parses and scales the data and returns it to **SerialPort1_DataReceived.** If **CheckSum** fails, the packet is not parsed.
- Upon **CheckSum** return, **SerialPort1_DataReceived** either displays the good packet and flashes the checksum indicator green or flashes the checksum indicator red.
- The cycle continues as each 31 byte packet is received 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 ExitToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ExitToolStripMenuItem.Click
        'quit app
        End
    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

        'determine action based on menu check
        If SampleToolStripMenuItem.Checked = False Then
            'event
            Call ReportEvent("Sampling started")
            'set menu check
            SampleToolStripMenuItem.Checked = True
            'disable combo
            cmbCommPort.Enabled = False
            'set serial port number
            SerialPort1.PortName() = "COM" + Trim(cmbCommPort.Text)
            'set receive threshold
            SerialPort1.ReceivedBytesThreshold = 31
            'open serial port
            SerialPort1.Open()
            'populate byte array with continuous command for 0xC2
            'decimal equivalent of 0xC4 = 196
            'decimal equivalent of 0xC1 = 193
            'decimal equivalent of 0x29 = 41
            'decimal equivalent of 0xC2 = 194
            ReDim arrBytes(0 To 3)
            arrBytes(0) = 196
            arrBytes(1) = 193
            arrBytes(2) = 41
            arrBytes(3) = 194
            'send command
            SerialPort1.Write(arrBytes, 0, 4)
        ElseIf SampleToolStripMenuItem.Checked = True Then
            'open serial port
            SerialPort1.Close()
            'event
            Call ReportEvent("Sampling stopped")
            'set menu check
            SampleToolStripMenuItem.Checked = False
            'set checksum indicator
            lblCheckSum.ForeColor = Color.Red
            'enable combo
            cmbCommPort.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 checksum indicator
        lblCheckSum.ForeColor = Color.Red
        'enable combo
        cmbCommPort.Enabled = True
        'exit
        Exit Sub
    End Sub
```

```
    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

        'Sets a value indicating whether to catch calls on the wrong
        thread that access a control's Handle property
        txtHeader.CheckForIllegalCrossThreadCalls = False
        txtAccelX.CheckForIllegalCrossThreadCalls = False
        txtAccelY.CheckForIllegalCrossThreadCalls = False
        txtAccelZ.CheckForIllegalCrossThreadCalls = False
        txtAngRateX.CheckForIllegalCrossThreadCalls = False
        txtAngRateY.CheckForIllegalCrossThreadCalls = False
        txtAngRateZ.CheckForIllegalCrossThreadCalls = False
        txtTimerTick.CheckForIllegalCrossThreadCalls = False
        txtMyCheckSum.CheckForIllegalCrossThreadCalls = False
        txtTheirCheckSum.CheckForIllegalCrossThreadCalls = False
    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 Function CheckSum() As Boolean
        'set error handler
        On Error GoTo ErrorHandler

        '-----calc TheirCheckSum
        'dim
        Dim X As Double
        'zero variable
        TheirCheckSum = 0
        'calc TheirCheckSum
        For X = 0 To 28
            TheirCheckSum = TheirCheckSum + CDbl(arrBytes(X))
        Next X

        'handle CheckSum rollover
        TheirCheckSum = TheirCheckSum Mod 65536

        'calc MyCheckSum
        MyCheckSum = (CDbl(arrBytes(29)) * 256) + arrBytes(30)

        'compare checksums
        If MyCheckSum = TheirCheckSum Then
            ''set checksum return
            CheckSum = True
            'calc datapoints
            MyHeader = arrBytes(0)
            'convert 32bit floating point values in IEEE-754 format
            arrTemp(0) = arrBytes(4)
            arrTemp(1) = arrBytes(3)
            arrTemp(2) = arrBytes(2)
            arrTemp(3) = arrBytes(1)
            AccelX = BitConverter.ToSingle(arrTemp, 0)
            'round
            AccelX = Math.Round(AccelX, 4)
            'convert 32bit floating point values in IEEE-754 format
            arrTemp(0) = arrBytes(8)
            arrTemp(1) = arrBytes(7)
            arrTemp(2) = arrBytes(6)
            arrTemp(3) = arrBytes(5)
            AccelY = BitConverter.ToSingle(arrTemp, 0)
            'round
            AccelY = Math.Round(AccelY, 4)
            'convert 32bit floating point values in IEEE-754 format
            arrTemp(0) = arrBytes(12)
            arrTemp(1) = arrBytes(11)
            arrTemp(2) = arrBytes(10)
            arrTemp(3) = arrBytes(9)
            AccelZ = BitConverter.ToSingle(arrTemp, 0)
            'round
            AccelZ = Math.Round(AccelZ, 4)
            'convert 32bit floating point values in IEEE-754 format
            arrTemp(0) = arrBytes(16)
            arrTemp(1) = arrBytes(15)
            arrTemp(2) = arrBytes(14)
            arrTemp(3) = arrBytes(13)
            AngRateX = BitConverter.ToSingle(arrTemp, 0)
            'round
            AngRateX = Math.Round(AngRateX, 4)
            'convert 32bit floating point values in IEEE-754 format
            arrTemp(0) = arrBytes(20)
            arrTemp(1) = arrBytes(19)
            arrTemp(2) = arrBytes(18)
            arrTemp(3) = arrBytes(17)
            AngRateY = BitConverter.ToSingle(arrTemp, 0)
            'round
            AngRateY = Math.Round(AngRateY, 4)
            'convert 32bit floating point values in IEEE-754 format
            arrTemp(0) = arrBytes(24)
            arrTemp(1) = arrBytes(23)
            arrTemp(2) = arrBytes(22)
            arrTemp(3) = arrBytes(21)
            AngRateZ = BitConverter.ToSingle(arrTemp, 0)
            'round
            AngRateZ = Math.Round(AngRateZ, 4)
            'calc timer tick
            TimerTick = CalcTimerTick(25)
        Else
            'set checksum return
            CheckSum = False
        End If
        'exit
        Exit Function
ErrorHandler:
        'set checksum return
        CheckSum = False
        'exit
        Exit Function
    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 checksum indicator
            lblCheckSum.ForeColor = Color.Red
            'exit
            Exit Sub
        End If

        'test 31 byte return and header
        If Not UBound(arrBytes) = 30 And Not arrBytes(0) = 194 Then
            'set checksum indicator
            lblCheckSum.ForeColor = Color.Red
            'exit
            Exit Sub
        End If

        'test checksum
        If CheckSum() = False Then
            'set checksum indicator
            lblCheckSum.ForeColor = Color.Red
            'exit
            Exit Sub
        End If

        'set checksum indicator
        lblCheckSum.ForeColor = Color.Green

        'display data
        txtHeader.Text = MyHeader
        txtAccelX.Text = AccelX
        txtAccelY.Text = AccelY
        txtAccelZ.Text = AccelZ
        txtAngRateX.Text = AngRateX
        txtAngRateY.Text = AngRateY
        txtAngRateZ.Text = AngRateZ
        txtTimerTick.Text = TimerTick
        txtMyCheckSum.Text = MyCheckSum
        txtTheirCheckSum.Text = TheirCheckSum

        Exit Sub
ErrorHandler:
        'close serial port
        If SerialPort1.IsOpen = True Then SerialPort1.Close()
    End Sub
End Class
```

Code Sample 2

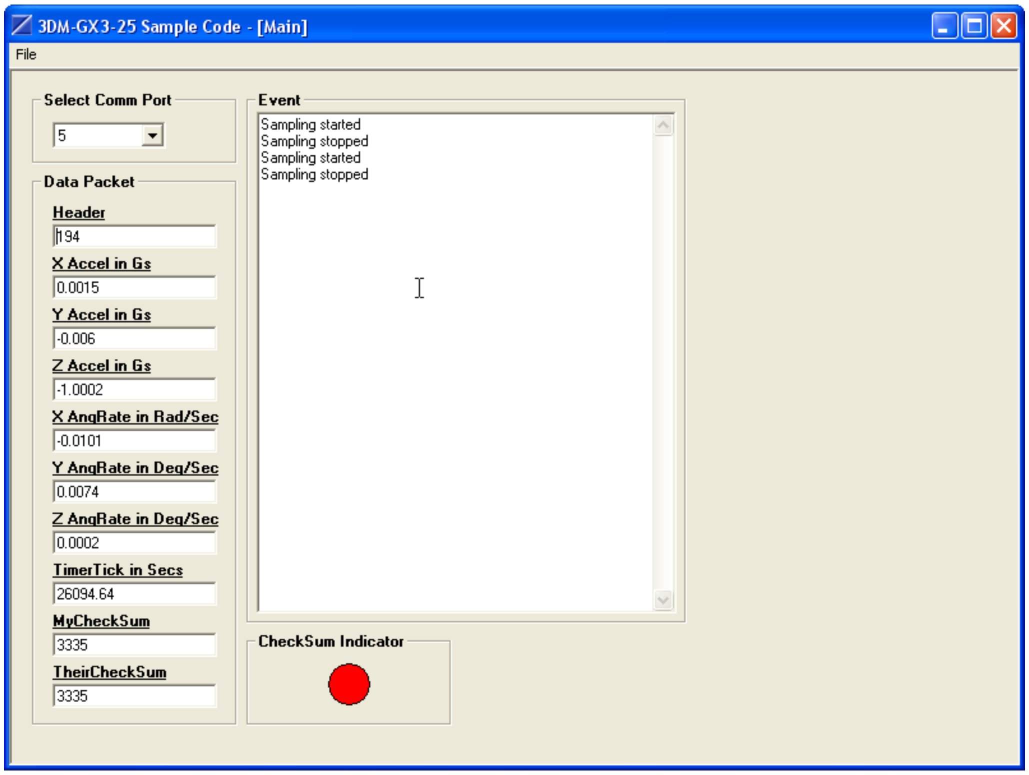
Visual Basic 6.0 Code Samples

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. The Form displays the data packet breakout (in this case, the 0xC2 packet). The Form has a red/green checksum indicator and an 'Event' textbox which displays messages to the user.



Select Comm Port: 5

Sampling started
Sampling stopped
Sampling started
Sampling stopped

Header	194
X Accel in Gs	0.0015
Y Accel in Gs	-0.006
Z Accel in Gs	-1.0002
X AngRate in Rad/Sec	-0.0101
Y AngRate in Deg/Sec	0.0074
Z AngRate in Deg/Sec	0.0002
TimerTick in Secs	26094.64
MyCheckSum	3335
TheirCheckSum	3335

Figure 5

Sample code

- On **Form_Load**, the combo box for the comm port selection is populated.
- The user clicks File and Sample on the menu bar.
- **mnuSample_Click** configures the comm port and sends the continuous mode command.

- **MSComm1_OnComm** fires as the result of the receive threshold hitting 31 bytes, checks the data packet integrity and sends the packet to the **CheckSum** function.
- **CheckSum** tests the checksum and if good, parses and scales the data and returns it to **MSComm1_OnComm.** If **CheckSum** fails, the packet is not parsed.
- Upon **CheckSum** return, **MSComm1_OnComm** either displays the good packet and flashes the LED green or flashes the LED red.
- The cycle continues as each 31 byte packet is received 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.
- **mnuExit_Click** terminates the application.

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

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

```
Private Sub mnuSample_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

'determine action based on menu check
If mnuSample.Checked = False Then
   'event
   Call ReportEvent("Sampling started")
   'set menu check
   mnuSample.Checked = True
   'disable combo
   cmbCommPort.Enabled = False
   'set comm port
   MSComm1.CommPort = Val(cmbCommPort.Text)
```

```
    'set receive threshold to 31 bytes
    MSComm1.RThreshold = 31
    'open comm port
    MSComm1.PortOpen = True
    'populate byte array with continuous command for 0xC2
    'decimal equivalent of 0xC4 = 196
    'decimal equivalent of 0xC1 = 193
    'decimal equivalent of 0x29 = 41
    'decimal equivalent of 0xC2 = 194
    arrBytes = ChrB(196) + ChrB(193) + ChrB(41) + ChrB(194)
    'send continuous command
    MSComm1.Output = arrBytes
ElseIf mnuSample.Checked = True Then
    'event
    Call ReportEvent("Sampling stopped")
    'set menu check
    mnuSample.Checked = False
    'close comm port
    MSComm1.PortOpen = False
    'enable combo
    cmbCommPort.Enabled = True
    'set checksum indicator
    shpLED.FillColor = vbRed
End If

Exit Sub
ErrorHandler:
    'set menu check
    mnuSample.Checked = False
    'close comm port
    If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
    'enable combo
    cmbCommPort.Enabled = True
    'event
    Call ReportEvent("Sampling errored: " + Err.Description)
    'set checksum indicator
    shpLED.FillColor = vbRed
    '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 31 byte return and header
If Not UBound(arrBytes) = 30 Or Not arrBytes(0) = 194 Then
   'set checksum indicator
   shpLED.FillColor = vbRed
   'exit sub
   Exit Sub
End If

'test checksum
If CheckSum = False Then
   'set checksum indicator
   shpLED.FillColor = vbRed
   'exit sub
   Exit Sub
End If

'set checksum indicator
shpLED.FillColor = vbGreen

'display data
txtHeader.Text = MyHeader
txtAccelX.Text = Round(AccelX, 4)
txtAccelY.Text = Round(AccelY, 4)
txtAccelZ.Text = Round(AccelZ, 4)
txtAngRateX.Text = Round(AngRateX, 4)
txtAngRateY.Text = Round(AngRateY, 4)
txtAngRateZ.Text = Round(AngRateZ, 4)
txtTimerTick.Text = TimerTick
txtMyCheckSum.Text = MyCheckSum
txtTheirCheckSum.Text = TheirCheckSum

End Sub
```

Private Function **CheckSum**() As Boolean

```
'set error handler
On Error GoTo ErrorHandler

'-----calc TheirCheckSum
'dim
Dim X As Double
'zero variable
```

```
TheirCheckSum = 0
'calc TheirCheckSum
For X = 0 To 28
   TheirCheckSum = TheirCheckSum + CDbl(arrBytes(X))
Next X

'handle CheckSum rollover
TheirCheckSum = TheirCheckSum Mod 65536

'calc MyCheckSum
MyCheckSum = (CDbl(arrBytes(29)) * 256) + arrBytes(30)

'compare checksums
If MyCheckSum = TheirCheckSum Then
   ''set checksum return
   CheckSum = True
   'calc datapoints
   MyHeader = arrBytes(0)
   AccelX = FloatConversion(1)
   AccelY = FloatConversion(5)
   AccelZ = FloatConversion(9)
   AngRateX = FloatConversion(13)
   AngRateY = FloatConversion(17)
   AngRateZ = FloatConversion(21)
   TimerTick = CalcTimerTick(25)
Else
   'set checksum return
   CheckSum = False
End If

'exit
Exit Function
ErrorHandler:
   'set checksum return
   CheckSum = False
   'exit
   Exit Function
End Function
```

Code Sample 3

Virtual Comm Port Driver

The MicroStrain 3DM-GX3™ orientation sensor is available with several communication interfaces as previously stated. In particular the USB interface uses a special driver installed in the Windows operating system.

The USB interface, from a physical standpoint, resides in a connection from the microcontroller, passing through the Micro-D connector and communication cable to a standard USB connector on the host computer.

This physical architecture is supported by a CDC class USB driver 'USBSER.SYS', typically found in the driver folder of the Windows System32. This driver is required for device operation as a Virtual COM Port to facilitate host communication. The required INF file is normally installed during installation of MicroStrain's data acquisition and display software.

With the installation of this driver, the developer will find that communication between host and sensor will be, for all intents and purposes, typical serial communication. The various coding languages will interact as if a standard serial port existed.

To determine if the driver is properly installed, connect the orientation sensor to the host and follow these steps:

- Click Start on the Windows desktop.
- Click Control Panel.
- Click System icon.
- System Properties window appears.
- Click Hardware tab.
- Click Device Manager.
- Device Manger window appears.
- Locate Ports(Com&LPT) in the tree.
- Double-click the + to further open the Ports tree.
- Do you see a device named 'MicroStrain Virtual COM Port (ComX)'? (with X representing a comm port number).
- If so, the driver is installed.
- If not, the orientation sensor is malfunctioning or the driver is not installed.

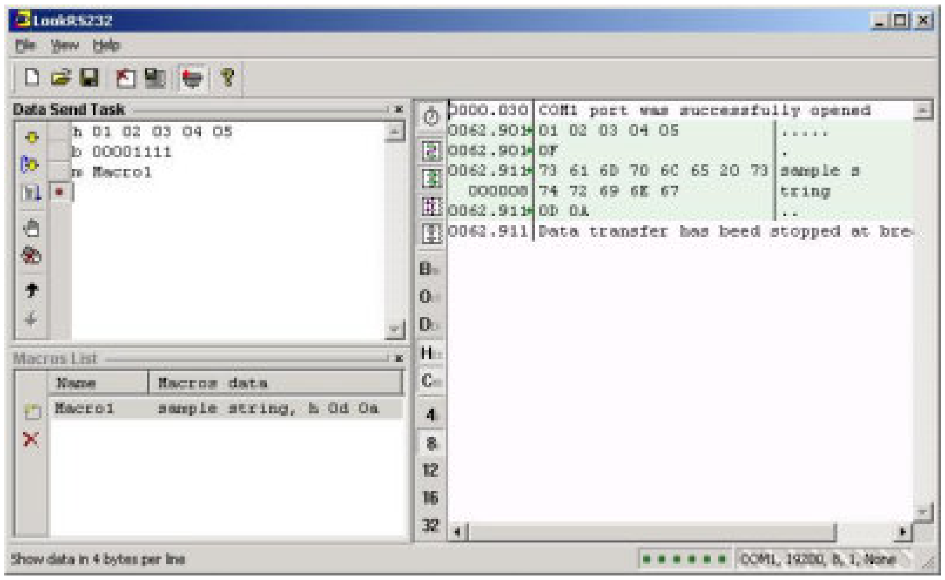
Suggested Debugging Tools

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.



```
h 01 02 03 04 05
b 00001111
m Macro1
```

Name	Macros data
Macro1	sample string, h 0d 0a

```
0000.030 COM1 port was successfully opened
0062.901 01 02 03 04 05              .....
0062.901 0F                          .
0062.911 73 61 6D 70 6C 65 20 73     sample s
  000008 74 72 69 6E 67              tring
0062.911 0D 0A                       ..
0062.911 Data transfer has beed stopped at bre
```

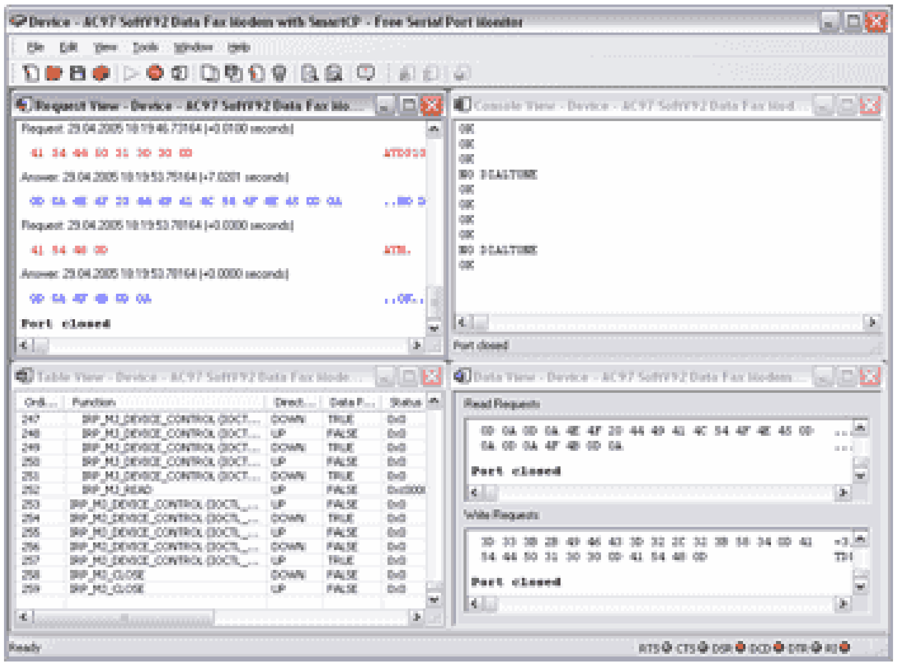
Show data in 4 bytes per line

COM1, 19200, 8, 1, None

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.


Device - AC97 SoftV92 Data Fax Modem with SmartCP - Free Serial Port Monitor
File Edit View Tools Window Help
Request View - Device - AC97 SoftV92 Data Fax Mo...
Console View - Device - AC97 SoftV92 Data Fax Mod...
Table View - Device - AC97 SoftV92 Data Fax Mode...
Data View - Device - AC97 SoftV92 Data Fax Modem...
Read Requests
Write Requests
Ready
RTS CTS DSR DCD DTR RI

```
Request: 29.04.2005 10:19:46.7364 (+0.0100 seconds)
 41 54 44 50 31 30 30 0D                          ATDP100.
Answer: 29.04.2005 10:19:53.7864 (+7.0201 seconds)
 0D 0A 4E 4F 20 44 49 41 4C 54 4F 4E 45 0D 0A    ..NO D
Request: 29.04.2005 10:19:53.7864 (+0.0000 seconds)
 41 54 48 0D                                      ATH.
Answer: 29.04.2005 10:19:53.7864 (+0.0000 seconds)
 0D 0A 4F 4B 0D 0A                                ..OK..
Port closed
```

```
OK
OK
OK
NO DIALTONE
OK
OK
OK
OK
NO DIALTONE
OK
```

Port closed

Ordi...	Function	Direct...	Data F...	Status
247	IRP_MJ_DEVICE_CONTROL (IOCT...	DOWN	TRUE	0x0
248	IRP_MJ_DEVICE_CONTROL (IOCT...	UP	FALSE	0x0
249	IRP_MJ_DEVICE_CONTROL (IOCT...	DOWN	TRUE	0x0
250	IRP_MJ_DEVICE_CONTROL (IOCT...	UP	FALSE	0x0
251	IRP_MJ_DEVICE_CONTROL (IOCT...	DOWN	TRUE	0x0
252	IRP_MJ_READ	UP	FALSE	0x0000
253	IRP_MJ_DEVICE_CONTROL (IOCTL_...	UP	FALSE	0x0
254	IRP_MJ_DEVICE_CONTROL (IOCTL_...	DOWN	TRUE	0x0
255	IRP_MJ_DEVICE_CONTROL (IOCTL_...	UP	FALSE	0x0
256	IRP_MJ_DEVICE_CONTROL (IOCTL_...	DOWN	FALSE	0x0
257	IRP_MJ_DEVICE_CONTROL (IOCTL_...	UP	TRUE	0x0
258	IRP_MJ_CLOSE	DOWN	FALSE	0x0
259	IRP_MJ_CLOSE	UP	FALSE	0x0

```
 0D 0A 0D 0A 4E 4F 20 44 49 41 4C 54 4F 4E 45 0D   ...
 0A 0D 0A 4F 4B 0D 0A                              ...
Port closed
```

```
 30 33 38 28 49 46 43 3D 32 2C 32 38 58 34 0D 41   +3
 54 44 50 31 30 30 0D 41 54 48 0D                  TD
Port closed
```

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**