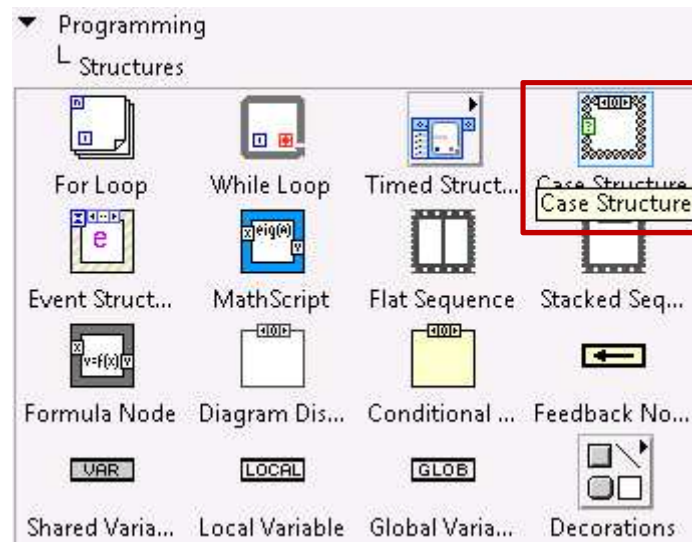


# LabVIEW Lecture 2

Ertugrul Karademir

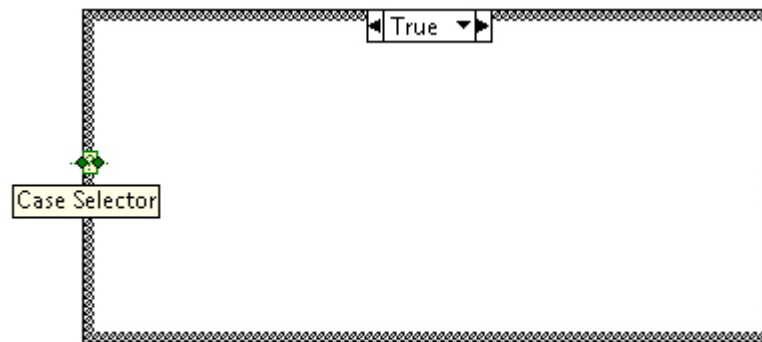
# Case Structure

- Just like if...elseif...else

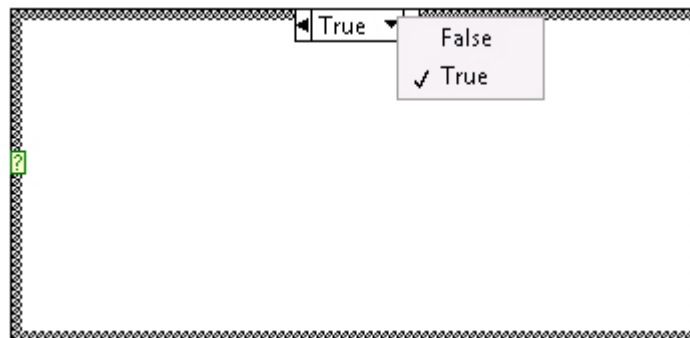


# Case Structure

- Connect the case port

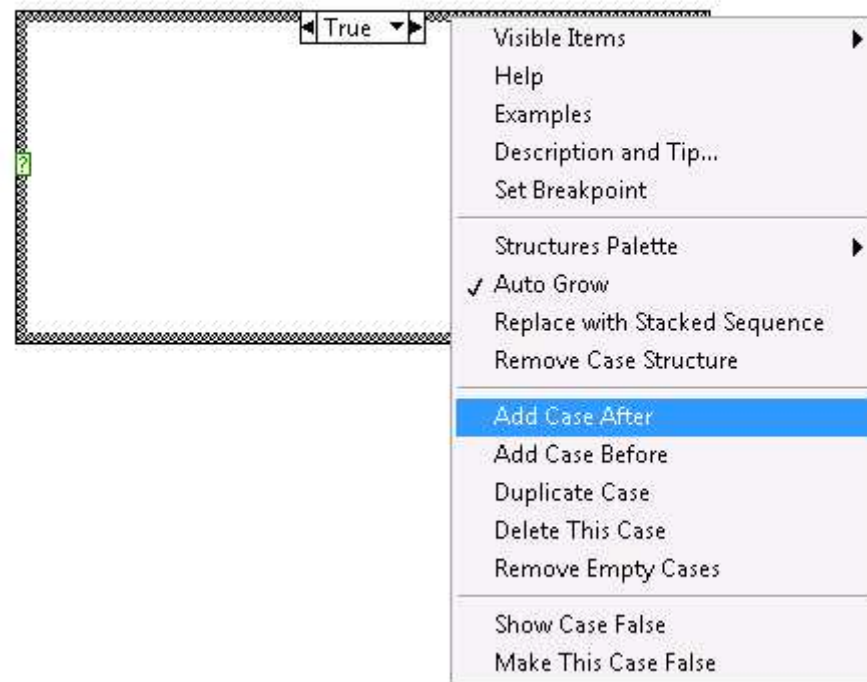


- Choose the condition you want to program



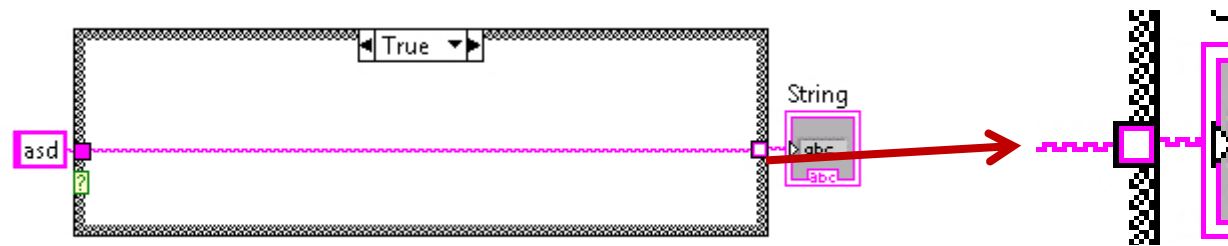
# Case Structure

- You can add more cases by right clicking

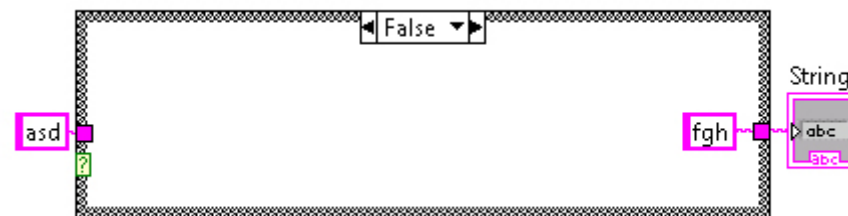


# Case Structure

- All terminals for all cases must be connected



This is empty because, False condition is not wired



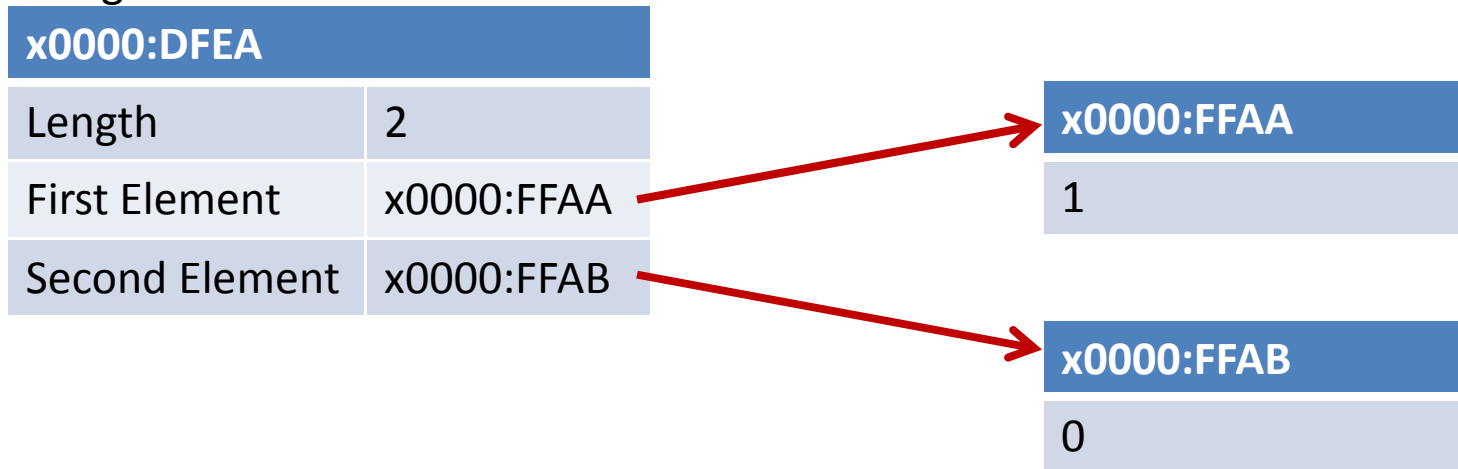
# Variable Types

- There are basically three kinds of variable types:
  1. Integers: One memory register is enough to store. Bit value of the register defines the limits of cardinality.
  2. Floating point: more than one memory register is needed (one to store base, one to store significant digits, etc.)
  3. String: Textual information. It is stored much like Arrays. Actually a string is an array of characters (which are integers with special meanings).

# Variable Types

- When number 10 is stored as a String it is represented more or less like this:

String "10"



# Variable Types

- When number 10 is stored as an Integer it is represented more or less like this:

Integer "10"

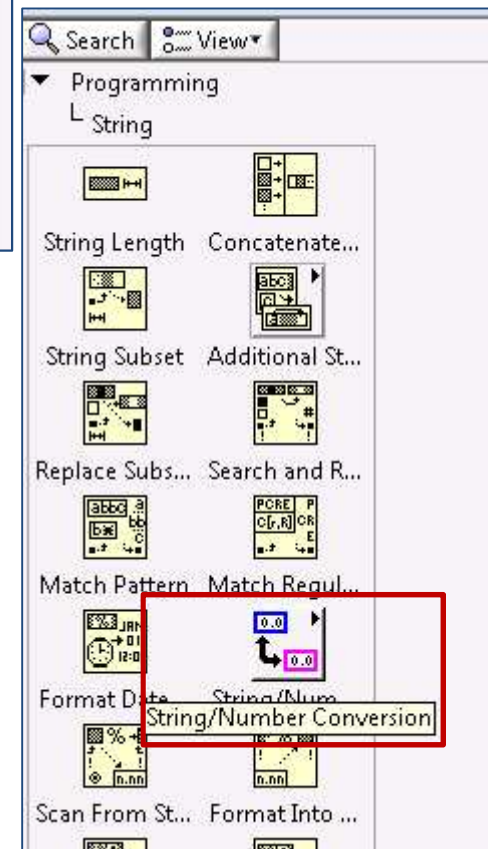
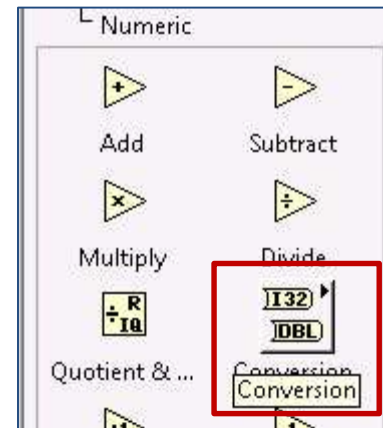
**x0000:ADFF**

10



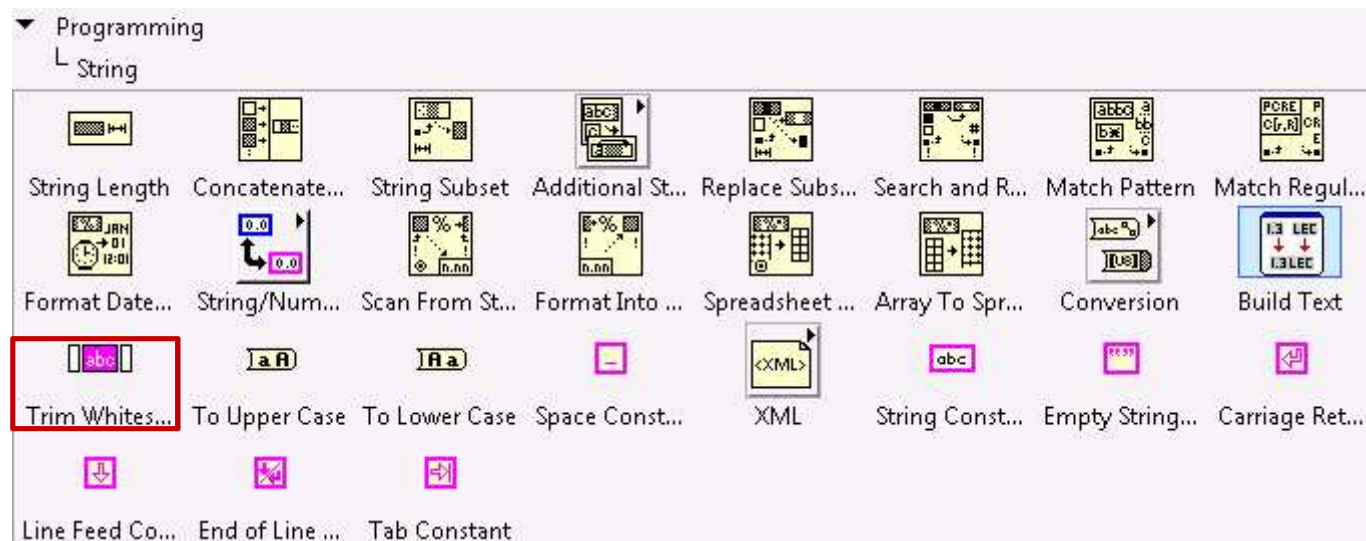
# Variable Types

- Controls we use to communicate with instruments need and produce String data.
- Numerical values need to be converted appropriately (type casting).



# String Operations

- Find your way in string palette
- It's intuitive
- Don't forget to trim whitespace out of returned data from the instrument

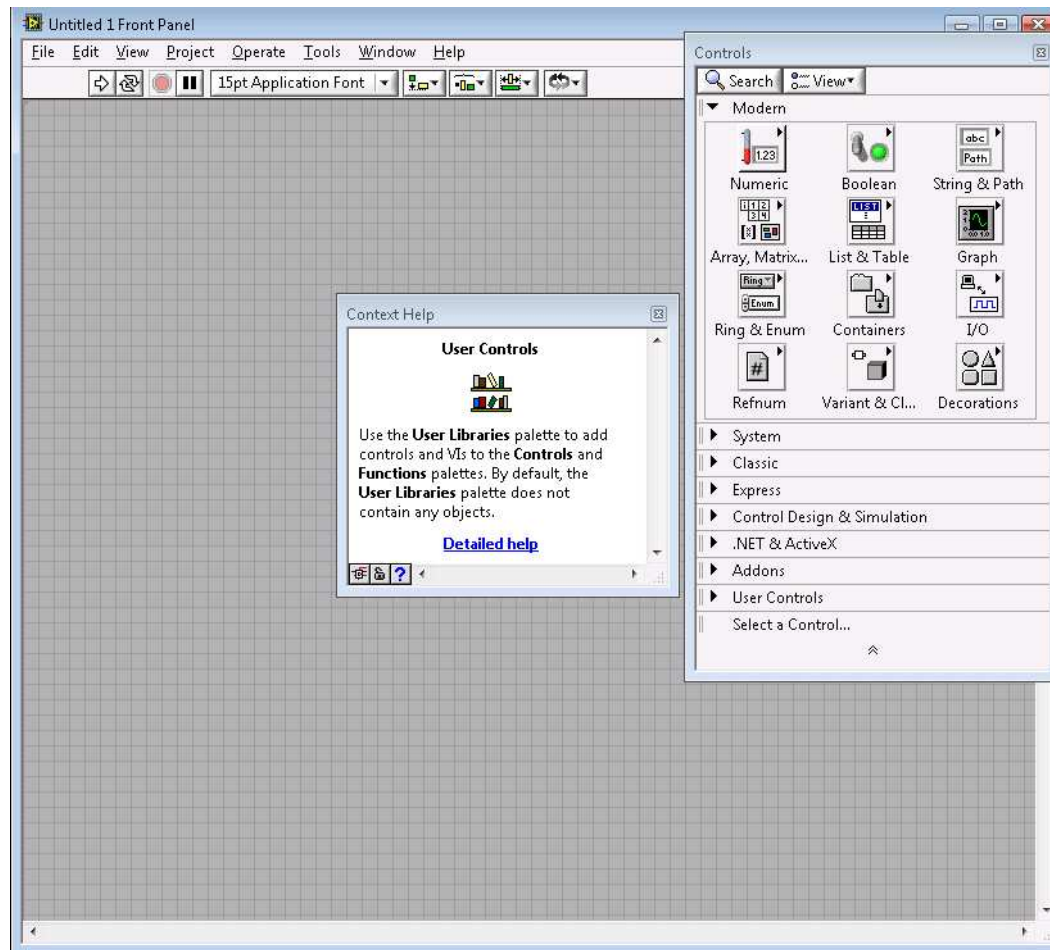


# More Complex Topics

- From here on we focus on more complex topics:
  - Arrays
  - File Input Output
  - Data Acquisition

# Modern Palette

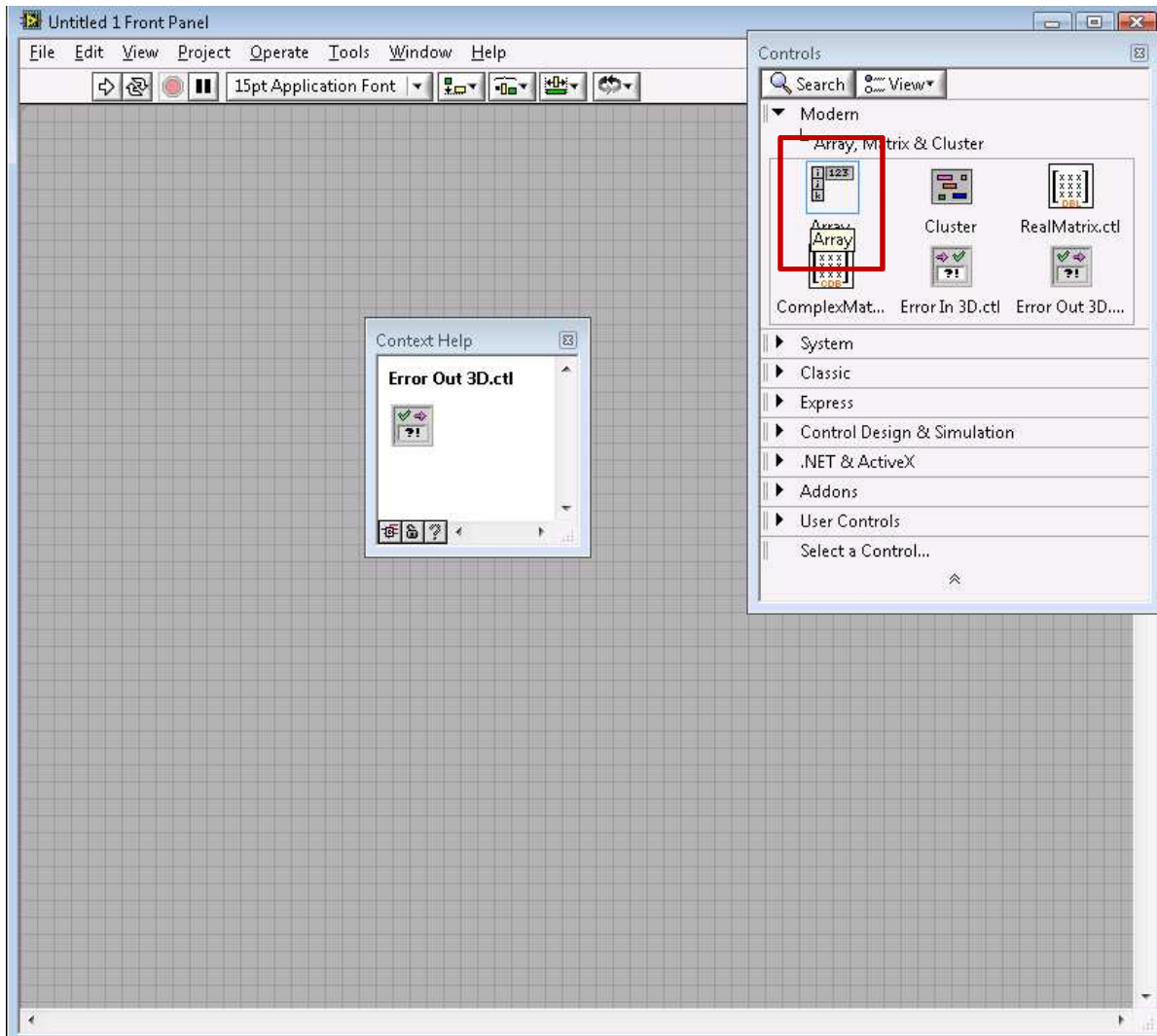
Sometimes we need more than what Express palette offers



# Arrays

- Arrays are created as empty shells
- In order to assign a type you should drop in a control.
- Array casts into the dropped control's type.

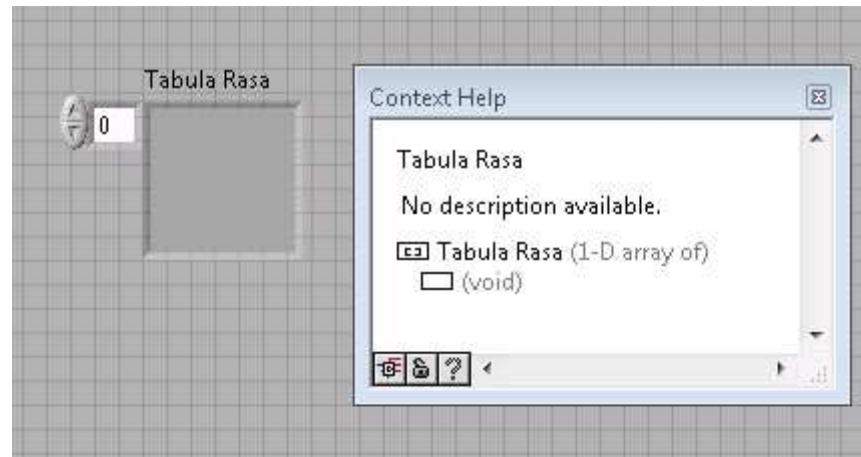
# Arrays



Array, Matrix & Cluster palette has the empty array shell.

# Arrays

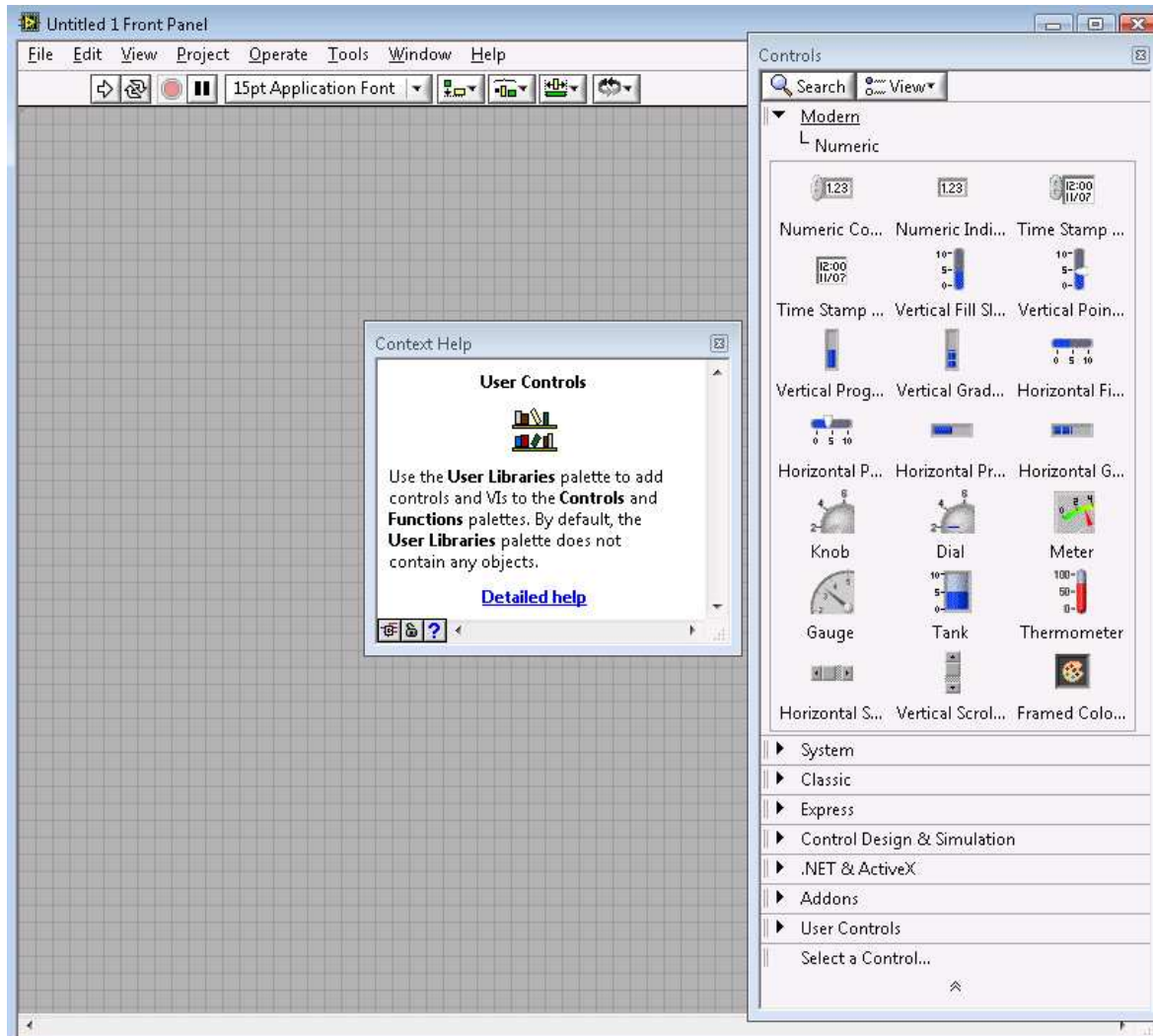
Create an empty array shell



LabVIEW creates a proxy for that array in block diagram



# Arrays

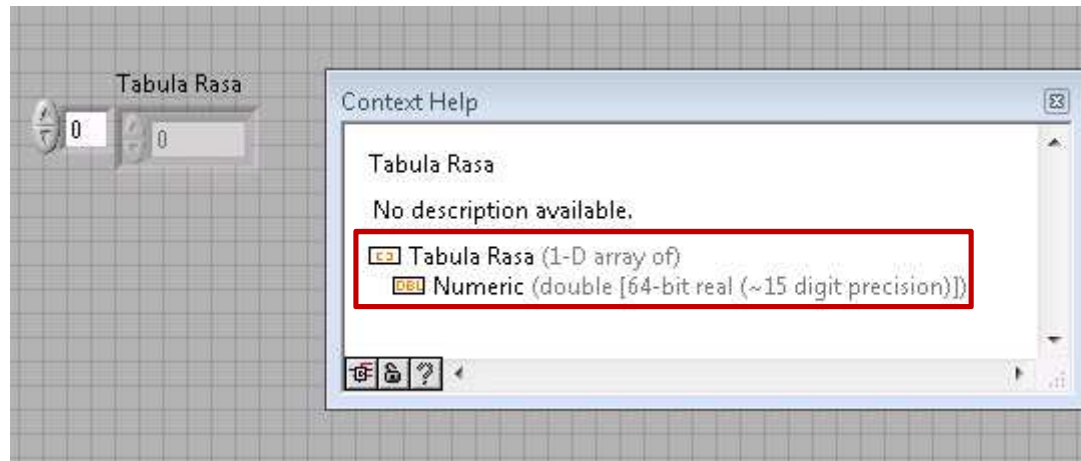


- Invoke Numeric Palette under Modern
- Drop a Numeric Control inside the empty box of the Array shell

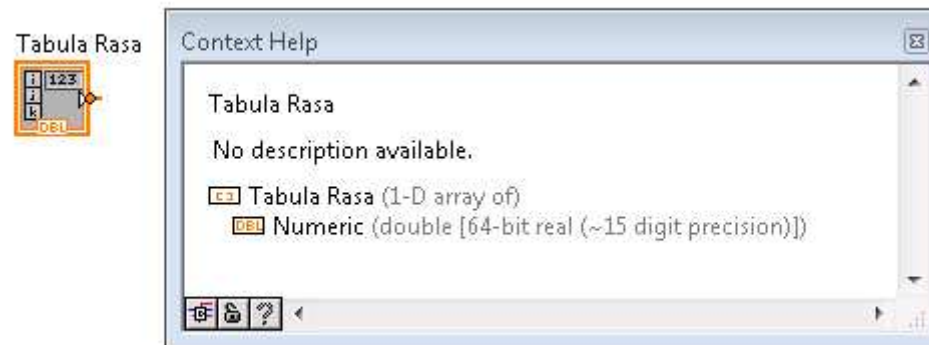


# Arrays

Now the array has a type

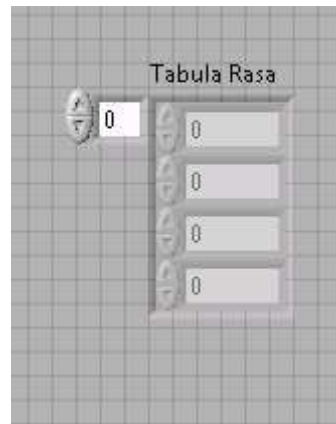


So does its proxy



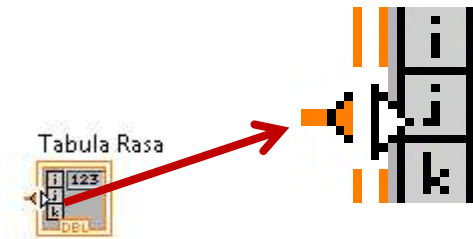
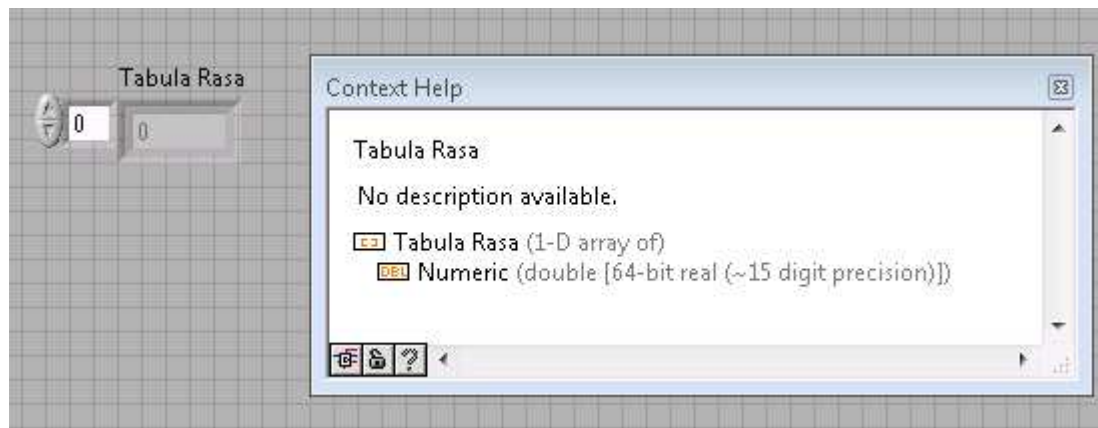
# Arrays

- You can expand the array container to show more elements at once
- Or click to the pager buttons to change the starting index



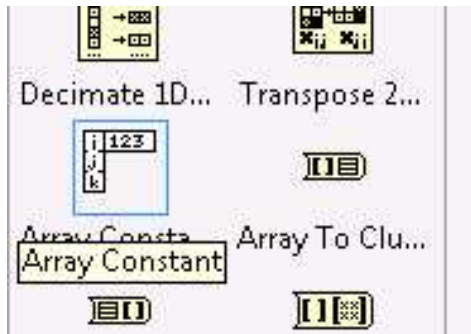
# Arrays

- Note that there is no array INDICATOR
- That is because you indicate that an Array is an indicator by dropping in an indicator inside that array shell



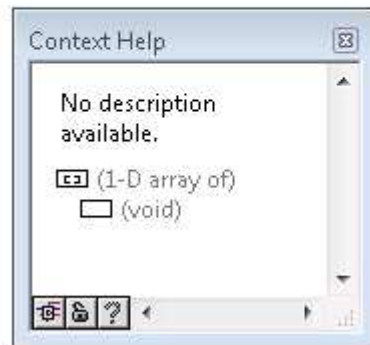
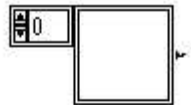
It is an indicator

# Arrays

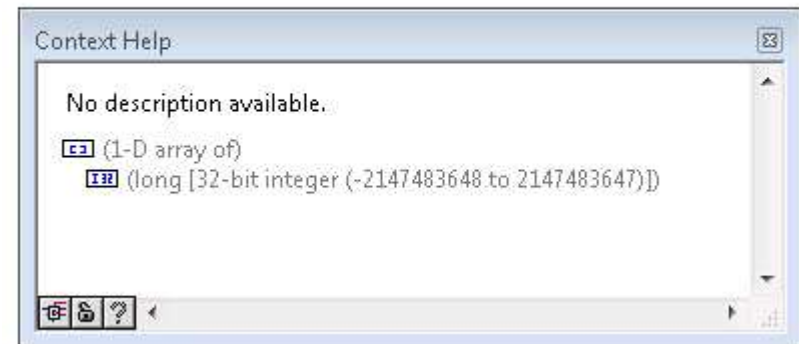


You can also create Array constants

Programming Palette > Array Palette >  
Array Constant

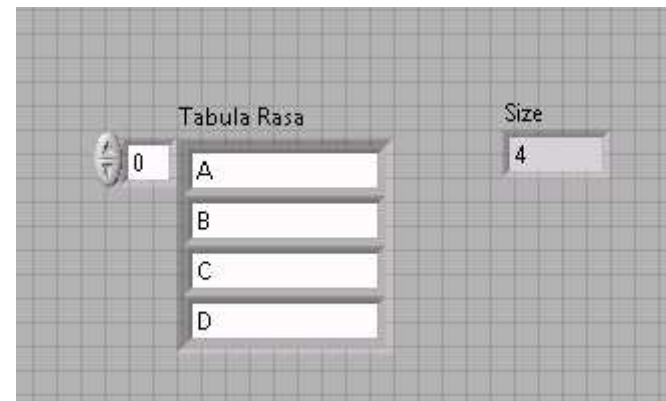
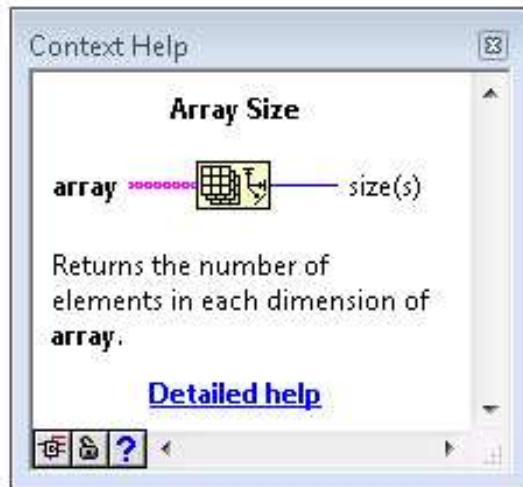
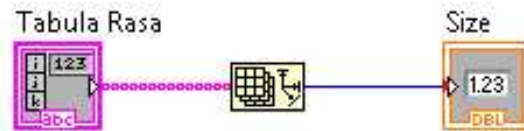


Drop an integer  
constant inside and get  
integer array



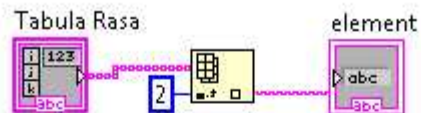
# Arrays

- Arrays palette has many useful Array functions.
- Like getting the size of an array



# Arrays

Get an element at a specific index.



Context Help

### Index Array

**n-dimension array** — [Grid Icon] — **element or subarray**  
**index 0** — [Grid Icon] —  
**index n-1** — [Grid Icon] —

Returns the **element or subarray** of **n-dimension array** at **index**.

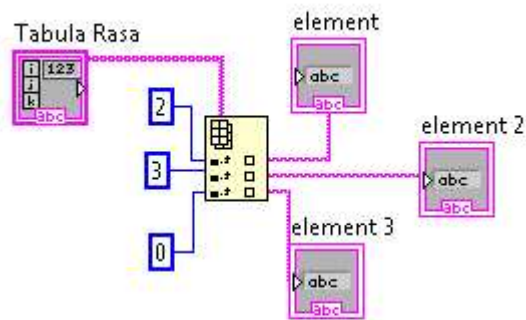
[Detailed help](#)

The context help window for the 'Index Array' block. It features a title bar 'Context Help' with a close button. The main content area shows the block's icon (a grid with a plus sign) and its inputs: 'n-dimension array', 'index 0', and 'index n-1'. An arrow points from the block to the text 'element or subarray'. Below this, a descriptive sentence reads: 'Returns the element or subarray of n-dimension array at index.' A blue link for 'Detailed help' is provided. At the bottom, there are navigation icons: a search icon, a home icon, a question mark icon, and a back arrow.

A screenshot of a software interface on a grid background. On the left, a control labeled 'Tabula Rasa' has a vertical slider set to '0'. Below the slider are four stacked input fields containing the letters 'A', 'B', 'C', and 'D'. On the right, a control labeled 'element' has a single output field containing the letter 'C'.


# Arrays


Or several indices (expand the icon)




Context Help

### Index Array


n-dimension array →  → element or subarray

index 0 →  → element or subarray

index n-1 →  → element or subarray

Returns the **element or subarray** of **n-dimension array** at **index**.

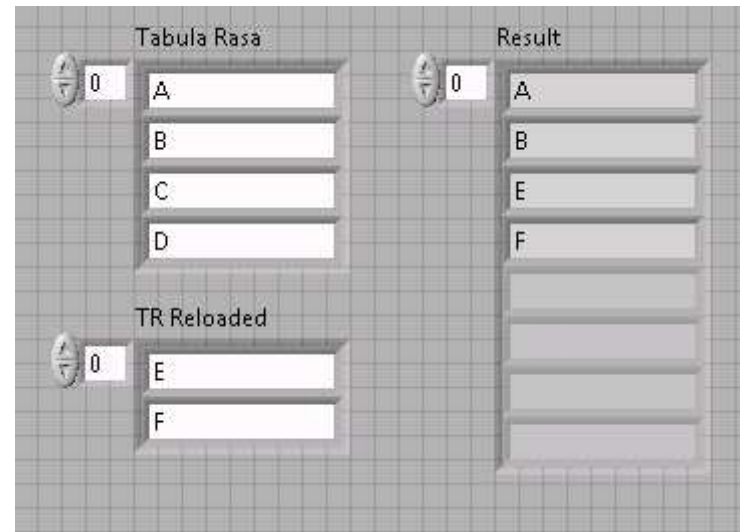
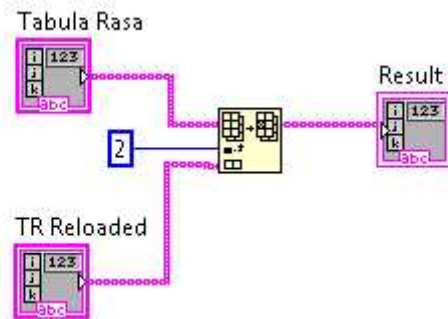
[Detailed help](#)



Tabula Rasa	element
0	C
A	element 2
B	D
C	element 3
D	A

# Arrays

## Replace some elements



Context Help

### Replace Array Subset

**n-dimension array** ———— output array  
index 0  
...  
index n-1  
**new element/subarray**

Replaces an element or subarray in an array at the point you specify in **index**.

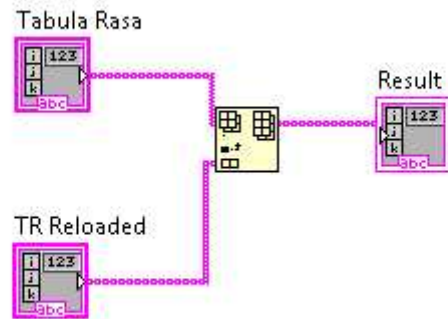
[Detailed help](#)

The screenshot shows the Context Help window for the 'Replace Array Subset' block. It features a diagram of the block with labels for its inputs and outputs. The 'n-dimension array' input is connected to the 'output array' output. The 'index' input is shown with a range from 0 to n-1. The 'new element/subarray' input is connected to the 'output array' output. Below the diagram, there is a text description of the block's function and a link to 'Detailed help'.



# Arrays

## Concatenate



Context Help

### Insert Into Array

The diagram shows the "Insert Into Array" block with the following labels:

- n-dim array**: Input array
- index 0**: First index
- ...**: Ellipsis
- index n-1**: Last index
- n or n-1 dim array**: Input subarray
- output array**: Output array

Inserts an element or subarray into **n-dim array** at the point you specify in **index**.

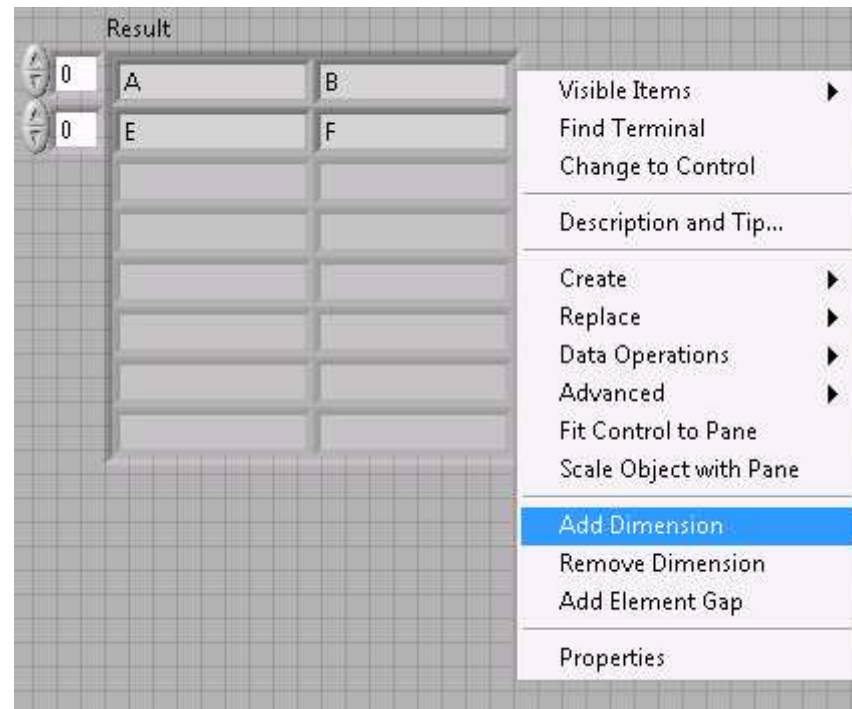
[Detailed help](#)

The screenshot shows a software interface with a grid background. It features three main components:

- Tabula Rasa**: A vertical list of four input fields containing the letters A, B, C, and D. A scroll wheel on the left is set to 0.
- TR Reloaded**: A vertical list of two input fields containing the letters E and F. A scroll wheel on the left is set to 0.
- Result**: A vertical list of six output fields containing the concatenated letters A, B, C, D, E, and F. A scroll wheel on the left is set to 0.

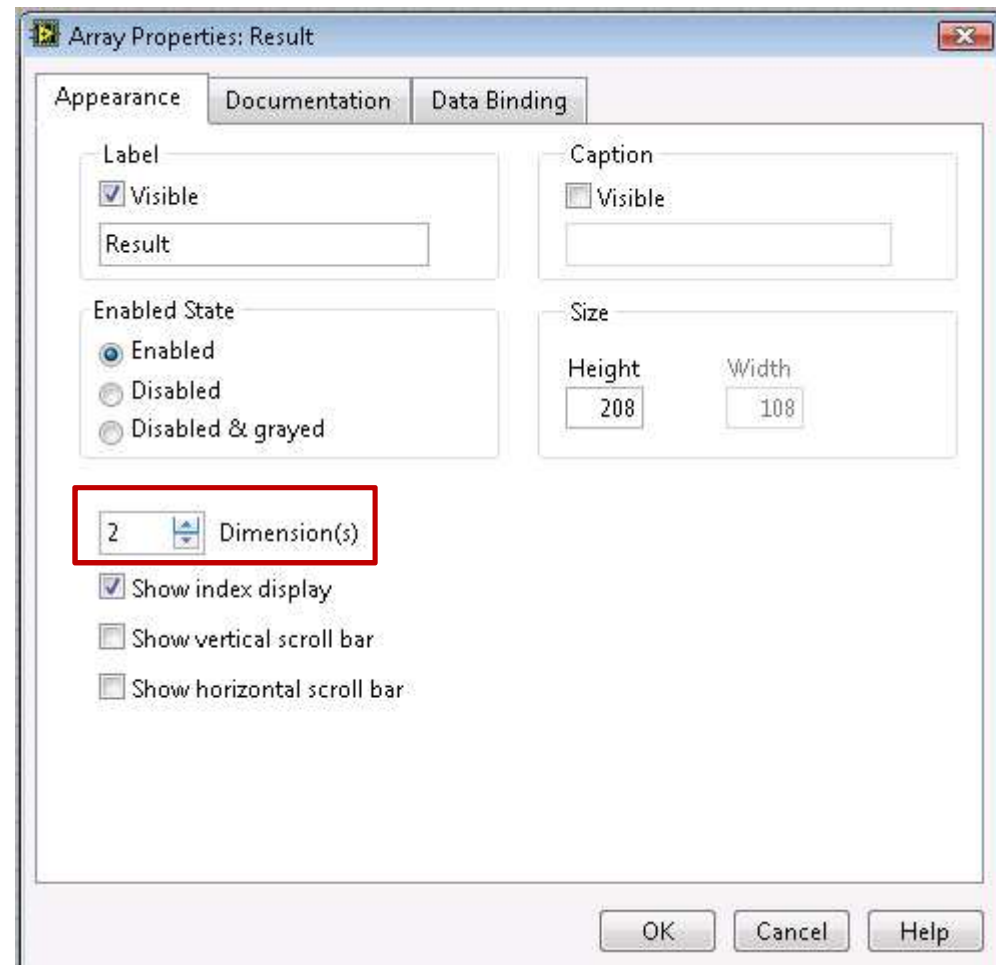
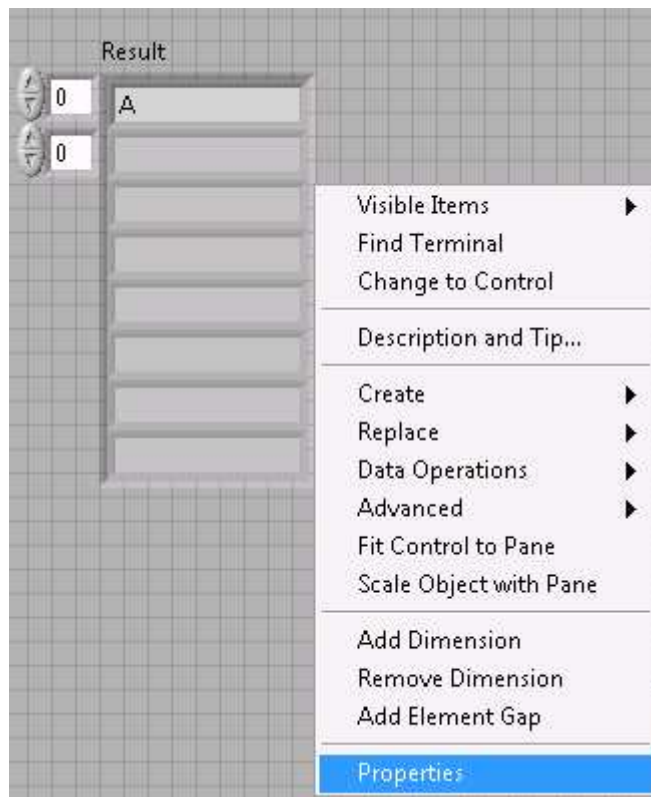
# Arrays

You can add dimensions to an array by right clicking onto the container



# Arrays

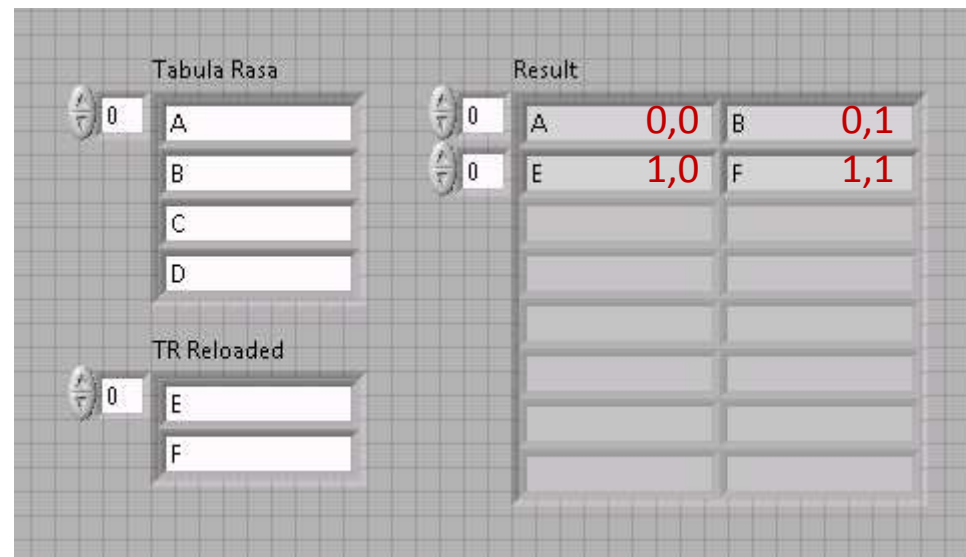
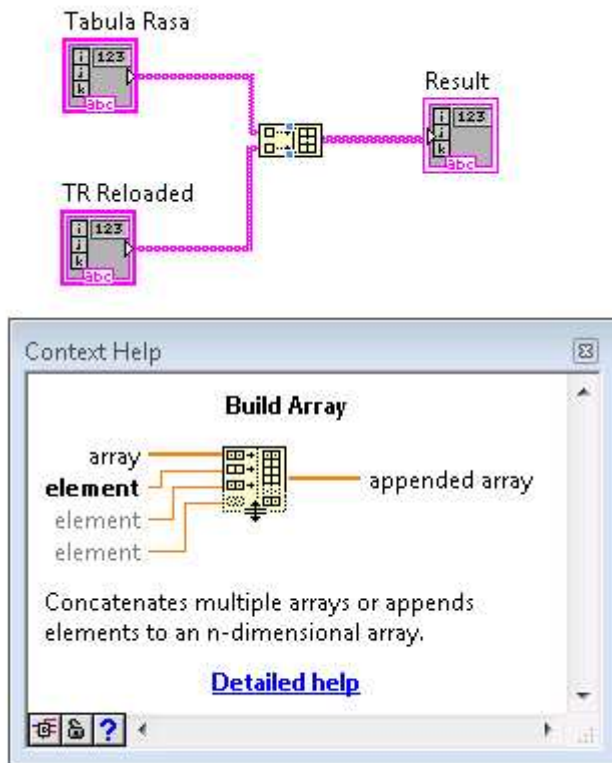
Or from array properties.



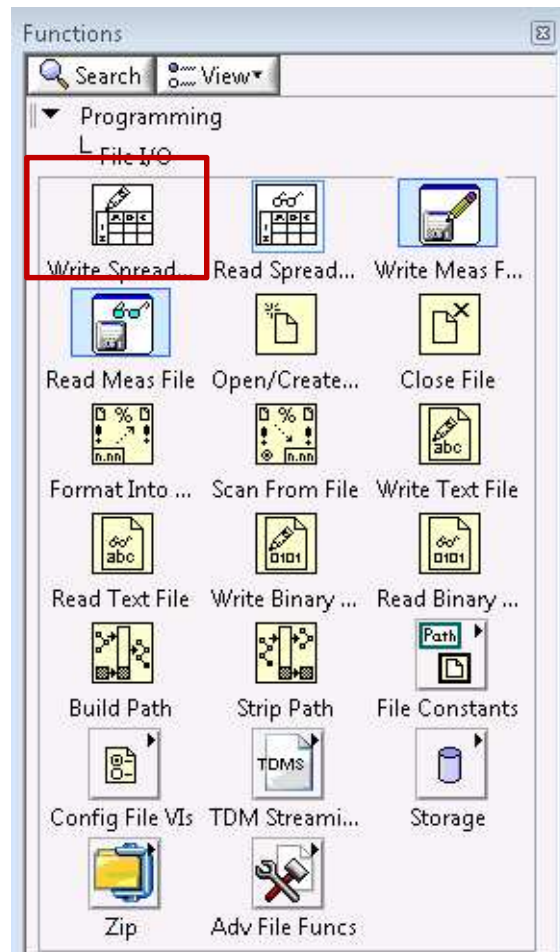
# Arrays

You can build more dimensional arrays by using Build Array function

Note that front panel representation has changed into row-wise indication rather than column-wise.

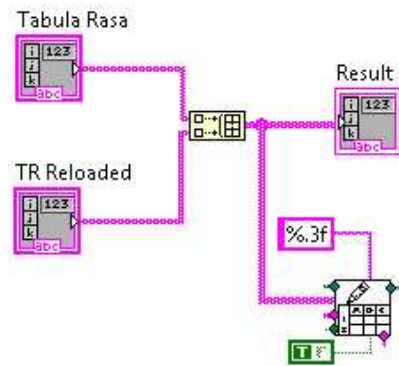


# File I/O



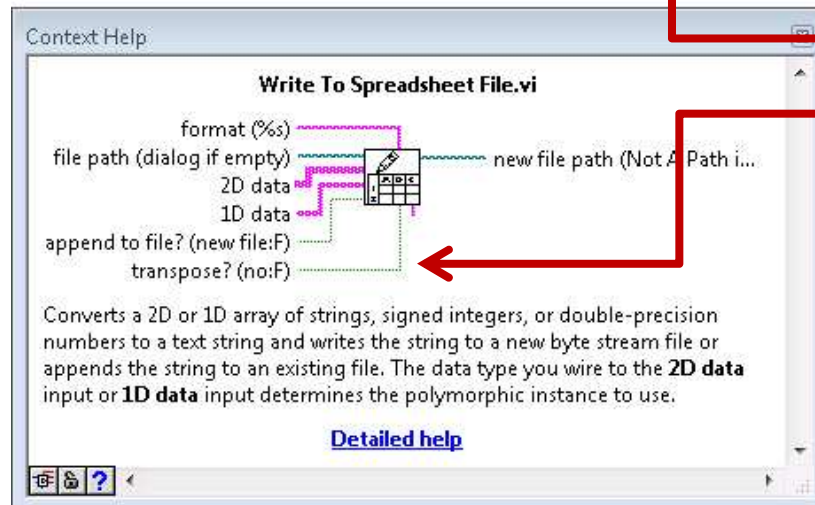
- LabVIEW allows extensive use of file stream operations
- We'll only use Write Spreadsheet function

# File I/O

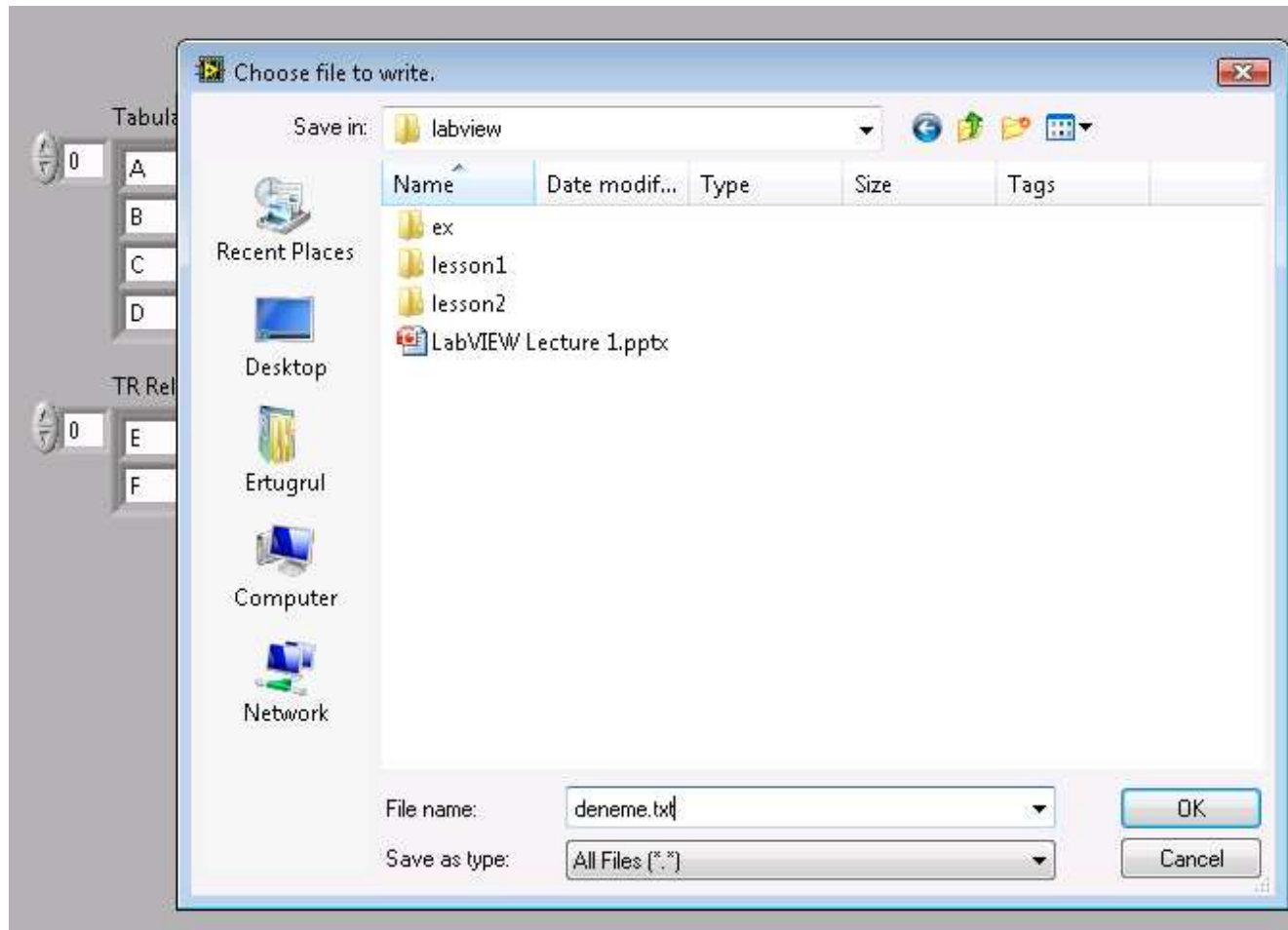


All operations are done in Block Diagram

Note that we've transposed input data

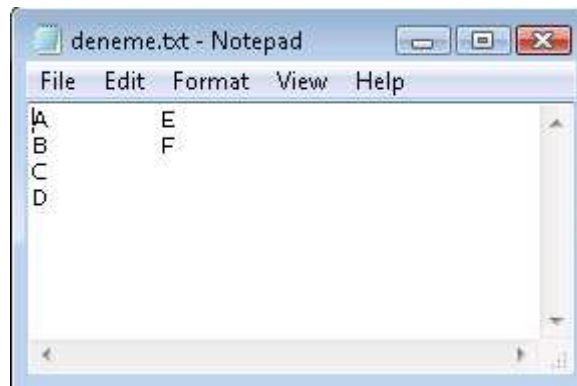


# File I/O



# File I/O

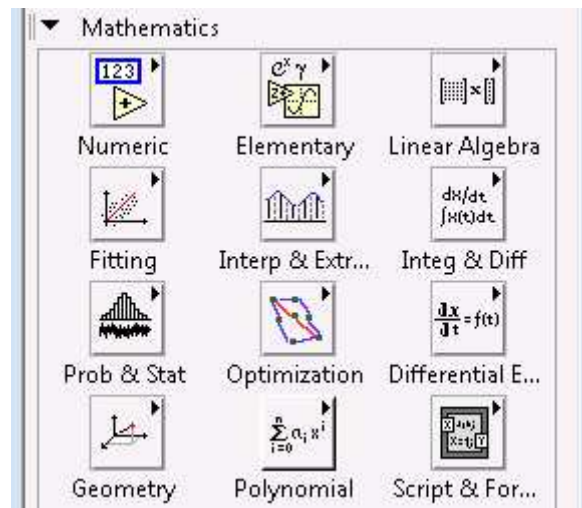
Once saved, we can use this data in other applications like MATLAB or Origin





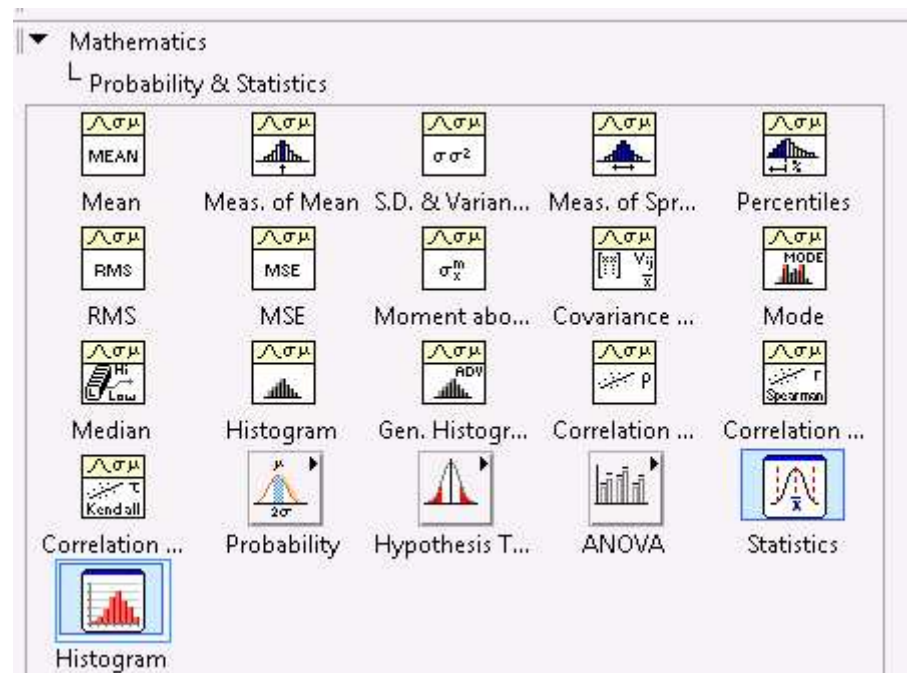
# More

Besides Programming palette, LabVIEW has an extensive selection of Mathematical functions under Mathematics palette.



# More

You may need some functions under Probability & Statistics palette.



# Data Acquisition

- What LabVIEW does best
- There are many ways to communicate with instruments:

– GPIB 

– RS232 / Serial Port 

– LPT / Parallel Port 

– USB 

– TCP/IP 

# GPIB

- Standardized as IEEE-488
- a.k.a. HP-IP; Hewlett Packard/General Purpose Interface Bus
- Has deep similarities to parallel port
- 15 instruments can be connected to one bus



# GPIB

- Every instrument has it, even the very old ones



# Connection

- Every instrument has a different
  - Command set
  - Data I/O pipeline
  - Response time
  - Buffer size
  - ...
- What to do?
  - RTFM

# Connection

- Internet is a good source for instrument manuals, even the old ones
- Generally instrument vendor site is enough
  - unless it stopped manufacturing that device or upgraded with a new model
  - even then, decent vendors have an archive for old documents
- Find a manual or you waste your time by guessing the command set!

# Connection

- Seek for the computer interface chapter



The screenshot shows the product page for the SR810/830 Lock-In Amplifier on the Stanford Research Systems website. The page includes a navigation menu, a search bar, and a list of product features. A red arrow points from the 'Manual' link in the right-hand menu to the 'Table of Contents' section in the adjacent image.

Stanford Research Systems

Products Prices & Ordering Downloads Support Contact Us My Account My Cart

Home > Products > Scientific Instruments > SR810/830

**Lock-In Amplifier**  
SR810 and SR830 — 100 kHz DSP lock-in amplifiers

- 1 mHz to 102.4 kHz frequency range
- >100 dB dynamic reserve
- 5 ppm/C stability
- 0.01 degree phase resolution
- Time constants from 10  $\mu$ s to 30 ks (up to 24 dB/oct rolloff)
- Auto-gain, -phase, -reserve and -offset
- Synthesized reference source

Prices  
Get a Quote  
Buy Online  
Datasheet  
Manual

## MODEL SR810 DSP Lock-In Amplifier

Table of Contents		
1-3	Aux Inputs (A/D Inputs)	4-24
1-5	Aux Outputs (D/A Outputs)	4-24
1-7	X and Y Outputs	4-24
	Signal Monitor Output	4-25
	Trigger Input	4-25
	TTL Sync Output	4-25
2-1	Preamp Connector	4-25
2-2	Using SRS Preamps	4-26
2-5		
2-7		
2-10	<b>PROGRAMMING</b>	
2-11	GPIB Communications	5-1
	RS-232 Communications	5-1
	Status Indicators and Queues	5-1
	Command Syntax	5-1
3-1	Interface Ready and Status	5-2
3-3	GET (Group Execute Trigger)	5-2
3-4		
3-5	<b>DETAILED COMMAND LIST</b>	5-3
3-7	Reference and Phase	5-4

Systems

744-9049  
kSRS.com

Inc.



# Connection

## Remote Programming

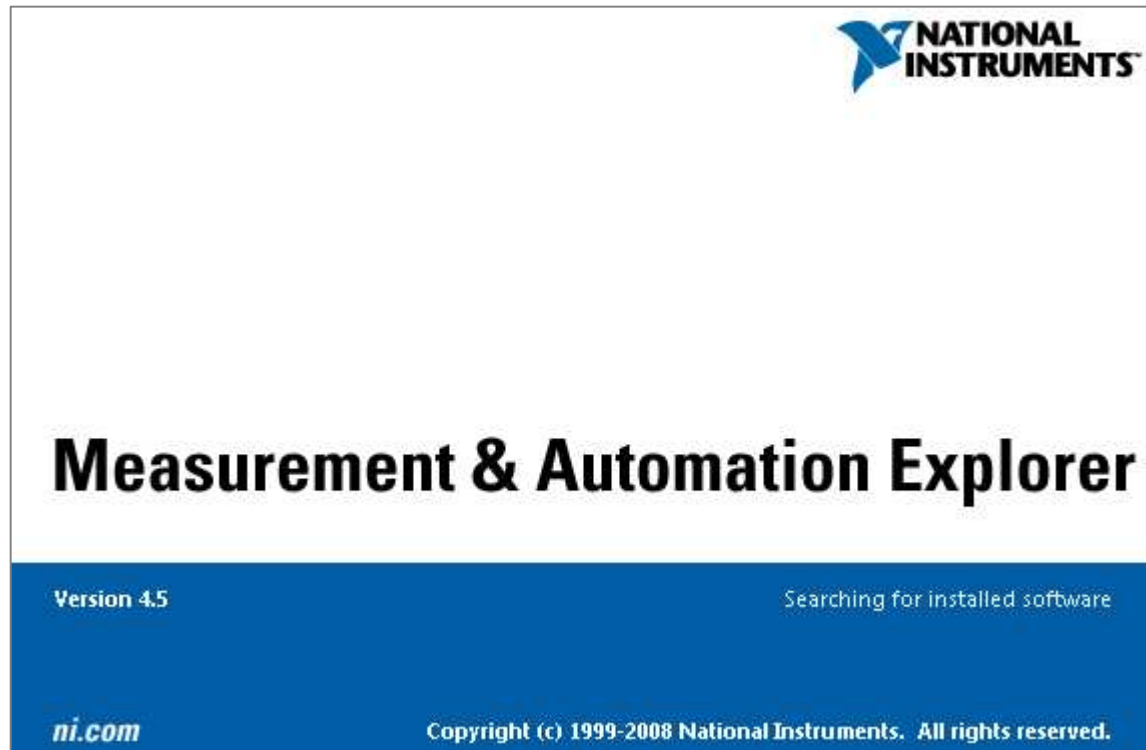
### REFERENCE and PHASE COMMANDS

PHAS (?) {x}	The PHAS command sets or queries the reference phase shift. The parameter x is the phase (real number of degrees). The PHAS x command will set the phase shift to x. The value of x will be rounded to 0.01°. The phase may be programmed from $-360.00 \leq x \leq 729.99$ and will be wrapped around at $\pm 180^\circ$ . For example, the PHAS 541.0 command will set the phase to $-179.00^\circ$ ( $541-360=181=-179$ ). The PHAS? queries the phase shift.
FMOD (?) {i}	The FMOD command sets or queries the reference source. The parameter i selects internal (i=1) or external (i=0).
FREQ (?) {f}	<p>The FREQ command sets or queries the reference frequency. The FREQ? query command will return the reference frequency (in internal or external mode).</p> <p>The FREQ f command sets the frequency of the internal oscillator. This command is allowed only if the reference source is internal. The parameter f is a frequency (real number of Hz). The value of f will be rounded to 5 digits or 0.0001 Hz, whichever is greater. The value of f is limited to <math>0.001 \leq f \leq 102000</math>. If the harmonic number is greater than 1, then the frequency is limited to <math>nx \leq 102 \text{ kHz}</math> where n is the harmonic number.</p>
RSLP (?) {i}	The RSLP command sets or queries the reference trigger when using the external reference mode. The parameter i selects sine zero crossing (i=0), TTL rising edge (i=1), or TTL falling edge (i=2). At frequencies below 1 Hz, the TTL reference must be used.
HARM (?) {i}	The HARM command sets or queries the detection harmonic. This parameter is an integer from 1 to 19999. The HARM i command will set the lock-in to detect at the ith harmonic of the reference frequency. The value of i is limited by $ix \leq 102 \text{ kHz}$ . If the value of i requires a detection frequency greater than 102 kHz, then the harmonic number will be set to the largest value of i such that $ix \leq 102 \text{ kHz}$ .
SLVL (?) {x}	The SLVL command sets or queries the amplitude of the sine output. The parameter x is a voltage (real number of Volts). The value of x will be rounded to 0.002V. The value of x is limited to $0.004 \leq x \leq 5.000$ .

- Of course understanding the command set has a prerequisite of understanding the instrument itself.
- Learn the instrument
- Learn what to do, without the computer
- Apply the scheme to LabVIEW

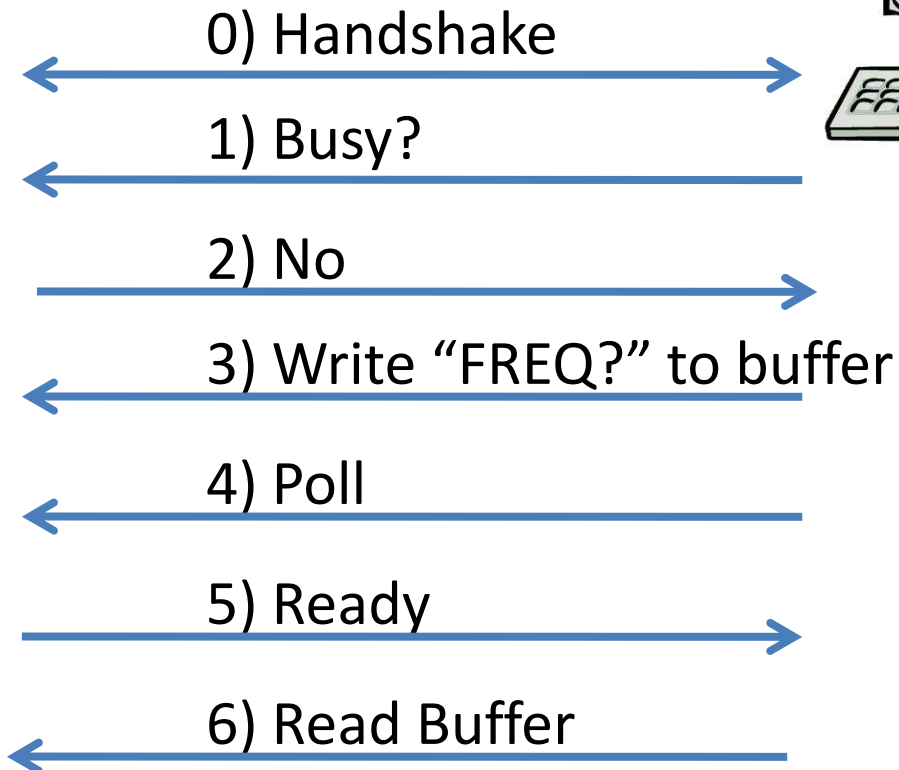
# Connection

- You can check the status of the instrument by employing Measurement and Automation Explorer



# Instrument I/O

- There is a general scheme of instrument interfacing:



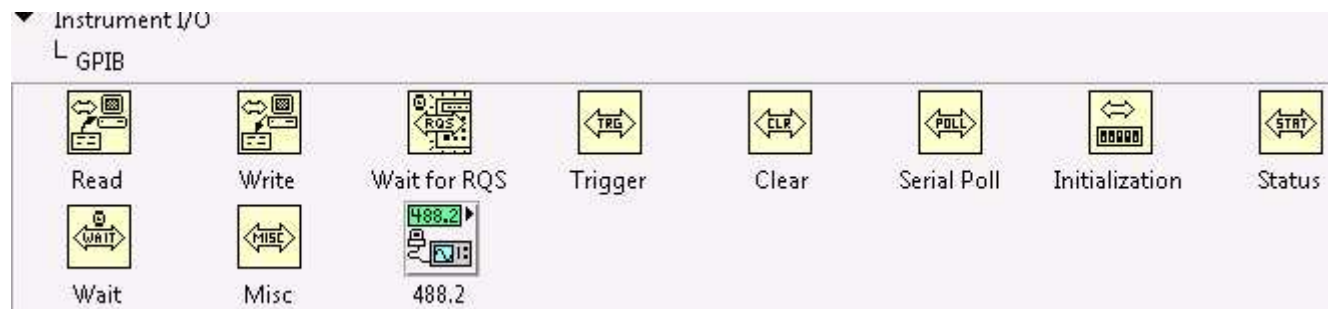
# Instrument I/O

- Seek Instrument I/O palette
- Choose an appropriate way from many



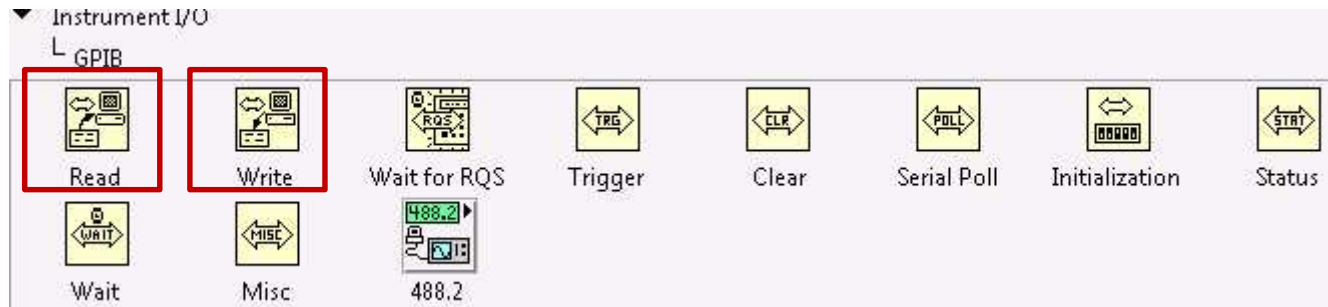
# Instrument I/O

- All actions that are needed can be found under that palette.
- You can do more advance stuff by invoking the advanced palette (488.2 in this case)



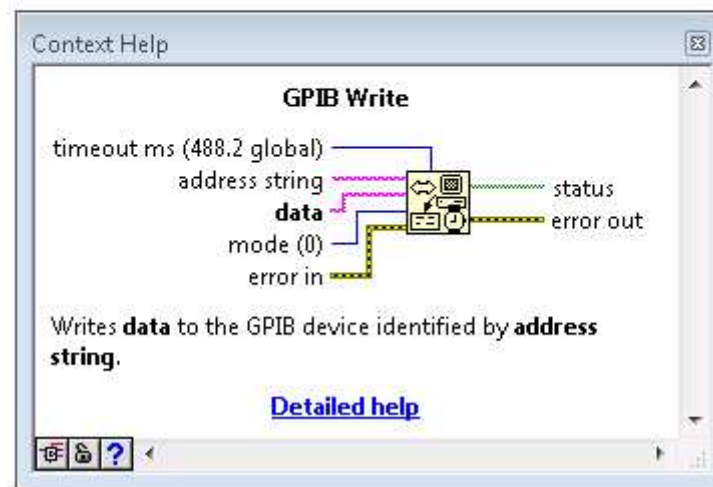
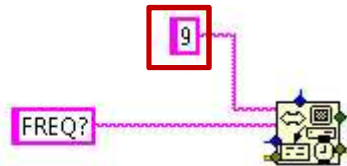
# Instrument I/O

- For modern instruments read and write actions are enough



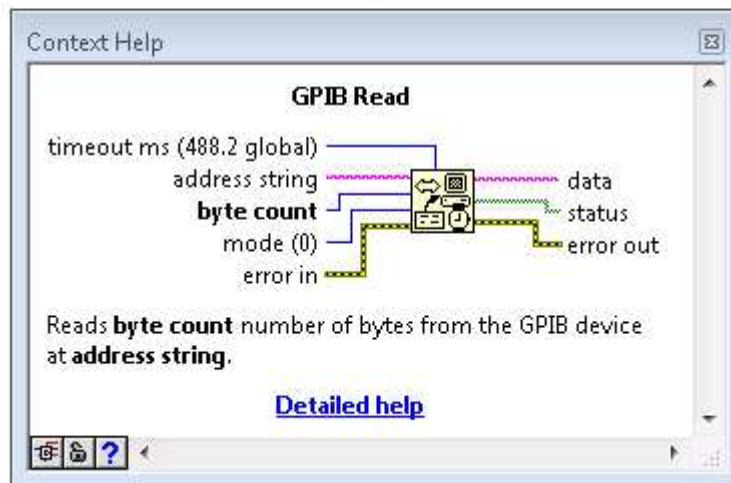
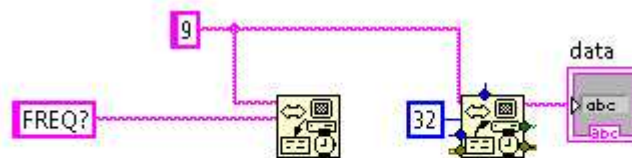
# Instrument I/O

- To send a command to the instrument use Write action
- You need the GPIB address of the instrument



# Instrument I/O

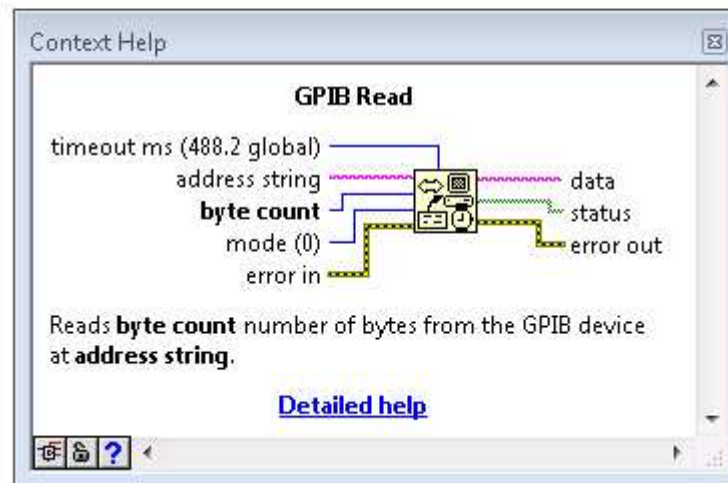
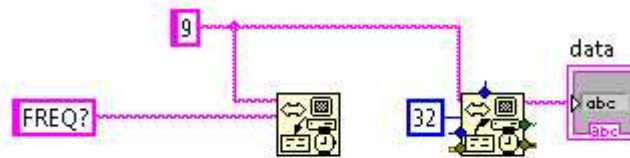
- To get the result of your command use Read action





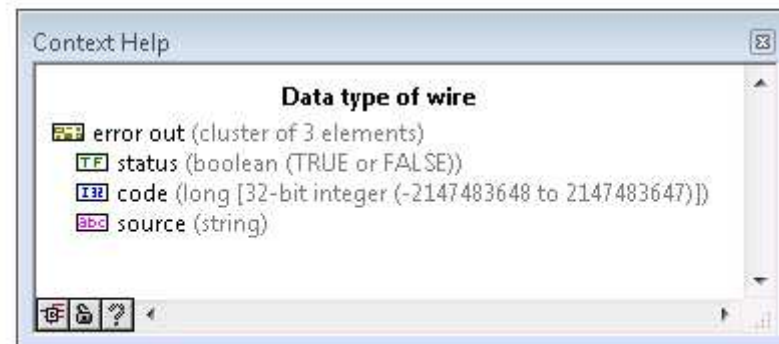
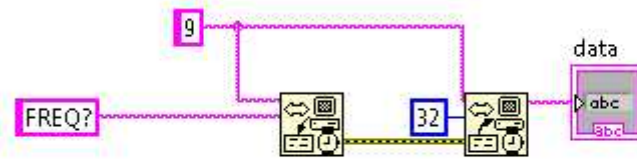
# Instrument I/O

- There is an error in this code. Can you guess?



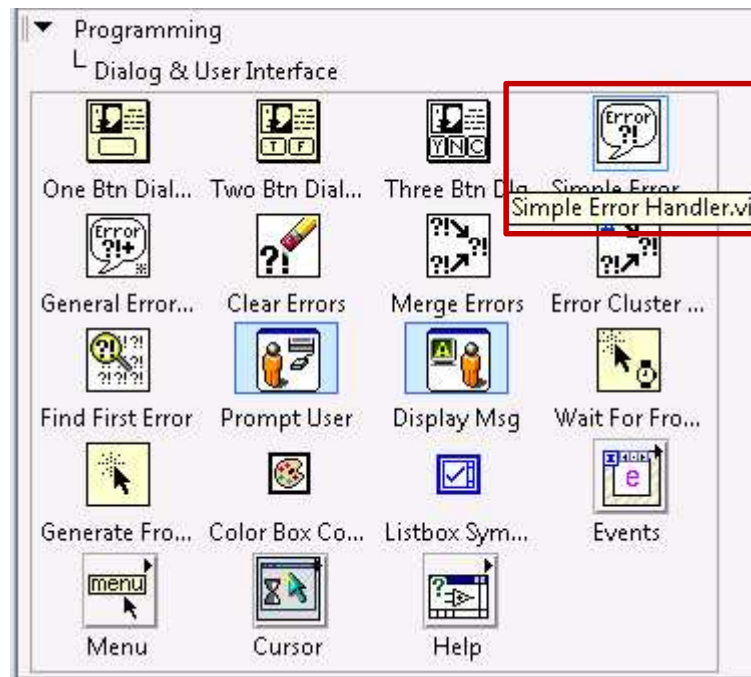
# Instrument I/O

- Much better

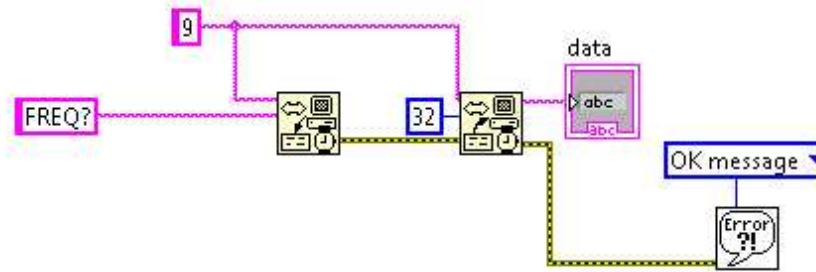


# Error Handling

- Simple error handling can be easily done by employing Simple Error Handler



# Error Handling



Context Help

### Simple Error Handler.vi

The diagram shows the 'Simple Error Handler.vi' block with the following labels and connections:

- type of dialog (OK msg:1) - connected to the top-left input of the block
- error in (no error) - connected to the bottom-left input of the block
- error? - connected to the top-right output of the block
- code out - connected to the middle-right output of the block
- source out - connected to the bottom-right output of the block
- error out - connected to the bottom-right output of the block
- message - connected to the bottom-right output of the block

Indicates whether an error occurred. If an error occurred, this VI returns a description of the error and optionally displays a dialog box.

[Detailed help](#)

Navigation icons: Home, Back, Forward, Search, Help, Close

# Error Handling

- You may not want to be bothered by some timeout error

