Übungen zu imc FAMOS II – Digital Kurs

- Block 3 - Doc. Rev.: 1.2- 27.08.2025

Gezielter Wissenstransfer - höhere Produktivität

Übung A

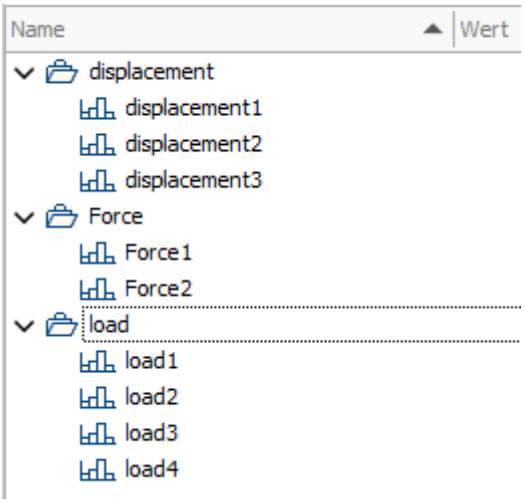
Übungsziel:

In dieser Übung lernen Sie den Umgang mit einer alternativen Methode zu **FileLoad()**, um Daten im FAMOS Format zu laden. Während mit **FileLoad()** eine Datei inklusive ihrem gesamten Inhalt unmittelbar geladen wird, kann mit der Funktion **FileOpenDSF()** die Datei zunächst nach Datenobjekten durchsucht werden. Somit kann der Umgang mit vorhandenen Messdaten direkt beim Laden beeinflusst werden. Auch selektives Laden aus einer Datei wird so möglich.

Aufgabenstellung:

Der Datensatz **LoadTestSample.dat** enthält sieben Datensätze aus 3 verschiedenen Kategorien (siehe auch Übungen aus Block 1). Legen Sie für jede Kategorie eine eigene Gruppe an und laden Sie die Datensätze in die zugehörigen Gruppen.

Ergebnis:



Nach dem Laden erhalten Sie drei Gruppen in der Variablenliste, welche die zugehörigen Datensätze beinhalten, siehe nebenstehenden Screenshot.

Übungsschritte:

- Öffnen Sie die Datei **LoadTestSample.dat** mit der Funktion **FileOpenDSF()** zum Lesen. Um das Schließen der Datei am Ende der Sequenz nicht zu vergessen, erstellen Sie direkt auch den entsprechenden Befehl zum Schließen.

```
file = FileOpenDSF(filename,0)
FileClose(file)
```

  Die Variable **filename** beinhaltet dabei den vollständigen Pfad zu der Datei **LoadTestSample.dat**.

Achtung: Das Argument für den Öffnungstyp sollte mit Bedacht gewählt werden. 0 für **Lesen** ist ungefährlich, aber eine 1 zum **Schreiben** überschreibt eine bestehende Datei ohne Nachfrage!

- Fragen Sie die Anzahl der Datenobjekte in der geöffneten Datei ab:

```
count = FileObjNum?(file)
```

- Erstellen Sie anschließend eine **For**-Schleife, in der die Datenobjekte einzeln abgearbeitet werden:

```
For i = 1 To count
   ; Code der Schleife
End
```

- Innerhalb der Schleife lesen Sie mit der Funktion **FileObjName?()** den Name des i-ten Datenobjekts aus.

```
name = FileObjName?(file,i)
```

- Über eine **If**-Abfrage soll dieser anschließend über einen Textvergleich seiner zugehörigen Gruppe zugeordnet werden. Ist die gesuchte Übereinstimmung vorhanden, fügen Sie dem Namen den gewünschten Gruppennamen gefolgt von einem Doppelpunkt hinzu. Beispielsweise für **Load**:

```
If TLike(name, "Load*",0)
   name = "Load:" + name
End
```

Erstellen Sie analog entsprechende **If**-Abfragen für **Displacement** und **Force**.

- Am Ende der Schleife können in den so neu entstandenen Namen mit Hilfe der Funktion **FileObjRead()** die Messdaten eingelesen werden.

```
<name> = FileObjRead(file,i)
```

Hinweis: Üblicherweise befindet sich in einer *.raw Datei von einem imc Messgerät nur ein Datenobjekt, bei *.dat können dies, wie im vorliegenden Beispiel, mehrere sein. Eine Gruppe in einer Datei wird dabei als ein Objekt gezählt, unabhängig von der Anzahl der Variablen in der Gruppe.

Eine mögliche vollständige Lösung zu dieser Übung finden Sie auf der nächsten Seite. In der Lösung wurden die Pfade und Dateinamen noch in einzelne Variablen geschrieben und der vollständige Name anschließend zusammengebaut. Zudem wurden alle temporären Variablen mit einem vorangestellten Unterstrich versehen und durch den **local** Befehl als temporäre Variablen definiert:


```
; Solution for the Exercise A Block 3
; by imc T&M Training Team
Local _*
_Basepath = "Please insert your Path"
_filename = _basepath + "LoadTestSample.dat"
_file = FileOpenDSF(_filename,0) ; important: only open for reading
not for writing
_count = FileObjNum?(_file)
For _i = 1 To _count
   _name = FileObjName?(_file,_i)
   ; the name may not follow the naming conventions of FAMOS!
   ; here we imply it does
   If TLike(_name, "Load*",0)
      _name = "Load:" + _name
   End
   If TLike(_name, "displ*",0)
      _name = "Displacement:" + _name
   End
   If TLike(_name, "forc*",0)
      _name = "Force:" + _name
   End
   <_name> = FileObjRead(_file,_i)
End
FileClose(_file)
```


Übung B

Übungsziel:

Diese Übung zeigt den Import von Daten außerhalb des FAMOS Formates. Konkret behandelt diese Übung den Import von Textdateien, wobei zwei mögliche Arten von Textdateien vorgestellt und verwendet werden. Einmal ASCII Dateien, die lange Messdatensätze enthalten und daher über die Funktion **FileOpenFAS()** und einem entsprechend erstellten Einlesefilter eingelesen werden, sowie Textdateien, die zusätzliche Informationen wie z.B. Teilenummern, Sondermesswerte eines Spezialmessgerätes oder ähnliches enthalten.

Aufgabenstellung:

Laden Sie die Messdaten aus der Datei **Beispiel_ASCII_Import.txt** mit Hilfe der Funktion **FileOpenFAS()** in entsprechende FAMOS-Variablen. Anschließend laden Sie die Zusatzinformationen wie die Namen der Maschine und des Prüfers sowie die rel. Luftfeuchte aus der Textdatei **Info_Measurement.txt**, indem Sie die Datei zeilenweise oder komplett einlesen und die jeweiligen Informationen mit Hilfe der Textfunktionen aus der Funktionsbibliothek extrahieren.

Ergebnis:

Nach dem Import stehen sowohl die Messdaten aus der Datei **Beispiel_ASCII_Import.txt** als auch die Zusatzinformationen aus der Datei **Info_Measurement.txt** in der Variablenliste zur Verfügung.

Übungsschritte:

Übung 1. Teil: Laden der Datei **Beispiel_ASCII_Import.txt**

- Kopieren Sie die Sequenz aus Übung A ohne den Block für die **If**-Abfragen und ersetzen Sie die Funktion **FileOpenDSF()** durch **FileOpenFAS()** mit dem ASC-II Importfilter:

```
file = FileOpenFAS(filename,
        "#ImportAscii1.dll|ASCII/Excel Import...", 0)
```

  Die vollständige Funktion lautet dann wie folgt:

```
; Solution for Exercise B Block 3
; by imc T&M training team
Local _*
_basepath = "Please insert your Path"
_filename = _basepath + "Beispiel_ASCII_Import.txt"
_file = FileOpenFAS(_filename,
        "#ImportAscii1.dll|ASCII/Excel Import...", 0)
_count = FileObjNum?(_file)
For _i = 1 To _count
   _name = FileObjName?(_file,_i)
   <_name> = FileObjRead(_file,_i)
End
FileClose(_file)
```


- Füllen Sie die **basepath** Variable mit dem Pfad der Datei **Beispiel_ASCII_Import.txt** aus den Beispieldaten und führen Sie die Sequenz aus.
  Da in diesem Fall kein Filter für die ASCII Datei angegeben wurde, öffnet sich der Assistent. Falls in diesem den Datenreihen keine Kanalnamen zugeordnet sind, werden diese automatisch als **Channel1** mit fortlaufender Nummerierung erzeugt.

  Tipp 1: Ein zuvor definierter ASCII Importfilter kann direkt in der FileOpenFAS() Funktion zugeordnet werden. Die Notation hierzu lautet:

  ```
  "#ImportAscii1.dll|ASCII/Excel Import...|Importfilter"
  ```

  Tipp 2: Die Öffnung des Assistenten kann durch Angabe des Parameters **Hide=1** verhindert werden:

  ```
  _file = FileOpenFAS(_filename,"#ImportAscii1.dll|ASCII/Excel
  Import...|Hide=1", 0)
  ```

  In diesem Fall verwendet der Assistent die zuletzt vorgenommenen Einstellungen, die letzte Nutzung durch den Anwender kann also die Automatik beeinflussen.

Übung 2. Teil: Extraktion von Zusatzinfos aus der Textdatei **Info_Measurement.txt**.

Variante 1:

Verwendung von **FileOpenASCII()** zum Öffnen einer Text Datei, anschließend wird mit **FileLineRead()** der Inhalt zeilenweise in Text-Array geladen. Dieses kann dann elementweise abgearbeitet und in die gewünschten Informationen zerlegt werden.

- Erstellen Sie eine neue Sequenz und erzeugen Sie ein leeres Textarray:

  ```
  lines = TxArrayCreate(0)
  ```

- Öffnen Sie die Datei **Info_Measurement.txt** mit der **FileOpenASCII()** Funktion zum Lesen. Auch bei dieser Funktion muss die Datei nach der Verwendung wieder von FAMOS geschlossen werden:

  ```
  file = FileOpenASCII(filename ,0,1)
  ; weiterer Code
  FileClose(file)
  ```

  Die Variable **filename** beinhaltet dabei wieder den vollständigen Pfad der Datei.
- Lesen Sie die Zeilen aus der geöffneten Datei in das zuvor erstellte Textarray ein:

  ```
  status = FileLineRead(file,lines,0)
  ```

  Die gesuchten Informationen stehen nun in den einzelnen Elementen des Text Arrays.
- Um die Werte auszulesen, müssen diese noch von den Beschreibungstexten getrennt werden. Verwenden Sie dazu den Doppelpunkt als Trennzeichen:

  ```
  lineparts = TxSplit(lines[1],":")
  ```

- Aus den Teilen können Sie nun eine entsprechende Variable befüllen. Um überschüssige Leerzeichen zu entfernen verwenden Sie die Funktion **TConv()**:

  ```
  SerialNumber = TConv(lineparts[2],6)
  ```


- Gehen Sie analog vor, um den **Tester** auszulesen:

```
lineparts = TxSplit(lines[2],":")
Tester = TConv(lineparts[2],4)
```

- Lesen Sie den Wert für **Humidity** aus, dieser soll als numerischer Wert in FAMOS erstellt werden. Dazu konvertieren Sie den Text mit Hilfe der Funktion **TtoSv()**:

```
lineparts = TxSplit(lines[3],":")
Humidity = TtoSv(lineparts[2], "a")
```

- Setzen Sie die Einheit der Variablen **Humidity**, indem Sie diese mit **TPart()** aus dem Textelement herausschneiden:

```
SetUnit(Humidity, TPart(lineparts[2], Tleng(lineparts[2]),1),1)
```

Eine mögliche vollständige Lösung zu dieser Übung finden Sie im Folgenden. In der Lösung wurden die Pfade und Dateinamen noch in einzelne Variablen geschrieben und der vollständige Name anschließend zusammengebaut. Zudem wurden alle temporären Variablen mit einem vorangestellten Unterstrich versehen und durch den **local** Befehl als temporäre Variablen definiert:

```
; Solution for Exercise C Block 3
; by imc T&M Training Team
Local _*
_lines = TxArrayCreate(0)
_Basepath =
_filename = _basepath + "Info_Measurment.txt"
_file = FileOpenASCII(_filename ,0,1)
_status = FileLineRead(_file,_lines,0)
FileClose(_file)
_lineparts= TxSplit(_lines[1],":")
SerialNumber = TConv(_lineparts[2],6)
_lineparts = TxSplit(_lines[2],":")
Tester = TConv(_lineparts[2],4)
_lineparts = TxSplit(_lines[3],":")
Humidity = TtoSV(_lineparts[2], "a")
SetUnit(Humidity,TPart(_lineparts[2],Tleng(_lineparts[2]),1),1)
```

Variante 2:

Anstatt die Datei zeilenweise einzulesen, lädt die Funktion **FileOpenFAS()** mit dem Argument **imc|text** die komplette Textdatei in eine Textvariable. Diese wird anschließend mit der Funktion **TxSplit()** zunächst in einzelne Zeilen zerlegt und die Teile können anschließend wie in Variante 1 weiter verwertet werden.

- Erstellen Sie eine neue Sequenz.
- Laden Sie die komplette Datei in eine Textvariable. Öffnen Sie dazu die Datei, lesen den Inhalt und schließen diese wieder:

```
file = FileOpenFas(filename,"imc/Text/auto",0)
content = FileObjRead(file,1)
FileClose(file)
```


- Die einzelnen Zeilen können anhand des Zeilenumbruchs identifiziert werden (ASCII Zeichen 13 und 10). Verwenden Sie diesen in der **TxSplit()** Funktion:

```
lines = TxSplit(content, "~013"+"~010")
```

Nun liegen die einzelnen Zeilen wieder in einem Textarray vor und können analog zu Variante 1 weiterverarbeitet werden.
Hinweis: Variante 1 ist oft bequemer für zeilenorientierte Texte, Variante 2 bietet sich für xml-Dateien oder nicht-zeilenorientierte Textdateien an.

Übung C

Übungsziel:

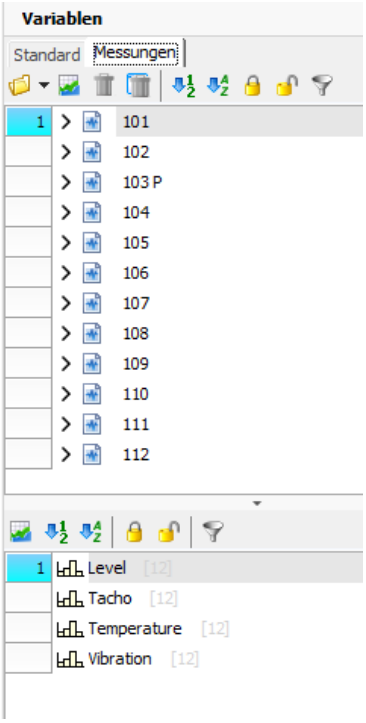
In dieser Übung lernen Sie, wie Sie mit Hilfe von Funktionen aus der Funktionsbibliothek auf das Dateisystem des PCs per Sequenz zugreifen können, um Ordner bzw. Dateien zu suchen und einzulesen. Neben dem Einlesen von Ordnerstrukturen und Dateilisten kommen außerdem Techniken zum Zerlegen von Dateipfaden sowie der direkten Zuweisung einer Messungszugehörigkeit zum Einsatz.

Aufgabenstellung:

Laden sie aus dem Ordner **Test** alle durchgeführten Bremsmessungen eines Zuges, so dass diese als Messungsnamen den jeweiligen Ordnernamen tragen aus dem sie importiert wurden.

Ergebnis:

Nach dem Laden des Ordners sind alle Dateien inklusive Messungszugehörigkeit geladen. Wechseln Sie im Variablenfenster auf den Tab **Messungen**, um die Messungsansicht zu erhalten.



Übungsschritte:

- Erstellen Sie eine neue Sequenz
- Untersuchen Sie den Ordner **Test** aus den Beispieldateien auf seinen Inhalt, einschließlich seiner Unterordner. Die Variable **basepath** beschreibt dabei den vollständigen Pfad des Zielordners:

```
list_of_dirs = FsFileListNew(basepath, "*",1, 0, 1)
; weiterer Code
FsFileListClose(list_of_dirs)
```


Das Ergebnis ist eine Liste aller vollständigen Pfade zu den gefundenen Ordnern. Analog zu geöffneten Dateien müssen auch Listen nach ihrer Verwendung wieder geschlossen werden, fügen Sie daher die Zeile direkt mit ein.

- Fragen Sie die Anzahl der gefundenen Ordner ab:

```
number_of_dirs = FsFileListGetCount(list_of_dirs)
```

- Mit der gefundenen Anzahl an Ordnern erstellen Sie eine Schleife über alle gefundenen Ordner, sodass diese nacheinander abgearbeitet werden können:

```
For i = 1 To number_of_dirs
    ; weiterer Code
End
```

- Innerhalb der Schleife finden Sie zunächst den aktuellen Ordner heraus:

```
act_dir = FsFileListGetName(list_of_dirs,i)
```

- Finden Sie alle Dateien innerhalb des aktuellen Ordners und speichern die vollständigen Pfade in einer Liste. Vergessen Sie auch bei dieser Liste nicht den Befehl zum Schließen der Liste nach ihrer Verwendung:

```
list_files = FsFileListNew(act_dir, "*.raw",0, 0, 0)
; weiterer Code
FsFileListClose(list_files)
```

- Fragen Sie die Anzahl der gefundenen Dateien im angegebenen Ordner ab:

```
number_of_files = FsFileListGetCount(list_files)
```

- Erstellen Sie mit Hilfe der gefundenen Anzahl eine weitere Schleife, welche alle Dateien nacheinander bearbeitet. Achten Sie darauf, dass Sie nicht die gleiche Indexvariable wie bei der bereits laufenden Schleife verwenden:

```
For j = 1 To number_of_files
    ; weiterer Code
End
```

- Schreiben Sie den Pfad der jeweiligen Dateien aus der Liste in eine Variable:

```
filename = FsFileListGetName(list_files , j)
```

- Öffnen Sie die Datei, analog zu Übung A, unter der Annahme, dass diese nur ein einzelnes Datenobjekt enthält und lesen Sie den Namen des Datenobjekts aus. Diese Annahme kann getroffen werden, da es sich hier um **.raw** Dateien handelt, die von einem imc Messsystem erstellt wurden. Solche Dateien beinhalten in der Regel nur ein einziges Datenobjekt. Vergessen Sie nicht die Datei am Ende wieder zu schließen.

```
; raw files normally only contain one dataset
file = FileOpenDSF(filename, 0)
name = FileObjName?(file,1)
; weiterer Code
FileClose(file)
```


- Der Variablenname soll noch um einen Messungsnamen ergänzt werden, welcher aus dem Dateinamen besteht. Schneiden Sie den Dateinamen aus dem vollständigen Pfad aus und laden Sie anschließend das Datenobjekt inklusive Messungszuordnung:

```
measname = FsSplitPath(filename , 6)
{<name>}@{<measname>} = FileObjRead(file, 1)
```

  Hinweis: Die geschweiften Klammern stellen sicher, dass die Namen in FAMOS keine ungültigen Zeichen beinhalten und ggf. vorher bereinigt werden.
- Speichern Sie die Sequenz und führen Sie sie aus.

Eine mögliche vollständige Lösung zu dieser Übung finden Sie im Folgenden. In der Lösung wurden die Pfade und Dateinamen noch in einzelne Variablen geschrieben und der vollständige Name anschließend zusammengebaut. Zudem wurden alle temporären Variablen mit einem vorangestellten Unterstrich versehen und durch den **local** Befehl als temporäre Variablen definiert:

```
; Solution for Exercise C Block 3
; by imc T&M Training Team
Local _*
_basepath = "Please adjust to your own path"
_basepath = "d:\xxx\Sample_Data\Test"
_list_of_dirs = FsFileListNew(_basepath, "*",1, 0, 1)
_number_of_dirs = FsFileListGetCount(_list_of_dirs )
For _i = 1 To _number_of_dirs
   _act_dir = FsFileListGetName(_list_of_dirs,_i)
   _list_files = FsFileListNew(_act_dir, "*.raw",0, 0, 0)
   _number_of_files = FsFileListGetCount(_list_files)
   For _j = 1 To _number_of_files
      _filename = FsFileListGetName(_list_files , _j)
      ; raw files normally only contain one Dataset
      _file = FileOpenDSF(_filename, 0)
      _name = FileObjName?(_file,1)
      _measname = FsSplitPath(_filename , 6)
      {<_name>}@{<_measname>} = FileObjRead(_file, 1)
      FileClose(_file)
   End
   FsFileListClose(_list_files)
End
FsFileListClose(_list_of_dirs)
```

