E. Clusters

Reasons To Use Clusters

Cluster vs. Arrays

Creating a Cluster Control and Constant

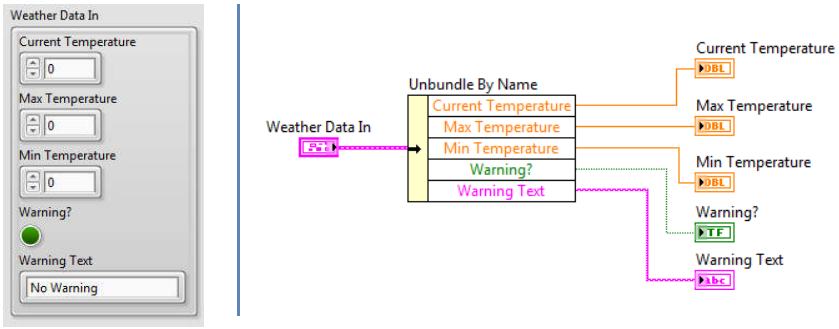
Ordering Items and Resizing Clusters

Disassembling and Modifying Clusters

Plotting Data

Clusters

- Clusters group data elements of mixed types.
- Clusters are similar to a record or a `struct` in text-based programming languages.



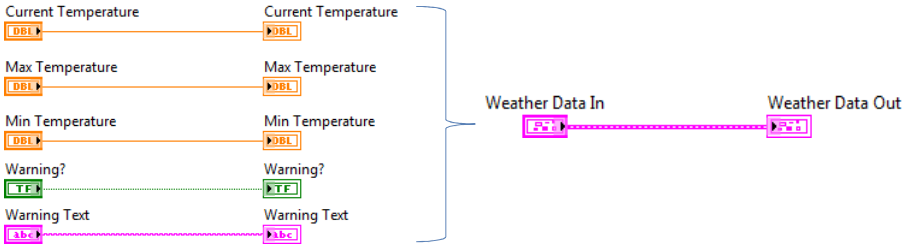
NATIONAL INSTRUMENTS | ni.com/training

Additional items to mention:

- Bundling several data elements into clusters eliminates wire clutter on the block diagram and reduces inputs and outputs for subVIs.
- Most clusters on the block diagram have a pink wire and data type terminal.
- Clusters of numeric values, sometimes referred to as points, have a brown wire and data type terminal.
- You can wire brown numeric clusters to Numeric functions to perform the same operation simultaneously on all elements of the cluster.

Why Use Clusters?

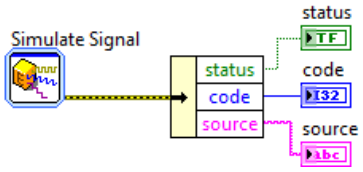
- Keep data organized.
  - Logically group related data values together.
  - Improve diagram readability by eliminating wire clutter.
- Reduce the number of connector pane terminals.

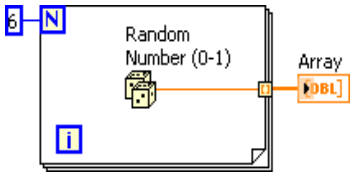


Clusters vs. Arrays

- Clusters are a fixed size.
- Clusters can contain mixed data types.
- Clusters can be a control, an indicator, or a constant.
  - All elements have to be controls, indicators, or constants.

- Arrays vary in size.
- Arrays contain only one data type.
- Arrays can be a control, an indicator, or a constant.



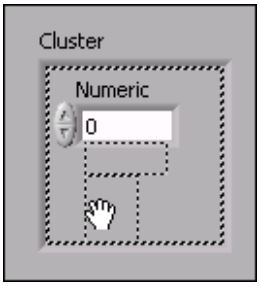


NATIONAL INSTRUMENTS | ni.com/training

Create a Cluster Control

For a new cluster:

1. On the front panel, select Cluster from the **Controls** palette.
2. Place a data object into the cluster shell.
3. Place additional data objects, if necessary, into the shell.



From block diagram terminal or wire:

1. Right-click and select **Create»Control** or **Create»Indicator**.

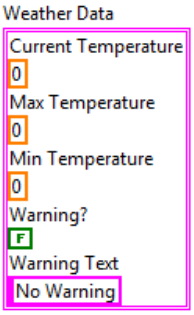
Create a Cluster Constant

For a new cluster:

1. On the block diagram, select Cluster Constant from the **Functions** palette.
2. Place a constant into the cluster shell.
3. Place additional data objects, if necessary, into the cluster shell.

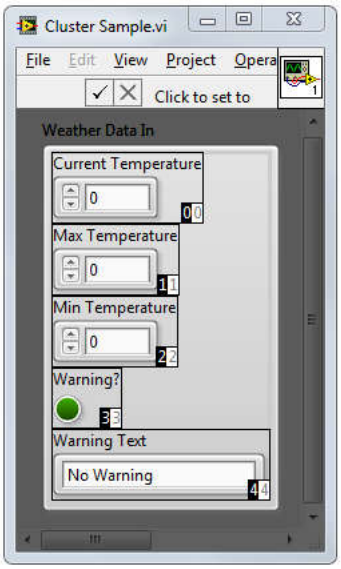
From block diagram terminal or wire:

1. Right-click and select **Create»Constant**.



Cluster 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**.



Notes:

The first object you place in the cluster is element 0, the second is element 1, and so on.

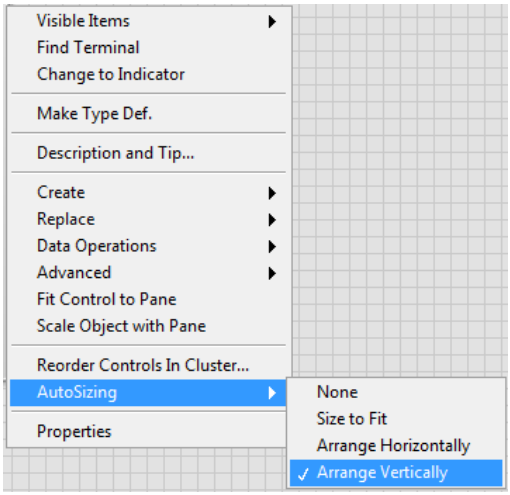
If you delete an element, the order adjusts automatically.

The cluster order determines the order in which the elements appear as terminals on the Bundle and Unbundle functions on the block diagram.

Use **AutoSizing»Arrange Horizontally** or **AutoSizing»Arrange Vertically** from the shortcut menu to arrange the elements in the cluster horizontally or vertically in order.

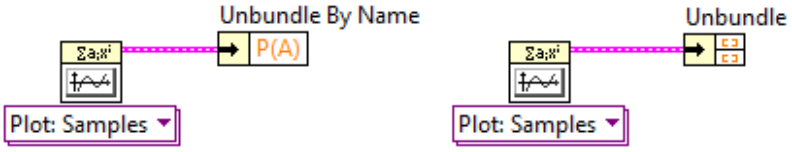
Autosizing Clusters

- Autosizing helps you arrange elements in clusters.
- NI recommends the following:
  - Arrange cluster elements vertically.
  - Arrange elements compactly.
  - Arranges elements in their preferred order.



Disassembling a Cluster

- Use the Unbundle By Name function whenever possible.
- Use Unbundle function when some or all cluster elements are unnamed.



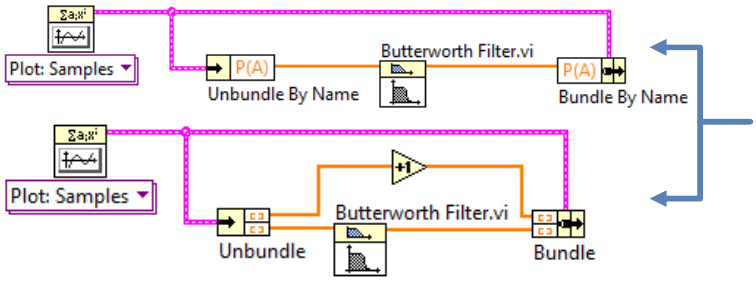
8

NATIONAL INSTRUMENTS | ni.com/training

In this slide, the Polynomial Plot.vi outputs an XY Graph as a cluster of two arrays (X and Y). The Y array is labeled as **P(A)** so you can access it using the Unbundle By Name function. However, the X array is unlabeled so you cannot access it using the Unbundle By Name function. To access data in the X array, use the Unbundle function.

Modifying a Cluster

- Use Bundle By Name whenever possible to access elements in a cluster.
- Use Bundle when some or all cluster elements are unnamed.



You must wire the **input cluster** terminal.

NATIONAL INSTRUMENTS | ni.com/training

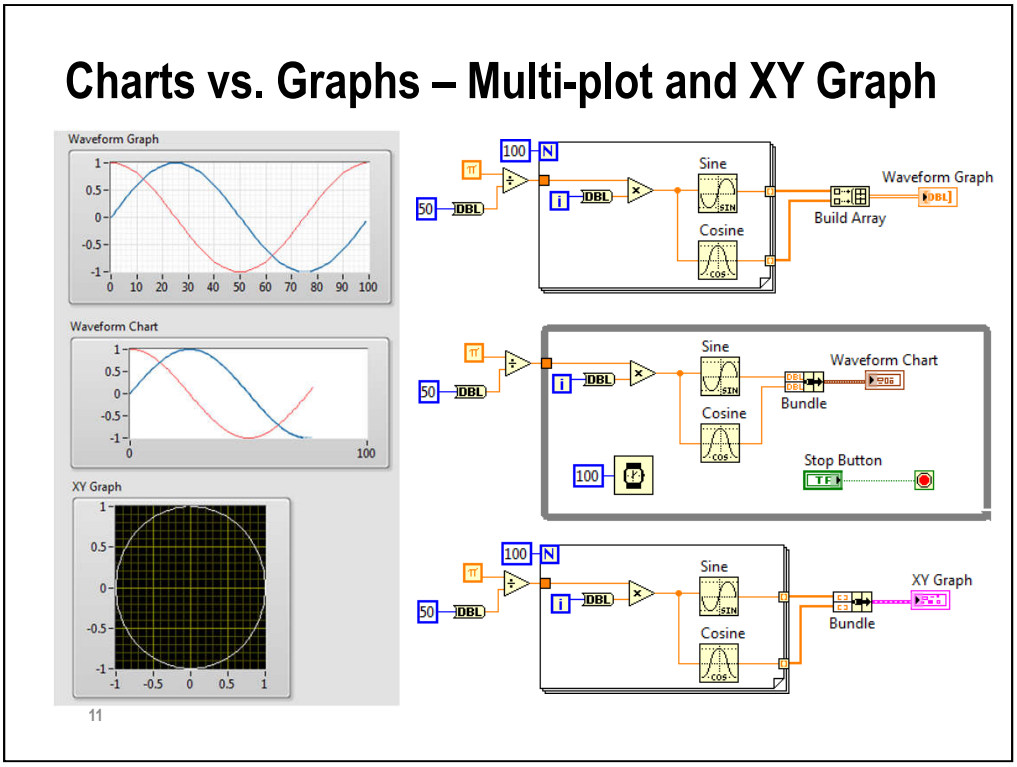
In this slide, both examples modify the **Y** array, (P(A), of the Polynomial Plot.vi with a Butterworth filter. In the bottom example, you also increment the X array of the Polynomial Plot.vi output by one.

Creating a Cluster on the Diagram

- Use the Bundle function to programmatically create a cluster on a block diagram.
- If the elements that are bundled have labels, you can access them using the Unbundle By Name function. Otherwise use the Unbundle function.

NATIONAL INSTRUMENTS | ni.com/training

This technique is typically used to create multi-plot charts which is discussed in more detail later.

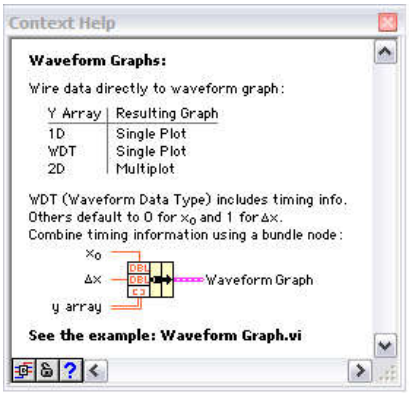


The Bundle function is often used to create multi-chart plot charts and XY plots.

The Build Array function is used to create multi-plot waveform graphs.

Plotting Data

Use the **Context Help** window to determine how to wire multi-plot data to Waveform Graphs, Charts and XY Graphs.



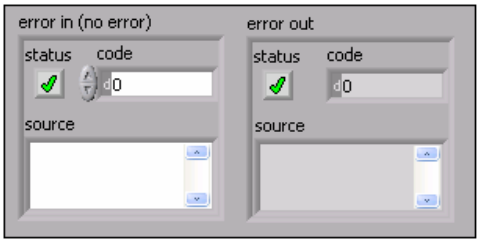
NATIONAL INSTRUMENTS | ni.com/training

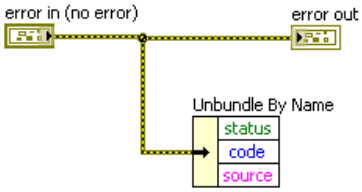
Use the Context Help and shipping examples to investigate all the possible ways to assemble data for various graphical displays.

LabVIEW examples and help provide extensive information on wiring data to the various graphical displays.

Error Clusters

- LabVIEW uses error clusters to pass error information.
- An error cluster contains the following elements:
  - **status**—Boolean value that reports True if an error occurs.
  - **code**—32-bit signed integer that identifies the error.
  - **source**—String that identifies where the error occurred.





F. Type Definitions

Type Definitions and Custom Data Types

Creating and Identifying Type Definitions

Controls

Strict Type Definitions

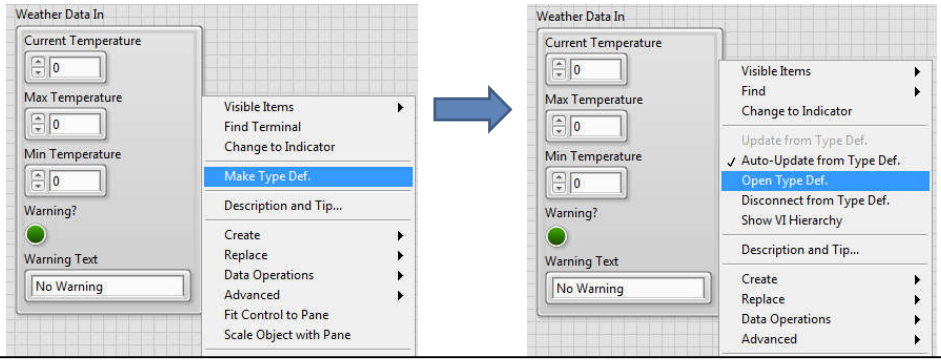
Type Definitions (Type Def)

- A type definition is a master copy of a custom data type (control, indicator, or constant).
  - A custom data type is saved in a .ctl file.
  - Instances of a type def are linked to the .ctl file.
- Instances can be controls, indicators, or constants.
- When the type def changes, instances automatically update.
  - Changes include data type changes, elements added, elements removed, and adding items to an enum.

- You can use type definitions to define custom arrays and clusters.
- A type definition is a master copy of a custom data type (custom control, indicator, or constant) that multiple VIs can use.
- When you use type definitions, you can update of all instances or copies of the custom data type in a single edit.

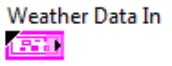
Creating Type Definitions (Type Def)

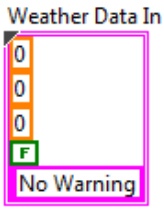
1. Right-click a control, indicator or constant and select **Make Typedef**.
1. Right-click the object again and select **Open Type Def**.
2. Edit control, if needed.
3. Save control as a .ctl file.

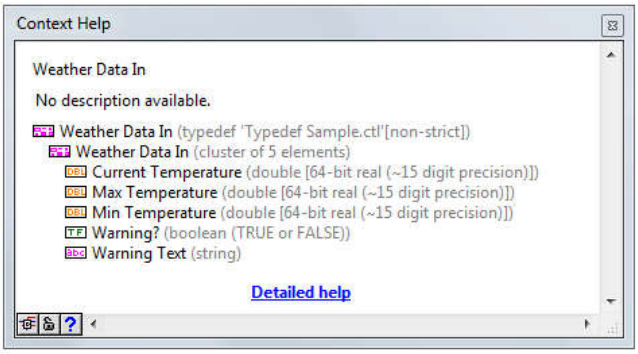


Identifying Type Definitions (Type Def)

- Look for a glyph marking the upper left corner of terminals and constants.
- Hover cursor over glyph to view tip strip.
- View Context Help while hovering cursor over terminal or constant.



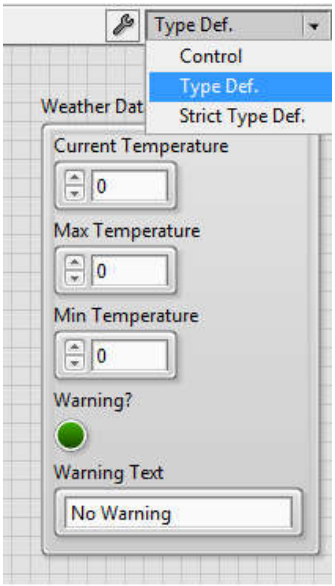




Other Control Options

You can save a custom control as:

- Control
- Type Definition
- Strict Type Definition



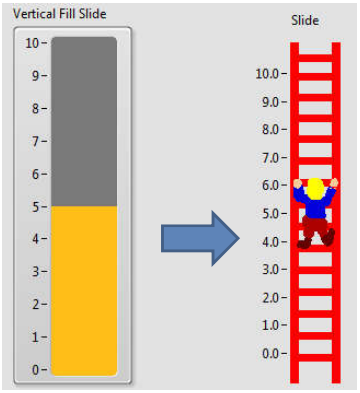
NATIONAL INSTRUMENTS | ni.com/training

LabVIEW has three kinds of custom controls (.ctl files):

- A regular custom control
  Used to create controls that behave like existing controls but look different.
- A type def or type definition control
  Used for changing all instances of a linked control in a single edit.
- A strict type def control
  Used for changing instances in a single edit.
  Used to ensure all instances have same appearance.

Control

- Instances are not linked to a.ctl file.
- Each instance is an independent copy of the control.
- Used to create controls that behave like existing controls but look different.



19

Refer to *LabVIEW* Help for information on how to use the Control Editor to customize controls.

Typically, if you create a custom control you want it linked to the .ctl file as a type definition.

Strict Type Definition

- Strict type definitions are similar to a type definition in that:
  - All instances link to .ctl file.
  - When attributes or data types change, all instances update.
    - Examples: Changing a knob to a dial, a round LED to a square LED, or a double to an integer.
- Strict type definitions enforce every aspect of a instance except label, description, and default value.
- Use strict type definitions to ensure all front panel instances have the same appearance.