



4D Systems

Application Note: 4D-AN-1010

4D-ViSi Application Overview

Document Date: 7th November 2011

Document Revision: 1.0

Description

This Application Note is dedicated to explaining the new 4D-ViSi Software Tool, which is part of the 4DWorkshop3 IDE suite. An overview of its basic functionality and uses will be explored. In order to use 4D-ViSi, the following items are required;

- Any 4D GFX Screen Module
 - 4D Programming Cable
 - micro-SD (μ SD) memory card
 - 4D-ViSi Software Tool
-

Application Overview

It is often difficult to design a graphical display without being able to see the immediate results of the application code. 4D-ViSi is the perfect software tool that allows the user to see the instant results of their desired graphical layout. Additionally, there is a selection of inbuilt dials, gauges and meters that can simply be dragged and dropped onto the simulated module display. From here, each can have properties edited and at the click of a button, all relevant code is produced in the user program. Each feature of 4D-ViSi will be outlined with examples below.



Setup Procedure

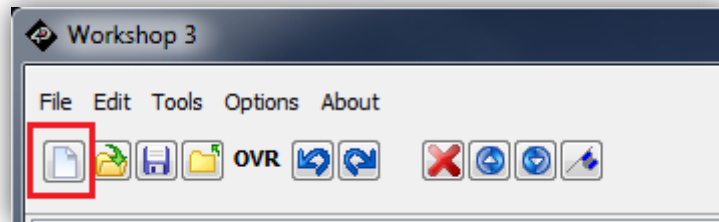
Firstly, you will need to download 4D-ViSi. It can be found from within the 4DWorkshop3 IDE product page on the 4D Systems website at the link below:

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

Simulation Procedure

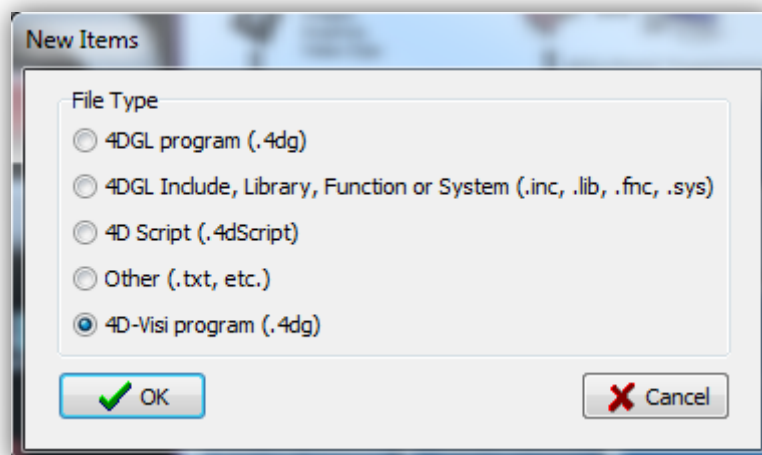
Become Familiar with the 4D-ViSi Environment

Open the 4D-ViSi Software Tool and notice that it looks virtually identical to 4DWorkshop3 IDE. Open a new file by either using the **File** dropdown box, or the **New White Page** icon in the top left hand corner.



Create a 4D-ViSi Program File

When creating a new project file, a window will appear that prompts for the type of file to be created. There is now a fifth option, which is a 4D-ViSi file. Select this option and click **OK**.

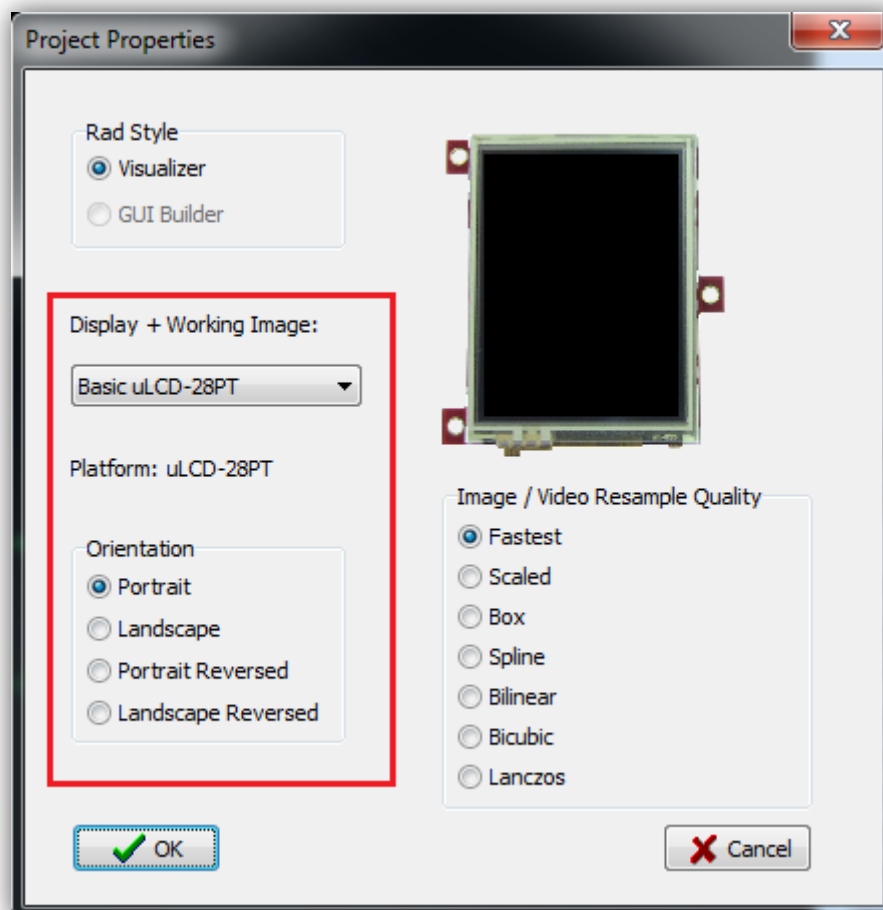


Project Properties

A new screen prompt will appear which now allows the user to create a unique project, specific to a certain module type. Start by selecting the module platform from the dropdown box in the centre of the screen. The following modules are available for development with 4D-ViSi:

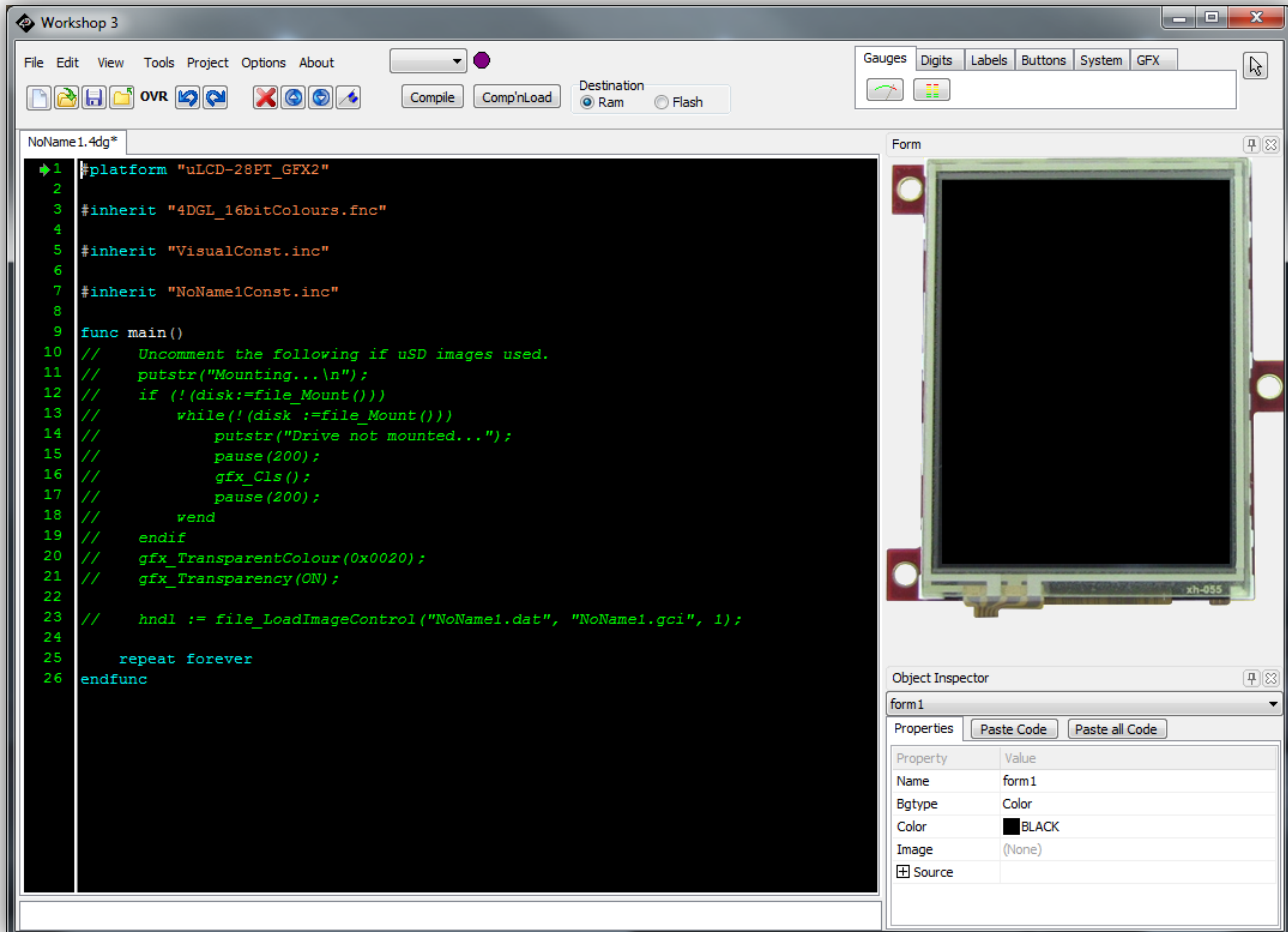
- uLCD-24PT
- uLCD-28PT
- uLCD-32PT
- uLCD-32PT (Bezel)
- uLCD-43PT
- uLCD-43PCT
- uLCD-32032-P1
- uOLED-32024-P1T
- uOLED-32028-P1T
- uVGA-II 320x240
- uVGA-II 320x240 (Monitor)
- uVGA-II 640x480 (Monitor)
- uVGA-II 800x480 (Monitor)

Select the desired **Orientation** of the layout and experiment with the options for the **Image/Video Resample Quality** and click **OK**.



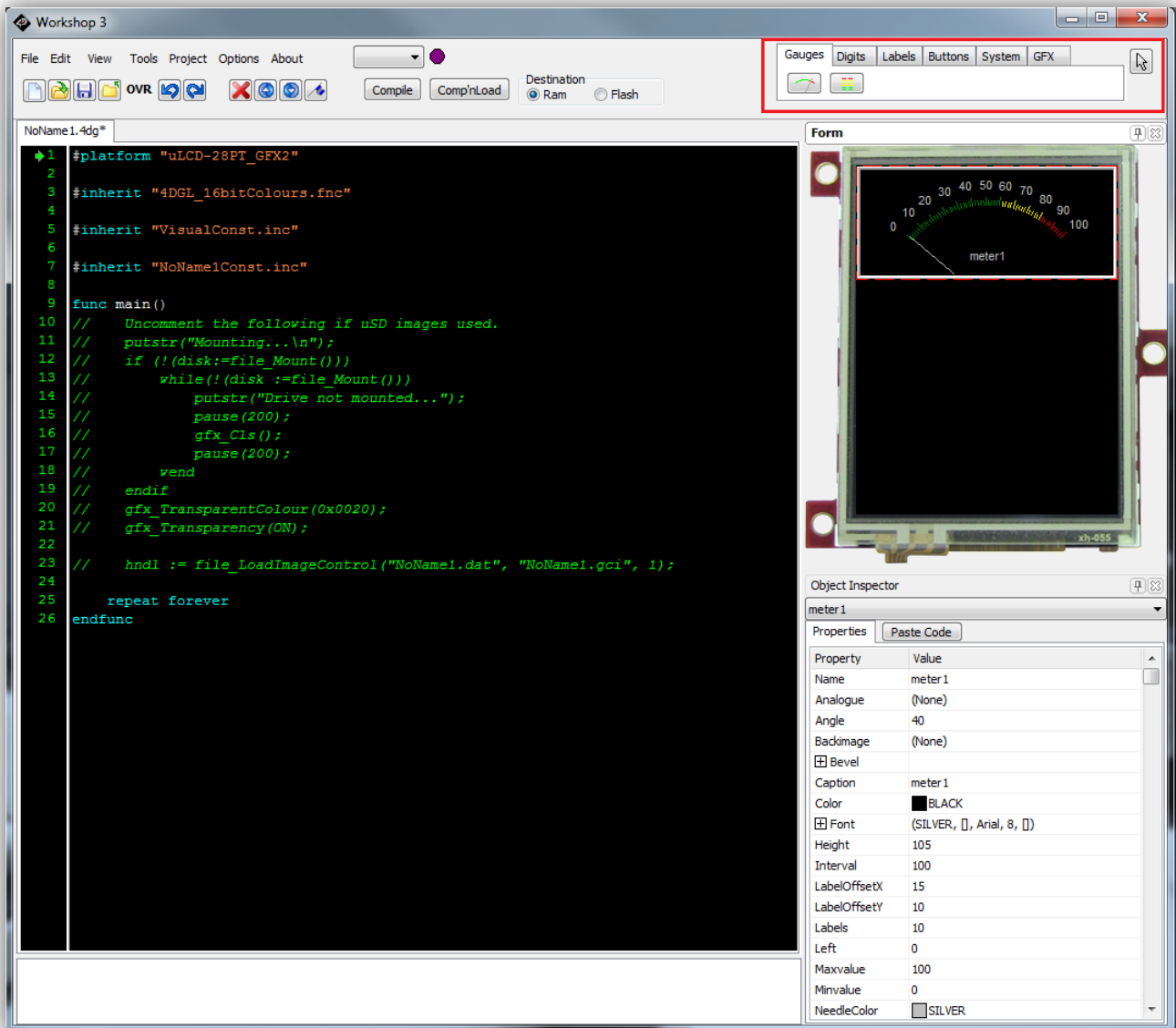
4D-ViSi Environment

The project environment will open with the new 4D-ViSi area on the right hand side of the panel. The example below shows a uLCD-28PT in Portrait mode.



Adding Graphical Objects

Become familiar with the 4D-ViSi layout by cycling through the tabs in the top right hand corner of the screen. Each tab contains a different category of graphical objects. Try placing a meter gauge by selecting the **Gauges** tab and then click on the first dial icon, followed by clicking anywhere on the display area. The meter will be placed in the area, which is now ready for positioning and customization if necessary. The following snapshot shows the gauge positioned at the top of the display area with editable properties below.



Customizing Graphical Objects

Now that the object has been placed, it can be fully customized to the desired layout by the user. Each object can be edited in one of two ways; either interactively on the display, or manually using the various fields underneath the display. Take some time to experiment with the properties available to change the way an object appears on the screen. The easiest approach to achieving the optimal layout is by adjusting the size and position of the object directly on the display by moving the cursor over each entity. It should be noted that the background display or overall binding entity is referred to as **form1**. Form1 can be edited by selecting it from the dropdown box in the **Object Inspector**. Once happy, move onto changing features such as colours, fonts, captions etc in the property fields below. Refer to the next image, which demonstrates this concept.

The screenshot displays the Workshop 3 IDE interface. On the left, the code editor shows the following code:

```

1 #platform "uLCD-28PT_GFX2"
2
3 #inherit "4DGL_16bitColours.fnc"
4
5 #inherit "VisualConst.inc"
6
7 #inherit "NoName1Const.inc"
8
9 func main()
10 // Uncomment the following if uSD images used.
11 // putstr("Mounting...\n");
12 // if (!(disk:=file_Mount()))
13 //     while(!(disk :=file_Mount()))
14 //         putstr("Drive not mounted...");
15 //         pause(200);
16 //         gfx_Cls();
17 //         pause(200);
18 //     wend
19 // endif
20 // gfx_TransparentColour(0x0020);
21 // gfx_Transparency(ON);
22
23 // hndl := file_LoadImageControl("NoName1.dat", "NoName1.goi", 1);
24
25 repeat forever
26 endfunc

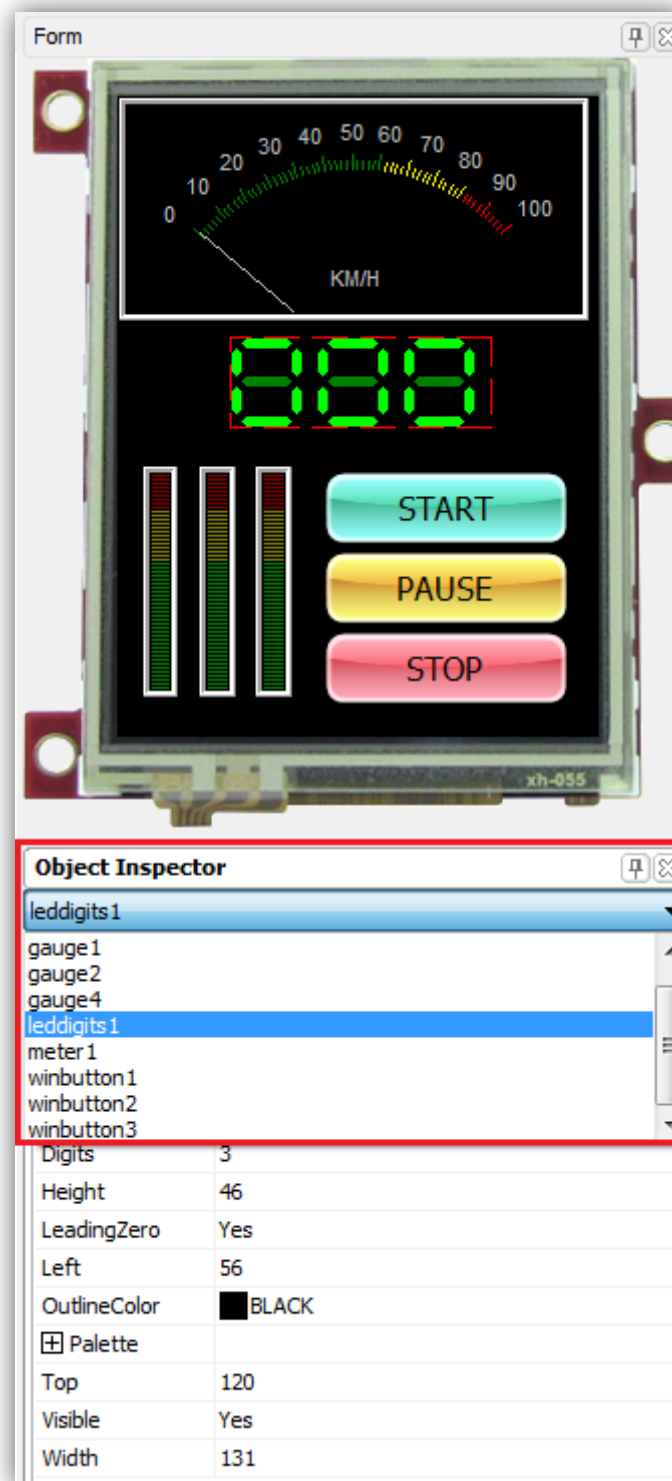
```

On the right, the graphical display shows a gauge labeled "meter1" with a needle and a scale from 0 to 90. The Object Inspector on the right shows the properties for the "meter1" object:

Property	Value
Name	meter 1
Analogue	(None)
Angle	40
Backimage	(None)
Bevel	
Caption	meter 1
Color	BLACK
Font	(SILVER, Arial, 8,)
Height	95
Interval	100
LabelOffsetX	15
LabelOffsetY	10
Labels	10
Left	44
Maxvalue	100
Minvalue	0
NeedleColor	SILVER

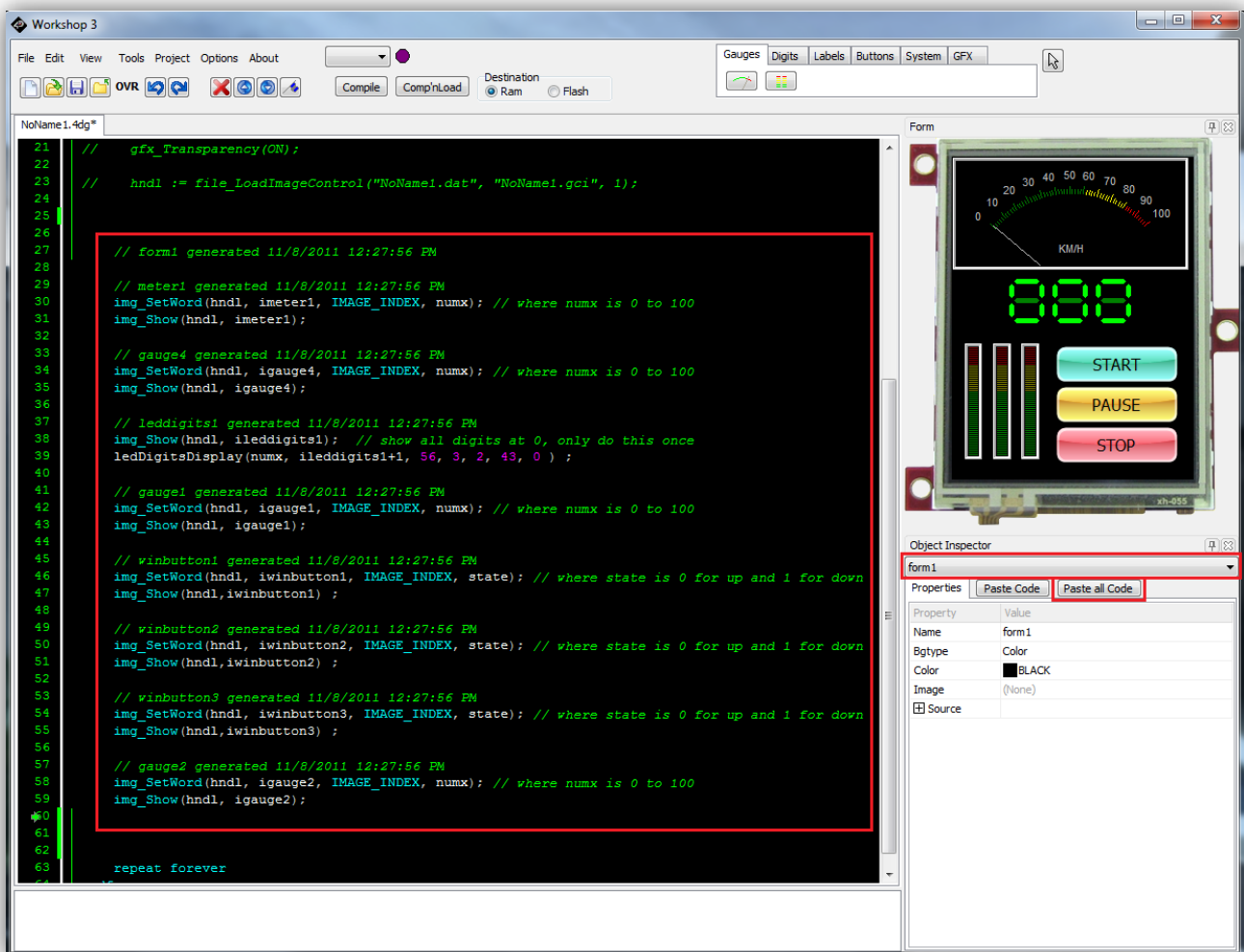
Changing Between Object Properties

If there are multiple graphical objects on the screen, it may be necessary to change between object properties. Do this by using the dropdown box located at the top of the **Object Inspector**. Alternatively, each object can be edited by clicking on the object in the module display area.

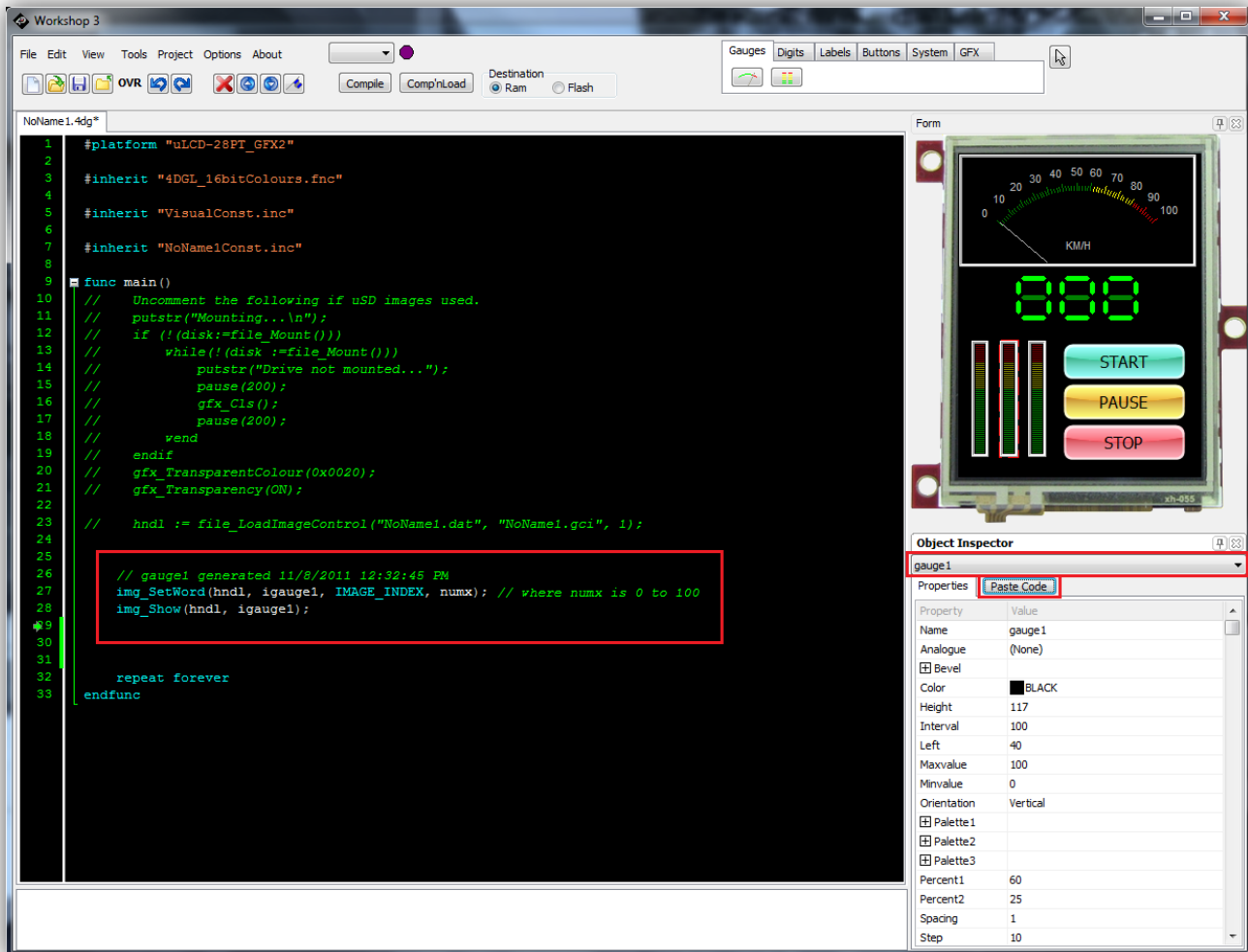


Inserting a Layout into the User Application Program

When an image layout is finalised, it is ready to have the relevant code behind each object inserted into the user program. From inside the application program, find the desired place to insert the image created using 4D-ViSi. There are two possible options for inserting code into the application. The user can either paste all of the code in one place for all of the objects created on the screen; or only paste code for an individual object. Pasting all of the code can only be done by selecting **form1**. All other objects only have the ability to generate code for themselves. The following images illustrate how to paste code into the user application. Select **form1**, then click **Paste all Code** just below it. Notice all of the code is inserted into the program.



Often, not all code will be needed in the one place in a given user program. For this reason, it will be necessary to paste code for a particular object only. The next image demonstrates this point. Select an **object** then click on **Paste Code**. Notice that only a couple of lines of code are inserted into the program.



Finalising a Program for Download

Rounding off a project requires the user to develop the necessary code to control the flow of how the application runs. This will involve writing control loops and sequences in between all of the image screens that have been created using 4D-ViSi. It is now apparent, that this method of application development has now significantly reduced in time, since all image layouts are generated by 4D-ViSi, which normally the user would have to do. Not only is time saved in this respect, but also code will no longer be required to be downloaded onto the module every time a visual change is made. Instant graphical changes are now seen in the simulated display, which means an application download will only need to be done once; at the completion of a user program.

Downloading an Application

When a 4D-ViSi program is compiled, a .gci and .dat file is generated automatically without going through the Graphics Composer software tool. Make sure that the module is plugged into the PC and the COM port is selected. A prompt will appear requiring that the .gci and .dat files are copied onto the micro-SD before proceeding. Before clicking OK, Insert the micro-SD card into the PC and locate the .gci and .dat file created during compile. These will be located in the same folder as the project file. Copy these to the micro-SD and insert the card back into the module and click OK. The application will finish downloading to the module and the application will now run.

Example Application Program

The following example was developed by the 4D Team to illustrate some of the features and possibilities with the new 4D-ViSi software tool. The screen exhibits three gauges, a counter and a meter. All three graphic tools are controlled by the same index and hence all reflect the same value between 1 and 100. Three buttons control the meters titled; Start, Pause and Stop. The Start button will begin the counter counting from either zero or wherever the counter is up to. The counter resets every time it reaches 100. The Pause button will stop the counter wherever it is up to; whilst the Stop button will reset the counter to zero. Examine the code to see just how simple it is to generate a fully graphic and interactive program with only a few lines of code. Comments will be placed throughout the code to explain certain areas in detail. Place the .4dg and .4DVisi file that comes with this application note into a project folder on the PC. Open the .4dg file through 4D-ViSi to load the example application. Be sure to follow the steps explained in the previous section for loading the .gci file and .dat file onto the micro-SD in order for the program to work.



```
#platform "uLCD-32032-P1_GFX2"
#inherit "4DGL_16bitColours.fnc"
#inherit "VisualConst.inc"
#inherit "4DViSi 4D-AN-1010Const.inc"
#inherit "ledDigitsDisplay.inc"

var x,y;
var mindex;

func main()

    putstr("Mounting...");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount())) // check micro-SD is loaded
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    gfx_TransparentColour(0x0020);
    gfx_Transparency(ON);

    hndl := file_LoadImageControl("4DVISI~1.dat", "4DVISI~1.gci", 1);

    img_Show(hndl,iwinbutton1); // meter1 generated 11/8/2011 2:05:10 PM
    img_Show(hndl,iwinbutton2); // winbutton2 generated 11/8/2011 2:05:10 PM
    img_Show(hndl,iwinbutton3); // winbutton3 generated 11/8/2011 2:05:10 PM
    img_Show(hndl, ileddigits1); // leddigits1 generated 11/10/2011 12:54:00 PM
    img_Show(hndl, igauge4); // gauge4 generated 11/10/2011 12:56:25 PM
    img_Show(hndl, igauge1); // gauge1 generated 11/10/2011 4:39:17 PM
    img_Show(hndl, igauge2); // gauge2 generated 11/10/2011 12:59:28 PM

    mindex := 0; // This is the index that controls the value of all the meters

    touch_Set(TOUCH_ENABLE); // enable touch

    repeat

        if(touch_Get(TOUCH_STATUS) == TOUCH_PRESSED) // scan for touch
            x := touch_Get(TOUCH_GETX); // get x coordinate
            y := touch_Get(TOUCH_GETY); // get y coordinate
            if( (x >= 104 && x <= 224) && (y >= 188 && y <= 223) ) // coordinates
                for Start Button
                    img_SetWord(hndl, iwinbutton3, IMAGE_INDEX, 1); // show start button
                depressed
                    img_Show(hndl,iwinbutton3);
                    pause(200);
                    img_SetWord(hndl, iwinbutton3, IMAGE_INDEX, 0); // release start
                button
                    img_Show(hndl,iwinbutton3);
                repeat
```

```

        mindex++; // increment the index
        if( mindex == 100 ) // if the index reaches 100, reset it to 0
            mindex := 0;
        endif
        img_SetWord(hndl, imeter1, IMAGE_INDEX, mindex); // update
imeter1
        img_Show(hndl, imeter1);
        ledDigitsDisplay(mindex, ileddigits1+1, 56, 3, 2, 43, 0 ); //
update leddigits1
        img_SetWord(hndl, igauge4, IMAGE_INDEX, mindex); // update
igauge4
        img_Show(hndl, igauge4);
        img_SetWord(hndl, igauge1, IMAGE_INDEX, mindex); // update
ugauge1
        img_Show(hndl, igauge1);
        img_SetWord(hndl, igauge2, IMAGE_INDEX, mindex); // update
igauge2
        img_Show(hndl, igauge2);
        if(touch_Get(TOUCH_STATUS) == TOUCH_PRESSED)
            x := touch_Get(TOUCH_GETX);
            y := touch_Get(TOUCH_GETY);
            if( ( (x >= 104 && x <= 224) && (y >= 228 && y <= 263) ) ||
( (x >= 104 && x <= 224) && (y >= 268 && y <= 303) ) )
                break; // if a touch is detected on either of the other
buttons, break out of this forever loop
            endif
        endif
        forever
        endif
        if( (x >= 104 && x <= 224) && (y >= 228 && y <= 263) ) // coordinates
for Pause Button
            img_SetWord(hndl, iwinbutton2, IMAGE_INDEX, 1); // show pause button
depressed
            img_Show(hndl,iwinbutton2);
            pause(200);
            img_SetWord(hndl, iwinbutton2, IMAGE_INDEX, 0); // release pause
button
            img_Show(hndl,iwinbutton2);
            img_SetWord(hndl, imeter1, IMAGE_INDEX, mindex); // update imeter1
            img_Show(hndl, imeter1);
            ledDigitsDisplay(mindex, ileddigits1+1, 56, 3, 2, 43, 0 ); // update
leddigits1
            img_SetWord(hndl, igauge4, IMAGE_INDEX, mindex); // update igauge4
            img_Show(hndl, igauge4);
            img_SetWord(hndl, igauge1, IMAGE_INDEX, mindex); // update igauge1
            img_Show(hndl, igauge1);
            img_SetWord(hndl, igauge2, IMAGE_INDEX, mindex); // update igauge2
            img_Show(hndl, igauge2);
        endif
        if( (x >= 104 && x <= 224) && (y >= 268 && y <= 303) ) // coordinates
for Stop Button
            img_SetWord(hndl, iwinbutton1, IMAGE_INDEX, 1); // show stop button
depressed

```

```
img_Show(hndl,iwinbutton1);
pause(200);
img_SetWord(hndl, iwinbutton1, IMAGE_INDEX, 0); // release stop
button
img_Show(hndl,iwinbutton1);
mindex := 0; // reset the index
img_SetWord(hndl, imeter1, IMAGE_INDEX, mindex); // update imeter1
img_Show(hndl, imeter1);
leddigits1
ledDigitsDisplay(mindex, ileddigits1+1, 56, 3, 2, 43, 0 ); // update
img_SetWord(hndl, igauge4, IMAGE_INDEX, mindex); // update igauge4
img_Show(hndl, igauge4);
img_SetWord(hndl, igauge1, IMAGE_INDEX, mindex); // update igauge1
img_Show(hndl, igauge1);
img_SetWord(hndl, igauge2, IMAGE_INDEX, mindex); // update igauge2
img_Show(hndl, igauge2);
endif
endif
forever
endfunc
```

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.