```
Attribute VB_Name = "mdlCommon"
Option Explicit

' 定数
'   メイン
Public Const MAIN_RANGE As String = "D3,D4,D5,D6,D7"
'   エラー情報
Public Const ERROR_RANGE As String = "D10,D11"
'   入力情報
Public Const INPUT_RANGE As String = "D14,D15,D16,D17,D18,D19,D20,D21,D22,D23,D24,D25,D26"
'   出力情報
Public Const OUTPUT_RANGE As String = "D29,D30,D31,D32,D33,D34,D35,D36,D37,D38,D39,D40,D41"
'   マスター
Public Const MASTER_RANGE As String = ""
'   非表示するシート情報
Public Const HIDDEN_RANGE As String = ""
'   シート名
Public Const SETINFO_SHEETNAME As String = "設定情報"
'   パラメーター
Public Const MAIN_PARA As String = "MAIN"
Public Const ERROR_PARA As String = "ERROR"
Public Const INPUT_PARA As String = "INPUT"
Public Const OUTPUT_PARA As String = "OUTPUT"
Public Const MASTER_PARA As String = "MASTER"
Public Const HIDDEN_PARA As String = "HIDDEN"
'   ファイルタイプ
Public Const PDF_FILETYPE As String = "PDFファイル,*.pdf"
Public Const EXCEL_FILETYPE As String = "Excel 97-2003 ブック (*.xls),*.xls,Excel ブック (*.xlsx),*.xlsx"
Public Const CSV_FILETYPE As String = "CSVファイル,*.csv"
Public Const TEXT_FILETYPE As String = "テキストファイル,*.txt"
' Functionプロシージャ
'*******************************************************************************
'* 処理名   |IsEmptyText
'* 機能     |空文字チェック
'*-------------------------------------------------------------------------------
'* 戻り値   |Boolean (True=値あり, False=値なし)
'* 引数     |strValue：対象文字列
'*******************************************************************************
Function IsEmptyText(strValue As String) As Boolean
    IsEmptyText = (strValue = "")

End Function

'*******************************************************************************
'* 処理名   |IsNumericEx
'* 機能     |数値チェック (半角数値の判定)
'*-------------------------------------------------------------------------------
'* 戻り値   |Boolean (True=半角数値、False=半角数値以外)
'* 引数     |strValue：対象文字列
'*******************************************************************************
Function IsNumericEx(strValue As String) As Boolean
    Dim objReg As New RegExp
```

```
    ' 検索条件：半角数値のみ
    objReg.Pattern = "^[+,-]?([1-9]¥d*|0)(¥.¥d+)?$"
    objReg.Global = True

    IsNumericEx = objReg.Test(strValue)

End Function

'*******************************************************************************
'* 処理名   |IsExistsFile
'* 機能     |ファイルの存在チェック
'*-------------------------------------------------------------------------------
'* 戻り値   |Boolean (True=存在する, False=存在しない)
'* 引数     |strPath：対象ファイルの絶対パス
'*******************************************************************************
Function IsExistsFile(strPath As String) As Boolean
    IsExistsFile = Dir(strPath) <> ""

End Function

'*******************************************************************************
'* 処理名   |IsExistsFolder
'* 機能     |フォルダの存在チェック
'*-------------------------------------------------------------------------------
'* 戻り値   |Boolean (True=存在する, False=存在しない)
'* 引数     |strPath：対象フォルダの絶対パス
'*******************************************************************************
Function IsExistsFolder(strPath As String) As Boolean
    IsExistsFolder = Dir(strPath, vbDirectory) <> ""

End Function

'*******************************************************************************
'* 処理名   |IsReadonlyFile
'* 機能     |ファイルの読み取り専用をチェック
'*-------------------------------------------------------------------------------
'* 戻り値   |Boolean (True=読み取り専用である, False=読み取り専用ではない)
'* 引数     |strPath：対象ファイルの絶対パス
'*******************************************************************************
Function IsReadonlyFile(strPath As String) As Boolean
    Dim vbResult As VbFileAttribute

    ' ファイルの属性を取得
    vbResult = GetAttr(strPath)

    ' 読み取り専用の判定
    IsReadonlyFile = ((vbResult And vbReadOnly) = vbReadOnly)

End Function

'*******************************************************************************
'* 処理名   |IsOpenedFile
'* 機能     |ファイルの開き状態をチェック
'*-------------------------------------------------------------------------------
```

```
'* 戻り値   |Boolean（True=ひらき状態，False=とじた状態）
'* 引数     |strPath：対象ファイルの絶対パス
'*********************************************************************************
Function IsOpenedFile(strPath As String) As Boolean
    Dim intFileno As Integer

    intFileno = FreeFile

    ' 既存ファイルを追記モードで開いた時の戻り値で判定
    On Error Resume Next
    Open strPath For Append As #intFileno
    Close #intFileno

    IsOpenedFile = (Err.Number >= 1)

End Function

'*********************************************************************************
'* 処理名   |IsExistsSheet
'* 機能     |シートの存在チェック
'*---------------------------------------------------------------------------------
'* 戻り値   |Boolean（True=存在する，False=存在しない）
'* 引数     |strSheetname：対象シート名
'*********************************************************************************
Function IsExistsSheet(strSheetname As String) As Boolean
    Dim wbTarget As Workbook
    Dim wsTarget As Worksheet
    Set wbTarget = ThisWorkbook
    Set wsTarget = Nothing

    On Error Resume Next
    Set wsTarget = wbTarget.Worksheets(strSheetname)
    On Error GoTo 0

    IsExistsSheet = (Not (wsTarget Is Nothing))

    Set wsTarget = Nothing
    Set wbTarget = Nothing

End Function

'*********************************************************************************
'* 処理名   |IsEmptyTablerequired
'* 機能     |必須項目チェック（対象はExcelの表）
'*---------------------------------------------------------------------------------
'* 戻り値   |Boolean（True=いずれか空文字あり，False=空文字なし or 0件）
'* 引数     |strSheetname：対象シート名，strRange：セル位置，aryCheckcol：必須項目
'*---------------------------------------------------------------------------------
'* 注意事項：1.対象シートは存在する事が前提となる。
'*             このプロシージャを呼び出す前にシートの存在チェック（IsExistsSheet）を、
'*             実行しエラー制御すること。
'*           2.列の開始位置の最大行数を対象にチェックが実行される。
'*********************************************************************************
Function IsEmptyTablerequired(strSheetname As String, strRange As String, aryCheckcol() As String) As Boolean
```

```
    Dim wbTarget As Workbook
    Dim wsTarget As Worksheet

    Dim rngTarget As Range
    Dim lngBeginrow As Long
    Dim lngBegincol As Long
    Dim lngEndrow As Long
    Dim lngEndcol As Long
    Dim strAddress As String
    Dim lngRow As Long
    Dim lngCount As Long
    Dim strValue As String

    Set rngTarget = Range(strRange)
    lngBeginrow = rngTarget.Row
    lngBegincol = rngTarget.Column

    Set wbTarget = ThisWorkbook
    Set wsTarget = wbTarget.Worksheets(strSheetname)

    With wsTarget
        lngEndrow = .Cells(Rows.Count, lngBegincol).End(xlUp).Row
        If lngBeginrow > lngEndrow Then
            ' データ0件の場合は終了
            IsEmptyTablerequired = False
            Exit Function
        End If
        lngEndcol = .Cells(lngBeginrow, Columns.Count).End(xlToLeft).Column

        For lngRow = lngBeginrow To lngEndrow
            For lngCount = 0 To UBound(aryCheckcol)
                strValue = .Cells(lngRow, CLng(aryCheckcol(lngCount)))
                strAddress = .Cells(lngRow, CLng(aryCheckcol(lngCount))).Address(False, False)
                Call SubSelectCell(strSheetname, strAddress)
                If IsEmptyText(strValue) Then
                    IsEmptyTablerequired = True
                    Exit Function
                End If
            Next
        Next
    End With

    IsEmptyTablerequired = False

    Set wbTarget = Nothing
    Set wsTarget = Nothing

End Function

'*************************************************************************************
'* 処理名   |FuncReadSetinfo
'* 機能     |設定情報の読み込み
'*-------------------------------------------------------------------------------------
'* 戻り値   |String()：指定した処理区分の情報
```

```
'* 引数　　|strClass：処理区分の名前
'*******************************************************************************
Function FuncReadSetinfo(strClass As String) As String()
    Dim arySetinfo() As String

    Select Case strClass
        Case mdlCommon.MAIN_PARA
            arySetinfo = Split(MAIN_RANGE, ",")
        Case mdlCommon.ERROR_PARA
            arySetinfo = Split(ERROR_RANGE, ",")
        Case mdlCommon.INPUT_PARA
            arySetinfo = Split(INPUT_RANGE, ",")
        Case mdlCommon.OUTPUT_PARA
            arySetinfo = Split(OUTPUT_RANGE, ",")
        Case mdlCommon.MASTER_PARA
            arySetinfo = Split(MASTER_RANGE, ",")
        Case mdlCommon.HIDDEN_PARA
            arySetinfo = Split(HIDDEN_RANGE, ",")
    End Select

    FuncReadSetinfo = arySetinfo

End Function

'*******************************************************************************
'* 処理名　|FuncExtractFolderpath
'* 機能　　|ファイルの絶対パスからフォルダのパスを抽出
'*------------------------------------------------------------------------------
'* 戻り値　|String：フォルダのパス
'* 引数　　|-
'*******************************************************************************
Public Function FuncExtractFolderpath(strPath As String) As String
    Dim aryPath() As String

    aryPath = Split(strPath, "¥")
    If UBound(aryPath) > 0 Then
        ReDim Preserve aryPath(UBound(aryPath) - 1)
    End If

    FuncExtractFolderpath = Join(aryPath, "¥")

End Function

'*******************************************************************************
'* 処理名　|FuncShowBreakmessage
'* 機能　　|処理中断のメッセージ表示
'*------------------------------------------------------------------------------
'* 戻り値　|Boolean（True=中断する，False=中断しない）
'* 引数　　|-
'*******************************************************************************
Public Function FuncShowBreakmessage() As Boolean
    Dim strMessage As String

    strMessage = "処理を中断しますか？" & vbCrLf & _
```

```
                vbCrLf & _
                ”　「いいえ」を選択し中断をキャンセルした場合でも、” & vbCrLf & _
                ”　中断したタイミングによってデータの不整合が発生します。” & vbCrLf & _
                vbCrLf & _
                ”　必ず最初から再処理してください。”
    FuncShowBreakmessage = (MsgBox(strMessage, vbQuestion + vbYesNo, ”確認”) = vbYes)

End Function

'*************************************************************************
'* 処理名　|FuncRetrieveMessage
'* 機能　　|メッセージの取得
'*-----------------------------------------------------------------------
'* 戻り値　|String：メッセージ内容
'* 引数　　|lngCode：対象ラベルコード
'*************************************************************************
Public Function FuncRetrieveMessage(strCode As String) As String
    Dim aryMessages(10, 1) As String
    ' 正常終了コード
    aryMessages(0, 0) = ”0”
    aryMessages(0, 1) = ”正常終了。”
    aryMessages(1, 0) = ”10”
    aryMessages(1, 1) = ”初期化を実行。”
    aryMessages(2, 0) = ”20”
    aryMessages(2, 1) = ”フォルダの作成処理が完了。”
    aryMessages(3, 0) = ”999”
    aryMessages(3, 1) = ””
    ' エラーメッセージ
    ' 未入力チェック
    aryMessages(4, 0) = ”-111”
    aryMessages(4, 1) = ”必須項目が未入力。”
    ' 数値チェック
    aryMessages(4, 0) = ”-112”
    aryMessages(4, 1) = ”数値項目で数値以外が入力。”
    ' 数値チェック
    aryMessages(4, 0) = ”-113”
    aryMessages(4, 1) = ”二重登録あり（1行内に複数の階層を入力）。”
    ' 矛盾チェック
    aryMessages(4, 0) = ”-114”
    aryMessages(4, 1) = ”矛盾あり（1行目が0階層以外で設定）。”
    ' 前後入力値チェック
    aryMessages(4, 0) = ”-115”
    aryMessages(4, 1) = ”前後の階層関係に誤り。”
    ' 存在チェック
    aryMessages(5, 0) = ”-211”
    aryMessages(5, 1) = ”参照できないファイルがあり。”
    aryMessages(6, 0) = ”-212”
    aryMessages(6, 1) = ”参照できないフォルダがあり。”
    ' 0件チェック
    aryMessages(7, 0) = ”-311”
    aryMessages(7, 1) = ”取り込んだデータが0件。”
    ' しおり初期化のエラー
    aryMessages(8, 0) = ”-411”
    aryMessages(8, 1) = ”しおりデータ初期化で失敗。”
```

```
    ' しおり書き込みのエラー
    aryMessages(8, 0) = "-511"
    aryMessages(8, 1) = "しおりデータ書き込みで失敗。"
    ' その他エラー
    aryMessages(9, 0) = "-901"
    aryMessages(9, 1) = "実行中に中断。"
    aryMessages(10, 0) = "-999"
    aryMessages(10, 1) = "例外が発生。"

    Dim lngCount As Long
    lngCount = 0
    For lngCount = LBound(aryMessages, 1) To UBound(aryMessages, 1)
        If aryMessages(lngCount, 0) = strCode Then
            FuncRetrieveMessage = aryMessages(lngCount, 1)
            Exit Function
        End If
    Next

    FuncRetrieveMessage = ""

End Function

' Subプロシージャ
'*********************************************************************************
'* 処理名   |SubSelectCell
'* 機能     |指定したセルを選択
'*---------------------------------------------------------------------------------
'* 戻り値   |なし
'* 引数     |strSheetname：対象シート名、strRange：対象セル
'*********************************************************************************
Sub SubSelectCell(strSheetname As String, strRange As String)
    With Worksheets(strSheetname)
        .Select
        .Range(strRange).Select
    End With

End Sub

'*********************************************************************************
'* 処理名   |SubClearSheet
'* 機能     |シートのクリア
'*---------------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |strSheetname：対象シート, strRange：開始セル位置
'*********************************************************************************
Sub SubClearSheet(strSheetname As String, strRange As String)
    Dim wbTarget As Workbook
    Dim wsTarget As Worksheet

    Dim rngTarget As Range
    Dim lngBeginrow As Long
    Dim lngBegincol As Long
    Dim lngEndrow As Long
```

```
    Set wbTarget = ThisWorkbook

    ' 対象シートがない場合でもエラーを発生させず処理を続行
    On Error Resume Next
    Set wsTarget = wbTarget.Worksheets(strSheetname)
    On Error GoTo 0

    Set rngTarget = Range(strRange)
    lngBeginrow = rngTarget.Row
    lngBegincol = rngTarget.Column

    ' シートがある場合
    If (IsExistsSheet(strSheetname)) Then
        With wsTarget
            lngEndrow = .Cells(Rows.Count, lngBegincol).End(xlUp).Row
            ' 表に値がない場合、最大行数を開始行に変更
            If lngBeginrow > lngEndrow Then
                lngEndrow = lngBeginrow
            End If
            With .Range(.Cells(lngBeginrow, lngBegincol), .Cells(lngEndrow, Columns.Count))
                .ClearContents
            End With
            With .Range(.Cells(lngBeginrow, lngBegincol), .Cells(Rows.Count, Columns.Count))
                .Interior.ColorIndex = 0
                .Borders.LineStyle = False
            End With
        End With
    End If

    Set wbTarget = Nothing
    Set wsTarget = Nothing

End Sub

'*****************************************************************************
'* 処理名   |SubCopySheet
'* 機能     |Excelシートのコピー
'*---------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |コピー元（strFmName=シート名, lngFmBeginrow=開始行, lngFmBegincol=開始列）
'*          |コピー先（strToName=シート名, lngToBeginrow=開始行, lngToBegincol=開始列）
'*****************************************************************************
Sub SubCopySheet(strFmName As String, lngFmBeginrow As Long, lngFmBegincol As Long, _
                 strToName As String, lngToBeginrow As Long, lngToBegincol As Long)
    Dim wbTarget As Workbook
    Dim wsFmSheet As Worksheet
    Dim wsToSheet As Worksheet

    Dim lngFmEndrow As Long
    Dim lngFmEndcol As Long
    Dim lngToEndrow As Long
    Dim lngToEndcol As Long

    Dim lngFmRow As Long
```

```
    Dim lngFmCol As Long
    Dim lngToRow As Long
    Dim lngToCol As Long

    Dim lngRow As Long

    Dim varFmCopydata As Variant
    Dim varToCopydata As Variant

    Set wbTarget = ThisWorkbook

    On Error Resume Next
    Set wsFmSheet = wbTarget.Worksheets(strFmName)
    Set wsToSheet = wbTarget.Worksheets(strToName)
    On Error GoTo 0

    ' シートがある場合
    If Not (wsFmSheet Is Nothing) And _
       Not (wsToSheet Is Nothing) Then
        With wsFmSheet
            lngFmEndrow = .Cells(Rows.Count, lngFmBegincol + 1).End(xlUp).Row
            If lngFmBeginrow > lngFmEndrow Then
                lngFmEndrow = lngFmBeginrow
            End If

            lngFmRow = lngFmBeginrow
            lngFmCol = lngFmBegincol
            lngToRow = lngToBeginrow
            lngToCol = lngToBegincol

            ReDim varToCopydata(lngFmEndrow - 1, 30)

            varFmCopydata = .Range(.Cells(lngFmRow, lngFmCol), .Cells(lngFmEndrow, lngFmCol + 30)).Value
        End With

        varToCopydata = varFmCopydata

        ' 編集処理（あれば）
        '   日付をYYYY-MM-DDで表示
        For lngRow = 1 To UBound(varToCopydata)
            varToCopydata(lngRow, 30) = "'" & varToCopydata(lngRow, 30)
        Next

        With wsToSheet
            .Range(.Cells(lngToRow, lngToCol), .Cells(lngFmEndrow, lngToCol + 30)).Value = varToCopydata
            ' 罫線の設定
            lngToEndrow = .Cells(Rows.Count, lngToBegincol).End(xlUp).Row
            lngToEndcol = .Cells(1, Columns.Count).End(xlToLeft).Column
            With .Range(.Cells(lngToBeginrow, lngToBegincol), .Cells(lngToEndrow, lngToEndcol))
                .Borders.LineStyle = True
            End With
        End With
    End If
```

```
    Set wsFmSheet = Nothing
    Set wsToSheet = Nothing
    Set wbTarget = Nothing

End Sub

'*****************************************************************************
'* 処理名   |SubLoadCsv
'* 機能     |CSVファイルの取り込み
'*---------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |strPath：対象ファイルの絶対パス, lngCsvrow：CSV開始行,
'*          |strSheetname：対象シート, strExcelrange：Excel開始セル位置,
'*          |[任意]strCharcode：文字コード
'*****************************************************************************
Sub SubLoadCsv(strPath As String, lngCsvrow As Long, _
               strSheetname As String, strExcelrange As String, _
               Optional strCharcode As String = "UTF8")
    Dim qtTarget As QueryTable
    Dim rngExcel As Range
    Dim lngExcelbeginrow As Long
    Dim lngExcelbegincol As Long
    Dim lngEndrow As Long
    Set rngExcel = Range(strExcelrange)
    lngExcelbeginrow = rngExcel.Row
    lngExcelbegincol = rngExcel.Column

    With Worksheets(strSheetname)
        lngEndrow = .Cells(Rows.Count, lngExcelbegincol).End(xlUp).Row
    End With

    If lngEndrow > lngExcelbeginrow Then
        lngExcelbeginrow = lngEndrow + 1
    End If

    Set qtTarget = Worksheets(strSheetname).QueryTables.Add(Connection:="TEXT;" & strPath, _
        Destination:=Worksheets(strSheetname).Cells(lngExcelbeginrow, lngExcelbegincol))

    With qtTarget
        .TextFileCommaDelimiter = True                          ' カンマ区切りの指定
        .TextFileParseType = xlDelimited                        ' 区切り文字の形式
        .TextFileTextQualifier = xlTextQualifierDoubleQuote     ' 引用符ありダブルクォーテーションを指定
        If strCharcode = "UTF8" Then
            .TextFilePlatform = 65001                           ' 文字コードUTF-8を指定
        Else
            .TextFilePlatform = 932                             ' 文字コードShift_JISを指定
        End If
       .TextFileStartRow = lngCsvrow                            ' 開始行の指定
       .RefreshStyle = xlOverwriteCells                         ' セルは追加せず上書きする
       .Refresh                                                 ' QueryTablesオブジェクトを更新し、シート上に出力
       .Delete                                                  ' QueryTables.Addメソッドで取り込んだCSVとの接続を解除
    End With

    Set qtTarget = Nothing
```

```
End Sub

'*****************************************************************************
'* 処理名   | SubSaveCsv
'* 機能     | CSVファイルの保存（文字コード：UTF-8）
'*---------------------------------------------------------------------------
'* 戻り値   | -
'* 引数     | strSheetname：対象シート, strExcelrange：Excel開始セル位置,
'*          | strSavepath：保存先, lngCsvrow：CSV開始行
'*****************************************************************************
Sub SubSaveCsv(strSheetname As String, strExcelrange As String, _
               strSavepath As String, lngCsvbeginrow As Long)
    Dim rngExcel As Range
    Dim lngExcelbeginrow As Long
    Dim lngExcelbegincol As Long
    Dim lngEndrow As Long
    Dim lngEndcol As Long
    Dim lngRow As Long
    Dim lngCol As Long
    Dim strCsvdata As String
    Dim strLine As String
    Dim strValue As String
    Dim strDelimiter As String
    Dim aryItems() As String

    Set rngExcel = Range(strExcelrange)
    lngExcelbeginrow = rngExcel.Row
    lngExcelbegincol = rngExcel.Column

    Dim objUtf8 As Object
    Dim objNonbom As Object
    Set objUtf8 = CreateObject("ADODB.Stream")
    Set objNonbom = CreateObject("ADODB.Stream")

    ' CSV形式の準備
    With Worksheets(strSheetname)
        lngEndrow = .Cells(Rows.Count, lngExcelbegincol).End(xlUp).Row
        lngEndcol = .Cells(lngExcelbeginrow - 1, Columns.Count).End(xlToLeft).Column
    End With

    If lngCsvbeginrow > lngEndrow Then
        lngEndrow = lngCsvbeginrow
    End If

    strDelimiter = ","

    For lngRow = lngExcelbeginrow To lngEndrow
        strLine = ""
        For lngCol = lngExcelbegincol To lngEndcol
            With Worksheets(strSheetname)
                strValue = .Cells(lngRow, lngCol).Value
            End With
            If strLine = "" Then
```

```
                strLine = strValue
            Else
                ReDim aryItems(1)
                aryItems(0) = strLine
                aryItems(1) = strValue
                strLine = Join(aryItems, strDelimiter)
            End If
        Next

        If strCsvdata = "" Then
            strCsvdata = strLine
        Else
            ReDim aryItems(1)
            aryItems(0) = strCsvdata
            aryItems(1) = strLine
            strCsvdata = Join(aryItems, vbCrLf)
        End If
    Next
    ' 最終行に空文字行
    ReDim aryItems(1)
    aryItems(0) = strCsvdata
    aryItems(1) = ""
    strCsvdata = Join(aryItems, vbCrLf)

    ' 保存
    With objUtf8
        .Charset = "UTF-8"
        .Open
        .WriteText strCsvdata
        .Position = 0
        .Type = 1                       ' Binary
        .Position = 3
    End With

    With objNonbom
        .Type = 1                       ' Binary
        .Open
        objUtf8.CopyTo objNonbom
        .SaveToFile strSavepath, 2      ' SaveCreateOverWrite
    End With

    objNonbom.Close
    objUtf8.Close

    Set objNonbom = Nothing
    Set objUtf8 = Nothing

End Sub

'*************************************************************************************
'* 処理名   |SubShowMessagebox
'* 機能     |メッセージの表示
'*-------------------------------------------------------------------------------------
'* 戻り値   |-
```

```
'* 引数　　| lngCode：対象メッセージコード, [任意]strAppend：追加メッセージ
'*********************************************************************************
Sub SubShowMessagebox(lngCode As Long, Optional strAppend As String = "")
    Dim strMessage As String
    Dim lngLevel As Long
    Dim strLevel As String

    strMessage = FuncRetrieveMessage(CStr(lngCode))
    strMessage = strMessage & vbCrLf & strAppend & vbCrLf

    If Left(CStr(lngCode), 1) = "-" Then
        lngLevel = vbCritical
        strLevel = "警告"
    Else
        lngLevel = vbInformation
        strLevel = "情報"
    End If

    strMessage = strMessage & vbCrLf & "Message Code：" & "[" & CStr(lngCode) & "]"

    MsgBox strMessage, vbOKOnly + lngLevel, strLevel

End Sub

'*********************************************************************************
'* 処理名　| SubDisplayMessage
'* 機能　　| 通知用のメッセージをシート表示
'*---------------------------------------------------------------------------------
'* 戻り値　| -
'* 引数　　| lngCode：対象メッセージコード、[任意]strAppend：追加メッセージ
'*********************************************************************************
Public Sub SubDisplayMessage(lngCode As Long, Optional strAppend As String = "")
    Dim arySetinfo() As String
    arySetinfo = FuncReadSetinfo(mdlCommon.MAIN_PARA)

    Dim strSheetname As String
    Dim strRange As String
    Dim strMessage As String
    Dim strDatetime As String
    Dim lngCount As Long

    strDatetime = Format(Now, "yyyy/mm/dd hh:mm:ss")

    With Worksheets(SETINFO_SHEETNAME)
        strSheetname = .Range(arySetinfo(0)).Value
        strRange = .Range(arySetinfo(1)).Value
    End With

    strMessage = FuncRetrieveMessage(CStr(lngCode))
    If Not (mdlCommon.IsEmptyText(strMessage)) Then
        strMessage = strDatetime & " " & _
                     strMessage & vbCrLf & _
                     strAppend
    End If
```

```
    With Worksheets(strSheetname)
        .Range(strRange).Font.ColorIndex = 11
        If Left(CStr(lngCode), 1) = "-" Then
            .Range(strRange).Font.ColorIndex = 3
        End If
        .Range(strRange).Value = strMessage
    End With

End Sub

'*********************************************************************************
'* 処理名   |SubWriteError
'* 機能     |エラー情報への書き込み
'*---------------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |lngCode：対象メッセージコード、[任意]strAppend：追加メッセージ
'*********************************************************************************
Public Sub SubWriteError(lngCode As Long, Optional strAppend As String = "")
    Dim strDatetime As String
    Dim strLevel As String
    Dim strMessage As String

    strDatetime = Format(Now, "yyyy/mm/dd hh:mm:ss")
    If Left(CStr(lngCode), 1) = "-" Then
        strLevel = "警告"
    Else
        strLevel = "情報"
    End If
    strMessage = FuncRetrieveMessage(CStr(lngCode))

    Dim arySetinfo() As String
    Dim strErrorsheet As String
    Dim strRange As String
    Dim rngTarget As Range
    Dim lngBeginrow As Long
    Dim lngBegincol As Long
    Dim lngEndrow As Long
    arySetinfo = mdlCommon.FuncReadSetinfo(mdlCommon.ERROR_PARA)
    With Worksheets(mdlCommon.SETINFO_SHEETNAME)
        strErrorsheet = .Range(arySetinfo(0)).Value
        strRange = .Range(arySetinfo(1)).Value
    End With

    Set rngTarget = Range(strRange)
    lngBeginrow = rngTarget.Row
    lngBegincol = rngTarget.Column

    With Worksheets(strErrorsheet)
        lngEndrow = .Cells(Rows.Count, lngBegincol).End(xlUp).Row

        .Cells(lngEndrow + 1, lngBegincol).Value = strDatetime
        .Cells(lngEndrow + 1, lngBegincol + 1).Value = strLevel
        .Cells(lngEndrow + 1, lngBegincol + 2).Value = strMessage
```

```
        .Cells(lngEndrow + 1, lngBegincol + 3).Value = strAppend

        With .Range(.Cells(lngBeginrow, lngBegincol), .Cells(lngEndrow + 1, lngBegincol + 3))
            .Borders.LineStyle = True
        End With
    End With
End Sub

'*********************************************************************************
'* 処理名   |SubDisplayStatusbar
'* 機能     |ステータスバーの表示
'*---------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |blValid (True=ステータスバー有効, False=無効), [任意]strAppend：追加メッセージ
'*********************************************************************************
Sub SubDisplayStatusbar(blValid As Boolean, Optional strAppend As String)
    If blValid Then
        Application.StatusBar = "実行中 ..." & strAppend
    Else
        Application.StatusBar = False
    End If

End Sub

'*********************************************************************************
'* 処理名   |SubOnSpeedup
'* 機能     |VBA処理スピードアップ設定
'*---------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |blValid (True=高速化有効, False=無効)
'*********************************************************************************
Sub SubOnSpeedup(blValid As Boolean)
    Dim strPath As String
    If blValid Then
        ' 処理高速化
        ' カーソルを待機中に設定
        '★Application.Cursor = xlWait

        ' 画面描写の停止を設定
        '★Application.ScreenUpdating = False
        ' 自動計算の停止を設定
        '★Application.Calculation = xlCalculationManual
        ' ユーザ操作禁止を設定
        '★Application.Interactive = False
    Else
        ' 画面描写を再開
        Application.ScreenUpdating = True
        ' 自動計算を再開
        Application.Calculation = xlCalculationAutomatic
        ' ユーザ操作を再開
        Application.Interactive = True
        ' カレントディレクトリ移動
        strPath = ThisWorkbook.Path
        ChDrive strPath
```

```
        ChDir strPath

        ' カーソルを元に戻す
        Application.Cursor = xlDefault
    End If

End Sub

'*********************************************************************************
'* 処理名   |SubVisibleSheet
'* 機能     |シート表示／非表示の設定
'*--------------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |strClass：対象グループ, blVisible (True=表示, False=非表示)
'*********************************************************************************
Sub SubVisibleSheet(strClass As String, blVisible As Boolean)
    Dim arySetinfo() As String
    arySetinfo = FuncReadSetinfo(strClass)

    Dim lngCount As Long
    Dim strSheetname As String

    For lngCount = 0 To UBound(arySetinfo)
        strSheetname = Worksheets(SETINFO_SHEETNAME).Range(arySetinfo(lngCount)).Value
        Worksheets(strSheetname).Visible = blVisible
    Next

End Sub

'*********************************************************************************
'* 処理名   |SubOpenFolder
'* 機能     |指定フォルダを開く
'*--------------------------------------------------------------------------------
'* 戻り値   |-
'* 引数     |strPath：対象フォルダの絶対パス
'*********************************************************************************
Sub SubOpenFolder(strPath As String)
On Error GoTo CATCH
    Shell "C:¥Windows¥explorer.exe " & strPath, vbNormalFocus
CATCH:

End Sub
```