



## ViSi Images and User Images

DOCUMENT DATE: **21<sup>st</sup> JANUARY 2019**  
DOCUMENT REVISION: **1.1**



## Description

This application note is intended to demonstrating to the user the usage of the image and user image widgets in ViSi.

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and other superseded modules which support the ViSi Genie environment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](http://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

<b>Description</b> .....	<b>2</b>
<b>Content</b> .....	<b>3</b>
<b>Application Overview</b> .....	<b>3</b>
<b>Setup Procedure</b> .....	<b>3</b>
<b>Create a New Project</b> .....	<b>3</b>
<b>Design the Project</b> .....	<b>4</b>
<i>The ViSi-based Image and User Image application</i> .....	<b>6</b>
The Include Section .....	<b>7</b>
The Main Program .....	<b>7</b>
The micro-SD Initialization .....	<b>7</b>
Displaying the Objects .....	<b>8</b>
The Repeat-forever Loop .....	<b>8</b>
<b>Run the Program</b> .....	<b>9</b>
<b>Proprietary Information</b> .....	<b>10</b>
<b>Disclaimer of Warranties &amp; Limitation of Liability</b> .....	<b>10</b>

## Application Overview

This application note is intended to demonstrating to the user the usage of the image and user image widgets in ViSi.

## Setup Procedure

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

[ViSi Getting Started - First Project for Picaso and Diablo16](#)

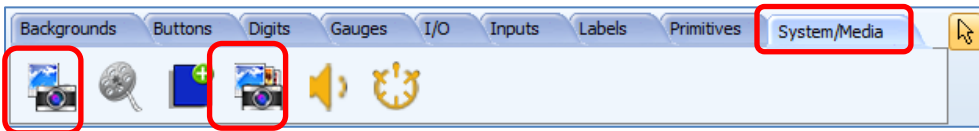
## Create a New Project

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

[ViSi Getting Started - First Project for Picaso and Diablo16](#)

## Design the Project

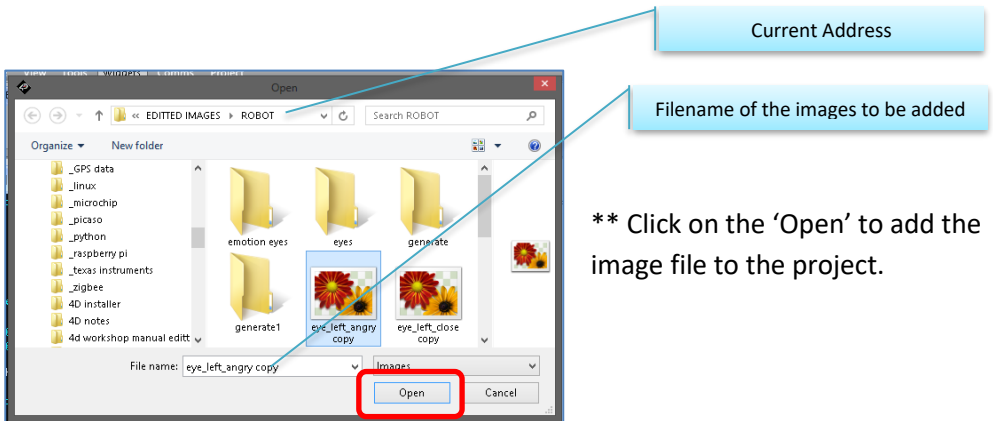
The ViSi environment Digits is a group of customizable widget from the Workshop IDE – ViSi environment. These objects allow the addition of a single images or a group of images. The image and user images widget can be found in the Widgets Menu >> Systems/Media tab.



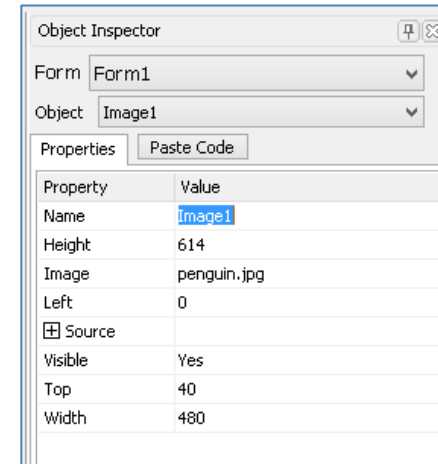
Images can be very useful in different applications like graphic user interfaces. It can also serve as a visual aid for users. Adding images in a project made in the ViSi environment is fairly simple. To add a single image to the project click on the Image objects icon.



After clicking on this icon a pop-up window will appear. This window will ask the user to locate and open the image files that are to be added in the project.

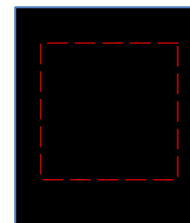
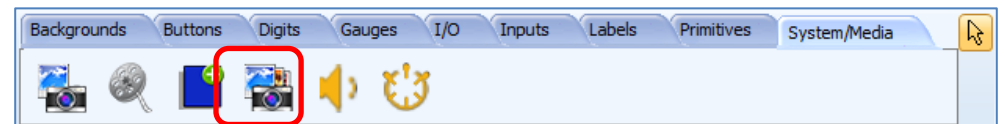


**\*\* Click on the 'Open' to add the image file to the project.**

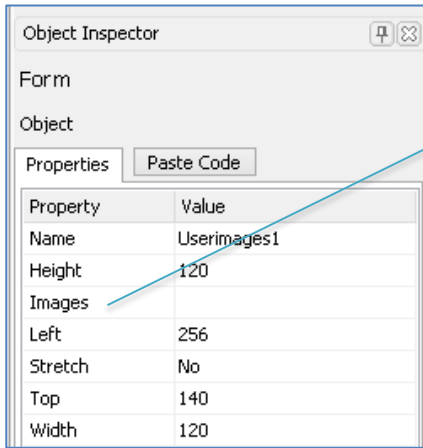


After the addition of the image file, the Object Inspector will display the filename of the image. The image can now be positioned on the display module according to the user's preference.

On the event that a group of images needs to be added to a project, the User Images object is the best choice. The User Images allows addition of several images all at the same time and addition afterwards.

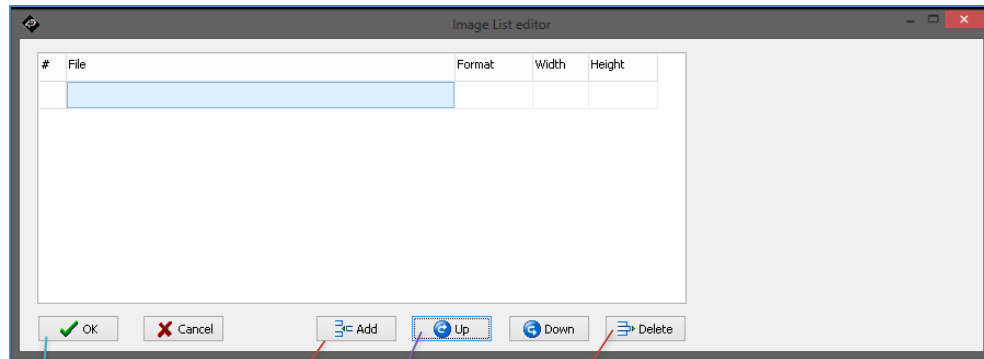


When a User Image object is first added in the project form, a broken line boarder is seen. Adding the images to the object is done is the Object Inspector properties.



Click on the "Images" option on the properties.

Clicking on the 'Images' will open a new window. This new window enables the user to locate the images that are to be added into the project.



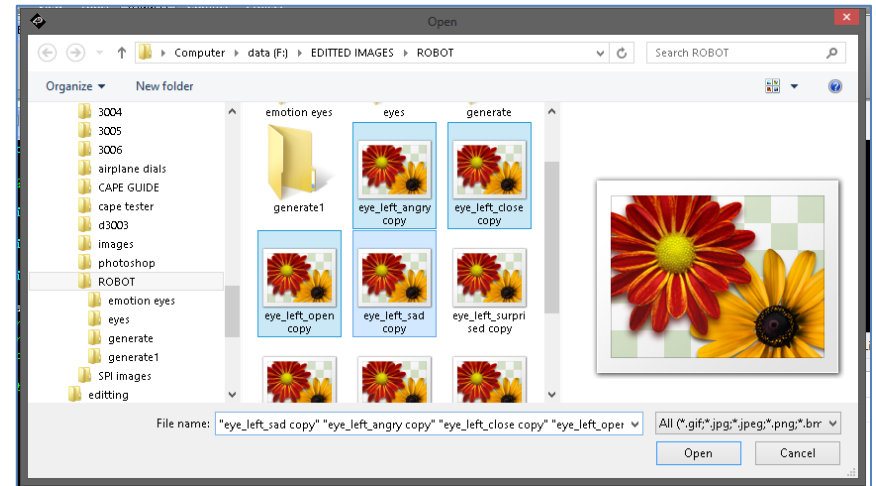
Removes added image(s) from the project

Up and down enables 'frame' number arrangements

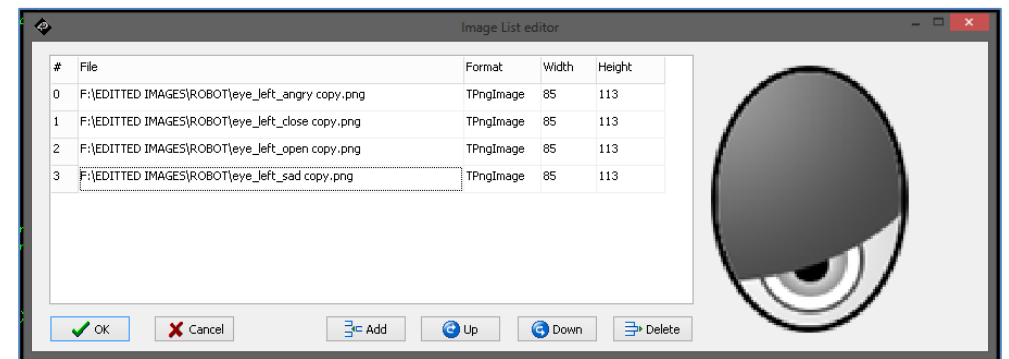
Locate and add image files to the project

Saves the added images to the project

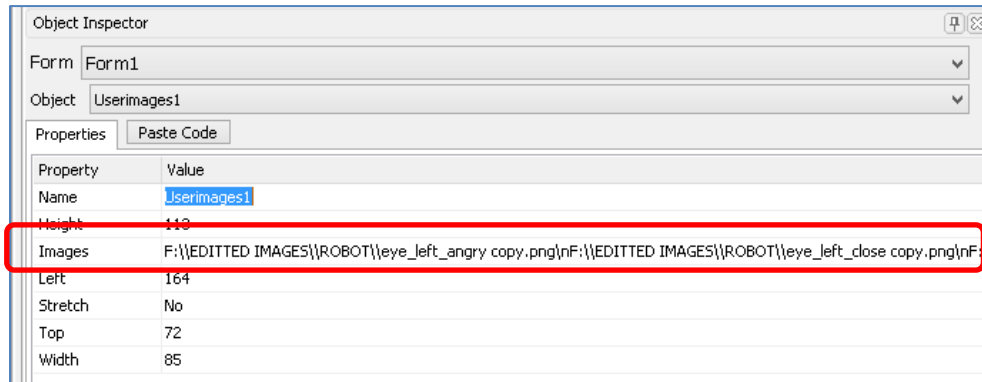
After clicking the 'Add' button, another window will appear enabling the user to locate the images. Several files can be selected at the same time and be added to the project. Clicking open will then result to the enlisting of the image files in the 'Image List Editor' window.



After all the files are added, the images can be arranged using the up/down button. The number on the first column identifies the frame number of the image file. Hitting the 'OK' button finalizes the addition and arranging of images.



Going back to the Object inspector, the once empty 'Images' options will now be populated with the image filename and location similar to the image show below.



Following all the steps discussed herein will have the 'User Image' objects ready for use in the coding area.

### The ViSi-based Image and User Image application

The aim of this application is to demonstrate the manner of how to use the Image and User Images object of the Workshop IDE.

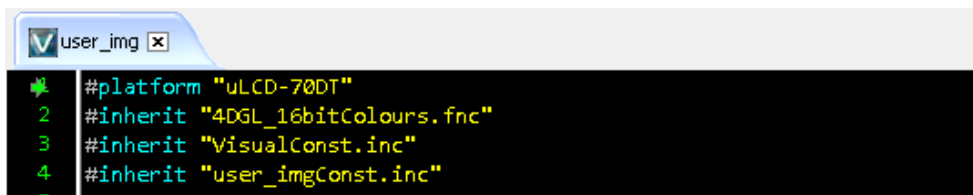


After all the objects have been laid-out, let's continue with the other half which involves the coding of the project. This will be presented in a sectional manner so as not to create confusion with the project. For an in-depth detail of the functions

used in this application note please refer to the [DIABLO16 Internal Functions Reference Manual](#).

### The Include Section

This project starts with the identification of the platform being used as declared by the #platform function. For the program to be able to function properly files are included herein using the #inherit function. Three other files are included automatically during creation of the new project, namely: 4DGL\_16bitColours.fnc, the VisualConst.inc, and user\_imgConst.inc. Notice that the last include file is almost the same as the project filename. This file is automatically generated as soon as project saving is done.



```

user_img x
1 #platform "uLCD-70DT"
2 #inherit "4DGL_16bitColours.fnc"
3 #inherit "VisualConst.inc"
4 #inherit "user_imgConst.inc"

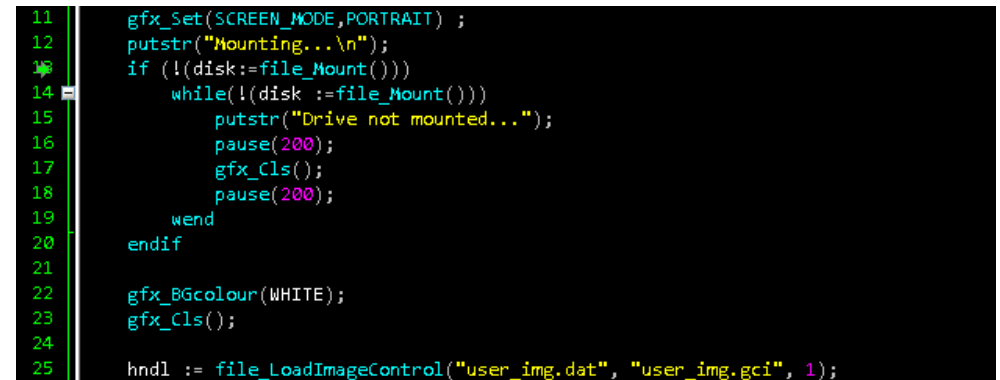
```

### The Main Program

The main program for this projects contains several sections: the initialization and the mounting of the micro-SD card, the initial displaying of the image objects, and a continuous repeat-forever loop. The main aim of the repeat-forever loop is to create an 'animated eye' for the image.

### The micro-SD Initialization

Let's start with the initialization of the uSD card. The uSD card contains all the image information about the objects used in the project. The object information and data are saved under a \*.DAT and a \*.GCI filename extension which is copied to the uSD during project compilation. Mounting of the disk in this application note was done using the following set of program statements.



```

11 gfx_Set(SCREEN_MODE, PORTRAIT) ;
12 putstr("Mounting...\n");
13 if (!disk:=file_Mount())
14     while(!disk:=file_Mount())
15         putstr("Drive not mounted...");
16         pause(200);
17         gfx_Cls();
18         pause(200);
19     wend
20 endif
21
22 gfx_BGcolour(WHITE);
23 gfx_Cls();
24
25 hndl := file_LoadImageControl("user_img.dat", "user_img.gci", 1);

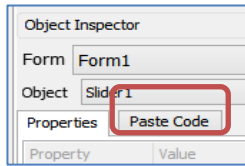
```

When starting a new project in the ViSi environment these set of statements are already included in the coding area. The last part of these set of statements uses a function file\_LoadImageControl() to call on the object data/information files on the uSD drive. This initializes the data to be called in using the variable 'hndl'. In addition, the filenames for the dat and gci files are automatically changed to the project filename as soon as the project is saved.

Added to the statements above are the screen orientation setup function and the touch enable function. The screen orientation is set to portrait thereby providing an 800x480 pixel dimension. This setup is attained using the gfx\_Set() function.

Having been able to load and initialize the uSD drive, the processor is now able to access the information stored therein. As mentioned from the previous section, the filenames with an extension of DAT and GCI has the image data and information. Therefore, the next part of the main program is to display all the objects that were placed on the Workshop IDE form viewer.

A special button from the Object Inspector can help reduce the time of coding of this part. The 'Paste Code' simply pastes object code.



### Displaying the Objects

In this part of the program, the `img_Show()` function simply calls out the object image and information found in the microSD drive. This set of statements call out the images from their handlers and are displayed on screen. Observe the characters with the green font colours. This denotes the available object frame numbers available for the particular User Image object.

```

27  img_Show(hndl,iImage1) ;
28  |
29  img_SetWord(hndl, iUserimages1, IMAGE_INDEX, frame) ; // where frame is 0 to 4
30  img_Show(hndl,iUserimages1) ;
31
32  img_SetWord(hndl, iUserimages2, IMAGE_INDEX, frame) ; // where frame is 0 to 4
33  img_Show(hndl,iUserimages2) ;

```

### The Repeat-forever Loop

This section of the main program contains statements that are continually run by the processor. For this project, a simple change in frame number is made. The change in frame numbers added with instances of `pause()` functions produces a simply user image 'animated' result. This program should result to the penguin image's eyes open/close animation.

```

35  repeat
36
37      frame:=0;
38      img_SetWord(hndl, iUserimages1, IMAGE_INDEX, frame) ; // where frame is 0 to 4
39      img_Show(hndl,iUserimages1) ;
40
41      img_SetWord(hndl, iUserimages2, IMAGE_INDEX, frame) ; // where frame is 0 to 4
42      img_Show(hndl,iUserimages2) ; |
43      pause(1000);
44
45
46      frame:=1;
47      img_SetWord(hndl, iUserimages1, IMAGE_INDEX, frame) ; // where frame is 0 to 4
48      img_Show(hndl,iUserimages1) ;
49
50      img_SetWord(hndl, iUserimages2, IMAGE_INDEX, frame) ; // where frame is 0 to 4
51      img_Show(hndl,iUserimages2) ;
52      pause(100);
53
54  forever

```

There are a total of five different frames added to each of the User Images in this project demonstration. Changing the second frame number between 2 to 4 will yield to different eye animation results.



## Run the Program

For instructions on how to save a **ViSi** project, how to connect the target display to the PC, how to select the program destination (this option is not available for Goldelox displays), and how to compile and download a program, please refer to the section “**Run the Program**” of the application note

### [ViSi Getting Started - First Project for Picaso and Diablo16](#)

The uLCD-32PTU and 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.