



ViSi Getting Started – First Project for Goldelox

DOCUMENT DATE: **21st April 2019**
DOCUMENT REVISION: **1.1**



Description

This application note shows how to program a 4D display module in the ViSi environment to make it print text and display an image on the screen.

Before getting started, the following are required:

- Any of the following 4D Goldelox display modules:

[uOLED-96-G2](#)

[uOLED-128-G2](#)

[uOLED-160-G2](#)

[uLCD-144-G2](#)

or any superseded module that supports the Serial environment

- [4D Programming Cable](#) / [μUSB-PA5/μUSB-PA5-II](#)
- [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

Description	2
Content	2
Application Overview	3
Setup Procedure	4
<i>Open a Project from the File Explorer</i>	4
<i>Open the Example in Workshop 4</i>	4
<i>How to Change the Target Display</i>	6
Create a New Project	7
<i>Launch Workshop 4</i>	7
<i>Create a New Project</i>	7
<i>Select ViSi</i>	9
Design the Project	10
<i>Default Code</i>	10
<i>Define the Platform</i>	10
<i>Include Files</i>	10
How to Open an Include File	10
<i>Constants</i>	11
<i>The Main Function</i>	12
Comments in 4DGL	13
Uncomment the uSD Card Initialization Routine	13
Print a String	13
The While Loop	14

Initialize the uSD Card	14
Delay	14
Clear the Screen	14
The Repeat-forever Loop	15
<i>Hello World</i>	15
Add a Static Text Object	15
Paste the Code for a Static Text Object	16
Run the Program	17
<i>Save the Project</i>	17
<i>Insert the uSD Card to the PC</i>	17
<i>Connect the Display Module to the PC</i>	18
<i>Compile and Download</i>	18
Proprietary Information	22
Disclaimer of Warranties & Limitation of Liability	22

Application Overview

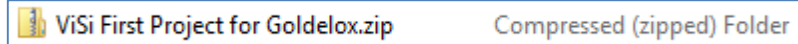
It is often difficult to design a graphical display without being able to see the immediate results of the application code. ViSi is the perfect environment that allows users to see the instant results of their desired graphical layout. There is a selection of inbuilt dials, gauges, and meters (called widgets) that can simply be dragged and dropped onto the simulated module display. From here, each object can have its properties edited and at the click of a button, all relevant code is produced in the user program.

This application note shows how to create a new project, how to select the target display module, how to connect a display module to the PC, and how to compile and download a simple “Hello-world” program to the target device. This application note also introduces the 4DGL (4D Graphics Language) and the basic use of the WYSIWYG (What-You-See-Is-What-You-Get) screen.

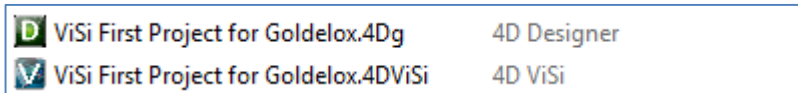
Setup Procedure

Open a Project from the File Explorer

This document comes with a demo ViSi project in a zip file.



In the file explorer window, extract the contents of the zip file to a desired location. The contents are a ViSi and a Designer files.



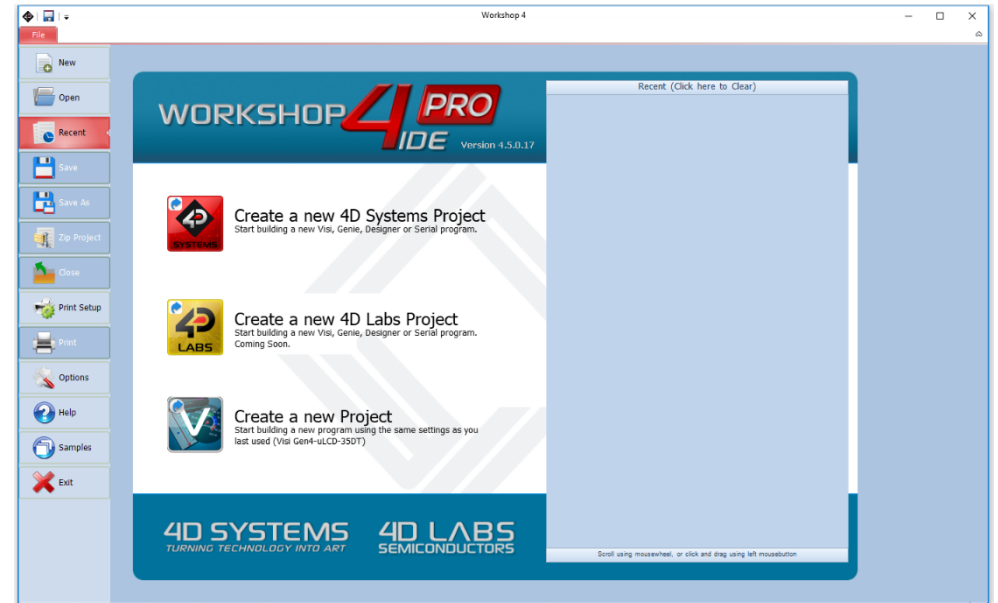
The user can open the project by double-clicking on any of the two files or by selecting them in Workshop 4. Users who want to learn how to create a new ViSi project may proceed to the next section.

Open the Example in Workshop 4

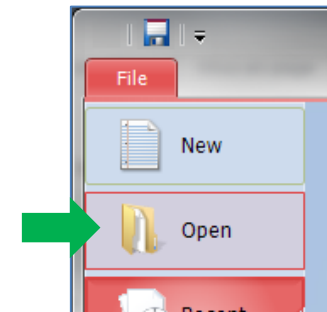
There is a shortcut for Workshop 4 on the desktop.



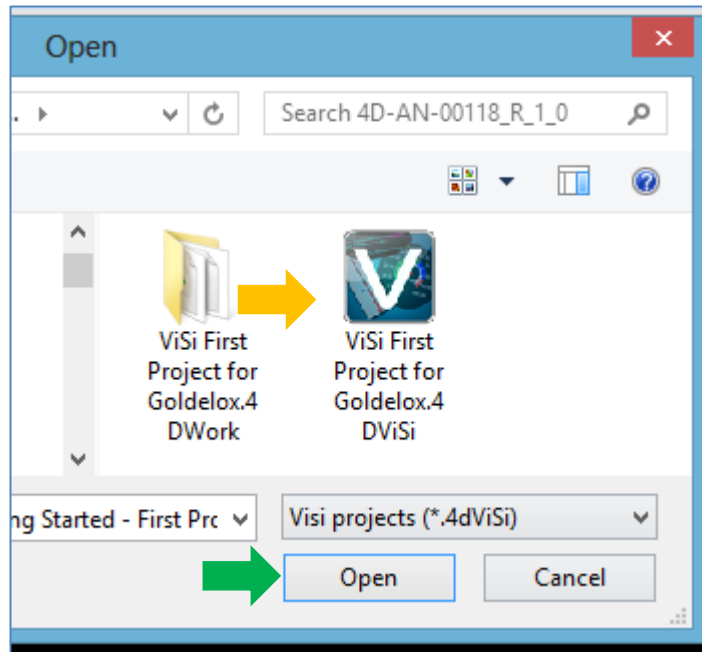
Launch Workshop 4 by double-clicking on the icon. Workshop 4 opens and displays the Recent page.



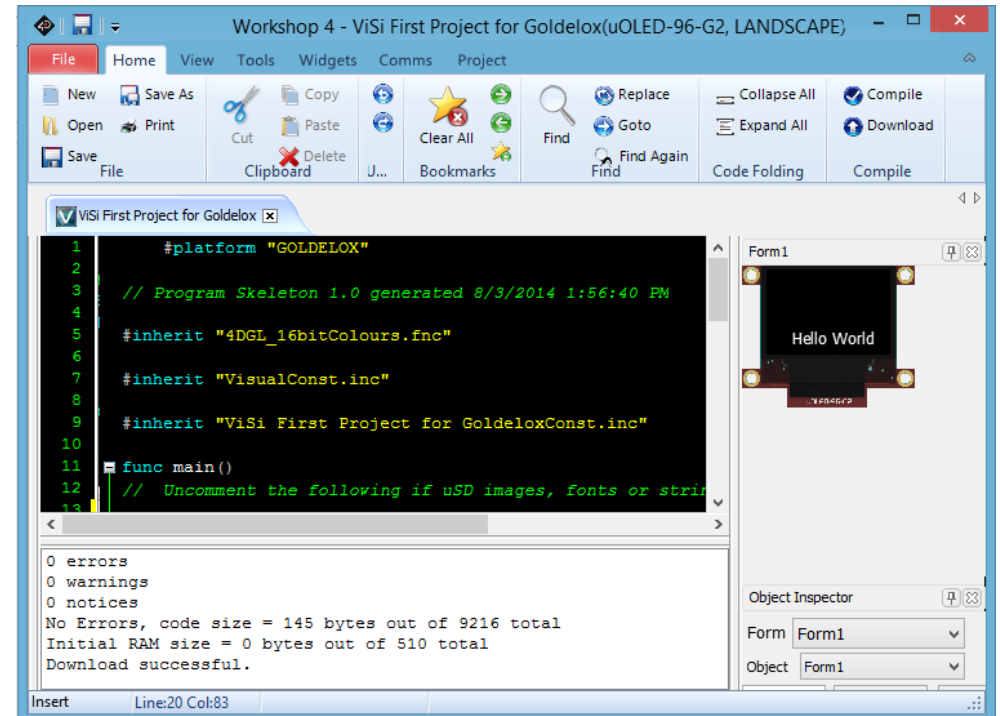
To load the existing project, click on Open.



A standard open window asks for a ViSi project. Select the demo file.

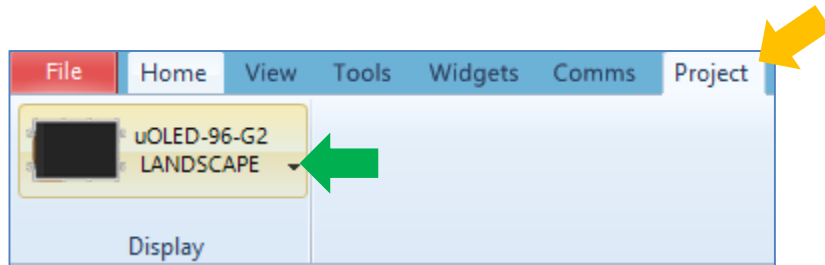


The project opens.

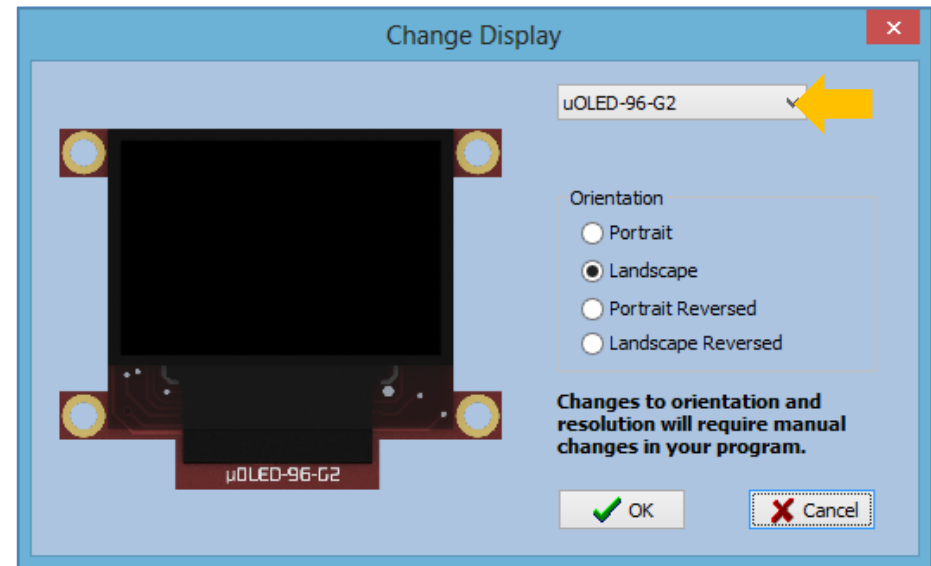


How to Change the Target Display

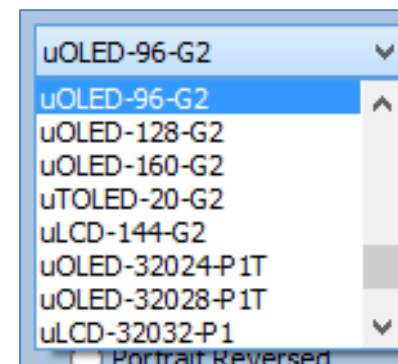
The attached project has its target display configured to be a uOLED-96-G2. To change the target device, go to the **Project** menu and click on the display button.



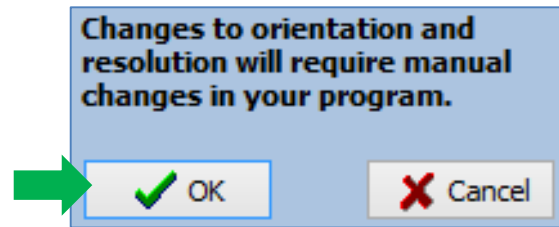
The Change Display window appears. Change the target display by clicking on the display name.



A drop-down menu will appear. Select your new target device.



Click OK to confirm when done.



Changing the orientation is done by manually editing the code. This will be discussed later.

Create a New Project

Launch Workshop 4

There is a shortcut for Workshop 4 on the desktop. Launch Workshop 4 by double-clicking on the icon.

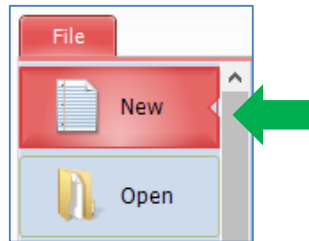


Create a New Project

Workshop 4 opens and displays the **Recent** page.



To create a new project, there are two options.
Click on the top left-most icon, New.



Or Click on the icon beside Create a new Project.



Create a new 4D Systems Project
Start building a new Visi, Genie, Designer or Serial program.

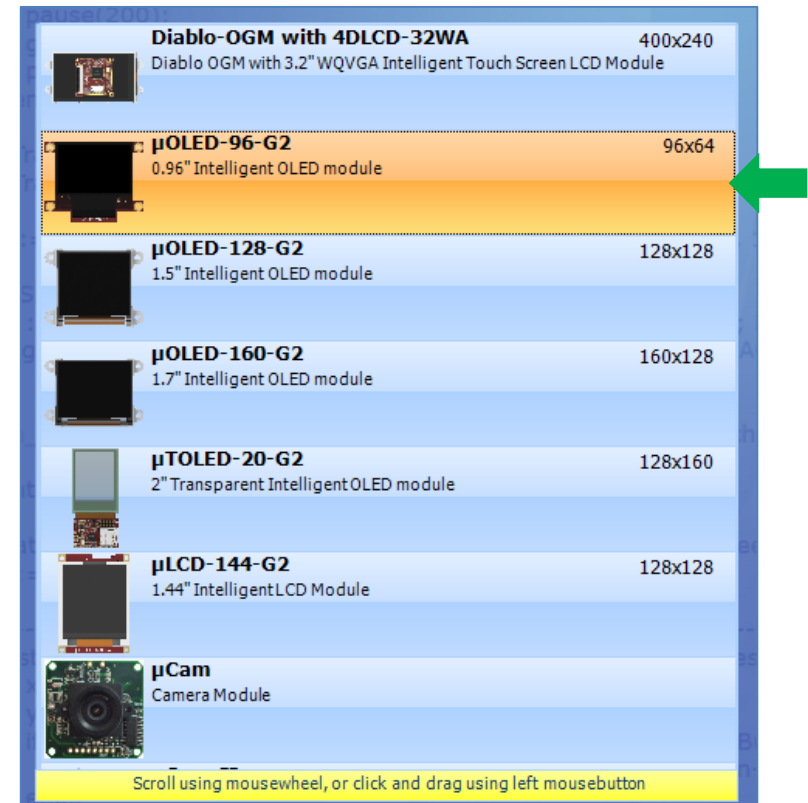


Create a new 4D Labs Project
Start building a new Visi, Genie, Designer or Serial program.
Coming Soon.

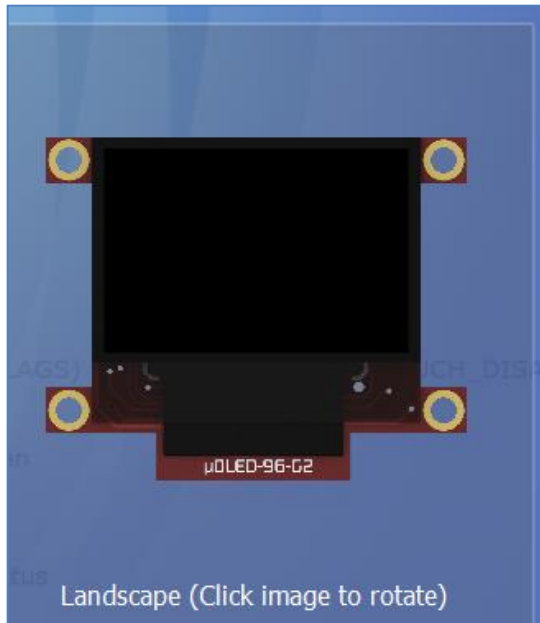


Create a new Project
Start building a new program using the same settings as you last used (Visi-Genie Gen4-uLCD-50DT)

The Choose-Your-Product window appears. Select the appropriate screen and preferred orientation. The screen used in this example is a **uOLED-96-G2 (landscape orientation)**.



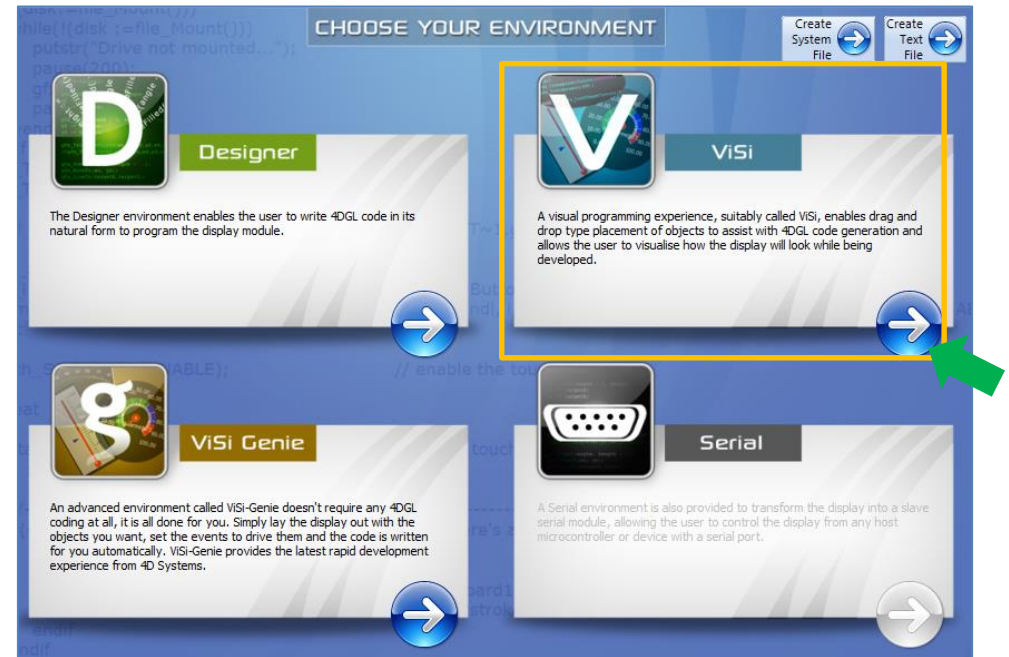
Select the desired orientation by clicking on the display on the right part of the Choose-Your-Product window.



The image of the display rotates as you click it. When done, click on the next button on the lower right part of the Choose-Your-Product window.

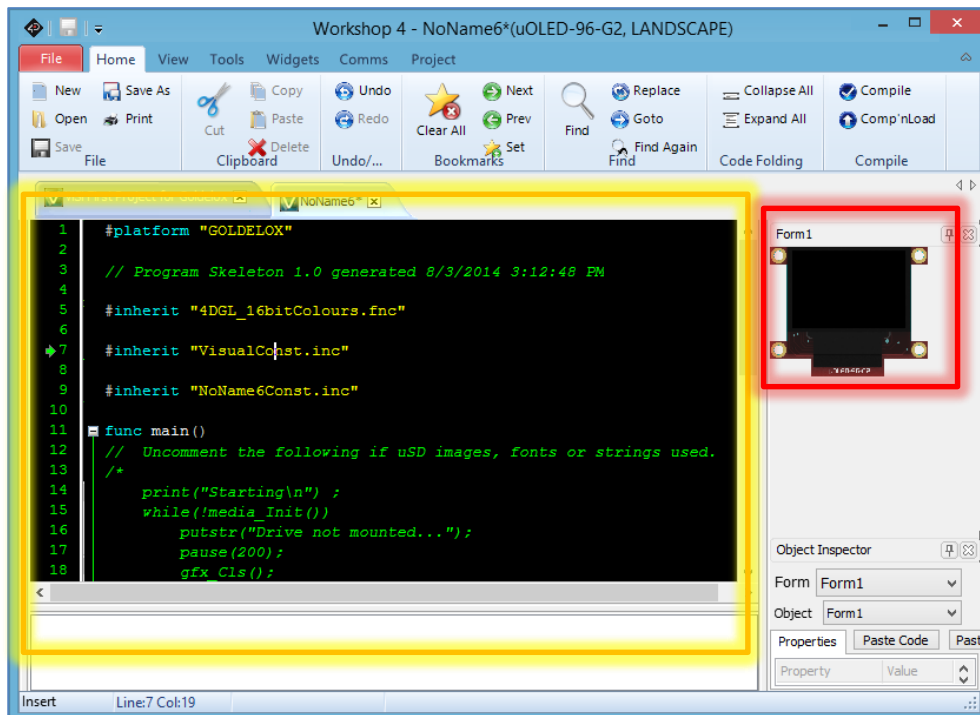


Select ViSi



Design the Project

Everything is now ready to start designing the project. Workshop 4 displays the code (left side – orange box) and the WYSIWYG screen (right side – red box).



Default Code

Workshop 4 automatically provides a default code – a program skeleton to which developers can simply add their codes. The program skeleton contains basic parts needed to start designing an application.

Define the Platform

The first line of the default code defines the platform or the processor.

```
1 #platform "GOLDELOX"
```

Include Files

To include a source file in 4DGL, use the pre-processor directive “#inherit”. The program skeleton has three include files.

```

5 #inherit "4DGL_16bitColours.fnc"
6
7 #inherit "VisualConst.inc"
8
9 #inherit "NoName6Const.inc"

```

The user can view the contents of an include file by following the instructions below.

How to Open an Include File

1. Put the cursor on the filename.

```

5 #inherit "4DGL_16bitColours.fnc"
6
7 #inherit "VisualConst.inc"
8
9 #inherit "NoName6Const.inc"

```

- Click the right mouse button and a menu will appear. Select the first option (**Open file at Cursor**).

```

5 #inherit "4DGL_16bitColours.fnc"
6
7 #inherit "VisualConst..."
8
9 #inherit "NoName6Const..."
10
11 func main()
12 // Uncomment the followi...
13 /*
14 print("Starting\n")

```

The context menu is open, showing options: Open file at Cursor (Ctrl+Alt+O), Undo (Ctrl+Z), Redo (Ctrl+Y), Copy (Ctrl+C), Cut (Ctrl+X), and Paste (Ctrl+V). A green arrow points to the 'Open file at Cursor' option.

- Workshop 4 now opens the file 4DGL_16bitColours.fnc.

```

5
6
7
8 #CONST
9 ALICEBLUE 0xF7DF
10 ANTIQUEWHITE 0xFF5A
11 AQUA 0x07FF
12 AQUAMARINE 0x7FFA
13 AZURE 0xF7FF
14 BEIGE 0xF7BB
15 BISQUE 0xFF38
16 BLACK 0x0000

```

The editor shows the file 4DGL_16bitColours.fnc open. The code defines a block of constants from line 8 to 16. A green arrow points from the context menu in the previous image to this screenshot.

The file contains **constants** and their values. **Values of constants do not change throughout the duration of the program.**

Constants

The file **4DGL_16bitColours.fnc** contains hexadecimal values of 140 commonly used colour constants. Scroll down to see all the defined colour constants. Take note of lines 8 and 149 below. This is how a block of constants is declared in 4DGL.

```

5
6
7
8 #CONST
9 ALICEBLUE 0xF7DF
10 ANTIQUEWHITE 0xFF5A
11 AQUA 0x07FF
12 AQUAMARINE 0x7FFA
13 AZURE 0xF7FF
14 BEIGE 0xF7BB
15 BISQUE 0xFF38
16 BLACK 0x0000
...
141 TOMATO 0xFB08
142 TURQUOISE 0x471A
143 VIOLET 0xEC1D
144 WHEAT 0xF6F6
145 WHITE 0xFFFF
146 WHITESMOKE 0xF7BE
147 YELLOW 0xFFE0
148 YELLOWGREEN 0x9E66
149 #END
150

```

The editor shows the full list of 140 color constants from line 8 to 149. The constants are listed in two columns. A green arrow points from the previous screenshot to this one.

The include file "**VisualConst.inc**" contains constants for line patterns. Here each of the constants is defined as a single-line entry.

```

1 // Line Patterns
2 #constant LPCOARSE 0xF0F0
3 #constant LPMEDIUM 0x3333
4 #constant LPFINE 0xAAAA
5 #constant LPDASHDOT 0x03CF
6 #constant LPDASHDOTDOT 0x0333
7 #constant LPSOLID 0x0000

```

The include file "**NoName6Const.inc**" does not exist prior to the compilation of the project. The final name of this include file will be that of the project when it is saved. This include file will contain, among others, constant values of memory locations.

The Main Function

Lines 5 to 14 make up the main function of the program.

```

1 #platform "GOLDELOX"
2
3 // Program Skeleton 1.0 generated 8/3/2014 3
4
5 #inherit "4DGL_16bitColours.fnc"
6
7 #inherit "VisualConst.inc"
8
9 #inherit "NoName6Const.inc"
10
11 func main()
12 // Uncomment the following if uSD images, f
13 /*
14 print("Starting\n") ;
15 while(!media_Init())
16     putstr("Drive not mounted...");
17     pause(200);
18     gfx_Cls();
19     pause(200);
20 wend
21 */
22
23 repeat
24     forever
25 endfunc

```

main

Note that the main function block starts with

```
5 func main()
```

and ends with

```
14 endfunc
```

This is how a function is defined in 4DGL.

Comments in 4DGL

Single-line comments in 4DGL look like as shown below.

```
// Uncomment the following if uSD images, fonts or strings used.
```

A block comment starts with “/*” and ends with “*/”, as shown below.

```
13 /*
14     print("Starting\n") ;
15     while(!media_Init())
16         putstr("Drive not mounted...");
17         pause(200);
18         gfx_Cls();
19         pause(200);
20     wend
21 */
```

Uncomment the uSD Card Initialization Routine

This application needs a μ SD card. Before the uSD card can be accessed, it has to be initialized first. Observe that in the main function, an entire block is commented out. This block (lines 14 to 20) is a uSD-card-initialization routine and it defines the behaviour of the program while waiting for a μ SD card to be inserted. Basically the routine flashes the string “Drive not mounted” while no uSD card is detected and successfully initialized. Remove the block comment symbols indicated below.

```
11 func main()
12 // Uncomment the following if uSD images, fonts or strings used.
13 /*
14     print("Starting\n") ;
15     while(!media_Init())
16         putstr("Drive not mounted...");
17         pause(200);
18         gfx_Cls();
19         pause(200);
20     wend
21 */
```

The code screen will be updated accordingly, showing the block as an actual part of the code.

```
11 func main()
12 // Uncomment the following if uSD images, fonts or strings used.
13
14     print("Starting\n") ;
15     while(!media_Init())
16         putstr("Drive not mounted...");
17         pause(200);
18         gfx_Cls();
19         pause(200);
20     wend
```

Print a String

The line

```
14     print("Starting\n") ;
```

will print the text “Starting” on the screen followed by a new line.

The While Loop

Lines 15 to 20 make up a while loop.

While loop

```

11 func main()
12 // Uncomment the following if uSD images, fonts o
13
14 print("Starting\n") ;
15 while(!media_Init())
16     putstr("Drive not mounted...");
17     pause(200);
18     gfx_Cls();
19     pause(200);
20 wend
  
```

Note that the while loop starts with

```
15 while(!media_Init())
```

and ends with

```
20 wend
```

A **loop** in a program performs instructions repetitively. The while loop has the basic syntax

```
while(condition)
    [statements]
wend
```

The statements or instructions inside the block are executed as long as the condition is true.

Initialize the uSD Card

The function

```
media_Init()
```

initializes the uSD card for further operations. The uSD card is connected to the SPI (Serial Peripheral Interface) of the Goldelox processor. The function returns a value of '1' if a memory card is present and successfully initialized, and a value of '0' otherwise.

The function **putstr()** is another command for printing strings.

```
putstr("Drive not mounted...");
```

Delay

The function

```
pause(200);
```

delays the execution of the program for 200 milliseconds.

Clear the Screen

To clear the screen, use the function

```
gfx_Cls();
```

The Repeat-forever Loop

The repeat-forever loop is another kind of loop. It starts with

```
23   repeat
```

and ends with

```
24   forever
```

The program will execute the statements between lines 23 and 24 indefinitely. Here the program actually does nothing indefinitely after printing the text "Hello World" since no instructions exist between lines 23 and 24. If this empty loop is omitted, the program exits the main function.

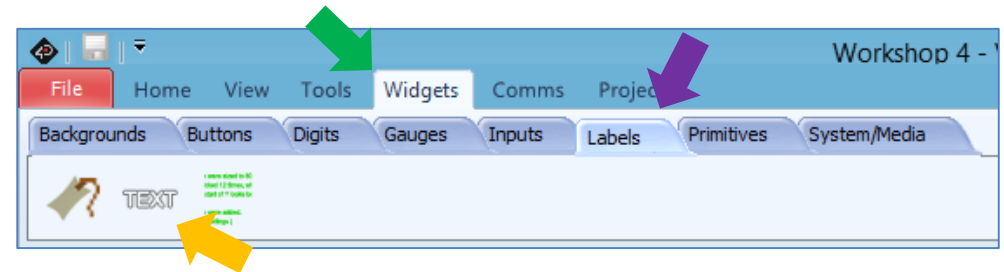
Hello World

Up to this point, the program does nothing after it has detected and initialized a uSD card. To make it print "Hello World" on the screen, write

```
11 func main()
12 // Uncomment the following if uSD imag
13
14 print("Starting\n");
15 while(!media_Init())
16     putstr("Drive not mounted...");
17     pause(200);
18     gfx_Cls();
19     pause(200);
20 wend
21
22 print("Hello World");
23
24     repeat
25     forever
26 endfunc
```

Add a Static Text Object

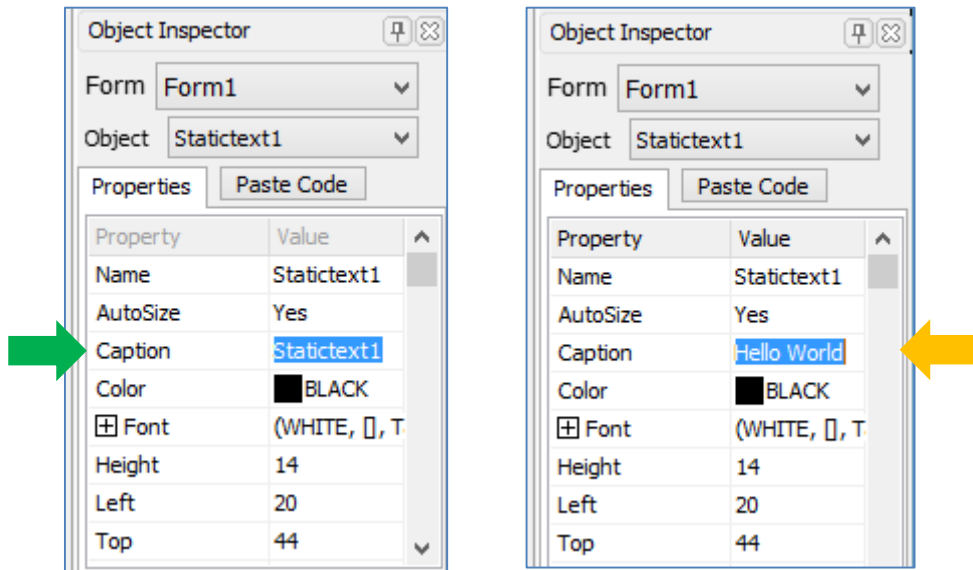
Go to the Widgets menu, select the Labels pane, and click on the static text icon.



Once the static text icon is selected, click on the WYSIWYG screen to place it. Drag the object to the bottom part of the WYSIWYG screen.



The Object Inspector shows the different properties of the object. The object has the name "Statictext1", which means that it is the first static text object added to the project. By default, an object has the same name and caption. Change the value of the property "Caption" from "Statictext1" to "Hello World".



The WYSIWYG screen is updated accordingly.



Paste the Code for a Static Text Object

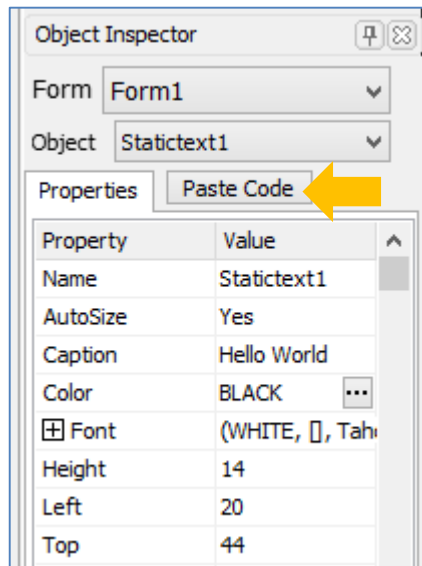
In the code area, place the cursor somewhere between the “print(“Hello World”);” statement and the repeat-forever loop.

```

11 func main()
12 // Uncomment the following if uSD image
13
14 print("Starting\n") ;
15 while(!media_Init())
16     putstr("Drive not mounted...");
17     pause(200);
18     gfx_Cls();
19     pause(200);
20 wend
21
22 print("Hello World");
23
24
25
26     repeat
27     forever
28     endfunc

```

With Statictext1 still selected, click on the “Paste Code” button.



Two new lines are added to the code. These are the commands for showing the object Statictext1 on the screen.

```

22     print("Hello World");
23
24
25     // Statictext1 1.0 generated 8/3/2014 8:44:18 PM
26     media_SetAdd(iStatictext1H, iStatictext1L) ;
27     media_Image(20, 44) ;           // show image
28
29
30     repeat
31     forever
32     endfunc

```

After executing the uSD-card-initialization routine, the program will now print the string "Hello World" and will display the object Statictext1.

Run the Program

Save the Project

Save the program with the desired file name first.

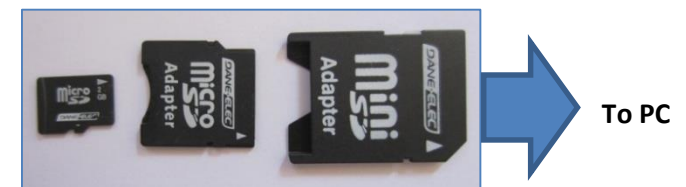


Insert the uSD Card to the PC

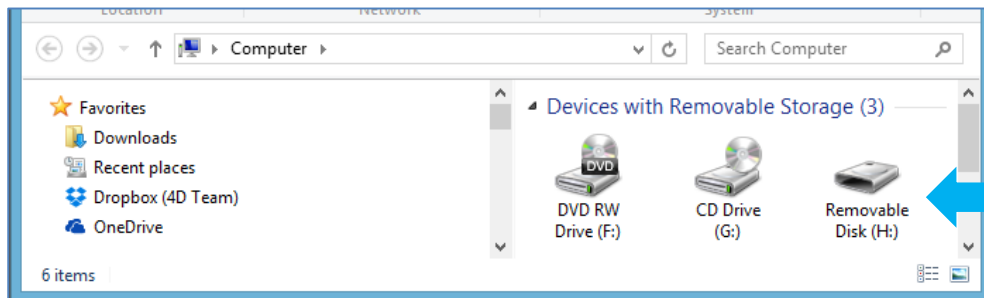
Insert the μ SD card into the USB adaptor and plug the USB adaptor into a USB port of the PC.



OR insert the μ SD card into a μ SD-to-SD card converter and plug the SD card converter into the SD card slot of the PC.



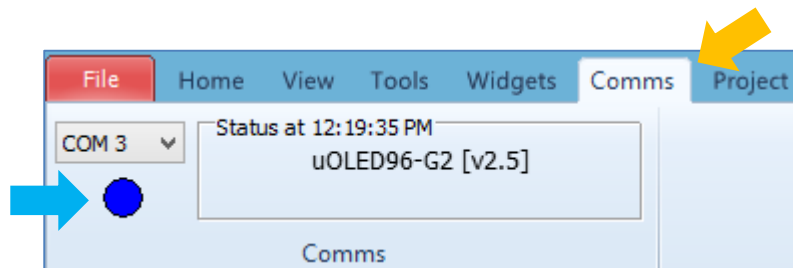
Check if the μ SD card is mounted, here it is mounted as drive H:



Warning: The ViSi application writes the graphics files in to the μ SD card in RAW format for GOLDELOX display modules. In other words, the FAT formatting will be removed. Please make sure you backup any important files/data you have on the μ SD card.

Connect the Display Module to the PC

Connect the display module to the PC using a [4D Programming Cable](#) or a [uUSB-PA5](#) adapter. Go to the Comms menu to check if the module is detected. The button should be blue in colour. Below is an example of how the Comms tab will look like if a uOLED-96-G2 is connected to the PC.

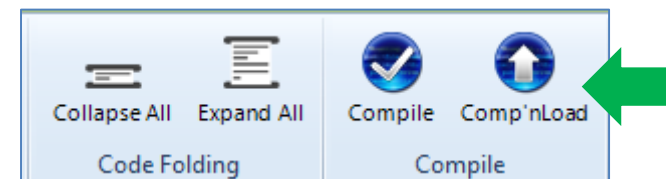


If not familiar with how to actually connect a 4D display to the PC using a 4D USB programming cable or a uUSB-PA5 and with how to update the firmware, the user should consult the application note below.

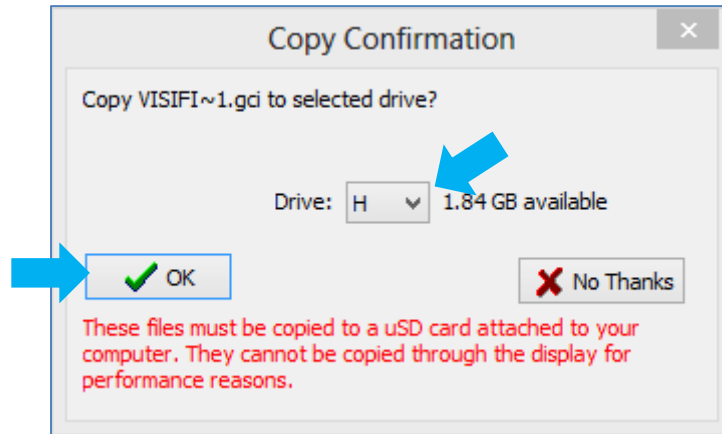
[General How to Update the PmmC for Goldelox](#)

Compile and Download

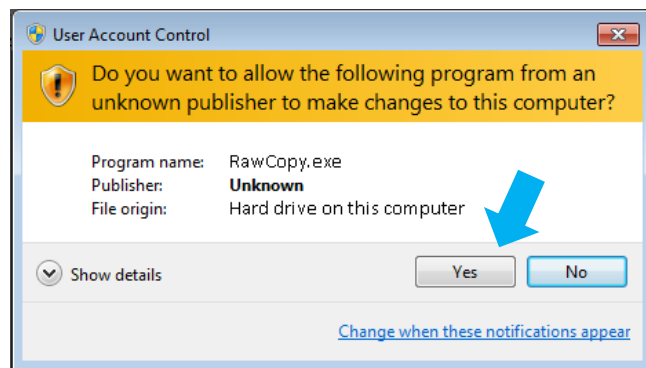
After making sure that the device is detected, go to the Home menu and click on the **Comp n'Load** or Compile and Download button.



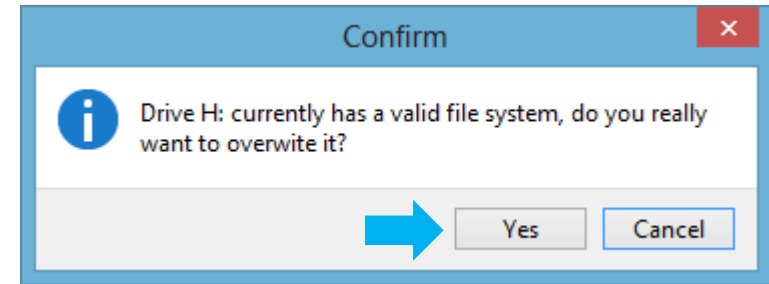
Workshop now builds the graphics files and copies them to the uSD card. The Copy Confirmation window appears. The user will be prompted to choose the correct drive for the memory card. Now choose the correct drive by clicking on the drop down arrow. Then click on OK.



Depending on the user's PC User Account Control settings, Windows might ask for a confirmation to run the program RawCopy.exe. This program copies the graphics files to the μ SD card in RAW format. Click Yes.



Another confirmation window appears. Click Yes only if you are ready to proceed.



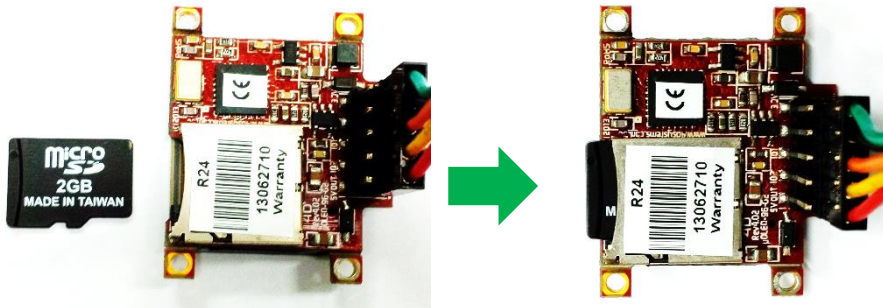
Workshop copies the graphics files to the μ SD card and downloads the program to the display module. The message box will look like as shown below after a successful download.

```
0 errors
0 warnings
0 notices
No Errors, code size = 217 bytes out of 9216 total
Initial RAM size = 0 bytes out of 510 total
Download successful.
```

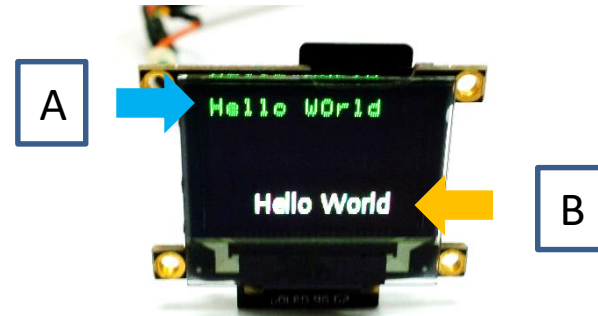
The program now runs and waits for a uSD card.



Remove the μ SD card from the PC and insert it into the μ SD card slot of the display module.



The program prints the string “Hello World” and displays Statictext1.



Item A is a string printed as a result of the instruction,

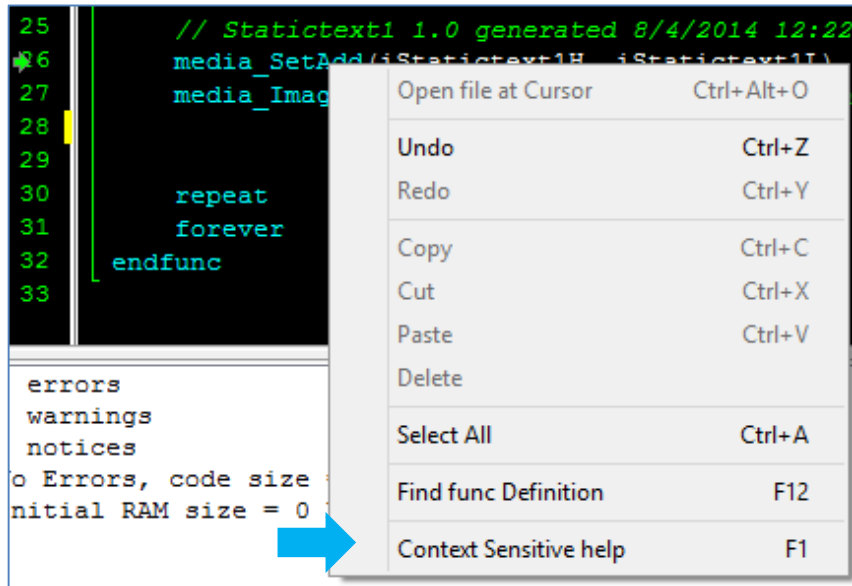
```
22 print("Hello World");
```

Item B is an image retrieved from the uSD card and is displayed as a result of the instructions

```
25 // Statictext1 1.0 generated 8/3/2014 8:44:
26 media_SetAdd(iStatictext1H, iStatictext1L)
27 media_Image(20, 44) ; // show in
```

These commands are discussed in another application note. Interested users may consult the internal functions reference manual. To open the manual, put the cursor on the desired command, click on the right mouse button, then choose “Context Sensitive help”.

```
25 // Statictext1 1.0 generated 8/4/2014 12:2
26 media_SetAdd(iStatictext1H, iStatictext1L)
27 media_Image(20, 44) ; // show in
```



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.