frmAbout - 1

```
' Reg Key Security Options...
Const KEY_ALL_ACCESS = &H2003F


' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1                        ' Unicode nul terminated string
Const REG_DWORD = 4                     ' 32-bit number


Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"


Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal
 lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, ByRef phkResult As Long) As L
ong
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA" (ByVal hKey As Long,
 ByVal lpValueName As String, ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData As String,
 ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long

Private Sub Form_Load()
    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
    lblTitle.Caption = App.Title
    lblDescription.Caption = App.Comments
End Sub




Private Sub cmdSysInfo_Click()
        Call StartSysInfo
End Sub


Private Sub cmdOK_Click()
        Unload Me
End Sub


Public Sub StartSysInfo()
    On Error GoTo SysInfoErr


        Dim rc As Long
        Dim SysInfoPath As String


        ' Try To Get System Info Program Path\Name From Registry...
        If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO, SysInfoPath) Then
        ' Try To Get System Info Program Path Only From Registry...
        ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC, gREGVALSYSINFOLOC, SysInfoPath) The
n
                ' Validate Existance Of Known 32 Bit File Version
                If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
                        SysInfoPath = SysInfoPath & "\MSINFO32.EXE"


                ' Error - File Can Not Be Found...
                Else
                        GoTo SysInfoErr
                End If
        ' Error - Registry Entry Can Not Be Found...
        Else
                GoTo SysInfoErr
        End If
```

```
        Call Shell(SysInfoPath, vbNormalFocus)

        Exit Sub
SysInfoErr:
        MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub


Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As String, ByRef KeyVal As
String) As Boolean
        Dim i As Long                                           ' Loop Counter
        Dim rc As Long                                          ' Return Code
        Dim hKey As Long                                        ' Handle To An Open Registry Key
        Dim hDepth As Long                                      '
        Dim KeyValType As Long                                  ' Data Type Of A Registry Key
        Dim tmpVal As String                                    ' Tempory Storage For A Registry Key
Value
        Dim KeyValSize As Long                                  ' Size Of Registry Key Variable
        '------------------------------------------------------------
        ' Open RegKey Under KeyRoot {HKEY_LOCAL_MACHINE...}
        '------------------------------------------------------------
        rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open Registry Key


        If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle Error...


        tmpVal = String$(1024, 0)                             ' Allocate Variable Space
        KeyValSize = 1024                                       ' Mark Variable Size


        '------------------------------------------------------------
        ' Retrieve Registry Key Value...
        '------------------------------------------------------------
        rc = RegQueryValueEx(hKey, SubKeyRef, 0, KeyValType, tmpVal, KeyValSize)    ' Get/Create Key
Value


        If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Handle Errors


        If (Asc(Mid(tmpVal, KeyValSize, 1)) = 0) Then           ' Win95 Adds Null Terminated String..
.
                tmpVal = Left(tmpVal, KeyValSize - 1)               ' Null Found, Extract From String
        Else                                                    ' WinNT Does NOT Null Terminate Strin
g...
                tmpVal = Left(tmpVal, KeyValSize)                   ' Null Not Found, Extract String
Only
        End If
        '------------------------------------------------------------
        ' Determine Key Value Type For Conversion...
        '------------------------------------------------------------
        Select Case KeyValType                                  ' Search Data Types...
        Case REG_SZ                                             ' String Registry Key Data Type
                KeyVal = tmpVal                                     ' Copy String Value
        Case REG_DWORD                                          ' Double Word Registry Key Data Type
                For i = Len(tmpVal) To 1 Step -1                    ' Convert Each Bit
                        KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))   ' Build Value Char. By Char.
                Next
                KeyVal = Format$("&h" + KeyVal)                     ' Convert Double Word To String
        End Select


        GetKeyValue = True                                      ' Return Success
        rc = RegCloseKey(hKey)                                  ' Close Registry Key
        Exit Function                                           ' Exit


GetKeyError:    ' Cleanup After An Error Has Occured...
        KeyVal = ""                                             ' Set Return Val To Empty String
```

```
        GetKeyValue = False                                     ' Return Failure
        rc = RegCloseKey(hKey)                                  ' Close Registry Key
End Function

Private Sub picIcon_Click()

End Sub
```

frmAbout - 1

```
VERSION 5.00
Begin VB.Form frmAbout
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Trimline Monitor (PD-2116-058)"
   ClientHeight    =   3105
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   5625
   ClipControls    =   0   'False
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   3105
   ScaleWidth      =   5625
   StartUpPosition =   1  'CenterOwner
   Tag             =   "About PD2116051"
   Begin VB.CommandButton cmdOK
      Cancel          =   -1  'True
      Caption         =   "OK"
      Default         =   -1  'True
      Height          =   345
      Left            =   4245
      TabIndex        =   0
      Tag             =   "OK"
      Top             =   2625
      Width           =   1260
   End
   Begin VB.CommandButton cmdSysInfo
      Caption         =   "&System Info..."
      Height          =   345
      Left            =   240
      TabIndex        =   1
      Tag             =   "&System Info..."
      Top             =   2640
      Width           =   1245
   End
   Begin VB.Label lblDescription
      Caption         =   "App Description"
      ForeColor       =   &H00000000&
      Height          =   1170
      Left            =   1050
      TabIndex        =   4
      Tag             =   "App Description"
      Top             =   1125
      Width           =   3885
   End
   Begin VB.Label lblTitle
      Caption         =   "Application Title"
      ForeColor       =   &H00000000&
      Height          =   480
      Left            =   1050
      TabIndex        =   3
      Tag             =   "Application Title"
      Top             =   240
      Width           =   3885
   End
   Begin VB.Line Line1
      BorderColor     =   &H00808080&
      BorderStyle     =   6  'Inside Solid
      Index           =   1
      X1              =   225
      X2              =   5450
      Y1              =   2430
      Y2              =   2430
   End
   Begin VB.Line Line1
      BorderColor     =   &H00FFFFFF&
      BorderWidth     =   2
      Index           =   0
      X1              =   240
      X2              =   5450
      Y1              =   2445
```

```
      Y2              =   2445
   End
   Begin VB.Label lblVersion
      Caption         =   "Version"
      Height          =   225
      Left            =   1080
      TabIndex        =   2
      Tag             =   "Version"
      Top             =   780
      Width           =   3885
   End
End
```

frmChangePassword - 1

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------
Private Sub Form_Activate()

    'Clear out the form
    tbOldPassword.Text = ""
    tbNewPassword.Text = ""
    tbNewPassword2.Text = ""
    tbOldPassword.SetFocus

    'Center Dialog over Main Form
    Call CenterWindow(Me)

End Sub


'--------------------------------------------------------------------------
' OK Button
'--------------------------------------------------------------------------
Private Sub bChange_Click()

    'Validate old password first
    If tbOldPassword.Text = SecretWord Or tbOldPassword = "none" Then

        'Verify NewPassword, and NewPassword2 match
        If tbNewPassword.Text = tbNewPassword2.Text Then
            'everything is good
            SecretWord = tbNewPassword.Text
            Call SaveSecretWord
            MsgBox "New password accepted!", vbInformation
        Else
            ' NewPassword and NewPassword2 do not match
            MsgBox "The two new password entries do not match!" + Chr(13) + _
            "The password was not changed!", vbCritical
        End If
```

```
    Else
        'OldPassword is not correct
        MsgBox "The old password is not correct!" + Chr(13) + _
        "The password was not changed!", vbCritical
    End If
    Unload Me

End Sub


'--------------------------------------------------------------------------
' CANCEL Button
'--------------------------------------------------------------------------
Private Sub bCancel_Click()

    'User cancled operation, go away
    Unload Me

End Sub
```

```
frmChangePassword - 1

VERSION 5.00
Begin VB.Form frmChangePassword
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Change Password"
   ClientHeight    =   2295
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   4200
   ControlBox      =   0   'False
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   2295
   ScaleWidth      =   4200
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton bCancel
      Caption         =   "Cancel"
      Height          =   495
      Left            =   1200
      TabIndex        =   7
      ToolTipText     =   "Cancel changing the password."
      Top             =   1680
      Width           =   1215
   End
   Begin VB.CommandButton bChange
      Caption         =   "Change Password"
      Height          =   495
      Left            =   2520
      TabIndex        =   6
      ToolTipText     =   "Try to change the password from the old one to the new one."
      Top             =   1680
      Width           =   1575
   End
   Begin VB.TextBox tbNewPassword2
      Height          =   375
      IMEMode         =   3  'DISABLE
      Left            =   1560
      PasswordChar    =   "*"
      TabIndex        =   5
      ToolTipText     =   "Enter the 2nd copy of the new password."
      Top             =   1080
      Width           =   2535
   End
   Begin VB.TextBox tbNewPassword
      Height          =   375
      IMEMode         =   3  'DISABLE
      Left            =   1560
      PasswordChar    =   "*"
      TabIndex        =   4
      ToolTipText     =   "Enter the 1st copy of the new password."
      Top             =   600
      Width           =   2535
   End
   Begin VB.TextBox tbOldPassword
      Height          =   375
      IMEMode         =   3  'DISABLE
      Left            =   1560
      PasswordChar    =   "*"
      TabIndex        =   3
      ToolTipText     =   "Enter the current password to authorize changing the password."
      Top             =   120
      Width           =   2535
   End
   Begin VB.Label Label3
      Caption         =   "New Password"
      Height          =   255
      Left            =   120
      TabIndex        =   2
      Top             =   1080
      Width           =   1215
   End
   Begin VB.Label Label2
```

```
frmChangePassword - 2

      Caption         =   "New Password"
      Height          =   255
      Left            =   120
      TabIndex        =   1
      Top             =   600
      Width           =   1215
   End
   Begin VB.Label Label1
      Caption         =   "Old Password"
      Height          =   255
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   1215
   End
End
```

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
Option Explicit

'Local Storage for the Schedule under Edit
Dim Config(1 To NUMSCHEDULE) As SCHEDULETYPE


'--------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------
Private Sub Form_Load()
    Dim Unit As Integer
    Dim i As Integer

    'Restore form to proper position
    Call LoadWindowPosition(Me)

    'Make a local copy of the Schedule Data
    For Unit = LBound(Config) To UBound(Config)

        'Copy the Schedule Name
        Config(Unit).Name = Schedule(Unit).Name

        'Copy the Shift Data
        For i = LBound(Config(Unit).Shift) To UBound(Config(Unit).Shift)
            Config(Unit).Shift(i).Rst = Schedule(Unit).Shift(i).Rst
            Config(Unit).Shift(i).Exp = Schedule(Unit).Shift(i).Exp
            Config(Unit).Shift(i).Sun = Schedule(Unit).Shift(i).Sun
            Config(Unit).Shift(i).Mon = Schedule(Unit).Shift(i).Mon
            Config(Unit).Shift(i).Tue = Schedule(Unit).Shift(i).Tue
            Config(Unit).Shift(i).Wed = Schedule(Unit).Shift(i).Wed
            Config(Unit).Shift(i).Thr = Schedule(Unit).Shift(i).Thr
            Config(Unit).Shift(i).Fri = Schedule(Unit).Shift(i).Fri
            Config(Unit).Shift(i).Sat = Schedule(Unit).Shift(i).Sat
            Config(Unit).Shift(i).Beg = Schedule(Unit).Shift(i).Beg
            Config(Unit).Shift(i).End = Schedule(Unit).Shift(i).End
        Next i

    Next Unit
```

```
    'Load the cSchedule Drop List
    For Unit = LBound(Config) To UBound(Config)
        cSchedule.AddItem "#" & Format(Unit, "0) ") & Config(Unit).Name
    Next Unit
    cSchedule.ListIndex = 0

End Sub


Private Sub Form_Unload(Cancel As Integer)

    'Save Form position
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------
' User Selected a Schedule, Load the Form
'--------------------------------------------------------------------------
Private Sub cSchedule_Click()
    Dim Unit As Integer
    Dim i As Integer

    'Identify Which Schedule is Selected
    Unit = cSchedule.ListIndex + 1

    'Populate the Form with the Selected Schedule
   tName = Config(Unit).Name

    'Copy the Shift Data into the Form
    For i = LBound(Config(Unit).Shift) To UBound(Config(Unit).Shift)
        cbShiftRst(i - 1) = Config(Unit).Shift(i).Rst
        cbShiftExp(i - 1) = Config(Unit).Shift(i).Exp
        cbShiftSun(i - 1) = Config(Unit).Shift(i).Sun
        cbShiftMon(i - 1) = Config(Unit).Shift(i).Mon
        cbShiftTue(i - 1) = Config(Unit).Shift(i).Tue
        cbShiftWed(i - 1) = Config(Unit).Shift(i).Wed
        cbShiftThr(i - 1) = Config(Unit).Shift(i).Thr
        cbShiftFri(i - 1) = Config(Unit).Shift(i).Fri
        cbShiftSat(i - 1) = Config(Unit).Shift(i).Sat
        meShiftBeg(i - 1) = Config(Unit).Shift(i).Beg
        meShiftEnd(i - 1) = Config(Unit).Shift(i).End
    Next i

End Sub


'--------------------------------------------------------------------------
' As the User Edits the Schedule, Keep the Config() Array in Sync
'--------------------------------------------------------------------------
'User Edited the Schedule Name
Private Sub tName_Change()
    Config(cSchedule.ListIndex + 1).Name = tName
    cSchedule.List(cSchedule.ListIndex) = "#" & Format(cSchedule.ListIndex + 1, "0) ") & Config(cSche
dule.ListIndex + 1).Name
End Sub

'User Edited a Reset Check Box
Private Sub cbShiftRst_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Rst = cbShiftRst(Index)
End Sub

'User Edited a Export Check Box
Private Sub cbShiftExp_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Exp = cbShiftExp(Index)
End Sub

'User Edited a Shift Sunday Check Box
Private Sub cbShiftSun_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Sun = cbShiftSun(Index)
End Sub
```

```
'User Edited a Shift Monday Check Box
Private Sub cbShiftMon_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Mon = cbShiftMon(Index)
End Sub

'User Edited a Shift Tuesday Check Box
Private Sub cbShiftTue_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Tue = cbShiftTue(Index)
End Sub

'User Edited a Shift Wednesday Check Box
Private Sub cbShiftWed_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Wed = cbShiftWed(Index)
End Sub

'User Edited a Shift Thrusday Check Box
Private Sub cbShiftThr_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Thr = cbShiftThr(Index)
End Sub

'User Edited a Shift Friday Check Box
Private Sub cbShiftFri_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Fri = cbShiftFri(Index)
End Sub

'User Edited a Shift Saturday Check Box
Private Sub cbShiftSat_Click(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Sat = cbShiftSat(Index)
End Sub

'User Edited a Shift Begin Time
Private Sub meShiftBeg_LostFocus(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).Beg = meShiftBeg(Index)
End Sub

'User Edited a Shift End Time
Private Sub meShiftEnd_LostFocus(Index As Integer)
    Config(cSchedule.ListIndex + 1).Shift(Index + 1).End = meShiftEnd(Index)
End Sub


'------------------------------------------------------------------------
' OK, Apply, and Cancel Buttons...
'------------------------------------------------------------------------
Private Sub pb_OK_Click()
    'Password Protected Operation
    If (CheckPassword(Me) = True) Then

        'Apply All Changes to the Configuration and go away
        Call ApplyChanges
        Unload Me

    End If

End Sub


Private Sub pb_Apply_Click()

    'Password Protected Operation
    If (CheckPassword(Me) = True) Then

        'Apply All Changes to the Configuration
        Call ApplyChanges

    End If

End Sub

Private Sub pb_Cancel_Click()
```

```
    'User clicked on Cancel
    Unload Me

End Sub


Public Sub ApplyChanges()
    Dim Unit As Integer
    Dim i As Integer

    'Save Schedule changes Back
    For Unit = LBound(Config) To UBound(Config)

        'Copy the Schedule Name
        Schedule(Unit).Name = Config(Unit).Name

        'Copy the Shift Data
        For i = LBound(Config(Unit).Shift) To UBound(Config(Unit).Shift)
            Schedule(Unit).Shift(i).Rst = Config(Unit).Shift(i).Rst
            Schedule(Unit).Shift(i).Exp = Config(Unit).Shift(i).Exp
            Schedule(Unit).Shift(i).Sun = Config(Unit).Shift(i).Sun
            Schedule(Unit).Shift(i).Mon = Config(Unit).Shift(i).Mon
            Schedule(Unit).Shift(i).Tue = Config(Unit).Shift(i).Tue
            Schedule(Unit).Shift(i).Wed = Config(Unit).Shift(i).Wed
            Schedule(Unit).Shift(i).Thr = Config(Unit).Shift(i).Thr
            Schedule(Unit).Shift(i).Fri = Config(Unit).Shift(i).Fri
            Schedule(Unit).Shift(i).Sat = Config(Unit).Shift(i).Sat
            Schedule(Unit).Shift(i).Beg = Config(Unit).Shift(i).Beg
            Schedule(Unit).Shift(i).End = Config(Unit).Shift(i).End
        Next i

    Next Unit

    'Save this configuration change to disk now!
    Call SaveAllConfiguration

End Sub
```

```
frmConfigSchedules - 1

VERSION 5.00
Object = "{C932BA88-4374-101B-A56C-00AA003668DC}#1.1#0"; "MSMASK32.OCX"
Begin VB.Form frmConfigSchedules
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Configure Weekly Schedules"
   ClientHeight    =   5160
   ClientLeft      =   255
   ClientTop       =   540
   ClientWidth     =   7905
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   5160
   ScaleWidth      =   7905
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   7
      Left            =   4800
      TabIndex        =   103
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   7
      Left            =   4440
      TabIndex        =   102
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   7
      Left            =   4080
      TabIndex        =   101
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   7
      Left            =   3720
      TabIndex        =   100
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   7
      Left            =   3360
      TabIndex        =   99
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   7
      Left            =   3000
      TabIndex        =   98
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSun
      Height          =   255
      Index           =   7
      Left            =   2640
      TabIndex        =   97
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
```

```
      Height          =   255
      Index           =   7
      Left            =   2040
      TabIndex        =   96
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   7
      Left            =   1440
      TabIndex        =   95
      Top             =   4200
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   6
      Left            =   4800
      TabIndex        =   92
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   6
      Left            =   4440
      TabIndex        =   91
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   6
      Left            =   4080
      TabIndex        =   90
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   6
      Left            =   3720
      TabIndex        =   89
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   6
      Left            =   3360
      TabIndex        =   88
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   6
      Left            =   3000
      TabIndex        =   87
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSun
      Height          =   255
      Index           =   6
      Left            =   2640
      TabIndex        =   86
      Top             =   3840
      Width           =   255
   End
```

```
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   6
      Left            =   2040
      TabIndex        =   85
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   6
      Left            =   1440
      TabIndex        =   84
      Top             =   3840
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   5
      Left            =   4800
      TabIndex        =   81
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   5
      Left            =   4440
      TabIndex        =   80
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   5
      Left            =   4080
      TabIndex        =   79
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   5
      Left            =   3720
      TabIndex        =   78
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   5
      Left            =   3360
      TabIndex        =   77
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   5
      Left            =   3000
      TabIndex        =   76
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSun
      Height          =   255
      Index           =   5
      Left            =   2640
      TabIndex        =   75
      Top             =   3480
```

```
      Width           =   255
   End
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   5
      Left            =   2040
      TabIndex        =   74
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   5
      Left            =   1440
      TabIndex        =   73
      Top             =   3480
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   4
      Left            =   4800
      TabIndex        =   70
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   4
      Left            =   4440
      TabIndex        =   69
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   4
      Left            =   4080
      TabIndex        =   68
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   4
      Left            =   3720
      TabIndex        =   67
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   4
      Left            =   3360
      TabIndex        =   66
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   4
      Left            =   3000
      TabIndex        =   65
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSun
      Height          =   255
      Index           =   4
      Left            =   2640
```

```
      TabIndex        =   64
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   4
      Left            =   2040
      TabIndex        =   63
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   4
      Left            =   1440
      TabIndex        =   62
      Top             =   3120
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   3
      Left            =   4800
      TabIndex        =   59
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   3
      Left            =   4440
      TabIndex        =   58
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   3
      Left            =   4080
      TabIndex        =   57
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   3
      Left            =   3720
      TabIndex        =   56
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   3
      Left            =   3360
      TabIndex        =   55
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   3
      Left            =   3000
      TabIndex        =   54
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSun
      Height          =   255
```

```
      Index           =   3
      Left            =   2640
      TabIndex        =   53
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   3
      Left            =   2040
      TabIndex        =   52
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   3
      Left            =   1440
      TabIndex        =   51
      Top             =   2760
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   2
      Left            =   4800
      TabIndex        =   48
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   2
      Left            =   4440
      TabIndex        =   47
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   2
      Left            =   4080
      TabIndex        =   46
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   2
      Left            =   3720
      TabIndex        =   45
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   2
      Left            =   3360
      TabIndex        =   44
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   2
      Left            =   3000
      TabIndex        =   43
      Top             =   2400
      Width           =   255
   End
```

```
   Begin VB.CheckBox cbShiftSun
      Height          =   255
      Index           =   2
      Left            =   2640
      TabIndex        =   42
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   2
      Left            =   2040
      TabIndex        =   41
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   2
      Left            =   1440
      TabIndex        =   40
      Top             =   2400
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   0
      Left            =   1440
      TabIndex        =   26
      ToolTipText     =   "Check to reset the Unit at the start of the shift."
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   0
      Left            =   2040
      TabIndex        =   25
      ToolTipText     =   "Check to export data to CSV at the End of shift."
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSun
      Height          =   255
      Index           =   0
      Left            =   2640
      TabIndex        =   24
      ToolTipText     =   "Sunday"
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   0
      Left            =   3000
      TabIndex        =   23
      ToolTipText     =   "Monday"
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   0
      Left            =   3360
      TabIndex        =   22
      ToolTipText     =   "Tuesday"
      Top             =   1680
      Width           =   255
```

```
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   0
      Left            =   3720
      TabIndex        =   21
      ToolTipText     =   "Wednesday"
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   0
      Left            =   4080
      TabIndex        =   20
      ToolTipText     =   "Thursday"
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   0
      Left            =   4440
      TabIndex        =   19
      ToolTipText     =   "Friday"
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   0
      Left            =   4800
      TabIndex        =   18
      ToolTipText     =   "Saturday"
      Top             =   1680
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSat
      Height          =   255
      Index           =   1
      Left            =   4800
      TabIndex        =   15
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftFri
      Height          =   255
      Index           =   1
      Left            =   4440
      TabIndex        =   14
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftThr
      Height          =   255
      Index           =   1
      Left            =   4080
      TabIndex        =   13
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftWed
      Height          =   255
      Index           =   1
      Left            =   3720
      TabIndex        =   12
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftTue
      Height          =   255
      Index           =   1
```

```
      Left            =   3360
      TabIndex        =   11
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftMon
      Height          =   255
      Index           =   1
      Left            =   3000
      TabIndex        =   10
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftSun
      Height          =   255
      Index           =   1
      Left            =   2640
      TabIndex        =   9
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftExp
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   1
      Left            =   2040
      TabIndex        =   8
      Top             =   2040
      Width           =   255
   End
   Begin VB.CheckBox cbShiftRst
      Alignment       =   1  'Right Justify
      Height          =   255
      Index           =   1
      Left            =   1440
      TabIndex        =   7
      Top             =   2040
      Width           =   255
   End
   Begin VB.CommandButton pb_Apply
      Caption         =   "Apply"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   5280
      TabIndex        =   5
      ToolTipText     =   "Apply the schedule adjustments."
      Top             =   4560
      Width           =   1215
   End
   Begin VB.ComboBox cSchedule
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   1440
      Style           =   2  'Dropdown List
      TabIndex        =   2
      ToolTipText     =   "Select a schedule to adjust."
```

```
      Top             =   120
      Width           =   6375
   End
   Begin VB.TextBox tName
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   18
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   1440
      TabIndex        =   1
      Text            =   "tName - Schedule Name"
      ToolTipText     =   "Enter a name for this schedule."
      Top             =   600
      Width           =   6375
   End
   Begin VB.CommandButton pb_OK
      Caption         =   "OK"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   6600
      TabIndex        =   0
      ToolTipText     =   "Apply the schedule adjustments and close the dialog."
      Top             =   4560
      Width           =   1215
   End
   Begin VB.CommandButton pb_Cancel
      Caption         =   "Cancel"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   3960
      TabIndex        =   6
      ToolTipText     =   "Cancel schedule adjustments."
      Top             =   4560
      Width           =   1215
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   1
      Left            =   5400
      TabIndex        =   16
      ToolTipText     =   " Enter the shift start time here"
      Top             =   2040
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
```

```
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   1
      Left            =   6720
      TabIndex        =   17
      ToolTipText     =   " Enter the shift end time here"
      Top             =   2040
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   0
      Left            =   5400
      TabIndex        =   27
      ToolTipText     =   " Enter the shift start time here"
      Top             =   1680
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   0
      Left            =   6720
      TabIndex        =   28
      ToolTipText     =   " Enter the shift end time here"
      Top             =   1680
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   2
      Left            =   5400
      TabIndex        =   49
      ToolTipText     =   " Enter the shift start time here"
      Top             =   2400
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   2
      Left            =   6720
      TabIndex        =   50
      ToolTipText     =   " Enter the shift end time here"
      Top             =   2400
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
```

```
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   3
      Left            =   5400
      TabIndex        =   60
      ToolTipText     =   " Enter the shift start time here"
      Top             =   2760
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   3
      Left            =   6720
      TabIndex        =   61
      ToolTipText     =   " Enter the shift end time here"
      Top             =   2760
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   4
      Left            =   5400
      TabIndex        =   71
      ToolTipText     =   " Enter the shift start time here"
      Top             =   3120
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   4
      Left            =   6720
      TabIndex        =   72
      ToolTipText     =   " Enter the shift end time here"
      Top             =   3120
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   5
      Left            =   5400
      TabIndex        =   82
      ToolTipText     =   " Enter the shift start time here"
      Top             =   3480
      Width           =   1095
      _ExtentX        =   1931
```

```
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   5
      Left            =   6720
      TabIndex        =   83
      ToolTipText     =   " Enter the shift end time here"
      Top             =   3480
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   6
      Left            =   5400
      TabIndex        =   93
      ToolTipText     =   " Enter the shift start time here"
      Top             =   3840
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   6
      Left            =   6720
      TabIndex        =   94
      ToolTipText     =   " Enter the shift end time here"
      Top             =   3840
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftBeg
      Height          =   255
      Index           =   7
      Left            =   5400
      TabIndex        =   104
      ToolTipText     =   " Enter the shift start time here"
      Top             =   4200
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin MSMask.MaskEdBox meShiftEnd
      Height          =   255
      Index           =   7
      Left            =   6720
      TabIndex        =   105
      ToolTipText     =   " Enter the shift end time here"
```

```
      Top             =   4200
      Width           =   1095
      _ExtentX        =   1931
      _ExtentY        =   450
      _Version        =   393216
      PromptInclude   =   0   'False
      Format          =   "hh:mm:ss AM/PM"
      PromptChar      =   "_"
   End
   Begin VB.Label Label10
      Caption         =   "8th Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   113
      Top             =   4200
      Width           =   855
   End
   Begin VB.Label Label9
      Caption         =   "7th Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   112
      Top             =   3840
      Width           =   855
   End
   Begin VB.Label Label8
      Caption         =   "6th Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   111
      Top             =   3480
      Width           =   855
   End
   Begin VB.Label Label7
      Caption         =   "5th Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   110
      Top             =   3120
      Width           =   855
   End
   Begin VB.Label Label6
      Caption         =   "4th Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   109
      Top             =   2760
      Width           =   855
   End
   Begin VB.Label Label5
      Caption         =   "3rd Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   108
      Top             =   2400
      Width           =   855
   End
   Begin VB.Label Label4
      Caption         =   "2nd Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   107
      Top             =   2040
      Width           =   855
   End
   Begin VB.Label Label3
      Caption         =   "1st Shift:"
      Height          =   255
      Left            =   480
      TabIndex        =   106
      Top             =   1680
      Width           =   855
```

```
   End
   Begin VB.Label lShiftRst
      Alignment       =   2  'Center
      Caption         =   "Reset"
      Height          =   255
      Left            =   1320
      TabIndex        =   39
      Top             =   1320
      Width           =   495
   End
   Begin VB.Label lShiftExp
      Alignment       =   2  'Center
      Caption         =   "Export"
      Height          =   255
      Left            =   1920
      TabIndex        =   38
      Top             =   1320
      Width           =   495
   End
   Begin VB.Label lShiftSun
      Caption         =   "S"
      Height          =   255
      Left            =   2640
      TabIndex        =   37
      Top             =   1320
      Width           =   255
   End
   Begin VB.Label lShiftMon
      Caption         =   "M"
      Height          =   255
      Left            =   3000
      TabIndex        =   36
      Top             =   1320
      Width           =   255
   End
   Begin VB.Label lShiftTue
      Caption         =   "T"
      Height          =   255
      Left            =   3360
      TabIndex        =   35
      Top             =   1320
      Width           =   255
   End
   Begin VB.Label lShiftWed
      Caption         =   "W"
      Height          =   255
      Left            =   3720
      TabIndex        =   34
      Top             =   1320
      Width           =   255
   End
   Begin VB.Label lShiftThr
      Caption         =   "T"
      Height          =   255
      Left            =   4080
      TabIndex        =   33
      Top             =   1320
      Width           =   255
   End
   Begin VB.Label lShiftFri
      Caption         =   "F"
      Height          =   255
      Left            =   4440
      TabIndex        =   32
      Top             =   1320
      Width           =   255
   End
   Begin VB.Label lShfitSat
      Caption         =   "S"
      Height          =   255
      Left            =   4800
      TabIndex        =   31
```

```
      Top             =   1320
      Width           =   255
   End
   Begin VB.Label lShiftBeg
      Alignment       =   2  'Center
      Caption         =   "Start Time"
      Height          =   255
      Left            =   5400
      TabIndex        =   30
      Top             =   1320
      Width           =   1095
   End
   Begin VB.Label lShiftEnd
      Alignment       =   2  'Center
      Caption         =   "End Time"
      Height          =   255
      Left            =   6720
      TabIndex        =   29
      ToolTipText     =   "Enter the shift end time"
      Top             =   1320
      Width           =   1095
   End
   Begin VB.Label Label2
      Caption         =   "Schedule Name:"
      Height          =   255
      Left            =   120
      TabIndex        =   4
      Top             =   600
      Width           =   1215
   End
   Begin VB.Label Label1
      Caption         =   "Schedule:"
      Height          =   255
      Left            =   120
      TabIndex        =   3
      Top             =   120
      Width           =   1215
   End
End
```

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
Option Explicit

'Declare Trimline Configuration Type
Private Type TLUCONFIGTYPE
    Name As String          'Trimline Symbolic Name
    Port As Integer         'Comport 1 to 16 (0 = disabled)
    Address As Integer      'Address character 32 to 126
    HostName As String      'TCP/IP Host Name
    Schedule As Integer     'Schedule Selection 1 to 64
End Type

'Local Config Storage
Dim Config(1 To NUMTLU) As TLUCONFIGTYPE


'--------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------
Private Sub Form_Load()
    Dim i As Integer

    'Restore form to proper position and size
    Call LoadWindowPosition(Me)

    'make a local copy of the Trimline configuation data
    For i = LBound(Config) To UBound(Config)
        Config(i).Name = Tlu(i).Name
        Config(i).Port = Tlu(i).Port
        Config(i).Address = Tlu(i).Address
        Config(i).HostName = Tlu(i).HostName
        Config(i).Schedule = Tlu(i).Schedule
    Next i

    'Program the Trimline Selection Combo
    For i = LBound(Config) To UBound(Config)
        cTlu.AddItem "#" & Format(i, "#) ") & Config(i).Name
    Next i

    'Program the Port Selection Combo
```

```
    For i = 0 To 16
        cPort.AddItem fmtPort(i)
    Next i

    'Program the Address Selection Combo
    cAddress.AddItem fmtAddr(0)
    cAddress.ItemData(cAddress.NewIndex) = 0
    For i = 33 To 126
        cAddress.AddItem fmtAddr(i)
        cAddress.ItemData(cAddress.NewIndex) = i
    Next i

    'Program the Schedule Selection Combo
    For i = 0 To UBound(Schedule)
        cSchedule.AddItem fmtSchedule(i)
    Next i

    'Select the 1st Trimline
    cTlu.ListIndex = 0

End Sub


Private Sub Form_Unload(Cancel As Integer)

    'Save form position and size
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------
' The User Selected a Trimline, Load the Form
'--------------------------------------------------------------------------
Private Sub cTlu_Click()
    Dim Unit As Integer

    'Identify the Selected Trimline
    Unit = cTlu.ListIndex + 1

    'Populate the form with the selected Trimline Config
    tName = Config(Unit).Name
    cPort.ListIndex = Config(Unit).Port
    If Config(Unit).Address > 32 And Config(Unit).Address < 127 Then
        cAddress.ListIndex = Config(Unit).Address - 32
    Else
        cAddress.ListIndex = 0
    End If
    tHostName = Config(Unit).HostName
    cSchedule.ListIndex = Config(Unit).Schedule

End Sub



'--------------------------------------------------------------------------
' As the User Edits the Trimline, Keep the Config() Array in Sync
'--------------------------------------------------------------------------
'User Edited the Trimline Name
Private Sub tName_Change()
    Config(cTlu.ListIndex + 1).Name = tName
    cTlu.List(cTlu.ListIndex) = "#" & Format(cTlu.ListIndex + 1, "0) ") & Config(cTlu.ListIndex + 1).
Name
End Sub

'User Editied the Port Selection
Private Sub cPort_Click()
    Config(cTlu.ListIndex + 1).Port = cPort.ListIndex
    If (cPort.ListIndex = 16) Then
        lAddress.Visible = False
        cAddress.Visible = False
        lHostName.Visible = True
```

```
        tHostName.Visible = True
    Else
        lAddress.Visible = True
        cAddress.Visible = True
        lHostName.Visible = False
        tHostName.Visible = False
    End If
End Sub

'User Edited the Address Selection
Private Sub cAddress_Click()
    Config(cTlu.ListIndex + 1).Address = cAddress.ItemData(cAddress.ListIndex)
End Sub

'User Edited the HostName
Private Sub thostName_Change()
    Config(cTlu.ListIndex + 1).HostName = tHostName
End Sub

'User Edited the Schedule Selection
Private Sub cSchedule_Click()
    Config(cTlu.ListIndex + 1).Schedule = cSchedule.ListIndex
End Sub


'------------------------------------------------------------------------
' OK and Cancel Buttons
'------------------------------------------------------------------------
Private Sub bOK_Click()

    'Password Protected Operation
    If (CheckPassword(Me) = True) Then

        'Apply Changes and then go away
        Call ApplyChanges
        Unload Me

    End If

End Sub


Private Sub bApply_Click()

    'Password Protected Operation
    If (CheckPassword(Me) = True) Then

        'Apply Changes
        Call ApplyChanges

    End If

End Sub


Private Sub bCancel_Click()

    'User selected cancel
    Unload Me

End Sub

Public Sub ApplyChanges()
    Dim i As Integer

    For i = LBound(Tlu) To UBound(Tlu)

        'Don't allow blank names
        If Len(Config(i).Name) = 0 Then
            Config(i).Name = "UNIT" + Str(i) 'default name
        End If
```

```
        'if either port or address are "(none)" force defaults
        If (Config(i).Port = 0 Or (Config(i).Address = 0 And Config(i).Port <> 16)) Then

            'Mark this Trimline as **UNUSED**
            Tlu(i).Name = "UNIT" + Str(i)
            Tlu(i).Port = 0
            Tlu(i).Address = 0
            Tlu(i).HostName = "(none)"
            Tlu(i).Schedule = 0

        Else

            'update the Tlu() config structure
            Tlu(i).Name = Config(i).Name
            Tlu(i).Port = Config(i).Port
            Tlu(i).Address = Config(i).Address
            Tlu(i).HostName = Config(i).HostName
            Tlu(i).Schedule = Config(i).Schedule

        End If


    Next i

    'Now update mass storage with the new values
    Call SaveAllConfiguration

End Sub
```

```
frmConfigTlus - 1

VERSION 5.00
Begin VB.Form frmConfigTlus
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Configure Counters"
   ClientHeight    =   3375
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   8025
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   3375
   ScaleWidth      =   8025
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox tHostName
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   1560
      TabIndex        =   4
      Text            =   "tHostName"
      ToolTipText     =   "Enter the host name or IP address of this PPT."
      Top             =   1800
      Width           =   6375
   End
   Begin VB.CommandButton bCancel
      Caption         =   "Cancel"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   3720
      TabIndex        =   9
      ToolTipText     =   "Cancel adjustments to the PPT configuration."
      Top             =   2760
      Width           =   1335
   End
   Begin VB.ComboBox cTlu
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   1560
      Style           =   2  'Dropdown List
      TabIndex        =   0
      ToolTipText     =   "Select a PPT to adjust."
      Top             =   120
      Width           =   6375
   End
   Begin VB.ComboBox cSchedule
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   9.75
```

```
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   360
   Left            =   1560
   Style           =   2  'Dropdown List
   TabIndex        =   5
   ToolTipText     =   "Select a schedule for this PPT."
   Top             =   2280
   Width           =   6375
End
Begin VB.CommandButton bApply
   Caption         =   "Apply"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   13.5
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   495
   Left            =   5160
   TabIndex        =   8
   ToolTipText     =   "Apply adjustments to the PPT configuration."
   Top             =   2760
   Width           =   1335
End
Begin VB.CommandButton bOK
   Caption         =   "OK"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   13.5
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   495
   Left            =   6600
   TabIndex        =   6
   ToolTipText     =   "Apply adjustments to the PPT configuration and close this dialog."
   Top             =   2760
   Width           =   1335
End
Begin VB.ComboBox cAddress
   BeginProperty Font
      Name            =   "Courier New"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   360
   Left            =   1560
   Style           =   2  'Dropdown List
   TabIndex        =   3
   ToolTipText     =   "Select the address of this PPT."
   Top             =   1800
   Width           =   6375
End
Begin VB.ComboBox cPort
   BeginProperty Font
      Name            =   "Courier New"
      Size            =   9.75
```

```
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   360
   Left            =   1560
   Style           =   2  'Dropdown List
   TabIndex        =   2
   ToolTipText     =   "Select the port used to communicate with this PPT."
   Top             =   1320
   Width           =   6375
End
Begin VB.TextBox tName
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   18
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   525
   Left            =   1560
   TabIndex        =   1
   Text            =   "tName - Unit Name"
   ToolTipText     =   "Enter a name for this PPT."
   Top             =   600
   Width           =   6375
End
Begin VB.Label lHostName
   Caption         =   "Host Name:"
   Height          =   255
   Left            =   120
   TabIndex        =   14
   Top             =   1800
   Width           =   1215
End
Begin VB.Label Label5
   Caption         =   "Schedule:"
   Height          =   255
   Left            =   120
   TabIndex        =   13
   Top             =   2280
   Width           =   1215
End
Begin VB.Label lAddress
   Caption         =   "Address:"
   Height          =   255
   Left            =   120
   TabIndex        =   12
   Top             =   1800
   Width           =   1215
End
Begin VB.Label lPort
   Caption         =   " Port:"
   Height          =   255
   Left            =   120
   TabIndex        =   11
   Top             =   1320
   Width           =   1215
End
Begin VB.Label lPPTName
   Caption         =   "Name:"
   Height          =   255
   Left            =   120
   TabIndex        =   10
   Top             =   720
   Width           =   1215
End
```

```
   Begin VB.Label lTlu
      Caption         =   "Counter:"
      Height          =   255
      Left            =   120
      TabIndex        =   7
      Top             =   120
      Width           =   1215
   End
End
```

```
'--------------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------------
Option Explicit

'Declare Totalizer Configuration Type
Private Type TOTALIZERCONFIGTYPE
    Name As String                          'Symbolic Name of this Totalizer
    Port As Integer                         'Comport 1 to 16 (0 = disabled)
    HostName As String                      'TCP/IP Host Name
    ActualAddress As Integer                'Address Character 32 to 126
    TluIsMember(1 To NUMTLU) As Integer     'Is this Trimline a used in the Calculation
End Type

'Local Config Storage
Dim Config(1 To NUMTOTALIZER) As TOTALIZERCONFIGTYPE


'--------------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------------
Private Sub Form_Load()
    Dim Unit As Integer
    Dim i As Integer

    'Restore form to proper position and size
    Call LoadWindowPosition(Me)

    'make a local copy of the totalizer display configuation data
    For Unit = LBound(Totalizer) To UBound(Totalizer)
        Config(Unit).Name = Totalizer(Unit).Name
        Config(Unit).Port = Totalizer(Unit).Port
        Config(Unit).HostName = Totalizer(Unit).HostName
        Config(Unit).ActualAddress = Totalizer(Unit).ActualAddress
        For i = LBound(Tlu) To UBound(Tlu)
            Config(Unit).TluIsMember(i) = Totalizer(Unit).TluIsMember(i)
        Next i
    Next Unit

    'Program the Totalizer Selection Combo
    For i = LBound(Config) To UBound(Config)
        cTotalizer.AddItem "#" & Format(i, "#) ") & Config(i).Name
```

```
    Next i

    'Program the Port Selection Combo
    For i = 0 To 16
        cPort.AddItem fmtPort(i)
    Next i

    'Program the Address Selection Combos
    cActualAddress.AddItem fmtAddr(0)
    cActualAddress.ItemData(cActualAddress.NewIndex) = 0
    For i = 33 To 126
        cActualAddress.AddItem fmtAddr(i)
        cActualAddress.ItemData(cActualAddress.NewIndex) = i
    Next i

    'Load the Trimline Selection List
    For i = LBound(Tlu) To UBound(Tlu)
        lTlu.AddItem Tlu(i).Name
    Next i

    'Select the 1st Totalizer Display
    cTotalizer.ListIndex = 0

End Sub


Private Sub Form_Unload(Cancel As Integer)

    'Save form position and size
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------
' The User Selected a Totalizer, Load the Form
'--------------------------------------------------------------------------
Private Sub cTotalizer_Click()
    Dim Unit As Integer
    Dim i As Integer

    'Identify the Selected Totalizer
    Unit = cTotalizer.ListIndex + 1

    'Populate the Totalizer Name
    tName = Config(Unit).Name

    'Populate the Totalizer Port Selection
    cPort.ListIndex = Config(Unit).Port

    'Populate the Totalizer Host Name
    tHostName = Config(Unit).HostName

    'Populate the Totalizer Actual Address Selection
    If Config(Unit).ActualAddress > 32 And Config(Unit).ActualAddress < 127 Then
        cActualAddress.ListIndex = Config(Unit).ActualAddress - 32
    Else
        cActualAddress.ListIndex = 0
    End If

    'Populate the Trimline List
    Call LoadTluList

End Sub


'--------------------------------------------------------------------------
' As the User Edits the Totalizer, Keep the Config() Array in Sync
'--------------------------------------------------------------------------
'User Edited the Totalizer Name
Private Sub tName_Change()
    Config(cTotalizer.ListIndex + 1).Name = tName
```

```
    cTotalizer.List(cTotalizer.ListIndex) = "#" & Format(cTotalizer.ListIndex + 1, "0) ") & Config(cT
otalizer.ListIndex + 1).Name
End Sub

'User Editied the Port Selection
Private Sub cPort_Click()
    Config(cTotalizer.ListIndex + 1).Port = cPort.ListIndex
    If (cPort.ListIndex = 16) Then
        lHostName.Enabled = True
        tHostName.Enabled = True
    Else
        lHostName.Enabled = False
        tHostName.Enabled = False
    End If
End Sub

'User Edited the HostName
Private Sub thostName_Change()
    Config(cTotalizer.ListIndex + 1).HostName = tHostName
End Sub

'User Edited the Actual Address Selection
Private Sub cActualAddress_Click()
    Config(cTotalizer.ListIndex + 1).ActualAddress = cActualAddress.ItemData(cActualAddress.ListIndex
)
End Sub

'User Editied the Member Trimline Selection
Private Sub lTlu_Click()
    Dim Unit As Integer
    Dim i As Integer

    Unit = cTotalizer.ListIndex + 1
    i = lTlu.ListIndex + 1
    Config(Unit).TluIsMember(i) = lTlu.Selected(i - 1)

End Sub

'Load the Trimline List for the Current Unit
Public Sub LoadTluList()
    Dim Unit As Integer
    Dim i As Integer

    Unit = cTotalizer.ListIndex + 1
    lTlu.Clear
    For i = LBound(Config(Unit).TluIsMember) To UBound(Config(Unit).TluIsMember)
        Call lTlu.AddItem("#" & i & ")" & Tlu(i).Name)
        lTlu.Selected(i - 1) = Config(Unit).TluIsMember(i)
    Next i

End Sub


'--------------------------------------------------------------------------
' OK, Apply, and Cancel Buttons
'--------------------------------------------------------------------------
Private Sub bOK_Click()

    'Password Protected Operation
    If (CheckPassword(Me) = True) Then

        Call ApplyChanges
        Unload Me

    End If

End Sub


Private Sub bApply_Click()

    'Password Protected Operation
```

```
    If (CheckPassword(Me) = True) Then

        Call ApplyChanges

    End If

End Sub


Private Sub bCancel_Click()

    'User selected cancel
    Unload Me

End Sub

Public Sub ApplyChanges()
    Dim Unit As Integer
    Dim i As Integer

    For Unit = LBound(Totalizer) To UBound(Totalizer)

        'Don't allow blank names
        If Len(Config(Unit).Name) = 0 Then
            Config(Unit).Name = "Totalizer" + Str(Unit) 'default name
        End If

        'if port is "(none)" force defaults
        If (Config(Unit).Port = 0) Then

            'Mark this Totalizer as **UNUSED**
            Totalizer(Unit).Name = "Totalizer" + Str(Unit)
            Totalizer(Unit).Port = 0
            Totalizer(Unit).HostName = "(none)"
            Totalizer(Unit).ActualAddress = 0
            Totalizer(Unit).Status = "(none)"
            For i = LBound(Totalizer(Unit).TluIsMember) To UBound(Totalizer(Unit).TluIsMember)
                Totalizer(Unit).TluIsMember(i) = 0
            Next i
        Else

            'update the Totalizer() config structure
            Totalizer(Unit).Name = Config(Unit).Name
            Totalizer(Unit).Port = Config(Unit).Port
            Totalizer(Unit).HostName = Config(Unit).HostName
            Totalizer(Unit).ActualAddress = Config(Unit).ActualAddress
            For i = LBound(Totalizer(Unit).TluIsMember) To UBound(Totalizer(Unit).TluIsMember)
                Totalizer(Unit).TluIsMember(i) = Config(Unit).TluIsMember(i)
            Next i

        End If

    Next Unit

    'Now update mass storage with the new values
    Call SaveAllConfiguration

End Sub
```

```
frmConfigTotalizers - 1

VERSION 5.00
Begin VB.Form frmConfigTotalizers
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Configure Totalizers"
   ClientHeight    =   5520
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   8295
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   5520
   ScaleWidth      =   8295
   StartUpPosition =   3  'Windows Default
   Begin VB.ComboBox cActualAddress
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   1800
      Style           =   2  'Dropdown List
      TabIndex        =   4
      ToolTipText     =   "Select the address of the total actual display field."
      Top             =   2280
      Width           =   6375
   End
   Begin VB.ComboBox cTotalizer
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   1800
      Style           =   2  'Dropdown List
      TabIndex        =   0
      ToolTipText     =   "Select a Totalizer to adjust."
      Top             =   120
      Width           =   6375
   End
   Begin VB.TextBox tHostName
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   9.75
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   1800
      TabIndex        =   3
      Text            =   "tHostName"
      ToolTipText     =   "Enter the host name or IP address of this totalizer display."
      Top             =   1800
      Width           =   6375
   End
   Begin VB.CommandButton bApply
      Caption         =   "Apply"
      BeginProperty Font
         Name            =   "MS Sans Serif"
```

```
      Size            =   13.5
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   495
   Left            =   5400
   TabIndex        =   8
   ToolTipText     =   "Apply adjustments to totalizer configuration."
   Top             =   4920
   Width           =   1335
End
Begin VB.ListBox lTlu
   Columns         =   2
   BeginProperty Font
      Name            =   "Courier New"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   1950
   Left            =   1800
   Style           =   1  'Checkbox
   TabIndex        =   5
   ToolTipText     =   "Select which PPTs are included in the total calculations."
   Top             =   2760
   Width           =   6375
End
Begin VB.CommandButton bCancel
   Caption         =   "Cancel"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   13.5
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   495
   Left            =   3960
   TabIndex        =   9
   ToolTipText     =   "Cancel adjustments to totalizer configuration."
   Top             =   4920
   Width           =   1335
End
Begin VB.CommandButton bOK
   Caption         =   "OK"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   13.5
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   495
   Left            =   6840
   TabIndex        =   7
   ToolTipText     =   "Apply adjustments to totalizer configuration and close the dialog."
   Top             =   4920
   Width           =   1335
End
Begin VB.ComboBox cPort
   BeginProperty Font
      Name            =   "Courier New"
```

```
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   360
   Left            =   1800
   Style           =   2  'Dropdown List
   TabIndex        =   2
   ToolTipText     =   "Select the port that is used to communicate with this totalizer display."
   Top             =   1320
   Width           =   6375
End
Begin VB.TextBox tName
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   18
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   525
   Left            =   1800
   TabIndex        =   1
   Text            =   "tName - PPT Name"
   ToolTipText     =   "Enter a name for this totalizer display."
   Top             =   600
   Width           =   6375
End
Begin VB.Label lActualAddress
   Caption         =   "Actual Address:"
   Height          =   255
   Left            =   120
   TabIndex        =   15
   Top             =   2280
   Width           =   1455
End
Begin VB.Label lHostName
   Caption         =   "Host Name:"
   Height          =   255
   Left            =   120
   TabIndex        =   14
   Top             =   1800
   Width           =   1455
End
Begin VB.Label lTotalizerFor
   Caption         =   "Totalizer for:"
   Height          =   255
   Left            =   120
   TabIndex        =   13
   Top             =   2760
   Width           =   1455
End
Begin VB.Label lAddress
   Caption         =   "Address:"
   Height          =   255
   Left            =   120
   TabIndex        =   12
   Top             =   1800
   Width           =   1215
End
Begin VB.Label lPort
   Caption         =   "Port:"
   Height          =   255
   Left            =   120
   TabIndex        =   11
   Top             =   1320
   Width           =   1455
```

```
   End
   Begin VB.Label lName
      Caption         =   "Name:"
      Height          =   255
      Left            =   120
      TabIndex        =   10
      Top             =   720
      Width           =   1455
   End
   Begin VB.Label lTotalizer
      Caption         =   "Totalizer Display:"
      Height          =   255
      Left            =   120
      TabIndex        =   6
      Top             =   120
      Width           =   1455
   End
End
```

```
'--------------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------------
Private Sub Form_Load()

    'Restore form to proper position
    Call LoadWindowPosition(Me)

End Sub


Private Sub Form_Activate()

    'We were just activated, force an update
    Call UpdateDetailData

End Sub


Private Sub Form_Unload(Cancel As Integer)

    'Save form position
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------------
' OK and Cancel Buttons...
'--------------------------------------------------------------------------------
Private Sub bAccept_Click()

    Dim Unit As Integer
    Dim Cmd As String

    'Password Protected Operation
```

```
    If (CheckPassword(Me) = True) Then

        'Write Actual
        If InStr(Me.Tag, "AC") <> 0 Then
            For Unit = LBound(Tlu) To UBound(Tlu)
                If Tlu(Unit).Port <> 0 Then
                    Cmd = "WAC" & RegVal6Digit(Me.tNewVal.Text)
                    Call Tlu(Unit).CmdQueue.Enqueue(Cmd)
                End If
            Next Unit
        End If

        'Go Away
        Unload Me

    End If

End Sub

Private Sub bCancel_Click()

    'User Canceled the update
    'go away
    Unload Me

End Sub


'------------------------------------------------------------------------
' Update the Form with current data
'------------------------------------------------------------------------
Private Sub UpdateDetailData()

    'Updates for Change Actual Dialog
    If InStr(Me.Tag, "AC") <> 0 Then
        Me.Caption = "Change All Actual Registers"
        Me.lTitle = Me.Caption
        Me.lNew = "New Actual:"
        Me.tNewVal.MaxLength = 6
    End If

End Sub
```

```
frmGWriteReg - 1

VERSION 5.00
Begin VB.Form frmGWriteReg
   BorderStyle     =   1  'Fixed Single
   Caption         =   "frmGWriteReg"
   ClientHeight    =   1920
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   6375
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   1920
   ScaleWidth      =   6375
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton bAccept
      Caption         =   "&OK"
      Default         =   -1  'True
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   3240
      TabIndex        =   3
      ToolTipText     =   "Click here to accept the change."
      Top             =   1320
      Width           =   3015
   End
   Begin VB.CommandButton bCancel
      Cancel          =   -1  'True
      Caption         =   "&Cancel"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   120
      TabIndex        =   2
      ToolTipText     =   "Click here to cancel change."
      Top             =   1320
      Width           =   3015
   End
   Begin VB.TextBox tNewVal
      Alignment       =   2  'Center
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   14.25
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   3240
      TabIndex        =   1
      ToolTipText     =   "Enter the new register value here."
      Top             =   720
      Width           =   3015
   End
   Begin VB.Label lNew
      Alignment       =   1  'Right Justify
```

```
      Caption         =   "New Reg:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   0
      TabIndex        =   4
      Top             =   720
      Width           =   2895
   End
   Begin VB.Label lTitle
      Alignment       =   2  'Center
      Caption         =   "Change All Register Registers to"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   6135
   End
End
```

frmMain - 1

```
'--------------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------------
Private Sub Form_Load()

    'Restore form to proper position and size
    Call LoadWindowPosition(Me)

    'Initialize the grid display style
    Grid1.HighLight = flexHighlightNever            'Turn off cell highlighting
    Grid1.ScrollTrack = True                        'Enable ScrollTracking
    Grid1.Cols = 3                                  'Set Table Width
    Grid1.Rows = 65                                 'Set Table Height
    Grid1.ColWidth(0) = (Grid1.ColWidth(0) * 3)     'Make the Name Column Tripple Width
    Grid1.ColWidth(1) = (Grid1.ColWidth(1) * 3)     'Make the Name Column Tripple Width
    Grid1.ColWidth(2) = (Grid1.ColWidth(2) * 3)     'Make the Status Column Triple Width
    Grid1.FillStyle = flexFillRepeat                'settings apply to table selections
    Grid1.Col = 0: Grid1.Row = 0                    'select top left cell
    Grid1.RowSel = Grid1.Rows - 1                   'select all of the first column
    Grid1.ColSel = Grid1.Cols - 1                   'select the entire table
    Grid1.CellAlignment = flexAlignCenterCenter     'Center Text in the Cell
    Grid1.Col = 0: Grid1.Row = 0                    'select top left cell
    Grid1.ColSel = Grid1.Cols - 1                   'select the top row
    Grid1.CellFontBold = True                       'set text to BOLD

    'Set the Grid Column Labels Text
    Grid1.TextArray(0) = "Unit Name"
    Grid1.TextArray(1) = "Actual Count"
    Grid1.TextArray(2) = "Unit Status"

    'put the curssor in the upper left corner
    Grid1.Col = 1
    Grid1.Row = 1

    'put a helpfull note in the status bar
    sbStatusBar.Panels(1).Text = "Click on a unit to bring up a detailed view."
```

```
frmMain - 2


End Sub


Private Sub Form_Resize()
    Dim Temp As Long

    'auto resize the flexgrid to fill the form at run time
    'leave room for the Status Bar at the bottom of the form
    Grid1.Top = 0
    Grid1.Left = 0
    Grid1.Width = frmMain.ScaleWidth
    Temp = frmMain.ScaleHeight - sbStatusBar.Height
    If Temp < 1 Then Temp = 1
    Grid1.Height = Temp

End Sub


Private Sub Form_Unload(Cancel As Integer)
    Dim i As Integer

    'Log Program Shutdown
    Call LogEvent("frmMain_Unload()", "Program Shutting Down")

    'Tell BackgroundFunction to pack it in
    OpState = False

    'close all sub forms
    For i = Forms.Count - 1 To 1 Step -1
        Unload Forms(i)
    Next

    'Save frmMain position and size
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------
' Pull Down Menus
'--------------------------------------------------------------------------
Private Sub mnuFileExport_Click()
    Dim File As String
    Dim DefaultCSV As String
    Dim Unit As Integer

    'Compute Ini File Name
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\SETTINGS.INI"

    'Compute Default CSV File Name
    DefaultCSV = GetCommonDocumentsFolder() & "\Counter.csv"

    'open the windows common 'Save As' dialog
    On Error GoTo errhandler
    CommonDialog1.CancelError = True
    CommonDialog1.Filter = "All Files (*.*)|*.*|Comma Seperated Values File (*.csv)|*.csv"
    CommonDialog1.FilterIndex = 2
    CommonDialog1.FileName = ReadIni(File, "FileNameCSV", "FileNameCSV", DefaultCSV)
    CommonDialog1.ShowSave

    'User clicked OK, Remember where the user wants this file saved
    Call WriteIni(File, "FileNameCSV", "FileNameCSV", CommonDialog1.FileName)

    'Export the Data
    For Unit = LBound(Tlu) To UBound(Tlu)
        If (Tlu(Unit).Port <> 0) Then
            Call CsvExport(Tlu(Unit), CommonDialog1.FileName, Now)
        End If
    Next Unit
```

```
    Exit Sub

errhandler:
    'user clicked cancel
    Beep
    Exit Sub

End Sub


Private Sub mFileViewTotalizers_Click()

    'Display the Totalizers Grid
    frmTotalizers.Show

End Sub


Private Sub mnuFileConfigSchedules_Click()

    'bring up the Configure Schedules Dialog
    frmConfigSchedules.Show

End Sub


Private Sub mnuFileConfigTlus_Click()

    'bring up the Configure Counters Dialog
    frmConfigTlus.Show

End Sub


Private Sub mnuFileConfigTotalizers_Click()

    'bring up the Configure Totalizer Displays Dialog
    frmConfigTotalizers.Show

End Sub


Private Sub mnuChangePassword_Click()

    'bring up the Change Password Dialog
    frmChangePassword.Show

End Sub


Private Sub mnuFileExit_Click()

    'Make the program terminate
    Unload Me

End Sub


Private Sub mnuHelpAbout_Click()

    'bring up the About dialog
    frmAbout.Show

End Sub


Private Sub mnuDebugEventLog_Click()

    'Toggle Debug Event Logging ON/OFF
    If (DebugFlag = False) Then
        mnuDebugEventLog.Checked = True
        DebugFlag = True
```

```
        Call SaveDebugFlag
        Call LogEvent("DebugLog()", "Debug Log Enabled")
        Call EnqueueMessage("Debug Event Logging Enabled!", "Logging to file DebugLog.txt", "")
    Else
        Call EnqueueMessage("Debug Event Logging Disabled!", "", "")
        Call LogEvent("DebugLog()", "Debug Log Disabled")
        mnuDebugEventLog.Checked = False
        DebugFlag = False
        Call SaveDebugFlag
    End If

End Sub


'--------------------------------------------------------------------------
' Pace Timer Data MSFlexGrid
'--------------------------------------------------------------------------
Private Sub Grid1_Click()
    Dim Unit As Integer
    Dim Register As Integer

    'Get the Row and Column clicked on
    Unit = Grid1.MouseRow
    Register = Grid1.MouseCol

    If (Unit >= 1) Then

        'Use selected a Trimline Detailed View, bring it up
        frmTluDetail.Tag = Unit
        frmTluDetail.Show

    Else

        'User selected a Register Column Heading
        'Bring up the Global Register Write Dialog
        Select Case Register

            Case 1  'Actual Register
                frmGWriteReg.Tag = "AC"
                frmGWriteReg.Show

        End Select

    End If

End Sub


Private Sub Grid1_KeyPress(KeyAscii As Integer)

    'filter keypresses in the grid
    If KeyAscii = Asc(vbCr) Then
        KeyAscii = 0
        Call Grid1_Click
    End If

End Sub


'--------------------------------------------------------------------------
' Status Bar at Bottom of Window
'--------------------------------------------------------------------------
Private Sub sbStatusBar_PanelClick(ByVal Panel As ComctlLib.Panel)

    'Toggle the RXBuf HexDump on and off
    If Val(Me.sbStatusBar.Tag) <> 1 Then
        Me.sbStatusBar.Tag = 1
    Else
        Me.sbStatusBar.Tag = 0
        sbStatusBar.Panels(1).Text = "Click on a unit to bing up a detailed view."
    End If
```

```
frmMain - 5

End Sub

'------------------------------------------------------------------------
' Periodically Check for Messages sent by the BackgroundFunction()
' That need to be shown to the system operator
'------------------------------------------------------------------------
Private Sub Timer1_Timer()

    'If there are any messages in the Message Queue show them to the user
    If (MsgQueue.Count > 0) Then

        'Summon a new messge box to display the message
        Dim frmM As New frmMessage
        frmM.lMessage = MsgQueue.Dequeue
        Beep
        frmM.Show

    End If

End Sub


'------------------------------------------------------------------------
' This function is called by BackgroundFunction() when updated
' Tlu(Unit) Data is available, and the Screen needs to be redrawn
'------------------------------------------------------------------------
Public Sub UpdateData(Unit As Integer)
    Dim RowLen As Integer
    Dim RowStart As Integer
    Dim Text As String

    'Get the number of cells per row
    'for calculating the textarray index
    RowLen = Grid1.Cols

    'Calculate the TextArray Index for the
    'Start of this row
    RowStart = Unit * RowLen

    'update this grid row
    With Tlu(Unit)
        Grid1.TextArray(RowStart + 0) = .Name
        Grid1.TextArray(RowStart + 1) = fmtNumber(.Actual.Value)
        Grid1.TextArray(RowStart + 2) = .Status
    End With

End Sub
```

```
frmMain - 1

VERSION 5.00
Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0"; "COMCTL32.OCX"
Object = "{5E9E78A0-531B-11CF-91F6-C2863C385E30}#1.0#0"; "MSFLXGRD.OCX"
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "comdlg32.ocx"
Object = "{248DD890-BB45-11CF-9ABC-0080C7E7B78D}#1.0#0"; "mswinsck.ocx"
Begin VB.Form frmMain
   Caption         =   "Trimline Monitor (PD-2116-058)"
   ClientHeight    =   3180
   ClientLeft      =   165
   ClientTop       =   855
   ClientWidth     =   11880
   LinkTopic       =   "Form1"
   ScaleHeight     =   3180
   ScaleWidth      =   11880
   StartUpPosition =   3  'Windows Default
   Begin MSWinsockLib.Winsock Winsock2
      Index           =   0
      Left            =   3960
      Top             =   720
      _ExtentX        =   741
      _ExtentY        =   741
      _Version        =   393216
   End
   Begin MSWinsockLib.Winsock Winsock1
      Index           =   0
      Left            =   3960
      Top             =   240
      _ExtentX        =   741
      _ExtentY        =   741
      _Version        =   393216
   End
   Begin VB.Timer Timer1
      Interval        =   250
      Left            =   3240
      Top             =   1560
   End
   Begin MSComDlg.CommonDialog CommonDialog1
      Left            =   3240
      Top             =   840
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   393216
   End
   Begin MSCommLib.MSComm MSComm1
      Left            =   3240
      Top             =   120
      _ExtentX        =   1005
      _ExtentY        =   1005
      _Version        =   393216
      DTREnable       =   -1  'True
   End
   Begin ComctlLib.StatusBar sbStatusBar
      Align           =   2  'Align Bottom
      Height          =   285
      Left            =   0
      TabIndex        =   1
      Top             =   2895
      Width           =   11880
      _ExtentX        =   20955
      _ExtentY        =   503
      SimpleText      =   ""
      _Version        =   327682
      BeginProperty Panels {0713E89E-850A-101B-AFC0-4210102A8DA7}
         NumPanels       =   2
         BeginProperty Panel1 {0713E89F-850A-101B-AFC0-4210102A8DA7}
            AutoSize        =   1
            Object.Width           =   17859
            TextSave        =   ""
            Key             =   ""
            Object.Tag             =   ""
         EndProperty
```

```
frmMain - 2

      BeginProperty Panel2 {0713E89F-850A-101B-AFC0-4210102A8DA7}
         Style          =   5
         TextSave       =   "11:16 AM"
         Key            =   ""
         Object.Tag             =   ""
      EndProperty
   EndProperty
   BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
      Name            =   "Courier New"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
End
Begin MSFlexGridLib.MSFlexGrid Grid1
   Height          =   2415
   Left            =   120
   TabIndex        =   0
   ToolTipText     =   "Click on a unit to bring up a detailed view. Click on a register name to cha
   Top             =   120
   Width           =   3015
   _ExtentX        =   5318
   _ExtentY        =   4260
   _Version        =   393216
End
Begin VB.Menu mnuFile
   Caption         =   "&File"
   Begin VB.Menu mnuFileExport
      Caption         =   "&Export Counter Data to CSV File..."
   End
   Begin VB.Menu mFileViewTotalizers
      Caption         =   "View Totalizers..."
   End
   Begin VB.Menu mSep1
      Caption         =   "-"
   End
   Begin VB.Menu mnuFileConfigSchedules
      Caption         =   "Configure &Schedules..."
   End
   Begin VB.Menu mnuFileConfigTlus
      Caption         =   "Configure &Counters..."
   End
   Begin VB.Menu mnuFileConfigTotalizers
      Caption         =   "Configure &Totalizers..."
   End
   Begin VB.Menu mnuChangePassword
      Caption         =   "Change Password..."
   End
   Begin VB.Menu mSep2
      Caption         =   "-"
   End
   Begin VB.Menu mnuFileExit
      Caption         =   "E&xit"
   End
End
Begin VB.Menu mnuHelp
   Caption         =   "&Help"
   Begin VB.Menu mnuHelpAbout
      Caption         =   "&About PPT Monitor..."
   End
   Begin VB.Menu mnuDebugEventLog
      Caption         =   "Debug Event Log"
   End
End
End
```

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------
Private Sub Form_Activate()

    'Center Message over Main Form
    Call CenterWindow(Me)

End Sub


'--------------------------------------------------------------------------
' OK Button...
'--------------------------------------------------------------------------
Private Sub pb_OK_Click()

    Unload Me

End Sub
```

frmMessage - 1

```
VERSION 5.00
Begin VB.Form frmMessage
   BorderStyle     =   1  'Fixed Single
   Caption         =   "PPT Monitor"
   ClientHeight    =   1905
   ClientLeft      =   45
   ClientTop       =   435
   ClientWidth     =   7095
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   1905
   ScaleWidth      =   7095
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton pb_OK
      Caption         =   "OK"
      Height          =   495
      Left            =   5760
      TabIndex        =   1
      ToolTipText     =   "Click here to clear the message."
      Top             =   1320
      Width           =   1215
   End
   Begin VB.Label lMessage
      Alignment       =   2  'Center
      BorderStyle     =   1  'Fixed Single
      Caption         =   "lMessage"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   1140
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   6885
   End
End
```

frmPassword - 1

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------
Private Sub Form_Activate()

    'Erase the password
    tbPassword.Text = ""
    tbPassword.SetFocus

    'Center Dialog over Main Form
    Call CenterWindow(Me)

End Sub


'--------------------------------------------------------------------------
' OK and Cancel Buttons
'--------------------------------------------------------------------------
Private Sub bOK_Click()

    'go away, the caller will check the password
    Me.Hide

End Sub


Private Sub bCancel_Click()

    'Erase the password and exit
    tbPassword.Text = ""
    Me.Hide

End Sub
```

frmPassword - 1

```
VERSION 5.00
Begin VB.Form frmPassword
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Password Required for Privileged Operation"
   ClientHeight    =   1815
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   5280
   ControlBox      =   0   'False
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   1815
   ScaleWidth      =   5280
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox tbPassword
      Height          =   375
      IMEMode         =   3  'DISABLE
      Left            =   1200
      PasswordChar    =   "*"
      TabIndex        =   3
      Top             =   720
      Width           =   3975
   End
   Begin VB.CommandButton bCancel
      Cancel          =   -1  'True
      Caption         =   "Cancel"
      Height          =   495
      Left            =   2640
      TabIndex        =   2
      Top             =   1200
      Width           =   1215
   End
   Begin VB.CommandButton bOK
      Caption         =   "OK"
      Default         =   -1  'True
      Height          =   495
      Left            =   3960
      TabIndex        =   1
      Top             =   1200
      Width           =   1215
   End
   Begin VB.Label Label2
      Height          =   495
      Left            =   120
      TabIndex        =   4
      Top             =   120
      Width           =   4935
   End
   Begin VB.Label Label1
      Caption         =   "Password:"
      Height          =   255
      Left            =   120
      TabIndex        =   0
      Top             =   840
      Width           =   855
   End
End
```

frmProgress - 1

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
Option Explicit
```

```
frmProgress - 1

VERSION 5.00
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin VB.Form frmProgress
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Window Title"
   ClientHeight    =   2430
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   6960
   ControlBox      =   0   'False
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   2430
   ScaleWidth      =   6960
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows Default
   Begin MSComctlLib.ProgressBar ProgressBar
      Height          =   735
      Left            =   120
      TabIndex        =   1
      Top             =   1560
      Width           =   6735
      _ExtentX        =   11880
      _ExtentY        =   1296
      _Version        =   393216
      Appearance      =   1
      Scrolling       =   1
   End
   Begin VB.Label lOperation
      Alignment       =   2  'Center
      Caption         =   "Starting Up..."
      BeginProperty Font
         Name            =   "Arial"
         Size            =   15.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   375
      Left            =   120
      TabIndex        =   2
      Top             =   960
      Width           =   6735
   End
   Begin VB.Label lAppName
      Alignment       =   2  'Center
      Caption         =   "App Name"
      BeginProperty Font
         Name            =   "Arial"
         Size            =   24
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   615
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   6735
   End
End
```

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------
Private Sub Form_Load()

    'Restore form to proper position
    Call LoadWindowPosition(Me)

End Sub


Private Sub Form_Activate()

    'Force an update when the form first pops up
    Call UpdateDetailData

End Sub

Private Sub Form_Unload(Cancel As Integer)

    'Save Form position
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------
' Change Register Buttons...
'--------------------------------------------------------------------------
Private Sub bActual_Click()

    'Bring up the dialog for initiating
    'a write to the Trimline Actual Register
    frmWriteReg.Tag = Trim(Str(Me.Tag)) + "AC"
    frmWriteReg.Show
```

```
frmTluDetail - 2

End Sub


'--------------------------------------------------------------------------
' OK - All Done, Go Away
'--------------------------------------------------------------------------
Private Sub bOK_Click()

    'user is done with the dialog
    Unload Me

End Sub


'--------------------------------------------------------------------------
' This function is called by BackgroundFunction() when updated
' Tlu(Unit) Data is available, and the Screen needs to be redrawn
'--------------------------------------------------------------------------
Public Sub UpdateData(Unit As Integer)

    'Module1.UpdateTluForms() calles this sub to tell us
    'Updated data is available in the Tlu() structure
    'for Trimline Unit, we may need to redraw ourselves
    If Unit = Me.Tag Then Call UpdateDetailData

End Sub


Private Sub UpdateDetailData()
    Dim Text As String

    'Update the Window Caption / Title
    Me.Caption = "Unit Detailed View - " + Tlu(Me.Tag).Name
    Me.lTitle = Me.Caption

    'Update the Numeric Value Fields
    With Tlu(Me.Tag)

        'Update Actual
        Me.tActual.Text = fmtNumber(.Actual.Value)
        Me.tsActual.Text = .Actual.Status


        'Update the Schedule Last Check Time
        Dim Schedule As Integer
        Schedule = Tlu(Me.Tag).Schedule
        If (Schedule <> 0) Then
            Me.lLastCheck.Caption = fmtSchedule(Schedule) & " " & .LastCheck
        Else
            Me.lLastCheck.Caption = "(none)"
        End If

    End With

End Sub
```

```
frmTluDetail - 1

VERSION 5.00
Begin VB.Form frmTluDetail
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Unit Detailed View -"
   ClientHeight    =   2535
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   8280
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   2535
   ScaleWidth      =   8280
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton bOK
      Caption         =   "OK"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   6600
      TabIndex        =   3
      Top             =   1920
      Width           =   1575
   End
   Begin VB.CommandButton bActual
      Caption         =   "Change"
      Height          =   360
      Left            =   7200
      TabIndex        =   2
      ToolTipText     =   "Click to change the Actual of the PPT."
      Top             =   1200
      Width           =   975
   End
   Begin VB.TextBox tsActual
      Alignment       =   2  'Center
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   3840
      Locked          =   -1  'True
      TabIndex        =   1
      TabStop         =   0   'False
      ToolTipText     =   "Comunications status for the Actual register."
      Top             =   1200
      Width           =   3255
   End
   Begin VB.TextBox tActual
      Alignment       =   2  'Center
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
```

```
      Left            =   2160
      Locked          =   -1  'True
      TabIndex        =   0
      TabStop         =   0   'False
      ToolTipText     =   "The last Actual read from the PPT."
      Top             =   1200
      Width           =   1575
   End
   Begin VB.Label lLastCheck
      Caption         =   "01/01/1900 12:00:00 AM"
      Height          =   255
      Left            =   120
      TabIndex        =   10
      Top             =   2160
      Width           =   6375
   End
   Begin VB.Label Label3
      Caption         =   "Schedule Processing:"
      Height          =   255
      Left            =   120
      TabIndex        =   9
      Top             =   1920
      Width           =   1815
   End
   Begin VB.Label Label10
      Alignment       =   2  'Center
      Caption         =   "Data Field"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   120
      TabIndex        =   8
      Top             =   720
      Width           =   1815
   End
   Begin VB.Label lTitle
      Alignment       =   2  'Center
      Caption         =   "lTitle"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   18
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   120
      TabIndex        =   7
      Top             =   120
      Width           =   7935
   End
   Begin VB.Label Label9
      Alignment       =   2  'Center
      Caption         =   "Data Field Status"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
```

```
      Height          =   360
      Left            =   3960
      TabIndex        =   6
      Top             =   720
      Width           =   3135
   End
   Begin VB.Label Label8
      Alignment       =   2  'Center
      Caption         =   "Value"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   2160
      TabIndex        =   5
      Top             =   720
      Width           =   1575
   End
   Begin VB.Label lActual
      Alignment       =   2  'Center
      Caption         =   "Actual:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   360
      Left            =   120
      TabIndex        =   4
      Top             =   1200
      Width           =   1815
   End
End
```

frmTotalizers - 1

```
'--------------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------------
Private Sub Form_Load()

    'Restore form to proper position and size
    Call LoadWindowPosition(Me)

    'Initialize the grid display style
    Grid1.HighLight = flexHighlightNever            'Turn off cell highlighting
    Grid1.ScrollTrack = True                        'Enable ScrollTracking
    Grid1.Cols = 3                                  'Set Table Width
    Grid1.Rows = 65                                 'Set Table Height
    Grid1.ColWidth(0) = (Grid1.ColWidth(0) * 3)     'Make the Name Column Triple Width
    Grid1.ColWidth(1) = (Grid1.ColWidth(1) * 3)     'Make the Name Column Triple Width
    Grid1.ColWidth(2) = (Grid1.ColWidth(2) * 3)     'Make the Status Column Triple Width
    Grid1.FillStyle = flexFillRepeat                'settings apply to table selections
    Grid1.Col = 0: Grid1.Row = 0                    'select top left cell
    Grid1.RowSel = Grid1.Rows - 1                   'select all of the first column
    Grid1.ColSel = Grid1.Cols - 1                   'select the entire table
    Grid1.CellAlignment = flexAlignCenterCenter     'Center Text in the Cell
    Grid1.Col = 0: Grid1.Row = 0                    'select top left cell
    Grid1.ColSel = Grid1.Cols - 1                   'select the top row
    Grid1.CellFontBold = True                       'set text to BOLD

    'Set the Grid Column Labels Text
    Grid1.TextArray(0) = "Totalizer Name"
    Grid1.TextArray(1) = "Total Actual"
    Grid1.TextArray(2) = "Status"

    'put the curssor in the upper left corner
    Grid1.Col = 1
    Grid1.Row = 1

End Sub
```

```
frmTotalizers - 2


Private Sub Form_Resize()

    'auto resize the flexgrid to fill the form at run time
    Grid1.Top = 0
    Grid1.Left = 0
    Grid1.Width = Me.ScaleWidth
    Grid1.Height = Me.ScaleHeight

End Sub


Private Sub Form_Unload(Cancel As Integer)

    'Save form position and size
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------
' Pace Timer Data MSFlexGrid
'--------------------------------------------------------------------------
Private Sub Grid1_Click()
    Dim Unit As Integer
    Dim Register As Integer

    'Get the Row and Column clicked on
    Unit = Grid1.MouseRow
    Register = Grid1.MouseCol

    If (Unit >= 1) Then

        'Use selected a Totalizer Detailed View, bring it up
        'frmTotalizerDetail.Tag = Unit
        'frmTotalizerDetail.Show

    End If

End Sub


Private Sub Grid1_KeyPress(KeyAscii As Integer)

    'filter keypresses in the grid
    If KeyAscii = Asc(vbCr) Then
        KeyAscii = 0
        Call Grid1_Click
    End If

End Sub


'--------------------------------------------------------------------------
' This function is called by BackgroundFunction() when updated
' Totalizer(Unit) Data is available, and the Screen needs to be redrawn
'--------------------------------------------------------------------------
Public Sub UpdateData(Unit As Integer)
    Dim RowLen As Integer
    Dim RowStart As Integer
    Dim Text As String

    'Get the number of cells per row
    'for calculating the textarray index
    RowLen = Grid1.Cols

    'Calculate the TextArray Index for the
    'Start of this row
    RowStart = Unit * RowLen

    'update this grid row
    With Totalizer(Unit)
```

```
        Grid1.TextArray(RowStart + 0) = .Name
        Grid1.TextArray(RowStart + 1) = fmtNumber(.TotalActual)
        Grid1.TextArray(RowStart + 2) = .Status
    End With

End Sub
```

```
frmTotalizers - 1

VERSION 5.00
Object = "{5E9E78A0-531B-11CF-91F6-C2863C385E30}#1.0#0"; "MSFLXGRD.OCX"
Begin VB.Form frmTotalizers
   Caption         =   "Totalizers"
   ClientHeight    =   3090
   ClientLeft      =   60
   ClientTop       =   450
   ClientWidth     =   12030
   LinkTopic       =   "Form1"
   ScaleHeight     =   3090
   ScaleWidth      =   12030
   StartUpPosition =   3  'Windows Default
   Begin MSFlexGridLib.MSFlexGrid Grid1
      Height          =   2415
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   3015
      _ExtentX        =   5318
      _ExtentY        =   4260
      _Version        =   393216
   End
End
```

```
'--------------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------------
Option Explicit


'--------------------------------------------------------------------------------
' Form Load / Activate / Resize / Unload
'--------------------------------------------------------------------------------
Private Sub Form_Load()

    'Restore form to proper position
    Call LoadWindowPosition(Me)

End Sub

Private Sub Form_Activate()

    'We were just activated, force an update
    Call UpdateDetailData

End Sub


Private Sub Form_Unload(Cancel As Integer)

    'Save form position
    Call SaveWindowPosition(Me)

End Sub


'--------------------------------------------------------------------------------
' OK and Cancel Buttons...
'--------------------------------------------------------------------------------
Private Sub bAccept_Click()

    'User Accepted the Edit Initiate the write
    Dim Cmd As String
    Dim Unit As Integer
    Unit = Val(Me.Tag)
```

```
    'Password Protected Operation
    If (CheckPassword(Me) = True) Then

        'Write Actual
        If InStr(Me.Tag, "AC") <> 0 Then
            Cmd = "WAC" & RegVal6Digit(Me.tNewVal.Text)
            Call Tlu(Unit).CmdQueue.Enqueue(Cmd)
        End If

        'Go Away
        Unload Me

    End If

End Sub


Private Sub bCancel_Click()

    'User Canceled the update
    'go away
    Unload Me

End Sub


'--------------------------------------------------------------------------
' This function is called by BackgroundFunction() when updated
' Tlu(Unit) Data is available, and the Screen needs to be redrawn
'--------------------------------------------------------------------------
Public Sub UpdateData(Unit As Integer)

    'UpdateTluForms calles this sub to tell us
    'Updated data is available in the Tlu() structure
    'if the updated unit is the one were showing
    'we need to update ourselves
    If Unit = Val(Me.Tag) Then Call UpdateDetailData

End Sub


Private Sub UpdateDetailData()

    'Here is where we do the work of keeping all the
    'controls on the form up to date with Tlu() data
    'Me.Tag contains the Unit/RegID were working with
    Dim Unit As Integer
    Unit = Val(Me.Tag)

    'Updates for Change Actual Dialog
    If InStr(Me.Tag, "AC") <> 0 Then
        Me.Caption = "Change Actual - " + Tlu(Unit).Name
        Me.lTitle = Me.Caption
        Me.lCurrent = "Current Actual:"
        Me.lNew = "New Actual:"
        Me.lCurrentVal = fmtNumber(Tlu(Unit).Actual.Value)
        Me.tNewVal.MaxLength = 6
    End If

End Sub
```

```
frmWriteReg - 1

VERSION 5.00
Begin VB.Form frmWriteReg
   BorderStyle     =   1  'Fixed Single
   Caption         =   "frmWriteReg"
   ClientHeight    =   2490
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   6390
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   2490
   ScaleWidth      =   6390
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton bAccept
      Caption         =   "&OK"
      Default         =   -1  'True
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   3240
      TabIndex        =   3
      ToolTipText     =   "Click here to accept the change."
      Top             =   1920
      Width           =   3015
   End
   Begin VB.CommandButton bCancel
      Cancel          =   -1  'True
      Caption         =   "&Cancel"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   120
      TabIndex        =   2
      ToolTipText     =   "Click here to cancel change."
      Top             =   1920
      Width           =   3015
   End
   Begin VB.TextBox tNewVal
      Alignment       =   2  'Center
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   14.25
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   3240
      TabIndex        =   1
      ToolTipText     =   "Enter the new register value here."
      Top             =   1320
      Width           =   3015
   End
   Begin VB.Label lNew
      Alignment       =   1  'Right Justify
```

```
      Caption         =   "New Reg:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   120
      TabIndex        =   6
      Top             =   1320
      Width           =   2895
   End
   Begin VB.Label lCurrentVal
      Alignment       =   2  'Center
      Caption         =   "000000"
      BeginProperty Font
         Name            =   "Courier New"
         Size            =   14.25
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   3240
      TabIndex        =   5
      ToolTipText     =   "This is the current register value."
      Top             =   720
      Width           =   2895
   End
   Begin VB.Label lCurrent
      Alignment       =   1  'Right Justify
      Caption         =   "Current Reg:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   120
      TabIndex        =   4
      Top             =   720
      Width           =   2895
   End
   Begin VB.Label lTitle
      Alignment       =   2  'Center
      Caption         =   "Change Register"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   13.5
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   450
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   6135
   End
End
```

```
'-----------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'-----------------------------------------------------------------------

'*************************************************************************************************
'Win32API for Saving a Profile Item to a Private INI file
'
'   lpszSection     - INI File Section String
'   lpszKey         - INI File Key String
'   lpszValue       - INI File Value String
'   lpszFile        - Full Path to the Private INI File to be queried
'
' If the function successfully writes the string to the INI file, the return value is nonzero
'*************************************************************************************************
Private Declare Function WritePrivateProfileString Lib "kernel32" _
   Alias "WritePrivateProfileStringA" ( _
   ByVal lpszSection As String, _
   ByVal lpszKey As String, _
   ByVal lpszValue As String, _
   ByVal lpszFile As String) As Long


'*************************************************************************************************
'Win32API for Retrieving a Profile Item from a Private INI file
'
'   lpszSection     - INI File Section String
'   lpszKey         - INI File Key String
'   lpszDefault     - INI File Default Value
'   lpszValue       - A pointer to a null-terminated string buffer that will receive the value
'   nSize           - The size of the lpszValue Buffer
'   lpszFile        - Full Path to the Private INI File to be queried
'
' Returns the number of bytes written into the lpszValue Buffer
'*************************************************************************************************
Private Declare Function GetPrivateProfileString Lib "kernel32" _
    Alias "GetPrivateProfileStringA" ( _
    ByVal lpszSection As String, _
    ByVal lpszKey As String, _
    ByVal lpszDefault As String, _
    ByVal lpszValue As String, _
    ByVal nSize As Long, _
    ByVal lpszFile As String) As Long
```

```
'*********************************************************************************************
'Win32API for Retrieving Special Operating System Folders by CSIDL Identifier
'
'   hWndOwner   - Reserved
'   nFolder     - CSIDL Folder Identifier
'   nToken      - An access token that can be used to represent a particular user (Win2K must be NULL
)
'   dwFlags     - Flags that specify the path to be returned (Current or Default)
'   lpszPath    - A pointer to a null-terminated string of length MAX_PATH which will receive the pat
h
'
' Returns
'   S_OK            - if successful, request path is written into lpszPath
'   S_FALSE         - The CSIDL in nFolder is valid, but the folder does not exist.
'   E_INVALIDARG    - The CSIDL in nFolder is not valid.
'*********************************************************************************************
Private Declare Function SHGetFolderPath Lib "shell32" _
    Alias "SHGetFolderPathA" ( _
    ByVal hWndOwner As Long, _
    ByVal nFolder As Long, _
    ByVal hToken As Long, _
    ByVal dwFlags As Long, _
    ByVal lpszPath As String) As Long


'*********************************************************************************************
'Constants requrired for SHGetFolderPath()
'nFolder - CSIDL Special Folder Identifiers
'dwFlags - Flag Values
'return  - Return Values
'*********************************************************************************************
'The file system directory that serves as a common repository for application-specific data.
'A typical path is C:\Documents and Settings\username\Application Data
Private Const CSIDL_APPDATA = &H1A

'The file system directory that contains application data for all users.
'A typical path is C:\Documents and Settings\All Users\Application Data
Private Const CSIDL_COMMON_APPDATA = &H23

'The file system directory that contains files and folders that appear on the desktop for all users.
'A typical path is C:\Documents and Settings\All Users\Desktop
Private Const CSIDL_COMMON_DESKTOPDIRECTORY = &H19

'The file system directory that contains documents that are common to all users.
'A typical paths is C:\Documents and Settings\All Users\Documents
Private Const CSIDL_COMMON_DOCUMENTS = &H2E

'The file system directory that contains the common program groups that appear on the Start menu for
all users.
'A typical path is C:\Documents and Settings\All Users\Start Menu\Programs
Private Const CSIDL_COMMON_PROGRAMS = &H17

'The file system directory that contains the programs and folders that appear on the Start menu for a
ll users.
'A typical path is C:\Documents and Settings\All Users\Start Menu
Private Const CSIDL_COMMON_STARTMENU = &H16

'The file system directory that contains the programs that appear in the Startup folder for all users
.
'A typical path is C:\Documents and Settings\All Users\Start Menu\Programs\Startup
Private Const CSIDL_COMMON_STARTUP = &H18

'The file system directory used to physically store file objects on the desktop.
'A typical path is C:\Documents and Settings\username\Desktop
Private Const CSIDL_DESKTOPDIRECTORY = &H10

'The file system directory that serves as a data repository for local (nonroaming) applications.
'A typical path is C:\Documents and Settings\username\Local Settings\Application Data
Private Const CSIDL_LOCAL_APPDATA = &H1C
```

```
'The file system directory that serves as a common repository for image files.
'A typical path is C:\Documents and Settings\username\My Documents\My Pictures
Private Const CSIDL_MYPICTURES = &H27

'The user's profile folder.
'A typical path is C:\Users\username
Private Const CSIDL_PROFILE = &H28

'The Program Files folder.
'A typical path is C:\Program Files
Private Const CSIDL_PROGRAM_FILES = &H26

'A folder for components that are shared across applications.
'A typical path is C:\Program Files\Common
Private Const CSIDL_PROGRAM_FILES_COMMON = &H2B

'The file system directory that contains the user's program groups.
'A typical path is C:\Documents and Settings\username\Start Menu\Programs
Private Const CSIDL_PROGRAMS = &H2

'The file system directory that contains Start menu items.
'A typical path is C:\Documents and Settings\username\Start Menu
Private Const CSIDL_STARTMENU = &HB

'The file system directory that corresponds to the user's Startup program group.
'A typical path is C:\Documents and Settings\username\Start Menu\Programs\Startup
Private Const CSIDL_STARTUP = &H7

'The Windows System folder.
'A typical path is C:\Windows\System32
Private Const CSIDL_SYSTEM = &H25

'The Windows directory or SYSROOT. This corresponds to the %windir% or %SYSTEMROOT% environment varia
bles.
'A typical path is C:\Windows
Private Const CSIDL_WINDOWS = &H24

'Combine with another CSIDL to force the creation of the associated folder if it does not exist
Private Const CSIDL_FLAG_CREATE = &H8000&

'Retrieve the folder's current path
Private Const SHGFP_TYPE_CURRENT = &H0

'Retrieve the folder's default path
Private Const SHGFP_TYPE_DEFAULT = &H1

'Requested Path is stored at lpzPath
Private Const S_OK = 0

'The CSIDL  in nFolder is valid, but the folder does not exist
Private Const S_FALSE = 1

'The CSIDL  in nFolder is not valid
Private Const E_INVALIDARG = &H80070057

'Win32 Maximim Path Length Limit
Private Const MAX_PATH = 260


'*********************************************************************************************
'A thin VB6 Wrapper for the Win32 SHGetFolderPath() API in Shell32.dll
'*********************************************************************************************
Public Function VbGetFolderPath(ByVal nFolder As Long, ByVal dwFlags As Long) As String
    Dim strPath As String
    Dim lngResult As Long

    'The OS, and VB6 do not aggree on the defintion of a string
    'to work around this, we have to force VB6 to allocate a
    'string buffer large enough to store the largest possible result
    strPath = Space$(MAX_PATH)

    'Ask the OS for the CSIDL specified path
```

```
    lngResult = SHGetFolderPath(0&, nFolder, 0&, dwFlags, strPath)

    'Parce the result of the call into SHGetFolderPath()
    Select Case lngResult

        Case S_OK           'Return Specified Path String to Caller
            VbGetFolderPath = Left$(strPath, InStr(strPath, vbNullChar) - 1)

        Case S_FALSE        'Raise an error
            Err.Raise vbObjectError + &HB112&, _
                "VbGetFolderPath()", _
                "Valid PathId but folder doesn't exist"

        Case E_INVALIDARG   'Raise an error
            Err.Raise vbObjectError + &HB114&, _
                "VbGetFolderPath()", _
                "Invalid PathId for requested function"

        Case Else           'Raise an Error
            Err.Raise vbObjectError + &HB116&, _
                "VbGetFolderPath()", _
                "System error " & CStr(lngResult)

    End Select
End Function


'*********************************************************************************************
'Function to detect if the program is running inside the IDE, or as a stand alone program
'returns TRUE if the program is running within the VB6 Integrated Development Environment
'*********************************************************************************************
Public Function InIDE() As Boolean
    On Error Resume Next    'Do not terminate on an error
    Debug.Assert 1 / 0      'If we are running under the IDE, this generates a divide by zero error
    InIDE = (Err <> 0)      'Reduce the Error Result to a Boolean True / False
End Function


'*********************************************************************************************
'A function to retrieve the directory where common application data is stored
'*********************************************************************************************
Public Function GetCommonAppDataFolder() As String

    'If we are running in the Integrated Development Environment
    If (InIDE() = True) Then
        'then returnt the Source Code Directory instead of the Real Common AppData Directory
        GetCommonAppDataFolder = App.Path
    Else
        'else return the Common Application Data Diretory (Varies by OS Version)
        GetCommonAppDataFolder = VbGetFolderPath(CSIDL_COMMON_APPDATA, SHGFP_TYPE_CURRENT)
        GetCommonAppDataFolder = GetCommonAppDataFolder & "\" & App.Title
    End If

End Function


'*********************************************************************************************
'A function to retrieve the directory where common documents are stored
'*********************************************************************************************
Public Function GetCommonDocumentsFolder() As String

    'If we are running in the Integrated Development Environment
    If (InIDE() = True) Then
        'then return the Source Code Directory instead of the Real Common Documents Directory
        GetCommonDocumentsFolder = App.Path
    Else
        'else return the Common Documents Directory (Varies by OS Version)
        GetCommonDocumentsFolder = VbGetFolderPath(CSIDL_COMMON_DOCUMENTS, SHGFP_TYPE_CURRENT)
        GetCommonDocumentsFolder = GetCommonDocumentsFolder & "\" & App.Title

    End If
```

```
IniFile - 5

End Function


'*********************************************************************************************
'A simple function to write a profile string to a private INI file
'*********************************************************************************************
Public Sub WriteIni(File As String, Section As String, Key As String, Value As String)
   Call WritePrivateProfileString(Section, Key, Value, File)
End Sub


'*********************************************************************************************
'A simple function to read a profile string from a private INI file
'*********************************************************************************************
Public Function ReadIni(File As String, Section As String, Key As String, Default As String) As Strin
g
    Dim Value As String
    Dim Size As Long

    'The OS, and VB6 do not aggree on the defintion of a string
    'to work around this, we have to force VB6 to allocate a
    'string buffer large enough to store the largest possible result
    Value = Space$(255)
    Size = Len(Value)

    'Ask the OS to read the INI file for us
    Size = GetPrivateProfileString(Section, Key, Default, Value, Size, File)

    'Parce the result of the GetPrivateProfileString OS call
    If (Size > 0) Then
        ReadIni = Mid(Value, 1, Size)
    Else
        ReadIni = ""
    End If
End Function


'*********************************************************************************************
'Make a Directory Path
'*********************************************************************************************
Public Sub MakePath(ByVal Path As String)
Dim directories As Variant
Dim i As Integer
Dim new_dir As String
Dim dir_path As String

    'Break the path into componet directories.
    directories = Split(Path, "\")

    'Walk the path, left to right, one directory at a time
    For i = LBound(directories) To UBound(directories)

        'Get the next directory in the path.
        new_dir = directories(i)

        'If the next directory is not zero length
        If Len(new_dir) > 0 Then

            'Append it to the working path
            dir_path = dir_path & new_dir & "\"

            'Make sure we don't just have a drive specification.
            If Right$(new_dir, 1) <> ":" Then

                'If the path does not exist
                If Dir$(dir_path, vbDirectory) = "" Then

                    'Make it.
                    MkDir dir_path

                End If
```

```
            End If

        End If

    Next i

End Sub


'*********************************************************************************************
'A Simple ROT13 to Scramble Passwords
'*********************************************************************************************
Public Function ROT13(SecretWord As String) As String
    Dim i As Integer
    Dim c As Integer

    For i = 1 To Len(SecretWord)
        c = Asc(Mid(SecretWord, i, 1))
        If (c >= Asc("A") And c <= Asc("Z")) Then
            c = c + 13
            If (c > Asc("Z")) Then c = c - 26
        ElseIf (c >= Asc("a") And c <= Asc("z")) Then
            c = c + 13
            If (c > Asc("z")) Then c = c - 26
        End If
        ROT13 = ROT13 & Chr(c)
    Next i

End Function
```

```
'--------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'--------------------------------------------------------------------------
' V2.0 01-09-2012 Rewrote Trimline Monitor from Scratch,
'                 Using PPT Monitor 4.0 as a starting point (PD-2116-051)
' V2.1 01-19-2010 Moved Data Store from Registry to Common Documents Folder
'                 Win7 = c:\users\public\documents
'                 XP   = c:\documents and settings\all users\documents
'                 Win98= c:\My Documents
'--------------------------------------------------------------------------
Option Explicit

'--------------------------------------------------------------------------
' Declare Data Structure/Storage for the Trimline Counters / Timers in RAM
'--------------------------------------------------------------------------
Public Const NUMTLU As Integer = 64

'Declare structure for tracking a TLU Register in RAM
Type TLUREG
    Value As Long               'Register Value (Read)
    Status As String            'Register Status
End Type

'Declare structure for tracking a single Trimline Unit in RAM
Type TLUTYPE
    Name As String              'Trimline Symbolic Name
    Port As Integer             'Comport 1 to 16 (0=Disabled, 1=COM1:, 16=TCP)
    Address As Integer          'Address character 32 to 126
    HostName As String          'TCP/IP Host Name
    Schedule As Integer         'Schedule Selection 1 to 64
    Actual As TLUREG            'Actual Count/Time Register
    Status As String            'Trimline Status
    LastCheck As Date           'Last Time Schedule was Checked
    CmdQueue As Queue           'Pending Commands for this Trimline
End Type

'Dim an array to store all data for all Trimlines's in RAM
Public Tlu(1 To NUMTLU) As TLUTYPE


'--------------------------------------------------------------------------
' Declare Data Structure/Storage for the Trimline Counters / Timers on Disk
```

```
'--------------------------------------------------------------------------
'Declare structure for tracking a TLU Register on Disk
Type TLUREGDISK
    Value As Long               'Register Value (Read)
    Status As String * 256      'Register Status
End Type

'Declare structure for tracking a single Trimline Unit on Disk
Type TLUTYPEDISK
    Name As String * 256        'Trimline Symbolic Name
    Port As Integer             'Comport 1 to 16 (0=Disabled, 1=COM1:, 16=TCP)
    Address As Integer          'Address character 32 to 126
    HostName As String * 256    'TCP/IP Host Name
    Schedule As Integer         'Schedule Selection 1 to 64
    Actual As TLUREGDISK        'Actual Count/Time Register
    Status As String * 256      'Trimline Status
    LastCheck As Date           'Last Time Schedule was Checked
End Type


'--------------------------------------------------------------------------
' Declare Data Structure/Storage for the Slave Totalizer Displays in RAM
'--------------------------------------------------------------------------
Public Const NUMTOTALIZER As Integer = 64

'Declare a structure for storing a Totalizer State in RAM
Type TOTALIZERTYPE
    Name As String                          'Symbolic Name of this Totalizer
    Port As Integer                         'Comport 1 to 16 (0=Disabled, 1=COM1:, 16=TCP))
    HostName As String                      'TCP/IP HostName
    ActualAddress As Integer                'Address Character 32 to 126
    TluIsMember(1 To NUMTLU) As Integer     'Is this Trimline Unit used in the Calculation
    TotalActual As Long                     'Result Total Actual
    Status As String                        'Communications Status
End Type

'Global Storrage for All Named Totalizer Displays in RAM
Public Totalizer(1 To NUMTOTALIZER) As TOTALIZERTYPE


'--------------------------------------------------------------------------
' Declare Data Structure/Storage for the Slave Totalizer Displays on Disk
'--------------------------------------------------------------------------
'Declare a structure for storing a Totalizer State on Disk
Type TOTALIZERTYPEDISK
    Name As String * 256                    'Symbolic Name of this Totalizer
    Port As Integer                         'Comport 1 to 16 (0=Disabled, 1=COM1:, 16=TCP))
    HostName As String * 256                'TCP/IP HostName
    ActualAddress As Integer                'Address Character 32 to 126
    TluIsMember(1 To NUMTLU) As Integer     'Is this Trimline Unit used in the Calculation
    TotalActual As Long                     'Result Total Actual
    Status As String * 256                  'Communications Status
End Type


'--------------------------------------------------------------------------
' Declare Data Structure/Storage for the Named Schedules in RAM
'--------------------------------------------------------------------------
Public Const NUMSCHEDULE As Integer = 64
Public Const NUMSHIFT As Integer = 8

'Declare a structure for storing a Scheduled Shift in RAM
Type SHIFTTYPE
    Rst As Integer      'Rest at start of shift?
    Exp As Integer      'CSV Export at end of shift?
    Sun As Integer      'Enabled on Sunday
    Mon As Integer      'Enabled on Monday
    Tue As Integer      'Enabled on Tuesday
    Wed As Integer      'Enabled on Wednesday
    Thr As Integer      'Enabled on Thursday
    Fri As Integer      'Enabled on Friday
    Sat As Integer      'Enabled on Saturday
```

```
    Beg As Date         'Begin Time
    End As Date         'End Time
End Type

'Declare a structure for storing a Named Schedule in RAM
Type SCHEDULETYPE
    Name As String                      'Schedule Symbolic Name
    Shift(1 To NUMSHIFT) As SHIFTTYPE   '8 Shifts per Named Schedule
End Type

'Global Storage for All Named Schedules in RAM
Public Schedule(1 To NUMSCHEDULE) As SCHEDULETYPE


'---------------------------------------------------------------------------
' Declare Data Structure/Storage for the Named Schedules on Disk
'---------------------------------------------------------------------------
'Declare a structure for storing a Scheduled Shift
Type SHIFTTYPEDISK
    Rst As Integer      'Rest at start of shift?
    Exp As Integer      'CSV Export at end of shift?
    Sun As Integer      'Enabled on Sunday
    Mon As Integer      'Enabled on Monday
    Tue As Integer      'Enabled on Tuesday
    Wed As Integer      'Enabled on Wednesday
    Thr As Integer      'Enabled on Thursday
    Fri As Integer      'Enabled on Friday
    Sat As Integer      'Enabled on Saturday
    Beg As Date         'Begin Time
    End As Date         'End Time
End Type

'Declare a structure for storing a Named Schedule on Disk
Type SCHEDULETYPEDISK
    Name As String * 256                'Schedule Symbolic Name
    Shift(1 To NUMSHIFT) As SHIFTTYPEDISK '8 Shifts per Named Schedule
End Type


'---------------------------------------------------------------------------
' Misc Global Variables for Internal State
'---------------------------------------------------------------------------
'Storage for misc program state
Public OpState As Boolean       'Make false to kill background functions
Public SecretWord As String     'Password to make changes
Public DebugFlag As Boolean     'Is Debuging Enabled?

'Queue for Program to User Messages
Public MsgQueue As New Queue

'---------------------------------------------------------------------------
' Declare Windows API Functions we Need...
'---------------------------------------------------------------------------
Private Declare Sub Sleep Lib "kernel32.dll" (ByVal dwMilliseconds As Long)


'###########################################################################
' MAIN()
' The whole program begins and ends here!
'###########################################################################
Public Sub Main()
    'Create the main form and show it
    frmMain.Show
    Call YieldProcessor

    'Load the System Configuration
    Call LoadAllConfiguration

    'Debug Log Start
    Call LogEvent("Main()", "Program Start")

    'Start the background comunications task
```

```
    OpState = True              'set to false to kill the background task
    Call BackgroundFunction     'This will not return until OpState is false

    'Debug Log End
    Call LogEvent("Main()", "Program End")

    'Save the system configuration before we end
    Call SaveAllConfiguration

    'Make sure the program terminates
    End

End Sub


'################################################################################
' Background Trimline Unit Communications
' Background Totalizer Communications
' Background Schedule Processing
'################################################################################
Public Sub BackgroundFunction()
    Dim Unit As Integer
    On Error Resume Next

    'This function runs in the "Background" and
    'handles all comunications with the Trimlines's
    'This sub is put in the background by carful use of DoEvents
    'Set OpState = False to kill this sub, or it loops forever

    'Load the Winsock Controls
    For Unit = LBound(Tlu) To UBound(Tlu)
        Load frmMain.Winsock1(Unit)
    Next Unit
    For Unit = LBound(Totalizer) To UBound(Totalizer)
        Load frmMain.Winsock2(Unit)
    Next Unit

    'For as long as the programm is running
    While OpState

        'Communicate with all possible Trimline units
        For Unit = LBound(Tlu) To UBound(Tlu)

            ' Yield to the main form for a bit
            Call YieldProcessor

            'Check to see if were still running, if not exit loop now
            If Not OpState Then Exit For

            'Try to Communicate with the Trimline Unit
            Call CommunicateWithTlu(Unit)

        Next Unit

        'Communicate with all possible Totalizer units
        For Unit = LBound(Totalizer) To UBound(Totalizer)

            ' Yield to the main form for a bit
            Call YieldProcessor

            'Check to see if were still running, if not exit loop now
            If Not OpState Then Exit For

            'Try to Communicate with the Totalizer
            Call CommunicateWithTotalizer(Unit)

        Next Unit

    Wend 'loop for as long as the program is running

    'Make sure the serial ports and winsocks are closed before exiting
    If (frmMain.MSComm1.PortOpen = True) Then frmMain.MSComm1.PortOpen = False
```

```
    For Unit = LBound(Tlu) To UBound(Tlu)
        If frmMain.Winsock1(Unit).State <> sckClosed Then
            Call frmMain.Winsock1(Unit).Close
        End If
    Next Unit
    For Unit = LBound(Totalizer) To UBound(Totalizer)
        If frmMain.Winsock2(Unit).State <> sckClosed Then
            Call frmMain.Winsock2(Unit).Close
        End If
    Next Unit

End Sub


'--------------------------------------------------------------------------
' Attempt to Communicate with the Specified Trimline Unit
'--------------------------------------------------------------------------
Public Sub CommunicateWithTlu(Unit As Integer)
    On Error Resume Next

    'If the Trimline is SERIAL
    If (Tlu(Unit).Port >= 1 And Tlu(Unit).Port <= 15) Then

        'Make sure the Associated Winsock Control is closed
        If frmMain.Winsock1(Unit).State <> sckClosed Then
            Call frmMain.Winsock1(Unit).Close
        End If

        'Communicate With Trimline Via Comport
        Call CommunicateWithTluViaCom(Unit)

    'If the Trimline is TCP/IP
    ElseIf (Tlu(Unit).Port = 16) Then

        'Communicate With the Trimline Via TCP/IP
        Call CommunicateWithTluViaTcp(Unit)

    'Else the Trimline is UNUSED / UNKNOWN
    Else

        'Make sure the Associated Winsock Control is closed
        If frmMain.Winsock1(Unit).State <> sckClosed Then
            Call frmMain.Winsock1(Unit).Close
        End If

        'Make Sure Trimline State is UNUSED
        Call EraseTluConfiguration(Unit)
        Call UpdateTluForms(Unit)
        Call YieldProcessor

    End If

End Sub


'--------------------------------------------------------------------------
' Attempt to Communicate with the Specified Trimline Via Serial Port
'--------------------------------------------------------------------------
Public Sub CommunicateWithTluViaCom(Unit As Integer)

    'If the currently select COM port is wrong for this Trimline
    If frmMain.MSComm1.CommPort <> Tlu(Unit).Port Then  'If the wrong COM is selected
        If (frmMain.MSComm1.PortOpen = True) Then       'And the COM control is Open
            frmMain.MSComm1.PortOpen = False            'close it to allow reprogramming
        End If
    End If

    'If the COM control is closed, try to open it
    If frmMain.MSComm1.PortOpen = False Then        'If the Com Control is Closed
        frmMain.MSComm1.CommPort = Tlu(Unit).Port   'Use the Specified Port
        frmMain.MSComm1.Handshaking = comNone       'With all handshacking disabled
        frmMain.MSComm1.Settings = "19200,n,8,1"    'at 19200 baud, no parity, 8 data, 1 stop
```

```
        frmMain.MSComm1.PortOpen = True             'Try to open the com control
    End If

    'If the Com Control is open, Exchange data with the Trimline Unit
    If frmMain.MSComm1.PortOpen = True Then

        If (OpState) Then Call CheckTluSchedule(Unit)
        If (OpState) Then Call WriteTluRegsViaCom(Unit)
        If (OpState) Then Call ReadTluRegsViaCom(Unit)
        Call SaveTluConfiguration(Unit)
        Call YieldProcessor

    Else

        'Comport failed to open
        'flag PortError on all registers for this Trimline Unit
        Call SetTluStatus(Unit, "COM Port Error")
        Call UpdateTluForms(Unit)

    End If

End Sub


'--------------------------------------------------------------------------
' Attempt to Communicate with the Specified Trimline via TCP/IP
'--------------------------------------------------------------------------
Public Sub CommunicateWithTluViaTcp(Unit As Integer)

    'TCP Connection State Machine...
    Select Case (frmMain.Winsock1(Unit).State)

        'TCP Socket is Closed
        Case sckClosed
            Call SetTluStatus(Unit, "TCP Closed")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Open a Connection to the Trimline Unit
            frmMain.Winsock1(Unit).Protocol = sckTCPProtocol
            frmMain.Winsock1(Unit).RemoteHost = Tlu(Unit).HostName
            frmMain.Winsock1(Unit).RemotePort = 3001
            frmMain.Winsock1(Unit).Connect

        'TCP Socket is Open
        Case sckOpen
            Call SetTluStatus(Unit, "TCP Open")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Invalid State, Close Socket
            Call frmMain.Winsock1(Unit).Close

        'TCP Socket is Listening for an Inbound Connection Request
        Case sckListening
            Call SetTluStatus(Unit, "TCP Listening...")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Invalid State, close Socket
            Call frmMain.Winsock1(Unit).Close

        'TCP Socket Inbound Connection Request is Pending
        Case sckConnectionPending
            Call SetTluStatus(Unit, "Connection Pending...")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Invalid State, Close Slocket
            Call frmMain.Winsock1(Unit).Close

        'TCP Socket is Performing a DNS Lookup
```

```
        Case sckResolvingHost
            Call SetTluStatus(Unit, "Resolving Host...")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Waiting on DNS to Resolve HostName...
            'Will Auto Advance to sckHostResolved or sckError

        'TCP Socket DNS Lookup Completed
        Case sckHostResolved
            Call SetTluStatus(Unit, "TCP Host Resolved")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'DNS Resolved HostName
            'Will Auto Advance to Connecting...

        'TCP Socket is Initiating an Outbound Connection Request
        Case sckConnecting
            Call SetTluStatus(Unit, "TCP Connecting...")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Waiting on 3 Way ACK from Trimline...
            'Will Auto Advance to sckConnected or sckError

        'TCP Socket is Connected to a Remote Host
        Case sckConnected
            'Call SetTluStatus(Unit, "TCP Connected")
            'Call UpdateTluForms(Unit)
            'Call YieldProcessor

            'Connected to a Trimline, Exchange Data or Disconnect
            If (frmMain.Winsock1(Unit).RemoteHost = Tlu(Unit).HostName) Then
                If (OpState) Then Call CheckTluSchedule(Unit)
                If (OpState) Then Call WriteTluRegsViaTcp(Unit)
                If (OpState) Then Call ReadTluRegsViaTcp(Unit)
                Call SaveTluConfiguration(Unit)
                Call YieldProcessor
            Else
                Call frmMain.Winsock1(Unit).Close
            End If

        'TCP Socket is Closing due to Remote Shutdown Request
        Case sckClosing
            Call SetTluStatus(Unit, "TCP Closing...")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Remote Host Wants to Close the Connection
            Call frmMain.Winsock1(Unit).Close

        'An Error has Occured
        Case sckError
            Call SetTluStatus(Unit, "TCP Error")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Invalid State, Close the Socket
            Call frmMain.Winsock1(Unit).Close

        'The TCP Socket is in an Unknown State
        Case Else
            Call SetTluStatus(Unit, "TCP Unknown State")
            Call UpdateTluForms(Unit)
            Call YieldProcessor

            'Invalid State, Close the Connection
            Call frmMain.Winsock1(Unit).Close

    End Select
```

```
End Sub


'--------------------------------------------------------------------------
' Attempt to Communicate with the Specified Totalizer
'--------------------------------------------------------------------------
Public Sub CommunicateWithTotalizer(Unit As Integer)
    On Error Resume Next

    'If the Totalizer is SERIAL
    If (Totalizer(Unit).Port >= 1 And Totalizer(Unit).Port <= 15) Then

        'Make sure the Associated Winsock Control is closed
        If frmMain.Winsock2(Unit).State <> sckClosed Then
            Call frmMain.Winsock2(Unit).Close
        End If

        'Communicate With Totalizer Via Comport
        Call ComputeTotalizerTotals(Unit)
        Call CommunicateWithTotalizerViaCom(Unit)
        Call UpdateTotalizerForms(Unit)

    'If the Totalizer is TCP/IP
    ElseIf (Totalizer(Unit).Port = 16) Then

        'Communicate With the Totalizer Via TCP/IP
        Call ComputeTotalizerTotals(Unit)
        Call CommunicateWithTotalizerViaTcp(Unit)
        Call UpdateTotalizerForms(Unit)

    'Else the Totalizer is UNUSED / UNKNOWN
    Else

        'Make sure the Associated Winsock Control is closed
        If frmMain.Winsock2(Unit).State <> sckClosed Then
            Call frmMain.Winsock2(Unit).Close
        End If

        'Make Sure Totalizer State is UNUSED
        Call EraseTotalizerConfiguration(Unit)
        Call UpdateTotalizerForms(Unit)
        Call YieldProcessor

    End If

End Sub


'--------------------------------------------------------------------------
' Compute New Totals for the Specified Totalizer
'--------------------------------------------------------------------------
Public Sub ComputeTotalizerTotals(Unit As Integer)
    Dim TotalActual As Long
    Dim i As Integer

    'Iterate over all Trimline Totalizer Members of Totalizer Group
    Totalizer(Unit).Status = "Calculating..."
    Call UpdateTotalizerForms(Unit)
    Call YieldProcessor
    For i = LBound(Totalizer(Unit).TluIsMember) To UBound(Totalizer(Unit).TluIsMember)
        If (Totalizer(Unit).TluIsMember(i) <> 0) Then
            TotalActual = TotalActual + Tlu(i).Actual.Value
        End If
    Next i

    'Update Totalizer Totals
    Totalizer(Unit).TotalActual = TotalActual
    Totalizer(Unit).Status = "Calculation Done"
    Call UpdateTotalizerForms(Unit)
    Call YieldProcessor

End Sub
```

```
'--------------------------------------------------------------------------
' Attempt to Communicate with the Specified Totalizer Via Serial Port
'--------------------------------------------------------------------------
Public Sub CommunicateWithTotalizerViaCom(Unit As Integer)

    'If the currently select COM port is wrong for this Totalizer
    If frmMain.MSComm1.CommPort <> Totalizer(Unit).Port Then    'If the wrong COM is selected
        If (frmMain.MSComm1.PortOpen = True) Then               'And the COM control is Open
            frmMain.MSComm1.PortOpen = False                    'close it to allow reprogramming
        End If
    End If

    'If the COM control is closed, try to open it
    If frmMain.MSComm1.PortOpen = False Then                'If the Com Control is Closed
        frmMain.MSComm1.CommPort = Totalizer(Unit).Port     'Use the Specified Port
        frmMain.MSComm1.Handshaking = comNone               'With all handshacking disabled
        frmMain.MSComm1.Settings = "19200,n,8,1"            'at 19200 baud, no parity, 8 data, 1 stop
        frmMain.MSComm1.PortOpen = True                     'Try to open the com control
    End If

    'If the Com Control is open, Send Data to the Totalizer
    If frmMain.MSComm1.PortOpen = True Then

        If (OpState) Then Call WriteTotalizerViaCom(Unit)
        Call YieldProcessor

    Else

        'Comport failed to open
        'flag PortError on all registers for this Trimline
        Totalizer(Unit).Status = "COM Port Error"
        Call UpdateTotalizerForms(Unit)

    End If

End Sub


'--------------------------------------------------------------------------
' Write Totalizer Data to Totalizer via Serial Port
'--------------------------------------------------------------------------
Public Sub WriteTotalizerViaCom(Unit As Integer)
    Dim STX As String
    Dim ActAddr As String
    Dim ETX As String
    Dim TXBuf As String
    Dim RXBuf As String

    With Totalizer(Unit)

        'Define strings to make the program more readable
        STX = Chr$(2)                       'CTRL-B, Start of transmition
        ActAddr = Chr(.ActualAddress)       'Field Address Character
        ETX = Chr$(3)                       'CTRL-C, End of transmition

        'Tell the world were working on this Totalizer
        .Status = "Writing..."
        Call UpdateTotalizerForms(Unit)

        'Flush the Receive Buffer
        While frmMain.MSComm1.InBufferCount
            RXBuf = frmMain.MSComm1.Input
        Wend
        RXBuf = ""

        'Transmit the Write Command to the Totalizer
        TXBuf = ""
        If (.ActualAddress <> 0) Then TXBuf = TXBuf & STX & ActAddr & fmtNumber(.TotalActual) & ETX
        frmMain.MSComm1.Output = TXBuf
```

```
    End With

    'Tell the world were working on this Totalizer
    Totalizer(Unit).Status = "Done"

End Sub


'--------------------------------------------------------------------------
' Attempt to Communicate with the Specified Totalizer via TCP/IP
'--------------------------------------------------------------------------
Public Sub CommunicateWithTotalizerViaTcp(Unit As Integer)

    'TCP Connection State Machine...
    Select Case (frmMain.Winsock2(Unit).State)

        'TCP Socket is Closed
        Case sckClosed
            Totalizer(Unit).Status = "TCP Closed"
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Open a Connection to the Totalizer
            frmMain.Winsock2(Unit).Protocol = sckTCPProtocol
            frmMain.Winsock2(Unit).RemoteHost = Totalizer(Unit).HostName
            frmMain.Winsock2(Unit).RemotePort = 3001
            frmMain.Winsock2(Unit).Connect

        'TCP Socket is Open
        Case sckOpen
            Totalizer(Unit).Status = "TCP Open"
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Invalid State, Close Socket
            Call frmMain.Winsock2(Unit).Close

        'TCP Socket is Listening for an Inbound Connection Request
        Case sckListening
            Totalizer(Unit).Status = "TCP Listening..."
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Invalid State, close Socket
            Call frmMain.Winsock2(Unit).Close

        'TCP Socket Inbound Connection Request is Pending
        Case sckConnectionPending
            Totalizer(Unit).Status = "Connection Pending..."
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Invalid State, Close Slocket
            Call frmMain.Winsock2(Unit).Close

        'TCP Socket is Performing a DNS Lookup
        Case sckResolvingHost
            Totalizer(Unit).Status = "Resolving Host..."
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Waiting on DNS to Resolve HostName...
            'Will Auto Advance to sckHostResolved or sckError

        'TCP Socket DNS Lookup Completed
        Case sckHostResolved
            Totalizer(Unit).Status = "TCP Host Resolved"
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'DNS Resolved HostName
            'Will Auto Advance to Connecting...
```

```
        'TCP Socket is Initiating an Outbound Connection Request
        Case sckConnecting
            Totalizer(Unit).Status = "TCP Connecting..."
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Waiting on 3 Way ACK from Totalizer...
            'Will Auto Advance to sckConnected or sckError

        'TCP Socket is Connected to a Remote Host
        Case sckConnected
            'Totalizer(Unit).Status = "TCP Connected"
            'Call UpdateTotalizerForms(Unit)
            'Call YieldProcessor

            'Connected to a Totalizer, Write Data or Disconnect
            If (frmMain.Winsock2(Unit).RemoteHost = Totalizer(Unit).HostName) Then
                If (OpState) Then Call WriteTotalizerViaTcp(Unit)
                Call SaveTotalizerConfiguration(Unit)
                Call YieldProcessor
            Else
                Call frmMain.Winsock2(Unit).Close
            End If

        'TCP Socket is Closing due to Remote Shutdown Request
        Case sckClosing
            Totalizer(Unit).Status = "TCP Closing..."
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Remote Host Wants to Close the Connection
            Call frmMain.Winsock2(Unit).Close

        'An Error has Occured
        Case sckError
            Totalizer(Unit).Status = "TCP Error"
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Invalid State, Close the Socket
            Call frmMain.Winsock2(Unit).Close

        'The TCP Socket is in an Unknown State
        Case Else
            Totalizer(Unit).Status = "TCP Unknown State"
            Call UpdateTotalizerForms(Unit)
            Call YieldProcessor

            'Invalid State, Close the Connection
            Call frmMain.Winsock2(Unit).Close

    End Select

End Sub


'--------------------------------------------------------------------------
' Write Totalizer Data to Totalizer via TCP/IP
'--------------------------------------------------------------------------
Public Sub WriteTotalizerViaTcp(Unit As Integer)
    Dim STX As String
    Dim ActAddr As String
    Dim ETX As String
    Dim TXBuf As String
    Dim Temp As String

    With Totalizer(Unit)

        'Define strings to make the program more readable
        STX = Chr$(2)                       'CTRL-B, Start of transmition
        ActAddr = Chr(.ActualAddress)       'Field Address Character
```

```
        ETX = Chr$(3)                       'CTRL-C, End of transmition

        'Tell the world were working on this Totalizer
        .Status = "Writing..."
        Call UpdateTotalizerForms(Unit)

        'Flush the Receive Buffer
        While frmMain.Winsock2(Unit).BytesReceived > 0
            Call frmMain.Winsock2(Unit).GetData(Temp)
        Wend

        'Transmit the Write Command to the Totalizer
        TXBuf = ""
        If (.ActualAddress <> 0) Then TXBuf = TXBuf & STX & ActAddr & fmtNumber(.TotalActual) & ETX
        Call frmMain.Winsock2(Unit).SendData(TXBuf)

        'Tell the world were working on this Totalizer
        .Status = "Done"

    End With

End Sub


'--------------------------------------------------------------------------
' Utility Function
' Set all Trimline Status Fields to the Specified Text
'--------------------------------------------------------------------------
Public Sub SetTluStatus(Unit As Integer, Status As String)

    'Set unit and all registers status fields to Status
    With Tlu(Unit)
        .Status = Status
        .Actual.Status = Status
    End With

End Sub


'--------------------------------------------------------------------------
' Advise all Open Forms that Updated Data for Tlu(Unit) is Available...
'--------------------------------------------------------------------------
Public Sub UpdateTluForms(Unit As Integer)

    'This function is called by Background Function when
    'there is an updated unit in Tlu() available
    'our job is to tell the necisary forms to update
    'themseles with the new Tlu(unit) data

    'Tell frmMain that Tlu(Unit) has updates
    Call frmMain.UpdateData(Unit)

    'Tell frmDetail that Tlu(Unit) has updates
    If frmTluDetail.Visible = True Then Call frmTluDetail.UpdateData(Unit)

    'Tell frmWriteReg that Tlu(Unit) has updates
    If frmWriteReg.Visible = True Then Call frmWriteReg.UpdateData(Unit)

End Sub


'--------------------------------------------------------------------------
' Advise all Open Forms that Updated Data for Totalizer(Unit) is Available...
'--------------------------------------------------------------------------
Public Sub UpdateTotalizerForms(Unit As Integer)

    'This function is called by Background Function when
    'there is an updated unit in Totalizer() available
    'our job is to tell the necisary forms to update
    'themseles with the new Totalizer(unit) data

    'Tell frmTotalizer that Totalizer(Unit) has updates
```

```
    Call frmTotalizers.UpdateData(Unit)

End Sub


'--------------------------------------------------------------------------
' Utilty Function
' Hex Dump the Specified String to the Status Bar, Used for Com Debugging
'--------------------------------------------------------------------------
Public Sub HexDump(RXBuf As String)

    'This function Hex Dumps the passed string
    'into the status bar. This is used to show
    'a hexdump of the last recieved packet on
    'the status bar. Should help with debuging
    Dim Text As String
    Dim Cnt As Integer

    If Val(frmMain.sbStatusBar.Tag) > 0 Then
        Text = "RXBuf: "
        For Cnt = 1 To Len(RXBuf)
            Text = Text + Format(Hex(Asc(Mid(RXBuf, Cnt, 1))), "@@ ")
        Next Cnt
        frmMain.sbStatusBar.Panels(1).Text = Text
    End If

End Sub


'--------------------------------------------------------------------------
' This function is used by the Background Task to Yield CPU Time / Cycles
' to both the forground forms in this app, and other apps in the computer
' Without the call to DoEvents, the forground windows would not run
' Without the call to Sleep, we would eat 100% of the available CPU cycles
'--------------------------------------------------------------------------
Public Sub YieldProcessor()

    'Be a good citizen and allow others to use the processor chip
    Call Sleep(5)   'Put this Application to Sleep for 5mS
    DoEvents        'Service the Windows Message Pump

End Sub


'##########################################################################
' Check the specified Trimline for scheduled events:
' Shift Start, Shift End, CsvExport
'##########################################################################
Public Sub CheckTluSchedule(Unit)
    Dim CurrentTime As Date
    Dim S As Date   'Start Checking for Scheduled events here
    Dim E As Date   'End Checking for Sheduled events here
    Dim T As Date   'The date/time we are checking for an event

    Dim rTlu As TLUTYPE         'Reference to the Trimline in Question
    Dim rSch As SCHEDULETYPE    'Reference to the Schedule in Question

    'Retrieve current date/time from the PC Time of Day Clock
    CurrentTime = Now

    'If this Trimline has an Assigned Schedule
    If Tlu(Unit).Schedule > 0 Then

        'Construct a Reference to the Trimline and Schedule we are Operating On
        rTlu = Tlu(Unit)
        rSch = Schedule(rTlu.Schedule)

        'If the last check time is in the future
        'Someone set the PC clock back, we need to jump back in time back to match
        If (rTlu.LastCheck > CurrentTime) Then rTlu.LastCheck = CurrentTime

        'Work out where we need to start checking from...
```

```
        S = rTlu.LastCheck + (1 / 86400#)                   'Start at last check time + 1 second
        If (S < CurrentTime - 1) Then S = CurrentTime - 1   'Unless that was more than 1 day ago

        'Work out how far we are checking too...
        E = S + (3600 / 86400#)                             'End at 1 hour after start
        If (E > CurrentTime) Then E = CurrentTime           'Unless that is in the future

        'Loop through every second between the starting time and the ending time
        'checking the schedule at each second for scheduled events
        For T = S To E Step (1 / 86400#)

            'Step A, Check for Shift End
            If (OpState) Then Call CheckShiftEnd(rTlu, rSch, T)

            'Step B, Check for Shift Start
            If (OpState) Then Call CheckShiftStart(rTlu, rSch, T)

            'Step C, Update the Schedule Last Checked Time
            If (OpState) Then Tlu(Unit).LastCheck = T

        Next T

    Else

        'This Trimline does not have an assigned schedule, so LastCheck is now!
        Tlu(Unit).LastCheck = CurrentTime

    End If

End Sub



'--------------------------------------------------------------------------
' Check the Specified Trimline for a Shift End Export Event
'--------------------------------------------------------------------------
Public Sub CheckShiftEnd(rTlu As TLUTYPE, rSch As SCHEDULETYPE, T As Date)
    Dim i As Integer
    Dim TOD As Date
    Dim DOW As Integer

    'Break the Time we are testing down into time of day, and day of week
    TOD = Format(T, "hh:mm:ss AM/PM")   'Time of Day (00:00:00 to 23:59:59)
    DOW = Weekday(T)                    'Day of Week (1 to 7)

    'Check all 8 shift end times
    For i = LBound(rSch.Shift) To UBound(rSch.Shift)

        'If this is a matching shift end time
        If (TOD = rSch.Shift(i).End) Then

            'if the shift DOES NOT span midnight
            'DayFlag testing is straiht up
            If (rSch.Shift(i).End >= rSch.Shift(i).Beg) Then

                'Check for Day of Week Match
                Select Case (DOW)
                    Case 1: If (rSch.Shift(i).Sun <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 2: If (rSch.Shift(i).Mon <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 3: If (rSch.Shift(i).Tue <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 4: If (rSch.Shift(i).Wed <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 5: If (rSch.Shift(i).Thr <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 6: If (rSch.Shift(i).Fri <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 7: If (rSch.Shift(i).Sat <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                End Select

            'else the shift DOES does span midnight
            'DayFlag testing is shifted by one day
            Else

                'Check for Day of Week Match
                Select Case (DOW)
```

```
                    Case 1: If (rSch.Shift(i).Sat <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 2: If (rSch.Shift(i).Sun <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 3: If (rSch.Shift(i).Mon <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 4: If (rSch.Shift(i).Tue <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 5: If (rSch.Shift(i).Wed <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 6: If (rSch.Shift(i).Thr <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                    Case 7: If (rSch.Shift(i).Fri <> 0) Then Call ShiftEnd(rTlu, rSch, i, T)
                End Select

            End If '(rSCH.Shift(i).End >= rSCH.Shift(i).Beg)

        End If '(TOD = rSCH.Shift(i).End)

    Next i

End Sub


'-------------------------------------------------------------------------
' If the Export option is checked, export the Trimline state to the CSV file
'-------------------------------------------------------------------------
Public Sub ShiftEnd(rTlu As TLUTYPE, rSch As SCHEDULETYPE, i As Integer, T As Date)
    Dim FName As String

    'If we need to export data...
    If (rSch.Shift(i).Exp <> 0) Then

        'Build the File name yyyy_mm_dd.csv where
        'yyyy is the 4D year number
        'mm is the month number 01 to 12
        'dd is the day number 01 to 31
        FName = GetCommonDocumentsFolder() & "\" & Format(T, "YYYY_MM_DD") & ".CSV"

        'Export the Trimline State to CSV
        Call CsvExport(rTlu, FName, T)

        'Debug Log Shift Export Event
        Call LogEvent("ShiftEnd()", rTlu.Name & " Shift End - Exporting Trimline State")

    End If

End Sub


'-------------------------------------------------------------------------
' Export the Specified Trimline State to the Specified CSV File
'-------------------------------------------------------------------------
Public Sub CsvExport(rTlu As TLUTYPE, FileName As String, CurrentTime As Date)

    'Open the file out output
    Open FileName For Append As 1

    'Append the Data Ledgend if the file is zero length
    If (LOF(1) <= 1) Then
        Write #1, "TIMESTAMP", _
                  "NAME", _
                  "ACTUAL"
    End If

    'Write out Trimline state
    Write #1, CurrentTime, _
              rTlu.Name, _
              rTlu.Actual.Value
    Close 1

End Sub


'-------------------------------------------------------------------------
' Check the specified Trimline for a Shift Start Event
'-------------------------------------------------------------------------
Public Sub CheckShiftStart(rTlu As TLUTYPE, rSch As SCHEDULETYPE, T As Date)
```

```
    Dim i As Integer
    Dim TOD As Date
    Dim DOW As Integer

    'Break the Time we are testing down into time of day, and day of week
    TOD = Format(T, "hh:mm:ss AM/PM")   'Time of Day (00:00:00 to 23:59:59)
    DOW = Weekday(T)                    'Day of Week (1 to 7)

    'Check all 8 shift start times
    For i = LBound(rSch.Shift) To UBound(rSch.Shift)

        'If this is a matching shift start time
        If (TOD = rSch.Shift(i).Beg) Then

            'Check for Day of Week Match
            Select Case (DOW)
                Case 1: If (rSch.Shift(i).Sun <> 0) Then Call ShiftStart(rTlu, rSch, i)
                Case 2: If (rSch.Shift(i).Mon <> 0) Then Call ShiftStart(rTlu, rSch, i)
                Case 3: If (rSch.Shift(i).Tue <> 0) Then Call ShiftStart(rTlu, rSch, i)
                Case 4: If (rSch.Shift(i).Wed <> 0) Then Call ShiftStart(rTlu, rSch, i)
                Case 5: If (rSch.Shift(i).Thr <> 0) Then Call ShiftStart(rTlu, rSch, i)
                Case 6: If (rSch.Shift(i).Fri <> 0) Then Call ShiftStart(rTlu, rSch, i)
                Case 7: If (rSch.Shift(i).Sat <> 0) Then Call ShiftStart(rTlu, rSch, i)
            End Select

        End If

    Next i

End Sub


'-------------------------------------------------------------------------
' Enqueue a Shift Start for the Specified Trimline Unit
' If the Reset Option is Checked, Enqueue a Reset
'-------------------------------------------------------------------------
Public Sub ShiftStart(rTlu As TLUTYPE, rSch As SCHEDULETYPE, i As Integer)

    'If this Shift Start Resets the Trimline...
    If (rSch.Shift(i).Rst <> 0) Then

        'Enqueue a Write to Reset the Trimline Unit
        Call rTlu.CmdQueue.Enqueue("WAC0000")    'Reset Actual

        'Debug Log Shift Start Event
        Call LogEvent("ShiftStart()", rTlu.Name & " - Shift Start, Resetting Trimline")

    End If

End Sub


'#########################################################################
' Background Trimline Writes Via Serial Port Communications
' Execute all Enqueued Register Writes to the Specified Trimline
'#########################################################################
Public Sub WriteTluRegsViaCom(Unit As Integer)

    'Process all pending writes before returning to caller
    While Tlu(Unit).CmdQueue.Count > 0
        Call WriteTluRegViaCom(Unit, Tlu(Unit).CmdQueue.Dequeue)
    Wend

End Sub


'-------------------------------------------------------------------------
' Execute the Specified Reg Write to the Specified Trimline Via Serial Communications
'-------------------------------------------------------------------------
Public Sub WriteTluRegViaCom(Unit As Integer, Command As String)
    Dim STX As String
    Dim Addr As String
```

```
    Dim ETX As String
    Dim ACK As String
    Dim NAK As String
    Dim TXBuf As String
    Dim RXBuf As String
    Dim TimeLimit As Single

    'Define strings to make the program more readable
    STX = Chr$(2)                   'CTRL-B, Start of transmition
    Addr = Chr(Tlu(Unit).Address)   'Trimline Address Character
    ETX = Chr$(3)                   'CTRL-C, End of transmition
    ACK = Chr$(6)                   'CTRL-F, Acknowledged
    NAK = Chr$(21)                  'CTRL-U, Not Acknowledged

    'Tell the world were working on this Trimline
    Tlu(Unit).Status = "Writing..."
    Call UpdateTluForms(Unit)

    'Log the Pending Write
    Call LogEvent("WriteTluRegViaCom()", Tlu(Unit).Name & " - Write " & Command)

    'Flush the Receive Buffer
    While frmMain.MSComm1.InBufferCount
        RXBuf = frmMain.MSComm1.Input
    Wend
    RXBuf = ""

    'Transmit the write command to the Trimline
    TXBuf = STX & Addr & Command & ETX      'Build Full command
    frmMain.MSComm1.Output = TXBuf          'Send Command

    'Setup for a 1/5 second timeout
    'Note: Timer is a Single that counts the number of seconds that have
    'elapsed since midnight with about 55mS of accuracy on Win95/98
    'The accuracy of this timer may vary on different operating systems
    'For example, I believe that it's accurate to about 10mS on WinNT/2K
    TimeLimit = Timer + 0.2

    'Loop wating for an ACK, NAK or Timeout condition
    While (Timer < TimeLimit And InStr(RXBuf, ACK) = 0 And InStr(RXBuf, NAK) = 0)

        'Be nice and yield to the main loop
        'While we wait for a responce from the Trimline
        Call YieldProcessor

        'Accumulate characters recieved via RS-485 in RxBuf
        'Note: Depending on the RS-485 converter used we
        'we may recieve our own transmition as an echo
        If frmMain.MSComm1.InBufferCount Then RXBuf = RXBuf + frmMain.MSComm1.Input

    Wend

    'HexDump the RXBuf to the status bar
    Call HexDump(RXBuf)

    'Now we need to deal with the ACK, NAK, or TimeOut Condition
    If InStr(RXBuf, ACK) > 0 Then

        'We have an ACK
        Tlu(Unit).Status = "Done"
        Call UpdateTluForms(Unit)
        Call LogEvent("WriteTluRegViaCom()", Tlu(Unit).Name & " - Write ACKed")

    ElseIf InStr(RXBuf, NAK) Then

        'Trimline responed with a NAK
        Tlu(Unit).Status = "NAK"
        Call UpdateTluForms(Unit)
        Call EnqueueMessage(Tlu(Unit).Name, "Trimline Rejected Command!", Command)
        Call LogEvent("WriteTluRegViaCom()", Tlu(Unit).Name & " - Write NAKed")

    Else
```

```
        'Timeout
        Tlu(Unit).Status = "TIMEOUT"
        Call UpdateTluForms(Unit)
        Call EnqueueMessage(Tlu(Unit).Name, "Timed Out Waiting for Response!", Command)
        Call LogEvent("WriteTluRegViaCom()", Tlu(Unit).Name & " - Write Timed Out")

    End If

End Sub


'##############################################################################
' Background Trimline Writes Via TCP/IP Communications
' Execute all Enqueued Register Writes to the Specified Trimline
'##############################################################################
Public Sub WriteTluRegsViaTcp(Unit As Integer)

    'Process all pending writes before returning to caller
    While Tlu(Unit).CmdQueue.Count > 0
        Call WriteTluRegViaTcp(Unit, Tlu(Unit).CmdQueue.Dequeue)
    Wend

End Sub


'------------------------------------------------------------------------------
' Execute the Specified Reg Write to the Specified Trimline Via TCP Communications
'------------------------------------------------------------------------------
Public Sub WriteTluRegViaTcp(Unit As Integer, Command As String)
    Dim STX As String
    Dim Addr As String
    Dim ETX As String
    Dim ACK As String
    Dim NAK As String
    Dim TXBuf As String
    Dim RXBuf As String
    Dim Temp As String
    Dim TimeLimit As Single

    'Define strings to make the program more readable
    STX = Chr$(2)                   'CTRL-B, Start of transmition
    Addr = Chr(Tlu(Unit).Address)   'Trimline Address Character
    Addr = "A"
    ETX = Chr$(3)                   'CTRL-C, End of transmition
    ACK = Chr$(6)                   'CTRL-F, Acknowledged
    NAK = Chr$(21)                  'CTRL-U, Not Acknowledged

    'Tell the world were working on this Trimline
    Tlu(Unit).Status = "Writing..."
    Call UpdateTluForms(Unit)

    'Log the Pending Write
    Call LogEvent("WriteTluRegViaTcp()", Tlu(Unit).Name & " - Write " & Command)

    'Flush the Receive Buffer
    While frmMain.Winsock1(Unit).BytesReceived > 0
        Call frmMain.Winsock1(Unit).GetData(Temp)
    Wend

    'Transmit the write command to the Trimline
    TXBuf = STX & Addr & Command & ETX          'Build Full command
    Call frmMain.Winsock1(Unit).SendData(TXBuf)  'Send command

    'Setup for a 1/2 second timeout
    'Note: Timer is a Single that counts the number of seconds that have
    'elapsed since midnight with about 55mS of accuracy on Win95/98
    'The accuracy of this timer may vary on different operating systems
    'For example, I believe that it's accurate to about 10mS on WinNT/2K
    TimeLimit = Timer + 0.5

    'Loop wating for an ACK, NAK or Timeout condition
```

```
    While (Timer < TimeLimit And InStr(RXBuf, ACK) = 0 And InStr(RXBuf, NAK) = 0)

        'Be nice and yield to the main loop
        'While we wait for a responce from the Trimline
        Call YieldProcessor

        'Accumulate characters recieved via TCP/IP in RxBuf
        If (frmMain.Winsock1(Unit).BytesReceived > 0) Then
            Call frmMain.Winsock1(Unit).GetData(Temp)
            RXBuf = RXBuf + Temp
        End If

    Wend

    'HexDump the RXBuf to the status bar
    Call HexDump(RXBuf)

    'Now we need to deal with the ACK, NAK, or TimeOut Condition
    If InStr(RXBuf, ACK) > 0 Then

        'We have an ACK
        Tlu(Unit).Status = "Done"
        Call UpdateTluForms(Unit)
        Call LogEvent("WriteTluRegViaTcp()", Tlu(Unit).Name & " - Write ACKed")

    ElseIf InStr(RXBuf, NAK) Then

        'Trimline responed with a NAK
        Tlu(Unit).Status = "NAK"
        Call UpdateTluForms(Unit)
        Call EnqueueMessage(Tlu(Unit).Name, "Trimline Rejected Command!", Command)
        Call LogEvent("WriteTluRegViaTcp()", Tlu(Unit).Name & " - Write NAKed")

    Else

        'Timeout
        Tlu(Unit).Status = "TIMEOUT"
        Call UpdateTluForms(Unit)
        Call EnqueueMessage(Tlu(Unit).Name, "Timed Out Waiting for Response!", Command)
        Call LogEvent("WriteTluRegViaTcp()", Tlu(Unit).Name & " - Write Timed Out")

    End If

End Sub


'#########################################################################
' Format Numeric Data for a Trimline Register Write Command
'#########################################################################
Public Function RegVal6Digit(Number As String) As String
    Dim Num As Long

    'Interpret String Input as a Number
    If IsNumeric(Number) Then
        Num = Val(Number)
    Else
        Num = 0
    End If

    'Clamp result to allowed range
    If (Num < 0) Then Num = 0
    If (Num > 999999) Then Num = 999999

    'Convert Back to String in RegVal Format and return to caller
    'The 6 Digit PPTs accept both 6 digit and 4 digit RegVals without issue
    'The 4 Digit PPTs will ignore the high 2 digits in a 6 Digit RegVal
    'but the AF-26xx-xxx series reqires the RegVal to be exactly 4 digits
    'So if Num fits in 4 digits, we format it to 4 digits for maximum compatibility
    'If not, we format for 6 digits as a 2nd best option. On 6 Digit PPTs, no problem
    'on 4 Digit PPTs, the high 2 digits will be ignored.  On AF-26xx-xxx the write
    'will be rejected as invalid.
    If (Num <= 9999) Then
```

```
        RegVal6Digit = Format(Num, "0000")
    Else
        RegVal6Digit = Format(Num, "000000")
    End If

End Function


'#############################################################################
' Background Trimline Reads via Serial Communications
' Retrieve the Specified Trimlines State
'#############################################################################
Public Sub ReadTluRegsViaCom(Unit As Integer)

    'Read back the Trimline State
    Call ReadTluRegViaCom(Unit, Tlu(Unit).Actual, "AC")

End Sub


'-----------------------------------------------------------------------------
' Read the Specified Register from the Specified Trimline via Serial Communications
'-----------------------------------------------------------------------------
Public Sub ReadTluRegViaCom(Unit As Integer, Reg As TLUREG, RegID As String)
    Dim STX As String
    Dim Addr As String
    Dim ETX As String
    Dim ACK As String
    Dim NAK As String
    Dim RXBuf As String
    Dim TXBuf As String
    Dim Temp  As String
    Dim TimeLimit As Single

    'Define strings to make the program more readable
    STX = Chr$(2)                   'CTRL-B, Start of transmition
    Addr = Chr(Tlu(Unit).Address)   'Trimline Address Character
    ETX = Chr$(3)                   'CTRL-C, End of transmition
    ACK = Chr$(6)                   'CTRL-F, Acknowledged
    NAK = Chr$(21)                  'CTRL-U, Not Acknowledged

    'Tell the world were working on this Trimline
    Tlu(Unit).Status = "Reading..."
    Reg.Status = "Read Pending..."
    Call UpdateTluForms(Unit)

    'Flush the Receive Buffer
    While frmMain.MSComm1.InBufferCount
        RXBuf = frmMain.MSComm1.Input
    Wend
    RXBuf = ""

    'Transmit the read command to the Trimline
    TXBuf = STX & Addr & "R" & RegID & ETX
    frmMain.MSComm1.Output = TXBuf

    'Setup for a 1/5 second timeout
    'Note: Timer is a Single that counts the number of seconds that have
    'elapsed since midnight with about 55mS of accuracy on Win95/98
    'The accuracy of this timer may vary on different operating systems
    'For example, I believe that it's accurate to about 10mS on WinNT/2K
    TimeLimit = Timer + 0.2

    'Loop wating for an ACK, NAK or Timeout condition
    While (Timer < TimeLimit And InStr(RXBuf, ACK) = 0 And InStr(RXBuf, NAK) = 0)

        'Be nice and yield to the main loop
        'While we wait for a responce from the Trimline
        Call YieldProcessor

        'Accumulate characters recieved via RS-485 in RxBuf
        'Note: Depending on the RS-485 converter used we
```

```
        'we may recieve our own transmition as an echo
        If frmMain.MSComm1.InBufferCount Then RXBuf = RXBuf + frmMain.MSComm1.Input

    Wend

    'HexDump the RXBuf to the status bar
    Call HexDump(RXBuf)

    'Now we need to deal with the ACK, NAK, or TimeOut Condition
    If InStr(RXBuf, ACK) > 0 Then

        'We have an ACK Packet, Extract the Register Value
        Temp = RXBuf
        If InStr(Temp, ETX) Then Temp = Mid(Temp, InStr(Temp, ETX) + 1)
        Reg.Value = Val(Temp)
        Reg.Status = "Done"
        Tlu(Unit).Status = "Done"
        Call UpdateTluForms(Unit)
        'Call LogEvent("ReadTluRegViaCom()", Tlu(Unit).Name & " - Read ACKed")

    ElseIf InStr(RXBuf, NAK) Then

        'Trimline responed with a NAK
        Reg.Value = 0
        Reg.Status = "NAK"
        Tlu(Unit).Status = "NAK"
        Call UpdateTluForms(Unit)
        'Call LogEvent("ReadTluRegViaCom()", Tlu(Unit).Name & " - Read NAKed")

    Else

        'Timeout
        Reg.Status = "TIMEOUT"
        Tlu(Unit).Status = "TIMEOUT"
        Call UpdateTluForms(Unit)
        'Call LogEvent("ReadTluRegViaCom()", Tlu(Unit).Name & " - Read Timed Out")

    End If

End Sub


'#########################################################################
' Background Trimline Reads via TCP/IP Communications
' Retrieve the Specified Trimline's State
'#########################################################################
Public Sub ReadTluRegsViaTcp(Unit As Integer)

    'Read back the Trimline State
    Call ReadTluRegViaTcp(Unit, Tlu(Unit).Actual, "AC")

End Sub


'-------------------------------------------------------------------------
' Read the Specified Register from the Specified Trimline via TCP/IP Communications
'-------------------------------------------------------------------------
Public Sub ReadTluRegViaTcp(Unit As Integer, Reg As TLUREG, RegID As String)
    Dim STX As String
    Dim Addr As String
    Dim ETX As String
    Dim ACK As String
    Dim NAK As String
    Dim RXBuf As String
    Dim TXBuf As String
    Dim Temp  As String
    Dim TimeLimit As Single

    'Define strings to make the program more readable
    STX = Chr$(2)                   'CTRL-B, Start of transmition
    Addr = Chr(Tlu(Unit).Address)   'Trimline Address Character
    Addr = "A"
```

```
    ETX = Chr$(3)                   'CTRL-C, End of transmition
    ACK = Chr$(6)                   'CTRL-F, Acknowledged
    NAK = Chr$(21)                  'CTRL-U, Not Acknowledged

    'Tell the world were working on this Trimline
    Tlu(Unit).Status = "Reading..."
    Reg.Status = "Read Pending..."
    Call UpdateTluForms(Unit)

    'Flush the Receive Buffer
    While frmMain.Winsock1(Unit).BytesReceived > 0
        Call frmMain.Winsock1(Unit).GetData(Temp)
    Wend

    'Transmit the read command to the Trimline
    TXBuf = STX & Addr & "R" & RegID & ETX
    Call frmMain.Winsock1(Unit).SendData(TXBuf)

    'Setup for a 1/2 second timeout
    'Note: Timer is a Single that counts the number of seconds that have
    'elapsed since midnight with about 55mS of accuracy on Win95/98
    'The accuracy of this timer may vary on different operating systems
    'For example, I believe that it's accurate to about 10mS on WinNT/2K
    TimeLimit = Timer + 0.5

    'Loop wating for an ACK, NAK or Timeout condition
    While (Timer < TimeLimit And InStr(RXBuf, ACK) = 0 And InStr(RXBuf, NAK) = 0)

        'Be nice and yield to the main loop
        'While we wait for a responce from the Trimline
        Call YieldProcessor

        'Accumulate characters recieved via TCP/IP in RxBuf
        If (frmMain.Winsock1(Unit).BytesReceived > 0) Then
            Call frmMain.Winsock1(Unit).GetData(Temp)
            RXBuf = RXBuf + Temp
        End If

    Wend

    'HexDump the RXBuf to the status bar
    Call HexDump(RXBuf)

    'Now we need to deal with the ACK, NAK, or TimeOut Condition
    If InStr(RXBuf, ACK) > 0 Then

        'We have an ACK Packet, Extract the Register Value
        Temp = RXBuf
        If InStr(Temp, ETX) Then Temp = Mid(Temp, InStr(Temp, ETX) + 1)
        Reg.Value = Val(Temp)
        Reg.Status = "Done"
        Tlu(Unit).Status = "Done"
        Call UpdateTluForms(Unit)
        'Call LogEvent("ReadTluRegViaTcp()", Tlu(Unit).Name & " - Read ACKed")

    ElseIf InStr(RXBuf, NAK) Then

        'Trimline responed with a NAK
        Reg.Value = 0
        Reg.Status = "NAK"
        Tlu(Unit).Status = "NAK"
        Call UpdateTluForms(Unit)
        'Call LogEvent("ReadTluRegViaTcp()", Tlu(Unit).Name & " - Read NAKed")

    Else

        'Timeout
        Reg.Status = "TIMEOUT"
        Tlu(Unit).Status = "TIMEOUT"
        Call UpdateTluForms(Unit)
        'Call LogEvent("ReadTluRegViaTcp()", Tlu(Unit).Name & " - Read Timed Out")
```

```
    End If

End Sub


'#############################################################################
' Utility Functions
' Load System Configuration from Disk
'#############################################################################
Public Sub LoadAllConfiguration()
    Dim Unit As Integer

    'Display the Progress Dialog
    frmProgress.Caption = App.Title & " Loading Configuration Data..."
    frmProgress.lAppName = App.Title
    frmProgress.lOperation = "Loading Configuration Data..."
    frmProgress.ProgressBar.Min = 1
    frmProgress.ProgressBar.Max = UBound(Tlu) + UBound(Totalizer) + UBound(Schedule)
    frmProgress.Show
    Call CenterWindow(frmProgress)

    'Load all Trimline Units Configuration
    For Unit = LBound(Tlu) To UBound(Tlu)
        frmProgress.ProgressBar.Value = Unit
        Call LoadTluConfiguration(Unit)
        Call YieldProcessor
    Next Unit

    'Load all Totalizers Configuration
    For Unit = LBound(Totalizer) To UBound(Totalizer)
        frmProgress.ProgressBar.Value = Unit + UBound(Tlu)
        Call LoadTotalizerConfiguration(Unit)
        Call YieldProcessor
    Next Unit

    'Load all Schedules Configuration
    For Unit = LBound(Schedule) To UBound(Schedule)
        frmProgress.ProgressBar.Value = Unit + UBound(Tlu) + UBound(Totalizer)
        Call LoadScheduleConfiguration(Unit)
        Call YieldProcessor
    Next Unit

    'Load the Secret Word
    Call LoadSecretWord

    'Load the Debug Log Flag
    Call LoadDebugFlag

    'All Done, Dismiss the Progress Dialog
    Call Unload(frmProgress)

End Sub


'-----------------------------------------------------------------------------
' Load One Trimline's Configuration from the Disk File
'-----------------------------------------------------------------------------
Public Sub LoadTluConfiguration(Unit As Integer)
    Dim File As String
    Dim DiskRec As TLUTYPEDISK

    'Work around Null Padding Issue
    DiskRec.Name = ""   'Will Fill String with Spaces

    'Get Trimline Unit Record from Disk
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\UNIT.DAT"
    Open File For Random As #1 Len = Len(DiskRec)
    Get #1, Unit, DiskRec
    Close #1
```

```
    With Tlu(Unit)

        'Copy Disk Record to Ram Record
        .Name = Trim(DiskRec.Name)
        .Port = DiskRec.Port
        .Address = DiskRec.Address
        .HostName = Trim(DiskRec.HostName)
        .Schedule = DiskRec.Schedule
        .Actual.Value = DiskRec.Actual.Value
        .Actual.Status = Trim(DiskRec.Actual.Status)
        .LastCheck = DiskRec.LastCheck
        Set .CmdQueue = New Queue

        'Data Validation
        If Len(.Name) = 0 Or Asc(.Name) = 0 Then .Name = "UNIT" + Str(Unit)
        If .Port < 0 Then .Port = 0
        If .Port > 16 Then .Port = 0
        If .Address < 33 Then .Address = 0
        If .Address > 126 Then .Address = 0
        If Len(.HostName) = 0 Or Asc(.HostName) = 0 Then .HostName = "(none)"
        If .Schedule < 0 Then .Schedule = 0
        If .Schedule > UBound(Schedule) Then .Schedule = 0

    End With 'Tlu(Unit)

End Sub


'---------------------------------------------------------------------------
' Load One Totalizers Configuration from the Disk File
'---------------------------------------------------------------------------
Public Sub LoadTotalizerConfiguration(Unit As Integer)
    Dim File As String
    Dim DiskRec As TOTALIZERTYPEDISK
    Dim i As Integer

    'Get Totalizer Unit Record from Disk
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\TOTALIZER.DAT"
    Open File For Random As #1 Len = Len(DiskRec)
    Get #1, Unit, DiskRec
    Close #1

    With Totalizer(Unit)

        'Copy Disk Record to Ram Record
        .Name = Trim(DiskRec.Name)
        .Port = DiskRec.Port
        .HostName = Trim(DiskRec.HostName)
        .ActualAddress = DiskRec.ActualAddress
        For i = LBound(.TluIsMember) To UBound(.TluIsMember)
            .TluIsMember(i) = DiskRec.TluIsMember(i)
        Next i
        .TotalActual = DiskRec.TotalActual
        .Status = Trim(DiskRec.Status)

        'Data Validation
        If Len(.Name) = 0 Or Asc(.Name) = 0 Then .Name = "Totalizer" + Str(Unit)
        If .Port < 0 Then .Port = 0
        If .Port > 16 Then .Port = 0
        If Len(.HostName) = 0 Or Asc(.HostName) = 0 Then .HostName = "(none)"
        If .ActualAddress < 33 Then .ActualAddress = 0
        If .ActualAddress > 126 Then .ActualAddress = 0

    End With 'Totalizer(Unit)

End Sub


'---------------------------------------------------------------------------
' Load One Named Schedule from the Disk File
```

```
'--------------------------------------------------------------------------
Public Sub LoadScheduleConfiguration(Unit As Integer)
    Dim File As String
    Dim DiskRec As SCHEDULETYPEDISK
    Dim i As Integer

    'Get Schedule Record from Disk
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\SCHEDULE.DAT"
    Open File For Random As #1 Len = Len(DiskRec)
    Get #1, Unit, DiskRec
    Close #1

    With Schedule(Unit)

        'Copy Disk Record to Ram Record
        .Name = Trim(DiskRec.Name)
        For i = LBound(.Shift) To UBound(.Shift)
            .Shift(i).Rst = DiskRec.Shift(i).Rst
            .Shift(i).Exp = DiskRec.Shift(i).Exp
            .Shift(i).Sun = DiskRec.Shift(i).Sun
            .Shift(i).Mon = DiskRec.Shift(i).Mon
            .Shift(i).Tue = DiskRec.Shift(i).Tue
            .Shift(i).Wed = DiskRec.Shift(i).Wed
            .Shift(i).Thr = DiskRec.Shift(i).Thr
            .Shift(i).Fri = DiskRec.Shift(i).Fri
            .Shift(i).Sat = DiskRec.Shift(i).Sat
            .Shift(i).Beg = DiskRec.Shift(i).Beg
            .Shift(i).End = DiskRec.Shift(i).End
        Next i

        'Data Validation
        If Len(.Name) = 0 Or Asc(.Name) = 0 Then .Name = "Schedule" + Str(Unit)

    End With 'Schedule(Unit)

End Sub


'--------------------------------------------------------------------------
' Load the Privileged Operation Password from the Ini File
'--------------------------------------------------------------------------
Public Sub LoadSecretWord()
    Dim File As String

    'Compute Ini File Name
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\SETTINGS.INI"

    'Get the Privileged Operation Password from the Ini file
    SecretWord = ROT13(ReadIni(File, "SecretWord", "SecretWord", ""))

End Sub


'--------------------------------------------------------------------------
' Load the Debug Flag from the Ini File
'--------------------------------------------------------------------------
Public Sub LoadDebugFlag()
    Dim File As String

    'Compute Ini File Name
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\SETTINGS.INI"

    'Load Debug Event Logging Setting from the Ini file
    If (Val(ReadIni(File, "DebugFlag", "DebugFlag", "0")) <> 0) Then
        frmMain.mnuDebugEventLog.Checked = True
        DebugFlag = True
```

```
    Else
        frmMain.mnuDebugEventLog.Checked = False
        DebugFlag = False
    End If

End Sub


'#############################################################################
' Utility Functions
' Save System Configuration to Disk
'#############################################################################
Public Sub SaveAllConfiguration()
    Dim Unit As Integer

    'Display the Progress Dialog
    frmProgress.Caption = App.Title & " Saving Configuration Data..."
    frmProgress.lAppName = App.Title
    frmProgress.lOperation = "Saving Configuration Data..."
    frmProgress.ProgressBar.Min = 1
    frmProgress.ProgressBar.Max = UBound(Tlu) + UBound(Totalizer) + UBound(Schedule)
    frmProgress.Show
    Call CenterWindow(frmProgress)

    'Save all Trimline's Configuration
    For Unit = LBound(Tlu) To UBound(Tlu)
        frmProgress.ProgressBar.Value = Unit
        Call SaveTluConfiguration(Unit)
        Call YieldProcessor
    Next Unit

    'Save all Totalizers Configuration
    For Unit = LBound(Totalizer) To UBound(Totalizer)
        frmProgress.ProgressBar.Value = Unit + UBound(Tlu)
        Call SaveTotalizerConfiguration(Unit)
        Call YieldProcessor
    Next Unit

    'Save all Schedules Configuration
    For Unit = LBound(Schedule) To UBound(Schedule)
        frmProgress.ProgressBar.Value = Unit + UBound(Tlu) + UBound(Totalizer)
        Call SaveScheduleConfiguration(Unit)
        Call YieldProcessor
    Next Unit

    'Save the Secret Word
    Call SaveSecretWord

    'Save the Debug Log Flag
    Call SaveDebugFlag

    'All Done, Dismiss the Progress Dialog
    Call Unload(frmProgress)

End Sub


'-----------------------------------------------------------------------------
' Save One Trimline's Configuration to Disk
'-----------------------------------------------------------------------------
Public Sub SaveTluConfiguration(Unit As Integer)
    Dim File As String
    Dim DiskRec As TLUTYPEDISK

    With Tlu(Unit)

        'Copy Ram Record to Disk Record
        DiskRec.Name = .Name
        DiskRec.Port = .Port
        DiskRec.Address = .Address
        DiskRec.HostName = .HostName
        DiskRec.Schedule = .Schedule
```

```
        DiskRec.Actual.Value = .Actual.Value
        DiskRec.Actual.Status = .Actual.Status
        DiskRec.LastCheck = .LastCheck

    End With 'Tlu(Unit)

    'Put Trimline Unit Record onto Disk
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\UNIT.DAT"
    Open File For Random As #1 Len = Len(DiskRec)
    Put #1, Unit, DiskRec
    Close #1

End Sub


'-------------------------------------------------------------------------
' Save One Totalizer Configuration to Disk
'-------------------------------------------------------------------------
Public Sub SaveTotalizerConfiguration(Unit As Integer)
    Dim File As String
    Dim DiskRec As TOTALIZERTYPEDISK
    Dim i As Integer

    With Totalizer(Unit)

        'Copy Ram Record to Disk Record
        DiskRec.Name = .Name
        DiskRec.Port = .Port
        DiskRec.HostName = .HostName
        DiskRec.ActualAddress = .ActualAddress
        For i = LBound(.TluIsMember) To UBound(.TluIsMember)
            DiskRec.TluIsMember(i) = .TluIsMember(i)
        Next i
        DiskRec.TotalActual = .TotalActual
        DiskRec.Status = .Status

    End With 'Totalizer(Unit)

    'Put Totalizer Record onto Disk
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\TOTALIZER.DAT"
    Open File For Random As #1 Len = Len(DiskRec)
    Put #1, Unit, DiskRec
    Close #1

End Sub


'-------------------------------------------------------------------------
' Save One Named Schedule to Disk
'-------------------------------------------------------------------------
Public Sub SaveScheduleConfiguration(Unit As Integer)
    Dim File As String
    Dim DiskRec As SCHEDULETYPEDISK
    Dim i As Integer

    With Schedule(Unit)

        'Copy RAM Record to Disk Record
        DiskRec.Name = Trim(.Name)
        For i = LBound(.Shift) To UBound(.Shift)
            DiskRec.Shift(i).Rst = .Shift(i).Rst
            DiskRec.Shift(i).Exp = .Shift(i).Exp
            DiskRec.Shift(i).Sun = .Shift(i).Sun
            DiskRec.Shift(i).Mon = .Shift(i).Mon
            DiskRec.Shift(i).Tue = .Shift(i).Tue
            DiskRec.Shift(i).Wed = .Shift(i).Wed
            DiskRec.Shift(i).Thr = .Shift(i).Thr
            DiskRec.Shift(i).Fri = .Shift(i).Fri
```

```
            DiskRec.Shift(i).Sat = .Shift(i).Sat
            DiskRec.Shift(i).Beg = .Shift(i).Beg
            DiskRec.Shift(i).End = .Shift(i).End
        Next i

    End With 'Schedule(Unit)

    'Put Schedule Record onto Disk
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\SCHEDULE.DAT"
    Open File For Random As #1 Len = Len(DiskRec)
    Put #1, Unit, DiskRec
    Close #1

End Sub


'-----------------------------------------------------------------------------
' Save the Priviledge Operation Password to the Ini File
'-----------------------------------------------------------------------------
Public Sub SaveSecretWord()
    Dim File As String

    'Compute Ini File Name
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\SETTINGS.INI"

    'Save Priviledge Operation Password to the Registery
    Call WriteIni(File, "SecretWord", "SecretWord", ROT13(SecretWord))

End Sub


'-----------------------------------------------------------------------------
' Save the Debug Flag to the Ini File
'-----------------------------------------------------------------------------
Public Sub SaveDebugFlag()
    Dim File As String

    'Compute Ini File Name
    File = GetCommonDocumentsFolder()
    Call MakePath(File)
    File = File & "\SETTINGS.INI"

    'Save the Debug Event Logging Setting to the Ini File
    If (DebugFlag <> False) Then
        Call WriteIni(File, "DebugFlag", "DebugFlag", "1")
    Else
        Call WriteIni(File, "DebugFlag", "DebugFlag", "0")
    End If

End Sub


'#############################################################################
' Utilty Function
' Clear all Data for the Specified Trimline  (Reset Trimline Config to Unused State)
'#############################################################################
Public Sub EraseTluConfiguration(Unit As Integer)

    Dim i As Integer

    'Set Tlu(Unit) to the **UNUSED ** state
    With Tlu(Unit)

        'Erase Trimline Config
        .Name = "UNIT" + Str(Unit)
        .Port = 0
        .Address = 0
        .HostName = "(none)"
```

```
        .Schedule = 0

        'Erase Register Data
        Call EraseReg(.Actual)

        'Erase LastCheck time and Status
        .Status = "(none)"
        .LastCheck = 0

        'Erase the Command Queue
        If (.CmdQueue.Count > 0) Then .CmdQueue.Clear

    End With

End Sub


'---------------------------------------------------------------------------
' Erase a Trimline Register
'---------------------------------------------------------------------------
Public Sub EraseReg(Reg As TLUREG)

    'Erase Register Settings
    With Reg
        .Value = 0
        .Status = "(none)"
    End With 'Reg

End Sub


'###########################################################################
' Utilty Function
' Clear all Data for the Specified Totalizer (Reset Totalizer Config to Unused State)
'###########################################################################
Public Sub EraseTotalizerConfiguration(Unit As Integer)
    Dim i As Integer

    'Set this Totalizer to the **UNUSED** State
    With Totalizer(Unit)
        .Name = "Totalizer" + Str(Unit)
        .Port = 0
        .HostName = "(none)"
        .ActualAddress = 0
        .Status = "(none)"
        For i = LBound(.TluIsMember) To UBound(.TluIsMember)
            .TluIsMember(i) = 0
        Next i

    End With

End Sub


'###########################################################################
' Utility Functions
' Format Data for Display on the PC Screen
'###########################################################################
'---------------------------------------------------------------------------
' Format Numeric Data for Display on the PC Screen
'---------------------------------------------------------------------------
Public Function fmtNumber(Number As Long) As String
    Dim Num As Long

    'Format a Number for Display
    Num = Number
    If (Num < -9999999) Then Num = -9999999
    If (Num > 99999999) Then Num = 99999999
    fmtNumber = Format(Format(Num, "#######0"), "@@@@@@@@")

End Function
```

```
'--------------------------------------------------------------------------
' Format Comport Data for Display on the PC Screen
'--------------------------------------------------------------------------
Public Function fmtPort(Port As Integer) As String

    If (Port = 0) Then
        fmtPort = "(none)"
    ElseIf (Port >= 1 And Port <= 15) Then
        fmtPort = "Com" + Trim(Str(Port)) + ":"
    ElseIf (Port = 16) Then
        fmtPort = "TCP/IP"
    Else
        fmtPort = "(unknown)"
    End If

End Function


'--------------------------------------------------------------------------
' Format Address Data for Display on the PC Screen
'--------------------------------------------------------------------------
Public Function fmtAddr(Addr As Integer) As String

    If (Addr >= 33 And Addr <= 126) Then
        fmtAddr = Chr$(Addr) + " (" + Hex(Addr) + "h)"
    Else
        fmtAddr = "(none)"
    End If

End Function


'--------------------------------------------------------------------------
' Format Schedule Selection for Display on the PC Screen
'--------------------------------------------------------------------------
Public Function fmtSchedule(Unit As Integer) As String

    If (Unit = 0) Then
        fmtSchedule = "(none)"
    Else
        fmtSchedule = "#" & Format(Unit, "0) ") & Schedule(Unit).Name
    End If

End Function


'##########################################################################
' Misc Utility Functions
' Yada Yada Yada...
'##########################################################################
'--------------------------------------------------------------------------
' Utiltity to center a dialog over the main form
'--------------------------------------------------------------------------
Public Sub CenterWindow(Frm As Form)

    Frm.Left = frmMain.Left + (frmMain.Width / 2) - (Frm.Width / 2)
    Frm.Top = frmMain.Top + (frmMain.Height / 2) - (Frm.Height / 2)
    If Frm.Left > Screen.Width - Frm.Width Then Frm.Left = Screen.Width - Frm.Width
    If Frm.Left < 0 Then Frm.Left = 0
    If Frm.Top > Screen.Height - Frm.Height Then Frm.Top = Screen.Height - Frm.Height
    If Frm.Top < 0 Then Frm.Top = 0

End Sub


'--------------------------------------------------------------------------
' Event Log Facility for Debugging
'--------------------------------------------------------------------------
Public Sub LogEvent(Func As String, What As String)
    Dim File As String
    Dim FmtStamp As String
    Dim FmtComp As String
```

```
    Dim FmtUser As String
    Dim FmtFunc As String
    Dim FmtWhat As String

    'If the debug event log is turnned on
    If (DebugFlag = True) Then

        'Construct the Event Log Line
        FmtStamp = Format(Now, "mm/dd/yy Hh:Nn:Ss ")
        FmtComp = Format(Environ$("computername"), "!@@@@@@@@@@@@@@@ ")
        FmtUser = Format(Environ$("username"), "!@@@@@@@@@@@@@@@ ")
        FmtFunc = Format(Func, "!@@@@@@@@@@@@@@@@ ")
        FmtWhat = Left(What, 80)

        'Construct Path to Log file
        File = GetCommonDocumentsFolder()
        Call MakePath(File)
        File = File & "\DebugLog.Txt"

        'Append the Log Line to the Log File
        Open File For Append As #1
        Print #1, FmtStamp; FmtComp; FmtUser; FmtFunc; FmtWhat
        Close #1

    End If

End Sub


'--------------------------------------------------------------------------
' Enqueue a Message to the PC Operator.
' The Main form will Display the Message at the Next Available Time
'--------------------------------------------------------------------------
Public Sub EnqueueMessage(L1 As String, L2 As String, L3 As String)

    'Place Message in Message Queue
    MsgQueue.Enqueue L1 & vbCrLf & L2 & vbCrLf & L3

End Sub


'--------------------------------------------------------------------------
' Perform a Password Check, Return True for Password Good, False for Failure
'--------------------------------------------------------------------------
Public Function CheckPassword(CallingForm As Form) As Boolean

    'If Password Checking is Enabled
    If (SecretWord <> "") Then

        'Summon the Password Check Dialog
        frmPassword.Show vbModal, CallingForm

        'Check Result
        If ((frmPassword.tbPassword = SecretWord) Or (frmPassword.tbPassword = "none")) Then

            'Password Match, Return True
            CheckPassword = True
            Unload frmPassword

        Else

            'Password Fail, Return False
            CheckPassword = False
            Call MsgBox("Incorect Password!", vbCritical + vbOKOnly, App.Title)
            Unload frmPassword

        End If

    Else

        'Password Checking is Disabled
        CheckPassword = True
```

```
    End If

End Function


'---------------------------------------------------------------------------
' Save a window position / size to the windows registry
'---------------------------------------------------------------------------
Public Sub SaveWindowPosition(Frm As Form)

    If Frm.WindowState <> vbMinimized Then

        'Save the forms position
        Call SaveSetting(App.Title, Frm.Name, "Left", Frm.Left)
        Call SaveSetting(App.Title, Frm.Name, "Top", Frm.Top)

        'If the form is resizable, save the size as well
        If Frm.BorderStyle = vbSizable Or Frm.BorderStyle = vbSizableToolWindow Then
            Call SaveSetting(App.Title, Frm.Name, "Width", Frm.Width)
            Call SaveSetting(App.Title, Frm.Name, "Height", Frm.Height)
        End If

    End If

End Sub


'---------------------------------------------------------------------------
'Load a window position / size from the windows registry
'---------------------------------------------------------------------------
Public Sub LoadWindowPosition(Frm As Form)

    'Load the forms position
    Frm.Left = GetSetting(App.Title, Frm.Name, "Left", Frm.Left)
    Frm.Top = GetSetting(App.Title, Frm.Name, "Top", Frm.Top)

    'If the form is resizable, load the size as well
    If Frm.BorderStyle = vbSizable Or Frm.BorderStyle = vbSizableToolWindow Then
        Frm.Width = GetSetting(App.Title, Frm.Name, "Width", Frm.Width)
        Frm.Height = GetSetting(App.Title, Frm.Name, "Height", Frm.Height)
    End If

    'Sanity check position / size
    If Frm.Left > Screen.Width - Frm.Width Then Frm.Left = Screen.Width - Frm.Width
    If Frm.Left < 0 Then Frm.Left = 0
    If Frm.Top > Screen.Height - Frm.Height Then Frm.Top = Screen.Height - Frm.Height
    If Frm.Top < 0 Then Frm.Top = 0

End Sub


'---------------------------------------------------------------------------
'Utility to Decode a Winsock State into Descriptive Text
'---------------------------------------------------------------------------
Public Function DecodeWinsockState(State As Integer) As String

    Select Case (State)
        Case sckClosed
            DecodeWinsockState = "Socket Closed"
        Case sckOpen
            DecodeWinsockState = "Socket Open"
        Case sckListening
            DecodeWinsockState = "Socket Listening..."
        Case sckConnectionPending
            DecodeWinsockState = "Socket Connection Pending..."
        Case sckResolvingHost
            DecodeWinsockState = "Socket Resolving Host..."
        Case sckHostResolved
            DecodeWinsockState = "Socket Host Resolved"
        Case sckConnecting
            DecodeWinsockState = "Socket Connecting..."
```

```
        Case sckConnected
            DecodeWinsockState = "Socket Connected"
        Case sckClosing
            DecodeWinsockState = "Socket Closing..."
        Case sckError
            DecodeWinsockState = "Socket Error"
        Case Else
            DecodeWinsockState = "Socket Unknown State"
    End Select

End Function
```

Queue - 1

```
'-------------------------------------------------------------------------
' This program was developed on a Win2K system using Visual Basic 6.0 for
' windows. The program source and binaries are distributed under a BSD
' Style License.
'
' Developed By:
' Matthew T. Linehan
' American LED-gible Inc.
' Phone: 614.851.1100
' email: engineering@ledgible.com
'
' Copyright (C) 2011, American LED-Gible Inc. All rights reserved.
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'
' Redistributions of source code must retain the above copyright notice, this
' list of conditions, and the following disclaimer. Redistributions in binary
' form must reproduce the above copyright notice, this list of conditions and
' the following disclaimer in the documentation and/or other materials provided
' with the distribution.
'
' THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
' ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
' LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
' CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
' SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
' INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
' CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
' POSSIBILITY OF SUCH DAMAGE.
'-------------------------------------------------------------------------
Private Data As Collection

'Class Initialization
Private Sub class_initialize()
    Set Data = New Collection
End Sub

'Class Termination
Private Sub Class_Terminate()
    Set Data = Nothing
End Sub

'Clear Queue Method
'Resets the queue to empty
Public Sub Clear()
    Set Data = New Collection
End Sub

'Get Queue Count Method
'Return number of objects in the queue
Public Function Count() As Long
    Count = Data.Count
End Function

'Enqueue Item Method
'Add an item to the queue FIFO
Public Sub Enqueue(Value As Variant)
    Data.Add Value
End Sub

'Dequeue Item Method
'Remove an item from the queue FIFO
Public Function Dequeue() As Variant
    Dequeue = Data.Item(1)
    Data.Remove 1
End Function

'Peek at Top Item Method
'Sneak a look at what is at the top of the queue
```

```
Public Function Peek() As Variant
    Peek = Data.Item(1)
End Function
```

Application Title

Version

App Description

[redacted]

[redacted]

Old Password [redacted]

New Password [redacted]

New Password [redacted]

[redacted] [redacted]

Schedule:

Schedule Name:

	Reset	Export	S	M	T	W	T	F	S	Start Time	End Time
1st Shift:											
2nd Shift:											
3rd Shift:											
4th Shift:											
5th Shift:											
6th Shift:											
7th Shift:											
8th Shift:											

Counter: [redacted]

Name: [redacted]

Port: [redacted]

Host Name: [redacted]

Schedule: [redacted]

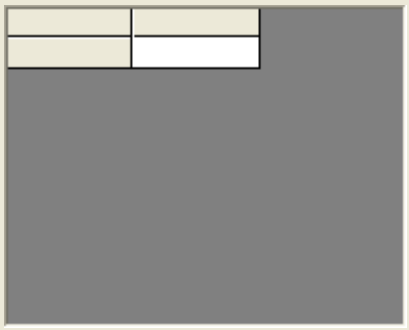
[redacted] [redacted] [redacted]


Totalizer Display:
Name:
Port:
Host Name:
Actual Address:
Totalizer for:

Change All Register Registers to

New Reg: [redacted]

[redacted] [redacted]





11:16 AM

IMessage

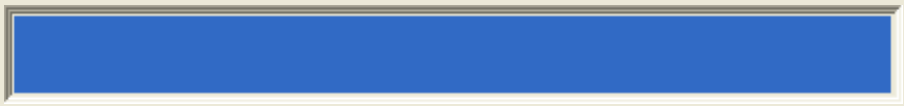
[redacted]

Password: [redacted]

[redacted] [redacted]

App Name

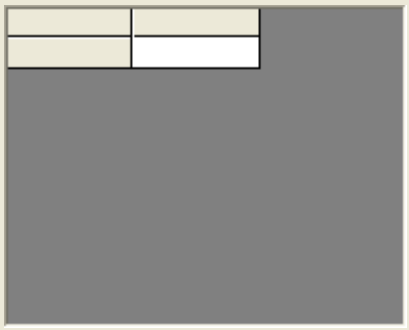
Starting Up...



lTitle

Data Field	Value	Data Field Status	
Actual:	[redacted]	[redacted]	[redacted]

Schedule Processing:
01/01/1900 12:00:00 AM [redacted]



Change Register

Current Reg: 000000

New Reg: [redacted]

[redacted] [redacted]