4D Systems

Quick Start Guide

PICASO-GFX2 Based Display Modules

Document Date: 30th November 2011

Document Revision: 1.0

Table of Contents

1. Introduction

This Quick Start Guide is an introduction to becoming familiar with PICASO driven Display Modules and the development tools associated with them. This guide should be treated only as a useful starting point and not as a comprehensive reference document. The primary aim of this guide is to quickly and effectively teach the essentials to setting up and developing an application on any one of the 4D PICASO-GFX2 display modules. Once the basics are mastered, developing more advanced and involved applications will flow much easier. **Section 7** lists a full range of detailed reference documents, which will be required during various stages of development.

The PICASO-GFX2 belongs to a family of processors powered by a highly optimised soft core virtual engine; **EVE** (Extensible Virtual Engine). EVE is a proprietary, high performance virtual processor, with an extensive byte-code instruction set optimised to execute compiled 4DGL programs. **4DGL** (4D Graphics Language) was specifically developed from ground up for the EVE engine core. It is a high level language, which is easy to learn and simple to understand, yet powerful enough to deliver many embedded graphics applications.

4DGL allows the developer to write applications in a high level syntax similar to popular languages such as BASIC, C and Pascal and run it directly on the PICASO-GFX2 processor, which is embedded in the; µLCD-24PT, µLCD-28PT, µLCD-32PT, µLCD43 and µVGA-II. It allows the user to take complete control of all available resources on that hardware platform; such as the Serial Ports, Graphics LCD Display, micro-SD memory card, I/O pins, etc. This eliminates the need for an external host microcontroller/processor to drive the module via serial commands. It provides the user with complete control over the hardware module, allowing them to quickly develop powerful applications.

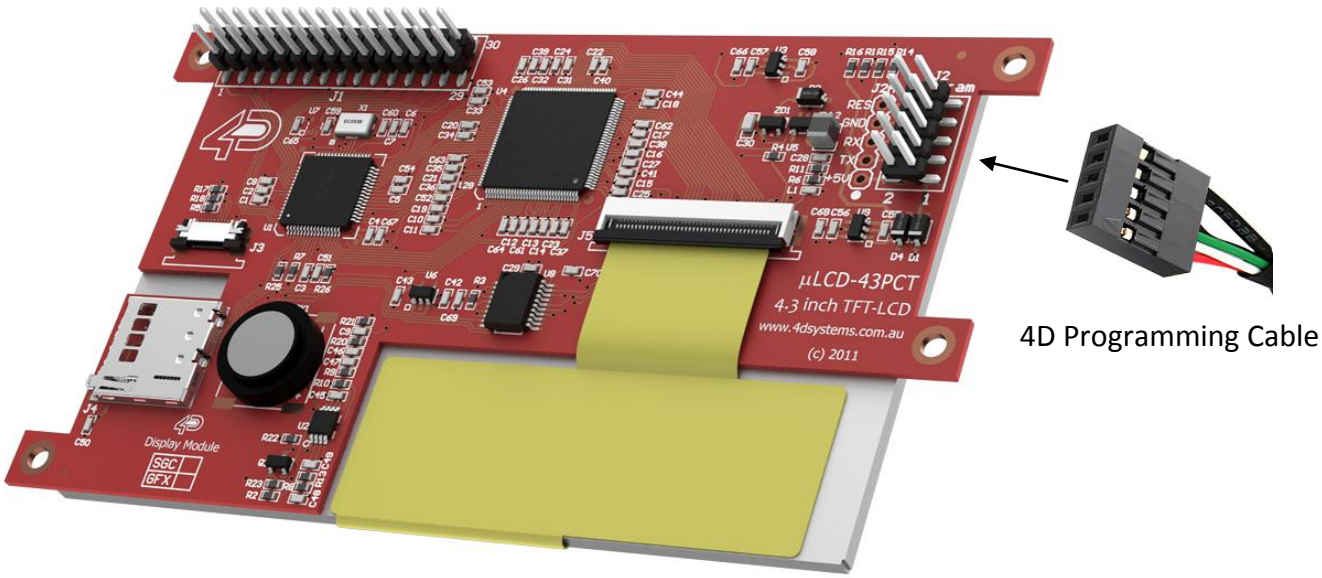
The 4D Graphics language provides power and flexibility that captures 'the art of programming' with a leisurely learning curve. Developing application programs is done using the 4D Workshop3 IDE, which incorporates a text editor, graphics tools, compiler and downloader in a user friendly environment.

2. Generic Features

Feature	µLCD-24PT	µLCD-28PT	µLCD-32PT	µLCD43	µVGA-II
2.4'' LCD TFT display	✓				
2.8'' LCD TFT display		✓			
3.2'' LCD TFT display			✓		
4.3'' LCD TFT display				✓	
RGB 65K true to life colours	✓	✓	✓	✓	✓
240 x RGB x 320 Resolution	✓	✓	✓		✓
480 x RGB x 272 Resolution				✓	
640 x RGB x 480 Resolution					✓
800 x RGB x 480 Resolution					✓
Custom Resolution X*Y = 405K (414720)					✓
Integrated 4-Wire Resistive Touch Panel	✓	✓	✓	✓	
Optional Non-Touch or Capacitive Touch Panel				✓	
5 pin interface to any host device: VCC, TX, RX, GND, and RESET	✓	✓	✓	✓	✓
PICASO-GFX2 processor	✓	✓	✓	✓	✓
14KB of Flash memory for user code	✓	✓	✓	✓	✓
14KB of SRAM for user variables	✓	✓	✓	✓	✓
Two Asynchronous hardware serial ports (COM0, COM1), TTL interface, with 300 baud to 256K baud	✓	✓	✓	✓	✓
One I2C interface (Master)	✓	✓	✓	✓	✓
Eight 16 bit timers with 1.0ms resolution	✓	✓	✓	✓	✓
13 General Purpose I/O pins. Upper 8 bits can be used as an I/O Bus for fast 8-bit parallel data transfers	✓	✓	✓	✓	✓
micro-SD memory card adaptor for multimedia storage and data logging purposes. HC memory card support is also available for cards larger than 4GB.	✓	✓	✓	✓	✓
DOS compatible file access (FAT16 format), as well as low level access to card memory	✓	✓	✓	✓	✓
Dedicated PWM Audio pin supports FAT16 audio WAV files and complex sound generation	✓	✓	✓	✓	✓
On-board audio amplifier with a 8Ω speaker for sound generation and WAV file playback	✓	✓	✓	✓	
Built in extensive 4DGL graphics and system library functions	✓	✓	✓	✓	✓
Display full colour images, animations, icons and video clips	✓	✓	✓	✓	✓
Supports all available Windows fonts and characters	✓	✓	✓	✓	✓
Two 2 x 15 pin headers for I/O expansion	✓	✓	✓		
One 2 x 15 pin header for I/O expansion				✓	
One 2 x 11 pin header for I/O expansion					✓
15 pin D-type standard VGA connector to interface to any external VGA monitor					✓
4.0V to 5.5V range operation (single supply).	✓	✓	✓	✓	✓
Four mounting tabs with 3mm holes for mechanical support			✓	✓	
Three mounting tabs with 3mm holes for mechanical support		✓			
RoHS Compliant	✓	✓	✓	✓	✓

3. Powering your Device

Powering a PICASO-GFX2 display module is as simple as connecting either a 4D Programming Cable, µUSB-MB5 or µUSB-CE5 to a PC. The latter two will require appropriate USB cables, thus the 4D Programming Cable is recommended for ease of use and direct connection.



4D Programming Cable

µUSB-CE5

µUSB-MB5

4D Programming Cable

Note: For further information on powering the device and the specific electrical ratings, refer to the following relevant document:

- uLCD-24PT-GFX-DS-revx.pdf
- uLCD-28PT-GFX-DS-revx.pdf
- uLCD-32PT-GFX-DS-revx.pdf
- uLCD43-GFX-DS-revx.pdf

4. Downloading an Application Program

Connecting the display module to the PC is one of the most important elements of getting started. While there are several methods that work, the recommended and easiest method is to use the 4D Programming cable. Connecting to the PC is important, because it allows you to update the PmmC file, which will be described in the next section. Additionally, it provides the means for deploying your 4DGL application code to the module.

All three interfacing methods have a five pin female header, which will connect to the five pin male programming header found on each module. Relevant drivers will need to be downloaded for the CP2102 (USB-to-UART) chip that is used in the µUSB-MB5 and the 4D Programming cable. The latest drivers can be downloaded from here:

- http://www.4dsystems.com.au/prod.php?id=18 (µUSB-MB5/4D Cable)
- http://www.4dsystems.com.au/prod.php?id=19 (µUSB-CE5)

5. Useful Software Tools for PICASO Display Modules

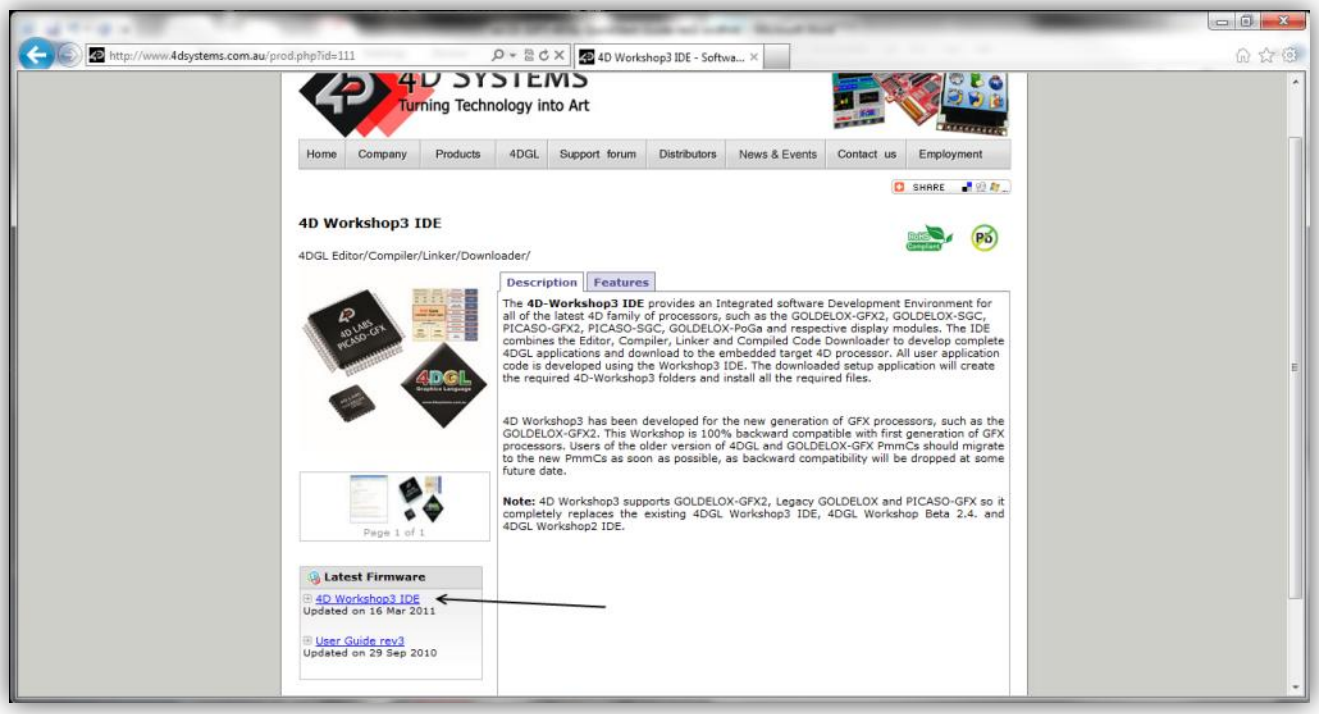
This section describes the software tools applicable for use with PICASO-GFX2 display modules and begins with a walkthrough of the installation process. Firstly, is the 4D Workshop3 IDE, which includes; Workshop, Graphics Composer, and PmmC Loader. All of the previously mentioned tools are contained within the one installer that can be downloaded from the link below.

- http://www.4dsystems.com.au/prod.php?id=111 (4D Workshop3 IDE)

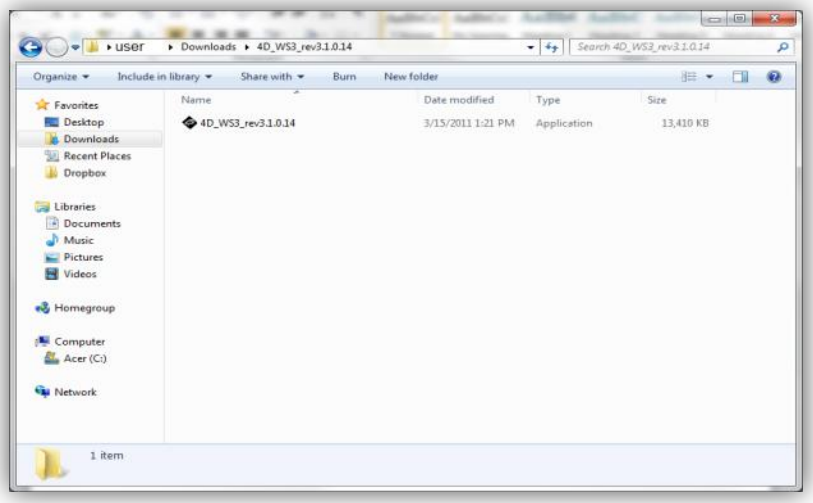
Installing 4D Workshop3 IDE

To begin, download the latest installer, which can be found at the following link:

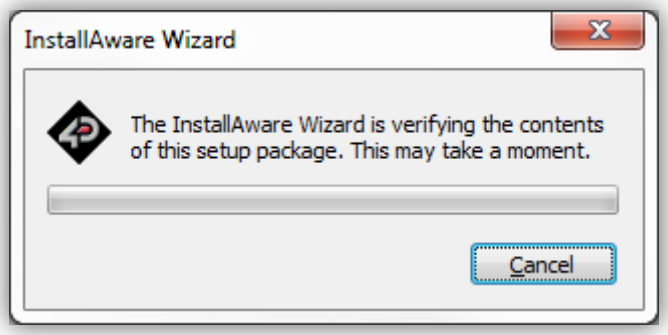
- http://www.4dsystems.com.au/prod.php?id=111



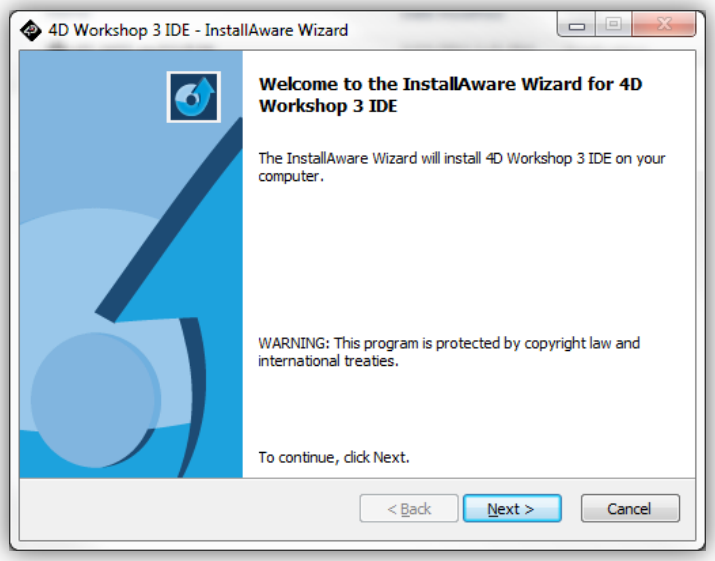
The download will contain a compressed .zip file. Extract the contents onto the PC and run the file as shown in the image below.



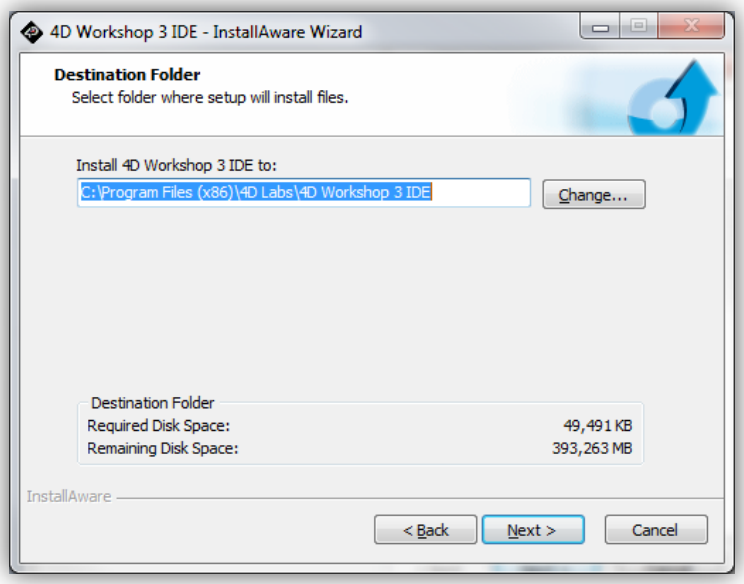
The following prompt will appear when the Installer begins.



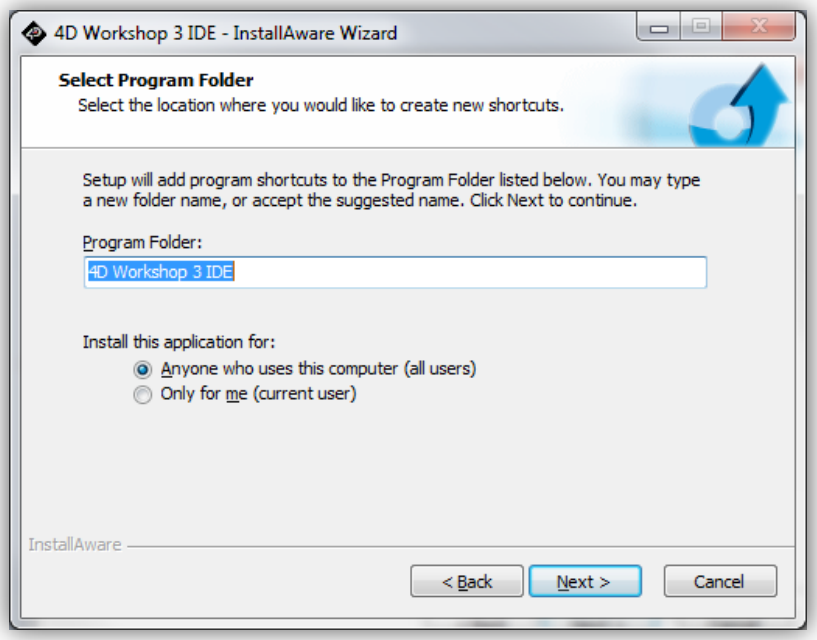
After the Installer finishes extracting the files, the Welcome screen is displayed. Click **Next** to begin.



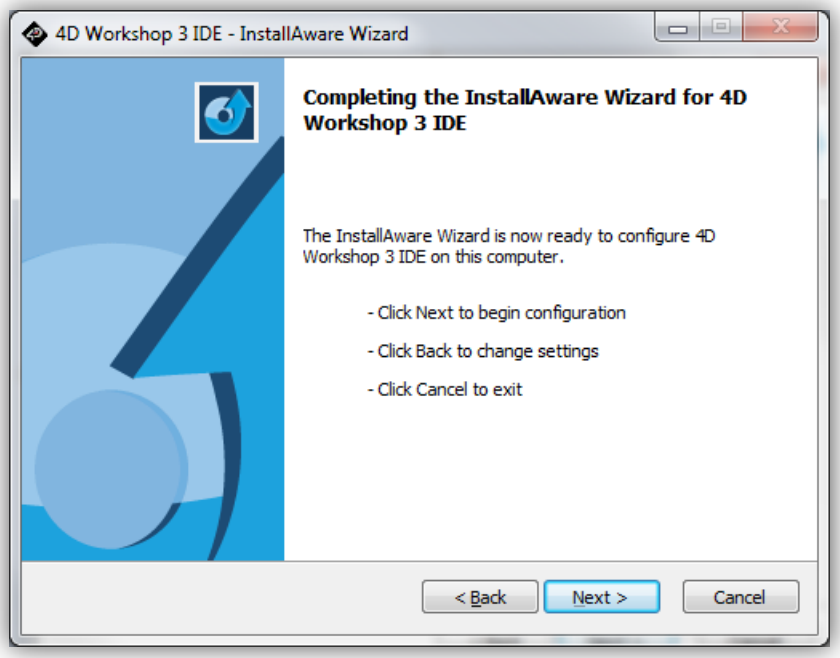
The installer will now ask where to install the Tools. Choose a convenient location and click **Next** to continue.



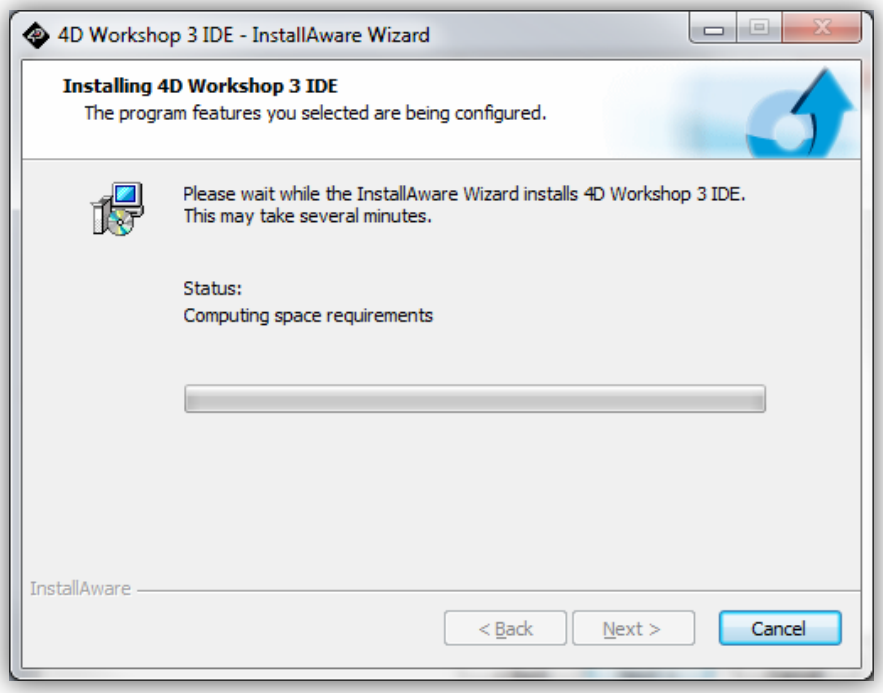
The Installer will ask who can access the application; the default choice is anyone who uses the computer. Select the desired option and click **Next** to continue.



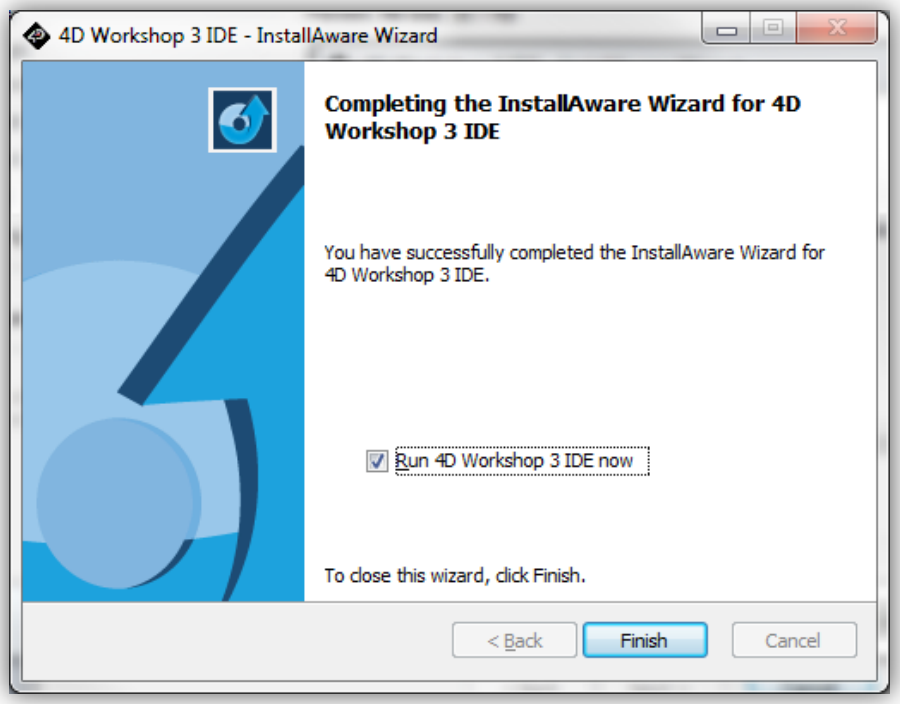
The installer is now ready to perform the install. Click **Next** to continue.



This installer will now display the progress of the installation.



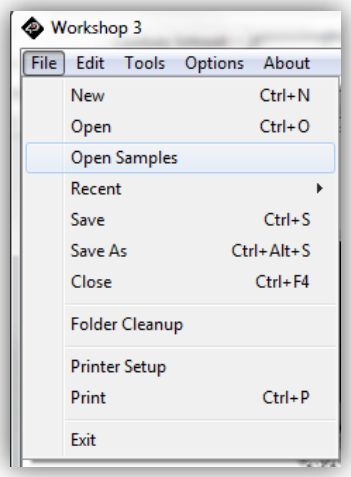
Once the installer is finished, the following prompt will appear. Click **Finish** to close the installer and leave the checkbox checked if you wish to run 4D Workshop3 IDE now.



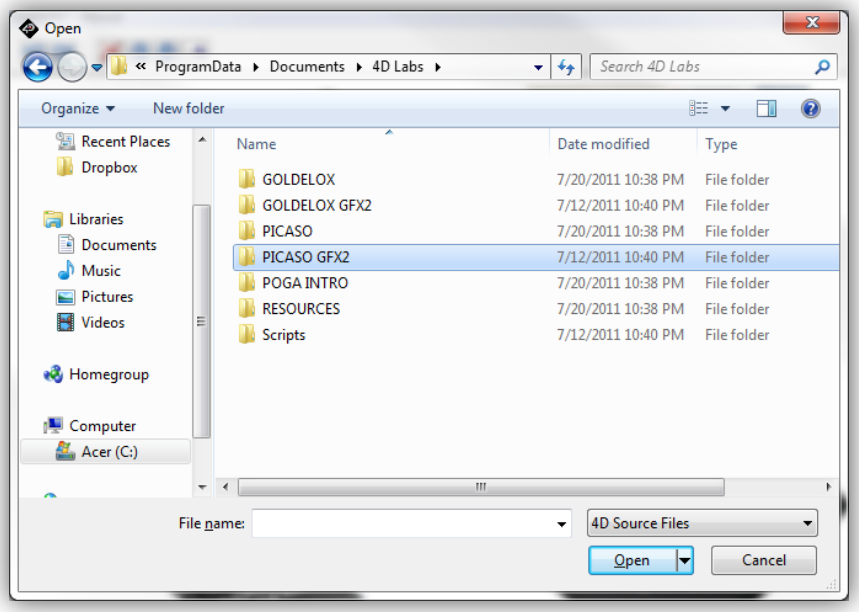
Workshop3

When Workshop3 runs, the following screen will appear.

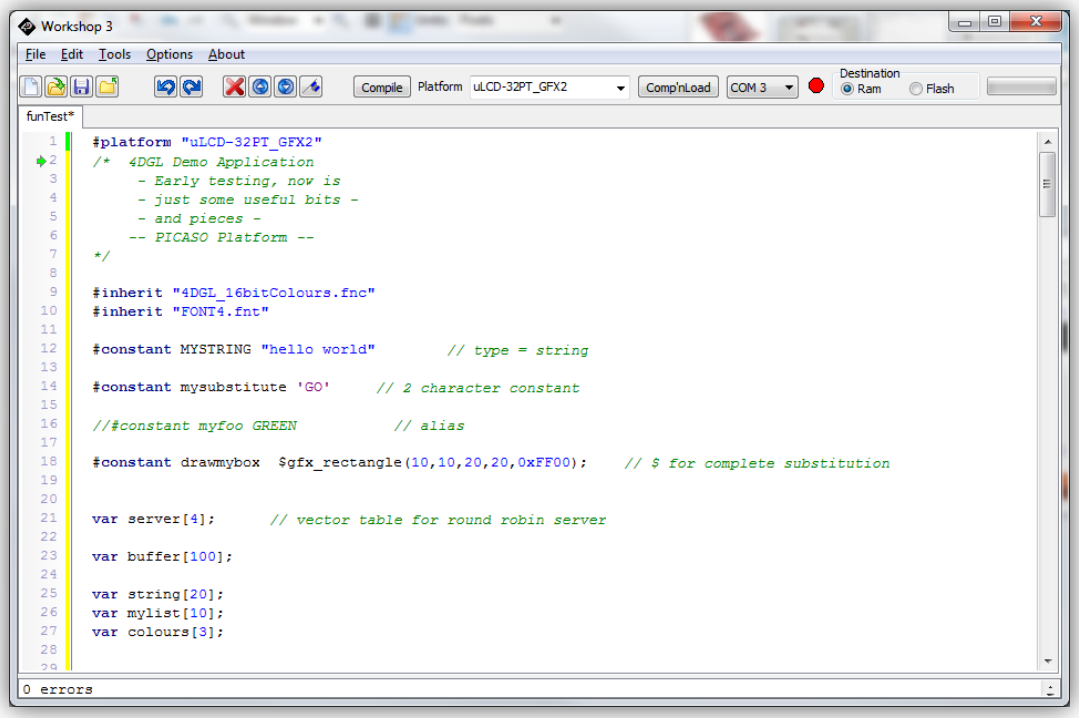
To get started, it is recommended that the samples included with the Workshop3 are examined. These provide a quick insight into how the 4DGL language works and are a known working application program that can be easily downloaded onto a module to illustrate various functions.



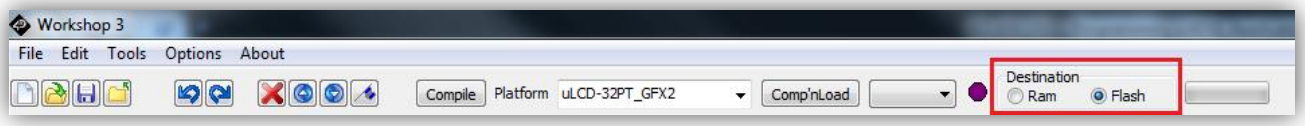
Open the PICASO-GFX2 folder as this will contain the compatible files for use.



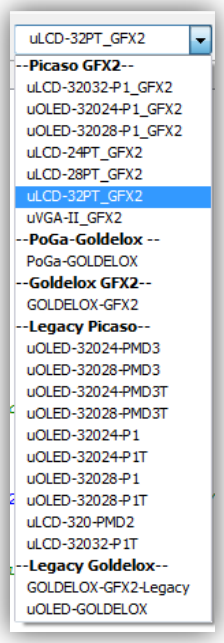
Once an example application program is opened, or a new project file is created, it will appear as the window seen below. Just like other programming languages, the syntax used is differentiated through the use of colours. Examine the code to get a feel for how it works.



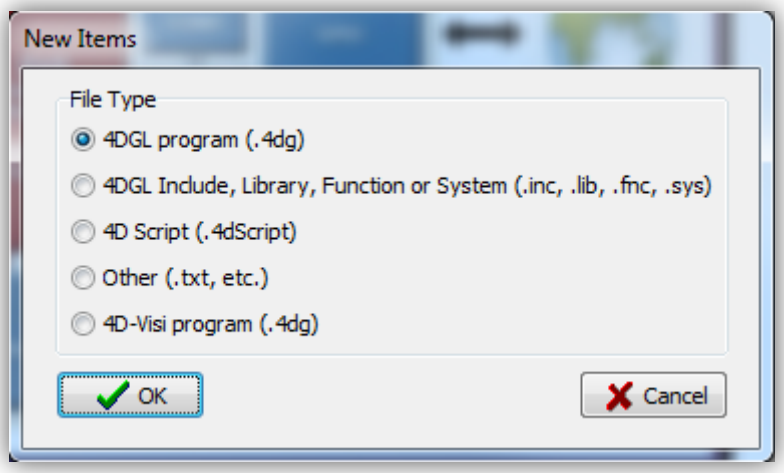
At the top of Workshop3 is a toolbar, which contains many important features for checking that the code will run. This includes; ensuring that the right platform is chosen, correct Comm Port selection and lastly, the destination. Ram is the default option and will not retain the program after a power down. Writing to Flash will hold the application in memory and will execute on every power up.



Select the correct module being used, which will appear in the first category of the dropdown box.

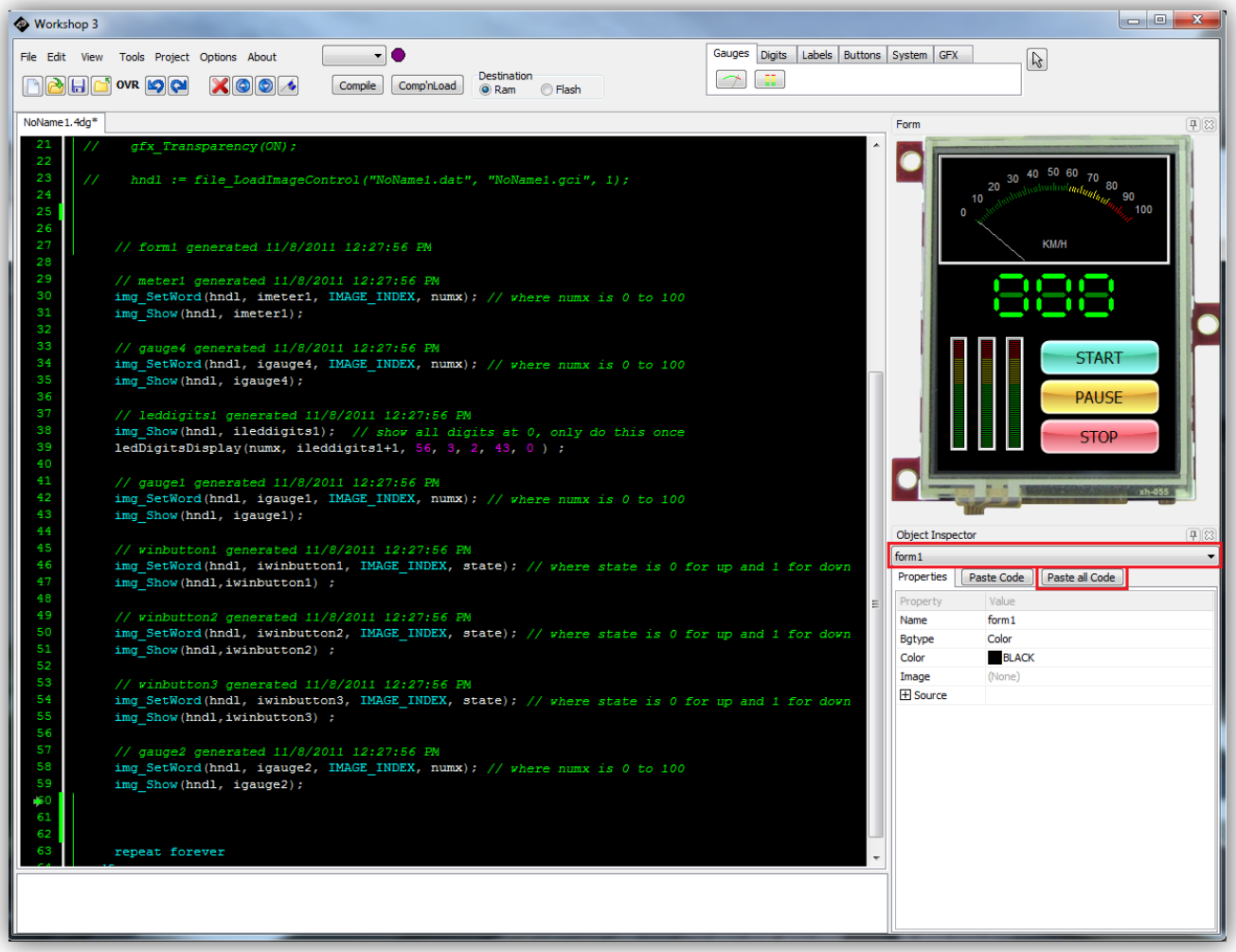


When choosing to create a new file, select the 4DGL file type when this window prompt appears. This file type will allow you to begin coding in 4DGL for use with a GFX module.



4D-ViSi

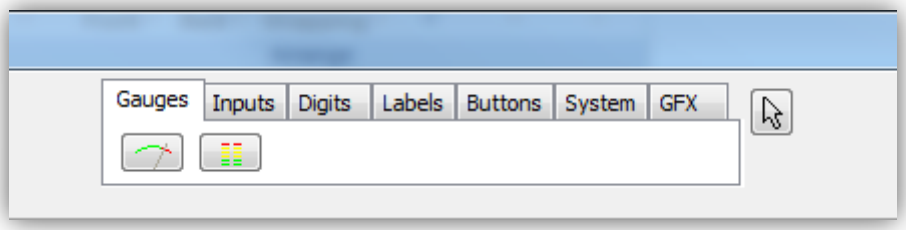
4D-Visi is the perfect software tool that is used to see instant results of a desired graphical layout for a 4D display module. It has a selection of inbuilt dials, gauges and meters that can be simply dragged and dropped onto the simulated display, allowing easy editing and positioning of graphical objects. This removes the need to download the application code to a module to see the coding effect take place. Each item has properties that can be edited and at the click of a button, all relevant code is produced in the user program.



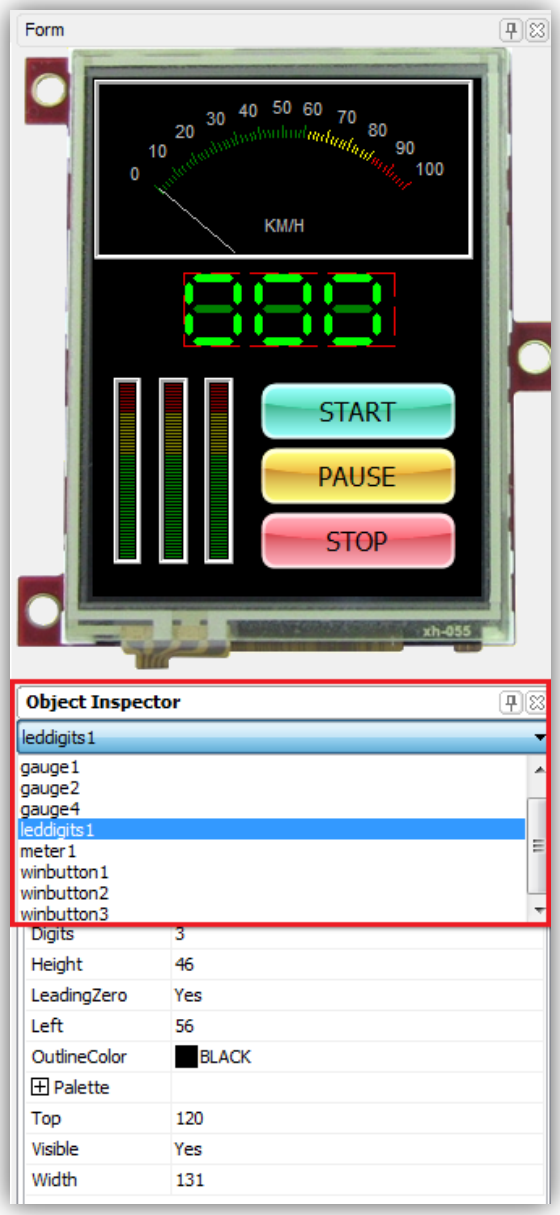
4D-ViSi reduces the amount of application development time, by reducing the ground work normally required to be carried out by the user. All that is required now from the programmer's perspective; is development to control 'how' the application will interact with input and output, not 'what' it will look like.

4D-ViSi is an update as part of the 4D Workshop3 IDE. Workshop is used in just the same way as before, only this time there are additional development tools available. Start by observing the

object toolbar across the top right hand area of the screen. Click on an object button, then click anywhere on the simulated display to place the object.

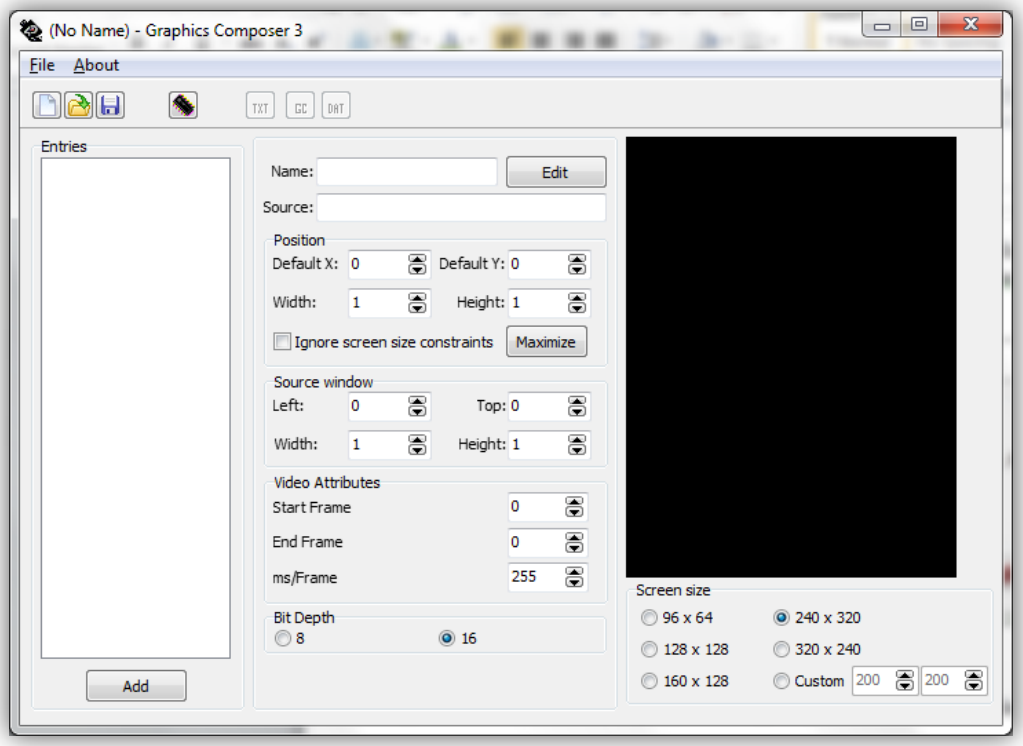


When an object is placed, it has editable fields that change different parameters associated with that object. The following illustration shows a complete built screen using 4D-ViSi. Note that once an object is complete it can be pasted into the application program by clicking **Paste Code**. Experiment with creating a screen in this way and notice how much quicker it is to develop in this way.

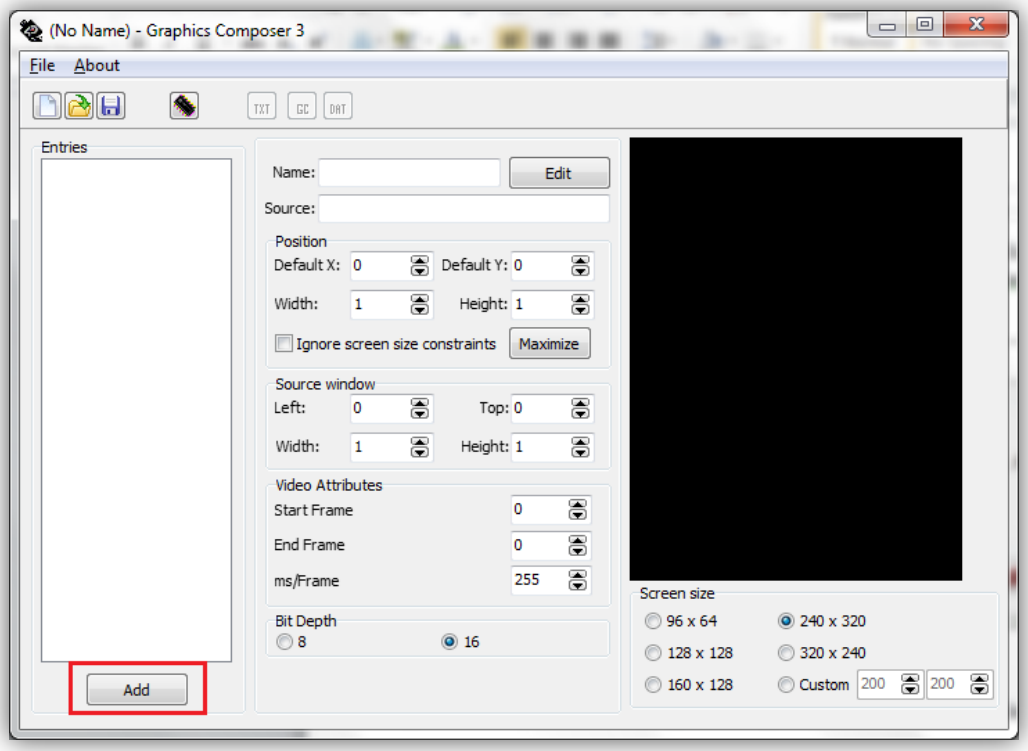


Graphics Composer

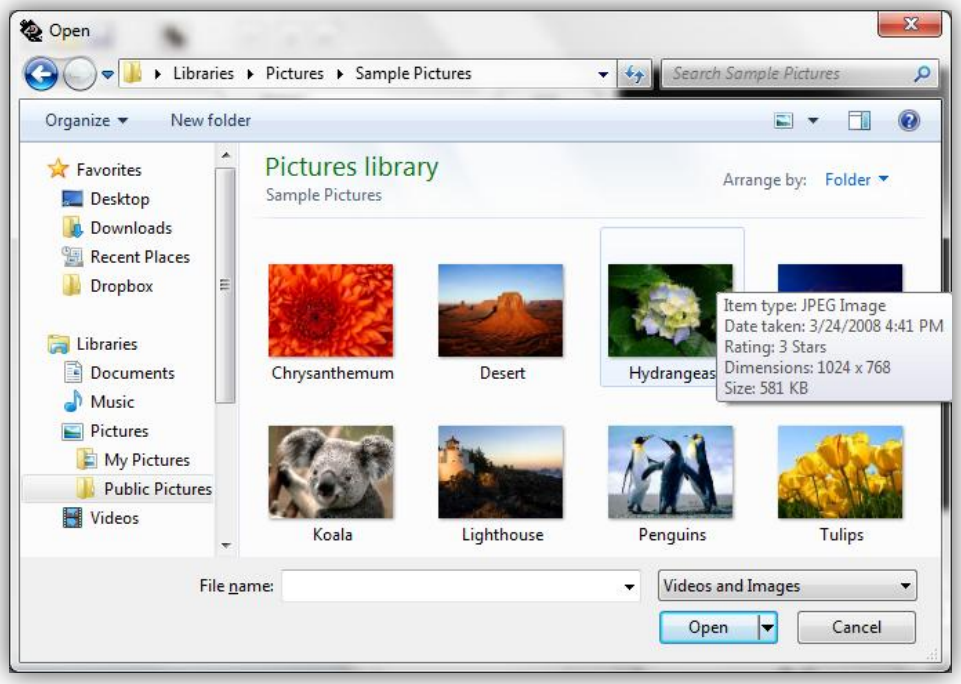
Graphics Composer software tool is used to decode images and videos for use with the GFX type display modules. A micro-SD card will be required for loading the resultant files for use with the display module.



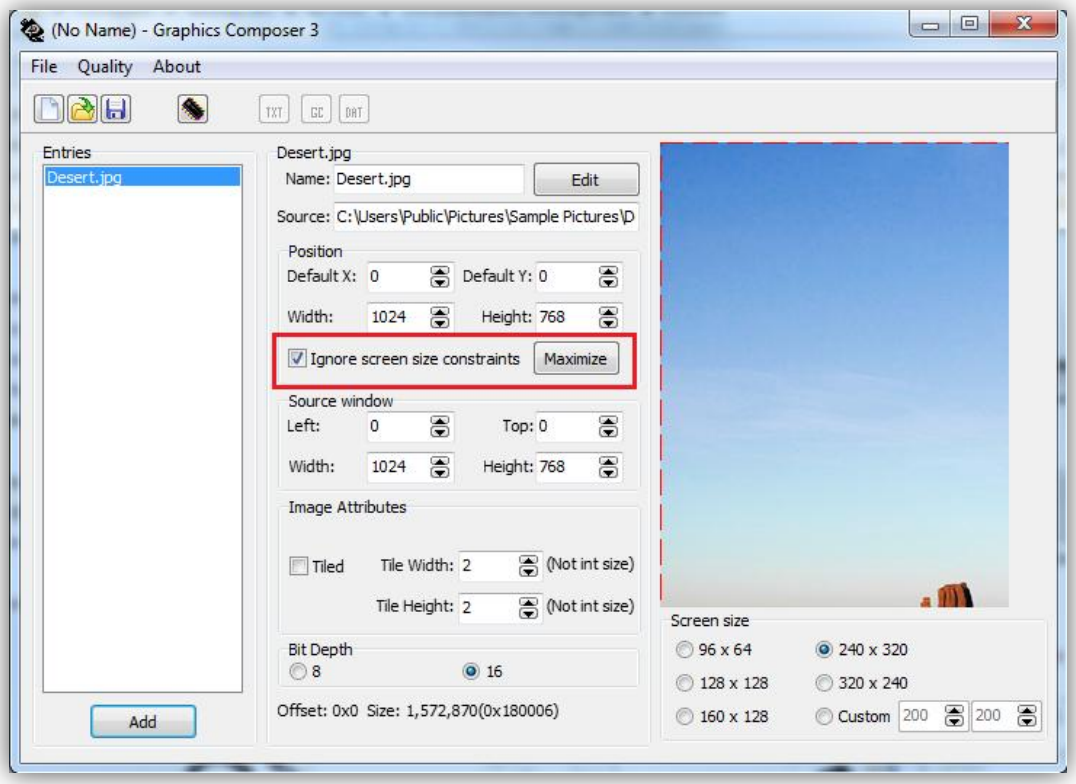
To add a new graphic element, click the **Add** button in the bottom left hand corner.



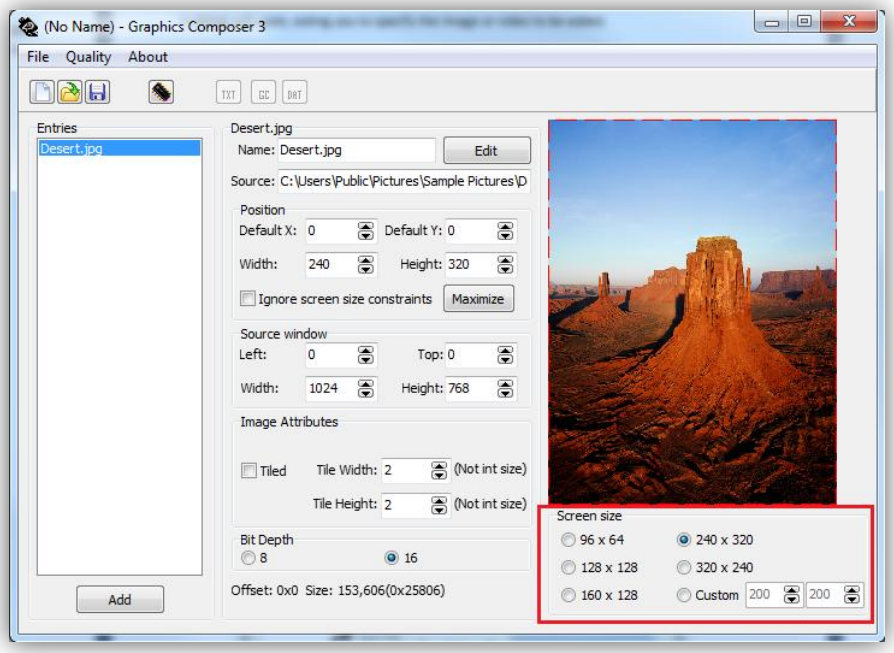
A dialog will appear asking for an image or video to be added.



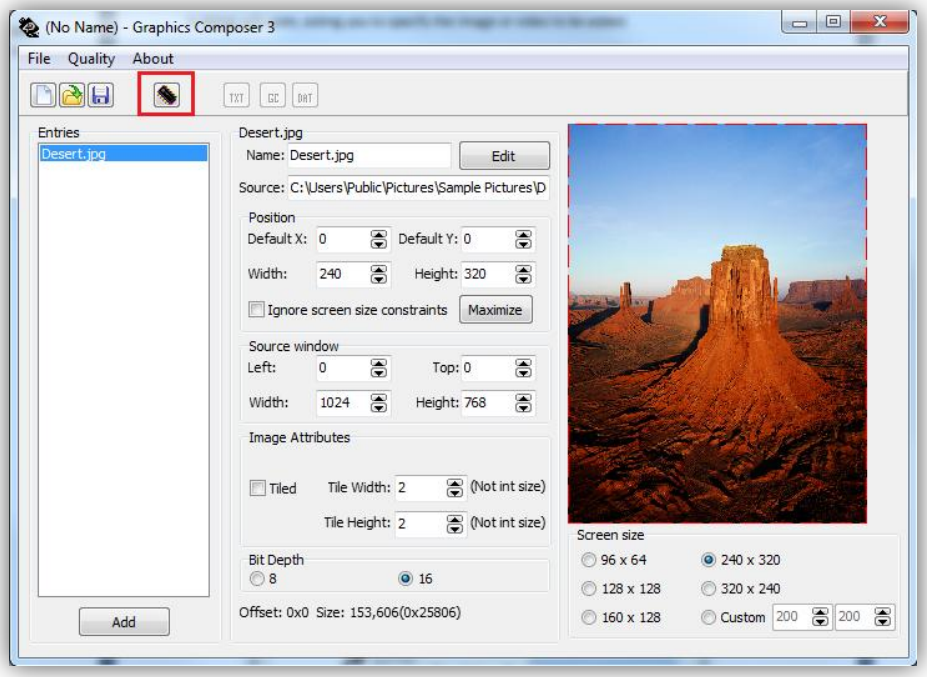
There is an option selected by default that will need to be unchecked to fit any image larger than 240x320 in the screen. Currently, this is the maximum size resolution available from any display module. An example below shows the picture not being displayed correctly because it exceeds the maximum frame size.



The window frame size can be changed to replicate the pixel width and height of the module being used. In this case the three smallest sized PICASO GFX modules have the same resolution of 240x320. If the µLCD43 or µVGA-II is being used; select the **Custom** option and enter a resolution.

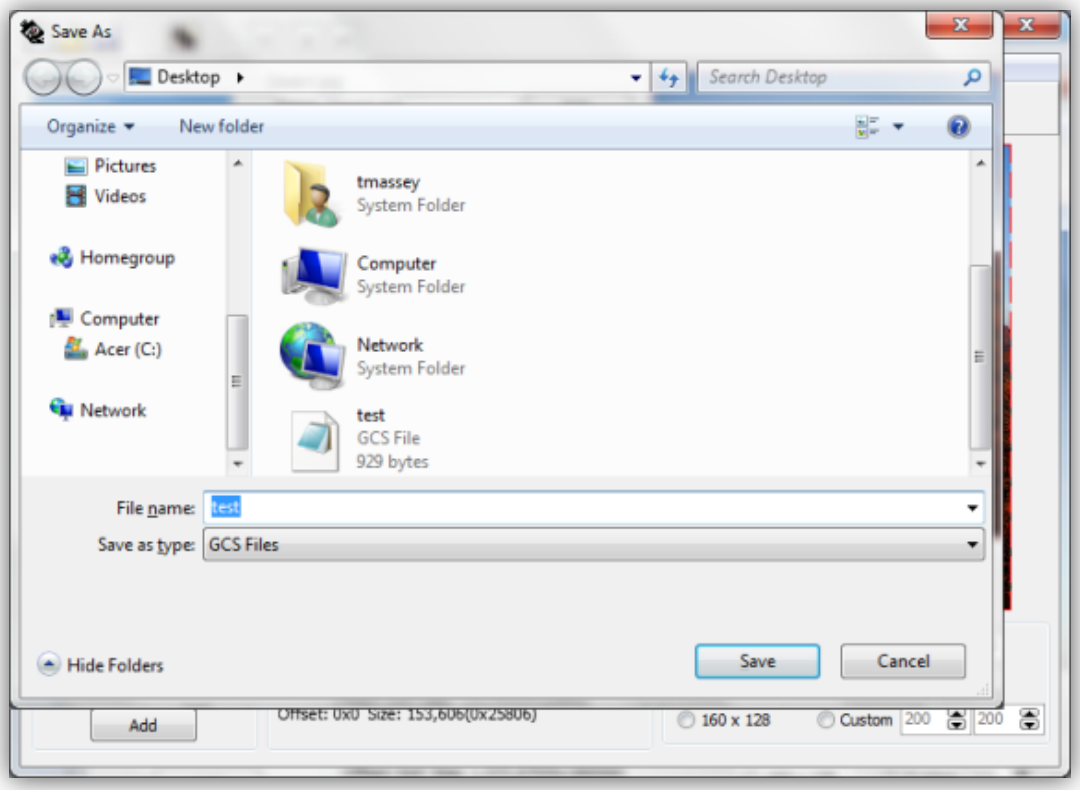


Now the conversion process can begin. Click the **chip** icon in the toolbar as shown.

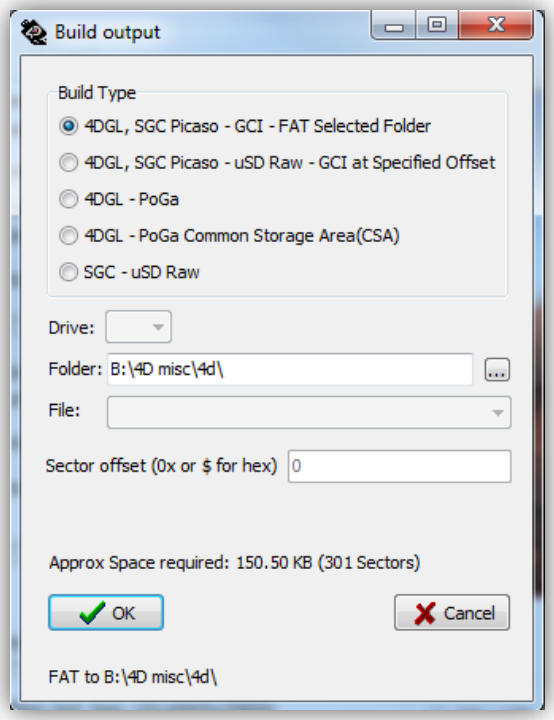


Note: The PICASO-GFX2 does not support 8-bit graphics. Please make sure the images/videos/animations you set in the Graphics Composer software tool are 16bit.

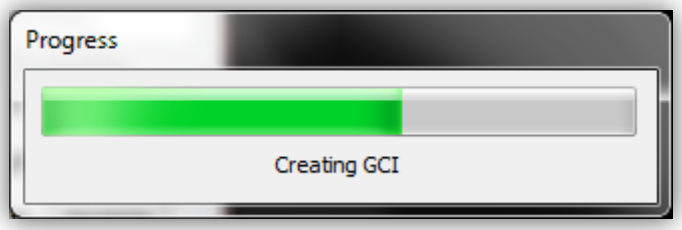
A prompt will ask for the project file to be saved. Note that this is a .gcs file which retains the project only and is not the output file of the graphics being converted.



After saving, the Build output screen will be displayed. Either of the first two options can be selected. There are two methods for displaying the image, which won't be discussed in detail here. For now, select the first option.



After clicking **OK**, insert a FAT16 formatted micro-SD card into the PC. The graphical objects will now be converted and saved to the micro-SD Card. When this screen disappears, the process is complete and you can now access images in code. A .dat and .gci file is created during the conversion. An excerpt of code below shows the required 4DGL commands to access the images stored on the micro-SD card. Additionally, a great sample is provided in the samples folder called AVIDEMO that will show how to play videos. Be aware, only the graphics are converted, no audio is saved.

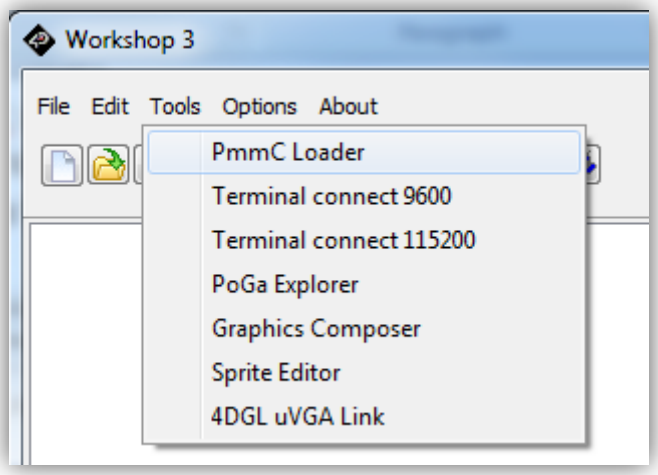



```
#platform "uOLED-32028-P1_GFX2"

/************************************************
* Filename: DisplayImage.4dg
* Created: 24th November 2011
* Updated 24th November 2011
* Author: 4D team
* Description: display an image using 4DGL
************************************************/

#inherit "4DGL_16bitColours.fnc"

var img;

func main()

    if (!file_Mount()) print("File error ",file_Error());

    img := file_LoadImageControl("test.dat", "test.gci", 0);
    img_Show(img, ALL);

    repeat
    forever

endfunc
```

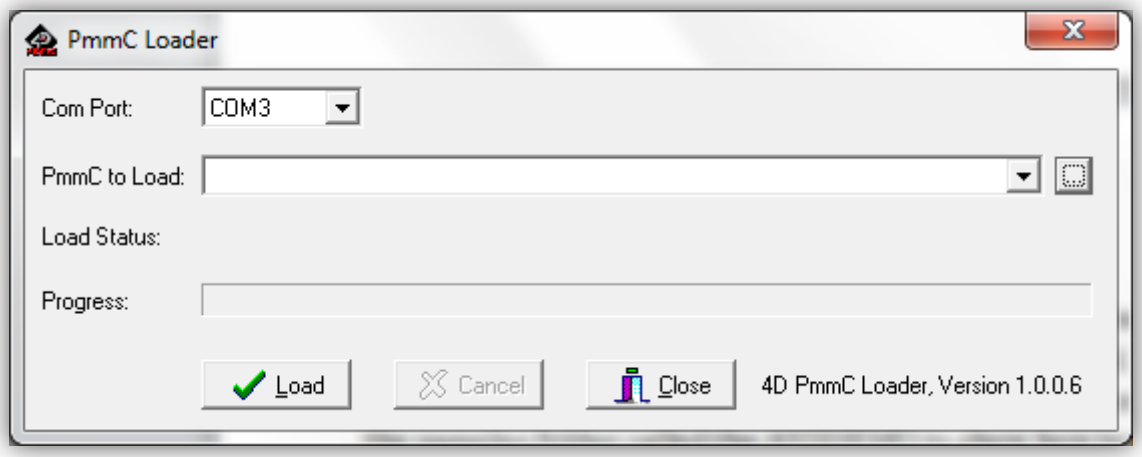
PmmC Loader

The PmmC Loader is used to update the latest firmware release known as a PmmC file. Since the PICASO-GFX2 is designed with the EVE Engine, the latest firmware version can be installed with the existing hardware. The PmmC loader comes as part of the 4D Workshop3 IDE and can be opened by clicking on **Tools** in the taskbar, then **PmmC Loader** as shown below. Alternatively, it can be downloaded separately as an individual software tool from the following product page:

http://www.4dsystems.com.au/prod.php?id=46



Downloading the latest PmmC file is as simple as connecting the device to a PC in the same way as shown in **Section 3**. Be sure to download the latest PmmC file, which can be found in the download tab on the website product page for each specific display module. Note that the PmmC file is different for every device and also determines whether a module is GFX or SGC. Browse and locate the PmmC file on the PC and click the **Load** button, which will begin updating the device.



6. What is 4DGL?

4DGL is an easy to learn programming language for use with GFX type modules. 4DGL stands for 4D Graphics Language and is based on languages such as C, Pascal and BASIC. It is a very high level language that is extremely easy to pick up with minor syntax differences and more built in functions. The following two documents are essential to have on hand when developing in 4DGL. To begin with, some basic examples will be examined to familiarize the user before scanning through the full range of capabilities the PICASO-GFX2 has to offer.

- 4DGL-Programmers-Reference-Manual-revx.pdf
- PICASO-GFX2-4DGL-Internal-Functions-revx.pdf

Syntax Basics

As stated above, the syntax and structure is similar to the C language minus the curly braces. Likewise programming in basic or vbscript, the syntax will seem very familiar with the minor exception of the ";" endings.

Code Comments

Code comments in 4DGL follow the C style syntax. A single line comment will look like the following:

```
//this is a single line comment
```

A multiline comment will look like the following:

```
/*
This is a sample
Multi Line Comment
*/
```

Just like C, a single line comment with the multiline syntax can also be achieved.

Identifiers

Identifiers are names used to reference Variables, Constants, and Methods. A valid Identifier must meet the following criteria:

- Must begin with an English letter of the alphabet or an underscore.
- Can contain only alphanumeric characters or an underscore
- Must not contain: ~ ! @ # $ % ^ & * ( ) " ' _ = + { } [ ] : ; <> ? / |

"White Space" will be ignored by the compiler. "White Space" is any of the following items: Tab, Space, and empty lines.

Variables

A variable is a temporary holding place for a changing value. In many cases, it is necessary to store a value be it a number, string (any text), or Byte (Raw HEX representation of a Character or Number). All Variables in 4DGL are **signed** 16 bit. Each variable is defined with the var statement. If defined outside the main function, it is a global variable.

The following are examples of valid variable declaration statements.

```
var foo, bar;
var foobar;
var myarray[100];
var myHexValues[3]:=[0x00,0x01,0x02];
var myNumbers[9]:=[1,2,3,4,5,6,7,8,9]; //Sets the initial array
```

Methods

A method is a breakout code block that will execute when called by a function that can have parameters, which can return a value.

Sub Routines

Sub routines are methods that can execute, but will not return a value. They must follow the syntax shown below. A subroutine can only reside inside a function body and is only visible inside the function.

```
mysub:
[some code to execute here]
endsub;
```

Functions

Functions are methods that will return a value and can accept parameters. They must follow the syntax shown below. Note that Parameters and Return values are optional with functions.

```
func foo(varinputpram)
      var somevalue:=0;
      [some code to execute here]
      return somevalue;
endfunc
```

Basic Logic Statements

Control flow in 4DGL is very similar to other programming languages. Typically, this is done as an expression with operators. However, instead of using parentheses, control flow is terminated by an ending statement. **Chapter 6** of the **4DGL Programmers Reference Manual** contains a full list of Expressions and Operators used in Logic Statements.

The Basic "if … else … endif"

```
var x:=0;
if(x == 1)
    x:=0;
else
    x:=1;
endif
```

while … wend

```
var i:=0;
var val:=5;
while (i<val)
    myval:=myvar*I;
    i++;
wend
```

repeat … until

```
var x:=0;
repeat
    x++;
    //DoSomething();
until (x==100);
```

repeat … forever (*Special note this is a great way to handle your main application loop)

```
repeat
    if(ShouldExitApp()) //calls a function checking for true.
        break; //This will exit the repeat
    endif
forever
```

There are other types of Logic Statements that sometimes provide a better way of managing the flow of the application. Refer to **Chapter 7** of the **4DGL Programmers Reference Manual** for more details.

Internal Functions

With many programming languages there is generally a library or runtime of built in methods that can be called to make it easier to work with the hardware it is designed for. 4DGL is no different. The PICASO-GFX2 has a library of in-built methods called internal functions. These methods are provided to make it easier when programming with 4DGL. While this quick start will not go through all of the internal functions available, there is an entire document dedicated to outlining all possible functions and their use; titled: "PICASO-GFX2-4DGL-Internal-Functions-revx.pdf".
The following section briefly touches on some of the most prominent functions.

Graphics Primitives (Graphics Internal Functions)

Naturally, there are multiple graphical functions dedicated to displaying, changing or manipulating objects that are screen related. Listed below are a select few examples.

Clear the Display

To clear the display, the "gfx_cls" function can be used, which leaves the screen blank.

```
gfx_Cls();
```

Display a string

To display a string on the screen, the "print" internal function can be used.

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

Circle

To draw a circle on the display, the "gfx_circle" internal function can be used.

```
gfx_Circle(50,50,30,RED); // a 30 Pixel Circle outline only.
gfx_CircleFilled(5,5,10,10,RED); // a 10 Pixel Red Filled Circle.
```

Line

To draw a line on the display, use the "gfx_line", "gfx_hline", or "gfx_vline" internal methods.

```
gfx_Line(100,100,10,10,RED);
gfx_Hline(50,10,80,RED);
gfx_Vline(20,30,70,RED);
```

Touch Functions

One of the keynote features on the PICASO-GFX2 is its internal touch detection functions.

Detect Region

This method allows a touch detection region to be set out on the screen. This setting will filter out any touch activity outside the region and only touch activity within that region will be reported by the status poll touch_Get(0); function.

```
// this is used to set a detection region other than the full display.
touch_DetectRegion(x1, y1, x2, y2);
```

Touch Set

This method sets various Touch Screen related parameters.

mode = 0 : Enable Touch Screen

mode = 1 : Disable Touch Screen

mode = 2 : Default Touch Region

```
touch_set(0); //Enable the Touch Screen
```

Touch Get

This function is used to detect a change in state, or a touch on the screen. Additionally, both the X and Y coordinates of the touched can be stored in variables as seen below.

```
state := touch_Get(TOUCH_STATUS); // get touchscreen status
x := touch_Get(TOUCH_GETX);
y := touch_Get(TOUCH_GETY);
if (state == TOUCH_PRESSED) // see if Exit hit
      if ( x >170 && y > 280 ) // EXIT button
            gfx_Cls();
            exit := -1;
      endif
      if (vertical)
            if ( x > 170 && (y > 240 && y < 270 ))// Horiz button
                  vertical := 0;
                  exit := 1;
            endif
            else
            if ( x > 170 && (y >200 && y < 230 ))// Vert button
                  vertical := 1;
                  exit := 2;
            endif
      endif
endif
```

micro-SD Flash Functions

Accessing the micro-SD card is an essential technique to learn how to use. Playing videos or displaying images is paramount to many applications and thus, the micro-SD functionality is the key to unlocking these features.

Media Init

Whenever accessing the micro-SD card, it must first be initialised and scanned for a card present in the slot. Both functions can be combined into a single statement as shown below.

```
while(!media_Init())
      gfx_Cls();
      pause(300);
      puts("Please insert SD card");
      pause(300);
wend
```

Media Image

The following method displays an image read from the micro-SD card at the specified coordinates.

```
while(media_Init()==0); // wait if no SD card detected
      media_SetAdd(0x0001, 0xDA00); // point to the books04 image
      media_Image(10,10);
      gfx_Clipping(ON); // turn off clipping to see the difference
      media_Image(-12,50); // show image off-screen to the left
      media_Image(50,-12); // show image off-screen at the top
repeat forever
```

There are many other methods such as writing to the micro-SD as well as playing videos. For more details refer to pages 126-140 of the PICASO-GFX2-4DGL-Internal-Functions-revx.pdf datasheet.

7. Where can I find more information?

As with all of 4D products, visiting the 4D Systems website will provide the latest information and documentation regarding any related query. The Forums are also a great place to look for help as other users may have already posted a similarly related question. The following documents expand on much of the detail listed in this Quick Start Guide and can be viewed by clicking on their respective links.

- PICASO-GFX2-DS-revx.pdf – Data Sheet for the PICASO-GFX2
- PICASO-GFX2-4DGL-Internal-Functions-revx.pdf – Internal Functions Documentation
- 4DGL-Programmers-Reference-Manual-revx.pdf – Detailed Information about 4DGL

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.