Smartist scripting tutorial

Samples reference document

This document references refers to files you can find in
[LASEREDITOR_DIR]\Doc\Tutorial
Where [LASEREDITOR_DIR] refers to LaserEditor installation directory.
All projects were built using the “c:\temp” directory as reference. If you’re using a different directory you need to change every reference to it in the text of the scripts.

1. Verify a file exists

Topics

Usage of objects in scripts

Sample goal

Given a file name and its path (c:\temp\datafile.txt) realize a script that print a message whether the file exists or not

```
Const DATAFILE = "c:\temp\datafile.txt"

Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists DATAFILE Then
   MsgBox "File exists"
Else
   MsgBox "File does not exists"
End If
```

Walkthrough

The `CreateObject` function is used to create OS managed external objects (i.e. ActiveX objects). In this case the `FileSystemObject` is created which, as its name says, grants access to the file system. Once created each object’s reference must be assigned to a variable trough the `Set` keyword. Methods and properties of the FileSystemObject object are described in the Microsoft manual included in the zip file shipping with this document.

2. Parse file input

Topics

Usage of the ‘TextStream’ object
Usage of the ‘Split’ function
Usage of Smartist interface in script engine (‘Document’ object

Sample goal

Read the content of a text file. Assume text is a semicolon-separated set of data: parse the data and assign its content to document objects.

```
Const DATAFILE = "c:\temp\datafile.txt"

Sub Project_OnQueryStart
   Set fso = CreateObject("Scripting.FileSystemObject")
   If fso.FileExists(DATAFILE) Then
```

```
      Set theFile = fso.OpenTextFile(DATAFILE)
      'read a line
      strLine = theFile.ReadLine
      'splitta l'input
      theData = Split(strLine,";")
      'verify the number of values is what expected
      If UBound(theData) <> 2 Then
         MsgBox "Numero di dati errato! " & UBound(theData)
         Exit Sub
      End If
      'assign values to objects
      For i = 0 To 2
         Set theString = Document.GetObject(i+1)
         theString.Text = theData(i)
         theString.Update
      Next
      Document.UpdateView
   Else
      MsgBox "File does not exists"
   End If
End Sub
```

Walkthrough

A TextStream object is created by FileSystemObject's method OpenTextFile. This objects offers functionalities to read/write a file.
The content of the file is parsed trough the `Split` function: this divides data input into an array that is used afterwards to assign values to objects.

3. Tracking limits

Topics

Usage of the Spooler object
Processing items

Sample goal

Given sample n°2 add a feature that allows the user to iterate from marking to tracking limits. When the user presses the start button limits tracking begins. If the user presses the button a second time the system starts marking.

```
Const DATAFILE = "c:\temp\datafile.txt"
DoLimits = True

Sub Project_OnQueryStart
   If DoLimits Then
      Set fso = CreateObject("Scripting.FileSystemObject")
      If fso.FileExists(DATAFILE) Then
         Set theFile = fso.OpenTextFile(DATAFILE)
         'read a line
         strLine = theFile.ReadLine
         'split input
         theData = Split(strLine,";")
         'verify the number of values is what expected
         If UBound(theData) <> 2 Then
            MsgBox "Numero di dati errato! " & UBound(theData)
            Exit Sub
         End If
         'assign values to objects
```

```
      For i = 0 To 2
        Set theString = Document.GetObject(i+1)
        theString.Text = theData(i)
        theString.Update
      Next
      Document.UpdateView

      'starts limits tracking
      Spooler.Reset
      Spooler.Open(False)
      Spooler.SendDocLimits(Document)
      Spooler.Close
      Spooler.Execute
      DoLimits = False
    Else
      MsgBox "File does not exists"
    End If
  Else
    Spooler.Break
    Spooler.Reset
    'mark
    Project.ProcessActiveItem
    DoLimits = True
  End If
End Sub
```

Walkthrough

The DoLimits flag is used to decide whether to track limit or engrave: it switch status each time a phase is completed.
The `Spooler` object is used to send data to the laser. Opening it with the False flag causes the spooler to work in limit mode. The operation sequence is always as indicated: first break (only if the laser is engraving), then Reset to clear any data left from a previous job, then Fill the spooler with object to be sent, finally close it and execute.

4. Loading a document

Topics

Distinguish a project and a document file
Smartist Document object

Sample goal

Read a text file of two lines. The first line contains the name of the document to load, the second line contains the data used to customize the layout to be engraved.

```
Const DATAFILE = "c:\temp\datafile2.txt"
Const LDX_PATH = "c:\temp\"

Sub Project_OnQueryStart
  Set fso = CreateObject("Scripting.FileSystemObject")
  If fso.FileExists(DATAFILE) Then
    Set theFile = fso.OpenTextFile(DATAFILE)
    'read first line, load corresponding document
    DocFile = theFile.ReadLine
    If fso.FileExists(LDX_PATH & DocFile) Then
      Document.Load(LDX_PATH & DocFile)
      Document.UpdateView
    Else
      MsgBox "File " & DocFile & " not found in " & LDX_PATH & vbCr &_
```

```
        "Processing interrupted"
      Exit Sub
    End If
    'read second line
    strLine = theFile.ReadLine
    'split input
    theData = Split(strLine,";")
    'assign values to objects
    For i = 0 To UBound(theData)
      Set theString = Document.GetObject(i+1)
      theString.Text = theData(i)
      theString.Update
    Next
    Document.UpdateView
  Else
    MsgBox "File does not exists"
  End If
End Sub
```

Walkthrough

The fist line of the file is the name of the document file to load. A document file is different from a project file as it only contains marking data (i.e. it does not contain scrip information or items). The UpdateView function is used to force a redrawing of document content.

5. Building a simple user interface

Topics

Usage of the Input object
Managing a cycle

Sample goal

Prompt the user with the name of the file to load. If the user insert an invalid datum, cycle trough the entire process until the datum is valid or the operator gives up.

```
Const LDX_PATH = "c:\temp\"

Sub Project_OnQueryStart
  Set fso = CreateObject("Scripting.FileSystemObject")

  cycle = True
  Do
    Input.QueryString "File name", "*.LDX"
    Input.Ask "Insert name of the file to load"

    'get file name from user input
    fileName = Input.GetString(0)

    'load the file
    If fso.FileExists(LDX_PATH & fileName) Then
      If Document.Load (LDX_PATH & fileName) Then
        Document.UpdateView
        'starts marking
        Project.ProcessActiveItem
        cycle = False
      End If
    End If
```

```
    If cycle Then
      If MsgBox("File does not exist or its type is not supported" & vbCrLf &_
        "Do you want to insert a new file?", vbYesNo) = vbNo Then
        cycle = False
      End If
    End If

  Loop While cycle


End Sub
```

Walkthrough

The input function queryNumber and QueryString can be used to add a line to the interface of the `Input` object; the Ask functions suspend script execution until the user closes the `Input` window. GetString is used to retrieve operator's input.

6. Managing signals

Topics

Usage of the IOPort object

Sample goal

Set a ready signal ON when the script is running, set it OFF when it is not or the laser is busy engraving.
Monitor an external signal: when the signal switches ON, start engraving

```
Const READY_OUT_MASK = &H2000
Const START_IN_MASK= &H1000

IoPort.CheckPort 0
IoPort.SetPort 0, READY_OUT_MASK

Sub IoPort_OnInputChange
  'intercepts external strat signal
  If IoPort.GetPort(0) And START_IN_MASK Then
    IoPort.Setport 0, 0
    Project.ProcessActiveItem
  End If
End Sub

Sub Project_OnItemEnd
  'enable READY signal again
  IoPort.SetPort 0, READY_OUT_MASK
End Sub

Sub Project_OnClose
  'reset script
  IoPort.ResetPort 0, READY_OUT_MASK
  IoPort.UncheckPort 0
End Sub
```

Walkthrough

The MASK constants defined at the beginning are used to determine what PIN(s) functions are operating on; e.g mask &H2000 means PIN n° 13 (as 13th bit is ON in its binery representation)

The CheckPort function starts port monitoring thread: each time an input signal changes an OnInputChange event is triggered.
The OnItemEnd event is triggered at the end of each spooler execution, the OnClose event happens when closing the program or switching to EDIT mode.

7. *Using a Timer*

Topics

Usage of the Timer object

Sample goal

Use a timer to seek a file. If the file exists load the document and prompt the user for marking.

```
Const DATAFILE = "c:\temp\datafile.txt"
Const LDX_PATH = "c:\temp\"

Tmr.SetTimer 0, 100
Set fso = CreateObject("Scripting.FileSystemObject")

Sub Tmr_OnTimer(nTimer)

   If nTimer <> 0 Then
      Exit Sub
   End If

   Tmr.KillTimer 0

   If fso.FileExists(DATAFILE) Then
      Set theFile = fso.OpenTextFile(DATAFILE)
      strLine = theFile.ReadLine
      If fso.FileExists(LDX_PATH & strLine) And Document.Load(LDX_PATH & strLine)
Then
         Document.UpdateView
         If MsgBox("Start engraving document?", vbYesNo) = vbYes Then
            Project.ProcessActiveItem
         Else
            Tmr.SetTimer 0, 100
         End If
      Else
         Tmr.SetTimer 0,100
      End If

      'delete file anyway
      theFile.Close
      fso.DeleteFile(DATAFILE)
   Else
      Tmr.SetTimer 0,100
   End If

End Sub

Sub Project_OnItemEnd
   Tmr.KillTimer 0
End Sub
```

Walkthrough

Smartist exports the timer object with the Tmr keyword. The keyword can be used to instantiate more than one timer: each timer is associated with a numeric ID associated to it with the SetTimer function. The second parameters of this function indicates the time interval the OnTimer function is issued.
The OnTimer event has the timer issuing it as parameter.
KillTimer is used to destroy the timer.

8. Serial port

Topics

Usage of the ComPort object

Sample goal

Read data from the Com port and print it in a string to be engraved

```
Const EndLineChar = 13

If ComPort.Open("COM1,19200,8,1,None") = False Then
   MsgBox "Failed to open COM port!"
Else
   'sets char/flag that raise OnRXFlag event
   ComPort.SetFlag EndLineChar
End If

' ComPort receives CR (end line char)
Sub ComPort_OnRXFlag

   buffer = ComPort.Read
   buffer = Replace(buffer,Chr(EndLineChar),vbNullString)
   Set theString = Document.GetObject(1)
   theString.Text = buffer
   theString.Update

End Sub
```

Walkthrough

The `EndLineChar` constant defines the last character code used in the data stream sent trough the serial cable. When this code is set trough the SetFlag function an OnRXFlag event is issued each time this code is found in the data stream. The content is read then last character is changed with a proper string terminator.

9. Using an ActiveX server

Topics

Using the script as an ActiveX client of Excel

Sample goal

Use the data read from a COM port stream to seek a record in an Excel worksheet. Use data in the recordset to update objects in document.

```
Const EndLineChar = 13

Public Const DBASE_PATH        = "C:\Temp\"
Public Const DBASE_ENTRY       = "LASER.xls"
Public Const SHEET_ENTRY       = "Foglio1"

' Apre il file XLS impiegando gli oggetti ActiveX di Exel
Set ExlApp = CreateObject("Excel.Application")
Set ExlBook = ExlApp.Workbooks.Open(DBASE_PATH & DBASE_ENTRY)
Set ExlSheet = ExlBook.Worksheets(SHEET_ENTRY)

If ComPort.Open("COM1,19200,8,1,None") = False Then
   MsgBox "Failed to open the COM port"
Else
   'sets char/flag that raise OnRXFlag event
   ComPort.SetFlag EndLineChar
End If

' ComPort receives CR (end line char)
Sub ComPort_OnRXFlag

   buffer = ComPort.read
   buffer = Replace(buffer,Chr(EndLineChar),vbNullString)

   LoadData(buffer)

End Sub

Sub LoadData(ID)
   'seek record in excel DB
   nRow = SearchRecord(ExlSheet, ID, 0)
   If nRow >= 0 Then
      Set theString = Document.GetObject(1)
      theString.Text = ExlSheet.Range("B" & nRow).Value
      theString.Update
      Document.UpdateView
   Else
      MsgBox "Record not found"
   End If

End Sub

Function SearchRecord(theSheet,MatchData, FirstRow)

   ' use EXCEL internal searching algorithm
   Set rng = theSheet.Columns("A").Find(MatchData)
   If rng Is Nothing Then
      SearchRecord = -1
   Else
      SearchRecord = rng.Row
   End If

End Function
```

Walkthrough

A connection to an ActiveX server is created trough the `CreateObject` function. Argument of the function is the server name. This name is unique and allows the OS to associate the name with application. This is usually provided by the application developer (Microsoft in this case) as well as for its interface documentation. The sample shows some of the functions associated to the object..