



**Innovative Teaching Approaches
in development of Software Designed
Instrumentation and its application
in real-time systems**

The Advanced Applications of LabVIEW

Lecture 1: Sequential state machine

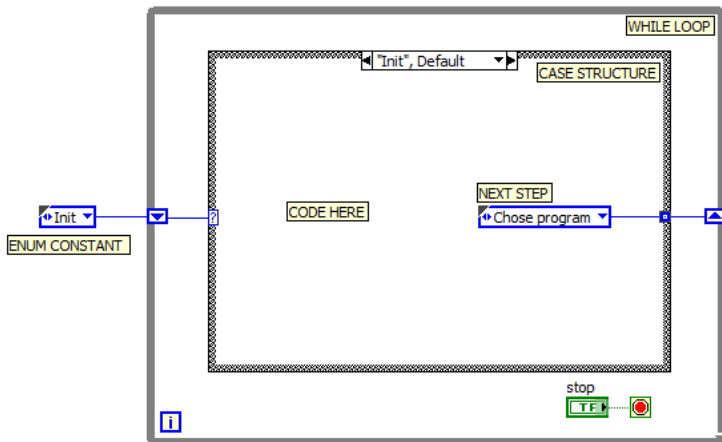
Co-funded by the
Erasmus+ Programme
of the European Union



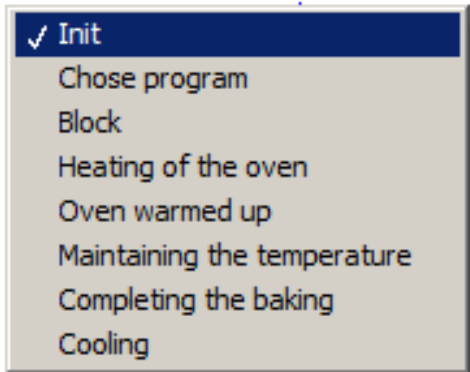
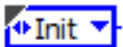
Application of state machine

- State machine can be used for all sequential tasks e.g. implementation of all functionality of machines, management of processes in experiments, etc.
- The state machine can be easily modified by adding another step as next frame in its case structure.
- The code is getting smaller and easy to read and modify.

Structure of state machine

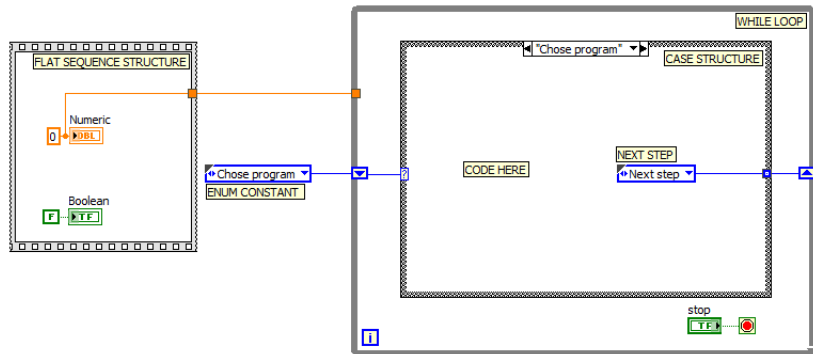


- Enum constant should include all necessary steps. You should draw algorithm of code using blocks. All blocks should be the steps included in enum.
- Remember to make type def from enum to easily modify it in future.



Initialization of variables

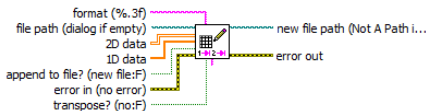
- Variables can be initialized in the beginning step in case structure (see previous slide).
- We can also use the *Flat Sequence Structure* to initialize the values of controls/indicators. Remember to connect the *Flat Sequence Structure* with *While loop* to make sure that *Flat Sequence Structure* will work first.



Initialization from the disk

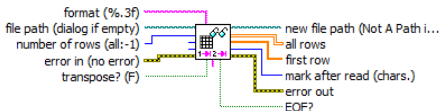
- Reading data from the disk is very useful especially if you want to modify it in the future.
- You can read configuration data from CSV files by using *Write/Read Delimited Spreadsheet* functions:

Write Delimited Spreadsheet.vi



Converts a 2D or 1D array of strings, signed integers, or double-precision numbers to a text string and writes the string to a new byte stream file or appends the string to an existing file. Wire data to the **2D data** input or **1D data** input to determine the polymorphic instance to use or manually select the instance.

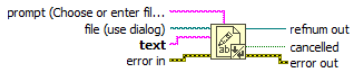
Read Delimited Spreadsheet.vi



Initialization from the disk

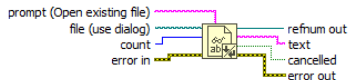
- You can read configuration data from CSV or TXT files by using functions:

Write to Text File



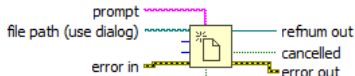
Writes a string or an array of strings as lines to a file. This function does not work for files inside an LLB.

Read from Text File



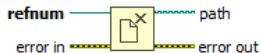
Reads a specified number of characters or lines from a byte stream file. This function does not work for files inside an LLB.

Open/Create/Replace File



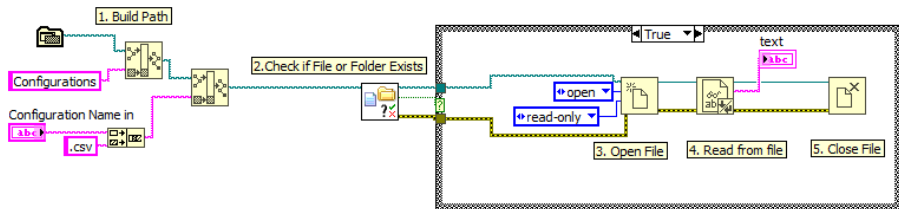
Opens an existing file, creates a new file, or replaces an existing file, programmatically or interactively using a file dialog box. This function does not work for files inside an LLB.

Close File



Closes an open file specified by **refnum** and returns the path to the file associated with the refnum.

An example of reading the data from CSV/TXT file



Initialization from the disk

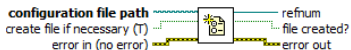
- You can read configuration data from initialization files (.INI).
- Initialization files contain sections and keys. A key is a constant value. A section is a group of related keys.

```
[Cotton]
water temperature = "40"
Spin speed = "400"
Duration of washing = "60"
Duration of centrifugation = "30"
Duration of rinsing = "30"

[Mixed]
water temperature = "40"
Spin speed = "800"
Duration of washing = "90"
Duration of centrifugation = "30"
Duration of rinsing = "30"
```

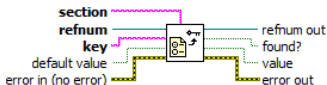
- You can read configuration data from initialization files (.INI) by using the functions:

NI_LVConfig.lvlib:Open Config Data.vi



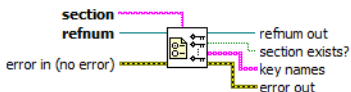
Opens a reference to the configuration data found in a platform-independent configuration file.

NI_LVConfig.lvlib:Read Key.vi



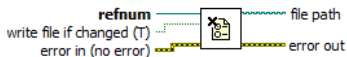
Reads a value associated with a key in a specified section from the configuration data identified by **refnum**. If the key does not exist, the VI returns the default value. This VI supports multibyte characters in strings. Wire data to the **default value** input to determine the polymorphic instance to use or manually select the instance.

NI_LVConfig.lvlib:Get Key Names.vi



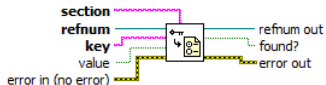
Gets the names of all keys in the specified section from the configuration data identified by **refnum**.

NI_LVConfig.lvlib:Close Config Data.vi



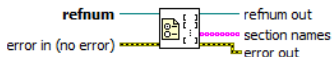
Writes data to the platform-independent configuration file identified by **refnum** and then closes the reference to that file.

NI_LVConfig.lvlib:Write Key.vi

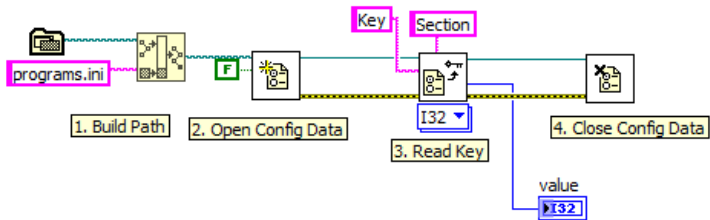


Writes a value to a key in a specified section of the configuration data identified by **refnum**. This VI modifies data in memory. To write data to disk, use the Close Config Data VI. Wire data to the **value** input to determine the polymorphic instance to use or manually select the instance.

NI_LVConfig.lvlib:Get Section Names.vi



Gets the names of all sections from the configuration data identified by **refnum**.





Itasdi

Thank you for your attention!

The lecture was prepared based on materials from "LabVIEW Core 3 Course Manual".

This project has been funded with support from the European Commission. This communication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.