X3 DLL User Guide

3/19/2008

1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

1. Introduction 3
1.1 Purpose 3
1.2 Scope 3
2 Installation Instructions 4
2.1 Installing Demo Software (Windows) 4
2.2 Installing Cables and Accessories 4
2.3 Installing Driver Software for X3 (Windows) 5
3 Getting Started 9
3.1 Run Demo (X3 Demo) 9
3.2 Hello-World Applications 10
4 Function Calls 13
4.1 X3_InitComm 13
4.2 X3_SetBaudRate 14
4.3 X3_CloseComm 15
4.4 X3_GetAllAngles 16
4.5 X3_GetAngle 17
4.6 X3_SetAngle 18
4.7 X3_GetAllAngleOffsets 20
4.8 X3_SetAngleOffset 21
4.9 X3_GetAllData 22
4.10 X3_GetAllDirections 23
4.11 X3_SetDirection 25
4.12 X3_GetDamping 26
4.13 X3_SetDamping 27
4.14 X3_GetAngleOutputRange 28
4.15 X3_SetAngleOutputRange 29
4.16 X3_GetDeviceInfo 30
4.17 X3_GetOutputConfig 31
4.18 X3_SetOutputConfig 33
4.19 X3_GetOutputUpdateRate 35
4.20 X3_SetOutputUpdateRate 36
4.21 X3_GetOutput 38
4.22 X3_SetOutput 39
4.23 X3_GetStartupDelay 40
4.24 X3_SetStartupDelay 41
5 Constants 42
6 Error Codes 42
7 Enumerations 43

1. Introduction

1.1 Purpose

This document describes how to install and use the X3 DLL on a PC running Windows using the RS-232 port or the US Digital USB-232 USB interface.
The features of the X3 device are made accessible by using the functions provided in the X3User.dll.

1.2 Scope

This document describes how to use each of the available interface methods provided by the X3. The following chapters are included:

- Installation Instructions
- Getting Started
- Function Calls
- Constants
- Error Codes
- Enumerations

2 Installation Instructions

2.1 Installing Demo Software (Windows)

Download the X3 demo software from US Digital website http://www.usdigital.com/software/X3/X3Demo.zip or insert the USD-SW CD and run the X3 Demo Installation.

The installation program will install the X3 Demo and the appropriate libraries and drivers.

2.2 Installing Cables and Accessories

2.2.1 X3 Cables and Accessories

There are two cabling options to choose from when it comes to RS232 communication with an X3 (RS232 cable or USB-232 adapter).

2.2.1.1 RS232 Cable

Connect the DB-9 connector on the RS232 cable CA-9376 to an available DB-9 COM port on laptop or PC.
Connect the 4-pin connector on the other end of the RS232 cable to the X3.
Connect PS-5 power supply power jack to the power jack on the RS232 cable and then plug in the PS-5 AC adapter to an AC outlet.

2.2.1.2 USB-232 Adapter

Connect the 5-pin connector end of a CA-3932 cable to the 5-pin connector of the USB-232 adapter.
Connect the 4-pin connector on the other end of the CA-3932 cable to the X3.

Connect the USB cable to the USB port of the USB-232 adapter and the other end into an available USB port on laptop or PC.

If this is the first time connecting a USB-232 adapter to a USB port, then Windows will display the Found New Hardware Wizard.

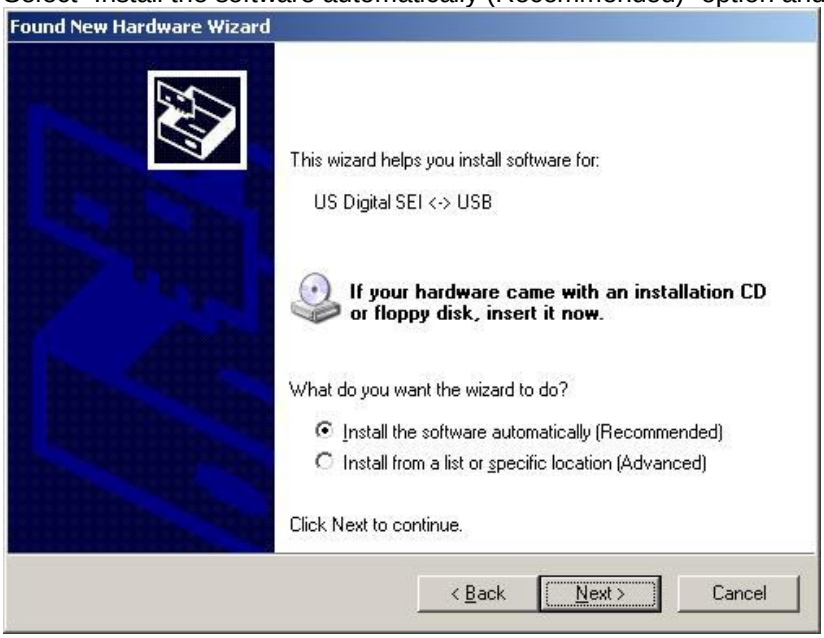
2.3 Installing Driver Software for X3 (Windows)

Skip this section if not using a USB-232 adapter.

2.3.1 Installation Instructions for Windows XP

After installing the X3 Software and inserting a USB-232 device into an available USB port, Windows will display the following “Found New Hardware Wizard” dialog.
Select “No, not this time” option and click “Next”.

Select “Install the software automatically (Recommended)” option and click “Next”.



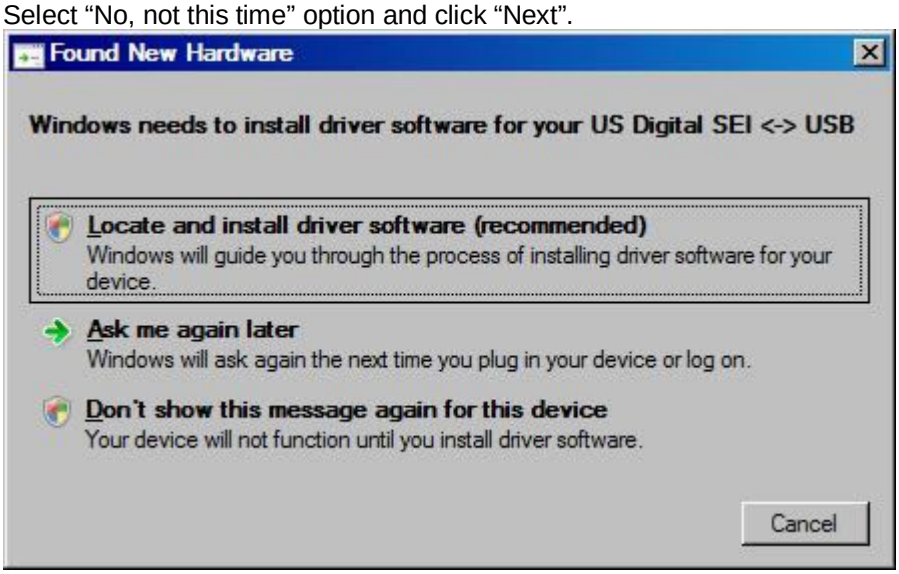
Windows will automatically search for the appropriate drivers.

The USB Serial drivers included in this installation are currently being Windows Logo certified. Until certified drivers are available, please click the “Continue Anyway” button when the following Hardware Installation dialog is displayed.

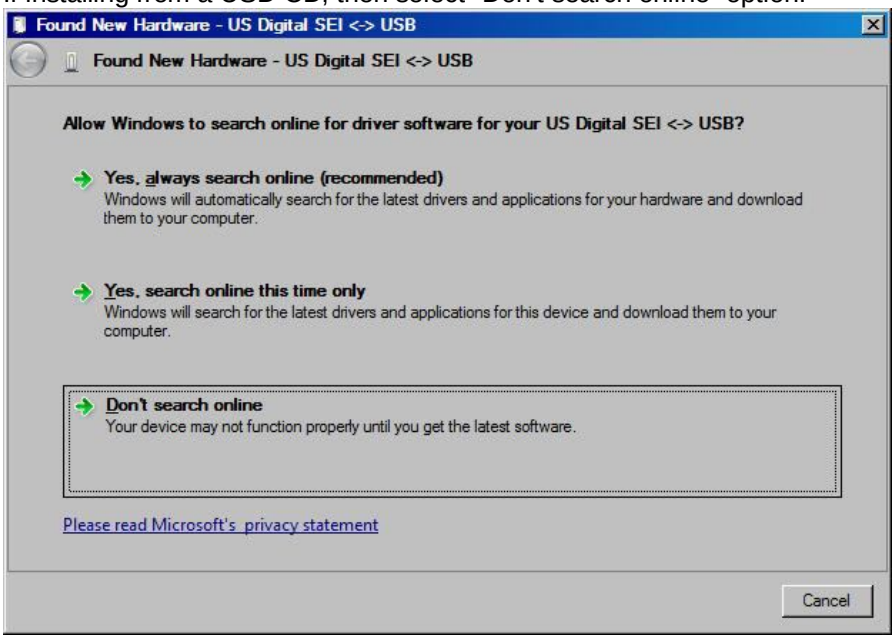
Note: The Found New Hardware Wizard will appear twice as two different drivers are required to support the virtual COM port.

2.3.2 Installation Instructions for Windows Vista

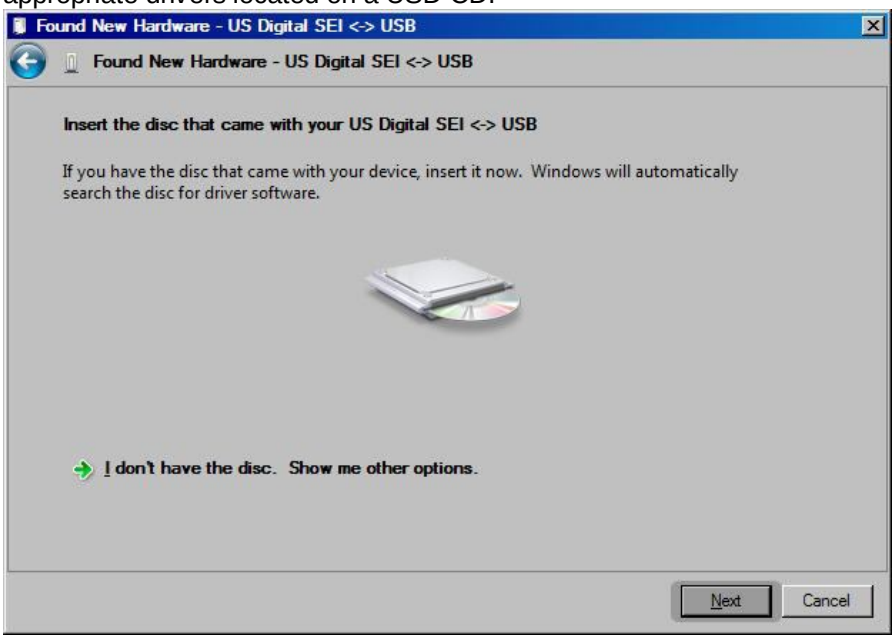
After installing the X3 Software and inserting a USB-232 device into an available USB port, Windows will display the following “Found New Hardware Wizard” dialog.
Select “No, not this time” option and click “Next”.



If installing from a USD CD, then select “Don’t search online” option.



Make sure that USD CD is inserted and click “Next.” Windows will automatically search for the appropriate drivers located on a USD CD.



The USB Serial drivers included in this installation are currently being Windows Logo certified. Until certified drivers are available, please select the “Install this driver software anyway” option when the following Hardware Installation dialog is displayed.

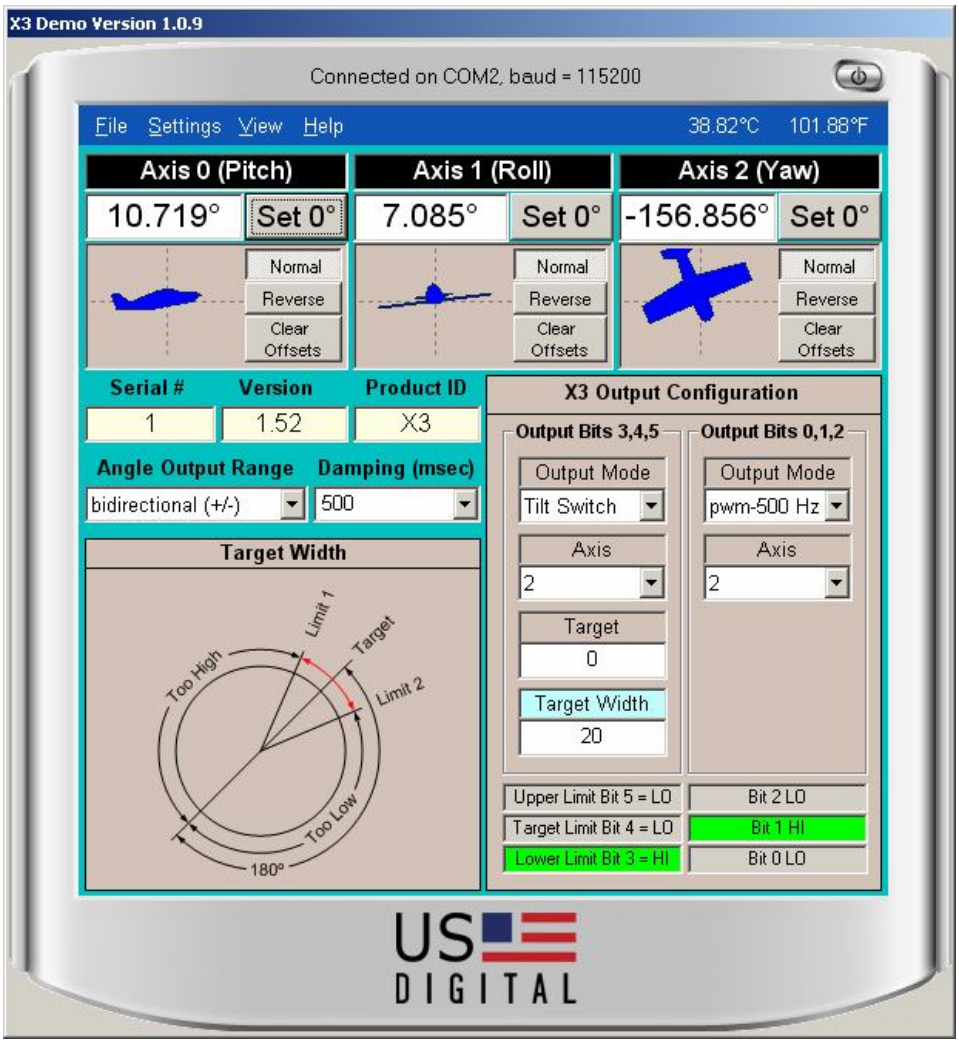
Note: The Hardware Installation dialog will be displayed twice as two different drivers are required to support the virtual COM port.

3 Getting Started

3.1 Run Demo (X3 Demo)

Once the X3 demo software is installed and the X3 device is connected to a laptop or PC, it's time to run the demo program.

Select the X3 Demo from the Start\Programs\US Digital\X3 Software menu.The following demo application window will be displayed. The X3 Demo will allow standard configuration changes to be made to an attached X3 device. The demo provides context sensitive tool-tip help information. Simply move mouse cursor over a user input to display tool-tip information. The source code for the X3 Demo is included in the X3 demo software installation.



3.2 Hello-World Applications

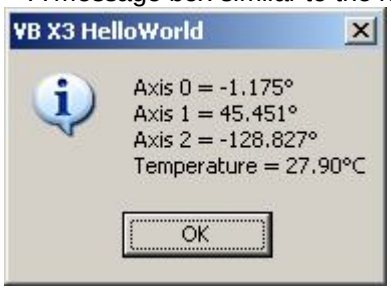
Now let's take a look at how easy it is to start communicating with the X3 device by writing a few lines of code.

3.2.1 VB Example

1. Open a new Visual Basic 6.0 and create a new Standard EXE project.
2. Include the X3UserDeclarations.bas file in project by Clicking on Project\Add Module. Click on the tab labeled Existing and then locate and open X3UserDeclarations.bas file.
3. Double click on the form to view the Form_Load subroutine and then paste the following code within the Form_Load sub:

```
Dim lResult        As Long
Dim bytComPort     As Byte
Dim bytAddress     As Byte
Dim dblAngles(2)   As Double
Dim dblTemperature  As Double

' Set the COM port number your device is attached to.
bytComPort = 2
bytAddress = 0

' Attempt to open the COM port.
' Run the X3ConfigUtility to determine which COM port you're using.
lResult = X3_InitComm(bytComPort, e115200)

' Check if we're able to open the COM port.
If X3_SUCCESS = lResult Then
   ' Get the angle of each axis.
   X3_GetAllAngles bytComPort, bytAddress, dblAngles(0), dblAngles(1), dblAngles(2), dblTemperature
   MsgBox "Axis 0 = " & Format(dblAngles(0), "0.000") & Chr(176) & vbCrLf & _
        "Axis 1 = " & Format(dblAngles(1), "0.000") & Chr(176) & vbCrLf & _
        "Axis 2 = " & Format(dblAngles(2), "0.000") & Chr(176) & vbCrLf & _
        "Temperature = " & Format(dblTemperature, "0.00") & Chr(176) & "C", vbInformation, "VB X3 HelloWorld"
Else
   MsgBox "Error: " & GetX3ErrorDesc(lResult), vbExclamation, _
        "Failed to communicate with X3 on COM" & bytComPort
End If

X3_CloseComm bytComPort
Unload Me
```

4. To run the application, press the F5 function key.
   A message box similar to the following will be displayed.



3.2.2 C Example

1. Open Microsoft Visual C++ and create a new Win32 Console Application by clicking on File New menu items.
2. Select the Projects Tab.
3. Click Win32 Console Application and enter X3HelloWorld for the Project name. Click OK.
4. Click on the option that says,"A Hello, World! Application." and then click the Finish button and the OK button on the New Project Information dialog box.
5. Copy the X3User.h and X3User.lib files to the new project directory.
6. Click on Project Settings menu items. Click on the Link tab. Add X3User.lib to the end of Object/library modules field and then click OK.
7. Click on C/C++ tab. Select Precompiled Headers from the Category drop-down menu and then select Not using precompiled headers option.
8. Click on the FlieView tab within the Workspace frame. Expand the file folders and then double-click on X3HelloWorld.cpp node to open the file. Add the #include "..\Common\X3User.h" statement just above the main function.
9. Replace the code within the main function with a copy of the following code:

```
// C Hello World.cpp : Defines the entry point for the console application.
//
#include <conio.h>
#include "stdio.h"
#include "windows.h"
#include "..\Common\X3User.h"

int main(int argc, char* argv[])
{
        long lResult = 0;
    unsigned char ucCOMPort;
        unsigned char ucAddress = 0;
    double dblAngles[3] = {0,0,0};
        double dblTemperature = 0;

        printf("X3 HelloWorld\n");

    // Set the COM port number your device is attached to.
    ucCOMPort = 2;

    // Attempt to open the COM port.
    // Run the X3ConfigUtility to determine which COM port you're using.
    lResult = X3_InitComm(ucCOMPort, e115200);

    // Check if we're able to open the COM port.
    if(lResult == X3_SUCCESS)
        {
        // Get the angle of each axis.
        lResult = X3_GetAllAngles(ucCOMPort, ucAddress, &dblAngles[0], &dblAngles[1],
&dblAngles[2], &dblTemperature);
            printf("Axis 0 = %.3f\nAxis 1 = %.3f\nAxis 2 = %.3f\n", dblAngles[0],
dblAngles[1], dblAngles[2]);
    }

    if(lResult)
        {
        printf("Error: result = %d, Failed to communicate with X3 on COM%d\n", lResult,
ucCOMPort);
        }
        else
        {
        X3_CloseComm(ucCOMPort);
    }
```

```
        return 0;
}
```

10. Compile the code by pressing F7 function key and run the compiled code from a DOS window so that the output may be viewed.

4 Function Calls

4.1 X3_InitComm

Description:

The X3_InitComm function opens a specified COMM port for communication. The parameters are validated and if the COMM port is successfully opened it is configured to operate at 115200 baud rate.

Passed Parameters:

ucComPort

Specifies the COMM port number to be opened. Valid range is from 1 to 255. COMM port 255 is unique in that it may be used by X3_CloseComm to close all opened COMM ports.

eBaudRateCode

Specifies the baud rate code.

```
enum eBaudRateCode
{
   e115200 = 0,
   e57600 = 1,
   e38400 = 2,
   e19200 = 3,
   e9600 = 4
};
```

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_InitComm(unsigned char ucComPort, enum eBaudRateCode eCode);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
eBaudRateCode eCode = e115200;
iResult = X3_InitComm(ucComPort, eCode);
if (iResult != X3_SUCCESS) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_InitComm Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal eCode As
eBaudRateCode) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim lResult As Long
Dim eCode As eBaudRateCode
bytComPort = 1    ' Use COM1...
eCode = e115200
```

```
lResult = X3_InitComm(bytComPort, eCode)
If lResult <> X3_SUCCESS Then
   ' Handle error...
End If
```

4.2 X3_SetBaudRate

Description:

The X3_SetBaudRate function sets the X3 to communicate at a different baud rate. Once the X3 has accepted the new baud rate, the opened COM is then updated to the new baud. Refer to **4.1 X3_InitComm** for the list of the available baud rate codes.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

eCode

Specifies the baud rate code.

```
enum eBaudRateCode
{
   e115200 = 0,
   e57600 = 1,
   e38400 = 2,
   e19200 = 3,
   e9600 = 4
};
```

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetBaudRate(unsigned char ucComPort, unsigned char ucAddress, enum
eBaudRateCode eCode);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
eBaudRateCode eCode = e115200;
iResult = X3_SetBaudRate(ucComPort, ucAddress, eCode);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetBaudRate Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal eCode As eBaudRateCode) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim eCode As eBaudRateCode
eCode = e115200
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_SetBaudRate(bytComPort, bytAddress, eCode)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.3 X3_CloseComm

Description:

The X3_CloseComm function closes a specified open COMM port.

Passed Parameters:

ucComPort

Specifies the COMM port number to be closed. Valid range is from 1 to 255. A value of 255 will cause all opened COMM ports to close.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_CloseComm(unsigned char ucComPort);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
iResult = X3_CloseComm(ucComPort);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_CloseComm Lib "X3User.dll" (ByVal bytComPort As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim lResult As Long
bytComPort = 1    ' Close COM1...
lResult = X3_CloseComm(bytComPort)
If lResult <> X3_SUCCESS Then
   ' Handle error...
```

End If

4.4 X3_GetAllAngles

Description:

The X3_GetAllAngles function retrieves the angle (in degrees) for each axis.
See X3_SetAngleOutputRange to determine if the range of values returned are positive and negative values (-180.000 to < 180.000) or just positive values (0 to < 359.990).

Passed Parameters:

ucComPort
Specifies the COMM port number to be accessed. Valid range is from 1 to 255.
ucAddress
Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.
pdblAngle0
Pointer to a double value which will receive angle for axis 0.
pdblAngle1
Pointer to a double value which will receive angle for axis 1.
pdblAngle2
Pointer to a double value which will receive angle for axis 2.
pdblTemperature
Pointer to a double value which will receive the devices temperature in Celsius (-40.00 to 150.00).

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetAllAngles(unsigned char ucComPort, unsigned char ucAddress, double *
pdblAngle0, double * pdblAngle1, double * pdblAngle2, double * pdblTemperautre);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
double dblAngles[3] = {0,0,0};
double dblTemperature = 0;
iResult = X3_GetAllAngles(ucComPort, ucAddress, dblAngles[0], &dblAngles[1], &dblAngles[2],
&dblTemperature);
if (iResult != X3_SUCCESS) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetAllAngles Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByRef dblAngle0 As Double, ByRef dblAngle1 As Double, ByRef dblAngle2 As Double, ByRef
dblTemperature As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim sngAngle As Single
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetAngle(bytComPort, bytAddress, eXAxis, sngAngle)
If lResult <> X3_SUCCESS Then
   ' Handle error...
End If
```

4.5 X3_GetAngle

Description:

The X3_GetAngle function gets one angle (in degrees) for a specified axis.
See X3_SetAngleOutputRange to determine if the range of values returned are positive and negative values (-180.000 to < 180.000) or just positive values (0 to < 359.990).

Passed Parameters:

ucComPort
Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

eAxis
Specifies the axis of whose angle should be returned. Valid range is from 0 to 2.

pdblAngle
Pointer to a double value which will receive the angle for the specified axis.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetAngle(unsigned char ucComPort, unsigned char ucAddress, enum eAxis axis,
double * pdblAngle);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
double dblAngle = 0;
iResult = X3_GetAngle(ucComPort, ucAddress, eXAxis, &dblAngle);
if (iResult != X3_SUCCESS) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetAngle Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As
Byte, ByVal axis As eAxis, ByRef dblAngle As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim dblAngle As Double
bytComPort = 1
bytComPort = 0
dblAngle = 0
lResult = X3_GetAngle(bytComPort, bytAddress, eXAxis, dblAngle)
If lResult <> X3_SUCCESS Then
   ' Handle error...
End If
```

4.6 X3_SetAngle

Description:

The X3_SetAngle function sets the angle (in degrees) for a specified axis.
Updates the specified axis offset to achieve the set angle at the current orientation.
See X3_SetAngleOutputRange to determine if the range of values returned are positive and negative values (-180.000 to 179.999) or just positive values (0 to 359.999).

Passed Parameters:

ucComPort
: Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
: Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

eAxis
: Specifies the axis of whose angle should be returned. Valid range is from 0 to 2.

pdblAngle
: Specifies the angle at which specified axis should be set. Valid range is from -180.00 to 179.999 or 0 to 359.999 degrees depending on the selected angle output range.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetAngle(unsigned char ucComPort, unsigned char ucAddress, enum eAxis axis,
double dblAngle);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
double dblAngle = 45.5;
iResult = X3_SetAngle(ucComPort, ucAddress, eXAxis, dblAngle);
if (iResult != X3_SUCCESS) {
   // Handle error…
```

}

VB Declaration:

```
Public Declare Function X3_SetAngle Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal axis As eAxis, ByVal dblAngle As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim dblAngle As Double
bytComPort = 1
bytComPort = 0
dblAngle = 45.5
lResult = X3_SetAngle(bytComPort, bytAddress, eXAxis, dblAngle)
If lResult <> X3_SUCCESS Then
   ' Handle error...
End If
```

4.7 X3_GetAllAngleOffsets

Description:

The X3_GetAngleOffset function retrieves the angle offsets (in degrees) for all axes.
See X3_SetAngleOutputRange to determine if the range of values returned are positive and negative values (-180.000 to 179.999) or just positive values (0 to 359.999).

Passed Parameters:

ucComPort
: Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
: Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pdblAngle0
: Pointer to a double value which will receive angle offset for axis 0.

pdblAngle1
: Pointer to a double value which will receive angle offset for axis 1.

pdblAngle2
: Pointer to a double value which will receive angle offset for axis 2.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetAngleOffset(unsigned char ucComPort, unsigned char ucAddress, double * pdblOffset0, double * pdblOffset1, double * pdblOffset2);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
double dblOffsets[3] = {0,0,0};
iResult = X3_GetAllAngleOffsets(ucComPort, ucAddress, &dblOffsets[0] , &dblOffsets[1] , &dblOffsets[2]);
if (iResult != X3_SUCCESS) {    // Handle error…}
```

VB Declaration:

```
Public Declare Function X3_GetAllAngleOffsets Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByRef dblOffset0 As Double, ByRef dblOffset1 As Double, ByRef dblOffset2 As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim dblOffsets(0 To 2) As Double
bytComPort = 1    ' Access COM1...
lResult = X3_GetAllAngleOffsets(bytComPort, bytAddress, dblOffsets(0), dblOffsets(1), dblOffsets(2))
If lResult <> X3_SUCCESS Then
   ' Handle error...
```

End If

4.8 X3_SetAngleOffset

Description:

The X3_SetAngleOffset function sets the angle offset (in degrees) for a specified axis.
See X3_SetAngleOutputRange to determine if the range of values returned are positive and negative values (-180.000 to 179.999) or just positive values (0 to 359.999).

Passed Parameters:

ucComPort
Specifies the COMM port number to be accessed. Valid range is from 1 to 255.
ucAddress
Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.
eAxis
Specifies the axis of whose angle should be returned. Valid range is from 0 to 2.
dblOffset
Specifies the new angle offset.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetAngleOffset(unsigned char ucComPort, unsigned char ucAddress, enum eAxis axis, double dblOffset);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
double dblOffset = 90.0;
iResult = X3_SetAngleOffset(ucComPort, ucAddress, eXAxis, dblOffset);
if (iResult != X3_SUCCESS) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetAngleOffset Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal axis As eAxis, ByVal dblOffset As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim dblOffset As Double
bytComPort = 1
dblOffset = 90.0
lResult = X3_SetAngleOffset(bytComPort, bytAddress, eXAxis, dblOffset)
If lResult <> X3_SUCCESS Then
```

```
    ' Handle error...
End If
```

4.9 X3_GetAllData

Description:

The X3_GetAllData function retrieves the angle (in degrees) for each axis, device temperature, acceleration outputs for each axis, and the device's serial number.

See X3_SetAngeOutputRange to determine if the range of values returned are positive and negative values (-180.000 to 179.999) or just positive values (0 to 359.999).

See also: **X3_GetAllAngles** and **X3_GetDeviceInfo**.

Passed Parameters:

ucComPort
: Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
: Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pdblAngle0
: Pointer to a double value which will receive angle for axis 0.

pdblAngle1
: Pointer to a double value which will receive angle for axis 1.

pdblAngle2
: Pointer to a double value which will receive angle for axis 2.

pdblTemperature
: Pointer to a double value which will receive the devices temperature in Celsius (-40.00 to 150.00).

pdblAccel0
: Pointer to a double value which will receive acceleration for axis 0.

pdblAccel1
: Pointer to a double value which will receive acceleration for axis 1.

pdblAccel2
: Pointer to a double value which will receive acceleration for axis 2.

plSerialNo
: Pointer to an unsigned long value which will receive the device's serial number.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetAllData(unsigned char ucComPort, unsigned char ucAddress, double * pdblAngle0, double * pdblAngle1, double * pdblAngle2, double * pdblTemperature, double * pdblAccel0, double * pdblAccel1, double * pdblAccel2, unsigned long * pulSerialNo);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
double dblAngle[3] = {0,0,0};
```

```
double dblTemperature = 0.0;
double dblAccel[3] = {0,0,0};
unsigned long ulSerialNo = 0;

iResult = X3_ReadAllData(ucComPort, ucAddress, &dblAngle[0] , &dblAngle[1] , &dblAngle[2],
&dblTemperature, &dblAccel[0], &dblAccel[1], &dblAccel[2], &ulSerialNo);
if (iResult != X3_SUCCESS) { // Handle error…}
```

VB Declaration:

```
Public Declare Function X3_GetAllData Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByRef dblAngle0 As Double, ByRef dblAngle1 As Double, ByRef dblAngle2 As Double, ByRef
dblTemperature As Double, ByRef dblAccel0 As Double, ByRef dblAccel1 As Double, ByRef dblAccel2
As Double, ByRef ulSerialNo As Long) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim dblAngles(0 To 2) As Double
Dim dblTemperature As Double
Dim dblAccels(0 To 2) As Double
Dim ulSerialNo As Long
bytComPort = 1 ' Access COM1...

lResult = X3_GetAllData(bytComPort, bytAddress, dblAngles(0), dblAngles(1), dblAngles(2),
dlbTemperature, dblAccels(0), dblAccels(1), dblAccels(2), ulSerialNo)
If lResult <> X3_SUCCESS Then
' Handle error...
End If
```

4.10 X3_GetAllDirections

Description:

The X3_GetAllDirections function retrieves counting direction of each axis (normal or reversed).

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pcDirections0

Pointer to axis 0 counter direction byte. 0 = normal, 1 = reversed.

pcDirections1

Pointer to axis 1 counter direction byte. 0 = normal, 1 = reversed.

pcDirections2

Pointer to axis 2 counter direction byte. 0 = normal, 1 = reversed.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetAllDirections(unsigned char ucComPort, unsigned char ucAddress, char * pcDirection0, char * pcDirection1, char * pcDirection2);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
char cDirections[3] = {0,0,0};
iResult = X3_GetAllDirections(ucComPort, ucAddress, &cDirections[0], &cDirections[1], &cDirections[2]);
if (iResult != X3_SUCCESS) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetAllDirections Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByRef bytDirection0 As Byte, ByRef bytDirection1 As Byte, ByRef bytDirection2 As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim bytDirecitons(0 To 2) As Single
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetAllDirections(bytComPort, bytAddress, bytDirecitons(0), bytDirecitons(1), bytDirecitons(2))
If lResult <> X3_SUCCESS Then
   ' Handle error...
End If
```

4.11 X3_SetDirection

Description:

The X3_SetDirection function set the counting direction for a specified axis (normal or reversed).

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

eAxis

Specifies the axis of whose counting direction should be set. Valid range is from 0 to 2.

```
enum eAxis {
        eXAxis = 0,
        ePitch = 0,
        eYAxis = 1,
        eRoll = 1,
        eZAxis = 2,
        eYaw = 2
};
```

cDirection

The counter direction byte. 0 = normal, 1 = reversed.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetDirection(unsigned char ucComPort, unsigned char ucAddress, enum eAxis axis, char cDirection);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
char cDirection = 1;
iResult = X3_SetDirection(ucComPort, ucAddress, eXAxis, cDirection);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetDirection Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal axis As eAxis, ByVal bytDirection As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
```

```
Dim bytDirection As Byte
bytComPort = 1
bytAddress = 0
bytDirection = 1
lResult = X3_SetDirection (bytComPort, bytAddress, eXAxis, bytDirection)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.12 X3_GetDamping

Description:

Requests the damping time (in milliseconds).

Electronic damping is achieved by averaging multiple samples together to improve accuracy and reduce noise. The longer the damping, the more positions are averaged together (more smoothing), the slower the response time, and the more stable the reported position. The number of samples per reported position can be calculated by the damping time divided by 1.56 milliseconds.

Example: When the damping is set to 125 milliseconds, each reported position will be the average of the previous 70 samples, 250 milliseconds will be the average of the previous 140 samples, etcetera. Each average position is updated and reported every 1.56 milliseconds regardless of the damping value. To most closely match the damping of the X3 to US Digital's optical encoder type inclinometers such as the T5, T6 and A2T, specify 125 milliseconds for standard damping, and 250 milliseconds for double damping.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

puiTime

Pointer to integer that will hold the damping code value return. Valid range is from 2 to 5,000 with values 0 and 1 being reserved.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetDamping(unsigned char ucComPort, unsigned char ucAddress, unsigned int *
puiTime);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned int uiTime = 0;
iResult = X3_GetDamping(ucComPort, ucAddress, &uiTime);
```

```
if (iResult != 0) {    // Handle error… }
```

VB Declaration:

```
Public Declare Function X3_GetDamping Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByRef lTime As Long) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim lTime As Long
bytAddress = 0
bytComPort = 1    ' Access COM1...
lResult = X3_GetDamping(bytComPort, bytAddress, lTime)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.13 X3_SetDamping

Description:

Set the damping time (in milliseconds).
Refer to X3_GetDamping for an expanded description.

5000=heavy damping (average of previous 5 seconds), 2=least damping (no averaging, least accurate). Accuracy diminishes with less damping.

Passed Parameters:

ucComPort
: Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
: Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

uiTime
: The new damping code value. Valid range is from 2 to 5,000 with values 0 and 1 being reserved.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetDamping(unsigned char ucComPort, unsigned char ucAddress, unsigned int
uiTime);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned int uiTime = 500;
iResult = X3_SetDamping(ucComPort, ucAddress, uiTime);
```

```
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetDamping Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal lCode As Long) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim lTime As Long
bytComPort = 1    ' Access COM1...
bytAddress = 0
lTime = 500
lResult = X3_SetDamping(bytComPort, bytAddress, lCode)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.14 X3_GetAngleOutputRange

Description:

The X3_GetAngleOutputRange function retrieves a single byte that indicates the angle output range for all three axes.
0 = bidirectional (+/-) (-180.000° to 179.999°)
1 = unidirectional (+) (0.000° to 359.999°)

Passed Parameters:

ucComPort
Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pcAngleOutputRange
Pointer to a byte that will receive a value of 0 or 1. See description above.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetAngleOutputRange(unsigned char ucComPort, unsigned char ucAddress, char * pcAngleOutputRange);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
```

```
char cAngleOutputRange = 0;
iResult = X3_GetAngleOutputRange(ucComPort, ucAddress, &cAngleOutputRange);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetAngleOutputRange Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByRef cAngleOutputRange As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim bytAngleOutputRange As Byte
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetAngleOutputRange(bytComPort, bytAddress, bytAngleOutputRange)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.15 X3_SetAngleOutputRange

Description:

The X3_SetAngleOutputRange function is used to set a single byte that determines the angle output range for all three axes.
0 = bidirectional (+/-) (-180.000° to 179.999°)
1 = unidirectional (+) (0.000° to 359.999°

Passed Parameters:

ucComPort
: Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
: Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

cAngleOutputRange
: A byte value of 0 or 1. See description above.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetAngleOutputRange(unsigned char ucComPort, unsigned char ucAddress, char cAngleOutputRange);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
```

```
unsigned char ucAddress = 0;
char cAngleOutputRange = 1;
iResult = X3_SetAngleOutputRange(ucComPort, ucAddress, cAngleOutputRange);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetAngleOutputRange Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal cAngleOutputRange As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim bytAngleOutputRange As Byte
bytAngleOutputRange = 1
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_SetAngleOutputRange(bytComPort, bytAddress, bytAngleOutputRange)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.16 X3_GetDeviceInfo

Description:

The X3_GetDeviceInfo function gets the following factory settings: serial number, firmware revision, and product type.

Passed Parameters:

ucComPort
: Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress
: Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pulSerialNo
: Pointer to a long value that will hold the returned serial number.

piFirmware
: Pointer to a 6 byte array of characters that will hold the returned firmware version.

piProduct
: Pointer to a 6 byte array of characters that will hold the returned product type.

puiCalibrationStatus
: Pointer to an unsigned integer. (Reserved)

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetDeviceInfo(unsigned char ucComPort, unsigned char ucAddress, unsigned long * pulSerialNo, char * pcFirmware, char * pcProduct, unsigned short * puiCalibrationStatus);
```

Example C Usage:

```
int iResult = 0;
unsigned long ulSerialNo = 0;
unsigned char ucAddress = 0;
char cFirmware[6];
char cProduct[6];
unsigned short uiCalibrationStatus = 0;
iResult = X3_GetDeviceInfo(ucComPort, ucAddress, &ulSerialNo, &cFirmware, &cProduct, uiCalibrationStatus);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetDeviceInfo Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByRef ulSerialNo As Long, ByRef cFirmware As Byte, ByRef cProduct As Byte, ByRef uiCalibrationStatus As Integer) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte

Dim lResult As Long
Dim ulSeriaNo As Long
Dim cFirmware(0 To 5) As Byte
Dim cProduct(0 To 5) As Byte
Dim uiCalibrationStatus As Integer
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetDeviceInfo(bytComPort, bytAddress, ulSerialNo, cFirmware, cProduct, uiCalibrationStatus)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.17 X3_GetOutputConfig

Description:

The X3_GetOutputConfig function gets the output configuration settings which determine the behavior of the X3's six output pins. The six outputs are divided into two groups of three. Each output group (012 or 345) is assigned an output mode (0ff, quadrature, tilt, pwm-500 Hz, pwm-250 Hz, pwm-125 Hz, pwm-62.5 Hz, pwm-31.3 Hz, pwm-15.6 Hz, pwm-7.8 Hz, or pwm-3.9 Hz), axis, resolution, target angle in degrees, and target width in degrees.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

group

Pointer to an enumerated type value (eGroup) which determines one of two output groups to retrieve.

```
enum eGroup
{
        eGroup012 = 0,
        eGroup345 = 1
};
```

pMode

Pointer to a byte that will hold the output mode for the specified group.

```
enum eOutputMode
{
        eOff = 0,
        eQuadrature = 1,
        eTilt = 2,
        ePWM500 = 3,
        ePWM250 = 4,
        ePWM125 = 5,
        ePWM62_5 = 6,
        ePWM31_3 = 7,
        ePWM15_6 = 8,
        ePWM7_8 = 9,
        ePWM3_9 = 10
};
```

pAxis

Pointer to a byte the will hold the returned axis assignment.

puiResolution

Pointer to an unsigned integer that will hold the specified axis resolution.

pdblTargetAngle

Pointer to the target angle in degrees.

pdblTargetWidth

Pointer to the target width in degrees.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetOutputConfig(unsigned char ucComPort, unsigned char ucAddress, enum eGroup group, enum eOutputMode * pMode, enum eAxis * pAxis, unsigned short * puiResolution, double * pdblTargetAngle, double * pdblTargetWidth);
```

Example C Usage:

```
int iResult = 0;
unsigned short uiCalibrationStatus = 0;
unsigned char ucAddress = 0;
enum eGroup group = eGroup012;
enum eAxis axis = eXAxis;
unsigned short uiResolution = 0;
```

```
double dblTarget = 0.0;
double dblTargetWidth = 0.0;

iResult = X3_GetOutputConfig(ucComPort, ucAddress, &group, &axis, &uiResolution, &dblTarget,
&dblTargetWidth);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetOutputConfig Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal
bytAddress As Byte, ByVal group As eGroup, ByRef eOutputMode As eOutputMode, ByRef pAxis As
eAxis, ByRef puiResolution As Integer, ByRef pdblTargetAngle As Double, ByRef pdblTargetWidth As
Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim group As eGroup
Dim eOutputMode As eOutputMode
Dim axis As eAxis
Dim uiResolution As Integer
Dim dblTargetAngle As Double
Dim dblTargetWidth As Double

bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetOutputConfig(bytComPort, bytAddress, group, eOutputMode, axis, uiResolution,
dblTargetAngle, dblTargetWidth)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.18 X3_SetOutputConfig

Description:

The X3_SetOutputConfig function sets the output configuration settings which determine the behavior of the X3's six output pins. The six outputs are divided into two groups of three. Each output group (012 or 345) is assigned an output mode (0ff, quadrature, tilt), axis, resolution, target angle in degrees, and target width in degrees.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

group

An enumerated type value (eGroup) which determines one of two output groups to retrieve.
enum eGroup

```
        {
                eGroup012 = 0,
                eGroup345 = 1
        };
```

mode

A byte that will hold the output mode for the specified group.

```
        enum eOutputMode
        {
                eOff = 0,
                eQuadrature = 1,
                eTilt = 2
                ePWM500 = 3,            // 500Hz
                ePWM250 = 4,            // 250Hz
                ePWM125 = 5,            // 125Hz
                ePWM62_5 = 6,           // 62.5Hz
                ePWM31_3 = 7,           // 31.3Hz
                ePWM15_6 = 8,           // 15.6Hz
                ePWM7_8 = 9,            // 7.8Hz
                ePWM3_9 = 10            // 3.9Hz
        };
```

axis

A byte the will hold the returned axis assignment.

puiResolution

An unsigned integer that will hold the specified axis resolution.

dblTargetAngle

Double target angle in degrees.

dblTargetWidth

Double target width in degrees.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetOutputConfig(unsigned char ucComPort, unsigned char ucAddress, enum eGroup group, enum eOutputMode  mode, enum eAxis axis, unsigned short uiResolution, double dblTargetAngle, double dblTargetWidth);
```

Example C Usage:

```
int iResult = 0;
unsigned short uiCalibrationStatus = 0;
unsigned char ucAddress = 0;
enum eGroup group = eGroup012;
enum eAxis axis = eXAxis;
unsigned short uiResolution = 9000;
double dblTarget = 0.0;
double dblTargetWidth = 0.0;

iResult = X3_SetOutputConfig(ucComPort, ucAddress, group, axis, uiResolution, dblTarget, dblTargetWidth);
if (iResult != 0) {
```

```
    // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetOutputConfig Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal
bytAddress As Byte, ByVal group As eGroup, ByVal eOutputMode As eOutputMode, ByVal axis As eAxis,
ByVal uiResolution As Integer, ByVal dblTargetAngle As Double, ByVal dblTargetWidth As Double) As
Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim group As eGroup
Dim eOutputMode As eOutputMode
Dim axis As eAxis
Dim uiResolution As Integer
Dim dblTargetAngle As Double
Dim dblTargetWidth As Double

bytComPort = 1    ' Access COM1...
bytAddress = 0
group = eGroup012
axis = eXAxis
uiResolution = 9000
dblTargetAngle = 0
dblTargetWidth = 0
lResult = X3_SetOutputConfig(bytComPort, bytAddress, group, eOutputMode, axis, uiResolution,
dblTargetAngle, dblTargetWidth)
If lResult <> 0 Then
    ' Handle error...
End If
```

4.19 X3_GetOutputUpdateRate

Description:

The X3_GetOutputUpdateRate reads the output update rate setting. It can be changed for users that need a slower output update rate for the manual, tilt and especially the quadrature output modes. The majority of users should leave the output update rate to its default (fastest) setting.

Note that this update rate affects all output signals only on the 8-pin connector. It does not affect the angle measurement rate or the serial interface data rate.

When the update rate is set to 0x01, the minimum quadrature state time is approximately 35µs and the average quadrature state time 53µs (19kHz). Each increment in the update rate value adds 2.6µs.

When the update rate is set to 0xFF or 0x00, the minimum quadrature state time is approximately 700µs and the average quadrature state time 1000µs (1kHz).

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pucCode

Pointer to a byte that will receive the code value.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetOutputUpdateRate(unsigned char ucComPort, unsigned char ucAddress, char * pucCode);
```

Example C Usage:

```
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned char ucCode;
iResult = X3_GetOutputUpdateRate(ucComPort, ucAddress, ucCode);
if (iResult != 0) {
// Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetOutputUpdateRate Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByRef bytCode As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim bytOutput As Byte
bytComPort = 1 ' Access COM1...
bytAddress = 0
lResult = X3_SetOutputUpdateRate(bytComPort, bytAddress, bytCode)
If lResult <> 0 Then
        ' Handle error…
End If
```

4.20 X3_SetOutputUpdateRate

Description:

The X3_SetOutputUpdateRate function allows the output update rate to be changed for users that need a slower output update rate for the manual, tilt and especially the quadrature output modes. The majority of users should leave the output update rate to its default (fastest) setting.

Note that this update rate affects all output signals only on the 8-pin connector. It does not affect the angle measurement rate or the serial interface data rate.

When the update rate is set to 0x01, the minimum quadrature state time is approximately 35µs and the average quadrature state time 53µs (19kHz). Each increment in the update rate value adds 2.6µs.

When the update rate is set to 0xFF or 0x00, the minimum quadrature state time is approximately 700µs and the average quadrature state time 1000µs (1kHz).

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

ucCode

A byte value. Valid range is from 0 to 255. Code 1 = 35µs ... 255 = 700µs.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetOutputUpdateRate(unsigned char ucComPort, unsigned char ucAddress, char ucCode);
```

Example C Usage:

```
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned char ucCode = 255;     // Set the delay to 700µs.
iResult = X3_SetOutputUpdateRate(ucComPort, ucAddress, ucCode);
if (iResult != 0) {
// Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetOutputUpdateRate Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal bytCode As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim bytOutput As Byte
```

```
bytCode = 255   ‘ Set the delay to 700µs.
bytComPort = 1 ' Access COM1...
bytAddress = 0
lResult = X3_SetOutputUpdateRate(bytComPort, bytAddress, bytCode)
If lResult <> 0 Then
        ' Handle error
End If
```

4.21 X3_GetOutput

Description:

The X3_GetOutput function reads the data currently output on the 6 bit output port. This function can be issued even when in Tilt or Quadrature modes.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pucOutput

A byte value reporting the state of each output port bit. A value of 1 means the output bit is high.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_GetOutput(unsigned char ucComPort, unsigned char ucAddress, char *
pucOutput);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned char ucOutput = 0;
iResult = X3_GetOutput(ucComPort, ucAddress, &ucOutput);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetOutput Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByRef pbytOutput As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim bytOutput As Byte
bytOutput = 0
```

```
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetOutput(bytComPort, bytAddress, bytOutput)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.22 X3_SetOutput

Description:

The X3_SetOutput function sets the state of the 6 bit output port bits. This function is valid only the output mode is set to Manual. Refer to X3_GetOutputConfig or X3_SetOutputConfig.

.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pucOutput

A byte value used to set the output port bit states. A value of 1 sets the output high.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall X3_SetOutput(unsigned char ucComPort, unsigned char ucAddress, char ucOutput);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned char ucOutput = 0x3F; // Set all six output bits high.
iResult = X3_SetOutput(ucComPort, ucAddress, ucOutput);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetOutput Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal pbytOutput As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim bytOutput As Byte
bytOutput = &h3f  ' Set all six output bits high...
```

```
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetOutput(bytComPort, bytAddress, bytOutput)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.23 X3_GetStartupDelay

Description:

The X3_GetStartupDelay function reads the startup delay. The startup delay in seconds is approximately (startup delay)/640. The factory default of 320 gives a startup delay of 0.5 seconds.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

pulDelay

A value reporting the startup delay.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
int __stdcall X3_GetStartupDelay(unsigned char ucComPort, unsigned char ucAddress, unsigned long *
pulDelay);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned long ulDelay = 0;
iResult = X3_GetStartupDelay(ucComPort, ucAddress, &ulDelay);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_GetStartupDelay Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal
bytAddress As Byte, ByRef ulDelay As Long) As Long  ' Get the startup delay in 1/640ths of a second.
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim ulDelay As Long
bytOutput = 0
```

```
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_GetStartupDelay (bytComPort, bytAddress, ulDelay)
If lResult <> 0 Then
   ' Handle error...
End If
```

4.24 X3_SetStartupDelay

Description:

This function allows the "startup delay" to be adjusted. On powerup in quadrature output mode, the index pulse is set high for a "startup delay" time interval before the A/B quadrature outputs become active. The index pulse is normally used to clear an external quadrature counter so it can maintain an absolute position count (See the **X3 Datasheet** for more details). Some external quadrature counters may take a few seconds to power up – the startup delay setting is used to ensure that the X3 maintains the initial index pulse for enough time to reset an external quadrature counter.

In manual/tilt/PWM modes, the outputs are deasserted until the startup delay expires.

The startup delay in seconds is approximately (startup delay)/640. The factory default of 320 gives a startup delay of 0.5 seconds.

Passed Parameters:

ucComPort

Specifies the COMM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 0 to 127. Use address 0 for X3 devices; all other values are reserved.

ulDelay

A value used to set the startup delay. Valid range is from 1 to 65534.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
int __stdcall X3_SetStartupDelay(unsigned char ucComPort, unsigned char ucAddress, unsigned long ulDelay);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned long ulDelay = 640; // Set startup delay to one second
iResult = X3_SetStartupDelay (ucComPort, ucAddress, ulDelay);
if (iResult != 0) {
   // Handle error…
}
```

VB Declaration:

```
Public Declare Function X3_SetStartupDelay Lib "X3User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal ulDelay As Long) As Long  ' Set the startup delay in 1/640ths of a second.
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim ulDelay As Long
ulDelay = 640  ' Set startup delay to one second
bytComPort = 1    ' Access COM1...
bytAddress = 0
lResult = X3_SetStartupDelay (bytComPort, bytAddress, ulDelay)
If lResult <> 0 Then
   ' Handle error...
End If
```

5 Constants

```
// X3 Constants
#define X3_MAX_COMM_PORTS          255     // Maximum number of COM ports.
#define X3_MAX_BAUD_RATE           115200  // Maximum baud rate
#define X3_MIN_DAMPING_CODE        2       // Minimum damping code.
#define X3_MAX_DAMPING_CODE        5000    // Maximum damping code.
```

6 Error Codes

```
// X3 Driver return codes
#define X3_SUCCESS                          0x00  // Success.
#define X3_INVALID_COMMAND                  0x01  // Invalid command
#define X3_COMMAND_LOCKED                   0x02  // Command locked
#define X3_INVALID_PARAMETER                0x03  // Invalid Parameter
#define X3_CHECKSUM_ERROR_TX                0x04  // Checksum error sent to the X3.
#define X3_CHECKSUM_ERROR_RX                0x05  // Checksum error sent from the X3.
#define X3_COMMAND_FAILED                   0x06  // Command failed.
#define X3_INVALID_COMPORT                  0x07  // Invalid com port specified.
#define X3_FAIL_TO_OPEN_COM_PORT            0x08  // Failed to com port.
#define X3_COMM_ERROR                       0x09  // Generic com error.
#define X3_COM_PORT_NOT_OPEN                0x0a  // Com port not open.
#define X3_FAILED_TO_SET_COM_PORT_TIMEOUT   0x0b  // Failed to set com port timeout.
#define X3_FAILED_TO_FLUSH_COM_PORT         0x0c  // Failed to purge the com port.
#define X3_FAILED_TO_CLEAR_COM_ERROR        0x0d  // Failed to clear com port error.
#define X3_INVALID_AXIS                     0x0e  // Invalid axis specified.
#define X3_COMM_TIMEOUT                     0x0f  // Failed to receive expected data.
```

7 Enumerations

```
enum eAxis
{
        eXAxis = 0,
        ePitch = 0,
        eYAxis = 1,
        eRoll = 1,
        eZAxis = 2,
        eYaw = 2
};

enum eGroup
{
        eGroup012 = 0,
        eGroup345 = 1
};

enum eOutputMode
{
        eOff = 0,
        eQuadrature = 1,
        eTilt = 2,
        ePWM500 = 3,        // 500Hz
        ePWM250 = 4,        // 250Hz
        ePWM125 = 5,        // 125Hz
        ePWM62_5 = 6,       // 62.5Hz
        ePWM31_3 = 7,       // 31.3Hz
        ePWM15_6 = 8,       // 15.6Hz
        ePWM7_8 = 9,        // 7.8Hz
        ePWM3_9 = 10        // 3.9Hz
};

enum eTemperatureRange
{
        eLow = 0,
        eMid = 1,
        eHigh = 2
};

enum eBaudRateCode
{
   e115200 = 0,
   e57600 = 1,
   e38400 = 2,
   e19200 = 3,
   e9600 = 4
};
```