Delcom DLL Manual

Delcom Products Inc.
45 Backus Ave
Danbury CT 06810
(914) 934-5170
(914) 934-5171 Fax
www.delcomproducts.com

Contents

1.0 Introduction

This document describes the function of the Delcom USB DLL. The Delcom USB DLL, herein referred to as the DLL, allows a user's application to interface with the Delcom USB peripheral products. The goal of the DLL is to provide the user with a simple method of communicating to the USB devices.

2.0 Requirements

2.1 Supports

The DLL supports Windows 98, ME, 2000 & XP.

2.2 Files

To use the DLL you will need the following files.

DelcomDLL.dll – This is the actual DLL file and must located in the directory your application is running from or the windows default DLL directory. The windows default DLL directory is usually C:\WINDOWS\SYSTEM.

DelcomDLL.lib – This is the library file, which your C linker will use to probably link your application to the DLL. This file must be located in the directory of your source code. This file is only required for C applications.

DelcomDLL.h – This is the C header file, which has the function prototyping in and useful constants that will be used when communicating with the DLL. This file must be located in the directory of your C source code. This file is only required for C applications.

DelcomDLL.bas – This is the VB header file, which has the function prototyping in and useful constants that will be used when communicating with the DLL. This file must be located in the directory of your VB project and added to the project as a module. This file is only required for VB applications.

3.0 Using the DLL in your applications.

3.1 C Applications

There are two ways to use the DLL in C applications, they are Link Implicitly and Link Explicitly.

Implicitly link is the easiest to use. In Implicitly linking the DLL is loaded when your application loads. To implicitly link to the DLL add the DLL library file to your project and include the DLL header file in your source code. Then simply call the DLL functions. All example C code in this manual uses the implicit method.

Explicitly linking allows your application to load the DLL at a defer time. For your application to Explicitly link to the DLL you will have to load the DLL with LoadLibrary() function. LoadLibrary loads the DLL, if not already loaded, and returns a handle to the DLL. Then using the DLL handle and the DLL function name of interest, call GetProcAddress() to obtain the address of the DLL function. GetProcAddress returns the function address, which you can use to call the DLL functions. When your done using the DLL you should call FreeLibrary() to release the DLL resources.

3.2 VB Applications

When using the DLL in a VB application you must declare the DLL functions. To declare DLL procedures use the Declare statement to prototype each function.

Declare Function *publicname* **Lib** "*libname*" **[Alias** "*alias*"**] [([[ByVal]** *variable* **[As** *type***] [,[ByVal]** *variable* **[As** *type***]]...])] As Type**

Example

```
Declare Function DelcomGetDeviceCount Lib “DelcomDLL.dll” Alias
        “DelcomGetDeviceCount” (ByVal Count As Long) As Long
```

All the DLL functions have been declared in the DelcomDLL.bas file. Simply add this file to your VB project and then call the DLL function from your code as you would any other function.

4.0 Quick Start Example

This sections illustrates a simply example using the DLL. In both examples the code simply scans for the devices, open's the first device found if any, places the LED on P1.1 into flash mode and then closes the connection.

4.1 C Example

Source Code:

```
// DelcomDLLExampleC
#include "stdafx.h"
#include "DelcomDLL.h"

int main(int argc, char* argv[])
{
        DeviceNameStruct Names[10];       // array to hold the device names found

        printf("Delcom DLL Test Example C\n");

        int found = DelcomScanDevices( USBDELVI, Names, 10) ;
        if(found )    {
             for(int i=0; i<found; i++)
                  printf("%s SN:%u\n", (char*)&Names[i],
                       DelcomReadDeviceSerialNum((char*)&Names[i], NULL ));
        }
        else
             printf("No USBDELVI devices found!\n");

        if(found) {          // open the device
                  HANDLE hUsb = DelcomOpenDevice((char*)&Names[0],0);                    DelcomLEDControl( hUsb,
REDLED, FLASH );                                    DelcomCloseDevice(hUsb);          // close the device
        }
        return(0);
}
```

Output:

```
Delcom DLL Test Example C
\\.\0000000000000029#{59bd73a6-822e-4684-9530-0754fe897113} SN:521
```

4.2 VB Example

Source Code:

```
Sub Form_Load()

   Dim DeviceName As String * MAXDEVICENAMELEN  ' Must predeclare size
   Dim Result As Long
   Dim DeviceHandle As Long
   Dim Packet As PacketStructure
   Dim DataExt As DataExtStructure


   Text1.Text = "Delcom DLL VB Sample" & Chr(13) & Chr(10)

   Result = DelcomGetNthDevice(USBIODS, 0, DeviceName)

   If (Result) Then
      Text1.Text = Text1.Text & "Device Found" & Chr(13) + Chr(10)
      Text1.Text = Text1.Text & DeviceName
      Text1.Text = Text1.Text & Chr(13) & Chr(10)

      DeviceHandle = DelcomOpenDevice(DeviceName, 0)
      Text1.Text = Text1.Text + "Flash Green LED" + Chr(13) + Chr(10)
      Result = DelcomLEDControl(DeviceHandle, GREENLED, LEDFLASH)
      Result = DelcomCloseDevice(DeviceHandle)

   Else
      Text1.Text = Text1.Text + "Device Not  Found" + Chr(13) + Chr(10)
   End If

End Sub
```

5.0 Functions

The DLL supports all of Delcom's USB products that use the Delcom USB driver. The DLL functions are divided into three groups, they are Common, Visual Indicator and USB I/O functions. Common functions are use with all of Delcom's USB products, while Visual Indicator (USBDELVI) and USB I/O (USBIODS) functions are device specific.

Prototyping for C and VB are shown under each function. All functions have _stdcall calling conventations.

NOTE. All data parameters must be pre-allocated in the users code.

5.1 Common Functions

DelcomGetDLLHandle - This function returns HANDLE to the DLL module.

```
C: HANDLE DelcomGetDLLHandle(void);
VB:  Public Declare Function DelcomGetDLLHandle Lib "DelcomDLL.dll" _
        () As Long
```

DelcomGetDLLVersion - This function returns the DLL version, see section 6.0 for release notes.

```
C: double DelcomGetDLLVersion(void);
VB: Public Declare Function DelcomGetDLLVersion Lib "DelcomDLL.dll" _
        () As Double
```

DelcomGetDLLDate – This function returns the DLL release date. The returns 1 on error. The passing char string pointer must be pre-allocated and must be at least 16 bytes long.

```
C: DWORD DelcomGetDLLDate(LPSTR DllDate);
VB: Public Declare Function DelcomGetDLLDate Lib "DelcomDLL.dll" _
          (ByVal datestring As String) As Long
```

DelcomVerboseControl – This function enables or disables the DLL warning and/or error messages. The warning/error messages are off by default. The function takes to two parameters an integer Mode and a character string called Header. A Mode value of nonzero turns this function on and a zero value will turn this function off. The second parameter is the Header parameter and is optional. The Header string is printed out in the title of the warning/error message. This can be useful in identifying which application caused the warning/error when multiple applications are using the DLL. If the optional Header parameter is not used set the Header value to NULL or zero. Users should enable this feature when developing/debugging there code.

```
C: DWORD DelcomVerboseControl( DWORD Mode,  LPSTR Header );
VB: Public Declare Function DelcomVerboseControl Lib "DelcomDLL.dll" _
        (ByVal Mode As Long, ByVal caption As String) As Long
```

DelcomGetNthDevice – This function returns a string of the Nth device found. The first parameter tells the DLL what type of device to search for. . A value of 1 will search for all the USBIODS devices, a value of 2 will search for all of the USBDELVI device and a value of 3 will search for all the USBNDSPY devices. The functions returns a true if the device was found otherwise false.
The second parameter is a zero based index of the device you want to open. Setting this parameter to zero will try to open the matching device on the device list. A value of one will try and open the second matching device on the list and so on. The third parameter is a pointer to a string of characters of which must be pre-allocated and must be at least 512 bytes long. On success the device name is return to the user to use in the DelcomOpenDevice() command.

C: DWORD DelcomGetNthDevice(DWORD Type, DWORD Nth, LPSTR Name);
VB: Public Declare Function DelcomGetNthDevice Lib "DelcomDLL.dll" _
(ByVal ProductType As Long, ByVal NthDevice As Long, ByVal DeviceName As String) As Long

DelcomScanDevices – This function returns an array of device names as character strings and is only available for C applications. The first parameter tells the DLL what type of device to search for. A value of zero will search for all the USBIODS devices and a value of 1 will search for all the USBDELVI devices. The functions returns an integer representing the number of matching devices found. If no devices are found zero is returned. The second parameter is a pointer to an array of character strings. The functions returns the device names found in this array. This array must be declared in the calling applications. The character strings must be 512 bytes long. The third parameter tells the function how big the array is. This value must be at least 1. The function will only fill the array with up to this value. You can arbitrarily set the size of the array and this value or you can use the DelcomGetDeviceCount() function to find the number of devices present before declaring the array size and this value. The application will use the device names returned to open communications with the USB device.

C: DWORD DelcomScanDevices(DWORD type, DeviceNameStruct[], DWORD Max);

DelcomGetDeviceCount – This function returns the number of USB devices that match the product type given. The function returns only the number of currently available USB matching devices available at the time the function is called. The integer parameter tells the DLL what type of devices to count. A value of zero will search for all the USBIODS devices and a value of 1 will search for all the USBDELVI devices. The functions returns zero if no matching device are found. This function can be used to determine how big the character string array passed to DelcomScanDevices() needs to be.

C: DWORD DelcomGetDeviceCount(DWORD Type);
VB: Public Declare Function DelcomGetDeviceCount Lib "DelcomDLL.dll" _
(ByVal ProductType As Long) As Long

DelcomOpenDevice – This function opens the USB device and returns a handle to the device. The first parameter passed is a character string containing a valid USB device name returned from the DelcomScanDevices() function. On success the function will return a handle to the USB device. This handle is then used for all communications to the USB device. The second parameter is for future use and should be sent to zero for future compatibility. If the function fails a zero value is returned.

C: HANDLE DelcomOpenDevice(LPSTR Name , DWORD Mode);
VB: Public Declare Function DelcomOpenDevice Lib "DelcomDLL.dll" _
(ByVal DeviceName As String, ByVal Mode As Long) As Long

DelcomCloseDevice – This function closes the USB device communications and release all resource used by the USB device. USB devices should be closed after the application is finished communicating with the device. If the USB device is not closed when the user's application terminates, the device will not be able to be open again. To recover from this condition reset the USB device.

C: DWORD DelcomCloseDevice(HANDLE hUsb);
VB: Public Declare Function DelcomCloseDevice Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long) As Long

DelcomReadDeviceVersion – This function returns the firmware version of the USB device. The USB device must have been already opened. A valid USB handle form DelcomOpenDevice() must be passed to the function. If unsuccessful zero is returned.

C: DWORD DelcomReadDeviceVersion(HANDLE hUsb);

```
Public Declare Function DelcomReadDeviceVersion Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long) As Long
```

DelcomReadDeviceSerialNum – This function returns the serial number for a USB device. The function has two modes. It can read the serial number of the USB device given the handle to the device or it can use the device name of the USB device. The first parameter is a character string containing a device name and the second parameter is a USB device handle. The parameter not used should be set to NULL or zero. On success the device returns the serial number of the USB device.

```
C: DWORD DelcomReadDeviceSerialNum(LPSTR Name, HANDLE hUsb);
VB: Public Declare Function DelcomReadDeviceSerialNum Lib"DelcomDLL.dll"_
        (ByVal DeviceName As String, ByVal DeviceHandle As Long) As Long
```

DelcomSendPacket – This function sends and receives a data packet from the USB device. The USB device must be already opened. The first parameter passed is the handle to the USB device. The second parameter is the data packet to send to the USB device and the last parameter is the data packet to receive from the USB device. The receive data packet is only required when the commands is of a read type. When not passing a receive packet set the last parameter to NULL or zero. On success the functions return zero and nonzero on error. The send packet can be 8 to 16 bytes long. The receive packet is always 8 bytes long. See the USB device manual for more information on the data packet format and structure.

```
C: DWORD DelcomSendPacket( HANDLE, pPacketStruct, pPacketStruct);
VB: Public Declare Function DelcomSendPacket Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long, ByRef PacketOut As PacketStructure, ByRef PacketIn As PacketStructure) As Long
```

DelcomSendHIDPacket – This function sends and receives G2 data packet from the USB device. The USB device must be already opened. The first parameter passed is the handle to the USB device. The second parameter is the data packet to send to the USB device. The any returned data is also return in the second parameter. The last parameter is the size of the data packet. Valid length are 8 or 16 bytes.
On success the functions return zero and nonzero on error. The send packet can be 8 to 16 bytes long. The receive packet is always 8 bytes long. See the USB device manual for more information on the data packet format and structure. This command should be used instead of the older DelcomSendPacket() command for all generation 2 (G2) and newer devcies.

```
C: DWORD DelcomSendHIDPacket( HANDLE, pHIDPacketStruct, Length);
VB: Public Declare Function DelcomSendHIDPacket Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long, ByRef PacketOut As HIDPacketStructure,  Length As Long ) As Long
```

5.2 Visual Indicator Functions

These commands refer to the Visual Indicator product, USBDELVI. Many of these functions are also supported in the USBIODS device, see the USB device manual for more information.

DelcomLEDControl – This function controls the state of the LED's. The first parameter passed is a valid USB handle. The second parameter is the LED color to control. 0=Green, 1=Red, 2=Blue, 3=Yellow and 4=Orange. The last parameter specifies the LED operating mode. They are 0=OFF, 1=ON and 2=FLASH. The functions returns zero on success and nonzero on error.

```
C: DWORD DelcomLEDControl( HANDLE hUsb, DWORD Color, DWORD Mode  );
VB: Public Declare Function DelcomLEDControl Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long, ByVal Color As Long, ByVal Mode As Long) As Long
```

DelcomLoadLedFreqDuty – This function loads the flash rate of the LED. The first parameter is the USB device handle. The second parameter is the LED color, 0=Green, 1=Red, 2=Blue, 3=Yellow and 4=Orange. The third parameter is a byte value representing the on time of the LED and the last parameter is a byte value representing the off time of the LED. The on and off parameters are in units of the prescalar see DelcomLoadPreScalar function. The functions returns zero on success and nonzero on error.

```
C: DWORD DelcomLoadLedFreqDuty(HANDLE hUsb, BYTE Color, BYTE Low, BYTE High );
VB: Public Declare Function DelcomLoadLedFreqDuty Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long, ByVal Color As Byte, ByVal Low As Byte, ByVal High As Byte) As Long
```

DelcomLoadPreScalar – This function loads the LED flash prescalar value. The first parameter is the USB device handle. The second parameter is the prescalar value. The units of the prescalar are mseconds. The default value is 10 mseconds. This prescalar value is global to all the LED flash rates. The functions returns zero on success and nonzero on error.

```
C: DWORD DelcomLoadPreScalar(HANDLE hUsb, BYTE PreScalar);
VB: Public Declare Function DelcomLoadPreScalar Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long, ByVal PreScalar As Byte) As Long
```

DelcomSyncLeds – This function synchronies all the LED flash rates. This function can be used with the DelcomLoadInitialPhaseDelay function to offset flash rates. The only parameter is the USB device handle. The functions returns zero on success and nonzero on error.

```
C: DWORD DelcomSyncLeds(HANDLE hUsb);
VB: Public Declare Function DelcomSyncLeds Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long) As Long
```

DelcomLoadInitialPhaseDelay – This function loads the LED flash rate offset. The first parameter is the USB device handle. The second parameter is the LED color, 0=Green, 1=Red, 2=Blue, 3=Yellow and 4=Orange. The last parameter is the offset value. The offset value is a byte with units of the prescalar. The functions returns zero on success and nonzero on error.

```
C: DWORD DelcomLoadInitialPhaseDelay(HANDLE hUsb, BYTE Color, BYTE Delay);
VB Public Declare Function DelcomLoadInitialPhaseDelay Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long, ByVal Color As Byte, ByVal Delay As Byte) As Long
```

DelcomGetButtonStatus – This function returns the current state of the device button. Returns one when the button is pressed and zero if not pressed.

```
C: DWORD DelcomGetButtonStatus(HANDLE hUsb );
VB: Public Declare Function DelcomGetButtonStatus Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long) As Long
```

DelcomEnableAutoConfirm – This function enables or disables the auto confirm mode. The first parameter is the handle to the device. The second parameter is the mode. 0=OFF and 1=ON. When the auto confirm mode is enabled the buzzer will sound when the button is pressed. The functions returns zero on success and nonzero on error.

```
C: DWORD DelcomEnableAutoConfirm(HANDLE hUsb, DWORD Mode);
VB: Public Declare Function DelcomEnableAutoConfirm Lib "DelcomDLL.dll" _
        (ByVal DeviceHandle As Long, ByVal Mode As Long) As Long
```

DelcomEnableAutoClear – This function enables or disables the auto clear mode. The first parameter is the handle to the device. The second parameter is the mode. 0=OFF and 1=ON. When the auto clear mode is enabled all LED's will be turned off when button is pressed. The functions returns zero on success and nonzero on error.

C: DWORD DelcomEnableAutoClear(HANDLE hUsb, DWORD Mode);
VB: Public Declare Function DelcomEnableAutoClear Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal Mode As Long) As Long

DelcomBuzzer – This function sets up the buzzer routine. The first parameter is the USB device handle. The second parameter is the mode, a value of zero will turn off the buzzer and a value of 1 will turn on the buzzer with the following parameters. The third parameter is the frequency of the buzzer. It is a byte and has units of 256us. The fourth parameter is the repeat value. The repeat value is a byte, a value of 0 zero places the buzzer in a continuous mode and a non zero value places the buzzer is a repeat mode. In repeat mode the buzzer will repeat for the number of times given in the repeat value. The fifth value is the buzzer on time and the last parameter is the buzzer off time. Both parameters are of type bytes. The functions returns zero on success and nonzero on error.

C: DWORD DelcomBuzzer(HANDLE hUsb, BYTE Mode, BYTE Freq, BYTE Repeat, BYTE OnTime, BYTE OffTime);

Public Declare Function DelcomBuzzer Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal Mode As Byte, ByVal Freq As Byte, ByVal Repeat As Byte, ByVal OnTime As Byte, ByVal OffTime As Byte) As Long

DelcomLEDPower – This function sets up LED Power. The first parameter is the USB device handle. The second parameter is the LED color, 0=Green, 1=Red, 2=Blue/Yellow. The third parameter is the percent of power. Valid range is from 0-100, where 0 is off and 100 is full power. Default power level is 80.

C: DWORD DelcomLEDPower(HANDLE hUsb, DWORD Color, DWORD Power);

VB: Private Declare Function DelcomLEDPower Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, Color As Long, Power As Long) As Long

5.3 USB I/O Functions

These commands refer to the USB IO chips. The first parameter is a handle to an opened USB device. Functions returns zero on success and nonzero on error unless otherwise stated.

DelcomWritePorts – This function writes the second data parameter byte to port 0 and the third data byte parameter to Port 1.

```
C: DWORD DelcomWritePorts(HANDLE hUsb, BYTE Port0, BYTE Port1 );
VB: Public Declare Function DelcomWritePorts Lib "DelcomDLL.dll" _
    (ByVal DeviceHandle As Long, ByVal Port0 As Byte, ByVal Port1 As Byte) As Long
```

DelcomReadPorts – This function reads the current port values and places port0 value in the second data parameter and port 1 in the third data parameters. Note that the byte data parameters are passed as a pointer and must be allocated by the user.

```
C: DWORD DelcomReadPorts(HANDLE hUsb, BYTE* Port0, BYTE* Port1 );
VB: Public Declare Function DelcomReadPorts Lib "DelcomDLL.dll" _
    (ByVal DeviceHandle As Long, ByRef Port0 As Byte, ByRef Port1 As Byte) As Long
```

DelcomWritePin – This function sets or resets an indivisual port pin. Al other port pins are left unchanged. The first parameter is the handle to the USB device. The second parameter is the port number (range 0-2). The third parameter is the pin number (range 0-7). The fourth parameter is the value (range 0-1). A value 0 resets the pin to the low/zero state and value of 1 sets the pin to a high/one state.

```
DWORD DelcomWritePin(HANDLE hUsb, BYTE Port, BYTE Pin, BYTE Value );
```

DelcomSetupPorts – This function setup the port pins modes. The first parameter is the handle to the USB device. The second parameter is the port number to setup (range 0-1). The third parameter is the mode 0 value and the fourth parameter is the mode1 value.

GPIO Mode table

Mode	Mode 1 Value	Mode 0 Value	Port type when data out is low	Port type when data out is high
A	0	0	Hi-Z / CMOS	Hi-Z / TTL
B	0	1	Medium (8mA) Sink / CMOS	High (30mA) Drive / CMOS
C – Default *Boot up State*	1	0	Low (2mA) Sink / CMOS	Pull up (14K) / CMOS
D	1	1	High (50mA) Sink / CMOS	High (30mA) Drive / CMOS

Maximum cumulative source drive current for all GPIO is 30mA.
Maximum cumulative sink drive current for all GPIO is 70mA.
See http://www.delcomproducts.com/downloads/cy7c637xx-B.pdf for more GPIO details

```
DWORD DelcomSetupPort(HANDLE hUsb, BYTE Port, BYTE Mode0, BYTE Mode1 );
```

DelcomWrite64Bit – This function writes 8 bytes of data, passed in the second data parameter to the 8 write latches. This command requires external hardware, see website for schematic. The data size is fixed to 8 bytes and the LSB is written to the write address zero.

C: DWORD DelcomWrite64Bit(HANDLE hUsb, LPSTR DataExt);
VB: Public Declare Function DelcomWrite64Bit Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByRef DataExt As DataExtStructure) As Long

DelcomRead64 – This function reads the 8 bytes from the 8 input latches. This command requires external hardware, see website for schematic. The data size is fixed at 8 bytes and the LSB is read from read address zero.

C: DWORD DelcomRead64Bit(HANDLE hUsb, LPSTR DataExt);
VB: Public Declare Function DelcomRead64Bit Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByRef DataExt As DataExtStructure) As Long

DelcomWriteI2C – This function writes to the I2C port. The second data parameter is the address/command byte. This byte is sent first over the I2C port. The third data parameter is the length of the optional data to be sent via the I2C port. If there is no data segment then set the length to zero. A maximum of 8 data bytes following the address/command byte may be sent. The optional data is placed in the fourth parameter. The LSB is first byte sent out after the address/command byte.

C: DWORD DelcomWriteI2C(HANDLE hUsb, BYTE CmdAdd, BYTE Length, LPSTR DataExt);
VB: Public Declare Function DelcomWriteI2C Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal CmdAdd As Byte, ByVal Length As Byte, ByRef DataExt As DataExtStructure) As Long

DelcomReadI2C – This function reads N bytes from the I2C port. The second data parameter is the address/command byte and is the first byte written to the I2C port. The bytes to be read follow this address/command byte. The third data parameter is the length of the bytes to read. Valid length range is 1 to 8. The data read is returned in the last parameter. Note, in most EEPROM type devices, reading the device with this command cause the internal address of the device to auto increment proportionally. See command below.

C: DWORD DelcomReadI2C(HANDLE hUsb, BYTE CmdAdd, BYTE Length, LPSTR DataExt);
VB: Public Declare Function DelcomReadI2C Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal CmdAdd As Byte, ByVal Length As Byte, ByRef DataExt As DataExtStructure) As Long

DelcomSelReadI2C – This function read N bytes of data from the I2C port at a selected address. This function is similar to the above function but allows the user to set the address in EEPROM type device. The second data parameter byte is the set selective address command. The third data parameter byte is the selected address. The fourth data parameter byte is the address/command to read the device. The fifth data parameter byte is the length of bytes to read from the I2C port. Valid length range is 1 to 8. The data read is returned in the last parameter.

C: DWORD DelcomSelReadI2C(HANDLE hUsb, BYTE SetAddCmd, BYTE Address, BYTE ReadCmd, BYTE Length, LPSTR DataExt);
VB: Public Declare Function DelcomSelReadI2C Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal SetAddCmd As Byte, ByVal Address As Byte, ByVal ReadCmd As Byte, ByVal Length As Byte, ByRef DataExt As DataExtStructure) As Long

DelcomWriteI2CEEPROM v0.6

This write N bytes of data to a serial I2C EEPROM device. The first parameter is the handle to the USB device. The second parameter is the start address of where you want to write the data to. The third parameter is the size of the data you want to write to the device. The fourth parameter is the control code, this is the 4byte control code that is sent in the first byte (top nibble) of the I2C protocol. It usually specifies the address of the I2C device. The fifth parameter is the write delay value in milli-seconds. Typical write delay value for EEPROM are 5 and 10ms. The sixth parameter is a pointer to the data to write to the EEPROM.

C: DWORD `DelcomWriteI2CEEPROM` (HANDLE hUsb, DWORD Address, DWORD Size, BYTE CtrlCode, BYTE WriteDelay, LPSTR Data);

DelcomWriteI2CEEPROM v0.6

This read N bytes of data from a serial I2C EEPROM device. The first parameter is the handle to the USB device. The second parameter is the start address of where you want to read the data from. The third parameter is the size of the data you want to read from the device. The fourth parameter is the control code, this is the 4byte control code that is sent in the first byte (top nibble) of the I2C protocol. It usually specifies the address of the I2C device. The fifth parameter is a pointer to the data where the read data is stored. Memory for the Data pointer must be pre-declared in the users code.

C: DWORD `DelcomReadI2CEEPROM` (HANDLE hUsb, DWORD Address, DWORD Size, BYTE CtrlCode, LPSTR Data);

DelcomRS232Ctrl – This function enables or disables the RS232 port. A non-zero value passed in the second parameter will turn the serial port on, and a zero value will turn it off. The third byte is the serial baud rate, ei 2400.

C: DWORD DelcomRS232Ctrl(HANDLE hUsb, DWORD Mode, DWORD Value);
VB: Public Declare Function DelcomRS232Ctrl Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal Mode As Long, ByVal Value As Long) As Long

DelcomWriteRS232 – This function writes N bytes to the RS232 port. The second data parameter is the length of data bytes to write. Valid lengths are 1 to 8. The third data parameter is the data to write to the RS232 port. The LSB is the first byte written.

C: DWORD DelcomWriteRS232(HANDLE hUsb, DWORD Length, LPSTR DataExt);
VB: Public Declare Function DelcomWriteRS232 Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal Length As Long, ByRef DataExt As DataExtStructure) As Long

DelcomReadRS232 – This function reads the RS232 port. The function returns the length of the received data in the RS232 buffer. If the receive buffer overflows the returned value is greater than 7. A zero value means there is no new received data in the RS232 buffer. The receive buffer has a maximum size of 7 bytes. The RS232 read data is returned in the second data parameter.

C: DWORD DelcomReadRS232(HANDLE hUsb, LPSTR DataExt);
VB: Public Declare Function DelcomReadRS232 Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByRef DataExt As DataExtStructure) As Long

SPI Commands

DelcomSPIWrite – This function writes up to 64 bits of data to the SPI port. Set clock count variable to the number bits to write and the DataExt to the data to write.

C: DWORD DelcomSPIWrite(HANDLE hUsb, BYTE ClockCount, LPSTR DataExt);
VB: Public Declare Function DelcomSPIWrite Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, , ByVal ClockCount As Byte, ByRef DataExt As DataExtStructure) As Long

DelcomSPISetClock – This function sets the SPI clock period size.
Set ClockPeriod variable to adjust the SPI clock period.

C: DWORD DelcomSPISetClock(HANDLE hUsb, BYTE ClockPeriod,);
VB: Public Declare Function DelcomSPISetClock Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, , ByVal ClockPeriod As Byte) As Long

DelcomSPIRead – This function returns the read data from the last SPI write function sent.
Data is returned in the DataExt variable.

C: DWORD DelcomSPIRead(HANDLE hUsb, LPSTR DataExt);
VB: Public Declare Function DelcomSPIRead Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByRef DataExt As DataExtStructure) As Long

DelcomSPIWr8Read64 – This function writes one byte (8bits) and then reads up to 64 bits of return data from the SPI port. Set clock count variable to the number bits to read, set the WrData variable the byte value to be written and the read data will be returned in the DataExt variable.

C: DWORD DelcomSPIWr8Read64(HANDLE hUsb, BYTE WrData, BYTE ClockCount, LPSTR DataExt);
VB: Public Declare Function DelcomSPI Wr8Read64 Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, , ByVal WrData As Byte, ByVal ClockCount As Byte, ByRef DataExt As DataExtStructure) As Long

5.3 USB Numeric Display Functions

These commands refer to the USB Numeric Display Products. The first parameter is a handle to an opened USB device. Functions returns zero on success and nonzero on error unless otherwise stated.

DelcomNumericMode – This function controls the display mode of the numeric display. The first parameter is the handle to the device. The second parameter is the Mode parameter and can be 0,1,2. A 0 turns the display off, a 1 turns the display on, and a 2 put the display in blink mode. The blink rate is controlled by the third parameter. Valid range is 1 to 255, were 255 is the slowest blink rate. Units are in 10ms.

```
C: DWORD DelcomNumericMode(HANDLE hUsb, BYTE Mode, BYTE Rate );
VB: Public Declare Function DelcomNumericMode Lib "DelcomDLL.dll" _
   (ByVal DeviceHandle As Long, ByVal Mode As Byte, ByVal Rate As Byte) As Long
```

DelcomNumericScanRate – This function controls the display scan rate of the numeric display. The first parameter is the handle to the device. The second parameter is the scan rate parameter. Valid range is 0 to 255, default scan rate is 10, 255 is the slowest scan rate.

```
C: DWORD DelcomNumericScanRate(HANDLE hUsb, BYTE ScanRate );
VB: Public Declare Function DelcomNumericScanRate Lib "DelcomDLL.dll" _
   (ByVal DeviceHandle As Long, ByVal ScanRate As Byte) As Long
```

DelcomNumericSetup – This function controls the number of digits that the numeric display is scan. The first parameter is the handle to the device. The second parameter is the number of digits to display. Current valid range is 0 to 8, default is 6.

```
C: DWORD DelcomNumericSetup(HANDLE hUsb, BYTE Digits );
VB: Public Declare Function DelcomNumericSetup Lib "DelcomDLL.dll" _
   (ByVal DeviceHandle As Long, ByVal Digits As Byte) As Long
```

DelcomNumericRaw – This function sends data to the display in raw format mode. The first parameter is the handle to the device. The second parameter is a string to 8 characters. This string must be at least 8 characters long and must be defined by the user. The LSB will be digit 0. This mode gives the user total control of the numeric display. See the USB Numeric Display Development Manual for raw display codes.

```
C: DWORD DelcomNumericRaw(HANDLE hUsb, LPSTR Str );
VB: Public Declare Function DelcomNumericRaw Lib "DelcomDLL.dll" _
   (ByVal DeviceHandle As Long, ByVal Str As String) As Long
```

DelcomNumericInteger – This function sends data to the display in decimal integer format. The first parameter is the handle to the device. The second parameter is a signed integer. This parameter can be positive or negative. A minus sign is prefixed to negative numbers if there is space to show it. The third parameter is the base parameter. Base parameter controls the placement off the decimal point. Valid values are; 0=no decimal point, 1=right decimal point, 10=tenth decimal point, 100=hundredth decimal point, and so on. Values are always right justified and fractions are padded with zeros.

```
C: DWORD DelcomNumericInteger(HANDLE hUsb, DWORD Number, DWORD Base );
```

VB: Public Declare Function DelcomNumericInteger Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal Number As Long, ByVal Base As Long) As Long

DelcomNumericHexaDecimal – This function sends data to the display in hexadecimal integer format. This function is the same as the above function accept the display radix 16.

C: DWORD DelcomNumericHexaDecimal(HANDLE hUsb, DWORD Number, DWORD Base);
VB: Public Declare Function DelcomNumericHexaDecimal Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal Number As Long, ByVal Base As Long) As Long

DelcomNumericDouble – This function sends data to the display in real numbers format. The first parameter is the handle to the device. The second parameter is a signed real number. This parameter can be positive or negative. A minus sign is prefixed to the number if there is space to show it. The third parameter is the base parameter. Base parameter controls the placement off the decimal point. Valid values are; 0=no decimal point, 1=right decimal point, 10=tenth decimal point, 100=hundredth decimal point, and so on. Values are always right justified and fractions are padded with zeros. Number are not rounded when precision is dropped.

C: DWORD DelcomNumericDouble(HANDLE hUsb, double Number, DWORD Base);
VB: Public Declare Function DelcomNumericDouble Lib "DelcomDLL.dll" _
(ByVal DeviceHandle As Long, ByVal Number As Double, ByVal Base As Long) As Long

Numeric Display Examples:

DelcomNumericInteger (hUsb, 123456, 0)	=> 123456
DelcomNumericInteger (hUsb, 123456, 1)	=> 123456.
DelcomNumericInteger (hUsb, 123456, 10)	=> 12345.6
DelcomNumericInteger (hUsb, -123456, 100)	=> -1234.56
DelcomNumericInteger (hUsb, 0, 100)	=> 0.00
DelcomNumericInteger (hUsb, 5, 100)	=> 0.05
DelcomNumericHexaDecimal (hUsb, 255, 0)	=> FF
DelcomNumericHexaDecimal (hUsb, 255, 10)	=> F.F
DelcomNumericHexaDecimal (hUsb, -10, 0)	=> -A
DelcomNumericDouble (hUsb, 123.45, 100)	=> 123.45
DelcomNumericDouble (hUsb, 0.45, 100)	=> 0.45
DelcomNumericDouble (hUsb, 0.05, 100)	=> 0.05
DelcomNumericDouble (hUsb,- 0.05, 100)	=> -0.05
DelcomNumericDouble (hUsb, 12.34, 0)	=> 0
DelcomNumericDouble (hUsb, 12.34, 1)	=> 12.
DelcomNumericDouble (hUsb, 12.34, 10)	=> 12.3
DelcomNumericDouble (hUsb, 12.34, 100)	=> 12.34
DelcomNumericDouble (hUsb, 12.34, 1000)	=> 12.340
DelcomNumericDouble (hUsb, -123.45, 100)	=> -123.45

6.0 Release Notes

Version	Date	
0.1	02/13/2003	-Initial Release
0.2	04/09/2003	-Added prototyping for VB support all functions are now _stdcall. -Added USBIODS function support. -Added DelcomGetNthDevice() command. -Changed the prototyping of DelcomGetDLLData().
0.3	08/04/2003	-Fixed memory leak error with DelcomGetNthDevice() function.
0.4	03/01/2004	-Fixed error with DelcomGetNthDevice() function when trying to get the device number 1 or bigger in VB. -Added Delcom Numeric Display Support.
0.5	05/25/2004	-Added DelcomLEDPower function.
0.6	02/12/2007	-Added the following functions DelcomWritePin() DelcomReadI2CEEPROM() DelcomWriteI2CEEPROM()
0.6	07/18/2008	-Added the following SPI functions `DelcomSPIWrite() DelcomSPISetClock() DelcomSPIRead() DelcomSPIWr8Read64()`
0.7	11/13/2008	-Added basic support for USB HID products.
0.8	11/18/2008	-Added full support for USB HID products.
0.9	02/09/2009	- Corrected DelcomNumericInteger function to correctly set the 8th digit to a minus sign when value is negative.
1.0	03/05/2009	- Corrected SendPacket function when sending generation I command 11-3 (read ports) when target is a generation II chip.
1.1	08/06/2010	-Fixed SetLEDPower function
1.2	02/13/2003	-Convert code to 64 bit.
1.3	02/13/2003	-Changed GetButtonStatus to check for failed HID read.
1.4	06/09/2011	-Added SetupPorts Functions
1.5	04/05/2012	-Changed HID translator code to retry on read failure.
1.6	07/20/2012	-Changed open G2 devices function to open in shared mode.
1.7	10/10/2016	-Changed all G2 commands to 16byte length, was 8 byte length.
1.8	10/14/2016	-Changed GetButtonStatus() to return -1 (0xFFFFFFFF) on error.

7.0 Errata

Currently the DLL will only allow one application to open a unique device at the same time.

If you would like to see a DLL function added to the DLL please email your request to techsupport@delcomproducts.com. If the request is merited we will add your requested function to the DLL.

8.0 Sample Code

8.1 Send and Read Data Packet

```
// DelcomDLLExampleC
#include "stdafx.h"
#include "DelcomDLL.h"
int main(int argc, char* argv[])
{
	PacketStruct Packet;
	char DeviceName[MaxDeviceLen];
	if(!DelcomGetNthDevice(USBIODS, 0, DeviceName)) return(0);

	HANDLE hUsb = DelcomOpenDevice((char*)DeviceName,0);
	// Write Packet
	Packet.Recipient = 8;		// always 8
	Packet.DeviceModel = 18; // always 18
	Packet.MajorCmd = 10;
	Packet.MinorCmd = 10;		// write port0 & port1
	Packet.DataLSB = 0xFF;		// set port0 to all high
	Packet.DataMSB = 0x00;		// set port1 to all low
	Packet.Length = 0;		// DataExt not used
	DelcomSendPacket(hUsb,&Packet,NULL);

	// Write Packet with Data Ext
	Packet.Recipient = 8;		// always 8
	Packet.DeviceModel = 18; // always 18
	Packet.MajorCmd = 10;
	Packet.MinorCmd = 60;		// write port0 & port1
	Packet.DataLSB = 0xFF;		// set port0 to all high
	Packet.DataMSB = 0x00;		// set port1 to all low
	Packet.Length = 3;		// DataExt used, sending 3 bytes
	Packet.ExtData[0] = 1;
	Packet.ExtData[1] = 2;
	Packet.ExtData[2] = 3;
	DelcomSendPacket(hUsb,&Packet,NULL);

	// Read Packet
	Packet.Recipient = 8;		// always 8
	Packet.DeviceModel = 18; // always 18
	Packet.MajorCmd = 11;
	Packet.MinorCmd = 0;		// read port0 & port1
	Packet.Length = 0;			// DataExt not used
	DelcomSendPacket(hUsb,&Packet,&Packet);
	printf("Port0=%X Port1=%X\n",((char*)&Packet)[0],((char*)&Packet)[1]);

	DelcomCloseDevice(hUsb);	// close the device

	return(0);
}
```

8.2 I2C Example

```
// DelcomDLLExampleC
#include "stdafx.h"
#include "DelcomDLL.h"
int main(int argc, char* argv[])
{
	char DataExt[8];

	char DeviceName[MaxDeviceLen];
	if(!DelcomGetNthDevice(USBIODS, 0, DeviceName)) return(0);

	HANDLE hUsb = DelcomOpenDevice((char*)DeviceName,0);
	// writes two bytes (0xFE & 0x34 ) to I2C at address/command 0x5E
	DataExt[0] = (char)0xFE;
	DataExt[1] = (char)0x34;
	DelcomWriteI2C( hUsb, 0x5E, 2, DataExt );

	// reads 8 bytes from I2C at the current address
	DelcomReadI2C(  hUsb, 0x9A, 8, DataExt );
	// LSB = DataExt[0];

	// reads 2 bytes from I2C at address 0x00
	DelcomSelReadI2C( hUsb, 0x1E, 0x00, 0x9A, 2, DataExt );
	// LSB = DataExt[0];

	DelcomCloseDevice(hUsb);		// close the device

	return 0;
}
```