



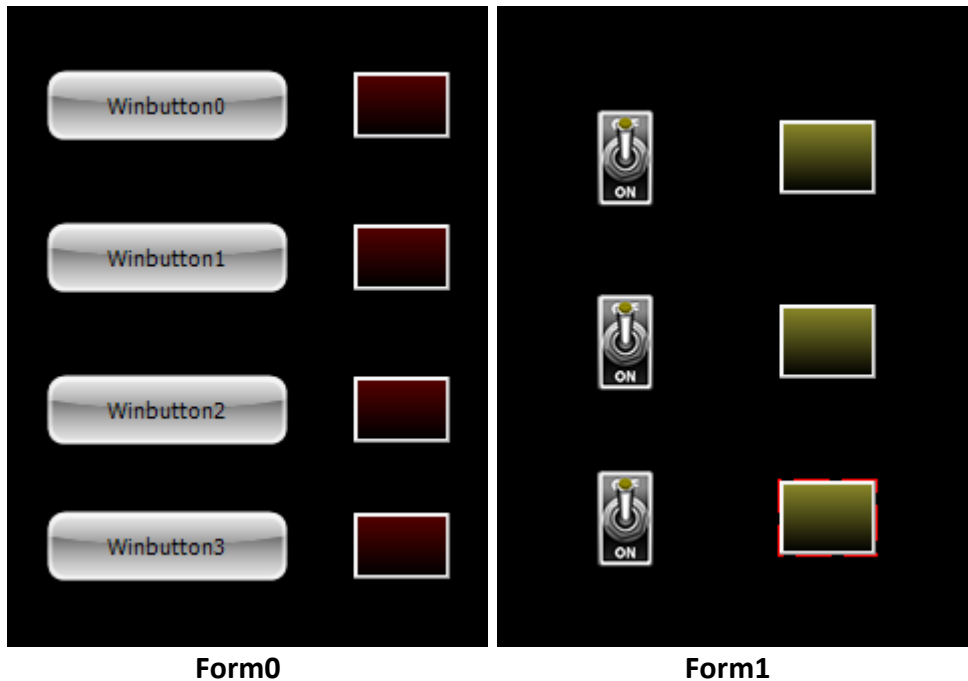
## ViSi-Genie Generated Header Files

DOCUMENT DATE: **24<sup>th</sup> MAY 2019**  
DOCUMENT REVISION: **1.1**



## Description

This application note shows how to configure Workshop to generate a header file containing a list of defined constants for the objects used in a ViSi-Genie project. For each object, the constant name can be the object name or the alias. The constant value is a two-byte hexadecimal value, the high byte and low byte of which are the object ID and the object index, respectively. Below is a screenshot image of the project used in this application note.



Before getting started, the following are required:

- Any of the following 4D Picaso and gen4 Picaso display modules:

[gen4-uLCD-24PT](#)    [gen4-uLCD-28PT](#)    [gen4-uLCD-32PT](#)  
[uLCD-24PTU](#)    [uLCD-32PTU](#)    [uVGA-III](#)

and other superseded modules which support the ViSi Genie environment

- The target module can also be a Diablo16 display

[gen4-uLCD-24D series](#)    [gen4-uLCD-28D series](#)    [gen4-uLCD-32D series](#)  
[gen4-uLCD-35D series](#)    [gen4-uLCD-43D series](#)    [gen4-uLCD-50D series](#)  
[gen4-uLCD-70D series](#)  
[uLCD-35DT](#)    [uLCD-43D series](#)    [uLCD-70DT](#)

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

- [4D Programming Cable](#) / [uUSB-PA5/uUSB-PA5-II](#) for non-gen4 displays(uLCD-xxx)
- [4D Programming Cable](#) & [gen4-PA](#) / [gen4-IB](#) / [4D-UPA](#) for gen4 displays (gen4-uLCD-xxx)
- [micro-SD \(μSD\)](#) memory card
- [Workshop 4 IDE](#) (installed according to the installation document)

- When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or has a working knowledge of the topics presented in these recommended application notes.

Content	
<b>Description</b>	<b>2</b>
<b>Content</b>	<b>3</b>
<b>Application Overview</b>	<b>4</b>
<b>Setup Procedure</b>	<b>5</b>
<b>Create a New Project</b>	<b>5</b>
<i>Create a New Project</i>	<b>5</b>
<b>Design the Project</b>	<b>5</b>
<i>Add Four Winbutton Objects to Form0</i>	<b>5</b>
Configure the OnChanged Event of the Winbutton Objects	<b>5</b>
<i>Add Four User LED Objects to Form0</i>	<b>7</b>
<i>Add a User Button Object to Form0</i>	<b>8</b>
Configure the OnChanged Event of Userbutton0	<b>8</b>
<i>Configure the OnActivate Event of Form0</i>	<b>8</b>
<i>Add a New Form to the Project</i>	<b>8</b>
<i>Add Three 4D Button Objects to Form1</i>	<b>9</b>
Configure the OnChanged Event of the 4D Button Objects	<b>9</b>
<i>Add Three User LED Objects to Form1</i>	<b>10</b>
<i>Add a User Button Object to Form1</i>	<b>11</b>
Configure the OnChanged Event of Userbutton1	<b>11</b>
<i>Configure the OnActivate Event of Form1</i>	<b>11</b>
<i>Model</i>	<b>12</b>
WRITE_MAGIC_BYTES	<b>12</b>

REPORT_EVENT	13
<i>Configure Workshop to Generate a Header File</i>	13
<b>Build and Upload the Project</b>	14
<b>Check the Generated Header File</b>	15
<i>Header File Filename Format</i>	15
<i>Open the Header File</i>	15
<i>Replace the Object Name with the Alias</i>	16
<i>The Genie Index Element</i>	17
<b>Identify the Messages</b>	18
<i>Use the GTX Tool to Analyse the Messages</i>	18
<i>Receive a Message from the Display Module</i>	19
REPORT_EVENT Message from a Winbutton Object	19
REPORT_EVENT Message from a Form Object	20
REPORT_EVENT Message from a 4D Button Object - On	21
REPORT_EVENT Message from a 4D Button Object - Off	21
<i>Send a Message to the Display Module</i>	22
Send a WRITE_OBJ Message to a User LED Object	22
Send a WRITE_OBJ Message to a Form Object	24
<b>Proprietary Information</b>	26
<b>Disclaimer of Warranties &amp; Limitation of Liability</b>	26

## Application Overview

When creating a ViSi-Genie project, objects have a standard name and an alias name. The standard name and the alias name are identical by default. The user can modify only the alias name of an object.

When writing the source code for the host program, it would be helpful for the user to be able to import the standard names or alias names of all objects, especially if the ViSi-Genie project is large, i.e. it contains several forms and objects. The user can then make use of the object names or aliases when writing the source code for the host program. This application note shows how this is done.

The ViSi-Genie project attached to this application note consists of two forms. The first form has four winbutton objects, four user LED objects, and a user button object. The second form has three 4D button objects, three user LED objects, and a user button object. Several of these objects are given a unique alias. It is then shown how Workshop is configured to generate a header file containing defined constants for all of the objects.

The GTX tool is then used to analyse the syntax of the messages coming from and sent to the display module. The message for an object is then related to the constant defined for it in the header file. The use of the defined constant (and the header file) is then illustrated using pseudo codes.

## 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)

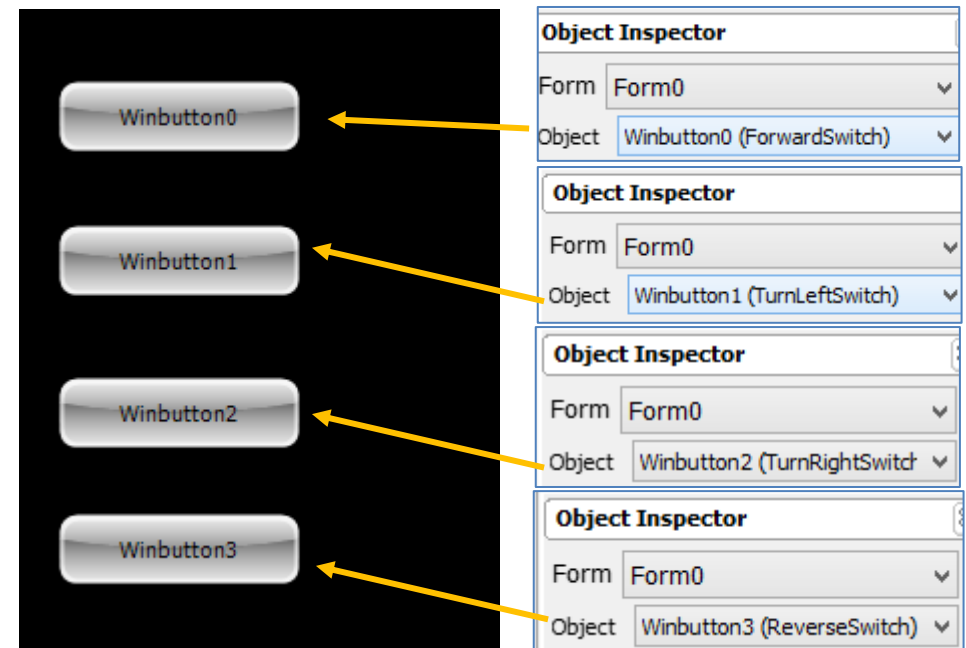
or

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

## Design the Project

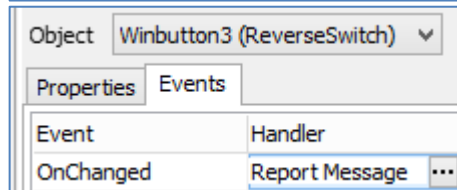
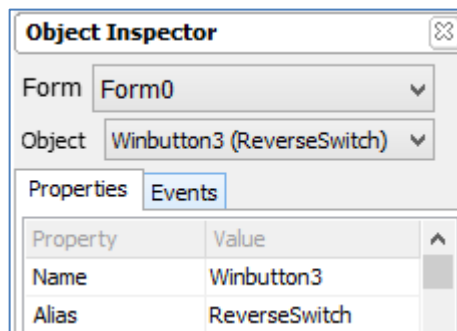
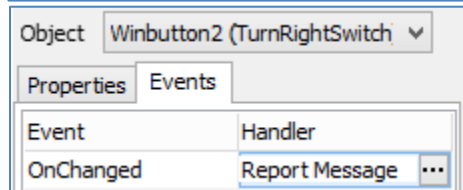
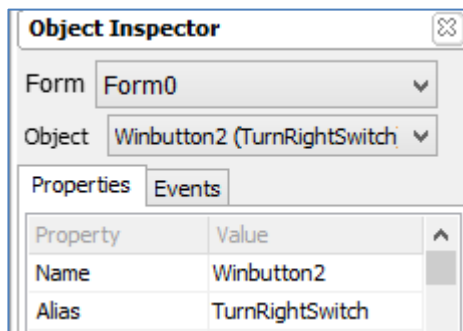
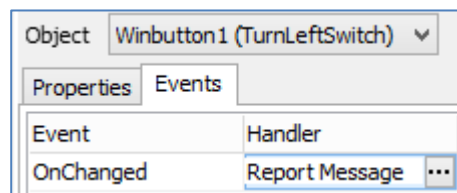
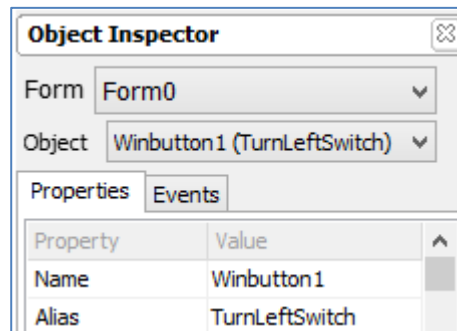
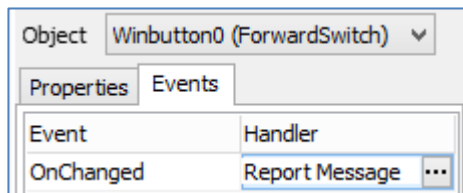
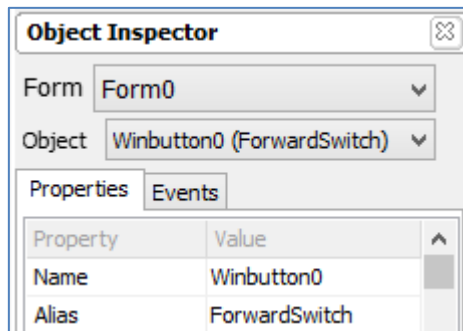
### Add Four Winbutton Objects to Form0

Four winbutton objects are added to Form0. These are **Winbutton0**, **Winbutton1**, **Winbutton2**, and **Winbutton3**. The alias name for a button is appended to the object name.



### Configure the OnChanged Event of the Winbutton Objects

Each winbutton object is configured to send a message to the serial port when touched.

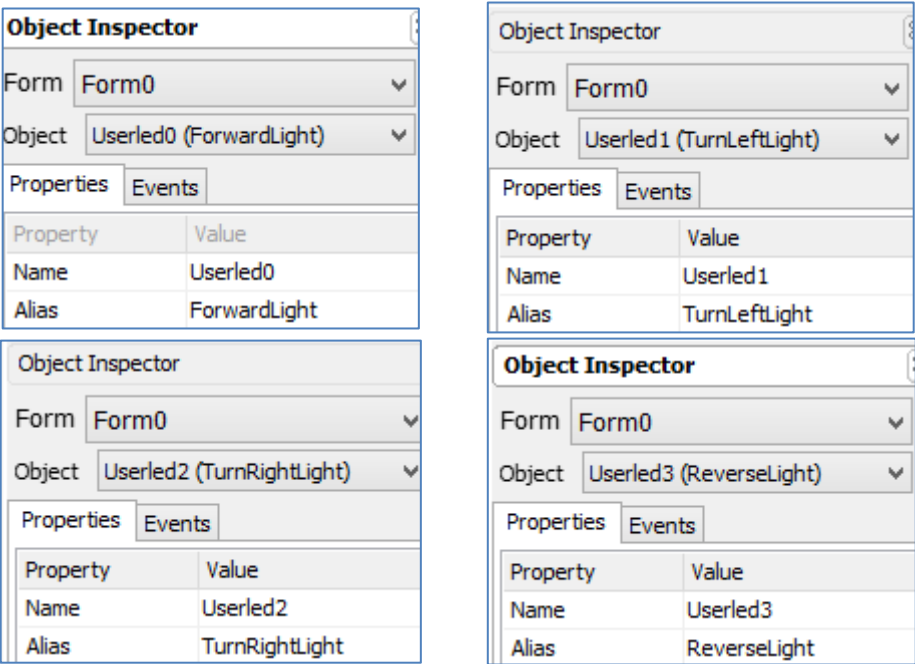
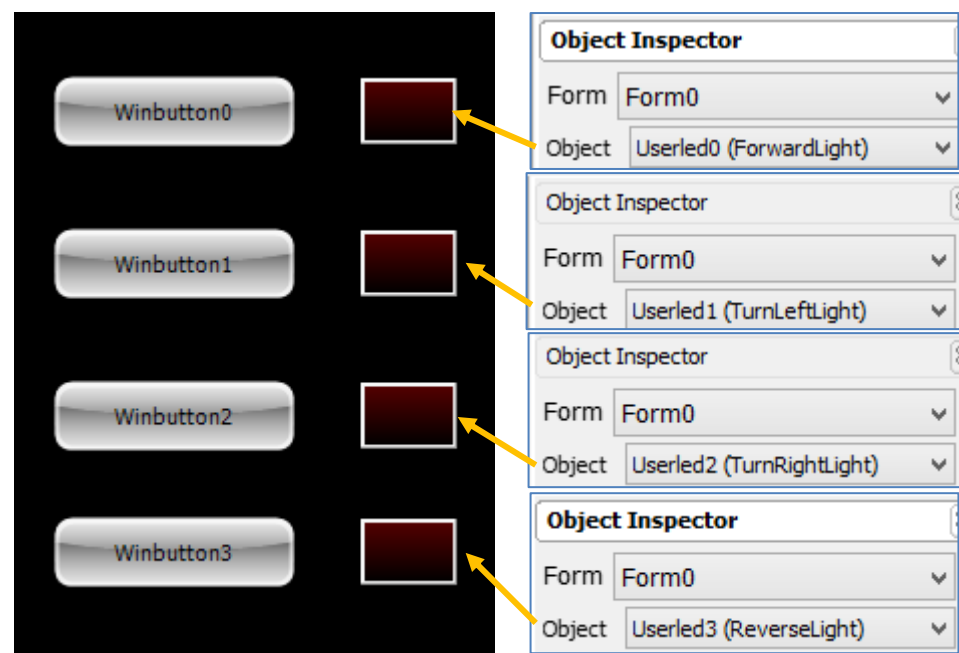


To know more about winbutton objects, their properties, and how they are added to a project, refer to the application note

[ViSi-Genie Advanced Buttons](#)

Add Four User LED Objects to Form0

Four user LED objects are added to Form0. These are **Userled0**, **Userled1**, **Userled2**, and **Userled3**. The alias name for a user LED object is appended to the object name.

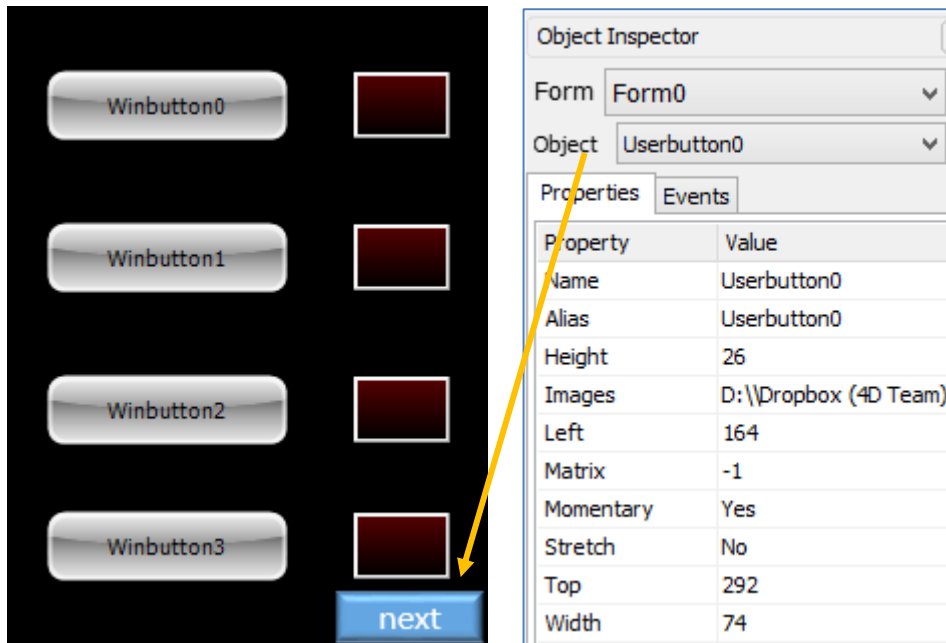


To know more about user LED objects, their properties, and how they are added to a project, refer to the application note

[ViSi-Genie Digital Displays](#)

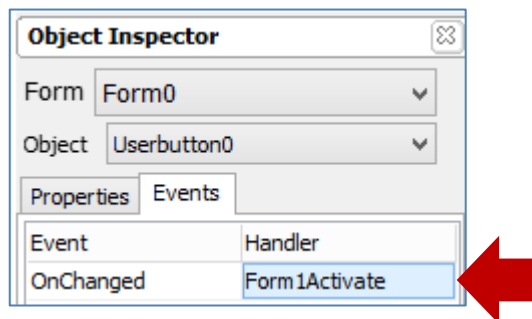
### Add a User Button Object to Form0

A user button object is added to Form0. This is **Userbutton0**.



### Configure the OnChanged Event of Userbutton0

Use **Userbutton0** for navigating to **Form1**.

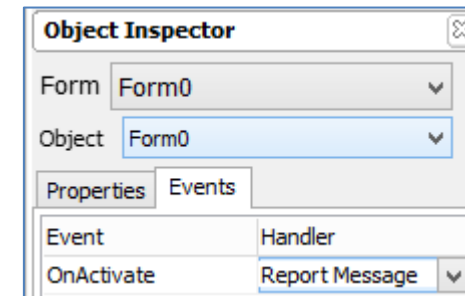


To know more about user button objects, their properties, and how they are added to a project, refer to the application note

[ViSi-Genie User Button](#)

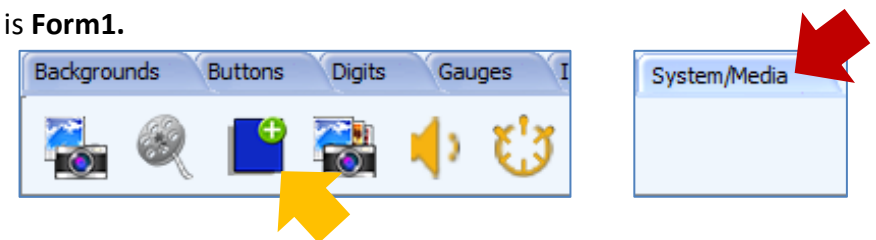
### Configure the OnActivate Event of Form0

Make **Form0** send a message to the serial port when it is activated.



### Add a New Form to the Project

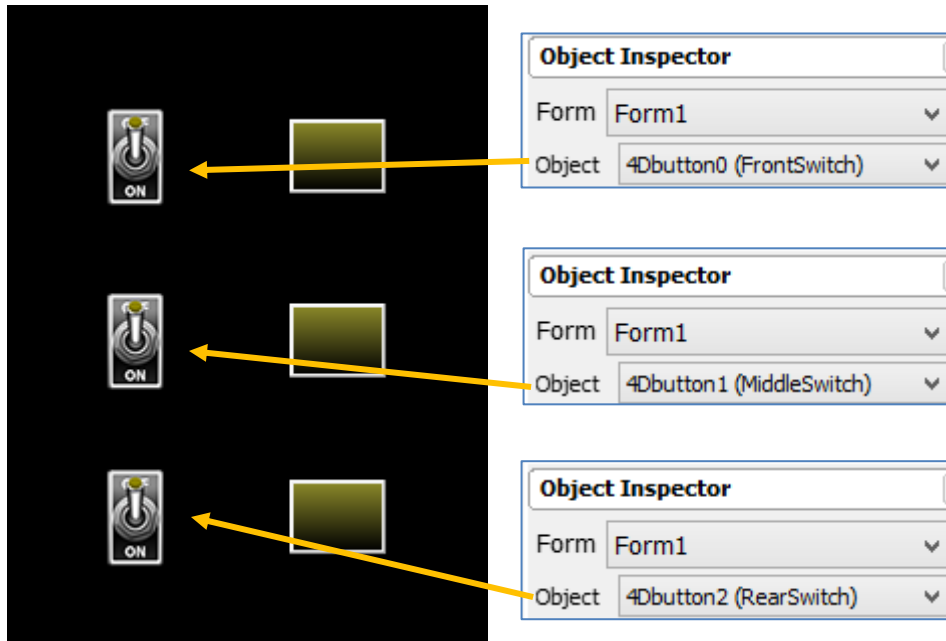
This is **Form1**.





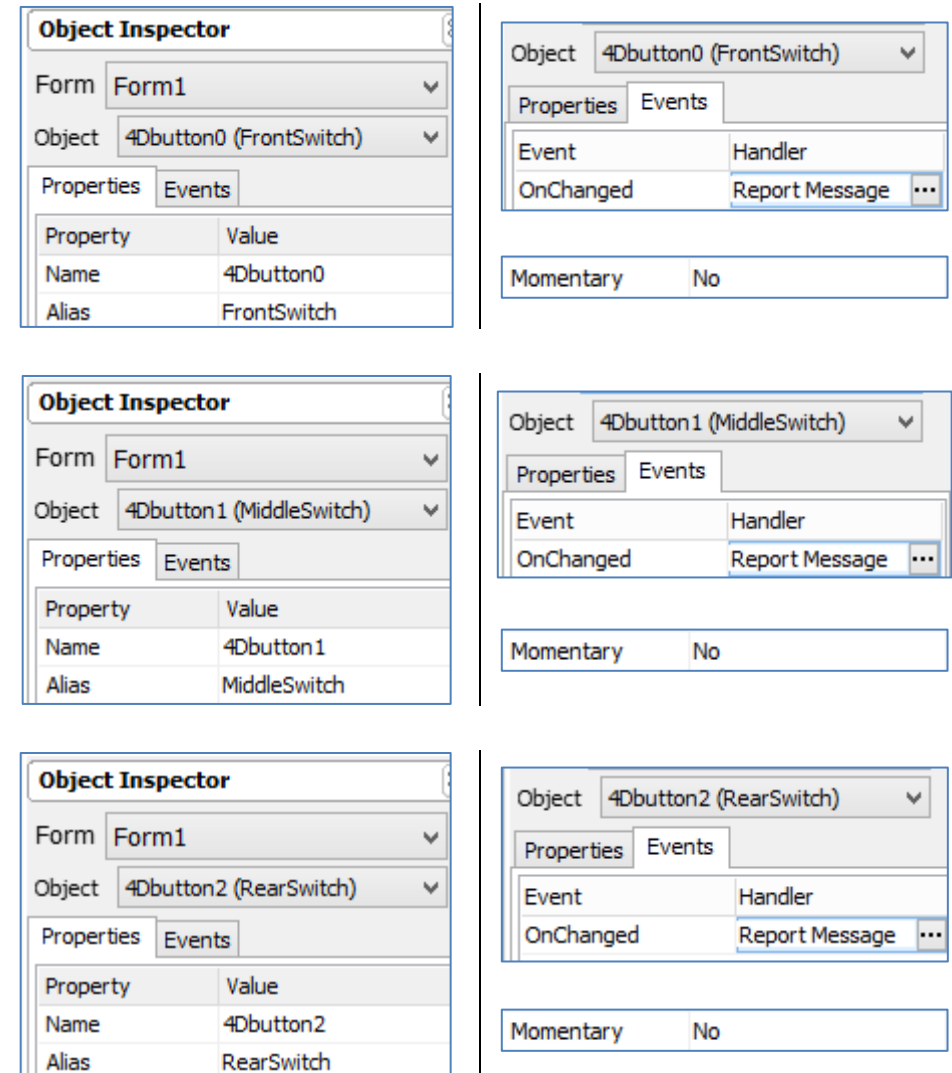
### Add Three 4D Button Objects to Form1

Three 4D button objects are added to Form0. These are **4Dbutton0**, **4Dbutton1**, and **4Dbutton2**. The alias name for a 4D button object is appended to the object name.



### Configure the OnChanged Event of the 4D Button Objects

Each 4D button object is configured to send a message to the serial port when touched.

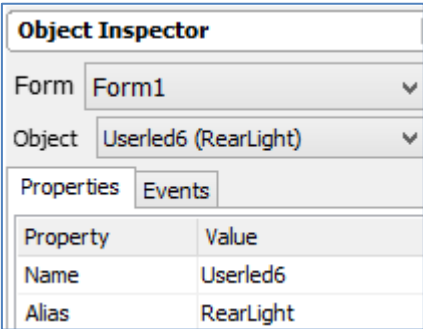
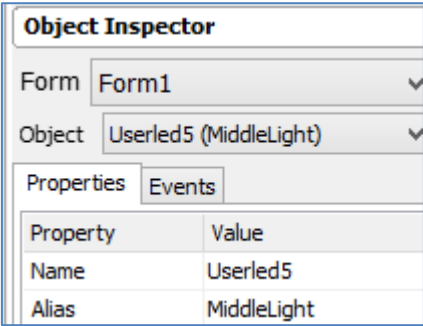
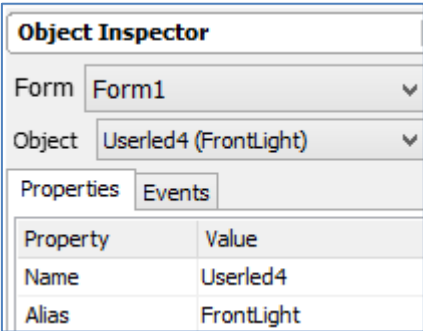
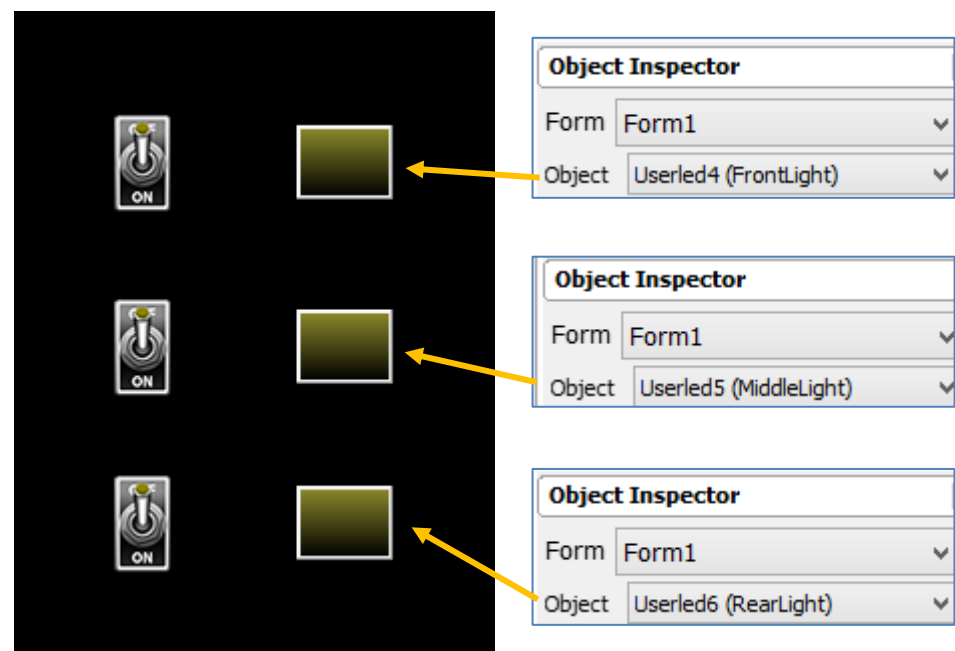


To know more about the 4D button objects, their properties, and how they are added to a project, refer to the application note

[ViSi-Genie 4D Buttons](#)

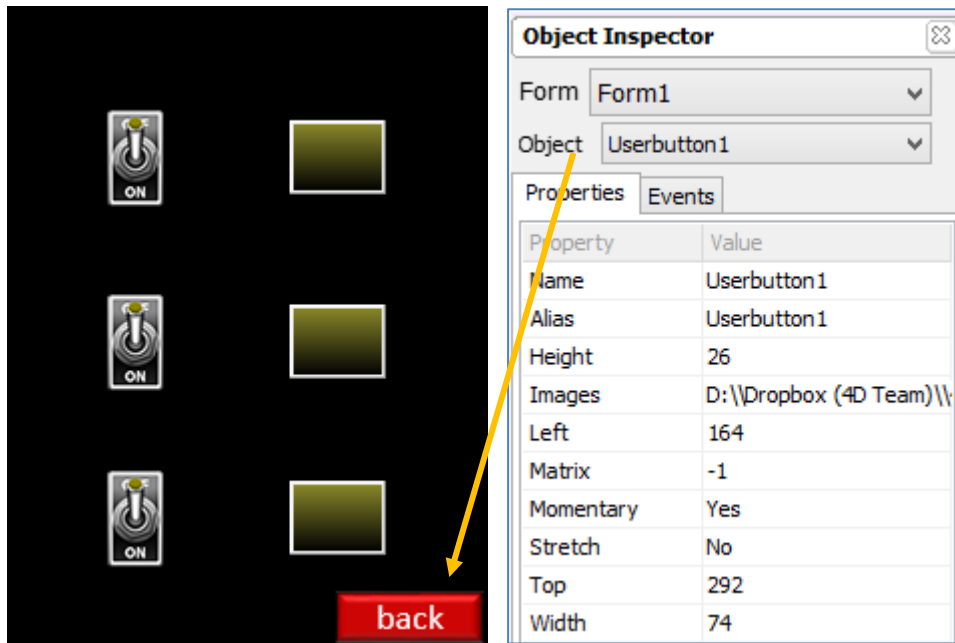
**Add Three User LED Objects to Form1**

Three user LED objects are added to **Form1**. These are **Userled4**, **Userled5**, and **Userled6**. The alias name for a user LED object is appended to the object name.



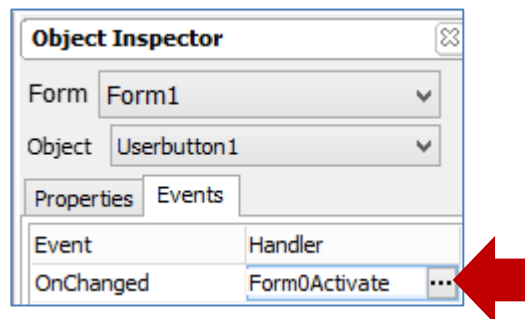
### Add a User Button Object to Form1

A user button object is added to Form0. This is **Userbutton1**.



### Configure the OnChanged Event of Userbutton1

Use **Userbutton1** for navigating to **Form0**.

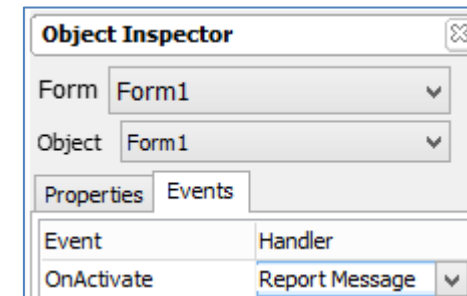


To know more about user button objects, their properties, and how they are added to a project, refer to the application note

[ViSi-Genie User Button](#)

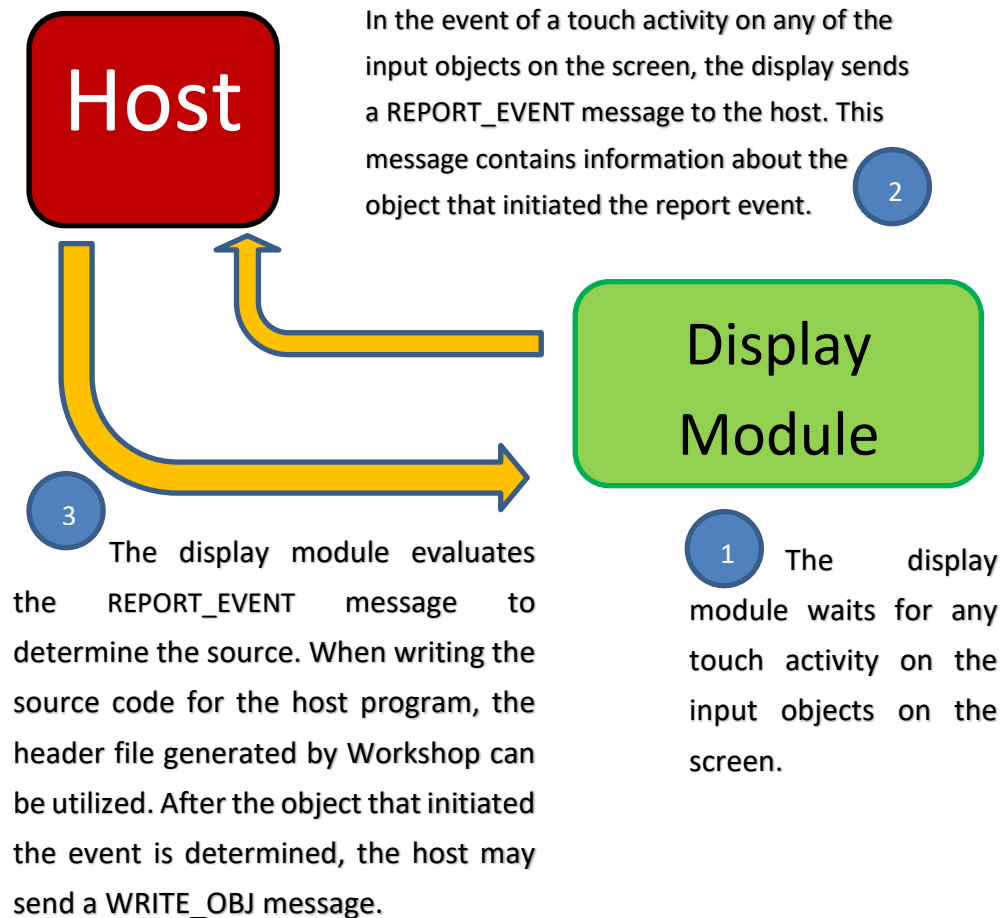
### Configure the OnActivate Event of Form1

Make **Form1** send a message to the serial port when it is activated.



## Model

Below is a model for an application wherein the host interacts with the display module through the use of WRITE\_OBJ and REPORT\_EVENT messages.



The **WRITE\_OBJ** and **REPORT\_EVENT** messages or commands are two complementary messages that are defined in the ViSi-Genie communications protocol.

## WRITE\_MAGIC\_BYTES

The standard format of WRITE\_OBJ message, as defined in section **2.1.2 (Command and Parameters Table)** of the ViSi-Genie Reference Manual is:

Command	Code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter N	Checksum
WRITE_OBJ	0x01	Object ID	Object Index	Value (msb)	Value (lsb)	-	Checksum

The section “**2.1.3.2 Write Object Value Message**” further says:

### Description

The host issues the Write Object command message when it wants to change the status of an individual object item. For example, Meter 3 value needs to be set to 50.

## REPORT\_EVENT

The standard format of a REPORT\_EVENT message, as defined in section **2.1.2 (Command and Parameters Table)** of the ViSi-Genie Reference Manual is:

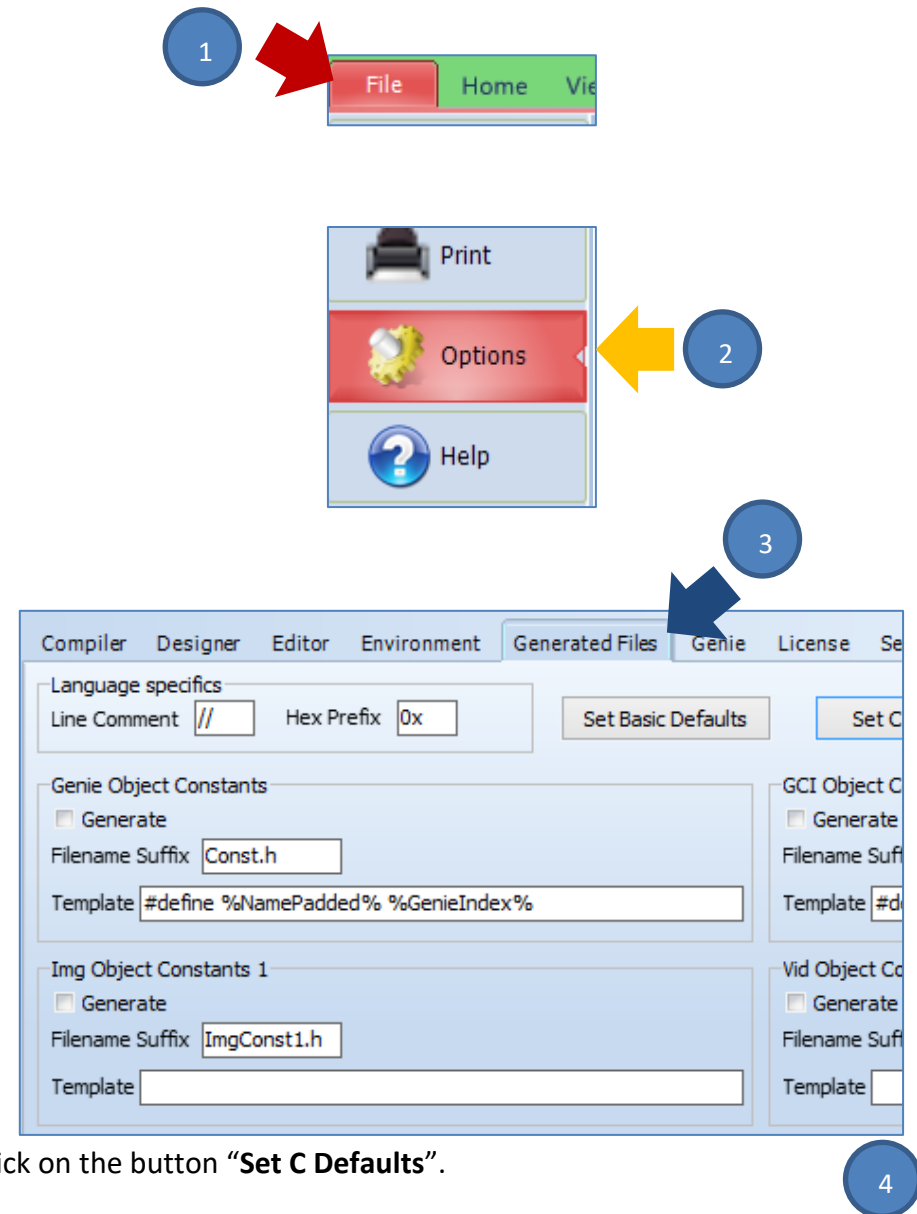
Command	Code	Para- meter 1	Para- meter 2	Para- meter 3	Para- meter 4	Para- meter N	Checksum
REPORT_ EVENT	0x07	Object ID	Object Index	Value (msb)	Value (lsb)	-	Checksum

The section “**2.1.3.7 Report Event Message**” further says:

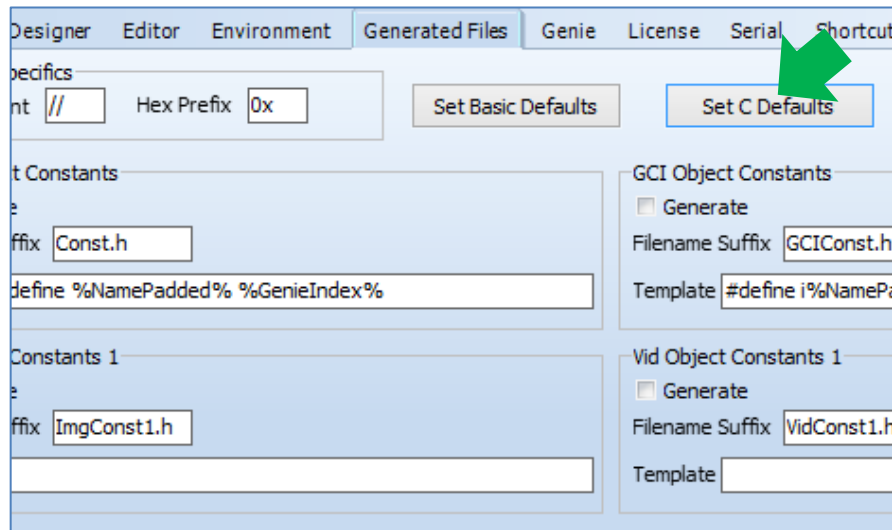
### Description

When designing the Genie display application in Workshop, each Object can be configured to report its status change without the host having to poll it (see Read Object Status message). If the object’s ‘Event Handler’ is set to ‘Report Event’ in the ‘Event’ tab, the display will transmit the object’s status upon any change. For example, Slider 3 object was set from 0 to 50 by the user.

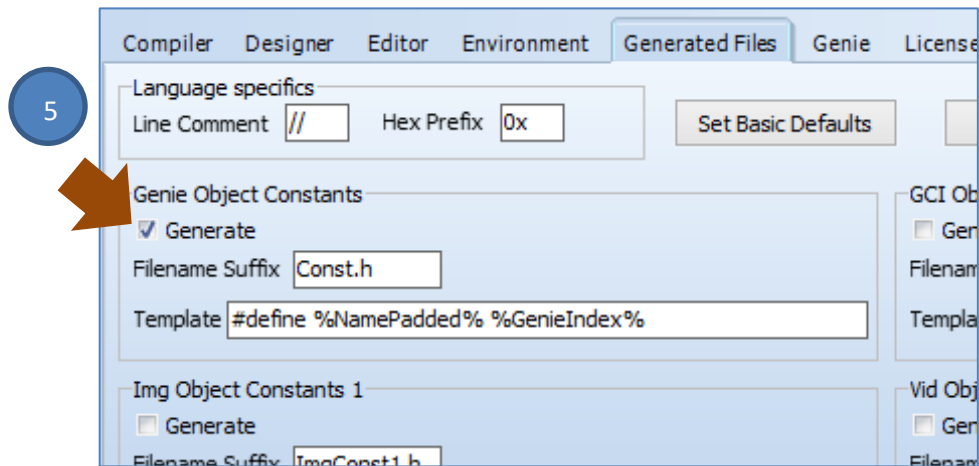
## Configure Workshop to Generate a Header File



Click on the button “Set C Defaults”.



Click on the tick box indicated below.



## 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.

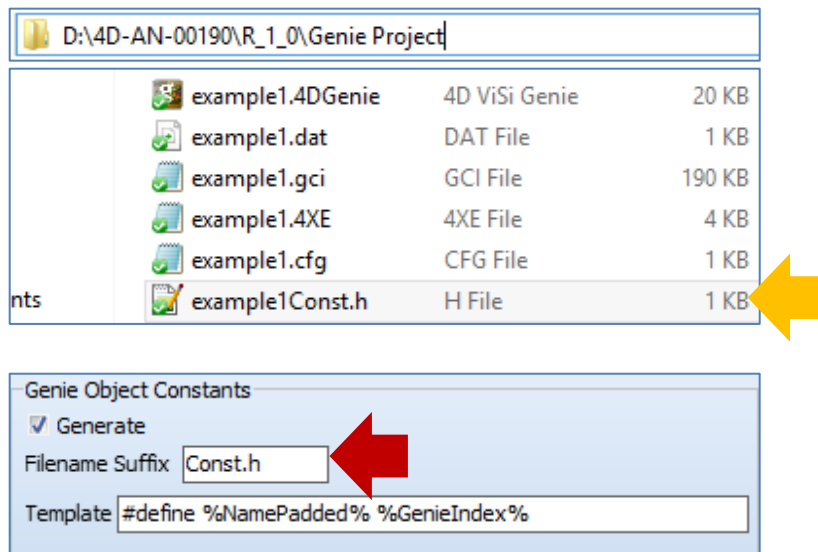
## Check the Generated Header File

### Header File Filename Format

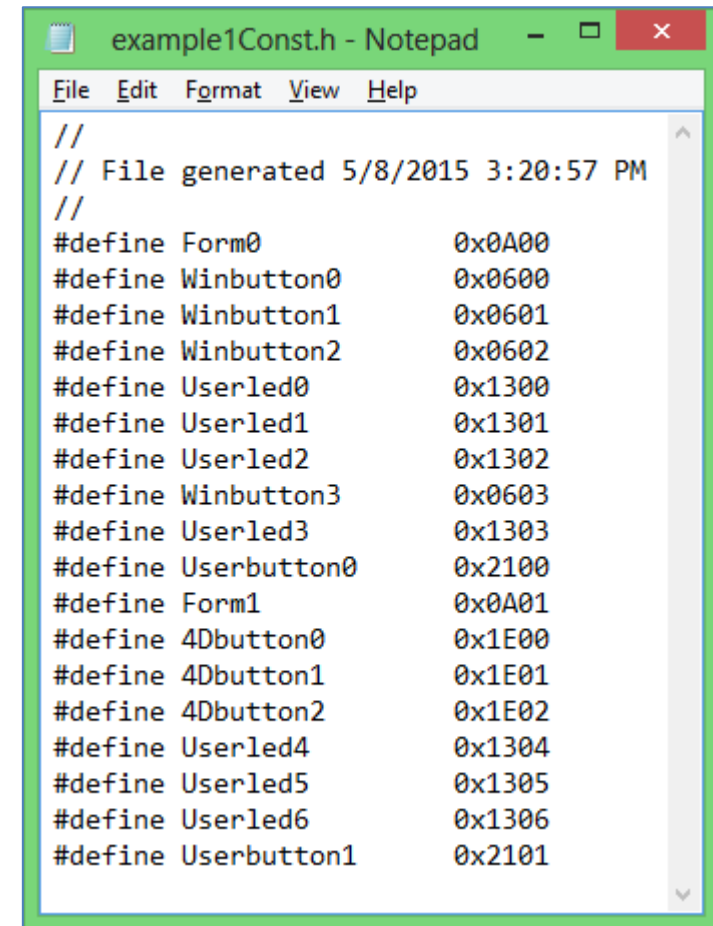
Go to the folder where the project files are located. For example, the files for this project are located in the folder shown below. The generated header file is

<project name> + “Const.h”

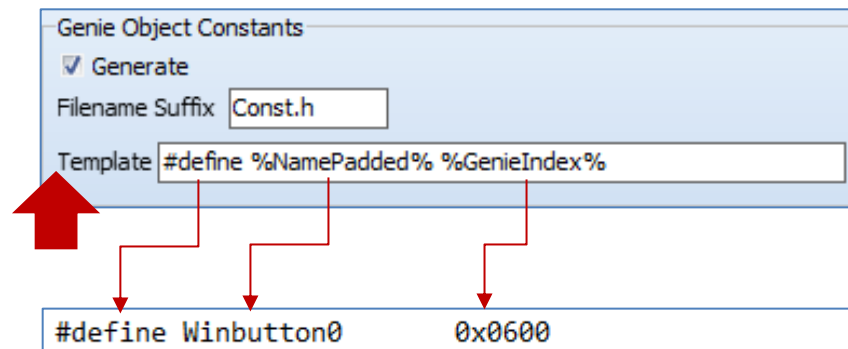
where “Const.h” is the default suffix set in Workshop.



### Open the Header File



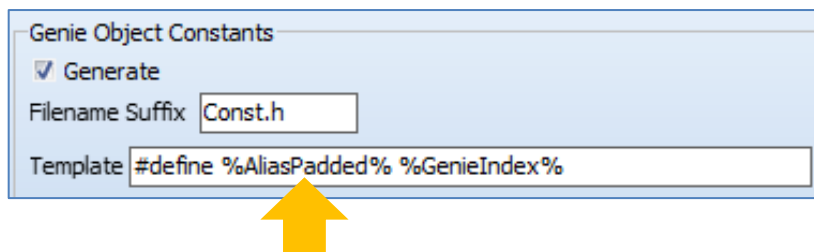
Note that the syntax follows that which was set in the “**Template**” field. We analyse the second item in the list.



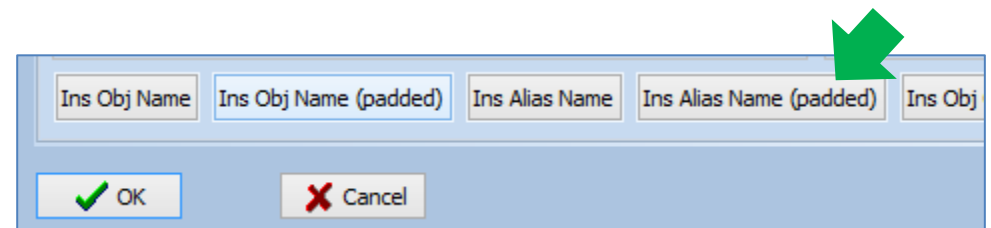
Note how the elements enclosed by a pair of percentage symbols are replaced with the actual values.

### Replace the Object Name with the Alias

Now go back to Workshop and change the value of the template field to that shown below.



Note that the available elements are listed below the Generated Files window. Click OK to make the new setting/s effective.



Compile and build the project again. The generated file should now be updated accordingly.



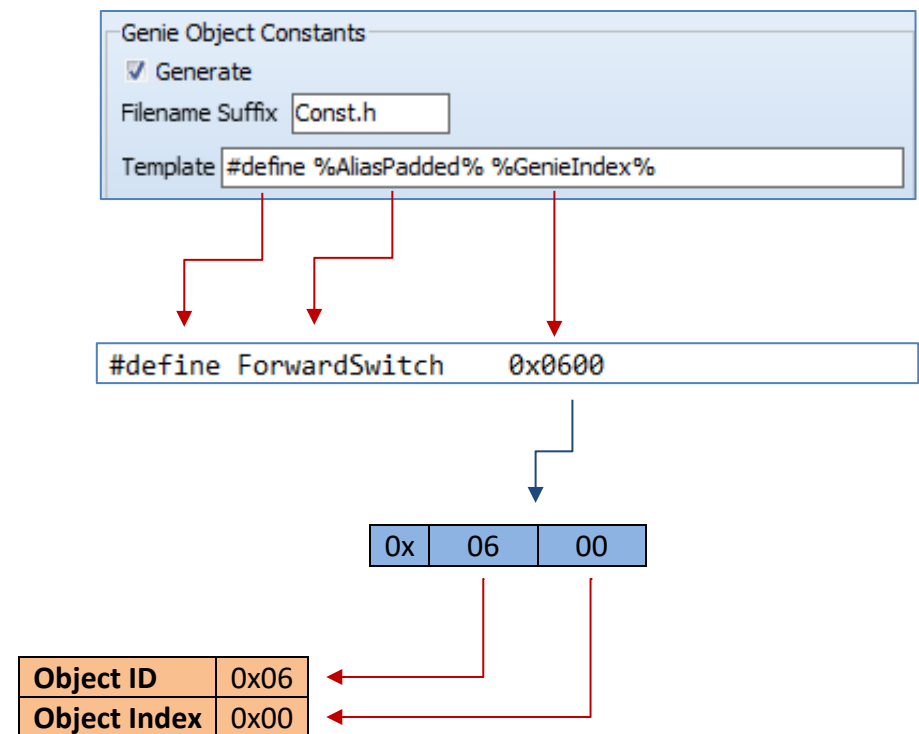
```

example1Const.h - Notepad
File Edit Format View Help
//
// File generated 5/8/2015 2:54:43 PM
//
#define Form0          0x0A00
#define ForwardSwitch  0x0600
#define TurnLeftSwitch 0x0601
#define TurnRightSwitch 0x0602
#define ForwardLight   0x1300
#define TurnLeftLight  0x1301
#define TurnRightLight 0x1302
#define ReverseSwitch  0x0603
#define ReverseLight   0x1303
#define Userbutton0     0x2100
#define Form1          0x0A01
#define FrontSwitch    0x1E00
#define MiddleSwitch   0x1E01
#define RearSwitch     0x1E02
#define FrontLight     0x1304
#define MiddleLight    0x1305
#define RearLight      0x1306
#define Userbutton1    0x2101

```

### The Genie Index Element

The Genie index element in the template field will be replaced with a 16-bit hexadecimal value. The high byte of this value is the Genie object ID. The low byte is the Genie object index. To illustrate using the second item in the header file:



The object ID for a winbutton object is 0x06. The section “**3.3. Object Summary Table**” of the [ViSi Genie Reference Manual](#) lists all Genie objects and their IDs. The index “0x00” means that ForwardSwitch is the first winbutton object of the project.

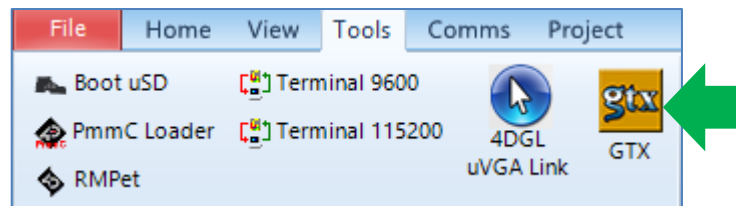
## Identify the Messages

The display module is going to send and receive messages to and from 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.

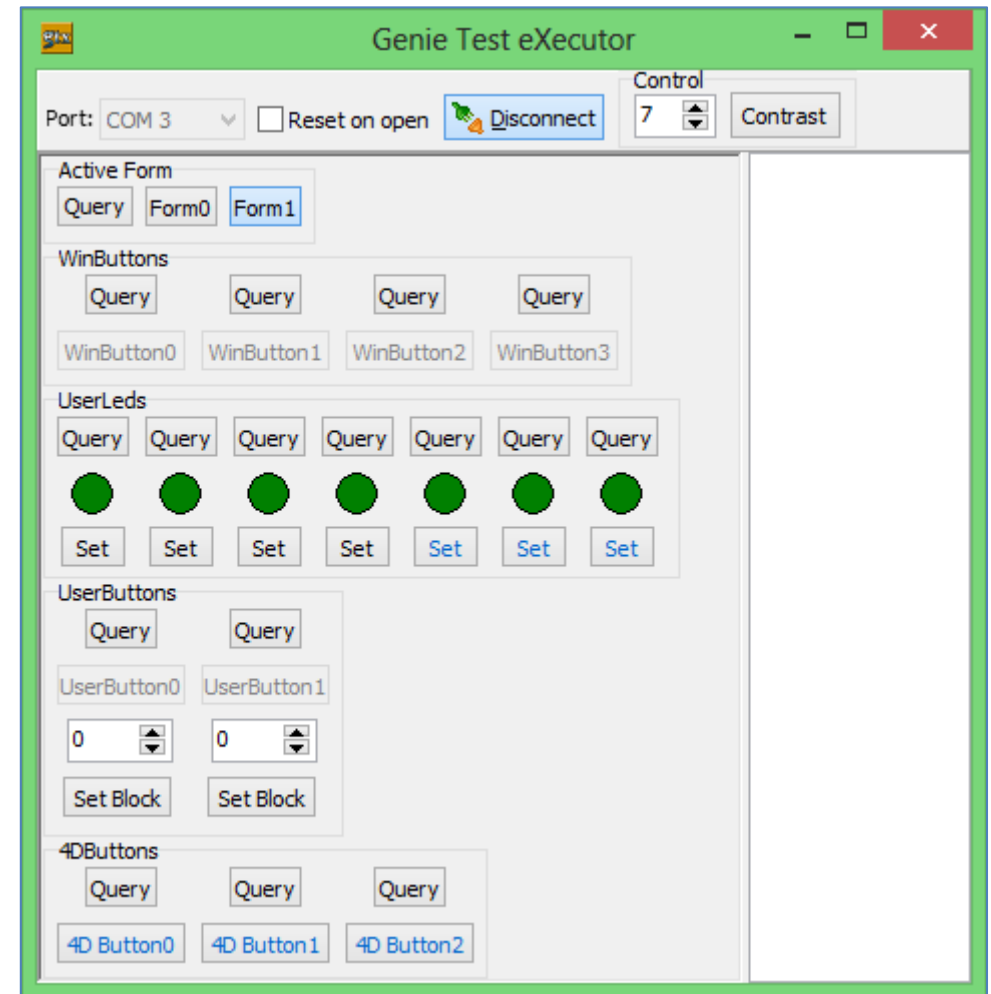
### Use the GTX Tool to Analyse the Messages

Using the GTX or **Genie Test eXecutor** tool is one option to get the messages sent by the display 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.

Under the Tools menu click on the GTX tool button.



The Genie Test eXecutor window appears.



## Receive a Message from the Display Module

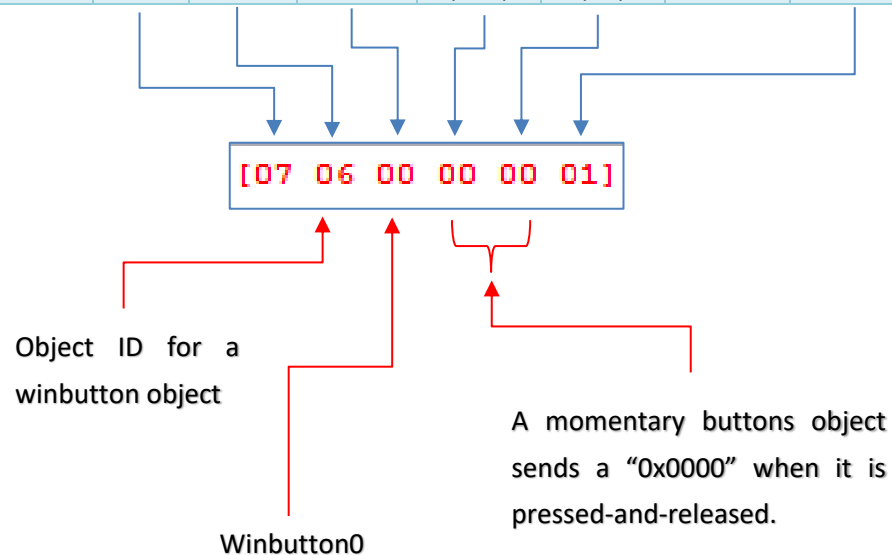
### REPORT\_EVENT Message from a Winbutton Object

On the display module, press Winbutton0. The display module will send a REPORT\_EVENT message to the PC.

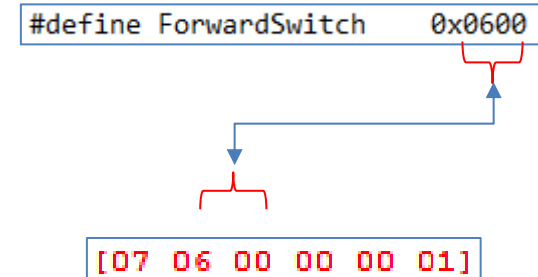
Winbutton Change 12:27:22.300 [07 06 00 00 00 01]

The format of this message is:

Command	Code	Para- meter 1	Para- meter 2	Para- meter 3	Para- meter 4	Para- meter N	Checksum
REPORT_ EVENT	0x07	Object ID	Object Index	Value (msb)	Value (lsb)	-	Checksum



Compare the Genie index element value of Winbutton0 in the header file to the received message.



When writing the source code for a program that will run on the host, the user can include the header file and used the defined constants when evaluating REPORT\_EVENT messages received from the display module. To illustrate using a pseudo C code:

```
#include <example1Const.h>
...

void eventHandler (uint8_t cmd, uint8_t id,
uint8_t index, uint16_t value){
    if(cmd == 0x07){ // if a REPORT_EVENT message
        if(id == (ForwardSwitch >> 8)){//if a winbutton
            if(index == ForwardSwitch){//if Winbutton0
                //do something here
            }
        }
    }
}
```

The above function gets called when a message is received from the display module. Assume that the calling function parses the elements of the message received from the serial port and passes them as arguments to *eventHandler(...)* accordingly.

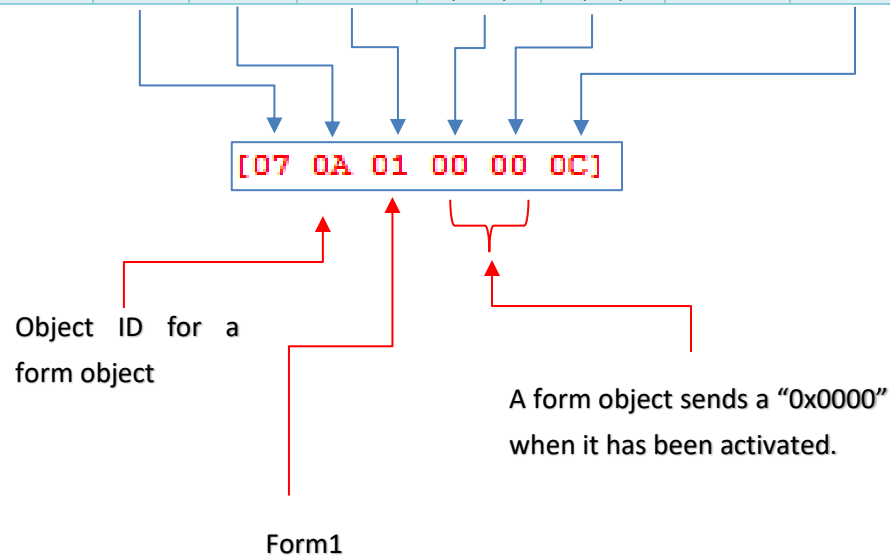
### REPORT\_EVENT Message from a Form Object

On the display module, press Userbutton0. The program on the display module will navigate to the second form and will send a REPORT\_EVENT message to the PC.

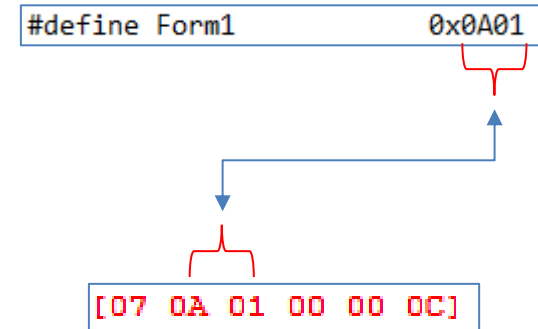
Form Change 12:48:22.210 [07 0A 01 00 00 0C]

The format of this message is:

Command	Code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter N	Checksum
REPORT_EVENT	0x07	Object ID	Object Index	Value (msb)	Value (lsb)	-	Checksum



Compare the Genie index element value of Form1 in the header file to the received message.



Example of usage through a pseudo C code:

```
#include <example1Const.h>
...

void eventHandler (uint8_t cmd, uint8_t id,
uint8_t index, uint16_t value){
    if(cmd == 0x07){ // if a REPORT_EVENT message
        if(id == (Form1 >> 8)){//if a form object
            if(index == Form1){//if Form1
                //do something here
            }
        }
    }
}
```

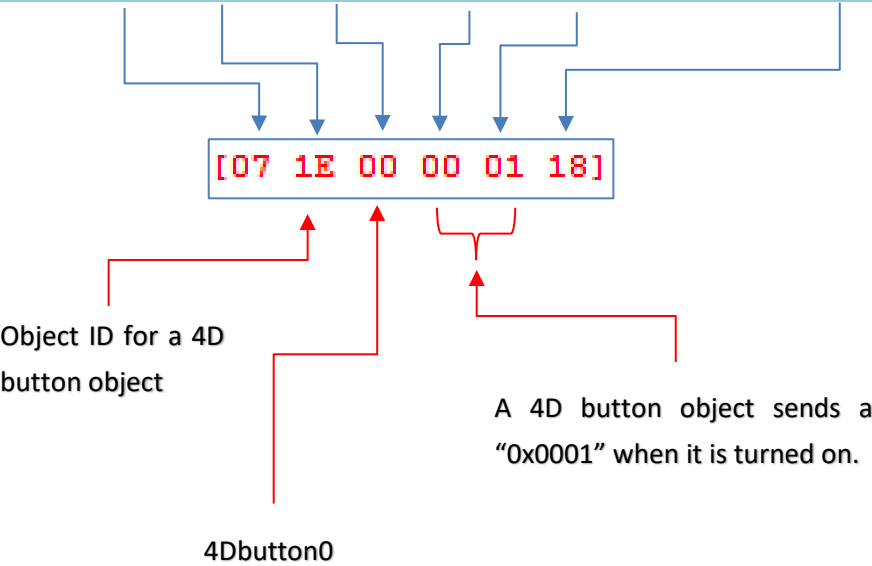
REPORT\_EVENT Message from a 4D Button Object - On

On Form1 of the display module, press 4Dbutton0. The program on the display module will send a REPORT\_EVENT message to the PC.

```
4Dbutton Change 12:54:53.643 [07 1E 00 00 01 18]
```

The format of this message is:

Command	Code	Para- meter 1	Para- meter 2	Para- meter 3	Para- meter 4	Para- meter N	Checksum
REPORT_ EVENT	0x07	Object ID	Object Index	Value (msb)	Value (lsb)	-	Checksum



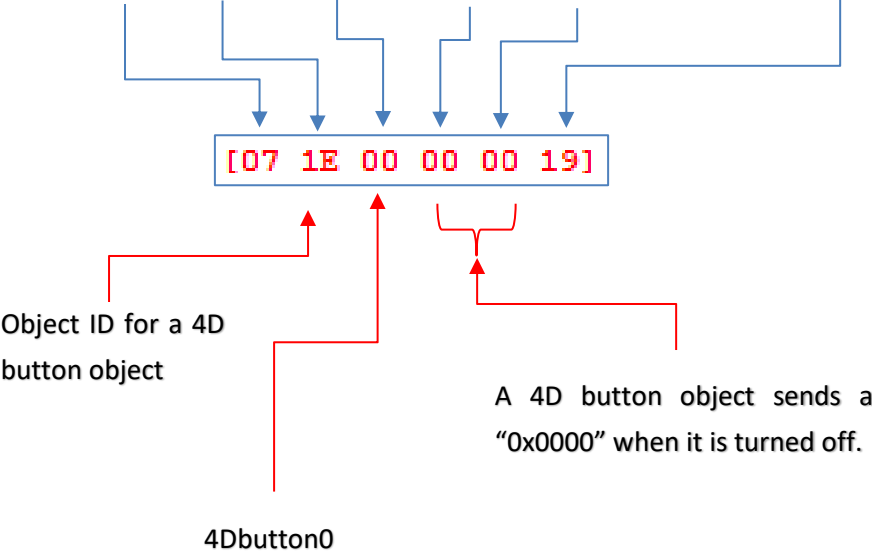
REPORT\_EVENT Message from a 4D Button Object - Off

On Form1 of the display module, press 4Dbutton0 again to turn it off. The program on the display module will send a REPORT\_EVENT message to the PC.

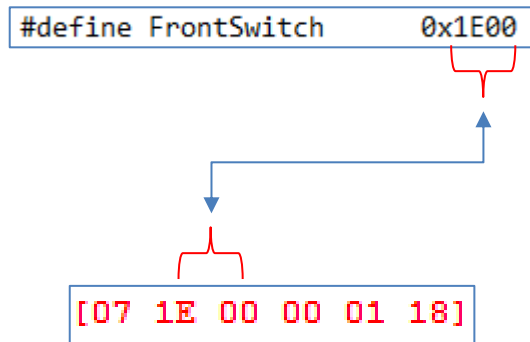
```
4Dbutton Change 12:58:58.223 [07 1E 00 00 00 19]
```

The format of this message is:

Command	Code	Para- meter 1	Para- meter 2	Para- meter 3	Para- meter 4	Para- meter N	Checksum
REPORT_ EVENT	0x07	Object ID	Object Index	Value (msb)	Value (lsb)	-	Checksum



Compare the Genie index element value of 4Dbutton0 in the header file to the received message.



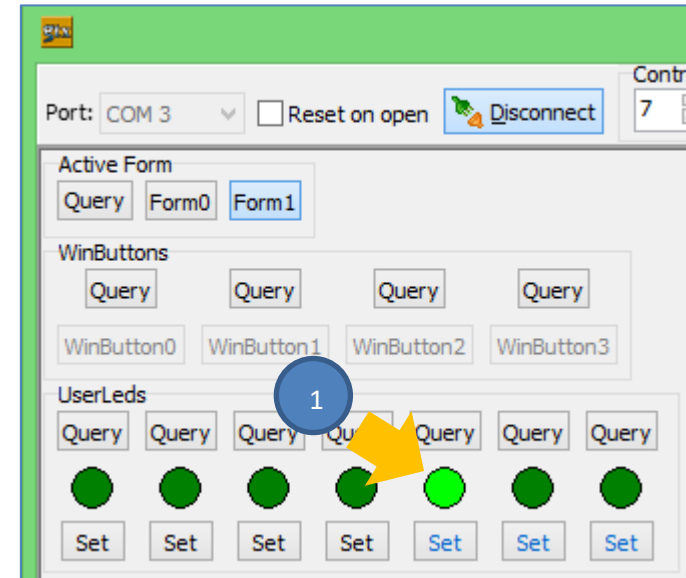
Example of usage through a pseudo C code:

```
#include <example1Const.h>
...
void eventHandler (uint8_t cmd, uint8_t id,
uint8_t index, uint16_t value){
    if(cmd == 0x07){ // if a REPORT_EVENT message
        if(id == (FrontSwitch >> 8)){//if a 4D button
            if(index == FrontSwitch){//if 4Dbutton0
                //do something here
            }
            else if(index == RearSwitch){//if 4Dbutton2
                //do something here
            }
            //set other conditions here
        }
    }
}
```

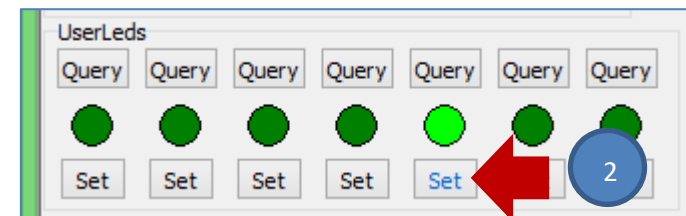
## Send a Message to the Display Module

### Send a WRITE\_OBJ Message to a User LED Object

On the GTX window, click on the green circle for Userled4.



Click on the "Set" button below the green circle.

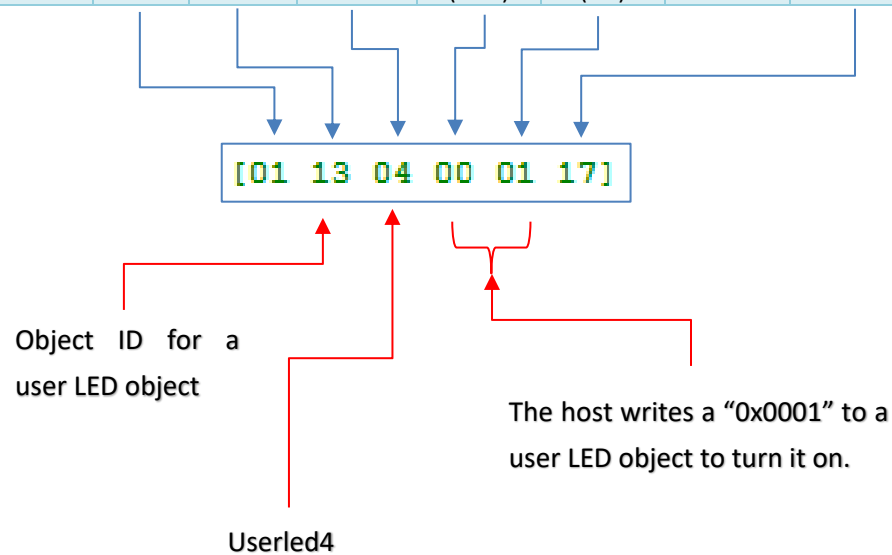


The GTX tool sends a WRITE\_OBJ message to the display module. Userled4 on the display module should now turn on.

```
Set Userled Value 13:43:08.132 [01 13 04 00 01 17]
```

The format of this message is:

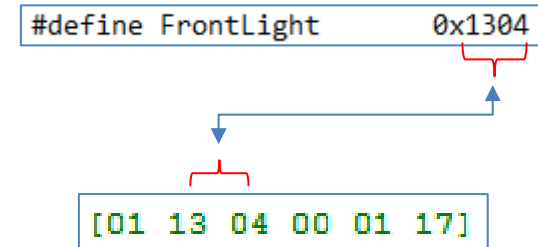
Command	Code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter N	Checksum
WRITE_OBJ	0x01	Object ID	Object Index	Value (msb)	Value (lsb)	-	Checksum



The display module sends back an ACK (acknowledgment) byte to the GTX tool.

```
ACK 13:43:08.156 [06]
```

Compare the Genie index element value of Userled4 in the header file to the received message.



When writing the source code for a program that will run on the host, the user can include the header file and use the defined constants when evaluating REPORT\_EVENT messages received from the display module. To illustrate using a pseudo C code:

```
void writeToDisplay(uint8_t cmd, uint8_t id,
uint8_t index, uint16_t value);
```

Above is a prototype for a function that sends a WRITE\_OBJ message to the display. Assume that it works and that the calling function just needs to specify the parameters. To turn on Userled4, we could write:

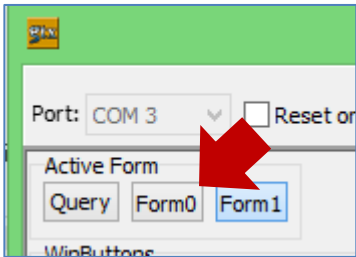
```
#include <example1Const.h>
...
writeToDisplay(0x01, FrontLight >> 8, FrontLight,
0x0001);
```

To turn off Userled4, we could write:

```
#include <example1Const.h>
...
writeToDisplay(0x01, FrontLight >> 8, FrontLight,
0x0000);
```

Send a WRITE\_OBJ Message to a Form Object

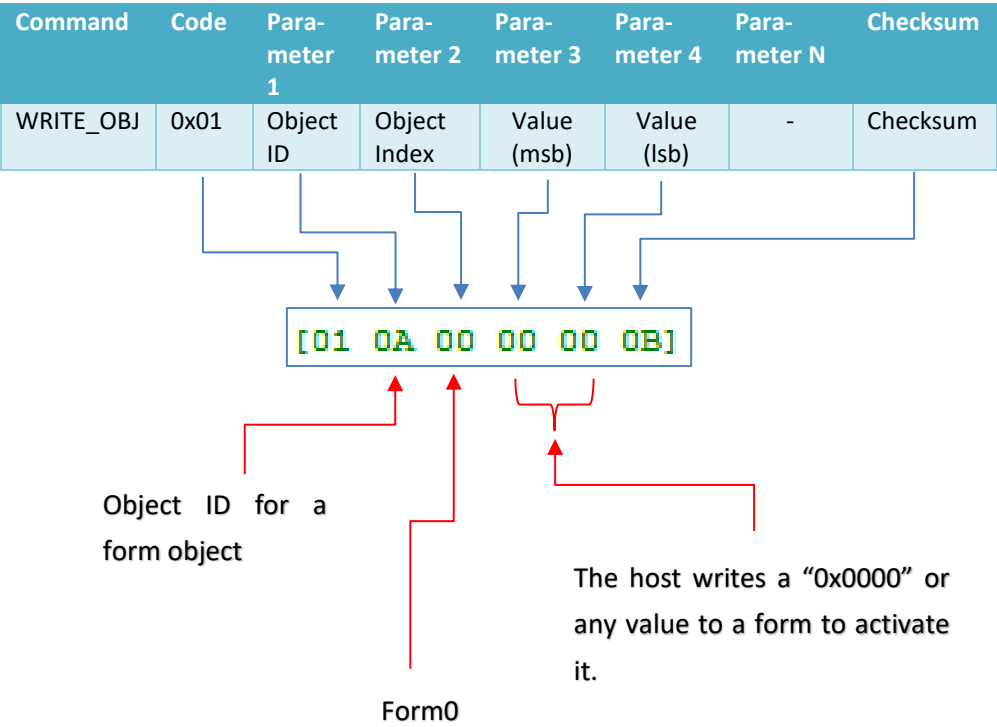
On the GTX window, click on the button “Form0” as indicated below.



The GTX tool sends a WRITE\_OBJ message to the display module. The program on the display module should now navigate to Form0.

```
Set Form Value 13:48:36.340 [01 0A 00 00 00 0B]
```

The format of this message is:

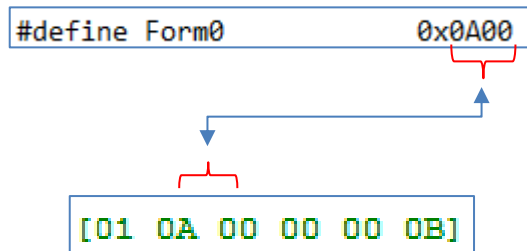


The display module sends back an ACK (acknowledgment) byte to the GTX tool.

```
ACK 13:48:36.437 [06]
```



Compare the Genie index element value of Userled4 in the header file to the received message.



Below are examples of usage through a pseudo C code.

To navigate to Form0:

```
#include <example1Const.h>
...
writeToDisplay(0x01, Form0 >> 8, Form0, 0x0000);
```

To navigate to Form1:

```
#include <example1Const.h>
...
writeToDisplay(0x01, Form1 >> 8, Form1, 0x0000);
```

## 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.