```
AD590_Cal - 1

Dim AD590_Temp As Double

Private Sub Command2_Click()
Unload AD590_Cal
End Sub

Private Sub Form_Load()
AD590_Cal.Top = 1500
AD590_Cal.Left = 4000

Call Convert_EMF_to_Temperature(Aux_AD590_EMF, AD590_Temp)
Text1.Text = Format(AD590_Temp, "0.00")

End Sub
Private Sub Command1_Click()

If IsNumeric(Text1.Text) Then
  AD590_Temp = Text1.Text
  Call Convert_Temperature_to_EMF(AD590_Temp, Aux_AD590_EMF)
End If

Unload AD590_Cal
End Sub
```

Auxiliary_form - 1

```
Option Explicit
Dim Current_Auxiliary_Data_Packet As Long
Dim Aux_Time(29), Aux_Channel(29), Aux_value(29) As Long
Dim Number_Aux_Records, Word_Result As Long
Dim Aux_position() As Long
Dim File_Number As Integer

Private Sub Form_Load()
Dim i, Counter As Long

ReDim Aux_position(No.Aux_Science)
'Find position of Auxiliary data packets
Counter = 0
For i = 1 To No.Science_Packets
  If Science(i).Description = " Auxiliary Data   " Then
    Counter = Counter + 1
    Aux_position(Counter) = i
  End If
Next i

'Determine which Lander data packet is being viwed
For i = 1 To No.Aux_Science
  If Aux_position(i) = Current_Science_Packet Then
    Current_Auxiliary_Data_Packet = i
  End If
Next i

Label1.Caption = No.Aux_Science
VScroll1.Max = No.Aux_Science
VScroll1.value = Current_Auxiliary_Data_Packet

'Display the temperature assummed for the AD590
Dim x As Double
Call Convert_EMF_to_Temperature(Aux_AD590_EMF, x)
Label13.Caption = Format(x, "0.00")

Update_Auxiliary_Packet_Info
End Sub

Public Sub Load_Lander_Data()
Dim i As Integer
Dim Aux_start As Long
Dim Time1, Time2 As Long

File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #File_Number

Aux_start = Science(Aux_position(Current_Auxiliary_Data_Packet)).Start
Get_Word (Aux_start + 18)
Number_Aux_Records = Word_Result

For i = 1 To Number_Aux_Records
  Get_Word (Aux_start + i * 8 + 12)
  Time1 = Word_Result
  If Time1 > 32767 Then Time1 = -1
  Get_Word (Aux_start + i * 8 + 14)
  Time2 = Word_Result
  Aux_Time(i) = Time1 * 65536 + Time2
  Get_Word (Aux_start + i * 8 + 16)
  Aux_Channel(i) = Word_Result
  Get_Word (Aux_start + i * 8 + 18)
  Aux_value(i) = Word_Result
Next i
Close #File_Number

End Sub

Public Sub Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte, Low_Byte As Long

Get #File_Number, Byte_Position, First_Byte
Get #File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
```

```
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Word_Result = High_Byte * 256 + Low_Byte

End Sub

Public Sub Update_Auxiliary_Packet_Info()
Dim i As Integer

Select Case Packet_Format
  Case "CSS", "QM"
    Load_Lander_Data
  Case "GRM", "FM"
    Load_Lander_Data   'Exactly the same format for Ptolemy format
  Case "RAL"
    Load_RAL_Data
End Select

'Initialise the display area (picture1)
Picture1.Cls
Picture1.PSet (0, 50), RGB(255, 255, 255)
Picture1.Print "Component"
Picture1.PSet (1500, 50), RGB(255, 255, 255)
Picture1.Print "Reading"
Picture1.PSet (2300, 50), RGB(255, 255, 255)
Picture1.Print "Units"
Picture1.PSet (2900, 50), RGB(255, 255, 255)
Picture1.Print "Lander time(s)"
Picture1.PSet (4400, 50), RGB(255, 255, 255)
Picture1.Print "Chan. ID"
Picture1.PSet (5300, 50), RGB(255, 255, 255)
Picture1.Print "Word"
Picture1.PSet (6000, 50), RGB(255, 255, 255)
Picture1.Print "ADC Voltage(V)"

Dim voltage, unit As Variant
Dim SensorReading As Double
Dim SensorID As Long
Dim j As Integer

For i = 1 To Number_Aux_Records
  j = i * 220 + 200
  SensorID = Get_Sensor_ID(Aux_Channel(i))
  Picture1.PSet (0, j), RGB(255, 255, 255)
  Picture1.Print Sensor(SensorID).Name
  Picture1.PSet (4800 - TextWidth(Aux_Channel(i)), j), RGB(255, 255, 255)
  Picture1.Print Aux_Channel(i)
  Picture1.PSet (5700 - TextWidth(Aux_value(i)), j), RGB(255, 255, 255)
  Picture1.Print Aux_value(i)
  Picture1.PSet (3500 - TextWidth(Aux_Time(i)), j), RGB(255, 255, 255)
  Picture1.Print Aux_Time(i)
  voltage = ADC_Voltage(Aux_value(i))
  voltage = Format(voltage, "0.00000")
  Picture1.PSet (6900 - TextWidth(voltage), j), RGB(255, 255, 255)
  Picture1.Print voltage

'Compensate voltage for assummed AD590 temperature
  Select Case SensorID
    Case 1 To 21, 27
      voltage = voltage + Aux_AD590_EMF / 10000
  End Select

  Call Get_Sensor_Reading(SensorID, voltage, SensorReading, unit)
  Picture1.PSet (2100 - TextWidth(Format(SensorReading, "0.00")), j), RGB(255, 255, 255)
  Picture1.Print Format(SensorReading, "0.00")
  If unit = "oC" Then
    Picture1.PSet (2300, j - 50), RGB(255, 255, 255)
    Picture1.Print "o"
    Picture1.PSet (2385, j), RGB(255, 255, 255)
    Picture1.Print "C"
  Else
    Picture1.PSet (2300, j), RGB(255, 255, 255)
    Picture1.Print unit
```

```
Auxiliary_form - 3

  End If
Next i

Label6.Caption = Current_Auxiliary_Data_Packet
Label7.Caption = Science(Aux_position(Current_Auxiliary_Data_Packet)).Sequence_Count
Label8.Caption = Science(Aux_position(Current_Auxiliary_Data_Packet)).Time
Label10.Caption = Number_Aux_Records

End Sub



Private Sub VScroll1_Change()

  Current_Auxiliary_Data_Packet = VScroll1.value
  Update_Auxiliary_Packet_Info

End Sub

Public Function ADC_Voltage(x) As Double
Dim local_x As Double

  local_x = x
  If local_x > 32767 Then local_x = local_x - 65536
  ADC_Voltage = local_x * 10 / 32768
End Function

Public Function Get_Sensor_ID(ChanID)
  Dim Component, x As Integer
  Component = 0

  For x = 1 To Number_of_Sensors
    If Sensor(x).ADC_Channel = ChanID Then
      Component = x
    End If
  Next x
  Get_Sensor_ID = Component

End Function


Public Sub Load_RAL_Data()

End Sub
```

```
Calibration - 1

Option Explicit
Dim Change_calibration As Boolean
Dim response
Dim i, j, FileNumber As Integer

Private Sub Command1_Click()

If Change_calibration = True Then

  response = MsgBox("Are you sure that you want to change the pressure sensor calibration values",
vbYesNo)
  If response = vbYes Then
    For i = 1 To Number_Pressure_Sensors
      If IsNumeric(Text2(i).Text) Then
        Pressure_Sensor(i).Atmosphere_value = Text2(i).Text
      End If
      If IsNumeric(Text3(i).Text) Then
        Pressure_Sensor(i).Zero_value = Text3(i).Text
      End If
    Next i
  End If
'Store the data to Pressure sensors.txt file
  FileNumber = FreeFile
  Open Ptolemy_Directory & "\EGSE_info\Pressure Sensors.txt" For Output As FileNumber
  Print #FileNumber, Number_Pressure_Sensors
  For j = 1 To Number_Pressure_Sensors
    Print #FileNumber, Pressure_Sensor(j).Sensor_Number & ", ";
    Print #FileNumber, Pressure_Sensor(j).Atmosphere_value & ", ";
    Print #FileNumber, Pressure_Sensor(j).Zero_value
  Next j
  Close #FileNumber

End If
Unload Calibration
End Sub

Private Sub Command2_Click()
'Cancel button clicked need to reload original sensor values
FileNumber = FreeFile
Open Ptolemy_Directory & "\EGSE_info\Pressure Sensors.txt" For Input As FileNumber
Input #FileNumber, Number_Pressure_Sensors
For i = 1 To Number_Pressure_Sensors
  Input #FileNumber, Pressure_Sensor(i).Sensor_Number
  Input #FileNumber, Pressure_Sensor(i).Atmosphere_value
  Input #FileNumber, Pressure_Sensor(i).Zero_value
Next i
Close #FileNumber

Unload Calibration
End Sub

Private Sub Command3_Click()

response = MsgBox("Warning - modifying the pressure sensor calibration will alter the display of th
e pressure sensor values", 1, "")
If response = 1 Then
  Change_calibration = True
  For i = 1 To Number_Pressure_Sensors
    Text2(i).Enabled = True
    Text3(i).Enabled = True
  Next i
Else
  FileNumber = FreeFile
  Open Ptolemy_Directory & "\EGSE_info\Pressure Sensors.txt" For Input As FileNumber
  Input #FileNumber, Number_Pressure_Sensors
  For i = 1 To Number_Pressure_Sensors
    Input #FileNumber, Pressure_Sensor(i).Sensor_Number
    Input #FileNumber, Pressure_Sensor(i).Atmosphere_value
    Input #FileNumber, Pressure_Sensor(i).Zero_value
  Next i
  Close #FileNumber
End If

End Sub

Private Sub Command4_Click()
Dim spaces As Double
```

```
Calibration - 2


response = MsgBox("Warning - modifying the pressure sensor calibration will alter the display of th
e pressure sensor values", 1, "")
If response = 1 Then
  Change_calibration = True
  FileNumber = FreeFile
  Open Ptolemy_Directory & "\EGSE_info\Default_Pressure Sensors.txt" For Input As FileNumber
  Input #FileNumber, Number_Pressure_Sensors
  ReDim Pressure_Sensor(Number_Pressure_Sensors)
  For i = 1 To Number_Pressure_Sensors
    Input #FileNumber, Pressure_Sensor(i).Sensor_Number
    Input #FileNumber, Pressure_Sensor(i).Atmosphere_value
    Input #FileNumber, Pressure_Sensor(i).Zero_value

    If Abs(Pressure_Sensor(i).Atmosphere_value) > 1 Then
      spaces = 6 - 2 * Int(Log10(Abs(Pressure_Sensor(i).Atmosphere_value) + 0.000001))
    Else
      spaces = 6
    End If
    If Pressure_Sensor(i).Atmosphere_value < 0 Then
      spaces = spaces - 1
    End If
    Text2(i).Text = Space(spaces) & Format(Pressure_Sensor(i).Atmosphere_value, "#0.0000")

    If Abs(Pressure_Sensor(i).Zero_value) > 1 Then
      spaces = 6 - 2 * Int(Log10(Abs(Pressure_Sensor(i).Zero_value) + 0.000001))
    Else
      spaces = 6
    End If
    If Pressure_Sensor(i).Zero_value < 0 Then
      spaces = spaces - 1
    End If
    Text3(i).Text = Space(spaces) & Format(Pressure_Sensor(i).Zero_value, "#0.0000")

  Next i
  Close #FileNumber

End If

End Sub

Private Sub Form_Load()
'Display the pressure sensor information
Dim i As Integer
Dim spaces  As Double
Calibration.Top = 1500
Calibration.Left = 4000
Change_calibration = False

For i = 1 To Number_Pressure_Sensors
  Load Label3(i)
  Label3(i).Caption = Sensor(Pressure_Sensor(i).Sensor_Number).Name
  Label3(i).Top = Label3(0).Top + 400 * (i)
  Label3(i).Visible = True

  Load Text2(i)
  Text2(i).Top = Text2(0).Top + 400 * (i - 1)
  If Abs(Pressure_Sensor(i).Atmosphere_value) > 1 Then
    spaces = 6 - 2 * Int(Log10(Abs(Pressure_Sensor(i).Atmosphere_value) + 0.000001))
  Else
    spaces = 6
  End If
  If Pressure_Sensor(i).Atmosphere_value < 0 Then
    spaces = spaces - 1
  End If
  Text2(i).Text = Space(spaces) & Format(Pressure_Sensor(i).Atmosphere_value, "#0.0000")
  Text2(i).Visible = True

  Load Text3(i)
  Text3(i).Top = Text3(0).Top + 400 * (i - 1)
  If Abs(Pressure_Sensor(i).Zero_value) > 1 Then
    spaces = 6 - 2 * Int(Log10(Abs(Pressure_Sensor(i).Zero_value) + 0.000001))
  Else
    spaces = 6
  End If
  If Pressure_Sensor(i).Zero_value < 0 Then
    spaces = spaces - 1
```

```
Calibration - 3

  End If

  Text3(i).Text = Space(spaces) & Format(Pressure_Sensor(i).Zero_value, "#0.0000")
  Text3(i).Visible = True
Next i
End Sub
```

```
Concise_Form - 1

Option Explicit
Dim File_Number As Integer
Dim Viewing_Pkt, Word_Result As Long

Private Sub Command1_Click()
Dim Collecting As Boolean

Collecting = EGSE_MDI.Timer1.Enabled
  If Collecting Then
    EGSE_MDI.Timer1.Enabled = False
    Command1.Caption = "Continue"
  Else
    EGSE_MDI.Timer1.Enabled = True
    Command1.Caption = "Pause"
  End If

End Sub

Private Sub Form_Load()

  Form_Concise_Data_Show = True
  If Packet_Format <> "" Then
    File_Number = FreeFile
    Open Egse_Data_File_Name For Binary As File_Number
    Length_of_Egse_File = LOF(File_Number)
    Close #File_Number
  End If

  VScroll1.Max = No.Lander_Packets
  Viewing_Pkt = 1
  If No.Lander_Packets = 0 Then
    VScroll1.Max = 1
  End If
  VScroll1.value = 1


End Sub

Private Sub Form_Unload(Cancel As Integer)
  Form_Concise_Data_Show = False
End Sub

Private Sub VScroll1_Change()
  Display_282_Byte_Packet (VScroll1.value)
End Sub

Private Sub Display_282_Byte_Packet(Number)

Dim i As Integer
Dim Start_Position As Long
Dim Packet_ID, Sequence_Control, Packet_Length, flags As Long
Dim Lander_Time_seconds, Lander_time_fractions As Long
Dim Lander_Packet_Position As Long
Dim Packet_Type(7) As String

Viewing_Pkt = Number
File_Number = FreeFile

For i = 0 To 7
  Packet_Type(i) = ""
Next i

  Select Case Packet_Format
    Case "CSS", "QM"
      Open Egse_Data_File_Name For Binary As #File_Number
      i = 0
      Lander_Packet_Position = 23
      Do
        Start_Position = 282 * (Viewing_Pkt - 1) + Lander_Packet_Position
        Get_Word (Start_Position)
        Packet_ID = Word_Result
        Get_Word (Start_Position + 2)
        Sequence_Control = Word_Result
        flags = Word_Result And &HC000
        Sequence_Control = Sequence_Control - flags
        Get_Word (Start_Position + 4)
        Packet_Length = Word_Result + 7
```

```
        Lander_Time_seconds = 0
        Get_Word (Start_Position + 6)
        If Word_Result > 32767 Then Word_Result = 32767
        Lander_Time_seconds = Word_Result * 256 * 256
        Get_Word (Start_Position + 8)
        Lander_Time_seconds = Lander_Time_seconds + Word_Result
        Get_Word (Start_Position + 10)
        Lander_time_fractions = Word_Result

      Lander_Packet_Position = Lander_Packet_Position + Packet_Length

      Select Case Packet_ID
        Case &HF34    'Housekeeping packets
          Get_Word (Start_Position + 16)
          Packet_Type(i) = " Unknown  "
          If Word_Result = 1 Then Packet_Type(i) = " Concise HK "
          If Word_Result = 2 Then Packet_Type(i) = " Complete HK"

        Case &HF37    'Progress events (normal and warning)
          Get_Word (Start_Position + 14)

          Select Case Word_Result / 256
            Case 1
              Packet_Type(i) = " Normal Progress Event   "
            Case 2
              Packet_Type(i) = " Warning Anomalous Event "
            Case 10
              Packet_Type(i) = " NPE - Memory Checksum   "
            Case Else
              Packet_Type(i) = " Unknown - progress event"

          End Select

        Case &HF31    'Acceptance or failure TC or
          Get_Word (Start_Position + 14)
          Select Case Word_Result / 256
            Case 1
              Packet_Type(i) = " Acceptance Success      "
            Case 2
              Packet_Type(i) = " Acceptance Failure      "
            Case Else
              Packet_Type(i) = " Unknown - acceptance    "
          End Select

        Case &HF39    'Ptolemy memory dump
          Packet_Type(i) = " Memory Dump "

        Case &HF3C    'Ptolemy Science Data
          Get_Word (Start_Position + 16)
          Packet_Type(i) = " Unknown          "
          If Word_Result = 1 Then Packet_Type(i) = " Auxiliary Data   "
          If Word_Result = 2 Then Packet_Type(i) = " GC spectrum      "
          If Word_Result = 3 Then Packet_Type(i) = " Isotope spectrum "

        Case &H0
'If the packet_ID is empty then move to end of lander packet
           Packet_Type(i) = " Empty "

        Case Else
'If the packet_ID is unknown then move to end of lander packet
           Packet_Type(i) = " Unknown "
      End Select
      i = i + 1
      Loop Until Lander_Packet_Position > 250 Or i = 8

      Close #File_Number


    Case "GRM", "FM"
      Open Egse_Data_File_Name For Binary As #File_Number
      i = 0
      Lander_Packet_Position = 23
      Do
        Start_Position = 280 * (Viewing_Pkt - 1) + Lander_Packet_Position
        Get_Word (Start_Position)
        Packet_ID = Word_Result
        Get_Word (Start_Position + 2)
```

```
        Sequence_Control = Word_Result
        flags = Word_Result And &HC000
        Sequence_Control = Sequence_Control - flags
        Get_Word (Start_Position + 4)
        Packet_Length = Word_Result + 7
        Lander_Time_seconds = 0
        Get_Word (Start_Position + 6)
        If Word_Result > 32767 Then Word_Result = 32767
        Lander_Time_seconds = Word_Result * 256 * 256
        Get_Word (Start_Position + 8)
        Lander_Time_seconds = Lander_Time_seconds + Word_Result
        Get_Word (Start_Position + 10)
        Lander_time_fractions = Word_Result

      Lander_Packet_Position = Lander_Packet_Position + Packet_Length

      Select Case Packet_ID
        Case &HF34    'Housekeeping packets
          Get_Word (Start_Position + 16)
          Packet_Type(i) = " Unknown  "
          If Word_Result = 1 Then Packet_Type(i) = " Concise HK "
          If Word_Result = 2 Then Packet_Type(i) = " Complete HK"

        Case &HF37    'Progress events (normal and warning)
          Get_Word (Start_Position + 14)
          Select Case Word_Result / 256
            Case 1
              Packet_Type(i) = " Normal Progress Event   "
            Case 2
              Packet_Type(i) = " Warning Anomalous Event "
            Case Else
              Packet_Type(i) = " Unknown - progress event"

          End Select

        Case &HF31    'Acceptance or failure TC or
          Get_Word (Start_Position + 14)
          Select Case Word_Result / 256
            Case 1
              Packet_Type(i) = " Acceptance Success      "
            Case 2
              Packet_Type(i) = " Acceptance Failure      "
            Case 10
              Packet_Type(i) = " NPE - Memory Checksum   "
            Case Else
              Packet_Type(i) = " Unknown - acceptance    "
          End Select

        Case &HF39    'Ptolemy memory dump
          Packet_Type(i) = " Memory Dump "

        Case &HF3C    'Ptolemy Science Data
          Get_Word (Start_Position + 16)
          Packet_Type(i) = " Unknown          "
          If Word_Result = 1 Then Packet_Type(i) = " Auxiliary Data   "
          If Word_Result = 2 Then Packet_Type(i) = " GC spectrum      "
          If Word_Result = 3 Then Packet_Type(i) = " Isotope spectrum "

        Case &H0
'If the packet_ID is empty then move to end of lander packet
           Packet_Type(i) = " Empty "

        Case Else
'If the packet_ID is unknown then move to end of lander packet
           Packet_Type(i) = " Unknown "
      End Select
      i = i + 1
      Loop Until Lander_Packet_Position > 250 Or i = 8

      Close #File_Number

    Case "RAL"
  End Select

  For i = 0 To 7
    Label16(i).Caption = Packet_Type(i)
  Next i
```

```
  Label15.Caption = VScroll1.value
End Sub

Public Sub Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte, Low_Byte As Long

If Byte_Position < 0 Then
  Word_Result = 0
  Exit Sub
End If

Get #File_Number, Byte_Position, First_Byte
Get #File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Word_Result = High_Byte * 256 + Low_Byte
End Sub

Public Sub Update_Concise_Form()
  VScroll1.Max = No.Lander_Packets
  Viewing_Pkt = No.Lander_Packets
  VScroll1.value = No.Lander_Packets
End Sub
```

```
Option Explicit
Dim FileNumber As Integer

Private Sub mnuAuxAD590Temp_Click()
AD590_Cal.Show
AD590_Cal.ZOrder (0)
End Sub

Private Sub mnuCalibrate_Pressure_Sensors_Click()
Calibration.Show
Calibration.ZOrder (0)
End Sub

Private Sub mnuDisplayByteReverse_Click()
'Byte_Reverse is used to see if the high and low bytes
'of a word should be swopped.

If Byte_Reverse = True Then
  Byte_Reverse = False
  mnuDisplayByteReverse.Checked = False
Else
  Byte_Reverse = True
  mnuDisplayByteReverse.Checked = True
End If

Call Reload_Active_Forms

End Sub

Private Sub mnuDisplayConciseData_Click()
Concise_Form.Show
Concise_Form.ZOrder (0)
End Sub

Private Sub mnuDisplayRawData_Click()
RawData.Show
RawData.ZOrder (0)
End Sub

Private Sub mnuDisplayRealTimeDataCollection_Click()

If mnuDisplayRealTimeDataCollection.Checked = False Then
  Egse_Data_File_Name = "c:\Documents and Settings\dja333\Desktop\RAL - CDMS\tmlog.dat"
  Packet_Format = "CSS"

  EGSE_MDI.Caption = "Ptolemy EGSE             File loaded - " & Egse_Data_File_Name

  Call Initialise_Packet_Numbers
  Call Determine_Number_of_Packets(No.Lander_Packets)
  Packet_Format = "CSS"
  Call Load_282_byte_Packet_Information
  Call Load_Sensor_Data
  Call Reload_Active_Forms
  Call Update_Old_Packet_Numbers

  Timer1.Enabled = True
  mnuDisplayRealTimeDataCollection.Checked = True
Else
  Timer1.Enabled = False
  mnuDisplayRealTimeDataCollection.Checked = False
End If

End Sub

Private Sub mnuDisplaySUmmary_Click()
Summary.Show
Summary.ZOrder (0)
End Sub


Private Sub mnuExit_Click()
'This routine will end the program session

End

End Sub
```

```
Private Sub mnuExportDockingStationCalibration_Click()
Dim Responce
If Egse_Data_File_Name <> "" Then
  Export_DS_Cal.Show
  Export_DS_Cal.Left = 4000
  Export_DS_Cal.Top = 1000
Else
  Responce = MsgBox("Please load an EGSE data file")
End If
End Sub

Private Sub mnuExportHousekeeping_Click()
Dim Responce
If Egse_Data_File_Name <> "" Then
  Export_HK.Show
  Export_HK.Left = 4000
  Export_HK.Top = 1000
Else
  Responce = MsgBox("Please load an EGSE data file")
End If

End Sub

Private Sub mnuExportScienceAuxilliary_Click()
Dim Responce
If Egse_Data_File_Name <> "" Then
  Export_Aux.Show
  Export_Aux.Left = 4000
  Export_Aux.Top = 1000
Else
  Responce = MsgBox("Please load an EGSE data file")
End If

End Sub

Private Sub mnuExportScienceSpectra_Click()
Dim Responce
Dim Number_spectra As Long
Dim show_form As Boolean

Number_spectra = No.Complete_Science_Spectra + No.Compact_Science
show_form = True
If Egse_Data_File_Name = "" Then
  Responce = MsgBox("Please load an EGSE data file")
  show_form = False
End If

If Number_spectra < 1 And show_form = True Then
  Responce = MsgBox("The EGSE data file contains no spectra")
  show_form = False
End If

If show_form = True Then
  Export_Spectra.Show
  Export_Spectra.Left = 4000
  Export_Spectra.Top = 1000
End If

End Sub

Private Sub mnuFileLoad_CSS_Click()
Packet_Format = "CSS"
Open_EGSE_File
End Sub

Private Sub mnuGeneraltests_Click()
TestForm.Show
End Sub

Private Sub mnuLoadFile_FM_Click()
Packet_Format = "FM"
Open_EGSE_File
End Sub

Private Sub mnuLoadFile_GRM_Click()
Packet_Format = "GRM"
Open_EGSE_File
```

```
EGSE_MDI - 3

End Sub

Public Sub Open_EGSE_File()

CommonDialog1.CancelError = True
On Error GoTo ErrorHandler      'Enable error detection
CommonDialog1.ShowOpen
Egse_Data_File_Name = CommonDialog1.FileName

'Check to see if the file opened is a valid file
FileNumber = FreeFile
Open CommonDialog1.FileName For Input As #FileNumber
Close #FileNumber
On Error GoTo 0                 'Turn off error handler

EGSE_MDI.Caption = "Ptolemy EGSE  --- " & Packet_Format & "         File loaded - " & Egse_Data_Fil
e_Name
'Set number of various packets to 0

Call Initialise_Packet_Numbers

Call Determine_Number_of_Packets(No.Lander_Packets)
If No.Lander_Packets > 0 Then
  Select Case Packet_Format
    Case "CSS", "QM"
      Call Load_282_byte_Packet_Information
    Case "GRM", "FM"
      Call Load_280_byte_Packet_Information
    Case "RAL"
      Call Load_RAL_Packet_Information
  End Select
End If

'Reload any other active forms
Call Load_Sensor_Data
Call Reload_Active_Forms
Call Update_Old_Packet_Numbers

' if real time data collection is active, then
' turn it off

Timer1.Enabled = False
mnuDisplayRealTimeDataCollection.Checked = False

Exit Sub

ErrorHandler:                   'Error detection routine

Select Case Err.Number
  Case 32755                    'Cancel button pressed then
                                'just exit sub
  Case 55
    Close #FileNumber

  Case Else                     'else explain cause of error
    MsgBox Err.Description & "; Error number " & Err.Number '

End Select

End Sub

Private Sub mnuLoadFile_QM_Click()
Packet_Format = "QM"
Open_EGSE_File
End Sub

Private Sub mnuLoadFile_RalFormat_Click()
Packet_Format = "RAL"
Open_EGSE_File
End Sub

Private Sub mnuShortTest_Click()
  ShortMessageLoad.Show
End Sub

Private Sub mnuTest_Cruise_Phase_mode_Click()
  Test_Type = "Cruise Phase"
```

```
  ShortMessageLoad.Show
End Sub

Private Sub mnuTest_EAFT_mode_Click()
  Test_Type = "EAFT"
  ShortMessageLoad.Show
End Sub

Private Sub Timer1_Timer()
Dim New_packets As Long

Call Determine_Number_of_Packets(New_packets)

If New_packets > No.Lander_Packets Then
  No.Lander_Packets = New_packets
  Select Case Packet_Format
    Case "CSS", "QM"
      Call Load_282_byte_Packet_Information
    Case "GRM", "FM"
      Call Load_280_byte_Packet_Information
    Case "RAL"
      Call Load_RAL_Packet_Information
  End Select

  Call Load_Sensor_Data
  Call Update_Active_Forms
  Call Update_Old_Packet_Numbers
End If

End Sub
```

Export_Aux - 1

```
Option Explicit
Dim File_Export_Name As String
Dim Data_File_Number, Export_File_Number, i, j, k As Integer
Dim Word_Result As Long

Private Sub Check2_Click()
If Check2.value = 1 Then
  Text1.Text = 1
  Text2.Text = No.Science_Packets
  Text1.Enabled = False
  Text2.Enabled = False
Else
  Text1.Enabled = True
  Text2.Enabled = True
End If


End Sub

Private Sub Command1_Click()
Dim Aux_start, Number_Aux_Records As Long
Dim Aux_Channel, Aux_value As Long
Dim Aux_voltage, Aux_AD590_EMF, Aux_reading As Double
Dim Aux_String As String
Dim Time1, Time2, Aux_Time As Long
Dim sensor_used As Integer
Dim Sensors_Used(40) As Boolean
Dim dummy

Export_File_Number = FreeFile
Data_File_Number = FreeFile + 1

'If using AD590 correction then calculate AD590 EMF
Dim AD590_Temp, AD590_EMF As Double
If IsNumeric(Text3.Text) Then
  AD590_Temp = Text3.Text
  Call Convert_Temperature_to_EMF(AD590_Temp, AD590_EMF)
Else
  AD590_Temp = 0
  AD590_EMF = 0
End If

'Check that the text values for the Science packet range
'is valid
Dim Sci_valid As Boolean
Sci_valid = True
If IsNumeric(Text1.Text) = False Then Sci_valid = False
If IsNumeric(Text2.Text) = False Then Sci_valid = False
If Sci_valid = True Then
  If Text1.Text < 1 Then Sci_valid = False
  If Text1.Text > No.Science_Packets Then Sci_valid = False
  If Text1.Text <> Int(Text1.Text) Then Sci_valid = False
  If Text2.Text < 1 Then Sci_valid = False
  If Text2.Text > No.Science_Packets Then Sci_valid = False
  If Text2.Text <> Int(Text2.Text) Then Sci_valid = False
  If Int(Text1.Text) > Int(Text2.Text) Then Sci_valid = False
End If
Dim Responce
If Sci_valid = False Then
  Responce = MsgBox("Science packet ranges not valid")
  Text1.Text = 1
  Text2.Text = No.Science_Packets
  Exit Sub
End If

CommonDialog1.CancelError = True
On Error GoTo ErrorHandler      'Enable error detection
CommonDialog1.ShowSave
File_Export_Name = CommonDialog1.FileName

'Check to see if the file opened is a valid file
Open CommonDialog1.FileName For Output As #Export_File_Number
Close #Export_File_Number
On Error GoTo 0                 'Turn off error handler

'Determine which sensors have been measured
Open Egse_Data_File_Name For Binary As #Data_File_Number
```

```
Export_Aux - 2

For i = Text1.Text To Text2.Text
  If Science(i).Description = " Auxiliary Data   " Then
    Aux_start = Science(i).Start
    Get_Word (Aux_start + 18)
    Number_Aux_Records = Word_Result
    For j = 1 To Number_Aux_Records
      Get_Word (Aux_start + j * 8 + 16)
      Aux_Channel = Word_Result
      sensor_used = Get_Sensor_ID(Aux_Channel)
      Sensors_Used(sensor_used) = True
    Next j
  End If
Next i
Close #Data_File_Number

'Export the data to file

Dim Export_Title As String
If Option4.value = True Then Export_Title = "Auxiliary Science - Raw data - ADC values"
If Option5.value = True Then Export_Title = "Auxiliary Science - Raw data - Voltage"
If Option6.value = True Then Export_Title = "Auxiliary Science - Calibrated Data (no AD590 correcti
on)"
If Option7.value = True Then Export_Title = "Auxiliary Science - Calibrated Data with AD590 correct
ion of " & AD590_Temp & " oC"

Open CommonDialog1.FileName For Output As #Export_File_Number
Open Egse_Data_File_Name For Binary As #Data_File_Number

Dim Big_String As String
Dim Deliminator_Type As Long
'Dim ADC_value As Double
'Dim AD590_EMF_Correction As Double

If Option1.value = True Then Deliminator_Type = 9
If Option2.value = True Then Deliminator_Type = 44
If Option3.value = True Then Deliminator_Type = 59

Print #Export_File_Number, Export_Title
'Output sensor headings, this is the complex bit that
'does all the calculations
  Big_String = "Sci number" & Chr$(Deliminator_Type) & "Lander time" & Chr$(Deliminator_Type)
  For j = 1 To Number_of_Sensors
    If Sensors_Used(j) = True Then
      Big_String = Big_String & Sensor(j).Name & Chr$(Deliminator_Type)
    End If
  Next j
  Print #Export_File_Number, Big_String

'Output sensor units
  Big_String = "     " & Chr$(Deliminator_Type) & "seconds" & Chr$(Deliminator_Type)
  For j = 1 To Number_of_Sensors
    If Sensors_Used(j) = True Then
      Big_String = Big_String & Sensor(j).unit & Chr$(Deliminator_Type)
    End If
  Next j
  Print #Export_File_Number, Big_String
  Print #Export_File_Number,

'Output sensor values
For i = Text1.Text To Text2.Text
  If Science(i).Description = " Auxiliary Data   " Then
    Aux_start = Science(i).Start
    Get_Word (Aux_start + 18)
    Number_Aux_Records = Word_Result
    For j = 1 To Number_Aux_Records
      Get_Word (Aux_start + j * 8 + 12)
      Time1 = Word_Result
      If Time1 > 32767 Then Time1 = -1
      Get_Word (Aux_start + j * 8 + 14)
      Time2 = Word_Result
      Aux_Time = Time1 * 65536 + Time2
      Get_Word (Aux_start + j * 8 + 16)
      Aux_Channel = Word_Result
      Get_Word (Aux_start + j * 8 + 18)
      Aux_value = Word_Result
      Aux_voltage = ADC_Voltage(Aux_value)
      sensor_used = Get_Sensor_ID(Aux_Channel)
```

```
      Big_String = i & Chr$(Deliminator_Type) & Aux_Time & Chr$(Deliminator_Type)
      For k = 1 To Number_of_Sensors
        If Sensors_Used(k) = True Then
          If k = sensor_used Then
            Select Case True
              Case Option4   ' Raw ADC data
                Aux_String = Aux_value
              Case Option5   ' Raw ADC voltage
                Aux_String = Aux_voltage
              Case Option6   ' sensor readings not calibrated for AD590 temperature
                Call Get_Sensor_Reading(k, Aux_voltage, Aux_reading, dummy)
                Aux_String = Aux_reading
              Case Option7   ' sensor readings calibrated for AD590 temperature
                Select Case k
                  Case 1 To 21, 27 ' Add AD590 EMF to thermocouple sensors
                    Aux_voltage = Aux_voltage + AD590_EMF / 10000
                End Select
                Call Get_Sensor_Reading(k, Aux_voltage, Aux_reading, dummy)
                Aux_String = Aux_reading
            End Select
            Big_String = Big_String & Aux_String & Chr$(Deliminator_Type)
          Else
            Big_String = Big_String & Chr$(Deliminator_Type)
          End If
        End If
      Next k
      Print #Export_File_Number, Big_String
    Next j
  End If
Next i

Close #Export_File_Number
Close #Data_File_Number

Exit Sub

ErrorHandler:                   'Error detection routine

Select Case Err.Number
  Case 32755                    'Cancel button pressed then
                                'just exit sub

  Case 55
    Close #Export_File_Number

  Case Else                     'else explain cause of error
    MsgBox Err.Description & "; Error number " & Err.Number '

End Select

End Sub

Private Sub Command2_Click()
Unload Export_Aux
End Sub

Private Sub Form_Load()
Option1.value = True
Option4.value = True
Text1.Text = 1
Text2.Text = No.Science_Packets
Check2.value = 1

End Sub

Public Function ADC_Voltage(x) As Double
Dim local_x As Double

  local_x = x
  If local_x > 32767 Then local_x = local_x - 65536
  ADC_Voltage = local_x * 10 / 32768
End Function

Public Sub Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
```

```
Export_Aux - 4

Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte, Low_Byte As Long

Get #Data_File_Number, Byte_Position, First_Byte
Get #Data_File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Word_Result = High_Byte * 256 + Low_Byte

End Sub
Public Function Get_Sensor_ID(ChanID)
  Dim Component, x As Integer
  Component = 0

  For x = 1 To Number_of_Sensors
    If Sensor(x).ADC_Channel = ChanID Then
      Component = x
    End If
  Next x
  Get_Sensor_ID = Component

End Function


Private Sub Option4_Click()
  Text3.Enabled = False
End Sub

Private Sub Option5_Click()
  Text3.Enabled = False
End Sub

Private Sub Option6_Click()
  Text3.Enabled = False
End Sub

Private Sub Option7_Click()
  Text3.Enabled = True
End Sub
```

```
Export_DS_Cal - 1

Option Explicit

Dim File_Export_Name As String
Dim Total_Number_DS_Cal_Packets As Long
Dim Info_packet_Numbers() As Long
Dim Export_File_Number, Data_File_Number As Integer

Private Sub Check2_Click()
If Check2.value = 1 Then
  Text1.Text = 1
  Text2.Text = Total_Number_DS_Cal_Packets
  Text1.Enabled = False
  Text2.Enabled = False
Else
  Text1.Enabled = True
  Text2.Enabled = True
End If

End Sub

Private Sub Command1_Click()
Dim i, j, k As Integer

Export_File_Number = FreeFile
Data_File_Number = FreeFile + 1

'Check that the text box range is valid
  Dim Text_Valid As Boolean
  Text_Valid = True
  If IsNumeric(Text1.Text) = False Then Text_Valid = False
  If IsNumeric(Text2.Text) = False Then Text_Valid = False
  If Text_Valid = True Then
    If Text1.Text < 1 Then Text_Valid = False
    If Text1.Text > Total_Number_DS_Cal_Packets Then Text_Valid = False
    If Text1.Text <> Int(Text1.Text) Then Text_Valid = False
    If Text2.Text < 1 Then Text_Valid = False
    If Text2.Text > Total_Number_DS_Cal_Packets Then Text_Valid = False
    If Text2.Text <> Int(Text2.Text) Then Text_Valid = False
    If Int(Text1.Text) > Int(Text2.Text) Then Text_Valid = False
  End If
  Dim Responce
  If Text_Valid = False Then
    Responce = MsgBox("DS calibration ranges not valid")
    Text1.Text = 1
    Text2.Text = Total_Number_DS_Cal_Packets
    Exit Sub
  End If

  CommonDialog1.CancelError = True
  On Error GoTo ErrorHandler      'Enable error detection
  CommonDialog1.ShowSave
  File_Export_Name = CommonDialog1.FileName

'Check to see if the file opened is a valid file
  Open CommonDialog1.FileName For Output As #Export_File_Number
  On Error GoTo 0                 'Turn off error handler

  Open Egse_Data_File_Name For Binary As #Data_File_Number

'  Export the data to excel
  Dim Big_String, Big_String2, dummy As String
  Dim Deliminator_Type As Long

  If Option1.value = True Then Deliminator_Type = 9
  If Option2.value = True Then Deliminator_Type = 44
  If Option3.value = True Then Deliminator_Type = 59

  Print #Export_File_Number, "Docking station calibration data acquired from " & Egse_Data_File_Nam
e
  If Option4.value = True Then
'   Export in raw data format
    Print #Export_File_Number, "Data in sensor ADC values"
    Print #Export_File_Number, ""
    Big_String = "Time" & Chr$(Deliminator_Type) & "ADC value"
  End If

  If Option5.value = True Then
```

```
Export_DS_Cal - 2

'   Export in raw data format
    Print #Export_File_Number, "Data in sensor voltages (V)"
    Print #Export_File_Number, ""
    Big_String = "Time" & Chr$(Deliminator_Type) & "Voltage"
  End If

  If Option4.value = True Then
'   Export in raw data format
    Print #Export_File_Number, "Data in calibrated position (mm)"
    Print #Export_File_Number, ""
    Big_String = "Time" & Chr$(Deliminator_Type) & "Position"
  End If

  Print #Export_File_Number, Big_String
  Print #Export_File_Number, ""

'Export the data
  Dim Start_time, Time1, Time2, Fractional_time, Measured_time As Double
  Dim Measured_value, Time_increment, Position As Double
  Dim x, Last_word As Long
  For i = Text1.Text To Text2.Text
    x = Info_packet_Numbers(i)
    Time2 = Information(x).Time
    Fractional_time = Get_Word(Information(x).Start + 10)
    Fractional_time = Fractional_time / 65536
    Time2 = Time2 + Fractional_time
    Last_word = Get_Word(Information(x).Start + 62)
    If i = Text1.Text Then
      Time_increment = 0.4552
      Start_time = Time2 - Time_increment * 23
    Else
      Time1 = Information(Info_packet_Numbers(i - 1)).Time
      Fractional_time = Get_Word(Information(Info_packet_Numbers(i - 1)).Start + 10)
      Fractional_time = Fractional_time / 65536
      Time1 = Time1 + Fractional_time
      If Last_word = 0 Then
        Time_increment = 0.4552
      Else
        Time_increment = (Time2 - Time1) / 23
      End If
      Start_time = Time1
    End If
    For j = 0 To 22
      Measured_time = Start_time + j * Time_increment
      Big_String = Format(Measured_time, "0.00")
      Measured_value = Get_Word(Information(x).Start + 18 + j * 2)

      If Option4.value = True Then
        Big_String = Big_String & Chr$(Deliminator_Type) & Measured_value
      End If

      If Option5.value = True Then
        Measured_value = Measured_value * 20 / 65535
        Big_String = Big_String & Chr$(Deliminator_Type) & Format(Measured_value, "0.000")
      End If

      If Option6.value = True Then
        Measured_value = Measured_value * 20 / 65535
        Call Get_Sensor_Reading(29, Measured_value, Position, dummy)
        Big_String = Big_String & Chr$(Deliminator_Type) & Format(Position, "0.000")
    End If
      Print #Export_File_Number, Big_String
    Next j
  Next i
  Close #Data_File_Number
  Close #Export_File_Number

Exit Sub

ErrorHandler:                   'Error detection routine

Select Case Err.Number
  Case 32755                    'Cancel button pressed then
                                'just exit sub

  Case 55
    Close #Export_File_Number
```

```
  Case Else                     'else explain cause of error
    MsgBox Err.Description & "; Error number " & Err.Number '

End Select
End Sub

Private Sub Command2_Click()
  Unload Export_DS_Cal
End Sub

Private Sub Form_Load()

' Determine the number of DS calibration sub-packets
  Dim i, Subtype, Event_ID As Long
  ReDim Info_packet_Numbers(No.Information_Packets)

  Data_File_Number = FreeFile
  Open Egse_Data_File_Name For Binary As #Data_File_Number

  Total_Number_DS_Cal_Packets = 0
  For i = 1 To No.Information_Packets
    If Information(i).Description = " Normal Progress Event   " Then
      Subtype = Get_Word(Information(i).Start + 14)
      Event_ID = Get_Word(Information(i).Start + 16)
      If Subtype = &H100 And Event_ID = 55114 Then
        Total_Number_DS_Cal_Packets = Total_Number_DS_Cal_Packets + 1
        Info_packet_Numbers(Total_Number_DS_Cal_Packets) = i
      End If
    End If
  Next i

  Close #Data_File_Number

  Option1.value = True
  Option4.value = True
  Text1.Text = 1
  Text2.Text = Total_Number_DS_Cal_Packets

  If Total_Number_DS_Cal_Packets = 0 Then
    Text1.Text = 0
    Command1.Enabled = False
  End If

End Sub

Public Function Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte, Low_Byte As Long

Get #Data_File_Number, Byte_Position, First_Byte
Get #Data_File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Get_Word = High_Byte * 256 + Low_Byte

End Function
```

Export_HK - 1

```
Option Explicit
Dim File_Export_Name As String
Dim FileNumber, i, j As Integer

Private Sub Check2_Click()
If Check2.value = 1 Then
  Text1.Text = 1
  Text2.Text = No.Housekeeping_Packets
  Text1.Enabled = False
  Text2.Enabled = False
Else
  Text1.Enabled = True
  Text2.Enabled = True
End If

End Sub

Private Sub Command1_Click()

'Text that the Housekeeping packet range is valid
Dim HK_valid As Boolean
HK_valid = True
If IsNumeric(Text1.Text) = False Then HK_valid = False
If IsNumeric(Text2.Text) = False Then HK_valid = False
If HK_valid = True Then
  If Text1.Text < 1 Then HK_valid = False
  If Text1.Text > No.Housekeeping_Packets Then HK_valid = False
  If Text1.Text <> Int(Text1.Text) Then HK_valid = False
  If Text2.Text < 1 Then HK_valid = False
  If Text2.Text > No.Housekeeping_Packets Then HK_valid = False
  If Text2.Text <> Int(Text2.Text) Then HK_valid = False
  If Int(Text1.Text) > Int(Text2.Text) Then HK_valid = False
End If
Dim Responce
If HK_valid = False Then
  Responce = MsgBox("Housekeeping packet ranges not valid")
  Text1.Text = 1
  Text2.Text = No.Housekeeping_Packets
  Exit Sub
End If

CommonDialog1.CancelError = True
On Error GoTo ErrorHandler      'Enable error detection
CommonDialog1.ShowSave
File_Export_Name = CommonDialog1.FileName

'Check to see if the file opened is a valid file
FileNumber = FreeFile
Open CommonDialog1.FileName For Output As #FileNumber
Close #FileNumber
On Error GoTo 0                 'Turn off error handler

'Export the data to file

Dim Export_Title As String
If Option4.value = True Then Export_Title = "Housekeeping - Raw data"
If Option5.value = True Then Export_Title = "Housekeeping - Calibrated Data (no AD590 correction)"
If Option6.value = True Then Export_Title = "Housekeeping - Calibrated Data (with AD590 correction)
"

Open CommonDialog1.FileName For Output As #FileNumber

Dim Big_String As String
Dim Deliminator_Type As Long
Dim ADC_value As Double
Dim AD590_EMF_Correction As Double

If Option1.value = True Then Deliminator_Type = 9
If Option2.value = True Then Deliminator_Type = 44
If Option3.value = True Then Deliminator_Type = 59

Print #FileNumber, Export_Title
'Output sensor headings
  Big_String = "HK number" & Chr$(Deliminator_Type) & "Lander time" & Chr$(Deliminator_Type)
  For j = 1 To Number_of_Sensors
    If Check1(j).value = 1 Then
      Big_String = Big_String & Sensor(j).Name & Chr$(Deliminator_Type)
```

```
    End If
  Next j
  Print #FileNumber, Big_String

'Output sensor units
  Big_String = "     " & Chr$(Deliminator_Type) & "seconds" & Chr$(Deliminator_Type)
  For j = 1 To Number_of_Sensors
    If Check1(j).value = 1 Then
      Big_String = Big_String & Sensor(j).unit & Chr$(Deliminator_Type)
    End If
  Next j
  Print #FileNumber, Big_String
  Print #FileNumber,


'Output sensor values
For i = Text1.Text To Text2.Text
  Big_String = i & Chr$(Deliminator_Type) & Housekeeping(i).Time & Chr$(Deliminator_Type)
  For j = 1 To Number_of_Sensors
    If Check1(j).value = 1 Then
      If Option4.value = True Then    ' Raw data
        Big_String = Big_String & Sensor_Bytes(j, i) & Chr$(Deliminator_Type)
      End If

      If Option5.value = True Then    ' Calibrated data, but with no AD590 correction
        Sensor(j).Byte_Value = Sensor_Bytes(j, i)
' not all byte values over 127 correspond to a negative voltage
        If Sensor(j).Byte_Value > 127 Then
          Select Case j
            Case 28, 29   'do nothing
            Case 32 To 36
            If Sensor(j).Byte_Value > 210 Then
              Sensor(j).Byte_Value = Sensor(j).Byte_Value - 256
            End If
            Case Else
              Sensor(j).Byte_Value = Sensor(j).Byte_Value - 256
          End Select
        End If
        ADC_value = Sensor(j).Byte_Value * 2 ^ Sensor(j).Bit_Shift
        Sensor(j).voltage = ADC_value * 10 / 32768
        Call Get_Sensor_Reading(j, Sensor(j).voltage, Sensor(j).Reading, Sensor(j).unit)
        Big_String = Big_String & Sensor(j).Reading & Chr$(Deliminator_Type)
      End If

      If Option6.value = True Then    ' Calibrated data, with AD590 correction
'Calculate temperature of AD590 (sensor number 28)
        Call Get_Sensor_Reading(28, Sensor(28).voltage, Sensor(28).Reading, Sensor(28).unit)
        Call Convert_Temperature_to_EMF(Sensor(28).Reading, AD590_EMF_Correction)
        AD590_EMF_Correction = AD590_EMF_Correction / 1000000

        Sensor(j).Byte_Value = Sensor_Bytes(j, i)
' not all byte values over 127 correspond to a negative voltage
        If Sensor(j).Byte_Value > 127 Then
          Select Case j
            Case 28, 29   'do nothing
            Case 32 To 36
            If Sensor(j).Byte_Value > 210 Then
              Sensor(j).Byte_Value = Sensor(j).Byte_Value - 256
            End If
            Case Else
              Sensor(j).Byte_Value = Sensor(j).Byte_Value - 256
          End Select
        End If
        ADC_value = Sensor(j).Byte_Value * 2 ^ Sensor(j).Bit_Shift
        Sensor(j).voltage = ADC_value * 10 / 32768
        Select Case j
          Case 1 To 21, 27  'Thermocouples, need to calibrate for effect of cold junction
                            ' and gain circuit
            Sensor(j).voltage = Sensor(j).voltage + AD590_EMF_Correction * 100
            Call Get_Sensor_Reading(j, Sensor(j).voltage, Sensor(j).Reading, Sensor(j).unit)
          Case Else
            Call Get_Sensor_Reading(j, Sensor(j).voltage, Sensor(j).Reading, Sensor(j).unit)
        End Select
        Big_String = Big_String & Sensor(j).Reading & Chr$(Deliminator_Type)
      End If

    End If
```

```
  Next j
  Print #FileNumber, Big_String
Next i

Close #FileNumber

Exit Sub

ErrorHandler:                    'Error detection routine

Select Case Err.Number
  Case 32755                     'Cancel button pressed then
                                 'just exit sub

  Case 55
    Close #FileNumber

  Case Else                      'else explain cause of error
    MsgBox Err.Description & "; Error number " & Err.Number '

End Select

End Sub

Private Sub Command2_Click()
Unload Export_HK
End Sub

Private Sub Form_Load()
Option1.value = True
Option4.value = True
Text1.Text = 1
Text2.Text = No.Housekeeping_Packets
Check2.value = 1
End Sub
```

Export_Spectra - 1

```
Option Explicit
Dim Total_Number_Spectra As Long
Dim File_Export_Name As String
Dim Spectra_Bins(), Bin_Counter() As Long
Dim Export_File_Number, Data_File_Number As Integer

Private Sub Check2_Click()
If Check2.value = 1 Then
  Text1.Text = 1
  Text2.Text = Total_Number_Spectra
  Text1.Enabled = False
  Text2.Enabled = False
Else
  Text1.Enabled = True
  Text2.Enabled = True
End If

End Sub

Private Sub Command1_Click()
Dim i, j, k As Integer
Dim Spectrum_Start_packet() As Long

ReDim Spectrum_Start_packet(Total_Number_Spectra, 10)

Export_File_Number = FreeFile
Data_File_Number = FreeFile + 1

' Find all the positions of the Complete spectra and GC spectra
  Dim GC_Counter, Complete_Counter, spectrum_sub_counter As Long
  GC_Counter = 0: Complete_Counter = 0

  For i = 1 To No.Science_Packets
    If Science(i).Description = " Isotope spectrum " Then
      If Science(i).flags And 2 Then
        Complete_Counter = Complete_Counter + 1
        Spectrum_Start_packet(Complete_Counter, 1) = i
        j = i + 1
        spectrum_sub_counter = 2
        Do Until j > No.Science_Packets
          Select Case Science(j).Description
            Case " GC spectrum      "
              j = No.Science_Packets
            Case " Isotope spectrum "
              If Science(j).flags And 2 Then
                j = No.Science_Packets
              Else
                Spectrum_Start_packet(Complete_Counter, spectrum_sub_counter) = j
                spectrum_sub_counter = spectrum_sub_counter + 1
              End If
          End Select
          j = j + 1
        Loop
      End If
    End If

    If Science(i).Description = " GC spectrum      " Then
      GC_Counter = GC_Counter + 1
      Spectrum_Start_packet(No.Complete_Science_Spectra + GC_Counter, 1) = i
    End If
  Next i

'Check that the text box range is valid
  Dim Text_Valid As Boolean
  Text_Valid = True
  If IsNumeric(Text1.Text) = False Then Text_Valid = False
  If IsNumeric(Text2.Text) = False Then Text_Valid = False
  If Text_Valid = True Then
    If Text1.Text < 1 Then Text_Valid = False
    If Text1.Text > Total_Number_Spectra Then Text_Valid = False
    If Text1.Text <> Int(Text1.Text) Then Text_Valid = False
    If Text2.Text < 1 Then Text_Valid = False
    If Text2.Text > Total_Number_Spectra Then Text_Valid = False
    If Text2.Text <> Int(Text2.Text) Then Text_Valid = False
    If Int(Text1.Text) > Int(Text2.Text) Then Text_Valid = False
  End If
  Dim Responce
```

```
  If Text_Valid = False Then
    Responce = MsgBox("Science packet ranges not valid")
    Text1.Text = 1
    Text2.Text = Total_Number_Spectra
    Exit Sub
  End If

  ReDim Spectra_Bins(Text2.Text - Text1.Text + 1, 1024)
  ReDim Bin_Counter(Text2.Text - Text1.Text + 1)

  CommonDialog1.CancelError = True
  On Error GoTo ErrorHandler      'Enable error detection
  CommonDialog1.ShowSave
  File_Export_Name = CommonDialog1.FileName

'Check to see if the file opened is a valid file
  Open CommonDialog1.FileName For Output As #Export_File_Number
  On Error GoTo 0                 'Turn off error handler

  Open Egse_Data_File_Name For Binary As #Data_File_Number

' Collect all the spectrum data
Dim Counter, spectrum_start_Byte As Long
Dim First_Bin, Number_of_Bins As Long
Dim Shift_Count, Mantisa As Long

  For i = Text1.Text To Text2.Text
    k = i - Text1.Text + 1 'k is the number of the spectrum
    Bin_Counter(k) = 0

    If i <= No.Complete_Science_Spectra Then
'     Acquire the data for the complete spectra
      Counter = 1
      Do
        spectrum_start_Byte = Science(Spectrum_Start_packet(i, Counter)).Start
        First_Bin = Get_Word(spectrum_start_Byte + 28)
        Number_of_Bins = Get_Word(spectrum_start_Byte + 30)
        For j = 0 To Number_of_Bins - 1
          Mantisa = Get_Word(spectrum_start_Byte + (16 + j) * 2)
          Shift_Count = Mantisa And &HF
          Mantisa = Int(Mantisa / 16)
          Spectra_Bins(k, First_Bin + j) = Mantisa * 2 ^ Shift_Count
          Bin_Counter(k) = Bin_Counter(k) + 1
        Next j
        Counter = Counter + 1
      Loop Until Spectrum_Start_packet(i, Counter) = 0 Or Counter = 10

    Else
'     Acquire the data for the GC spectra

    End If

  Next i

'  Export the data to excel
  Dim Big_String, Big_String2 As String
  Dim Deliminator_Type As Long

  If Option1.value = True Then Deliminator_Type = 9
  If Option2.value = True Then Deliminator_Type = 44
  If Option3.value = True Then Deliminator_Type = 59

  Print #Export_File_Number, "Spectra acquired from " & Egse_Data_File_Name
  If Option4.value = True Then
'   Export in raw data format
    Print #Export_File_Number, "Spectra in raw data format"
    Print #Export_File_Number, ""
    Big_String = "Bin No."
    For j = Text1.Text To Text2.Text
      Big_String = Big_String & Chr$(Deliminator_Type) & "Spec. " & j
    Next j
    Print #Export_File_Number, Big_String
    Print #Export_File_Number, ""

    For i = 0 To 1024
      Big_String = i
      For j = 1 To (Text2.Text - Text1.Text + 1)
```

```
        If i <= Bin_Counter(j) Then
          Big_String = Big_String & Chr$(Deliminator_Type) & Spectra_Bins(j, i)
        Else
          Big_String = Big_String & Chr$(Deliminator_Type) & ""
        End If
      Next j
      Print #Export_File_Number, Big_String
    Next i
  End If

  If Option5.value = True Then
'   Export data suitable for excel graphs
    Print #Export_File_Number, "Spectra in format suitable for excel graphs"
    Print #Export_File_Number, ""
    Big_String = "Bin No."
    For j = Text1.Text To Text2.Text
      Big_String = Big_String & Chr$(Deliminator_Type) & "Spec. " & j
    Next j
    Print #Export_File_Number, Big_String
    Print #Export_File_Number, ""

    For i = 0 To 1024
      Big_String = i
      Big_String2 = i + 1
      For j = 1 To (Text2.Text - Text1.Text + 1)
        If i <= Bin_Counter(j) Then
          Big_String = Big_String & Chr$(Deliminator_Type) & Spectra_Bins(j, i)
          Big_String2 = Big_String2 & Chr$(Deliminator_Type) & Spectra_Bins(j, i)
        Else
          Big_String = Big_String & Chr$(Deliminator_Type) & ""
          Big_String2 = Big_String2 & Chr$(Deliminator_Type) & ""
        End If
      Next j
      Print #Export_File_Number, Big_String
      Print #Export_File_Number, Big_String2
    Next i
  End If

  Close #Data_File_Number
  Close #Export_File_Number

Exit Sub


ErrorHandler:                   'Error detection routine

Select Case Err.Number
  Case 32755                    'Cancel button pressed then
                                'just exit sub

  Case 55
    Close #Export_File_Number

  Case Else                     'else explain cause of error
    MsgBox Err.Description & "; Error number " & Err.Number '

End Select
End Sub

Private Sub Command2_Click()
  Unload Export_Spectra
End Sub

Private Sub Form_Load()
  Total_Number_Spectra = No.Complete_Science_Spectra + No.Compact_Science
  Option1.value = True
  Option4.value = True
  Text1.Text = 1
  Text2.Text = Total_Number_Spectra

End Sub

Public Function Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
```

```
Dim High_Byte, Low_Byte As Long

Get #Data_File_Number, Byte_Position, First_Byte
Get #Data_File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Get_Word = High_Byte * 256 + Low_Byte

End Function
```

Housekeeping_form - 1

```
Option Explicit
Dim AD590_EMF_Correction As Double
Dim Mode_Sequence, Mode_Line As Long
Dim Mode_Name As String
Dim Show_HK_Complete As Boolean

Private Sub Command1_Click()
If Show_HK_Complete = True Then
  Show_HK_Complete = False
  Command1.Caption = "More..."
  Update_Sensor_readings (Current_Housekeeping_Packet)
Else
  Show_HK_Complete = True
  Command1.Caption = "Sensors"
  Show_HK_Extra
End If
End Sub

Private Sub Form_Load()

Form_Housekeeping_show = True

VScroll1.Max = No.Housekeeping_Packets

If No.Housekeeping_Packets > 0 Then
  VScroll1.value = Current_Housekeeping_Packet
  Update_Sensor_readings (Current_Housekeeping_Packet)
Else
  VScroll1.Max = 1
End If

End Sub

Public Sub Update_Sensor_readings(x)
Dim i As Integer
Dim ADC_value, Error_ADC_value As Double
Dim AD590_correction As Double

Label1.Caption = "Packet type - " & Housekeeping(Current_Housekeeping_Packet).Description
If Housekeeping(Current_Housekeeping_Packet).Description = " Complete " Then
  Command1.Visible = True
Else
  Command1.Visible = False
End If
Label6.Caption = Current_Housekeeping_Packet
Label7.Caption = Housekeeping(Current_Housekeeping_Packet).Sequence_Count
Label8.Caption = Housekeeping(Current_Housekeeping_Packet).Time
Label10.Caption = No.Housekeeping_Packets

'Load the sensor data and
'calculate the voltage for each sensor
For i = 1 To Number_of_Sensors
  Sensor(i).Byte_Value = Sensor_Bytes(i, Current_Housekeeping_Packet)
' not all byte values over 127 correspond to a negative voltage
  If Sensor(i).Byte_Value > 127 Then
    Select Case i
      Case 28, 29
        'do nothing
      Case 32 To 36
        If Sensor(i).Byte_Value > 210 Then
          Sensor(i).Byte_Value = Sensor(i).Byte_Value - 256
        End If
      Case Else
        Sensor(i).Byte_Value = Sensor(i).Byte_Value - 256
    End Select
  End If
  ADC_value = Sensor(i).Byte_Value * 2 ^ Sensor(i).Bit_Shift
  Error_ADC_value = (Sensor(i).Byte_Value + 1) * 2 ^ Sensor(i).Bit_Shift
  Sensor(i).voltage = ADC_value * 10 / 32768
  Sensor(i).Voltage_Error = Error_ADC_value * 10 / 32768
Next i

'Calculate temperature of AD590 (sensor number 28)
Call Get_Sensor_Reading(28, Sensor(28).voltage, Sensor(28).Reading, Sensor(28).unit)
Call Convert_Temperature_to_EMF(Sensor(28).Reading, AD590_EMF_Correction)
AD590_EMF_Correction = AD590_EMF_Correction / 1000000
```

```
Housekeeping_form - 2

For i = 1 To Number_of_Sensors
  Select Case i
    Case 1 To 21, 27  'Thermocouples, need to calibrate for effect of cold junction
                      ' and gain circuit
      Sensor(i).voltage = Sensor(i).voltage + AD590_EMF_Correction * 100
      Sensor(i).Voltage_Error = Sensor(i).Voltage_Error + AD590_EMF_Correction * 100
      Call Get_Sensor_Reading(i, Sensor(i).voltage, Sensor(i).Reading, Sensor(i).unit)
      Call Get_Sensor_Reading(i, Sensor(i).Voltage_Error, Sensor(i).Reading_Error, Sensor(i).unit)
    Case Else
      Call Get_Sensor_Reading(i, Sensor(i).voltage, Sensor(i).Reading, Sensor(i).unit)
      Call Get_Sensor_Reading(i, Sensor(i).Voltage_Error, Sensor(i).Reading_Error, Sensor(i).unit)
  End Select
Next i


Picture1.Cls
Picture1.ForeColor = RGB(0, 0, 255)
Picture1.Print "Reactors"
For i = 1 To 9              'Reactors 1 to 13
  Call xPrint_Line(i, 0, i, 0)
Next i
Call xPrint_Line(10, 0, 27, 0)  'Reactor 14
Call xPrint_Line(11, 0, 10, 0)  'Reactor 15
Call xPrint_Line(12, 0, 20, 0)   'Sample Oven

Picture1.PSet (0, 2800), RGB(255, 255, 255)
Picture1.Print "Lindau Valves"
For i = 11 To 15
  Call xPrint_Line(i + 4, 0, i, 1)
Next i

Picture1.PSet (0, 4200), RGB(255, 255, 255)
Picture1.Print "Heaters"
For i = 16 To 19
  Call xPrint_Line(i + 6, 0, i, 1)
Next i
Call xPrint_Line(26, 0, 21, 1)

'Print AD590 result
Call xPrint_Line(1, 1, 28, 1)

'Print Pressure sensors
Picture1.PSet (3200, 600), RGB(255, 255, 255)
Picture1.Print "Pressure Sensors"
For i = 22 To 26
  Call xPrint_Line(i - 18, 1, i, 2)
Next i

'Print Ion Trap sensors
Picture1.PSet (3200, 2200), RGB(255, 255, 255)
Picture1.Print "Ion trap"
  Call xPrint_Line(12, 1, 31, 2)
  Call xPrint_Line(13, 1, 30, 2)
  Call xPrint_Line(14, 1, 36, 2)

'Print Supply rail voltages and currents
Picture1.PSet (3200, 3200), RGB(255, 255, 255)
Picture1.Print "Supply rails"
  Call xPrint_Line(17, 1, 33, 2)
  Call xPrint_Line(18, 1, 35, 2)
  Call xPrint_Line(19, 1, 32, 2)
  Call xPrint_Line(20, 1, 34, 2)

'Print Docking station position
Picture1.PSet (3200, 4400), RGB(255, 255, 255)
Picture1.Print "Docking Station Position"
  Call xPrint_Line(23, 1, 29, 2)


'Print Mode result

Select Case Packet_Format
  Case "CSS", "QM"
    Load_Mode_Data
  Case "GRM", "FM"
    Load_Mode_Data  'Exactly the same format for Ptolemy format
  Case "RAL"
```

```
Housekeeping_form - 3

End Select

Label13.Caption = Mode_Sequence & "  -  " & Mode_Name
Label14.Caption = Mode_Line

End Sub

Public Sub Load_Mode_Data()
'This loads the Mode number and Command Event number
Dim File_Number As Integer
Dim Sensor_word As Byte
Dim i As Integer

File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #File_Number

If Not Byte_Reverse Then
  Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 18, Sensor_word
  Mode_Sequence = Sensor_word
  Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 20, Sensor_word
  If Sensor_word > 128 Then Sensor_word = 127
  Mode_Line = Sensor_word * 256
  Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 21, Sensor_word
  Mode_Line = Mode_Line + Sensor_word
Else
  Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 19, Sensor_word
  Mode_Sequence = Sensor_word
  Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 21, Sensor_word
  If Sensor_word > 128 Then Sensor_word = 127
  Mode_Line = Sensor_word * 256
  Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 20, Sensor_word
  Mode_Line = Mode_Line + Sensor_word
End If

Close #File_Number
Select Case Mode_Sequence
  Case 0
    Mode_Name = "Standby"
  Case 1
    Mode_Name = "Ground Test"
  Case 2
    Mode_Name = "Post Launch"
  Case 3
    Mode_Name = "Cruise Phase"
  Case 4
    Mode_Name = "Instrument Checkout"
  Case 5
    Mode_Name = "HTO Conditioning"
  Case 6
    Mode_Name = "MTO Conditioning"
  Case 7
    Mode_Name = "CASE Conditioning"
  Case 8
    Mode_Name = "Survival Evaluation"
  Case 9
    Mode_Name = "Helium Tank Rupture"
  Case 10
    Mode_Name = "Dynamic Pre-operations"
  Case 11
    Mode_Name = "Calibration"
  Case 12
    Mode_Name = "Ice Core Analysis(HTO)"
  Case 13
    Mode_Name = "Atmosphere Analysis"
  Case 14
    Mode_Name = "Silicate Analysis"
  Case 15
    Mode_Name = "Ice Core Analysis(MTO)"
  Case 16
    Mode_Name = "Additional Science"
  Case 255
    Mode_Name = "Safe"
  Case Else
    Mode_Name = "Unknown"
End Select
End Sub
```

```
Public Sub xPrint_Line(Line_Number, Column_Number, Sensor_Number, Dec_place)
'This routine displays sensor data to the screen
Dim Reading1, Reading2 As Variant

Reading1 = Sensor(Sensor_Number).Reading
Reading2 = Abs(Sensor(Sensor_Number).Reading - Sensor(Sensor_Number).Reading_Error)
Select Case Dec_place
  Case 0
    Reading1 = Format(Reading1, "###0")
    Reading2 = Format(Reading2, "###0")
  Case 1
    Reading1 = Format(Reading1, "###0.0")
    Reading2 = Format(Reading2, "###0.0")
  Case 2
    Reading1 = Format(Reading1, "###0.00")
    Reading2 = Format(Reading2, "###0.00")
  Case 3
    Reading1 = Format(Reading1, "###0.000")
    Reading2 = Format(Reading2, "###0.000")
End Select
  Picture1.PSet (Column_Number * 3200 + 100, Line_Number * 200), RGB(255, 255, 255)
  Picture1.Print Sensor(Sensor_Number).Name
  Picture1.PSet (Column_Number * 3200 + 1800 - TextWidth(Reading1), Line_Number * 200), RGB(255, 25
5, 255)
  Picture1.Print Reading1; "  +/-  ";
  Picture1.Print Reading2; "  ";
'  Picture1.PSet (Column_Number * 3200 + 2400 - TextWidth(Reading2), Line_Number * 200), RGB(255, 2
55, 255)
  Select Case Sensor(Sensor_Number).unit
    Case "oC"
      Picture1.PSet (Picture1.CurrentX, Picture1.CurrentY - 40), RGB(255, 255, 255)
      Picture1.Print "o";
      Picture1.PSet (Picture1.CurrentX, Picture1.CurrentY + 40), RGB(255, 255, 255)
      Picture1.Print "C";
    Case Else
      Picture1.Print Sensor(Sensor_Number).unit
  End Select
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form_Housekeeping_show = False
End Sub


Private Sub VScroll1_Change()
Current_Housekeeping_Packet = VScroll1.value
Show_HK_Complete = False
Command1.Caption = "More..."

Update_Sensor_readings (Current_Housekeeping_Packet)

End Sub

Public Sub UpdateRealtime()

VScroll1.Max = No.Housekeeping_Packets
VScroll1.value = No.Housekeeping_Packets

End Sub
Public Sub Update()

VScroll1.Max = No.Housekeeping_Packets
VScroll1.value = Current_Housekeeping_Packet

End Sub

Public Sub Show_HK_Extra()
Dim HK_Complete_Bytes(16) As Long
Dim HK_result As String
Dim x As Byte
Dim File_Number As Integer
Dim i, j As Integer

Picture1.Cls

'Get the complete HK information
File_Number = FreeFile
```

```
Housekeeping_form - 5

Open Egse_Data_File_Name For Binary As #File_Number

If Not Byte_Reverse Then
  For i = 1 To 16
    Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 62 + i * 2, x
    HK_Complete_Bytes(i) = x
    HK_Complete_Bytes(i) = HK_Complete_Bytes(i) * 256
    Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 63 + i * 2, x
    HK_Complete_Bytes(i) = HK_Complete_Bytes(i) + x
  Next i
Else
  For i = 1 To 16
    Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 63 + i * 2, x
    HK_Complete_Bytes(i) = x
    HK_Complete_Bytes(i) = HK_Complete_Bytes(i) * 256
    Get #File_Number, Housekeeping(Current_Housekeeping_Packet).Start + 62 + i * 2, x
    HK_Complete_Bytes(i) = HK_Complete_Bytes(i) + x
  Next i
End If

'Display the results
Dim EnName(16) As String
Dim x_pos, y_pos As Long
'Display results for valve enable register

EnName(1) = "VC": EnName(2) = "VB": EnName(3) = "VA": EnName(4) = "V16"
EnName(5) = "V15": EnName(6) = "V14": EnName(7) = "V13": EnName(8) = "V11"
EnName(9) = "V10": EnName(10) = "V9": EnName(11) = "V8": EnName(12) = "V7"
EnName(13) = "V6": EnName(14) = "V4": EnName(15) = "V2": EnName(16) = "V1"

Picture1.PSet (0, 0), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(2))
Picture1.Print "Valve enable register    " & HK_result
For i = 1 To 16
  If i < 9 Then
    y_pos = 200
    x_pos = i * 600
  Else
    y_pos = 600
    x_pos = i * 600 - 4800
  End If
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print EnName(i)
  Picture1.PSet (x_pos, y_pos + 200), RGB(255, 255, 255)
  If 2 ^ (16 - i) And HK_Complete_Bytes(2) Then
    Picture1.Print " En"
  Else
    Picture1.Print "  - "
  End If
Next i

'Display results for critical functions enable register

EnName(1) = "Ds Up": EnName(2) = "Ds Dn": EnName(3) = "HT": EnName(4) = "SMA2"
EnName(5) = "SMA1"

Picture1.PSet (0, 1200), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(3))
Picture1.Print "Critical functions enable register    " & HK_result
For i = 1 To 5
  y_pos = 1400
  x_pos = i * 1000 - 400
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print EnName(i)
  Picture1.PSet (x_pos, y_pos + 200), RGB(255, 255, 255)
  Select Case i     'Only display for critical functions available
    Case 1 To 3
      If 2 ^ (7 - i) And HK_Complete_Bytes(2) Then
        Picture1.Print " En"
      Else
        Picture1.Print "  - "
      End If
    Case 4, 5
      If 2 ^ (5 - i) And HK_Complete_Bytes(3) Then
        Picture1.Print " En"
      Else
        Picture1.Print "  - "
```

```
      End If
  End Select
Next i

'Display results for PWM enable register

EnName(1) = "Oven": EnName(2) = "Pipe": EnName(3) = "Ion": EnName(4) = "Enc2"
EnName(5) = "Enc1": EnName(6) = "GC": EnName(7) = "R15": EnName(8) = "R13"
EnName(9) = "R9/14": EnName(10) = "R8": EnName(11) = "R7": EnName(12) = "R6"
EnName(13) = "R5": EnName(14) = "R4": EnName(15) = "R2": EnName(16) = "R1"

Picture1.PSet (0, 2000), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(4))
Picture1.Print "PWM enable register    " & HK_result
For i = 1 To 16
  If i < 9 Then
    y_pos = 2200
    x_pos = i * 600
  Else
    y_pos = 2600
    x_pos = i * 600 - 4800
  End If
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print EnName(i)
  Picture1.PSet (x_pos, y_pos + 200), RGB(255, 255, 255)
  If 2 ^ (16 - i) And HK_Complete_Bytes(4) Then
    Picture1.Print " En"
  Else
    Picture1.Print "  - "
  End If
Next i

'Display contents of Valve Registers
Picture1.PSet (0, 3200), RGB(255, 255, 255)
Picture1.Print "Control Register Contents:         Word(Hex)                    Binary"
Picture1.PSet (500, 3400), RGB(255, 255, 255)
Picture1.Print "DAC control"
Picture1.PSet (500, 3600), RGB(255, 255, 255)
Picture1.Print "Valve control"
Picture1.PSet (500, 3800), RGB(255, 255, 255)
Picture1.Print "CF control"
Picture1.PSet (500, 4000), RGB(255, 255, 255)
Picture1.Print "PWM control"
For i = 1 To 4
  Picture1.PSet (2500, 3200 + 200 * i), RGB(255, 255, 255)
  HK_result = Convert_to_Hex(HK_Complete_Bytes(4 + i))
  Picture1.Print HK_result
  HK_result = ""
  For j = 1 To 16
    If 2 ^ (16 - j) And HK_Complete_Bytes(4 + i) Then
      HK_result = HK_result & "1 "
    Else
      HK_result = HK_result & "0 "
    End If
  Next j
  Picture1.PSet (3500, 3200 + 200 * i), RGB(255, 255, 255)
  Picture1.Print HK_result
Next i

'Display all other words
Picture1.PSet (0, 4400), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(1))
Picture1.Print "Background task "
Picture1.PSet (2400, 4400), RGB(255, 255, 255)
Picture1.Print HK_result

Picture1.PSet (3300, 4400), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(9))
Picture1.Print "RIU status word  "
Picture1.PSet (5700, 4400), RGB(255, 255, 255)
Picture1.Print HK_result

Picture1.PSet (0, 4600), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(12))
Picture1.Print "Mode event sequence state"
Picture1.PSet (2400, 4600), RGB(255, 255, 255)
Picture1.Print HK_result
```

```
Picture1.PSet (3300, 4600), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(10))
Picture1.Print "Last service request word "
Picture1.PSet (5700, 4600), RGB(255, 255, 255)
Picture1.Print HK_result

Picture1.PSet (0, 4800), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(13))
Picture1.Print "Science data transmission state   "
Picture1.PSet (2400, 4800), RGB(255, 255, 255)
Picture1.Print HK_result

Picture1.PSet (3300, 4800), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(11))
Picture1.Print "Last service request word sent "
Picture1.PSet (5700, 4800), RGB(255, 255, 255)
Picture1.Print HK_result

Picture1.PSet (0, 5000), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(14))
Picture1.Print "Current memory test address  "
Picture1.PSet (2400, 5000), RGB(255, 255, 255)
Picture1.Print HK_result

Picture1.PSet (0, 5200), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(15))
Picture1.Print "Number TC verify packets waiting transmission  "
Picture1.PSet (3500, 5200), RGB(255, 255, 255)
Picture1.Print HK_result

Picture1.PSet (0, 5400), RGB(255, 255, 255)
HK_result = Convert_to_Hex(HK_Complete_Bytes(16))
Picture1.Print "Number TC event packets waiting transmission "
Picture1.PSet (3500, 5400), RGB(255, 255, 255)
Picture1.Print HK_result

End Sub

Public Function Convert_to_Hex(z)
'This routine converts a number into a hex string with leading zero's
Dim Temp As String

Temp = Hex(z)
Select Case z
  Case Is < 16
    Temp = "000" & Temp
  Case Is < 16 * 16
    Temp = "00" & Temp
  Case Is < 16 * 16 * 16
    Temp = "0" & Temp
End Select

Convert_to_Hex = Temp

End Function
```

Information_Form - 1

```
Option Explicit
Dim Info_words(64) As Long
Dim Display_Word(32) As String
Private Type Lines
  Text As String
  colour As Long
End Type
Dim Line1 As Lines
Dim Line2 As Lines
Dim Line3 As Lines
Dim Line4 As Lines
Dim Line5 As Lines

Private Sub Form_Load()
Form_Information_Show = True

VScroll1.Max = No.Information_Packets

If No.Information_Packets > 0 Then
  VScroll1.value = Current_Information_Packet
Else
  VScroll1.Max = 1
End If

End Sub

Private Sub Form_Unload(Cancel As Integer)
Form_Information_Show = False
End Sub

Public Sub Update()
  VScroll1.Max = No.Information_Packets
  VScroll1.value = Current_Information_Packet
End Sub

Private Sub VScroll1_Change()
Current_Information_Packet = VScroll1.value
Label2.Caption = "Lander Pkt  " & Information(Current_Information_Packet).Lander_pkt
Label3.Caption = "Information Pkt " & Current_Information_Packet
Label5.Caption = Information(Current_Information_Packet).Description
Label7.Caption = Information(Current_Information_Packet).Sequence_Count
Label9.Caption = Information(Current_Information_Packet).Time
Select Case Packet_Format
  Case "CSS", "QM"
    Load_Information (Current_Information_Packet)
    Translate_Information
  Case "GRM", "FM"  ' same as lander packet
    Load_Information (Current_Information_Packet)
    Translate_Information
  Case "RAL"
    Load_Ral_Information (Current_Information_Packet)
End Select

End Sub

Public Sub Load_Information(Number)
'This routine displays the Ptolemy packet requested
'in picture box 1

Dim File_Number, i As Integer
Dim Start_Position As Long
Dim x(64) As Byte
Dim Display_Byte(64) As String

File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #File_Number
Start_Position = Information(Current_Information_Packet).Start - 1
For i = 1 To Information(Current_Information_Packet).Length
  Get #File_Number, Start_Position + i, x(i)
Next i
Close #File_Number
'Display the data in the picture box

'First put the raw data into hex format
For i = 1 To Information(Current_Information_Packet).Length
  Display_Byte(i) = Hex(x(i))
  If x(i) < 16 Then Display_Byte(i) = "0" & Display_Byte(i)
```

```
Information_Form - 2

Next i

'Next put the bytes into words and swop bytes if byte swop is set to TRUE
For i = 1 To Information(Current_Information_Packet).Length / 2
  If Byte_Reverse = True Then
    Display_Word(i) = Display_Byte(i * 2) & Display_Byte(i * 2 - 1)
    Info_words(i) = x(i * 2)
    Info_words(i) = Info_words(i) * 256 + x(i * 2 - 1)
  Else
    Display_Word(i) = Display_Byte(i * 2 - 1) & Display_Byte(i * 2)
    Info_words(i) = x(i * 2 - 1)
    Info_words(i) = Info_words(i) * 256 + x(i * 2)
  End If
Next i

'Finally display the packet in the picture box
Picture1.Cls
Picture1.ForeColor = RGB(0, 0, 255)

Dim x_pos, y_pos As Single
For i = 1 To Information(Current_Information_Packet).Length / 2
  y_pos = Int((i - 1) / 8)
  x_pos = (i - y_pos * 8)
  If x_pos > 4 Then x_pos = x_pos + 0.3
  y_pos = y_pos * 250
  x_pos = x_pos * 700 - 500
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print Display_Word(i)
Next i

End Sub

Public Sub Load_Ral_Information(Number)
'This routine displays the RAL packet requested
'in picture box 1.  Note RAL packets are not translated.
Dim File_Number, i, j As Integer
Dim x(136)

File_Number = FreeFile
Open Egse_Data_File_Name For Input As #File_Number
i = 0
Do
  For j = 1 To 136
    Input #File_Number, x(j)
  Next j
  i = i + 1
Loop Until i = Information(Number).Lander_pkt

Close #File_Number

Picture1.Cls
Picture1.ForeColor = RGB(0, 0, 255)

Dim x_pos, y_pos As Single
For i = 1 To 136
  y_pos = Int((i - 1) / 8)
  x_pos = (i - y_pos * 8)
  y_pos = y_pos * 250
  x_pos = x_pos * 700 - 500
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print x(i)
Next i

End Sub

Public Sub Translate_Information()
'This routine translates the information packet for display.
'A long routine but it contains a lot of information.
Dim dummy

Line1.Text = "": Line1.colour = RGB(0, 0, 0)
Line2.Text = "": Line2.colour = RGB(0, 0, 0)
Line3.Text = "": Line3.colour = RGB(0, 0, 0)
Line4.Text = "": Line4.colour = RGB(0, 0, 0)
Line5.Text = "": Line5.colour = RGB(0, 0, 0)
Picture2.Cls
```

```
Information_Form - 3

Select Case Information(Current_Information_Packet).Description

  Case " Acceptance Success      "
    Line1.Text = "Packet ID of accepted packet         " & Display_Word(9)
    Line2.Text = "Sequence control accepted packet " & Display_Word(10)

  Case " Acceptance Failure      "
    Line1.Text = "Packet ID of rejected packet         " & Display_Word(9)
    Line2.Text = "Sequence control rejected packet " & Display_Word(10)
    Line1.colour = RGB(255, 0, 0): Line2.colour = RGB(255, 0, 0)
    Line3.colour = RGB(255, 0, 0): Line4.colour = RGB(255, 0, 0)
    Line5.colour = RGB(255, 0, 0)

    Dim TC_Type, TC_Sub_Type As Long
    Dim TC_Description As String

    TC_Type = Int(Info_words(12) / 256)
    TC_Sub_Type = Info_words(12) - TC_Type * 256
    Select Case TC_Type
      Case 6
        If TC_Sub_Type = 2 Then TC_Description = "Load memory"
        If TC_Sub_Type = 5 Then TC_Description = "Dump memory"
        If TC_Sub_Type = 9 Then TC_Description = "Check memory"
      Case 17
        If TC_Sub_Type = 17 Then TC_Description = "Connection test"
      Case 193
        If TC_Sub_Type = 0 Then TC_Description = "Standby"
        If TC_Sub_Type = 1 Then TC_Description = "Ground Test"
        If TC_Sub_Type = 2 Then TC_Description = "Post Launch"
        If TC_Sub_Type = 3 Then TC_Description = "Cruise Phase"
        If TC_Sub_Type = 4 Then TC_Description = "Instrument Checkout"
        If TC_Sub_Type = 5 Then TC_Description = "HTO Conditioning"
        If TC_Sub_Type = 6 Then TC_Description = "MTO Conditioning"
        If TC_Sub_Type = 7 Then TC_Description = "CASE Conditioning"
        If TC_Sub_Type = 8 Then TC_Description = "Survival Evaluation"
        If TC_Sub_Type = 9 Then TC_Description = "Helium Tank Rupture"
        If TC_Sub_Type = 10 Then TC_Description = "Dynamic Pre-operations"
        If TC_Sub_Type = 11 Then TC_Description = "Calibration"
        If TC_Sub_Type = 12 Then TC_Description = "Ice Core Analysis (HTO)"
        If TC_Sub_Type = 13 Then TC_Description = "Atmosphere Analysis"
        If TC_Sub_Type = 14 Then TC_Description = "Silicate Analysis"
        If TC_Sub_Type = 15 Then TC_Description = "Ice Core Analysis (MTO)"
        If TC_Sub_Type = 16 Then TC_Description = "Additional Science"
        If TC_Sub_Type = 255 Then TC_Description = "Safe"
      Case 194
        If TC_Sub_Type = 1 Then TC_Description = "Hazardous Function"
      Case 195
        If TC_Sub_Type = 1 Then TC_Description = "Parameter update"
    End Select
    Line4.Text = "TC type  " & TC_Type & "    TC sub-type " & TC_Sub_Type & "    -    " & TC_Descri
ption

    Select Case Info_words(11)     'Failure code ID
      Case 1
        Line3.Text = "Incomplete packet "
        Line5.Text = Info_words(13) & " bytes in header, " & Info_words(14) & " bytes actually rece
ived"
      Case 2
        Line3.Text = "Incorrect Checksum "
        Line5.Text = "Checksum received: " & Display_Word(13) & "  Calculated checksum: " & Display
_Word(14)
      Case 3
        Line3.Text = "Incorrect Application ID"
      Case 4
        Line3.Text = "Invalid Command Code"
      Case 5
        Line3.Text = "Not allowed in this mode/state"
        Line5.Text = "Current operating mode or SD2 status " & Info_words(13)
      Case 6
        Line3.Text = "Packet data field inconsistent"
        Line5.Text = "Further manual decoding necessary"
      Case Else
        Line3.Text = "Error code, " & Info_words(11) & ", not known"
        Line5.Text = "You'll have to sort out what's happening"
      End Select

  Case " Unknown - acceptance    "
```

```
Information_Form - 4

    Line1.Text = "Unknown Acceptance Packet"
    Line2.Text = "I haven't a clue what's happening"

  Case " Normal Progress Event   "
    Select Case Info_words(9)    '  The event ID
      Case 55103
        Line1.Text = "WGA memory check at " & Info_words(10) * 256 + Info_words(11) & " seconds"
        Dim i As Integer
        Dim x As Long
        x = 0
        For i = 14 To 32
         x = x + Info_words(i)
        Next i
        If x > 0 Then
          Line2.Text = "Errors in WGA memory map detected"
        Else
          Line2.Text = "No errors in WGA memory map detected"
        End If
      Case 55107
        Line1.Text = "Mode " & Info_words(10) & " execution just completed"
      Case 55001
        Line1.Text = "Ptolemy Power-on start"
        Dim Pass As Boolean
        Pass = True
        If Info_words(10) <> 43520 Then Pass = False 'Hex AA00
        For i = 11 To 27
          If Info_words(i) <> 0 Then Pass = False
        Next i
        If Pass = True Then
          Line2.Text = "Normal startup - no anomalies detected"
        Else
          Line2.colour = RGB(255, 0, 0)
          Line2.Text = "Anomalies detected, there are lots of start-up parameters - use the TC/TM d
ocument"
        End If
        Line3.Text = "Selected RAM data page  " & Info_words(28)
        Line4.Text = "Selected RAM code page " & Info_words(29)
      Case 55005
        Line1.Text = "Operating mode selection"
        Dim Current_Mode As Long
        Current_Mode = Int(Info_words(10) / 256)
        Line2.Text = "Current mode  " & Current_Mode & " at line " & Info_words(10) - Current_Mode
* 256
        Line3.Text = "Selected mode " & Int(Info_words(11) / 256)
        Line4.Text = "Mode parameters   " & Display_Word(12) & "   " & Display_Word(13) & "   " & D
isplay_Word(14)
      Case 55010
        Line1.Text = "SD2 Backup RAM Received"
        Line2.Text = "SD2 status " & Display_Word(10)
        Line3.Text = "Drill depth   " & Display_Word(11)
        Line4.Text = "Carousel position " & Display_Word(12) & " Oven number " & Display_Word(13)
      Case 55011
        Line1.Text = "Ptolemy Backup RAM Received"
        Line2.Text = "Carousel use state " & Display_Word(10) & "         RF Calibration word" & Di
splay_Word(11)
        Line3.Text = "Docking station motor,     upper position " & Display_Word(12) & ",   lower p
osition " & Display_Word(13)
        Line4.Text = "Docking station sensor value,   docked " & Display_Word(14) & ",         undo
cked " & Display_Word(15)
      Case 55113
        Line1.Text = "RF Calibration report"
      Case 55114
        Line1.Text = "Docking station sensor data"
      Case Else
        Line1.Text = "Unknown Event ID"
        Line2.Text = "You're on you own for this one"
    End Select

  Case " Warning Anomalous Event "
    Line1.colour = RGB(255, 0, 0): Line2.colour = RGB(255, 0, 0)
    Line3.colour = RGB(255, 0, 0): Line4.colour = RGB(255, 0, 0)
    Line5.colour = RGB(255, 0, 0)

    Select Case Info_words(9)    '  The event ID
      Case 55101
        Line1.Text = "Monitor mode event - Timed out"
      Case 55102
```

```
        Line1.Text = "WGA communication error"
      Case 55104
        Line1.Text = "Scan function in WGA does not match that written"
      Case 55105
        Line1.Text = "HT did not ramp to required value within timeout period"
      Case 55106
        Line1.Text = "Docking station failed to dock/undock"
      Case 55108
        Line1.Text = "Parameters for a mode event are incorrect"
        Line2.Text = "Operating mode " & Info_words(10) & ", at line number " & Info_words(11)
        Line4.Text = "Mode event parameters causing problem"
        Line5.Text = Display_Word(12) & "   " & Display_Word(13) & "   " & Display_Word(14) & "   "
        Line5.Text = Line5.Text & Display_Word(15) & "   " & Display_Word(16) & "   " & Display_Wor
d(17)
      Case 55109
        Line1.Text = "No RAM page available for science spectra storage"
      Case 55110
        Line1.Text = "Spectra storage data page is full"
      Case 55111
        Line1.Text = "Science data packets buffer is full"
      Case 55112
        Line1.Text = "No RAM page available for Science data packet storage"
      Case 55002
        Line1.Text = "Ptolemy Failure"
        Line2.Text = "DAC Control Register                    " & Display_Word(11)
        Line3.Text = "PWM Control Register                  " & Display_Word(12)
        Line4.Text = "Valve Control Register                  " & Display_Word(13)
        Line5.Text = "Critical Functions Control Register " & Display_Word(14)
      Case 55003
        Line1.Text = "Ptolemy Timeout"
      Case 55004
        Line1.Text = "RSST checksum failure"
      Case 55006
        Line1.Text = "Memory check failure"
      Case 55007
        Line1.Text = "Safe limit violation"
        Line2.Text = Get_Sensor_Name(Info_words(10)) & " reading exceeded safe limit"
        Line3.Text = "Sensor ADC reading " & Display_Word(11) & "    -   "
        dummy = get_line_result(Info_words(10), Info_words(11))
        Line3.Text = Line3.Text & dummy
        Line4.Text = "Maximum safe limit " & Display_Word(12) & "(hex)  -   "
        dummy = get_line_result(Info_words(10), Info_words(12))
        Line4.Text = Line4.Text & dummy
        Line5.Text = "Minimum safe limit " & Display_Word(13) & "(hex)  -   "
        dummy = get_line_result(Info_words(10), Info_words(13))
        Line5.Text = Line5.Text & dummy
      Case 55008
        Line1.Text = "Operating limit excursion"
        Line2.Text = Get_Sensor_Name(Info_words(10)) & " reading exceeded normal operating limit"
        Line3.Text = "Sensor ADC reading " & Display_Word(11) & "(hex)    -   "
        dummy = get_line_result(Info_words(10), Info_words(11))
        Line3.Text = Line3.Text & dummy
        Line4.Text = "Maximum operating limit " & Display_Word(12) & "(hex)  -   "
        dummy = get_line_result(Info_words(10), Info_words(12))
        Line4.Text = Line4.Text & dummy
        Line5.Text = "Minimum operating limit  " & Display_Word(13) & "(hex)  -   "
        dummy = get_line_result(Info_words(10), Info_words(13))
        Line5.Text = Line5.Text & dummy
      Case 55009
        Line1.Text = "Operating limit return"
        Line2.Text = Get_Sensor_Name(Info_words(10)) & " reading returned to normal operating range
"
        Line3.Text = "Sensor ADC reading " & Display_Word(11) & "    -   "
        dummy = get_line_result(Info_words(10), Info_words(11))
        Line3.Text = Line3.Text & dummy
        Line4.Text = "Maximum operating limit " & Display_Word(12) & "(hex)  -   "
        dummy = get_line_result(Info_words(10), Info_words(12))
        Line4.Text = Line4.Text & dummy
        Line5.Text = "Minimum operating limit  " & Display_Word(13) & "(hex)  -   "
        dummy = get_line_result(Info_words(10), Info_words(13))
        Line5.Text = Line5.Text & dummy
      Case 0
        Line1.colour = RGB(0, 0, 0)
        Line3.colour = RGB(0, 0, 0)
        Line1.Text = "Connection Test"
        Line3.Text = "All OK"
      Case Else
```

```
        Line1.Text = "Unknown cause of anomalous warning packet"
    End Select

  Case " Unknown - progress event"
    Line1.Text = "An unknown progress event occurred"

  Case " Memory Dump             "
    Line1.Text = "Memory dump          Total of " & Info_words(9) Mod 256 & " blocks of memory"
    Select Case Int(Info_words(9) / 256)
      Case 150          'Hex 96 - PROM
        Line2.Text = "Memory type - PROM"
      Case 151          'Hex 97 - EEPROM
        Line2.Text = "Memory type - EEPROM"
      Case 152          'Hex 98 - RAM
        Line2.Text = "Memory type - RAM"
      Case Else
        Line2.Text = "Unknown memory type"
        Line2.colour = RGB(255, 0, 0)
    End Select
    Display_Memory_Data

  Case " NPE - Memory Checksum   "
    Dim Number_blocks As Long
    Number_blocks = Info_words(9) Mod 256

    Select Case Int(Info_words(9) / 256)
      Case 150          'Hex 96 - PROM
        dummy = "PROM"
      Case 151          'Hex 97 - EEPROM
        dummy = "EEPROM"
      Case 152          'Hex 98 - RAM
        dummy = "RAM"
      Case Else
        dummy = "Unknown memory type"
    End Select
    dummy = "Memory Checksum packet of " & dummy & ".    "
    If Number_blocks = 1 Then
      Line1.Text = dummy & Number_blocks & " Checksum report."
    Else
      Line1.Text = dummy & Number_blocks & " Checksum reports."
    End If
    Display_Memory_Checksum

  Case "  Empty  "
    Line1.Text = "Looks like an empty packet, perhaps Ptolemy is bored"
  Case " Unknown "
    Line1.Text = "Unknown packet - user to interpret"
End Select

Picture2.PSet (0, 0), RGB(255, 255, 255)
Picture2.ForeColor = Line1.colour
Picture2.Print Line1.Text
Picture2.ForeColor = Line2.colour
Picture2.Print Line2.Text
Picture2.ForeColor = Line3.colour
Picture2.Print Line3.Text
Picture2.ForeColor = Line4.colour
Picture2.Print Line4.Text
Picture2.ForeColor = Line5.colour
Picture2.Print Line5.Text

End Sub

Public Sub Display_Memory_Data()
'This routine displays the data from a memory dump packet

Dim Word_counter, Block_length  As Integer
Dim i, j As Integer
Dim X_Position, Y_Position As Long

X_Position = 0: Y_Position = 600
Word_counter = 10

For i = 1 To Info_words(9) Mod 256        'for each memory block
  Picture2.ForeColor = RGB(255, 0, 0)
  Picture2.PSet (X_Position, Y_Position), RGB(255, 255, 255)
  Picture2.Print Display_Word(Word_counter)
```

```
  X_Position = X_Position + 700
  If X_Position > 4900 Then
    X_Position = 0: Y_Position = Y_Position + 200
  End If
  Picture2.PSet (X_Position, Y_Position), RGB(255, 255, 255)
  Picture2.Print Display_Word(Word_counter + 1)
  X_Position = X_Position + 700
  If X_Position > 4900 Then
    X_Position = 0: Y_Position = Y_Position + 200
  End If

  Picture2.ForeColor = RGB(0, 0, 0)       'Print length of memory block
  Picture2.PSet (X_Position, Y_Position), RGB(255, 255, 255)
  Picture2.Print Display_Word(Word_counter + 2)
  X_Position = X_Position + 700
  If X_Position > 4900 Then
    X_Position = 0: Y_Position = Y_Position + 200
  End If

  Block_length = Info_words(Word_counter + 2)
  Picture2.ForeColor = RGB(0, 0, 255)      'Print memory block
  For j = 1 To Block_length
    Picture2.PSet (X_Position, Y_Position), RGB(255, 255, 255)
    Picture2.Print Display_Word(Word_counter + 2 + j)
    X_Position = X_Position + 700
    If X_Position > 4900 Then
      X_Position = 0: Y_Position = Y_Position + 200
    End If
  Next j
  Word_counter = Word_counter + 3 + Block_length
Next i

End Sub

Public Function Get_Sensor_Name(channel_number)
Dim i As Integer
Dim Name As String

Name = "Unknown sensor"
For i = 1 To Number_of_Sensors
 If Sensor(i).ADC_Channel = channel_number Then
   Name = Sensor(i).Name
 End If
Next i
Get_Sensor_Name = Name

End Function

Public Function get_line_result(ADC_Chan, ADC_Reading)
Dim i As Integer
Dim ID As Integer
Dim value As Long
Dim voltage, Result As Double
Dim unit As String

ID = 255
For i = 1 To Number_of_Sensors
 If Sensor(i).ADC_Channel = ADC_Chan Then
   ID = i
 End If
Next i
value = ADC_Reading
If value > 32767 Then value = value - 65535
voltage = value * 10 / 32768
Call Get_Sensor_Reading(ID, voltage, Result, unit)
get_line_result = Format(Result, "0.000") & " " & unit
End Function

Public Sub Display_Memory_Checksum()

'This routine displays the data from a memory checksum packet

Dim i, Number_blocks  As Long
Dim dummy As String

Number_blocks = Info_words(9) Mod 256
dummy = "Checksum "
```

```
Picture2.ForeColor = RGB(0, 0, 0)
For i = 1 To Number_blocks
  Picture2.PSet (0, i * 200), RGB(255, 255, 255)
  Picture2.Print "Checksum " & i
  Picture2.PSet (1200, i * 200), RGB(255, 255, 255)
  Picture2.Print "Address  " & Display_Word(6 + i * 4) & "  " & Display_Word(7 + i * 4)
  Picture2.PSet (3000, i * 200), RGB(255, 255, 255)
  Picture2.Print "Words ";
  Picture2.Print Display_Word(8 + i * 4)
  Picture2.PSet (4500, i * 200), RGB(255, 255, 255)
  Picture2.Print "Checksum  =  ";
  Picture2.Print Display_Word(9 + i * 4)
Next i

End Sub
```

```
Option Explicit
Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte As Byte
Dim Low_Bye As Byte

Dim Display_Packet_Number As Long
Dim File_Number As Integer

Private Sub Form_Load()
RawData.Caption = Egse_Data_File_Name
Form_Raw_Data_Show = True

'Display number of packets and the first lander packet raw data
Label1.Caption = "Number of Lander packets = " & No.Lander_Packets
VScroll1.Max = No.Lander_Packets
If No.Lander_Packets = 0 Then VScroll1.Max = 1   'A fix to stop computer crashing if there are no p
ackets
VScroll1.value = 1
Display_Packet_Number = 1
Label2.Caption = "Lander packet number " & Display_Packet_Number
Select Case Packet_Format
  Case "CSS", "QM"
    Display_282_Byte_Packet (Display_Packet_Number)
  Case "GRM", "FM"
    Display_280_Byte_Packet (Display_Packet_Number)
  Case "RAL"
    Display_RAL_Packet (Display_Packet_Number)
End Select

End Sub

Public Sub Display_282_Byte_Packet(Number)
'This routine displays the Lander packet requested
'in list box 1

Dim File_Number, i As Integer
Dim Start_Position As Long
Dim x(282) As Byte
Dim Display_Byte(282), Display_Word(141) As String

File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #File_Number
Start_Position = (Number - 1) * 282
For i = 1 To 282
  Get #File_Number, Start_Position + i, x(i)
Next i

'Display the data in the picture box

'First put the raw data into hex format
For i = 1 To 282
  Display_Byte(i) = Hex(x(i))
  If x(i) < 16 Then Display_Byte(i) = "0" & Display_Byte(i)
Next i

'Next put the bytes into words and swop bytes if byte swop is set to TRUE
For i = 1 To 141
  If Byte_Reverse = True Then
    Display_Word(i) = Display_Byte(i * 2) & Display_Byte(i * 2 - 1)
  Else
    Display_Word(i) = Display_Byte(i * 2 - 1) & Display_Byte(i * 2)
  End If
Next i

'Finally display the packet in the picture box
Picture1.Cls
Dim x_pos, y_pos As Single
For i = 1 To 141
  y_pos = Int((i - 1) / 8)
  x_pos = (i - y_pos * 8)
  If x_pos > 4 Then x_pos = x_pos + 0.3
  y_pos = y_pos * 250
  x_pos = x_pos * 700 - 500
  If i <= 11 Or i > 139 Then
    Picture1.ForeColor = RGB(255, 0, 0)
  Else
```

```
    Picture1.ForeColor = RGB(0, 0, 255)
  End If
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print Display_Word(i)
Next i
Close #File_Number
End Sub
Public Sub Display_280_Byte_Packet(Number)
'This routine displays the Ptolemy packet requested
'in picture box 1

Dim File_Number, i As Integer
Dim Start_Position As Long
Dim x(280) As Byte
Dim Display_Byte(280), Display_Word(140) As String

File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #File_Number
Start_Position = (Number - 1) * 280
For i = 1 To 280
  Get #File_Number, Start_Position + i, x(i)
Next i

'Display the data in the picture box

'First put the raw data into hex format
For i = 1 To 280
  Display_Byte(i) = Hex(x(i))
  If x(i) < 16 Then Display_Byte(i) = "0" & Display_Byte(i)
Next i

'Next put the bytes into words and swop bytes if byte swop is set to TRUE
For i = 1 To 140
  If Byte_Reverse = True Then
    Display_Word(i) = Display_Byte(i * 2) & Display_Byte(i * 2 - 1)
  Else
    Display_Word(i) = Display_Byte(i * 2 - 1) & Display_Byte(i * 2)
  End If
Next i

'Finally display the packet in the picture box
Picture1.Cls
Dim x_pos, y_pos As Single
For i = 1 To 140
  y_pos = Int((i - 1) / 8)
  x_pos = (i - y_pos * 8)
  If x_pos > 4 Then x_pos = x_pos + 0.3
  y_pos = y_pos * 250
  x_pos = x_pos * 700 - 500
  If i <= 11 Or i > 139 Then
    Picture1.ForeColor = RGB(255, 0, 0)
  Else
    Picture1.ForeColor = RGB(0, 0, 255)
  End If
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print Display_Word(i)
Next i
Close #File_Number
End Sub
Public Sub Display_RAL_Packet(Number)
'This routine displays the RAL packet requested
'in picture box 1
Dim File_Number, i, j As Integer
Dim x(136)

File_Number = FreeFile
Open Egse_Data_File_Name For Input As #File_Number
i = 0
Do
  For j = 1 To 136
    Input #File_Number, x(j)
  Next j
  i = i + 1
Loop Until i = Number
Picture1.Cls
Picture1.ForeColor = RGB(0, 0, 255)
```

```
RawData - 3

Dim x_pos, y_pos As Single
For i = 1 To 136
  y_pos = Int((i - 1) / 8)
  x_pos = (i - y_pos * 8)
  y_pos = y_pos * 250
  x_pos = x_pos * 700 - 500
  Picture1.PSet (x_pos, y_pos), RGB(255, 255, 255)
  Picture1.Print x(i)
Next i
Close #File_Number

End Sub

Private Sub Form_Unload(Cancel As Integer)
Form_Raw_Data_Show = False
End Sub

Private Sub VScroll1_Change()

Display_Packet_Number = VScroll1.value
Label2.Caption = "Lander packet number " & Display_Packet_Number
Select Case Packet_Format
  Case "CSS", "QM"
    Display_282_Byte_Packet (Display_Packet_Number)
  Case "GRM", "FM"
    Display_280_Byte_Packet (Display_Packet_Number)
  Case "RAL"
    Display_RAL_Packet (Display_Packet_Number)
End Select

End Sub

Public Sub UpdateRawData()

'Display number of packets and the first lander packet raw data
Label1.Caption = "Number of Lander packets = " & No.Lander_Packets
VScroll1.Max = No.Lander_Packets
If No.Lander_Packets = 0 Then VScroll1.Max = 1    'A fix to stop computer crashing if there are no
packets
VScroll1.value = VScroll1.Max
Display_Packet_Number = No.Lander_Packets
Label2.Caption = "Lander packet number " & Display_Packet_Number
Select Case Packet_Format
  Case "CSS", "QM"
    Display_282_Byte_Packet (Display_Packet_Number)
  Case "GRM", "FM"
    Display_280_Byte_Packet (Display_Packet_Number)
  Case "RAL"
    Display_RAL_Packet (Display_Packet_Number)
End Select
End Sub
```

```
ShortMessageLoad - 1

Option Explicit
Dim Aux_ShortTest As Boolean
Dim Sequence_Count As Long
Dim Word_Result As Long
Dim File_Number As Integer
Dim Listbox() As Long

Private Sub Command2_Click()
  Unload ShortMessageLoad
End Sub

Private Sub Form_Load()
Dim i, x As Long
Dim dummy, dummy2 As String
Dim Number_SF_Tests As Long

ReDim Listbox(No.Science_Packets)
'Determine which Aux data packets could be the start of a short test
'Valid auxiliary data packets are loaded into the list box.
Number_SF_Tests = 0
File_Number = FreeFile
If Egse_Data_File_Name <> "" Then
  Open Egse_Data_File_Name For Binary As #File_Number
End If

For i = 1 To No.Science_Packets
  Aux_ShortTest = True
  Sequence_Count = Science(i).Sequence_Count

' Check that all the packets within the 7 sequence counts are
' auxiliary data packets
  x = i
  Do
    If Science(x).Sequence_Count <= Sequence_Count + 7 Then
      If Science(x).Description <> " Auxiliary Data   " Then
        Aux_ShortTest = False
        x = No.Science_Packets + 1
      End If
      x = x + 1
    Else
      x = No.Science_Packets + 1
    End If
  Loop Until x > No.Science_Packets Or x > i + 7

'  Check a few of the fields in the first auxiliary data packet
'  are consistent with the short form test.

'  Check that the number of records in the first auxiliary data
'  packet = 29.
  Get_Word (Science(i).Start + 18)
  If Word_Result <> 29 Then
    Aux_ShortTest = False
  End If
'  Check that the first sensor channel is for pressure gauge 1
'  (channel number 23) for Cruise Phase test and pressure gauge 2
'  (channel number 24) for EAFT test
  Get_Word (Science(i).Start + 24)
  Select Case Test_Type
    Case "Cruise Phase"
      If Word_Result <> 23 Then
        Aux_ShortTest = False
      End If
    Case "EAFT"
      If Word_Result <> 24 Then
        Aux_ShortTest = False
      End If
    Case Else
    Stop
  End Select

'  Check that the 8th sensor channel is for nanotips
'  (channel number 64)
  Get_Word (Science(i).Start + 80)
  If Word_Result <> 64 Then
    Aux_ShortTest = False
  End If
'  Check that the 27th sensor channel is for Reactor 1
```

```
'  (channel number 0)
  Get_Word (Science(i).Start + 232)
  If Word_Result <> 0 Then
    Aux_ShortTest = False
  End If
'  Check that the 29th sensor channel is for 28V current
'  (channel number 144)
  Get_Word (Science(i).Start + 248)
  If Word_Result <> 144 Then
    Aux_ShortTest = False
  End If

  If Aux_ShortTest = True Then
    Number_SF_Tests = Number_SF_Tests + 1
    Listbox(Number_SF_Tests) = i
    dummy2 = Number_SF_Tests
    dummy = dummy2 & Space(6 - Len(dummy2)) & "Short form test"
    dummy2 = Science(i).Sequence_Count
    dummy = dummy & Space(9 - Len(dummy2)) & dummy2
    dummy2 = Science(i).Time
    dummy = dummy & Space(15 - Len(dummy2)) & dummy2
    List1.AddItem dummy
  End If
Next i

Close #File_Number

Select Case Number_SF_Tests
  Case 0
    Label1.Caption = "There were no Short form tests collected during this session"
    Label2.Caption = "Press 'cancel' to exit"
  Case 1
    Label1.Caption = "There was 1 Short form test collected during this session"
    Label2.Caption = "Double click on test to view or Press 'cancel' to exit"
  Case Else
    Label1.Caption = "There were " & Number_SF_Tests & " Short form tests collected during this ses
sion"
    Label2.Caption = "Double click on test to view or Press 'cancel' to exit"
End Select

End Sub

Public Sub Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte, Low_Byte As Long

Get #File_Number, Byte_Position, First_Byte
Get #File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Word_Result = High_Byte * 256 + Low_Byte

End Sub

Private Sub List1_DblClick()
  Test_Start_Packet = Listbox(List1.ListIndex + 1)
  Unload ShortTest
  ShortTest.Show
  Unload ShortMessageLoad
End Sub
```

```
Option Explicit

Dim Short_Packets(8) As Long
Dim Found_Packet(8) As Boolean
Dim RF_Calibration_Found As Boolean
Dim Start_Sequence_Count As Long
Dim Channel_Numbers(232) As Integer
Dim Short_Test_Words(232), Word_Result As Long
Dim Short_Test_Voltages(232), Short_Test_Results(232) As Double
Dim Min_5V(38), Max_5V(38) As Double
Dim Min_28V(38), Max_28V(38) As Double
Dim Min_Range(38), Max_Range(38) As Double
Dim Short_Test_Units(232) As String
Dim File_Number As Integer
Dim AD590_EMF, RF_Frequency As Double
Dim Result As Boolean
Dim Start_time As Long

Private Sub Form_Load()

Dim i, x As Long
Dim Range_File_Number As Integer
Dim Range_File_Name As String

Erase Short_Test_Words(), Short_Test_Voltages(), Channel_Numbers()
Erase Short_Test_Results(), Short_Test_Units()

For i = 1 To 8
  Short_Packets(i) = 0
  Found_Packet(i) = False
Next i
Start_Sequence_Count = Science(Test_Start_Packet).Sequence_Count
Start_time = Science(Test_Start_Packet).Time
Label36.Caption = "Test started at Lander time " & Start_time & " seconds"

'First find out whether all the short form test packets
'are available.  Packets may have been lost or the test
'stopped early.  Found_Packet is used to flag that a packet
'with the correct sequence count has been found.  All packets must
'have been collected within 900 seconds of the test start time
Dim Valid_packet As Boolean
x = Test_Start_Packet
Do
  Valid_packet = True
  If Science(x).Sequence_Count < Start_Sequence_Count Then Valid_packet = False
  If Science(x).Sequence_Count > Start_Sequence_Count + 7 Then Valid_packet = False
  If Science(x).Description <> " Auxiliary Data   " Then Valid_packet = False
  If Science(x).Time < Start_time Then Valid_packet = False
  If Science(x).Time > Start_time + 900 Then Valid_packet = False
  If Valid_packet = True Then
    i = Science(x).Sequence_Count - Start_Sequence_Count + 1
    If Found_Packet(i) = False Then
      Found_Packet(i) = True
      Short_Packets(i) = x
    End If
  End If
  If Science(x).Sequence_Count > Start_Sequence_Count + 7 Then
    x = No.Science_Packets
  End If
  x = x + 1
Loop Until x > No.Science_Packets


'Load the minimum and maximum ranges for the various components from
' C:\Ptolemy EGSE\Component Ranges.txt
' Numbers 1 to 5   Pressure sensors
'         6 to 10  Various other sensors
'        11 to 16  Heaters
'        17 to 21  Large Reactors
'        22 to 26  Small Reactors and
'        27 to 31  L-Valves
'        32,   33  5V voltage and current
'        34,   35  28V voltage and current
'        36        RF frequency

Range_File_Number = FreeFile
Range_File_Name = Ptolemy_Directory & "\EGSE_Info\Test Ranges\"
```

```
ShortTest - 2

Range_File_Name = Range_File_Name & Packet_Format & " " & Test_Type & " Mode.txt"
Open Range_File_Name For Input As #Range_File_Number
Dim dummy As String
For i = 1 To 38
  Input #Range_File_Number, dummy, Min_5V(i), Max_5V(i)
  Input #Range_File_Number, Min_28V(i), Max_28V(i)
  Input #Range_File_Number, Min_Range(i), Max_Range(i)
Next i
Close #Range_File_Number

Load_Test_Data

Select Case Test_Type
  Case "Cruise Phase"
    Display_Cruise_Results
  Case "EAFT"
    Display_Limited_Cruise_Results
  Case Else
    Stop
  End Select

End Sub

Public Sub Load_Test_Data()
Dim i, j As Integer
Dim Number_Aux_Records As Integer
Dim Start_Position, ST_Position As Long

File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #File_Number

For i = 1 To 8
  If Short_Packets(i) <> 0 Then
    Start_Position = Science(Short_Packets(i)).Start
    Get_Word (Start_Position + 18)
    Number_Aux_Records = Word_Result

    For j = 1 To Number_Aux_Records
      ST_Position = (i - 1) * 29 + j
      Get_Word (Start_Position + j * 8 + 16)
      Channel_Numbers(ST_Position) = Get_Sensor_ID(Word_Result)
      Get_Word (Start_Position + j * 8 + 18)
      Short_Test_Words(ST_Position) = Word_Result
      If Word_Result > 32767 Then
        Word_Result = Word_Result - 65536
      End If
      Short_Test_Voltages(ST_Position) = Word_Result * 10 / 32768
    Next j
  End If
Next i

'Find the information packet that contains the RF calibration report
Dim x As Integer

RF_Calibration_Found = False
x = 1
Do
  If Information(x).Time > Start_time And Information(x).Time < Start_time + 900 Then
    Get_Word (Information(x).Start)
    If Word_Result = &HF37 Then  'If Progress Event Report
      Get_Word (Information(x).Start + 16)
      If Word_Result = 55113 Then  'If this is an RF calibration report then get RF frequency word
        If RF_Calibration_Found = False Then
          RF_Calibration_Found = True
          Get_Word (Information(x).Start + 56)
          RF_Frequency = Word_Result And &HFFF
          RF_Frequency = Convert_RF_Word_to_Frequency(RF_Frequency)
          Label11(4).Caption = Format(RF_Frequency, "0.00")
          x = No.Information_Packets
        End If
      End If
    End If
  End If
  x = x + 1
Loop Until x > No.Information_Packets

Close #File_Number
```

```
ShortTest - 3

End Sub

Public Function Get_Sensor_ID(ChanID)
  Dim Component, k As Integer
  Component = 0
  For k = 1 To Number_of_Sensors
    If Sensor(k).ADC_Channel = ChanID Then
      Component = k
    End If
  Next k
  Get_Sensor_ID = Component
End Function

Public Sub Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte, Low_Byte As Long

Get #File_Number, Byte_Position, First_Byte
Get #File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Word_Result = High_Byte * 256 + Low_Byte

End Sub

Public Sub Display_Cruise_Results()
Dim AD590_Temperature As Double
Dim i As Integer

'Determine the AD590 temperature (the 6th and 224th readings)
Dim AD5901, AD5902 As Boolean
AD5901 = False: AD5902 = False
If Channel_Numbers(6) = 28 Then
  Call Get_Sensor_Reading(Channel_Numbers(6), Short_Test_Voltages(6), Short_Test_Results(6), Short_
Test_Units(6))
  AD5901 = True
End If
If Channel_Numbers(224) = 28 Then
  Call Get_Sensor_Reading(Channel_Numbers(224), Short_Test_Voltages(224), Short_Test_Results(224),
Short_Test_Units(224))
  AD5902 = True
End If

If AD5901 = True And AD5902 = True Then
  AD590_Temperature = (Short_Test_Results(6) + Short_Test_Results(224)) / 2
End If
If AD5901 = True And AD5902 = False Then
  AD590_Temperature = Short_Test_Results(6)
End If
If AD5901 = False And AD5902 = True Then
  AD590_Temperature = Short_Test_Results(224)
End If
If AD5901 = False And AD5902 = False Then
  AD590_Temperature = 0
End If
Call Convert_Temperature_to_EMF(AD590_Temperature, AD590_EMF)

Dim Temp_voltage As Double
For i = 1 To 232
  Select Case Channel_Numbers(i)
    Case 1 To 21, 27
      Temp_voltage = Short_Test_Voltages(i) + AD590_EMF / 10000
      Call Get_Sensor_Reading(Channel_Numbers(i), Temp_voltage, Short_Test_Results(i), Short_Test_U
nits(i))
    Case Else
      Call Get_Sensor_Reading(Channel_Numbers(i), Short_Test_Voltages(i), Short_Test_Results(i), Sh
ort_Test_Units(i))
  End Select
Next i
```

```
Display_Cruise_Sensor_Results

Display_Valve_Results

Display_InjectorValve_Results

Display_Heater_Results

Display_Large_Reactor_Results

Display_Small_Reactor_Results

Display_Cruise_L_Valve_Results

Display_Cruise_Summary_Results

End Sub
Public Sub Display_Limited_Cruise_Results()
Dim AD590_Temperature As Double
Dim i As Integer
Label38.Caption = "Extended AFT"
'Determine the AD590 temperature (the 6th and 212th readings)
Dim AD5901, AD5902 As Boolean
AD5901 = False: AD5902 = False
If Channel_Numbers(6) = 28 Then
  Call Get_Sensor_Reading(Channel_Numbers(6), Short_Test_Voltages(6), Short_Test_Results(6), Short_
Test_Units(6))
  AD5901 = True
End If
If Channel_Numbers(212) = 28 Then
  Call Get_Sensor_Reading(Channel_Numbers(212), Short_Test_Voltages(212), Short_Test_Results(212),
Short_Test_Units(212))
  AD5902 = True
End If

If AD5901 = True And AD5902 = True Then
  AD590_Temperature = (Short_Test_Results(6) + Short_Test_Results(212)) / 2
End If
If AD5901 = True And AD5902 = False Then
  AD590_Temperature = Short_Test_Results(6)
End If
If AD5901 = False And AD5902 = True Then
  AD590_Temperature = Short_Test_Results(212)
End If
If AD5901 = False And AD5902 = False Then
  AD590_Temperature = 0
End If
Call Convert_Temperature_to_EMF(AD590_Temperature, AD590_EMF)

Dim Temp_voltage As Double
For i = 1 To 232
  Select Case Channel_Numbers(i)
    Case 1 To 21, 27
      Temp_voltage = Short_Test_Voltages(i) + AD590_EMF / 10000
      Call Get_Sensor_Reading(Channel_Numbers(i), Temp_voltage, Short_Test_Results(i), Short_Test_U
nits(i))
    Case Else
      Call Get_Sensor_Reading(Channel_Numbers(i), Short_Test_Voltages(i), Short_Test_Results(i), Sh
ort_Test_Units(i))
  End Select
Next i

Display_Limited_Cruise_Sensor_Results

Display_Valve_Results

Display_InjectorValve_Results

Display_Heater_Results

Display_Large_Reactor_Results

Display_Small_Reactor_Results

Display_Limited_Cruise_L_Valve_Results
```

```
ShortTest - 5

Display_Limited_Cruise_Summary_Results

End Sub
Public Sub Display_Cruise_Sensor_Results()
Dim i As Integer

'numbers 1 to 5, Pressure sensors,
'number 6, AD590 Temperature,
'number 7, Docking Station,
'number 8, Nano-tips,
'number 9, Detector voltage and
'number 10, RF calibration
For i = 1 To 10
  Label4(i - 1).Caption = Format(Short_Test_Results(i), "0.00")
  Label5(i - 1).Caption = Format(Short_Test_Results(218 + i), "0.00")
Next i

'Pass criteria:
'Pressure gauges, Pressure > -0.2 bar and less than 1.2 bar
'AD590 Temperature > -70oC and < 70oC
'      change in temperature less than 20oC
'Docking station, Position > 0mm and < 20mm
'Nano-tips, Current > -10uA and < -10uA
'Detector voltage, Voltage > -0.1 and < 0.1kV
'RF calibration, Voltage > -10V and < 10V

'Pressure sensors Test
For i = 1 To 5
  Result = True
  If Short_Test_Results(i) < Min_Range(i) Then Result = False
  If Short_Test_Results(i) > Max_Range(i) Then Result = False
  If Short_Test_Results(218 + i) < Min_Range(i) Then Result = False
  If Short_Test_Results(218 + i) > Max_Range(i) Then Result = False
  If Result = False Then
    Label7(i - 1).ForeColor = RGB(255, 0, 0)
    Label7(i - 1).Caption = "Fail"
  End If
Next i

'AD590 Test
Result = True
If Short_Test_Results(6) < Min_Range(6) Then Result = False
If Short_Test_Results(6) > Max_Range(6) Then Result = False
If Short_Test_Results(224) < Min_Range(6) Then Result = False
If Short_Test_Results(224) > Max_Range(6) Then Result = False
If Abs(Short_Test_Results(6) - Short_Test_Results(224)) > 20 Then Result = False
If Result = False Then
  Label7(5).ForeColor = RGB(255, 0, 0)
  Label7(5).Caption = "Fail"
End If

'Docking station test
Result = True
If Short_Test_Results(7) < Min_Range(7) Then Result = False
If Short_Test_Results(7) > Max_Range(7) Then Result = False
If Short_Test_Results(225) < Min_Range(7) Then Result = False
If Short_Test_Results(225) > Max_Range(7) Then Result = False
If Result = False Then
  Label7(6).ForeColor = RGB(255, 0, 0)
  Label7(6).Caption = "Fail"
End If

'Nano-tips
Result = True
If Short_Test_Results(8) < Min_Range(8) Then Result = False
If Short_Test_Results(8) > Max_Range(8) Then Result = False
If Short_Test_Results(226) < Min_Range(8) Then Result = False
If Short_Test_Results(226) > Max_Range(8) Then Result = False
If Result = False Then
  Label7(7).ForeColor = RGB(255, 0, 0)
  Label7(7).Caption = "Fail"
End If

'Detector voltage
Result = True
If Short_Test_Results(9) < Min_Range(9) Then Result = False
If Short_Test_Results(9) > Max_Range(9) Then Result = False
```

```
ShortTest - 6

If Short_Test_Results(227) < Min_Range(9) Then Result = False
If Short_Test_Results(227) > Max_Range(9) Then Result = False
If Result = False Then
  Label7(8).ForeColor = RGB(255, 0, 0)
  Label7(8).Caption = "Fail"
End If

'RF Calibration
Result = True
If Short_Test_Results(10) < Min_Range(10) Then Result = False
If Short_Test_Results(10) > Max_Range(10) Then Result = False
If Short_Test_Results(228) < Min_Range(10) Then Result = False
If Short_Test_Results(228) > Max_Range(10) Then Result = False
If Result = False Then
  Label7(9).ForeColor = RGB(255, 0, 0)
  Label7(9).Caption = "Fail"
End If

End Sub
Public Sub Display_Limited_Cruise_Sensor_Results()
Dim i As Integer

'numbers 1 to 5, Pressure sensors,
'number 6, AD590 Temperature,
'number 7, Docking Station,
'number 8, Nano-tips,
'number 9, Detector voltage and
'number 10, RF calibration
For i = 1 To 10
  Label4(i - 1).Caption = Format(Short_Test_Results(i), "0.00")
  Label5(i - 1).Caption = Format(Short_Test_Results(206 + i), "0.00")
Next i

'Pass criteria:
'Pressure gauges, Pressure > -0.2 bar and less than 1.2 bar
'AD590 Temperature > -70oC and < 70oC
'      change in temperature less than 20oC
'Docking station, Position > 0mm and < 20mm
'Nano-tips, Current > -10uA and < -10uA
'Detector voltage, Voltage > -0.1 and < 0.1kV
'RF calibration, Voltage > -10V and < 10V

'Pressure sensors Test
For i = 1 To 5
  Result = True
  If Short_Test_Results(i) < Min_Range(i) Then Result = False
  If Short_Test_Results(i) > Max_Range(i) Then Result = False
  If Short_Test_Results(206 + i) < Min_Range(i) Then Result = False
  If Short_Test_Results(206 + i) > Max_Range(i) Then Result = False
  If Result = False Then
    Label7(i - 1).ForeColor = RGB(255, 0, 0)
    Label7(i - 1).Caption = "Fail"
  End If
Next i

'AD590 Test
Result = True
If Short_Test_Results(6) < Min_Range(6) Then Result = False
If Short_Test_Results(6) > Max_Range(6) Then Result = False
If Short_Test_Results(212) < Min_Range(6) Then Result = False
If Short_Test_Results(212) > Max_Range(6) Then Result = False
If Abs(Short_Test_Results(6) - Short_Test_Results(212)) > 20 Then Result = False
If Result = False Then
  Label7(5).ForeColor = RGB(255, 0, 0)
  Label7(5).Caption = "Fail"
End If

'Docking station test
Result = True
If Short_Test_Results(7) < Min_Range(7) Then Result = False
If Short_Test_Results(7) > Max_Range(7) Then Result = False
If Short_Test_Results(213) < Min_Range(7) Then Result = False
If Short_Test_Results(213) > Max_Range(7) Then Result = False
If Result = False Then
  Label7(6).ForeColor = RGB(255, 0, 0)
  Label7(6).Caption = "Fail"
End If
```

```
'Nano-tips
Result = True
If Short_Test_Results(8) < Min_Range(8) Then Result = False
If Short_Test_Results(8) > Max_Range(8) Then Result = False
If Short_Test_Results(214) < Min_Range(8) Then Result = False
If Short_Test_Results(214) > Max_Range(8) Then Result = False
If Result = False Then
  Label7(7).ForeColor = RGB(255, 0, 0)
  Label7(7).Caption = "Fail"
End If

'Detector voltage
Result = True
If Short_Test_Results(9) < Min_Range(9) Then Result = False
If Short_Test_Results(9) > Max_Range(9) Then Result = False
If Short_Test_Results(215) < Min_Range(9) Then Result = False
If Short_Test_Results(215) > Max_Range(9) Then Result = False
If Result = False Then
  Label7(8).ForeColor = RGB(255, 0, 0)
  Label7(8).Caption = "Fail"
End If

'RF Calibration
Result = True
If Short_Test_Results(10) < Min_Range(10) Then Result = False
If Short_Test_Results(10) > Max_Range(10) Then Result = False
If Short_Test_Results(216) < Min_Range(10) Then Result = False
If Short_Test_Results(216) > Max_Range(10) Then Result = False
If Result = False Then
  Label7(9).ForeColor = RGB(255, 0, 0)
  Label7(9).Caption = "Fail"
End If

End Sub

Public Sub Display_Valve_Results()
Dim i As Integer
Dim Current_5V_On, Current_5V_Off As Double
Dim Current_28V_On, Current_28V_Off As Double
Dim Valve_28V_Current, Valve_5V_Current As Double

For i = 1 To 12
  Current_5V_Off = (Short_Test_Results(i * 7 + 18) + Short_Test_Results(i * 7 + 25)) / 2
  Current_28V_Off = (Short_Test_Results(i * 7 + 19) + Short_Test_Results(i * 7 + 26)) / 2
  Current_5V_On = (Short_Test_Results(i * 7 + 21) + Short_Test_Results(i * 7 + 23)) / 2
  Current_28V_On = (Short_Test_Results(i * 7 + 22) + Short_Test_Results(i * 7 + 24)) / 2
  Valve_28V_Current = Current_28V_On - Current_28V_Off
  Valve_5V_Current = Current_5V_On - Current_5V_Off
  Label15(i - 1).Caption = Format(Valve_28V_Current, "0.0")
  Label16(i - 1).Caption = Format(Valve_5V_Current, "0.0")

'Valve Test criteria
'5V current > -10mA and < 10mA
'28V current > 20mA and < 40mA
  Result = True
  If Valve_5V_Current < Min_5V(37) Or Valve_5V_Current > Max_5V(37) Then Result = False
  If Valve_28V_Current < Min_28V(37) Or Valve_28V_Current > Max_28V(37) Then Result = False
  If Result = False Then
    Label17(i - 1).ForeColor = RGB(255, 0, 0)
    Label17(i - 1).Caption = "Fail"
  End If

Next i

End Sub

Public Sub Display_InjectorValve_Results()
Dim i As Integer
Dim Current_5V_On, Current_5V_Off As Double
Dim Current_28V_On, Current_28V_Off As Double
Dim Inj_Valve_28V_Current, Inj_Valve_5V_Current As Double

For i = 1 To 3
  Current_5V_Off = (Short_Test_Results(i * 4 + 9) + Short_Test_Results(i * 4 + 13)) / 2
  Current_28V_Off = (Short_Test_Results(i * 4 + 10) + Short_Test_Results(i * 4 + 14)) / 2
  Current_5V_On = Short_Test_Results(i * 4 + 11)
```

```
  Current_28V_On = Short_Test_Results(i * 4 + 12)
  Inj_Valve_28V_Current = Current_28V_On - Current_28V_Off
  Inj_Valve_5V_Current = Current_5V_On - Current_5V_Off
  Label21(i - 1).Caption = Format(Inj_Valve_28V_Current, "0.0")
  Label22(i - 1).Caption = Format(Inj_Valve_5V_Current, "0.0")

'Injector Valve Test criteria
'5V current > -10mA and < 10mA
'28V current > 70mA and < 120mA
  Result = True
  If Inj_Valve_5V_Current < Min_5V(38) Or Inj_Valve_5V_Current > Max_5V(38) Then Result = False
  If Inj_Valve_28V_Current < Min_28V(38) Or Inj_Valve_28V_Current > Max_28V(38) Then Result = False
  If Result = False Then
    Label23(i - 1).ForeColor = RGB(255, 0, 0)
    Label23(i - 1).Caption = "Fail"
  End If

Next i

End Sub

Public Sub Display_Heater_Results()
Dim i As Integer
Dim Current_5V_On, Current_5V_Off As Double
Dim Current_28V_On, Current_28V_Off As Double
Dim Heater_28V_Current, Heater_5V_Current As Double
Dim Temperature_increase As Double

For i = 1 To 6
  Current_5V_Off = (Short_Test_Results(i * 8 + 101) + Short_Test_Results(i * 8 + 109)) / 2
  Current_28V_Off = (Short_Test_Results(i * 8 + 102) + Short_Test_Results(i * 8 + 110)) / 2
  Current_5V_On = (Short_Test_Results(i * 8 + 104) + Short_Test_Results(i * 8 + 106)) / 2
  Current_28V_On = (Short_Test_Results(i * 8 + 105) + Short_Test_Results(i * 8 + 107)) / 2
  Heater_28V_Current = Current_28V_On - Current_28V_Off
  Heater_5V_Current = Current_5V_On - Current_5V_Off
  Label26(i - 1).Caption = Format(Heater_28V_Current, "0.0")
  Label27(i - 1).Caption = Format(Heater_5V_Current, "0.0")
  Label28(i - 1).Caption = Format(Short_Test_Results(i * 8 + 103), "0.0")
  Label29(i - 1).Caption = Format(Short_Test_Results(i * 8 + 108), "0.0")
  Temperature_increase = Short_Test_Results(i * 8 + 108) - Short_Test_Results(i * 8 + 103)
  Label30(i - 1).Caption = Format(Temperature_increase, "0.0")

'Heater test criteria:
'  5V current > Min_5V and < Max_5V
'  28V current > Min_28V and < Max_28V
'  Temperature increase within expected range for the heater
  Result = True
  If Heater_5V_Current < Min_5V(i + 10) Then Result = False
  If Heater_5V_Current > Max_5V(i + 10) Then Result = False
  If Heater_28V_Current < Min_28V(i + 10) Then Result = False
  If Heater_28V_Current > Max_28V(i + 10) Then Result = False
  If Temperature_increase < Min_Range(i + 10) Then Result = False
  If Temperature_increase > Max_Range(i + 10) Then Result = False
  If Result = False Then
    Label31(i - 1).ForeColor = RGB(255, 0, 0)
    Label31(i - 1).Caption = "Fail"
  End If

Next i

End Sub

Public Sub Display_Large_Reactor_Results()
Dim i As Integer
Dim Temperature_increase As Double

For i = 1 To 5
  Label28(i + 5).Caption = Format(Short_Test_Results(i + 158), "0.0")
  Label29(i + 5).Caption = Format(Short_Test_Results(i + 163), "0.0")
  Temperature_increase = Short_Test_Results(i + 163) - Short_Test_Results(i + 158)
  Label30(i + 5).Caption = Format(Temperature_increase, "0.0")


'Large reactor test criteria:
'  Temperature increase within expected range for the reactor
  Result = True
  If Temperature_increase < Min_Range(i + 16) Then Result = False
```

```
  If Temperature_increase > Max_Range(i + 16) Then Result = False
  If Result = False Then
    Label31(i + 5).ForeColor = RGB(255, 0, 0)
    Label31(i + 5).Caption = "Fail"
  End If

Next i

End Sub

Public Sub Display_Small_Reactor_Results()
Dim i As Integer
Dim Temperature_increase As Double

For i = 1 To 5
  Label28(i + 10).Caption = Format(Short_Test_Results(i + 168), "0.0")
  Label29(i + 10).Caption = Format(Short_Test_Results(i + 173), "0.0")
  Temperature_increase = Short_Test_Results(i + 173) - Short_Test_Results(i + 168)
  Label30(i + 10).Caption = Format(Temperature_increase, "0.0")

'Small reactor test criteria:
'  Temperature increase within expected range for the reactor
  Result = True
  If Temperature_increase < Min_Range(i + 21) Then Result = False
  If Temperature_increase > Max_Range(i + 21) Then Result = False
  If Result = False Then
    Label31(i + 10).ForeColor = RGB(255, 0, 0)
    Label31(i + 10).Caption = "Fail"
  End If

Next i

End Sub

Public Sub Display_Cruise_L_Valve_Results()
Dim i As Integer
Dim Current_5V_On, Current_5V_Off As Double
Dim Current_28V_On, Current_28V_Off As Double
Dim L_Valve_28V_Current, L_Valve_5V_Current As Double
Dim Temperature_increase As Double

For i = 1 To 5
  If i = 5 Then   'L-Valve 7 only has one current measurement
    Current_5V_Off = Short_Test_Results(i * 8 + 171)
    Current_28V_Off = Short_Test_Results(i * 8 + 172)
  Else
    Current_5V_Off = (Short_Test_Results(i * 8 + 171) + Short_Test_Results(i * 8 + 179)) / 2
    Current_28V_Off = (Short_Test_Results(i * 8 + 172) + Short_Test_Results(i * 8 + 180)) / 2
  End If
  Current_5V_On = (Short_Test_Results(i * 8 + 174) + Short_Test_Results(i * 8 + 176)) / 2
  Current_28V_On = (Short_Test_Results(i * 8 + 175) + Short_Test_Results(i * 8 + 177)) / 2
  L_Valve_28V_Current = Current_28V_On - Current_28V_Off
  L_Valve_5V_Current = Current_5V_On - Current_5V_Off
  Label26(i + 15).Caption = Format(L_Valve_28V_Current, "0.0")
  Label27(i + 15).Caption = Format(L_Valve_5V_Current, "0.0")
  Label28(i + 15).Caption = Format(Short_Test_Results(i * 8 + 173), "0.0")
  Label29(i + 15).Caption = Format(Short_Test_Results(i * 8 + 178), "0.0")
  Temperature_increase = Short_Test_Results(i * 8 + 178) - Short_Test_Results(i * 8 + 173)
  Label30(i + 15).Caption = Format(Temperature_increase, "0.0")

'L-Valve test criteria:
'  5V current > Min_5V and < Max_5V
'  28V current > Min_28V and < Max_28V
'  Temperature increase within expected range for the L-valve
  Result = True
  If L_Valve_5V_Current < Min_5V(i + 26) Then Result = False
  If L_Valve_5V_Current > Max_5V(i + 26) Then Result = False
  If L_Valve_28V_Current < Min_28V(i + 26) Then Result = False
  If L_Valve_28V_Current > Max_28V(i + 26) Then Result = False
  If Temperature_increase < Min_Range(i + 26) Then Result = False
  If Temperature_increase > Max_Range(i + 26) Then Result = False
  If Result = False Then
    Label31(i + 15).ForeColor = RGB(255, 0, 0)
    Label31(i + 15).Caption = "Fail"
  End If

Next i
```

```
ShortTest - 10


End Sub
Public Sub Display_Limited_Cruise_L_Valve_Results()
Dim i As Integer
Dim Current_5V_On, Current_5V_Off As Double
Dim Current_28V_On, Current_28V_Off As Double
Dim L_Valve_28V_Current, L_Valve_5V_Current As Double
Dim Temperature_increase As Double

For i = 1 To 5
  Select Case i
    Case 1, 2
      Current_5V_Off = (Short_Test_Results(179) + Short_Test_Results(189)) / 2
      Current_28V_Off = (Short_Test_Results(180) + Short_Test_Results(190)) / 2
      Current_5V_On = (Short_Test_Results(183) + Short_Test_Results(185)) / 2
      Current_28V_On = (Short_Test_Results(184) + Short_Test_Results(186)) / 2
    Case 3, 4
      Current_5V_Off = (Short_Test_Results(189) + Short_Test_Results(199)) / 2
      Current_28V_Off = (Short_Test_Results(190) + Short_Test_Results(200)) / 2
      Current_5V_On = (Short_Test_Results(193) + Short_Test_Results(195)) / 2
      Current_28V_On = (Short_Test_Results(194) + Short_Test_Results(196)) / 2
    Case 5
      Current_5V_Off = Short_Test_Results(199)
      Current_28V_Off = Short_Test_Results(200)
      Current_5V_On = (Short_Test_Results(202) + Short_Test_Results(204)) / 2
      Current_28V_On = (Short_Test_Results(203) + Short_Test_Results(205)) / 2
  End Select

  L_Valve_28V_Current = Current_28V_On - Current_28V_Off
  L_Valve_5V_Current = Current_5V_On - Current_5V_Off
  'Assume that for L-valves 1 to 4 that the current is divided equally
  'between the 2 L-valves as they are operated simultaneously
  If i <> 5 Then
    L_Valve_28V_Current = L_Valve_28V_Current / 2
    L_Valve_5V_Current = L_Valve_5V_Current / 2
  End If
  Label26(i + 15).Caption = Format(L_Valve_28V_Current, "0.0")
  Label27(i + 15).Caption = Format(L_Valve_5V_Current, "0.0")
  Select Case i
    Case 1
      Temperature_increase = Short_Test_Results(187) - Short_Test_Results(181)
      Label28(16).Caption = Format(Short_Test_Results(181), "0.0")
      Label29(16).Caption = Format(Short_Test_Results(187), "0.0")
    Case 2
      Temperature_increase = Short_Test_Results(188) - Short_Test_Results(182)
      Label28(17).Caption = Format(Short_Test_Results(182), "0.0")
      Label29(17).Caption = Format(Short_Test_Results(188), "0.0")
    Case 3
      Temperature_increase = Short_Test_Results(197) - Short_Test_Results(191)
      Label28(18).Caption = Format(Short_Test_Results(191), "0.0")
      Label29(18).Caption = Format(Short_Test_Results(197), "0.0")
    Case 4
      Temperature_increase = Short_Test_Results(198) - Short_Test_Results(192)
      Label28(19).Caption = Format(Short_Test_Results(192), "0.0")
      Label29(19).Caption = Format(Short_Test_Results(198), "0.0")
    Case 5
      Temperature_increase = Short_Test_Results(206) - Short_Test_Results(201)
      Label28(20).Caption = Format(Short_Test_Results(201), "0.0")
      Label29(20).Caption = Format(Short_Test_Results(206), "0.0")
  End Select
  Label30(i + 15).Caption = Format(Temperature_increase, "0.0")

'L-Valve test criteria:
'  5V current > Min_5V and < Max_5V
'  28V current > Min_28V and < Max_28V
'  Temperature increase within expected range for the L-valve
  Result = True
  If L_Valve_5V_Current < Min_5V(i + 26) Then Result = False
  If L_Valve_5V_Current > Max_5V(i + 26) Then Result = False
  If L_Valve_28V_Current < Min_28V(i + 26) Then Result = False
  If L_Valve_28V_Current > Max_28V(i + 26) Then Result = False
  If Temperature_increase < Min_Range(i + 26) Then Result = False
  If Temperature_increase > Max_Range(i + 26) Then Result = False
  If Result = False Then
    Label31(i + 15).ForeColor = RGB(255, 0, 0)
    Label31(i + 15).Caption = "Fail"
  End If
```

```
ShortTest - 11


Next i

End Sub
Public Sub Display_Cruise_Summary_Results()
Dim i As Integer

Dim Voltage_5V, Current_5V As Double
Dim Voltage_28V, Current_28V As Double

'5V supply voltage
Voltage_5V = (Short_Test_Results(11) + Short_Test_Results(229)) / 2
Label11(0).Caption = Format(Voltage_5V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Voltage_5V < Min_Range(32) Then Result = False
If Voltage_5V > Max_Range(32) Then Result = False
If Result = False Then
  Label33(0).ForeColor = RGB(255, 0, 0)
  Label33(0).Caption = "Fail"
End If

'5V supply current
Current_5V = (Short_Test_Results(13) + Short_Test_Results(17) + Short_Test_Results(21) + Short_Test
_Results(25)) / 4
Label11(1).Caption = Format(Current_5V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Current_5V < Min_Range(33) Then Result = False
If Current_5V > Max_Range(33) Then Result = False
If Result = False Then
  Label33(1).ForeColor = RGB(255, 0, 0)
  Label33(1).Caption = "Fail"
End If

'28V supply voltage
Voltage_28V = (Short_Test_Results(12) + Short_Test_Results(230)) / 2
Label11(2).Caption = Format(Voltage_28V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Voltage_28V < Min_Range(34) Then Result = False
If Voltage_28V > Max_Range(34) Then Result = False
If Result = False Then
  Label33(2).ForeColor = RGB(255, 0, 0)
  Label33(2).Caption = "Fail"
End If

'28V supply current
Current_28V = (Short_Test_Results(14) + Short_Test_Results(18) + Short_Test_Results(22) + Short_Tes
t_Results(26)) / 4
Label11(3).Caption = Format(Current_28V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Current_28V < Min_Range(35) Then Result = False
If Current_28V > Max_Range(35) Then Result = False
If Result = False Then
  Label33(3).ForeColor = RGB(255, 0, 0)
  Label33(3).Caption = "Fail"
End If

'Check that the RF frequency is within expected frequency
If RF_Calibration_Found = True Then
  Result = True
  If RF_Frequency < Min_Range(36) Then Result = False
  If RF_Frequency > Max_Range(36) Then Result = False
Else
  Result = False
  Label11(4).Caption = "n.m."
End If
If Result = False Then
  Label33(4).ForeColor = RGB(255, 0, 0)
  Label33(4).Caption = "Fail"
End If

'Check each sub-system component for pass/fail satatus

Result = True
```

```
ShortTest - 12


'Sensors
For i = 0 To 9
  If Label7(i).Caption <> "Pass" Then
    Label2(0).ForeColor = RGB(255, 0, 0)
    Label2(0).Caption = "Fail"
  End If
Next i

'Valves
For i = 0 To 11
  If Label17(i).Caption <> "Pass" Then
    Label2(1).ForeColor = RGB(255, 0, 0)
    Label2(1).Caption = "Fail"
  End If
Next i

'Injector Valves
For i = 0 To 2
  If Label23(i).Caption <> "Pass" Then
    Label2(2).ForeColor = RGB(255, 0, 0)
    Label2(2).Caption = "Fail"
  End If
Next i

'Heaters
For i = 0 To 5
  If Label31(i).Caption <> "Pass" Then
    Label2(3).ForeColor = RGB(255, 0, 0)
    Label2(3).Caption = "Fail"
  End If
Next i

'Large Reactors
For i = 6 To 10
  If Label31(i).Caption <> "Pass" Then
    Label2(4).ForeColor = RGB(255, 0, 0)
    Label2(4).Caption = "Fail"
  End If
Next i

'Small Reactors
For i = 11 To 15
  If Label31(i).Caption <> "Pass" Then
    Label2(5).ForeColor = RGB(255, 0, 0)
    Label2(5).Caption = "Fail"
  End If
Next i

'L-Valves
For i = 16 To 20
  If Label31(i).Caption <> "Pass" Then
    Label2(6).ForeColor = RGB(255, 0, 0)
    Label2(6).Caption = "Fail"
  End If
Next i

'Voltages and RF
For i = 0 To 4
  If Label33(i).Caption <> "Pass" Then
    Label2(7).ForeColor = RGB(255, 0, 0)
    Label2(7).Caption = "Fail"
  End If
Next i

'Print Overall Pass/Fail result
Result = True
For i = 0 To 7
  If Label2(i).Caption <> "Pass" Then Result = False
Next i
If Result = False Then
  Label35.ForeColor = RGB(255, 0, 0)
  Label35.Caption = "Fail"
End If

'Determine whether all science data packets were collected
Dim Missing_Packets As Integer
```

```
ShortTest - 13

Dim rpt_str As String

Missing_Packets = 0
For i = 1 To 8
  If Found_Packet(i) = False Then
    Missing_Packets = Missing_Packets + 1
  End If
Next i
If Missing_Packets > 0 Then
  If Missing_Packets = 1 Then
    rpt_str = "There was 1 missing science data packet (pkt no. "
  Else
    rpt_str = "There were " & Missing_Packets & " missing science data packets (pkt nos. "
  End If
  For i = 1 To 8
    If Found_Packet(i) = 0 Then
      Select Case Missing_Packets
        Case 1
          rpt_str = rpt_str & i & ")"
        Case 2
          rpt_str = rpt_str & i & " and "
        Case Else
          rpt_str = rpt_str & i & ", "
      End Select
      Missing_Packets = Missing_Packets - 1
    End If
  Next i
  Label37.ForeColor = RGB(255, 0, 0)
  Label37.Caption = rpt_str
End If
End Sub
Public Sub Display_Limited_Cruise_Summary_Results()
Dim i As Integer

Dim Voltage_5V, Current_5V As Double
Dim Voltage_28V, Current_28V As Double

'5V supply voltage
Voltage_5V = (Short_Test_Results(11) + Short_Test_Results(217)) / 2
Label11(0).Caption = Format(Voltage_5V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Voltage_5V < Min_Range(32) Then Result = False
If Voltage_5V > Max_Range(32) Then Result = False
If Result = False Then
  Label33(0).ForeColor = RGB(255, 0, 0)
  Label33(0).Caption = "Fail"
End If

'5V supply current
Current_5V = (Short_Test_Results(13) + Short_Test_Results(17) + Short_Test_Results(21) + Short_Test
_Results(25)) / 4
Label11(1).Caption = Format(Current_5V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Current_5V < Min_Range(33) Then Result = False
If Current_5V > Max_Range(33) Then Result = False
If Result = False Then
  Label33(1).ForeColor = RGB(255, 0, 0)
  Label33(1).Caption = "Fail"
End If

'28V supply voltage
Voltage_28V = (Short_Test_Results(12) + Short_Test_Results(218)) / 2
Label11(2).Caption = Format(Voltage_28V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Voltage_28V < Min_Range(34) Then Result = False
If Voltage_28V > Max_Range(34) Then Result = False
If Result = False Then
  Label33(2).ForeColor = RGB(255, 0, 0)
  Label33(2).Caption = "Fail"
End If

'28V supply current
Current_28V = (Short_Test_Results(14) + Short_Test_Results(18) + Short_Test_Results(22) + Short_Tes
t_Results(26)) / 4
```

```
Label11(3).Caption = Format(Current_28V, "0.00")
'Check the voltage rails, pass criteria
Result = True
If Current_28V < Min_Range(35) Then Result = False
If Current_28V > Max_Range(35) Then Result = False
If Result = False Then
  Label33(3).ForeColor = RGB(255, 0, 0)
  Label33(3).Caption = "Fail"
End If

'RF frequency is not measured
  Label11(4).Caption = "n.m."

'Check each sub-system component for pass/fail satatus

Result = True

'Sensors
For i = 0 To 9
  If Label7(i).Caption <> "Pass" Then
    Label2(0).ForeColor = RGB(255, 0, 0)
    Label2(0).Caption = "Fail"
  End If
Next i

'Valves
For i = 0 To 11
  If Label17(i).Caption <> "Pass" Then
    Label2(1).ForeColor = RGB(255, 0, 0)
    Label2(1).Caption = "Fail"
  End If
Next i

'Injector Valves
For i = 0 To 2
  If Label23(i).Caption <> "Pass" Then
    Label2(2).ForeColor = RGB(255, 0, 0)
    Label2(2).Caption = "Fail"
  End If
Next i

'Heaters
For i = 0 To 5
  If Label31(i).Caption <> "Pass" Then
    Label2(3).ForeColor = RGB(255, 0, 0)
    Label2(3).Caption = "Fail"
  End If
Next i

'Large Reactors
For i = 6 To 10
  If Label31(i).Caption <> "Pass" Then
    Label2(4).ForeColor = RGB(255, 0, 0)
    Label2(4).Caption = "Fail"
  End If
Next i

'Small Reactors
For i = 11 To 15
  If Label31(i).Caption <> "Pass" Then
    Label2(5).ForeColor = RGB(255, 0, 0)
    Label2(5).Caption = "Fail"
  End If
Next i

'L-Valves
For i = 16 To 20
  If Label31(i).Caption <> "Pass" Then
    Label2(6).ForeColor = RGB(255, 0, 0)
    Label2(6).Caption = "Fail"
  End If
Next i

'Voltages and RF
For i = 0 To 4
  If Label33(i).Caption <> "Pass" Then
    Label2(7).ForeColor = RGB(255, 0, 0)
```

```
    Label2(7).Caption = "Fail"
  End If
Next i

'Print Overall Pass/Fail result
Result = True
For i = 0 To 7
  If Label2(i).Caption <> "Pass" Then Result = False
Next i
If Result = False Then
  Label35.ForeColor = RGB(255, 0, 0)
  Label35.Caption = "Fail"
End If

'Determine whether all science data packets were collected
Dim Missing_Packets As Integer
Dim rpt_str As String

Missing_Packets = 0
For i = 1 To 8
  If Found_Packet(i) = False Then
    Missing_Packets = Missing_Packets + 1
  End If
Next i
If Missing_Packets > 0 Then
  If Missing_Packets = 1 Then
    rpt_str = "There was 1 missing science data packet (pkt no. "
  Else
    rpt_str = "There were " & Missing_Packets & " missing science data packets (pkt nos. "
  End If
  For i = 1 To 8
    If Found_Packet(i) = 0 Then
      Select Case Missing_Packets
        Case 1
          rpt_str = rpt_str & i & ")"
        Case 2
          rpt_str = rpt_str & i & " and "
        Case Else
          rpt_str = rpt_str & i & ", "
      End Select
      Missing_Packets = Missing_Packets - 1
    End If
  Next i
  Label37.ForeColor = RGB(255, 0, 0)
  Label37.Caption = rpt_str
End If
End Sub
```

```
Spectra_Form - 1

Option Explicit
Dim Spectrum_Displayed As Long
Dim Y_Gain, X_Gain As Double
Dim X_Max, X_Min As Integer
Dim Spectrum_Start_packet() As Long
Dim Total_Number_Spectra As Long
Dim File_Number As Integer
Dim spectrum_start_Byte As Long
Dim Bin(1024), Bin_Counter As Long
Dim Sub_Packet_viewed, Total_number_sub_packets As Long
Dim Complete_spectrum As Boolean    'True if spectrum viewed is a complete spectrum

Private Sub Command2_Click()
  X_Gain = Int(X_Gain * 2)
  X_Max = X_Min + 1024 / X_Gain
  Update_Graph
End Sub

Private Sub Command1_Click()
  X_Gain = Int(X_Gain / 2)
  X_Max = X_Min + 1024 / X_Gain
  If X_Max > 1024 Then X_Max = 1024
  X_Min = X_Max - 1024 / X_Gain
  Update_Graph
End Sub

Private Sub Command3_Click()
  Select Case Y_Gain
    Case 0.01 To 0.03, 0.1 To 0.3, 1 To 3, 10 To 30
      Y_Gain = Y_Gain * 2
    Case 0.04
      Y_Gain = 0.1
    Case 0.4
      Y_Gain = 1
    Case 4
      Y_Gain = 10
    Case 40
      Y_Gain = 100
  End Select
  Update_Graph
End Sub
Private Sub Command4_Click()
  Select Case Y_Gain
    Case 0.015 To 0.09, 0.15 To 0.9, 1.5 To 9, 15 To 90
      Y_Gain = Y_Gain / 2
    Case 0.1
      Y_Gain = 0.04
    Case 1
      Y_Gain = 0.4
    Case 10
      Y_Gain = 4
    Case 100
      Y_Gain = 40
  End Select
  Update_Graph
End Sub

Private Sub Command5_Click()
  X_Min = X_Min - 1024 / (X_Gain * 8)
  If X_Min < 0 Then X_Min = 0
  X_Max = X_Min + 1024 / X_Gain
  Update_Graph
End Sub

Private Sub Command6_Click()
  X_Max = X_Max + 1024 / (X_Gain * 8)
  If X_Max > 1024 Then X_Max = 1024
  X_Min = X_Max - 1024 / X_Gain
  Update_Graph
End Sub

Private Sub Form_Load()
  Dim i, j As Integer
  Total_Number_Spectra = No.Complete_Science_Spectra + No.Compact_Science
  ReDim Spectrum_Start_packet(Total_Number_Spectra, 10)

' Find all the positions of the Complete spectra and GC spectra
```

```
Spectra_Form - 2

  Dim GC_Counter, Complete_Counter, spectrum_sub_counter As Long
  GC_Counter = 0: Complete_Counter = 0

  For i = 1 To No.Science_Packets
    If Science(i).Description = " Isotope spectrum " Then
      If Science(i).flags And 2 Then
        Complete_Counter = Complete_Counter + 1
        Spectrum_Start_packet(Complete_Counter, 1) = i
        j = i + 1
        spectrum_sub_counter = 2
        Do Until j > No.Science_Packets
          Select Case Science(j).Description
            Case " GC spectrum      "
              j = No.Science_Packets
            Case " Isotope spectrum "
              If Science(j).flags And 2 Then
                j = No.Science_Packets
              Else
                Spectrum_Start_packet(Complete_Counter, spectrum_sub_counter) = j
                spectrum_sub_counter = spectrum_sub_counter + 1
              End If
          End Select
          j = j + 1
        Loop
      End If
    End If

    If Science(i).Description = " GC spectrum      " Then
      GC_Counter = GC_Counter + 1
      Spectrum_Start_packet(No.Complete_Science_Spectra + GC_Counter, 1) = i
    End If
  Next i

  Label3.Caption = No.Complete_Science_Spectra
  Label4.Caption = No.Compact_Science

'Enter the graph control labels
  Y_Gain = 1: X_Gain = 1
  X_Max = 1024: X_Min = 0
  Update_Graph

  If Total_Number_Spectra > 0 Then
    Spectrum_Displayed = 1
    Get_Spectrum_Data (Spectrum_Displayed)
    Draw_Graph_Axis
    Draw_Spectrum
    HScroll1.Max = No.Complete_Science_Spectra + No.Compact_Science
  Else
    Spectrum_Displayed = 0
    Draw_Graph_Axis
    Label5.ForeColor = RGB(255, 0, 0)
    Label5.Caption = "No Spectra Acquired"
    HScroll1.Enabled = False
  End If
End Sub

Public Sub Draw_Graph_Axis()
  Dim X_label, Y_Label As String
  Dim z As Single
  Dim i As Integer
  Picture1.Cls
  Picture1.Scale (-100, 1100)-(1100, -150)
  Picture1.Line (0, 0)-(0, 1000), RGB(0, 0, 0)
  Picture1.Line (0, 0)-(1024, 0), RGB(0, 0, 0)

' Print X-axis
  Picture1.FontSize = 6
  Picture1.ForeColor = RGB(0, 0, 255)
  For i = 0 To 8
    X_label = CStr(X_Min + i * 128 / X_Gain)
    z = Picture1.TextWidth(X_label)
    Picture1.PSet (i * 128 - 0.5 * z, -10), RGB(255, 255, 255)
    Picture1.Print X_label
    Picture1.Line (i * 128, 0)-(i * 128, 10), RGB(0, 0, 0)
  Next i
  Picture1.FontSize = 10
  Picture1.PSet (400, -50), RGB(255, 255, 255)
```

```
Spectra_Form - 3

  Picture1.Print "Bin number"

'Print Y-axis
  Picture1.FontSize = 6
  Picture1.ForeColor = RGB(0, 0, 255)
  For i = 1 To 10
    Picture1.Line (0, i * 100)-(5, i * 100), RGB(0, 0, 0)
    Y_Label = CStr(Int(100 * Y_Gain) * i)
    z = Picture1.TextWidth(Y_Label)
    Picture1.PSet (-5 - z, i * 100 + 20), RGB(255, 255, 255)
    Picture1.Print Y_Label
  Next i

End Sub

Public Sub Get_Spectrum_Data(S)
  Dim i As Integer
  Dim Counter As Long
  Dim Number_of_Bins, First_Bin As Long
  Dim Mantisa, Shift_Count As Long

File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #File_Number
'Initialise positions of spectrum packets

spectrum_start_Byte = Science(Spectrum_Start_packet(S, 1)).Start
'Determine if the spectrum to be displayed is a
'complete spectrum or a GC spectrum
  If S > No.Complete_Science_Spectra Then
    Label5.Caption = "GC spectrum number " & S - No.Complete_Science_Spectra
    Complete_spectrum = False
  Else
    Label5.Caption = "Complete spectrum number " & S
    Complete_spectrum = True
    Sub_Packet_viewed = 1
    Total_number_sub_packets = 0
    For i = 1 To 10
      If Spectrum_Start_packet(S, i) <> 0 Then
        Total_number_sub_packets = Total_number_sub_packets + 1
      End If
    Next i
  End If

  If Complete_spectrum = True Then
' Compile data for complete spectrum
  Counter = 1: Bin_Counter = 0
  Do
    spectrum_start_Byte = Science(Spectrum_Start_packet(S, Counter)).Start
    First_Bin = Get_Word(spectrum_start_Byte + 28)
    Number_of_Bins = Get_Word(spectrum_start_Byte + 30)
    For i = 0 To Number_of_Bins - 1
      Mantisa = Get_Word(spectrum_start_Byte + (16 + i) * 2)
      Shift_Count = Mantisa And &HF
      Mantisa = Int(Mantisa / 16)
      Bin(First_Bin + i) = Mantisa * 2 ^ Shift_Count
      Bin_Counter = Bin_Counter + 1
    Next i
    Counter = Counter + 1
  Loop Until Spectrum_Start_packet(S, Counter) = 0 Or Counter = 10

  Else
' Compile data for GC spectrum
  End If

  Close #File_Number

' Enter the spectrum information labels
  spectrum_start_Byte = Science(Spectrum_Start_packet(S, 1)).Start
  Display_Spectrum_Information (spectrum_start_Byte)

End Sub

Public Function Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
```

```
Spectra_Form - 4

Dim High_Byte, Low_Byte As Long

Get #File_Number, Byte_Position, First_Byte
Get #File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
  Low_Byte = Second_Byte
End If
Get_Word = High_Byte * 256 + Low_Byte

End Function

Public Sub Display_Spectrum_Information(SSB)
  File_Number = FreeFile
  Open Egse_Data_File_Name For Binary As #File_Number

  If Complete_spectrum = True Then
    Label14.Visible = True
    HScroll2.Enabled = True
    HScroll2.Visible = True
    Label14.Caption = "Sub-packet " & Sub_Packet_viewed & " viewed of " & Total_number_sub_packets
    HScroll2.Max = Total_number_sub_packets
  Else
    Label14.Visible = False
    HScroll2.Enabled = False
    HScroll2.Visible = False
  End If

  Dim DEU_Termination As Long
  Dim Possible_Data_Loss As Long

  DEU_Termination = Get_Word(SSB + 18) And &H8000
  Possible_Data_Loss = Get_Word(SSB + 18) And &H4000

  If DEU_Termination <> 0 Then
    Label26.ForeColor = RGB(255, 0, 0)
    Label26.Caption = "YES"
  Else
    Label26.ForeColor = RGB(0, 0, 0)
    Label26.Caption = "No"
  End If

  If Possible_Data_Loss <> 0 Then
    Label27.ForeColor = RGB(255, 0, 0)
    Label27.Caption = "YES"
  Else
    Label27.ForeColor = RGB(0, 0, 0)
    Label27.Caption = "No"
  End If

  Label24.Caption = Get_Word(SSB + 28)
  Label25.Caption = Get_Word(SSB + 30)
  Label28.Caption = Get_Word(SSB + 20)
  Label29.Caption = Get_Word(SSB + 22)
  Label30.Caption = Get_Word(SSB + 24)
  Label31.Caption = Get_Word(SSB + 26)

  Close #File_Number
End Sub

Public Sub Draw_Spectrum()
Dim Bin_Plot(1024) As Double
Dim i As Integer

  For i = 0 To 1024
    Bin_Plot(i) = Bin(i) / Y_Gain
    If Bin_Plot(i) > 2000 Then Bin_Plot(i) = 2000
  Next i

  For i = X_Min To X_Max
    If i <= Bin_Counter Then
      Picture1.Line ((i - X_Min) * X_Gain, Bin_Plot(i))-((i + 1 - X_Min) * X_Gain, Bin_Plot(i)), RG
B(255, 0, 0)
      Picture1.Line ((i + 1 - X_Min) * X_Gain, Bin_Plot(i))-((i + 1 - X_Min) * X_Gain, Bin_Plot(i +
```

```
 1)), RGB(255, 0, 0)
    End If
  Next i

End Sub

Public Sub Update_Graph()
'This routine enters the values into the graph control frame.
'If any buttons are at their maximum range then that button is
' disabled
  Command1.Enabled = True: Command2.Enabled = True
  Command3.Enabled = True: Command4.Enabled = True
  Command5.Enabled = True: Command6.Enabled = True
  Label20.Caption = X_Max - X_Min
  Label21.Caption = 1000 * Y_Gain
  Label22.Caption = X_Min
  Label23.Caption = X_Max

  If X_Gain <= 1 Then Command1.Enabled = False
  If X_Gain >= 31 Then Command2.Enabled = False
  If Y_Gain >= 100 Then Command3.Enabled = False
  If Y_Gain <= 0.01 Then Command4.Enabled = False
  If X_Min <= 0 Then Command5.Enabled = False
  If X_Max >= 1024 Then Command6.Enabled = False

  Draw_Graph_Axis
  Draw_Spectrum

End Sub

Private Sub HScroll1_Change()
  Get_Spectrum_Data (HScroll1.value)
  Update_Graph
End Sub

Private Sub HScroll2_Change()
  Dim Packet_Start_Byte As Long
  Sub_Packet_viewed = HScroll2.value
  Packet_Start_Byte = Science(Spectrum_Start_packet(Spectrum_Displayed, Sub_Packet_viewed)).Start
  Display_Spectrum_Information (Packet_Start_Byte)
End Sub
```

Summary - 1

```
Option Explicit
Dim Start_HK_Packet, Start_Sc_Packet, Start_Inf_Packet As Long

Private Sub Form_Load()
Form_Summary_Show = True
Label13.Caption = Egse_Data_File_Name

Start_HK_Packet = 0
Start_Sc_Packet = 0
Start_Inf_Packet = 0

Enter_Label_Information

If No.Lander_Packets > 0 Then
  Enter_ListBox_Information
End If

End Sub

Private Sub Form_Unload(Cancel As Integer)
Form_Summary_Show = False
End Sub

Private Sub List1_DblClick()
 Current_Housekeeping_Packet = List1.ListIndex + 1
 If Form_Housekeeping_show = False Then
   Housekeeping_form.Show
 Else
   Housekeeping_form.ZOrder (0)
 End If
 Housekeeping_form.Update
End Sub

Private Sub List2_DblClick()
 Current_Science_Packet = List2.ListIndex + 1
 Select Case Science(Current_Science_Packet).Description
   Case " Auxiliary Data   "
      Unload Auxiliary_form
      Auxiliary_form.Show
      Auxiliary_form.ZOrder (0)
   Case " GC spectrum      "
      Spectra_Form.Show
      Spectra_Form.ZOrder (0)
   Case " Isotope spectrum "
      Spectra_Form.Show
      Spectra_Form.ZOrder (0)
 End Select

End Sub

Private Sub List3_DblClick()
 Current_Information_Packet = List3.ListIndex + 1
 If Form_Information_Show = False Then
   Information_Form.Show
 Else
   Information_Form.ZOrder (0)
 End If
 Information_Form.Update
End Sub

Public Sub UpDateSummaryForm()

Enter_Label_Information

Start_HK_Packet = Old_No.Housekeeping_Packets
Start_Sc_Packet = Old_No.Science_Packets
Start_Inf_Packet = Old_No.Information_Packets

Enter_ListBox_Information

'Set list box position to the end of the list boxes
List1.ListIndex = No.Housekeeping_Packets - 1
List2.ListIndex = No.Science_Packets - 1
List3.ListIndex = No.Information_Packets - 1

End Sub
```

```
Public Sub Enter_Label_Information()
'Enter the number of packets onto the summary form
Label17(0).Caption = No.Lander_Packets
Label17(1).Caption = No.Housekeeping_Packets
Label17(2).Caption = No.Science_Packets
Label17(3).Caption = No.Information_Packets
Label17(4).Caption = No.Information_Packets + No.Science_Packets + No.Housekeeping_Packets

'Enter the various types of packet in the frames section
Label12.Caption = No.Concise_HK
Label18.Caption = No.Complete_HK
If No.Housekeeping_Packets > 0 Then
  Label19(0).Caption = Housekeeping(1).Sequence_Count
  Label21(0).Caption = Housekeeping(No.Housekeeping_Packets).Sequence_Count
Else
  Label19(0).Caption = "- N/A -"
  Label21(0).Caption = "- N/A -"
End If

Label22.Caption = No.Aux_Science
Label23.Caption = No.Compact_Science
Label24.Caption = No.Complete_Science
Label26.Caption = No.Complete_Science_Spectra
If No.Science_Packets > 0 Then
  Label19(1).Caption = Science(1).Sequence_Count
  Label21(1).Caption = Science(No.Science_Packets).Sequence_Count
Else
  Label19(1).Caption = "- N/A -"
  Label21(1).Caption = "- N/A -"
End If

Label27.Caption = No.Normal_Progress_Event
Label28.Caption = No.Warning_Event
Label29.Caption = No.Memory_Dump
Label35.Caption = No.Memory_Checksum
Label30.Caption = No.Accept_success
Label31.Caption = No.Accept_Failure
Label32.Caption = No.Unknown_Packet
Label33.Caption = No.Empty_Packet

End Sub

Public Sub Enter_ListBox_Information()
Dim i As Long
Dim dummy, dummy2 As String

'ListBox for Housekeeping Packets
For i = Start_HK_Packet + 1 To No.Housekeeping_Packets
  dummy2 = i
  dummy = dummy2 & Space(8 - Len(dummy2)) & Housekeeping(i).Description
  dummy2 = Housekeeping(i).Sequence_Count
  dummy = dummy & Space(10 - Len(dummy2)) & dummy2
  dummy2 = Housekeeping(i).Time
  dummy = dummy & Space(11 - Len(dummy2)) & dummy2
  dummy2 = Housekeeping(i).Lander_pkt
  dummy = dummy & Space(9 - Len(dummy2)) & dummy2
  List1.AddItem dummy
Next i

'ListBox for Scence Packets
For i = Start_Sc_Packet + 1 To No.Science_Packets
  dummy2 = i
  dummy = dummy2 & Space(8 - Len(dummy2)) & Science(i).Description
  dummy2 = Science(i).Sequence_Count
  dummy = dummy & Space(9 - Len(dummy2)) & dummy2
  dummy2 = Science(i).Time
  dummy = dummy & Space(11 - Len(dummy2)) & dummy2
  dummy2 = Science(i).Lander_pkt
  dummy = dummy & Space(9 - Len(dummy2)) & dummy2
  If Science(i).Description = " Isotope spectrum " Then
    If Science(i).flags And 2 Then dummy = dummy & "   Start"
    If Science(i).flags And 1 Then dummy = dummy & "   End"
  End If
  List2.AddItem dummy
Next i

'Listbox for Information Packets
```

```
Summary - 3

For i = Start_Inf_Packet + 1 To No.Information_Packets
If Information(i).Description = " Unknown " Or Information(i).Description = "  Empty  " Then
  dummy2 = i
  dummy = i & Space(12 - Len(dummy2))
  dummy2 = Information(i).Lander_pkt
  List3.AddItem dummy & Information(i).Description & Space(38 - Len(dummy2)) & dummy2
 Else
  dummy2 = i
  dummy = dummy2 & Space(8 - Len(dummy2)) & Information(i).Description
  dummy2 = Information(i).Sequence_Count
  dummy = dummy & Space(8 - Len(dummy2)) & dummy2
  dummy2 = Information(i).Time
  dummy = dummy & Space(9 - Len(dummy2)) & dummy2
  dummy2 = Information(i).Lander_pkt
  dummy = dummy & Space(9 - Len(dummy2)) & dummy2
  List3.AddItem dummy
End If
Next i
End Sub
```

TestForm - 1

```
Option Explicit
Dim x As Long

Private Sub Command1_Click()
Dim sensorx As Integer
Dim voltagex, Result As Double
Dim unit

sensorx = Text1.Text
voltagex = Text2.Text
Call Get_Sensor_Reading(sensorx, voltagex, Result, unit)
Text4.Text = Result
Text5.Text = unit
End Sub

Private Sub Form_Load()

x = &H8000
Text1.Text = 0

Text2.Text = x
End Sub
```

```
Egse_Code - 1

Option Explicit

Global Ptolemy_Directory As String
Global Byte_Reverse As Boolean          'If set to TRUE then high and low bytes are swopped
Global Egse_Data_File_Name As String
Global Packet_Format As String
Global Test_Type As String
Global Length_of_Egse_File As Long
Global Form_Raw_Data_Show As Boolean    'Status of Raw_Data form
Global Form_Summary_Show As Boolean     'Status of Summary form
Global Form_Housekeeping_show As Boolean 'Status of Housekeeping form
Global Form_Monitor_Show As Boolean     'Status of Monitor form
Global Form_Concise_Data_Show As Boolean 'Status of ConciseData form
Global Form_Information_Show As Boolean  'Status of Information form

Type Packet_Description
  Lander_pkt As Long
  Length As Long
  Start As Long
  Sequence_Count As Long
  flags As Long
  Time As Long
  Description As String
End Type
Public Housekeeping() As Packet_Description
Public Science() As Packet_Description
Public Information() As Packet_Description

Type Number_of_Packets
  Lander_Packets As Long: Housekeeping_Packets As Long
  Science_Packets As Long: Information_Packets As Long
  Complete_HK As Long: Concise_HK As Long
  Aux_Science As Long: Compact_Science As Long
  Complete_Science As Long: Complete_Science_Spectra As Long
  Normal_Progress_Event As Long: Warning_Event As Long
  Memory_Checksum As Long
  Memory_Dump As Long: Accept_success As Long
  Accept_Failure As Long: Unknown_Packet As Long
  Empty_Packet As Long
End Type
Global No As Number_of_Packets
Global Old_No As Number_of_Packets
Global Current_Housekeeping_Packet As Long
Global Current_Science_Packet As Long
Global Current_Information_Packet As Long
Global Test_Start_Packet As Long

Type Sensor_Description
  Mnu As String
  Packet_Position As Long
  ADC_Channel As Integer
  Byte_Value As Long
  Bit_Shift As Long
  Name As String
  voltage As Double
  Voltage_Error As Double
  Reading As Double
  Reading_Error As Double
  unit As String
End Type
Public Sensor() As Sensor_Description
Global Number_of_Sensors As Integer
Global Aux_AD590_EMF As Double      'EMF correction for temperature of cold junction

Public Sensor_Bytes() As Byte  ' Storage for all housekeeing sensor data

Type P_sensors_Description
  Sensor_Number As Integer
  Atmosphere_value As Double
  Zero_value As Double
End Type
Public Pressure_Sensor() As P_sensors_Description
Global Number_Pressure_Sensors As Integer

'Local values for thermocouple calibration
Dim Data_EMF(100) As Double
Dim Data_Temperature(100) As Double
```

```
Egse_Code - 2

Dim Gradient(100), Change_of_gradient(100) As Double
Dim Temp_Gradient(100), Temp_Change_of_gradient(100) As Double
Dim Number_Temperature_Points As Integer

'Local variables for Egse file information
Dim Word_Result As Long
Dim EGSE_File_Number As Integer

Public Sub Main()

'Load thermocouple calibration data and calculate gradients
'and rate of change of gradients for each point
Dim i, File_Number As Integer
Dim Thermocouple_Data_File, dummy As Variant

'Find the Ptolemy Directory
    File_Number = FreeFile
    Open "Ptolemy_Files.txt" For Input As File_Number
    Line Input #File_Number, dummy
    Line Input #File_Number, Ptolemy_Directory
    Close #File_Number

'Opens Source File of resulting temperatures for the value of the calculated emf voltages

    File_Number = FreeFile
    Open Ptolemy_Directory & "\EGSE_info\N-Type.txt" For Input As File_Number

'loads source file line by line until the EOF is reached
    Number_Temperature_Points = 0
    Do While Not EOF(File_Number)
        Number_Temperature_Points = Number_Temperature_Points + 1
        Input #File_Number, Data_EMF(Number_Temperature_Points), Data_Temperature(Number_Temperatur
e_Points)
    Loop
    Close File_Number

'Calculate the gradient at each point.
    Dim X1, X2, X3, Y1, Y2, Y3 As Double
    Dim Grad1, Grad2 As Double
    For i = 2 To Number_Temperature_Points - 1
        X1 = Data_EMF(i - 1): Y1 = Data_Temperature(i - 1)
        X2 = Data_EMF(i): Y2 = Data_Temperature(i)
        X3 = Data_EMF(i + 1): Y3 = Data_Temperature(i + 1)
        Grad1 = (Y2 - Y1) / (X2 - X1)
        Grad2 = (Y3 - Y2) / (X3 - X2)
        Gradient(i) = (Grad1 * (X3 - X2) + Grad2 * (X2 - X1)) / (X3 - X1)
    Next i

'Calculate rate of change of gradient
    For i = 2 To Number_Temperature_Points - 2
        Grad1 = Gradient(i)
        Grad2 = Gradient(i + 1)
        X1 = Data_EMF(i)
        X2 = Data_EMF(i + 1)
        Change_of_gradient(i) = (Grad2 - Grad1) / (X2 - X1)
    Next i

'Calculate the gradient at each point for inverse graph.
    For i = 2 To Number_Temperature_Points - 1
        Temp_Gradient(i) = 1 / Gradient(i)
    Next i

'Calculate rate of change of gradient for inverse graph
    For i = 2 To Number_Temperature_Points - 2
        Grad1 = Temp_Gradient(i)
        Grad2 = Temp_Gradient(i + 1)
        X1 = Data_Temperature(i)
        X2 = Data_Temperature(i + 1)
        Temp_Change_of_gradient(i) = (Grad2 - Grad1) / (X2 - X1)
    Next i

'Load sensor information
    File_Number = FreeFile
    Open Ptolemy_Directory & "\EGSE_info\Sensors.txt" For Input As File_Number

    Input #File_Number, Number_of_Sensors
    ReDim Sensor(Number_of_Sensors)
```

Egse_Code - 3

```
    For i = 1 To Number_of_Sensors
        Input #File_Number, Sensor(i).Mnu
        Input #File_Number, Sensor(i).Packet_Position
        Input #File_Number, Sensor(i).ADC_Channel
        Input #File_Number, Sensor(i).Bit_Shift
        Input #File_Number, Sensor(i).Name
    Next i
    Close File_Number

    Aux_AD590_EMF = 0

'Load pressure sensor calibration data
    File_Number = FreeFile
    Open Ptolemy_Directory & "\EGSE_info\Pressure Sensors.txt" For Input As File_Number
    Input #File_Number, Number_Pressure_Sensors
    ReDim Pressure_Sensor(Number_Pressure_Sensors)
    For i = 1 To Number_Pressure_Sensors
        Input #File_Number, Pressure_Sensor(i).Sensor_Number
        Input #File_Number, Pressure_Sensor(i).Atmosphere_value
        Input #File_Number, Pressure_Sensor(i).Zero_value
    Next i
    Close #File_Number

'Show the Main document interface
    EGSE_MDI.Show
    Byte_Reverse = True
    EGSE_MDI.mnuDisplayByteReverse.Checked = True

End Sub

Public Sub Determine_Number_of_Packets(Result)
'Determine the size of the file
'This uses the fact that:
'   a lander packet is 282 bytes long,
'   a Ptolemy packet is 256 bytes long,
'   and a RAL packet is 136 entries long.
'The number of Lander packets is returned in result

Dim File_Length As Long
Dim result_value As Double
Dim File_Number As Integer

File_Number = FreeFile
Select Case Packet_Format
  Case "CSS", "QM"
    Open Egse_Data_File_Name For Binary As #File_Number
    File_Length = LOF(File_Number)
    Close #File_Number
    result_value = File_Length / 282

  Case "GRM", "FM"
    Open Egse_Data_File_Name For Binary As #File_Number
    File_Length = LOF(File_Number)
    Close #File_Number
    result_value = File_Length / 256

  Case "RAL"
    Open Egse_Data_File_Name For Input As #File_Number
    Dim Number_of_Entries, y
    Number_of_Entries = 0
    Do While Not EOF(File_Number)
      Input #File_Number, y
      Number_of_Entries = Number_of_Entries + 1
    Loop
    result_value = Number_of_Entries / 136 - 1
    Close #File_Number

End Select

If result_value <> Int(result_value) Then
  result_value = Int(result_value + 0.999)
End If
Result = result_value
End Sub

Static Function Log10(x)
     Log10 = Log(x) / Log(10#)
```

```
Egse_Code - 4

End Function

Public Sub Convert_EMF_to_Temperature(EMF, Temperature)
'EMF is in micro-Volts
'Temperature is in degrees Centigrade
Dim First_Point, Second_Point As Integer
Dim Delta_EMF As Double

If EMF < Data_EMF(2) Then
  Temperature = -280
  Exit Sub
End If

If EMF > Data_EMF(Number_Temperature_Points - 1) Then
  Temperature = 2000
  Exit Sub
End If

Second_Point = 1
Do
  Second_Point = Second_Point + 1
Loop Until Data_EMF(Second_Point) > EMF
First_Point = Second_Point - 1

Delta_EMF = EMF - Data_EMF(First_Point)
Temperature = Data_Temperature(First_Point) + Delta_EMF * Gradient(First_Point) + Delta_EMF ^ 2 * C
hange_of_gradient(First_Point) / 2

End Sub
Public Sub Convert_Temperature_to_EMF(Temperature, EMF)
'EMF is in micro-Volts
'Temperature is in degrees Centigrade
Dim First_Point, Second_Point As Integer
Dim Delta_Temperature As Double

If Temperature < Data_Temperature(2) Then
  EMF = -6600
  Exit Sub
End If

If Temperature > Data_Temperature(Number_Temperature_Points - 1) Then
  EMF = 50000
  Exit Sub
End If

Second_Point = 1
Do
  Second_Point = Second_Point + 1
Loop Until Data_Temperature(Second_Point) > Temperature
First_Point = Second_Point - 1

Delta_Temperature = Temperature - Data_Temperature(First_Point)
EMF = Data_EMF(First_Point) + Delta_Temperature * Temp_Gradient(First_Point) + Delta_Temperature ^
2 * (Temp_Change_of_gradient(First_Point) / 2)

End Sub

Sub Get_Sensor_Reading(x, y, z, u)
'This routine calculates the reading for sensor x, from its
'voltage y and returns the value in z.  Unit is stored as u
Dim Result As Double
  z = 0

  Select Case x
    Case 1 To 21, 27    'N-type thermocouples
      Call Convert_EMF_to_Temperature(y * 10000, Result)
      z = Result
      u = "oC"
    Case 22 To 25       'Pressure sensors - result in bar
      y = y * 10 - Pressure_Sensor(x - 21).Zero_value
      z = y / (Pressure_Sensor(x - 21).Atmosphere_value - Pressure_Sensor(x - 21).Zero_value)
      u = "bar"
    Case 26             'Pressure sensors - result in mbar
      y = y * 10 - Pressure_Sensor(x - 21).Zero_value
      z = 1000 * y / (Pressure_Sensor(x - 21).Atmosphere_value - Pressure_Sensor(x - 21).Zero_value
)
      u = "mbar"
```

```
Egse_Code - 5

    Case 28             'AD590
      z = y * 100 - 273
      u = "oC"
    Case 29             'Docking station
      z = y * 5.2
      u = "mm"
    Case 30             'Nano-tip current
      z = y * 100
      u = "uA"
    Case 31             'Detector voltage
      z = y
      u = "kV"
    Case 32             '5V voltage monitor
      z = y * 2
      u = "V"
    Case 33             '28V voltage monitor
      z = y * 10
      u = "V"
    Case 34             '5V current monitor
      z = y * 1000
      u = "mA"
    Case 35             '28V current monitor
      z = y * 1000
      u = "mA"
    Case 36             'RF calibration
      z = y * 100
      u = "V"
  End Select

End Sub

Public Sub Load_Sensor_Data()

ReDim Preserve Sensor_Bytes(Number_of_Sensors, No.Housekeeping_Packets)

Dim File_Number As Integer
Dim Byte_Value As Byte
Dim i, j As Integer
Dim x As Double
Dim y As Boolean

Select Case Packet_Format
  Case "CSS", "QM", "GRM", "FM"

    File_Number = FreeFile
    Open Egse_Data_File_Name For Binary As #File_Number

    For j = Old_No.Housekeeping_Packets + 1 To No.Housekeeping_Packets
      If Not Byte_Reverse Then
        For i = 1 To Number_of_Sensors
          Get #File_Number, Housekeeping(j).Start + Sensor(i).Packet_Position, Byte_Value
          Sensor_Bytes(i, j) = Byte_Value
        Next i
      Else
        For i = 1 To Number_of_Sensors
          x = Sensor(i).Packet_Position
          x = Int(x / 2) * 4 + 1 - x 'Believe it or not, this swops over the packet positions
          Get #File_Number, Housekeeping(j).Start + x, Byte_Value
          Sensor_Bytes(i, j) = Byte_Value
        Next i
      End If
    Next j
    Close #File_Number

  Case "RAL"

    Dim Result(136) As Variant
    Dim dummy As Double
    Dim Counter As Long

    File_Number = FreeFile
    Open Egse_Data_File_Name For Input As #File_Number

'The first 136 items contain header information
    For j = 1 To 136
      Input #File_Number, Result(j)
    Next j
```

Egse_Code - 6

```
    Counter = 0
    For i = 1 To No.Lander_Packets - 1
      For j = 1 To 136
        Input #File_Number, Result(j)
      Next j

      If Result(5) = &H734 Then
'Convert all values to positive readings
        For j = 1 To 136
          If Result(j) < 0 Then Result(j) = Result(j) + 65536
        Next j

        Counter = Counter + 1
'Input all the sensor readings
        For j = 1 To Number_of_Sensors
          x = Sensor(j).Packet_Position
          x = x / 2
          If x = Int(x) Then y = True Else y = False
          x = Int(x)
          If y Then
            dummy = Int(Result(x + 9) / 256)
            Sensor_Bytes(j, Counter) = dummy
          Else
            dummy = Int(Result(x + 9) / 256)
            dummy = Result(x + 9) - 256 * dummy
            Sensor_Bytes(j, Counter) = dummy
          End If
        Next j
      End If
    Next i

    Close #File_Number

End Select

End Sub

Public Sub Reload_Active_Forms()
Dim Form_x, Form_y As Long

If Form_Raw_Data_Show = True Then
  Form_x = RawData.Left
  Form_y = RawData.Top
  Unload RawData
  RawData.Show
  RawData.Left = Form_x
  RawData.Top = Form_y
End If

If Form_Summary_Show = True Then
  Form_x = Summary.Left
  Form_y = Summary.Top
  Unload Summary
  Summary.Show
  Summary.Left = Form_x
  Summary.Top = Form_y
End If

If Form_Concise_Data_Show = True Then
  Form_x = Concise_Form.Left
  Form_y = Concise_Form.Top
  Unload Concise_Form
  Concise_Form.Show
  Concise_Form.Left = Form_x
  Concise_Form.Top = Form_y
End If

If Form_Housekeeping_show = True Then
  Form_x = Housekeeping_form.Left
  Form_y = Housekeeping_form.Top
  Unload Housekeeping_form
  Housekeeping_form.Show
  Housekeeping_form.Left = Form_x
  Housekeeping_form.Top = Form_y
End If
```

```
Egse_Code - 7

End Sub

Public Sub Update_Active_Forms()

If Form_Raw_Data_Show = True Then
  RawData.UpdateRawData
End If

If Form_Summary_Show = True Then
  Summary.UpDateSummaryForm
End If

If Form_Concise_Data_Show = True Then
  Concise_Form.Update_Concise_Form
End If

If Form_Housekeeping_show = True Then
  Housekeeping_form.UpdateRealtime
End If

End Sub
Public Sub Load_282_byte_Packet_Information()
Dim i As Long
Dim Start_Position As Long
Dim Packet_ID, Sequence_Control, Packet_Length, flags As Long
Dim Lander_Time_seconds, Lander_time_fractions As Long
Dim Lander_Packet_Position As Long
Dim Packet_Type As String

ReDim Preserve Housekeeping(No.Lander_Packets * 4)
ReDim Preserve Science(No.Lander_Packets)
ReDim Preserve Information(No.Lander_Packets * 8)

EGSE_File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #EGSE_File_Number

For i = Old_No.Lander_Packets + 1 To No.Lander_Packets
  Lander_Packet_Position = 23
  Do
    Start_Position = 282 * (i - 1) + Lander_Packet_Position
    Get_Word (Start_Position)
    Packet_ID = Word_Result
    Get_Word (Start_Position + 2)
    Sequence_Control = Word_Result
    flags = Sequence_Control And &HC000
    Sequence_Control = Sequence_Control - flags
    flags = flags / 16384
    Get_Word (Start_Position + 4)
    Packet_Length = Word_Result + 7
    Lander_Time_seconds = 0
    Get_Word (Start_Position + 6)
    If Word_Result > 32767 Then Word_Result = -1 'A fix if lander time > 68 years
    Lander_Time_seconds = Word_Result * 256 * 256
    Get_Word (Start_Position + 8)
    Lander_Time_seconds = Lander_Time_seconds + Word_Result
    Get_Word (Start_Position + 10)
    Lander_time_fractions = Word_Result

    Lander_Packet_Position = Lander_Packet_Position + Packet_Length

    Select Case Packet_ID
      Case &HF34    'Housekeeping packets
        No.Housekeeping_Packets = No.Housekeeping_Packets + 1
        Get_Word (Start_Position + 16)
        Packet_Type = " Unknown  "
        If Word_Result = 1 Then Packet_Type = " Concise  ": No.Concise_HK = No.Concise_HK + 1
        If Word_Result = 2 Then Packet_Type = " Complete ": No.Complete_HK = No.Complete_HK + 1
        Housekeeping(No.Housekeeping_Packets).Lander_pkt = i
        Housekeeping(No.Housekeeping_Packets).Length = Packet_Length
        Housekeeping(No.Housekeeping_Packets).Start = Start_Position
        Housekeeping(No.Housekeeping_Packets).Sequence_Count = Sequence_Control
        Housekeeping(No.Housekeeping_Packets).Time = Lander_Time_seconds
        Housekeeping(No.Housekeeping_Packets).Description = Packet_Type

      Case &HF37    'Progress events (normal and warning)
        No.Information_Packets = No.Information_Packets + 1
        Get_Word (Start_Position + 14)
```

```
Egse_Code - 8

        Select Case Word_Result / 256
          Case 1
            Packet_Type = " Normal Progress Event   "
            No.Normal_Progress_Event = No.Normal_Progress_Event + 1
          Case 2
            Packet_Type = " Warning Anomalous Event "
            No.Warning_Event = No.Warning_Event + 1
          Case 10
            Packet_Type = " NPE - Memory Checksum   "
            No.Normal_Progress_Event = No.Normal_Progress_Event + 1
          Case Else
            Packet_Type = " Unknown - progress event"
            No.Unknown_Packet = No.Unknown_Packet + 1
        End Select
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &HF31    'Acceptance or failure TC or
        No.Information_Packets = No.Information_Packets + 1
        Get_Word (Start_Position + 14)
        Select Case Word_Result / 256
          Case 1
            Packet_Type = " Acceptance Success      "
            No.Accept_success = No.Accept_success + 1
          Case 2
            Packet_Type = " Acceptance Failure      "
            No.Accept_Failure = No.Accept_Failure + 1
          Case 10
            Packet_Type = " Memory Checksum         "
            No.Memory_Checksum = No.Memory_Checksum + 1
          Case Else
            Packet_Type = " Unknown - acceptance    "
            No.Unknown_Packet = No.Unknown_Packet + 1
        End Select
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &HF39    'Ptolemy memory dump
        No.Information_Packets = No.Information_Packets + 1
        Packet_Type = " Memory Dump             "
        No.Memory_Dump = No.Memory_Dump + 1
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &HF3C    'Ptolemy Science Data
        No.Science_Packets = No.Science_Packets + 1
        Get_Word (Start_Position + 16)
        Packet_Type = " Unknown          "
        If Word_Result = 1 Then Packet_Type = " Auxiliary Data   ": No.Aux_Science = No.Aux_Science
 + 1
        If Word_Result = 2 Then Packet_Type = " GC spectrum      ": No.Compact_Science = No.Compact
_Science + 1
        If Word_Result = 3 Then Packet_Type = " Isotope spectrum ": No.Complete_Science = No.Comple
te_Science + 1
        Science(No.Science_Packets).Lander_pkt = i
        Science(No.Science_Packets).Length = Packet_Length
        Science(No.Science_Packets).Start = Start_Position
        Science(No.Science_Packets).Sequence_Count = Sequence_Control
        Science(No.Science_Packets).flags = flags
        If Packet_Type = " Isotope spectrum " Then
          If flags And 2 Then No.Complete_Science_Spectra = No.Complete_Science_Spectra + 1
        End If
        Science(No.Science_Packets).Time = Lander_Time_seconds
        Science(No.Science_Packets).Description = Packet_Type
```

```
Egse_Code - 9

      Case &H0
'If the packet_ID is empty then move to end of lander packet
        No.Information_Packets = No.Information_Packets + 1
        Packet_Type = "  Empty  "
        No.Empty_Packet = No.Empty_Packet + 1
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = -1
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = -1
        Information(No.Information_Packets).Time = -1
        Information(No.Information_Packets).Description = Packet_Type

        Lander_Packet_Position = 282

      Case Else
'If the packet_ID is unknown then move to end of lander packet
        No.Unknown_Packet = No.Unknown_Packet + 1
        Packet_Type = " Unknown "
        No.Information_Packets = No.Information_Packets + 1
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = -1
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = -1
        Information(No.Information_Packets).Time = -1
        Information(No.Information_Packets).Description = Packet_Type

        Lander_Packet_Position = 282
    End Select

  Loop Until Lander_Packet_Position > 250
Next i

Close #EGSE_File_Number

ReDim Preserve Housekeeping(No.Housekeeping_Packets)
ReDim Preserve Science(No.Science_Packets)
ReDim Preserve Information(No.Information_Packets)

End Sub

Public Sub Load_280_byte_Packet_Information()
Dim i As Long
Dim Start_Position As Long
Dim Packet_ID, Sequence_Control, Packet_Length, flags As Long
Dim Lander_Time_seconds, Lander_time_fractions As Long
Dim Lander_Packet_Position As Long
Dim Packet_Type As String

ReDim Preserve Housekeeping(No.Lander_Packets * 4)
ReDim Preserve Science(No.Lander_Packets)
ReDim Preserve Information(No.Lander_Packets * 8)

EGSE_File_Number = FreeFile
Open Egse_Data_File_Name For Binary As #EGSE_File_Number

For i = Old_No.Lander_Packets + 1 To No.Lander_Packets
  Lander_Packet_Position = 23
  Do
    Start_Position = 280 * (i - 1) + Lander_Packet_Position
    Get_Word (Start_Position)
    Packet_ID = Word_Result
    Get_Word (Start_Position + 2)
    Sequence_Control = Word_Result
    flags = Sequence_Control And &HC000
    Sequence_Control = Sequence_Control - flags
    flags = flags / 16384
    Get_Word (Start_Position + 4)
    Packet_Length = Word_Result + 7
    Lander_Time_seconds = 0
    Get_Word (Start_Position + 6)
    If Word_Result > 32767 Then Word_Result = -1 'A fix if lander time > 68 years
    Lander_Time_seconds = Word_Result * 256 * 256
    Get_Word (Start_Position + 8)
    Lander_Time_seconds = Lander_Time_seconds + Word_Result
    Get_Word (Start_Position + 10)
    Lander_time_fractions = Word_Result
```

```
    Lander_Packet_Position = Lander_Packet_Position + Packet_Length

    Select Case Packet_ID
      Case &HF34    'Housekeeping packets
        No.Housekeeping_Packets = No.Housekeeping_Packets + 1
        Get_Word (Start_Position + 16)
        Packet_Type = " Unknown  "
        If Word_Result = 1 Then Packet_Type = " Concise  ": No.Concise_HK = No.Concise_HK + 1
        If Word_Result = 2 Then Packet_Type = " Complete ": No.Complete_HK = No.Complete_HK + 1
        Housekeeping(No.Housekeeping_Packets).Lander_pkt = i
        Housekeeping(No.Housekeeping_Packets).Length = Packet_Length
        Housekeeping(No.Housekeeping_Packets).Start = Start_Position
        Housekeeping(No.Housekeeping_Packets).Sequence_Count = Sequence_Control
        Housekeeping(No.Housekeeping_Packets).Time = Lander_Time_seconds
        Housekeeping(No.Housekeeping_Packets).Description = Packet_Type

      Case &HF37    'Progress events (normal and warning)
        No.Information_Packets = No.Information_Packets + 1
        Get_Word (Start_Position + 14)
        Select Case Word_Result / 256
          Case 1
            Packet_Type = " Normal Progress Event   "
            No.Normal_Progress_Event = No.Normal_Progress_Event + 1
          Case 2
            Packet_Type = " Warning Anomalous Event "
            No.Warning_Event = No.Warning_Event + 1
          Case 10
            Packet_Type = " NPE - Memory Checksum   "
            No.Normal_Progress_Event = No.Normal_Progress_Event + 1
          Case Else
            Packet_Type = " Unknown - progress event"
            No.Unknown_Packet = No.Unknown_Packet + 1
        End Select
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &HF31    'Acceptance or failure TC or
        No.Information_Packets = No.Information_Packets + 1
        Get_Word (Start_Position + 14)
        Select Case Word_Result / 256
          Case 1
            Packet_Type = " Acceptance Success      "
            No.Accept_success = No.Accept_success + 1
          Case 2
            Packet_Type = " Acceptance Failure      "
            No.Accept_Failure = No.Accept_Failure + 1
          Case 10
            Packet_Type = " Memory Checksum         "
            No.Memory_Checksum = No.Memory_Checksum + 1
          Case Else
            Packet_Type = " Unknown - acceptance    "
            No.Unknown_Packet = No.Unknown_Packet + 1
        End Select
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &HF39    'Ptolemy memory dump
        No.Information_Packets = No.Information_Packets + 1
        Packet_Type = " Memory Dump             "
        No.Memory_Dump = No.Memory_Dump + 1
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &HF3C    'Ptolemy Science Data
        No.Science_Packets = No.Science_Packets + 1
```

```
Egse_Code - 11

        Get_Word (Start_Position + 16)
        Packet_Type = " Unknown          "
        If Word_Result = 1 Then Packet_Type = " Auxiliary Data   ": No.Aux_Science = No.Aux_Science
 + 1
        If Word_Result = 2 Then Packet_Type = " GC spectrum      ": No.Compact_Science = No.Compact
_Science + 1
        If Word_Result = 3 Then Packet_Type = " Isotope spectrum ": No.Complete_Science = No.Comple
te_Science + 1
        Science(No.Science_Packets).Lander_pkt = i
        Science(No.Science_Packets).Length = Packet_Length
        Science(No.Science_Packets).Start = Start_Position
        Science(No.Science_Packets).Sequence_Count = Sequence_Control
        Science(No.Science_Packets).flags = flags
        If Packet_Type = " Isotope spectrum " Then
          If flags And 1 Then No.Complete_Science_Spectra = No.Complete_Science_Spectra + 1
        End If
        Science(No.Science_Packets).Time = Lander_Time_seconds
        Science(No.Science_Packets).Description = Packet_Type

      Case &H0
'If the packet_ID is empty then move to end of lander packet
        No.Information_Packets = No.Information_Packets + 1
        Packet_Type = "  Empty  "
        No.Empty_Packet = No.Empty_Packet + 1
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = -1
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = -1
        Information(No.Information_Packets).Time = -1
        Information(No.Information_Packets).Description = Packet_Type

        Lander_Packet_Position = 282

      Case Else
'If the packet_ID is unknown then move to end of lander packet
        No.Unknown_Packet = No.Unknown_Packet + 1
        Packet_Type = " Unknown "
        No.Information_Packets = No.Information_Packets + 1
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = -1
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = -1
        Information(No.Information_Packets).Time = -1
        Information(No.Information_Packets).Description = Packet_Type

        Lander_Packet_Position = 282
    End Select

  Loop Until Lander_Packet_Position > 250
Next i

Close #EGSE_File_Number

ReDim Preserve Housekeeping(No.Housekeeping_Packets)
ReDim Preserve Science(No.Science_Packets)
ReDim Preserve Information(No.Information_Packets)

End Sub

Public Sub Load_RAL_Packet_Information()
Dim i, j As Integer
Dim x(136)
Dim Start_Position As Long
Dim Packet_ID, Sequence_Control, Packet_Length, flags As Long
Dim Lander_Time_seconds, Lander_time_fractions As Long
Dim Lander_Packet_Position As Long
Dim Packet_Type As String
Dim dummy, dummy2 As String

ReDim Preserve Housekeeping(No.Lander_Packets)
ReDim Preserve Science(No.Lander_Packets)
ReDim Preserve Information(No.Lander_Packets)

EGSE_File_Number = FreeFile
Open Egse_Data_File_Name For Input As #EGSE_File_Number

'  this first loop is to get past the first RAL packet which contains
```

```
Egse_Code - 12

'  header information
For j = 1 To 136
  Input #EGSE_File_Number, x(j)
Next j

For i = 2 To No.Lander_Packets
  For j = 1 To 136
    Input #EGSE_File_Number, x(j)
  Next j
  Packet_ID = x(5)
  flags = x(6)
  Sequence_Control = x(7)
  Packet_Length = x(8)
  Lander_Time_seconds = x(9)
  Lander_time_fractions = x(10)

    Select Case Packet_ID
      Case &H734    'Housekeeping packets
        No.Housekeeping_Packets = No.Housekeeping_Packets + 1
        Packet_Type = " Unknown  "
        If x(17) = 1 Then Packet_Type = " Concise  ": No.Concise_HK = No.Concise_HK + 1
        If x(17) = 2 Then Packet_Type = " Complete ": No.Complete_HK = No.Complete_HK + 1
        Housekeeping(No.Housekeeping_Packets).Lander_pkt = i
        Housekeeping(No.Housekeeping_Packets).Length = Packet_Length
        Housekeeping(No.Housekeeping_Packets).Start = Start_Position
        Housekeeping(No.Housekeeping_Packets).Sequence_Count = Sequence_Control
        Housekeeping(No.Housekeeping_Packets).Time = Lander_Time_seconds
        Housekeeping(No.Housekeeping_Packets).Description = Packet_Type

      Case &H737    'Progress events (normal and warning)
        No.Information_Packets = No.Information_Packets + 1
        Select Case x(15)
          Case 1
            Packet_Type = " Normal Progress Event   "
            No.Normal_Progress_Event = No.Normal_Progress_Event + 1
          Case 2
            Packet_Type = " Warning Anomalous Event "
            No.Warning_Event = No.Warning_Event + 1
          Case Else
            Packet_Type = " Unknown - progress event"
            No.Unknown_Packet = No.Unknown_Packet + 1
        End Select
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &H731    'Acceptance or failure TC or
        No.Information_Packets = No.Information_Packets + 1
        Select Case x(15)
          Case 1
            Packet_Type = " Acceptance Success      "
            No.Accept_success = No.Accept_success + 1
          Case 2
            Packet_Type = " Acceptance Failure      "
            No.Accept_Failure = No.Accept_Failure + 1
          Case Else
            Packet_Type = " Unknown - acceptance    "
            No.Unknown_Packet = No.Unknown_Packet + 1
        End Select
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &H739    'Ptolemy memory dump
        No.Information_Packets = No.Information_Packets + 1
        Packet_Type = " Memory Dump             "
        No.Memory_Dump = No.Memory_Dump + 1
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = Packet_Length
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = Sequence_Control
```

```
Egse_Code - 13

        Information(No.Information_Packets).Time = Lander_Time_seconds
        Information(No.Information_Packets).Description = Packet_Type

      Case &H73C    'Ptolemy Science Data
        No.Science_Packets = No.Science_Packets + 1
        Packet_Type = " Unknown          "
        If x(17) = 1 Then Packet_Type = " Auxiliary Data   ": No.Aux_Science = No.Aux_Science + 1
        If x(17) = 2 Then Packet_Type = " GC spectrum      ": No.Compact_Science = No.Compact_Scien
ce + 1
        If x(17) = 3 Then Packet_Type = " Isotope spectrum ": No.Complete_Science = No.Complete_Sci
ence + 1
        Science(No.Science_Packets).Lander_pkt = i
        Science(No.Science_Packets).Length = Packet_Length
        Science(No.Science_Packets).Start = Start_Position
        Science(No.Science_Packets).Sequence_Count = Sequence_Control
        Science(No.Science_Packets).flags = flags
        If Packet_Type = " Isotope spectrum " Then
          If flags And 1 Then No.Complete_Science_Spectra = No.Complete_Science_Spectra + 1
        End If
        Science(No.Science_Packets).Time = Lander_Time_seconds
        Science(No.Science_Packets).Description = Packet_Type

      Case &H0
'If the packet_ID is empty then move to end of lander packet
        No.Information_Packets = No.Information_Packets + 1
        No.Empty_Packet = No.Empty_Packet + 1
        Packet_Type = "  Empty  "
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = -1
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = -1
        Information(No.Information_Packets).Time = -1
        Information(No.Information_Packets).Description = Packet_Type

      Case Else
'If the packet_ID is unknown then move to end of lander packet
        No.Information_Packets = No.Information_Packets + 1
        No.Unknown_Packet = No.Unknown_Packet + 1
        Packet_Type = " Unknown "
        Information(No.Information_Packets).Lander_pkt = i
        Information(No.Information_Packets).Length = -1
        Information(No.Information_Packets).Start = Start_Position
        Information(No.Information_Packets).Sequence_Count = -1
        Information(No.Information_Packets).Time = -1
        Information(No.Information_Packets).Description = Packet_Type

    End Select

Next i
Close #EGSE_File_Number

ReDim Preserve Housekeeping(No.Housekeeping_Packets)
ReDim Preserve Science(No.Science_Packets)
ReDim Preserve Information(No.Information_Packets)

End Sub


Public Sub Get_Word(Byte_Position)
'This routine returns the word at Byte_Position in
'Word_Result
Dim First_Byte As Byte
Dim Second_Byte As Byte
Dim High_Byte, Low_Byte As Long

If Byte_Position < 0 Then
  Word_Result = 0
  Exit Sub
End If

Get #EGSE_File_Number, Byte_Position, First_Byte
Get #EGSE_File_Number, Byte_Position + 1, Second_Byte
If Byte_Reverse Then
  High_Byte = Second_Byte
  Low_Byte = First_Byte
Else
  High_Byte = First_Byte
```

```
Egse_Code - 14

  Low_Byte = Second_Byte
End If
Word_Result = High_Byte * 256 + Low_Byte

End Sub

Public Sub Initialise_Packet_Numbers()
Current_Housekeeping_Packet = 1
Current_Science_Packet = 0
Current_Information_Packet = 0
No.Lander_Packets = 0: No.Housekeeping_Packets = 0
No.Science_Packets = 0: No.Information_Packets = 0
No.Complete_HK = 0: No.Concise_HK = 0
No.Aux_Science = 0: No.Compact_Science = 0
No.Complete_Science = 0: No.Complete_Science_Spectra = 0
No.Normal_Progress_Event = 0: No.Warning_Event = 0
No.Memory_Dump = 0: No.Accept_success = 0
No.Accept_Failure = 0: No.Unknown_Packet = 0
No.Empty_Packet = 0

Old_No.Lander_Packets = 0: Old_No.Housekeeping_Packets = 0
Old_No.Science_Packets = 0: Old_No.Information_Packets = 0
Old_No.Complete_HK = 0: Old_No.Concise_HK = 0
Old_No.Aux_Science = 0: Old_No.Compact_Science = 0
Old_No.Complete_Science = 0: Old_No.Complete_Science_Spectra = 0
Old_No.Normal_Progress_Event = 0: Old_No.Warning_Event = 0
Old_No.Memory_Dump = 0: Old_No.Accept_success = 0
Old_No.Accept_Failure = 0: Old_No.Unknown_Packet = 0
Old_No.Empty_Packet = 0

End Sub

Public Sub Update_Old_Packet_Numbers()
'This routine sets the old packet numbers to the current set
'of packet numbers
Old_No.Lander_Packets = No.Lander_Packets:
Old_No.Housekeeping_Packets = No.Housekeeping_Packets
Old_No.Science_Packets = No.Science_Packets
Old_No.Information_Packets = No.Information_Packets
Old_No.Complete_HK = No.Complete_HK
Old_No.Concise_HK = No.Concise_HK
Old_No.Aux_Science = No.Aux_Science
Old_No.Compact_Science = No.Compact_Science
Old_No.Complete_Science = No.Complete_Science
Old_No.Complete_Science_Spectra = No.Complete_Science_Spectra
Old_No.Normal_Progress_Event = No.Normal_Progress_Event
Old_No.Warning_Event = No.Warning_Event
Old_No.Memory_Dump = No.Memory_Dump
Old_No.Accept_success = No.Accept_success
Old_No.Accept_Failure = No.Accept_Failure
Old_No.Unknown_Packet = No.Unknown_Packet
Old_No.Empty_Packet = No.Empty_Packet

End Sub

Public Function Convert_RF_Word_to_Frequency(Frequency_word)
Dim bit As Double

bit = (572.6 - 535.8) / 512
Convert_RF_Word_to_Frequency = 388.6 + Frequency_word * bit
End Function
```

Egse_History - 1

```
'  27 - March - 2001   Version 1.0
'  Start writting code whilst waiting for FM testing.
'
'  27 - April - 2001  Version 2.0
'  Writting code to display data collection in real time
'  Main software control is from the MDI timer1
'
'  9 - May - 2001  Version 2.1
'  Add code to display the information packets
'  Add code to calibrate pressure sensors
'
'  27 - May - 2001  Version 2.2
'  add code to display the short (air) test
'
'  13 - June - 2001  Ptolemy EGSE Version 1.0
'  Program re-named "Pto EGSE" now in Ptolemy EGSE folder
'  Add routines to export HK and auxiliary data to Excel
'
'  17 - July - 2001  Ptolemy EGSE Version 1.1
'  Program form to display science spectra.  Monitor
'  form removed, this was too complicated and not
'  really required.

'  9 - Nov - 2001
'  The "Concise" form now reads Ptolemy packets
'  Corrected error reporting back-up RAM information
'  Removed bug that prevented user exporting ranges of
'  files to excel.
'  Added form to export docking station calibration data
'  Corrected error so that several short tests could be viewed

'  20 - Feb - 2002
'  A quick fix so that the EGSE can work from any file and folder
'
'  New version 1.5
'  6 - July - 204
'  Change load File menu for CSS, FM, QM and GRM type packets
'  Add new test for EAFT (Limited Cruise phase mode).
```