



# ViSi Genie 4D Buttons

DOCUMENT DATE: **11<sup>th</sup> APRIL 2019**  
DOCUMENT REVISION: **1.1**



## Description

This application note provides a first hands-on example with ViSi-Genie and describes all the steps related to a project.

Before getting started, the following are required:

- Workshop 4 has been installed according to the document Workshop 4 Installation;
- The user is familiar with the Workshop 4 environment and with the fundamentals of ViSi-Genie, as described in Workshop 4 User Guide and ViSi-Genie User Guide;
- 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 discussed in these recommended application notes.

















## Content

<b>Description</b> .....	<b>2</b>
<b>Content</b> .....	<b>2</b>
<b>Application Overview</b> .....	<b>3</b>
<b>Setup Procedure</b> .....	<b>4</b>
<b>Create a New Project</b> .....	<b>4</b>
<i>Create a New Project</i> .....	<b>4</b>
<b>Design the Project</b> .....	<b>5</b>
<i>Adding a Momentary 4D Button</i> .....	<b>5</b>
Setting the Button Size .....	<b>6</b>
Setting the Button Style .....	<b>7</b>
<i>Naming of Objects</i> .....	<b>8</b>
<i>Adding a Static Text</i> .....	<b>8</b>
<i>Adding a New Form</i> .....	<b>10</b>
<i>Linking Forms</i> .....	<b>10</b>
<i>Adding a Toggle 4D Button</i> .....	<b>11</b>
Changing the Button Size and Style .....	<b>12</b>
Configuring a 4D Button as a Toggle Button .....	<b>12</b>
<i>Adding a User LED</i> .....	<b>12</b>
Linking Objects .....	<b>13</b>
Configuring the User LED to Report a Message .....	<b>13</b>
<i>Create a Button Matrix</i> .....	<b>14</b>
Adding Four 4D Buttons .....	<b>15</b>

Configuring the Four 4D Buttons as a Matrix	15
Configuring the Four 4D Buttons to Report a Message	16
Complete the Form	17
The Project	17
<b>Build and Upload the Project</b> .....	<b>19</b>
<b>Identify the Messages</b> .....	<b>19</b>
<i>Use the GTX Tool to Analyse the Messages</i> .....	19
Launch the GTX Tool	19
<i>The 4D Buttons</i> .....	20
Report Event	20
Input and Output Objects	23
Toggle a 4D Button using the GTX Tool	23
Polling a 4D Button	24
<b>Proprietary Information</b> .....	<b>25</b>
<b>Disclaimer of Warranties &amp; Limitation of Liability</b> .....	<b>25</b>

## 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 users to see the instant results of their desired graphical layout with this large selection of gauges and meters (called objects or widgets). The user can simply click on the desired widget to select it and click on the simulated display to place the widget. Shown below are the 4D button widgets.

4D Buttons					
Name	Icon		Name	Icon	
Button01			Rocker03		
Button02			Slider01		
Rocker01			Toggle01		
Rocker02			Toggle02		

There are eight 4D buttons available in Workshop. For each 4D button, the user can choose among the predefined sizes and styles. 4D buttons can be turned in to a momentary (default), toggle, or matrix type. Each of these configurations has its own application. The simple project developed in this application note demonstrates the basic use of 4D buttons.

The section **“Identify the Messages”** discusses the format of the messages coming from and going to a 4D button using the GTX Tool in Workshop. An understanding of this section is essential for users who intend to interface the display to an external host.

The 4D buttons, together with the user and animated buttons, are the latest addition to the growing list of widgets available in Workshop (4.1.0.11). For the user button, the user provides the image for each state. For the animated button, the user has the option of choosing the image for each frame or state as well as setting the frame delay. When the animated button is pressed and released, the frames are sequentially displayed at the predefined interval. Workshop versions prior to 4.1.0.11 have only one available button which is the win or fancy button.

To know more of the differences and similarities between the various button widgets in Workshop, users are encouraged to read the dedicated application notes. Other widgets very similar to buttons are the DIP switch and rocker switch which are under the inputs pane. These objects have a few more additional features and properties that can be defined by the user as compared to the widgets under the buttons pane.

## 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)

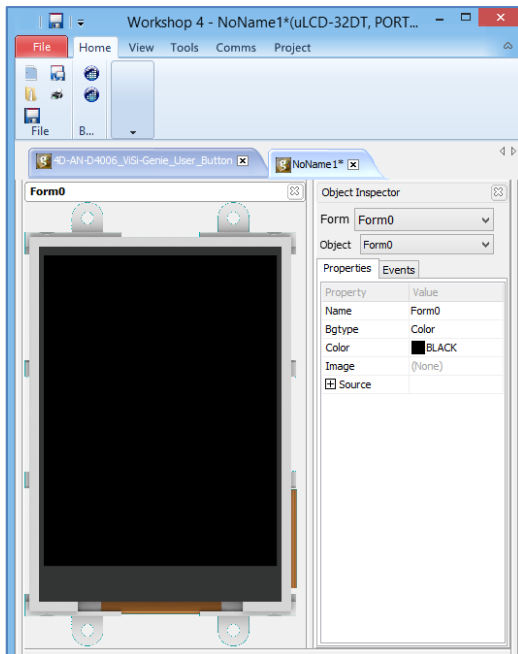
or

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

## Design the Project

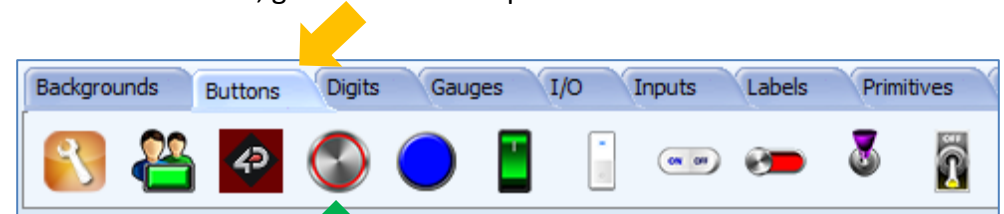
Everything is now ready to start designing the project. **Workshop 4** displays an empty screen, called **Form0**. A **form** is like a page on the screen. The form can contain **widgets** or **objects** like trackbars, sliders, displays or keyboards.

Below is an empty form.

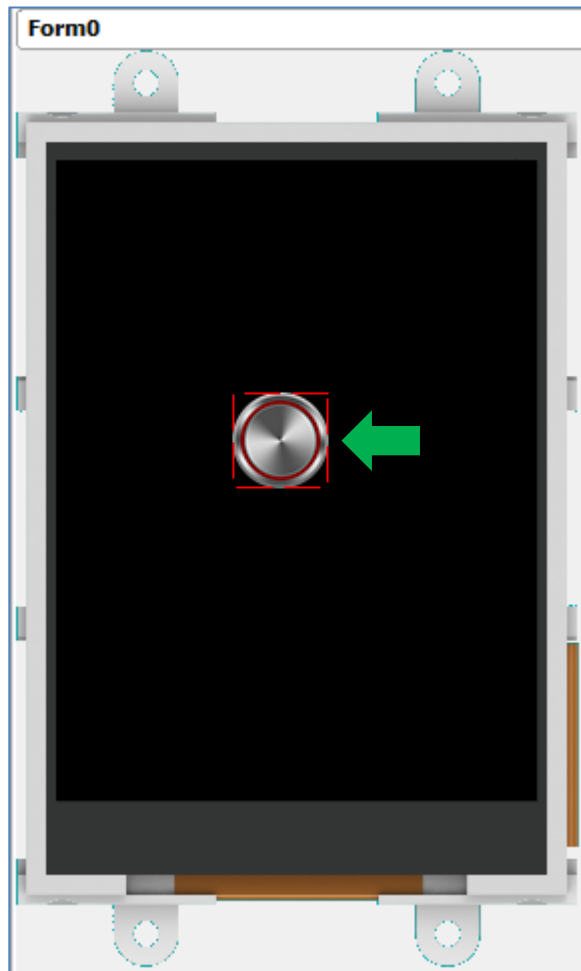


## Adding a Momentary 4D Button

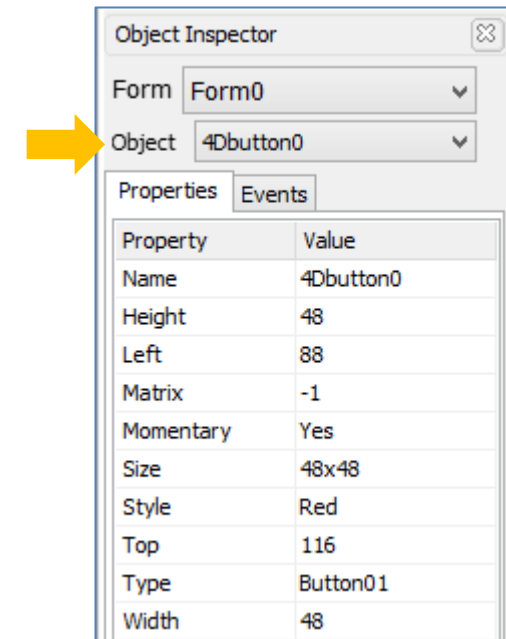
To add a 4Dbutton, go to the **Buttons** pane then click on the **Button01** icon.



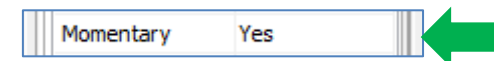
Click on the **WYSIWYG** (What-You-See-Is-What-You-Get) screen to put the object in place. The WYSIWYG screen simulates the actual appearance of the display module screen.



The object can be dragged to any desired location. The **Object Inspector** on the right part of the screen displays all the properties of the newly created 4D button object named **4Dbutton0**.

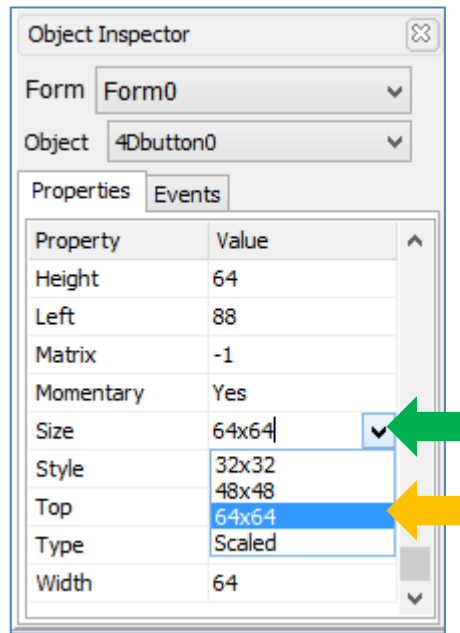


By default, a 4D button is a momentary button. Note that the property “Momentary” is set to “Yes”.



### Setting the Button Size

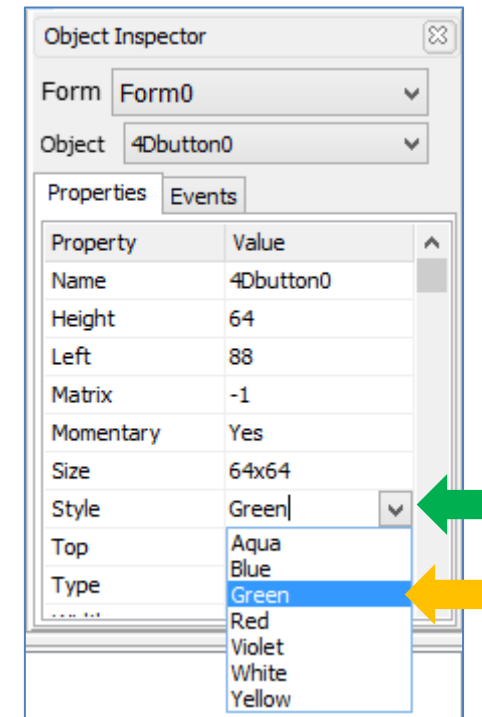
Button01 has three predefined sizes. Change the size by clicking on the drop-down menu arrow of the property “Size” in the object inspector and selecting “64x64” (pixels) as shown below.



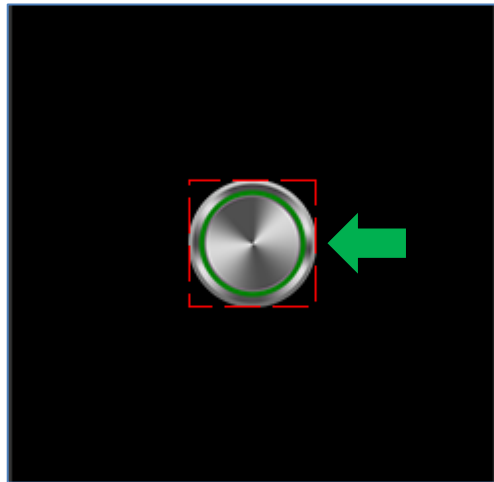
The user can also scale the size of the object by selecting the last option “Scaled”.

#### Setting the Button Style

Change the color of the button light from red to green by selecting the appropriate option under the property “Style” in the object inspector.

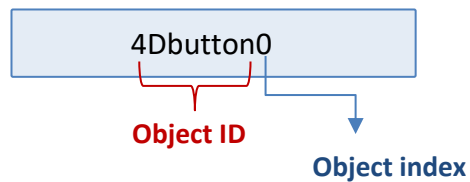


The WYSIWYG screen is updated accordingly.



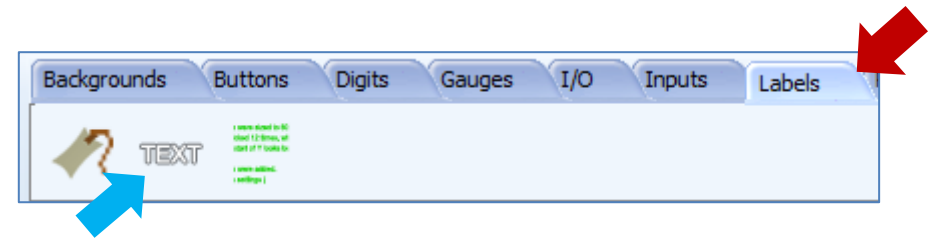
### Naming of Objects

Naming is important to differentiate between objects of the same kind. For instance, suppose the user adds another 4D button object to the WYSIWYG screen. This object will be given the name 4Dbutton1– it being the second 4D button object in the program. The third 4D button will be given the name 4Dbutton2, and so on. An object's name therefore identifies its kind and its unique index number. It has an ID (or type) and an index.



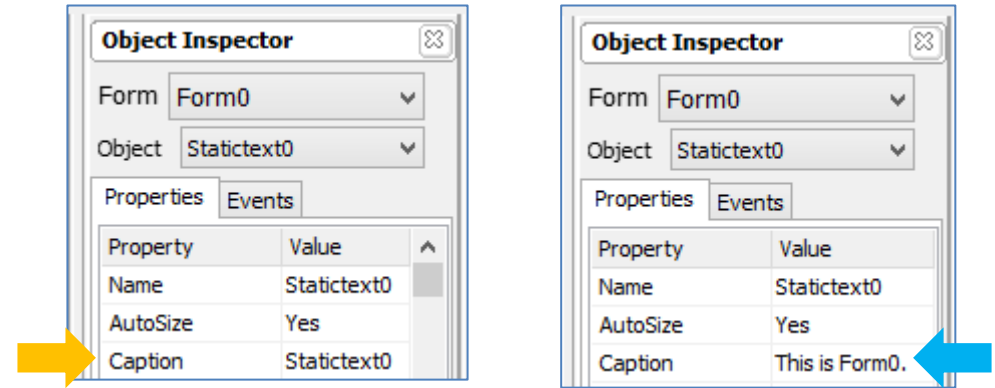
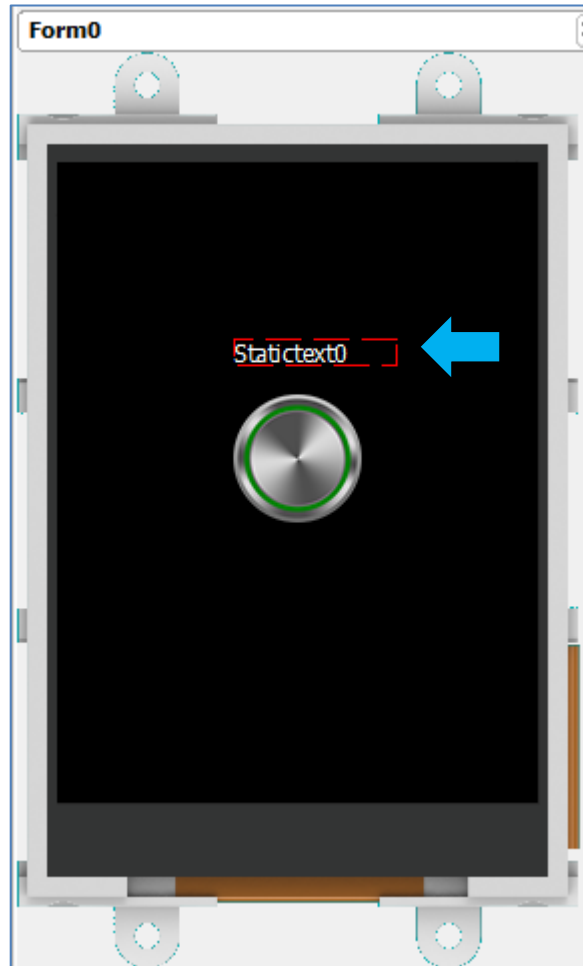
### Adding a Static Text

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

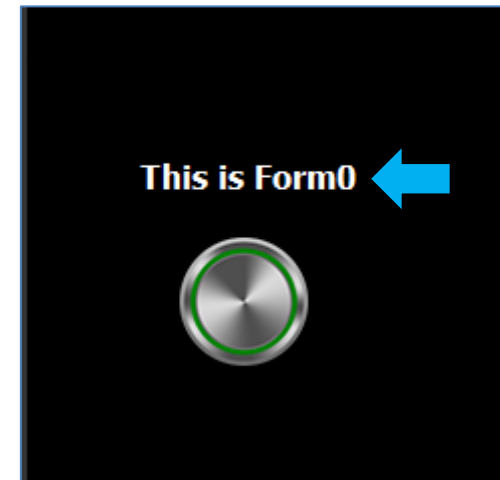


Click on the WYSIWYG screen to place the object. Drag it to any desired location.





Experimentation with the other properties of the static text is left to the user as an exercise. The WYSIWYG screen is updated accordingly.



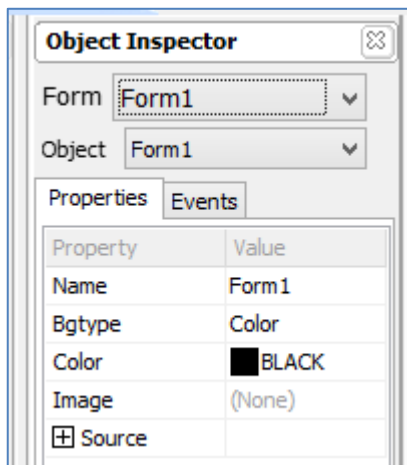
Change the caption in the object inspector.

## Adding a New Form

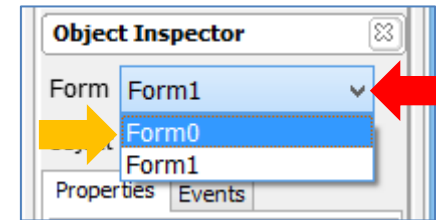
A new form is added the purpose of which is to demonstrate the use of a 4D button configured for toggling. To add a new form, go to the System/Media pane and select the form icon.



A new blank form, named Form1, is generated.



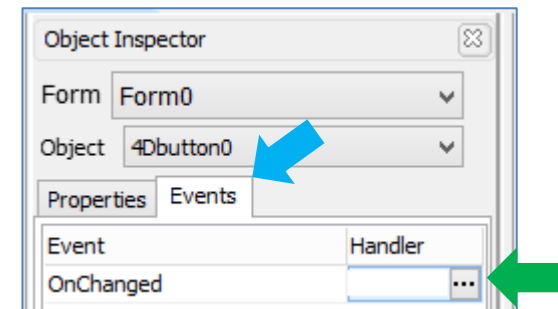
To navigate back to Form0 in Workshop, click on the drop-down menu arrow of the object inspector as shown below and select Form0.



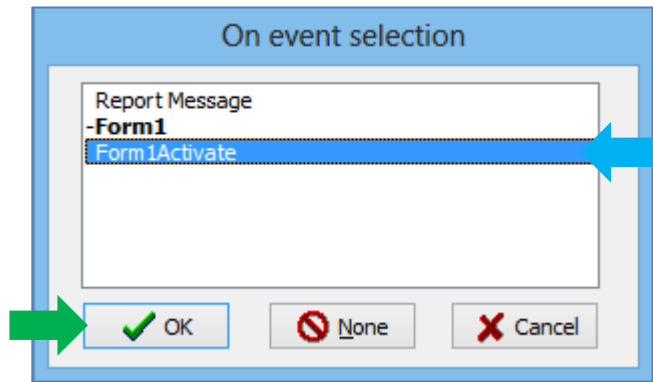
Do this to navigate between forms.

## Linking Forms

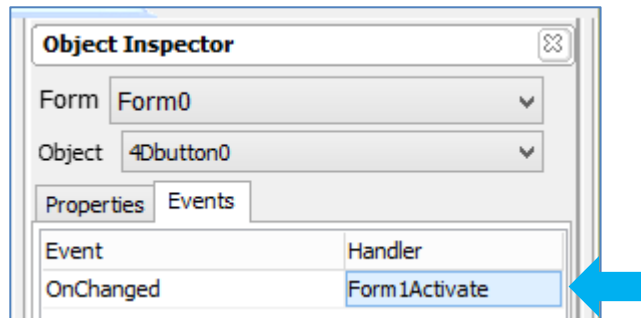
Now that there are two forms, a button in Form0 can be used to display Form1 (and vice-versa) when the application runs. Here 4Dbutton0 of Form0 is configured as a navigation button. In Form0, select 4Dbutton0, go to the object inspector, and select the Events tab. Click on the ellipsis dots of the OnChanged event.



The On event selection window appears. Choose “Form1Activate” and click OK.

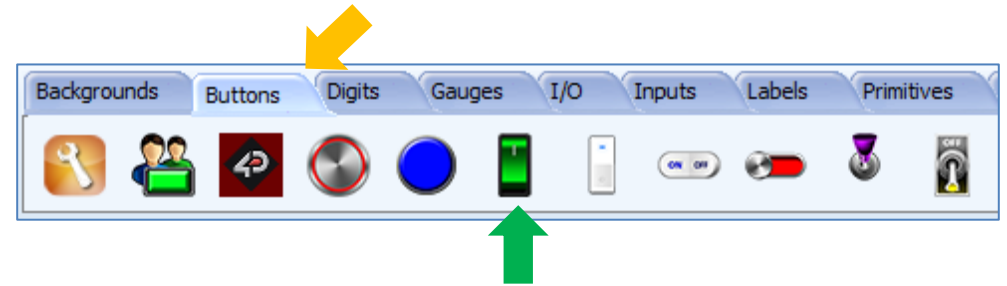


The object inspector is updated accordingly.

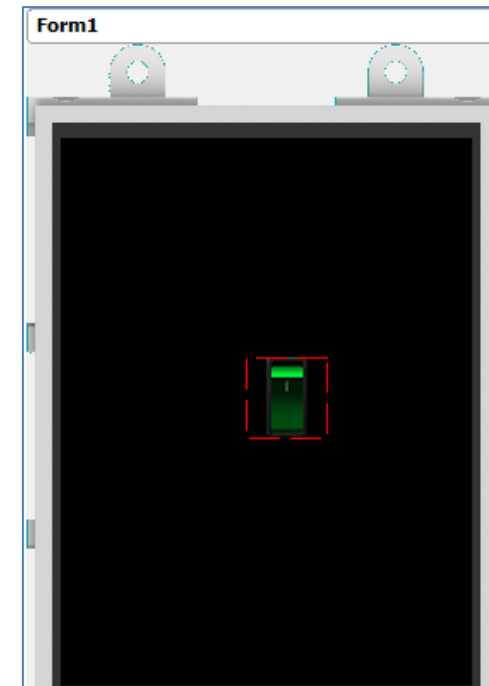


### Adding a Toggle 4D Button

Another 4D button (4Dbutton1) is added in Form1. This button will be configured as a toggle button. A toggle button is enabled when pressed, and stays enabled when released. Another press-and-release is needed to disable it. For 4Dbutton1, a Rocker01 widget is used.

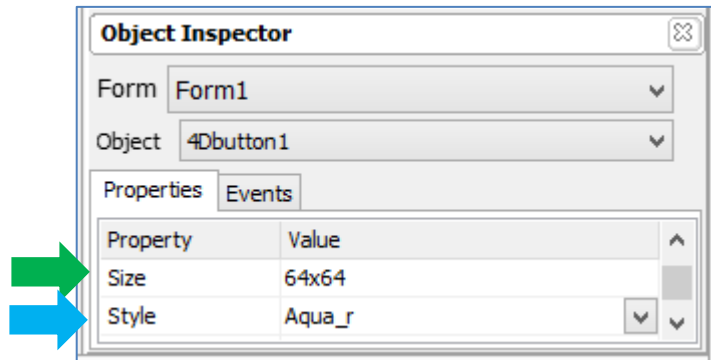


The WYSIWYG screen is updated.



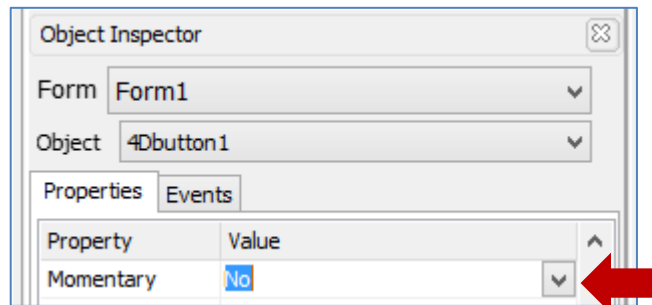
### Changing the Button Size and Style

Change the button size to “64x64” (pixels) and the style to “Aqua\_r”. The “r” in “Aqua\_r” stands for “reversed”.

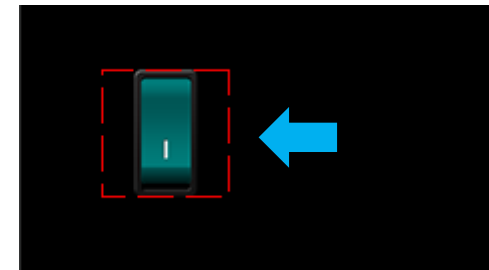


### Configuring a 4D Button as a Toggle Button

To configure 4Dbutton1 to behave as a toggle button, go to the object inspector and set the property “Momentary” to “No”.

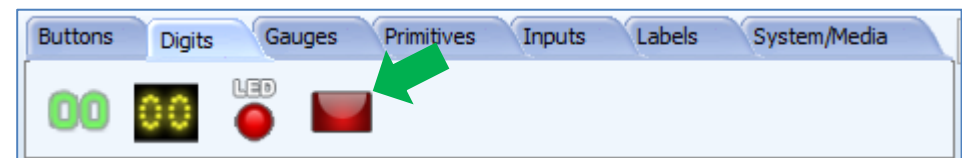


The WYSIWYG screen is updated.

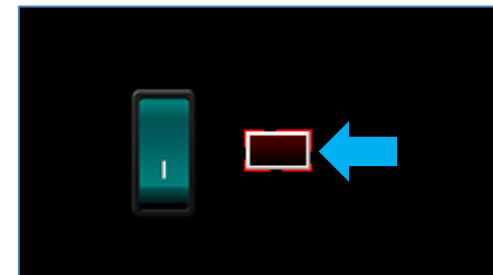


### Adding a User LED

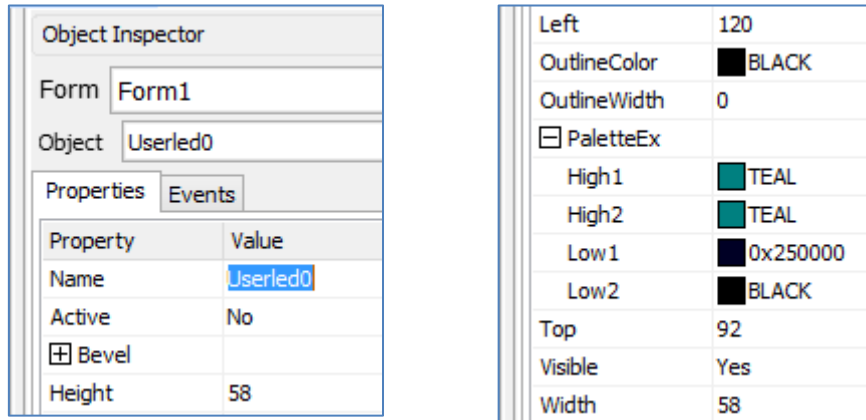
To add a user LED, go to the Digits pane and select the user LED icon.



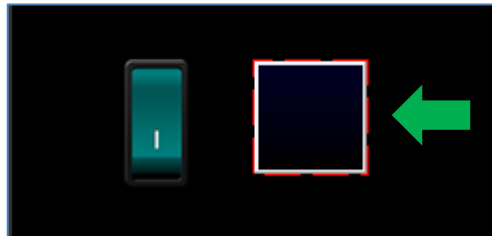
Click on the WYSIWYG screen to place it.



The object can be dragged and resized. The properties can be edited in the Object Inspector. Userled0 in Form1 has the following properties:

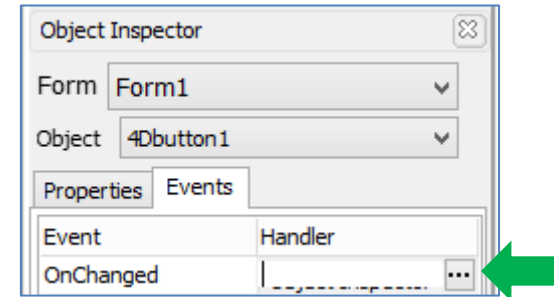


The updated appearance of the WYSIWYG screen is shown below.

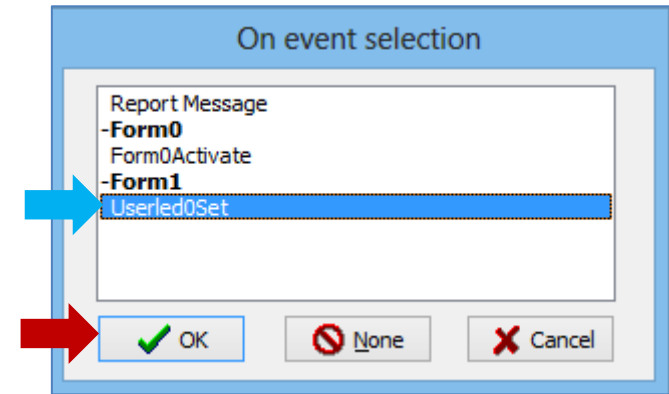


**Linking Objects**

Now that 4Dbutton1 is a toggle button, it can be used to control Userled0. Go to 4Dbutton1's object inspector and click on the ellipsis dots of the onChanged event.



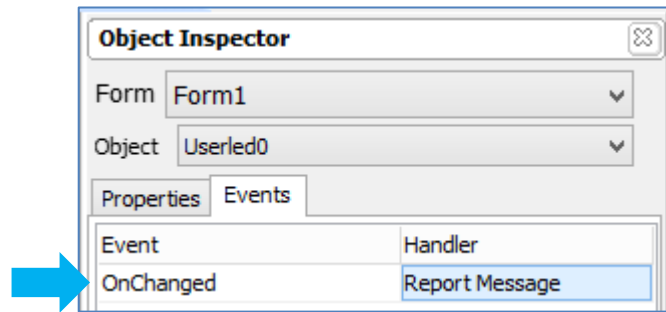
The On event selection window appears. Select Userled0Set and click OK.



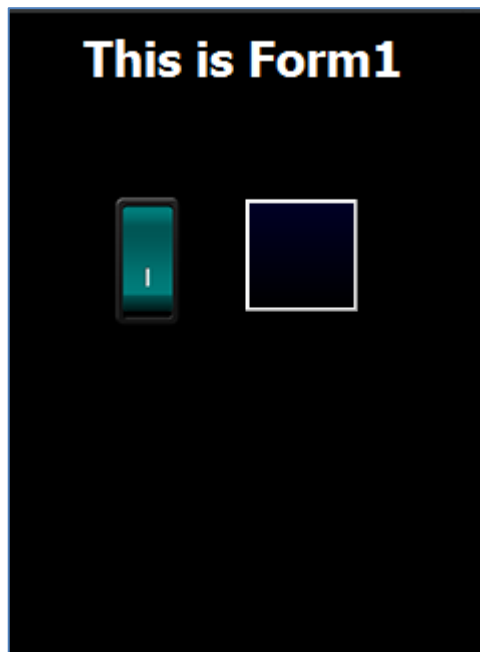
When the program runs, 4Dbutton1 will toggle Userled0.

**Configuring the User LED to Report a Message**

Userled0 can be configured to report a message to an external host controller when its status has changed. Go to its object inspector and configure the onChanged event as shown below.

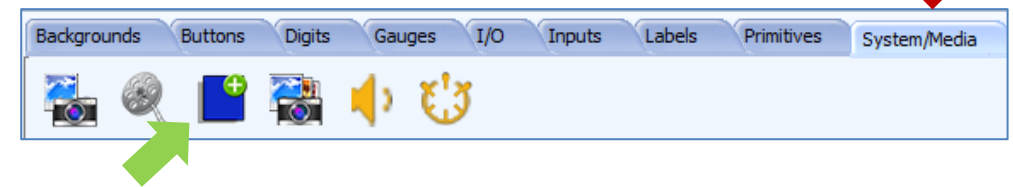


Add another static text object to give the form a label. When done, the form should look like as shown below.



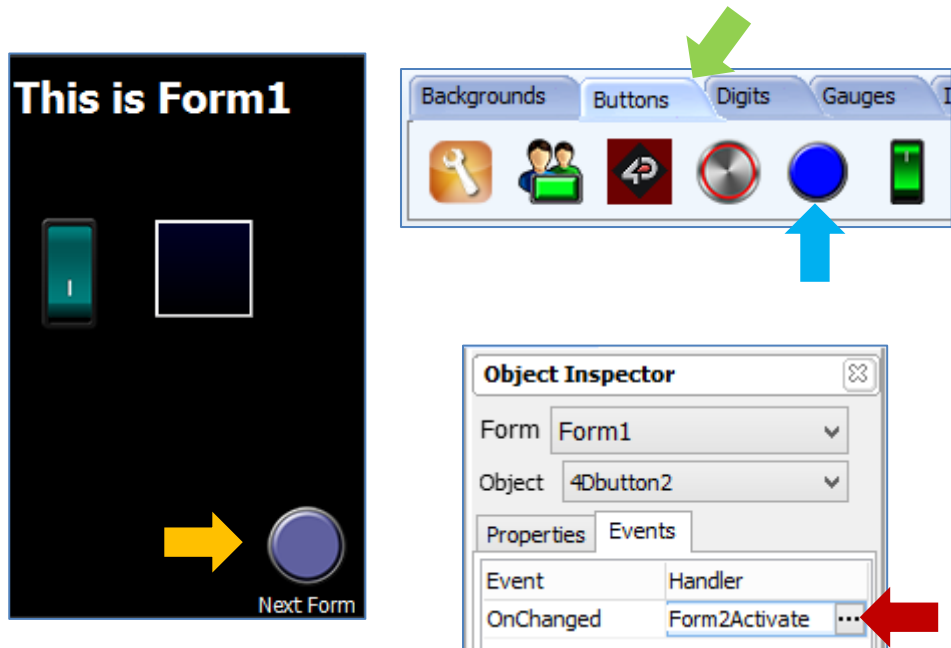
### Create a Button Matrix

Add another form to the project – Form2.



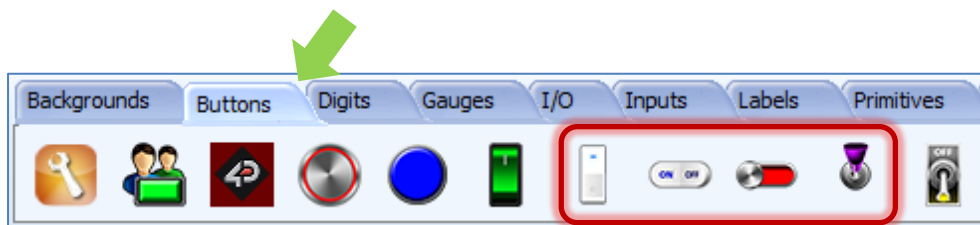
This form will contain four 4D buttons which form a matrix. Buttons in a matrix behave such that only one button can be enabled at a time. Enabling another button disables all the other buttons. A matrix of buttons can be conveniently used as a GUI menu.

First, create a navigation button to link Form1 to Form2. In Form1, a momentary 4D button (Button02) is added together with a label.

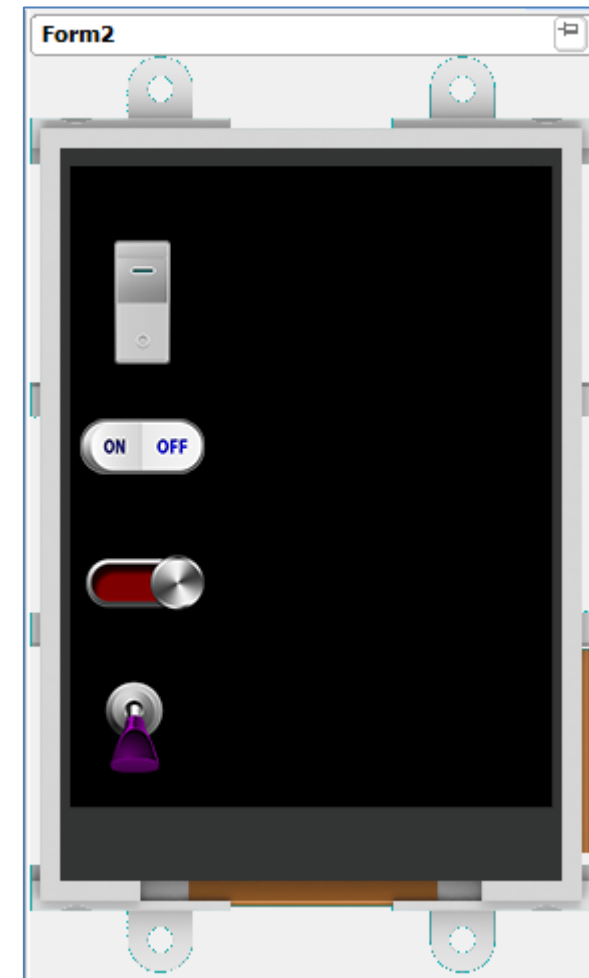


### Adding Four 4D Buttons

Add four 4D buttons to Form2. These are 4Dbutton3, 4Dbutton4, 4Dbutton5, and 4Dbutton6. Use the widgets indicated below.

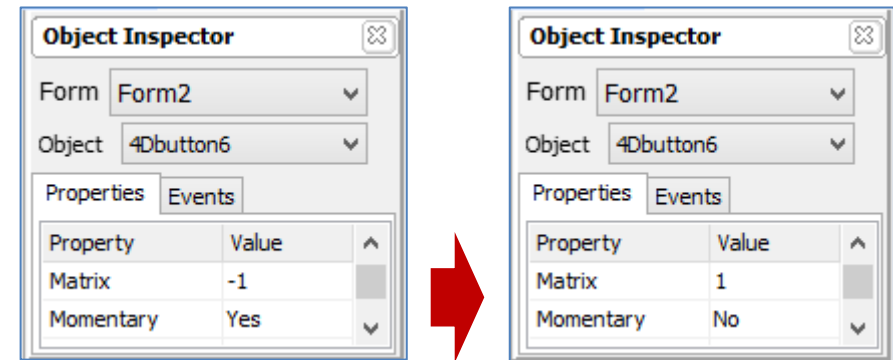
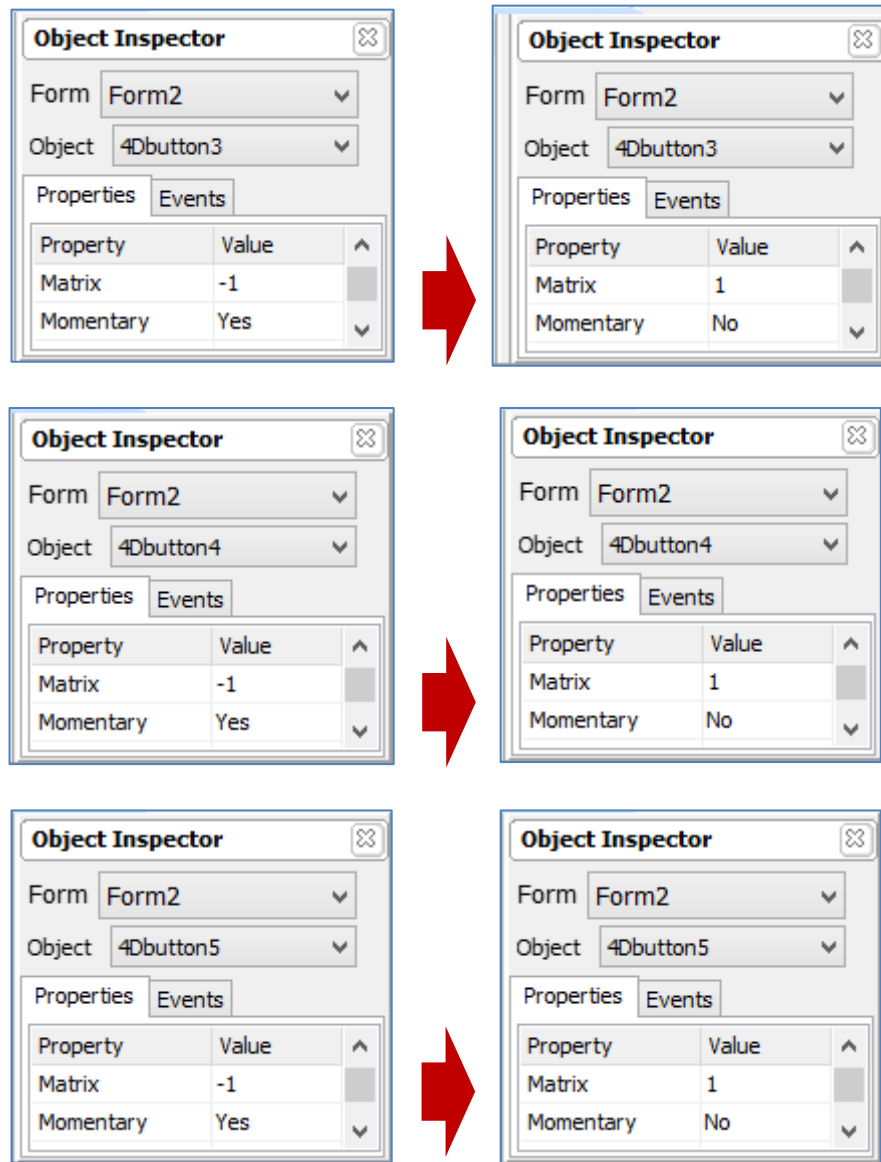


The WYSIWYG screen is updated. Each of the four buttons will be configured as a toggle button.



### Configuring the Four 4D Buttons as a Matrix

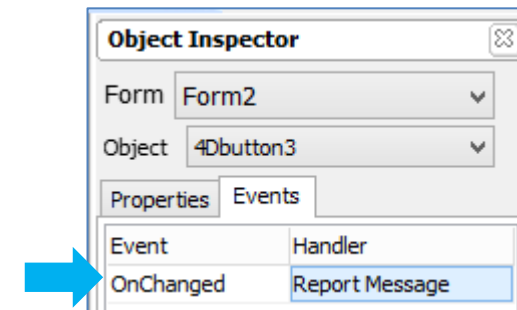
To group these buttons into a matrix, edit the property "Matrix" in the object inspector. Set "Momentary" to "No" as well.



It is important that all buttons grouped share the same matrix number, otherwise pressing on one button will not release the other buttons of the group.

#### Configuring the Four 4D Buttons to Report a Message

A 4D button can be configured to report a message to an external host controller when its status has changed. Go to the object inspector of 4Dbutton3 and configure the onChanged event as shown below.

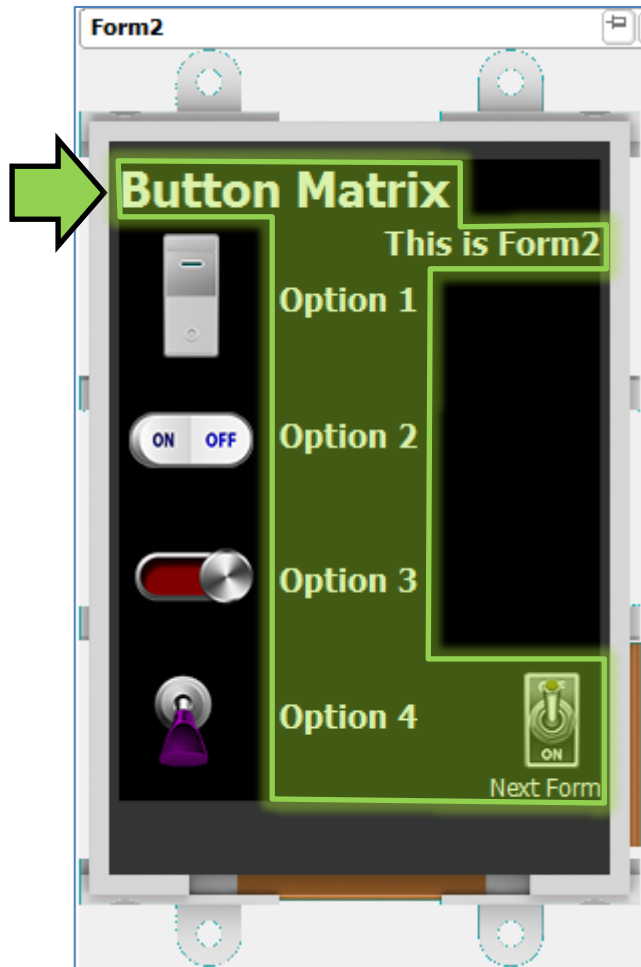


Do the same for 4Dbuttons 4 to 6.

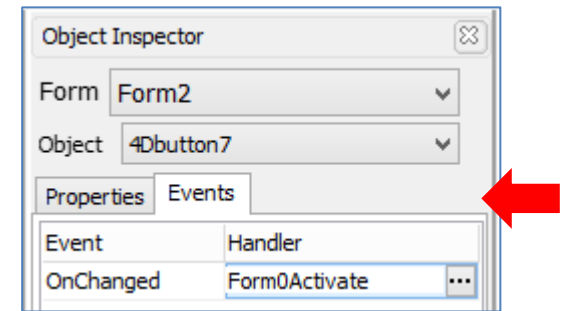
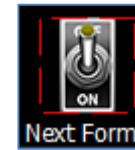


**Complete the Form**

Add the static objects and a navigation button (4Dbutton7) to Form2.



4Dbutton7 is linked to the first form – Form0.

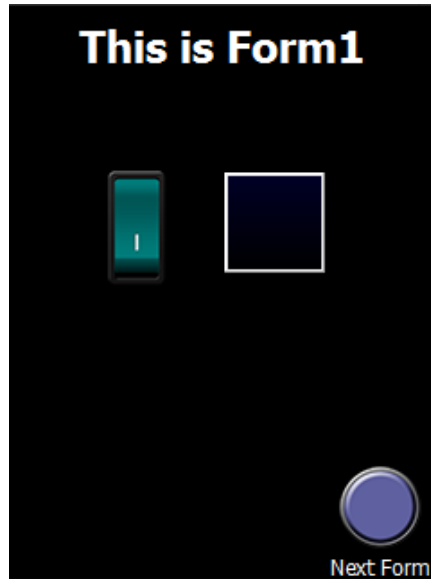


**The Project**

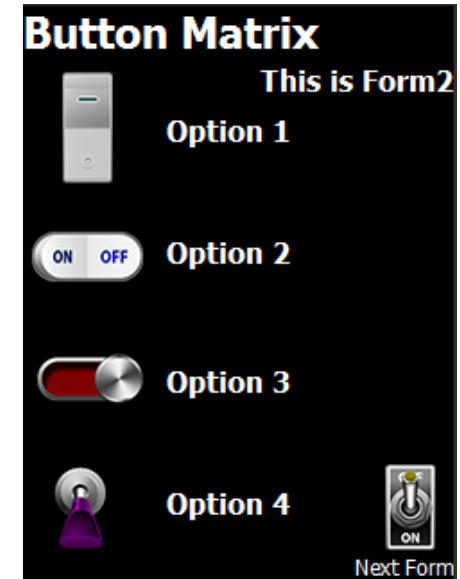
The project now has three forms. In Form0 is 4Dbutton0 which is a momentary button used for displaying the next form – Form1.



In Form1 are 4Dbutton1 and 4Dbutton2. 4Dbutton1 is a toggle button used to control a user LED. 4Dbutton2 is a momentary navigation button used to display the next form – Form2.



In Form2 are 4Dbutton3, 4Dbutton4, 4Dbutton5, 4Dbutton6, and 4Dbutton7. 4Dbutton3 to 6 are toggle buttons that form a matrix while 4Dbutton7 is a momentary button for navigating back to the first form.



## 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)

or

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

The uLCD-32PTU and/or the uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.

## Identify the Messages

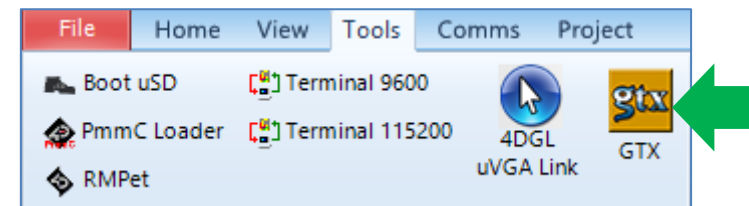
The display module is going to receive and send messages from and to an external host. This section explains to the user how to interpret these messages. An understanding of this section is necessary for users who intend to interface the display to a host. The [ViSi Genie Reference Manual](#) is recommended for advanced users.

### Use the GTX Tool to Analyse the Messages

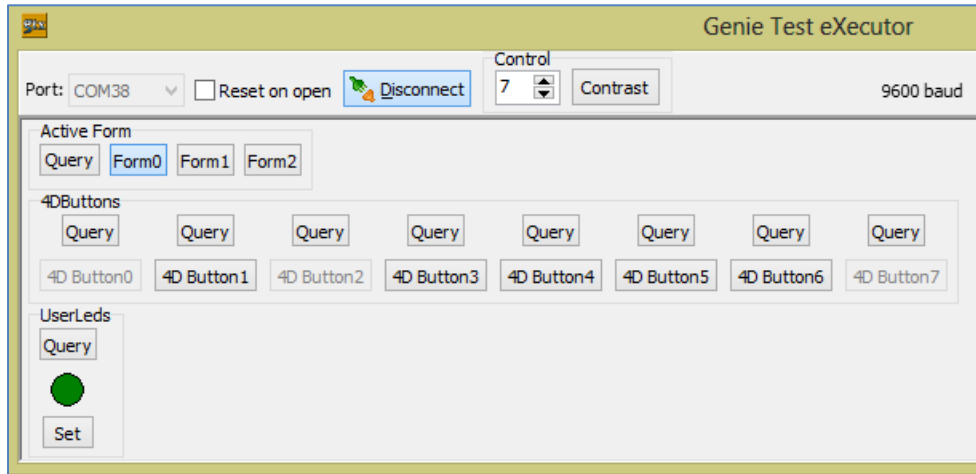
Using the GTX or **Genie Test eXecutor** tool is the first option to get the messages sent by the screen to the host. Here the PC will be the host. The GTX tool is a part of the Workshop 4 IDE. It allows the user to receive, observe, and send messages from and to the display module. It is an essential debugging tool.

### Launch the GTX Tool

Under Tools menu click on the GTX tool button.



A new window appears, with the forms, 4D buttons, and the user LED created previously.



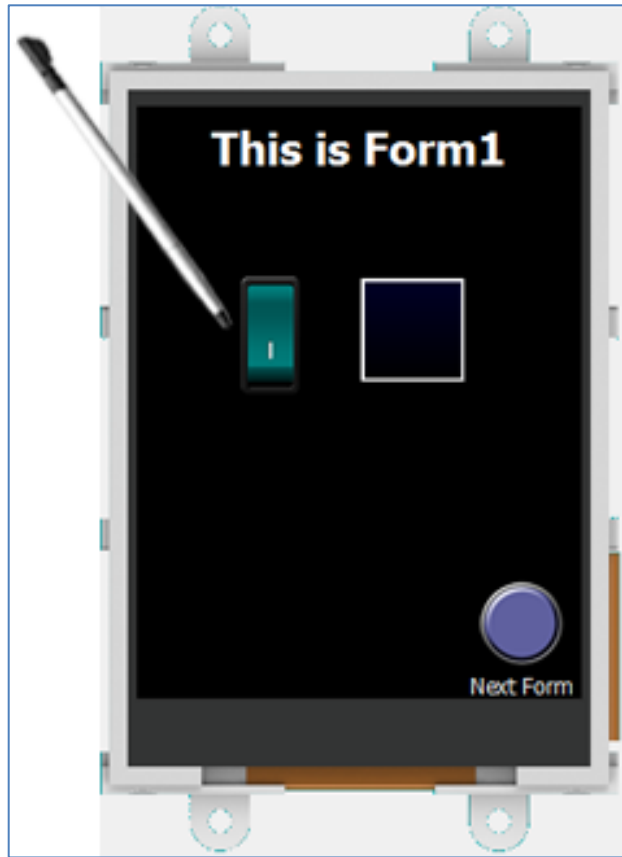
## The 4D Buttons

### Report Event

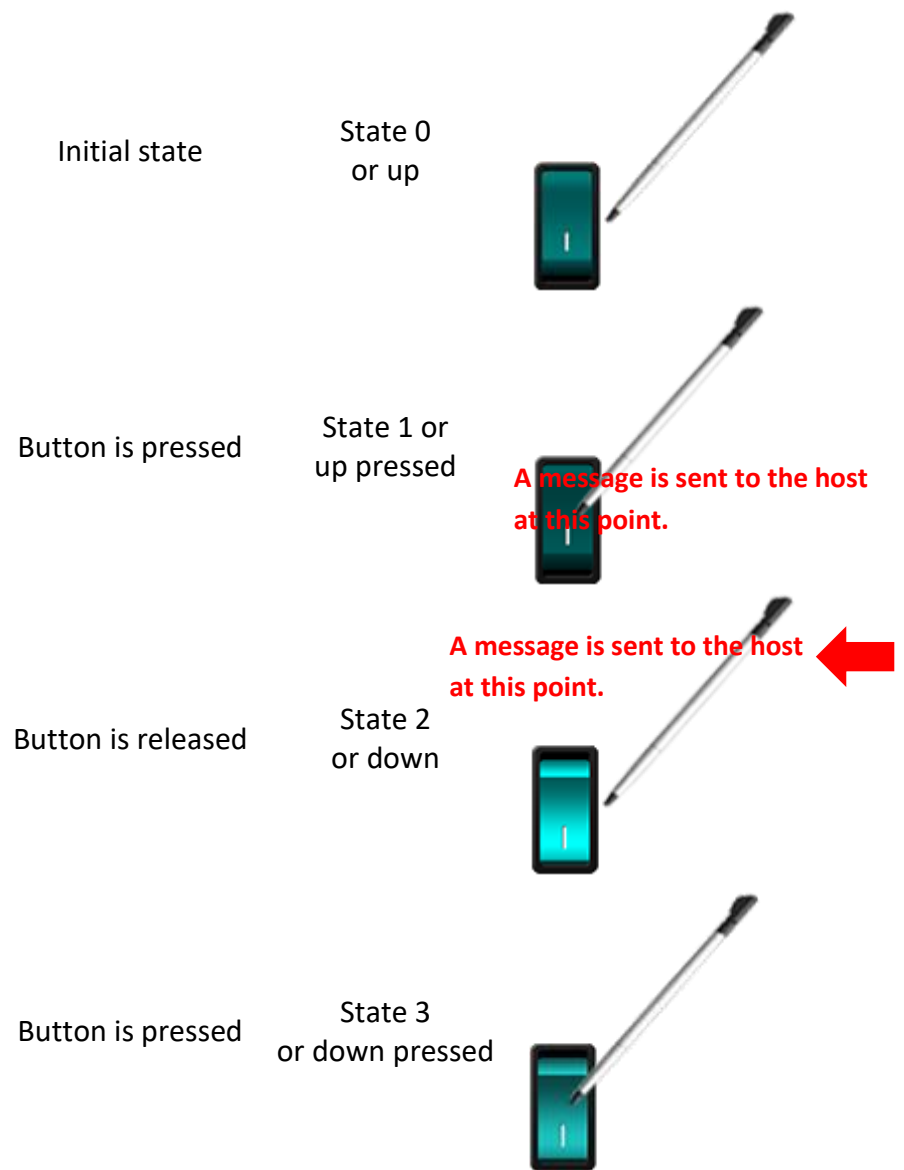
When the program starts, navigate to Form1 (the second form) by pressing the navigation button on Form0 (the first form) of the display module screen.



When Form1 is displayed, toggle the user LED by playing with 4Dbutton1.



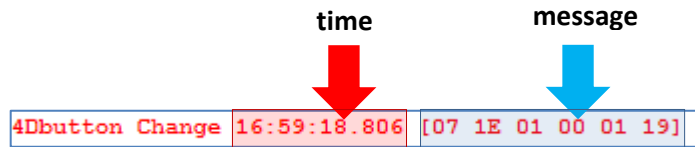
By following the cycle illustrated below, the user will observe that a message is sent to the host upon the release of a press.



Observe the white area on the right part of the GTX tool window. It displays the message received from the display module.

```
4Dbutton Change 16:59:18.806 [07 1E 01 00 01 19]
```

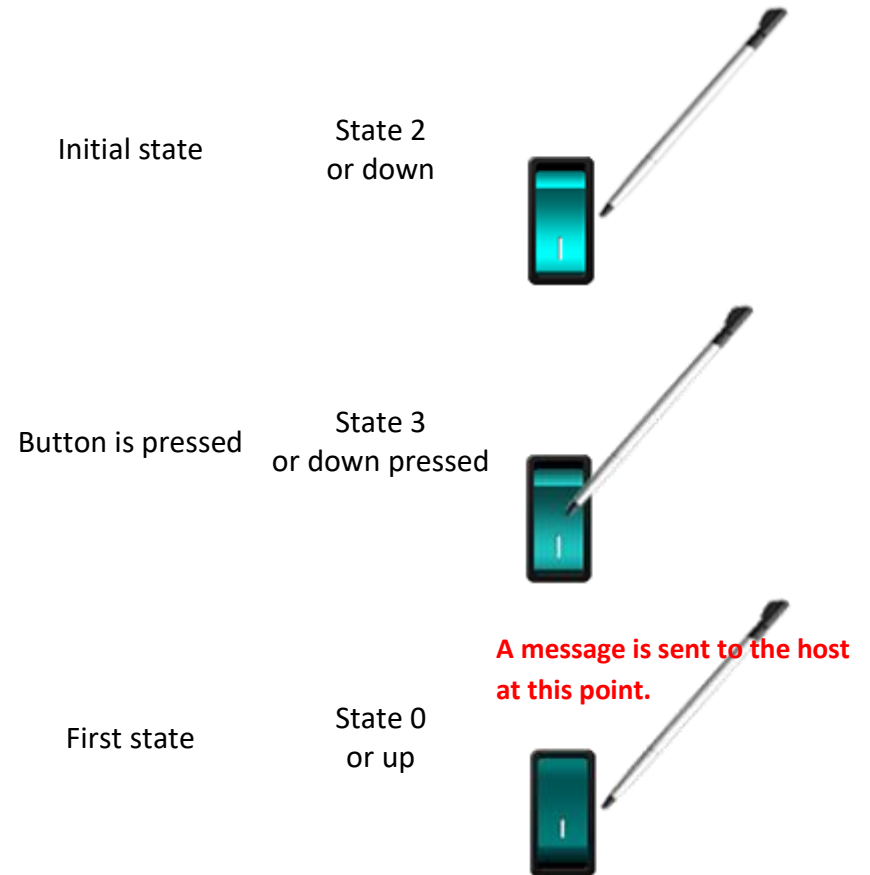
The actual message bytes are those inside the brackets. These values are in hexadecimal. The figure preceding the actual message is the computer time at which the message is sent. A label is also included to tell the observer what the message represents.



The message received is formatted according to the following pattern:

Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
07	1E	01	00	01	19
REPORT_EVENT	4D button	Second	0x0001		

The message is from 4Dbutton0, and it contains the hexadecimal value “0x00001”. When a toggle button is enabled, the value is **0x00001** or simply **1** in decimal. When a toggle button is disabled, the value is **0x0000** or simply **0** in decimal. Verify this by toggling the button back to its very first state.



The message sent when the 4D button has just been disabled is shown below.

```
4Dbutton Change 08:22:37.614 [07 1E 01 00 00 18]
```

The checksum is a means for the host to verify if the message received is correct. This byte is calculated by XOR'ing all bytes in the message from (and

including) the CMD or command byte to the last parameter byte. Then, the result is appended to the end to yield the checksum byte. If the message is correct, XOR'ing all the bytes (including the checksum byte) will give a result of zero. Checking the integrity of a message using the checksum byte shall be handled by the host.

### Input and Output Objects

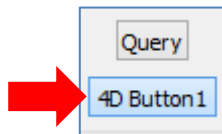
Remember that when designing Form1 earlier, Userled0 was configured to report a message when its status has changed. 4Dbutton1 on the other hand was only configured to toggle Userled0. The user therefore might have expected the message to be similar to:

UserLED Change 17:23:26.552 [07 13 00 00 01 32]

The object ID of a user LED is 0x13. See section 3.3 of the [ViSi Genie Reference Manual](#) for a full list of object IDs. The above is not the case however since a user LED is classified as an output object. Only input objects such as a 4D button can initiate an event. The message therefore is actually from 4Dbutton1, although it is Userled0 that was configured to report a message. To learn more about the input-output classification of Genie objects, refer to section 7 of the [ViSi-Genie User Guide](#).

### Toggle a 4D Button using the GTX Tool

In the GTX tool window, click on the button indicated below.



Note that 4Dbutton1 in Form1 of the display module screen has changed its state. Also, the white area on the right displays

- in **green** the messages sent to the display module
- and in **red** the messages received from the display module

```
Set 4Dbutton Value 08:28:01.814 [01 1E 01 00 01 1F]
ACK 08:28:01.868 [06]
```

The message sent is formatted according to the following pattern:

Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
01	1E	01	00	01	1F
WRITE_OBJ	4D button	Second	0x0001		

The message stands for “Write to the second 4D button object on the display module the value **0x0001**”.

ACK = 0x06 as shown below

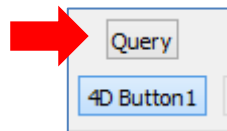
```
ACK 08:28:01.868 [06]
```

is an acknowledgment from the display module which means that it has understood the message.

### Polling a 4D Button

If the 4D button (or any other output object linked to it) was not configured to report a message when its status has changed, it is still possible (although sometimes not recommended for practical applications) to know its current status (enabled or disabled) by polling.

To do be able to do this, click on the Query button.



Messages are sent to and received from the display module.

```
Request 4Dbutton Value 08:40:45.001 [00 1E 01 1F]
4Dbutton Value 08:40:45.027 [05 1E 01 00 00 1A]
```

The messages are formatted according to the following pattern:

Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
00	1E	01	-	-	1F
READ_OBJ	4D button	Second	N/A		
05	1E	01	00	00	1A
REPORT_OBJ	4D button	Second	0x0000		

The host sends a READ\_OBJ command specifically asking for the value (or status) of the second 4D button object. The display module then responds with the current value of that 4D button. Communication between a 4D display module programmed with a ViSi-Genie application and an external host controller must follow the ViSi-Genie Communications Protocol, which is defined in the [ViSi Genie Reference Manual](#).



## Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

## Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.