APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

Smart Widgets: DropDown Menu

DOCUMENT DATE:	9th MAY 2020
DOCUMENT REVISION:	1.0

Description

This application note shows how to create a dropdown menu that overlaps a Gauge for Picaso, Diablo16 and Pixxi touch screen display modules.

Before getting started, the following are required:

Hardware

- Any 4D Systems display module powered by any of the following processors:
  - Diablo16
  - Picaso
  - Pixxi28/44
- Programming Adaptor for target display module
- uSD Card
- USB Card Reader

Software

- Workshop4
- This requires the **PRO** version of Workshop4

Note: *Using a non-4D programming interface could damage the processor and void the warranty.*

This application note is applicable to all touch screen 4D displays. However, Smart Gauges can also be used on non-touch displays.

Content

Application Overview

The Smart Widgets Editor tool enables PRO version users to easily create custom widgets of their own design. It allows the user to create Sliders, Knobs and Gauges.

The purpose of this application note is to introduce the PRO version exclusive tool and to discuss how to create a tank using a Smart Gauge widget. This application note uses the ViSi-Genie environment.

Setup Procedure

For instructions on how to launch Workshop4, how to open a **ViSi-Genie** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note

- ViSi-Genie Getting Started - First Project for Diablo16 Display Modules
- ViSi-Genie Getting Started - First Project for Picaso Displays
- ViSi-Genie Getting Started - First Project for Pixxi Display Modules

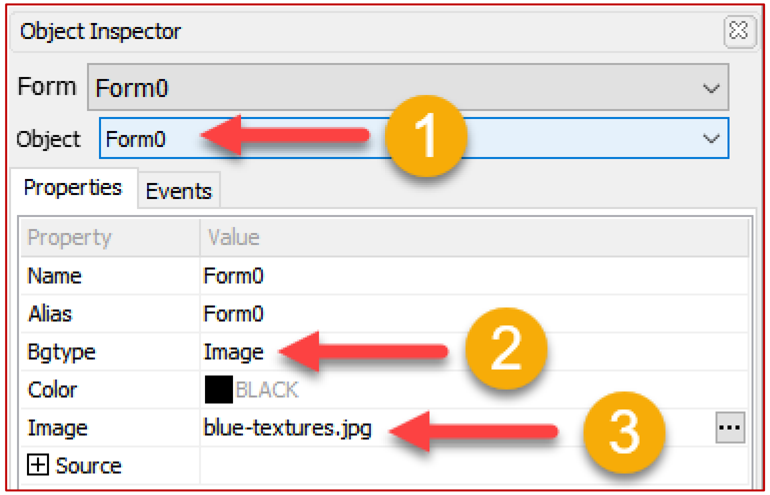
Create a New Project

For instructions on how to create a new **ViSi-Genie** project, please refer to the section "**Create a New Project**" of the application note

- ViSi-Genie Getting Started - First Project for Diablo16 Display Modules
- ViSi-Genie Getting Started - First Project for Picaso Displays
- ViSi-Genie Getting Started - First Project for Pixxi Display Modules

Design the Project

For this application, gen4-uLCD-43DT will be used for the project. Same procedure is applicable for any Picaso, Diablo16 and Pixxi display module. First, set the background image of Form0.


Object Inspector

Form: Form0

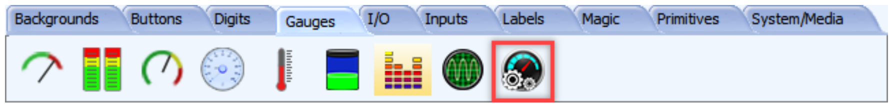
Object: Form0

Properties | Events

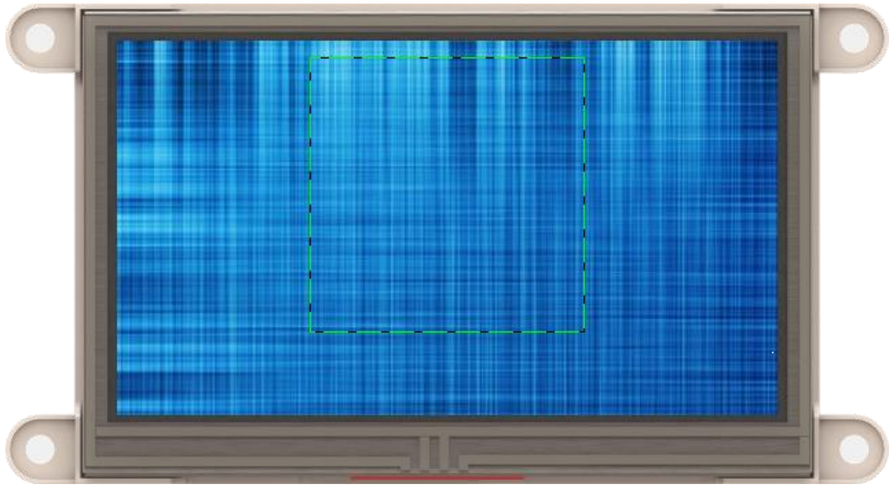
Property	Value
Name	Form0
Alias	Form0
Bgtype	Image
Color	■ BLACK
Image	blue-textures.jpg
⊞ Source	

Add a Smart Gauge Object

Add a Smart Gauge widget to your ViSi-Genie project. It can be found on the Gauges tab on the Widgets Pane.



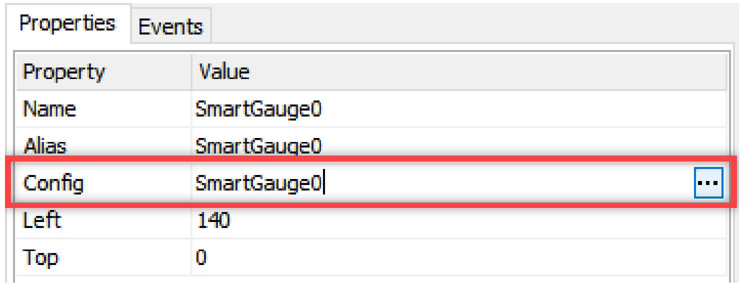
Simply click on this icon to select it. Then place it on the WYSIWYG area.



As displayed on the previous image, the widget appears empty when placed in the WYSIWYG area.

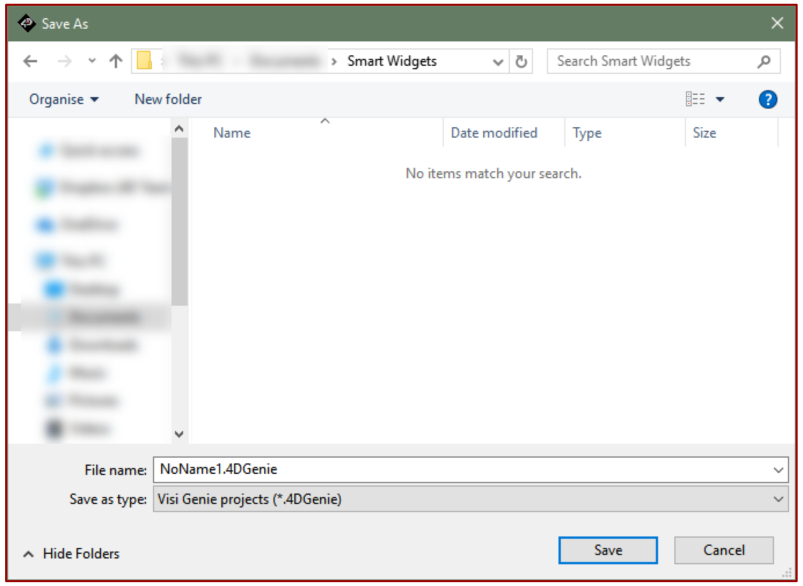
Open the Smart Widgets Editor Tool

Open the Smart Widgets Editor tool by clicking on ··· of **Config** in the Object Inspector Properties tab.



Property	Value
Name	SmartGauge0
Alias	SmartGauge0
Config	SmartGauge0
Left	140
Top	0

The tool requires that the project is already saved before the tool opens. Therefore, since on this case, it hasn't been saved yet, Workshop4 will automatically prompt the user to save

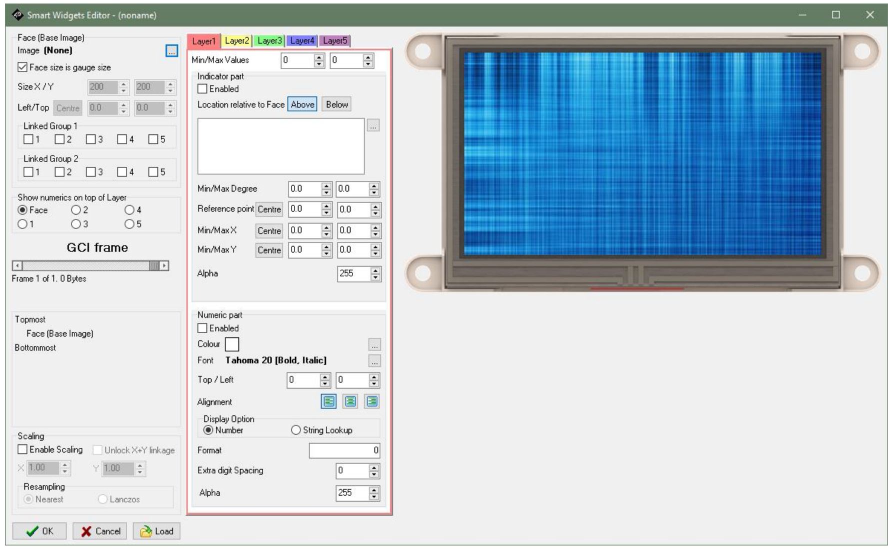


No items match your search.

File name: NoName1.4DGenie

Save as type: Visi Genie projects (*.4DGenie)

Save the project to desired location. The tool will open after the project has been saved.



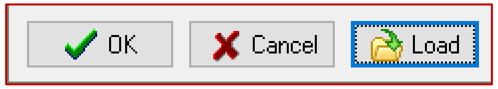
As shown in the image, this tool has a lot of parts. The next steps will focus only on the minimum tool functionalities required to make a basic circular progress bar.

For detailed discussion on how each part works, please refer to the **Smart Widgets Editor User Guide**.

Design New Gauge or Load Saved Gauge

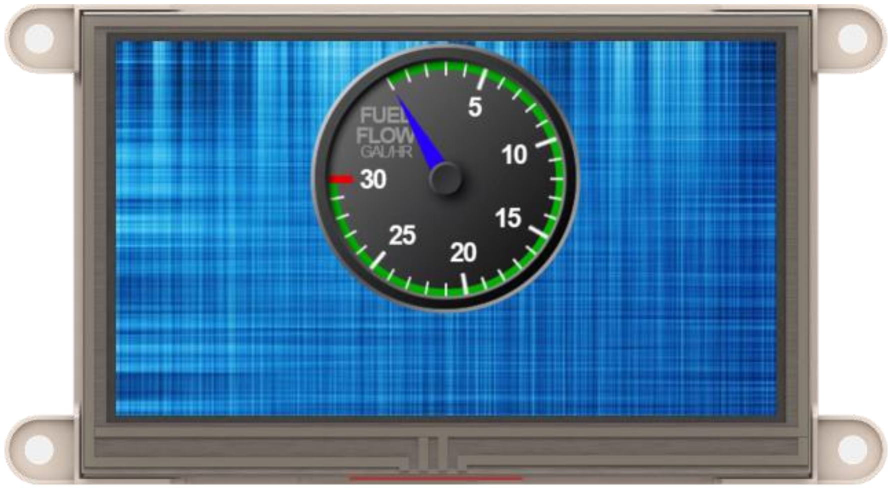
You may design a new Gauge or use a previously saved one before adding the dropdown menu.

Ensure that you have at least one unused layer for the Dropdown menu. For this appnote project, a previously saved gauge will be used.



The following buttons can be found on the bottom-left area of the editor window. Click on **Load.** It will automatically open to the directory of smart widgets from Workshop4.

Find **SimpleGauge.4Dsmart** and click **Open**.

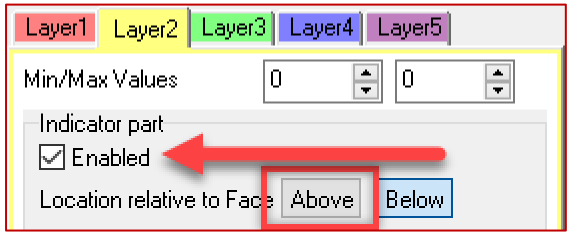


At the time, this appnote was written the gauge is still available. In case it is not available or there are any issues with the copy found, you may also find it the appnote project folder under the same filename.

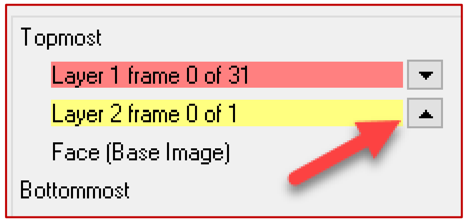
The editor should display the gauge after it loads the 4Dsmart file as shown in the previous image.

Preparing a Layer for DropDown Menu

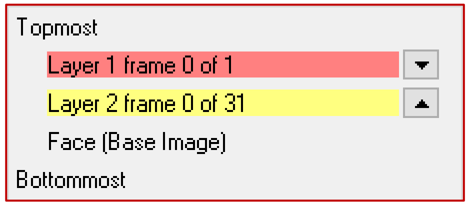
Enable another layer. This will be used for the dropdown menu. Ensure that it is the topmost layer. If you're following the same procedure as the appnote, you should be able to enable Layer 2.



Place it above the base/face image. And swap it with Layer 1.



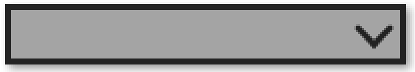
After swapping the 2 Layers, you'll notice that the 31 frames of Layer 1 is transferred to Layer 2:



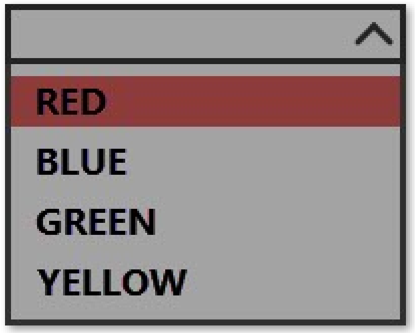
The transparent part of the base image will be filled up with liquid. This liquid part can be another image which moves vertically or even diagonally to fill the tank. To implement this, we need a layer containing a manipulated image. We will use Layer 1 for this purpose.

Designing a DropDown Menu

A dropdown menu should have both close:



And open state:



Additionally, for both states, it should show the currently selected item.

For that purpose, the open dropdown menu state needs to have a frame for each item. Additionally, there needs to be multiple duplicates of close state so that it equals the number of open states.

In this appnote, the frames used for the layer are:

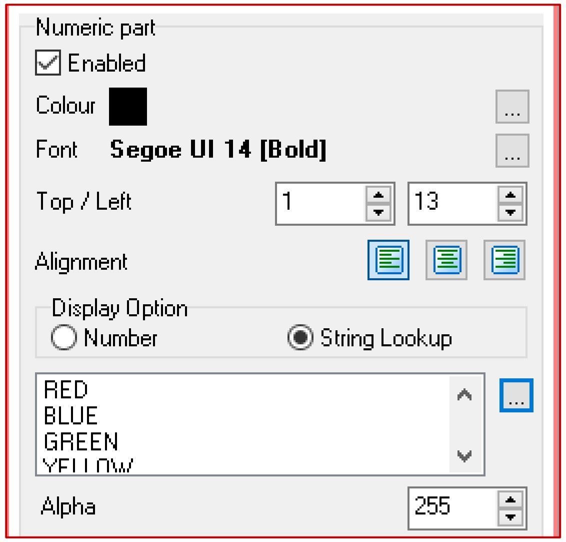
- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Open_Dropdown_RED.png
- Open_Dropdown_Blue.png
- Open_Dropdown_Green.png
- Open_Dropdown_Yellow.png

Enable the Numerical Part and set it to String Lookup. The string should contain the following:

- RED
- BLUE
- GREEN
- YELLOW
- RED
- BLUE
- GREEN
- YELLOW

Position the strings correctly and choose your desired settings.

This appnote project uses the configuration below:

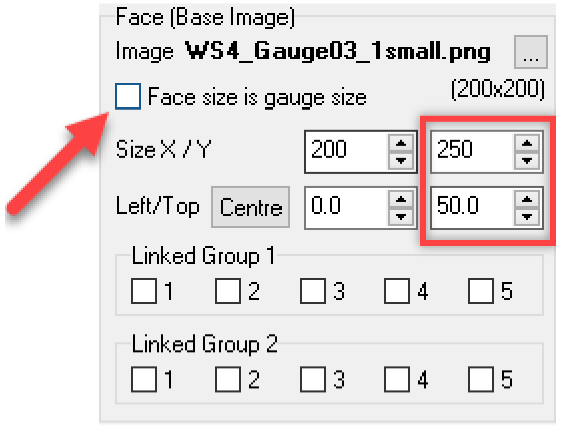


Numeric part
[✓] Enabled
Colour ■
Font Segoe UI 14 [Bold]
Top / Left 1 13
Alignment
Display Option
◯ Number ● String Lookup

RED
BLUE
GREEN
YELLOW

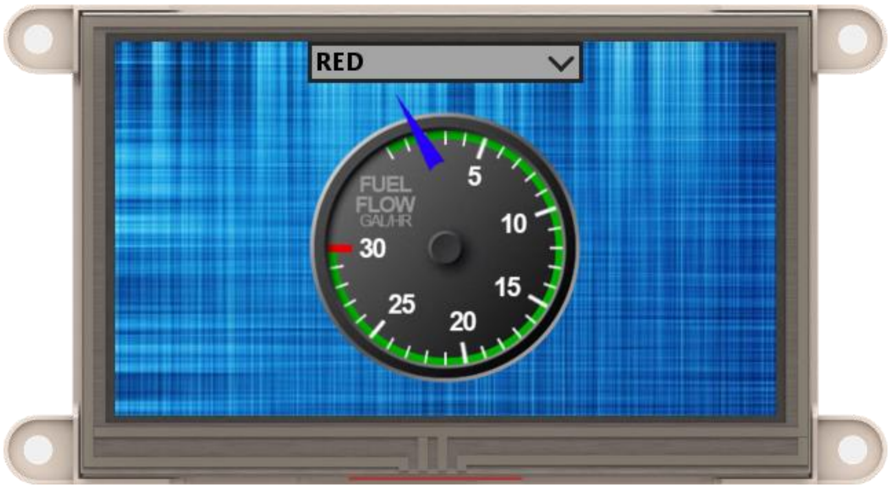
Alpha 255

Finalize the gauge by bringing the gauge down so that when the dropdown menu is open, the gauge is not covered by it.

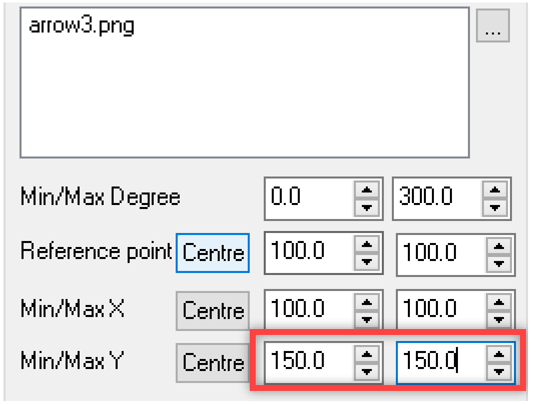


Face (Base Image)
Image WS4_Gauge03_1small.png
☐ Face size is gauge size (200x200)
Size X / Y 200 250
Left/Top Centre 0.0 50.0
Linked Group 1
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5
Linked Group 2
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Adjust the face size and top position accordingly.



You'll notice that the needle was left behind by the base image.



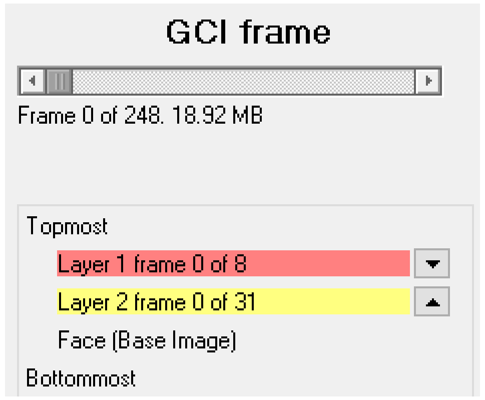
arrow3.png

Min/Max Degree 0.0 300.0
Reference point Centre 100.0 100.0
Min/Max X Centre 100.0 100.0
Min/Max Y Centre 150.0 150.0

Adjust the Y position accordingly

Checking the Order of Frames

The order of frames can easily be observed by utilizing the GCI frame slider.



Frame 0 of 248. 18.92 MB

Layer 1 frame 0 of 8

Layer 2 frame 0 of 31

You can slowly observe the frame by utilizing the buttons at both ends of the slider.

The first few press on the button on the right end of the slider will show that the dropdown menu changes state from frame 0 to 7; moving a total of 8 frames. Afterwards, it will return to the layer's frame 0 (1st frame) which is when the gauge will move 1 unit.

The GCI frame can be computed as shown below:

$$frame_{GCI} = frame_{L1} + 8\,(frame_{L2})$$

Where

$frame$ is from 0 to N.

The Layer1 frames can also be broken to 2 values.

Following the arrangement of the frames:

- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Closed_Dropdown.png
- Open_Dropdown_RED.png
- Open_Dropdown_Blue.png
- Open_Dropdown_Green.png
- Open_Dropdown_Yellow.png

It can be noticed that it will finish going through the 4 colors before moving to the Open state which will then go through the 4 colors then go back to the first frame.

That being the case, the formula can be expanded to:

$$frame_{GCI} = frame_{L1} + 8\,(frame_{L2})$$

$$frame_{GCI} = 4\,(state) + color + 8\,(frame_{L2})$$

Where:

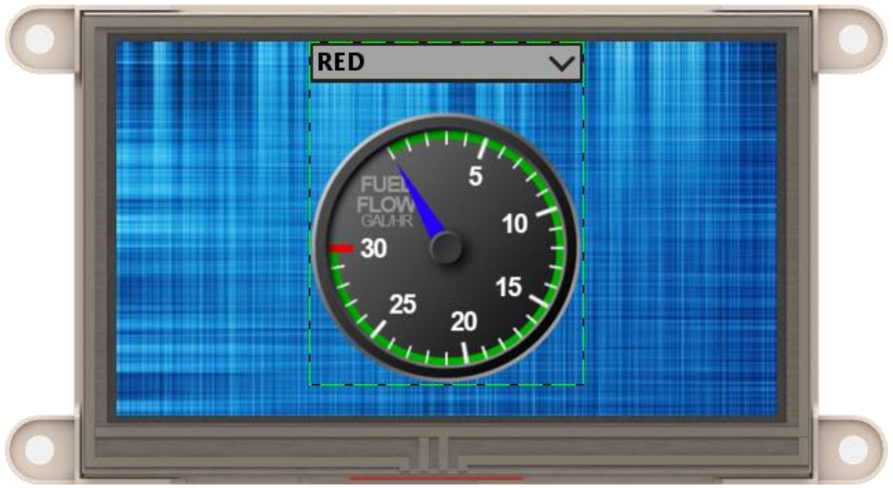
$state$ is 0 for closed and 1 for opened dropdown menu

$color$ is 0 to 3 for Red, Blue, Green and Yellow

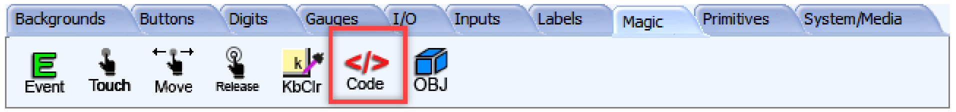
This equation will be used in the code later.

Use SmartGauge as Input Object

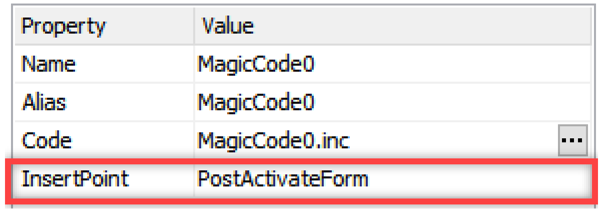
SmartGauges are non-input objects by default. For this step, 4DGL code will be added to the project.



Touch functionality should be enabled for the SmartGauge object when currently at the form containing it and disabled if not. For that add a **Magic Code** to the project.



Set the insert point to be **PostActivateForm**



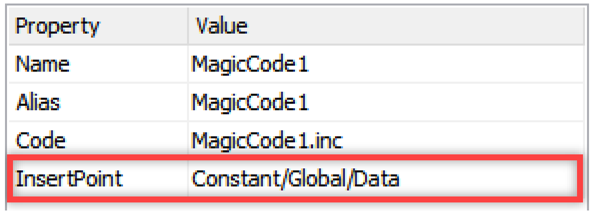
Property	Value
Name	MagicCode0
Alias	MagicCode0
Code	MagicCode0.inc
InsertPoint	PostActivateForm

Open the editor by clicking on ··· of the row **Code**

Then add the following lines of code.

```
if (CurrentForm == 0)
  img_ClearAttributes(hndl, iSmartGauge0, I_TOUCH_DISABLE);
else
  img_SetAttributes(hndl, iSmartGauge0, I_TOUCH_DISABLE);
endif
```

Add another **Magic Code** and set its insertion point to **Global**



Property	Value
Name	MagicCode1
Alias	MagicCode1
Code	MagicCode1.inc
InsertPoint	Constant/Global/Data

```
var gaugeValue[4];
var selectedItem;
var menuState;
```

Insert the above lines of code to declare the following variables and array. Notice that they are values relating to the frame's formula.

We need to program what it will do when a touch event occurs. Add **Magic Touch**, **Magic Release** and **Magic Move** objects to the project. Then add the following lines of code to each of the three.

```
if (ImageTouched == iSmartGauge0)
    ImageTouched := -1;
endif
```

Open **Magic Touch** object add these lines of code inside the ***if*** condition.

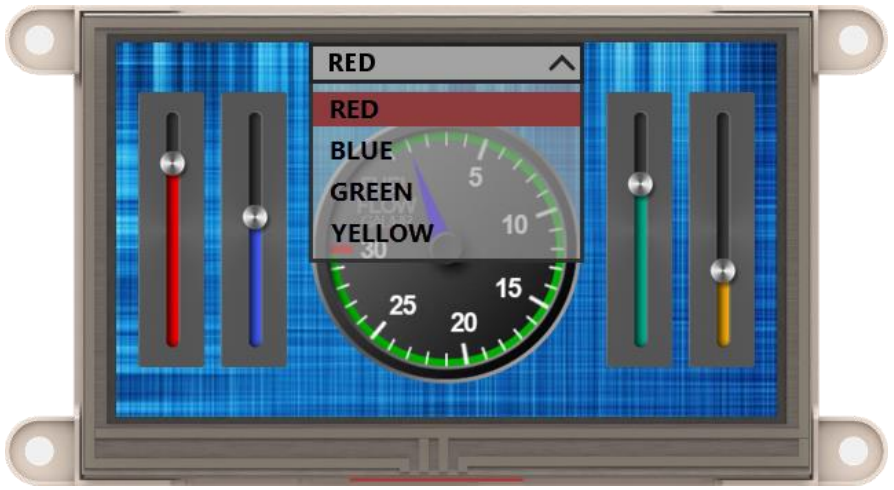
```
var item, topPos;
topPos := img_GetWord(hndl, iSmartGauge0, IMAGE_YPOS);
item   := (TouchYpos - topPos) / 32 - 1;

if (item < 0)
    menuState     := !menuState;
else if (menuState && item < 4)
    selectedItem := item;
endif

updateGauge();
```

As you may have noticed, there is a function used in the above code which we haven't defined yet. Go back to the **Global Magic Code** and add these lines.

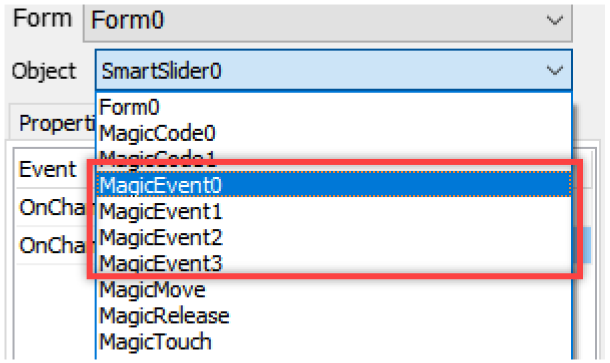
```
func updateGauge()
    var frame, gaugeVal;
    gaugeVal := gaugeValue[selectedItem];
    frame    := 4 * menuState + selectedItem + 8 * gaugeVal;

    WriteObject(tSmartGauge, 0, frame);
Endfunc
```

Add Multiple Input Objects

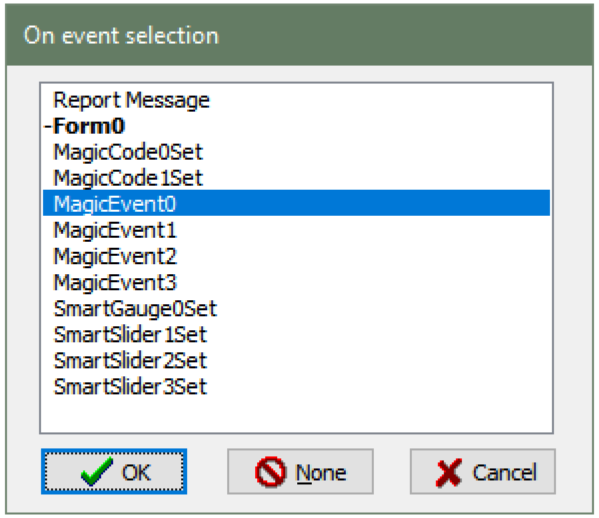
Add four slider or knob widgets to the project. The project included with this appnote uses four SmartSliders.



Add a Magic Event for each of the four input objects.



Set the **OnChanging** event of each input object to a Magic Event.



Ensure that each of them has unique **Magic Events**

For each of the **Magic Event** objects. Insert these lines of code inside the function.

```
gaugeValue[n] := newval;
updateGauge();
```

You need to replace ***n*** with 0 to 3 so it is unique for each **Magic Event**

Run the Program

For instructions on how to save a **ViSi-Genie** project, how to connect the target display to the PC, how to select the program destination, and how to compile and download a program, please refer to the section "**Run the Program**" of the application note

- ViSi-Genie Getting Started - First Project for Diablo16 Display Modules
- ViSi-Genie Getting Started - First Project for Picaso Displays
- ViSi-Genie Getting Started - First Project for Pixxi Display Modules