APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

# ViSi-Genie Starter Kit Demos

DOCUMENT DATE: **13th APRIL 2019**
DOCUMENT REVISION: **1.1**

WWW.4DSYSTEMS.COM.AU

# Description

This application note is dedicated to illustrating how to create an application that contains a simple music player, a video player, and an image viewer. Some of the input and output objects used in the ViSi Genie environment are also shown. In order to get started, the following are required:

- Any of the following 4D Picaso display modules:

| | | |
|---|---|---|
| gen4-uLCD-24PT | gen4-uLCD-28PT | gen4-uLCD-32PT |
| uLCD-24PTU | uLCD-28PTU | uVGA-III |

and other superseded modules which support the ViSi Genie environment

- The target module can also be a Diablo16 display

| | | |
|---|---|---|
| gen4-uLCD-24D series | gen4-uLCD-28D series | gen4-uLCD-32D series |
| gen4-uLCD-35D series | gen4-uLCD-43D series | gen4-uLCD-50D series |
| gen4-uLCD-70D series | | |
| uLCD-35DT | uLCD-43D Series | uLCD-70DT |

Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor. The display module used in this application note is the uLCD-32PTU, which is a Picaso/diablo16 display. This application note is applicable to Diablo16 display modules as well.

- 4D Programming Cable / µUSB-PA5/uUSBPA5-II
  for non-gen4 displays (uLCD-xxx)
- 4D Programming Cable & gen4-IB / 4D-UPA / gen4-PA
  for gen4 displays (gen4-uLCD-xxx)
- micro-SD (µSD) memory card
- Workshop 4 IDE (installed according to the installation document)
- Any Arduino board with a UART serial port
- 4D Arduino Adaptor Shield (optional) or connecting wires
- Arduino IDE
- When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or has a working knowledge of the topics presented in these recommended application notes.

# Content

## Application Overview

It is often difficult to design a graphical display without being able to see the immediate results of the application code. ViSi-Genie is the perfect software tool that allows the user to see the instant results of his or her desired graphical layout with this large selection of gauges and meters that can simply be dragged and dropped onto the simulated module display.

The following are examples of objects used in this application.


Form


Video


Button


Sound

Text


Keyboard


LED Digits


Image


Panel


Cool gauge


Slider


Timer

Each object can have properties edited at the click of a button, all relevant code is produced in the user program.

There are nine different projects combined in this application. Each project uses one or more forms, which are in turn linked together by customised buttons. The user can choose a project from the menu screen, and can navigate from the project screen back to the menu screen using buttons.

## Setup Procedure

For instructions on how to launch Workshop 4, 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 Picaso Displays** (for Picaso)
or
**ViSi Genie Getting Started – First Project for Diablo16 Displays** (for Diablo16).

# Create a New Project

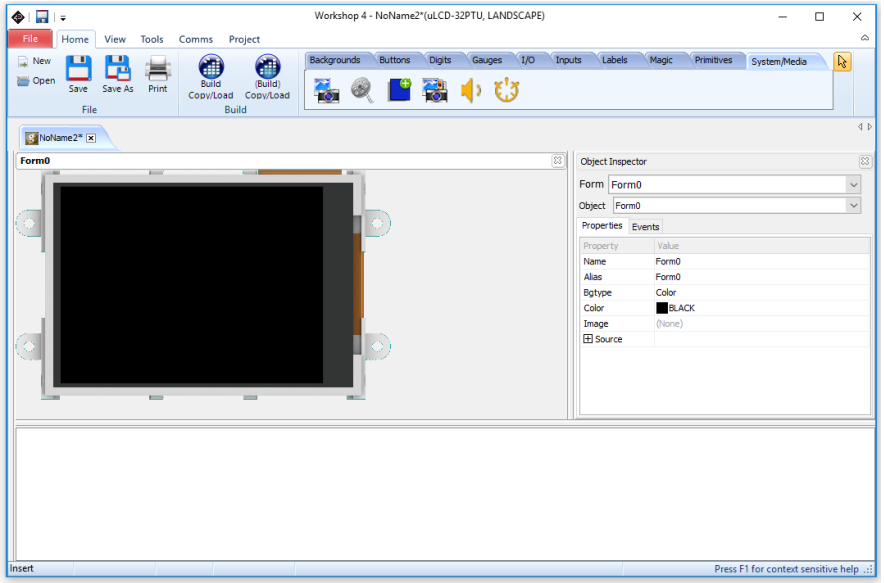
## Create a New Project

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 Picaso Displays](#) (for Picaso)
or
[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16).

# Design the Project

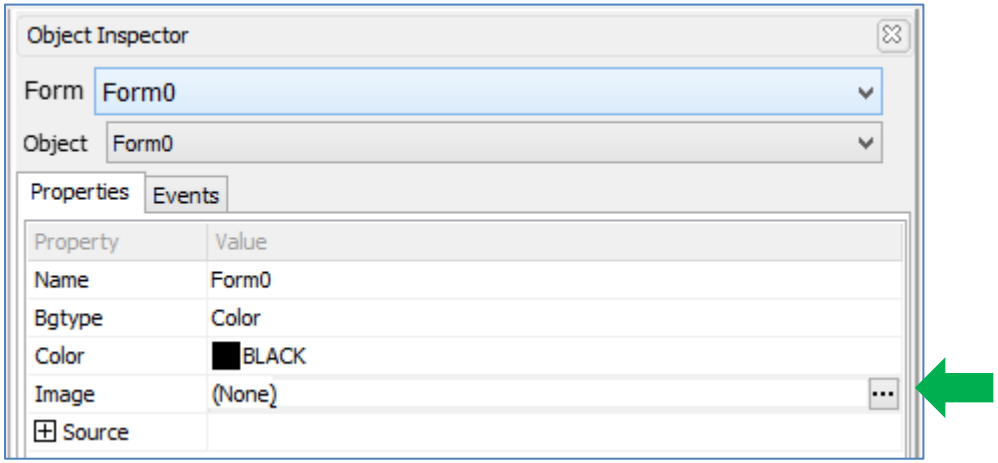
Everything is now ready to start designing the project.
**Workshop 4** displays an empty screen, called **Form0**.
A **form** is like a page on the screen. The form includes **objects**, like sliders, displays or keyboards. Below is an empty form.




## Create the Splash Screen

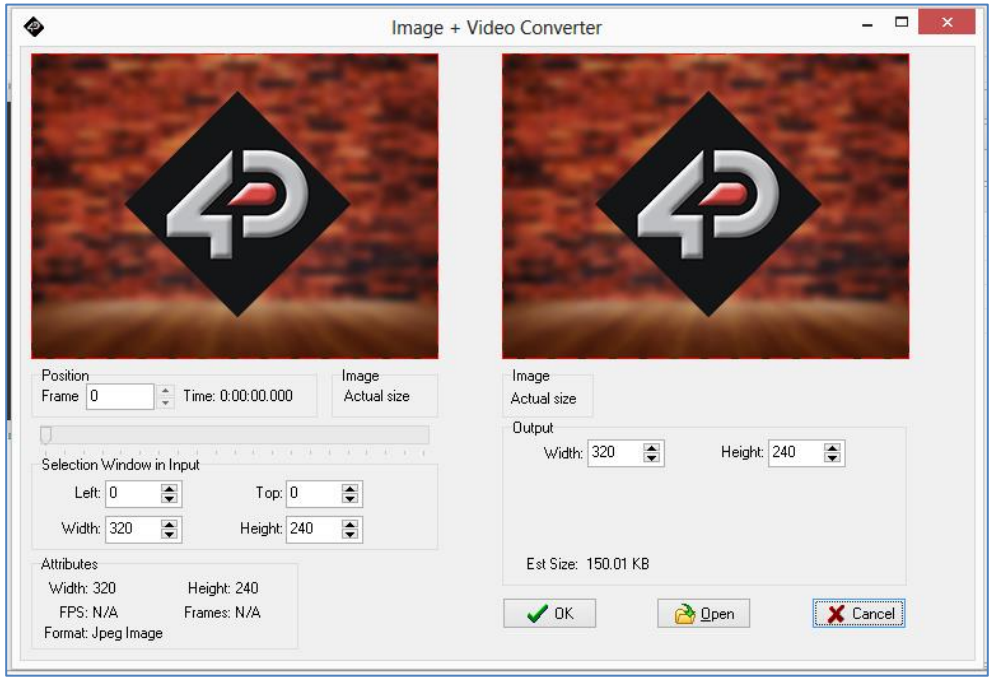
### Add a Background Image

In the **Object Inspector**, click on the ... symbol of the **Image** property.



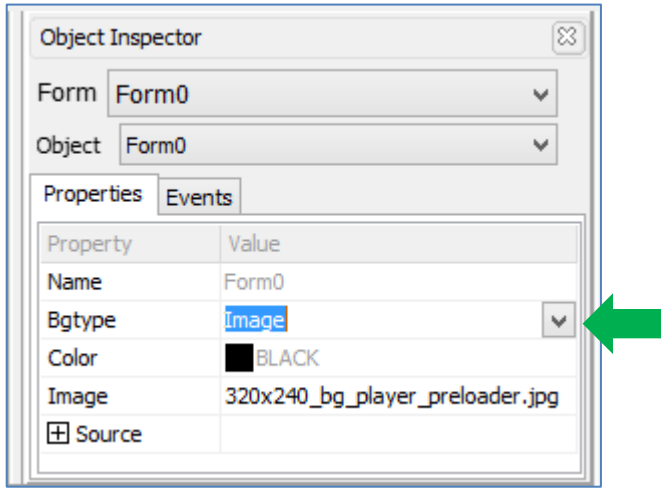

The standard Open file window appears and asks for an image. Browse for any background image file desired. The 4D splash screen image is found here: **...\StarterKitDemos.ImgData\.**

Upon clicking **Open**, the **Image-Video Converter** appears and provides all the parameters for the image.

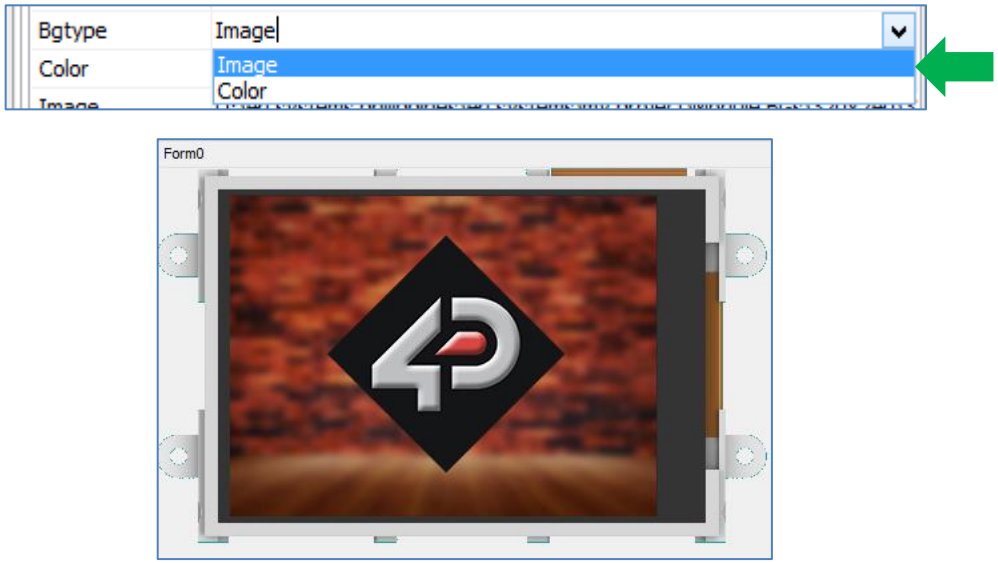



The left side of the window shows the original input image and its properties; the right side, the output image and its properties. The pixel dimensions of the 4D splash screen image file exactly fit the pixel dimension of the display LCD (in this example, 240 by 320 pixels). To know the pixel dimensions of your screen, kindly refer to the specification sheet.
The original input image will be automatically scaled and resized to fit the target screen.

Click **OK** then under the **Object Inspector**, click on the symbol of the **Bgtype** property.

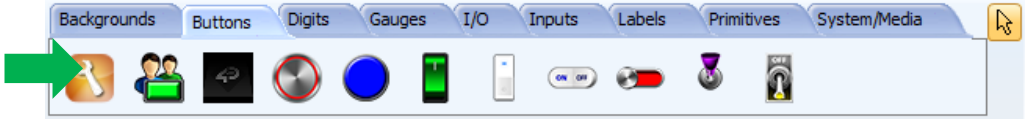



A dropdown menu appears. Choose **Image**. The background image should now be displayed.




## Create a Button

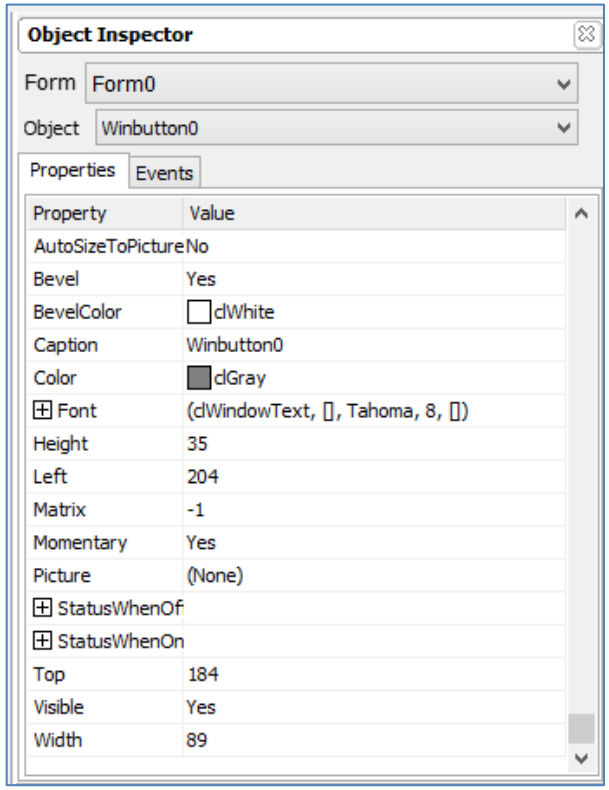
To add a button, go to the **Buttons** pane then click on the **button** icon.




Click on the **WYSIWYG** screen to place the button.


The button can be dragged to any desired location.

The **Object Inspector** on the right part of the screen displays all the properties of the new button object, **Winbutton0**.




Feel free to experiment with the different properties. To know more about buttons, refer to ViSi-Genie Advanced-Buttons.

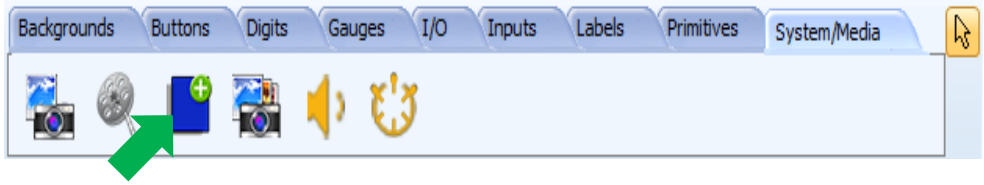
Now set the following properties for **Winbutton0**:

| Property | Value |
|---|---|
| **Caption** | click to continue… |
| **Height** | 30 |
| **Width** | 102 |
| **Left** | 216 |
| **Top** | 208 |

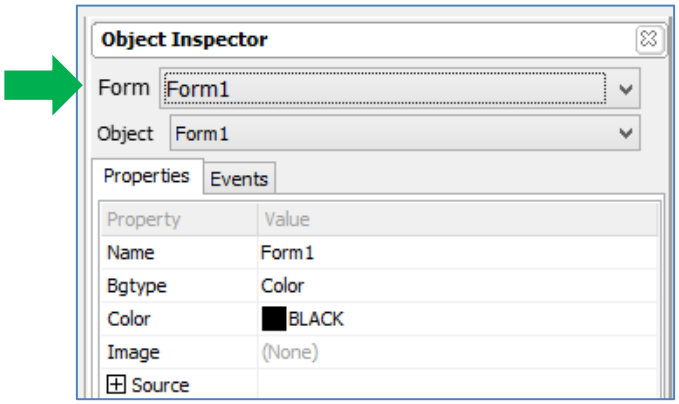
## Create the Menu Screen

### Create a New Form

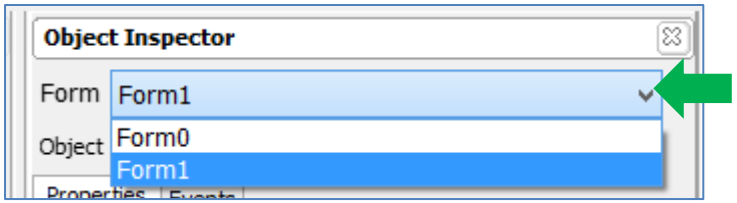
A new form is needed for the menu screen. To add a new form, go to the **System/Media** pane, and click on the **form** icon.




Upon clicking on the Form icon, a new form with a default black background is displayed on the WYSIWYG screen. Also note that the **Object Inspector** now shows a new form name, **Form1**.



To go back to **Form0** (the splash screen), click on  and choose **Form0**.




**Form0** should now be displayed on the screen. Do this to navigate between forms.

## Add a Background Image

Follow the procedure described in page 7 **(Create the Splash Screen – Add a Background Image)** to add a background image for the menu screen.

The 4D menu background image is found here: **…\StarterKitDemos.ImgData\.**

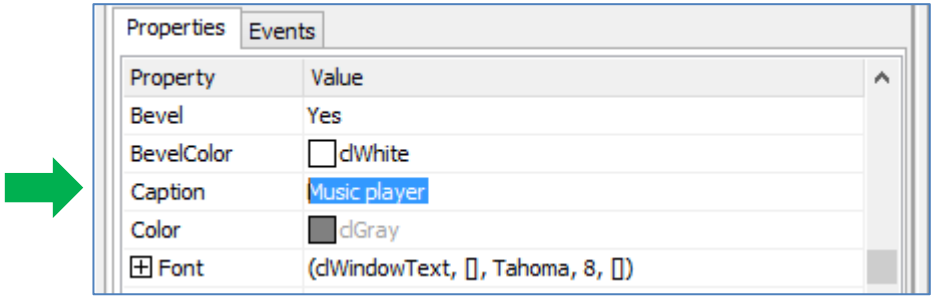
## Create the Menu Buttons

To add a button, go to the **Buttons** pane then click on the **button** icon.


Click on the WYSIWYG screen to place the button. The new button is now displayed on the screen.

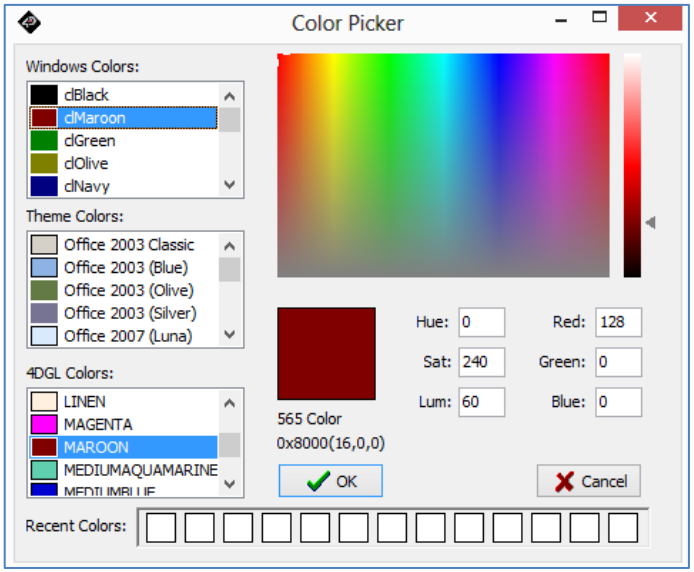

The **Object Inspector** displays the properties of the new button. Note that the name for the new button is **Winbutton1**. This property cannot be edited. By default, button caption is the same as button name.

To change the caption, go to the **Object Inspector** and edit the Caption property. Change "**Winbutton1**" to "**Music player**".

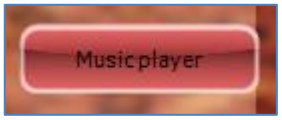


To change the button color, click on the ··· symbol of the Color line.


A color picker window shall now appear.




Choose Maroon then click **OK**. **Winbutton1** should be updated accordingly.

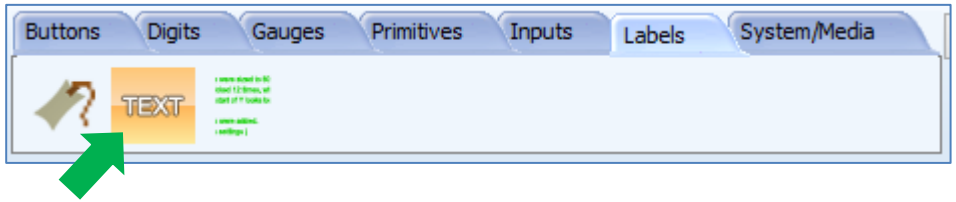



Repeat the same procedure to create the other buttons.
The buttons in the 4D menu screen have the following properties:

| Name | Caption | Color |
|---|---|---|
| **Winbutton1** | Music player | Maroon |
| **Winbutton2** | Video player | Sienna |
| **Winbutton3** | Image viewer | Office 2003 (Olive) |
| **Winbutton4** | Others | clGray |

## Add a Static Text Object

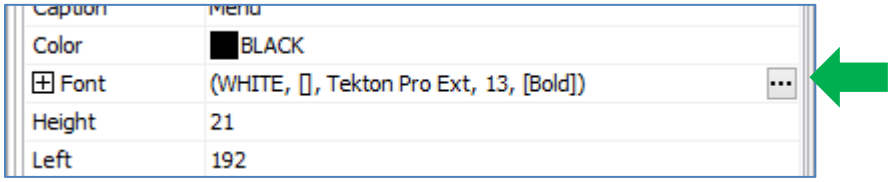
To add a **static text** object, go to the **Labels** pane then click on the **static text** icon.




Click on the WYSIWYG screen to place the text object.


The object **Statictext0** now appears on the screen. It can be dragged to any desired location. The **Object Inspector** displays the properties of the object. Change the caption from "**Statictext0**" to "**Menu**".

To change the font properties, click on the ··· symbol of the **Font** property.



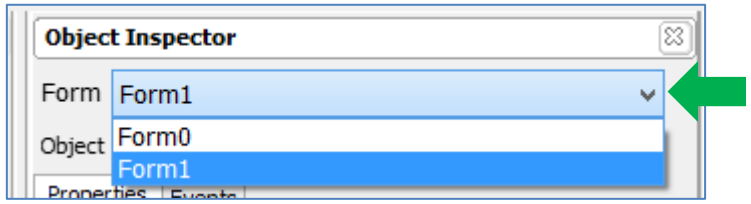

A font editor window will now appear. Choose the desired font properties then click **OK**.

The menu screen is now complete.


## Linking Forms

Now there are two form objects, **Form0** (the splash screen) and **Form1** (the menu screen). Upon power up, the module will display the splash screen, then the user will press the "click-to-continue…" button (**Winbutton0** in this example) to go to the menu screen. To do this, we associate **Winbutton0** to the act of opening **Form1** (menu screen).

Go back to **Form0** by clicking on the symbol and choosing **Form0** in the **Object Inspector**.

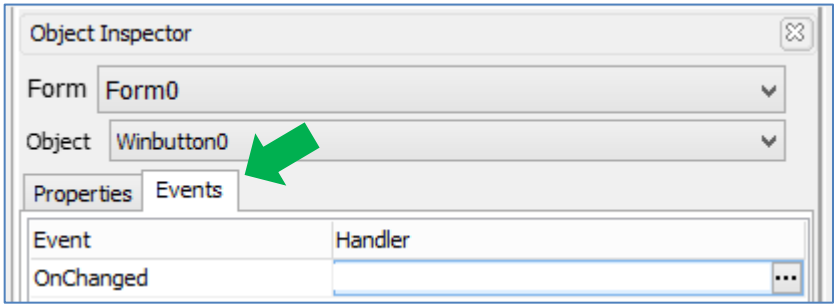



**Form0** should now be displayed on the WYSIWYG screen.

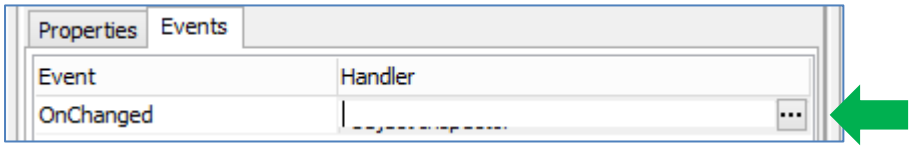
Click on **Winbutton0** to highlight it.


When **Winbutton0** is highlighted, its properties are displayed in the **Object Inspector**.

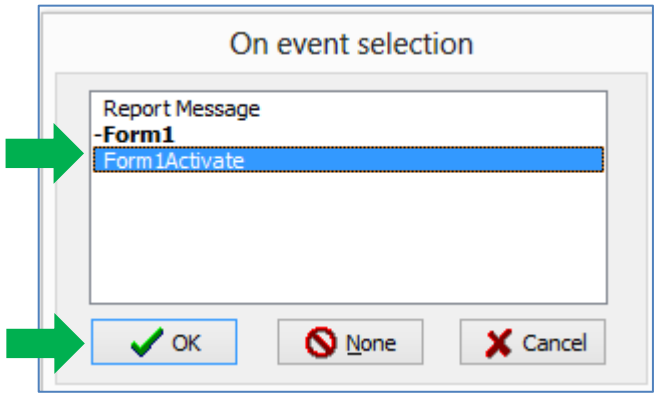
Now click on the **Events** pane.




**Winbutton0** generates the event **onChanged** when pressed. We will associate this to **Form1**. Click on the ··· symbol in the **onChanged** line.




An **On event selection** window appears. Choose **Form1Activate** then click **OK**.

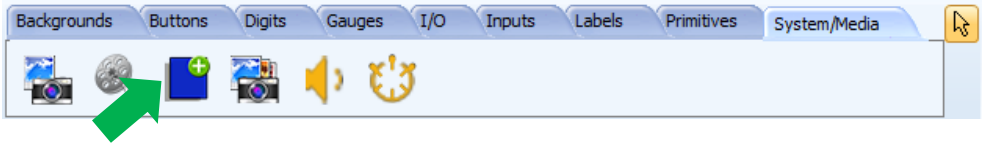



At this point you might want to test your program. Jump to page 65 **(Build and Upload the Project)** to do this. The module should display the splash screen (**Form0**) upon power up. When the "click-to-continue…" button is pressed and released, the menu screen (**Form1**) should be displayed. To know more about linking forms to buttons, refer to [ViSi-Genie Advanced-Buttons](page 23 – Button-Based Menu) (page 23 – Button-Based Menu).

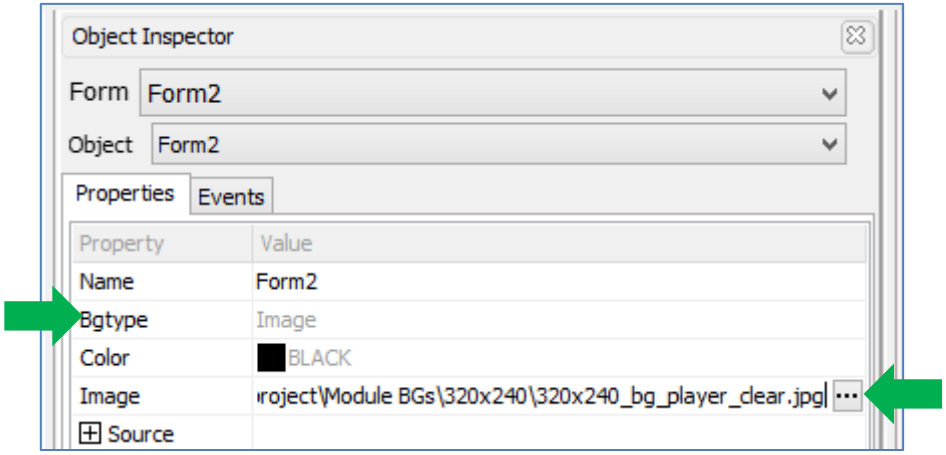
## Create a Music Player

### Create a New Form with a Background Image

Now add a new form for the music player. In this example, this is **Form2**.




Add a background image to the form. The 4D music player background image is found here: **…\StarterKitDemos.ImgData\.**



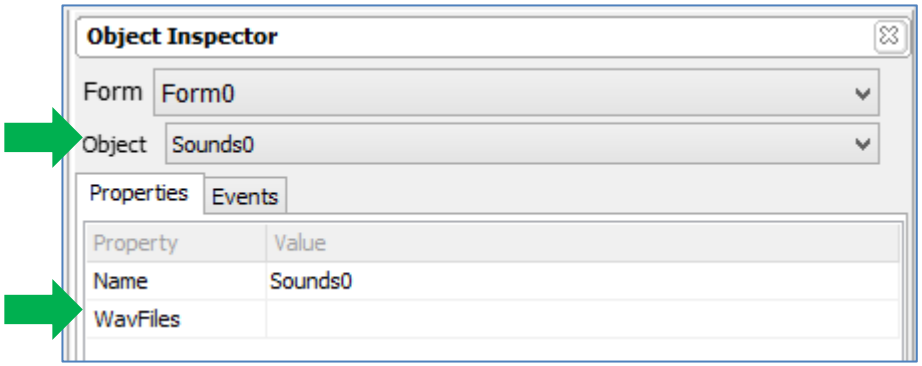

The background image is now displayed.


## Add a Sounds Object

Go to the **System/Media** pane and click on the Sounds icon.

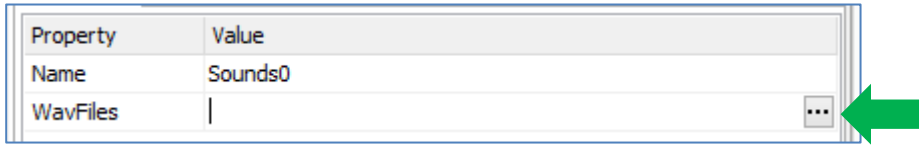

There is no need to click on the WYSIWYG screen to place it as the Sounds object is a hidden object. Also, note that the WYSISWYG screen automatically displays **Form0**. The Sounds object is always under Form0. Only one Sounds object can be added, but it can contain multiple tracks.

The **Object Inspector** now shows the newly added **Sounds0** object. It is empty and contains no tracks to play.

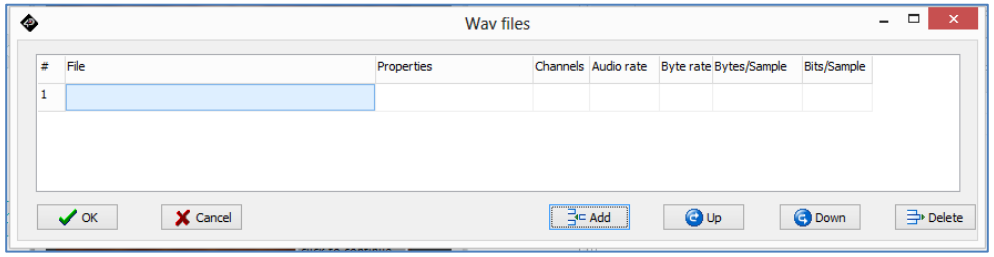


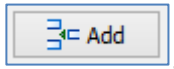

## Add Tracks to the Sounds Object

Click on the **WavFiles** property and click on ⋯.

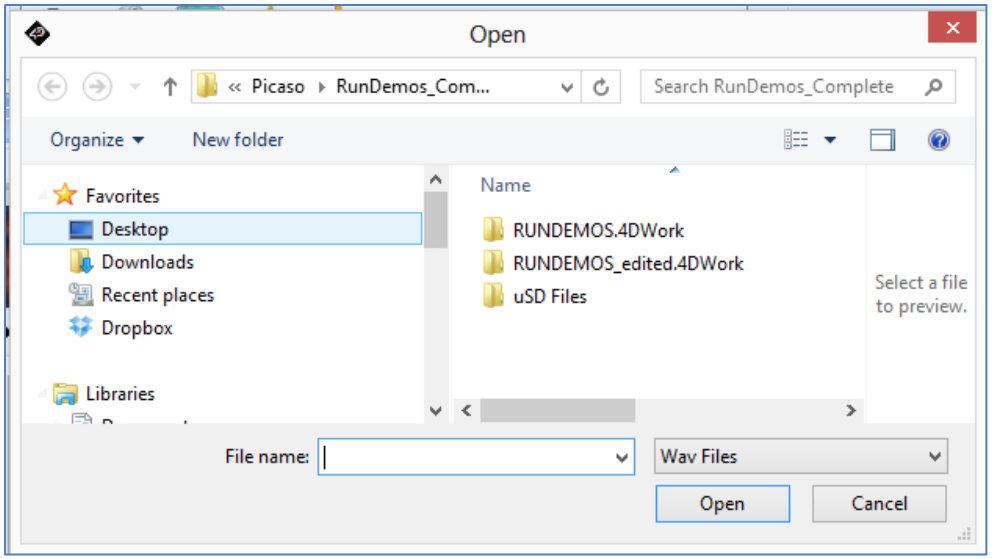



A WAV files window appears and shows all the tracks available.

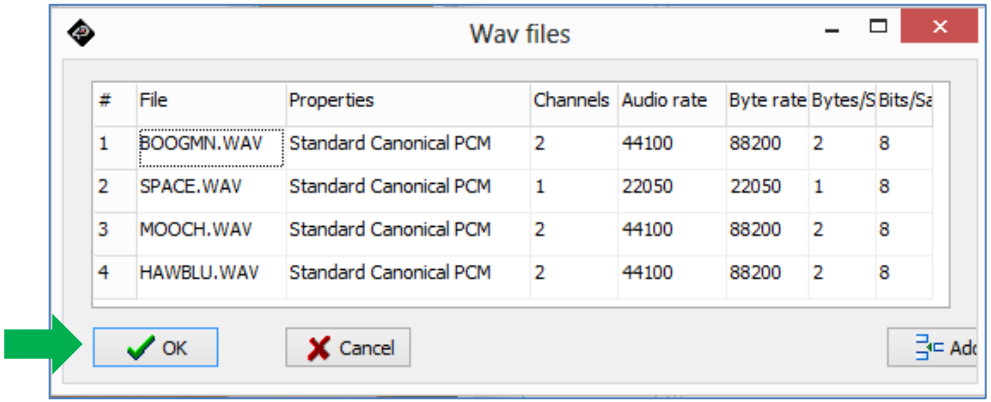


The list is empty. To add a track, click on the Add button, 

.
The standard Open file window appears.




Browse for any desired WAV file, and click **Open**. The track list will be updated. In this example, four WAV files are added. The wav files used in this example are found here: **…\StarterKitDemos.ImgData\.**




## Add Customized Track Buttons to Control the Sounds Object

Go back to **Form2** (the music player screen) and add a button object.


Change the Caption property value from "Winbutton5" to "Track 1".

A button can display an icon. Click on  in the Picture property field.


The standard Open file window asks for an image. Browse for any desired image file. The image file used in this example is located here: **…\StarterKitDemos.ImgData\. Winbutton5** now is displayed below.

To make **Winbutton5** look nicer, input the following property values.

Properties | Events

| Property | Value |
|---|---|
| Name | Winbutton5 |
| ⊟ Appearance | |
| Alignment | Center |
| Layout | None |
| PictureAlignment | Center |
| Height | 25 |
| Left | 48 |
| Top | 60 |
| Width | 120 |

The customized button will now look like as shown below.

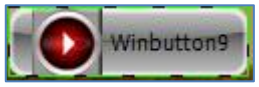

Add another button below the Track 1 button, but this time use another image file as an icon. The image file used in this example is found here: **…\StarterKitDemos.ImgData\.**

Repeat this procedure to come up with four track buttons, as shown below. Change the captions accordingly.


## Add Customized Play, Pause, and Stop Buttons to Control the Sounds Object

Now add another button to the music player screen. This will be the **Play** button. In the Object inspector, click on ··· in the Picture property field.


The standard Open file window asks for an image. Browse for any desired image file to be used as a play button icon. The image file used in this example is located here: **...\StarterKitDemos.ImgData**. **Winbutton9** is now displayed below.




Now apply the following property values to the Play button.

| Property | Value |
|---|---|
| Name | Winbutton9 |
| ⊟ Appearance | |
| Alignment | Center |
| Layout | None |
| PictureAlignmer | Center |
| Caption | |
| Height | 35 |
| Left | 56 |
| Top | 172 |
| Width | 35 |

The final appearance of the play button is shown below.


Add two more button objects – the **Pause** and **Stop** buttons.

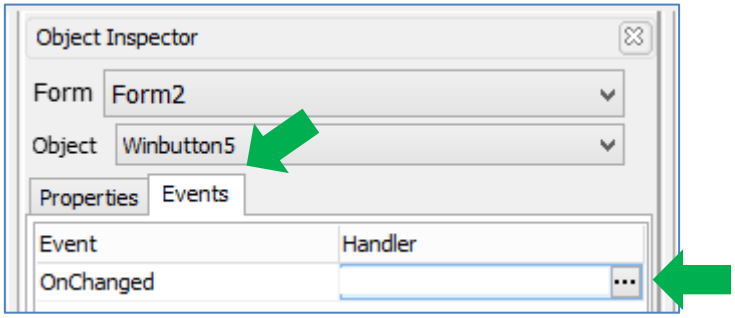
The image files used as play, pause, and stop icons in this example are located in a single folder. The Play, Pause, and Stop buttons have the same properties, only the icon image files are different.
When finished, the music player screen will look like as shown below.


## Control the Sounds Object Using the Track Buttons

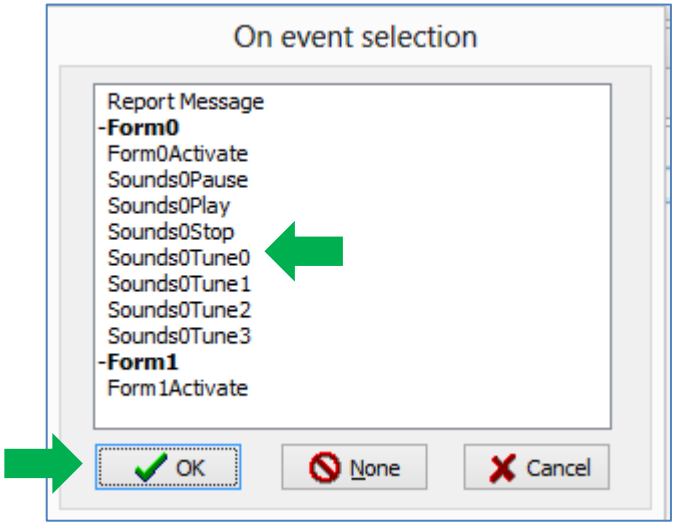
Now we will associate each of the four track buttons to the corresponding WAV file in the track list.

| Button name | Button caption | Wav file to play |
|---|---|---|
| **Winbutton5** | Track 1 | BOOGMN |
| **Winbutton6** | Track 2 | SPACE |
| **Winbutton7** | Track 3 | MOOCH |
| **Winbutton8** | Track 4 | HAWBLU |

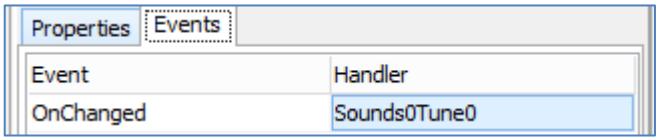
Click on **Winbutton5** or the **Track 1** button. In the Object Inspector, go to the Events pane and click on the ··· symbol.




An On event selection window now appears. Choose **Sounds0Tune0** then click **OK**.

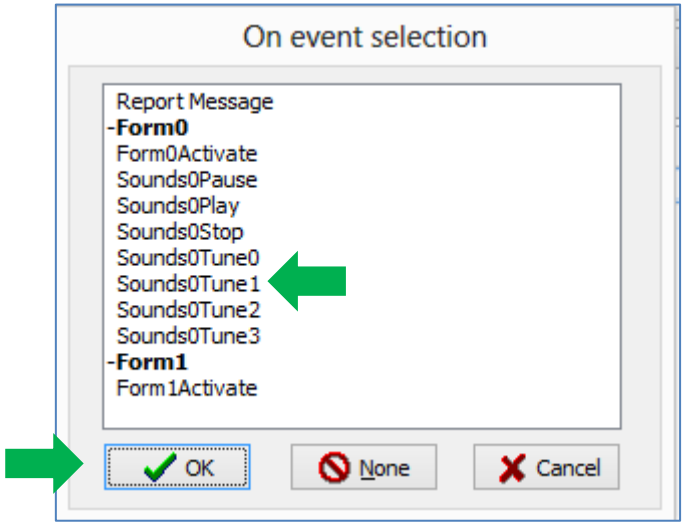



The Events pane is updated.




When the **Track 1** button is pressed and released, the **onChanged** event is raised and sends the command **Sounds0Tune0**. The command **Sounds0Tune0** stands for *Tell the Sounds0 object to play the first song in the track list*.

Now select the **Track 2** button. In the Object Inspector, go to the Events pane and click on the ... symbol to open the **On event selection** window. This time choose **Sounds0Tune1** and click **OK**.




When the **Track 2** button is pressed and released, the **onChanged** event is raised and sends the command **Sounds0Tune1**. The command **Sounds0Tune1** stands for *Tell the Sounds0 object to play the 2nd song in the track list*. The table below shows the association of the buttons to the wav files.

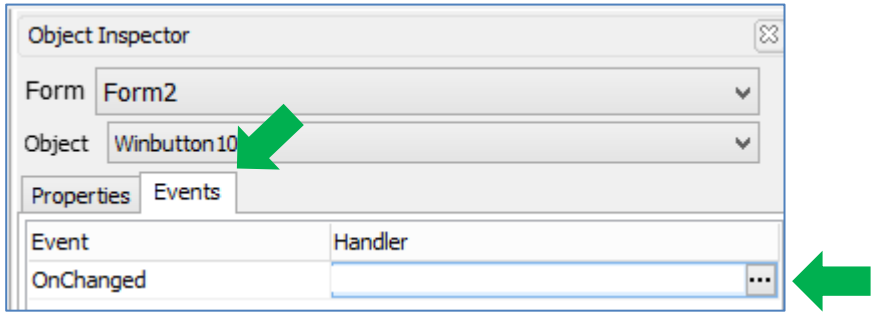
| Button name | Button caption | Command | Wav file to play |
|---|---|---|---|
| **Winbutton5** | Track 1 | Sounds0Tune0 | BOOGMN |
| **Winbutton6** | Track 2 | Sounds0Tune1 | SPACE |
| **Winbutton7** | Track 3 | Sounds0Tune2 | MOOCH |
| **Winbutton8** | Track 4 | Sounds0Tune3 | HAWBLU |

Using the table above as a guide, continue associating the two remaining buttons.

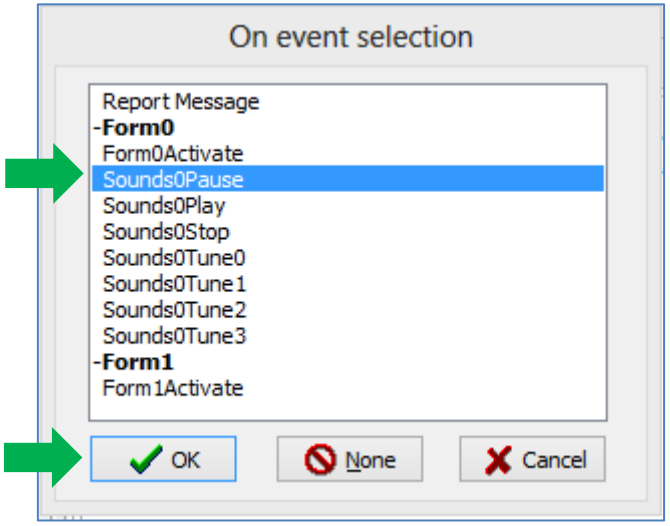
### Control the Sounds Object Using the Play, Pause, and Stop Buttons

When the **Track 1** button is pressed and released, the first song in the track list will be played. To add more control over the Sounds object, a pause button can be added. On the WYSIWYG screen, click on the **Pause** button.


In the Object Inspector, go to the Events pane and click on the ... symbol to open the **On event selection** window.



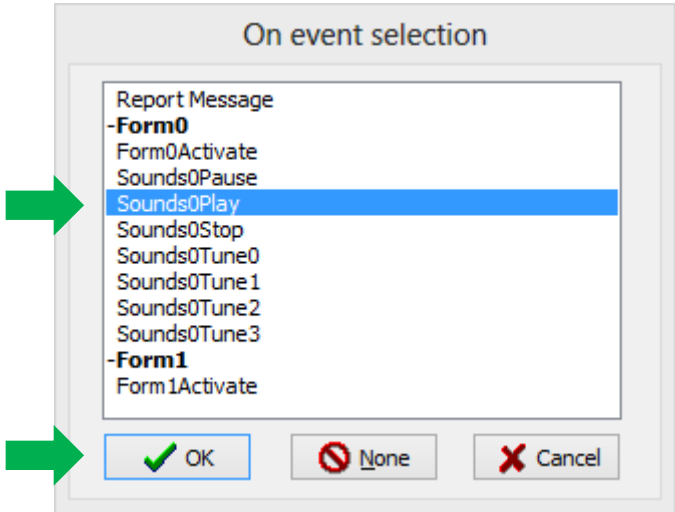

The On event selection window opens.




Select **Sounds0Pause** then click **OK**. The command **Sounds0Pause** stands for *Tell the **Sounds0** object to pause playing the track.*

To resume playing the track, we will use the **Play** button. Click on the Play button on the WYSIWYG screen to configure it.

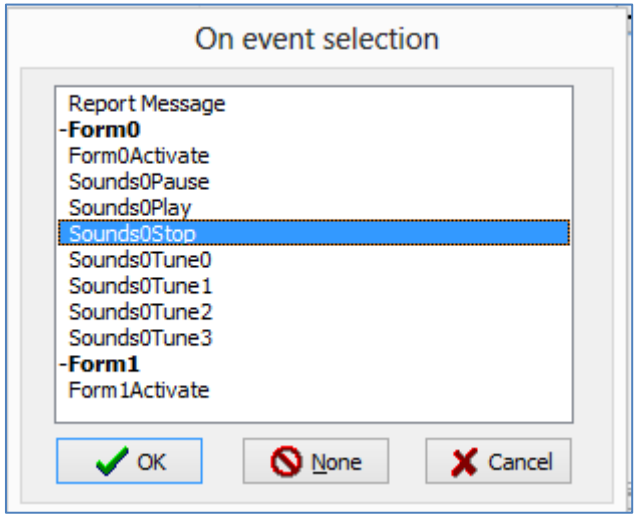

In the Object Inspector, go to the Events pane and click on the ... symbol to open the **On event selection** window. Select Sounds0Play then click OK.




The command Sounds0Play stands for *Tell the **Sounds0** object to resume playing the track being currently paused.*

To stop playing the track, the **Stop** button can be configured to send the command **Sounds0Stop** when pressed and released. Follow the steps taken when configuring the **Pause** and **Play** buttons. This time, select **Sounds0Stop** when the **On event selection** window appears.

**Stop button**

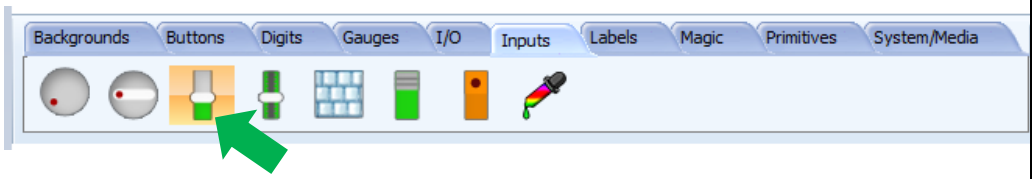



**Command**

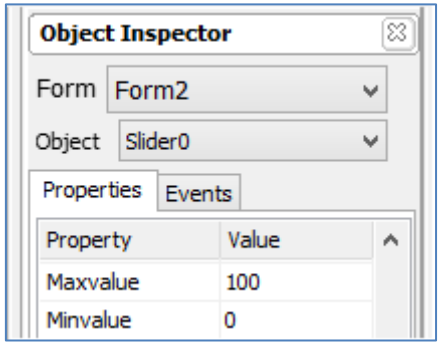
The command **Sounds0Stop** stands for *Tell the **Sounds0** object to stop playing the track*.

## Control the Volume Using a Slider Object

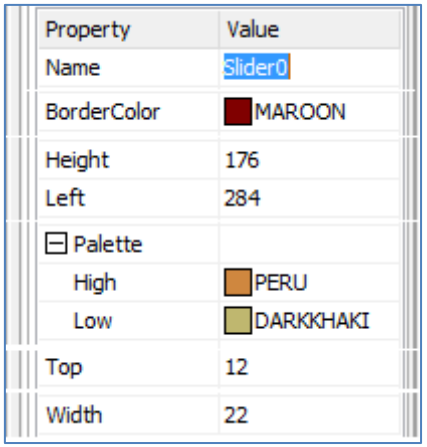
It is also possible to control the volume of the Sounds object. A **Slider** object can be used to accomplish this. Go the **Inputs** pane and click on the **slider** icon.




When the **Slider** icon is highlighted, click on the WYSIWYG screen to place a slider object. Drag the object to any desired location. Minimum volume is 0 and maximum is 100, so the properties of the slider object should be defined properly in the **Object Inspector**.

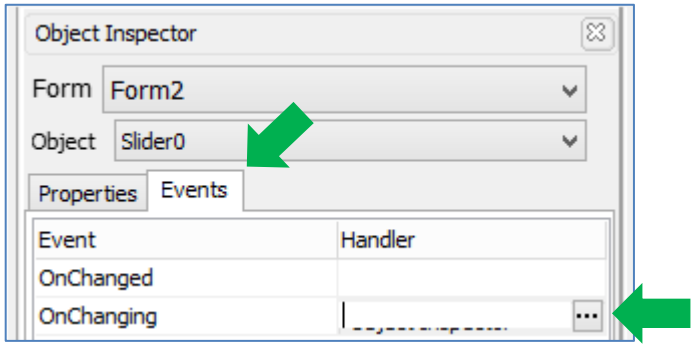



The slider object in the 4D music player screen has the following additional properties:

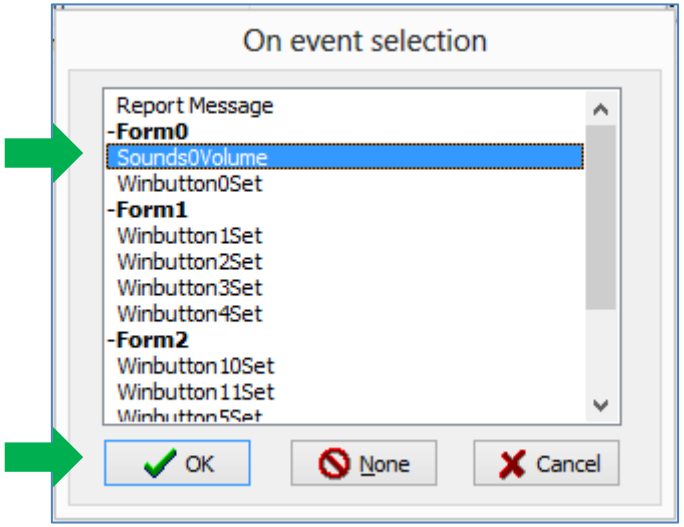


The updated WYSIWYG screen is now shown below.

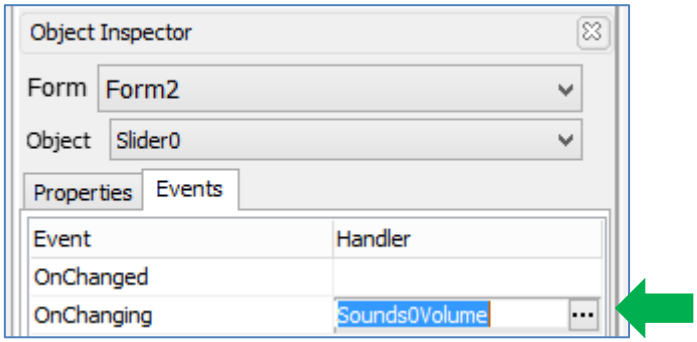

Go to the **Events** pane of the **Slider** object and click on the ... symbol in the **OnChanging** line.




The **On event selection** window appears. Select **Sounds0Volume** and click **OK**.



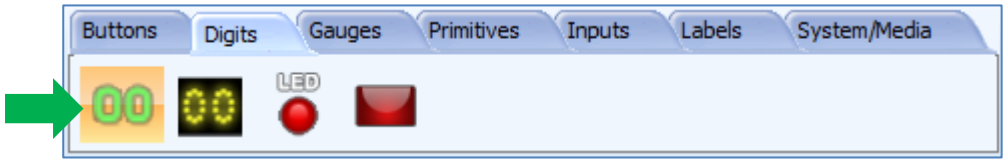

The Events pane is now updated.



The command Sounds0Volume stands for *Tell the **Sounds0** object to set volume to the value sent*.

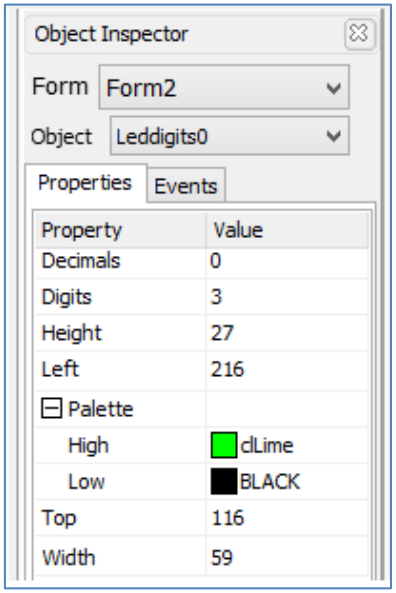
### Add a LED Digits Object to Display the Volume Level

The **LED digits** object will display the volume level when the **slider** object is moved. To add a LED digits object, go to the **Digits** pane and select the first icon.



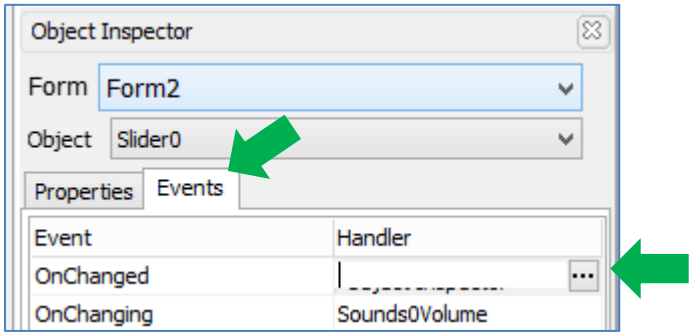

Click on the WYSIWYG screen to place it.

Go to the Object inspector and set the following property values.

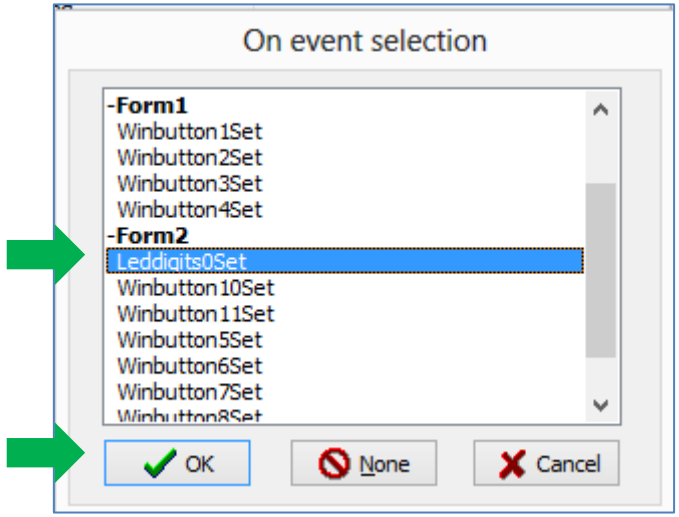



The updated appearance of the LED digits object is shown below.


Now we associate the **Slider** object to the **LED digits** object. First, select the **Slider** object. Go to the **Object Inspector** and go to the **Events** pane. Click on the ··· symbol in the **OnChanged** line.



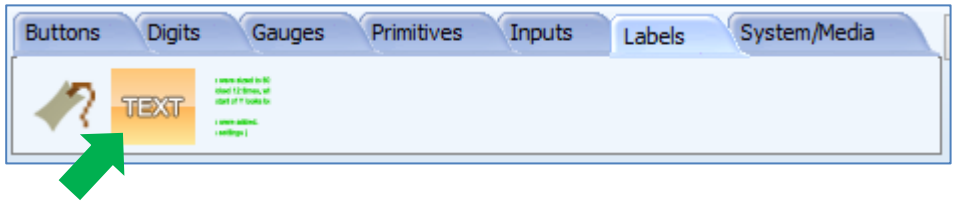

When the **On event selection** window appears, choose **Leddigits0Set** under **Form2** (the music player screen) and click **OK**.




The command **Leddigits0Set** stands for *Tell the **Leddigits0** object to display the value sent*.

## Add Static Text Objects

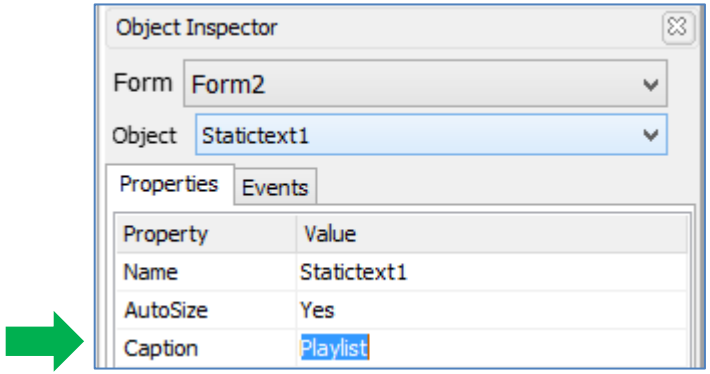
We will now add the texts “Playlist” and “volume”.




Click on the WYSIWYG screen to place a text object.

The object Statictext1 now appears on the screen. It can be dragged to any desired location. The Object Inspector displays the properties of the object.

Change the caption from "**Statictext1**" to "**Playlist**". Edit the other properties as desired.

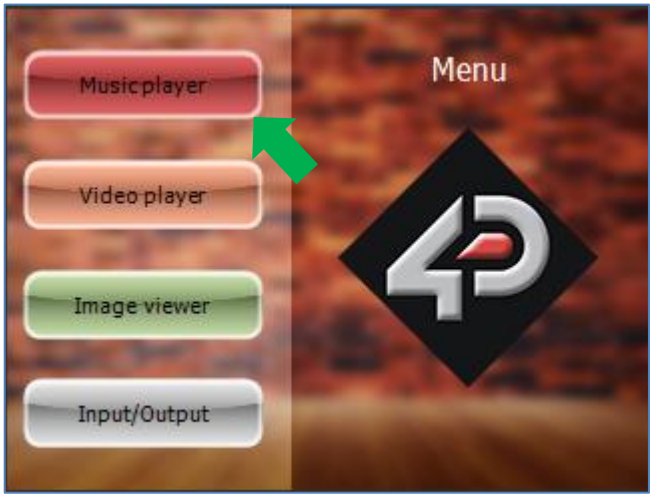



Add another text object (volume) to the WYSWYG screen by following the same procedure. This time change the caption from "**Statictext2**" to "**volume**". When finished, the screen should look similar to the one shown below.

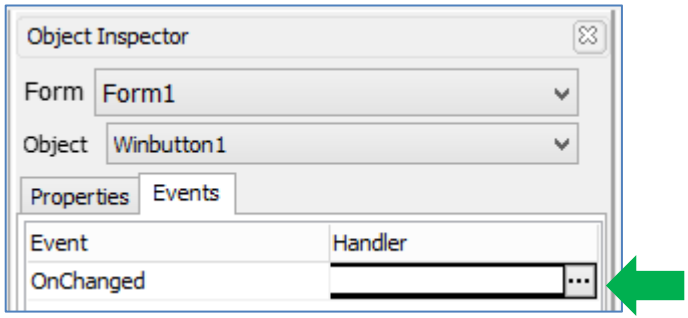

## Link the Music Player to the Menu Screen

Jump back to **Form1** (the menu screen) by choosing Form1 in the Object Inspector.

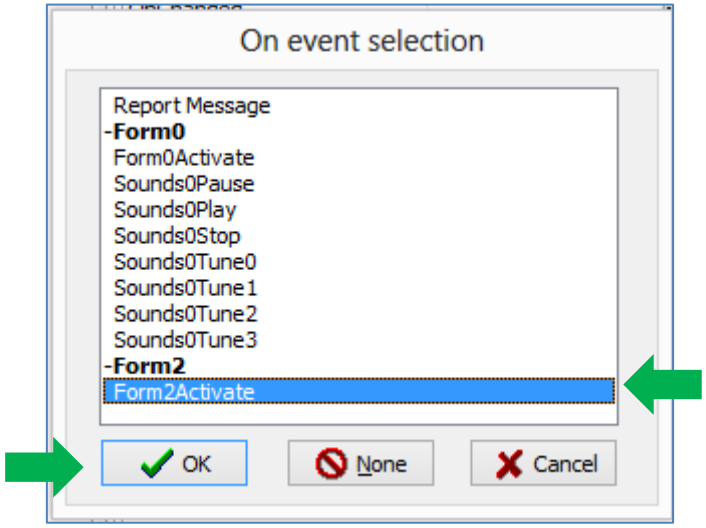

The menu screen should now be displayed. Select the Music player button.




In the Object Inspector, go to the Events pane and click on the ··· symbol.

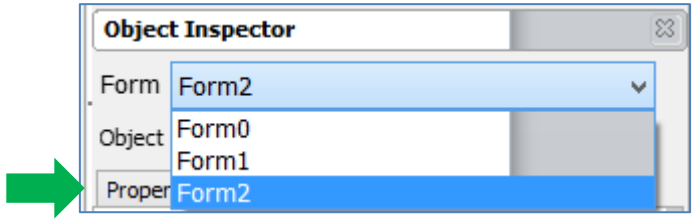



The On event selection window appears. Select **Form2Activate** then click **OK**.




When the Music player button is pressed and released, the screen will display the music player screen. Now we need to add a **Home** button to allow the user to navigate back to the menu screen.

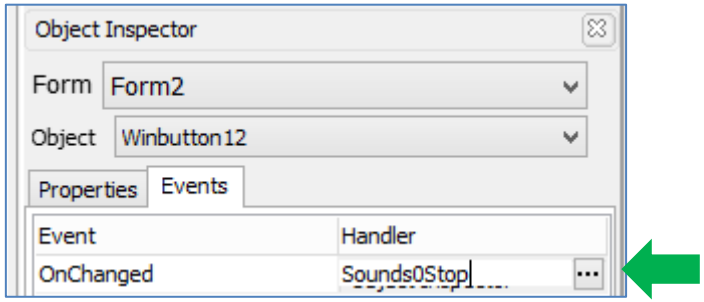
Go to **Form2** (the menu screen) by choosing Form2 in the Object Inspector.



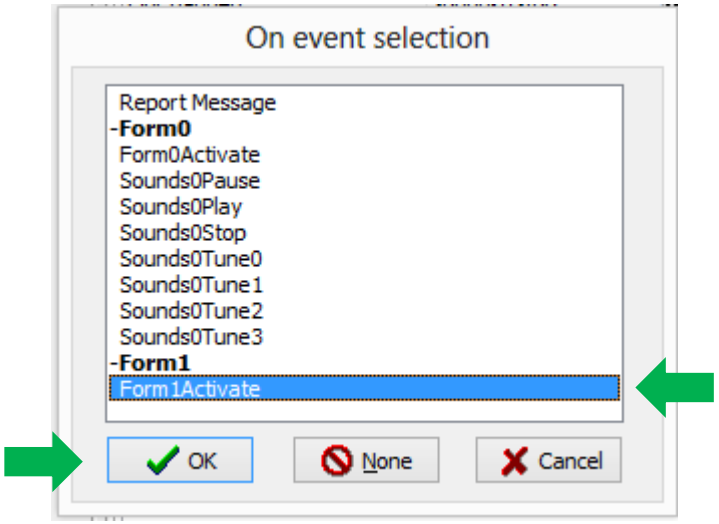

When Form2 is displayed, click on any of the Play, Pause, and Stop buttons. In this example, the Stop button is selected.


Press **Ctrl + C** to copy, and **Ctrl + V** to paste. A copy of the Stop button is placed on the screen.


Drag the button to the lower right part of the screen or to any desired location. Go to the Object Inspector and remove the existing caption property value (leave it as blank). Replace the icon image file. The 4D Home button icon image file is found here: **...\StarterKitDemos.ImgData**. Go to the Events pane and click on the ··· symbol to change the existing command.



On the On event selection window, choose **Form1Activate** instead, and then click **OK**.




When the **Home** button is pressed and released, the menu screen will be displayed. The complete music player screen is now shown below.


Now is a good time to test your program. Go to page 65 **(Build and Upload the Project)** of this application note to do this. The display module used in this example has a resolution of 240x320 pixels. For screens with a resolution other than 240x320 pixels, the location of the objects on the screen may need to be adjusted. To know the resolution of your screen, refer to the specification sheet.

To know more about the **Sounds** object, refer to ViSi-Genie Play Sound. To know more about **input** objects such as the **Slider**, refer to ViSi-Genie Inputs. To know more about the **Button** object, refer to ViSi-Genie Advanced-Buttons.

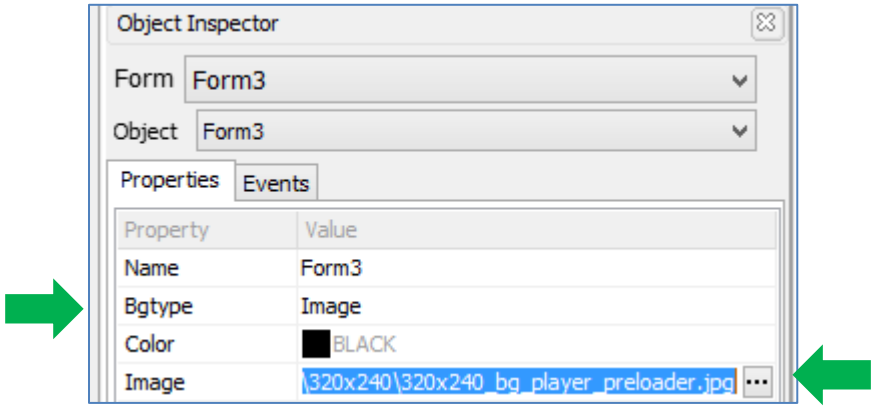
## Create a Video Player

### Create a New Form with a Background Image

The video player in this example has three forms – **Form3**, **Form4**, and **Form5**. Each form displays a video – **Video0**, **Video1**, and **Video2**.

Now create the first form for the video player, **Form3**. Go to the **System/Media** pane and click on the form icon.

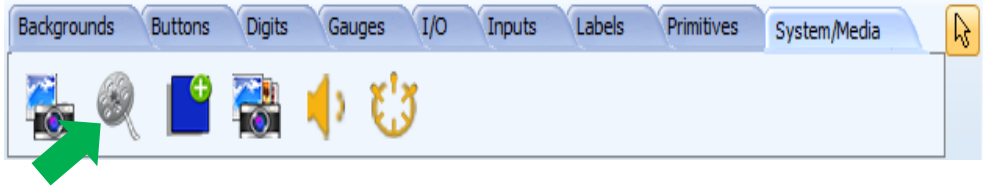

A new form will be displayed. Add a background image to the form. The 4D video player background image is located here**:**
**…\StarterKitDemos.ImgData\.**




The WYSIWYG screen is updated accordingly.

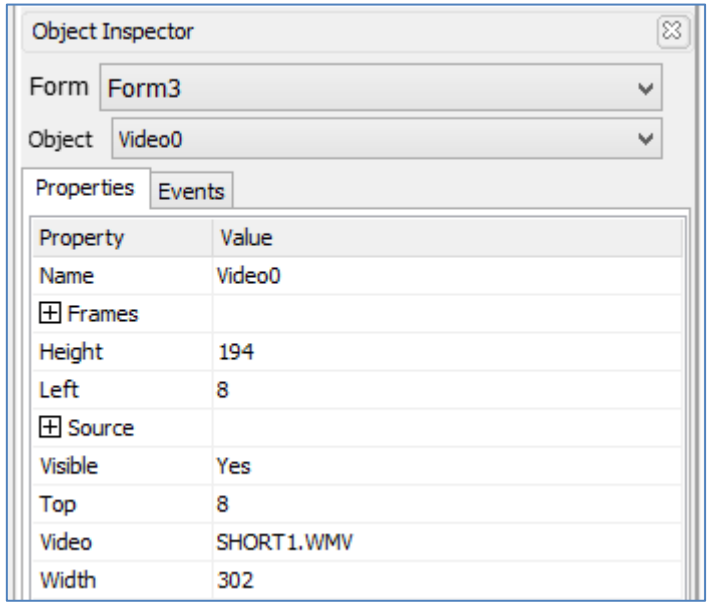

### Add a Video Object

In the System/Media pane, select the video icon. Click on the WYSIWYG screen to place it.

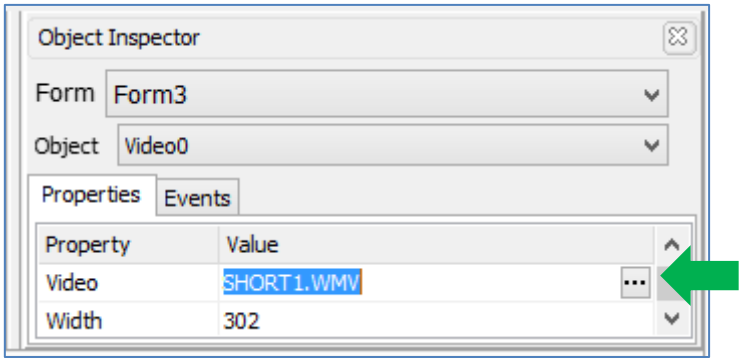



A standard Open file window appears. Browse for any desired video file. The video file used in this application is found here: **…\StarterKitDemos.ImgData\.** The new video object, **Video0**, is now placed on the screen.

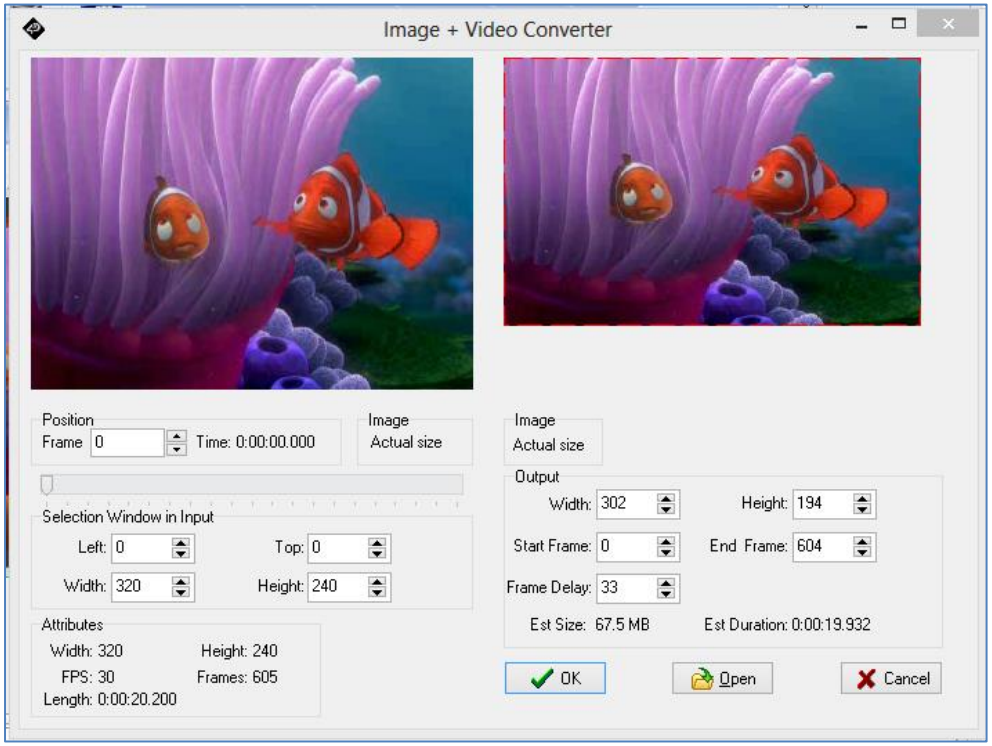

The video object can be resized and dragged to any location. Its properties can also be edited in the Object Inspector. Feel free to experiment with the values. **Video0** in this example has the following properties:




Now go to the Object Inspector and click on the ··· symbol in the Video property line.

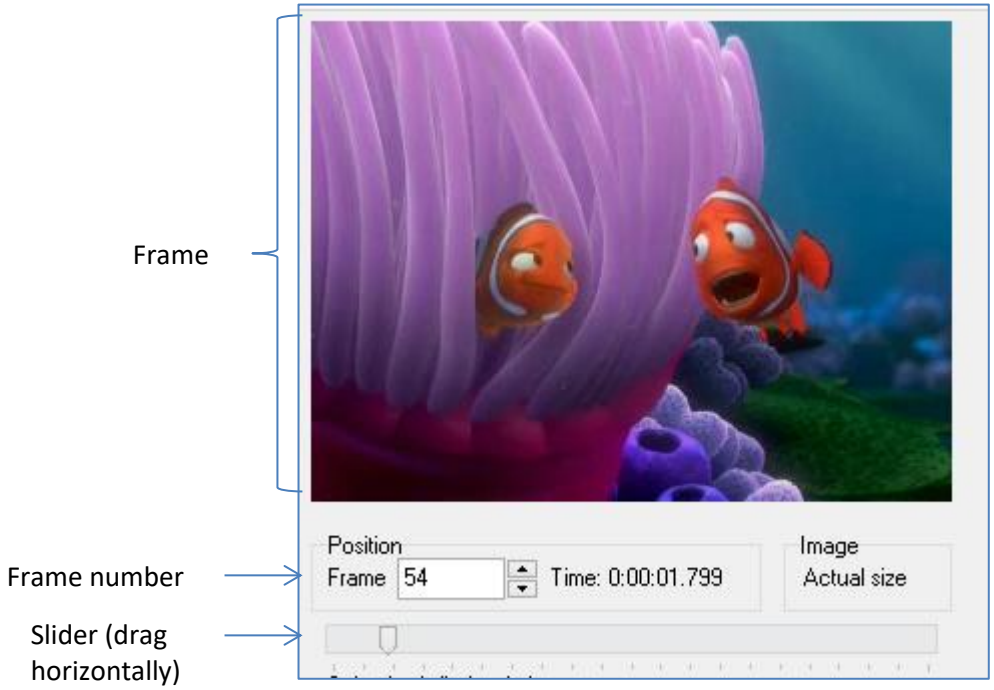



The **Image + Video Converter** window appears.

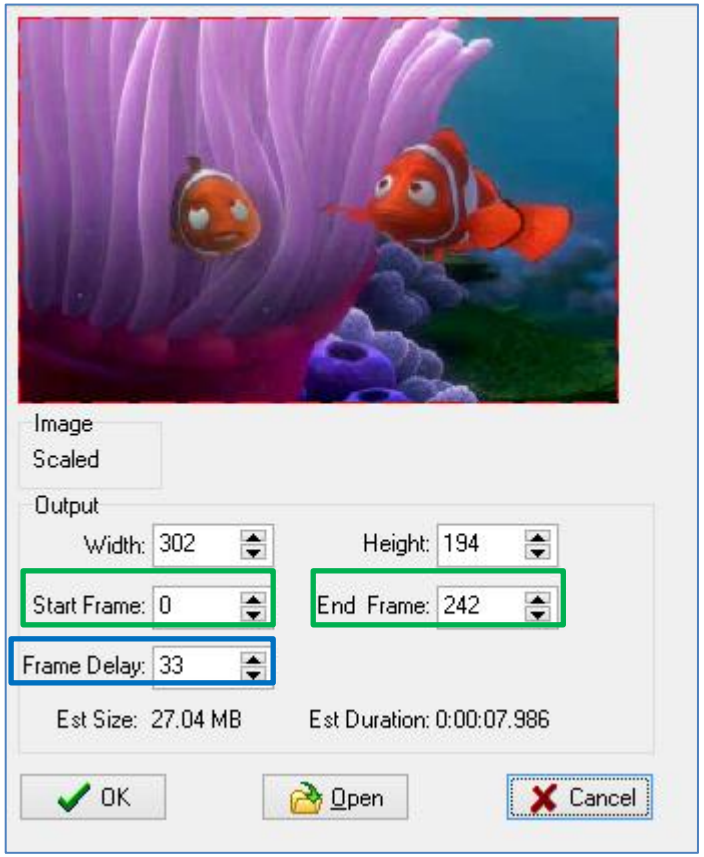



The left side shows information about the original input video; the right side shows information about the output video. Properties of the video object can also be edited here.

Below the input video, there is a slider for selecting a frame. Drag the slider horizontally and note how the frame shown and the frame number are updated accordingly.




**Form3** plays a sequence of the video file - frames 0 to 242 only. This is possible by editing the **Start and End Frame** values in the **Image + Video Converter** window (right side).

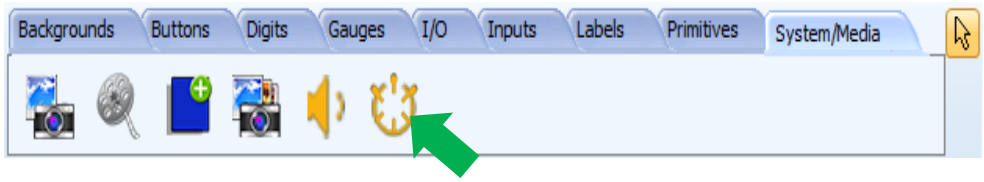



Frames 0 to 242 can be viewed using the frame slider. Also, note that the value for **Frame Delay** is **33**, the unit of which is in milliseconds. Frame delay defines how fast the frames are displayed. Click **OK** to close the **Image + Video Converter** window.

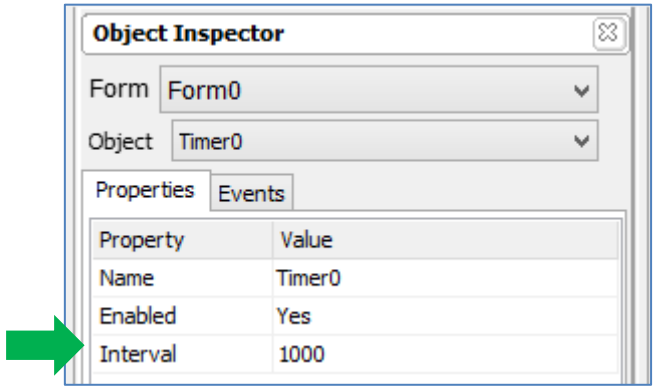
Refer to ViSi-Genie Play Video for a more detailed discussion of the parameters in the **Image + Converter** window and to know more about playing a video object in ViSi Genie in general.

### Add a Timer Object

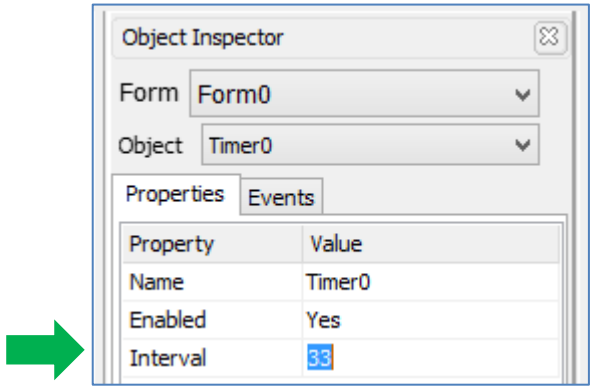
A **timer** object is needed to play the video. Go to the System/Media pane and select the **timer** icon.



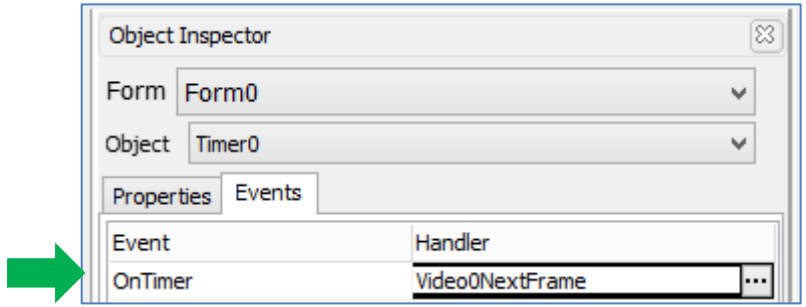

Note that the WYSISWYG screen displays **Form0**. The **timer** object is always under **Form0**, just like the **Sounds** object. The **timer** object raises an event at a given pace, for example every 1000 milliseconds.



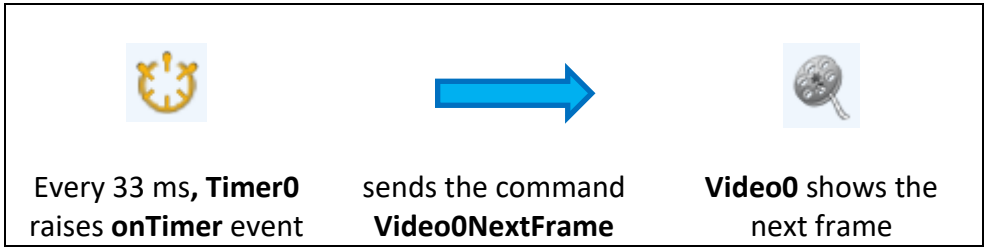

Now change the value of **Interval** from **1000** to **33** (milliseconds). Every 33 ms, **Timer0** raises the event called **OnTimer**.




The **OnTimer** event can be configured to send the command **Video0NextFrame** to **Video0**.

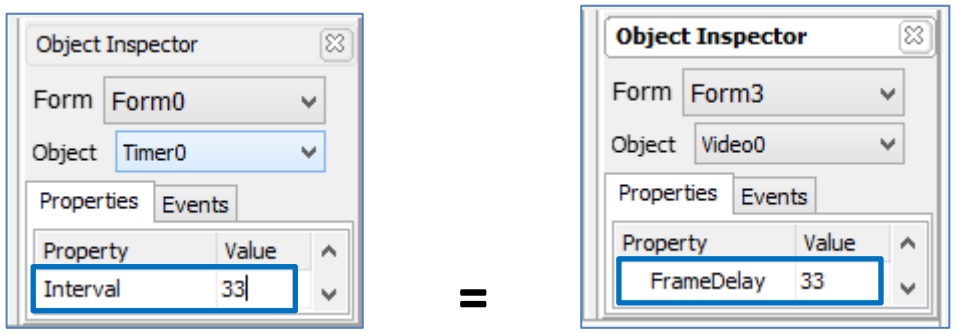



The command **VideoNextFrame** stands for *Tell the **Video0** object to show the next frame*.




Every 33 ms, a new frame will be displayed. As a video is a succession of still images or frames, changing the frames quickly enough creates the impression of a movie.

It is also important to note that the value of **Frame Delay** (Video0) is equal to the value of **Interval** (Timer0).

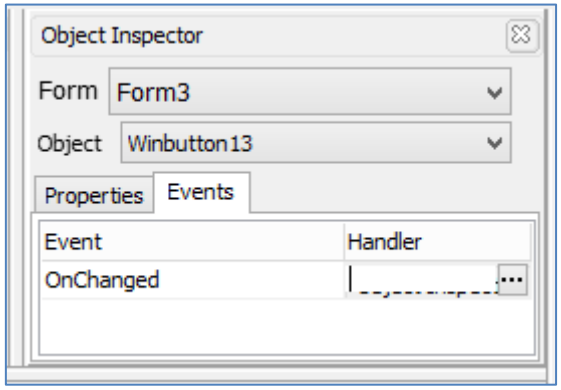



### Add Play, Pause, and Back Buttons to Control the Video Object

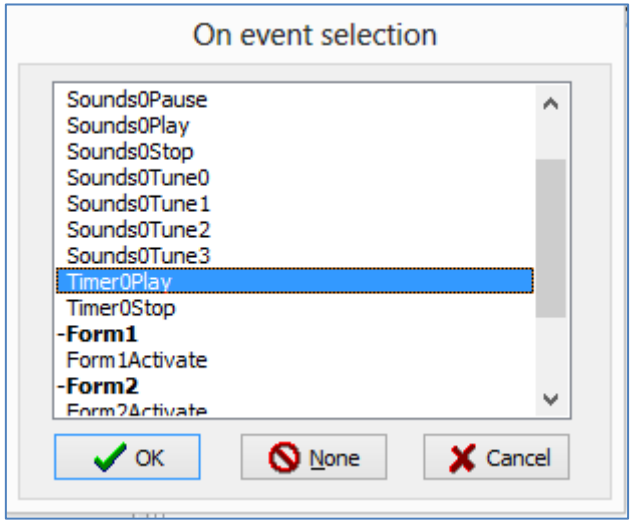
Now create the Play, Pause and Back buttons. These are similar to the Play, Pause, and Stop buttons previously created in the Music Player. Follow the procedure described in page 17 **(Create a Music Player - Add Customized Play, Pause, and Stop Buttons to Control the Sounds Object**) to create the buttons for the video player. The icon image file used for the Back button is found here: **…\StarterKitDemos.ImgData\.**

When finished, the screen will look similar to the one shown below.


Select the **Play** button, **Winbutton13** in this example. In the **Object Inspector**, go to the **Events** pane. Click on the ··· symbol to open the **On event selection** window.




Select Timer0Play then click OK.



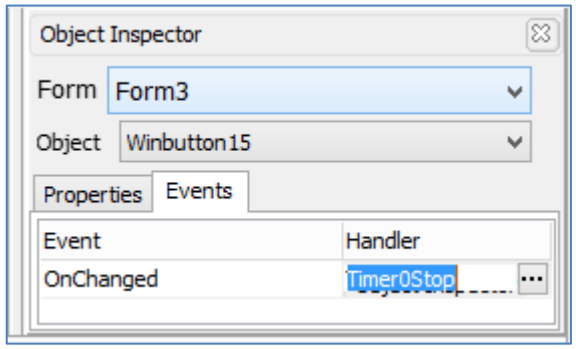

The command **Timer0Play** stands for *Tell the **Timer0** object to start the timer and raise an event every defined delay*.

The logic is that the play button controls the timer, which in turn controls the video. Therefore, the play button controls the video indirectly through the timer.

| | | | | |
|---|---|---|---|---|
| When pressed, **Play** raises **OnChanged** event | Sends the command **Timer0Play** | Every 33 ms, **Timer0** raises **onTimer** event | sends the command **Video0Next Frame** | **Video0** shows the next frame |

Again, the video frame delay must be equal to the timer interval.

Now configure the **Pause** button as shown below.



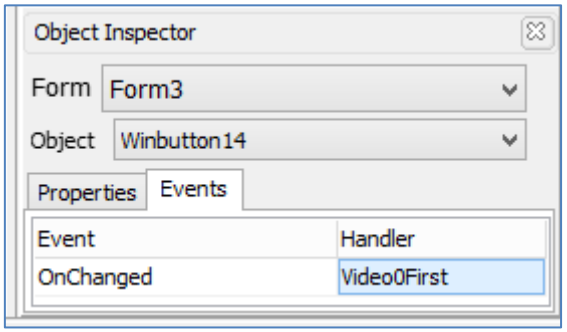

The command Timer0Stop stands for Tell the Timer0 object to stop the timer and no longer raise events.

Similarly, the logic is that the pause button controls the timer, which in turn controls the video. Therefore, the pause button controls the video indirectly through the timer.

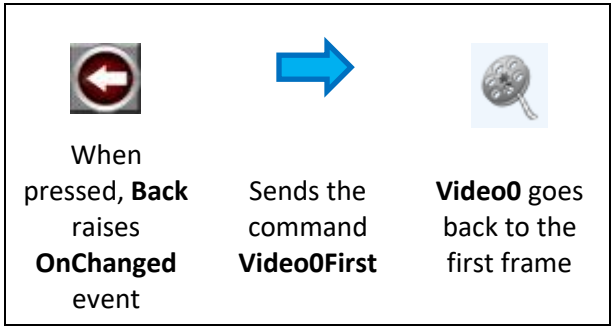
| | | | | |
|---|---|---|---|---|
| When pressed, **Pause** raises **OnChanged** event | Sends the command **Timer0Stop** | **Timer0** stops | Does nothing | **Video0** stops playing |

To resume after pause, press the **Play** button again.

The **Back** button is used here to show the first frame of the video object. Configure the Back button as shown below.



The command **Video0Previous** stands for *Tell the **Video0** object to show the first frame*. The **Back** button controls the **Video0** object directly in this case, without the use of the **Timer0** object.




### Create the 2nd and 3rd Forms for the Video Player

Follow the procedure for creating the first form of the video player (**Form3**) in creating the second and third forms, **Form4** and **Form5**. Three different video files can be played, using three different timer objects.

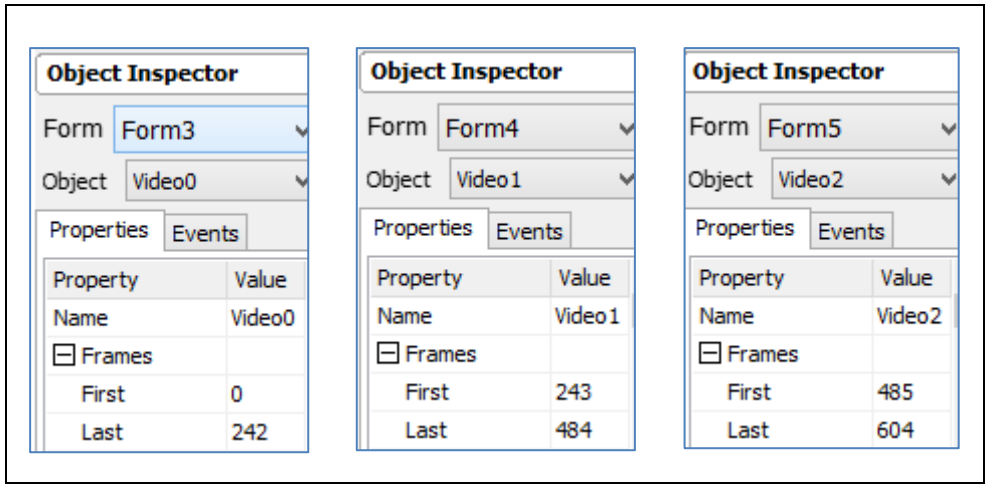
| Form | File | Video Object | Timer Object |
|---|---|---|---|
| **Form3** | File 1 | Video0 | Timer0 |
| **Form4** | File 2 | Video1 | Timer1 |
| **Form5** | File 3 | Video2 | Timer2 |

Each timer object corresponds to a video object, so be careful in configuring the control buttons in each form.

The video player in this application uses a single video file for all the three forms. The first form shows the beginning part of the video, the second form shows the middle part, and the third form shows the last part.

| Form | File | Video Object | Part | Frames | Timer Object |
|---|---|---|---|---|---|
| **Form3** | File 1 | Video0 | Beginning | 0 to 242 | Timer0 |
| **Form4** | File 1 | Video1 | Middle | 243 to 484 | Timer1 |
| **Form5** | File 1 | Video2 | End | 485 to 604 | Timer2 |

This is possible by configuring the **First** and **Last Frame** values of each video object in the Object Inspector.

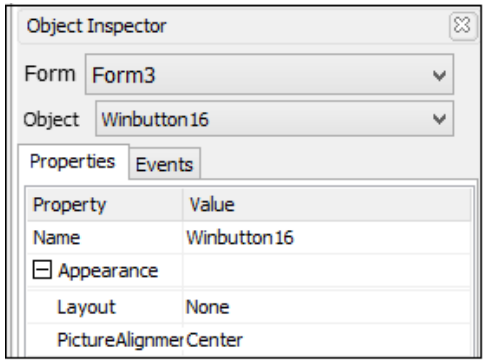


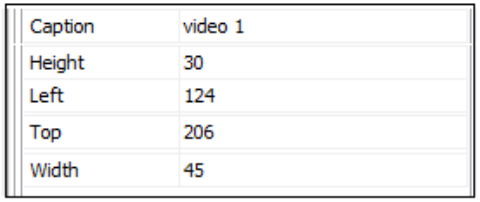
## Create the Navigation Buttons

After having created the three similar forms for the video player, we will now create the navigation buttons. See the final appearance of the video player below.


Pressing and releasing each of the form navigation buttons (video 1, video 2, and video 3) displays the corresponding form (Form3, Form4, and Form5). The home button is used to navigate back to the menu screen.

The form navigation buttons used in this application have the following properties.



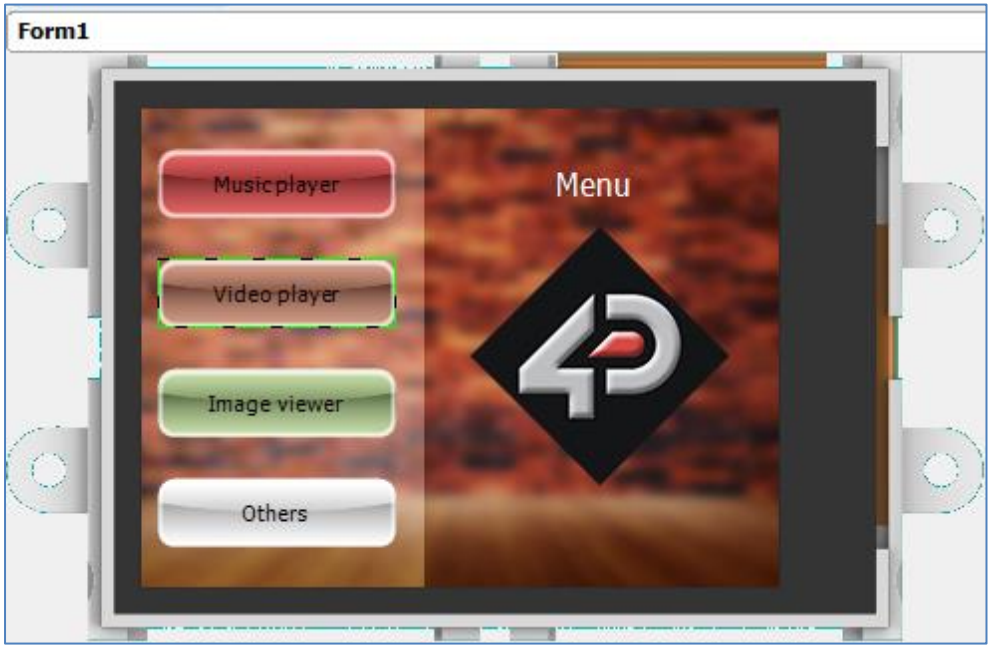



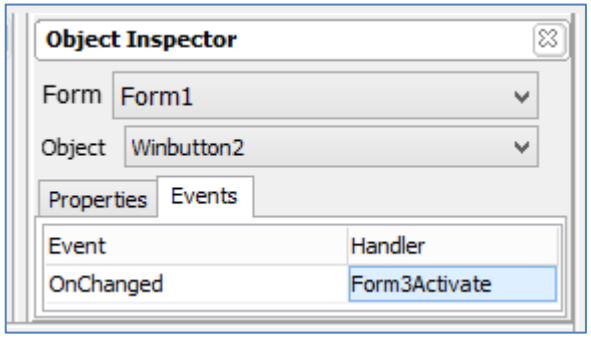

The image file used as icon for the video navigation buttons is found here: **…\StarterKitDemos.ImgData**.

Now create the two remaining video navigation buttons as well as the home button. The home button is identical to the music player home button. Do the same for the second and third forms. Linking of the navigation buttons to the corresponding forms is left as an exercise for the reader.

## Link the Video Player to the Menu Screen

The final step, before testing your program, is to link the first form of the video player (Form3 in this example) to the menu screen. Go to the menu screen and configure the video player button as shown below.
Refer to ViSi-Genie Play Video to know more about playing a video object in ViSi Genie.




Now you can test your program. Jump to page 65 to for the procedure. Check if the control and navigation buttons are working properly.

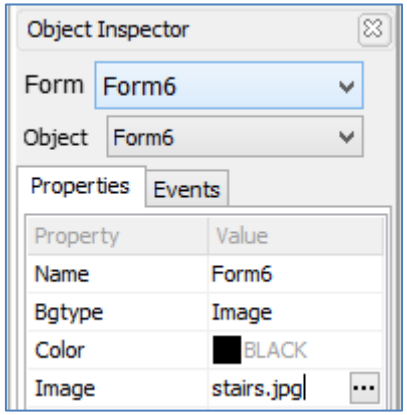
## Create an Image Viewer

The image viewer in this application is composed of four forms, each displaying an image and containing navigation buttons. Each page has next, back, and home buttons.

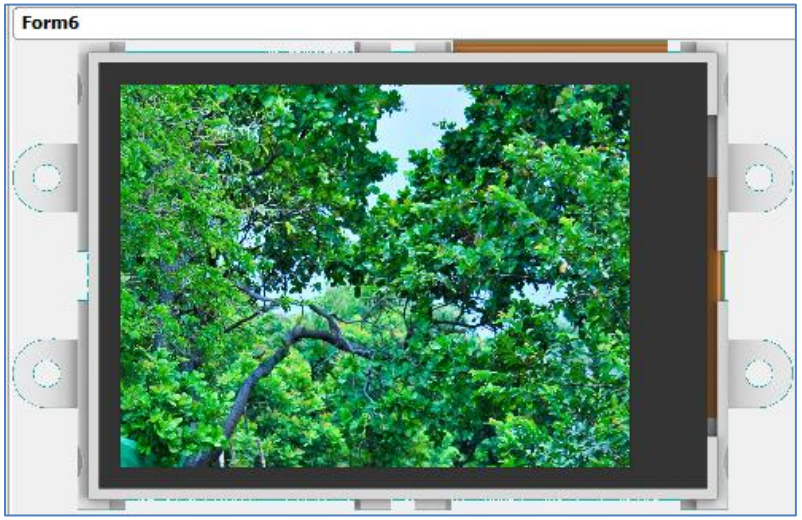
### Create the First Form

Add a new form to the image viewer.


Add a background image for the new form, Form6 in this example. This will be the first image shown. The image file used in this application is found here: **...\StarterKitDemos.ImgData\.**



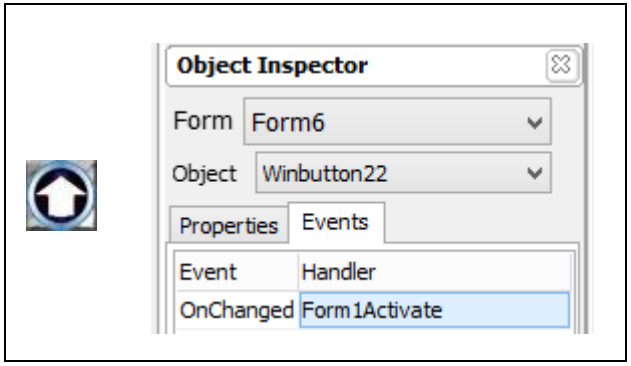

The WYSIWYG screen is updated accordingly.




## Create the Navigation Buttons

Now add the navigation buttons shown below.


The image files used as icons are found here: **…\StarterKitDemos.ImgData\.** Now link the Home button to the menu screen (Form1).



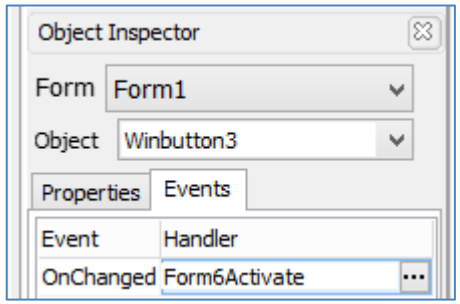

## Create the 2nd, 3rd, and 4th Forms

Follow the procedure for creating the first form to create the three remaining forms of the image viewer. Use any image file desired. The image files used as background images in this example are found here: **…\StarterKitDemos.ImgData\.** Also, do not forget to configure the navigation buttons in each form properly. The Next button will allow the user to navigate to the next form which shows another image. The Back button will allow the user to navigate back to the previous form.

## Link the Image Viewer to the Menu Screen

Go back to the Menu screen and select the Image Viewer button.


Configure it as shown below.



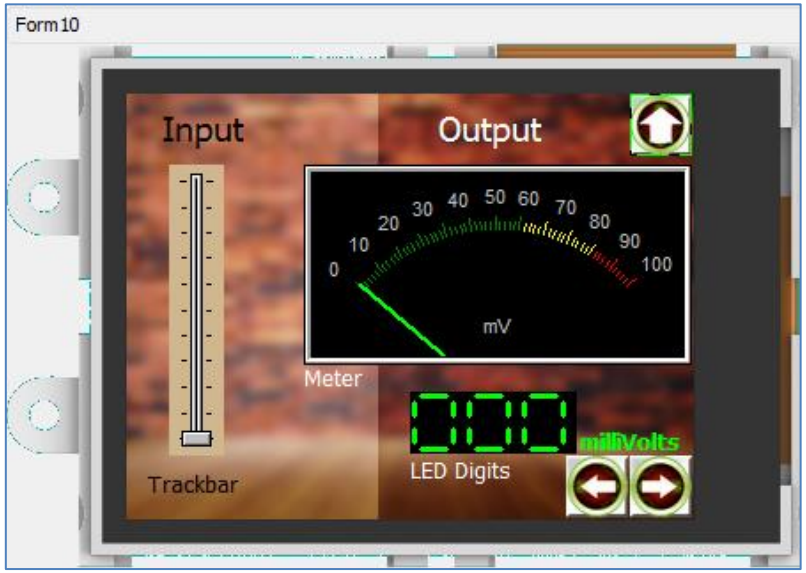

**Form6** here is the first form of the image player. When the Image Viewer button in the menu screen is pressed and released, the first form of the Image Viewer will be displayed.

To test your program, go to page 65. Check if the navigation buttons for the image viewer are working properly.

## Input and Output Objects: Trackbar – Meter and LED Digits

The three remaining forms in this application show the reader some of the different input and output objects used in ViSi Genie and the basic ways of using them. For a more detailed discussion of input, output, and combined objects, refer to ViSi-Genie-User-Guide.

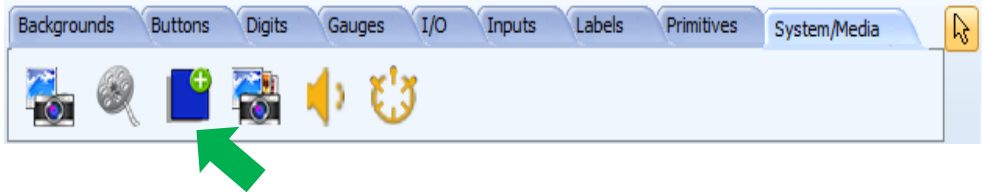
Shown below is the complete appearance of the Trackbar – Meter and LED Digits form (Form10 in this example).



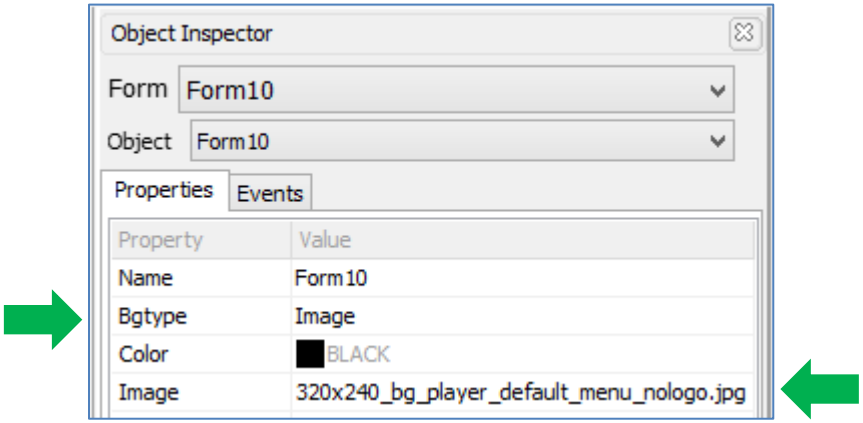

The user vertically drags the slider of the Trackbar object, which is the input. The needle of the Meter object points to the corresponding value in

response to the movement of the Trackbar slider in real time. The LED Digits object will then display the value when the user is done moving the Trackbar slider. The Meter and LED Digits objects are the output. The Next, Back, and Home buttons are added for navigational purposes.

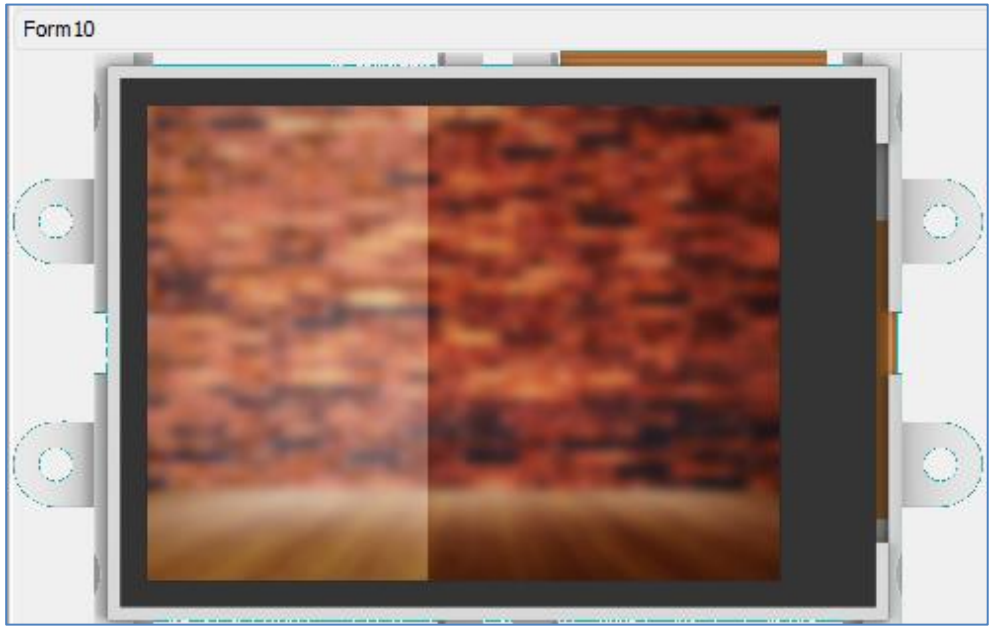
## Create a New Form with a Background Image




A new form will be displayed. Add a background image to the form. The 4D input - output background image is located here:
**...\StarterKitDemos.ImgData\.**

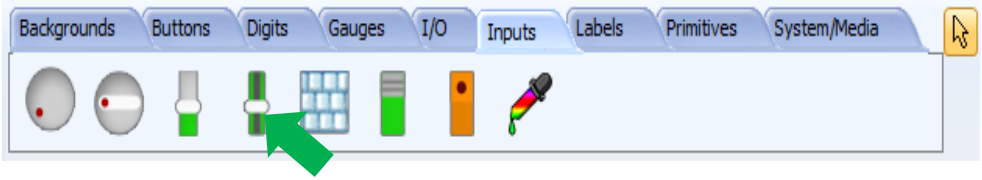



The WYSIWYG screen is updated accordingly.

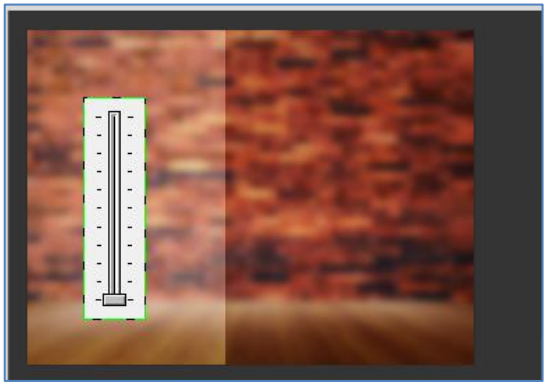



## Add a Trackbar Object

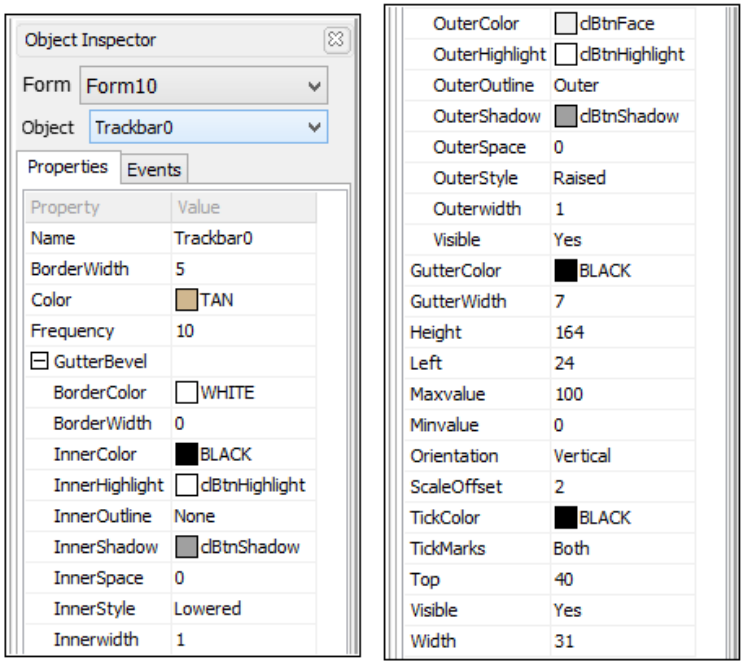
Go to the Inputs pane and click on the Trackbar icon.



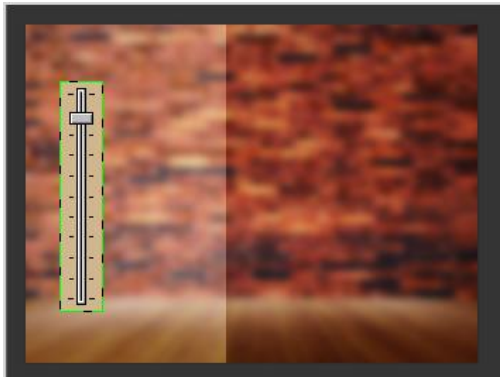

Click on the WYSIWYG screen to place it.



The object can be dragged and resized as desired. Feel free to experiment with the properties in the Object Inspector. The Trackbar object in this application has the following properties.

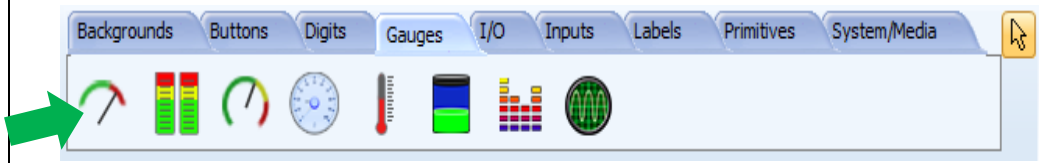



The customised Trackbar object (Trackbar0) is updated in the WYSIWYG screen.

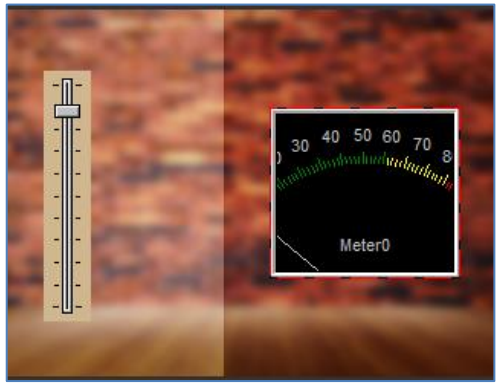


## Create a Meter Object

Go to the Gauges pane and select the Meter icon.

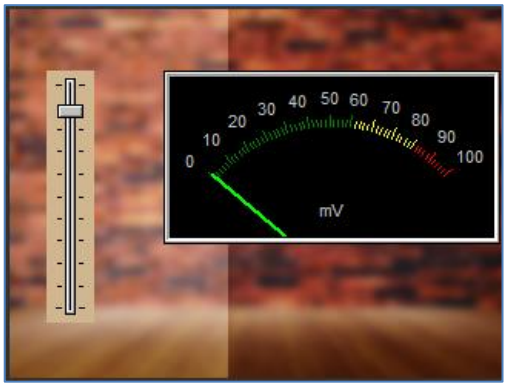



Click on the WYSIWYG screen to place it.



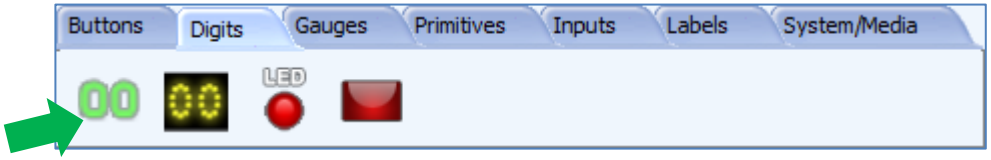

The object can be dragged and resized as desired. Feel free to experiment with the properties in the Object Inspector. The Meter object in this application has the following properties.

The customised Trackbar object (Trackbar0) is updated.

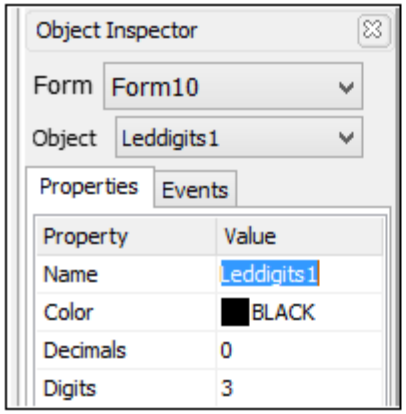


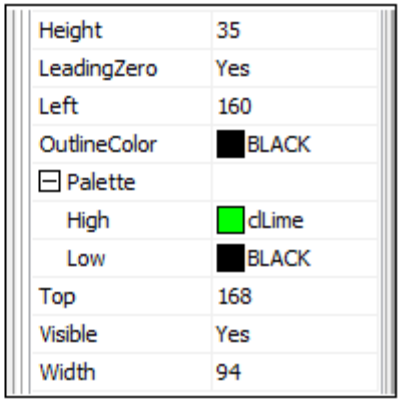

## Create a LED Digits Object

Go to the Digits pane and click on the LED Digits icon.

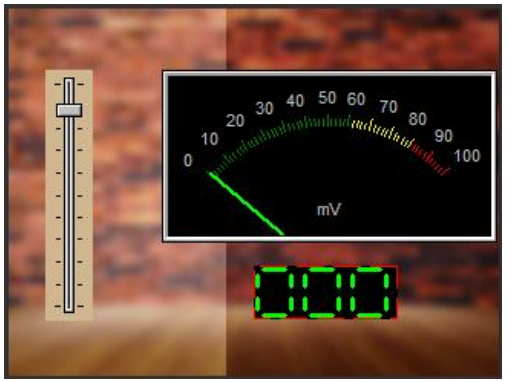



Click on the WYSIWYG screen to place it. The LED Digits object in this example has the following properties.



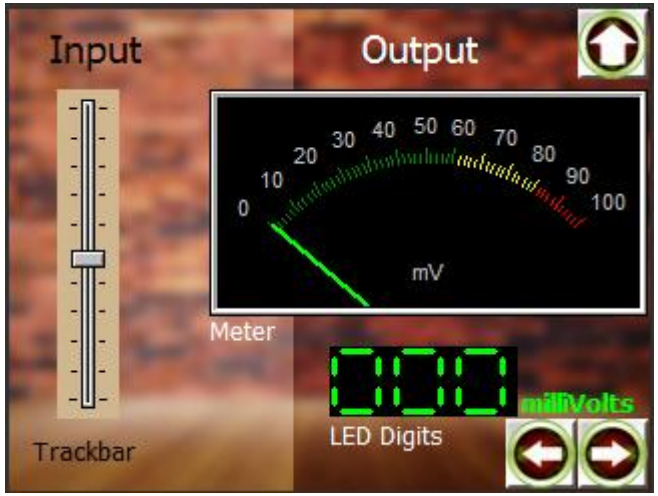




The WYSIWYG screen is updated.

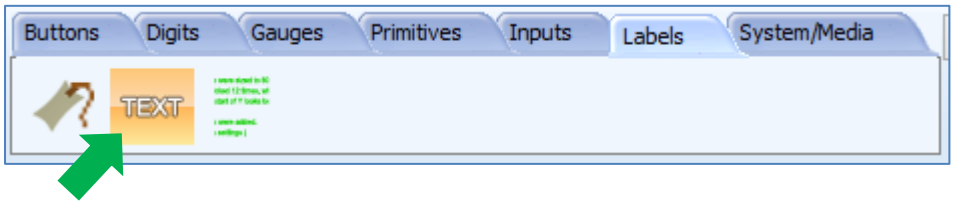


## Add Static Text Objects

Text objects will be added to label the different parts of the form, as shown below.




We will now add the text "Input".



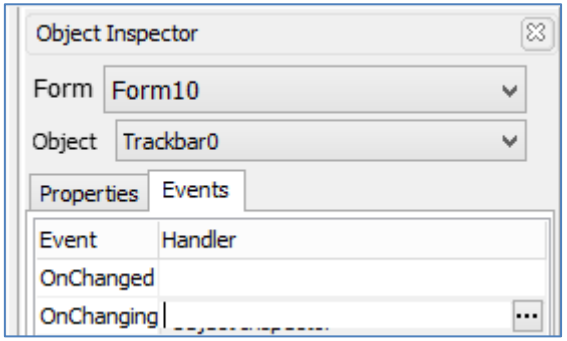

Click on the WYSIWYG screen to place a text object.

The object Statictext3 now appears on the screen. It can be dragged to any desired location. The Object Inspector displays the properties of the object. Change the caption from "**Statictext3**" to "**Input**". Edit the other properties as desired. Do the same for all the remaining text labels.

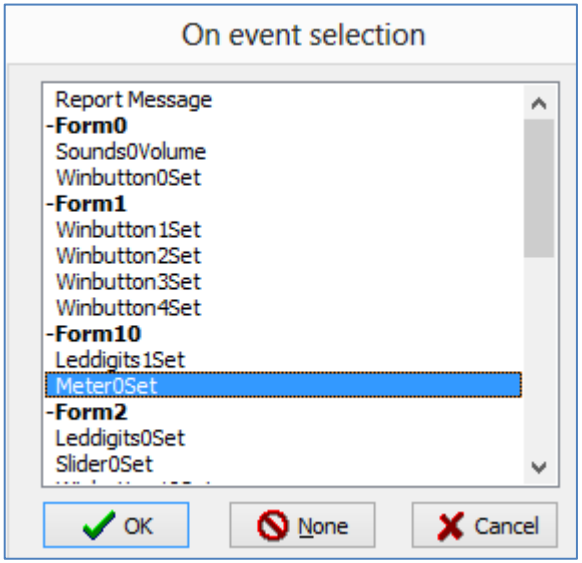
## Link the Input and Output Objects

Now, the objects need to be linked – moving the trackbar updates the meter and the LED digit display.

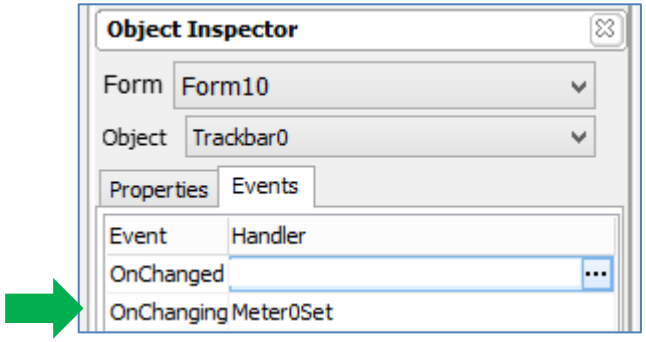
Select Trackbar0 and go to the Events pane of the Object inspector. Click on the ··· symbol in the OnChanging line to open the On selection window.



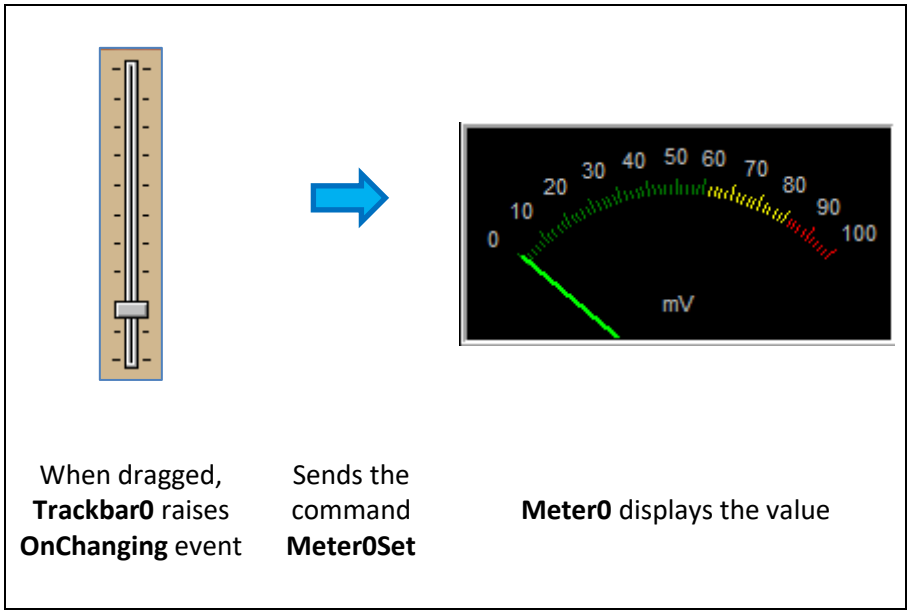

Select Meter0Set and click OK.



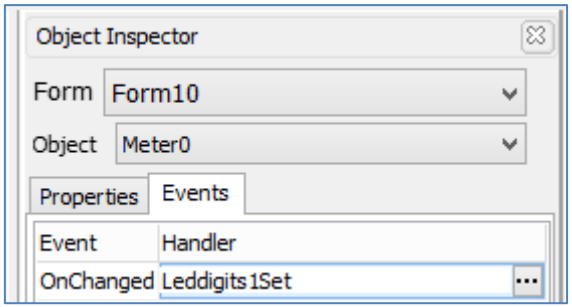

Note that Meter0Set is under Form10. When working with multiple forms and objects, things may get mixed up.
The Events pane for Trackbar0 is now updated accordingly.



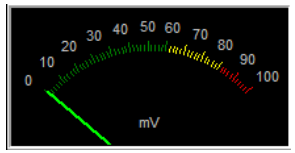

Moving Trackbar0 raises the **OnChanging** event. When the **OnChanging** event arises, a message is sent to **Meter0** with the value.




Select **Meter0** and configure it as shown below.



When **Meter0** receives and displays a new value, it raises the **OnChanged** event. When the **OnChanged** event arises, a message is sent to **Leddigits1** with the value.

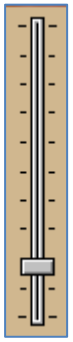


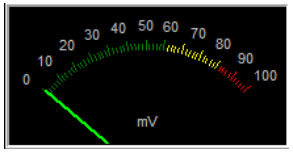

When **Meter0** receives a new value, it raises the **OnChanged** event

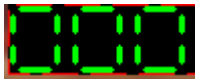
Sends the command **Leddigits1set**

**Leddigits1** displays the value

In summary:





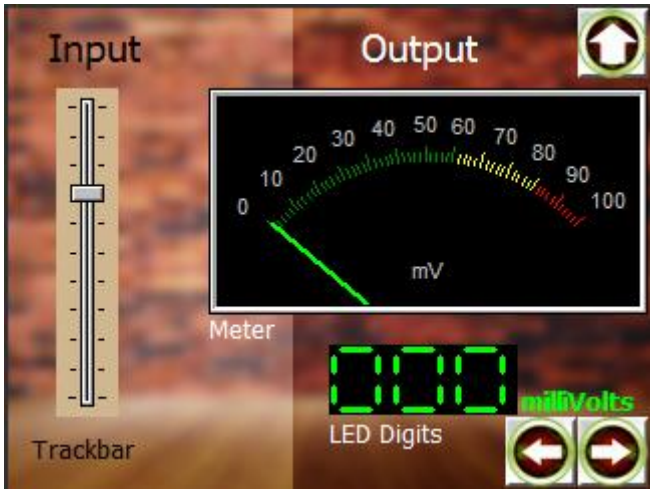
**OnChanging**

**OnChanged**

For more information on OnChanged and OnChanging events, refer to [ViSi-Genie onChanging and onChanged Events].

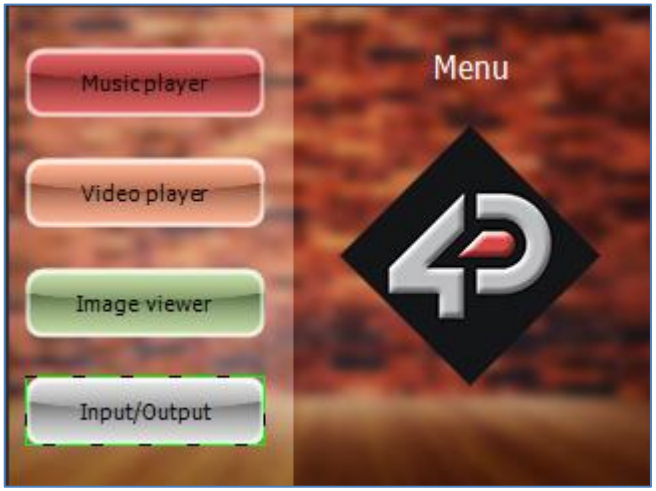
### Add the Navigation Buttons

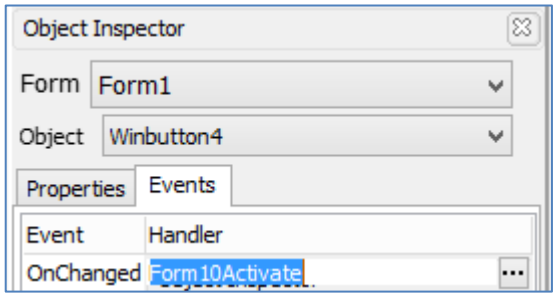
The navigation buttons are similar to those created in the image viewer. Creating and configuring these buttons are left as an exercise for the reader. The image files used as button icons in this form are found here: **…\StarterKitDemos.ImgData\.** When finished the final form should look similar to the one shown below.




### Link the Trackbar – Meter and LED Digits Form to the Menu Screen

Before testing your program, go to the menu screen and configure the Input/Output button as shown below.



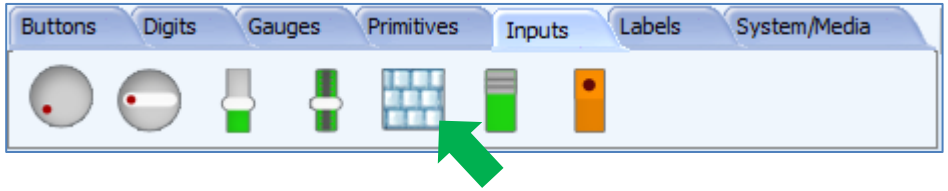

Go to page 65 for the procedure on how to build and upload your project.

## Input and Output Objects: Keyboard – External Host Processor

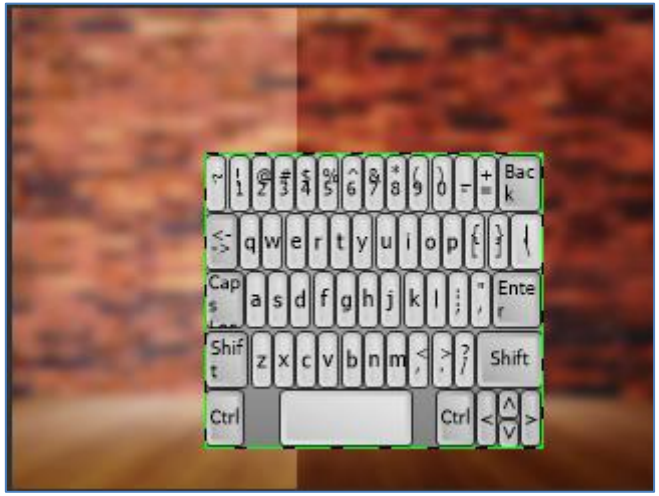
This form shows the possibility of using a customized keyboard as an input and an external host processor as an output.

### Add a Customized Keyboard Object

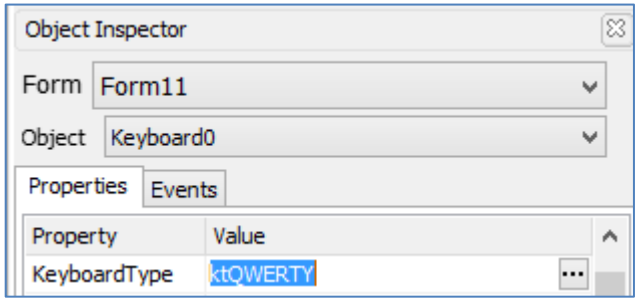
After having created a new form with a background image, go to the Inputs pane and select the keyboard icon.



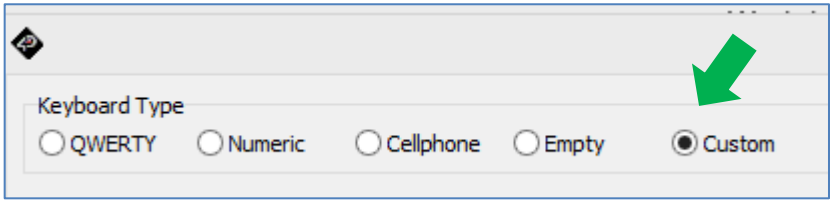

Click on the WYSIWYG screen to place it.



The object can be dragged and resized. The properties can be edited in the Object Inspector. Now click on the ··· symbol in the KeyboardType line of the Object Inspector.

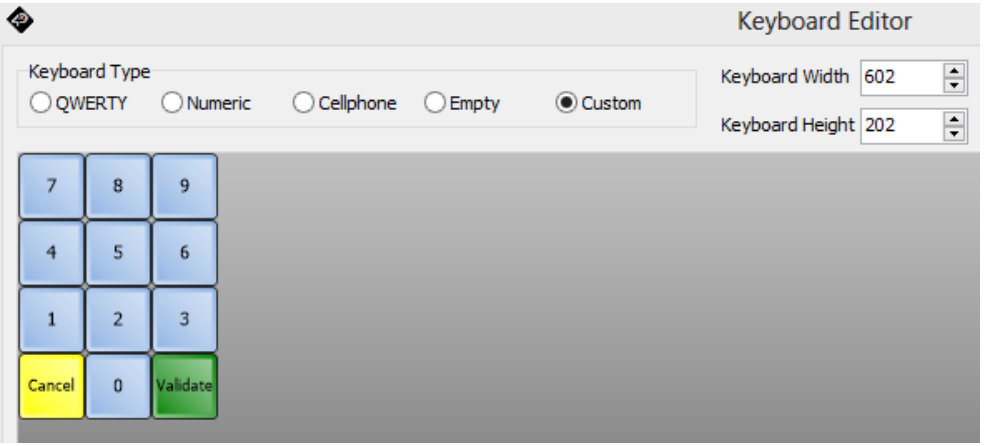



The Keyboard Editor window appears. Select Custom for the Keyboard Type.




Then click on the **Load** button on the lower left part of the window.


A standard Open window file appears and asks for a keyboard layout file. The keyboard layout file used in this application is found here: **…\StarterKitDemos.ImgData\.** After selecting the file, click open. A customized keyboard now appears.




Change the Keyboard Width and Height to 121 and 161, respectively. Click OK. The WYSIWYG screen is now updated.

For a detailed discussion on customizing keyboards, refer to ViSi-Genie Customised Keyboard.

### Add Image Objects

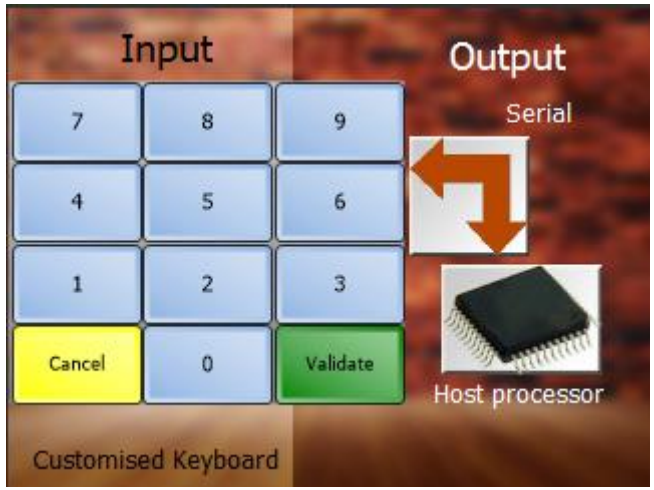
To add an image object, go to the System/Media pane and click on the image icon.


Click on the WYSIWYG screen to place the object. A standard Open window appears. The image files used in this example are found here: **...\StarterKitDemos.ImgData\.** The image objects can be resized and dragged. The properties can be edited in the Object Inspector. The WYSIWYG screen is now updated.


Refer to ViSi-Genie Show Image for more information on adding image objects in ViSi Genie.

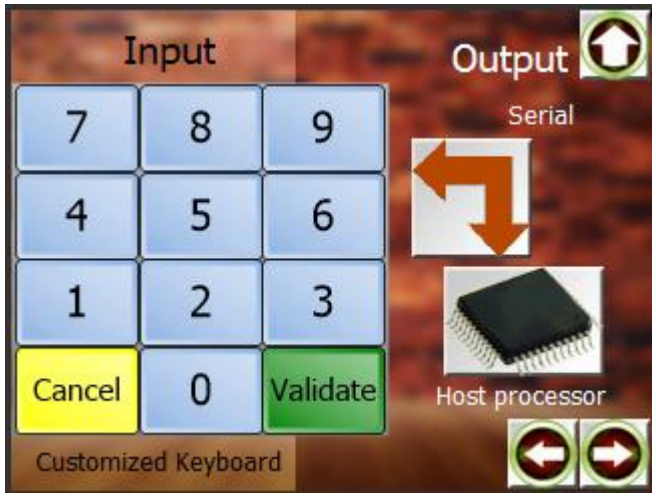
### Add Static Text Objects

The text objects are used to label the different parts of the form. Follow the procedure described in the previous forms to complete the labels as shown below.




### Add the Navigation Buttons

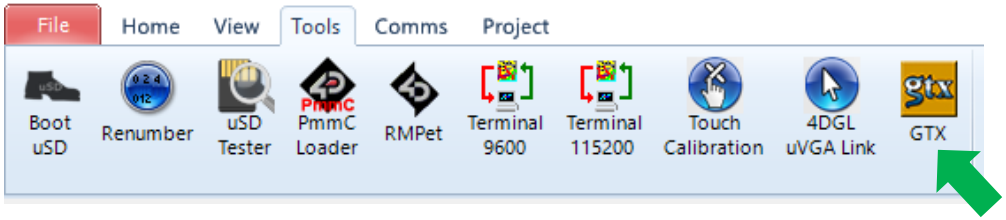
The navigation buttons are similar to those created in the image viewer. Creating and configuring these buttons are left as an exercise for the reader. The image files used as button icons in this form are found here: **...\StarterKitDemos.ImgData\.** When finished the final form should look similar to the one shown below.



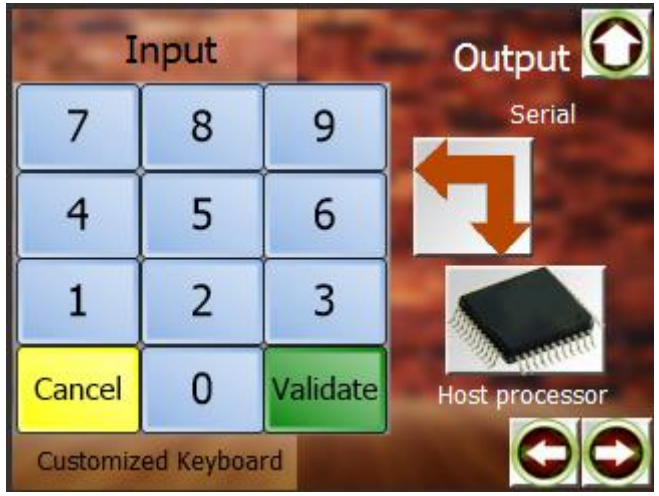

Build and upload the program. Go to page 65 for the procedure.

### Serial Data

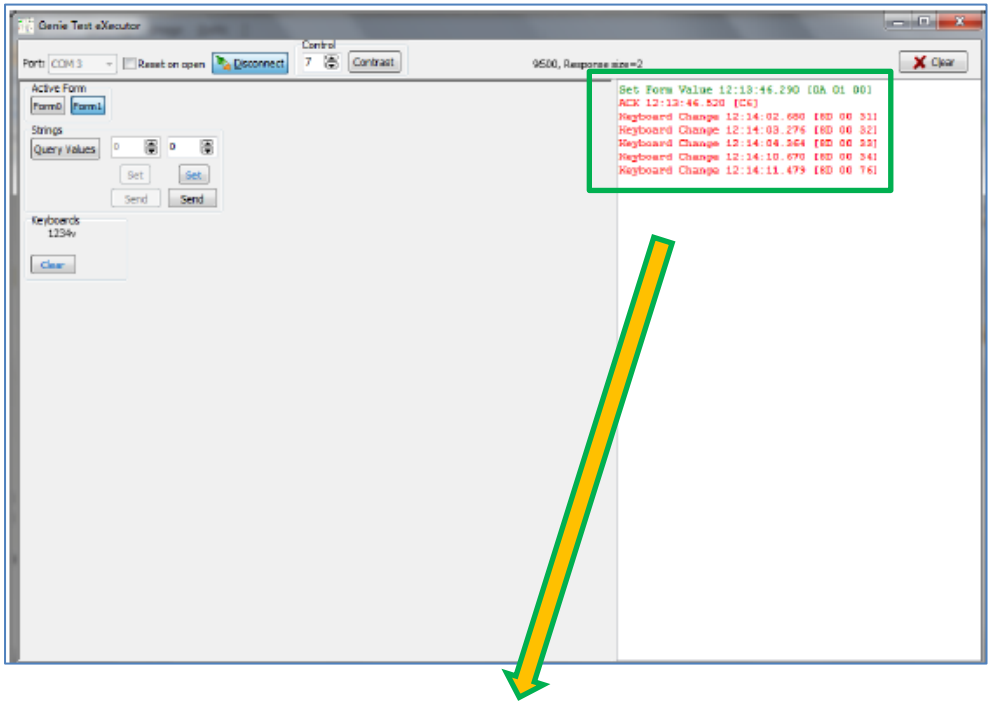
To view the serial data from the keyboard, we will use the GTX (Genie Test Executor) tool. Eject the uSD card from the PC and plug it in to the uSD slot of the display module. Select the Tools menu and click on the GTX button.




A new window appears and displays the forms and objects created. Click on Form11 to activate the customised keyboard form. The module will now display the keyboard form.




Press and release any of the keyboard buttons and notice the corresponding outputs on the right side of the GTX window.



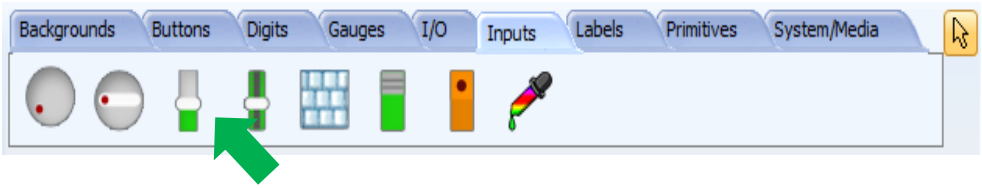

Set Form Value 12:30:58.060 [01 0A 01 00 00 0A]
ACK 12:30:58.278 [06]
Keyboard Change 12:31:05.860 [07 0D 00 00 31 3B]
Keyboard Change 12:31:06.796 [07 0D 00 00 32 38]
Keyboard Change 12:31:07.545 [07 0D 00 00 33 39]
Keyboard Change 12:31:08.605 [07 0D 00 00 34 3E]
Keyboard Change 12:31:09.448 [07 0D 00 00 76 7C]

The message in green is from the PC to the display module. The messages in red come from the display module and go to the PC. Here the display module is sending report event messages to the PC. The PC can be replaced with any host controller. To understand more about the ViSi Genie communications protocol, refer to ViSi-Genie-Reference-Manual.

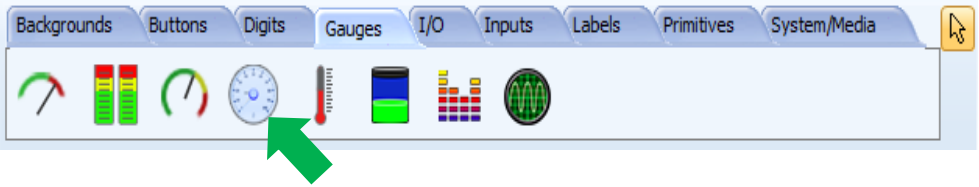
## Input and Output Objects: Slider – Cool Gauge and LED Digits

### Similar Forms


This form is very much similar to the **Trackbar – Meter and LED Digits** form (**Form10**). The **trackbar** is replaced with a horizontal **slider**, and the **meter** is replaced with **a cool gauge**. The **slider** icon is found in the **Inputs** pane, beside the **trackbar** icon.



The **cool gauge** is located in the **Gauges** pane.




The procedure for creating **Form10 (Input and Output Objects: Trackbar – Meter and LED Digits)** can be used to create this form. Don't forget to configure the navigation buttons properly and link this form to the **Next** button of **Form11 (Input and Output Objects: Keyboard – External Host Processor).**

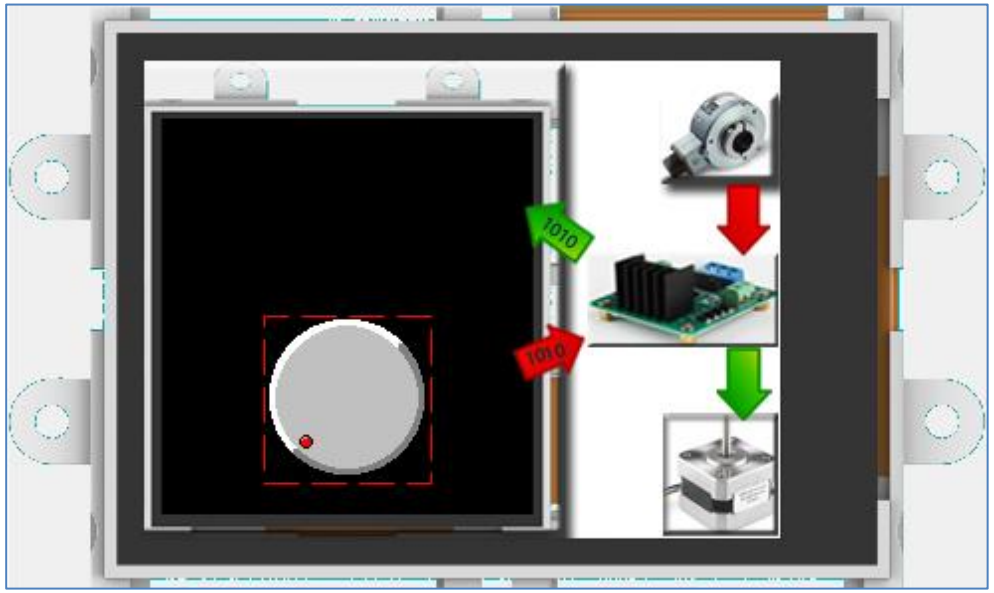
## Input and Output Objects: Knob – LED Digits

This form shows how to use a knob as an input and a LED digits object as an output. The background image also sugggests the possibility of using the Picaso/diablo16 display module in controlling and monitoring an external system through serial communication. The knob, when moved, is configured to send data to an external host – a microcontroller-driver circuit. The MCU-driver unit controls the rotation of a motor. The rotary encoder, coupled to the shaft of the motor in a manner which depends on the application, returns data to the MCU. This data is used to interpret the direction of rotation and angle of the motor shaft with respect to a reference.

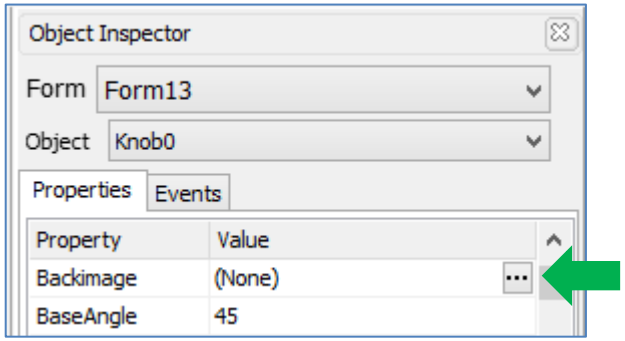
### Add a Knob Object

Add a new form with a background image. The image file used in this example can be found here: **…\StarterKitDemos.ImgData**. To add a knob, go to the Inputs pane and click on the knob button.

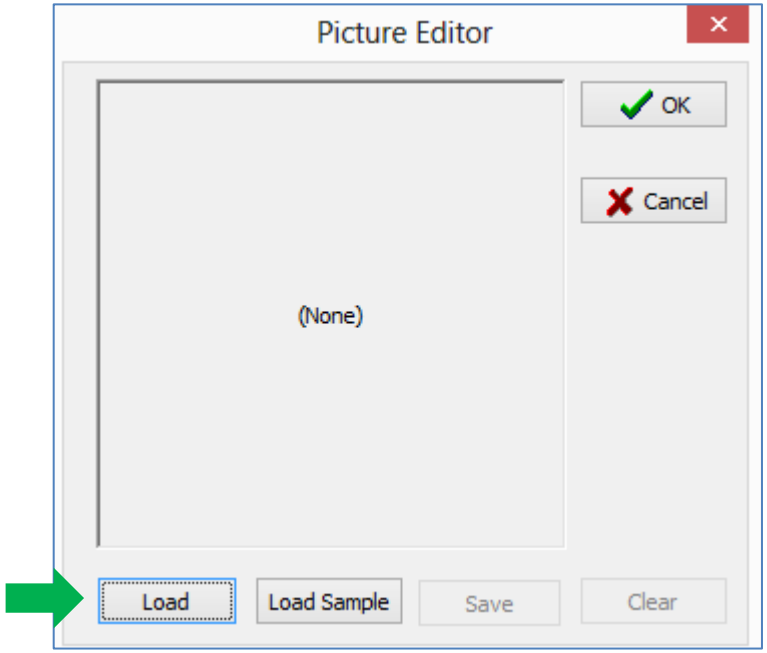

Click on the WYSIWYG screen to place it. The figure below shows the form with a background image and a knob.



The Object Inspector shows the different properties of the knob. Now we will add a dial scale. Click on the ... symbol of the **Backimage** property.

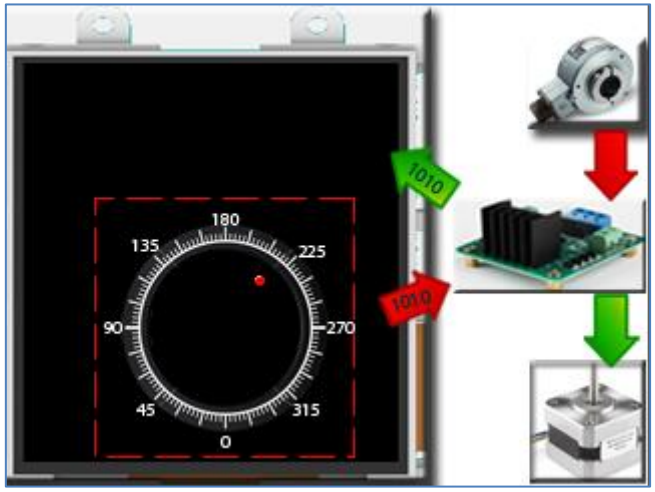



A Picture Editor window appears. Click on **Load**, **browse** for the desired image file, and click **OK**.

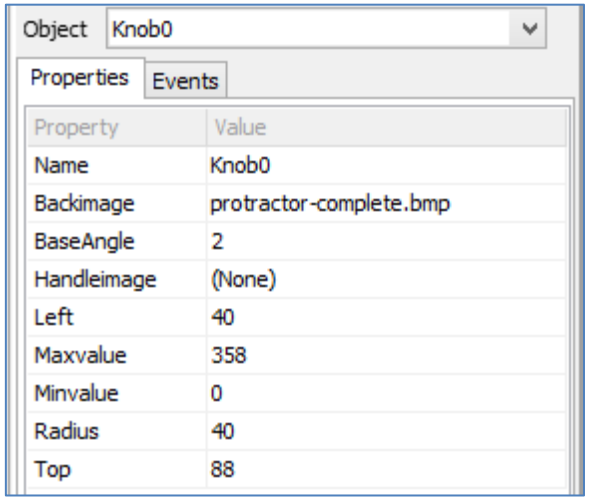



The image file used in this example is found here: **...\StarterKitDemos.ImgData**.

The WYSIWYG screen is updated.




Now apply the following additional properties to the knob.

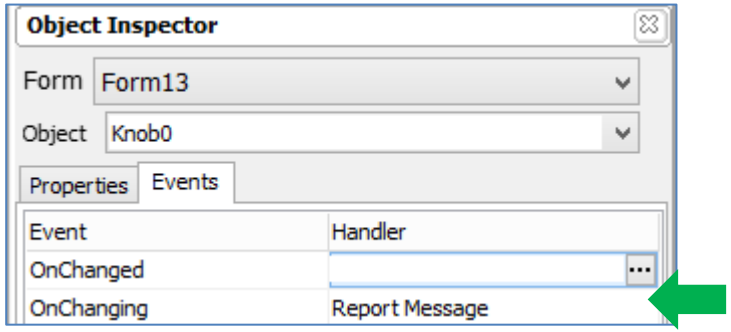


### Add a LED Digits, Static Text, and Navigation Button Objects

Follow the procedures described previously in adding a LED digits and static text objects. Do the same for the navigation buttons. The static text objects are used to label the different parts of the form. Also, don't forget to configure the navigation buttons properly and to link this form to the next button of Form12. The complete form is shown below.

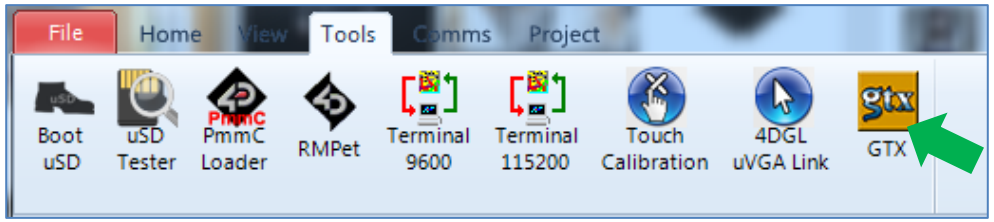

### Serial Data from the Knob and to the LED Digits Object

We will now read and write data from the knob and to the LED digits object with the use of the GTX (Genie Text Executor) tool. The host controller is the PC in this case. In other applications, the PC can be replaced with a microcontroller. Configure the knob as shown below.

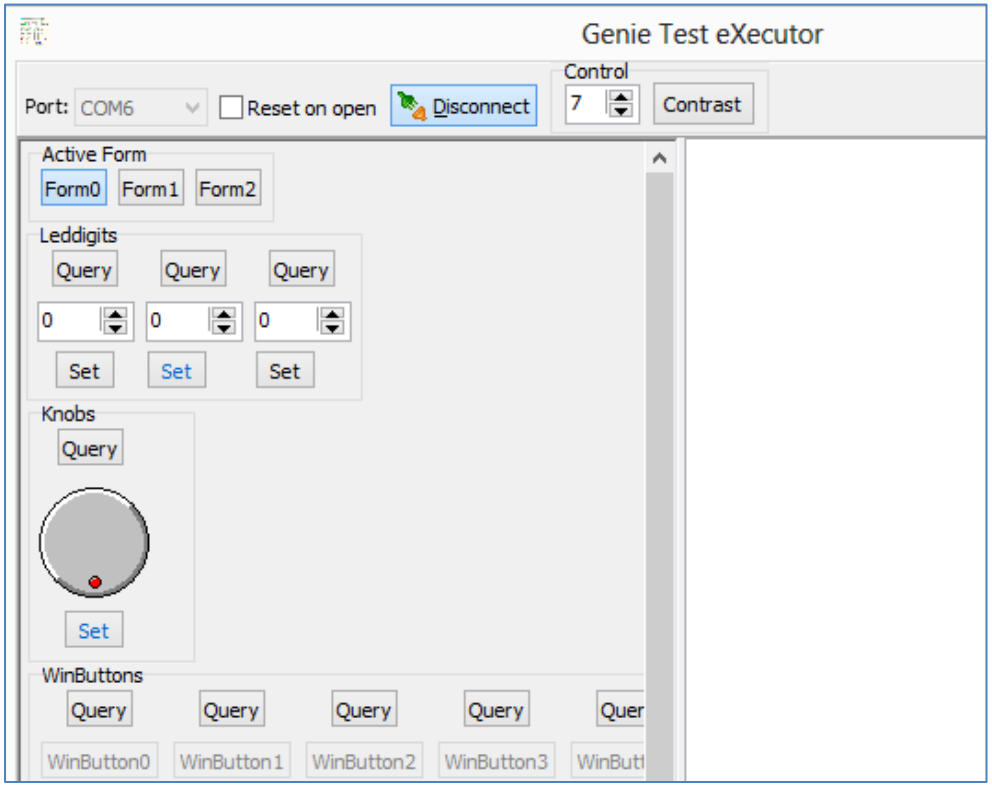



When Knob0 is turned, it raises the OnChanging event. When the OnChanging event arises, a message is sent to the PC.

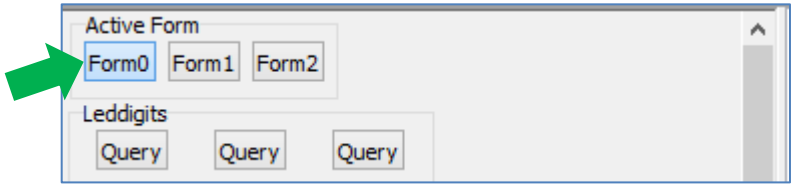
To open the GTX tool, go the Tools menu, then click on the GTX tool.



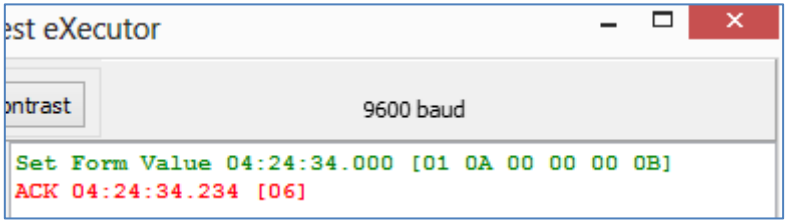

A new window now appears showing the different objects of the application.




There are three forms in this specific window, Form0 being the one we are working on. The program is reduced to three forms instead of thirteen at this point only, to simplify the tutorial. Now click on the Form0 button to display the first form.




Notice that on the right part of the screen, you can see the messages sent to and received from the display module.



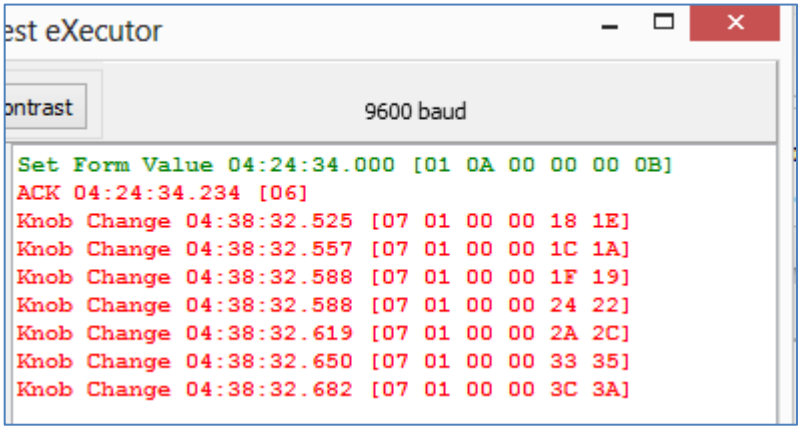

The message in green is from the PC to the display module; the message in red is from the display module to the PC. All values are in hexadecimal. The message in green is formatted according to the following pattern:

| Command | Object Type | Object Index | Value MSB | Value LSB | Checksum |
|---|---|---|---|---|---|
| **01** | 0A | 00 | 00 | 00 | 0B |
| **WRITE_OBJECT** | Form | Number 1 | | | |

To display Form1 or the second form (if a program has multiple forms), the value of the object index is changed to 0x01; for the third form, 0x02, and so on.

The message in red is an acknowledgment from the display module.

Now move the knob handle on the display module. Notice the new set of messages on the white area of the GTX window.

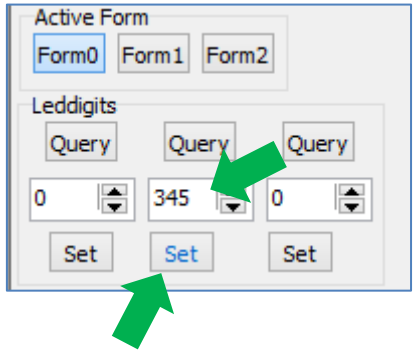



To interpret the first of the new set of messages:

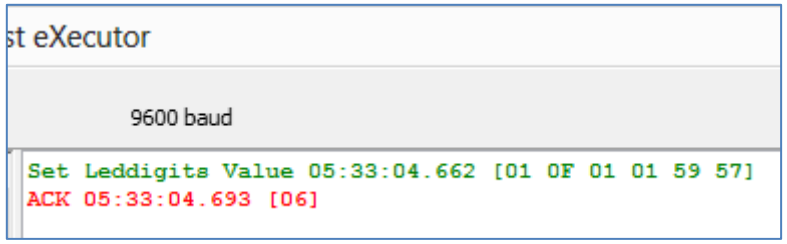
| Command | Object Type | Object Index | Value MSB | Value LSB | Checksum |
|---|---|---|---|---|---|
| **07** | 01 | 00 | 00 | 18 | 1E |
| **REPORT_EVENT** | Knob | Number 1 | 0x0018 or 24 decimal | | |

The minimum and maximum values for the knob are 0 and 358 (decimal), or **0x0000** and **0x0166**. The messages are sent when the knob is being moved because **Message** has been defined for the event **OnChanging**, which is raised as many times as long as the slider is being moved.

Now input the value "**345**" for the middle box under the Leddigits label and click on the set button.




Again, the program has been simplified at this point to have only three LED digits objects. The boxes from left to right correspond to Leddigits0, Leddigits1, and Leddigits2, respectively. We are interested in Leddigits1 (the middle box). To check which LED digits object you should set, close the GTX window, and check the name of the LED digits object controlled by Knob0, or whichever knob object you are working with.
After having clicked the set button, notice that the LED digits object on the display module now shows the value "345".
The white area also displays a new set of messages.

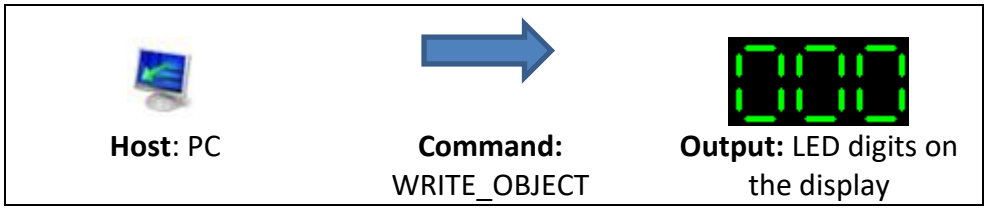



To interpret:

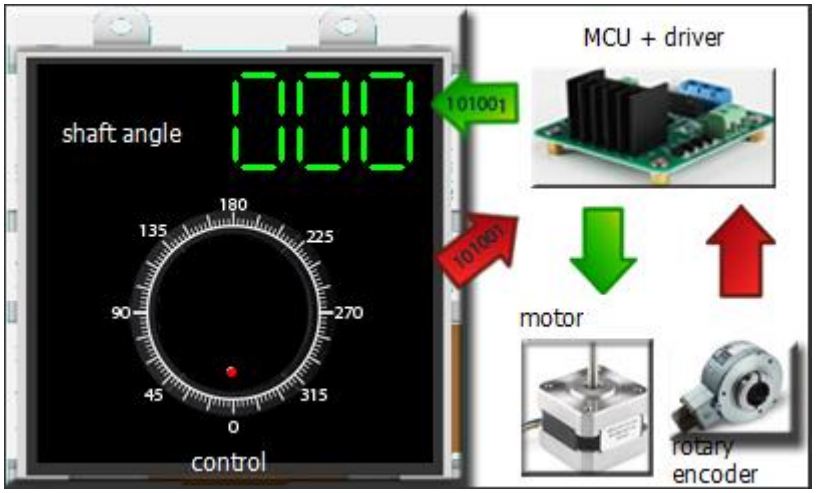
| Command | Object Type | Object Index | Value MSB | Value LSB | Checksum |
|---|---|---|---|---|---|
| **01** | 0F | 01 | 01 | 59 | 57 |
| **WRITE_OBJECT** | LED digits | Number 2 | 0x0159 or 345 in decimal | | |

Again, the message in red is an acknowledgement from the display module.

In summary:


Now we replace the PC with a microcontroller with a motor-encoder system.




The format for the messages transmitted between the Picaso/diablo16 display module and the host in this appication note is defined in the Genie Standard Protocol. For further references, refer to the following documents:

[ViSi Genie Reference Manual]

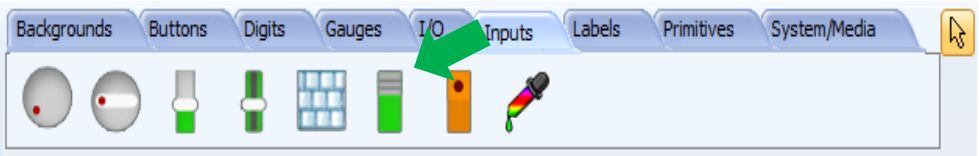
[ViSi-Genie Connection to a Host with Red Green Blue Control]

### Input and Output Objects: DIP Switch – User LED and LED Digits

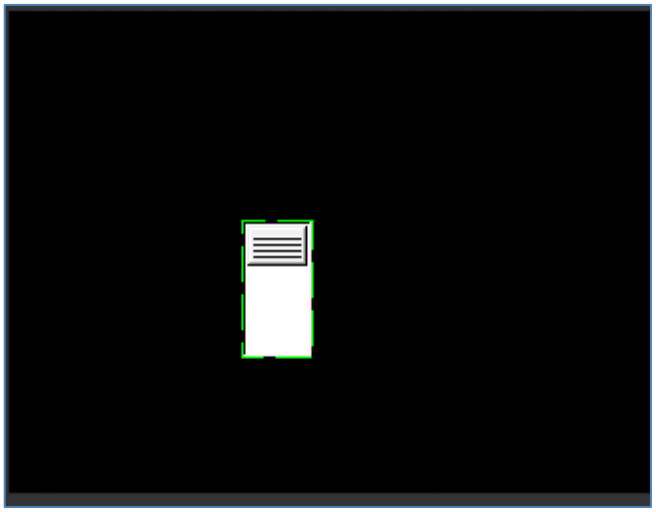
This form shows the basic use of DIP switches and user LEDs.

#### Add a DIP Switch Object

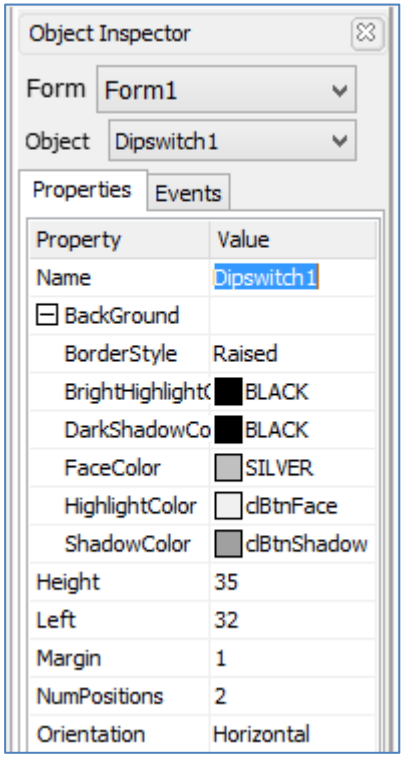
Create a new form. To add a DIP switch, go to the inputs pane and click on the DIP switch icon.

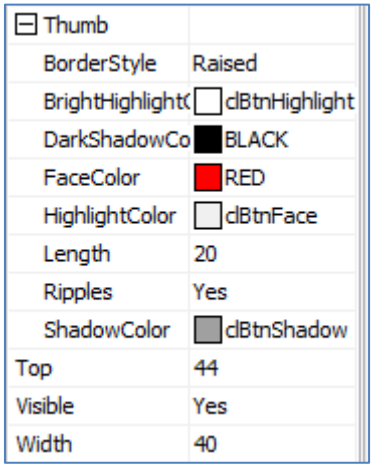



Click on the WYSIWYG screen to place it.



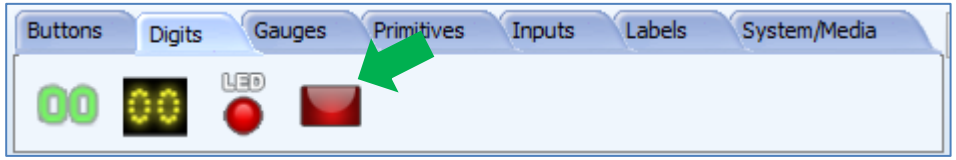
The object can be dragged and resized. The properties can be edited in the Object Inspector. Apply the following properties to the DIP switch:




## Add a User LED Object

To add a user LED, go to the Digits pane and select the user LED icon.




Click on the WYSIWYG screen to place it.

The object can be dragged and resized. The properties can be edited in the Object Inspector. Apply the following properties to the user LED:

Object Inspector

Form: Form14

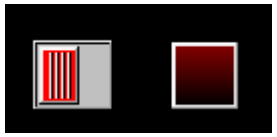
Object: Userled0

Properties | Events

| Property | Value |
|---|---|
| Name | Userled0 |
| ⊞ Bevel | \| |
| Height | 22 |
| Left | 225 |
| OutlineColor | BLACK |
| OutlineWidth | 0 |

| ⊟ PaletteEx | |
|---|---|
| High1 | clRed |
| High2 | BLACK |
| Low1 | 0x000051 |
| Low2 | BLACK |
| Top | 122 |
| Visible | Yes |
| Width | 32 |

When done, the user LED, together with the DIP switch, should look as shown below:



## Link the User LED to the DIP Switch

To make the DIP switch control the user LED, configure it as shown below.

Object Inspector

Form: Form14

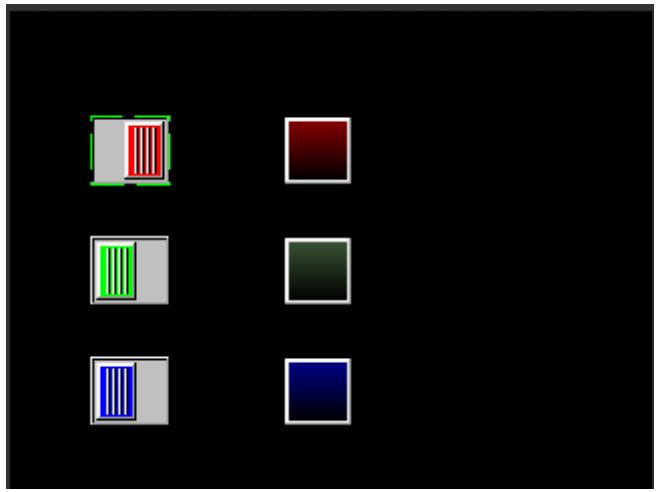
Object: Dipswitch1

Properties | Events

| Event | Handler |
|---|---|
| OnChanged | Userled0Set |
| OnChanging | |

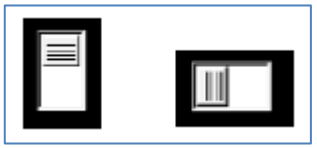
## Create Additional Pairs of User LED and DIP Switch

Now create two more sets with the colors green and blue. Also configure the DIP switches to control the corresponding user LEDs. When finished, the form should look as shown below.



To test the program, go to page 65 for instructions. Note that the DIP switch has two positions by default, with:

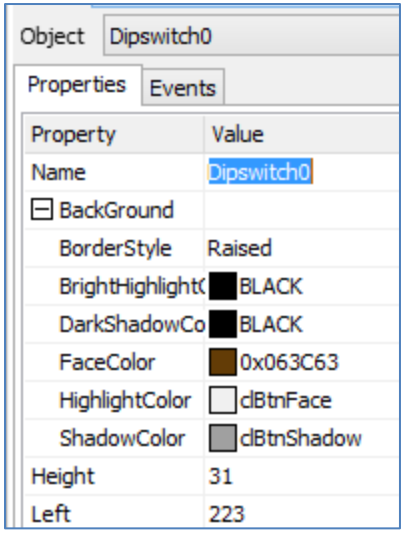
**0** or **off** as value when the switch is on top or leftmost position



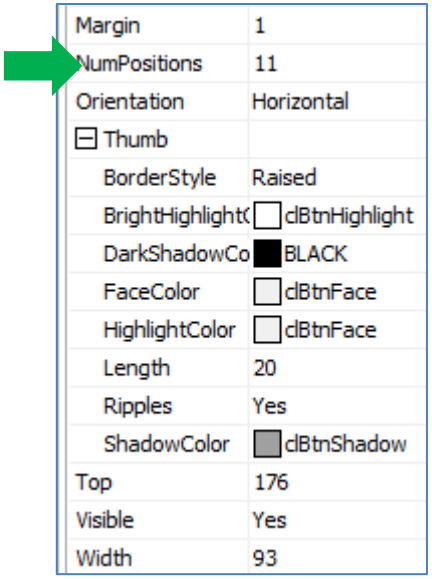
**1** or **on** as value when the switch is on the bottom or rightmost postion

## Add a DIP Switch – LED Digits Pair

A DIP switch can be configured to have more than two positions. Add another DIP- switch object and apply the following properties:



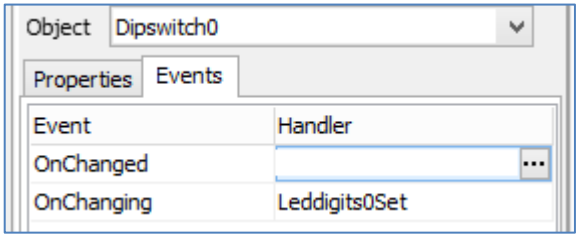
| Object | Dipswitch0 |
|---|---|
| Property | Value |
| Name | Dipswitch0 |
| BackGround | |
| BorderStyle | Raised |
| BrightHighlight( | BLACK |
| DarkShadowCo | BLACK |
| FaceColor | 0x063C63 |
| HighlightColor | clBtnFace |
| ShadowColor | clBtnShadow |
| Height | 31 |
| Left | 223 |



| | |
|---|---|
| Margin | 1 |
| NumPositions | 11 |
| Orientation | Horizontal |
| Thumb | |
| BorderStyle | Raised |
| BrightHighlight( | clBtnHighlight |
| DarkShadowCo | BLACK |
| FaceColor | clBtnFace |
| HighlightColor | clBtnFace |
| Length | 20 |
| Ripples | Yes |
| ShadowColor | clBtnShadow |
| Top | 176 |
| Visible | Yes |
| Width | 93 |

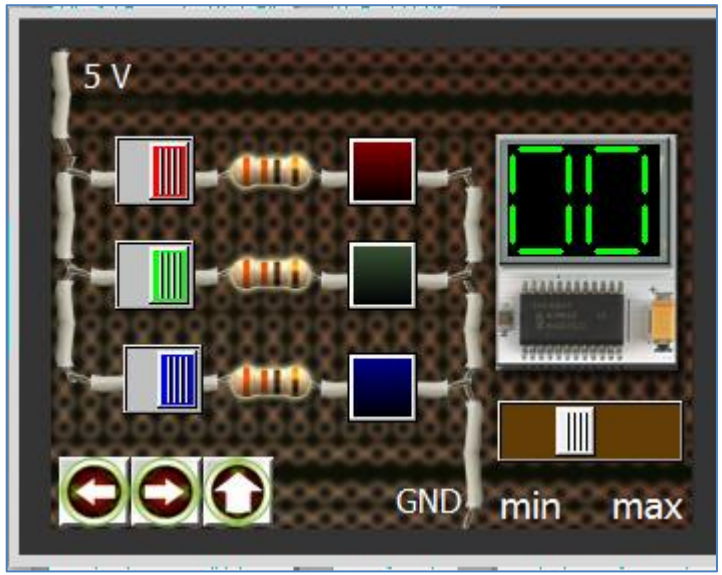
Note that the **NumPositions** property has a value of 11. When done, the DIP switch will look as shown below.

Now add a LED digits object which will display the position of the DIP switch. Also, configure the DIP switch as shown below to control the LED digits object.




### Add Navigation Buttons, Static Objects, and a Background Image

The navigation buttons used here are similar to those created in the previous forms. Make sure that they are configured correctly. Also, configure the next button of Form13(the previous form) to display this form when pressed and released. To make the interface more intuitive, a background image has been added (located here: **StarterKitDemos.ImgData**). The final apperance of the form is shown below.




For detailed information on the use of the DIP switch and the user LED, refer to:

[ViSi-Genie Inputs]

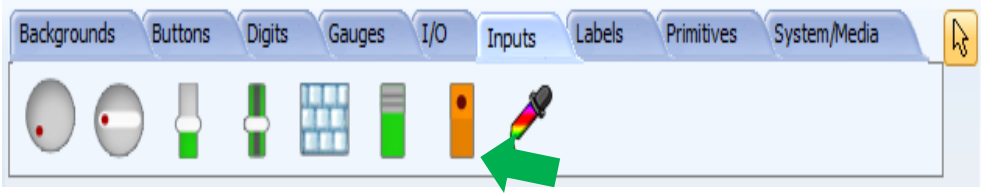
[ViSi-Genie Digital Displays]

## Input and Output Objects: Rocker and Rotary Switches – LED and LED Digits

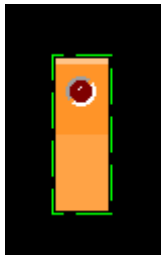
This form shows the basic use of the rocker switch, the rotary switch, and the LED.

### Add a Rocker Switch

Create a new form. To add a rocker switch, go to the Inputs pane and click on the rocker switch icon.




Click on the WYSIWYG screen to place it.



Apply the following properties to the rocker switch:

Object Inspector

Form Form15

Object Rockerswitch0

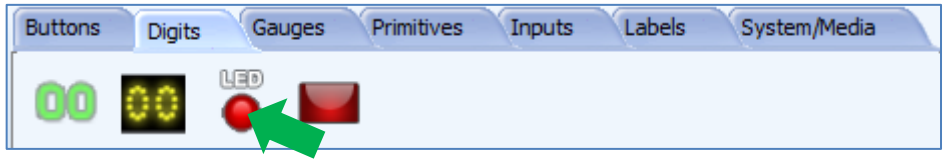
Properties | Events

| Property | Value |
|---|---|
| Name | Rockerswitch0 |
| BorderColor | BLACK |
| BorderWidth | 2 |
| ClickRect | Whole |
| Height | 80 |
| LED | |
| UseDefaultColors | Yes |
| AutoInactiveColor | Yes |
| Centered | Yes |

| | |
|---|---|
| ColorActive | clRed |
| ColorDarkShadow | BLACK |
| ColorHighlight | clBtnHighlight |
| ColorInactive | 0x00007F |
| ColorShadow | clBtnShadow |
| Height | 15 |
| Left | 7 |
| Shape | Ellipse |
| ShowReflection | Yes |
| Top | 8 |
| Visible | Yes |
| Width | 15 |
| Left | 140 |
| Orientation | Top |

## Add an LED Object

To add an LED, go to the digits pane and click on the LED icon.




Click on the WYSIWYG screen to place it then apply the following properties:

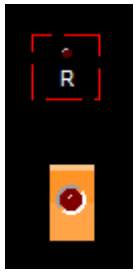
Object Led1

Properties | Events

| Property | Value |
|---|---|
| Name | Led1 |
| Caption | R |
| Color | BLACK |
| Font | (clWhite, [], Arial, 8, []) |
| Glyphs | (None) |
| Height | 33 |
| Layout | Top |

| | |
|---|---|
| LedType | Rounded |
| Left | 184 |
| Palette | |
| High | clRed |
| Low | 0x000051 |
| Spacing | 1 |
| Top | 56 |
| Visible | Yes |
| Width | 34 |

When done, the LED, together with the rocker switch, will look as shown below.




## Link the LED to the Rocker Switch

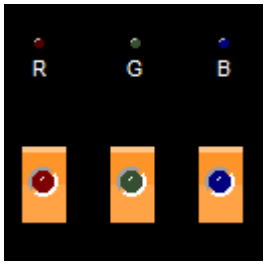
Configure the rocker switch as shown below.

Object Rockerswitch0

Properties | Events

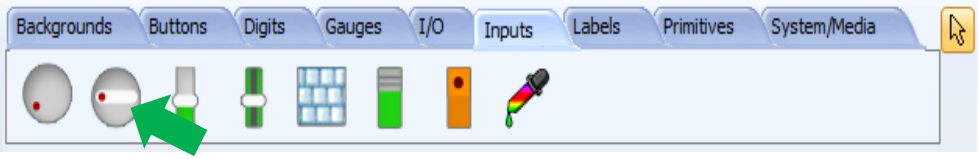
| Event | Handler |
|---|---|
| OnChanged | Led1Set |

Now create two more pairs – blue and green in color. Also, configure each rocker switch to correspond to the proper LED.

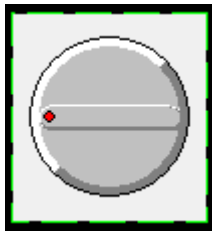



## Add a Rotary Switch – LED Digits Pair

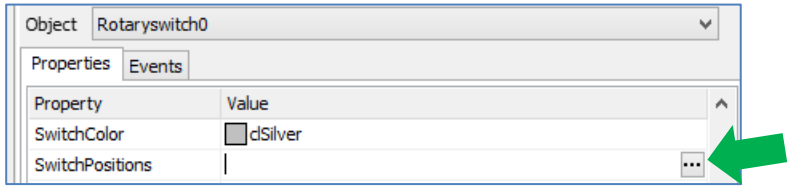
To add a rotary switch, go to the Inputs pane and click on the rotary switch icon.



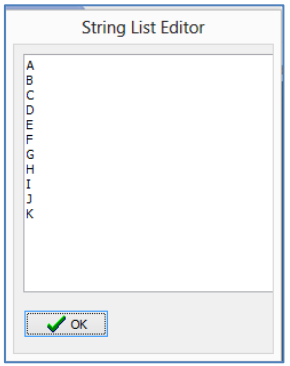

Click on WYSIWYG screen to place it.



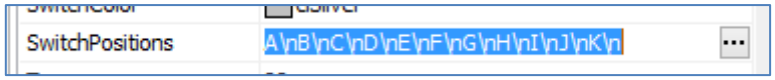
In the Object Inspector, click on the ··· symbol of the SwitchPositions property line.




The String List Editor window appears. Type in the letters from A to K then click OK.




The value of the SwitchPositions property is updated.




Apply the following additional properties to the rotary switch.

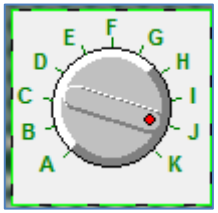
Object Rotaryswitch0

Properties Events

| Property | Value |
|---|---|
| Name | Rotaryswitch0 |
| ButtonColor | RED |
| Color | clBtnFace |
| Font | (GREEN, [], Arial, 8, [Bold]) |
| Height | 98 |
| LabelsOffset | 10 |
| Left | 52 |

| | |
|---|---|
| Radius | 30 |
| ShowLabel | Yes |
| SwitchAngleEnd | 315 |
| SwitchAngleStart | 45 |
| SwitchColor | clSilver |
| SwitchPositions | A\nB\nC\nD\nE\nF\nG\ |
| Top | 136 |
| Width | 100 |
| WinchColor | clSilver |
| WinchOffset | 10 |

When done, the rotary switch will look as shown below.




Now add a LED digits object to display the value of the rotary switch when turned. Configure the rotary switch to control the LED digits as shown below.

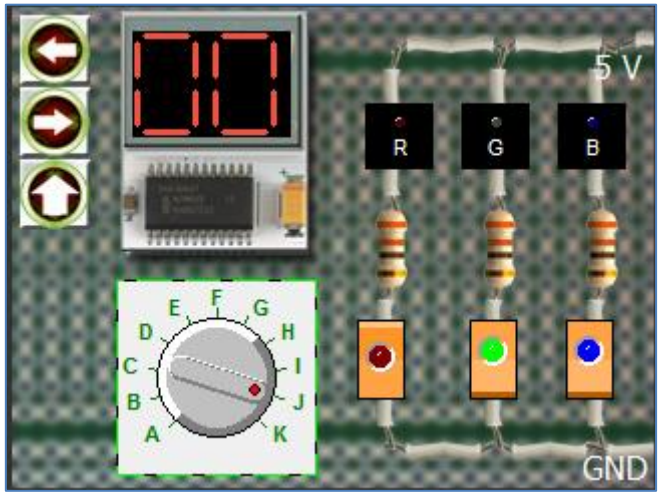
Object Rotaryswitch0

Properties Events

| Event | Handler |
|---|---|
| OnChanged | |
| OnChanging | Leddigits2Set |

## Add Navigation Buttons, Static Objects, and a Background Image

The navigation buttons used here are similar to those created in the previous forms. Make sure that they are configured correctly. Also, configure the next button of Form14 (the previous form) to display this form when pressed and released. To make the interface more intuitive, a background image has been added (located here: **StarterKitDemos.ImgData**). The final apperance of the form is shown below.




For detailed information on the use of the rotary switch and the LED, refer to:

[ViSi-Genie Inputs]

[ViSi-Genie Digital Displays]

## Build and Upload the Project

For instructions on how to build and upload a ViSi-Genie project to the target display, please refer to the section "**Build and Upload the Project**" of the application note

**ViSi Genie Getting Started – First Project for Picaso Displays** (for Picaso/diablo16)
or
**ViSi Genie Getting Started – First Project for Diablo16 Displays** (for Diablo16).

## Proprietary Information



## Disclaimer of Warranties & Limitation of Liability