

```

' This module is common to all of the Example programs
' It declares the {Open, Read, Write, Close} calls for the USB device
' These user-calls are translated into OS system calls
' This module also contains several support routines used by all of the examples
'
' Declare module-wide variables
Private HidHandle As Long
Public Function OpenUSBdevice(NameOfDevice$) As Boolean
' This function searches the system HID tables for NameOfDevice$
' If found then it opens the device and returns TRUE, else it returns FALSE
Dim HidGuid As Guid: Dim Success As Boolean: Dim Openned As Boolean: Dim Buffer(256) As Byte
Dim DeviceInterfaceData As Device_Interface_Data
Dim FunctionClassDeviceData As Device_Interface_Detail
'
' First, get the HID class identifier
Call HidD_GetHidGuid(HidGuid.Data(0))
' Get a handle for the Plug and Play node, request currently active HID devices
PnPHandle& = SetupDiGetClassDevs(HidGuid.Data(0), 0, 0, &HID)
If (PnPHandle& = -1) Then ErrorExit ("Could not attach to PnP node")
'
HidEntry& = 0: Openned = False
DeviceInterfaceData.cbSize = 28 'Length of data structure in bytes
' Look through the table of HID devices
Do While SetupDiEnumDeviceInterfaces(PnPHandle&, 0, HidGuid.Data(0), HidEntry&, DeviceInterfaceData.cbSize)
' There is a device here, get it's name
FunctionClassDeviceData.cbSize = 5
Success = SetupDiGetDeviceInterfaceDetail(PnPHandle&, DeviceInterfaceData.cbSize, _
FunctionClassDeviceData.cbSize, UBound(FunctionClassDeviceData.DataPath), BytesReturned&, 0)
If (Success = 0) Then ErrorExit ("Could not get the name of this HID device")
' Convert returned C string to Visual Basic String
hidname$ = "": i& = 0
Do While FunctionClassDeviceData.DataPath(i&) <> 0
hidname$ = hidname$ & Chr$(FunctionClassDeviceData.DataPath(i&)): i& = i& + 1: Loop
' Can now open this HID device
Dim SA As Security_Attributes
HidHandle& = CreateFile(hidname$, &HC0000000, 3, SA, 3, 0, 0)
If (HidHandle = -1) Then ErrorExit ("Could not open HID device")
' Is it OUR HID device?
If HidD_GetProductString(HidHandle&, AddressFor(Buffer(0)), UBound(Buffer)) Then
DeviceName$ = "": i& = 0
Do While Buffer(i&) <> 0: DeviceName$ = DeviceName$ & Chr$(Buffer(i&)): i& = i& + 2: Loop
If (StrComp(DeviceName$, NameOfDevice$) = 0) Then
Openned = True: Exit Do: End If
End If 'HidD_GetProductString
Call CloseHandle(HidHandle&) ' Was not OUR HID device
HidEntry& = HidEntry& + 1 ' Check next entry
Loop 'SetupDiEnumDeviceInterfaces returns FALSE when there are no more entries
SetupDiDestroyDeviceInfoList (PnPHandle&)
OpenUSBdevice = Openned
End Function

Public Sub ReadUSBdevice(BufferPtr&, ByteCount&)
' This subroutine "reads" from an opened USB device
' This routine gets an Input Report from the USB device and returns the data
' NOTE that ReadFile is a BLOCKING system call, ie it will wait for the USB device to respond
' Do not configure the USB device to "Generate report only on change" since the program
' will appear to 'hang'
' Use a local buffer so that the ReportID (=0) at ReportBuffer(0) may be removed
Dim ReportBuffer(256) As Byte
If ByteCount& > 254 Then ErrorExit ("Maximum ByteCount for ReadUSBdevice is 254")
If ByteCount& < 1 Then ErrorExit ("Minimum ByteCount for ReadUSBdevice is 1")
Success = ReadFile(HidHandle&, AddressFor(ReportBuffer(0)), ByteCount& + 1, BytesReturned&, 0)
If (Success = 0) Then ErrorExit ("Could not get an Input Report")
Call CopyBuffer(AddressFor(ReportBuffer(1)), BufferPtr&, BytesReturned& - 1)
End Sub

Public Sub WriteUSBdevice(BufferPtr&, ByteCount&)
' This subroutine "writes" to an opened USB device
' Copy the user buffer into a local buffer so that a ReportID (=0) may be prepended
' The first byte will contain the ReportID (=0)
Dim ReportBuffer(256) As Byte
If ByteCount& > 254 Then ErrorExit ("Maximum ByteCount for WriteUSBdevice is 254")
Call CopyBuffer(BufferPtr&, AddressFor(ReportBuffer(1)), ByteCount&)
ReportBuffer(0) = 0 ' ReportID
Success = WriteFile(HidHandle&, AddressFor(ReportBuffer(0)), ByteCount& + 1, BytesWritten&, 0)
If (Success = 0) Then ErrorExit ("Could not write an Output Report")
End Sub

Public Sub CloseUSBdevice()

```

```
' This subroutine closes the USB device that we have been using
Call CloseHandle(HidHandle&)
End Sub
```

```
Public Function ReturnHexByte(Text$) As Byte
' Converts the first two characters of text$ into a byte
Dim Value As Byte
Utext$ = UCase(Text$) ' Convert to uppercase for search
HexString$ = "0123456789ABCDEF" ' Non-Hex characters = 0
Value = 0
For i& = 0 To 15
    If Mid(Utext$, 1, 1) = Mid(HexString$, i& + 1, 1) Then Value = Value + (16 * i&)
    If Mid(Utext$, 2, 1) = Mid(HexString$, i& + 1, 1) Then Value = Value + i&
Next i&
ReturnHexByte = Value
End Function
```

```
Public Function TwoHexCharacters$(Value As Byte)
HexString$ = "0123456789ABCDEF"
TwoHexCharacters$ = Mid(HexString$, Int(Value / 16) + 1, 1) & Mid(HexString$, Int(Value And &HF) + 1, 1)
End Function
```

```
Public Function TwoDecimalCharacters$(Value As Byte)
DecimalString$ = "0123456789"
Tens& = Int(Value / 10): Units& = Value - (10 * Tens&)
TwoDecimalCharacters$ = Mid(DecimalString$, Tens& + 1, 1) & Mid(DecimalString$, Units& + 1, 1)
End Function
```

```
Public Function ThreeDecimalCharacters$(Value As Byte)
h& = Int(Value / 100): t& = Int((Value - (100 * h&)) / 10): u& = Value - (100 * h&) - (10 * t&)
ThreeDecimalCharacters$ = h& & t& & u&
End Function
Public Sub ErrorExit(Reason$)
```

```
ErrorCode = GetLastError()
Call MsgBox(Reason$, vbCritical)
Stop
End Sub
```