APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi Genie Pin Input and Output for Picaso Display Modules

DOCUMENT DATE: **13th 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.
- A breadboard, two 1 kilo-ohm resistors, a 10 kilo-ohm resistor, two LEDs, a 100 nano-Farad ceramic capacitor, and jumper wires

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 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 pin input and pin output widgets.

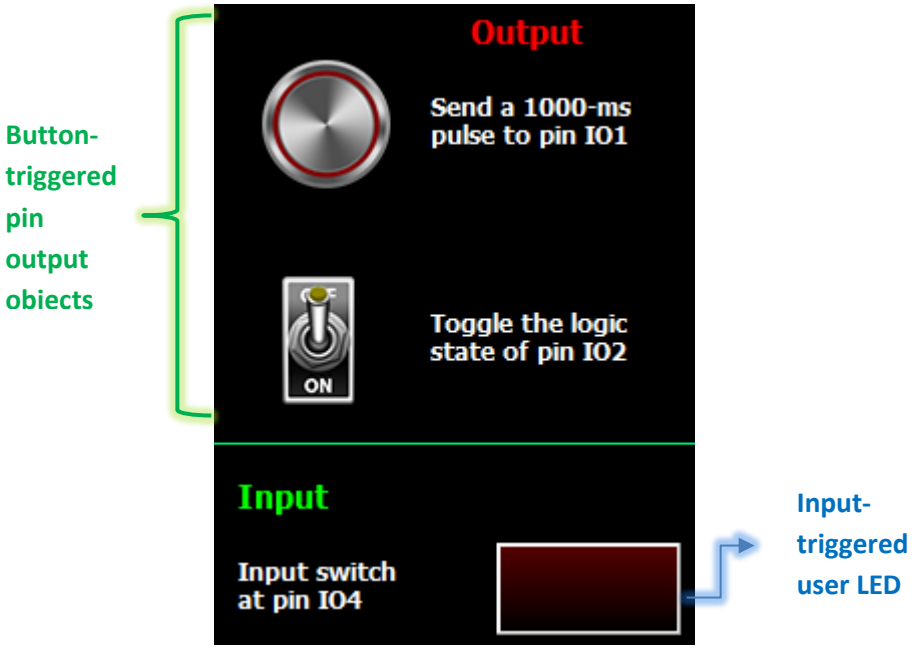
I/O WIDGETS

Pin Input **Pin Output**

With the addition of these I/O (Input/Output) widgets in Workshop, the user now has the option of utilizing the GPIO (General Purpose Input and Output) pins of the display module in the ViSi-Genie environment. The user can read the status of a specific pin when it's configured as a pin input object. Conversely, the user can control the state of a pin when it's configured as a pin output object. It is not advisable however to configure a pin as both an input and an output as undesirable results may occur.

The project developed in this application note discusses the basics of using pin input and pin output objects. As shown in the figure to the right, the project uses buttons to trigger the pin output objects. Pin output objects can either be pulsed or non-pulsed. For the pin input object, a user LED is used

as an indicator. The pin input and output objects are 'invisible' and they always reside in Form0.



A simple circuit is needed for testing this project. The section **"Create a Circuit for Testing the Project"** shows the schematic diagram for the circuit and how to connect it to the display module.

The section **"Identify the Messages"** discusses the format of I/O-widget-related messages 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.

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)

Create a New Project

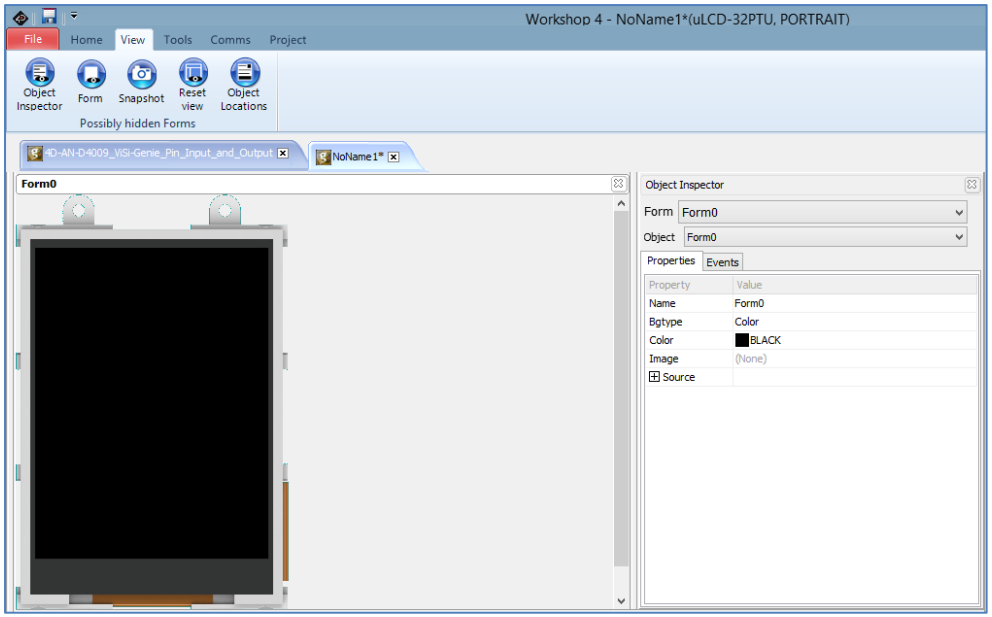
Create a New Project

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)

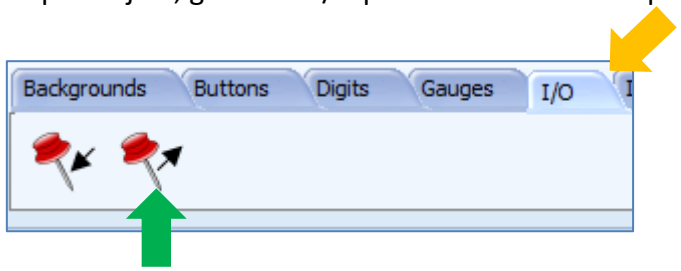
Design the Project

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. A form can contain **widgets** or **objects** like trackbars, sliders, displays, keyboards, and others.
Below is an empty form.

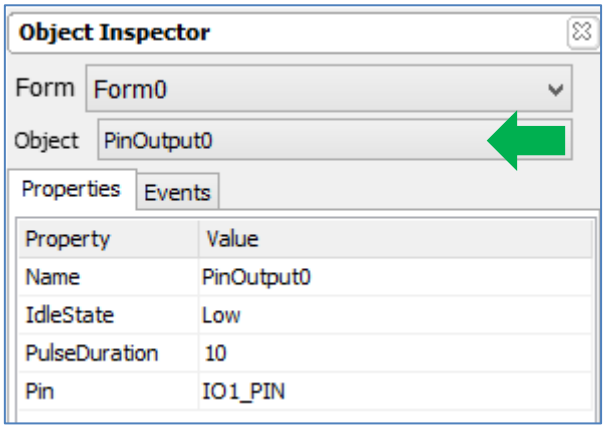


Adding a Pin Output (Pulsed) Object

To add a pin output object, go to the I/O pane and click on the pin output icon.



Click on the WYISWYG screen to place the object. The pin output object always resides in Form0. The object inspector lists all the default properties of the newly-added pin output object named PinOutput0.



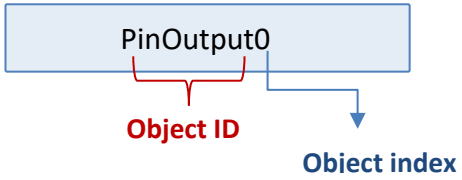
The property **IdleState** defines the logic state of the pin when it's inactive. When **IdleState** is "**Low**", the active state therefore is "**High**". The property **PulseDuration** defines the length of time in milliseconds for which the pin is maintained active. With these settings therefore, when PinOutuput0 is

triggered, pin IO1 will be made active for 10 milliseconds only, then made inactive or idle again until the next occurrence of the triggering event. In other words, a 10-millisecond-pulse is sent to pin IO1 when PinOutput0 is triggered. Increase the pulse duration to 1000 milliseconds.

Note that when the value of **PulseDuration** is set to zero, the pin maintains its current state indefinitely, i.e., it stays active (High in this example) until it is triggered to go idle (Low in this example). Conversely, it stays idle until it is triggered to go active.

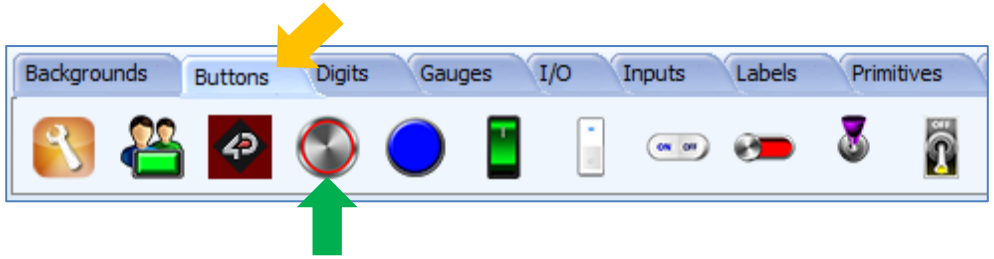
Naming of Objects

Naming is important to differentiate between objects of the same kind. For instance, suppose the user adds another pin output object to the WYSIWYG screen. This object will be given the name PinOutput1– it being the second pin output object in the program. The third pin output object will be given the name PinOutput2, 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.

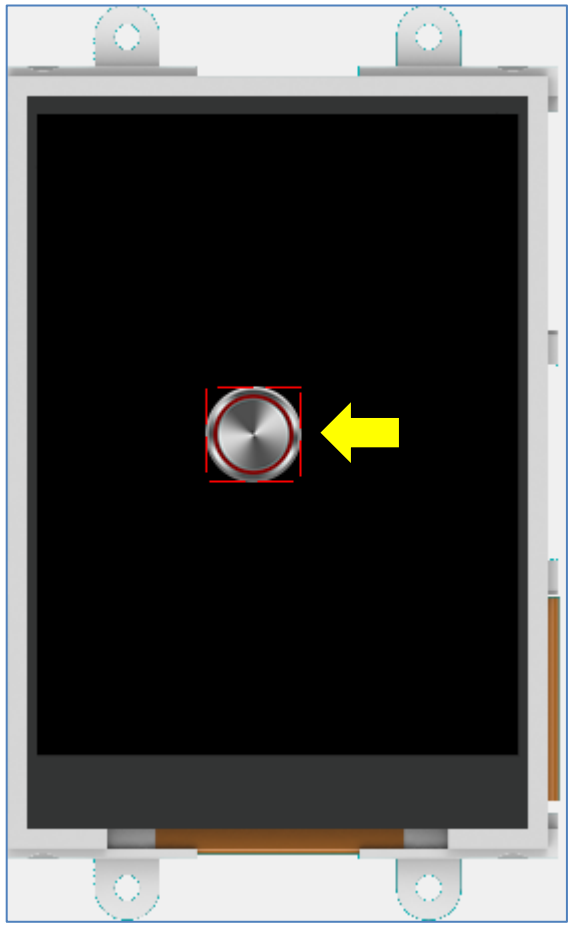


Adding a Momentary 4D Button

A momentary 4D button is now added to be used for triggering PinOutput0. A momentary button is enabled when pressed and disabled upon release. Under the buttons pane, select the Button01 icon.

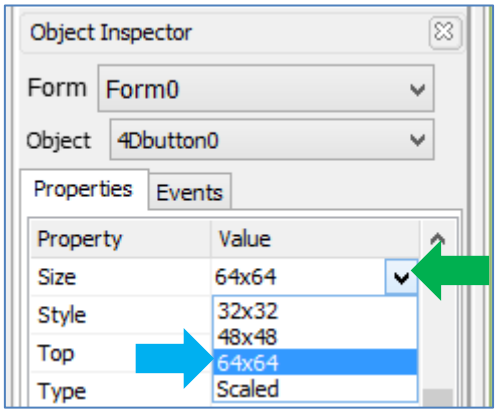


Click on the WYSIWYG screen to place the object.

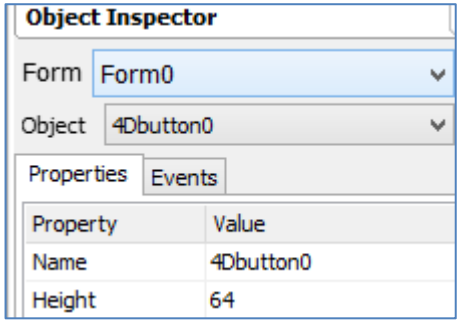


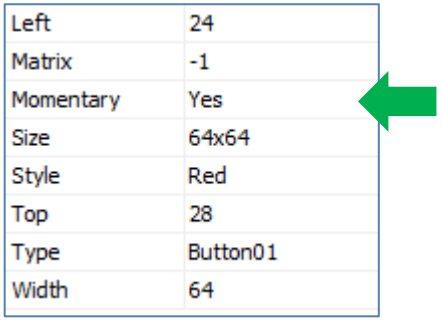
Changing the Size of the Button

The object inspector lists all the properties of the newly-added 4D button object named 4Dbutton0. Change the button size to “64x64” (pixels). To learn more about 4D buttons, refer to ViSi-Genie 4D Buttons.



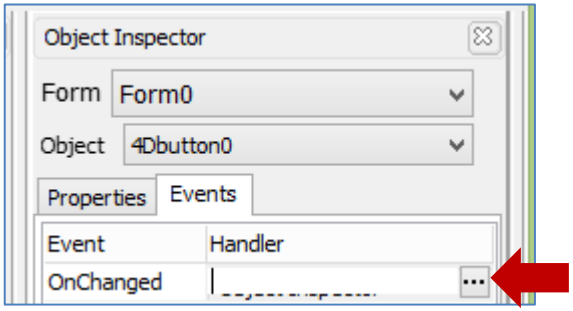
4Dbutton0 now has the following properties. Note that a 4D button is a momentary button by default.



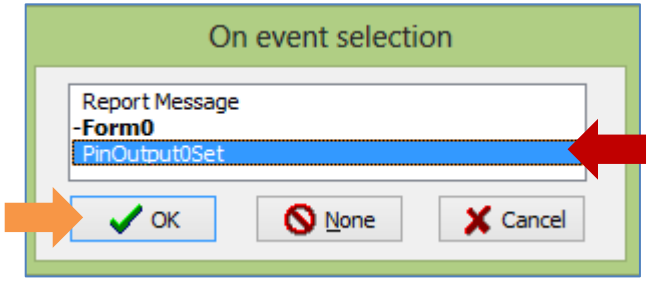


Configuring a 4D Button to control a Pin Output Object

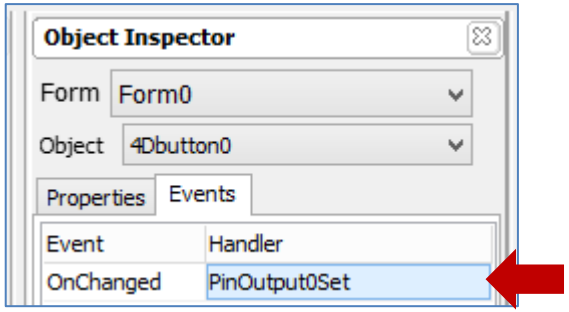
Go to the Events tab and click on the ellipsis dots.



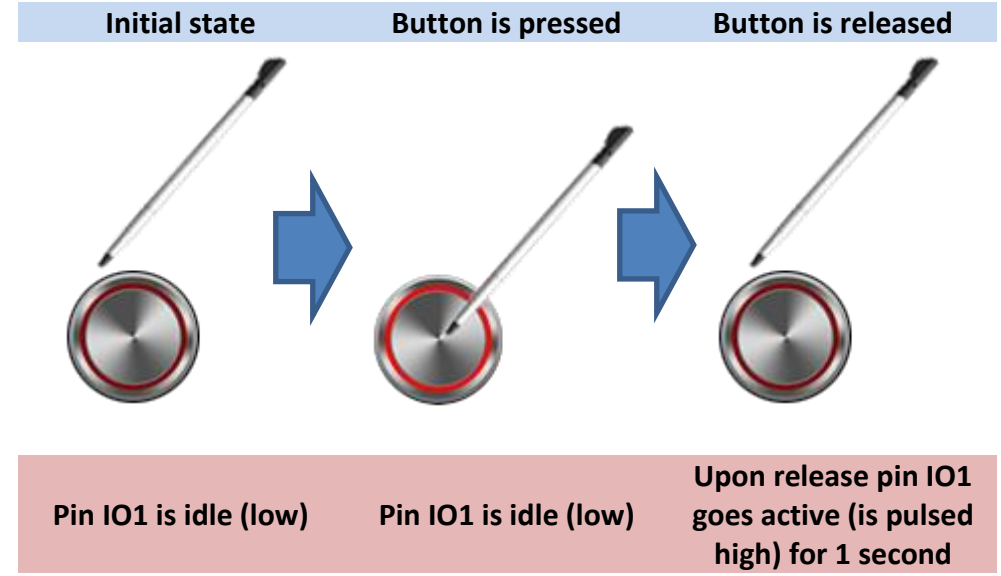
The On event selection window appears. Choose PinOutput0Set then click OK.



The OnChanged event property of 4Dbutton0 is updated accordingly.

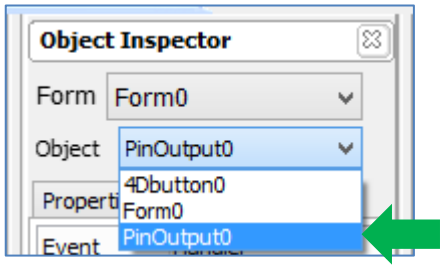


When the program runs, pressing and releasing 4Dbutton0 will send a 1000-millisecond-pulse to pin IO1. To illustrate:

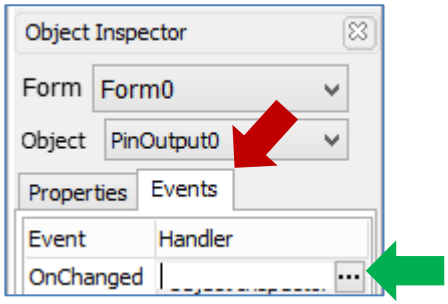


Configuring a Pin Output Object to Report an Event

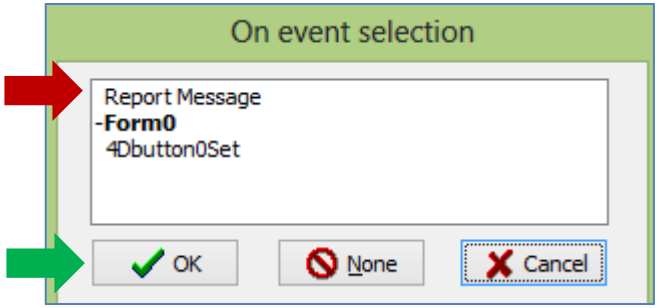
PinOutput0 can be configured to report a message to an external host controller when its status has changed. In the object inspector, select PinOutput0.



Go to the Events tab and click on the ellipsis dots of the OnChanged event.

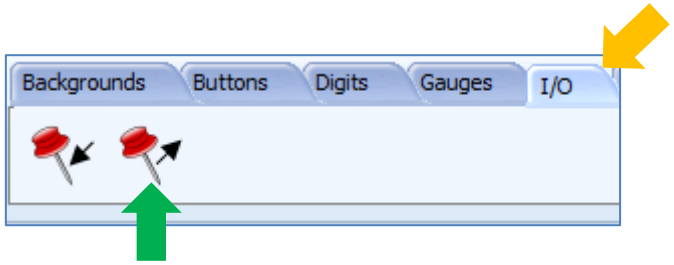


The On event selection window appears. Choose "**Report Message**" and click OK.

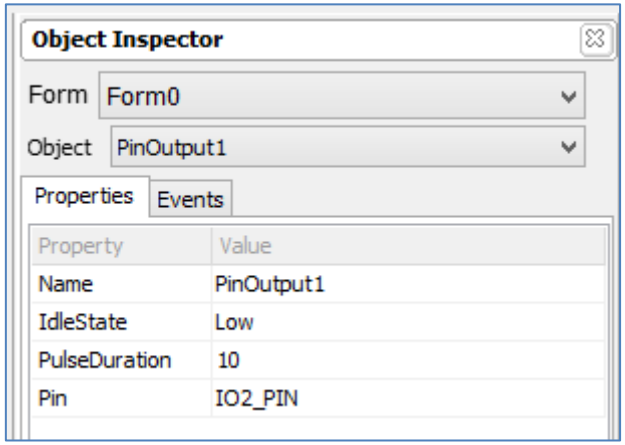


Adding a Pin Output (non-Pulsed) Object

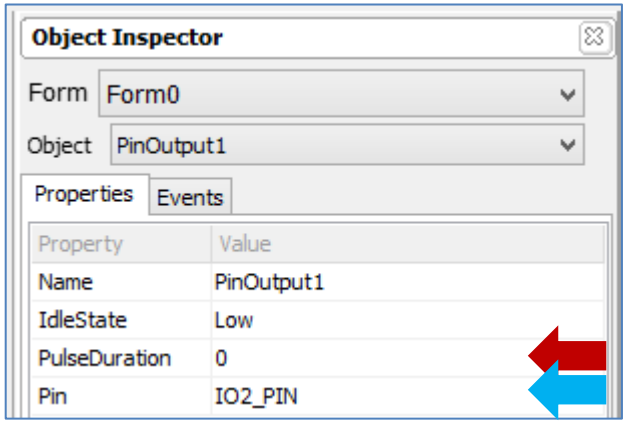
Go to the I/O pane and click on the pin output icon.



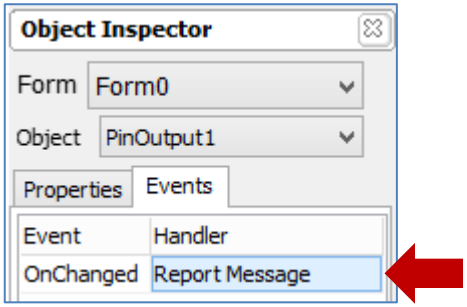
Click on the WYISWYG screen to place the object. The object inspector lists all the default properties of the newly-added pin output object named PinOutput1.



Change the value of PulseDuration to "0" and Pin to IO2. Again, when the value of **PulseDuration** is set to zero, the pin maintains its current state indefinitely.

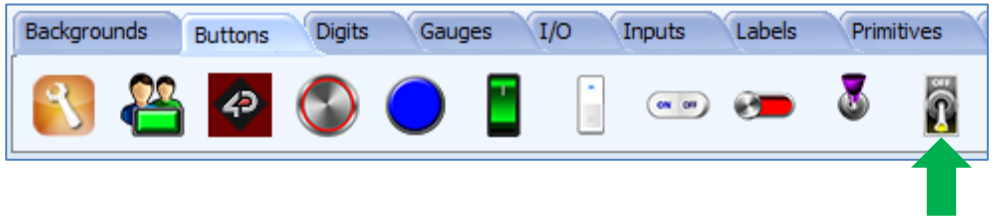


PinOutput1 is now a non-pulsed pin output object unlike PinOutput0. Configure PinOutput1 to report a message when its status has changed.

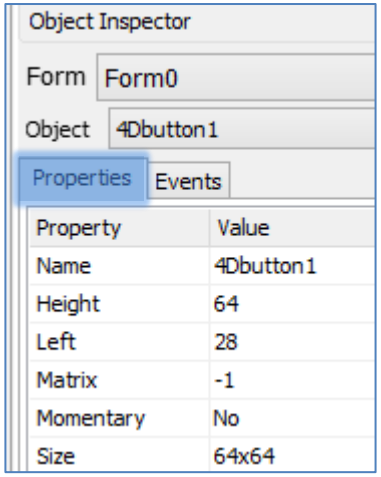


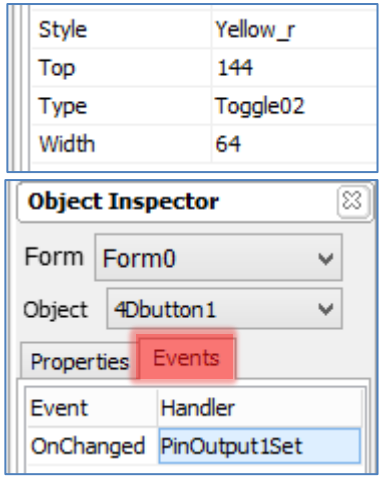
Adding a Toggle 4D Button

A toggle 4D button object is added to trigger PinOutput1. A toggle button is enabled when pressed, and stays enabled when released. Another press-and-release is needed to disable it. Under the buttons pane, select the Toggle02 icon.

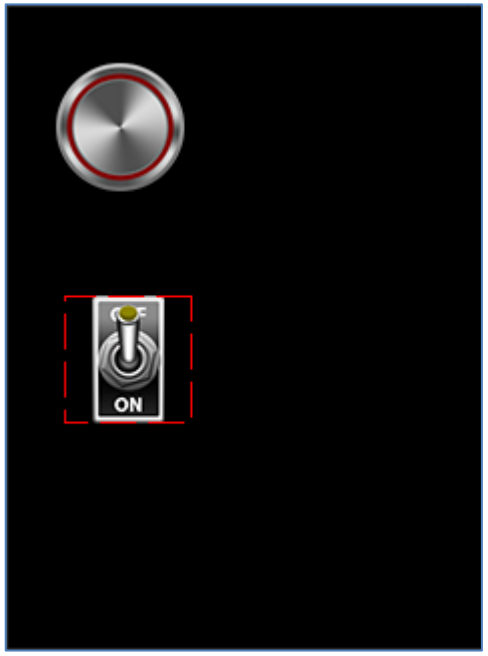


Click on the WYSIWYG screen to place the object. Change the **size** to **64x64** and set the value of the **OnChanged** event property accordingly. Set "**Momentary**" to "**No**". Shown below are the properties of 4dbutton1.

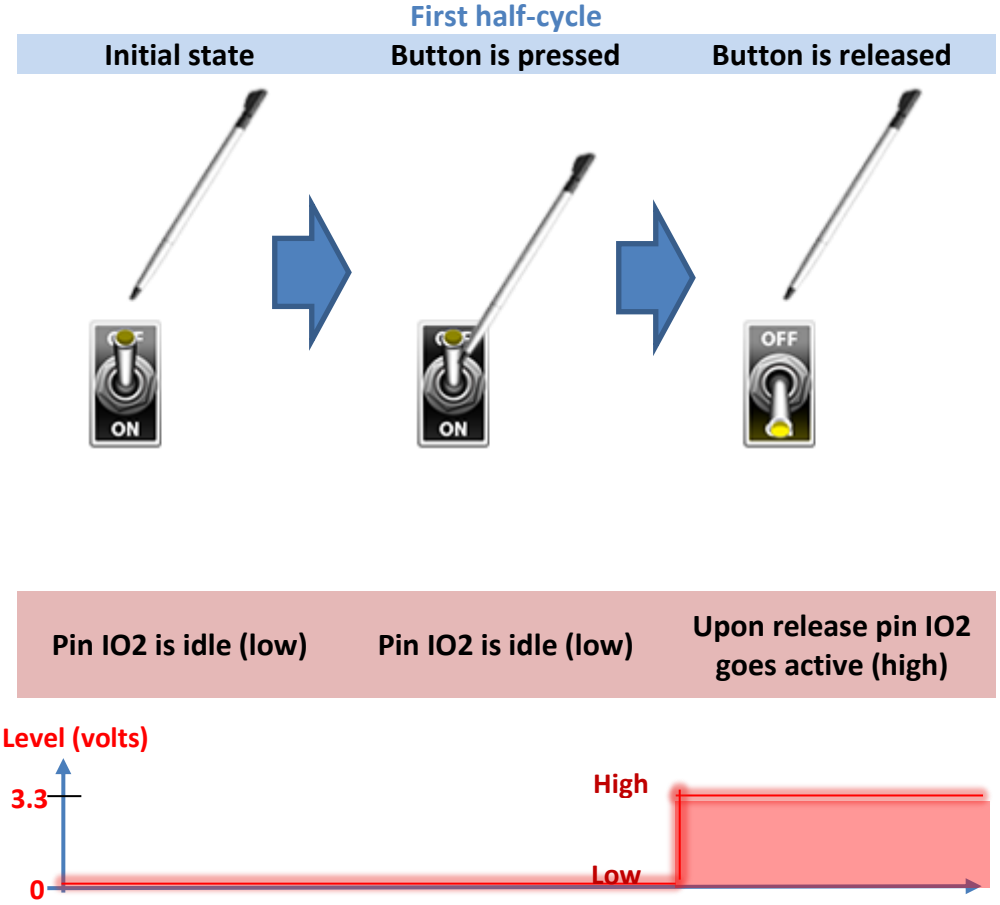


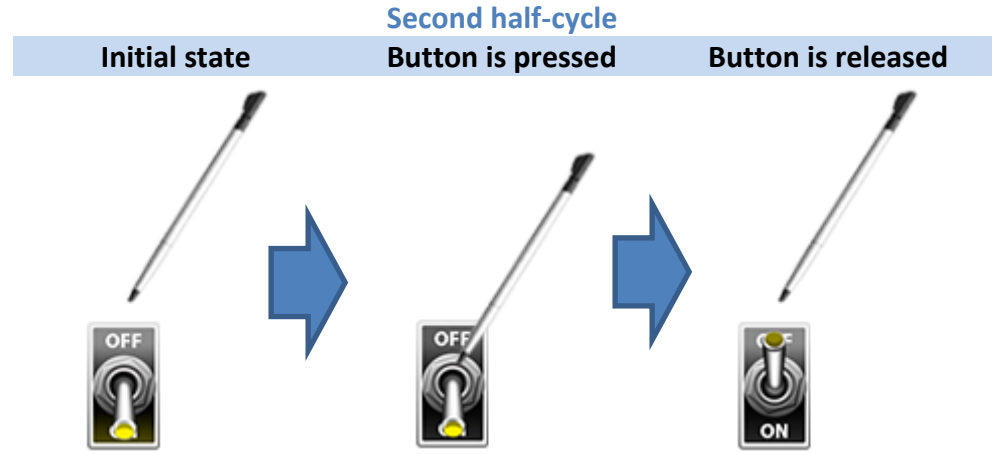


The WYSIWYG screen is updated accordingly.



When the program runs, 4dbutton1 will toggle the logic state of pin IO2. To illustrate:





Pin IO2 is active(high)	Pin IO2 is active(high)	Upon release pin IO2 goes idle (low)

The actual duration of a pulse from a pulsed pin output object is not exactly equal to the value specified by the property **PulseDuration;** it is slightly longer.

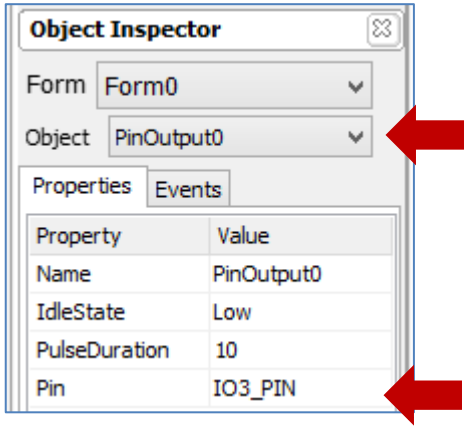
Multiple pin output objects can use the same pin. It is the user's responsibility to manage such usage in a reasonable way.

It makes sense to link a momentary button to a pulsed output pin object only and similarly to link a toggled button to a non-pulsed output pin object only. Of course, when it becomes necessary to do the non-apparent, the user can try experimenting with other combinations.

For some display modules such as the uLCD-32PTU, several of the GPIO pins have a special function. For instance, on page 8 of the datasheet for the uLCD-32PTU (a display module's datasheet can be downloaded from its product page at www.4dsystems.com.au) it says:

> **IO3 pins (Peripheral Supply pin):**
> IO3 is controllable via the processor, or via the H2 Interface pin driven by an external circuit. If IO3 is set as OUTPUT and driven HIGH the µSD and Display are enabled, and disabled when driven LO. Set as INPUT to use an external circuit to drive this pin.

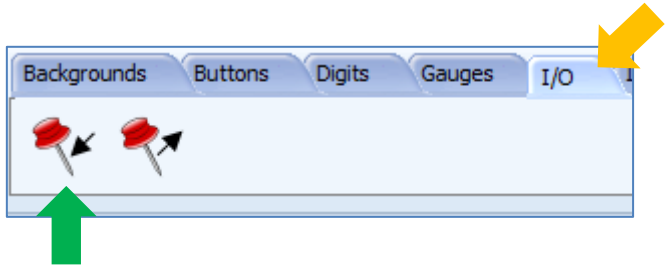
Therefore a program may not run on a uLCD-32PTU when the user has added a pin output object with the value of the property "**Pin**" set as "**IO3_PIN**", as shown below.



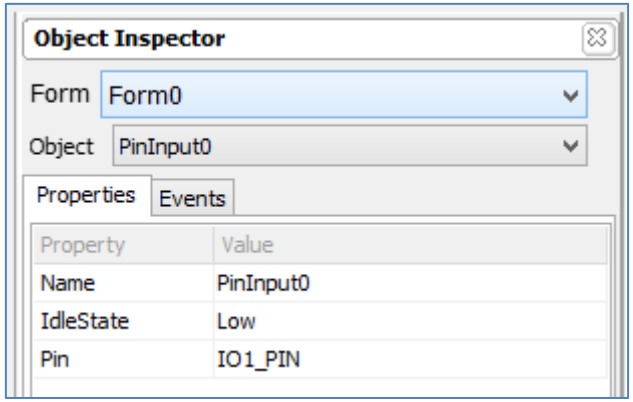
This is because at the start of the program, pin IO3_PIN is already set to low (**IdleState**). This will disable the display and the uSD card. The user has no way of driving the pin high again since the display is turned off. Due to this, it is therefore strongly recommended for users to know the functions of the GPIO pins of a display module first before using them. Again, a display module's datasheet can be downloaded from its product page at www.4dsystems.com.au.

Adding a Pin Input Object

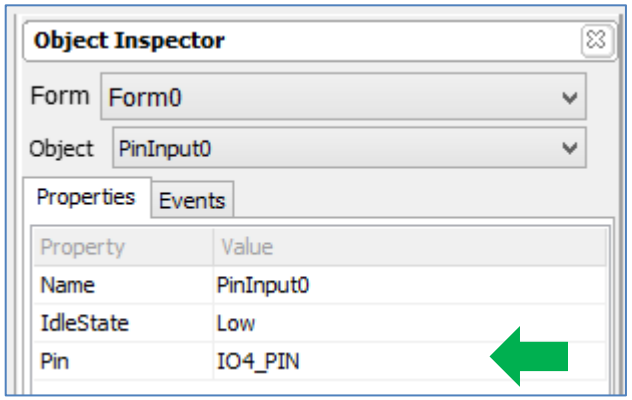
To add a pin input object, go to the I/O pane and click on the pin input icon.



Click on the WYISWYG screen to place the object. The object inspector lists all the default properties of the newly-added pin input object named PinInput0. A pin input object (like a pin output object) will always reside in Form0.

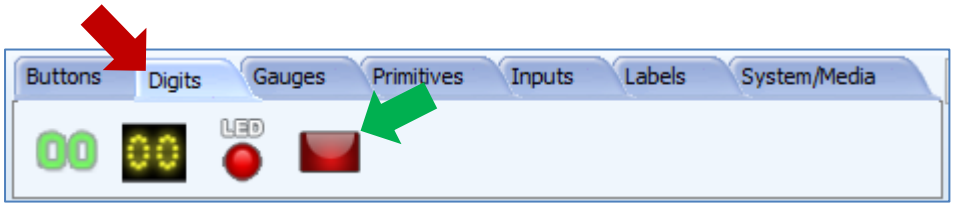


Since pin IO1 was already used for PinOutput0, change the value of **Pin** to **IO4**.

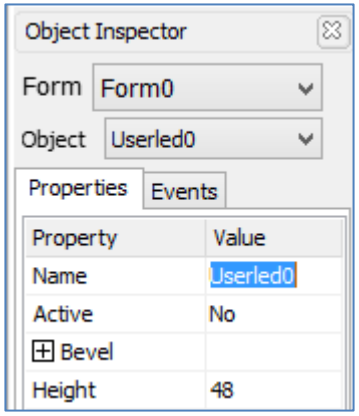


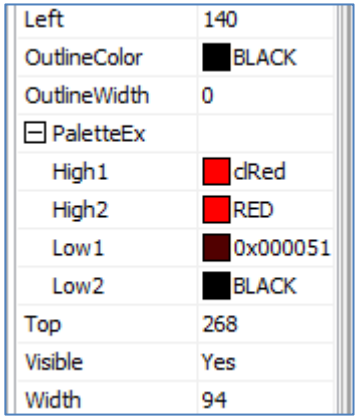
Adding a User LED

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

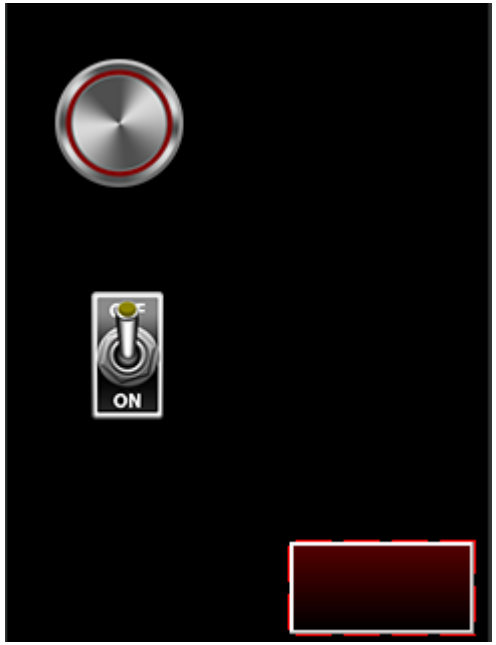


Click on the WYSIWYG screen to place the object. It can be dragged to any location and resized to the desired dimensions. The properties can be edited in the Object Inspector. Userled0 has the following properties:



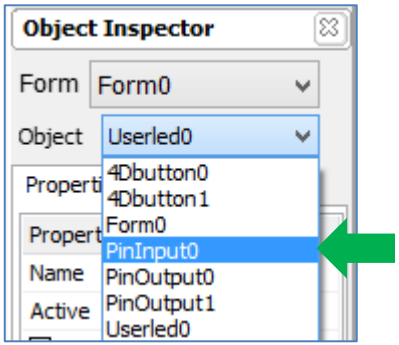


The updated appearance of the WYSIWYG screen is shown below.

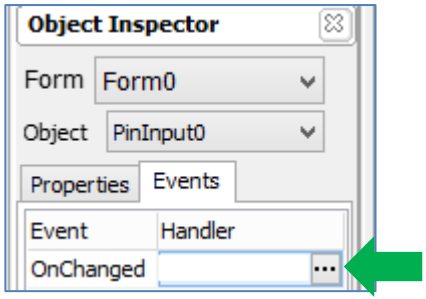


Configuring a Pin Input to Control a User LED

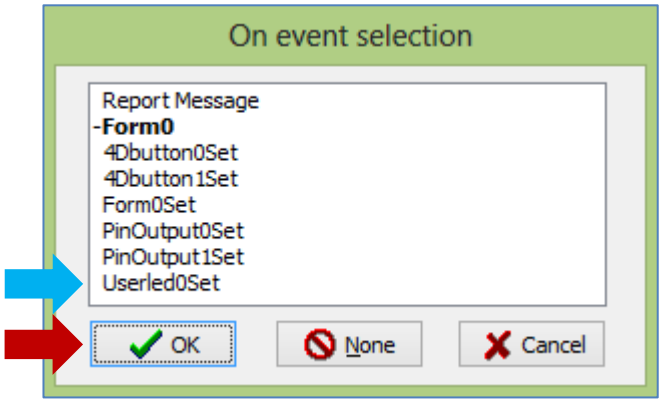
Now Userled0 can be used to indicate the status of pin IO4. In the object inspector, select PinInput0.



Go to the Events tab and click on the ellipsis dots of the OnChanged event.



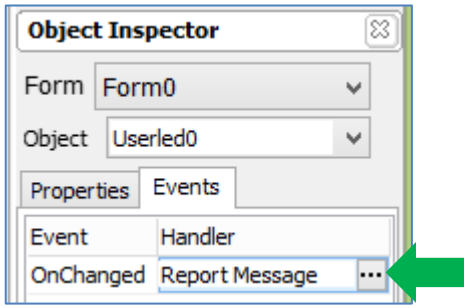
The On event selection window appears. Select Userled0Set and click OK.



When the program runs, the logic state of pin IO4 will dictate the state of Userled0.

Configuring a User LED to Report an Event

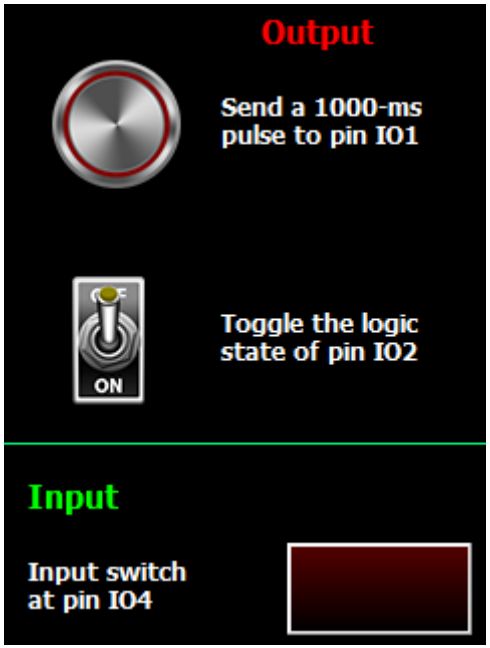
Userled0 can be configured to report a message when its status has changed. Select Userled0 on the WYSIWYG screen and go to the Events tab of its object inspector. Configure the OnChanged event as shown below.



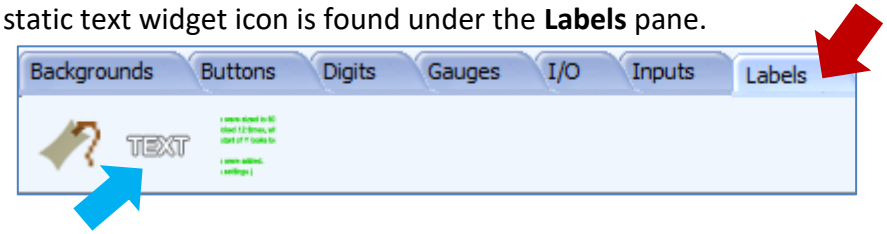
Note: Multiple pin input objects can use the same pin. It is the user's responsibility to manage such usage in a reasonable way. Do not set a pin to both input and output as undesirable result may occur

Completing the Form

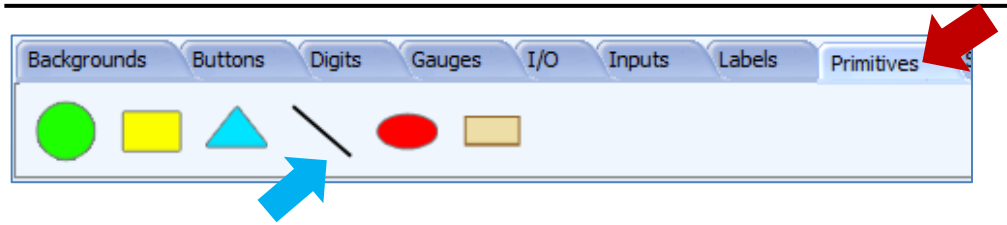
Static text objects are now added to make the project more presentable.



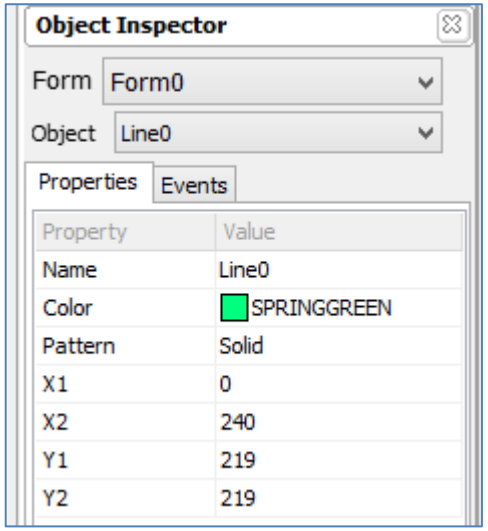
The static text widget icon is found under the **Labels** pane.



Adding a static text object and experimenting with its properties are now left to the user as an exercise. The horizontal line across the form is a line object, the icon of which is found under the **Primitives** pane.



Line0 of the demo project has the following properties.



Build and Upload the Project

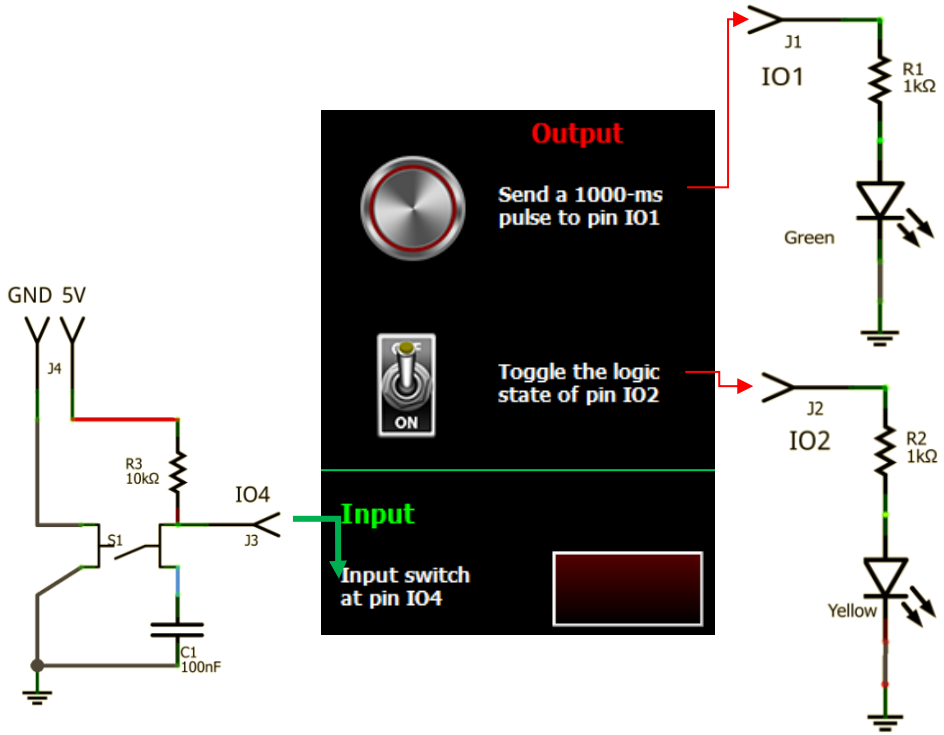
For instructions on how to build and upload a ViSi-Genie project to the target display, please refer to the section "**Build and Upload the Project**" of the application note

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

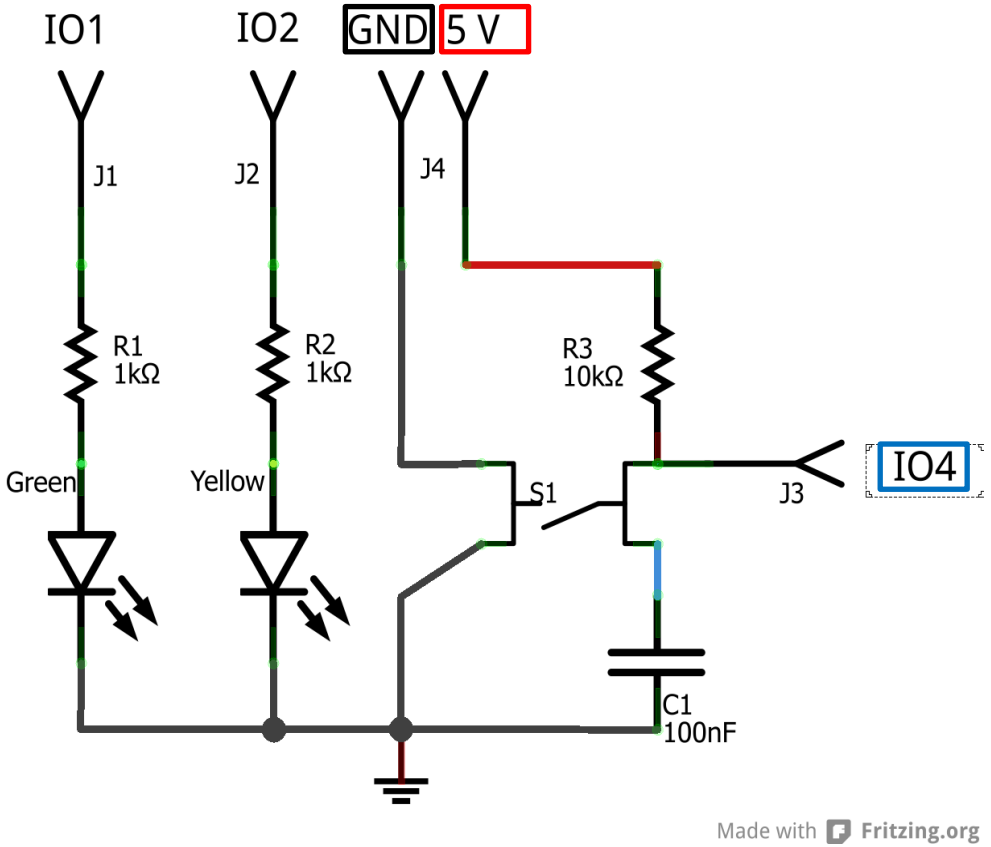
The uLCD-32PTU display module is commonly used as examples, but the procedure is the same for other displays.

Create a Circuit for Testing the Project

Since the project involves the use of a pin input and two pin output objects, it is therefore necessary to create a circuit to test it. For the pin output objects, two LEDs are used to indicate the status of the pins used. For the pin input object, a simple pushbutton switch is used. The following sections now present the hardware part of the project.



Schematic Diagram

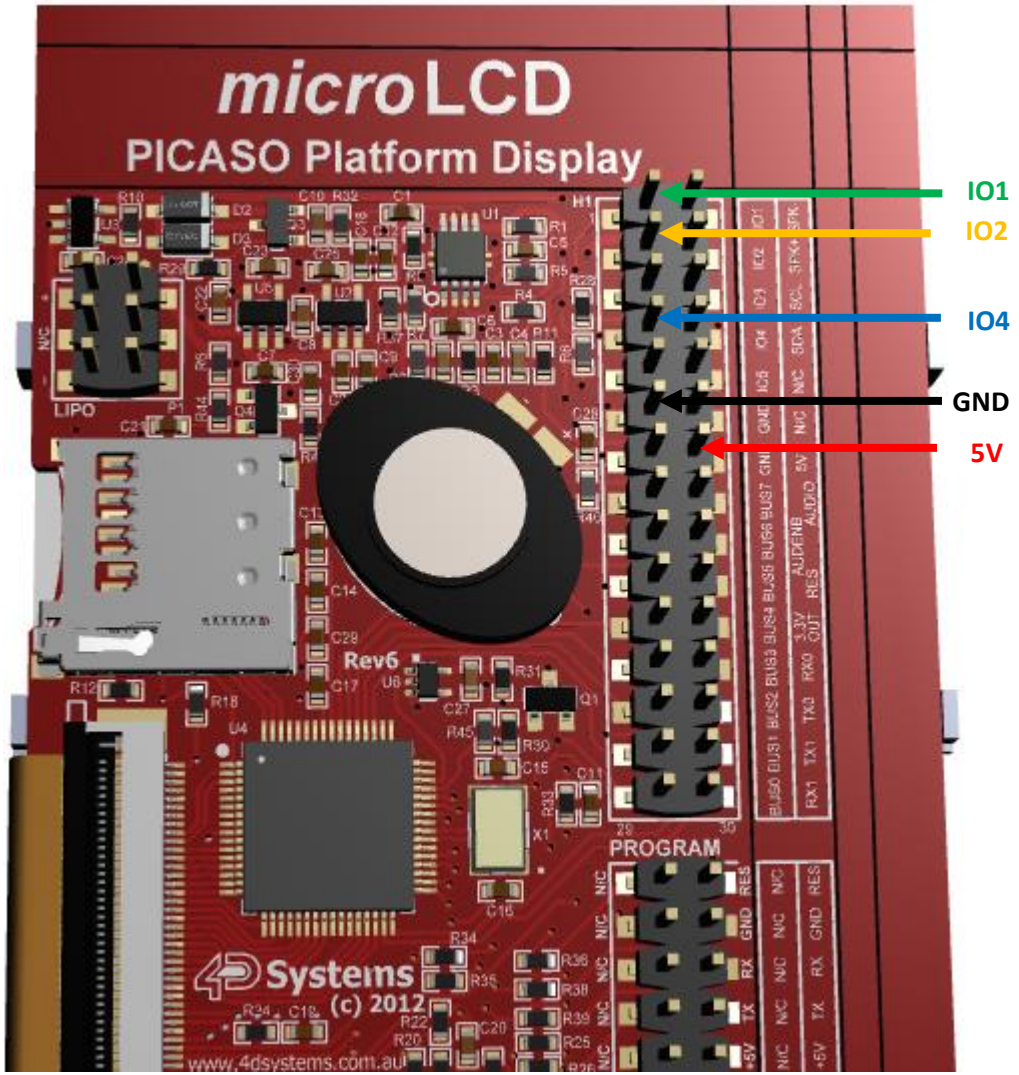


Pin Configuration of the Display Module

The pins IO1, IO2, and IO4 of the uLCD-32PTU can be accessed from the I/O expansion header H1. The datasheet for the uLCD-32PTU can be downloaded from its product page. The pin configuration for header H1 is shown on page 6 (for datasheet revision 1.9).

H1 Pin Outs (I/O Expansion Header)			
Pin	**Symbol**	**I/O**	**Description**
1	IO1	I/O	General Purpose Input Output 1 Pin
2	SPK-	O	Speaker Output –ve, for external Speaker
3	IO2	I/O	General Purpose Input Output 2 Pin
4	SPK+	O	Speaker Output +ve, for external Speaker
5	IO3	I/O	General Purpose Input Output 3 Pin
6	SCL	O	I²C Clock Output
7	IO4	I/O	General Purpose Input Output 4 Pin
8	SDA	I/O	I²C Bidirectional Data
9	IO5	I/O	General Purpose Input Output 5 Pin
10	N/C	-	Not connected
11	GND	P	Supply Ground
12	N/C	-	Not connected
13	GND	P	Supply Ground
14	+5V	P	Supply Input +ve, 4.0V to 5.5V, 5.0V Nominal
15	BUS7	I/O	IO Bus (BUS0..7) bit 7
16	AUDIO	I/O	Audio Input or Output, TTL Line level
17	BUS6	I/O	IO Bus (BUS0..7) bit 6
18	AUDENB	I	Audio Amplified Enable, Active High (5V)
19	BUS5	I/O	IO Bus (BUS0..7) bit 5
20	RES	I	Master Reset, Active Low (GND) (Refer H2 Pinout)
21	BUS4	I/O	IO Bus (BUS0..7) bit 4
22	3.3V OUT	P	3.3V Output, 20mA Max supply
23	BUS3	I/O	IO Bus (BUS0..7) bit 3
24	RX0	I	Asynchronous serial port 0 receive pin. COM0 (same as the RX pin on the H2 Programming Header)
25	BUS2	I/O	IO Bus (BUS0..7) bit 2
26	TX0	O	Asynchronous serial port 0 transmit pin. COM0 (same as the TX pin on the H2 Programming Header)
27	BUS1	I/O	IO Bus (BUS0..7) bit 1
28	TX1	O	Asynchronous serial port 1 transmit pin. COM1
29	BUS0	I/O	IO Bus (BUS0..7) bit 0
30	RX1	I	Asynchronous serial port 1 receive pin. COM1

Connect the circuit to the display module accordingly.



Identify the Messages

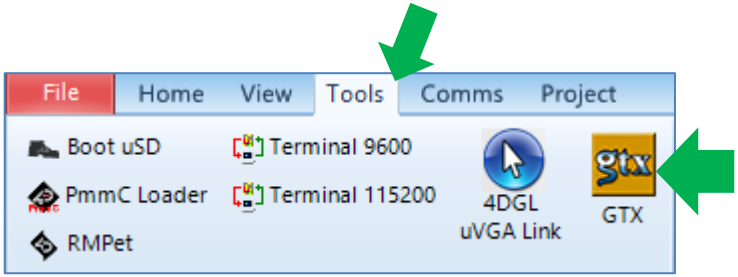
The display module is going to receive and send messages from and to an external host. This section explains to the user how to interpret these messages. An understanding of this section is necessary for users who intend to interface the display to a host. The ViSi Genie Reference Manual is recommended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

Use the GTX Tool to Analyse the Messages

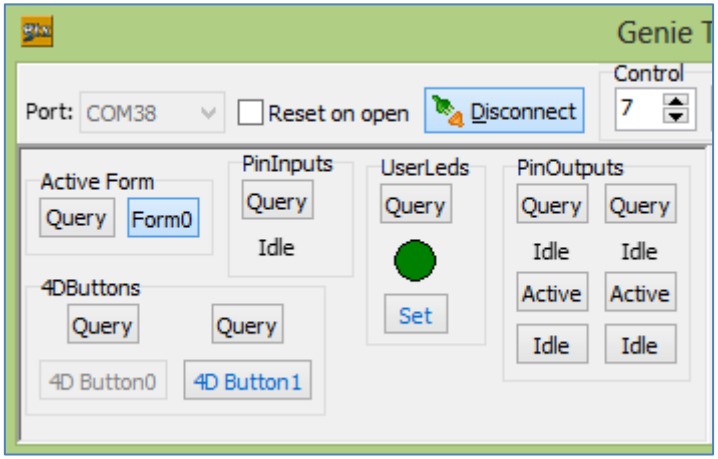
Using the GTX or **Genie Test eXecutor** tool is the first option to get the messages sent by the screen to the host. Here the PC will be the host. The GTX tool is a part of the Workshop 4 IDE. It allows the user to receive, observe, and send messages from and to the display module. It is an essential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

Launch the GTX Tool

Under the Tools menu click on the GTX tool button.



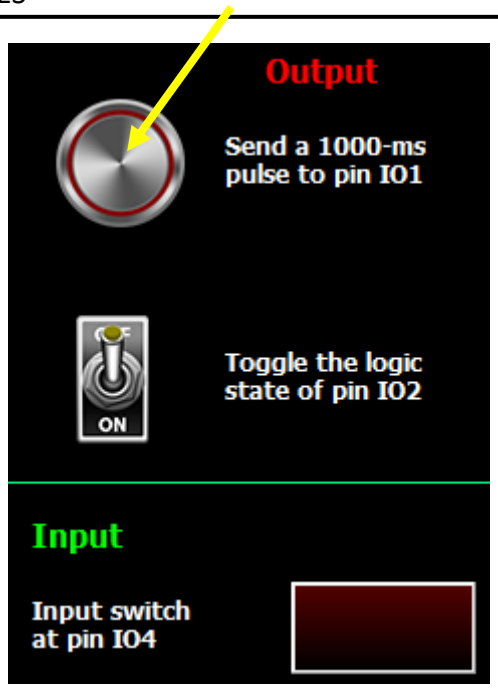
A new window appears showing all the objects of the project.



The Pin Output Object

Report Event

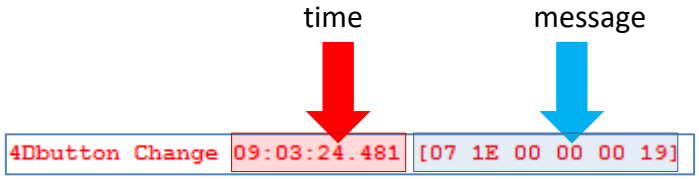
When the program starts, press and release 4Dbutton0 on the display module screen. Linked to 4Dbutton0 is PinOutput0. Remember that PinOutput0 was configured earlier to report a message when its status has changed.



Upon releasing 4Dbutton0, the green LED should light up for one second (approximate). Also, a message is sent from the display module to the PC. The message is displayed on the white area on the right part of the GTX Tool window.

```
4Dbutton Change 09:03:24.481 [07 1E 00 00 00 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:

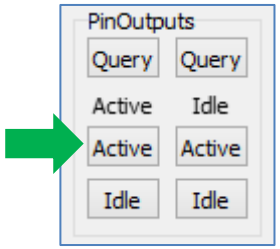
Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
07	1E	00	00	00	19
REPORT_EVENT	4D button	first	0x0000		

The message is from 4Dbutton0, and it contains the hexadecimal value “0x0000”. The value (MSB and LSB) contained in a message coming from a button reflects its most current status. A momentary button normally sends the value “0x0000” only, which means that it has just been released (i.e., its latest state is “disabled”).

Note : Although PinOutput0 has been configured to report a message when its status has changed, the GTX tool window message area labels the message as coming from a 4D button. This is because only input objects such as 4D button can initiate an event. Output objects such as a pin output cannot initiate an event. To learn more about the input-output classification of Genie objects, refer to section 7 of the ViSi-Genie User Guide

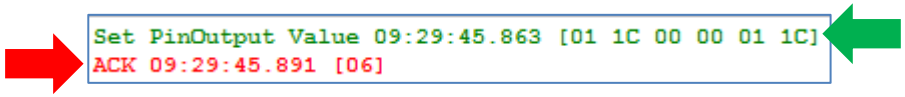
Control a Pin Output Object using the GTX Tool

In the GTX tool window, click on the “Active” button shown below.



Note that the green LED lights up for one second. Also, the GTX tool window message area displays:

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



The message sent is formatted according to the following pattern:

Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
01	1C	00	00	01	1C

WRITE_OBJ	Pin output	First	0x0001	

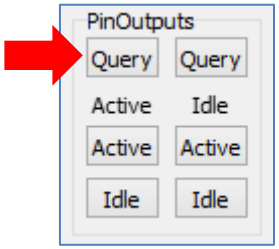
The message stands for “Write to the first pin output object on the display module the value **0x0001**”.

ACK = 0x06 as shown below

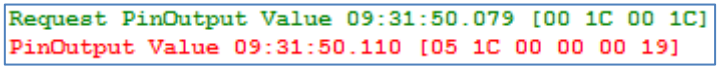
is an acknowledgment from the display module which means that it has understood the message.

Polling a Pin Output Object

If a pin output object was not configured to report a message when its status has changed, it is still possible to know its current status by polling. To do this, click on the Query button.



Messages are sent to and received from the display module.



The messages are formatted according to the following pattern:

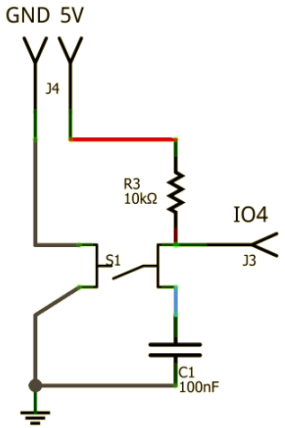
Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
00	1C	00	-	-	1C
READ_OBJ	Pin output	First	N/A		
05	1C	00	00	00	19
REPORT_OBJ	Pin output	First	0x0000		

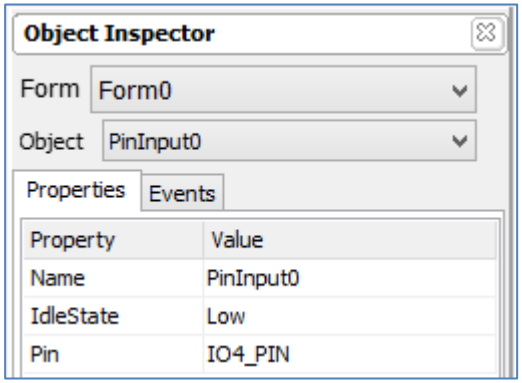
The host sends a READ_OBJ command specifically asking for the value or status of the first pin output object. The display module then responds with the value of that pin output object.

The Pin Input Object

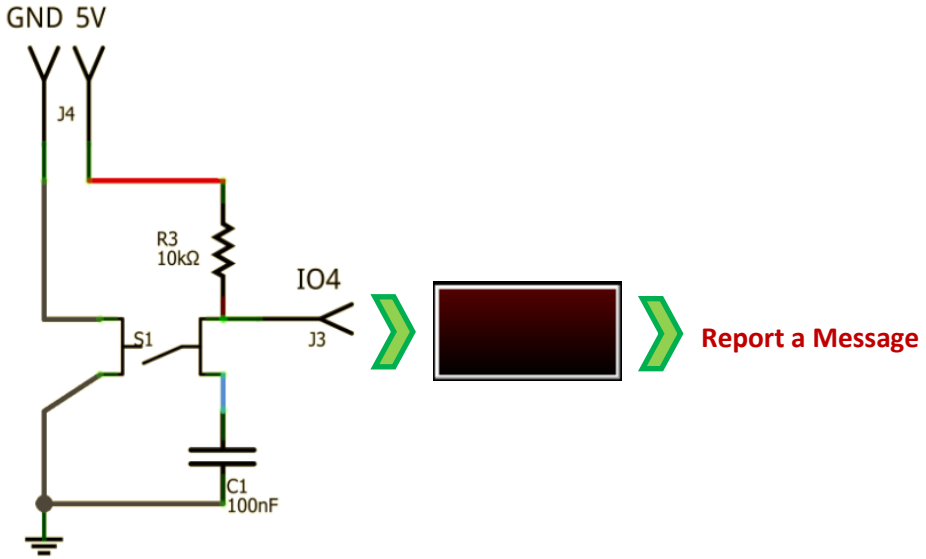
Report Event

Looking at the pushbutton switch circuit, the voltage level at pin IO4 when the switch is open is 5 volts (HIGH). When the switch is closed, the voltage at pin IO4 goes down to 0 volt (LOW).





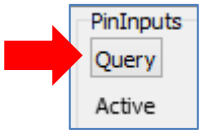
Userled0 is linked to PinInput0. Also remember that Userled0 was configured earlier to report a message when its status has changed. To illustrate:



When the pushbutton is switched on and then off, the following messages are displayed on the GTX tool window.

```
PinInput Change 10:00:05.095 [07 1D 00 00 00 1A]
PinInput Change 10:00:05.816 [07 1D 00 00 01 1B]
```

The object ID of a pin input is 0x1D. See section 3.3 of the ViSi Genie Reference Manual for a full list of object IDs. The user can also poll the display for the current status of the pin input object.



The following messages are displayed on the GTX tool window.

```
Request PinInput Value 10:03:48.013 [00 1D 00 1D]
PinInput Value 10:03:48.045 [05 1D 00 00 01 19]
```

Again, note that although it is Userled0 that was configured to report a message when its status has changed, the GTX tool window message area labels the message as coming from a pin input object. A pin input object is classified as an input; a user LED is classified as an output.

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

Disclaimer of Warranties & Limitation of Liability