



Designer Getting Started – First Project

DOCUMENT DATE: **16th APRIL 2019**
DOCUMENT REVISION: **1.1**



Description

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

Before getting started, the following are required:

- Any Picaso, Diablo16, or Goldelox display module. Visit www.4dsystems.com.au to see the latest products using any of these graphics processors.
- [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)
- [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.
- This application note requires that the reader has a basic knowledge of any programming language such as C.

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
Select a Picaso Display Module	8
Select a Diablo16 Display Module	9
Select a Goldelox Display Module	10
<i>Select Designer</i>	10
Design the Project	11
<i>The Hello-World Program</i>	11
Hello World - Picaso	11
Hello World – Diablo16	11
Hello World – Goldelox	11
<i>Define the Platform</i>	12
Picaso	12
Diablo16	12

Goldelox	12
<i>Include Files</i>	12
Hexadecimal Colour Values	12
How to Open Include Files	12
<i>Defining Constants</i>	13
<i>The Main Function</i>	14
Set the Orientation	14
Comments in 4DGL	14
Print a String	15
The Repeat-forever Loop	15
Run the Program	15
<i>Save the Project</i>	15
<i>Connect the Display Module to the PC</i>	15
<i>Program Destination</i>	16
<i>Compile and Download</i>	16
Proprietary Information	17
Disclaimer of Warranties & Limitation of Liability	17

Application Overview

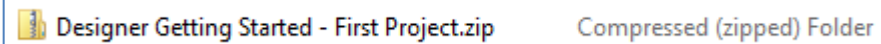
The Designer environment enables the user to write 4DGL code in its natural form to program the display module. 4DGL is a graphics oriented language allowing rapid application development, and the syntax structure was designed using elements of popular languages such as C, Basic, Pascal and others. Programmers familiar with these languages will feel right at home with 4DGL.

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 basics of 4DGL (4D Graphics Language).

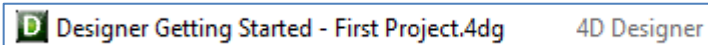
Setup Procedure

Open a Project from the File Explorer

This document comes with a demo Designer program in a zip file.



In the file explorer window, extract the content of the zip file to a desired location. The content is a Designer project file.



The user can open the file by double-clicking on it or by selecting it from Workshop 4. Users who want to learn how to create a new Designer program, proceed to the next section.

Open the Example in Workshop 4

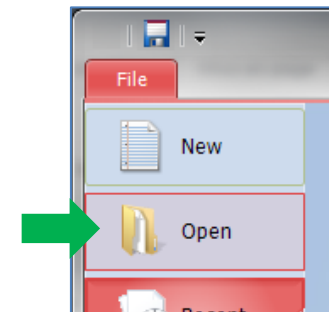
There is a shortcut for Workshop 4 on the desktop.



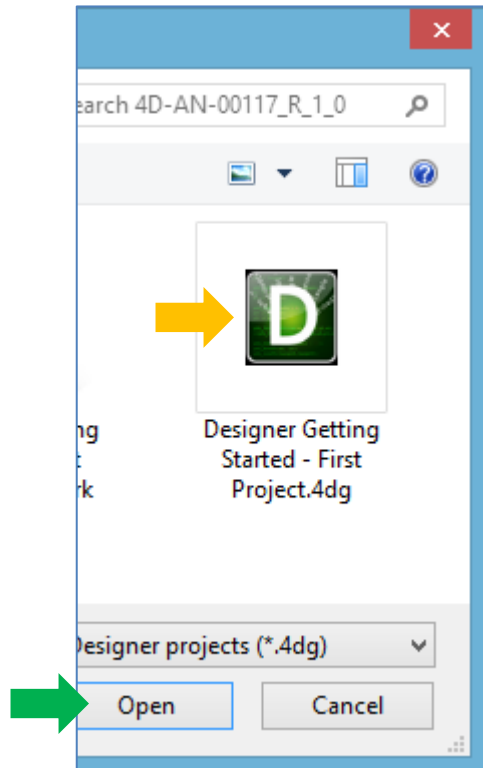
Launch Workshop 4 by double-clicking on the icon. Workshop 4 opens and displays the Recent page.



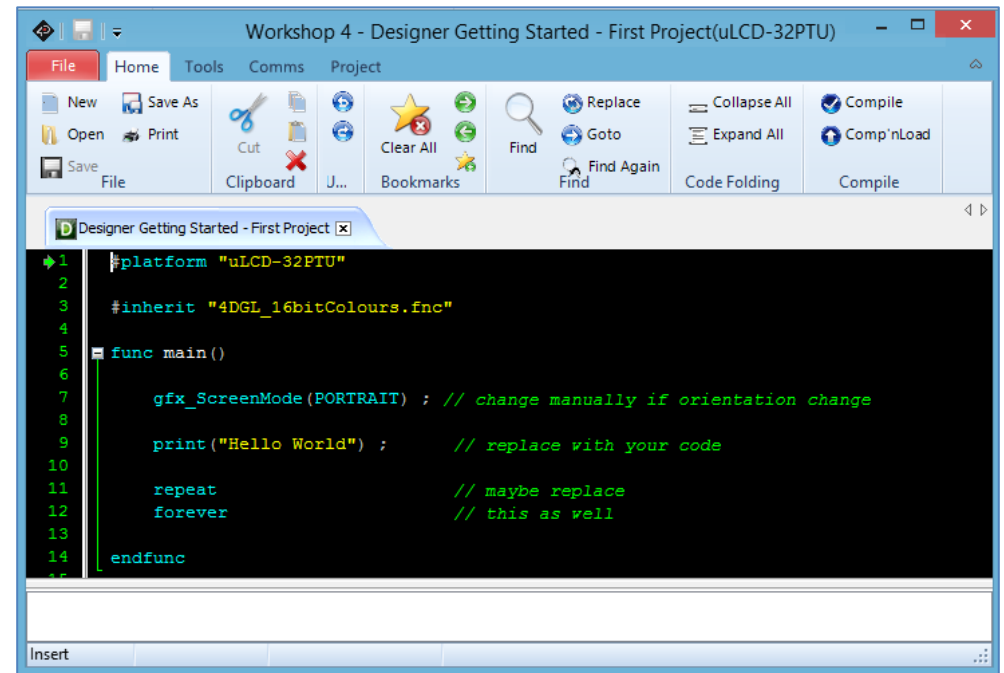
To load the existing project, click on Open.



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

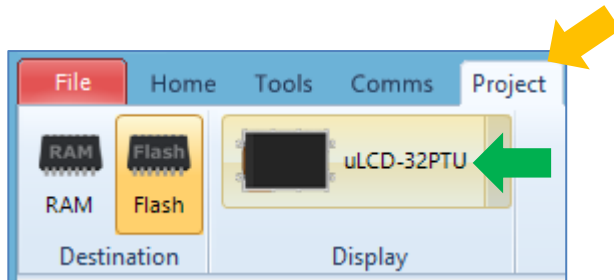


The project opens.

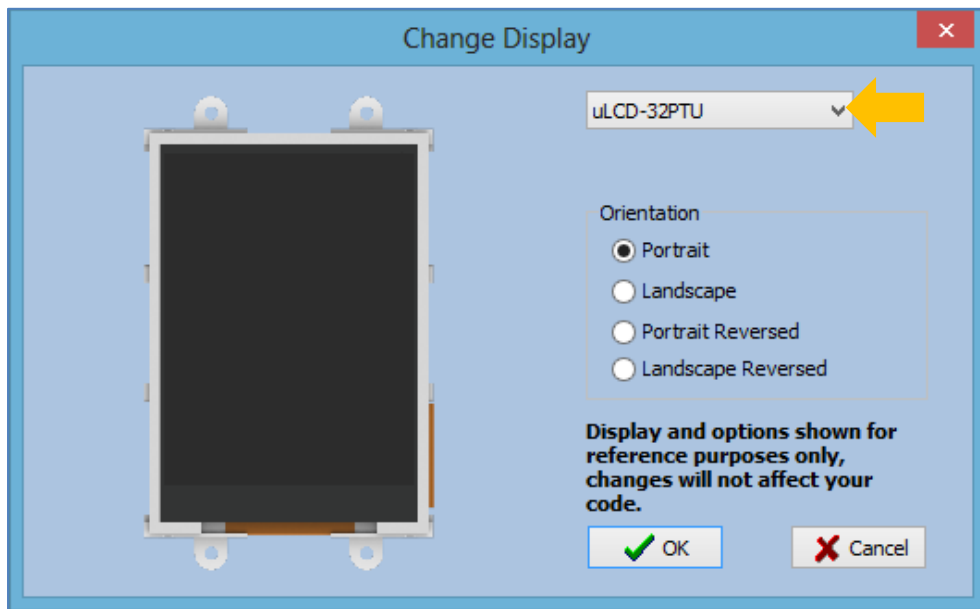


How to Change the Target Display

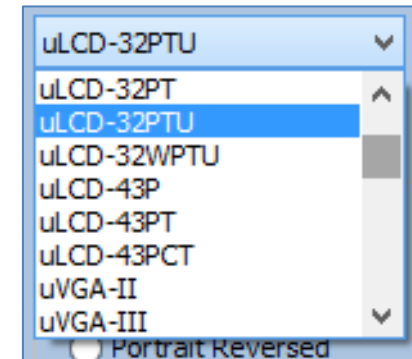
The attached project has its target display configured to be a uLCD-32PTU. To change the target device, go to the **Project** menu and click on the display button.



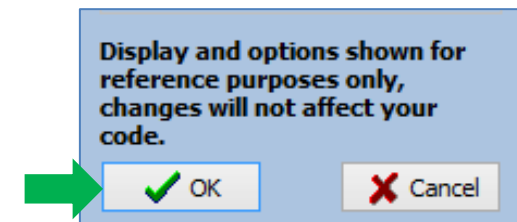
The Change Display window appears. Change the target display by clicking on the display name.



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.



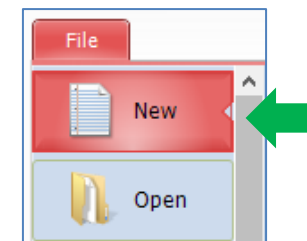
Launch Workshop 4 by double-clicking on the icon. Workshop 4 opens and displays the Recent page.

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.



Select a Picaso Display Module

The Choose-Your-Product window appears. Select the appropriate screen and preferred orientation. The screen used in this example is a **uLCD-32PTU (Portrait orientation)**.



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

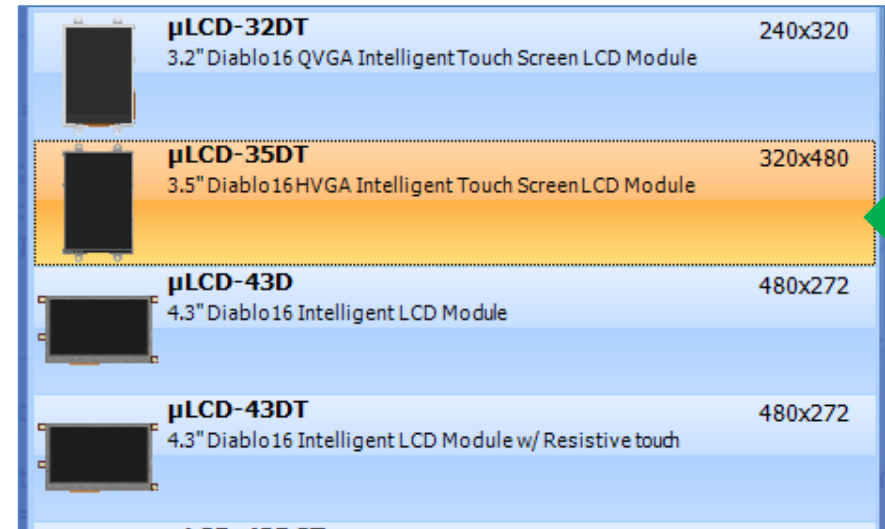


The image of the display rotates as you click it. When done, click on the next button on the lower right part of the Choose-Your-Product window.



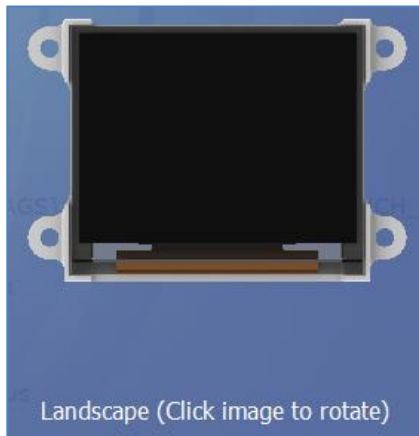
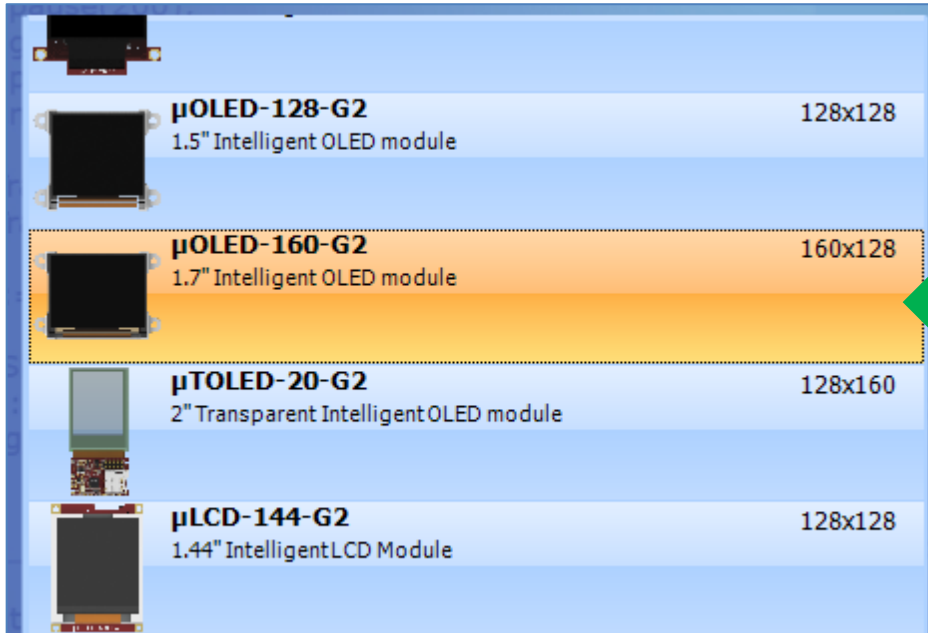
Select a Diablo16 Display Module

To select a Diablo16 display module such as the uLCD-35DT in portrait orientation,

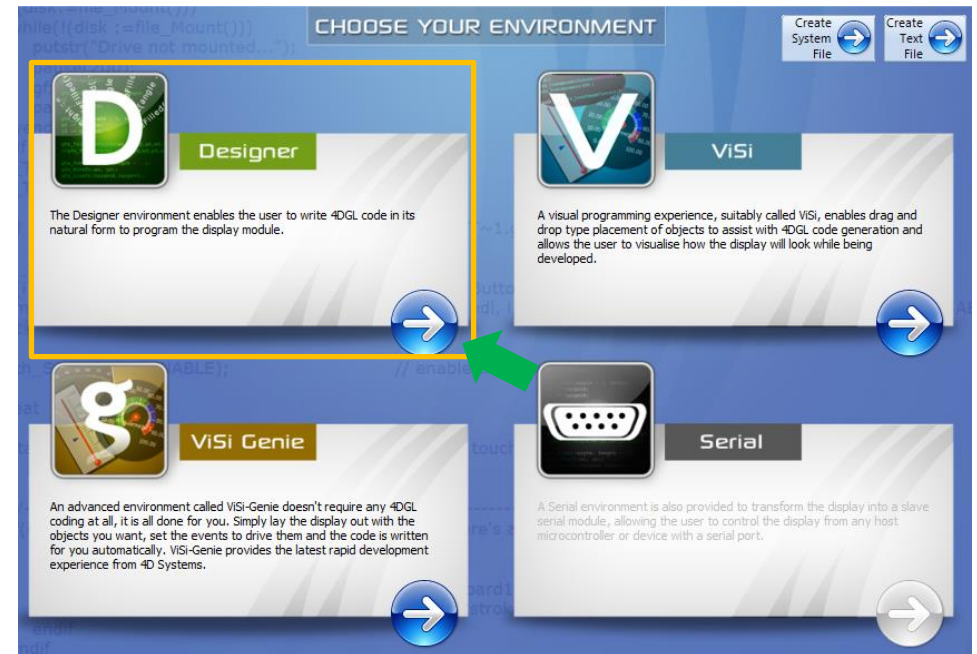


Select a Goldelox Display Module

To select a Goldelox display module such as the uLCD-160-G2 in landscape orientation,



Select Designer



Design the Project

The Hello-World Program

Everything is now ready to start designing a project. Workshop automatically provides the program skeleton, which contains basic parts needed to start designing an application. This is the “hello-world” program.

Hello World - Picaso

```

1  #platform "uLCD-32PTU"
2
3  #inherit "4DGL_16bitColours.fnc"
4
5  func main()
6
7      gfx_ScreenMode(PORTRAIT) ; // change manual
8
9      print("Hello World") ;      // replace with
10
11      repeat                    // maybe replace
12      forever                  // this as well
13
14  endfunc

```

Hello World - Diablo16

```

1  #platform "uLCD-35DT"
2
3  #inherit "4DGL_16bitColours.fnc"
4
5  func main()
6
7      gfx_ScreenMode(PORTRAIT) ; // change manual
8
9      print("Hello World") ;      // replace with
10
11      repeat                    // maybe replace
12      forever                  // this as well
13
14  endfunc

```

Hello World – Goldelox

```

1  #platform "GOLDELOX"
2
3  #inherit "4DGL_16bitColours.fnc"
4
5  func main()
6
7      gfx_ScreenMode(LANDSCAPE) ; // change manual
8
9      print("Hello World") ;      // replace with
10
11      repeat                    // maybe replace
12      forever                  // this as well
13
14  endfunc

```

Define the Platform

The first line of the hello world program defines the specific target display module.

Picasso

```
1 #platform "uLCD-32PTU"
```

Diablo16

```
1 #platform "uLCD-35DT"
```

Goldelox

For Goldelox display modules, the first line 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 hello world programs for the Picasso, Diablo16, and Goldelox platforms have one common include file.

```
3 #inherit "4DGL_16bitColours.fnc"
```


Hexadecimal Colour Values

4DGL_16bitColours.fnc contains hexadecimal values for different colours. The user can view the contents of **4DGL_16bitColours.fnc** by following the instructions below.

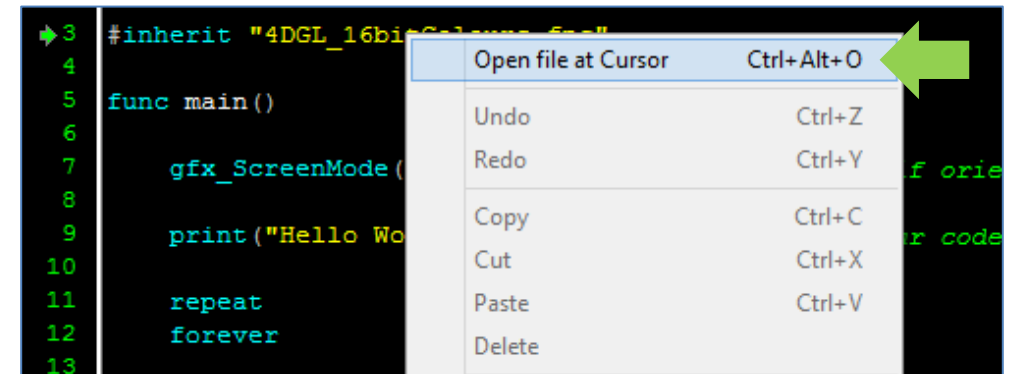
How to Open Include Files

1. Put the cursor on the filename.

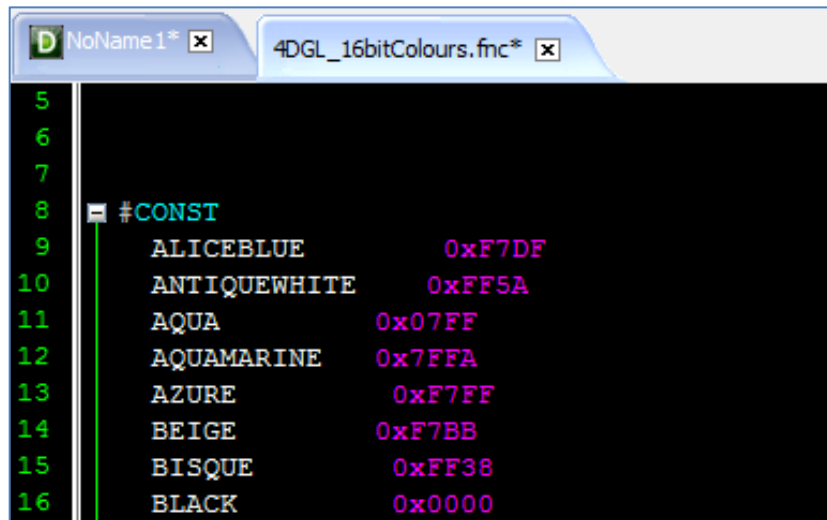
```
1 #platform "uLCD-32PTU"
2
3 #inherit "4DGL_16bitColours.fnc"
4
```



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



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

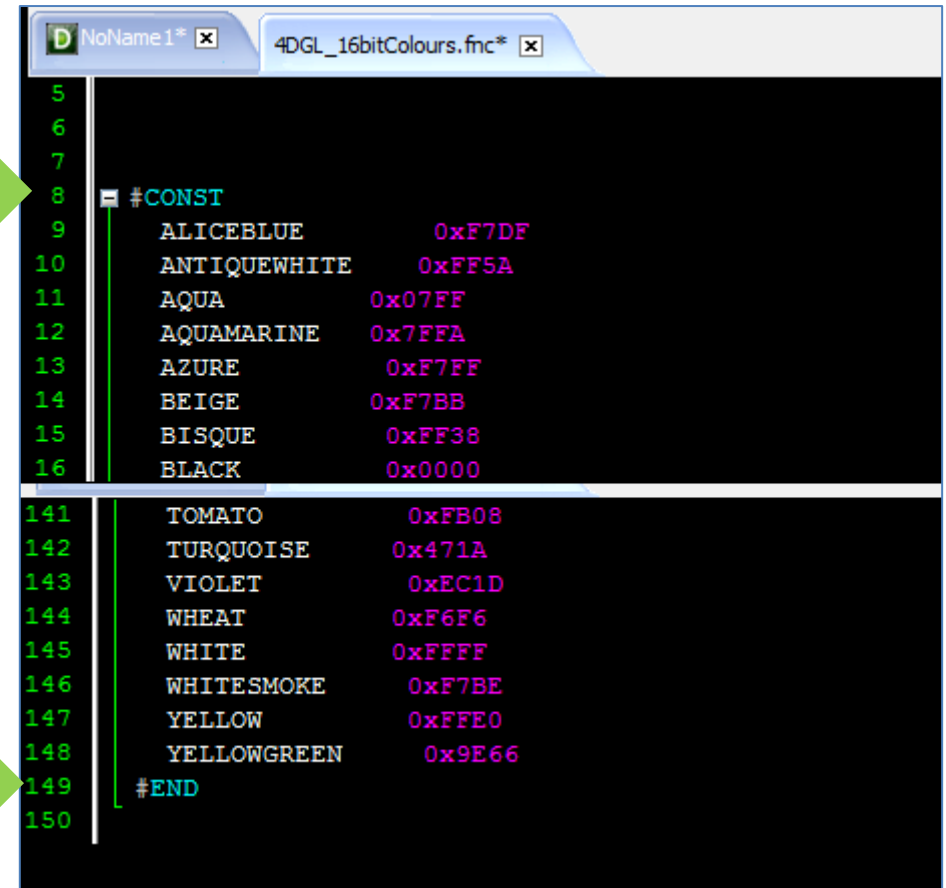


```
5
6
7
8 #CONST
9 ALICEBLUE      0xF7DF
10 ANTIQUEWHITE   0xFF5A
11 AQUA           0x07FF
12 AQUAMARINE     0x7FFA
13 AZURE          0xF7FF
14 BEIGE          0xF7BB
15 BISQUE         0xFF38
16 BLACK          0x0000
```

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

Defining Constants

The file **4DGL_16bitColours.fnc** contains hexadecimal values of 140 commonly used colours. Scroll down to see all the colours defined. Take note of lines 8 and 149 indicated below.



```
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
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
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 figure above shows how a block of constants is declared.

The Main Function

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

main {

```

1  #platform "uLCD-32PTU"
2
3  #inherit "4DGL_16bitColours.fnc"
4
5  func main()
6
7      gfx_ScreenMode(PORTRAIT) ; // change manually if orientation
8
9      print("Hello World") ;      // replace with your own code
10
11      repeat                      // maybe replace with a loop
12      forever                    // this as well
13
14  endfunc
  
```

Note that the block starts with

```
5 func main()
```

and ends with

```
14 endfunc
```

This is how a function is defined in 4DGL.

Set the Orientation

Line 7 sets the orientation of the screen.

```
7 gfx_ScreenMode(PORTRAIT) ; // change manually if orientation
```

This line is automatically generated by Workshop when the user selects an orientation in the Choose-Your-Product window while creating a new project. Possible modes for orientations are **LANDSCAPE**, **LANDSCAPE_R**, **PORTRAIT**, and **PORTRAIT_R**. These correspond to landscape, landscape reversed, portrait, and portrait reversed, respectively.

```

gfx_ScreenMode(PORTRAIT) ;      // portrait orientation
gfx_ScreenMode(PORTRAIT_R) ;   // reversed portrait orientation
gfx_ScreenMode(LANDSCAPE) ;    // landscape orientation
gfx_ScreenMode(LANDSCAPE_R) ;  // reversed landscape orientation
  
```

To change the display orientation of an already-created project, the user has to manually edit the code. The Change Display window under the Project menu will have no effect on the already-generated code for setting the orientation.

Comments in 4DGL

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

```
gfx_ScreenMode(PORTRAIT) ; // change manually if orientation change
```

A single-line comment

Print a String

The line

```
9 print("Hello World") ; // replace with your code
```

will print the text “Hello World” on the screen.

The Repeat-forever Loop

Writing the lines

```
11 repeat // maybe replace
```

and

```
12 forever // this as well
```

is one way of declaring a loop. A **loop** in a program performs instructions repetitively. The program will execute any statement between lines 11 and 12 indefinitely. Here the program actually does nothing indefinitely after printing the text “Hello World” since no instructions exist between lines 11 and 12. If this empty loop is omitted, the program exits the main function. All of this happens so fast that the user will not even see the text “Hello World” printed on the screen.

Run the Program

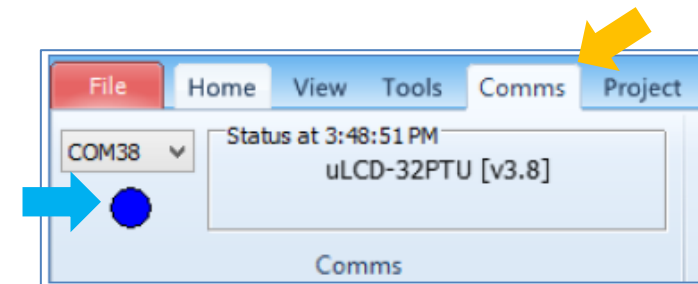
Save the Project

Save the program with the desired file name first.



Connect the Display Module to the PC

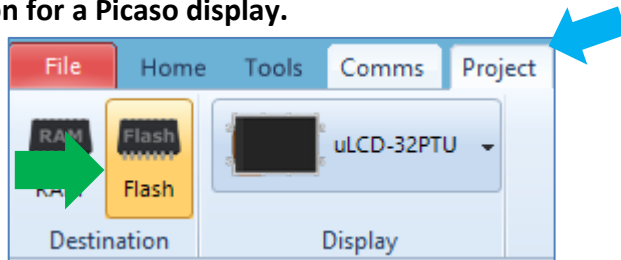
Connect the display module to the PC using a [4D Programming Cable](#) or a [uUSB-PA5](#) adapter. Go to the Comms menu to check if the module is detected. The button should be blue in colour. Below is an example of how the Comms tab will look like if a uLCD-32PTU is connected to the PC.



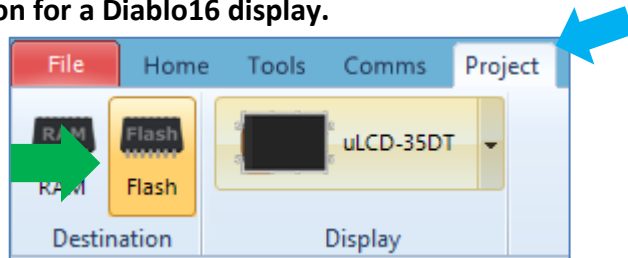
Program Destination

For Picaso and Diablo16 display modules, the user has the option of downloading the program to any of the two available memory locations – RAM and Flash. RAM is volatile and the program is lost after a power cycle. A program downloaded to Flash remains after a power cycle. Go to the Project menu and select Flash as the destination.

Flash destination for a Picaso display.



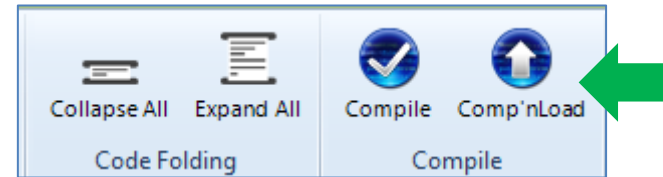
Flash destination for a Diablo16 display.



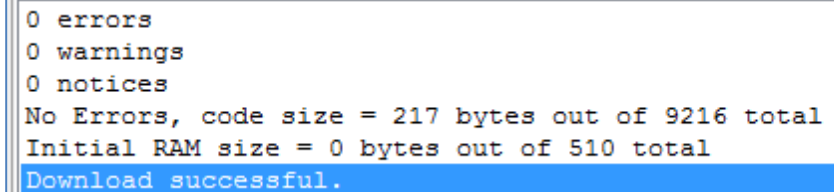
The Goldelox processor has a limited RAM size such that the program can be loaded to Flash only.

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 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 displays “Hello World” on the screen. Note that a μ SD card is not needed here since the program does not display images from the μ SD card yet.

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.