USER GUIDE

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

Workshop 4 - ViSi User Guide

Document Date: 23rd September 2013
Document Revision: 1.2

Contents

1. Description **4**

2. Application Overview **4**

3. Setup Procedure **4**

4. Launch Workshop 4 .... **5**

5. Create a New Project **6**

6. Select ViSi **9**

7. The Main Screen **10**

- 7.1. Area 1: Menus 11
- 7.2. Area 2: Ribbon with Icons 11
- 7.3. Area 3: Code 12
- 7.4. Area 4: Form and WYSIWYG Screen 12
- 7.5. Area 5: Object Inspector 13
- 7.6. Area 6: Message Window 13
- 7.7. Move the Form and Object Inspector 13

8. A First ViSi Project **15**

- 8.1. Adding Graphical Objects 15
- 8.2. Customizing Graphical Objects 16
- 8.3. Changing Between Object Properties 17
- 8.4. Inserting a Layout into the User Application Program 18
- 8.5. Finalising a Program for Download 19
- 8.6. Downloading an Application 19
- 8.7. Example Application Program 19

9. Objects **21**

- 9.1. Buttons Object 21
- 9.2. Digits Objects 22
- 9.3. Gauges Objects 23
- 9.4. Primitives Objects 25
- 9.5. Inputs Objects 27
- 9.6. Labels Objects 31
- 9.7. System/Media Objects 33
- 9.8. Selection Tool 36

10. Connect the Screen **37**

11. Insert the Micro-SD Card **38**

12. Communication Terminal **39**

13. Application Notes **42**

14. Revision History **43**

15. Legal Notice **44**

16. Contact Information **44**

1. Description

This Application Note is dedicated to explaining the new 4D-ViSi Software Tool, which is part of the Workshop 4 IDE suite.

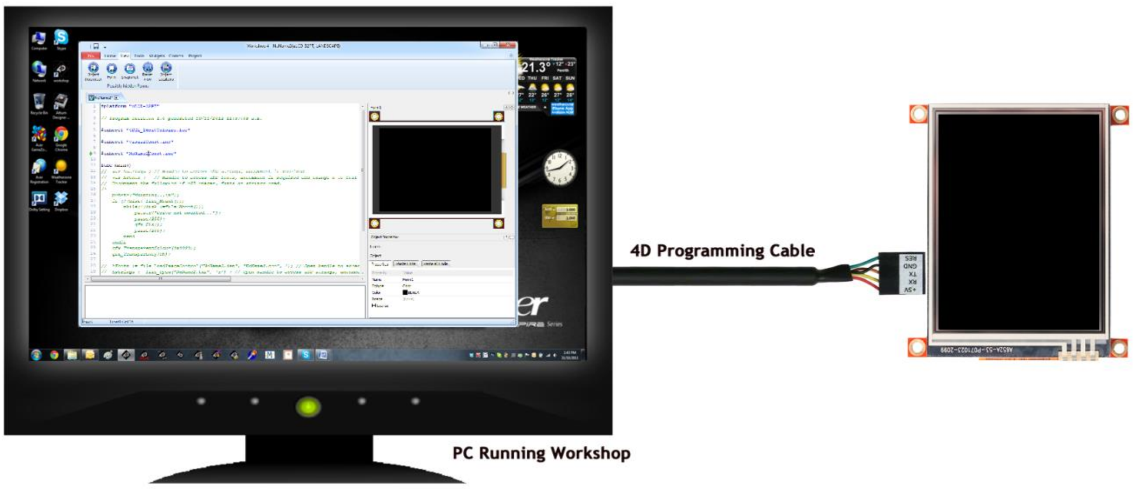
An overview of its basic functionality and uses will be explored. In order to use 4D-ViSi, the following items are required:

- Any 4D Systems Display Module
- 4D Programming Cable or Adaptor
- A micro-SD memory card
- 4D Systems Workshop 4 IDE

2. 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.

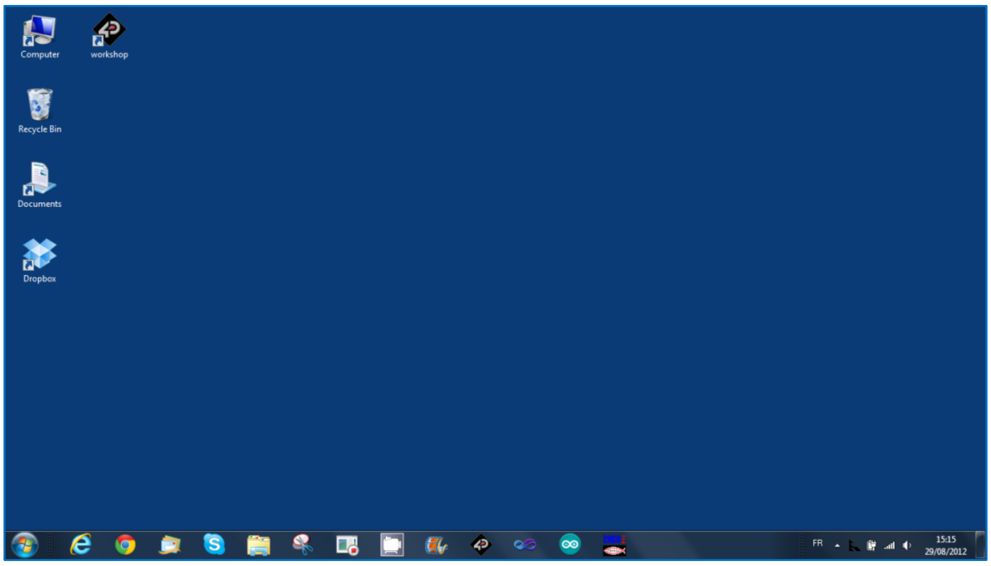


3. Setup Procedure

Firstly, you will need to download 4D-ViSi. It can be found from within the 4D Workshop 4 IDE product page on the 4D Systems website, www.4dsystems.com.au

4. Launch Workshop 4

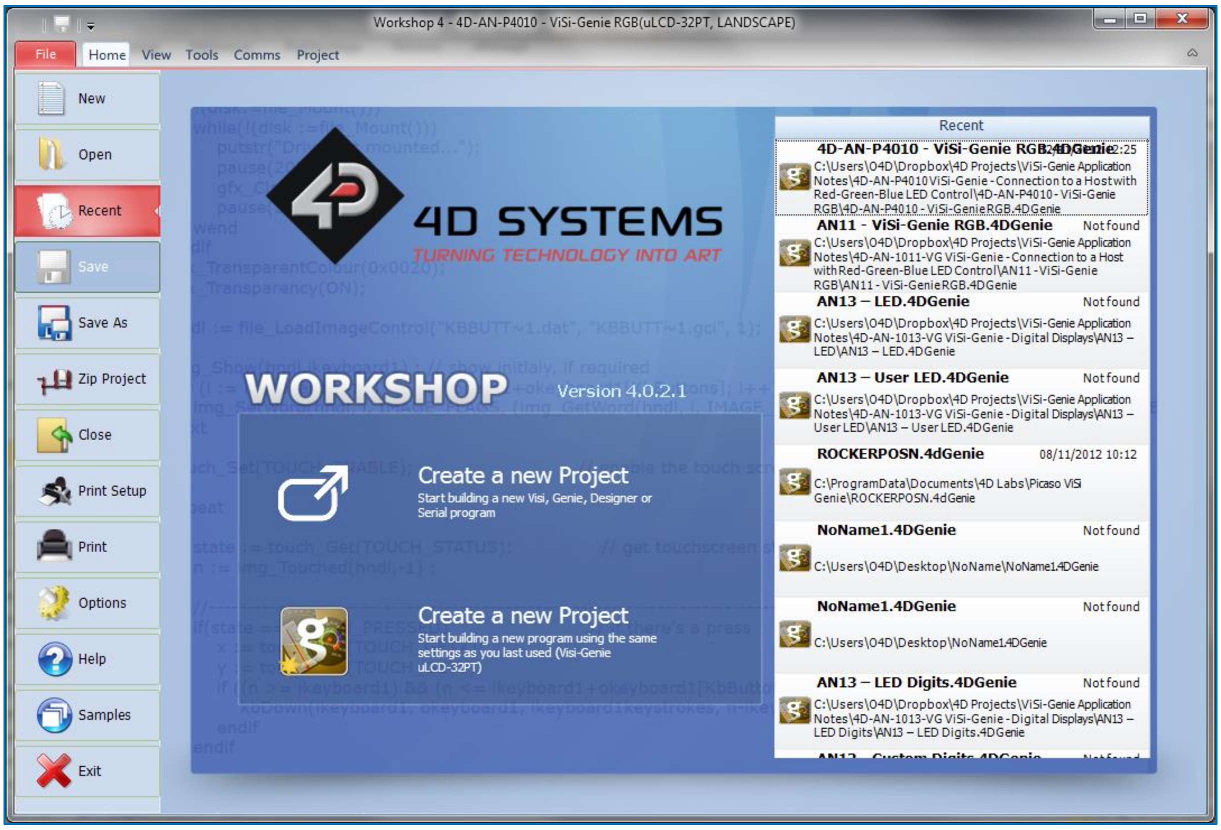
There is an alias for 4D Workshop on the desktop:



Launch 4D Workshop by double-clicking on the icon:

5. Create a New Project

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


Workshop 4 - 4D-AN-P4010 - ViSi-Genie RGB(uLCD-32PT, LANDSCAPE)
File
Home
View
Tools
Comms
Project
New
Open
Recent
Save
Save As
Zip Project
Close
Print Setup
Print
Options
Help
Samples
Exit
4D SYSTEMS
TURNING TECHNOLOGY INTO ART
WORKSHOP
Version 4.0.2.1
Create a new Project
Start building a new Visi, Genie, Designer or Serial program
Create a new Project
Start building a new program using the same settings as you last used (Visi-Genie uLCD-32PT)
Recent

4D-AN-P4010 - ViSi-Genie RGB.4DGenie 2:25
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Application Notes\4D-AN-P4010 ViSi-Genie - Connection to a Host with Red-Green-Blue LED Control\4D-AN-P4010 - ViSi-Genie RGB\4D-AN-P4010 - ViSi-Genie RGB.4DGenie

AN11 - ViSi-Genie RGB.4DGenie Not found
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Application Notes\4D-AN-1011-VG ViSi-Genie - Connection to a Host with Red-Green-Blue LED Control\AN11 - ViSi-Genie RGB\AN11 - ViSi-Genie RGB.4DGenie

AN13 – LED.4DGenie Not found
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Application Notes\4D-AN-1013-VG ViSi-Genie - Digital Displays\AN13 – LED\AN13 – LED.4DGenie

AN13 – User LED.4DGenie Not found
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Application Notes\4D-AN-1013-VG ViSi-Genie - Digital Displays\AN13 – User LED\AN13 – User LED.4DGenie

ROCKERPOSN.4dGenie 08/11/2012 10:12
C:\ProgramData\Documents\4D Labs\Picaso ViSi Genie\ROCKERPOSN.4dGenie

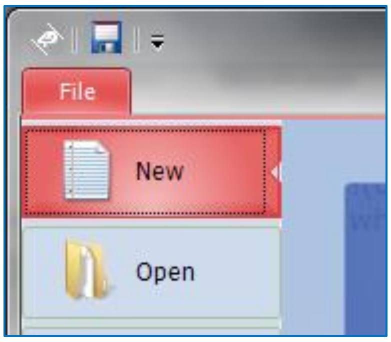
NoName1.4DGenie Not found
C:\Users\O4D\Desktop\NoName\NoName1.4DGenie

NoName1.4DGenie Not found
C:\Users\O4D\Desktop\NoName1.4DGenie

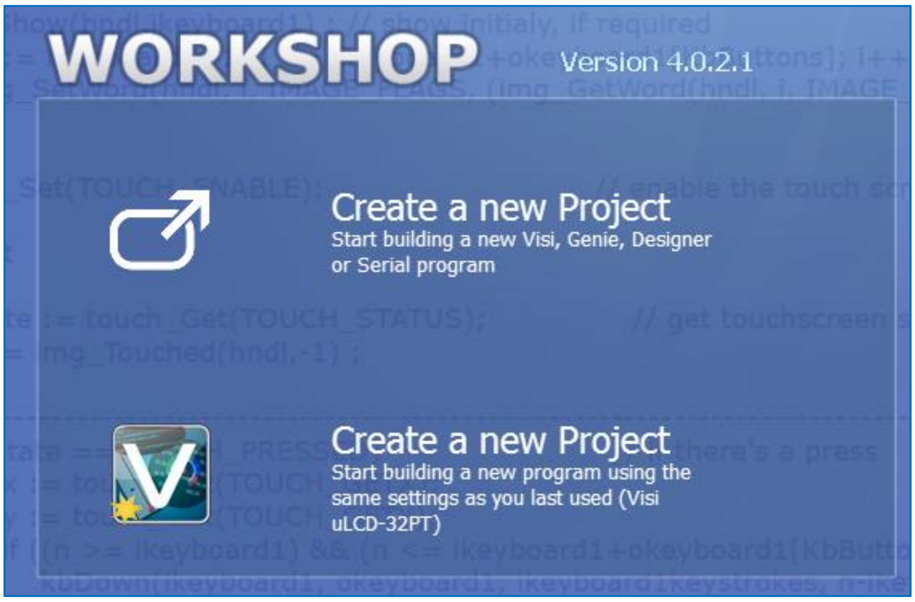
AN13 – LED Digits.4DGenie Not found
C:\Users\O4D\Dropbox\4D Projects\ViSi-Genie Application Notes\4D-AN-1013-VG ViSi-Genie - Digital Displays\AN13 – LED Digits\AN13 – LED Digits.4DGenie

To create a new program, there are multiple options:

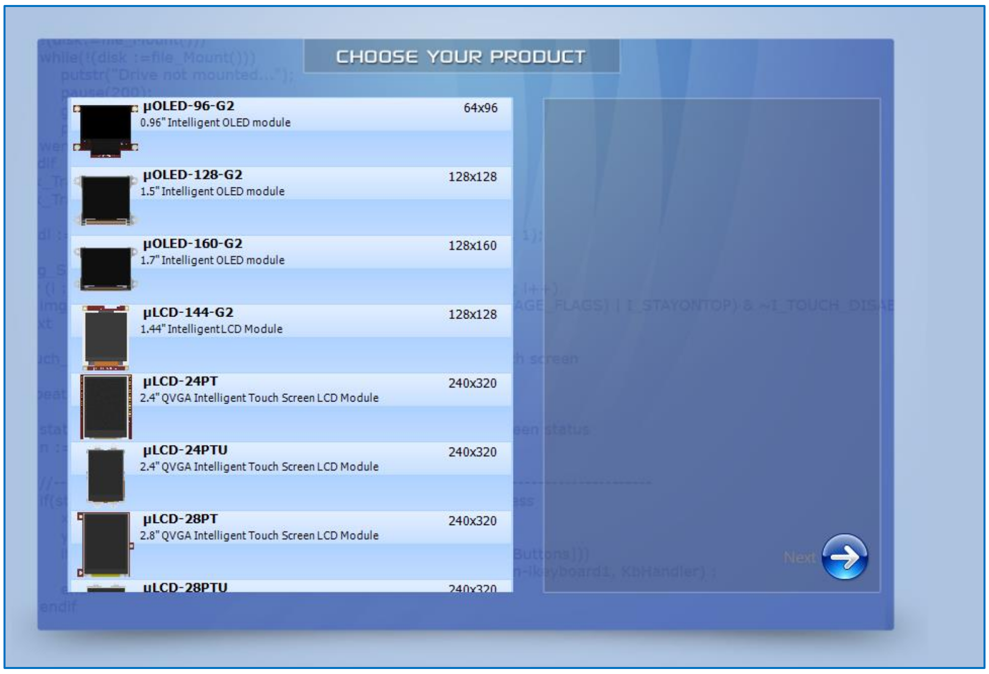
- Click on the top left-most icon **New**



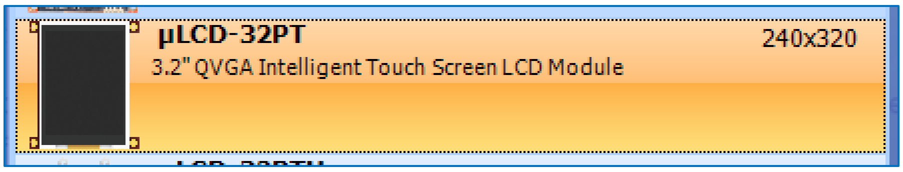
- Click on the icon close to **Create a New Project** on top or, if the settings have been already defined, click on the icon close to **Create a New Project** on bottom:


WORKSHOP
Version 4.0.2.1
Create a new Project
Start building a new Visi, Genie, Designer or Serial program
Create a new Project
Start building a new program using the same settings as you last used (Visi uLCD-32PT)

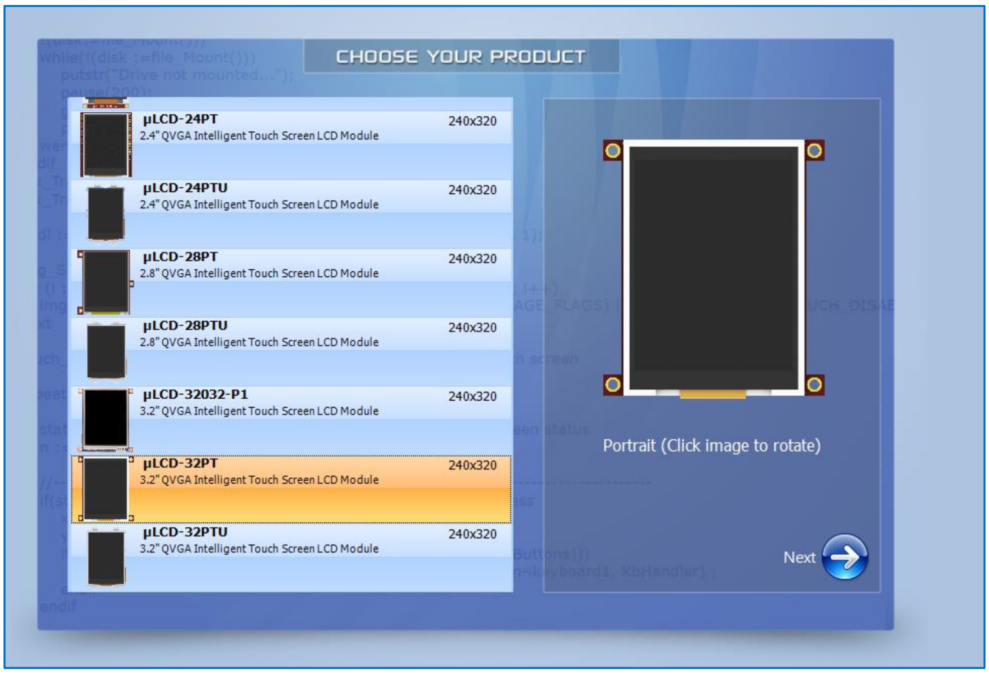
All those options update the main window with the selection of the screen:


CHOOSE YOUR PRODUCT
µOLED-96-G2 64x96
0.96" Intelligent OLED module
µOLED-128-G2 128x128
1.5" Intelligent OLED module
µOLED-160-G2 128x160
1.7" Intelligent OLED module
µLCD-144-G2 128x128
1.44" Intelligent LCD Module
µLCD-24PT 240x320
2.4" QVGA Intelligent Touch Screen LCD Module
µLCD-24PTU 240x320
2.4" QVGA Intelligent Touch Screen LCD Module
µLCD-28PT 240x320
2.8" QVGA Intelligent Touch Screen LCD Module
µLCD-28PTU 240x320
Next

Select the screen, here the µLCD-32PT:

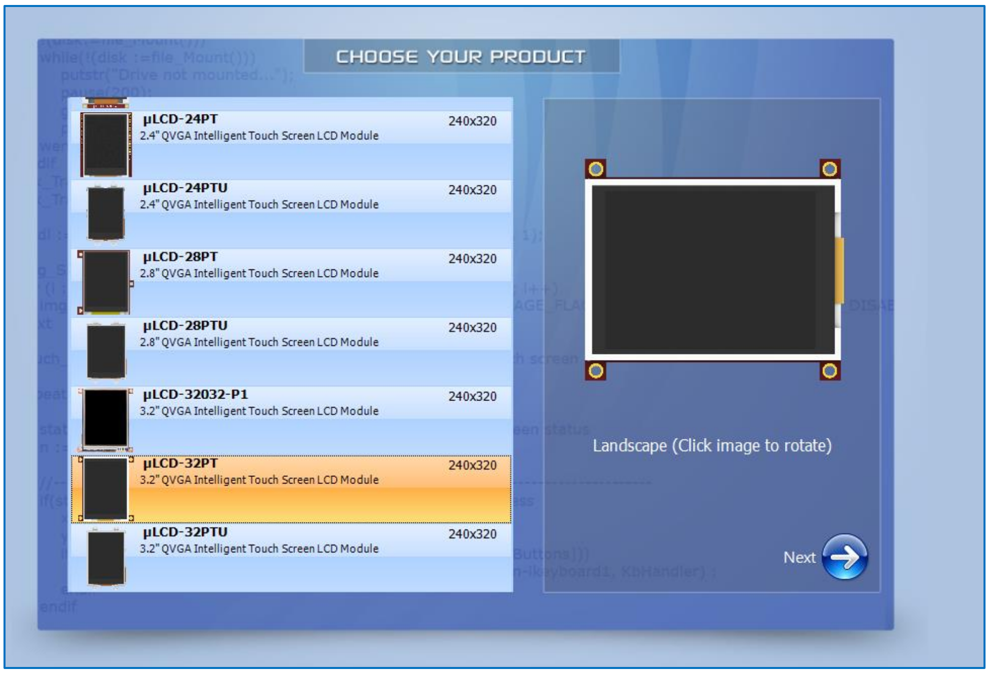


The selected screen is displayed:


CHOOSE YOUR PRODUCT
μLCD-24PT 2.4" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-24PTU 2.4" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-28PT 2.8" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-28PTU 2.8" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-32032-P1 3.2" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-32PT 3.2" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-32PTU 3.2" QVGA Intelligent Touch Screen LCD Module 240x320
Portrait (Click image to rotate)
Next

Orientation is portrait by default.

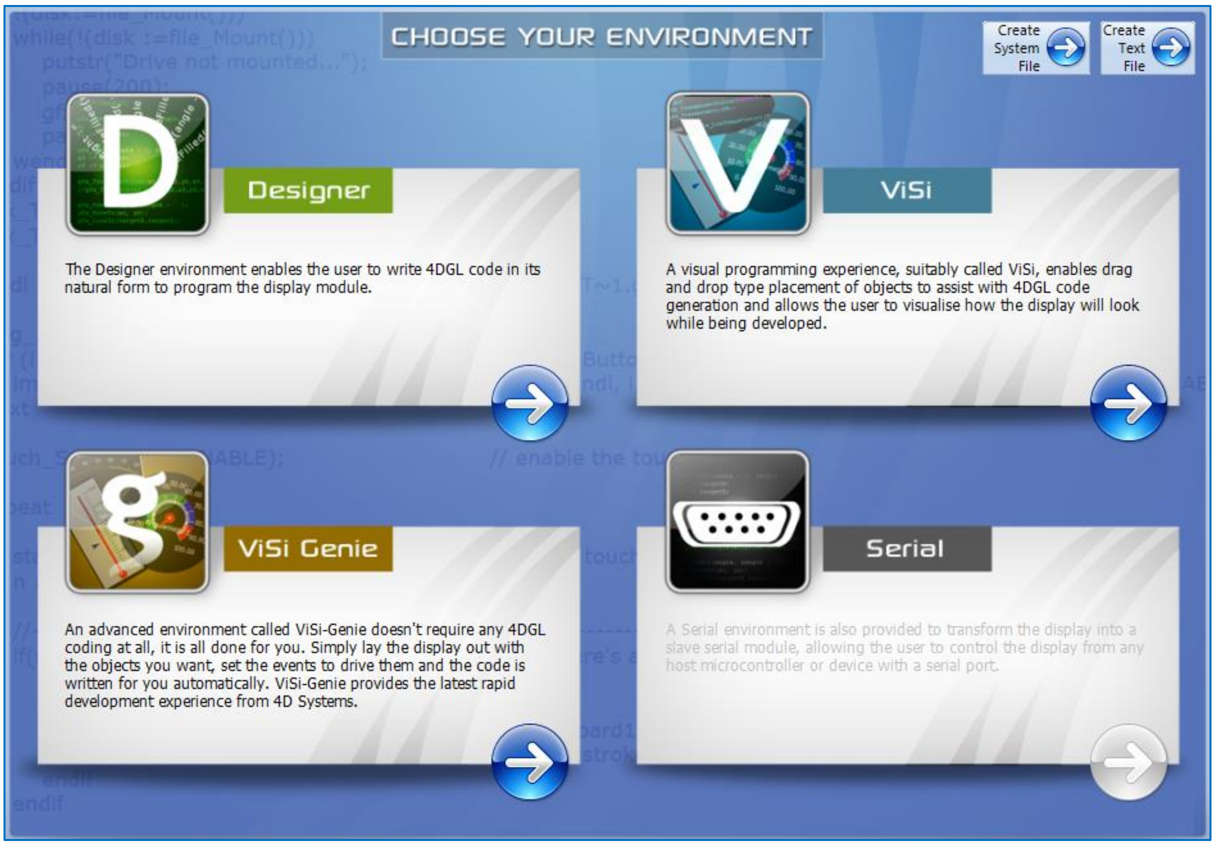
To set it to landscape, just click on the image of the screen to rotate it:


CHOOSE YOUR PRODUCT
μLCD-24PT 2.4" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-24PTU 2.4" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-28PT 2.8" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-28PTU 2.8" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-32032-P1 3.2" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-32PT 3.2" QVGA Intelligent Touch Screen LCD Module 240x320
μLCD-32PTU 3.2" QVGA Intelligent Touch Screen LCD Module 240x320
Landscape (Click image to rotate)
Next

Press **Next** to proceed:

6. Select ViSi

The main window now asks for the kind of project:



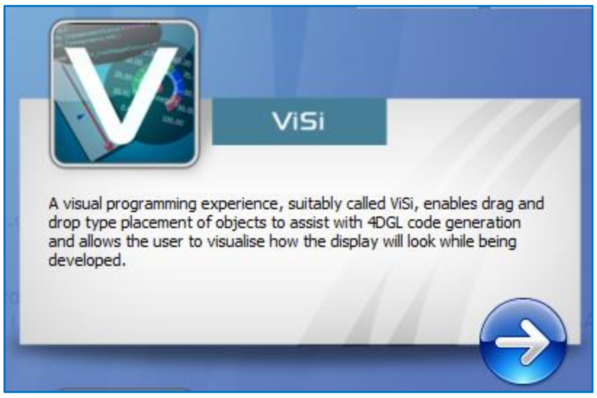
The Designer environment enables the user to write 4DGL code in its natural form to program the display module.

A visual programming experience, suitably called ViSi, enables drag and drop type placement of objects to assist with 4DGL code generation and allows the user to visualise how the display will look while being developed.

An advanced environment called ViSi-Genie doesn't require any 4DGL coding at all, it is all done for you. Simply lay the display out with the objects you want, set the events to drive them and the code is written for you automatically. ViSi-Genie provides the latest rapid development experience from 4D Systems.

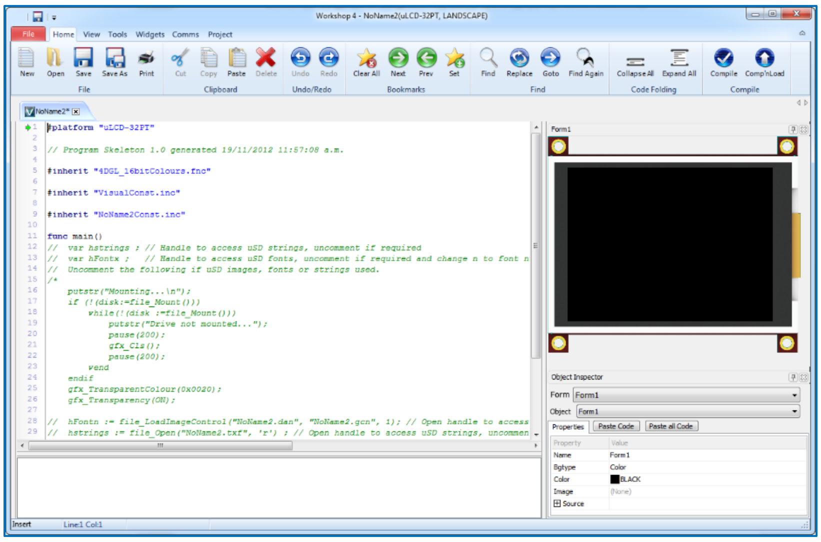
A Serial environment is also provided to transform the display into a slave serial module, allowing the user to control the display from any host microcontroller or device with a serial port.

To select ViSi Genie, just click on the blue arrow:



A visual programming experience, suitably called ViSi, enables drag and drop type placement of objects to assist with 4DGL code generation and allows the user to visualise how the display will look while being developed.

The development environment is now displayed:


Workshop 4 - NoName2(uLCD-32PT, LANDSCAPE)
File Home View Tools Widgets Comms Project
New Open Save Save As Print Cut Copy Paste Delete Undo Redo Clear All Next Prev Set Find Replace Goto Find Again Collapse All Expand All Compile Comp'nLoad
File Clipboard Undo/Redo Bookmarks Find Code Folding Compile
NoName2*
Form1
Object Inspector
Properties Paste Code Paste all Code
Insert Line:1 Col:1

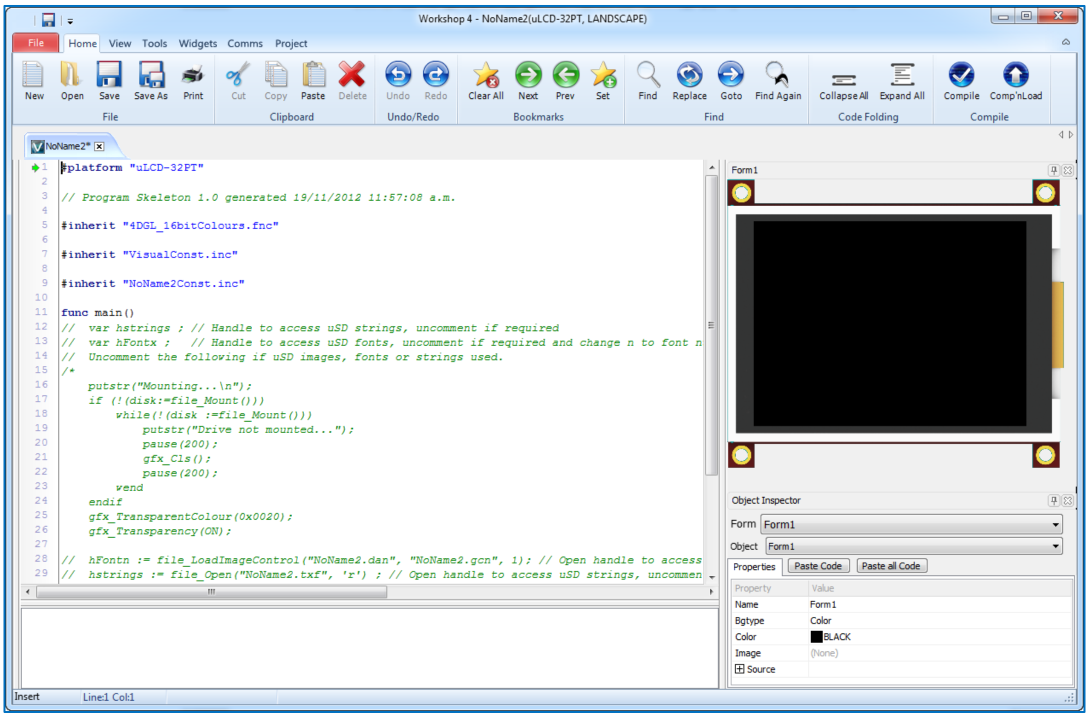
```
#platform "uLCD-32PT"

// Program Skeleton 1.0 generated 19/11/2012 11:57:08 a.m.

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName2Const.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required and change n to font n
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    gfx_TransparentColour(0x0020);
    gfx_Transparency(ON);

//  hFontn := file_LoadImageControl("NoName2.dan", "NoName2.gcn", 1); // Open handle to access
//  hstrings := file_Open("NoName2.txt", 'r') ; // Open handle to access uSD strings, uncommen
```

Form: Form1
Object: Form1

Property	Value
Name	Form1
Bgtype	Color
Color	■BLACK
Image	(None)
⊞ Source	

7. The Main Screen

The main screen appears:


Workshop 4 - NoName2(uLCD-32PT, LANDSCAPE)
File Home View Tools Widgets Comms Project
New Open Save Save As Print Cut Copy Paste Delete Undo Redo Clear All Next Prev Set Find Replace Goto Find Again Collapse All Expand All Compile Comp'nLoad
File Clipboard Undo/Redo Bookmarks Find Code Folding Compile
NoName2*
Form1
Object Inspector
Properties Paste Code Paste all Code
Insert Line:1 Col:1

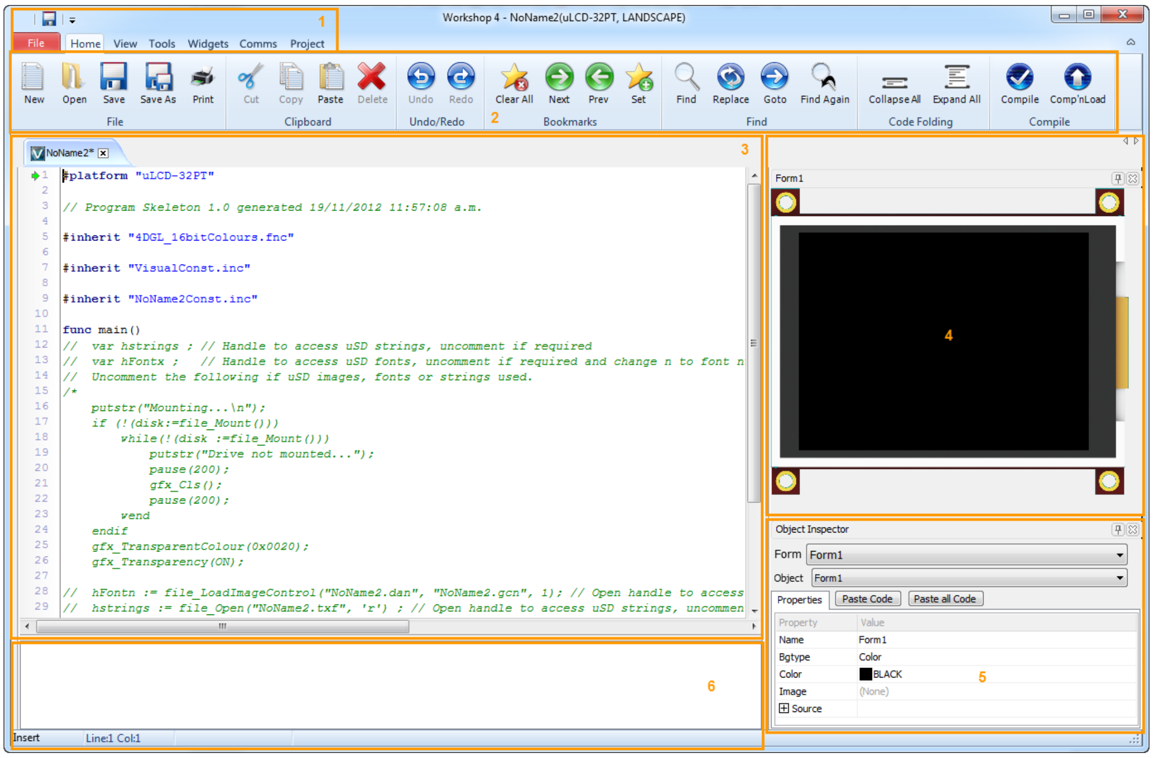
```
#platform "uLCD-32PT"

// Program Skeleton 1.0 generated 19/11/2012 11:57:08 a.m.

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName2Const.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required and change n to font n
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    gfx_TransparentColour(0x0020);
    gfx_Transparency(ON);

//  hFontn := file_LoadImageControl("NoName2.dan", "NoName2.gcn", 1); // Open handle to access
//  hstrings := file_Open("NoName2.txt", 'r') ; // Open handle to access uSD strings, uncommen
```

Form: Form1
Object: Form1

Property	Value
Name	Form1
Bgtype	Color
Color	■BLACK
Image	(None)
⊞ Source	

Let’s detail the different areas.

There are six different areas, from left to right, for top to bottom:



```
#platform "uLCD-32PT"

// Program Skeleton 1.0 generated 19/11/2012 11:57:08 a.m.

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName2Const.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required and change n to font n
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    gfx_TransparentColour(0x0020);
    gfx_Transparency(ON);

//  hFontn := file_LoadImageControl("NoName2.dan", "NoName2.gcn", 1); // Open handle to access
//  hstrings := file_Open("NoName2.txf", 'r') ; // Open handle to access uSD strings, uncommen
```

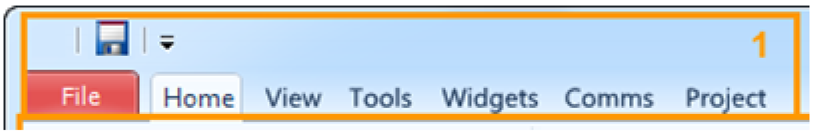
Form: Form1
Object: Form1

Property	Value
Name	Form1
Bgtype	Color
Color	■BLACK
Image	(None)
⊞ Source	

1. Menus;
2. Ribbon with icons;
3. Code;
4. Form and WYSIWYG screen where to place the objects;
5. Object inspector, where properties and events are defined;
6. Messages about errors, warnings and notices.

7.1. Area 1: Menus

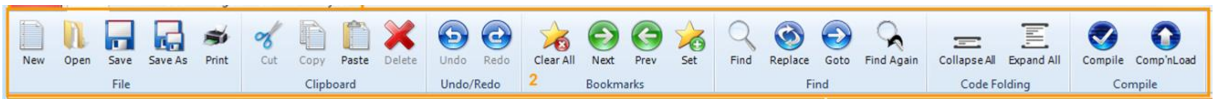
The menus include standard Windows options. Each menu displays a specific ribbon.



The debugger called **Genie Test Executor** is located under the Tool menu.

7.2. Area 2: Ribbon with Icons

For the Home menu, the ribbon includes the file related buttons and the objects grouped in panes:

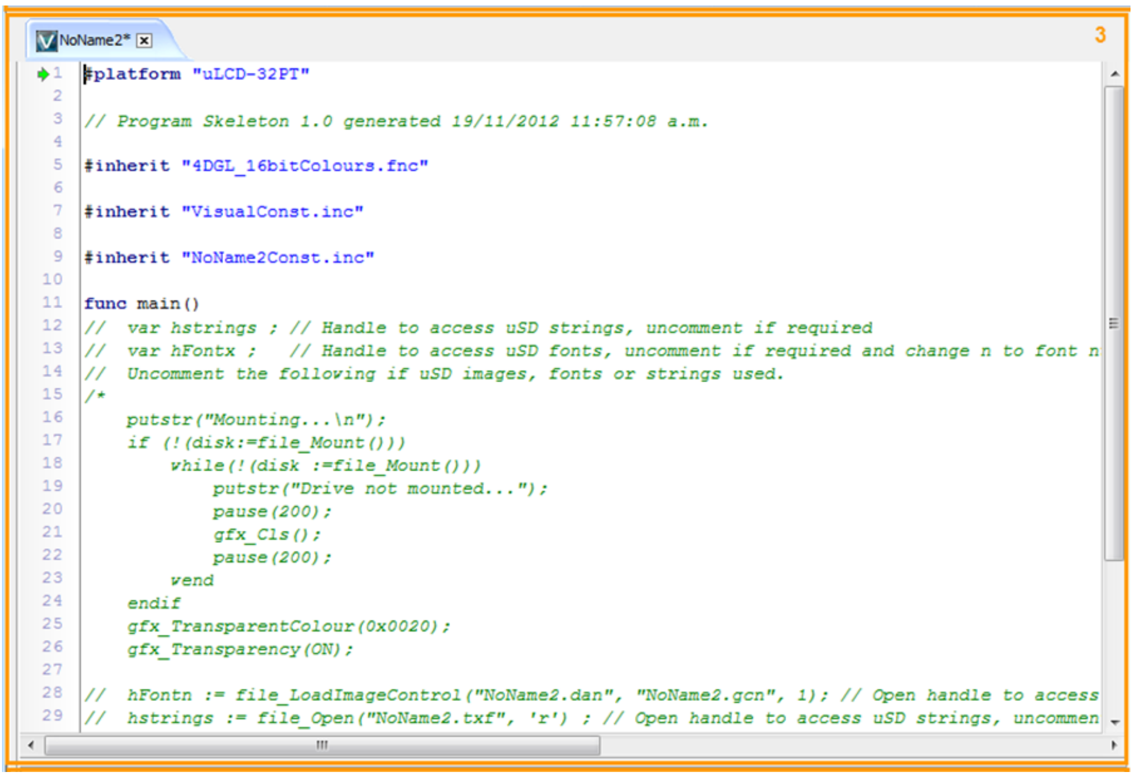


The icons related to the files include **New** project, **Open** project, **Save** project, **Save as** project, **Print** project, and **Build** project.

The **Compile** button compiles the project while the **Comp'nLoad** button compiles and uploads it to the screen.

7.3. Area 3: Code

On top of the code, the open projects are displayed:

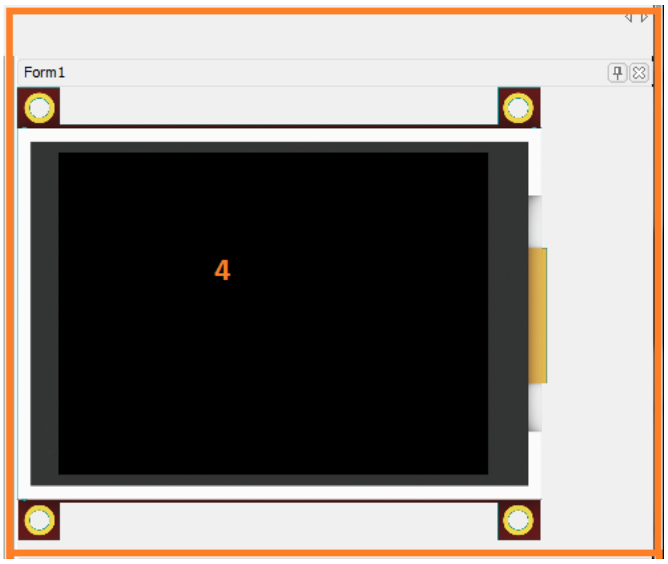


```
#platform "uLCD-32PT"

// Program Skeleton 1.0 generated 19/11/2012 11:57:08 a.m.

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName2Const.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required and change n to font n
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    gfx_TransparentColour(0x0020);
    gfx_Transparency(ON);

//  hFontn := file_LoadImageControl("NoName2.dan", "NoName2.gcn", 1); // Open handle to access
//  hstrings := file_Open("NoName2.txf", 'r') ; // Open handle to access uSD strings, uncommen
```

Click on the tab to open a project or on the cross to close it.

7.4. Area 4: Form and WYSIWYG Screen

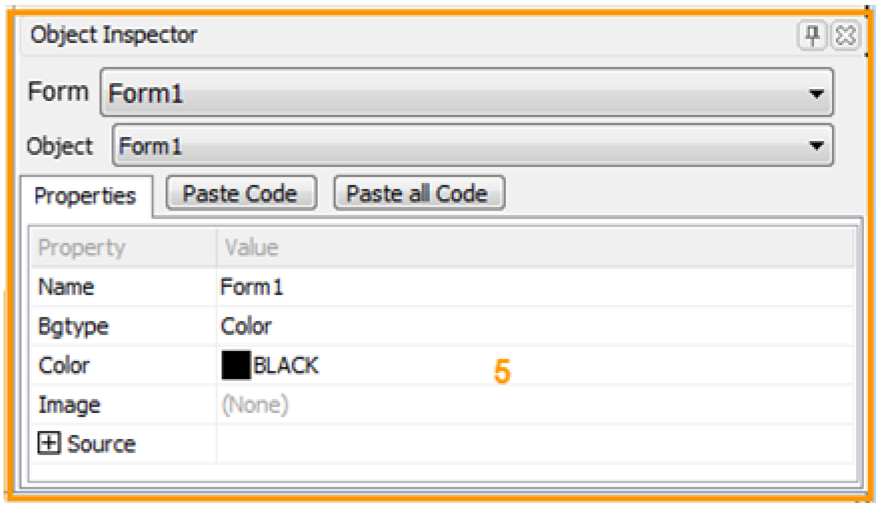
The form represents a What-You-See-Is-What-You-Get (WYSIWYG) screen.



The active form is displayed there, with its objects. Objects are picked from the panes and can be resized and moved.

Click on an object to select it.

7.5. Area 5: Object Inspector


Object Inspector

Form Form1

Object Form1

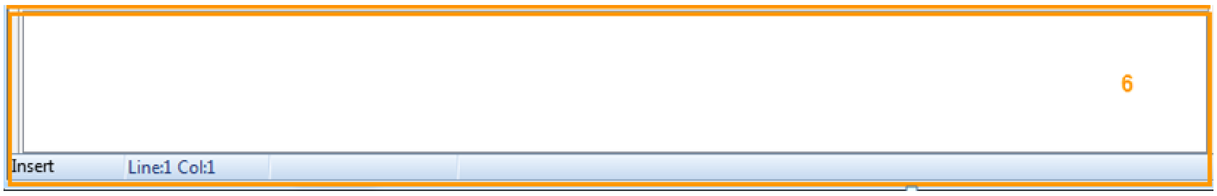
Property	Value
Name	Form1
Bgtype	Color
Color	■ BLACK
Image	(None)
⊞ Source	

The object inspector provides all the information on the selected object:

- properties, as size and position;
- and events, where actions are defined.

7.6. Area 6: Message Window

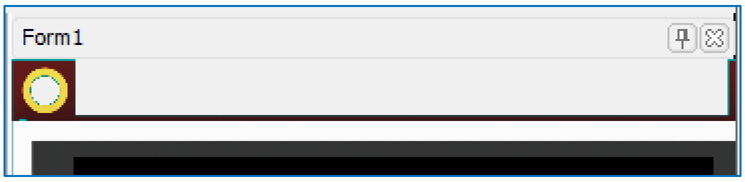
The message window displays errors, warnings and notices after the project is built.



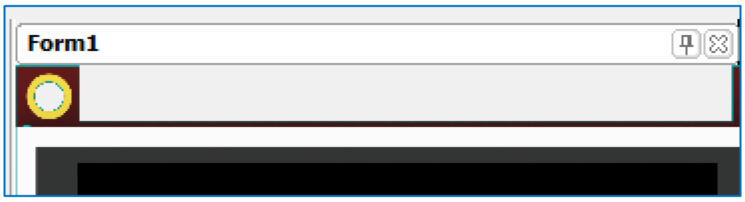
Before starting using the Workshop 4, we need to connect the screen and prepare a micro-SD card.

7.7. Move the Form and Object Inspector

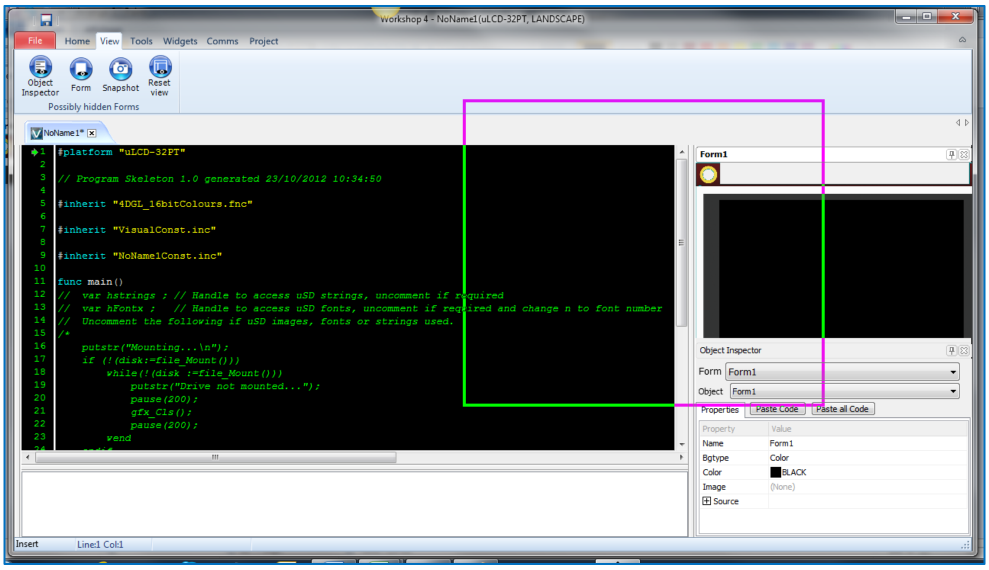
The Form and Object Inspector can be moved for a customised interface.
Click on the title bar of the form…



…to activate it…



…then click-and-drag to place the form on the desire place…

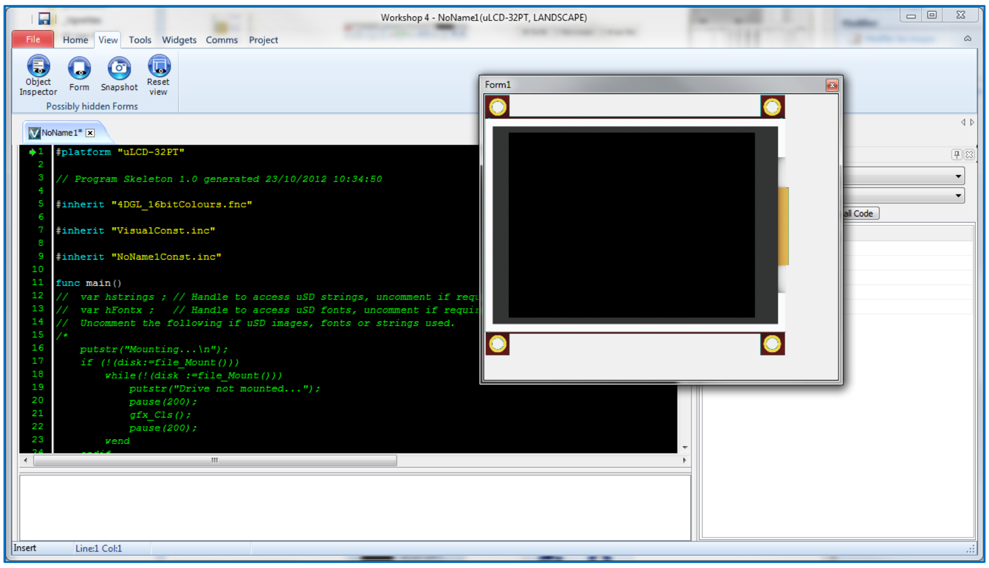


```
#platform "uLCD-32PT"

// Program Skeleton 1.0 generated 23/10/2012 10:34:50

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName1Const.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required and change n to font number
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
```

Form: Form1
Object: Form1

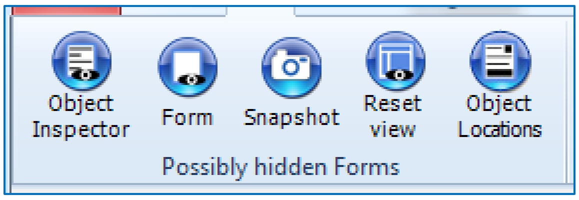
Property	Value
Name	Form1
Bgtype	Color
Color	■ BLACK
Image	(None)
⊞ Source	

…and finally release the button.



```
#platform "uLCD-32PT"

// Program Skeleton 1.0 generated 23/10/2012 10:34:50

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName1Const.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if requ
//  var hFontx ;   // Handle to access uSD fonts, uncomment if requi
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
```

In case the Form and Object Inspector are hidden, just click on the relevant button on the **View** menu to display them again:



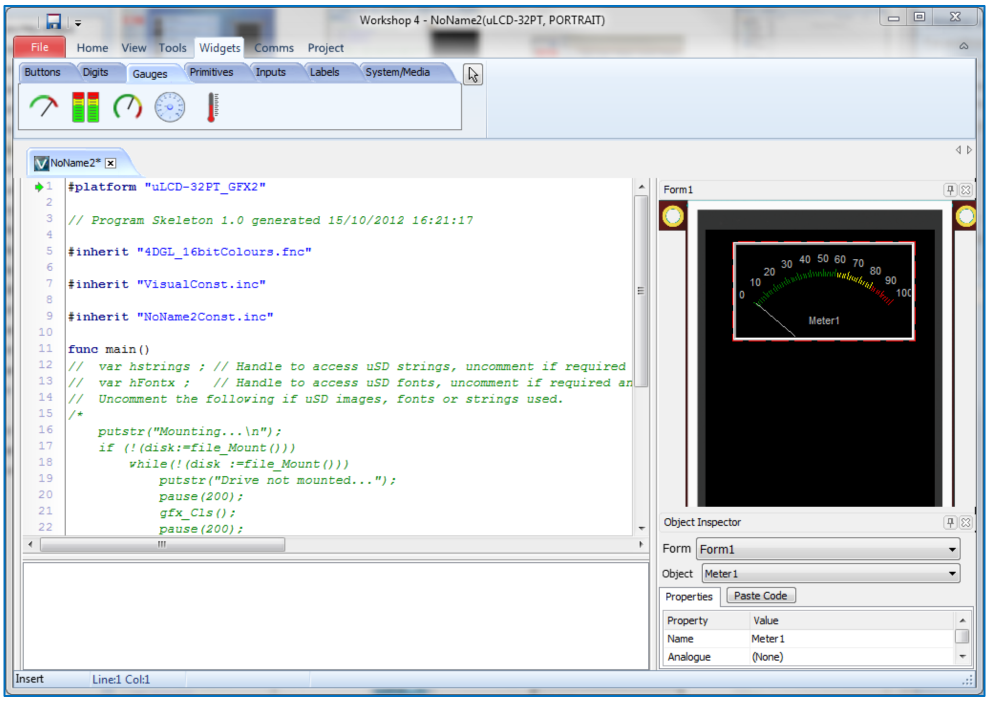
8. A First ViSi Project

8.1. Adding Graphical Objects

Become familiar with the 4D-ViSi layout by cycling through the tabs in the **Widgets** menu. 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.



```
#platform "uLCD-32PT_GFX2"

// Program Skeleton 1.0 generated 15/10/2012 16:21:17

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName2Const.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required an
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
```

Form Form1
Object Meter1

Properties | Paste Code

Property	Value
Name	Meter1
Analogue	(None)

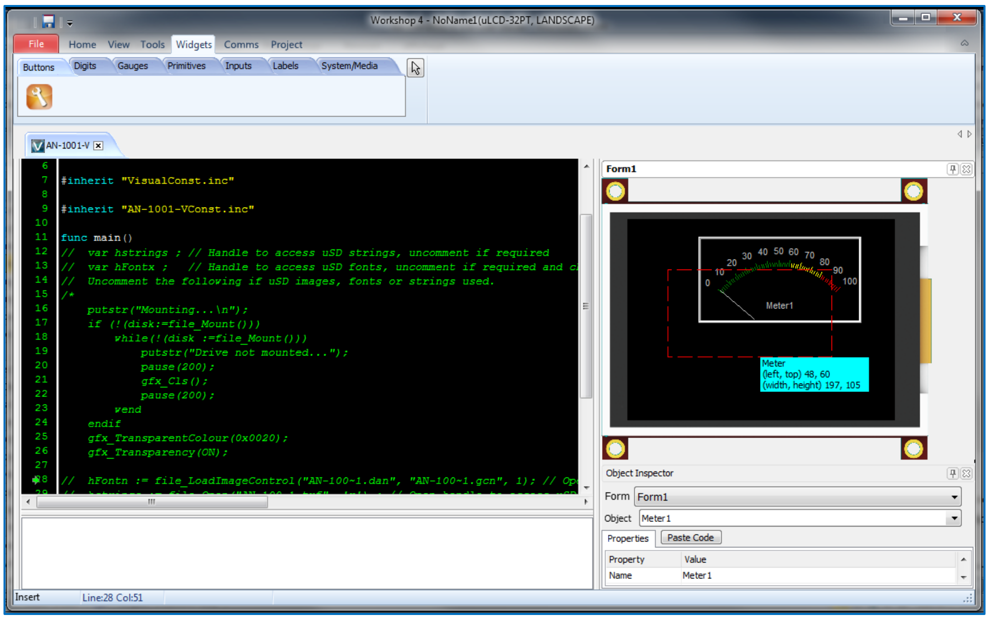
8.2. Customizing Graphical Objects

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,… in the property fields below.

Refer to the next image, which demonstrates this concept.



```

#inherit "VisualConst.inc"

#inherit "AN-1001-VConst.inc"

func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required and c
//  Uncomment the following if uSD images, fonts or strings used.
/*
    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    gfx_TransparentColour(0x0020);
    gfx_Transparency(ON);

//  hFontn := file_LoadImageControl("AN-100~1.dan", "AN-100~1.gcn", 1); // Op
```

Meter
(left, top) 48, 60
(width, height) 197, 105

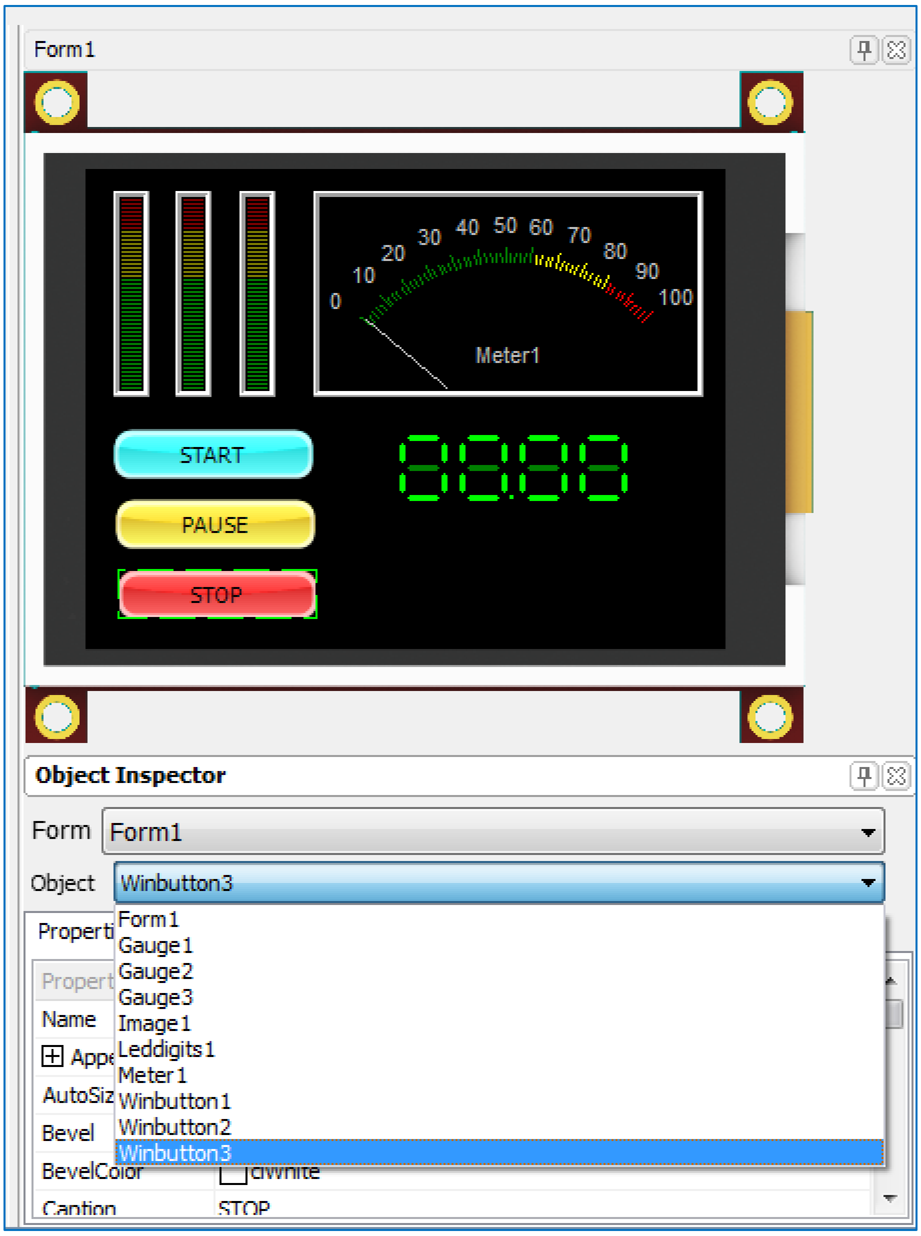
Form: Form1

Object: Meter1

Property	Value
Name	Meter1

8.3. 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.



Form Form1

Object Winbutton3

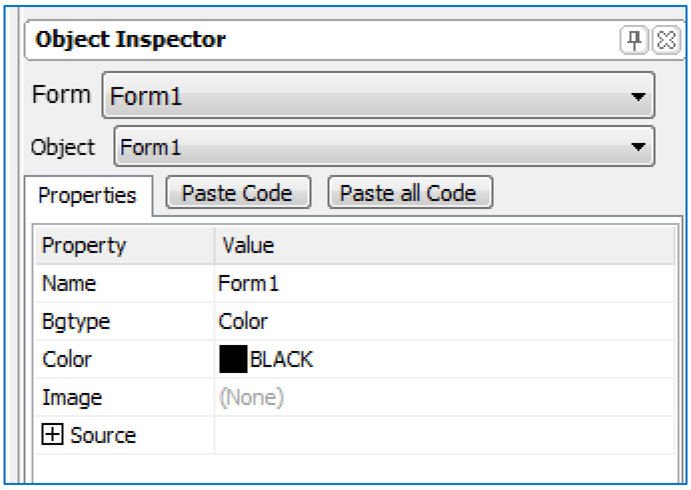
Form1
Gauge1
Gauge2
Gauge3
Image1
Leddigits1
Meter1
Winbutton1
Winbutton2
Winbutton3

Properti
Propert
Name
Appe
AutoSiz
Bevel
BevelColor clWhite
Caption STOP

8.4. 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 ViSi.

There a two possible options for inserting code into the application.


Object Inspector

Form: Form1
Object: Form1

Properties | Paste Code | Paste all Code

Property	Value
Name	Form1
Bgtype	Color
Color	■BLACK
Image	(None)
⊞ Source	

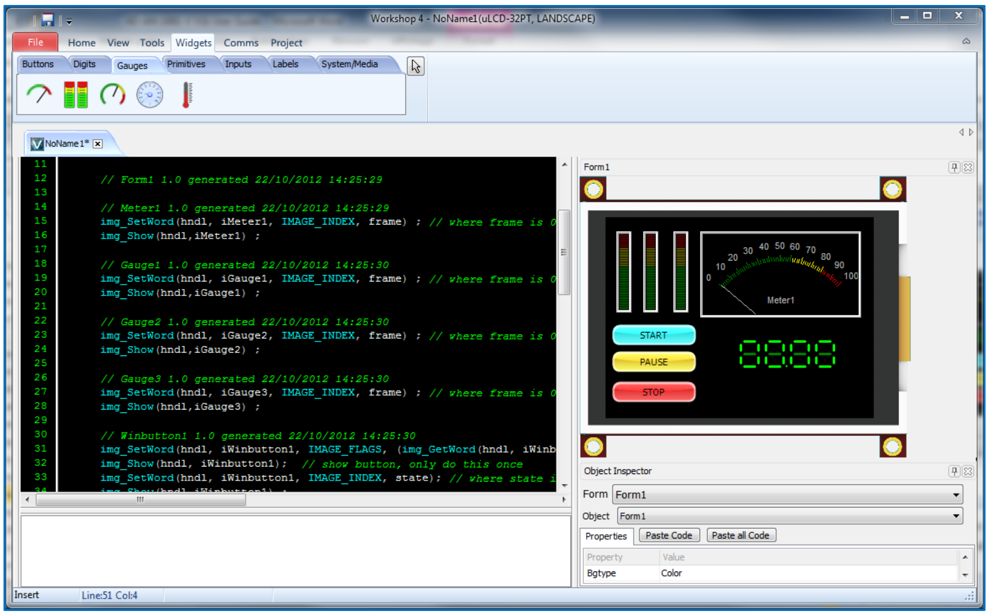
The user can:

- either paste for an individual object;
- or paste all the code in one place for all of the objects created on the screen.

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:



```
// Form1 1.0 generated 22/10/2012 14:25:29

// Meter1 1.0 generated 22/10/2012 14:25:29
img_SetWord(hndl, iMeter1, IMAGE_INDEX, frame) ; // where frame is 0
img_Show(hndl,iMeter1) ;

// Gauge1 1.0 generated 22/10/2012 14:25:30
img_SetWord(hndl, iGauge1, IMAGE_INDEX, frame) ; // where frame is 0
img_Show(hndl,iGauge1) ;

// Gauge2 1.0 generated 22/10/2012 14:25:30
img_SetWord(hndl, iGauge2, IMAGE_INDEX, frame) ; // where frame is 0
img_Show(hndl,iGauge2) ;

// Gauge3 1.0 generated 22/10/2012 14:25:30
img_SetWord(hndl, iGauge3, IMAGE_INDEX, frame) ; // where frame is 0
img_Show(hndl,iGauge3) ;

// Winbutton1 1.0 generated 22/10/2012 14:25:30
img_SetWord(hndl, iWinbutton1, IMAGE_FLAGS, (img_GetWord(hndl, iWinb
img_Show(hndl, iWinbutton1);  // show button, only do this once
img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, state); // where state i
```

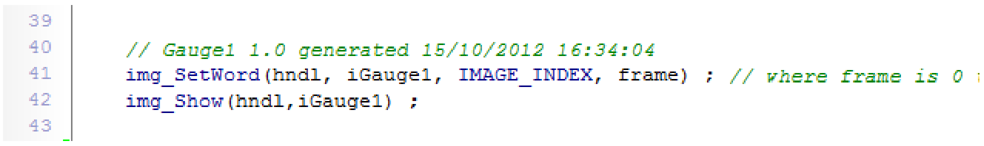
Form: Form1
Object: Form1

Properties | Paste Code | Paste all Code

Property	Value
Bgtype	Color

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 –here, **Gauge1**– then click on **Paste Code**. Notice that only a couple of lines of code are inserted into the program.



```
// Gauge1 1.0 generated 15/10/2012 16:34:04
img_SetWord(hndl, iGauge1, IMAGE_INDEX, frame) ; // where frame is 0
img_Show(hndl,iGauge1) ;
```

8.5. 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.

8.6. 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 card 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 card 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.

8.7. 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.

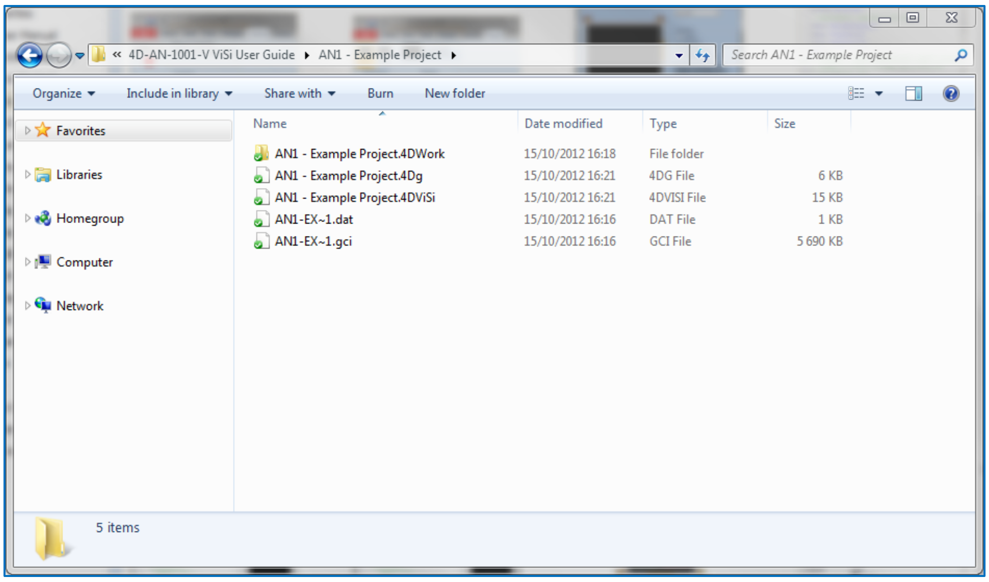
The screen exhibits three gauges, a counter and a meter. All three graphic tools are controlled by the same index and hence all reflect the same value between 1 and 100.

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.



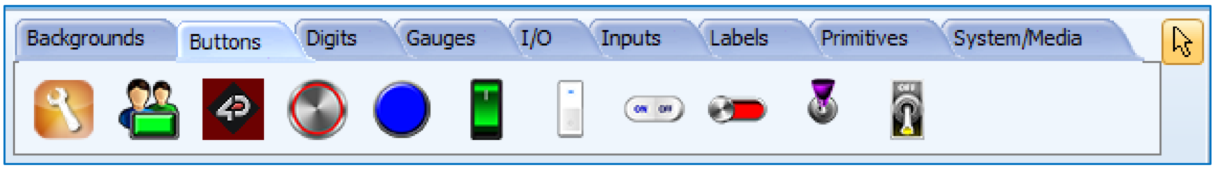
Name	Date modified	Type	Size
AN1 - Example Project.4DWork	15/10/2012 16:18	File folder	
AN1 - Example Project.4Dg	15/10/2012 16:21	4DG File	6 KB
AN1 - Example Project.4DViSi	15/10/2012 16:21	4DVISI File	15 KB
AN1-EX~1.dat	15/10/2012 16:16	DAT File	1 KB
AN1-EX~1.gci	15/10/2012 16:16	GCI File	5 690 KB

9. Objects

ViSi provides an easy method of designing complex Graphics User Interface applications. The user drags and drops objects on his simulated screen and the code is updated accordingly.

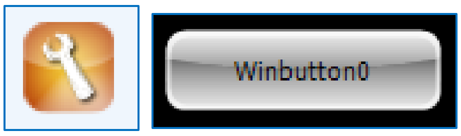
Each object is presented with its button on the left and an example on the right when used on a form.

9.1. Buttons Object

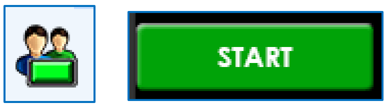


The Buttons pane contains the following widgets:

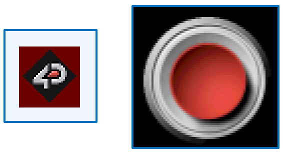
- Win Button



- User Button

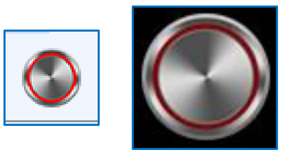


- Animated Button

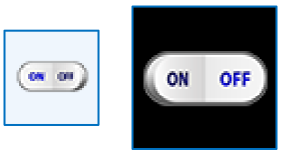


- 4D Buttons

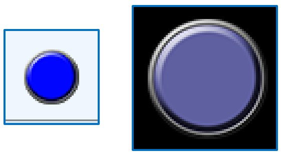
Button01



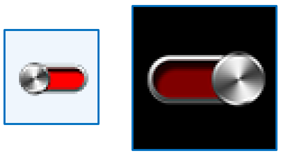
Rocker03



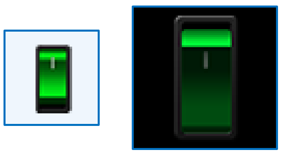
Button02



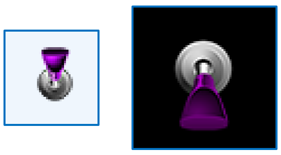
Slider01



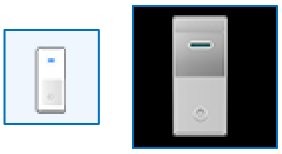
Rocker01



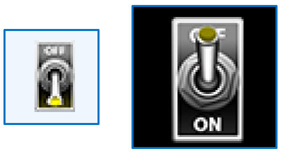
Toggle01



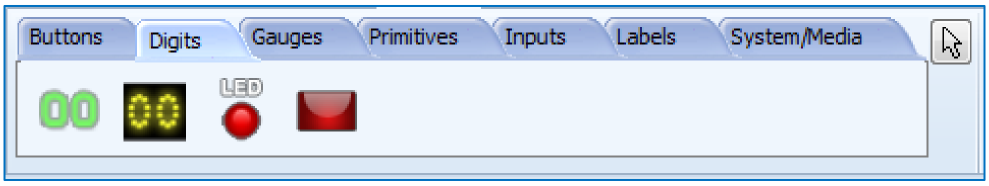
Rocker02



Toggle02

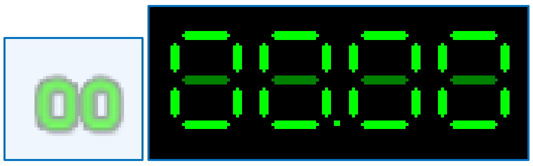


9.2. Digits Objects

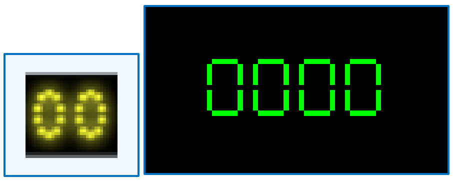


The Digits pane contains 4 different displays.

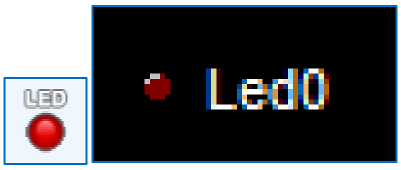
- LED Digits



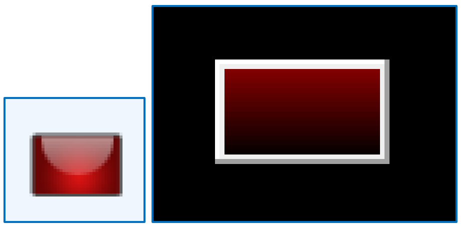
- Custom Digits



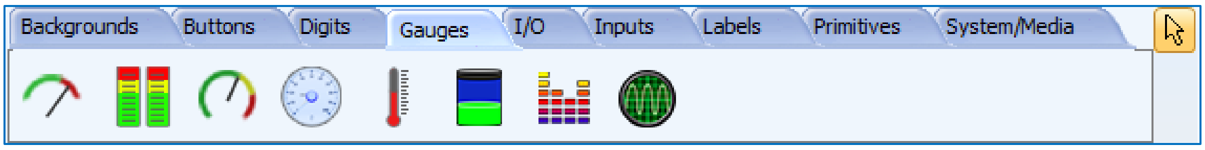
- LED



- User LED

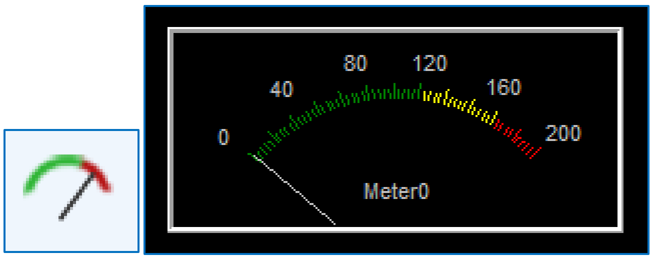


9.3. Gauges Objects

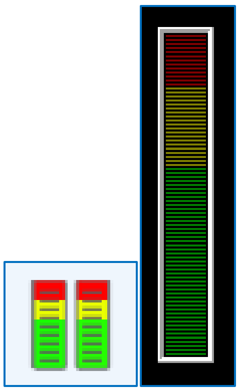


The Gauges pane contains five specialised displays:

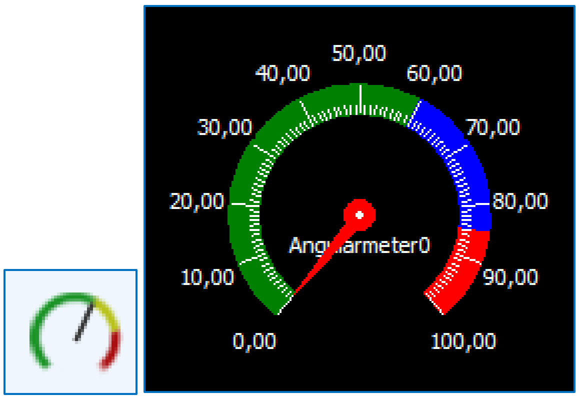
- Meter



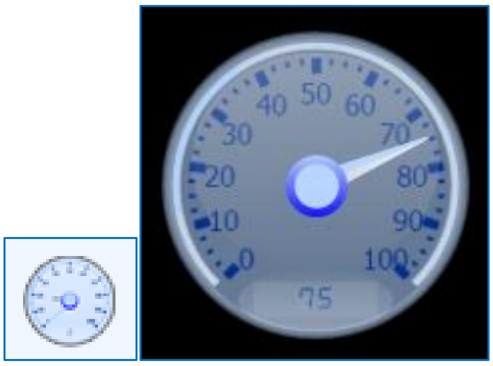
- Gauge



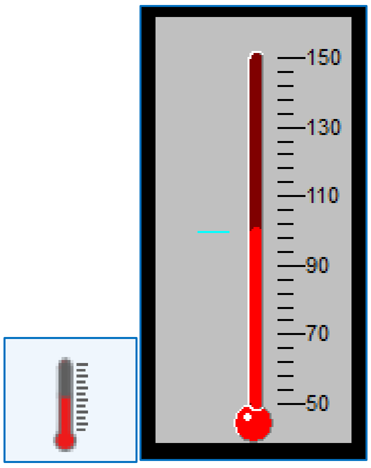
- Angular Meter



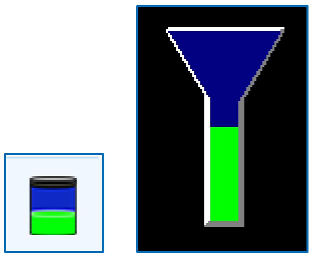
- Cool Gauge



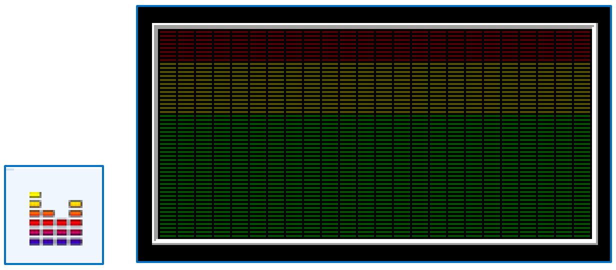
- Thermometer



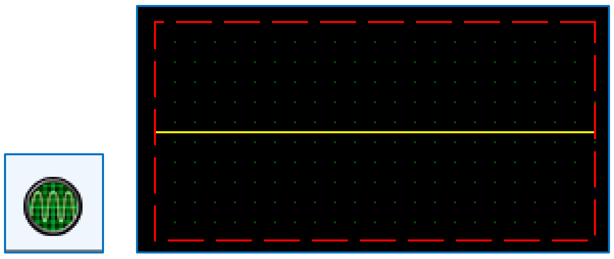
- Tank



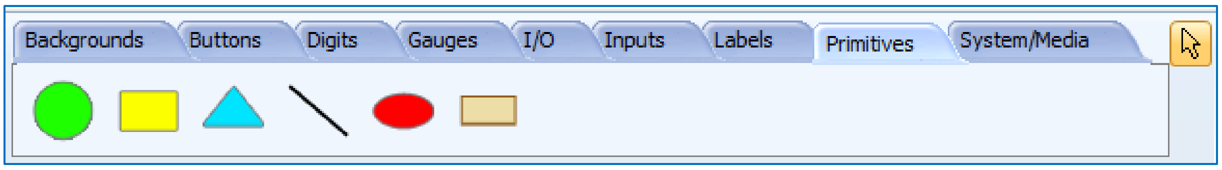
- Spectrum



- Scope

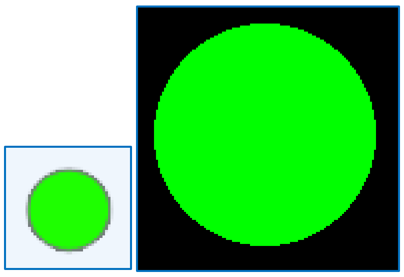


9.4. Primitives Objects

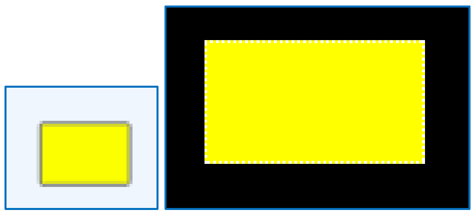


The Primitives pane offers 6 standard static drawings and 2 dynamic objects.

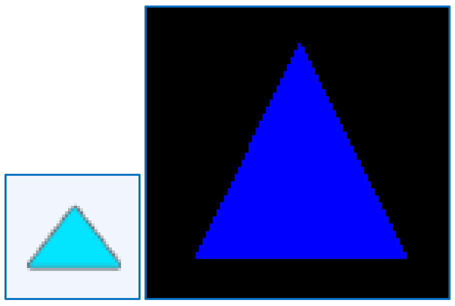
- Circle



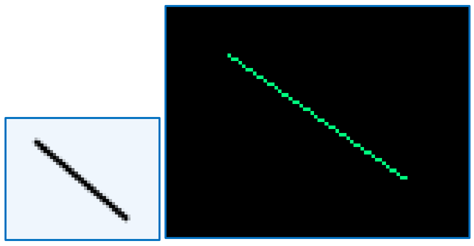
- Rectangle



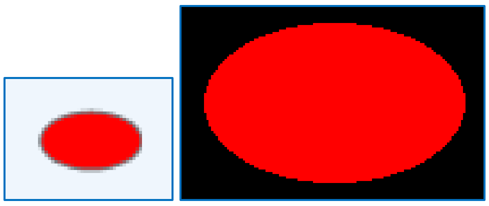
- Triangle



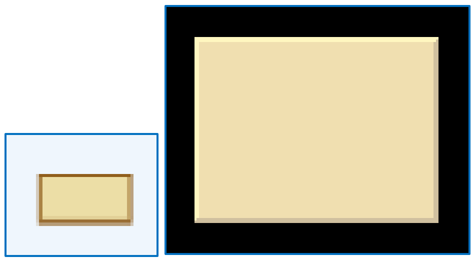
- Line



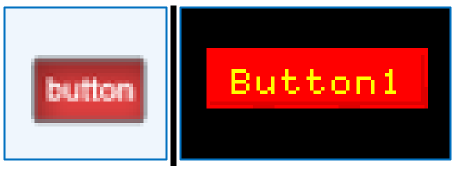
- Ellipse



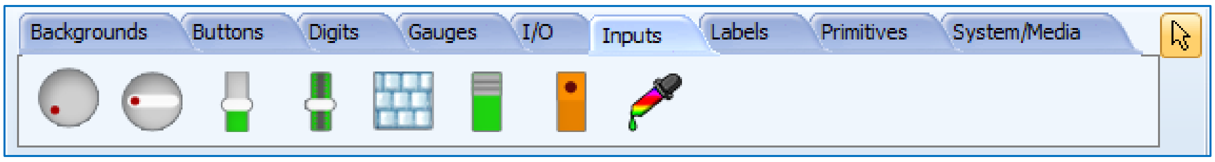
- Panel



- Button

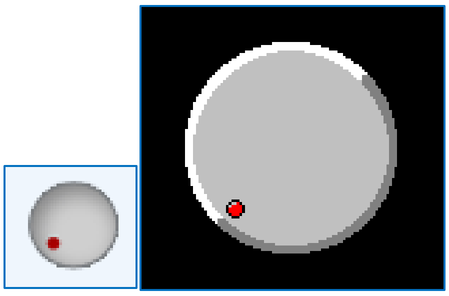


9.5. Inputs Objects



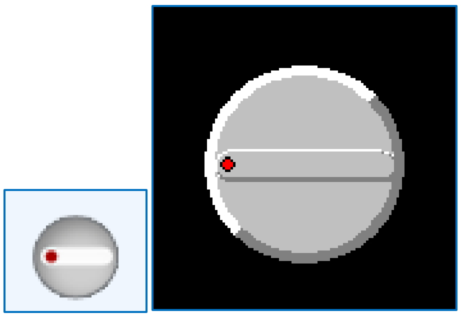
The Inputs pane contains rotary selectors, linear selectors, keyboards and switches.

- Knob



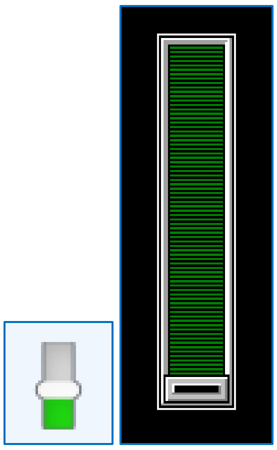
The minimum and maximum angles, the back and the handle can be customised.

- Rotary Switch



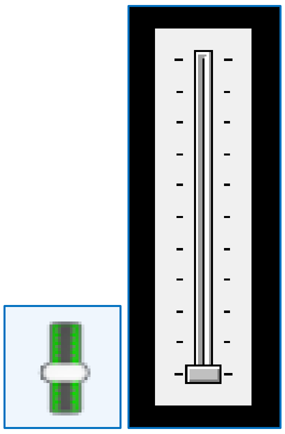
The minimum and maximum angles, the positions and labels, the switch and the winch colours can be customised.

- Slider



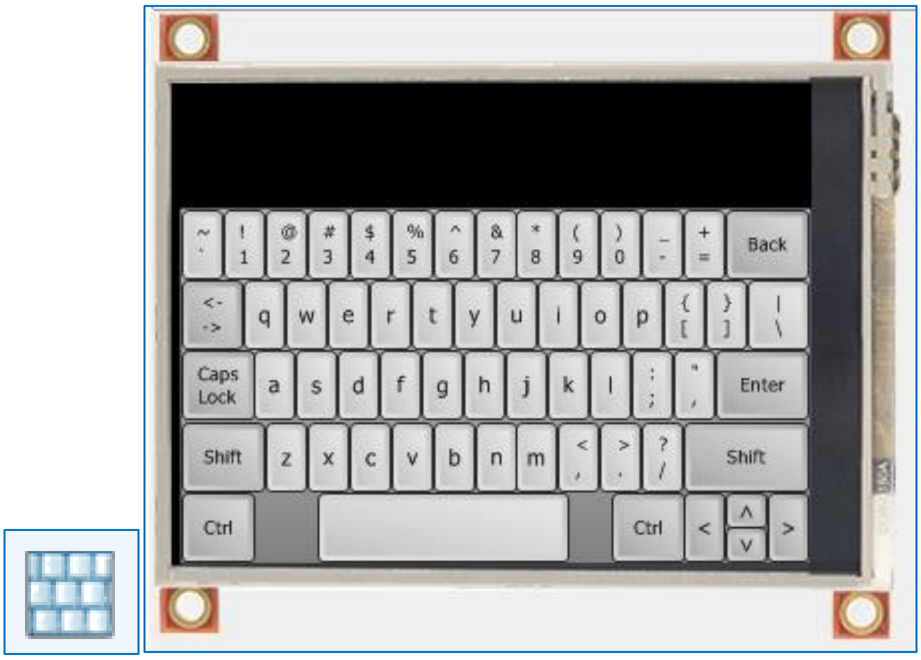
The minimum and maximum values, the vertical or horizontal orientations, the colours can be customised.

- Track-bar



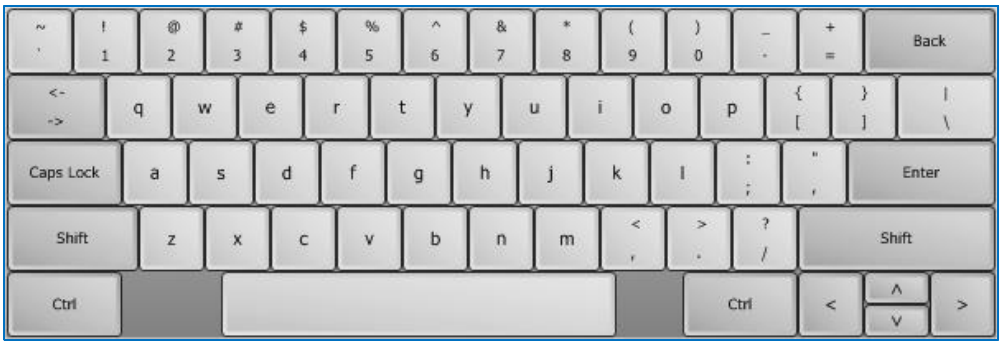
The minimum and maximum values, the vertical or horizontal orientations, the frequency and ticks, the colours can be customised.

- Keyboard

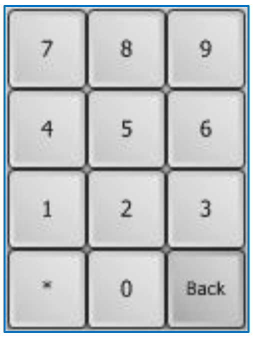


ViSi-Genie comes with various defined keyboards:

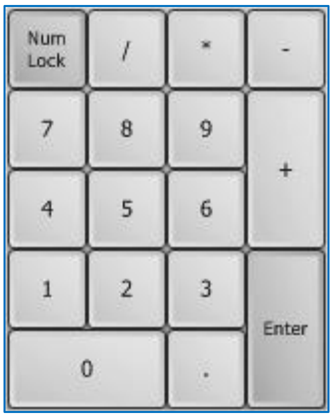
- QWERTY keyboard, by default,



- Cell-phone keyboard

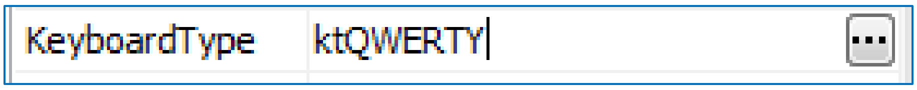


- Numeric keyboard



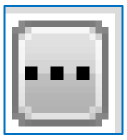
- And even a customised keyboard.

The different keyboards are selected by clicking on the **KeyboardType** property:

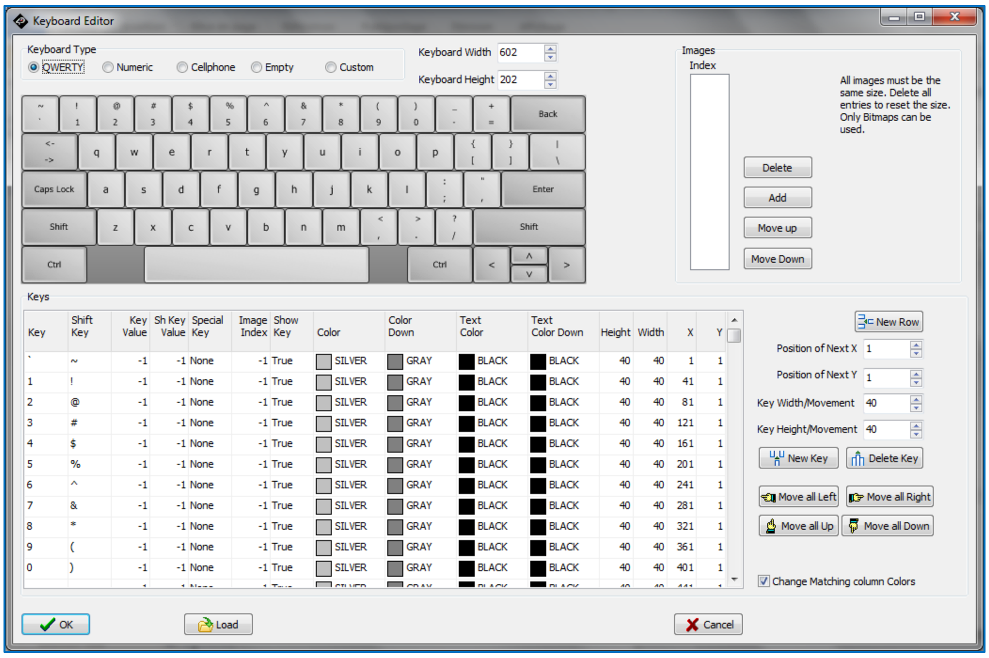


KeyboardType ktQWERTY

Click on the button



to launch the Keyboard Editor:


Keyboard Editor

Keyboard Type
[✓ QWERTY] Numeric Cellphone Empty Custom

Keyboard Width 602

Keyboard Height 202

Images
Index

All images must be the same size. Delete all entries to reset the size. Only Bitmaps can be used.

Keys

Key	Shift Key	Key Value	Sh Key Value	Special Key	Image Index	Show Key	Color	Color Down	Text Color	Text Color Down	Height	Width	X	Y
`	~	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	1	1
1	!	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	41	1
2	@	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	81	1
3	#	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	121	1
4	$	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	161	1
5	%	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	201	1
6	^	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	241	1
7	&	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	281	1
8	*	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	321	1
9	(	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	361	1
0	)	-1	-1	None	-1	True	SILVER	GRAY	BLACK	BLACK	40	40	401	1

Position of Next X 1

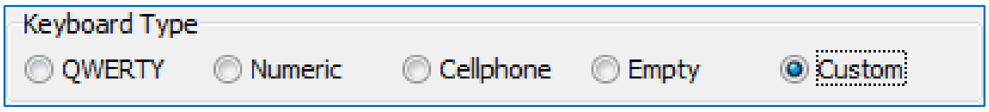
Position of Next Y 1

Key Width/Movement 40

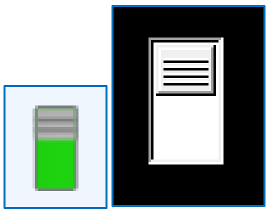
Key Height/Movement 40

[✓] Change Matching column Colors

The Keyboard Editor allows to select and customise the keyboard:

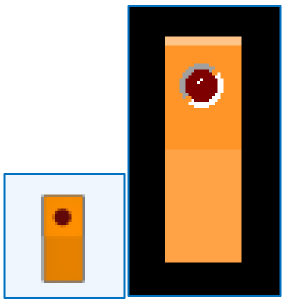

Keyboard Type
QWERTY Numeric Cellphone Empty [✓ Custom]

- DIP Switch



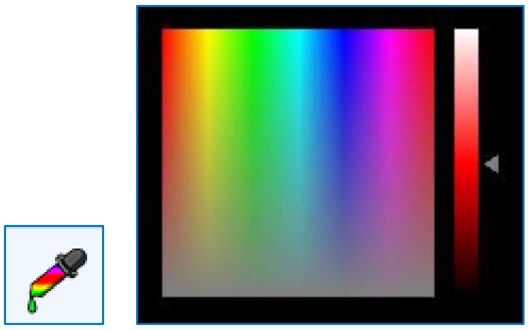
The number of positions of the switch can be specified, 2 as shown or more.

- Rocker Switch

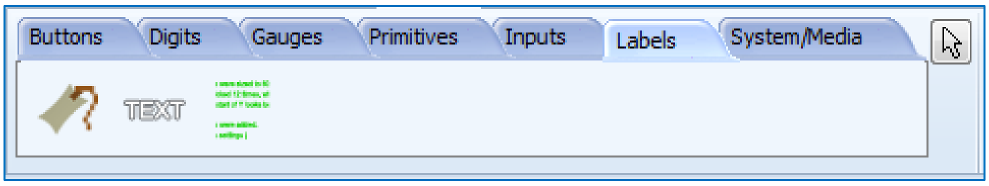


When on, the red LED is turned on.

- Color Picker

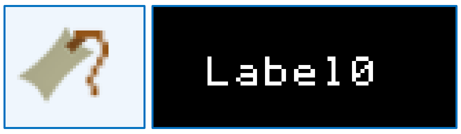


9.6. Labels Objects

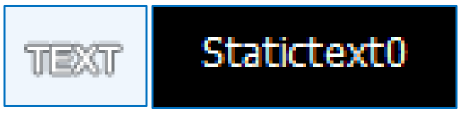


The Labels pane offers three different objects to display text.

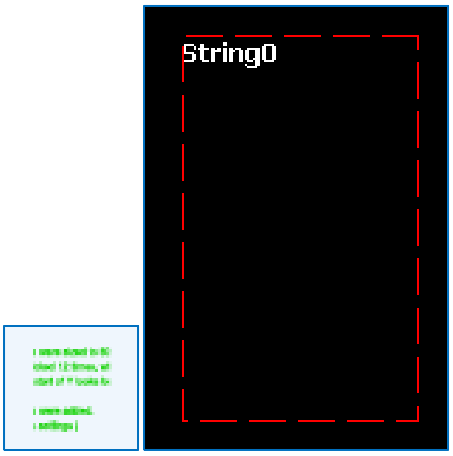
- Label



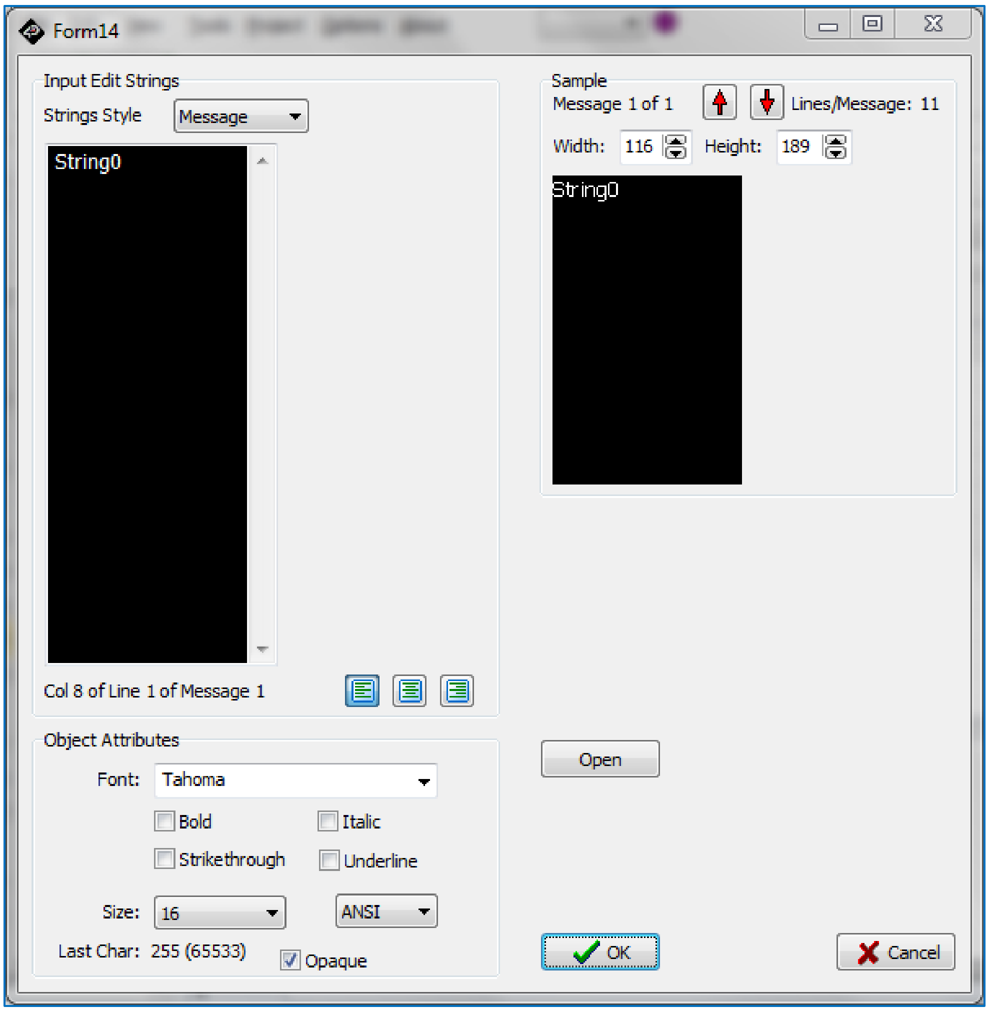
- Static Text



- Strings

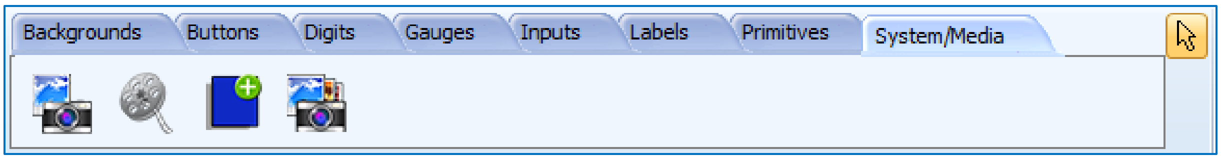


The text is defined by:



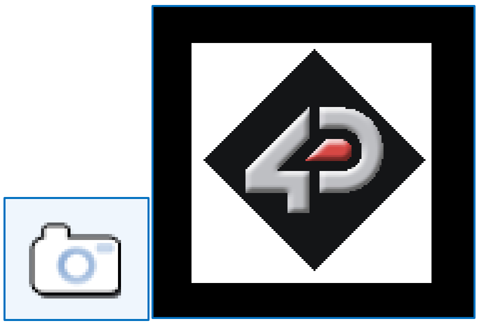
Font, size, ANSI or Unicode can be defined.

9.7. System/Media Objects

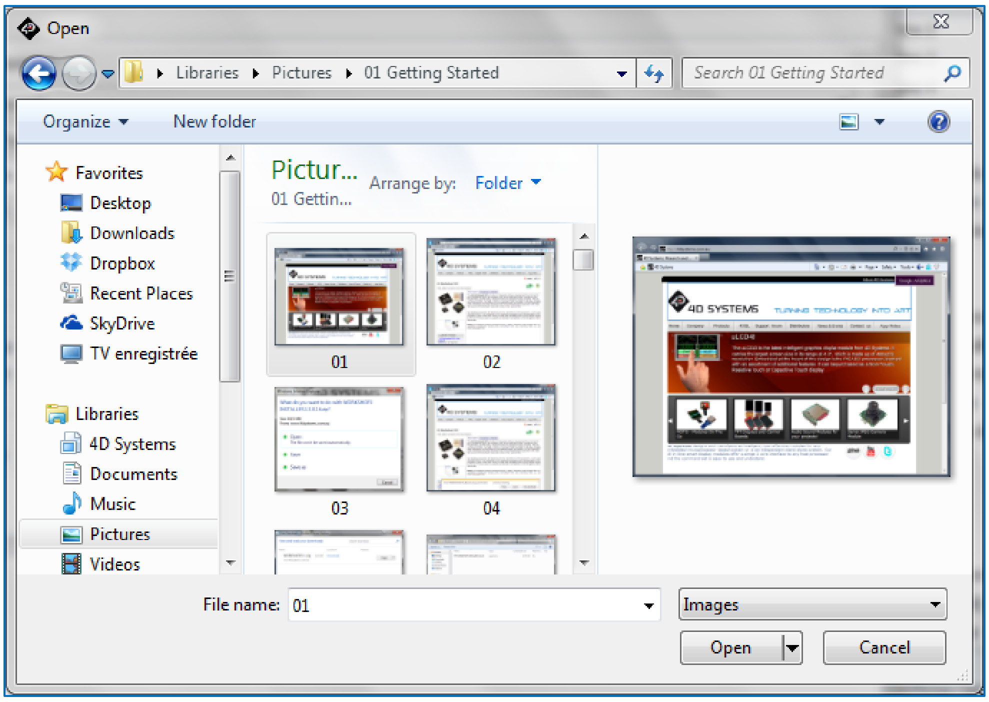


The System pane includes the image, video, form, and user images objects.

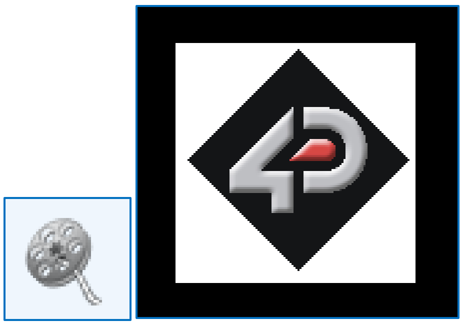
- Image



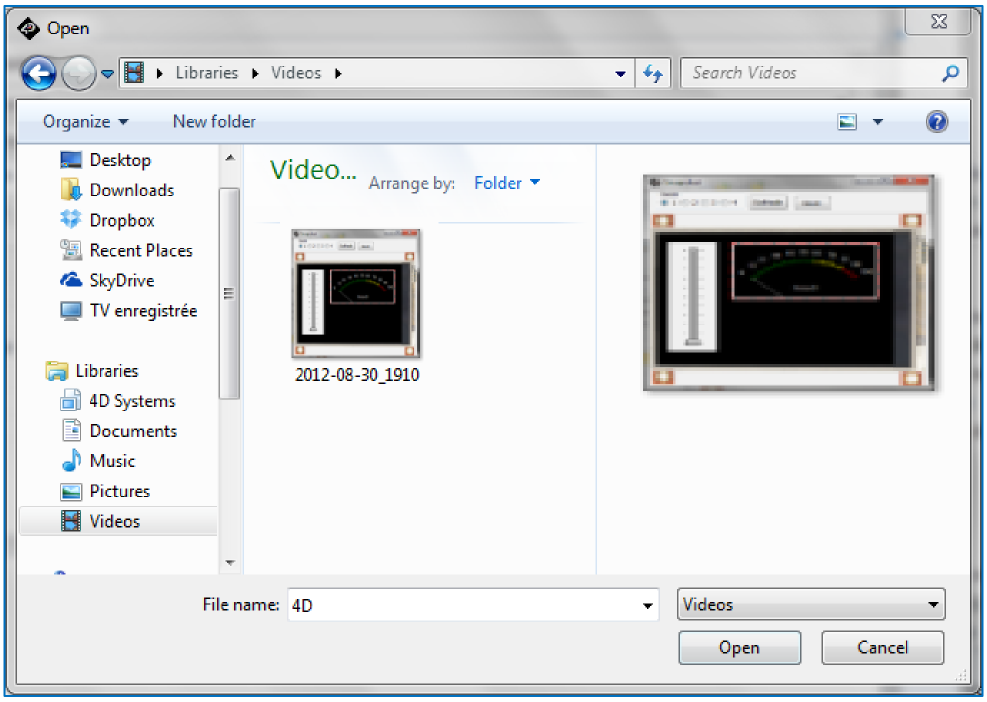
The image is selected through an Open window:



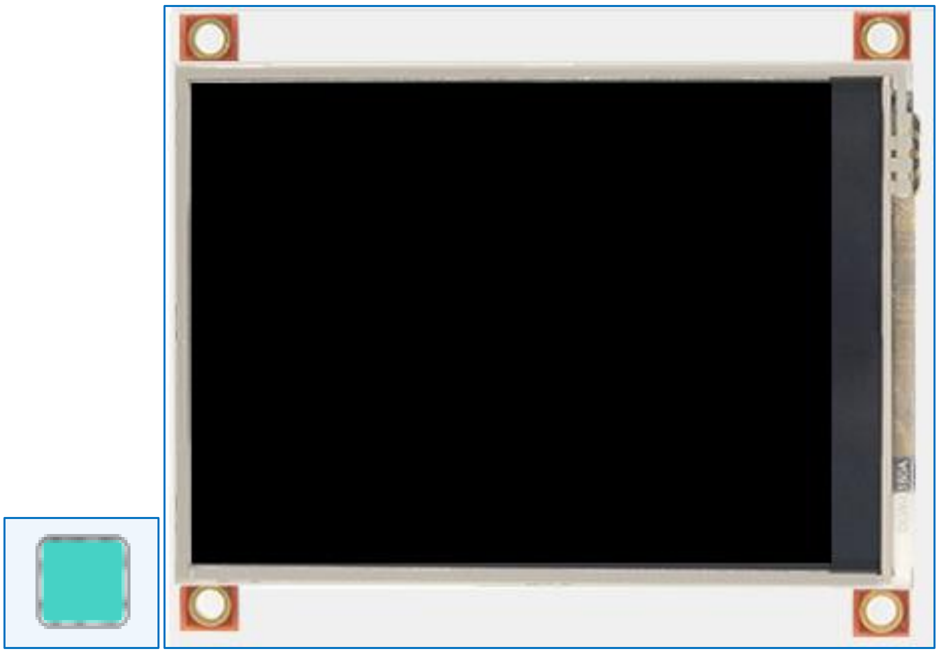
- Video



The video is selected through an Open window:



- Form



The Form creates a new empty form and adds it to the project.

- User Images

The user images object represents an easy way to build a slideshow by joining together a sequence of images in one place. The images are selected through the Image List editor window.

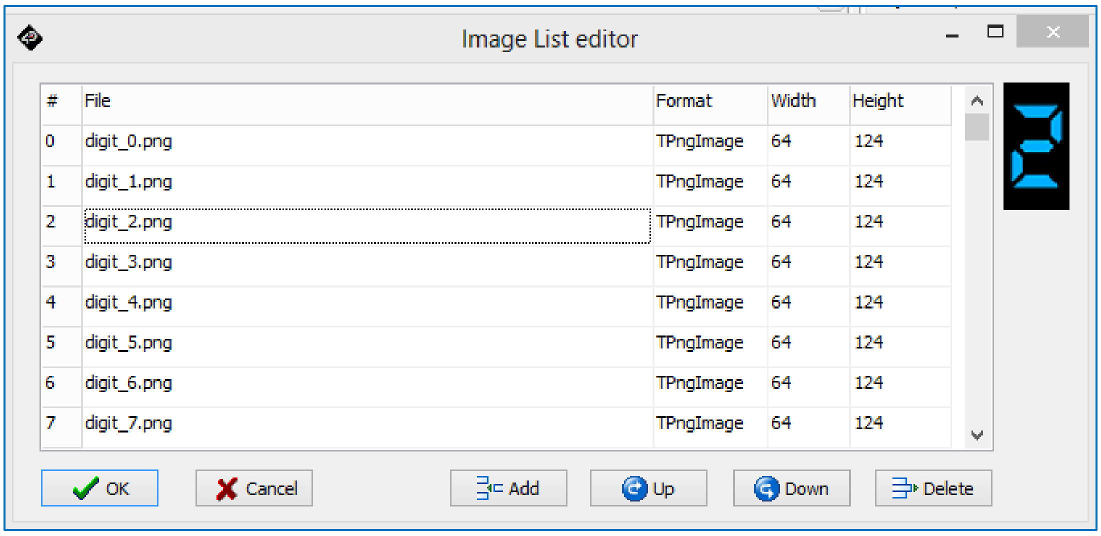
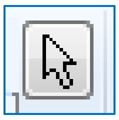


Image List editor

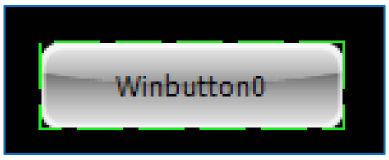
#	File	Format	Width	Height
0	digit_0.png	TPngImage	64	124
1	digit_1.png	TPngImage	64	124
2	digit_2.png	TPngImage	64	124
3	digit_3.png	TPngImage	64	124
4	digit_4.png	TPngImage	64	124
5	digit_5.png	TPngImage	64	124
6	digit_6.png	TPngImage	64	124
7	digit_7.png	TPngImage	64	124

9.8. Selection Tool

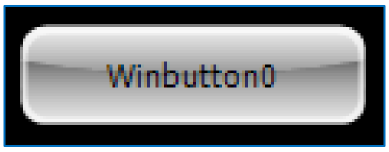


The arrow is used to deselect and object.

To select an object, just click on it: green or red dotted lines appear.

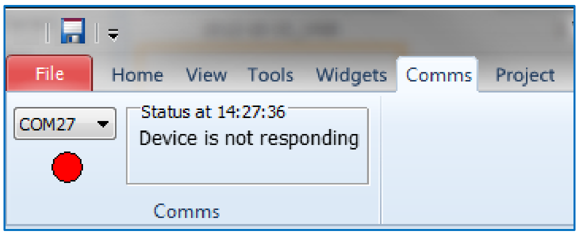


To deselect an object, just click again: the dotted lines disappear.



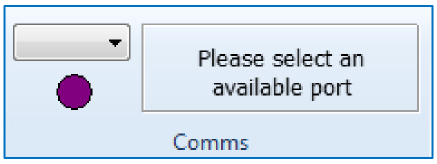
10. Connect the Screen

Connect the screen to a USB port with the 4D Systems programming cable and select the **Comms** menu:



Status at 14:27:36
Device is not responding

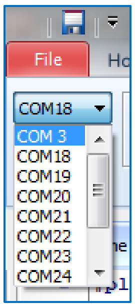
Above the **Comms** section, the **violet** light mentions no screen is currently connected.



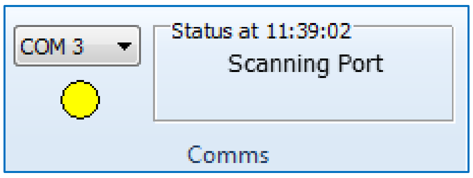
Please select an available port

Connect the 4D Systems to the screen and plug the cable into the USB port.

Click on the drop-down list and select the COM port, here COM3.

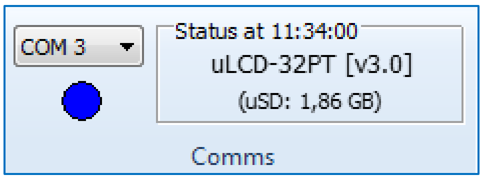


The light turns yellow while the connection is being established:



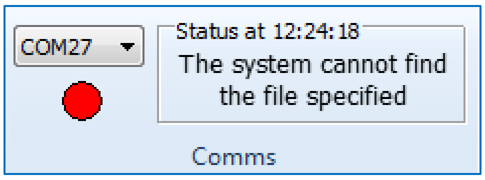
Status at 11:39:02
Scanning Port

Finally, the light goes **blue** when the connection is established.



Status at 11:34:00
uLCD-32PT [v3.0]
(uSD: 1,86 GB)

The light turns **red** when no screen is attached to the selected port:



Status at 12:24:18
The system cannot find the file specified

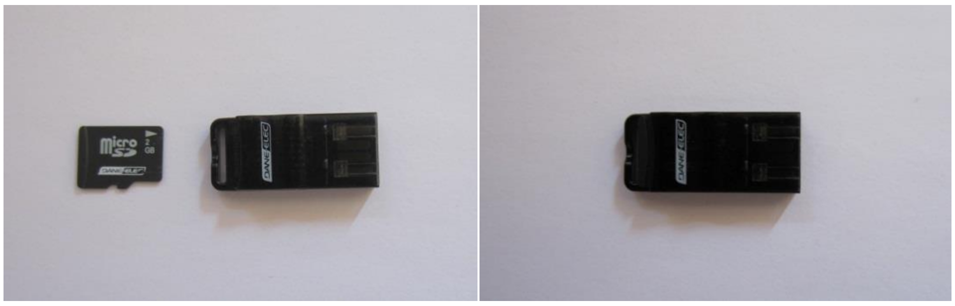
11. Insert the Micro-SD Card

The Picaso modules, micro-SD card shall be FAT16-formatted. Partition can't exceed 4 GB.

For Goldelox modules, the micro-SD card shall not be formatted at all, it requires the SD card to be RAW.

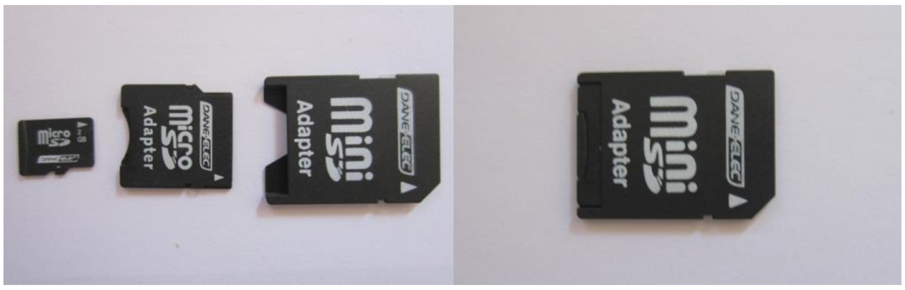
To connect the micro-SD card, either

- Insert the micro-SD card into the USB adaptor and plug the USB adaptor into an USB port of the PC.

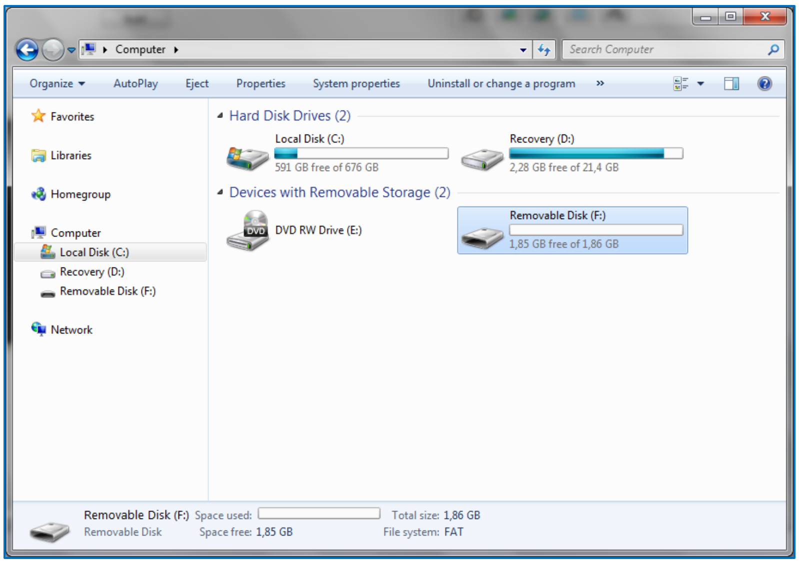


Or

- Insert the micro-SD card into a micro-SD to SD card converter and plug the SD card converter into the SD card slot of the PC.



Check the micro-SD card is mounted, here as drive E:.



Computer

Organize AutoPlay Eject Properties System properties Uninstall or change a program

Hard Disk Drives (2)

Local Disk (C:) 591 GB free of 676 GB

Recovery (D:) 2,28 GB free of 21,4 GB

Devices with Removable Storage (2)

DVD RW Drive (E:)

Removable Disk (F:) 1,85 GB free of 1,86 GB

Removable Disk (F:) Space used: Total size: 1,86 GB

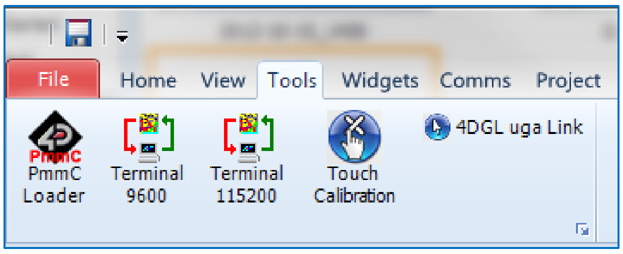
Removable Disk Space free: 1,85 GB File system: FAT

For Goldelox, if prompted to format the SD card, click no/cancel. Leave the card unformatted and Workshop4 will handle the rest.

12. Communication Terminal

An alternative to the debugger is the Terminal.

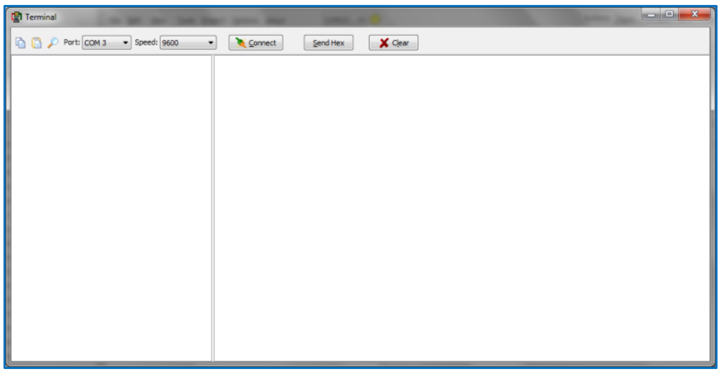
To launch the Terminal, select the **Tools** menu…



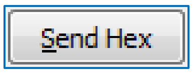
…and

- Click '**Terminal connect 9600**' to open the currently selected com port at 9600 baud in the Terminal program.
- Click '**Terminal connect 115200**' to open the currently selected com port at 115200 baud in the Terminal program.

A new screen appears:



To send the commands on hexadecimal format, press

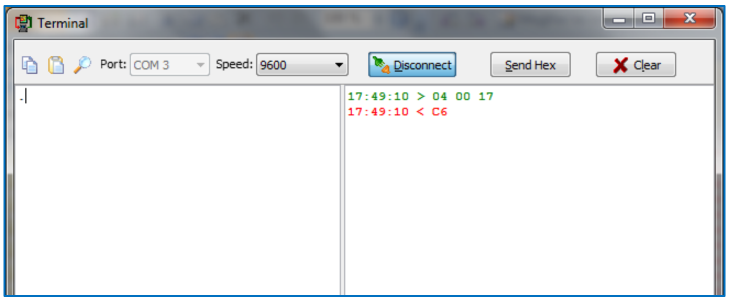


The commands sent by the host and the messages sent by the screen are the same as with the **Genie Test Executor** debugger.

The white area on the right displays

- In **green** the messages sent to the screen;
- And in **red** the messages received from the screen:

Here, the command *Set Slider0 to value 0x17* is sent, or **04 00 17** displayed in green on the terminal window.



```
17:49:10 > 04 00 17
17:49:10 < C6
```

```
#platform "uLCD-32PT"
#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);
         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
   endif

   forever
endfunc
```

13. Application Notes

For a more detailed presentation of the objects with examples, please refer to the corresponding application notes:

Reference	Content
4D-AN-P3002	Adding and Configuring a DIP Switch
4D-AN-P3003	GPIO Bus Control Using DIP Switch Objects
4D-AN-P3004	Adding and Configuring Fancy Buttons
4D-AN-P3005	Adding Image and Video
4D-AN-P4001	Getting Started - First Project with ViSi-Genie
4D-AN-P4002	ViSi-Genie - onChanging and onChanged Events
4D-AN-P4003	ViSi-Genie - Customised Keyboard
4D-AN-P4004	ViSi-Genie - Advanced Buttons
4D-AN-P4005	ViSi-Genie - Show Image
4D-AN-P4006	ViSi-Genie - Play Sound
4D-AN-P4007	ViSi-Genie - Play Video
4D-AN-P4008	ViSi-Genie - Gauges
4D-AN-P4009	ViSi-Genie - Inputs
4D-AN-P4010	ViSi-Genie - Connection to an Arduino Host with Red-Green-Blue LED Control
4D-AN-P4011	ViSi-Genie - Using Combined Objects
4D-AN-P4012	ViSi-Genie - Digital Displays
4D-AN-P4013	ViSi-Genie - Labels, Texts and Strings
4D-AN-D4002	ViSi-Genie - Tank
4D-AN-D4003	ViSi-Genie - Spectrum
4D-AN-D4004	ViSi-Genie - Single Trace Scope
4D-AN-D4005	ViSi-Genie - Color Picker
4D-AN-D4006	ViSi-Genie - User Button
4D-AN-D4007	ViSi-Genie - Animated Button
4D-AN-D4008	ViSi-Genie - 4D Buttons
4D-AN-D4009	ViSi-Genie - Pin Input and Output

For an exhaustive reference on ViSi-Genie objects, please refer to the ViSi-Genie Reference Manual.

14. Revision History

Revision	Revision Content	Revision Date
1.0	First Release	Nov 16, 2012
1.1	Amended microSD detail	July 7, 2013
1.2	Added the new widgets	Sept 23, 2013

15. Legal Notice

16. Contact Information

For Technical Support: support@4dsystems.com.au

For Sales Support: sales@4dsystems.com.au

Website: www.4dsystems.com.au