```
'Silicon Chip EPROM Programmer and Reader V1.3 -- written by Jim Rowe
'Last revised 21/02/04

Option Explicit

Dim intBaseAddress As Integer:              'printer port base address
Dim intReadAddress As Integer:              'printer port status address
Dim intModeAddress As Integer:              'printer port control address
Dim lngEPROMAddress As Long:                'current EPROM address
Dim lngStartAddress As Long:                'EPROM starting addr
Dim lngEndAddress As Long:                  'EPROM top addr
Dim lngStopAddress As Long:                 'stop addr for task
Dim bytRomData(63, 4095) As Byte:           'data storage array
Dim bytDeviceConfig As Byte:                'device configuration byte
Dim bytMode As Byte:                        'programmer mode byte
Dim bytPulseWidth As Byte:                  'programming pulse width code
Dim bytPulseMult As Byte:                   'programming pulse multiplier
Dim bytReadData As Byte:                    'data byte from reading device
Dim bytWriteData As Byte:                   'data byte for writing to dev
Dim strConfigFile As String:                'device configuration filename
Dim strDataFile As String:                  'data filename
Dim strCrLf As String:                      'Cr + Lf string
Dim strPrompt As String:                    'prompt string for dialogs
Dim intResp As Integer:                     'return value from MsgBox dialogs
Dim strResp As String:                      'return string from InputBox dialogs
Dim lngHexVal As Long:                      'long variable from hex input
Dim blnHexFlag As Boolean:                  'flag byte for hex input check
Dim blnVerifyFlag As Boolean:               'flag for verify after program
Dim blnRangeOK As Boolean:                  'flag for range input validity
Dim blnDevConfigFlag As Boolean:            'flag for device config opened
Dim blnPreProgOK:                           'flag for pre program checking
Dim intProgress As Integer:                 'value variable for progress bar

'Declarations for calling the IO.DLL procedures for port communication
Private Declare Sub PortOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Byte)
Private Declare Function PortIn Lib "IO.DLL" (ByVal Port As Integer) As Byte
Private Declare Function IsDriverInstalled Lib "IO.DLL" () As Boolean

Private Sub Form_Load()
    Dim strFetchString As String
    intBaseAddress = &H378: 'set default values
    lngEPROMAddress = 0
```

```
    lngStartAddress = 0
    lngEndAddress = 4095
    bytDeviceConfig = 0
    bytMode = CByte(0)
    bytPulseWidth = 205
    bytPulseMult = 1
    blnVerifyFlag = False
    blnDevConfigFlag = False
    prgProgBar1.Visible = False: 'hide and initialise progress bar
    prgProgBar1.Max = 100
    strDataFile = "": ' null data file name until we open one...
    strCrLf = Chr(13) + Chr(10)
    lblCurrDevAddressDisp.Caption = "00000 hex"
    On Error GoTo ErrorHandler
    Open "C:\Progra~1\EPROMProg\EpromPrg.cfg" For Input As #3
    Input #3, strFetchString: 'skip past header
    Input #3, strFetchString: '& get base addr string
    intBaseAddress = CInt(strFetchString)
    strFetchString = Hex(intBaseAddress) + " hex": 'and display
    strFetchString = "Programmer Base Address: " + strFetchString
    lblBaseAddressDisp.Caption = strFetchString
    intReadAddress = intBaseAddress + 1: 'set read & mode addresses
    intModeAddress = intBaseAddress + 2
    Input #3, strConfigFile: 'then get last device cfg filename
    Call OpenDevConfig
    Close #3: 'close config file
    PortOut intBaseAddress, CByte(0): '& initialise programmer
    PortOut intModeAddress, bytMode
    Exit Sub

ErrorHandler:
    Exit Sub
End Sub

Private Sub mnuAbout_Click()
    strPrompt = "Silicon Chip Windows-based EPROM Programmer" + strCrLf
    strPrompt = strPrompt + "Version 1.3, written by Jim Rowe" + strCrLf
    strPrompt = strPrompt + "(Last modified & debugged 21/02/2004)"
    intResp = MsgBox(strPrompt, vbOKOnly, "About this program")
End Sub

Private Sub mnuCheckErasure_Click()
    Dim strReturn As String
```

```
    Dim blnEraseFlag As Boolean
    If blnDevConfigFlag = False Then
        strPrompt = "Device configuration not set!": 'no config set, so warn
        intResp = MsgBox(strPrompt, vbOKOnly, "Configuration not set")
        Exit Sub: 'and bail out
    End If
    prgProgBar1.Value = 0: 'initialise progress bar
    prgProgBar1.Visible = True: 'and make it visible
    lngEPROMAddress = lngStartAddress: 'now set to begin at starting address
    bytMode = CByte(0): 'and set mode for reading, before we get going
    PortOut intModeAddress, bytMode
    blnEraseFlag = True: 'set flag for good result unless we find it's not so
    Do Until lngEPROMAddress > lngEndAddress
        Call DownloadAddress: 'send next address to programmer
        bytReadData = bytReadROMData(): 'read data byte
        If bytReadData <> CByte(&HFF) Then
            blnEraseFlag = False
            Exit Do: 'not erased, so bail out of test loop
        End If
        intProgress = CInt((lngEPROMAddress / lngEndAddress) * 100): 'OK so far, so
        prgProgBar1.Value = intProgress: 'update progress bar
        lngEPROMAddress = lngEPROMAddress + 1: 'increment address & try again
    Loop
    prgProgBar1.Visible = False: 'finished, so turn off progress bar
    If blnEraseFlag = False Then
        strPrompt = "Data still present in EPROM at address: "
        strPrompt = strPrompt + Hex(lngEPROMAddress) + " hex"
        intResp = MsgBox(strPrompt, vbOKOnly, "Device not fully erased")
        Exit Sub
    Else
        strPrompt = "Device is fully erased!"
        intResp = MsgBox(strPrompt, vbOKOnly, "Checked Erasure")
    End If
End Sub

Private Sub mnuFileCloseData_Click()
    Close #1
End Sub

Private Sub mnuFileOpenBinary_Click()
    Dim lngFileLength As Long
    Dim intNoOf4KBytes As Integer, int4KBlockCtr As Integer
    Dim intColCtr As Integer, intBytCtr As Integer
```

```
    dbComDlg1.Filter = "Binary data file (*.bin)| *.bin"
    dbComDlg1.CancelError = True
    On Error GoTo dbErrorHandler
    dbComDlg1.ShowOpen
    strDataFile = dbComDlg1.FileName
    Open strDataFile For Binary As #1
    lngFileLength = LOF(1): 'find length of file
    lngStartAddress = 0
    lngEndAddress = lngFileLength - 1
    intNoOf4KBytes = (lngFileLength \ 4096): 'and hence no of 4K blocks
    For int4KBlockCtr = 1 To intNoOf4KBytes
       intColCtr = 0 + (int4KBlockCtr - 1): 'begin reading a 4K block
       For intBytCtr = 0 To 4095
          Get #1, , bytRomData(intColCtr, intBytCtr)
       Next intBytCtr
    Next int4KBlockCtr
    Close #1
    strPrompt = "File " + strDataFile + " is now open." + strCrLf
    strPrompt = strPrompt + "(" + CStr(lngFileLength) + " bytes)"
    intResp = MsgBox(strPrompt, vbOKOnly, "Binary file opened")
    lblCurrentDataFile.Caption = strDataFile
    Exit Sub

dbErrorHandler:
    Exit Sub
End Sub

Private Sub mnuFileOpenDeviceConfig_Click()
    Dim lngCapacity As Long, strDevName As String
    Dim bytTestBit As Byte
    dbComDlg1.Filter = "Device config file (*.ecf)| *.ecf"
    dbComDlg1.CancelError = True
    On Error GoTo dbErrorHandler
    dbComDlg1.ShowOpen
    strConfigFile = dbComDlg1.FileName
    Call OpenDevConfig
    Exit Sub

dbErrorHandler:
    Exit Sub
End Sub

Private Sub mnuFileExit_Click()
```

```
    Open "C:\Progra~1\EPROMProg\EpromPrg.cfg" For Output As #3
    Print #3, "Config file for SC EPROM Programmer"
    Print #3, CStr(intBaseAddress)
    Print #3, strConfigFile
    Close #3
    End
End Sub

Private Sub mnuFileSaveBinary_Click()
    Dim lngFileLength As Long
    Dim strFileExists As String
    Dim intNoOf4KBytes As Integer, int4KBlockCtr As Integer
    Dim intColCtr As Integer, intBytCtr As Integer
    dbComDlg1.Filter = "Binary data file (*.bin)| *.bin"
    dbComDlg1.CancelError = True
    On Error GoTo dbErrorHandler
    dbComDlg1.ShowSave
    strDataFile = dbComDlg1.FileName: 'get filename
    strFileExists = Dir(strDataFile): '& check if it already exists
    If strFileExists <> "" Then
       strPrompt = strDataFile + " already exists." + strCrLf
       strPrompt = strPrompt + "Want to overwrite?"
       intResp = MsgBox(strPrompt, vbYesNo, "Confirm overwrite")
       If intResp = 7 Then Exit Sub
    End If
    Open strDataFile For Binary As #1
    lngFileLength = lngEndAddress + 1: 'find file length
    intNoOf4KBytes = (lngFileLength \ 4096): 'and hence no of 4K blocks
    For int4KBlockCtr = 1 To intNoOf4KBytes
       intColCtr = 0 + (int4KBlockCtr - 1): 'begin saving a 4K block
       For intBytCtr = 0 To 4095
          Put #1, , bytRomData(intColCtr, intBytCtr)
       Next intBytCtr
    Next int4KBlockCtr
    Close #1
    strPrompt = "File " + strDataFile + " has been saved." + strCrLf
    strPrompt = strPrompt + "(" + CStr(lngFileLength) + " bytes)"
    intResp = MsgBox(strPrompt, vbOKOnly, "Binary file saved")
    lblCurrentDataFile.Caption = strDataFile
    Exit Sub

dbErrorHandler:
    Exit Sub
```

```
End Sub

Private Sub mnuFileSaveConfig_Click()
   Dim strTitle As String, strDevName As String, strCapVal As String
   Dim lngCapacity As Long
   strPrompt = "Give Device Name:"
   strDevName = InputBox(strPrompt, "Device name entry")
   If strDevName = "" Then strDevName = "(No name given)" + Space(17)
   strPrompt = "Give its capacity, in Kbits:"
   Do
   strCapVal = InputBox(strPrompt, "Device capacity entry")
   If strCapVal <> "" Then Exit Do
   Loop
   lngCapacity = CLng(Val(strCapVal) * 1024)
   strPrompt = "Have you set the programming pulse width?"
   strTitle = "Confirming that pulse width has been set"
   intResp = MsgBox(strPrompt, vbYesNoCancel, strTitle)
   If intResp = 2 Or intResp = 7 Then Exit Sub: 'Cancel or No...
   strPrompt = "Are the device config settings OK?"
   strTitle = "Confirming that configuration has been set"
   intResp = MsgBox(strPrompt, vbYesNoCancel, strTitle)
   If intResp = 2 Or intResp = 7 Then Exit Sub: 'Cancel or No...
   dbComDlg1.Filter = "Device config file (*.ecf)| *.ecf"
   dbComDlg1.CancelError = True
   On Error GoTo dbErrorHandler
   dbComDlg1.ShowSave
   strConfigFile = dbComDlg1.FileName
   Open strConfigFile For Output As #2
   Print #2, strDevName
   Print #2, lngCapacity
   Print #2, bytPulseWidth
   Print #2, bytPulseMult
   Print #2, bytDeviceConfig
   Close #2
   blnDevConfigFlag = True
   Exit Sub

dbErrorHandler:
   Exit Sub
End Sub

Private Sub mnuOpenIntelHex_Click()
   Dim lngUSBA As Long, lngRecOffset As Long
```

```
Dim strDataLine As String
Dim strValidRec As String, strReturn As String
Dim intRecLen As Integer, intRecType As Integer
Dim intColCtr As Integer, intBytCtr As Integer, intCtr As Integer
Dim blnEOFflag As Boolean
blnEOFflag = False
lngUSBA = 0
dbComDlg1.Filter = "Intel Hex data file (*.hex)| *.hex"
dbComDlg1.CancelError = True
On Error GoTo dbErrorHandler
dbComDlg1.ShowOpen
strDataFile = dbComDlg1.FileName
Open strDataFile For Input As #1
Do
   Line Input #1, strDataLine
   strValidRec = Left(strDataLine, 1)
   If strValidRec <> ":" Then
      strPrompt = "Invalid record! Aborting..."
      intResp = MsgBox(strPrompt, vbOKOnly, "Record start ID invalid")
      Exit Sub
   End If
   strResp = Mid(strDataLine, 2, 2): 'valid record, so extract RecLen chars
   strReturn = strConvHex(): 'and go check they're hex
   If blnHexFlag = False Then
      blnEOFflag = False
      Exit Do: 'wasn't valid hex, so bail out of loop
   End If
   intRecLen = 2 * Val(strReturn): 'it was hex, so fetch as RecLen in chars
   strResp = Mid(strDataLine, 4, 4): 'now get AddrOffset
   strReturn = strConvHex(): '& make sure it's valid hex
   If blnHexFlag = False Then
      blnEOFflag = False
      Exit Do: 'wasn't valid hex, so bail out of loop
   End If
   lngRecOffset = CLng(Val(strReturn)): 'was hex, so fetch as RecOffset
   strResp = Mid(strDataLine, 8, 2): 'now get RecType chars
   strReturn = strConvHex(): 'check to make sure they're hex
   If blnHexFlag = False Then
      blnEOFflag = False
      Exit Do: 'wasn't valid hex, so bail out of loop
   End If
   intRecType = Val(strReturn): 'was hex, so make it RecType
   Select Case intRecType
```

```
        Case 1
           blnEOFflag = True: 'we are at end of file, so prepare to leave
        Case 2
           strResp = Mid(strDataLine, 10, 4): 'an ESA record, so get USBA chars
           strReturn = strConvHex(): 'check they're valid hex
           If blnHexFlag = False Then
              blnEOFflag = False
              Exit Do: 'wasn't valid hex, so bail out of loop
           End If
           lngUSBA = CLng(Val(strReturn) * 16): 'was hex, so work out USBA
        Case 0
           For intCtr = 0 To (intRecLen - 2) Step 2
              strResp = Mid(strDataLine, (10 + intCtr), 2): 'data record, so process...
              strReturn = strConvHex(): 'check valid hex chars
              If blnHexFlag = False Then
                 blnEOFflag = False
                 Exit Do: 'wasn't valid hex, so bail out of loop
              End If
              lngEPROMAddress = lngUSBA + lngRecOffset + (intCtr / 2): 'valid, so get address
              intColCtr = lngEPROMAddress \ 4096: 'work out which 4K block we're in
              intBytCtr = CInt(lngEPROMAddress - 4096 * CLng(intColCtr)): '& offset in block
              bytRomData(intColCtr, intBytCtr) = CByte(Val(strReturn)): 'and save
           Next intCtr
        End Select
     Loop Until blnEOFflag = True
     If blnHexFlag = False Then
        strPrompt = "File " + strDataFile + " is corrupted!" + strCrLf
        strPrompt = strPrompt + "(Not fully loaded)"
        intResp = MsgBox(strPrompt, vbOKOnly, "Invalid hex found")
        Close #1
     Else
        Close #1: 'this is normal EOF exit path
        strPrompt = "File " + strDataFile + " is now open." + strCrLf
        strPrompt = strPrompt + "(" + CStr(lngEPROMAddress + 1) + " bytes)"
        intResp = MsgBox(strPrompt, vbOKOnly, "Intel Hex file opened")
        lblCurrentDataFile.Caption = strDataFile
     End If
     Exit Sub

dbErrorHandler:
     Exit Sub
End Sub
```

```
Private Sub mnuOpenMotorolaSRec_Click()
   Dim lngRecOffset As Long
   Dim strDataLine As String, strValidRec As String
   Dim strRecType As String, strReturn As String
   Dim intRecLen As Integer, intRecType As Integer
   Dim intAddrLen As Integer, intCtr As Integer
   Dim intDataLen As Integer, intFDataChar
   Dim intColCtr As Integer, intBytCtr As Integer
   Dim blnEOFflag As Boolean
   blnEOFflag = False
   dbComDlg1.Filter = "Motorola S-Record file (*.MOT)| *.MOT"
   dbComDlg1.CancelError = True
   On Error GoTo dbErrorHandler
   dbComDlg1.ShowOpen
   strDataFile = dbComDlg1.FileName
   Open strDataFile For Input As #1
   Do
      Line Input #1, strDataLine
      strValidRec = Left(strDataLine, 1): 'check we have an S record
      If strValidRec <> "S" Then
         strPrompt = "Invalid record! Aborting..."
         intResp = MsgBox(strPrompt, vbOKOnly, "Record start ID invalid")
         Exit Sub
      End If
      strRecType = Left(strDataLine, 2): 'now find record type
      Select Case strRecType
         Case "S9", "S8", "S7"
            blnEOFflag = True: 'an end of file record, so prepare to leave
            Exit Do
         Case "S3"
            intAddrLen = 8: 'record has a 4-byte address
         Case "S2"
            intAddrLen = 6: 'record has a 3-byte address
         Case "S1"
            intAddrLen = 4: 'record has a 2-byte address
         Case Else
            GoTo NoData: 'probably a header record, so skip it
      End Select
      strResp = Mid(strDataLine, 3, 2): 'a data record, so extract RecLen chars
      strReturn = strConvHex(): 'and go check they're hex
      If blnHexFlag = False Then
         blnEOFflag = False
         Exit Do: 'wasn't valid hex, so bail out of loop
```

```
      End If
      intRecLen = 2 * Val(strReturn): 'it was hex, so fetch as RecLen in chars
      strResp = Mid(strDataLine, 5, intAddrLen): 'now get AddrOffset
      strReturn = strConvHex(): '& make sure it's valid hex
      If blnHexFlag = False Then
         blnEOFflag = False
         Exit Do: 'wasn't valid hex, so bail out of loop
      End If
      lngRecOffset = CLng(Val(strReturn)): 'was hex, so fetch as RecOffset
      intDataLen = intRecLen - (intAddrLen + 2): 'work out no of data chars
      intFDataChar = 5 + intAddrLen: 'and index for first data char
      For intCtr = 0 To intDataLen Step 2
            strResp = Mid(strDataLine, (intFDataChar + intCtr), 2): '& get going
            strReturn = strConvHex(): 'check valid hex chars
            If blnHexFlag = False Then
               blnEOFflag = False
               Exit Do: 'wasn't valid hex, so bail out of loop
            End If
            lngEPROMAddress = lngRecOffset + (intCtr / 2): 'valid, so get address
            intColCtr = lngEPROMAddress \ 4096: 'work out which 4K block we're in
            intBytCtr = CInt(lngEPROMAddress - 4096 * CLng(intColCtr)): '& offset in block
            bytRomData(intColCtr, intBytCtr) = CByte(Val(strReturn)): 'and save data
      Next intCtr
NoData:
      Loop Until blnEOFflag = True
   If blnHexFlag = False Then
      strPrompt = "File " + strDataFile + " is corrupted!" + strCrLf
      strPrompt = strPrompt + "(Not fully loaded)"
      intResp = MsgBox(strPrompt, vbOKOnly, "Invalid hex found")
      Close #1
   Else
      Close #1: 'this is normal EOF exit path
      strPrompt = "File " + strDataFile + " is now open." + strCrLf
      strPrompt = strPrompt + "(" + CStr(lngEPROMAddress) + " bytes)"
      intResp = MsgBox(strPrompt, vbOKOnly, "Motorola S-Record file opened")
      lblCurrentDataFile.Caption = strDataFile
   End If
   Exit Sub

dbErrorHandler:
   Exit Sub
End Sub
```

```
Private Sub mnuProgAddrRange_Click()
   Dim lngLoopCtr As Long, blnReturn As Boolean
   Dim intColCtr As Integer, intRowCtr As Integer
   Call FindAddrRange: 'go get start & stop addresses
   If blnRangeOK = False Then
      Exit Sub: 'no success in getting range, so abort
   End If
   Call PreProgCheck
   If blnPreProgOK = False Then
      Exit Sub: 'check failed, so abort
   End If
   bytMode = CByte(4): 'set to programming mode and get going
   prgProgBar1.Value = 0: 'initialise progress bar
   prgProgBar1.Visible = True: 'and make it visible
   For lngLoopCtr = lngStartAddress To lngStopAddress
      lngEPROMAddress = lngLoopCtr
      Call DownloadAddress: 'send address to programmer
      intColCtr = CInt(lngEPROMAddress \ 4096): 'get storage array indices
      intRowCtr = CInt(lngEPROMAddress - (4096 * CLng(intColCtr)))
      bytWriteData = bytRomData(intColCtr, intRowCtr): '& fetch data byte
      blnReturn = blnWriteTheByte(bytWriteData): 'before sending it to EPROM
      If blnReturn = False Then
         prgProgBar1.Visible = False: 'programming failed, so turn off progress bar
         strPrompt = "Device programming failed at" + strCrLf
         strPrompt = strPrompt + "address" + Hex(lngEPROMAddress) + " hex"
         intResp = MsgBox(strPrompt, vbOKOnly, "Programming failed")
         Exit Sub
      End If
      If lngStopAddress = 0 Or lngStopAddress = lngStartAddress Then
         intProgress = 100: 'because we're at zero, and/or with only 1 to do
      Else
      intProgress = CInt(((lngEPROMAddress - lngStartAddress) / (lngStopAddress - lngStartAddress))
* 100)
      End If
      prgProgBar1.Value = intProgress: 'updating progress bar
   Next lngLoopCtr: 'that byte must have programmed OK, so loop & do next one
   prgProgBar1.Visible = False: 'finished, so turn off progress bar
   strPrompt = "Data programming complete...": 'advise user
   intResp = MsgBox(strPrompt, vbOKOnly, "Programming completed")
   bytMode = CByte(0): PortOut intModeAddress, bytMode: '& back to read mode for
return
End Sub
```

```
Private Sub mnuProgDev_Click()
   Dim intColCtr As Integer, intRowCtr As Integer
   Dim lngLoopCtr As Long, blnReturn As Boolean
   Call PreProgCheck
   If blnPreProgOK = False Then
      Exit Sub: 'check failed somehow, so abort
   End If
   strPrompt = "Sure you want to program complete device?"
   intResp = MsgBox(strPrompt, vbYesNoCancel, "Full device programming")
   If intResp <> 6 Then
      Exit Sub: 'not approved, so bail out
   End If
   Call VerifyCheck: 'all seems OK, so get verify flag and then begin
   bytMode = CByte(4): 'set to programming mode
   prgProgBar1.Value = 0: 'initialise progress bar
   prgProgBar1.Visible = True: 'and make it visible
   For lngLoopCtr = 0 To lngEndAddress
      lngEPROMAddress = lngLoopCtr
      Call DownloadAddress: 'send address to programmer
      intColCtr = CInt(lngEPROMAddress \ 4096): 'get storage array indices
      intRowCtr = CInt(lngEPROMAddress - (4096 * CLng(intColCtr)))
      bytWriteData = bytRomData(intColCtr, intRowCtr): '& fetch data byte
      blnReturn = blnWriteTheByte(bytWriteData): 'before sending it to EPROM
      If blnReturn = False Then
         prgProgBar1.Visible = False: 'failed, so turn off progress bar etc
         strPrompt = "Device programming failed at" + strCrLf
         strPrompt = strPrompt + "address" + Hex(lngEPROMAddress) + " hex"
         intResp = MsgBox(strPrompt, vbOKOnly, "Programming failed")
         Exit Sub
      End If
      intProgress = CInt((lngEPROMAddress / lngEndAddress) * 100)
      prgProgBar1.Value = intProgress: 'updating progress bar
   Next lngLoopCtr: 'that byte programmed OK, so loop & do next one
   prgProgBar1.Visible = False: 'finished, so turn off progress bar
   If blnVerifyFlag = False Then
      strPrompt = "Data programming complete..."
      intResp = MsgBox(strPrompt, vbOKOnly, "Programming completed")
      bytMode = CByte(0): PortOut intModeAddress, bytMode: 'back to read mode
      Exit Sub
   Else
      lngEPROMAddress = 0: 'wanted to verify after programming, so go do it
      lngStopAddress = lngEndAddress
      Call VerifyROMData: 'go verify ROM data
```

```
   End If
End Sub

Private Sub mnuProgVerify_Click()
   lngEPROMAddress = 0
   lngStopAddress = lngEndAddress
   Call VerifyROMData
End Sub

Private Sub mnuReadEPROMData_Click()
   Dim strReturn As String
   Dim intColCtr As Integer, intRowCtr As Integer
   Call FindAddrRange: 'go and get address range
   If blnRangeOK = False Then
      Exit Sub: 'range input wasn't successful, so bail out
   End If
   prgProgBar1.Value = 0: 'initialise progress bar
   prgProgBar1.Visible = True: 'and make it visible
   lngEPROMAddress = lngStartAddress: 'now set to starting address
   bytMode = CByte(0): 'and set mode for reading, before we get going
   PortOut intModeAddress, bytMode
   Do Until lngEPROMAddress > lngStopAddress
      Call DownloadAddress: 'send next address to programmer
      bytReadData = bytReadROMData(): 'read data byte
      intColCtr = CInt(lngEPROMAddress \ 4096): 'get storage array indices
      intRowCtr = CInt(lngEPROMAddress - (4096 * CLng(intColCtr)))
      bytRomData(intColCtr, intRowCtr) = bytReadData: '& save data there
      If lngStopAddress = 0 Then
         intProgress = 100: 'because we're at zero, & with only 1 to do
      Else
         intProgress = CInt((lngEPROMAddress / lngStopAddress) * 100)
      End If
      prgProgBar1.Value = intProgress: 'updating progress bar
      lngEPROMAddress = lngEPROMAddress + 1: 'now increment address
   Loop
   prgProgBar1.Visible = False: 'finished, so turn off progress bar
   strPrompt = "Data has been read..."
   intResp = MsgBox(strPrompt, vbOKOnly, "Reading Done")
   lblCurrentDataFile.Caption = "(Data read from EPROM)"
End Sub

Private Sub mnuSetPortAddress278_Click()
   intBaseAddress = &H278: intReadAddress = &H279: intModeAddress = &H27A
```

```
   lblBaseAddressDisp.Caption = "Programmer Base Address: 278 hex"
End Sub

Private Sub mnuSetPortAddress378_Click()
   intBaseAddress = &H378: intReadAddress = &H379: intModeAddress = &H37A
   lblBaseAddressDisp.Caption = "Programmer Base Address: 378 hex"
End Sub

Private Sub mnuSetPortAddress3BC_Click()
   intBaseAddress = &H3BC: intReadAddress = &H3BD: intModeAddress = &H3BE
   lblBaseAddressDisp.Caption = "Programmer Base Address: 3BC hex"
End Sub

Private Sub mnuSetProgPulse_Click()
   Dim strResp As String
   Dim strPrompt As String, lngLength As Long
   bytPulseMult = 1: 'initialise pulse multiplier to unity
   strPrompt = "Give required PGM* pulse duration" + strCrLf
   strPrompt = strPrompt + "in microseconds (1-50,000):"
   Do
      strResp = InputBox(strPrompt, "Set Pulse Duration", 50)
      If strResp = "" Then Exit Sub: 'User clicked cancel button so bail out
      lngLength = Val(strResp)
      If lngLength > 0 And lngLength <= 50000 Then Exit Do
   Loop
   If lngLength > 255 Then
      bytPulseMult = CByte(lngLength \ 250): 'long delay, so set multiplier
      lngLength = 250: 'and set basic pulse length to 250us
   End If
   bytPulseWidth = CByte(255 - lngLength): 'now set width code
   Call DisplayPulse: 'display it on screen
   Call SendPulseWidth: 'and send to programmer
End Sub

Private Sub mnuTest_Click()
   Dim intConfTest As Integer
   Dim strPrompt As String
   strPrompt = "Ready to test link to Programmer?" + strCrLf
   strPrompt = strPrompt + "(Will light 'Load PGM* Duration' LED)"
   intConfTest = MsgBox(strPrompt, vbOKCancel, "Test link")
   If intConfTest = 2 Then Exit Sub
   PortOut intBaseAddress, CByte(0)
   PortOut intModeAddress, CByte(6)
```

```
    strPrompt = "Is the Programmer's LED glowing?"
    intConfTest = MsgBox(strPrompt, vbOKCancel, "LED Flash Verify")
    PortOut intModeAddress, CByte(0)

End Sub

Private Sub mnuViewData_Click()
    Dim strViewStrg As String, strOffset As String
    Dim lngBytCtr As Long, lngBlockStart As Long
    Dim lngRange As Long, intNoOf4KBlox As Integer
    Dim int4KBlockCtr As Integer, intRecCtr As Integer
    Dim intRecBytCtr As Integer, intRowCtr As Integer
    Dim intNoOfLines As Integer
    Dim bytNewByte As Byte, strNewHex As String
    lstViewer.Clear
    lngRange = lngEndAddress + 1: 'work out total range
    intNoOf4KBlox = CInt(lngRange \ 4096): 'find how many 4K blocks
    If intNoOf4KBlox = 0 Then
       intNoOf4KBlox = 1: 'make sure we view at least part of a block
       intNoOfLines = CInt(lngRange \ 16): 'and find how many lines
    Else
       intNoOfLines = 255
    End If
    For int4KBlockCtr = 1 To intNoOf4KBlox: 'view-a-4K-block loop starts
       lngBlockStart = 0 + (4096& * CLng(int4KBlockCtr - 1))
       For intRecCtr = 0 To intNoOfLines: 'start of loop to count 16-byte lines
          lngBytCtr = lngBlockStart + (16 * intRecCtr)
          strViewStrg = ""
          strOffset = Right(Hex(lngBytCtr), 5): 'fetch offset for line
          If Len(strOffset) < 5 Then
             strOffset = String(5 - Len(strOffset), "0") + strOffset
          End If
          strViewStrg = strViewStrg + strOffset + "  ": '& add to string
          For intRecBytCtr = 0 To 15: 'then add record data itself
             intRowCtr = (intRecCtr * 16) + intRecBytCtr
             bytNewByte = bytRomData((int4KBlockCtr - 1), intRowCtr)
             strNewHex = strHexExp(Hex(bytNewByte))
             strViewStrg = strViewStrg + " " + strNewHex
          Next intRecBytCtr
          lstViewer.AddItem strViewStrg: 'add that 16-byte line to viewerlist
       Next intRecCtr
    Next int4KBlockCtr
End Sub
```

```
Private Sub optPin22a_Click()
   bytDeviceConfig = bytDeviceConfig And &HFD: 'clicked on PGM*, so CF1 = 0
   Call SendConfig
End Sub

Private Sub optPin22b_Click()
   bytDeviceConfig = bytDeviceConfig Or &H2: 'clicked on CE*, so CF1 = 1
   Call SendConfig
End Sub

Private Sub optPin24a_Click()
   bytDeviceConfig = bytDeviceConfig And &HFB
   Call SendConfig
End Sub

Private Sub optPin24b_Click()
   bytDeviceConfig = bytDeviceConfig Or &H4
   Call SendConfig
End Sub

Private Sub optPin29a_Click()
   bytDeviceConfig = bytDeviceConfig And &HF7
   Call SendConfig
End Sub

Private Sub optPin29b_Click()
   bytDeviceConfig = bytDeviceConfig Or &H8
   Call SendConfig
End Sub

Private Sub optPin30a_Click()
   bytDeviceConfig = bytDeviceConfig And &HEF
   Call SendConfig
End Sub

Private Sub optPin30b_Click()
   bytDeviceConfig = bytDeviceConfig Or &H10
   Call SendConfig
End Sub

Private Sub optPin3a_Click()
   bytDeviceConfig = bytDeviceConfig And &HFE
```

```
    Call SendConfig
End Sub

Private Sub optPin3b_Click()
    bytDeviceConfig = bytDeviceConfig Or &H1
    Call SendConfig
End Sub

Private Sub optVccProga_Click()
    bytDeviceConfig = bytDeviceConfig And &HBF
    Call SendConfig
End Sub

Private Sub optVccProgb_Click()
    bytDeviceConfig = bytDeviceConfig Or &H40
    Call SendConfig
End Sub

Private Sub optVccReada_Click()
    bytDeviceConfig = bytDeviceConfig And &HDF
    Call SendConfig
End Sub

Private Sub optVccReadb_Click()
    bytDeviceConfig = bytDeviceConfig Or &H20
    Call SendConfig
End Sub

Private Sub optVppVoltsa_Click()
    bytDeviceConfig = bytDeviceConfig And &H7F
    Call SendConfig
End Sub

Private Sub optVppVoltsb_Click()
    bytDeviceConfig = bytDeviceConfig Or &H80
    Call SendConfig
End Sub

Public Function bytTidyNibble(bytNibble As Byte) As Byte
    bytNibble = bytNibble And &HF0: 'strip away unwanted bits
    bytTidyNibble = bytNibble Xor &H80: 'then reinvert B7 for return
End Function
```

```
Public Function bytReadROMData() As Byte
   Dim bytTemp As Byte
   PortOut intModeAddress, CByte(0): 'ensure we're in RDH mode
   bytTemp = PortIn(intReadAddress): 'fetch hi nibble from EPROM
   bytReadData = bytTidyNibble(bytTemp): 'then tidy it up
   PortOut intModeAddress, CByte(2): 'now set to read low nibble
   bytTemp = PortIn(intReadAddress): 'fetch it too
   bytTemp = bytTidyNibble(bytTemp): '& also tidy it
   bytTemp = CByte(bytTemp \ 16): 'then move the bits down
   bytReadData = bytReadData Or bytTemp: 'finally glue them together
   PortOut intModeAddress, CByte(0): 'before going back to RDH mode
   bytReadROMData = bytReadData: 'and preparing for return

End Function

Public Sub DownloadAddress()
   Dim bytTopAddr As Byte, bytMidAddr As Byte, bytLowAddr As Byte
   Dim lngWkgAddr As Long
   lngWkgAddr = lngEPROMAddress: 'set wkg address, work out bytes
   If lngWkgAddr < 65536 Then
      bytTopAddr = 0
      Else: bytTopAddr = CByte(lngWkgAddr \ 65536)
   End If
   lngWkgAddr = lngWkgAddr - (bytTopAddr * 65536)
   If lngWkgAddr < 256 Then
      bytMidAddr = 0
      Else: bytMidAddr = CByte(lngWkgAddr \ 256)
   End If
   lngWkgAddr = lngWkgAddr - (CLng(bytMidAddr) * 256)
   bytLowAddr = CByte(lngWkgAddr)
   PortOut intModeAddress, CByte(0): 'make sure we're not in prog mode
   PortOut intBaseAddress, bytTopAddr: 'download high bits
   PortOut intModeAddress, CByte(14)
   PortOut intModeAddress, CByte(0)
   PortOut intBaseAddress, bytMidAddr: 'and mid byte
   PortOut intModeAddress, CByte(8)
   PortOut intModeAddress, CByte(0)
   PortOut intBaseAddress, bytLowAddr: 'and low byte
   PortOut intModeAddress, CByte(10)
   PortOut intModeAddress, CByte(0): 'then back to RDH mode
   lblCurrDevAddressDisp.Caption = Hex(lngEPROMAddress) + "hex": '& display address
End Sub
```

```
Public Function strConvHex() As String
   Dim strChar As String, intCtr As Integer
   Dim lngMult As Long, lngConvValue As Long
   lngConvValue = 0
   For intCtr = 1 To Len(strResp)
      strChar = Mid(strResp, intCtr, 1)
      Select Case strChar
         Case "0" To "9"
            blnHexFlag = True
         Case "A" To "F"
            blnHexFlag = True
         Case "a" To "f"
            blnHexFlag = True
         Case Else
            blnHexFlag = False
      End Select
   Next intCtr
   If blnHexFlag = False Then Exit Function: 'non-hex char found...
   For intCtr = Len(strResp) To 1 Step -1
      strChar = Mid(strResp, intCtr, 1)
      lngMult = 16 ^ (Len(strResp) - intCtr)
      Select Case strChar
         Case "0" To "9"
            lngConvValue = lngConvValue + (Val(strChar) * lngMult)
         Case "A", "a"
            lngConvValue = lngConvValue + (10 * lngMult)
         Case "B", "b"
            lngConvValue = lngConvValue + (11 * lngMult)
         Case "C", "c"
            lngConvValue = lngConvValue + (12 * lngMult)
         Case "D", "d"
            lngConvValue = lngConvValue + (13 * lngMult)
         Case "E", "e"
            lngConvValue = lngConvValue + (14 * lngMult)
         Case "F", "f"
            lngConvValue = lngConvValue + (15 * lngMult)
      End Select
   Next intCtr
   strConvHex = CStr(lngConvValue)
End Function

Public Sub DisplayPulse()
   Dim strCapHeader As String, strTrueWidth As String
```

```
    Dim sngTrueWidth As Single
    strCapHeader = "Programming Pulse Width: "
    sngTrueWidth = CSng(255 - bytPulseWidth): 'work out true pulse width from code
    If bytPulseMult > 1 Then
       sngTrueWidth = CSng(CSng(bytPulseMult) * (sngTrueWidth / 1000)): 'bigger, so expand
       strTrueWidth = CStr(sngTrueWidth): '& display in milliseconds
       lblPulseDurationDisp.Caption = strCapHeader + strTrueWidth + "ms"
    Else
       strTrueWidth = CStr(sngTrueWidth): 'less than 255, so use it as is
    lblPulseDurationDisp.Caption = strCapHeader + strTrueWidth + "us"
    End If
End Sub

Public Sub OpenDevConfig()
    Dim strDevName As String, lngCapacity As Long
    Dim lngByteSize As Long, bytTestBit As Byte
    On Error GoTo ErrHandler
    Open strConfigFile For Input As #2
    Input #2, strDevName: 'fetch device name
    lblDevConfigDesc.Caption = strDevName: '& display
    Input #2, lngCapacity: 'fetch capacity in bits
    lngByteSize = lngCapacity \ 8: 'work out number of bytes
    lngStartAddress = 0: 'use it to set address range
    lngEndAddress = lngByteSize - 1
    lblChipOrgDesc.Caption = CStr(lngByteSize) + " x 8": '& display
    Input #2, bytPulseWidth: 'fetch pulse width
    Input #2, bytPulseMult: 'and multiplier
    Call DisplayPulse: 'display full width
    Call SendPulseWidth: 'and send basic width to programmer
    Input #2, bytDeviceConfig: 'fetch config byte, then show settings
       bytTestBit = bytDeviceConfig And 1
       If bytTestBit = 0 Then
          optPin3a.Value = True: optPin3b.Value = False
       Else:
          optPin3b.Value = True: optPin3a.Value = False
       End If
       bytTestBit = bytDeviceConfig And 2
       If bytTestBit = 0 Then
          optPin22a.Value = True: optPin22b.Value = False
       Else:
          optPin22b.Value = True: optPin22a.Value = False
       End If
       bytTestBit = bytDeviceConfig And 4
```

```
    If bytTestBit = 0 Then
       optPin24a.Value = True: optPin24b.Value = False
    Else:
       optPin24b.Value = True: optPin24a.Value = False
    End If
    bytTestBit = bytDeviceConfig And 8
    If bytTestBit = 0 Then
       optPin29a.Value = True: optPin29b.Value = False
    Else:
       optPin29b.Value = True: optPin29a.Value = False
    End If
    bytTestBit = bytDeviceConfig And 16
    If bytTestBit = 0 Then
       optPin30a.Value = True: optPin30b.Value = False
    Else:
       optPin30b.Value = True: optPin30a.Value = False
    End If
    bytTestBit = bytDeviceConfig And 32
    If bytTestBit = 0 Then
       optVccReada.Value = True: optVccReadb.Value = False
    Else:
       optVccReadb.Value = True: optVccReada.Value = False
    End If
    bytTestBit = bytDeviceConfig And 64
    If bytTestBit = 0 Then
       optVccProga.Value = True: optVccProgb.Value = False
    Else:
       optVccProgb.Value = True: optVccProga.Value = False
    End If
    bytTestBit = bytDeviceConfig And 128
    If bytTestBit = 0 Then
       optVppVoltsa.Value = True: optVppVoltsb.Value = False
    Else:
       optVppVoltsb.Value = True: optVppVoltsa.Value = False
    End If
  Close #2
  Call SendConfig: 'finally send config to programmer to update it
  blnDevConfigFlag = True: 'we have a valid config, so set flag
  Exit Sub

ErrHandler:
  Exit Sub
```

```
End Sub

Public Function strHexExp(strIn As String) As String
   If Len(strIn) = 1 Then strIn = "0" + strIn
   strHexExp = strIn
End Function

Public Sub SendConfig()
   PortOut intBaseAddress, bytDeviceConfig: 'put config byte on data bus
   PortOut intModeAddress, CByte(12): 'pulse LCF line to store
   PortOut intModeAddress, bytMode: '& restore bytMode before return
End Sub

Public Sub SendPulseWidth()
   PortOut intBaseAddress, bytPulseWidth: 'download width/duration
   PortOut intModeAddress, CByte(6): 'send pulse to load into register
   PortOut intModeAddress, bytMode: 'then restore bytMode before return
End Sub

Public Sub VerifyROMData()
   Dim strReturn As String, blnMatch As Boolean
   Dim intConfTest As Integer, intColCtr As Integer, intRowCtr As Integer
   lngEPROMAddress = 0: blnMatch = True
   prgProgBar1.Value = 0: 'initialise progress bar
   prgProgBar1.Visible = True: 'and make it visible
   PortOut intModeAddress, CByte(0): 'set programmer for read mode
   Do Until lngEPROMAddress > lngStopAddress
      Call DownloadAddress: 'send address to programmer
      bytReadData = bytReadROMData: 'read data byte from PROM
      intColCtr = CInt(lngEPROMAddress \ 4096): 'get storage array indices
      intRowCtr = CInt(lngEPROMAddress - (4096 * CLng(intColCtr)))
      If bytReadData <> bytRomData(intColCtr, intRowCtr) Then
      blnMatch = False: Exit Do
      End If
      intProgress = CInt((lngEPROMAddress / lngStopAddress) * 100)
      prgProgBar1.Value = intProgress: 'updating progress bar
      lngEPROMAddress = lngEPROMAddress + 1: 'matched, so increment address & continue
   Loop
   prgProgBar1.Visible = False: 'finished, so turn off progress bar
   If blnMatch = True Then
      strPrompt = "Device Verifies OK!"
      intResp = MsgBox(strPrompt, vbOKOnly, "Verify completed")
      Exit Sub
```

```
    Else
        strPrompt = "Device verifying failed at" + strCrLf
        strPrompt = strPrompt + "address" + Hex(lngEPROMAddress) + " hex"
        intResp = MsgBox(strPrompt, vbOKOnly, "Verifying problem")
    End If
End Sub

Public Function blnWriteTheByte(bytDataOut As Byte) As Boolean
    Dim intLoopCtr As Integer, bytReadit As Byte
    PortOut intModeAddress, bytMode: 'first ensure programmer is in prog mode
    PortOut intBaseAddress, bytDataOut: 'then send data byte to it
    For intLoopCtr = 1 To 10
        PortOut intModeAddress, bytMode: 'three more prog mode cmds for slight delay
        PortOut intModeAddress, bytMode: 'to allow OE* time to rise & settle
        PortOut intModeAddress, bytMode: ' at least 2us before PGM* pulse starts
        Call SendProgPulse: 'now send a prog trigger pulse & wait until PGM* ends
        PortOut intModeAddress, CByte(0): 'then swing over to read mode
        bytReadit = bytReadROMData: 'so we can check if data is stored yet
        If bytReadit = bytDataOut Then
            blnWriteTheByte = True: 'yes, data seems to be in ROM address
            Exit Function: 'so return
        End If
        PortOut intModeAddress, bytMode: 'having another try, so return to prog mode
        PortOut intModeAddress, bytMode: 'with a bit more delay for OE* settling...
    Next intLoopCtr
    blnWriteTheByte = False: 'left loop after 10 tries -- programming failed
    PortOut intModeAddress, CByte(0): 'so switch back to read mode & return
End Function

Public Sub SendProgPulse()
    Dim bytTemp As Byte, bytLoopCtr As Byte
    For bytLoopCtr = 1 To bytPulseMult
        PortOut intModeAddress, bytMode + 1: 'send a programming trigger pulse
        PortOut intModeAddress, bytMode: ' and end it
        Do
            bytTemp = PortIn(intReadAddress): 'now wait until main prog pulse ends
            bytTemp = bytTemp And 8: 'masking out PGM* flag bit
        Loop Until bytTemp <> 0
    Next bytLoopCtr: 'loop back if we're sending more than one prog pulse
End Sub

Public Sub FindAddrRange()
    Dim strReturn As String
```

```
strPrompt = "Give first address, in hex:"
Do
   strResp = InputBox(strPrompt, "Set start address", "00000")
   If strResp = "" Then
      Exit Do: 'user clicked cancel, so leave loop
   End If
   strReturn = strConvHex(): 'go check hex
   If blnHexFlag = False Then
      Exit Do: 'wasn't valid hex, so again leave loop
   End If
   lngHexVal = CLng(Val(strReturn)): 'was hex, so get & check
   If lngHexVal >= 0 And lngHexVal < lngEndAddress Then
      Exit Do: 'all seems OK, so leave loop with head high!
   End If
Loop
If strResp = "" Then
   blnRangeOK = False: 'user clicked cancel, so lower range flag
   Exit Sub: 'and bail out
End If
If blnHexFlag = False Then
   blnRangeOK = False: 'they gave invalid entry, so likewise
   Exit Sub
End If
blnRangeOK = True: 'hex was OK, so raise range flag for time being...
lngStartAddress = lngHexVal: '& make entry start address
strPrompt = "Give last address, in hex:"
Do
   strResp = InputBox(strPrompt, "Set stop address", Hex(lngEndAddress))
   If strResp = "" Then
      Exit Do: 'they clicked cancel, so exit loop
   End If
   strReturn = strConvHex(): 'go check hex, get dec value as string
   If blnHexFlag = False Then
      Exit Do: 'wasn't valid hex
   End If
   lngHexVal = CLng(Val(strReturn)): 'it was hex, so get & check
   If lngHexVal >= lngStartAddress And lngHexVal <= lngEndAddress Then
      Exit Do
   End If
Loop
If strResp = "" Then
   blnRangeOK = False: 'user clicked cancel again, so lower flag
   Exit Sub: 'and bail out
```

```
    End If
    If blnHexFlag = False Then
        blnRangeOK = False: 'they gave invalid entry, so likewise
        Exit Sub
    End If
    blnRangeOK = True: 'this one seems OK too, so raise flag again
    lngStopAddress = lngHexVal: '& set as stop address before leaving

End Sub

Public Sub PreProgCheck()
    If strDataFile = "" Then
        strPrompt = "No data file opened as yet!"
        intResp = MsgBox(strPrompt, vbOKOnly, "No programming data")
        blnPreProgOK = False: ' no data to program with, so lower flag
        Exit Sub: 'and bail out
    End If
    If blnDevConfigFlag = False Then
        strPrompt = "Device configuration not set!"
        intResp = MsgBox(strPrompt, vbOKOnly, "Configuration not set")
        blnPreProgOK = False: 'no configuration set, so again lower flag
        Exit Sub: 'and bail out
    End If
    strPrompt = "Have you set the programming pulse width?"
    intResp = MsgBox(strPrompt, vbYesNoCancel, "Pulse width checking")
    If intResp <> 6 Then
        blnPreProgOK = False: 'cancel or no again, so lower flag
        Exit Sub: 'and go
    End If
    blnPreProgOK = True: 'OK so far, so raise flag and return

End Sub

Public Sub VerifyCheck()
    strPrompt = "Verify data after programming?"
    intResp = MsgBox(strPrompt, vbYesNoCancel, "Verify after program")
    If intResp = 6 Then
        blnVerifyFlag = True
    Else: blnVerifyFlag = False
    End If

End Sub
```