



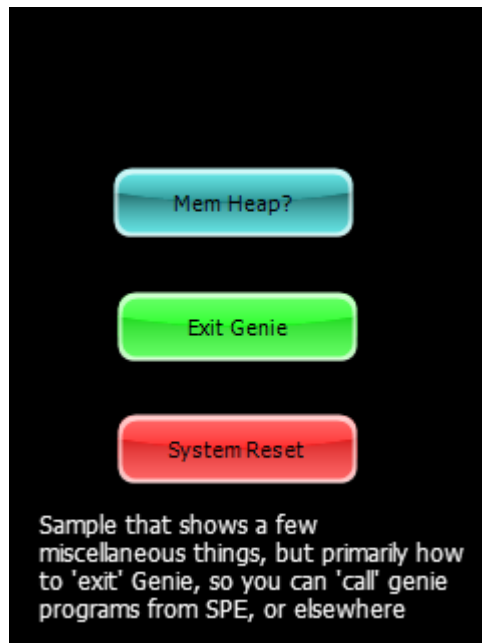
ViSi-Genie Magic Exit Genie

DOCUMENT DATE: **21th MAY 2019**
DOCUMENT REVISION: **1.1**



Description

This application note shows how to use the **Magic Release** object. The Magic Release object is under the Genie Magic pane in Workshop 4 Pro. It contains a 4DGL code that is executed every time a touch release action occurs. Below is a screenshot image of the project for this document.



Note 1: The ViSi-Genie project for this application is found in Worskhop. Go to the File menu -> Samples -> ViSi Genie Magic (Picaso/Diablo16) -> **ExitGenie.4DGenie**.

Note 2: **Worskhop Pro** is needed for this application. Before getting started, the following are required:

- Any of the following 4D Picaso and gen4 Picaso display modules:

[gen4-uLCD-24PT](#) [gen4-uLCD-28PT](#) [gen4-uLCD-32PT](#)
[uLCD-24PTU](#) [uLCD-32PTU](#) [uVGA-III](#)

and other superseded modules which support the ViSi Genie environment

- The target module can also be a Diablo16 display

[gen4-uLCD-24D series](#) [gen4-uLCD-28D series](#) [gen4-uLCD-32D series](#)
[gen4-uLCD-35D series](#) [gen4-uLCD-43D series](#) [gen4-uLCD-50D series](#)
[gen4-uLCD-70D series](#)
[uLCD-35DT](#) [uLCD-43D series](#) [uLCD-70DT](#)

Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor. The display module used in this application note is the uLCD-32PTU, which is a Picaso display. This application note is applicable to Diablo16 display modules as well.

- [4D Programming Cable](#) / [uUSB-PA5/uUSB-PA5-II](#) for non-gen4 displays(uLCD-xxx)
- [4D Programming Cable](#) & [gen4-PA](#) / [gen4-IB](#) / [4D-UPA](#) for gen4 displays (gen4-uLCD-xxx)
- [micro-SD \(μSD\)](#) memory card
- [Workshop 4 IDE](#) (installed according to the installation document)
- When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or

has a working knowledge of the topics presented in these recommended application notes.

Content

Description	2
Content	3
Application Overview	4
Setup Procedure	4
Create a New Project	5
<i>Create a New Project</i>	5
Design the Project	5
<i>Add Three Winbutton Objects to Form0</i>	5
Select the Object Icon	5
Click on the WYSIWYG Screen	5
Change the Caption of Winbutton0	6
Change the Color of Winbutton0	6
<i>Add a Static Text Object to Form0</i>	7
Select the Object Icon	7
Click on the WYSIWYG Screen	7
Change the Caption for Statictext0	8
<i>Add a Magic Release Object to Form0</i>	8
<i>Understanding the 4DGL Code for MagicRelease</i>	8
Working Model	8
Check the Object on which a TOUCH_RELEASED has Occurred	8
Print the Available Memory	9
Exit Genie	9

Reset the Display	9
<i>Use MagicRelease to Load a Child Program</i>	9
Copy the 4XE File to the uSD Card	10
Function for Loading a Child Program	10
Expected Output: Picaso Display	11
Expected Output: Diablo16 Display	11
Generate the 4XE File for the SPE Application	12
Build and Upload the Project	12
The uLCD-32PTU and/or the uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.	12
Proprietary Information	13
Disclaimer of Warranties & Limitation of Liability	13

Application Overview

Certain applications may need the execution of another program besides the Genie program during runtime. Hence, a means for “exiting” the Genie program and loading another program is needed. In this application, the **Magic Release** object is used to demonstrate how to exit the Genie program and load another program (a child program) from the uSD card during runtime. The SPE application is used as the child program.

Setup Procedure

For instructions on how to launch Workshop 4, how to open a ViSi-Genie project, and how to change the target display, kindly refer to the section “**Setup Procedure**” of the application note:

[ViSi Genie Getting Started – First Project for Picaso Displays](#) (for Picaso)

or

[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16).

Create a New Project

Create a New Project

For instructions on how to create a new ViSi-Genie project, please refer to the section “**Create a New Project**” of the application note

[ViSi Genie Getting Started – First Project for Picaso Displays](#) (for Picaso)

or

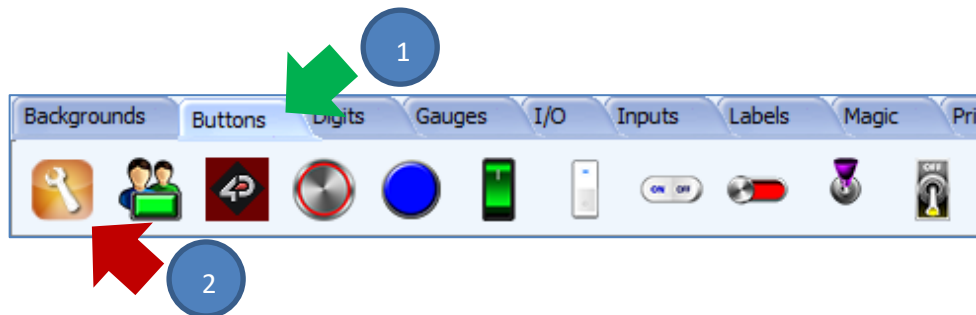
[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16)

Design the Project

A uLCD-32PTU (portrait orientation) will be used for this application note.

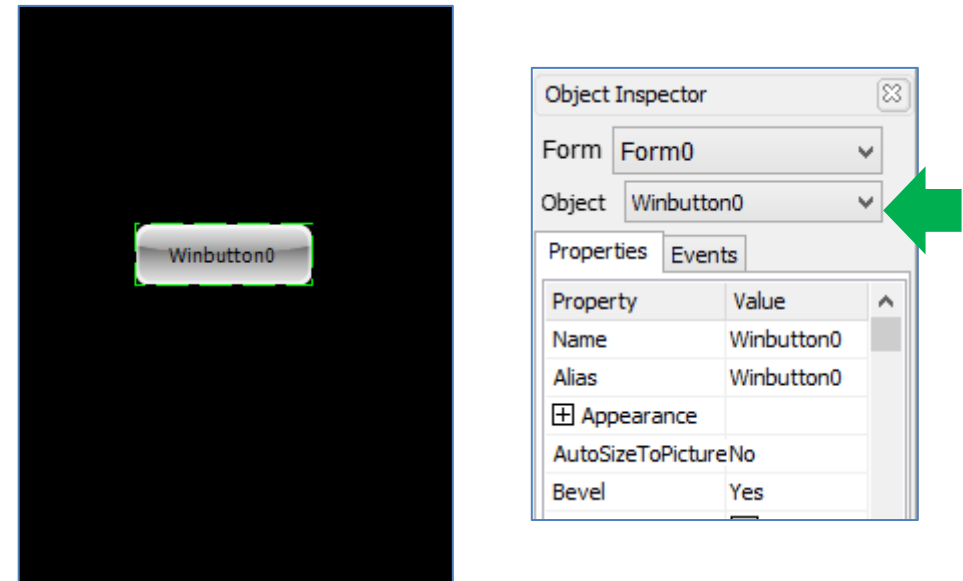
Add Three Winbutton Objects to Form0

Select the Object Icon



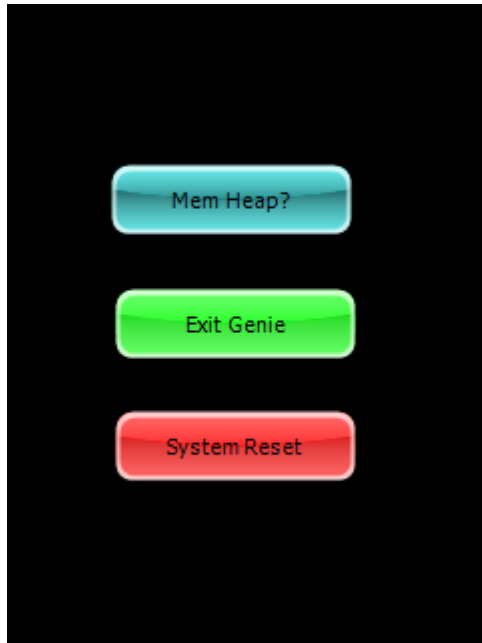
Click on the WYSIWYG Screen

Upon clicking on the WYSIWYG screen, a winbutton object is created. This is **Winbutton0**.



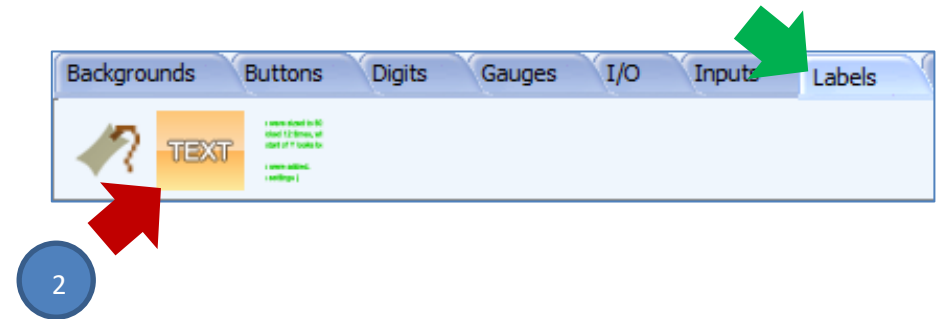
Many of the properties are self-explanatory. The user is encouraged to experiment with the property values.

Follow the same procedure to create the two other buttons. When, done the WYSIWYG screen should look like as shown below.



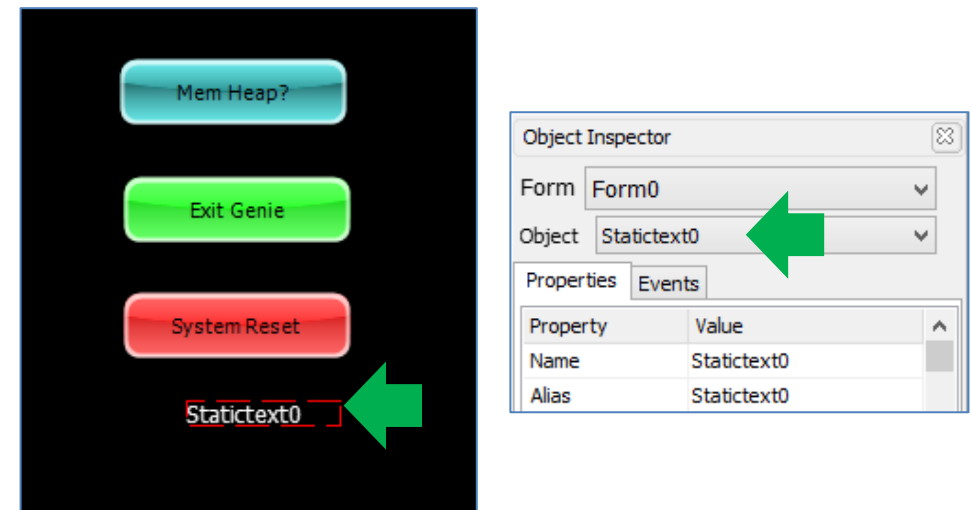
Add a Static Text Object to Form0

Select the Object Icon

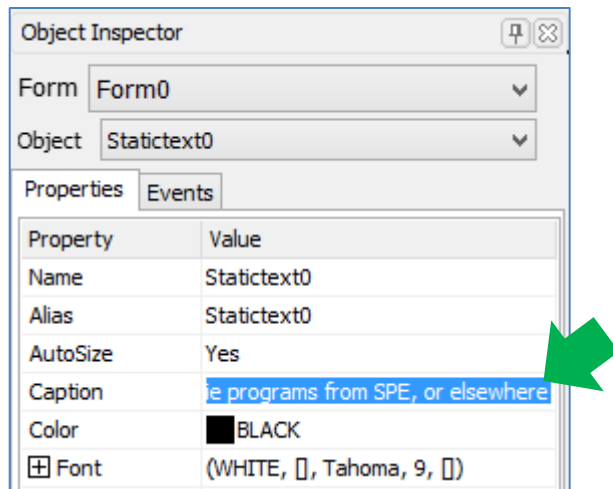


Click on the WYSIWYG Screen

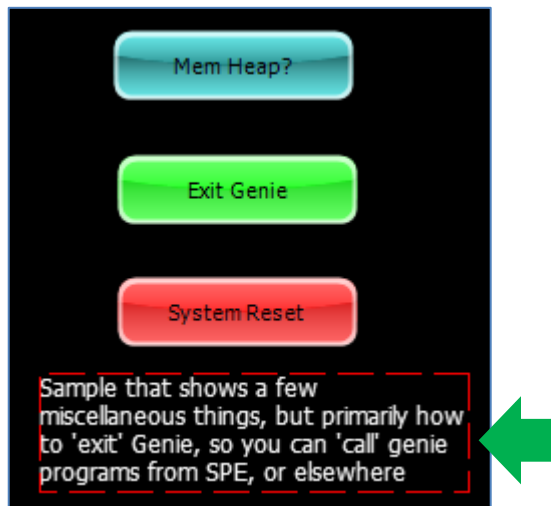
Click on the WYSIWYG screen to place the static text object. This is **Statictext0**.



Change the Caption for Statictext0



Statictext0 is updated accordingly.



Add a Magic Release Object to Form0

See the application note [ViSi-Genie How to Add Magic Objects](#) for the procedures for adding a Magic Release object and opening the code for it.

Understanding the 4DGL Code for MagicRelease

Working Model

To facilitate this discussion, the model below is used.

1. Genie constantly waits for touch actions.
2. Genie stores the index of the object on which a touch action occurred into the variable "ImageTouched".
3. Genie determines the type of the touch action. There are three types of touch actions:
 - a. TOUCH_PRESSED
 - b. TOUCH_RELEASED
 - c. TOUCH_MOVING

The names of the constants define themselves. If the touch action is a "TOUCH_RELEASED", Genie calls on the routine inside the object **MagicRelease**.

4. At this point, Genie normally goes back to step 1, unless **MagicRelease** says otherwise.

Check the Object on which a TOUCH_RELEASED has Occurred

To know the index of the object on which a TOUCH_RELEASED action has occurred, we have to compare the value of the variable **ImageTouched** against the indices of the objects on the form. Hence, we write:


```

13  if (ImageTouched == iWinbutton0)
14      gfx_MoveTo(0,0) ;
15      print(mem_Heap()) ;
16  else if (ImageTouched == iWinbutton1)
17      return ;
18  else if (ImageTouched == iWinbutton2)
19      SystemReset() ;
20  endif

```

Note how the index of an object is associated with its name.

Name of object	Index of object
Winbutton0	iWinbutton0
Winbutton1	iWinbutton1
Winbutton2	iWinbutton2

Print the Available Memory

If a **TOUCH_RELEASED** action has occurred on **Winbutton0**, the cursor is moved to the origin and the byte size of the largest chunk of memory available in the heap is printed.

```

13  if (ImageTouched == iWinbutton0)
14      gfx_MoveTo(0,0) ;
15      print(mem_Heap()) ;

```

For more information on the two 4DGL functions above, refer to the Picaso or Diablo16 Internal Functions Reference Manual. **Right-click** on the 4DGL function name text and choose “**Context Sensitive help**” to open the reference manuals.

Exit Genie

If a **TOUCH_RELEASED** action has occurred on **Winbutton1**, the Genie program is “closed” and nothing happens after.

```

16  else if (ImageTouched == iWinbutton1)
17      return ;

```

Reset the Display

If a **TOUCH_RELEASED** action has occurred on **Winbutton2**, the program restarts.

```

18  else if (ImageTouched == iWinbutton2)
19      SystemReset() ;

```

SystemReset() resets and restarts the program. It is the equivalent of a ‘cold boot’ (i.e. a total hardware reset). There is a two-second delay before the program restarts. This is due to the EVE boot procedure time.

Use MagicRelease to Load a Child Program

There are 4DGL functions for loading a program from the uSD card. The program to be loaded is called the child program. The program that does the loading is the parent program. Before a child program can be loaded, its 4XE file must first be on the uSD card.

Copy the 4XE File to the uSD Card

Attached to this application note is the folder “uSD card file”. The folder contains two files:

Name	Type	Date modified	Size
spe211D.4XE	4XE File	4/12/2015 8:27 PM	5 KB
spe213P.4XE	4XE File	4/12/2015 8:06 PM	4 KB

These are 4XE files of the SPE application.

Spe211D.4XE is the 4XE file for the SPE application (SPE2 revision 1.1) compiled for a uLCD-35DT, which is a Diablo16 display.

Spe213P.4XE is the 4XE file for the SPE application (SPE2 revision 1.3) compiled for a uLCD-32PTU, which is a Picaso display.

The filenames will be hardcoded in **MagicRelease**. Choose which filename to include in the code and which file to copy to the uSD card according to your target display module. To learn how to generate the 4XE file for the SPE application, refer to the section “**Generate the 4XE File for the SPE Application**”.

Function for Loading a Child Program

Shown below is a modified version of **MagicRelease**.

```

13  if (ImageTouched == iWinbutton0)
14      gfx_MoveTo(0,0) ;
15      print(mem_Heap()) ;
16  else if (ImageTouched == iWinbutton1)
17      gfx_Cls() ;
18      // send a message to the host
19      file_Exec("spe213P.4XE", 0) ; // for
20      //file_Exec("spe211D.4XE", 0) ; // for
21  else if (ImageTouched == iWinbutton2)
22      SystemReset() ;
23  endif

```

Choose according to your display. Here the target is a Picaso display module.

```

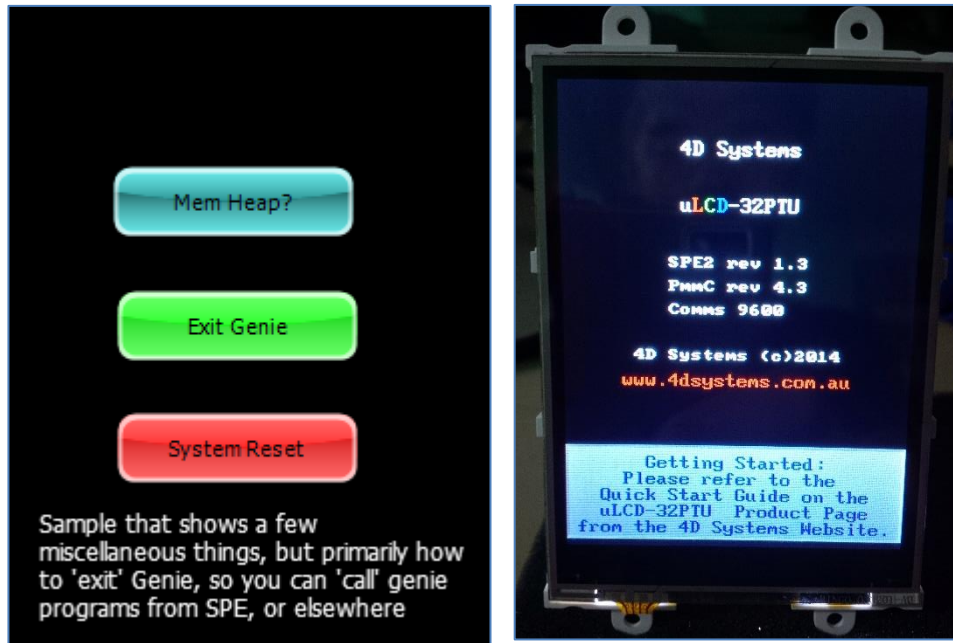
19  file_Exec("spe213P.4XE", 0) ; // for

```

Note: When creating a filename for 4XE files, follow the 8.3 filename format.

Expected Output: Picaso Display

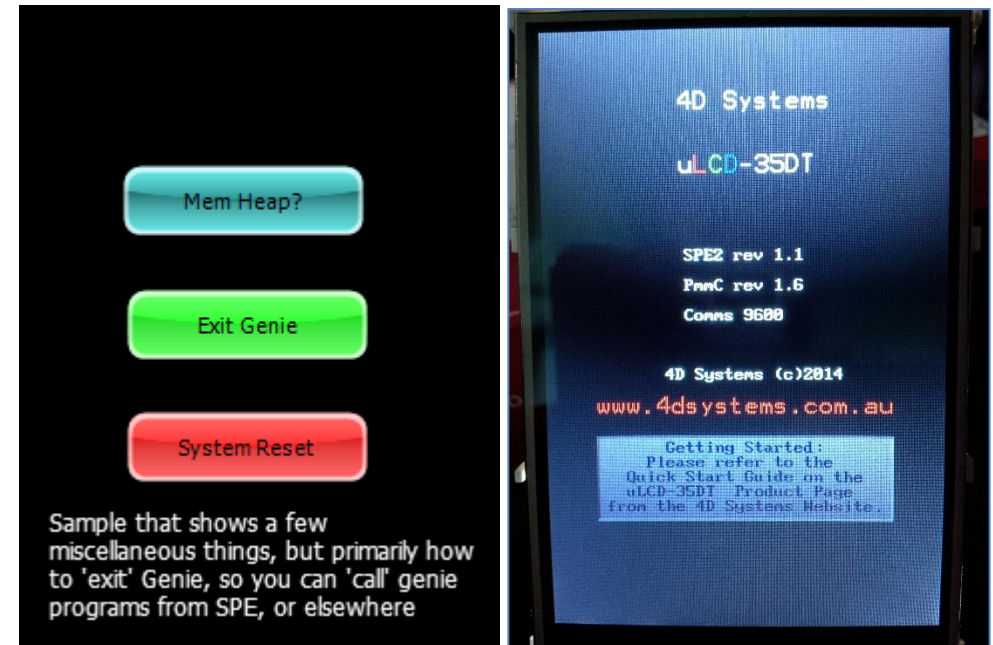
After pressing and releasing **Winbutton1**, the SPE application should now be loaded. Note that there will be a delay.



1. Press and release the Exit Genie button.
2. The SPE application now loads.

Expected Output: Diablo16 Display

After pressing and releasing **Winbutton1**, the SPE application should now be loaded. Note that there will be a delay.



1. Press and release the Exit Genie button.
2. The SPE application now loads.

Generate the 4XE File for the SPE Application

Step 1. Load the SPE application to the display.

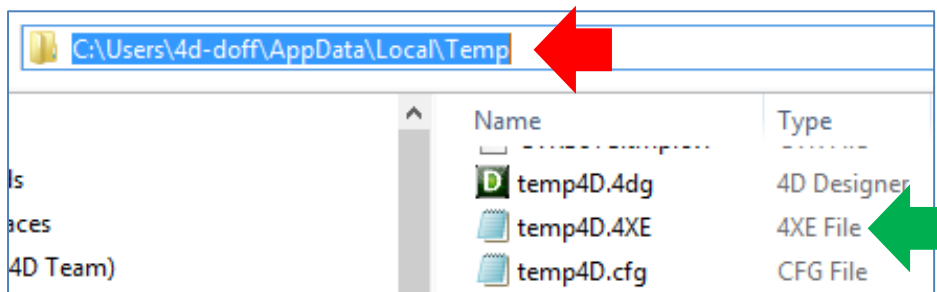
See the application notes for this.

[Serial Picaso Getting Started - The SPE Application](#)

[Serial Diablo16 Getting Started - The SPE Application](#)

Step 2. Locate the 4XE file for the SPE application.

The SPE application will be loaded to the flash memory of the display. However, we are interested with the 4XE file. Workshop places a copy of the 4XE file inside a temporary Windows folder. To illustrate,



The file “**temp4D.4XE**” is the 4XE file of the latest application loaded to the display. Rename the file if needed and copy it to the uSD card. Change the filename in the **MagicRelease** code as well.

Build and Upload the Project

For instructions on how to build and upload a ViSi-Genie project to the target display, please refer to the section “**Build and Upload the Project**” of the application note

[ViSi Genie Getting Started – First Project for Picaso Displays](#) (for Picaso)

or

[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16).

The uLCD-32PTU and/or the uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.

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.