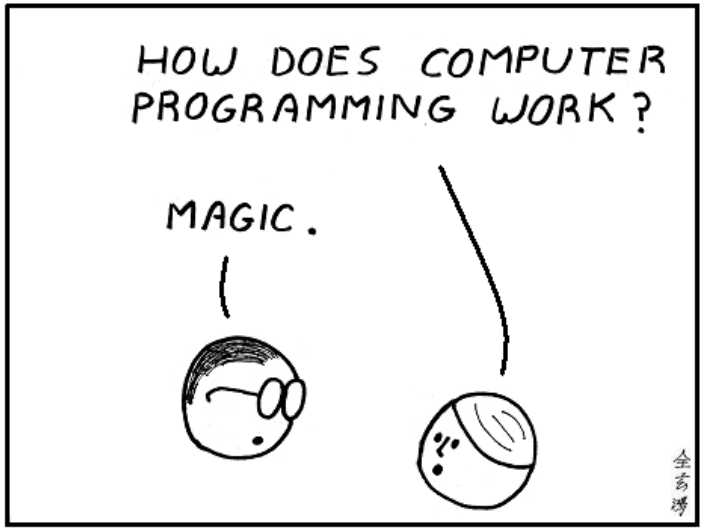
ESM Practical

Controlling laboratory experiments in LabVIEW



Dr. Thomas Sebastian & Dr. Thomas Meyer

THATec Innovation GmbH

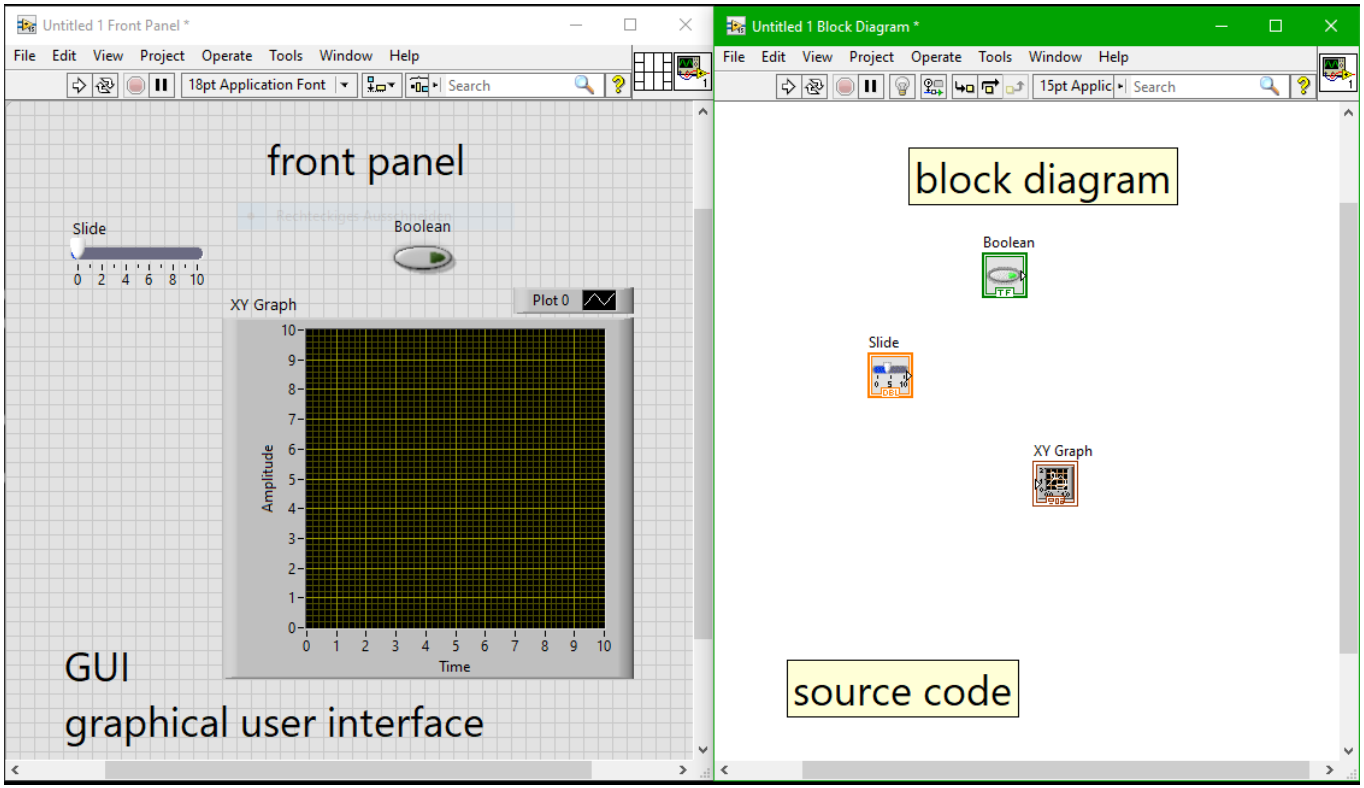
LabVIEW?

- Programming language and IDE (integrated development environment) from *National Instruments*
- Graphical programming
- Focus on
  - Graphical user interface (GUI)
  - Device I/O
  - Rapid development in the lab
  - Tools for data evaluation/visualization

Overview

- Getting started 1
  - Basic concepts and core components
  - Control structures 1 (loops, if/case structure, …)
  - ➢ Develop your first program
- Getting started 2
  - Control structures 2
  - subVIs: reusable code
  - ➢ Extend your code
- Quick tour through the LabVIEW toolbox
  - Multithreading
  - Data storage and visualization
  - Device I/O
- ➢ Hands on! Control devices

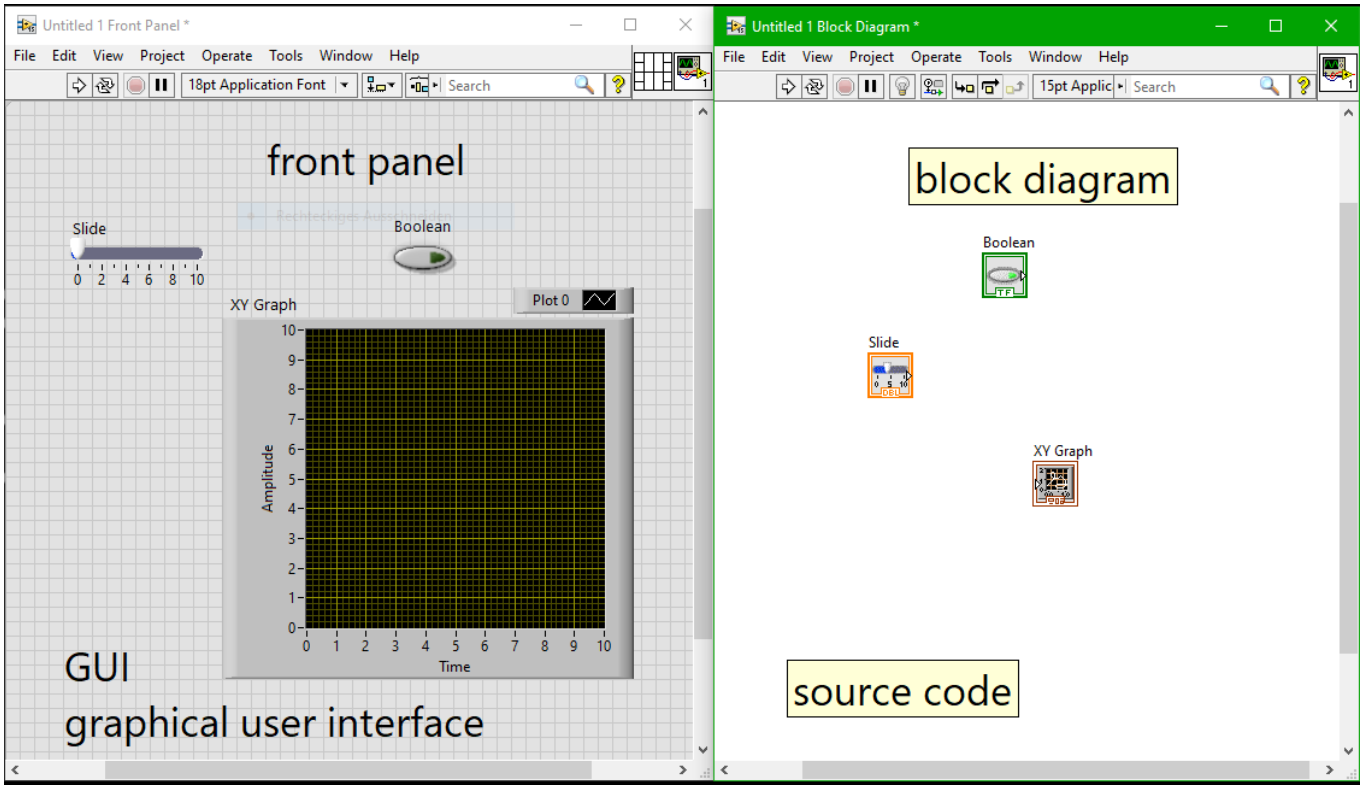
Core components



IDE, GUI, and graphical programming

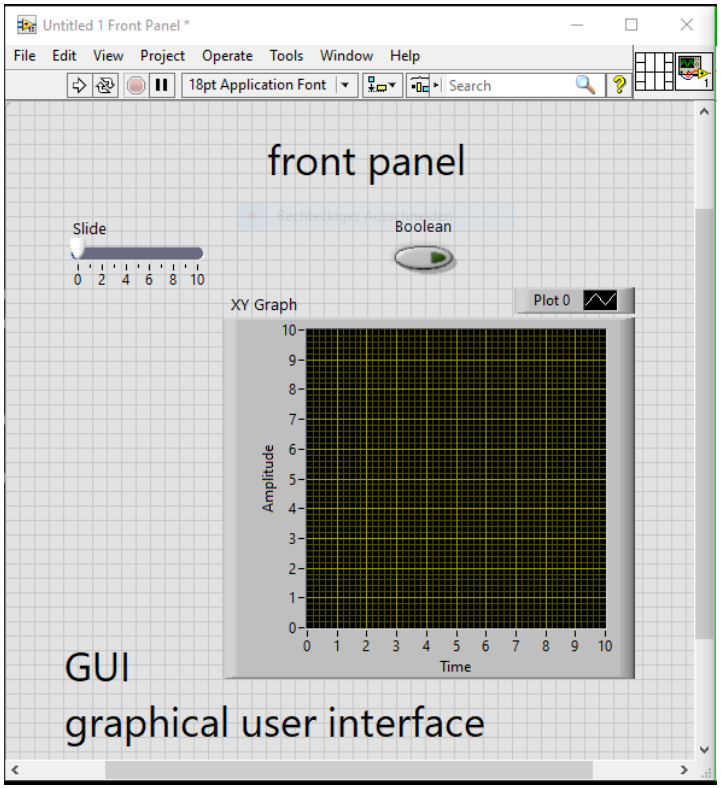
- No need for external compiler
- GUI and source code at the same time
- Switch between windows with Crtl+e

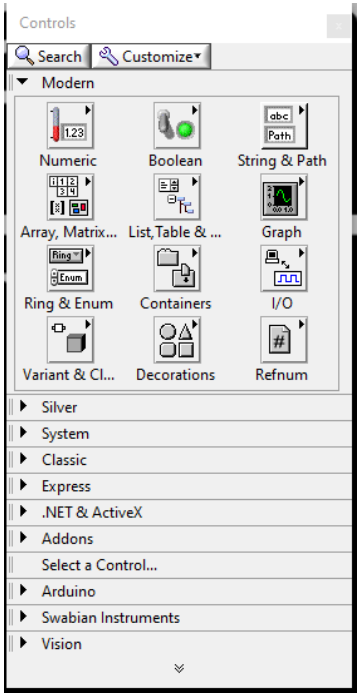
Core components



- Programs/functions/routines: VI: virtual instruments, *.vi files

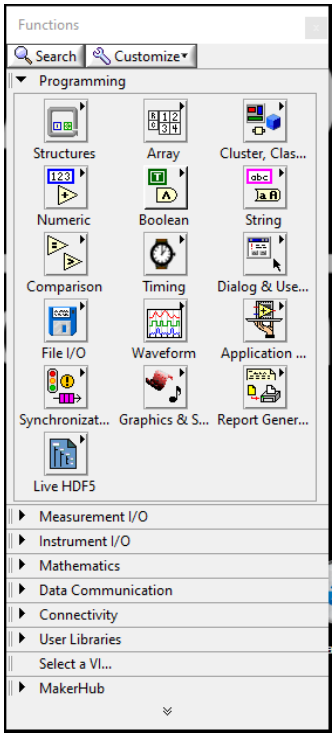
Core components

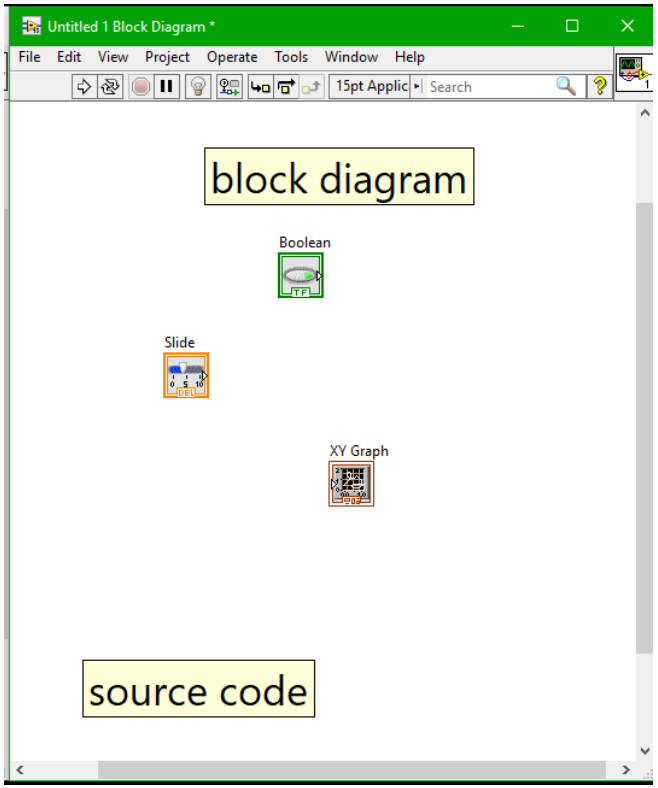




- Programs/functions/routines: VI: virtual instruments, *.vi files
- Tools typically available via right-click menu
  - Front panel: add new objects
  - Block diagram: add new functions

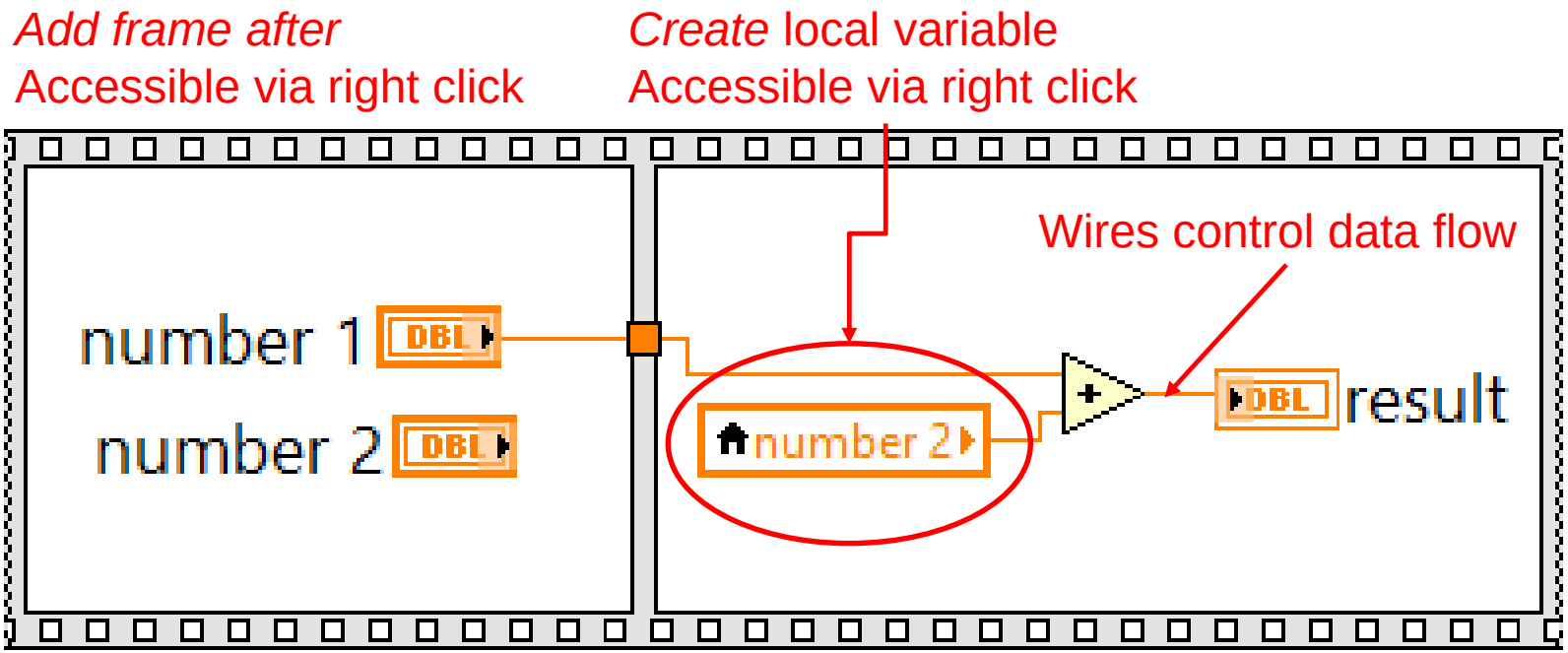
Core components





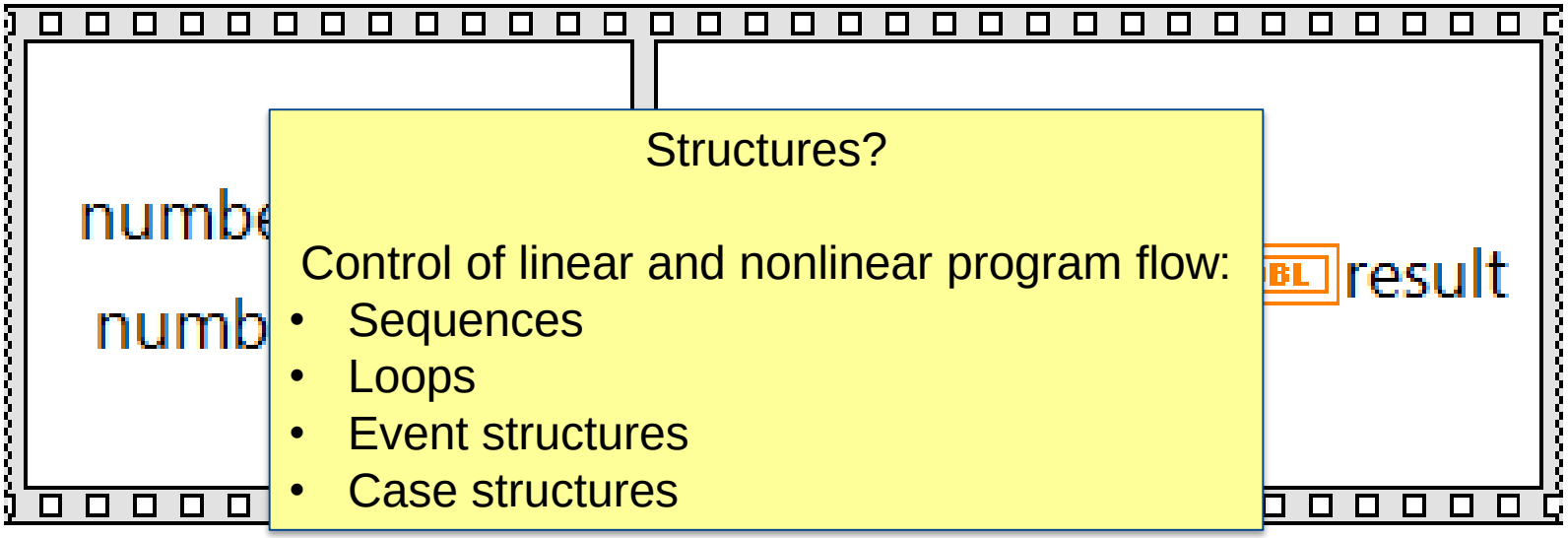
- Tools typically available via right-click menu
  - Block diagram: add new functions, structures, …
  - All actions available via right-click on objects:
    Change properties, create local variables, add frame…

Core components



- Programs/functions/routines: VI: virtual instruments, *.vi files
- Tools typically available via right-click menu
  - Front panel: add new objects
  - Block diagram: add new functions, structures, …

Core components

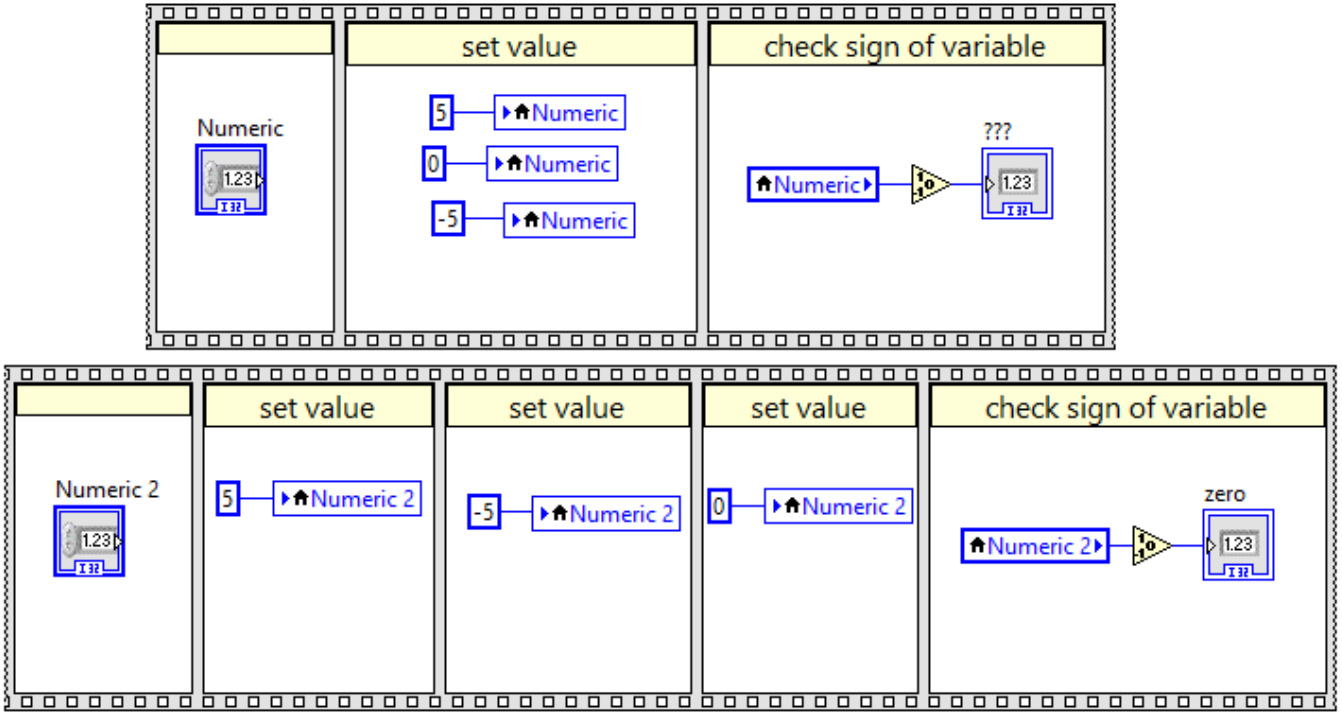


- Programs/functions/routines: VI: virtual instruments, *.vi files
- Tools typically available via right-click menu
  - Front panel: add new objects
  - Block diagram: add new functions, structures, …

Core components

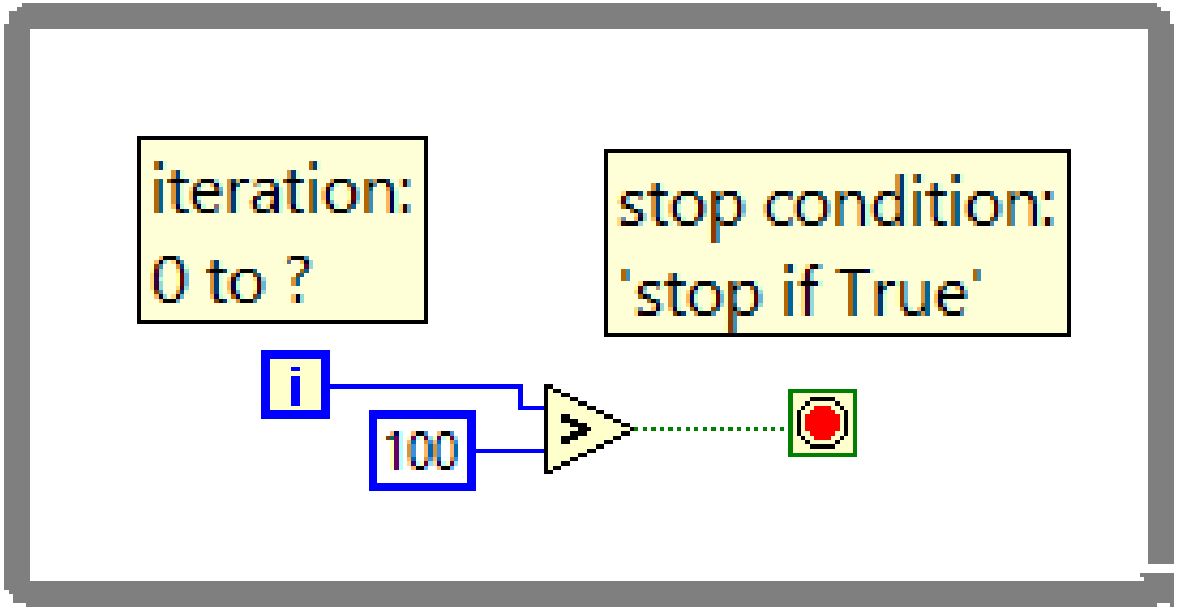
Sequence

- Control of linear program flow
- Next frame is executed only after completion of previous one



Core components

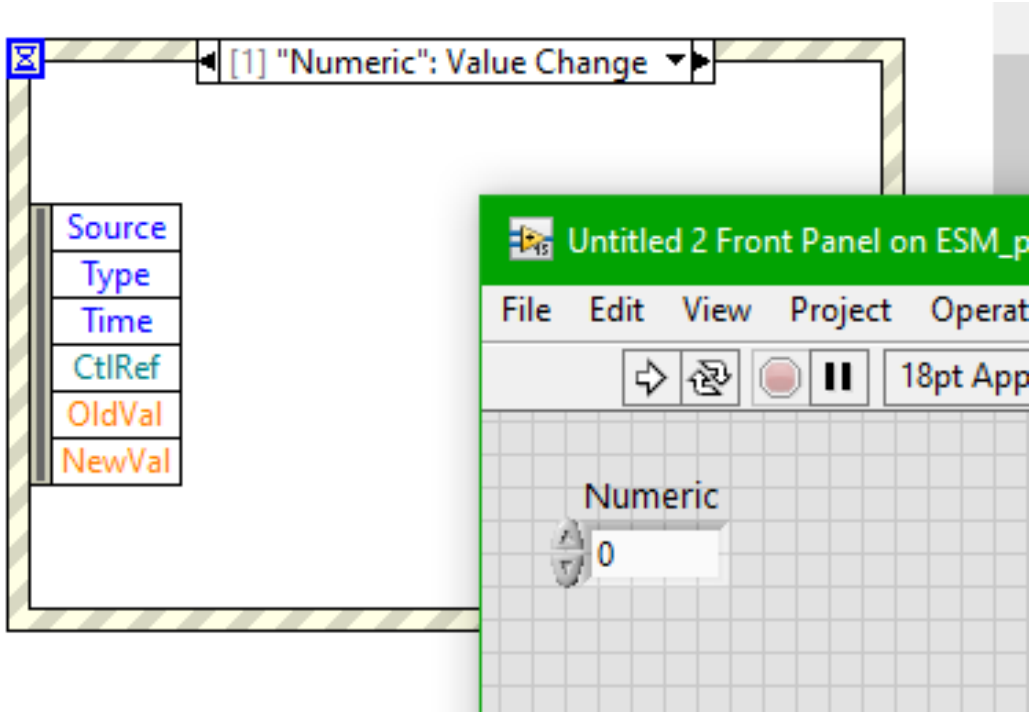
While loop: Operates until stop condition is met



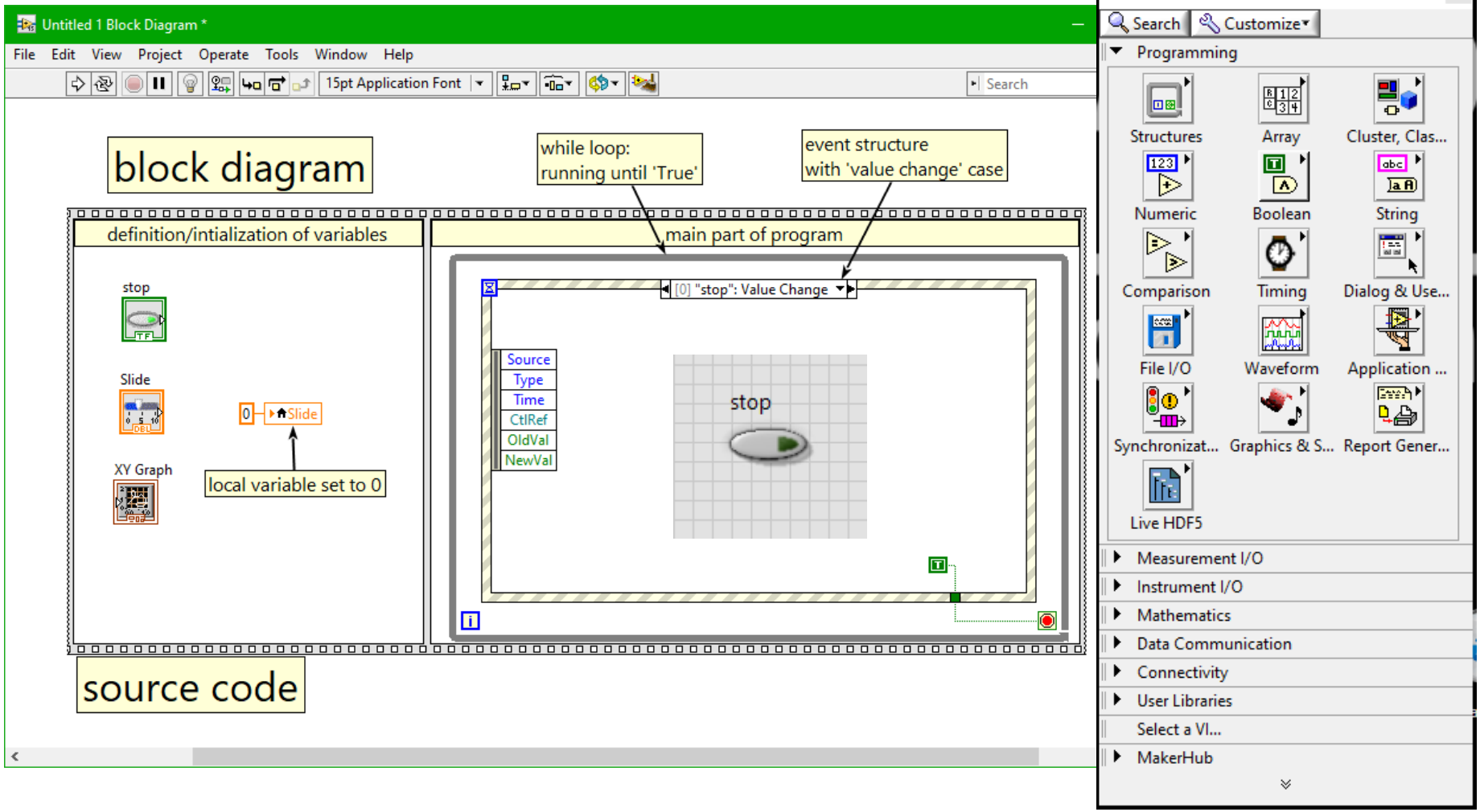
Core components

Event stucture

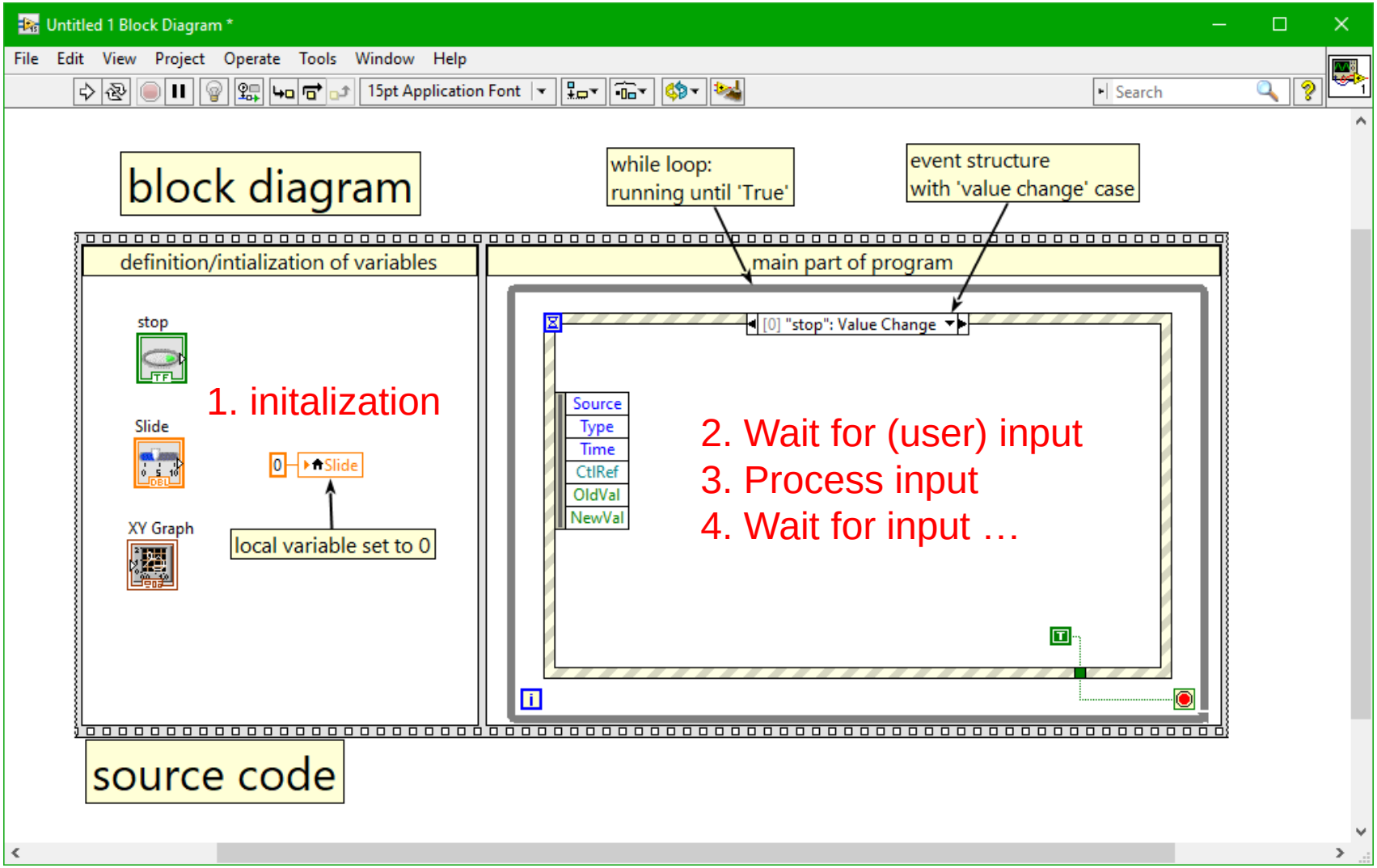
- *Listens* to user interactions
- Button clicked, value changed, …



Core components

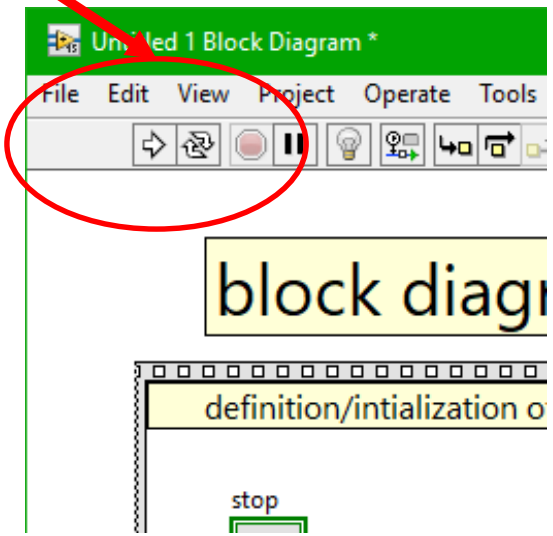


Core components



Develope your first VI

Start your program!



1) Start LabVIEW and create a project *ESM2019*
2) Add a new VI *main.VI* to this project
3) Add objects to the front panel
   - three numerical variables
   - one Boolean
   - one string
4) What is the difference between *controls* and *indicators*?
5) Block diagram:
   - Arrange variables in a sequence
   - Initialize variables with start values (“Hello, World!”)
   - Add numerical values
   - Create a while loop and an event structure
     - Add numerical values in case of a *value change*
     - Stop program when Boolean is switched

- Help window (Ctrl+h)
- Switch windows (Ctrl+e)
- Navigation window (Ctrl+n)
- Right-click helps!

Overview

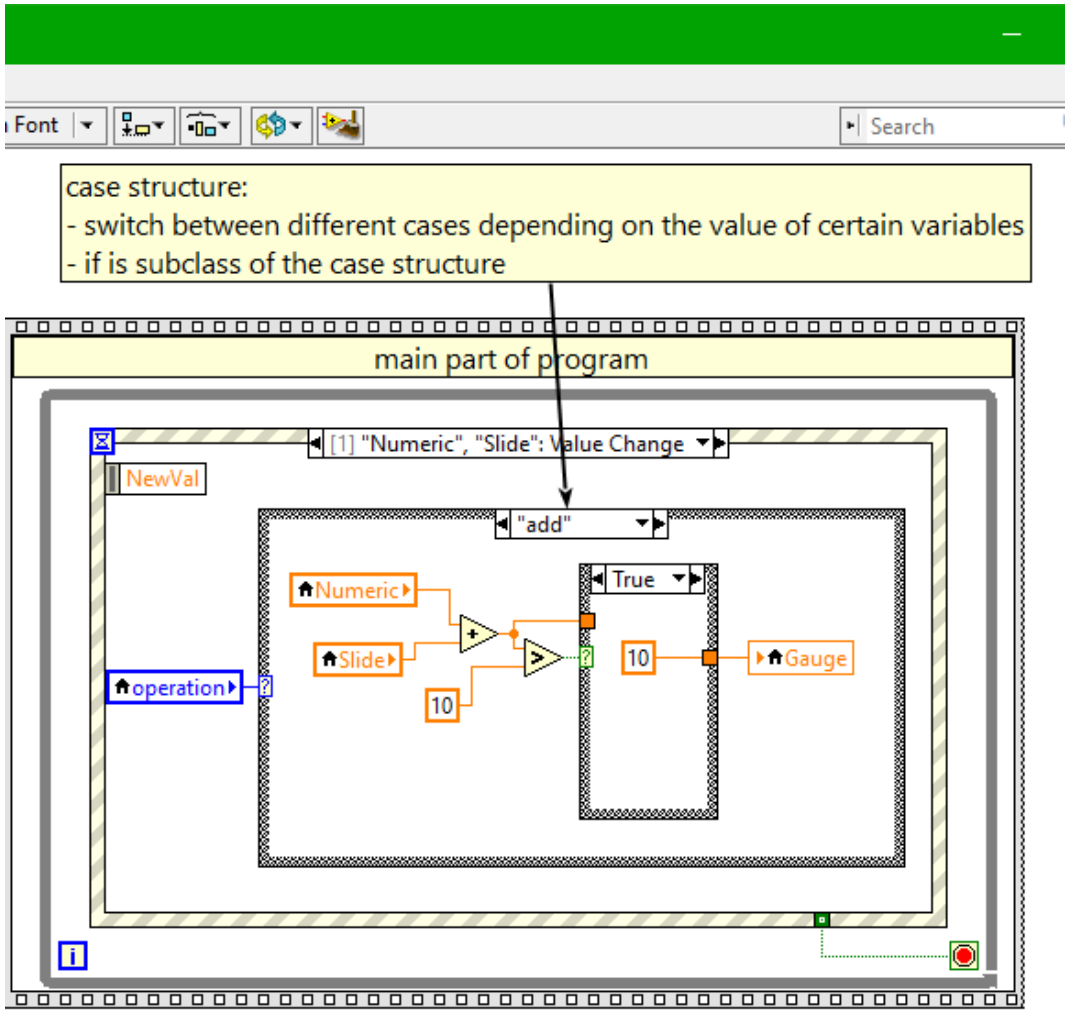
- Getting started 1
  - Basic concepts and core components
  - Control structures 1 (loops, if/case structure, …)
  - ➢ Develop your first program

- Getting started 2
  - Control structures 2
  - subVIs: reusable code
  - ➢ Extend your code

- Quick tour through the LabVIEW toolbox
  - Multithreading
  - Data storage and visualization
  - Device I/O

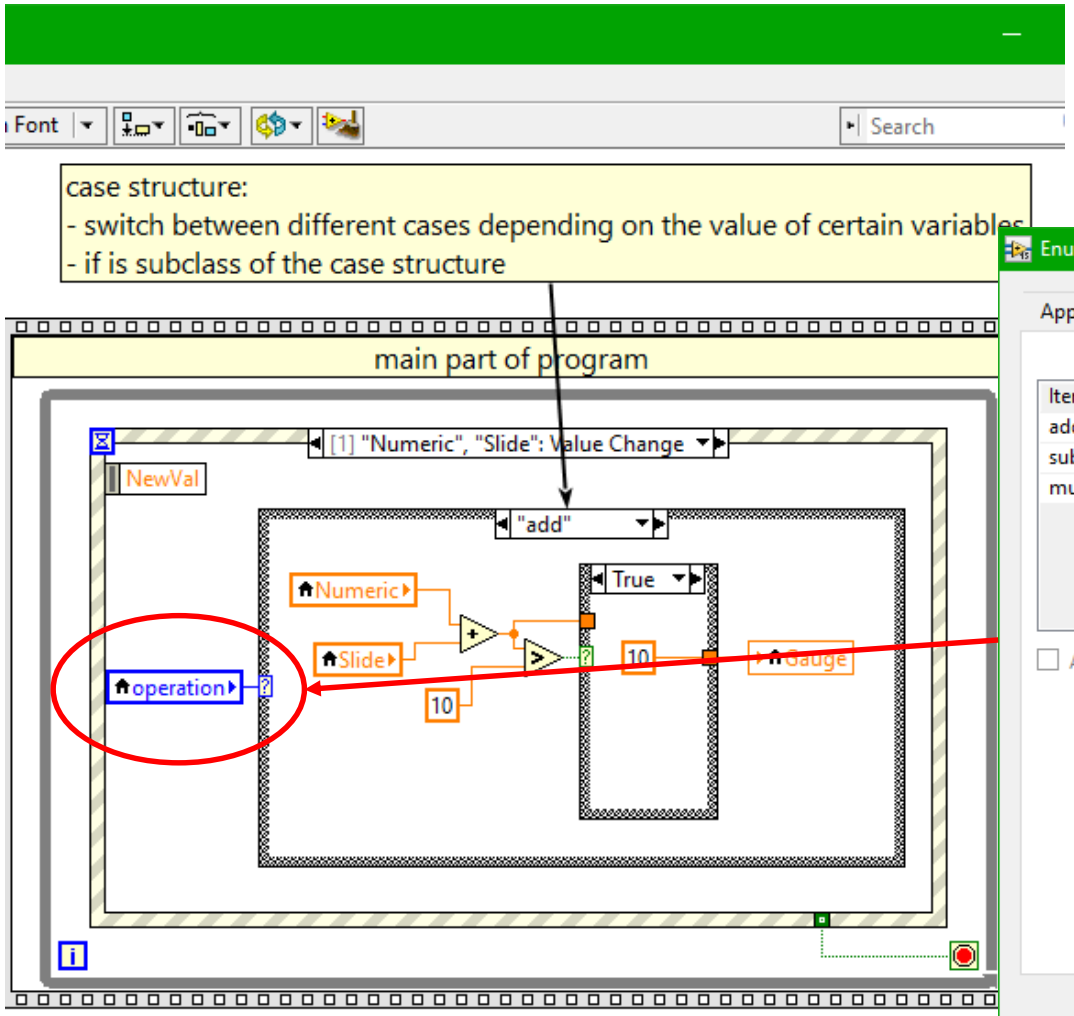
- ➢ Hands on! Control devices

Case structure / if

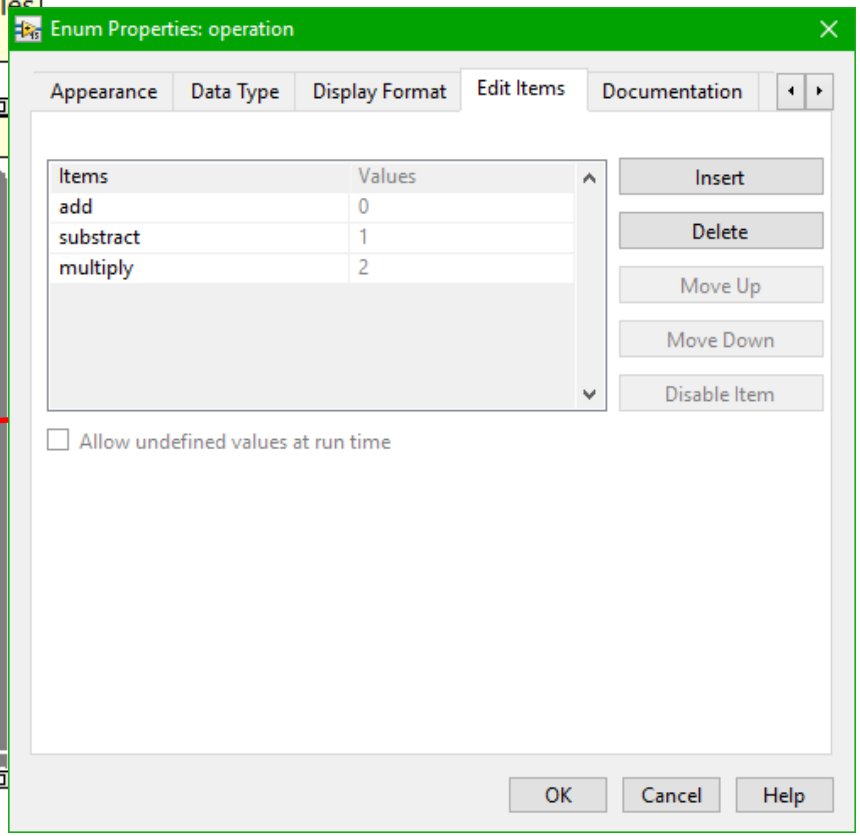


1) Choose functions depending on values

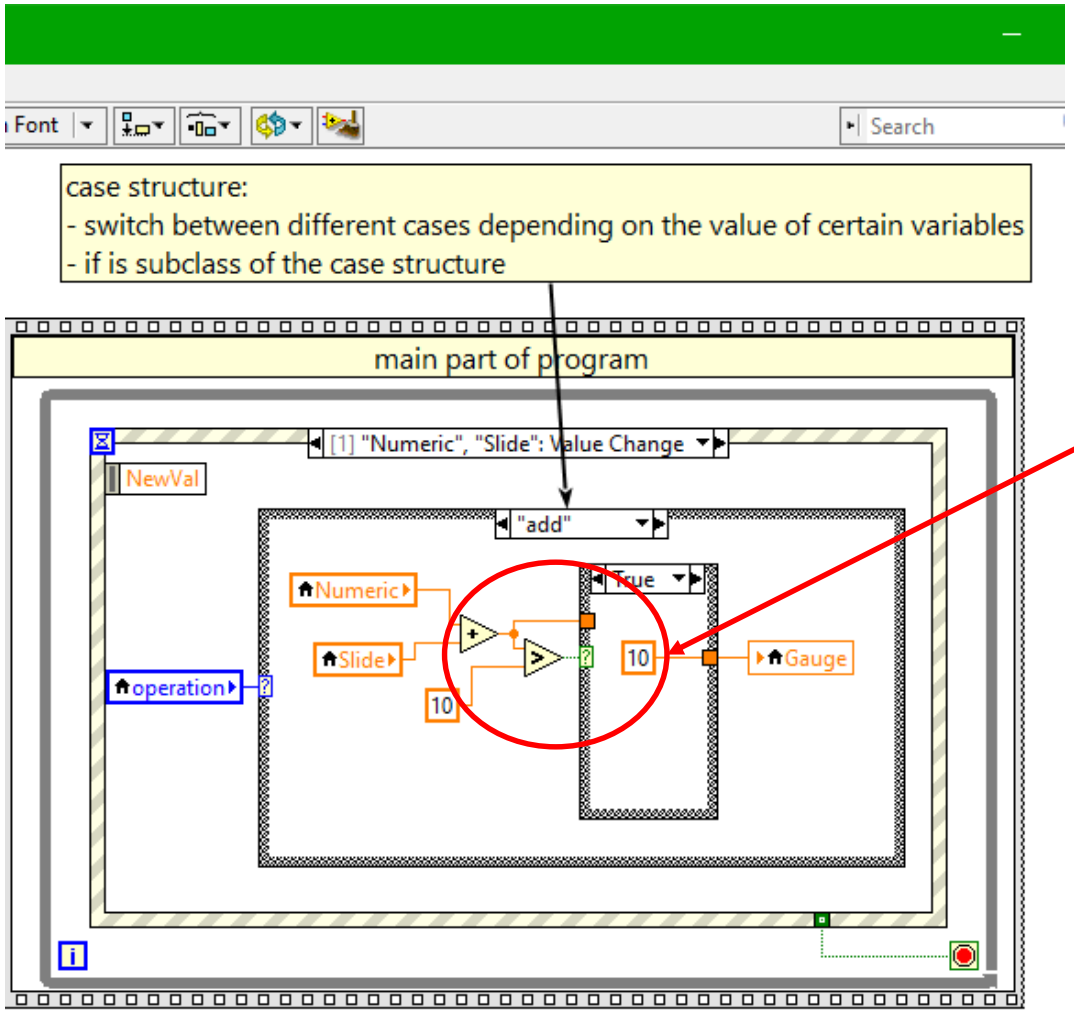
Case structure / if



1) Choose functions depending on values
2) Check *Rings* and *Enums*

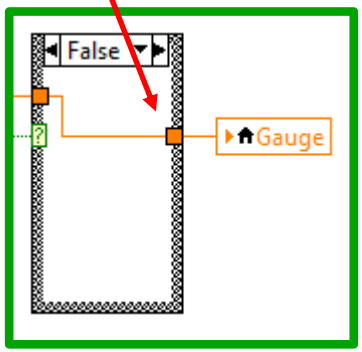


Case structure / if

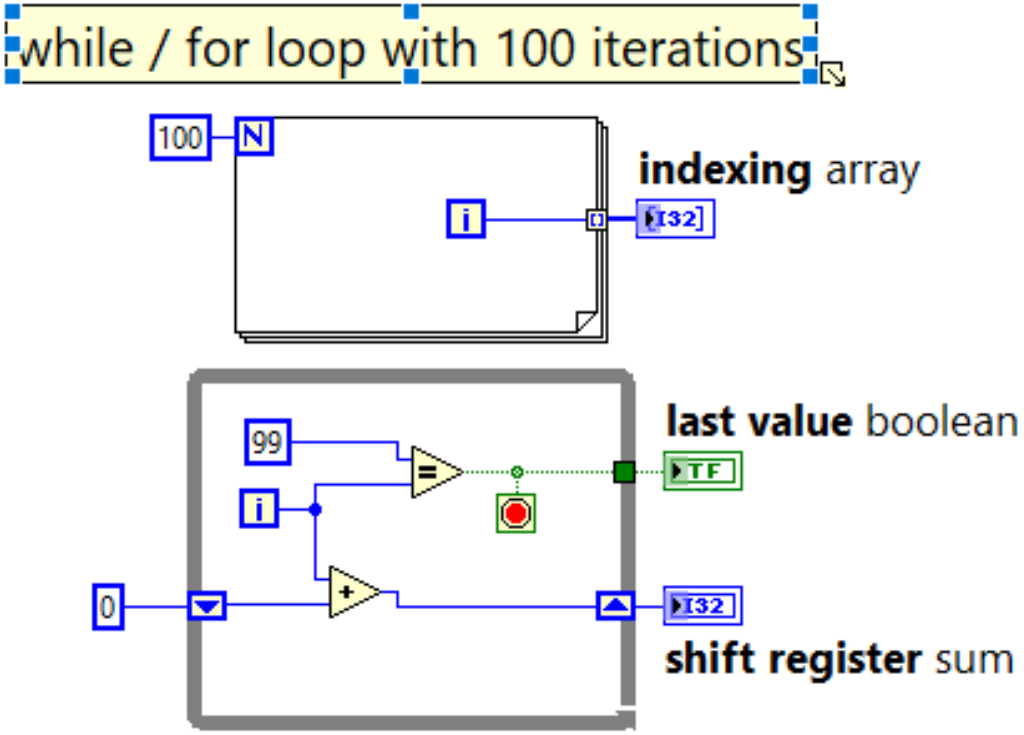


1) Choose functions depending on values
2) Check *Rings* and *Enums*

3) Realize ***if*** with Booleans
4) Create output depending on case



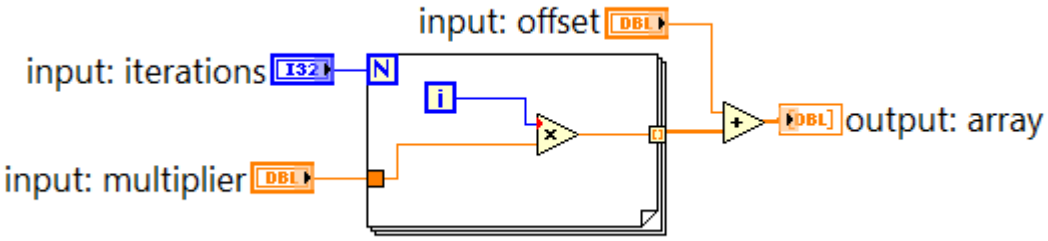
While and for loops



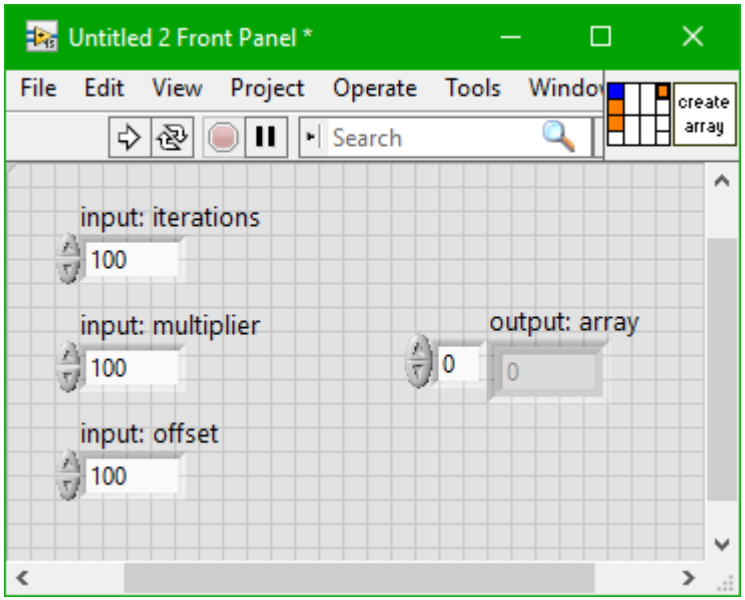
1) **While:** Repeat action until condition is met
2) **For:** Repeat action a predefined number of times (condition possible)
3) Helpful in- & output types
   - Last value
   - Indexing (works on both sides)
   - Shift register

Reuse your code: sub VIs

Create functions you can reuse at different locations in your program



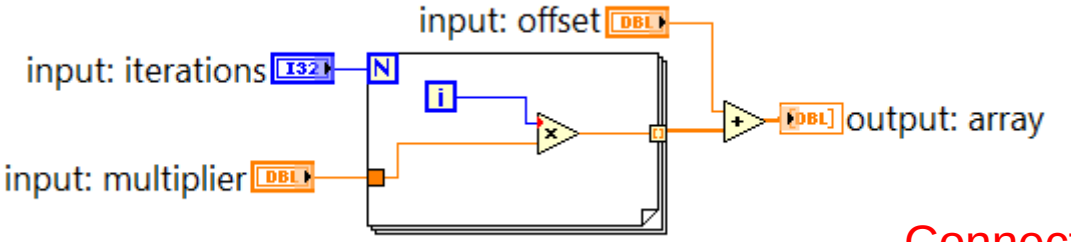
- Often without user interaction
- No need for event structure

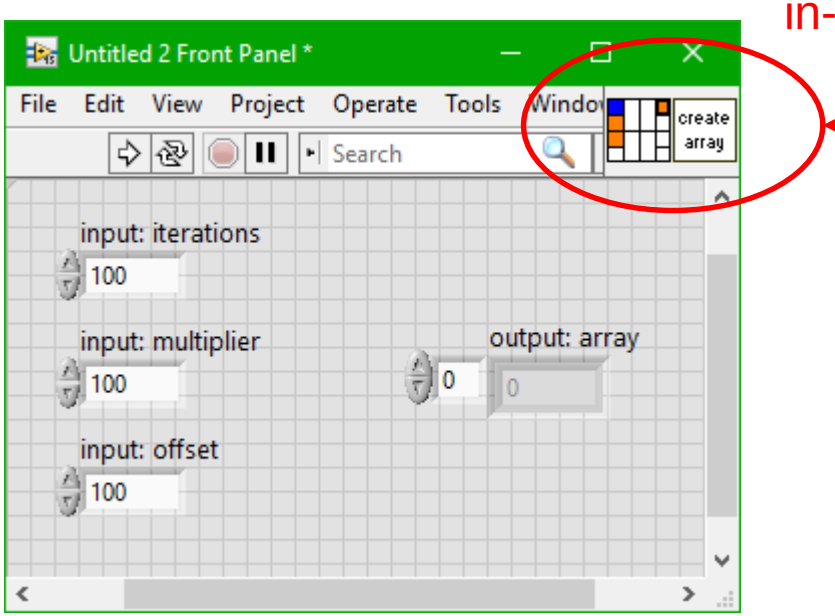


1) Write sub VI

Reuse your code: sub VIs

Create functions you can reuse at different locations in your program

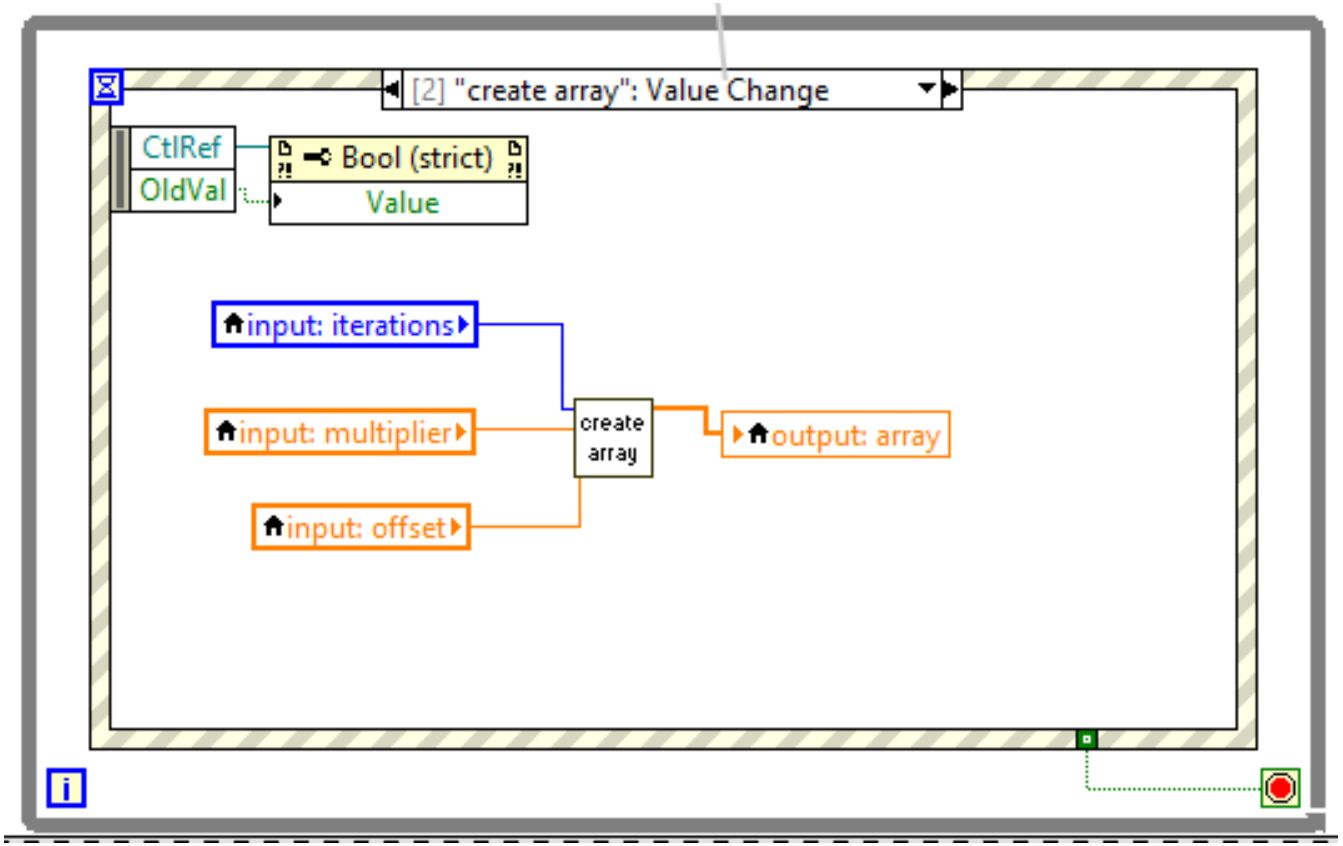




Connector pane: control in-/output variables

1) Write sub VI
2) Define in- & output via connector pane at the front panel

Reuse your code: sub VIs



3) Place sub VI in main VI via drag&drop from the project window
4) Connect in- and output variables

Extend your first VI

1) Create a new VI *calculator* with
   - Two numerics as input
   - One numeric as output
   - Option to choose the mathematical function
   - No need for while loop or event structure
2) Create a new VI *loop* that
   - Adds all even numbers up to a user-specified maximum
   - Creates an array
     - with numbers starting at a user-defined minimum
     - with constant stepsize between entries
     - with user-defined number of elements
   - Which in- and outputs are required?
3) Use these VIs as sub VIs in your first *main.vi*

Overview

- Getting started 1
  - Basic concepts and core components
  - Control structures 1 (loops, if/case structure, …)
  - ➢ Develop your first program

- Getting started 2
  - Control structures 2
  - subVIs: reusable code
  - ➢ Extend your code

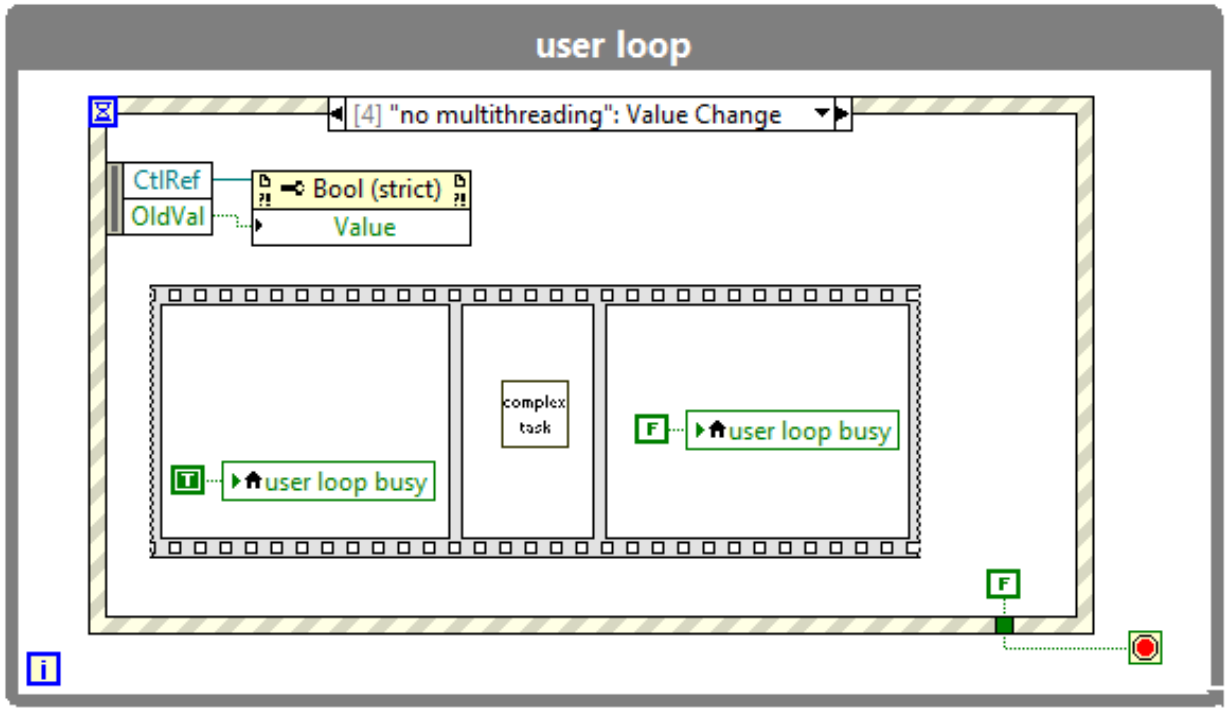
- Quick tour through the LabVIEW toolbox
  - Multithreading
  - Data storage and visualization
  - Device I/O

- ➢ Hands on! Control devices

Multithreading and communication

Multithreading?

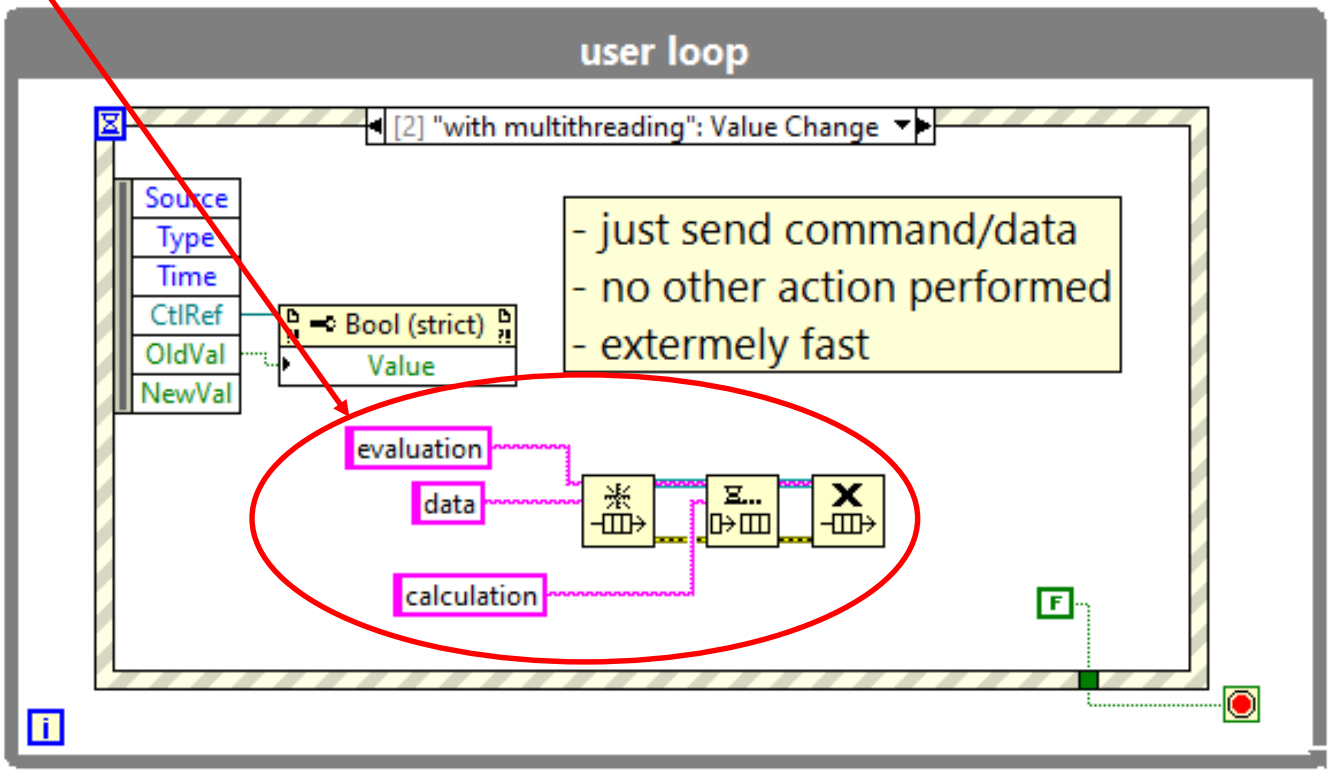
- Parallel execution of code
- Keep program responsive to user interactions
- Separate *slow* operations from *fast* operations and *waiting*
- Build modular software



Multithreading and communication

Multithreading in LabVIEW?

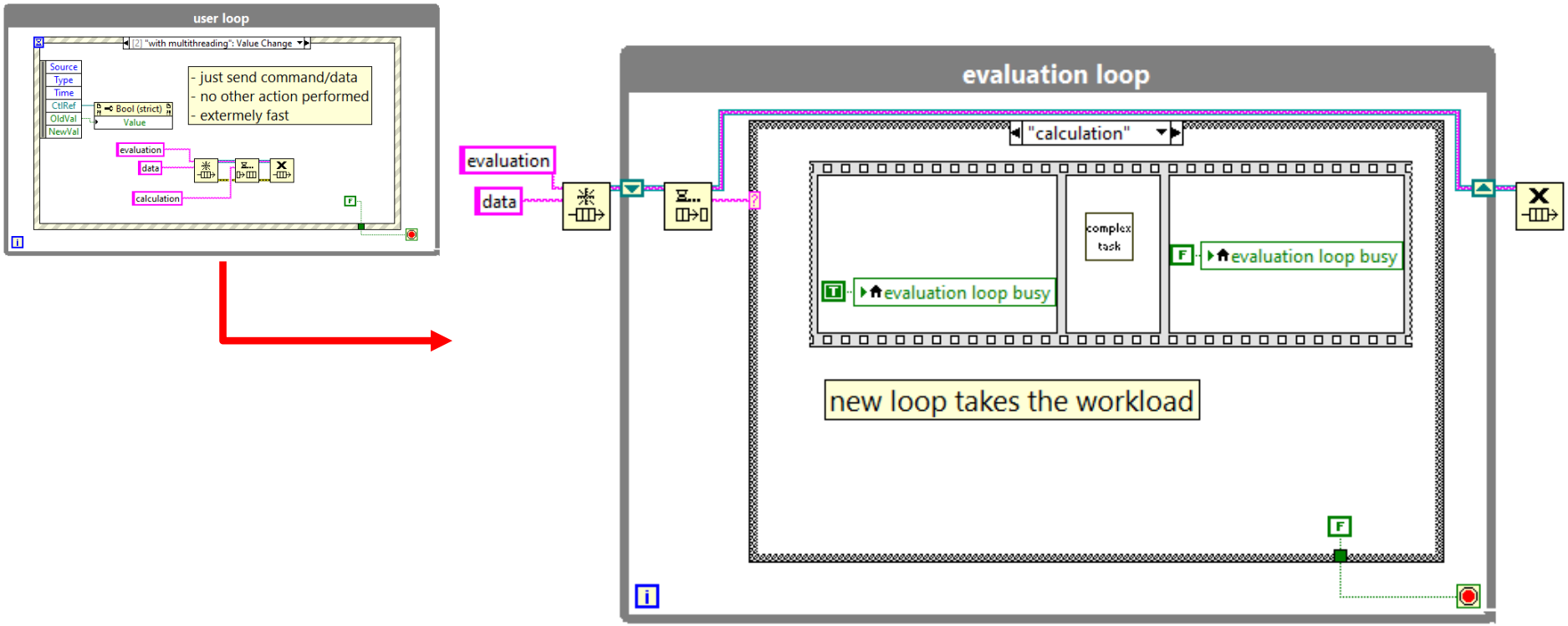
- Use several while loops
- Use *queues* and *notifiers* to communicate
- Similar to a function call



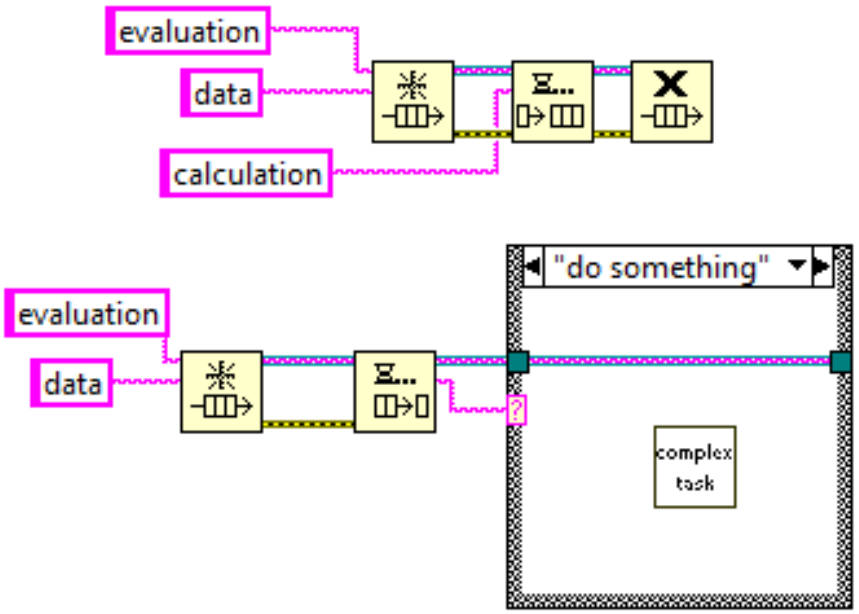
Multithreading and communication

Multithreading in LabVIEW?

- Use several while loops
- Use *queues* and *notifiers* to communicate
- Similar to a function call



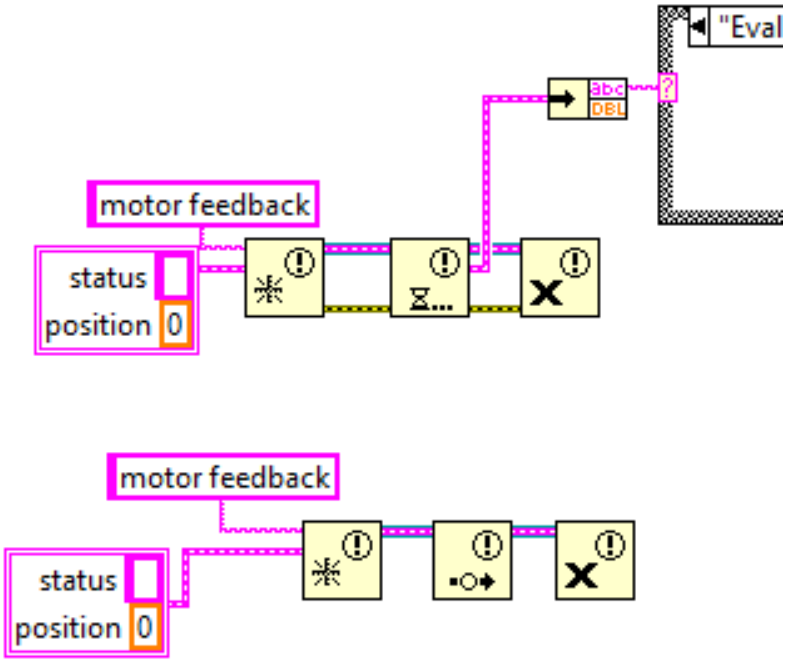
Multithreading and communication



LabVIEW queues

- List of user-defined *data type*
  - Example: String
- Can be accessed via *name* project-wide
  - Example: *evaluation*
- Many senders BUT only one recipient
- Basic functions:
  - Obtain
  - Enqueue: sender
  - Dequeue: recipient
  - Release

Multithreading and communication



LabVIEW notifiers

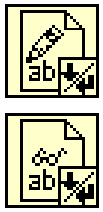
- Single element of user-defined *data type*
  - Example: cluster of String & Float
- Can be accessed via *name* project-wide
  - Example: *motor feedback*
- Many senders AND many recipient
- Basic functions:
  - Obtain
  - Send: sender
  - Wait on: recipient
  - Release

Data storage

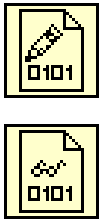
- **Text files**
  - Human-readable format
  - Easy import into several programs
  - Only simple data sets (tables)
  - Huge amount of data
- **Binary files**
  - Data format needs to be documented / maintained
  - Custom import functions required
  - Minimum data size
- **Complex file formats**
  - Technical Data Management Streaming (TDMS), NI
  - Hierarchical Data Format (HDF), HDF group (open source)
  - Database formats

Data storage & visulization

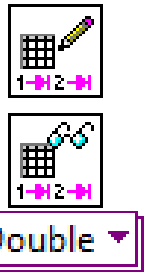
Read and write text files



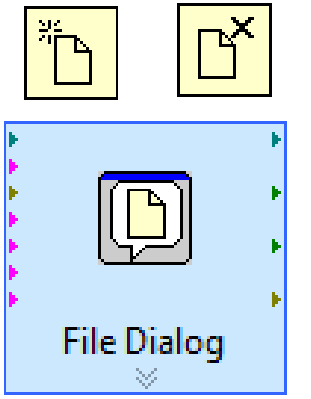
Read and write binary files



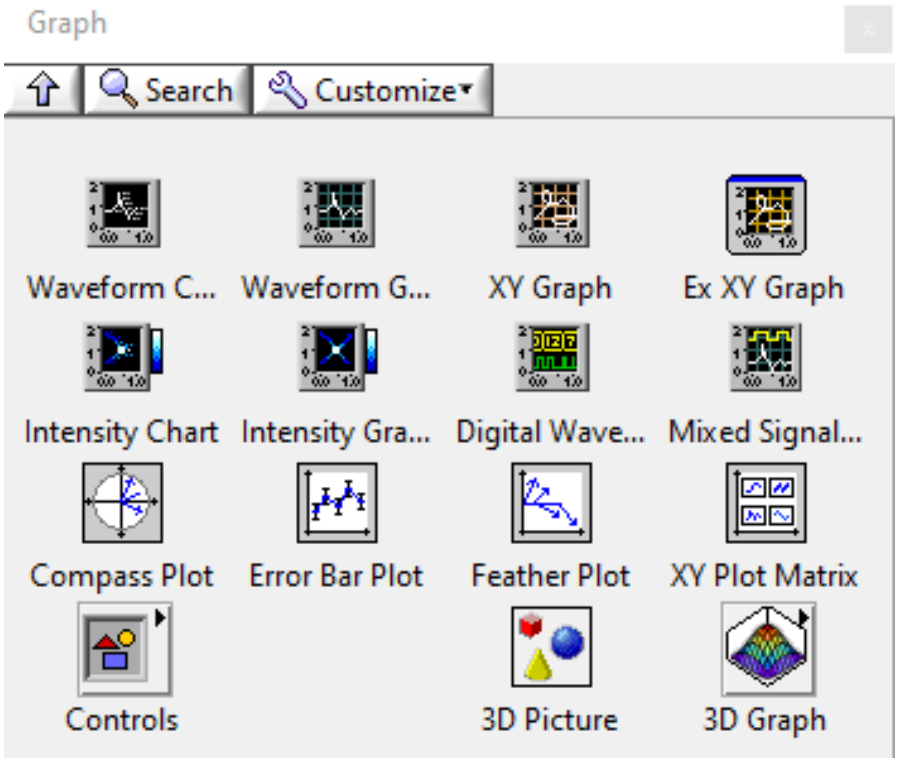
Read and write spreadsheet files



General file I/O



- Several tools available
- File I/O palette
- Graph palette

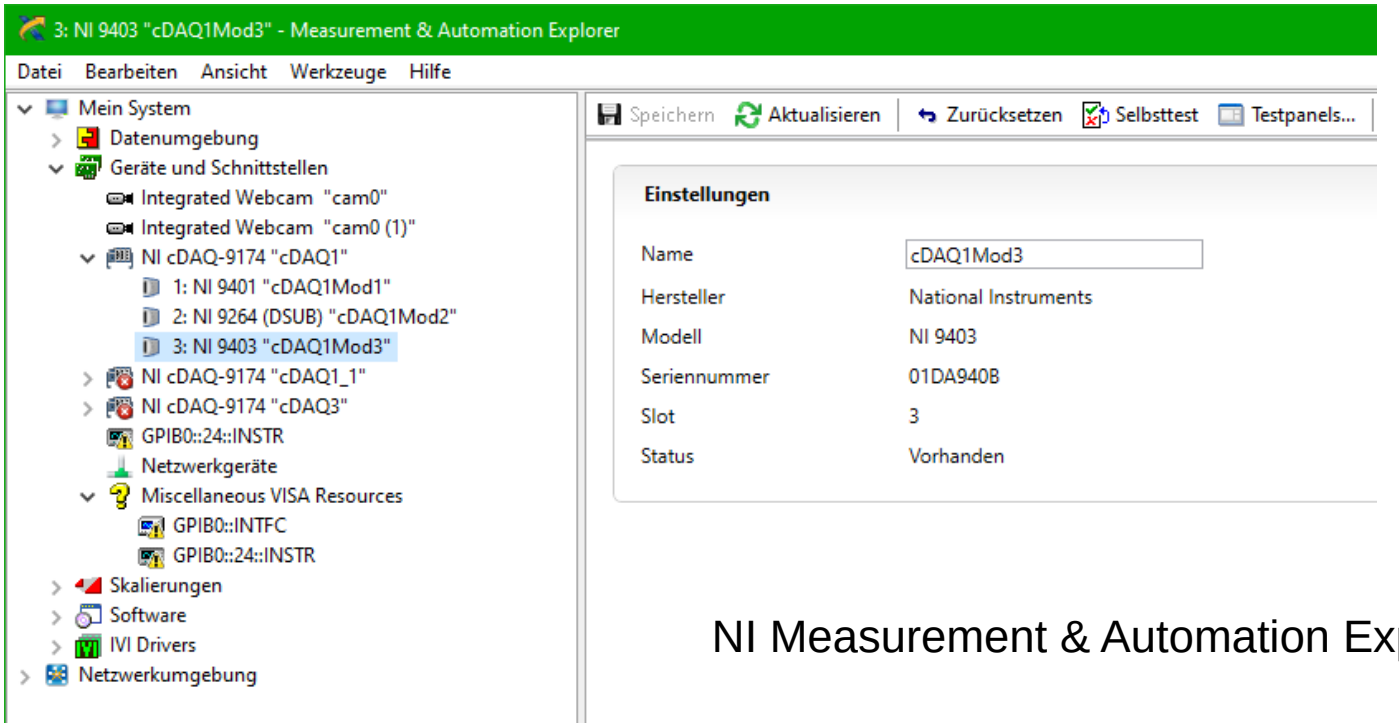


Hardware I/O

Typical hardware / software interfaces for lab devices

- **VISA protocol** (virtual instrument software architecture):
  - Well documented and easy syntax (SCPI commands)
  - Identify device with command **IDN?\n*
  - Major hardware companies use it
  - Major hardware companies offer VIs for device control
  - Typically supports several hardware interfaces
    - LAN, GPIB, USB, serial interface
  - Directly accessible from NI MAX

Hardware I/O



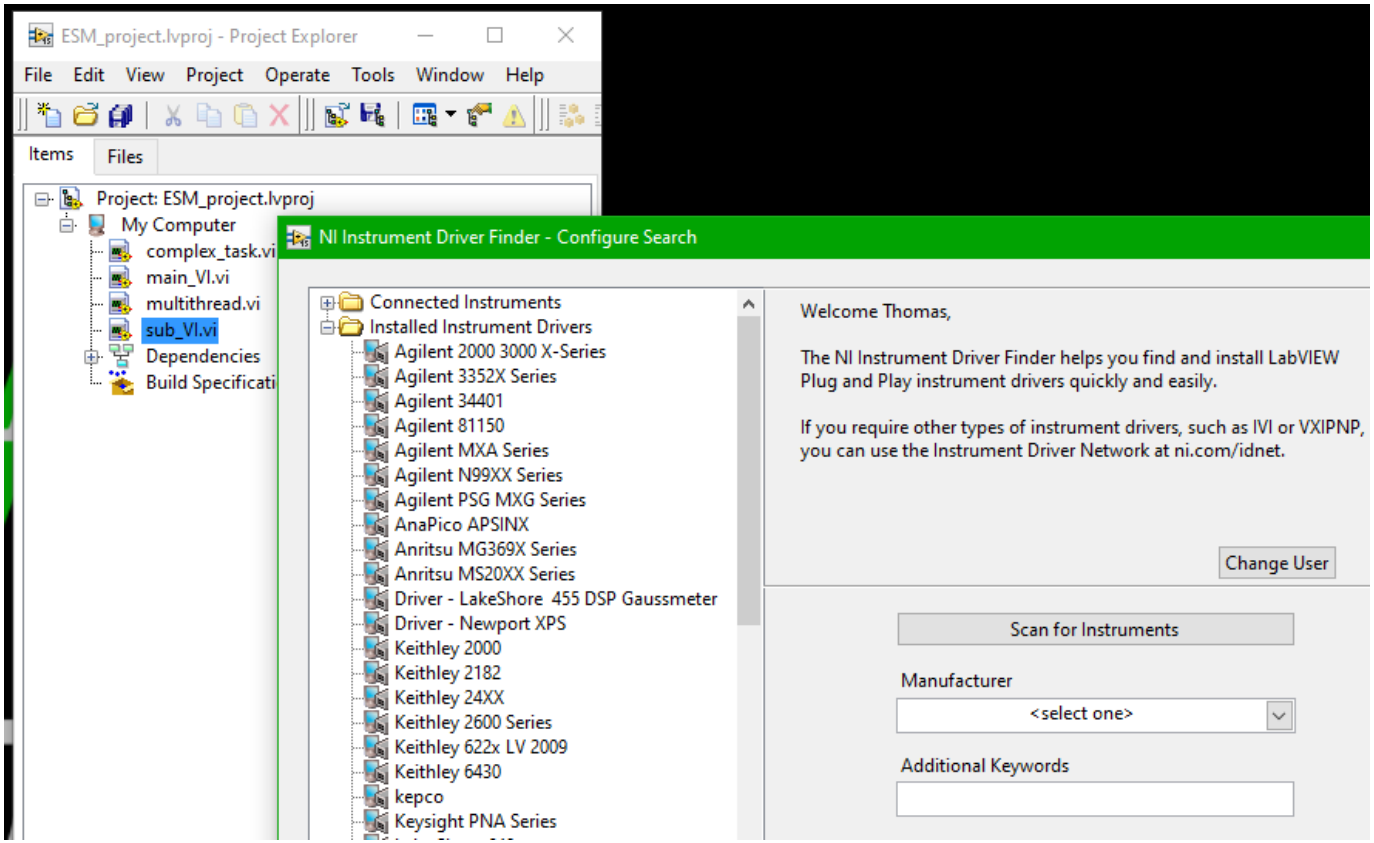
NI Measurement & Automation Explorer – NI MAX

- Find available hardware
- Configure interfaces
- Communicate in interactive session

Hardware I/O

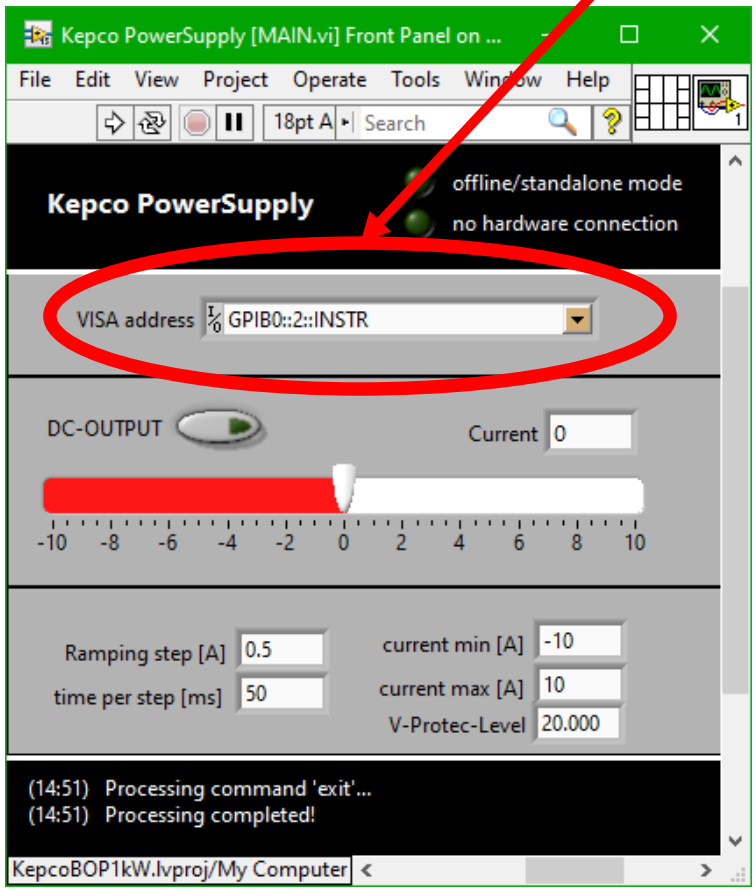
How to get drivers:

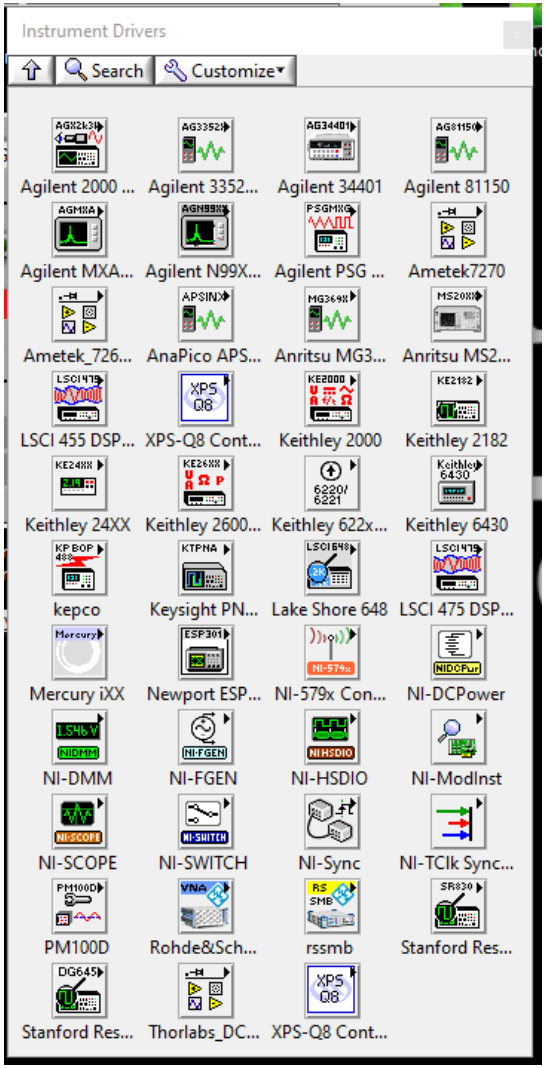
- Web search
- LabVIEW menu → Tools → Instrumentation → Find Instruments Drivers…



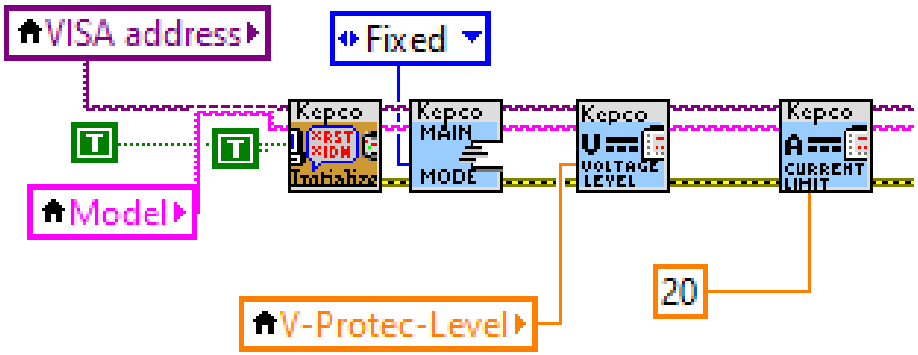
Hardware I/O

VISA address

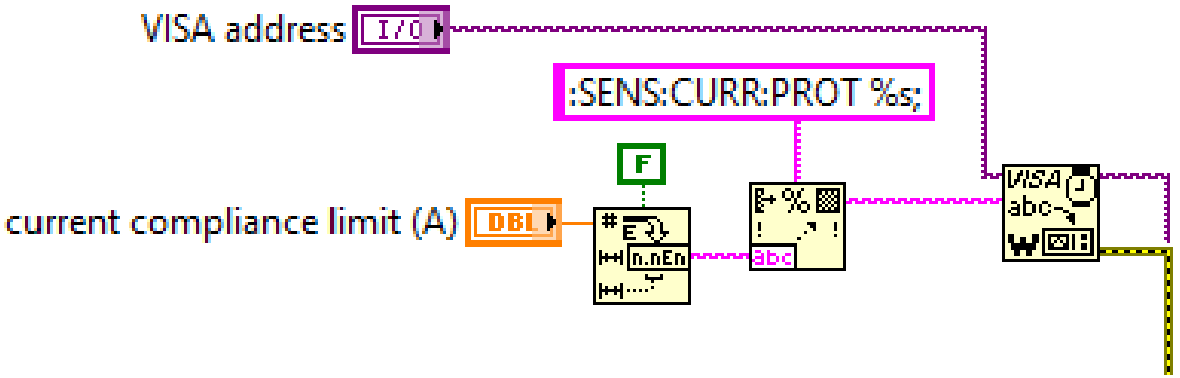




Hardware I/O



- Use manufacturer VIs
- Use basic VISA read/write functions



Hardware I/O

Typical hardware / software interfaces for lab devices

- **VISA protocol** (virtual instrument software architecture)
- **DLLs** (dynamic link library)
  - External programs with in-&output values
  - Manufacturers often provide drivers and VIs
- **Serial interface**
  - Sends/receives strings
  - VISA or custom syntax
  - Directly accessible from NI MAX
- **ActiveX, .NET**

Hardware I/O

Serial interface

- Sends/receives strings
- VISA or custom syntax
- Directly accessible from NI MAX

Arduino with servo motor

1. Identify device in NI MAX
2. Test commands interactively
3. Write VI for the control of the servo

Optional: install new device drivers via the LabVIEW runtime menu

Available hardware

- **DAQ hardware:** data acquisition and generation of analog/digital signals
  - LabJack: communication via DLL with manufacturer Vis
  - Control of Kepco power supply via control voltage
  - Readout out Hall sensor output voltage
- **Linear/rotary stages** from Thorlabs via ActiveX
- **DC source** from Keithley (VISA)
- **Nanovoltmeter** from Keithley (VISA)
- **DC source** from Keisight (VISA)