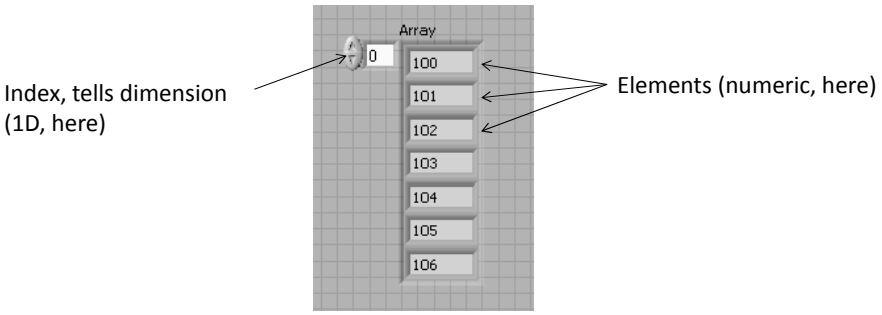
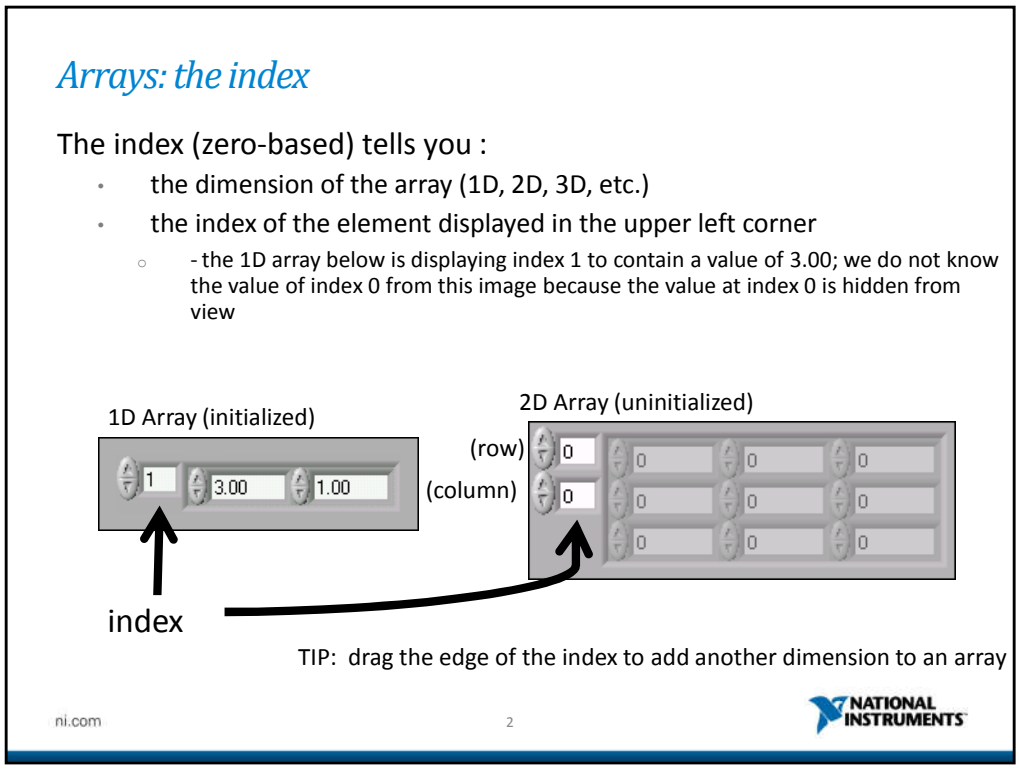
Arrays

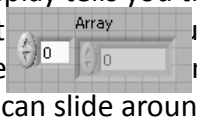
- An array consists of elements and dimensions
  - Elements: data that make up the array
  - Dimension: the length, height, or depth of an array
    - $(2^{31})–1$ elements per dimension, memory permitting
    - 1 or more dimensions

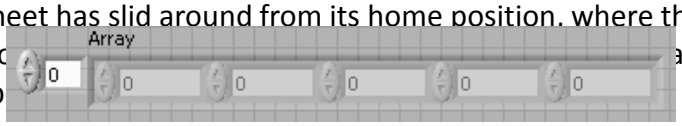


- **Array:** Arrays group data elements of the same type. An array consists of elements and dimensions. El



To add dimensions to an array one at a time, right-click the index display and select **Add Dimension** from the shortcut menu. You also can use the Positioning tool to resize the index display until you have as many dimensions as you want.

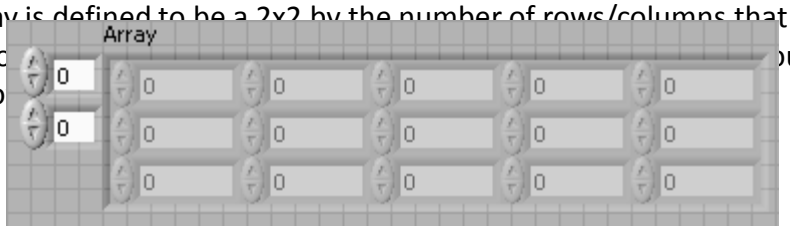
The index display tells you the index (location) of the element that is currently displayed in t 

upper right-hand corner of the array. Imagine the border of the ndow into a giant speadsheet of numbers, where the spreadsheet can slide around behind the window. The index will tell you how much the spread sheet has slid around from its home position, where the window shows the top left c 

array is hiding index 0 and p

The index is zero-based, meaning it begins it counting with zero.

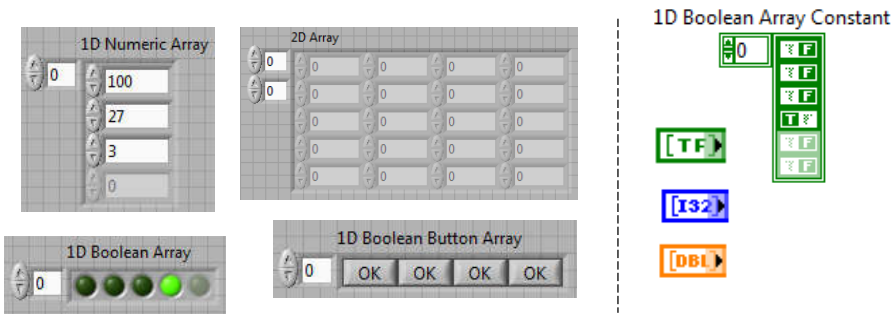
Uninitialized a 

gray background and grayed-out numbers. If you have a value a array (which would be a 2x2 array), then column 3 will be uninitialized and all columns less than 3 (0,1,2) must be initialized . Keep in mind an array is defined to be a 2x2 by the number of rows/columns that are initialized, nc 

ough the "array windo

Arrays: the index

Arrays can be of many different data types, but only one data type at a time



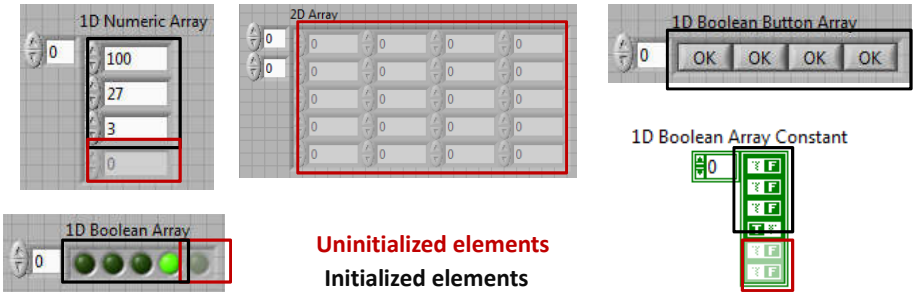
ni.com 3 NATIONAL INSTRUMENTS

All the elements in an array must have the same data type, because arrays, by definition, are a way to gro

Arrays: Initialized and Uninitialized

If the elements of an array are grayed out, then those elements are uninitialized.

Uninitialized elements do not have a value and are place holders



ni.com 4 NATIONAL INSTRUMENTS

The uninitialized elements are outlined in red; the initialized e

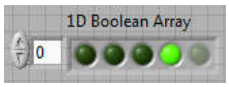
Initialized elements have a value. Uninitialized elements are

When determining the size of an array, only the initialized ele

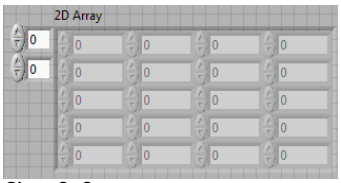
Arrays: Initialized and Uninitialized

Size is the number of initialized elements in each dimension

Dimension is how the elements are organized



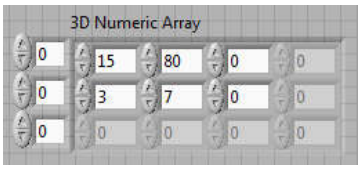
Size: 4
Dimension: 1D



Size: 0x0
Dimension: 2D

1D Boolean Array Constant

Size: 4
Dimension: 1D



Size: 2x3x?
Dimension: 3D

Size: at least 4 – more elements could be hidden
Dimension: 1D

ni.com 5 NATIONAL INSTRUMENTS

To determine the size of a 1D array, count the number of initi

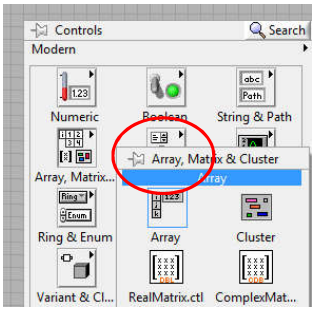
To determine the size of a 2D array, count the number of initi

5

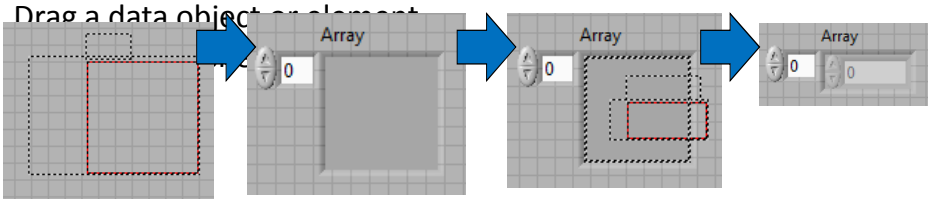
Creating an Array

Place an array shell on the front panel

- From the Controls»Modern»Array, Matrix, and Cluster subpalette, select the Array icon.



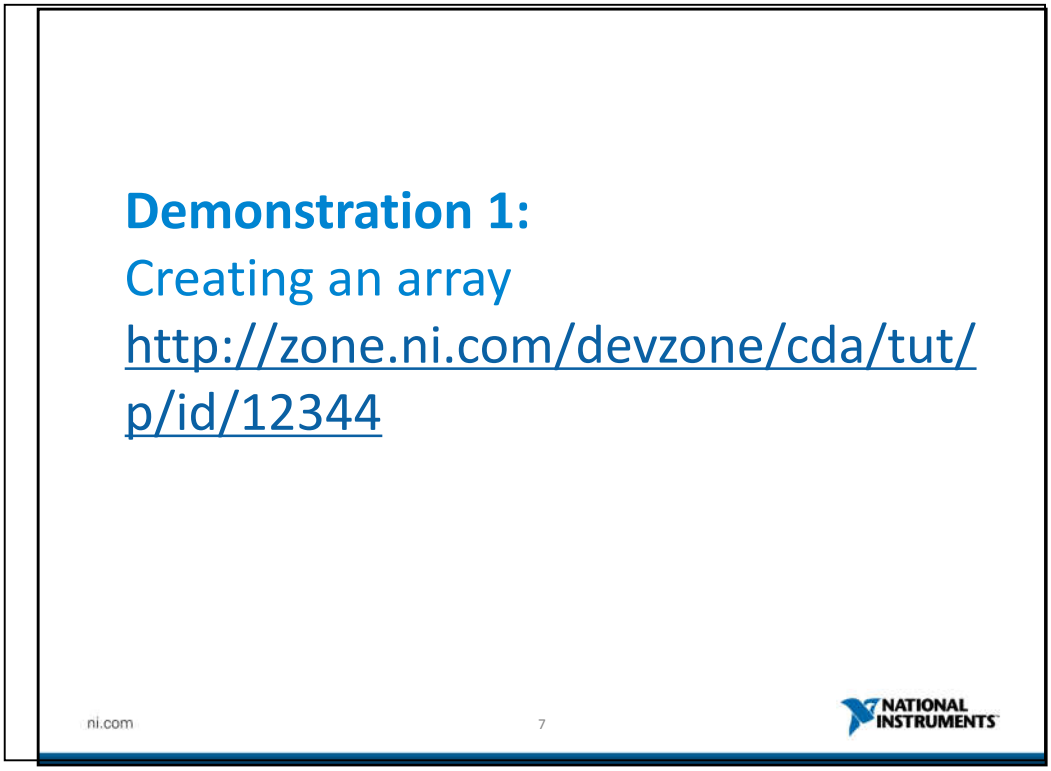
Drag a data object or element



ni.com 6 NATIONAL INSTRUMENTS

To create an array control or indicator as shown, select an array on the **Controls»Modern»Array, Matrix, and Cluster** palette, place it on the front panel, and drag a control or indicator into the array shell. If you attempt to drag an invalid control or indicator such as an XY graph into the array shell, you are unable to drop the control or indicator in the array shell. Once a valid item is placed in the array shell, the array shell shrinks to fit around the control or indicator. You can then drag the edge of the array to display more elements.

Array shells are available on the front panel and block diagram, but you must insert an object in the array shell before you use the array on the block diagram. Otherwise, the array terminal appears black with an empty bracket.



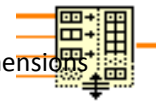
http://zone.ni.com/devzone/cda/tut/p/id/12344

Array Functions

Array Size:

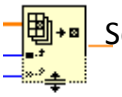
- Returns the number of elements in each dimension of the array

Build Array:



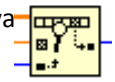
- Adds elements or dimensions to an array

Index Array:



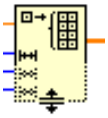
- Returns the value at the index you specify

Search 1D Array:



- Returns the index of the va you specify; if not found, returns -1

Initialize Array:



- Creates an initialized array of the dimension and data type you specify

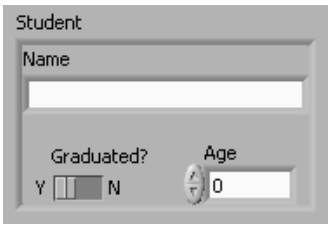
NATIONAL INSTRUMENTS

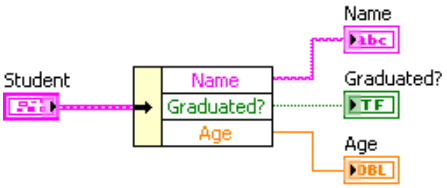
There are many more array functions; however, these are the ones that are often used.

Clusters

• Clusters group data elements of mixed types

• Similar to a record or a struct in text-based programming languages





ni.com 9 NATIONAL INSTRUMENTS

Cluster: Clusters group data elements of mixed types. Often the objects have a relationship with one ano

Cluster are created in a similar fashion to arrays: place a shell, then add object into the shell.

Cluster Functions

- **Bundle** (there is also **Unbundle**)
  - Assembles a cluster from individual elements.
- **Bundle By Name** (there is also **Unbundle By Name**)
  - Replaces one or more cluster elements. This function refers to cluster elements by name instead of by their position in the cluster.
- **Cluster Constant**
  - Use this constant to supply a constant cluster value to the block diagram.

Bundle By Name **Bundle** **Cluster Constant**

ni.com 10 NATIONAL INSTRUMENTS

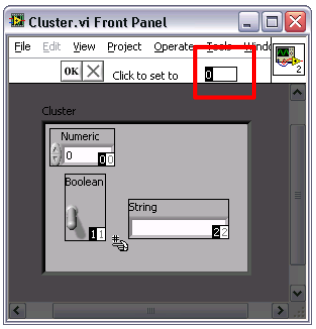
The terms bundle and cluster are closely related in LabVIEW.

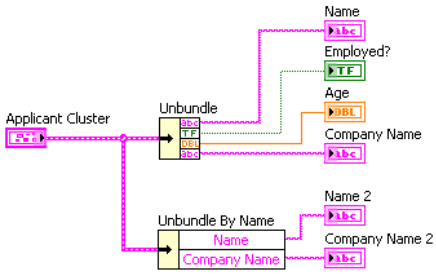
Example: You use a bundle function to create a cluster. You use an unbundle function to extract the parts of a cluster.
Bundle - Forms a cluster containing the given objects in the specified order.
Bundle by Name - Updates input cluster's element values (the object must have an owned label). *requires input cluster*
Unbundle - Splits a cluster into each of its individual elements by data type.
Unbundle by Name - Returns the cluster elements whose names you specify.

Two major difference between Unbundle/Bundle and Unbundle By Name/Bundle By Name: (1) "By Name" = label of elements visible (not just the data type. (2) Bundle By Name requires an input cluster and updates, instead of creating a new cluster.

Clusters: Order

- Cluster elements have a logical order unrelated to their position in the shell
- You can view and modify the cluster order by right-clicking the cluster border and selecting **Reorder Controls In Cluster** from the shortcut menu
- Order determines how cluster is unbundled





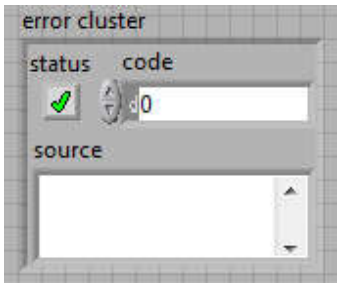
The order in clusters is important to…
-the user if they want to tab through the objects (ie, input valu
-the programmer because the order is what defines the order

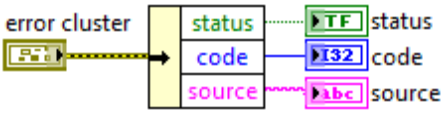
The initial order of a cluster is determined by the order in whi

When reordering, the number displayed in the menu bar (out

Error Cluster

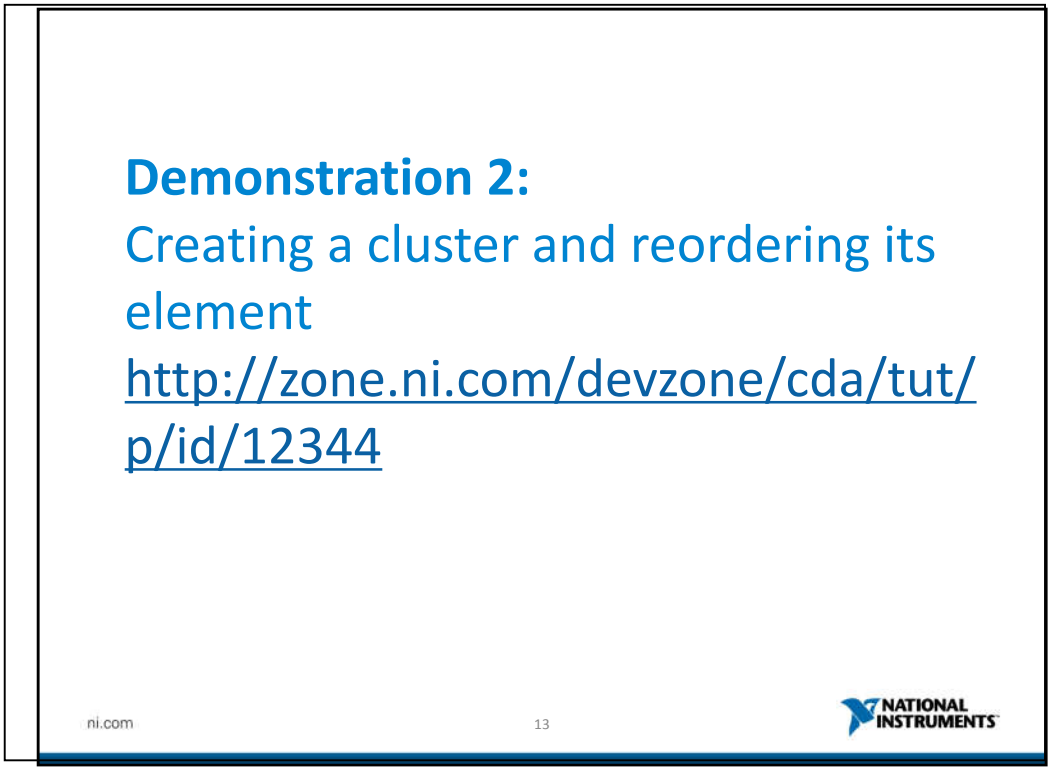
- Three parts:
  - Status – Boolean, TRUE when an error occurred
  - Code - numeric (I32), identifies which error occurred
  - Source – string, identifies where the error occurred





Note: Warnings have a code and source, but the status is FALSE

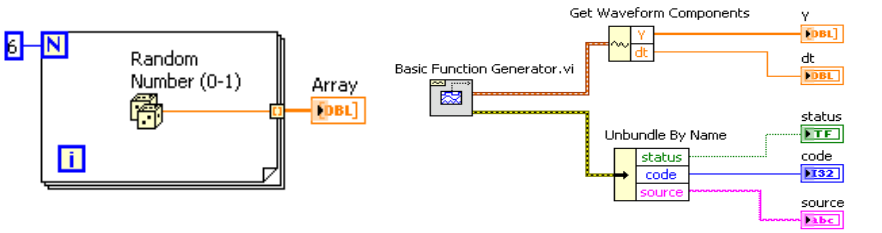
NATIONAL INSTRUMENTS



http://zone.ni.com/devzone/cda/tut/p/id/12344

Clusters vs. Arrays

- Clusters differ from arrays in that they are a fixed size
- Clusters can contain mixed data types; arrays contain only one data type
- Like an array, a cluster is either a control or an indicator and cannot contain a mixture of controls and indicators
- You can create a cluster of clusters, cluster of arrays, array of clusters, but NOT an array of arrays

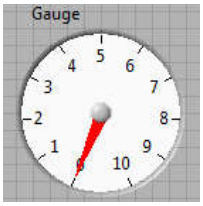


Cluster are a fixed size because you cannot add an object to an existing cluster – only change the values within the object. You can create a new cluster which contains an existing cluster (cluster within a cluster), but this is still a *new* cluster.

Great article about Arrays and clusters: http://zone.ni.com/devzone/cda/tut/p/id/7571

Customizing Controls

LabVIEW ships with multiple palettes of controls available to you. However, you may want to customize them or create your own



Standard LabVIEW Gauge

Customized LabVIEW Gauge

ni.com 15 NATIONAL INSTRUMENTS

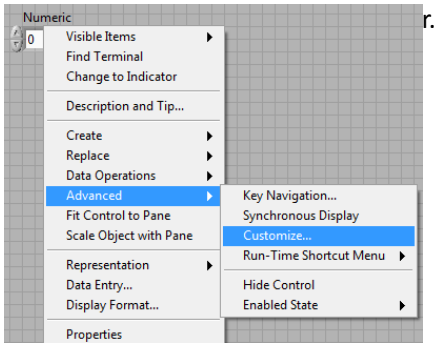
Depending on you application, you may want customize a control. For example, if your VI will be running on a touch screen, you might want to make the increment/decrement buttons on a numeric control much larger. If you creating something that will be presented or sold, you may want to use a custom color scheme and make the control a bit “flashier” like the gauge in this slide customized with an NI theme.

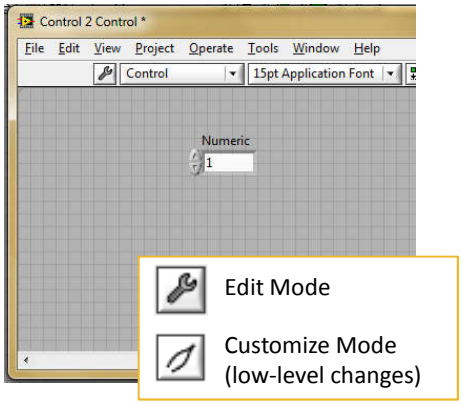
Not only do you have the ability to customize controls cosmetically, but you can save them for use again later. Furthermore, there is a way to link each instance of a custom control to the saved file so if you want to change something down the road, you don’t have to edit each and every one.

Creating Custom Controls

How to create:

- Right click on the control; select **Advanced >> Customize**
- r.





ni.com 16 NATIONAL INSTRUMENTS

In order to create a custom control or type def, you must right-click on an object on the FRONT PANEL. Then go to **Advanced >> Customize…** to pull up the Control Editor window. In the Control Editor Window you can add/remove things from a cluster, edit the items listed in an enum, change the color of an LED, move increment/decrement arrows to the top/bottom of digital display instead of to the side, and much more!

The tweezer button next to the drop-down takes you to low level editing where you can pull apart the layers that create the object (ie, shadow, foreground, background, etc)

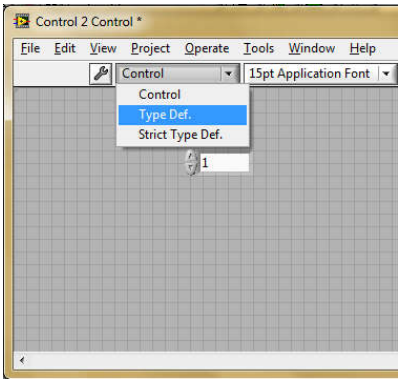
Saving Custom Controls

Three options:

- Custom Control
- Type Definition
- Strict Type Definition

All 3 options create a *.ctl file.

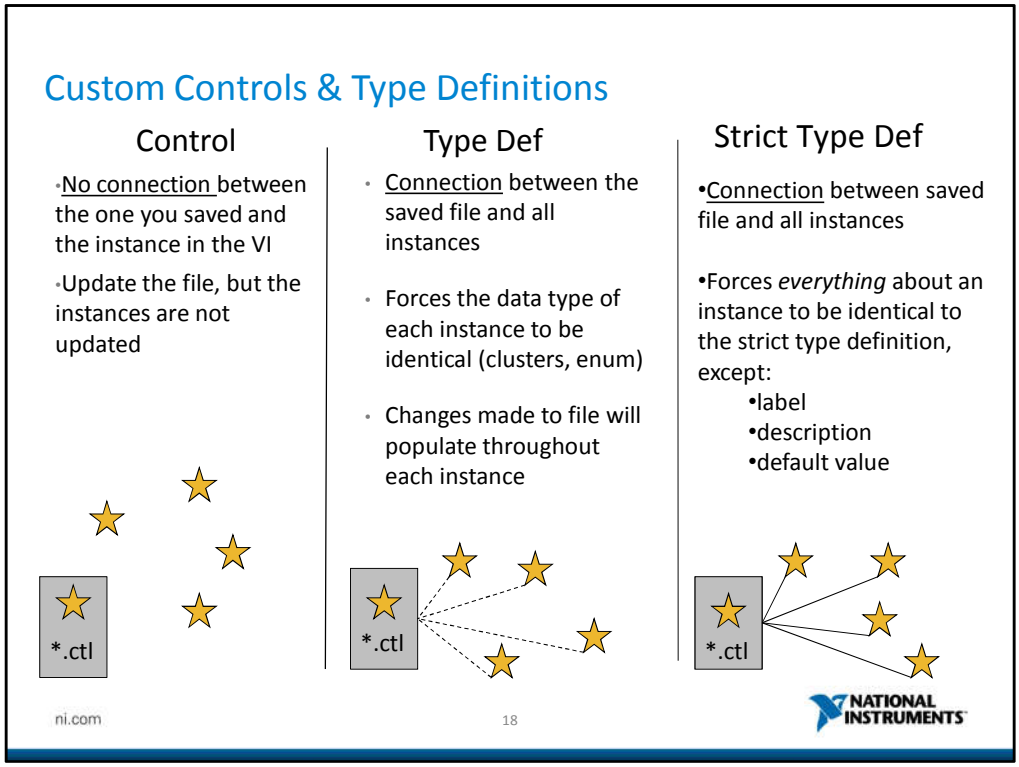
Once you save the custom control it takes on the name of the option you chose.



When you are ready to save your customization, you have three options, which are displayed in the drop down box on the toolbar:
--Custom Control
--Type Definition
--Strict Type Definition

All 3 options create a *.ctl file, but behave differently. The next slide explains the difference between these options, as they are very useful when used properly.

Depending on the option you choose, the file created will typically be referred to as that option. For example, if you save as a type definition, it is common to refer to that file as a type definition (or type def, for short).



A custom control, type def or strict type def all create a *.ctl file, which stores the customizations you make to a front panel control. The thing that is special about type defs and strict type defs is that every instance of that control you use on your block diagram is linked to the *.ctl and will update if you make changes to the file.

The purpose of a *.ctl file saved as a Custom Control is to prevent you from going through the potentially lengthy process of creating it again- the file provides easy access to create another.

The purpose of a type def is to make sure that the data type is consistent (including the item list for an enum- which is particularly helpful when building state machines) for each instance.

Strict typ defs are just that, strict. Everything must be the same (most noticeable is cosmetic changes –size of LED, color, etc). To be able to tell them apart, the label should be different (labels should always be unique and descriptive). Descriptions can be different because they might serve slightly different purposes (LED indicating different warnings in a system – you would want to be able to describe which LED represents which process more than just using the label). Last, the default value can be different. Everything else is the same.