frmMain

Imports System.Net.Sockets

Imports System.Text

Imports System.IO.Ports

Public Class frmMain

Public rClient As New TcpClient() 'TCP connection object

Public rStream As NetworkStream 'TCP buffer access

Public rSerial As New SerialPort 'Serial IO

Dim Packet As New rAGWPEMsg 'Assembles AGWPE Frames

Sub connect()

Try

If Not rClient.Connected Then

Dim Port As Integer = nbrPort.Value

Dim tAdd As Long

'Get IP

Dim IP1, IP2, IP3, IP4 As Long

IP1 = nbrIP1.Value

IP2 = nbrIP2.Value

IP3 = nbrIP3.Value

IP4 = nbrIP4.Value

tAdd = IP1 + (IP2 \* &H100) + (IP3 \* &H10000) + (IP4 \* &H1000000)

Dim IPadd As New System.Net.IPAddress(tAdd)

Try

'Attempt connection

rClient.Connect(IPadd, Port)

'Get stream

rStream = rClient.GetStream()

If rStream.CanRead And rStream.CanWrite Then

'If stream accessible

lblConnect.Visible = True

rMessage("AGWPE Connected")

ElseIf rStream.CanRead = False Then

Call rMessage("Cannot read")

rStream.Close()

ElseIf rStream.CanWrite = False Then

Call rMessage("Cannot write")

rStream.Close()

End If

Catch ex As Exception

MsgBox("Connection Failed. Ensure AGWPE is running. Loopback port must be enabled on specified IP and port. Error = " & ex.Message)

lblConnect.Visible = False

End Try

Else

MsgBox("Already Connected")

End If

Catch ex As Exception

MsgBox("Error while connecting. Error = " & ex.Message)

End Try

End Sub

Sub ToAGWPE(ByVal mode As String, Optional ByVal position As String = "")

Dim msg() As Byte 'AGWPE frame

Dim extra() As Byte 'Extra info e.g. AX.25 frames

Try

'General purpose sub for comms with AGWPE

Select Case mode

Case Is = "R", "G", "m"

'R=Version number request, G=Port list request, m=begin monitoring(not used in auto mode)

msg = Packet.ToBytes(mode)

Case Is = "g"

'Port capabilities request

If lstPorts.SelectedIndex > 0 Then

msg = Packet.ToBytes(mode, lstPorts.SelectedIndex - 1)

Else

MsgBox("Please select a port")

Exit Sub

End If

Case Is = "M"

'This transmits APRS standard frames. However APRS Analyser does not accept APRS standard.

'Therefore M is for APRSPoint only, for APRS Analyser as well, use M\*

If lstPorts.SelectedIndex > 0 Then

'Position contains the reformatted balloon frame

'In auto mode position is passed from the reformatting subroutine

If position = "" Then

If optAssemble.Checked = True Then position = Manual\_Assemble()

If optManual.Checked = True Then position = txtPosition.Text

End If

extra = Packet.APRSFrame(position, txtCallTo.Text, txtCallsign.Text)

msg = Packet.ToBytes(mode, lstPorts.SelectedIndex - 1, , txtCallsign.Text, txtCallTo.Text, extra.Length + 1)

Else

MsgBox("Please select a port")

Exit Sub

End If

Case Is = "M\*"

'MAJOR USE - this mode transmits actual frames through AGWPE

'This is for plotting to APRSPoint and interface with APRS Analyser

If lstPorts.SelectedIndex > 0 Then

If position = "" Then

position = Manual\_Assemble()

End If

extra = Packet.APRSFrame(position)

msg = Packet.ToBytes(mode, lstPorts.SelectedIndex - 1, , txtCallsign.Text, txtCallTo.Text, extra.Length + 1)

Else

MsgBox("Please select a port")

Exit Sub

End If

End Select

Catch ex As Exception

MsgBox("Error while assembling AGWPE frame. Error = " & ex.Message)

End Try

'Now write assembled frames to the TCP buffer

Try

Select Case mode

Case Is = "M", "M\*"

Dim write(msg.Length + extra.Length) As Byte

Dim feedback As String

'Combine AGWPE frame and balloon frame

Buffer.BlockCopy(msg, 0, write, 0, msg.Length)

Buffer.BlockCopy(extra, 0, write, msg.Length, extra.Length)

rStream.Write(write, 0, write.Length)

'Log frame send and contents of send

Call rMessage("Sent " & mode & " packet")

feedback = System.Text.Encoding.ASCII.GetChars(extra)

Call rMessage(feedback)

Case Is = "m"

rStream.Write(msg, 0, msg.Length)

Call rMessage("Sent " & mode & " packet")

Exit Sub

Case Else

rStream.Write(msg, 0, msg.Length)

Call rMessage("Sent " & mode & " packet")

End Select

If mode <> "M" And mode <> "M\*" And mode <> "m" Then

'Only await response on frames which provide one - otherwise program locks up

Dim response(rClient.ReceiveBufferSize) As Byte

rStream.Read(response, 0, response.Length)

Packet.FromBytes(response)

Select Case mode

Case Is = "R"

'Version number

Call rMessage("Received version number (" & Packet.DataKind & " packet.) " & "Version = " & Packet.majVers & "." & Packet.minVers)

lblVersion.Text = "AGWPE Version: " & Packet.majVers & "." & Packet.minVers

Case Is = "G"

'Ports list

If Packet.nPorts > 0 Then

'Create port list

lstPorts.Items.Clear()

lstPorts.Items.Add("Available Ports")

For rcount = 0 To Packet.nPorts - 1

lstPorts.Items.Add(Packet.PortInfo(rcount, 0))

Next

Else

lstPorts.Items.Clear()

lstPorts.Items.Add("No Available Ports")

End If

Call rMessage("Received Port List (G packet)")

Case Is = "g"

'Port capabilities

Dim ActivePort As Integer = (lstPorts.SelectedIndex - 1)

lstCapabilities.Items.Clear()

lstCapabilities.Items.Add("Port " & (ActivePort + 1) & ": ")

lstCapabilities.Items.Add("Baud: " & Packet.PortInfo(ActivePort, 1))

lstCapabilities.Items.Add("Traffic: " & Packet.PortInfo(ActivePort, 2))

lstCapabilities.Items.Add("TX Delay: " & Packet.PortInfo(ActivePort, 3))

lstCapabilities.Items.Add("TX Tail: " & Packet.PortInfo(ActivePort, 4))

lstCapabilities.Items.Add("Persist: " & Packet.PortInfo(ActivePort, 5))

lstCapabilities.Items.Add("SlotTime: " & Packet.PortInfo(ActivePort, 6))

lstCapabilities.Items.Add("MaxFrame: " & Packet.PortInfo(ActivePort, 7))

lstCapabilities.Items.Add("Active Connections: " & Packet.PortInfo(ActivePort, 8))

lstCapabilities.Items.Add("Bytes in last 2 mins: " & Packet.PortInfo(ActivePort, 9))

Call rMessage("Received Port Capabilities (g packet)")

Case Is = "U"

'Frame report - mirrors sent frame. Not used in auto mode

Call rMessage(Encoding.ASCII.GetChars(response))

End Select

End If

Catch ex As Exception

MsgBox("Error while sending AGWPE frame. Error = " & ex.Message)

lblConnect.Visible = False

End Try

End Sub

Private Sub btnAuto\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAuto.Click

Try

'Begins auto routine

Call connect()

If lblConnect.Visible = True Then

Call ToAGWPE("R") 'Fetch version

Call ToAGWPE("G") 'Fetch ports

If chkDefaultPort.Checked = True Then

lstPorts.SelectedIndex = nbrDefaultPort.Value 'Select default port

Call ToAGWPE("g") 'Fetch port capabilities

End If

Call rMessage("Sending Test Report")

Call ToAGWPE("M\*", Manual\_Assemble) 'Send test location report

If chkAuto.Checked = True Then 'In full auto mode

Call rMessage("Starting Serial Listener")

Call SerialConnect() 'Initialise serial connection

If rSerial.IsOpen Then

lblSerialConnect.Visible = True

rMessage("Serial Connected")

End If

End If

End If

Catch ex As Exception

MsgBox("Error in auto routine. Error = " & ex.Message)

End Try

End Sub

Private Sub lstPorts\_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs) Handles lstPorts.DoubleClick

Call ToAGWPE("g")

'get port capabilities

End Sub

Private Sub chkDefaultPort\_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles chkDefaultPort.CheckedChanged

If chkDefaultPort.Checked = True Then

nbrDefaultPort.Enabled = True

Else

nbrDefaultPort.Enabled = False

End If

End Sub

Public Function Manual\_Assemble()

'only used in manual mode - creates balloon report frame from manual settings

Dim time As String

Dim latd As String

Dim latm As String

Dim lats As String

Dim longd As String

Dim longm As String

Dim longs As String

Dim heading As String

Dim speed As String

Dim altitude As String

Dim custom As String

time = txtTime.Text

latd = txtLatD.Text

latm = txtLatM.Text

lats = txtLatS.Text

longd = txtLongD.Text

longm = txtLongM.Text

longs = txtLongS.Text

heading = txtHeading.Text

speed = txtSpeed.Text

altitude = txtAltitude.Text

custom = txtCustom.Text

Manual\_Assemble = AssembleExtras(time, latd, latm, lats, longd, longm, longs, heading, speed, altitude, custom)

End Function

Public Sub Receive()

Try

'Event handler for serial connection

Dim received As String

Dim forward() As Byte

received = rSerial.ReadExisting()

'take available serial data

Call rMessage("Received Serial Message: " & received)

'log unprocessed message

Call rMessage("Processing...")

forward = Encoding.ASCII.GetBytes(received)

'convert to byte array

Dim message As String

message = Decode(forward)

'decode UKHAS format into balloon format

Call ToAGWPE("M\*", message)

'send frame to APRS Analyser and APRSPoint

Catch ex As Exception

MsgBox("Error receiving serial communication. Error = " & ex.Message)

End Try

End Sub

Public Function Decode(ByVal received() As Byte)

Try

Dim data(7) As String

'seven UKHAS frame fields

Dim rcount As Integer

Dim temp As Char

rcount = 1

If Microsoft.VisualBasic.Chr(received(0)) = "$" And Microsoft.VisualBasic.Chr(received(1)) = "$" Then

For rcounter = 2 To (received.Length - 1)

'first two characters are $$ so skip

temp = Microsoft.VisualBasic.Chr(received(rcounter)) 'Gets next character in string

If temp <> "," Then 'if not new field

If temp = "" Then Exit For 'if reached end of frame

If rcount <> 3 Then

data(rcount) = data(rcount) & temp

Else 'remove colons from time field

If temp <> ":" Then

data(rcount) = data(rcount) & temp

End If

End If

Else : rcount = rcount + 1

End If

Next

Else

MsgBox("Frame corrupted")

End If

'creates individual fields from UKHAS frame data

Dim time As String

Dim latd As String

Dim latm As String

Dim lats As String

Dim longd As String

Dim longm As String

Dim longs As String

Dim heading As String

Dim speed As String

Dim altitude As String

Dim custom As String

'ensures correct length by adding leading zeroes

If data(3).Length = 6 Then

time = data(3)

Else

time = data(3)

For rcount = 1 To (6 - data(3).Length)

time = "0" & time

Next

End If

'converts from degrees to degrees and minutes

latd = Int(data(4))

If latd.Length < 2 Then

For rcount = 1 To (2 - latd.Length)

latd = "0" & latd

Next

End If

latm = Int((data(4) \* 60) - (latd \* 60))

If latm.Length < 2 Then

For rcount = 1 To (2 - latm.Length)

latm = "0" & latm

Next

End If

lats = Int((data(4) \* 6000) - (latd \* 6000) - (latm \* 100))

If lats.Length < 2 Then

For rcount = 1 To (2 - lats.Length)

lats = "0" & lats

Next

End If

longd = Int(data(5))

If longd.Length < 3 Then

For rcount = 1 To (3 - longd.Length)

longd = "0" & longd

Next

End If

longm = Int((data(5) \* 60) - (longd \* 60))

If longm.Length < 2 Then

For rcount = 1 To (2 - longm.Length)

longm = "0" & longm

Next

End If

longs = Int((data(5) \* 6000) - (longd \* 6000) - (longm \* 100))

If longs.Length < 2 Then

For rcount = 1 To (2 - longs.Length)

longs = "0" & longs

Next

End If

'these are not reported in UKHAS, so are set to nothing in balloon frame

heading = "000"

speed = "000"

If data(6).Length = 5 Then

altitude = data(6)

Else

altitude = data(6)

For rcount = 1 To (5 - data(6).Length)

altitude = "0" & altitude

Next

End If

'balloon sensor data

custom = data(7)

'builds balloon format frame from individual fields

Decode = AssembleExtras(time, latd, latm, lats, longd, longm, longs, heading, speed, altitude, custom)

Catch ex As Exception

MsgBox("Error while decoding UKHAS frame. Error = " & ex.Message)

Decode = ""

End Try

End Function

Public Function AssembleExtras(ByVal time As String, ByVal latd As String, ByVal latm As String, ByVal lats As String, ByVal longd As String, ByVal longm As String, ByVal longs As String, ByVal heading As String, ByVal speed As String, ByVal altitude As String, ByVal custom As String)

Dim Position As String

'builds balloon format frame from individual fields

Position = "@"

Position = Position & time & "h"

Position = Position & latd

Position = Position & latm & "."

Position = Position & lats & "N/"

Position = Position & longd

Position = Position & longm & "."

Position = Position & longs & "W"

Position = Position & heading & "/"

Position = Position & speed & "/A="

Position = Position & altitude

Position = Position & custom

AssembleExtras = Position

End Function

Public Sub SerialConnect()

Try

Dim rParity As Integer

Dim rBaud As Integer

Dim rPort As String

Dim rStopBits As Integer

Select Case cmbParity.Text

Case Is = "None"

rParity = 0

Case Is = "Odd"

rParity = 1

Case Is = "Even"

rParity = 2

Case Is = "Mark"

rParity = 3

Case Is = "Space"

rParity = 4

End Select

Select Case cmbStopBits.Text

Case Is = "One"

rStopBits = 1

Case Is = "Two"

rStopBits = 2

Case Is = "OnePointFive"

rStopBits = 3

End Select

rBaud = nbrBaud.Value

rPort = cmbSerialPorts.Text

'set serial parameters

rSerial.Parity = rParity

rSerial.BaudRate = rBaud

rSerial.PortName = rPort

rSerial.StopBits = rStopBits

Catch ex As Exception

MsgBox("Error while setting serial properties. Error = " & ex.Message)

End Try

Try

'attempt to open connection, and set event handler for incoming transmission

rSerial.Open()

AddHandler rSerial.DataReceived, AddressOf Receive

Catch ex As Exception

MsgBox("Serial Port could not be opened. Ensure selected port exists and is available. Error = " & ex.Message)

End Try

End Sub

Private Sub frmMain\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Dim ports As String() = SerialPort.GetPortNames()

'populate serial port list

' Display each port name

Dim port As String

For Each port In ports

cmbSerialPorts.Items.Add(port)

Next port

'import settings

chkDefaultPort.Checked = My.Settings.Default\_Port

txtCallsign.Text = My.Settings.Callsign

txtCallTo.Text = My.Settings.Callto

txtSave.Text = My.Settings.Log\_filepath

cmbSerialPorts.Text = My.Settings.Serial\_Port

nbrBaud.Value = My.Settings.Baud\_Rate

cmbStopBits.Text = My.Settings.Stop\_Bits

cmbParity.Text = My.Settings.Parity

chkAuto.Checked = My.Settings.Full\_Auto

End Sub

Private Sub chkManual\_CheckedChanged\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles chkManual.CheckedChanged

If chkManual.Checked = False Then

grpManual.Enabled = False

Else

grpManual.Enabled = True

End If

End Sub

Private Sub btnSerial\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSerial.Click

'manual serial connect

Call SerialConnect()

End Sub

Public Sub rMessage(ByVal text As String)

Try

'used to log messages, it displays messages and logs them to file

lstMessages.Items.Add(text)

Dim filename As String = txtSave.Text

If My.Computer.FileSystem.FileExists(filename) Then

'If filename exists, append

My.Computer.FileSystem.WriteAllText(filename, Today & " " & TimeOfDay & " ", True)

My.Computer.FileSystem.WriteAllText(filename, text, True)

My.Computer.FileSystem.WriteAllText(filename, Chr(13) & Chr(10), True)

Else

'if filename does not exist, log all previous messages

For Each item In lstMessages.Items

My.Computer.FileSystem.WriteAllText(filename, Today & " " & TimeOfDay & " ", True)

My.Computer.FileSystem.WriteAllText(filename, item, True)

My.Computer.FileSystem.WriteAllText(filename, Chr(13) & Chr(10), True)

Next

End If

Catch ex As Exception

MsgBox("Error while logging. Error = " & ex.Message)

End Try

End Sub

Private Sub btnBrowse\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnBrowse.Click

'select filename

dlogSave.ShowDialog()

txtSave.Text = dlogSave.FileName

End Sub

Private Sub btnSave\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSave.Click

'save settings

My.Settings.Default\_Port = chkDefaultPort.Checked

My.Settings.Callsign = txtCallsign.Text()

My.Settings.Callto = txtCallTo.Text()

My.Settings.Log\_filepath = txtSave.Text()

My.Settings.Serial\_Port = cmbSerialPorts.Text

My.Settings.Baud\_Rate = nbrBaud.Value

My.Settings.Stop\_Bits = cmbStopBits.Text

My.Settings.Parity = cmbParity.Text()

My.Settings.Full\_Auto = chkAuto.Checked()

My.Settings.Save()

End Sub

Private Sub btnSend\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSend.Click

'Not used in auto - manually sends single frame

Call ToAGWPE(cmbFrameSelect.Text)

End Sub

Private Sub btnConnect\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnConnect.Click

Call connect()

End Sub

Private Sub btnDisconnect\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDisconnect.Click

rStream.Close()

rClient.Close()

lblConnect.Visible = False

MsgBox("Program must be restarted to reconnect.")

End Sub

Private Sub btnRefresh\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRefresh.Click

'refreshes connection statuses

lblConnect.Visible = rClient.Connected

lblSerialConnect.Visible = rSerial.IsOpen

End Sub

Private Sub btnDecode\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDecode.Click

'manual mode decode

MsgBox(Decode(Encoding.ASCII.GetBytes(txtUKHAS.Text)))

Dim message As String

message = Decode(Encoding.ASCII.GetBytes(txtUKHAS.Text))

Call ToAGWPE("M\*", message)

End Sub

End Class

rAGWPEMsg

Imports System.Net.Sockets

Imports System.Text

Imports System.IO.Ports

Public Class rAGWPEMsg

Public Port As Byte 'All have defaults, since most are not used in every type of packet

Public DataKind As Char 'DataKind has no default, since it is used in every packet

Public PID As Byte

Public CallFrom As String

Public CallTo As String

Public DataLen As Int32

'Received data

Public majVers As Int16

Public minVers As Int16

Public nPorts As Integer

Public PortInfo(,) As Object

Private msg(35) As Byte 'byte array of AGWPE frame to be sent

Public Sub New()

'initialise

Port = 0

PID = 0

CallFrom = Nothing

CallTo = Nothing

DataLen = 0

End Sub

Public Function ToBytes(ByVal DataKind As Char, Optional ByVal Port As Byte = 0, Optional ByVal PID As Byte = 0, Optional ByVal CallFrom As String = Nothing, Optional ByVal CallTo As String = Nothing, Optional ByVal DataLen As Int32 = 0, Optional ByVal User As Int32 = 0)

'All have defaults, since most are not used in every type of packet

'DataKind has no default, since it is used in every packet

Dim msg(35) As Byte

Dim temp() As Byte 'used as inbetween for callsign byte conversions

Dim rmod As Integer = 0

For rCount = 0 To 35

msg(rCount) = 0

Next

'built up according to AGWPE format

msg(0) = Port

msg(4) = Microsoft.VisualBasic.AscW(DataKind)

msg(6) = PID

If CallFrom <> "" Then

temp = Encoding.ASCII.GetBytes(CallFrom)

For rCount = 0 To (temp.Length - 1)

msg(rCount + 8) = temp(rCount)

Next

End If

If CallTo <> "" Then

temp = Encoding.ASCII.GetBytes(CallTo)

For rCount = 0 To (temp.Length - 1)

msg(rCount + 18) = temp(rCount)

Next

End If

msg(28) = (DataLen And &HFF)

msg(29) = ((DataLen And &HFF00) / &H100)

msg(30) = ((DataLen And &HFF0000) / &H10000)

msg(31) = ((DataLen And &HFF000000) / &H1000000)

'return byte array

ToBytes = msg

End Function

Public Function APRSFrame(ByVal Info As String, Optional ByVal Destination As String = "", Optional ByVal Source As String = "")

Try

Dim output() As Byte 'array to return

Dim temp1() As Byte 'callto

Dim temp2() As Byte 'callsign

Dim temp3() As Byte 'custom info (i.e. balloon sensor data)

Dim length As Integer 'length of frame

If Destination <> "" Then 'M frame - APRS standard

temp1 = Encoding.ASCII.GetBytes(Destination)

temp2 = Encoding.ASCII.GetBytes(Source)

temp3 = Encoding.ASCII.GetBytes(Info)

length = temp1.Length + temp2.Length + temp3.Length

ReDim output(length + 5)

'copy fields into output in order

Buffer.BlockCopy(temp1, 0, output, 1, 7)

Buffer.BlockCopy(temp2, 0, output, temp1.Length + 1, 7)

Buffer.BlockCopy(temp3, 0, output, temp1.Length + temp2.Length + 3, temp3.Length)

output(0) = 126

output(15) = 3

output(16) = 240

output(19 + Info.Length) = 126

Else 'M\* frame - for APRS Analyser comms

'contains no callsigns

temp3 = Encoding.ASCII.GetBytes(Info)

ReDim output(temp3.Length)

Buffer.BlockCopy(temp3, 0, output, 0, temp3.Length)

End If

APRSFrame = output

Catch ex As Exception

MsgBox("Error while building balloon frame. Error = " & ex.Message)

APRSFrame = ""

End Try

End Function

Public Sub FromBytes(ByVal response() As Byte)

Try

Dim temp(9) As Byte 'go-between for conversions

'frame is according to known structure, so fields can be extracted

Port = response(0)

DataKind = Microsoft.VisualBasic.Chr(response(4))

PID = response(6)

For rCount = 8 To 17

temp(rCount - 8) = response(rCount)

Next

CallFrom = Encoding.ASCII.GetChars(temp)

For rCount = 18 To 27

temp(rCount - 18) = response(rCount)

Next

CallTo = Encoding.ASCII.GetChars(temp)

DataLen = response(28) + (response(29) \* &H100) + (response(30) \* &H10000) + (response(31) \* &H1000000)

'once frame is decoded, decode any following data

If DataLen > 0 Then

ReDim temp(response.Length - 36)

For rcount = 36 To (response.Length - 1)

temp(rcount - 36) = response(rcount)

Next

decodeExtras(temp, DataKind)

End If

Catch ex As Exception

MsgBox("Error while decoding AGWPE frame. Error = " & ex.Message)

End Try

End Sub

Private Sub decodeExtras(ByVal info() As Byte, ByVal datakind As Char)

Try

'decodes following information

Select Case datakind

Case Is = "R" 'Version Number

majVers = info(0) + (info(1) \* &H100)

minVers = info(4) + (info(5) \* &H100)

Case Is = "G" 'Port Info

nPorts = Microsoft.VisualBasic.Val(Microsoft.VisualBasic.Chr(info(0)))

If nPorts > 0 Then

ReDim PortInfo(nPorts - 1, 9)

Dim temp As String

Dim CurrentPort As Integer = 0

For rCount = 2 To (info.Length - 1)

temp = Microsoft.VisualBasic.Chr(info(rCount)) 'Gets next character in info string

If temp <> ";" Then 'If still in same section

PortInfo(CurrentPort, 0) = PortInfo(CurrentPort, 0) & temp 'Add next char to string

Else

CurrentPort = CurrentPort + 1 'If delimiter found, goto next port

If CurrentPort > nPorts - 1 Then 'if no more ports

Exit For 'stop reading

End If

End If

Next

End If

Case Is = "g" 'Port Capabilities

Dim temp As Int32

Dim PortNum As Integer = (frmMain.lstPorts.SelectedIndex - 1)

For rcount = 0 To 7

PortInfo(PortNum, rcount + 1) = info(rcount)

Next rcount

temp = info(8) + (info(9) \* &H100) + (info(10) \* &H10000) + (info(11) \* &H1000000)

PortInfo(PortNum, 9) = temp

Case Is = "U" 'Monitoring frame

frmMain.lstMessages.Items.Add(Encoding.ASCII.GetChars(info))

End Select

Catch ex As Exception

MsgBox("Error while decoding additional information following AGWPE frame. Error = " & ex.Message)

End Try

End Sub

End Class