APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi Genie Single Trace Scope

DOCUMENT DATE: **15th 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.

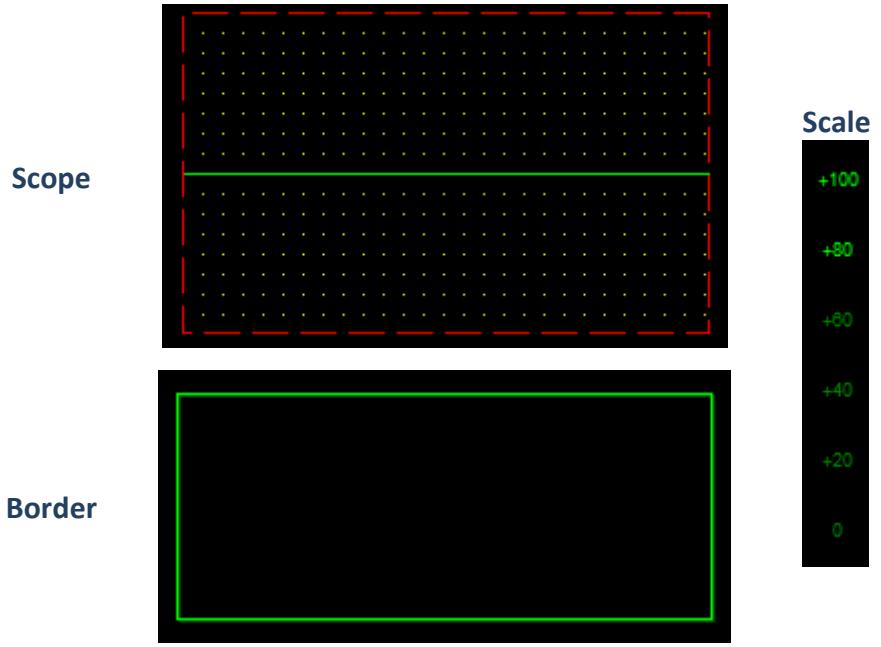
Before getting started, the following are required:

- 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

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. The following are examples of widgets used in this application note.



The project developed in this application note demonstrates the basic use of the scope (single trace). The scope can display a maximum of four signal

traces on the screen. It is designed to be driven by an external host controller. A section of this application note discusses the format of the messages for controlling the scope using the GTX Tool in Workshop.

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

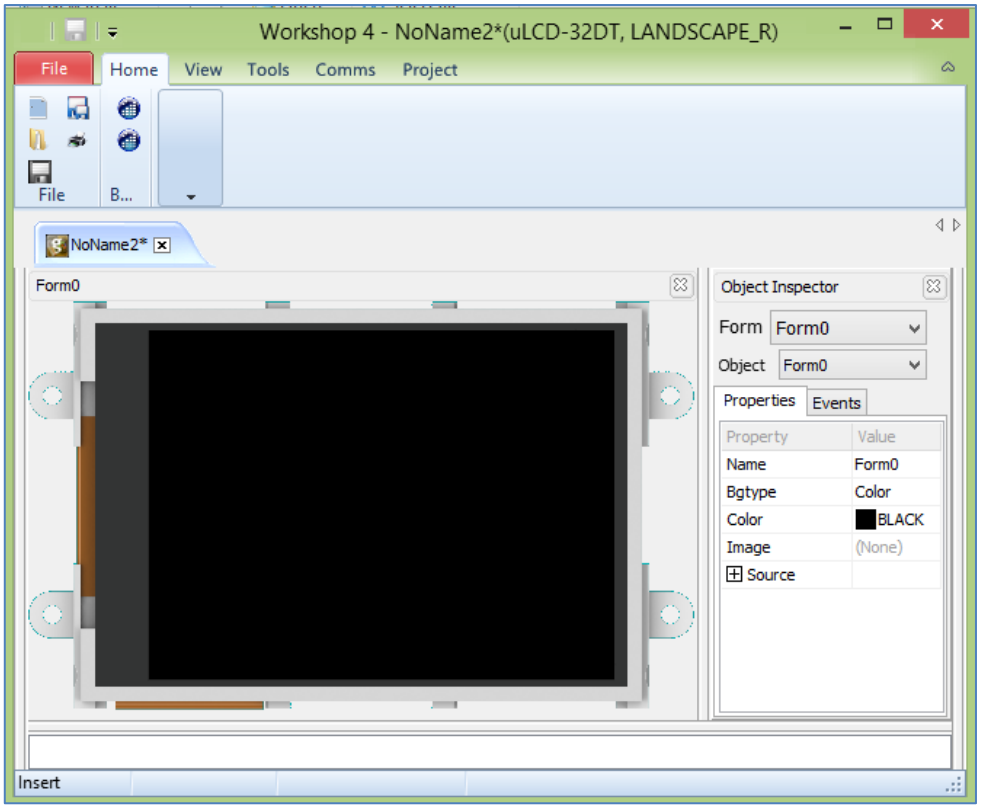
Create a New Project

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

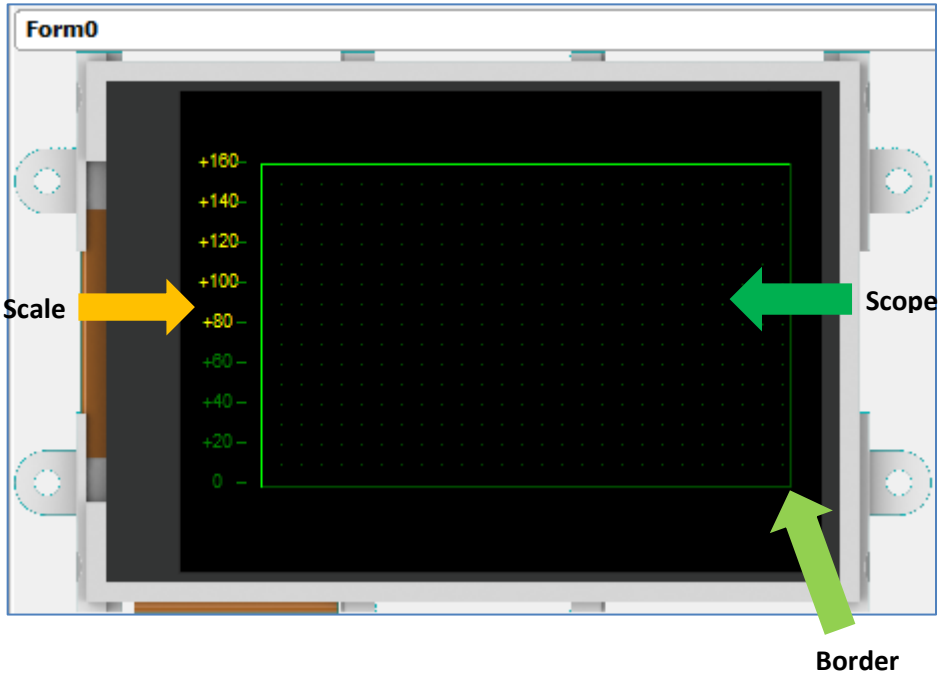
ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso)
or
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16).

Design the Project

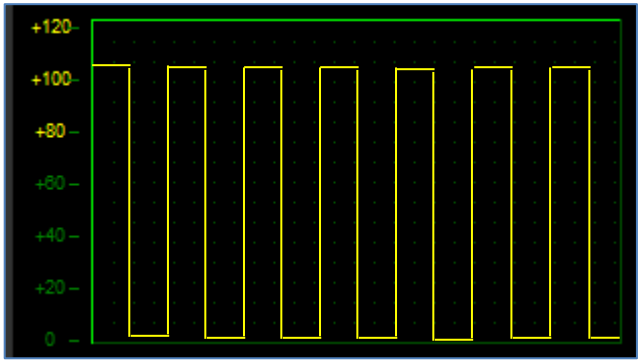
Everything is now ready to start designing the project. **Workshop 4** displays an empty screen, called **Form0**. A **form** is like a page on the screen. The form can contain **widgets** or **objects**, like trackbars, sliders, displays or keyboards. Below is an empty form.



At the end of this section, the user will able to create a form with three objects. The final form will look like as shown below, excluding the labels.



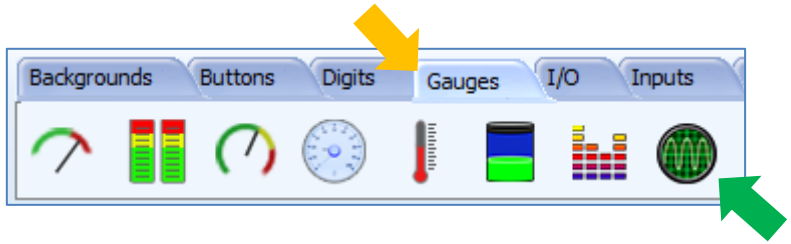
When the program runs, the display module will look like as shown below.



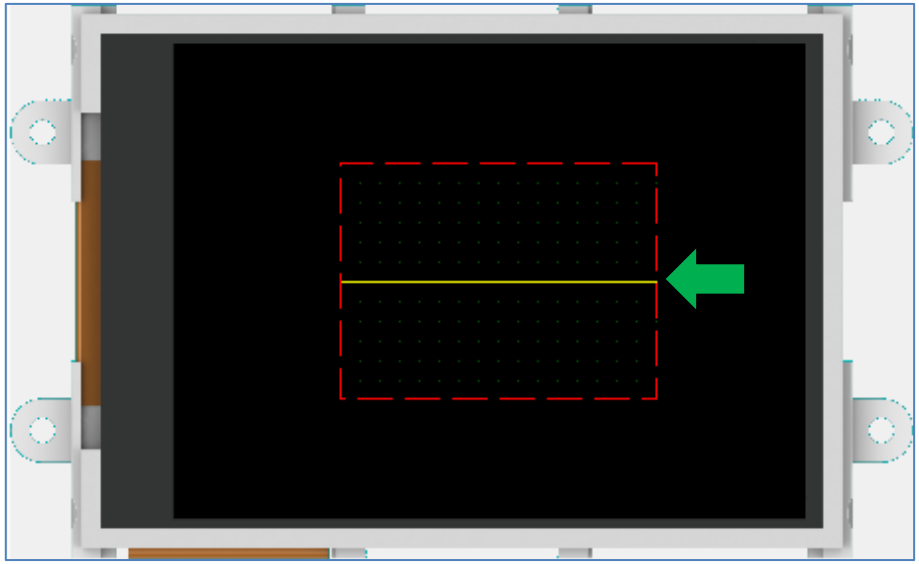
The procedure for adding each of these objects will now be discussed.

Adding a Scope

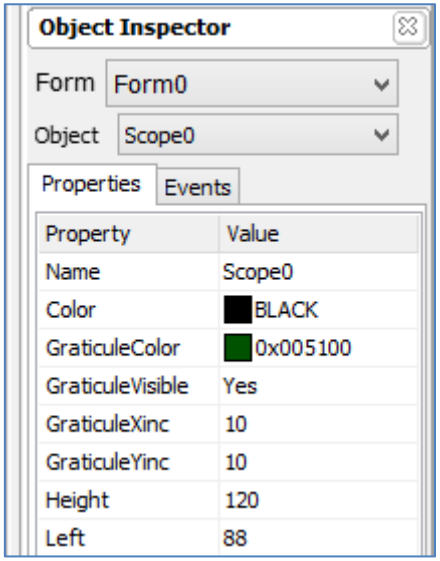
To add a scope, go to the **Gauges** pane then click on the **scope** icon.

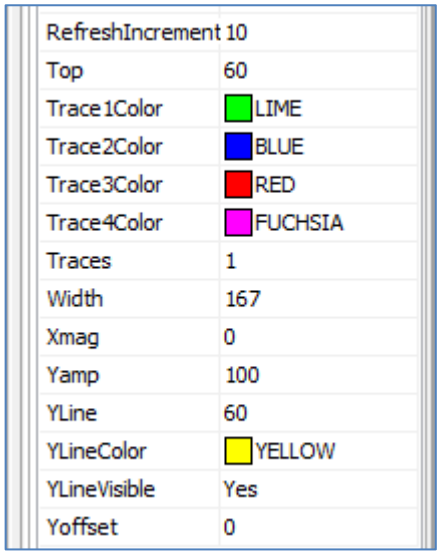


Click on the **WYSIWYG** (What-You-See-Is-What-You-Get) screen to put the object in place. The WYSIWYG screen simulates the actual appearance of the display module screen.

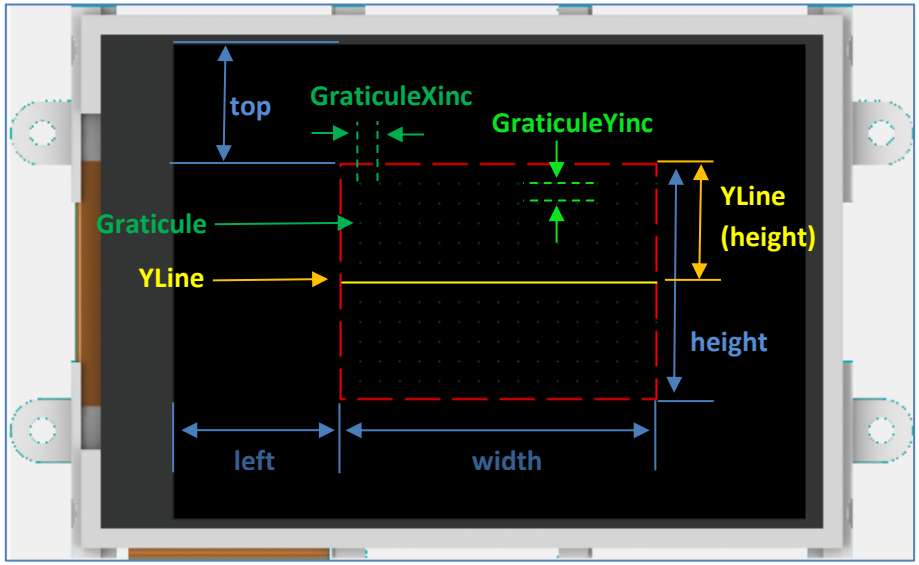


The object can be dragged to any desired location and resized to the desired dimensions. The **Object Inspector** on the right part of the screen displays all the properties of the newly created scope object named **Scope0**.

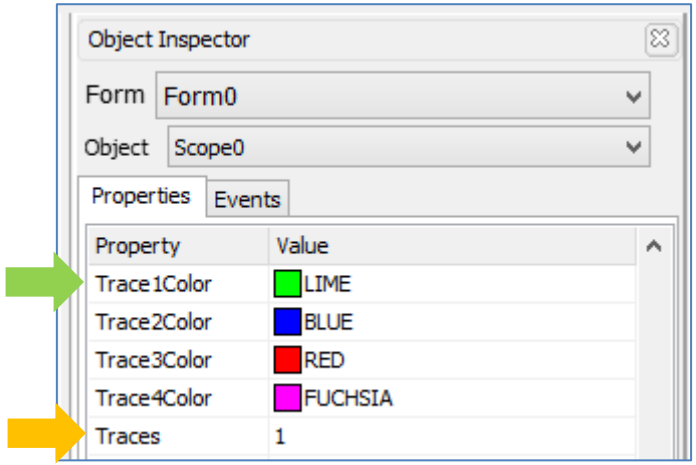




Take time to experiment with the different properties. The image below illustrates the definition of some of the properties.

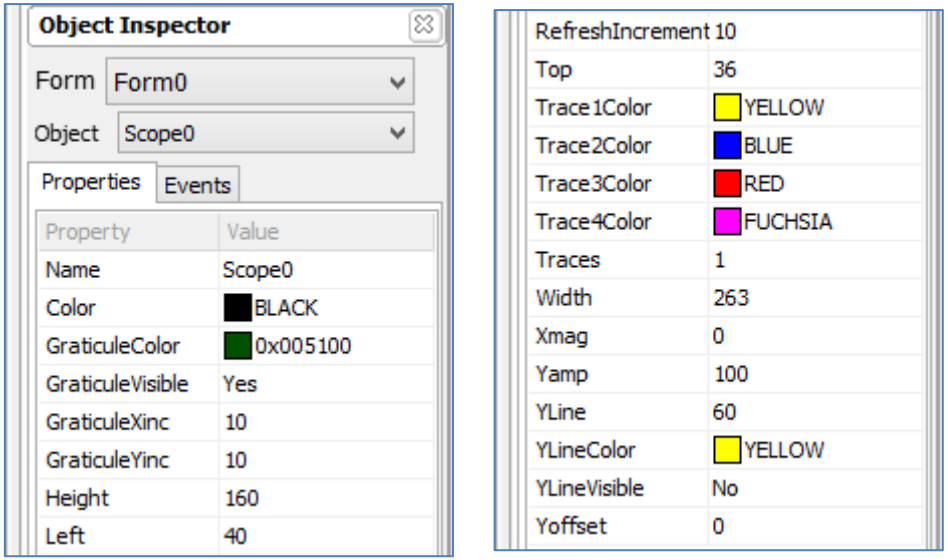


In the object inspector, try altering each of the properties indicated in the image above and observe the effects on the WYSIWYG screen. When the program runs, a single signal will be traced across the scope. The number of traces in a scope and their colours are specified by the properties below.

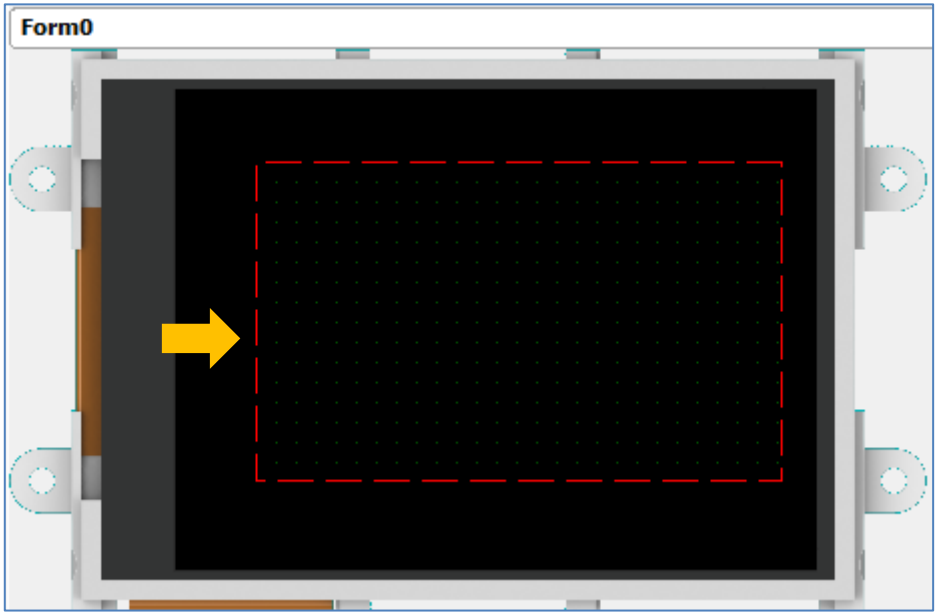


There are a maximum of four traces, the colours which are defined by the **TraceXColor** property. In the example above, only a single trace will be displayed (lime in colour). The other properties and their purpose will be explained later.

The scope used in this project has the following properties.

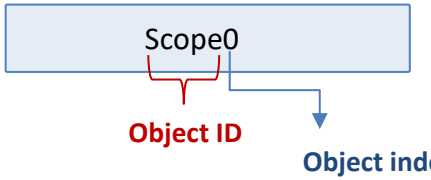


The updated appearance of the WYSIWYG screen is shown below.



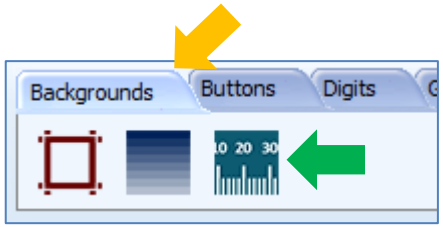
Naming of Objects

Naming is important to differentiate between objects of the same kind. For instance, suppose the user adds another scope object to the WYSIWYG screen. This object will be given the name Scope1 – it being the second scope object in the program. The third scope will be given the name Scope2, 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.

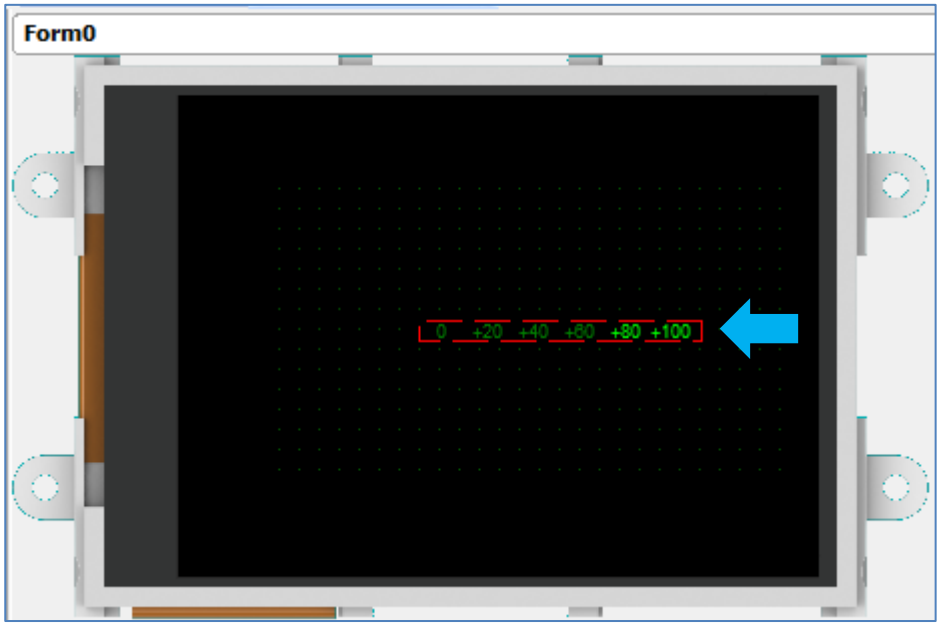


Adding a Scale

To add a scale, go to the **Backgrounds** pane and click on the **scale** icon.



Click on the WYSIWYG screen to place the object.



The **Object Inspector** on the right part of the screen displays all the properties of the newly created scale named **Scale0**. The scale object used in this project has the following properties.

Object Inspector

Form: Form0

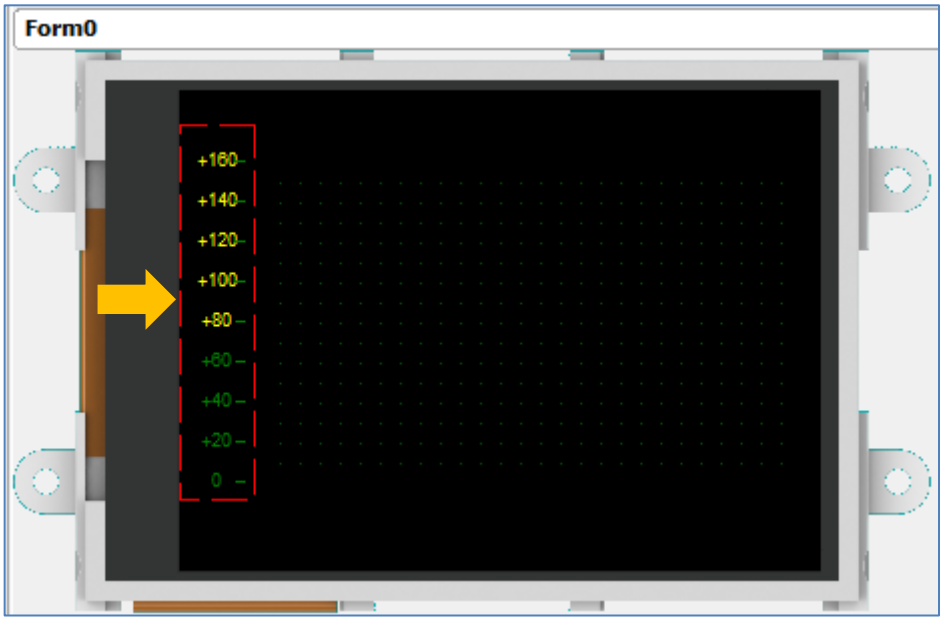
Object: Scale0

Properties | Events

Property	Value
Name	Scale0
Alignment	Center
Color	BLACK
Digits	3
Font	(GREEN, [], Arial, 7, [])
Height	188
Layout	Center
LeadingZero	No
Left	0

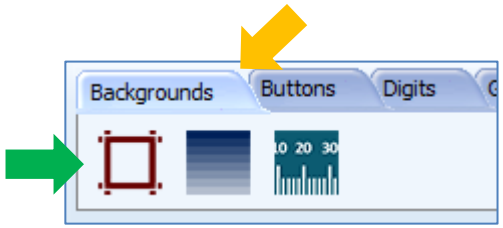
Maxvalue	160
Minvalue	0
Orientation	Vertical
PeakColor	YELLOW
PeakLevel	80
Scalecolor	GREEN
ScaleOffset	5
ShowSign	Yes
TickMarks	BottomRight
Ticks	8
TicksHeight	5
TicksWidth	1
Top	17
Transparent	No
Width	38

Shown below is the final appearance of the scale.

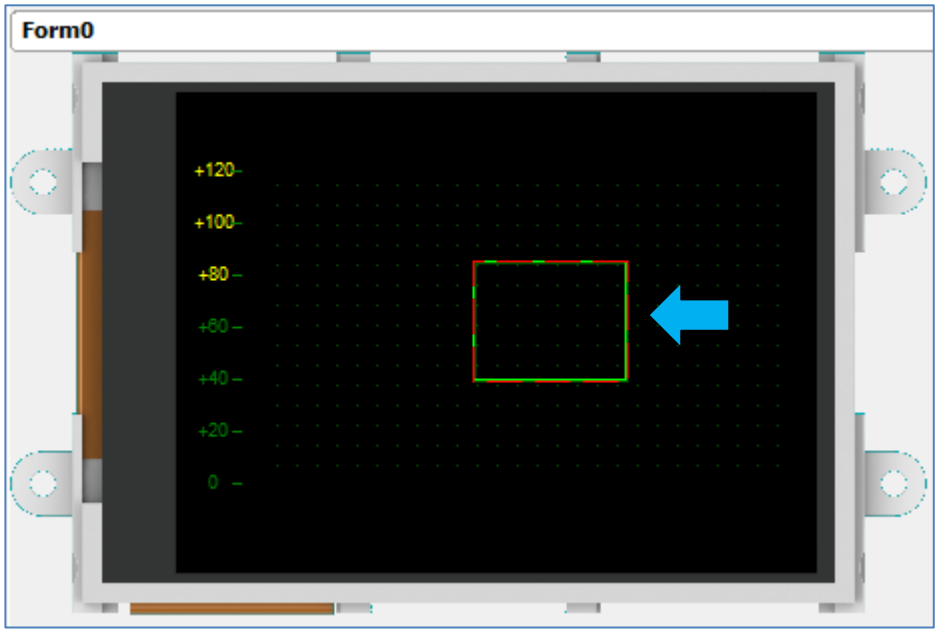


Adding a Border

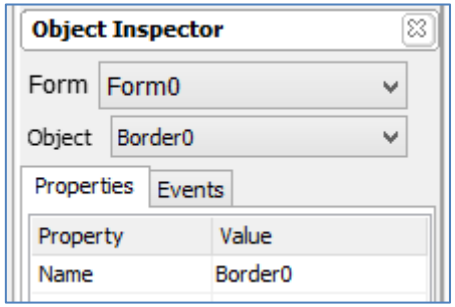
To add a border, go to the **Backgrounds** pane and click on the **border** icon.

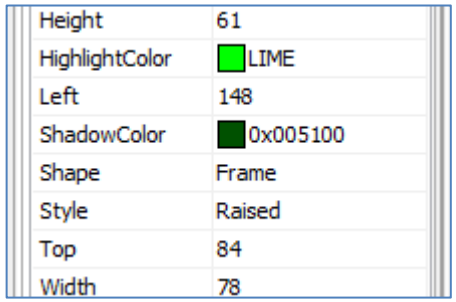


Click on the WYSIWYG screen to place the object.

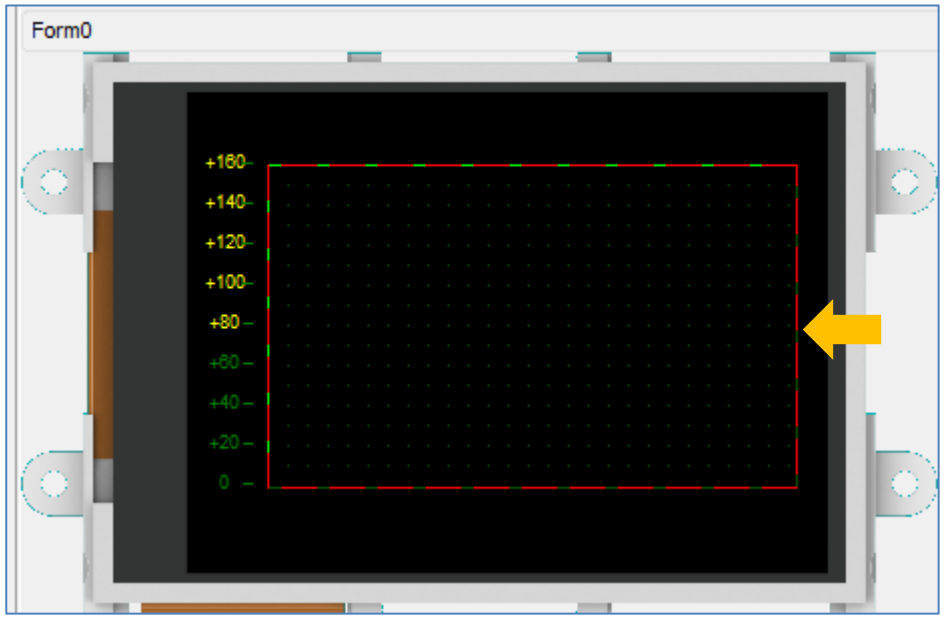


The **Object Inspector** on the right part of the screen displays all the properties of the newly created border named **Border0**. The background border used in this project has the following properties.



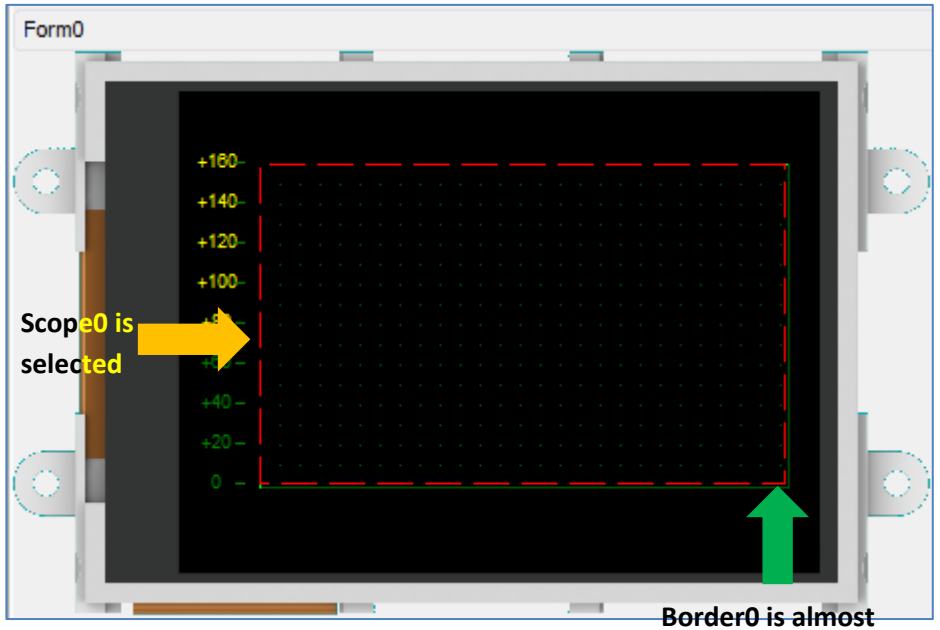


Shown below is the final appearance of the border object.

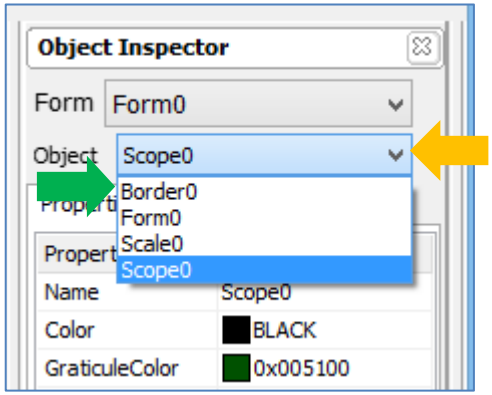


Selecting Objects

To inspect or edit the properties of an object, simply click on it in the WYSIWYG screen. However when one of two objects occupying a common area on the WYSIWYG screen is selected (a background object and a foreground object for example), it may be difficult to select the other object. An example of this case is the background border behind the scope in this project. To illustrate:



To select a hidden object, simply go to the object inspector of the currently selected object, and click on the drop-down arrow of the object line.



Observe that the drop-down menu lists all the objects. Select an object (Border0 for example) to make it reappear on the WYSIWYG screen.

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

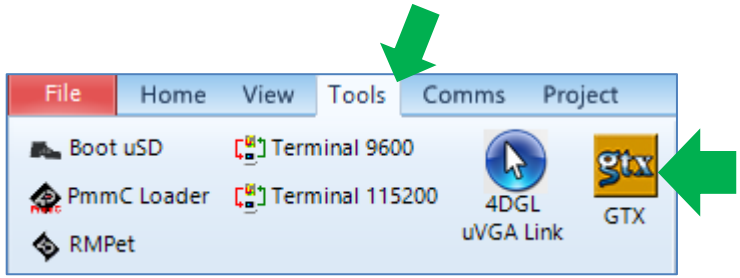
The display module is going to receive and send messages from and to an external host. This section explains to the user how to interpret these messages. An understanding of this section is necessary for users who intend to interface the display to a host. The ViSi Genie Reference Manual is recommended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

Use the GTX Tool to Analyse the Messages

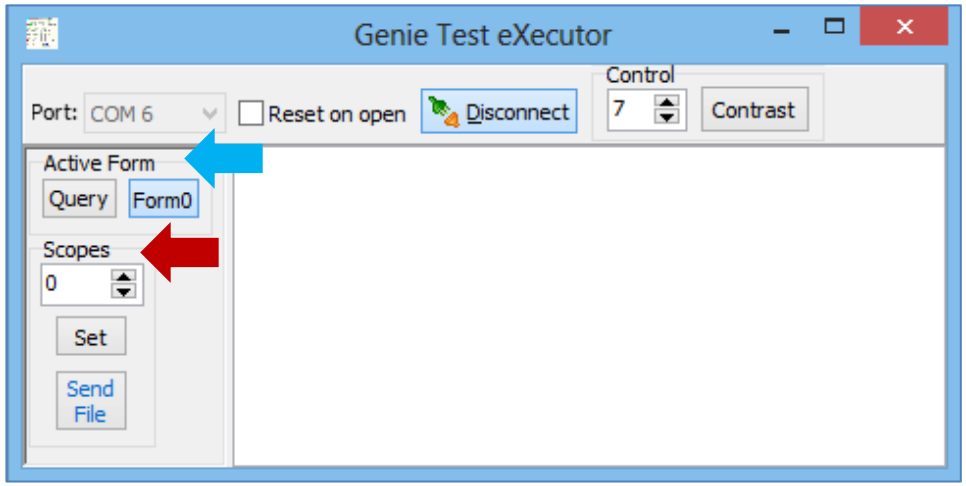
Using the GTX or **Genie Test eXecutor** tool is the first option to get the messages sent by the screen to the host. Here the PC will be the host. The GTX tool is a part of the Workshop 4 IDE. It allows the user to receive, observe, and send messages from and to the display module. It is an essential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

Launch the GTX Tool

Under Tools menu click on the GTX tool button.



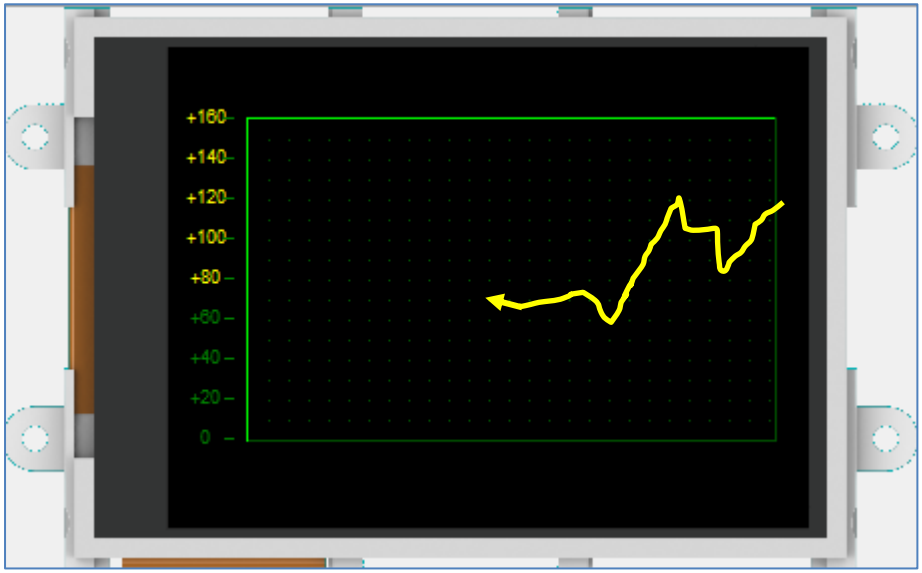
A new window appears, with the form and the scope object created previously.



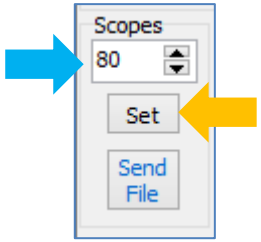
The Scope

Send Values to the Scope

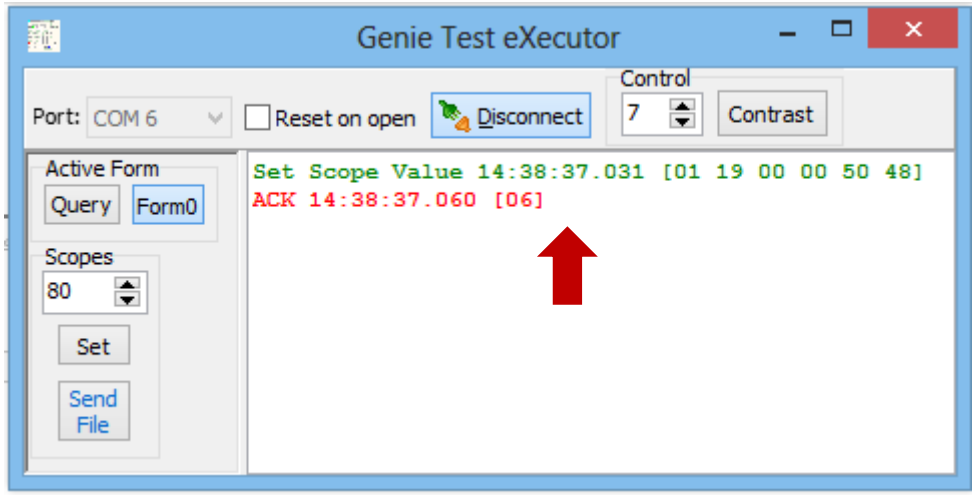
The scope, designed to be controlled by an external host controller, receives values and plots them on the screen. The signal trace moves horizontally from right to left as new values are received. To illustrate:



Enter the integer “80” into the text box and press set.



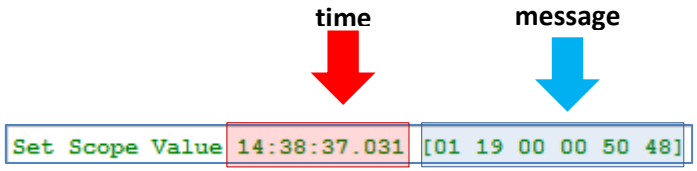
Observe the message box. Messages are sent to and received from the display module.



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

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 sent is formatted according to the following pattern below.

Command	Object Type	Object Index	Value MSB	Value LSB	Checksum
01	19	00	00	50	48
WRITE_OBJ	Scope	First	0x0050		

The message stands for "Write to the first scope object on the display module the value **0x0050**." Converting the hexadecimal value **0x0050** to decimal will yield the value **80**.

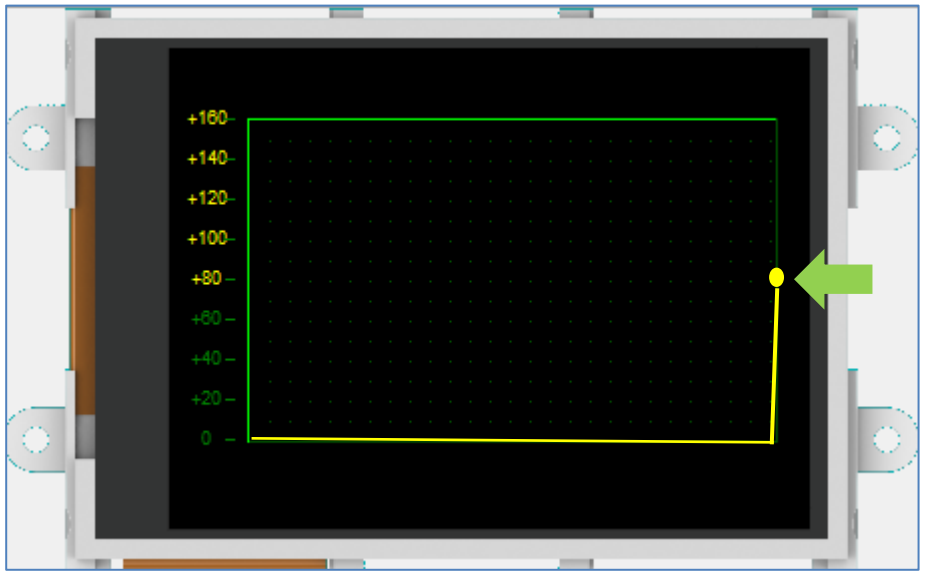
The checksum is a means for the host to verify if the message received is correct. This byte is calculated by XOR'ing all bytes in the message from (and including) the CMD or command byte to the last parameter byte. Then, the result is appended to the end to yield the checksum byte. If the message is 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.

ACK = 0x06 as shown below

```
ACK 14:38:37.060 [06]
```

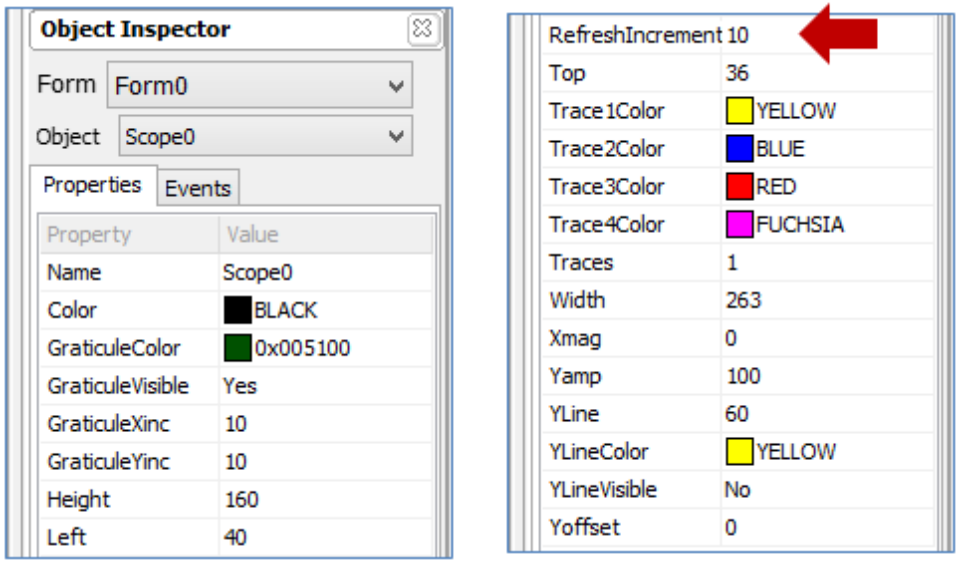
is an acknowledgment from the display module which means that it has understood the message.

After having received the value 0x0050, the scope is expected to plot a point on the right part of the screen.



The size of the point (1 pixel) is exaggerated here. Note that a horizontal line is initially drawn at the bottom of the scope to indicate the location of the

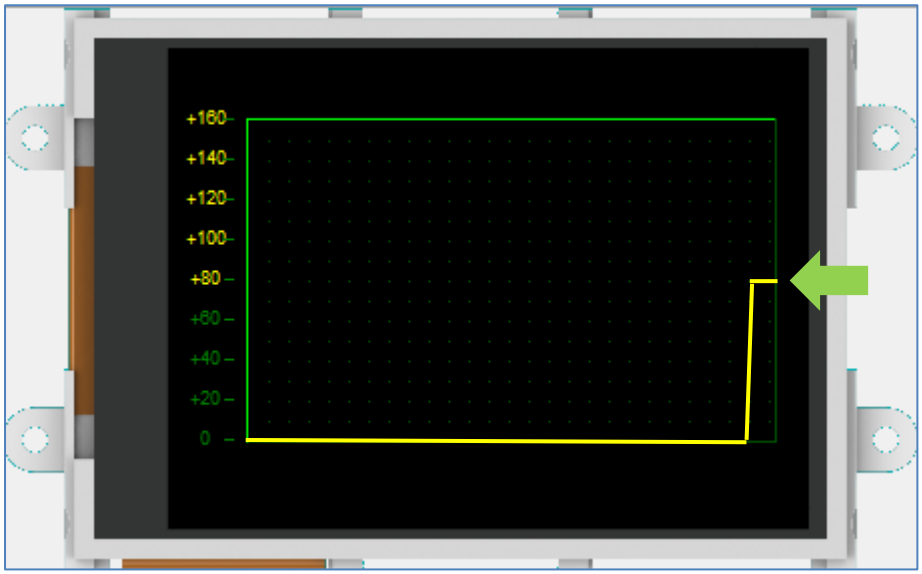
reference line (the x axis or the line y = 0). However, no point is drawn yet because of the value of the **RefreshIncrement** property. This property determines when or how often the scope updates or redraws the screen.



This means that the scope will display a trace only when it has received **10 values**. To make the scope display the values therefore, send nine more integers to the display module. Remember that the scope has a height of 60 pixels. Hence it will plot a value between 0 and 160. In this example, ten integers (value = 80) are sent to the display module.

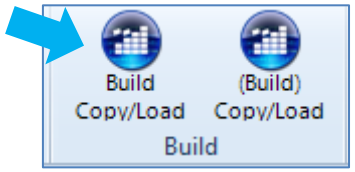
```
Set Scope Value 17:12:08.645 [01 19 00 00 50 48]
ACK 17:12:08.685 [06]
Set Scope Value 17:12:09.026 [01 19 00 00 50 48]
ACK 17:12:09.062 [06]
Set Scope Value 17:12:09.410 [01 19 00 00 50 48]
ACK 17:12:09.439 [06]
Set Scope Value 17:12:09.804 [01 19 00 00 50 48]
ACK 17:12:09.841 [06]
Set Scope Value 17:12:10.210 [01 19 00 00 50 48]
ACK 17:12:10.247 [06]
Set Scope Value 17:12:10.604 [01 19 00 00 50 48]
ACK 17:12:10.653 [06]
Set Scope Value 17:12:11.035 [01 19 00 00 50 48]
ACK 17:12:11.094 [06]
Set Scope Value 17:12:11.394 [01 19 00 00 50 48]
ACK 17:12:11.433 [06]
Set Scope Value 17:12:12.966 [01 19 00 00 50 48]
ACK 17:12:12.996 [06]
Set Scope Value 17:12:13.224 [01 19 00 00 50 48]
ACK 17:12:13.277 [06]
```

Again, the fourth and fifth bytes of the messages are the value of the points to be drawn by the scope. After having received 10 values, the scope now draws the points



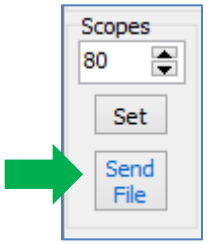
The RefreshIncrement Property

Experiment with the **RefreshIncrement** property and see the results. It controls how frequently the scope is updated. To make the scope update at the fastest rate, set **RefreshIncrement** to 1. Now every time a value is received, the scope immediately draws it on the screen. Remember that each time a property of any object is edited, the program needs to be compiled and downloaded to the display module again for the changes to take effect. Before doing this however, close the GTX tool window or 'disconnect' it from the display module first.

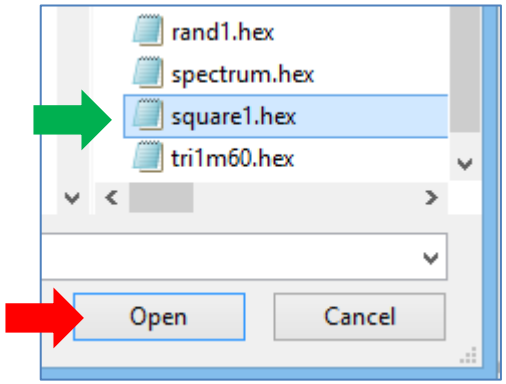


Send a Hex File

To see the scope in action, click on the **Send File** button.



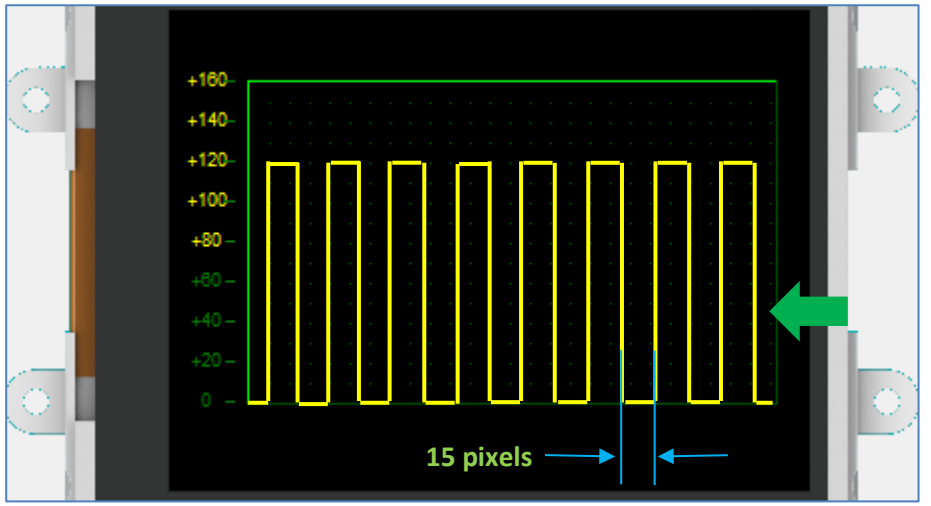
A standard open window asks for a hex file. Navigate to the documents folder of your Workshop IDE (for example: C:\Users\Public\Documents\4D Labs\Picaso ViSi Genie). Select the file "**square1.hex**".



A succession of messages is sent to the display module. Take note of the value bytes. This file is for demo purposes only.

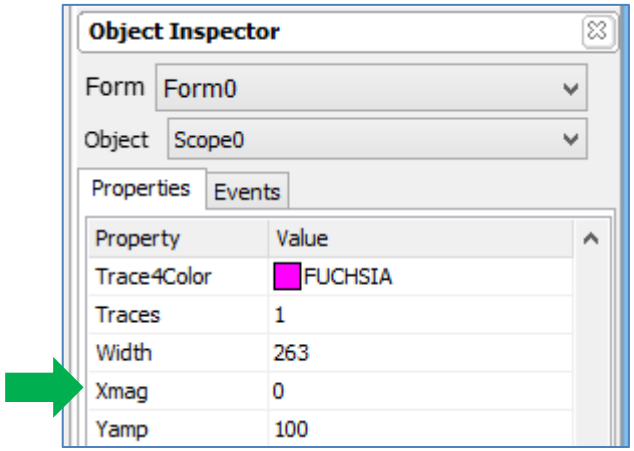
```
Set Scope Value 17:33:23.943 [01 19 00 00 00 18]
ACK 17:33:23.971 [06]
Set Scope Value 17:33:23.979 [01 19 00 00 00 18]
ACK 17:33:24.004 [06]
Set Scope Value 17:33:24.016 [01 19 00 00 00 18]
ACK 17:33:24.030 [06]
Set Scope Value 17:33:24.040 [01 19 00 00 00 18]
ACK 17:33:24.066 [06]
Set Scope Value 17:33:24.076 [01 19 00 00 00 18]
ACK 17:33:24.103 [06]
Set Scope Value 17:33:24.112 [01 19 00 00 78 60]
ACK 17:33:24.127 [06]
Set Scope Value 17:33:24.136 [01 19 00 00 78 60]
ACK 17:33:24.165 [06]
Set Scope Value 17:33:24.174 [01 19 00 00 78 60]
ACK 17:33:24.189 [06]
Set Scope Value 17:33:24.199 [01 19 00 00 78 60]
```

The hex file alternately sends fifteen 0x0000's and fifteen 0x0078's. In decimal these are fifteen 0's and fifteen 120's. The scope now displays the waveform.

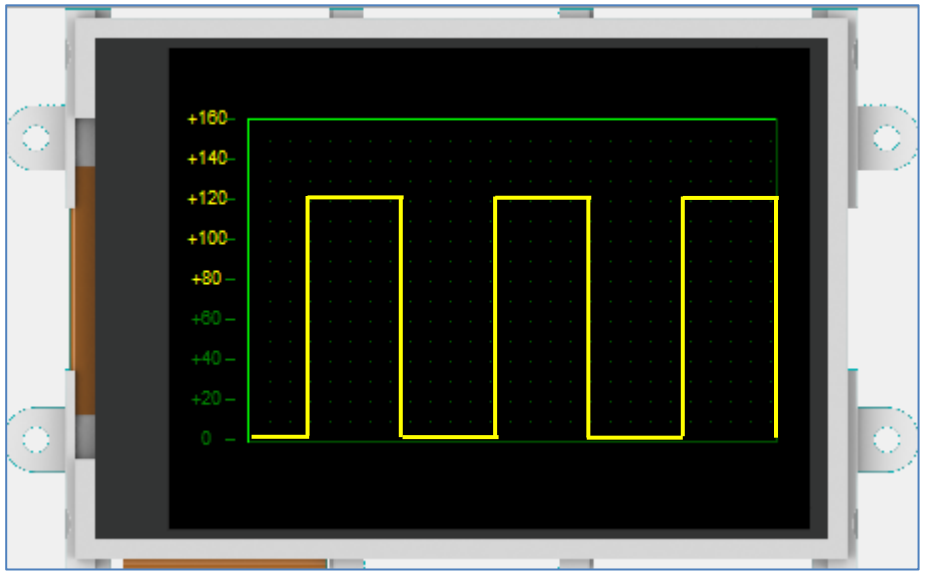


Scale the Waveform along the X axis

The parameter **Xmag** is used to compress or expand values in the X direction. Normally 0 is for 'standard'.

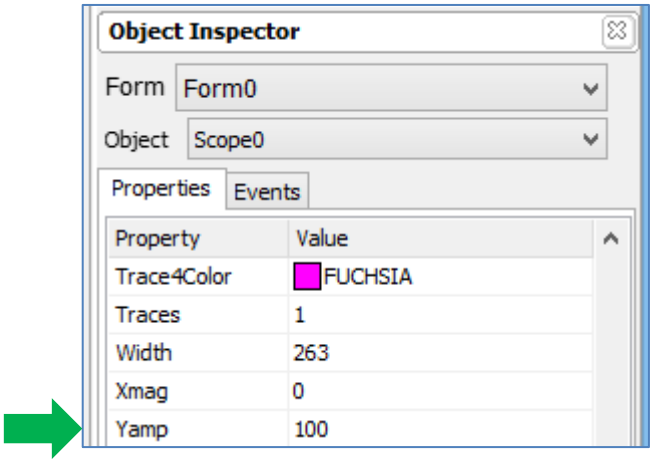


Use a positive integer to expand or a negative integer to compress the values. **Be aware that negative values significantly increase the amount of internal memory required to hold scope values and that such values should be used carefully.** When **Xmag** is set to 3 for example, the resulting graph for "**square1.hex**" is shown below.

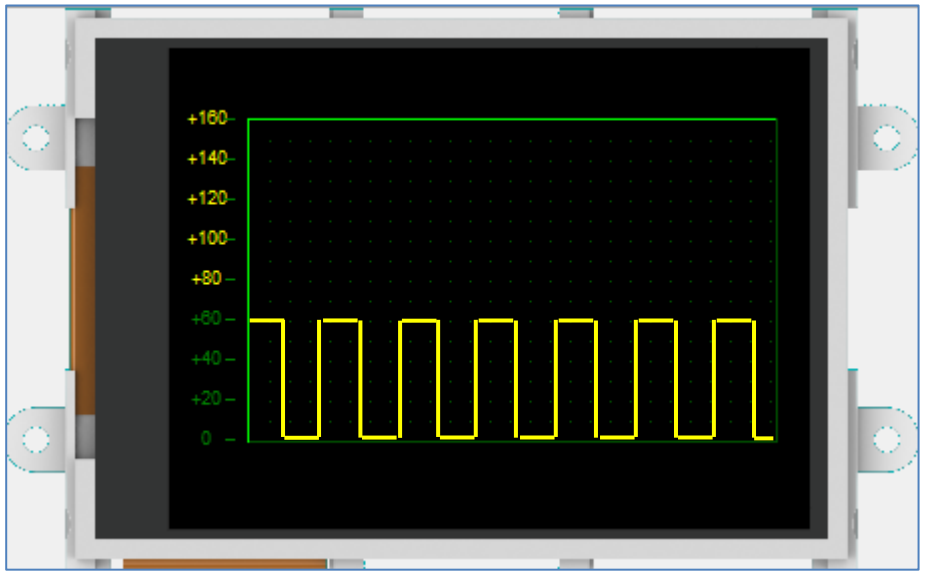


Scale the Waveform along the Y axis

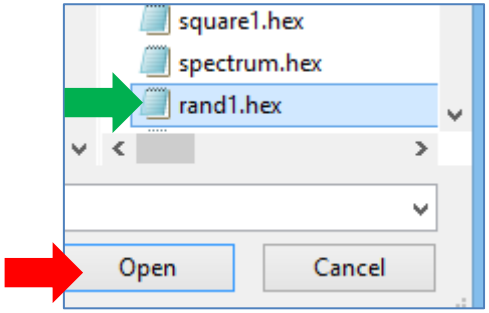
The parameter **Yamp** amplifies the values in the Y direction - 100 is unity and 200 is maximum.



To attenuate the signal amplitude by fifty percent for instance, set **Yamp** to 50. The result is shown below.

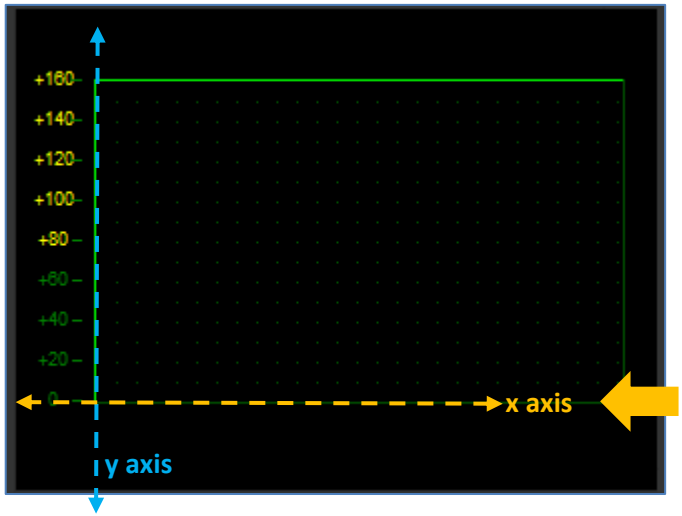


Also try experimenting with the **RefereshIncrement**, **Xmag**, and **Yamp** properties along with sending the hex file "**rand1.hex**". The file sends random values between 0 and 120 to the scope. This file is for demo only.

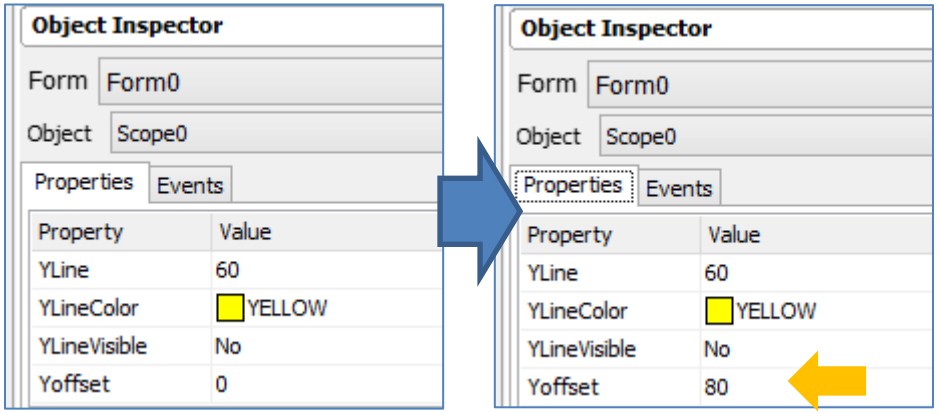


Plot Values in the negative Y direction

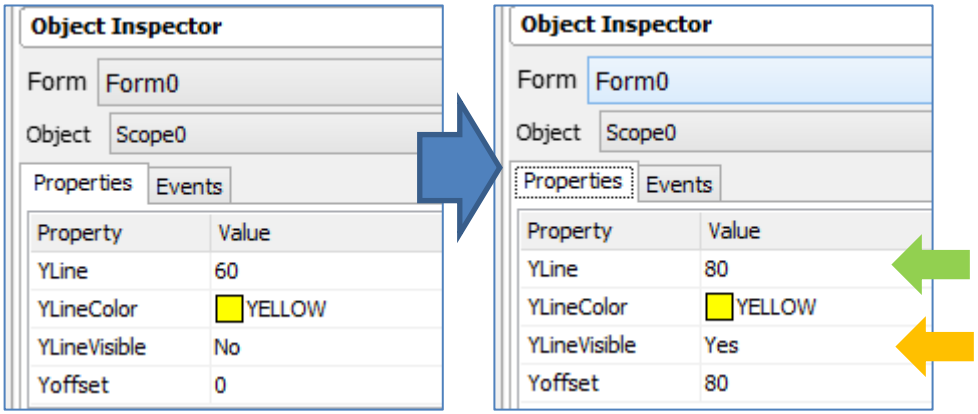
To make the scope plot points in the negative Y region, the **Yoffset** property must be edited. By default the x-axis or the line y = 0 is at the bottom of the scope.



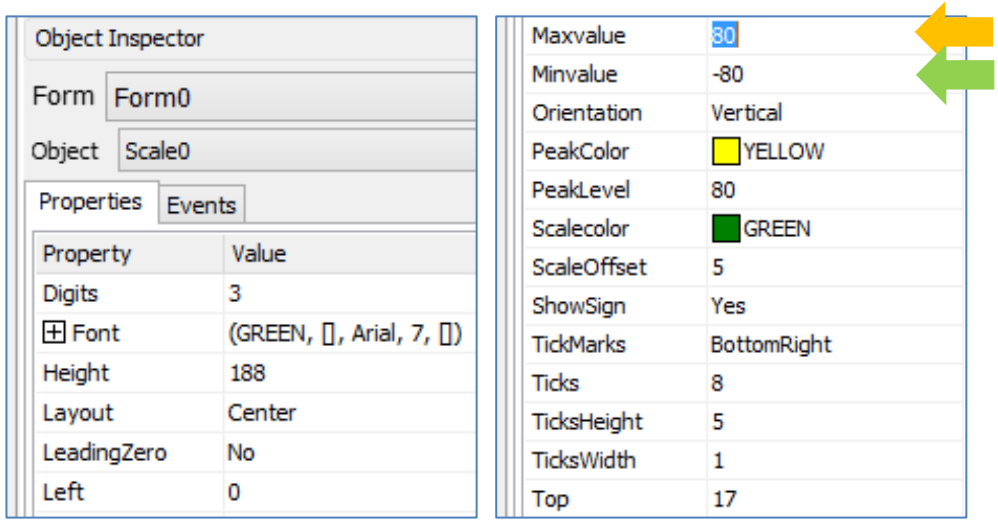
To move the x-axis up, edit the property **Yoffset**. Here the value is changed from **0** (default) to **80**.



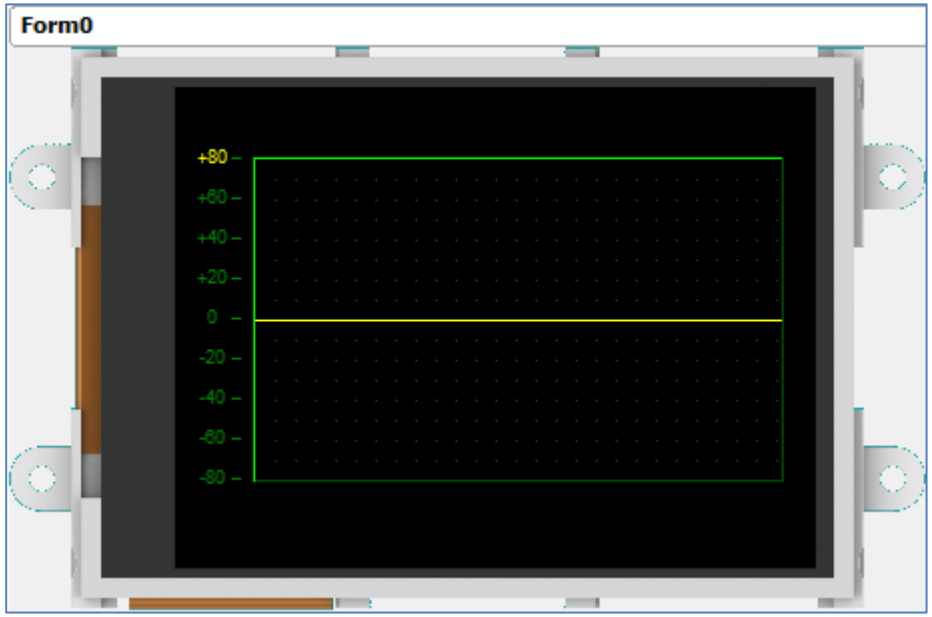
Make the scope draw a horizontal line to represent the x-axis by editing the **YLine** properties.



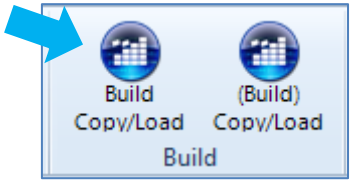
Change the properties of the scale as well.



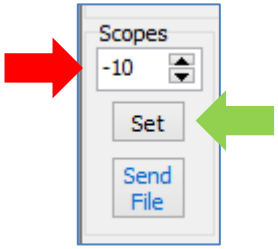
The final appearance of the form is shown below.



Build and download the program.



In the GTX tool window if the user repetitively sends the value '-10' to the scope



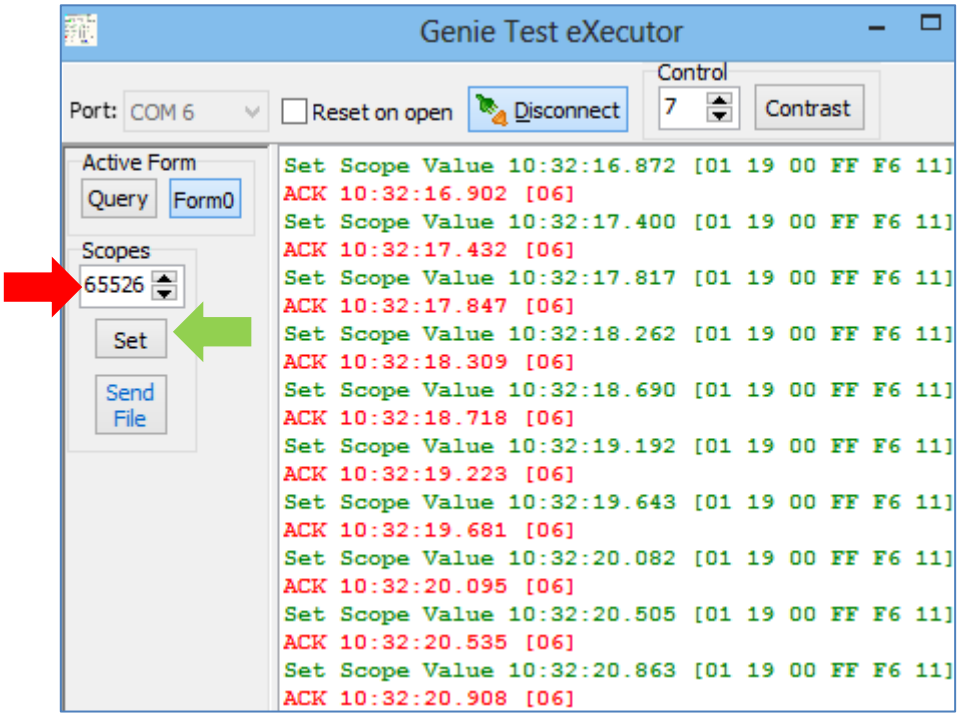
the values sent are as shown below.

```
Set Scope Value 09:36:32.855 [01 19 00 00 F6 EE]
ACK 09:36:32.905 [06]
Set Scope Value 09:36:33.091 [01 19 00 00 F6 EE]
ACK 09:36:33.124 [06]
Set Scope Value 09:36:33.261 [01 19 00 00 F6 EE]
```

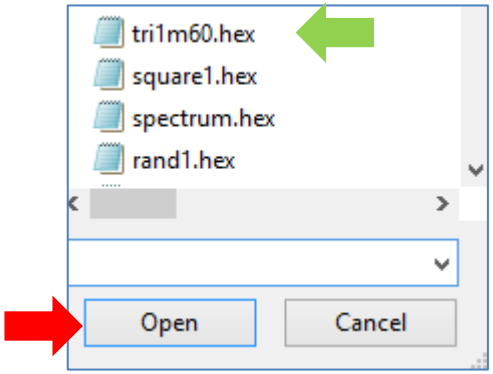
These are actually negative integers in 8-bit two's-complement notation. The scope however is expecting a 16-bit value, hence the negative integers must be represented in a 16-bit two's-complement notation. Also, note that the scope is not able to graph the values properly since it actually interprets them as positive values (**0x00F6** is **246** in decimal). To get the 16-bit two's complement notation of a negative integer, simply subtract its absolute value from 2^{16} or 65536. To illustrate:

Negative integer	Absolute value	2^{16} – absolute value	Hexadecimal
-10	10	65526	0xFFF6
-60	60	65476	0xFFC4
-80	80	65456	0xFFB0

To make the scope display the line y = -10, repetitively send 65526 in the GTX tool window.



To see more of the scope in action, send the hex file "**tri1m60.hex**".



All communications 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